User Manual

Version: 2

ni logic Pvt. Ltd,
25/B-5, Bandal Complex,
Bhusari Colony, Paud Road, Kothrud,
Pune – 411 038,
Maharashtra, India
Ph: +91-20-5286948
www.ni2designs.com

Table of Contents

Chapter 1
Introduction to Programmable Logic

What is Programmable Logic?

In the world of digital electronic systems, there are three basic kinds of devices: memory, microprocessors, and logic. Memory devices store random information such as the contents of a spreadsheet or database. Microprocessors execute software instructions to perform a wide variety of tasks such as running a word processing program or video game. Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.

Fixed Logic versus Programmable Logic

Logic devices can be classified into two broad categories - fixed and programmable. As the name suggests, the circuits in a fixed logic device are permanent, they perform one function or set of functions - once manufactured, they cannot be changed. On the other hand, programmable logic devices (PLDs) are standard, off-the-shelf parts that offer customers a wide range of logic capacity, features, speed, and voltage characteristics - and these devices can be changed at any time to perform any number of functions.

With fixed logic devices, the time required to go from design, to prototypes, to a final manufacturing run can take from several months to more than a year, depending on the complexity of the device. And, if the device does not work properly, or if the requirements change, a new design must be developed. The up-front work of designing and verifying fixed logic devices involves substantial "non-recurring engineering" costs, or NRE. These NRE costs can run from a few hundred thousand to several million dollars.

With programmable logic devices, designers use inexpensive software tools to quickly develop, simulate, and test their designs. Then, a design can be quickly programmed into a device, and immediately tested in a live circuit. There are no NRE costs and the final design is completed much faster than that of a custom, fixed logic device.

Another key benefit of using PLDs is that during the design phase customers can change the circuitry as often as they want until the design operates to their satisfaction. That's because PLDs are based on re-writable memory technology - to change the design, the device is simply reprogrammed. Once the design is final, customers can go into immediate production by simply programming as many PLDs as they need with the final software design file.

CPLDs and FPGAs

The two major types of programmable logic devices are field programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs). Of the two, FPGAs offer the highest amount of logic density, the most features, and the highest performance. The largest FPGA provides millions of "system gates" (the relative density of logic). These advanced devices also offer features such as built-in hardwired IP cores (such as the IBM Power PC, PCI cores, microcontrollers, peripherals, etc), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies. FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation, telecommunications, and digital signal processing.

CPLDs, by contrast, offer much smaller amounts of logic - up to about 10,000 gates. But CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications. Low power CPLDs are also available and are very inexpensive, making them ideal for cost-sensitive, battery-operated, portable applications such as mobile phones and digital handheld assistants.

The PLD Advantage

Fixed logic devices and PLDs both have their advantages. Fixed logic devices, for example, are often more appropriate for large volume applications because they can be mass-produced more economically. For certain applications where the very highest performance is required, fixed logic devices may also be the best choice.

However, programmable logic devices offer a number of important advantages over fixed logic devices, including:

- PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and the results of design changes can be seen immediately in working parts.
- PLDs do not require long lead times for prototypes or production parts - the PLDs are already on a distributor's shelf and ready for shipment.
- PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets - PLD suppliers incur those costs when they design their programmable devices and are able to amortize those costs over the multi-year lifespan of a given line of PLDs.
- PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory. Customers who use fixed logic devices often end up with excess inventory which must be scrapped, or if demand for their product surges, they may be caught short of parts and face production delays.
- PLDs can be reprogrammed even after a piece of equipment is shipped to a customer.

Conclusion

The value of programmable logic has always been its ability to shorten development cycles for electronic equipment manufacturers and help them get their product to market faster. As PLD suppliers continue to integrate more functions inside their devices, reduce costs, and increase the availability of time-saving IP cores, programmable logic is certain to expand its popularity with digital designers.

Prototyping kits for Programmable Logic devices

With the advent of any programmable device whether it be an EEPROM, PAL, PLA, microcontroller, microchip or a FPGA, the need of programmer was mandatory. The application or usage of programmer was just to download the programming file in the device. But the FPGAs were exceptional case.

Applications like prototyping, product development and learning of VLSI converted the programmers of FPGA into a more complex protoboards, where the designers can program the FPGAs, develop and verify the design and finally can go for production after satisfactory results.

Manufacturing of these trainer protoboards was earlier done by foreign companies and were costlier, but day by day Indian manufacturers developed their own protoboards for the Indian market, and thus VLSI designers, engineers and educational industry of India got easy access to these protoboards.

They are available in a wide range depending on type of device used.

Applications of FPGA/CPLD protoboards are:

1. ASIC prototyping
2. Product development
3. Verification of designs in CPLDs and FPGAs.
4. Helping users to exploit architectural features of CPLDs and FPGAs.
5. Performing a wide range of experiments, by actually downloading the designs into physical devices.
6. Understanding use of HDL's.

FPGAs and CPLDs have seen an exponential rise in their architectures and hence their applications. In mid '90s PLDs were used in packing of digital logic into them, but today devices are much more complex and available with inbuilt hard-wired microprocessors, PCI cores, peripherals, etc. Thus their protoboards/kits also need to be designed accordingly; so as to exploit the complete architecture and feature set of PLDs a designer needs some basic specifications on the protoboards

The most demanding feature set of any PLD Protoboard is as follows:

1. Ability to program the PLD in circuit
2. User inputs and output
3. Displays like 7-segment and LCDs
4. Keyboard
5. Clock circuit
6. Power ON Reset
7. User interface I/Os
8. On board power supply
9. Any bus interface
10. Easy to use and operate
11. etc.

Needs of today's protoboards

VLSI trainer protoboards or kits are extensively used in prototyping of FPGA based product designs, training and for experimental purposes. All of them posses feature set towards programming of FPGAs and just few user input-output facilities. Very few protoboards offer a complete solution for integration of various user modules with ease. With the advancement in the system requirements and specifications, designers are finding difficult in development of FPGA based designs.

The increase in complexity of design has made a significant change in FPGA architecture. With Virtex-II pro device from Xilinx and Stratrix/Excalibur from Altera designers can integrate a complete system in them. With millions of system gate capacity and inbuilt microprocessors they give flexibility to designers to develop smaller and more complex designs.

Technology Interface

As the FPGA designs accounts the integration of dense memories, analog interface, DSP microprocessors and microcontrollers, many vendors are unable to provide a platform where a designer can integrate all kinds of modules with the FPGA.

High speed performance

Integration of other technology modules is not only the issue with complex FPGA designs. The most important part in FPGA designs is to achieve the target speed of operation. Many designs are required to operate more than 60MHz and how many actually guarantee this speed of operation with bad PCB designs, congested boards, bad connectors, and poor power supplies.

Generally the vendors use the berg connectors for the user interface and FPGA attachment, which cannot perform more that 20-30 MHz, the signals get deteriorated and poor single integrity of design is the outcome.

FPGA stacking

Also an FPGA designer needs a freedom to use any FPGA for his application, i.e., he may opt to use FPGAs from different vendors according to the design requirements and project needs. Most of the vendors have failed to give this kind of feature. FPGA stacking feature is most demanding one in the design of an FPGA protoboard, which means a designer can interface more than one FPGA together and that too from different vendors. Even if the foreign vendors have come with feature of FPGA stacking there is limitation on number of FPGAs and generally they use BERG pins for stacking which are unable to perform at high frequencies.

Easy user interface

Designers have always faced problems now and then in attaching external world signals with the protoboards. When it comes to external design module interface the problems increases as the interface is poor and testing becomes a headache.

Easy user interface has always the key point in designing these protoboards, for this manufacturers provide options of 'D' type connectors, BERG connector, or other connectors. These connectors have problems like limitation on number of I/Os, poor signal integrity and hassle of wires.

Up gradation

Looking at the customer needs the products are continuously up graded or redesigned. The system should be capable of up grading the system components and giving the product some added features to compete in the market. Generally protoboards don't have this kind of feature with them. After couple of project completions the designers find their existing protoboards old and difficult to upgrade. Where as to be in market the protoboards should posses the ability to upgrade themselves with the latest FPGAs, CPLDs, increase in number of user I/Os, interface with new technologies, replacement of existing FPGA with higher speed device, etc.

Chapter 2
About USDP

Overview of Universal System Development Protoboard

VLSI trainer protoboards or kits are extensively used in prototyping of FPGA based product designs, training and for experimental purposes. All of them posses features towards programming of FPGAs and just few user input-output facilities. Very few protoboards offer a complete solution for integration of various user modules with ease. With the advancement in the system requirements and specifications, designers are finding difficulty in development of FPGA based designs.

As the FPGA designs accounts the integration of dense memories, analog interface, DSP microprocessors and microcontrollers, many vendors are unable to provide a platform where a designer can integrate all kinds of modules with the FPGA.

With increase in use and demand of FPGAs in the communication, telecom, power electronics, motion control and educational industries the protoboards are in huge demand, thus **ni logic** took initiative to provide an excellent prototyping platform for these industries.

Our R&D arm **ni2 designs** have come out with the complete solution towards the problems of trainer kits in DSP, VLSI and Microprocessors.

ni2 designs have designed prototyping protoboards with the concept of backplane interface of various modules, giving a complete solution for the platform integration of VLSI, DSP, microprocessors and power electronics applications.

With this product **ni logic** is aiming to give the industry a product that can be used in various applications, thus solving the problems of designers for high speed designs and platform integration. With this unique idea product, **ni logic** will stand ahead in market segment of PLD products and will prove with its sales figures in coming years.

Features of USDP

- Easy to use and implement system designs.
- Slot cards for FPGA from Altera, Xilinx and other vendors with package support up to FG256.
- Stacking of multiple FPGAs (can be of different vendors).
- 64 bit general purpose bus interface with FPGA.
- 77 bit bus sharing between FPGAs.
- High performance backplane, good frequency response upto 80MHz designs (**one of the fastest protoboards available in India**).
- On board JTAG circuit for downloading.
- Multiple configuration options with JTAG chaining of devices.
- User selectable configuration modes, using either FLASH PROM / JTAG.
- 32 Digital I/Ps and O/Ps, each can be configured as input or output giving flexibility to designers.
- On board system reset circuit.
- Configuration reset circuit.
- Four seven segment Multiplexed display.
- 4x4 membrane keypad.
- On board crystal oscillator socket (user can select his desired oscillators).
- General-purpose user area for interface of user clock circuit.
- Ability to use Clock management circuits of FPGAs.
- Proper configuration of FPGAS with high-speed clocks through special scheme.
- SMPS.
- Support for different I/O Standards.
- A complete I/O bank for user VREF interface, using DB25 connector.
- Parallel port interface.
- Easily accessible user I/Os.
- 3 on board 120-pin connector for Add-on card interface.
- Stacking of maximum three Add-on card modules possible of different technologies.
- 89c51 Microcontroller card for traditional 8051 applications with ISP support.
- PIC Microcontroller card for industrial based applications.
- Memory Card for data intensive applications.
- High performance ADC/DAC add-on card.
- Power module for motion control and electro-mechanical applications.
- Intensive user manual support with various examples and source codes.
- And many more.

Backplane features

- Supports interface with dual daughter PLD cards
- Supports FPGAs from reputed PLD vendors like Xilinx, Altera, etc.
- Stacking of maximum two PLD daughter cards possible, with on-board chaining circuit.
- 3 PCI connector based application specific add-on cards
- 64 bit bus sharing between application add-on cards and PLD daughter cards
- 77 bit bus sharing between PLD daughter cards
- 32 Digital I/Ps and O/Ps, each can be configured as input or output giving flexibility to designers.
- On board system reset circuit.
- Four seven segment Multiplexed display.
- 4x4 switch matrix keyboard interface header.
- On board crystal oscillator socket (user can select his desired oscillators).
- High performance backplane PCB.

PLD Daughter cards Available

Different PLD modules are available for different devices from vendors like Xilinx, Altera, etc. The user can prototype upto **1 million (10,00,000)** gate count designs.

Here is the list of PLD daughter add-on cards and their feature set;

Features of PLD Modules

Xilinx FPGA Card

- Supports Spartan-II device family of FPGA's.
- On board Regulators for VCCINT and VCCIO generations.
- User selectable configuration modes.
- Facility to program through JTAG, Slave serial and Master serial modes.
- Onboard PROM support.
- Support for different I/O Standards.
- High speed interface with other add-on cards.
- Capability to use special clock management features of Spartan-II.
- A complete I/O bank for user VREF interface, using DB25 connector.
- Parallel port interface directly from add-on card of FPGA.

Xilinx		
PLD Family/Architecture	**Device Available**	**Package**
Spartan -II	XC2S50, XC2S100, XC2S150, XC2S200	PQ208

Altera FPGA Card

- Supports ACEX 1K device family of FPGA's.
- On board Regulators for VCCINT and VCCIO generations.
- User selectable configuration modes.
- Onboard PROM support.
- High speed interface with other add-on cards.
- Parallel port interface directly from add-on card of FPGA

Altera		
PLD Family/Architecture	**Device Available**	**Package**
ACEX 1K	EP1K50, EP1K100	PQ208
Note: Customized daughter cards are also available for other devices		

List of Application Add-on modules

Here is the list of add-on modules that are supported by "Universal System Development Platform". All cards are made on edge connector of 120 pins.

1. Micro-controller Card

- Philips 89C51 RD2 controller.
- RS-232 interface.
- On board serial EEPROM.
- On board serial RTC.
- In system serial programmable (ISP).
- All the I/Os are accessible to FPGA with connector in between.
- Available with standard codes of I2C, timers, data transfer, etc.

2. PIC Micro-controller Card

- Microchip 16F877 PIC microcontroller .
- In system Programmable.
- On board RTC and RS 232 Interface.
- Interrupt port.
- All ports accessible through edge connector.

3. Memory Add-On Card

- Add on module for Memory intensive applications.
- Total 2MB data storage capacity.
- Four 512KBx 2 SRAM.
- Direct high speed interface with FPGA modules.
- 70ns of access time.

4. ADC/DAC Add-On card

- 4-channel ADC.
- 2-channel DAC.
- Sampling rate upto 400 KSPS.
- 8-bit ADC resolution.
- 12 –bit DAC resolution.
- 0-5V or +/- 2.5V input voltage.

5. Power Electronics module

- IGBT based O/P drive, with current rating upto 10Amps.
- Dual high current relays.
- Stepper motor controller circuit.
- 5 optically isolated O/Ps.
- Step down transformer (/100) for line monitoring applications.
- Separately available in isolated board.
- High current capacity connectors.
- Easily interfaced with motors and other circuits.
- Optimum choice for Pulse Width Modulation (PWM), motion control, line monitoring & control applications.

6. General Purpose Add-On Card (*Free* of charge)

- General purpose layouts.
- Helps in prototyping electronic circuits, with FPGA interface.
- Useful in mixed signals designs.

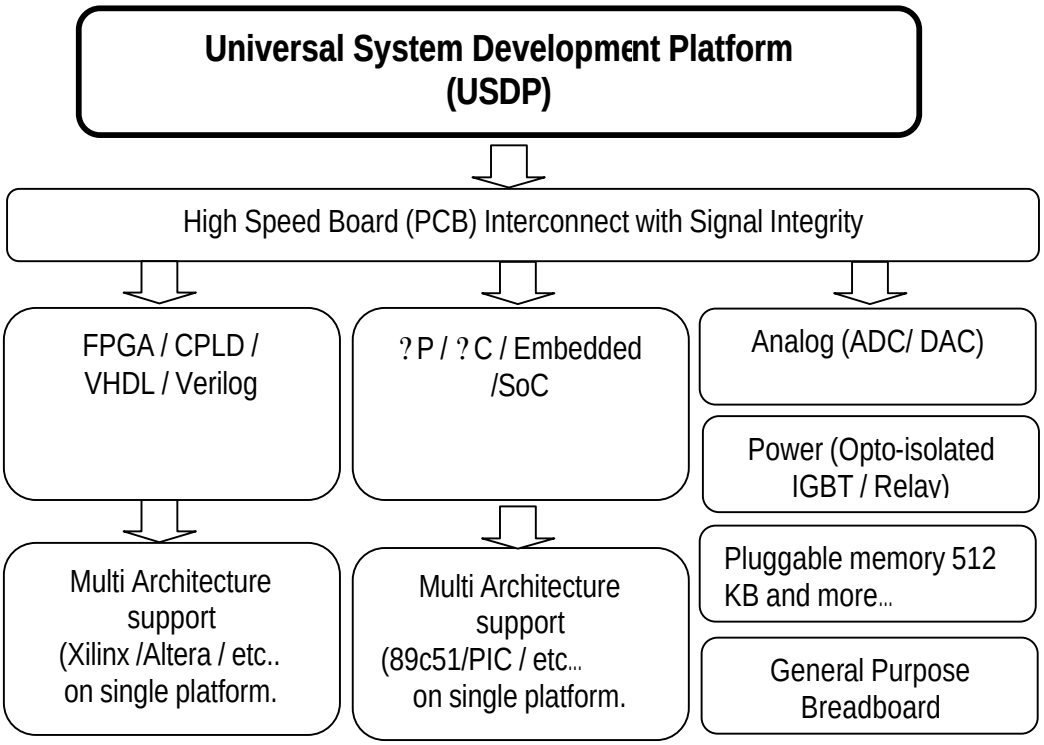
Other Accessories Provided

- JTAG Cable for programming of FPGAs.
- Serial cable for RS-232 communication from 89c51
- PIC programming Cable.
- PIC Serial Cable for RS-232 communication.
- ADC/DAC cables for interfacing external world signals.
- SMPS.
- 4x4 membrane keypad with cable.
- 16x2 character LCD with cable.
- User Manual with instructions, reference designs, examples, sources codes and reference datasheets.
- CD-ROM containing user manual, source codes, reference designs and programming softwares.

Note: The above modules comes in a package, many modules can be optional to user.

ni logic Pvt. Ltd holds the rights to modify the product features and specification without notice.

Chapter 3
Universal System Development Platform (USDP) Overview



Training Features

? Single board / platform for all types of training courses / on Embedded and VLSI.

? Suitable for both Advanced and Beginner Courses / Trainings and workshops.

? Easy customization of board for training needs of different architectures in VLSI / Embedded.

? Easy and Simple to use with very User-friendly interface, making learning the platform operation easy for students /beginners.

? Exhaustive Students manual and Instructors guide available with solved examples, covering basic and advanced topics.

Development Features

? Single Board / Platform for complete system design covering all parts in the system like FPGA / CPLD / ?P / ?C / Embedded / SoC on a single platform.

? Multi architecture support for FPGA /CPLD / ?P / ?C allowing evaluation of design performance on different devices like Xilinx Altera, etc... in VLSI and 89c51,PIC, etc... in embedded.

? High speed and reliable System Interconnection Bus with proper signal integrity and low skew capable of board / system level data transfer upto 80 MHZ reliably.

? Specially mounted High speed gold plated and shielded connectors for signal integrity.

? Specially designed Clock network for low board level skew between components.

? Specially designed High current capacity power supply for High Speed system,

Advantages of Universal System Development Platform over other protoboards

Here are the technological advantages of **USDP** with other protoboards.

- Easy system design and development.
- Good learning and design implementation platform.
- Single platform for multiple technologies.
- Stacking of multiple FPGAs which makes multiple FPGA designs development possible.
- Higher performance.
- Long chain of Add-on modules for all type of applications.
- Capable for future advancement and up gradation with new technologies.
- Large number of user interface.
- General purpose PCB for user circuit interface.

Applications of USDP

- Prototyping of FPGA designs
- Research and development of high-speed FPGA designs.
- Design and development of FPGA based DSP designs and algorithms
- Understanding of various FPGA architectures
- To train designers how to exploit architectural features of FPGAs from different vendors
- Performing a wide range of experiments, by actually downloading the designs into FPGA
- Understanding the basics of HDL's and digital logic interface
- Robotics and Motion control applications.
- Mictrocontroller based applications.
- PC controlled design development.
- Many more... ... ...

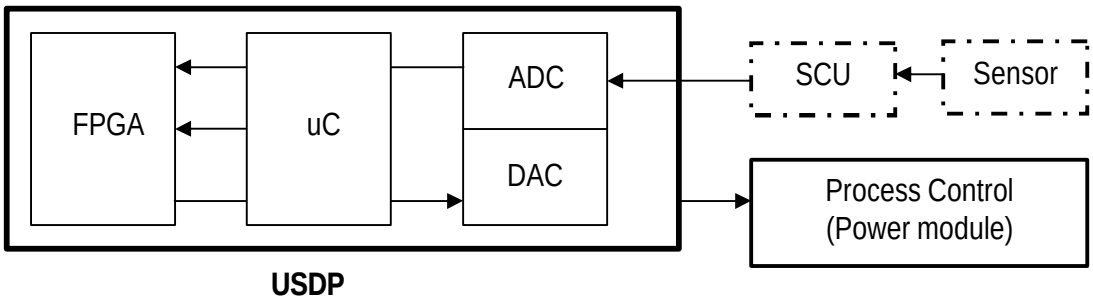
Examples:

Temperature controller

USDP is ideal for temperature control applications. Only a temperature sensor has to be interfaced along with its signal conditioning unit(SCU) to ADC. After the digitization of temperature values, user can manipulate the values and control the process using FPGA or the microcontroller cards.

Modules Can be used

PIC card / 89C51 card, FPGA card, General Purpose PCB card, ADC/DAC Card and power module.



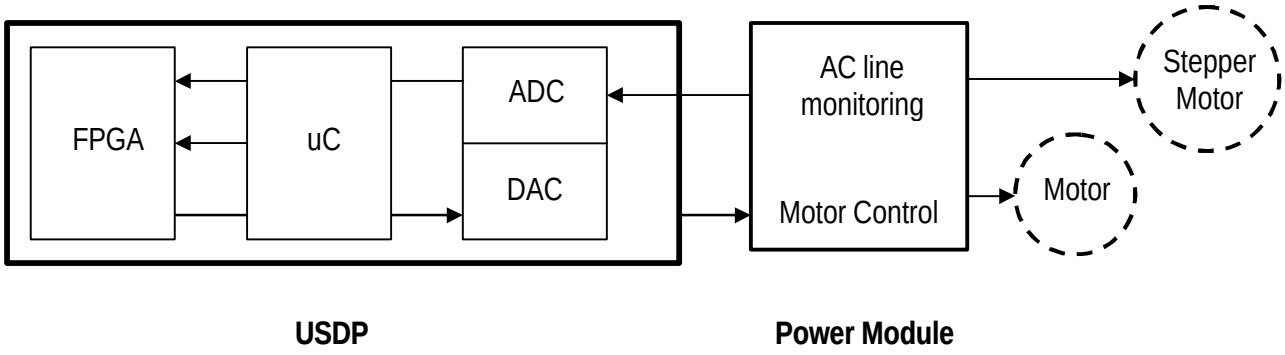
Motion Control

USDP is ideal for robotics applications development. As Stepper motors are widely used and plays vital role in robotics environment for implementing arms, handles, moving/rotating mechanisms.

With the use of external power module USDP can prove good platform for motion control applications. With onboard stepper motor controller circuit, AC line step down transformer, relays, optically isolated outputs, etc., power module makes USDP as a complete system development platform.

Modules used

PIC card / 89C51 card, FPGA card, General Purpose PCB card, Power module.

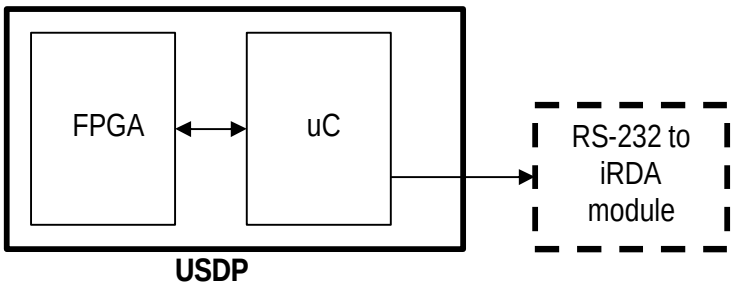


Infrared Communication

USDP is designed to meet a variety of application needs with distinct advantages that enable the embedded system designer to easily add infrared wireless connectivity. IRDA IC's can be easily assembled on General purpose PCB's and data communication can be achieved using the micro controller or FPGA/CPLD.

Modules Can be used

PIC card / 89C51 card, FPGA card, General Purpose PCB card, IRDA IC's.

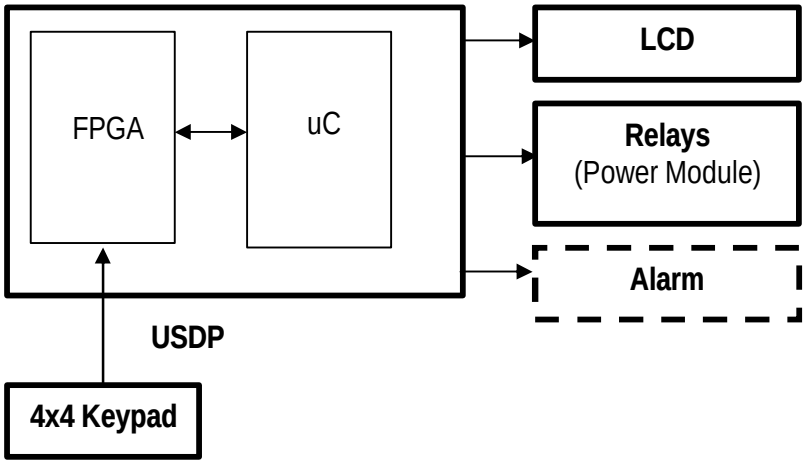


Access control system

Today many applications are developed for security and access controlling. The basic applications of access control can be developed and prototyped on USDP. The basic model consists of keypad interface for password entering, solonoid for door open & close which can be replaced with relays here, user display for welcome notes, menus and messages and protocol for security maintenance.

Modules Can be used

PIC card / 89C51 card, FPGA card, General Purpose PCB card, LCD, Power module and keypad.

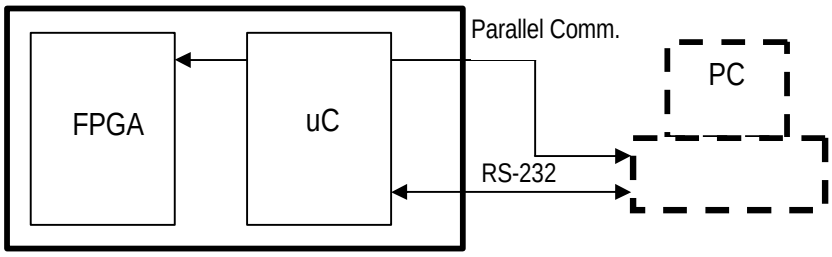


PC based applications

Today most of the applications are having PC communication or control. For communication with PC **USDP** provides both **serial** and **parallel** communication. Parallel port connector is provided on FPGA cards, which can used to communicate with PC from its parallel port. Also every microcontroller is equipped with RS-232 communication port for serial communication with PC. Hence USDP is worth getting product for such PC controlled application development.

Modules Can be used

PIC card / 89C51 card, FPGA card.



8051 microcontroller-based applications

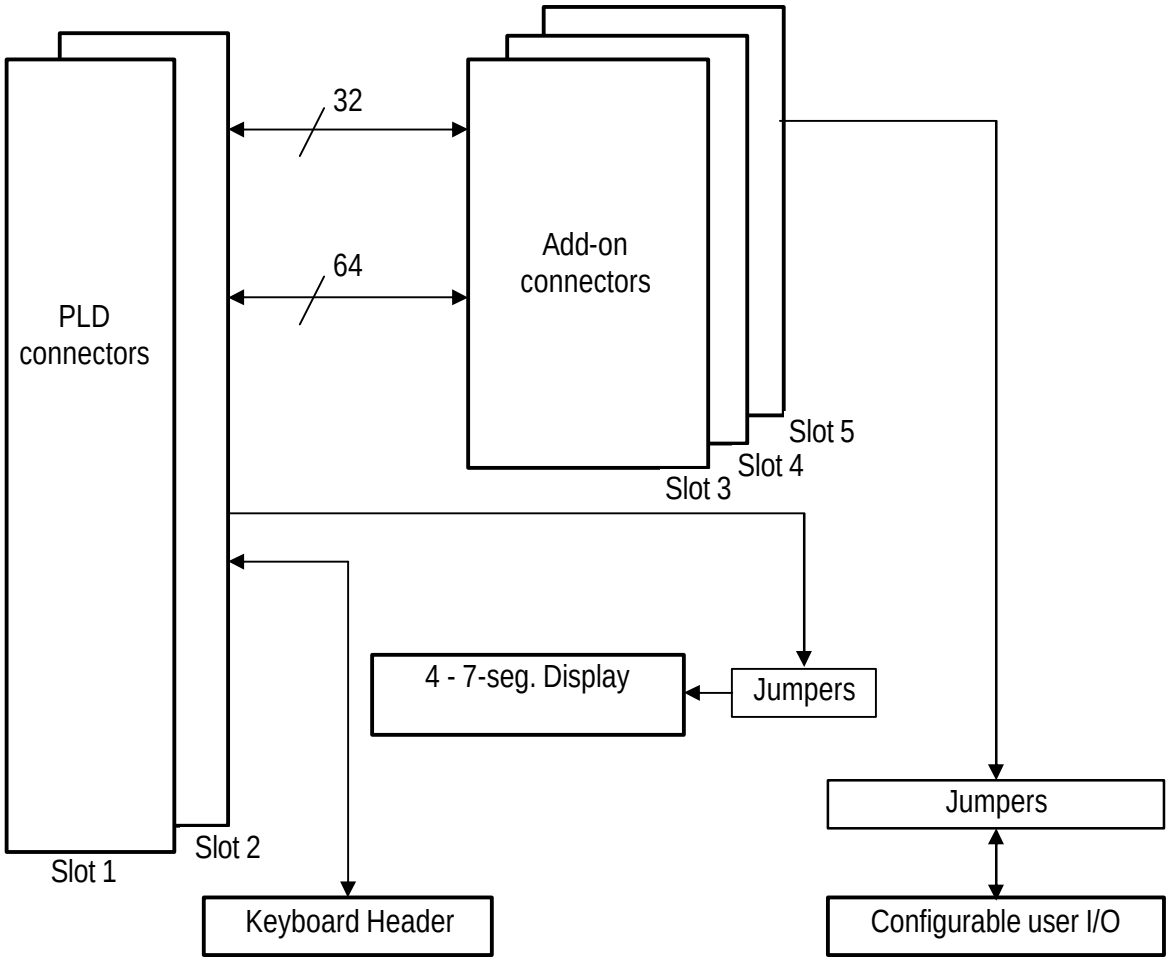
USDP can give excellent support of development and implementation of microcontroller based application. The user gets two top of the line microcontroller development cards based on 89c51RD2 and PIC 16F877. Using these controller cards the designer gets the choice for selecting the controller depending on the application requirement. As 89c51 is based on 8051 architecture the students feels comfortable and gets the exposure for practical design development and also the PIC 16F877 which is RISC based architecture gives them a real learning experience.

Other Digital experiments and Practicals

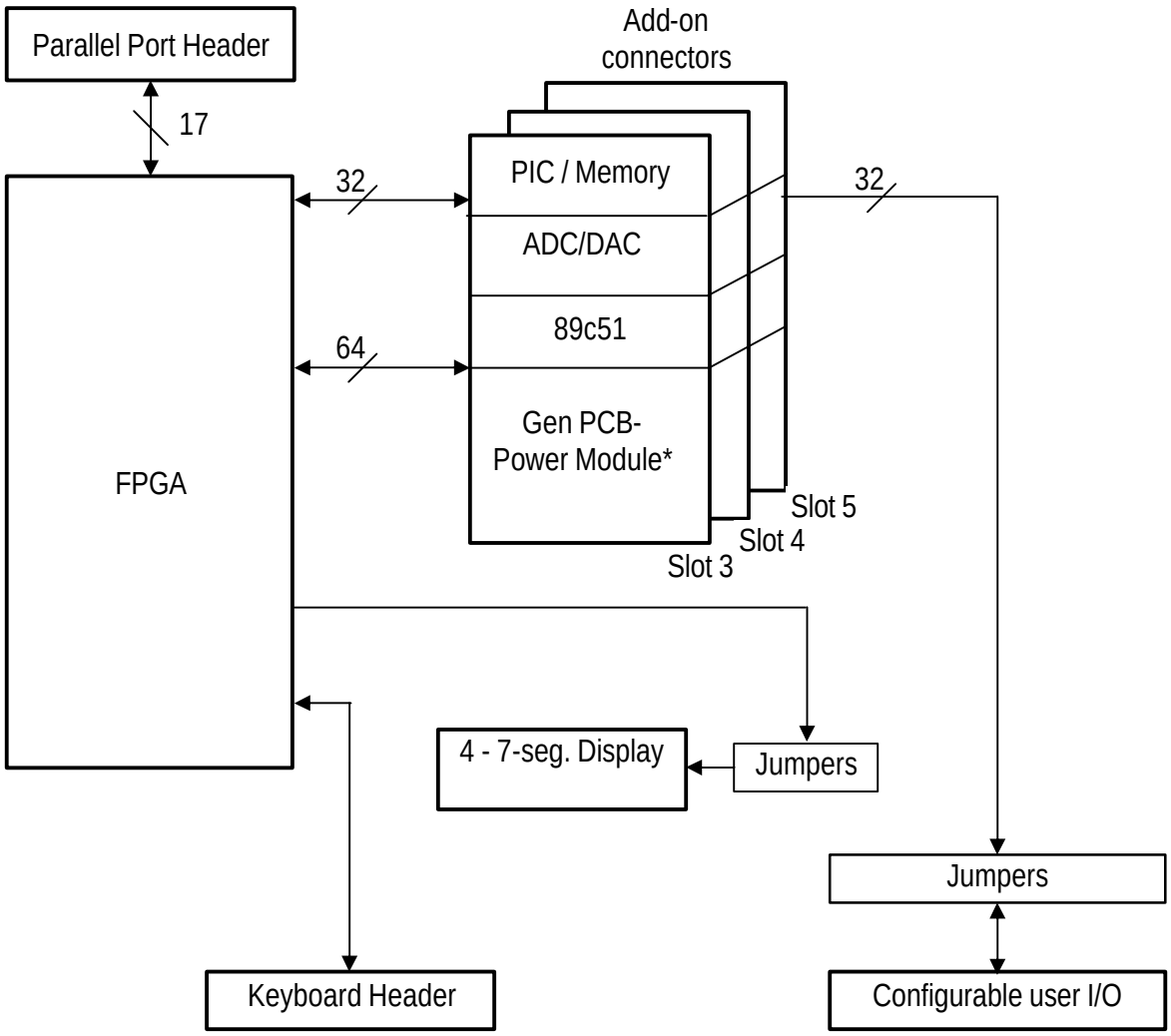
Design of 8/16/24/32 bit counters & shift registers, Adders and subtractors, 4-bit and 8-bit ALUs, Timer designs IC8254 & IC8253, 8255 PPI design, Micro-processor design development using HDL, All digital logic gates and functions.

Chapter 4
System Diagrams

This chapter has the required system diagrams of USDP.



USDP Base board connectivity



Note: * General purpose PCB and power module can be mapped anywhere on the add-on connectors.

FPGA Connectivity

Add-On Connector Pin Out Table

A1	LD0	**A2**	LD1	**B1**	LD2	**B2**	LD3
A3	LD4	**A4**	LD5	**B3**	LD6	**B4**	LD7
A5	LC0	**A6**	LC1	**B5**	LC2	**B6**	LC3
A7	LC4	**A8**	LC5	**B7**	LC6	**B8**	LC7
A9	LB0	**A10**	LB1	**B9**	LB2	**B10**	LB3
A11	LB4	**A12**	LB5	**B11**	LB6	**B12**	LB7
A13	LA0	**A14**	LA1	**B13**	LA2	**B14**	LA3
A15	LA4	**A16**	LA5	**B15**	LA6	**B16**	LA7
A17	IO0	**A18**	IO1	**B17**	IO2	**B18**	IO3
A19	IO4	**A20**	IO5	**B19**	IO6	**B20**	IO7
A21	IO8	**A22**	IO9	**B21**	IO10	**B22**	IO11
A23	IO12	**A24**	IO13	**B23**	IO14	**B24**	IO15
A25	IO16	**A26**	IO17	**B25**	IO18	**B26**	IO19
A27	IO20	**A28**	IO21	**B27**	IO22	**B28**	IO23
A29	IO24	**A30**	IO25	**B29**	IO26	**B30**	IO27
A31	IO28	**A32**	IO29	**B31**	IO30	**B32**	IO31
A33	IO32	**A34**	IO33	**B33**	IO34	**B34**	IO35
A35	IO36	**A36**	IO37	**B35**	IO38	**B36**	IO39
A37	IO40	**A38**	IO41	**B37**	IO42	**B38**	IO43
A39	IO44	**A40**	IO45	**B39**	IO46	**B40**	IO47
A41	IO48	**A42**	IO49	**B41**	IO50	**B42**	IO51
A43	IO52	**A44**	IO53	**B43**	IO54	**B44**	IO55
A45	IO56	**A46**	IO57	**B45**	IO58	**B46**	IO59
A47	IO60	**A48**	IO61	**B47**	IO62	**B48**	IO63
A49	**GND**	**A50**	**GND**	**B49**	**GND**	**B50**	**GND**
A51	**GND**	**A52**	**GND**	**B51**	**GND**	**B52**	**GND**
A53	**+5V**	**A54**	**+5V**	**B53**	**+5V**	**B54**	**+5V**
A55	**+5V**	**A56**	**+5V**	**B55**	**+5V**	**B56**	**+5V**
A57	**-5V**	**A58**	**-5V**	**B57**	**-5V**	**B58**	**-5V**
A59	**-5V**	**A60**	**-5V**	**B59**	**-5V**	**B60**	**-5V**

A1-A14 and **B1-B14** are LEDs
IO0 to **IO63** are general-purpose bus on the connector.
All cards are mapped on above layout, kindly refer the below pin out detail for individual card layout.

Add-On Connector Layout Detail With Adaptor Module Connections

	PIC	Memory	LEDs		PIC	Memory	LEDs
A1	AN0/RA0	A16	LD0	**A2**	AN1/RA1	A17	LD1
A3	AN2/RA2	A18	LD4	**A4**	AN3/RA3	D0	LD5
A5	AN4/RA4	D1	LC0	**A6**	AN4/RA5	D2	LC1
A7	RD0	D3	LC4	**A8**	RD1	D4	LC5
A9	RD2	D5	LB0	**A10**	RD3	D6	LB1
A11	RD4	D7	LB4	**A12**	RD5	RD/	LB5
A13	RD6	WR/	LA0	**A14**	RD7	A19	LA1
A15	MCLR	A20	LA4	**A16**			LA5
	ADC/DAC				**ADC/DAC**		
A17	D0			**A18**	D1		
A19	D4			**A20**	D5		
A21	D8			**A22**	D9		
A23	EN1			**A24**	EN2		
A25	DB0			**A26**	DB1		
A27	DB4			**A28**	DB5		
A29	INT			**A30**	RDY		
	89c51				**89c51**		
A31	AD0			**A32**	AD2		
A33	AD4			**A34**	AD6		
A35	A8			**A36**	A10		
A37	A12			**A38**	A14		
A39	PORT1_1			**A40**	PORT1_3		
A41	PORT1_5			**A42**	PORT1_7		
A43	INT0			**A44**	INT1		
A45	ALE			**A46**	FRST		
A47		IO60		**A48**		IO61	

PIC, Memory, LEDs share the connections

ADC/DAC Card connections

89c51 Card connections

	PIC	Memory	LEDs		PIC	Memory	LEDs
B1	RC0	A0	LD2	**B2**	RC1	A1	LD3
B3	RC2	A2	LD6	**B4**	RC3	A3	LD7
B5	RC4	A4	LC2	**B6**	RC5	A5	LC3
B7	RC6	A6	LC6	**B8**	RC7	A7	LC7
B9	RB0/INT	A8	LB2	**B10**	RB1	A9	LB3
B11	RB2	A10	LB6	**B12**	RD3	A11	LB7
B13	RB4	A12	LA2	**B14**	RB5	A13	LA3
B15	RB6	A14	LA6	**B16**	RB7	A15	LA7
	ADC/DAC				**ADC/DAC**		
B17	D2			**B18**	D3		
B19	D6			**B20**	D7		
B21	D10			**B22**	D11		
B23	A0			**B24**	A1		
B25	DB2			**B26**	DB3		
B27	DB6			**B28**	DB7		
B29	CS			**B30**	RD		
	89c51				**89c51**		
B31	AD1			**B32**	AD3		
B33	AD5			**B34**	AD7		
B35	A9			**B36**	A11		
B37	A13			**B38**	A15		
B39	PORT1_2			**B40**	PORT1_4		
B41	PORT1_6			**B42**	PORT1_8		
B43	T0			**B44**	T1		
B45	RD_51			**B46**	WR_51		
B47		IO62		**B48**		IO63	

PIC, Memory, LEDs share the connections

ADC/DAC Card connections

89c51 Card connections

Chapter 5
Pin Assignment

This chapter will go through the pin assignment of FPGAs with various add-on modules. The three PCI connectors on board share the bus. All modules can be plugged on any one of the connector, as the connectors are sharing the common bus.
Here is the add-on connector pin out with Altera **ACEX-1K FPGA** and Xilinx **Spartan-II FPGA.**

	Acex-1k	Spartan-II		Acex-1k	Spartan-II		Acex-1k	Spartan-II		Acex-1k	Spartan-II
A1	36	34	**A2**	37	35	**B1**	38	36	**B2**	39	37
A3	40	41	**A4**	41	42	**B3**	44	43	**B4**	45	44
A5	46	45	**A6**	47	46	**B5**	53	47	**B6**	54	48
A7	55	49	**A8**	56	57	**B7**	57	58	**B8**	58	59
A9	60	60	**A10**	61	61	**B9**	63	62	**B10**	64	63
A11	65	67	**A12**	67	68	**B11**	68	69	**B12**	69	70
A13	70	71	**A14**	71	73	**B13**	73	74	**B14**	74	75
A15	75	81	**A16**	85	82	**B15**	86	83	**B16**	87	84
A17	92	87	**A18**	90	86	**B17**	89	3	**B18**	88	4
A19	96	94	**A20**	95	90	**B19**	94	89	**B20**	93	88
A21	101	98	**A22**	100	97	**B21**	99	96	**B22**	97	95
A23	111	102	**A24**	104	101	**B23**	103	100	**B24**	102	99
A25	115	111	**A26**	114	110	**B25**	113	109	**B26**	112	108
A27	121	115	**A28**	120	114	**B27**	119	113	**B28**	116	112
A29	127	122	**A30**	126	121	**B29**	125	120	**B30**	122	119
A31	133	127	**A32**	132	126	**B31**	131	125	**B32**	128	123
A33	139	134	**A34**	136	133	**B33**	135	132	**B34**	134	129
A35	143	139	**A36**	142	138	**B35**	141	136	**B36**	140	135
A37	149	146	**A38**	148	142	**B37**	147	141	**B38**	144	140
A39	159	150	**A40**	158	149	**B39**	157	148	**B40**	150	147
A41	163	163	**A42**	162	162	**B41**	161	152	**B42**	160	151
A43	168	167	**A44**	167	166	**B43**	166	165	**B44**	164	164
A45	173	174	**A46**	172	173	**B45**	170	172	**B46**	169	168
A47	177	179	**A48**	176	178	**B47**	175	176	**B48**	174	175

Note: Both the FPGAs share the bus on add-on connector. User has to take care, that **no two pins are defined as output**, as in that case there would be short on the bus and may damage FPGA I/Os.
User can define the following combination of FPGAs shared I/Os;

FPGA1	**FPGA2**	
Input	Input	Allowed
Output	Input	Allowed
Input	Output	Allowed
Output	**Output**	**Not allowed**

ACEX1K FPGA Pin detail

Device: **EP1kxx PQ208**

Clock & Reset	
Clock	79
Reset	180

Note: **Reset** is active **LOW.**

Configurable I/Os							
LA7	36	**LB7**	46	**LC7**	60	**LD7**	70
LA6	37	**LB6**	47	**LC6**	61	**LD6**	71
LA5	38	**LB5**	53	**LC5**	63	**LD5**	73
LA4	39	**LB4**	54	**LC4**	64	**LD4**	74
LA3	40	**LB3**	55	**LC3**	65	**LD3**	75
LA2	41	**LB2**	56	**LC2**	67	**LD2**	85
LA1	44	**LB1**	57	**LC1**	68	**LD1**	86
LA0	45	**LB0**	58	**LC0**	69	**LD0**	87

7 Segments		Display Enable		Keypad Header			
segA	31	**DISP1**	18	**SL0**	8	**RL0**	3
segB	30	**DISP2**	16	**SL1**	13	**RL1**	11
segC	29	**DISP3**	17	**SL2**	9	**RL2**	7
segD	28	**DISP4**	15	**SL3**	14	**RL3**	12
segE	27						
segF	26						
segG	25						
segDP	24						

Parallel Port Connector (DB-25)			
Par1	205	**Par10**	193
Par2	203	**Par11**	192
Par3	202	**Par12**	191
Par4	200	**Par13**	190
Par5	199	**Par14**	189
Par6	198	**Par15**	187
Par7	197	**Par16**	186
Par8	196	**Par17**	179
Par9	195		
Pin 18 – 25 of DB-25 connector are ground			

Note: Both the FPGAs share the above I/Os. User has to take care, that **no two pins are defined as output**, as in that case there would be short on the bus and may damage FPGA I/Os. Also tri-state the unused pins of FPGA by changing settings of your EDA tool while pin locking.

User can define the following combination of FPGAs shared I/Os;

FPGA1	FPGA2	
Input	Input	Allowed
Output	Input	Allowed
Input	Output	Allowed
Output	**Output**	**Not allowed**

Spartan-II FPGA Pin detail

Device: **XC2Sxx PQ208**

Clock and Reset	
Clock (GCK0)	80
GCK2	182
GCK3	185
Reset	5

Note: **Reset** is active **LOW.**

Configurable I/Os							
LA7	34	**LB7**	45	**LC7**	60	**LD7**	71
LA6	35	**LB6**	46	**LC6**	61	**LD6**	73
LA5	36	**LB5**	47	**LC5**	62	**LD5**	74
LA4	37	**LB4**	48	**LC4**	63	**LD4**	75
LA3	41	**LB3**	49	**LC3**	67	**LD3**	81
LA2	42	**LB2**	57	**LC2**	68	**LD2**	82
LA1	43	**LB1**	58	**LC1**	69	**LD1**	83
LA0	44	**LB0**	59	**LC0**	70	**LD0**	84

7 Segments		Display Enable		Keypad Header			
segA	33	**DISP1**	17	**SL0**	8	**RL0**	6
segB	31	**DISP2**	18	**SL1**	15	**RL1**	10
segC	30	**DISP3**	20	**SL2**	9	**RL2**	7
segD	29	**DISP4**	21	**SL3**	16	**RL3**	14
segE	27						
segF	24						
segG	23						
segDP	22						

Parallel Port Connector (DB-25)			
Par1	206	**Par10**	192
Par2	205	**Par11**	191
Par3	204	**Par12**	188
Par4	202	**Par13**	187
Par5	201	**Par14**	203
Par6	199	**Par15**	200
Par7	195	**Par16**	189
Par8	194	**Par17**	181
Par9	193		
Pin 18 – 25 of DB-25 connector are ground			

Note: Both the FPGAs share the above I/Os. User has to take care, that **no two pins are defined as output**, as in that case there would be short on the bus and may damage FPGA I/Os.

