

Advanced Direct Drive LCD

Description: Using the Renesas API to create real world LCD products

Objectives

Create a Slider and create an anti-aliased font

Prerequisites: Completed Lab1 and Lab2 and Lab3

Lab Materials:

Please verify you have the following materials at your lab station.

- PC with Windows XP or Vista
- E10A Debugger
- Renesas H8S/2378 LCD Kit and LCD Panel
- HEW Version 4.06 or newer
- H8S Tools Version 6.20 or newer
- Printed Copy of the Lab Material
- Lab4 Workspace (created by unzipping DirectLCD_Labs.exe)

Skill Level : Intermediate to Advanced knowledge of C programming. Intermediate knowledge of Renesas MCU's and Basic understanding of LCD Displays

Time to Complete Lab1: 30 Minutes
Time to Complete Lab2: 20 Minutes
Time to Complete Lab3: 20 Minutes
Time to Complete Lab4: 20 Minutes

Lab Sections

- | | | |
|----------|--|-----------------------------------|
| 1 | Setting up the Hardware, Compiling the Workspace and Changing background images | Time to complete task: 30 minutes |
| 2 | Presenting Multiple Bitmaps and Animation | Time to complete task: 20 minutes |
| 3 | Adding a Button and Watching the Touchscreen | Time to complete task: 20 minutes |
| 4 | Adding a Slider and an Anti-Aliased Font | Time to complete task: 20 minutes |
-
- | | | |
|----------|---|-----------------------------------|
| 4 | Adding a Slider and an Anti-Aliased Font | Time to complete task: 20 minutes |
|----------|---|-----------------------------------|

Overview:

In this section we will add an additional slider and create an anti-aliased font.

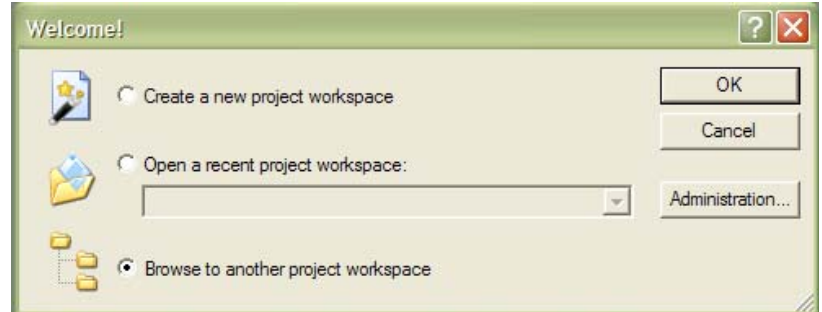
Lab4 – Section 1) Loading and Compiling the Program

1. Open HEW (High Performance Embedded Workshop) by double clicking the icon on the desktop OR under Start->Renesas->High Performance Embedded Workshop in the Renesas Program Group



