CONTENTS

1.0 INTRODUCTION ………………………………………………… 2

2.0 DESCRIPTION OF THE CIRCUIT…………………………… 4

3.0 INSTALLING THE HARDWARE AND DRIVER SOFTWARE ……………………………………… 6

4.0 DRIVER LIBRARIES DESCRIPTION……………………… 10

5.0 APPLICATION DEVELOPMENT USING DRIVERS……… 37

Appendix A: Schematics ……………………………… 52

Appendix B: Component layout ……………………… 55

Appendix C: Connector Details ……………………… 56

ESA PCIDIOT

PCI DIGITAL I/O TIMER CARD FOR PCs

1.0 INTRODUCTION:

Electro Systems Associates Pvt. Ltd. manufactures a variety of microprocessor trainers, development/debugging tools and microcomputer development systems useful for educational institutions and R&D labs.

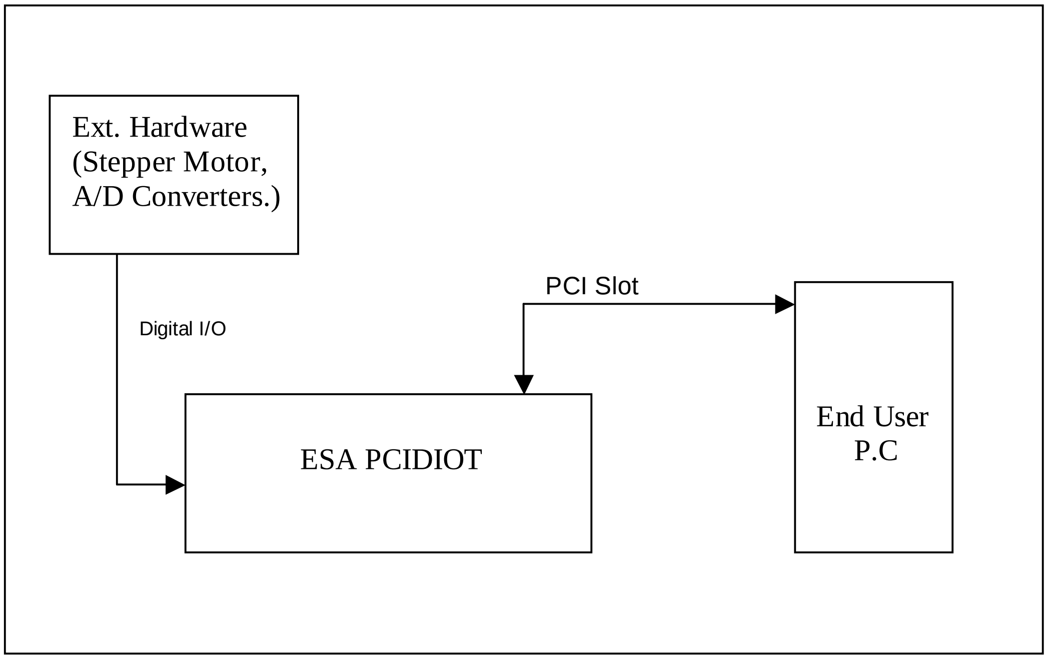
ESA PCIDIOT card is a PCI based Digital Input/Output timer card for PC compatible systems. The card contains two 8255 programmed peripheral interface (PPI) which provide 48 programmable I/O lines for the user and one 8254 programmable interval timer which provide three programmable counter/timers to the user.

ESA PCIDIOT can be plugged into any one of the free PCI slots of the system. This card is accompanied by a Driver CD, which contains Drivers & supporting files.

CARD SPECIFICATIONS:

8255	**:**	Two Nos. – Provide 48 I/O lines
8254	**:**	One No. - Provides 3 Timers OUT Lines 3 Timer GATE Lines 3 Timer CLK Lines
JUMPERS	**:**	Used for setting the Input clock selection to thetimer.
Power Supply	**:**	The card draws power from the system itself. No external Power Supply required.
System	**:**	Any PC compatible system with PCI slots.

Block Diagram



2.0 DESCRIPTION OF THE CIRCUIT:

The card uses a popular PCI Bridge (U8) to interface, two 8255s at U5 & U6 and one 8254 at U3, to the PC through PCI Bus. The two 8255s provide six programmable 8-bit I/O ports.

The 24 I/O lines of U5 are brought to J2, a 26-pin berg connector. The 24 I/O lines of U6 are brought to J3, a 25-pin D-Type connector and also to J4, a 26-pin berg connector. If required, user can buy from Electro Systems Associates Pvt. Ltd., a 25-pin D type female adapter for J2.

The 74LS245 at U9 is a Bi-directional buffer for data bus.

The PIT, 8254 at U3 has three 16-bit programmable timers /counters and can operate up to 2.0 MHz. The OUT, GATE and CLK lines of the PIT are brought to J1, a 15-pin D-Type Connector. Please refer to Appendix C for all connector details.
The jumpers JP1, JP2 and JP3 are used for connecting external or system clock to the clock input of timer of 8254.

JUMPER	CONNECTION	CLOCK USED
JP1	1 – 2 2 – 3	EXT CLOCK to Timer 2 SYS CLOCK to Timer2
JP2	1 – 2 2 – 3	EXT CLOCK to Timer1 SYS CLOCK to Timer1
JP3	1 – 2 2 – 3	EXT CLOCK to Timer0 SYS CLOCK to Timer0

NOTE:

The GATE0, GATE1, GATE2 signals of the 8254 should be controlled by user as per his requirements.

Packing List:

Before you begin installing **ESA PCIDIOT** Hardware, please make sure that the following materials have been shipped to you.

- Ø **ESA PCIDIOT** Hardware.
- Ø 24 pin FRC Cable.
- Ø **ESA PCIDIOT** Software CD containing Windows Driver Software & Sample applications with source developed using VC++ 6.0, VB 6.0 Labwindows\CVI, Turbo C , MASM32 & MASM.
- Ø **ESA PCIDIOT** User's Manual.

Minimum System Requirements:

- Ø IBM Compatible Pentium machine or above.
- Ø Windows 98/Windows NT/Windows 2000/ Windows XP/Windows Me.
- Ø Microsoft Visual Studio / Lab Windows- CVI Development Environment.
- Ø 64 MB of RAM.
- Ø Empty PCI Slot.
- Ø CD-ROM Drive.

NOTE:

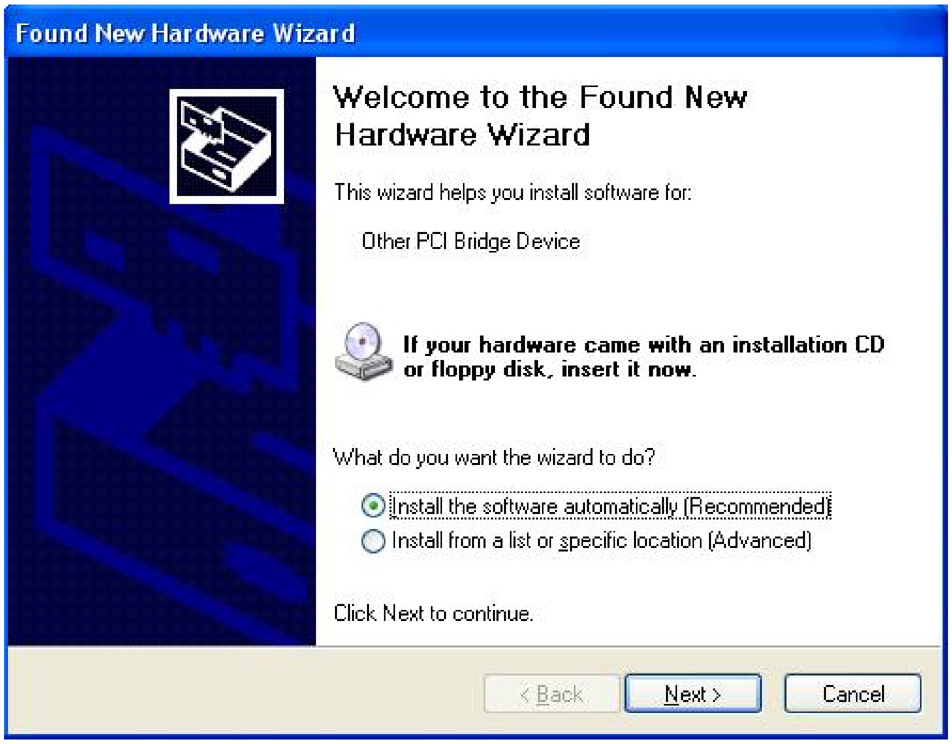
To use the **ESA PCIDIOT** card in **Windows (98/NT/2000/XP)** Environment user need to install Windows driver software and library files available on the Driver Software CD.

To use the card in **DOS**, please refer chapter 5.5 (**DOS** Application Development).

3.0 Installing the Hardware & Driver Software:

1. Switch off, the PC.
2. Remove the power cable from the PC.
3. Plug the **ESA PCIDIOT** Hardware in the free PCI slot available on the PC Motherboard.
4. Plug the power cable to the PC.
5. Switch on, the PC.
6. Windows OS will detect a new hardware and asks for the Installation of Driver.

NOTE:
The user must have administrative privileges on the target computer in order to install the driver for **Windows 2000/XP/NT**.


