MPI IDL

Integrated Design Laboratory

USER'S GUIDE

Table of Contents

Overview

Welcome

MicroPak Industries Inc. is committed to providing useful and innovative solutions for your LCD designs. The Integrated Design Lab (IDL) will help speed and simpify your LCD designs by better managing your source code and testing procedures. This software is an ever evolving work that will be used with all our controller boards starting with the MPIS133 and MPIS133X.

Introduction

The MPI IDL not only works as a terminal emulator so you can test you source code but has a built-in source code generator. This program quickly allows you to build MPIS133 code with point and click menus. Since the MPIS133 code is somewhat encoded in nature the source code builder translates English menu selections into the MPIS133 code to reduce the time needed for coding. In addition the source code builder will reduce errors by reducing the typing that is involved.

The MPI IDL also contains a Bit Map Creator program that will greatly reduce your time in generating user defined bit maps for display on a LCD. It allows you to point and click to create a graphic image or picture that will be converted into the correct MPI code sequence. It can be used with bit maps up to 64x64 bits which is the largest bit map the MPIS133 can handle. The MPIS133 chip can handle up to 16 bit maps of this size.

All command functions are contained within the program including plot pixels, draw horizontal lines, draw vertical lines, draw vectors, draw rectangles and circles, create windows, create graphs and bitmaps, and text functions. The main group headings are located in one pull-down menu with all of its subgroups in the other pull-down menu. Once you have selected a command you can enter the subroutine variables and then create the command line. Since only about 1/3 of the commands have variables associated with them most of the command lines are just selected and enter into your source code with a point and click.

In addition to the code builder there is a text editor so you can modify your code. All of the basic function are built-in including undo, cut, paste, copy along with file manipulation such as open, save and save as. There are also features that allow you to sync the PC with the MPS133 and send your source code to the MPIS133 for testing.

The MPI IDL has a special function built-in. You can format your source code into an assembler table format for insertion into your code or a BASIC string structure. This feature saves time and prevents typographical errors when inserting source code into a larger program. With all of these features within one working environment it will become much simpler to have your project up and running within a few hours.

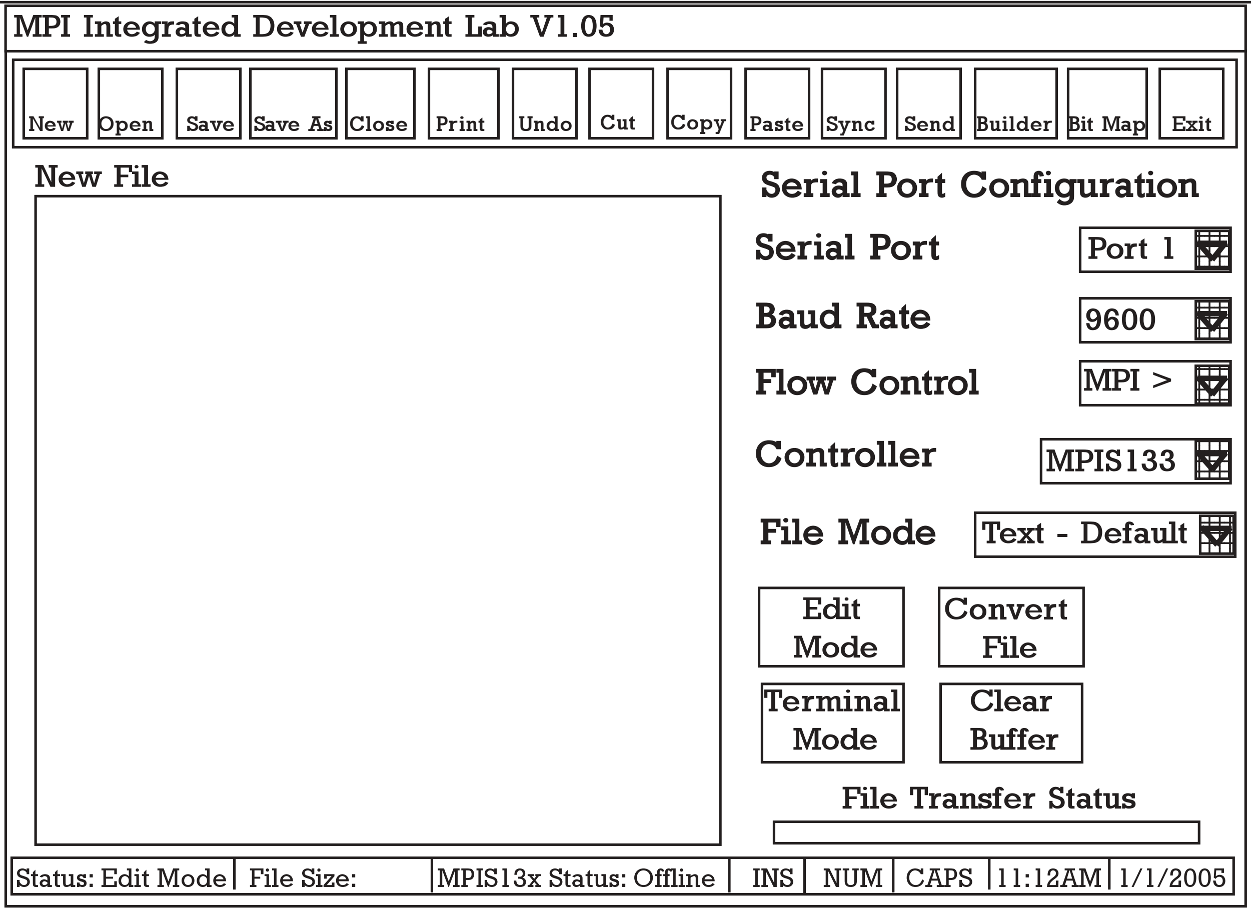
Chapter 1. Basic Functions

Getting Started

The MPI IDL is a Windows based program that runs only in a 32-bit environment under Windows 98/ME/XT/2000 and any of the newer Windows versions. The software in loaded onto your computer through a Setup program that is located on the distribution CD. This program will guide you through the process of moving the program from the CD to your computer. An entry will be placed on you program menu from which you can start the MPI IDL software.

NOTE: When the setup program runs it analyzes the amount of disk space available. The setup program incorrectly reports that there is insufficient disk space. Ignore this warning and install the software. It will install correctly. Once the program is installed select the MPI IDL from your program menu and execute the entry to load the program.

Initial Start-up Screen



MPI Integrated Development Lab V1.05

Serial Port Configuration

Serial Port	Port 1
Baud Rate	9600
Flow Control	MPI >
Controller	MPIS133
File Mode	Text - Default

Status: Edit Mode | File Size: | MPIS13x Status: Offline | INS | NUM | CAPS | 11:12AM | 1/1/2005

Figure 1

ToolBar

The MPI IDL has several features on the screen. At the top is the toolbar. This is a collection of buttons associated with file management, text editing and communications with the MPIS13x series controllers.

The first button will create a New file for your source code. This button will clear the text window and reset your mode to edit. Always use this feature when starting a new session. The next button will Open a file on your disk. It uses the same format as the Windows open file session. When you load a file the name of the file will appear above the text window. You can change the disk or folder and select a file you want to edit or transmit to the MPIS133. The Save button automatically saves the file your are currently working on. You must save the file before you transmit the file to the MPIS133. The Save As ... button allows you to save a new file with a new name or save a current file under a new name. This is exactly the same feature you have on your Windows based system. The Close button will close the current file and create a New session. You will be asked to save you work if it has not already been saved. The Print button will send the current document to your printer.

The next 4 buttons are used to edit you source code that appears in the text window. These features work the same as any standard text editor. You can Cut, Copy, Paste and Undo your last cut. The Sync button will sync the PC serial port with the MPIS133. You should use this feature before sending any data to the MPIS133 to make sure it is properly connected to the serial port. The Send button will send the source code in the text window to the MPIS133. This allows you to test your source code. Transmission to the MPIS133 is based upon the settings in the Serial Port Configuration area to the right of the text window. In addition the File Transfer Status progress bar indicates the % of completion of the file transfer.

The Builder button opens a new window that is used to simplify source code building. This feature will allow you to point and click to build you source code without worrying about where each parameter goes in the command line. This will greatly reduce the time necessary to produce your code and reduce typographical errors. See the Builder section for more information. The Bitmap button opens a new window to the Bit Map Creator program. This program will allow you to create user defined bit maps in image or picture form and the program wil convert the picture to the correct code sequence. The Exit button will close the program and return you to the desktop.

Serial Port Configuration

The Serial Port Configuration section is used to setup your communications port with the MPIS133. Each section uses a pull-down list box to select the setting. You can select between serial port 1 to serial port 4 on your PC. You have the option of selecting 9600 or 19200 for baud rate.

You can select the handshaking mode for correct communications with the MPIS133. The choices depend you the controller selection. The standard setting is "MPI >". This will work with all of MPI's controllers. You should select the controller that you are using as the Builder program will vary its contents depending on the controller. Different controllers have different commands and the Builder needs to know which controller you are using to properly create the source code. Not all controllers are currently available as some are still in testing.

Convert File Button

The file mode selects the format of the source code. This list box is used when converting from text to assembler or BASIC format. Normally the mode is in text - default mode. If you want to convert to a different format for use in your program then first select the mode and then convert the file by pressing the Convert Button. Once the conversion is done the new file will be save with the proper file type (.asm or .bas). You can edit the new file of copy and paste it into you regular program source code.

Edit Mode/Terminal Mode Buttons

The Edit Mode button places the system in edit mode. This meas the text window is used for creating source code. The terminal mode button places the text window in terminal emulation mode. In this mode you can send codes directly to the MPIS133 just as if you where using HyperTerminal. In this mode none of the texted is store and you can use the Clear Buffer button to clear the text window at any time. When in the terminal mode the information is still saved from the edit mode and you can switch back and forth between the two modes. The heading at the top of the text window indicates which mode you are in along with the status bar at the bottom of the program window.

Status Bar

The Status Bar provides information about the mode in are in either edit or terminal; the size of the file in bytes in the text window; the status of the MPIS13x (Offline, Ready, Not Responding); the status of the insert, numeric, and caps keys; along with the current time and date.

Chapter 2: Code Builder

Overview

Code Builder is a program that will simplify your source code writing. Since MPIS133 code is a command line format with embedded variables in the line it may sometimes become confusing as to the proper position of the variable within the line. Code Builder will eliminate this problem. In addition you will select the variables from pull-down list boxes in plain English. This greatly increasing the accuracy of the code by eliminating typographical errors. The screen below is an example of the Builder screen.

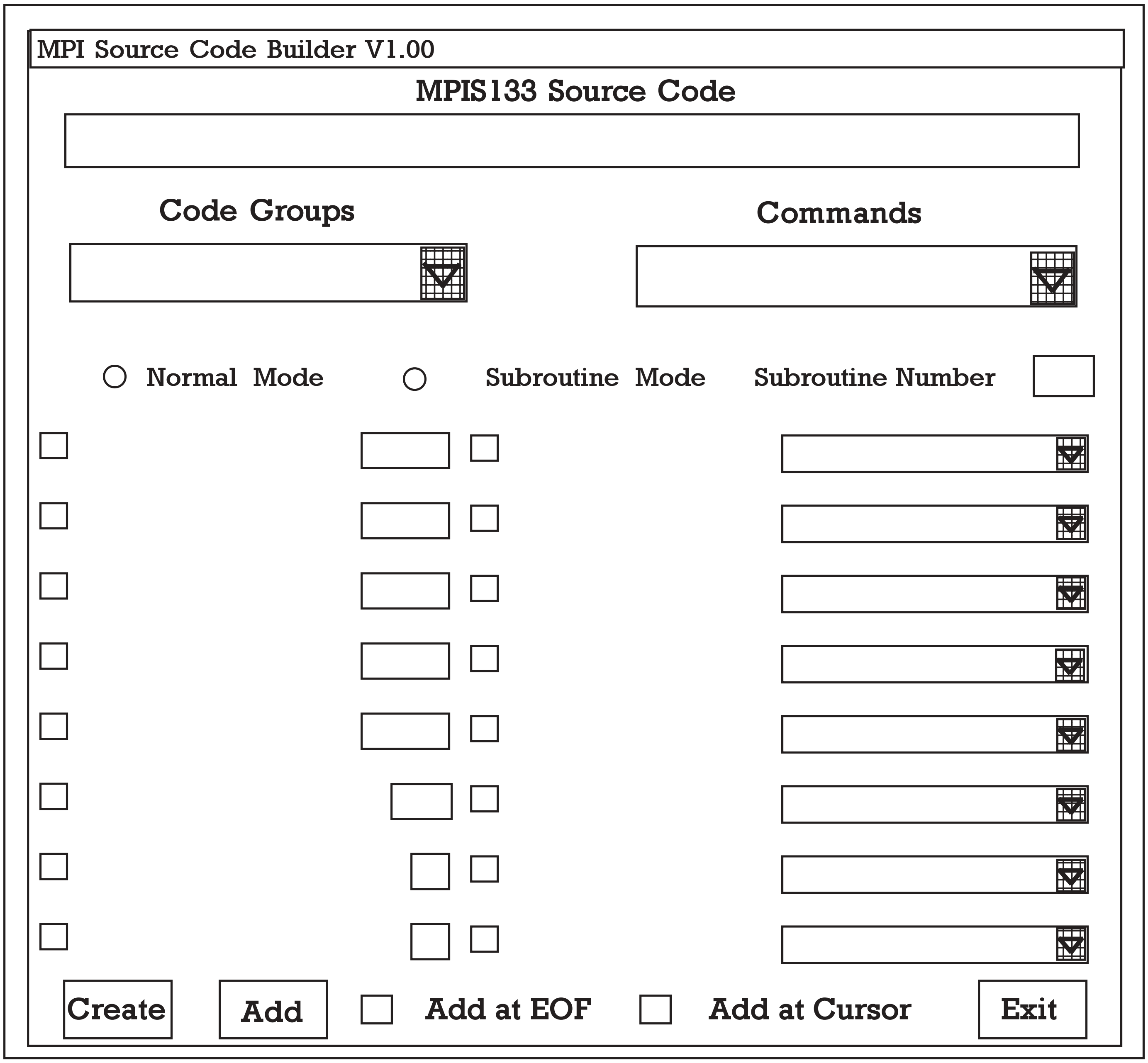


Figure 2

Code Builder is designed to work with a point and click method. Almost 2/3 of the commands are single commands without any variables. This means you can select the command and place it directly into your source code. The builder is divided into sections. The top section is the source code line itself. Next are the code groups and commands. The code groups are divided into sections such as cursor control, graphics, text, etc. Once you have selected a code group the commands available in that group will be listed in the command list box. If the command does not require any variables then you can add the command directly to the text window with the Add button.

Under these list boxes are the buttons for Normal Mode, Subroutine Mode, and Subroutine Number. In the Normal mode the code is added just as it looks in the code line. In the Subroutine Mode the prefix SLxxx is added to the code with the xxx being the Subroutine Number. That means you must set the subroutine number and click the subroutine button to generate code that will be placed in memory as a subroutine to be executed later. In this mode every instruction will receive the prefix. To exit this mode you must click on the Normal Mode button.

Next are the variable input sections. The small boxes in front of each input section or list box will be checked if the variable is required. The section on the left will require a text or number input by the user. The section on the right is composed of list boxes that you pull-down to select your entry. Once you have filled in the section then you must click on the Create button to load your selection into the command line. You will see the command line update after you click the button. If the command line is correct then you can add your code by clicking on the Add button. If the data is incorrect you can edit any of the variables and click on the Create button again to make your changes.

You must create the code before adding it to your listing otherwise you will place a code segment with letters instead of number in the source listing. The bottom section has 2 buttons for placing or adding your command line at the End of File or at the current Cursor Location. This will allow you to correct or insert code when you need to make changes.

