自动代码生成工具——ECCoder 使用说明

文档说明

本文档更新历史

版本	制定者	日期	更新内容
1/0	刘略	2018.8.11	初始版本
1/0/1	刘略	2018.8.13	添加 EEPROM 模块说明，生成代码结构说明。
1/1/2	刘略	2018.11.28	1.1.2 更新，增加了系统级 tlc 文件，新增用户自定义配置，模块功能变更,新增 CAN 管理和数据管理工具。
1/1/3	刘略	2018.12.20	1.1.3 更新，PSWT 输出模块增加了诊断信息输出端口。
1/1/4	刘略	2019.1.16	1.1.4 更新，集成了 Layout Toolbox 中功能，只需要安装 ECCoder 即可。
1/2/0	刘略	2019.2.12	1.2.0 更新，添加 MATLAB 内建帮助文档。更新了自动集成功能。
1/3/0	刘略	2019.4.8	1.3.0 更新，增加了自动编译功能，增加了系统管理模块分类，增加了包管理功能，增加了系统配置功能。
1/3/1	刘略	2019.5.24	1.3.1 更新，增加通过脚本方式添加标定/EEPROM 变量；包管理功能完善，增加了覆盖旧版本 package 功能；增加 pwm 输出模块；CAN 配置模块增加 MO 数量限制；增加了集成后 A2L 自动合并功能和独立 A2L 合并工具；增加了 DEMO 样例；激活界面可使用文件激活，激活码添加了日期限制（旧版激活码不再适用）。
1/3/2	刘略	2019.7.5	1.3.2 更新，增加了复杂驱动模块及相应说明。
1/3/3	刘略	2019.7.23	1.3.3 更新，复杂驱动中增加了加速传感器模块。demo 中增加了指引样例 start。
1/4/0	刘略	2019.8.8	1.4.0 更新,增加了 TC275 平台支持，TLC 代码生成架构变更。增加了 PWM Input 模块。修复了 A2L 合并工具无法正确合并的问题。修复了 CAN TX/RX 模块无法复制的问题。更新了 start 样例中部分内容。 基于扩展性和稳定性考虑，从该版本开始推荐使用脚本/mat/数据字

			典方式管理标定和 EEPROM 变量，配合 simulink 原生 constant/lookuptable 模块进行建模。不再推荐使用 ECCoder 中 Parameter 模块。
1/4/1	刘略	2019.9.3	1.4.1 更新，增加了氢气瓶阀配置及驱动模块。修复了若干 TLC 文件问题。
1/4/2	刘略	2019.9.16	1.4.2 更新，修复了 TC275 平台 CAN 回调函数 BUG，增加了 TCU 双通道位置及转速传感器模块。
1/4/3	刘略	2019.9.30	1.4.3 更新，复杂驱动中增加了 TC234 平台发送复用模块。若干模块界面优化。
1/4/4	刘略	2019.10.24	1.4.4 更新，修正了 TC275 平台 PWM 输入输出模块端口描述错误。调整所有输出模块为直馈模式。
1/4/5	刘略	2019.11.28	1.4.5 更新，新增了 H 桥驱动模块，增加了诊断类型。
1/4/6	刘略	2019.12.10	1.4.6 更新，增加了模拟量输出 AD Ouput 模块。
1/4/7	刘略	2019.12.20	1.4.7 更新，增加了 H 桥驱动模块和氢喷模块的 TC275 平台代码生成支持。
1/4/8	刘略	2020.5.22	1.4.8 更新，增加了 SENT 通讯 Fast/Slow 模块。修正了说明书中 PWM Out 频率精度错误。
1/4/9	刘略	2020.7.2	1.4.9 更新，TC275 平台 PWM IN 模块，取消周期/持续时间模式，周期数据类型为 uint32，精度 1us/bit。
1/5/0	刘略	2021.3.26	1.5.0 更新，增加了加速度传感器模块在 TC275 平台中的支持，增加了 TCU 转速传感器模块。
1/5/4	刘略	2021.9.3	1.5.4 更新，增加了氢喷分组控制和氢喷独立控制模块。
1/5/5	刘略	2021.9.17	1.5.5 更新，增加了获取氢喷状态模块。

目录

1 概览

1.1 环境要求

ECCoder 基于 Matlab R2016b 版本开发，在低于该版本 Matlab 上运行该软件可能会产生兼容性问题。该软件视图部分基于开源 GUI 工具箱 GUI Layout Toolbox 2.3.1 开发，该工具箱已集成至 ECCoder 中。

ECCoder 使用中需依赖特定工具箱，下表中列出了 Matlab R2016b 中 ECCoder 所需的工具箱：

产品（Matlab R2016b）	备注
MATLAB 9.1	MATLAB 基础环境
Simulink 8.8	Simulink 仿真环境
Embedded Coder 6.11	用于生成嵌入式代码(ert.tlc)
Simulink Coder 8.11	用于 Simulink 生成代码
Stateflow 8.8	选配，模型中使用 Stateflow Chart
Vehicle Network Toolbox 3.2	选配，用于解析 CAN 通讯 DBC 文件

1.2 功能概述

随着自动代码生成技术的发展，利用可视化模型实现控制器的快速原型开发已经成为可能，Matlab 的实时工作间 RTW 作为应用最为广泛的自动代码生成工具已经为汽车行业所熟知。但 Matlab RTW 主要针对上层算法模型进行代码生成，如果用户想要在目标硬件中运行它们还需要手动编写与硬件相关的底层驱动代码。在原型开发阶段算法模型往往需要快速迭代，重复编写驱动接口过程繁琐、效率低下。且许多算法工程师对于底层往往并不熟悉，大量的集成工作使其精力难以专注于算法模型开发。鉴于此，ECCoder 应运而生。

ECCoder 现阶段主要功能是为模型提供底层硬件接口，接口以 Simulink 模块库的形式开放给用户，用户使用简单的拖拽即可添加至模型中。模块均配有相应

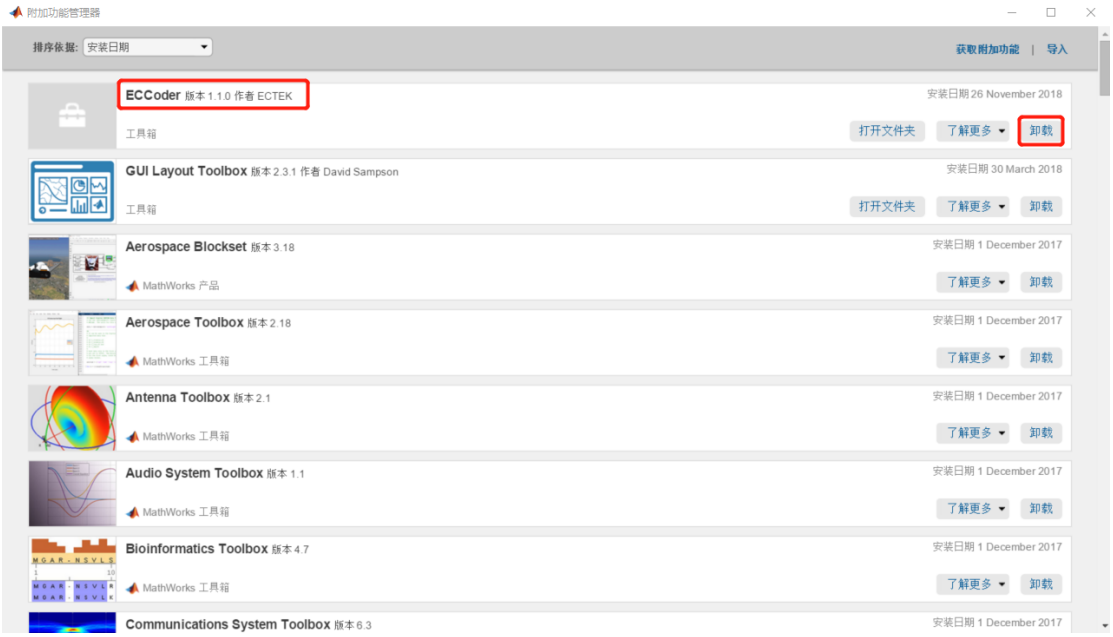
的视图界面，用户可以轻松进行配置。这些模块在代码生成中会直接生成可直接执行的接口代码，只需进行简单配置修改，即可在嵌入式设备中运行。

2 ECCoder 安装说明

ECCoder 需要进行安装之后才能进行使用，安装步骤如下：

2.1 删除旧版 ECCoder 工具箱（可选）

在 Matlab 主视图中选择附加功能，点击管理附加功能。在打开的页面中，找到旧版 ECCoder，选择卸载。卸载时请确保无 Simulink 工程或 ECCoder 相关功能正在运行。（如果之前未安装过 ECCoder，请忽略此步骤）



2.2 安装 ECCoder 工具箱

将 ECCoder 安装包(.mltbx)文件拖拽至 Matlab 命令行中，根据提示完成安装即可。

2.3 激活与配置

安装完成后，需要激活 ECCoder 才能进行正常使用，否则无法进行代码生成和自动编译。在命令行中输入：

eccoder.activation;

复制页面中的 System ID 并提供给 ECTEK，获取到 Key 之后粘贴到 Activation Key 栏中即可完成激活。对于之前激活过，但是升级或重新安装工具箱的情况，可直接使用之前激活码激活或在卸载旧版工具箱之前另存工具箱路径 bin 文件夹下的 license.dat 文件，在安装新的工具箱之后加载该文件即可。激活后重新输入该命令可查看 license 有效日期。

在正式使用之前，还需要对某些 IDE 路径进行配置（供自动编译功能使用，如果无需自动编译可不指定）。在命令行中输入：

eccoder. systemconfig;

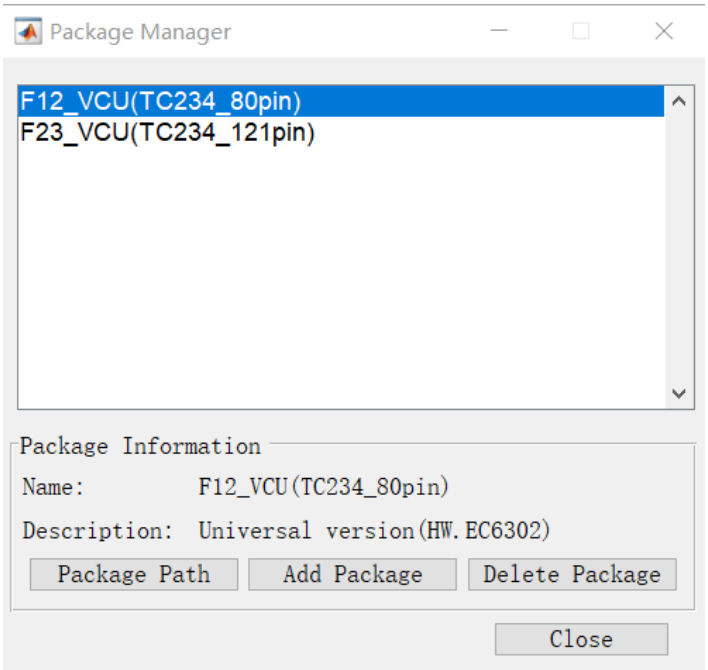
在界面中点击 browser 按钮，分别指定 Hightec Eclipse IDE 路径和 Code Warrior 路径，指定完成后点击 OK 即可。Hightec 编译器文件名为 eclipsec.exe，一般保存在 HIGHTEC\ide\htc-ide-v2.X.X(eclipse-v1.X.X)路径下。

2.4 扩展包管理

ECCoder 采用包的形式来实现对额外硬件平台的支持。在命令行中输入：

eccoder. packagemanager;

来打开扩展包管理界面。



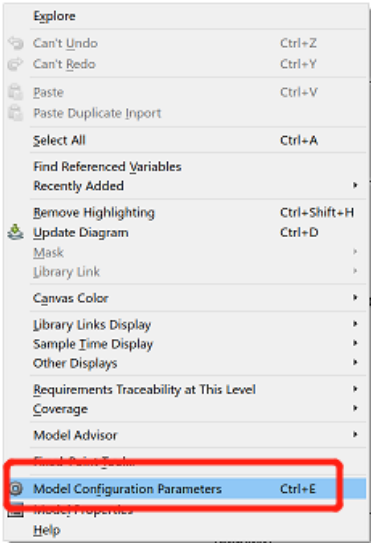
在界面中，用户可以添加/删除扩展包。点击 Add Package 按钮，选择由 ECTEK 提供的.ecpkg 文件即可添加。每一个扩展包对应一个相应的硬件平台，添加扩展

包成功后，即可在模型 Configuration – Code Generation – Platform 页面 Hardware Platform 列表中发现相应平台（如果 ECCoder 仅包含唯一一个包则无法删除）。如果添加的 package 名称在列表中已存在，会将旧包文件覆盖。

可以点击 Package Path 按钮打开包路径，路径中 pin_def.pdf 文件为该平台硬件接插件定义及硬件规格；Monitor.zip 压缩包为该平台底层 a2l 文件及 ECKA 监控工程；DefaultProject.zip 压缩包为该平台底层 Hightec IDE 工程文件，包含了底层库及上层基础服务代码；UserDefine.xml 为用户自定义配置样例文件。该路径中内容均为只读，如有需要请另行拷贝，切勿直接修改，修改后可能会导致 ECCoder 编译出错。

