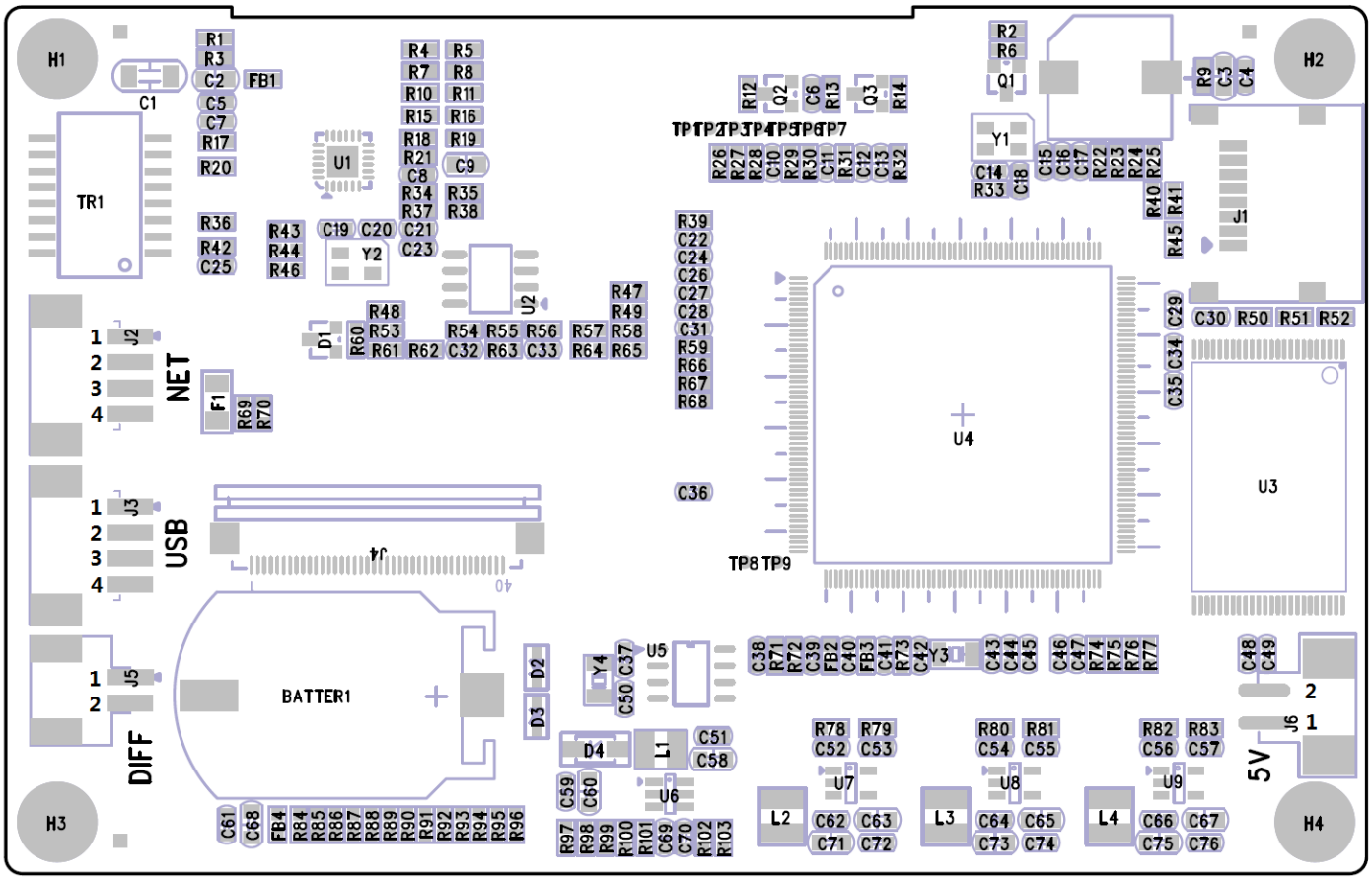
安之谋 HMI972 人机界面 CE 开发手册

版 权 声 明

北京安之谋科技有限公司，多年来一直致力于高质量嵌入式软硬件的开发。由安之谋科技提供 HMI972 人机界面平台可运行独家提供的 CE6，除了具有常见的功能之外，还提供了各种方便客户二次开发和生产的功能。

1. 硬件位号图（4.3 寸）



2. 安装 SDK

安装 SDK 要求本机已经安装了 Visual Studio 2005 SP1。该 SDK 支持虚拟机调试和真机调试。
安装完 SDK 之后，就可以打开 HMITest 工程进行编译和调试了。HMITest 工程源代码请自行下载。

3. 访问 GPIO

设备名 PIO1
控制代码：

```
// IOCTL Code For GPIO
#define IOCTL_SET_DIR                   0x01
#define IOCTL_READ_DATA                 0x02
```

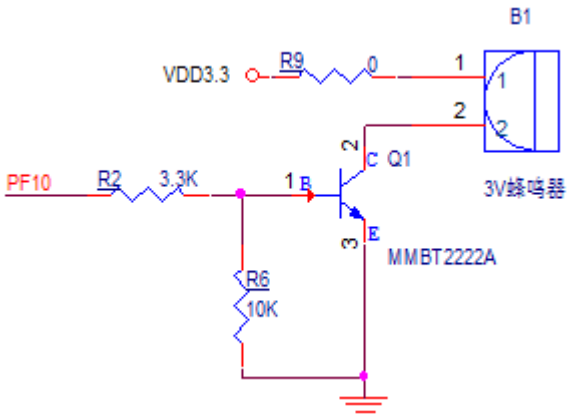
```
#define IOCTL_WRITE_DATA                0x03
#define IOCTL_SET_INTERRUPT             0x04
```

IOCTL_SET_DIR：设置 GPIO 的输入输出方向
IOCTL_READ_DATA：读取 GPIO 的输入值
IOCTL_WRITE_DATA：设置 GPIO 的输出值
IOCTL_SET_INTERRUPT：设置 GPIO 中断的类型
示例代码:

```
DWORD dwIOParam[3];
HANDLE hPioHandler;
hPioHandler = CreateFile(TEXT("PIO1:"), GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);
if(hPioHandler != INVALID_HANDLE_VALUE)
{
    dwIOParam[0]= GPIO_B;        //GPIO 为 PB.6
    dwIOParam[1]= 6;             //GPIO 为 PB.6
    dwIOParam[2]= GPIO_OUTPUT;   //设置为 GPIO_OUTPUT，如果是读取就设置为 GPIO_INPUT
    DeviceIoControl(hPioHandler, IOCTL_SET_DIR, dwIOParam, 3*sizeof(DWORD), NULL, 0, NULL, NULL);


    dwIOParam[0]= GPIO_B;    // GPIO 为 PB.6
    dwIOParam[1]= 6;         // GPIO 为 PB.6
    dwIOParam[2]= 0;         //这里 0 是输出低电平点亮，1 是输出高电平关掉。
    DeviceIoControl(hPioHandler, IOCTL_WRITE_DATA, dwIOParam, 3*sizeof(DWORD), NULL, 0, NULL, NULL);

     CloseHandle(hPioHandler);
}
```

