How to create a simple ColdFire and Altera FPGA IOC (Draft)

W. Eric Norum

September 28, 2005

Contents

Chapter 1

Introduction

This tutorial presents a step-by-step series of operations for creating a simple EPICS application for an Arcturus uCDIMM ColdFire 5282 module attached to an Altera FPGA development kit. The following software and hardware components are assumed to be in place:

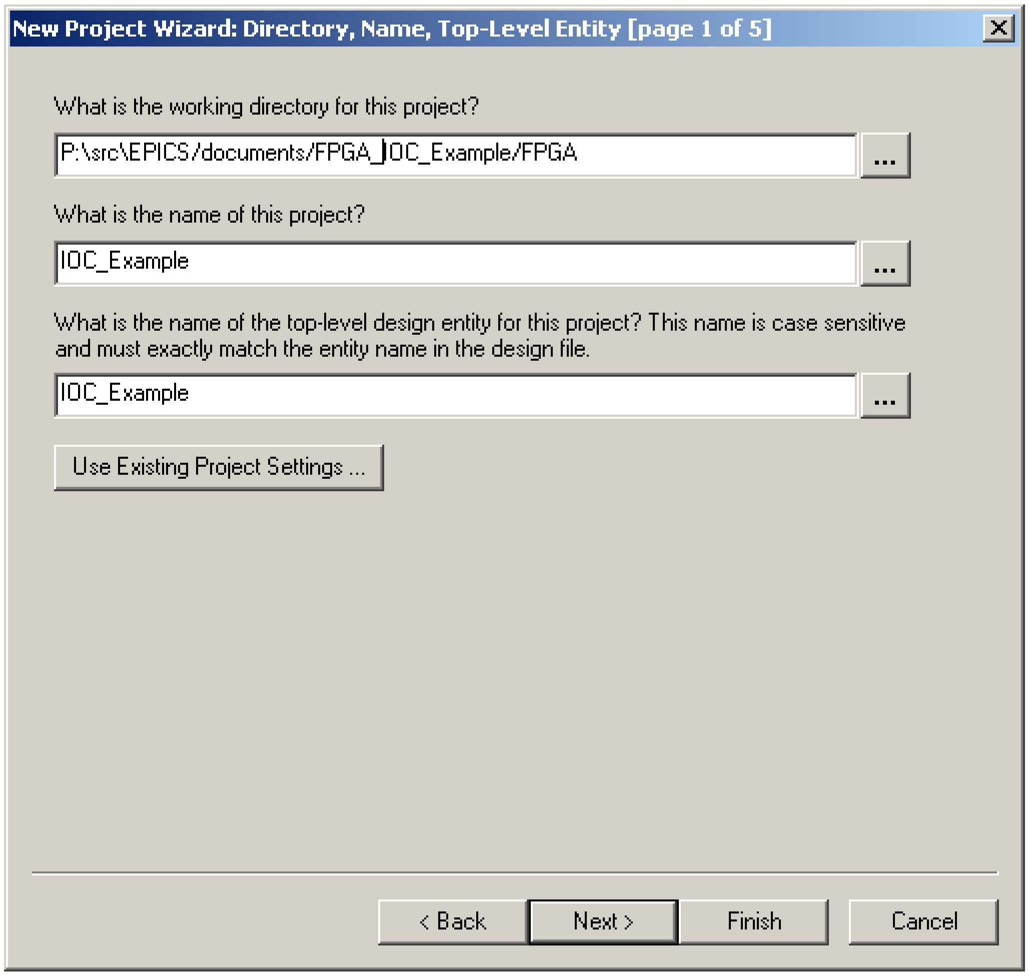
- Arcturus uCDIMM ColdFire 5282
- RTEMS with m68k tool chain and uC5282 board-support package
- EPICS version R3.14.7 or greater with RTEMS-uC5282 target
- Altera FPGA development kit with at least two sets of expansion prototype connectors.
- ColdFire/Stratix adapter card
- Quartus version 5.0 or greater and Altera SOPC builder
- ColdFire bridge SOPC component
- Console reset detect Quartus component (optional)

This tutorial is written for use with an Altera Stratix II DSP development kit. The changes required for use with other development kits are summarized in appendix B.

Chapter 2

FPGA application

1. Use the Quartus "New Project Wizard" to create a new project. I called the project `IOC_Example` with same name for the top-level entity:


New Project Wizard: Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

P:\src\EPICS/documents/FPGA_IOC_Example/FPGA

What is the name of this project?

