Emtronix

EM9280 工控主板数据手册

感谢您购买英创信息技术有限公司的产品：**EM9280 工控主板。**

EM9280 是一款面向工业自动化领域的高性价比嵌入式主板，以 FreeScale 的 iMX283 为其硬件核心，EM9280 通过预装完整的操作系统及接口驱动，为用户构造了可直接使用的通用嵌入式核心平台。目前 EM9280 可选择预装 Windows CE6.0（R3）或 Linux-2.6.35 两种平台，用户应用程序开发方面，对 CE 平台可直接使用 Microsoft 提供的著名软件开发工具 Visual Studio 2005 进行应用开发；对 Linux 平台可采用英创公司提供的 Eclipse 集成开发环境（Windows 版本），其编译生成的程序可直接运行与 EM9280。英创公司针对 EM9280 提供了完整的接口低层驱动以及丰富的应用程序范例，用户可在此基础上方便、快速地开发出各种工控产品。

EM9280 主要特点：

- **支持多达 8 路标准串口：**EM9280 是英创公司第一款带有 8 路标准异步串口（UART）的工控主板产品，8 串口配置可满足目前工控产品绝大部分应用需求，从而加快客户整机产品的开发速度，同时降低其成本。

- **强大的硬件配置：**EM9280 的 CPU 为 ARM926EJ-S，其运行主频高达 454MHz。配以 128MB 的系统内存，使 EM9280 能满足各种大型应用程序的运行需求。在显示方面，EM9280 即可支持彩色 LCD 显示，也可支持单色 LCD 显示，为客户整机设计提供了更为灵活的选择。

- **完备的标准接口资源：**除了 8 路标准串口外，EM9280 还配置了以下标准接口，以满足不同应用需求。这些接口包括：（1）1 路 10M/100M 以太网接口；（2）USB2.0 高速主控接口及 USB2.0 OTG 接口；（3）1 路 SPI 接口；（4）1 路 I2C 接口总线；（5）4 路 PWM 输出；（6）2 路 AD 输入；（7）32 位 GPIO。

- **紧凑的外形尺寸：**EM9280 的外形尺寸继续保持了经典的 74mm×53mm 规格，该规格是业界尺寸最小的 ARM9 工控主板之一，模块采用坚固的 IDC 插针，可非常方便的插入用户的产品底板上，快速搭建各种工控产品。

- **极高性价比：**EM9280 的售价与英创的现有产品 EM9160 完全一样（百片批量价仅￥350/片），因此其性价比得到进一步的提升，与其他同类的 ARM9 产品相比具有强劲的竞争力。特别适合运用于运行环境恶劣，无人值守、连续 24 小时工作、对成本敏感的各种应用领域。同时也可作为 EM9160、EM9161 的升级换代产品。

- **开发门槛低：**作为工控主板产品，EM9280 将预装操作系统（CE6.0 或 Linux-2.6.35）以及标准的驱动程序接口（API），使客户无需了解主板内部的技术细节，就可充分利用其功能为自身产品服务。无论是微软的 Visual Studio 2005（或后续版本），还是开源的 Eclipse IDE，都是业界主流的开发工具，且很容易掌握其基本的使用方法。用户只要掌握 C/C++的基本编程手段（包括多线程设计），熟悉自身产品的功能需求，就可顺利完成应用程序的开发。使用 EM9280，并不一定需要客户具备 CE 或 Linux 操作系统的专门知识，因此说 EM9280 的应用开发门槛是很低的，可满足各种应用需求，各种的开发团队使用。

本手册详细介绍了 EM9280 的硬件配置、管脚定义及相关的技术指标，供用户使用时备查。此外，英创公司针对评估底板的使用编写有《EM9280 开发评估底板手册》。这两个手册都包含在英创为用户提供的产品开发光盘里面，用户也可以登录英创公司的网站下载相关资料的最新版本。

用户还可以访问英创公司网站或直接与英创公司联系以获得 EM9280 的其他相关资料。英创信息技术有限公司联系方式如下：

地址：成都市高新区高朋大道 5 号博士创业园 B 座 404# 邮编：610041

联系电话：028-86180660 传真：028-85141028

网址：http://www.emtronix.com 电子邮件：support@emtronix.com

注意：本手册的相关技术内容将会不断的完善，请客户适时从公司网站下载最新版本的数据手册，恕不另行通知。

目　　录

1、主要技术指标

核心单元

- 454MHz 主频的 ARM9 CPU
- 核心芯片为 Freescale 的 iMX283
- 128MB DDR2 系统内存，用户可用空间大于 100MB
- 128MB FLASH 存储器，其中用户文件空间 75MB
- USB 接口支持 U 盘即插即用
- 实时时钟 RTC，具有掉电保护功能
- 硬件看门狗（WDT），防止系统死锁
- 专用调试串口（115200，8-N-1）

串口通讯配置

- 8 路标准 UART 串口，其中 5 路为高速串口，波特率可达 3Mbps
- 各路串口基本特性如下：

CE 名称	Linux 名称	串口速度	功能简要说明
COM2	ttyS1	高速串口	支持 RTS/CTS 硬件流控。
COM3	ttyS2	高速串口	3 线制，RS232 电平接口。
COM4	ttyS3	高速串口	3 线制，TTL 电平。
COM5	ttyS4	高速串口	3 线制，TTL 电平。
COM6	ttyS5	高速串口	3 线制 TTL 电平。与 GPIO 复用管脚。
COM7	ttyS6	低速串口	3 线制，波特率不高于 19200bps，可作为 RS485 通讯应用。
COM8	ttyS7	低速串口	
COM9	ttyS8	低速串口	

其他通讯接口

- 1 路以太网接口，10M/100M 自适应
- 1 路 USB 高速主控接口（HOST）
- 1 路 USB OTG 接口，支持微软的 ActiveSync 通讯协议

- 1 路 I2C 接口，主控模式，最高波特率 400kbps，与 GPIO 复用管脚
- 1 路 SPI 接口，主控半双工模式，最高波特率 10Mbps，与 GPIO 复用管脚
- 1 路音频播放数字接口，I2S 数据格式，与 GPIO 复用管脚

显示单元

- 缺省配置为 TFT 彩色 LCD 接口（RGB 各 6-bit + 同步时钟信号）
- 彩色显示，分辨率从 480×272 至 1024×768 均可支持（使用 CN3）
- 单色显示，分辨率为 160×160，4-bit 灰度级（使用 CN4）
- 对彩色显示，支持 4 线制电阻触摸屏。

数字及模拟监控单元

- 32 位通用 GPIO0 – GPIO31，各位方向独立可控。
- 部分 GPIO 与系统的其他通讯功能复用管脚。
- 支持中断触发的数字输入状态监测功能。
- GPIO24 – GPIO27 支持外部中断触发功能，上升沿有效。
- 2 路低速 AD 采集通道，AD 分辨率为 12-bit。
- 支持对主板环境温度、供电电压的实时监测。

电源及模块机械参数

- 供电电压：+5V±5%，平均工作电流 170mA
- 工作温度：-10℃至 60℃；工业级（-40℃至 80℃）可选
- 模块外形尺寸：74mm×53mm
- 2 个 36 芯坚固 IDC 双排插针（0.1”）对称分布于模块的两侧
- 独立 LCD 显示接口，ZIF40 插座，英创标准信号定义。

