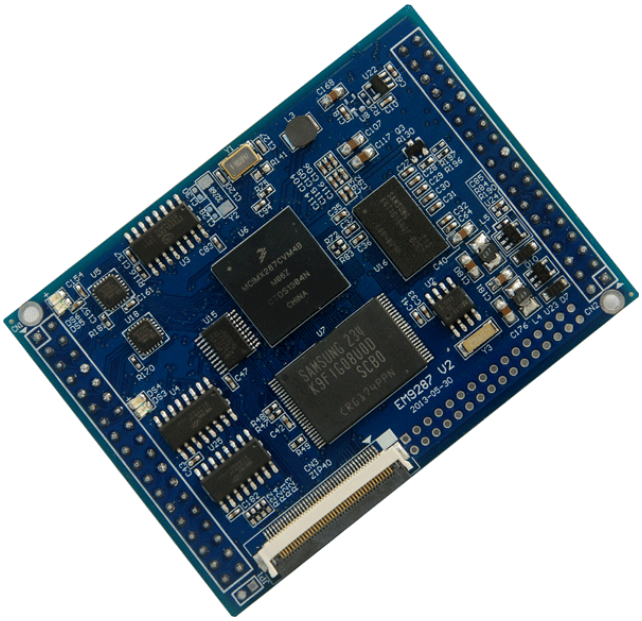
Emtronix



# EM9287 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：**EM9287 工控主板。**

EM9287 是一款面向工业自动化领域的高性价比嵌入式主板，以丰富的通讯接口为特色。EM9287 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 EM9287 可选择预装 Windows CE6.0（R3）或 Linux-3.9.7 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于 EM9287。英创公司针对 EM9287 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。

EM9287 主要特点：

- **双以太网接口：**EM9287 支持 2 路独立的以太网接口（10M/100M 自适应），双网口配置使 EM9287 特别适合运用于工业设备的通讯、管理领域。454MHz 主频的 CPU（ARM926EJ-S 内核）可满足绝大部分应用中对网络数据的及时处理。

- **双 CAN 与多串口：**EM9287 可支持 2 路 CAN 总线接口，以及多达 7 路标准串口。CAN 与串口的灵活配置组合可满足目前大多数工控通讯领域对现场总线的应用需求，从而加快客户整机产品的开发速度，同时降低其成本。

- **完备的标准接口资源：**除了上述通讯接口外，EM9287 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）2 路 USB 主控接口及 1 路 USB OTG 接口；（2）1 路 SPI 接口；（3）1 路 I2C 接口总线；（4）4 路 PWM 输出；（5）2 路 AD 输入；（6）32 位 GPIO。

- **高端人机接口配置：**EM9287 可支持高分辨率彩色 LCD 显示（1024×768），同时支持触摸屏，使 EM9287 同样可作为智能终端设备的选择。

- **紧凑的外形尺寸：**EM9287 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。

- **开发门槛低：**作为工控主板产品，EM9287 将预装操作系统（CE6.0 或 Linux-3.9.7）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 EM9287，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 EM9287 的应用开发门槛是很低的，可满足各种原因需求，各种的开发团队使用。

本手册详细介绍了 EM9287 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《EM9287 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 EM9287 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 iMX287
- 128MB DDR2 系统内存，用户可用空间约 100MB
- 128MB FLASH 存储器，其中用户文件空间 75MB
- 2 路 USB 主口，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

**网络与 CAN 接口**

- 2 路以太网接口，10M/100M 自适应
- 2 路 CAN 总线接口，与 GPIO 复用管脚。

**串口通讯配置**

- 总共 7 路用户可用串口，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 | 串口类型 | 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，最高波特率 19200，作 RS485 通讯为宜 。与 GPIO 复用管脚。 |
| COM8 | ttyS7 | 低速串口 | |
| - | ttyAM0 | 调试串口 | 固定参数：115200bps、8-N-1 |

**其他通讯接口**

- 2 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
- 4 路 PWM 输出，每路输出频率、占空比均可独立设置。