4. 控制蜂鸣器

蜂鸣器控制电路：



示例代码，切换蜂鸣器的状态

```
DWORD dwIOParam[3];
BYTE bIOVal=0;
HANDLE hPioHandler;
hPioHandler = CreateFile(TEXT("PIO1:"), GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);
if(hPioHandler != INVALID_HANDLE_VALUE)
{
    dwIOParam[0]= GPIO_F;          //蜂鸣器对应的 GPIO 为 PF.10
    dwIOParam[1]= 10;               //蜂鸣器对应的 GPIO 为 PF.10
```

```
    dwIOParam[2]= GPIO_OUTPUT; //设置为 GPIO_OUTPUT
    DeviceIoControl(hPioHandler, IOCTL_SET_DIR, dwIOParam, 3*sizeof(DWORD), NULL, 0, NULL, NULL);

    dwIOParam[0]= GPIO_F;    //蜂鸣器对应的 GPIO 为 PF.10
    dwIOParam[1]= 10;                //蜂鸣器对应的 GPIO 为 PF.10
    DeviceIoControl(hPioHandler, IOCTL_READ_DATA, dwIOParam, 3*sizeof(DWORD), &bIOVal, sizeof(BYTE), NULL, NULL);

    dwIOParam[0]= GPIO_F;    //蜂鸣器对应的 GPIO 为 PF.10
    dwIOParam[1]= 10;                //蜂鸣器对应的 GPIO 为 PF.10
    dwIOParam[2]= bIOVal?0:1;    //蜂鸣器状态切换
    DeviceIoControl(hPioHandler, IOCTL_WRITE_DATA, dwIOParam, 3*sizeof(DWORD), NULL, 0, NULL, NULL);

    CloseHandle(hPioHandler);
}
```

5. 控制 LCD 背光

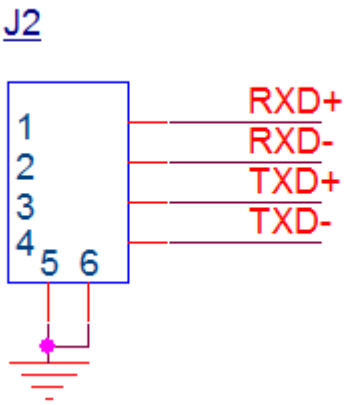
控制背光用到两个 IO。PG.10 用于控制背光的开和关。PB.2 输出的是 PWM 信号，用于控制背光的亮度。
PG.10 的控制代码，参考普通的 IO 控制代码。
背光亮度控制代码如下：

```
HKEY hReg;
HANDLE hEventNotify;
hEventNotify = CreateEvent(NULL, FALSE, FALSE, TEXT("BackLightChangeEvent"));
if(RegOpenKeyEx(HKEY_CURRENT_USER, TEXT("ControlPanel\\BackLight"), 0, 0, &hReg)==ERROR_SUCCESS)
{
    DWORD dwType, dwSize, dwLight;
    dwType = REG_DWORD;
    dwSize = sizeof(DWORD);
    dwLight = 128; //这里设置的是背光的亮度，范围是 0~256，0 最暗，256 最亮。
    if(RegSetValueEx(hReg, TEXT("BackLightValue"), NULL, dwType, (LPBYTE) &dwLight, dwSize)==ERROR_SUCCESS)
    {
        SetEvent(hEventNotify);
    }
    RegCloseKey(hReg);
}
以上方法控制背光亮度，在系统重启之后仍然有效。
```

背光亮度的控制，也可以通过 CE 桌面，进入控制面板>显示>背景光>高级，进行设置。

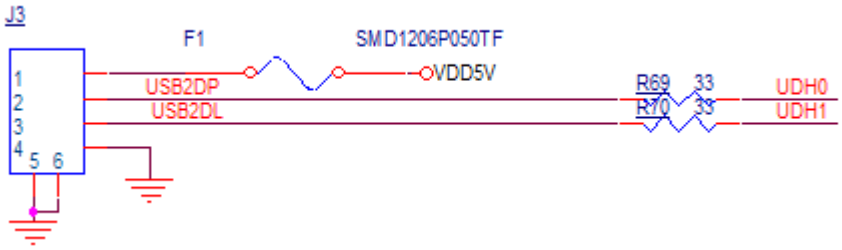
6. 网口

网口默认设置为 DHCP。可使用同一网段的 PC 机和板子连接，进行远程调试和开发。具体步骤参看《Windows CE 6.0 基于以太网的远程调试》。
网口采用 PH2.0-4P 接口，接口位号为 J2。需要按照如下线序接 RJ45 头：



7. USB 主口

USB 主口采用 PH2.0-4P 接口，接口位号为 J3。线序如下：



请按照正确线序制作转接线。

8. 软件复位

软件复位，可使用以下代码：

```
#define IOCTL_KLIB_USER             256
#define IOCTL_USER_REBOOT   CTL_CODE(FILE_DEVICE_HAL, IOCTL_KLIB_USER + 301, METHOD_BUFFERED, FILE_ANY_ACCESS)
HANDLE  hFile;
DWORD nBootType=1;
hFile = CreateFile(TEXT("KIP1:"), GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);
if(hFile != INVALID_HANDLE_VALUE)
{
    DeviceIoControl(hFile, IOCTL_USER_REBOOT, &nBootType, sizeof(DWORD), NULL, 0, NULL, NULL);
    CloseHandle(hFile);
}
```

9. 看门狗

看门狗驱动设备节点为 WTD1:，该驱动提供了以下控制接口。

打开看门狗

接口代码： WATCHDOG_ENABLE

说明：打开看门狗之后，看门狗开始以 2.5 秒左右的最大喂狗间隔开始运行，如果 2.5 秒内没有喂狗动作，系统会复位重启。

关闭看门狗

接口代码： WATCHDOG_DISABLE

说明：关掉看门狗

开始内核自动喂狗

接口代码： WATCHDOG_START_AUTO_FEED

说明：如果使用内核自动喂狗，内核会每隔 1 秒喂一次狗，这一功能用于监控内核是否死机。

手动单次喂狗

接口代码： WATCHDOG_FEED

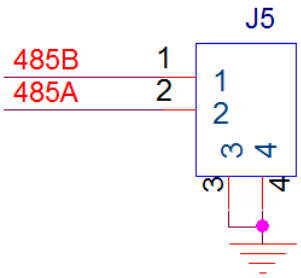
说明：应用程序手动喂狗，可用于监视应用程序是否死掉。如果使用这种喂狗方式，就一定不要再打开自动喂狗模式。