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

Other PCI Bridge Device

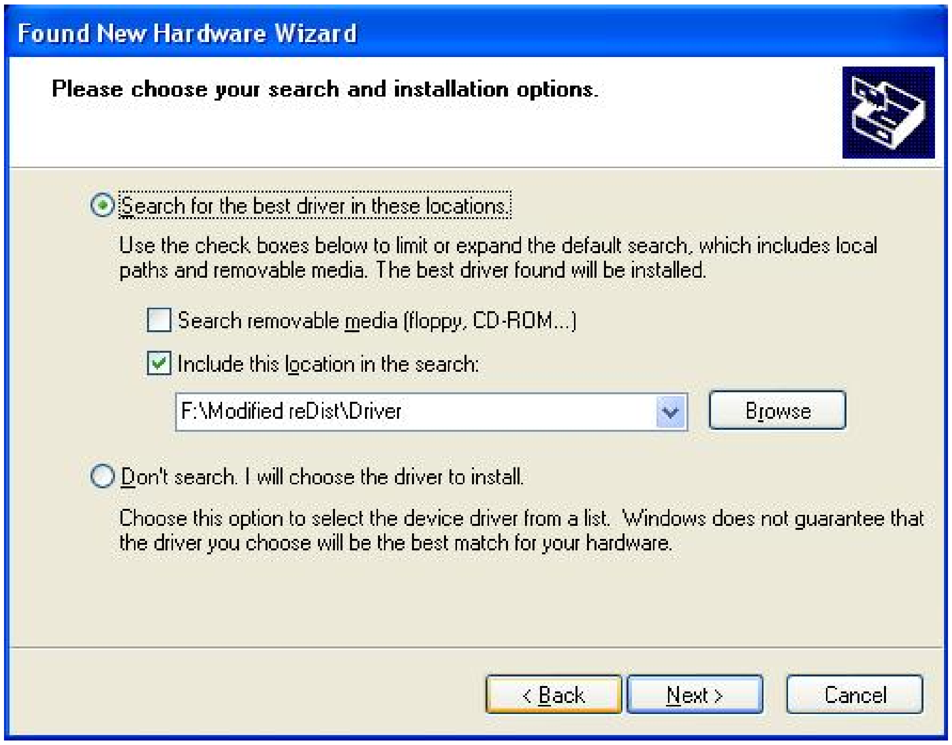
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

[✓ Install the software automatically (Recommended)]
Install from a list or specific location (Advanced)

Click Next to continue.

If user selects **"Install the software automatically"**, it will search for the suitable drivers in local drives and external media. In Case it does not detect any suitable driver, select "**Install from a list or specific location**", you should then see the following window.



Please choose your search and installation options.

(●) Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

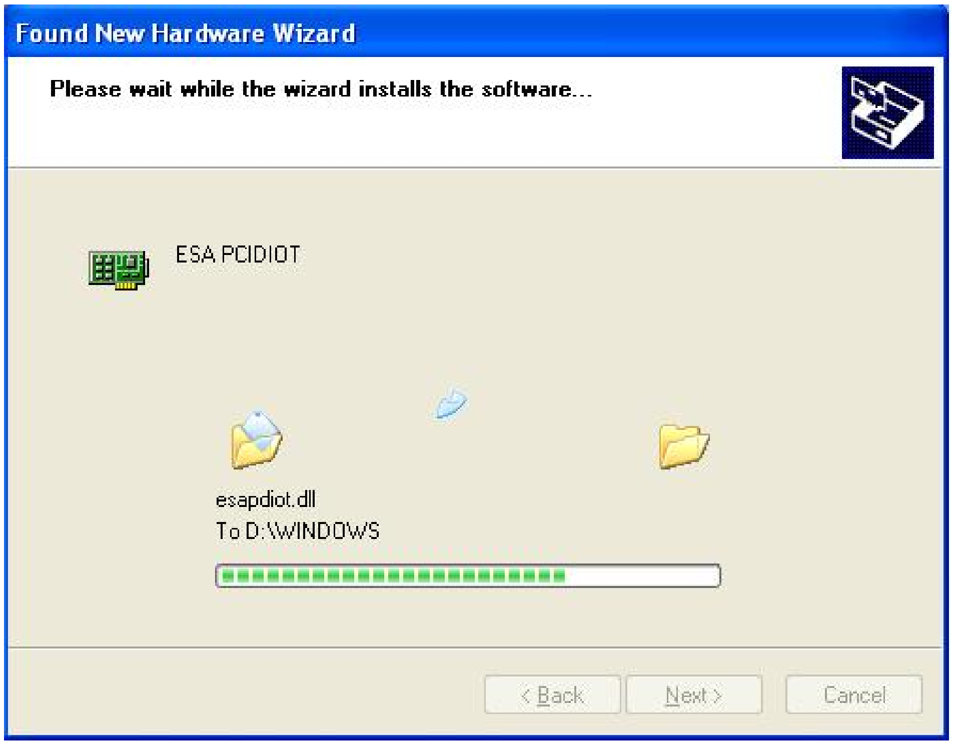
☐ Search removable media (floppy, CD-ROM...)

☑ Include this location in the search:

F:\Modified reDist\Driver

( ) Don't search. I will choose the driver to install.

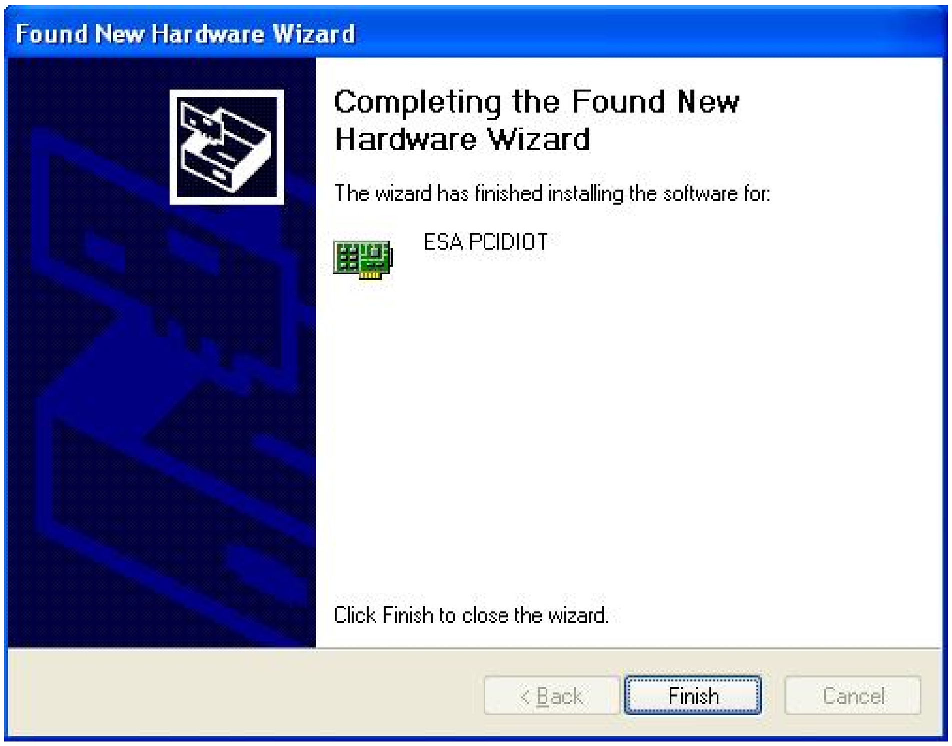
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.



Please wait while the wizard installs the software...

ESA PCIDIOT

esapdiot.dll
To D:\WINDOWS


Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

ESA PCIDIOT

Click Finish to close the wizard.

After Installation is completed, run the following command from the Command Prompt.

(WindowsXP)

G:\Driver\wdreg –inf **C:\windows\system32\drivers\windrvr6.inf** install

(Windows 2000)

G:\Driver\wdreg –inf **C:\winnt\system32\drivers\windrvr6.inf** install

(Windows 98)

G:\Driver\wdreg16 –inf **C:\windows\system32\drivers\windrvr6.inf** install

NOTE: Windows 98 requires a reboot after Installation of the driver for proper working of the **ESA PCIDIOT** card.

Batch files for the above are provided in the Drivers CD, file names are **reg98.bat** (**Windows 98**), **reg2k.bat** (**Windows 2000**) and **regXP.bat** (**Windows XP**).

User has to take care of the Drive names in the batch file while running these batch files.

This Registering could be done for the reboot free installation of the Driver. This could be done at the first time of installation; Next time onwards driver will be activated automatically.

NOTE:
Windows NT Operating Systems doesn't support Plug & Play feature. User has to take care of the driver installation for the card.

Go to the WINNT folder in the Drivers CD.

Edit "install.bat". Confirm the Directory Paths of WINDOWS NT installation.

Edit "uninst.bat". Confirm the Directory Paths.

For installing the driver, Double click on "install.bat" or run "install.bat" from the command window.

For uninstalling the driver, close all the applications that are using this driver. Double click on "uninst.bat" or run "uninst.bat" from the command windows.

Uninstalling the Driver:

Delete the Devices listed in **Device Manager** Under "**ESA**"("**Hardware**" Tab From "**My Computer**" Properties) like **"ESA PCIDIOT"** .

Delete "**windrvr6.sys**" & "**windrvr6.inf**" from **"%windir%\system32\drivers"**

Delete **"esapdiot.dll"** from **"%windir%"** (Ex: **C:\Windows or C:\winnt**)

Delete **"oemxx.inf"** (**Windows2k/XP**) from **"%windir%\inf"** directory or **"esa*.inf"** from **"%windir%\inf\other"** (**Windows 98**).

Restart the PC.

NOTE:

On **Windows 2000/XP/NT**, the **inf** files will be created with **"oemXX.inf"** under **"%windir%\inf"**. To find the **inf** file corresponding to **ESA PCIDIOT** card, user can search the **INF** directory for the **"ESA PCIDIOT"** as a search text.

4.0 Driver Libraries Description:

Function Reference:

1) ESAPCIDIOT_Open()

PURPOSE

Provides Device Handler to access Driver kernel module. All other APIs use the handle provided by this function, and therefore this function must be called before calling any other API.

