采用有限状态机实现控制指令的可靠检测

施先旺[1]，刘婷婷[1]，李国良[2]
(1. 西安航天动力研究所，陕西 西安 710100；2. 中国航天科技集团软件评测中心，北京 100037)

摘 要：采用有限状态机设计了一种控制指令检测方法，确保发动机工作状态的可靠控制。介绍了控制指令特征及检测要求，提出了基于有限状态机的检测方法，说明了软件设计，并给出了测试、验证方法及结果。有限状态机模型有助于解决复杂的实时性问题。

关键词：有限状态机；时序控制；发动机控制系统；软件可靠性
中图分类号：V434-34 **文献标识码**：A **文章编号**：1672-9374 (2011) 05-0063-06

Control instruction detection realized by finite state machine

SHI Xian-wang[1], LIU Ting-ting[1], LI Guo-liang[2]
(1. Xi'an Aerospace Propulsion Institute, Xi'an 710100, China; 2. Software Test and Evaluation Center, China Aerospace Science and Technology Corporation, Beijing 100037, China)

Abstract: The finite state machine is adopted to design a method for control instruction detection, which can guarantee the reliable control of the engine. The characteristics of control instructions, detection requirements and detection method based on finite state machine are introduced. The software implementation is explained. The testing, verification methods and final results are provided. The application of finite state machine is of benefit to the solution of complicated problems in real-time domain.

Keywords: finite state machine; time sequence control; engine control system; software reliability

0 引言

在某发动机控制系统中，根据系统主控计算机发出的控制指令，使发动机执行相应的增压、起动、关机及紧急关机等程序，控制指令的可靠检测是实现发动机工作状态控制必须首先要解决的关键问题。根据控制指令特征，采用有限状态机设计了一种检测方法，大大提高了检测的可靠性。

收稿日期：2010-10-21；**修回日期**：2010-12-28
基金项目：中国航天科技集团公司支撑项目
作者简介：施先旺（1968—），男，研究员，研究领域为火箭发动机检测与控制、嵌入式系统等

1 指令特征与检测要求

控制指令包括增压、起动、关机、紧急关机、燃料排放及备用指令等，共 7 种，共用一个 16 位数字量端口。控制指令由系统主控计算机发出，指令信号驱动电气系统指令继电器，使相应继电器触点断开或闭合。在发动机控制系统中，继电器触点状态改变使指令检测电路输出逻辑电平变化，经由控制指令接口输入。控制系统软件实时检测控制指令接口状态，即时识别控制指令，并启动相应的控制时序。

控制指令信号特征如下：

1) 指令采用逻辑电平表示，高电平有效，低电平无效；

2) 指令信号高电平持续时间≮5 s；

3) 任意时刻有且仅有一个有效指令。

发动机工作过程中，通过执行上述指令，严格按照设定的通电顺序、通电时间，使给定的电爆阀动作，完成增压、系统充填、通电点火、起动、燃烧室工作、切断燃料供应使燃烧室停止工作等设定的程序。系统中电爆阀只能动作一次，且任何一个指令的工作时序、执行次序也必须确保一次成功。因此，必须确保控制指令准确、可靠、实时识别，既不可漏检，亦不能误检。在控制指令产生、传输、检测、执行各环节中，执行环节不可逆转。为提高任务可靠性，应能识别错误指令、干扰产生的伪指令、继电器触点失效产生的假指令等，防止误操作。

2 指令检测方法

2.1 技术途径

在计算机系统中，一般数字量的检测只需直接读取数字量端口即可。为提高可靠性，可以采用重复读取的方法，或进一步采用数字滤波算法。但对于本文涉及的系统，仅仅采取这些措施是远远不够的。

为提高指令检测的可靠性，必须充分利用指令的各种特征，包括上述信号特征以及隐含的任务特征，即：

1) 必须利用信号的宽度，多次、等间隔检测；

2) 同时检测所有信号，根据“任意时刻有且仅有一个有效指令”，丢弃无效信号；

3) 检测信号上升沿，最大限度地确认信号是主动产生的；

4) 指令只能执行一次，指令执行后应对此指令予以屏蔽；

5) 必须考虑指令的生存周期，且任意时刻只能有一个指令在执行。

根据上述原则，对数字量的检测不再是一般的静态读取过程，而是一种多信号按时间序列读取、比较、判断的动态过程。在这一过程中，指令识别过程事实上被划分为几种特定状态，每一种状态不仅与数字量端口状态有关，也与前一状态有关。