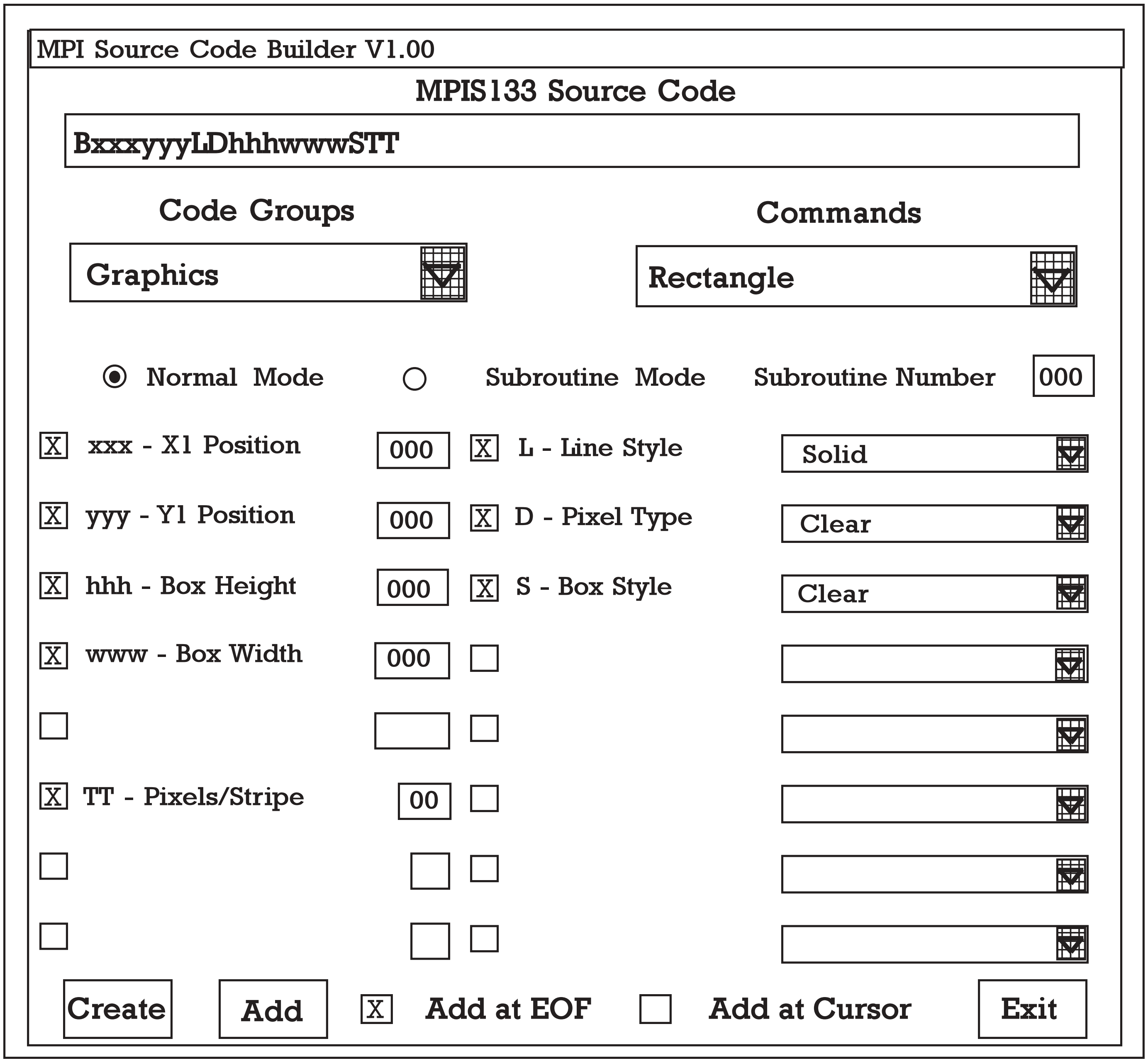
Some commands such as text commands require you to type in the text. In these entries you can select the direction and font attribute from the variable section but then you have to place the cursor in the command line text box and add your text. You only have to do this with 3 text commands and the user bitmap commands.

While you are in the builder mode you can switch back and forth between the text window and the builder. This will allow you to test the code as you write. After you have written a certain amount of code then switch to the text window by clicking on the window and save the source code. You can then send the code to the MPIS133 with the Send button and test your code. You can make corrections directly in the text window or use the Builder to add or correct code.

If you are writing subroutines you can download the subroutines with the Send button. After you have downloaded the code you can switch to the Terminal Mode and type SXaaa with aaa = to the subroutine number and test the subroutine. You can switch back and forth between the edit mode and terminal mode and make corrections as needed. As you can see with the integrated design lab you will greatly increase your productivity and reduce the time necessary finish and test your code.

Code Builder Example

The following is an example of what the screen will look like and the entries necessary when creating the command for rectangle.



MPI Source Code Builder V1.00

MPIS133 Source Code

BxxxyyyLDhhhwwwSTT

Code Groups: Graphics

Commands: Rectangle

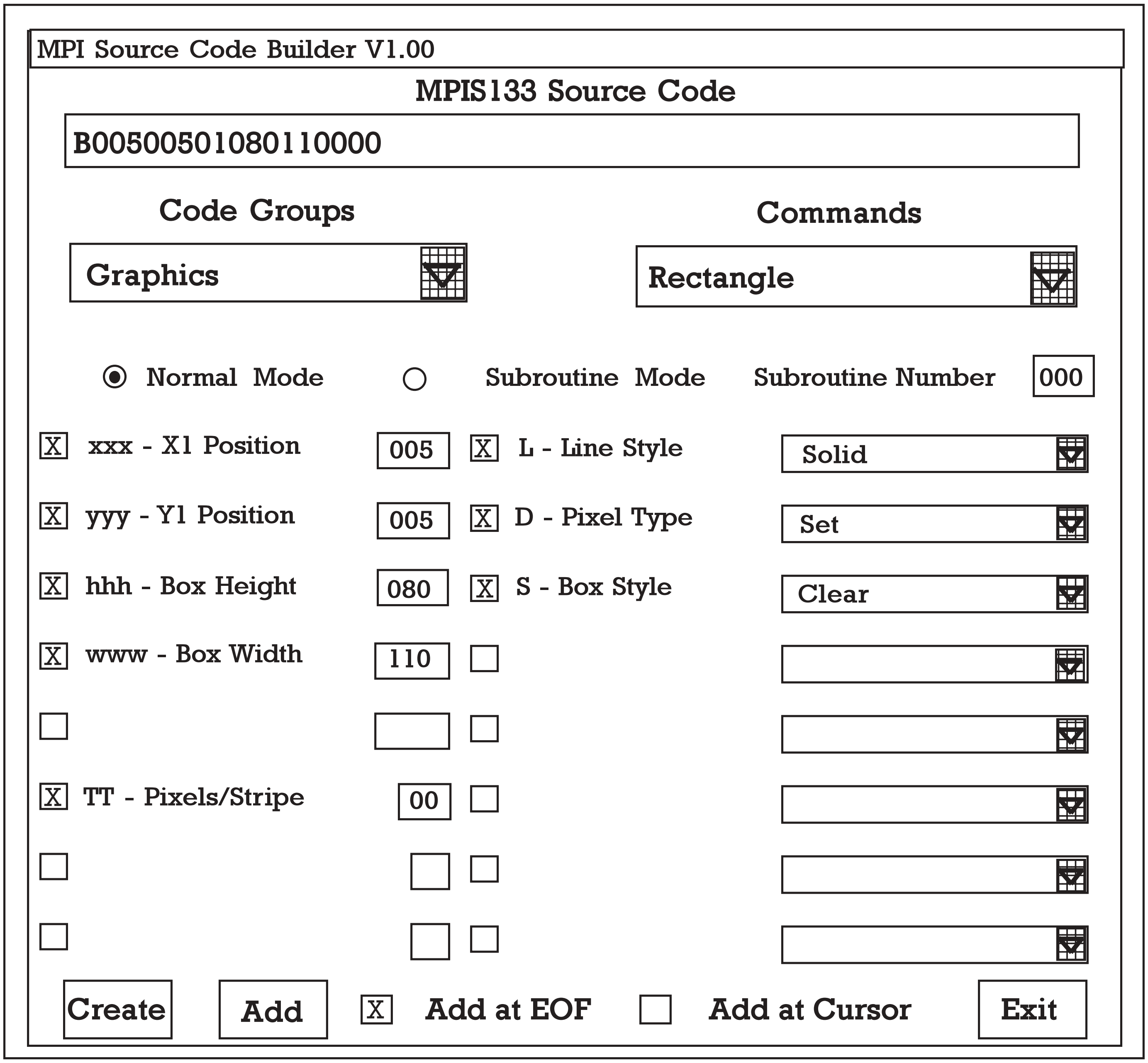
[✓ Normal Mode] Subroutine Mode

Subroutine Number: 000

X	xxx - X1 Position	000	X	L - Line Style	Solid
X	yyy - Y1 Position	000	X	D - Pixel Type	Clear
X	hhh - Box Height	000	X	S - Box Style	Clear
X	www - Box Width	000			
X	TT - Pixels/Stripe	00			

[✓ Add at EOF] Add at Cursor

When you select the Graphics code group and Rectangle in the Command list box then the generic form of the command will appear in the Command line text box along with the variables and their names in the variable sections. Notice that all of the input variables are zeroed. To enter data click on the text box or highlight the text and enter your new data. It is very important to maintain the correct field length. If the data field requires 3 digits you must enter 3 digits even with leading zeros.


MPI Source Code Builder V1.00

MPIS133 Source Code

B0050050108011 0000

Code Groups	Commands
Graphics	Rectangle

(● Normal Mode) ○ Subroutine Mode — Subroutine Number: 000

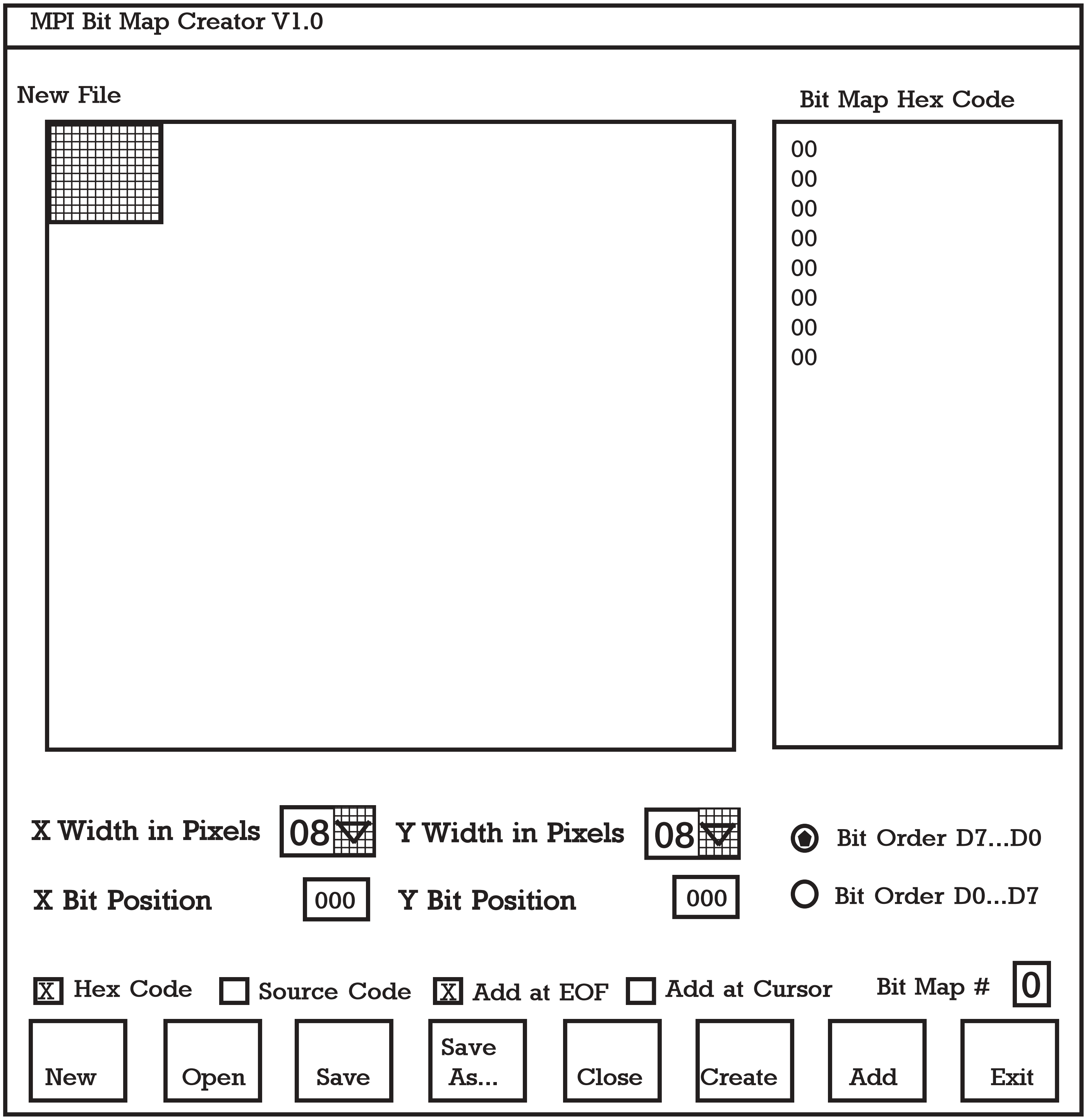
[X] xxx - X1 Position	005	[X] L - Line Style	Solid
[X] yyy - Y1 Position	005	[X] D - Pixel Type	Set
[X] hhh - Box Height	080	[X] S - Box Style	Clear
[X] www - Box Width	110	[ ]	
[ ]		[ ]	
[X] TT - Pixels/Stripe	00	[ ]	
[ ]		[ ]	
[ ]		[ ]	

To change the Line Style use the list box to select your entry. Do the same for Pixel Type and Box Style. If the entry is the one you want you do not have to make any changes. When you are satisfied with you data click on the Create Button to load the command line. After you create the command line click on the Add button to move the command line to your source code text window. You can continue with this process until you have finished your code.

You can switch to the text window at any time, save your source code and send it to the MPIS133 to test your progress. For commands without any variables you just have to Add the command line to your source code with the Add button.

Bit Map Creator

The Bit Map Creator is used to create bit maps or icons for use in your programs. The keypads in the MPIS133 code can use a 24x16 bitmap or icon in the center of the keypad to help explain its function. You can also create any image you may need up to 64x64 bits. Since you can create up to 16 user defined bit maps you can paste these together on the screen to create any size image you may need up to 256x256 bits. The screen below is an example of the Bit Map Creator when you start the program by pressing the Bit Map button.



MPI Bit Map Creator V1.0

New File

Bit Map Hex Code

```
00
00
00
00
00
00
00
00
```

X Width in Pixels: 08
Y Width in Pixels: 08
[✓ Bit Order D7...D0] Bit Order D0...D7
X Bit Position: 000
Y Bit Position: 000
[✓ Hex Code] Source Code [✓ Add at EOF] Add at Cursor
Bit Map #: 0

The Bit Map Creator is divided into sections. The left hand graphic area is the bit map picture. This is the area where you create your bit map. To the right of this picture is the Hex Code/Source Code listing. This listing changes as you turn on and turn off individual pixels to create your bit map. Above the bit map picture is the name of the file where you have stored the bit map. On a new project the name will be new file.

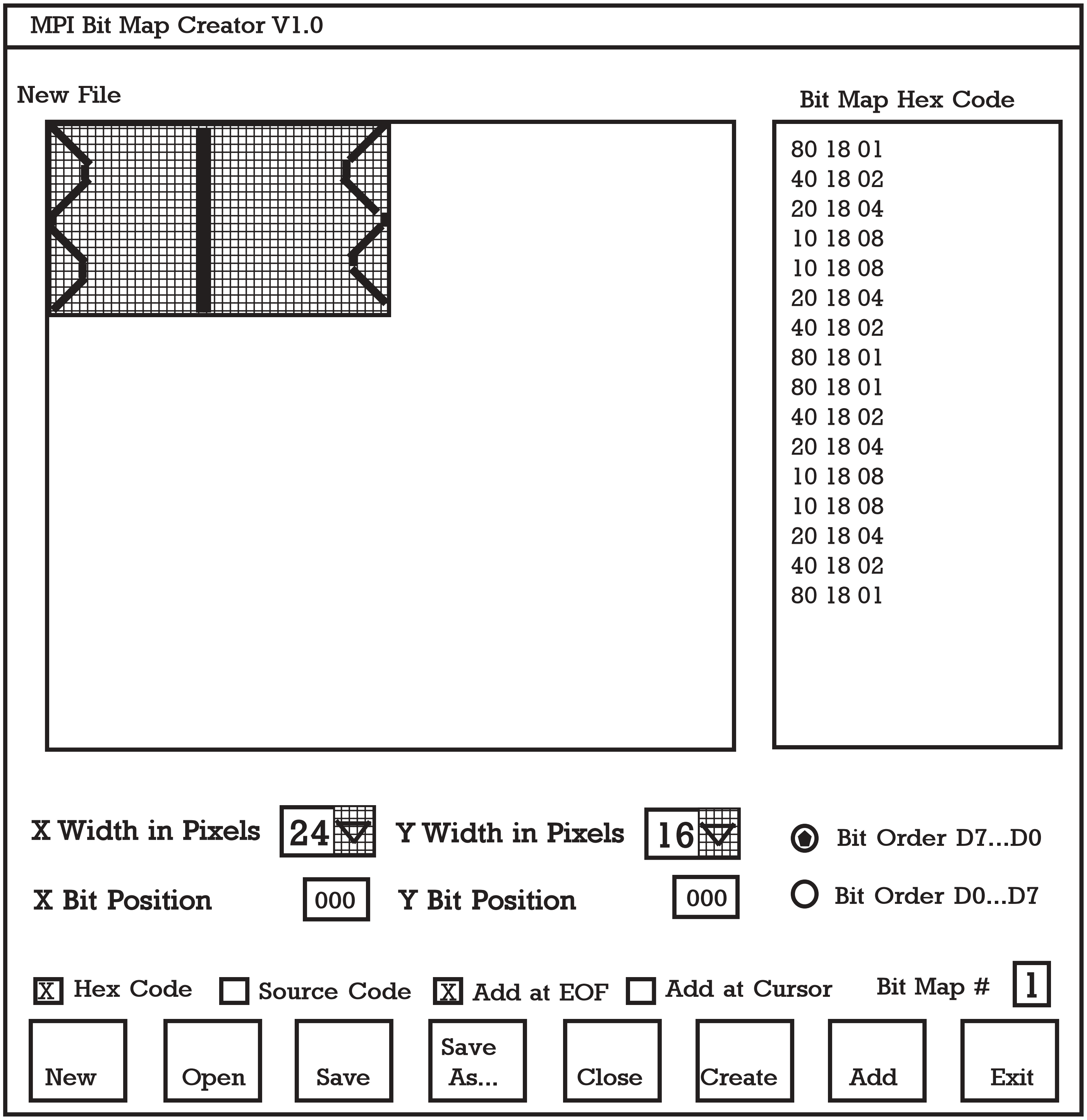
Under the bit map picture are two list boxes where you set the size of the bit map. The range of the x and y pixels are from 8 to 64. As you change these values the picture will resize and clear any work you may have done. It is important to store your work before you change these settings. Under the list boxes are the actual pixel locations within the bitmap. These numbers change as the cursor moves over the bitmap. The option buttons under the text box are for setting the bit order. You can select bit D7 through D0 which is the default setting and the setting the SED1335 chip uses when writing to the LCD. You can also select D0 through D7. This setting will be used when we support different controllers in the future.

