

4

Designing Your Own Soft Modules

Objectives

- Learn how to create circuit schematics with OrCAD
- Learn how to export a circuit from OrCAD as an EDIF netlist.
- Learn how to import an EDIF netlist into the FastChip library as a new soft module.
- Learn how to use the imported module in CSoC designs.

Why Design Your Own Soft Modules?

By this time you have created designs using soft modules and the 8032 MCU alone and in combination. The keyboard interfaces in the previous chapter required you to connect multiple soft modules together and then interface it to the MCU. Two problems with this approach are:

- The soft modules are interconnected only through the names of the signal nets and this makes it difficult to visualize the flow of information between the modules.
- A subgroup of interconnected soft modules can't be transferred from one FastChip project to another so design re-use is difficult.

This chapter will show how these problems are solved by using OrCAD Capture. Capture lets you draw your circuitry as an easily-understood schematic built from standard logic symbols with interconnecting wires and buses. You can export your circuit as a netlist file in EDIF format and then import the netlist into the library of any FastChip project. The imported module acts just like the standard soft modules in the FastChip library. Simply instantiating the imported module will bring all the functionality of your circuit into your FastChip project. The rest of this chapter will show an example of how to design a new soft module with OrCAD Capture and use it in a FastChip project.

Design 4.1 - PS/2 Keyboard Interface Module

In Chapter 3 you built an interface from soft modules that received a serial bit stream from a keyboard and interrupted the 8032 MCU when a complete scan code was available. Now you will rebuild that interface as a single soft module using OrCAD Capture.

The schematic for the keyboard interface circuit is repeated in Figure 19. Falling edges of the **ps2_clock** signal strobe keyboard scan code bits from **ps2_data** into the **ps2_data_sreg** shift register. Each rising edge of **ps2_clock** sets the **rcv_active** flip-flop which indicates the receiver circuit is gathering scan code bits. The **bit_timer** counter is also cleared whenever **ps2_clock** is low. Once **ps2_clock** stays high at the end of the scan code transmission, the 25 MHz BusClock will have sufficient time to increment the counter until bit **bit_timer[11]** goes high. Once **rcv_active** and **bit_time[11]** are both set, this drives the **rcv_int_comb** signal high which 1) clears the **rcv_active** flip-flop indicating the receiver is no longer active, and 2) sets the **rcv_int** flip-flop that strobes the scan code from the shift register into the **rcvData** register and sends an interrupt (**INTR0**) to the 8032 MCU. When the MCU responds to this interrupt, it reads the scan code from the logical address of the **rcvData** register. The presence of the **rcvData** register address on the CSI address bus triggers a selector. The **RdSel** signal goes high and this gates the scan code onto the CSI data bus while simultaneously clearing the interrupt from the **rcv_int** flip-flop.

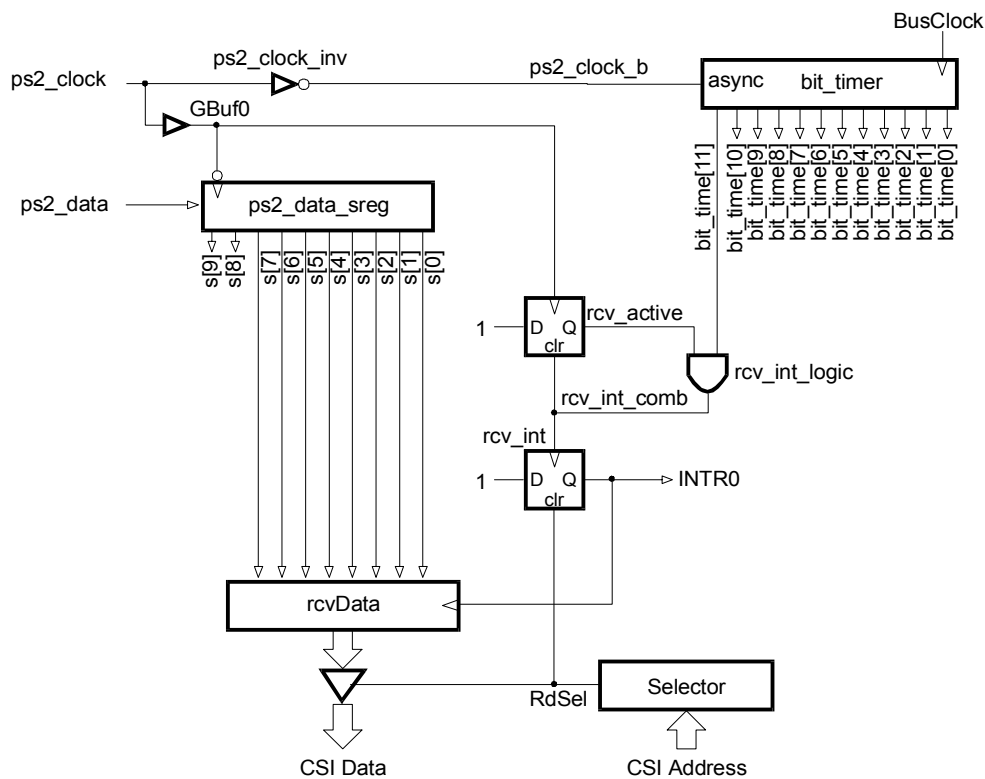
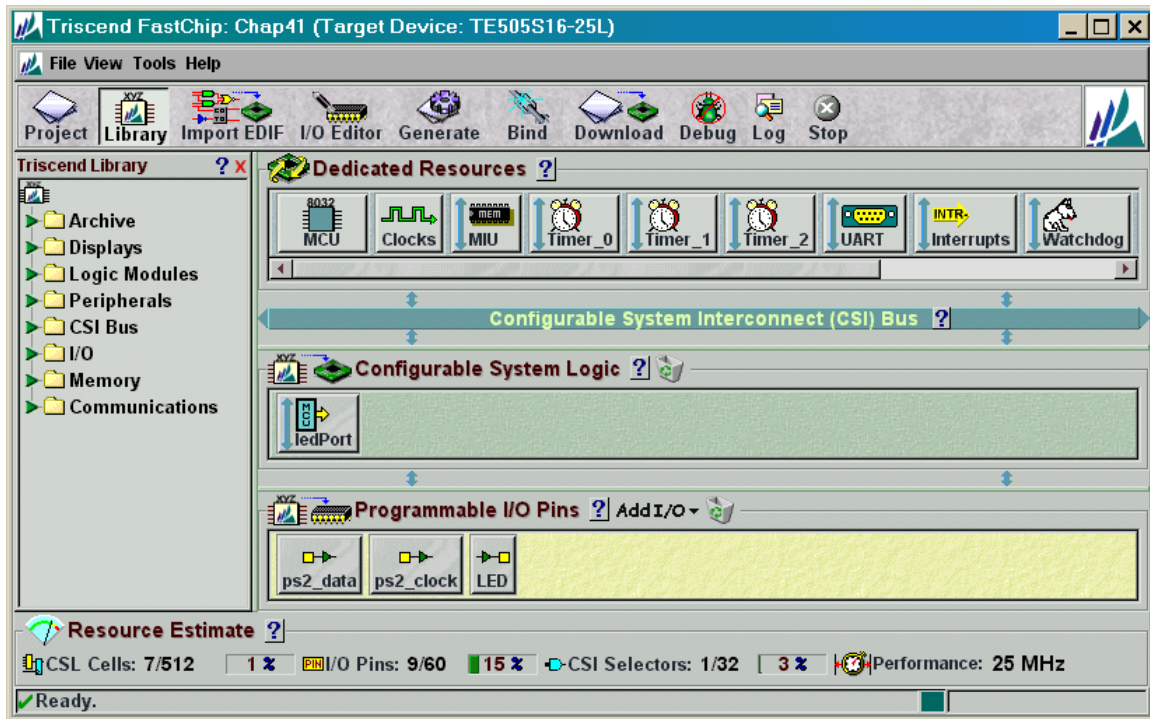


Figure 19: Schematic of a PS/2 keyboard interface circuit that uses interrupts.

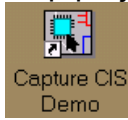
Starting the Design

You can start this design by loading the **Chap31** design into FastChip. Then delete all the soft modules in the keyboard interface and save the project under the name **Chap41**.



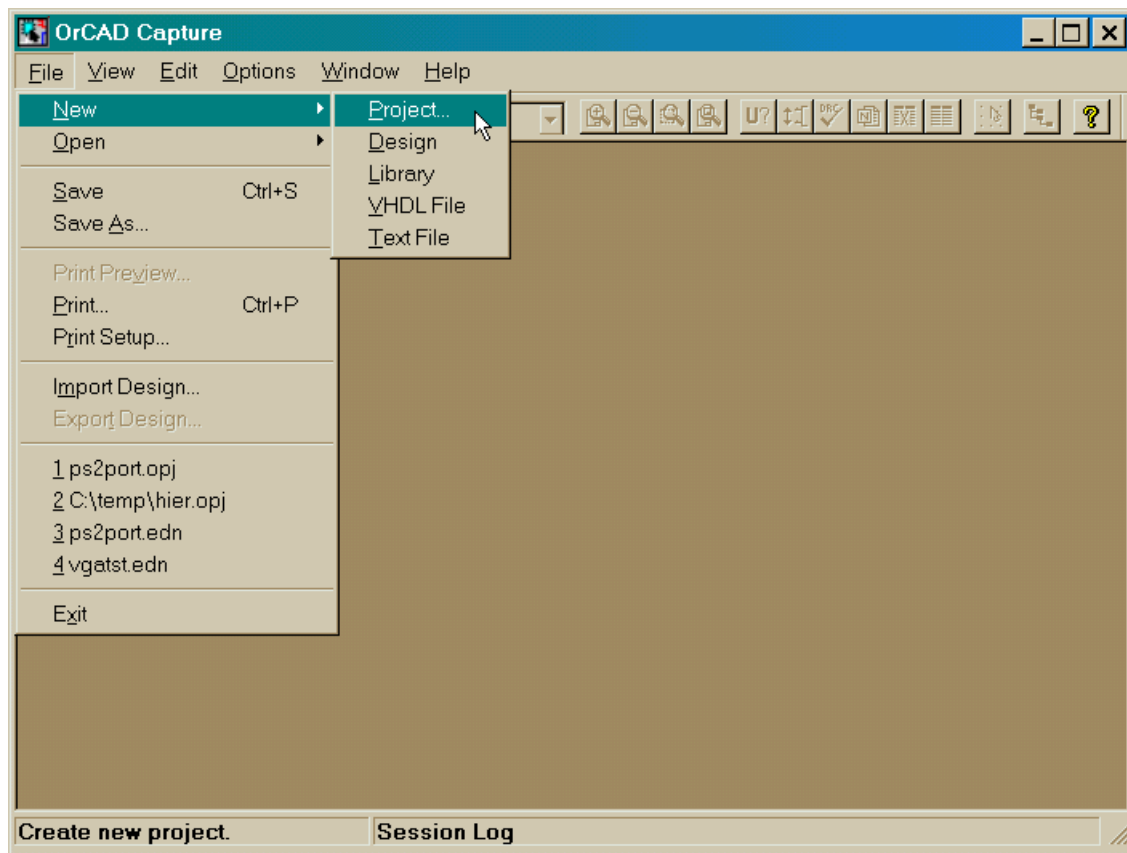
Starting a Project with OrCAD Capture

Now that you have the FastChip project started, create a folder named orcad in the Chap41 FastChip project folder. Your OrCAD schematic files will be stored there. Then

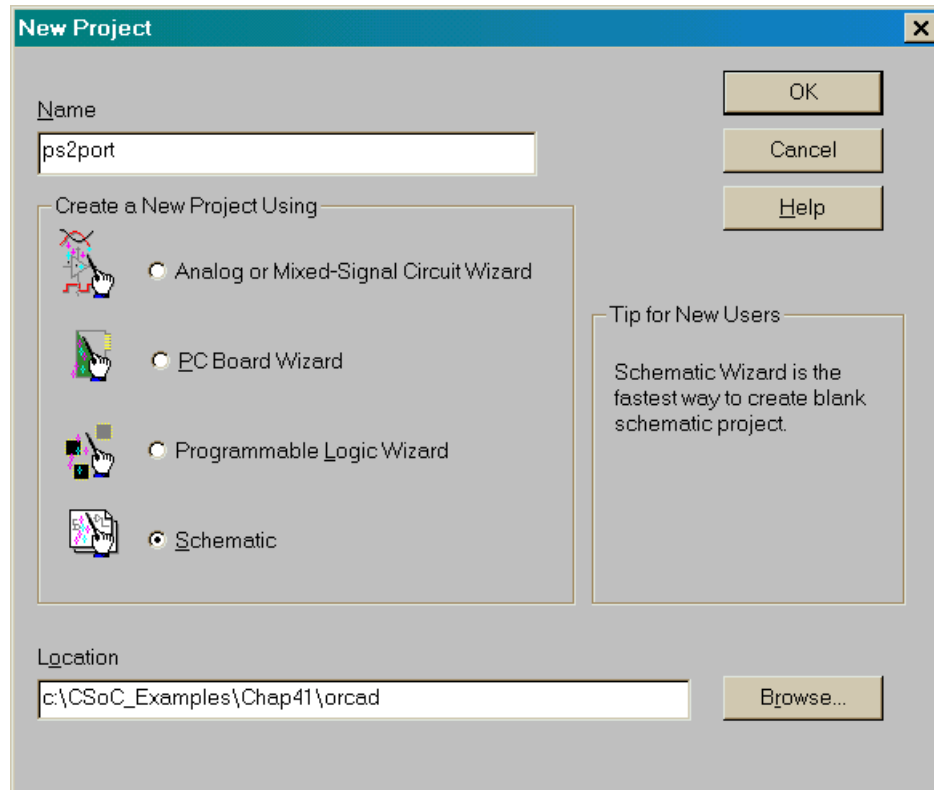


click on the icon to start the OrCAD Capture software.

In the **OrCAD Capture** window, select File⇒New⇒Project... to start a new Capture project.



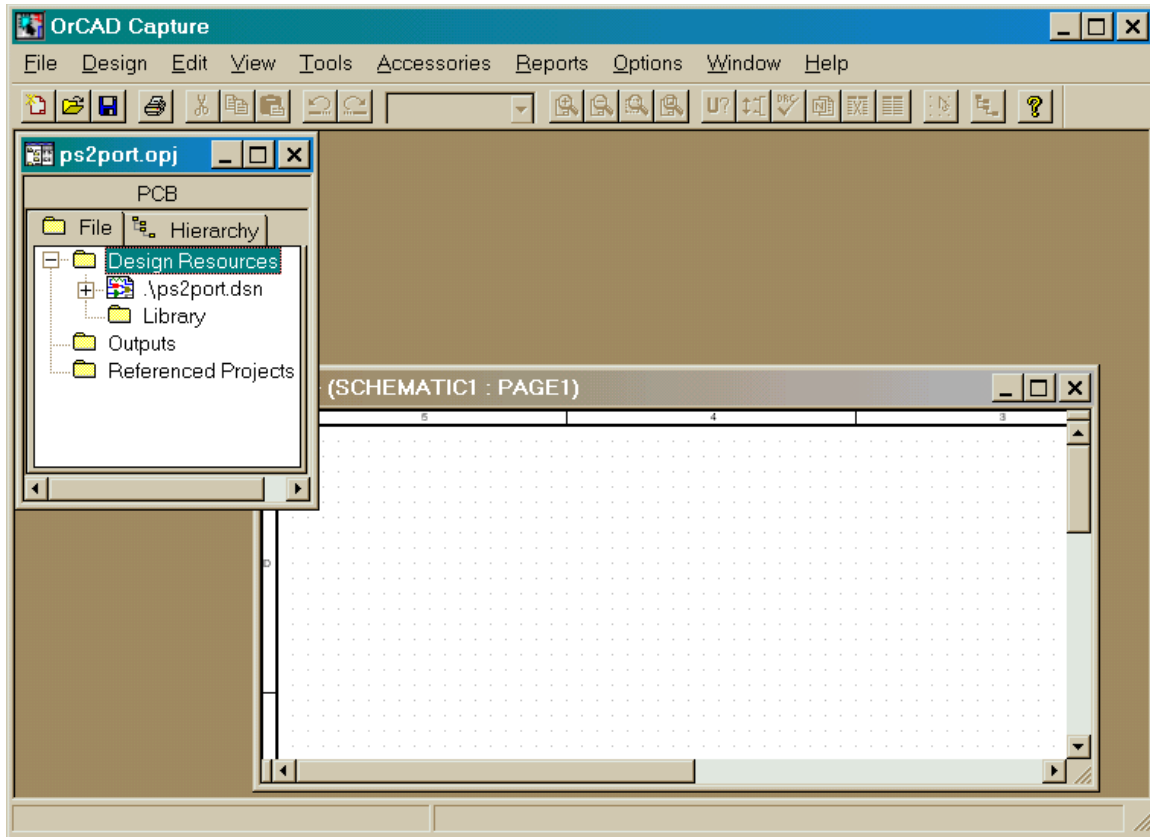
In the **New Project** window that appears, click on the Schematic radio button since your design will be done as a schematic. In the Name box, type `ps2port` as the name of your keyboard interface module. Then use the Browse button to select the `orcad` folder under the `Chap41` project folder as the location for the Capture project files. Then click on OK to accept these settings.



