## 版权声明

本手册版权归属北京安之谋科技有限责任公司(以下简称"安之谋科技")所有, 并保留一切权力。非经安之谋科技同意(书面形式),任何单位及个人不得擅自 摘录本手册部分或全部,违者我们将追究其法律责任。 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; //GPI0 为 PB.6 dwIOParam[1]= 6; //GPI0 为 PB.6 dwIOParam[2]= GPI0\_OUTPUT; //设置为 GPI0\_OUTPUT, 如果是读取就设置为 GPI0\_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; // GPI0 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:

DWORD dwIOParam[3]; BYTE bIOVal=0; HANDLE hPioHandler;

{



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; //峰鸣器对应的 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。PG10 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。 PG10 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); } UL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; DeviceloControl(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.ldType = CAN\_STD\_ID; //接收标准帧 tMsg.msg.ld = 0X7ff; //接收 ID DeviceloControl(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,DeviceloControl函数返回FALSE。
    DWORD dwRecvCount=0;
    bRet=DeviceloControl(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,DeviceloControl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.ld
                 = 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。

# 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);



# 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,

	新建项目	tage (		
	项目类型( <u>P</u> ):	模板(I):		
	<ul> <li>→ Visual C++</li> <li>ATL</li> <li>CLR</li> <li>General</li> <li>MFC</li> <li>Smart Device</li> <li>Win32</li> <li>→ 具地语言</li> <li>○ 具地语言</li> <li>□ 具地语是型</li> <li>▶ Platform Builder for CE 6.0</li> </ul>	Visual Studio 已完装的模板 副 ATL Smart Device Project 器 MFC Smart Device ActiveX Cor 器 MFC Smart Device OLL 器 MFC Smart Device OLL 影的模板 词 撞索联环模板	ntrol	
	An application for Windows Mobile	and other Windows CE-based devices that uses the Microsoft Foundation Class	ss Library	
	名称(N): 《输入名称》			
	位置(L): C:\Users\Admini	strator\Documents\Visual Studio 2005\Projects 🗸	浏览(B)	
	解决方案名称( <u>M</u> ): <輸入名称>	☑ 创建解决方案的目录(D)		
		确定	取消	
在选择平台的界面中, 取	当默认的"Pocke	tPC 2003". 使用"ARMDEVICE	ECER3 Emulator SDK",如下图	•
	MFC Smart Device Application	Wizard - testapplication		•
	Platform	15		
	Overview	Select platform SDKs to be added to the current project.		
	Platforms Application Type Document Template Strings User Interface Features Advanced Features Generated Classes	Installed SDKs: Pocket PC 2003 Smarthone 2003 Platform Builder ( ( ( ( ( ( ( ( ( ( ( ( (	Jator SDK	
		ARMDEVICE CE6R3 Emulator SDK Instruction sets: ARMV4I		
		< Previous Next > Finish	Cancel	

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。

新建项目	n - n - 2 - 34 -	and the second second	- 8 -		? ×
项目类型(P):	模	板①:			
● Visual C++ ● 其他语言 ● Visual C# ● Visual C# ● Smart D ● Smart D ● Smart D ● Smart D ● Smart D ● Smart D ● Starter H ● Remote ● Starter H ● Starter	s evice et PC 2003 tows CE 5.0 e Gits Tools Framework	Visual Studio 已安装的模板 <mark>Povice Application</mark> 我的模板 現象联机模板			
A project for creat 名称(N):	ing a .NET Compact Fra	mework 2.0 forms application	for Pocket PC 2003 and late	er	
位置(L):	C:\Users\Administrator	r\Documents\Visual Studio 200	5\Projects	<b>-</b>	浏览( <u>B</u> )
解决方案名称(M):	DeviceApplication2		☑ 创建解决方案的目录(D)		
				确定	取消

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

🥠 Dev	viceAp	plication	12 - N	Aicro	soft Vis	ual Studio				
<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	Pro	ect	<u>B</u> uild	<u>D</u> ebug	Ta <u>r</u> get	D <u>a</u> ta	Format	<u>T</u> o
: 🛅 -	-	💕 🔛	1	添加	🛛 Windo	ws 窗体(F	)		bug	J
	후 네	리   1		添加	0用户控(	4 <u>(U</u> )			美	: 음:
解决方	案资源	管理器 -	9	添加	0组件( <u>N</u>	)			己始了	页
	🍙 🍺		23	添加	噗( <u>C</u> )					
灵 🖗	<b>決方案</b>	Devic	-	添加	ௌ <u></u> ज्ज(₩	)	Ctrl+	Shift+A	•	
ė 🗗	🛚 Devi	iceAppl	:::	添加	1现有项(	<u>G</u> )	Shif	t+Alt+A		
	• 🚾 P	ropertie	$\times$	删降	余( <u>D</u> )			Delete		
	- 🔟 K	eterenc		从现	而日中排	余(J)				
l I	- 🔮 P	rogram	a	显	示所有文化	4( <u>O</u> )				
				<u>V</u> is	io UML				•	
				添加	o引用( <u>R</u> )					
				添加	🛛 Web 🗄	引用(E)				
				设为	り启动项	∃( <u>A</u> )				
<				C <u>h</u> a	ange Ta	rget Platfo	orm			
) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1	飘图	🕑 Catal	c	De	viceApp	ication2 🛙	重性( <u>P</u> )	Alt+F7		
2891日			-							

