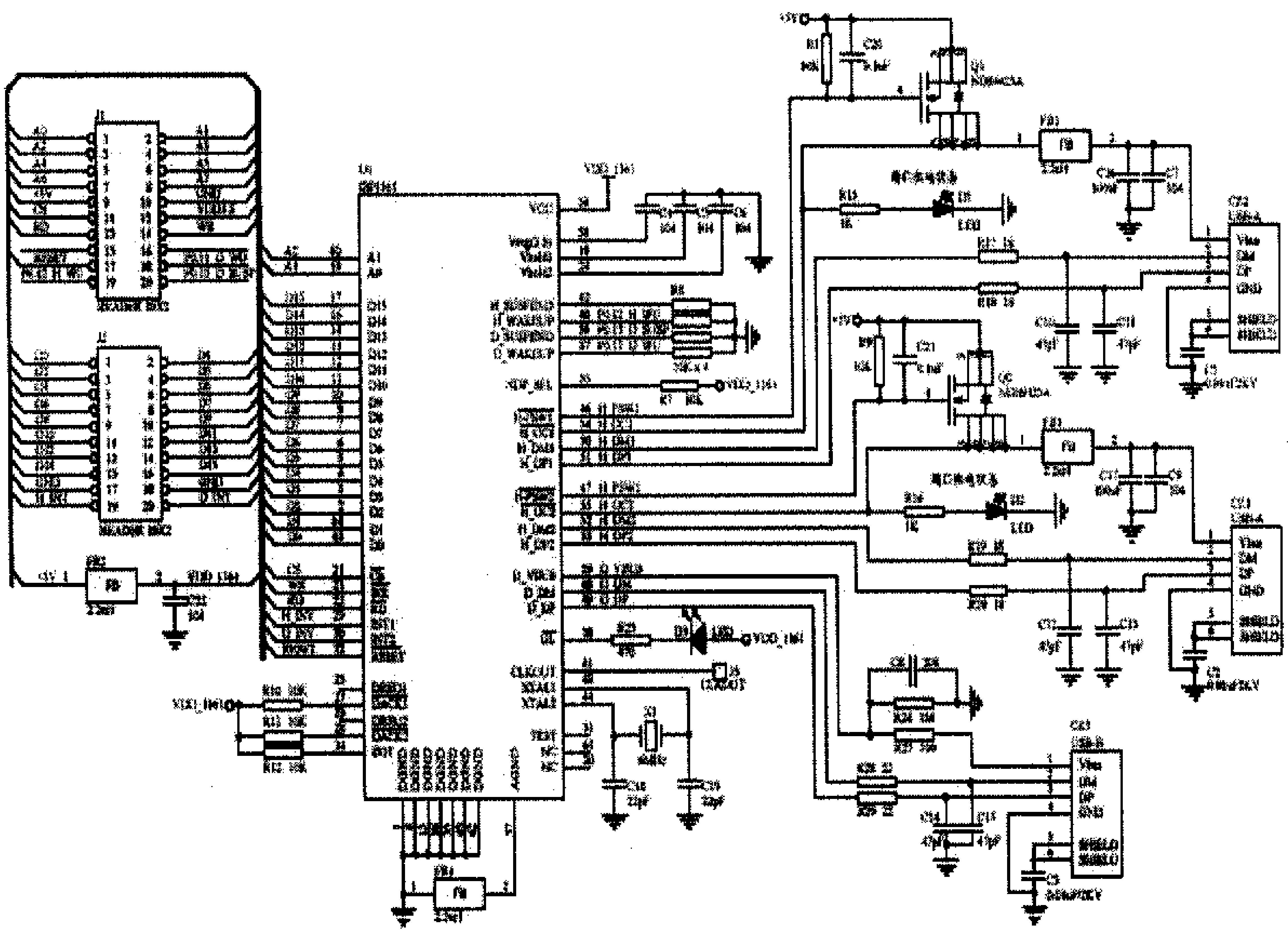

图 4-7 ISP1161 部分电路

4.7.3 设备驱动程序开发

按 4.6 节介绍的方法配置环境，选用 VC++编译器 Makefile 项目创建开发框架，它生成典型 WDM PnP 驱动程序处理例程，入口和卸载例程 FireInit.c,即插即用例程 FirePnp.c,分发例程 FireDispatch.c,电源管理例程 FirePower.c。下面给出入口和卸载例程 FireInit.c,即插即用例程 FirePnp.c 主要程序清单。