**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 均可配置（使用 CN3）
- 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，输入输出独立可控。
- 部分 GPIO 与系统的其他通讯功能复用管脚。
- 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%，最小工作电流 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。

| 电流 | 网络连接 | CPU 负载 | 其它 |
|---|---|---|---|
| 170mA | - | 1% | - |
| 200mA | +eth0 | 1% | - |
| 230mA | +eth0, +eth1 | 1% | - |

| | | | |
|---|---|---|---|
| 340mA | +eth0, +eth1 | 98% | - |

EM9287 工作电流参数表

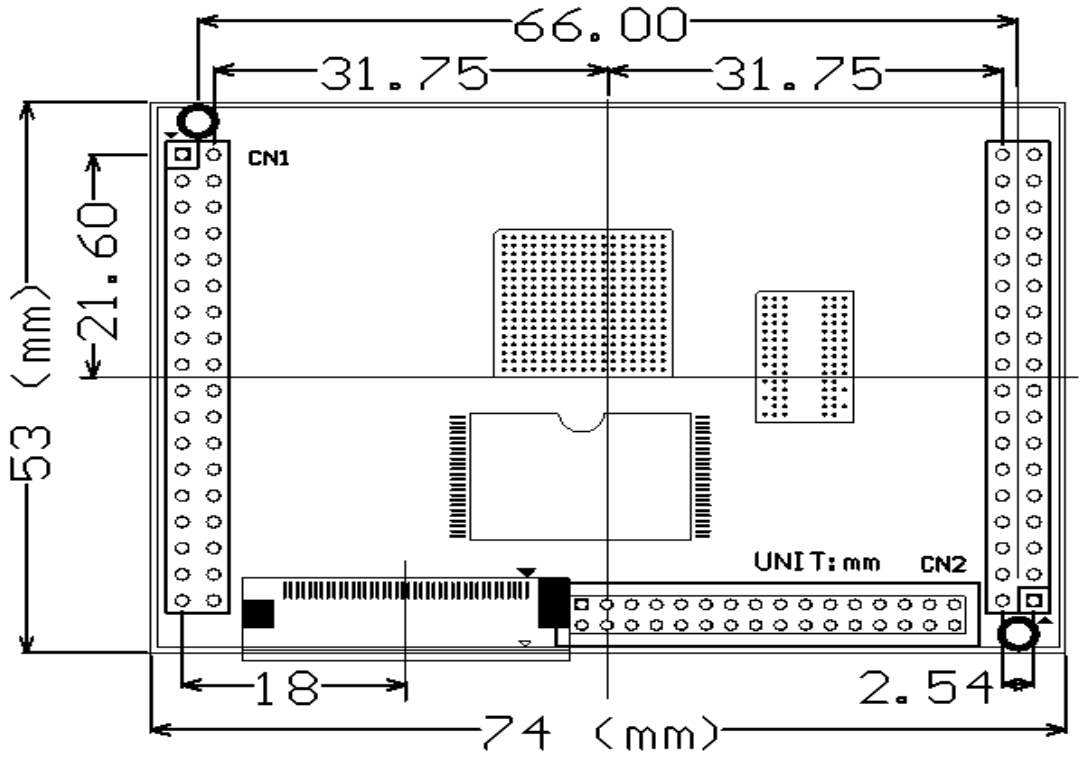
**CE 平台基本软件环境**

- 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-3.9.7 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

