倍福控制系统的 CAN2.0 通讯解决方案

Version 1.0.1

倍福广州 陈利君 2018.01.08

Based on Version 1.0.1
改动：对 Output 变量 RxCounter 的含义解释。

1. 背景介绍

控制器局域网总线（CAN，Controller Area Network）是世界上应用最广泛的现场总线之一，CAN 接口可能是性价比最高的总线接口，通讯速度可达 1Mbps，百度搜索显示最便宜的 CAN 接口，含贴片式 CAN 微控制器和收发器，价格在 10 元以内。与此同时，CAN 总线的实现也比绝大多数现场总线要简单，因此深得研发人员的青睐。在可选多种总线接口的自动化仪表或者设备中，通常 CAN 接口是最便宜甚至是免费的选项，而某些经济型产品甚至只支持 CAN 总线，尤其是一些新研发的国产设备，或者厂家开发的自用设备。

CAN 通讯的最大特点是它工作于**多主**方式。CAN 网络中没有主站从站的说法，统一叫做 CAN 节点。CAN 节点也没有地址，每个节点都可以主动向总线发送数据。多节点同时发送数据时，根据报文标识符来竞争总线访问优先权。什么是报文标识符呢？请留意后文的解释。

CAN 通讯还有一个特点：所有节点总是同时收到相同数据，所以各节点之间数据通信实时性强，并且容易构成冗余结构，提高系统可靠性和系统灵活性。相比于 RS-485 在实时性、可靠性方面优势明显，因为后者只能构成主从式结构系统，通信方式也只能以主站轮询方式进行。

从发展历史来看，CAN 总线诞生于 1986 年，没有查到 CANopen 的诞生年份，百度搜索到最早的 CANopen 文档发布于 1996 年。而 Profibus 成为欧洲标准现场总线也是在这一年。我个人理解是，CAN 总线是针对 RS485 通讯的种种不足而推出的一种总线。其后的 CanOpen、Profibus、DeviceNet，则分别在 CAN 和 RS485 的基础上，区分出了主站和从站，毕竟在生产设备上，变频器和变频器之间不必通讯，而所有变频器都要和控制器通讯。用一个专门的硬件来实现与底层设备的数据通讯，而不必占用控制器的 CPU 来运行 CAN 或 RS485 通讯代码，这是现场总线分出主站和从站的根本目的。所以，CAN 相对于 RS485 是先进的，相对于 CANopen、Profibus 等总线，使用起来就不那么方便了，这也是我为什么要写这篇文章的原因。

对于今天的自动化工程师来说，已经熟悉了 Profibus、Devicenet、CanOpen 的使用，习惯了通过简单的参数配置，总线主站就能自行完成底层通讯，而不必深入了解这些通讯的底层协议。但在使用 CAN 通讯时，还必须得自行编写 PLC 上的 CAN 通讯代码，所以有必要对 CAN 通讯做些常识性的了解，才能理解 PLC 的 CAN 通讯代码应该怎么写，为什么要这样写。

2. 倍福控制系统实现 CAN2.0 通讯的硬件准备

2.1 CANopen 主站模块或者 BK51xx 耦合器

倍福控制系统要实现 CAN2.0 通讯，必须要有 CANopen 主站模块，即以下型号之一的硬件：

EL6751-0000：CANopen 网关 EL 模块，适用于任何带 EtherCAT 总线的倍福控制系统。

CXxxxx-M510：CX 控制器上扩展的 CANopen 主站，适用于 CX 嵌入式控制器。

FC51xx ： PCI、PCIe 或 Mini PCI 接口的 CANopen 主站卡，适用于带相应插槽的 IPC
BXxxxx ： 倍福早期的 BX 系列控制器上集成的 SSB 接口。

