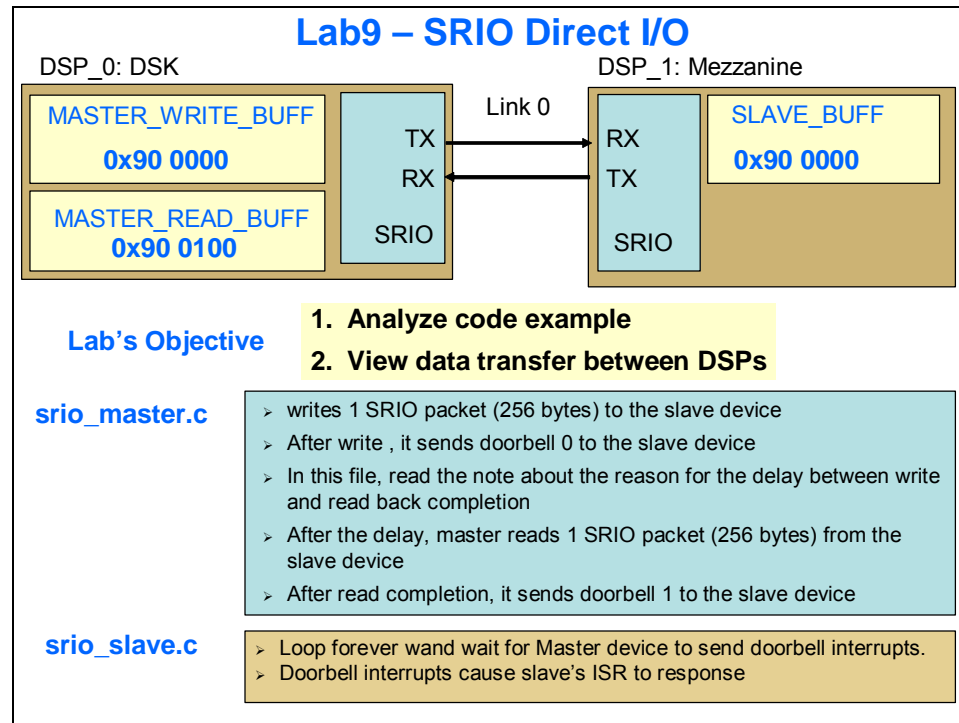


## Lab 9: Using C6455 SRIO



### Lab Overview:

The goal of this lab is for you to be familiarized with the process of using the Serial Rapid Input Output (SRIO) for C6455. You will learn to use Direct IO to access the SRIO. To gain this basic knowledge you will:

- Observe and analyze the master-slave example
- Observe the Direct IO programming method used with CSL and SRIO
- Run some tests

## Lab 9 Procedure

### Part 1 – Running the Code

You will be running code on the DSK and on the Mezzanine card. It does not matter which one is the master device and which one is the slave device. In the lab procedure, we pick CPU\_0 (the DSK) as the master and the CPU\_1 (the Mezzanine card) as the slave. This is intended so that in the future, we will port this code example to the audio project we used in previous labs.

#### 1. **Connect C6455 DSK to the Mezzanine EVM card (with power disconnected).**

- Quit CCS
- Remove power from the DSK
- Connect the Mezzanine card.
- Plug back in the power connection to the DSK.

---

Note: if the Mezzanine card does not work properly, make sure the card is seated firmly in the socket. If it is, you may need to pull it out slightly to get it to work. We've seen this happen on multiple boards. Spectrum Digital is aware of the issue and is working to resolve this.

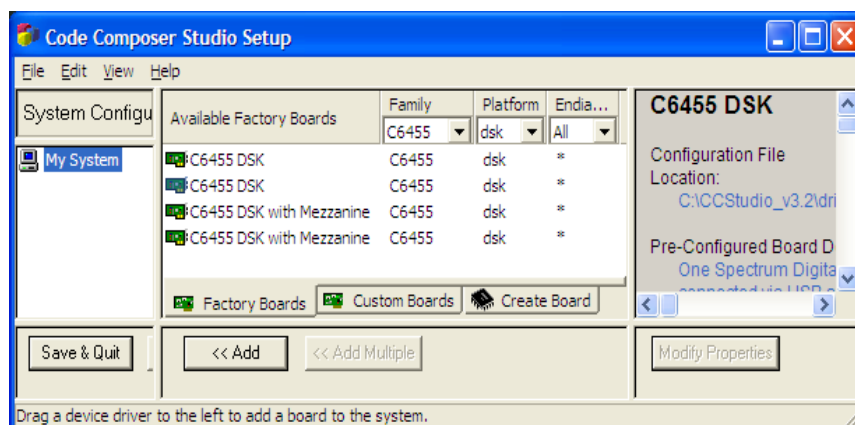
---

## 2. Set up the Parallel Debug Manager

- **Start the CCS Setup utility using its desktop icon.**

Be aware there are two CCS icons, one for setup, and the other to start the CCS application. You want the **Setup CCSStudio v3.2** icon.

