

发动机工况实时调节软件设计

施先旺, 王鹏武

(西安航天动力研究所, 陕西 西安 710100)

摘 要: 介绍了基于 DSP/BIOS 的某发动机工况实时调节软件的设计, 包括需求分析、技术方案分析、软件设计与实现等。利用 DSP/BIOS 基于任务优先级的多线程机制, 完成步进脉冲分配、调节器控制、工况参数解算及工况调节等, 实现了发动机工况的实时、可靠调节。软件经过充分的测试、验证, 其功能和性能完全满足要求。

关键词: 发动机控制系统; 发动机工况调节; 步进电机控制; 燃料调节器; DSP/BIOS

中图分类号: V434-34 **文献标识码:** A **文章编号:** 1672-9374 (2012) 05-0070-07

Design of real-time regulation software for engine power

SHI Xian-wang, WANG Peng-wu

(Xi'an Aerospace Propulsion Institute, Xi'an 710100, China)

Abstract: The design of a real-time engine power regulation software based on DSP/BIOS is introduced, including requirement analysis, design approaches and implementation of software. With the multi-threading capability based on task priority provided by DSP/BIOS, the tasks including pulse arrangement for step motor driver, regulator control, resolving of power parameters and power adjustment were fulfilled, thus reliable real-time power regulation was realized. The verification and validation of the software show that its function and performance meet the requirements thoroughly.

Keywords: engine control system; engine power regulation; step motor control; fuel regulator; DSP/BIOS

0 引言

为满足用户要求, 某发动机控制系统需要对发动机工况进行实时调节, 并同时完成数据采集、通讯、时序控制等多种实时任务。基于 DSP/BIOS 嵌入式实时多任务操作系统设计了实时控

制软件, 实现了发动机工况的全量程大范围调节, 拓展了其性能和应用范围。

1 发动机工况调节原理

某发动机系统中, 在燃料、助燃剂主管路上安装了调节器, 通过控制调节器开度来控制燃料

收稿日期: 2010-10-04; 修回日期: 2011-11-08

基金项目: 国家 863 项目(2006AA702303)

作者简介: 施先旺 (1968—), 男, 研究员, 研究领域为发动机检测与控制、嵌入式系统

和助燃剂流量,从而达到调节发动机工况的目的。调节器采用步进电机作为调节部件,步进电机为三相混合式,工作在定频定速、单双六拍方式。工作中步进电机驱动调节器阀芯,从而改变调节器阀门开度。

在发动机工作过程中,系统控制计算机根据运动控制要求,向发动机控制系统发送工况调节指令;控制系统检测到工况调节指令后,解析工况设定值,按设定速率逐步换算为 2 个调节器步进电机电作脉冲数并按设定的频率、脉冲数、各相通电顺序改变步进电机各相励磁状态;步进电机运动方向和运动距离改变,使调节器阀门开度发生定量变化,从而使流过调节器的流量达到预期值,发动机工况随即自动调整为设定值。

2 软件需求分析

2.1 发动机控制系统

发动机控制系统采用了 32 位嵌入式 DSP 微控制器 TMS320F2812,外部时钟 30 MHz,工作频率 120 MHz,扩展 RS-422 串口、离散量输出端口。TMS320F2812 可采用专用 C/C++ 编译器开发,可运行嵌入式实时操作系统 DSP/BIOS。

工作过程中,控制系统软件实时接收 RS-422 通讯包,识别、解析工况设定值,解算步进电机实时运动方向和运动距离,进而通过软件完成脉冲分配,改变离散量输出状态,从而改变 2 个电机各三相线圈的励磁状态。工况调节过程相关的数据、状态通过 RS-422 数据包实时上传。

2.2 工况自动调节功能分析

为确保系统状态以及发动机工况调节过程中工作稳定性,必须满足工况调节速率、调节范围、调节精度、调节稳定性、调节器动作协调性等要求。为此,实现工况自动调节应包括以下 6 项内容:

1) 控制调节器步进电机,按设定工作频率、脉冲数、方向,完成脉冲分配;

2) 检测调节器位移传感器触点,确认调节器阀芯位置;

3) 使调节器自动回额定定点,即精确调整为额定工况对应的阀门开度;

4) 保持两台调节器同步,即按相同速率和各自设定动作脉冲数,同向同时动作;

5) 根据发动机各工况与脉冲数关系自动获取两个调节器对应的脉冲数,按工况调节速率、步长,使工况达到设定值;