User can define the following combination of FPGAs shared I/Os;

FPGA1	FPGA2	
Input	Input	Allowed
Output	Input	Allowed
Input	Output	Allowed
Output	**Output**	**Not allowed**

PIC 16F877 Micro controller Pin details*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A1	AN0/RA0	36	34	**A2**	AN1/RA1	37	35
A3	AN2/RA2	40	41	**A4**	AN3/RA3	41	42
A5	AN4/RA4	46	45	**A6**	AN4/RA5	47	46
A7	RD0	55	49	**A8**	RD1	56	57
A9	RD2	60	60	**A10**	RD3	61	61
A11	RD4	65	67	**A12**	RD5	67	68
A13	RD6	70	71	**A14**	RD7	71	73
A15	MCLR/	75	81				

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B1	RC0	38	36	**B2**	RC1	39	37
B3	RC2	44	43	**B4**	RC3	45	44
B5	RC4	53	47	**B6**	RC5	54	48
B7	RC6	57	58	**B8**	RC7	58	59
B9	RB0/INT	63	62	**B10**	RB1	64	63
B11	RB2	68	69	**B12**	RD3	69	70
B13	RB4	73	74	**B14**	RB5	74	75
B15	RB6	86	83	**B16**	RB7	87	84

MCLR/ is active **LOW** reset.

*Refer the device datasheet for pin description and functioning.

Note: PIC micro controller and SRAM memory module share the same bus, kindly use one module one at a time on the 3 slots.

SRAM Module Pin details*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A1	A16	36	34	**A2**	A17	37	35
A3	A18	40	41	**A4**	D0	41	42
A5	D1	46	45	**A6**	D2	47	46
A7	D3	55	49	**A8**	D4	56	57
A9	D5	60	60	**A10**	D6	61	61
A11	D7	65	67	**A12**	RD/	67	68
A13	WR/	70	71	**A14**	A19	71	73
A15	A20	75	81				

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B1	A0	38	36	**B2**	A1	39	37
B3	A2	44	43	**B4**	A3	45	44
B5	A4	53	47	**B6**	A5	54	48
B7	A6	57	58	**B8**	A7	58	59
B9	A8	63	62	**B10**	A9	64	63
B11	A10	68	69	**B12**	A11	69	70
B13	A12	73	74	**B14**	A13	74	75
B15	A14	86	83	**B16**	A15	87	84

*Refer the device datasheet for pin description and functioning.

Note: PIC micro controller and SRAM memory module share the same bus, kindly use one module one at a time on the 3 slots.

ADC/DAC Module Pin Detail*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A17	D0	92	87	**A18**	D1	90	86
A19	D4	96	94	**A20**	D5	95	90
A21	D8	101	98	**A22**	D9	100	97
A23	EN1	111	102	**A24**	EN2	104	101
A25	DB0	115	111	**A26**	DB1	114	110
A27	DB4	121	115	**A28**	DB5	120	114
A29	INT	127	122	**A30**	RDY	126	121

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B17	D2	89	3	**B18**	D3	88	4
B19	D6	94	89	**B20**	D7	93	88
B21	D10	99	96	**B22**	D11	97	95
B23	A0	103	100	**B24**	A1	102	99
B25	DB2	113	109	**B26**	DB3	112	108
B27	DB6	119	113	**B28**	DB7	116	112
B29	CS	125	120	**B30**	RD	122	119

ADC header Detail

DAC O/P (J1)

Pin1	DAC Channel 1
Pin 2	DAC Channel 2
Pin 3	Ground

ADC I/P (J2)

Pin1	ADC Channel 1
Pin 2	ADC Channel 2
Pin 3	ADC Channel 3
Pin 4	ADC Channel 4
Pin 5	Ground

*Refer the device datasheet for pin description and functioning.

89c51 Module Pin Detail*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A31	AD0	133	127	**A32**	AD2	132	126
A33	AD4	139	134	**A34**	AD6	136	133
A35	A8	143	139	**A36**	A10	142	138
A37	A12	149	146	**A38**	A14	148	142
A39	PORT1_1	159	150	**A40**	PORT1_3	158	149
A41	PORT1_5	163	163	**A42**	PORT1_7	162	162
A43	INT0	168	167	**A44**	INT1	167	166
A45	ALE	173	174	**A46**	FRST	172	173

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B31	AD1	131	125	**B32**	AD3	128	123
B33	AD5	135	132	**B34**	AD7	134	129
B35	A9	141	136	**B36**	A11	140	135
B37	A13	147	141	**B38**	A15	144	140
B39	PORT1_2	157	148	**B40**	PORT1_4	150	147
B41	PORT1_6	161	152	**B42**	PORT1_8	160	151
B43	T0	166	165	**B44**	T1	164	164
B45	RD_51	170	172	**B46**	WR_51	169	168

FRST is active **HIGH** reset.

*Refer the device datasheet for pin description and functioning.

Power Module Pin Detail

Power module is independent of USDP baseboard, and can me mapped to any of the I/Os of the adaptors available.
Here by are the header details of the power module.

JP1 (AC I/P)	
Pin No.	**Signal**
1	Line
2	Neutral

JP2 (DC O/P)	
Pin No.	**Signal**
1	VDC
2	AGND (Analog ground)

JP4 (AC Line, transformer)	
Pin No.	**Signal**
1	Primary1
2	Primary2

JP5 (Step O/P / 100)	
Pin No.	**Signal**
1	Secondary1
2	Common
3	Secondary2

JP8 (Stepper Motor)	
Pin No.	**Signal**
1	W1
2	W2
3	W3
4	W4
5	DGND (digital ground)

JP6 (Optically Isolated O/Ps)	
Pin No.	**Signal**
1	OP1
2	OP2
3	OP3
4	OP4
5	OP5
6	IGND (Isolated ground)
7	DGND (Digital ground)

JP7 (Relay1)		JP9 (Relay2)
Pin No.	**Signal**	**Signal**
1	Common	Common
2	NO	NC
3	NO	NC

J2 (IGBT)		
Pin no.	**Signal**	**Description**
1	G	Gate
2	C	Collector
3	E	Emitter

JP10 (FPGA Interface)				
Pin No.	**Signal**	**Acex 1K**	**Spartan-II**	**Description**
1	IP1	205	206	Isolated I/P 1
2	IP2	203	205	Isolated I/P 2
3	IP3	202	204	Isolated I/P 3
4	IP4	200	202	Isolated I/P 4
5	IP5	199	201	Isolated I/P 5
6	RELAY1	198	199	Relay control 1
7	RELAY2	197	195	Relay control 2
8	W2	196	194	Winding 2 control
9	W1	195	193	Winding 1 control
10	W4	193	192	Winding 4 control
11	W3	192	191	Winding 3 control
12-16	DGND (Digital ground)			

Note: The above pin assignment is for the power electronics module cable provided along with USDP. Kindly use the same cable to interface the power electronics module.

Chapter 6
Jumpers and Headers
Pin description and Settings

In this chapter we will have a look on the jumper settings of all the USDP modules.

USDP Baseboard

Clock (JP1)

Pin 1	Clock
Pin 2	GCK0
Pin 3	Ground

To use the oscillator clock short pin 1 and 2. It is recommended to ground the GCK0 during programming.

JPDIS

These four jumpers are used to disable the 7-segment displays. Just remove the jumpers to isolate to displays from FPGAs.

JPA, JPB, JPC & JPD

These jumpers are used to select the I/O mode of the LEDs.
Total 32 configurable I/Os are provided on the USDP baseboard. To configure them as I/P user has to insert the jumper in between, and remove the jumper to make them as O/P.
The switch configured as input to FPGA should not be connected to pin which is configure as output in FPGA, else there would be a hardware short, and may cause damage to the device.
User has to take this care during VHDL coding and pin assignment.

Power Headers (J8 & J9)

There are power supply headers; users can connect the SMPS to any one of these headers. Two headers are provided just for the sake of convenience for connecting the SMPS from any of the directions.

SLOT SEL (JP3)

This is slot selector header. User can use the provided selector card for selecting the slot.

Slot 1	Slot 2
Short 1-2	Short 2-3

PLD SEL (JP2)

This is PLD vendor selector header. User can use the provided selector card for selecting the PLD vendors.

Xilinx	Altera
Short 1-2	Short 2-3

J7 (Keypad)

Pin No.	Signal	Pin No.	Signal
1	VCC	2	GND
3	SL3	4	SL2
5	SL1	6	SL0
7	RL3	8	RL2
9	RL1	10	RL0

Programming Port (J6)

User has to connect provided programming cable to this port and PC's parallel for programming of FPGAs.

Xilinx FPGA module

Programming Mode selection Switch (U7)

Modes	M0	M1	M2
JTAG	1	0	1
Slave Serial	1	1	1
Master Serial	0	0	0

Mode selection jumpers (JP4, JP5, JP6, JP7)

Serial	**JTAG**
Short 1-2	Short 2-3

PROM BYPASS (J3 & J4)

To Chain PROM	**To Bypass PROM**
Short 1 –2	Short 2-3

Shorting 1-2 will bring the PROM in chain with FPGA. In this case user can configure the PROM and use for programming file storage.
Shorting 2-3 will remove the PROM from chain, and only FPGA would be connected to programming port.

Global Clock Buffers

Pin 1	GCK2	GCK3
Pin 2	GND	GND

Note: These buffers are extra from GCK0.

Altera FPGA Module

Programming Mode selection jumpers (J3, J8)

	MSEL1	MSEL0
Passive Serial (PS)	0	0
JTAG	0	1

For logic low short 2-3 and for logic short 1-2 for logic High

Mode selection jumpers (J4, J5, J6, J7)

Serial	**JTAG**
Short 2-3	Short 1-2

PROM BYPASS (J1 & J2)

To Chain PROM	**To Bypass PROM**
Short 1 –2	Short 2-3

Shorting 1-2 will bring the PROM in chain with FPGA. In this case user can configure the PROM and use for programming file storage.
Shorting 2-3 will remove the PROM from chain, and only FPGA would be connected to programming port.

89c51 Module

Square Wave o/p (JP3)

Pin1	1Hz clock from RTC.
Pin 2	Ground

Interrupt/Timer (JP4)

Pin1	Interrupt 0 (INT0) I/P
Pin 2	Interrupt 1 (INT1) I/P
Pin 3	Timer 1 O/P
Pin 4	Timer 0 O/P

JP1 & JP2

JP1	Port1_1	SDA
JP2	Port1_2	SCLK

JP1 & JP2 are selection jumper for serial clock and data from RTC and EEPROM. Both these signals are shared with the base bus of USDP. So if you are these two ports of 89c51 with the base bus then remove the jumpers from the JP1 & JP2 to isolate them from the base bus.

PIC Microcontroller Module

Programming jumper

JP9, JP10 & JP11 are programming selection jumpers. User has to short 1-2 during programming and after programming the PIC controller, short 2-3.

JP1

Pin1	1Hz clock from RTC.
Pin 2	Ground

JP2

Pin1	Ra4
Pin 2	Rb0/int
Pin 3	Ground

JP3

Pin1	PCRX
Pin 2	PCTX
Pin 3	Ground

RX2 (JP5)

Port RC7 is used for RX channel of RS-232, also this port pin is shard with base bus and by removing this jumper you can isolate the RS-232 from the base bus.

TX (JP6)

Port RC6 is used for TX channel of RS-232, also this port pin is shard with base bus and by removing this jumper you can isolate the RS-232 from the base bus.

SDA (JP7)

Port RC4 is used for serial data for RTC, also this port pin is shard with base bus and by removing this jumper you can isolate the RTC from the base bus.

SCLK (JP8)

Port RC3 is used for clock for RTC, also this port pin is shard with base bus and by removing this jumper you can isolate the RTC from the base bus.

Analog I/Ps (Header for inbuilt ADC I/Ps for PIC controller)

Pin1	AN0
Pin 2	AN1
Pin 3	AN2
Pin 4	AN3
Pin 5	AN4
Pin 6	AN5
Pin 7	AN6
Pin 8	AN7
Pin 9	Ground

JP9, JP10 & JP11

During programming the PIC controller, **short pins 2-3**, and after programming the controller, **short pins 1-2** for using with FPGA.

ADC/DAC Module

DAC O/P (J1)

Pin1	DAC Channel 1
Pin 2	DAC Channel 2
Pin 3	Ground

ADC I/P (J2)

Pin1	ADC Channel 1
Pin 2	ADC Channel 2
Pin 3	ADC Channel 3
Pin 4	ADC Channel 4
Pin 5	Ground

JP2
This jumper is for setting the reference voltage. This module has onboard reference voltage generation, to use that short 2-3.

Power Electronics Module

As there is no settings in this module, for pin description and header information, please refer pin assignment chapter

General purpose PCB

All the I/Os of base bus are brought on this PCB. User can build their custom circuit on the board and interface with any of the module pins with the help of headers provided on the PCB. The I/O numbering has been indicated nearby the headers, users can refer the pin assignment chapter for further information about the bus description and numbering.
Also this PCB contains headers for +5V, -5V and ground signals onboard.

JP6 (LCD Header)

Pin No.	Signal	Pin No.	Signal
1	GND	2	+5V
3	NC	4	RS
5	NC	6	EN
7	D0	8	D1
9	D2	10	D3
11	D4	12	D5
13	D6	14	D7
15	NC	16	NC

JP1 (LCD I/Ps)

Pin No.	Signal	Pin No.	Signal
1	D0	2	RS
3	D2	4	EN
5	D4	6	D1
7	D6	8	D3
9	D7	10	D5

Note: The LCD module pins are brought on a separate header (JP1), users can interface LCD module to any of the modules I/Os with the provided cable.

Chapter 7
Precautions

Please follow the guidelines below while using USDP and take the mentioned precautions.

General

- Verify the power ON LED status after applying power to the trainer.
- Connect the 25 pin D connector of the cable to the trainer only after confirming the above
- Insert the modules vertically in the slots; slowly and with care, as inserting the modules in angular form will cause damage to connector pins.
- Do not press the modules after inserting them on board, else this can cause damage to connector pins.
- During downloading make sure that the jumper selections are proper
- Before implementation, it is necessary to lock the pins in user constraint file (UCF) as per the mode selected.
- For downloading the bit stream, the downloading circuit requires a stable supply; hence it is recommended to use the power supply supplied with the trainer only.

FPGA modules

- Insert the module vertically in the slots.
- Do not touch the FPGA while handling the module, as it may get damage due to static charge on your body.
- Do program the FPGA as per the Configuration chapter.
- While stacking both FPGA modules, do not make the same pin location/bus as outputs.

Memory module

- As the memory module shares the bus with PIC controller, it is recommended **not to use** PIC controller while using memory module.

Power Electronics module

- If working on the high voltages, then it is recommended to use gloves while handling module.
- Keep the hands off the terminal blocks (AC & DC) as you may get shock from it.
- Do not exceed the ratings of devices, else it may cause damage to them.

ADC/DAC module

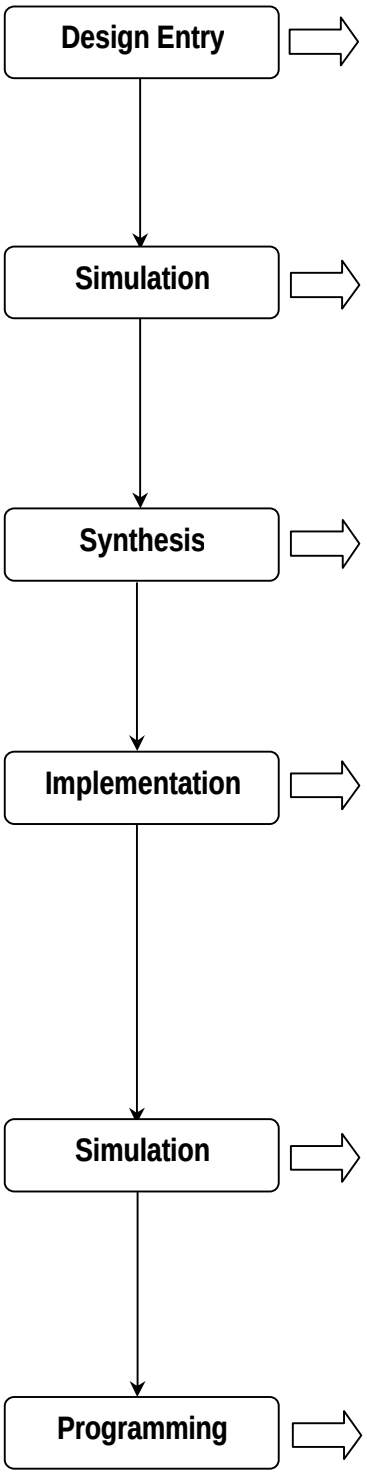
- Apply the analog signals in the range of specifications.
- Step down the high voltage/current signals.

Chapter 8
Using EDA tools

USDP can works on various technologies altogether. The currently supported technologies are VLSI, micro-controller and DSP.
All these technologies have different design flows and software tools for development purposes. For VLSI technology, USDP works on Altera and Xilinx devices right now, and RISC & 89c51 controllers.
To work on the following provided devices, this chapter will cover the design flow and methodology of EDA tools required for building applications on USDP.
We will cover the design flow following technologies;

- PLD design flow
- Micro-controller design flow

PLD Design Flow



HDL (Verilog, VHDL)
Hardware **D**escription **L**anguage.
Very **H**igh speed Integrated circuit **HDL.**
Verilog having 'C' based constructs.

Functional Verification
Functional simulation of the design.
No timings are considered.

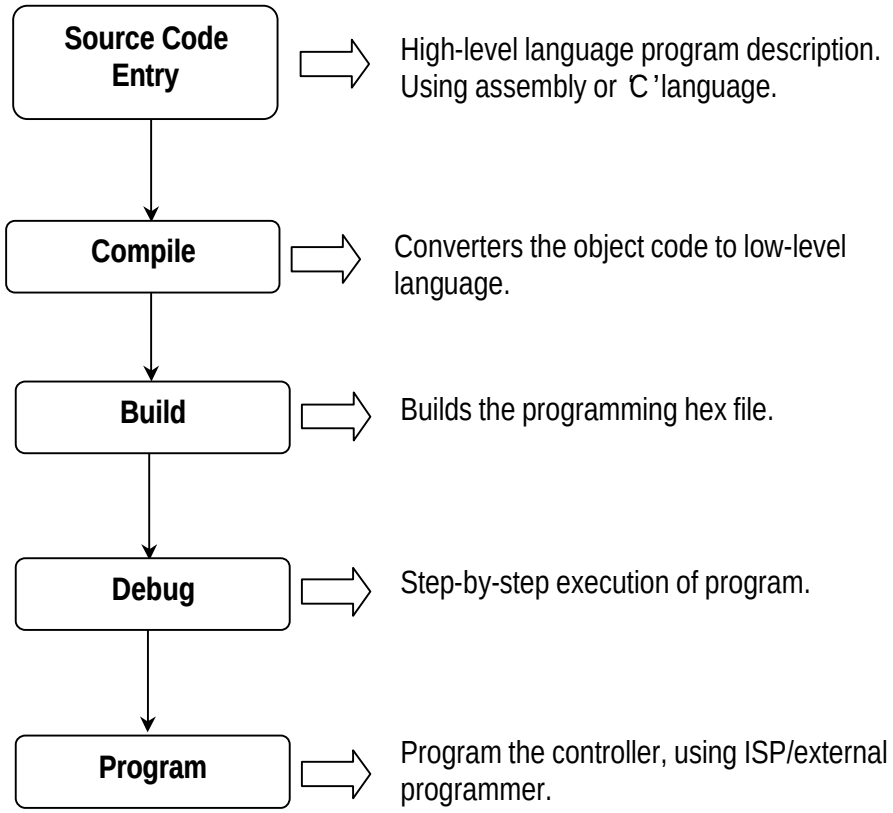
Process for converting design specifications into gate level netlist.
Needs synthesis library containing target technology information.

Combined name for processes like for translation, floorplanning, mapping, place & route, bit file generation.
For locking input and output signal to the particular pins of the device user must write UCF (User Constraint File) before implementation.
Output of implementation is .BIT for FPGA and .JED file for CPLD, which can be directly program into the target device.

Dynamic timing analysis
Functional simulation of the design, running it on the desired operating frequency.
Timings are considered.

This is the process by which user can physically download the design programming files from PC to target device using programming cable.
To program CPLD, select Boundary scan (JTAG) mode.
To program FPGA, select Boundary scan, slave serial mode or Master serial Mode.

Microcontroller Design Flow



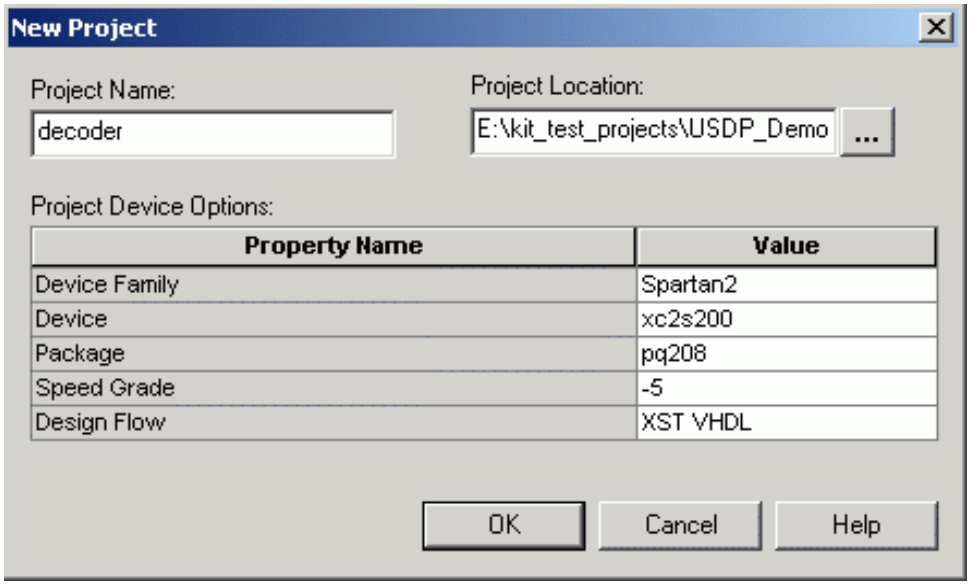
Using Xilinx ISE Series Software

In this section we will see how a project can be created in Xilinx and Altera EDA tools, and how we can proceed to use USDP to perform our experiments using FPGAs
We take the example of 3:8 decoder with enable inputs and implement on both vendor devices.

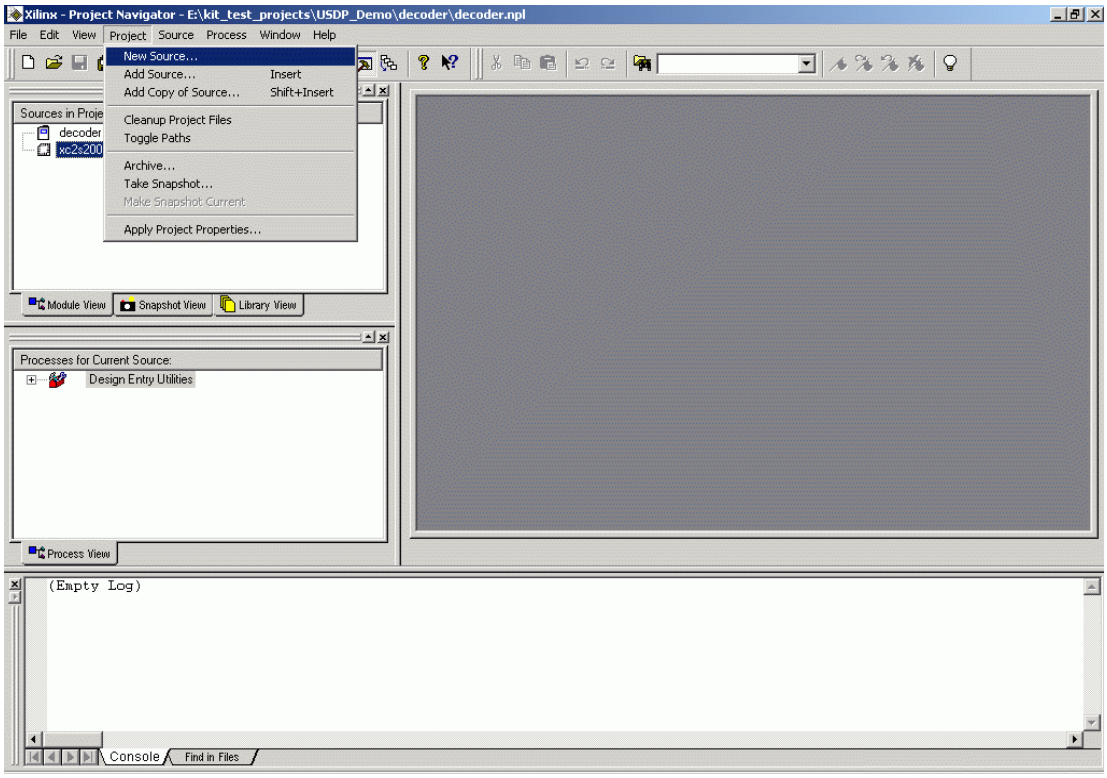
Design flow for Xilinx ISE series softwares

Step 1: Open ISE webpack software.

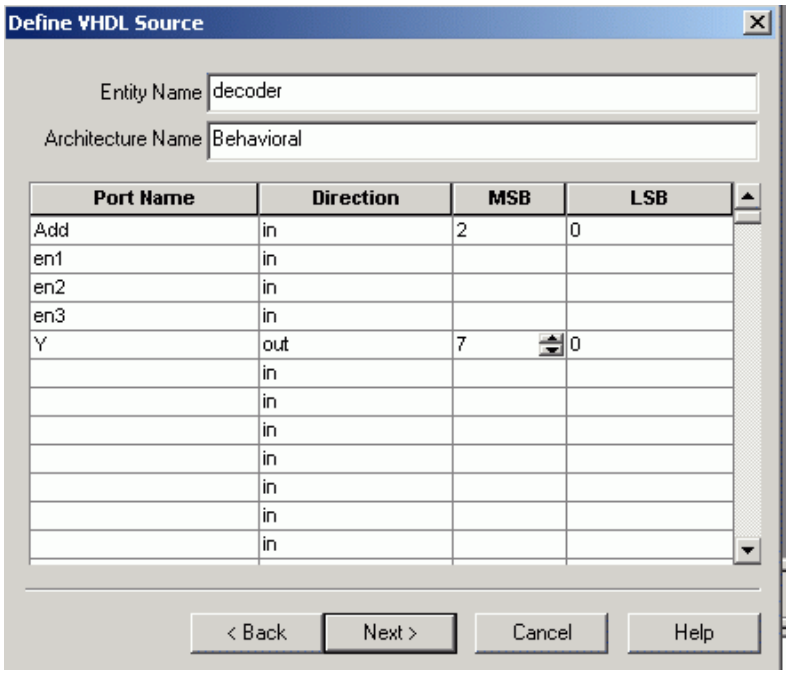
Step 2: Create new project



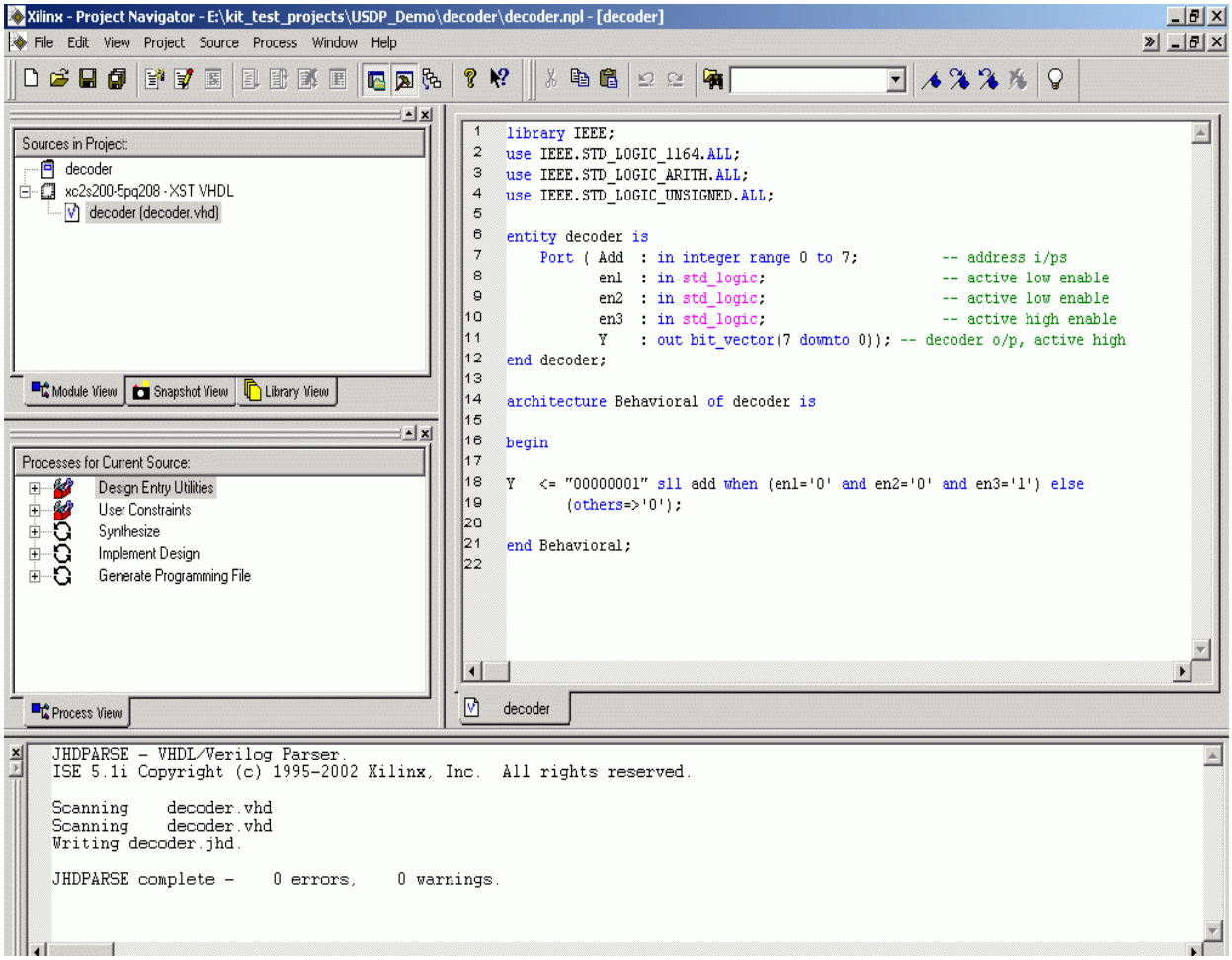
Step 3: Go to project menu and select new source



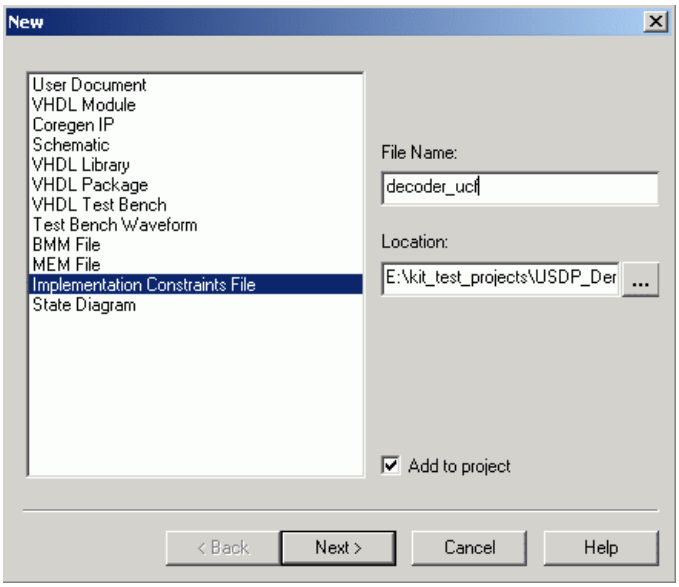
Step 4: Select VHDL source file, name it **decoder,** click next, and enter entity I/Os as **Add, en1, en2, en3 & Y.**



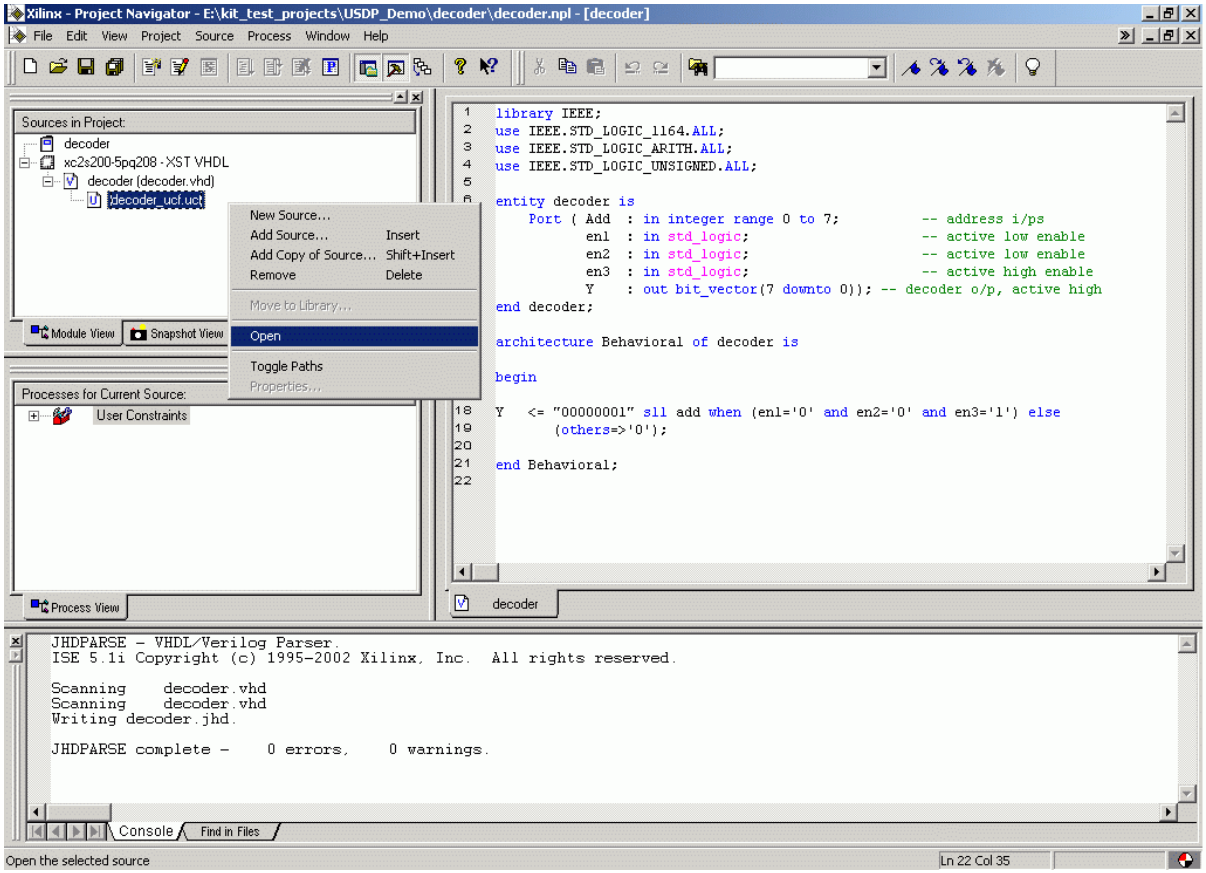
Step 5: Write VHDL code for **decoder.**



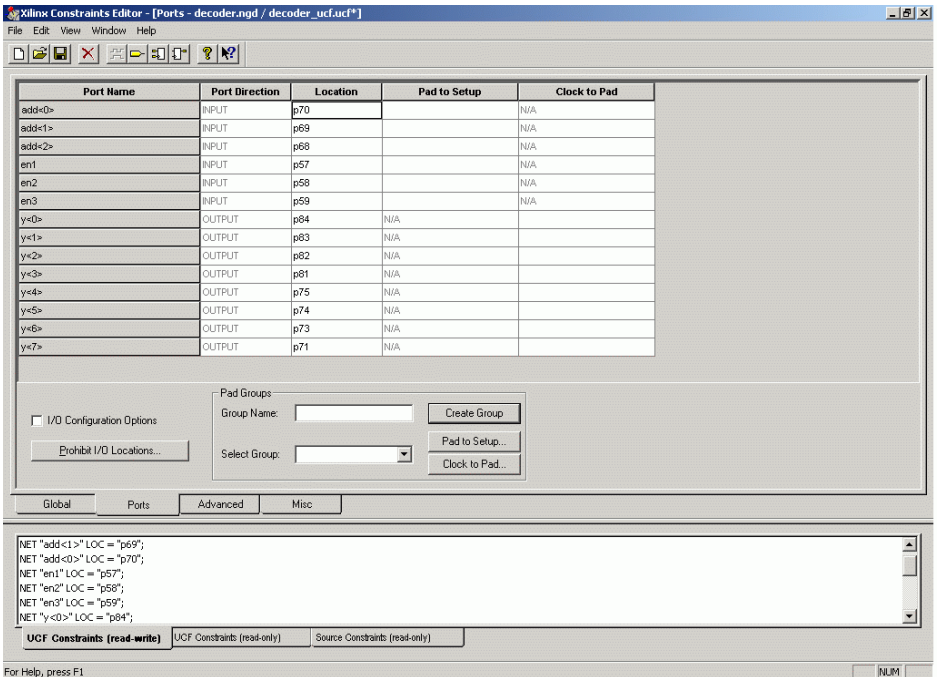
Step 6: Create new source file for implementation constraint file. Name it **decoder_UCF**, and associate with the corresponding design file.



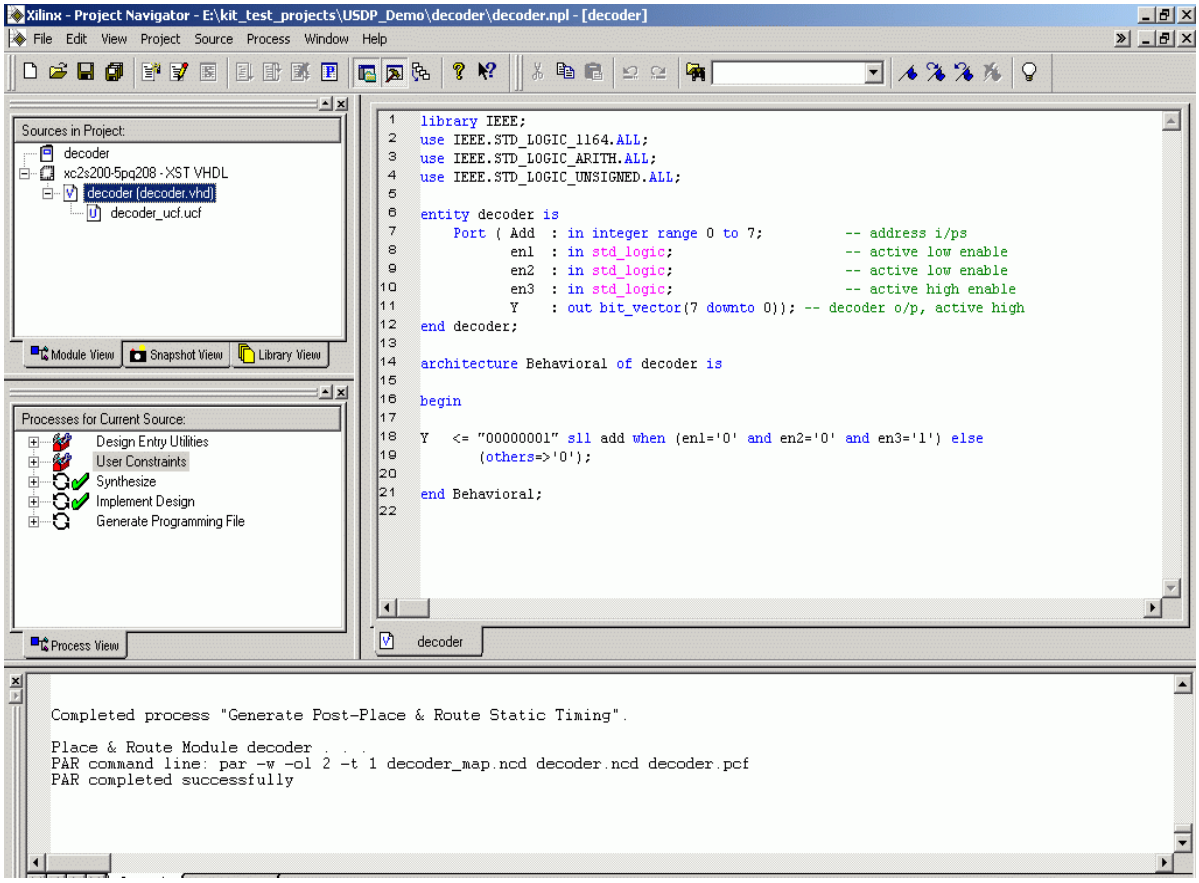
Step 7: To assign the pin location of the design entity, open the UCF file, which in turn will open the constraint editor where we have to lock the I/Os of design to a particular pin number.



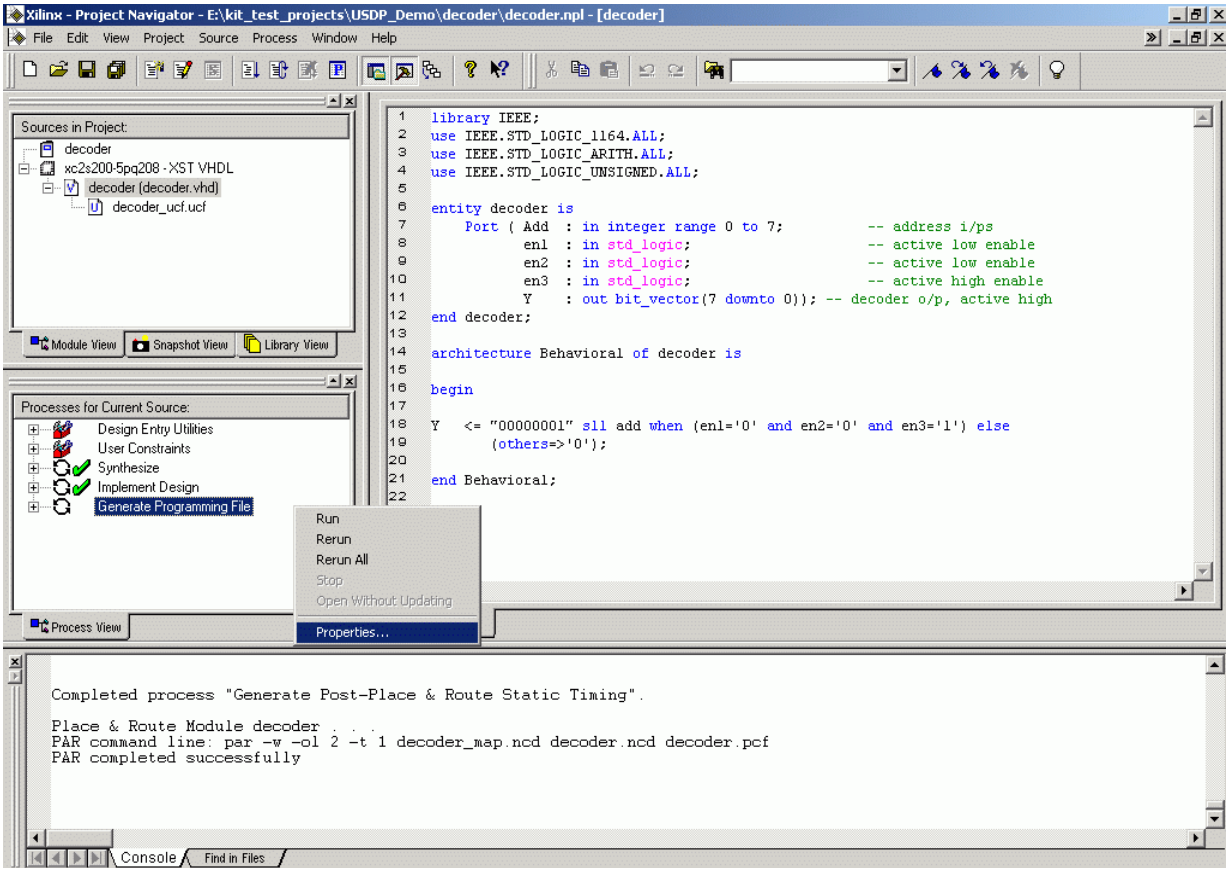
Step 8: Once the constraint editor is opened, goto ports tab; and assign the pins by referring the Pin assignment chapter.
Eg: **net add<0> loc =p84;**



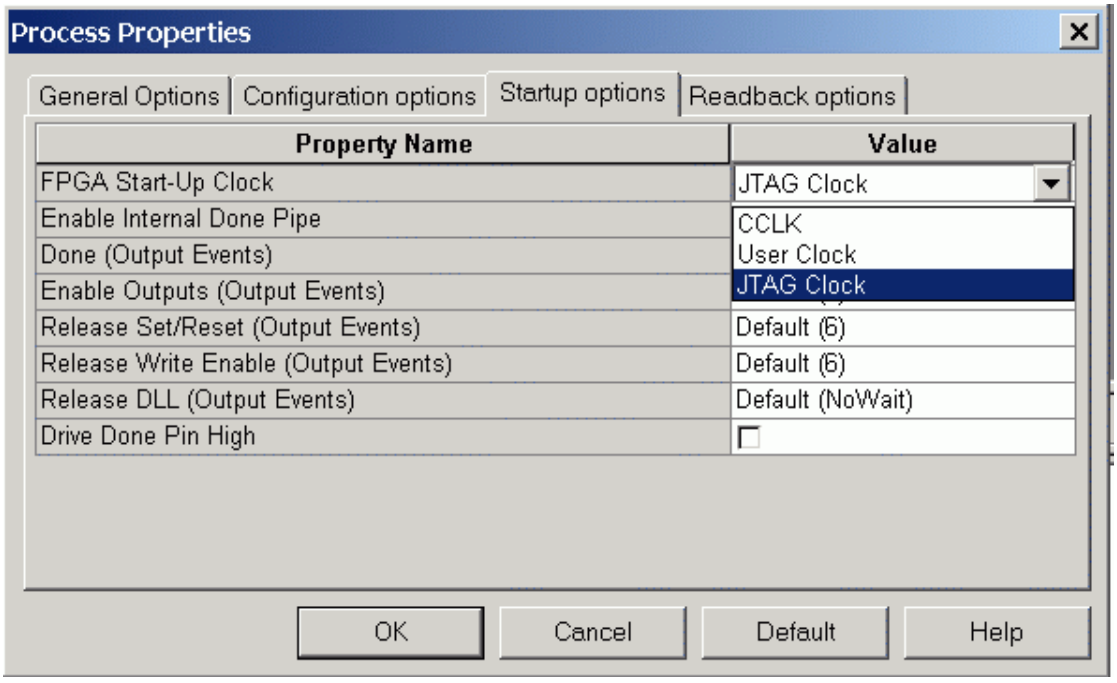
Step 9: Save the UCF file and come back to project navigator. Now selecting the **decoder** design file, run the **synthesis** process, there after **Implementation,** which in turn will place and route the design on the target FPGA.



