RTX 平台下实时全软件数控雕刻系统①

王正友[1], 黄林林[2], 张国贤[2]

[1](上海出版印刷高等专科学校 出版与传播系, 上海 200093)
[2](上海大学 机电工程与自动化学院, 上海 200072)

摘 要: 计算机仿真设计与制造需要极高的实时性和可控性, 由于 Windows 操作系统本身实时性的缺陷, 给计算机仿真应用带来了一些不确定因素. 经过充分论证, 引入了 IntervalZero 公司的基于 Windows 的实时仿真解决方案 RTX, 并结合某型号数控雕刻机的设计要求, 总结出了 Windows 环境下 RTX 实时仿真应用系统的设计方法及 RTX 环境下全软件数控雕刻系统主要功能模块程序编写, 并将这种设计方法成功应用于某数控雕刻系统中. 实际雕刻实例表明 RTX 平台下全软件程序能够满足数控雕刻系统对实时性要求.

关键词: 实时扩展; 共享内存; 数控雕刻; 操作系统; 直线插补; 结构

Real-Time Full Software CNC Engraving System Based on RTX Platform

WANG Zheng-You[1], HUANG Lin-Lin[2], ZHANG Guo-Xian[2]

[1](Department of Publication and Communication, Shanghai Publishing and Printing College, Shanghai 200093, China)
[2](School of Electrical Engineering and Automation, Shanghai University, Shanghai 200072, China)

Abstract: The computer simulation design and manufacturing needs to be a very high level of real-time and controllability, Windows operating system itself as real-time nature of the defect, brought some uncertainties to the computer simulation applications. After full argument in this paper, the author has introduced the RTX, one based on the real-time Windows simulation solutions from IntervalZero Inc. Combined with the design requirements of a particular model of CNC engraving machine, we conclude that the RTX real-time simulation applications in a Windows environment, and we write the full software CNC engraving main function modules of the program in the RTX, and this design methodology was successfully applied to a certain CNC engraving system, The actual carving examples show that the full software programs can meet the requirements of CNC engraving system in real-time based on the RTX platform.

Key words: real-time eXtension; share memory; CNC engraving; operating system; linear interpolation; structure

1 引言

计算机图形技术的日益发展给零件的数控仿真加工带来了十分便利. 实用而可靠的仿真系统必须具备两个条件, 其一仿真系统能够逼近实物图形, 加工时实时性与可靠性高; 其二仿真系统具备好的交互性, 通用性即直观的图形界面和数据处理能力. Windows 操作系统能够满足第二条件的要求, 但它是个分时通用操作系统, 任何事件都通过消息队列排队, 当系统资源被无限占用或者系统定时器被更高优先级的中断占用时, 系统便不能实时中断, 系统的工作方式为抢占式. 市场上已有的实时操作系统虽能满足实时性的要求, 但是大部分后续数据处理能力不强大, 尤其是图形界面功能. 实时性与通用性这对矛盾怎么解决呢? 目前通用的解决方案是使用上下位机的结构, 上位机是基于 Windows 平台进行任务和程序设计及图形界面处理等非实时性任务, 下位机是基于实时操作系统进行实时性任务的操作.

目前, 一些专用数控三维雕刻设备都采用异构平台的上下位机工作模式, 能够解决实时性与图形界面之间的矛盾, 但是, 专用三维雕刻设备造价高, 维护

① 收稿时间:2012-05-03;收到修改稿时间:2012-06-02

性和扩展性很不方便.

RTX (Real-Time eXtension)的出现为这一问题的解决提供了可能, RTX 基于 Windows 应用环境, 支持微软 Visual studio 开发工具, 开发难度低、扩展性好、成本低、维护方便.

2 RTX[1-5]简介

RTX 是美国 IntervalZero 公司开发的一种基于 Windows 平台的纯软件硬实时扩展子系统, 该系统提供了对中断要求、输入与输出及计算机内存的精确控制, 并能按照优先级来执行指定任务, 完成任务的可靠性为 100%.

RTX 通过在 Windows 的硬件抽象层(HAL)增加实时扩展, 增加独立中断间隔, 来实现基于优先级的抢占式实时任务的管理与调度. RTX 修改 HAL 主要有 3 个目的:

(1) 借助中断隔离使 Windows 2007 线程和 Windows 2007 管理设备即不能干扰实时 RTSS (Real-Time Systems Symposium)进程, 也不能屏蔽实时 RTSS 管理设备. 确保了构建在 RTX 基础上的实时应用系统在 Windows 2007 系统死机时候, 健壮地运行而不受影响.

