

# ECUCoder™ 使用说明书

——ECUCoder for MPC55xx

## 目录

ECUCoder 介绍 .....	5
版本信息 .....	6
软件安装前的准备 .....	7
一、 软件安装.....	8
常见问题及解决方法: .....	9
二、 软件设置.....	10
三、 快速入门.....	11
四、 基本输入与输出.....	14
1. 获取输入量 .....	14
1.1 获取开关量输入的值.....	14
1.2 获取模拟量输入的值.....	15
1.3 获取频率量输入的值.....	16
2. 控制输出量 .....	19
2.1 控制功率驱动开关.....	19
2.2 控制模拟量输出.....	21
2.3 控制功率驱动 PWM.....	22
2.4 控制直流电机.....	24
2.5 控制恒流驱动.....	26
五、 基本 CAN 通信与 LIN 通信 .....	29
1. CAN 模块初始化设置 .....	29
1.1 设置波特率.....	29
1.2 其它设置.....	30
2. CAN 通信模块库 .....	30
3. 发送 CAN 报文 .....	31
4. 接收指定 ID 的 CAN 报文.....	34
4.1 定时接收 CAN 报文 .....	34
4.2 中断接收 CAN 报文 .....	36
5. 接收任意 ID 的 CAN 报文.....	38
5.1 定时接收 CAN 报文 .....	38

5.2 中断接收 CAN 报文 .....	40
6. LIN 模块初始化设置 .....	42
7. LIN 通信模块库 .....	42
8. 发送 LIN 报文 .....	43
9. 接收 LIN 报文 .....	46
<b>六、 测量与标定 .....</b>	<b>48</b>
1. 设置 CCP 参数 .....	48
2. 添加测量量 .....	48
2.1 添加测量量：使用模块方式 .....	48
2.2 添加测量量：使用信号属性方式 .....	50
3. 添加标定量 .....	52
2.1 添加标量标定量：使用模块方式 .....	53
2.2 添加标量标定量：使用命令行方式 .....	55
2.3 添加一维查表标定量 .....	58
2.4 添加二维查表标定量 .....	60
<b>七、 DBC 工具的使用 .....</b>	<b>63</b>
1. DBC 解析模块自动生成 .....	63
2. DBC 解析模块与 CAN 模块的集成 .....	66
2.1 发送报文的模型集成 .....	66
2.2 接收报文的模型集成 .....	67
<b>八、 控制器的管理 .....</b>	<b>68</b>
1. 单片机管理 .....	68
1.1 添加初始化任务 .....	69
1.2 添加周期性任务 .....	71
2. 电源管理 .....	72
2.1 电源监控 .....	73
2.2 电源控制 .....	73
3. 存储器管理 .....	73
3.1 使能 Flash 存储器 .....	74
3.2 读写 Flash 存储器 .....	75

九、	如何建立简洁高效的模型 .....	77
1.	为模块指定数据类型 .....	77
1.1	需要指定数据类型的模块.....	77
1.2	指定数据类型的常用方法.....	77
2.	明确任务的执行周期与执行顺序 .....	78

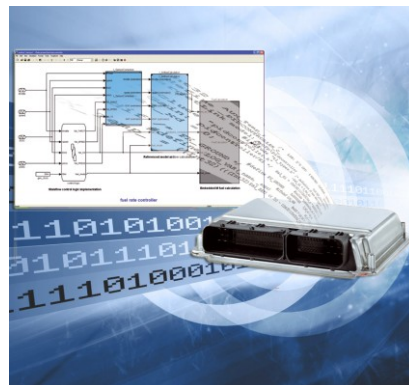
## ECUCoder 介绍

ECUCoder 是基于 MATLAB/Simulink 的全自动代码生成工具，用于配置 ECU 控制算法模型与基础软件模型，并自动生成产品代码。支持 NXP/飞思卡尔、意法等知名厂家的汽车电控系统主流芯片。

ECUCoder 提供了功能强大的基础软件 Simulink 模块库，可以通过友好的用户界面便捷、直观地配置基础软件参数并由 Simulink 模型自动生成基础软件代码。由于软件可以灵活、深层次地访问并配置基础软件参数，模型生成的基础软件代码可以支持控制器快速原型及产品开发两个阶段。

### ECUCoder 目前支持如下处理器

- ✓ NXP/Freescale 飞思卡尔 MPC55xx/56xx/57xx 系列
- ✓ NXP/Freescale 飞思卡尔 S12/S12x 系列
- ✓ ST 意法半导体 SPC56X 系列
- ✓ 可根据用户需求提供针对其它处理器的定制服务



### ECUCoder 应用方式

- ☆ 基于目标控制器的全自动代码生成
  - ★ RapidECU 快速原型控制器
  - ★ 客户自研或供应商控制器
- ☆ 基于目标微控制器芯片的全自动代码生成

### ECUCoder 目前提供如下基础软件模块

- 底层驱动
- 标定协议
- 引导加载程序
- 实时操作系统
- 通信协议栈
- 诊断协议栈
- 可根据用户需求提供针对其它功能模块的定制服务

### ECUCoder 的主要特点

- ✓ 自动代码同时生成基础软件与应用软件，无需手动集成；
- ✓ 模型自动优化配置，无需手动设置 Simulink 配置参数；
- ✓ 功能强大的 GUI 界面，可直接从模型访问并配置整个基础软件；
- ✓ 后台自动调用编译器，无需手动干预；
- ✓ 代码可靠，代码可读性与执行效率良好折中；
- ✓ 同时提供芯片级模块库与控制器级模块库，支持用户自主开发的控制器硬件

## 版本信息

版本号	更改信息	发布日期
2.14	历史版本	2016.4.12
2.15	添加对 U2M 硬件的支持，U2M 电机驱动电流采样方式与 U1M 不同，需要使用不同的软件模块	2016.5.17
2.16	完善大功率电机驱动模块（独立说明书） 添加 USBCAN 卡 WIN10 64 位操作系统驱动	2016.5.29
2.17	修正大功率电机驱动模块中低压配置 BUG	2016.6.7
2.18	添加 XGP 控制器硬件支持子库，完善大功率电机驱动模块（强制角）	2016.7.2
2.20	修正 CVT 控制器高边驱动 BUG 并添加电源监控模块	2016.7.22
2.21	XGP 子库 EEPROM 读写数据类型修改为 single，添加频率量的脉宽读取功能	2016.8.17
2.22	XGP 子库开关量读取模块添加管脚 56 添加 FCE 控制器硬件支持子库	2016.9.14
2.23	优化 ecucoder_cal()函数以加快执行速度	2016.10.10
2.24	修正 FCE 控制器低边驱动 BUG FCE 与 XGP 模拟输入模块添加单位 添加 CCRI 控制器硬件支持子库	2016.10.16
2.25	修正 CCRI 控制器开关量输入与频率量输入模块 BUG	2016.11.17
2.26	修改电平持续时间的测量方法 修正 CAN 发送模块参数 BUG	2016.12.2
2.27	添加独立的 U2 控制器硬件支持子库	2016.12.12
2.28	修改 A2L 文件 UINT32 数据上限格式以消除与 CANape 不兼容的隐患	2017.1.8
2.29	修正 U2 控制器频率量脉宽读取 BUG 添加模块库软件版本号模块	2017.3.8
2.30	添加 FCE 控制器双端 PWM 驱动模块	2017.3.10
2.31	升级 CCRI 控制器硬件支持子库 添加测量模块并更新说明书	2017.5.2
2.32	修正 CCRI 控制器低端驱动 BUG	2017.5.15

## 软件安装前的准备

### 安装与使用要求：

最低硬件配置：

- ✓ CPU：双核及以上；内存：4G 及以上；空闲硬盘空间：1000M 及以上

软件环境要求：

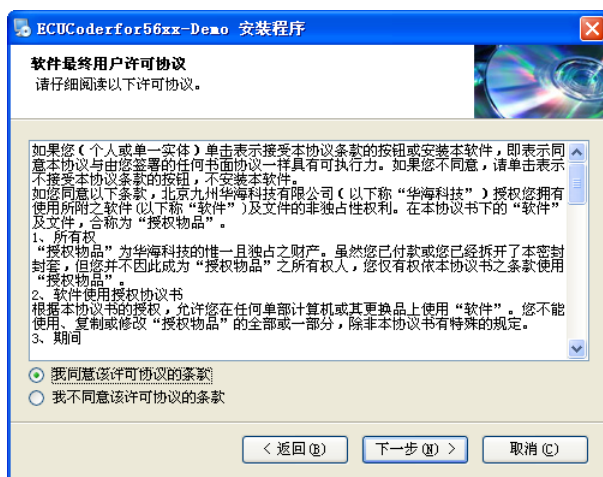
- ✓ 操作系统：Windows XP/Vista/7/10 32 位版本或者 64 位版本
- ✓ MATLAB 版本：R2010a - R2014a 32 位 Windows 版本
- ✓ 编译器：CodeWarrior 2.9/2.10

其它要求：

- ✓ 软件安装用户必须是计算机管理员账户

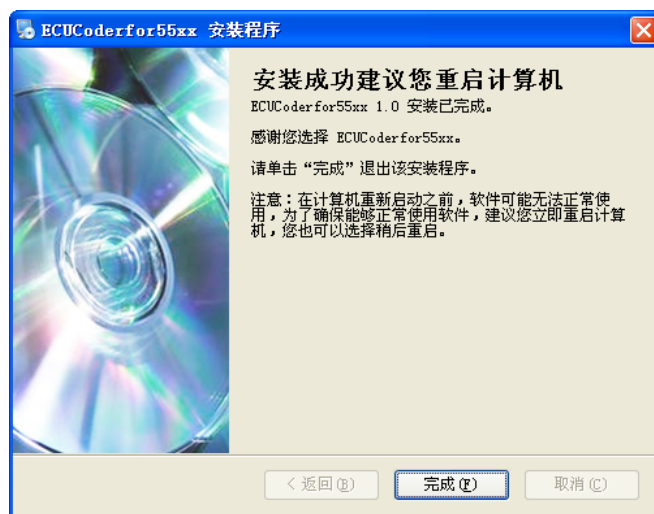
## 一、 软件安装

1. 插入 ECUCoder 软件加密狗。
2. 双击软件安装包文件，依次点击“下一步”，选择“我同意许可协议的条款”；

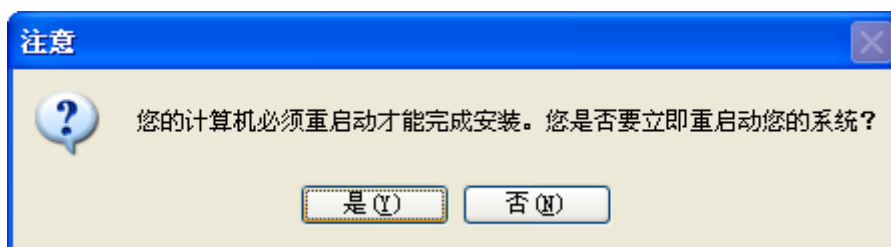


点击“下一步”，填写用户“名称”与“公司”，选择“安装路径”<sup>①</sup>；

点击“下一步”，再点击“下一步”，等待软件安装进度；



安装成功，提示重启计算机；



选择“是”立即重启计算机<sup>②</sup>。

3. 软件安装完毕。

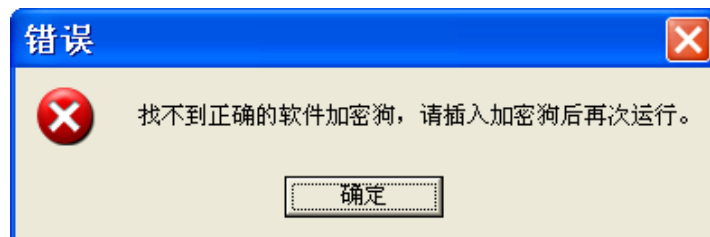


## 安装提示

- ① 此处安装路径中不可出现中文路径，否则可能导致软件无法正常使用。
- ② 重启计算机之后软件方可正常使用，否则可能导致软件无法正常使用。

## 常见问题及解决方法：

问题 1：安装过程中弹出如下对话框。



问题原因：计算机上没有插入正确的软件加密狗。

解决方法：插入 ECUCoder 软件配套加密狗。

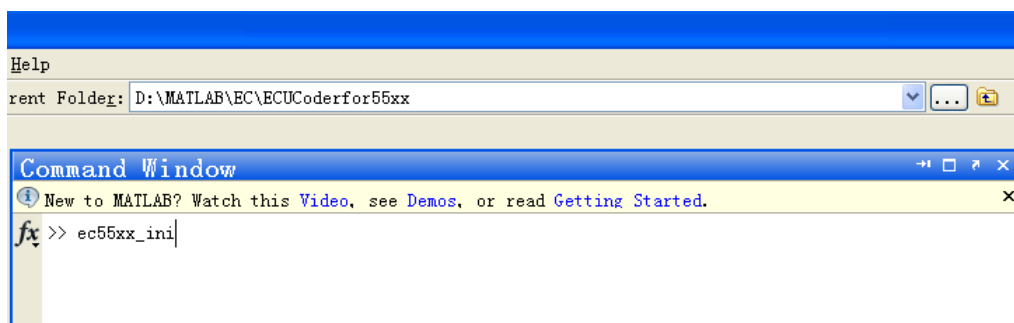
问题 2：软件安装后无法使用。

问题原因：未重启计算机。

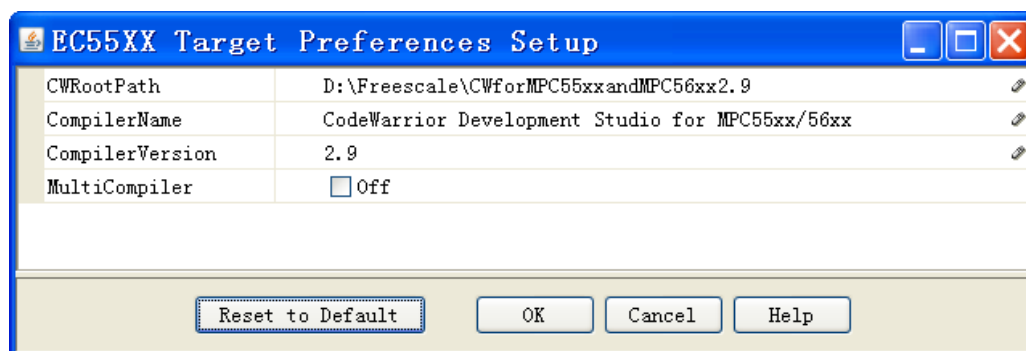
解决方法：重启计算机。

## 二、 软件设置

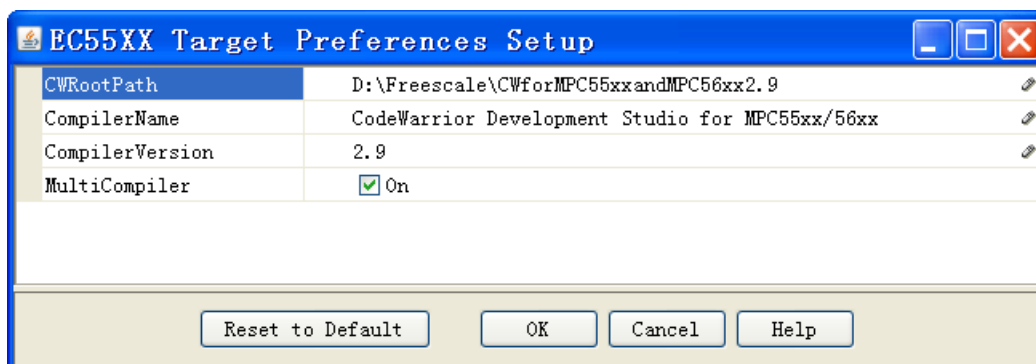
ECUCoder 软件安装完成并且重启电脑后，打开 MATLAB 软件（要求版本 R2010a 及以上），选择 ECUCoder 软件根目录，在 MATLAB 命令窗口输入“ec55xx\_ini”命令，如下图所示，根据弹出窗口的提示进行相应设置即可。



- a) 如果当前计算机上只安装了 1 个版本的 CodeWarrior 软件（要求版本 CW for MPC55xx, MPC56xx 2.9 及以上）的话，在弹出的对话框中点击 **Reset to Default** 按钮，如下图所示，然后点击 **OK** 即完成软件



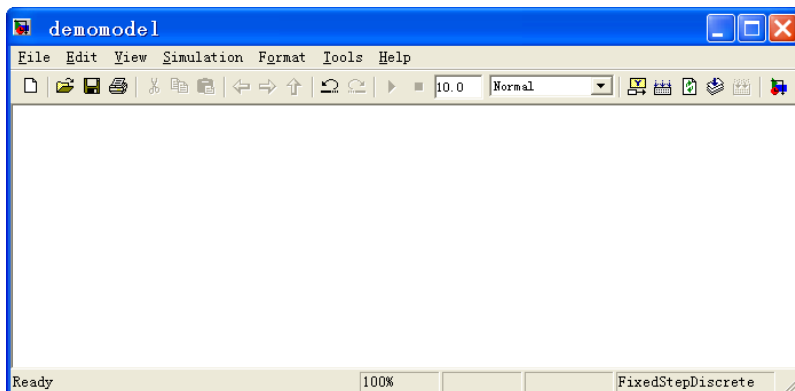
- b) 如果当前计算机上安装了多个版本的 CodeWarrior 软件的话，在弹出的对话框中勾选 **MultiCompiler** 选项后面的小方框，并在 **CWRootPath** 选项后面输入 CodeWarrior 软件的安装路径（注意：此处 CodeWarrior 软件的安装路径中不允许有空格以及中文路径，需要在安装 CodeWarrior 软件时将路径中的空格删除，并确保安装路径中无中文路径。）



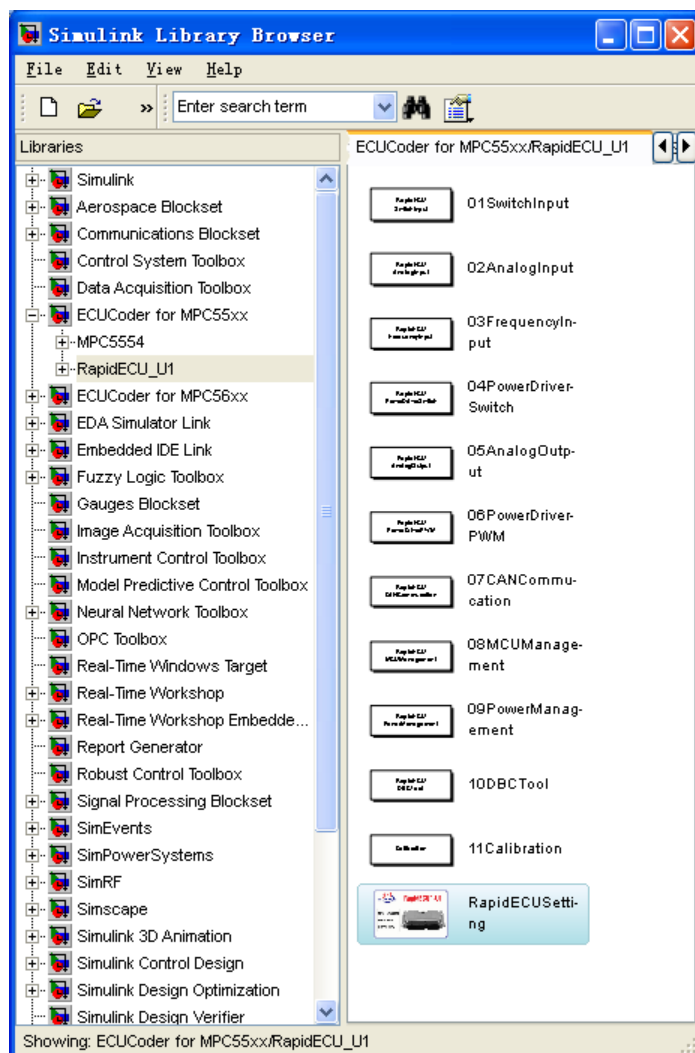
软件设置完成。

### 三、 快速入门

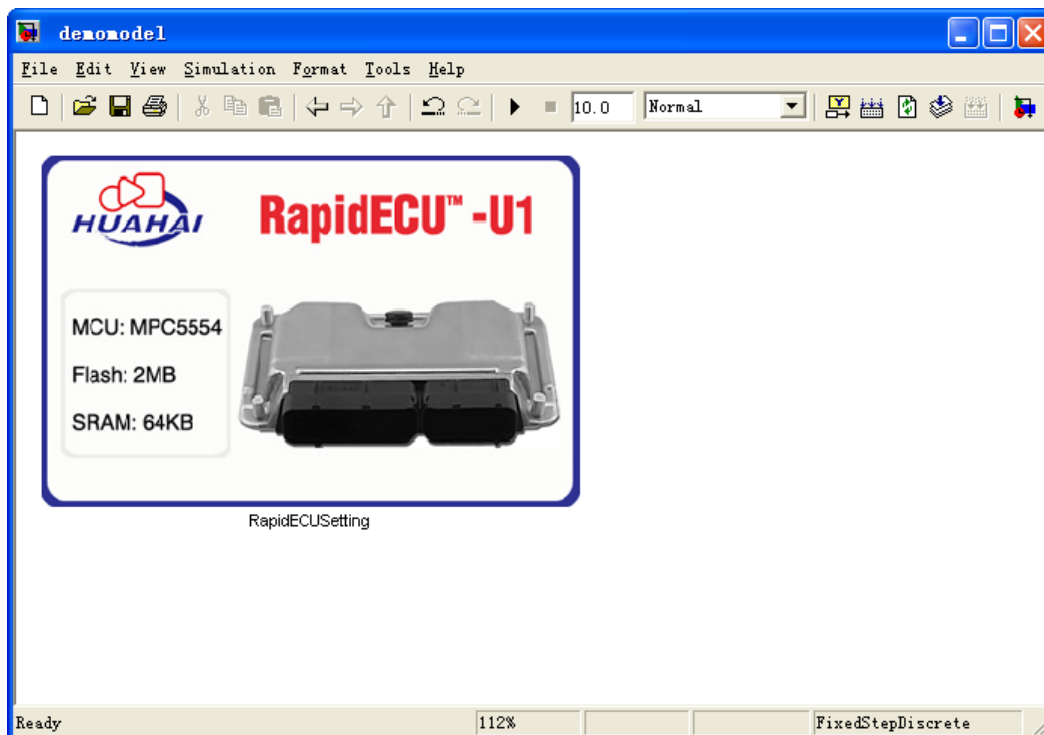
1. 打开 MATLAB 软件，将 MATLAB 路径切换到合适的工作路径下，打开 Simulink，新建 1 个模型，并将模型命名保存，如下图所示。



2. 点击进入 Simulink 的 ECUCoder for MPC55xx 模块库，点击 RapidECU\_U1 子库，选择 RapidECUSetting 模块，如下图所示。



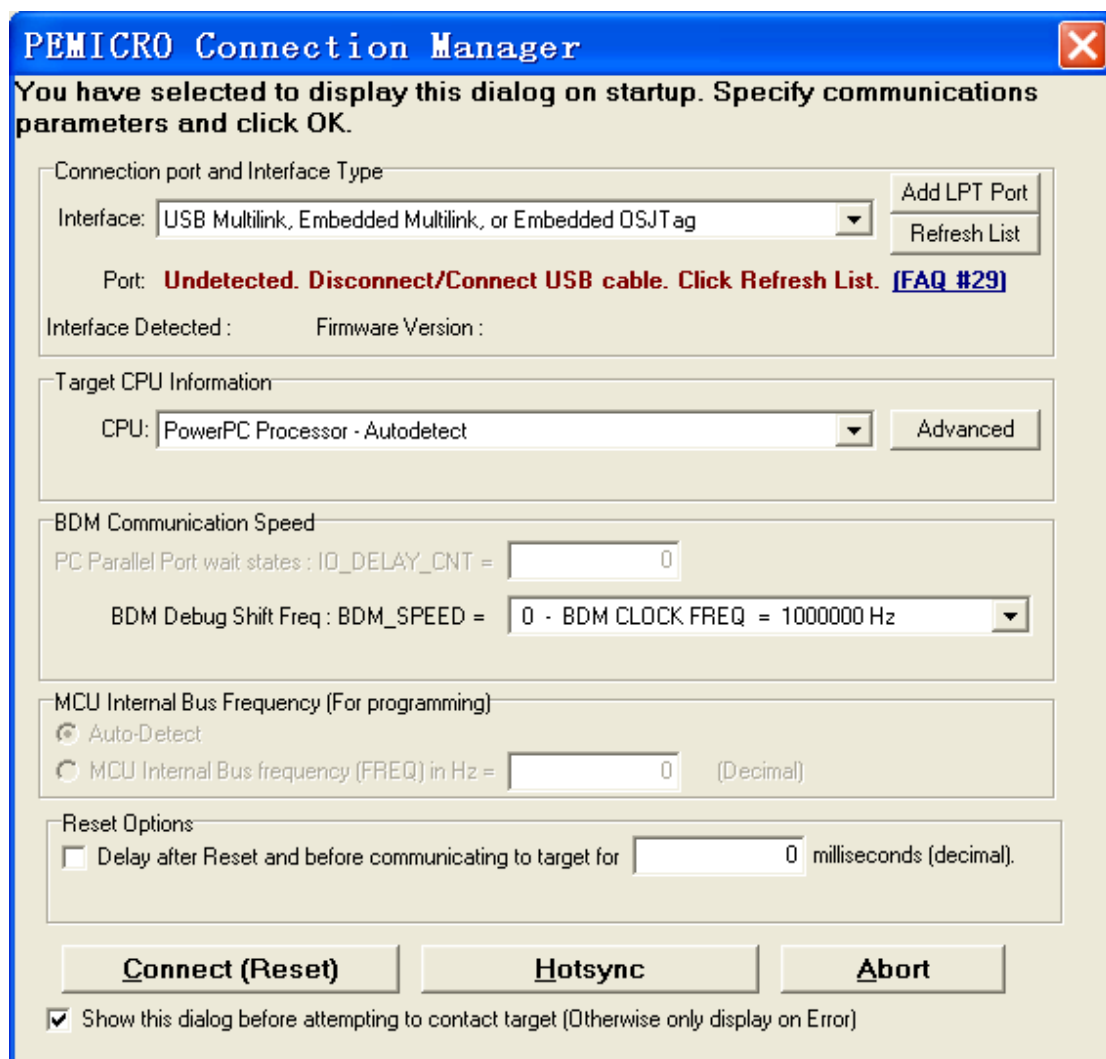
- 将该模块拖入到新建的模型中，如下图所示，模型即设置完成。点击模型右上角的 **Incremental Build** 按钮，或者点击键盘的 **Ctrl+B** 按键，模型即可自动完成代码生成、编译、连接等步骤，同时生成 A2L 数据库文件与可下载文件。



- \* 自动生成的代码报告如下图所示，可以查看，也可以直接点击 **OK** 确定。



\*\* 编译连接完成时弹出的对话框如下图所示。



通常情况下，使用 **Bootloader** 功能进行程序烧写，此时点击 **Abort** 按钮，然后启动烧写程序即可。如果需要使用烧写器进行程序烧写的话，在确保烧写器已经正确连接的情况下，点击 **Connect(Reset)**按钮进行程序烧写。

## 四、 基本输入与输出

### 1. 获取输入量

#### 1.1 获取开关量输入的值

点击 SwitchInput 子库，如下图所示，库中包含 1 个名称为 SwitchInput 的模块，该模块的作用见表 4-1。

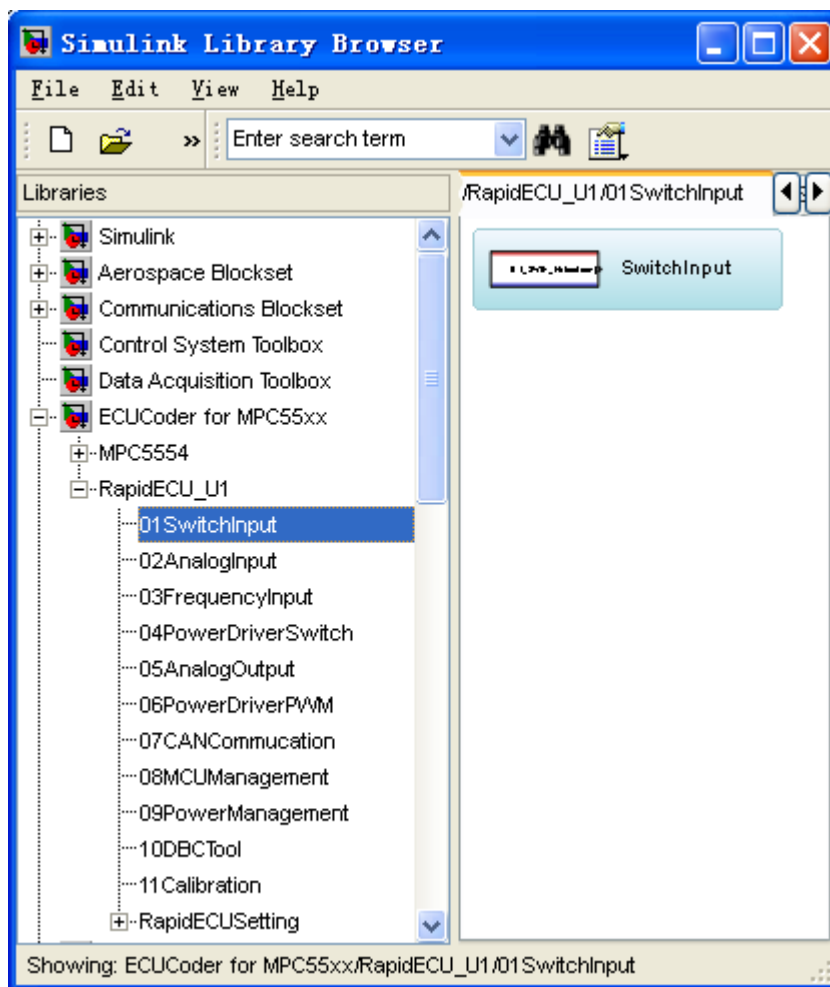
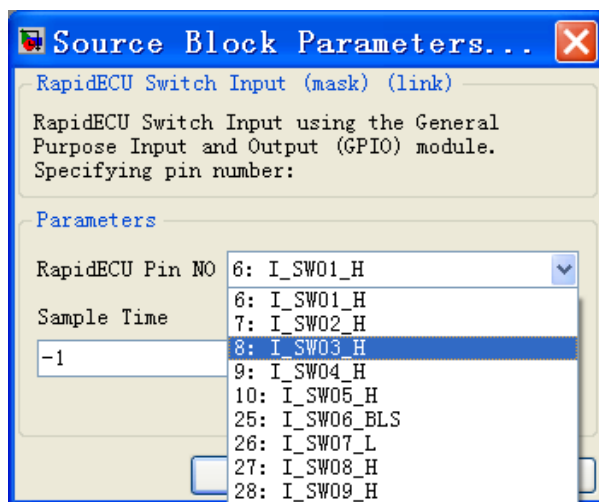


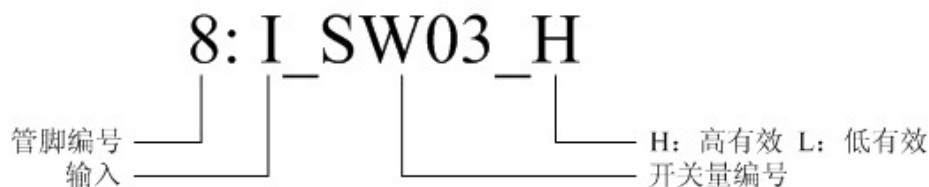
表 4-1 SwitchInput 子库

模块名称	描述	数据类型	备注
SwitchInput	开关量输入	boolean	1: 高电平 0: 低电平

将模块拖入到模型中，并选择相应的管脚，即可获取相应管脚的输入值。例如，下图所示为选择管脚 8: I\_SW03\_H，即可获取控制器管脚 8 的输入值。



管脚命名方式如下图:



