3.9 GPIOLed 数码管程序

3.9.1 实验目的

1．熟悉 Embedded Visual C++集成开发环境以及相关配置。

2．利用 Embedded Visual C++编写一个 GPIO 的 LED 应用程序。

3.9.2 实验内容

参照本实验指导书的步骤，编写一个通过 GPIO 控制 LED 的应用程序。

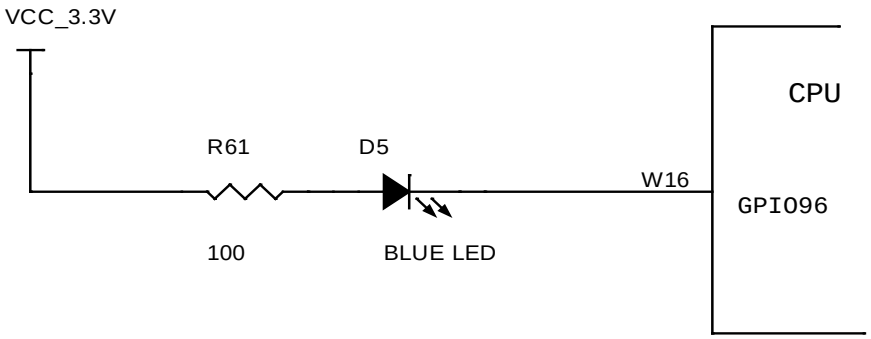
3.9.3 实验设备

1．OURS-PXA270-EP 实验仪，烧录有 WINCE 的 Flash，交叉网线及 USB 连接线。

2．PC 操作系统，Embedded Visual C++集成开发环境。

3.9.4 实验原理及说明

理解本实验的硬件原理图。



凡是操作系统控制外部设备，即使是最简单的硬件电路，也是需要驱动的。本实验涉及的外部硬件只有电阻，蓝色发光二极管。我们使用自己编写的驱动程序与应用程序控制 GPIO96 的电平。通过 LED 的亮灭来判断，是否 CPU 做出了正确的响应。

相关的 CPU 寄存器有需要了解 GPDR, GPSR, GPCR, GPLR, GAFR。这部分的细节最好参考 CPU 的文档 Developers_Manual.pdf。