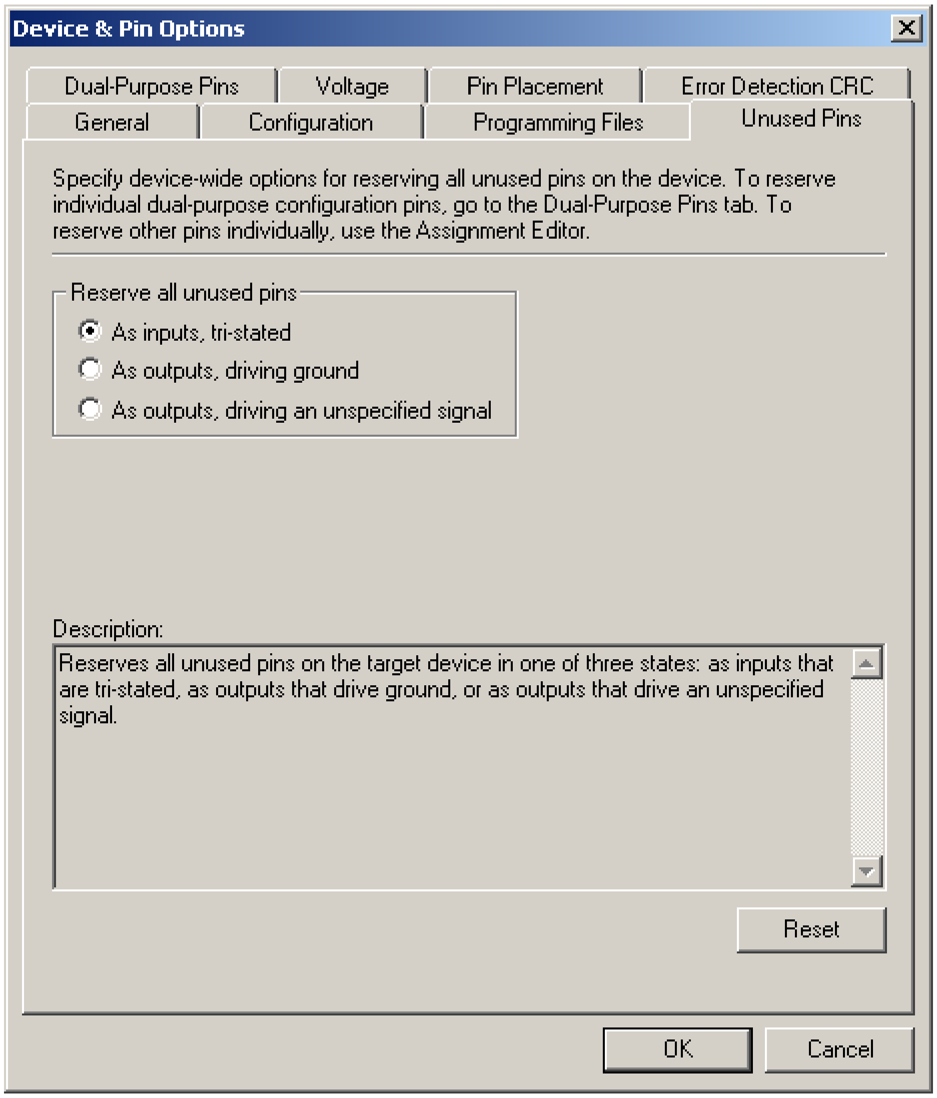
IOC_Example

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

IOC_Example

2. Ensure that unused pins are treated as inputs (Assignments→Device..., Device & Pin Options, Unused Pins tab). Not all the ColdFire signals are used by this example and Quartus helpfully grounds all the corresponding

pins if this step is omitted!



Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor.

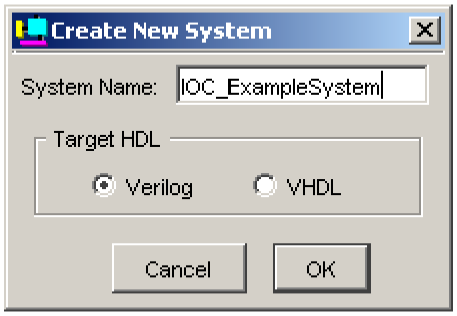
Reserve all unused pins

[✓ As inputs, tri-stated]
As outputs, driving ground
As outputs, driving an unspecified signal

Description:

Reserves all unused pins on the target device in one of three states: as inputs that are tri-stated, as outputs that drive ground, or as outputs that drive an unspecified signal.

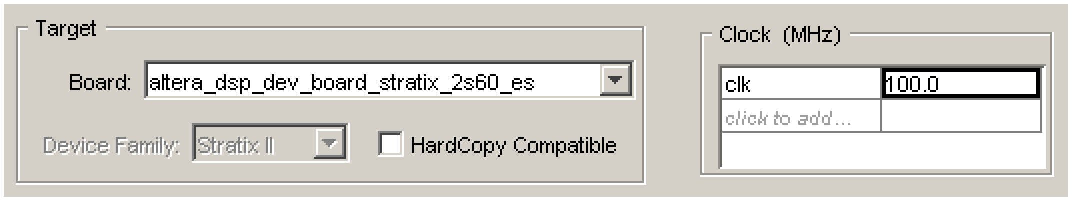
3. Start the SOPC Builder (Tools→SOPC Builder. . .) and create a new SOPC system. I called the SOPC system `IOC_ExampleSystem`:



System Name: IOC_ExampleSystem

Target HDL: [✓ Verilog] VHDL

4. Set the SOPC Board and Clock parameters:

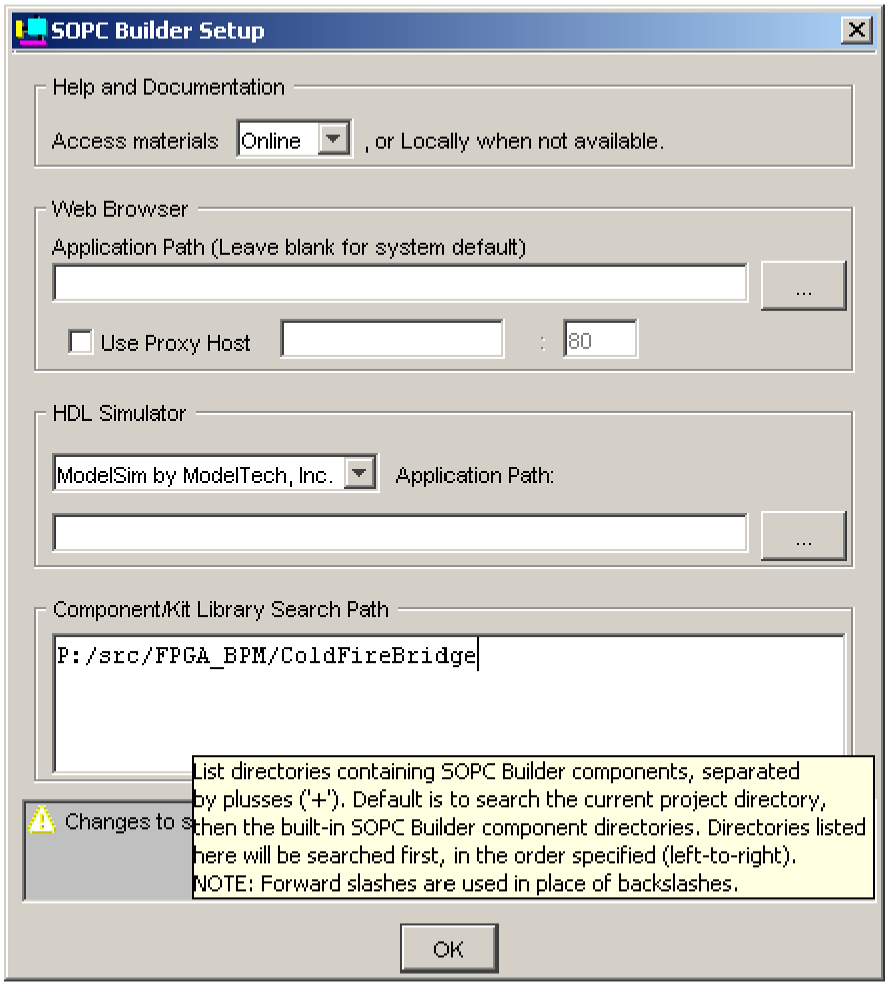


Board: altera_dsp_dev_board_stratix_2s60_es

Device Family: Stratix II

clk	100.0
click to add...	

5. If you have not done so in a previous SOPC project, add the directory containing the ColdFire Bridge SOPC component to the list of directories which the SOPC Builder will search (File→SOPC Builder Setup. . .). Note that this list of directories applies to SOPC Builder sessions in **all** projects. Note also that changes to this list do not take effect until the SOPC Builder is next started. When the SOPC Builder is started and searches the directory containing the ColdFire Bridge SOPC component it will add the ColdFire Bridge component to the list of User Logic components.