## 1.2 获取模拟量输入的值

点击 AnalogInput 子库, 如下图所示, 库中包含 3 个模块, 分别为 AnalogInput 模块、AnalogOutputMonitor 模块与 PowerVoltageMonitor 模块, 各个模块的作用见表 4-2。

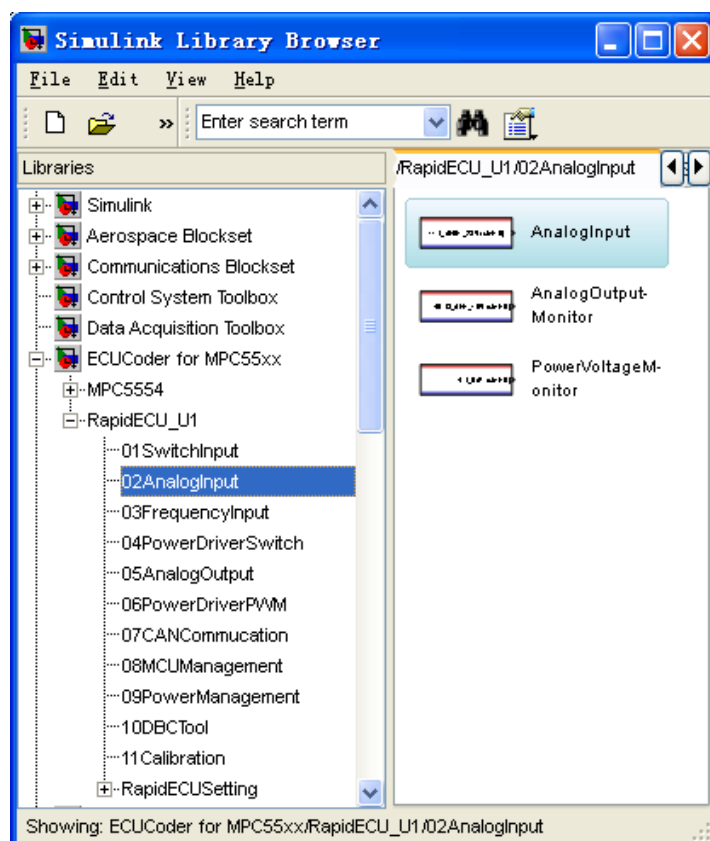
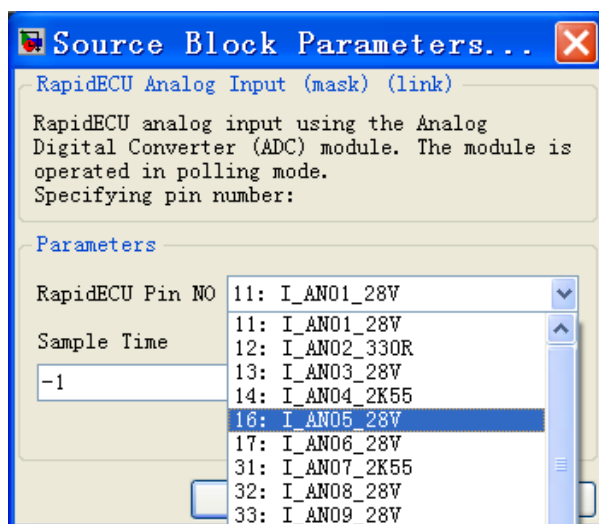


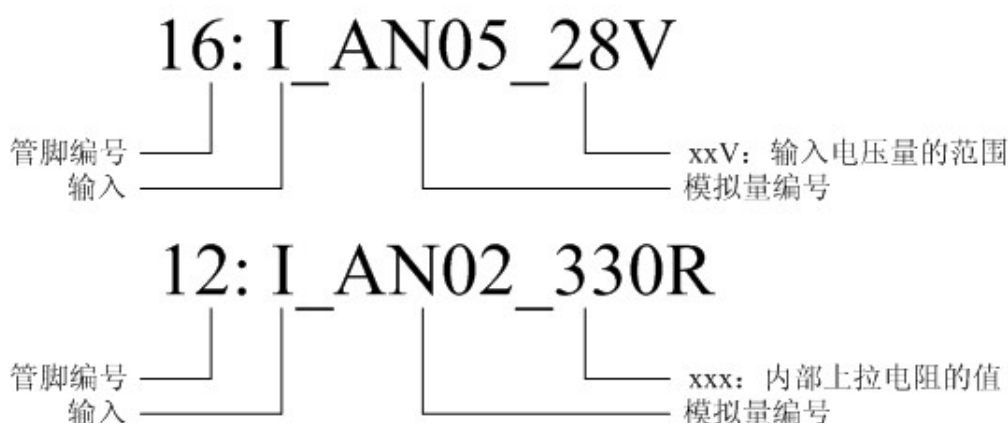
表 4-2 AnalogInput 子库

模块名称	描述	数据类型	单位
AnalogInput	外部模拟量输入	uint16	mV
AnalogOutputMonitor	模拟量输出监控	uint16	mV
PowerVoltageMonitor	控制器内部电源监控	uint16	mV

将需要使用的模块拖入到模型中，并选择相应的管脚或者功能模块，即可获取相应管脚或者功能模块的输入值。例如，下图所示为外部模拟量输入 AnalogInput 模块，选择管脚 16: I\_AN05\_28V，即可获取控制器管脚 16 的输入值。



管脚命名方式如下图：



### 1.3 获取频率量输入的值

点击 FrequencyInput 子库，如下图所示，库中包含两个模块，分别为 FrequencyRead 模块与 LevelRead



模块，各个模块的作用见表 4-3。

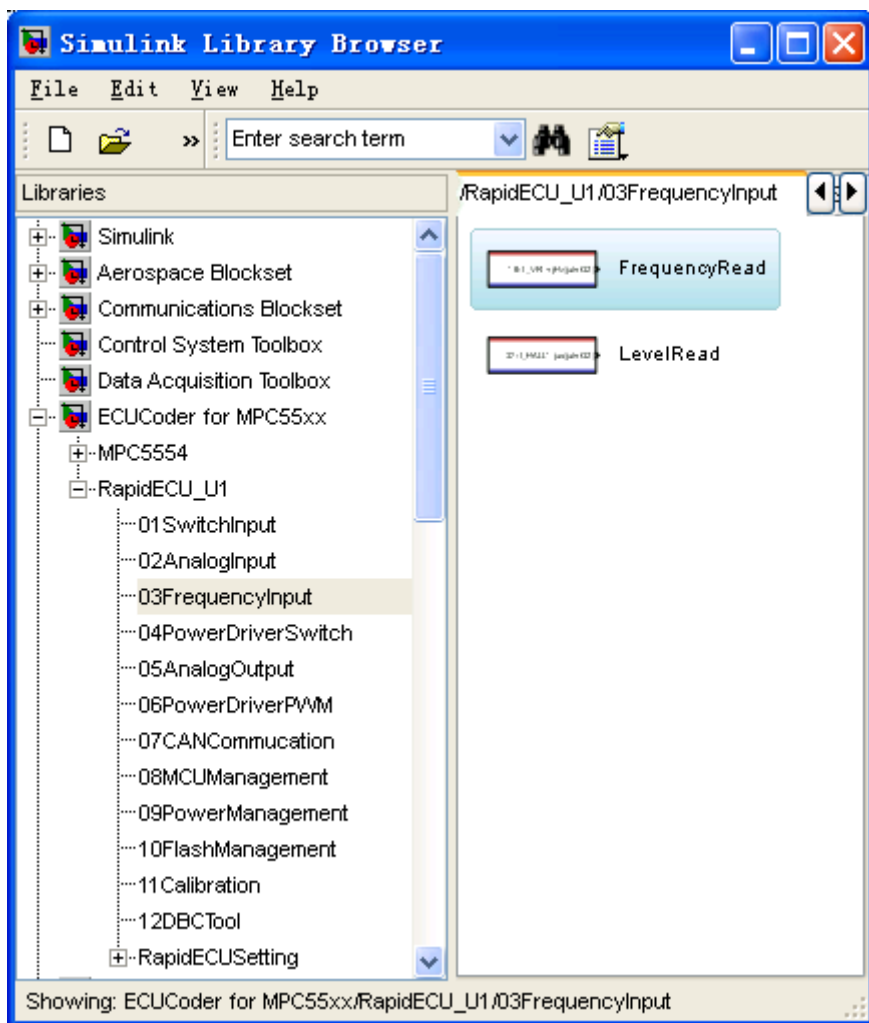
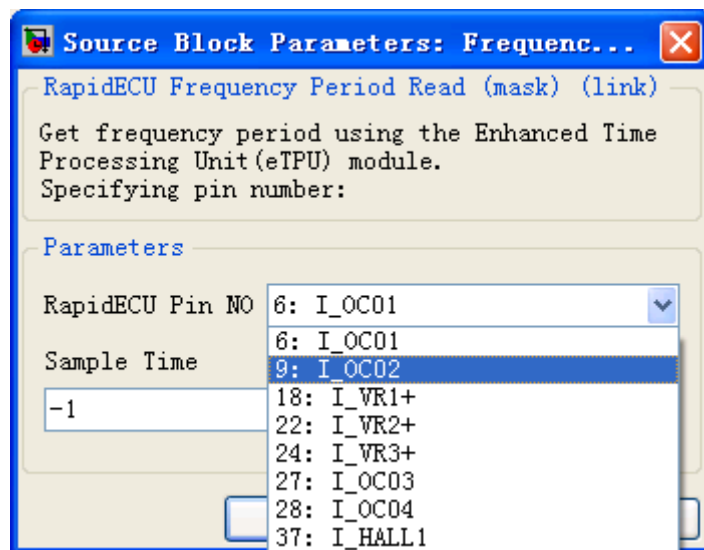


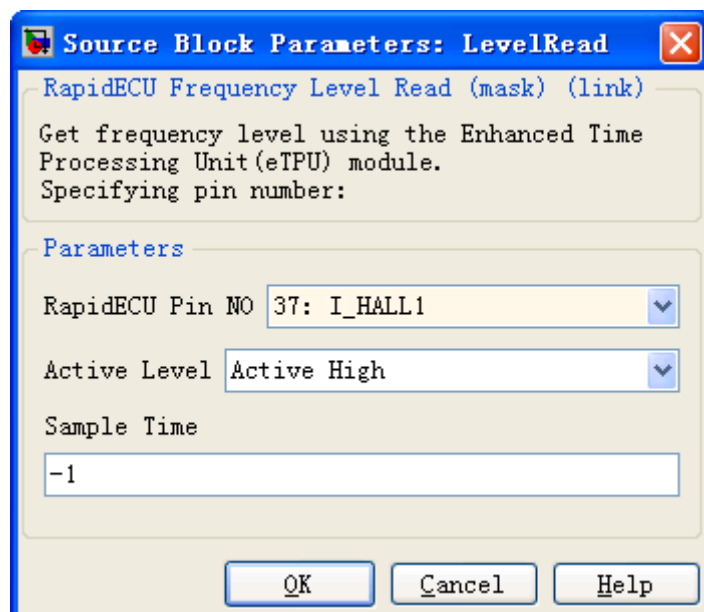
表 4-3 FrequencyInput 子库

模块名称	描述	数据类型	单位
FrequencyRead	频率量频率输入	uint32	Hz
LevelRead	频率量电平输入	uint32	us

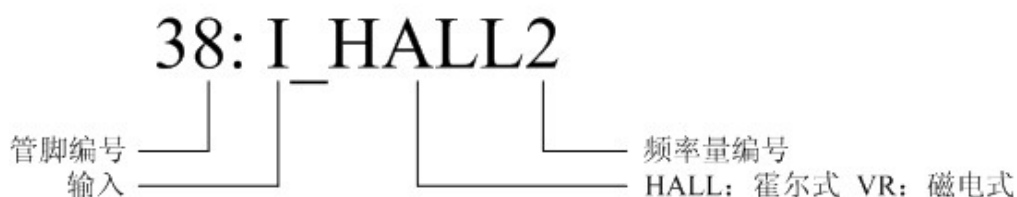
**FrequencyRead 模块的使用：**将模块拖入到模型中，选择相应的管脚，即可获取相应管脚输入频率信号频率的值。例如，下图所示为选择管脚 9: I\_OC02，即可获取控制器管脚 9 输入频率信号频率的值。



**LevelRead 模块的使用：**将模块拖入到模型中，设置需要采集的电平类型，并选择相应的管脚，即可获取相应管脚的输入值。例如，下图所示为选择管脚 37: I\_HALL1，即可获取控制器管脚 37 输入信号高电平持续的时间。



管脚命名方式如下图：



## 2. 控制输出量

### 2.1 控制功率驱动开关

点击 PowerDriverSwitch 子库，如下图所示，库中包含两个模块，分别为 PowerDriverSwitch 模块，PowerDriverSwitchDiagnosis 模块与 HighSideSwitchStatus 模块，模块的作用见表 4-4。

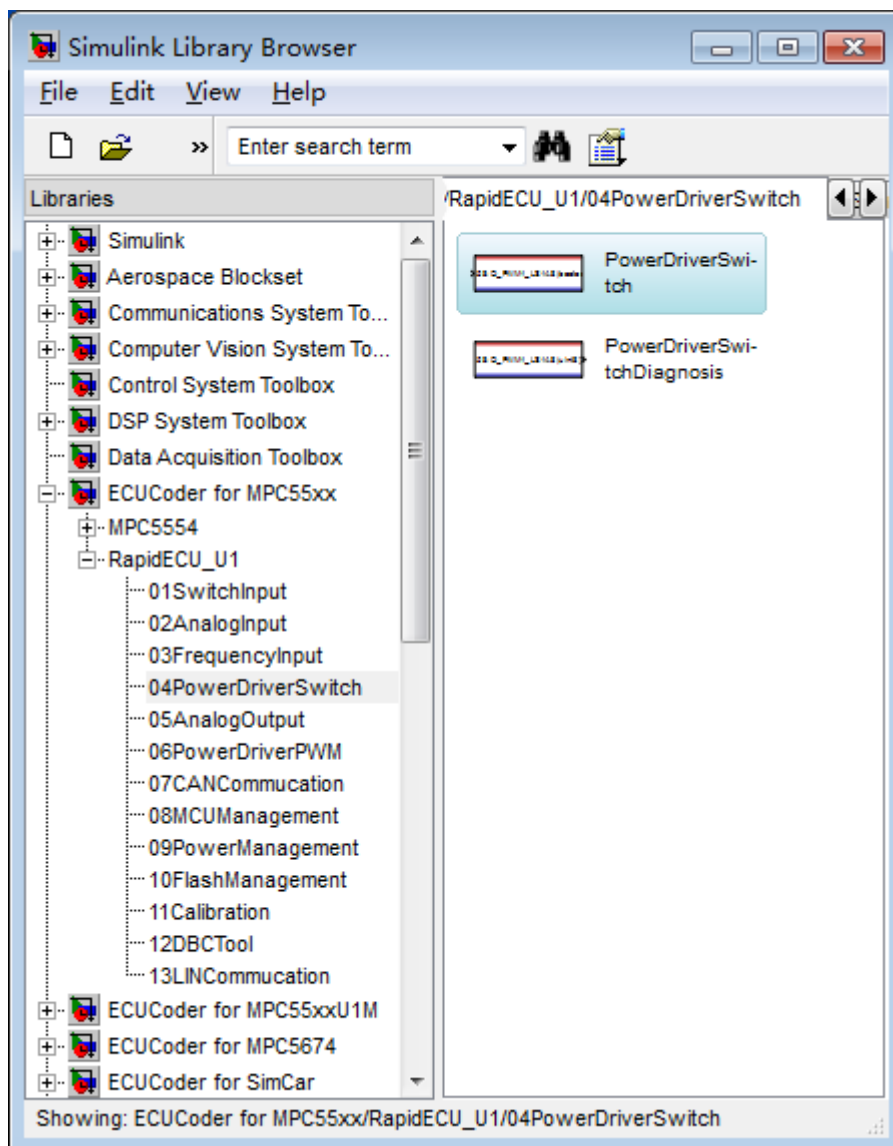
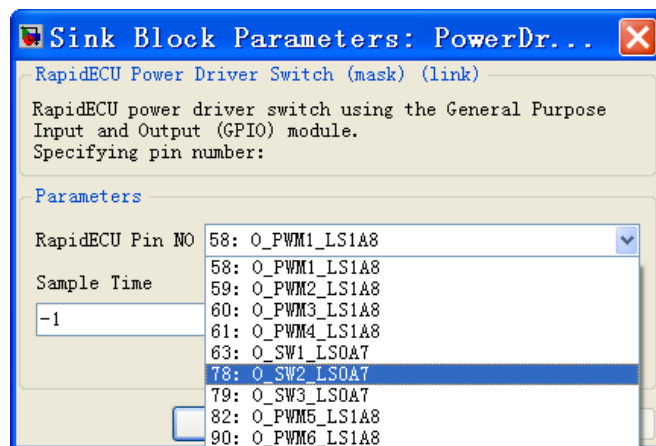


表 4-4 PowerDriverSwitch 子库

模块名称	描述	数据类型	备注
PowerDriverSwitch	功率驱动开关	boolean	1: 闭合 0: 断开
PowerDriverSwitchDiagnosis	功率驱动开关诊断	uint8	0: 短路到地

			1: 开路 2: 短路到电源或者过载、过温故障 3: 正常
HighSideSwitchStatus	高端功率驱动开关状态	boolean	正常状态下输出为 0 时状态也为 0，输出为 1 时状态也为 1； 开路或者短路到电源时状态始终为 1； 过热或者短路到地时状态时钟为 0。

将 PowerDriverSwitch 模块拖入到模型中，并选择相应的管脚，即可控制相应管脚的输出值。例如，下图所示为选择管脚 78: O\_SW2\_LS0A7，即可控制控制器管脚 78 的输出值。



**注意：**由于部分功率驱动开关与功率驱动 PWM 复用控制器管脚（比如管脚 58: O\_PWM1\_LS1A8），在一个模型中，一个控制器管脚只能选择一种功能使用，不可将同一管脚同时用作功率驱动开关与功率驱动 PWM。

管脚命名方式如下图：



## 2.2 控制模拟量输出

点击 AnalogOutput 子库，如下图所示，库中包含两个模块，分别为 AnalogOutput 模块与 AnalogOutputMonitor 模块，各个模块的作用见表 4-5。

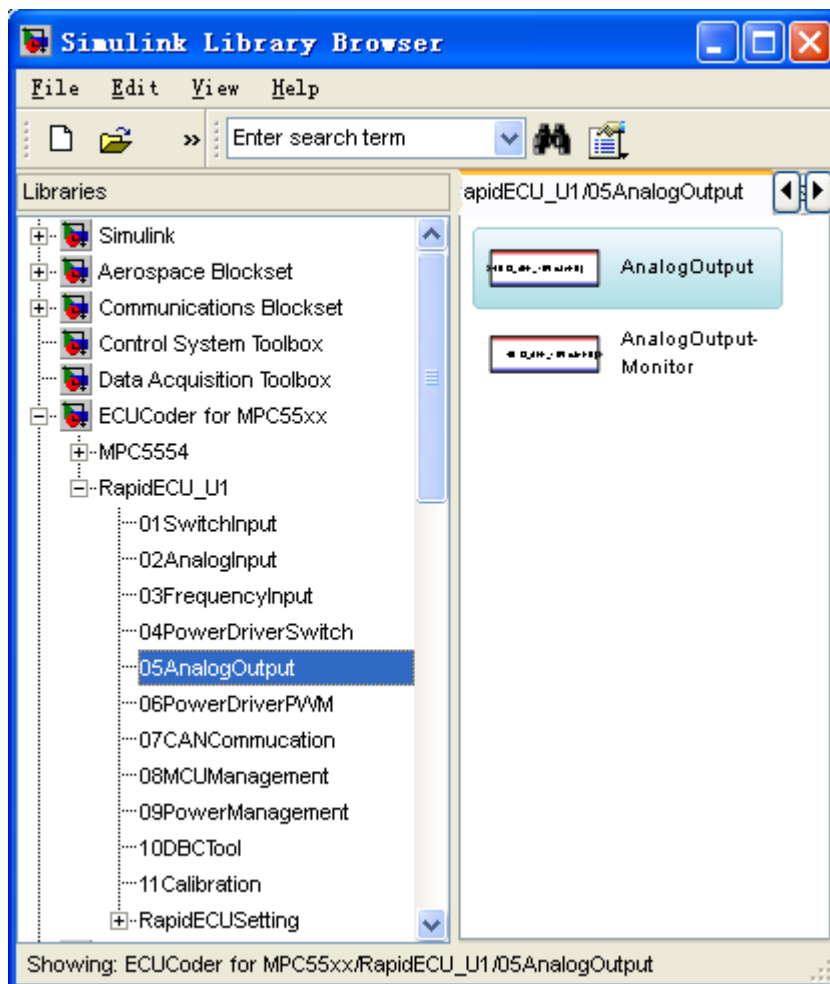
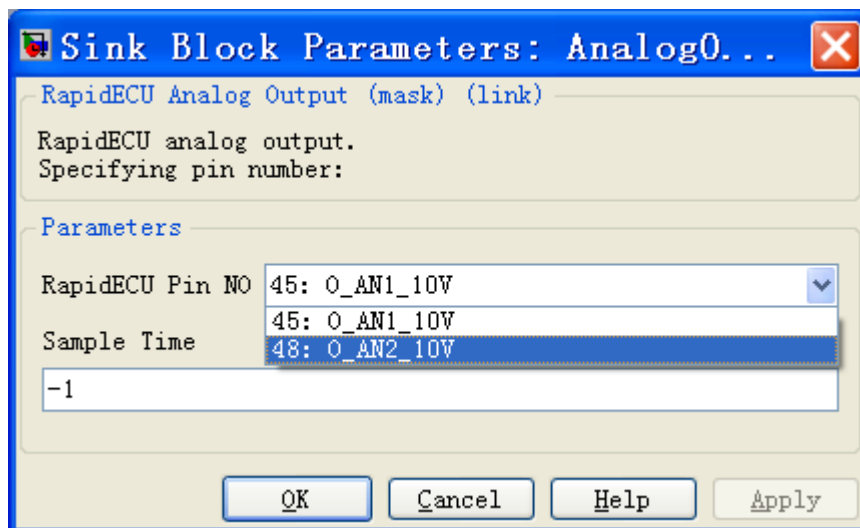


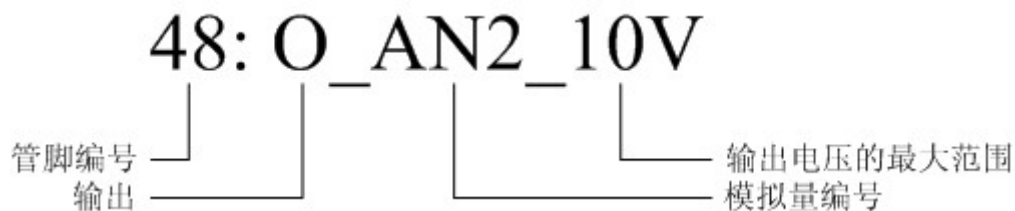
表 4-5 AnalogOutput 子库

模块名称	描述	数据类型	单位
AnalogOutput	模拟量输出	uint16	mV
AnalogOutputMonitor	模拟量输出监控	uint16	mV

将需要使用的模块拖入到模型中，并选择相应的管脚，即可控制相应管脚的输出值。例如，下图所示为模拟量输出 AnalogOutput 模块，选择管脚 48: O\_AN2\_10V，即可控制控制器管脚 48 的输入值。



管脚命名方式如下图：



## 2.3 控制功率驱动 PWM

点击 PowerDriverPWM 子库，如下图所示，库中包含 1 个名为 PowerDriverPWM 的模块，模块的作用见表 4-6。

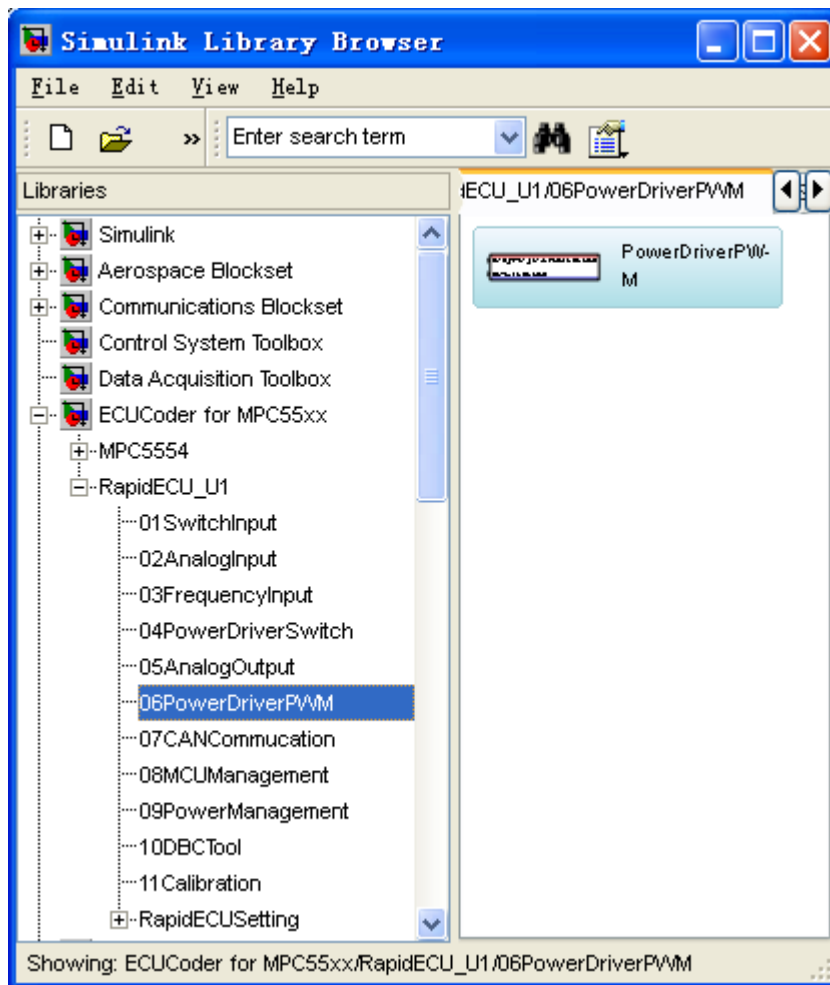
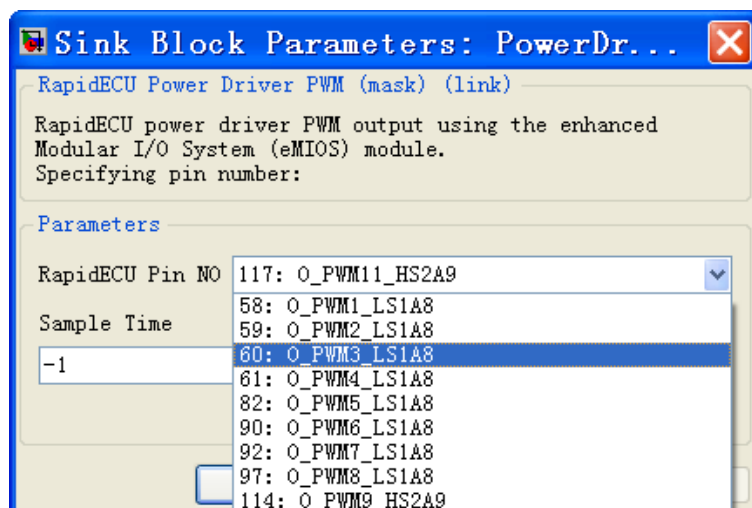


表 4-6 PowerDriverPWM 子库

模块名称	描述	数据类型	备注
PowerDriverPWM	功率驱动 PWM	uint32	输入端口 1: PWM 周期, 单位 us 输入端口 2: 导通时间, 单位 us

将模块拖入到模型中，并选择相应的管脚，即可控制相应管脚的输出值。例如，下图所示为选择管脚 60: O\_PWM3\_LS1A8，即可控制控制器管脚 60 的输出值。



**注意：**由于功率驱动 PWM 与部分功率驱动开关复用控制器管脚，在一个模型中，一个控制器管脚只能选择一种功能使用，不可将同一管脚同时用作功率驱动开关与功率驱动 PWM。

管脚命名方式如下图：



## 2.4 控制直流电机

点击 MotorDriver 子库，如下图所示，库中包含 6 个模块，分别为 Motor1 模块、Motor1Current 模块、Motro2 模块、Motor2Current 模块、Motor3 模块与 Motor3Current 模块，各个模块的作用见表 4-7。



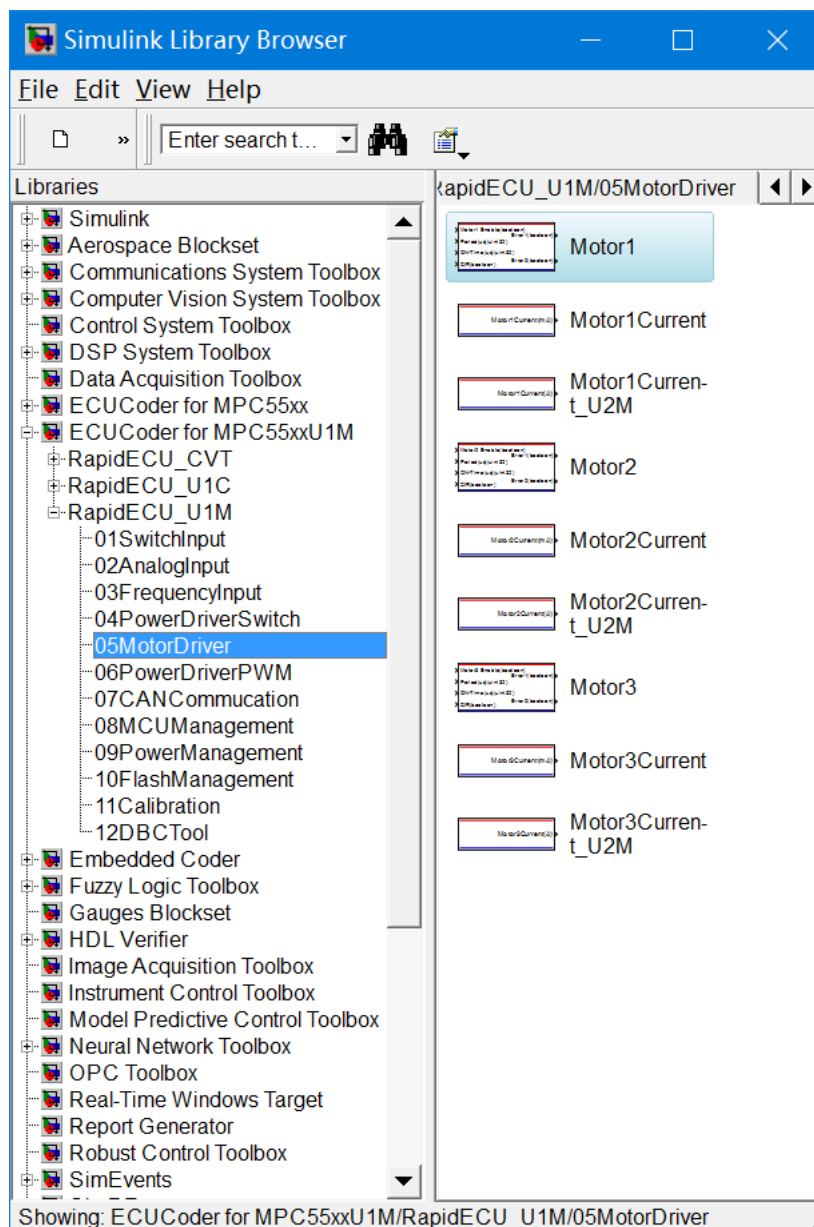


表 4-7 MotorDriver 子库

模块名称	描述	备注
Motor1	直流电机 1 驱动	输入端口 1: 直流电机驱动使能, 1 为使能, 0 为禁止 输入端口 2: 电机驱动 PWM 周期, 单位 us
Motor2	直流电机 2 驱动	输入端口 3: 电机驱动 PWM 导通时间, 单位 us 输入端口 4: 直流电机转向
Motor3	直流电机 3 驱动	输出端口 1: 电机驱动 H 桥 1 故障 (过流、短路、过热, 该端口在 U2M 控制器中无效, U2M 通过电流值判断故障)