(2) RT-HAL 将 Windows 计时器的最小计时单位从 1 毫秒降到了 100 纳秒, 并提供三个同步(与计时器)时钟, 确保了实时操作系统精确执行任务.

(3) Windows 系统停止保护. 除中断管理和更快的定时器服务外, 实时 HAL 也提供了对 Windows 关机管理. 无论 Windows 系统是否正常关机, HAL 均能清除一切进程并复位硬件, 同时向操作者发出警示.

3 应用分析

数控雕刻机要达到计算机仿真系统的要求, 必须具备两个条件[6]: 其一是实时功能部分, 如硬件采集、网络通信及仿真计算等; 其二是非实时功能部分, 如生成编译目标文件、状态显示、存储数据等. 这种模式的实现是采用构建相同的计算机节点硬件平台, 分别利用 VC 的工程向导创建 MFC 工程与 RTX 工程, 通过共享内存通信, 然后按要求的功能分别在各自的工程模块内进行编码实现. 对于必须按时间点运行的进程、具有确定的实时任务由 RTX 处理, 对于初始参数设定、运行状态显示及存储等控制功能由 MFC(人机界面)非实时进程完成. 实时任务与非实时任务两个进程独立运行, 互不干扰, 两者之间的通信通过共享内存和事件体进行, 共享内存实现数据交互, 事件体实现进程同步.

4 系统的实现

4.1 系统平台开发

4.1.1 硬件平台

全软件数控系统与专用数控三维雕刻设备及传统的 PC+运动控制卡结构相比, 系统硬件结构简单、编程方便, 主要硬件有:

(1) PC 机、并口端、并口线;

(2) 步进电机驱动器(根据机械结构要求选择, 本系统采用的为一块航宇设计的 TA8435H 四轴雕刻机步进电机驱动器);

(3) 步进电机(根据机械结构要求为 4 个 42 型).

PC 机与步进电机驱动器的通讯工作由并口端和并口线完成, 图 1 是系统硬件结构.

4.1.2 软件平台

操作系统采用 Windows 2007, 其一是用户界面优化, 其二是代码优化, Windows 2007 适用范围更广.

实时操作系统采用 IntervalZero 公司的 RTX 2011, RTX 2011 显著降低基于 Windows 系统的开发成本, 并且可以支持最多包含 32 个处理器的系统.

Windows 2007 平台应用程序开发环境采用的是 Microsoft Visual Studio 2010, 该开发环境功能强大、设计到部署的开发过程简单.

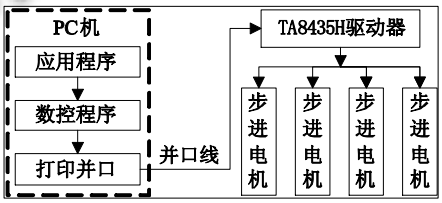


图 1 系统硬件结构

该软件平台具有以下特点:

(1) 具有丰富的人机交互界面和网络通讯协议, 丰富的 Win32 应用程序接口几乎可运行所有应用程序、支持多种开发工具.

(2) 数控系统对于实时性要求很高, 一般都在 1ms 以下, 对于 IRQs, I/O 和内存, RTX 2011 提供了精确的控制, 根据优先级来执行指定的任务, 任务执行

时可靠性达到 100%.

(3) Visual Studio 2010 是基于 Windows 平台的可视化编程环境(IDE). 强大的扩展性和灵活性, 给开发跨平台应用程序带来了方便, 成为现今广泛应用的 C++[7]语言集成开发环境.

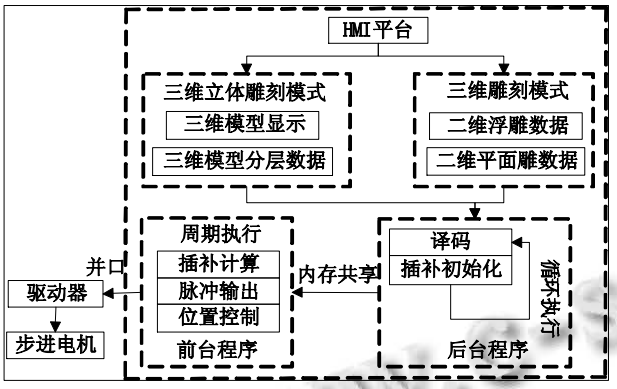


图 2 数控雕刻系统结构

4.1.3 系统结构

全软件数控雕刻系统的结构如图 2 所示. 数控功能由 Win32 进程和 RTSS 进程共同完成, 并由 RTSS 子系统创建共享内存来交换数据.