Access materials Online , or Locally when not available.

Application Path (Leave blank for system default)

Use Proxy Host : 80

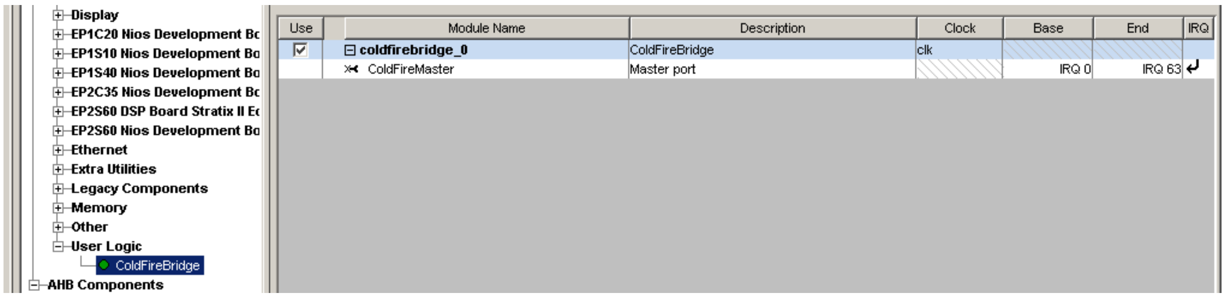
ModelSim by ModelTech, Inc. Application Path:

P:/src/FPGA_BPM/ColdFireBridge

Changes to s

List directories containing SOPC Builder components, separated by plusses ('+'). Default is to search the current project directory, then the built-in SOPC Builder component directories. Directories listed here will be searched first, in the order specified (left-to-right). NOTE: Forward slashes are used in place of backslashes.

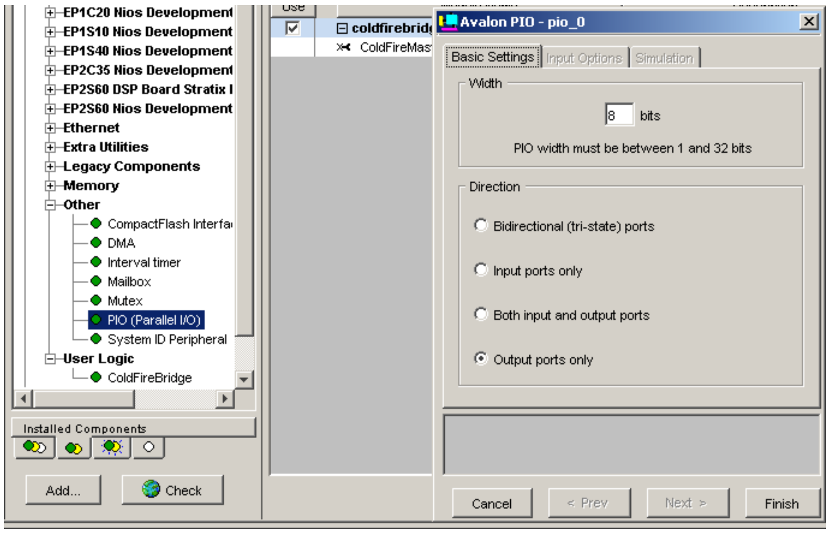
6. Double-click the ColdFire Bridge component to add it to the SOPC design:



Display
EP1C20 Nios Development Bc
EP1S10 Nios Development Bo
EP1S40 Nios Development Bo
EP2C35 Nios Development Bc
EP2S60 DSP Board Stratix II Ec
EP2S60 Nios Development Bo
Ethernet
Extra Utilities
Legacy Components
Memory
Other
User Logic
ColdFireBridge
AHB Components

Use	Module Name	Description	Clock	Base	End	IRQ
☑	⊟ **coldfirebridge_0**	ColdFireBridge	clk			
	ColdFireMaster	Master port		IRQ 0	IRQ 63	

7. Add an 8-bit output port to the SOPC design:



EP1C20 Nios Development
EP1S10 Nios Development
EP1S40 Nios Development
EP2C35 Nios Development
EP2S60 DSP Board Stratix I
EP2S60 Nios Development
Ethernet
Extra Utilities
Legacy Components
Memory
Other
CompactFlash Interfa
DMA
Interval timer
Mailbox
Mutex
PIO (Parallel I/O)
System ID Peripheral
User Logic
ColdFireBridge

Installed Components

Add... Check

coldfirebrid
ColdFireMas

Avalon PIO - pio_0

Basic Settings | Input Options | Simulation

Width

8 bits

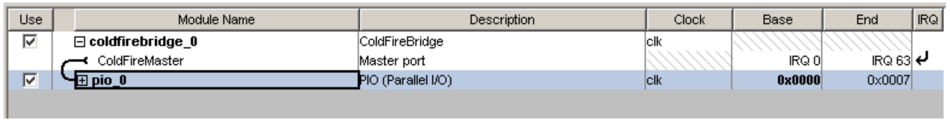
PIO width must be between 1 and 32 bits

Direction

( ) Bidirectional (tri-state) ports
( ) Input ports only
( ) Both input and output ports
(•) Output ports only

Cancel | < Prev | Next > | Finish

The SOPC components and connections should then be:

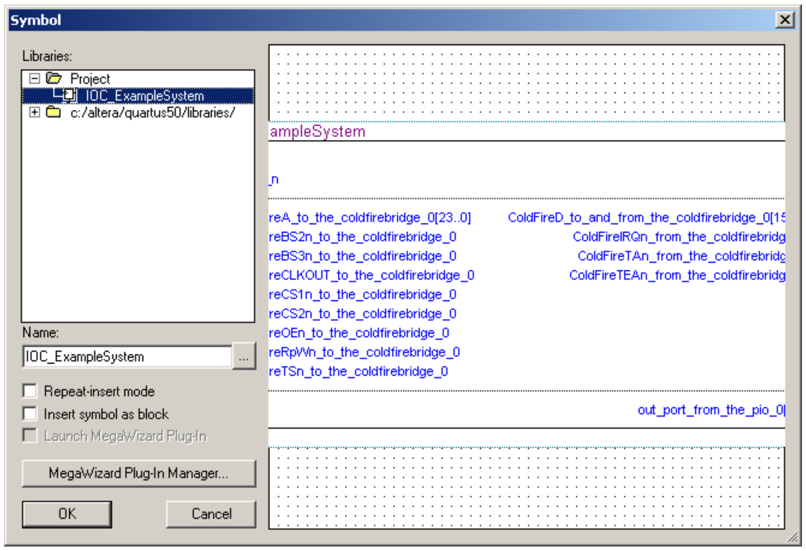


Use	Module Name	Description	Clock	Base	End	IRQ
☑	⊟ **coldfirebridge_0**	ColdFireBridge	clk			
	ColdFireMaster	Master port		IRQ 0	IRQ 63	
☑	⊞ **pio_0**	PIO (Parallel I/O)	clk	**0x0000**	0x0007	

8. Click the `Generate` button to create the SOPC block.

9. Create the top-level entity design file. In the Quartus window select File→New, create a new Block Diagram/Schematic File and save it as `IOC_Example`.