PROTOTYPE

int ESAPCIDIOT_Open(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

RETURN VALUE

Returns 0 on success,

1 if CardNo is not matching with the Existing **ESAPCIDOT48** cards.

2 if No **ESA PCIDIOT** card existing.

EXAMPLE

```
int  dwStatus;
dwStatus = ESAPCIDIOT_Open();
if (dwStatus == 2) {
        Message Box(NULL,” No ESA PCIDIOT Cards Found”,”ERROR”,NULL);
        Exit(0);
}
if (dwStatus == 1) {
        Message Box(NULL,” Card No not matching with the existing
                        cards”,”ERROR”,NULL);
        Exit(0);
}
if (dwStatus ==  0) {
        Message Box(NULL,” Card Found”,” INFO…”,NULL);
        …………..
}
```

2) ESAPCIDIOT_Close()

PURPOSE

Closes the Device Handle and frees resources allocated for the Device which was created by **ESAPCIDIOT_Open()** at start.

PROTOTYPE

void ESAPCIDIOT_Close(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

RETURN VALUE

None

EXAMPLE

ESAPCIDIOT_Close(1);

3) Write_82551CR()

PURPOSE

Writes the data to the 8255-1 Command Register of **ESA PCIDIOT** Hardware.

PROTOTYPE

```
void Write_82551CR(unsigned char CardNumber, unsigned char Data)
```

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

```
Write_82551CR(1,0x80);
```

4) Write_82551PortA()

PURPOSE

Writes the data to the 8255-1 PortA of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_82551PortA(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

Write_82551PortA(1,0x80);

5) Write_82551PortB()

PURPOSE

Writes the data to the 8255-1 PortB of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_82551PortB(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

Write_82551PortB(1,0x80);

6) Write_82551PortC()

PURPOSE

Writes the data to the 8255-1 PortC of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_82551PortC(unsigned char CardNumber, unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

Write_82551PortC(1,0x80);

7) Read_82551PortA()

PURPOSE

Reads the data from the 8255-1 PortA of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_82551PortA(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8255-1 PortA of ESA PCIDIOT Hardware.

EXAMPLE

Data = Read_82551PortA(1);

8) Read_82551PortB()

PURPOSE

Reads the data from the 8255-1 PortB of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_82551PortB(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8255-1 PortB of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_82551PortB(1);

9) Read_82551PortC()

PURPOSE

Reads the data from the 8255-1 PortC of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_82551PortC(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8255-1 PortC of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_82551PortC(1);

10) Write_82552CR()

PURPOSE

Writes the data to the 8255-2 Command Register of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_82552CR(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

Write_82552CR(1,0x80);

11) Write_82552PortA()

PURPOSE

Writes the data to the 8255-2 PortA of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_82552PortA(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

Write_82552PortA(1,0x80);

12) Write_82552PortB()

PURPOSE

Writes the data to the 8255-2 PortB of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_82552PortB(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

Write_82552PortB(1,0x80);

13) Write_82552PortC()

PURPOSE

Writes the data to the 8255-2 PortC of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_82552PortC(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the ports.

RETURN VALUE

NONE

EXAMPLE

Write_82552PortC(1,0x80);

14) Read_82552PortA()

PURPOSE

Reads the data from the 8255-2 PortA of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_82552PortA(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8255-2 PortA of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_82552PortA(1);

15) Read_82552PortB()

PURPOSE

Reads the data from the 8255-2 PortB of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_82552PortB(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8255-2 PortB of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_82552PortB(1);

16) Read_82552PortC()

PURPOSE

Reads the data from the 8255-2 PortC of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_82552PortC(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8255-2 PortC of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_82552PortC(1);

17) Write_8254CR()

PURPOSE

Writes the data to the 8254 Command Register of **ESA PCIDIOT** Hardware.

PROTOTYPE

```
void Write_8254CR(unsigned char CardNumber,unsigned char Data)
```

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on the command register.

RETURN VALUE

NONE

EXAMPLE

```
Write_8254CR(1,0x30);
```

18) Latch_Timer()

PURPOSE

Reads the data from specified 8254 timers on the fly of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned short Latch_Timer(unsigned char CardNumber, unsigned char Timerno)

PARAMETERS

Name	Type	Input/Output
Timerno	unsigned char	Input 0 --- Timer 0 1 --- Timer 1 2 --- Timer 2

RETURN VALUE

Returns the data read from the specified 8254 Timer of **ESA PCIDIOT** Hardware.

EXAMPLE

```
unsigned short  Data;
Data = Latch_Timer(1,0);
```

19) Write_Timer0()

PURPOSE

Writes the data to the 8254 Timer0 of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_Timer0(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on Timer0.

RETURN VALUE

NONE

EXAMPLE

Write_Timer0(1,0x80);

20) Write_Timer1()

PURPOSE

Writes the data to the 8254 Timer1 of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_Timer1(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on Timer1.

RETURN VALUE

NONE

EXAMPLE

Write_Timer1(1,0x80);

21) Write_Timer2()

PURPOSE

Writes the data to the 8254 Timer2 of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_Timer2(unsigned char CardNumber, unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written on Timer2.

RETURN VALUE

NONE

EXAMPLE

Write_Timer2(1,0x80);

22) Read_Timer0()

PURPOSE

Reads the data from the 8254 Timer0 of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_Timer0(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8254 Timer0 of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_Timer0(1);

23) Read_Timer1()

PURPOSE

Reads the data from the 8254 Timer1 of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_Timer1(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8254 Timer1 of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_Timer1(1);

24) Read_Timer2()

PURPOSE

Reads the data from the 8254 Timer2 of **ESA PCIDIOT** Hardware.

PROTOTYPE

unsigned char Read_Timer2(unsigned char CardNumber)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.

RETURN VALUE

Returns the data read from the 8254 Timer2 of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = Read_Timer2(1);

25) outportb()

PURPOSE

Write the BYTE data to the specified address. This address should be in the address range of Selected Card resources. Use “Chkdiot” utility to know the Card resources.

PROTOTYPE

void outportb(unsigned char CardNumber, unsigned int PortAddr, unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	Unsigned char	Input
PortAddr	Unsigned int	Input
Data	Unsigned Char	Input

RETURN VALUE

NONE

EXAMPLE

Outportb(1,0xd803, 0x80);

26) inportb()

PURPOSE

Reads the BYTE data from the specified address. This address should be in the address range of Selected Card resources. Use “Chkdiot” utility to know the Card resources.

PROTOTYPE

unsigned char inportb(unsigned char CardNumber, unsigned int PortAddr)

PARAMETERS

Name	Type	Input/Output
CardNumber	Unsigned char	Input
PortAddr	Unsigned int	Input

RETURN VALUE

Returns the data read from the specified address of **ESA PCIDIOT** Hardware.

EXAMPLE

Data = inportb(1,0x8000);

27) IntEnable()

PURPOSE

Enables the interrupt on ESAPCIDIOT card.

PROTOTYPE

void IntEnable(unsigned char CardNumber, P9050_INT_HANDLER
funcIntHnadler,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
funcIntHandler	P9050_INT_HANDLER	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
funcIntHandler	Interrupt service routine to be executed, when interrupt occurs.
Data	Used for specifying to enable interrupt 1 or 2

RETURN VALUE

NONE

EXAMPLE

IntEnable(1, IntrRoutine,1);   //Enabling Interrupt 1 LINT1

28) IntDisable()

PURPOSE

Disbales the interrupts on ESAPCIDIOT Card.

PROTOTYPE

```
void IntDisable(void);
```

PARAMETERS

None

RETURN VALUE

None

EXAMPLE

```
IntDisable();
```

29) Write_INTCSR()

PURPOSE

Writes the data to the Interrupt control and status register of **ESA PCIDIOT** Hardware.

PROTOTYPE

void Write_INTCSR(unsigned char CardNumber,unsigned char Data)

PARAMETERS

Name	Type	Input/Output
CardNumber	unsigned char	Input
Data	unsigned char	Input

DESCRIPTION

Name	Description
CardNumber	Used for specifying the number of the card, on which the operation to be done.
Data	Used for specifying the data to be written to INTCSR register.

RETURN VALUE

NONE

EXAMPLE

Write_INTCSR(1,0x43);

5.0 APPLICATION DEVELOPMENT USING DRIVER LIBRARIES:

Section I describes about the application development in **VC++ 6.0.**

Section II describes about the application development in **VB 6.0.**

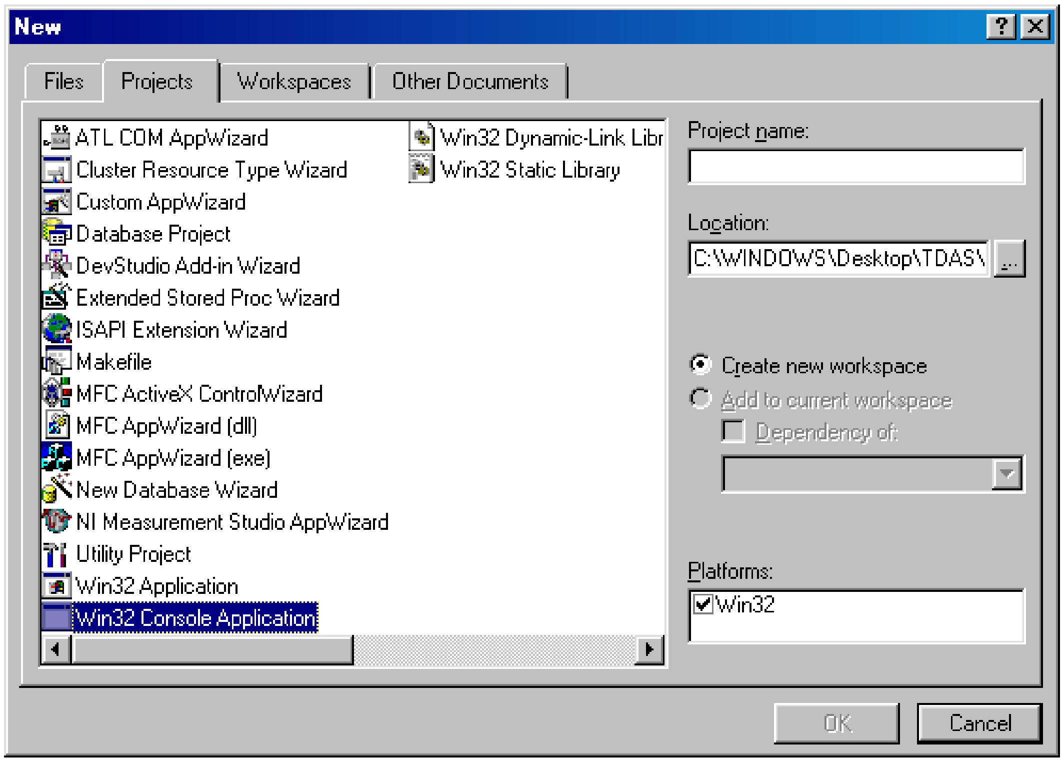
Section III describes about the application development in **Lab windows/CVI.**

Section IV describes about the application development in **DOS** environment using **Turbo C compiler**(User can use any 16-bit native compilers).

