USB2.0 接口可编程控制模块

USB20D 使用说明

西安达泰电子有限责任公司

+86-29-85272421，85277568，FAX:+86-29-85277554

西安市朱雀大街南段明德门凯旋广场 D323

E-MAIL：dataie@gmail.com　data029@126.com

本公司其他系列产品图片及详细资料，欢迎查看网站 http://www.dataie.com

本文档更新日期：2008-09-23　版本号：V3.7

该产品在不断改进功能，新增或修改功能的说明以最新版本为准。恕不另行通知。

首先，感谢您选用 USB20D 模块，我公司将一如既往地为您提供优质的产品和服务！

敬告用户，请您在使用前，仔细阅读本手册！本手册适用于 USB20D 模块，阅读时请您留意本模块的备注。

目　录

1 USB20D 概述及其设备安装

1.1 USB20D 概述

USB20D是由西安达泰电子有限责任公司设计的USB2.0 设备通用接口模块，它隐藏了通过USB总线进行数据传输所需的繁琐技术细节。应用程序通过调用本模块提供的函数，可以把相应的功能转变成模块硬件接口上的一系列脉冲和电平，发送到外围逻辑，进行指定的数据传输，从而极大地简化USB设备的设计工作。

本模块提供两种数据传输模式：地址 I/O 模式、DMA 批量数据传输模式。以设计一个 A/D 数据采集器为例，可以使用地址 I/O 模式执行初始化、设置采样参数、读取状态等功能，使用 DMA 批量数据传输模式读取采样得到的大批量数据。

本模块是一个 USB2.0 设备，同时也兼容 USB1.1 标准，但是会降低数据传输速度。

第 2 章详细介绍了硬件接口的总线和控制信号的功能。

第 3 章详细介绍了相关函数的功能和使用说明。

特点

标准 USB 接口，高性能 USB 接口器件，符合通用串行总线 USB2.0 版规范

高速 DMA 读写控制，读写速度大于 25Mbyte/Sec

系统驱动文件，DLL 动态连接库，用户不必编写任何驱动程序

SMT 工艺，低功耗系统，超小体积模块化设计，无需外接电源

简化的标准外部总线： 16 位或 8 位数据总线可设置，双向；高 8 位复用；
5 位地址总线，单向输出；9 根读写控制线。

提供一个 48MHz 或 30MHz 的时钟输出

提供 5V 电源输出。

应用

USB20D 可以作为用户系统的嵌入式模块使用，用户无需深入了解 USB 的协议及底层控制方法，就像操作 PC 总线一样，通过 USB 接口实现对用户系统的控制。

对于笔记本电脑来说，使用 USB 接口的意义更加重大，通用的 USB 接口不仅使笔记本电脑对外的连接变得方便，更可以使笔记本电脑生产厂商不再需要为不同配件在主板上安置不同的接口，这使主板的线路、组件的数量以及复杂程度都有不同程度的削减，从而使系统运行中的散热问题得到了改善。也将促进更高主频的处理器可以迅速应用在移动计算机中，使笔记本电脑与桌面 PC 的差距进一步缩小。

USB20D 控制模块发挥了 USB2.0 高速数据传输的特点，尤其适合于高速数据采集及图像数据传输设备，用于医疗、地震、振动、监控、虚拟仪器、科研实验室、工业生产现场领域的数据采集设备，特别是为便携式笔记本电脑和日益流行的掌上电脑数据采集提供了极为广阔的发展空间。

1.2 USB20D 设备的安装

特别提示：在首次使用 USB20D 模块之前，首先安装由生产厂家（西安达泰电子有限责任公司）提供的设备驱动安装程序！(程序在 USB20D_Setup 目录下)

1.2.1 安装驱动程序

打开计算机，进入 Windows98/2000/XP 系统，待启动完成后，建议将达泰资料光盘 USB20D_Setup 目录拷贝到您的硬盘，运行由厂商提供的 USB20D_Setup 程序。该程序可以自动识别操作系统。**注意：此时不要连接 USB20D 模块！**

USB20D_Setup.exe 将弹出以下界面：

点击[安装 USB20D]按键，按照提示安装即可。

1.2.2 安装 USB20D 模块

将 USB 四芯电缆扁平的一端插入计算机后面的任意一个 USB 端口,将另一端插入 USB20D

的 USB_T 插座上，硬件连接即完成。此时电脑提示发现新的 USB 设备，弹出上图界面：

选择从列表或指定位置安装（高级），点击[下一步]。

电脑提示以下信息：

选择[在搜索中包括这个位置]，驱动程序位于 C:\Program Files\USB20D_Driver 目录下。点击[下一步]，电脑提示以下信息：

点击[仍然继续]，按照提示即可完成 USB20D Loader Driver 安装。

然后安装 USB20D 接口驱动程序。此时会弹出以下界面：

选择[从列表或指定位置安装（高级）]，点击[下一步]，电脑提示以下信息：

选择[在搜索中包括这个位置]，驱动程序位于 C:\Program Files\USB20D_Driver 目录下。点击[下一步]，电脑提示以下信息：

点击[仍然继续]，按照提示即可完成 USB20D General Interface Module 安装。

安装成功后，USB20D 模块上的绿色指示灯点亮。此时建议重新启动电脑！

1.2.3 安装结果验证

打开从 Windows“开始”菜单中单击“设置”进入“控制面板”窗口，双击“系统”图标，弹出“系统 属性”对话框，在对话框中单击“设备管理器”标签，然后在“计算机”树形列表中双击“通用串行总线控制器”，检查此项目中是否有“USB20D:USB2.0 General Interface Module”等字样。若有，表示 USB 设备已成功安装，否则，说明您的安装过程出现了问题，请试着再安装，或向硬件供应商求助。

1.2.4 疑难问题解答

如果当您正确连接 USB 设备后，屏幕上没有任何反应，也没有出现“ USB20D:USB2.0 General Interface Module ”，有可能您的 USB 端口出现了问题。请进入《安装结果验证》中所述的“ 资源管理器 ”窗口中，检查树形列表中是否有“ 通用串行总线控制器 ”项目，若有，通常在这个项目中还应有两个子项“ USB Universal Host Controller ”、“ Usb Root Hub ”，以上项目如若缺一项，那意味着在您的系统中 USB 控制器存在问题，那么您还应试着安装 USB 总线驱动程序，它们都在 Windows 的安装盘上。

有些电脑主板的 USB2.0 接口需要驱动，否则系统工作于 USB1.1 模式下，请检查是否有上图中的子项：“ USB2.0 Enhanced Universal Host Controller ”

如果在安装过程中出现以下提示，这表明 USB20D 驱动安装不正常，这可能是安装过程中操作不对，您可以执行 USB20D_Setup.exe 选择**卸载 USB20D 驱动**，然后**重新启动电脑**，按照以上步骤重新安装。

2 硬件描述

本模块提供了 8 根或 16 根数据总线、5 根地址总线、3 根地址 IO 所需的控制信号、9 根批量传输所需的控制信号、以及其他的辅助控制信号。

2.1 辅助控制信号

本模块提供了 2 根辅助控制信号：DMAING、CLKOUT。

2.1.1 DMAING “正在批量数据传输”指示信号

本信号为数据传输模式指示信号，由本模块输出。
高电平：指示工作于批量数据传输模式。
低电平：指示工作于地址 IO 模式。

本信号由函数 USB20D_STARTDMA 设置为高电平，由函数 USB20D_ENDDMA 设置为低电平。

2.1.2 CLKOUT 模块内单片机时钟输出信号

本信号输出模块内单片机的时钟。
本信号由函数 USB20D_SETCPUCS 控制，可以设定时钟的频率、是否输出、是否翻转。

2.2 数据总线

本模块在 DMA 模式时，既可以通过函数设为 16 位的数据总线，也可以设为 8 位的数据总线；在 IO 模式时，只能为 8 位的数据总线。两种数据传输模式共用低 8 位数据总线。当工作于“地址 IO 模式”并且 nPWR 信号有效，或者工作于“批量传输模式”、nDMARD 信号有效并且 nDMACS、nDMAOE 有效时，数据总线处于输出状态，否则数据总线处于输入状态。

2.3 地址总线

本模块在 IO 模式时提供 5 位宽的地址总线，地址总线总是由本模块输出。

以下函数可以改变地址：

USB20D_SETADDRESS
USB20D_INPUT
USB20D_OUTPUT
USB20D_MULTINPUT
USB20D_MULTOUTPUT
USB20D_MIXEDIO

当使用后三个函数时，地址会根据函数的参数而改变，函数调用结束后，地址为函数参数指定的最后一个地址。

2.4 地址 I/O 模式所需的控制信号

本模块提供了 3 个地址 I/O 所需的控制信号：nPWR、nPRD、nPWAIT。

2.4.1 nPWR 地址写

本信号为地址写的写脉冲，输出一个低电平脉冲。本脉冲由函数

USB20D_OUTPUT

USB20D_MULTOUTPUT

产生。

当进行一次地址写时，本模块首先更新地址总线、把数据总线定义为输出并输出数据，然后使 PWR=0；接着判断 PWAIT 状态，等待 PWAIT=1（此时可以等待外部单片机等慢速逻辑执行）或者超时（超时时间由模块内单片机的工作频率决定，具体时间待测）；最后，使 PWR=1、把数据总线定义为输入。完成一次地址写。

2.4.2 nPRD 地址读

本信号为地址读的读脉冲，输出一个低电平脉冲。本脉冲由函数

USB20D_INPUT

USB20D_MULTINPUT

产生。

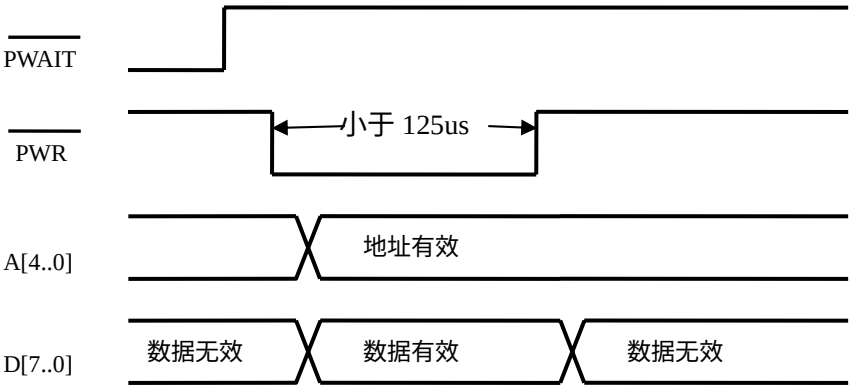
当进行一次地址读时，本模块首先更新地址总线、把数据总线定义为输入，使 PRD=0；接着判断 PWAIT 状态，等待 PWAIT=1 或者超时；然后从地址总线读取数据，并把此数据返回主机；最后，使 PRD=1。完成一次地址读。