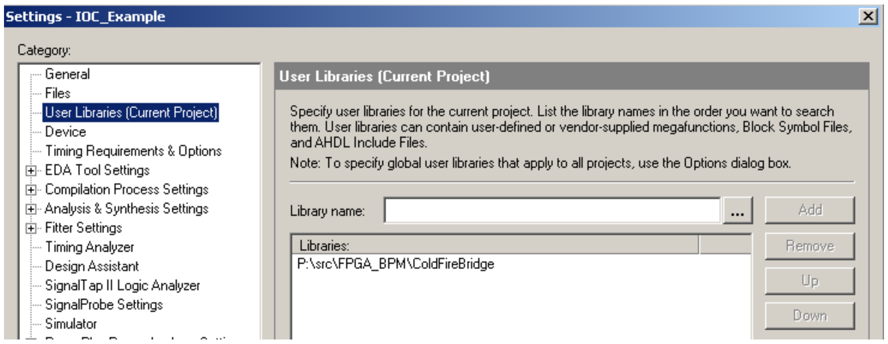
10. Double-click in the `IOC_Example.bdf` design window and add the `IOC_ExampleSystem` SOPC block to the design:


Symbol
Libraries:
Project
IOC_ExampleSystem
c:/altera/quartus50/libraries/
Name:
IOC_ExampleSystem
Repeat-insert mode
Insert symbol as block
Launch MegaWizard Plug-In
MegaWizard Plug-In Manager...
OK
Cancel
ampleSystem
_n
reA_to_the_coldfirebridge_0[23..0]
reBS2n_to_the_coldfirebridge_0
reBS3n_to_the_coldfirebridge_0
reCLKOUT_to_the_coldfirebridge_0
reCS1n_to_the_coldfirebridge_0
reCS2n_to_the_coldfirebridge_0
reOEn_to_the_coldfirebridge_0
reRpWn_to_the_coldfirebridge_0
reTSn_to_the_coldfirebridge_0
ColdFireD_to_and_from_the_coldfirebridge_0[15
ColdFireIRQn_from_the_coldfirebridg
ColdFireTAn_from_the_coldfirebridg
ColdFireTEAn_from_the_coldfirebridg
out_port_from_the_pio_0[

11. Add I/O pins and assign them to the appropriate FPGA pins using the Assignment Editor. A list of the pin assignments for several different development kits is included in appendix B. The complete system should then appear as shown on the following page.

12. Make some changes to some settings (Assignments→Settings. . .):

    User Libraries (Current Project) Add the path to the directory containing the ColdFire Bridge SOPC component:

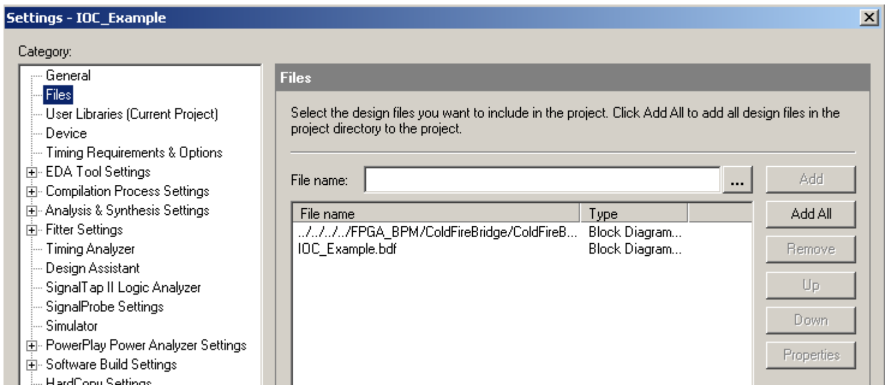


Specify user libraries for the current project. List the library names in the order you want to search them. User libraries can contain user-defined or vendor-supplied megafunctions, Block Symbol Files, and AHDL Include Files.

Note: To specify global user libraries that apply to all projects, use the Options dialog box.

P:\src\FPGA_BPM\ColdFireBridge

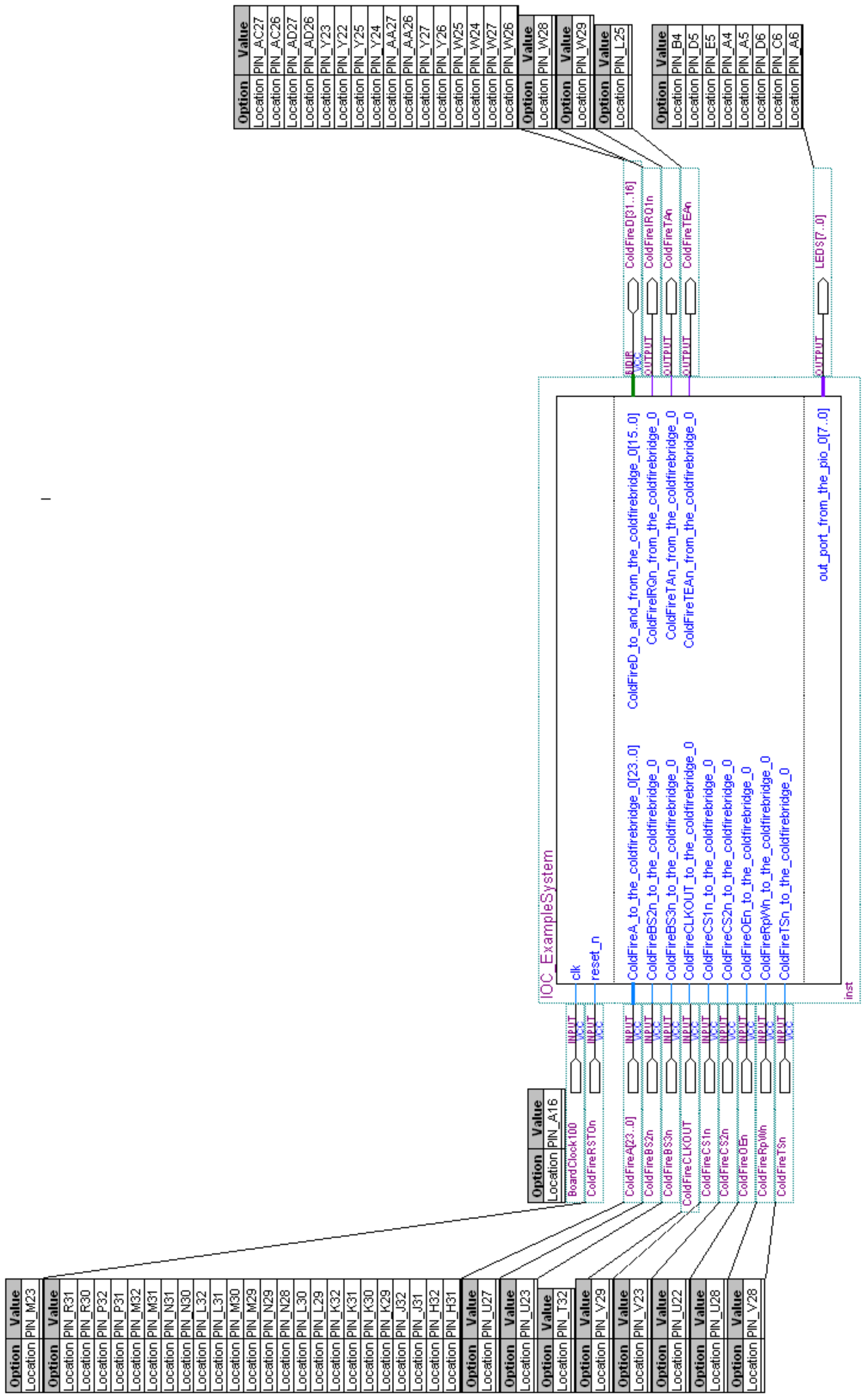
    Files Quartus II has a bug which causes it to generate a bad system unless the `ColdFireBridge.bdf` design file is explicitly mentioned in the list of application files:



Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name	Type
../../../../FPGA_BPM/ColdFireBridge/ColdFireB...	Block Diagram...
IOC_Example.bdf	Block Diagram...

13. Compile the project (you’ll see lots of warnings. . .) and load it into the FPGA.


IOC_ExampleSystem
inst
clk
reset_n
ColdFireA_to_the_coldfirebridge_0[23..0]
ColdFireBS2n_to_the_coldfirebridge_0
ColdFireBS3n_to_the_coldfirebridge_0
ColdFireCLKOUT_to_the_coldfirebridge_0
ColdFireCS1n_to_the_coldfirebridge_0
ColdFireCS2n_to_the_coldfirebridge_0
ColdFireOEn_to_the_coldfirebridge_0
ColdFireRpWn_to_the_coldfirebridge_0
ColdFireTSn_to_the_coldfirebridge_0
ColdFireD_to_and_from_the_coldfirebridge_0[15..0]
ColdFireIRQn_from_the_coldfirebridge_0
ColdFireTAn_from_the_coldfirebridge_0
ColdFireTEAn_from_the_coldfirebridge_0
out_port_from_the_pio_0[7..0]
BoardClock100
ColdFireRSTOn
ColdFireA[23..0]
ColdFireBS2n
ColdFireBS3n
ColdFireCLKOUT
ColdFireCS1n
ColdFireCS2n
ColdFireOEn
ColdFireRpWn
ColdFireTSn
ColdFireD[31..16]
ColdFireIRQ1n
ColdFireTAn
ColdFireTEAn
LEDS[7..0]
INPUT
VCC
OUTPUT
BIDIR

Option	Value
Location	PIN_AC27
Location	PIN_AC26
Location	PIN_AD27
Location	PIN_AD26
Location	PIN_Y23
Location	PIN_Y22
Location	PIN_Y25
Location	PIN_Y24
Location	PIN_AA27
Location	PIN_AA26
Location	PIN_Y27
Location	PIN_Y26
Location	PIN_W25
Location	PIN_W24
Location	PIN_W27
Location	PIN_W26

Option	Value
Location	PIN_W28

Option	Value
Location	PIN_W29

Option	Value
Location	PIN_L25

Option	Value
Location	PIN_B4
Location	PIN_D5
Location	PIN_E5
Location	PIN_A4
Location	PIN_A5
Location	PIN_D6
Location	PIN_C6
Location	PIN_A6

Option	Value
Location	PIN_A16

Option	Value
Location	PIN_M23

Option	Value
Location	PIN_R31
Location	PIN_R30
Location	PIN_P32
Location	PIN_P31
Location	PIN_M32
Location	PIN_M31
Location	PIN_N31
Location	PIN_N30
Location	PIN_L32
Location	PIN_L31
Location	PIN_M30
Location	PIN_M29
Location	PIN_N29
Location	PIN_N28
Location	PIN_L30
Location	PIN_L29
Location	PIN_K32
Location	PIN_K31
Location	PIN_K30
Location	PIN_K29
Location	PIN_J32
Location	PIN_J31
Location	PIN_H32
Location	PIN_H31

Option	Value
Location	PIN_U27

Option	Value
Location	PIN_U23

Option	Value
Location	PIN_T32

Option	Value
Location	PIN_V29

Option	Value
Location	PIN_V23

Option	Value
Location	PIN_U22

Option	Value
Location	PIN_U28

Option	Value
Location	PIN_V28

Chapter 3

EPICS application

The steps listed below show how to create an example EPICS IOC application which uses the ASYN I/O environment to control the LEDs on the FPGA card.

1. Create a new EPICS `<TOP>` directory and make a new application in it. You must specify the full path to the makeBaseApp.pl script in your EPICS installation:

   ```
   /···/makeBaseApp.pl -t ioc ledDriver
   /···/makeBaseApp.pl -t ioc -i -a RTEMS-uC5282 ledDriver
   ```

2. Edit configure/CONFIG to enable only the RTEMS-uC5282 target:

   ```
   CROSS_COMPILER_TARGET_ARCHS = RTEMS-uC5282
   ```

3. Edit configure/RELEASE to specify the location of ASYN support:

   ```
   ASYN=/···/modules/soft/asyn
   ```

4. Edit ledDriverApp/src/Makefile:

   - Build only for RTEMS IOC targets (change the `PROD_IOC` line to `PROD_RTEMS`):

     ```
     PROD_RTEMS = ledDriver
     ```

   - Add asyn support:

     ```
     ledDriver_DBD += asyn.dbd
     ```

   - Add the asyn library to `ledDriver_LIBS` (before the `EPICS_BASE_IOC_LIBS` line which is already there):

     ```
     ledDriver_LIBS += asyn
     ledDriver_LIBS += $(EPICS_BASE_IOC_LIBS)
     ```

   - Add the FPGA device support dbd file (to be written in a following step):

     ```
     ledDriver_DBD += ledDeviceSupport.dbd
     ```

   - Add the FPGA device support source file (to be written in a following step):

     ```
     ledDriver_SRCS += ledDeviceSupport.c
     ```

5. Edit ledDriverApp/Db/Makefile and add the line:

```
DB += ledDriver.db
```

6. Create the ledDriverApp/Db/ledDriver.db file referred to in the previous step. The file should contain:

   ```
   record(longout,"leds") {
       field(DTYP,"asynInt32")
       field(OUT,"@asyn(ledDriver 0 0)")
   }
   ```

   The three values in the `OUT` field are:

   (a) The port name.