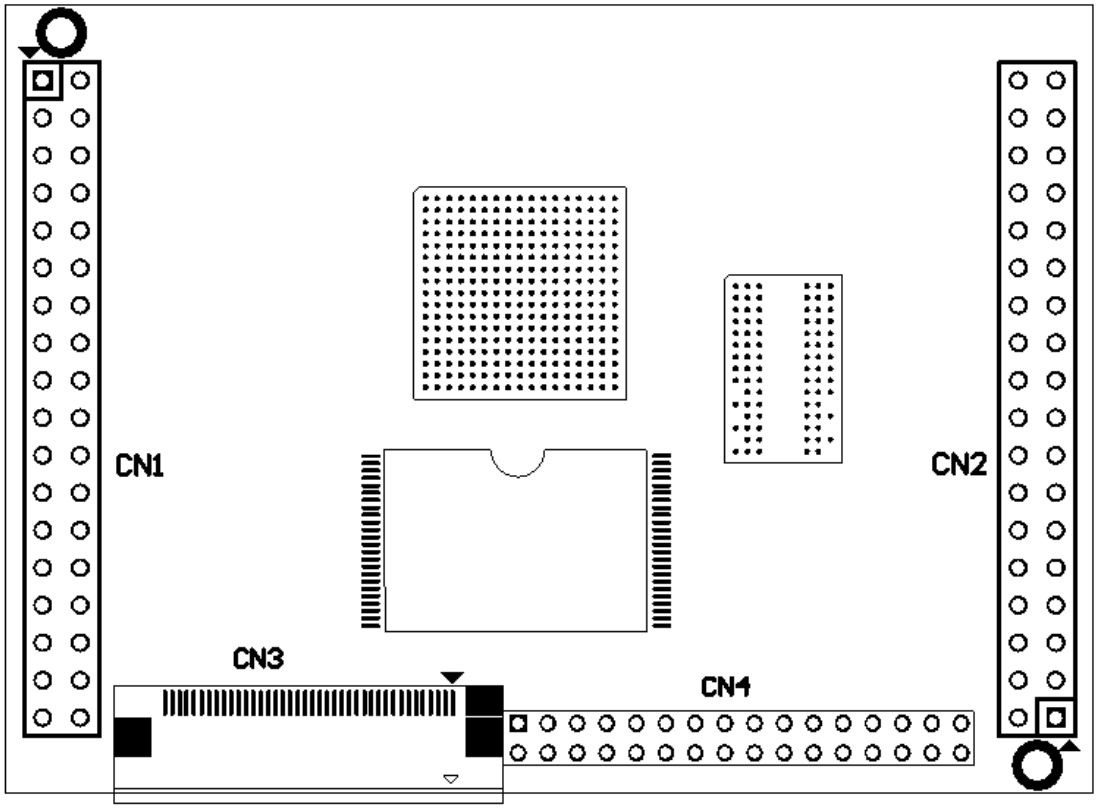



EM9287 外形尺寸示意图（2.54mm = 1”）

# 3、模块信号管脚功能描述

EM9287 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 EM9287 板卡的固定以及与应用主板的信号连接两个功能。EM9287 共有 4 组信号插针，分别编号为 CN1、CN2、CN3 和 CN4，其中的 CN1 和 CN2 分别位于 EM9287 板卡模块的两端，为 2 组标准 2.54mm(0.1 英寸)间距 IDC36 针双列直插管脚，EM9287 正是通过 CN1 和 CN2 与应用底板连接在一起的；EM9287 的 CN3 为 40 芯 FPC0.5 扁平线连接座，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。CN4 为扩展并行总线引出信号，在 EM9287 中预留为 32 芯 2.0mm 双排孔，在使用时，插接 2.0 双排插针即可引出。CN3 与 CN4 是复用相同的硬资源，因此在使用时，CN3 与 CN4 只能选其中一个，EM9287 默认为 CN3，仅在 Linux 系统中支持 CN4 接口的应用，使用 CN4 时，需将 CN3 去掉。

EM9287 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。




EM9287 的 CN1 – CN4 所在位置示意图

以下对 EM9287 所有管脚信号列表逐一说明。

### 3.1 EM9287 的 CN1 信号定义

EM9287 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。

CN1 各管脚的定义如下：

| 信号名称及简要描述 | CN1 | | 信号名称及简要描述 |
|---|---|---|---|
| | PIN | PIN | |
| LINK1n，Eth0 连接/传送指示 | 1 | 2 | SPEED1n，Eth0 速度指示 |
| TPTX1+，以太网差分输出 | 3 | 4 | TPTX1-，以太网差分输出 |
| TPRX1+，以太网差分输入 | 5 | 6 | TPRX1-，以太网差分输入 |
| VDD_CMT1，网络变压器公共端 | 7 | 8 | VDD_CMT2，网络变压器公共端 |
| TPTX2+，以太网差分输出 | 9 | 10 | TPTX2-，以太网差分输出 |
| TPRX2+，以太网差分输入 | 11 | 12 | TPRX2-，以太网差分输入 |
| LINK2n，Eth1 连接/传送指示 | 13 | 14 | SPEED2n，Eth1 速度指示 |
| 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 |
| ttyS1_RXD（COM2） | 19 | 20 | ttyS1_TXD（COM2） |
| ttyS2_RXD（COM3），232 电平 | 21 | 22 | ttyS2_TXD（COM3），232 电平 |
| ttyS3_RXD（COM4） | 23 | 24 | ttyS3_TXD（COM4） |
| ttyS4_RXD（COM5） | 25 | 26 | ttyS4_TXD（COM5）/ DBGSL# |
| GPIO0 / ttyS1_CTS#（COM2） | 27 | 28 | GPIO1 / ttyS1_RTS#（COM2） |
| GPIO2 / ttyS3_CTS#（COM4） | 29 | 30 | GPIO3 / ttyS3_RTS#（COM4） |
| GPIO4 / ttyS4_CTS#（COM5） | 31 | 32 | GPIO5 / ttyS4_RTS#（COM5） |
| GPIO6 / PWM1 | 33 | 34 | GPIO7 / PWM2 |
| GPIO8 / CAN1_RXD | 35 | 36 | GPIO9 / CAN1_TXD |