2. On the welcome screen choose 'Browse to another project workspace', then click <OK>

Or if HEW is already open, go to File -> Open Workspace

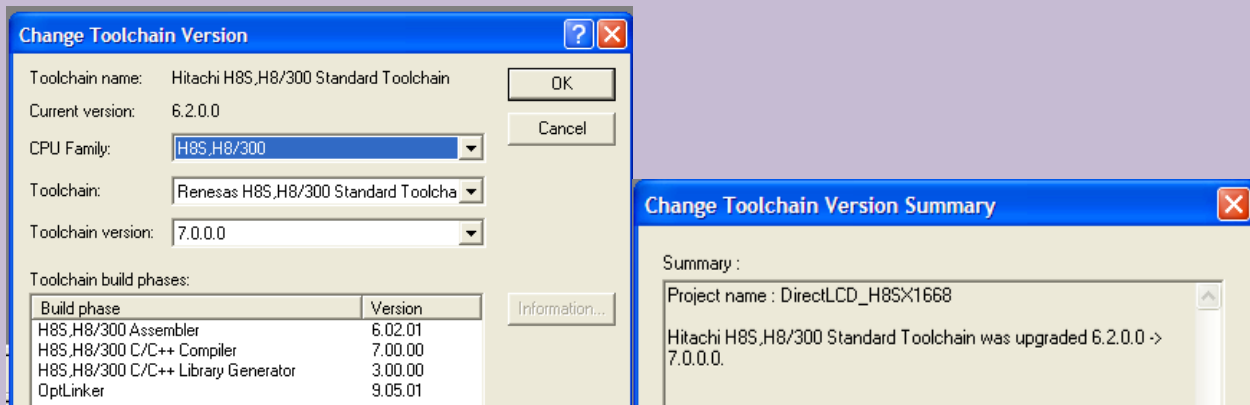


3. Browse to: C:\Workspace\DirectLCD\Lab4\Lab4.hws and <Select>.

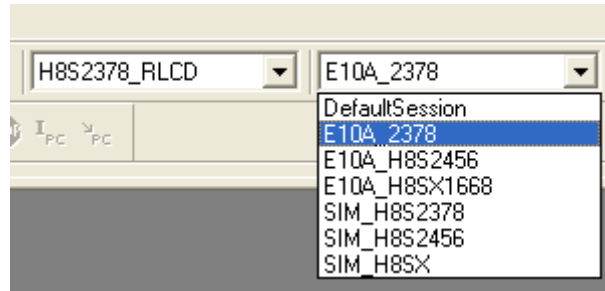
By Default – HEW will use workspaces under the C:\workspace\ directory on your PC – although it has the flexibility to use workspaces located anywhere on your machine.

If you are not using a preconfigured Renesas Laptop:

If the saved project does not match the compiler version that the project was created with (which is likely) – the project will be updated to the new version. Note that this conversion is one-way and you cannot go backwards to the old version. Click through the dialogs as necessary to complete the conversion.




- Click the debug session pull down and make sure <E10A_2378> is selected as shown below.



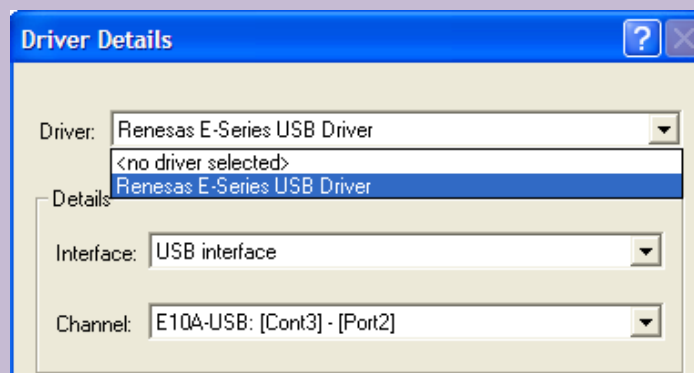
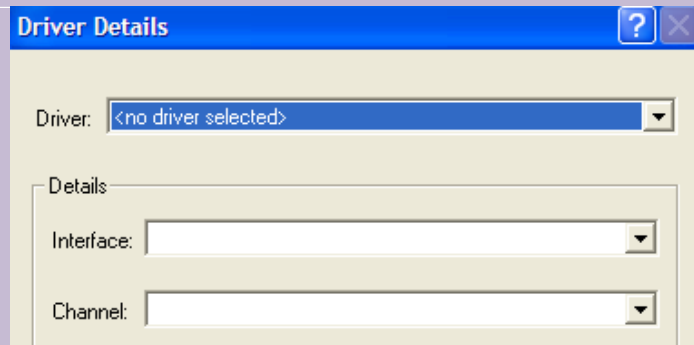
Note: The other session choices configure the project to connect to different Target Micros through an E10A or for debugging using the built in Simulator (which does not simulate an actual target LCD).

- As detailed in **Lab1 – Section 1) Setting up the Hardware** - Check that your demo board has power, and that the E10A is properly connected to the target board and a USB port of your PC.

6. Click the CONNECT icon  which will connect your PC to the E10A debugger as well as your target micro.

If you are not using a preconfigured Renesas Laptop:

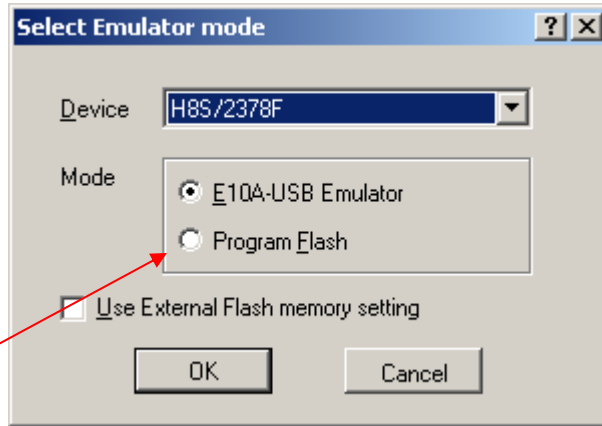
If you see a screen like the following:



Attempt to select 'Renesas E-Series USB Driver' from the 'Driver' pull down menu. If you are not able to select the E10A (Renesas E-Series USB Driver) from the driver pull down menu - you will need install the E10A drivers on your system. The drivers are available by going to www.renesas.com -> Dev Tool Tab -> Emulation and Debugging -> On-Chip Debuggers (E10A-USB) -> Click on link Labeled E10A-USB -> Downloads -> Pick E10A-USB Emulator – look for version for H8S in the list -> Enter Renesas ID and Password, agree to the disclaimer, download and install.