CE 平台基本软件环境

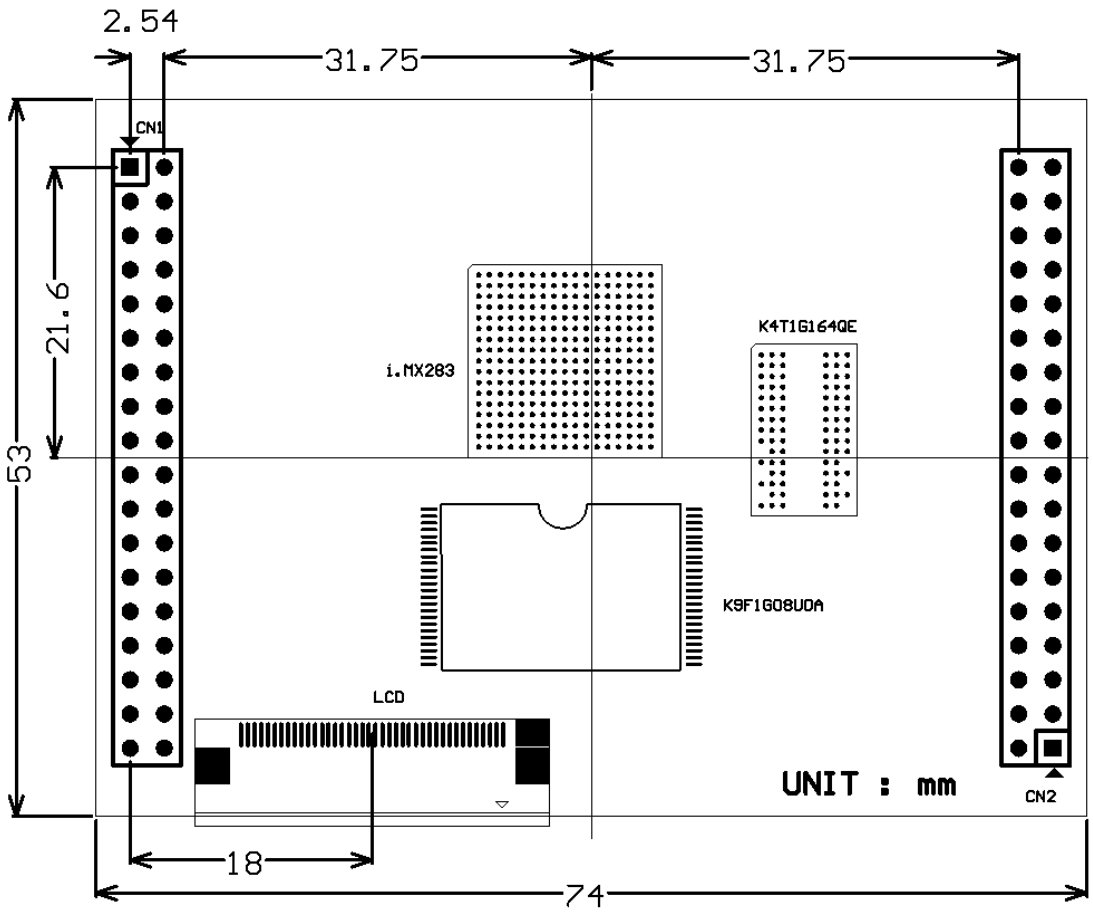
- 预装 Windows CE6.0 实时多任务操作系统
- 提供相应 SDK 开发包，包括各种接口驱动程序 API
- 支持 VS2005 应用程序集成开发环境
- 采用 BinFS 文件系统，启动时间缩短至 7 秒水平。

- 支持以太网口（TCP/IP）、USB 口（ActiveSync）应用程序源码调试
- 支持 telnet、FTP、Web 等常规网络应用
- 支持 ActiveSync 方式的文件管理及微软的远程调试工具集。
- 支持用户自行修改开机启动画面
- 提供典型应用参考程序源码

Linux 平台基本软件环境

- 预装 Linux-2.6 操作系统，完备的设备驱动程序。
- 基于 Windows 平台的 eclipse 集成开发环境直接开发应用程序。
- 基于 Windows 平台的 NFS，让程序调试极为方便。
- 支持 Telnet、FTP 等常规系统调试管理手段。
- 支持用户自行修改开机启动画面。
- 精心安排的应用开发入门演示程序源码。
- 多种面向应用的典型应用框架程序源码。

2、外形尺寸

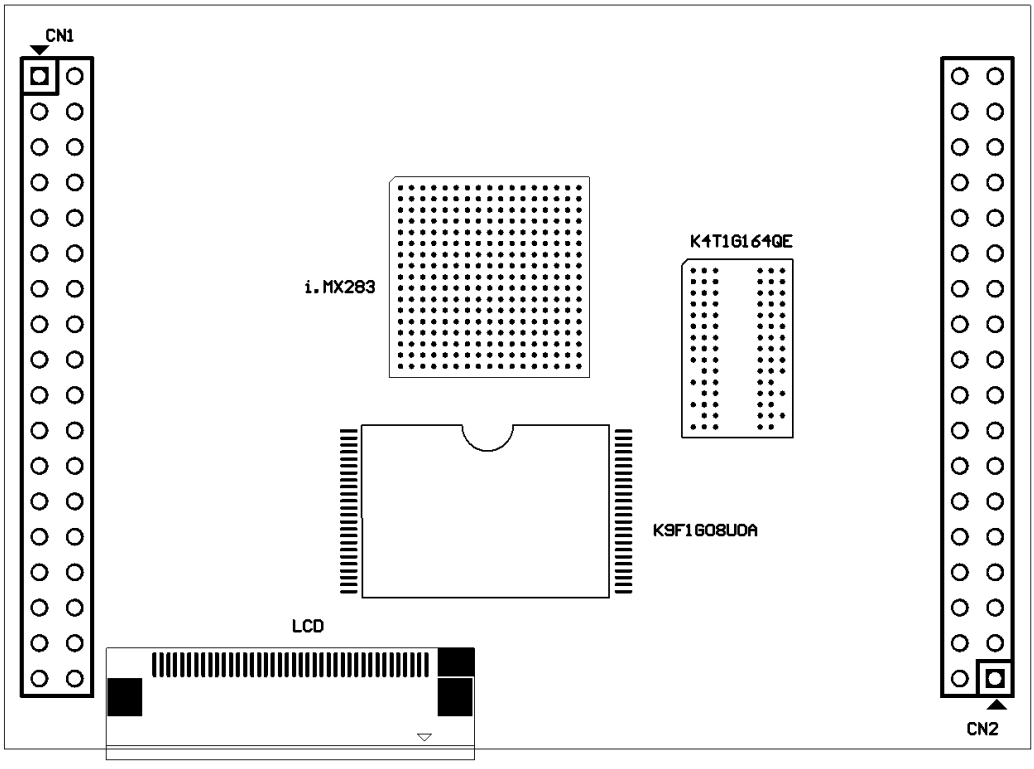


EM9280 外形尺寸示意图（2.54mm = 1”）

3、模块信号管脚功能描述

EM9280 的使用是以模块形式，通过板上的相关插针，插在应用主板上，同时实现 EM9280 板卡的固定以及与应用主板的信号连接两个功能。EM9280 共有 3 组信号插针，分别编号为 CN1、CN2 和 CN3，其中的 CN1 和 CN2 分别位于 EM9280 板卡模块的两端，为 2 组标准 0.1 英寸间距 IDC36 针双列直插管脚，EM9280 正是通过 CN1 和 CN2 与应用底板连接在一起的；EM9280 的 CN3 为 40 芯 ZIF 插座，主要引出彩色 LCD 及触摸屏的相关信号，实际应用中通过 40 芯的扁平软带线与 LCD 相连。为了方便起见，在后续对串口的说明中，均采用 CE 的串口名称（即 COM2…COM9）。

EM9280 所有管脚的信号电平，均为 LVTTL（3.3V）电平。除非特殊说明，输入管脚应避免接入 5V 电平信号。对低电平有效的信号，信号名称后均带“#”表示。