The check boxes are next and they select either hex code or source code that will be displayed in the text box. Source code will not display until it has been created. The next two boxes determine where the source code will be added in the main text window of the MPI IDL program. Just like the code builder program you can add text at the end of file or at the current cursor position. The last text box lets you select the bit map number assigned to the current bit map you are working on. Just type in a number between 0 and F. This allows up to 16 bit maps to be stored on the MPIS133 system.

The buttons at the bottom of the window are for file managment and bit map creation. The first button starts a new session clearing the bit map window and resetting everything to zero and 8x8 bits in size. The second button will open a (.mbm) file on your disk which is the format that MPI bit maps are store under. The third button will save the current bit map to disk. The fourth button will save a bit map under the name that you choose and its location on the disk. The fifth button closes the current file and resets the system for a new session. The next button is used to create the source code when you have finished with your bit map picture. If you picture is correct and the source code has been created then you can use the next button to add the source code to you main code. The last button will exit the system and unload the bit map program.

Bit Map Creator Example

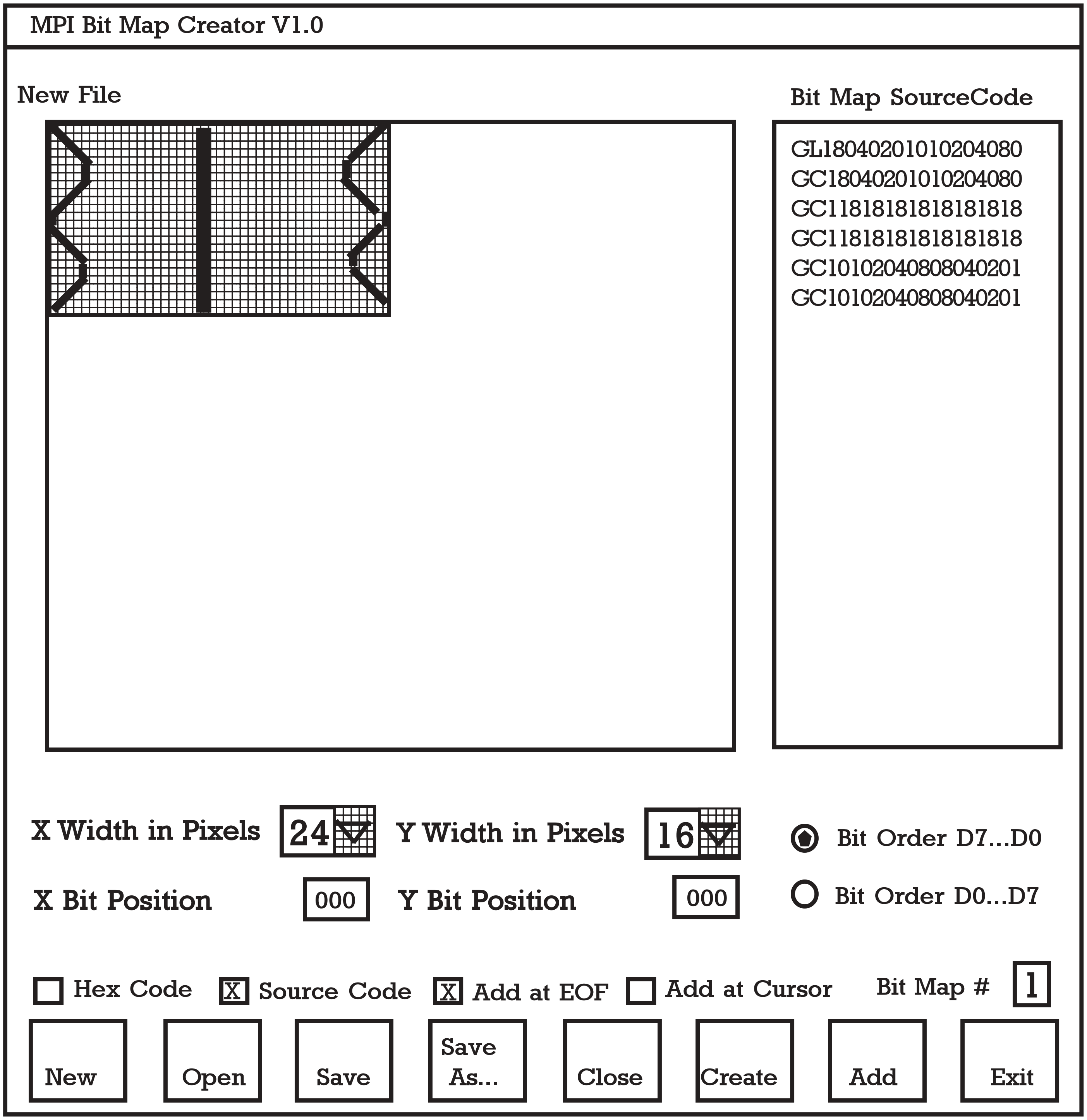
The following is an example of how to use the bit map creator. Once you have loaded the creator select the size of the bit map. Select the bit map number you plan to assign to the bit map. Use your cursor and click on the pixels to highlight or turn them on. If you click on the pixel again it will turn off. Continue until you are satisfied with the bit map. Once you have finished save your work with the Save As... button. Then press the Create button to create the source code. Select the area where the new code will be added and press the Add button. You may use the Code Builder program to allocate the memory area with the proper source code. Add that code then switch to the bit map creator and add the bit map code. Finally switch back to the code builder to continue with your program. The following figure is an example of what the screen may look like.



```
80 18 01
40 18 02
20 18 04
10 18 08
10 18 08
20 18 04
40 18 02
80 18 01
80 18 01
40 18 02
20 18 04
10 18 08
10 18 08
20 18 04
40 18 02
80 18 01
```

X Width in Pixels: 24
Y Width in Pixels: 16
X Bit Position: 000
Y Bit Position: 000
[● Bit Order D7...D0] ○ Bit Order D0...D7
[✓ Hex Code] ☐ Source Code [✓ Add at EOF] ☐ Add at Cursor
Bit Map #: 1

Once the file has been saved you can create the source code by pressing the Create button. When the Create button is press the Source Code check box is highlighted and the label on the text box changes to let you know what is in the text box. Any time you do work on the bitmap the text box will automatically change back to the hex code display. The next page shows what a screen looks like after the Create button is pressed..


MPI Bit Map Creator V1.0

New File

Bit Map SourceCode

```
GL1804020101020408O
GC1804020101020408O
GC1181818181818181818
GC1181818181818181818
GC1010204080804020l
GC1010204080804020l
```

X Width in Pixels 24

Y Width in Pixels 16

[✓ Bit Order D7...D0]

Bit Order D0...D7

X Bit Position 000

Y Bit Position 000

Hex Code [✓ Source Code] [✓ Add at EOF] Add at Cursor

Bit Map # 1

Now that the source code has been created you can add it to you program with the Add button. Be sure to select the proper check box for EOF or current cursor position before you press the Add button. You can always edit the source code and redo your addition. Once you are finished you can Exit the program and continue with your source coding building.

The following pages contain a quick reference guide to the commands. It is the same as is in the MPIS133 manual.

Quick Reference

Command	Name	Description
AxxxyyyLDhhhwwwGTT	Graph X,Y Axis	Starting Location upper left hand corner: xxx=<000:639>, yyy=<000:255> Grid Style: L=<0:7> 0=No Grid 1=Dotted/q2 2=Dotted/q3 3=Dotted/q4 4=Dashed/q4 5=Dashed/Dotted/q4 6=Dashed/q5 7=Dashed/q7 Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR Y-Axis Length: hhh=<000:255> X-Axis Length; www=<000:639> Graph Type: G=<0:2> 0= 1st Quadrant 1= 1st/4th Quad. 2= 1/2/3/4 Quad # of Pixels/Tic: TT=<00-99> Default: 10 pixs/Tic
BxxxyyyLDhhhwwwSTT	Rectangle	Starting Location upper left hand corner: xxx=<000:639>, yyy=<000:255> Line Style: L=<0:7> 0=Solid 1=Dotted/q2 2=Dotted/q3 3=Dotted/q4 4=Dashed/q4 5=Dashed/Dotted/q4 6=Dashed/q5 7=Dashed/q7 Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR

Quick Reference

Command	Name	Description
BxxxyyyLDhhhwwwSTT	Rectangle Continued	Rectangle Height: hhh=<000:255> Rectangle Width; www=<000:639> Box Style: S=<0:7> 0= Clear 1= Solid 2= Vertical Stripes 3= Horizontal Stripes 4= Slanted Stripes 5= Erase Area 6=Clear Inside Only 7=Double Line Border # of Pixels between Stripes: TT=<00-99> Default: 3 pixs/Stripe
C+	Cursor On	Turns cursor on, on layer 1
C-	Cursor Off	Turns cursor off, on layer 1
CA	Cursor UnderLine	Changes cursor to underline style on layer 1
CB	Cursor Block	Changes cursor to block style on layer 1
CH	Cursor Home	Move cursor to home position at Column 0, Row 0 on layer 1
CR	Cursor Right	Moves cursor right 1 column on layer 1
CL	Cursor Left	Moves cursor left 1 column on layer 1
CU	Cursor Up	Moves cursor up 1 row on layer 1
CD	Cursor Down	Moves cursor down 1 row on layer 1
C0	Cursor Blinking Off	Turns cursor blinking off
C1	Cursor Blink 1 Htz	Blinks current cursor style at 1 Htz
C2	Cursor Blink 2 Htz	Blinks current cursor style a 2 Htz
CS	Clear Screen	Clears the screen on layer 1 and 2
CC	Clear Screen	Clears the screen on layer 1 only

Quick Reference

Command	Name	Description
CE	Clear Screen	Clear screen from current character position to end of screen on layer 1. Use command L to set current character position.
CG	Clear Screen	Clear screen layer 2 only
CT	Clear Screen	Clear screen from current character position to top of screen on layer 1. Use command L to set current character position.
CxxxyyyLDvvvhhhFTWABC	Define Clock/Timer	Defines Real Time Clock or Timer based on CK2/CK3 command Starting Location upper left hand corner of clock: xxx=<000:639>, yyy=<000:255> Time Format: L=<0:3> 0=HH:MM 12 hour clock 1=HH:MM 24 hour clock 2=HH:MM:SS 12 hr clock 3=HH:MM:SS 24 hr clock Text Attribute: D=<0:8> Same as the b command in the text command S Starting Location upper left hand corner of the date display: vvv=<000:255>, hhh=<000:639> Font Size: F=<0:3> 0= 5x7 1= 7x9 2= 11x13 3= 15x17 Date Display T=<0-1> 0=Display Off 1= Display On Date Format: W=<0-1> 0=MM/DD/YYYY 1=MM/DD/YY Time Separator: A=<Ascii> Date Separator: B=<Ascii> European Format C=<0,1> 0=European format off 1=European format on
CKUxxxyyyT	Define Day of the Week	Sets the Day of the Week Location on the screen and turns the day of the week display on/off. xxx=<000:639>, yyy=<000:255> Day of the Week Display T=<0-1> 0=Display Off 1= Display On
CKSThhmmssd	Set Clock/Timer	Sets the Time and Day of the Week hh<00-23>=Hours mm<00-59>=Minutes ss<00-59>=Seconds d=Day of the Week <0-6> Sun-Sat Note: the Time is set as a 24 hour clock 00-23:59:59. The time is then adjusted according to the format specified.

Quick Reference

Command	Name	Description
CKSDmmddyyyy	Set Date	Sets the Date in the following format mm<01-12>=Month dd<01-31>=Day yyyy<2002- xxxx>=Year
CKSXnn	Set Subroutine # on Timer Alarm	Sets the Sub # nn <00-63> that is executed when to timer reaches 0 in countdown mode
CKRT	Read Clock	Sends current time to the host over the RS-232 Serial Port
CKRD	Read Date	Sends current date to the host over the RS-232 Serial Port
CK0	Suspend Clock Display	Stops the update of the clock on the screen.
CK1	Resume Clock Display	Resumes updating clock display
CK2	Set as Clock	Sets the clock to function as a clock. (Default condition)
CK3	Set as Timer	Sets the clock to function as a timer.
CK4	Update Every Minute	Updates screen display every minute. (Default condition)
CK5	Update Every Second	Updates screen display every second. This command must be used when the clock is in the timer mode.
CK6	Start Timer	Starts Timer
CK7	Stop Timer	Stops Timer
CK8	Reset Timer	Reset s Timer to value defined in the Set time mode.
CKT	Display Time	Displays the clock on the screen as defined by the CxxxyyyLDvvvhhhFTW command.
CKD	Display Date	Displays the date on the screen as defined by the CxxxyyyLDvvvhhhFTW command.

Quick Reference

Command	Name	Description
CKC	Display Day of the Week	Displays the day of the week using the current cursor location and font designation. The following commands update clock/timer registers based on CK2/CK3 commands
CKH+	Increment Hours by 1	Increment the hours register by 1
CKH-	Decrement Hours by 1	Decrement the hours register by 1
CKM+	Increment Minutes by 1	Increment the minutes register by 1
CKM-	Decrement Minutes by 1	Decrement the minutes register by 1
CKX+	Increment Seconds by 1	Increment the seconds register by 1
CKX-	Decrement Seconds by 1	Decrement the Seconds register by 1
CKW+	Increment Day of Week by 1	Increment the day of week register by 1
CKW-	Decrement Day of Week by 1	Decrement the day of week register by 1
CKN+	Increment Month by 1	Increment the month register by 1
CKN-	Decrement Month by 1	Decrement the month register by 1
CKA+	Increment Day of Month by 1	Increment the day of month register by 1
CKA-	Decrement Day of Month by 1	Decrement the day of month register by 1
CKY+	Increment Year by 1	Increment the year register by 1
CKY-	Decrement Year by 1	Decrement the year register by 1
CK+	Set Timer to count up	Timer mode set to stopwatch
CK-	Set Timer to count down	Timer mode set to countdown timer

Quick Reference

Command	Name	Description
CK#	Suspend Timer Display	Stops the update of the timer on the screen
CK*	Resume Timer Display	Resumes the update of the timer on the screen.
CK@	Set Timer Alarm	Turns the Timer Alarm on
CK%	Clear Timer Alarm	Turns the Timer Alarm off
D+	Display On	Turns the display on.
D-	Display Off	Turns the display off.
D2	Or Layer 1/2	Simple overlay 1 or 2
D3	Xor Layer 1/2	Reverse overlay (1 xor 2)
E0-	Echo Off	Turns Serial Port Character Echo off.
E0+	Echo On	Turns Serial Port Character Echo on.
E1-	Layer 1 Off	Turns layer 1 off.
E1+	Layer 1 On	Turns layer 1 on.
E1*	Layer 1 Blink 2 Htz	Blinks layer 1 at 2 Htz
E1%	Layer 1 Blink 16 Htz	Blinks layer 1 at 16 Htz. This creates a half-tone effect.
E2-	Layer 2 Off	Turns layer 2 off.
E2+	Layer 2 On	Turns layer 2 on.
E2*	Layer 2 Blink 2 Htz	Blinks layer 2 at 2 Htz
E2%	Layer 2 Blink 16 Htz	Blinks layer 2 at 16 Htz. This creates a half-tone effect.
Fa	Select Font	Font Number: a=<0:3> 0= 5x7 2=11x13 1= 7x9 3=15x17 Font select applies to layer 2 only. Layer 1 displays the built- in font which is 5x7.

Quick Reference

Command	Name	Description
GxxxyyyQn	Bitmap	Size of Bitmap in bytes - 512 bytes/bitmap maximum: xxx=<000:aaa>, yyy=<000:bbb> aaa*bbb must be <=512 bytes - i.e. 64x64 bits Bitmap Action: Q=<A, D, R> A= allocate bitmap memory based on xxx*yyy=# of bytes D= Display bitmap at location xxx, yyy. R=release bitmap memory based on xxx*yyy=# of bytes Bitmap assignment #: n=<0:F> n= reference number that defines the bitmap in hexidecimal. There are 16 user defined bitmaps.
Gcnhhhhhhhhhhhhhhhh	Load Bitmap	Bitmap Action: C=<L,C> L=Load bitmap to bitmap memory location based on n. This comand loads the first 8 bytes into bitmap memory. C=Continue loading bitmap memory. This command is to load the remaining bytes to memory. The C command is used as many times as neces sary to complete the memory loading process. Bitmap assignment #: n=<0:7> n= reference number that defines the bitmap Data: <00:FF> Up to 20 bytes of hexidecimal bitmap data. Note: Data is loaded and read from memory by columns. Their entire first column of yyy length is loaded before the second column and this continues until all columns are loaded.

