# 基于 UML 状态图和 Qt 状态机框架的 IEC104 规约的分析与实现

余 存，黄利军，黄浩然，申艳红，张 睿，崔晓优，贾帅峰

(许继电气直流输电系统公司，河南 许昌 461000)

**摘要**：为了提高软件的可维护性和可扩展性，降低规约开发的复杂度，基于统一建模语言(UML)的状态图理论对 IEC 60870-5-104 规约主站端进行了分析。抽象定义了动作、事件和状态，利用 UML 状态图分层和并发特性设计了规约逻辑状态图，并使用 Qt 的自动机框架进行了软件实现。结果表明，设计出的状态图层次结构分明，逻辑关系清晰，各功能模块间相互独立，而且降低了耦合度，编程实现简单明了，体现了这种方法的优越性，对其他复杂通信规约的开发也具有一定的参考价值。

**关键词**：IEC 60870-5-104；统一建模语言 UML；状态图；有限状态机；Qt

## Analysis and implementation of IEC 104 based on UML statechart and Qt state machine framework

YU Cun, HUANG Lijun, HUANG Haoran, SHEN Yanhong, ZHANG Rui, CUI Xiaoyou, JIA Shuaifeng
(XJ HVDC Electric Power Transmission Company, Xuchang 461000, China)

**Abstract:** In order to improve maintainability and expansibility of software, reduce the complexity of protocol development, based on the statechart theory of Unified Modeling Language (UML), this paper analyzes the master side of IEC 60870-5-104, abstracts and defines actions, events and states; designs the statecharts of protocol logic based on the hierarchy and parallel characteristics and implements the software using Qt state machine framework. The results show that the hierarchy structure of statecharts is distinct, the logic is clear, every function model is independent of each other, and the coupling is reduced; programming implementation using Qt state machine framework is easy, showing it is a good solution. This kind of analysis and implementation method has also good reference value to the development of other complex communication protocol.

**Key words:** IEC 60870-5-104; Unified Modeling Language; statechart; finite-state machine; Qt

中图分类号： TM76 文献标识码：A 文章编号： 1674-3415(2015)15-0118-08

## 0 引言

IEC 60870-5-104[1]规约是 IEC 60870-5-101 远动通信规约的以太网实现方式，与 IEC101 规约相比，增加了防止报文丢失及异步确认机制使数据传输更加可靠和高效，因此不仅在厂站与调度系统的通信中使用，也应用于变电站监控系统、智能辅助系统、EMS 系统和铁路系统装置通信中[2-6]。

IEC104 通信规约作为一种事件驱动的系统，如果根据流程图，使用 if-else 或 switch-case 语句来编程实现的方法，会引入大量的条件分支语句，这使得编写的程序需要花费大量时间进行调试，同时这样的程序可读性和可扩展性都比较差，不利于后期功能扩充和维护。

基于有限状态自动机的设计和实现方法(设计模式中称之为状态模式)通过把复杂的判断逻辑条件转移到表示不同状态的一系列类中，在事件驱动模块的设计实现中广泛应用，如文献[7-9]都使用了状态机模型进行分析设计。

有限状态自动机理论也被应用到了 IEC104 规约的设计和实现上[10]，这是一个很好的尝试，但效果不太理想，设计出的状态图可读性差、不易扩充。造成这些问题的原因是传统有限状态自动机缺乏层次和并发的支持，很难清晰地表示像 IEC104 规约

这种复杂系统。

Harel 状态图[11]通过对传统有限状态自动机进行扩展，克服了有限自动机的缺陷，扩展的内容包括层次、并发和通信[12]。UML 状态图[13]是 Harel 状态图面向对象的表示方式。

本文基于 UML 状态图对 IEC104 规约主站端进行分析，抽象定义了动作、事件和状态，设计了表达规约逻辑关系的状态图，并利用 Qt[14]状态机框架进行实现。

## 1　IEC104 规约的控制机制

与 IEC101 规约相比，IEC104 规约增加了使数据传输更加可靠和高效的各种机制，包括：

(1) 防止报文丢失和报文重复传送机制：规约定义了发送序列号和接收序列号，发送数据方在收到接收方的确认后才删除已发送数据。若一端判断出接收序号不连续，将主动断开链路重新建立连接，发送方会在连接重新建立后重新传送未被确认的数据。

(2) 异步确认机制：主站(客户)端接收到从站(服务器)端的变化信息后，并不立即进行确认，而是等定时器 t2 超时或者需要确认的数据帧达到一定数目 *n* 时，才发送数据确认报文。

(3) 测试机制：长期没有数据传输时(定时器 t3 超时)会启动测试过程。一方发起测试请求时，必须在规定时间内(*t*1时间内)收到对方的响应，否则会关闭链路，断开连接。