参考代码如下所示:

```
HANDLE hWDTHandler = INVALID_HANDLE_VALUE;

void init_watchdog()
{
        if(hWDTHandler==INVALID_HANDLE_VALUE)
      {
            hWDTHandler = CreateFile(TEXT("WTD1:"), GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);
              RETAILMSG(1, (TEXT("WTD = %x\r\n"), (UINT)hWDTHandler));
      }
}


void enable_watchdog(int mode)
{
      if(hWDTHandler != INVALID_HANDLE_VALUE)
      {
            DeviceIoControl(hWDTHandler, WATCHDOG_ENABLE, NULL, 0, NULL, 0, NULL, NULL);
            if(mode ==0) //manually feed
            {
            }
            if(mode ==1) //auto feed in kernel
            {
                  DeviceIoControl(hWDTHandler, WATCHDOG_START_AUTO_FEED, NULL, 0, NULL, 0, NULL, NULL);
            }
       }
}

void disable_watchdog()
{
      if(hWDTHandler != INVALID_HANDLE_VALUE)
      {
            DeviceIoControl(hWDTHandler, WATCHDOG_DISABLE, NULL, 0, NULL, 0, NULL, NULL);
       }
}

void feed_watchdog()
{
      if(hWDTHandler != INVALID_HANDLE_VALUE)
      {
            DeviceIoControl(hWDTHandler, WATCHDOG_FEED, NULL, 0, NULL, 0, NULL, NULL);
       }
}
```

10. RS485 串口 COM3

COM3 串口为 RS485 电平的半双工差分接口，接口位号为 J5。

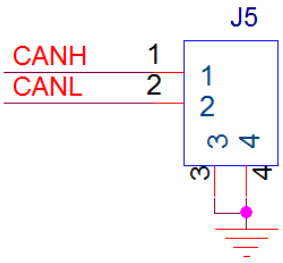


该串口为通用串口，编程接口参看 Windows 串口编程的相关文档即可。

开发板采用了 CAN 接口和 RS485 接口的二选一设计，CAN 和 RS485 同是使用 J5 作为外部接口。通过焊接不同的芯片来实现这一功能。使用 RS485 功能时，请确认硬件是否匹配。

11. CAN 接口

CAN 接口位号为 J5。



开发板采用了 CAN 接口和 RS485 接口的二选一设计，CAN 和 RS485 同是使用 J5 作为外部接口。通过焊接不同的芯片来实现这一功能。使用 CAN 功能时，请确认硬件是否匹配。

设备名 CBS1
控制代码：

```
#define IOCTL_CAN_TX                (1)    //发送一帧数据
#define IOCTL_CAN_RX                (2)    //接收一帧数据
#define IOCTL_CAN_SETUP_RX          (3)    //设置接收的参数
#define IOCTL_CAN_SET_BAUD          (4)    //设置波特率
```

打开设备

```
hCanHandler = CreateFile(TEXT("CBS1:"), GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);
```

设置波特率为 500K，并获得实际波特率。

```
DWORD dwRealBaud, dwBaud = 500000;
DeviceIoControl(hCanHandler, IOCTL_CAN_SET_BAUD, &dwBaud, sizeof(DWORD), &dwRealBaud, sizeof(DWORD), NULL, NULL);
```

设置接收参数，并开始接收循环数据。

```
CAN_IOCTL_MSG tMsg;
tMsg.nMsgObj = RX_MSG_INDEX(0);         //使用接收窗口 0
tMsg.msg.IdType      =    CAN_STD_ID;       //接收标准帧
tMsg.msg.Id             = 0X7ff;                  //接收 ID
DeviceIoControl(hCanHandler, IOCTL_CAN_SETUP_RX, &tMsg, sizeof(CAN_IOCTL_MSG), NULL, 0, NULL, NULL);
BYTE* pRecvData = (BYTE*)malloc(65536);;
```

```
while(TRUE)
{
	tMsg.nMsgObj = RX_MSG_INDEX(0);		//使用接收窗口 0
	tMsg.nWaitMs = 1000;				//超时设置为 1 秒, 1 秒后没有接收到数据，DeviceIoControl 函数返回 FALSE。
	DWORD dwRecvCount=0;
	bRet=DeviceIoControl(hCanHandler, IOCTL_CAN_RX, &tMsg, sizeof(CAN_IOCTL_MSG), pRecvData, 65536, &dwRecvCount, NULL);
	if(bRet)
	{
		...
	}
}
free((void*)pRecvData);
```

发送一帧数据

```
CAN_IOCTL_MSG tMsg;
tMsg.nMsgObj = TX_MSG_INDEX(0);
tMsg.nWaitMs = 1000;                        //发送超时设置为 1 秒，1 秒后没有发送成功，DeviceIoControl 函数返回 FALSE。
tMsg.msg.FrameType= DATA_FRAME;             //发送数据帧
tMsg.msg.IdType      = CAN_STD_ID;          //发送标准帧
tMsg.msg.Id          = 0X7FF;               //ID
tMsg.msg.DLC         = 8;                   //数据长度
tMsg.msg.Data[0]    = 0x5A;
tMsg.msg.Data[1]    = 0x5A;
tMsg.msg.Data[2]    = 0x5A;
tMsg.msg.Data[3]    = 0x5A;
tMsg.msg.Data[4]    = 0x5A;
tMsg.msg.Data[5]    = 0x5A;
tMsg.msg.Data[6]    = 0x5A;
tMsg.msg.Data[7]    = 0x5A;
bRet=DeviceIoControl(hCanHandler, IOCTL_CAN_TX, &tMsg, sizeof(CAN_IOCTL_MSG), NULL, 0, NULL, NULL);

if(bRet)
{
	TRACE(TEXT("CAN 发送成功\r\n"));
}
else
{
	TRACE(TEXT("CAN 发送失败\r\n"));
}
```

更加详细的 CAN 接口使用方法，请参考我司提供的 CANTest 程序源代码。
HMI977 的 CAN 接口和 RS485 采用的是二选一设计，具体参看文档《NUC97X 的串口 CAN 兼容设计》

12. 用 U 盘升级固件,开机画面和 App

固件升级：

把 ENKUP.bin, EBOOT.bin，Logo.bmp 等文件拷贝到 U 盘根目录，然后把 U 盘插到 USB 主口上。系统会自动弹出升级界面，点升级按钮，升级完成后重新启动板子即可。

开机画面格式：

开机画面 Logo.bmp 文件为 32 位 RGB，RGBX 格式，行序倒置。画面大小和 LCD 大小一致(480x272)。用 PS 等软件默认保存的 32 为 BMP 文件为 XRGB 格式，不能直接使用。建议用 PS 等软件打开我司提供的 Logo.bmp 文件，进行编辑后再保存。

App 升级：

在电脑自行制作应用程序升级包 UpdateApp.zip (升级包内的所有文件会由升级程序自动拷贝到\NandFlash\App 目录下)。在 zip 文件内的根目录中，创建 autorun.ini，文件内容为需要开机自动执行的 EXE，每个 EXE 一行，EXE 路径写以\NandFlash\App 开头的路径。

把 UpdateApp.zip 拷贝到 U 盘根目录。在板子上插入 U 盘，然后按照屏幕提示升级即可。(升级时无需退出原来的程序) 。升级完成拔掉 U 盘断电重启。系统启动后自动运行 autorun.ini 指定的程序，不再进入桌面。