6) 接收系统控制计算机工况设定命令,回传工况调节过程数据。

3 软件设计

3.1 DSP/BIOS 简介

为实现上述步进电机控制、调节器检测与控制、工况解算及控制、实时通讯等功能,满足实时性、可靠性及安全性要求,工况调节软件基于实时多任务架构设计,采用了 TI 公司嵌入式实时操作系统 DSP/BIOS。

DSP/BIOS 包括优化的实时函数库、实时操作系统服务、易用的配置工具、实时分析工具等。DSP/BIOS 实现了一个基于固定优先级的抢先式调度器,任务调度以线程优先级为基础,高优先级的线程可以抢占低优先级的线程并执行。DSP/BIOS 中按优先级从高到低有 4 类线程:硬件中断 HWI、软件中断 SWI、任务 TSK、后台线程 IDL,以及为周期性任务特别设计的硬件定时 CLK、软件定时 PRD,其中 CLK 为 HWI 级别线程,PRD 为 SWI 级别线程。

3.2 任务划分

对于 DSP/BIOS,任务就是上述 4 类 6 种线程。采用实时结构化分析方法,将工况调节过程划分为:

1) 硬件定时 CLK_Motor,回调函数名为 clk_Motor,周期为 2.5 ms,产生 400 Hz 时钟,驱动软件环形分配器,为调节器步进电机分配脉冲。

2) 软件中断 SWI_MotorRepos,回调函数名为 swiMotorRepos,当且仅当两台调节器电机均完成自动回额定操作时自动触发,完成工况调节过程初始化操作。

3) 软件定时 PRD_Power, 回调函数名为 prdPower, 周期为 500 ms, 按要求的工况调节速率解算调节器步进电机电作脉冲数并提交。

4) 任务 TSK_MotorReposO, 回调函数名为 tskMotorReposO, 执行助燃剂调节器自动回额定; 任务 TSK_MotorReposF, 回调函数名为 tskMotorReposF, 执行燃料调节器自动回额定。

另外还包括相关的实时通讯线程 (任务), 本文略。

各任务之间的关系参见图 1。

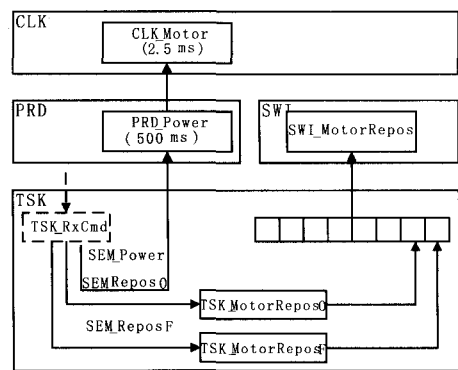


图 1 工况调节任务层次图

Fig. 1 Task hierarchy of power regulation process

表 1 二进制信号量				
Tab. 1 Binary semaphores				
序号	信号量名称	发送线程	接收线程	用途
1	SEM_Power	TSK_RxCmd	PRD_Power	发送工况调节命令
2	SEM_ReposO	TSK_RxCmd	TSK_MotorReposO	助燃剂调节器回额定
3	SEM_ReposF	TSK_RxCmd	TSK_MotorReposF	燃料调节器回额定