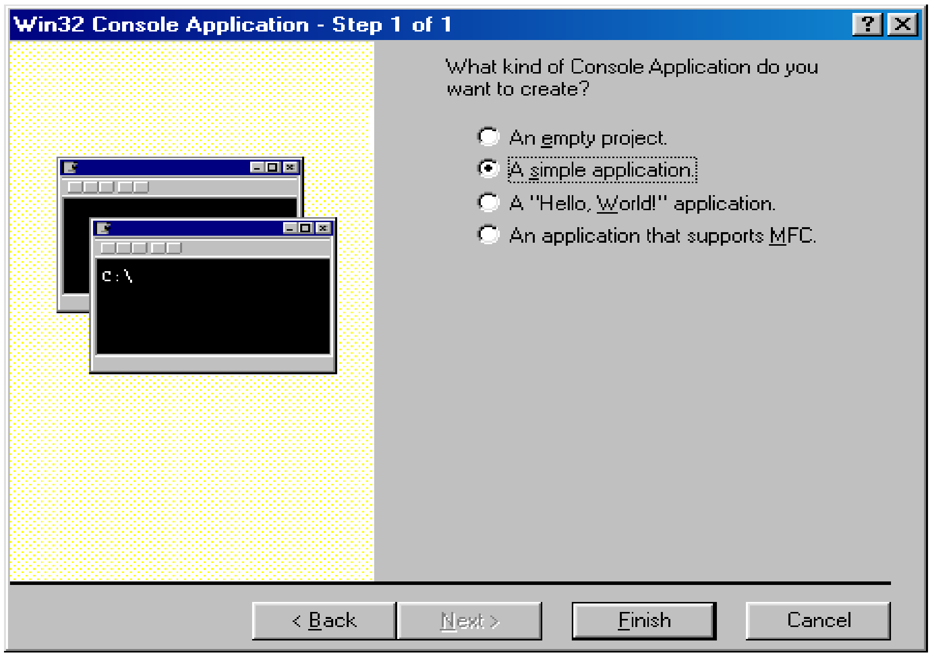
5.1) Visual C++ 6.0 (VC++)

Creating a New Console Application Project in Visual C++ 6.0:

1. Start the Microsoft Developer Studio.
2. Choose **New** from the **File** Menu.
3. Select **Projects** Tab. You should then see the following Dialog Box.



4. Enter the Project name and location where project-working folder should be created.
5. Click **OK** button.
6. You should then see the following Dialog Box.



What kind of Console Application do you want to create?

- An empty project.
- [✓ A simple application.]
- A "Hello, World!" application.
- An application that supports MFC.

7. Select the Simple Application and click **Finish**.

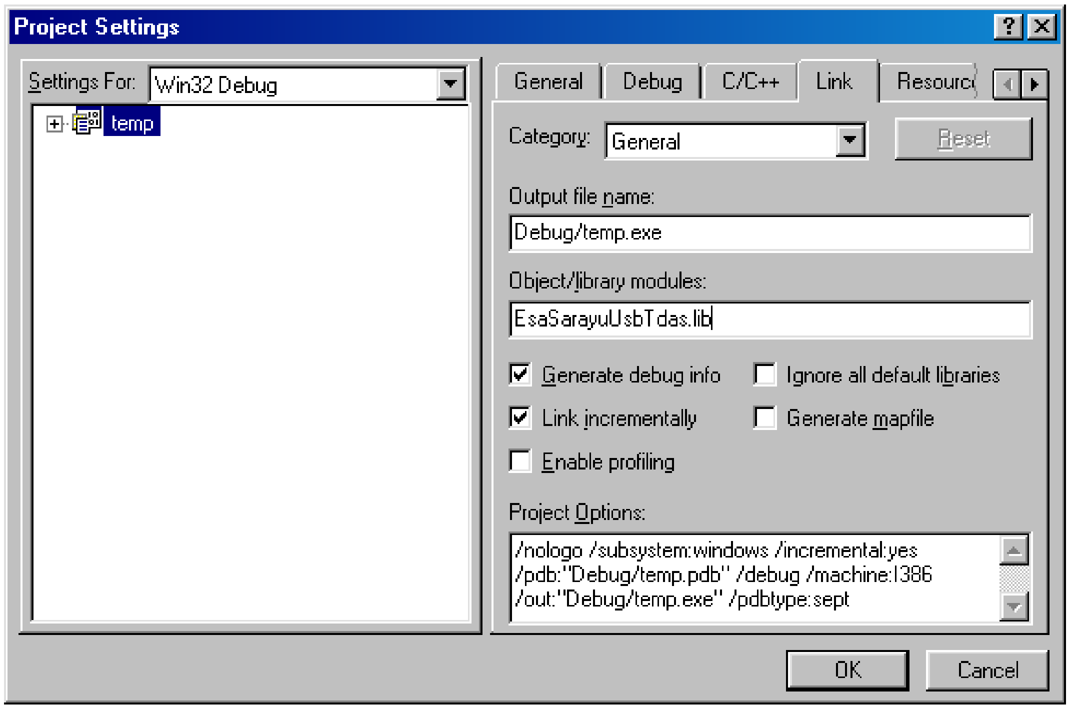
8. Copy all Files from **Visual C_C++_LABWINDOWS_LIB** folder available in Drivers CD to current working

   directory.

9. Open the Application cpp file and add “#include “esapdiot.h” and

   write the application using the Driver Libraries.

11.Select **Project** -> **Settings**. You should then see the following dialog box.



Settings For: Win32 Debug

Category: General

Output file name: Debug/temp.exe

Object/library modules: EsaSarayuUsbTdas.lib

[✓] Generate debug info
[ ] Ignore all default libraries
[✓] Link incrementally
[ ] Generate mapfile
[ ] Enable profiling

Project Options:

```
/nologo /subsystem:windows /incremental:yes
/pdb:"Debug/temp.pdb" /debug /machine:I386
/out:"Debug/temp.exe" /pdbtype:sept
```

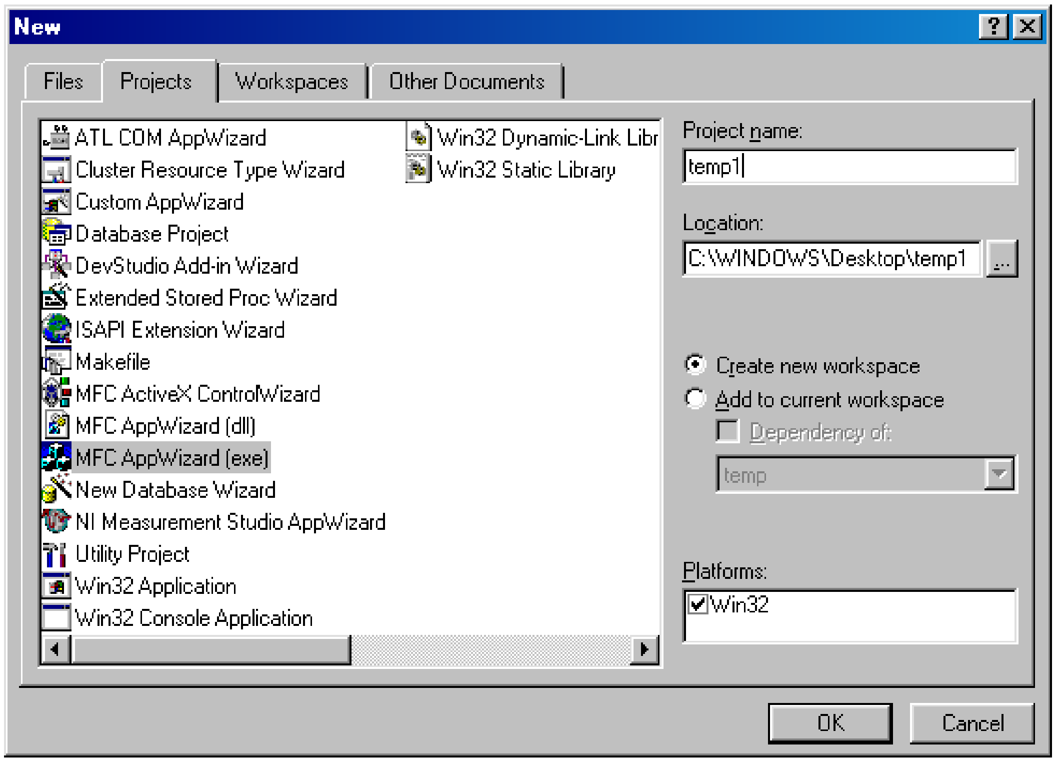
13. Select **Link** Tab in the Dialog box.
14. Specify **“Esapdiot.lib”** at **Object/library modules** Textbox.
15. Click **OK** button.
16. Build the Application From **Build** Menu.
17. Run the Application.