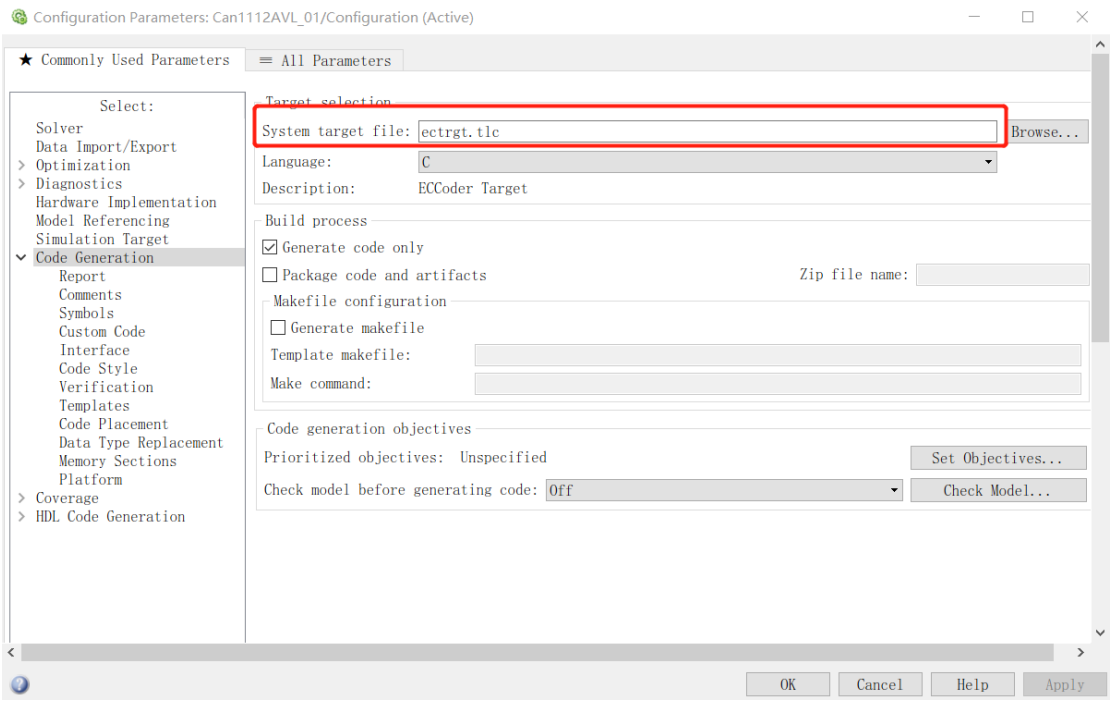
3 模型配置参数 Model Configuration Parameters

在模型搭建前，首先需要对模型进行配置。在模型空白处单击右键，选择 Model Configuration Parameters。



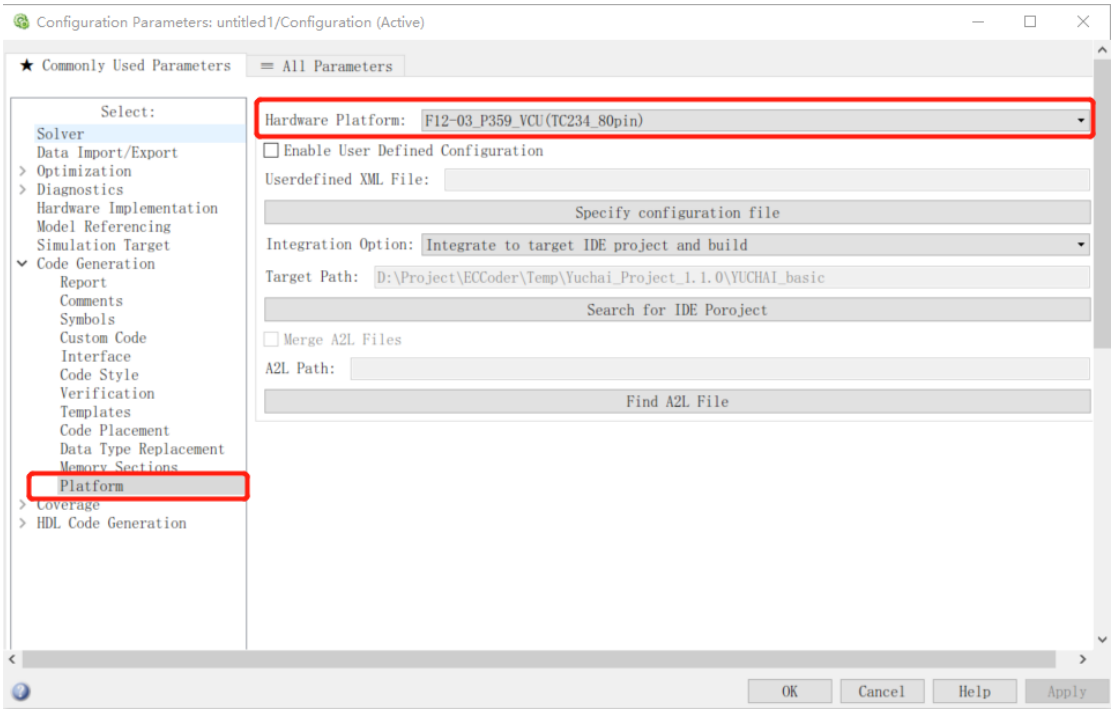
3.1 选择 tlc 文件

在 Configuration 窗口左侧菜单中选择 Code Generation 选项，选择 System target file 为 ectrgt.tlc。



3.2 选择硬件平台

在 Code Generation 菜单下找到 Platform 选项卡，在 Hardware Platform 一项选择模型对应的硬件平台。



3.3 自动配置参数

选择 ectrgt.tlc 后，ECCoder 会对模型某些参数进行自动配置。默认自动修改的参数如下：

表.自动配置参数列表

页面	参数	值
Solver	Type	Fixed-step
	Solver	discrete
	Fixed-step size	0.01
Data Import/Export	Time	off
	Output	off
Optimization	BlockReduction	off
	Default for underspecified data type	single
	InitFltsAndDblsToZero	off
	SimCompilerOptimization	on
	DefaultParameterBehavior	Inlined
Diagnostics	InheritedTsInSrcMsg	none
Code Generation	GenerateMakefile	off
	RTWVerbose	off
	SupportComplex	off
	SupportNonFinite	off
	IncludeMdlTerminateFcn	off
	GenerateASAP2	on

自动配置的模型参数均为推荐设置，在了解其功能的前提下，可由用户自行修改。

4 模块功能介绍

模块根据功能不同可分为六类：CAN 通讯模块、诊断模块、IO 接口模块、变量模块、系统管理模块和复杂驱动模块。模块库(ECCoder_Library)中已经将六种模块分类打包。

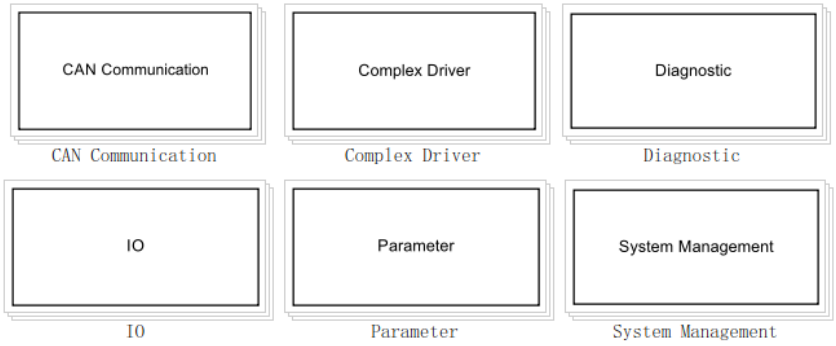


图.ECCoder Library 模块库

4.1 CAN 通讯模块

CAN 通讯模块用来配置和检查 CAN 通道工作状态，并用于发送和接收 CAN 消息。

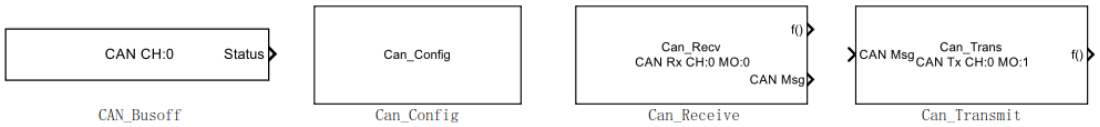


图.CAN 通讯模块

4.1.1 CAN 通讯配置模块 CAN Config

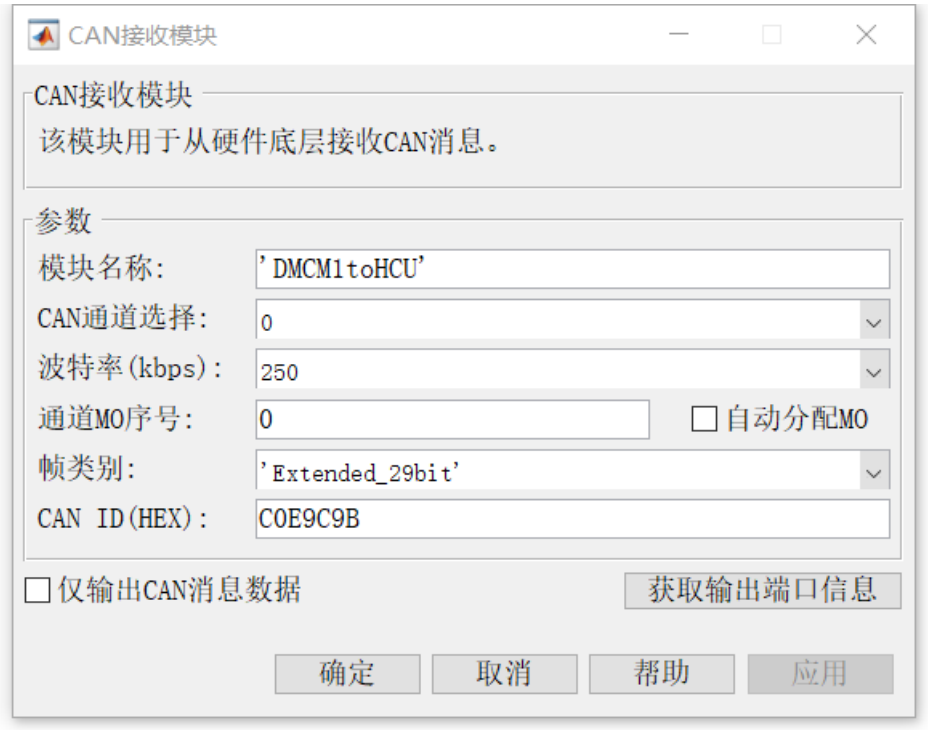
图.CAN 管理模块

CAN 通讯模块用于配置各 CAN 通道波特率，以及在代码生成中负责 CAN 相关文件配置。所以当模型中需要使用 CAN 相关功能时，必须首先添加 CAN 配置模块（CAN 发送/接收模块无法在没有 CAN 配置模块的情况下工作）。CAN 配置

模块在模型中最多只能存在一个，添加多个 CAN 模块时会提示错误。

CAN 配置模块中可对各通道波特率进行选择，共有四种波特率可选：125/250/500/1000 kbps。CAN 配置模块还可以直接打开 CAN 管理工具。

4.1.2 CAN 接收模块 CAN Receive



CAN 接收模块用于从底层接收 CAN 消息，各参数需满足如下条件：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- CAN 通道选择：硬件支持的 CAN 通道列表；
- 波特率：当前选定通道的波特率，共有 125/250/500/1000 kbps 四档可选。该值也可在 CAN 配置模块中统一修改，二者互相实时更新；
- 通道 MO 号：该接收模块所使用的 MO 序号，数值越低则代表其优先级越高。可以手动指定，如勾选自动分配，则按照模块创建的先后顺序自动排列。同一通道下的 CAN 收发模块 MO 序号不可重复，否则会提示错误信息；
- 帧类别：接收消息的 ID 格式，有标准帧”Standard”和扩展帧”Extended”可选。

- CAN ID：接收消息的 CAN ID，用十六进制数表示；
- 仅输出 CAN 消息数据：非勾选情况下输出端口 2 为 CAN Message 格式，输出包含 ID、数据长度、帧格式、消息数据等信息的结构体类型。输出的消息不能直接读取数据，需用 can unpack 模块进行解析。勾选状态下只输出维度为 8 的 CAN 消息数据，可直接读取；