Quick Reference

Command	Name	Description
HxxxyyyLDdddwww	Horizontal Line	Starting Location of line: xxx=<000:639>, yyy=<000:255> Line Style: L=<0:7> 0=Solid 1=Dotted/q2 2=Dotted/q3 3=Dotted/q4 4=Dashed/q4 5=Dashed/Dotted/q4 6=Dashed/q5 7=Dashed/q7 Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR Dummy Variable: ddd=<000> Line Length; www=<000:639>
Ia	Backlight Timer	Timer Interval: a=<0:9> 0=Timer Off 1-9= # of 30 Second Intervals the light will be on before it is turned off. Time: 30 sec - 4 min 30 sec
JA	Select KeyPad	Selects keypad 1x8 input on the 8 bit port Each bit is treated as a single input key.
JB	Select IR Touchscreen	Selects the input port as a 8-bit port. This will read the port as a single byte. It is used with our IR touchscreen and IR touchscreen con troller.
JC	Select RTS 8bit Mode	Selects the input device as the RTS controller and sets the controller to the 8-bit mode.
JD	Select RTS 12 bit Mode	Selects the input device as the RTS controller and sets the controller to the 12-bit mode. This is the default mode.
JE	Select RTS KeyPad Mode	Selects the input device as the RTS controller and sets the controller to the 8 bit keypad mode. This mode overlays a 10x8 keypad matrix on the display. This will provide you will 80 key pad inputs on any size screen. You must use the JXaaabbb and JYaaabbb com mands to calibrate your touchscreen before using this mode. This mode is available in 8 bit resolution only. It is used with the Key events commands to call subroutines when a key is pressed or released.

Quick Reference

Command	Name	Description
JF	Read and Xmit RTS Data	Reads and transmits data from the RTS.
JG	Pen Enable Mode Off	Turns the Pen interrupt mode off.
JH	Pen Enable Mode On	Turns the Pen interrupt mode on. When the RTS is touched the MPIS133 automatically reads and sends the data over ther serial port.
JI	Hex Mode On	Turns the Hex mode on so data transmission is in hexidecimal Raabbccdd.
JJ	Decimal Mode On	Turns the decimal mode on. All RTS data are transmitted in base 10 format.
JK	Input Port On	Turns Port D into a keypad touchscreen input port.
JL	General I/O Port On	Turns Port D into a general I/O port.
JT1	Read Temp Degrees C	Reads Temp sensor and prints in Degrees C
JT0	Read Temp Degrees F	Reads Temp sensor and prints in Degrees F
JV0	Read Voltage Channel 0	Reads voltage on CH 0 and transmits
JV1	Read Voltage Channel 1	Reads voltage on CH 1 and transmits
JXaaabbb	Store X calibration data	Stores X calibration data. aaa=<000-255> X high value bbb=<000-255> X low value.
JYaaabbb	Store Y calibration data	Stores X calibration data. Same as above for Y. Decimal Data input only.
Jxxxyyy	Initialize LCD	Sets and stores the resolution of the LCD and initializes the LCD. xxx=<032-640> yyy=<032-240>
Kna	Soft Key Pad	Key Pad position: n=<X,L,R,T,B> X=1 Key @ current cursor position i.e. Kn0 L= Place <a> # of keys on left side of screen R= Place <a> # of keys on right side of screen T= Place <a> # of keys on top of screen B= Place <a> # of keys on bottom of screen Number of Keys in Keypad: a=<0, 4-8> 0= 1 key at current cursor location 4-8= Number of keys that are drawn at location <n> on the display

KxxxyyyLDab	Soft Key Pad	<xxx>,<yyy> = Location of Keypad
		L=0, D=<0:2> 0=Clear 1= Set 2= XOR Keypad
		<a,b> a= Number of keys in the X direction
		b= Number of Keys in the Y direction

KDnn	Deactivate Event Key	Turns off Key number nn to deactivate event driven subroutine. <nn>=01-80
KE	Erase all Event Keys	Turns off all 80 event driven subroutine keys.
KPnnsss	Assign Event Key when Pressed	Assigns key <nn> to subroutine <ss> <nn>=01-80 and <sss>=000-127
KUnnsss	Assign Event Key when Released	Assigns key <nn> to subroutine <ss> <nn>=01-80 and <sss>=000-127

Quick Reference

Command	Name	Description
KSnnn	Assign Key/Touchscreen Debounce Delay	Assigns delay from 2.5ms to 637ms <nnn>=001-255 Default is 200
L+	Back Light On	Turn the Backlight on
L-	Back Light Off	Turn the Backlight off.
Lxxxyyy	Move cursor	Locate cursor on layer 1@: xxx=<000:079>, yyy=<000:029>
LxxxyyyLDvvvhhhab	Define LED Font	Location of LED Display: xxx=<000:639>, yyy=<000:255> Line Style: L=<0:7> **See Horizontal Line** Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR Vertical Segment Length: vvv=<000:239> Horizontal Seqment Length; hhh=<000:639> Segment Width: a=<0:2> Font Assignment #: b=<0:7>
LDbnnnnnnnnnnnnnnnn	Display Led Numbers	Font Assignment #: b=<0:7> n= <0-9 . - + space>
Mxxxyyy	Move cursor	Locate cursor on layer 2@: xxx=<000:639>, yyy=<000:255>
N+	Increase Contrast	Increase Contrast 1 unit 39ohms/bit
N-	Decrease Contrast	Decrease Contrast 1 unit 39ohms/bit
N*	Reset Contrast	Reset Contrast to start up setting
Nhh	User Contrast Setting	Sets contrast to a hexidecimal user defined setting . hh=<00:FF>
OxxxyyyQnn	Built-in Bitmaps	Location of Bitmap: xxx=<000:639>, yyy=<000:255> Bitmap Action: Q=<D, R,O,X> D= Display bitmap R= Reverse bitmap image O= Inclusively overlay image (ior) X= Exclusively overlay image (xor) Bitmap assignment #: n=<0:99> n= Bitmap reference number
PIq	Read Port	Read Port q <q>=3 Data is sent over the serial port.
PWq bbb	Write Port	Write Port q <q>=3 <bbb> = decimal data to be sent over the port.

Quick Reference

Command	Name	Description
PxxxyyyD	Plot Pixel	Location of Pixel: xxx=<000:639>, yyy=<000:255> Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR
Qchhhhhhhhhhhhhhh	Plot Byte Data	Byte Data Action: C=<L,T> L=Load byte data to display memory starting at current cursor location. Cursor location is set with the Mxxxyyy command. T=Translate data before loading to display memory. Normally the SED 1330 displays data bits b7 through b0. With this parameter you can invert the data to display b0 through b7. This is a quick way to correct bitmap or other display data that may be inverted. Data: <00:FF> Up to 20 bytes of hexidecimal bitmap data. Note: Data is loaded and read from memory by rows. All data must start on a byte boundary. After initial cursor location is set all data is written sequentially from that point.
R-	Normal Video	Normal video display on layer 2
R+	Reverse Video	Reverse video display on layer 2
R0	8K RAM	Selects 8K SED1330 controller
R1	32K RAM	Selects 32K SED1330 controller
R2	Xon/Xoff Control Off	Turns Serial Port XON/XOFF control Off
R3	Xon/Xoff Control On	Turns Serial Port XON/XOFF control On

Quick Reference

Command	Name	Description

Sabccccccccccccccccc	Print Text Layer 2	Text is printed at the current cursor location defined by command M on layer 2. Command F selects the font to be printed. The default font is 1. The command prints up to 40 bytes at a time.
		
		Cursor Direction after each character is printed: a=<R,V>
		a=R, Cursor Right Horizontal display
		a=V, Cursor Down Vertical display
		
		Text Attribute: b=<0:8>
		b=0, Normal Text , Replace Mode
		b=1, Reverse Text, Replace Mode
		b=2, Underline Text, Replace Mode
		b=3, Normal Text, Overlay Mode
		b=4, Reverse Text, Overlay Mode
		b=5, Undeline Text, Overlay Mode
		b=6, Normal Text, Exclusively Or
		b=7, Reverse Text, Exclusively Or
		b=8, Underline Text, Exclusively Or

SLaaaccccccccccccccccc	Load Subroutine	Loads 1 command at a time to designated memory area. aaa=Subroutine Number <00-127> c=Command data up to 40 bytes
SXbbb		Execute Subroutine bbb=Subroutine Number <00-127>
SE		Erase Subroutine memory
SM		Display the number of subroutine memory bytes free.
SZn		Assign memory area n=<0,1> 0=Bit map memory area 1=Window memory area

SSn	Save Screen	Saves current screen to memory location n
		
		n=0 Bit Map memory area
		n=1 Window memory area

SWn	Restore Screen	Restores current screen to memory location n
		
		n=0 Bit Map memory area
		n=1 Window memory area

Quick Reference

Command	Name	Description

Tacccccccccccccccc	Print Text Layer 1	Text is printed at the current cursor location defined by command L on layer 1. This command prints up to 40 bytes at a time
		
		Cursor Direction after each character is printed: a=<R,L,U,D>
		a=R, Cursor Right Horizontal display
		a=D, Cursor Down Vertical display
		a=L, Cursor Left Horizontal display
		a=U, Cursor Up Vertical display
		c=ASCII Character Code

UxxxyyyLDrrrdddSaazzz	Circle	Center of the circle/ellipse: xxx=<000:639>, yyy=<000:255> Line Style: L=<0:7> 0=Solid 1=Dotted/q2 2=Dotted/q3 3=Dotted/q4 4=Dashed/q4 5=Dashed/Dotted/q4 6=Dashed/q5 7=Dashed/q7 Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR Circle Radius: rrr=<004:175> # of Degrees in the circle: End of Arc ddd=<001:360> Start of Arc in Degrees zzz=<000:359> Note: ddd>zzz Change in magnitude of Y radius for ellipse creation: S=<0,1> 0= Decrease Y Radius 1=Increase Y Radius Note: Use the <S> and <aa> parameters to correct for aspect ratio of your display. If a circle drawn on your display looks like an ellipse then increase/decrease the Y radius to make the circle round. # of Pixels for Y radius correction: aa=<00:99>
VxxxyyyLDdddwww	Vertical Line	Starting Location of line: xxx=<000:639>, yyy=<000:255>

Quick Reference

Command	Name	Description
VxxxyyyLDdddwww	Vertical Line Continued	Line Style: L=<0:7> 0=Solid 1=Dotted/q2 2=Dotted/q3 3=Dotted/q4 4=Dashed/q4 5=Dashed/Dotted/q4 6=Dashed/q5 7=Dashed/q7 Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR Dummy Variable: ddd=<000> Line Length; www=<000:639>
WxxxyyyQn	Window	Size of Window in bytes - 1024 bytes/window maximum: xxx=<000:aaa>, yyy=<000:bbb> (aaa*bbb)/8 must be <=1024 bytes i.e.128x64 bits. See Text for allocating more memory. Bit Map Action: Q=<A, D, R,C> A= allocate window memory based on xxx*yyy=# of bytes D= Display/Open window at location xxx, yyy. R=release window memory based on xxx*yyy=# of bytes C=Close window that is open at location xxx,yyy. Window assignment #: n=<0:7> n= reference number that defines the window
Xahhhhhhhhhhhhhhh	User Initialization	See Detailed Summary
Yn	Serial Port Baud Rate	Baud Rate: n=<0-9> Bits/Second Low Speed: 0=9600, 1=19200, 2=38400, 3=57600 High Speed: 4=115200, 5=250000, 6=375000, 7=500000, 8=625000, 9=1250000

Quick Reference

Command	Name	Description
YxxxyyyLDvvvhhhab	Define Bar Graph	Location of Bar Graph: xxx=<000:639>, yyy=<000:255> Upper left hand corner Line Style: L=<0:7> See Horizontal Line Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR Vertical Bar Graph Length: vvv=<000:239> Horizontal Bar Graph Length; hhh=<000:639> Bar Graph Type: a=<0:5> <0>= Solid No Border <1>= Solid Single Line Border <2>= Solid Double Line Border <3>= Segmented No Border <4>= Segmented Single Line Border <5>= Segmented Double Line Border Bar Graph Assignment #: b=<0:7>
YDbnnn	Display Bar Graph	Bar Graph Assignment #: b=<0:7> nnn= Current length of bar graph in pixels <000:639>
ZxxxyyyLDaaabbb	Vector	Starting Location of vector: xxx=<000:639>, yyy=<000:255> Line Style: L=<0:7> 0=Solid 1=Dotted/q2 2=Dotted/q3 3=Dotted/q4 4=Dashed/q4 5=Dashed/Dotted/q4 6=Dashed/q5 7=Dashed/q7 Pixel Type: D=<0:2> 0=Clear 1= Set 2= XOR Ending Location of vector: aaa=<000:639>, bbb=<000:255>

Chapter 3: Builder Reference Guide

Introduction

This section provides a detailed explanation of the command language. The commands are grouped by their function. For example all commands that control cursor functions are listed together. In addition, examples are given in certain cases to better explain the functionality of the command. Commands not implemented and listed in the quick reference guide are reserved for future applications.

BackLight Control

Commands *L+,L-*, and *Ia* control the function of the backlight inverter.

Backlight Inverter On

Command *L+* is issued to turn the inverter backlight on. This command sets pin 9 to its high state (+5 volts). The pin will source 25mA of current which is able to drive a small low current +5 volt relay. This means the *L+* can be used to turn any device on through the proper interface. If the current requirements exceed 25mA then use pin 9 to turn a NPN transistor on which in turn will control a larger relay on the device you want to turn on,

BackLight Inverter Off

Command *L-* is used to turn the backlight inverter off. This command clears pin 9 to its low state (Ground). The pin will sink 25mA of the current. The pin can be used for other functions just like *L+* command.

Timer Interval

Command *Ia* is used to control how long pin 9 stays in its high state. <a> can vary from 0 to 9. 0 indicates the backlight inverter is in the manual mode controlled strictly by *L+,L-.* Each value of a from 1 to 9 increases the time the backlight is on by 27 seconds.

Example: >*I4*

This will set the timer to 4*27 or 1 minute and 54 seconds. After 1 minute and 54 seconds the routine will automatically issue a *L-* command and turn off the light. The maximum on time for the timer is 4 minutes and 5 seconds. This value is stored in EEPROM. Once it is set the value will be reloaded on start-up of the MPIS133X.

Bar Graph Definition

The *YxxxyyyLDvvvhhhab* command is use to define bar graphs for display on the screen. Up to 8 different active bar graphs can be defined. <xxx>, <yyy> parameters define the upper left hand corner of the bar graph. <L> defines the line style that will be used when the border is displayed if a border has been selected. <L> must range between 0 and 7.

<D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <vvv> defines the maximum vertical length in pixels of the bar graph display. <vvv> must range between 0 and 255. <hhh> defines the maximum horizontal length in pixels of the bar graph display. <hhh> must range between 0 and 639. <a> defines the type bar graph to be displayed and border if required.

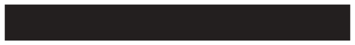
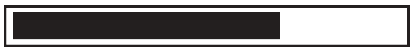
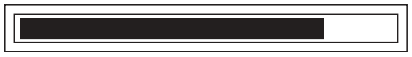
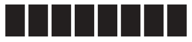
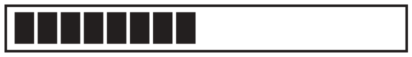
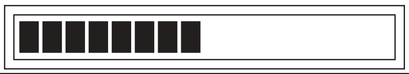
<a> must range between 0 to 5. When a=0 the bar graph is solid without a border. When a=1 the bar graph is solid with a single line rectangular border. When a=2 the bar graph is solid with a double line rectangular border. When a=3 the bar graph is segmented without a border. When a=4 the bar graph is segmented with a single line rectangular border. When a=5 the bar graph is segmented with a double line rectangular border.

<b> defines the bar graph assignment number to the current bar graph parameters. <b> must range between 0-7. This allows you to define up to 8 active bar graphs at one time. When you display a bar graph you only have to use the assignment parameter with the bar graph length in pixels to be displayed. Since all the information about the bar graph is save in a buffer you can quickly update a number bar graphs with different characteristics on the screen.

Bar graphs can be either horizontal or vertical depending on the bar graph definition. Since you determine the actual length and width of the graph the software will display a vertical graph when the vertical length is greater than the horizontal length. When the opposite is true the software will display a horizontal bar graph. You can use the axis command to display a graph and then use the bar graph command without a border to display information on the graph. The bar graphs with borders can be use to display information such as how full a battery is from 0 to 100%.

Bar Graph Type

The following table shows the possible bar graph types in a horizontal orientation.

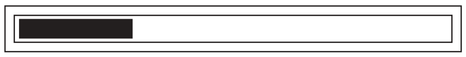
a	Bar Graph Style	Border
0	Solid no border	
1	Solid	
2	Solid	
3	Segmented no border	
4	Segmented	
5	Segmented	

Bar Graph Display

The *YDannn* command is use to display the bar graphs on the screen as defined by the previous command. Since up to 8 different active bar graphs can be defined the <a>=0-7 parameter assigns a definition buffer to the bar graph to be displayed. The parameter <nnn> defines the number of pixels (length) to be displayed. The parameter <nnn> can range from <000:639>. To clear a bar graph send *YDa000* and the bar graph will be reset to zero.

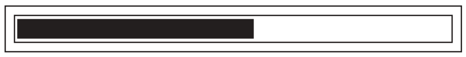
When sending data to the screen you are sending the total number of pixels you wanted displayed on the screen.