2.4.3 nPWAIT 等待外围逻辑

本信号为一个输入信号，一般情况下，当外部逻辑为单片机等慢速逻辑时，需要使用本信号；如果外部逻辑为一个 CPLD 则可以悬空本信号管脚。

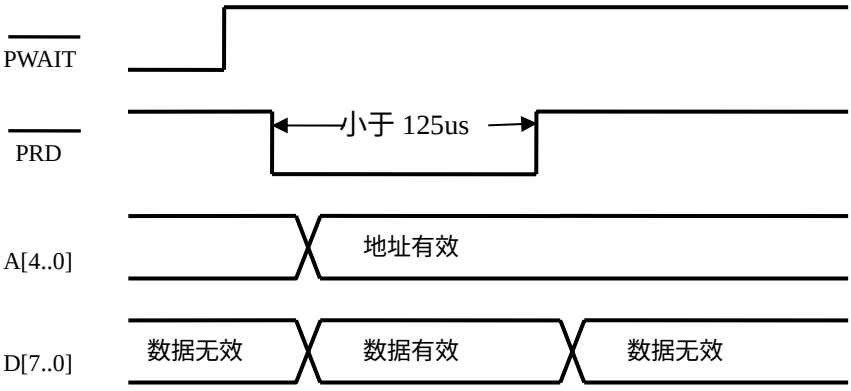
当外部逻辑为单片机等慢速逻辑时，先由外部逻辑把本信号拉低，则本模块在地址 IO 时会在 nPWR、nPRD 脉冲有效后会插入等待周期，等待外部逻辑执行完指定的读/写后，外部逻辑把本信号拉高，本模块检测到 nPWAIT=1 后，结束 nPWR、nPRD 脉冲，外部逻辑检测到 nPWR、nPRD 无效后，再次使 nPWAIT=0，准备好下一次读/写。

2.4.4 地址 I/O 模式的时序图



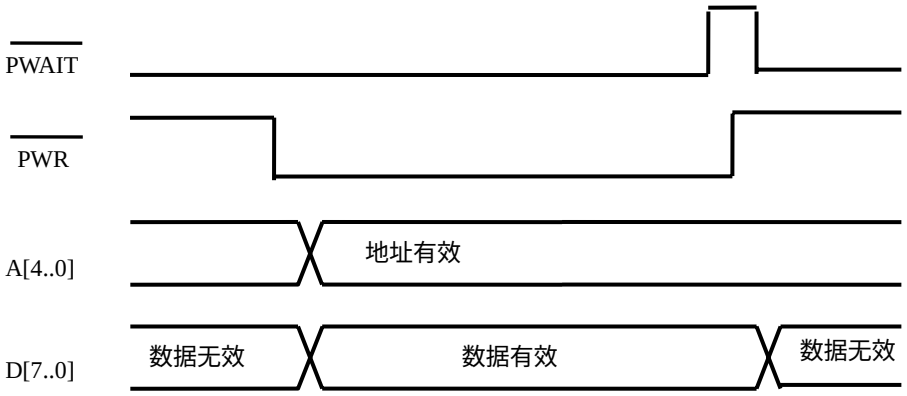
无等待周期的 I/O 写时序图

在 I/O 模式下，数据线为 8 位双向总线，地址线有锁存



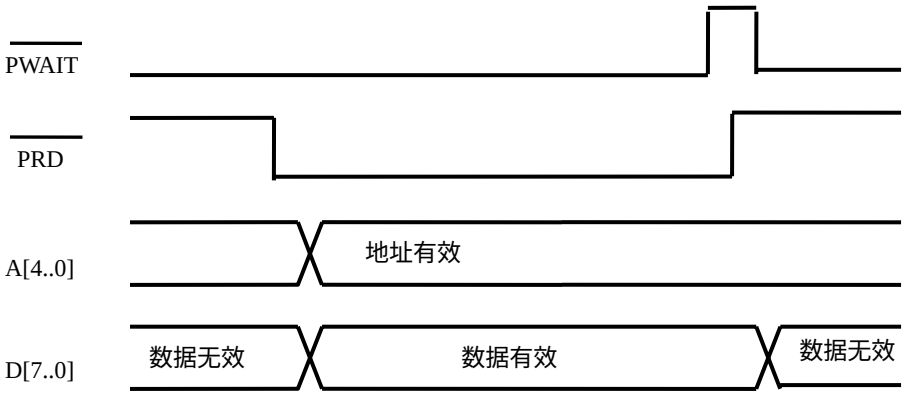
无等待周期的 I/O 读时序图

在 I/O 模式下，数据线为 8 位双向总线，地址线有锁存



有等待周期的 I/O 写时序图

在 I/O 模式下，数据线为 8 位双向总线，地址线有锁存



有等待周期的 I/O 读时序图

在 I/O 模式下，数据线为 8 位双向总线，地址线有锁存

2.5 DMA 批量数据传输所需的控制信号

本模块提供了 9 个批量数据传输所需要的信号。它们分别是：nDMACS、nDMARD、nDMAWR、DMADIR、nDMAOE、nPKTEND、nFIFOEMPTY、nFIFOFULL、IFCLK 。

2.5.1 nDMACS

模块选择信号，由外部逻辑提供，低电平有效。

当整个设备需要本模块与其他外部逻辑需要共享数据总线时，使 nDMACS=1 可以使本模块断开与外部总线的连接，本模块将忽略 nDMARD、nDMAWR、nDMAOE、nPKTEND 信号。

2.5.2 nDMARD

由外部逻辑提供，低电平脉冲有效。

当使用 USB20D_DMAWRITE 从主机向设备写数据时，数据首先从主机传送到本模块内的缓冲区内，外部逻辑使用 nDMARD 脉冲从本模块的缓冲区内读取主机发来的数据。

当 nDMAOE 有效时，读到的数据在 nDMARD 为低电平时从数据总线输出；当 nDMAOE 无效时，数据总线悬空，但是 nDMARD 脉冲依然有效，只不过数据不能输出。

2.5.3 nDMAWR

由外部逻辑提供，低电平脉冲有效。

当使用 USB20D_DMAREAD 主机从设备读数据时，外部逻辑首先使用 nDMAWR 脉冲把数据写入本模块的缓冲区内，然后数据从本模块内的缓冲区内传输到主机。

2.5.4 DMADIR

读/写控制信号，由本模块输出。

高电平：批量读数据，数据从设备传向主机。

低电平：批量写数据，数据从主机传向设备。

2.5.5 nDMAOE

由外部逻辑提供，低电平有效。

当 nDMAOE 有效时，外部逻辑从本模块内部缓冲区读数据时，在 nDMARD 为低电平时读到的数据从数据总线输出；当 nDMAOE 无效时，数据总线悬空，但是 nDMARD 脉冲依然有效，只不过数据不能输出。

2.5.6 nPKTEND

由外部逻辑提供，低电平脉冲有效。

在批量读数据时，外部逻辑使用 nDMAWR 把数据写入本模块内部缓冲区，每写满一个数据包后（如果本模块连接在 USB2.0 总线上则每个数据包为 512 字节，如果连接在 USB1.1 总线上则数据包为 64 字节），数据会自动传送到主机。如果需要传输一个不满的数据包（“ 短包 ”）比如需要传输 31 个字节，则外部逻辑应该在写 31 个字节数据后，产生一个 nPKTEND 脉冲。本模块接收到一个 nPKTEND 脉冲后，会把接收到 “ 短包 ” 发送回主机。

2.5.7 nFIFOEMPTY

本模块内部缓冲区 “ 空 ” 标志，由本模块输出，低电平有效。本信号在批量写数据时使用。

本模块内部提供了 2048 字节的批量写数据缓冲区，在批量写数据时，主机首先发送数

据到本模块的内部缓冲区，本模块接收到数据后，会使本信号无效，表示内部缓冲区已经有数据供外部逻辑读取。外部逻辑检测到本信号无效，开始从本模块缓冲区读取数据，数据全部读出后，本信号重新有效，此时外部逻辑应停止从本模块读取数据。

2.5.8 nFIFOFULL

本模块内部缓冲区“ 满 ”标志，由本模块输出，低电平有效。本信号在批量读数据时使用。

本模块内部提供了 2048 字节的批量读数据缓冲区，在批量读数据时，如果本模块的内部缓冲区不满，则模块使本信号无效，表示可以向本模块内部缓冲区写数据。外部逻辑检测到正在批量读数据而且本信号无效，开始向本模块写数据，每写满一个标准数据包（如果本模块连接在 USB2.0 总线上则每个数据包为 512 字节，如果连接在 USB1.1 总线上则数据包为 64 字节）后，如果主机正在使用 USB20D_DMAREAD 函数读取数据，则数据自动发送到主机。如果主机一直没有读取数据，则在外部逻辑写满 2048 个字节后，模块内部缓冲区满，本信号有效，此时外部逻辑应停止写数据。