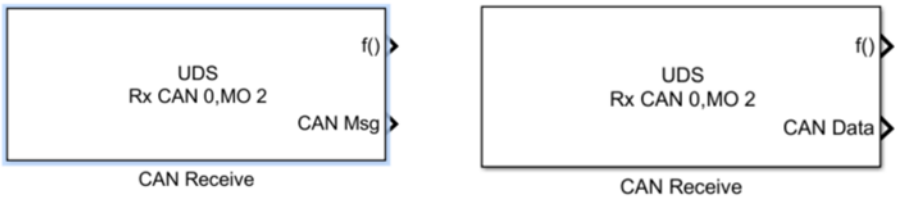


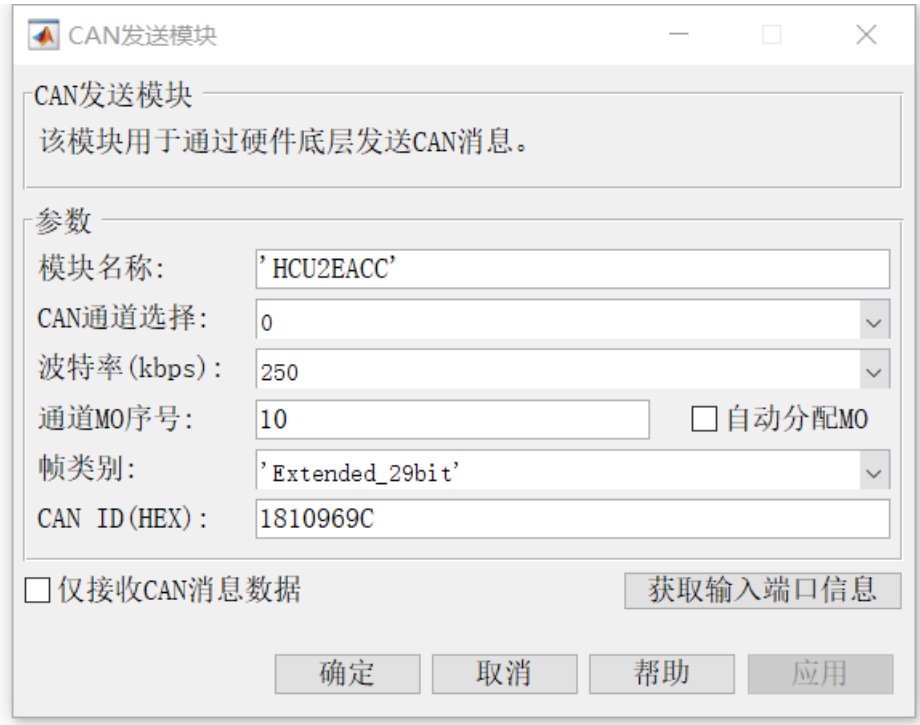
图.不同输出格式下的 CAN 接收模块

- 获取输出端口信息：在未勾选“仅输出 CAN 消息数据”情况下可用。如果模块输出端口已连接 can unpack 模块，点击该按钮则自动将 unpack 模块中的 CAN ID 和帧格式信息填入该模块，简化操作（仅在信号线直连情况下可用，如中间包含其他模块或被子系统分隔则无法获取）。

模型包含了两个输出端口：

- 输出端口 1 - f()：用于触发 triggered subsystem，当模块接收到信息之后 f()发出触发信号。Triggered subsystem 中的内容在生成代码中作为 CAN 接收回调函数生成。
- 输出端口 2 – CAN Msg/CAN Data：输出 CAN Message 或者 CAN 消息数据。

4.1.3 CAN 发送模块 CAN Transmit



CAN 接收模块用于通过底层发送 CAN 消息，各参数需满足如下条件：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- CAN 通道选择：硬件支持的 CAN 通道列表；
- 波特率：当前选定通道的波特率，共有 125/250/500/1000 kbps 四档可选。该值也可在 CAN 配置模块中统一修改，二者互相实时更新；
- 通道 MO 号：该接受模块所使用的 MO 序号，数值越低则代表其优先级越高。可以手动指定，如勾选自动分配，则按照模块创建的先后顺序自动排列。同一通道下的 CAN 收发模块 MO 序号不可重复，否则会提示错误信息；
- 帧类别：接收消息的 ID 格式，有标准帧”Standard”和扩展帧”Extended”可选。
- CAN ID：接收消息的 CAN ID，用十六进制数表示；
- 仅接收 CAN 消息数据：非勾选情况下输入端口为 CAN Message 格式，输出包含 ID、数据长度、帧格式、消息数据等信息的结构体类型。接收的消息不需用 can pack 模块进行打包。勾选状态下只接收维度为 8 的 CAN

消息数据；

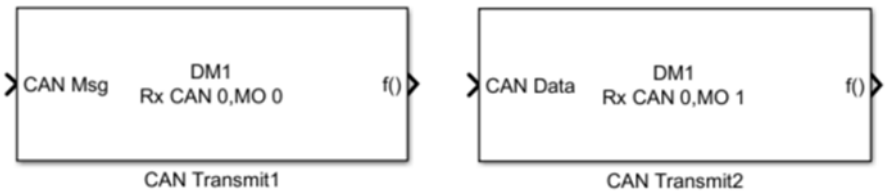


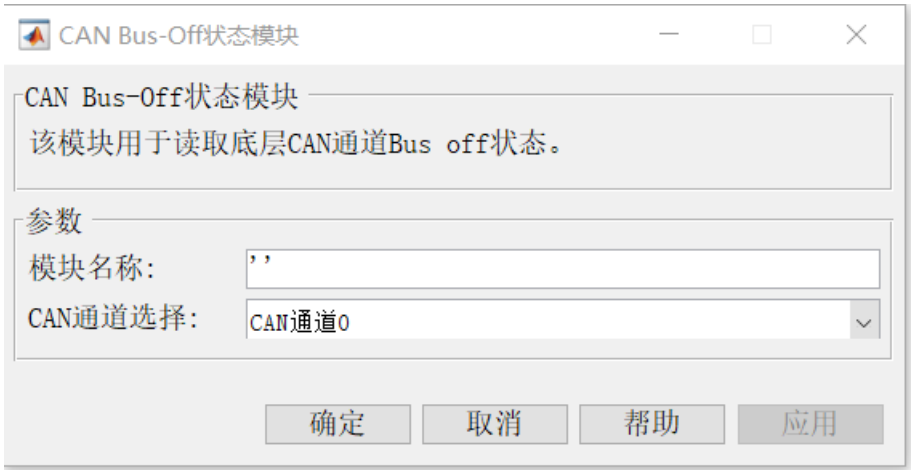
图.不同输入格式下的 CAN 发送模块

- 获取输入端口信息：在未勾选“仅接收 CAN 消息数据”情况下可用。如果模块输入端口已连接 can pack 模块，点击该按钮则自动将 pack 模块中的 CAN ID 和帧格式信息填入该模块，简化操作（仅在信号线直连情况下可用，如中间包含其他模块或被子系统分隔则无法获取）。

CAN 发送模块共有一输入一输出两个端口：

- 输出端口 1 - f()：用于触发 triggered subsystem，与 CAN 接收模块类似。
- 输入端口 1 – CAN Msg/CAN Data：输入 CAN Message 或者 CAN 消息数据用于发送。

4.1.4 CAN Bus-off 诊断模块

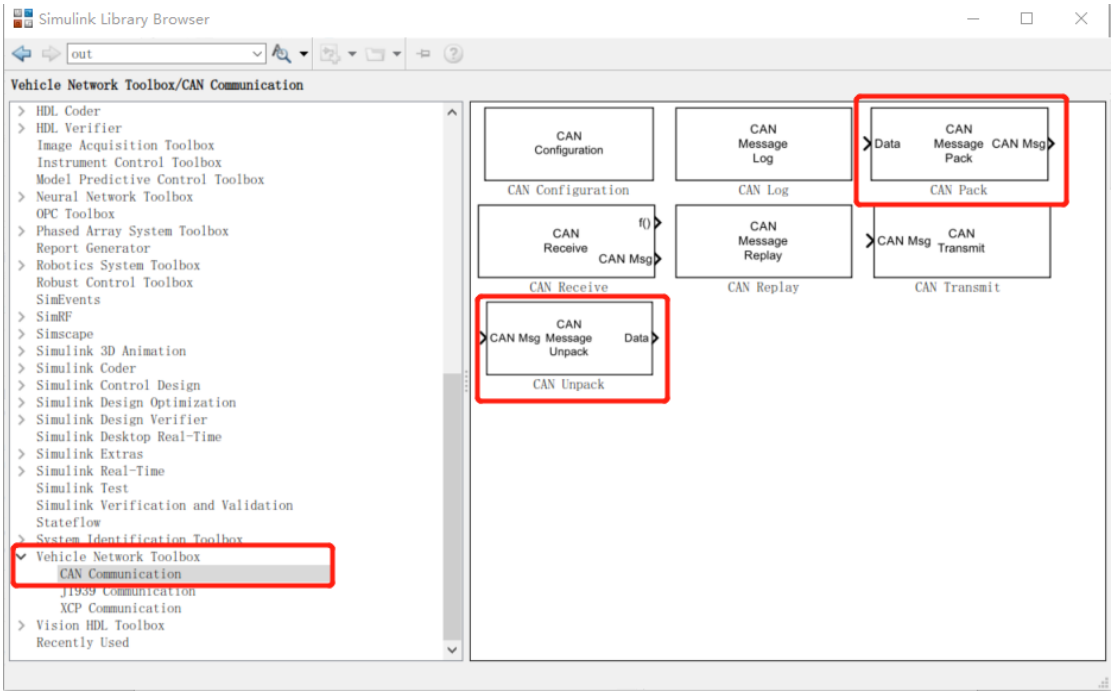


CAN Bus-off 诊断模块用于检查特定 CAN 通道的运行状况，各参数说明如下：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- CAN 通道选择：硬件支持的 CAN 通道列表。

模块有一个输出端口，用于指示 CAN 通道运行情况，输出的数据类型为 uint8。当输出为 1 时，说明该通道处于 Bus-off 状态；当输出为 0 时，说明该通道工作正常。

4.1.5 CAN Pack/Unpack 模块



在实际与 CAN 相关的应用中，与 ECCoder 中的收发模块配合使用的还有 Vehicle Network 工具箱中的 CAN Pack/Unpack 模块。这两个模块不仅可对 CAN 消息进行打包/拆包，将 ID、帧格式、数据域等信息从 CAN Message 中分离出来，还可以解析 DBC 文件，将 CAN 信号直接转化为物理值供模型使用。