CN1 各个管脚信号简要说明如下：

| PIN# | 信号名称 | 方向 | 描述 |
|---|---|---|---|
| 1 | LINK1n | O | 以太网 Eth0 连接 / 传送指示，低电平有效。 |
| 2 | SPEED1n | O | 以太网 Eth0 速度指示，低电平有效。 |
| 3 | TPTX1+ | O | 以太网 Eth0 差分输出信号 |
| 4 | TPTX1- | O | 以太网 Eth0 差分输出信号 |

| | | | |
|---|---|---|---|
| 5 | TPRX1+ | I | 以太网 Eth0 差分输入信号 |
| 6 | TPRX1- | I | 以太网 Eth0 差分输入信号 |
| 7 | VDD_MCT1 | O | 以太网 Eth0 的网络变压器信号公共端 |
| 8 | VDD_MCT2 | O | 以太网 Eth1 的网络变压器信号公共端 |
| 9 | TPTX2+ | O | 以太网 Eth1 差分输出信号 |
| 10 | TPTX2- | O | 以太网 Eth1 差分输出信号 |
| 11 | TPRX2+ | I | 以太网 Eth1 差分输入信号 |
| 12 | TPRX2- | I | 以太网 Eth1 差分输入信号 |
| 13 | LINK2n | O | 以太网 Eth1 连接 / 传送指示，低电平有效。 |
| 14 | SPEED2n | O | 以太网 Eth1 速度指示，低电平有效。 |
| 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），系统运行状态选择信号 DBGSL# |
| 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，与 COM4 口的 CTS#复用管脚。 |
| 30 | GPIO3 | I / O | 通用数字 IO，与 COM4 口的 RTS#复用管脚。 |
| 31 | GPIO4 | I / O | 通用数字 IO，与 COM5 口的 CTS#复用管脚。 |
| 32 | GPIO5 | I / O | 通用数字 IO，与 COM5 口的 RTS#复用管脚。 |

| | | | |
|---|---|---|---|
| 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，与 CAN1_RXD 复用管脚。 |
| 36 | GPIO9 | I / O | 通用数字 IO，与 CAN1_TXD 复用管脚。 |

**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。

**EM9287 的异步串口，**在 CE 系统中的编号从 COM2 开始，7 路串口分别为 COM2 – COM8，其中 COM6 – COM8 配置在 CN2，与 GPIO10 - GPIO15 复用管脚，COM7 配置在 CN2，与 GPIO12 - GPIO13 复用管脚，COM8 配置在 CN2，与 GPIO14 - GPIO15 复用管脚，并且同时复用 CAN2 总线。其他 4 路串口均在 CN1，且采用专用管脚引出，以突出 EM9287 串口的可用性，同时也提高了 EM9287 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 19200bps，仅支持 8-bit 数据位，作 RS485 通讯最佳。 |
| COM8 | ttyS7 | 低速串口 | |

**EM9287 的串口分成两类，**其中 COM2 – COM6 为高速串口，其波特率可达 3Mbps；COM7 – COM8 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 – COM8 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。

EM9287 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。

在缺省状态下，系统启动后 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。

**EM9287 的运行状态设置：**

EM9287 的 BGSL#设置信号与 COM5_TX 信号复用，在 EM9287 上电启动时，COM5_TX 输出被禁止，系统会读取此时该管脚的电平状态，以配置系统的运行模式。

启动时，DBGSL#通过 5.1K 电阻接到地（状态为低），这时 EM9287 将进入调试状态：系统启动后，会自动复制 USB 盘中的 userinfo.txt 配置文件到 EM9287 的 NandFlash 下；DBGSL#悬空（状态为高），EM9287 将进入运行状态，若此时文件 userinfo.txt 包含客户应用程序的有效信息，该应用程序将被系统启动。需要注意，DBGSL#不能直接对地短接，必须经过 5.1K 电阻后，才能与地相接，否则会导致 COM5 功能异常，甚至造成硬件损坏。

### 3.2 EM9287 的 CN2 信号定义

EM9287 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 EM9287 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# |
| 电源地（GND） | 5 | 6 | 电源地（GND） |
| USB_OTG_D+ | 7 | 8 | USB_OTG_D- |
| USB_OTG_UID | 9 | 10 | BATT3V |
| DBG_COM_RX | 11 | 12 | DBG_COM_TX |
| USB2_HD+，USB2 Host | 13 | 14 | USB2_HD-，USB2 Host |
| GPIO10 / ttyS5_RXD（COM6） | 15 | 16 | GPIO11 / ttyS5_TXD（COM6） |
| GPIO12 / ttyS6_RXD（COM7） | 17 | 18 | GPIO13 / ttyS6_TXD（COM7） |
| GPIO14 / ttyS7_RXD（COM8） | 19 | 20 | GPIO15 / ttyS7_TXD（COM8） |
| 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 |