2.2 有限状态机

根据以上分析，采用有限状态机为指令信号检测过程建模。有限状态机是一种概念上的机器，常见于数字时序电路设计。有限状态机由一组状态集、一个起始状态、一组输入符号集、一个映射输入符号和当前状态到下一状态的转换函数组成。当输入符号串时，有限状态机随即进入起始状态，在任意特定时刻，只能处于其中一种状态。状态的转换依赖于当前状态、输入符号（触发条件）和转换函数，转换时间理论上为零。为此，根据指令特征和检测要求，从指令发出到指令执行结束，定义空闲、等待、检测、有效、执行、超时、完成 7 个状态，各状态及其对应的输入符号集、状态转换函数列如下。

1) S0，空闲（FSA_DIC_IDLE）：起始状态，指令执行超时或指令已执行时转入。所有信号均无效时转入等待状态。

2) S1，等待（FSA_DIC_WAIT）：等待控制指令。多个信号有效或单一信号有效但对应指令已执行时保持，检测状态信号异常时转入。当且仅当出现一个未执行的有效信号时转入检测状态。

3) S2，检测（FSA_DIC_COUNT）：检测控

制指令。在 N 次采样期间某单一信号应保持有效。信号异常转入等待状态，重新检测；正常则依次设置指令字、发信号激活对应的发动机时序控制过程、转入有效状态。

4) S3，有效（FSA_DIC_CAPTURED）：指令就绪、等待执行。检测发动机时序控制过程发出的执行信号并累计等待时间，若等待时间超出设定值则转入超时状态，若信号检测到则转执行状态。

5) S4，执行（FSA_DIC_HOLD）：指令执行中。检测发动机时序控制过程发出的完成信号并累计等待时间，若等待时间超出设定值则转超时状态，若信号检测到则转完成状态。

6) S5，超时（FSA_DIC_TIMEOUT）：指令超时。直接转空闲状态。

7) S6，完成（FSA_DIC_DONE），指令完成。屏蔽已执行指令并转空闲状态。

基于有限状态机的指令检测过程参见图 1。

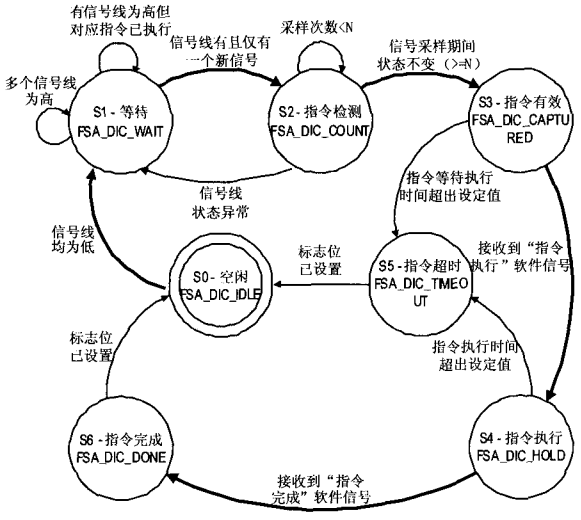

图 1 基于有限状态机的指令检测过程

Fig. 1 Detection process based on finite state machine

2.3 可靠性分析

2.3.1 正常指令检测

结合图 1，分析本方法是否能够充分利用 2.1 所列 5 个信号特征：

1) 在检测状态（S2），要求指令信号在 N 次采样期间保持有效，满足特征 1)；

2) 在等待状态（S1），当且仅当出现一个未执行的有效信号时才能转入检测状态，而在检测状态，采样期间状态变化即视为异常并转入等待状态，满足特征 2)；

3) 在空闲状态（S0，初始状态），当且仅当所有信号均无效时转入等待状态，启动指令检测状态机，且在等待状态只能由单一、未执行的有效信号触发转入检测状态，满足特征 3)；

4) 在完成状态（S6），屏蔽已执行指令，满足特征 4)；

5) 状态覆盖了指令发出到执行结束全过程，且一次只处理一个指令，满足特征 5)。

显然，对于正常指令，本方法可以可靠识别并提交执行。

2.3.2 异常信号处理

对于错误指令、干扰产生的伪指令、继电器触点失效产生的假指令等，本方法均能正确予以过滤，最大限度防止误操作。

指令执行期间（图 1 中 S3~S6）发出的指令、高电平持续时间小于采样时间的指令视为错误指令，结合图 1 分析可知，错误指令将被忽略。

对于指令信号线可能串入的干扰，由于必须符合 2.1 所列 5 个信号特征，故其被错误识别为正常指令的概率极低，可以忽略；同时，通过提高指令信号高电平持续时间（$\lesssim$5s），当某指令信号被干扰时，本方法可以在指令信号持续期间自动抛弃异常信号段，多次重新检测，直至有效识别，可能导致延迟执行，但极大地提高了执行成功的概率；在某信号被检测期间，若信号线出现异常，检测过程将自动重新启动，避免在存在干扰的条件下确认指令有效，也进一步提高了可靠性。