2.5.9 IFCLK

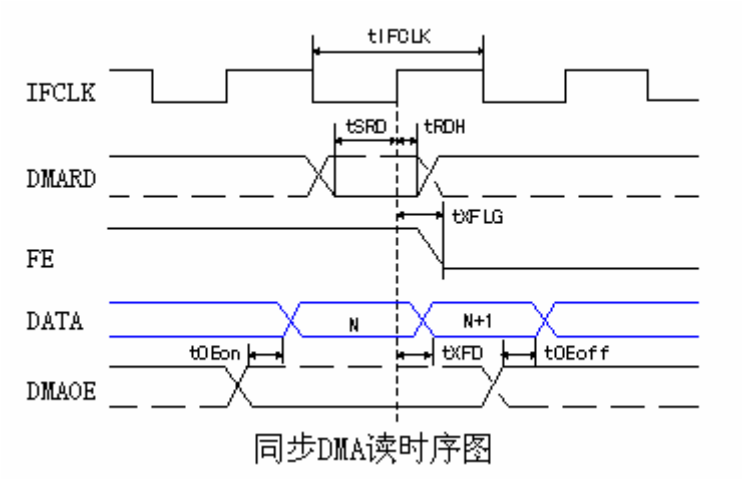
接口时钟信号，可以由本模块提供，此时该信号为输出，也可以由外部逻辑提供，此时信号为输入。

本信号主要在同步 DMA 时用作 DMA 读/写的同步时钟，也可以在异步 DMA 时作为一个高速时钟提供给外部逻辑。

本时钟信号如果由本模块提供，则可以选择时钟频率为 48MHz/30MHz；如果由外部逻辑提供，则有效的频率范围为 5MHz ~ 48MHz。

可以使用 USB20D_SETIFCFG 函数设置本信号，可以设置本时钟是否由本模块提供、模块提供的时钟频率（48MHz/30MHz）、时钟是否翻转、DMA 工作于同步/不同步方式选择。

2.5.10 DMA 时序图



同步DMA读时序图

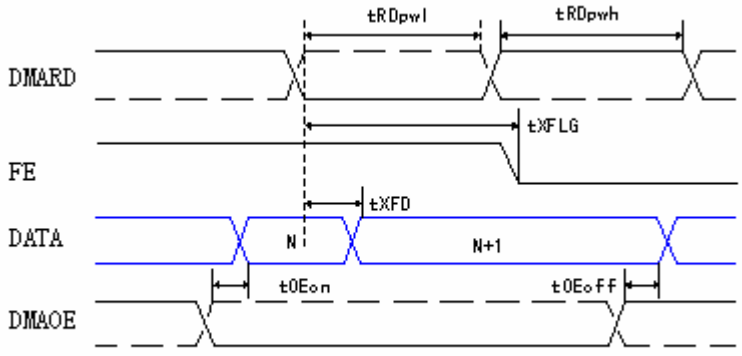
注：图中 FE 表示模块上的 FIFOEMPTY 信号线

使用模块内部同步时钟进行同步读时的参数表：

参数	说明	最小值	最大值	单位
t_{IFCLK}	IFCLK 周期	20.83		ns
t_{SRD}	DMARD 有效到同步时钟有效所需的建立时间	18.7		ns
t_{RDH}	DMARD 的保持时间	0		ns
t_{Oeon}	DMAOE 有效到数据输出有效的建立时间		10.5	ns
t_{Oeoff}	DMAOE 无效后数据的保持时间		10.5	ns
t_{XFLG}	同步时钟到“缓冲区空”标志有效的传播延迟		9.5	ns
t_{XFD}	同步时钟到下一数据有效所需的传播延迟		11	ns

使用外部同步时钟进行同步读时的参数表：

参数	说明	最小值	最大值	单位
t_{IFCLK}	IFCLK 周期	20.83	200	ns
t_{SRD}	DMARD 有效到同步时钟有效所需的建立时间	12.7		ns
t_{RDH}	DMARD 的保持时间	3.7		ns
t_{Oeon}	DMAOE 有效到数据输出有效的建立时间		10.5	ns
t_{Oeoff}	DMAOE 无效后数据的保持时间		10.5	ns
t_{XFLG}	同步时钟到“缓冲区空”标志有效的传播延迟		13.5	ns
t_{XFD}	同步时钟到下一数据有效所需的传播延迟		15	ns

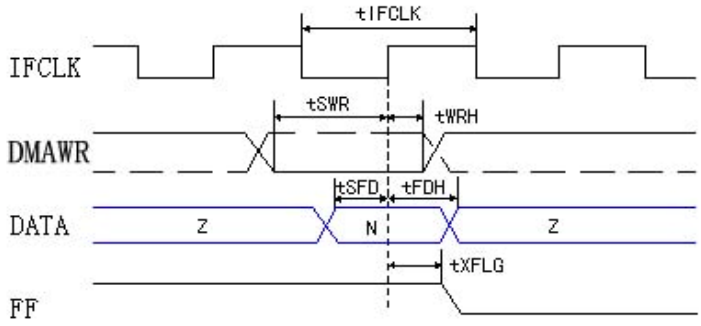


异步DMA读时序图

注：图中 FE 表示模块上 FIFOEMPTY 信号线

异步读时的参数表：

参数	说明	最小值	最大值	单位
t_{RDpwl}	DMARD 脉冲的有效低电平时间	50		ns
t_{RDpwh}	DMARD 脉冲的有效高电平时间	50		ns
t_{XFLG}	脉冲下降沿到“缓冲区空”标志有效的传播延迟		70	ns
t_{XFD}	脉冲下降沿到下一数据有效所需的传播延迟		15	ns
t_{Oeon}	DMAOE 有效到数据输出有效的建立时间		10.5	ns
t_{Oeoff}	DMAOE 无效后数据的保持时间		10.5	ns



同步DMA写时序图

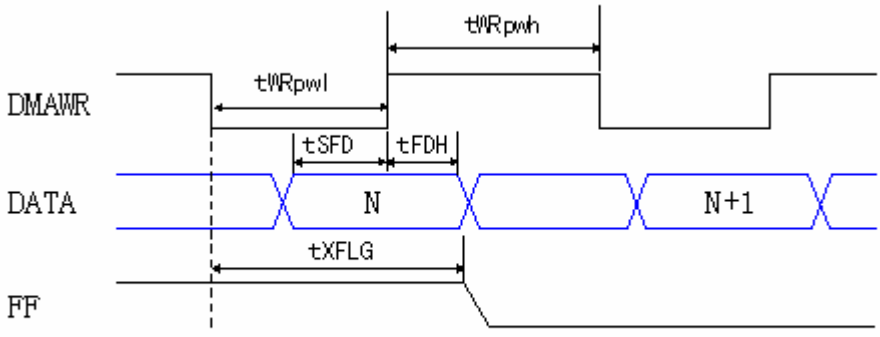
注：FF 表示 FIFOFULL 信号线

使用模块内部同步时钟进行同步写时的参数表：

参数	说明	最小值	最大值	单位
t_{IFCLK}	IFCLK 周期	20.83		ns
t_{SWR}	DMAWR 有效到同步时钟有效所需的建立时间	18.1		ns
t_{WRH}	DMAWR 的保持时间	0		ns
t_{SFD}	数据有效到同步时钟的建立时间	9.2		ns
t_{FDH}	同步时钟后，数据保持时间	0		ns
t_{XFLG}	同步时钟到“缓冲区空”标志有效的传播延迟		9.5	ns

使用外部同步时钟进行同步写时的参数表：

参数	说明	最小值	最大值	单位
t_{IFCLK}	IFCLK 周期	20.83	200	ns
t_{SWR}	DMAWR 有效到同步时钟有效所需的建立时间	12.1		ns
t_{WRH}	DMAWR 的保持时间	3.6		ns
t_{SFD}	数据有效到同步时钟的建立时间	3.2		ns
t_{FDH}	同步时钟后，数据保持时间	4.5		ns
t_{XFLG}	同步时钟到“缓冲区空”标志有效的传播延迟		13.5	ns



异步DMA写时序图

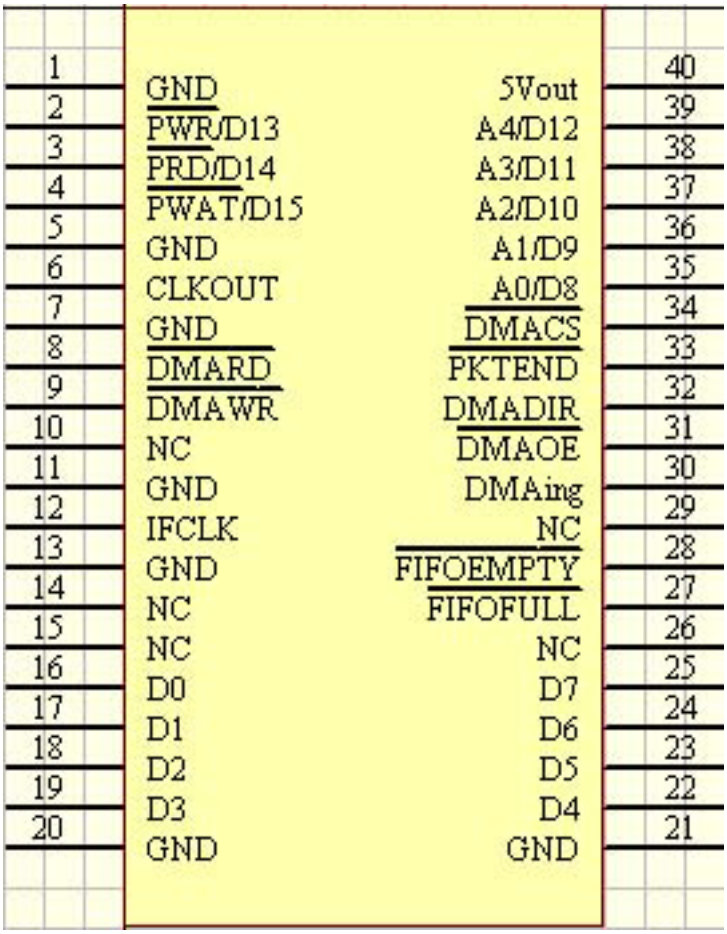
注：FF 表示模块上的 FIFOFULL 信号线

异步 DMA 写时的参数表：

参数	说明	最小值	最大值	单位
t_{WRpwl}	DMAWR 脉冲的有效低电平时间	50		ns
t_{WRpwh}	DMAWR 脉冲的有效高电平时间	70		ns
t_{SFD}	数据有效到脉冲上升沿的建立时间	10		
t_{FDH}	数据保持时间	10		
t_{XFLG}	脉冲下降沿到“缓冲区满”标志有效的传播延迟		70	ns