该数控系统结构中的各模块可模块化设计, 互相配合与协同工作, 最终完成数控系统的控制功能. 各模块功能描述如下:

(1) HMI 平台是数控系统和用户之间的对话窗口. 灵活性大, 也是最直接面对用户的程序窗口, 可根据用户需求设计人机交互界面.

(2) 数据处理软件主要包括以下两方面:

① 雕刻模型数据生成: 三维模型加载显示、三维模型分层数据、二维模型数据输入等;

② 通用数控软件: 负责系统控制、插补初始化、数据通讯等.

(3) 控制软件包括以下三方面:

① 运动控制: 控制系统需适用于双雕刻头雕刻机, 基本控制功能是实现四个坐标轴的运动控制, 采用 DDA 直线插补实现三轴联动.

② 速度控制: 数控雕刻机的加工特点是加工面积小、累计运动线程长、频繁启停, 要提高雕刻加工效率, 要求指令运动速度高, 加速快, 宜采用直线加减速法实现步进电机的加、减速功能.

③ 控制精度: 根据具体配套的雕刻机机械结构进行设置.

(4) 并口输出脉冲, 发送运动控制指令, 实现各轴的运动.

4.2 系统工作流程

由于系统采用以步进电机作为动力装置的开环控制系统, 因此其工作流程及数据流是单向进行的. 如图 3 所示.

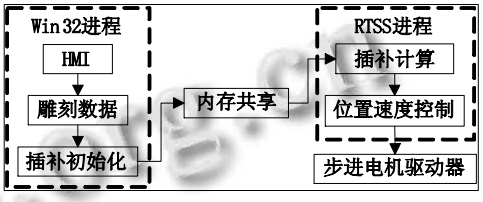


图 3 系统数据处理流程

数控雕刻机通过人机界面, 输入保存于计算机存储介质或外部存储介质上的具有一定格式的雕刻数据, 然后进行插补计算前的一些数据准备工作. 由于这些数据处理过程是非实时性的, 可在插补计算之前完成, 因此为不影响 RTSS 进程的实时性能, 将这些数据处理工作统归在 Win32 环境下完成. 在完成了数据初始化之后, 通过内存共享, 传输大量的数据到 RTSS 进程, 以供插补计算. 在给定起点、终点信息, 按照一定的插补算法, 实时计算雕刻机各轴运动量, 并向步进电机驱动器输出进给脉冲. 进给脉冲经过驱动器转换及放大, 控制各轴步进电机的角位移及速度, 以实现各轴的运动控制.

根据双雕刻头分层雕刻、累积成形的雕刻思想, 该数控系统首先将输入的雕刻数据进行偶数化插值, 使得数据在雕刻机的左右雕刻头对称分布, 以实现两雕刻头 Z 轴和 B 轴的三轴联动, 完成一层的雕刻工作, 然后 Y 轴上移 ΔY, 进入下一层雕刻, 如此往复, 直至完成雕刻.

5 系统主要功能模块

5.1 直线插补实现[8,9]

5.1.1 直线插补原理

设圆平面内关于原点对称有两条弧线 AB、CD, AB 起点为 A(^{r1}A,$^{\varphi}A$), 终点为 B(^{r1}B,$^{\varphi}B$); CD 起点为 C(^{r2}C,$^{\varphi}C$), 终点为 D(^{r2}D,$^{\varphi}D$). 系统采用柱面坐标, 并且两圆弧关于原点对称, 因此两圆弧可由两直线 AB、CD 实现, 也可唯一确定 R1、R2、Ψ 轴

直线段.

$$\begin{cases} R1={}^{r1}B-{}^{r1}A \\ R2={}^{r2}D-{}^{r2}C \\ \Psi={}^{\varphi}B-{}^{\varphi}A={}^{\varphi}D-{}^{\varphi}C \end{cases} \quad (1)$$

从而可得到直线段的参数表达式:

$$\begin{cases} R1={}^{r1}A+({}^{r1}B-{}^{r1}A)t \\ R2={}^{r2}C+({}^{r2}D-{}^{r2}C)t \\ \Psi={}^{\varphi}A+({}^{\varphi}B-{}^{\varphi}A)t \end{cases} \quad (2)$$

其中(0≤t≤1)

通过将方程组两边对参数 t 求导, 得到如下微分表达式:

$$\begin{cases} dR1=({}^{r1}B-{}^{r1}A)dt \\ dR2=({}^{r2}D-{}^{r2}C)dt \\ d\Psi=({}^{\varphi}B-{}^{\varphi}A)dt \end{cases} \quad (3)$$

其中(0≤t≤1)

然后对(3)式进行离散化, 用累加代替积分可得空间直线的累加函数:

$$\begin{cases} R1_i=\sum_{i=0}^{m}({}^{r1}B-{}^{r1}A)\Delta t \\ R2_i=\sum_{i=0}^{m}({}^{r2}D-{}^{r2}C)\Delta t \\ \Psi_i=\sum_{i=0}^{m}({}^{\varphi}B-{}^{\varphi}A)\Delta t \end{cases} \quad (4)$$

其中 m 为从起点到当前点的累加次数.