如对制作升级包有疑问，可参考我司提供的示例 UpdateApp.zip

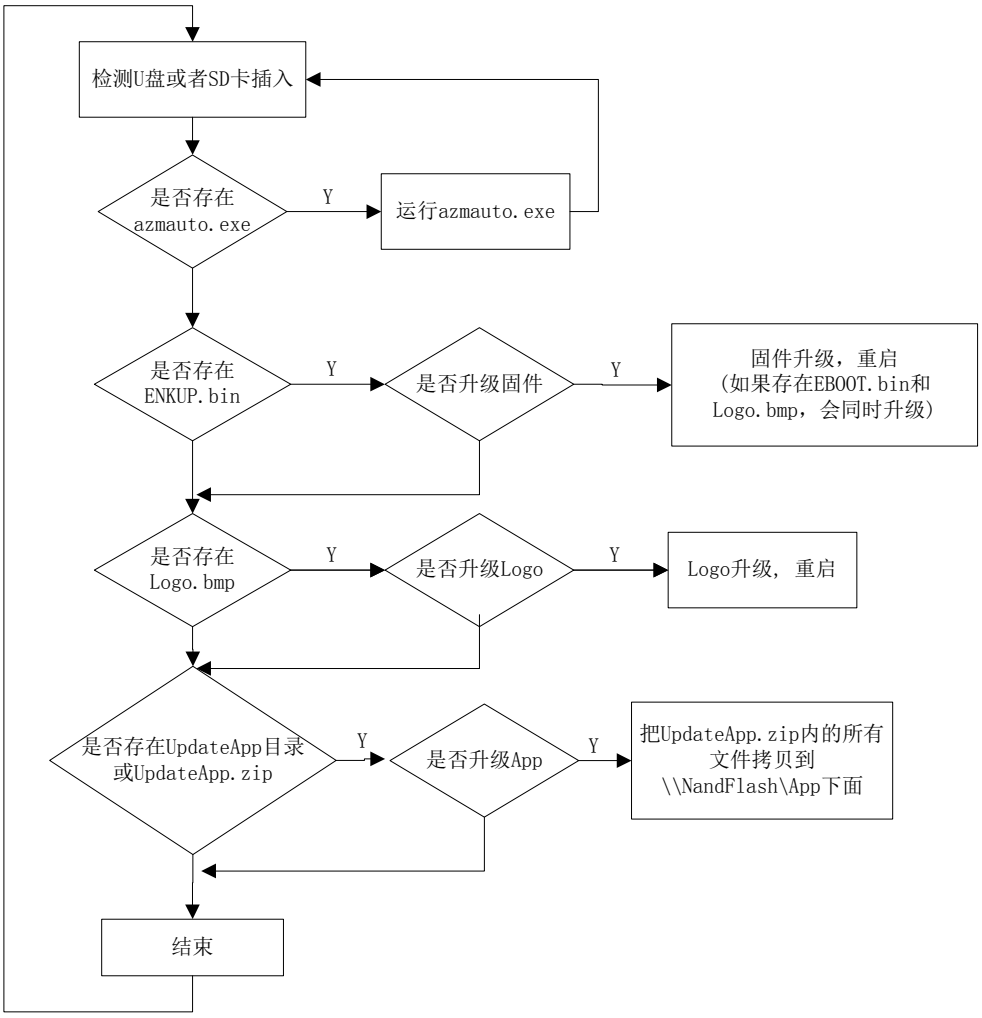
以上升级方式同样适用于 SD 卡。可以关机状态先插入有升级文件的 SD 卡，然后再开机。

默认固件带有此项功能，如果实际产品不需要这项功能，使用如下代码屏蔽 Launch130 即可。

```
HKEY hReg;
if(RegOpenKeyEx(HKEY_LOCAL_MACHINE, TEXT("init"), 0, 0, &hReg)==ERROR_SUCCESS)
{
    RegDeleteValue(hReg, TEXT("Launch130"));
    RegCloseKey(hReg);
}
```

13. U 盘自动升级和运行程序的逻辑

系统对 U 盘插入后处理逻辑如下：



14. 唯一序列号

HMI977 提供了一个唯一序列号供应用程序使用。该序列号可以通过获取网卡 MAC 地址获得。

```
DWORD dwLen;
  IP_ADAPTER_INFO mInfo={0};
  TCHAR szShowID[256]={0};

  dwLen = sizeof(IP_ADAPTER_INFO);
```

```
GetAdaptersInfo(&mInfo, &dwLen);
swprintf_s(szShowID, sizeof(szShowID)/sizeof(TCHAR),
        TEXT("Mac Address & Board Unique ID : %02X-%02X-%02X-%02X-%02X-%02X"),
        mInfo.Address[0], mInfo.Address[1], mInfo.Address[2],
        mInfo.Address[3], mInfo.Address[4], mInfo.Address[5]);
MessageBox(NULL, szShowID, TEXT("ID"), MB_OK|MB_ICONINFORMATION);
```

如果是不带网卡的定制板，也可以用下面这个接口获取该序列号：

```
#define HAL_GET_HARDWARE_ID                 102
DWORD dwLen;
HANDLE    hFile;
BYTE mMacAddr[8]={0};
TCHAR szShowID[256]={0};

hFile = CreateFile(TEXT("KIP1:"), GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);
if(hFile != INVALID_HANDLE_VALUE)
{
        DeviceIoControl(hFile, HAL_GET_HARDWARE_ID, NULL, 0, mMacAddr, sizeof(mMacAddr), &dwLen, NULL);
        CloseHandle(hFile);
}

swprintf_s(szShowID, sizeof(szShowID)/sizeof(TCHAR),
        TEXT("Mac Address & Board Unique ID : %02X-%02X-%02X-%02X-%02X-%02X"),
        mMacAddr[0], mMacAddr[1], mMacAddr[2],
        mMacAddr[3], mMacAddr[4], mMacAddr[5]);
MessageBox(NULL, szShowID, TEXT("ID"), MB_OK|MB_ICONINFORMATION);
```

15. 开机进度条

开机过程的进度条，默认在启动应用程序的时候就跳到 100%并不再刷新。如果有的应用程序启动比较慢，可以通过注册表，让进度条多走一会儿。

注册表选项如下，默认值为 0。如果需要多走 5 秒，则写入 5 即可。

```
[HKEY_LOCAL_MACHINE\ARMDEVICE]
"DelayOffProgress"=dword:0
```