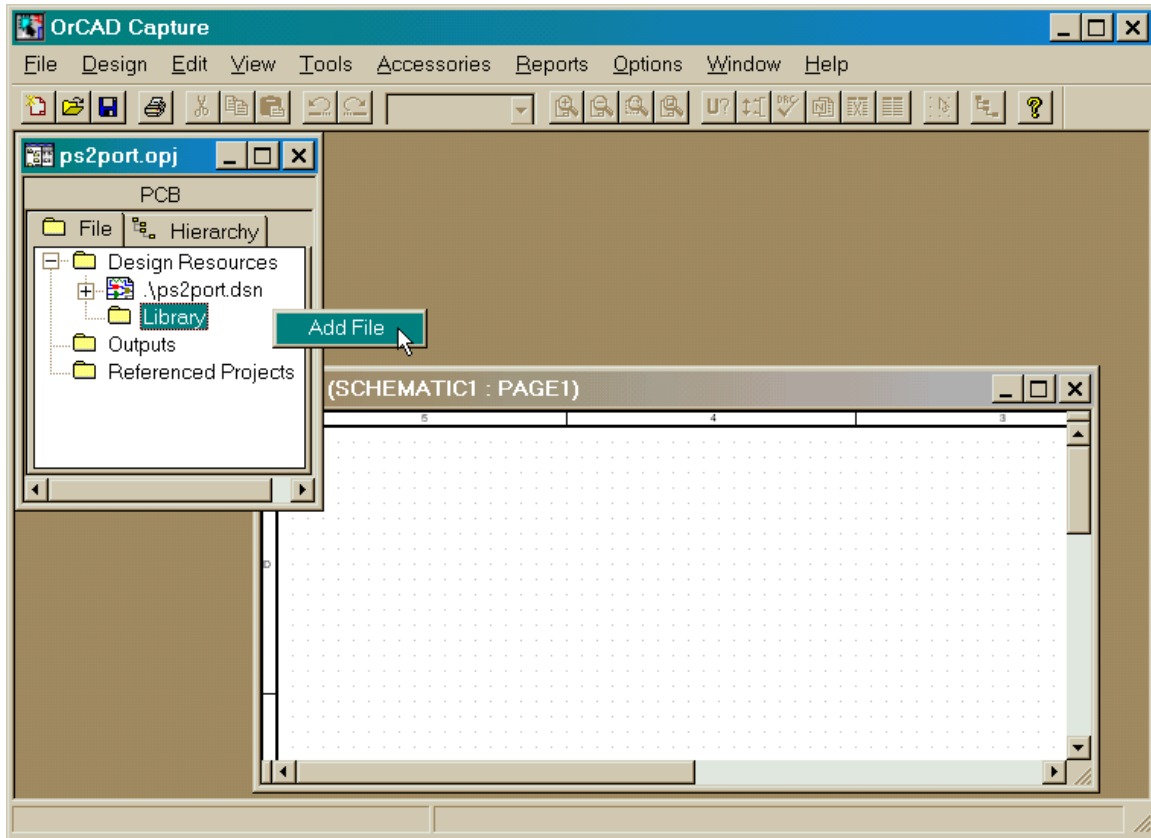
Now the OrCAD Capture window will contain two subwindows:

ps2port.opj: This window shows the organization of the entities in this project. The File and Hierarchy tabs let you select whether you want to view the organization of the project files or the circuitry components.

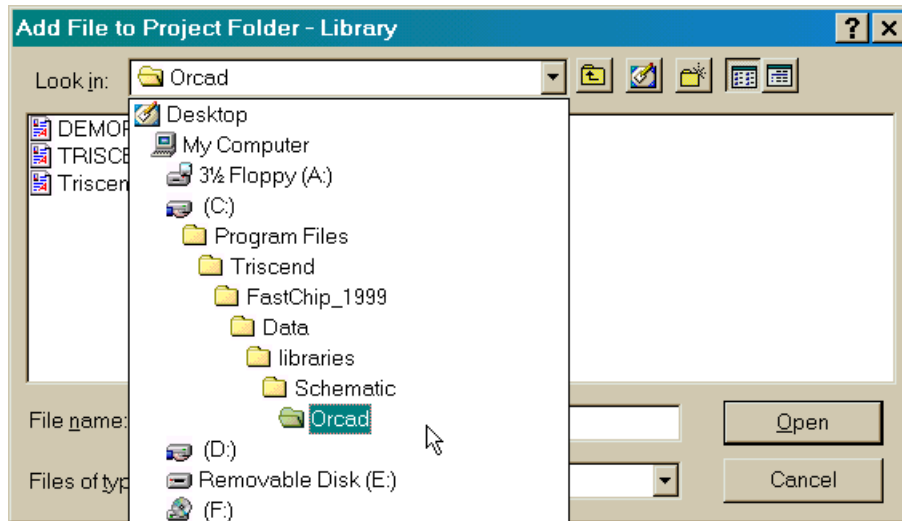
SCHEMATIC1:PAGE1: You will draw your schematic in this window.



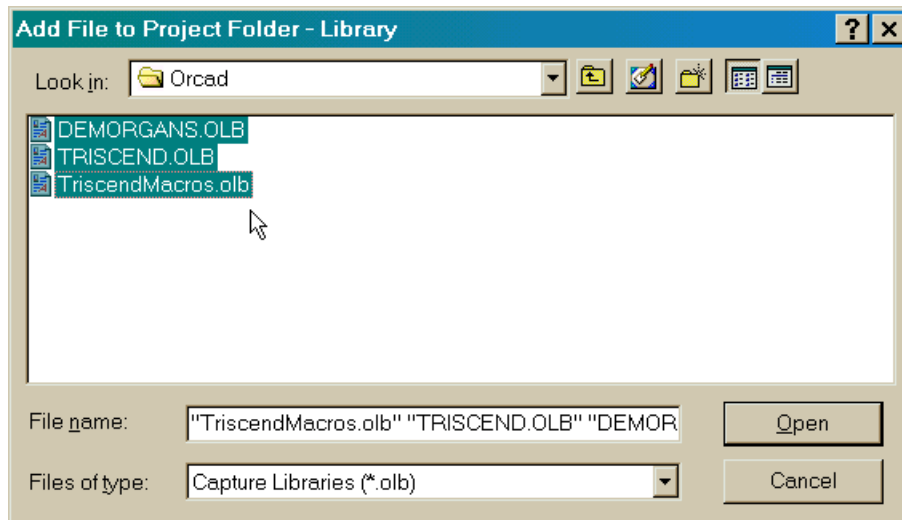
Now you need a library of circuit elements that you can interconnect to build your circuit. Triscend supplies several libraries of components that are specially designed for the CSL matrix of the CSoC. You should only use components found in the Triscend libraries. To add these libraries to your OrCAD project, right-click on the Library folder in the **ps2port.opj** window as shown below. Then select Add File in the pop-up menu that appears.



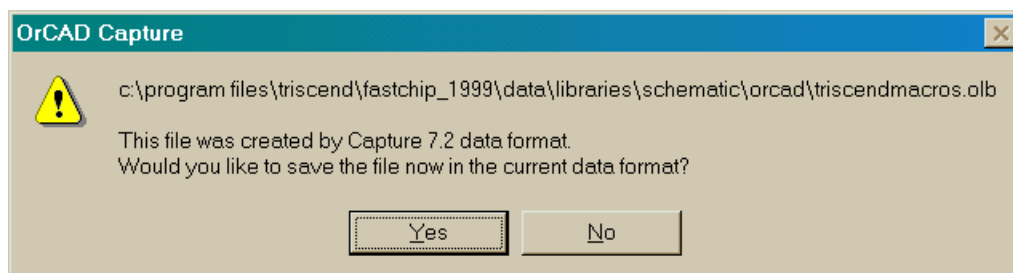
In the **Add Files to Project Folder - Library** window, steer your way through the folder hierarchy shown below to find the Triscend schematic libraries.



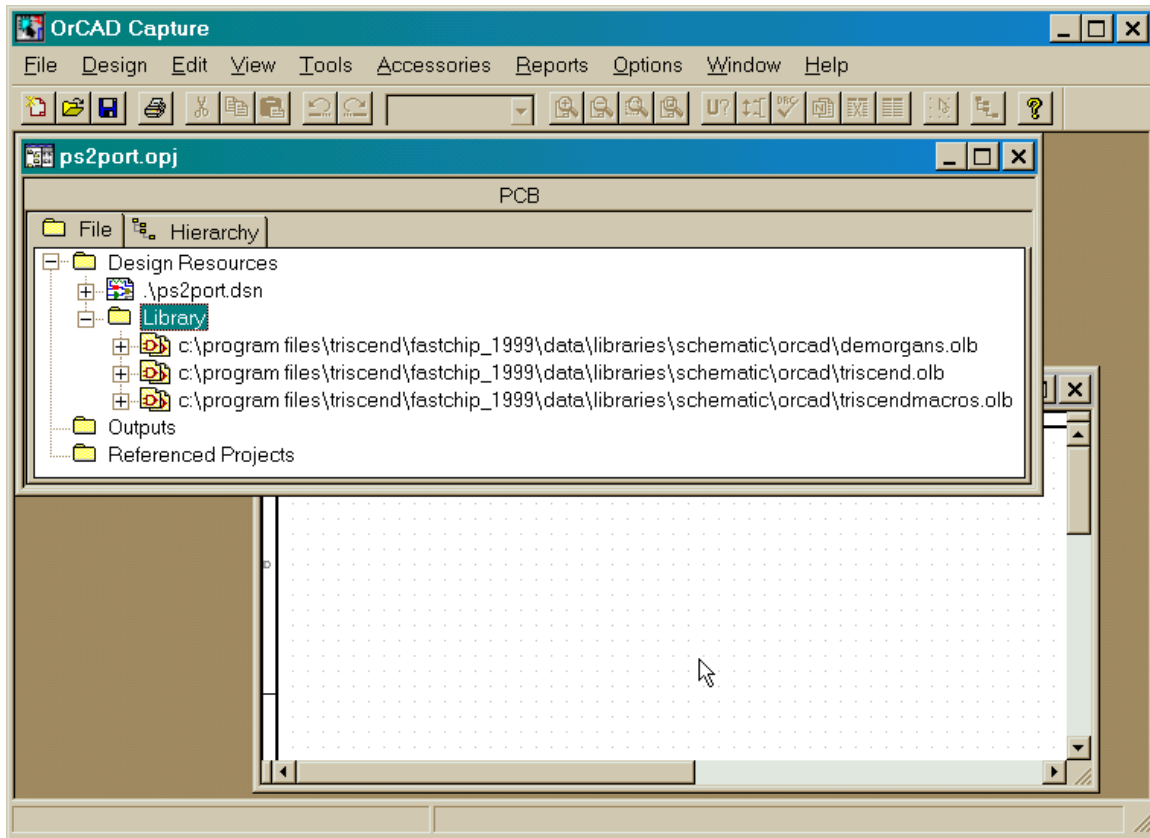
Select all three of the Triscend schematic libraries in the folder and click on Open.



Once you click on Open, you may receive the message shown below. The Triscend libraries are stored in a format for version 7.2 of OrCAD Capture, but you have version 9.1 of Capture. You can elect to convert the Triscend libraries to version 9.1 or not. (I left all my Triscend libraries in version 7.2 format.)

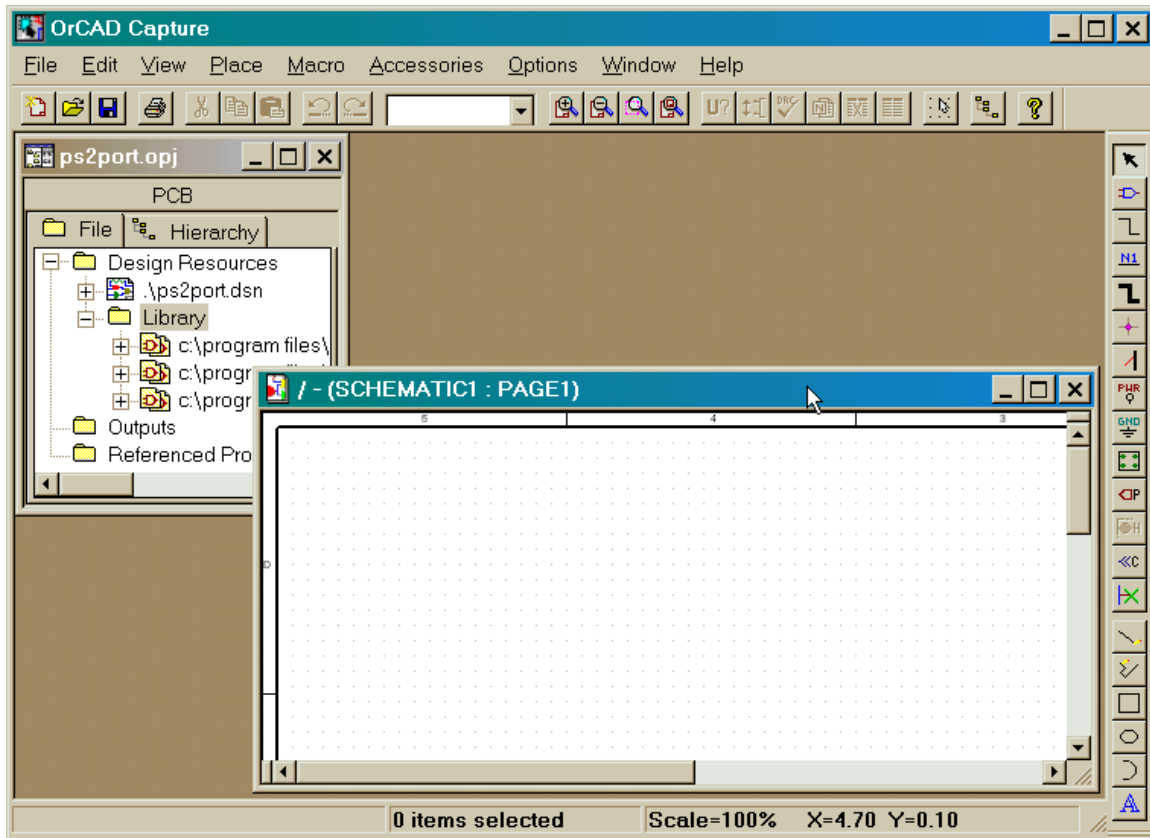



Now the Triscend libraries should be listed under the Library folder in the **ps2port.opj** window.