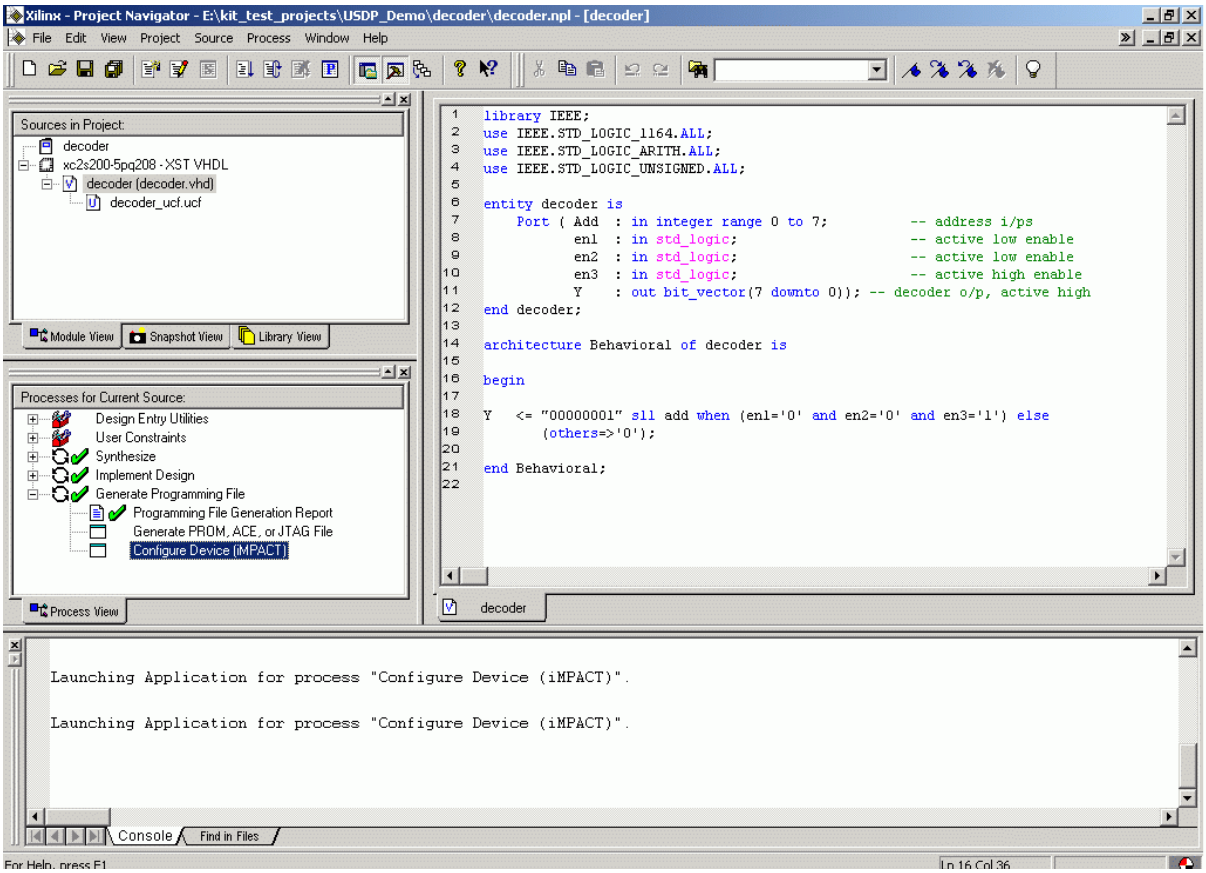
Step 10: Now we need to generate the programming file to configure the FPGA. The Xilinx FPGA supports, JTAG, Slave-serial, and Master-serial (through PROM) for configuration. User has to set proper mode during the programming file generation, else FPGA won t be configured.
For this right click on the **Generate Programming File** option and click properties on the opened menu.



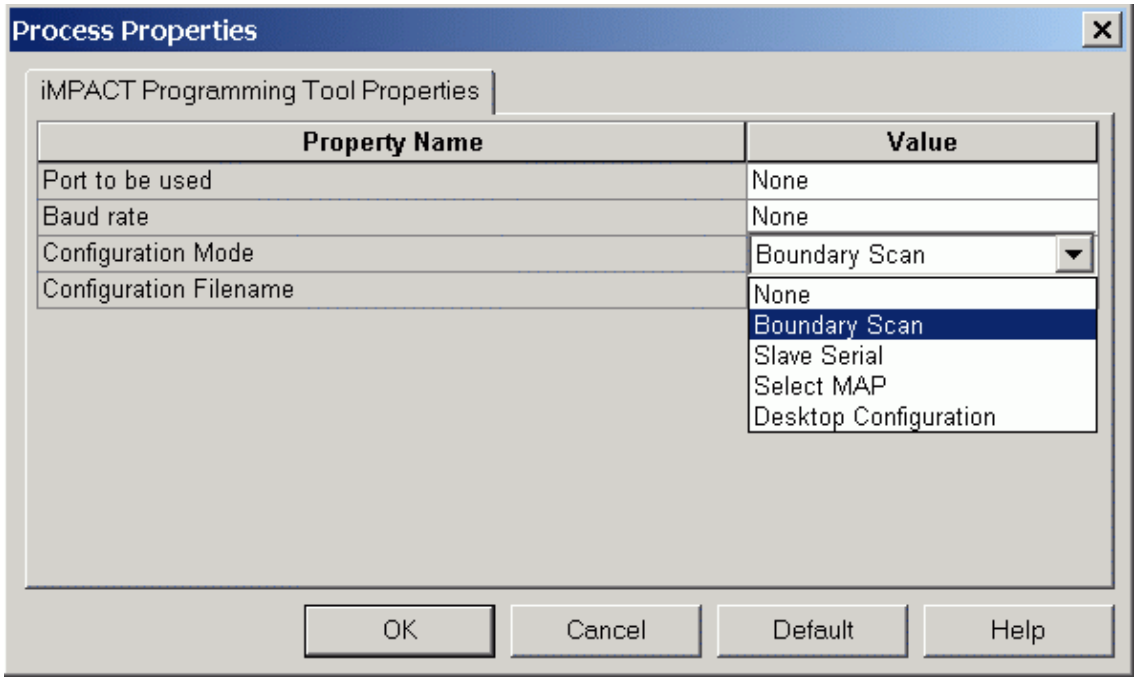
Step 11: In the opened window, go to **startup-options**, and select **JTAG clock** in **FPGA Start-Up Clock** option. This in turn will generate the programming file for **JTAG** mode programming.
To program the FPGA in **Slave Serial** mode, user has to select **CCLK clock** instead of JTAG Clock in **the startup-options**.



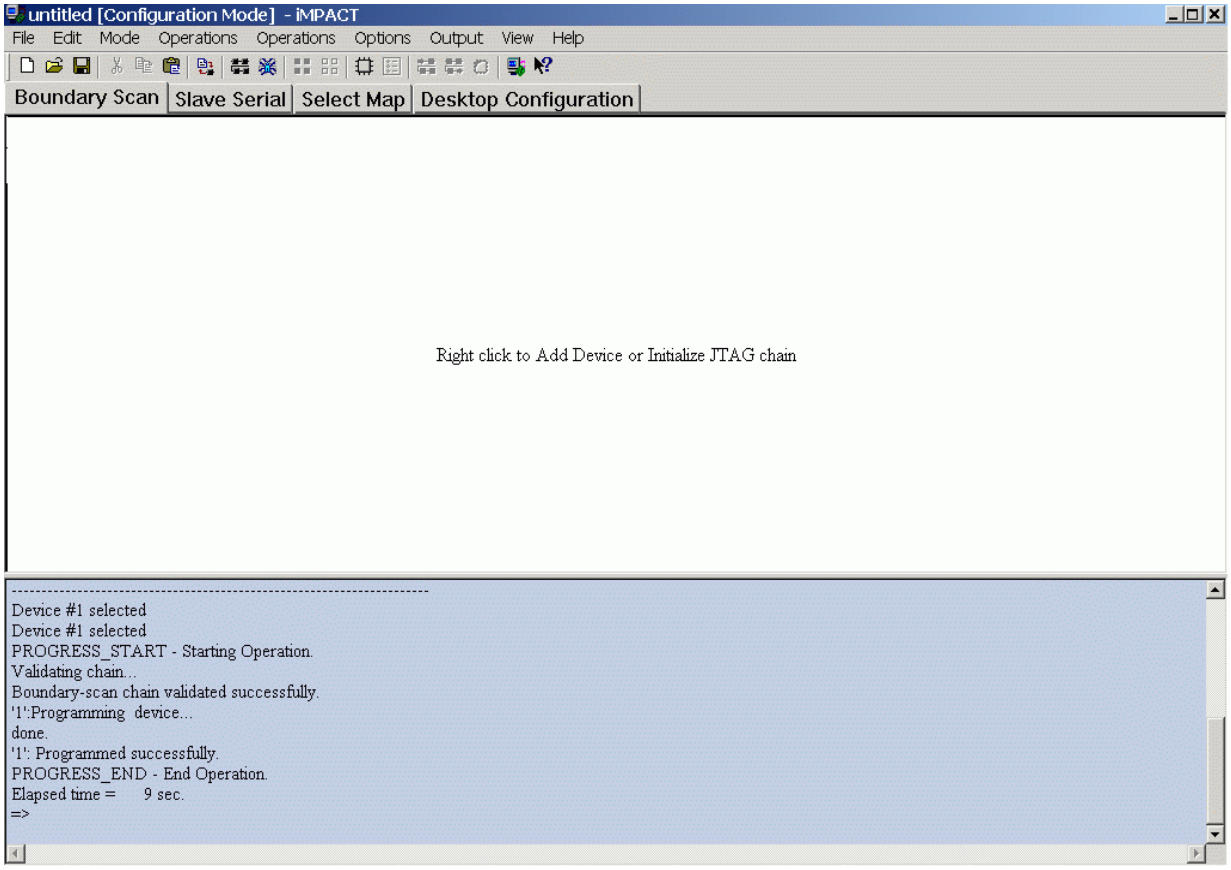
Step 12: Now run the **Generate Programming File** option, which will in turn generate the programming file for FPGA configuration. Left click on the **Generate Programming File** option, which will show the programmer below, named as Configure Devices (iMPACT).



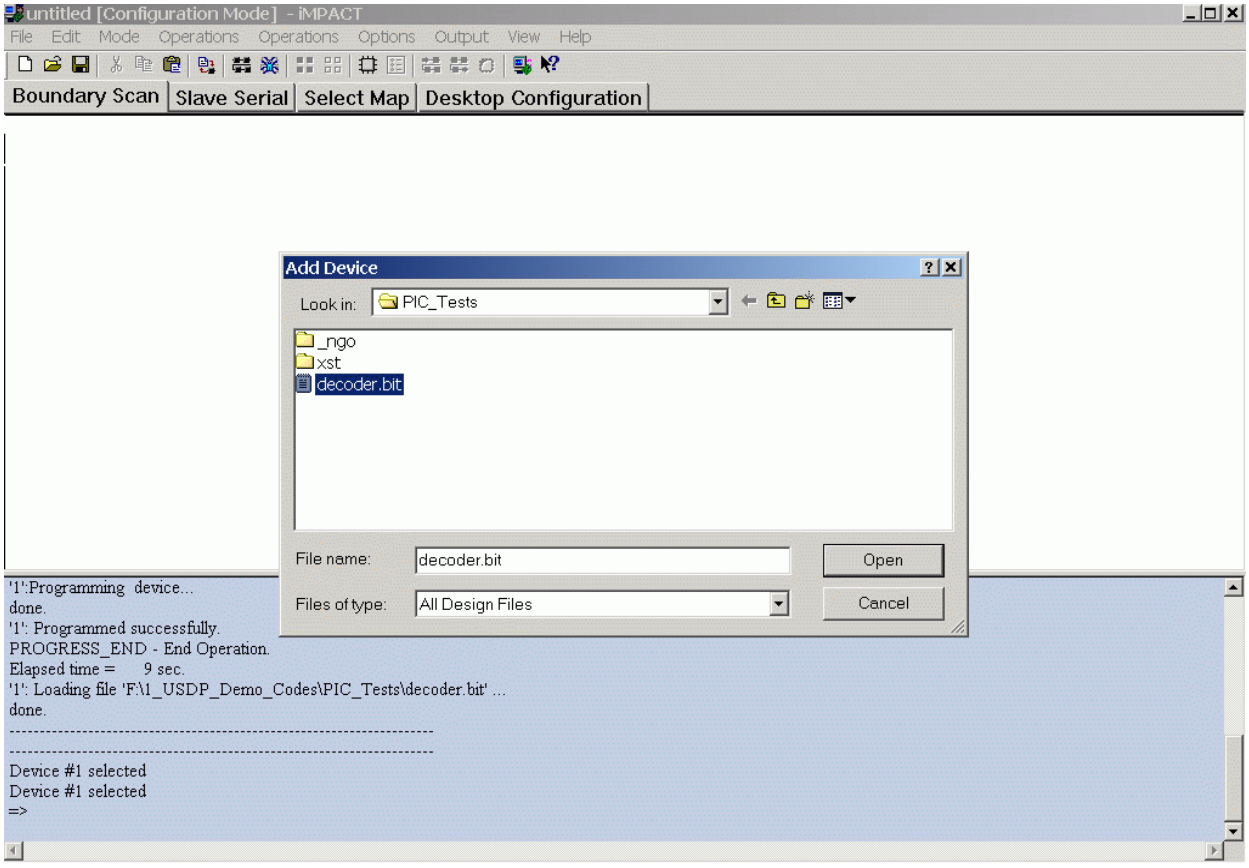
Step 13: Open the properties of iMPACT programmer, and set the **port** as **LPT1**, **configuration mode** as **Boundary Scan** for JTAG or PROM configuration. **Slave Serial** for serial programming. Leave the other options in default, and click OK and come back to main window.



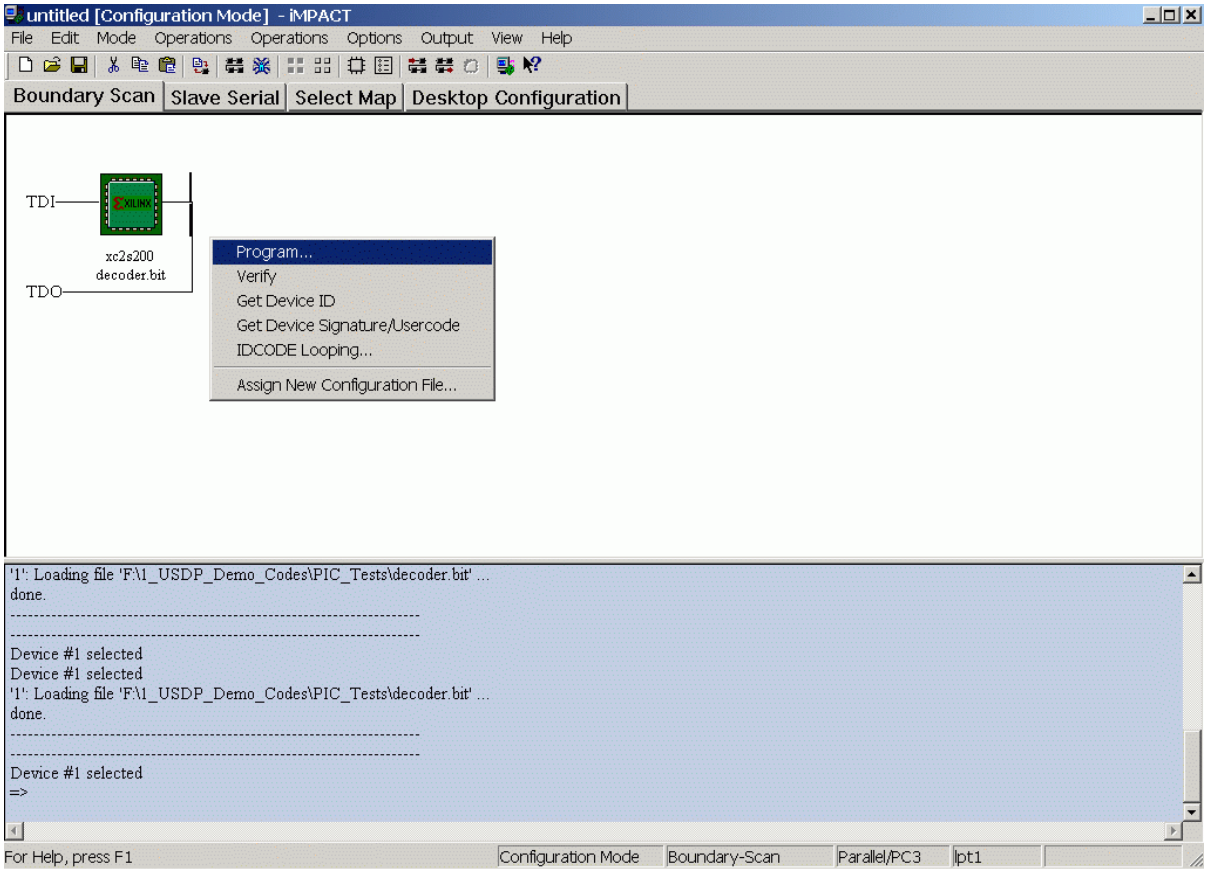
Step 14: Run the iMPACT programmer.



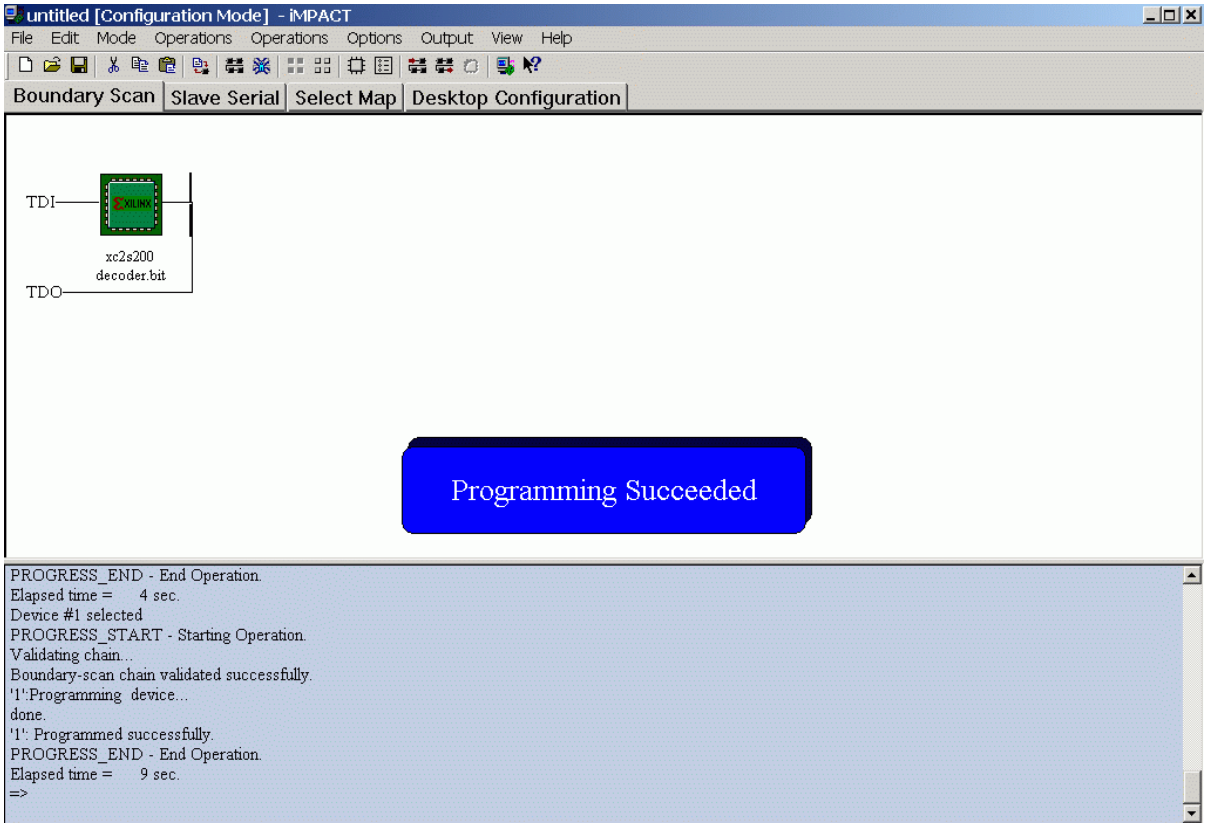
Step 15: Right click to add the device-programming file, browse to your project folder, and select the currently generated BIT file.



Step 16: Select the device file (the color would go green). Right click on the device file and select **program** option.



Step 17: Programming would start after clicking OK, if the programming is successful then the message would display on the screen.



Now check the functionality on the board and verify it by applying different inputs.

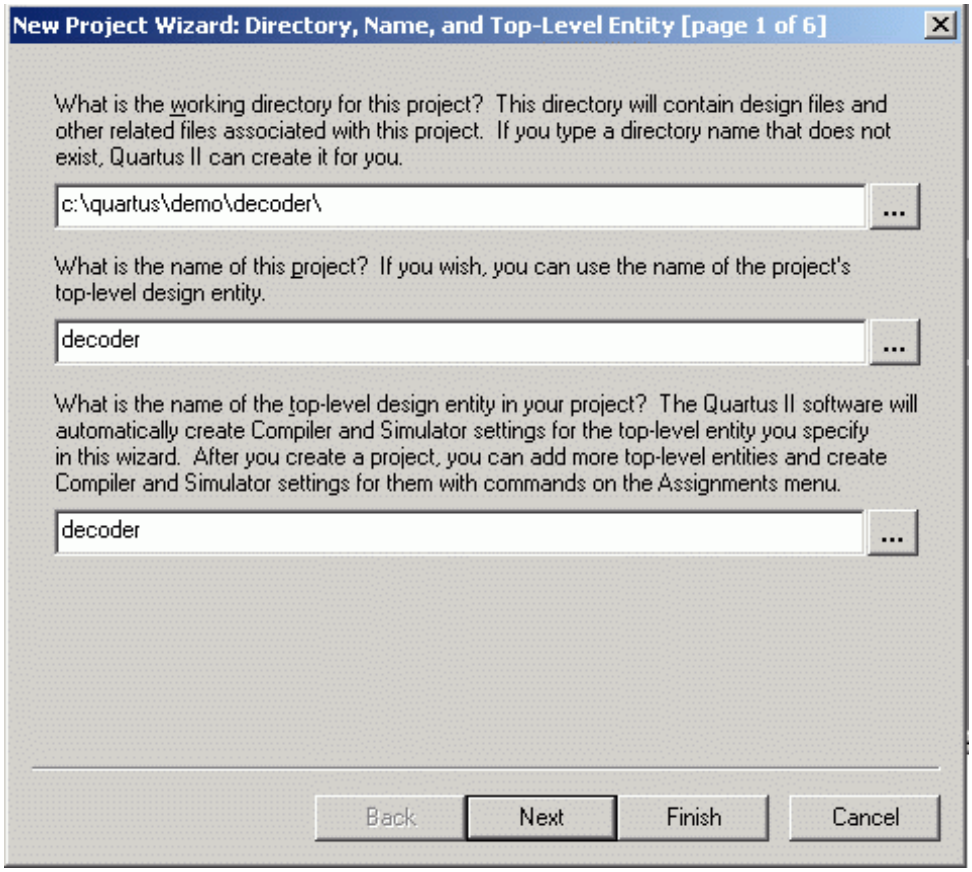
Design Flow for Quartus-II series of software of Altera.

Install Quartus-II (version 3.0 & above) software on your machine, the supported platforms are windows NT/XP/2000.

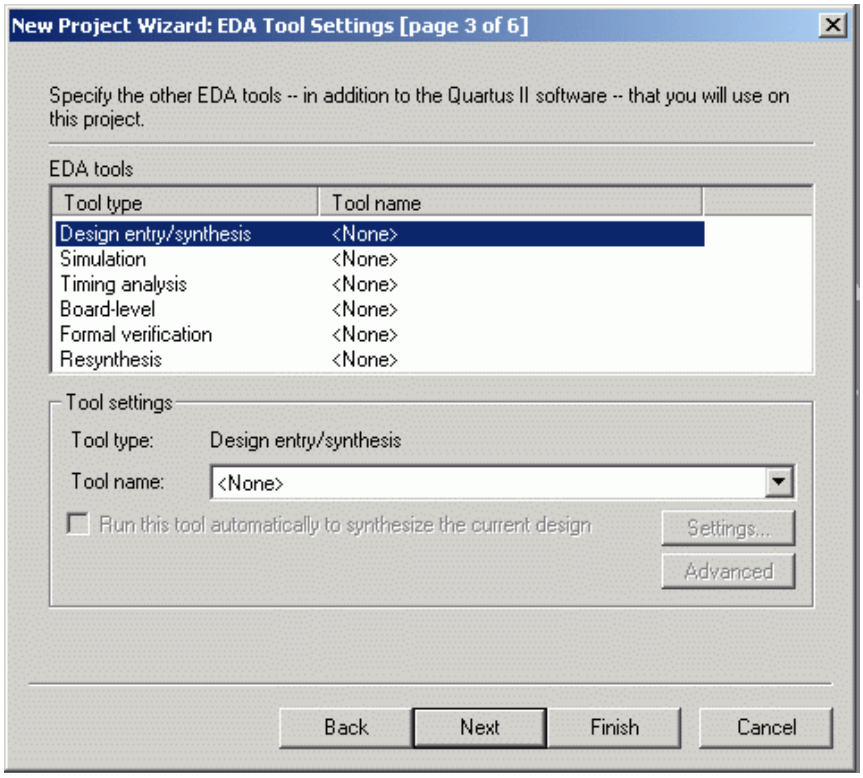
We take the same **decoder** example for implementing on the Altera ACEX1K FPGA.

Step 1: Start Quartus-II (version 3.0 & above) software.

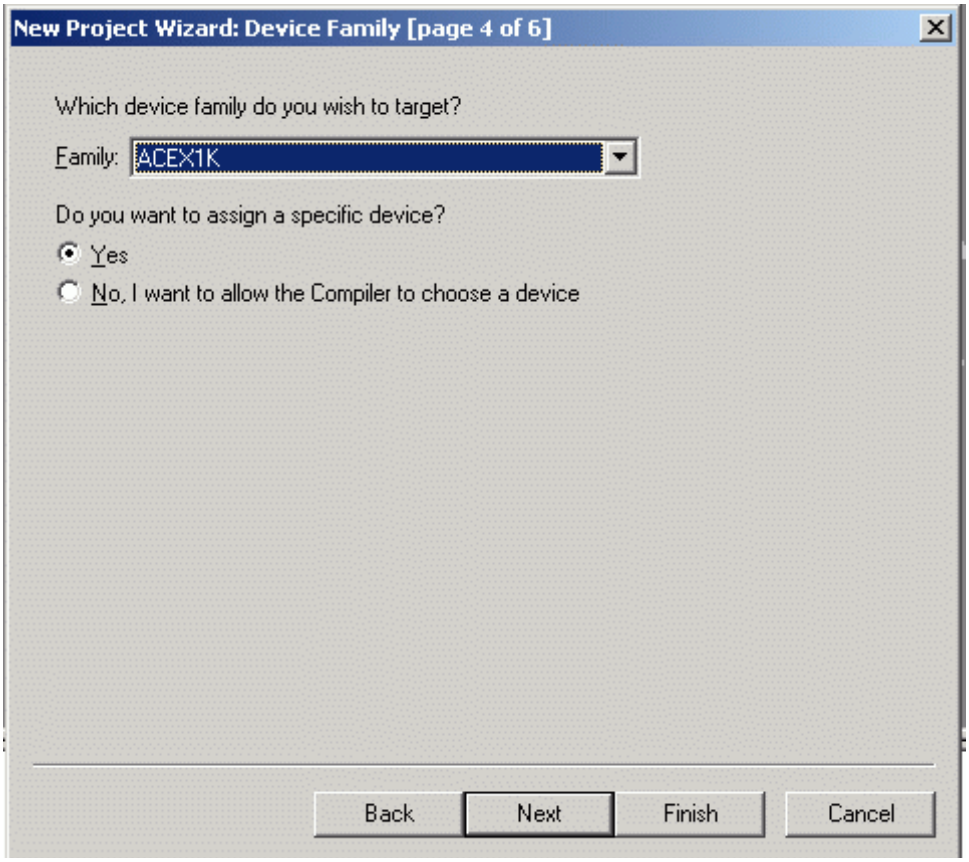
Step 2: For new project creation, go to **File** option and select new project wizard. In the opened window, specify project location and design and entity name. For eg. Entity name **decoder**, and top design name also **decoder**. **Click "next".**



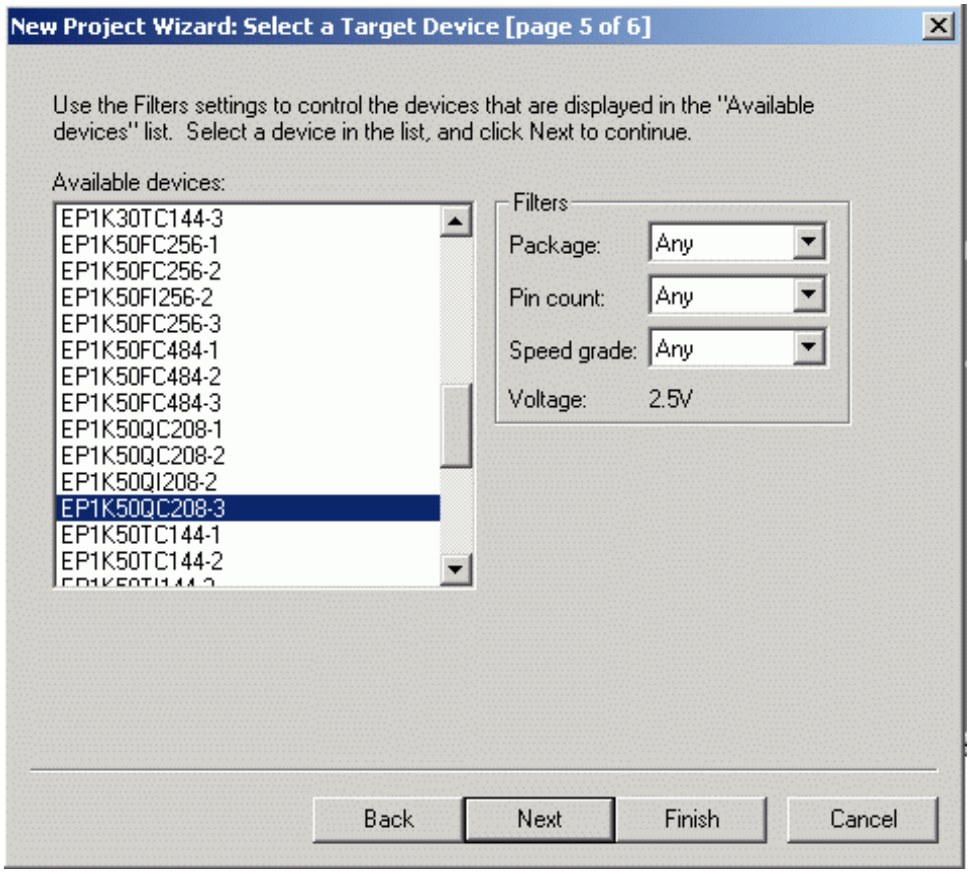
Step 3: Click "next" button till you reach EDA tool settings window, there keep all options as none, which in default will select the inbuilt design tools and softwares for the de sign processing. **Click "next".**



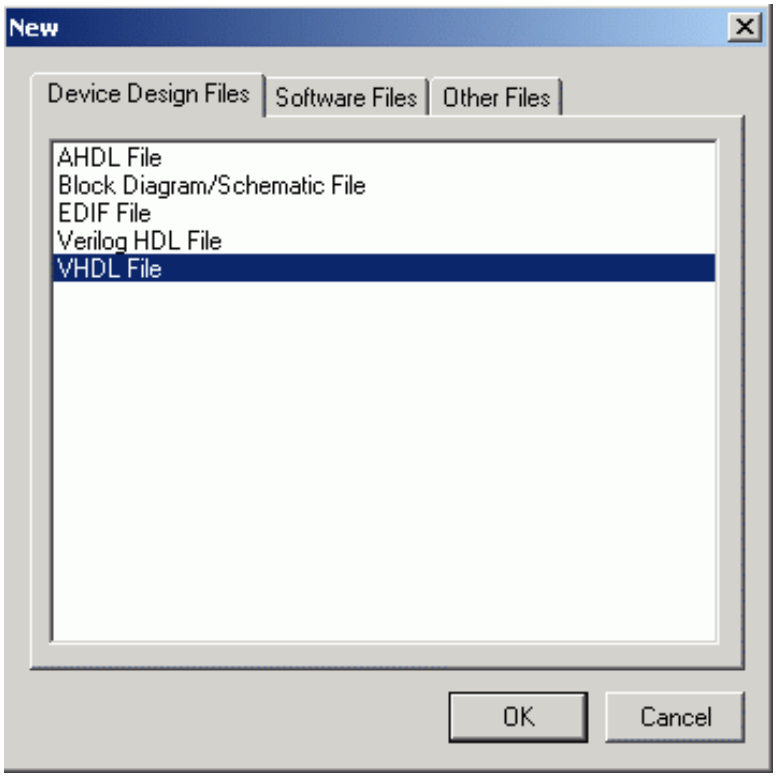
Step 3: Select ACEX1K device family in the next window. **Click "next".**



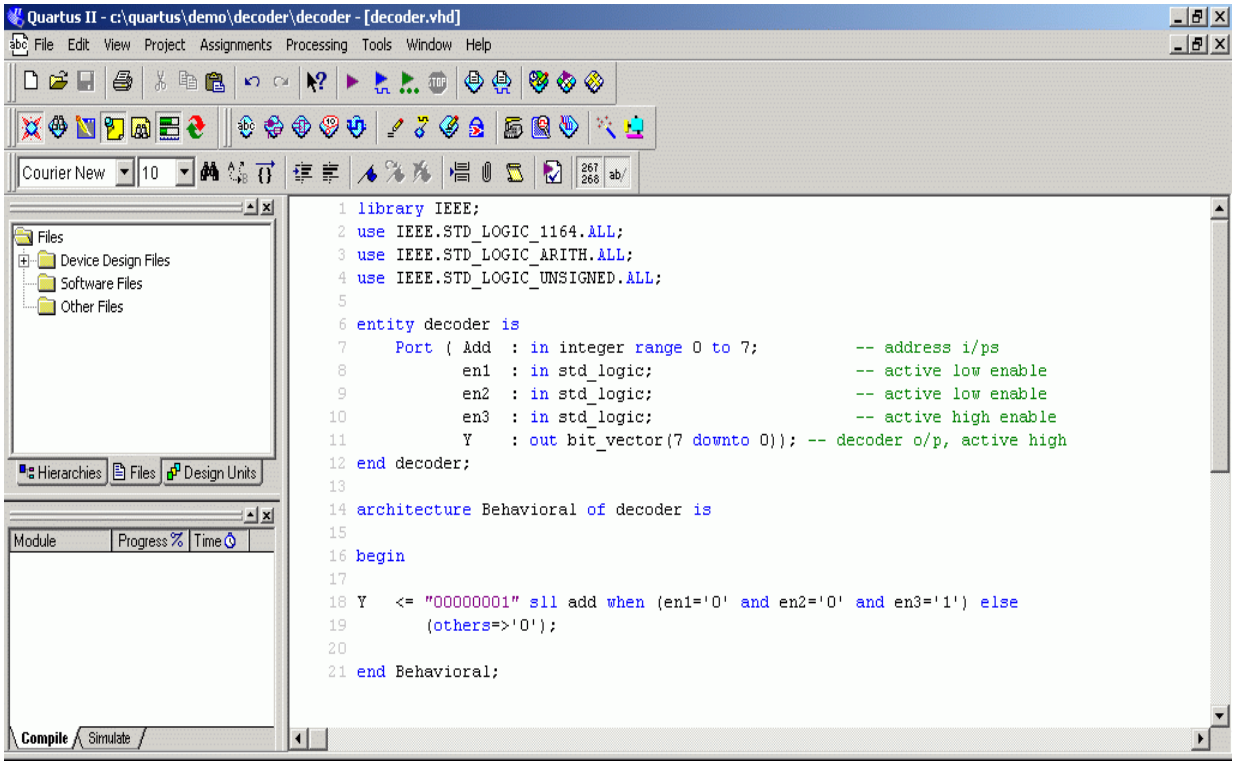
Step 4: In the next window, select the device as **EP1k50QC208-3. Click "next".**



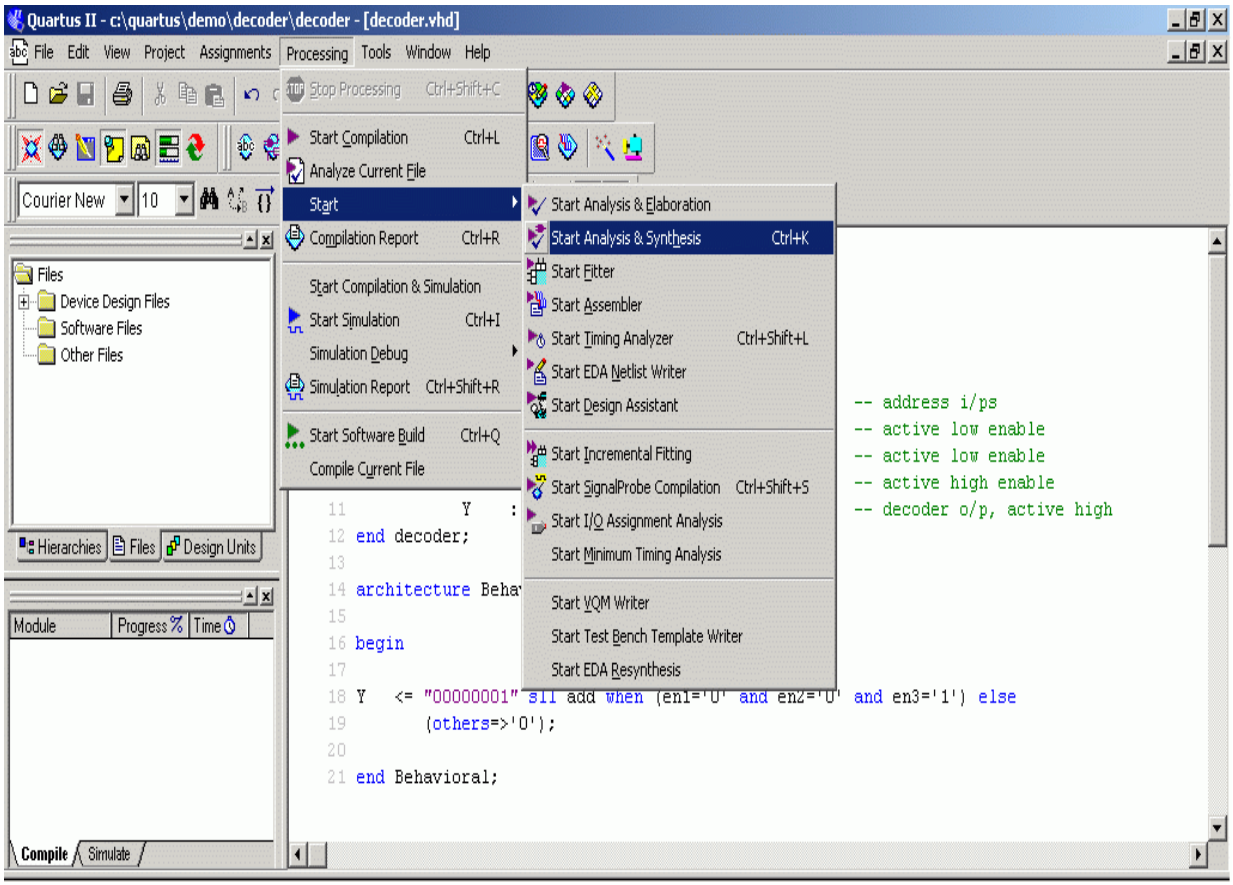
Step 5: Click "Finish". And the new project would be created. Now we need to make and add new design file in the project. So goto **"File"** menu, and click **"New"**, and select **VHDL File**, in the "device design files" tab. **Click "OK".**



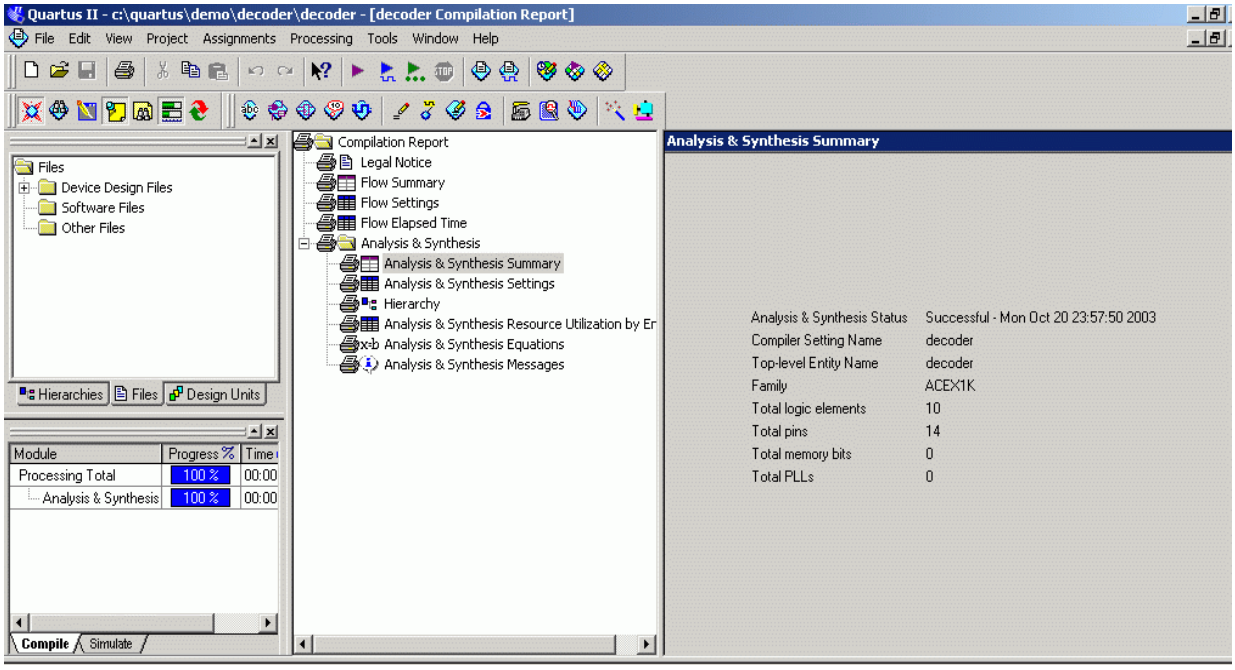
Step 6: Write the VHDL code for 3:8 decoder, and save the file as **"deoder.vhd"**



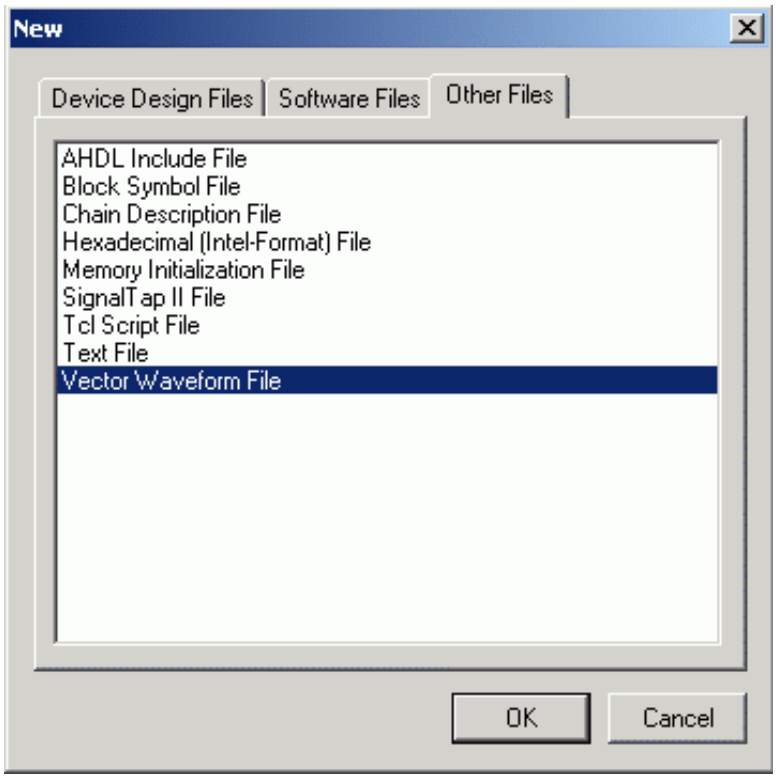
Step 7: Now goto **processing** menu- then – **start** -, click **start analysis and synthesis.**



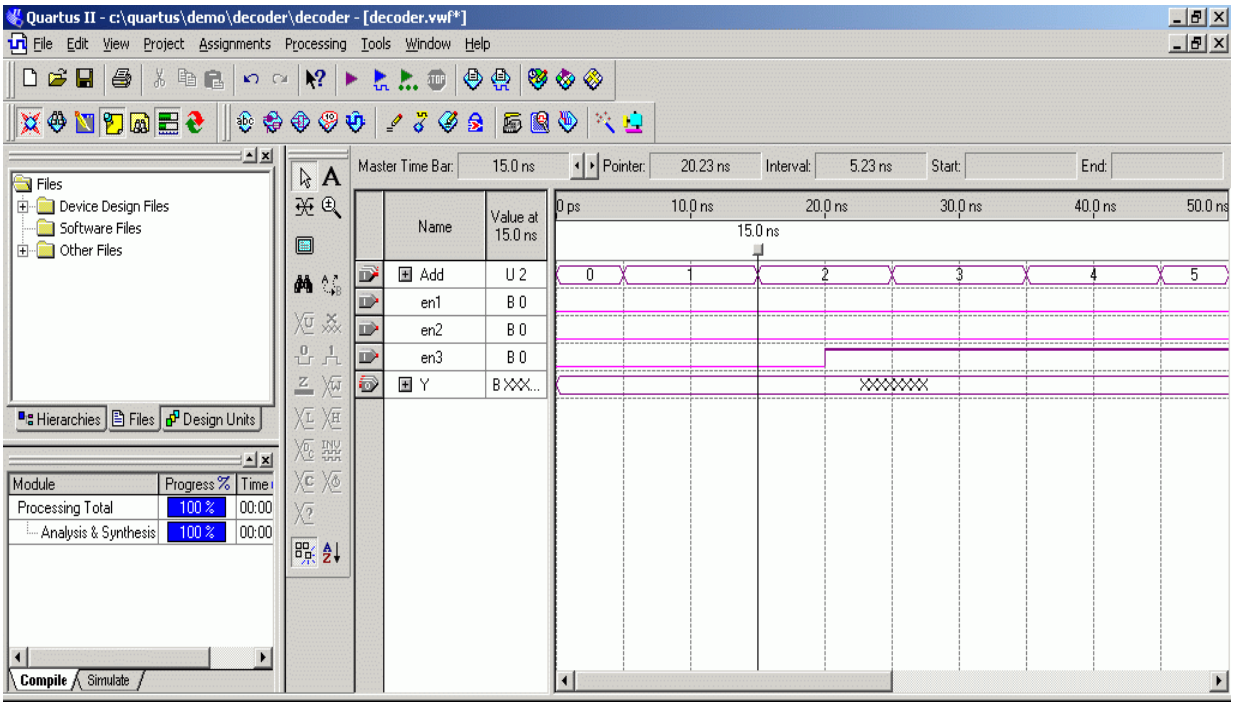
Step 8: Read the synthesis reports



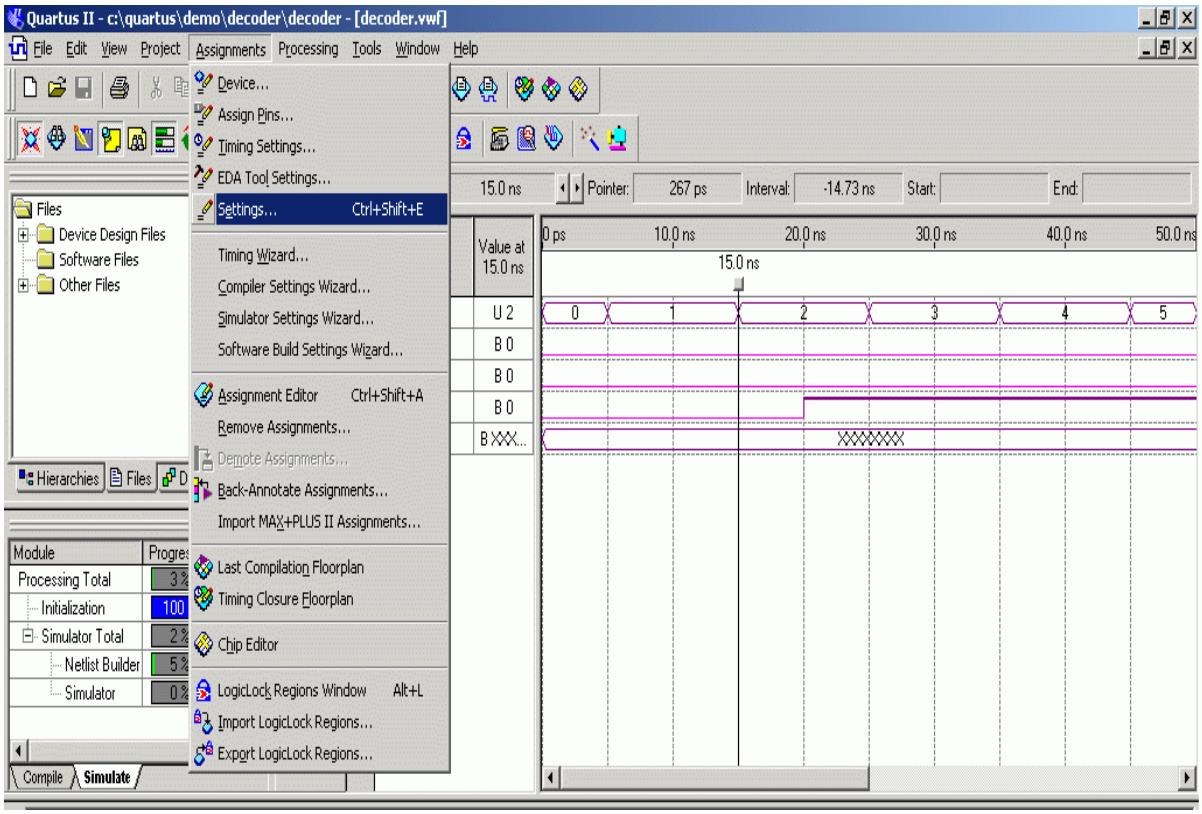
Step 9: For performing simulation, we need to create stimuli file from where we can apply input signals and watch the o/p waveforms.
Goto **file** menu, and click **new** file, goto **other files** tab, and select **"vector waveform file"** option.