16. 使用 Visual Studio 2005 开发 C++应用

使用 Visual Studio 2005 开发 WinCE 的 C++程序，需要安装以下内容：

先安装 Visual Studio 2005，选择 C++相关组件

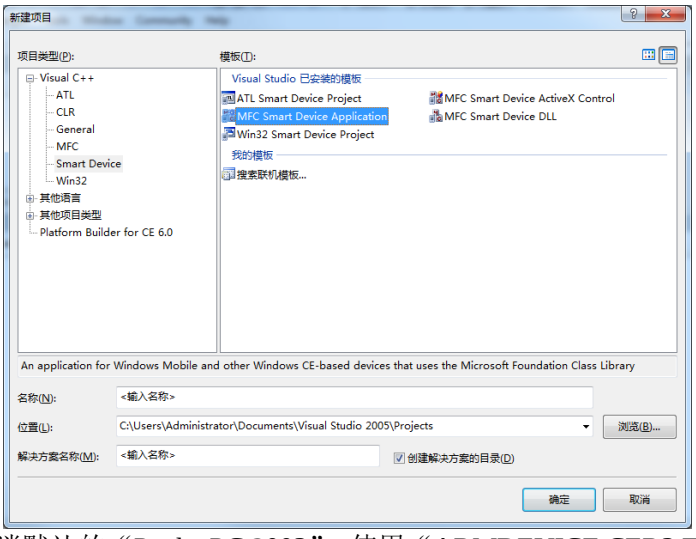
再安装 Visual Studio 2005 Service Pack 1

如果是 Win7 或 Win10 系统，还需要安装 Visual Studio 2005 Service Pack 1 Update for Windows Vista

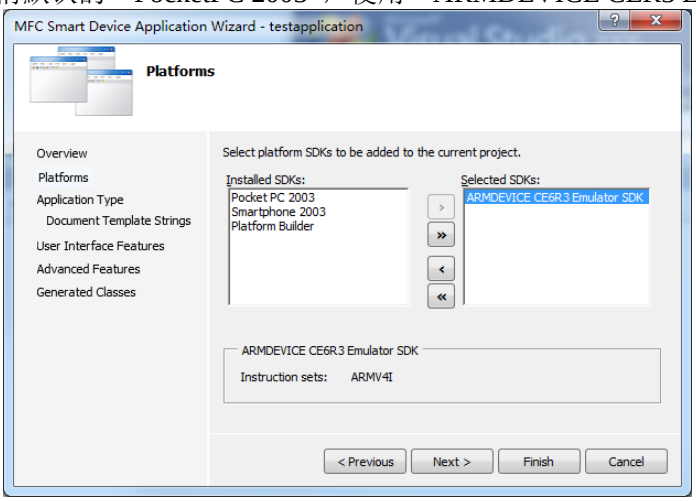
最后安装我司提供的 SDK 包 ARMDEVICE_WCE6R3SDK.msi

按照如下步骤，我们新建一个 C++工程，然后进行联机调试。（以下示例使用英文版 Visual Studio 2005）

首先新建一个“MFC Smart Device Application”类型的工程，



在选择平台的界面中，取消默认的“PocketPC 2003”，使用“ARMDEVICE CER3 Emulator SDK”，如下图：



成功创建工程之后，请参照《安之谋科技 CE6 开发板基于以太网的远程调试》来进行远程调试。

17. 使用 Visual Studio 2005 开发 C#应用

使用 Visual Studio 2005 开发 WinCE 的 C#程序，需要安装以下内容：

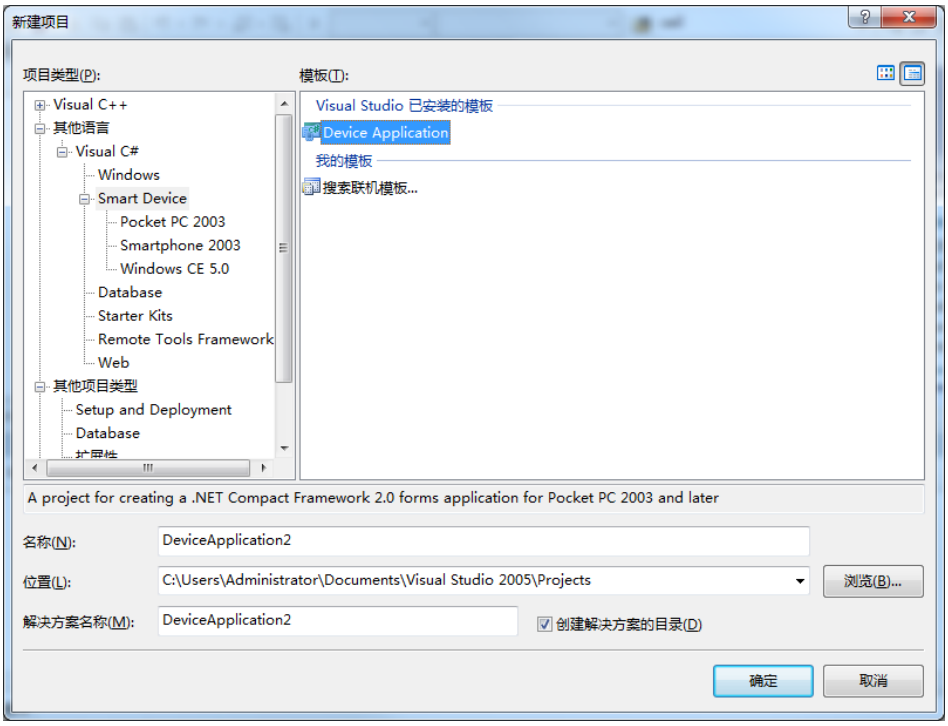
先安装 Visual Studio 2005，选择 C#相关组件

再安装 Visual Studio 2005 Service Pack 1

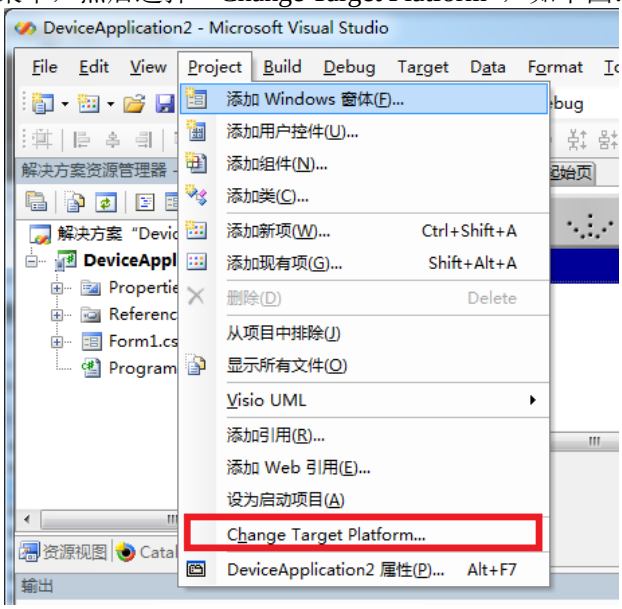
如果是 Win7 或 Win10 系统，还需要安装 Visual Studio 2005 Service Pack 1 Update for Windows Vista