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

Change Target Platform	8 ×	
Select another platform that targets the s of .NET Compact Framework.	ame version	
Current platform:		
Pocket PC 2003		
Change to:		
Windows Mobile 5.0 Posket PC SDK		
Windows Mobile 5.0 Pocket PC SDK Windows Mobile 5.0 Smartphone SDK		
Windows CE 5.0		
ARMDEVICE CE6R3 Emulator SDK		

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,

新建项目					?	х
项目类型(P):		模板(T):		.NET Framework 3.5	~ ::	1
Visual C++ ATL CLR 常规 MFC 智能设备 Win32 其他语言 其他项目类型		Visual Studie 已受要的模板 ■ ATL 智能没错页目 ▲ MFC 智能没错 DLL ② Win32 智能没错项目 我的模板 ③ 搜索联机模板	調整MFC 智能设备 調整MFC 智能设备	骨 ActiveX 控件 输应用程序		
用于 Windows Mo	obile 和其他基于 Windo	ows CE 的设备的、使用 Microsoft 基	國美產的应用程序			
名称(N):	testapp					
位置(L):	C:\Users\john\Docu	iments\Visual Studio 2008\Projects		~	浏览(B)	
解决方案名称(M):	testapp		☑ 创建解决方案的目录(D	)		
				确定	取消	

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:

MIC 目記区用近内住分向寺 - 16	stapp i A
平台	
<ul> <li>概述</li> <li>平台</li> <li>应用程序类型</li> <li>文档模板字符串</li> <li>用户界面功能</li> <li>高级功能</li> <li>生成的类</li> </ul>	法择要添加到当前项目中的 Platform SDK。 已安装的 SDK(I): Pocket FC 2003 Smartphone 2003 Windows Mobile 5.0 Pocket I Windows Mobile 5.0 Smartpho
	ARMOEVICE CE6R3 Emulator SDK 指令集: ARMV4I 〈 上一步 下一步 〉 完成 取消

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"。

新建项目			? ×
项目类型(P):		模板(T): .NET Framework 3.5	× III 📰
ATL CLR 常规 MFC 智能设备 Win32 Visual C# Window Web 智能设备 > Office 数据库 Reportin WCF Workflo	s Ig W	Visual Studio 已安装的模板 ☑ 智能设备项目 我的模板 ☑ 搜索联机模板	
用于智能设备应用程	序的项目。在下一个对	话框中选择目标平台、Framework 版本和模板。	
名称(N):	SmartDeviceProject	2	
位置(L):	C:\Users\john\Docu	ments\Visual Studio 2008\Projects v	浏览(B)
解决方案(S):	创建新解决方案	~ ☑ 创建解决方案的目录(D)	
解决方案名称(M):	SmartDeviceProject	2	
		确定	取消

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。

添加新智能设备项目 - SmartDevicePro	ject2		?	$\times$
目标平台(P):	Windows Mobile 5.0 Pocket PC SDK			~
.NET Compact Framework 版本(C):	.NET Compact Framework Version 2.0			~
模板(T):				
○ 公式 1000 日日 2000 日 2000 日日 2000 日 2000 日日 2000 日	用 控件库 空项目	证::::::::::::::::::::::::::::::::::::	Nobile 5. K 平台创 t 0 窗体应	0建用程
		确完	取消	
	L	H U/ALL	-9015	

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:

		Const Davies Project? Microsoft Viewal Studio
	0-	
		添加Windows 窗体(F)
	±	
	94 1	raxJL8出中(IN) Form1.cs [设计]
		7850日来でのAD C+-I、SLift A
		活動和加速(G)。 Shift+Alt+A
		Active in Acti
	œr	
		添加 Web 引用(F)
		设为启动项目(A)
	44	刷新项目工具箱项(T)
		更改目标平台(H)
		升级项目(T)
	e	SmartDeviceProject2
在随后的对话框中 <b>,</b> 将目标平台设置	为	"ARMDEVICE CE6R3 Emulator SDK"即
		更改目标平台 ? X
		请选择另一个面向同一 .NET Compact Framework 版本的平台。 当前平台:
		Windows Mobile 5.0 Pocket PC SDK
		国政力(C)・
		SCR/3(2).
		Pocket PC 2003 Windows CE Windows Mobile 5.0 Smartphone SDK

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

1	
应用程序	
生成	
生成事件	部署选项
调试	目标设备(T):
资源	ARMDEVICE CE6R3 Emulator SDK Emulator
引用路径	输出文件文件夹(O):
签名	\NandFlash
·设备*	□ 部署 .NET Compact Framework 的最新版本(包括 Service Pack)(D)
	Authenticode 签名
	用此证书对项目输出进行签名(S)
	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。

#### 培训和技术支持 22.

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