CN2 各个管脚信号简要说明如下：

| PIN# | 信号名称 | 方向 | 描述 |
|---|---|---|---|
| 1 | +5V 电源输入 | P | 给 EM9287 供电的电源脚，要求为 5V/5% |
| 2 | +5V 电源输入 | P | |
| 3 | USB_OTG_VBUS | I/O | USB_OTG 的电源，作为 HOST 时，向外提供 5V 电源 |
| 4 | RSTIN# | I | EM9287 复位输入信号，低电平复位 |
| 5 | GND | P | EM9287 电源及信号参考地 |
| 6 | GND | P | |
| 7 | USB_OTG_D+ | I/O | USB_OTG 差分信号+ |

| | | | |
|---|---|---|---|
| 8 | USB_OTG_D- | I/O | USB_OTG 差分信号- |
| 9 | USB_OTG_UID | I | USB_OTG 工作模式输入，高：作为设备口；低：作为主控口。 |
| 10 | BATT3V | P | EM9287 实时钟 RTC 的后备电池输入,3.0V |
| 11 | DBG_COM_RX | I | 调试串口输入，RS232 电平 |
| 12 | DBG_COM_TX | O | 调试串口输出，RS232 电平 |
| 13 | USB2_HD+ | I/O | USB2 口差分信号+ |
| 14 | USB2_HD- | I/O | USB2 口差分信号 |
| 15 | GPIO10 | I/O | 通用 I/O，复用串口 COM6_RXD 信号 |
| 16 | GPIO11 | I/O | 通用 I/O，复用串口 COM6_TXD 信号 |
| 17 | GPIO12 | I/O | 通用 I/O，复用串口 COM7_RXD 信号 |
| 18 | GPIO13 | I/O | 通用 I/O，复用串口 COM6_TXD 信号 |
| 19 | GPIO14 | I/O | 通用 I/O，复用串口 COM8_RXD 信号 |
| 20 | GPIO15 | I/O | 通用 I/O，复用串口 COM8_TXD 信号 |
| 21 | GPIO16 | I/O | 通用 I/O |
| 22 | GPIO17 | I/O | 通用 I/O |
| 23 | GPIO18 | I/O | 通用 I/O |
| 24 | GPIO19 | I/O | 通用 I/O |
| 25 | GPIO20 / PWM3 | I/O | 通用 I/O，复用 PWM3 输出通道 |
| 26 | GPIO21 / PWM4 | I/O | 通用 I/O，复用 PWM4 输出通道 |
| 27 | GPIO22 / I2C_SDA | I/O | 通用 I/O，复用 I2C 总线 SDA 数据信号 |
| 28 | GPIO23 / I2C_SCL | I/O | 通用 I/O，复用 I2C 总线 SCL 时钟信号 |
| 29 | GPIO24 / IRQ1 | I/O | 通用 I/O，复用外部中断源 1 |
| 30 | GPIO25 / IRQ2 | I/O | 通用 I/O，复用外部中断源 2 |
| 31 | GPIO26 / IRQ3 | I/O | 通用 I/O，复用外部中断源 3 |
| 32 | GPIO27 / IRQ4 | I/O | 通用 I/O，复用外部中断源 4 |
| 33 | GPIO28 / SPI_MISO | I/O | 通用 I/O，复用 SPI 总线 MISO 信号 |
| 34 | GPIO29 / SPI_MOSI | I/O | 通用 I/O，复用 SPI 总线 MOSI 信号 |
| 35 | GPIO30 / SPI_SCLK | I/O | 通用 I/O，复用 SPI 总线 SCK 时钟信号 |

| | | | |
|---|---|---|---|
| 36 | GPIO31 / SPI_CS0N | I/O | 通用 I/O，复用 SPI 总线 CS 片选使能信号 |

**关于 CN2 中相关信号的进一步说明：**

**RSTIN#为对板卡的复位输入，**不用时，可悬空。低电平输入对板卡硬件复位，RSTIN#变高后 50ms – 100ms 系统方开始启动，以保证供电电压已稳定。

**USB OTG 端口，**EM9287 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 EM9287 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 EM9287 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 EM9287 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 EM9287 作为设备端时，外部 USB 主控将通过 USB_OTG_VBUS 输入 5V 电源，但 EM9287 并不使用这个电源。

**调试串口 DBG_COM，**在 Linux 平台也成为控制台终端 console（设备名称 ttyAM0）。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。

**串口 ttyS5 – ttyS7，**该 3 路串口与 GPIO 复用管脚。同时 GPIO14 和 GPIO15 还分别与 CAN2_RXD 和 CAN2_TXD 复用管脚。在实际应用中只能打开 ttyS7 或 CAN2，禁止同时打开，否则出现错误状态。