对于继电器触点，若在检测程序启动前即已失效，当失效造成某信号保持为低电平（无效）时该信号对应指令亦将失效，保持为高电平时将导致所有指令失效。若在指令检测过程中失效，视其发生在检测的不同阶段，将导致某指令或所有指令失效，在极端恶劣的情况下也可能不可避免地产生假指令，但这种情形出现的概率极低，可以忽略。

3 软件设计

控制软件是基于嵌入式实时操作系统设计的，采用 C 语言实现。为实现上述指令检测算法，定义了状态机、指令检测任务、指令执行任务、指令执行信号量等。

3.1 状态机

状态机定义如下：

```
typedef enum _FSAStatDiInsCap
{
FSA_DIC_IDLE,          /* S0: IDLE, waiting for signal cleared   */
   FSA_DIC_WAIT,       /* S1: WAIT, waiting for rising edge      */
   FSA_DIC_COUNT,      /* S2: COUNT, count to validate signal    */
   FSA_DIC_CAPTURED,   /* S3: CAPTURED, instruction verified     */
   FSA_DIC_HOLD,       /* S4: HOLD, FSA held during execution    */
   FSA_DIC_TIMEOUT,    /* S5: TIMEOUT, timeouts matched          */
   FSA_DIC_DONE        /* S6: DONE, process finished             */
} FSAStatDiInsCap。
```

3.2 任务及信号量

指令检测任务 tskEngineSeqCmd 为周期性任务，执行周期为 7.5 ms，优先级较高；指令执行任务为 tskEngineAct，优先级较低。指令检测任务结构如下：

```
    void tskEngineSeqCmd  (void)
{
    for  (;;)
    {
     TSK_sleep (DIC_PERIOD_TICKS) ;
……
    }
}  /* End of tskEngineSeqCmd ()  */
```

相关的信号量包括：

SEM_EngineSeqCmd：指令检测任务在判明指令有效时提交，由指令执行任务检测；

SEM_EngineAct：指令执行任务在指令执行前提交，指令检测任务在有效（FSA_DIC_CAPTURED）状态检测到后即转入执行（FSA_DIC_HOLD）状态；

SEM_SEM_EngineDone：指令执行任务在指令执行结束后提交，指令检测任务在执行（FSA_DIC_HOLD）状态检测到后即转入完成（FSA_DIC_DONE）状态。

3.3 指令检测过程

构造对应每个状态的 switch 语句，在每个分支中对相关状态进行判断，执行相应动作并完成状态转换。

```
switch  (s_FSADiInsCap)
{
    case FSA_DIC_IDLE:    /* S0: IDLE, waiting for signal cleared    */
    if  (DIC_LINE_NOTHING == s_wFSADiThis)

{  //No singal
          ……
          s_FSADiInsCap = FSA_DIC_WAIT;    // Shift to S1
}
    break;
   case FSA_DIC_WAIT:    /* S1: WAIT, waiting for rising edge       */
    if      (    (    DIC_LINE_BOOST          == s_wFSADiThis)  ||
                 (     DIC_LINE_IGNITE        == s_wFSADiThis)  ||
                 (     DIC_LINE_CUTOFF        == s_wFSADiThis)  ||
                 (     DIC_LINE_EXPELEMG      == s_wFSADiThis))
    {  //Certain one and only line set, starts
          ……
    s_FSADiInsCap = FSA_DIC_COUNT;    //Shift to S2
```

```
    }
    // ALL other line states, keep quiet!

  break;
  case FSA_DIC_COUNT:    /* S2: COUNT, count to validate signal    */
  if (s_wFSADiThis == s_wFSADiLast)
    { //Digital line remains the same, counts
      s_wFSAStatCnt++;
      if (DIC_COUNT == s_wFSAStatCnt)
       {
         ......
         s_FSADiInsCap = FSA_DIC_CAPTURED; //Shift to S3
          SEM_postBinary ( &SEM_EngineSe - qCmd) ; //Signal: command captured
       }
    }
  else //Digital line varied, reset FSA to detect again
    {
     s_FSADiInsCap = FSA_DIC_WAIT;    // Shift to S1
    }
  break;
case FSA_DIC_CAPTURED: /* S3: CAPTURED, instruction verified  */
  bStat = SEM_pendBinary ( &SEM_EngineAct, DIC_DECAYED_TICKS) ; //Wait
    if (bStat)
    { // Get signal, instruction being carried out
      ......
       s_FSADiInsCap = FSA_DIC_HOLD;    // Shift to S4
    }
  else //Instruction decayed, reset FSA
    {
      ......
       s_FSADiInsCap = FSA_DIC_TIMEOUT; //Shift to S5
    }
  break;
case FSA_DIC_HOLD:   /* S4: HOLD, FSA held during execution  */
  bStat = SEM_pendBinary (&SEM_EngineDone, DIC_TIMEOUT_TICKS) ; //Wait
  if (bStat)
    {  //Get signal, execution finished
      ......
       s_FSADiInsCap = FSA_DIC_DONE;    // Shift to S6
    }
    else //Execution timeout, shift to timeout phase
    {
    ......
s_FSADiInsCap = FSA_DIC_TIMEOUT; //Shift to S5
    }
  break;
  case FSA_DIC_TIMEOUT: /* S5: TIMEOUT, timeouts matched   */
  s_FSADiInsCap = FSA_DIC_IDLE;
  break;
  case FSA_DIC_DONE:   /* S6: DONE, process finished     */
  s_wFSADiMask & = ~s_wFSADiNow; //Set mask to disable THIS instruction
  s_FSADiInsCap = FSA_DIC_IDLE; //Shift to S0
  break;
  default:
  break;
}
```