Example:

```
        #include  <stdafx.h>
        #include  “Esapdiot.h”
        int main(void)
        {
                unsigned int dwError;
                dwError = ESAPCIDIOT_Open();
                Write_82551CR(0x80);
                Write_82552CR(0x9b);
                while(!kbhit()) {
                        Write_82551PortA(0x55);
                        Write_82551PortB(0xAA);
                        Write_82551PortB(0xFF);
                        if (Read_82552PortA() == 0x55)
                                printf(“\r\n PortA Good”);
                        Else
                                Printf(“\r\n PortA Bad”);
                        if (Read_82552PortB() == 0x55)
                                printf(“\r\n PortB Good”);
                        Else
                                Printf(“\r\n PortB Bad”);
                        if (Read_82552PortC() == 0x55)
                                printf(“\r\n PortC Good”);
                        Else
                                Printf(“\r\n PortC Bad”);
                }
                ESAPCIDIOT_Close();
                return 0;
}
```

Creating a MFC Application Project in Visual C++ 6.0:

1. Start the Microsoft Developer Studio.
2. Choose **New** from the **File** Menu.
3. Select **Projects** Tab. You should then see the following Dialog Box.



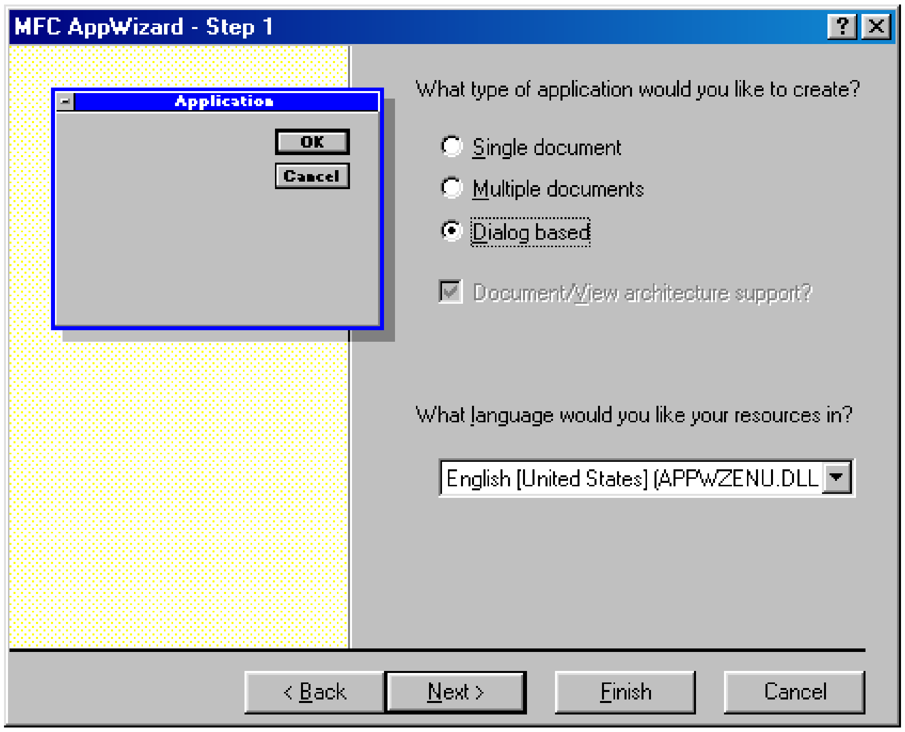
Project name: temp1

Location: C:\WINDOWS\Desktop\temp1

[✓ Create new workspace] Add to current workspace

Platforms: [✓ Win32]

4. Enter the Project name and location where project-working folder should be created.
5. Click **OK** button.
6. You should then see the following Dialog Box.



What type of application would you like to create?

( ) Single document
( ) Multiple documents
(●) Dialog based

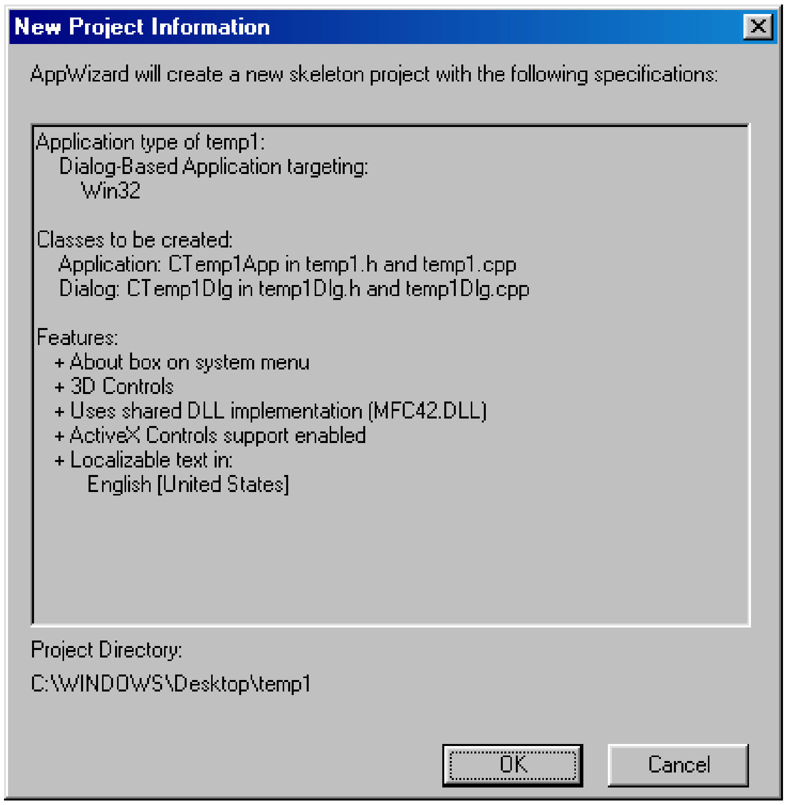
[✓] Document/View architecture support?

What language would you like your resources in?

English [United States] (APPWZENU.DLL

7) Select **Dialog based** radio button and click Finish.

8) You should then see the following Dialog box with added classes Information.



AppWizard will create a new skeleton project with the following specifications:

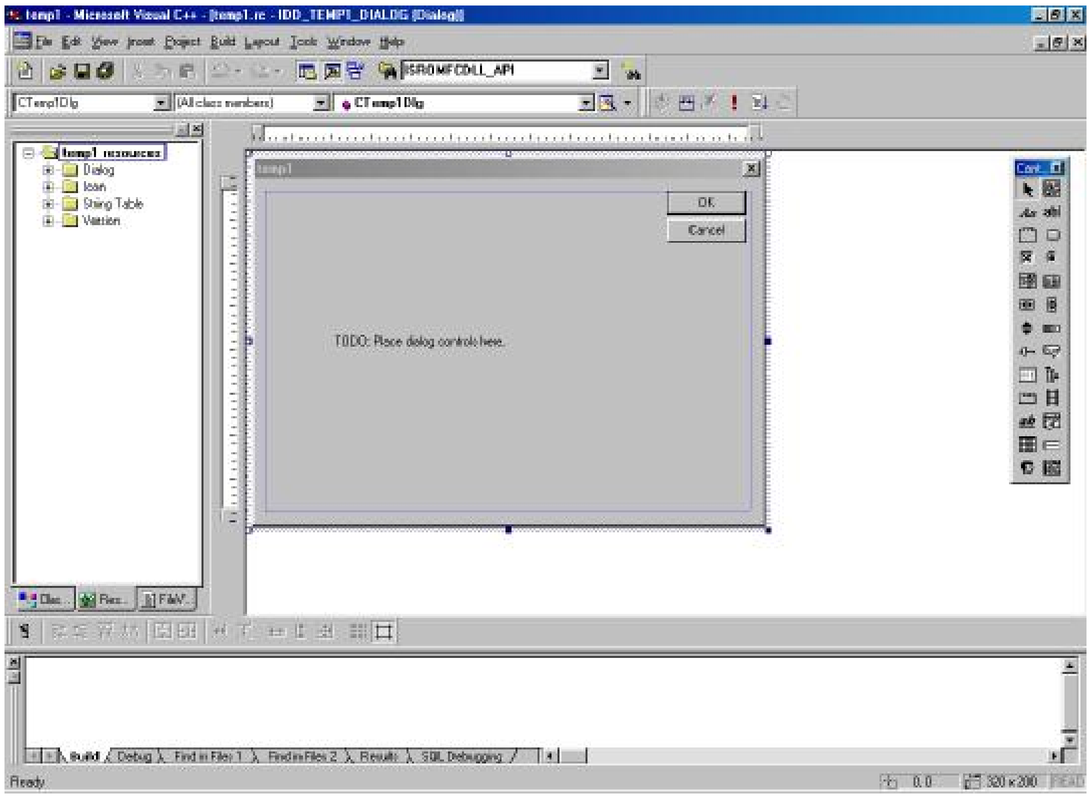
```
Application type of temp1:
    Dialog-Based Application targeting:
        Win32

Classes to be created:
    Application: CTemp1App in temp1.h and temp1.cpp
    Dialog: CTemp1Dlg in temp1Dlg.h and temp1Dlg.cpp

Features:
   + About box on system menu
   + 3D Controls
   + Uses shared DLL implementation (MFC42.DLL)
   + ActiveX Controls support enabled
   + Localizable text in:
         English [United States]
```

Project Directory:

C:\WINDOWS\Desktop\temp1

9) Click **OK** Button.