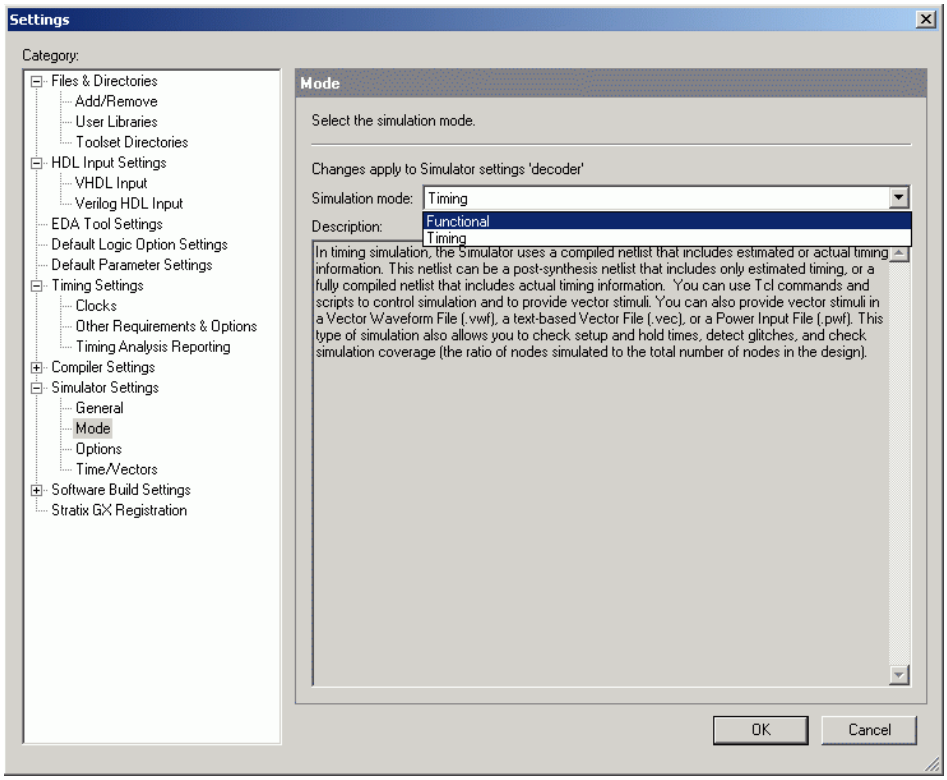
Step 10: Add the entity signals in the waveform window, and apply different sets of value to check the functionality. Save the file as the same name of entity, **decoder.vwf**



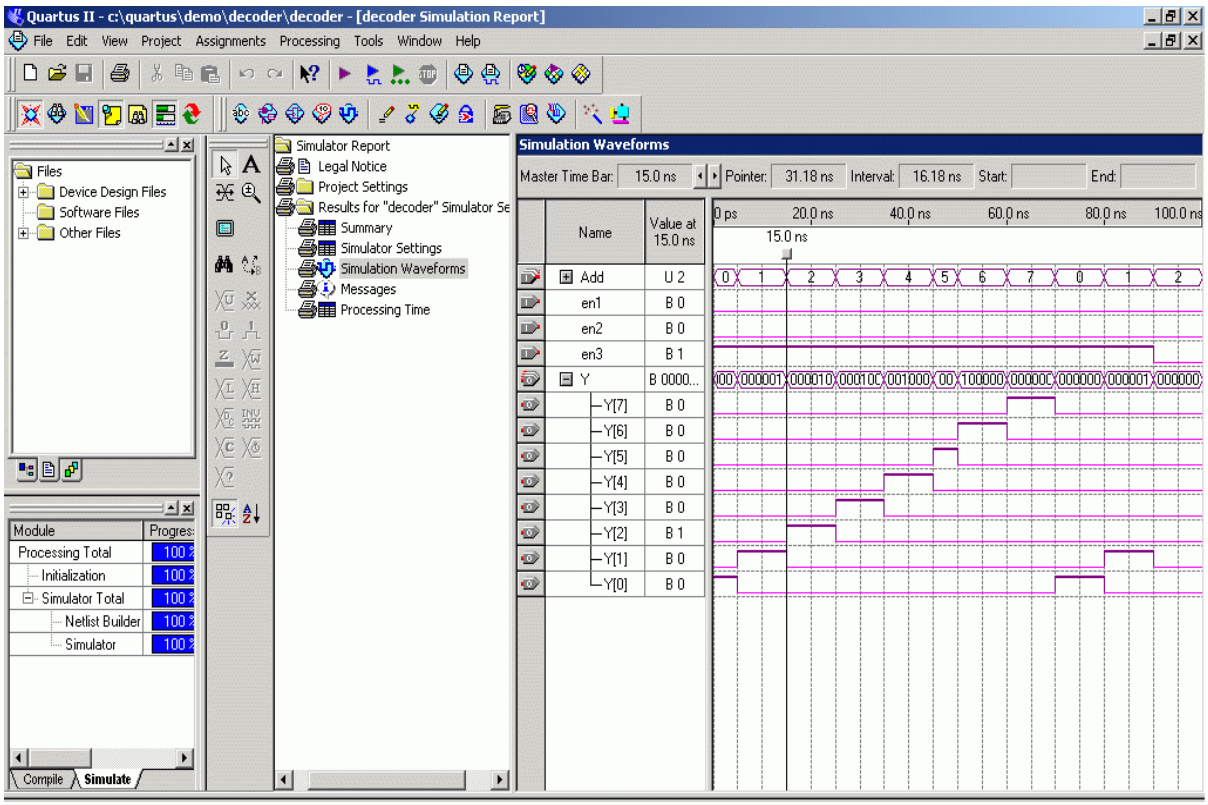
Step 11: In Altera Quartus-II software you can perform **Functional** and **Timing** simulation. For simulation mode settings, go to **assignments** menu, and click **settings**.



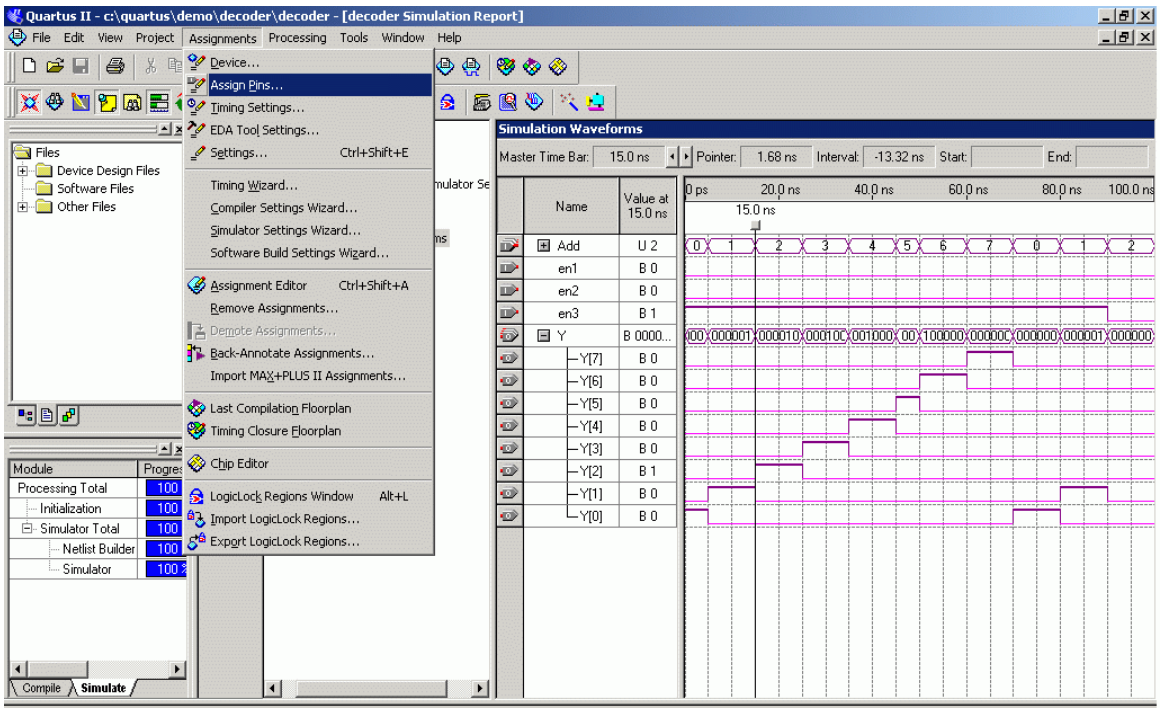
Step 12: Now goto **simulator setting**, then to **mode**, and in the right-hand side window, select the simulation mode to **Functional**.



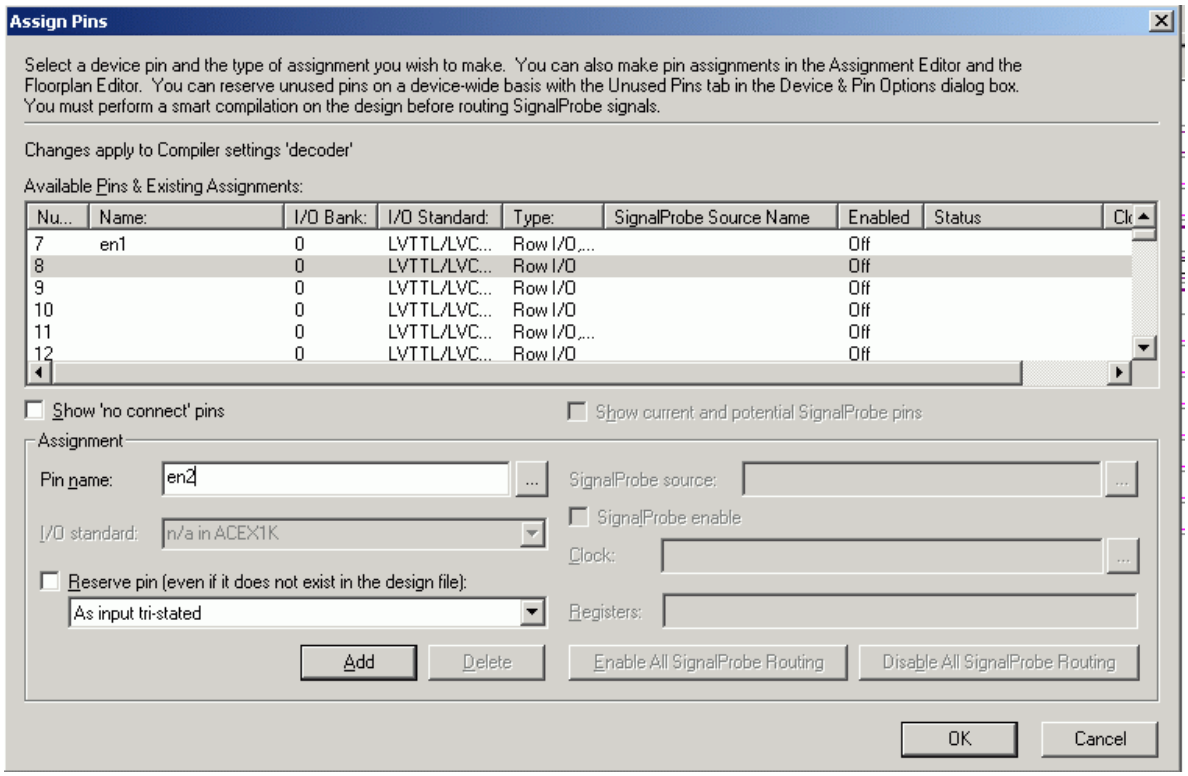
Step 13: After clicking OK, come back to main window, and goto processing window, and click start simulation, the Quartus-II will start the simulation the result would appear in couple of minutes. Observe t he results, if found bugs, then change VHDL code and start simulation again.



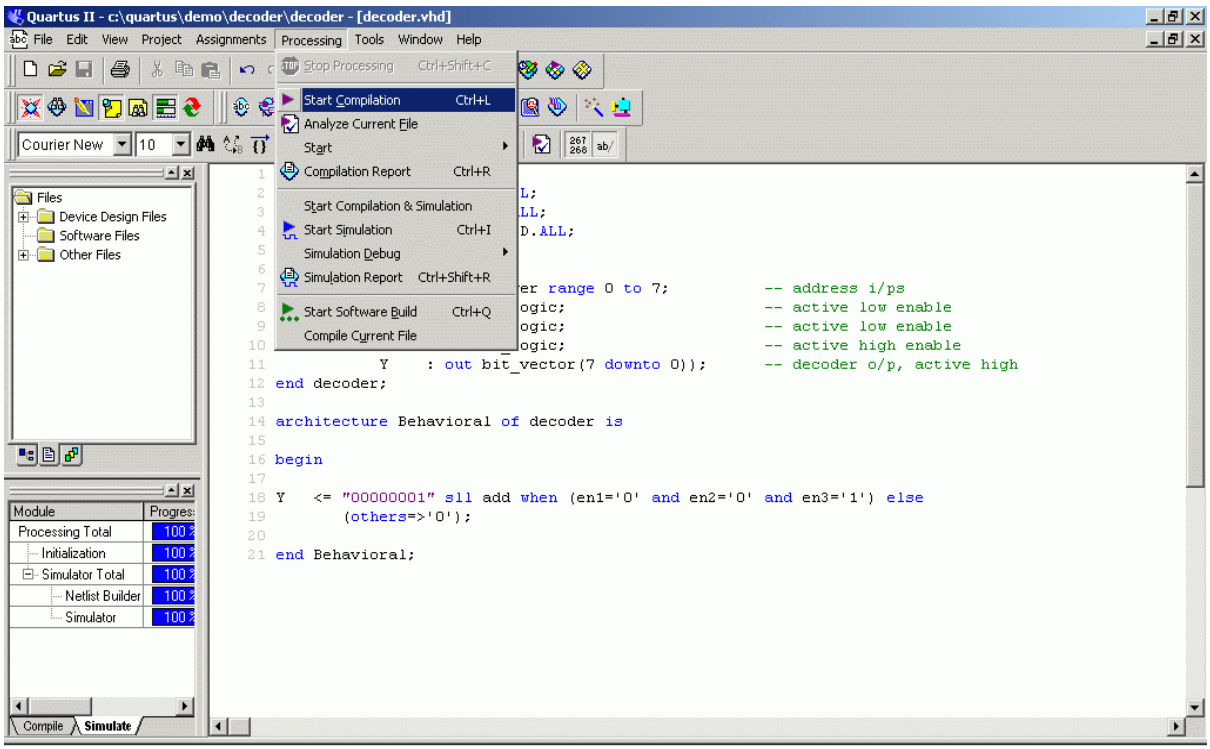
Step 14: Once the simulation results are found correct, then we need to implement the design in the target device. For this we need to lock our design I/Os with the Kit I/O pin details. Goto **assignment** menu, click "**assign pin"** option.



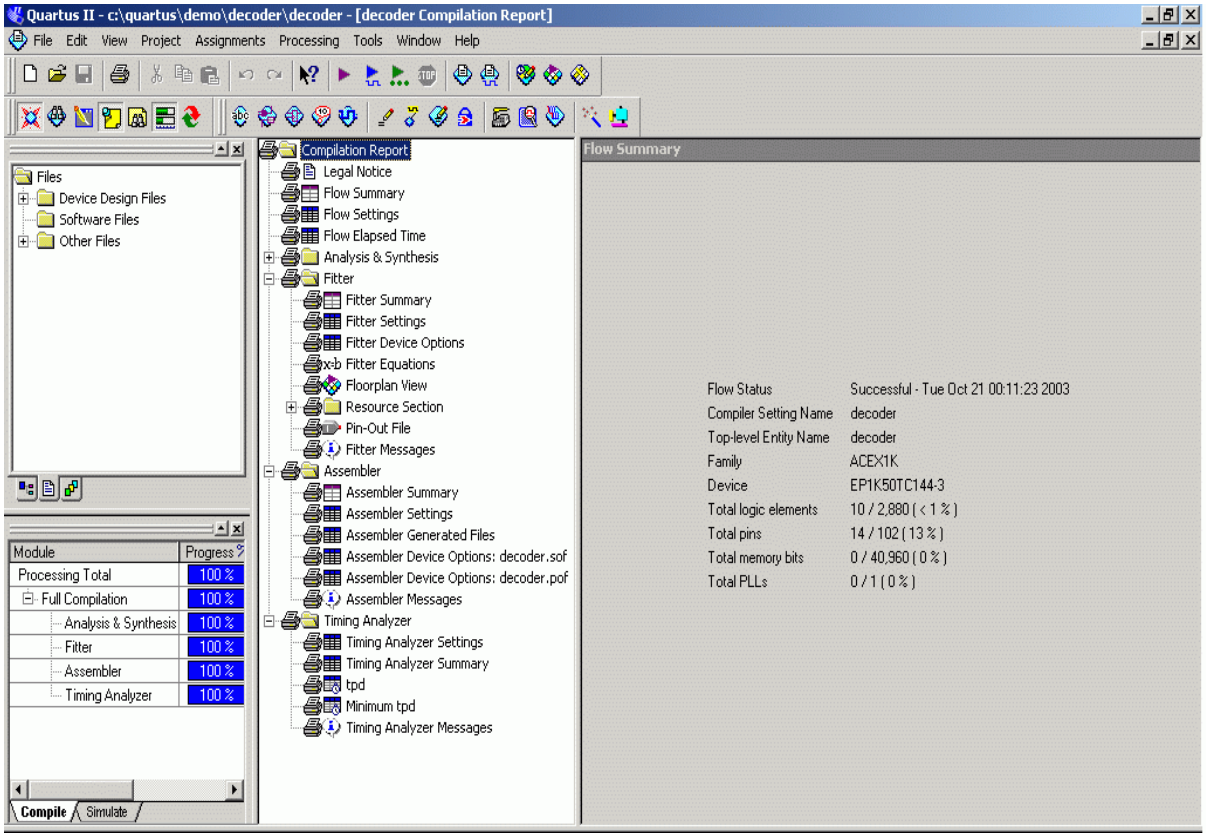
Step 15: Looking at the pin assignment chapter, lock the ACEX 1K FPGA I/O with the particular pin no., for this select the I/O number on the LHS, name the design I/O in the bottom pin name option, and then click add, the particular signal will be locked to that pin number



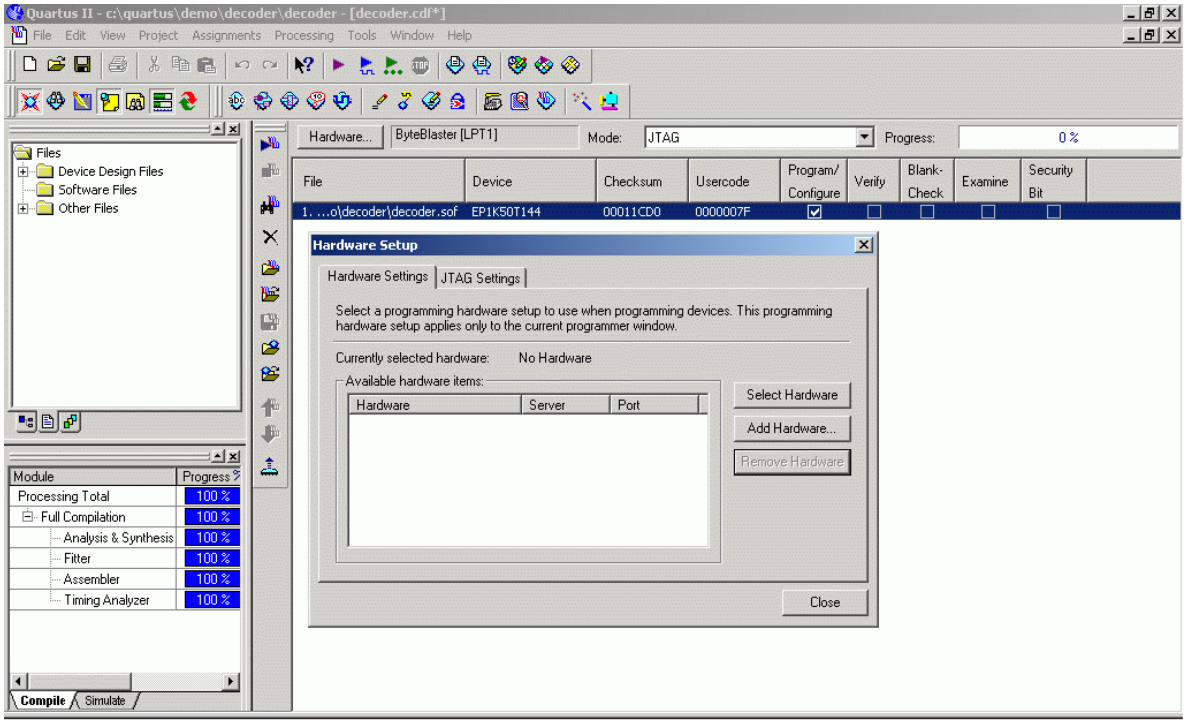
Step 16: Once the pin assignment is over, come back to main window. Now we need to implement the design on the particular device. So goto **processing** menu, and click **start compilation** process. Which will place & route the design in FPGA and generate the programming file.



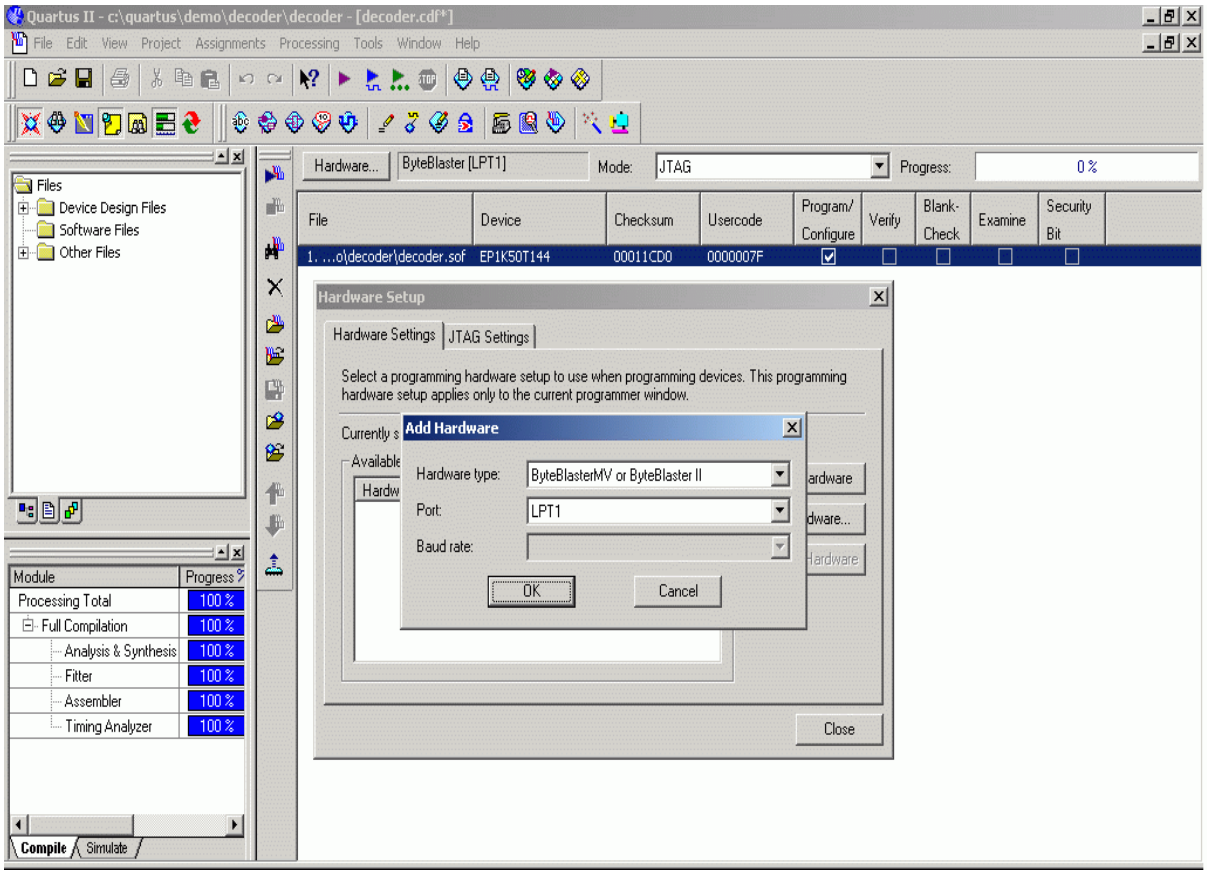
Step 17: Once the **compilation** process is over, user can check the **reports** and see the floorplan window.



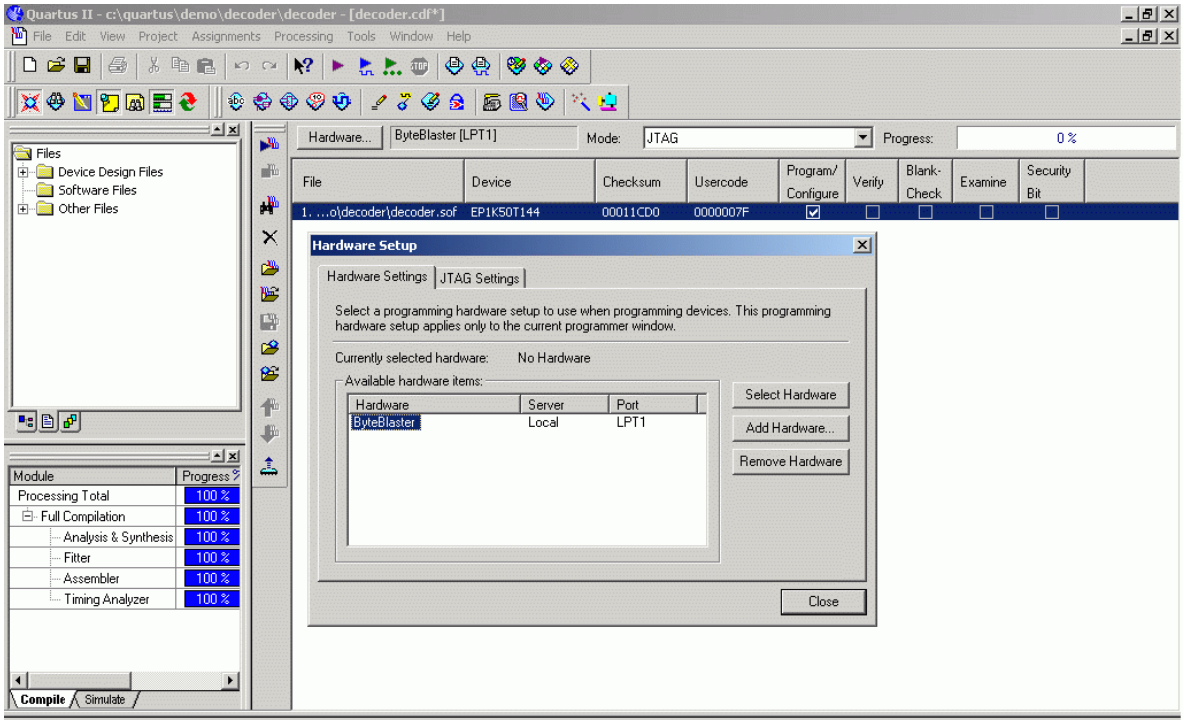
Step 18: Now we need to program CPLD, for this goto tools menu, and click programmer. Which will open the programmer; the software will automatically add the programming file (decoder.pof). In the opened window select the **program/configure** option. Now we need to select the programming hardware, for which click the hardware tab on the top LHS of programming window.



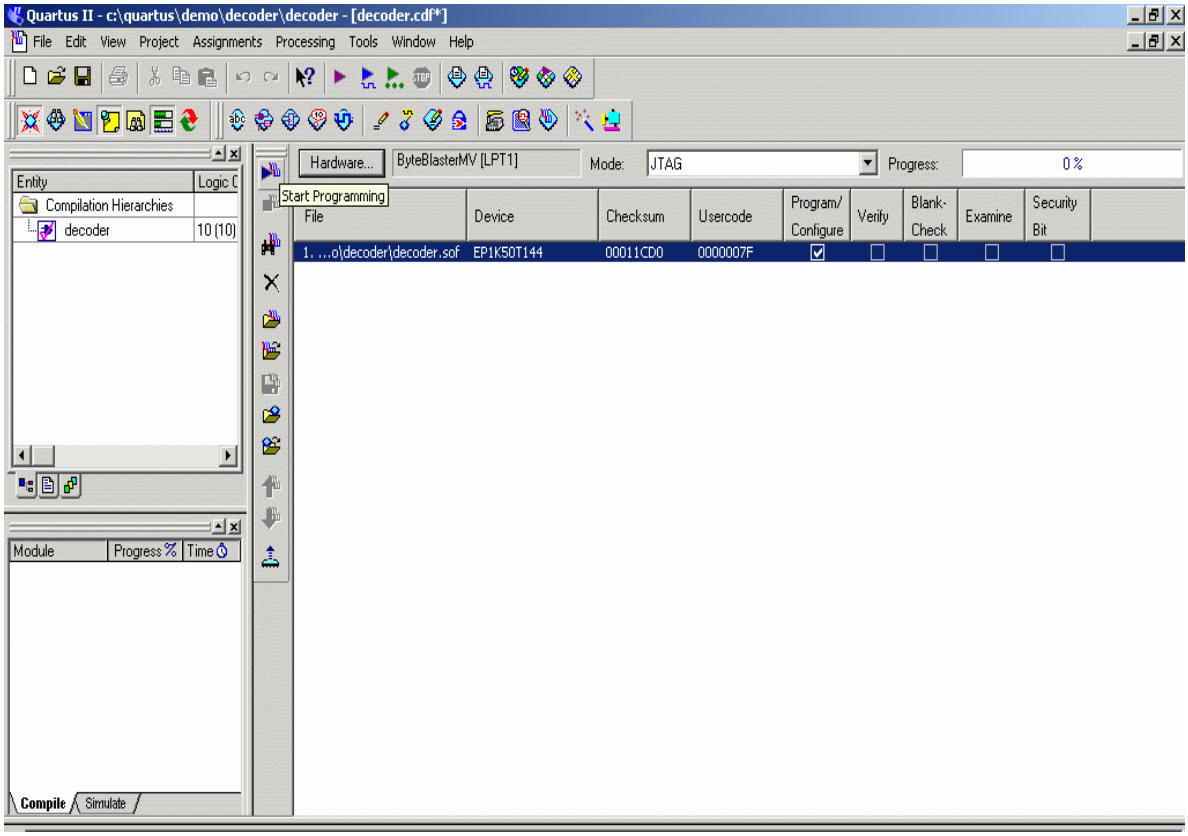
Step 19: In the opened window Click **add hardware** tab, and select the hardware type as **ByteBlasterMV or ByteBlaster II** and port as **LPT1**.



Step 20: Come back to **hardware setup** window, and click the **select hardware** tab and close the window.



Step 21: Now click the **start-programming** button (play symbol) on the top LHS of the programming window (keep the program/configure option selected).



Check the design functionality on the board, by applying signal from switches or other points.

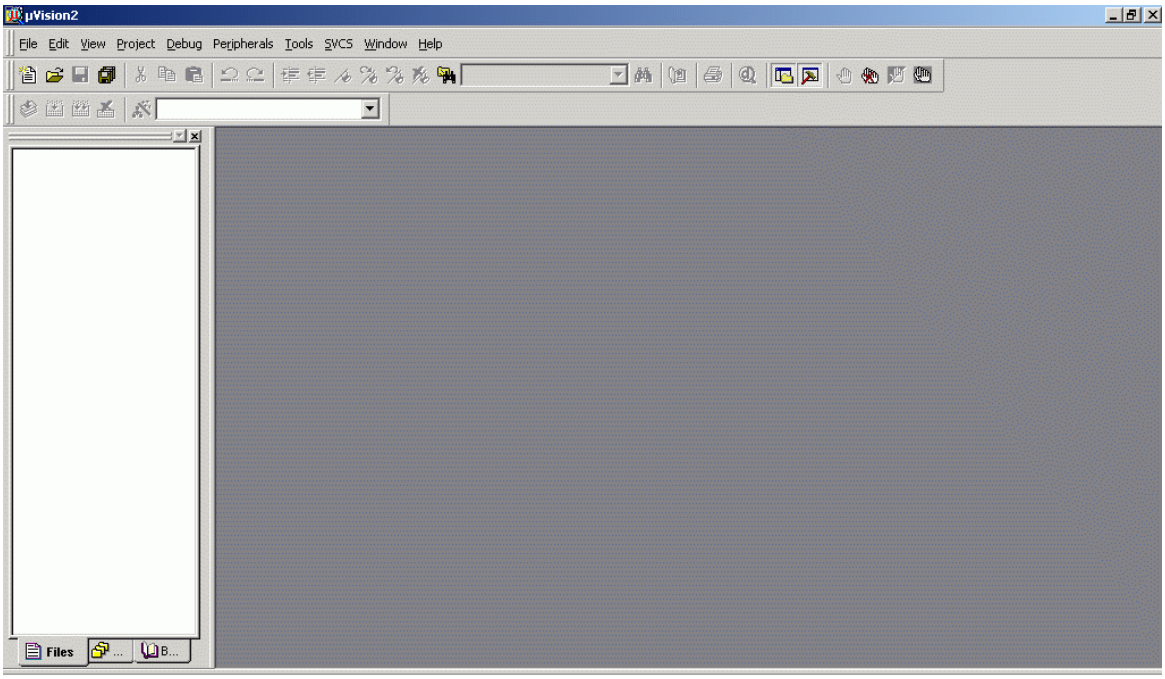
Using Keil Compiler

Designers can use compiler from **Keil**, one of the compilers available in market. **Keil** is a cross compiler supporting the 8051 based architecture controllers from various vendors.
Compiler support the source codes written in 'C' and assembly languages.

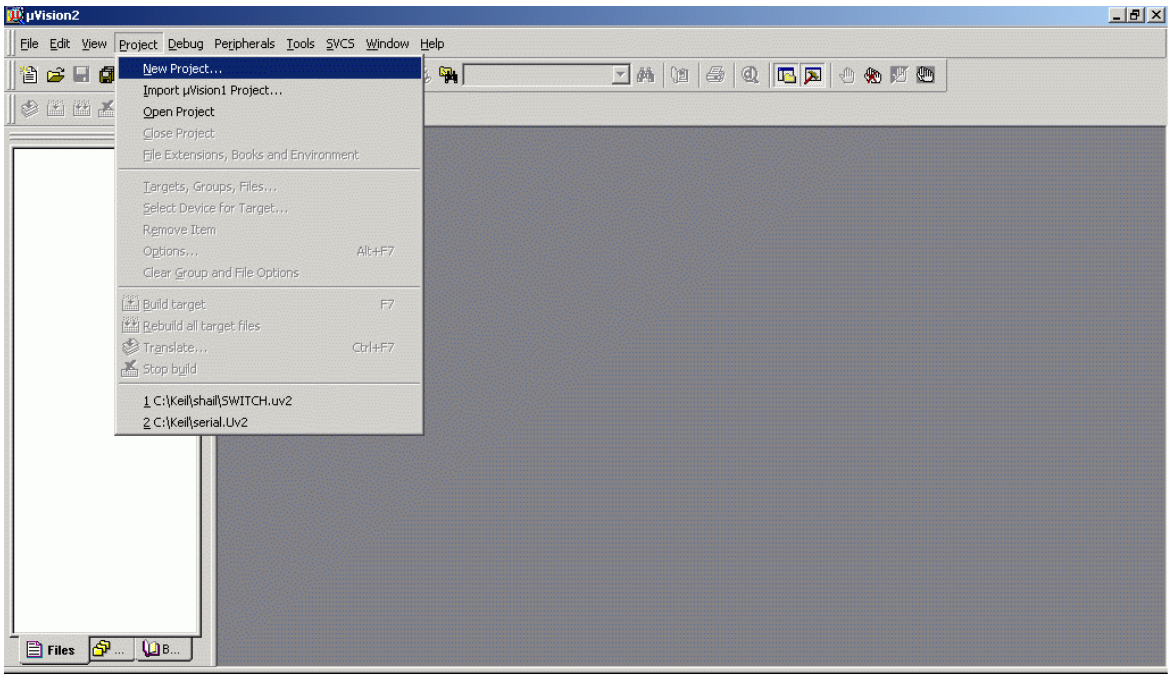
For starting up with USDP, we have made a design flow guide for using the **keil** compiler, but for more information users can surf the help index of keil compiler,

Install the **Keil** compiler from provided CD-ROM; you can use the evaluation version at start up, which supports the program code upto 2KB, which is sufficient for small development purposes.

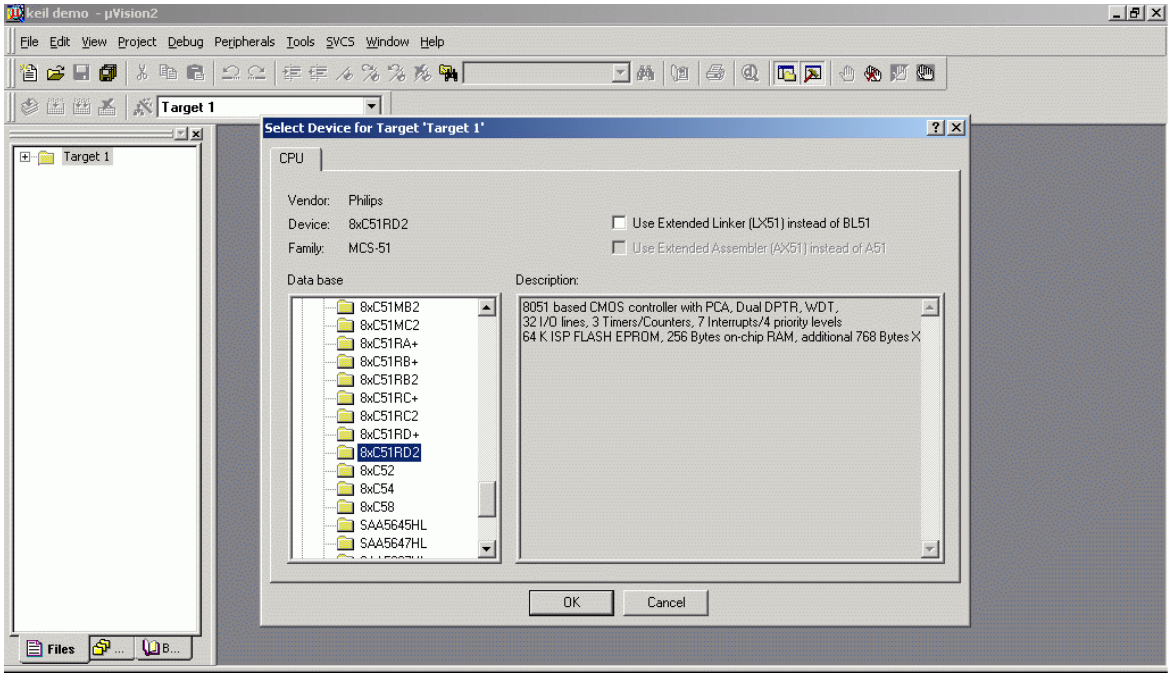
Step 1: Run the **keil** compiler EXE, which in turn will open the compiler.



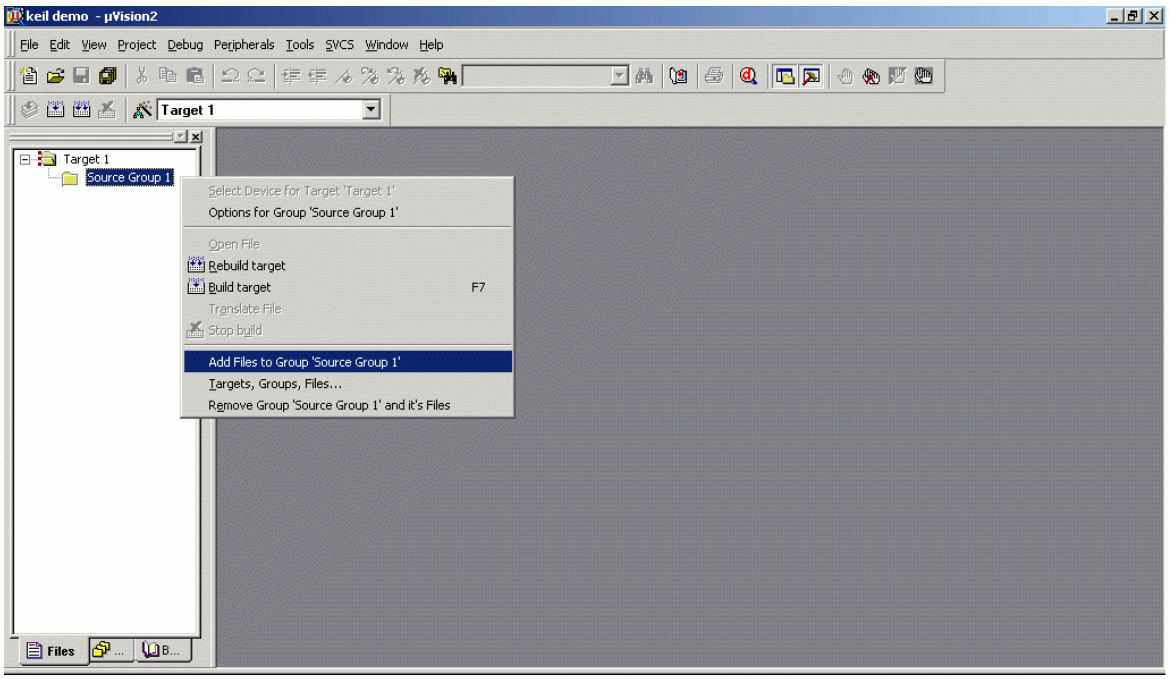
Step 2: Go to project menu and left click the **new project** option.



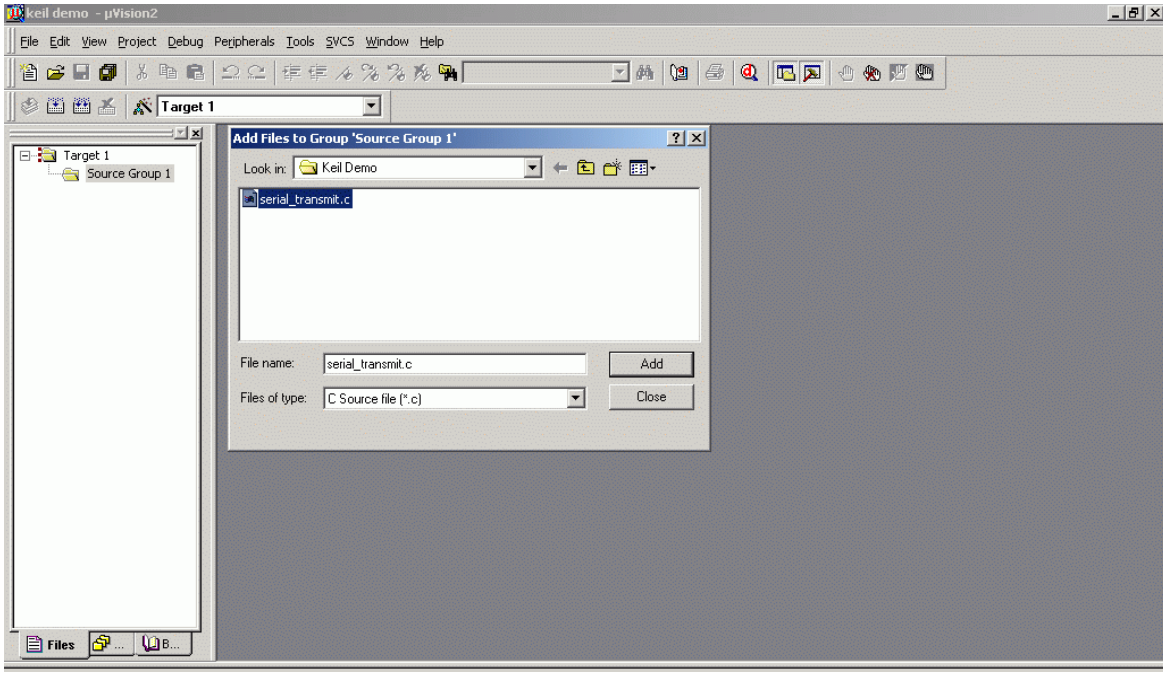
Step 3: In the opened window user has to give project name and select the folder for the project creation, for example, name it **keil demo.** Click OK, and in the next window, you have to select the device to work on, the vendor name is Philips, and the device number is 8xC51RD2. The window will look like as below.