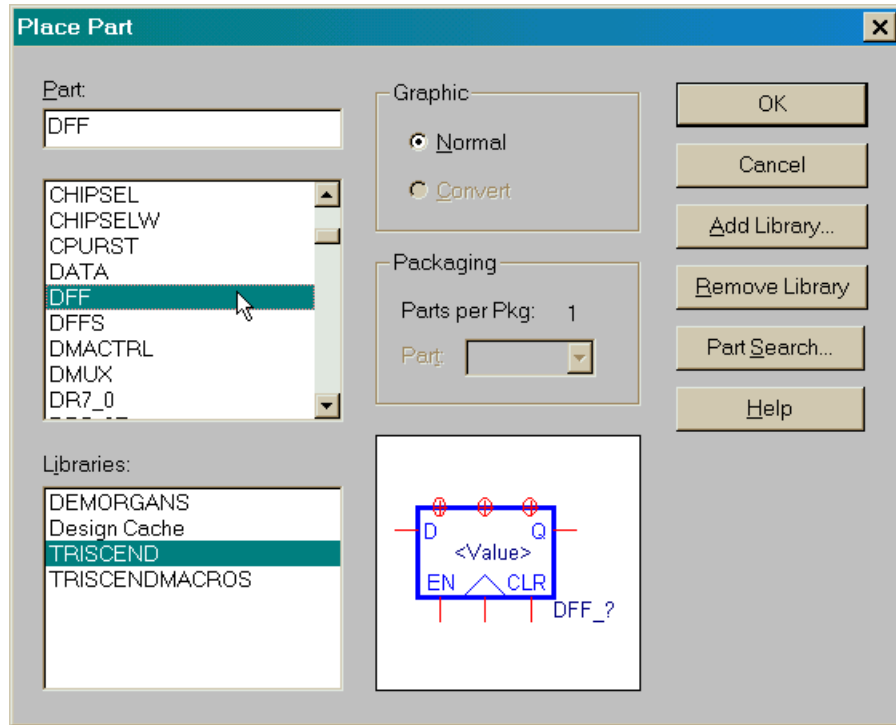


Drawing a Schematic with OrCAD capture

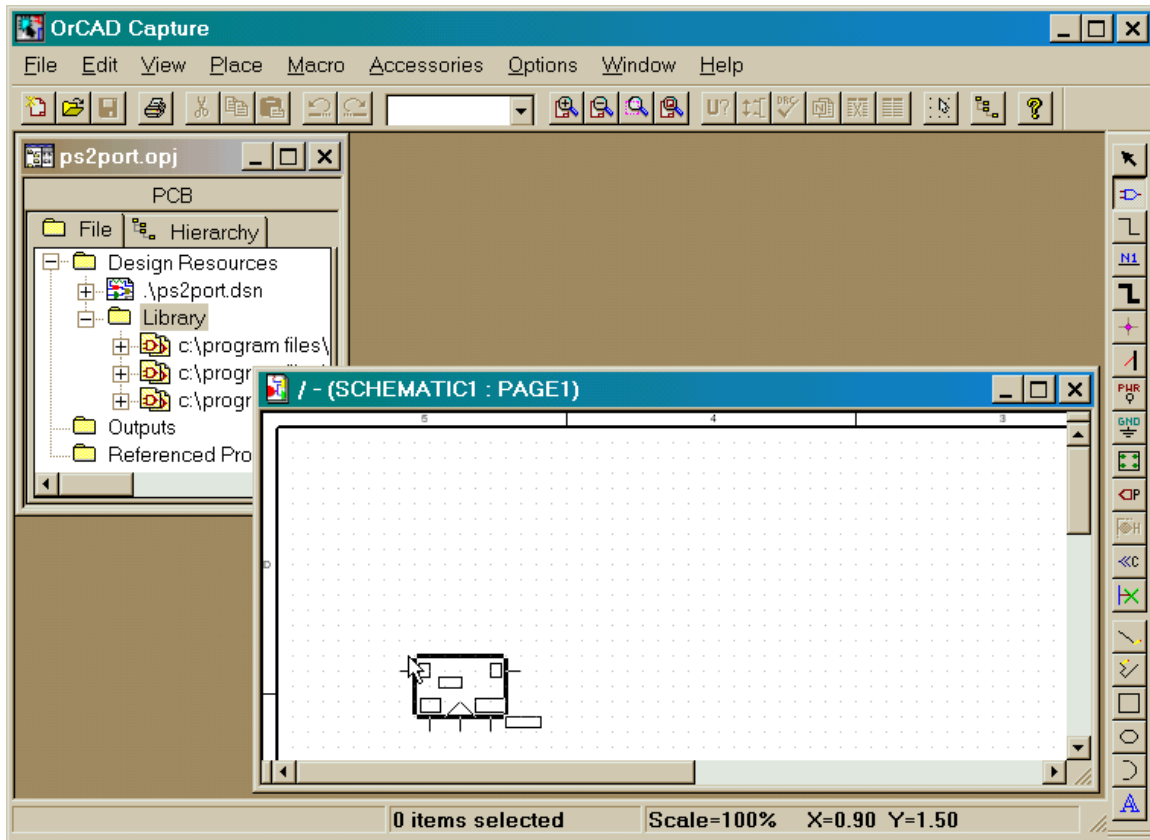
With the Triscend libraries in place, you can start drawing your keyboard interface schematic. Click in the **SCHEMATIC1** window and the tools for entering a schematic appear along the right-edge of the **OrCAD Capture** window.



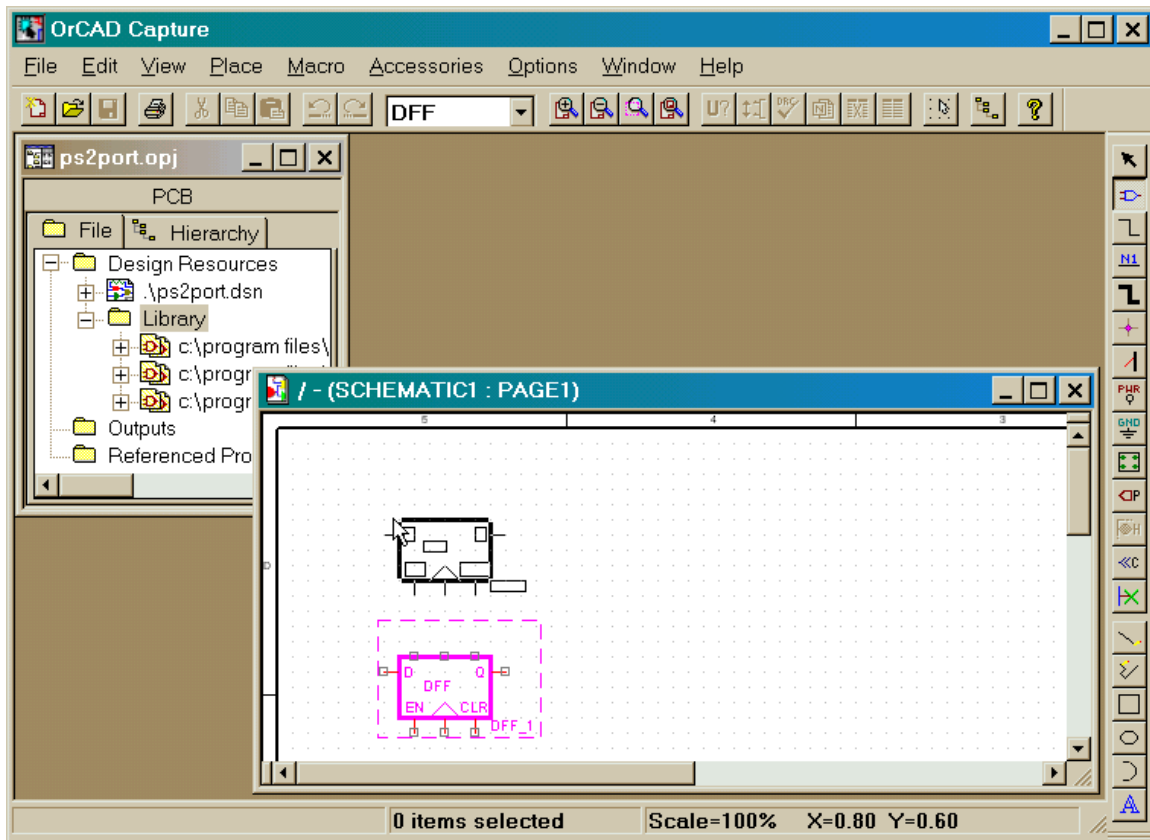
To start, you will use the part placement tool to place parts from the Triscend libraries into your schematic. Clicking on  brings up the **Place Part** window. You can highlight one or more of the entries in the Libraries area and the parts contained in the highlighted libraries are listed in the Part area. When you highlight one of the entries in the part list (a D flip-flop in this case), a small symbol for the part will appear. Once you find the part you want, click on OK.



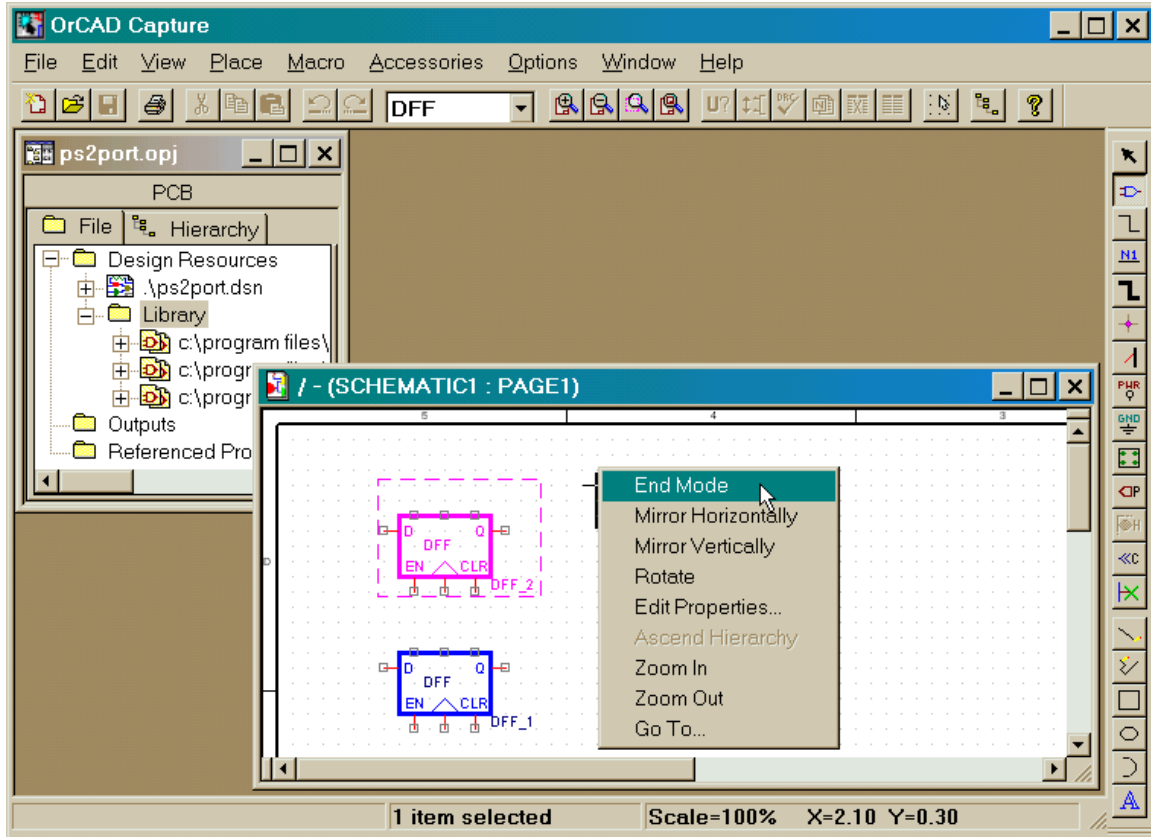
Now the part symbol is attached to the cursor as you move it within the **SCHEMATIC1** window. Clicking with the mouse drops a copy of the part symbol into the window.



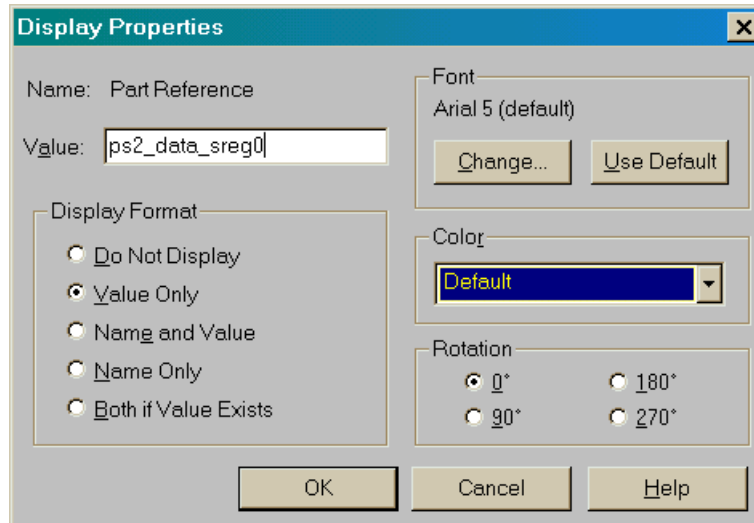
The part symbol remains attached to your cursor even after you click the mouse. This lets you add several independent copies of a part without having to go back to the **Place Part** window each time.




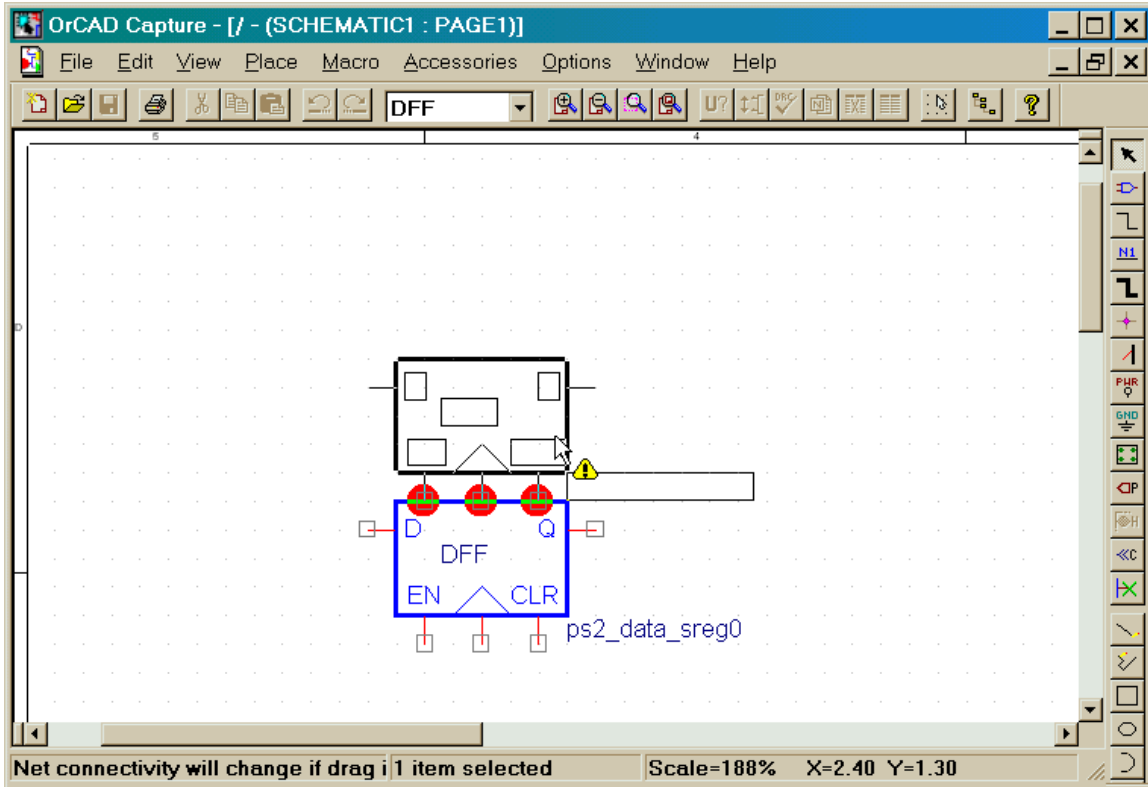
When you want to stop adding a particular part to your schematic, you can right-click with the mouse to make the following pop-up menu appear. Select End Mode and the part symbol will disappear from the cursor. You can also get the same effect by pressing the ESC key.




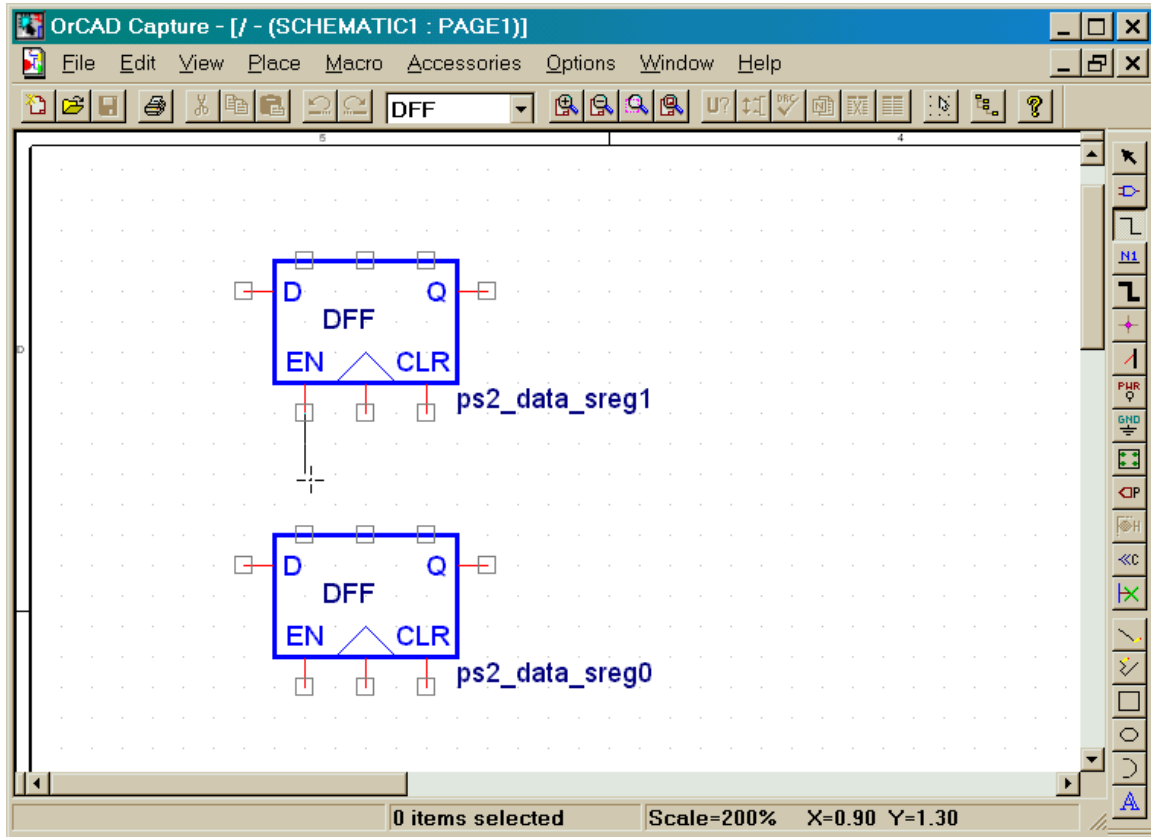
Each part that you drop into a schematic is given a default name (**DFF_1** and **DFF_2** in this example). You can give a part a more descriptive name by double-clicking on the part name. Then just type the part name you want in the Value field of the **Display Properties** window that appears.