EM9280 的 CN1 – CN3 所在位置示意图

以下对 EM9280 所有管脚信号列表逐一说明。

3.1 EM9280 的 CN1 信号定义

EM9280 的 CN1 主要包括以太网接口、异步串口、USB、GPIO 等信号；而 CN2 主要包括数字 IO、USB_OTG 端口、电源输入等信号。CN1 和 CN2 的管脚编号均为奇偶排交错顺序编号，且 1#管脚标志为方形焊盘。

PIN#	信号名称	方向	描述
1	LINK#	O	以太网 Eth0 连接 / 传送指示，低电平有效。
2	SPEED#	O	以太网 Eth0 速度指示，低电平有效。
3	TPTX+	O	以太网差分输出信号
4	TPTX-	O	以太网差分输出信号
5	TPRX+	I	以太网差分输入信号
6	TPRX-	I	以太网差分输入信号
7	VDD_MCT	O	以太网口的网络变压器信号公共端
8	-		系统保留
9	RXD7	I	COM7 数据输入，TTL 电平（3.3V）
10	TXD7	O	COM7 数据输出，TTL 电平（3.3V）
11	RXD8	I	COM8 数据输入，TTL 电平（3.3V）
12	TXD8	O	COM8 数据输出，TTL 电平（3.3V）
13	RXD9	I	COM9 数据输入，TTL 电平（3.3V）
14	TXD9	O	COM9 数据输出，TTL 电平（3.3V）
15	USB1_HD+	I / O	USB 主控口的差分输入输出。
16	USB1_HD-	I / O	USB 主控口的差分输入输出。
17	AIN1	I	AD 输入通道 1，量程 0 – 3V，分辨率 12-bit。
18	AIN2	I	AD 输入通道 2，量程 0 – 3V，分辨率 12-bit。
19	RXD2	I	COM2 数据输入，TTL 电平（3.3V）
20	TXD2	O	COM2 数据输出，TTL 电平（3.3V）
21	COM3_RX	I	COM3 数据输入，RS232 电平（±9V）
22	COM3_TX	O	COM3 数据输出，RS232 电平（±9V）

23	RXD4	I	COM4 数据输入，TTL 电平（3.3V）
24	TXD4	O	COM4 数据输出，TTL 电平（3.3V）
25	RXD5	I	COM5 口数据输入，TTL 电平（3.3V）
26	TXD5	O	COM5 口数据输出，TTL 电平（3.3V）
27	GPIO0	I / O	通用数字 IO，与 COM2 口的 CTS#复用管脚。
28	GPIO1	I / O	通用数字 IO，与 COM2 口的 RTS#复用管脚。
29	GPIO2	I / O	通用数字 IO，支持中断触发的输入状态监测功能。
30	GPIO3	I / O	通用数字 IO，支持中断触发的输入状态监测功能。
31	GPIO4	I / O	通用数字 IO，支持中断触发的输入状态监测功能。
32	GPIO5	I / O	通用数字 IO，支持中断触发的输入状态监测功能。
33	GPIO6	I / O	通用数字 IO，与 PWM1 复用管脚。
34	GPIO7	I / O	通用数字 IO，与 PWM2 复用管脚。
35	GPIO8	I / O	通用数字 IO，支持中断触发的输入状态监测功能。
36	GPIO9	I / O	通用数字 IO，支持中断触发的输入状态监测功能。

关于 CN1 中相关信号的进一步说明：

为了提高管脚的利用率，以太网口的状态指示只提供单路低电平有效输出，需要外部提供 3.3V 偏置，才能点亮相应的 LED。为了提高整机的电磁兼容性能，通常情况下网络变压器应布局在客户应用底板上，且尽可能靠近网络的 RJ45 插座。

EM9280 的异步串口，在 CE 系统中的编号从 COM2 开始，8 路串口分别为 COM2 – COM9，其中 COM6 配置在 CN2，且与 GPIO10 - GPIO11 复用管脚。其他的串口均在 CN1，且采用专用管脚引出，以突出 EM9280 串口的可用性，同时也提高了 EM9280 的 GPIO 的利用率。在 Linux 系统中，串口的编号则从 ttyS1 开始，与 CE 串口的对应关系如下：

CE 名称	Linux 名称	串口速度	功能简要说明
COM2	ttyS1	高速串口	支持 RTS/CTS 硬件流控。
COM3	ttyS2	高速串口	3 线制，RS232 电平接口。
COM4	ttyS3	高速串口	3 线制，TTL 电平。
COM5	ttyS4	高速串口	3 线制，TTL 电平。

COM6	ttyS5	高速串口	3 线制 TTL 电平。与 GPIO 复用管脚。
COM7	ttyS6	低速串口	3 线制，波特率不高于 19200bps，仅支持 8-bit 数据位，可作为 RS485 通讯应用。
COM8	ttyS7	低速串口	
COM9	ttyS8	低速串口	

EM9280 的串口分成两类，其中 COM2 – COM6 为高速串口，其波特率可达 3Mbps；COM7 – COM9 为低速串口，波特率为 1200bps – 19200bps，数据位为 8-bit，支持奇偶校验、MARK / SPACE 设置。所以 COM7 – COM9 更适合作为 RS485 使用。在工业现场中的 RS485 接口，当传输距离较长时，往往采用硬件方向控制，来提高通讯的抗干扰能力。EM9280 支持选择 GPIO 作为硬件方向控制功能。具体操作方法为：

1. 通过 DeviceIoControl 来指定具体作为 RTS 的 GPIO 管脚。以 GPIO24 为例

```
#include “bsp_drivers.h”

DWORD dwRTSPin = GPIO24;
bRet = DeviceIoControl(m_hSer,          // file handle to the driver
    IOCTL_SET_UART_RTS_PIN,             // I/O control code
    &dwRTSPin,                          // in buffer
    sizeof(DWORD),                      // in buffer size
    NULL,                               // out buffer
    0,                                  // out buffer size
    NULL,                               // pointer to number of bytes returned
    NULL);                              // ignored (=NULL)
```

2. 设置控制参数：

dcb.fRtsControl = RTS_CONTROL_TOGGLE

可用作硬件 RTS 方向控制的 GPIO 管脚有：GPIO6 – GPIO7；GPIO20 – GPIO31。若应用程序选择其他 GPIO 作为 RTS，设置函数将返回 FALSE。

CN1 中的具有复用功能的 GPIO 如下表所示：

GPIO0	与 COM2 口的 CTS#复用管脚。
GPIO1	与 COM2 口的 RTS#复用管脚。
GPIO6	与 PWM1 复用管脚。

GPIO7	与 PWM2 复用管脚。

在缺省状态下，以上 GPIO 管脚均为数字输入，当应用程序打开 COM2 设备文件“COM2:”且设置控制参数 **dcb.fOutxCtsFlow** = TRUE 时，GPIO0 - GPIO1 将分别作为 CTS#和 RTS#。否则 COM2 为普通的三线制串口。对 PWM 输出，打开相应的设备文件（“PWM1:”、“PWM2:”）时，对应管脚将自动转为 PWM 输出功能，而不需要专门的切换操作。

SD 卡接口：EM9280 标准版为 8 个通用串口，通过硬件配置，EM9280 可以引出一路 SD 卡接口，占用串口 7、8、9 对应管脚，即 EM9280 SD 卡版的配置为一路 SD 卡接口，5 路高速串口。EM9280 最大支持 32G 的 SD 卡，SD 卡信号占用 CN1 的管脚如下：

CN1 管脚	EM9280 标准版	EM9280 SD 卡版
8	系统保留	SD_DET*
9	RXD7	SD_CMD
10	TXD7	SD_SCK
11	RXD8	SD_D0
12	TXD8	SD_D1
13	RXD9	SD_D2
14	TXD9	SD_D3