   (b) The address (unused by this driver, but still needed)

   (c) The timeout value, in seconds (unused by this driver, but still needed).

7. Create ledDriverApp/src/ledDeviceSupport.dbd with the contents:

   ```
   registrar(ledDriverDeviceSupportRegistrar)
   ```

8. Create ledDriverApp/src/ledDeviceSupport.c. A complete listing of is included in appendix A. Much of this file is common to all ASYN drivers. The following points describe the lines of particular interest to this application.

   12 When the SOPC Builder generates a system it creates a C header file describing the system components. This header file can be included explicitly or, as I have done, can be symbolically linked to a file in the application source directory:

   ```
   ln -s ../../../FPGA/coldfirebridge_0_map/system.h ledDriverApp/src/SOPC.h
   ```

   14 The SOPC address space appears in the ColdFire address space at the locations mapped to $\overline{\text{CS1}}$ and $\overline{\text{CS2}}$. The RTEMS board-support package sets up these chip selects at locations 30000000_{16} and 31000000_{16}, respectively.

   15 The '+1' is required because the ColdFire bus is 16-bit big-endian so the least-significant byte on the data bus appears at an odd address.

   23 The table of asynInt32 methods can't be statically initialized (like the asynCommon methods are at line 55) since the asynInt32Base initialize method (invoked at line 99) writes to the table to override some methods.

   41 The check for the existence of the I/O port turns off all of the LEDs. A real application might want to do something different.

   65 The line of code that actually performs the I/O operation.

   72 More complete device support would probably do nothing more in this routine than register an iocsh command. The actual ASYN registration calls would then be called from this command when invoked by the st.cmd script. This would allow the port name to be set from the st.cmd script rather than being burned into the program.

   80 The arguments to the registerPort method are:

   (a) The port name.

   (b) The port attributes. This driver is is not 'multi device' and does not block.

   (c) The autoconnect flag. This driver wants to be automatically reconnected.

   (d) The priority of the I/O thread (unused for this driver).

   (e) The stack size of the I/O thread (unused for this driver).

9. Run `make` to compile the application.

10. Use the uCDIMM ColdFire 5282 setenv command to set environment variables as shown below. The exact values will differ as appropriate for your network numbers and NFS server:

```
B$ setenv IPADDR0 www.xxx.yyy.56
B$ setenv HOSTNAME ioccoldfire2
B$ setenv BOOTFILE ucdimm.boot
B$ setenv NAMESERVER www.xxx.yyy.167
B$ setenv NETMASK 255.255.252.0
B$ setenv SERVER www.xxx.yyy.167
B$ setenv NFSMOUNT nfsserver:/export/homes:/home
B$ setenv CMDLINE /···/FPGA_IOC_Example/EPICS/iocBoot/iocledDriver/st.cmd
B$ printenv
FACTORY=Arcturus Networks Inc.
REVISION=uC5282 Rev 1.0 4MB External Flash
SERIAL=X42B20ADC-0130C
CONSOLE=ttyS0
KERNEL=0:linux.bin
KERNEL_ARGS=root=/dev/rom0
HWADDR0=00:06:3B:00:53:0C
FW_VERSION=180001
_0=10000000:400000:RW
RAMIMAGE=yes
IPADDR0=www.xxx.yyy.56
CACHE=on
HOSTNAME=ioccoldfire2
BOOTFILE=ucdimm.boot
NAMESERVER=www.xxx.yyy.167
NETMASK=255.255.252.0
SERVER=www.xxx.yyy.167
NFSMOUNT=nfssrv:/export/homes:/home
CMDLINE=.../FPGA_IOC_Example/EPICS/iocBoot/iocledDriver/st.cmd
```

11. Download and execute the application:

   - Start the TFTP server on the ColdFire:

   ```
   B$ tftp
   uCTFTP Console 1.0 is running ...
   ```

   - Use the tftp program on your workstation to transfer the executable image to the ColdFire:

   ```
   tftp> binary
   tftp> connect www.xxx.yyy.56
   tftp> put ledDriver.boot
   ```

   - When the executable image has been transferred press the `<ESC>` key to the ColdFire to terminate the TFTP server. Use the `goram` command to start the IOC:

   ```
   Downloading..............................................
   B$ goram
   Go from RAM!
   Go from 0x40000
   NTPSERVER environment variable missing -- using www.xxx.yyy.167

   ***** Initializing network *****
   ```