(4) 启/停传输控制机制：主站端利用启动数据传输(STARTDT)完成启动数据传输过程后，被控站才开始主动上送数据的传输。

正是由于这些机制的存在为编程实现增加了较高的复杂度。

## 2　状态图设计

UML 状态图是有限自动机的理论在计算机科学领域的实现。同时 UML 组织对其又进行了扩展，增加了层次和并发特性。状态图由状态、事件、动作及状态间的转换构成。在设计一个状态图之前，需要对问题领域有深刻的理解以便能准确抽象出状态、事件和动作。

### 2.1 事件及动作定义

设计状态图的第一步是对系统中可能出现的事件及需要执行的动作进行抽象和定义。

动作是一个可执行的操作，像改变一个变量的值，进行 I/O 操作或调用一个函数都是动作。在 IEC104 规约中，规约主站端(客户端)首先需要向从站端(服务器端)发起网络连接请求，“向从站端(服务器端)发起网络连接请求”就是一个动作，如表 1 动作定义中的 Act1；启动定时器也是一个动作，如表 1 中的动作 Act3。

事件在某个时间点发生，可以触发系统状态转换，比如定时器的超时，接收到一个数据包等。动作的执行也会产生事件，比如表 1 中的动作 Act1，其执行结果有两个，连接失败或者成功，“连接成功”和“连接失败”就是两个事件，如表 2 事件定义中的 Evt1 和 Evt2。

**表 1 动作定义**

Table 1 Definition of actions

| 动作名称 | 含义 |
|---|---|
| Act1 | 向服务器发送连接请求 |
| Act2 | 发送启动数据传输请求报文 |
| Act3 | 启动定时器 t1 |
| Act4 | 停止定时器 t1 |
| Act5 | 启动定时器 t2 |
| Act6 | 需要确认的帧计数加 1 并解析数据 |
| Act7 | 发送 S 格式的帧 |
| Act8 | 停止定时器 t2 |
| Act9 | 发送测试请求报文 |
| Act10 | 启动定时器 t3 |
| Act11 | 停止定时器 t3 |
| Act12 | 重启定时器 t3 |
| Act13 | 发送测试响应报文 |
| Act14 | 发送控制命令报文 |

**表 2 事件定义**

Table 2 Definition of events

| 事件名称 | 含义 |
|---|---|
| Evt1 | 连接成功 |
| Evt2 | 连接失败 |
| Evt3 | 成功发送启动数据传输请求报文 |
| Evt4 | 接收到启动数据传输响应报文 |
| Evt5 | 定时器 t1 超时 |
| Evt6 | 需要确认的 I 格式的帧达到最大数 *n* |
| Evt7 | 定时器 t2 超时 |
| Evt8 | 成功发送测试请求报文 |
| Evt9 | 定时器 t3 超时 |
| Evt10 | 收到测试响应报文 |
| Evt11 | 收到一帧正确帧序号的 I 格式帧 |
| Evt12 | 收到一帧错误帧序号的 I 格式帧 |
| Evt13 | I 格式帧处理完毕 |
| Evt14 | 成功发送 S 格式帧 |
| Evt15 | 收到测试请求报文 |
| Evt16 | 成功发送测试响应报文 |
| Evt17 | 接收到人机界面的控制命令 |
| Evt18 | 成功将命令发送给对端(装置) |
| Evt19 | 成功接收到正确的命令响应报文 |
| Evt20 | 接收到不匹配的命令响应报文 |
| Evt21 | 接收到命令响应报文超时 |

事件和动作的定义是对问题域深刻理解进而抽象出的。表 2 列出了 IEC104 规约中可能出现的事件。表 1 列出了所有的动作。在设计的最初阶段，可能并不能准确、完整地定义出所有的事件和动作，在状态图构建阶段，可以进一步添加事件和动作。

### 2.2 状态抽象与状态图构建

#### 2.2.1 状态图的表示方法

在 UML 状态图中状态用圆角矩形表示，状态的名称标识在矩形框内。初始状态用实心圆点表示。转换使用带箭头的线表示，箭头起点端为源状态，箭头末端为目标状态，转换的触发事件、动作及监护条件以“事件 e[监护条件 c]/[动作 a]”的形式在线上进行标注，含义是当事件 e 发生时，如果监护条件 c 为真，那么系统就会执行动作 a 并且系统状态转换到目标状态。其中转换中的监护条件和动作都是可选的。状态的入口动作用“entry/动作”表示，并标注在状态矩形框内；出口动作用“exit/动作”表示，并标注在状态矩形框内。

UML 状态图支持组合状态和并行状态。嵌套在另外一个状态中的状态称之为子状态，一个含有子状态的状态被称作组合状态。不包含任何子状态的状态称为原子状态。如果子状态比较复杂，需要构建单独的细化的状态图。组合状态中的多个并行状态间用虚线隔开。