详见该模块中 HELP 文档。

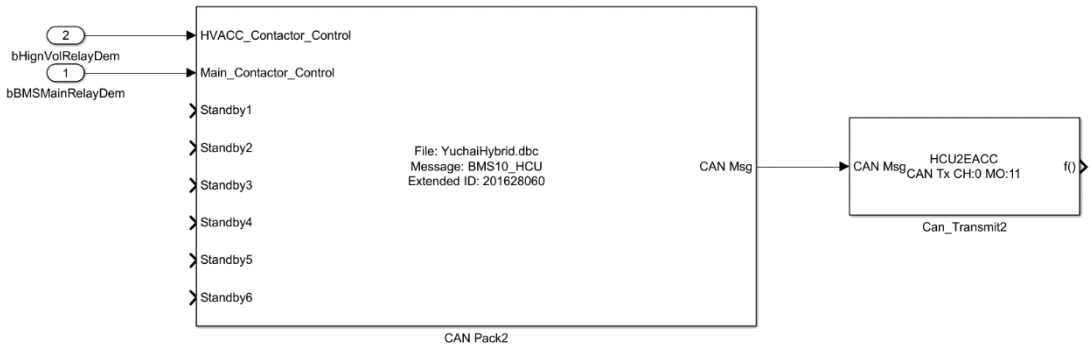


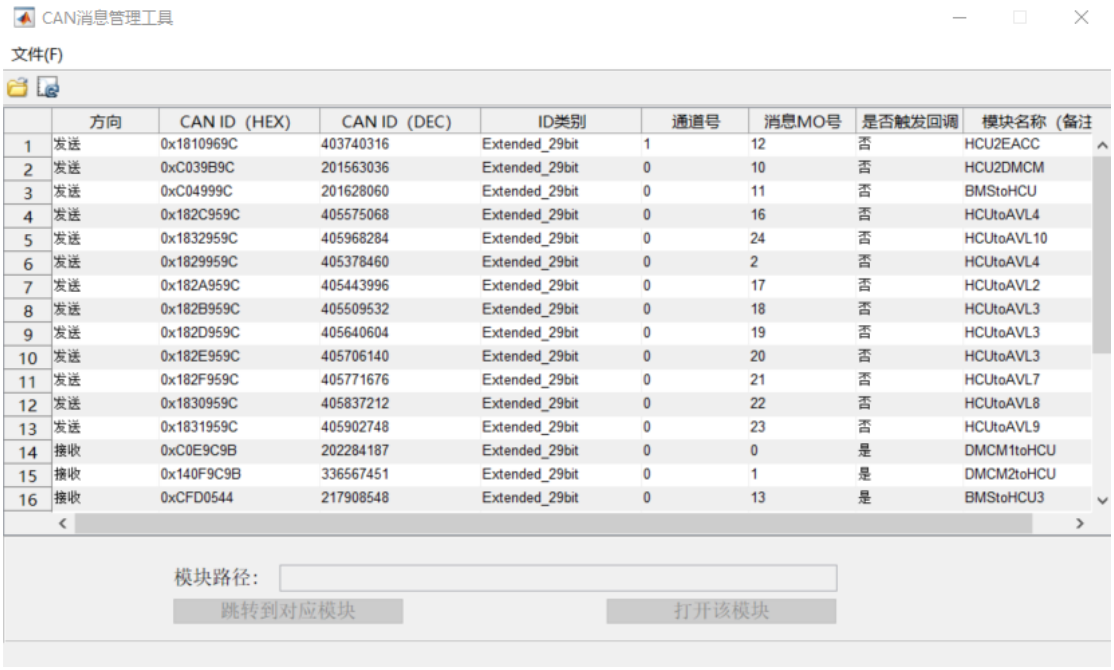
图.CAN Pack 模块与 CAN 发送模块配合使用

4.1.6 CAN 管理工具

CAN 管理工具提供了一个简洁的方式，用于管理模型中的 CAN 收发模块。工具有两种打开方式：

1.在 CAN Config 模块中，点击“打开 CAN 管理模块”按钮，即可直接使用。

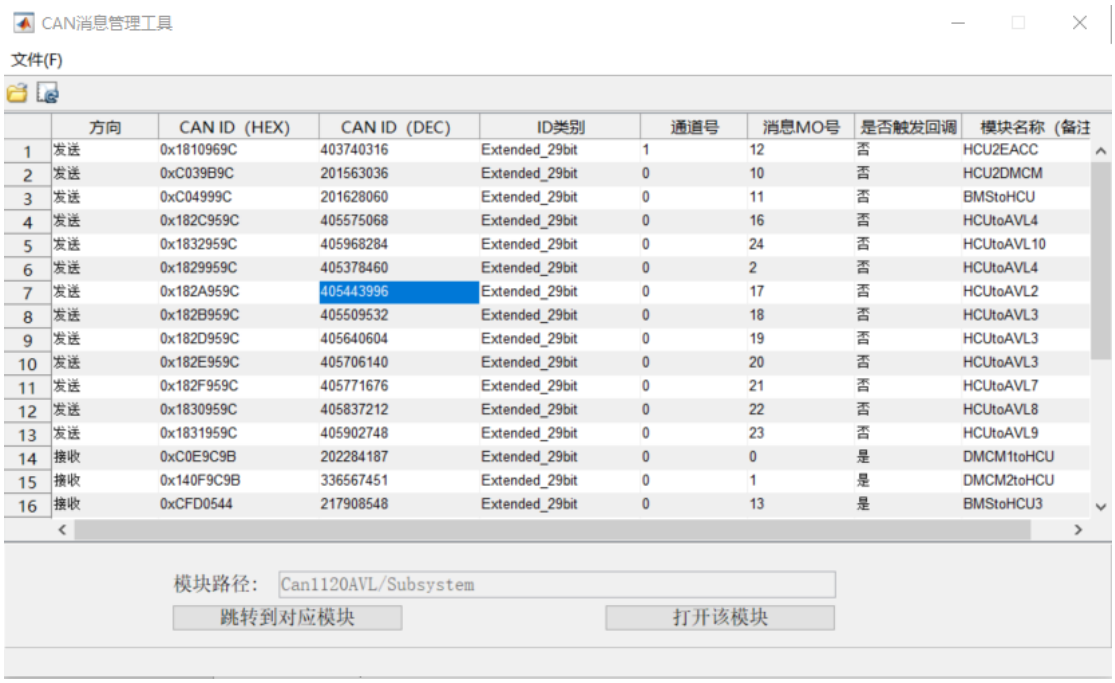
2.在命令行中输入：eccoder.canmgr，打开模块后，关联 slx 文件即可使用。



	方向	CAN ID（HEX）	CAN ID（DEC）	ID类别	通道号	消息MO号	是否触发回调	模块名称（备注
1	发送	0x1810969C	403740316	Extended_29bit	1	12	否	HCU2EACC
2	发送	0xC039B9C	201563036	Extended_29bit	0	10	否	HCU2DMCM
3	发送	0xC04999C	201628060	Extended_29bit	0	11	否	BMStoHCU
4	发送	0x182C959C	405575068	Extended_29bit	0	16	否	HCUtoAVL4
5	发送	0x1832959C	405968284	Extended_29bit	0	24	否	HCUtoAVL10
6	发送	0x1829959C	405378460	Extended_29bit	0	2	否	HCUtoAVL4
7	发送	0x182A959C	405443996	Extended_29bit	0	17	否	HCUtoAVL2
8	发送	0x182B959C	405509532	Extended_29bit	0	18	否	HCUtoAVL3
9	发送	0x182D959C	405640604	Extended_29bit	0	19	否	HCUtoAVL3
10	发送	0x182E959C	405706140	Extended_29bit	0	20	否	HCUtoAVL3
11	发送	0x182F959C	405771676	Extended_29bit	0	21	否	HCUtoAVL7
12	发送	0x1830959C	405837212	Extended_29bit	0	22	否	HCUtoAVL8
13	发送	0x1831959C	405902748	Extended_29bit	0	23	否	HCUtoAVL9
14	接收	0xC0E9C9B	202284187	Extended_29bit	0	0	是	DMCM1toHCU
15	接收	0x140F9C9B	336567451	Extended_29bit	0	1	是	DMCM2toHCU
16	接收	0xCFD0544	217908548	Extended_29bit	0	13	是	BMStoHCU3

图.CAN 管理工具界面

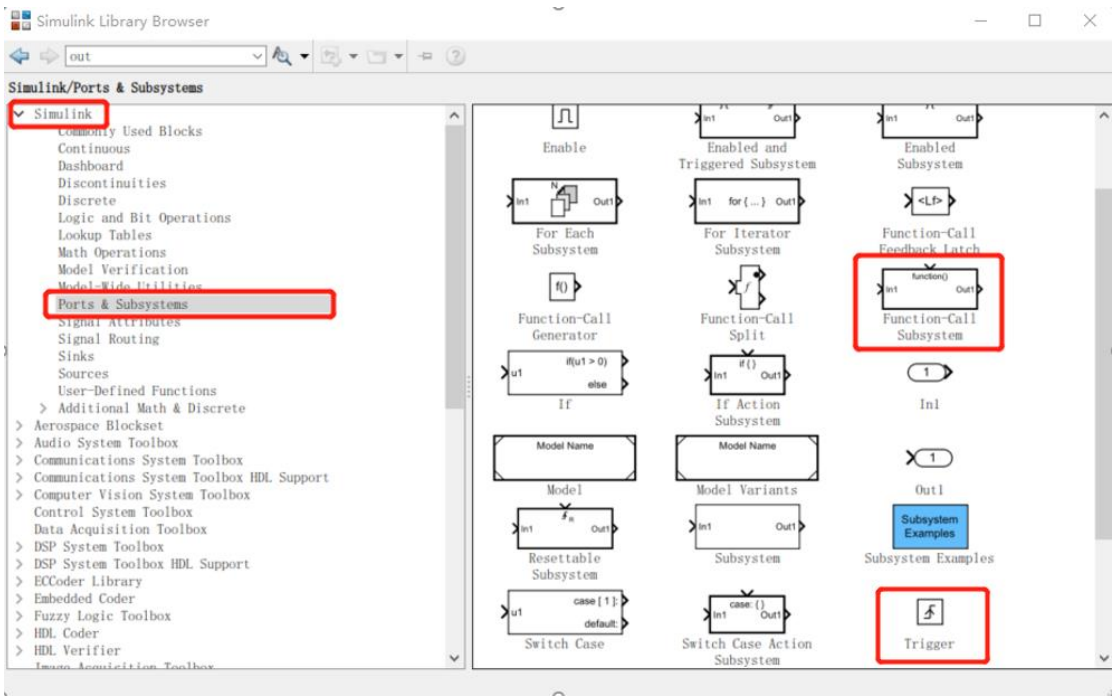
CAN 管理工具扫描关联模型中所有的 CAN 接收和发送模块，并将信息罗列在表格中。在表格中选中某一行，在模块路径中即显示对应 CAN 收发模块在模型中的路径，可通过下方按钮跳转至该模块或直接打开模块设置界面。



	方向	CAN ID (HEX)	CAN ID (DEC)	ID类别	通道号	消息MO号	是否触发回调	模块名称 (备注
1	发送	0x1810969C	403740316	Extended_29bit	1	12	否	HCU2EACC
2	发送	0xC039B9C	201563036	Extended_29bit	0	10	否	HCU2DMCM
3	发送	0xC04999C	201628060	Extended_29bit	0	11	否	BMStoHCU
4	发送	0x182C959C	405575068	Extended_29bit	0	16	否	HCUtoAVL4
5	发送	0x1832959C	405968284	Extended_29bit	0	24	否	HCUtoAVL10
6	发送	0x1829959C	405378460	Extended_29bit	0	2	否	HCUtoAVL4
7	发送	0x182A959C	405443996	Extended_29bit	0	17	否	HCUtoAVL2
8	发送	0x182B959C	405509532	Extended_29bit	0	18	否	HCUtoAVL3
9	发送	0x182D959C	405640604	Extended_29bit	0	19	否	HCUtoAVL3
10	发送	0x182E959C	405706140	Extended_29bit	0	20	否	HCUtoAVL3
11	发送	0x182F959C	405771676	Extended_29bit	0	21	否	HCUtoAVL7
12	发送	0x1830959C	405837212	Extended_29bit	0	22	否	HCUtoAVL8
13	发送	0x1831959C	405902748	Extended_29bit	0	23	否	HCUtoAVL9
14	接收	0xC0E9C9B	202284187	Extended_29bit	0	0	是	DMCM1toHCU
15	接收	0x140F9C9B	336567451	Extended_29bit	0	1	是	DMCM2toHCU
16	接收	0xCFD0544	217908548	Extended_29bit	0	13	是	BMStoHCU3

图.显示模块路径

4.1.7 Function-call Triggered 子系统（回调函数）



CAN 收发模块中均留有回调函数触发信号输出端口，回调函数在 simulink 中通过 Function-Call Subsystem 实现，可在模块库中直接找到该模块。也可通过在普通的 subsystem 中添加一个 Trigger 模块实现（Trigger 类型设置为 Function-call 即可）。

该类子系统原理为内部模型不随模型整体采样时间执行，只有当接收到一次有效的 function-call trigger 函数时，才会触发执行一次。当 CAN 收发模块成功接收或发送一次 CAN 消息时均会在 f()端口输出一次触发信号。将回调函数的内容存放在子系统中由 f()触发，即可实现回调功能。

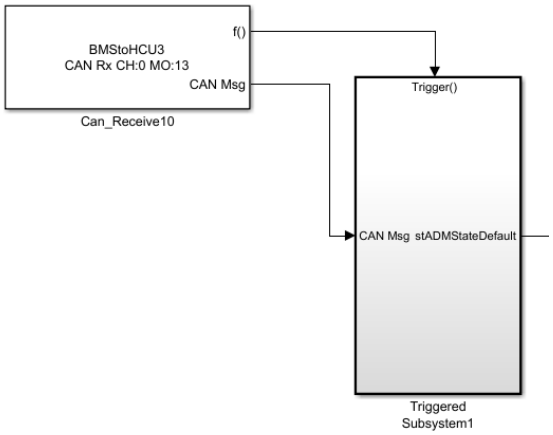


图.通过 Triggered Subsystem 实现回调函数功能

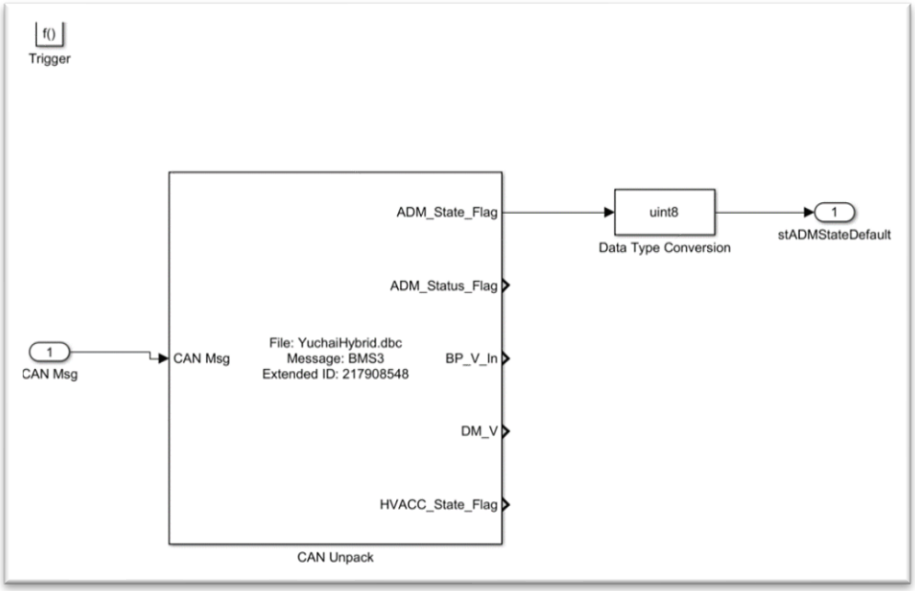


图.子系统内部模型

将 Unpack 模块放在回调中可避免每次采样任务都会执行解析 DBC 步骤，只有在接收到消息时才会进行解析。可大量减少解析次数，节省控制器计算资源（DBC 解析为位操作，占用算力较大）。

4.2 故障诊断模块

故障诊断模块用于向底层报告异常状态，并获取当前故障水平、读取异常状

态、读取 FID 状态。

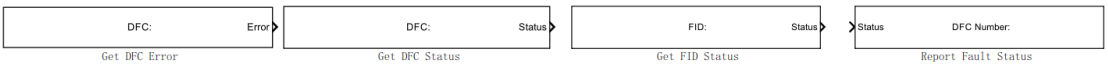


图.故障诊断模块

DFC(Diagnostic fault check)，故障检查项，每个 DFC 对应一个故障码。上层向诊断系统报告某一 DFC 状态（正常/异常），该状态也可被查询（正常/异常），诊断系统会结合之前状态和报告次数决定 DFC 当前状态（部分故障/完全故障/部分正常/完全正常），并在故障发生时禁用某些功能 FID(Function Identifier)，FID 实际上为一个标志位，有启用/禁用两种状态，不同 FID 对应着不同功能。