至于倍福产品作为 CAN 节点集成到第三方的控制系统，最典型的应用就是 BK51xx 耦合器的 CAN 通讯。由于 BK51xx 只是一个耦合器，不能编程，所以只能作为 CAN 节点在事件触发的方式下交换数据。关于这种应用最重要的一点是，要在初始化时修改对象字典 0x6423 的值为 1，否则模拟量信号无法通讯。详细文档见《第三方 CANopen 主站与 BK5120 通讯设置》，作者倍福长沙办 刘端健。

在用第三方 CAN 控制器 EPEC 与 BK5120 通讯时，在主站控制器依次发以下报文即可实现与耦合器后所有类型 IO 模块的数据通讯。

Cob-id	字节长度	报文	含义
000	2	82 00	Reset
601	8	2F 23 64 00 01 00 00 00	修改 0x6423 值为 1
000	2	01 01	start

以下内容都针对倍福做主控系统的应用。

2.2 CAN 网络接线

CANopen 主站卡连接其它 CAN 设备的接口形式有 9 针 D 形头和 5 针连接器两种：

或者

9 针 D 形头的针脚定义：

Pin	Assignment
2	CAN low (CAN-)
3	CAN ground (internally connected to pin 6)
5	Screen
6	CAN ground (internally connected to pin 3)
7	CAN high (CAN+)

针脚 3 和 6 是内部短接在一起的。

CAN 通讯需要接 120 欧的终端电阻。尽量使用标准的 CAN 电缆，在倍福的 CAN 电缆订货号为 ZB5200。临时测试时至少要使用 0.34mm2 的双绞屏蔽电缆。屏蔽层单端接地以避免环流。

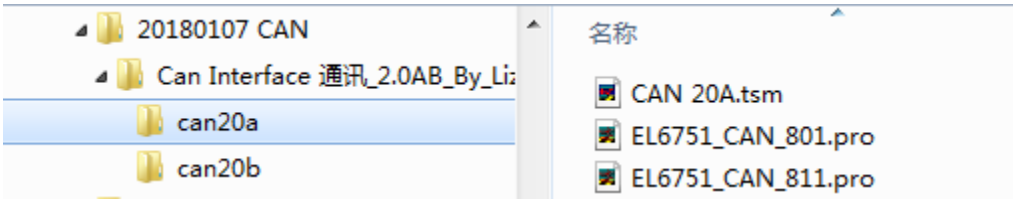
网络拓朴及主干、分支长度，直接影响网络支持的最高波特率，从 125kbps 到 1Mbps 不等。理论上有一个“线长/速度”对应表，实际应用中推荐直接使用默认值 250kbps，如果不能满足要求再根据实际情况选择更高或者更低的波特率。