最后安装我司提供的 SDK 包 ARMDEVICE_WCE6R3SDK.msi

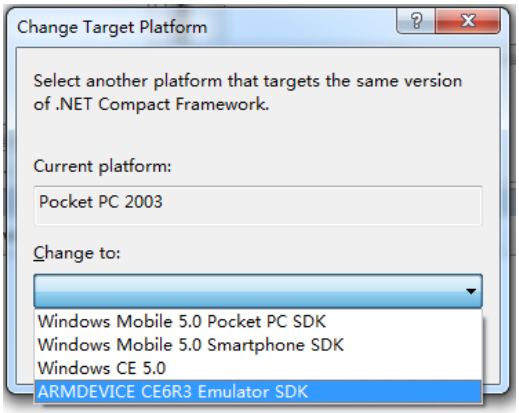
按照如下步骤，我们新建一个 C#工程，然后进行联机调试。（以下示例使用英文版 Visual Studio 2005）
在创建 Smart Device 类型的 C#工程的时候，CE6R3 的并不在设备类型内，没有关系，需要随便选一个就行。



成功创建工程之后，打开 Project 菜单，然后选择“Change Target Platform”，如下图：



在随后的对话框中，将目标平台设置为“ARMDEVICE CE6R3 Emulator SDK”即可。



最后请参照《安之谋科技 CE6 开发板基于以太网的远程调试》来进行远程调试。

18. 使用 Visual Studio 2008 开发 C++应用

使用 Visual Studio 2008 开发 WinCE 的 C++程序，需要安装以下内容：

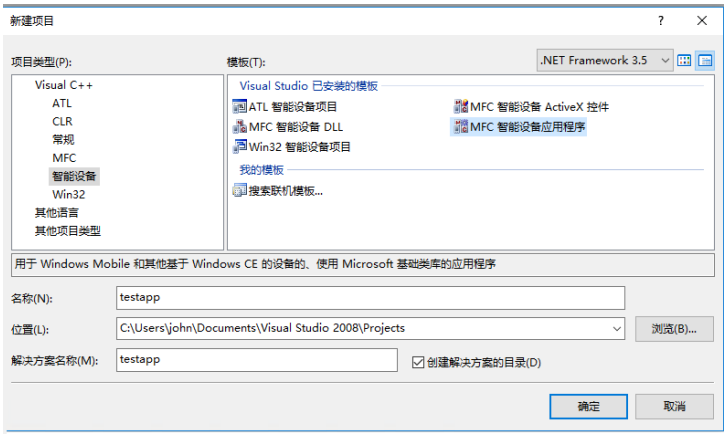
先安装 Visual Studio 2008，选择 C++相关组件

再安装 Visual Studio 2008 Service Pack 1

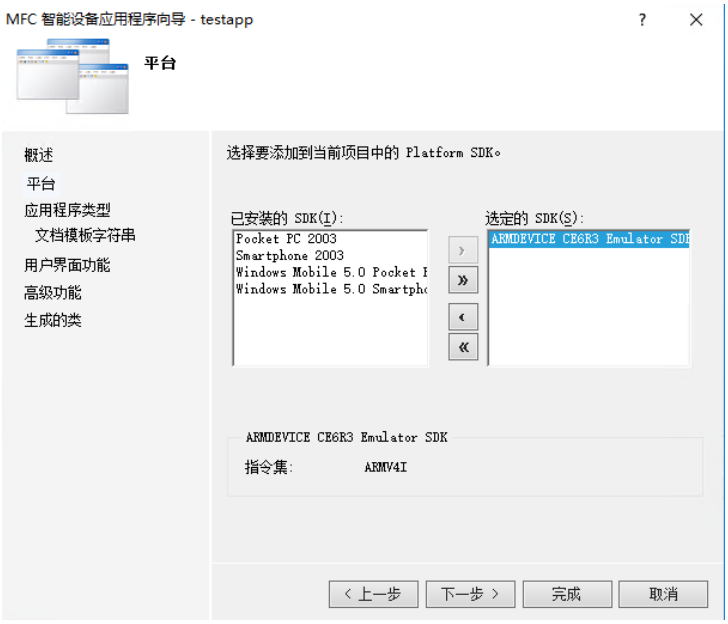
最后安装我司提供的 SDK 包 ARMDEVICE_WCE6R3SDK.msi

按照如下步骤，我们新建一个 C++工程，然后进行联机调试。（以下示例使用中文版 Visual Studio 2008）

首先新建一个“MFC 智能设备应用程序”类型的工程，