表中 TSK_RxCmd 为系统控制计算机遥控命令处理任务。

任务之间的通讯和行为同步采用二进制信号量结合共享内存实现。二进制信号量参见表 1。

3.3 任务结构图

调节器步进电机脉冲分配过程见图 2。

调节器回额定过程见图 3。

由工况解算调节器步进电机电作脉冲数过程见图 4。

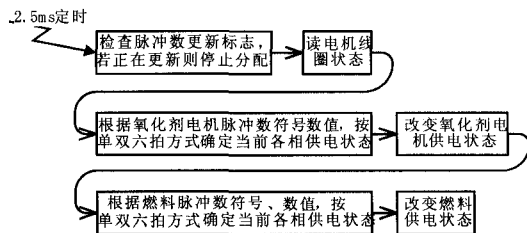


图 2 调节器步进电机脉冲分配

Fig. 2 Pulse distribution process of regulator stepping motor

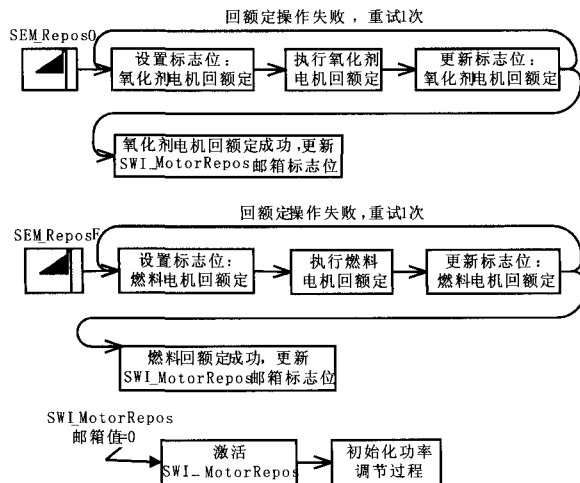


图 3 调节器回额定

Fig. 3 Process to restore regulators to rated position

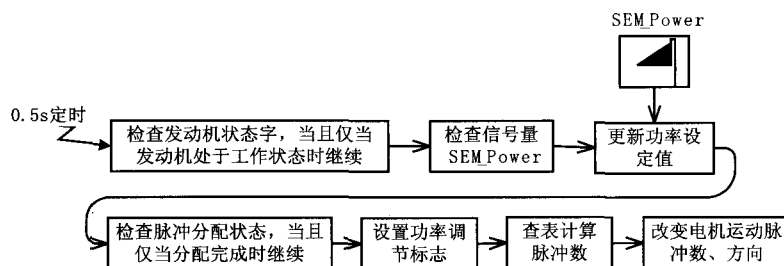


图 4 工况解算与控制

Fig. 4 Process of power calculation and adjustment

3.4 关键算法设计

3.4.1 脉冲分配

调节器步进电机工作在单双六拍方式, 脉冲相序为 A-AB-B-BC-C-CA-A 时所发脉冲定义为负脉冲, 电机负向运动, 调节器阀杆伸出, 阀门关小; 反之, 脉冲相序为 A-AC-C-CB-B-BA-A 时为正脉冲, 电机正向运动, 调节器阀杆缩回, 阀门开大。

采用查表方式确定当前电机相位, 相序为:

```
static const UINT16 sc_wPhaseTbl
```

```
[MOTOR_PHASE_MAX] =
```

```
{
```

```
    1, 5, 4, 6, 2, 3 /* Positive coils sequence: A,  
    AC, C, CB, B, BA */
```

```
};
```

助燃剂调节器电机脉冲分配程序片段如下:

```
// Distribute pulse: motor of oxidant regulator
```

```
if (s_wOStepCnt < s_wOStepSet)
```

```
{
```

```
    /* Set s_wMotorsStat bit 0, atomically */
```

```
    ATM_oru (&s_wMotorsStat,
```

```
MOTOR_STAT_BUSY_0);
```

```
    /* Get phase/pulse for this turn*/
```

```
    if (MOTOR_DIR_INC == s_iODir)
```

```
    {
```

```
        /* Check the " 10" contact, stay there if
```

```
reached, moves on if not */
```

```
        if (REGPOS_REACHED_NOT ==
```

```
REGPOS_O10)
```

```
{
```

```
    s_iOPhase++; /* Next phase */
```

```
    if (s_iOPhase >= MOTOR_PHASE_MAX)
```

```
    {
```

```
        s_iOPhase = 0;
```

```
    }
```

```
}
```

```
}
```

```
else if (MOTOR_DIR_DEC == s_iODir)
```

```
{
```

```
    /* Check the " 0" contact, stay there if  
reached, moves on if not */
```

```
    if (REGPOS_REACHED_NOT ==  
REGPOS_00)
```

```
    {
```

```
        s_iOPhase--; /* Next phase */
```

```
        if (s_iOPhase < 0)
```

```
        {
```

```
            s_iOPhase = MOTOR_PHASE_MAX -
```

```
1;
```

```
        }
```

```
    }
```

```
}
```

```
/* Update pin states of each coil */
```

```
PORT_MOTOR_O = sc_wPhaseTbl
```

```
[s_iOPhase];
```

```
/* Counts for next step
```

```
*/
```

```
s_wOStepCnt++;
```

```

}
else
{
    /* Clear s_wMotorsStat bit 0, atomically */
    ATM_andu (&s_wMotorsStat, ~
MOTOR_STAT_BUSY_O) ;
}

3.4.2 调节器回额定

    在调节器内部有一个绝缘棒，其外表面按一定间隔涂敷编号为 0~10 的 11 个金属导电带，各导电带之间为绝缘带。调节器阀杆运动带动触片，使触片导通/绝缘状态改变，检测触片状态变化，即可得知调节器阀门开度。在发动机起动前，必须使调节器调整到额定工况位置并锁定，方法是以导电带 5 靠近导电带 4 一侧为基准，向调节器发送一定数量脉冲，使触片运动到指定位置，此过程称为调节器回额定。

    为准确设定额定工况，从而确保工况调节精度，必须准确无误地完成回额定操作，为此必须准确地将触片定位于基准位置。考虑到机械回差，首先以较大步长使触片移至导电带 0；继而以较小步长运动并检测基准位置。为排除机械毛刺影响，检测导电带 5 时采用了数字滤波技术。另外，还需考虑调节器卡滞、触片失效、检测信号失效等因素，同时尽可能缩短回额定时间，并支持 2 台调节器同时回额定。

    以助燃剂路为例，调节器回额定程序片段到如下：
/* Sailing, for rated position, directly and
determinately
* as long as mamimum pulses exhausted or
reached successfully
*/
do
{
    /* Move by preset steps */
    MotorMoveBySteps (MOTOR_TYPE_O, iSteps) ;

    /* Wait for the movement to finish */
    TSK_sleep (iStepsABS) ;

    wPulse += iStepsABS;

    /* Creditable to read regulator contacts: oxidant */
    bStat = REGPOS_O;

    /* Deal with states to capture contact 1/0 first,
then contact 5 */
    if (REPOS_PHASE_1 == iReposStat)
    {
        /* To approach contact 1/0 */
        /* Sailing, for contact 1/0 */
        if (bStat == bLastStat)
        {
            if ((REGPOS_CONTACT1 == bStat) ||
/* Counts if reached */
(REGPOS_CONTACT0 == bStat))
/* Counts for 0 too */
            {
                iCapCnt++;
            }
            else /* Reset capture times counter */
            {
                iCapCnt = 0;
            }
        }
        else /* Reset capture times counter */
        {
            iCapCnt = 0;
        }
        bLastStat = bStat;

        /* State transfers if captured or pulses ex-
hausted */
        if (((SAIL_CONTACT1_CNT == iCapCnt) ||
(wPulse > MOTOR_STEPS_MAX))
        {
            iReposStat = REPOS_PHASE_5;
            iSteps = SAIL_P_STEPS;
            iStepsABS = iSteps;
            iCapCnt = 0;
        }
    }
}