上层算法通过获取 DFC 状态和 FID 状态，进一步做出判断。

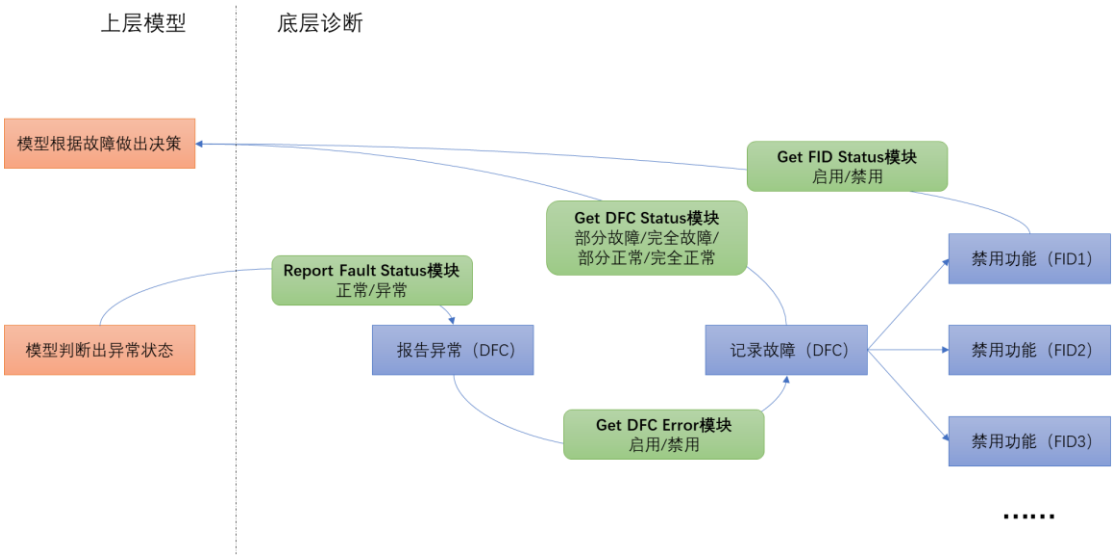
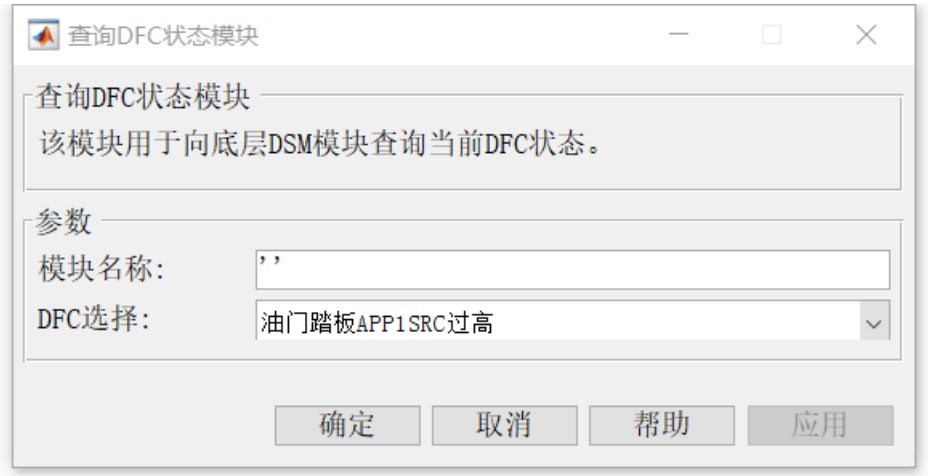


图.上层算法与故障诊断模块(DSM)进行信息交换

4.2.1 获取 DFC 状态模块 Get DFC Status



该模块用于获取所选择 DFC 的当前状态。

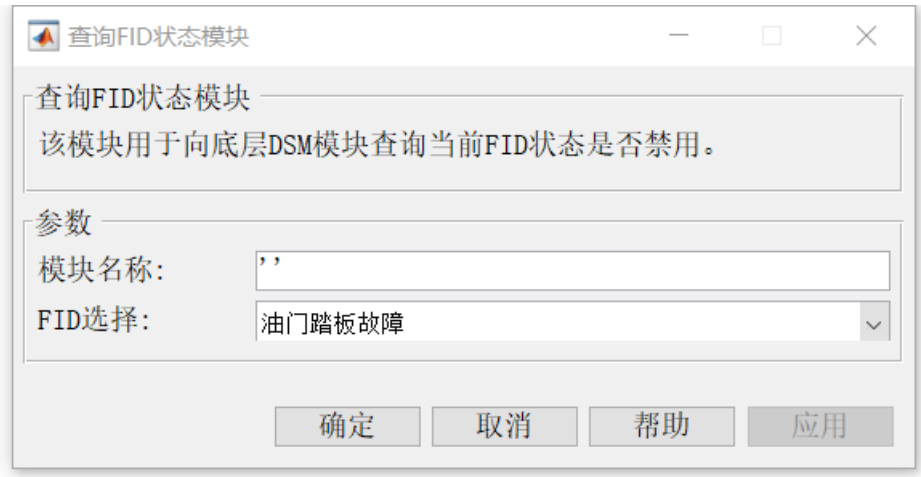
- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- DFC 选择：选择模块对应的 DFC。

模块输出端口为查询到的 DFC 状态（数据类型 uint8），对应关系如下表：

故障状态	值（uint8）
DFC 部分故障	1
DFC 完全故障	2
DFC 部分恢复（正常）	3
DFC 完全恢复（正常）	4

表.故障状态与对应值

4.2.2 获取 FID 状态模块 Get FID Status



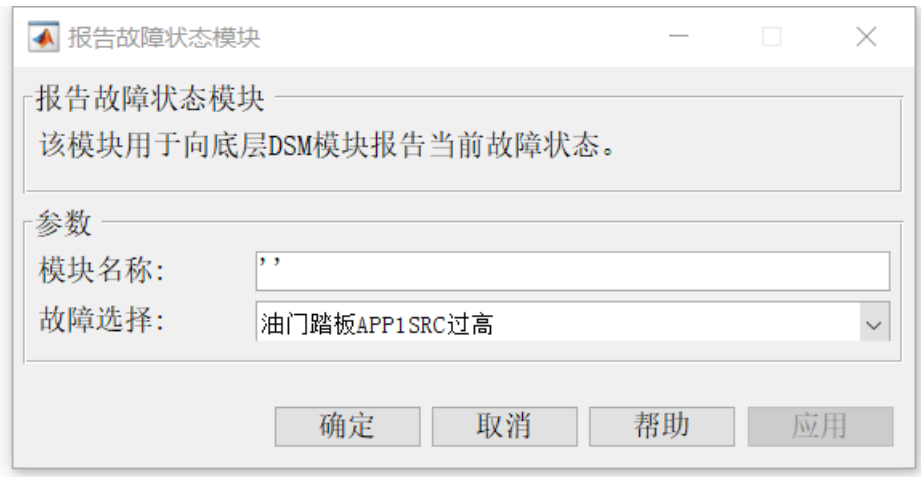
该模块用于查询 FID 禁用状态，模块参数：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- FID 选择：选择模块对应的 FID。

模块输出端口为查询到的 DFC 状态（数据类型 uint8），值为：

- 未触发 FID（FID 正常启用）- 0
- 触发 FID（FID 已被禁用） - 1

4.2.3 报告故障状态模块 Report Fault Status



该模块用于报告异常状态，模块参数：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- DFC 选择：选择模块对应的 DFC。

模块输入为故障状态（数据类型 uint8）：

- 正常 - 0
- 异常 - 1

4.2.4 获取 DFC 异常模块 Get DFC Error

该模块用于查询当前 DFC 是否存在异常，模块参数：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- DFC 选择：选择模块对应的 DFC。

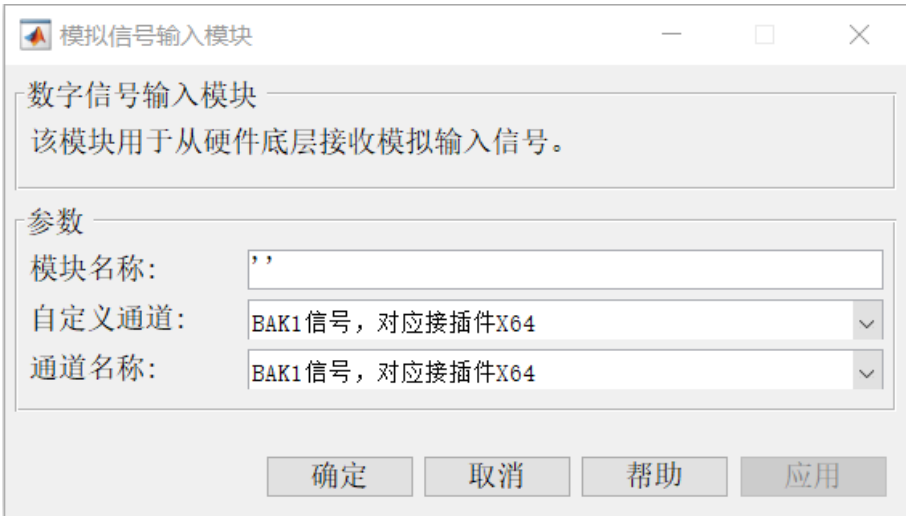
模块输出状态为故障状态（数据类型 uint8）：

- 正常 - 0
- 异常 - 1

4.3 输入输出接口模块

IO 接口模块为模型提供了读取线束管脚信号和向线束管脚发送信号的功能。IO 接口模块分为五种类型：数字信号输入、模拟信号输入、PWM 输入、高低边驱动输出和 PWM 驱动输出。

4.3.1 模拟信号输入模块 AD Input

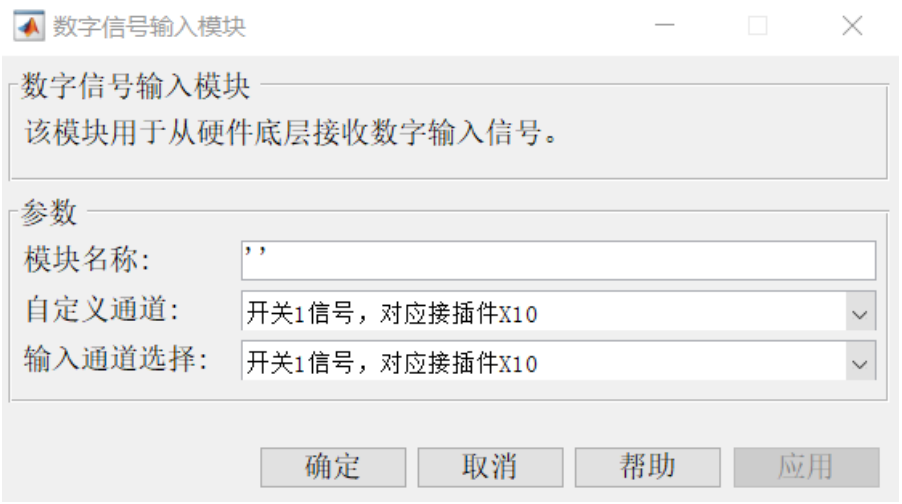


该模块可读取模拟输入信号，参数为：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 自定义通道：用户自定义通道名称，与实际硬件通道对应；
- 通道名称：该项为信号所对应的通道，后面数字为实际硬件管脚。

模块输出值即为获取到的信号量，数据类型为 uint16，精度为 1/4095 V/bit。

4.3.2 数字信号输入模块 Digital Input

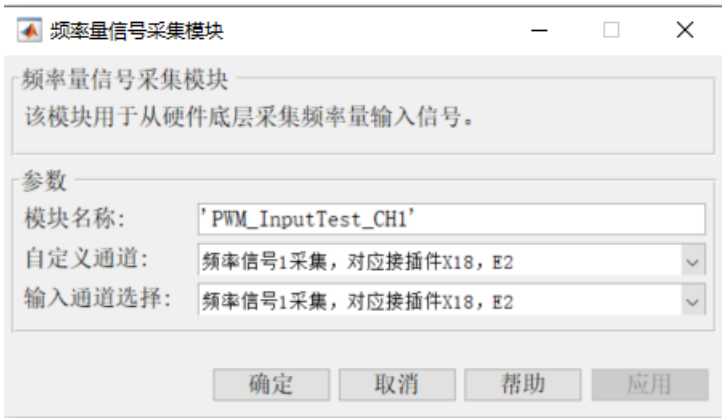


该模块可读取数字输入信号，参数为：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 自定义通道：用户自定义通道名称，与实际硬件通道对应；
- 通道名称：该项为信号所对应的通道，后面数字为实际硬件管脚。

模块输出值即为获取到的信号量，数据类型为 uint8。

4.3.3 PWM 信号输入模块 PWM Input



该模块可读取 PWM 信号，参数为：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 自定义通道：用户自定义通道名称，与实际硬件通道对应；

- 通道名称：该项为信号所对应的通道，后面数字为实际硬件管脚。

TC234 平台

在频率/占空比模式下，两个输出端口分别为频率和占空比。

- 频率：数据类型 uint32，精度 0.1Hz/bit；
- 占空比：数据类型 uint16，精度 0.01%/bit。

TC275 平台

在频率/占空比模式下，两个输出端口分别为周期和占空比。

- 周期：数据类型 uint32，单位:us，转换关系:1<->1us；
- 占空比：数据类型 uint16，单位:%，转换关系:0~10000<->0~100%。