若取 $\Delta t=1$, 则上式可写为近似微分形式:

$$\begin{cases} \Delta R1={}^{r1}B-{}^{r1}A \\ \Delta R2={}^{r2}D-{}^{r2}C \\ \Delta\Psi={}^{\varphi}B-{}^{\varphi}A \end{cases} \quad (5)$$

通过(5)式可以看出, 当 AB 从点 A 向点 B 移动, 同时 CD 从点 C 向点 D 移动过程中, 每隔单位时间 Δt, R1、R2、Ψ轴分别累加△R1、△R2、△Ψ, 当累加值超过累加器容量时, 产生一个溢出脉冲, 同时累加器的值对累加器容量取模. 如此循环往复, 点 A 逐渐向 B 逼近, 点 C 逐渐向 D 逼近.

为提高插补效率, 也就是每次插补至少在一个轴上能产生溢出脉冲, 应使插补累加器容量 S 在满足要求的情况下(S≥max(△R1,△R2,△Ψ), 尽可能取小, 也就是 S=max(△R1,△R2,△Ψ), 这样就可以保证每次都是有效插补, 减少插补时间, 提高插补效率. 不过此方法仅限于直线插补, 在累加函数的曲线插补中将不能适用.

直线插补的终点判断比较简单. 首先, 通过各轴脉冲当量计算出各轴所要求的脉冲数. 在插补过程中, 记录各轴已溢出脉冲数, 判断是否达到所需脉冲数, 如果达到, 则表示该轴已到达插补终点, 如所有轴都已到达插补终点, 则插补结束.

5.1.2 直线插补流程

数字积分直线插补流程如图 4 所示, 各变量已相应转化为脉冲数, 在下面的进程通讯及线程同步中需要调用这些变量值, 其中各变量定义如下: F、R1、R2 分别为Ψ、R1、R2 轴被积函数值; JF、JR1、JR2 为各脉冲输出数余量; SF、SR1、SR2 为累加值余量; S 为累加器容量, S=max(F,R1,R2). 初始化时, 为加快累加速度, SF=SR1=SR2=S/2; JF=F; JR1=R1; JR2=R2. 以 RTX 提供的分辨率 100 纳秒, 定时间隔 100 微秒的定时器, 作为插补计算定时器, 每 100 微秒执行一次插补计算, 各轴进行一次被积函数值累加.

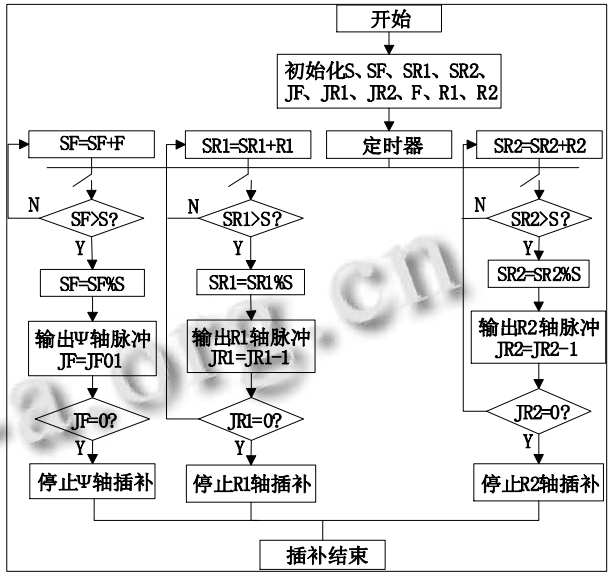


图 4　直线插补流程

5.2 进程通讯及线程同步

5.2.1 进程通讯

RTX 提供了多种进程通讯方式, 如事件、互斥及共享内存. 该数控系统通过共享内存方式, 进行 Win32 进程与 RTSS 进程间的数据通讯. 由于数据是从 Win32 进程单方向朝 RTSS 进程传输, 因此将用于存储数据的数据结构体定义在 RTSS 进程中; 然后在 Win32 进程工作区中定义一个指针指向该结构体, 并创建一