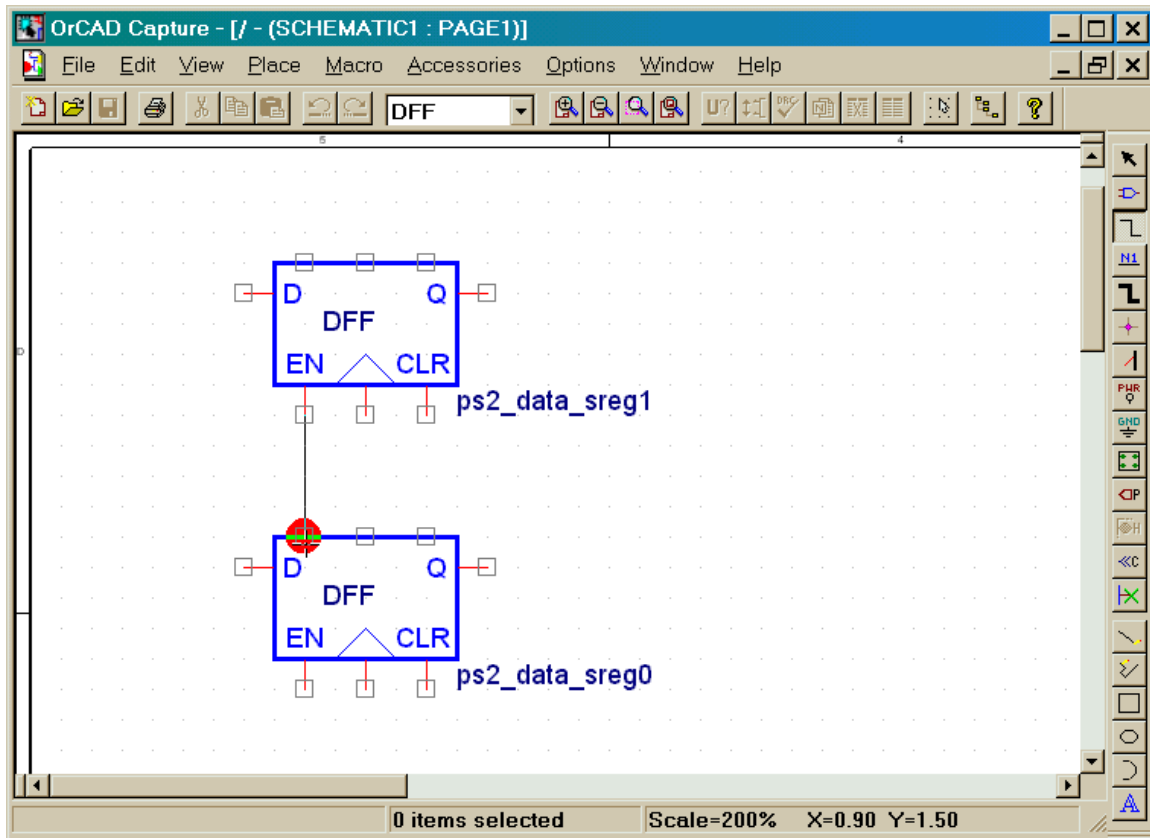
Once you add the parts to your schematic, you can rearrange them by clicking on the  tool and then click-and-drag the symbol for the part you want to move. Drag it where you want and then release it. If you drag one part close to another, the pins on each part will be highlighted. This indicates a connection will be established if you drop the part at that location. This is a quick way to attach parts to build larger functions. For example, a byte-wide register can be built by concatenating eight D flip-flops together.



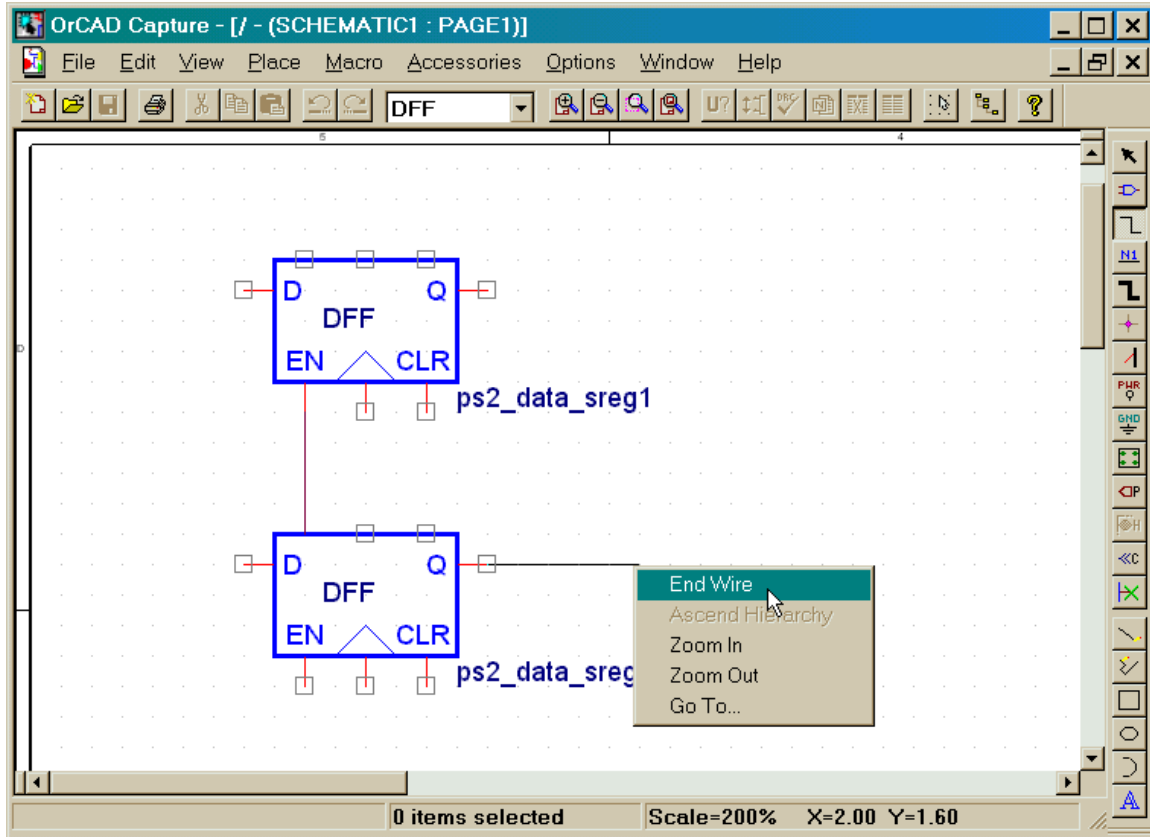
However, in most cases you will connect parts using the wiring tool. Click on the  tool and then click on a pin of one of the part symbols. As you move the mouse, a wire will extend from the pin. Click the mouse each time you want to change the direction of the wire.




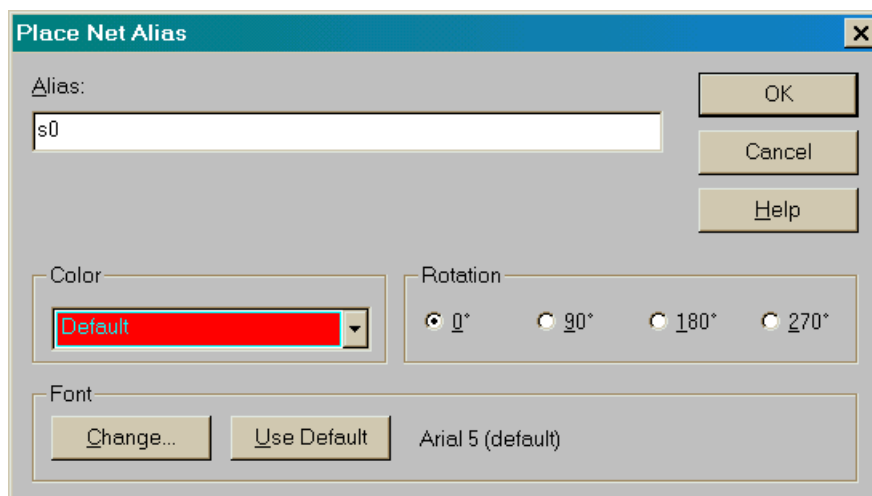
Once your wire gets close enough, a pin will be highlighted. Clicking the mouse will terminate the wire on that pin and establish a net connection between the two parts.



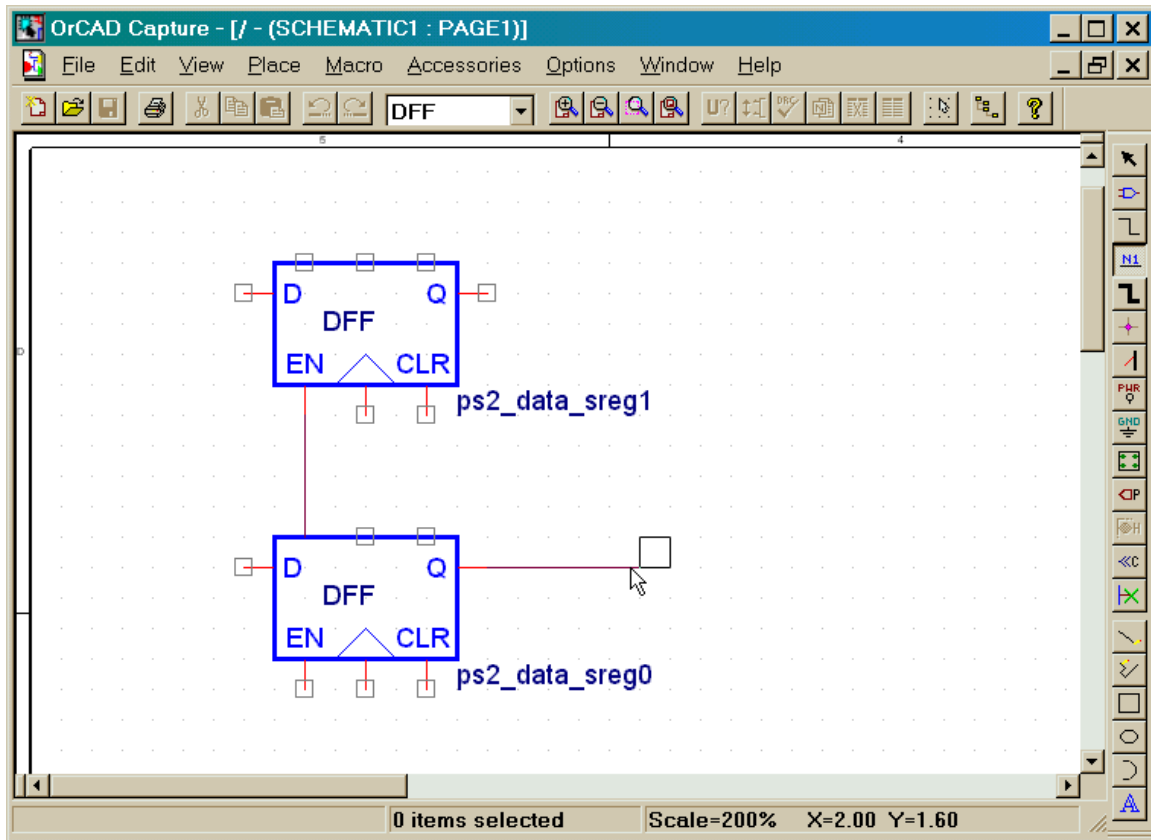
You can also attach a wire to a pin without terminating it at another pin. Just right-click while drawing the wire and select End Wire on the pop-up menu as shown below.



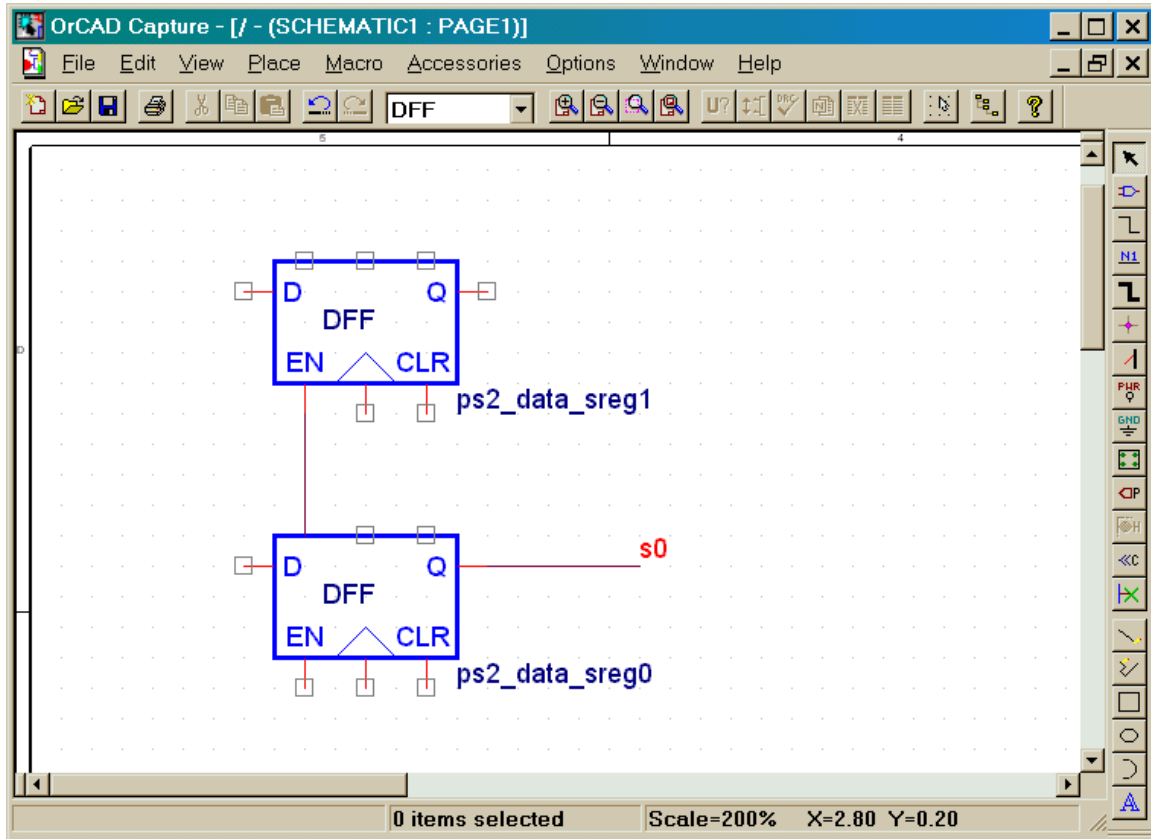
You can assign an alias to any wire by clicking on the  tool. Type the name you want to assign to the net in the Alias box of the **Place Net Alias** window that appears. Then click OK.