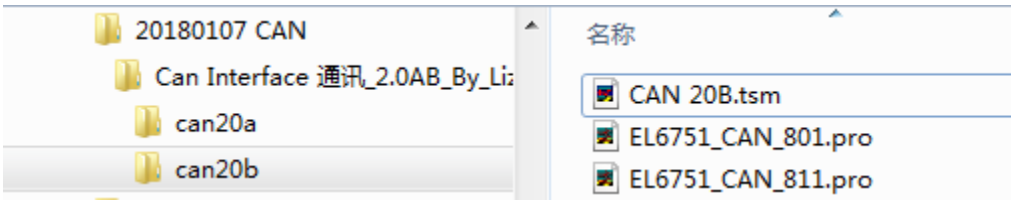
3. 测试归档文件：

归档文件应包含 PLC 程序、硬件配置，用两个 EL6751 来做 CAN 通讯，以说明使用方法。

Can Interface 通讯_2.0AB_By_Lizzy.zip

其中包括 CAN 2.0A 即（11 位标识符）和 CAN 2.0B（即 29 位标识符）两种代码。

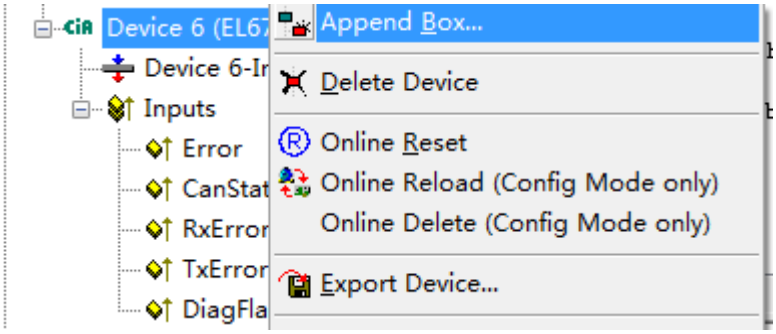


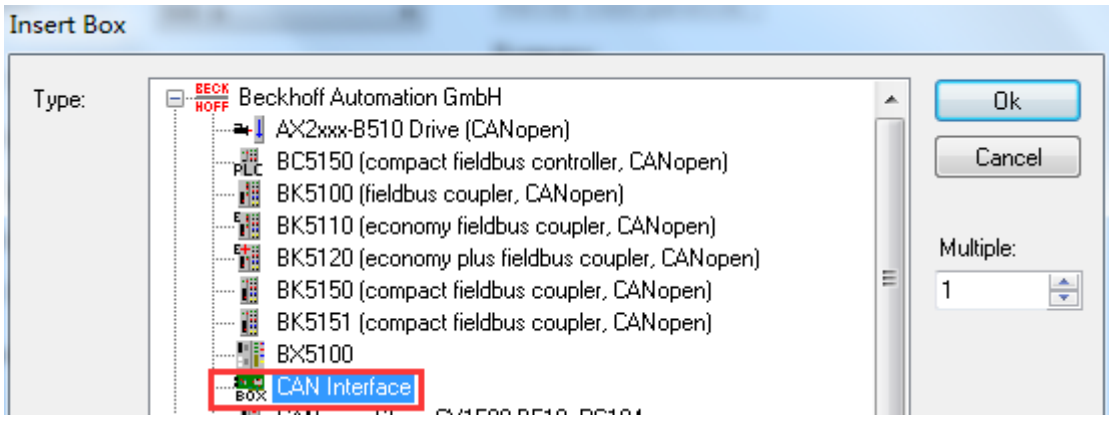


例中的两个 EL6751 分别由一套 PLC 程序控制，实际项目中参考任一程序均可。

4. 参数配置

4.1 扫描或者手动添加 CANopen 主站设备后，手动添加 CAN Interface。注意一个主站模块只能添加一个 CAN Interface。

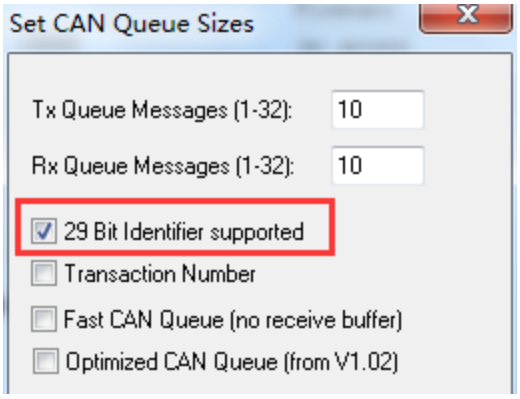




4.2 设置 CAN Interface 的传输队列

如果勾选“29 Bit Identifier supported”，则使用 CAN 2.0B，否则使用 11 位标识符的 CAN2.0A。

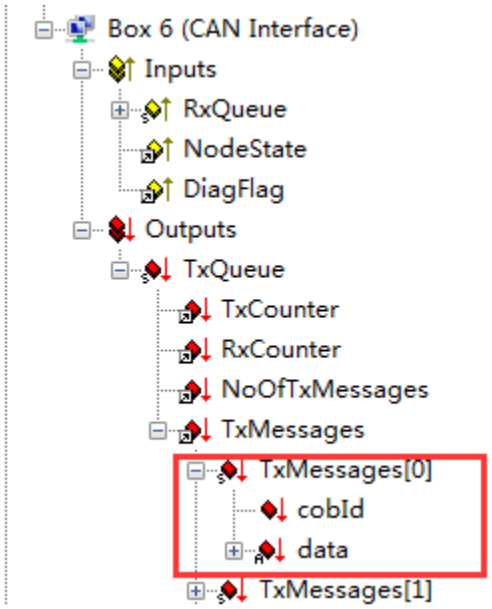
“Tx Queue Messages”和“Rx Queue Messages”直接使用默认值“10”，这表示一个控制周期最多缓存 10 条接收消息，而一个周期发送的消息数量也最多 10 条。