#### 2.2.2 状态图构建

系统状态的构建过程是一个递归过程，包含三步：

(1)根据系统或模块的功能，将其划分成多个子系统或子模块，每个子系统或子模块抽象定义为一个状态，填写在状态定义表中。

(2)根据这些子系统或子模块之间的关系，定义事件、动作及初始状态，然后根据状态之间的转换关系设计该层次的状态图。

(3)对每一个子系统或子模块，重复步骤(1)和步骤(2)。

表 3 给出了状态图构建过程中抽象出的所有状态定义。

以下是 IEC104 规约主站端状态图的详细构建过程。

主站端程序启动后，首先尝试连接服务器(执行动作 Act1)，如果失败(事件 Evt2)则需要再次发起连接；如果成功(事件 Evt1)，则表示连接从站成功了。因此最高层的状态图可划分为连接成功状态(表 3 中的 B)和未连接状态(表 3 中的 A)，当组合状态 B 的子状态中出现定时器 t1 超时(事件 Evt5)或收到错误序号的 I 格式的帧(事件 Evt12)，系统将从状态 B 转到状态 A。结合动作和事件定义，图 1 给出了 IEC104 规约最高层的状态图。

**表 3 状态定义**

Table 3 Definition of states

| 名称 | 描述 |
|---|---|
| S | IEC104 规约状态总图 |
| A | 未连接状态 |
| B | 连接成功状态 |
| B1 | 连接成功，将要发送启动数据传输请求状态 |
| B2 | 等待启动数据传输响应状态 |
| F | 收到启动数据传输响应，可以进行数据传输状态 |
| F1 | I 格式数据帧处理 |
| F2 | 定时器 t2 超时处理 |
| F3 | 定时器 t3 超时处理 |
| F4 | 控制命令处理 |
| F5 | 测试请求处理 |
| F11 | 空闲状态，等待 I 格式数据帧 |
| F12 | 接收到正确 I 格式数据帧状态，并进行处理 |
| F13 | 需要确认的 I 格式帧达到 *N*，需要确认状态 |
| F21 | 等待定时器 t2 超时 |
| F22 | 定时器 t2 超时状态 |
| F31 | 等待定时器 t3 超时 |
| F32 | 定时器 t3 超时状态，发送测试请求报文 |
| F33 | 等待测试响应状态 |
| F41 | 等待人机界面的控制命令 |
| F42 | 处理来自人机界面的控制命令，发送给装置 |
| F43 | 等待命令响应 |
| F51 | 等待对端的测试请求 |
| F52 | 收到对端的测试请求状态 |

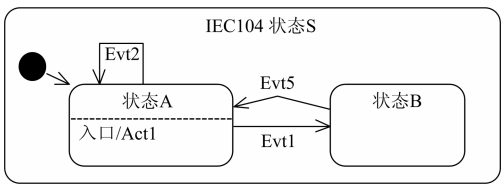


**图 1 IEC104 规约的顶层状态图**

Fig. 1 IEC104 protocol statechart of high-level part

当进入连接成功状态后，主站首先发送启动数据传输请求报文(执行动作 Act2)，然后等待启动数据传输响应报文。如果在 *t*1 时间内，收到启动数据传输响应报文(事件 Evt4)，就可以进行数据传输了；反之，如果没有收到响应报文，则事件 Evt5 会发生，该事件由状态 B2 的父状态 B 处理，这个转换在图 1 中表示。因此，连接成功状态可以划分为 B1、B2 和 F 三个子状态。B2 是等待启动数据传输响应报文状态，F 为数据传输状态。状态 F 是一个持久的状

态，正常情况下，系统将一直处于该状态，结合动作和事件定义，图 2 是状态 B 的状态图。

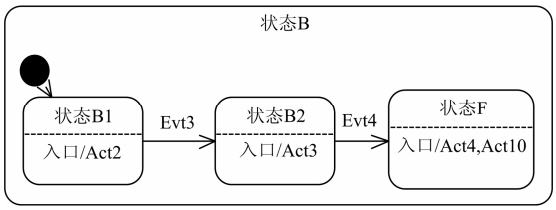


**图 2 连接成功状态 B 的状态图**

Fig. 2 Statechart of connect success state B

按照规约功能要求，状态 F 又可以划分为 5 个并行的状态，F1、F2、F3、F4 和 F5。F1 为 I 格式数据处理功能模块，F2 为定时器 t2 处理模块，F3 为定时器 t3 处理模块，F4 用于处理来自人机界面的控制命令，F5 用于处理来自对端的测试请求。结合动作和事件定义，图 3 显示了状态 F 的状态图。