10) You should then see the following windows.



11) Add the controls as per requirement.

12) Add Callback functions for the controls by using the driver libraries.

13) Copy all Files from **Visual C_C++_LABWINDOWS_LIB** folder, which is available in Drivers CD.

14) Select **Project** -> **Settings**. Select **Link** Tab in the Dialog box.

15) Specify **“Esapdiot.lib”** at **Object/library modules** Textbox.

16) Click **OK** button.

17) Build the application from **Build** Menu.

18) Run the application.

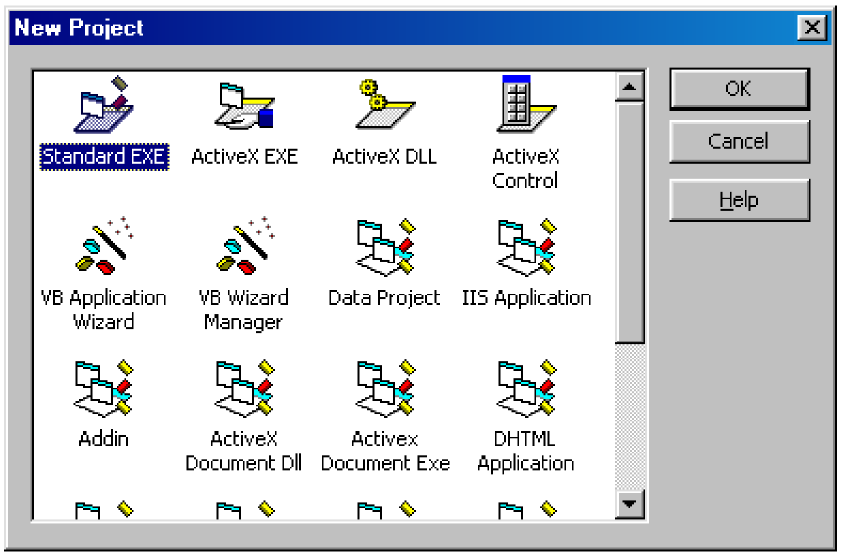
Example Source is given in the Driver software CD under ExampleApp.Source Folder.

5.2) Visual Basic 6.0 (VB)

Creating a Application Project in Visual BASIC 6.0:

1) Start Microsoft Visual Basic 6.0 environment.

2) Choose **New Project** from the **File** menu. You should then see the following dialog.



3) Select **Standard EXE** and click OK Button.

4) copy all files from **Visual Basic_LIB** folder to the current project directory.

5) Add "**esapdiot.bas**" module to the current project by right clicking on the project window -> add module option.

6) Place the controls(command button, text boxes etc..,) in the form.

7)Use the Driver Libraries as per the application requirement.

8) Build the Project

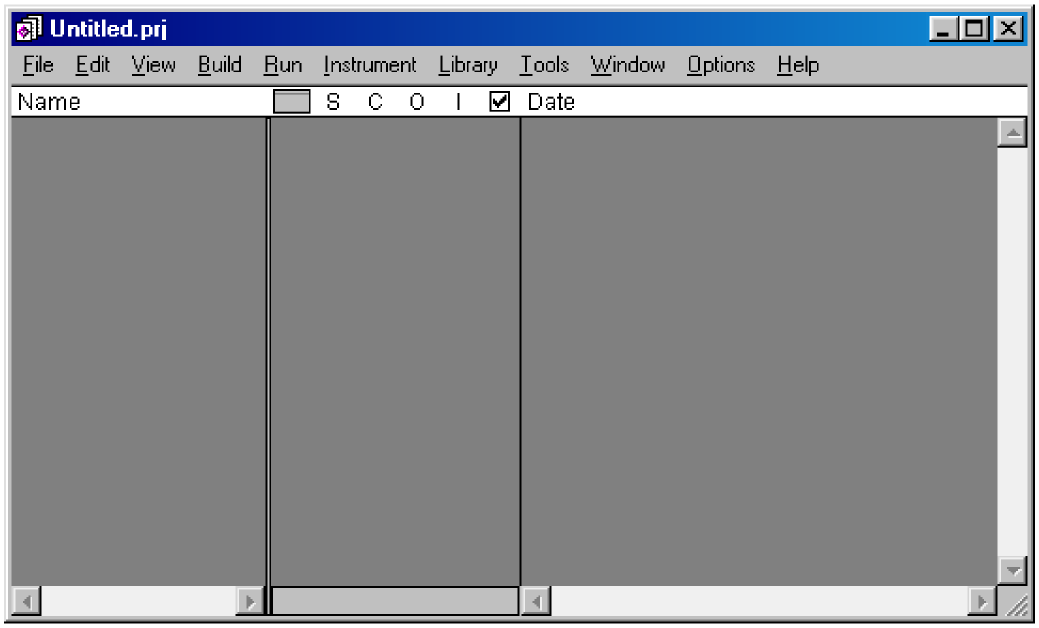
9) Run the application.

Example Source is given in the Driver software CD under ExampleApp.Source Folder.

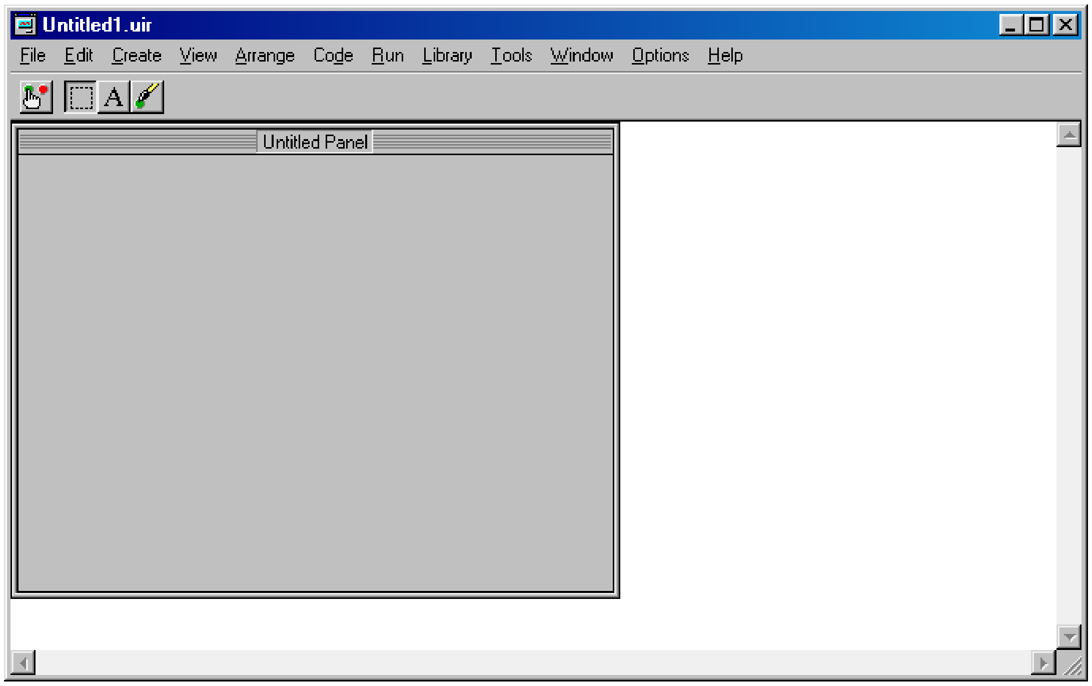
5.3) Lab Windows/CVI 6.0

Creating Project in Lab Windows/CVI 6.0:

1. Start the Lab Windows/CVI 6.0.
2. Create a new project in Lab Windows/CVI 6.0.
3. You should then see the following window.


Untitled.prj
File Edit View Build Run Instrument Library Tools Window Options Help
Name S C O I Date

4. Save the project.
5. Copy all Files from the **Visual C_C++_LABWINDOWS_LIB** Folder, which are available on Drivers CD to the current Project Directory.
6. Create a new uir file from **File** Menu.
7. You should then see the following window.


Untitled1.uir
File Edit Create View Arrange Code Run Library Tools Window Options Help
Untitled Panel

8. Right Click on the Panel and add the controls required. Add callbacks to the controls.
9. Create new “C” source file and write the code using driver libraries.
10. Add .uir, .c & .lib files to the project from the **Edit** Menu.
11. Build the Application.
12. Run the Application.

Example Source is given in Drivers CD under “ExampleApp.Source” Folder.

5.4) DOS Application Development (TURBO C & MASM)

NOTE: User can write his applications using inportb & outportb function with TURBO C Compiler (or) using IN & OUT instruction with MASM under windows 98 & 95. But coming to Windows NT, 2000, XP the user mode application does not have the read/write permissions of the I/O ports. User need to use either Visual C++, Visual Basic, Labwindows or MASM32 development environment to access the I/O port under the control of device driver supplied by ESA. Please refer the below description for further understanding.

A problem that plagues Windows NT/2000 and Windows XP is it's strict control over I/O ports. Unlike Windows 9x & ME, Windows NT/2000/XP will cause an exception (Privileged Instruction) if an attempt is made to access an IO port that a user mode program is not privileged to talk too. Actually it's not Windows NT that does this, but any 386 or higher processor running in protected mode.