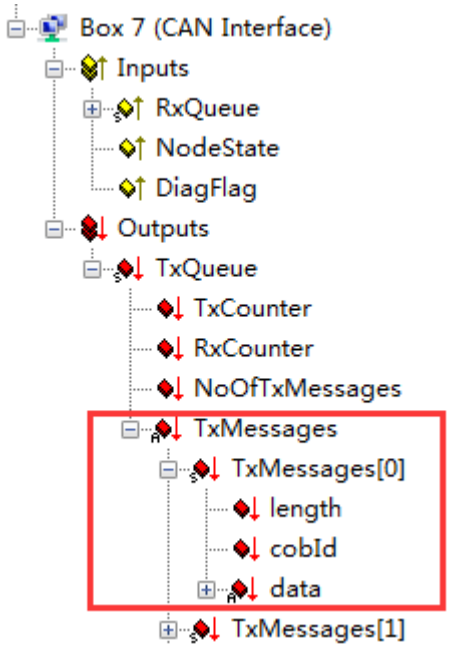
4.3 CAN Interface 接口消息（“Message”）的构成

如果使用默认的Message数量，成功添加CAN Interface后，发送和接收都会有10个Message，根据 CAN 协议的类型，在设备的 Process Data 中有以下内容：

CAN 2.0A 的接口变量



CAN 2.0B 的接口变量



发送和接收消息的数量：默认都为 10 个，在 TwinCAT 2 中，如果需要修改消息数量，必须删除 CAN Interface 再重新添加。

CAN2.0A 的消息和 CAN2.0B 的消息格式有所不同：

2.0A 的 Message 中，包含 cobid 和 data，其中 cobid 只有 2 字节 16 位，其 Bit 0-3 表示 Data 中实际发送的字节数，Bit 4 表示 RTR，而 Bit 5-15 表示 Message 的 cobid，所以 CAN 2.0A 的 Cobid 不会超过 16#7FF。

2.0B 的 message 中，包含 cobid、data 和 Length，其中 cobid 有 4 字节，其 Bit 31 为 False 表示 11 位标识符，为 True 表示 29 位标志符，Bit 30 表示 RTR。Bit0-28 表示 cobid，所以 CAN 2.0B 的 Cobid 最大值为 16# 3FFF FFFF。

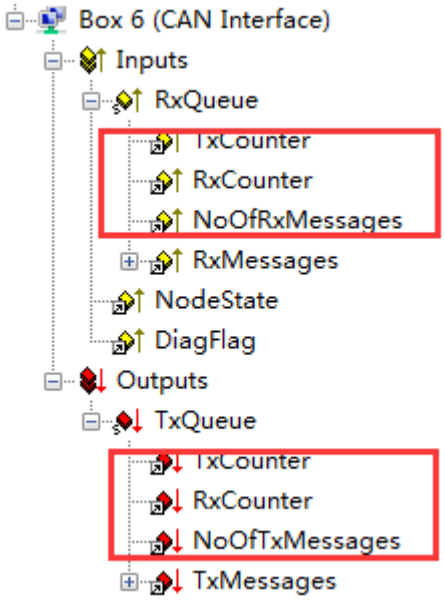
Data 中实际发送的字节数则由 Length 表示，虽然 CAN 2.0B 的 Length 是 uint 型，其值也不会大于 8。因为 Data 总是 8 字节的数组，表示一个 Message 能够交换的数据最多不超过 8 字节。为了