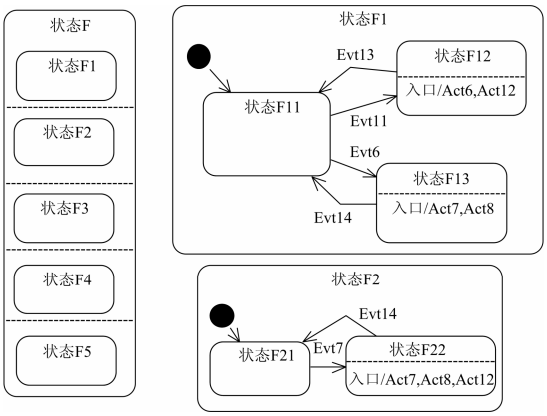


**图 3 状态 F、F1 和 F2 的状态图**

Fig. 3 Statecharts of state F, F1and F2

状态 F1 可划分为 3 个子状态，F11、F12 和 F13。F11 为空闲状态，等待 I 格式数据帧；F12 为 I 格式帧处理状态，在该状态下，对收到的 I 格式数据帧进行处理并重启定时器 t3(对应动作 Act6 和 Act12)；F13 为需要确认的 I 格式帧达到临界值 *n* 状态，需要发送 S 格式的数据确认帧并停止定时器 t2(动作 Act7 和 Act8)。F11 为 F1 的初始状态。结合动作和事件定义，状态 F1 的状态图在图 3 中显示。

状态 F2 包含 2 个子状态：F21 和 F22。F21 为等待定时器 t2 超时状态，F22 为定时器 t2 超时状态。F21 为 F2 的初始状态，在 F21 状态下，若定时器 t2 超时(事件 Evt7)，则系统进入 F22 状态。结合动作和事件定义，状态 F2 的状态图在图 3 中显示。

状态 F3 包含 3 个子状态：F31、F32 和 F33。F31 为等待定时器 t3 超时状态，F32 为定时器 t3 超时状态，F33 为等待测试响应状态。F31 为 F3 的初始状态，在 F31 状态下，若定时器 t3 超时(事件 Evt9)，则系统进入 F32 状态；系统进入 F32 状态后，发送测试请求报文、启动定时器 t1(动作 Act9 和 Act3)，并进入 F33 状态等待测试应答，若在 *t*1 时间内收到测试应答(事件 Evt10)，则系统转到状态 F31，在转换过程中停止定时器 t1 并重启定时器 t3；否则定时器 t1 超时(事件 Evt5)，导致 F31 的祖先状态 B 转到状态 A。结合动作和事件定义，状态 F3 的状态图在图 4 中显示。

状态 F4 包含 3 个子状态，F41、F42 和 F43。F41 为等待控制命令状态，F42 为处理控制命令状态，F43 为等待命令响应状态。F41 为 F4 的初始状态，在 F41 状态下，如果人机界面下发控制命令(事件 Evt17)，则系统进入 F42 状态，在该状态下将命令信息组织成 104 规约报文并发送给对端，然后进入 F43，等待响应。结合动作和事件定义，状态 F4 的状态图在图 4 中显示。

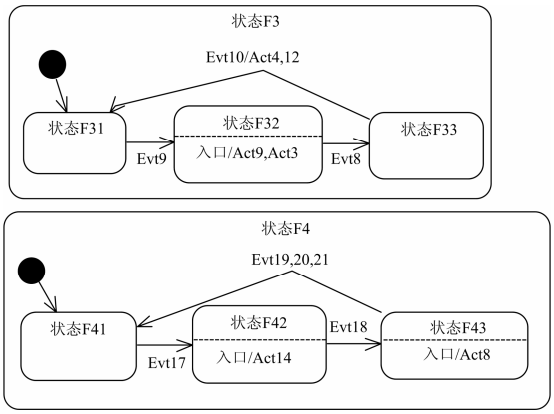


**图 4 状态 F3 和 F4 的状态图**

Fig. 4 Statecharts of state F3 and F4

状态 F5 包含 2 个子状态，F51 和 F52。F51 为等待测试请求状态，F52 为收到测试请求状态。F51 为 F5 的初始状态，在 F51 状态下，如果收到对端的测试请求(事件 Evt15)，则系统进入 F52 状态。在 F52 状态，发送测试应答报文，成功后，系统重新回到 F51 状态。结合动作和事件定义，图 5 是状态 F5 的状态图。