		输出端口 2: 电机驱动 H 桥 2 故障 (过流、短路、过热, 该端口在 U2M 控制器中无效, U2M 通过电流值判断故障)
Motor1Current~ Motor3Current	电机驱动电流读取	输出端口: 电机驱动电流, 单位 mA (该模块在 U2M 控制器中无效, U2M 通过其它模块读取电流)
Motor1Current_U2M~ Motor3Current_U2M	电机驱动电流读取	输出端口: 电机驱动电流, 单位 A (该模块只在 U2M 控制器中有效), 电机驱动电流的合理值在 0A~20A 之间, 当电流大于 20A 时, 端口出现过流、短路、过热等故障, 需立即禁止直流电机驱动模块, 否则有可能导致控制器硬件损坏

将需要使用的电机驱动模块拖入到模型中, 通过输入端口输入适当的值, 即可控制与相应管脚相连的直流电机。

## 2.5 控制恒流驱动

点击 ConstantCurrentDriver 子库 (或者 PowerDriverPWM 子库), 如下图所示, 库中包含两个模块, 分别为 ConstantCurrentDriver 模块与 CurrentRead 模块, 各个模块的作用见表 4-8。

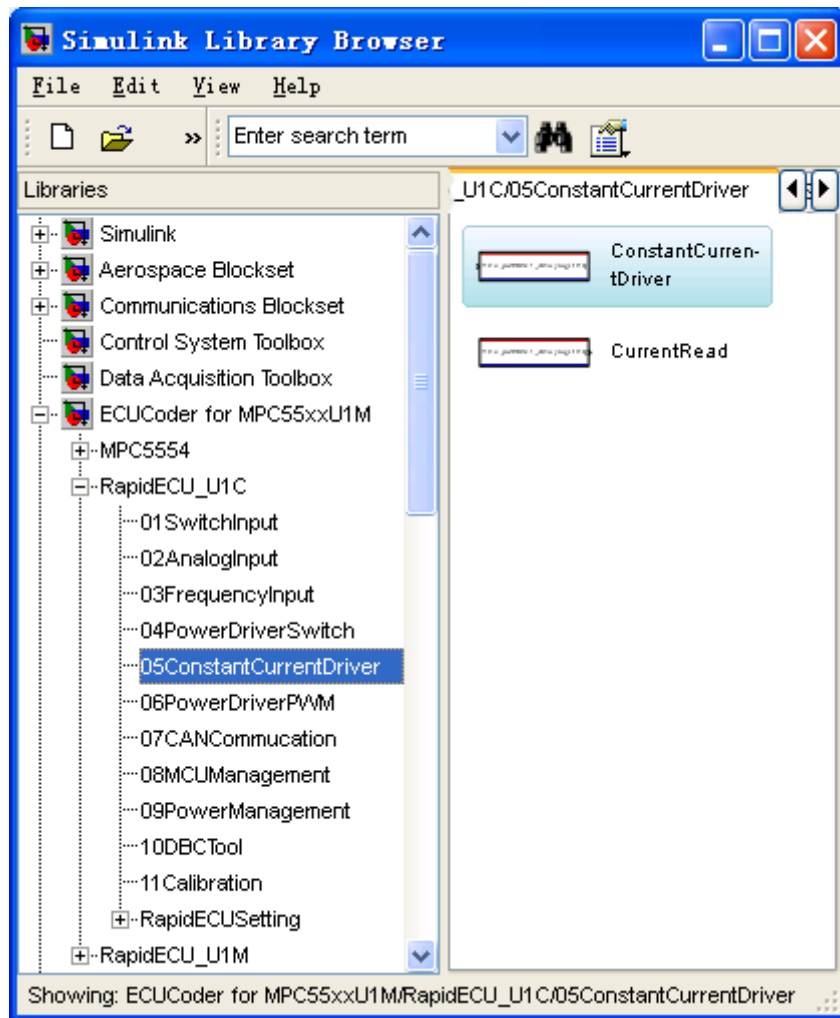


表 4-8 ConstantCurrentDriver 子库

模块名称	描述	备注
ConstantCurrentDriver	恒流驱动	输入端口 1: 目标电流, 单位 mA 输入端口 2: 颤振使能, 1 为使能, 0 为禁止 输入端口 3: 颤振频率, 单位 Hz, 输入值小于等于 PWM 频率的 1/4 输入端口 4: 颤振幅值, 单位 mA
CurrentRead	恒流驱动实际电流读取	输出端口: 实际电流, 单位 mA

将需要使用的模块拖入到模型中, 选择相应的管脚并设置模块参数, 即可控制相应管脚的输出电流。例如, 下图所示为恒流驱动 ConstantCurrentDriver 模块, 双击模块, 出现的对话框如下图所示, 图中各个选项的说明参见表 4-9。

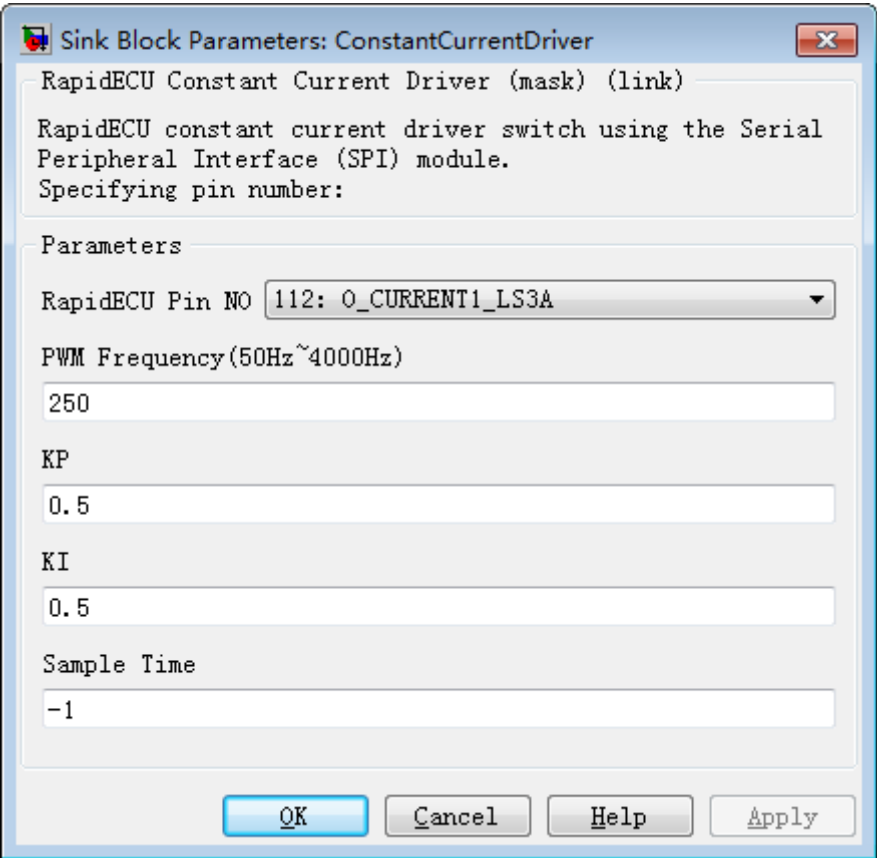


表 4-9 ConstantCurrentDriver 模块参数

选项	选项说明	可选值
RapidECU Pin NO	控制器管脚	控制器上用作恒流驱动的管脚编号
PWM Frequency	PWM 频率	50Hz~4000Hz
KP 、 KI	PI 控制比例与积分参数	自定义
Sample Time	采样时间	自定义

管脚命名方式如下图：



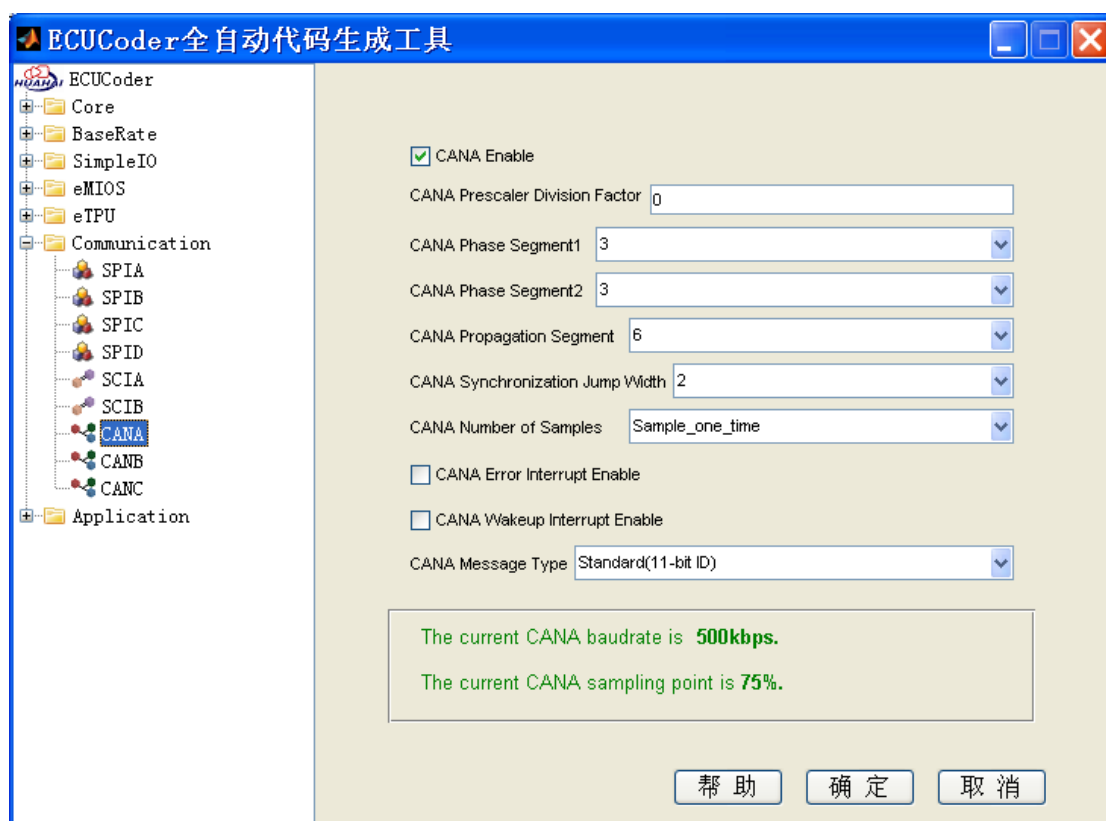
## 五、 基本 CAN 通信与 LIN 通信

### 1. CAN 模块初始化设置

#### 1.1 设置波特率

控制器 CAN 模块默认设置波特率为 500kbps，采样点为 75%，如控制器所在 CAN 总线网络的波特率为 500kbps，则无需再设置波特率。如波特率为其它数值，则需要设置波特率，具体设置方法举例如下。

双击 RapidECUSetting 模块，出现控制器设置图形化界面，从左边的树中选择 Commucation-CANA 选项，出现的界面如下图所示。



影响波特率的选项包括 Prescaler Division Factor( PRESDIV )、Phase Segment1( PSEG1 )、Phase Segment2( PSEG2 )与 Propagation Segment( PROPSEG )四个选项。波特率与采样点设置结果在界面右下方以绿色字体表示。通过选择各个选项的值，可以得到想要的波特率与采样点。

波特率与采样点计算公式如下：

$$\text{波特率} = 8000 / ( \text{PRESDIV} + 1 ) / ( \text{PSEG1} + \text{PSEG2} + \text{PROPSEG} + 4 )$$

$$\text{采样点} = ( \text{PSEG1} + \text{PROPSEG} + 3 ) / ( \text{PSEG1} + \text{PSEG2} + \text{PROPSEG} + 4 )$$

波特率设置例子 1(500k, 75%):

Prescaler Division Factor(PRES DIV): 0

Phase Segment1(PSEG1): 3

Phase Segment2(PSEG2): 3

Propagation Segment(PROPSEG): 6

设置结果: 波特率为 500kbps, 采样点为 75%。

波特率设置例子 2(250k, 75%):

Prescaler Division Factor(PRES DIV): 1

Phase Segment1(PSEG1): 3

Phase Segment2(PSEG2): 3

Propagation Segment(PROPSEG): 6

设置结果: 波特率为 250kbps, 采样点为 75%。

## 1.2 其它设置

影响 CAN 模块初始化配置的其余选项如表 5-1 所示, 可以根据需要进行相应设置。

表 5-1 CAN 初始化设置选项

选项	选项说明	可选值
Synchronization Jump Width	同步跳转宽度	1、2、3、4
Number of Samples	采样数	1 次采样、3 次采样
Error Interrupt Enable	错误中断使能	禁止、使能
Wakeup Interrupt Enable	唤醒中断使能	禁止、使能
Message Type	报文类型	标准帧、扩展帧

## 2. CAN 通信模块库

点击 CANCommucation 子库, 如下图所示, 库中包含 5 个模块, 分别为 BusOffandErrorInterrupt 模块、CANReceive 模块、CANReceiveAll 模块 CANTransmit 模块与 ReceiveandTransmitInterrupt 模块, 各个模块的作用参见表 5-2。

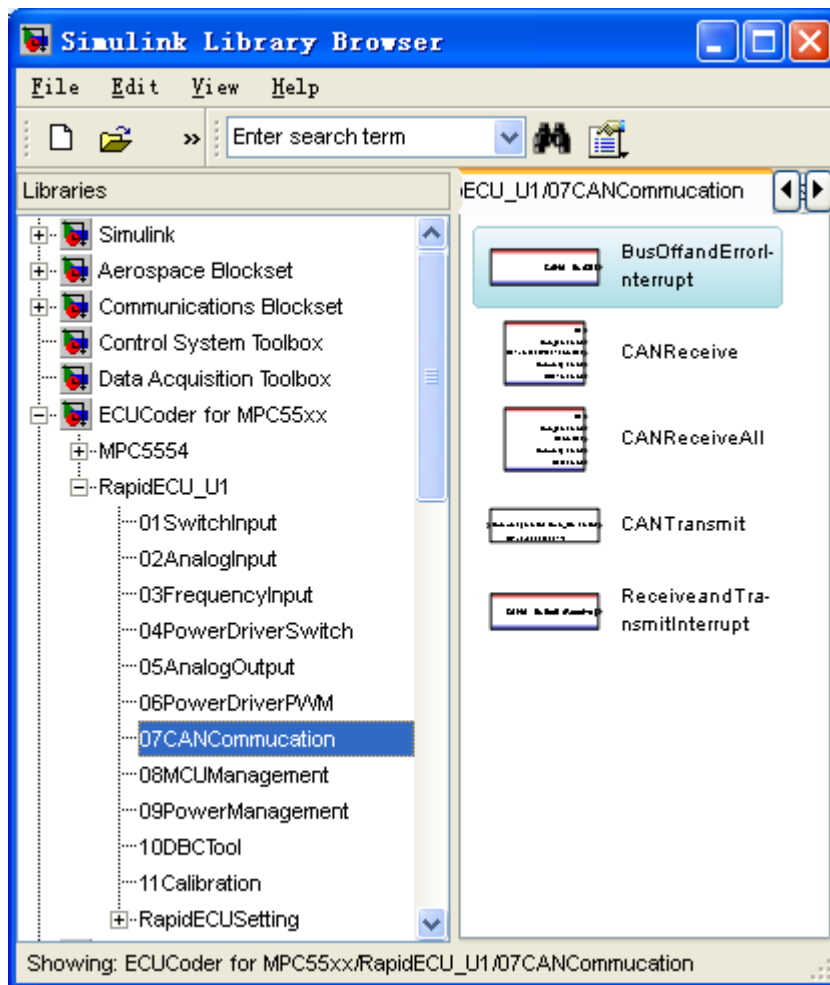


表 5-2 CANCommucation 子库

模块名称	描述
BusOffandErrorInterrupt	总线关闭中断与错误中断模块
CANReceive	CAN 接收模块（指定 ID）
CANReceiveAll	CAN 接收模块（任意 ID）
CANTransmit	CAN 发送模块
ReceiveandTransmitInterrupt	接收中断与发送中断模块

### 3. 发送 CAN 报文

将 CANTransmit 模块拖入到模型中，并进行必要的设置，可用于发送 CAN 报文。双击 CANTransmit 模块，出现的对话框如下图所示，图中各个选项的说明参见表 5-3。

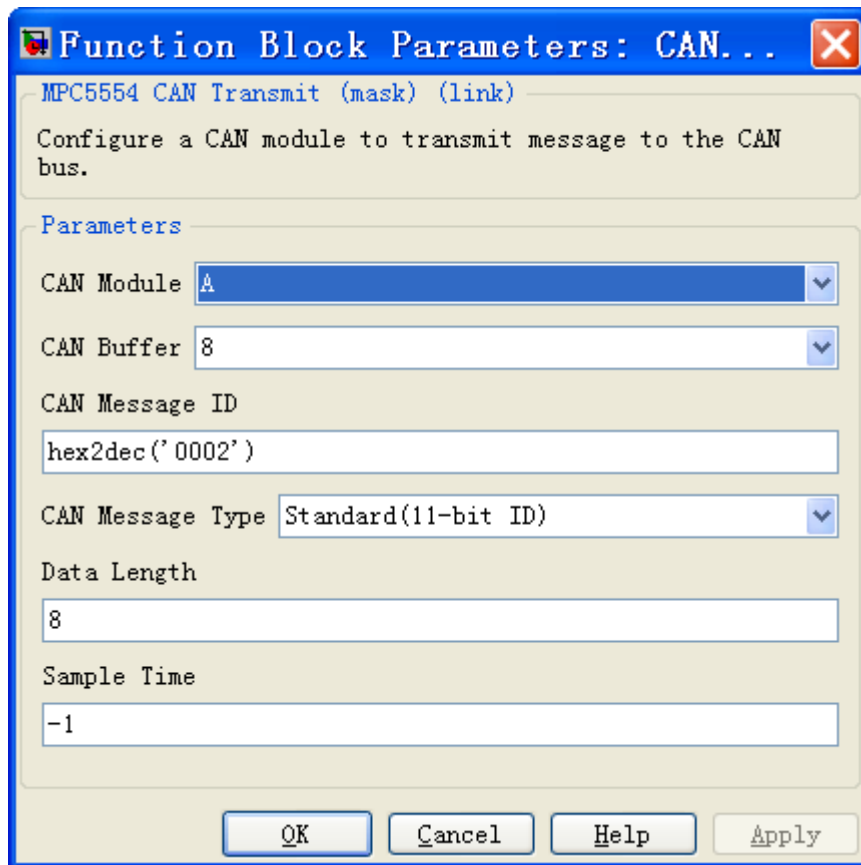
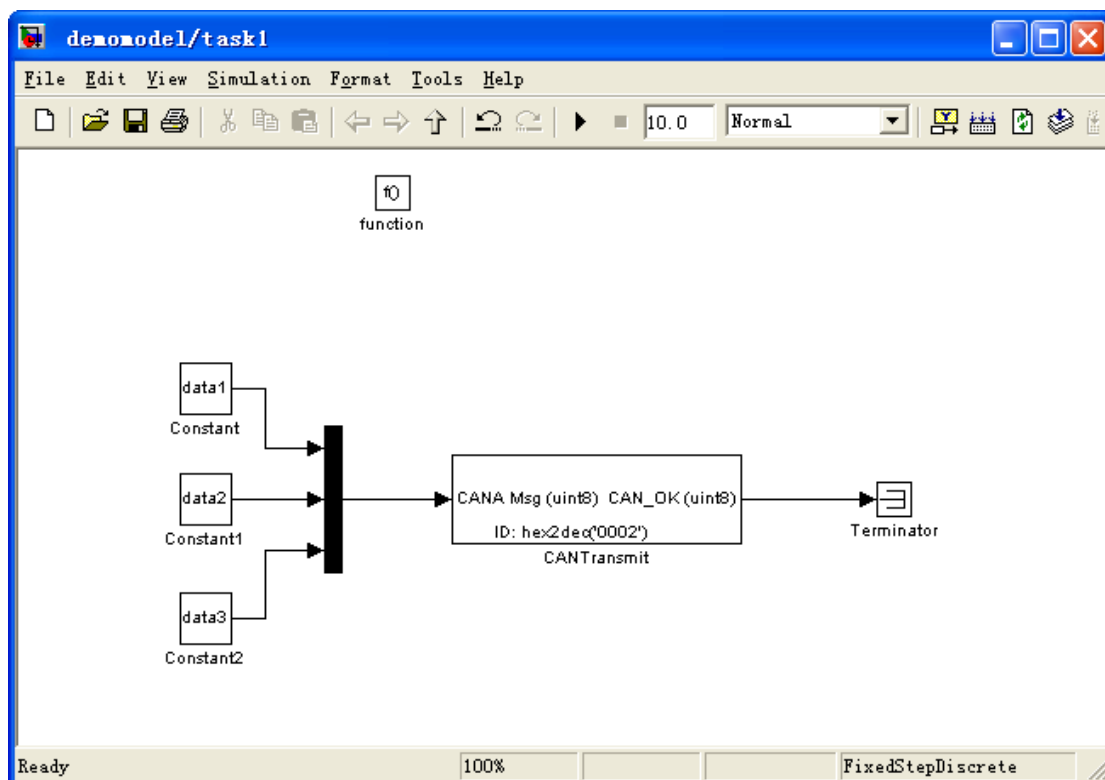
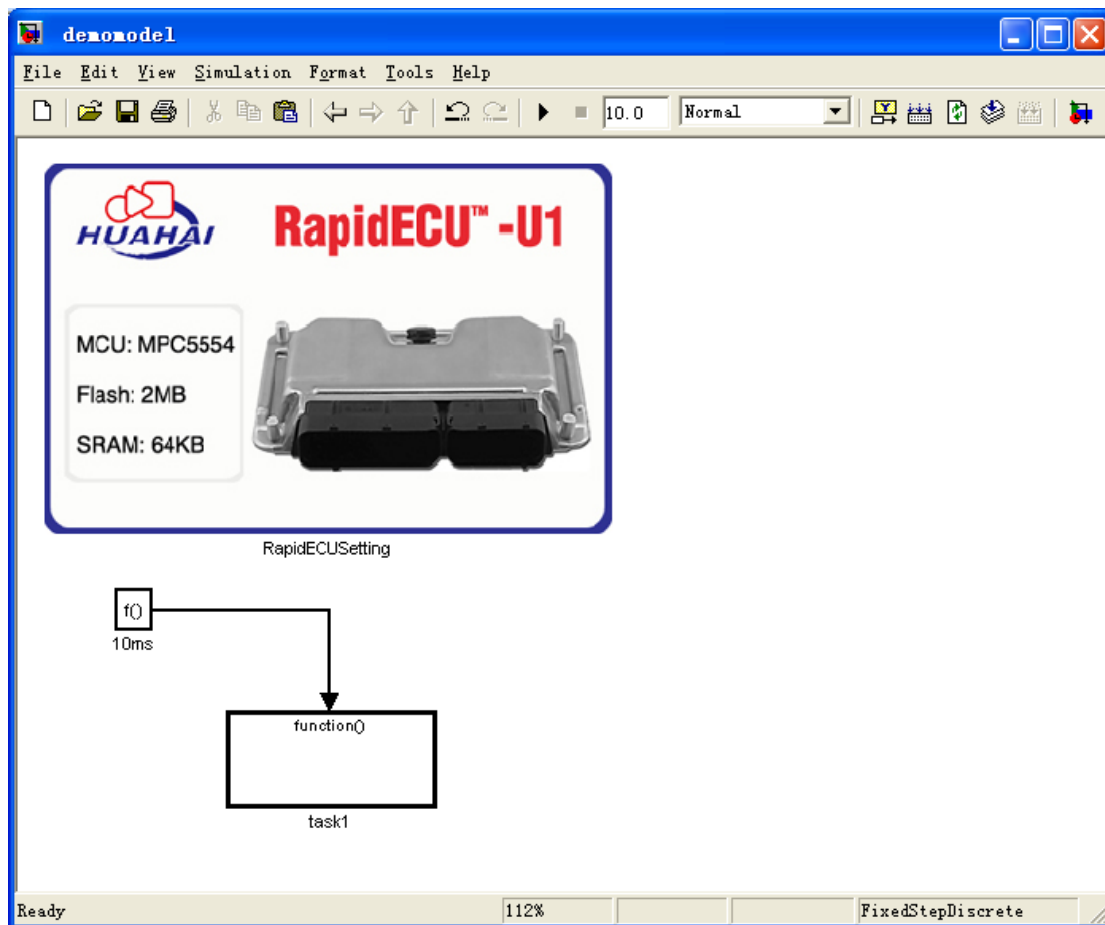


表 5-3 CANTransmit 模块参数

选项	选项说明	可选值
CAN Module	CAN 模块	A、、 B、 C
CAN Buffer	CAN 缓存	8~13、 15、 56~63
CAN Message ID	CAN 报文 ID	ID（十六进制）
CANB Message Type	报文类型	标准帧、扩展帧
Data Length	数据长度	1~8
Sample Time	采样时间	自定义

利用 CAN A 模块的 Buffer 8 发送 ID 为 0x0002，数据长度为 3 的 CAN 报文（标准帧）的设置如上图所示，发送间隔为 0.01s。模型如下图所示，模型输入与输出端口的数据类型都为 uint8，模块输出为 1 时表明发送成功。





## 4. 接收指定 ID 的 CAN 报文

### 4.1 定时接收 CAN 报文

将 CANReceive 模块拖入到模型中，并进行必要的设置，可用于接收 CAN 报文。双击 CANReceive 模块，出现的对话框如下图所示，图中各个选项的说明参见表 5-4。

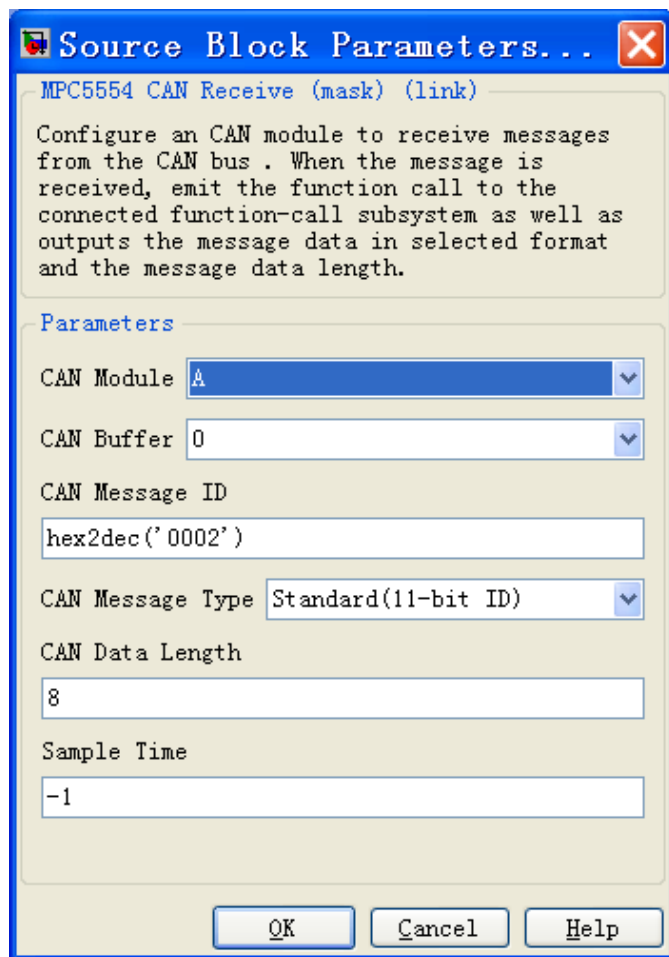


表 5-4 CANReceive 模块参数

选项	选项说明	可选值
CAN Module	CAN 模块	A、B、C
CAN Buffer	CAN 缓存	0~7、16~55
CAN Message ID	CAN 报文 ID	ID（十六进制）
CAN Message Type	报文类型	标准帧、扩展帧
CAN Data Length	数据长度	1~8
Sample Time	采样时间	自定义

利用 CAN A 模块的 Buffer 0 接收 ID 为 0x0002，数据长度为 8 的 CAN 报文的设置，如上图所示，接收间隔为 0.01s。模型如下图所示，模块各个端口的说明参见表 5-5。