- SD_DET 是 SD 卡插入检测信号，高电平表示 SD 卡插入。

下表是 CN1 更直观的简要说明

信号名称及简要描述	CN1		信号名称及简要描述
	PIN	PIN	
LINK#，Eth0 连接/传送指示	1	2	SPEED#，Eth0 速度指示
TPTX+，以太网差分输出	3	4	TPTX-，以太网差分输出
TPRX+，以太网差分输入	5	6	TPRX-，以太网差分输入
VDD_CMT，网络变压器公共端	7	8	系统保留
RXD7，COM7 串行输入	9	10	TXD7，COM7 串行输出
RXD8，COM9 串行输入	11	12	TXD8，COM9 串行输出
RXD9，COM9 串行输入	13	14	TXD9，COM9 串行输出
USB1_HD+，USB1 Host 信号	15	16	USB1_HD-，USB1 Host 信号
AIN1，0 – 3V 量程	17	18	AIN2，0 – 3V 量程
RXD2，COM2 串行输入	19	20	TXD2，COM2 串行输出
COM3_RX，COM3 串入，232 电平	21	22	COM3_TX，COM3 串出，232 电平
RXD4，COM4 串行输入	23	24	TXD4，COM4 串行输出
RXD5，COM5 串行输入	25	26	TXD5，COM5 串行输出
GPIO0 / CTS2#（COM2）	27	28	GPIO1 / RTS2#（COM2）
GPIO2	29	30	GPIO3
GPIO4	31	32	GPIO5
GPIO6 / PWM1	33	34	GPIO7 / PWM2
GPIO8	35	36	GPIO9

3.2 EM9280 的 CN2 信号定义

EM9280 的 CN2 管脚，以数字 IO 作为其基本的功能，应用程序即可通过调用 EM9280 SDK 提供的 API 函数实现 DIO 操作。CN2 各管脚的定义如下：

信号名称及简要描述	CN2		信号名称及简要描述
	PIN	PIN	
+5V 电源输入	1	2	+5V 电源输入
USB_OTG_VBUS	3	4	RSTIN#，外部复位输入
电源地（GND）	5	6	电源地（GND）
USB_OTG_D+	7	8	USB_OTG_D-
USB_OTG_ID	9	10	BATT3V，3.3V 电池输入
DBG_COM_RX，232 电平	11	12	DBG_COM_TX，232 电平
RSTOUT#，复位输出	13	14	DBGSL#，调试模式选择输入
GPIO10 / RXD6，COM6 串入	15	16	GPIO11 / TXD6，COM6 串出
GPIO12	17	18	GPIO13
GPIO14	19	20	GPIO15
GPIO16	21	22	GPIO17
GPIO18	23	24	GPIO19
GPIO20 / PWM3	25	26	GPIO21 / PWM4
GPIO22 / I2C_SDA	27	28	GPIO23 / I2C_SCL
GPIO24 / IRQ1	29	30	GPIO25 / IRQ2
GPIO26 / IRQ3	31	32	GPIO27 / IRQ4
GPIO28 / SPI_MISO	33	34	GPIO29 / SPI_MOSI
GPIO30 / SPI_SCLK	35	36	GPIO31 / SPI_CS0N

关于 CN2 中相关信号的进一步说明：

DBGSL#信号用于选择系统启动的工作状态，DBGSL#设置为低并启动系统时，EM9280 将进入调试状态；DBGSL#设置为高或悬空并启动系统时，EM9280 将进入运行状态，若此时文件 userinfo.txt 包含有效信息，客户的应用的应用程序将被启动。

RSTIN#为对板卡的复位输入，不用时，可悬空。RSTOUT#为 EM9280 提供的复位输出脉冲，低有效，可程序控制，用于对扩展电路单元的复位。

RSTOUT#为对外的复位输出，平时为高电平，应用程序可调用系统 API 输出低电平脉冲的复位脉冲，脉冲宽度可设置。

USB OTG 端口，EM9280 包含一个标准 USB OTG 接口，共 5 条引线：

USB OTG 接口定义	简要说明
USB_OTG_D+	USB OTG 双向差分数据线
USB_OTG_D-	USB OTG 双向差分数据线
USB_OTG_VBUS	双向电源
GND	公共地
USB_OTG_ID	连接类型标志，带上拉电阻。

上述 5 条引线可直接接到底板的微型 AB 插座(mini-AB)。在通常情况下，若连接带线使 USB_OTG_ID 变低（即微型 A 插头），则 EM9280 将作为主控端；若连接带线使 USB_OTG_ID 保持高电平（即微型 B 插头），则 EM9280 将作为设备端。在实际使用中，USB OTG 将通过主机通信协议（HNP）根据实际连接的设备类型，动态切换主机和设备角色。因此即使 USB_OTG_ID 的电平与设备类型不符，同样可以实现正常连接。

当 EM9280 作为主控端时，将通过 USB_OTG_VBUS 向连接的 USB 设备提供+5V 电源，电流不超过 500mA。当 EM9280 作为设备端时，外部 USB 主控将通过 USB_OTG_VBUS 输入 5V 电源，但 EM9280 并不适用这个电源。

调试串口 DBG_COM，主要是用于输出系统的相关信息，在正常使用中不需要引出调试串口。但在开发阶段，调试串口的输出的信息是有帮助的。调试串口的电平为标准的 RS232 电平（±9V），波特率为 115200bps，数据帧格式为 8-N-1。

CN2 中的具有复用功能的 GPIO 如下表所示：

GPIO10	与 COM6 口的 RXD 复用管脚。COM6 设备文件名 = “COM6:”
GPIO11	与 COM6 口的 TXD 复用管脚。
GPIO20	与 PWM3 复用管脚。PWM3 设备文件名 = “PWM3:”
GPIO21	与 PWM4 复用管脚。PWM4 设备文件名 = “PWM4:”
GPIO22	与 I2C 总线的 SDA 复用管脚。I2C 设备文件名 = “I2C1:”
GPIO23	与 I2C 总线的 SCL 复用管脚。
GPIO24	与 IRQ1 复用管脚。IRQ1 设备文件名 = “IRQ1:”
GPIO25	与 IRQ2 复用管脚。IRQ2 设备文件名 = “IRQ2:”
GPIO26	与 IRQ3 复用管脚。IRQ3 设备文件名 = “IRQ3:”
GPIO27	与 IRQ4 复用管脚。IRQ4 设备文件名 = “IRQ4:”
GPIO28	与 SPI 接口的数据串入 MISO 复用管脚。SPI 设备文件名 = “SPI1:”
GPIO29	与 SPI 接口的数据串出 MOSI 复用管脚。
GPIO30	与 SPI 接口的同步时钟 SCLK 复用管脚。
GPIO31	与 SPI 接口的片选控制 CS0N 复用管脚。

在缺省状态下，以上 GPIO 管脚均为数字输入，当应用程序打开相应的设备驱动程序时，对应的管脚会自动切换到复用的功能管脚。

3.3 EM9280 的 CN3 信号定义

EM9280 的缺省显示模式为彩色 LCD 显示接口，CN3 插座主要是引出 LCD 显示输出信号以及引入触摸屏的模拟输入信号。具体信号定义如下：

PIN#	信号名称	方向	信号简要描述
1	GND	P	公共地
2	DCLK	O	串行像素时钟输出（Stream Pixel Clock）
3	HSYNC#	O	行同步脉冲，低有效
4	VSYNC#	O	场同步脉冲（或帧同步脉冲），低有效
5	GND	P	公共地
6-11	**R0 – R5**	**O**	**6-bit 红色分量输出信号，R0 为 LSB，R5 为 MSB。**
12	GND	P	公共地
13-18	**G0 – G5**	**O**	**6-bit 绿色分量输出信号，G0 为 LSB，G5 为 MSB**
19	GND	P	公共地
20-25	**B0 – B5**	**O**	**6-bit 蓝色分量输出信号，B0 为 LSB，B5 为 MSB**
26	GND	P	公共地
27	DE	O	显示使能控制信号
28-29	+3.3V	P	3.3V 电源输出，最大输出电流<200mA
30	BLIGHT#	O	背光控制信号，低电平有效；LCD 显示时有效。
31	-	O	输出固定高电平，系统保留。
32	GND	P	公共地
33-34	+5.0V	P	5V 电源输出，最大输出电流<200mA
35	GND	P	公共地
36	Xm	I	触摸屏 X 方向差分输入-
37	Xp	I	触摸屏 X 方向差分输入+
38	Ym	I	触摸屏 Y 方向差分输入-
39	Yp	I	触摸屏 Y 方向差分输入+
40	GND	P	公共地