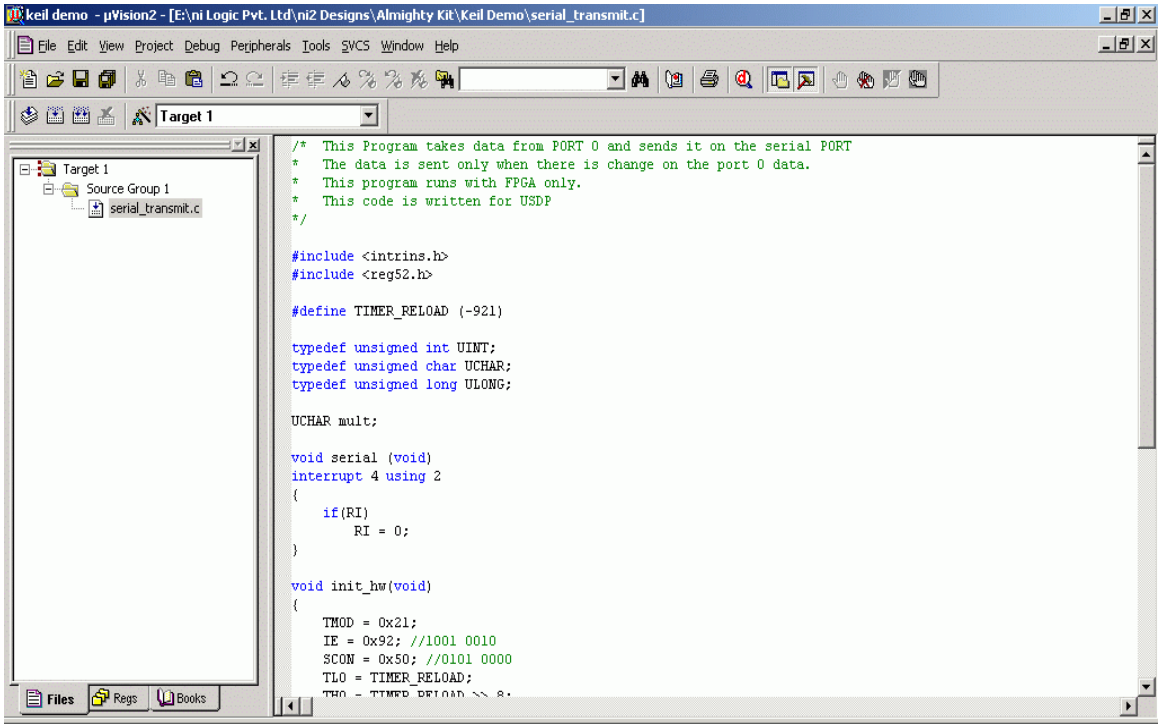
Step 4: Click ok on the above window, and the project would be created. Now the user has to add or create the source code for the controller. For example for now designer can use the source code provided along with the USDP code examples. So right click on **source group** and click **add files** to group.



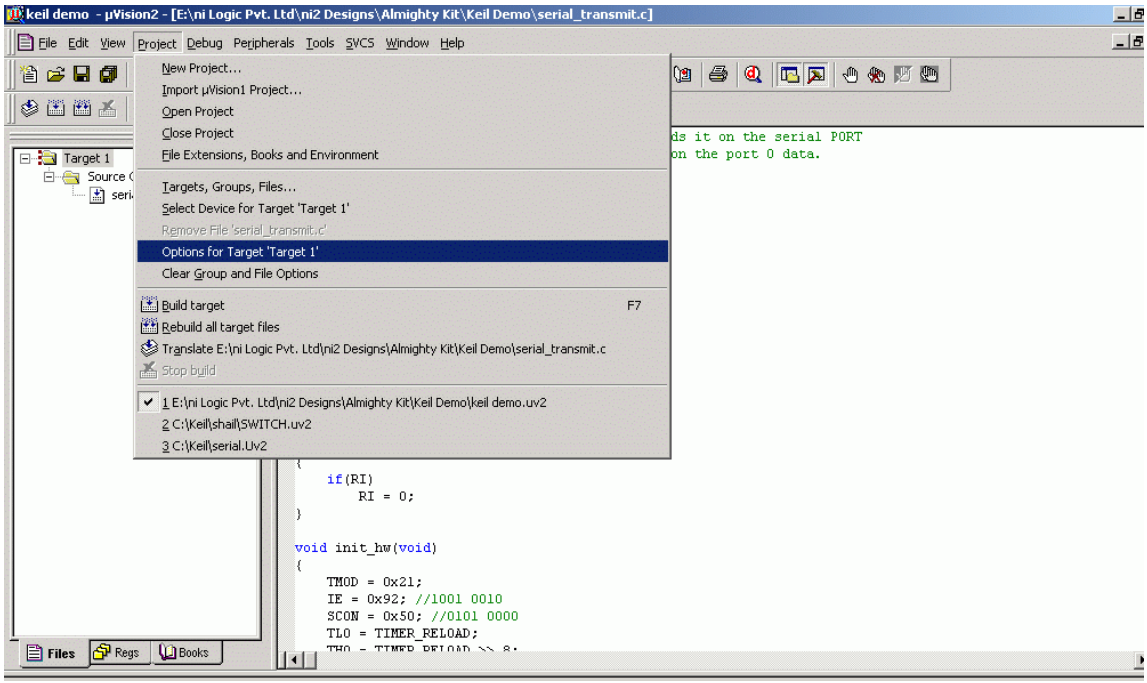
Step 5: Browse for the demo source code **serial_transmit.c** and click add to insert it in the project.



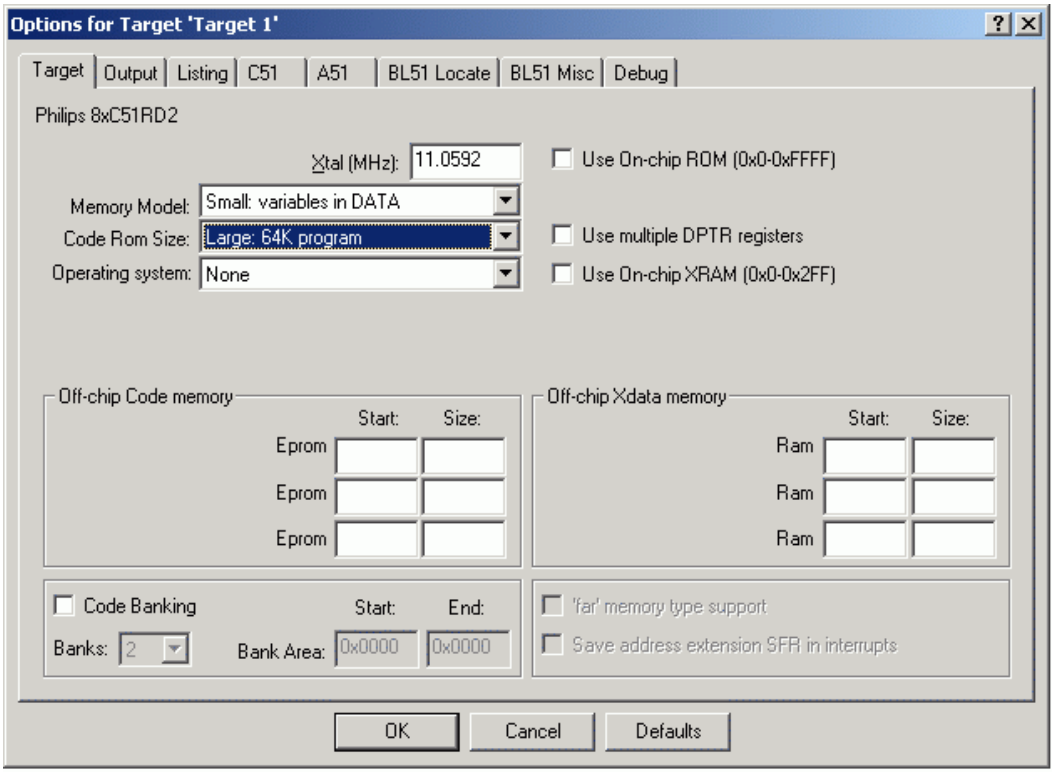
Step 6: User can have a look on the demo source code, which takes data from Port 0 and sends it serially to PC. The source code is in 'C' language; users can also use assembly code for designing.



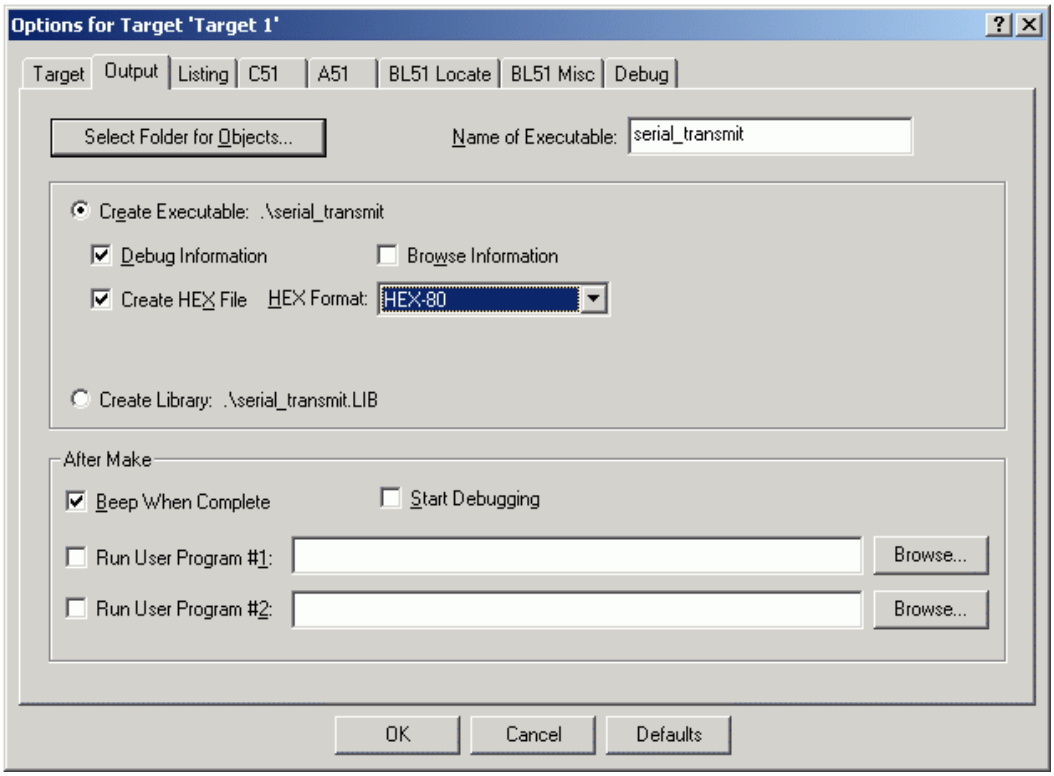
Step 7: For building the program code or HEX code, we need to set some parameters in the compiler. Go to project options, and click **option for target**.



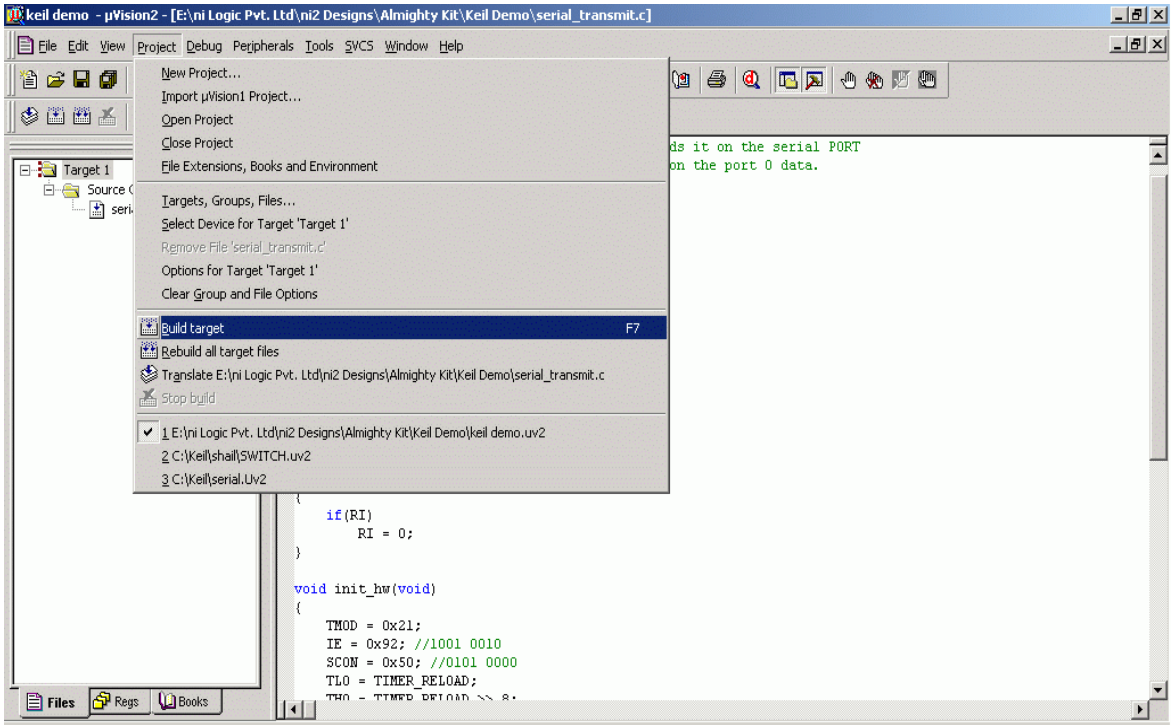
Step 8: In the opened window, go to **target** tab, and in the window, set the **Xtal** frequency as **11.0592 MHz.**



Step 9: Now goto **output** tab and set the **name of executable** file as **serial_transmit**; set the option create executable; and set **create HEX file** option. Finally click **OK** on the bottom side and come out of the window.



Step 10: Now to build the HEX file, goto **project** menu, and click the **build target** option.



This will generate the HEX file in the project folder, which you can use to program the controller using the provided Flash Magic programming software.

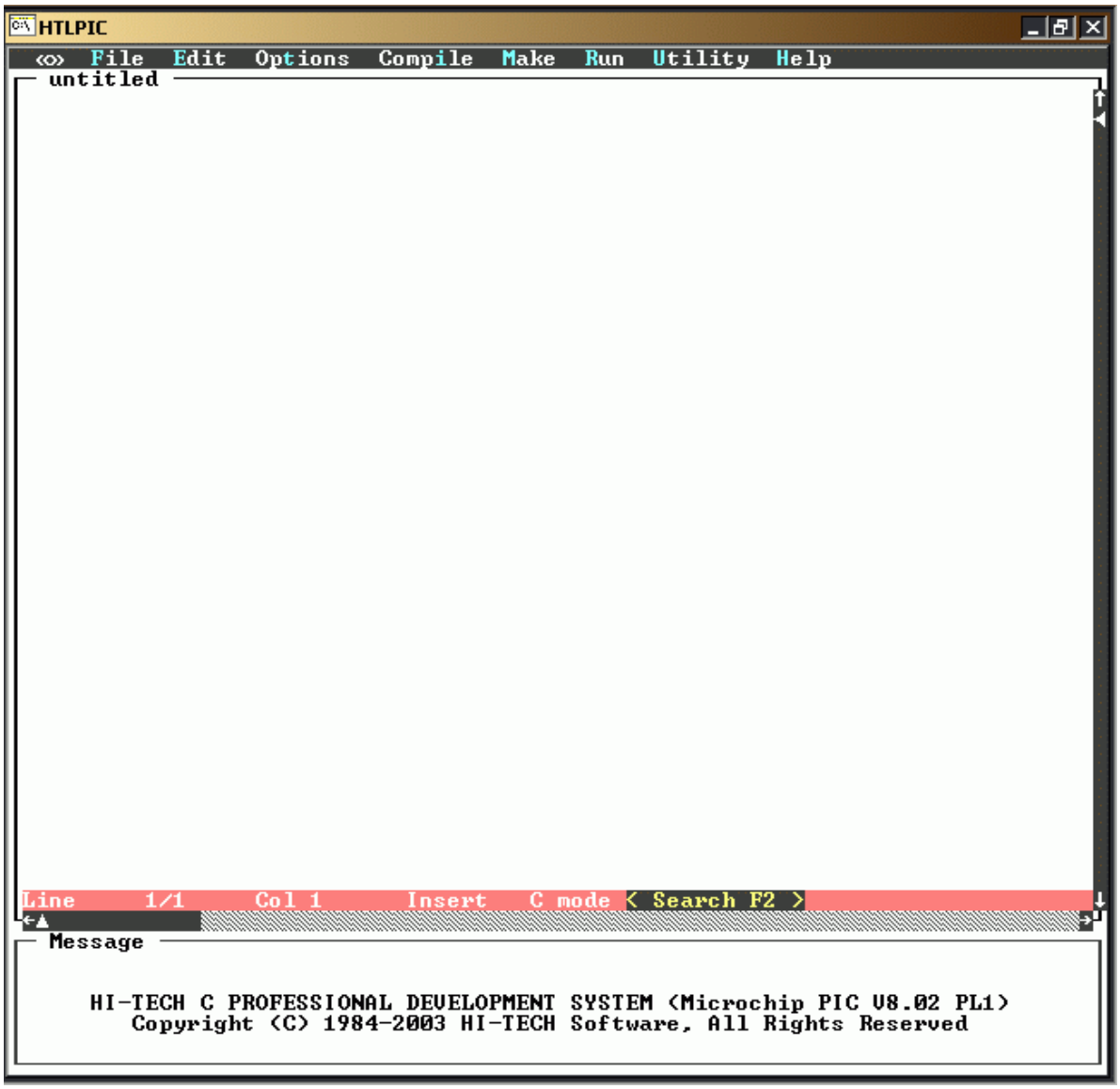
Using PIC C Lite Compiler

Designers can use **PIC C Lite** compiler from **HI-TECH Software**, one of the compilers available in market.
PIC C Lite is a DOS based cross compiler supporting **Microchip** PIC microcontrollers.
Compiler support the source codes written in 'C' and assembly languages, and works on Win 98/2000 platforms.
The choice of compiler is solely dependent on the user; they can also use the compilers available from other vendors, like MPLAB or others.

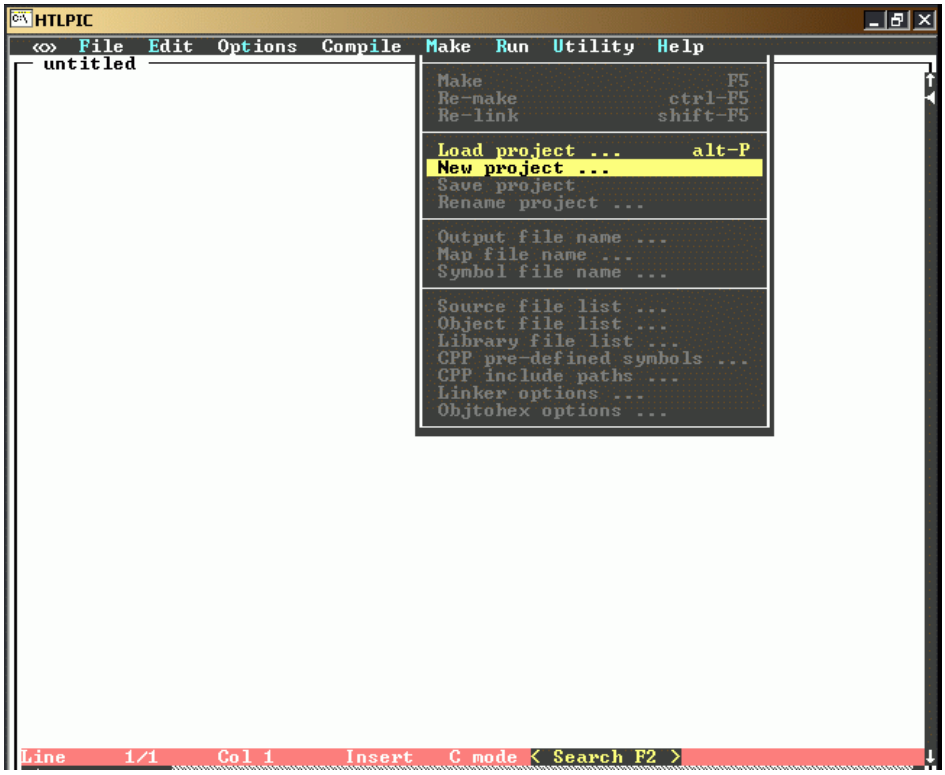
For starting up with USDP, we have made a design flow guide for using the **PIC C Lite** compiler, but for more information users can surf the help index of the compiler,

Install the **PIC C Lite** compiler from provided CD-ROM; you can use the evaluation version at start up, which supports the program code upto 2KB, which is sufficient for small development purposes.

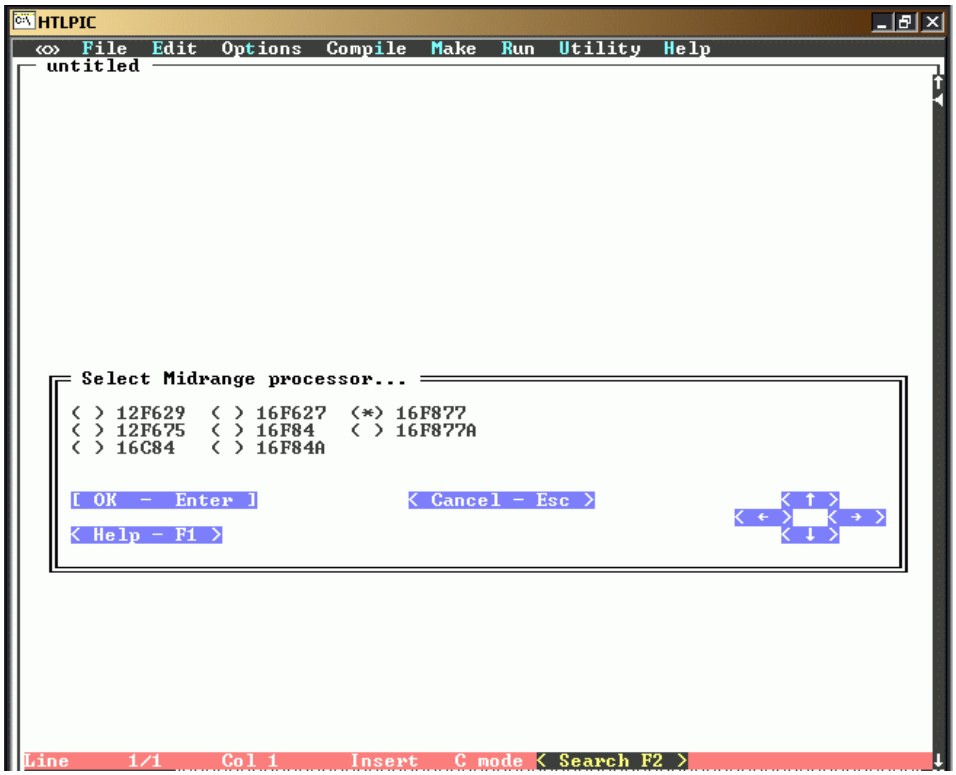
Step 1: Run the **PIC C Lite** compiler EXE, which in turn will open the compiler.



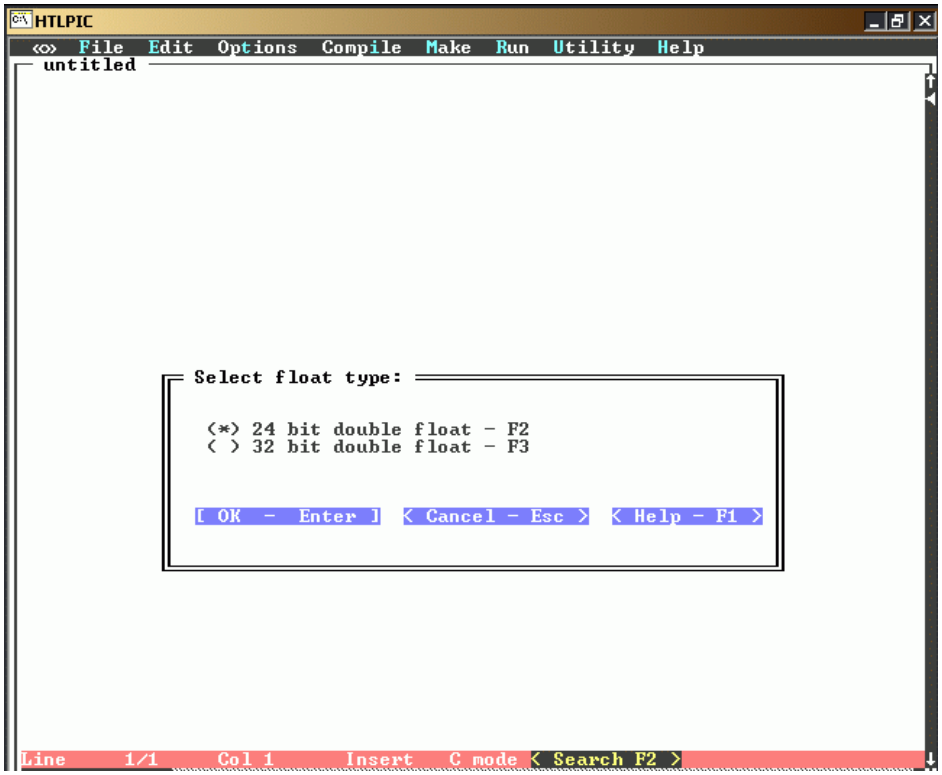
Step 2: To create new project goto **Make** menu, and select **New project** option.



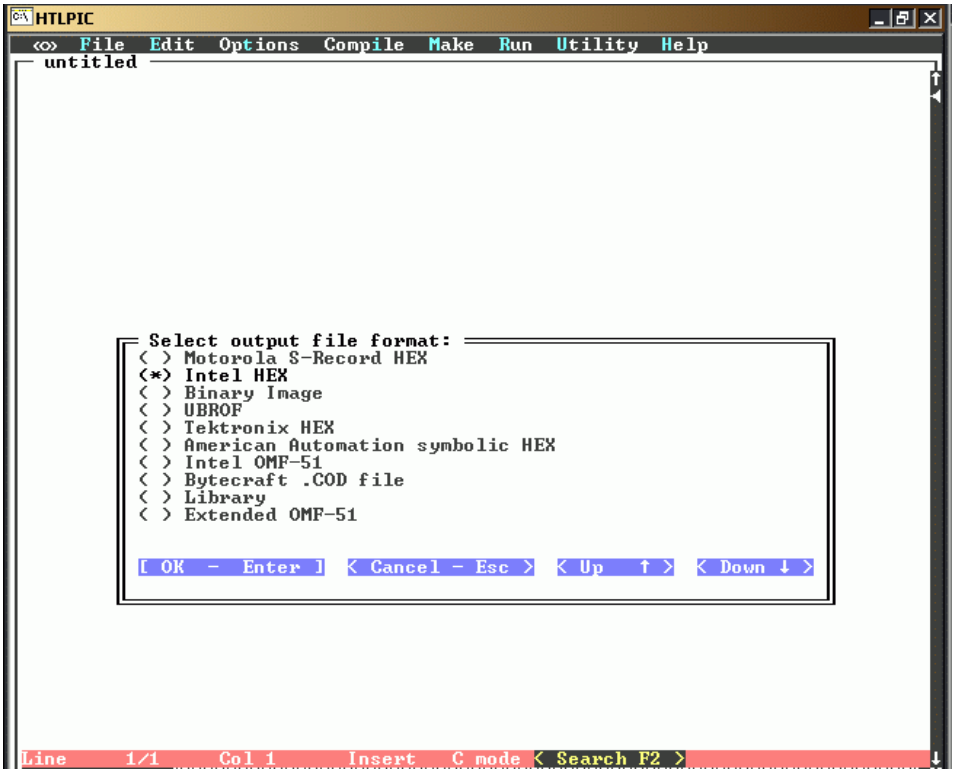
Step 3: After giving the project name and location, you need to select the device. The provided PIC module consists of **16F877** processor.



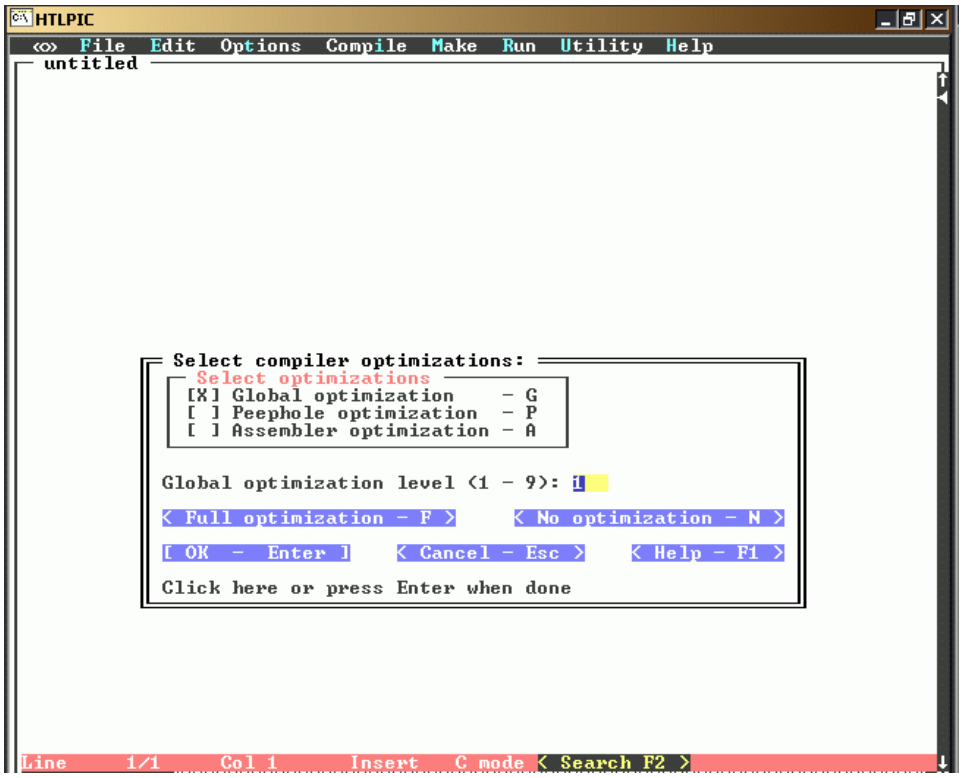
Step 4: Select the float type according to your design requirement, else select **24 bit double float file** format.



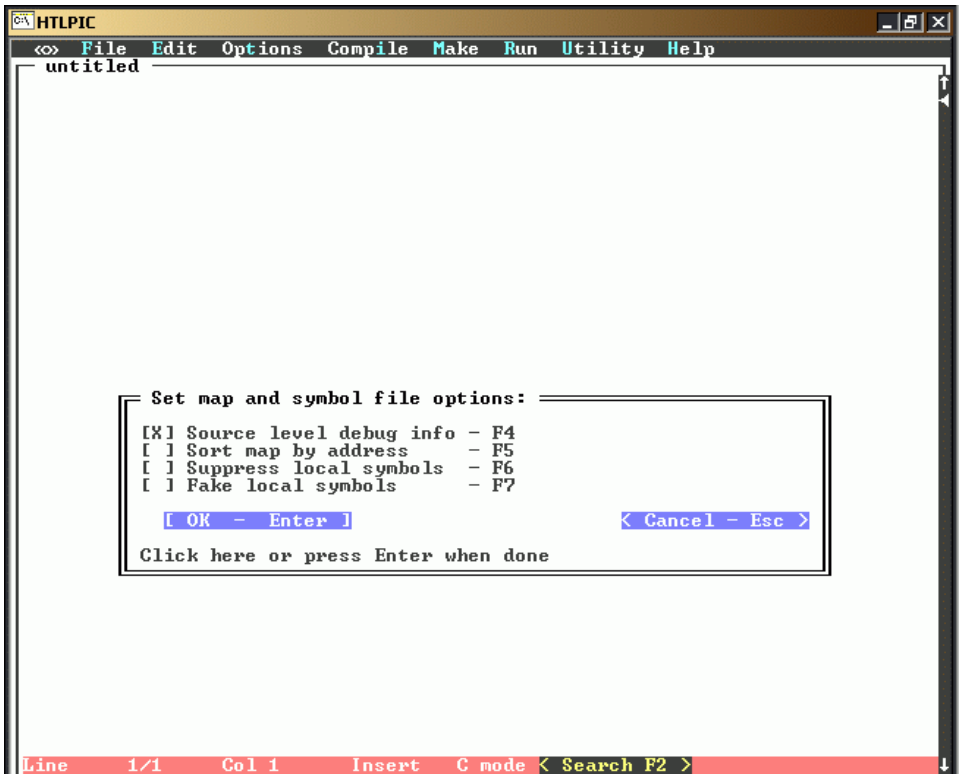
Step 5: In the next window select the output file format as **Intel HEX** file.



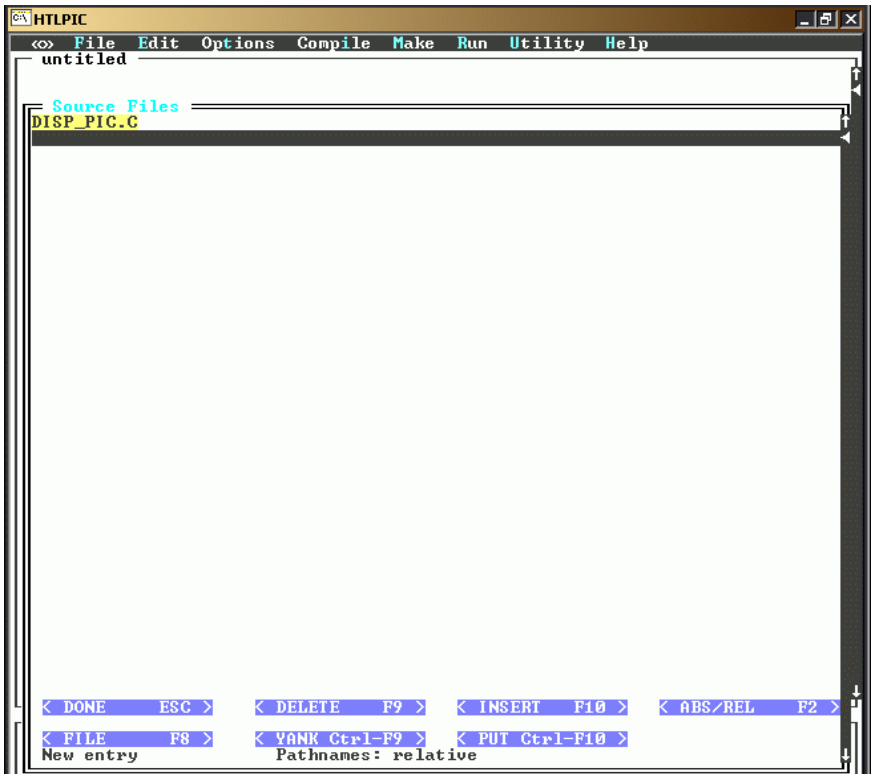
Step 6: Select the **compiler optimization** according to your requirement; by default you can select **Global Optimization.**



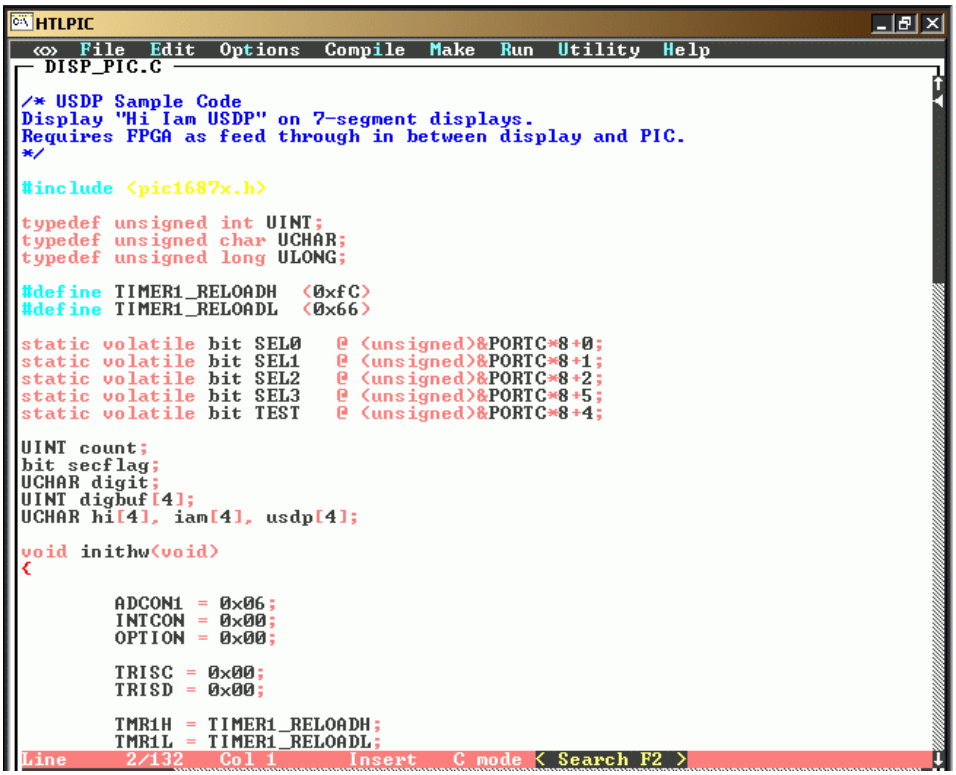
Step 7: Now user has to select **mapping** and **symbol file** option. User can select the options according to his requirement, or select **for source level debug info.**



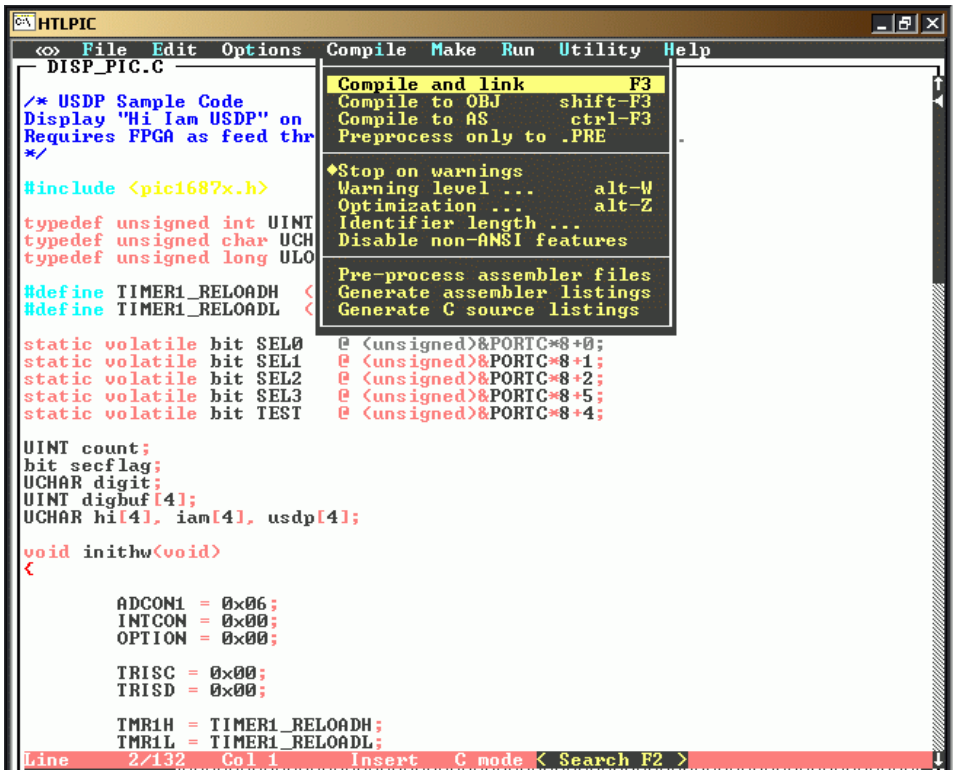
Step 8: Now add the source files in the project. Right now, add the provided **disp_pic.c** source file provided along with the USDP. Browse to the folder by pressing FILE option and add the said source file in the list.



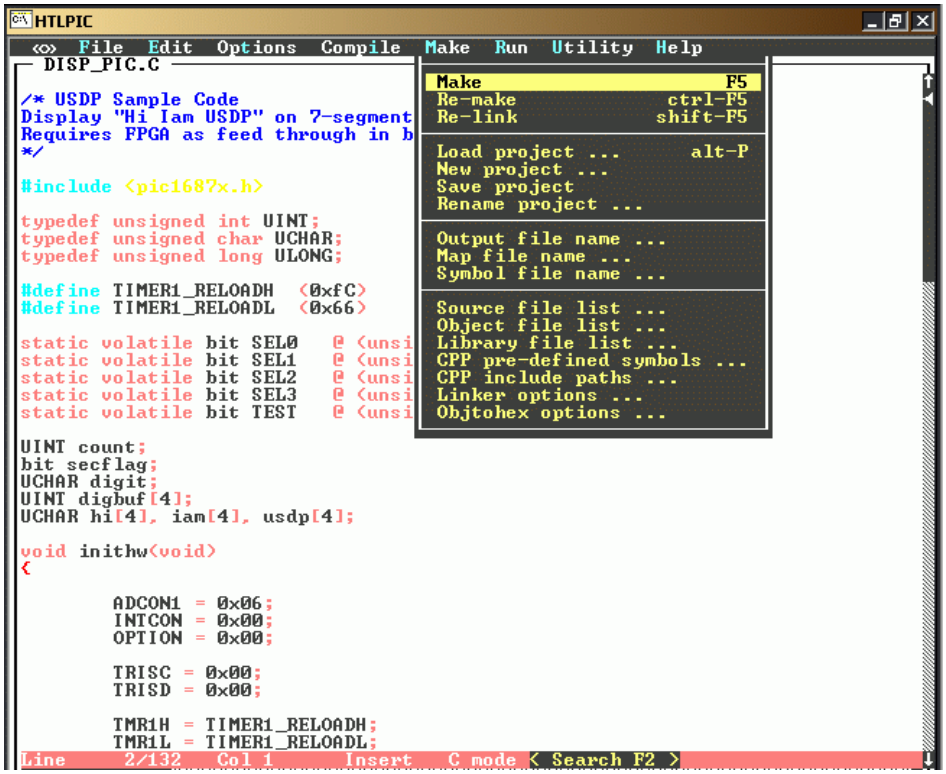
Step 9: Now the project has been created. User can edit the source file in the editor or check the code.



Step 10: The next step is to compile the source code. For compilation and linking the source code, go to **Compile** menu and select **Compile and Link option.**



Step 11: If the compilation process is over then the HEX will be generated in the project location, on the name of source code.. If user wants to run the **Make** process then they can go to **Make** menu and select Make option. This will create the HEX file on the project name.



Users can now use this HEX code to program the PIC controller using provided **PROGPIC** programming software.

Chapter 9
Configuration

This chapter covers the configuration process required for the programming/configuration of FPGAs and micro-controllers.
USDP supports multiple FPGA stacking with the PLD connectors (Slot1 and Slot2). This gives the flexibility for implementing multiple FPGA based designs, or users can also use FPGA independently.
Users can use the following combination of FPGAS.

Stacking Options		
	Slot 1	**Slot 2**
Choice 1	Xilinx	-
Choice 2	Altera	-
Choice 3	Xilinx	Xilinx
Choice 4	Altera	Altera
Choice 5	Xilinx	Altera

Note: Both the PLD slots share the bus, users has to take care about the pin mode while using the both slots together. No two pins should be output; else there may be chances of hardware short.

Till now the user is familiar with the Xilinx and Altera EDA tools for development of HDL based designs.
Now further more we will cover the necessary steps required to configure the devices.

Configuring Xilinx FPGAs

- Select the configuration mode and generate the programming file accordingly.
- Set the mode selection switch position on Xilinx module with reference to your selected programming mode.
- Check the PROM jumper setting for using or bypassing the PROM.
- Set the jumper selection for JTAG/serial mode.
- Inset the Xilinx module on any one of the PLD slots (Slot1 or Slot2).
- Connect the programming cable.
- Short the 1-2 connections of PLD sel header on the baseboard, to select the Xilinx programmer.
- Select the slot you want to program.
- Turn on the board supply.
- Run the programmer on the Xilinx ISE series software.

Note: For jumper/header setting refer chapter **Jumper Setting**, and for Xilinx ISE flow refer chapter EDA tools.

Configuring Altera FPGAs

- Select the configuration mode and generate the programming file accordingly.
- Set the mode selection jumpers on Altera module with reference to your selected programming mode.
- Check the PROM jumper setting for using or bypassing the PROM.
- Set the jumper selection for JTAG/serial mode.
- Inset the Altera module on any one of the PLD slots (Slot1 or Slot2).
- Connect the programming cable.
- Short the 2-3 connections of PLD sel header on the baseboard, to select the Altera programmer.
- Select the slot you want to program.
- Turn on the board supply.
- Run the programmer on the Altera Quartus series software.

Note: For jumper/header setting refer chapter jumper setting, and for Altera Quartus flow refer chapter EDA tools.

Slot selection

There are total five slots on the baseboard of USDP. The first two (Slot1 & Slot2) are PLD connector slots where user can insert provided FPGA modules. The remaining PCI based connector slots (Slot3, Slot4 & Slot5) are used for add-on module insertion.

FPGA module can be inserted in Slot1 & Slot2, but for programming the FPGAs, user has to select the Slot which he has used and the device vendor he intends to program; this can be done by using the selector cards provided along with the USDP board. The PLD selector card can be used to select the PLD vendor and the Slot selector card can be used to select the slot. Refer the table below for the combination.

PLD Selector	
Altera	Xilinx
1-2	2-3

Slot Selector	
Slot 1	Slot 2
1-2	2-3

Programming Modes

Both the PLD vendors Xilinx and Altera modules support various programming modes to download the configuration file in FPGA.

Here is the table for programming modes for Xilinx and Altera modules.

For Xilinx FPGA module

Modes	M0	M1	M2
JTAG	1	0	1
Slave Serial	1	1	1
Master Serial	0	0	0

The same table is also printed on the module, which the users can refer while programming. To apply logic high and low on the mode pins, users have to use the DIP switch provided. By turning ON, the switch will apply logic HIGH on the mode pin and by turning OFF, the switch position will put logic LOW on the mode pins.