节省带宽，如果只交换一个 Int 整数型变量，就没必要把 8 个字节都发送出去。所以需要用 Length 或者 Cobid 的特定位来控制发送的字节长度。

消息（“Message”）是各个 CAN 节点之间通讯的基本单位。一个最小的通讯，就是在两个节点之间的 CAN 通讯，一个节点上配置一个 TxMessage，而另一个节点上配置一个 RxMessage。只要节点的 RxMessage 中收到的消息 Cobid 与发送方一致，就证明通讯成功了。

Can Interface 的 RxMessage 和 TxMessage 结构完全相同，所以 CAN 网络中要么全部用 CAN2.0 A，要么全部用 CAN 2.0B。

4.4 CAN Interface 接口中消息队列的状态的控制



在 RxQueue（接收队列）和 TxQueue（发送队列）中，都有同样的 3 个变量：

在 RxQueue 下的变量含义：

TxCounter：已发送计数，发送一次自动加 1，

RxCounter：已接收计数，接收一次自动加 1，

NoOfRxMessages：自上次接收以来重新收到的 Message 数量

在 TxQueue 下的变量含义：

TxCounter：发送计数，有变化则触发发送一次，

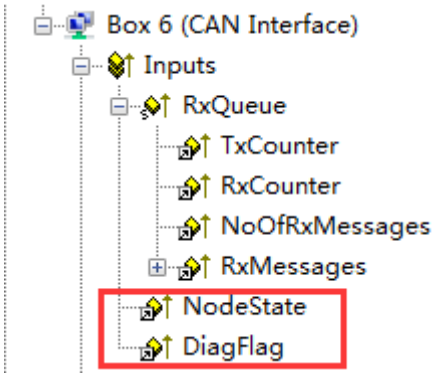
RxCounter：接收计数，当与 RxQueue 中的 RxCounter 相等时可触发重新接收；

NoOfTxMessages：本次计划发送的 Message 数量；

TxQueue 的 TxCounter 只要有变化就会触发硬件模块发送 Message，最简单的变化是自累加 1，发送的 Message 数据，就是 Message 数组的 10 个元素中的第 1 到 NoOfTxMessages 个。

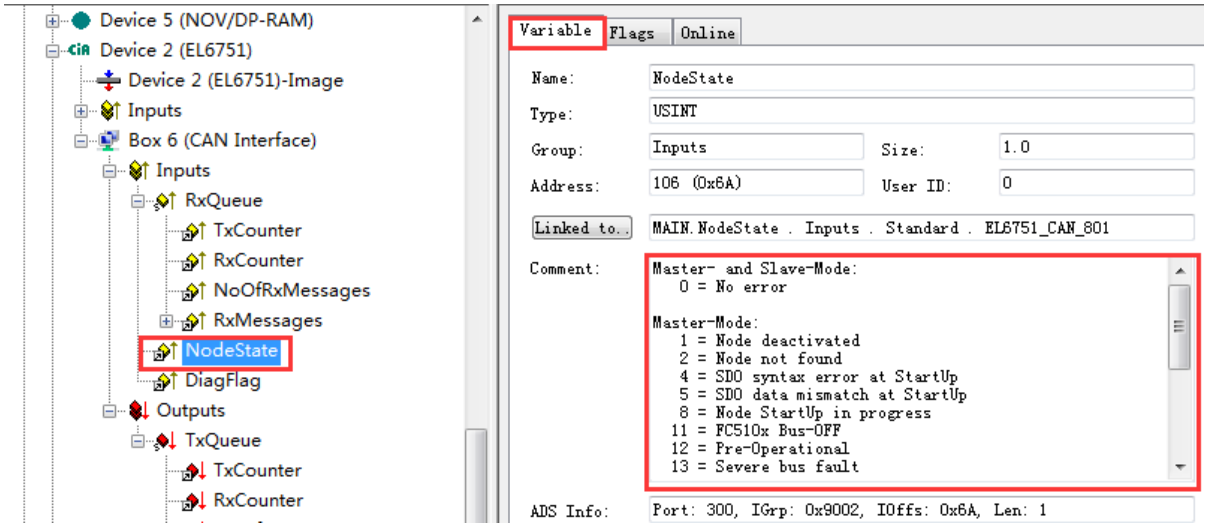
RxQueue 的 RxCounter 有增加时，表示有收到新的 Message，在 PLC 程序中把收到的 Message 处理完后，把 TxQueue 的 RxCounter 置为与 RxQueue 的 RxCounter 相等，则会触发接收硬件重新接收数据，即把 NoOfRxMessages 清零，重新收到的第 1 个 Message 放在数组的第 1 个元素，第 n 个 Message 放在数组的第 n 个元素，每收到一个 Message，就把 NoOfRxMessages 递增 1.直到下一次 TxQueue 的 RxCounter 与 RxQueue 的 RxCounter 相等。