关于 CN3 中相关信号的进一步说明：

- DCLK 下降沿更新 RGB 数据，上升沿用于显示设备锁存数据。
- LCD_PWR 信号也可用于 LCD 的背光电源控制。
- EM9280 支持的典型 LCD 显示格式包括：

  480×272，LCD 尺寸为 4.3”，具有很高的性价比；

  640×480，LCD 尺寸一般为 5.6” – 6.4”；

  800×480，LCD 尺寸为 7” – 8”；EM9280 缺省设置

  800×600，LCD 尺寸为 8.4” – 10.4”；一般需转为 LVDS 接口

  1024×768，LCD 尺寸为 10.4” – 12.1”；一般需转为 LVDS 接口
- 触摸屏的输入电阻一般在 200Ω 至 600Ω 这一范围。
- CN3 和 CN4 同一时间只能使用其中一个，否则可能造成电路损坏。

3.4 EM9280 的 CN4 信号定义

CN4 是专门针对 160×160 单色显示屏而设置的另一个 LCD 接口，该单色屏被广泛应用于低成本智能终端设备上。CN4 为 18 芯 0.5mm 间距单排插座，支持扁平带线直接连接到 LCD 模块上。具体信号定义如下：

PIN#	信号名称	方向	信号简要描述
1	GND	P	公共地
2	LCD_RS	O	命令寄存器 / 数据寄存器选择
3	LCD_WR#	O	LCD 写脉冲，低有效。
4	LCD_RD#	O	LCD 读脉冲，低有效。
5	LCD_CS#	O	LCD 读写总线周期片选信号，低有效。
6	LCD_RESET#	O	LCD 复位控制，低有效。
7	+3.3V	P	3.3V 电源输出，最大输出电流<100mA
8	LCD_D0	I / O	LCD 接口总线数据，D0 为 LSB
9	LCD_D1	I / O	LCD 接口总线数据
10	LCD_D2	I / O	LCD 接口总线数据
11	LCD_D3	I / O	LCD 接口总线数据
12	LCD_D4	I / O	LCD 接口总线数据
13	LCD_D5	I / O	LCD 接口总线数据
14	LCD_D6	I / O	LCD 接口总线数据
15	LCD_D7	I / O	LCD 接口总线数据，D7 为 MSB
16	BACKLIGHT#	O	背光控制信号，低电平有效；LCD 显示时有效。
17	NC	P	公共地
18	+3.3V	P	3.3V 电源输出，最大输出电流<100mA

注意：

- CN3 和 CN4 同一时间只能使用其中一个，否则可能造成电路损坏。
- CN4 在主板的背面，与 CN3 处于同侧。

4、基本电气特性与注意事项

在客户的应用设计中，EM9280 是作为整个系统的部件之一，与客户的应用底板、电源等其他部件协同工作的。因此在设计中，需详细了解 EM9280 各个管脚的电气特性，以做到系统各个部件间的各项指标的合理配合。

4.1 EM9280 的额定参数

参数名称	最小值	最大值	简要说明
+5V 直流瞬态输入	-0.3V	+7.0V	持续时间小于 30ms。
+5V 直流稳态输入	-0.3V	+7.0V	
GPIO 管脚输入电压	-0.3V	+3.63V	不兼容 5VTTL 电平输入。
GPIO/LCD 人体静电阈值	-	2kV	实际人体静电很容易超阈值。
CPU 基片工作温度	-40℃	85℃	应用程序可监测
CN3 插座电源输出功率	-	200mA	+5V 和+3.3V 二组电源输出
GPIO 信号总的驱动能力	-	±90mA	包括输入输出方式

4.2 RS232 输入输出特性

EM9280 的串口 COM3 缺省配置为 RS232 电平，其输入输出（RX / TX）特性如下表所示：

	Min（最小值）	**Max**（最大值）	简要说明
输入范围	-25V	25V	
输入负载	3kΩ	7kΩ	
输出电压	±5V	±9V	负载条件：3kΩ - 7kΩ

4.3 低速串口输入输出特性

低速串口 COM7 – COM9 的接口电平为 3.3VTTL，其 DC 电气参数如下表所示：

	Min（最小值）	**Max**（最大值）	简要说明
V_{IL}	0	1.0V	输入低电平

V_{IH}	2.3V	3.3V	输入高电平
V_{OL}	-	0.33V	输出低电平
V_{OH}	2.97V	-	输出高电平
I_{OH}	-4mA	-5mA	输出高电平时源电流
I_{OL}	4mA	10mA	输出低电平时吸电流

4.4 以太网口的基本参数

	典型值	简要说明
差分输出电压	2.0V	100BASE-TX 模式
差分输出电流	26mA	100BASE-TX 模式
差分输出电压	2.5V	100BASE-T 模式
VDD_MCT	3.3V	共模偏置电压，100Ω 终端电阻

4.5 3.3V TTL 信号的基本参数

EM9280 共引出 32 位通用数字 IO(也称为 GPIO),均为 3.3V TTL 电平。此外,EM9280 的 COM2、COM4、COM5 的 RXD 和 TXD 也为 3.3VTTL 电平信号，其 DC 电气特性与 EM9280 的 GPIO 是完全一致的。这些信号管脚的具体电气参数如下表所示：

	Min（最小值）	**Max**（最大值）	简要说明
V_{IL}	-	0.8V	输入低电平
V_{IH}	2.0V	3.3V	输入高电平
V_{OL}	-	0.4V	输出低电平
V_{OH}	2.4V	-	输出高电平
I_{OH}	-8mA	-	输出高电平时源电流
I_{OL}	8mA	-	输出低电平时吸电流
I_{IL}	-	10uA	输入低电平时的泄漏电流
I_{IH}	-	10uA	输入高电平时的泄漏电流