When you open CC\_Setup, you should see a screen similar to this:

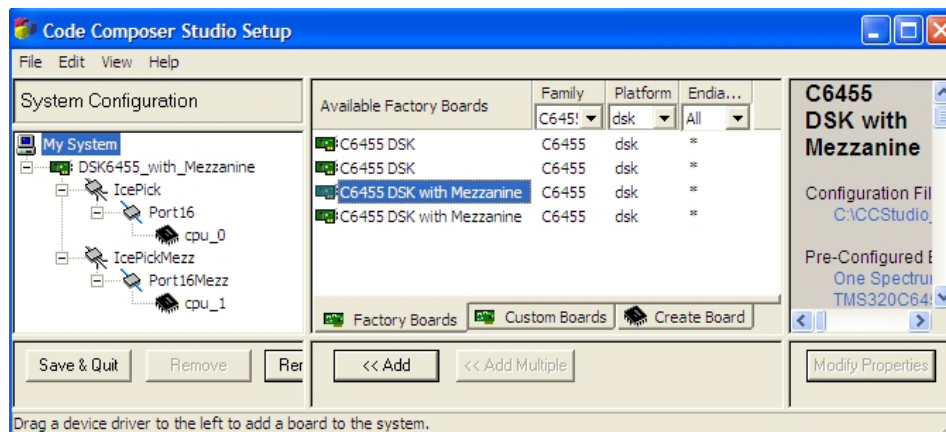


- **Clear any old system configurations.**

If there are any boards/simulators listed under *My System* under *System Configuration*, click the **Remove All** button to clear the configuration.

- **Use the filters to select the correct Factory Board.**

To the right of Available Factory Boards, you will see 3 filters (Family, Platform and Endianness). Use the drop down boxes and make the selections shown to select the correct board.



- **Add the proper factory board.**

Click on the **“C6455 DSK with Mezzanine”** and select << Add. This board should now show up under *My System*.

- **Select Save and Quit.**
- **When prompted to start CCS, click Yes (or click on the CCS 3.2 icon to launch CCS with Parallel Debug Manager).**

**3. Connect the boards and open CCS windows for each CPU.**

Select:

Debug → Connect

This should connect both boards (no red circle with slash on cpu\_0 and cpu\_1).

Select:

Open → cpu\_0

This will open a CCS window pointing to cpu\_0 (i.e. the DSK). Now open cpu\_1 as well (the CPU on the Mezzanine card). Now you have two CCS windows open – one for each CPU:

- cpu\_0: DSK
- cpu\_1: Mezzanine card

**4. Open master project for cpu\_0 (DSK).**

Make sure you have the active CCS window for cpu\_0 on your screen. For cpu\_0, open the project `srio_master.pjt` under the directory path:

```
C:\IW64x+\labs\SRIO_MasterSlave_DIO\master
```

**5. Open slave project for cpu\_1 (Mezzanine).**

Now, switch to the CCS window for cpu\_1. Open the project `srio_slave.pjt` under the directory path:

```
C:\IW64x+\labs\SRIO_MasterSlave_DIO\slave
```

## 6. Scan through and analyze the source files

In `master.pjt`, there are 3 source files. In `slave.pjt`, there are 2 source files. Note that the `SetUp_Srio.c` is the same exact source file for both projects. These files are stored under:

```
C:\iw64x+\labs\SRIO_MasterSlave_DIO\src
```

The code flow under `main()` in `srio_master.c` is as follows:

- Master writes 1 SRIO packet (256 bytes) to the slave device
- After write completion, it sends doorbell 0 to the slave device
- After the delay, master reads 1 SRIO packet (256 bytes) from the slave device
- After read completion, it sends doorbell 1 to the slave device

The main code has 2 loops. The inner loop runs 10 times to send 10 SRIO packets then stops. The outer loop waits for a user input. You can trigger this command via any means that you can think off. In this lab, we will use the GEL command which is shown in the steps below.

---

Note: For SRIO, you want to optimize it by architecting your system to do only writes, never reads. For example, rather than trying to read a buffer from another device through SRIO, you would instead do a write to let that device know to write that buffer to you. The reason for this is that you get CPU stalls while waiting for each read to complete whereas having the other device write the data into your memory allows the CPU to keep crunching along and then you can get an interrupt at the end of the transfer.

---

The code flow in `srio_slave.c` is as follows:

- Loop forever and wait for the Master device to send a doorbell interrupt.

Look into the slave ISR to see how the doorbells are handled.

---

Note: This is the portion of the code that you will need to modify to handle doorbell messaging. If you did not have the master do the read, then you can use doorbells to signal the slave to write the data back to the master device.

---

## 7. Build, Load, & Run.

First, build the code on slave side (`cpu_1`), then run it.

Second, build code on the master side (`cpu_0`), then run it.

Observe the messages in the stdout for both master & slave CCS windows.

## 8. Re-run the test

Load GEL file for master (cpu\_0) project:

File → Load GEL ...

`C:\iw64x+\labs\SRIO_MasterSlave_DIO\src\Control.GEL`

Run GEL command:

GEL → Next Run Dialog – Set\_NextRun.

Enter a 1 then click on the Execute button.

You should see another set of 10 runs. Click Done.

## 9. Open Memory Windows

In the master project, open a Memory window address at 0x90 0000. This is the location of MASTER\_WRITE\_BUFF (pSrioData = 0x90 0000). Look in the header file, srio\_Lab.h, and see the definition of MASTER\_WRITE\_BUFF. Also, look in srio\_master.c (line 22) and you can see the pointer (\*pSrioData) set to MASTER\_WRITE\_BUFF.

In the slave project, open a Memory window address at 0x90 0000 (SLAVE\_BUFF) .

Re-size the memory windows of both CCS windows so that you can see both memory windows (master and slave) at the same time.

## 10. View the data transfer

To see the data transfer, we need to set a breakpoint in both projects. First, if the master and slave are running, halt both processors. The animate key (instead of run) will run to a breakpoint in the slave code, then halt and display the results in the memory window, then it will run again.

Set a breakpoint in srio\_master.c at line 40 (while loop). Set a breakpoint in srio\_slave.c on line 100 (while loop).

Click on the slave's (cpu\_1) animation button (just underneath the Halt button).

Run the master (cpu\_0). If needed, click on Set\_NextRun Execute to continue with the transfer.

To observe how the data transfers from the master's CPU memory to slave's CPU memory, use the GEL command in the master code to re-run the loop.



**You're Done**