CAN 通信简介

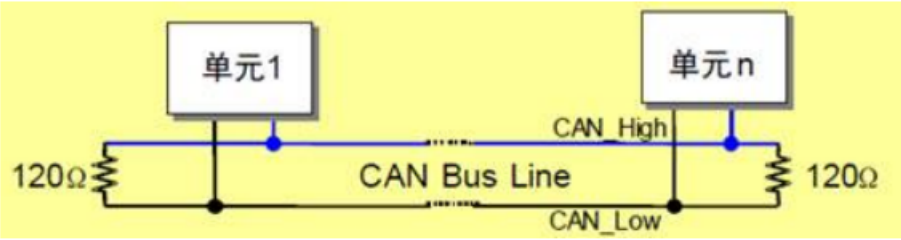
来源:湖南爱力

作者：赵宇威

1. CAN 网络的用途

CAN 网络类似于家庭网络，电脑连上网线就可以上淘宝买东西，淘宝将货品信息通过网线传输给了电脑，用户的购买信息可以通过网线传输给淘宝。汽车领域各个模块通信方式，大多为 CAN 通信，比如在家用车上面，显示屏显示空调档位，是空调控制器通过 CAN 发送档位信息给显示屏，显示屏也通过在屏幕中操作发送打开/关闭空调指令给空调控制器，这也是通过 CAN。

2. CAN 网络的构成



如上图蓝色的线是 CAN_H 线，黑色的线是 CAN_L 线，同一个 CAN 网络可以接多个 CAN 设备，他们的接线方式全都一样，控制器上面的 CAN_H 引脚接到 CAN 网络中的 CAN_H 线，控制器上面的 CAN_L 引脚接到 CAN 网络中的 CAN_L 线即可完成链接，在整个 CAN 网络的起始端与结束端需要加 120Ω的终端电阻。

CAN 网络与家庭网络一样存在不同的网速，也称为波特率，CAN 网络常用的波特率有 250k，500k 等。CAN 网络上的设备单元拥有适配波特率，同一个 CAN 网络上所有的设备的适配波特率需要一致，设备单元的适配波特率一般可修改（以适配不同波特率的 CAN 网络）。

3. CAN 网络的优势

一般情况下，控制器的引脚越多，成本越高，控制器常用的引脚一般有数字量输入输出，模拟量输入输出(此处解释一下数字量模拟量的概念，数字量只有两个值 0 和 1，也对应着否与是，模拟量对应一个具体的数据，例如 12V 电压，4-20ma 电流等等)。例如需要控制器做程序，当输入点 IN1 为 1 时，输出点 OUT1 为 12V，假如需要有十个这样的逻辑关系，则需要十个输入十个输出共 20 个点位。若用了 CAN 通信，10 个数字量输入均可通过 CAN 网络传递到控制器，这样可以通过 CAN 线传递来的数字量信号控制输出点的输出，节约了十个输入点位。

补充知识，对于数字量的输入一般有高电平有效与低电平有效两种，高电平有效就是当数字量输入引脚的电压等于电源电压时候，数字量输入值为 1，否则为 0；低电平有效的时候是当数字量输入引脚的电压等于 0V 时候，数字量输入为 1，否则为 0。

4. CAN 信号的格式

CAN 网络上的信息传递是通过 CAN 报文传输，CAN 报文是以帧的形式发送与接收，一

帧就是一个报文，一个报文是由一个 COB ID 与报文内容组成，COB ID 即报文的名称，报文内容就是传递的具体数据。（工作中使用的 CAN 报文 ID 只需了解 COB ID 即可，CAN 的 ID 很长分为许多部分，有兴趣可私下交流）

COB ID 常用的两种格式，一种是标准帧，一种是拓展帧，本质上即长度不同，一般标准帧的格式是 0x123，0x45（3 位十六进制数）6 等，拓展帧一般是 0x0CFF42FC（8 位十六进制数）等。(0x123 表示的是十六进制的 123，不是十进制的 123，学习报文最重要的是掌握十六进制与二进制与十进制之间的转化)

报文的内容为八个字节组成，一般的格式是 FF 2F 14 2C 00 11 00 00 这样的形式，这里面的数据均为 16 进制数据，解析方式如下。

0x123	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7	BYTE8
	FF	2F	14	2C	00	11	00	00

一共有八个字节，分别命名为 byte1~8（CAN 的标准格式 byte 的命名是由 1 开始，但是有的厂家会用 0 作为开始 byte0~7，这个需要根据报文定义来确认，不同的定义只影响名称，不影响字节的数据），每一个 byte 是由八个二进制数字组成，以 byte2 举例，0x2F 用二进制表达就是 0010 1111（0x2=0010，0xF=1111），分别对应 0~7 位

Byte2	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x2F	0	0	1	0	1	1	1	1

这里切记，byte 中的位，是从右往左数的而且是从 0 开始到 7（CAN 的标准格式 bit 的命名是由 0 开始，但是有的厂家会用 1 作为开始 bit1~8，这个需要根据报文定义来确认，不同的定义只影响名称，不影响位的数据），比如想判断 0x123 的 2.5 位的值，在表格中找到 bit5 的数据，00（1）0 1111 这一位值为 1，如果想判断 2.1 位的值，就是 0010 11（1）1 这一位值为 1。

每一个报文都要有对应的报文定义，即收发双方定好的协议

0x123	报文定义（周期 100ms）	
Byte1	Bit0~Bit7	/
Byte2	Bit0	/
	Bit1	/
	Bit2	/
	Bit3	/
	Bit4	/
	Bit5	控制信号
	Bit6	/
	Bit7	/
Byte3	Bit0~Bit7	/
Byte4	Bit0~Bit7	/
Byte5	Bit0~Bit7	/
Byte6	Bit0~Bit7	/
Byte7	Bit0~Bit7	/
Byte8	Bit0~Bit7	/