4 测试与验证

为检验所提出的控制指令检测方法，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测试，方法得到了全面、充分的验证。

4.1 单元测试阶段

首先进行指令检测任务测试，以 tskEngine-SeqCmd 执行周期的测试为重点。在任务执行过程中插入数字量输出指令，并采用记录仪持续记录分析输出信号频率，应能观测到稳定的、周期为 2×7.5 ms 的方波。

其次进行状态机测试。对于每一个指令，构造指令信号，覆盖每一个状态，确认状态转换、状态处理过程无误。为了验证对异常信号的处理能力，在 FMEA 分析基础上，采用软件构造各种异常信号，通过指令模拟器发送执行，检查状态机对应的处理过程，将处理结果与预估结果比对确认。

4.2 组装测试阶段

激活指令执行任务 tskEngineAct，重复上述状态机测试过程。此阶段重点是确认指令检测过程和指令执行过程的协调性。

4.3 确认测试及系统联试阶段

对于每一个控制指令，监测指令执行过程和结果，分析控制系统软件发送的状态监测数据，反复确认指令检测、执行过程。

5 结束语

采用有限状态机设计的指令信号检测方法，能够可靠识别控制指令，且能有效屏蔽错误指令、干扰产生的伪指令、继电器触点失效产生的假指令等，为提高发动机工作状态控制的可靠性创造了条件。有限状态机模型有助于解决复杂的实时性问题，值得推广。

参考文献：

[1] (印度)卡莫尔 R. 陈曙晖等译. 嵌入式系统-体系结构、编程与设计[M]. 北京: 清华大学出版社, 2005.

[2] 王凌, 宋扬, 李国林, 等. 基于有限状态机的飞行器自毁系统时序控制设计[J]. 现代电子技术, 2009 (8): 8-11.

[3] 杨瑞霞. 运用状态机提高嵌入式软件效率[J]. 单片机与嵌入式系统应用. 2009 (5): 20-25.

[4] 李凌鹏, 孙文. 有限状态机在防空作战仿真中的应用[J]. 电光与控制, 2005 (5): 80-82.

[5] 周新蕾, 刘正高. 航天软件可靠性安全性技术应用发展趋势[J]. 质量与可靠性, 2006 (3): 45-47, 53.

[6] 杨军丽. 星载设备嵌入式软件可靠性仿真测试方法设计[D]. 北京: 中国科学院研究生院空间科学与应用研究中心, 2007.

[7] CHAPMAN M.The final word on the 8051 [M/OL]. [2007-06-24]. http://www.ebookee.net/The-Final-Word-on-the-8051.

[8] Texas Instruments. TMS320x281x DSP system control and interrupts reference guide [R]. USA: Texas Instruments, 2008.

[9] Texas Instruments. TMS320 DSP/BIOS user's guide [R]. USA: Texas Instruments, 2004.

(编辑：陈红霞)

(上接第 29 页)

[4] BOSSON R, GOIRAND B. Design of a high performance low cost hydrogen turbopump for VESCO engine, AIAA 99-2191 [R]. USA: AIAA, 1999.

[5] ALLIOT P, MARCHAL N, GOIRAND B, et al. The VINCI hydrogen turbopump development status, AIAA 2002-40 [R]. USA: AIAA, 2002.

[6] 马皓晨, 刘厚林, 关醒凡, 等. 双流道叶轮水力设计 CAD 中流道参数的确定[J]. 排灌机械, 2002 (2) :15-20.

[7] 梁开洪, 曹树良, 陈炎, 等. 入流角对圆截面 90°弯管内高雷诺数流动的影响[J]. 清华大学学报, 2009(12): 69-73.

[8] GOIRAND B, GALLARDO J F, BOSSON R. VINCI hydrogen turbopump: a new step in safe, faster and cheaper developments, AIAA 2006-3156 [R]. USA: AIAA, 2006.

[9] KAUPERT K A. Comparison of a vaneless and vaned cross-over configuration for a cryogenic send-out pump, AIAA 2004-5629 [R]. USA: AIAA, 2004.

(编辑：王建喜)