块共享内存，将其内存指针赋予数据结构体指针．最后，在 RTSS 进程中打开该共享内存并将该内存指针赋予指向数据结构体指针．这样就实现了 Win32 与 RTSS 进程间的数据通讯了．

本数控系统所定义的数据结构体如下所示:

```
struct RTXdata
{   BOOL ResetH0,Start,Stop,StartCB; //各程序段的执行开关
    int ChangeRR;           //R2 轴运动方向
    int ChangeRL;           //R1 轴运动方向
    int ChangeF;            //Ψ轴运动方向
    int ChangeH;            //Y 轴运动方向
    int slicenum;           //雕刻模型层数
    int pointnum;           //雕刻模型每层数据点数
    long sliceH[500];       //雕刻模型各层高度
    long theta;             //雕刻模型 F 值
    long rLeft[500][500];   //雕刻模型 R1 值
    long rRight[500][500];  //雕刻模型 R2 值
};
```

Win32 进程中创建共享内存的实现代码如下:

```
BOOL OpenSharedMemory()
{   //create shared memory for RTSS process
    If (!(hSharedMemory = RtCreateSharedMemory(PAGE_READWRITE,0,(DWORD)sizeof(RTXdata),sharedmemory,&location)))
    {   System::Windows::Forms::MessageBox::Show("Can't open rtss shared memory");
    return FALSE;
    }
    gmdata = (struct RTXdata*)location;
    //将创建的指向该共享内存的指针赋给 Win32 环境下声明的数据结构体指针
    return TRUE;
}
```

RTSS 进程中打开共享内存的实现代码如下:

```
if((hgmdata= RtOpenSharedMemory(SHM_MAP_WRITE, FALSE, sharedmemory,&location)) == NULL)
    {   RtWprintf(L"RtOpenSharedMemory error = %d\n", GetLastError());
    ExitProcess(1);
    }
gmdata = (struct RTXdata*)location; //将打开的指向该共享内存的指针赋给 RTSS 环境下声明的数据结构体指针
```

5.2.2 线程同步

在设计中采用基于时间片的循环轮转调度策略．以 RTX 提供的分辨率 100 纳秒，定时间隔 100 微秒的定时器，作为插补计算线程的定时器；以定时间隔 500 微秒的定时器，作为接收限位信号及各轴进给脉冲输出线程的定时器．

这样就有可能会出现，系统在运行各轴进给脉冲输出线程时，由于时间片用完，还未及时输出脉冲，系统又进入了定时时间短的插补线程，造成了两个或多个线程在时间片轮转调度下交替进行，使得输出脉冲混乱，脉冲频率不稳的情况．因此，必须为系统加上线程互斥同步机制．

常用的线程互斥同步多采用互斥体对象，RTX 也提供了用于创建互斥体对象的 RtCreateMutex 函数．

首先，在主程序下通过语句 hMutex= RtCreate Mutex(NULL，FALSE，NULL)创建互斥体对象，并赋给 hMutex；然后在各线程开始处添加 RtWait For SingleObject(hMutex，INFINITE)，该语句表示无限时间等待获取空闲的互斥体对象 hMutex；最后，在该线程结束前，通过语句 RtReleaseMutex(hMutex)释放之前所获取的互斥体对象 hMutex．最后一步是至关重要的，如果已获取互斥体对象的线程，在结束前未释放该互斥体对象，则其他线程将得不到执行．

此外，互斥无法限制访问者对资源的访问顺序，即访问是无序的，因此还必须在互斥的基础上，通过其它机制实现访问者对资源的有序访问，系统采用布尔值实现逻辑上有序访问．

5.3 并口通讯

系统使用计算机上的打印并口，作为与步进电机驱动器通讯的接口．各引脚定义如表 1.

表 1 并口各引脚定义

PIN1	PIN4	PIN14	PIN7	PIN17	PIN2
Y 轴允许	Y 轴方向	Y 轴脉冲	R2 轴允许	R2 轴方向	R2 轴脉冲
PIN6	PIN8	PIN3	PIN6	PIN9	PIN5
Ψ轴允许	Ψ轴方向	Ψ轴脉冲	R1 轴允许	R1 轴方向	R1 轴脉冲
PIN16	PIN10	PIN11	PIN12	PIN13	
主轴电机	Y 轴限位	R2 轴限位	Ψ轴限位	R1 轴限位	

在程序开发过程中采用图 5 所示的并口开发调试工具作为辅助工具，可以更加方便、有效的进行并口通讯程序的开发. 虽然 Windows 禁止对底层接口的直接访问，但是 RTX 提供了可对底层接口的直接进行读写操作的函数 RtReadPortUchar 和 RtWritePortUchar. 在进行对并口的读写操作之前，只需通过对 RTX Properties 上的硬件配置做简单的设置，将打印并口置于 RTX 管理之下即可.