0x456	报文定义（周期 100ms）	
Byte1	Bit0~Bit7	/
Byte2	Bit0	/

	Bit1	故障反馈
	Bit2	/
	Bit3	/
	Bit4	/
	Bit5	/
	Bit6	/
	Bit7	/
Byte3~Byte4		反馈电压（uint 型数据，byte3 为低位）
Byte5	Bit0~Bit7	/
Byte6	Bit0~Bit7	/
Byte7	Bit0~Bit7	/
Byte8	Bit0~Bit7	/

按照上述表格:

控制器 A 发送报文 0x123 至控制器 B，控制器 B 反馈 0x456 至控制器 A，当 0x123 的 2.5 位为 1 的时候，控制器 B 引脚 X 输出一个 5V 的电压，若 X 未故障，则反馈 0x456 的 2.1 位为 1 并将电压以 0x456 的 byte3~4 发送至 A。(0x123 及 0x456 均以 100ms 的周期发送)

此时需要一个控制器 A 的数字量输入针脚 Y 与 0x123 的 2.5 位对应，当 Y 输入为 1 的时候，0x123 的 2.5 位为 1，并发送至控制器 B，当控制器 B 接收 0x123 的 2.5 位为 1 的时候，让模拟量输出 X 针脚一个 5V 的电压，并且判断 X 是否有故障，当无故障的时候，控制器 B 发送报文 0x456 的 2.1 位变为 1，此时控制器 A 可以收到控制器 B 发送来的 0x456 报文（实现此过程需要通过程序，若对程序有兴趣可私下交流）。

流程如下:

1. 0x123 与 0x456 均以 100ms 的周期在总线上发送，初始值均为 00 00 00 00 00 00 00 00
2. 控制器 A 的 Y 引脚输入为 1
3. 控制器 A 发送的 0x123 值变为 00 20 00 00 00 00 00 00
4. 控制器 B 接收到 0x123 的数据，并判断 0x123 的 2.5 位由 0 变为 1
5. 控制器 B 使 X 引脚输出 5V 电压
6. 控制器 B 判断 X 引脚是否有故障
7. 此时判断 X 无故障，0x456 的值变化为 00 02 88 13 00 00 00 00
8. 控制器 A 可以接收到 0x456 的值并确认控制器 B 的 X 引脚无故障正常输出了

这是用 A 的输入引脚去控制 B 的输出引脚。

0x456 中 byte3~4 的数据解释：设电压单位为 mv，5V 的电压值对应在控制器内部的数据是 5000，转化为十六进制的数即为 0x13 88，即此时 0x456 的内容为 00 02 88 13 00 00 00 00（根据报文表格要求，低位在 byte3，所以对于 0x13 88 来说 88 是低位，应该在 byte3 的位置，表格中描述的是 byte3~byte4 对应一个数，并且 byte3 是低位，这种说法叫低位在前，若是 byte4 的低位则是低位在后)。

报文传输模拟量需要将模拟量值转化为十六进制数再传输，报文本身不会出现负数，但可以通过补码来传输负数（此内容较为复杂，若有兴趣可私下与笔者探讨）

用报文的好处就是，同一个网络上可以存在许多的报文，如果数字量模拟量如果想远程传输给其他控制器，特别是数量较多的时候，通过报文传输会大大节约控制器点数。

5. CAN 报文的触发类型

CAN 报文的发送有多种类型，接收无类型，但是接收报文需要注意接收周期性的报文的时候，需要设置连续 5 个周期之内若是未收到报文，则立马报警并停止工作，此用意是防止 CAN 总线故障时候设备还按照设定前的工作状态工作，会导致故障发生。

信号发送类型		相关属性	行为描述
cyclic	cyclic	GenMsgCycleTime	周期发送
OnEvent	OnWrite	-	写信号后发送
	OnChange	-	信号值改变后发送
	OnWriteWithRepetition	GenMsgNrOfRepetition	写信号后重复发送
	OnChangeWithRepetition	GenMsgNrOfRepetition	信号值改变后重复发送
IfActive	IfActive	GenMsgCycleTimeFast; GenSigInactiveValue	信号激活后，以快速周期发送报文，否则不发送。
	IfActiveWithRepetition		

6. CAN 报文上层应用

在 CAN 的基础上，can 报文有 CANOPEN 与 J1939 协议，在 CAN 的底层协议中，报文内容是由用户自己定义的，比如用户想定义 0x123 的 2.5 是表示某某开关，用户可在程序中就按照这个逻辑写，但是 CANOPEN 与 J1939 是生产商出厂的时候就定义好了报文内容的，比如发动机就是用的 J1939 协议，所有的发动机报文都是同一套，CANOPEN 也是一样的，不过 CANOPEN 中的报文形式与自由 CAN 有所不同，若是有兴趣可以与笔者私下探讨。

7. 数据类型表格

Data type	min. value	max. value	size in the memory
BOOL	FALSE	TRUE	8 bits = 1 byte
BYTE	0	255	8 bits = 1 byte
WORD	0	65 535	16 bits = 2 bytes
DWORD	0	4 294 967 295	32 bits = 4 bytes
SINT	-128	127	8 bits = 1 byte
USINT	0	255	8 bits = 1 byte
INT	-32 768	32 767	16 bits = 2 bytes
UINT	0	65 535	16 bits = 2 bytes
DINT	-2 147 483 648	2 147 483 647	32 bits = 4 bytes
UDINT	0	4 294 967 295	32 bits = 4 bytes
REAL	-3.402823466 •1038	3.402823466 •1038	32 bits = 4 bytes
ULINT	0	18 446 744 073 709 551 615	64 Bit = 8 Bytes
STRING			number of char. + 1