2.6 模块封装

2.6.1 模块的管脚定义



注意：

1．标为“NC”的管脚需要悬空，不要与用户板连接，否则有可能出现故障。

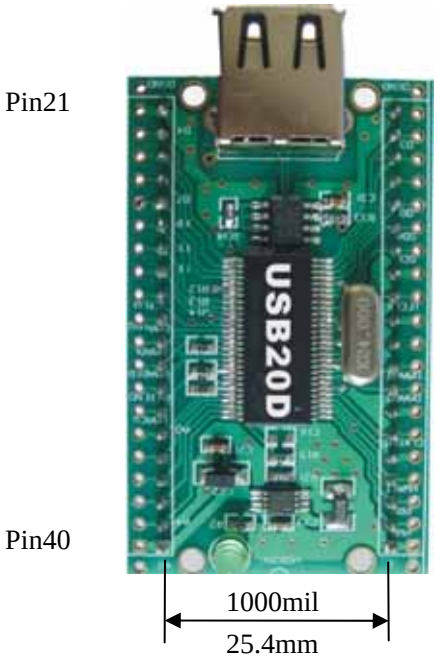
2．Pin40 的“5Vout”是本模块的输出电源，可用电流不要超过 200mA。

管脚说明：

名称	定义	方向	说明
D[7..0]	数据总线	双向	双向 DMA 数据总线，低 8 位。
D[15..8]	数据总线	双向	双向 DMA 数据总线，高 8 位。
DMADIR	传输方向	输出	由模块输出 DMA 控制器产生的方向控制信号。 1 – 输入：从设备到主机。 0 – 输出：从主机到设备。
nDMAWR	写数据	输入	低电平脉冲有效。当数据传输方向为“输入”（从设备到主机）时，在此信号的上升沿，把数据写入模块。
nDMARD	读数据	输入	低电平脉冲有效。当数据传输方向为“输出”（从主机到设备）时，在此信号的上升沿后一段时间，下一有效数据出现在数据总线。
nDMAOE	输出使能	输入	低电平有效。当数据传输方向为“输出”时，如果为低电平，则有效数据出现在数据总线上。
nDMACS	模块选通	输入	模块选通，低电平有效。
nFIFOFULL	缓冲区满	输出	低电平有效。一般在“输入”时使用本信号。外部 DMA 控制器只有当本信号为高电平（缓冲区有空间）时才可以发“写数据”脉冲。
nFIFOEMPTY	缓冲区空	输出	低电平有效。一般在“输出”时使用本信号。外部 DMA 控制器只有当本信号为高电平（缓冲区有数据）时才可以发“读数据”脉冲。
nPKTEND	结束写	输入	由外部逻辑提供，低电平脉冲有效。
IFCLk	接口时钟信号	输出/输入	可以由本模块提供，此时该信号为输出，也可以由外部逻辑提供，此时信号为输入。
A[4..0]	地址线	输出	5 位地址线，与 D[12..8]复用
nPWR	写地址	输出	低电平有效，与 D13 复用
nPRD	读地址	输出	低电平有效，与 D14 复用
nPWAIT	地址等待	输入	低电平有效，与 D15 复用
DMAING	批量数据传输指示		本信号为数据传输模式指示信号，由本模块输出。 高电平：指示工作于批量数据传输模式。 低电平：指示工作于地址 IO 模式。
CLOCKOUT	时钟输出	输出	
5Vout	输出电源	输出	提供外部设备使用 5V 电源，电流小于 200mA

2.6.2 模块封装

本模块使用自定义的 DIP40 封装，管脚间距为 2.54mm(100mil)，双排管脚宽度为 25.4mm(1000mil)，模块外型尺寸：长×宽×高= 55.88mm ×33.78×10mm，单位用 mil 表示的话为长×宽= 2200mil ×1330mil，USB20D 模块的图片：



3 库函数使用说明

本模块以动态连接库的形式提供了一系列的 API 函数，用于简化应用程序的编写。

函数封装在 USB20D.DLL 中。函数分为以下四类：初始化函数、地址 IO 函数、批量传输函数、参数设置函数。

3.1 初始化函数

初始化函数主要用于完成本模块软件硬件的初始化工作，它包含以下四个函数：

USB20D_EnumDeviceCount

USB20D_Init

USB20D_Done

USB20D_WorkAtHighSpeed

3.1.1 USB20D_EnumDeviceCount

VC 原型： int _stdcall USB20D_EnumDeviceCount();

Delphi 调用： function USB20D_EnumDeviceCount : Integer;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_EnumDeviceCount Lib "USB20D" () As Long

VB.NET调用： Public Declare Function USB20D_EnumDeviceCount Lib "USB20D" () As Integer

功能： 确定当前系统上连接了几个本模块。

返回值： 返回连接到当前系统的本模块的个数。如果个数为 0，说明没有模块 USB20D 连接到当前系统。

3.1.2 USB20D_Init

VC 原型： HANDLE _stdcall USB20D_Init(int DevNo, int I2CChip);

Delphi 调用： function USB20D_Init(DevNo : integer; I2CChip : integer) : DevHandle;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_Init Lib "USB20D" (ByVal DevNo As Long, ByVal I2CChip As Long) As Long

VB.NET调用： Public Declare Function USB20D_Init Lib "USB20D" (ByVal DevNo As Integer, ByVal I2CCHhip As Integer) As Integer

功能： 与指定的模块建立连接。

入口参数：

DevNo 第几个连接到当前系统的模块。DevNo = 0 表示第 1 个连接到当前系统的模块。

I2CChip 本模块上的 I2C 器件型号。在此 I2C24c01 = 1;

返回值： 如果成功与指定的设备建立连接，则返回一个句柄，该句柄被后面的所有函数使用，**作为后面所有函数的第一个参数 DevHandle（后面不再说明）**。如果没有成功与指定的设备建立连接，则返回 0。可以使用函数 USB20D_GetLastError 得到错误代码，使用函数 USB20D_GetLastErrorStrC 得到错误提示字符串。

3.1.3 USB20D_Done

VC 原型： void _stdcall USB20D_Done(HANDLE DevHandle);

Delphi 调用： procedure USB20D_Done(Hnd: DevHandle);StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_Done Lib "USB20D" (ByVal DevHandle)

VB.NET调用： Public Declare Function USB20D_Done Lib "USB20D" (ByVal devHandle As Integer) As Integer

功能： 断开与模块的连接。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

3.1.4 USB20D_WorkAtHighSpeed

VC 原型： BOOL _stdcall USB20D_WorkAtHighSpeed(HANDLE DevHandle);

Delphi 调用： function USB20D_WorkAtHighSpeed(Hnd : DevHandle) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_WorkAtHighSpeed Lib "USB20D" (ByVal DevHandle As Long) As Boolean

VB.NET调用： Public Declare Function USB20D_WorkAtHighSpeed Lib "USB20D" (ByVal devHandle As Integer) As Boolean

功能： 检测模块是否工作于高速模式。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

返回值： 如果模块连接到一个 USB2.0 总线上，而且成功的枚举成高速工作模式，则返回 True（“真”）；否则返回 False。

3.2 地址 IO 函数

本类函数完成地址 IO 的功能，包含以下六个函数：

USB20D_SetAddress
USB20D_Input
USB20D_Output
USB20D_MultInput
USB20D_MultOutput
USB20D_MixedIO

3.2.1 USB20D_SetAddress

VC 原型： BOOL _stdcall USB20D_SetAddress(HANDLE DevHandle, int Addr);

Delphi 调用： function USB20D_SetAddress(Hnd : DevHandle; Addr : integer) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用：
```
Declare Function USB20D_SetAddress Lib "USB20D" (ByVal DevHandle As Long, ByVal iAdd As Long) As Boolean
```

VB.NET调用：
```
Public Declare Function USB20D_SetAddress Lib "USB20D" (ByVal devHandle As Integer, ByVal iAdd As Integer) As Boolean
```

功能： 设置地址总线上的地址。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

Addr 需要设置的地址，只有最低 5 位有效。

返回值： 如果设置成功则返回 True（“真”）。否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.2.2 USB20D_Input

VC 原型： int _stdcall USB20D_Input(HANDLE DevHandle, int Addr);

Delphi 调用： function USB20D_Input(Hnd : DevHandle; Addr : Integer) : Integer;StdCall; External 'USB20D.DLL';

VB6.0 调用：
```
Declare Function USB20D_Input Lib "USB20D" (ByVal DevHandle As Long, ByVal iAdd As Integer) As Integer
```

VB.NET调用：
```
Public Declare Function USB20D_Input Lib "USB20D" (ByVal devHandle As Integer, ByVal iAdd As Integer) As Integer
```

功能： 从指定地址（Addr）读一个字节数据。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

Addr 需要读的地址，只有最低 5 位有效。

返回值： 返回读到的数据。没有是否成功的指示，但是如果有错误发生，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.2.3 USB20D_Output

VC 原型： int _stdcall USB20D_Output(HANDLE DevHandle, int Addr, int Data);

Delphi 调用： function USB20D_Output(Hnd : DevHandle; Addr, Data : Integer) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_Output Lib "USB20D" (ByVal DevHandle As Long, ByVal iAdd As Long, ByVal iData As Long) As Boolean

VB.NET调用： Public Declare Function USB20D_Output Lib "USB20D" (ByVal devHandle As Integer, ByVal iAdd As Integer, ByVal iData As Integer) As Boolean

功能： 向指定地址（Addr）写一个字节数据（Data）。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

Addr 需要写的地址，只有最低 5 位有效。

Data 需要写的数据，只有最低字节有效。

返回值： 返回是否写数据成功。如果出错，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.2.4 USB20D_MultInput

VC 原型： BOOL _stdcall USB20D_MultInput(HANDLE DevHandle, int Num, void* Buf);

Delphi 调用： function USB20D_MultInput(Hnd : DevHandle; Num : integer; Buf: Pointer) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_MultInput Lib "USB20D" (ByVal DevHandle As Long, ByVal Num As Long, ByRef buf As Any) As Boolean