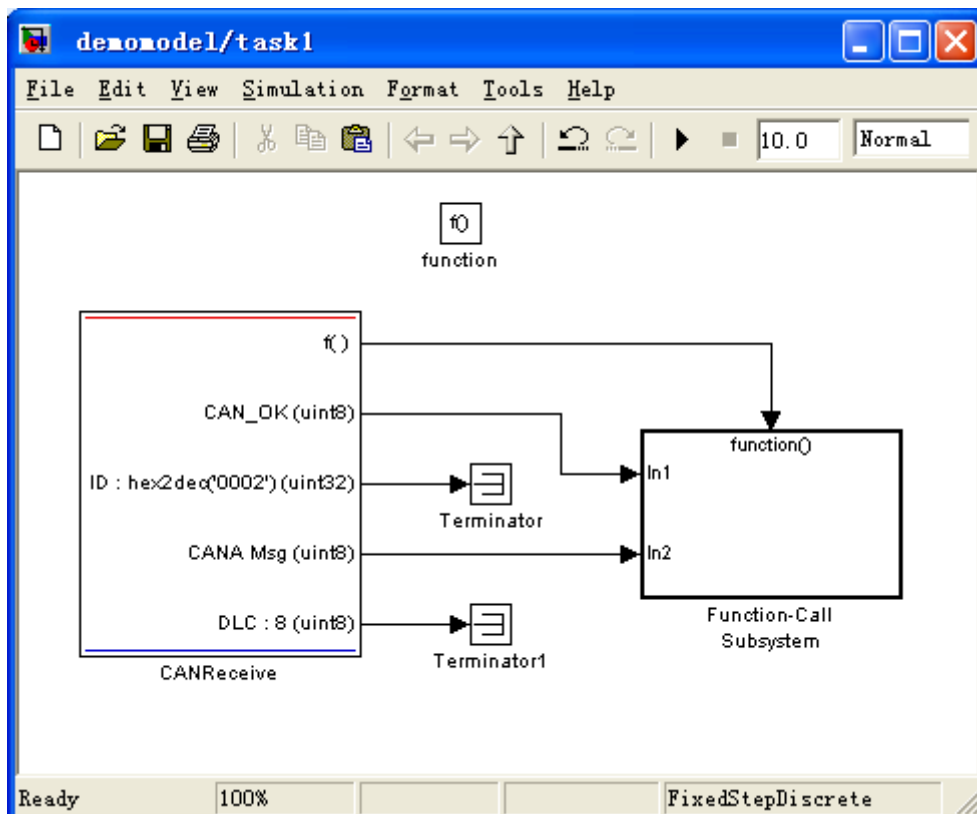
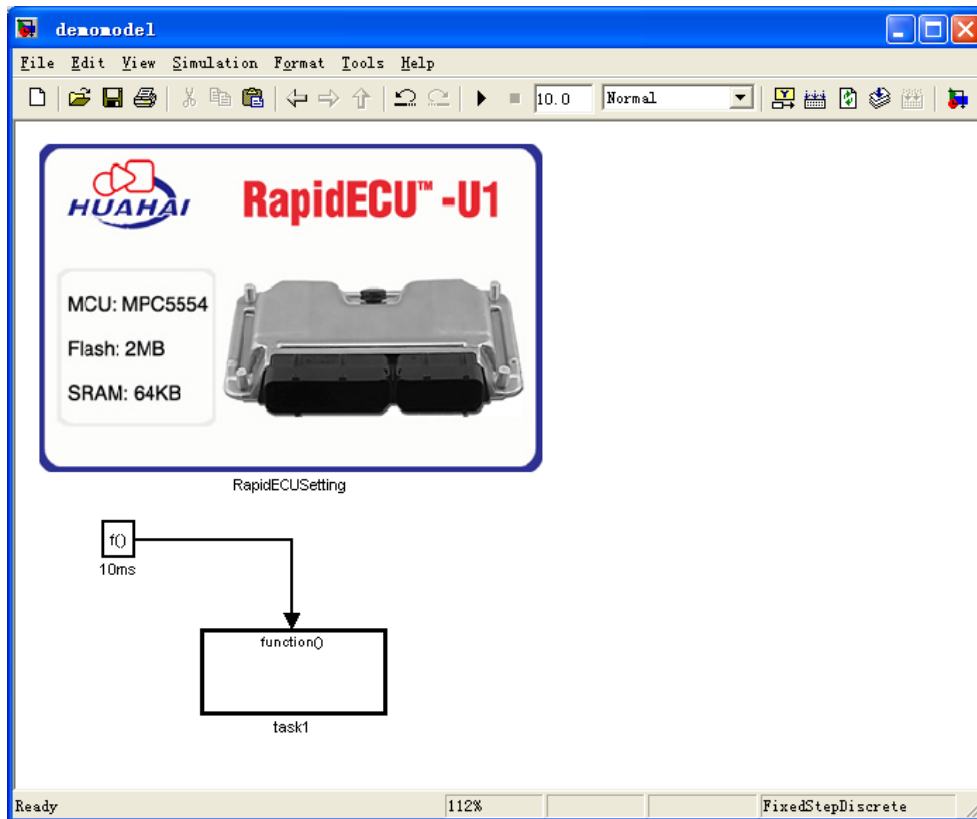
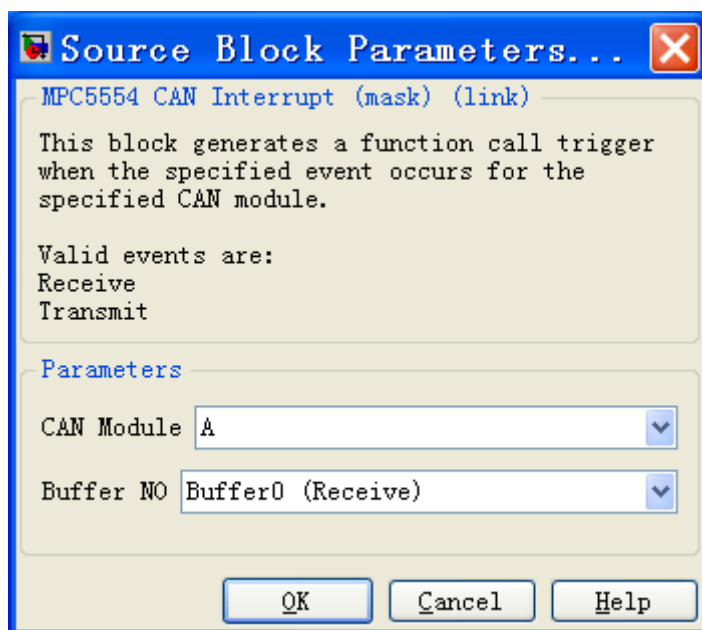


表 5-5 CANReceive 模块端口

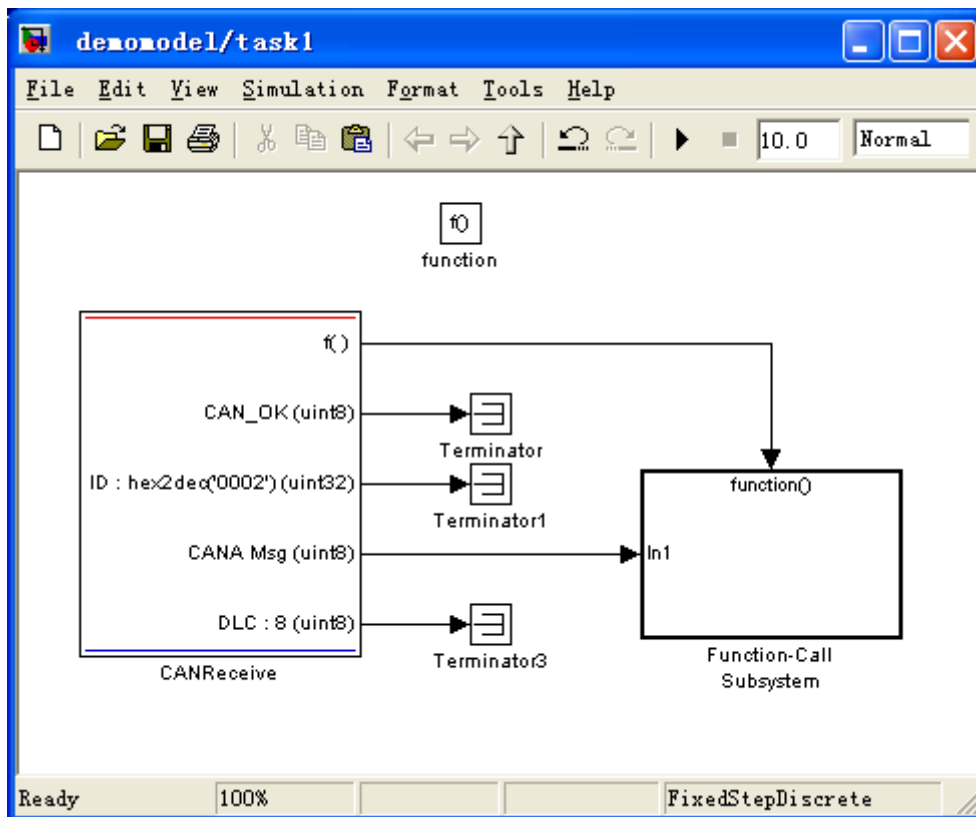
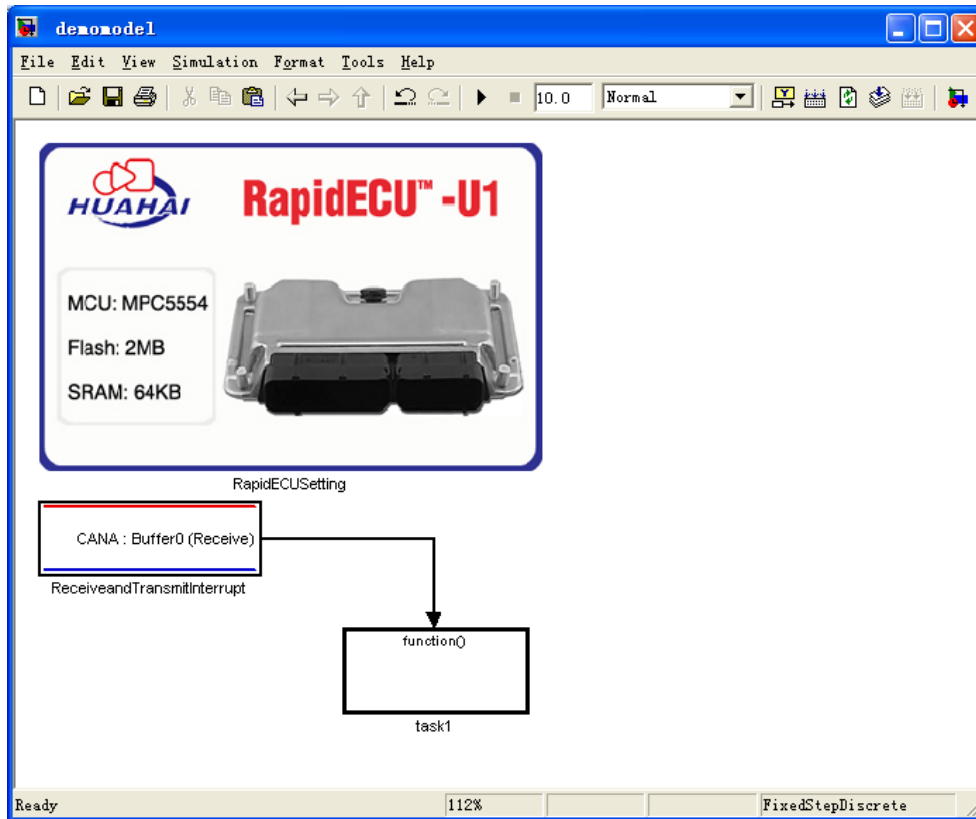
端口	端口说明	数据类型
f()	触发	无
CAN_OK	是否接收到 CAN 报文，接收到为 1，否则为 0	uint8
ID	CAN 报文 ID	uint32
CAN Msg	CAN 报文数据	uint8
DLC	数据长度	uint8

## 4.2 中断接收 CAN 报文

将 ReceiveandTransmitInterrupt 模块拖入到模型顶层，并进行必要的设置，可用于中断接收 CAN 报文。双击 ReceiveandTransmitInterrupt 模块，出现的对话框如下图所示，两个选项分别用于选择 CAN 模块与 Buffer。



利用 CAN A 模块的 Buffer 0 中断接收 CAN 报文的设置如上图所示，模型如下图所示。



## 5. 接收任意 ID 的 CAN 报文

### 5.1 定时接收 CAN 报文

将 CANReceiveAll 模块拖入到模型中, 并进行必要的设置, 可用于接收 CAN 报文。双击 CANReceiveAll 模块, 出现的对话框如下图所示, 图中各个选项的说明参见表 5-6。

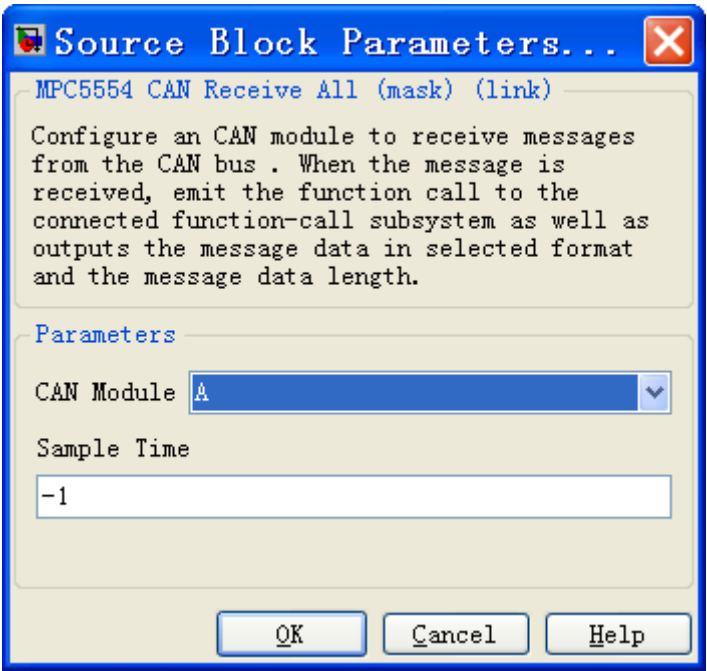


表 5-6 CANReceiveAll 模块参数

选项	选项说明	可选值
CAN Module	CAN 模块	A、B、C
Sample Time	采样时间	自定义

利用 CAN A 模块接收任意 ID 的 CAN 报文的设置如上图所示, 接收间隔为 0.01s。模型如下图所示, 模块各个端口的说明参见表 5-7。

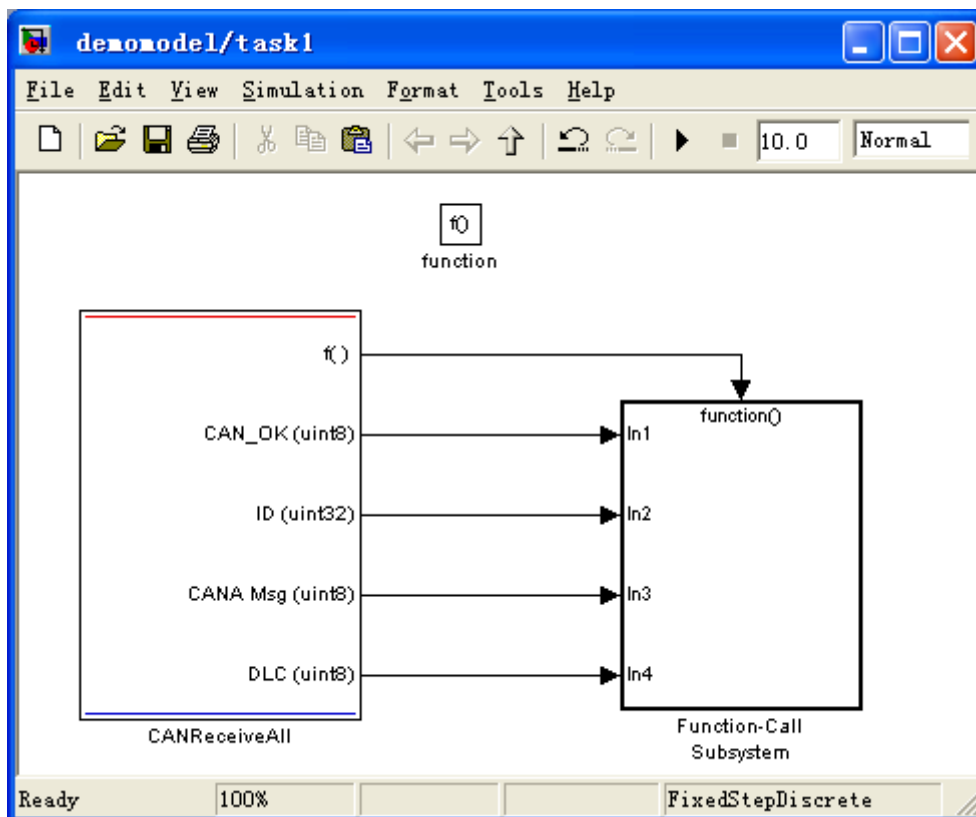
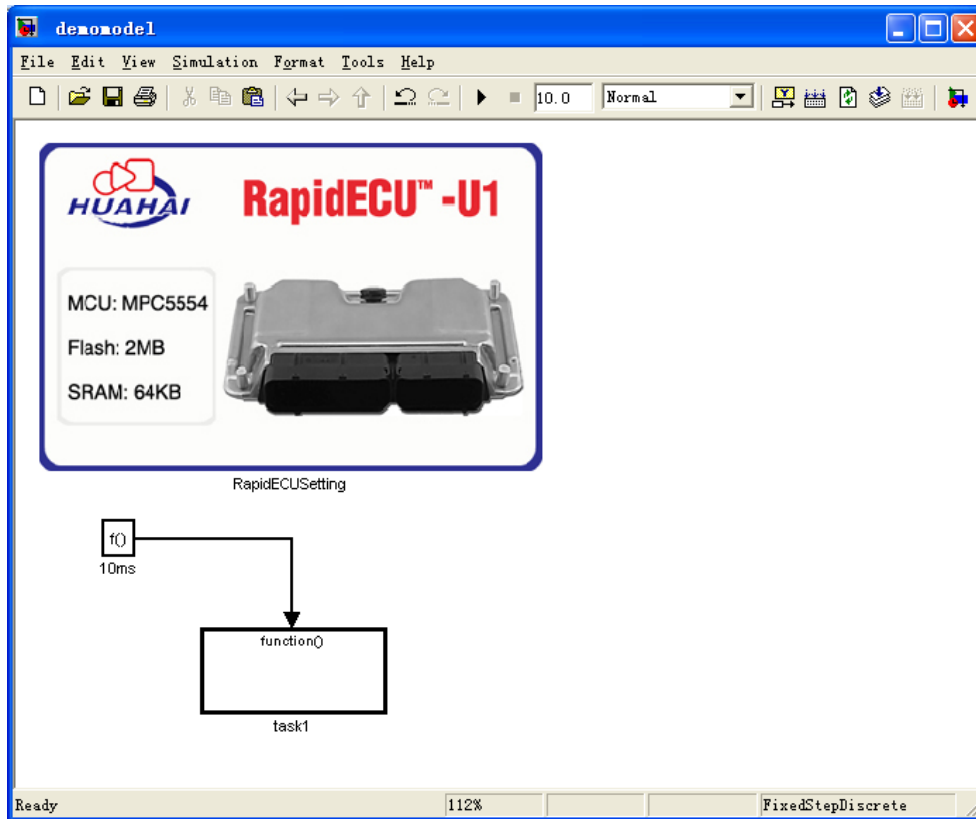
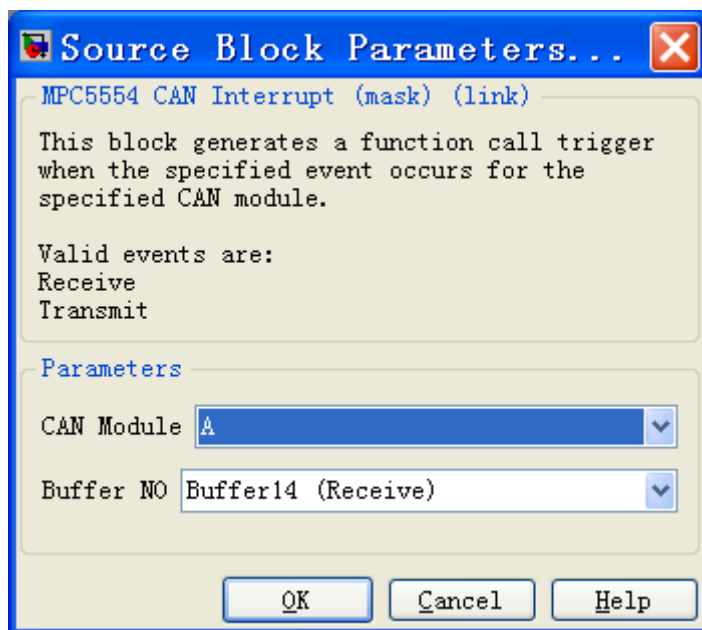


表 5-7 CANReceive 模块端口

端口	端口说明	数据类型
f()	触发	无
CAN_OK	是否接收到 CAN 报文，接收到为 1，否则为 0	uint8
ID	CAN 报文 ID	uint32
CAN Msg	CAN 报文数据	uint8
DLC	数据长度	uint8

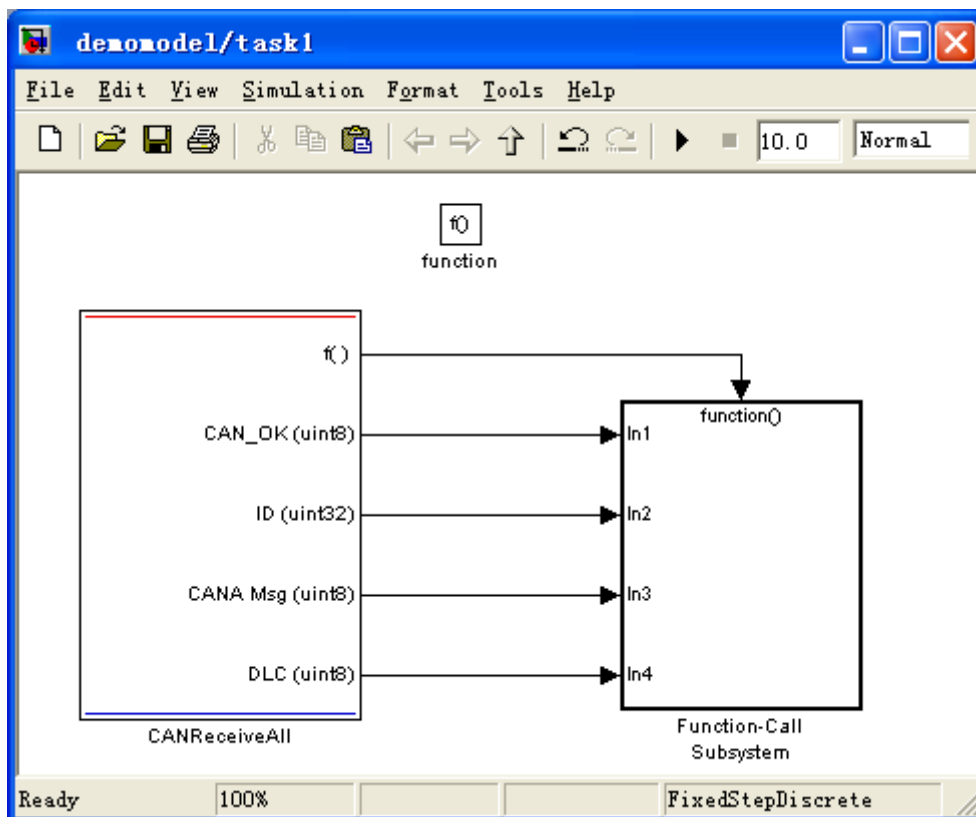
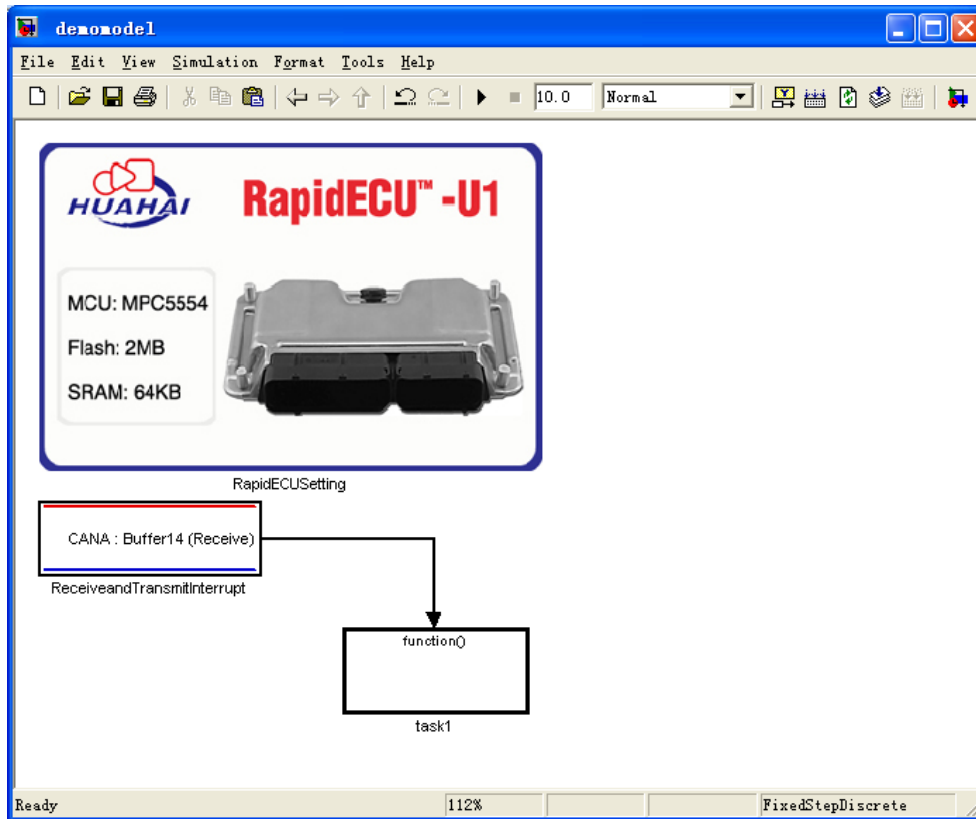
## 5.2 中断接收 CAN 报文

将 ReceiveandTransmitInterrupt 模块拖入到模型中，并进行必要的设置，可用于中断接收 CAN 报文。双击 ReceiveandTransmitInterrupt 模块，出现的对话框如下图所示，两个选项分别用于选择 CAN 模块与 Buffer，此处 Buffer NO 选项必须选择 Buffer14。



利用 CAN A 模块中断接收 CAN 报文的设置如上图所示，模型如下图所示。

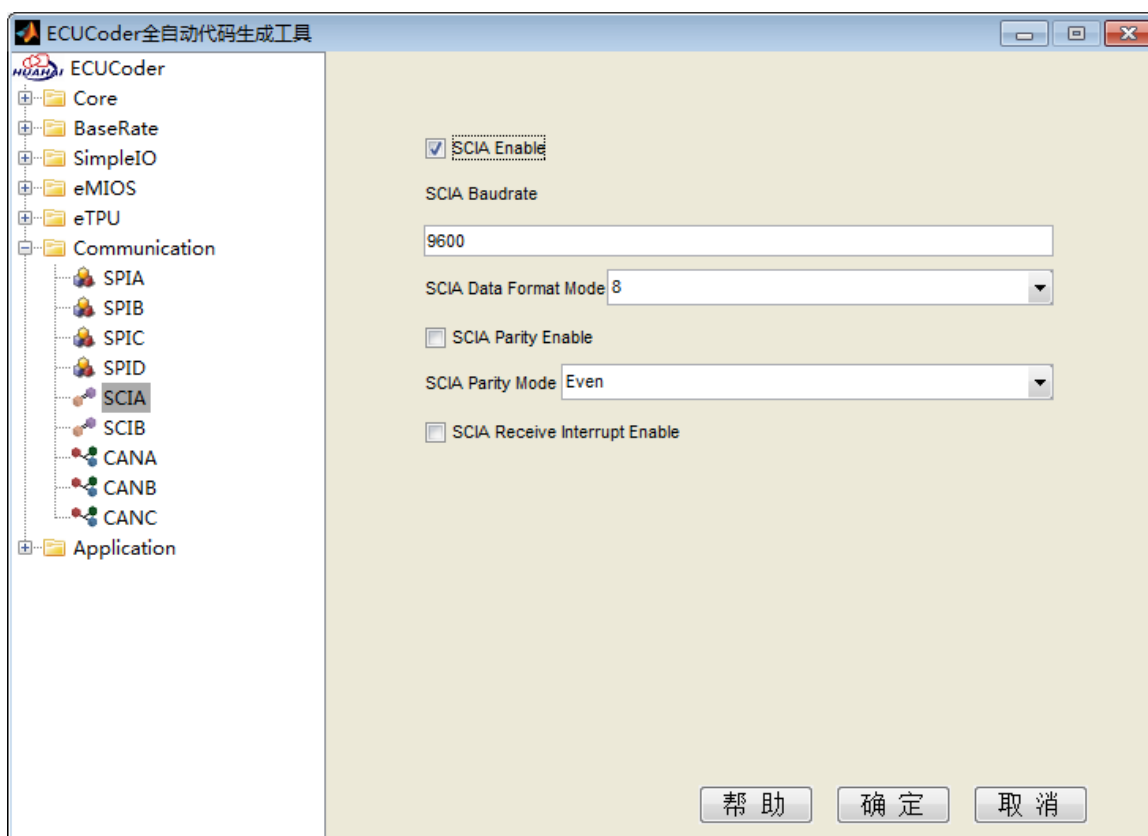




## 6. LIN 模块初始化设置

控制器 LIN 模块默认设置为禁止，如果需要使用 LIN 模块，首先需要使能 LIN 模块。双击 RapidECUSetting 模块，出现控制器设置图形化界面，从左边的树中选择 Commucation-SCIA 选项，出现的界面如下图所示，勾选其中的 SCIA Enable 复选框以使能控制器 LIN 模块。在 SCIA Baudrate 编辑框中输入 LIN 总线波特率，此处默认设置为 9600，即波特率为 9600bps。

控制器 LIN 模块只支持主机工作模式，不支持从机工作模式。



## 7. LIN 通信模块库

点击 LINCommucation 子库，如下图所示，库中包含 3 个模块，分别为 LINRXEnable 模块、LINReceive 模块、LINTransmit 模块，各个模块的作用参见表 5-8。

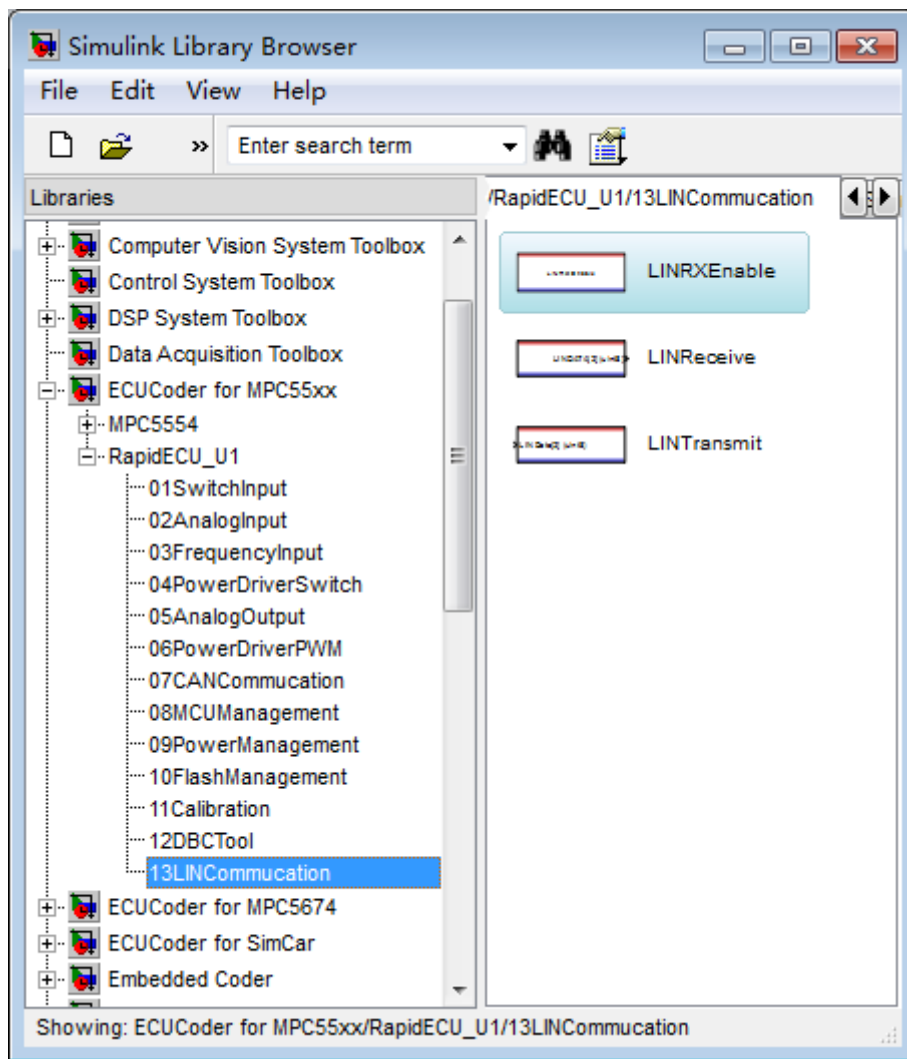
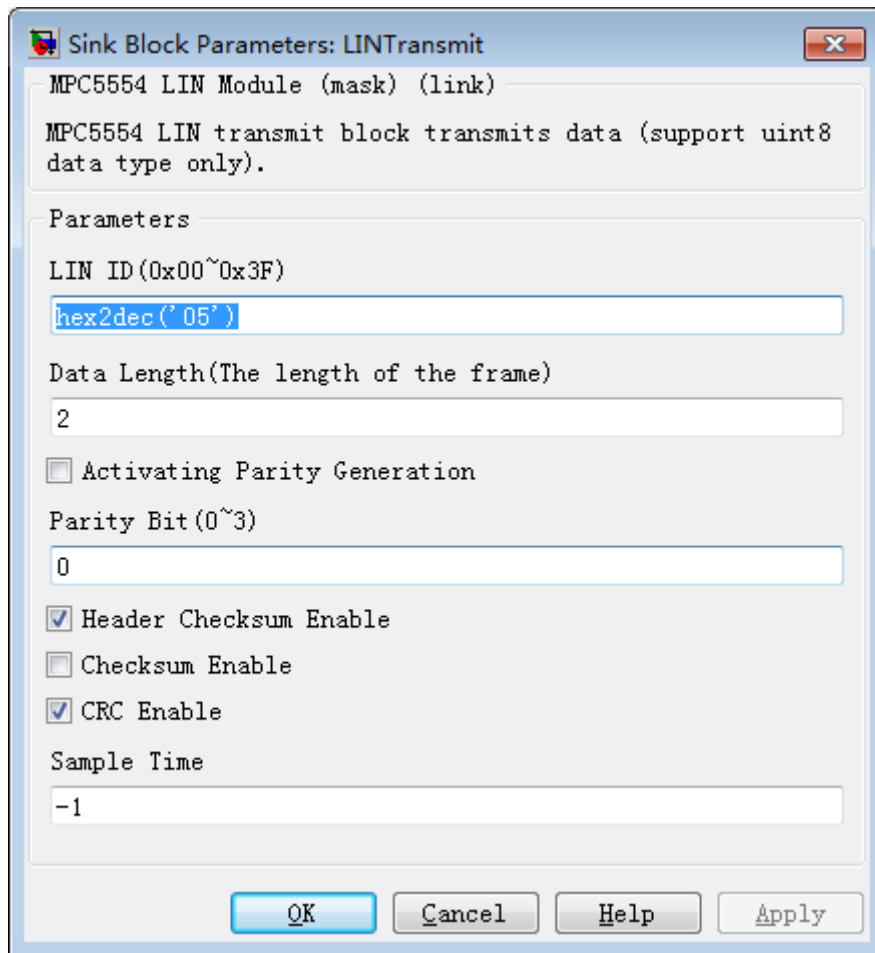


表 5-8 LINCommucation 子库

模块名称	描述
LINRXEnable	LIN 总线接收功能使能
LINReceive	LIN 接收模块
LINTransmit	LIN 发送模块

## 8. 发送 LIN 报文

将 LINTransmit 模块拖入到模型中，并进行必要的设置，可用于发送 LIN 报文。双击 LINTransmit 模块，出现的对话框如下图所示，图中各个选项的说明参见表 5-9。



**Sink Block Parameters: LINTransmit**

MPC5554 LIN Module (mask) (link)

MPC5554 LIN transmit block transmits data (support uint8 data type only).