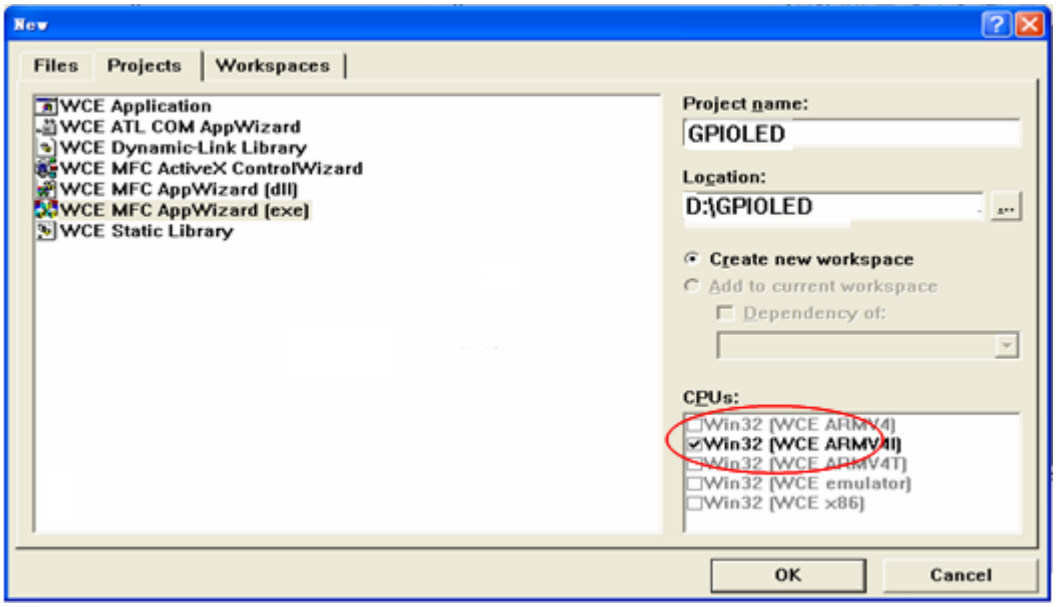
3.9.5 实验步骤

本实验的步骤和 LED 实验的步骤很类似，可以参考前面的实验步骤建立环境。

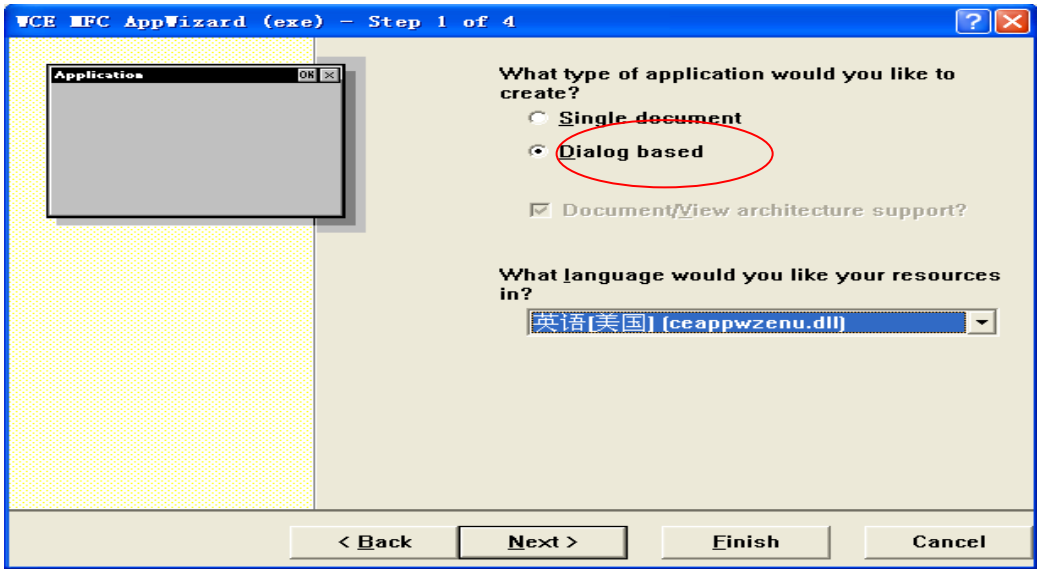
1．在 3.2 节定制好的平台基础上添加 GPIOLED 驱动，在 Third Party 中找到我们安装的 BSP 中的 GPIO LED 的驱动，将其加入内核，并编译。

2．根据 3.2 的方法，编译内核。

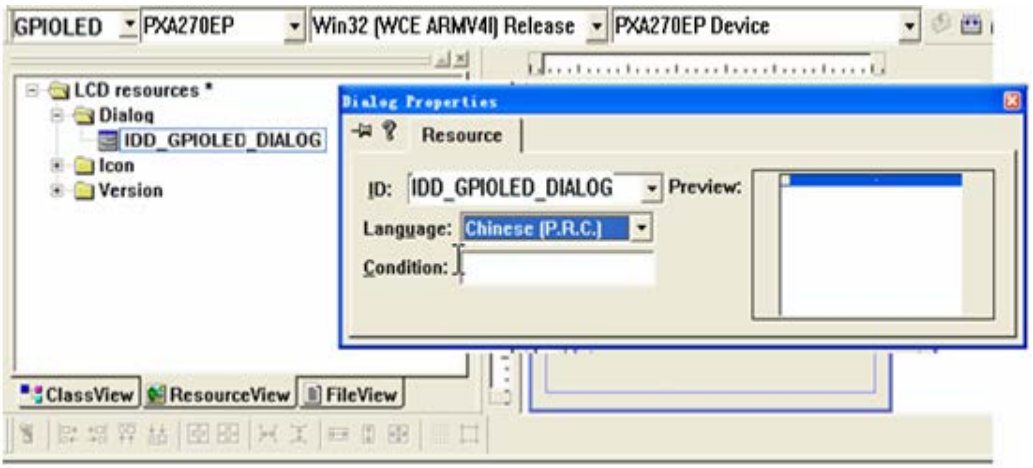
3．在机器已安装实验箱所附带的 SDK 的基础上，启动 MS eMbedded Visual C++ 4.0，利用 EVC 的工程建立向导，建立一个新的工程如下图所示，Project 里面我们选择 WCE MFC AppWizard[exe]，工程名字我们叫 GPIOLED，其中右下的 CPUs 选择 Win32（WCE ARMV4I），选择完成以后，点击 OK，进入下一个画面。之所以选择 Win32（WCE ARMV4I）是因为我们将在试验箱上运行编写的程序。如果要在 PC 机的模拟器上运行编写的程序，我们可以选择 WIN32 （WCE emulator）。



4．我们将要用对话框制作我们的程序，为此我们在下图的 What type of application would you like to create?选项里面选择 Dialog based，其他的选项为默认就可以。由于我们无需设置其他的选项，所以直接点击 Finish。



5．在 Resource View 里面的 IDD_GPIOLED_DIALOG 右键选择 properties，将弹出如下图所示对话框，Language 中选择 Chinese（P.R.C）。只有这样，在以后的对话框中写入中文才不会出现乱码的情况。



6．在 ID 为 IDD_GPIOLED_DIALOG 对话框上，编辑如下控件，包括三个：Caption 为“亮”的 button 控件，其 ID 为 IDC_BUTTON _ON；Caption 为“灭”的 button 控件，其 ID 为 IDC_BUTTON _OFF；Caption 为“闪烁”的 button 控件，其 ID 为 IDC_BUTTON _FLASH。

7．在 GPIOLEDDlg.h 中，添加如下成员变量：

public：

```
HANDLE m_hDev; //声明了一个句柄，用来接收 WriteFile 的返回值
BOOL    m_bValue; // m_bValue 是一个二进制的变量，用来控制 LED 的状态
DWORD m_dwBytes; //声明了一个双字类型的变量，用作 WriteFile 的参数
```