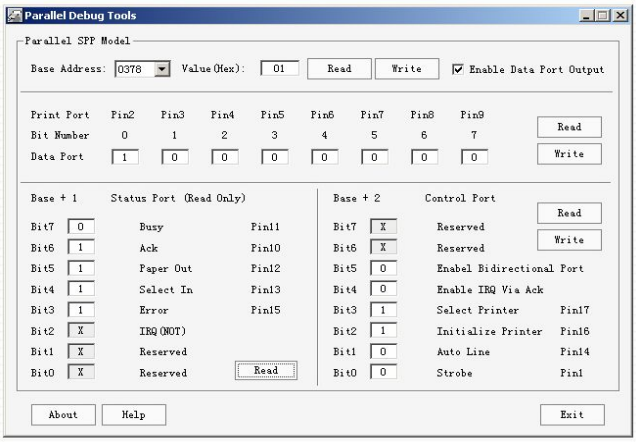


图 5　并口开发调试工具

5.4 加减速控制[10-12]

为避免步进电机频繁启停或速度突变时发生丢步、过冲、堵转等现象，系统还必须对步进电机进行加减速控制. 因此必须在电机加减速控制前，进行数据预处理. 本系统根据各相连微小线段间的夹角，以5°为临界值，进行加减速段的划分，确定一条具有完整加减速段 S. 这样可将相连的夹角小于 5° 的多条直线段，当作一条具有完整加减速段 S，进行统一的加减速控制，减少电机频繁的加减速.

其流程如图 6 所示，其中各变量定义如下：数据点总数 n；i 点坐标(x_i, y_i)；i 点矢量 l_i，初始化为 $l_0=\sqrt{x_0^2+y_0^2}$；i 点与原点夹角 θ_i，初始化为 $\theta_0=\arcsin\frac{y_0}{l_0}$；当前加减速段 S，初始化为 $S_0=l_0$；第 i 条加减速段总脉冲数 P_{Si}.

电机加减速常用的方法有直线加减速法和指数加减速法. 本系统采用算法较简单的直线加减速法，其原理如图 7 所示.

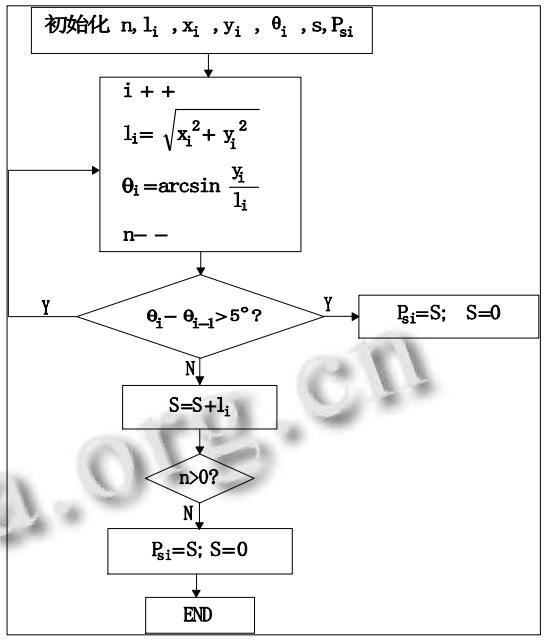


图 6　数据预处理流程

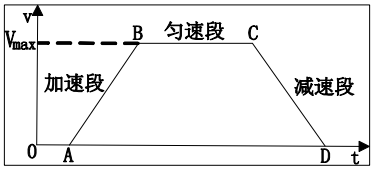


图 7　直线加减速法

直线加减速法流程如图 8 所示，其中各变量定义及初始化如下：当前位移值 S；位移累加值 S_i；速度 V_i，初速度 $V_0=0$；最大速度 V_{max}、加速度 a，可根据系统要求具体定义；当前加减速段总脉冲数 P_{sj}；加速段已输出脉冲数 P_{Add}；已输出总脉冲数 P_{All}.