4.3.4 高低边驱动输出模块 PSWT Output

该模块输出高低边驱动信号，参数为：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 自定义通道：用户自定义通道名称，与实际硬件通道对应；
- 输出通道选择：该项为信号所对应的通道，后面数字为实际硬件管脚；
- 输出通道诊断信息：勾选该选项后，模块会增加一个输出端口，用于输出通道诊断信息。诊断信息为数据类型为 uint8 的四维数组，所代表内

容如下表所示：

信号序号	诊断内容	值含义
0	开路 *DSM_DIAG_OPENLOAD*	0x00:无故障 0x01:有故障 0xFF:故障状态未知或无效
1	短路 *DSM_DIAG_SHORTCIRCUIT*	
2	短路到地 *DSM_DIAG_SHORTTOGND*	
3	短路到电源 *DSM_DIAG_SHORTTOPWR*	

表中内容详见 HardwareLib.h 中枚举类型 *DSM_idxDiagTypeE_TYPE* 与 *DSM_stDiagE_TYPE*，高低边驱动输出通道诊断功能函数为 *PSwtDrv_GetChanDiagInfo*。

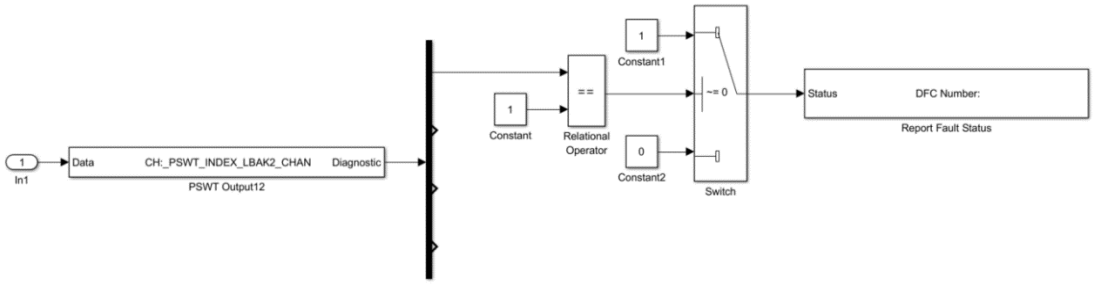
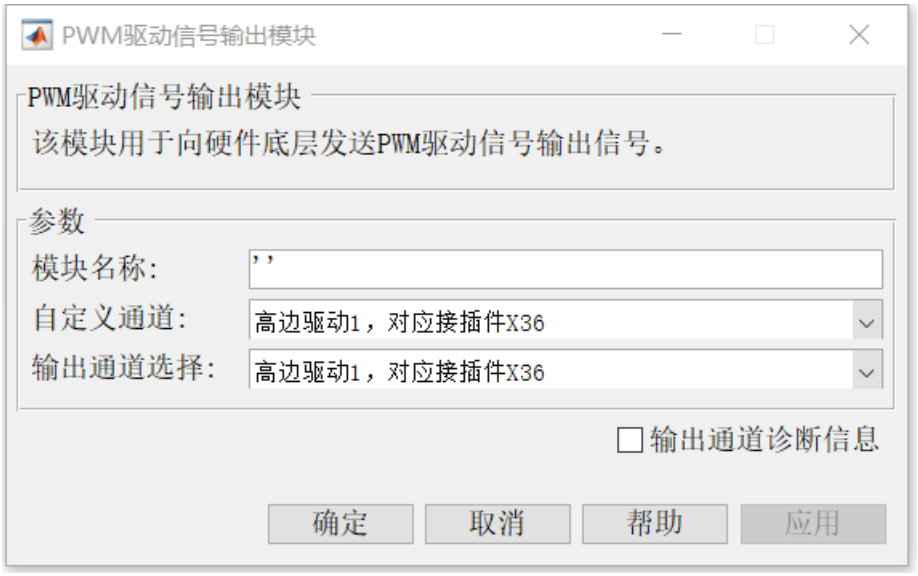


图.高低边输出诊断功能与 Report DFC 模块配合使用样例

模块输入值即为信号输出，数据类型为 uint8，输入为 1 时通道打开，否则为关闭。

4.3.5 PWM 驱动输出模块 PWM Output



该模块输出高低边驱动信号，参数为：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 自定义通道：用户自定义通道名称，与实际硬件通道对应；
- 输出通道选择：该项为信号所对应的通道，后面数字为实际硬件管脚；
- 输出通道诊断信息：勾选该选项后，模块会增加一个输出端口，用于输出通道诊断信息。诊断信息为数据类型为 uint8 的四维数组，所代表内容如下表所示：

信号序号	诊断内容	值含义
0	开路 *DSM_DIAG_OPENLOAD*	0x00:无故障 0x01:有故障 0xFF:故障状态未知或无效
1	短路 *DSM_DIAG_SHORTCIRCUIT*	
2	短路到地 *DSM_DIAG_SHORTTOGND*	
3	短路到电源 *DSM_DIAG_SHORTTOPWR*	

表中内容详见 HardwareLib.h 中枚举类型 *DSM_idxDiagTypeE_TYPE* 与 *DSM_stDiagE_TYPE*，高低边驱动输出通道诊断功能函数为 *PPWMDrv_GetChanDiagInfo*。

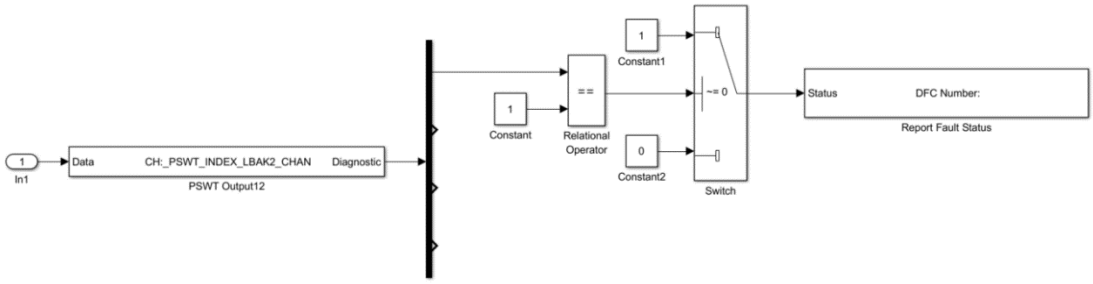


图.高低边输出诊断功能与 Report DFC 模块配合使用样例

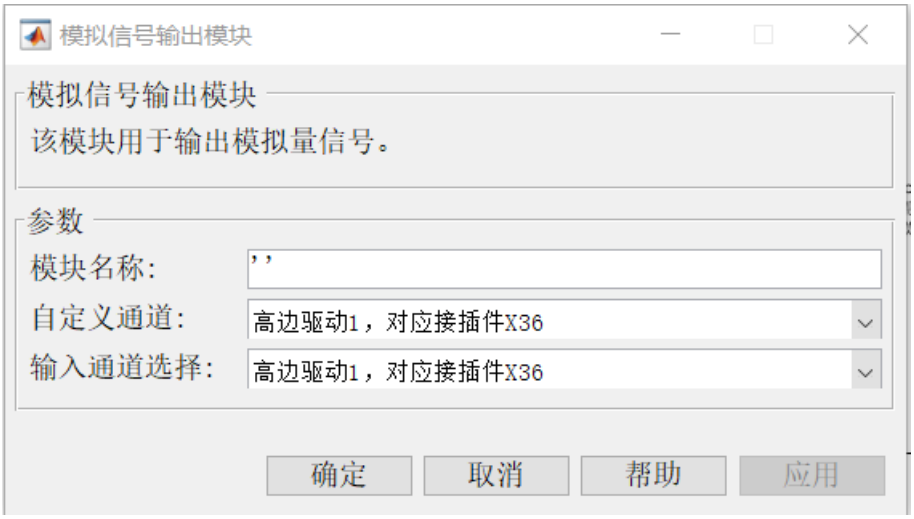
TC234 平台

模块有两个输入值，第一个端口 Frequency 为 PWM 信号频率，数据类型为 uint32，精度为 0.1Hz/bit；第二个端口 Duty 为占空比，数据类型为 uint16，精度为 0.01%/bit。

TC275 平台

模块有两个输入值，第一个端口 Frequency 为 PWM 信号频率，数据类型为 uint32，精度为 0.1Hz/bit；第二个端口 Duty 为占空比，数据类型为 uint16，精度为 0.01%/bit。

4.3.6 模拟量输出模块 AD Output



该模块输出模拟量信号，参数为：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 自定义通道：用户自定义通道名称，与实际硬件通道对应；
- 输出通道选择：该项为信号所对应的通道，后面数字为实际硬件管脚。

模块有一个输入端口，数据类型为 uint16，电压值精度 1/4095V/bit。

4.4 变量模块

变量模块替代了 simulink 中内建的 constant 和 lookup table 模块，通过在 workspace 中新建 Simulink.Parameter 变量，让用户在代码生成时能直接生成可通过 CAN 标定的标定变量或保存在 EEPROM 中的变量。

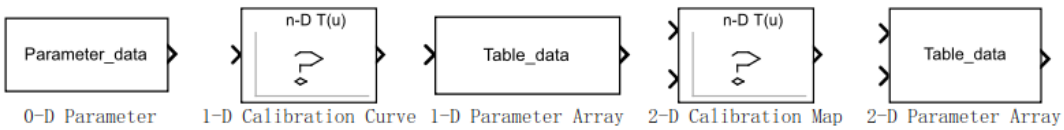
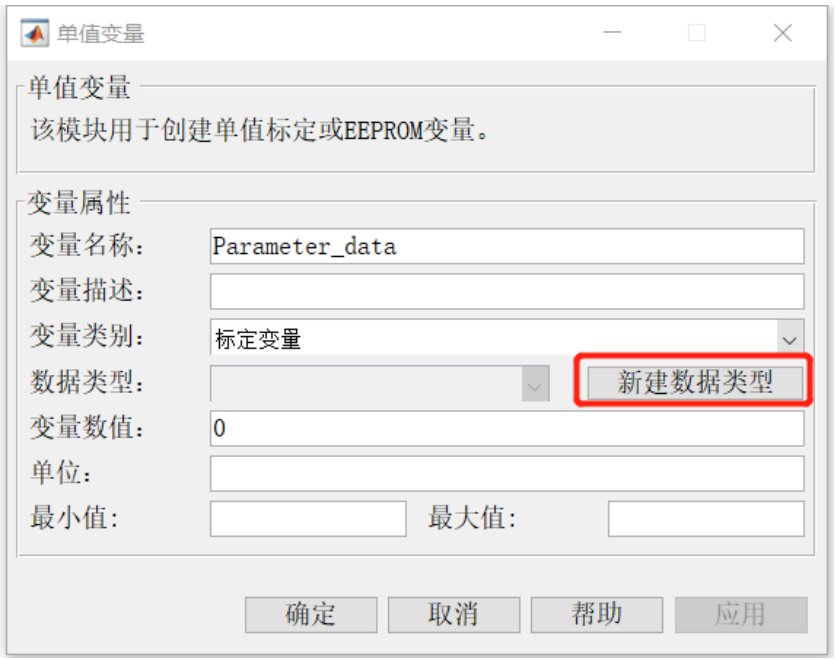


图.变量模块

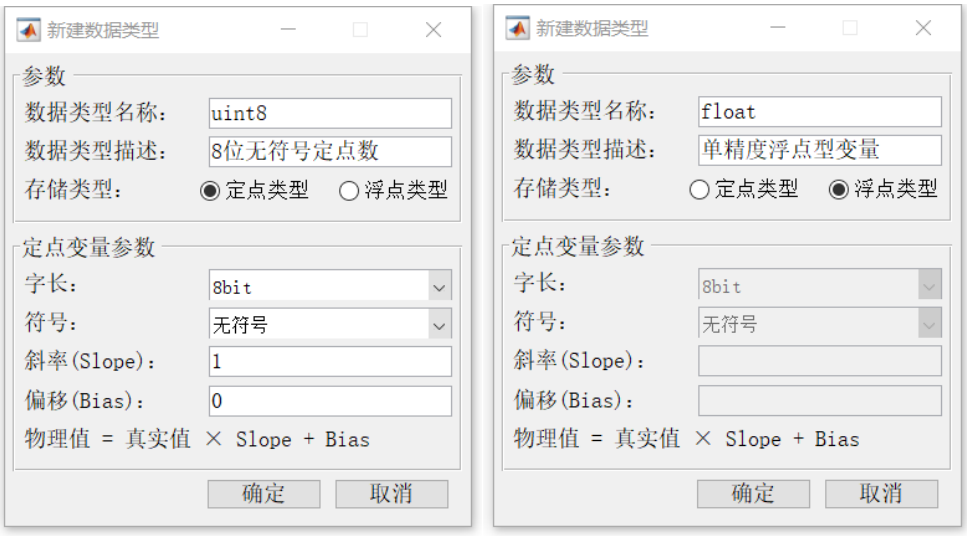
4.4.1 新建数据类型

在新建变量之前需要首先新建数据类型，在每个变量模块中均有【新建数据类型】按钮。点击按钮后在弹出页面中新建数据类型。



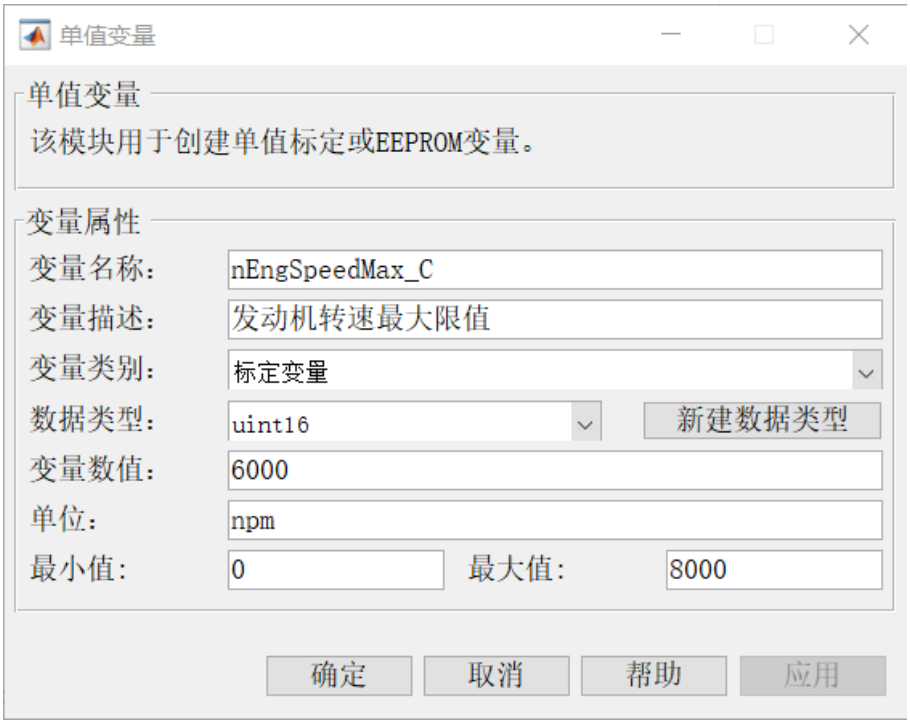
添加数据类型首先需要为数据类型指定名称，名称命名规则与 matlab 变量命名规则一致，还可为数据类型添加描述（可选）。数据类型提供两种类别：定点与浮点。如果选择为定点类型，则需要在定点变量参数一栏为类型指定长度和转换关系。