Example: *YD0025*



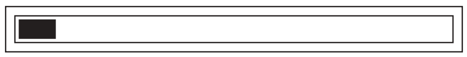
This creates a bar graph 25 pixels in length. All bar graphs are initialized to 0 when they are created. In this example it is assumed that the bar graph definition is for a solid graph with a double line border.

Example: *YD0050*



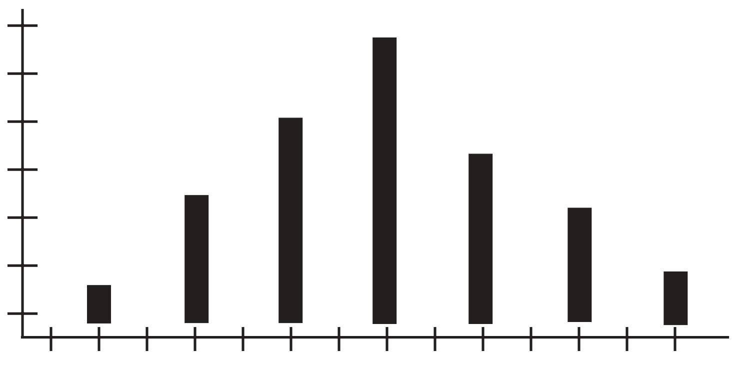
This time the graph is extened to a total length of 50 pixels. To make the bar graph smaller send a number less that then last entry which was 50.

Example: *YD0010*



This defines a bar graph of only 10 pixels in length. You are sending the total length not the change in length. When using the segmented graph remember the bar graph is created with a bit mask. The first bit is zero therefore the first line in the bar graph will be missing. It will become the first space between segments on the ninth line from the start of the bar graph. Use the solid graph when displaying very accurate data such as battery power. Use the segmented bar graph when displaying data with greater tolerance because you will have a line of data that is displayed as a space every 8 pixels to create the segmented graph.

Once the bar graph definition is in place you can change the length of each bar graph by using the LD command. Remember you can control up to 8 bar graph displays on the screen simultaneously. If you need to control more you will have to define that bar graph display before it is written to the screen. By using the bar graph command without a border and the axis command you can create graphs like this.



Bitmaps

User Defined

The commands *GxxxyyyQn* and *Gcnhhhhhhhhhhhhhhhh* allow the user to generate bitmaps of your own creation. **If your system has 8K RAM then you are limited to 6 bitmaps and the memory is shared with the *Window* command. You must you care when allocating bitmap and window buffers. The first 2 bitmaps buffers occupy the same space as the first window buffer 0. Bitmap buffer 2 and 3 occupy the same space as window buffer 1. Bitmap buffer 4 and 5 occupy the same space as window buffer 2. You can occupy one buffer space or the other but not both at the same time. LCD's with 640x200 pixels or more will require some if not all of this memory due to the large requirements of memory for screen display.** Up to 16 different bitmaps can be defined with the 32K system each 512 bytes in size. Each bitmap can be 64x64 bits in size for example. If you need a bitmap larger than 512 bytes, divide the bitmap into more than one section and paste the bitmaps together on the screen. Bitmap memory is reusable. You allocate the memory block with <Q>=A. This will allocate xxx*yyy memory bytes and clear those memory bytes to H'00'.

Example: >*G003016AB*

This allocates 48 bytes (3*16) of memory and assigns this memory block as bitmap 11. Remember there are 8 bits/byte in the X direction. Therefore this bitmap is 24*16 bits in size. To reuse the memory block you must tell the system to release it, This is done as follows.

Example: >*G003016RB*

This releases the previous 48 bytes of memory and clears the RAM values to H'00'. Remember to allocate the memory block before you release the memory block.

To display the bitmap you must first load the RAM memory with the values you want to display. This is done with the *Gcnhhhhhhhhhhhhhhhh.* After the memory has been loaded it can be displayed with the <Q>=D parameter. When using this part of the command the xxx and yyy designate the starting location of the upper left hand point of the bitmap. Bitmaps can be plotted on any bit boundaries in the X direction and Y direction.

Example: >*G008016D1*

This will display bitmap 1 at pixel location x=008, y=016. The bitmap must be pre-defined or nothing will be displayed as the memory locations all contain H'00'.

Loading User Defined Bitmaps

Loading bitmap data into memory is accomplished with the *Gcnhhhhhhhhhhhhhhhh* command. The <c> parameter tells the system to either <c>= L start loading the bitmap or <c>=C continue loading from the last byte stored in memory. The <n> parameter tells the system which bitmap memory block to load the data in. <n> can range from 0-F which in hexidecimal (0-15 decimal) and represents the current bitmap number the user will define. <hh> is the actual bitmap data is hexidecimal format. You can load up to 20 bytes of data per command line. Data is loaded by column. The first column is loaded, then the second and so on until all data has been stored.

Example: >*GL000000000001F1010*

This command loads bitmap 0 starting from the first RAM byte with the hex data 00 00 00 00 00 1F 10 10.

Example:>*GC010101F0000000000*

This command continues to load bitmap 0 from the last data point that was entered into RAM with the hex data 10 10 1F 00 00 00 00 00. Remember you have 16 availible bitmap memory blocks that are 512 bytes in length. You can reuse these blocks by releasing the memory block <Q>=R parameter.

Built-in Bitmaps

The MPI S133 has 50 built-in bitmaps that were designed to fit inside the soft keys the MPI S133 can display anywhere on the screen. These bitmaps are 24x16 bits in size. Use the *OxxxyyyQnn* command to display these bitmaps. <xxx>, <yyy> define the starting location of the upper left hand point of the bit map. To display a bitmap <Q>=D is the parameter used. To display the reverse image of a bitmap <Q>=R is the parameter used. To overlay the display use <Q>=O parameter. To inverse overlay (XOR) use the <Q>=X parameter. <nn> parameter defines the bitmap to be display. This number ranges from 00-49.

Example: >*O288013D04*

This command will display bitmap #4 at location X=288, Y=013. You can use the reverse image display to indicate that the key has been pressed. Display the normal image initially, then display the reverse image when the key in pressed. After the key is released display the normal image again. This method provides a simple yet effective way of indicating to your end user that an action has taken placed. The following is a listing of bit maps availible in this release version.

Bitmap #	Bit Map Image
00	Clear Right Arrow
01	Solid Right Arrow
02	Clear Left Arrow
03	Solid Left Arrow
04	Clear Up Arrow
05	Solid Up Arrow
06	Clear Down Arrow
07	Solid Down Arrow
08	Clear Return Arrow
09	Solid Return Arrow
10	Light Bulb Off
11	Light Bulb On
12	Water Faucet On
13	Clock

Bitmap #	Bitmap Image
14	Lock - Security
15	Phone
16	Clear Circle - Record Button
17	Solid Circle
18	Clear Square - Stop Button
19	Solid Square
20	Clear Left Triangle - Reverse Button
21	Solid Left Triangle
22	Clear Right Triangle - Forward Button
23	Solid Right Triangle
24	Clear Up Triangle - Eject Button
25	Solid Up Triangle
26	Clear Down Triangle
27	Solid Down Triangle
28	Pause Button
29	Circle with a + inside
30	Circle with a - inside
31	Rain Clouds
32	Graphics Button
33	Folder
34	Circle/Slash - No Button
35	Bell
36	Battery
37	Fuel Gauge
38	Temperature Gauge
39	Oil Pressure
40	High Beam Indicator
41	Low Oil Pressure
42	Low Fuel
43	Check ABS
44	Mail
45	Clouds
46	Sun
47	Person
48	Swimming Pool
49	Solar Panel
50-99	Not implemented at this time.

The bitmaps were designed for the worst case aspect ratio of .71. Therefore the circles will become elliptical on displays with aspect ratios greater than .71. Also squares will be come rectangles on displays with aspect ratios greater than .71. These small changes will still provide very useful, nice looking bitmaps inside the soft keys.

Circle/Ellipse

The MPI S133X will generate circles/ellipses with ease. Use the *UxxxyyyLDrrrdddSaazzz* command to display a circle or an ellipse on the screen. <xxx>, <yyy> define the center of the circle. <L> defines the line style that will be used when the object is plotted. <L> must range between 0 and 7. <D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <rrr> defines the radius of the circle. <rrr> must range between 4 and 175. <ddd> defines the number of degrees in the circle that are to be plotted or the end of the arc function. <ddd> must range between 1 and 360 degrees. <zzz> defines the start of the arc function. <zzz> must be less than <ddd> and between 0 and 359 degrees. <S> indicates the direction the Y axis will be moved to create an ellipse or correct the aspect ratio of the display. <S>=0 means the Y axis radius will be **decreased** by the number of pixels defined by parameter <aa>. <S>=1 means the Y axis radius will be **increased** by the number of pixels defined by parameter <aa>. <aa> defines the number of pixels that the Y axis will be increased or decreased. <aa>=00 means that no correction will occur.

Line Style

The following table shows the line style possiblities.

L	Line Style	Line
0	Solid	___________________
1	Dotted/q2	....
2	Dotted/q3	. . . .
3	Dotted/q4	. . . .
4	Dashed/q4	- - - - - - - - - - - - - - -
5	Dashed/Dotted/q4	_._._._._._._._._._._
6	Dashed/q6	_ _ _ _ _ _ _ _ _ _ _ _ _
7	Dashed/q7	___________________

Example: *U050075010303600000*

This command draws a circle with a center at 50,75 and a radius of 30 pixels. The circle is a solid line, 360 degrees arc radius with no aspect ratio correction.

Example: *U050075010303600009*

This command draws an ellipse with a center at 50,75 a major radius of 30 pixels, a minor radius of 21 pixels with a solid line all the way around the ellipse.

Example: *U05007501030060000015*

This command draws an arc of 45 degrees in the first quadrant between 15 and 60 degrees 30 pixels from the center at 50,75.

Clock

The MPI S133X will generate a real time clock/timer using on board timers linked to the timing crystal. The clock is accurate to approximately 15 seconds a month but is volatile like the clock in your car and must be set when power is turned off. The *CK2* command must be used to set the clock mode. This command must be used before you define the clock or set or read the clock/date. This is the default mode. Version 3.0 stores the clock values every hour. If power is lost the clock will revert back to the last hour values saving time when resetting the clock.

Use the *CxxxyyyLDvvvhhhFTWABC* command to define the clock and date function on the screen. <xxx>, <yyy> defines the upper left hand corner of the clock display. <L> defines the format that will be used when the clock is displayed. <L> must range between 0 and 3. <D> defines the text attribute associated with the display. This is the same attribute for the text display command *S* on layer 2. See that section for more information on the text attribute. <D> must range between 0 and 8. <vvv>, <hhh> defines the upper left hand corner of the date display. <F> indicates the font to be used when the clock or date are displayed. <F> must range between 0 and 3. <T>= turns the date display on or off. <T> must range between 0 and 1. <W>=defines the date format that will be used when displaying the date on the screen. <W> must range between 0 and 1. <A> defines the separator used between the time digits. <A> can be any Ascii value. <B> defines the separator used between the date digits <B> can be any Ascii value. <C> turns the European date format on/off. 0= EU format off and 1= EU format on. (DD.MM.YYYY)

Setting the Clock/Date

The MPI S133X sets the time and date with two commands. *CKSThhmmssd* is used to set the time. The clock must be set in a 24 hour format. <hh> sets the hours parameter. The value must range between 00 and 23. <mm> sets the minutes parameter. The value must range between 00 and 59. <ss> sets the seconds parameter. The value must range between 00 and 59. <d> sets the day of the week. The range is 0 to 6 which corresponds to Sunday through Saturday. The clock does not compensate for daylight savings time.

Example: >*CKST1420304*

In this example the clock is set to 2:20:30 pm or 14:20:30 in the 24 hour format and the day of the week is set to Thursday.

CKSDmmddyyyy is used to set the date. <mm> sets the month parameter. The value must range between 01 and 12. <dd> sets the day of the month parameter. The value must range between 01 and 31. <yyyy> sets the year parameter. The value must range between 2002 and what ever year it might be. The date function does not compensate for leap year.

Example: >*CKSD03102002*

In this example the date is set to March 10, 2002.

Reading the Clock/Date

The MPI S133X uses two commands to read the current time/date and send that information to the host computer through the serial port.

CKRT will read the current time. *CKRD* will read the current date. The time is sent in the following format HH:MM:SS and the date is sent in this format MM/DD/YYYY.

Controlling the Clock

The MPI S133X uses eight commands to control the function of the clock. *CK0* suspends the display of the clock. This command is very useful when changing to a screen that does not use the clock display. Sending this command keep the display from appearing on that screen. It also can be used when sending a large amount of information to the controller. By suspending the display the throughput can be increased. This is the default condition when the controller boots. *CK1* resumes the display of the clock. This command must be used to start the display after the controller boots.

CK2 defines the clock as a clock . This is the default mode. This mode updates both time and date functions. *CK3* defines the clock as a timer changing its function to that of a stop-watch or up/down timer. The *CK4* command sets the automatic update of the clock at 1 minute intervals. The *CK5* command sets the automatic update of the clock to 1 second. The default mode is 1 minute. *CK6* starts the Timer function. *CK7* stops the Timer Functions. *CK8* Resets the Timer.

Displaying the Time/Date/Day of the Week

The MPI S133X uses three commands to display the time, date, and day of the week on the screen. The *CKT* command will display the time on the screen based on the parameters defined in the *CxxxyyyLDvvvhhhFTW* command. The *CKD* command will also display the date on the screen based on the parameters defined in the *CxxxyyyLDvvvhhhFTW* command. The *CKUxxxyyyT* command defines the location of the day of the week (DoW). The *CKC* command will display the day of the week as an abbreviate word (i.e Sunday=Sun) based on the Time/Date font/style definition and *CKUxxxyyyT* location. <xxx> must range between 000 and 639. <yyy> must range between 000 and 255. <T> must range between 0-1. T=0 indicates the DoW display is turned off. T=1 turns the DoW display on.

Setting the time/date/day of the week through key subroutines

The MPI S133X uses fourteen commands the allow the you to set the clock/date/day of the week through the use of Key Event Subroutines. These command will increment or decrement the clock registers by 1 based on the command. The following table describes the commands.

Command	Function	Command	Function
CKH+	Increase Hours	CKN+	Increase Month
CKH-	Decrease Hours	CKN-	Decrease Month
CKM+	Increase Minutes	CKA+	Increase Day of the Month
CKM-	Decrease Minutes	CKA-	Decrease Day of the Month
CKX+	Increase Seconds	CKY+	Increase Year
CKX-	Decrease Seconds	CKY-	Decrease Year
CKW+	Increase Day of the Week		
CKW-	Decrease Day of the Week		

Contrast Control

The contrast of a LCD can be controlled through 4 software commands. Control of the contrast by software requires extra hardware (see figure 5). Contrast is increased which makes the screen darker by sending command *N+*. Contrast is decreased which makes the screen lighter by sending command *N-*. The *N** is used to reset the contrast to the factory setting that is determined by the display type setting on the DIP switch. This will reset the contrast voltage to the voltage recommended by the manufacturer of the LCD based on the average voltage for the display resolution of your LCD over a number of manufacturers. *Nhh* command will set the contrast to a hexidecimal value defined by hh. 80h is approximately 5000 ohms (-9 volts). Contrast adjustment voltage is nonlinear. The contrast voltage can be adjusted from approximately -6 volts (H'FF') to -20 volts (H'00'). Decreasing the number will increase the contrast. You can read the ambient temperature and through the manufacturer's temperature vs. voltage table correctly update the contrast as the ambient temperature changes. Values are stored in EEPROM.

Cursor Control

There are fourteen commands used to control the cursor. Thirteen are involved with the cursor located on layer 1 which is the text layer. The last command controls the cursor on layer 2 which is the graphics/text layer.

Text Cursor Control

To turn the text cursor on use the *C+* command. To turn the cursor off use the *C-* command.

Cursor Style

When the system starts-up the default cursor is a block non-blinking cursor locate at character position 0,0. To change the cursor to underline cursor send command *CA*. To change the cursor back to a block cursor send command *CB*. To blink any style cursor at 1 Htz send the command *C1*. To blink any style cursor at 2 Htz send the command *C2*. To stop any cursor style from blinking send the command *C0*.

Cursor Direction

To send the cursor to its home position (0,0) send the command *CH*. To move the cursor right 1 character position send the command *CR*. To move the cursor left 1 character position send the command *CL*. To move the cursor up 1 character position send the command *CU*. To move the cursor down 1 character position send the command *CD.*

Text Cursor Position

To place the text cursor at any character location send the command *Lxxxyyy*. This will place the cursor at location xxx,yyy. The acceptable range for xxx is 0 to 79. The acceptable range for yyy is 0 to 29. All data is entered as 3 digits . If xxx=1 then xxx is entered as 001. Text is printed at the current cursor position. Use the *Lxxxyyy* command to place text anywhere on layer 1.

Graphic's Cursor Position

To place the graphic's cursor at any pixel location send the command *Mxxxyyy*. This will place the cursor at location xxx,yyy. The acceptable range for xxx is 0 to 639. The acceptable range for yyy is 0 to 255. All data is entered as 3 digits . xxx=1 is entered as 001. Text is printed at the current cursor position. Use the *Mxxxyyy* command to place text anywhere on layer 2. **Note: The graphic cursor is not a visible cursor.**

Display Control

There are four commands used to control the LCD display. The *D+* command turns the display on. The *D-* command turns the display off. If you are building a complex screen, the screen can be turned off while it is being built, then turned on. This allows for a cleaner interface. These commands turn both layers on and off. The *D2* command will <or> layer 1 and layer2. This is a simple overlay and is the default condition. The *D3* command will <xor> layer 1 and layer 2. This will create an interesting effect. Experiment with it to see.