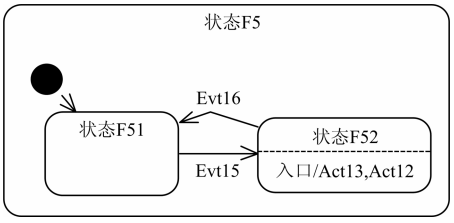


**图 5 状态 F5 状态图**

Fig. 5 Statechart of state F5

从图 1 到图 5 的状态图可以看出，分层设计的状态图层次结构分明，逻辑关系清晰；利用并行特性使各个功能模块相互独立，易于扩展新的功能。

## 3 基于 Qt 状态机框架的实现

Qt 最初是一个 C++的 GUI 开发组件[14]，现在已经发展成为一个可以开发在多种硬件平台和操作系统下运行的应用程序的框架，并在很多软件系统的开发中使用，如风电场监控系统的开发[15]。Qt 状态机框架提供了一个可以在 Qt 应用程序中嵌入状态图的元素和语义的 API 函数和执行模块。该框架与 Qt 的元对象系统紧密结合；例如，状态之间的转换可以由信号来触发等[16]。状态机框架由事件系统来驱动。

Qt 的状态机基于 Harel 的状态图，支持分层和并行，与 UML 状态图在语义上是等价的。因此在设计完状态图后，使用 Qt 状态机编程实现是比较简单的事情。

实现 IEC104 规约时，需要创建一个单独的线程用于不间断的接收报文。此外，三个类 ProtocolManager、ProtocolState 和 ProtocolApp 用于完成主要功能，规约管理类 ProtocolManager 用于规约报文的组织发送和报文解析，ProtocolState 是状态类的抽象，用于创建具体的类，ProtocolApp 类根据具体的状态图构建状态机。图 6 的 UML 类图显示了这三个类之间的关系。

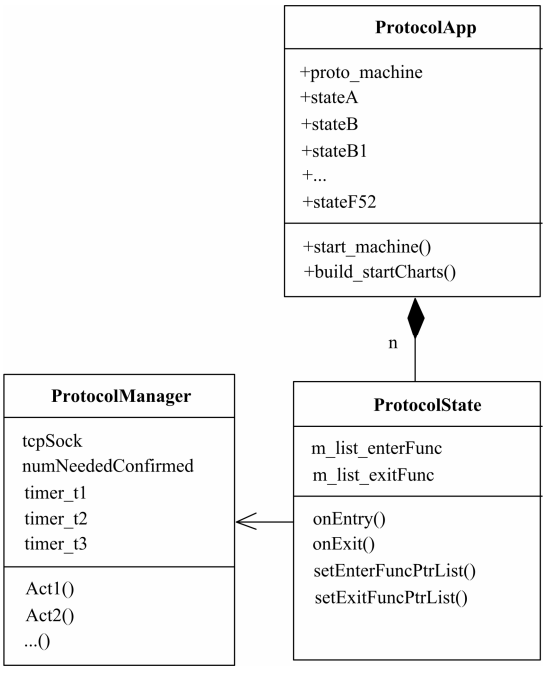


**图 6 规约实现的 UML 类图**

Fig. 6 UML class diagram of protocol implementation

编程时，表 1 中的事件转换为 Qt 的信号。类 ProtocolManager 继承自 Qobject，并包含了所有的这些信号及表 2 中的所有动作。

对于表 3 中定义的每一个状态实例，其差异之处在于在入口动作或出口动作，使用继承自 QState 的 ProtocolState 类来实现该功能。在实际实现时，由于单个状态可能有不止一个入口动作或出口动作，因此在 ProtocolState 类用 QList 类型的变量 m_list_enterFunc 保存单个状态的全部入口动作，QList 类型的变量 m_list_ exitFunc 用于保存单个状态的全部出口动作。ProtocolState 的 onEntry 需要重写，执行所有的入口动作，即对 list 列表中的每一个函数，都执行一次，核心代码为

```
for (int i = 0; i < m_list_enterFunc.size(); ++i)
  {
   ptr=m_list_enterFunc[i];
   if(ptr) (m_p_Link->*ptr)(event);
  }
```

ProtocolState 类的出口函数 onExit 实现与 onEntry 类似，不再列出。

ProtocolApp 类用于构建状态转换。ProtocolApp 类有一个有限状态机 QStateMachine 类型的变量 proto_machine，并包含表 1 中所有状态的实例。构建过程包括创建各个状态的实例，设置状态实例的入口动作和出口动作，设置组合状态的初始状态及调用 addTransition 创建状态间的转换。比如，对于状态 A 和状态 B，首先使用 stateA =new ProtocolState ();stateB=new ProtocolState ();创建状态 A 和 B 的实例，然后使用语句

```
QList<enterFunc>ptrList;
ptrList.push_back(&(ProtocolManager::Act1));
```

stateA-> setEnterFuncPtrList (ptrList);将状态 A 的入口函数 Act1(向服务器发送连接请求)放进状态 A 的实例 stateA 的 m_list_enterFunc 中。