该直线加减速法中加速度和减速度是相等的恒定值，根据速度公式：

加速阶段为：
$$V_i=V_{i-1}+a\Delta t$$
$$S_i=V_{i-1}\Delta t+\frac{1}{2}\Delta t^2$$

匀速阶段为：
$$V=V_{max}$$
$$S_i=V_{max}\Delta t$$

减速阶段为：
$$V_i=V_{i-1}-a\Delta t$$
$$S_i=V_{i-1}\Delta t-\frac{1}{2}\Delta t^2$$

随着 S 的累加，当 $S\geqslant\delta$(δ为长轴脉冲当量)时，执行一次 DDA 直线插补，因此，当在加速阶段时，随着 v 的增大，S 累加速度加快，DDA 直线插补率提高，脉冲输出频率上升，步进电机加速；当在匀速阶段时，

S 以稳定的速度 V_{max} 累加，系统插补速率稳定，脉冲输出频率恒定，步进电机匀速运行；在减速阶段时，随着 v 的减小，S 累加速度减慢，插补速率降低，脉冲输出频率减小，电机减速运行直至速度为零．这样即可实现步进电机的直线加减速控制．

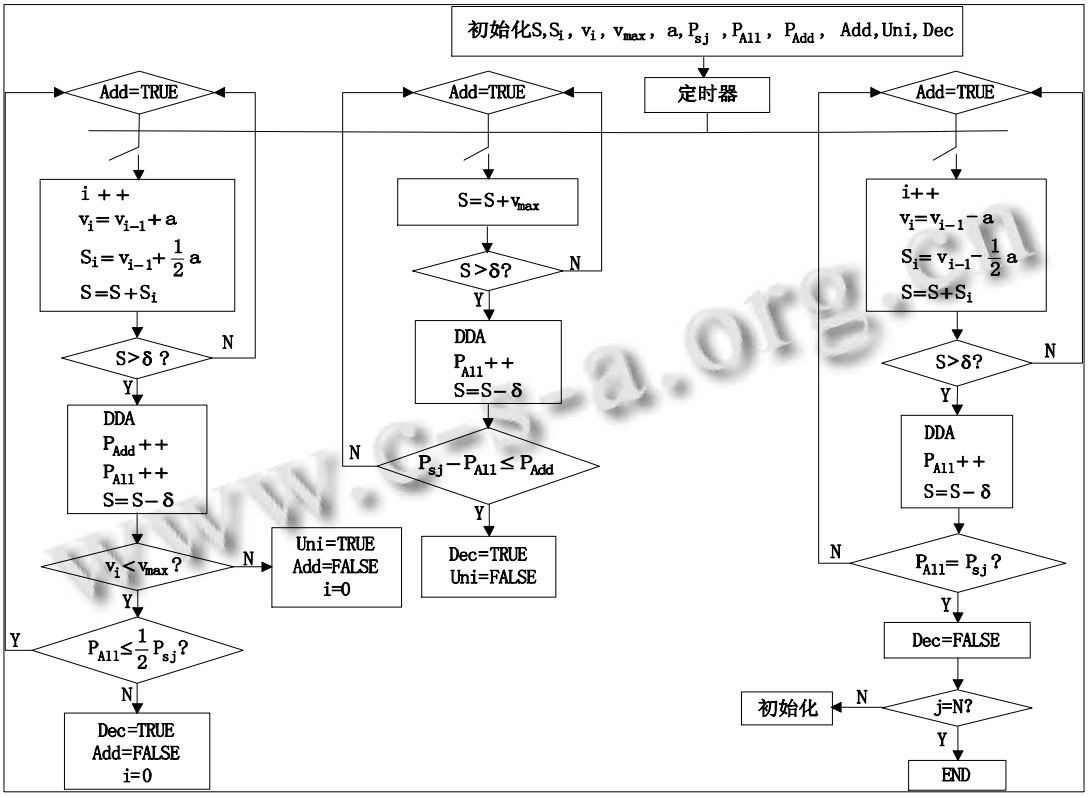


图 8　加减速控制流程

6　结语

采用美国 IntervalZero 公司开发的基于 Windows 系统的实时解决方案 RTX，解决了实时应用系统中实时性与通用性之间的矛盾，成本低、稳定性高、开放性好、程序开发容易、代码移植性和扩展性好，并且具备友好的人机交互界面．

本文详细介绍了 RTX 架构，任务调度机制，基于 Win32 API 的实时 API——RtWinAPI，说明了 RTX 实现硬实时的机理．并在 Windows 2007+RTX 2011 的实时扩展平台上，设计开发了全软件数控雕刻原型系统．详细阐明了系统结构，工作流程及各主要功能模块的实现，包括直线插补的实现，Win32 与 RTSS 进程间的通讯、同步，并口与步进电机间的通讯及步进电机的加减速控制．经过三维和二维的雕刻(图 9，图 10，图 11)测试，该系统实时性能高，运行可靠，为将来 RTX 在数控雕刻数控化系统中推广使用提供了可行性．