Note: The fourth switch is Not Connected (NC) and is not in use.

For Altera FPGA Module

	MSEL1	MSEL0
Passive Serial (PS)	0	0
JTAG	0	1

Users can use the selection jumpers (J3 & J8) provided on module to select the configuration mode.
For logic LOW short 2-3 and for logic short 1-2 for logic HIGH.

Jumper setting for mode selection

Apart from mode selection pin settings, user also has to select the programming pins for programming.

For Xilinx FPGA module

Jumpers	**Serial**	**JTAG**
JP4, JP5, JP6, JP7	Short 1-2	Short 2-3

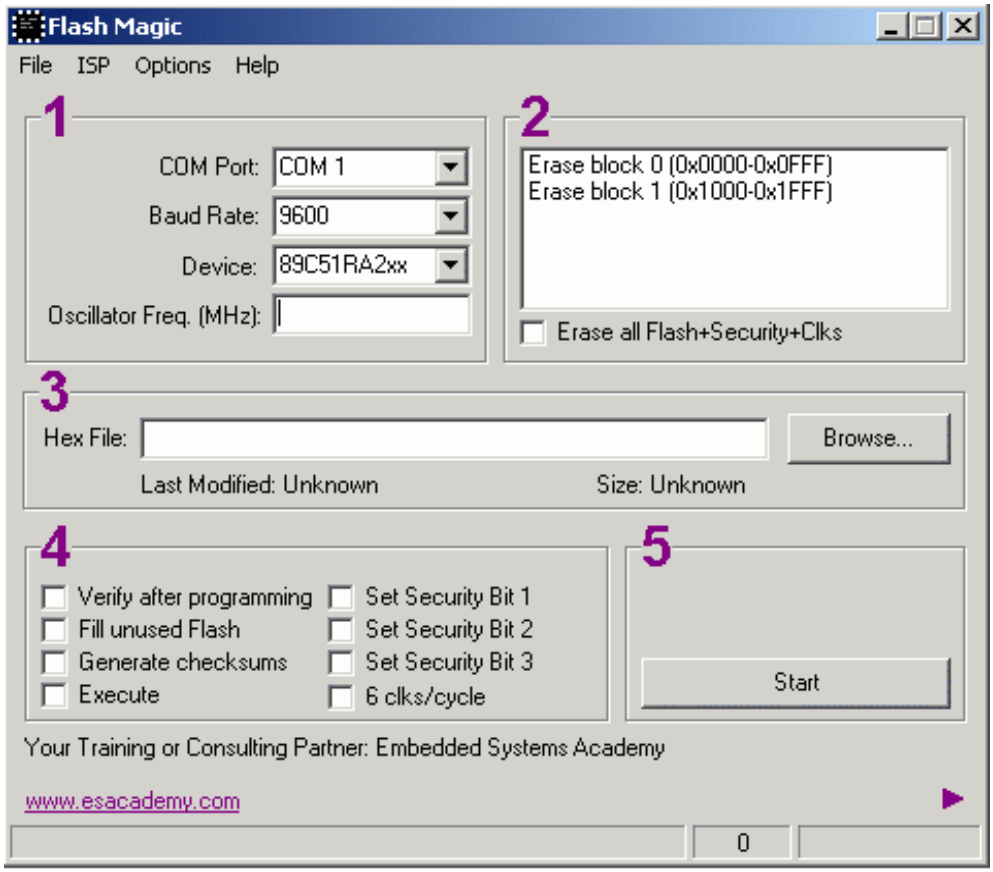
For Altera FPGA Module

Jumpers	**Serial**	**JTAG**
J4, J5, J6, J7	Short 2-3	Short 1-2

Programming 89c51RD2 microcontroller

Using the Flash Magic programmer, designers can program the 89c51 controller.
After building the HEX file from compiler, designers can refer the below shown steps to program the controller.

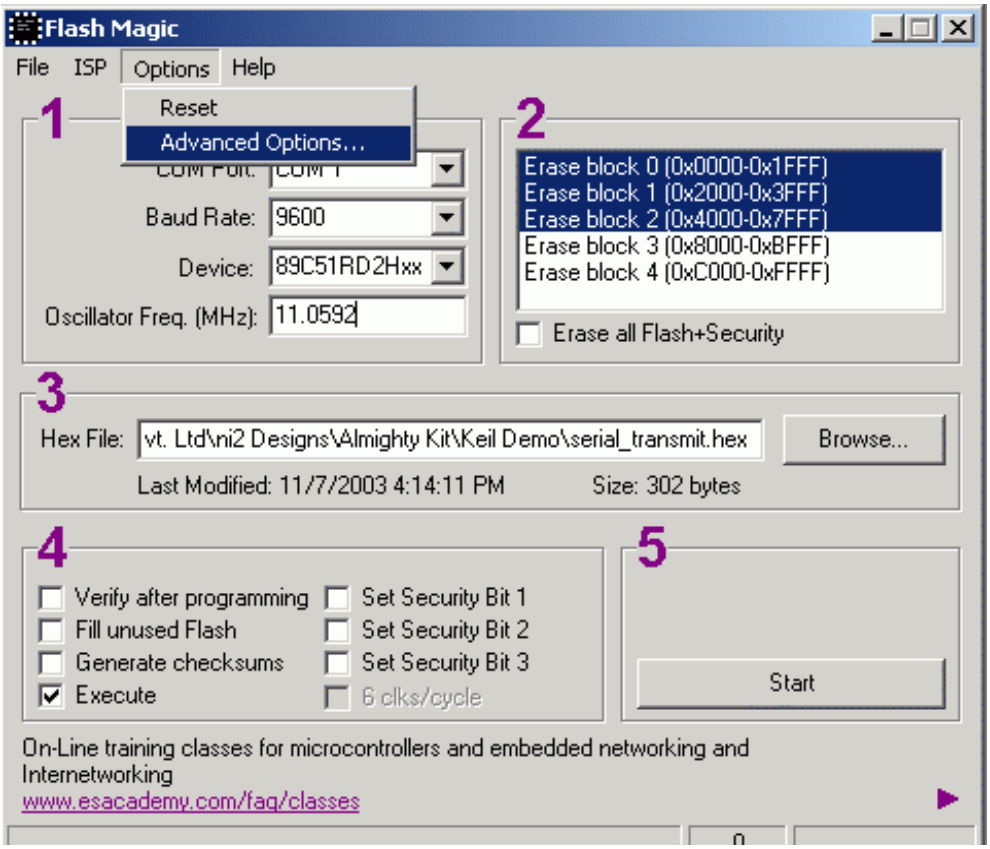
Insert the controller module on baseboard; connect the serial cable provided to module and PC's serial port. Turn ON the power supply; now run the EXE of Flash Magic programmer on your PC. The below shown window will open up.



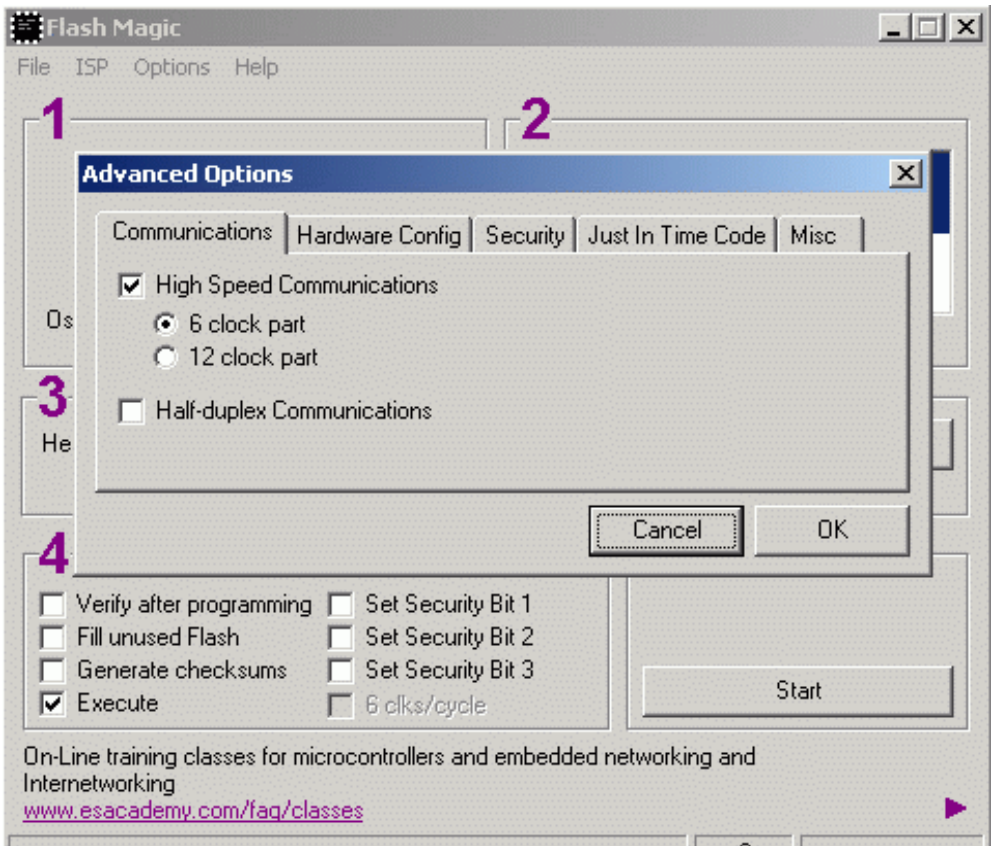
The above software is been divided in 5 sections.
1 - For device selection and settings.
2 - For program memory block erasure selection.
3 – Selection of HEX file.
4 – Programming options, and security settings.
5 – Starting programming.

So, one bye one we will look on all the sections and settings required to work on this particular controller module.

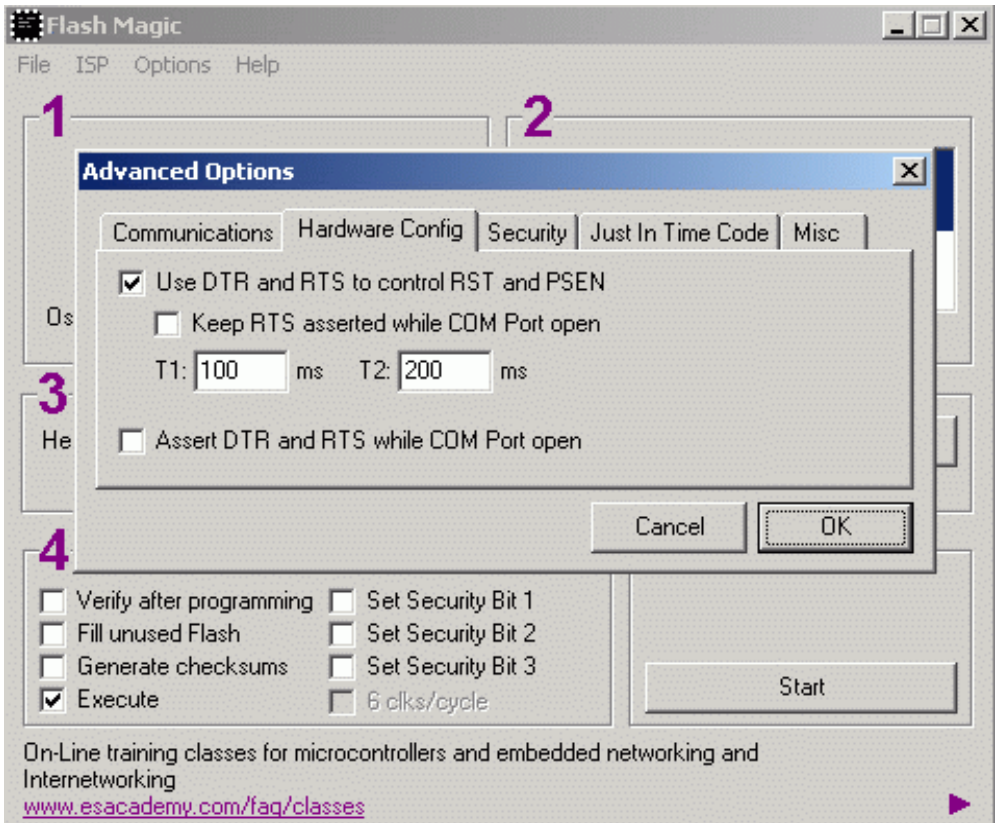
Step 1: Before the settings of all 5 sections, go to **option** menu and click **advance options.**



Step 2: In the opened window, go to **communications** tab, and set the option **High Speed Communication** and **6-clock part.**

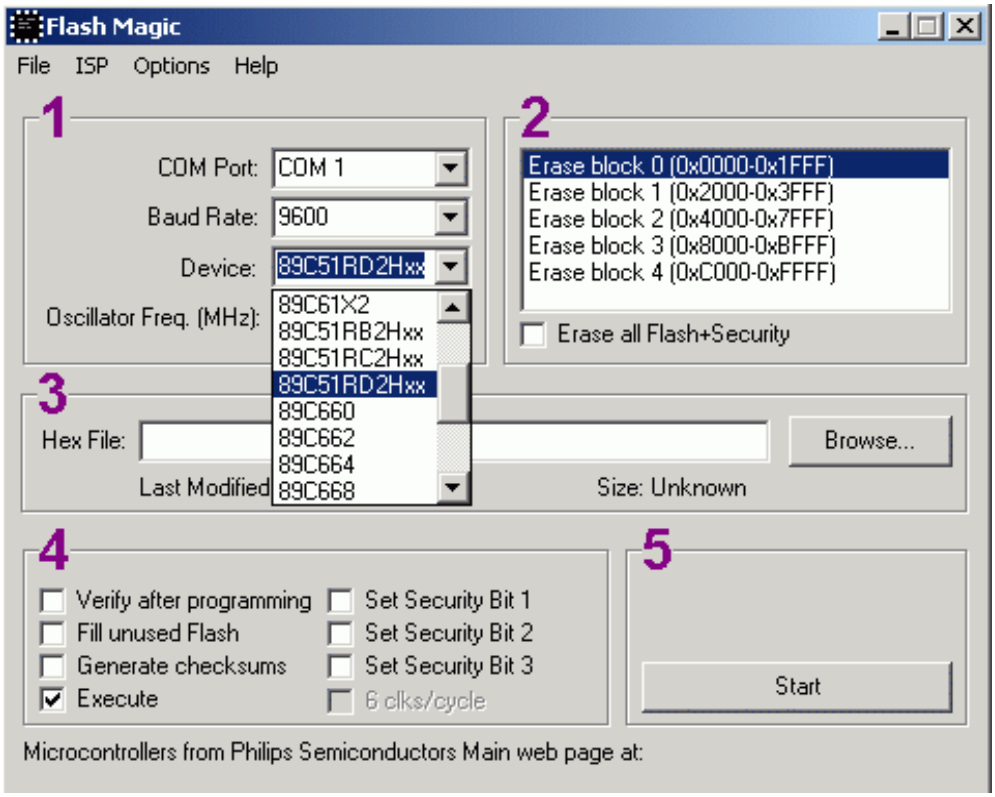


Step 3: Now go to **hardware config** tab, and set the option **Use DTR and RTS to control RST and PSEN.** Also set the **T1** value by **100 ms** and **T2** by **200 ms**. now click **OK** and go back to main window.

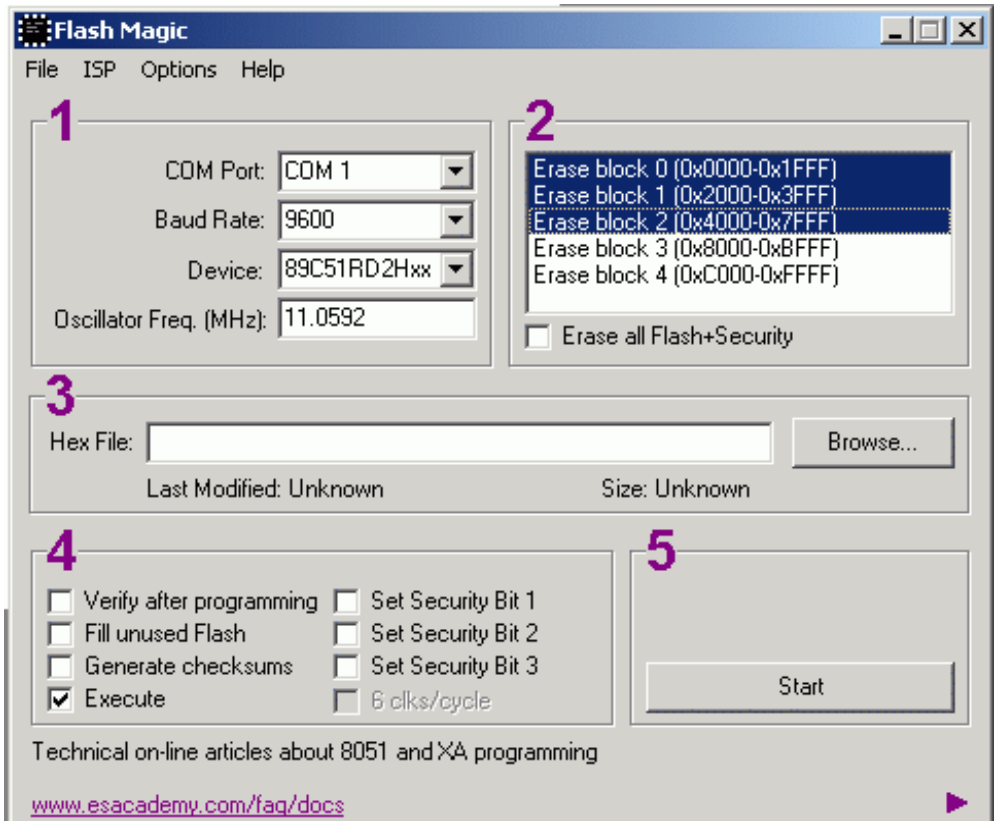


In case there is problem in connecting the device, then change the T1 & T2 values by 50 & 100 respectively.

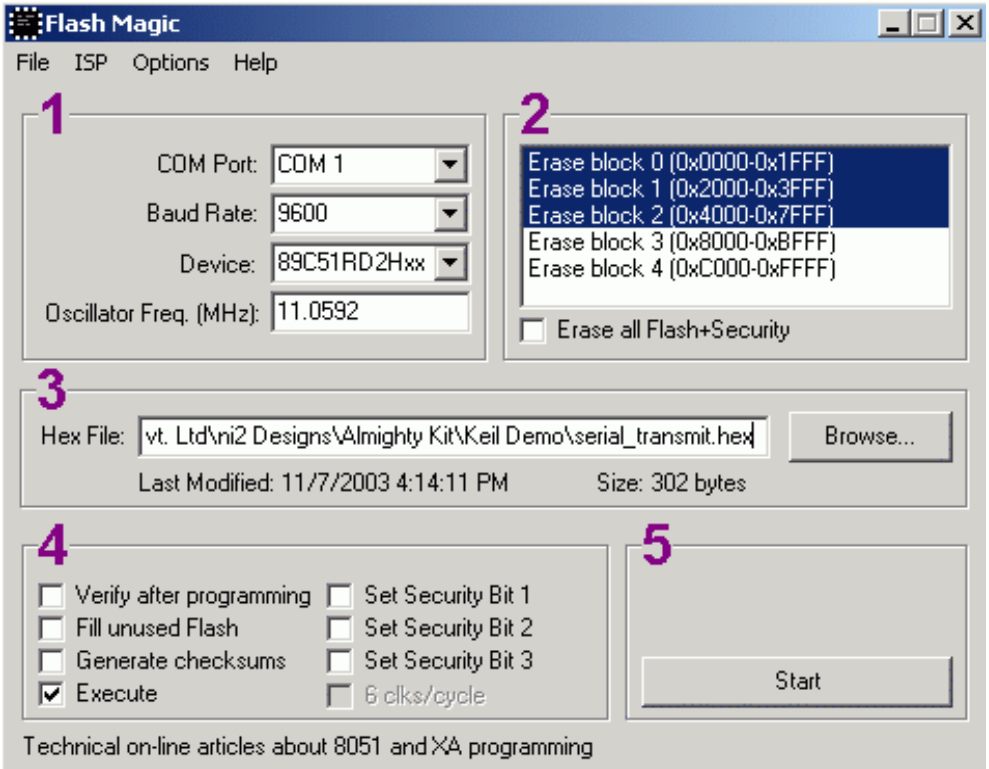
Step 4: In the main window, for the **1st** section, use the following settings.
COM port = **COM1**; Baud rate = **9600**; Device = **89C51RD2Hxx**; Oscillator Freq.(MHz) = **11.0592**.



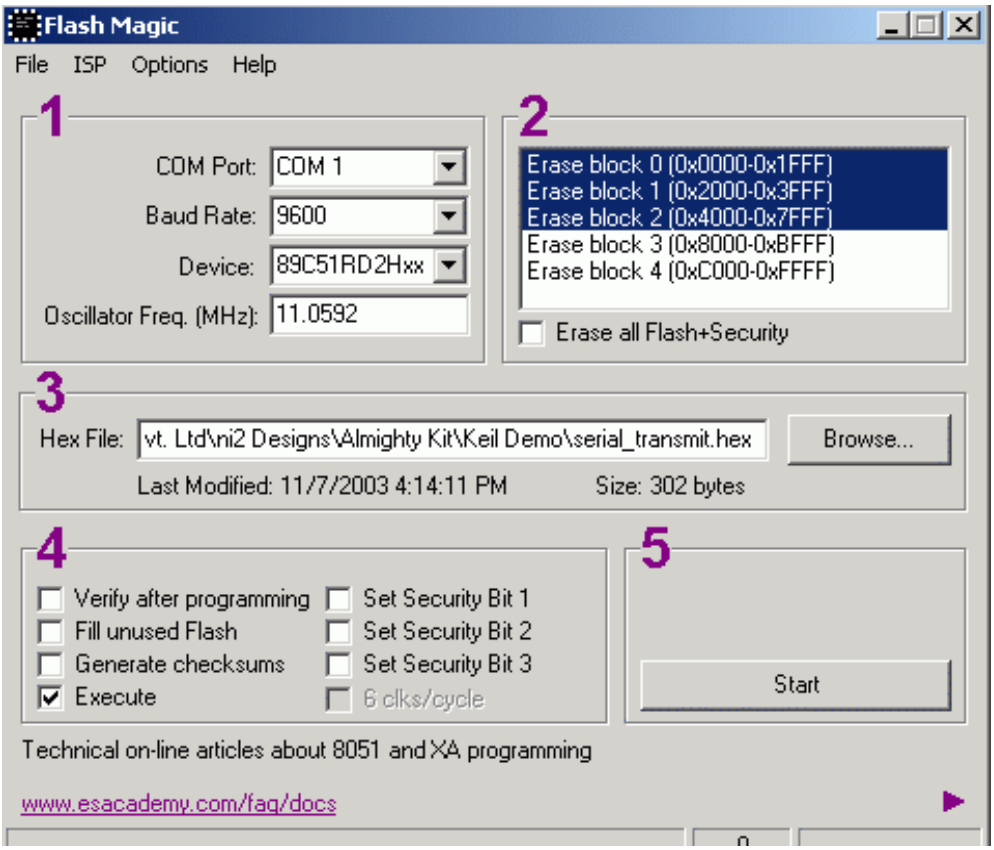
Step 5: In the **2nd** section, select the number of block to be erased; generally up till block 1 or 2 are needed for small code of program. These are code memory locks which needs to be erased to over write the new program on to them.



Step 6: In the **3rd** section, you have to give the programming **HEX** file; so browse to the folder containing the HEX file and click OK. You can browse for sample HEX file provided along with the USDP, **serial_transmit.hex.**



Step 7: In the **4th** section set the **Execute** option and clear all other options. This option is to execute the code after programming the controller.



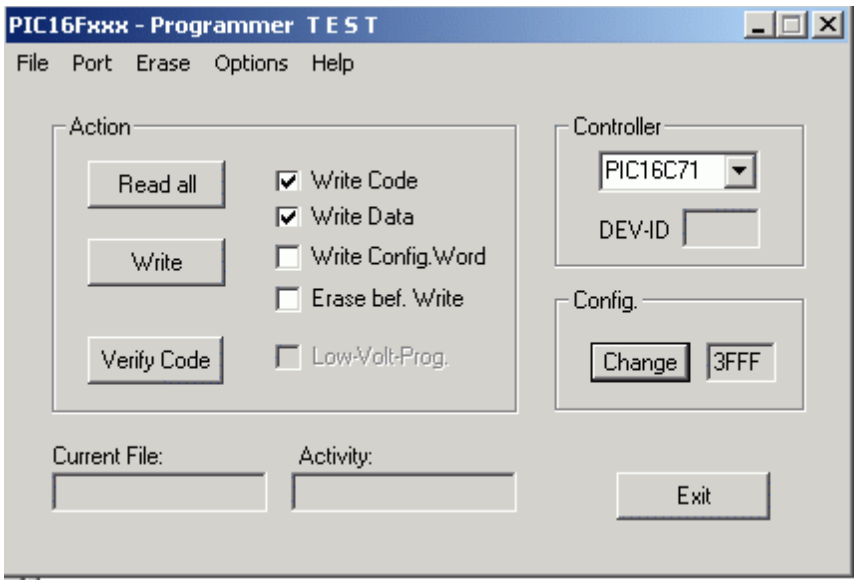
Step 8: Now in the 5th section press the **start** button, which in turn start the programming of controller. After displaying of **Finished** message on the bottom side of window, start working on your application.

Note: If any modification is there in the source code, then regenerate the HEX file and repeat steps 6 to 8.
The above program works on windows 98/2000/XP platforms.
The above sample program works in conjunction with FPGA, so after programming the 89c51 controller configure the FPGA with the sample code provided for it. For more information refer the chapter **Using Add-on Modules.**

Programming PIC16F877 controller

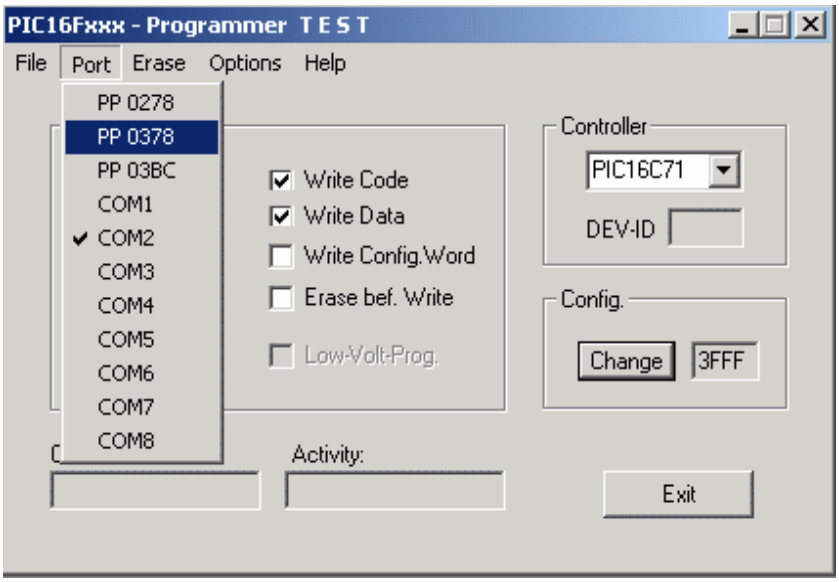
Using the **ProgPIC** programmer, designers can program the PIC16F877 controller module.
After building the HEX file from compiler, designers can refer the below shown steps to program the controller.

Insert the controller module on baseboard; connect the serial cable provided to module and PC's parallel port. Connect the +18V adaptor to Casio plug on the module. This adaptor is used to generate high voltage pulse to reset the PIC, remove this adaptor after programming the PIC controller.
Turn ON the power supply; now run the EXE of **ProgPIC** programmer on your PC. The below shown window will open up.

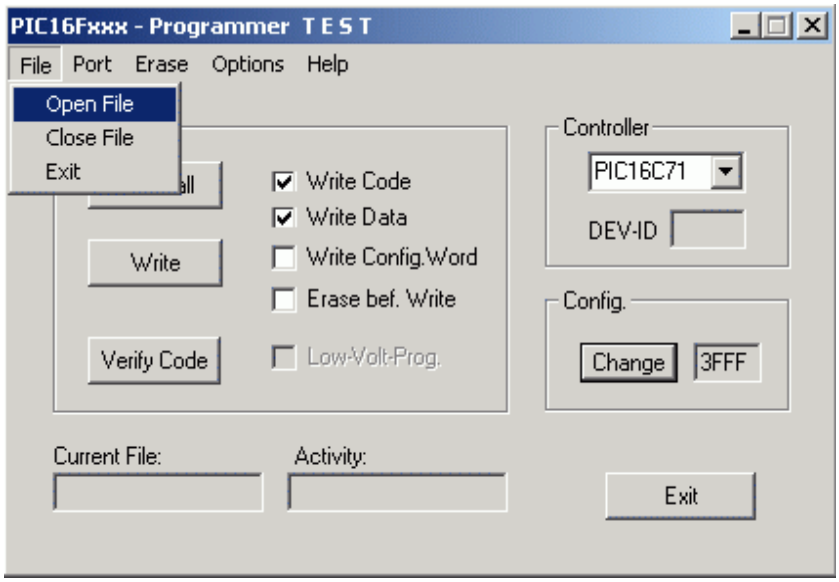


The designer needs to set the programming port, device, configuration word, and couple of option to use the above programmer, once looking at the pictorial steps below, it will become more easy to use the programmer.

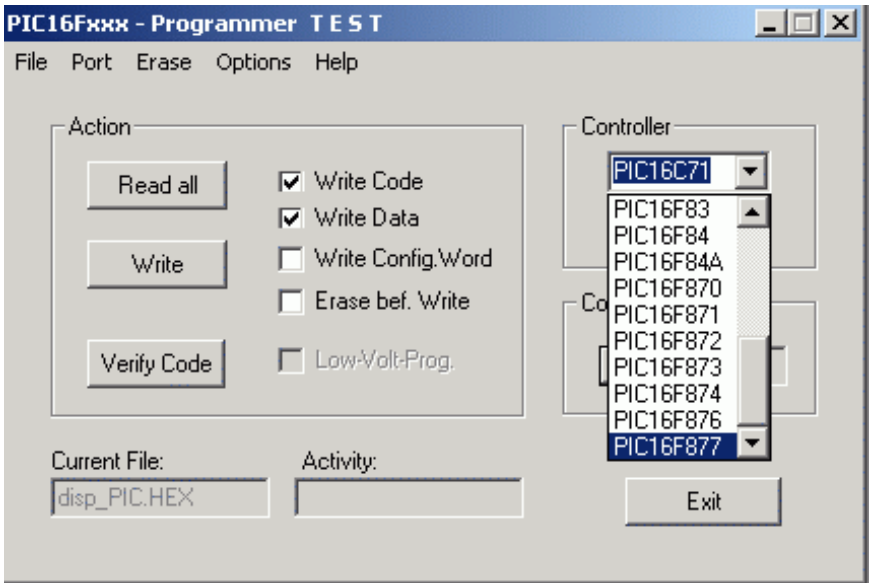
Step 1: Go to **Port** menu, and select the **PP 0378H** option to use the parallel port for programming.



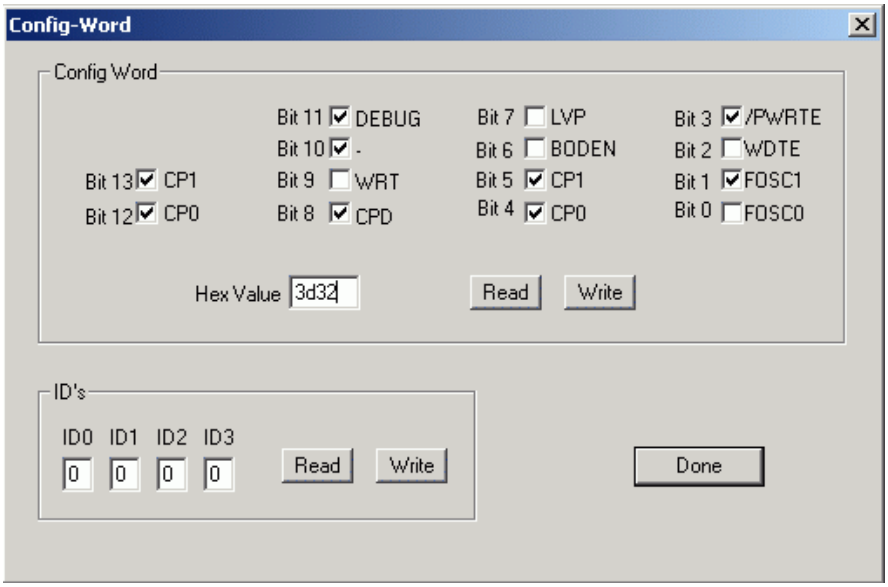
Step 2: Now go to File menu and click **open file** option, there after you have to browse to your programming HEX file. You can select the sample program provided along with the USDP, **disp_PIC.hex**



Step 3: Go to **Controller** option on main window, and select PIC part number as **16F877**.

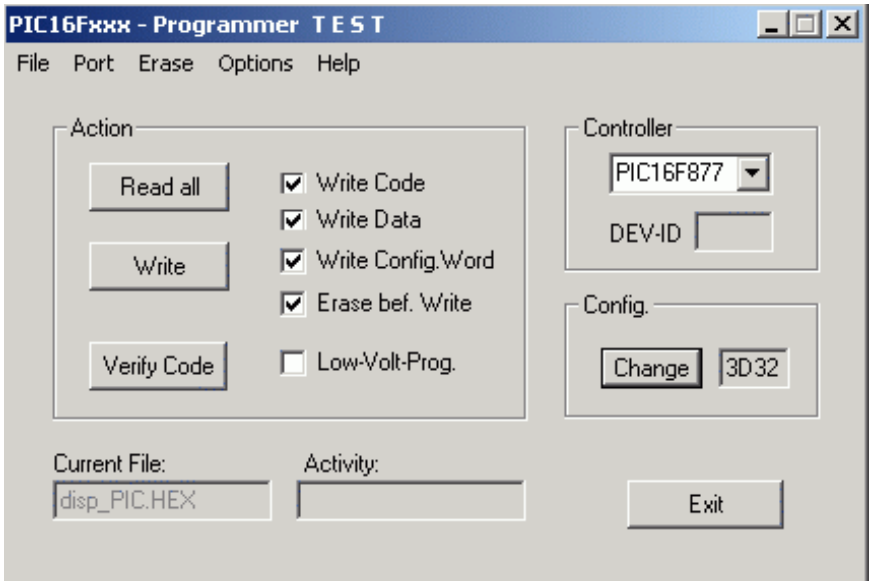


Step 4: The PIC controller works on the **configuration word** setting; the configuration word can be modified by clicking the change button on the **Config**. option. This will open a new window for setting the configuration word. For the sample program the configuration word value is **3D32 (in HEX),** which you can fill in the window and press **write** to write in the controller. User can also make changes according to their design for changing the configuration word.



Note: The configuration word can also be included in the source code.

Step 5: Press **Done** on the above window, and come back to main window. Here set the following option shown in the bottom picture. (Write code; write data; write config word; Erase before write).



Now the required settings are made to program the PIC microcontroller. Now press the **write** button to start programming the device. After display of **OK** on the activity window, start working on your application.

Note: The above program works on windows 98, to work on windows 2000/XP install the provided **DlportIO** driver.
The above sample program works in conjunction with FPGA, so after programming the PIC controller configure the FPGA with the sample code provided for it. For more information refer the chapter **Using Add-on Modules.**

Chapter 10
Using Add-on Modules

In this chapter user will cover the basics on using the add-on modules provided along with the USDP.
The motive of chapter is to let the user know about the module, executions steps, block diagrams and details for using the modules.

Here is the list of modules to be covered under this chapter;

1. USDP Base Board
2. Xilinx FPGA Module.
3. Altera FPGA Module.
4. ADC/DAC Module.
5. 89c51RD2 Module.
6. PIC uC Module.
7. SRAM Memory Card.
8. Power Electronics Module.
9. General Purpose PCB.

1. USDP Base Board

USDP baseboard is used to interface the various modules with each other. Designers can use this baseboard for developing their application with switches, -segment displays, keypad and connectors provided.

Jumper and Header settings

Clock (JP1)

Pin 1	Clock
Pin 2	GCK0
Pin 3	Ground

To use the oscillator clock short pin 1 and 2. It is recommended to ground the GCK0 during programming.

JPDIS

These four jumpers are used to disable the 7-segment displays. Just remove the jumpers to isolate to displays from FPGAs.

JPA, JPB, JPC & JPD

These jumpers are used to select the I/O mode of the LEDs.
Total 32 configurable I/Os are provided on the USDP baseboard. To configure them as I/P user has to insert the jumper in between, and remove the jumper to make them as O/P.

Power Headers (J8 & J9)

There are power supply headers; users can connect the SMPS to any one of these headers. Two headers are provided just for the sake of convenience for connecting the SMPS from any of the directions.

SLOT SEL (JP3)

This is slot selector header. User can use the provided selector card for selecting the slot.

Slot 1	**Slot 2**
Short 1-2	Short 2-3

PLD SEL (JP2)

This is PLD vendor selector header. User can use the provided selector card for selecting the PLD vendors.

Xilinx	**Altera**
Short 1-2	Short 2-3

J7 (Keypad)

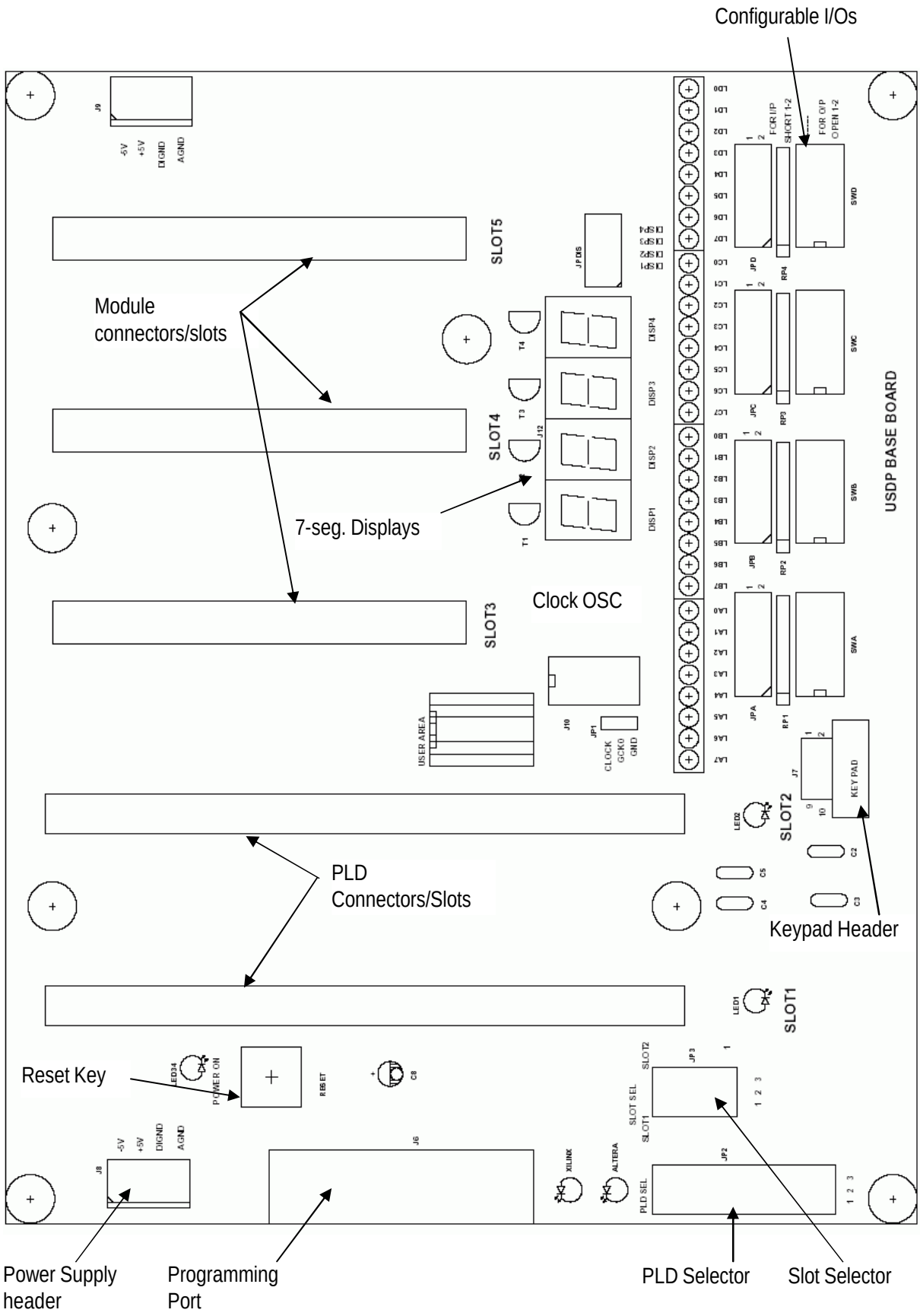
Pin No.	**Signal**	**Pin No.**	**Signal**
1	VCC	2	GND
3	SL3	4	SL2
5	SL1	6	SL0
7	RL3	8	RL2
9	RL1	10	RL0

Programming Port (J6)

User has to connect provided programming cable to this port and PC's parallel for programming of FPGAs.

Board Idents

In this section, idents of all boards are shown; users can refer this section for any component listing.



USDP Base Board Ident

2. Xilinx FPGA Module

The Xilinx FPGA module contains a Spartan-II family FPGA with a choice of 50,000 gates or 200,000 gates FPGA. With a total pins 208 and maximum 147 user I/O pins. Users can develop designs requiring high density and higher number of I/Os with this FPGA.
There are total 96 I/O pins brought down to the base bus for add-on module connection. User can use this bus for their application design with the add-on module. Also 32 bits out this bus are been shared with configurable I/Os.
The bank (0) is been brought on DB-25 connector with its VREF pins. Designers can use this bank for different voltage signaling. Also this DB-25 connector can be used to interface with the PC's parallel port for communication.

We have provided a sample program, which implements the hardware of 3:8 decoder IC 74xx238, which is active high decoder with 3 enable signals.
Here is the procedure for using the above code with this FPGA module.

Procedure for using module

- Write source code in **VHDL/Verilog** using the Xilinx ISE series software.
- Pin lock the entity I/Os with configurable I/Os.
- Run the FPGA design flow, from synthesis to implementation of design on FPGA.
- Set the programming mode **JTAG** or **slave serial** in **generate programming file** option.
- Generate the BIT file.
- Insert the FPGA module on slot 1 or slot 2.
- Set the PROM bypass jumpers to position 2-3.
- Set the programming mode in **JTAG** or **slave serial.**
- Set the jumper position of configurable I/Os for input or output for entity pins.
- Connect the programming cable.
- Turn on the **USDP** power supply.
- Run the programming tool **impact**.
- Program the FPGA.
- Check the application.

Source code for FPGA

Source code name = **decoder.vhd**
Pin lock file name = **decoder.ucf**

Note: For further information on Xilinx Spartan-II FPGAs, kindly refer the device datasheet.

Jumper Settings

Programming Mode selection Switch (U7)

Modes	M0	M1	M2
JTAG	1	0	1
Slave Serial	1	1	1
Master Serial	0	0	0

Mode selection jumpers (JP4, JP5, JP6, JP7)

Serial	**JTAG**
Short 1-2	Short 2-3

PROM BYPASS (J3 & J4)

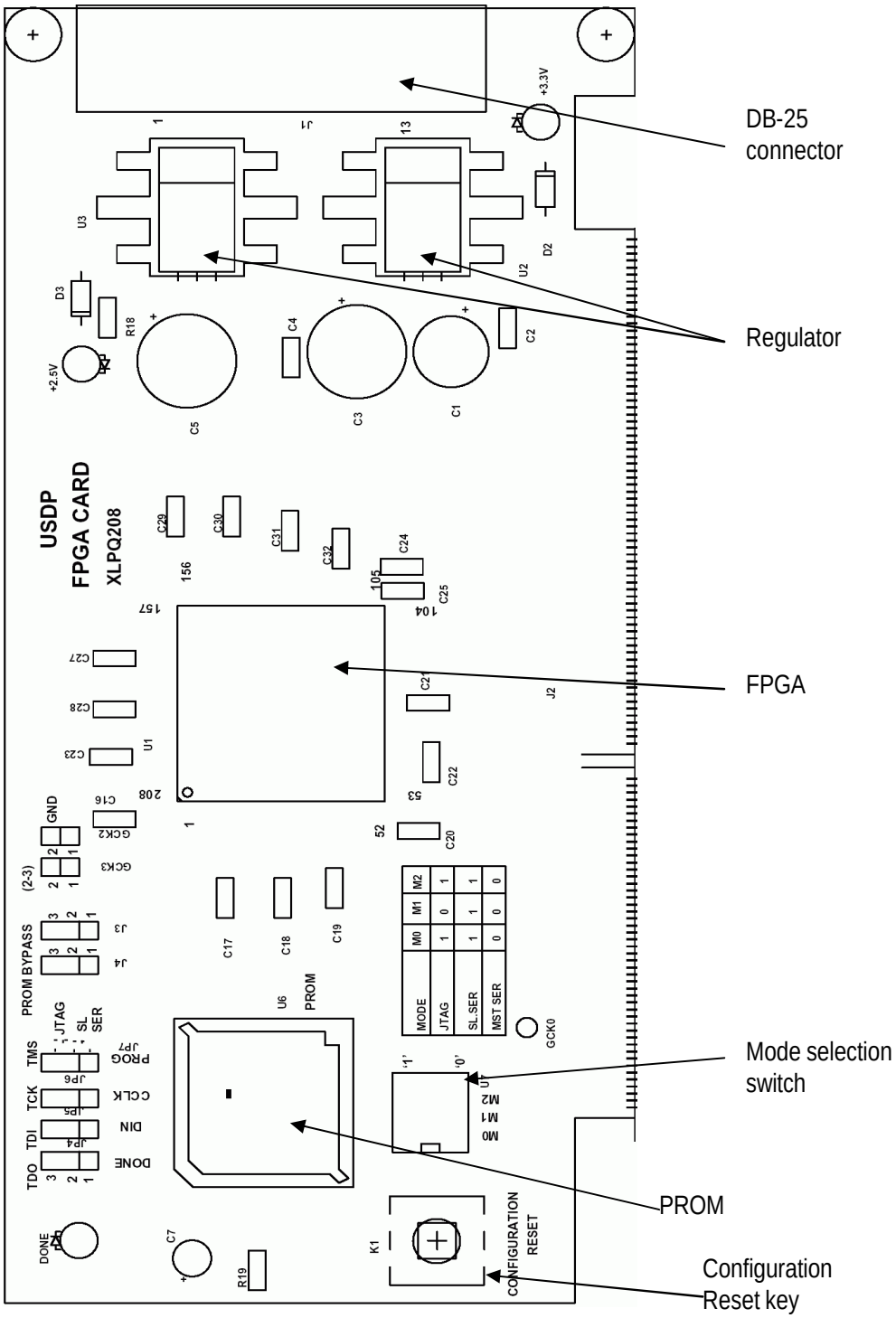
To Chain PROM	To Bypass PROM
Short 1 –2	Short 2-3

Shorting 1-2 will bring the PROM in chain with FPGA. In this case user can configure the PROM and use for programming file storage.
Shorting 2-3 will remove the PROM from chain, and only FPGA would be connected to programming port.