```

```

        wPulse = 0;
    }
}
else if (REPOS_PHASE_5 == iReposStat)
{
    /* To approach contact 5 */
    /* Sail on, now for contact 5, directly and de-
terminately */
    if (REGPOS_CONTACT5 == bStat) /*
Counts if reached */
    {
        iCapCnt++;
    }
    else /* Reset capture times counter */
    {
        iCapCnt = 0;
    }

    /* State transfers if captured or pulses ex-
hausted */
    if (SAIL_CONTACT5_CNT == iCapCnt)
    { /* Captured successfully */
        /* Set regulator to rated position: oxidant
*/

        iSteps = gwNnO - iCapCnt * iStepsABS +
1;

        MotorMoveBySteps (MOTOR_TYPE_O, iS-
teps) ;

        /* Wait for the movement finishes
        if (iSteps < 0)
        {
            iSteps = -iSteps;
        }
        TSK_sleep (iSteps) ;

        /* Set flag: finish normally */
        wResult = REPOS_STAT_PASS;
    }
    else if (wPulse > MOTOR_STEPS_MAX)
    { /* Pulses exhausted but still not captured

```

```

yet, timeout */
        wResult = REPOS_STAT_FAIL;
    }
}
} while (REPOS_STAT_NOW == wResult) ;

```

3.4.3 工况解算

理论计算和试验表明, 发动机工况与两个调节器阀门开度中间存在对应关系, 表现为工况与改变两个调节器当前位置所需脉冲数的对照表, 采用查表法确定指定工况对应的两个脉冲数并提交, 即可实现工况调节。

在发动机工作过程中, 由于脉冲分配及调节器电机动作速度较快, 而发动机工况重新调整并稳定过程较慢, 为此需要对工况调节速率进行限制; 为确保发动机安全可靠工作, 必须确保两台调节器同时、同步动作, 以维持混合比不变, 因而必须确保脉冲数同时分配。为此引入 PRD_Power, 设定工况调节周期为 0.5 s, 在每个调节周期, 工况在当前基础上向要求的调节方向递增或递减 100 W, 亦即在“工况-脉冲数”对照表中前进或后退一行, 确认脉冲数后再同时提交。工况解算程序略。

4 测试与验证

在单元测试、组装测试、确认测试及系统联试各阶段进行了大量的测试, 对工况调节软件进行了全面、充分的验证, 相关的步进电机控制、调节器自动回额定及工况解算等关键功能的测试简单介绍如下。