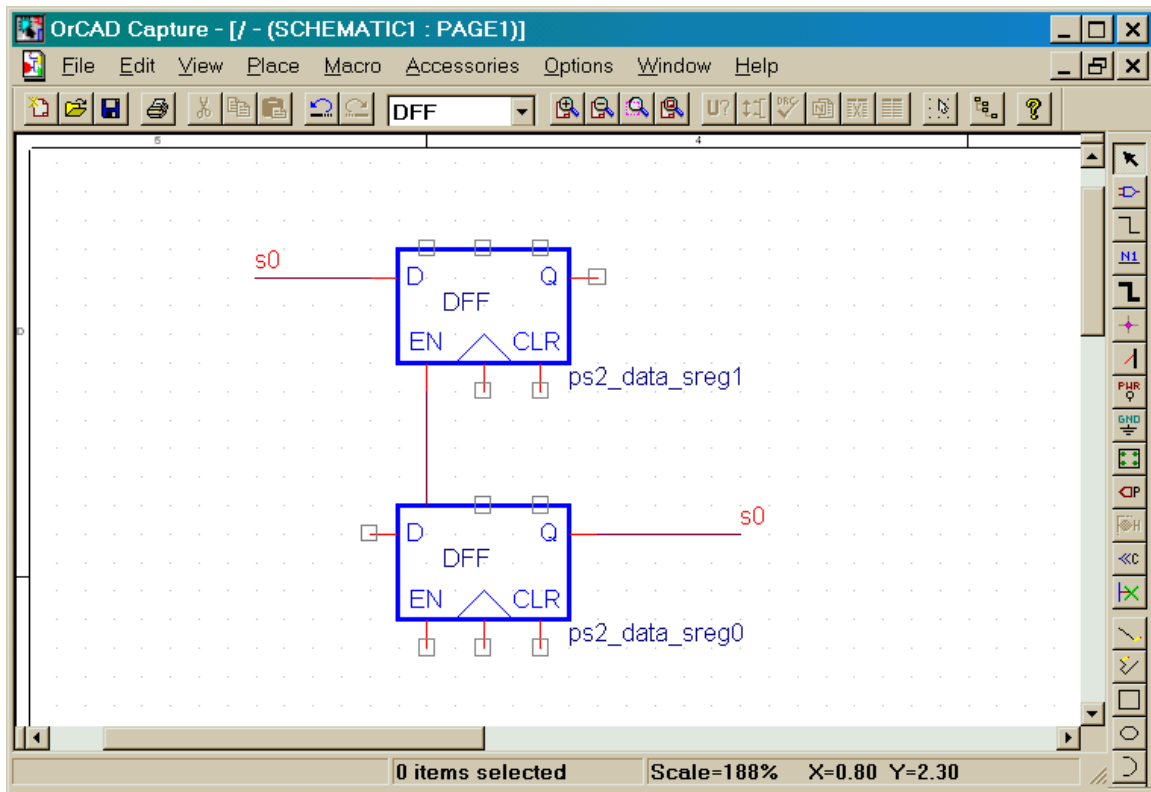
The new name for the wire will be attached to the cursor. Then just click your mouse on the wire you want to name.




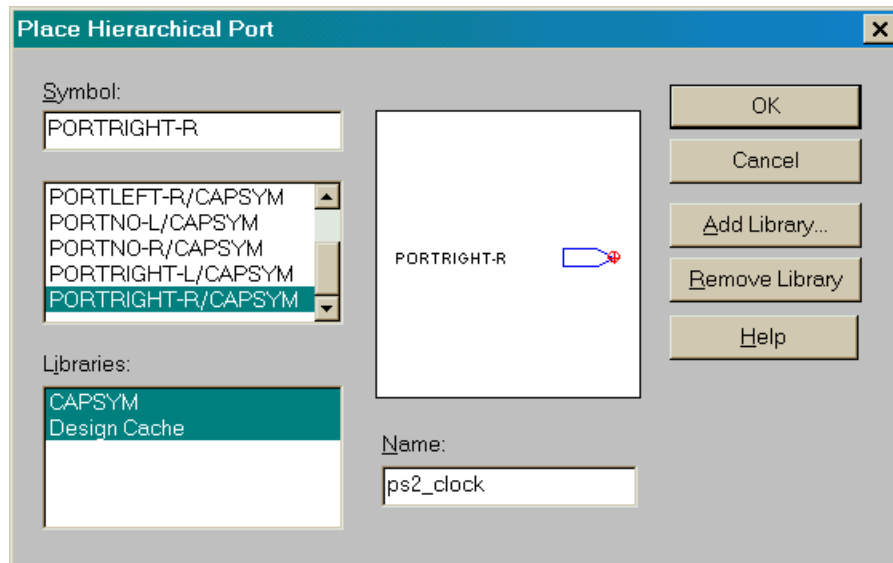
This attaches the alias to the net.



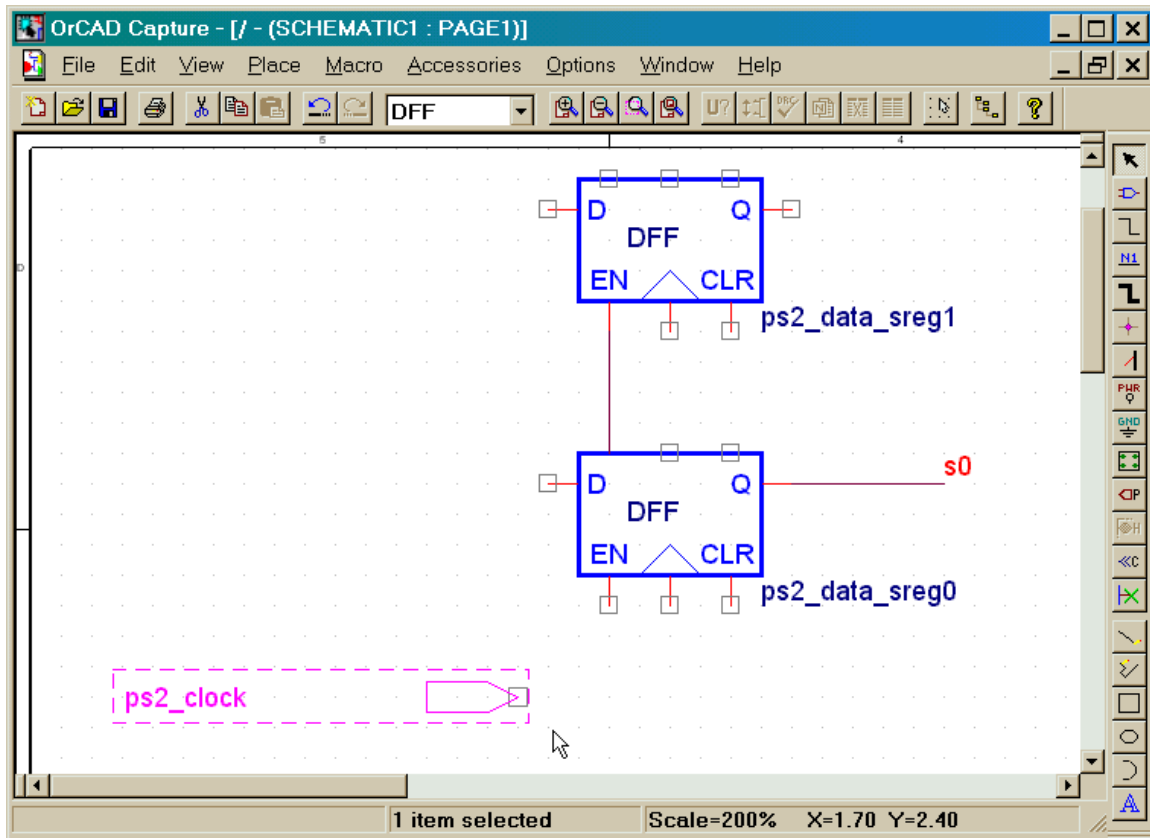
You can connect pins by attaching wires to each of them and assigning the same alias to each wire as shown below. This lets you avoid cluttering your schematic with many crossing wires.



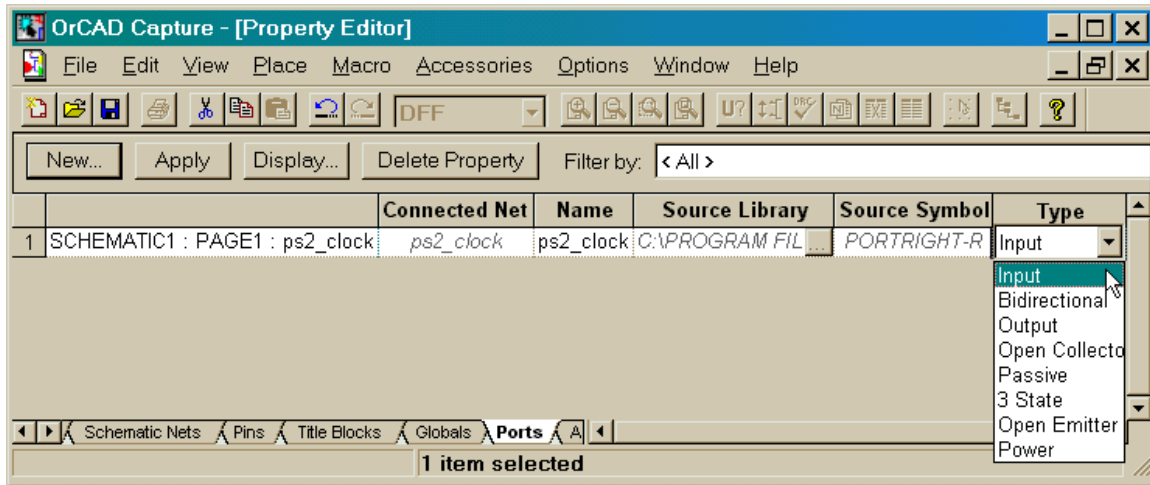
You can place parts and connect them with wires, but you also need a way to pass signals in and out of your circuit if it is going to serve as a module in a larger design. The module I/O is performed by ports. To add a port, click on the  tool. The **Place Hierarchical Port** window will appear with a list of the various ports you can drop into your schematic. Highlight one of the ports (I'm partial to **PORTRIGHT** but it really doesn't make much difference) and type a name for the port in the Name field. Then click on OK.



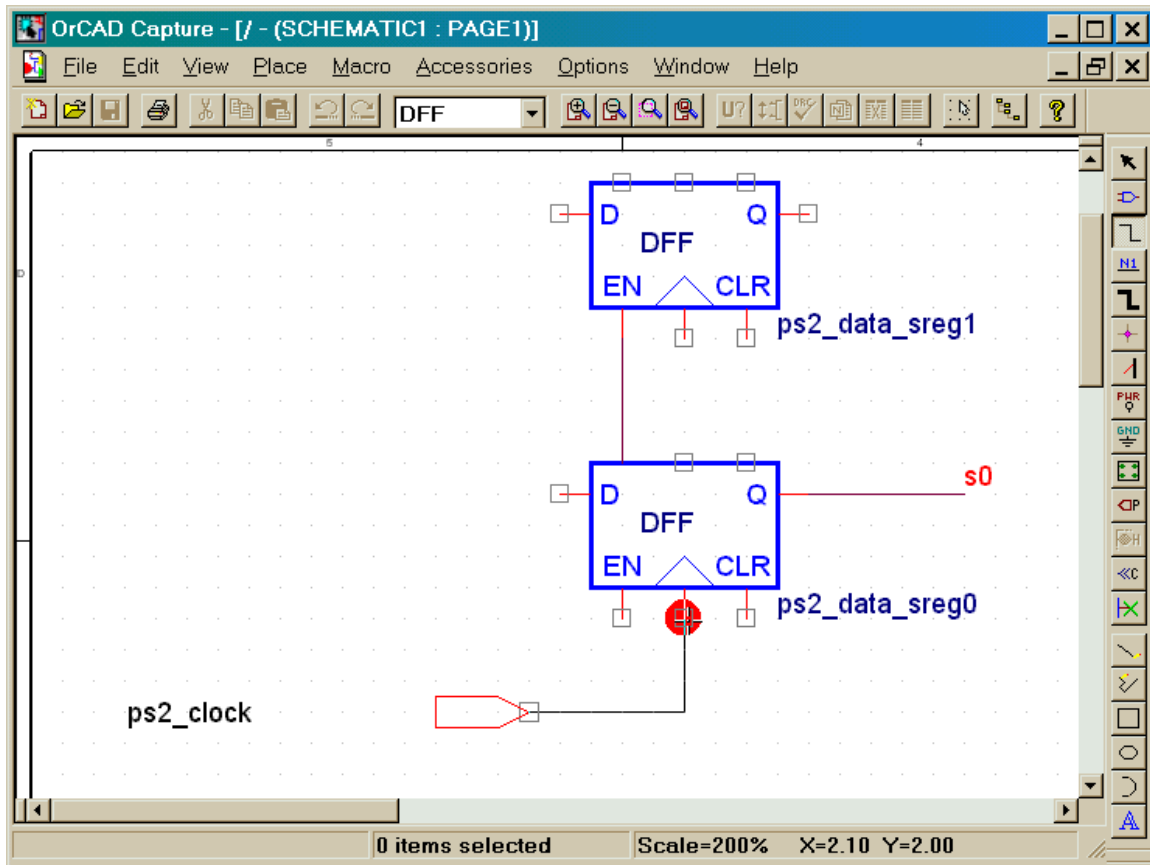
Now the port symbol will be attached to your cursor. Click the mouse to drop the port into the schematic.



Next you need to set the direction of the I/O port. Even though the symbol may appear to be that of an input, that doesn't really mean it is. You should always explicitly set the direction for the port. Double-click the port symbol to bring up the **Property Editor** window for this port instance. Select the direction of the port from the drop-down menu under the Type heading. In this case, the port brings the clock signal from the PS/2 port into the circuit, so the Input type is selected. I generally use only ports of type Input and Output when designing a module for a Triscend CSoC. If I need a bi-directional port I will add two ports - one for input and one for output.

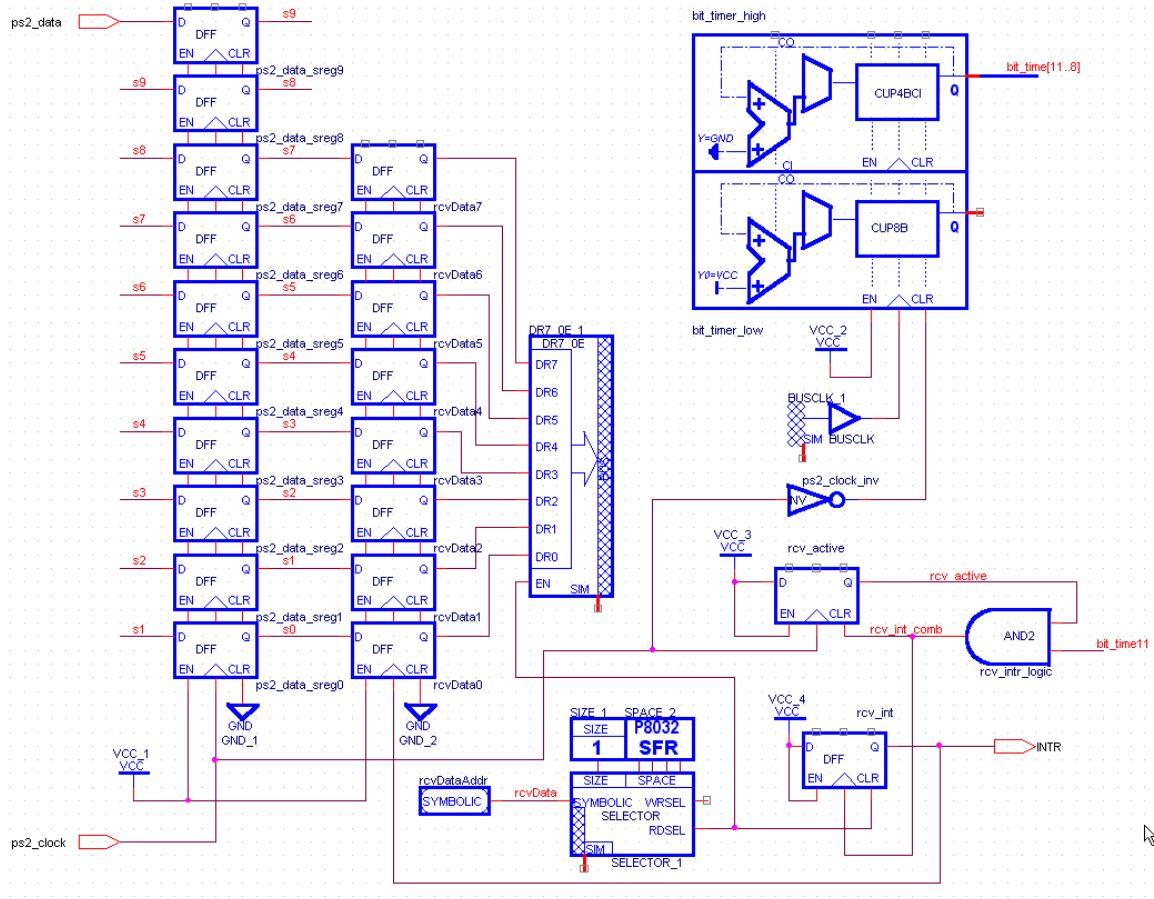


After setting the port direction, click on the Apply button and close the window. Now you can use wires to connect the port to the pins of other parts in your schematic.