在状态 A 下发生事件 Evt1 系统会转换到状态 B，因此使用语句 stateA->addTransition((const QObject*)( m_pLink), SIGNAL(Evt 1()), stateB);来完成转换定义。

由于状态 B 是组合状态，包含多个子状态，因此需要为其设置初始状态，根据图 2 可知，状态 B 的初始状态是 B1，那么使用语句 stateB-> setInitialState(stateB1);即可完成该设置。

对于包含多个并行子状态的组合状态 F，创建方法与其他不同，需要使用并行标记；语句 stateF= new ProtocolState (QState::ParallelStates, stateB)，用于状态 F 的创建工作。其他各个状态的具体实现与

上述的状态 A 和 B 类似，不再列出编程实现语句。

当所有状态的状态实例创建、状态间转换、入口动作和出口动作及复合状态的初始状态设置工作完成后，调用 proto_machine.addState(stateA); proto_machine.addState(stateB);将最顶层状态 A 和 B 加入到状态机变量 proto_ machine 中，并使用 proto_machin.setInitialState(stateA);语句设置状态 A 为初始化状态，完成这些工作后就可以调用 proto_machine.start();来启动状态机了。

相对于传统编程方法，基于 Qt 状态机的实现不仅消除了表示规约逻辑的大量条件分支语句，而且编程简单，大大节省了开发时间。

目前软件可以运行在 windows 7 及 Debian linux 操作系统上，所使用的 Qt 的版本号是 5.4.0。

## 4　功能测试

为了更有效地进行测试，我们用 Qt 开发了界面，主要包括两部分：第一部分为实际状态图的信息显示，该部分用于显示每一个状态是否活动(Active)，第二部分用于记录输出状态转换过程及收发报文。状态用矩形框表示，矩形框的线宽用于表示状态是否活动，最细的线宽用于表示从来没有活动的状态，当前活动的状态用最粗的线宽表示，曾经活动但当前不活动的用中等线宽表示。

功能测试分两类：一类是正常的功能测试，另一类为异常情况的测试。图 7 和图 8 显示了正常的功能测试结果。图 7 中，主站端成功连接从站端后，发送启动数据传输请求，收到响应后，状态 F 就成为当前的活动状态，因此状态 A、B1 和 B2 用中等宽度的线表示，状态 B、状态 F 均为最粗的线宽。状态 F 的并行子状态 F1、F2、F3、F4 和 F5 也均为最粗的线。F11、F21、F31、F41 和 F51 作为 F1、F2、F3、F4 和 F5 的初始状态也处于活动状态

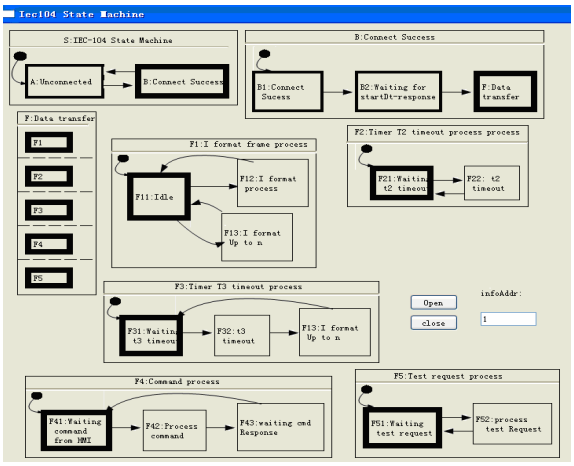


图 7　没有数据或命令的正常功能测试

Fig. 7 Normal test result without any data or command

(Active)，因此也是最粗的线。其他从未进入的状态都是最细的线。

图 8 中进行了控制命令测试，因此 F42、F43 曾经活动过，变为中等线宽；收到了 I 格式的帧也导致 F12 状态曾经活动过，也变为中等线宽；定时器 t3 超时导致测试请求报文发出，因此 F32、F33 为中等线宽；图 8 中定时器 t2 超时，导致 F22 状态变为活动，发送完 S 格式帧后又转到 F21 状态，因此 F22 状态为中等线宽。

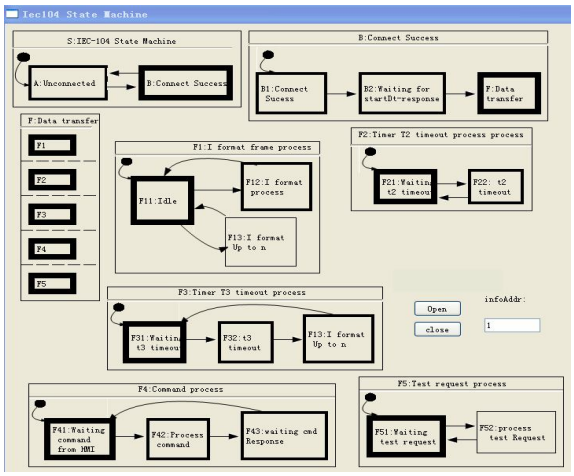


图 8　包含数据和命令的功能测试

Fig. 8 Test result including commands and I format frames

图 9 是一种异常测试情况，主站端成功连接从站端后，发送启动数据传输请求，然后处于等待启动数据传输响应状态。如果在规定事件内，没有收到响应报文，则系统断开连接，转到状态 A，重新连接从站。因此，状态 A 和 B1 为中等线宽，状态 B 和 B2 为最粗线宽，其他的状态为最细线宽。