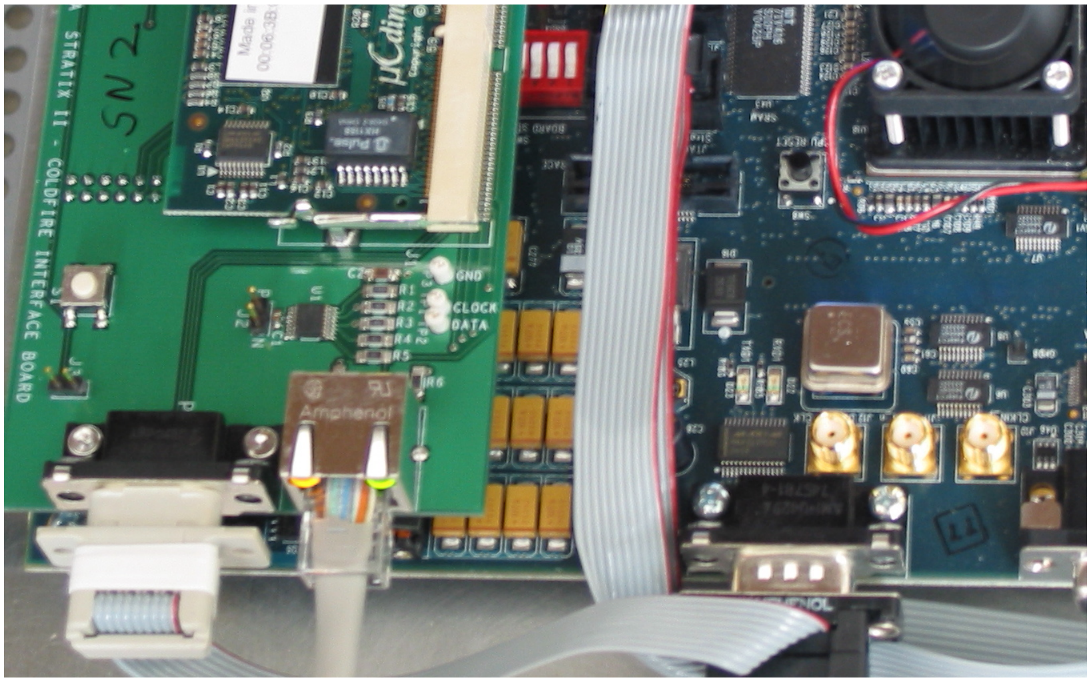
```
Startup after External reset.
fs1: Ethernet address: 00:06:3b:00:53:0c
***** Initializing NFS *****
This is RTEMS-RPCIOD Release $Name:  $
($Id: tutorial.tex,v 1.13 2005/09/28 22:11:06 norume Exp $)

Till Straumann, Stanford/SLAC/SSRL 2002
See LICENSE file for licensing info
This is RTEMS-NFS $Name:  $
($Id: tutorial.tex,v 1.13 2005/09/28 22:11:06 norume Exp $)

Till Straumann, Stanford/SLAC/SSRL 2002
See LICENSE file for licensing info
Trying to mount www.xxx.yyy.167:/export/homes on /home
***** Initializing NTP *****
***** Starting EPICS application *****
## Example RTEMS startup script
## You may have to change ledDriver to something else
## everywhere it appears in this file
#< envPaths
## Register all support components
dbLoadDatabase("../../dbd/ledDriver.dbd",0,0)
ledDriver_registerRecordDeviceDriver(pdbbase)
## Load record instances
dbLoadRecords("../../db/ledDriver.db","user=norume")
iocInit()
Starting iocInit
############################################################################
###  EPICS IOC CORE built on Jul 25 2005
###  EPICS R3.14.7 $$Name:  $$ $$Date: 2005/09/28 22:11:06 $$
############################################################################
iocInit: All initialization complete
## Start any sequence programs
#seq sncledDriver,"user=norume"
ioccoldfire2>
```

Chapter 4

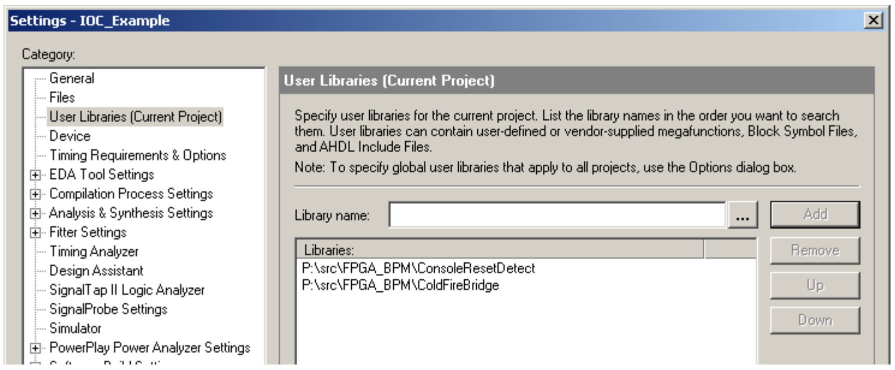
Remote Reset

All VME IOCs at the APS have a card which monitors the console received-data line and generates a system reset when a particular sequence (three consecutive control-X, control-Y or control-Z characters or any consecutive combinations of these characters) of characters is detected. It is quite easy to add this capability to the ColdFire/FPGA IOC.

1. Make a special serial line cable which will connect the serial received-data and ground lines of the ColdFire console port to the 9-pin connector on the FPGA development kit. The following picture shows an example. The 9-pin connector plugged in to the FPGA development kit has only two pins (received-data and ground).



2. Add the path to the directory containing the Console Reset Detect component to the list of application user libraries:


Settings - IOC_Example

Category:

- General
- Files
- User Libraries (Current Project)
- Device
- Timing Requirements & Options
- EDA Tool Settings
- Compilation Process Settings
- Analysis & Synthesis Settings
- Fitter Settings
- Timing Analyzer
- Design Assistant
- SignalTap II Logic Analyzer
- SignalProbe Settings
- Simulator
- PowerPlay Power Analyzer Settings

User Libraries (Current Project)

Specify user libraries for the current project. List the library names in the order you want to search them. User libraries can contain user-defined or vendor-supplied megafunctions, Block Symbol Files, and AHDL Include Files.

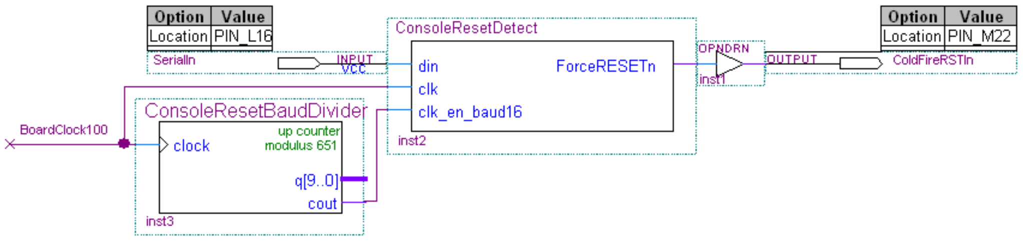
Note: To specify global user libraries that apply to all projects, use the Options dialog box.

Library name:

Libraries:

```
P:\src\FPGA_BPM\ConsoleResetDetect
P:\src\FPGA_BPM\ColdFireBridge
```

3. Add the following components to the application top-level entity (`IOC_Example.bdf`). The ConsoleResetBaudDivider is a simple modulus-651 counter which sets the serial line speed at 9600 baud ($100000000 \div (9600 \times 16) = 651$). If you're using a different system clock or a different serial line speed you'll have to replace this component with your own counter.