在选择平台的界面中，取消默认的“PocketPC 2003”，使用“ARMDEVICE CER3 Emulator SDK”，如下图：



成功创建工程之后，请参照《安之谋科技 CE6 开发板基于以太网的远程调试》来进行远程调试。

19. 使用 Visual Studio 2008 开发 C#应用

使用 Visual Studio 2008 开发 WinCE 的 C#程序，需要安装以下内容：

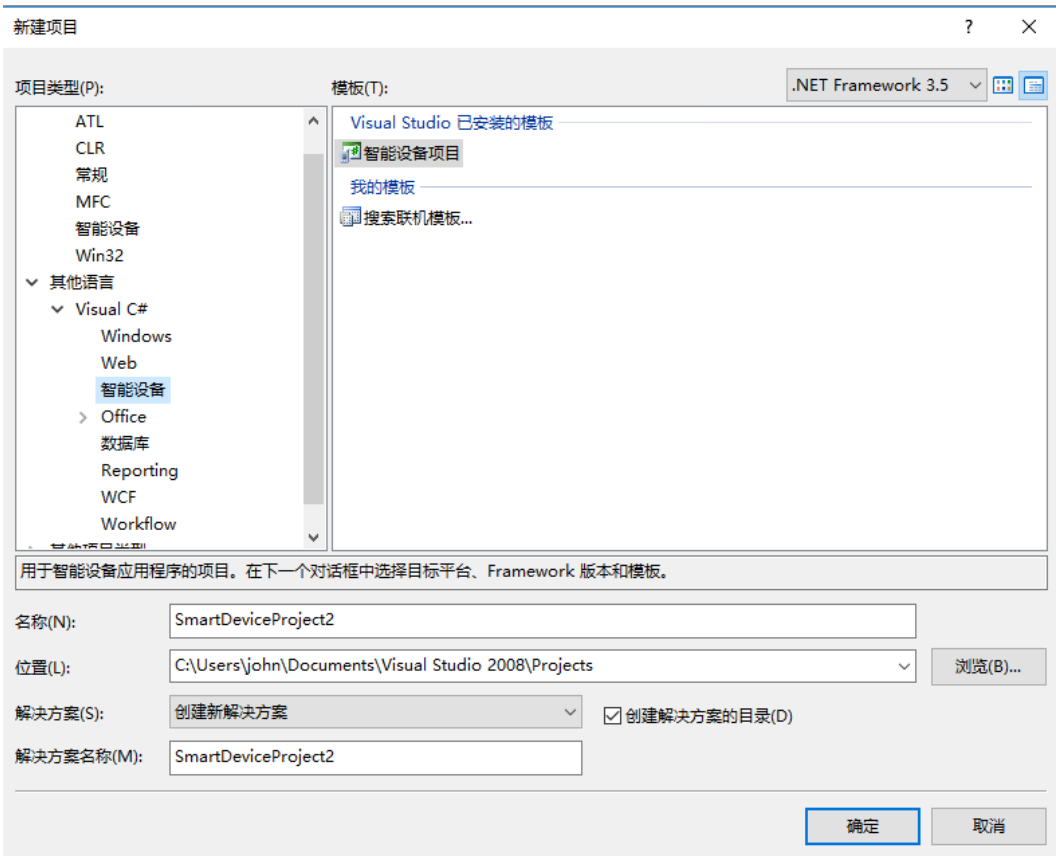
先安装 Visual Studio 2008，选择 C#相关组件

再安装 Visual Studio 2008 Service Pack 1

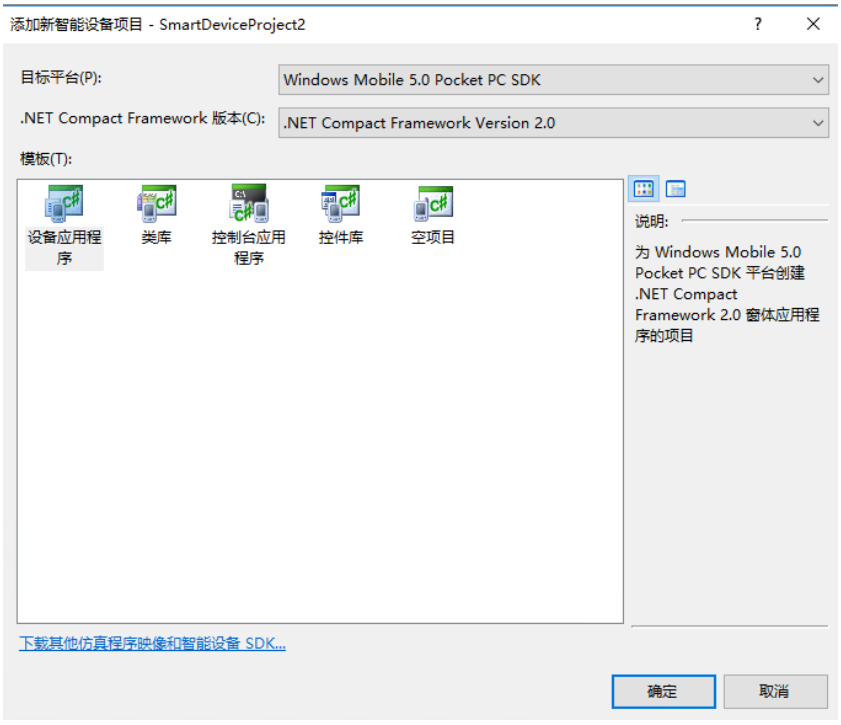
最后安装我司提供的 SDK 包 ARMDEVICE_WCE6R3SDK.msi

按照如下步骤，我们新建一个 C#工程，然后进行联机调试。（以下示例使用中文版 Visual Studio 2008）

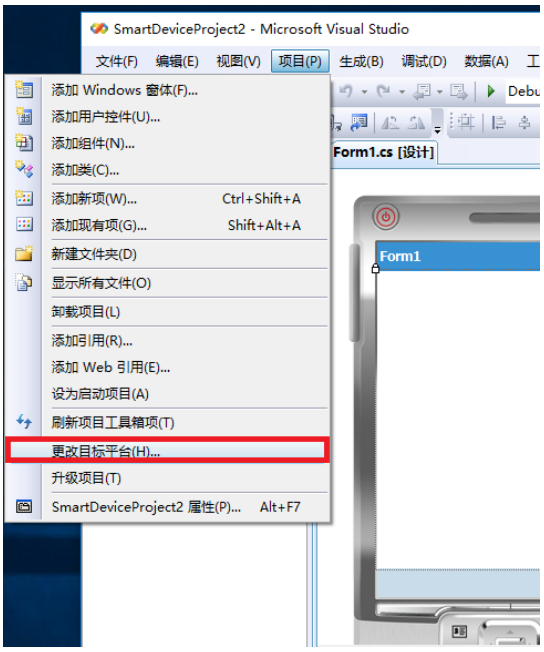
首先创建“智能设备项目”。



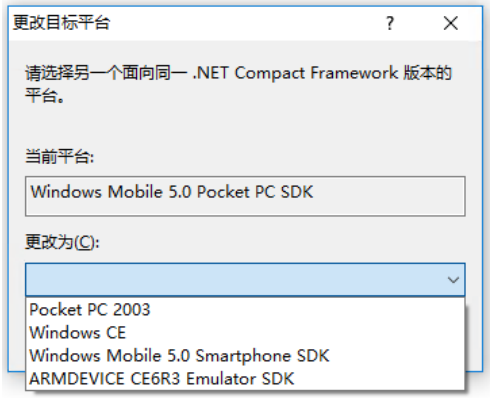
在下一步设置目标平台的向导界面中，CE6R3 的并不在内，没有关系，需要随便选一个就行。.Net 版本建议选 2.0，因为出厂固件默认带 2.0 版本的.Net 库。



成功创建工程之后，打开项目菜单，然后选择“更改目标平台”，如下图：



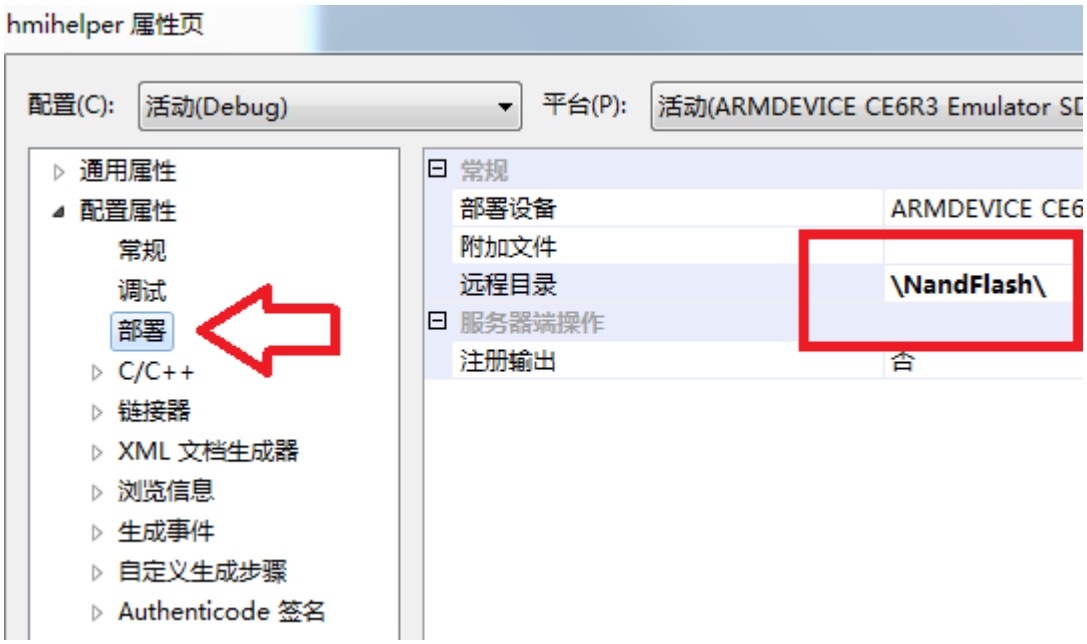
在随后的对话框中，将目标平台设置为“ARMDEVICE CE6R3 Emulator SDK”即可。