VB.NET调用： Public Declare Function USB20D_MultInput Lib "USB20D" (ByVal devHandle As Integer, ByVal Num As Integer, ByRef Buf As Object) As Boolean

功能： 多地址读。共需要从 Num 个地址读取 Num 个字节的数据。需要读数据的地址由 Buf 指向的字节数组指定，读到的数据存入 Buf 指定的数组。它把本来需要多个 USB20D_Input 函数的功能用一个函数来完成，这样做可以提高效率。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

Num 需要读的个数。此值应≤60。

Buf 指向一个字节数组。它既作为一个入口参数，又作为一个出口参数。作为入口参数，它指向的数组存有应用程序设置的需要读数据的地址，其格式为：

Buf[0] = 地址 0

Buf[1] = 地址 1

Buf[2] = 地址 2

……

作为一个出口参数，它指向的数组存有从指定的地址读到的数据，共应用程序使用，其格式为：

Buf[0] = 数据 0

Buf[1] = 数据 1

Buf[2] = 数据 2

……

返回值： 返回是否读数据成功。如果出错则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.2.5 USB20D_MultOutput

VC 原型： int _stdcall USB20D_Output(HANDLE DevHandle, int Addr, int Data);

Delphi 调用： function USB20D_MultOutput(Hnd : DevHandle; Num : integer; Buf: Pointer) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： `Declare Function USB20D_MultOutput Lib "USB20D" (ByVal DevHandle As Long, ByVal Num As Long, ByRef buf As Any) As Boolean`

VB.NET 调用： `Public Declare Function USB20D_MultOutput Lib "USB20D" (ByVal devHandle As Integer, ByVal Num As Integer, ByRef Buf As Object) As Boolean`

功能： 多地址写。共需要向 Num 个地址写 Num 个字节的数据。需要写的地址和数据由 Buf 指向的字节数组指定。它把本来需要多个 USB20D_Output 函数的功能用一个函数来完成，这样做可以提高效率。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

Num 需要写的个数。此值应≤30。

Buf 指向一个字节数组。它指向的数组既存有地址又存有数据，其格式为：

Buf[0] = 地址 0 Buf[1] = 数据 0
Buf[2] = 地址 1 Buf[3] = 数据 1
Buf[4] = 地址 2 Buf[5] = 数据 2
……

返回值： 返回是否写数据成功。如果出错则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.2.6 USB20D_MixedIO

VC 原型： BOOL _stdcall USB20D_MixedIO(HANDLE DevHandle, int Num, void* Buf);

Delphi 调用： function USB20D_MixedIO(Hnd : DevHandle; Num : integer; Buf: Pointer) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： `Declare Function USB20D_MixedIO Lib "USB20D" (ByVal DevHandle As Long, ByVal Num As Long, ByRef buf As Any) As Boolean`

VB.NET 调用： `Public Declare Function USB20D_MixedIO Lib "USB20D" (ByVal devHandle As Integer, ByVal Num As Integer, ByRef Buf As Object) As Boolean`

功能： 多地址混合读写。共需要向 Num 个地址读/写 Num 个字节的数据。需要读/写的地址和需要写的数据由 Buf 指向的字节数组指定。它把混合了多个 USB20D_Output 和 USB20D_Input 函数的功能，这样做可以提高效率。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

Num 需要读/写的个数。此值应≤30。

Buf 指向一个字节数组。它既作为一个入口参数，又作为一个出口参数。作为入口参数，它指向的数组既存有地址又存有数据，其格式为：

Buf[0] = 地址 0 Buf[1] = 数据 0
Buf[2] = 地址 1 Buf[3] = 数据 1
Buf[4] = 地址 2 Buf[5] = 数据 2
……

“地址 x”的最低 5 位为有效地址，如果“地址 x”的最高位为 1 则说明从地址读数据，紧跟其后的“数据 x”被忽略但是其位置仍保留。如果“地址 x”的最高位为 0 则说明从地址写数据，紧跟其后的“数据 x”被写入“地址 x”。作为出口参数，其指向的数组存有从指定地址读取到的数据，格式为：

Buf[0] = 数据 0
Buf[1] = 数据 1
Buf[2] = 数据 2
……

返回值： 返回是否写数据成功。如果出错则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.3 批量传输函数

本类函数完成传输大批量数据的功能，包含以下 12 个函数：

USB20D_SetDMAClk
USB20D_StartDMA
USB20D_EndDMA
USB20D_DMARead
USB20D_DMAWrite
USB20D_ResetDMAFIFO
USB20D_DMAOutFIFOEmpty
USB20D_DMAOutFIFOFull
USB20D_DMAInFIFOEmpty
USB20D_DMAInFIFOFull
USB20D_DMAFIFOStatus
USB20D_UnlockAfterDMA

3.3.1 USB20D_SetDMAClk

VC 原型： BOOL _stdcall USB20D_SetDMAClk (HANDLE DevHandle, BOOL IntIFClk, BOOL IntClk48MHz, BOOL IntClkOutEn, BOOL IFClkInvert, BOOL AsyncFIFO, BOOL OnePulseEarly);

Delphi 调用： function USB20D_SetDMAClk(Hnd : DevHandle; IntIFCLK, IntClk48MHz, IntClkOutEn, IFCLKInvert, AsyncFIFO, OnePulseEarly : BOOL) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用：
```
Declare Function USB20D_SetDMAClk Lib "USB20D" (ByVal DevHandle As Long, ByVal IntIFCLK As Boolean, ByVal intClk48MHz As Boolean, ByVal IntClkOutEn As Boolean, ByVal IFCLKInvert As Boolean, ByVal AsyncFIFO As Boolean, ByVal OnePulseEarly As Boolean) As Boolean
```

VB.NET调用：
```
Public Declare Function USB20D_SetDMAClk Lib "USB20D" (ByVal devHandle As Integer, ByVal IntIFCLK As Boolean, ByVal intClk48MHz As Boolean, ByVal IntClkOutEn As Boolean, ByVal IFCLKInvert As Boolean, ByVal AsyncFIFO As Boolean, ByVal OnePulseEarly As Boolean) As Boolean
```

功能： 设置DMA传输模式及参数。

入口参数：

IntIFCLK True 表示使用模块内部的同步时钟；False 表示使用外部逻辑的同步时钟。

IntClk48MHz True 表示模块内部的同步时钟为 48MHz；False 表示模块内部的同步时钟 30MHz。

IntClkOutEn True 表示可以输出模块内部的同步时钟；False 表示不输出。

IFCLKInvert True 表示翻转模块内部的同步时钟；False 表示不翻转。

AsyncFIFO True 表示使用异步 DMA 传输模式，False 表示使用同步 DMA 传输模式。

OnePulsEarly True 表示同步 DMA 传输时，FIFO 标志提前一个时钟周期指示，以配合外部逻辑工作。

返回值： 如果设置成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

使用说明： 有两种 DMA 传输模式：同步和异步。当 DMA 速率较慢时(例如采用单片机控制时)，建议使用异步 DMA 传输模式；当 DMA 速率较快时（例如采用 FPGA/CPLD 控制时），使用同步模式可以提高传输的可靠度。

3.3.2 USB20D_StartDMA

VC 原型： Int _stdcall USB20D_StartDMA(HANDLE DevHandle, BOOL DMARead, BOOL DB16, int Reserved);

Delphi 调用： function USB20D_StartDMA(Hnd : DevHandle; DMARead, DB16: BOOL; Reserved : integer) : BOOL; StdCall; External 'USB20D.DLL';

VB6.0 调用：
```
Declare Function USB20D_StartDMA Lib "USB20D" (ByVal DevHandle As Long, ByVal DMARead As Boolean, ByVal DB16 As Boolean, ByVal Reserved As Long) As Boolean
```

VB.NET 调用：
```
Public Declare Function USB20D_StartDMA Lib "USB20D" (ByVal devHandle As Integer, ByVal DMARead As Boolean, ByVal DB16 As Boolean, ByVal Reserved As Integer) As Boolean
```

功能： 启动批量读写数据功能。此函数执行后，硬件接口上的以下控制信号发生改变：
DMAING = 1
当 DMARead = “ 假 ”，进行 DMAWrite 时，DMADIR = 0
当 DMARead = “ 真 ”，进行 DMARead 时，DMADIR = 1

入口参数：

DMARead “ 真 ” 则开始执行 DMA 读（数据从设备到主机）。“ 假 ” DMA 写（数据从主机到设备）。

DB16 “ 真 ” 则使用 16 位 DMA。“ 假 ” 使用 8 位 DMA。

Reserved 保留。调用时，该参数应为 0。

返回值： 如果启动成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.3.3 USB20D_EndDMA

VC 原型： int _stdcall USB20D_EndDMA(HANDLE DevHandle);

Delphi 调用： function USB20D_EndDMA(Hnd: DevHandle) : BOOL; StdCall; External 'USB20D.DLL';

VB6.0 调用：
```
Declare Function USB20D_ EndDMA Lib "USB20D" (ByVal devHandle As Long) As Boolean
```

VB.NET调用：
```
Public Declare Function USB20D_EndDMA Lib "USB20D" (ByVal devHandle As Integer) As Boolean
```

功能： 结束批量数据传输功能。此函数执行后，硬件接口上的以下控制信号发生改变：
DMAING = 0

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

返回值： 如果结束成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.3.4 USB20D_DMARead

VC 原型： BOOL _stdcall USB20D_DMARead(HANDLE DevHandle, void* Buffer, DWORD DataNeed, DWORD* DataRead, long OverTime);

Delphi 调用： function USB20D_DMARead(Hnd : DevHandle; Buffer: Pointer; DataNeed : integer; Var

DataRead : integer; OverTime: integer): BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_DMARead Lib "USB20D" (ByVal DevHandle As Long, ByRef Buffer As Any, ByVal DataNeed As Long, ByRef DataRead As Any, ByVal OverTime As Long) As Boolean

VB.NET调用： Public Declare Function USB20D_DMARead Lib "USB20D" (ByVal devHandle As Integer, ByRef Buffer As Object, ByVal DataNeed As Integer, ByRef DataRead As Object, ByVal OverTime As Integer) As Boolean