定点类型支持字长为 8 位、16 位、32 位三种，可选择有符号与无符号。斜率和偏移为必填项，以确定物理值和真实值之间关系。在界面中用户以物理值为变量赋值，在生成的代码中变量则以真实值生成。如选择浮点类型，则无需指定其他参数。浮点类型默认为且只支持单精度类型(single/float)。



点击确定后数据类型即新建至 workspace 中，workspace 中的数据类型可被所有标定变量模块使用，并不限于当前模块。

4.4.2 单值变量模块 0-D Parameter（不推荐）

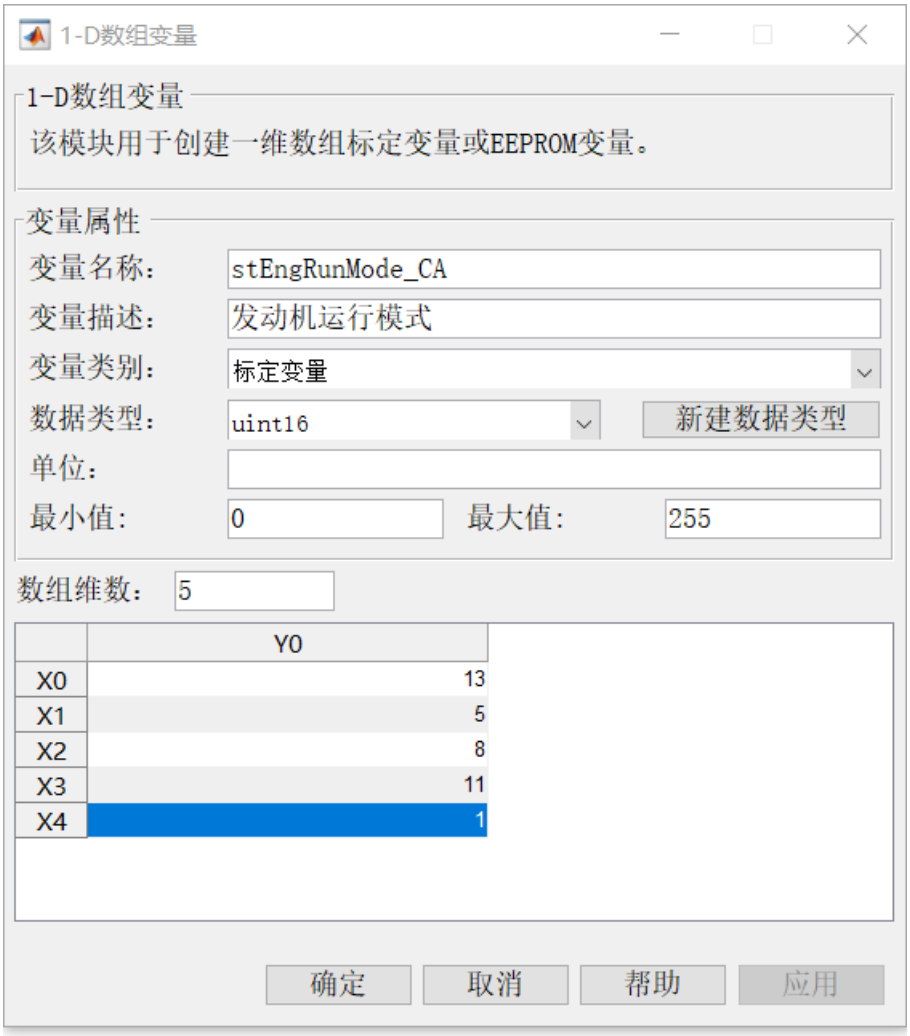


该模块用于生成单值变量，各参数需满足如下要求：

- 变量名称一栏为必填，且不可使用默认值(Parameter_data)；
- 变量描述为选填，起注释作用；
- 变量类别为必选，选择变量是标定变量或 EEPROM 变量；
- 数据类型为必选，弹出菜单中会显示 workspace 中所有数据类型供选择，若当前没有数据类型则需新建；
- 变量数据为必填，且必须为有效数字；
- 单位为选填；
- 最小值最大值为必填，最小值需要小于最大值，且最大值最小值不能超过当前数据类型的表示范围。例：数据类型为 8 位无符号定点数，偏移为 10，斜率为 1，则其表示范围为[10,265]。如最大值最小值超过该范围则点击确定时会报错。

点击确定后，模块会对各项参数是否合法以及变量数值是否在范围内进行校验，如果存在错误则会弹出提示；如果校验通过则会在 workspace 中创建该变量。

4.4.3 一维数组变量模块 1-D Parameter Array（不推荐）



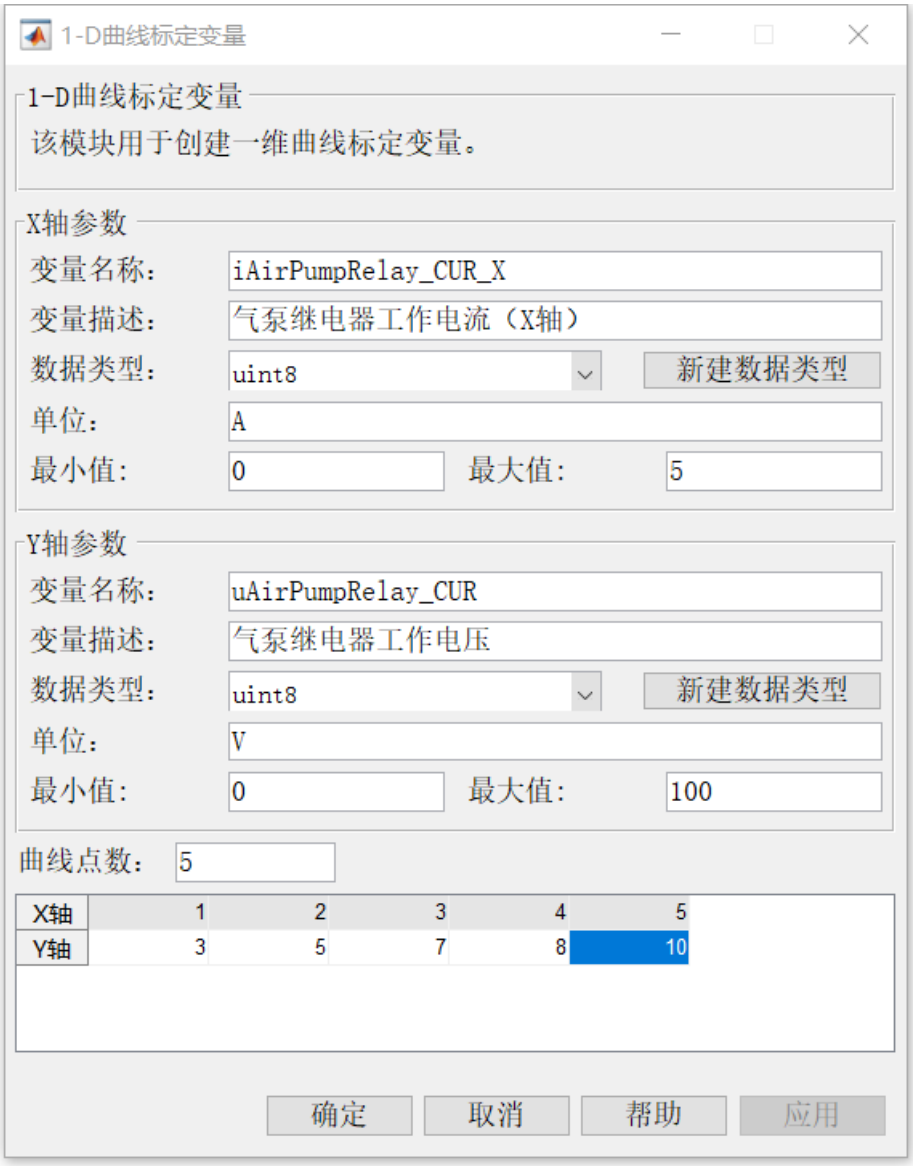
该模块用于生成一维数组变量，各参数需满足如下要求：

- 变量名称一栏为必填，且不可使用默认值(Table_data)；
- 变量描述为选填，起注释作用；
- 变量类别为必选，选择变量是标定变量或 EEPROM 变量；
- 数据类型为必选，弹出菜单中会显示 workspace 中所有数据类型供选择，若当前没有数据类型则需新建；
- 单位为选填；
- 最小值最大值为必填，最小值需要小于最大值，且最大值最小值不能超过当前数据类型的表示范围；
- 数组维数为必填（默认为 1），该参数决定了数组中的元素数量，该值改变时，下方的数据区大小也会随之改变。

- 数据区内所有单元内值均需满足数据类型和最小值和最大值范围。

点击确定后，模块会对各项参数是否合法以及变量数值是否在范围内进行校验，如果存在错误则会弹出提示；如果校验通过则会在 workspace 中创建该变量数组。在输入端口输入索引数，输出端口即会输出数组中相应值。输入必须为整数，且不得超过数组维数。

4.4.4 一维曲线标定模块 1-D Calibration Curve（不推荐）



该模块用于生成曲线标定变量，并对曲线进行查表操作，各参数需满足如下要求：

- 变量名称一栏为必填，且不可使用默认值(X_Data,Y_Data)；
- 变量描述为选填，起注释作用；

- 数据类型为必选，弹出菜单中会显示 workspace 中所有数据类型供选择，若当前没有数据类型则需新建；
- 单位为选填；
- 最小值最大值为必填，最小值需要小于最大值，且最大值最小值不能超过当前数据类型的表示范围；
- 曲线点数为必填（默认为 1），该参数决定了曲线(X/Y 轴)各有多少点；
- 数据区内 X 轴和 Y 轴单元格内值需分别满足上方对应的 X 轴/Y 轴数据类型和最小值和最大值范围。

点击确定后，模块会对各项参数是否合法以及变量数值是否在范围内进行校验，如果存在错误则会弹出提示；如果校验通过则会在 workspace 中创建 X 轴 Y 轴两个数据变量。该模块会根据曲线对输入进行查表后输出，对于曲线 X 轴外的输入，会使用端值进行输出；对于范围内的输入，则通过线性查表输出。