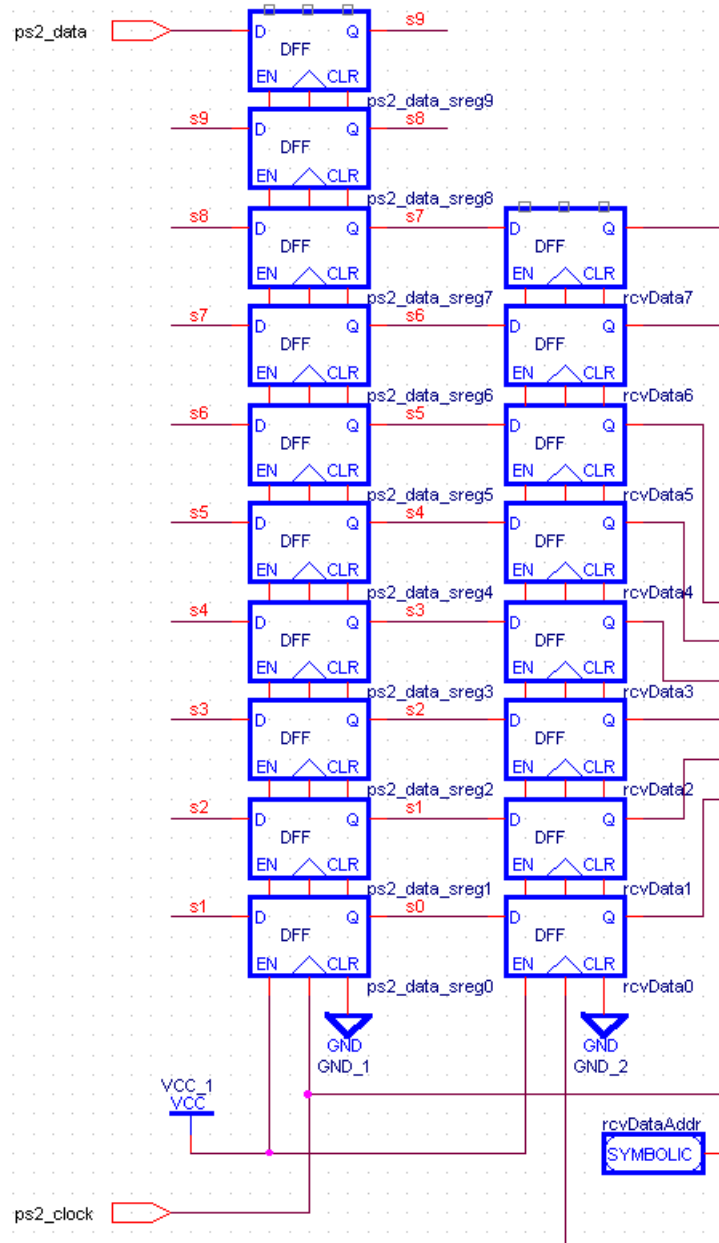


This completes my short introduction to the tools for entering a schematic. OrCAD Capture has many features and shortcuts that I haven't covered, so you should read the documentation if you want to learn more.

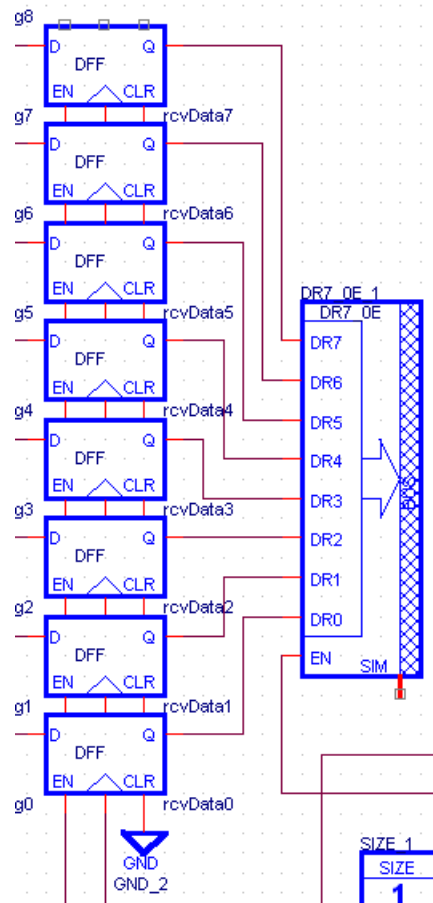
The one-page Capture schematic for the keyboard interface circuit is shown below. It follows the structure and naming of the circuit in Figure 19. Each section of the circuit will be shown in more detail.



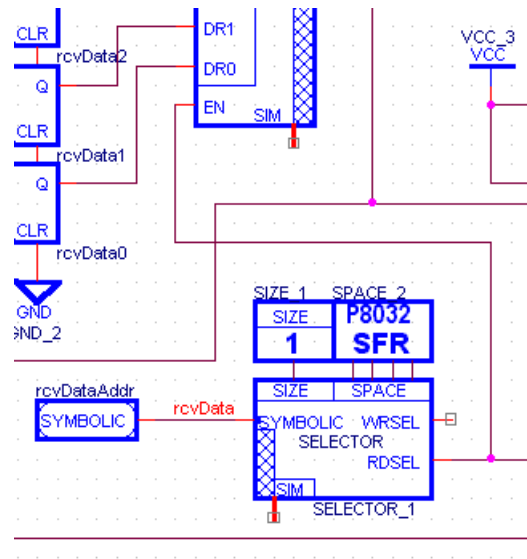
A 10-bit shift register gathers the scan code bits from the **ps2_data** input port as pulses are applied to the **ps2_clock** port. The eight scan code bits are clocked into the **rcvData** register when the **rcv_int** flip-flop is set.



The output of the **rcvData** register is gated onto the CSI data bus through the **DR7_OE_1** component when its **EN** input is driven high.



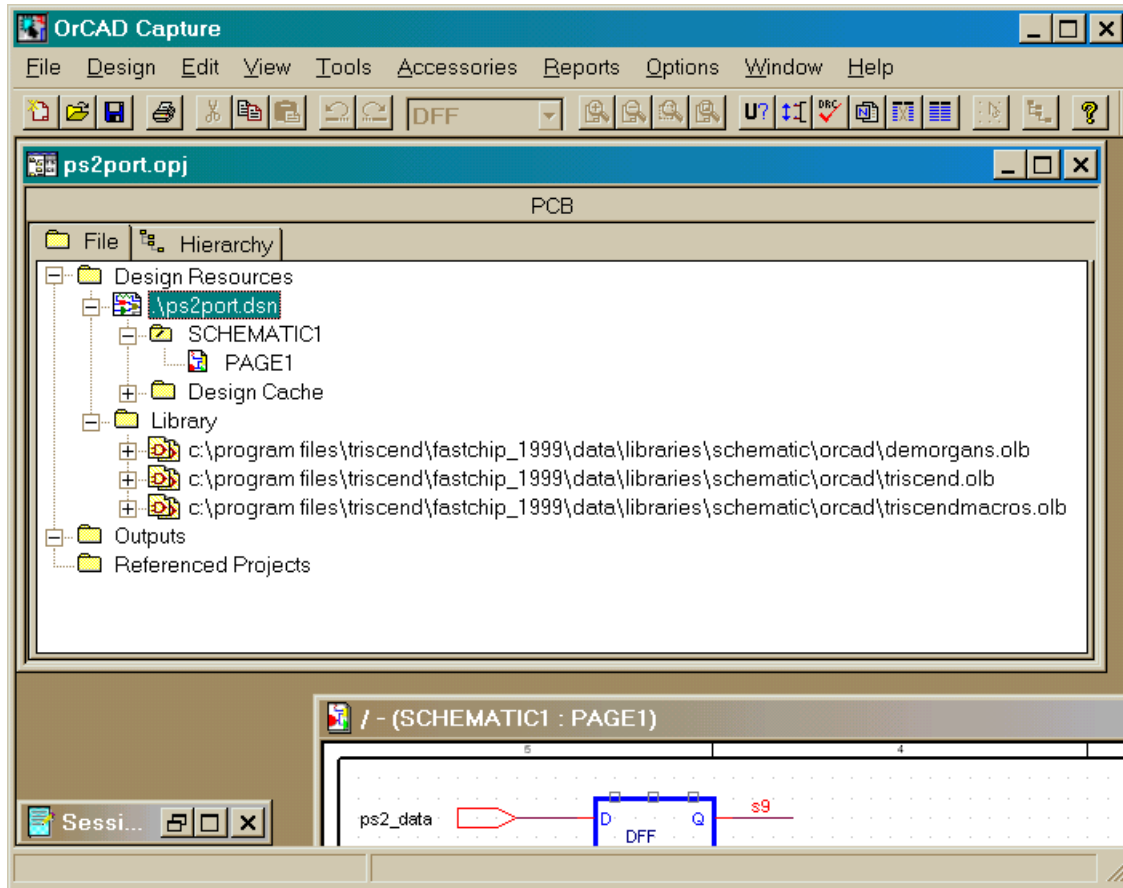
The **EN** input of the **DR7_OE_1** component is driven by the **RDSEL** output of the **SELECTOR_1** component. In this case, **SELECTOR_1** responds to a single address in the 8032 SFR address space when this address appears on the CSI address bus. This is expressed in the schematic by attaching the **SIZE1** and **P8032 SFR** components to the **SELECTOR** component. The actual SFR address is not specified. Instead, a **SYMBOLIC** component is attached to the **SELECTOR** through a wire with the alias **rcvData**. The FastChip software will replace the symbolic address with a logical address in the SFR space of the 8032. The combination of the **DR7_OE**, **SELECTOR**, **SIZE**, **SFR**, and **SYMBOLIC** parts replaces the function of the single **Status Register** soft module used in the keyboard interface of the previous chapter.



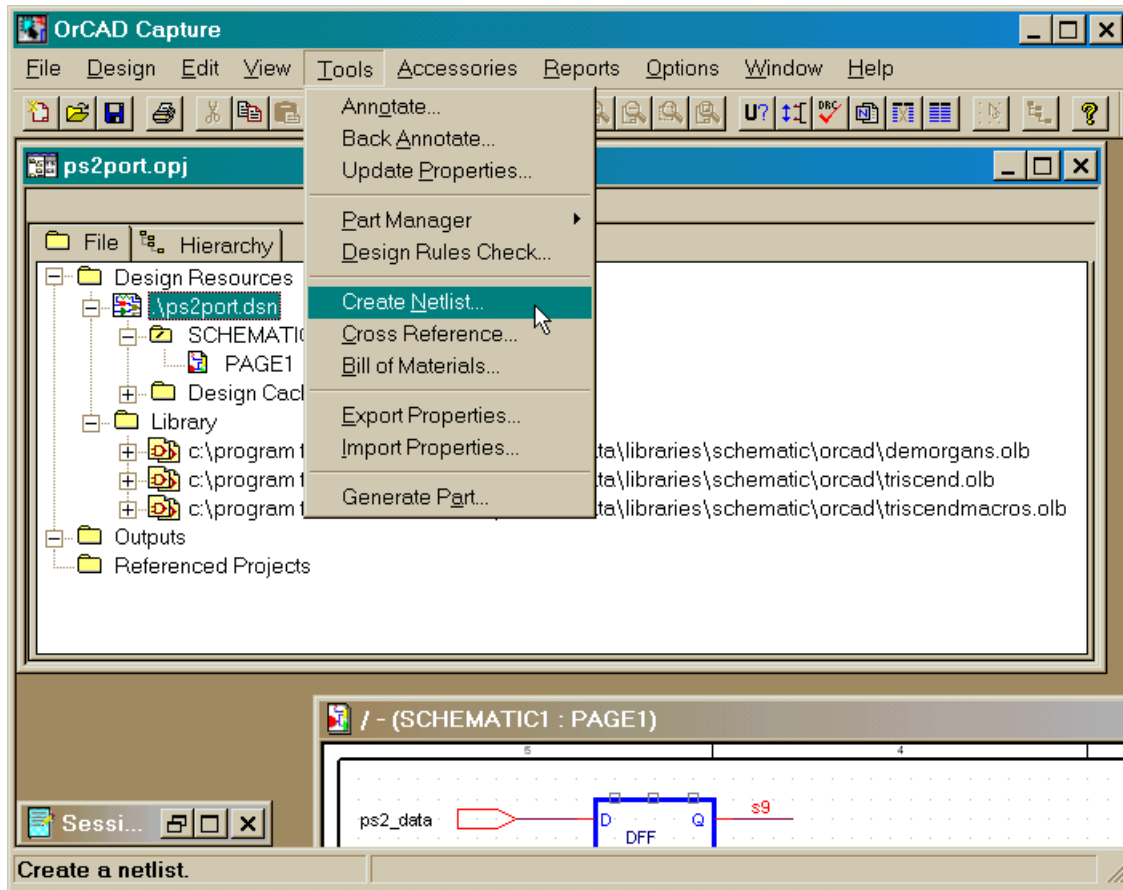
The **bit_timer** is built from an eight-bit up-counter with no carry input (**CUP8B**) coupled to a four-bit up-counter with a carry input (**CUP4BCI**). The carry output of **bit_timer_low** drives the carry input of **bit_timer_high**. The counters are clocked by the main clock of the CSoc by using the **BUSCLK** component. The counters are cleared whenever the **CLR** input goes high (i.e., whenever the **ps2_clock** signal is low).

Exporting the Netlist for a Schematic

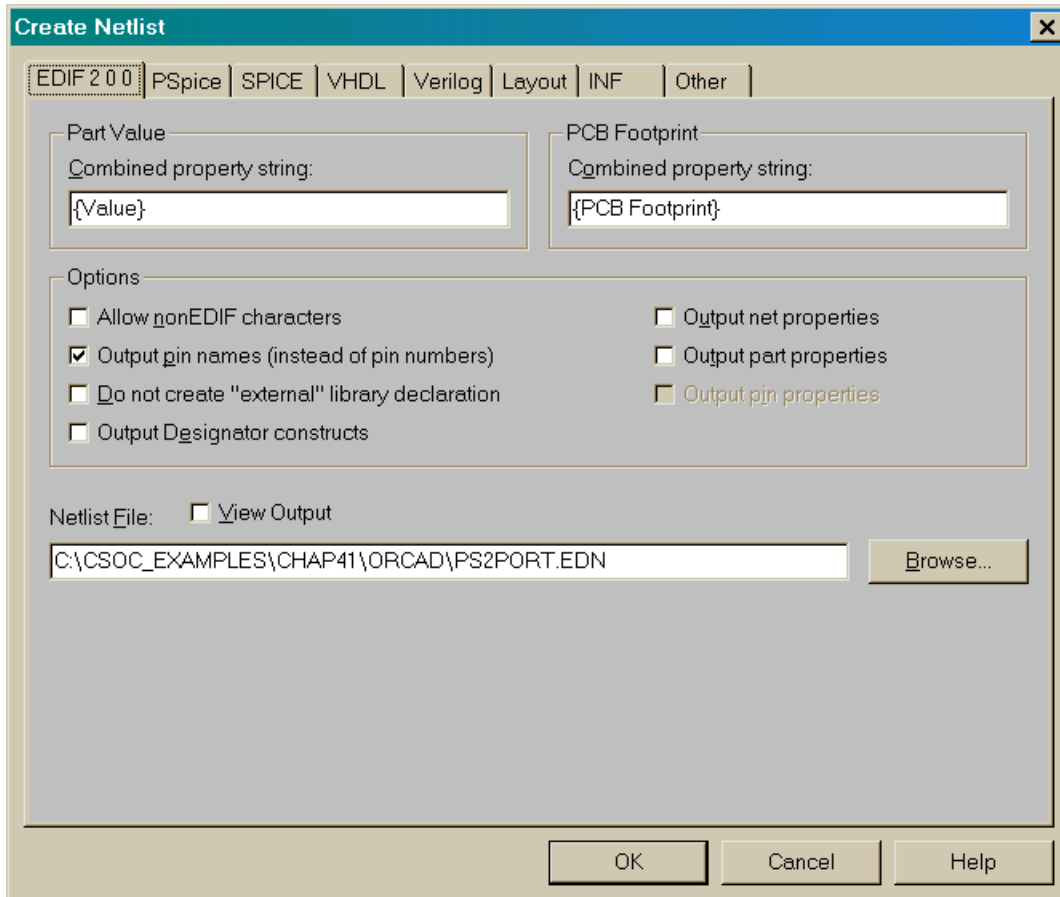
After placing all the parts and adding the wiring in the **SCHEMATIC1** window, you can return to the **ps2port.opj** window and highlight the **.ps2port.dsn** entry. This causes the schematic drawing tools to disappear from the right-edge of the Capture window and it enables the menu functions for exporting the schematic netlist to FastChip.