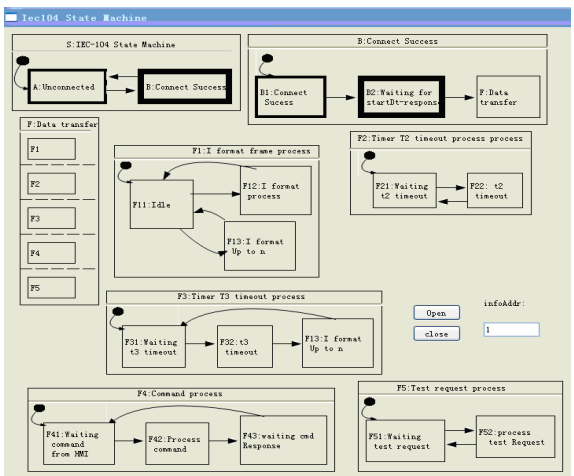


图 9　未收到启动数据确认的异常测试

Fig. 9 Abnormal test result without start-dt-response

图 10 为状态转换和报文收发记录，该记录显示的状态转换与图形化的显示是一致的。

测试结果表明本文的分析和实现方法满足 IEC104 规约的功能要求。

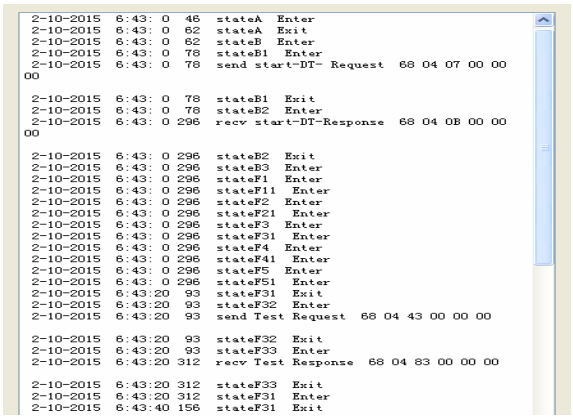


**图 10 状态转换过程和报文收发的记录**

Fig. 10 Test records with state transition and telegram

## 5 结语

本文基于 UML 状态图对 IEC104 规约主站端进行了分析，定义了状态、事件、动作及状态间的转换，并使用 Qt 状态机框架进行了实现。本文的分析、设计和实现方法具有以下优点：

(1) 基于分层和并行的特性设计的状态图层次结构清晰，逻辑关系明确；各功能模块相互独立，这些都为今后的维护和扩展提供了便利。

(2) 基于 Qt 状态机框架的实现不仅大大减少了编程的复杂度，把开发人员从程序编码中解放出来，专注于规约逻辑，而且可以使本软件仅需要微小修改就可以在不同的硬件平台和操作系统上运行，保证了软件的可移植性。

同时本文的分析和实现方法对其他复杂通信规约甚至其他事件驱动系统的开发也具有很好的参考价值。

## 参考文献

[1] 全国电力系统管理和信息交换标准化技术委员会. DL/T634.5104-2009 远动设备及系统第 5-104 部分: 传输规约采用标准传输协议子集的 IEC60870-5-101 网络访问[S]. 北京: 中国电力出版社, 2009.
National Standardization Technique Committee for Power System Management and Information Exchange. DL/T634.5104-2009 telecontrol equipment and systems 5-104: network access for iEC 60870-5-101 using standard transport profiles[S]. Beijing: China Electric Power Press, 2009.

[2] 梁竹靓, 韩兵, 彭永, 等. IEC60870-5-104 规约在分布式电力监控系统中的应用[J]. 电力系统保护与控制, 2011, 39(4): 124-127.
LIANG Zhuliang, HAN Bing, PENG Yong, et al. Application of IEC60870-5-104 protocol in distributed electric supervisory control system[J]. Power System Protection and Control, 2011, 39(4): 124-127.

[3] 韩小军, 蔡东升, 黄琦, 等. 基于 IEC104 远动规约的智能变电站辅助平台测试系统设计与实现[J]. 电测与仪表, 2014, 51(14): 104-109.
HAN Xiaojun, CAI Dongsheng, HUANG Qi, et al. Design and implementation of a test system for the smart substation auxiliary platform based on the IEC104 telecontrol protocol[J]. Electrical Measurement & Instrumentation, 2014, 51(14): 104-109.