图 9　卡通人偶三维雕刻

图 10　二维平面雕刻

图 11　二维浮雕

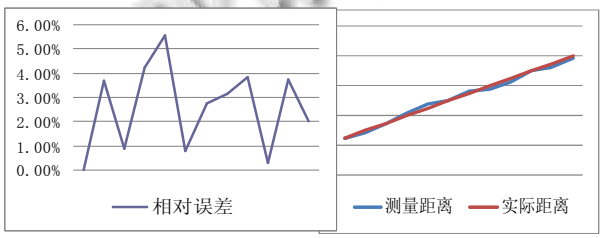
参考文献

1 张蕾.基于 RTX 的纯软件数控系统的研究.秦皇岛:燕山大学,2006.

(下转第 37 页)

表 1 前车车距测量结果表

标定参数	α:0.805655 β:1.202577		
实验次数	测量距离	实际距离	相对误差
1	20.00m	20m	0.00%
2	41.47m	40m	3.67%
3	60.52m	60m	0.87%
4	72.98m	70m	4.25%
5	84.45m	80m	5.56%
6	90.72m	90m	0.80%
7	102.70m	100m	2.70%
8	113.49m	110m	3.17%
9	124.12m	120m	3.43%
10	130.38m	130m	0.29%
11	145.22m	140m	3.73%
12	153.00m	150m	2.00%



(a) 测量误差图 (b) 测量距离与实际距离比较图

图 10 车距测量结果图

5 结语

本文以 ARM11 器件 SAMSUNG S3C6410 为核心搭建了安全车距监测系统的硬件平台，并采用多特征信息融合和单目视觉原理相实现车距的实时监测. 实验结果表明，该系统在满足实时性的同时能有效、准确的检测前方车辆的车距，能适用于智能汽车、安全辅助驾驶和车辆预警等系统中.

参考文献

1 梁忠原.基于车-车通信安全距离模型的驾驶员辅助决策研究[硕士论文学位].哈尔滨:哈尔滨工业大学,2010.1–12.

2 TVP5150A datasheet. Texas Instruments Incorporated.March 2004.

3 The SAMSUNG S3C6410 Mobile Processor Data Manual. Literature Number: DS-08-SLSI-001.SAMSUNG Electronics Co, April 2008.

4 代科学,李国辉,涂丹,袁见.监控视频运动目标检测背景技术的研究现状和展望.中国图象图形学报,2006,11(7):919–927.

5 姚敏.数字图像处理.北京:机械工业出版社,2006.256–258.

6 徐友春.智能车辆视觉与 GPS 综合导航方法的研究[博士论文论文].长春:吉林大学,2001.5–12.

7 谢云,杨宜民.自主足球机器人的单目视觉自定位方法.微电子学与计算机,2005,22(10):129–132.

8 Katsuki R, Ota J, Mizuta T, Kito T, Arai T, Ueyama T, Nishiyama T. Design of an artificial mark to determine 3D pose by monocular vision. IEEE International Conference on Robotics and Automation, 2003,1(9):995–1000.

9 于洪川,吴福朝,袁波.基于主动视觉的摄像机自标定方法.机器人,1999,21(1):l–7.

(上接第 7 页)

2 陈宗雨,王立峰,郭伟,李从心.一种新型开放式数控原型系统的开发.机床与液压,2006,(10):180–182.

3 王普,张蕾,郝立伟.基于 RTX 的纯软件数控系统的研究.燕山大学学报,2007,31(6):513–516.

4 李宏科.一种基于 RTX 的实时系统的实现.装备制造技术.2006,(3):55–57.

5 田昊,潘清.RTX 实时效果测试及应用.计算机系统应用,2007,16(2):103–106.

6 黄键,宋晓,薛顺虎.RTX 平台下实时仿真系统的设计方法.计算机应用与研究,2009,26(4):167–169.

7 Kruglinski DJ, Wingo S, Shepherd G. Visual C ++ 6.0 技术内幕.北京:希望电子出版社,1999.

8 王爱玲.现代数控原理及控制系统.长沙:国防工业出版社,2003.

9 江小勇.雕刻机数控代码自动生成的研究.南京:河海大学,2007.

10 蒋志冬,丁庆生.基于 FPGA 实现的多轴数控雕刻机系统.机电产品开发与创新,2006,19(1):129–131.

11 肖本贤.喷墨绘图机中步进电机的升降速控制.微机电.2001,34(4):28–33.

12 徐志明,陈金成,冯正进,蒋厚宗.Windows 平台上三轴联动数控雕刻机的开发.制造技术与机床,2002,(4):16–18.