Accessing I/O Ports in protected mode is governed by two events, The I/O privilege level (IOPL) in the EFLAGS register and the I/O permission bit map of a Task State Segment (TSS). Under Windows NT, there are only two I/O privilege levels used, level 0 & level 3. User mode programs will run in privilege level 3, while device drivers and the kernel will run in privilege level 0, commonly referred to as ring 0. This allows the trusted operating system and drivers running in kernel mode to access the ports, while preventing less trusted user mode processes from touching the I/O ports and causing conflicts. All user mode programs should talk to a device driver, which arbitrates access. The I/O permission bitmap can be used to allow programs not privileged enough (I.e. user mode programs) the ability to access certain I/O ports. When an I/O instruction is executed, the processors will first check if the task is privileged enough to access the ports. Should this be the case, the I/O instruction will be executed. However if the task is not allowed to do I/O, the processor will then check the I/O permission bitmap. The I/O permission bitmap, as the name suggests uses a single bit to represent each I/O address. If the bit corresponding to a port is set, then the instruction will generate an exception however if the bit is clear then the I/O operation will proceed. This gives a means to allow certain processes to access certain ports. There is one I/O permission bitmap per task.

Creating Application in TURBOC:

1) Run the "Chkdiot" utility from Drivers CD to know the **ESA PCIDOT** Card resources. This utility lists the
   i) 8255-1 & 8255-2 Command Register, PortA, PortB & PortC Address
   ii) 8254 Timer Command Register, Timer0, Timer1 & Timer2 Address.
   iii) Number of **ESA PCIDIOT** Cards Existing.

2) Open the Turbo C editor and create a new file. Use the listed addresses of the card resources with inportb() & outportb() libraries, which is available under "dos.h".

EXAMPLE:

```
#include <stdio.h>
#include <dos.h>

void main(void)
{
        // Make 8255-1 all port outports
        outportb(0xd803,0x80);
        // Make 8255-2 all port imports
        outportb(0xdc03,0x9b);

        while (!kbhit())
        {
                outportb(0xd800,0x55);
                if (inportb(0xdc00) == 0x55)
                        printf("PortA Good");
        }
}
```

More Examples was given in the **ESA PCIDIOT** Drivers CD.

MASM Examples are also given in the CD.

5.5) MASM32 Application Development

Note:- MASM32 installation pack can be download from the web. It is not in the scope of our supply.

Copy files from MASM32_LIB folder to current working directory.

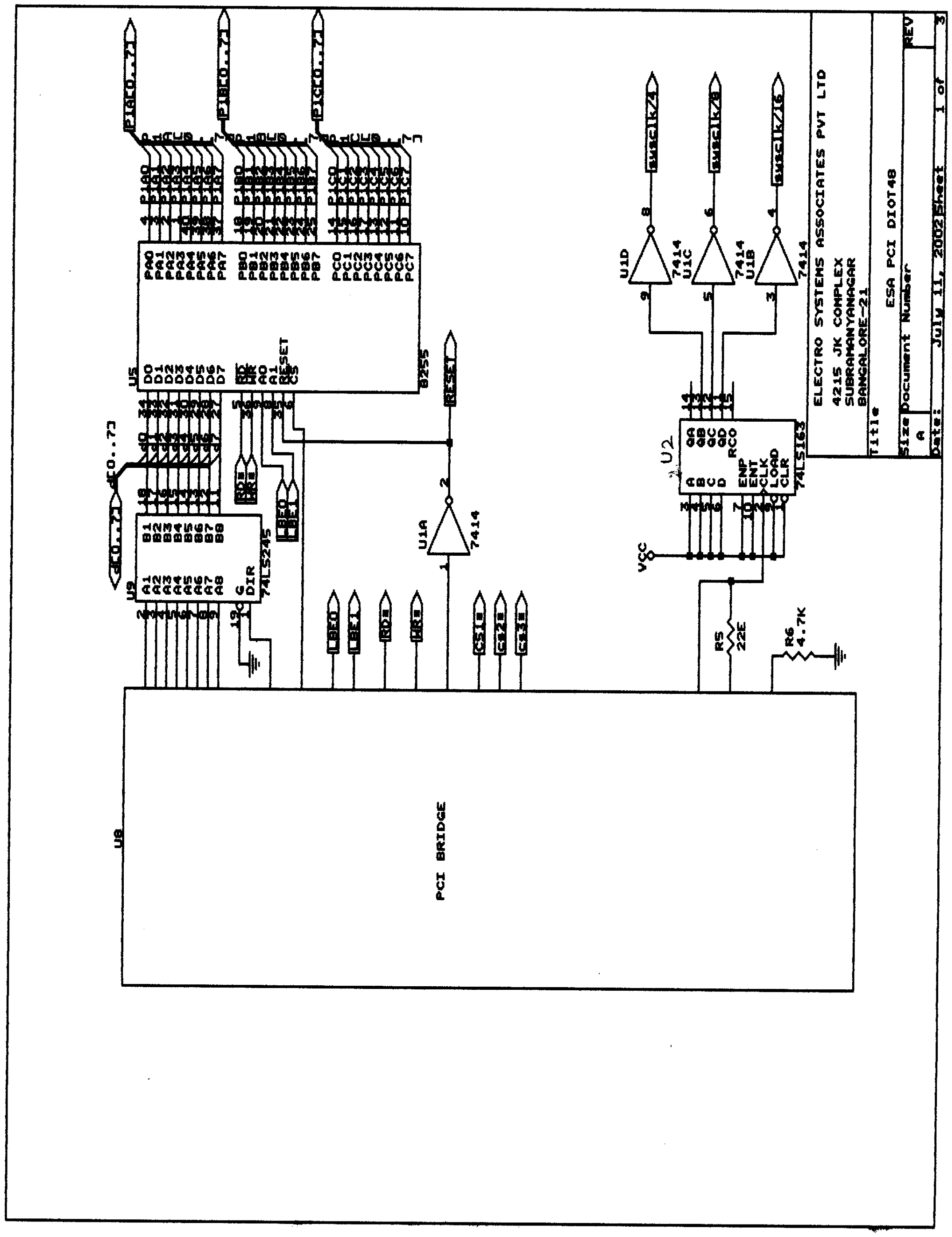
Include esadiot.inc to your application.

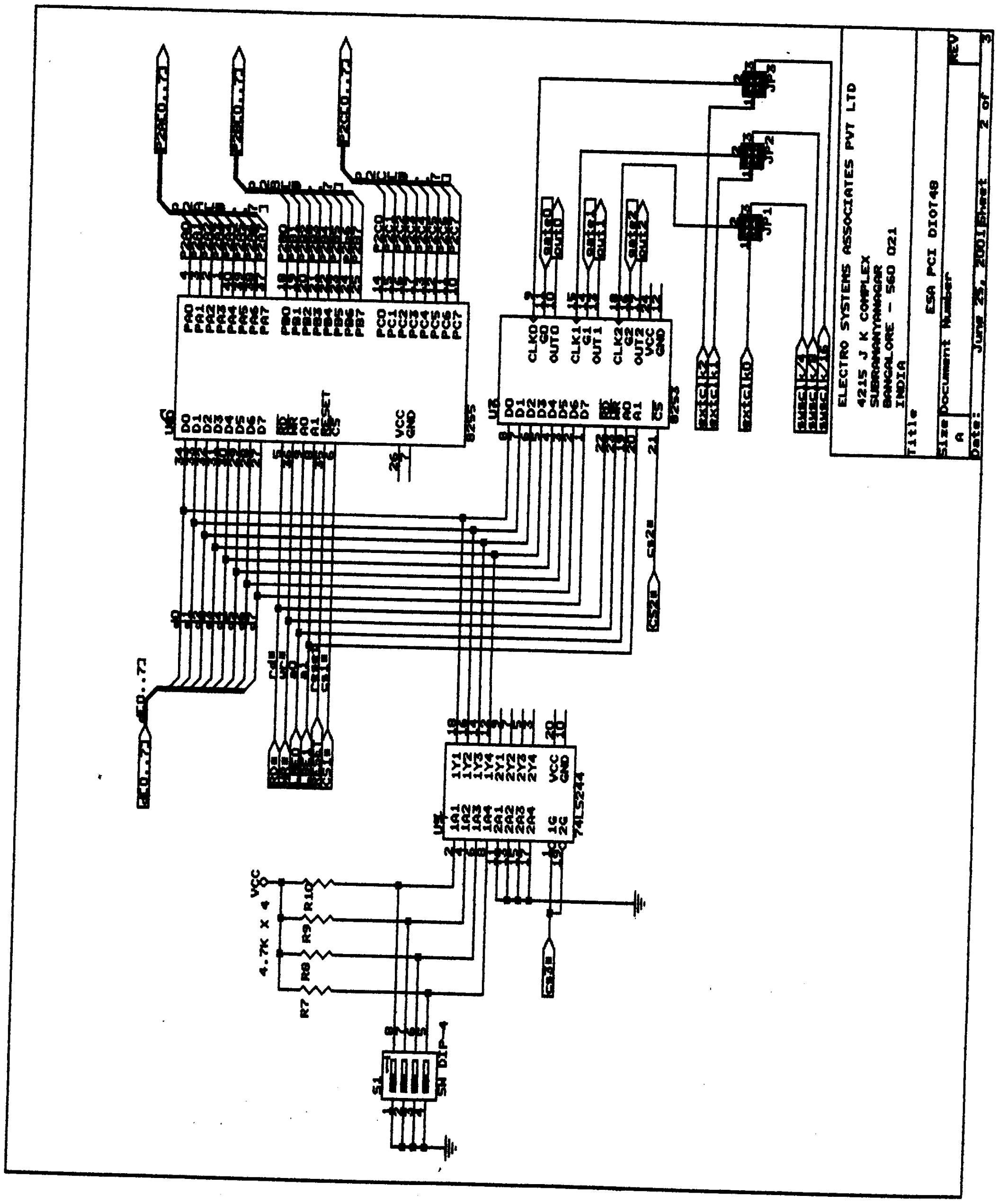
Write the application using the libraries included in “esadiot.inc”