[4] 陆圣芝, 薛钟兵. 基于 EMS 一体化的保护定值远方操作研究与应用[J]. 电力系统保护与控制, 2014, 42(20): 150-154.
LU Shengzhi, XUE Zhongbing. Research and application on relay protection setting value remote operation based on EMS integration[J]. Power System Protection and Control, 2014, 42(20): 150-154.

[5] 史超美, 王琳, 李强. 铁路 10 kV 自闭/贯通线故障定位装置通信接口的设计与实现[J]. 电网与清洁能源, 2014, 30(5): 18-21.
SHI Chaomei, WANG Lin, LI Qiang. Design and implementation of communication interface for fault location and isolation device for of location device for the 10 kV railway auto-blocked and run-through line[J]. Power System and Clean Energy, 2014, 30(5): 18-21.

[6] 万勇. 一种基于云计算技术的 SCADA 系统设计[J]. 高压电器, 2013, 49(7): 89-92.
WAN Yong. SCADA system design based on cloud computing technology[J]. High Voltage Apparatus, 2013, 49(7): 89-92.

[7] 李振兴, 尹项根, 张哲, 等. 广域保护多 Agent 系统动态协作机制[J]. 电力系统保护与控制[J]. 电力系统保护与控制, 2012, 40(3): 36-40.
LI Zhenxing, YIN Xianggen, ZHANG Zhe, et al. Dynamic cooperation mechanism of wide area protection system based on multi-Agent[J]. Power System Protection and Control, 2012, 40(3): 36-40.

[8] 韩烨, 刘志刚, 李文帆. 基于 LabVIEW 的新型电能质

量分析软件开发[J]. 电力系统保护与控制, 2012, 40(16): 121-125.
HAN Ye, LIU Zhigang, LI Wenfan. Developing a new power quality analysis software based on LabVIEW[J]. Power System Protection and Control, 2012, 40(16): 121-125.

[9] 李力, 李志平, 王亮, 等. 稳定控制装置中策略搜索匹配状态机模型[J]. 电力系统自动化, 2012, 36(17): 86-89
LI Li, LI Zhiping, WANG Liang, et al. A finite-state machine model for strategy table search and match of standardized stability control device[J]. Automation of Electric Power Systems, 2012, 36(17): 86-89.

[10] 傅钦翠, 陈剑云. 基于 FSM 的 IEC60870-5-104 规约的实现[J]. 继电器, 2008, 36(10): 45-48.
FU Qincui, CHEN Jianyun. Implementation of IEC60870-5-104 protocol based on FSM[J]. Relay, 2008, 36(10): 45-48.

[11] HAREL. Statecharts: a visual formalism for complex systems[J]. Science of Computer Programming, 1987, 8: 231-274.

[12] VIDANAPATHIRANA A C, DEWASURENDRA S D, ABEYRATNE S G, et al. Statechart based modeling and controller implementation of complex reactive systems[C] // Proceedings on Symposium of 6th IEEE International Conference on Industrial and Information Systems in Sri Lanka, Peradeniya , August 16-31, 2011: 493-498.

[13] OMG (February 2009). OMG unified modeling language (OMG UML)[M]. Superstructure Version 2.2".

[14] http://qt-project.org[EB/OL].

[15] 田英, 徐丽杰, 王玮, 等. 基于统一平台的风电场中央监控系统主站设计[J]. 电力系统及其自动化学报, 2014, 26(2): 55-59.
TIAN Ying, XU Lijie, WANG Wei, et al. Design of main station in windfarm center monitoring system based on unified platform[J]. Proceedings of the CSU-EPSA, 2014, 26(2): 55-59.

[16] 袁媛, 王玥, 黄少华, 等. 基于 Qt 的状态转换机制的研究[J]. 现代电子技术, 2013, 36(8): 121-124.
YUAN Yuan, WANG Yue, HUANG Shaohua, et al. Research of state transition mechanism based on Qt[J]. Modern Electronics Technique, 2013, 36(8): 121-124.

**收稿日期：2015-03-09；　修回日期：2015-04-29**

**作者简介：**

余　存(1983-)，男，硕士研究生，工程师，研究方向为高压直流输电SCADA系统；E-mail: yucun2006@126.com

黄利军(1969-)，男，本科，高级工程师，研究方向为电力系统及其自动化；E-mail: lijunhuang@xjgc.com

黄浩然(1976-)，男，本科，高级工程师，研究方向为HMI监控系统开发。E-mail: haoranh@xjgc.com

（编辑　魏小丽）