4.4 CAN Interface 接口的诊断信息



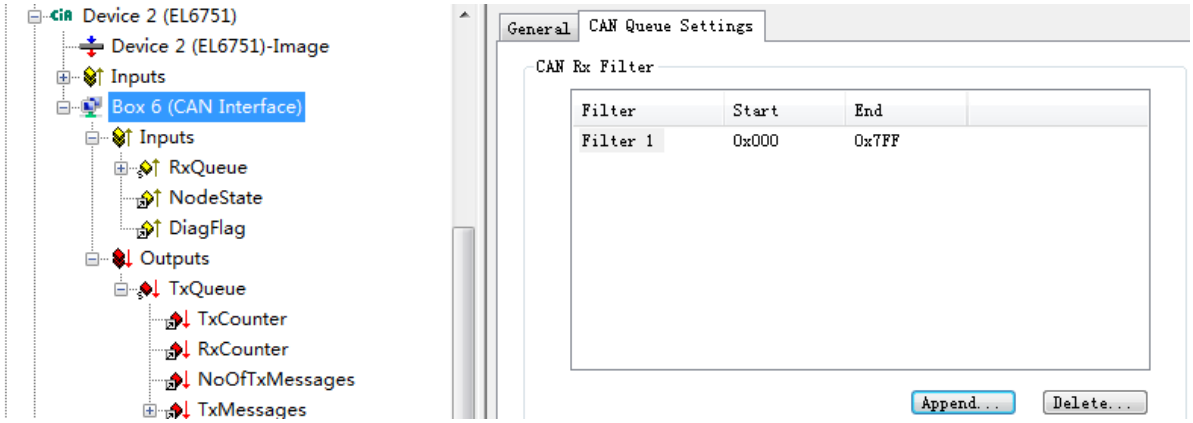
诊断信息有两个变量：Node State 和 DiagFlag

正常通讯时二者都应为 0，如果不为 0，根据变量的 Comment 可以查到 Node State 的值对应含义，但对于 CAN 通讯，不一定全部适合。



4.5 关于 Message 过滤

CAN 通讯接收 Message 时，可以设置只接收部份 Cobid 的信息，这就是 Rx Filter，如图：



5. 代码解释。

5.1 CAN2.0 A 和 CAN2.0B 的程序

两套 PLC 分别控制两个 EL6751，仅仅是 Cobid 赋的初值不同。左按钮发送，右按钮接收。

而 CAN2.0A 的程序和 CAN2.0B 的程序，只有 Cobid 的赋值语句不同。在 CAN2.0A 的程序中：

```
0003 IF NOT CfgFlag THEN
0004     stTxMessage[1].CobId:=SHL(CobID1,5) + DataLength1;(*16#181;   高11位表示CobID,低4位表示发送字节数*)
0005     stTxMessage[2].CobId:=SHL(CobID2,5) + DataLength2;(*16#281;*)
0006     CfgFlag:=TRUE;
0007 END_IF
```