### 3.3 EM9287 的 CN3 信号定义

EM9287 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 | LCD_PWM | 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。
- EM9287 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”；EM9287 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。

### 3.4 EM9287 的 CN4 信号定义

CN4 采用 32 芯（IDC32）2mm 间距的双排针焊接孔，主要是用于引出并行扩展接口。使用时，需要焊接 32 芯、2.0mm 双排插针。并行扩展接口是针对需要扩展专用电路单元的客户考虑的，它与 LCD 接口复用管脚，即若使用并行扩展接口，就不能使用 LCD 显示接口。EM9287 的并行扩展接口是可选功能，在标准的 EM9287 产品中，并不焊接 CN4。

CN4 的信号定义如下：

| 信号名称及简要描述 | CN4 | | 信号名称及简要描述 |
|---|---|---|---|
| | PIN | PIN | |
| NC | 1 | 2 | ISA_CS#，ISA 总线片选信号 |
| ISA_RD#，ISA 总线读信号 | 3 | 4 | ISA_WE#，ISA 总线写信号 |
| ISA_A0，ISA 地址总线 | 5 | 6 | ISA_A1，ISA 地址总线 |
| ISA_A2，ISA 地址总线 | 7 | 8 | ISA_A3，ISA 地址总线 |
| ISA_A4，ISA 地址总线 | 9 | 10 | ISA_A5，ISA 地址总线 |
| ISA_A6，ISA 地址总线 | 11 | 12 | ISA_A7，ISA 地址总线 |
| ISA_D0，ISA 数据总线 | 13 | 14 | ISA_D1，ISA 数据总线 |
| ISA_D2，ISA 数据总线 | 15 | 16 | ISA_D3，ISA 数据总线 |
| ISA_D4，ISA 数据总线 | 17 | 18 | ISA_D5，ISA 数据总线 |
| ISA_D6，ISA 数据总线 | 19 | 20 | ISA_D7，ISA 数据总线 |
| ISA_D8，ISA 数据总线 | 21 | 22 | ISA_D9，ISA 数据总线 |
| ISA_D10，ISA 数据总线 | 23 | 24 | ISA_D11，ISA 数据总线 |
| ISA_D12，ISA 数据总线 | 25 | 26 | ISA_D13，ISA 数据总线 |
| ISA_D14，ISA 数据总线 | 27 | 28 | ISA_D15，ISA 数据总线 |
| GND | 29 | 30 | GND |
| 3.3V 电源输出 | 31 | 32 | 3.3V 电源输出 |

EM9287 的扩展总线只实现单纯的读写功能，而复位输出，中断输入等功能，都可利用 EM9287 已有的功能。扩展总线中，ISA_A0 – ISA_A7 为地址总线，ISA_D0 – ISA_D15 为数据总线。每个地址单元代表一个字（WORD = 16-bit），共可访问 256 个地址单元；当使用 8-bit 数据宽度时，实际仍然是 16-bit 数据总线，由应用层忽略高位字节。

**关于 CN3 与 CN4 的特别说明：**

CN3 是 TFT 彩色 LCD 接口，CN4 是并行总线接口，也成为 ISA 扩展总线。

仅在 Linux 系统中，才支持 ISA 总线资源。在系统中，CN3 与 CN4 使用的是同一硬件

资源，实现不同应用的功能。因此在使用时，选用 LCD 接口，则在 EM9287 上焊接 CN3，而 CN4 保留为空。在 Linux 系统中且需要使用 ISA 总线时，在 EM9287 上焊接 CN4，CN3 保留为空。默认情况下，系统支持为 LCD 接口，即焊接 CN3。

# 4、基本电气特性与注意事项

在客户的应用设计中，EM9287 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 EM9287 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 EM9287 的额定参数

| 参数名称 | 最小值 | 典型值 | 最大值 | 简要说明 |
|---|---|---|---|---|
| +5V 直流瞬态输入 | -0.3V | +5.0V | +7.0V | 最大电压持续时间小于 30ms。 |
| 工作电流 | 160mA | 170mA | 300mA | 连接上两个网络时测试最大电流 |
| GPIO 管脚输入电压 | -0.3V | +3.3V | +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

EM9287 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 |
|---|---|---|---|
| $\mathbf{V_{IL}}$ | 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