Option	Value
Location	PIN_L16

Option	Value
Location	PIN_M22

4. Recompile the project and load it into the FPGA. You should now have the ability to remotely reset the ColdFire.

Appendix A

ledDeviceSupport.c

```
/*
 * ASYN Int32 driver for simple FPGA application
 */
#include <epicsStdio.h>
#include <epicsExport.h>
#include <cantProceed.h>

#include <asynDriver.h>
#include <asynInt32.h>
#include <devLib.h>

#include "SOPC.h"   /* Symbolic link to SOPC-generated system header file */

#define AVALON_BASE  0x30000000    /* Base of Avalon space in ColdFire space */
#define OPTR ((epicsUInt8 *)(AVALON_BASE+PIO_0_BASE+1))

/*
 * Driver private storage
 */
typedef struct drvPvt {
    asynInterface common;
    asynInterface asynInt32;
    asynInt32     asynInt32Methods;
    volatile epicsUInt8 *optr;
} drvPvt;

/*
 * asynCommon methods
 */
static void
report(void *pvt, FILE *fp, int details)
{
}

static asynStatus
connect(void *pvt, asynUser *pasynUser)
```

```
{
    drvPvt *pdrvPvt = (drvPvt *)pvt;
    epicsUInt8 dummy = 0;

    if (devWriteProbe(sizeof(dummy), pdrvPvt->optr, &dummy) != 0) {
        asynPrint(pasynUser, ASYN_TRACE_ERROR, "ledDriver: memory probe failed\n");
        return asynError;
    }
    pasynManager->exceptionConnect(pasynUser);
    return asynSuccess;
}

static asynStatus
disconnect(void *pvt, asynUser *pasynUser)
{
    pasynManager->exceptionDisconnect(pasynUser);
    return asynSuccess;
}
static asynCommon common = { report, connect, disconnect };

/*
 * asynInt32 methods
 */
static asynStatus
int32Write(void *pvt, asynUser *pasynUser, epicsInt32 value)
{
    drvPvt *pdrvPvt = (drvPvt *)pvt;

    *pdrvPvt->optr = value;
    return asynSuccess;
}

/*
 * Register ourself with ASYN
 */
static void ledDriverDeviceSupportRegistrar(void)
{
    drvPvt *pdrvPvt;
    asynStatus status;
    const char *portName = "ledDriver";

    pdrvPvt = callocMustSucceed(sizeof(drvPvt), 1, "ledDriver");

    status = pasynManager->registerPort(portName, 0, 1, 0, 0);
    if(status != asynSuccess) {
        printf("ledDriver registerDriver failed\n");
        return;
    }

    pdrvPvt->common.interfaceType = asynCommonType;
    pdrvPvt->common.pinterface  = (void *)&common;
    pdrvPvt->common.drvPvt = pdrvPvt;
    status = pasynManager->registerInterface(portName, &pdrvPvt->common);
    if (status != asynSuccess) {
        printf("ledDriver registerInterface failed\n");
        return;
    }

    pdrvPvt->asynInt32Methods.write = int32Write;
    pdrvPvt->asynInt32.interfaceType = asynInt32Type;
    pdrvPvt->asynInt32.pinterface  = &pdrvPvt->asynInt32Methods;
    pdrvPvt->asynInt32.drvPvt = pdrvPvt;
    status = pasynInt32Base->initialize(portName, &pdrvPvt->asynInt32);
    if (status != asynSuccess) {
        printf("ledDriver pasynInt32Base->initialize failed\n");
        return;
    }
    pdrvPvt->optr = OPTR;
}
epicsExportRegistrar(ledDriverDeviceSupportRegistrar);
```

Appendix B

Alternate Pinouts

The information in this tutorial applies to several Altera development kits, including:

1. Stratix II DSP development kit(EP2S60), 100 MHz clock.
2. Stratix II NIOS development kit (EP2S60), ”BoardClock100” input is a 50 MHz clock.
3. Cyclone II NIOS development kit (EP2C35), ”BoardClock100” input is a 50 MHz clock, expansion prototype connectors require extensions to clear tall components on development kit.

The pin assignments for each of these kits are presented in the following table.

Signal	1	2	3
BoardClock100	A16	AF15	N2
ColdFireA[0]	R31	E8	E25
ColdFireA[1]	R30	J9	F24
ColdFireA[2]	P32	F8	F23
ColdFireA[3]	P31	A3	J21
ColdFireA[4]	M32	C4	J20
ColdFireA[5]	M31	C3	F25
ColdFireA[6]	N31	C5	F26
ColdFireA[7]	N30	K10	N18
ColdFireA[8]	L32	H9	P18
ColdFireA[9]	L31	G9	G23
ColdFireA[10]	M30	A5	G24
ColdFireA[11]	M29	B5	G25
ColdFireA[12]	N29	D6	G26
ColdFireA[13]	N28	A6	H23
ColdFireA[14]	L30	H10	H24
ColdFireA[15]	L29	K11	J23
ColdFireA[16]	K32	F10	J24
ColdFireA[17]	K31	A7	H25
ColdFireA[18]	K30	C7	H26
ColdFireA[19]	K29	D7	K18
ColdFireA[20]	J32	A8	K19
ColdFireA[21]	J31	G10	K23

ColdFireA[22]	H32	J11	K24
ColdFireA[23]	H31	F11	J25
ColdFireBS2n	U27	A24	AD19
ColdFireBS3n	U23	B24	AC19
ColdFireCLKOUT	T32	AC14	N26
ColdFireCS1n	V29	D18	AC20
ColdFireCS2n	V23	J18	AB20
ColdFireD[16]	AC27	H15	AE24[1]
ColdFireD[17]	AC26	J15	T21
ColdFireD[18]	AD27	C16	V22
ColdFireD[19]	AD26	A17	AF23
ColdFireD[20]	Y23	C17	AE23
ColdFireD[21]	Y22	A18	AC22
ColdFireD[22]	Y25	F17	AB21
ColdFireD[23]	Y24	K16	AD23
ColdFireD[24]	AA27	G17	AD22
ColdFireD[25]	AA26	A19	AC21
ColdFireD[26]	Y27	C18	AD21
ColdFireD[27]	Y26	C19	AF22
ColdFireD[28]	W25	A20	AE22
ColdFireD[29]	W24	J17	V18
ColdFireD[30]	W27	A21	W19
ColdFireD[31]	W26	C20	U17
ColdFireIRQ1n	W28	A22	AF21
ColdFireOEn	U22	K17	AA17
ColdFireRSTIn	M22	H18	V17
ColdFireRSTOn	M23	J14	W17
ColdFireRpWn	U28	B22	AE20
ColdFireTAn	W29	C21	U18
ColdFireTEAn	L25	H17	AA18
ColdFireTSn	V28	C22	AF20
LEDS[0]	B4	AD26	AC10
LEDS[1]	D5	AD25	W11
LEDS[2]	E5	AC25	W12
LEDS[3]	A4	AC24	AE8
LEDS[4]	A5	AB24	AF8
LEDS[5]	D6	AB23	AE7
LEDS[6]	C6	AB26	AF7
LEDS[7]	A6	AB25	AA11
SerialIn	L16	H7	AB15

1. Pin AE24 on the Cyclone II kit is a dual-purpose pin and needs to be configured as Assignments→Device..., Device & Pin Options, Dual-Purpose Pins tab): Change nCEO from "Use as programming pin" to "Use as regular IO".