Select Font

The *Fa* command selects the font that will be display with the *<S>* command. Currently there are four fonts loaded in the MPI S133X ROM. <a>=0 will select the 5x7 built-in font. <a>=1 will select the 7x9 font. <a>=2 will select the 11x13 font. <a>=3 will select the 15x17 font. The 7x9 is the default font when the system boots. Font select applies to text that will be displayed on layer 2 only.

Graph - X,Y Axis Setup

The *AxxxyyyLDhhhwwwGTT* command will draw a X,Y axis setup with a number of different options. <xxx>, <yyy> parameters define the upper left hand corner of the rectangle that defines the area the graph will occupy. <L> defines the grid style that will be used when the axis is plotted. <L> must range between 0 and 7. <D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <hhh> defines the Y axis length. <hhh> must range between 0 and 255. <www> defines the X axis length. <hhh> must range between 0 and 639. <G> defines the graph type. <G> must range between 0 to 2. <TT> defines the number of pixels/tic mark on each axis. <TT>=00 means each tic mark will be separated by 10 pixels. <TT> must range between 0 to 99.

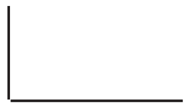
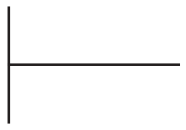
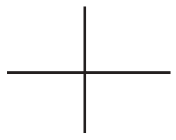
The <hhh>, <www> parameters define the rectangle where the graph will reside. If the graph is the first quadrant then the rectangular area will contain the entire first quadrant. If the graph is first and fourth quadrant the rectangular area will contain both quadrants.

The tic marks on the X and Y axis are separated by 10 pixels in the default condition. You can set this number to any pixel count between 0 and 99. This will allow you to define the scale on the X and Y axis.

You can define different types of graphs. Use the following chart to set the <G> parameter.

Graph Type

The following table shows the graph type possiblities.

G	Graph type	Graph
0	1st Quadrant - No Frame	
1	1st/4th Quadrant - No Frame	
2	Four Quadrants - No Frame	

Grid Style

The following table shows the grid style possiblities.

L	Grid Style	Line
0	No Grid	____________________
1	Dotted/q2	....
2	Dotted/q3	. . . .
3	Dotted/q4	. . . .
4	Dashed/q4	- - - - - - - - - - - - - - -
5	Dashed/Dotted/q4	_._._._._._._._._._._
6	Dashed/q6	_ _ _ _ _ _ _ _ _ _ _ _ _
7	Dashed/q7	____________________

Since <xxx>, <yyy> define the upper left hand corner of the axis area with the origin of the axis located at <xxx>,<yyy+hhh> for graph type 1. For graph type 2 the origin is <xxx>,(yyy+hhh/2). For graph type 3 the origin is <xxx+www/2>,<yyy+hhh/2>. The tic marks on the x and y axis will extend 2 pixels on each side of the respective axis. For the tic marks to space evenly on the axises the length of x and y must be divisible by the tic mark length. With the default length of 10 pixels the x and y axis must be a multiple of 10 to complete each axis with a tic mark.

The grid linies are lines extending from the X and Y tic marks. These allow you program to display data on the screen that is easier to interpret. The grid lines correspond to the line style routine and add greatly to the presentation of your data.

Example: >*A05005021100200020*

This command will draw a 1st quadrant graph with a dotted grid at 50,50 and tic marks at a 20 pixel interval.

Horizontal Line

To generate horizontal lines use the *HxxxyyyLDdddwww* command. <xxx>, <yyy> define the starting point of the line. <L> defines the line style that will be used when the object is plotted. <L> must range between 0 and 7. <D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <ddd> is a dummy variable. <ddd> should be 000. <www> defines the length of the line. <www> must range between 0 and 639.

Line Style

The following table shows the line style possiblities.

L	Line Style	Line
0	Solid	________________
1	Dotted/q2	....
2	Dotted/q3	. . . .
3	Dotted/q4	. . . .
4	Dashed/q4	- - - - - - - - - - - - - -
5	Dashed/Dotted/q4	_._._._._._._._._._._
6	Dashed/q6	_ _ _ _ _ _ _ _ _ _ _ _
7	Dashed/q7	________________

Key Input

Input from the key port will support up to 8 keys. When a key is press the MPI S133X will transmit the key code *Kn* with n=1 to 8. When the key has been released the system will transmit the ready prompt ">" indicating the system is ready for another command. The system cannot accept any data as long as the key is pressed. If you use the XON/XOFF control the XOFF code is sent as soon as the key is pressed and XON is sent as soon as it has been released. The same principal applies to the touchcreen mode. In this mode the code *Tnn* is sent with n=01 to 99 corresponding to the touchscreen position that has been activated. Once the touch position is released the ready prompt ">" will be sent. Commands can not be sent while the system is busy reading the touchscreen.

KeyPads

The *Kna* command will display keypads on the screen. You can display individual keys or there are 4 built-in positions that will display between 4-8 keys as a group. <n> parameter defines the type and location of the keys. When <n>=X only 1 key will be placed on the screen at the current cursor location. The current cursor location is set by the *Mxxxyyy* command. When <n>=L <a> number of key pads are placed on the left margin of the screen. When <n>=R <a> number of key pads are placed on the right margin of the screen. When <n>=T <a> number of key pads are placed on the top margin of the screen. When <n>=B <a> number of key pads are placed on the bottom margin of the screen. <a> parameter defines the number of keys to be placed on the screen. When <a>=0, 1 key is placed on the screen. When <a>=(4-8), between 4-8 keys are placed at one of the defined margins of the screen based on parameter <n>. Every keypad has an interior area that will display a 24x16 bit bitmap. Bitmaps are mapped X+4, Y+3 pixels from the location of the keypad. Use the tables on the following pages for X,Y location of built-in keypads and bit map locations. The built-in keypad routines require a minimum 128 pixels in the Y direction and 160 pixels in the X direction. If you have a smaller display use the *M* command and place 1 keypad at a time or use the *KxxxyyyLDab* command .

The *KxxxyyyLDab* command will display a keypad matrix on the screen. <xxx,yyy> will set the current cursor location of the upper left hand point of the keypad. L is not used and equals zero. D <0:2> determines if the keypad is erased <0>, written<1>, or Xor <2> on the screen. <a,b> set the size of the keypad matrix in the x,y direction.

K0500250134

This command write a 3x4 keypad at location 50,25 on the screen.

Keypads in the keypad matrix are offset 40 pixels in the X direction and 27 pixels in the Y direction. To place a bitmap in a keypad always offset the bitmap 4 pixels in the x direction and 3 pixels in the Y direction for the upper left hand corner. Then add the X and Y offsets to find the location of the other keypads in the matrix. In the previous example K0500250134, the first bitmap is locates at 54,28. The next bitmap for the next keypad in that same row is located at 94,28. The bitmap location for the first keypad in the second row is 54,55.

The following table shows the offsets between keypads and bitmaps for the KLn and KRn commands. The starting location for KLn is (4,10). The starting location for the KRn is (X display resolution - 36,10). The bitmap location within the keypad is (+4,+3). This means you take the keypad location and add 4 to the X value and 3 to the Y value and this will give you the starting location. The table displays the offset based on the display resolution in the Y direction vs. the command KXn. As you can tell when a value is left blank (---) the display is too small to display the keypad command correctly. If you execute a command that will not fit on the display, the display will show garbage. If you need different offsets or a different configuration then the KXn command provides then use the KxxxyyyLDab command to display the keypad that you desire.

Display Resoltuion	Ka4	Ka5	Ka6	Ka7	Ka8
128	29	-----	-----	-----	-----
160	40	30	-----	-----	-----
192	50	38	30	-----	------
200	52	40	32	27	------
240	66	50	40	33	28
256	72	54	43	36	31

The table for KLn and KRn commands displays the offsets in pixels between keypads based on display resolution. Remember the starting location for KLn is (4,10). The starting location for the KRn is (X display resolution - 36,10). The bitmap location within the keypad is (+4,+3).

The next table on the following page shows the offsets between keypads and bitmaps for the KTn and KBn commands. The starting location for KTn is (4,4). The starting location for the KBn is (4,Y display resolution - 24). The bitmap location within the keypad is (+4,+3). This means you take the keypad location and add 4 to the X value and 3 to the Y value and this will give you the starting location. The table displays the offset based on the display resolution in the X direction vs. the command KXn. As you can tell when a value is left blank (---) the display is too small to display the keypad command correctly. If you execute a command that will not fit on the display, the display will show garbage. If you need different offsets or a different configuration then the KXn command provides then use the KxxxyyyLDab command to display the keypad that you desire.

Display Resoltuion	Ka4	Ka5	Ka6	Ka7	Ka8
160	40	-----	-----	-----	-----
192	51	38	-----	-----	-----
240	67	50	40	-----	------
256	72	54	43	36	------
320	93	70	56	47	40
480	147	110	88	73	63
640	200	150	120	100	86

The table for KTn and KBn commands displays the offsets in pixels between keypads based on display resolution. Remember the starting location for KTn is (4,4). The starting location for the KBn is (4, Y display resolution - 26). The bitmap location within the keypad is (+4,+3).

Key/Touchscreen Debounce Delay

The *KSnnn* command is used to set the debounce delay for a key or touchscreen input. Since different devices require a different amount of dead time to prevent a multiple inputs, this command is used to set that delay. <nnn> must range between 001 and 255. This will set a software delay between 2.5ms and 637ms in length. The default delay on start up is 500ms.

Usually a switch or key input will require around 200ms and a IR touchscreen around 400ms. The default value is set high to insure proper conditioning of the input device. If you set the value to low you will receive multiple inputs of the same key right after you process the first input. This is very evident when you use the subroutine function from a key/touchscreen event. You can tune this delay for the best response for your application.

The delay occurs before the ready prompt ">" is sent to the user. This means the key number will be held nnn*2.5ms before the prompt is sent to prevent the system from accepting any more information or processing any more key/touchscreen data.

Key Subroutine Events

Key Pressed Subroutine Event

The *KPnnsss* command is used to assign a stored subroutine<sss> to key number <nn>. This subroutine is execute when the key is pressed. <nn> must range between 01-80 depending on the number of keys you have on you system. <sss> must range between 00-127 corresponding to the subroutine number you have previously stored with the subroutine command. See page 50. The term key is interchangable with touch point on a touch screen. The MPI S133X currently supports up to 8 individual keys and up to 80 touch points on a touch screen.

Example: *KP01003*

This command will assign key number 01 to subroutine 003. When the key is pressed the subroutine will execute.

Key Released Subroutine Event

The *KUnnsss* command is used to assign a stored subroutine<sss> to key number <nn>. This subroutine is execute when the key is released. <nn> must range between 01-80 depending on the number of keys you have on you system. <sss> must range between 00-127 corresponding to the subroutine number you have previously stored with the subroutine command. See page 50. The term key is interchangable with touch point on a touch screen.

Example: *KU01004*

This command will assign key number 01 to subroutine 004. When the key is released the subroutine will execute.

Key Deactivate Subroutine Event

The *KDnn* command is use to deactivate a key to prevent a subroutine from executing. <nn> must range between 01-80. Once a key has been assigned it can be reassigned with the *KPnnsss* or *KUnnsss* commands. To deactivate the key you must use *KDnn*.

Example: *KD01*

This will deactivate Key number 01 in both the pressed state and released state.

Deactivate all Keys

The *KE* command is use to deactivate all the keys 01-80. This will clear the key event index. Any key pressed after this command will not execute any subroutines but still will send the number of the key or touch screen point to the user over the serial port.

Key Programming Notes

The MPI S133X will send a key code when the key is pressed and the ">" character when the key is released. The system is busy and cannot receive any commands until you receive the ">" character. This makes a key event the same as any other command. You still look for the ">" character to indicate the system is ready. If you use XON/XOFF you will receive a XOFF while the system is processing the key and XON when it is ready for data.

LCD Initialization

The *Jxxxyyy* command is used to set the LCD display resolution and initialization parameters. This command replaces the dip switch used in previous versions. The command allows for a wide range of LCD resolutions far exceeding the range of the dip switch. It also replaces the X command used previously to initialize LCD's that where not listed for the dip switch.

xxx is the resolution of the LCD in pixels in the X direction. xxx must range between 32 and 640 pixels. yyy is the resolution of the LCD in pixels in the Y direction. yyy must range between 32 and 240 pixels. Once the command has been entered the values are stored in EEPROM and will be recalled everytime the systems boots. The values are used calculate the register values on the SED1330/5 chip. After the values are calculated the chip is initialized and a block cursor is set in the upper left hand corner.

The contrast is also set using general values for each range of LCD between 32x32 to 640x240. Each manufacturer is different and therefore the contrast may not be right. If the screen is too dark decrease the value with *N-* command until the screen contrast looks right. This value is also stored so the contrast will be correct for that temperature everytime the LCD starts up.

If the contrast is to light or the cursor does not appear then use the *N+* command to increase the contrast until is reaches a value that you are satisfied with. Remember the LCD contrast changes with temperature. As the temperature goes up the contrast voltage should go down. If you measure the temperature of the display you can create a routine that automatically adjusts the contrast based on temperature.

Example: *J320240*

This sets the resolution of the display to 320 X pixels and 240 Y pixels. The display is then initialized and the contast voltage is set. All values are stored in EEPROM.

LED Font Definition

The *LxxxyyyLDvvvhhhab* command is use to define LED style numeric fonts for display on the screen. Up to 8 different active fonts can be defined. <xxx>, <yyy> parameters define the upper left hand corner of the LED display. <L> defines the line style that will be used when the LED font is displayed. <L> must range between 0 and 7. Since the LED display is generated with vectors you can define the style of the vectors for different effects.

<D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <vvv> defines the length in pixels of the vertical segments in the LED display. <vvv> must range between 0 and 255. <hhh> defines the length in pixels of the horizontal segments in the LED display. <hhh> must range between 0 and 639. <a> defines the width of the vertical and horizontal segments.

<a> must range between 0 to 2. When a=0 the segments are 1 pixel in width. When a=1 the segments are 3 pixels in width. When a=2 the segments are 5 pixels in width. Remember the larger and wider the segments, the longer it takes to display the LED font on the screen.

Since you can define all aspects of the LED font you can create scalable fonts from very small (.10 inch) to over 4 inches per digit. <b> defines the LED assignment number to the current LED parameters. <b> must range between 0-7. This allows you to define up to 8 active fonts at one time. When you display a number you only have to use the assignment parameter with the actual number to be displayed. Since all the information about the font is save in a buffer you can quickly update numbers of different sizes and characteristics on the screen.

Cell size for each number is defined by the largest segment width. Since the largest width currently availible is 5 pixels when you display a number with 1 pixel line width the starting location for that first horizontal segment is Y+3, X+4. The same holds true for the location of the left most vertical segments at X+3, Y+4. This means the maximum cell size for each number is hhh+10, vvv*2+4 for segment width of 5. When using a smaller segment widths the number will fit inside this maximum cell size. Spacing between numbers also varies with segment width. Spacing is determined as follows: When a=0 spacing = hhh+10. When a=1 spacing = hhh+16. When a=2 spacing = hhh+22. With this information you will be able to determine total height and width of your LED display. This will allow you to enclose the display with different borders.

Line Style

The following table shows the line style possiblities.

L	Line Style	Line
0	Solid	________________
1	Dotted/q2	....
2	Dotted/q3	. . . .
3	Dotted/q4	. . . .
4	Dashed/q4	- - - - - - - - - - - - - -
5	Dashed/Dotted/q4	_._._._._._._._._._._
6	Dashed/q6	_ _ _ _ _ _ _ _ _ _ _ _ _
7	Dashed/q7	_________________

LED Display

The *LDannnnnnnnnnnnnnnnn* command is use to display the LED style numeric fonts on the screen as defined by the previous command. Since up to 8 different active fonts can be defined the <a>=0-7 parameter assigns a definition buffer to the number to be displayed. The parameter <n> can be of any length up to 40 characters. The parameter <n> can consists of a number 0-9 or a period <.>, plus sign <+>, negative sign <-> or a space < >.

With these extra characters you can place a decimal point anywhere in the number. You can move the decimal point at anytime. The <+> or <-> sign can be used accentuate the direction of the number. The space can provide leading zero blanking to suppress zeros in front of the first significant number. Once the font has been defined you can increase the length of the number by adding more digits. If you decrease the length of the number add spaces to clear the digits not used.

Since you define the height and width of each number experiment width different combinations to see what effects you can create. In addition , the plus sign <+> has some special requirements. For a symmetrical effect the vertical segment must be center in the horizontal segment. This can only happen if both the vertical and horizontal lengths are odd. If one or both the lengths are even then the <+> will be offset by 1 pixel. The vertical and horizontal lengths of the <+> are the same as defined for the vertical and horizontal lengths of each segment. If the digit is symmetrical then the <+> will also be symmetrical. Also displays with aspect ratios of less then 1 will require the horizontal length to be longer than the vertical length to preserve symmetry.