Global Clock Buffers

Pin 1	GCK2	GCK3
Pin 2	GND	GND

Note: These buffers are extra from GCK0.



Xilinx FPGA Adaptor Board Ident

Spartan-II FPGA Pin detail

Device: **XC2Sxx PQ208**

Clock and Reset	
Clock (GCK0)	80
GCK2	182
GCK3	185
Reset	5

Note: **Reset** is active **LOW.**

Configurable I/Os							
LA7	34	**LB7**	45	**LC7**	60	**LD7**	71
LA6	35	**LB6**	46	**LC6**	61	**LD6**	73
LA5	36	**LB5**	47	**LC5**	62	**LD5**	74
LA4	37	**LB4**	48	**LC4**	63	**LD4**	75
LA3	41	**LB3**	49	**LC3**	67	**LD3**	81
LA2	42	**LB2**	57	**LC2**	68	**LD2**	82
LA1	43	**LB1**	58	**LC1**	69	**LD1**	83
LA0	44	**LB0**	59	**LC0**	70	**LD0**	84

7 Segments		Display Enable		Keypad Header			
segA	33	**DISP1**	17	**SL0**	8	**RL0**	6
segB	31	**DISP2**	18	**SL1**	15	**RL1**	10
segC	30	**DISP3**	20	**SL2**	9	**RL2**	7
segD	29	**DISP4**	21	**SL3**	16	**RL3**	14
segE	27						
segF	24						
segG	23						
segDP	22						

Parallel Port Connector (DB-25)			
Par1	206	**Par10**	192
Par2	205	**Par11**	191
Par3	204	**Par12**	188
Par4	202	**Par13**	187
Par5	201	**Par14**	203
Par6	199	**Par15**	200
Par7	195	**Par16**	189
Par8	194	**Par17**	181
Par9	193		
Pin 18 – 25 of DB-25 connector are ground			

Note: Both the FPGAs share the above I/Os. User has to take care, that **no two pins are defined as output**, as in that case there would be short on the bus and may damage FPGA I/Os.

User can define the following combination of FPGAs shared I/Os;

FPGA1	**FPGA2**	
Input	Input	Allowed
Output	Input	Allowed
Input	Output	Allowed
Output	**Output**	**Not allowed**

3. Altera FPGA Module

The Altera FPGA module contains an ACEX 1K family FPGA with a choice of 50,000 gates or 100,000 gates FPGA. With a total pins 208 and maximum 147 user I/O pins, users can develop designs requiring high density and higher number of I/Os.
There are total 96 I/O pins brought down to the base bus for add-on module connection. User can use this bus for their application design with the add-on module. Also 32 bits out this bus are been shared with configurable I/Os.
Few I/Os are also brought on DB-25 connector. Designers can use this connector to interface with the PC's parallel port for communication.
We have provided a sample program, which implements the hardware of 3:8 decoder IC 74xx238, which is active high decoder with 3 enable signals.
Here is the procedure for using the above code with this FPGA module.

Procedure for using module

- Write source code in **VHDL/Verilog** using the Altera Quartus series software.
- Pin lock the entity I/Os with configurable I/Os.
- Run the FPGA design flow, from synthesis to implementation of design on FPGA.
- Generate the BIT file for JTAG or serial mode.
- Insert the FPGA module on slot 1 or slot 2.
- Set the programming mode in **JTAG** or **slave serial.**
- Set the PROM bypass jumpers to position 2-3.
- Set the jumper position of configurable I/Os for input or output for entity pins.
- Connect the programming cable.
- Turn on the **USDP** power supply.
- Program the FPGA.
- Check the application.

Source code for FPGA

Source code name = **decoder.vhd**
Pin lock file name = **decoder.ucf**

Note: For further information on Altera ACEX 1K FPGAs, kindly refer the device datasheet.

Jumper Settings

Programming Mode selection jumpers (J3, J8)

	MSEL1	MSEL0
Passive Serial (PS)	0	0
JTAG	0	1

For logic low short 2-3 and for logic short 1-2 for logic High

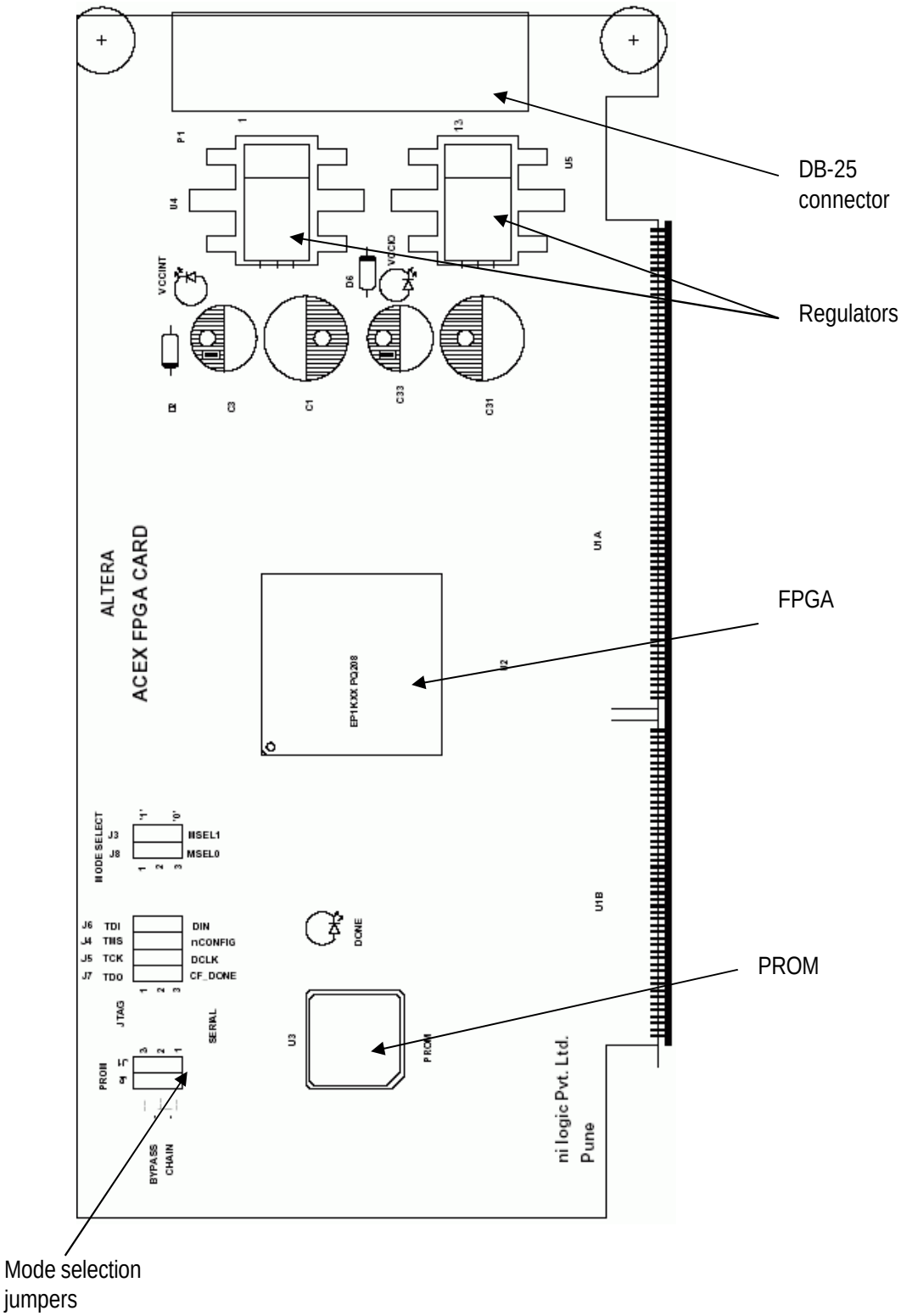
Mode selection jumpers (J4, J5, J6, J7)

Serial	JTAG
Short 2-3	Short 1-2

PROM BYPASS (J1 & J2)

To Chain PROM	To Bypass PROM
Short 1 –2	Short 2-3

Shorting 1-2 will bring the PROM in chain with FPGA. In this case user can configure the PROM and use for programming file storage.
Shorting 2-3 will remove the PROM from chain, and only FPGA would be connected to programming port.



ACEX FPGA Adaptor Board Ident

ACEX1K FPGA Pin detail

Device: **EP1kxx PQ208**

Clock & Reset	
Clock	79
Reset	180

Note: **Reset** is active **LOW.**

Configurable I/Os							
LA7	70	**LB7**	60	**LC7**	46	**LD7**	36
LA6	71	**LB6**	61	**LC6**	47	**LD6**	37
LA5	73	**LB5**	63	**LC5**	53	**LD5**	38
LA4	74	**LB4**	64	**LC4**	54	**LD4**	39
LA3	75	**LB3**	65	**LC3**	55	**LD3**	40
LA2	85	**LB2**	67	**LC2**	56	**LD2**	41
LA1	86	**LB1**	68	**LC1**	57	**LD1**	44
LA0	87	**LB0**	69	**LC0**	58	**LD0**	45

7 Segments		Display Enable		Keypad Header			
segA	31	**DISP1**	15	**SL0**	8	**RL0**	3
segB	30	**DISP2**	16	**SL1**	13	**RL1**	11
segC	29	**DISP3**	17	**SL2**	9	**RL2**	7
segD	28	**DISP4**	18	**SL3**	14	**RL3**	12
segE	27						
segF	26						
segG	25						
segDP	24						

Parallel Port Connector (DB-25)			
Par1	205	**Par10**	193
Par2	203	**Par11**	192
Par3	202	**Par12**	191
Par4	200	**Par13**	190
Par5	199	**Par14**	189
Par6	198	**Par15**	187
Par7	197	**Par16**	186
Par8	196	**Par17**	179
Par9	195		
Pin 18 – 25 of DB-25 connector are ground			

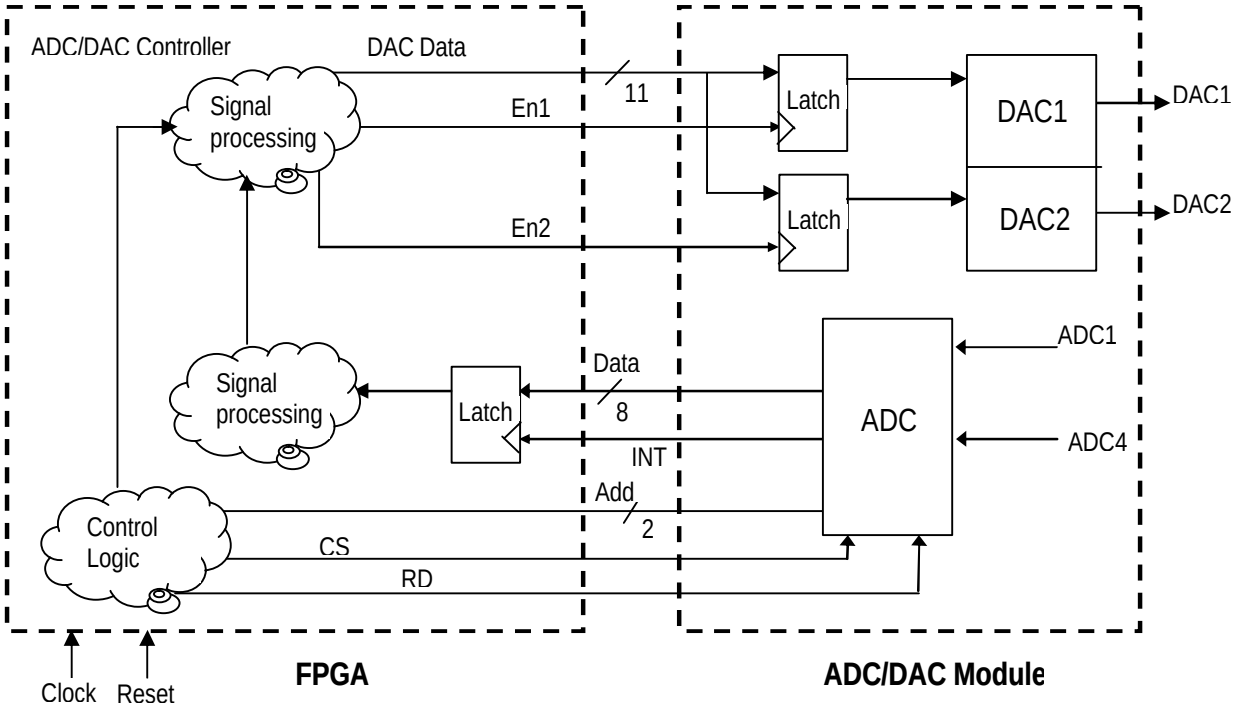
Note: Both the FPGAs share the above I/Os. User has to take care, that **no two pins are defined as output**, as in that case there would be short on the bus and may damage FPGA I/Os.

User can define the following combination of FPGAs shared I/Os;

FPGA1	FPGA2	
Input	Input	Allowed
Output	Input	Allowed
Input	Output	Allowed
Output	**Output**	**Not allowed**

4. ADC/DAC Module

The ADC/DAC module provided along with the USDP has 4 channel ADC, 2 channel DAC on board. This module is directly connected to the FPGAs, and does not have direct connections with other modules.
To use this module user has to design ADC/DAC controller in the FPGA to get the data from ADC and to put data on DAC channels.



Block diagram of the ADC

The above diagram shows how the interconnection can be done with the ADC/DAC module. The sampled data from ADC would be latched inside the FPGA, and as the module has on board latches for DAC, designer can control these latches and store data over there for DAC conversion.
Designer can use the sampled value from ADC for his signal processing and after that he can reconstruct the wave by storing the data in the on board latches for DAC.

We have provided a sample controller for ADC/DAC module that works as feed though circuit between ADC and DAC. User can use this source code for the check of module.

Procedure for using module

- Add the provided source code Xilinx ISE or Altera Quartus series software.
- Run the procedure for using FPGA module (refer previous pages).
- Insert the ADC/DAC module in Slot 3, Slot 4 or Slot 5.
- Connect the signal generator or analog source to ADC's channel 0.
- Connect the CRO probe to DAC's channel 0.
- Turn on the **USDP** power supply.
- Program the FPGA.
- Check the application.

Source code for FPGA

Source code name = **ADC_DAC_feed.vhd**
Pin lock file name = **ADC_DAC_feed.ucf**

For further working of ADC and DAC, kindly refer the datasheet provided along with the protoboard.

ADC/DAC Module Pin Detail*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A17	D0	92	87	**A18**	D1	90	86
A19	D4	96	94	**A20**	D5	95	90
A21	D8	101	98	**A22**	D9	100	97
A23	EN1	111	102	**A24**	EN2	104	101
A25	DB0	115	111	**A26**	DB1	114	110
A27	DB4	121	115	**A28**	DB5	120	114
A29	INT	127	122	**A30**	RDY	126	121

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B17	D2	89	3	**B18**	D3	88	4
B19	D6	94	89	**B20**	D7	93	88
B21	D10	99	96	**B22**	D11	97	95
B23	A0	103	100	**B24**	A1	102	99
B25	DB2	113	109	**B26**	DB3	112	108
B27	DB6	119	113	**B28**	DB7	116	112
B29	CS	125	120	**B30**	RD	122	119

Header Details

DAC O/P (J1)

Pin1	DAC Channel 1
Pin 2	DAC Channel 2
Pin 3	Ground

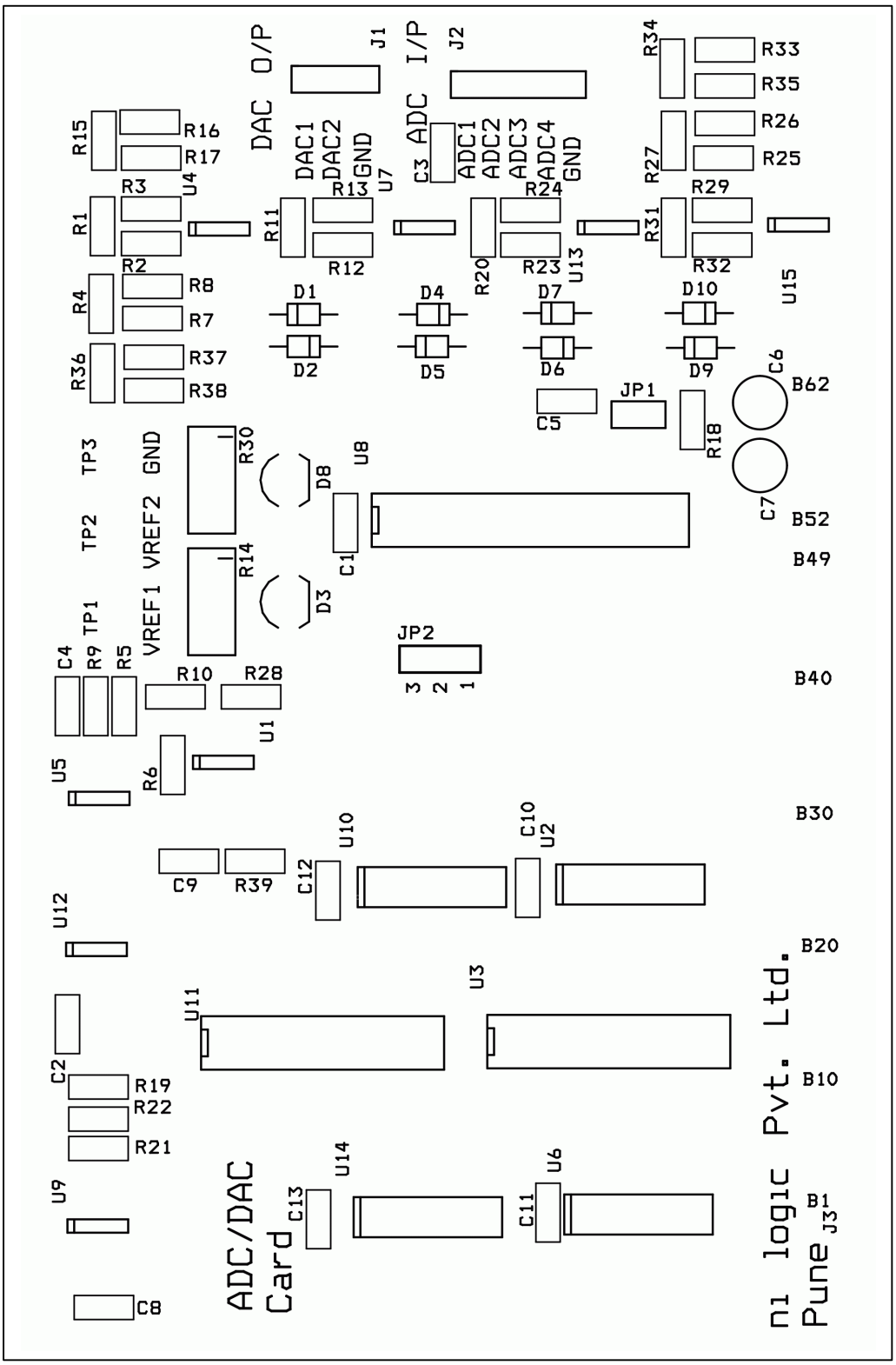
ADC I/P (J2)

Pin1	ADC Channel 1
Pin 2	ADC Channel 2
Pin 3	ADC Channel 3
Pin 4	ADC Channel 4
Pin 5	Ground

JP2

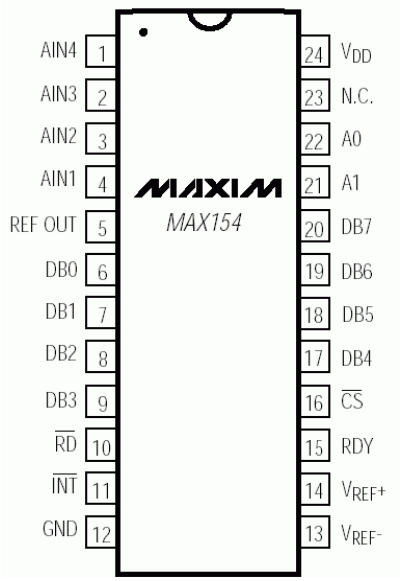
This jumper is for setting the reference voltage. This module has onboard reference voltage generation, to use that short 2-3.

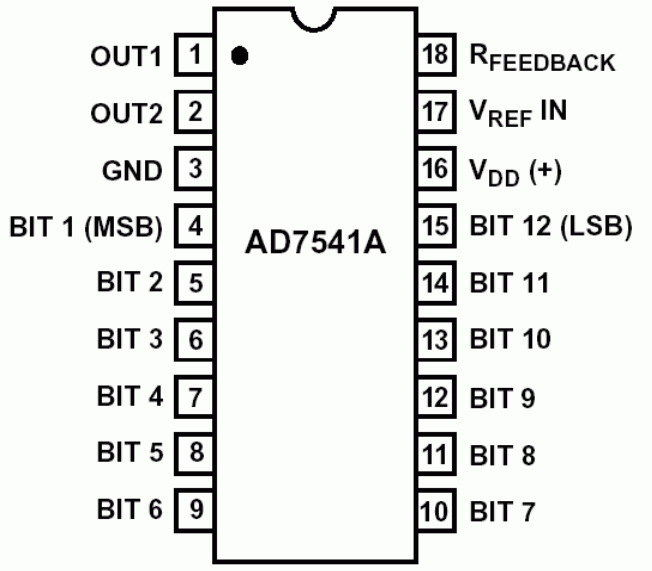
Note: ADC/DAC work in linear monotonic format with –VREF = 00000000 and +VREF= 11111111. So take care of numbering system while designing the logic.



ADC/DAC Card Board Ident

MAX154 ADC Pin out





ADC7541A DAC Pin out

5. 89c51RD2 Module

This module is based on renowned industry standard 8051 architecture based controller. The uC is 89c51RD2 controller from Philips, which is **In System Programmable (ISP).**
All the I/Os have been brought on the edge connector and are connected with the FPGAs, user can program the uC and use it with the FPGA. The timer & interrupt pins are brought on the separate headers, which can be used to interface with the external world.
The module also contains on board RTC and EEPROM, which user can use for his application.

Procedure for using module

- Write source code in **C** or **assembly** language in **keil** compiler or some other compiler.
- Generate the HEX file.
- Connect the serial cable.
- Turn on the **USDP** power supply.
- Program the microcontroller using the **Flash Magic** programmer.
- Configure the FPGA for microcontroller interface logic.
- Reset the controller through FPGA.
- Check the application.

We have provided a sample program, which takes data from FPGA on one port and sends it serially through the RS-232 port.
For further information on 89c51RD2, kindly refer the datasheet provided along with the protoboard.

Source code for FPGA
Source code name = **feed_89c51.vhd**
Pin lock file name = **feed_89c51.ucf**

Source code for 89c51
Source code name = **serial_transmit.c**
Programming file name = **serial_transmit.hex**

Note: For further information on using **Keil** compiler and **Flash Magic** software, kindly refer the chapters **Using EDA tools** & **Configuration**.

89c51 Module Pin Detail*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A31	AD0	133	127	**A32**	AD2	132	126
A33	AD4	139	134	**A34**	AD6	136	133
A35	A8	143	139	**A36**	A10	142	138
A37	A12	149	146	**A38**	A14	148	142
A39	PORT1_1	159	150	**A40**	PORT1_3	158	149
A41	PORT1_5	163	163	**A42**	PORT1_7	162	162
A43	INT0	168	167	**A44**	INT1	167	166
A45	ALE	173	174	**A46**	FRST	172	173

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B31	AD1	131	125	**B32**	AD3	128	123
B33	AD5	135	132	**B34**	AD7	134	129
B35	A9	141	136	**B36**	A11	140	135
B37	A13	147	141	**B38**	A15	144	140
B39	PORT1_2	157	148	**B40**	PORT1_4	150	147
B41	PORT1_6	161	152	**B42**	PORT1_8	160	151
B43	T0	166	165	**B44**	T1	164	164
B45	RD_51	170	172	**B46**	WR_51	169	168

Jumper & Header settings

Square Wave o/p (JP3)

Pin1	1Hz clock from RTC.
Pin 2	Ground

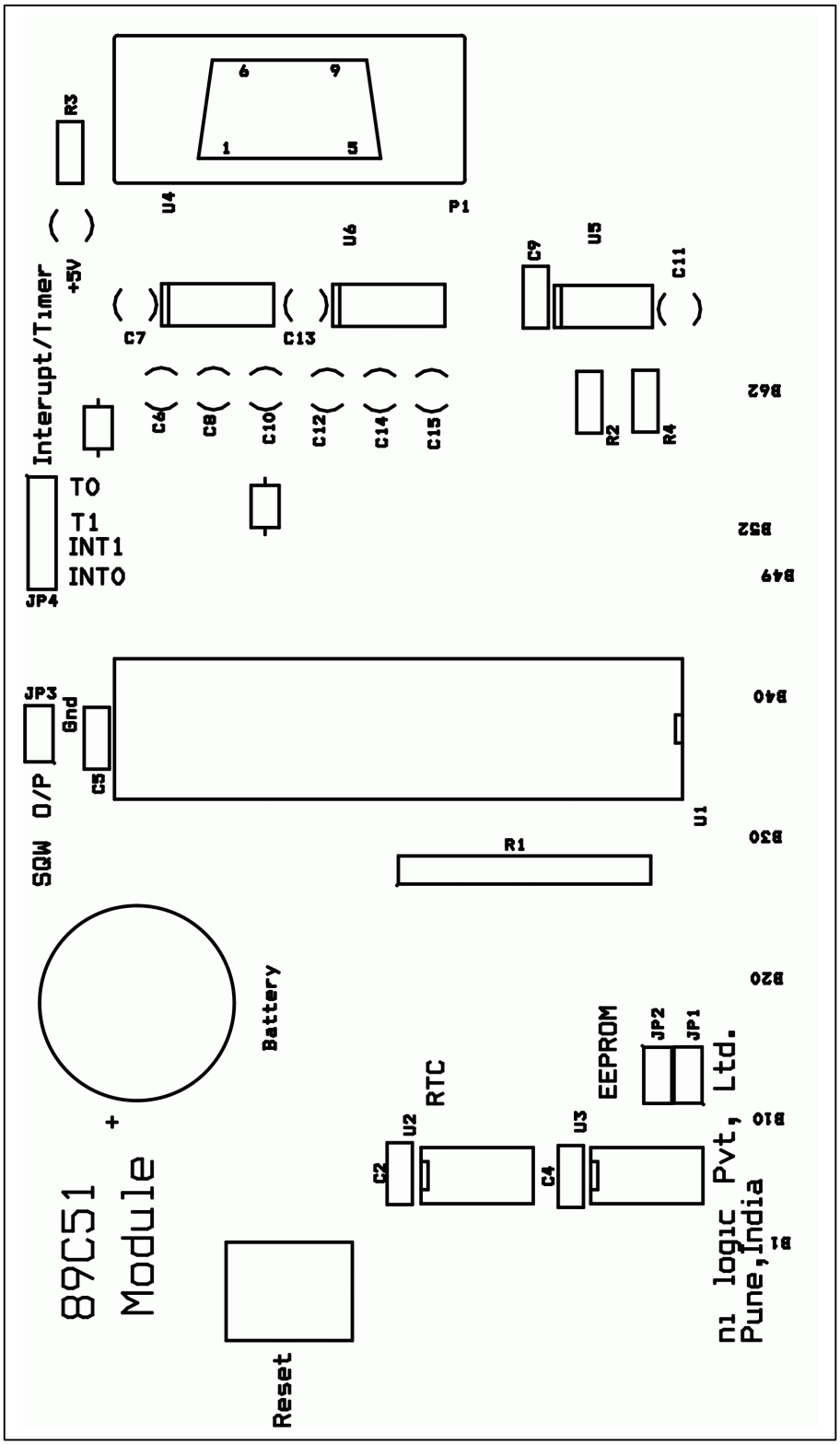
Interrupt/Timer (JP4)

Pin1	Interrupt 0 (INT0) I/P
Pin 2	Interrupt 1 (INT1) I/P
Pin 3	Timer 1 O/P
Pin 4	Timer 0 O/P

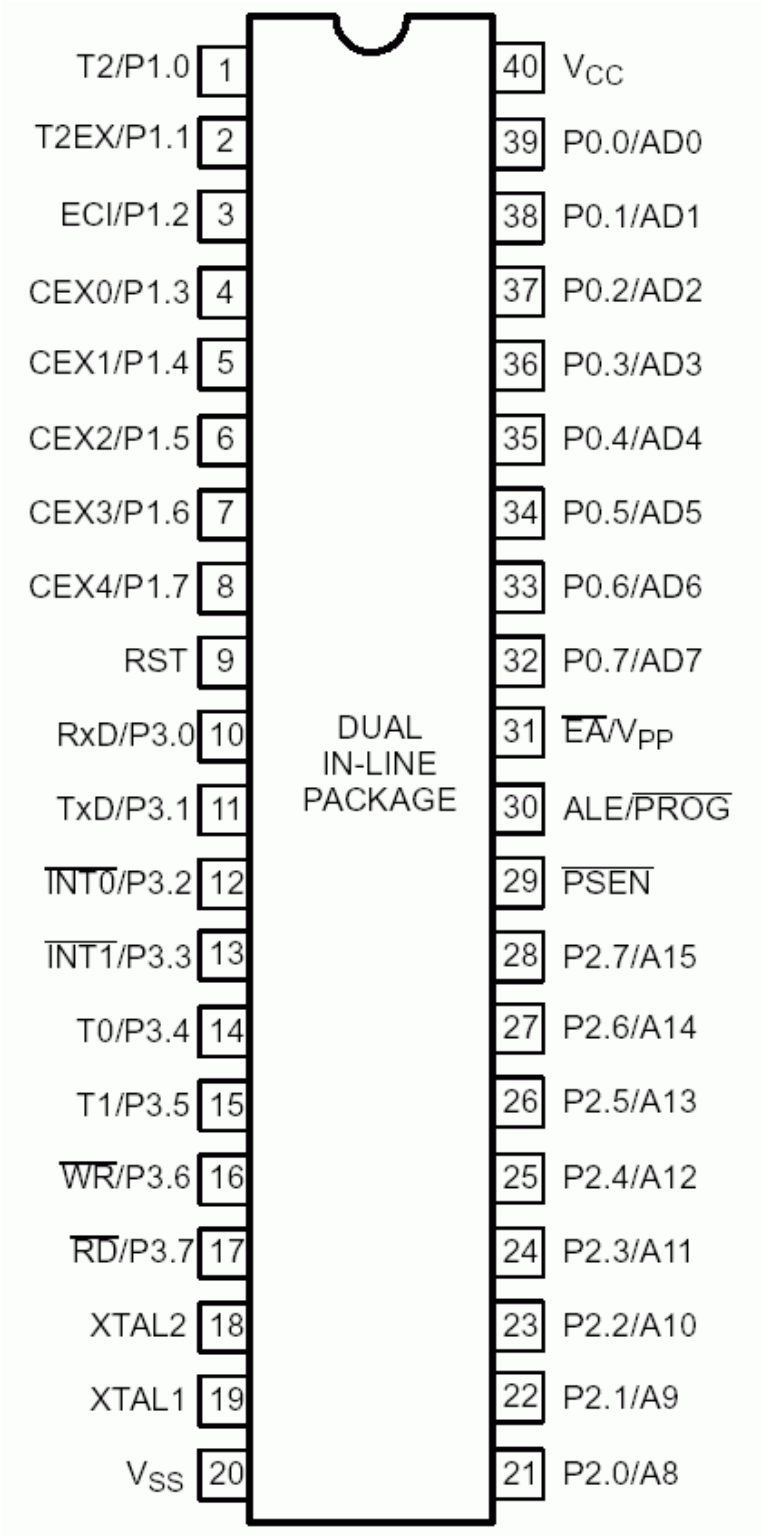
JP1 & JP2

JP1	Port1_1	SDA
JP2	Port1_2	SCLK

JP1 & JP2 are selection jumper for serial clock and data from RTC and EEPROM. Both these signals are shared with the base bus of USDP. So if you are these two ports of 89c51 with the base bus then remove the jumpers from the JP1 & JP2 to isolate them from the base bus.



89c51 Microcontroller Card Board Ident



89c51RD2Hxx Pin out

6. PIC uC Module

This module use RISC based architecture controller. The PIC16F877 controller is from Micro Chip, which is **In System Programmable (ISP).** This is leading controller used in industries for product design and is widely used for his ease of use and enriched features.
All the I/Os of controller have been brought on the edge connector and are connected with the FPGAs; user can program the uC and use it with the FPGA. The interrupt pins are brought on the separate headers, which can be used to interface with the external world.
The module also contains on board RTC which user can use for his application.

Procedure for using module

- Write source code in **C** or **assembly** language in compiler.
- Generate the HEX file.
- Connect the PIC parallel cable.
- Turn on the **USDP** power supply.
- Program the microcontroller using the **PIC PRO** programmer provided (also use the +18V adaptor).
- Remove the +18V adaptor after programming the PIC.
- Configure the FPGA for microcontroller interface logic.
- Reset the controller through FPGA.
- Check the application.

We have provided a sample program, which runs a message on the 7-segment display though the FPGA. All the control is from PIC controller and the FPGA is working like a feed through circuit.

For further information on **PIC16F877**, kindly refer the datasheet provided along with the protoboard.

Source code for FPGA
Source code name = **feed_PIC.vhd**
Pin lock file name = **feed_PIC.ucf**

Source code for PIC
Source code name = **disp_pic.c**
Programming file name = **disp_pic.hex**

Note: For further information on using **compiler** and **PIC PRO** software, kindly refer the chapters **Using EDA tools** & **Configuration**.

PIC 16F877 Micro controller Pin details*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A1	AN0/RA0	36	34	**A2**	AN1/RA1	37	35
A3	AN2/RA2	40	41	**A4**	AN3/RA3	41	42
A5	AN4/RA4	46	45	**A6**	AN4/RA5	47	46
A7	RD0	55	49	**A8**	RD1	56	57
A9	RD2	60	60	**A10**	RD3	61	61
A11	RD4	65	67	**A12**	RD5	67	68
A13	RD6	70	71	**A14**	RD7	71	73
A15	MCLR/	75	81				

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B1	RC0	38	36	**B2**	RC1	39	37
B3	RC2	44	43	**B4**	RC3	45	44
B5	RC4	53	47	**B6**	RC5	54	48
B7	RC6	57	58	**B8**	RC7	58	59
B9	RB0/INT	63	62	**B10**	RB1	64	63
B11	RB2	68	69	**B12**	RD3	69	70
B13	RB4	73	74	**B14**	RB5	74	75
B15	RB6	86	83	**B16**	RB7	87	84

Jumper & Header Settings

Programming jumper
JP9, JP10 & JP11 are programming selection jumpers. User has to short 1-2 during programming and after programming the PIC controller, short 2-3.

JP1

JP1			JP2			JP3	
Pin1	1Hz clock from RTC.		Pin1	Ra4		Pin1	PCRX
Pin 2	Ground		Pin 2	Rb0/int		Pin 2	PCTX
			Pin 3	Ground		Pin 3	Ground

RX2 (JP5)
Port RC7 is used for RX channel of RS-232, also this port pin is shard with base bus and by removing this jumper you can isolate the RS-232 from the base bus.

TX (JP6)
Port RC6 is used for TX channel of RS-232, also this port pin is shard with base bus and by removing this jumper you can isolate the RS-232 from the base bus.

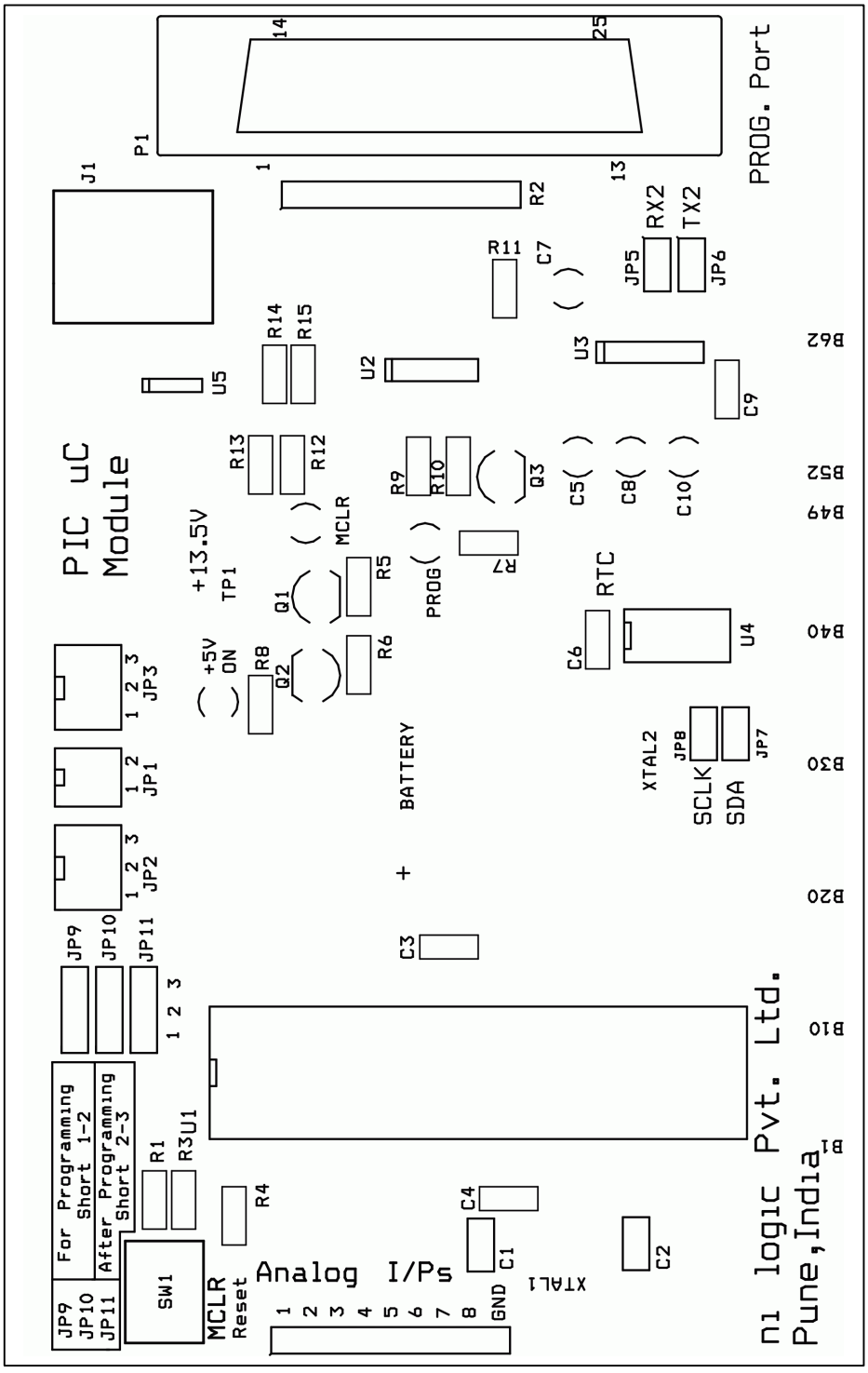
SDA (JP7)
Port RC4 is used for serial data for RTC, also this port pin is shard with base bus and by removing this jum per you can isolate the RTC from the base bus.

SCLK (JP8)
Port RC3 is used for clock for RTC, also this port pin is shard with base bus and by removing this jumper you can isolate the RTC from the base bus.

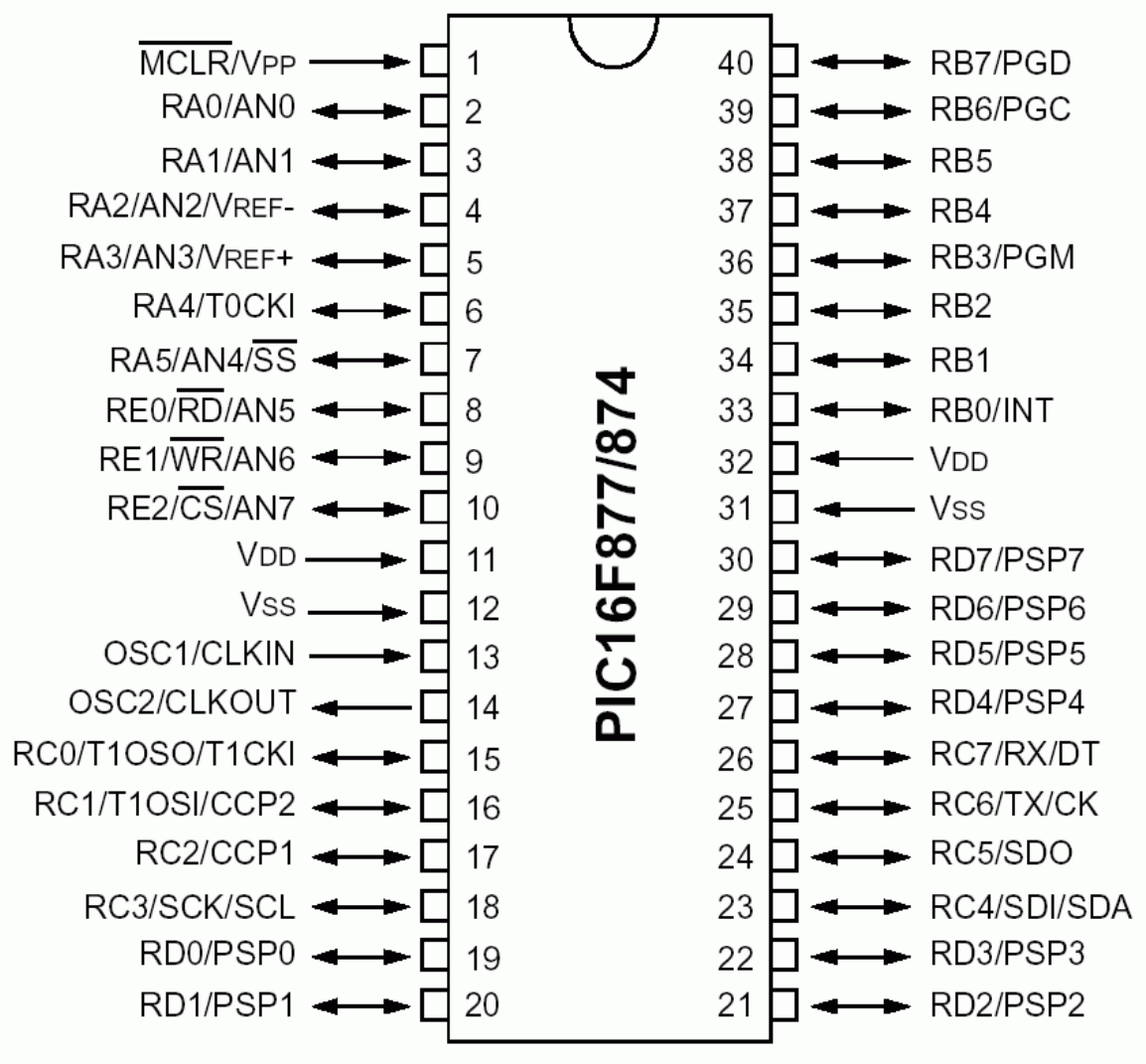
JP9, JP10 & JP11
During programming the PIC controller, **short pins 2-3**, and after programming the controller, **short pins 1-2** for using with FPGA.

Analog I/Ps (Header for inbuilt ADC I/Ps for PIC controller)

Pin1	AN0
Pin 2	AN1
Pin 3	AN2
Pin 4	AN3
Pin 5	AN4
Pin 6	AN5
Pin 7	AN6
Pin 8	AN7
Pin 9	Ground



PIC-16F877 Microcontroller Board Ident

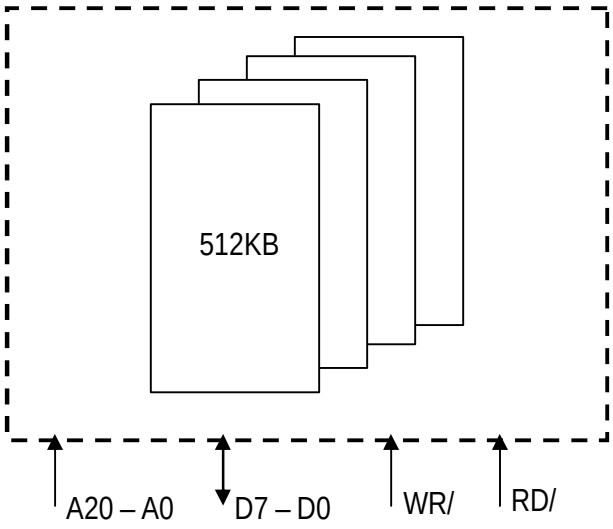


PIC 16F877 Controller Pin out

7. SRAM Memory Card

User gets a SRAM memory module along with the USDP. This has the total 2MB of data storage. The SRAM module contains four 512K x 8 memory chips which user can use according to his application requirement.
All the I/Os of memories are brought on the edge connector, sharing with the PIC slot bus.
This creates choice for user for inserting a memory card or PIC card in any 3 of the slots. User should not insert both, the SRAM module and PIC module together on the board, as there may be conflict on the bus.

SRAM module contains on board address decoder for generating chip select for four memory chips, this makes the module more easier in use by just providing the address lines and control signals for data transfer.