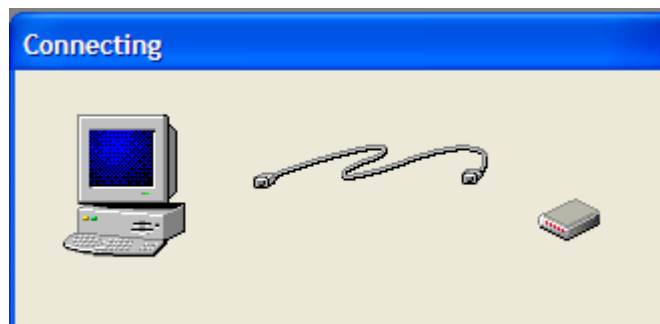
After you make sure you have installed the E10A drivers and selecting the Renesas E-Series USB Driver does not allow you to connect, you may find that unplugging and re-plugging the USB cable – or actually moving the USB cable to a different port on your machine may allow the Emulator to see the Driver. You may also reach a mode where the only solution is to close HEW and restart and reconnect to the emulator.

7. You will see the following Dialog box. Make sure your target (in this lab the H8S/2378F) is selected as the device and the settings are as shown, then click <OK>:

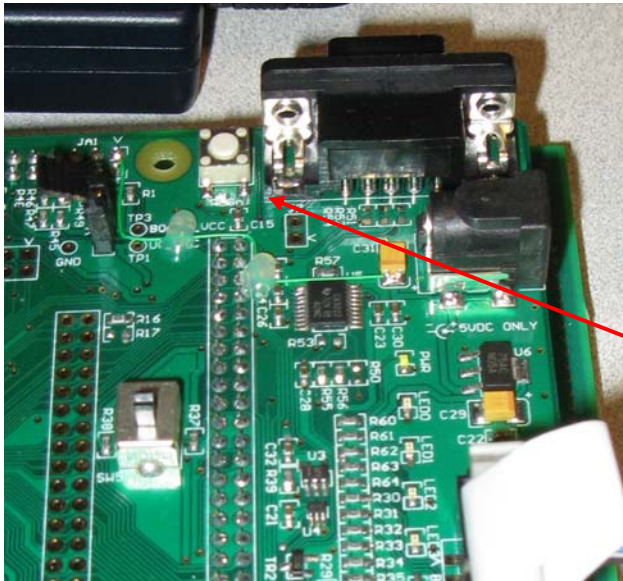


Note: The Program Flash Option allows you to program the micro's flash, however once finished, HEW will not be able to connect to the target through the debugger. Typically, you would use this option to test your final code.

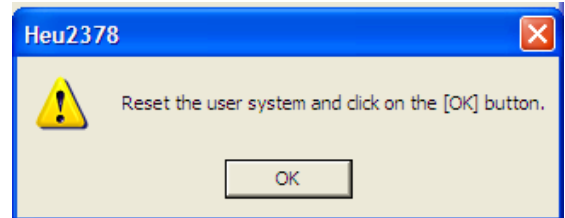
8. You will briefly see the following window as the PC connects to the E10A debugger.



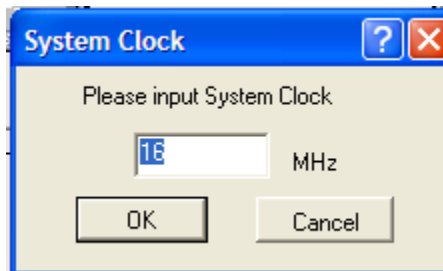
9. When prompted, press the Reset Switch on the board, then click <OK>. **Note:** this is located at the upper right of the demo board next to the RS232 connector and should **not** be confused with the BOOT switch on the lower right corner of the board.