功能： 批量读数据。读到的数据存入Buffer指定的数据接收缓冲区。

入口参数：

DevHandle	USB20D_Init 函数的返回值作为句柄。
Buffer	指向数据接收缓冲区。此缓冲区由应用程序分配。
DataNeed	需要读取的数据长度，此值应≤接收缓冲区大小。
OverTime	超时时间，以 ms 为单位。如果没有在指定的时间内读到指定长度的数据，则表示读数据不成功，读数据超时，函数返回错误。

出口参数：

DataRead	读到的数据长度。如果函数执行成功，此值应等于 DataNeed。

返回值： 如果读数据成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.3.5 USB20D_DMAWrite

VC 原型： BOOL _stdcall USB20D_DMAWrite(HANDLE DevHandle, void* Buffer, DWORD DataSize, DWORD* DataWritten, long OverTime);

Delphi 调用： function USB20D_DMAWrite(Hnd : DevHandle; Buffer: Pointer; DataSize : integer; Var DataWritten : integer; OverTime: integer): BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_DMAWrite Lib "USB20D" (ByVal DevHandle As Long, ByRef Buffer As Any, ByVal DataSize As Long, ByRef DataWritten As Any, ByVal OverTime As Long) As Boolean

VB.NET调用： Public Declare Function USB20D_DMAWrite Lib "USB20D" (ByVal DevHandle As Integer, ByRef Buffer As Object, ByVal DataSize As Integer, ByRef DataWritten As Object, ByVal OverTime As Integer) As Boolean

功能： 批量写数据。把 Buffer 指定的数据发送缓冲区内的数据发送到设备。

入口参数：

DevHandle	USB20D_Init 函数的返回值作为句柄。
Buffer	指向数据发送缓冲区。此缓冲区由应用程序分配，并填充需要发送的数据。
DataSize	需要发送的数据长度，此值应≤发送缓冲区大小。
OverTime	超时时间，以 ms 为单位。如果没有在指定的时间内把指定长度的数据的数据发送到设备，则表示写数据不成功，超时，函数返回错误。

出口参数：

DataWritten	写出的数据长度。如果函数执行成功，此值应等于 DataSize。

返回值： 如果写数据成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.3.6 USB20D_ResetDMAFIFO

VC 原型： BOOL _stdcall USB20D_ResetDMAFIFO(HANDLE DevHandle);

Delphi 调用： function USB20D_ResetDMAFIFO (Hnd: DevHandle);StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_ResetDMAFIFO Lib "USB20D" (ByVal DevHandle As Long) As Boolean

VB.NET调用： Public Declare Function USB20D_ResetDMAFIFO Lib "USB20D" (ByVal DevHandle As Integer) As Boolean

功能： 清空本模块内的缓冲区。

返回值： 如果设置成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

使用说明： 由于本模块内有缓冲区，数据都要在缓冲区内暂存，这样就有可能在缓冲区内存有一些“旧”的数据。可以使用本函数清除这些“旧”数据。比如，一个 AD 采样设备，在需要输入数据时，可能缓冲区内有以前采集到的数据，为了使这次读回来的数据都是最新的数据，可以先清空缓冲区。

注意： 输入和输出缓冲区将被同时清空。不能在 DMA 模式调用本函数，否则会导致不可预测的结果。

3.3.7 USB20D_DMAOutFIFOEmpty

VC 原型： BOOL _stdcall USB20D_DMAOutFIFOEmpty(HANDLE DevHandle);

Delphi 调用： function USB20D_ResetDMAFIFO (Hnd: DevHandle);StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_ResetDMAFIFO Lib "USB20D" (ByVal DevHandle As Long) As Boolean

VB.NET调用： Public Declare Function USB20D_ResetDMAFIFO Lib "USB20D" (ByVal DevHandle As Integer) As Boolean

功能： 查询本模块内的缓冲区的状态。

返回值： 如果输出缓冲区空，则返回“真”；否则，返回“假”。

3.3.8 USB20D_DMAOutFIFOFull

VC 原型： BOOL _stdcall USB20D_DMAOutFIFOFull(HANDLE DevHandle);

Delphi 调用： function USB20D_DMAOutFIFOFull(Hnd : DevHandle) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_DMAOutFIFOFull Lib "USB20D" (ByVal DevHandle As Long) As Boolean

VB.NET 调用： Public Declare Function USB20D_ DMAOutFIFOFull Lib "USB20D" (ByVal DevHandle As Integer) As Boolean

功能： 查询本模块内的缓冲区的状态。

返回值： 如果输出缓冲区满，则返回“真”；否则，返回“假”。

3.3.9 USB20D_DMAInFIFOEmpty

VC 原型： BOOL _stdcall USB20D_DMAInFIFOEmpty(HANDLE DevHandle);

Delphi 调用： function USB20D_DMAInFIFOFull(Hnd : DevHandle) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_ DMAInFIFOFull Lib "USB20D" (ByVal DevHandle As Long) As Boolean

VB.NET 调用： Public Declare Function USB20D_ DMAInFIFOFull Lib "USB20D" (ByVal DevHandle As Integer) As Boolean

功能： 查询本模块内的缓冲区的状态。

返回值： 如果输入缓冲区空，则返回“真”；否则，返回“假”。

3.3.10 USB20D_DMAFIFOStatus

VC 原型： BOOL _stdcall USB20D_DMAFIFOStatus(HANDLE DevHandle, BOOL * OutFIFOEmpty, BOOL * OutFIFOFull, BOOL * InFIFOEmpty, BOOL * InFIFOFull);

Delphi 调用： function USB20D_DMAFIFOStatus(Hnd : DevHandle; Var OutFIFOEmpty, OutFIFOFull, InFIFOEmpty, InFIFOFull: BOOL) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_ DMAFIFOStatus Lib "USB20D" (ByVal DevHandle As Long, ByVal OutFIFOEmpty As Boolean, ByVal OutFIFOFull As Boolean, ByVal InFIFOEmpty As Boolean, ByVal InFIFOFull As Boolean) As Boolean

VB.NET 调用： Public Declare Function USB20D_ DMAFIFOStatus Lib "USB20D" (ByVal DevHandle As Integer, ByVal OutFIFOEmpty As Boolean, ByVal OutFIFOFull As Boolean, ByVal InFIFOEmpty As Boolean, ByVal InFIFOFull As Boolean) As Boolean

功能： 查询本模块内的缓冲区的状态。

返回值： 本函数是上面四个函数的集合，。

3.3.11 USB20D_UnlockAfterDMA

VC 原型： BOOL _stdcall USB20D_UnlockAfterDMA(HANDLEDevHandle, BOOL SendUnlockCode);

Delphi 调用： function USB20D_UnlockAfterDMA(Hnd : DevHandle; SendUnlockCode : BOOL) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_ UnlockAfterDMA Lib "USB20D" (ByVal DevHandle As Long, ByVal SendUnlockCode As Boolean) As Boolean

VB.NET 调用： Public Declare Function USB20D_ UnlockAfterDMA Lib "USB20D" (ByVal DevHandle As Integer, ByVal SendUnlockCode As Boolean) As Boolean

功能： 防止数据线干扰。

入口参数：

SendUnlockCode　True 表示需要在 EndDMA 的时候发送 DMA 解锁密码；False 表示不发送 DMA 解锁密码。

返回值：　如果设置成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

使用说明：　因为使用了 16 位的 DMA 数据总线，所以，地址 IO 的读/写脉冲与数据线复用，这就有可能在读/写脉冲上产生干扰，导致一次意外的地址 IO。这将对使用 CPLD/FPGA 作为接口的设备产生影响，可能会导致外围设备不能正常工作。为了避免这种情况出现，设计了一个 DMA 锁定逻辑（参见 CPLD/FPGA 接口示例）。启动 DMA 后，逻辑锁定，禁止后续的地址 IO；结束 DMA 后，逻辑保持锁定，直到本模块输出解锁密码，才解除锁定，后续的 IO 才能继续操作。

3.4 通用函数

本类函数完成有关本模块的一些通用功能，包含以下三个函数：

USB20D_SetCPUCS

USB20D_GetLastError

USB20D_GetLastErrorStrC

3.4.1 USB20D_SetCPUClk

VC 原型： BOOL _stdcall USB20D_SetCPUClk(HANDLE DevHandle, int CpuSpeed, BOOL ClkOutEnable,BOOL ClkInvert);

Delphi 调用： function USB20D_SetCPUClk(Hnd : DevHandle; CpuSpeed : Integer; ClkOutEnable, ClkInvert : BOOL) : BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_SetCPUClk Lib "USB20D" (ByVal DevHandle As Long, ByVal CupSpeed As Long, ByVal ClkOutEnable As Boolean, ByVal ClkInvert As Boolean) As Boolean

VB.NET 调用： Public Declare Function USB20D_SetCPUClk Lib "USB20D" (ByVal DevHandle As Long, ByVal CupSpeed As Long, ByVal ClkOutEnable As Boolean, ByVal ClkInvert As Boolean) As Boolean

功能： 设置模块内单片机的工作频率。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

CpuSpeed 本参数设置单片机的工作频率。它只可以是以下三个数值之一：

CPU12MHz = 0x0; (十进制 0)

CPU24MHz = 0x08; (十进制 8)

CPU48MHz = 0x10; (十进制 16)

ClkOutEnable 为 True 则把单片机的工作时钟在硬件接口的 CLKOUT 引脚输出。

ClkInvert 为 True 则把在硬件接口的 CLKOUT 引脚输出的时钟信号翻转。

返回值： 如果设置成功则返回 True，否则返回 False，可以使用 USB20D_GetLastError 函数得到错误代码，使用 USB20D_GetLastErrorStrC 函数得到错误提示字符串。

3.4.2 USB20D_GetLastError

VC 原型： int _stdcall USB20d_GetLastError(HANDLE DevHandle);

Delphi 调用： function USB20D_GetLastError(Hnd : DevHandle) : Integer; StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_ GetLastError Lib "USB20D" (ByVal DevHandle As Long) As Long

VB.NET 调用：Public Declare Function USB20D_ GetLastError Lib "USB20D" (ByVal DevHandle As Integer) As Integer