Parameters

LIN ID(0x00~0x3F)

Data Length(The length of the frame)

☐ Activating Parity Generation

Parity Bit(0~3)

☒ Header Checksum Enable

☐ Checksum Enable

☒ CRC Enable

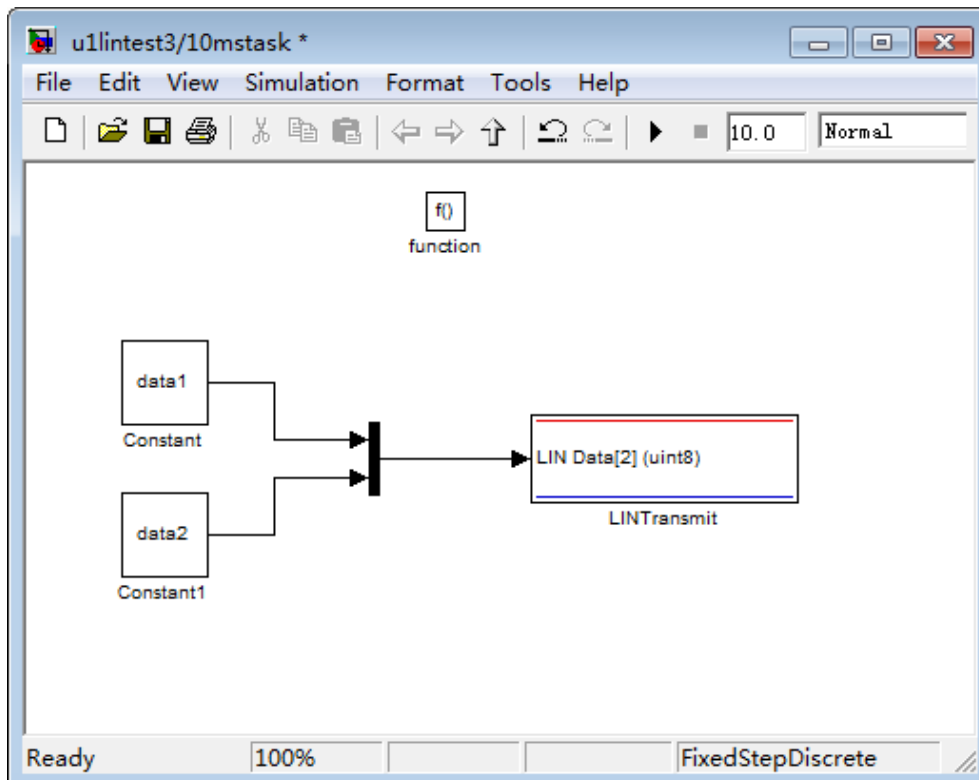
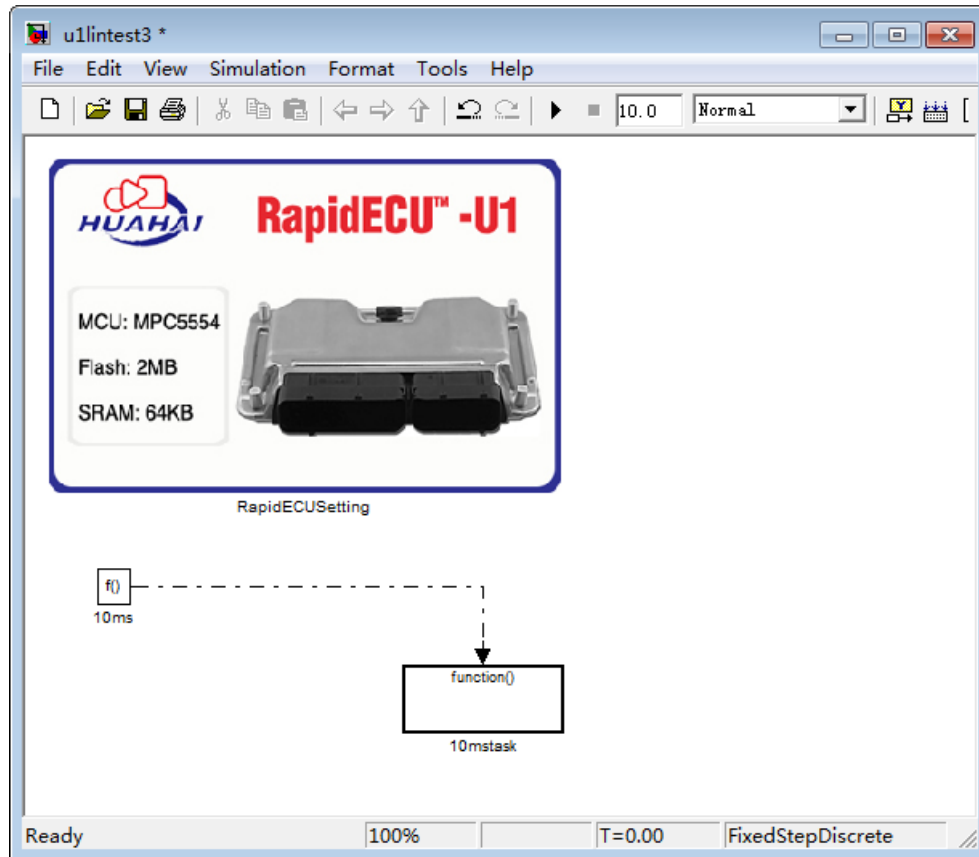
Sample Time

OK Cancel Help Apply

表 5-9 LINTransmit 模块参数

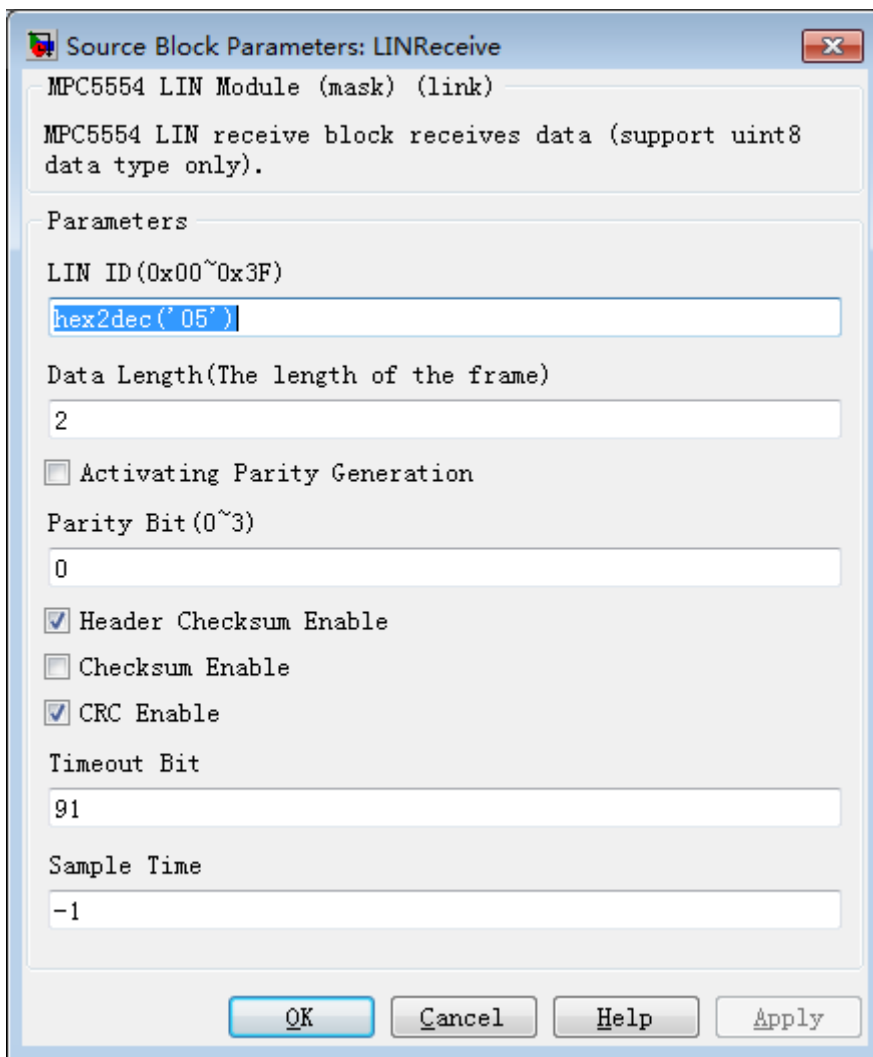
选项	选项说明	可选值
LIN ID	LIN 总线 ID	0x00~0x3F，其中 0x3C~0x3F 保留作为命令与扩展帧标识符
Data Length	数据长度	0~255，LIN 总线规范中一般使用 2、4 或者 8
Activating Parity Generation	报文头中校验位自动生成	禁止、使能
Parity Bit	校验位	0~3，如果勾选了 Activating Parity Generation，则忽略本选项
Header Checksum Enable	报文头校验和使能	禁止、使能
Checksum Enable	校验和使能	禁止、使能
CRC Enable	CRC 使能	禁止、使能
Sample Time	采样时间	自定义

利用 LIN 模块发送 ID 为 0x05，数据长度为 2 的 LIN 报文的设置如上图所示，发送间隔为 0.01s。模型如下图所示，模块输入端口的数据类型为 uint8。



## 9. 接收 LIN 报文

为了接收 LIN 报文，首先需要将 LINRXEnable 模块拖入模型顶层，然后将 LINReceive 模块拖入到模型中，并进行必要的设置，用于接收 LIN 报文。双击 LINReceive 模块，出现的对话框如下图所示，图中各个选项的说明参见表 5-10。



The dialog box titled "Source Block Parameters: LINReceive" contains the following fields and controls:

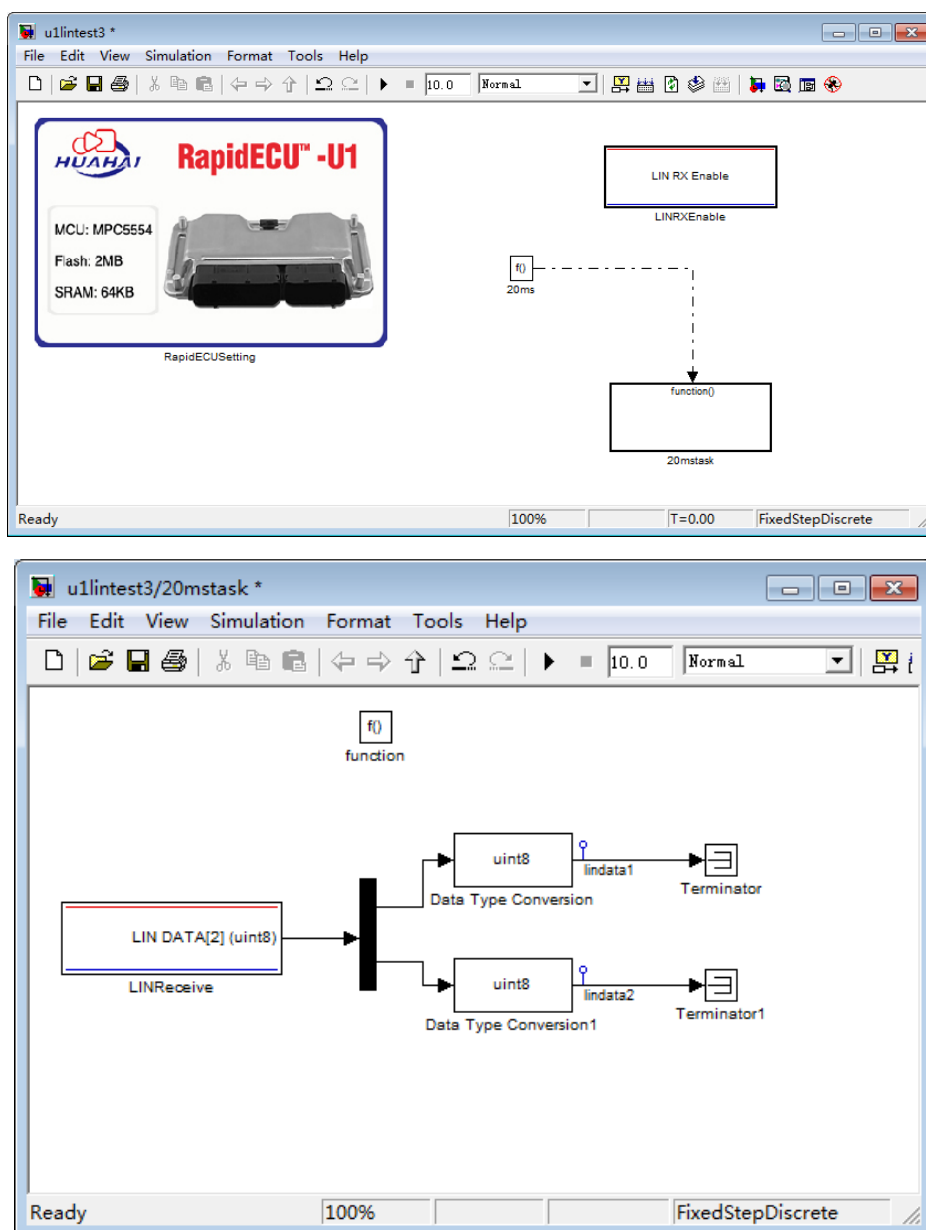
- MPC5554 LIN Module (mask) (link)**: MPC5554 LIN receive block receives data (support uint8 data type only).
- Parameters**:
  - LIN ID (0x00~0x3F)**:
  - Data Length (The length of the frame)**:
  - ☐ **Activating Parity Generation**
  - Parity Bit (0~3)**:
  - ☒ **Header Checksum Enable**
  - ☐ **Checksum Enable**
  - ☒ **CRC Enable**
  - Timeout Bit**:
  - Sample Time**:
- Buttons**: OK, Cancel, Help, Apply.

表 5-10 LINReceive 模块参数

选项	选项说明	可选值
LIN ID	LIN 总线 ID	0x00~0x3F，其中 0x3C~0x3F 保留作为命令与扩展帧标识符
Data Length	数据长度	0~255，LIN 总线规范中一般使用 2、4 或者 8
Activating Parity Generation	报文头中校验位自动生成	禁止、使能

Parity Bit	校验位	0~3, 如果勾选了 Activating Parity Generation, 则忽略本选项
Header Checksum Enable	报文头校验和使能	禁止、使能
Checksum Enable	校验和使能	禁止、使能
CRC Enable	CRC 使能	禁止、使能
Timeout Bit	接收超时位	0~1023
Sample Time	采样时间	自定义

利用 LIN 模块接收 ID 为 0x05, 数据长度为 2 的 LIN 报文的设置如上图所示, 接收间隔为 0.02s。模型如下图所示, 模块输出端口的数据类型为 uint8。



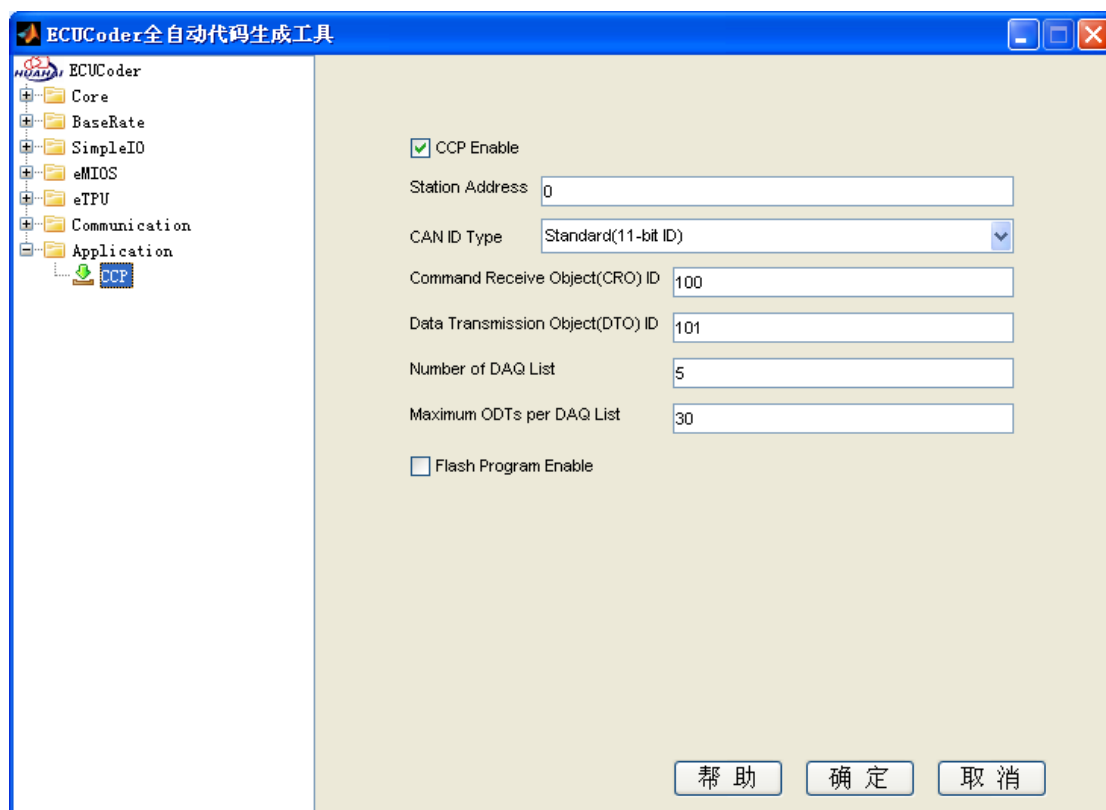
## 六、 测量与标定

为了优化控制算法，一般需要对控制器参数进行测量与标定。为满足这个要求，首先需要设置 CCP 参数，然后需要在模型中添加测量量与标定量。

### 1. 设置 CCP 参数

控制器 CCP 模块默认设置 Station Address 为 0，标准帧，CRO ID 为 100（十六进制），DTO ID 为 101（十六进制），CCP 模块设置应与用户对控制器所要求的 CCP 设置保持一致，设置方法如下。

双击 RapidECUSetting 模块，出现控制器设置图形化界面，从左边的树状菜单中选择 Application-CCP 选项，出现的界面如下图所示，可根据需要修改相关参数。



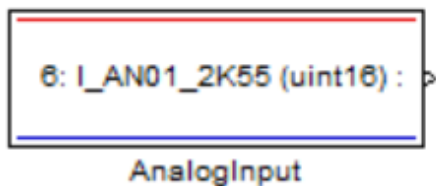
### 2. 添加测量量

当需要在上位机测量标定软件中观察一个变量时，就需要在模型中添加该测量量，添加完之后，该测量量会自动添加到模型所生成的 A2L 文件中供测量标定软件调用。

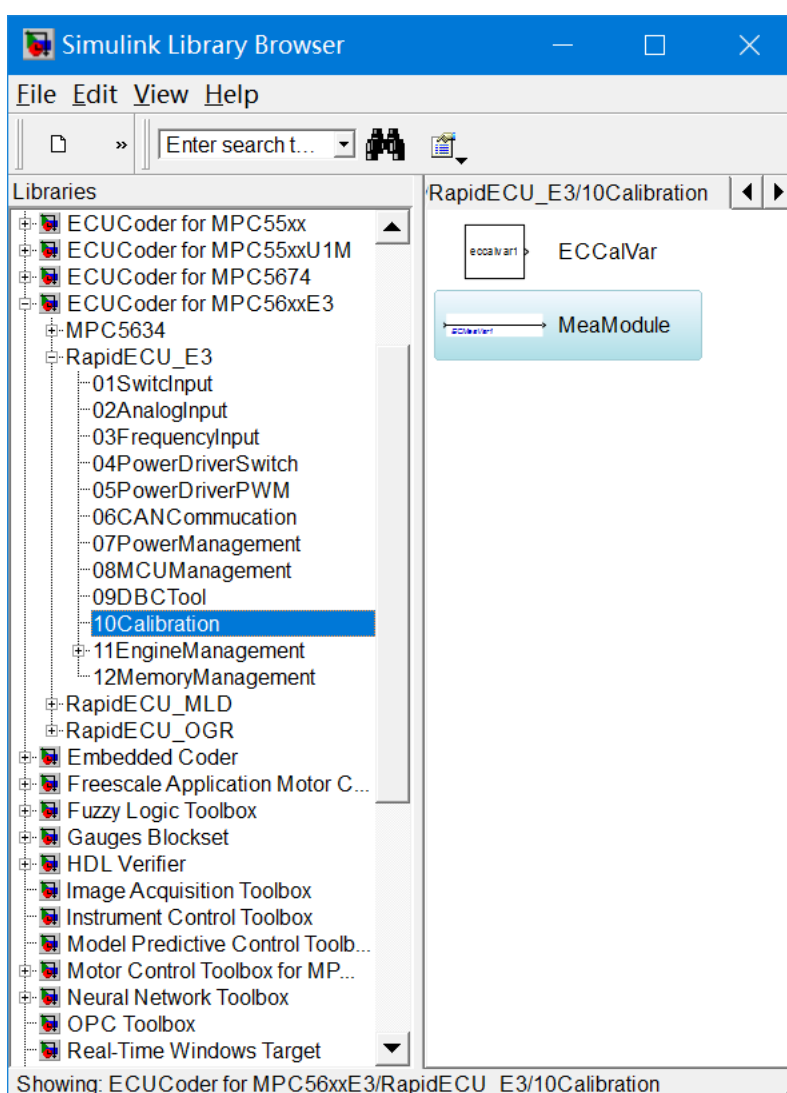
#### 2.1 添加测量量：使用模块方式

如下图所示，需要实时观察管脚 6 的输入。

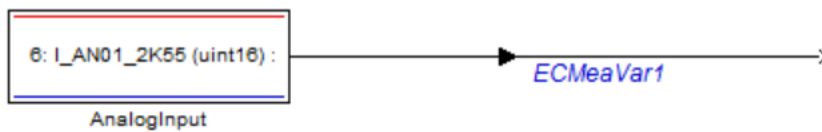




点击 **Calibration** 子库，如下图所示，库中包含 1 个名为 **MeaModule** 的模块，该模块的参数可以实时在线测量。



将 **MeaModule** 模块拖入模型中，并与需要的模块相连接，如下图所示。双击模块，可以按照 MATLAB 变量命名规则对参数任意命名。



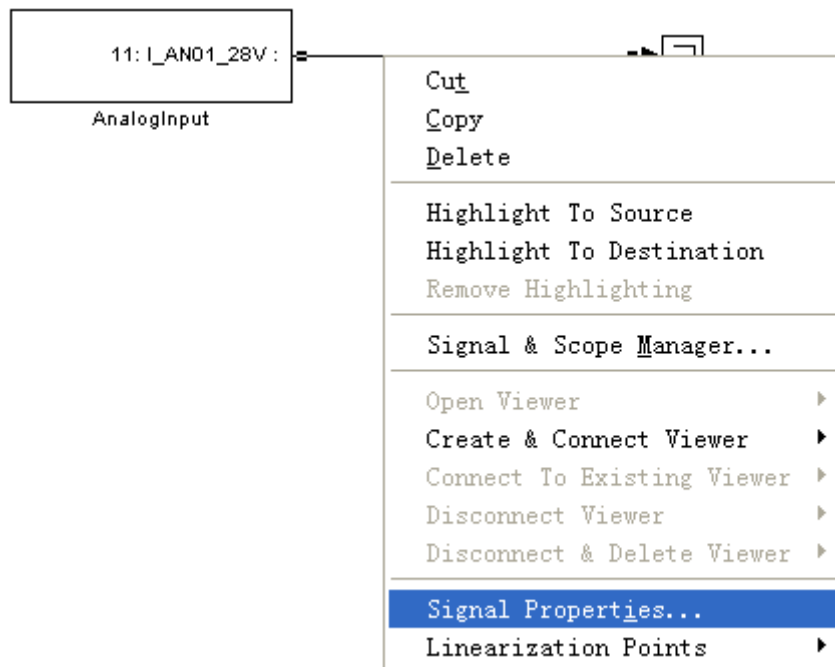
在之后自动生成的 A2L 文件中，会出现变量名为该模块参数名的信号的具体描述，如下图所示。

```

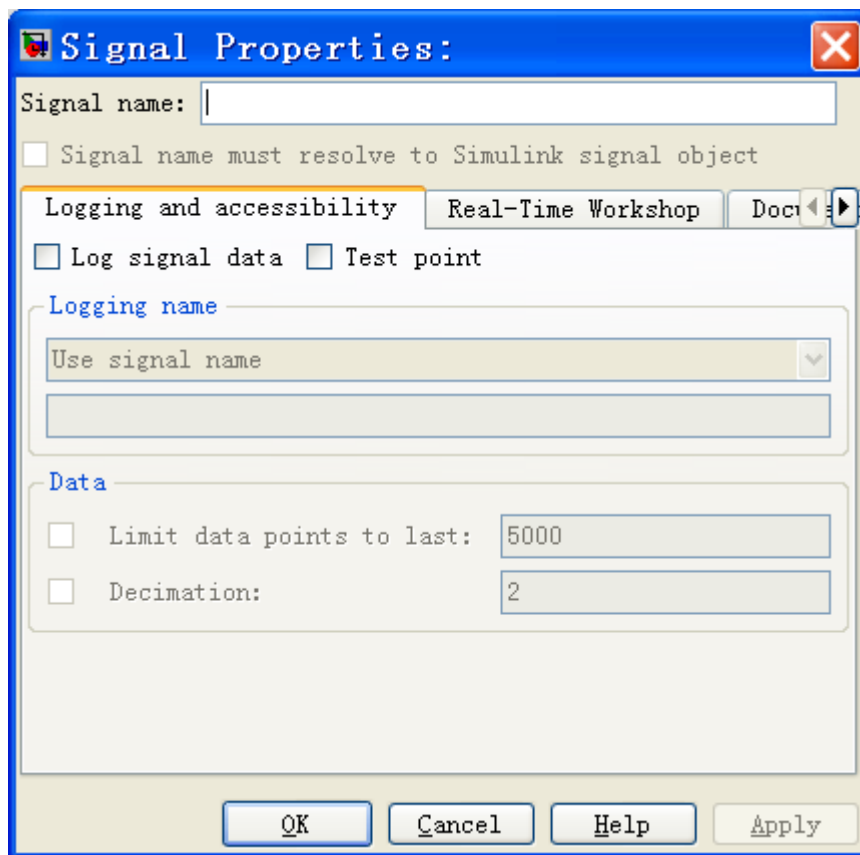
/begin MEASUREMENT
/* Name                */      ECMeaVar1
/* Long identifier     */      ""
/* Data type           */      UWORD
/* Conversion method   */      Chip5634Test2_CM_uint16
/* Resolution (Not used) */      0
/* Accuracy (Not used) */      0
/* Lower limit         */      0
/* Upper limit         */      65535
ECU_ADDRESS              0x4000186c
/end MEASUREMENT
    
```

## 2.2 添加测量量：使用信号属性方式

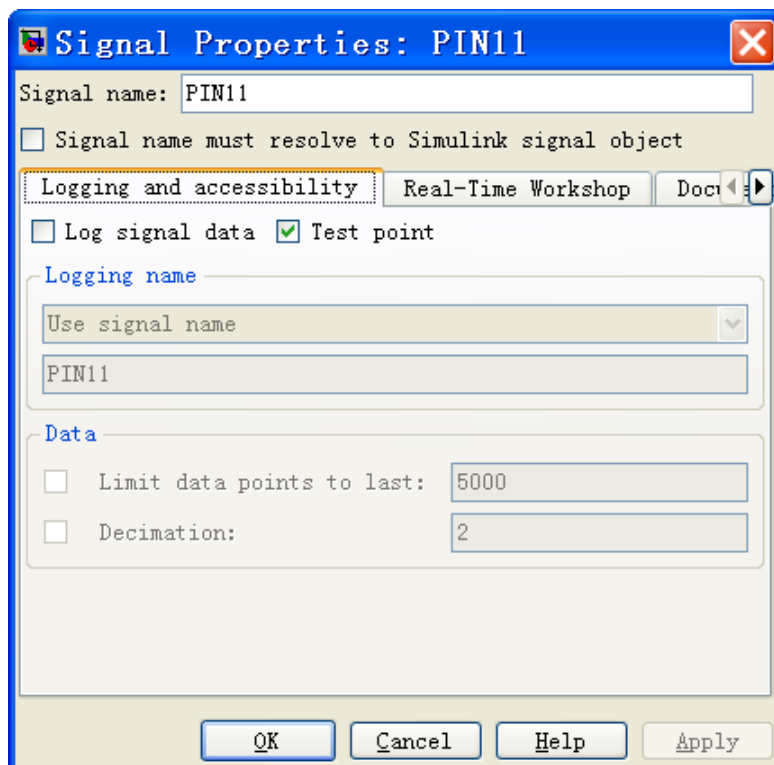
鼠标右击要测量信号所在的信号线，在出现的菜单中选择 **Signal Properties**，如下图所示。