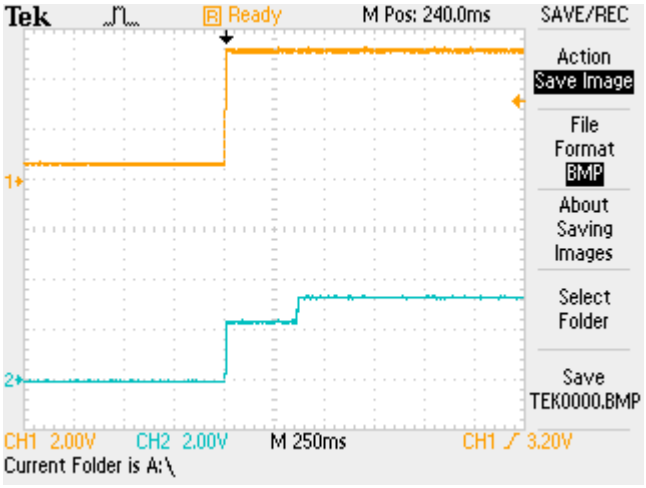
| | 典型值 | 简要说明 |
|---|---|---|
| 差分输出电压 | 2.0V | 100BASE-TX 模式 |
| 差分输出电流 | 26mA | 100BASE-TX 模式 |
| 差分输出电压 | 2.5V | 100BASE-T 模式 |
| VDD_MCT | 3.3V | 共模偏置电压，100Ω 终端电阻 |

### 4.5 3.3V TTL 信号的基本参数

EM9287 共引出 32 位通用数字 IO（也称为 GPIO），均为 3.3V TTL 电平。此外，EM9287 的 COM2、COM4、COM5 的 RXD 和 TXD 也为 3.3VTTL 电平信号，其 DC 电气特性与 EM9287 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 |

### 4.6 GPIO 上电时序




EM9281 GPIO 上电时序

（橙色线为板卡供电电源 DC5V，绿色线为 GPIO0-GPIO31 上电时序）

系统上电后，电源监测功能开始延时等待，并输出低电平给系统复位，GPIO 引脚嵌位在 2.5V 左右。上电后约 300ms，电源检测完毕，输出高电平，系统正常启动。

在使用 GPIO 作为继电器或其它相关关键执行机构的控制信号时，建议在所使用的 GPIO 信号线上串入驱动器，且驱动器的输入高电平最低电压应低于 2.5V，以防止上电时，出现误动作，影响设备安全。

如：74LVC245，工作电源电压在 3.3V 时，输入高电平的最低电压为 2V。

| Symbol | Parameter | Conditions | –40 °C to +85 °C | | | –40 °C to +125 °C | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | | Min | Typ[1] | Max | Min | Max | |
| $V_{IH}$ | HIGH-level input voltage | $V_{CC}$ = 1.2 V | 1.08 | - | - | 1.08 | - | V |
| | | $V_{CC}$ = 1.65 V to 1.95 V | 0.65 × $V_{CC}$ | - | - | 0.65 × $V_{CC}$ | - | V |
| | | $V_{CC}$ = 2.3 V to 2.7 V | 1.7 | - | - | 1.7 | - | V |
| | | $V_{CC}$ = 2.7 V to 3.6 V | 2.0 | - | - | 2.0 | - | V |

### 4.7 设计注意事项

1. EM9287 的核心 CPU 芯片 iMX287 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 EM9287 获得很高的性能价格比。对接入 EM9287 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 EM9287 上，可能造成核心芯片电源单元的损坏。
2. EM9287 上 CN1 – CN3 的大部分 LVTTL 信号均直接来自于系统的核心 CPU 芯片 iMX287，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. EM9287 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 EM9287 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 EM9287 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 EM9287 的 GPIO 管脚。
6. EM9287 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 EM9287 的 USB 数据收发单元，因此强烈推荐客户的应用底板参考 EM9287 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、EM9287 的相关功能的说明

本节中，主要对 EM9287 的各个功能模块进行简要的说明，涉及到的例程代码均均来自于 CE 系统的范例程序。关于相应的 Linux 程序范例，请参考《Linux 工控主板应用程序编程手册（EM9280 V2.0）》。