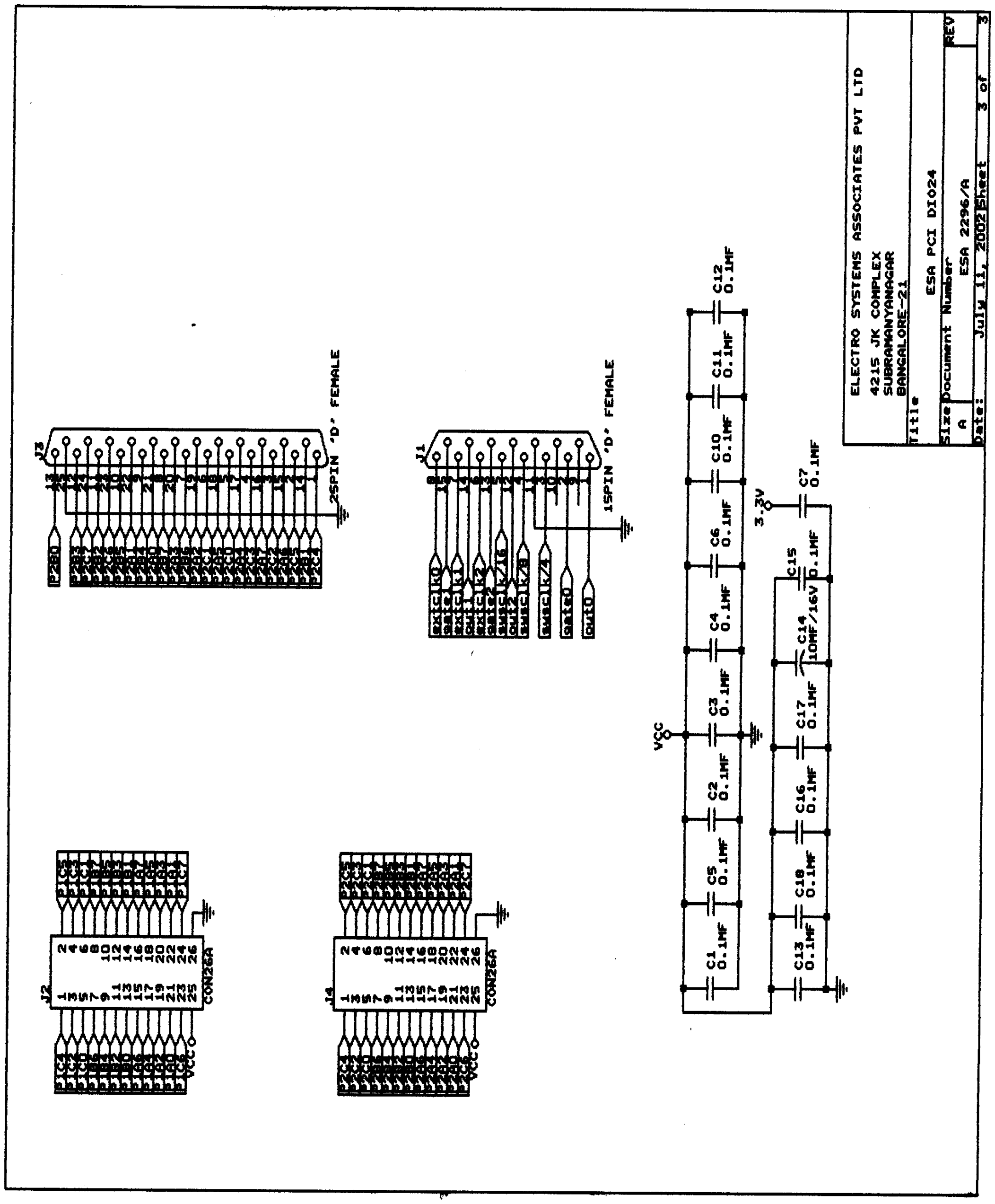
Please refer the examples given in MASM32 folder of Driver CD. “Makeit.bat” contains the assembler and linker commands.

Appendix A

Schematics

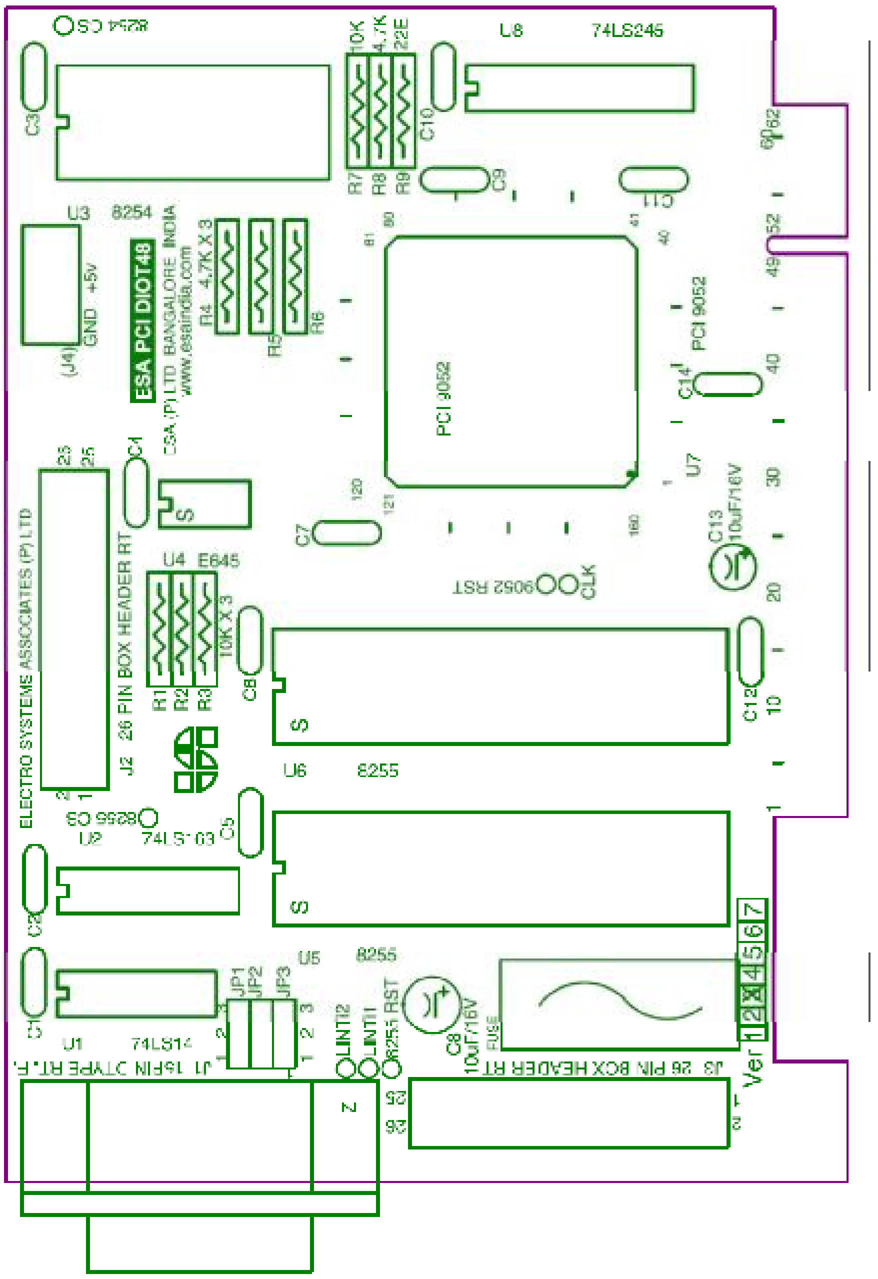

ELECTRO SYSTEMS ASSOCIATES PVT LTD
4215 JK COMPLEX
SUBRAMANYANAGAR
BANGALORE-21
Title
ESA PCI DIOT48
Size A
Document Number
REV
Date: July 11, 2002
Sheet 1 of 3
PCI BRIDGE
U8
U5
8255
U9
74LS245
U2
74LS163
U1A
U1B
U1C
U1D
7414
R5 22E
R6 4.7K
VCC
RESET
LBE0
LBE1
RD#
WR#
CS1#
CS2#
CS3#
P1A[0..7]
P1B[0..7]
P1C[0..7]
sysclk/4
sysclk/8
sysclk/16
dC[0..7]


ELECTRO SYSTEMS ASSOCIATES PVT LTD
4215 J K COMPLEX
SUBRAMANYANAGAR
BANGALORE - 560 021
INDIA
Title
ESA PCI DIO748
Size
A
Document Number
REV
Date:
June 25, 2001
Sheet 2 of 3
8255
8253
74LS244
SW DIP-4
4.7K X 4
VCC
R7
R8
R9
R10
S1
JP1
JP2
JP3



Appendix B

Component Layout



Appendix C

Connector Details

J1-15 Pin Connector D-Type used by 8254 Timer:

Signal	**25-Pin Female connector**
OUT 0	1
GATE 0	2
SYSCLK/4	3
SYSCLK/8	4
SYSCLK/16	5
EXTCLK2	6
EXTCLK1	7
EXTCLK0	8
NC	9
NC	10
GND	11
OUT 2	12
GATE 2	13
OUT1	14
GATE 1	15

J2 – 26 Berg Connector:

Signal	8255 PIN(U5)	26PIN Berg Connector
P1A0	4	21
P1A1	3	22
P1A2	2	19
P1A3	1	20
P1A4	40	17
P1A5	39	18
P1A6	38	15
P1A7	37	16
P1B0	18	13
P1B1	19	14
P1B2	20	11
P1B3	21	12
P1B4	22	9
P1B5	23	10
P1B6	24	7
P1B7	25	8
P1C0	14	5
P1C1	15	6
P1C2	16	3
P1C3	17	4
P1C4	13	1
P1C5	12	2
P1C6	11	23
P1C7	10	24
VCC	26	25
GND	7	26