EM9280 的复位输出 RSTOUT#，也为 3.3VTTL 电平，其输出特性如下：

	Min（最小值）	**Max**（最大值）	简要说明

V_{OL}	-	0.25V	输出低电平，I_{OL} = 10mA
V_{OH}	2.8V	-	输出高电平，I_{OH} = -10mA

4.6 设计注意事项

1. EM9280 的核心 CPU 芯片 iMX283 内部还包含了一个电源管理单元，正是利用该电源管理单元使 EM9280 获得很高的性能价格比。对接入 EM9280 的+5V 电源有以下要求：电源上电时的电压过冲脉冲时间小于 30ms，同时脉冲的占空比小于 0.05%。例如，过冲脉冲的脉宽为 100us，则脉冲周期需大于 200ms。长时间过电压施加在 EM9280 上，可能造成核心芯片电源单元的损坏。
2. EM9280 上 CN1 – CN3 的大部分 LVTTL 信号均直接来自于系统的核心 CPU 芯片 iMX283，包括 GPIO 信号、LCD 的信号。它们抗人体静电的能力只有 2kV，这不是一个很高的阈值，冬季人体静电达到 4-5kV 是很容易发生的。
3. EM9280 的 GPIO 管脚不是 5V 输入兼容的，尽管在通电状态下接入个别 5V 电平信号不会影响系统工作。但若长时间接入 5V 信号，不能保证信号管脚不被损坏。此外在 EM9280 上电前，若接入 5V 电平信号的管脚较多，还可能会影响系统正常启动。
4. CN3 是 LCD 的专用插座，为了方便 LCD 屏的连接，CN3 上包含了+5V 和+3.3V 的电源输出，可满足大部分 LCD 屏的信号接口电路的需要。在安装扁平带线时，需特别注意管脚的一一对应及可靠的接触。信号管脚错位，可能会导致电源输出被短接，从而引起 EM9280 的损坏。
5. 尽管单个 GPIO 的驱动能力能够达到±8mA，但仍需在设计中应避免 GPIO 总的输入输出电流和超过额定驱动能力的阈值。长时间超阈值可能会导致 GPIO 管脚的损坏。对有可能存在超驱动能力阈值的应用，强烈建议在应用底板上增加驱动芯片（如 74HC245），通过把电流负载转移到驱动芯片上，来保护 EM9280 的 GPIO 管脚。
6. EM9280 的 USB 接口，在拔插过程中，会产生瞬间的浪涌电压，该电压有可能损坏 EM9280 的 USB 数据收发单元，因此强烈推荐客户的应用底板参考 EM9280 开发评估底板的相关电路，在 USB 接口处增加 ESD 保护芯片，并在电源回路中串入磁珠。

5、EM9280 的相关功能的说明

5.1 WDT 看门狗定时器

EM9280 直接使用了 iMX283 芯片内部的独立看门狗定时器，系统启动后设置看门狗的超时时间为 10 秒，且由 WinCE 内核的 Watchdog 线程按 5 秒间隔对看门狗进行刷新。此模式可以防止应用程序占用 CPU 的死循环，但对应用程序异常退出或挂起没有作用。

EM9280 为应用程序设计了专门的 WDT 驱动程序，应用程序可通过打开 WDT 设备文件“WDT1:”来接管对看门狗的操作，使之更为全面的监管应用程序行为的有效性。应用程序接管看门狗后，需按 5 秒的间隔对看门狗进行刷新操作。用户应用程序可通过 Read 函数来获取 WDT 加载周期，通过 Write 函数执行看门狗刷新操作，主要代码如下：

打开 WDT 文件

```
HANDLE hWDT;
hWDT = CreateFile(_T("WDT1:"),                  // name of device
    GENERIC_READ|GENERIC_WRITE,                 // desired access
    FILE_SHARE_READ|FILE_SHARE_WRITE,           // sharing mode
    NULL,                                       // security attributes (ignored)
    OPEN_EXISTING,                              // creation disposition
    FILE_FLAG_RANDOM_ACCESS,                    // flags/attributes
    NULL);                                      // template file (ignored)
```

获取 WDT 刷新周期

```
DWORD     dwWDTPeriod;                          //in ms
DWORD     dwNumberOfBytesRead = 0;
BOOL      bRet;

bRet = ReadFile(hWDT, &dwWDTPeriod, sizeof(DWORD),
                  &dwNumberOfBytesRead, NULL);
```

执行 WDT 刷新操作

```
DWORD     dwNumberOfBytesWritten = 0;
BOOL      bRet;

bRet = WriteFile(hWDT, NULL, 0, &dwNumberOfBytesWritten, NULL);
```

5.2 USB 接口

EM9280 可提供 2 个 USB 端口：一个高速主控接口，和一个 USB OTG 接口。EM9280 的 USB 主控接口可直接与标准 U 盘相连，EM9280 会自动把 U 盘中的系统配置文件 userinfo.txt 拷贝到系统中，并按照 userinfo.txt 设置 IP 等参数，最后启动用户的应用程序。USB 主控口也可支持标准的键盘、鼠标等设备。EM9280 的 USB OTG 接口，即可作为 USB 主控接口使用，也可作为 USB 设备接口使用。作为 USB 设备接口的一个典型应用，就是支持 Microsoft 的 ActiveSync 传输协议，用户可利用它方便的实现对 EM9280 文件的管理，也可以利用 ActiveSync 来调试应用程序。另外 ActiveSync 还把 USB 设备口映射成串口，占用串口逻辑号 COM1，所以 EM9280 真正的物理串口对应的逻辑编号从 COM2 开始。主控 USB 的供电电路很简单，布置在 EM9280 的评估底板上，客户在设计自己的应用底板时，可参考该电路。

5.3 UART 异步串口

EM9280 物理上有 8 个串口， 8 个物理串口分别对应的逻辑编号为 COM2 – COM9，其中只有 COM6 口与 GPIO10 – 11 复用管脚，其他的 7 个串口均配置在 CN1 上，且具有独立使用的信号管脚。EM9280 的这种设计主要是充分发挥其多串口的功能，另一方面，由于串口与 GPIO 的复用很少，相应的也提高了 GPIO 的利用率。此外 EM9280 板上还保留了调试串口的引出插针。调试串口的波特率固定为 115200bps，帧格式则为 8-N-1，主要用于系统输出相关信息，以便于系统的维护，用户原则上可以不关心它。

EM9280 的 8 个串口按最高波特率分为高速串口 COM2 – COM6 和低速串口 COM7 – COM9。高速串口的最高波特率可达 3Mbps，而低速串口允许的波特率在 1200bps – 19200bps 之间，数据固定为 8-bit，支持奇偶校验、MARK / SPACE 设置。因此低速串口更适合作为 RS485 端口来应用。EM9280 在 RS485 驱动方面，除了可以采用 TXD 自动控制数据收发方向切换（具体电路请参考 EM9280 开发评估底板电路原理图）外，还可选择一位 GPIO 作为 RTS，实现硬件方向控制。可作为 RTS 硬件方向控制的 GPIO 有：GPIO6、GPIO7、GPIO20 – GPIO31。在应用软件方面，需要主要代码如下：

```
打开串口设备文件

HANDLE hSer;
hSer = CreateFile(_T("COM7:"),                  // name of device
    GENERIC_READ|GENERIC_WRITE,          // desired access
    FILE_SHARE_READ|FILE_SHARE_WRITE,// sharing mode
```

```
        NULL,                                   // security attributes (ignored)
        OPEN_EXISTING,                          // creation disposition
        FILE_FLAG_RANDOM_ACCESS,                // flags/attributes
        NULL);                                  // template file (ignored)
```

设置一位 GPIO 作为 RTS

```
DWORD dwRtsGpioPin = GPIO26;                    //选择 GPIO26 作为 RTS

If（!DeviceIoControl（hSer,
                        IOCTL_SET_UART_RTS_PIN,
                        & dwRtsGpioPin, sizeof(DWORD),
                        NULL, 0,
                        NULL, NULL））
{
    // 出错处理。。。
}
```

设置串口 RTS 控制模式

```
DCB    SerDCB;

SerDCB.DCBlength = sizeof(DCB);
GetCommState(hSer, &SerDCB );                   // 从驱动读取当前DCB
SerDCB.fRtsControl = RTS_CONTROL_TOGGLE;
SetCommState(hSer, &SerDCB );                   // 再设置回驱动
```

高速串口中，只有 COM2 配置有 RTS/CTS 硬件握手功能，而其他都是常规的三线制串口。由于 RTS/CTS 硬件握手功能的应用并不是很多，同时考虑充分利用 GPIO 的功能，在打开“COM2:”时，RTS/CTS 硬件握手功能并没有激活，而对应管脚 GPIO0、GPIO1 继续保持为 GPIO 状态。应用程序需通过设置才能激活 RTS/CTS 硬件握手功能：