Example: *L0200200101501710*

This defines a LED display at upper left hand location 20,20 with vertical segment length of 15 pixels and horizontal segments length of 17 pixels. The segment width is 3 pixels and it has been assigned to buffer #0. The display is composed of solid vectors and written to memory.

Example: *LD0-5.23*

This will display the number -5.23 according to the defined parameters of buffer 0. In this case the number will be displayed at location 20,20 with the above defined characteristics.

Once the LED definition is in place you can change the number by using the LD command. This will cause the display to be updated with the new number. The decimal place is a right hand decimal. It can moved to any location in the display at anytime. Remember you can control up to 8 LED displays on the screen simultaneously. If you need to control more you will have to define that LED display font before it is written to the screen.

Display Layer Control

There are 8 commands that control functions on the text layer (layer 1) and graphics-text layer (layer 2). The *E1-* command turns layer 1 off. The *E1+* command turns layer 1 on. The *E2-* command turns layer 2 off. The *E2+* command turns layer 2 on. These commands can be useful when building a complex screen. You can turn the layer off, then write to the screen, then turn the layer back on. This is sometimes a easier on your end user. The *E1** command will blink layer 1 at 2 htz. This will allow you highlight something on the screen and bring it to the end user's attention. The *E2** will blink layer 2 at 2 htz. The *E1%* will blink layer 1 at 16 htz. This gives the effect of a halftone. This effect can also be used to bring a section of the screen to the user's attention or indicated that the section is not in use at this time. The *E2%* command will blink layer 2 at 16 htz.

Plot Pixel

To generate a single pixel on the screen use the *PxxxyyyD* command. <xxx>, <yyy> define the location of the pixel. <xxx> must range between 0 and 639 with <yyy> must range between 0 and 255. <D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2.

Example: >*P0230770*

This command erases the pixel at location 23,77.

Plot Byte

The *Qchhhhhhhhhhhhhhhh* command is used to plot byte data directly to display RAM. The <c> parameter tells the system to either <c>= L start loading the byte data at the current cursor location or <c>=T translate the data then load the data to display memory. <hh> is the actual byte data is hexidecimal format. You can load up to 20 bytes of data per command line. Data is loaded by row. The first row is loaded, then the second and so on until all data has been stored.

Example: >*QL0A0B0C0D0E1F1010*

This command loads byte data starting at the current cursor location with the hex data 0A 0B 0C 0D 0E 1F 10 10.

Example:>*QT1010F01010100F*

This command first translates the data 10 10 F0 10 10 10 0F to 08 08 0F 08 08 08 F0 then it is loaded to the display RAM at the current cursor location. This command is useful when the bitmap data that is store in a file is inverted from the way the SED1330 stores the data on the screen.

Cursor location is set with the *Mxxxyyy* command. Once the cursor location is set all data is written sequentially to display RAM. You can chage the starting location at any time. This allows you to create a full screen bit map or to download a picture to the LCD.

Once the bitmap has been displayed you can store the screen with the *SSn* command. The screen can then be restored with the *SWn* command. The combination of these commands allows you to create a bit map the size of the entire screen.

Execution time of this command is very quick even though you are loading only 20 bytes of data at a time. Transmission time is the factor that creates the greatest delay in using this command. If you run at 9600 baud and are downloading a 320x240 screen which is 9600 bytes of data it would take approximately 22-23 seconds to download and execute the command 480 times. This may be unacceptable in certain circumstances.

By using a faster baud, 1.25Megabits per second, transmission time drops to around 0.43 seconds and execution time approximately the same time. With a total time of 0.86 seconds for the screen download this is acceptable in almost any circumstance. It should be noted that with transmission rates of 1.25MB/second you will get display jitter or snow as the MPI S133X and SED1330 are updating display RAM faster than the frame rate. To eliminate this turn the screen off with *D-* command, download the data, then turn the screen on with the *D+* command. This provides a very nice transition between screens.

Port Commands

There are four commands that are used to control Port D on the MPI S133X board. In previous versions Port D was exclusively used as an input port for keypad or IR touchscreen use. In version 3.0 the port can be defined as a general I/O port allowing you the control or read information from another device. When the system boots Port D is defined as an input port for keypad/touchscreen use.

Use the *JK* command to set Port D to keypad/touchscreen use. This is the default condition. Use the *JL* command to set Port D for general I/O use. When this command is executed Port D can be controlled with the *PIq* and *PWqbbb* commands. When *JL* is executed it turns off sections of code that would read data from an input device. The RTS input data is still available over the SPI interface.

Example: *JL*
Set Port D to general I/O port.

The *PIq* command will read the 8-bit data from the port and transmit that data over the serial port. <q> is the port number. On the MPI S133X board only port 3 is available for use. Therefore <q> is always 3. Remember this port has pull-up resistors on each bit. Therefore if you read the port when it is empty you get a reading of FF or 255.

Example: *PI3*

The port is read and data sent over the serial port in hexidecimal format hh. Since pull-up resistors are used on each bit you may have to compliment the data you read to use it in the correct format. If bit 0 is low level the data read from the port will be FE in hexidecimal. The compliment of this data is 01 hex. Depending on your use the compliment of the port data may be easier to use.

The *PWqbbb* command writes data to the port. Once again <q>=3 and <bbb> is the data you want to write to the port. <bbb> must range between 000 and 255 and is in decimal format. Remember that Port D has pull up resistors on each bit. This data is latched on the port until it is changed by another *PWqbbb* command or the port is redefined by the *PIq* or *JK* commands.

Example: *PWq015*

This command will write 15 in decimal or 0F in hexidecimal or 00001111 in binary to the port and the data is latched or held. Remember to use all 3 places when sending data.

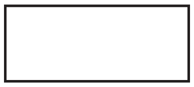
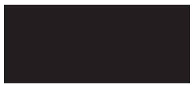
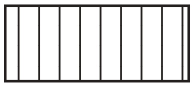
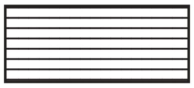
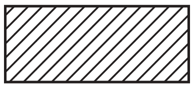
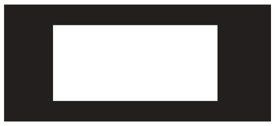
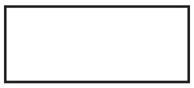
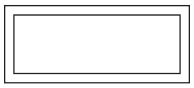
The use of the port commands will allow you greater flexability in your design and will relieve your cpu of cycle time and load.

Rectangle

The *BxxxyyyLDhhhwwwSTT* command will draw a rectangle with a number of different options. <xxx>, <yyy> parameters define the upper left hand corner of the rectangle. <L> defines the line style that will be used when the rectangle is plotted. <L> must range between 0 and 7. <D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <hhh> defines the height of the rectangle. <hhh> must range between 0 and 255. <www> defines the width of the rectangle. <hhh> must range between 0 and 639. <S> defines the rectangle style. <S> must range between 0 to 7. <TT> defines the number of pixels/line when the rectangle is filled with horizontal, vertical, of diagonal line. <TT>=00 means each line will be separated by 3 pixels. <TT> must range between 0 to 99.

Use the <TT> command to vary the distance between lines inside the rectangle to achieve the effect you are looking for. If you use horizontal lines then display the same rectangle with vertical lines on top of the first rectangle you can create a cross-hatched effect. Experiment with the different fills to create new fills. You can also you the <D>=2 command to create even more effects.

There are eight possible styles that the rectangle can display. The following table lists the different styles and display an example of what the rectangle would look like.

<S>	Rectangle Style	Example
0	Clear	
1	Solid	
2	Vertical lines/stripes	
3	Horizontal lines/stripes	
4	Diagonal lines/stripes	
5	Erase area defined by rectangle	
6	Erase area inside rectangle	
7	Double line border around rectangle	

Line Style

The following table shows the line style possiblities.

L	Line Style	Line
0	Solid	________________
1	Dotted/q2	....
2	Dotted/q3	. . . .
3	Dotted/q4	. . . .
4	Dashed/q4	- - - - - - - - - - - - - -
5	Dashed/Dotted/q4	_._._._._._._._._._._
6	Dashed/q6	_ _ _ _ _ _ _ _ _ _ _ _ _
7	Dashed/q7	____________________

Resistive TouchScreen Commands

This section explains the commands used to control and read data from the resistive touchscreen interface. This is an optional feature and the chip must be present for the commands to work. The MPIS133X uses 12 commands to control and display data from the MXB7846 resistive touchscreen controller.

Input Type

The *JA* command selects a 1x8 keypad as the input device. The *JB* command selects the IR touchscreen controller as the input device changing the input port to a 8-bit port. All the other J commands select or control the resistive touchscreen controller. If the RTS controller has been selected the the 8 bit port is unused and can be configured as a I/O port using the Port commands.

The *JC* command selects the RTS controller and sets it to the 8 bit mode. The *JD* command selects the RTS controller and sets it to the 12 bit mode. This is the default mode.

The *JE* command selects the RTS controller and sets it to the 10x8 keypad matrix mode with 8 bits of resolution. In this mode a virtual 10x8 keypad is overlayed on the screen. The keys are numbered 01-80. This mode emulates the IR touchscreen mode with a higher resolution. When a key is pressed the MPIS133X sends *Txx* with xx=<01-80> corresponding to the key that was pressed. All key events can be active in this mode allowing for subroutine execution when a key is pressed and released.

The *JF* command reads and transmits the current reading from the RTS. If the RTS is not active the reading will be 0 for X and 255 or 4095 for Y. The data sent is in the form Rxxxxyyyy. xxxx is either 8 or 12 bits of data in Hex or Decimal format. The same applies to Y.

The *JG* command turns the Pen enable mode off. When this mode is off data can only be obtained through the *JF* command. The *JH* command turns the Pen enable mode on. When the touchscreen is activated the MPIS133X automatically reads and transmits the data.

The *JI* command transmits RTS data over the serial port in hexidecimal format. The *JJ* command transmits RTS data over the serial port in decimal format.

The *JXaaabbb* command stores the calibration data needed for the 10x8 matrix mode. All resistive touchscreens read a voltage when touched. The value never goes to 0 or to the reference voltage. The calibration factors fix this problem and the factors are different for every touchscreen. aaa is the highest X value measured in the 8 bit mode when the screen is active. bbb is the lowest X value measured on the screen in the X mode. Set the screen to read in the 8 bit mode with pen enable and touch along the far left and right of the screen with a stylus not your finger to obtain the highest and lowest readings. The readings are in decimal format. Enter these reading with this command. The *JYaaabbb* command does the same thing for the Y calibration data. aaa is the highest Y value in decimal format and bbb is the lowest Y value in decimal format. Measure all along each border to obtain the lowest and highest values.

The JT1 command transmits the current temperature reading in degrees C on the RTS chip. The JT0 command transmits the current temperature reading in degrees F. The JV0 command reads and transmits the voltage reading on channel 0. The JV1 command reads and transmits the voltage reading on channel 1.

Screen Functions

There are 5 commands that clear the screen layers. The *CC* command clears the text screen which is layer 1. The *CE* command clears the text screen from the current cursor position to the end of the screen. The *CT* command clears the screen from the current cursor position to the top of the screen. The *CG* comand clears the graphics/text screen which is layer 2. The *CS* command clears the entire display layer 1 and layer 2.

Save and Restore LCD Display

There are 2 commands that control saving and restoring the current screen display on your LCD. The *SSn* command will save the screen at memory location n and the *SWn* command will restore the screen from memory location n. <n> must be either 0 or 1. 0 indicates the memory location is the same as the Bit Map storage area. 1 indicates the memory location is the same as the Window area. Saving a screen will use all of the memory area making it mutually exclusive with the Bit Map or Window areas. The n=0 or 1 applies only the systems with 32K of RAM. IMPORTANT RESTRICTIONS: **The *SSn* commands are for 32K systems only. There is not enough memory in the 8K system to use this command . LCD's with 320x240 pixels resolution or greater must use n=0. This command will use all of the bit map memory and some of the window memory with resolutions greater than or equal to 320x240 pixels. You can still use the window command but must start with window #4 minimum to avoid overwriting memory saved with the *SSn* command command. LCD's 640x240 cannot use function as it exceeds the amount of memory available. Remember the subroutine *CMDS* commands automatically adjust for the increase in memory usage. Be sure to start with at least Window #4 so you do not overwrite subroutine memory if you are using the subroutine function with LCD's of 320x240 or greater in size.**

Example: *SS1*

Result: The current screen image will be saved in the Window memory area.

Example: *SW1*

Result: The information saved in the Window memory area will be restored on the LCD screen.

The above example is for a system that has 32K of RAM. Not valid with an 8K RAM system.

Note: You can clear any memory area with the *SZn* and *SE* command. First use the *SZn* command to indicate the memory area you want to erase. Then execute the *SE* command to erase that area. See page 42 for information on the *SZn* and *SE* commands.

Select RAM Size

Command *Ra* is used to select amount of RAM your SED1330 controller uses. *R0* selects 8K of RAM and *R1* selects 32K of RAM. Use this command **only** when your LCD SED 1330 controller RAM size is different then the size specified in the table on page 9. You should not have to use this command unless you have used the user defined LCD parameter or have an updated SED1330 controller board with extra RAM. In this case you must tell the system exactly how much RAM is being used by the SED 1330 controller. This command will reinitialize the LCD, erase the screen and place the cursor at location (0,0). **Value is stored in EEPROM.**

Serial Port Baud Rate

This section is for experienced users only; users that need to run the MPI S133X at a baud rate setting other than the start up value of 9600 baud. After sending this command you must change the baud rate of the host CPU to receive the ready ">" character.

Use baud rates up to 115200 through the serial voltage converter chip. Above 115200 you should use the direct connect port J6. This port is a standard TTL level port (+5/Gnd) which can handle the higher baud rates. **Do not use this port with a converter chip that produces +/-12 volts. It will destroy the main chip.**

The *Yn* command will allow you to change the baud rate of the MPI S133X through software to improve the performance of your system. Improper use of this command will cause loss of the serial port connection. If this happens you must shut down the MPI S133X and bring all 8 bits of the key input to ground and reboot. This will load the default baud rate of 9600 for MPI S133X start-up. The following describes the baud rate associated with n.

0=9600	1=19200	2=38400	3=57600	4=115200
5=250000	6=375000	7=500000	8=625000	9=1250000.

The serial port baud rate is stored in EEPROM.

Serial Port Character Echo

There are 2 commands used to control character echo. This feature echoes each character as it is sent to the MPI S133X. This allows the user to verify the input and to use the MPI S133X in the terminal mode. If you want to increase you throughput then turn the echo off with *E0-.* To turn the echo on use *E0+.* Echo On is the default state when the system boots. This value is stored in EEPROM

Serial Port Software Flow Control

There are 2 commands used to control the serial port software flow. *R2* turns software control off and *R3* turns software control on.

Serial port software control uses the well established XON/XOFF control codes found in many commercial software packages. XON (H'11') is sent to the terminal program to indicate that the MPI S133X is ready for serial transmission. The MPI S133X sends XOFF (H'13') when the cpu cannot accept any more data. When the system is ready for more data it again sends XON and data transmission continues. Since each command takes a different time frame to execute, the use of XON/XOFF makes it easy for the controlling PC or CPU to know when to send data. By using a commercially acceptable method the MPI S133X can easily interface with programs like HyperTerminal and run at a very high speeds.

The ">" character is still available for serial port software control. By adding the XON/XOFF protocol this adds an additional method for determining when the MPI S133X is ready for data. This value is stored in EEPROM.

Subroutine Functions

The subroutine commands will allow you to predefine and load commands for fast execution by eliminating transmission time when code is executed multiples times. Up to 127 subroutines can be defined. In systems with 32K of RAM, 8K can be allocated for subroutine storage. In systems with 8K of RAM, approximately 3K of storage can be allocated. By storing the command on initialization for multiple screens you can quickly change between screens. All commands can be stored **except the subroutine commands.** The system is not recursive. If you execute a subroutine command within a subroutine the system will lock up. **NOTE: LCD's with 320x240 pixels or greater require a large amount of screen memory. The subroutine assign memory command automatically shifts storage up by the amount of extra memory required by the screen. You should use only *SZ1* command and store subroutines in window memory if you plan to use the *SSn* and *SWn* commands with LCD 320x240 or greater. Remember to execute a *SE* command after the *SZ1* command to activate this memory.**

Assign Subroutine Memory

The *SZn* command assigns the storage location for subroutine data. <n>=0 will assign the data storage area to the same area as the user defined bit maps. When <n>=1 the assigned storage area will be the same as the window area. When using an 8K RAM system all three areas are the same and therefore the same restrictions outlined on page 25 apply. n=0 is required with the 8K system.

Display Subroutine Memory

The *SM* command will send the number of free memory bytes left in the subroutine memory area to the user through the serial port.

Example: *SM*

Result: 8192 Bytes Free

When using a system with 32K of RAM and executing the *SM* command after the system initializes the MPI S133X will return 8192 bytes free, the maximum available space for subroutine storage. The command will return less than 8192 bytes free on systems with LCD' of 320x240 pixels or more and 32K of RAM. This memory adjustment is to compensate for the extra memory required of screen usage and the *SSn* and *SWn* commands.

Erase Subroutine Memory

The *SE* command erases the current assigned memory and sets all index values to 0. This command is used any time you want to erase the subroutines stored in memory and store new ones.