在入口和卸载例程中主要有 DriverEntry 例程（与 4 .2 节介绍的基本相同)、AddFireDevice 例程和 FireDrvUnload 例程。AddFireDevice 例程主要是创建保护器内核设备名和链接名并设置相应字段，程序清单如下：

```
    NTSTATUS AddFireDevice(IN PDRIVER_OBJECT DriverObject,IN PDEVICE_OBJECT
PhyscialDeviceObject)
{
    NTSTATUS ntStatus=STATUS_SUCCESS;
    WCHAR KernelDeviceNameBuffer[ ]=L"\\Device\\Phiusb-0";
    UNICODE_STRING KernelDeviceNameUnicode;
    WCHAR UserDeviceLinkBuffer[ ]=L"\\SybDevice\\Phiusb-0";
    UNICODE_STRING UserDeviceLinkUnicode;
    PDEVICE_OBJECT fdo = NULL;
    PDEVICE_EXTENSION pdx;
    RtlInitUnicodeString(&KernelDeviceNameUnicode, KernelDeviceNameBuffer);
    ntStatus=IoCreatDevice(DeviceObject,sizeof(DEVICE_EXTENSION),
                           &KernelDeviceNameUnicode,
                           FILE_DEVICE_UNKNOWN,0,FALSE,&fdo);
    if(!NT_SUCCESS(ntStatus)) return ntStatus;
    RtlInitUnicodeString(&UserDeviceLinkUnicode, UserDeviceLinkBuffer);
    ntStatus=IoCreatSymbolicLink(&UserDeviceLinkUnicode, KernelDeviceNameUnicode);
    pdx=( PDEVICE_EXTENSION)(fdo->DeviceExtension);
    RtlCopyMemory(pdx->DeviceLinkName,
                           UserDeviceLinkBuffer,sizeof(UserDeviceLinkBuffer));
    pdx->OpenHandles=0;
    pdx-> ConfigurationHandle=NULL;
    pdx-> DeviceDescriptor=NULL;
    pdx-> Interface=NULL;
    fdo->Flag & = ~ DO_DEVICE_INITIALIZING;
    fdo->Flag | = DO_DIRECT_IO;

    pdx-> PhyscialDeviceObject = PhyscialDeviceObject;
    pdx->LowerDeviceObject = IoAttachDeviceToDeviceStack(fdo,
                                                   PhyscialDeviceObject);
    Pdx->Usages = 1;
    KeInitializeEvent(&pdx->evRemove,NotificationEvent,FALSE);
    return ntStatus;
}
```

卸载例程可以什么都不作，卸载工作可由 Pnp 中 IRP_MN_REMOVE_DEVICE 请求来完成处理。

在插即用例程 FirePnp.c 中，主要包括 IRP_MJ_PNP 处理例程，该例程针对 IRP_MJ_PNP 请求（如 4.6 节列举）中的主要的 6 个次功能代码进行处理，其余的则沿设备栈传递至下层驱动程序。程序清单如下：

```
 NTSTATUS FirePnpIrp(IN PDEVICE_OBJECT fdo,IN PIRP Irp)
{
   NTSTATUS ntstatus = STATUS_SUCCESS;
   PIO_STACK_LOCATION IrpStack;
   PDEVICE_EXTENSION pdx = fdo->DeviceExtension;
   ULONG MinorFunction;
   if(!LockDevice(fdo))
       return Complete(Irp,STATUS_DELETE_PENDING,0);
   IrpStack = IoGetCurrentStackLocation(Irp);
   MinorFunction = IrpStack->MinorFunction;
   Switch(MinorFunction)
   {
      case   IRP_MN_START_DEVICE:
                 ntStatus = PnpHandleStartDevice(fdo,Irp);
                 break;
      case   IRP_MN_STOP_DEVICE:
                 ntStatus = PnpHandleStopDevice(fdo,Irp);
                 break;
     case   IRP_MN_REMOVE_DEVICE:
                 ntStatus = PnpHandleRemoveDevice(fdo,Irp);
                 break;
      case   IRP_MN_QUERY_STOP_DEVICE:
                 ntStatus = PnpHandleQueryStopDevice(fdo,Irp);
                 break;
      case   IRP_MN_CANCEL_STOP_DEVICE:
```

```
                    ntStatus = PnpHandleCancelStopDevice(fdo,Irp);
                    break;
        case   IRP_MN_QUERY_CAPABILITIES:
              {
                    PDEVICE_ CAPABILITIES pdc =
                              IrpStack->Paramrters.DeviceCapabilities. Capabilities;
                    if(pdc->Version < 1){
                                    ntStatus=PnPHandleDefault(fdo,Irp);
                                    break;
                     }
                    ntStack=ForwardAndWait(fdo,Irp);
                    if(NT_SUCCESS(ntStatus)){
                          pdc=IrpStack-> Paramrters.DeviceCapabilities. Capabilities;
                          pdc->SurpriseRemovealOK=TURE;
                    }
                    ntStatus = CompleteRequest(Irp,ntStatus,Irp->IoStatus.Information;
                    }
                    break;
                    default:
                       ntStatus = PnpHandleDefault(fdo,Irp);
             }
            if (MinorFunction ! = IRP_MN_REMOVE_DEVICE)
                    UnlockDevice(fdo);
            Return ntStatus;
    }
```