功能： 得到最后一次传输的错误代码。

入口参数：

DevHandle USB20D_Init 函数的返回值作为句柄。

返回值： 返回最后一次传输的错误代码。

3.4.3 USB20D_GetLastErrorStrC

VC 原型： BOOL _stdcall USB20d_GetLastErrorStrC(HANDLE DevHandle, int* Size,

PCHAR Buf);

Delphi 调用： function USB20D_GetLastErrorStrC(Hnd : DevHandle; Var Size: Integer; PCh : PChar): BOOL;StdCall; External 'USB20D.DLL';

VB6.0 调用： Declare Function USB20D_GetLastErrorStrC Lib "USB20D" (ByVal DevHandle As Long, ByRef Size As Integer, ByVal PCh As String) As Boolean

VB.NET 调用： Declare Function USB20D_GetLastErrorStrC Lib "USB20D" (ByVal DevHandle As Long, ByRef Size As Integer, ByVal PCh As String) As Boolean

功能： 得到最后一次传输的错误提示字符串。

入口参数：

Size PCh 指向的缓冲区大小。

PCh 指向由应用程序分配的接收缓冲区。

入口参数：

Size 如果入口的 Size 值小于提示字符串长度，则函数不能成功执行，此参数返回成功接收提示字符串所需要的缓冲区大小。

返回值： 如果函数执行成功，则返回 True。

3.5 错误代码

- USB20D_ERR_Success = 0;
- USB20D_ERR_DeviceAlreadyOpen = 1; //初始化设备时，设备曾经被初始化过一次。
- USB20D_ERR_CannotFindDevice = 2; //初始化设备时，没有发现设备。
- USB20D_ERR_CannotOpenInfoOutPipe = 3; //初始化设备时，已经发现设备，但是不能打开辅助输出管道。
- USB20D_ERR_CannotOpenInfoInPipe = 4; //初始化设备时，已经发现设备，但是不能打开辅助输入管道。
- USB20D_ERR_CannotOpenMainOutPipe = 5; //初始化设备时，已经发现设备，但是不能打开主输出管道。
- USB20D_ERR_CannotOpenMainInPipe = 6; //初始化设备时，已经发现设备，但是不能打开主输入管道。
- USB20D_ERR_CannotCreateEndEvent = 7; //初始化设备是，没能创建终止事件。

- USB20D_ERR_IOTimeOver = 8; //读写操作超时。
- USB20D_ERR_IOOverlapError = 9; //读写操作错误(重叠)。
- USB20D_ERR_IOError = 10; //读写操作错误。

- USB20D_ERR_DeviceNotConnected = 11; //设备没有连接。
- USB20D_ERR_LastIONotCompleted = 12; //没有结束上次操作。
- USB20D_ERR_IllegalParameter = 13; //函数的参数不合适。
- USB20D_ERR_IllegalDevice = 14; //不是我们的设备。

3.6 在应用程序中引用动态连接库中的函数

3.6.1 在 VC 中引用

详细说明请参考资料光盘中提供的 USB20D.H 文档

3.6.2 在 Delphi 中引用

详细说明请参考资料光盘中提供的 USB20D_Delphi_Declare.DOC 文档

3.6.3 在 VB 中引用

详细说明请参考资料光盘中提供的 USB20D_VB_Declare.DOC 文档

3.6.4 在 VB.NET 中引用

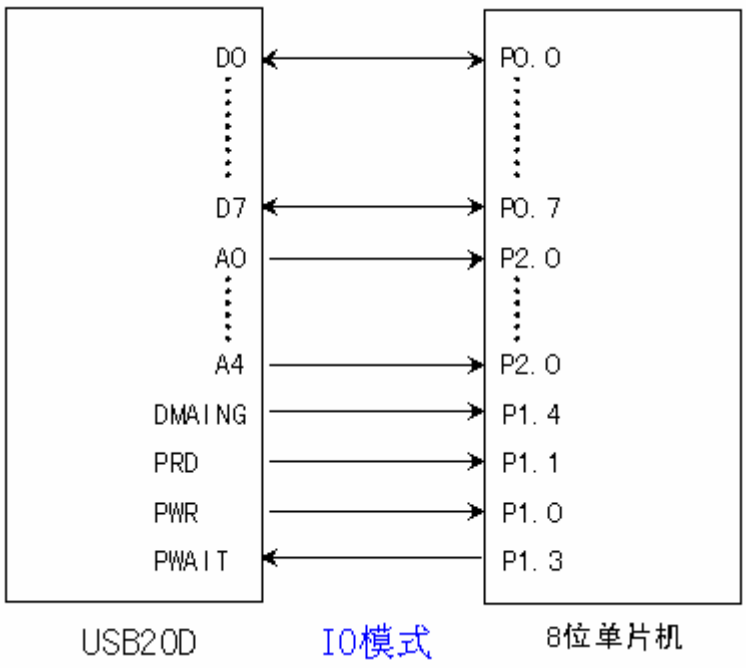
详细说明请参考资料光盘中提供的 USB20D_VBNET_Declare.DOC 文档