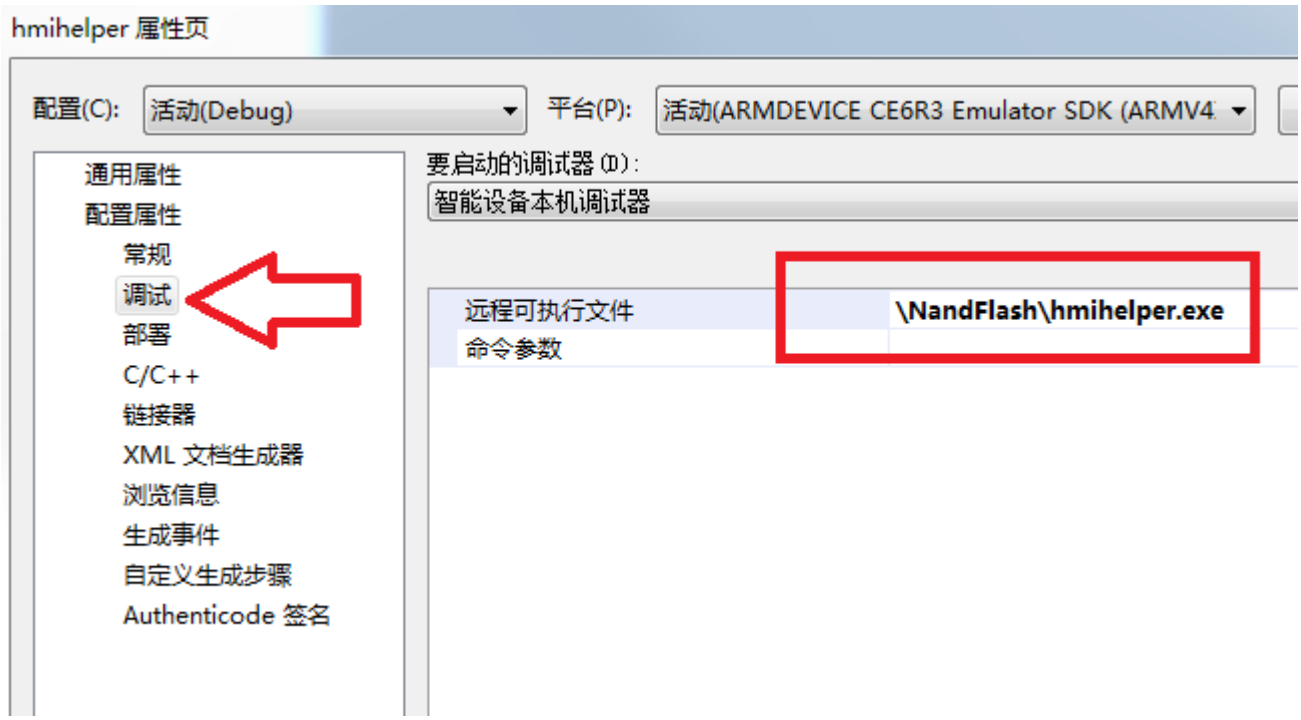
最后请参照《安之谋科技 CE6 开发板基于以太网的远程调试》来进行远程调试。

20. 设置 VisualStudio 工程的部署目录和调试目录

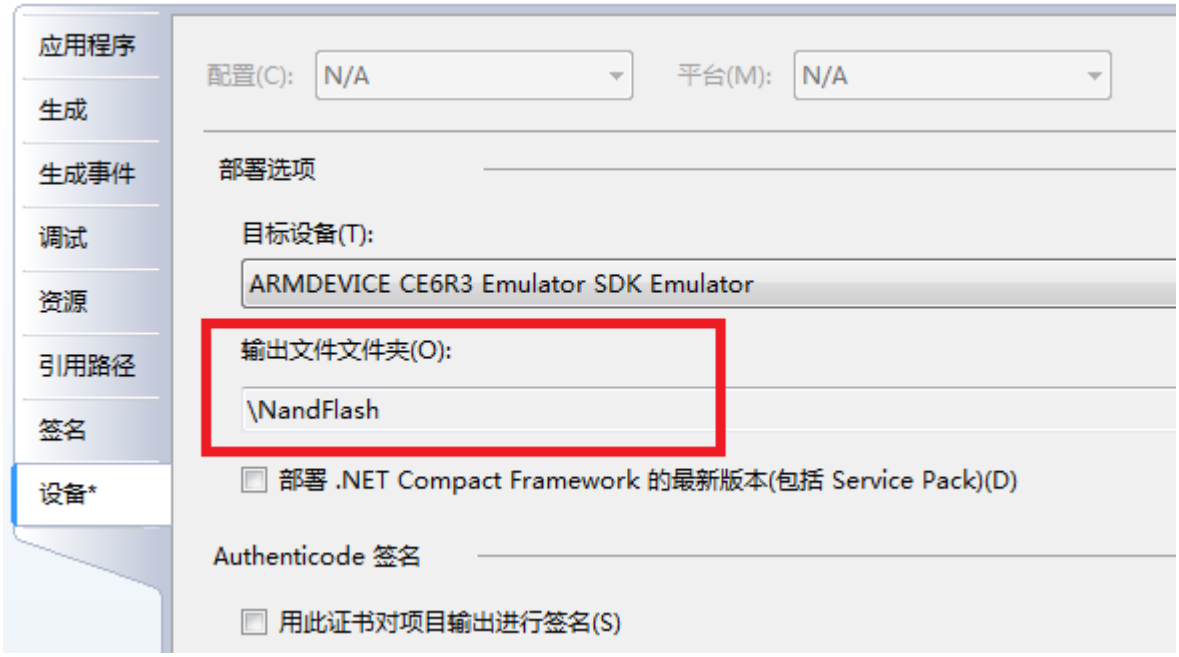
默认固件采用的 ROMRAM 类型文件系统，只有\NandFlash 目录支持存储，因此调试的时候，需要把工程的部署目录和调试目录都修改成\NandFlash 目录下。



VS2005/VS2008 C++ 修改部署目录



VS2005/VS2008 C++ 修改调试目录



VS2005/VS2008 C# 修改输出目录

C#调试的时候，不要选择“部署.NET Compact Framwork 的最新版本”，而应该直接在板子上下载对应.NET 版本的固件。

21. C#库和参考例程

我司提供的一个含源码的配套库供 C#程序调用，可以实现 GPIO 的读写，板子的复位等操作。
详情参看文档《HMI97X 的 DotNet 开发例程和配套库》和相关代码。

22. 培训和技术支持

HMI972 人机界面的 Windows CE 6.0 BSP，目前市面上只有安之谋科技提供。同时安之谋科技也提供对该 BSP 的技术支持和培训服务。

如有需要可联系我们。

Email:contacts@armdevice.com