4.1 步进电机控制

4.1.1 检查控制信号

在步进电机控制信号生成模块中插入桩函数, 将信号经指定的 GPIO 引脚输出, 采用多通道示波器检查方波的上升沿、下降沿、占空比、周期等参数; 持续观察一段时间, 确认波形稳定, 无跳变、畸变。

4.1.2 验证脉冲分配算法

对于每一路调节器, 将步进电机各相控制脉冲信号连接到逻辑分析仪输入端, 捕获分配脉冲

序列, 检查脉冲分配波形和数值应符合步进电机三相六拍分配率, 即各项相序按以下次序反复出现:

正方向: $A \rightarrow AC \rightarrow C \rightarrow CB \rightarrow B \rightarrow BA \rightarrow A \cdots$

反方向: $A \rightarrow AB \rightarrow B \rightarrow BC \rightarrow C \rightarrow CA \rightarrow A \cdots$

4.1.3 确认步进电机旋转方向及角度控制功能

在外部监控计算机中, 采用 LabWindows/CVI 编写测试软件, 经 RS-422 通讯接口发送设定调节器电机脉冲数指令; 连接带百分表的调节器、电机相位指示器, 检查设定值与实测值应一致, 电机动作过程应平顺、无噪音。

4.2 调节器回额定

在外部监控计算机中, 采用 LabWindows/CVI 编写测试软件, 经 RS-422 通讯接口发送调节器回额定指令, 并显示调节器内部位移传感器触点状态; 连接带百分表的调节器、电机相位指示器, 检查调节器回额定过程、回额定精度、电机初始相位和最终相位。

4.3 工况解算

包括查表测试、调节速率测试、边界测试、可靠性测试。

编写驱动模块产生任意工况值, 经脉冲数解算模块处理后, 产生的脉冲数被桩模块接收, 并与理论值对比。工况值的选取应至少包括有效等价类值、无效等价类值和边界数据值; 对于任意正常工况值, 应能获得准确对应的一组脉冲值; 对于无效工况值, 应妥善处理, 不会产生软件执行异常或脉冲值异常。

5 结论

工况调节技术是发动机控制软件的核心, 也是涉及面广、难度大的综合性技术。采用 DSP/

BIOS, 基于实时多任务架构, 将复杂的并发处理转换为一组相互协作的实时任务, 使得软件结构清晰、性能优良、运行稳定可靠, 顺利实现了发动机工况的实时、可靠调节, 在发动机的自动控制及工况调节技术方面取得了突破。

参考文献:

- [1] 刘宝廷, 程树康. 步进电机及其驱动控制系统[M]. 哈尔滨: 哈尔滨工业大学出版社, 1997.
- [2] 程翔宇, 李迪, 冯寿廷, 等. 基于 DSP/BIOS 的运动控制系统设计[J]. 机床与液压, 2007, 35(5): 157-160.
- [3] 张文山, 刘增明. 基于 DSP/BIOS 的空空导弹飞控软件仿真[J]. 航空兵器, 2007 (3): 57-60.
- [4] 丁宜栋, 华芸. 基于 DSP/BIOS 的软件系统设计[J]. 舰船电子工程, 2001 (5): 53-56.
- [5] 李进. 实时操作系统 DSP/BIOS 在 DSP 开发中的应用[J]. 微电子技术, 2003, 31(4): 49-52.
- [6] 张锡勇, 王丛岭. 基于单片机和 CPLD 的步进电机细分驱动系统[J]. 现代电子技术, 2006 (9): 63-66.
- [7] Auslander D M. Design and implementation of real time software for control of mechanical systems[M/OL]. [2006-09-26]. [http:// www.netyi.net/...3d-46c7-9523-b4284ed20bd3.htm](http://www.netyi.net/...3d-46c7-9523-b4284ed20bd3.htm).
- [8] Texas Instruments. TMS320x281x DSP system control and interrupts reference guide [M]. USA: Texas Instruments, 2008.
- [9] Texas Instruments. TMS320 DSP/BIOS user's guide[M]. USA: Texas Instruments, 2004.
- [10] Texas Instruments. TMS320 using the DSPBIOS kernel in real-time DSP applications[M]. USA: Texas Instruments, 2001.
- [11] 黄丽莲, 李子绪. 高性能中频采样系统的设计与实现[J]. 电子设计工程, 2010, 18(12): 1-4.
- [12] 雷能芳. 基于 DSP Builder 数字信号处理器的 FPGA 设计[J]. 电子设计工程, 2010, 18(11): 128-130.

(编辑: 马 杰)