Load Subroutine Memory

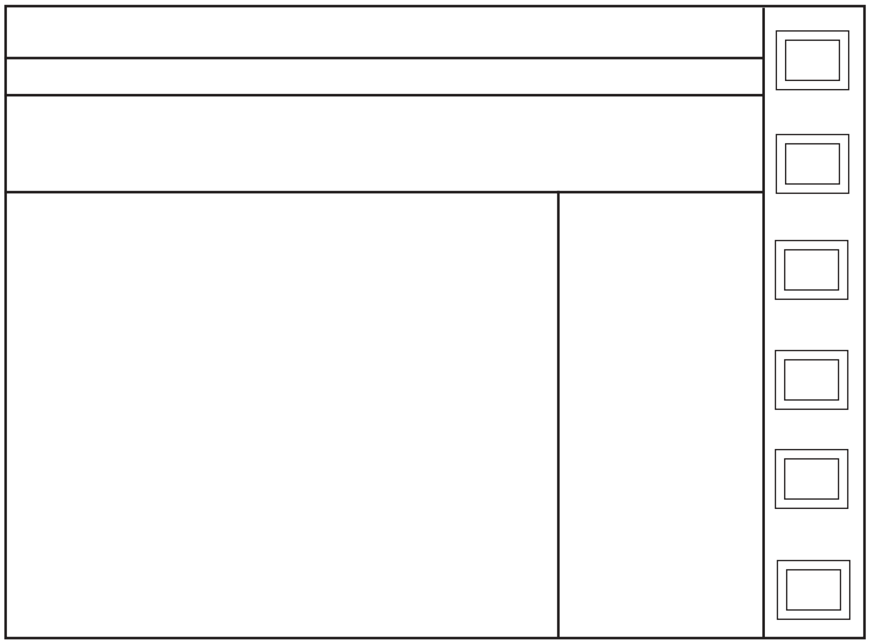
The *SLaaaccccccccccccccccc* command controls the loading of command data into memory. Before this command can be execute the subroutine memory must be assigned and erased. Each subroutine must be loaded completely before you define another subroutine. Sub-routines are save sequentially and once you have define the next subroutine you can not add to the previous subroutine.

The <aaa> parameter defines the subroutine number. Each subroutine must have a unique number associated with it. <aaa> can range from 00-127. Up to 40 characters can be sent with each command. Each <c> is an ASCII character of a MPI S133X display command. Commands must be sent 1 command at a time. To load 3 commands you must send the SL command 3 times. You must complete loading subroutine 00 before you start with subroutine 01.

Code Example: Display

```
SL000B000000011993190
SL000H00001401000281
SL000H00002601000281
SL000H00006001000281
SL000V28100001000199
SL000V18006001000139
SL000KR6
```

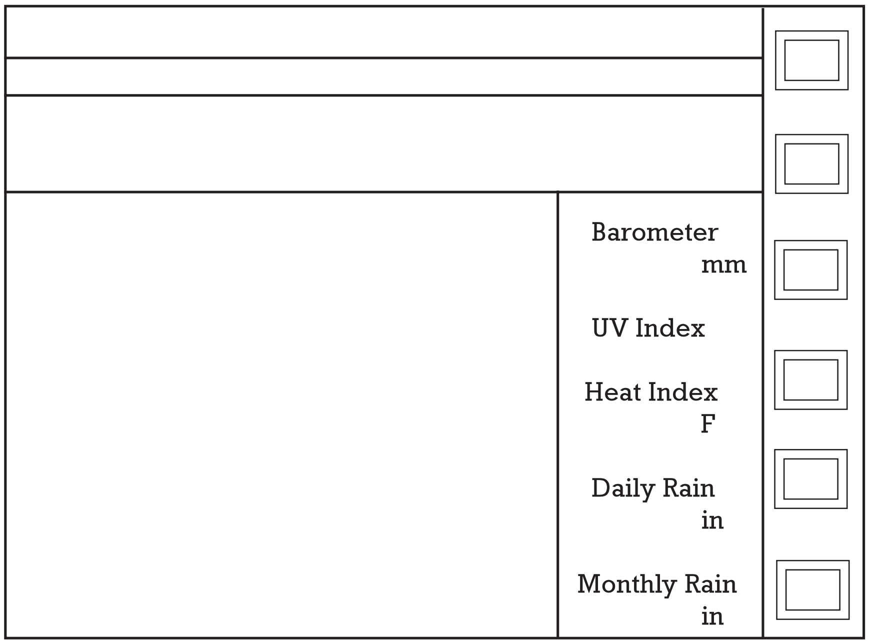
Then type : *SX000*



In this example to subroutine will draw the diagram above each time it is executed with the *SX00* 0command.

Example:

```
SL001M200062
SL001SR3Barometer
SL001M260075
SL001SR3mm
SL001M204088
SL001SR3UV Index
SL001M192114
SL001SR3Heat Index
SL001M260126
SL001SR3F
SL001M192140
SL001SR3Daily Rain
SL001M260152
SL001SR3in
SL001M184166
SL001SR3Monthly Rain
SL001M260178
SL001SR3in
```



Then type :*SX001*

Executing the second subroutine will place the text on the screen as shown in the example at the bottom of the previous page. Subroutines can be any length up to the maximum memory you have available. Subroutines must be reloaded on power up everytime since the SED1335 display memory is volatile. Subroutines save you transmission time and your CPU program memory.

Execute Subroutine Program

The *SXaaa* command executes the subroutine program stored in memory. <aaa> is the subroutine number to be executed. <aaa> must range between 00-127. See examples on the previous page.

Text Display

There are 2 commands that control the display of text on the screen. The *Taccccccccccccccccc* command controls layer1 and displays the built-in SED1330 font on that layer. The *Sabcccccccccccccccc* command controls layer 2 and displays the MPI S133X built-in fonts on the graphics/text layer which is layer 2.

Text Display on Layer 1

Use the *Taccccccccccccccccc* command to display text on the first layer. The <a> parameter defines the direction the cursor will move after the character is displayed. This allows you display data in any of four different directions with one simple command. When <a>=R the cursor will move to the right. When <a>=L the cursor will move to the left. When <a>=U the cursor will move up. When <a>=d the cursor will move down. Use the *Lxxxyyy* command to set the initial cursor position. This command can send up to 40 characters per command line.

Example: >TDHello

This command will display the following text at the cursor position defined by the *Lxxxyyy* command.

```
H
e
l
l
o
```

Example: >TRHello

This command will display the following text at the cursor position defined by the *Lxxxyyy* command.

```
Hello
```

Text Display on Layer 2

Use the *Sabcccccccccccccccc* command to display text on the second layer. The <a> parameter defines the direction the cursor will move after the character is displayed. This allows you display data in any of two different directions with one simple command. When <a>=R the cursor will move to the right. When <a>=V the cursor will move in the vertical direction. Use the <b> command the change the attributes associated with the text displayed. <b>=0,3,6 will display normal text in the replacement mode, overlay mode and exclusively or'ed mode respectively. <b>=1,4,7 will display the text in reverse video in the replacement mode, overlay mode and exclusively or'ed mode respectively. <b>=2,5,8 will display underlined text in the replacement mode, overlay mode and exclusively or'ed mode respectively. You can mix and match text atributes. Remember the text attributes only apply to the text currently being printed on the screen with the current command. Up to 40 characters can be sent with each command. Each <c> is an ASCII character to be displayed on the screen. Remember use the *Mxxxyyy* command to set the initial cursor position. The *Fa* font select command defines the current default font. Font 1 is the default font on power up.

Example: >SR5Hello

Hello

This command will display Hello underlined in the overlay mode.

Timer

The MPI S133X will generate a real time clock/timer using on board timers linked to the cpu timing crystal. The clock/timer is accurate to approximately 15 seconds a month but is volatile like the clock in your car and must be set when power is turned off. In this section the timer portion of the clock will be explained.

The *CK3* command must be used before this command. This selects the Timer mode and stores the variables in the correct registers.

The Timer is independent of the clock. Each timepiece has its own registers which are updated independently of each other. The only restriction is that they share they same display routine so only one either the clock or the timer can be actively displayed and updated automatically on the screen at any given time. Both will continue to run in the background and can be sent over the serial port and read at any time.

Use the *CxxxyyyLDvvvhhhFTW* command to define the timer on the screen. <xxx>, <yyy> defines the upper left hand corner of the clock display. <L> is reserved and should be 0. <D> is the text attribute for the timer. This is the same attribute for the text display command *S* on layer 2. See that section for more information on the text attribute. <D> must range between 0 and 8. <vvv>, <hhh> are reserved and should be set to 000,000. <F> indicates the font to be used when the timer is displayed. <F> must range between 0 and 3. <T> is reserved and should be set to 0. <W> is reserved and should be set to 0.

Setting the Timer

The MPI S133X sets the timer with the same command as the clock. The *CK3* command is used to select the timer mode and must be used before the timer is defined or the timer is set or read. *CKSThhmmss* is used to set the timer. The timer must be set in a 24 hour format. <hh> sets the hours parameter. The value must range between 00 and 23. <mm> sets the minutes parameter. The value must range between 00 and 59. <ss> sets the seconds parameter. The value must range between 00 and 59. This is the value the timer is reset to when the reset command *CK8* is sent. The timer is set to stopwatch mode by *CK+* which is the default mode. In this mode the clock will count in the up direction. To set the timer in the countdown mode use *CK-* command.

Example: > *CK-*

CKST001500

In this example the timer is set to countdown mode with a preset value of 15 minutes. When the reset command *CK8* is sent the timer will reset to 15 minutes.

Example: > *CK+*

CKST000000

In this example the timer is set to stopwatch mode with a preset value of 00:00:00. When the reset command *CK8* is sent the timer will reset to 00:00:00.

Reading the Timer

The MPI S133X uses the *CKRT* command to send current timer information to the host computer through the serial port. Remember to send the *CK3* command first to select the timer mode before the read command. The time is sent in the following format HH:MM:SS .

Controlling the Timer

The MPI S133X uses eight commands to control the function of the timer. *CK3* command selects the timer mode and *CK2* selects the clock mode. Remember to use the appropriate command before you define, read or set the timer. *CK5* selects 1 second update period and should be selected before activating the timer. *CK#* suspends the display of the timer. This is the default mode and if you want to see the timer on the screen you need to resume the display with *CK**. The *CK#* command is very useful when running the timer in background and running the clock display on the screen. Sending this command keep the timer display from appearing on the screen. *CK1* can then be used to resume the display of the clock on the screen . Then issue the *CK3* command to select the timer mode. You can now read the timer over the serial port while the clock updates on the LCD screen. The CK* will activate the timer display and stop the clock display.

CK+ defines the timer as a stopwatch and counts in the up direction . This mode is the default mode. *CK-* defines the clock as a countdown timer. *CK8* resets the timer to the value defined by the *CKSThhmmss* command.

Remember the *CK5* command sets the automatic update of the clock to 1 second. This must be set or the seconds will not update on the timer. *CK6* starts the Timer function. *CK7* stops the Timer Function.

Example:
```
CK5
CK1
CK#
CK6
CK3
CKRT
```

In this example the code sets the timer to update every second. Starts the real time clock on the screen. Turns the timer display off. Starts the time. Selects the timer mode and reads the timer value over the serial port while the clock continues to run on the screen.

Timer Alarm

The Timer is equipped with an alarm function when the timer is in the countdown mode. This alarm will execute a subroutine when it reaches 00:00:00 on the timer. The subroutine could send you the time as an indicator of the time-out over the serial port or execute a screen change or any other combination of functions that exist in the MPI S133X command set.

Three commands are used to control the alarm function. *CK@* turns the alarm on. The next time the timer reaches 00:00:00 in the countdown mode the alarm will be activated. *CK#* turns the alarm off. This function will manually turn the alarm off. When the alarm is active and the timer reaches 00:00:00 besides executing a subroutine the program will execute *CK#* and turn the alarm off so it is only executed once. The Timer is stopped. Also the display is put in the suspended mode deactivating the update of the timer. Then the subroutine is executed.

CKSXnn command loads the subroutine number that is to be executed when the alarm is set. <nn> is the subroutine number and must range between 00-63.

To set the timer alarm you need to load the initial value in the down counter with the *CKSThhmmss* command. Next load a subroutine with the commands you want executed. See the subroutine section for more information. Load the subroutine number with the *CKSXnn* command. Activate the alarm with the *CK@* command. Turn the display on and start the timer. Set the timer up as a countdown timer with the *CK-* command.

```
Example1:   SL01D-
            CK3
            CKST000500
            CKSX01
            CK@
            CK-
            CK*
            CK5
            CK8
            CK6
```

```
Example 2:  SL01CK3
            SL01CKRT
            SL01CK2
```

In example 1 above the code will turn the display off after 5 minutes. Make sure you turn the timer display on with the *CK** command and that the interval is set to 1 second with the *CK5* command.

In the second example the subroutine has been changed to send 00:00:00 to the host over the serial port to indicate that an alarm has occurred. The host computer can then act on the alarm.

Vector

To generate a vector use the *ZxxxyyyLDaaabbb* command. <xxx>, <yyy> define the starting point of the vector. <L> defines the line style that will be used when the object is plotted. <L> must range between 0 and 7. <D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <aaa> is the X coordinate of the ending point of the vector. <aaa> must be between 0 and 639. <bbb> is the Y coordinate of the ending point of the vector. <bbb> must range between 0 and 255.

Line Style

The following table shows the line style possiblities.

L	Line Style	Line
0	Solid	________________
1	Dotted/q2	....
2	Dotted/q3	. . . .
3	Dotted/q4	. . . .
4	Dashed/q4	- - - - - - - - - - - - - -
5	Dashed/Dotted/q4	_._._._._._._._._._._
6	Dashed/q6	_ _ _ _ _ _ _ _ _ _ _ _ _
7	Dashed/q7	_________________

Example: >Z01002301055111

This command draws a vector from 10,23 to 55,111. This vector is drawn as a solid line.

Vertical Line

To generate Vertical lines use the *VxxxyyyLDdddwww* command. <xxx>, <yyy> define the starting point of the line. <L> defines the line style that will be used when the object is plotted. <L> must range between 0 and 7. <D> tells the system to either clear the pixel, set the pixel, or exclusively or the pixel with the value in memory. <D> must range between 0 and 2. <ddd> is a dummy variable. <ddd> should be 000. <www> defines the length of the line. <www> must range between 0 and 255.

Line Style

The following table shows the line style possiblities.

L	Line Style	Line
0	Solid	————————————
1	Dotted/q2	....
2	Dotted/q3	. . . .
3	Dotted/q4	. . . .
4	Dashed/q4	- - - - - - - - - - - - - -
5	Dashed/Dotted/q4	_._._._._._._._._._._
6	Dashed/q6	_ _ _ _ _ _ _ _ _ _ _ _ _
7	Dashed/q7	____________________

Video Control

There are two commands in the video control section. These commands control layer 2 only. The *R+* will reverse video the entire display on layer 2. The *R-* will return the display to normal.

It is very important to remember when the video is reversed you need to change the logic when setting and clearing pixels in each of the commands that has a *D* parameter. If you set a pixel *D=1* in normal video you must clear the pixel in reverse video to get the same effect (*D=0*). To simplify the process you can use *D=2*. This will properly set the pixel on a normal screen as well as a reverse video screen, If you write the same command to the screen again with *D=2* then it will erase the information you just wrote to the screen. Some negative image LCD screens need the R+ command when all data written using this negative logic. Commands that do not have a *D* parameter will automatically change the logic of pixel setting when the screen is in reverse video mode.

Windows

The *WxxxyyyQn* command will simplify window control on the display. The command will save the underlying screen data before you open a window. The system will restore the screen data after the window is closed. In addition it will draw the window frame in a double line configuration. You only have to fill the window with you own information. <xxx>, <yyy> defines the upper left hand corner of the rectangle that defines the window area in bits when <Q>=D or C. When <Q>=D then window is opened or displayed on the screen at the <xxx>, <yyy> coordinates. When <Q>=C the window is closed on the screen and the underlying screen data is restored at the coordinates <xxx>, <yyy>. The <xxx>, <yyy> parameters define the size of the window in bits where the <Q> parameter = A or R. When the <Q> parameter = A the system allocates (<xxx>*<yyy>)/8 bytes of memory. When the <Q> parameter = R the system will release (<xxx>*<yyy>)/8 bytes and clear the memory internally. Then window assignment number <n> defines the handle for the window. Up to 8 windows can be defined by the system with 32K of RAM on your SED 1330. **Only 3 windows can be defined by a system with 8K of RAM. (See page 25 for more explanation.)** Each window is allocated 1024 bytes. This will allow a window of 128x64 bits to be opened for example.

Example: >W080040A0

This will allocate a window that is 80x40 pixels in size and assign that window the assignment number 0.

Example:> W020020D0

This will display window 0 at 20,20 on the screen.

It is possible to allocate more memory per window block by reducing the number of total windows that can exist. If you reduce the total number to four windows, then each window can be 2048 bytes in size i.e. 128x128 bits. If you choose to create larger windows you must keep track of where they are stored. In this case window 0 is stored in <n>=0 ,1. Window 2 will be the next availible window and it is stored at <n>=2 ,3. Remember to reference window 0 use <n>=0 even though is occupies 2 1k memory blocks. To reference the next window use <n>=2. If you need 2 very large windows then each window can be 4096 bytes in size i.e. 255x128 bits. The <xxx> parameter is limited to 255, the maximum width of a window. In this case window 0 will be stored at <n>=0,1,2,3 and window 4 is the next availible window stored at <n>=4,5,6,7. To reference the first window use <n>=0. The reference the next window use <n>=4.