Reset Switch

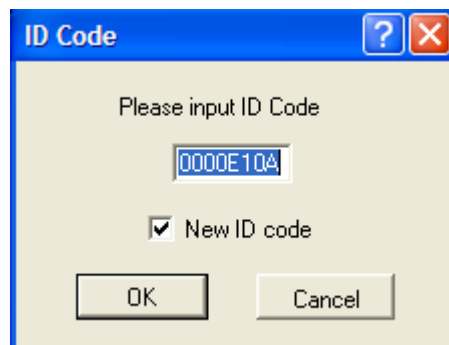


10. Enter the System Clock – which for the H8S/2378F demo needs to be set to 16 mhz (the board crystal frequency):



Note: This allows the emulator to synch up to the target for communications

11. Input your ID Code which for the training is 0000E10A:

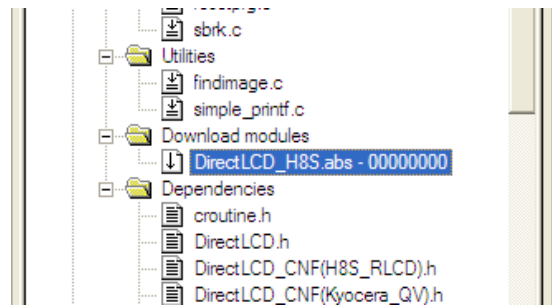


Note: This code allows you to secure access to the flash. Anyone trying to connect to this micro must provide this code or erase all flash before they can connect.

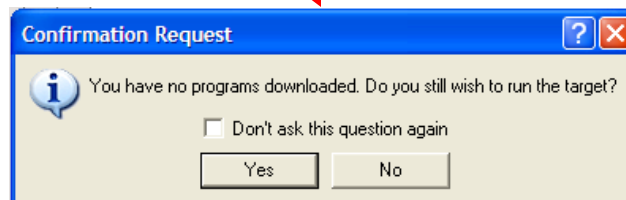
12. Click the BUILD ALL icon 


 Your code should build with No Errors. If you receive an L1500 Stack Warning, please ignore.

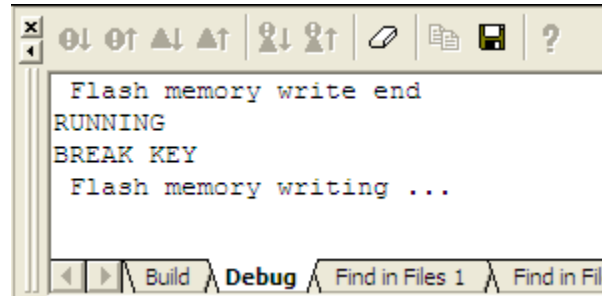
13. Your compiled code must be downloaded into the target. To do this, locate the download module in the workspace window (ends in .abs) and double click on it. **Note:** When you see an empty box to the left of the file name, it has not been downloaded to the target. A box with an arrow in it signifies it has been downloaded. This step may be done automatically depending on your project settings.




Note: You will receive this dialog box if you attempt to run the micro with no code loaded.



14. Click on the RESET GO icon  to execute the code – but be aware that there will be an ~10 second delay while the code is downloaded into the target, during which the target LCD screen will remain blank. You will see the following information in the Debug window while the Flash is being downloaded:

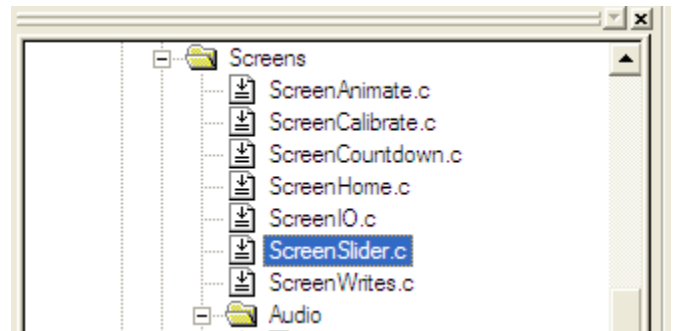


You will see a Renesas Title screen, followed after a 5 second delay with a screen with 6 touch buttons on it.

15. Press the <Slider> Button on the touchscreen.
16. Press the Slider Button on the touchscreen. You should see 3 slider bars (Red, Green and Blue).
17. Click the STOP icon  to stop the processor and allow the program to be recompiled.

Lab4 – Section 2) Adding a Slider

18. Next we will modify the code to add a new button. In your project Explorer – open the file ScreenSlider.c by double clicking on it.



19. At or near line 53 – uncomment (by removing the //) the function prototype:
`// static void SliderNew(ICON_type const *pS, EVENT_MSG const *pMsg);`

What are we doing?

We are adding a C function prototype for the SliderNew() subroutine.

20. At or near line 57 – change the type SLIDERSTATE_type from State[3]; to State[4]; This makes 4 sliders instead of 3.

21. At or near line 71 – uncomment (by removing //) the array definition:

```
// { &pBMP_Slider, T_SchemeRed, SliderNew, SX(0.050), SY(0.050) },
```

What are we doing?