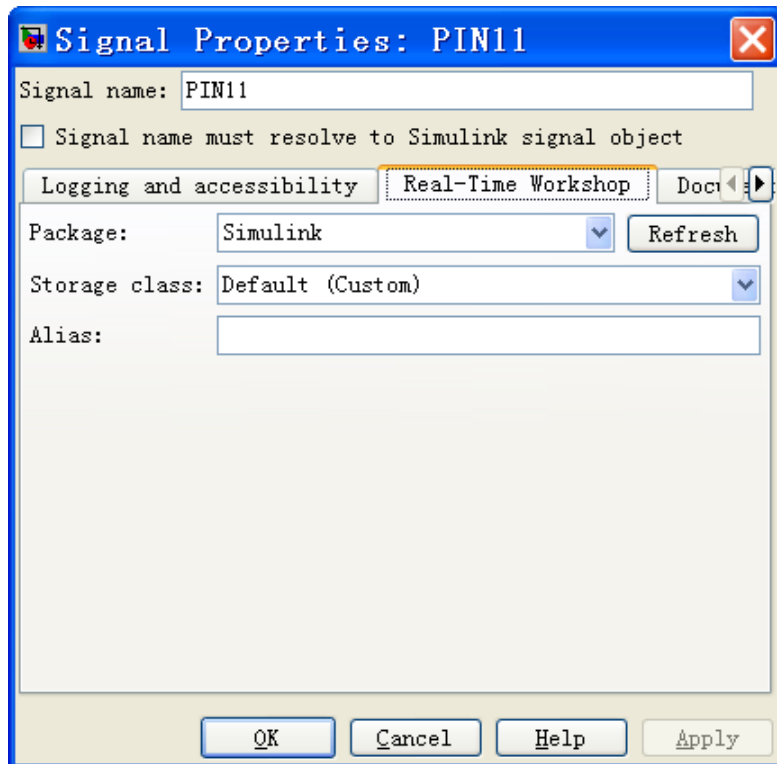
选择之后，弹出的对话框如下图所示。



首先，需要在 **Signal name** 框中输入信号的名称，并勾选 **Logging and accessibility** 选项卡的 **Test point** 复选框（也可不勾选，为提高模型可读性，建议勾选），如下图所示。



然后，点击 Real-Time Workshop 选项卡，在 Package 下拉菜单中选择 Simulink，在 Storage class 下拉菜单中选择 Default(Custom)，如下图所示。



设置完成，点击 OK 即完成模型测量信号的添加。

在之后自动生成的 A2L 文件中，会出现添加过的信号的具体描述，如下图所示。

```

/begin MEASUREMENT
/* Name */ PIN11
/* Long identifier */ ""
/* Data type */ UWORD
/* Conversion method */ demomodel_COMPU_METHOD_1
/* Resolution (Not used) */ 0
/* Accuracy (Not used) */ 0
/* Lower limit */ 0
/* Upper limit */ 65535
ECU_ADDRESS 0x40001e96
/end MEASUREMENT
    
```

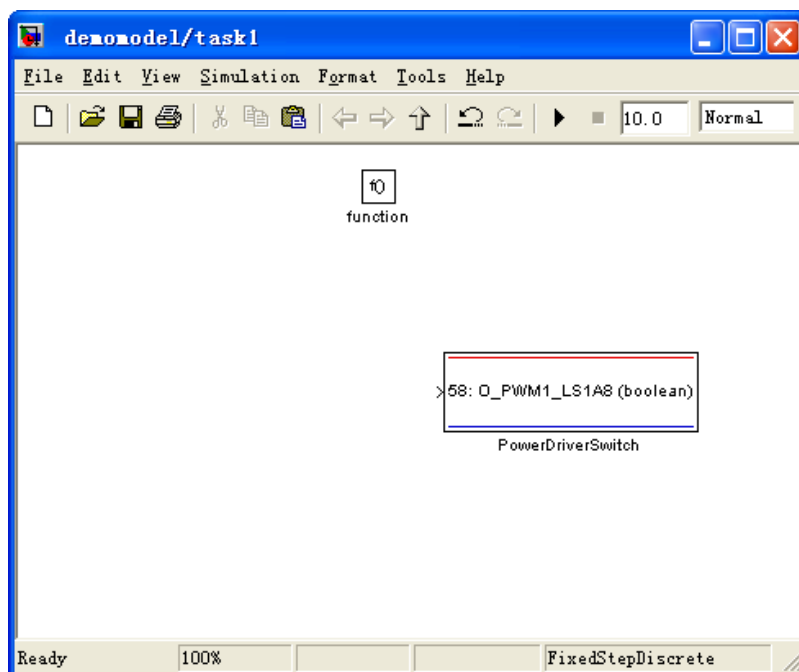
### 3. 添加标定量

当需要在上位机测量标定软件中实时在线调整一个变量时，就需要在模型中添加该标定量，添加完之

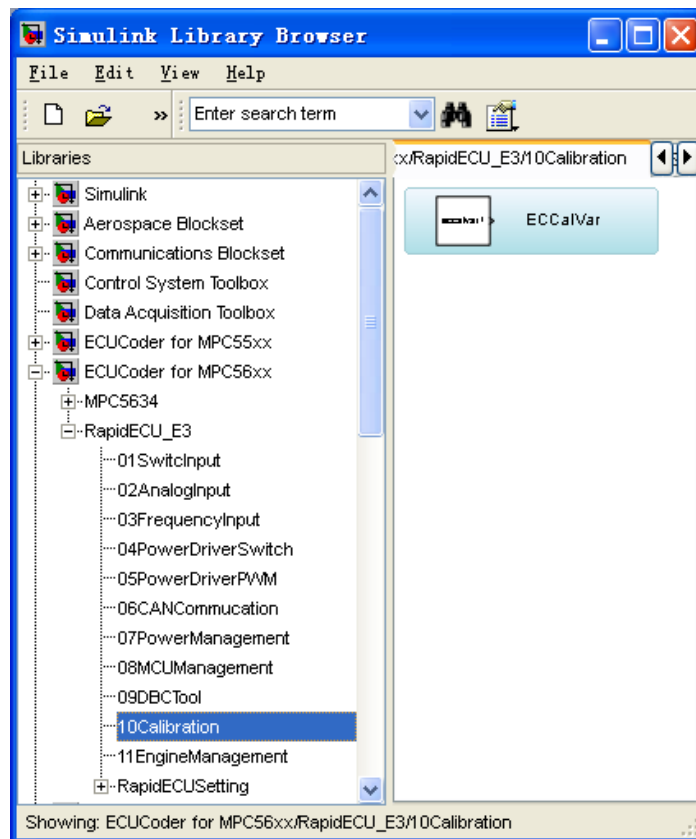
后，该标定量会自动添加到模型所生成的 A2L 文件中供测量标定软件调用。ECUCoder 提供了两种添加标定量的方法，一种是使用 ECUCoder 标定模块，适合于标定变量少，并且需要标定的变量都是标量的情况；另一种是使用 ECUCoder 标定命令，适合于标定变量较多的情况。

## 2.1 添加标量标定量：使用模块方式

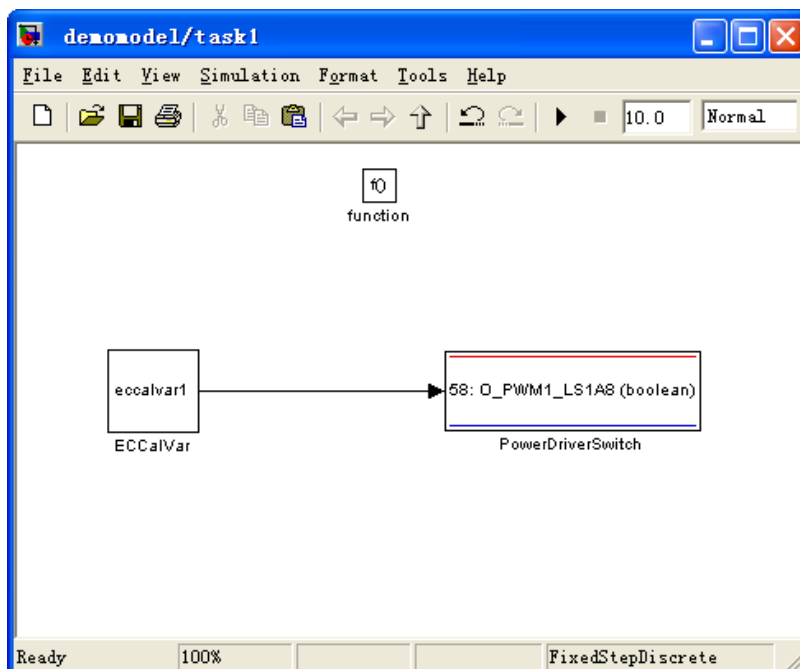
如下图所示，需要实时在线调整管脚 58 的输出。



点击 Calibration 子库，如下图所示，库中包含 1 个名为 ECCalVar 的模块，该模块的参数可以实时在线调整。



将 ECCalVar 模块拖入模型中，并与需要的模块相连接，如下图所示。双击模块，可以按照 MATLAB 变量命名规则对参数任意命名。



在之后自动生成的 A2L 文件中，会出现变量名为该模块参数名的信号的具体描述，如下图所示。

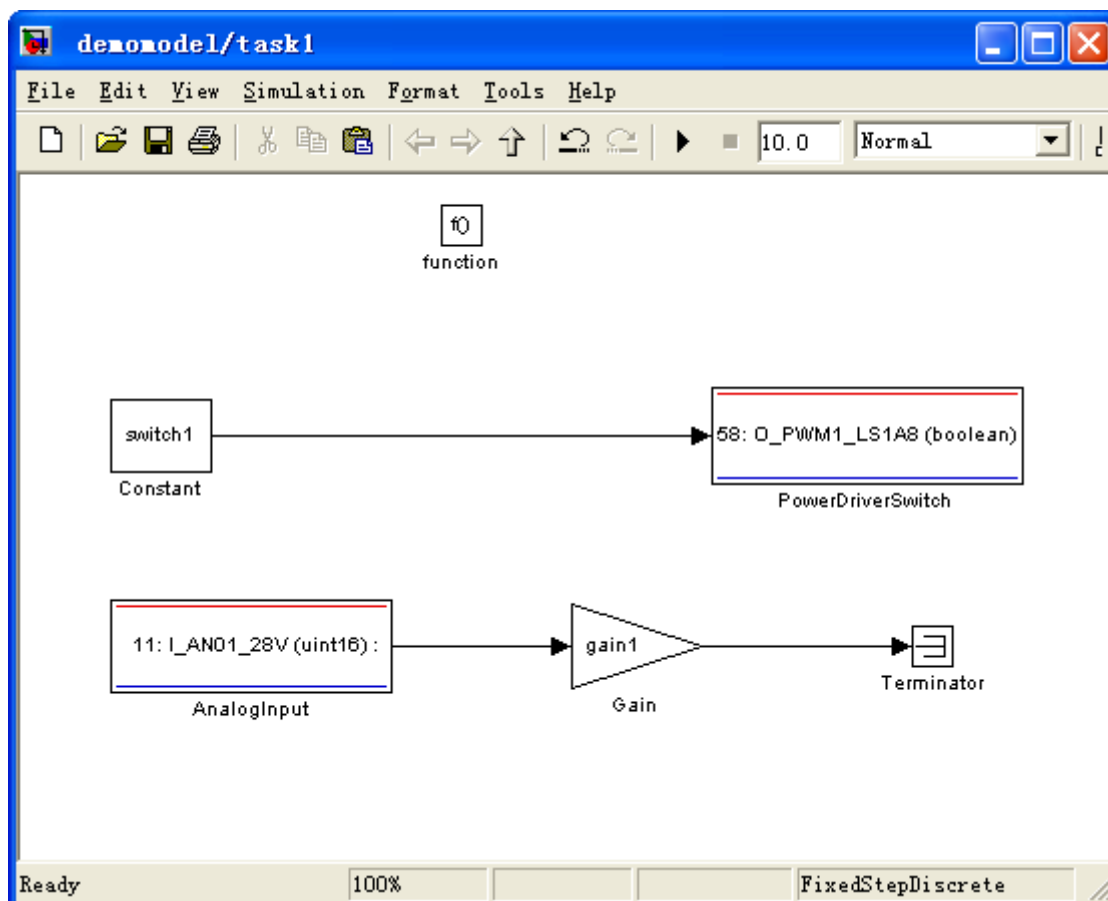
```

/begin CHARACTERISTIC
/* Name */ eccalvar1
/* Long Identifier */ ""
/* Type */ VALUE
/* Memory Address */ 0x40001b90
/* Record Layout */ Scalar_BOOLEAN
/* Maximum Difference */ 0
/* Conversion Method */ demomodel_COMPU_METHOD_1
/* Lower Limit */ 0
/* Upper Limit */ 1
/end CHARACTERISTIC

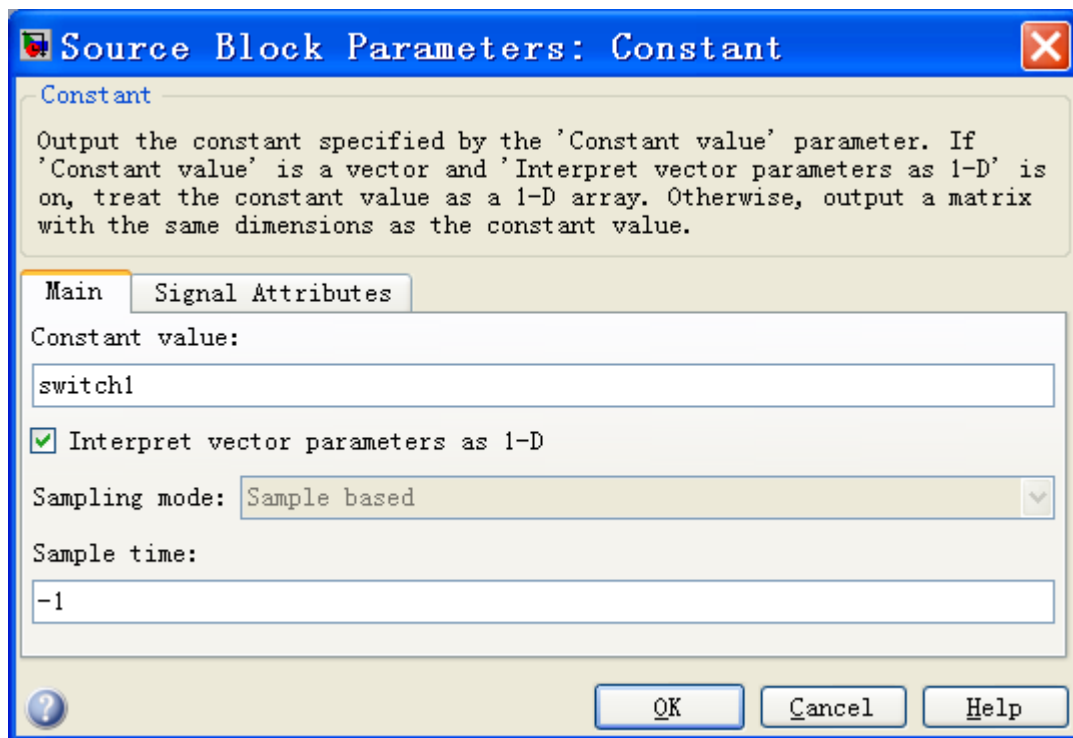
```

## 2.2 添加标量标定量：使用命令行方式

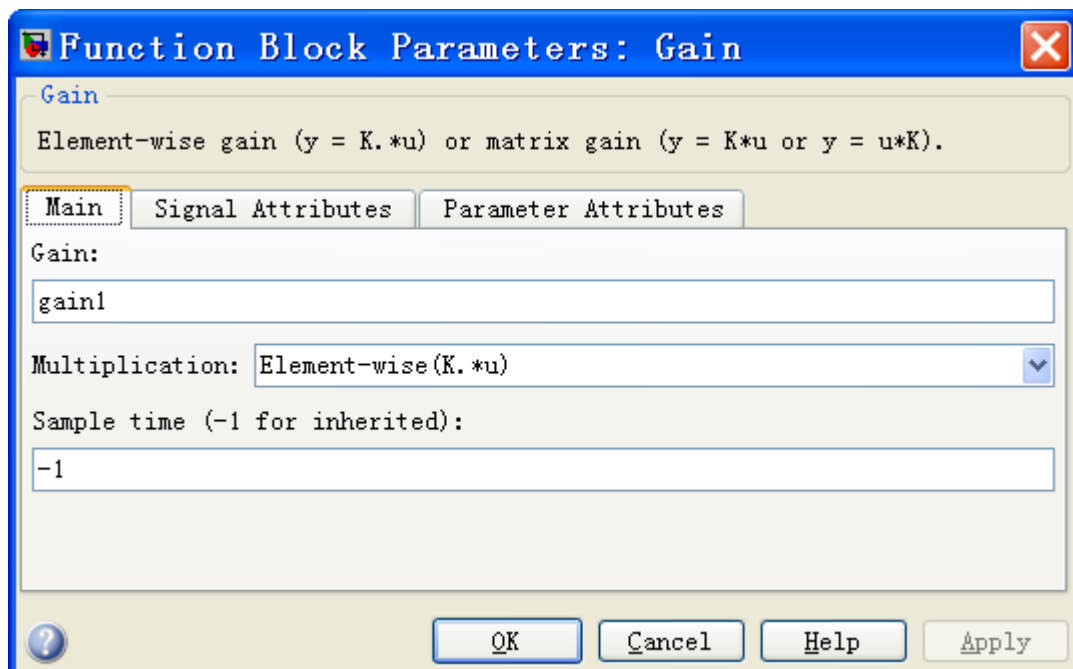
如下图所示需要实时在线调整 Constant 模块与 Gain 模块的值。



双击 Constant 模块，在 Constant value 框中输入 switch1，Sample time 框中输入-1，如下图所示，数据类型选为 boolean。



双击 Gain 模块，在 Gain 框中输入 gain1，Sample time 框中输入-1，如下图所示，输出数据类型选为 uint16，参数数据类型选为 uint16。





新建一个 M 文件，在文件中输入以下 MATLAB 代码：

```
%add calibration parameter switch1  
  
switch1=0;  
  
switch1=ecucoder_cal(switch1);  
  
  
%add calibration parameter gain1  
  
gain1=1;  
  
gain1=ecucoder_cal(gain1);
```

以上代码进行了两步操作：

第一步，给变量赋值；

第二步，通过函数 `ecucoder_cal` 将变量设置为标定变量。

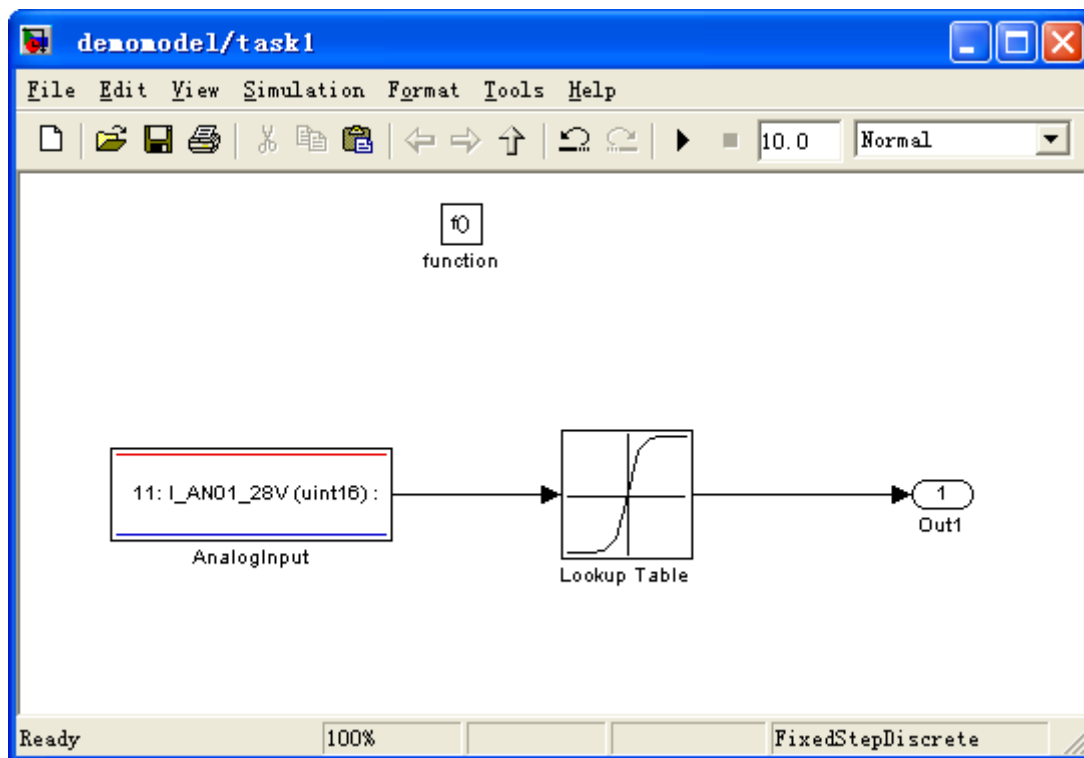
运行该 M 文件（也可在 MATLAB 工作空间直接键入上述 MATLAB 代码，为方便参数管理，建议使用 M 文件），即完成模型标定量 `switch1` 与 `gain1` 的添加。

在之后自动生成的 A2L 文件中，会出现添加过的信号的具体描述，如下图所示。

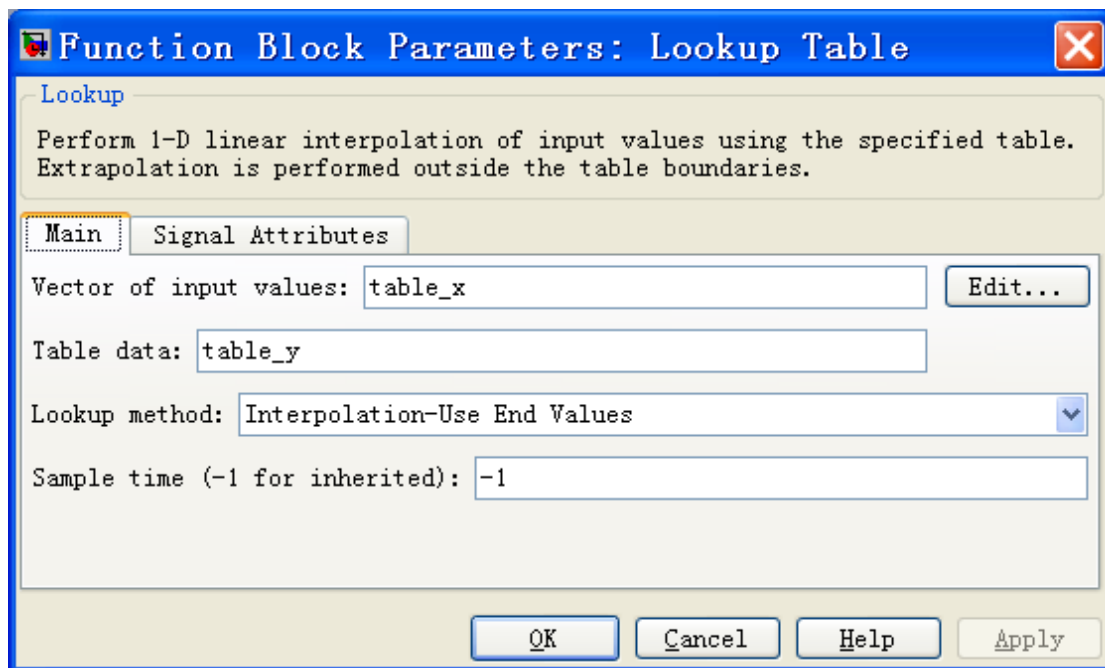
```
/begin CHARACTERISTIC  
  /* Name          */      gain1  
  /* Long Identifier */      ""  
  /* Type          */      VALUE  
  /* Memory Address */      0x40001e60  
  /* Record Layout  */      Scalar_UWORD  
  /* Maximum Difference */    0  
  /* Conversion Method */    demomodel_COMPU_METHOD_1  
  /* Lower Limit    */      0  
  /* Upper Limit    */      65535  
/end CHARACTERISTIC  
  
/begin CHARACTERISTIC  
  /* Name          */      switch1  
  /* Long Identifier */      ""  
  /* Type          */      VALUE  
  /* Memory Address */      0x40001e90  
  /* Record Layout  */      Scalar_BOOLEAN  
  /* Maximum Difference */    0  
  /* Conversion Method */    demomodel_COMPU_METHOD_2  
  /* Lower Limit    */      0  
  /* Upper Limit    */      1  
/end CHARACTERISTIC
```

## 2.3 添加一维查表标定量

如下图所示需要实时在线调整 LookupTable 模块的值。



双击 LookupTable 模块，在 Vector of values 框中输入 table\_x，Table data 框中输入 table\_y，Lookup method 框中选择 Interpolation-Use End Values， Sample time 框中输入-1，如下图所示，数据类型选为 uint16。



新建一个 M 文件，在文件中输入以下 MATLAB 代码：

```
%add calibration parameter table_x

table_x =[0:1000:30000];

table_x =ecucoder_cal(table_x);

%add calibration parameter table_y

table_y =[0:100:3000];

table_y =ecucoder_cal(table_y);
```

以上代码进行了两步操作：

第一步，给变量赋值；

第二步，通过函数 `ecucoder_cal` 将变量设置为标定变量。

运行该 M 文件（也可在 MATLAB 工作空间直接键入上述 MATLAB 代码，为方便参数管理，建议使用 M 文件），即完成模型标定量 `table_x` 与 `table_y` 的添加。

在之后自动生成的 A2L 文件中，会出现添加过的信号的具体描述，如下图所示。

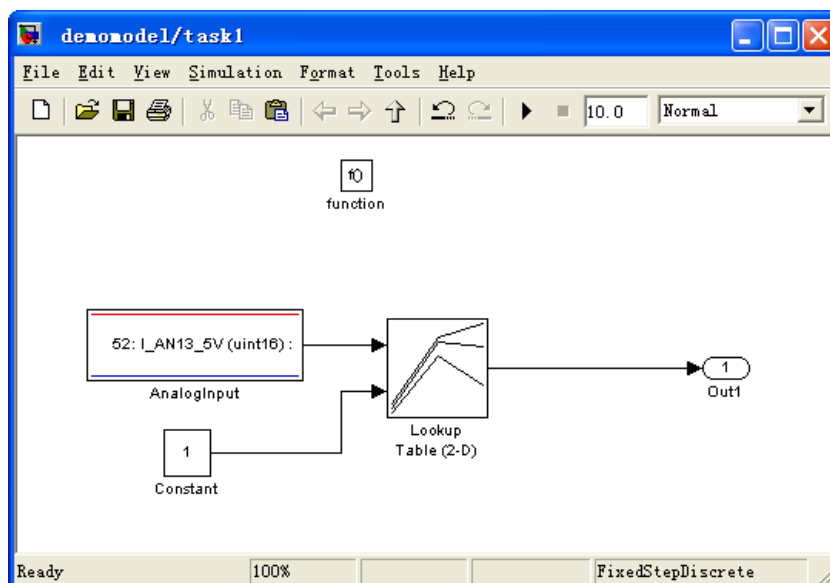
```

/begin CHARACTERISTIC
/* Name */ table_y
/* Long Identifier */ ""
/* Characteristic Type */ CURVE
/* Memory Address */ 0x40001e70
/* Record Layout */ Lookup1D_UWORD
/* Maxdiff */ 0
/* Conversion Method */ demomodel_COMPU_METHOD_1
/* Lower Limit */ 0
/* Upper Limit */ 65535
/begin AXIS_DESCR
/* Description of X-Axis Points */
/* Axis Type */ COM_AXIS
/* Reference to Input */ NO_INPUT_QUANTITY
/* Conversion Method */ demomodel_COMPU_METHOD_1
/* Number of Axis Pts */ 31
/* Lower Limit */ 0
/* Upper Limit */ 65535
AXIS_PTS_REF table_x
/end AXIS_DESCR
/end CHARACTERISTIC
/begin AXIS_PTS
/* Name */ table_x
/* Long Identifier */ ""
/* Memory Address */ 0x40001e30
/* Input Quantity */ NO_INPUT_QUANTITY
/* Record Layout */ Lookup1D_X_UWORD
/* Maximum Difference */ 0
/* Conversion Method */ demomodel_COMPU_METHOD_1
/* Number of Axis Pts */ 31
/* Lower Limit */ 0
/* Upper Limit */ 65535
/end AXIS_PTS

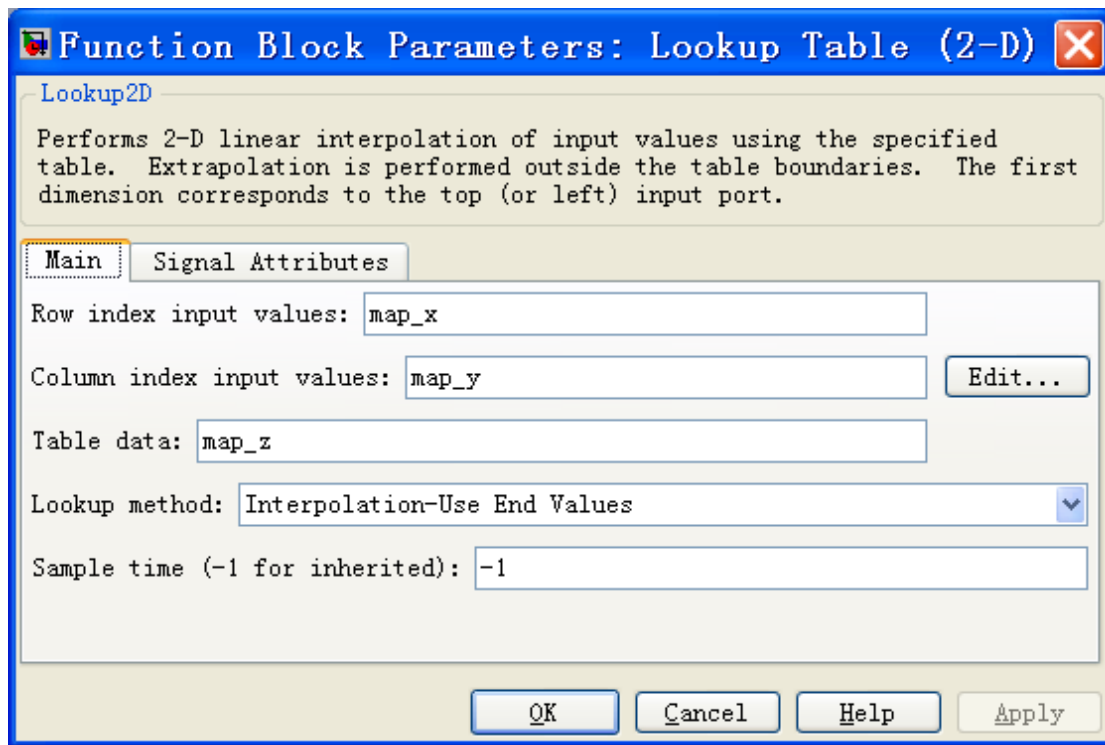
```

## 2.4 添加二维查表标定量

如下图所示需要实时在线调整 LookupTable(2-D)模块的值。



双击 LookupTable(2-D)模块，在 Row index input values 框中输入 map\_x，Column index input values 框中输入 map\_y，Table data 框中输入 map\_z，Lookup method 框中选择 Interpolation-Use End Values，Sample time 框中输入 -1，如下图所示，数据类型选为 uint16。



新建一个 M 文件，在文件中输入以下 MATLAB 代码：

```
%%add calibration parameter map_x
map_x=[1000:1000:3000];
map_x=ecucoder_cal(map_x);

%%add calibration parameter map_y
map_y=[1:3];
map_y=ecucoder_cal(map_y);

%%add calibration parameter map_z
map_z=[1 2 3;4 5 6;7 8 9];
map_z=ecucoder_cal(map_z);
```

以上代码进行了两步操作：

第一步，给变量赋值；

第二步，通过函数 `ecucoder_cal` 将变量设置为标定变量。

运行该 M 文件（也可在 MATLAB 工作空间直接键入上述 MATLAB 代码，为方便参数管理，建议使用 M 文件），即完成模型标定量 `map_x`、`map_y` 与 `map_z` 的添加。