而在 CAN2.0B 的程序中是这样的

```
0003 IF NOT CfgFlag THEN
0004     stTxMessage[1].CobId:=CobID1;
0005     stTxMessage[1].Length:=DataLength1;
0006     stTxMessage[2].CobId:=CobID2;
0007     stTxMessage[2].Length:=DataLength2;
0008     CfgFlag:=TRUE;
0009 END_IF
```

本 Demo 程序都只是最基本的设置，没有对 CAN 协议中的 RTR 和 29 位中的 Bit31 进行处理。

在 Main 中判断是否成功发送或者接收数据，然后进行简单处理。其中注释掉的代码用于自动连续发送和连续接收。

```
0009 IF TxCounter_output=TxCounter_input  THEN(*此时表示数据已发送*);
0010     NoOfRxMessages_output := NumberOfMessagesToSend;
0011     (*TxCounter_output := TxCounter_output + 1;*)(*发送完一次紧接着发送第二次*)
0012 END_IF
0013
0014 IF RxCounter_output<> RxCounter_input THEN (*此时表示收到了数据*)
0015     FOR i := 1 TO NoOfRxMessages_input DO
0016         MessageReceived[i].CobId:= stRxMessage[i].CobId;
0017         MessageReceived[i].Data:=stRxMessage[i].Data;
0018     END_FOR
0019     (*RxCounter_output:= RxCounter_Input;*) (*自动等待下一次接收*)
0020 END_IF
```

在 Action_hmi 中，是从界面按钮触发接收和发送的程序代码：

```
0001 r_trig_Send(CLK:=Send , Q=> );
0002 r_trig_Receive(CLK:=Receive , Q=> );
0003
0004 IF r_trig_Send.Q THEN
0005     (*发送按钮上升沿，输出变量TxCounter加1，以触发硬件EL6751的发送动作。*)
0006     TxCounter_output := TxCounter_output +1;
0007     FOR i:=0 TO 7 DO
0008         stTxMessage[1].Data[i]:=stTxMessage[1].Data[i]+UINT_TO_USINT(i+1);
0009         stTxMessage[2].Data[i]:=stTxMessage[2].Data[i]-UINT_TO_USINT(i+1);
0010     END_FOR
0011 END_IF
0012
0013 IF Receive AND RxCounter_output<> RxCounter_input THEN
0014     (*按下接收按钮时，如果二者不相等，表示有收到新的Message*)
0015     (*将二者设置为相等，以触发PLC程序从硬件EL6751的缓存中接收数据*)
0016     RxCounter_output:= RxCounter_Input;
0017 END_IF
```

6. 注意事项

6.1 为了避免自上次接收以来，新接收的消息超出接收队列的最大数量（e.g 10 个），应尽可能及时处理收到的 Message，并触发重新接收数据。特别是 CAN 设备不经握手，无条件快速连续发送消息时，或者网络中有多个 CAN 节点不受调度随机发送消息时，如果接收消息不及时，就容易漏掉部分信息。如果实在来不及接收，可以把接收队列的数组容量 Rx Queue Messages 设置大一些。

6.2 发送数量则刚好相反，如果每个周期都改变输出的 TxCounter 就会出现每个周期都向总线发送数据的情况，考虑到这一点之后，如果不是你想要效果，就应该给触发条件加一个上升沿。

6.3 实际使用中发现：如果 Rx 和 Tx 的消息的设置数量不相同，如果硬件是 CX5020-M510，则数量只受 Tx Queue Message 控制，而 EL6751 就没有这个问题。至于其它 CXxxxx-M510 是否有这个现象，尚未测试。

6.4 实际使用中发现，在 TwinCAT 2 中，最新版本的 EL6751 只有 CAN2.0 A 模式下，可以设置过滤，此时只要降低 EL6751 的 Firmware 到 EL6751-16-V0112.efw 版本就可以。在 TC3 下没有问题。

6.5 关于波特率要一致、CobID 不能重复、要接终端电阻、屏蔽接地、CAN H 和 CAN L 不要接反等常识问题，本文虽然没有强调，但发现通讯不上的时候，也需要逐项检查。