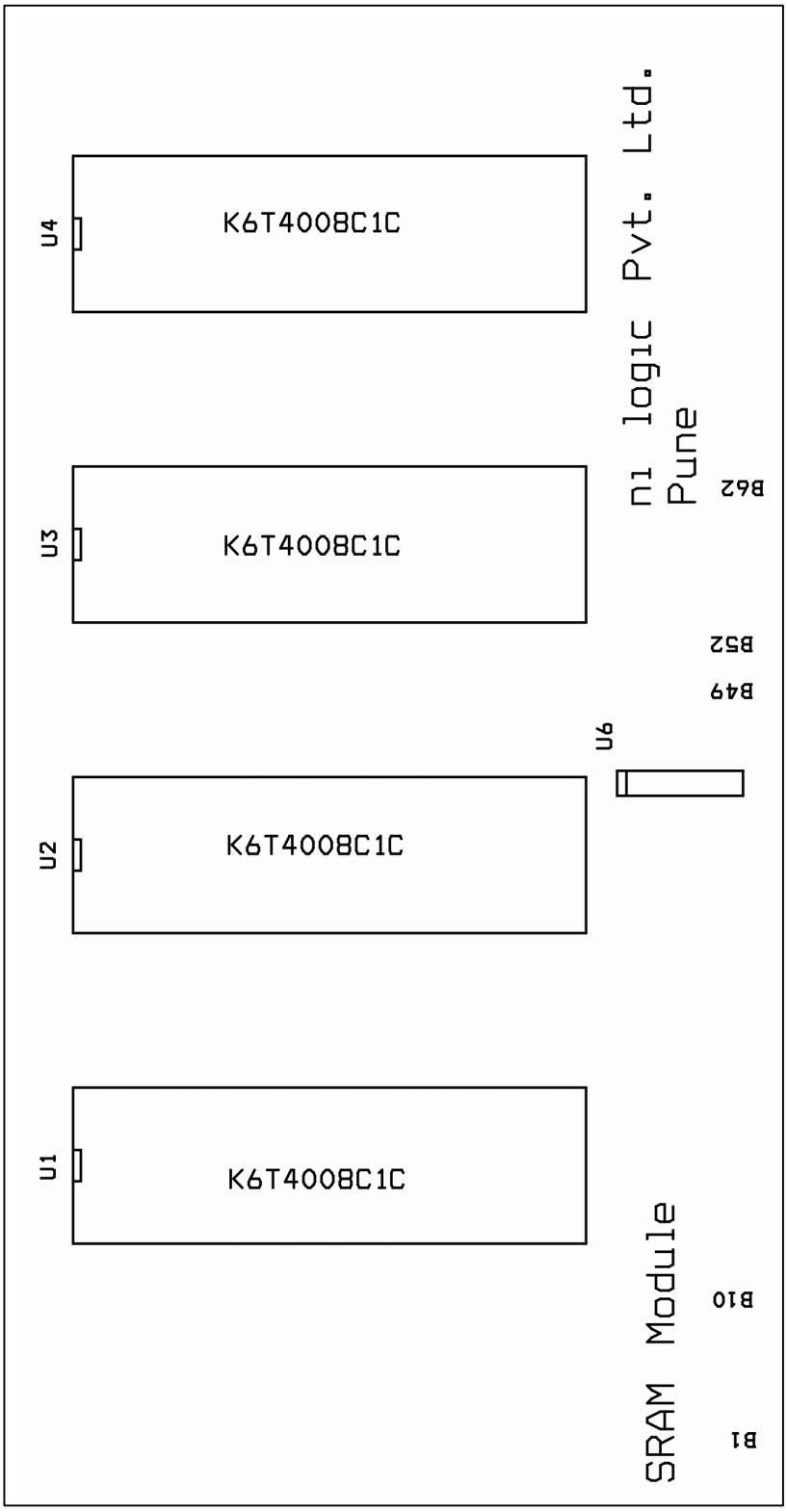


SRAM Memory Module

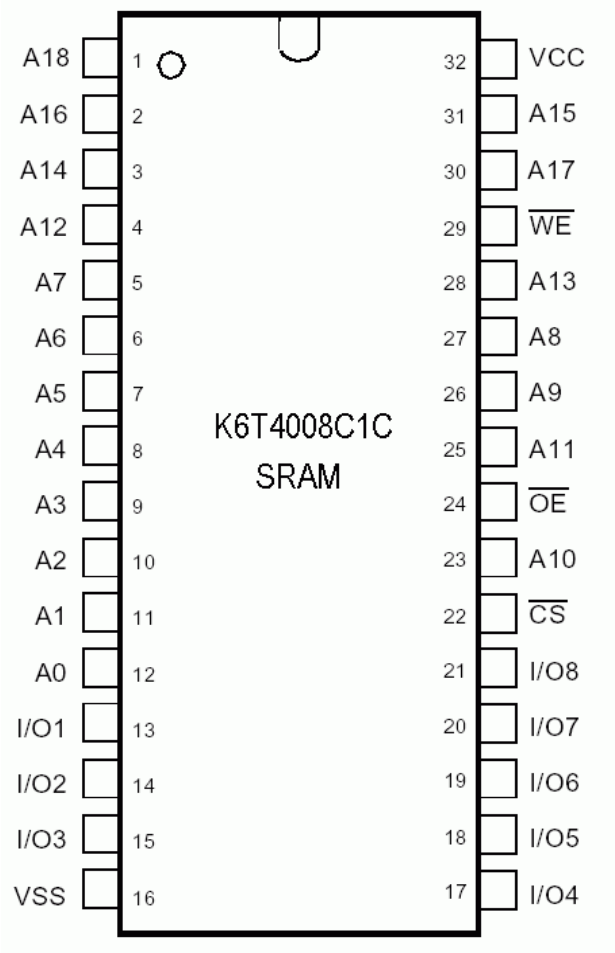
SRAM Module Pin details*

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
A1	A16	36	34	**A2**	A17	37	35
A3	A18	40	41	**A4**	D0	41	42
A5	D1	46	45	**A6**	D2	47	46
A7	D3	55	49	**A8**	D4	56	57
A9	D5	60	60	**A10**	D6	61	61
A11	D7	65	67	**A12**	RD/	67	68
A13	WR/	70	71	**A14**	A19	71	73
A15	A20	75	81				

		Acex 1K	Spartan-II			Acex 1K	Spartan-II
B1	A0	38	36	**B2**	A1	39	37
B3	A2	44	43	**B4**	A3	45	44
B5	A4	53	47	**B6**	A5	54	48
B7	A6	57	58	**B8**	A7	58	59
B9	A8	63	62	**B10**	A9	64	63
B11	A10	68	69	**B12**	A11	69	70
B13	A12	73	74	**B14**	A13	74	75
B15	A14	86	83	**B16**	A15	87	84



SRAM Memory Module Board Ident



K6T4008C1C SRAM Memory Pin out

8. Power Electronics Module

Power Electronics Module is external module provided along with USDP for power electronics based applications.
The module is be divided in 6 sections,

- Relays
- Step down transformer.
- Isolated O/Ps.
- IGBT.
- High current rectifier.
- Stepper motor controller.

We would describe individual section usage; further designer can join the parts together depending on their application.

? Relay section

Two optically isolated relays have been provided onboard. User can turn relays ON by applying logic high on the given control I/P pins.
The relay pins are taken out on terminal headers from where user can interface his circuit for mechanical switch operation. For example user can connect light bulb, heater, buzzer, etc. for visual demonstration.

? Step down transformer

A divide by 100 transformer is been provided onboard, which user can use for line monitoring applications.
User can connect the AC line to its I/P header (JP4), which in turn get a divided by 100 O/P voltage on its secondary winding header (JP5). The secondary winding is center tapped, that means user will get the divide by 100 voltage on the outer pins of winding, the center tap can be used as a reference point.
User can connect this divided O/P voltage to ADC and use the digitized signal for the processing of AC line signal. For example designer can use for frequency measurement of AC line, voltage measurement, line monitoring, etc applications.

? Isolated O/Ps

Five optically isolated O/Ps are provided onboard. User can drive the I/P signal pins high to turn ON the optical isolator transistor. The emitter of optical transistors are been shorted together and name as IGND, user has to use all O/Ps with this common signal. Designer has to make digital ground (DGND) & IGND common if he is working with the same reference voltage levels.
The collector & IGND of optical transistors are brought on the terminal header (JP6) from where user can interface his logic.

? IGBT

A high current rating IGBT is provided onboard for motor control or high current based applications. The IGBT pins are brought on the terminal header (J2) from where user can interface his logic.
As the triggering logic varies from application to application, we have not given the circuit onboard, and user has to design and interface his own triggering logic to drive the IGBT ON/OFF.

Note: While designing the triggering logic of IGBT, use optical isolator in between the FPGA/uC and IGBT. Do not short the analog ground (AGND) & digital ground (DGND) directly, instead use a fuse in between both ground to protect from over flow of current in digital ground.

? **High current rectifier**

A High current rectifier is provided onboard, which can be used for rectifying AC lines upto 10 Amps of current. Designer can apply AC I/P to rectifier from its terminal header (JP1) and get the rectified DC O/P from terminal header (JP3).

Note: Take care while handling the rectifier terminal headers, as due to high voltage you may get shock, so please use rubber glove while handling the power module.

? **Stepper motor controller**

Power module has onboard transistor based stepper motor driver circuit. Designers can interface their uni-polar stepper motor on the provided terminal header (JP8). This circuit can drive motors of +12V/0.5 amps.
User can apply phase control signals to the stepper motor I/P pins in proper sequence to run the stepper motor. The sequence of signal will determine the motor direction and stepping mode.

Note: Connect the motor winding the proper sequence, else the motor won't rotate and will keep on vibrating.

Source codes for Power Module (FPGA)

? Relay section
Source code name = **relay.vhd**
Pin lock file name = **relay.ucf**

? Stepper motor section
Source code name = **stepper.vhd**
Pin lock file name = **stepper.ucf**

Note: Take extensive care while handling the power module in high voltages, as there may be chance of shock in case of improper handling.
Check your design before connecting with the power module from your instructor.
ni logic will not take responsibility in case of any damages due to false design practice and improper handling.

Power Module Pin Detail

Power module is independent of USDP baseboard, and can me mapped to any of the I/Os of the adaptors available.
Here by are the header details of the power module.

JP1 (AC I/P)	
Pin No.	**Signal**
1	Line
2	Neutral

JP2 (DC O/P)	
Pin No.	**Signal**
1	VDC
2	AGND (Analog ground)

JP4 (AC Line, transformer)	
Pin No.	**Signal**
1	Primary1
2	Primary2

JP5 (Step O/P / 100)	
Pin No.	**Signal**
1	Secondary1
2	Common
3	Secondary2

JP8 (Stepper Motor)	
Pin No.	**Signal**
1	W1
2	W2
3	W3
4	W4
5	DGND (digital ground)

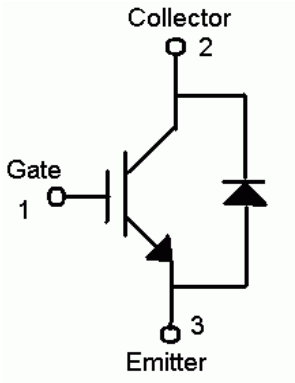
JP6 (Optically Isolated O/Ps)	
Pin No.	**Signal**
1	OP1
2	OP2
3	OP3
4	OP4
5	OP5
6	IGND (Isolated ground)
7	DGND (Digital ground)

JP7 (Relay1)		JP9 (Relay2)
Pin No.	**Signal**	**Signal**
1	Common	Common
2	NO	NC
3	NO	NC

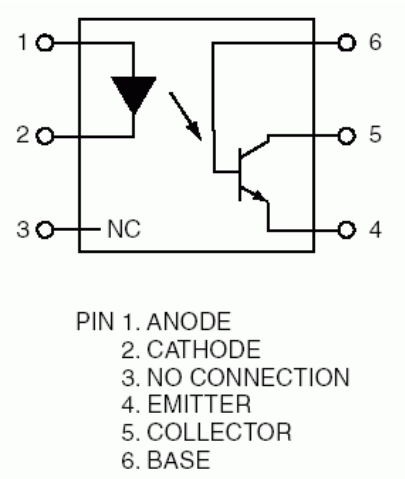
J2 (IGBT)		
Pin no.	**Signal**	**Description**
1	G	Gate
2	C	Collector
3	E	Emitter

JP10 (FPGA Interface)				
Pin No.	**Signal**	**Acex 1K**	**Spartan-II**	**Description**
1	IP1	205	206	Isolated I/P 1
2	IP2	203	205	Isolated I/P 2
3	IP3	202	204	Isolated I/P 3
4	IP4	200	202	Isolated I/P 4
5	IP5	199	201	Isolated I/P 5
6	RELAY1	198	199	Relay control 1
7	RELAY2	197	195	Relay control 2
8	W2	196	194	Winding 2 control
9	W1	195	193	Winding 1 control
10	W4	193	192	Winding 4 control
11	W3	192	191	Winding 3 control
12-16	DGND (Digital ground)			

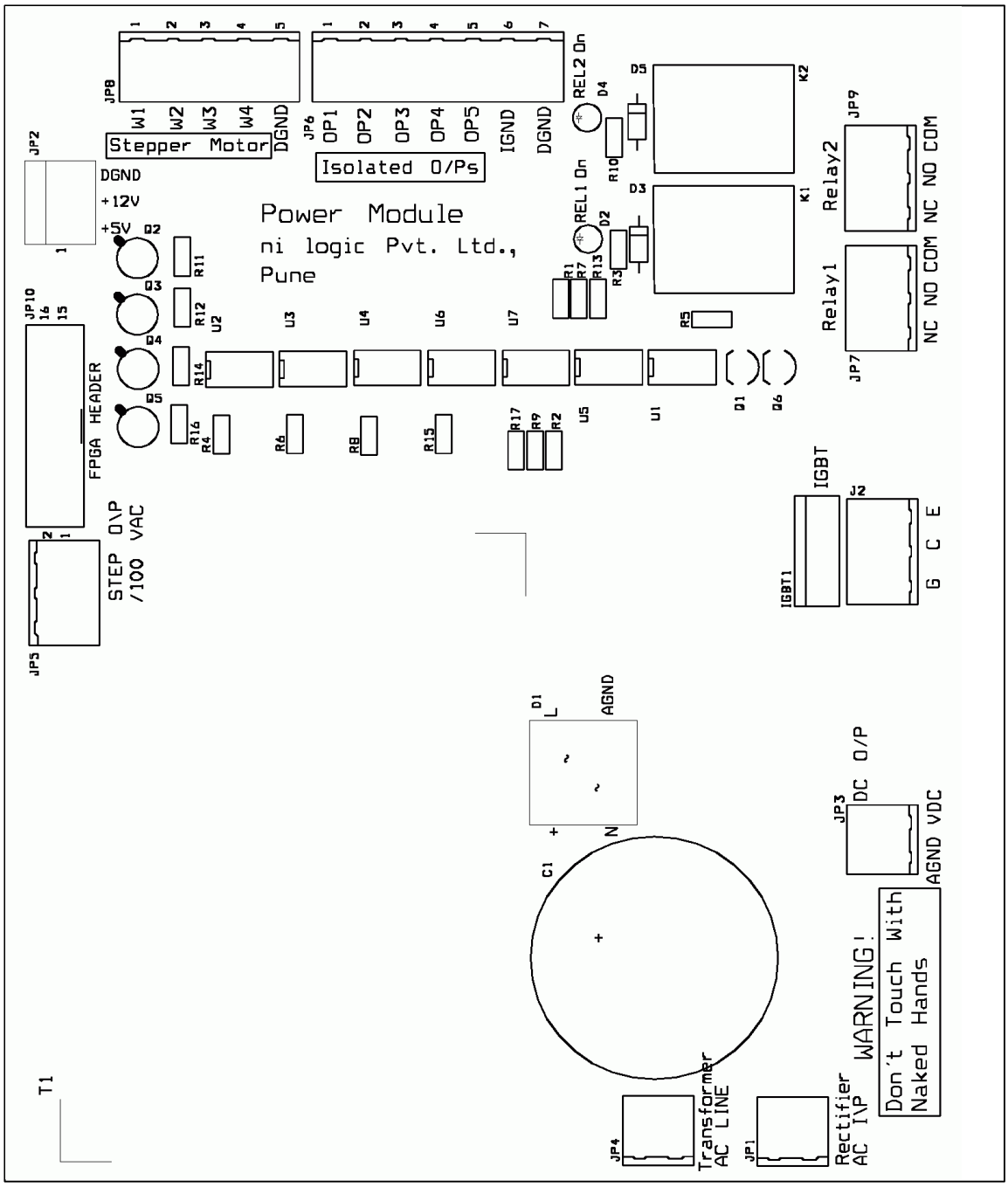
Note: Connect the power electronics module to FPGA with the provided cable only.



IGBT 1MB60D Circuit Schematic



Opto-coupler MCT2E Pin out



Power Module Board Ident

9. General Purpose PCB

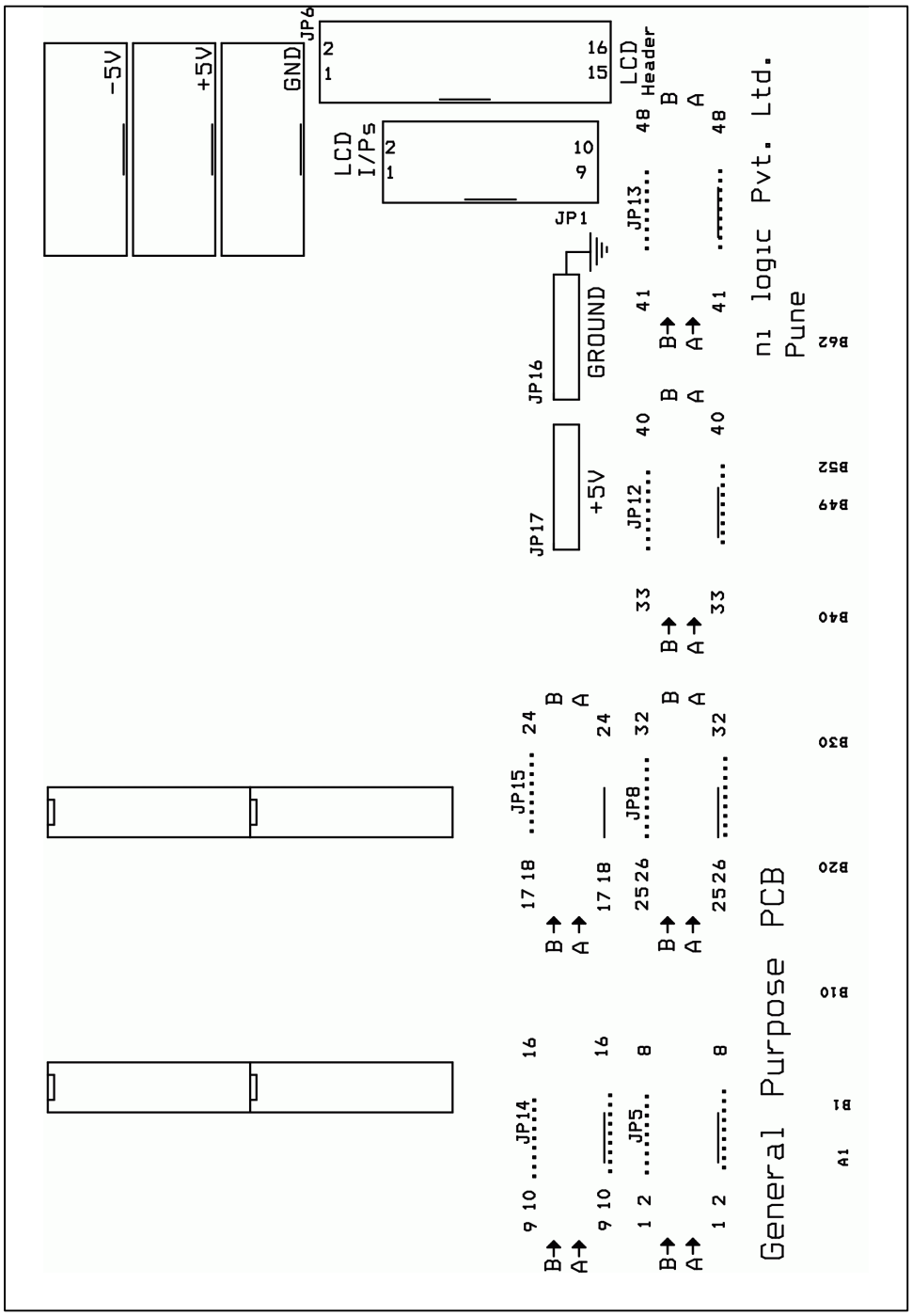
General Purpose PCB can be used for glue logic design, analog circuit, or any other custom circuit interface with the USDP modules.
The LCD header provided on the PCB can be used to drive the LCD module provided along with USDP. The header (JP6) connects with the LCD module, and the LCD I/Os header (JP13) can be connected to the general I/Os provided on board. This gives flexibility to interface the LCD with any one of the USDP modules.

JP6 (LCD Header)

Pin No.	Signal	Pin No.	Signal
1	GND	2	+5V
3	NC	4	RS
5	NC	6	EN
7	D0	8	D1
9	D2	10	D3
11	D4	12	D5
13	D6	14	D7
15	NC	16	NC

JP1 (LCD I/Ps)

Pin No.	Signal	Pin No.	Signal
1	D0	2	RS
3	D2	4	EN
5	D4	6	D1
7	D6	8	D3
9	D7	10	D5

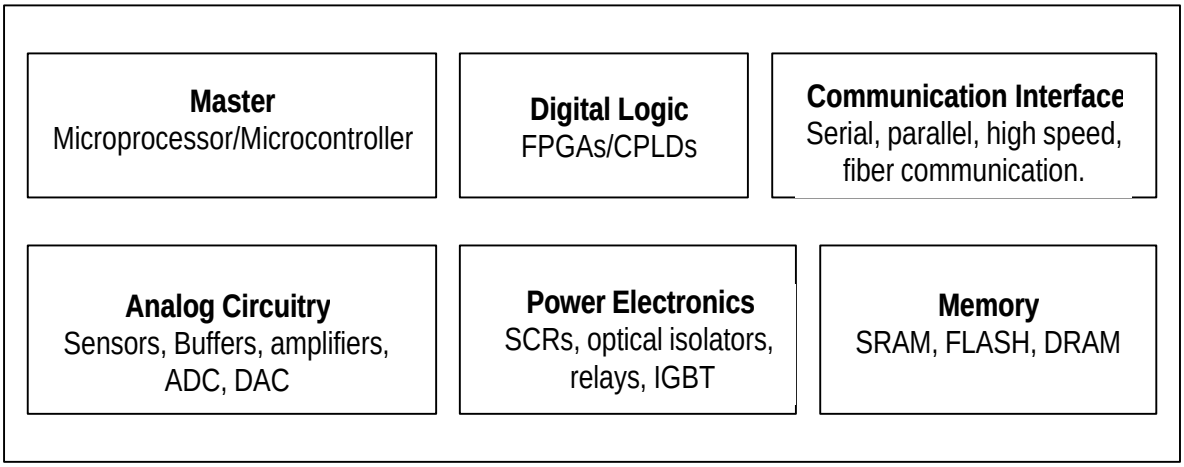


General Purpose PCB Board Ident

Chapter 11
Designing Application on USDP using provided modules

Any electronic system is developed for an application solving a problem or giving ease to product usage.
A system can be defined as "A group of independent but inter related elements comprising a unified whole".
Or "A processing platform, where all element work together for a goal, with the given instructions. "

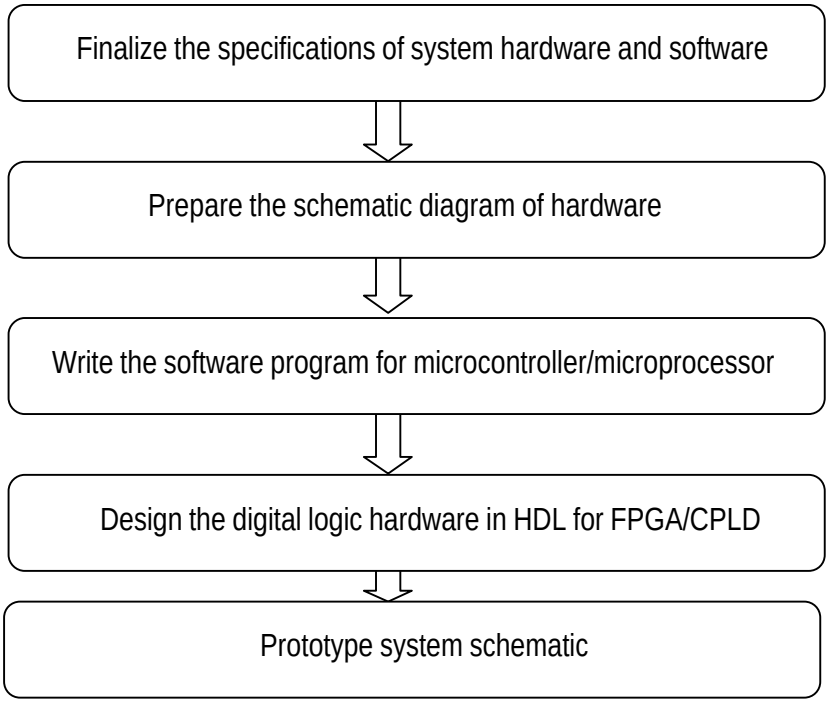
An basic electronic system (see figure below) consists of a master which can be microprocessor or a microcontroller, a memory store to the instructions or data, analog circuitry to interface with the external world, digital logic to process the data at high speed and power electronics components to control high voltage/current peripherals.



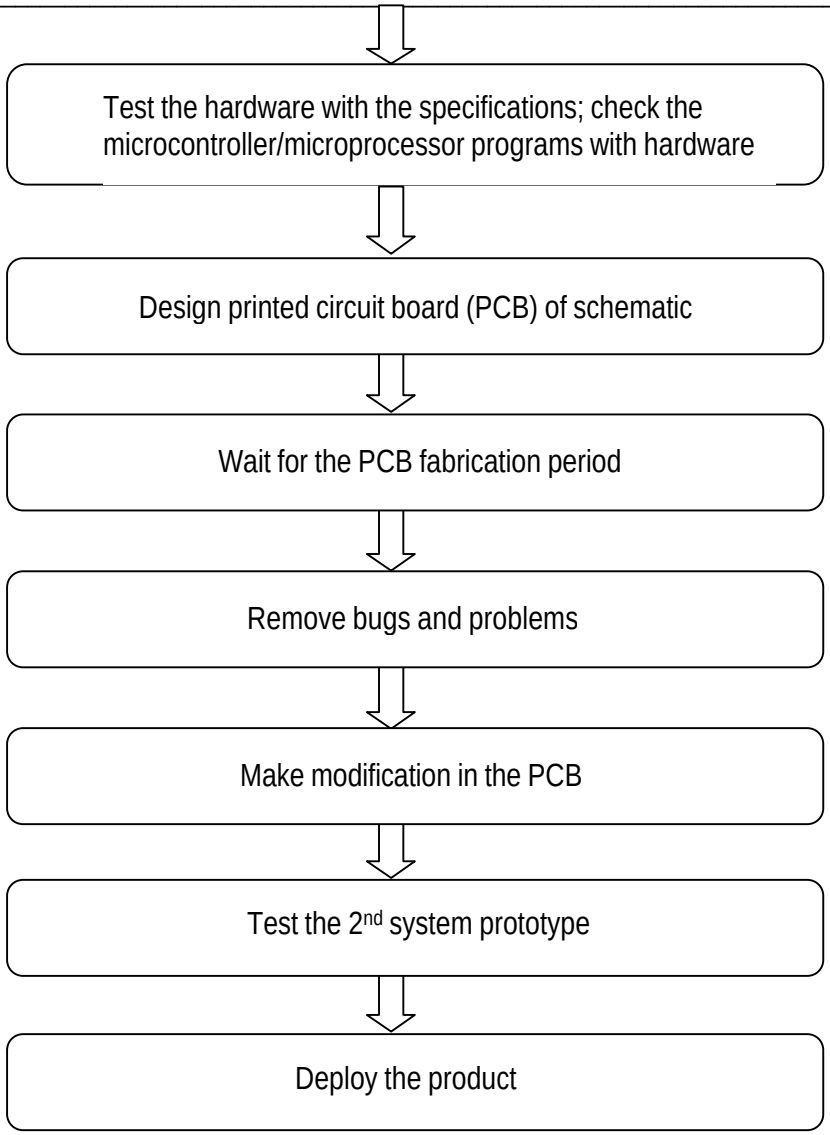
Block diagram of electronic system

With the above basic components, designer can develop an electronic system for any application. More or less few components can be added or removed, but with the today's market scenario and product requirement, designers need the above system blocks to be in their system.
Now we need to see that what are the design steps through which a system passes before getting into the market or its deployment.

Design Flow / steps



.
.



The above flow contains the basic steps, which are majorly followed in industry. Looking at the above steps the most problematic stages are prototyping the system schematic and testing the hardware, which consumes the maximum amount of time.
As the prototyping is generally done on breadboards or the general purpose PCB, this takes intensive care on making it and many times there are errors in building the circuit. Also during testing, designers should be given flexibility to modify the schematic instantly and to carry various tests for the available schematic.
And it is hectic task to design with these traditional steps, also they don't offer flexibility and designers don't get the privilege to market the product in short span of time.

Now here **USDP** gives the advantage over traditional methods of design and other protoboards available. As it is universal platform and provided nearly all the modules for system design, designers can just plug the required modules and design/verify their application in very short span of time.
USDP also serves as a very good platform to train students on concept of electronic system designing. As all the basic technology platforms are integrated on **USDP** (VLSI, Microcontroller, power electronics, communication, etc), students can work on projects where they can have real life practical experience for system design. This will make them more mature about practical concepts of electronics, and give them chance to learn real life project development.
Also the modules are independent to each other and do not depend on the base board, thus any further modules which **ni logic** will introduce can be plugged on existing system and up gradation of **USDP** can be easily done.
Only the designer has to take care that the all the modules are connected with FPGA independently, thus user has to included FPGA in their designs, but further after verifying logic, they can exclude it while going for final product design.

Chapter 12
An Application Implementation on USDP

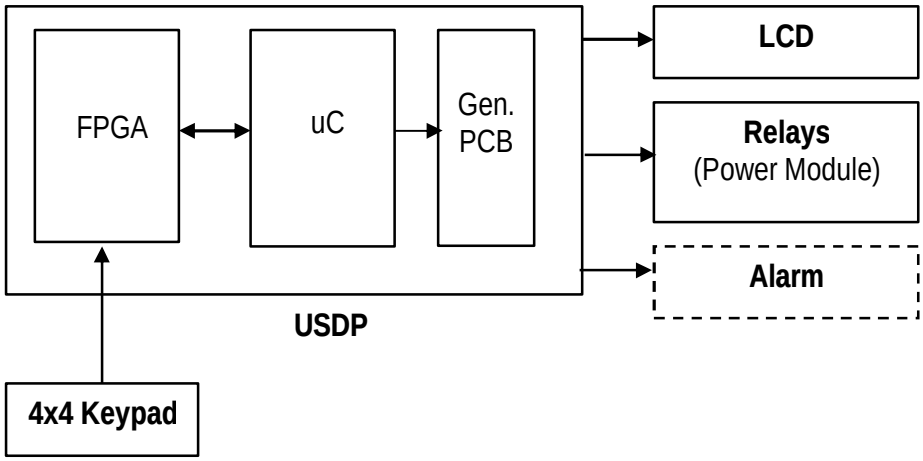
In this chapter we take an example for developing a real life example on USDP.

Application: Access control system

Today many applications are developed for security and access controlling. The basic applications of access control can be developed and prototyped on USDP. The basic model consists of keypad interface for password entering, solenoid for door open & close which can be replaced with relays here, user display for welcome notes, menus and messages displays.

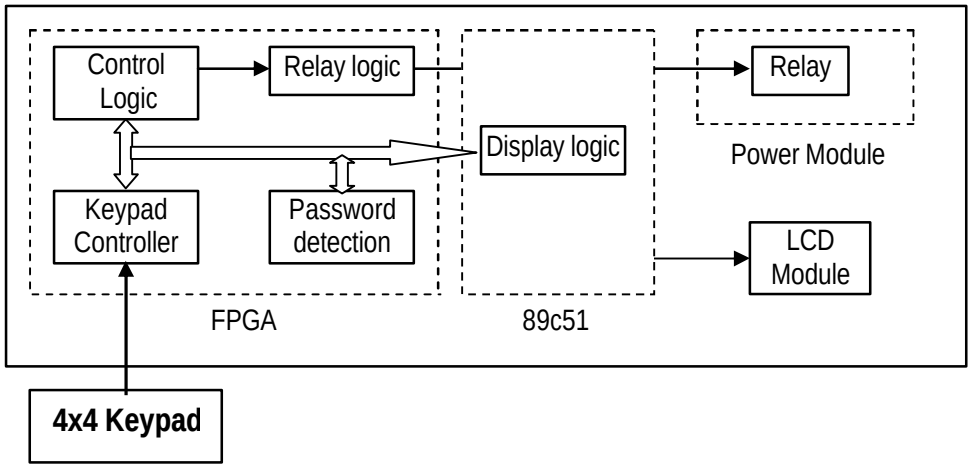
Modules to be used

89C51 module, Xilinx FPGA module, LCD module, general purpose PCB, Power module and keypad.



Application description:

We keep the example very simple; user reaches to gate and reads the message, **"Enter the Password",** then he has to enter the 3-digit password (0 – 9) to open the gate (relay will indicate the gate opening), after entering the password he has to enter the **"F1"** key, which in turn will take the password. If the p assword is right then the relay will turn ON, and LCD will display **"Password OK"**. If the password is wrong then the relay will remain OFF, and a LED will be glown indicating false password and LCD will display **"Password Not Matched".**
This is a simple access control system, further user can modify the given source codes to make it more complex in terms of number of trials, buzzer O/Ps, and change of password, etc.
Password will be **"143".**



Block diagram of Access Control System

Beginning with application:
User has to get together the modules specified, and use them to run the application on USDP.
Here is the procedure to be followed to begin with application:

- Insert 89c51 Module on USDP
- Program it with the given programming file **access_control.hex**.
- Turn off the USDP power supply.
- Insert general purpose PCB in USDP & make connections with LCD module.
- Interface the USDP with power module and keypad.
- Now insert the Xilinx FPGA module in USDP, and program it with the given source code **access_control.vhd**.
- Reset the system and check the application.

Designers can use the sample codes provided along with USDP. The list of codes to be used for this application is listed below. User can refer these codes for their other applications and modify accordingly to their requirements.

Source code for FPGA
Source code name = **access_control.vhd**
Pin lock file name = **access_control.ucf**

Source code for 89c51
Source code name = **access_control.c**
Header file name **= LCD_routine.c**
Programming file name = **access_control.hex**

Note: Save the **LCD_routine.c** file in your project folder while compiling the **access_control.c** file.

Chapter 13
Sample Codes

Here is the list of sample codes provided along with the USDP.

ADC/DAC Module

Code description:
This code works as feed through circuit between the ADC and DAC. The data coming from ADC is stored in latch and given to DAC for reconstruction of signal.

Working with code:
User has to insert ADC/DAC module, connect the analog signal on the ADC1 channel, and CRO probe to DAC1 channel (user can modify the code for channel selection). Program the Xilinx/Altera FPGA module for the given VHDL source code and pin constraint file. After programming the FPGA, check the waveforms on the CRO, which would be nearly the same analog signal, the difference would be due to sampling interval and amplitude difference due to resolution of ADC and DAC.

Source code for FPGA
Source code name = **ADC_DAC_feed.vhd**
Pin lock file name = **ADC_DAC_feed.ucf**

89c51 Module

Code description:
The 89c51 source code takes data from PORT0 and transmits it to serially to PC whenever there is change on the port data.

Working with code:
Program the 89c51 with the given HEX file, there after program the FPGA with the provided feed through circuit between switch and 89c51. After programming FPGA, change the position of LD7 to LD0 to observe the changes on the serial port of PC.
To the read he data from serial port of PC, users can write a program in 'C', or can use the COM port reader software provided along with USDP.

Source code for FPGA
Source code name = **feed_89c51.vhd**
Pin lock file name = **feed_89c51.ucf**

Source code for 89c51
Source code name = **serial_transmit.c**
Programming file name = **serial_transmit.hex**

PIC Module

Code description:
This source code has a message-displaying program. This displays **"HI Iam USDP"** on the 7-segment displays. Users can use this code for their reference or modify according to their requirement.

Working with code:
All the PIC I/Os are shared with the LEDs also, so remove all the jumpers of switches so that there is no conflict after programming the controller. Now, Program the PIC microcontroller using the provided programmer software. Use the HEX file provided with the USDP.

There after program the FPGA with the provided feed through circuit VHDL code. This VHDL code acts like the feed through between the reset switch and the 7-segment displays. After programming the FPGA, reset the controller and check the message display on the 7-segment displays.

Note: As the memory module also shares the same bus with PIC, so remove the memory module while using the PIC module.

Source code for FPGA
Source code name = **feed_PIC.vhd**
Pin lock file name = **feed_PIC.ucf**

Source code for PIC
Source code name = **disp_pic.c**
Programming file name = **disp_pic.hex**

Power Electronics Module

Stepper Motor Controller Section
Code description:
This source code can control uni polar stepper motors. The code can control the direction, speed and stepping of motor. This can be done with the help of switches provided on board.

Working with code:
Insert he FPGA module on USDP slot, connect the power module interface cable with FPGA through the parallel port connector provided.
Connect the stepper motor winding on the header (JP8), while connecting connect the winding the same phase as mentioned on the motor.
Program the FPGA with the given source code, and control the motor signals from the switches.

Source code for FPGA
Stepper motor section
Source code name = **stepper.vhd**
Pin lock file name = **stepper.ucf**

Relay section
Code description:
This source code can control the relays provided on the power module. User can turn ON the relay by providing logic High on the relay I/P pins.

Working with code:
Insert he FPGA module on USDP slot, connect the power module interface cable with FPGA through general purpose PCB or the parallel port connector provided. In this case user has to take care of pin assignment.
Connect your application signals on the relay headers (JP7 & JP9), while connecting take care for Normally Open (NO), Normally Close (NC) and Common (COM) ports of relay.
Program the FPGA with the given source code, and control the relays from the switches.

Source code name = **relay.vhd**
Pin lock file name = **relay.ucf**

Keypad Controller

Code description:
This code is the 4x4 keypad controller. Users can press the keys and see the key value on the 7-segment displays. Further users can modify this code for their applications.

Working with code:
Insert he FPGA module on USDP slot, connect the keypad with the cable provided. Program the FPGA with the given source code. Press the keys of keypad and check the displayed value on the 7-segment displays.

Source code name = **keypad.vhd**
Pin lock file name = **keypad.ucf**

LCD Module

Code description:
This code is generalized code written in 'C' for message displaying on LCD module. The code written is for 89c51RD2 controller. At bottom of source code, there is a **dispstr** function, which displays the written message in its body on LCD. Users can modify this **text** line to change the message to be displayed.

Working with code:
Insert the 89c51 module in USDP; connect the LCD module to 89c51 through the general purpose PCB.
Program the 89c51 controller with the given HEX code, and check the message displayed.

Source code name = **LCD_USDP.c**
Programming file name = **LCD_USDP.hex**

Access control application

Code description:
Sample codes are provided along with USDP for access control application. Samples are provided for FPGA and 89c51 controller. The FPGA consists the control logic, and controller has the LCD logic.

Working with code:
Insert the 89c51 module in USDP; connect the LCD module to 89c51 through the general purpose PCB. Connect the keypad to its header; make connections with the power electronics module through general purpose PCB.

Program the 89c51 controller with the programmer provided; turn OFF the power supply, insert the FPGA module & program it with the given source code and pin lock file.

Source code for FPGA
Source code name = **access_control.vhd**
Pin lock file name = **access_control.ucf**

Source code for 89c51
Source code name = **access_control.c**
Header file name **= LCD_routine.c**
Programming file name = **access_control.hex**

Chapter 14
Glossary of Terms

ASIC (Application Specific Integrated Circuit)

A custom integrated circuit designed specifically for one end product or a closely related family of end products.

Analog, Digital and Mixed Signal

A circuit used to count the number of events is generally digital. Sometimes chip are called mixed signal chips which means that they contain both analog and digital circuits

Analog-to-Digital Converter (ADC)

An electronic circuit that converts a continuously varying signal (temperature, pressure, voltage, etc.) into digital zeroes and ones that can be processed by a microprocessor or microcontroller. Converts an analog signal sample to a digital representation suitable for digital processing and switching.

Asynchronous

Asynchronous system (computer, circuit, device) is one in which events are not executed in a regular time relationships. They are timing independent. Each event or operation is performed upon receipt of a signal generated by the completion of a previous event or operation, or upon availability of the system resources required by the event or operation.

C

Common programming language used in science, engineering and DSP. Also comes in the more advanced C++.

Concurrency

The ability of an electronic circuit to do several (or at least two) different things at the same time. Contrast with computer programs, which usually execute only one instruction at a time unless the program is running on a processor with multiple, concurrent execution units.

Combinational Logic

Combinational logic is purely functional logic, which does not maintain any internal state. Thus it will provide the same result for the same input no matter what sequence the input is sent.

CPLD (Complex Programmable Logic Device)

A programmable IC which is more complex than the original Programmable Logic Devices such as AMD's (originally MMI's) PALs but somewhat less complex than Field Programmable Logic Arrays.

Digital Signal Processor (DSP)

A processor system specialized for the computation of signal processing algorithms. It usually consists of many programmable processor elements interconnected via networks to each other and to memory, sensors, displays and other external devices. It is often distinguished from general purpose-or data processors in that is must operate in real-time; it often has a much higher data input rate. and it usuall y must perform a higher percentage of mathematical, often floating-point, operations.

Digital-to-Analog Converter (DAC)

A circuit that translates a signal from a numeric, digital representation used by microprocessors and microcontrollers into an analog signal. Converts a digital word to an analog value.

EDIF (Electronic Design Interchange Format)

A standard representation format for describing electronic circuits, used to allow the interchange of circuit design information between EDA tools.

FPGA (Field Programmable Gate Array)

An integrated circuit containing a large number of logic cells or gates that can be programmably configured after the IC has been manufactured. Some FPGAs use fuses for this programming and others store the configuration in an on chip EEPROM or RAM memory. Fuse programmed parts cannot be reprogrammed so they can only be configured once. EEPROM based FPGAs can be erased and reprogrammed so they can be configured many times. RAM based FPGAs can be reconfigured quickly, even while the circuit is in operation.

Finite Impulse Response (FIR)

An impulse response that has a finite number of nonzero values. Often used to indicate that a filter is carried out by using convolution, rather than recursion.

HDL (Hardware Description Language)

A synthetic computer based language used for the formal description of electronic circuits. An HDL can describe a circuit's operation, its design, and a set of tests to verify circuit operation through simulation. The two most popular digital HDLs are VHDL and Verilog. An analog HDL called AHDL is under development by many vendors. HDLs make it easier to develop very large designs through formal software engineering methods that define ways to divide a large team project into smaller modules that can be implemented by individual team members.

Hardware/Software Codesign

The simultaneous development of product hardware and software. This design approach is more difficult than a serial design which first develops the hardware and then the software that will run on the hardware but the benefit is a reduced time to market. To develop software before hardware is ready, software developers often create a behavioral model of the hardware which can run the software and thus prove its function.

Interrupt

An input to a processor that signals the occurrence of an outside event; the processor's response to an interrupt is to save the current machine state and execute a predefined subprogram.
The subprogram restores the machine state on exit and the processor continues in the original program

Joint Test Action Group (JTAG)

The Joint Test Action Group. This group created the foundation for the IEEE work.
Set of specifications that enable board and chip level functional verification of a board during production. Committee that established the Test Access Port (TAP) and boundary-scan architecture defined in IEEE Standard 1149.1-1990.

Liquid-Crystal Display (LCD)

The screen technology commonly used in notebook and smaller computers

Logic

The sequence of functions performed by hardware or software. Hardware logic is made up of circuits that perform an operation. Software logic is the sequence of instructions in a program.

Moore's Law

An empirical law developed and later revised by Intel's Gordon Moore which predicts that the IC industry is capable of doubling the number of transistors on a silicon chip every 18 months (originally every year) resulting in declining IC prices and increasing performance. Most design cycles in the electronics industry including embedded system development firmly rely on Moore's law.

Net List

A computer file (sometimes a printed listing) containing a list of the signals in an electronic design and all of the circuit elements (transistors, resistors, capacitors, ICs, etc.) connected to that signal in th e design.

PLCC (Plastic Leaded Chip Carrier)

A low cost IC package (usually square). PLCCs have interconnection leads on either two (usually only for memory chips) or all four sides (for logic and ASIC chips).

PLD (Programmable Logic Device)

The generic term for all programmable logic ICs including PLAs (programmable logic arrays), PALs, CPLDs (complex PLDs), and FPGAs (field programmable gate arrays).

Pipelining

Splitting the CPU into a number of stages, which allows multiple instructions to be executed concurrently

PROM (Programmable Read Only Memory)

An integrated circuit that store programs and data in many embedded systems. PROM stores and retains information even when the power is off but it can only be programmed or initialized once.

RTL (Register Transfer Level or Register Transfer Logic)

A register level description of a digital electronic circuit. Registers store intermediate information between clock cycles in a digital circuit, so an RTL description describes what intermediate information is stored, where it is stored within the design, and how that information moves through the design as it operates.

Simulation

Modeling of an electronic circuit (or any other physical system) using computer based algorithms and programming. Simulations can model designs at many levels of abstraction (system, gate, transistor, etc.). Simulation allows engineers to test designs without actually building them and thus can help speed the development of complex electronic systems. However, the simulations are only as good as the mathematical models used to describe the systems; inaccurate models lead to inaccurate simulations. Therefore, accurate component models are essential for accurate simulations.

Synchronous

A digital circuit where all of the operations occur in lock step to a master clock signal A mode of transmission in which the sending and receiving terminal equipment are operating continually at the same rate and are maintained in a desired phase relationship by an appropriate means An operation or operations that are controlled or synchronized by a clocking signal.

Synthesis (also Logic Synthesis)

A computer process that transforms a circuit description from one level of abstraction to a lower level, usually towards some physical implementation. Synthesis is to hardware design what compilation is to software development. In fact, logic synthesis was originally called hardware compilation.

System-on-a-Chip (SoC)

Combining several chips with different functions onto one, single chip.

TCK

Test Clock, a TAP pin used to supply clocks to the TAP Controller.

TDI

Test Data In, a TAP pin used to shift the test data in to the TAP Controller.

TDO

Test Data Out, a TAP pin used to shift the test data out from the TAP Controller.

TMS

Test Mode Select, a TAP pin that provides the stimulus to change the state of the TAP Controller.

USDP

Universal Development Platform, a integrated platform where designers can put different technology modules all together for their system design.

User Constraints File (UCF)

A user created ASCII file for storing timing constraints and location constraints for a design implementation.

Verilog

A hardware description language developed by Gateway Design Automation (now part of Cadence) in the 1980s which became very popular with ASIC and IC designers.

VHDL (VHSIC Hardware Description Language)

A hardware description language developed in the 1980s by IBM, Texas Instruments, and Intermetrics under US government contract for the Department of Defense's VHSIC (Very High Speed Integrated Circuit) program. VHDL enjoys a growing popularity with ASIC designers as VHDL development tools mature.

Chapter 15
Troubleshooting

Errors while programming FPGA

There may be errors while programming FPGA, this may be due to many reasons, kindly check the following steps to recover the error.

- FPGA modules should be properly inserted.
- Check the jumper settings of FPGA module.
- Check the programming mode selection settings.
- See that the programming cable is properly inserted.
- The PLD & slot selector cards are in proper position.
- Ground the clock (GCK0) I/P during programming; use it after programming the FPGA.
- In case of Xilinx, open the iMPACT programmer with selected programming mode only.
- The parallel port of PC should be in ECP/EPP mode; else there would be connection errors.

Errors while programming 89c51

There may be errors while programming 89c51, kindly check the following steps to recover the error.

- Module should be properly inserted.
- The RS-232 programming cable should be properly inserted.
- Check the Flash Magic programmer settings (refer chapter configuration).

Errors while programming PIC controller

There may be errors while programming PIC16F877, kindly check the following steps to recover the error.

- Module should be properly inserted.
- The programming cable should be properly inserted.
- Check the ProgPIC programmer settings (refer chapter configuration).
- Check the jumper settings of PIC module.
- Plug in the +18V adaptor on the plug socket of module.

Controller modules not working properly

- After programming of 89c51 & PIC controllers, you have reset them from FPGA or from keys; else they won t function properly sometimes.
- Check the I/O connections properly.
- Check the reset logic.
- Check the source code thoroughly, there may be infinite loop or some other problem in code.

ADC/DAC Module not working properly

- Insert the module properly.
- Check the analog signal connections.
- Check the channel is properly selected.
- Check the ADC & DAC logic.

Disclaimer

ni logic pvt. Ltd., Pune takes the liability to replace the module/product for any design fault from our side.

ni logic pvt. Ltd., Pune does not take any responsibility of failures or damages caused to product due to incorrect design practices, misuse, improper handling and not following the datasheet specifications of devices.