在之后自动生成的 A2L 文件中，会出现添加过的信号的具体描述，如下图所示。

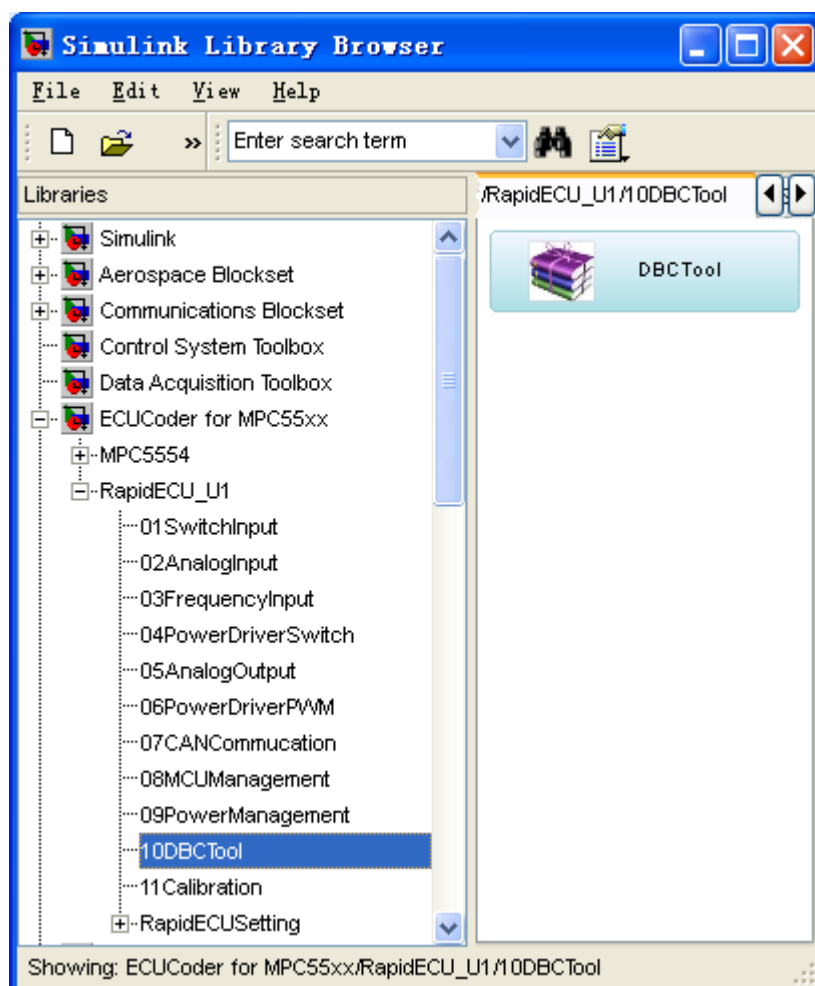
```
/begin CHARACTERISTIC
/* Name */ map_z
/* Long Identifier */ ""
/* Characteristic Type */ MAP
/* Memory Address */ 0x40001e30
/* Record Layout */ Lookup2D_UWORD
/* Maxdiff */ 0
/* Conversion Method */ demomodel_COMPU_METHOD_1
/* Lower Limit */ 0
/* Upper Limit */ 65535
/begin AXIS_DESCR
/* Description of Y-Axis Points */
/* Axis Type */ COM_AXIS
/* Reference to Input */ NO_INPUT_QUANTITY
/* Conversion Method */ demomodel_COMPU_METHOD_1
/* Number of Axis Pts */ 3
/* Lower Limit */ 0
/* Upper Limit */ 65535
AXIS_PTS_REF map_y
/end AXIS_DESCR
/begin AXIS_DESCR
/* Description of X-Axis Points */
/* Axis Type */ COM_AXIS
/* Reference to Input */ NO_INPUT_QUANTITY
/* Conversion Method */ demomodel_COMPU_METHOD_1
/* Number of Axis Pts */ 3
/* Lower Limit */ 0
/* Upper Limit */ 65535
AXIS_PTS_REF map_x
/end AXIS_DESCR
/end CHARACTERISTIC
/begin AXIS_PTS
/* Name */ map_x
/* Long Identifier */ ""
/* Memory Address */ 0x40001e78
/* Input Quantity */ NO_INPUT_QUANTITY
```

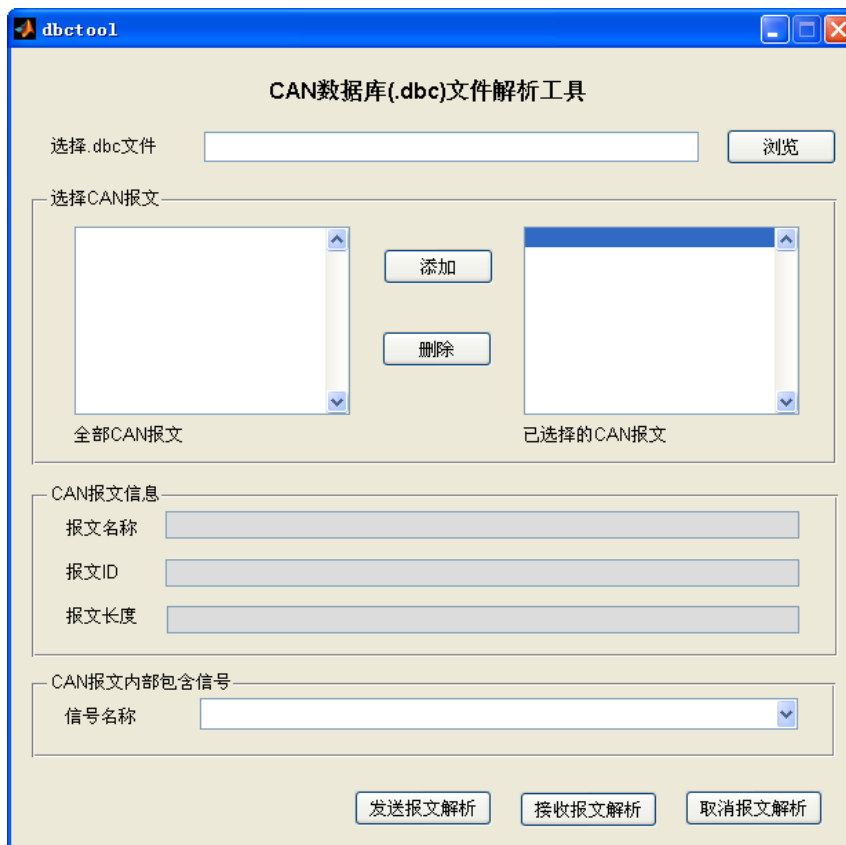
## 七、 DBC 工具的使用

DBC 文件是一种描述 CAN 总线网络信息的常用文件。很多情况下，需要将信号转换成 CAN 报文数据来发送或者将收到的 CAN 报文数据转换成信号，可以利用手工编程或者建模的方式来实现这个过程，但是比较繁琐，工作量较大，并且当通信协议修改的时候代码与模型也需要相应更改，非常不便。利用 ECUCoder 自带的 DBC 工具可以实现 DBC 文件导入与解析自动化。

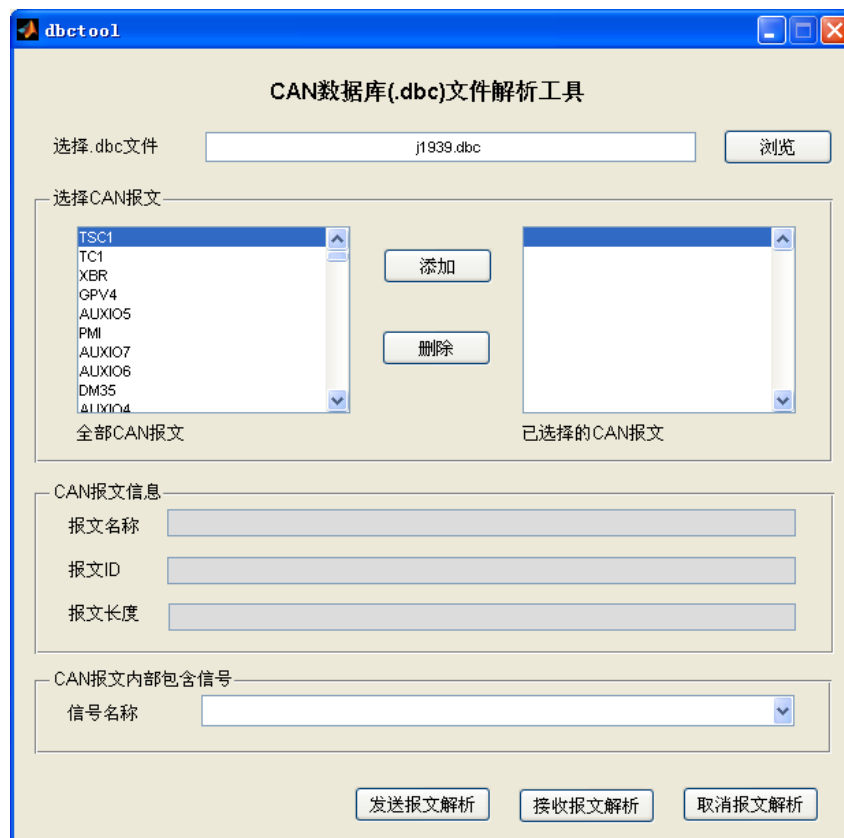
### 1. DBC 解析模块自动生成

- 1) 点击 DBCTool 子库，如下图所示，库中包含 1 个名称为 DBCTool 的模块，双击该模块，即可打开 DBC 工具，如下图所示。





- 2) 点击“浏览”按钮，选择 DBC 文件，之后，该文件中包含的报文会出现在“选择 CAN 报文”窗口内左侧的列表中，如下图所示。

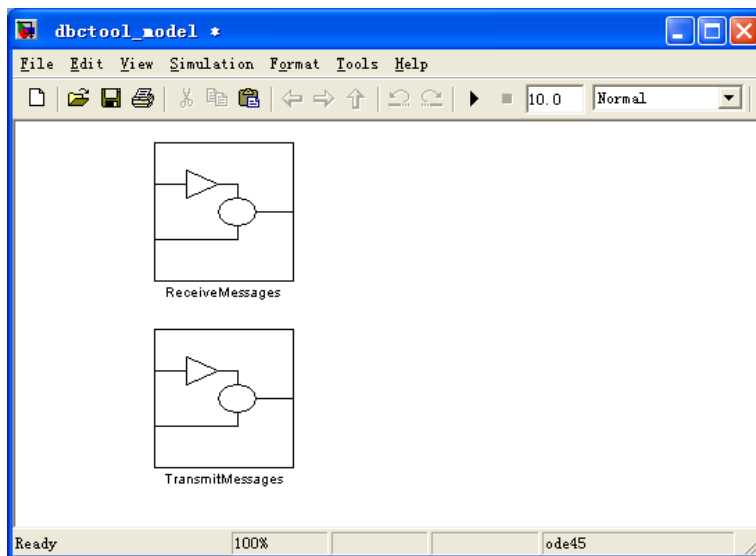




- 3) 在“选择 CAN 报文”窗口中用鼠标选取需要解析的报文，按住键盘上的 **Ctrl** 键可以选取多个报文，选取完报文后，点击“添加”按钮完成报文的选择，如下图所示。利用“删除”按钮可以将已经选择的报文删除掉。如果需要查询每个报文的相应信息，在报文列表中单选该报文，将会在“CAN 报文信息”窗口中显示报文的基本信息。“CAN 报文内部包含信号”窗口中显示被选中报文内部包含各个信号的名称。



- 4) 点击“发送报文解析”按钮生成发送报文的模型，点击“接收报文解析”按钮生成接收报文的模型。生成的模型名称为 `dbctool_model`，其中发送报文的模型位于模型的 `TransmitMessages` 模块中，接收报文的模型位于模型的 `ReceiveMessages` 模块中，如下图所示。



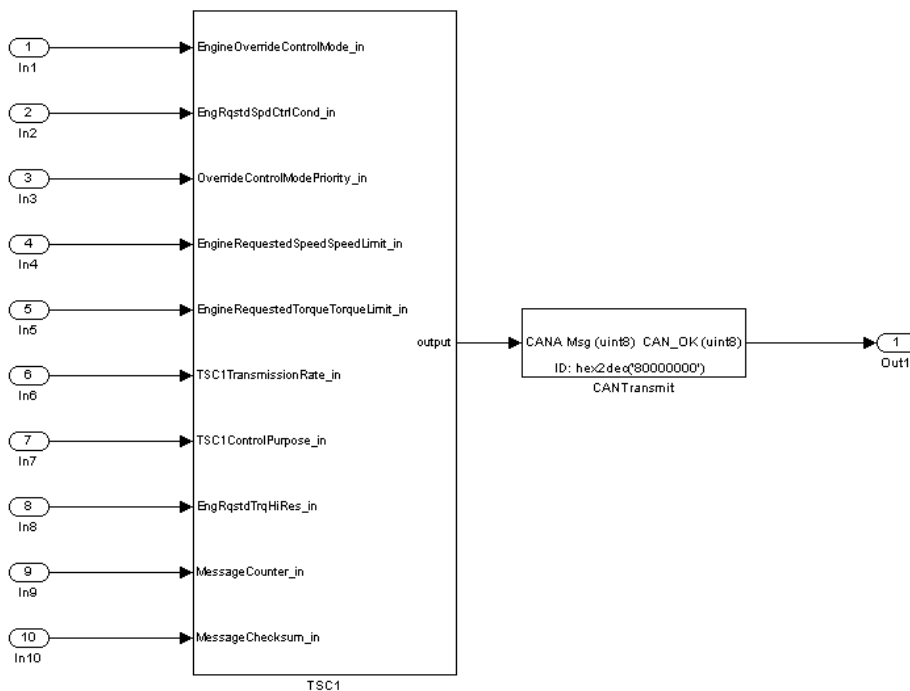
5) 保存模型，点击“取消报文解析”可关闭 DBC 工具主界面。

## 2. DBC 解析模块与 CAN 模块的集成

发送报文的模块需要与 CANTransmit 模块集成，接收报文的模块需要与 CANReceive 模块集成。

### 2.1 发送报文的模型集成

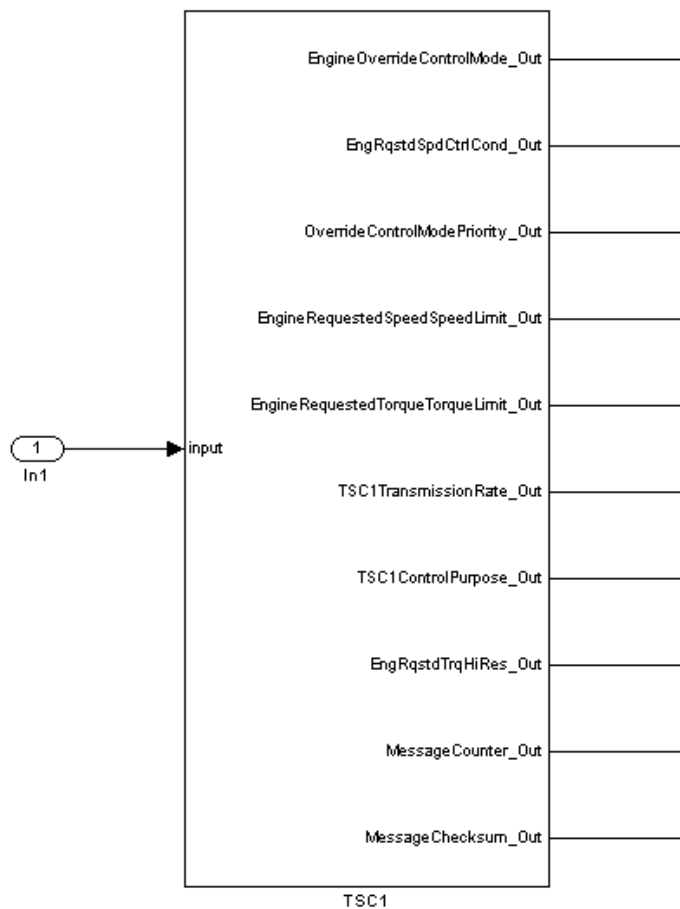
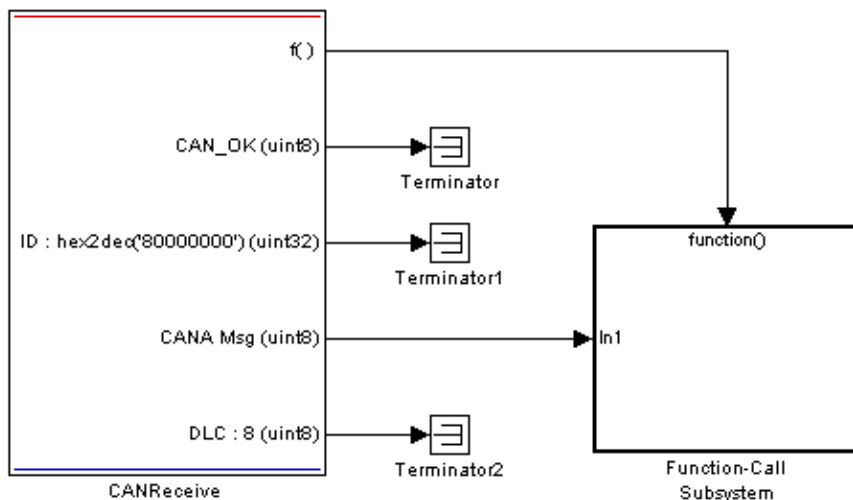
双击 DBC 解析模型的 TXMessage 子模块，将其中的模块复制到用户自己的模型，并将模块的输入与用户模型中的信号相连，模块的输出与 CANTransmit 模块相连，一个集成后的模型如下图所示。



注意将 CANTransmit 模块的 ID 设置为解析报文的 ID，数据长度设置为 8。

## 2.2 接收报文的模型集成

双击 DBC 解析模型的 RXMessage 子模块，将其中的模块复制到用户自己的模型，并将模块的输入与 CANReceive 模块相连，模块的输出与用户模型中的信号相连，一个集成后的模型与下图所示。



注意将 CANReceive 模块的 ID 设置为解析报文的 ID，数据长度设置为 8。

## 八、 控制器的管理

控制器管理为进一步使用控制器提供了简单易用的模块，主要包括单片机管理，电源管理与存储器管理等功能。

### 1. 单片机管理

点击 MCUManagement 子库，如下图所示，库中包含 4 个模块，分别为 CPUReset 模块、Initialization 模块、PIT 模块与 WDGFeed 模块，各个模块的作用见表 8-1。

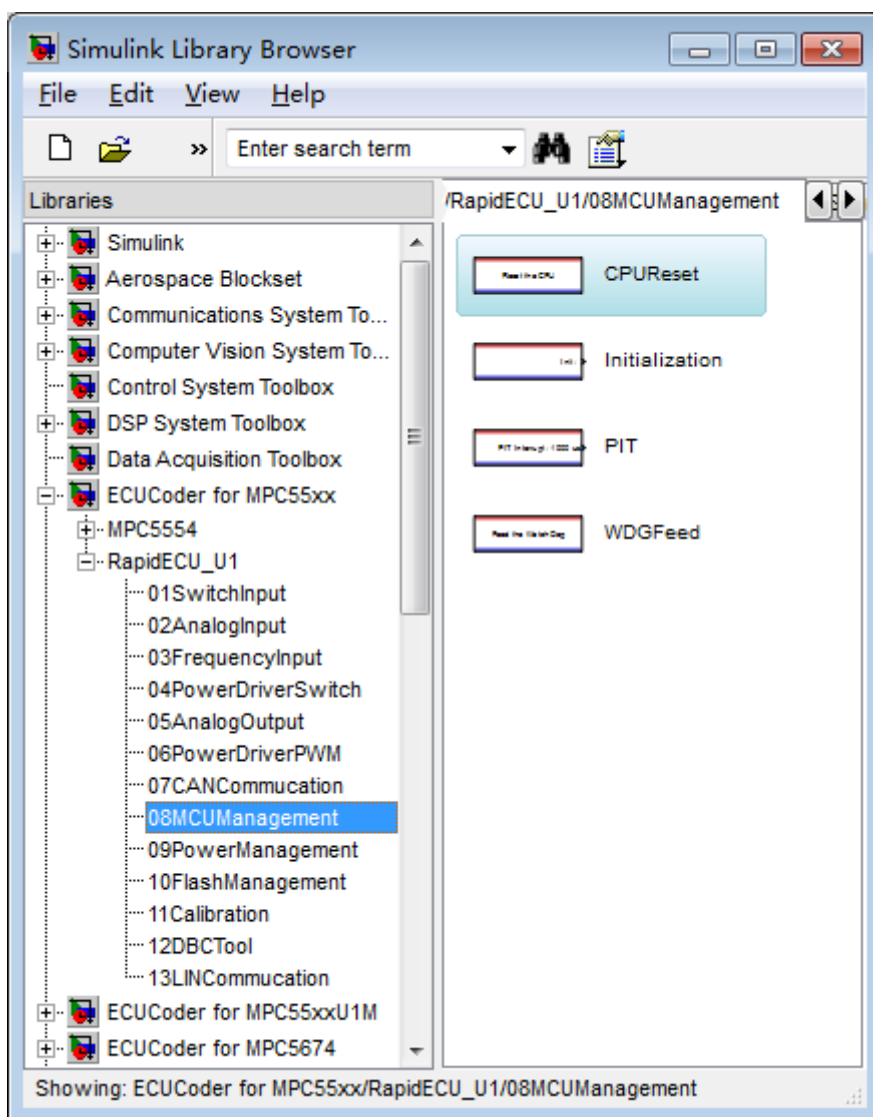


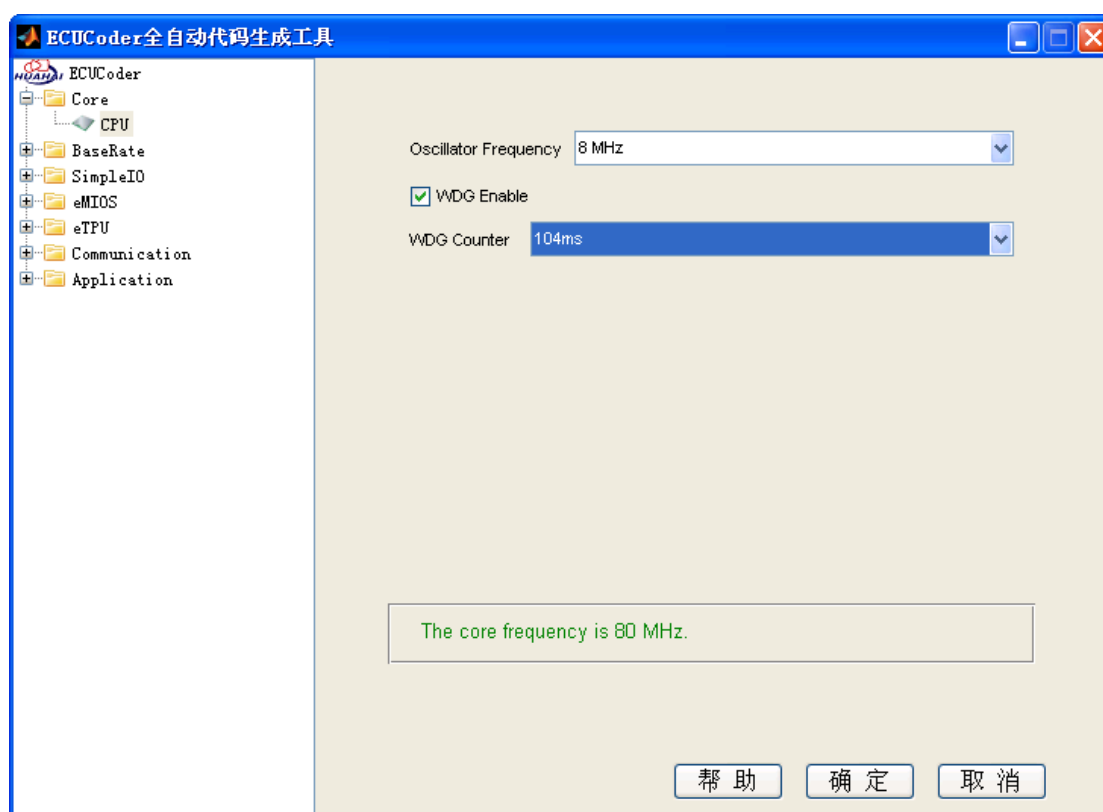
表 8-1 MCUManagement 子库

模块名称	描述
CPUReset	CPU 复位操作，用来复位单片机处理器。

Initialization	初始化模块，用来触发初始化任务。
PIT	周期性中断模块，用来触发周期性任务。
WDGFeed	喂狗操作，用来对单片机看门狗进行喂狗，防止看门狗计数器溢出引发单片机复位。

有些情况下，当控制器程序出现异常时，需要对控制器进行复位操作。将 **CPUReset** 模块置于模型的特定部分，可根据要求对控制器进行复位操作。

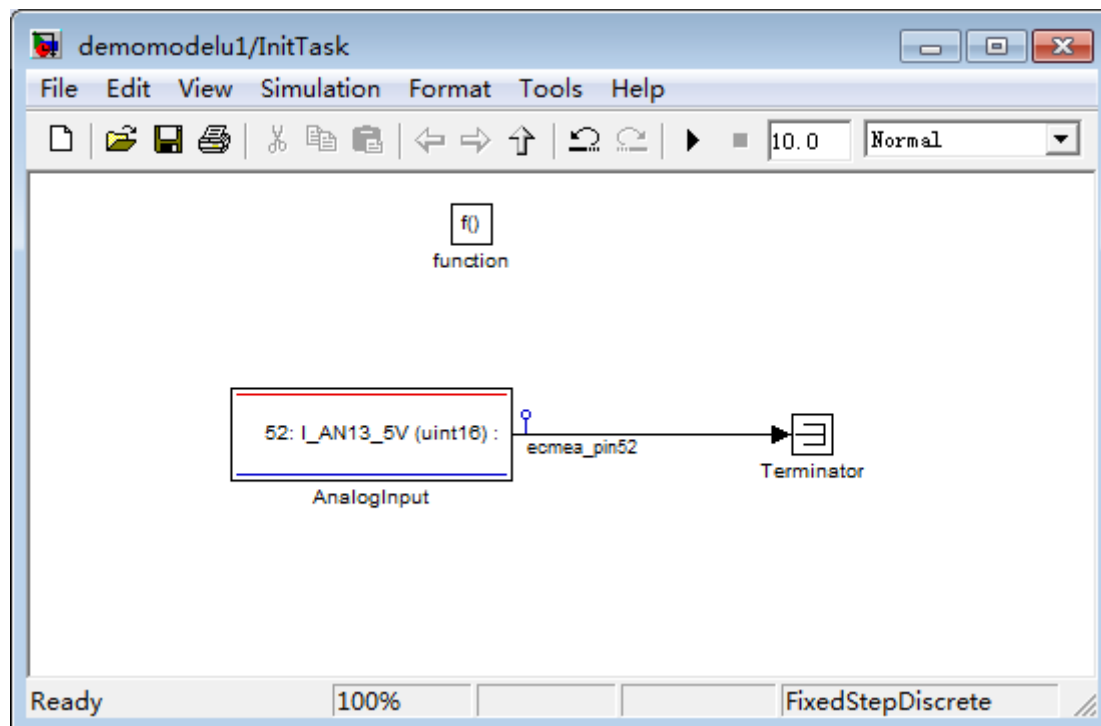
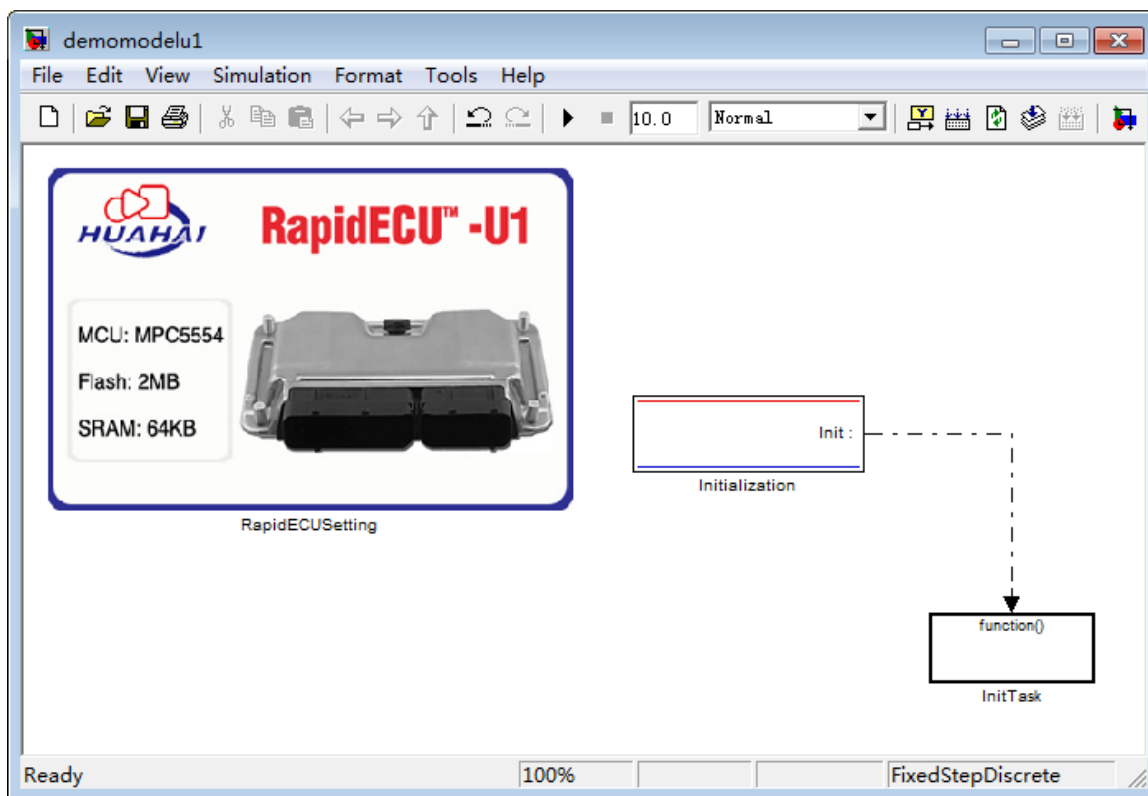
看门狗是一种常用的用于监控单片机运行状况的设备，当程序没有按照设计者的要求运行（没有按时喂狗）时，导致看门狗计数器溢出，引发单片机复位。要使用看门狗功能，首先需要使能看门狗，方法为双击 **RapidECUSetting** 模块，点击 **Core-CPU**，如下图所示，选择下图中 **WDG Enable** 前的复选框，使能看门狗，**WDG Counter** 为看门狗计数器的值，选择一个合适的值作为看门狗计数器溢出的时间，如下图所示，看门狗计数器溢出时间为 **104ms**，也就是说，程序必须在每 **104ms** 以内喂狗，一旦超出 **104ms** 不喂狗，单片机复位。将 **WDGFeed** 模块置于一个周期性触发的子系统中，程序即按照该周期反复喂狗，保证控制器程序的正常执行。



## 1.1 添加初始化任务

有些任务仅需要在控制器上电时执行一次，而不是周期性执行，利用 **Initialization** 模块可实现此功能，即添加初始化任务，模块的基本用法如下。

将 Initialization 模块拖入模型顶层，并与相应的任务模块相连，如下图所示，可在 InitTask 模块中添加各类初始化任务，比如读取某个模拟量输入管脚的电压值。