By adding a 4th item to the ICON_type Icons[] array, we are defining a new slider (&pBMP_Slider), use the red colorizing scheme (T_SchemeRed), call the function “SliderNew” when touched and be located 5% over from the left of the X axis (0.050) and 5% up from the bottom of the screen on the Y axis (0.050).

20. At lines 192 to 197, uncomment (by removing the /* before and */ after) the new subroutine that will be called when a button event is detected.

```
/*
static void SliderNew(ICON_type const *pS, EVENT_MSG const *pMsg)
{
    Sliders(pS, pMsg, 3 );
}
*/
```

What are we doing?

This routine is called every time a matching slider event is detected. It calls the function Sliders(pS,pMsg,3); which actually draws the slider.

22. Click the BUILD ALL icon  to rebuild the project

23. Click the RESET GO icon  to RUN the code. Don't forget – the Flash memory write takes ~10 seconds.

24. Press the <Slider> button. You should see a new slider at the bottom of the screen. It will have the same behavior as the other sliders – but it will not affect the color box.

25. Click the STOP icon  to stop the processor.

Lab4 – Section 3) Generating and Using an Anti-Aliased Font

26. What is an anti-aliased Font? In the picture below, note the standard font on the left and the same font and size rendered using anti-aliasing on the right.

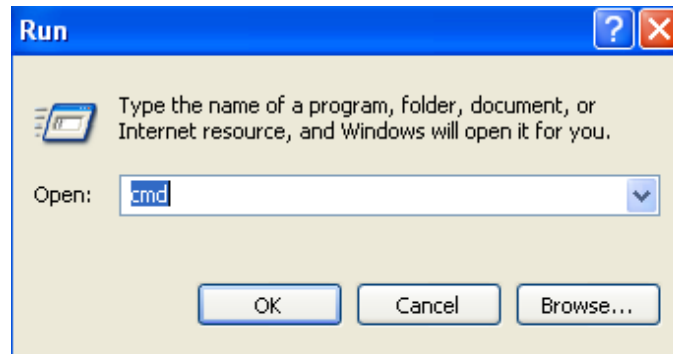


27. Click the RESET GO icon  to RUN the code.

28. Click the <Countdown> button. Note the large font on the left is a **NON** anti-aliased font (called Impact). Note how “jagged” the number edges look.

29. Click the STOP icon  to stop the processor.

30. Next we are going to create a replacement anti-aliased Arial font. Open a command window by clicking Start -> Run -> CMD



31. In the Command window - navigate to C:\Workspace\DirectLCD\Lab4\Resources directory.
- Note:** Use `cd \` to move to the C:\ directory (the root) and '`cd Directoryname`' to change to a new directory.
- Note:** You can move to the correct directory in one command once you are at the root directory by typing `C:\Workspace\DirectLCD\Lab4\Resources`. Type `DIR` if you aren't sure what directory the Command Window currently points to or what subdirectories are in the current directory.

```

C:\ C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings>cd \
C:\>cd Documents and Settings
C:\Documents and Settings>
C:\Documents and Settings>cd \
C:\>cd workspace\directlcd\lab4\resources
C:\Workspace\DirectLCD\Lab4\Resources>_
    
```

32. In the Command window - type the command:

```
..\gapi_font_gen.exe -R 1 -i ..\numbers.uni -p 80 -o fontTimer C:\windows\fonts\arial.ttf
```

OR

```
..\fonttest.bat
```

What are we doing?

The batch file fonttest.bat actually contains the same command as the same as the line above – it just is easier to type. The command goes up a directory (..) from the Resources directory to the \Lab4 directory and runs gapi_font_gen.exe – then it creates the fontTimer.efnt file using a windows font (arial in this case) stored in the windows fonts folder. Note that it creates a font 80 characters high. It also uses numbers.uni as input (-i). Numbers.uni contains the ascii characters to output (-o) to the fontTimer.efnt file.

Note: The gapi_font_gen.exe program can create any size font from any .ttf (truetype) font.

33. Click the BUILD ALL icon  to rebuild the project

34. Click the RESET GO icon  to RUN the code. Don't forget – the Flash memory write takes ~10 seconds.

35. Press the <Countdown> button. Note the 80 pixel tall anti-aliased numbers, which are now rendered in the Arial font rather than Impact font.

Lab 4 Wrapup:

In this 30 minute lab we have:

- 1) Added a new slider
- 2) Created a new anti-aliased font

This lab has completed the review of the GAPI tutorials. You should be able to take what you have learned in Labs1-4 and create a user interface for use in a basic menuing system.