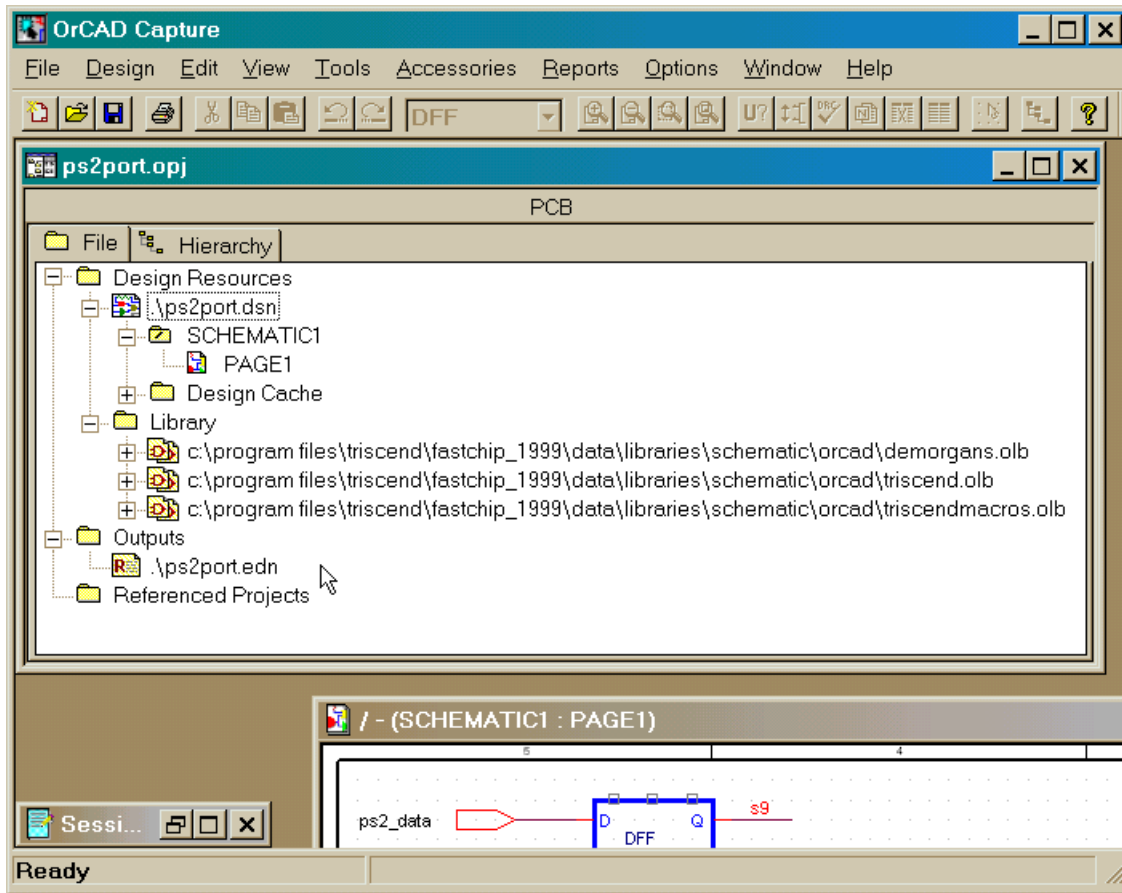
To begin exporting the keyboard interface netlist, click on the Tools⇒Create Netlist... menu item as shown below.



Click on the EDIF 2.0.0 tab in the **Create Netlist** window that appears. FastChip imports modules from EDIF netlists. The only option box that should be checked is the one which outputs pin names instead of numbers in the EDIF file. Then use the Browse button to select the orcad folder under the Chap41 project folder. The keyboard interface netlist will be stored in the PS2PORT.EDN EDIF file in this folder. Click on OK to begin the export process.



Once the netlist has been exported as an EDIF file, you can see it listed in the File tab of the **ps2port.opj** window under the Outputs folder.

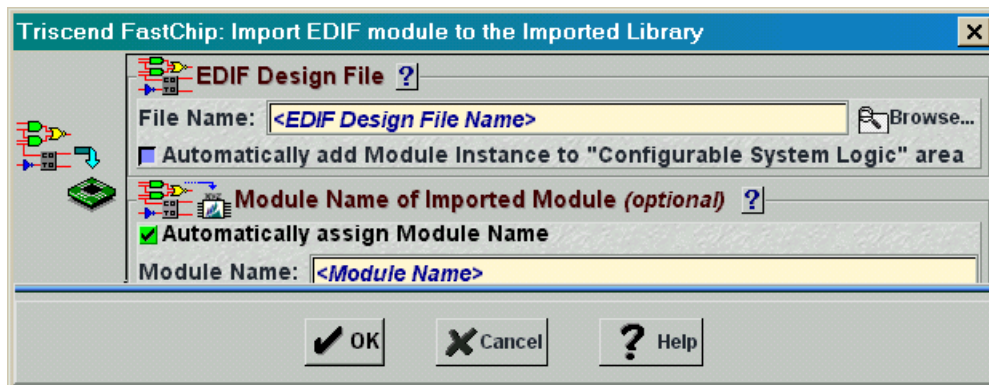


Importing a Netlist into a FastChip Project

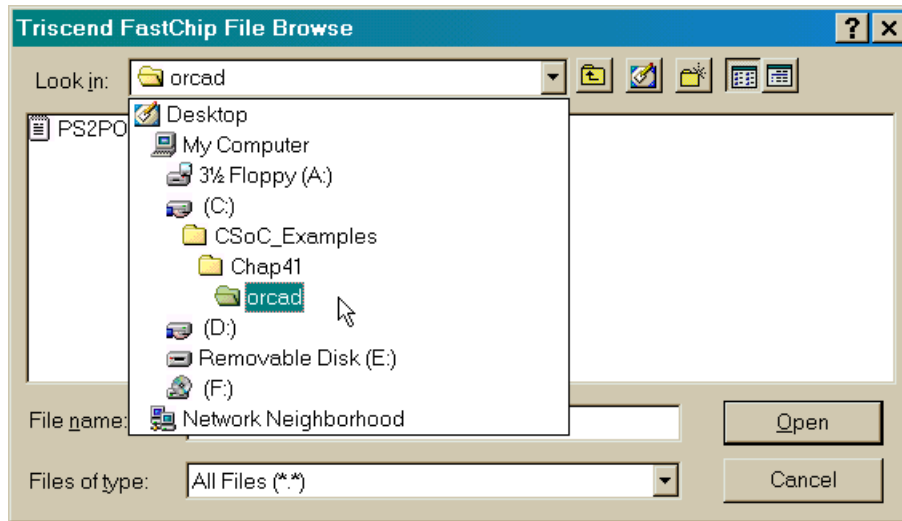
To begin importing the keyboard interface to your FastChip project, click on the



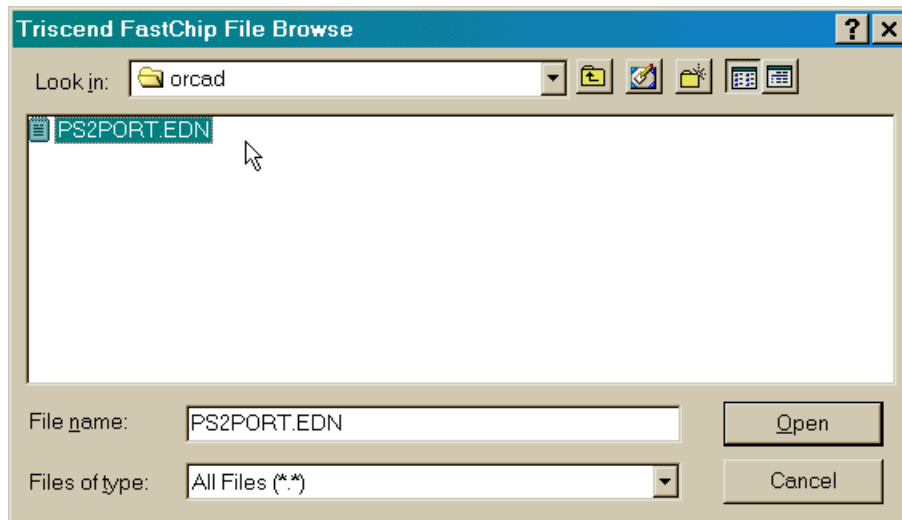
icon in the FastChip project window toolbar. **The Import EDIF module to the Imported Library window appears.**



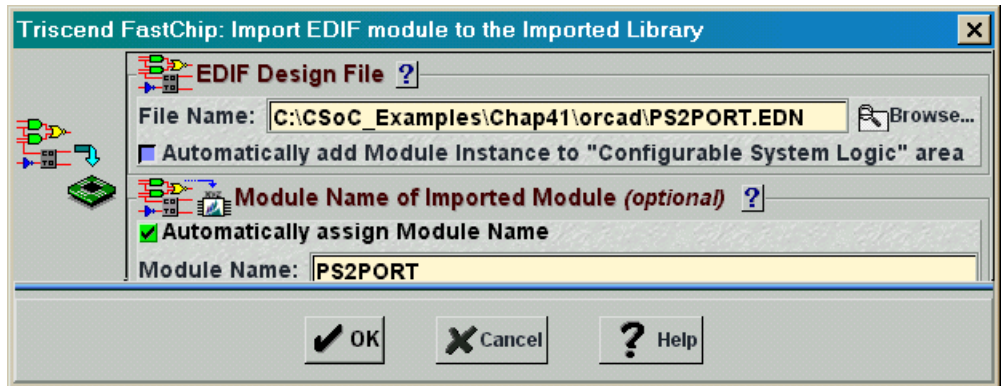
Click on the Browse button and steer your way into the orcad folder under the Chap41 project folder.



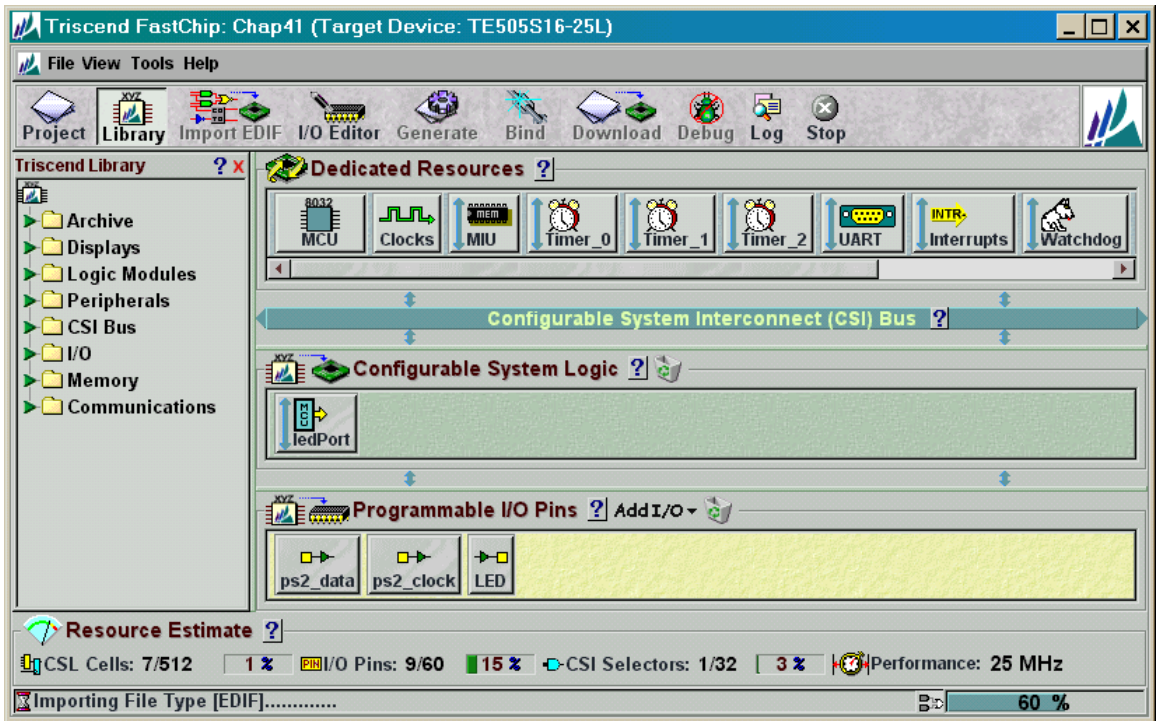
You should highlight the PS2PORT.EDN EDIF file and click on the Open button.



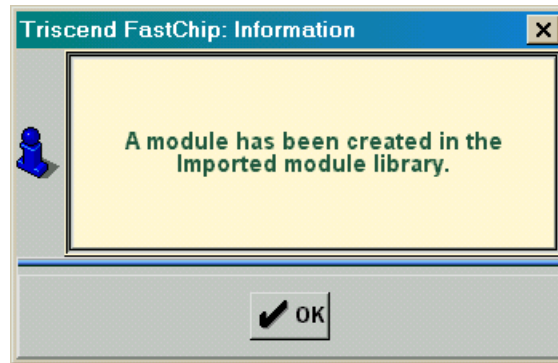
Now the path to the EDIF file should be listed in the File Name box and the Module Name field has been automatically updated with the name of your OrCAD project.



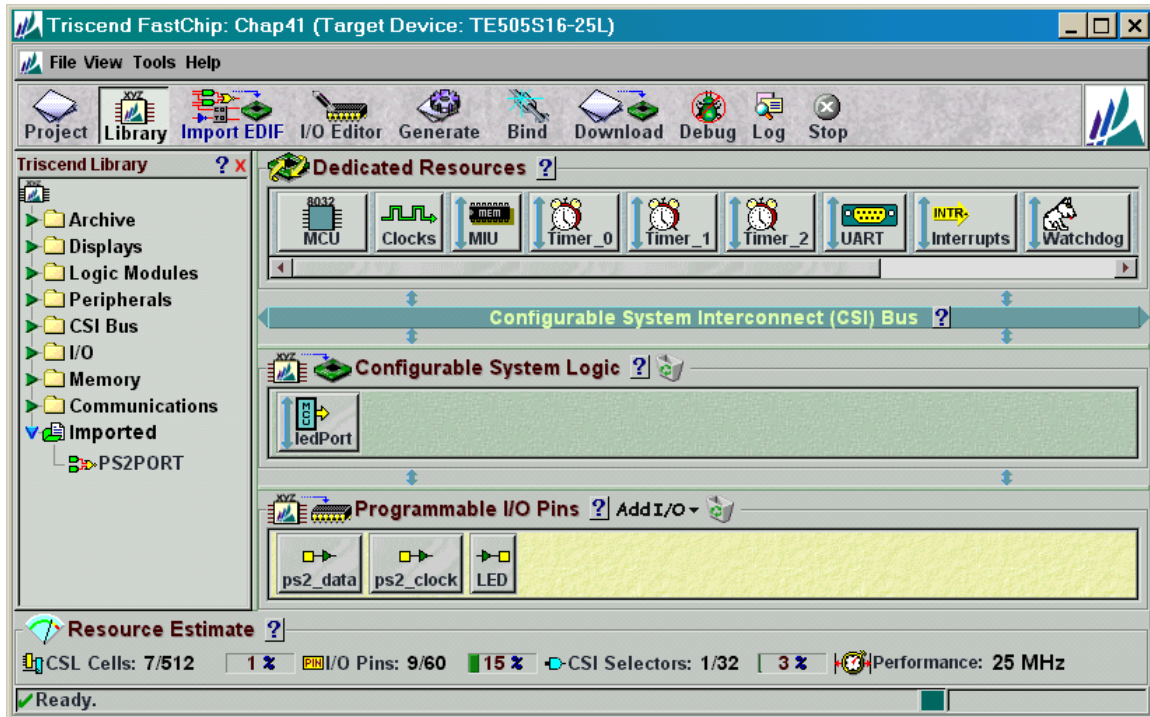
Once you click on OK, the EDIF file import process begins. The progress is reported in the lower right-hand corner of the FastChip project window.



You should receive the following status message after the import process completes. Click on OK to proceed.