激活串口 RTS/CTS 硬件握手功能

```
DCB    SerDCB;

SerDCB.DCBlength = sizeof(DCB);
GetCommState(hSer, &SerDCB );                   // 从驱动读取当前DCB
SerDCB.fRtsControl = RTS_CONTROL_HANDSHAKE;
SetCommState(hSer, &SerDCB );                   // 再设置回驱动
```

5.4 I²C 接口

EM9280 的 I²C 接口为 2 线制标准 I²C 接口，信号电平为 3.3V 的 TTL 电平（LVTTL），最高传输波特率为 400kbps。在使用 I2C 接口时，应对 SCL 和 SDA 两个信号线均加 10K 的上拉电阻，在高波特率的情况下，上拉电阻是必须的。EM9280 板上已固化了面向 I²C 接口的 WinCE 标准驱动程序，应用程序只需打开文件名为“I2C1:”的文件对象，就可通过标准的 ReadFile（…）和 WriteFile（…）函数进行 I²C 数据传输了。

基本的 I²C 数据结构如下：

```
typedef struct
{
    BYTE    uHwAddr;    // 7-bit I2C器件地址 + 1-bit 读写标志（LSB）
    DWORD dwCmd;        // 对I2C器件发送的命令，可选
    PBYTE   pDatBuf;    // 指向存储读写数据的buffer
    DWORD dwDatLen;     // 需要读写数据的字节长度
} I2C_INFO, *PI2C_INFO;
```

在上述结构中，dwCmd = 0xFFFFFFFF，表示无效命令，驱动程序会跳过命令的发送；若 dwCmd 的最高位（D31）= 0，表示为单字节命令，最低字节（D7 – D0）将作为命令被发送；若 dwCmd 的最高位（D31）= 1，表示为双字节命令，驱动程序会首先发送高字节（D15 – D8），然后再发送低字节（D7 – D0）命令。dwCmd 通常为 I²C 器件寄存器的起始地址。I²C 操作的主要代码如下：

打开 I²C 文件

```
HANDLE hI2C;
hI2C = CreateFile(_T("I2C1:"),                  // name of device
    GENERIC_READ|GENERIC_WRITE,                 // desired access
    FILE_SHARE_READ|FILE_SHARE_WRITE,           // sharing mode
    NULL,                                       // security attributes (ignored)
    OPEN_EXISTING,                              // creation disposition
    FILE_FLAG_RANDOM_ACCESS,                    // flags/attributes
    NULL);                                      // template file (ignored)
```

从 I²C 器件读取数据

```
I2C_INFO    I2cInfo;
DWORD       dwNumberOfBytesRead = 0;
BOOL        bRet;
```

```
// I2C 数据结构初始化，注意数据 buffer 指针一定要指向有效 buffer
// … …
bRet = ReadFile(hI2C, & I2cInfo, sizeof(I2cInfo),
                    &dwNumberOfBytesRead, NULL);
```

向 I^2C 器件写入数据

```
I2C_INFO    I2cInfo;
DWORD       dwNumberOfBytesWritten = 0;
BOOL        bRet;

// I2C 数据结构初始化，注意数据 buffer 指针一定要指向有效 buffer
// … …
bRet = WriteFile(hI2C, &I2cInfo, sizeof(I2cInfo),
                    &dwNumberOfBytesWritten, NULL);
```

用户可从 EM9280 的资料光盘中的 I2C 应用范例了解其详细的使用方法。

5.5 SPI 同步串口

EM9280 的 SPI 接口为 4 线制标准 SPI 接口，信号电平为 3.3V 的 TTL 电平（LVTTL），最高传输波特率为 10Mbps。主要应用于设备内部各功能单元之间的短距离高速传输。EM9280 板上已固化了面向 SPI 接口的 WinCE 标准驱动程序，应用程序只需打开文件名为“SPI1:”的文件对象，就可通过 SPI 进行数据传输了。用户可从 EM9280 的资料光盘中的 SPI 应用范例了解其详细的使用方法。

5.6 PWM 脉冲输出

EM9280 共有 4 路 PWM 输出，其最高输出频率可达 12MHz，但如果希望保证一定精度的占空比（1%的精度），则输出最高频率只能到 240KHz。EM9280 板上已固化了面向 PWM 接口的 WinCE 标准驱动程序，应用程序只需打开文件名为“PWM1:”- “PWM4:”的文件对象，再通过 WriteFile 设置启动 PWM 脉冲的参数（频率和占空比）即可，应用程序也可通过 WriteFile 随时停止 PWM 的输出。典型的 PWM 应用，包括为红外串口提供调制信号（38.5KHz，50%占空比）、为 ISO7816 提供时钟信号（3.5712MHz，9600bps 波特率）。

基本的 PWM 数据结构如下：

```
typedef struct
{
    DWORD    dwFreq;          // 单位为Hz, = 0: 停止PWM输出
    DWORD    dwDuty;          // 取值范围与分辨率有关，单位1% - 0.01%
    DWORD    dwResolution;    // 分辨率，取值 = 1、10、100
} PWM_INFO, *PPWM_INFO;
```

上述结构中，参数 dwFreq 表示输出的脉冲频率，单位为 Hz。dwFreq 的取值范围 10Hz - 12000000Hz（12MHz），当 dwFreq = 0 时，表示 PWM 停止输出。分辨率 dwResolution 参数决定占空比参数 dwDuty 的取值范围，具体关系如下表所示：

dwResolution	dwDuty 取值范围	占空比单位
1	1 – 99	%
10	1 – 999	0.1%
100	1 – 9999	0.01%

注意：只有在输出频率较低时，高分辨率占空比才有意义。PWM 操作的主要代码如下：

打开 PWM 文件

```
HANDLE hPWM;
hPWM = CreateFile(_T("PWM1:"),                  // name of device
    GENERIC_READ|GENERIC_WRITE,                 // desired access
    FILE_SHARE_READ|FILE_SHARE_WRITE,           // sharing mode
    NULL,                                       // security attributes (ignored)
    OPEN_EXISTING,                              // creation disposition
    FILE_FLAG_RANDOM_ACCESS,                    // flags/attributes
    NULL);                                      // template file (ignored)
```

执行 PWM 输出操作

```
PWM_INFO  PwmInfo;
DWORD     dwNumberOfBytesWritten = 0;
BOOL      bRet;

PwmInfo.dwFreq = 3571200;          // ISO7816 时钟：3.5712MHz
PwmInfo.dwDuty = 50;               // 占空比 50%
PwmInfo.dwResolution = 1;
bRet = WriteFile(hPWM, &PwmInfo, sizeof(PwmInfo),
                 &dwNumberOfBytesWritten, NULL);
```

获取 PWM 设置的实际参数

```
PWM_INFO  PwmInfo;
DWORD     dwNumberOfBytesRead = 0;
BOOL      bRet;

bRet = ReadFile(hPWM, & PwmInfo, sizeof(PwmInfo),
                  &dwNumberOfBytesRead, NULL);
```

关闭设备文件，将停止 PWM 脉冲输出。

5.7 IRQ 外部中断

EM9280 共有 4 路外部中断输入 IRQ1 – IRQ4，分别与 GPIO24 – GPIO27 复用管脚。当应用程序打开 IRQ 驱动程序对应的设备文件“IRQ1:”- “IRQ4:”后，外部中断输入上升沿正脉冲，脉冲宽度大于 100us，驱动程序将响应该上升沿中断，并产生事件通知处于等待中的应用线程。典型代码包括：