8．双击“亮”button 控件，为按钮添加消息响应函数 afx_msg void OnButtonOn (); 并在实现中添加如下代码：

```
void CGPIOLEDDlg::OnButtonOn()
{
    // TODO: Add your control notification handler code here
    //设置 LED 亮
    KillTimer(1);
    m_bValue = FALSE;
    WriteFile(m_hDev,&m_bValue,4,&m_dwBytes,NULL);
}
```

9．双击“灭”button 控件，为按钮添加消息响应函数 afx_msg void OnButtonOff (); 并在实现中添加如下代码：

```
void CGPIOLEDDlg::OnButtonOff()
{
    // TODO: Add your control notification handler code here
    //设置 LED 熄灭
    KillTimer(1);
    m_bValue = TRUE;
    WriteFile(m_hDev,&m_bValue,4,&m_dwBytes,NULL);
}
```

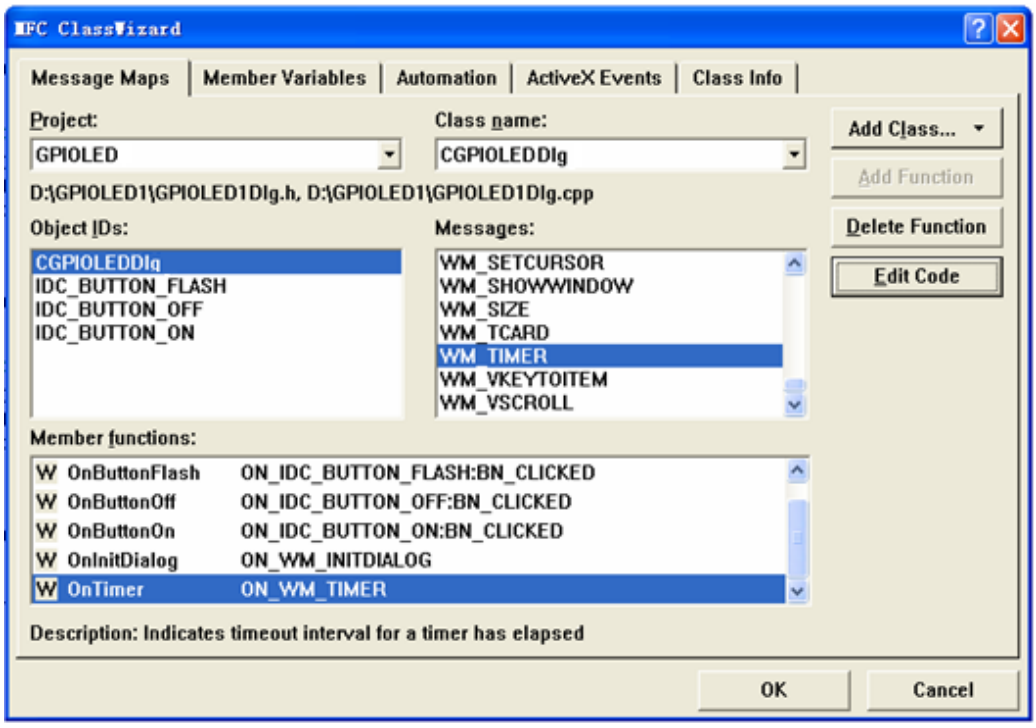
10．双击“闪烁”button 控件，为按钮添加消息响应函数 afx_msg void OnButtonFlash ();并在实现中添加如下代码：

```
void CGPIOLEDDlg::OnButtonFlash()
{
    // TODO: Add your control notification handler code here
    SetTimer(1,500,NULL);//  设置定时器
}
```

11. 使用 ClassWizard 为 CGPIOLEDDlg 加入 WM_TIMER 的消息响应函数 OnTimer()，如图所示，点击“Edit Code”按钮。

```
void CGPIOLEDDlg::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    //设置 LED 闪烁
    m_bValue = !m_bValue;
    WriteFile(m_hDev,&m_bValue,4,&m_dwBytes,NULL);
```

CDialog::OnTimer(nIDEvent);
}



12．这个地方还有个细节需要注意，在对话框初始化的时候，我们需要打开设备，并且做一个确认，如果失败，则弹出提示对话框。这部分程序代码添加在 OnInitDialog 中，具体代码如下：

```
// TODO: Add extra initialization here
  m_hDev = CreateFile(_T("GIO1:"),GENERIC_WRITE,0,NULL,
       OPEN_EXISTING,0,0);

  if(m_hDev == INVALID_HANDLE_VALUE)
  {
       AfxMessageBox(_T("打开设备失败"));
  }
```

13．在我们 EVC 环境下，选择 Build->Rebuild All，生成可执行文件。

14．接下来按照前面的实验 3.7 的第 12—15 步内容建立宿主机与实验箱的连接。

15．运行程序以后，可以通过选择亮灭闪烁控制按钮，控制灯的状态。