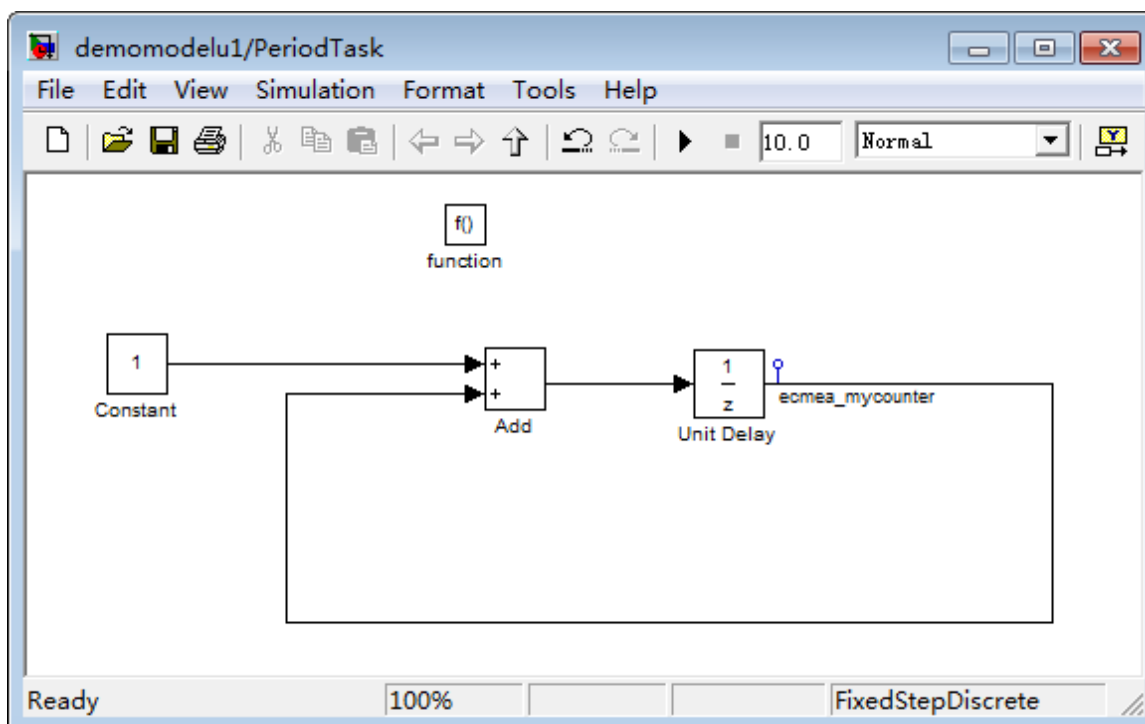
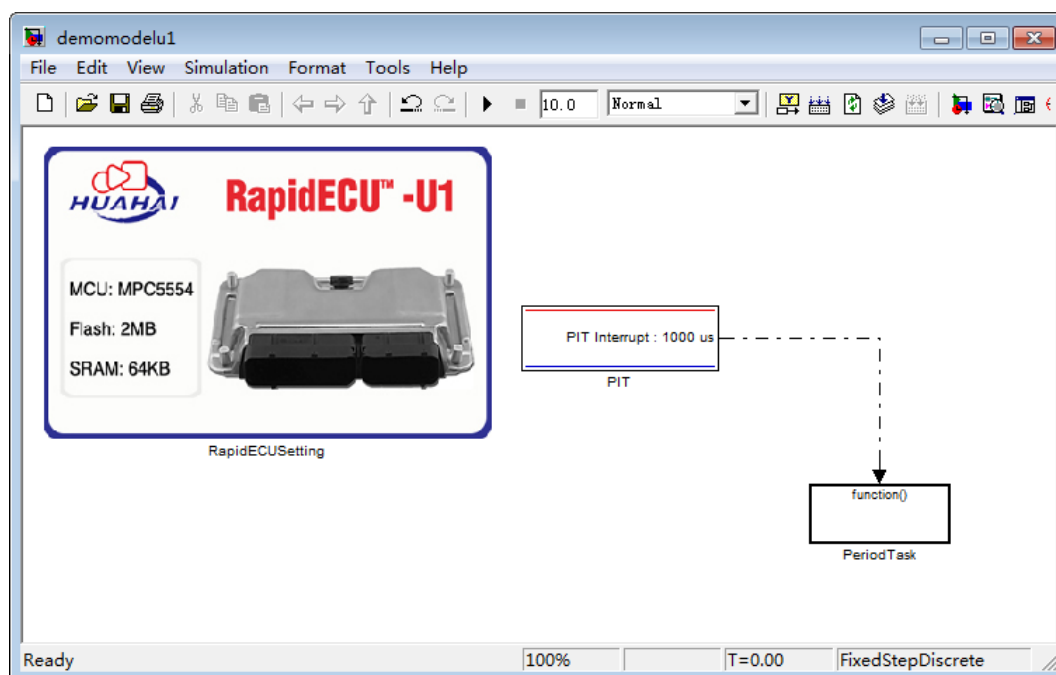


**注意：**请勿在初始化任务中添加周期性任务事件，比如接收 CAN 报文，否则将无法保证正确执行任务。

## 1.2 添加周期性任务

有些任务需要按照一定时序严格执行，利用 PIT 模块可实现此功能，即添加周期性任务，模块的基本用法如下。

将 PIT 模块拖入模型顶层，并与相应的任务模块相连，如下图所示，可在 PeriodTask 模块中添加各类周期性任务，比如实现一个软件计数器。



**注意:** PIT 模块采用硬件中断实现，请勿在周期性任务中添加大运算量任务，否则将无法保证正确执行任务。

## 2. 电源管理

点击 PowerManagement 子库，如下图所示，库中包含四个模块，分别为 AfterRunSwitch 模块、KL15Monitor 模块、MainRelaySwitch 模块与 PowerVoltageMonitor 模块，各个模块的作用见表 8-2。

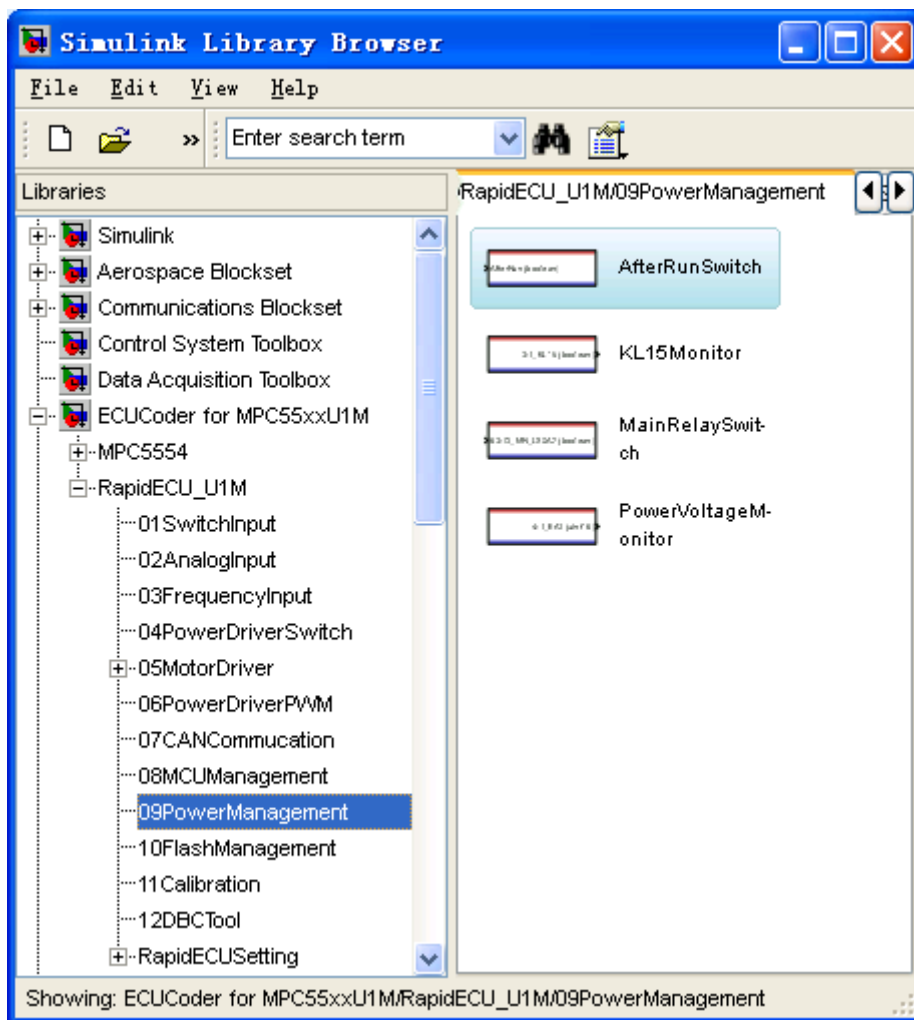


表 8-2 PowerManagement 子库

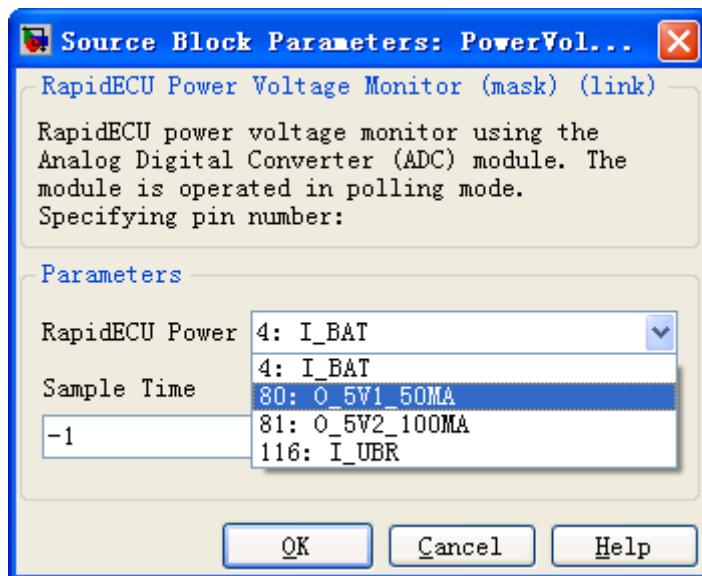
模块名称	描述
AfterRunSwitch	掉电延时开关，与 KL15 构成或逻辑来给控制器供电。
KL15Monitor	KL15 监控，用于监控点火开关 KL15 信号。
MainRelaySwitch	主继电器控制开关，用于控制主继电器的通断。
PowerVoltageMonitor	电源电压监控，用于监控控制器的各个电源。



## 2.1 电源监控

KL15Monitor 模块与 PowerVoltageMonitor 模块用于电源监控。其中 KL15Monitor 模块用于监控点火开关 KL15 信号，模块输出为 0 时表示点火开关断开，模块输出为 1 时表示点火开关闭合。

PowerVoltageMonitor 模块用于监控控制器的各个电源，将模块拖入到模型中，并选择相应的管脚，即可获取该管脚对应电源的值。例如，下图所示为 PowerVoltageMonitor 模块，选择管脚 80: O\_5V1\_50mA，即可获取控制器管脚 80 的输出值。



## 2.2 电源控制

AfterRunSwitch 模块与 MainRelaySwitch 模块用于电源控制。其中 AfterRunSwitch 模块为掉电延时模块，与 KL15 构成或逻辑来给控制器供电，即 AfterRunSwitch 与 KL15 中间只要一个为 1，则控制器有电。在 KL15 闭合控制器上电初始化时，将 AfterRunSwitch 模块置 1，则当 KL15 断开时，控制器仍然有电，只有将 AfterRunSwitch 模块清 0 后控制器才会掉电，即可实现掉电延时。

MainRelaySwitch 模块为主继电器控制开关，用于控制主继电器的通断。主继电器的一个控制端接蓄电池正极，另一个控制端接主继电器控制开关管脚。当 MainRelaySwitch 模块输入为 0 时，主继电器断开，当 MainRelaySwitch 模块输入为 1 时，主继电器闭合。

## 3. 存储器管理

点击 FlashManagement 子库，如下图所示，库中包含三个模块，分别为 FlashEnable 模块、FlashRead 模块与 FlashWrite 模块，各个模块的作用见表 8-3。

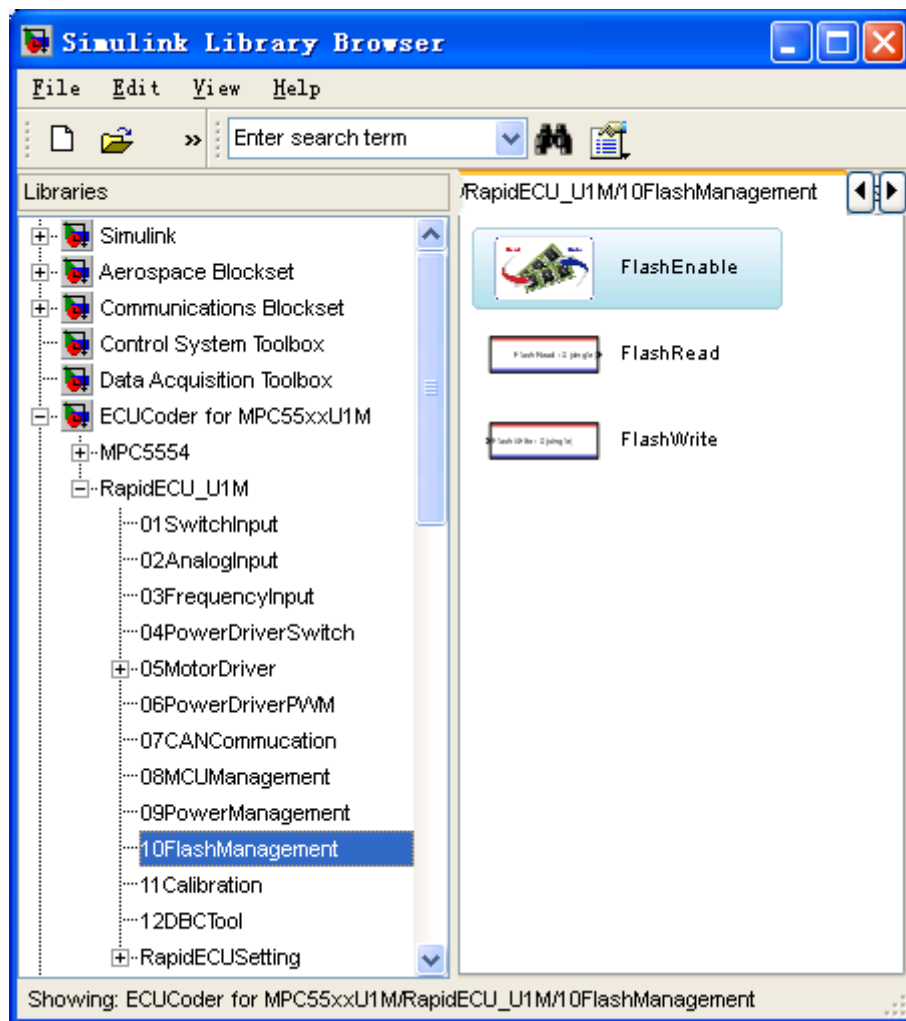


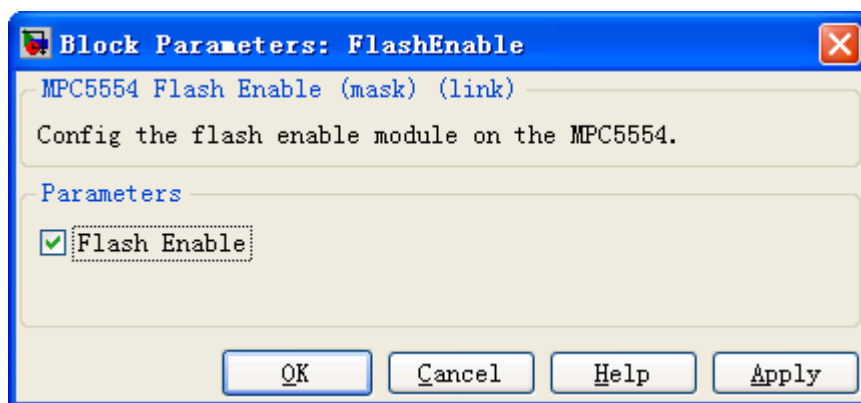
表 8-3 FlashManagement 子库

模块名称	描述
FlashEnable	Flash 使能模块。
FlashRead	Flash 读模块，从 Flash 存储器中读取数据。
FlashWrite	Flash 写模块，往 Flash 存储器中写入数据。

### 3.1 使能 Flash 存储器

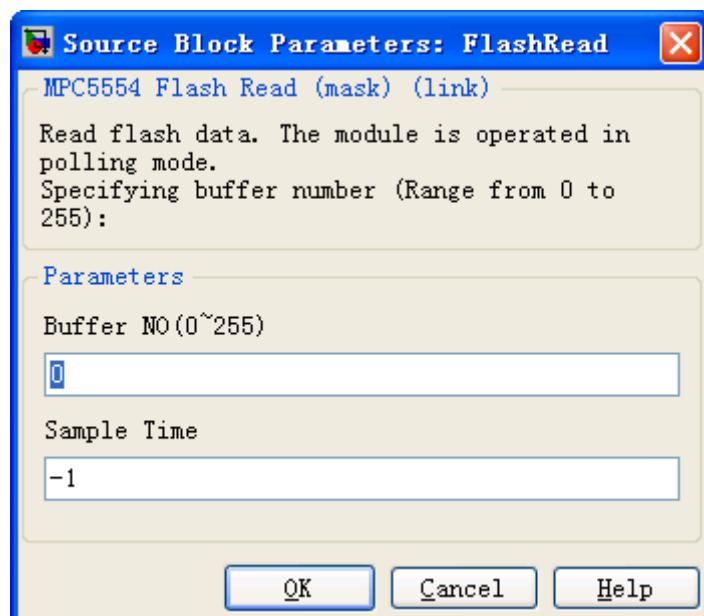
Flash 存储器可以存储数据，并且控制器掉电数据不会消失。要使用 Flash 存储器，首先需要使能 Flash 存储器，方法如下：

将 FlashEnable 模块拖入模型顶层，双击模块，勾选 Flash Enable 前的复选框，如下图所示，然后点击“OK”按钮，这样就使能了 Flash 存储器的读写功能。

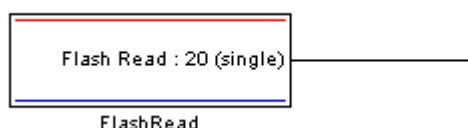


## 3.2 读写 Flash 存储器

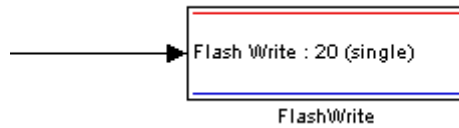
控制器提供了 1KB 的 Flash 存储器供用户自由使用，该存储空间分为 256 个缓存，可以存储 256 个 32 位数据。使用 FlashRead 模块可以从 Flash 存储器中读取数据，双击 FlashRead 模块，弹出如下对话框，其中 Buffer NO 即为缓存编号，可用值为 0~255。



使用 FlashRead 模块可以从 Flash 存储器中读取数据，例如，下图所示为从缓存 20 中读取数据。



使用 FlashWrite 模块可以往 Flash 存储器中写入数据，例如，下图所示为往缓存 20 中写入数据。



**注意：**使用 FlashWrite 模块往 Flash 存储器中写入数据并不是在程序执行时立即将数据写入 Flash，而是在控制器掉电（常电保持，点火开关断开）时统一将所有更改的缓存数据写入 Flash。

## 九、 如何建立简洁高效的模型

自动代码生成工具的基本功能是将 Simulink 模型转换为代码，如果 Simulink 模型本身是错误的，那么生成代码的正确性必然难以保证。为了提高 Simulink 模型的正确性与可读性，建模者需要掌握如何建立简洁高效的模型，养成良好的建模习惯。

### 1. 为模块指定数据类型

利用自动代码生成技术可将 Simulink 模型自动转换成代码，绝大部分自动生成的代码都是以 C 语言的形式提供的，而在 C 语言中所有的常量、变量、参数、函数返回值等等都必须有明确的数据类型。因此，为了保证生成 C 代码的高效可靠，必须指定 Simulink 模型中模块的数据类型。

**注：**在非嵌入式建模中，一般不需要指定数据类型也能生成正确的代码，但是此类代码没有经过任何优化，占用较多硬件资源，执行效率低下，并且可读性与可维护性较差，无法应用到产品尤其是汽车级产品控制器当中。

#### 1.1 需要指定数据类型的模块

对于大部分模块，比如 Constant 模块、Add 模块、Subtract 模块、Product 模块、Divide 模块、Switch 模块等等，需要指定输出数据类型。

对于一些带参数运算的模块，比如 Gain 模块、Lookup Table 模块、Lookup Table (2-D)模块等等，除了需要指定输出数据类型之外，还需要指定参数的数据类型。

对于一些无数据类型的模块，比如 Mux 模块、Demux 模块、Unit Delay 模块、Ground 模块、Terminator 模块等等，则不需要指定数据类型。

#### 1.2 指定数据类型的常用方法

无论哪种硬件平台，整型数运算的效率总是要高于其它数据类型，因此，使用整型数的模型所生成的代码效率也更高。在指定模块数据类型的时候，可以按照以下方法来操作：

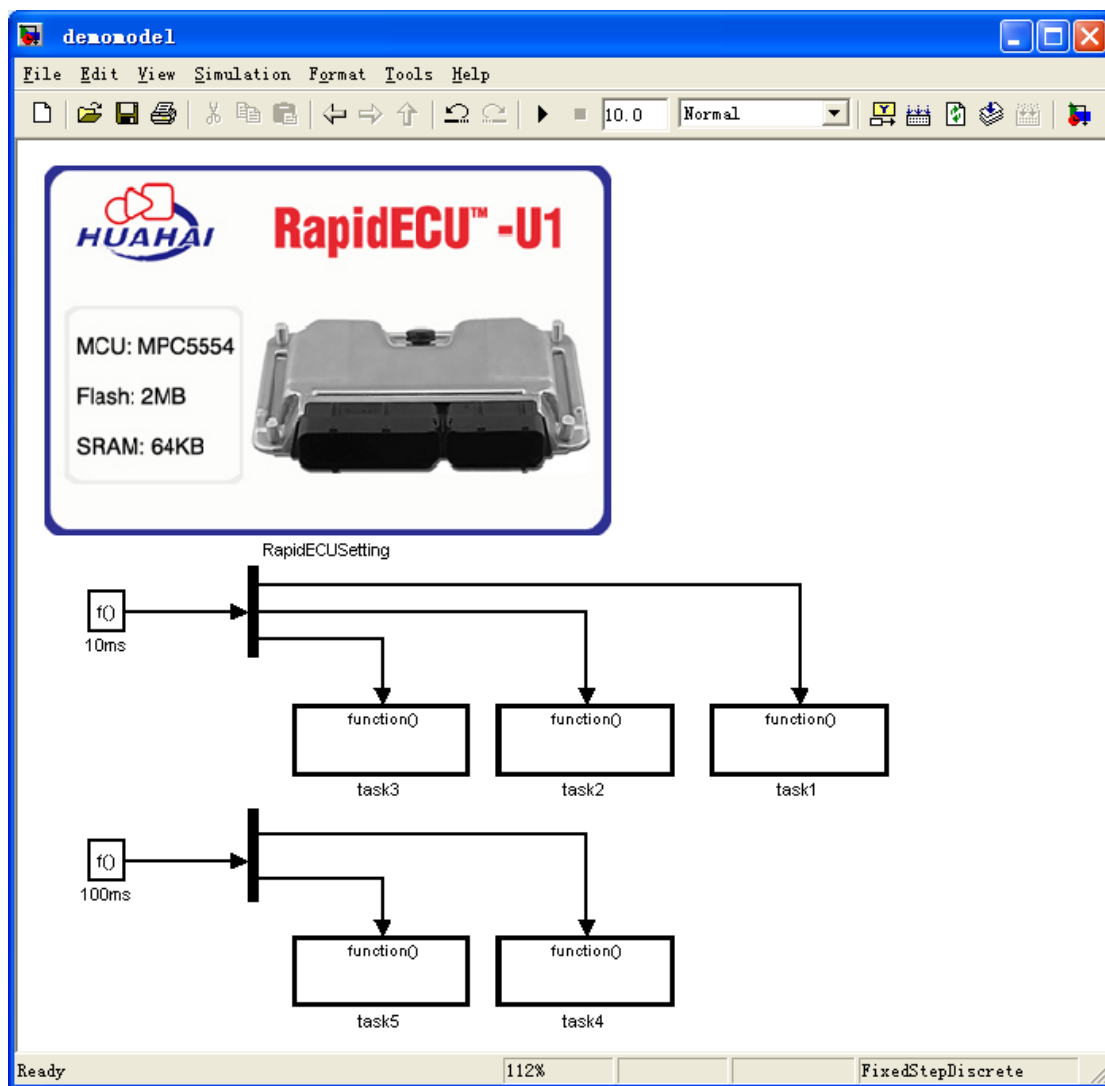
- (1)在开发初期不关注数据类型的话可以将所有数据类型指定为 single 类型(尽量避免 double,因为 double 占用的硬件资源较多)；
- (2) 如果已经明确知道该数据的数据类型为整型数的话，指定为整型数；
- (3) 如果已经明确知道该数据的数据类型为非整型数的话，指定为 single 类型；
- (4) 如果不能明确知道该数据的数据类型的话，指定为 single 类型；

(5) 硬件 IO 模块的数据类型都是整型数 (boolean、uint8、uint16、uint32)，在模型中必要的地方加上 Data Type Conversion 模块来进行强制数据转换。

## 2. 明确任务的执行周期与执行顺序

在控制器内部的任务中，有些是按周期执行的，有些是中断执行的，对于周期性任务，必须明确任务的执行周期与执行顺序，对于中断任务，必须明确任务的执行条件、优先级与执行顺序。

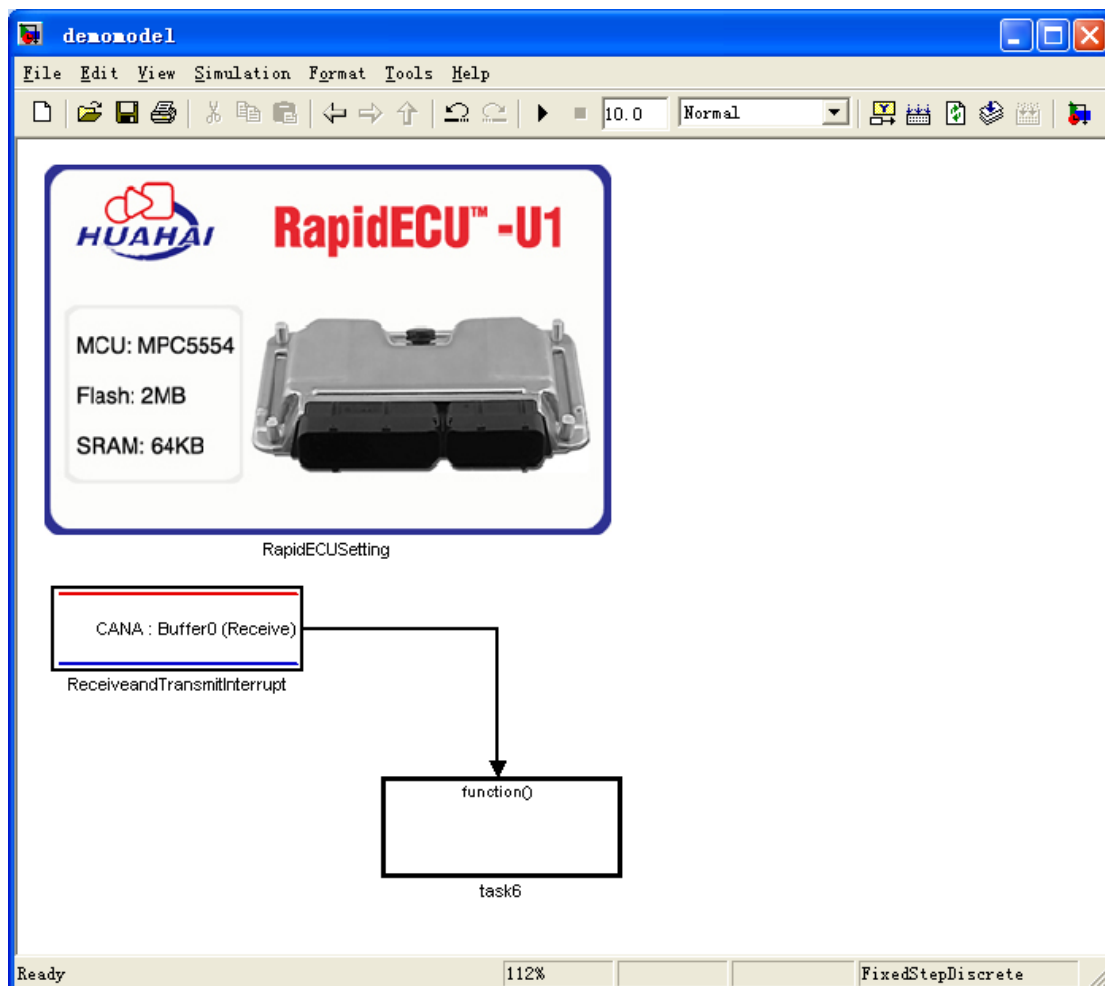
对于周期性任务，利用 Simulink 库中的 Function-Call Generator 模块，可以指定任务执行的周期，利用 Simulink 库中的 Demux 模块，可以指定任务执行的顺序。如下图所示模型，一共包括 5 个任务，其中 task1、task2、task3 的执行周期是 10ms，执行顺序是 task1、task2、task3 顺序执行；其中 task4、task5 的执行周期是 100ms，执行顺序是 task4、task5 顺序执行。



利用 Function-Call Generator 来指定任务执行的周期与顺序逻辑清晰，建模简单，模型的可读性好，

是最常用的建模方式。即使是最简单的单一任务模型，也推荐使用此种方式来建模。

对于中断任务，利用中断触发模块来触发任务。如下图所示模型，任务 task6 为 CANA 模块 Buffer0 中断触发的中断任务。



在实际控制器内部往往有较多任务，包含周期性任务与中断任务，在建模时，一般需要用到较多 Function-Call Generator 模块与中断触发模块，为了使整个模型层次清晰，推荐将所有的 Function-Call Generator 模块与中断触发模块置于模型顶层。



## 华海科技简介

华海科技是专业的汽车电控系统解决方案供应商，国家级高新技术企业，公司成立于2011年，长期致力于为汽车、机械、教育等行业的用户提供面向新能源和智能化领域应用的电控系统解决方案。

公司拥有10年以上经验的技术研 【战略布局】

发团队，与清华大学等知名高校，美国RTI等国际领先公司深入合作。公司秉承卓越和创新的精神，凭借高质量的产品，不断提升的工具平台，全方位的技术服务与领先的中小批量电控系统解决方案，赢得了众多客户的好评。

截止2017年，华海科技电控产品已为多家主机厂十余款量产车型批量配套。



华海科技具备的核心技术能力包括：

- 新能源汽车电控系统集成开发
- 发动机管理系统开发
- AMT/AT/DCT/CVT 自动变速器控制器开发
- 嵌入式软件开发
- 硬件在环仿真系统开发
- 基于虚拟现实的驾驶模拟平台开发

华海科技联系方式：

技术中心：北京九州华海科技有限公司

地址：北京亦庄经济技术开发区凉水河一街超明园综合楼1层

电话：010-84670398

产业化基地：浙江华亦海汽车电子科技有限公司

地址：浙江省嘉兴市秀洲区加创路321号上海交大科技园

电话：0573-82791181

邮箱：[support@ecucoder.com](mailto:support@ecucoder.com)

网址：[www.ecucoder.com](http://www.ecucoder.com)