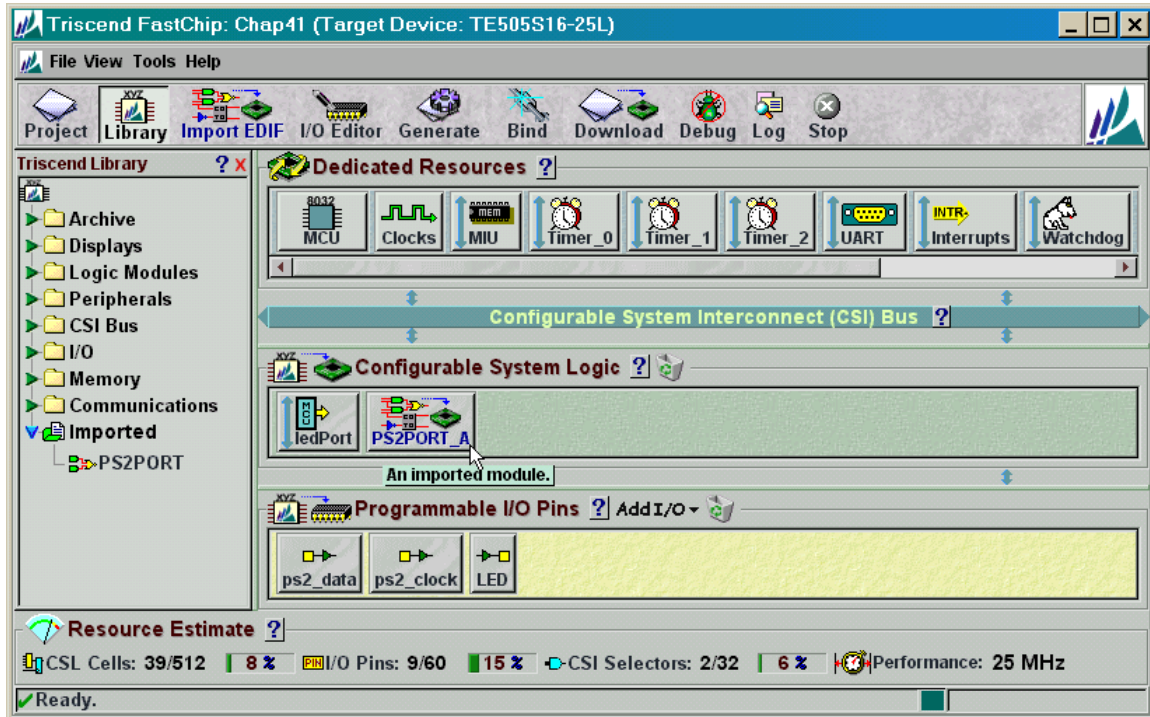


Now you should see a new entry labeled Imported in the Triscend Library area. Expanding this entry shows that the keyboard interface circuit has been added to the library.

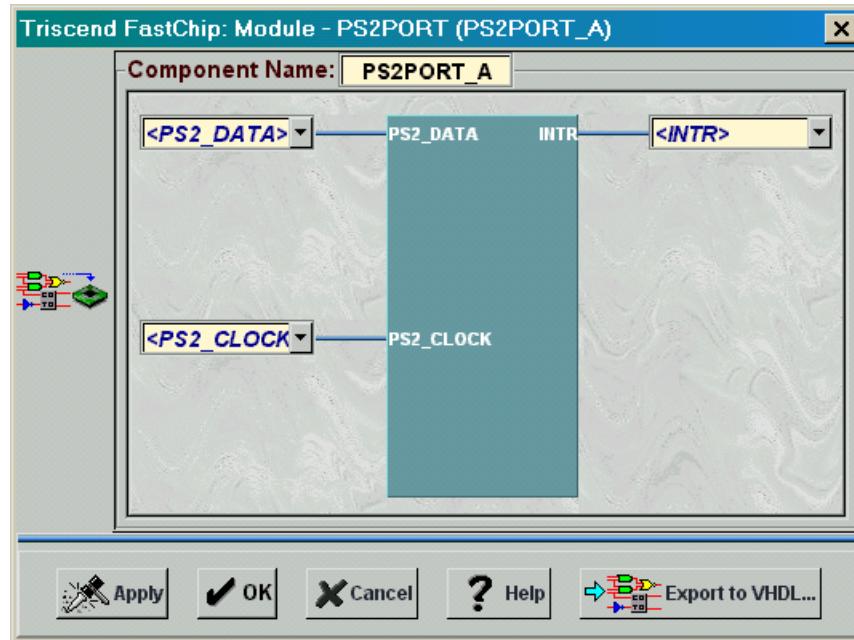


Using the Imported Module

You can drag your newly-imported **PS2PORT** module into the Configurable System Logic area just like any other soft module.

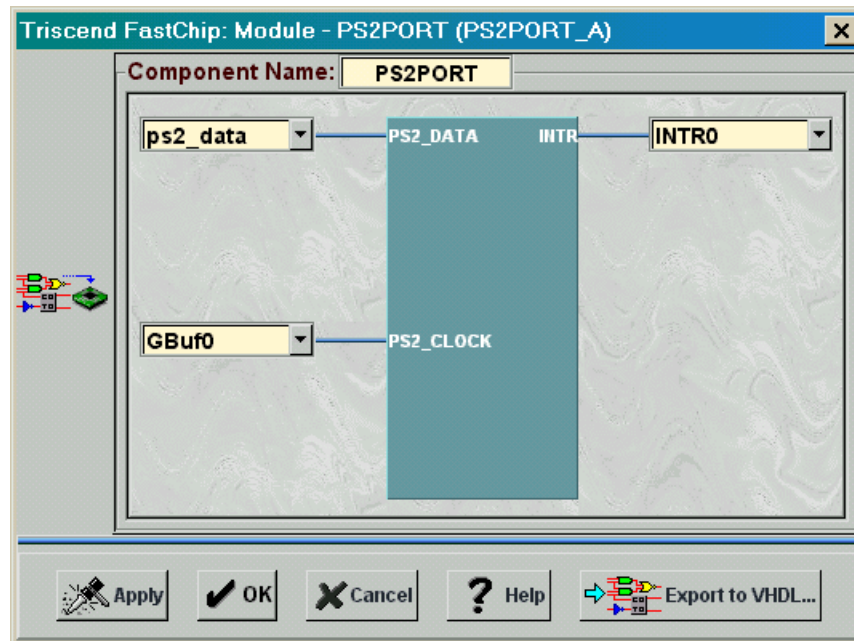


Then click on the **PS2PORT_A** module to make the **Module - PS2PORT** window appear. The input and output ports in the original schematic are arranged along the left and right side of the module block, respectively. The port names in the schematic are also used as the default labels for the input and output nets.

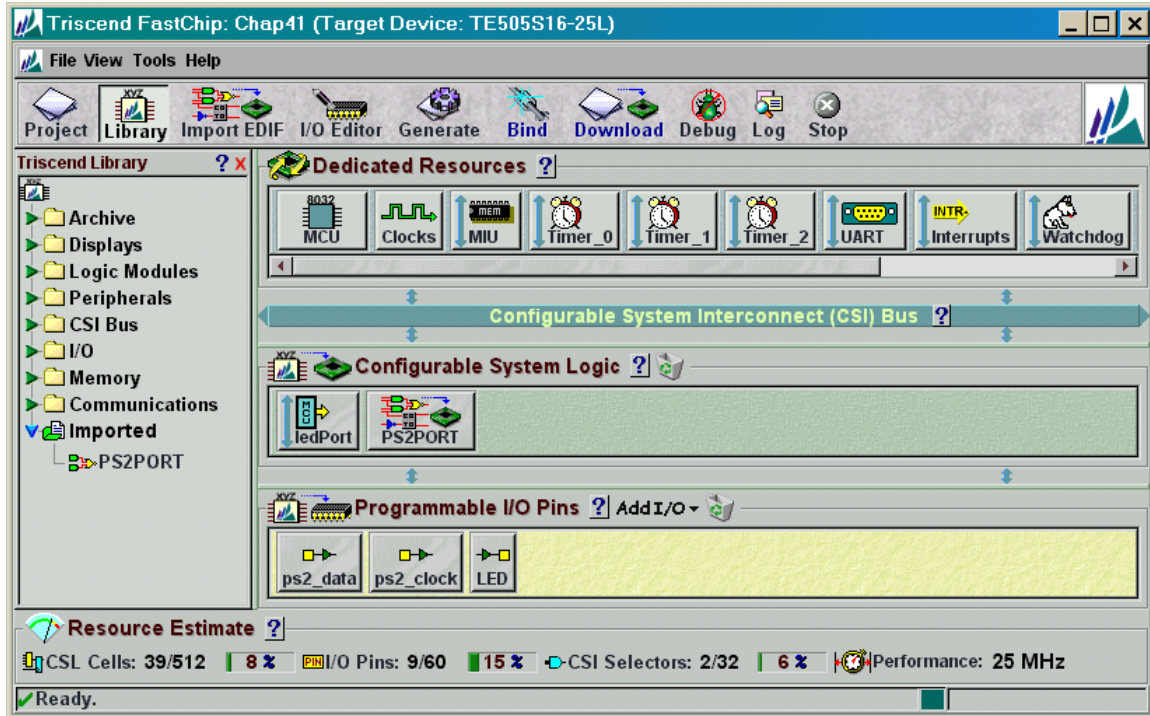


You can rename the module to **PS2PORT**. The **PS2_DATA** input of the keyboard interface should be connected to the **ps2_data** input pin of the FastChip project. The keyboard clock signal arrives through the **ps2_clock** input pin and this drives the **GBuf0** global clock buffer. The output of **GBuf0** drives the clock input of the keyboard interface module. The output which indicates the presence of a keyboard scan code is connected to interrupt signal 0 (**INTR0**) of the 8032 MCU.

Note that some of the keyboard interface module I/O is not explicitly visible. The **BusClock** is implicitly input to the keyboard module by the inclusion of the **BUSCLK** component in the schematic. The interface to the CSI address and data buses is also implicitly defined by the use of the **DR7_OE**, **SELECTOR**, **SIZE**, **SFR**, and **SYMBOLIC** components.



After clicking OK in the **Module - PS2PORT** window, your FastChip project window should appear as follows.



The modules and their interconnections have been instantiated. The pin assignments should already be set as shown in Table 13.

Table 13: Pin assignments and functions for the keyboard interface design.

Signal	Pin	CSoc Board Resource
ps2_clock	47	PS/2 clock input
ps2_data	51	PS/2 data input
LED.0	35	LED digit segment A
LED.1	39	LED digit segment B
LED.2	43	LED digit segment C
LED.3	41	LED digit segment D
LED.4	40	LED digit segment E
LED.5	34	LED digit segment F
LED.6	36	LED digit segment G

Press the Generate icon on the toolbar and FastChip will create the chap41.h header file (Listing 9). Now you can check how the FastChip software handled the **SYMBOLIC** address component attached to the **SELECTOR**. Note that on line 34 the register which holds the scan code is declared to be of type `sfr`. So it was placed in the special function register address space of the 8032 as you specified by using the **SFR** component in your schematic. The name of the register was formed by concatenating the module name (PS2PORT) with the alias for the net that connects the **SYMBOLIC** component to the **SELECTOR** component (RCVDATA).

Listing 9: Top of the header file generated by the FastChip software for the keyboard interface.

```

1 // Generated 5/28/00 9:12 AM By FastChip Version 1999 Build 30
2
3 ///////////////////////////////////////////////////////////////////
4 //
5 // -----
6 // ----- GENERATED CODE -----
7 // -----
8 // The code in this header file was generated automatically for your
9 // project by Triscend FastChip. Please DO NOT EDIT this header file.
10 // It will be overwritten the next time FastChip generates code for
11 // your project.
12 //
13 ///////////////////////////////////////////////////////////////////
14
15 //===== Required symbol and macro definitions =====
16
17 #ifdef PROTOTYPE_ONLY
18 # define CHAR_XDATA(name,location) extern volatile unsigned char xdata name;
19 # define CHAR_ARRAY_XDATA(name,location,size) extern volatile unsigned char
20 xdata name[size];
21 #else
22 # define CHAR_XDATA(name,location) volatile unsigned char xdata name _at_
23 location;
24 # define CHAR_ARRAY_XDATA(name,location,size) volatile unsigned char xdata
25 name[size] _at_ location;
26 #endif
27
28 //===== BEGIN SOFT MODULE REGISTER DECLARATIONS =====
29
30 //----- Module ledPort
31 CHAR_XDATA (ledPort,0xefff)
32
33 //----- Module PS2PORT
34 sfr PS2PORT_RCVDATA = 0x9a;
35
36 //===== END SOFT MODULE REGISTER DECLARATIONS =====

```

The next step is to write your application code. Create a keil folder within your Chap41 FastChip project folder. Then start the Keil IDE and add the C code in Listing 10 to the **chap41** Keil project. The main routine (lines 3-7) just initializes the 8032 MCU and enters an infinite-loop. All the work is actually done in the `displayPs2Data` interrupt subroutine (lines 28-47). This subroutine is called when the keyboard interface

generates an INT0 interrupt (interrupt identifier 0). The interrupt subroutine reads the keyboard scan code from the **PS2PORT_RCVDATA** register on line 33 and this also clears the interrupt flag in the keyboard interface. Then the scan codes in the table defined on lines 13–26 are searched. If a matching scan code is found in the table, the subroutine writes the associated LED segment activation pattern to the **ledPort** register. This displays the digit for the key that was pressed. (The table only has the scan codes for the digits 0–9.) If the received scan code can't be found in the table, the subroutine displays an E on the LED digit.

Listing 10: Keyboard interface interrupt-handling code.

```

1  #include "..\Chap41.h"
2
3  main()
4  {
5      Chap41_INIT();
6      while(1);
7  }
8
9
10 #define ERROR 0x79;
11
12 // translate keyboard scan codes to LED segment activations
13 typedef struct{ unsigned char ps2Data, led; } ps2XlateEntry;
14 ps2XlateEntry ps2XlateTbl[] =
15 {
16     { 0x16, 0x06 }, // "1"
17     { 0x1E, 0x5B }, // "2"
18     { 0x26, 0x4F }, // "3"
19     { 0x25, 0x66 }, // "4"
20     { 0x2E, 0x6D }, // "5"
21     { 0x36, 0x7D }, // "6"
22     { 0x3D, 0x07 }, // "7"
23     { 0x3E, 0x7F }, // "8"
24     { 0x46, 0x6F }, // "9"
25     { 0x45, 0x3F } // "0"
26 };
27
28 static void displayPs2Data() interrupt 0 using 0
29 {
30     unsigned int i;
31     unsigned char c;
32
33     c = PS2PORT_RCVDATA; // get the scan code
34 // (also clears interrupt)
35
36 // search the translation table for the scan code
37     for(i=0; i<sizeof(ps2XlateTbl)/sizeof(ps2XlateEntry); i++)

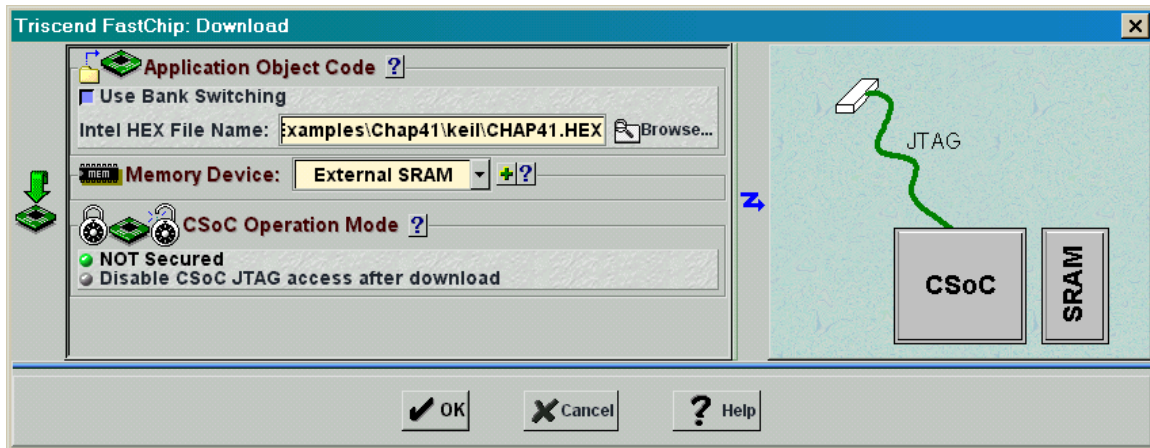
```

```

38         if (ps2XlateTbl[i].ps2Data == c)
39             { // found a matching scan code in the table so
40               // display the digit on the LED
41 ledPort = ps2XlateTbl[i].led;
42             return;
43           }
44
45 // no matching scan code was found, so display "E"
46 ledPort = ERROR;
47 }

```

Once you set the compiler and linker options as you did in the previous chapters, you can compile and link the **Chap41** Keil project. Then re-enter the FastChip project window and bind your design. Download the keyboard interface circuitry and the 8032 program in the Chap41.HEX file to your CSoc Board.



Finally, use dScope to establish a debugging link to the CSoc Board and then reset and execute the application program. At this point, you should be able to type on the numeric keys of a keyboard attached to the PS/2 port of your CSoc Board and see the numbers appear on the LED digit.