## 5.1 CAN 接口

EM9287 有两路 FLEXCAN，支持 CAN2.0b，支持 1Mbps、512Kbps、256Kbps 等速率。CAN 总线与 GPIO 复用管脚，在系统上电启动后，默认是 GPIO 状态，当调用驱动程序打开 CAN 设备后，相应的 GPIO 引脚资源将自动分配为 CAN 引脚。CAN 的基本操作如下：

```
class EM9170_CAN    m_CAN;
CAN_FILTER          Filter;
CAN_PACKET          Pkt;        //设置 FILTER 参数
BOOL                bResult;

//memset( &Filter, sizeof(CAN_FILTER), 0 );
Filter.dwGroup = 0;
Filter.dwType = CAN_PACKET_TYPE_STANDARD;   //标准帧
Filter.dwID = 0x03;
Filter.dwMask = 0;                          //均不检查，全部接收
Filter.dwRTR = 0;

bResult = m_CAN.OpenCAN( 1, 250000, &Filter ); //打开 CAN 设备
if( bResult==FALSE )
    return -1;

//设置 CAN Packet
memset( &Pkt, sizeof(CAN_PACKET), 0 );
Pkt.dwType = CAN_PACKET_TYPE_STANDARD;
Pkt.dwID = 0x03;
Pkt.dwPrio = 0;
Pkt.dwDatLen = 8;
for( i1=0;   i1<(int)Pkt.dwDatLen;   i1++ ) //填充 CAN 要发送的数据
{
    Pkt.ucDat[i1] = 0x31 + i1;
}
```

打开并初始化 CAN 以后，就可以调用 m_CAN.WriteCAN 和 m_CAN.ReadCAN 进行 CAN 通讯了。

## 5.2 WDT 看门狗定时器

EM9287 直接使用了 iMX287 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。

EM9287 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);
```

### 5.3 USB 接口

EM9287 可提供 2 个 USB 端口：一个高速主控接口，和一个 USB OTG 接口。EM9287 的 USB 主控接口可直接与标准 U 盘相连，EM9287 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。EM9287 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 EM9287 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 EM9287 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 EM9287 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。

### 5.4 UART 异步串口

EM9287 物理上有 7 个串口， 7 个物理串口分别对应的逻辑编号为 COM2 – COM8。此外 EM9287 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。

EM9287 的 7 个串口按最高波特率分为高速串口 COM2 – COM6 和低速串口 COM7 – COM8。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。EM9287 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 EM9287 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
```

## 5.5 $I^2C$ 接口

EM9287 的 $I^2C$ 接口为 2 线制标准 $I^2C$ 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。EM9287 板上已固化了面向 $I^2C$ 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 $I^2C$ 数据传输了。

基本的 $I^2C$ 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 $I^2C$ 器件寄存器的起始地址。$I^2C$ 操作的主要代码如下：

打开 $I^2C$ 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 $I^2C$ 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 I²C 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 EM9287 的资料光盘中的 I2C 应用范例了解其详细的使用方法。

### 5.6 SPI 同步串口

EM9287 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。EM9287 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 EM9287 的资料光盘中的 SPI 应用范例了解其详细的使用方法。

### 5.7 PWM 脉冲输出

EM9287 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。EM9287 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。

## 5.8 IRQ 外部中断

EM9287 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
    //… …
}
```

### 5.9 AD 模拟通道

EM9287 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，EM9287 还可提供对输入的+5V 电源电压、EM9287 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  EM9287_DAQ_VOLTAGE_CH0          0
#define  EM9287_DAQ_VOLTAGE_CH1          1
#define  EM9287_DAQ_VDD_5V               2
#define  EM9287_DAQ_CPU_TEMPERATURE      6
#define  EM9287_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 = EM9287_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 = EM9287_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。EM9287 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。

## 5.10 GPIO 通用数字 IO

EM9287 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。EM9287 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。EM9287 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 | CTS4# | 与 COM4 口的 CTS#复用管脚。 |
| GPIO3 | RTS4# | 与 COM4 口的 RTS#复用管脚。 |
| GPIO4 | CTS5# | 与 COM5 口的 CTS#复用管脚。 |
| GPIO5 | RTS5# | 与 COM5 口的 RTS#复用管脚。 |
| GPIO6 | PWM1 | 与 PWM1 复用管脚。 |
| GPIO7 | PWM2 | 与 PWM2 复用管脚。 |
| GPIO8 | CAN1_RXD | 与 CAN1 的 RXD 复用管脚 |
| GPIO9 | CAN1_TXD | 与 CAN1 的 TXD 复用管脚 |
| GPIO10 | RXD6 | 与 COM6 口的 RXD 复用管脚。 |
| GPIO11 | TXD6 | 与 COM6 口的 TXD 复用管脚。 |
| GPIO12 | RXD7 | 与 COM7 口的 RXD 复用管脚。 |
| GPIO13 | TXD7 | 与 COM7 口的 TXD 复用管脚。 |
| GPIO14 | RXD8/CNA2_RXD | 与 COM8 及 CAN2 口的 RXD 复用管脚。 |
| GPIO15 | TXD8/CAN2_TXD | 与 COM8 及 CAN2 口的 TXD 复用管脚。 |
| 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。