打开 IRQ 文件

```
HANDLE hIRQ;
hIRQ = CreateFile(_T("IRQ1:"),                  // name of device
    GENERIC_READ|GENERIC_WRITE,          // desired access
    FILE_SHARE_READ|FILE_SHARE_WRITE,// sharing mode
    NULL,                                          // security attributes (ignored)
    OPEN_EXISTING,                           // creation disposition
    FILE_FLAG_RANDOM_ACCESS,          // flags/attributes
    NULL);                                         // template file (ignored)
```

等待 IRQ 时间子程序

```
DWORD WaitIRQEvent（HANDLE hIRQ, DWORD dwTimeoutMS）
{
    DWORD dwReturn = 0;

    If（!DeviceIoControl（hIRQ,
                          IOCTL_WAIT_FOR_IRQ,
                          &dwTimeouMS, sizeof(DWORD),   //超时时间单位为 ms
                          &dwReturn, sizeof(DWORD),
                          NULL, NULL））
    {
        //出错
        dwReturn = WAIT_FAILED;
    }
    Return dwReturn;
```

```
}
```

应用线程等待中断事件

```
DWORD dwTimeoutMS = 5000;          //超时时间设置为 5 秒
DWORD dwReturn;

dwReturn = WaitIRQEvent（hIRQ, dwTimeoutMS）;
if（dwReturn == WAIT_OBJECT_0）
{
    //外部中断发生，进行中断处理
    //… …
}
else if（dwReturn == WAIT_TIMEOUT）
{
    //超时处理
    //… …
}
else
{
    //出错处理
    //… …
}
```

5.8 AD 模拟通道

EM9280 共有 2 路低速的模拟 AD 通道 AIN1 和 AIN2，输入量程为 0 – 3.6V，AD 分辨率 12-bit。所谓低速通道，表示这两个通道只能用于外部的直流或慢变化类型的信号。除此之外，EM9280 还可提供对输入的+5V 电源电压、EM9280 板卡本身运行的环境温度以及核心 CPU 基片温度的监测。应用程序在打开 AD 驱动程序对应的设备文件“LCD1:”后，可多次调用 ReadFile 来读取各类数据。需要设置的命令及参数如下：

```
#define  EM9280_DAQ_VOLTAGE_CH0          0
#define  EM9280_DAQ_VOLTAGE_CH1          1
#define  EM9280_DAQ_VDD_5V               2
#define  EM9280_DAQ_CPU_TEMPERATURE      6
#define  EM9280_DAQ_BOARD_TEMPERATURE    7

typedef struct
{
    DWORD    dwCmd;        // 命令码 = 0, 1, 2, ....
    DWORD    dwData;       // 返回的AD数据
```

```
    char        UnitName[16];    // 返回的单位字串: "mV", "Kalvin"等
} DAQ_INFO, *PDAQ_INFO;
```

注意返回的温度参数均为开氏温度，转换成摄氏温度，大致减去 273 即可。在此基础上，应用程序的典型代码如下：

打开 AD 文件

```
HANDLE hLRADC;
hLRADC = CreateFile(_T("LDC1:"),                    // name of device
    GENERIC_READ|GENERIC_WRITE,                     // desired access
    FILE_SHARE_READ|FILE_SHARE_WRITE,               // sharing mode
    NULL,                                           // security attributes (ignored)
    OPEN_EXISTING,                                  // creation disposition
    FILE_FLAG_RANDOM_ACCESS,                        // flags/attributes
    NULL);                                          // template file (ignored)
```

读取模拟输入通道数据

```
DAQ_INFO   DaqInfo;
DWORD      dwNumberOfBytesRead = 0;
BOOL       bRet;
float      f1;
char       Unit[16];

DaqInfo.dwCmd = EM9280_DAQ_VOLTAGE_CH0;    //读取AIN1通道数据
dwNumberOfBytesRead = 0;
bRet = ReadFile(hLRADC, &DaqInfo, sizeof(DAQ_INFO),
                &dwNumberOfBytesRead, NULL);
if(!bRet)
{
    //出错处理
    //… …
}
sscanf(DaqInfo.UnitName, "%f%s", &f1, Unit);
printf("AIN1 = %d(%s) -> %.1f%s\n", DaqInfo.dwData, DaqInfo.UnitName,
                                    (DaqInfo.dwData * f1), Unit);
```

在读取 AIN1 和 AIN2 通道的数据时，返回的 dwData 仅仅是 AD 的量化数据，在返回的单位字串中，包括了量化电压间隔 f1 和单位，所以需要用 sccanf 解析之。实际测得的输入电压则为 DaqInfo.dwData * f1。

读取板卡环境温度数据

```
DaqInfo.dwCmd = EM9280_DAQ_BOARD_TEMPERATURE;  //读取板卡温度
dwNumberOfBytesRead = 0;
bRet = ReadFile(hLRADC, &DaqInfo, sizeof(DAQ_INFO),
                    &dwNumberOfBytesRead, NULL);
```

返回的数据 dwData 为开氏温度，大致减去 273 即为摄氏温度。EM9280 是利用二极管的温度特性进行测温的。由于二极管器件的离散性，精度不是很高，一般误差在±3 度。在实际应用中，一般只有当环境温度很高时（比如 45℃或以上），才有温度检测的意义。因此该功能对设备运行于高温环境是有积极意义的。

5.9 GPIO 通用数字 IO

EM9280 的 32 位 GPIO0 – GPIO31 均为可独立方向可设置的通用数字 IO，所有 GPIO 的上电初始状态均为输入状态带上拉电阻。EM9280 为应用程序提供了操作 GPIO 的驱动程序，其设备文件名为“PIO1:”。有关 GPIO 操作的使用方法在相应的范例代码中有详细的中文说明，这里不再赘述。EM9280 为了保持模块的紧凑尺寸及机械强度，其 GPIO 与主板的其他接口功能采用了管脚复用的设计，具体复用情况如下表所示：

管脚#	复用功能 1	简要说明
GPIO0	CTS2#	与 COM2 口的 CTS#复用管脚。
GPIO1	RTS2#	与 COM2 口的 RTS#复用管脚。
GPIO2		
GPIO3		
GPIO4		
GPIO5		
GPIO6	PWM1	与 PWM1 复用管脚。
GPIO7	PWM2	与 PWM2 复用管脚。
GPIO8		
GPIO9		
GPIO10	RXD6	与 COM6 口的 RXD 复用管脚。
GPIO11	TXD6	与 COM6 口的 TXD 复用管脚。
GPIO12		
GPIO13		
GPIO14		
GPIO15		
GPIO16		
GPIO17		
GPIO18		
GPIO19		
GPIO20	PWM3	与 PWM3 复用管脚。
GPIO21	PWM4	与 PWM4 复用管脚。

GPIO22	I2C_SDA	与 I2C 总线的 SDA 复用管脚。
GPIO23	I2C_SCL	与 I2C 总线的 SCL 复用管脚。
GPIO24	IRQ1	与 IRQ1 复用管脚。
GPIO25	IRQ2	与 IRQ2 复用管脚。
GPIO26	IRQ3	与 IRQ3 复用管脚。
GPIO27	IRQ4	与 IRQ4 复用管脚。
GPIO28	SPI_MISO	与 SPI 接口的数据串入 MISO 复用管脚。
GPIO29	SPI_MOSI	与 SPI 接口的数据串出 MOSI 复用管脚。
GPIO30	SPI_SCLK	与 SPI 接口的同步时钟 SCLK 复用管脚。
GPIO31	SPI_CS0N	与 SPI 接口的片选控制 CS0N 复用管脚。

在系统启动后的初始状态，所有的 GPIO 都是有效的，一旦应用程序打开某个接口的设备文件，则对应的 GPIO 功能将被禁止。注意即使应用程序关闭了设备文件，对应的 GPIO 功能同样是被禁止的。因为在嵌入式系统中，不可能存在一条管脚动态复用的情况。