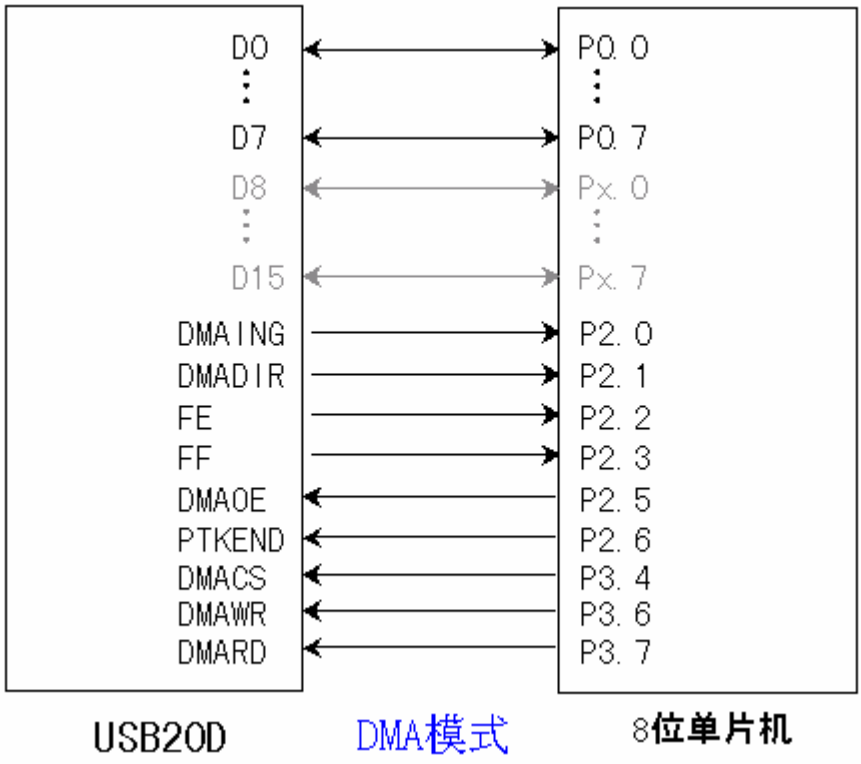
4 应用实例

4.1 外围控制器是单片机

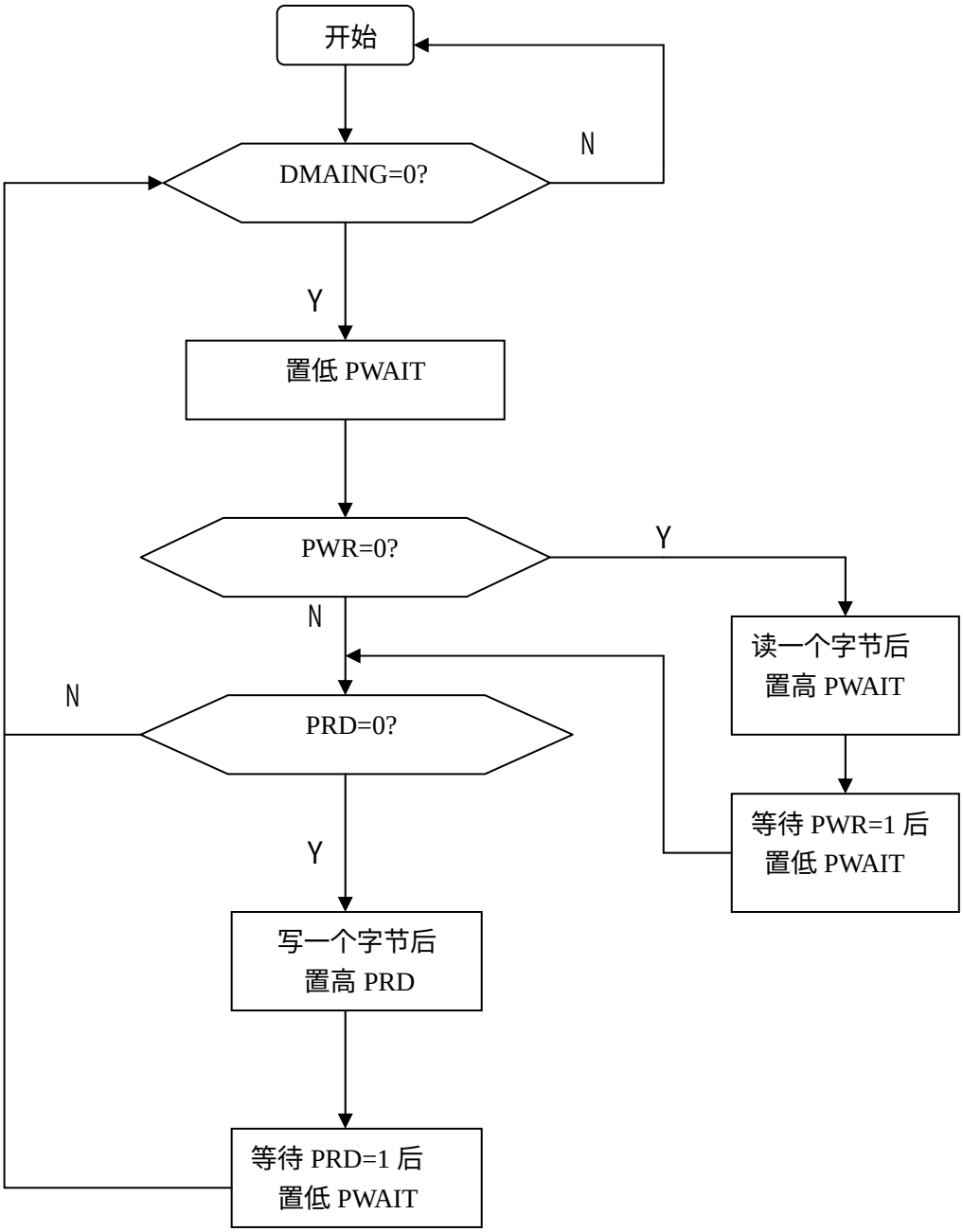
采用 8 位单片机进行异步 DMA 数据传输

4.1.1 硬件连接框图

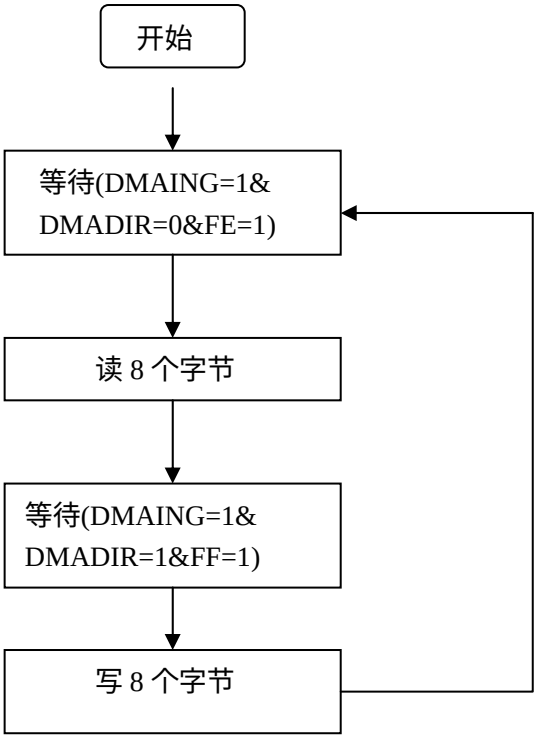



D0
D7
D8
D15
DMAING
DMADIR
FE
FF
DMAOE
PTKEND
DMACS
DMAWR
DMARD
P0.0
P0.7
Px.0
Px.7
P2.0
P2.1
P2.2
P2.3
P2.5
P2.6
P3.4
P3.6
P3.7
USB20D
DMA模式
8位单片机

4.1.2 单片机控制流程



IO 模式流程图

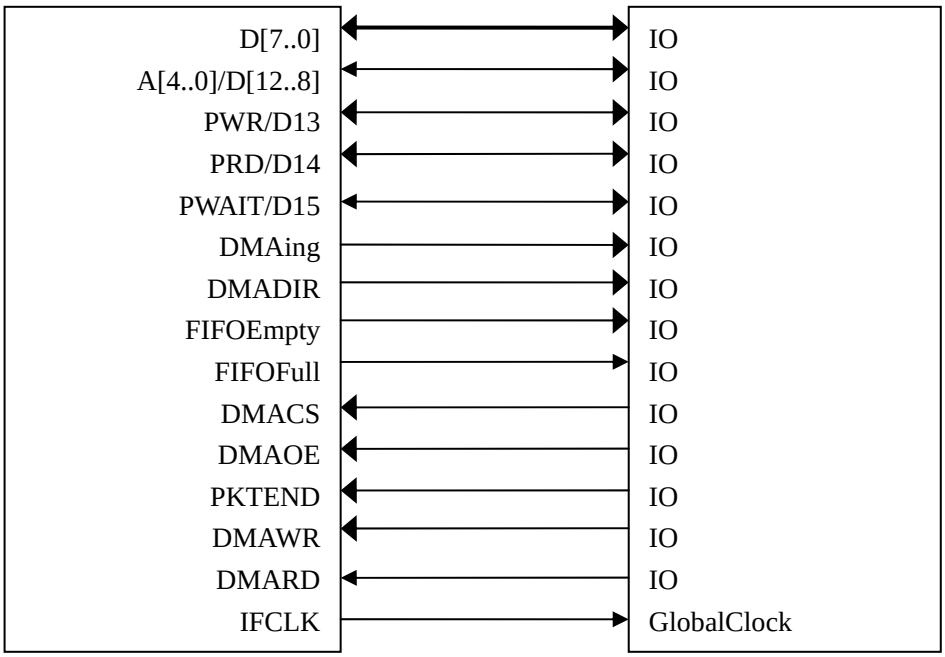


DMA 模式流程图

4.2 外围逻辑是 CPLD

采用 CPLD 进行同步 DMA 数据传输

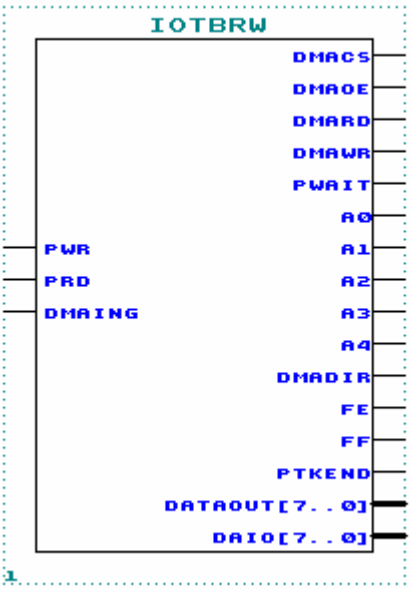
4.2.1 硬件连接框图



USB20D 模块　　方法一　　CPLD

4.2.2 CPLD 程序

下面以 VHDL 语言为例，示例如何设计 CPLD。
顶层设计：

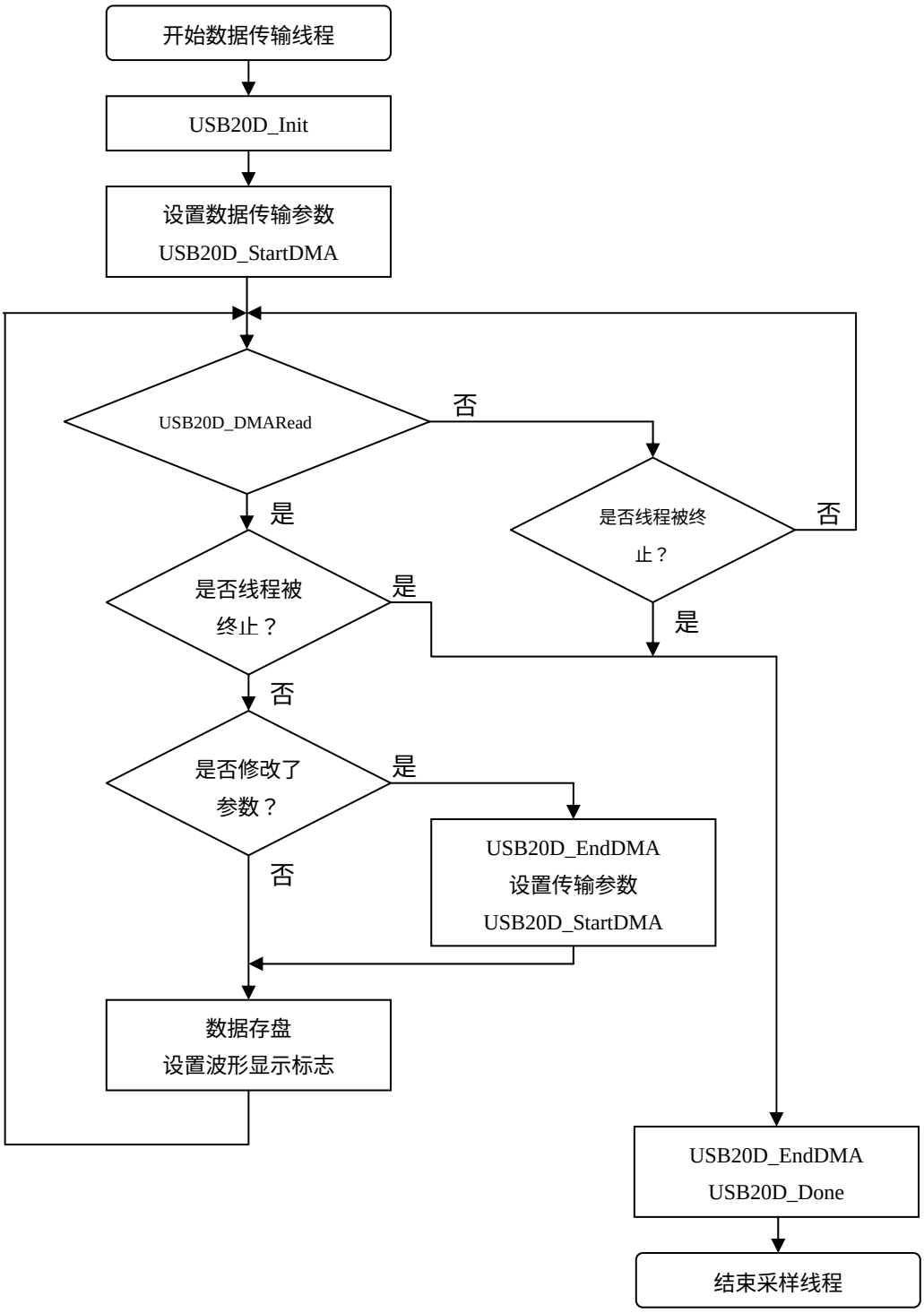


注：以上单片机，CPLD 及上位机例程见本公司所带附件。

4.3 主机应用程序

为了提高应用程序的运行效率、提高数据传输速率，主机应用程序应该使用多线程技术，把所有和数据传输相关的任务都放在“数据传输线程”中完成。

下面是一个典型的数据采集应用软件的数据传输线程的流程框图：



用户可以参考光盘中提供的**《USB2.0 模块技术问答》**。