4.4.5 二维数组变量模块 2-D Parameter Array（不推荐）

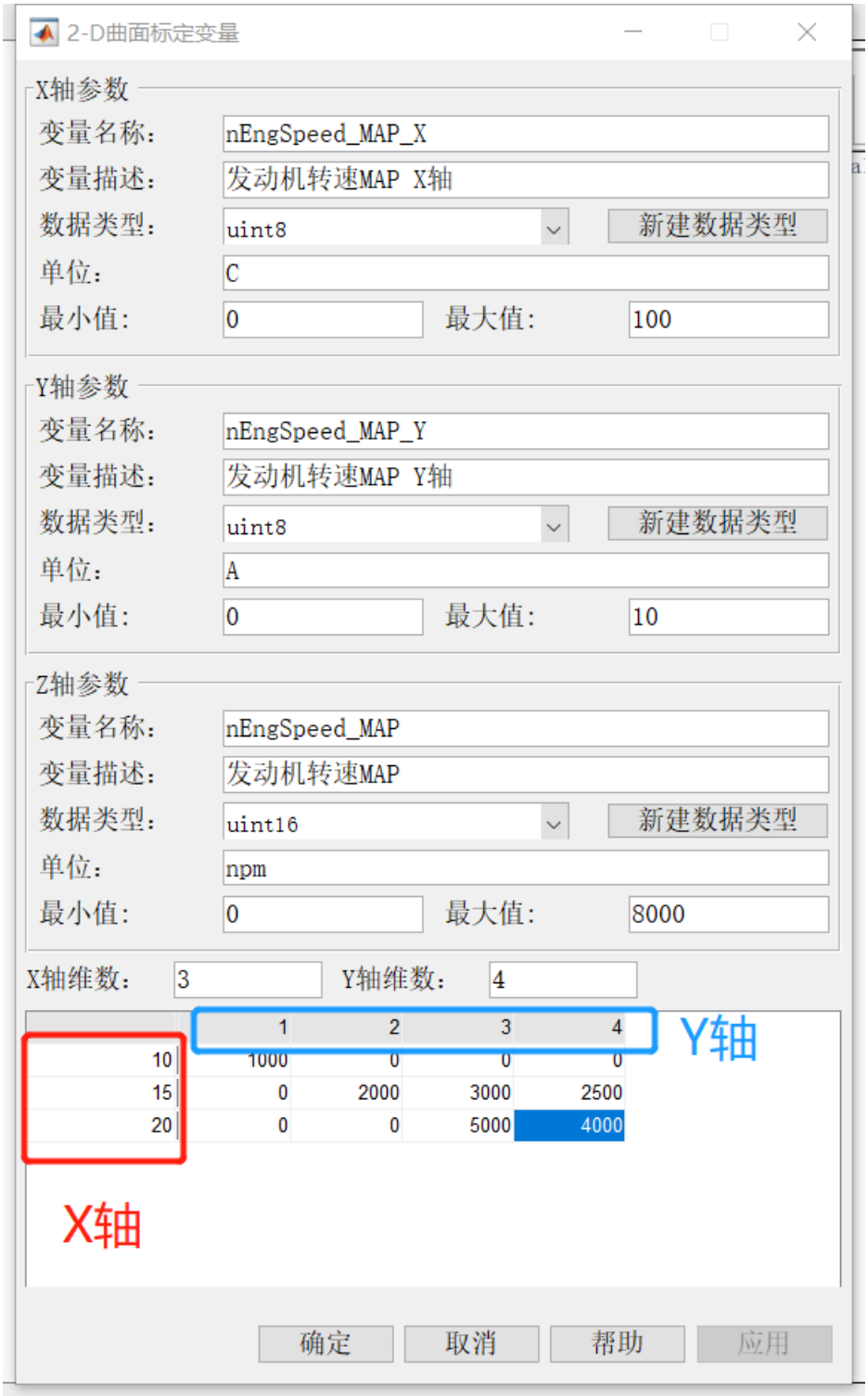
该模块用于生成二维数组变量，并对数组值进行索引，各参数需满足如下要求：

- 变量名称一栏为必填，且不可使用默认值(Table_data)；
- 变量描述为选填，起注释作用；
- 变量类别为必选，选择变量是标定变量或 EEPROM 变量；
- 数据类型为必选，弹出菜单中会显示 workspace 中所有数据类型供选择，若当前没有数据类型则需新建；
- 单位为选填；
- 最小值最大值为必填，最小值需要小于最大值，且最大值最小值不能超过当前数据类型的表示范围；
- X 轴维数/Y 轴维数为必填，决定二维数组大小，底部数据区域大小会随之变化；

■ 数据区内所有单元格内数据需满足上方数据类型和最小值和最大值范围。

点击确定后，模块会对各项参数是否合法以及变量数值是否在范围内进行校验，如果存在错误则会弹出提示；如果校验通过则会在 workspace 中创建一个二维数组变量。该模块会根据两个输入端口进行索引，第一个端口为 X 轴索引，第二个端口为 Y 索引，索引必须为整数且不超过 X/Y 轴维度。

4.4.6 二维 Map 标定模块 2-D Calibration Map（不推荐）



该模块用于生成曲面标定变量，并对曲面进行查表操作，各参数需满足如下要求：

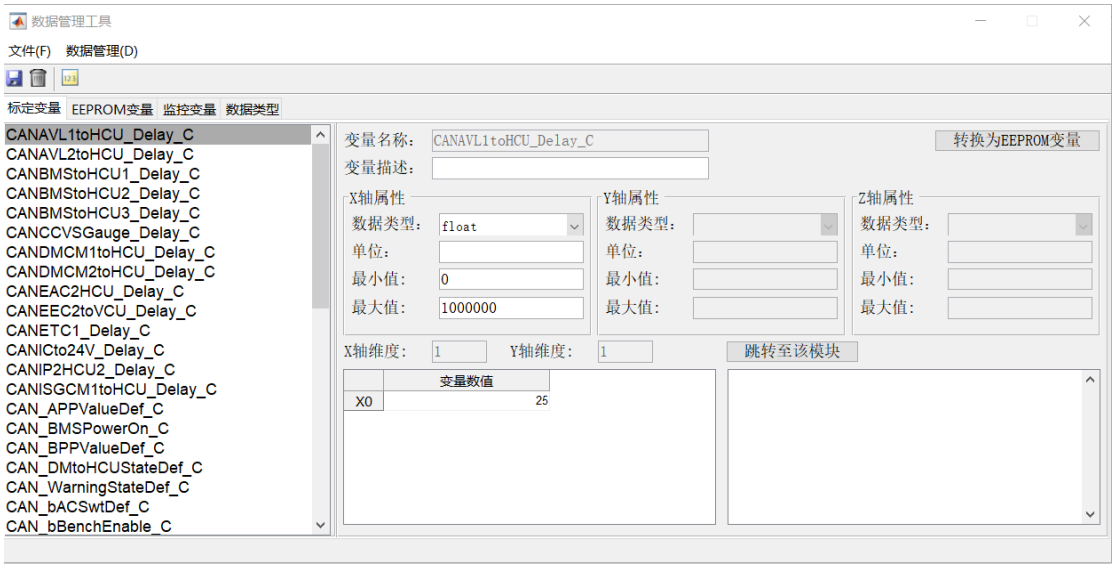
- 变量名称一栏为必填，且不可使用默认值(X_Data,Y_Data, Z_Data)；
- 变量描述为选填，起注释作用；

- 数据类型为必选，弹出菜单中会显示 workspace 中所有数据类型供选择，若当前没有数据类型则需新建；
- 单位为选填；
- 最小值最大值为必填，最小值需要小于最大值，且最大值最小值不能超过当前数据类型的表示范围；
- X/Y 轴维数为必填（默认为 1），该参数决定了曲面(X/Y 轴)各有多少点；
- 数据区内 X 轴/Y 轴/Z 轴单元格内值需分别满足上方对应的 X 轴/Y 轴/Z 轴数据类型和最小值和最大值范围。

点击确定后，模块会对各项参数是否合法以及变量数值是否在范围内进行校验，如果存在错误则会弹出提示；如果校验通过则会在 workspace 中创建 X 轴/Y 轴/Z 轴三个数据变量。该模块会根据曲线对输入进行查表后输出，对于范围外的输入，会使用端值进行输出；对于范围内的输入，则通过线性查表输出。

4.4.7 数据管理工具 Data Manager

数据管理模块用于管理模型和 workspace 中的数据，通过在命令行中输入 eccoder.datamgr 打开。



打开数据管理工具后，即会自动扫描 workspace 中变量，显示在视图当中。点击文件-打开，选择 slx 文件与数据管理工具进行关联。关联后可显示变量在模型中的使用情况并定位到模型中路径。还可查看数据类型与变量之间的引用关系。同时，也可对变量、参数等进行修改。

4.4.8 使用脚本新建变量（推荐）

除了采用特定模块在模型中建立和使用标定/EEPROM 变量的方式外，ECCoder 提供了采用脚本/命令行方式新建变量的方式。通过脚本建立的 Parameter 型变量与通过上述章节模块建立的变量具有相同的属性，且同样保存在 Matlab Workspace 中。与之不同的是，命令行建立变量无需使用 ECCoder Library 中 Parameter 模块，可直接在 Constant, Lookup Table 等 simulink 内建模块中使用，更加贴合 simulink 原生方式。用户可根据实际需求和使用习惯来决定用何种方式新建和管理变量。

以下代码用于新建标定变量 a：

```
a=ECCoder_MemCtrl.Parameter;
a.CoderInfo.CustomStorageClass = 'Global';
a.CoderInfo.CustomAttributes.MemorySection = 'MemConst';
a.Description = 'description';
a.Unit = 'V';
a.DataType = 'double';
a.Value = [10 20];
a.Max = 100;
a.Min = 0;
```

以下代码用于新建 EEPROM 变量 b：

```
b=ECCoder_MemCtrl.Parameter;
b.CoderInfo.CustomStorageClass = 'Global';
b.CoderInfo.CustomAttributes.MemorySection = 'MemEEPROM';
b.Description = 'description';
b.Unit = 'V';
b.DataType = 'double';
b.Value = [10 20];
b.Max = 100;
b.Min = 0;
```

红色部分为必要属性，黑色部分为可选属性。对于由脚本新建的变量，除保存为.mat 的方式外，可将上述代码保存为.m 脚本文件，在加载模型之前运行。

4.4.9 变量模块使用注意事项

4.4.9.1 模型的保存

使用标定变量模块创建的变量都保存在 workspace 中，并非保存在模块中。模块只是对 workspace 中的变量值进行读取。当 Matlab 关闭后，workspace 即会自动清空，再次打开工程文件时标定变量模块会找不到变量。

所以每次保存模型工程文件 slx 之前均需要将 workspace 保存为 mat 文件，在下次打开模型工程时首先将对应的 mat 加载至 workspace 中，模型才可以正常运行。

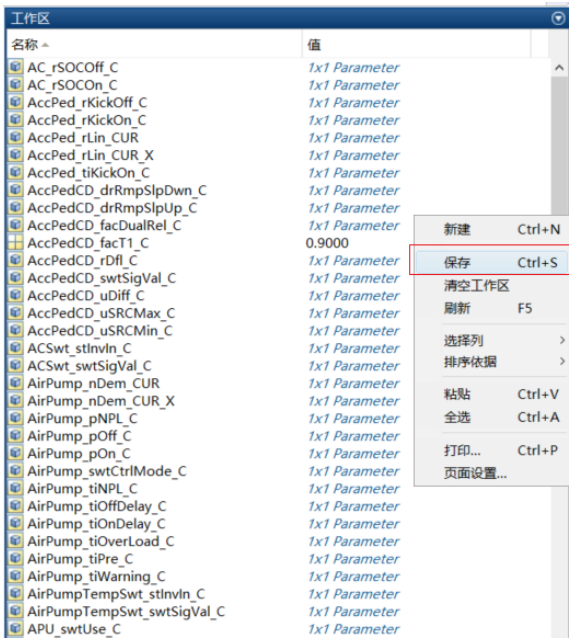


图.保存 Workspace 中数据类型和变量

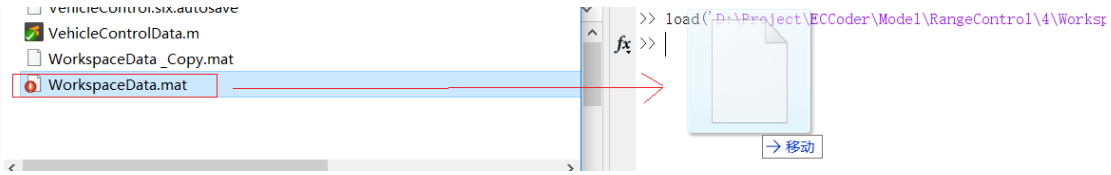
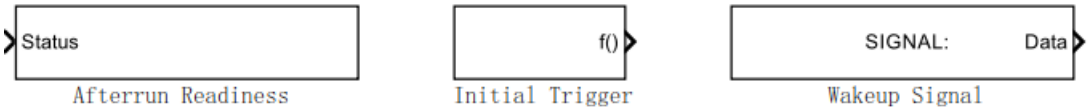


图.加载 mat 文件至 workspace

4.4.9.2 不支持的模块

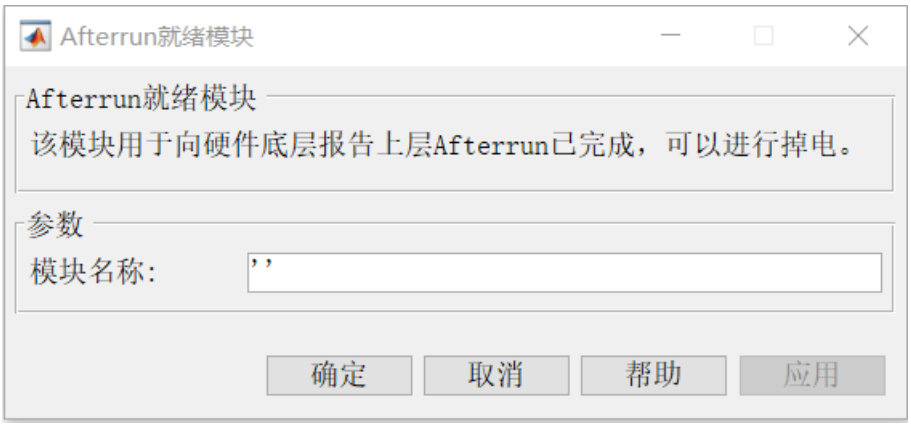
Simulink 中存在着某些模块不支持 Parameter 类型变量作为输入参数，目前已知传递函数相关模块大部分均为此类。

4.5 系统管理模块



系统管理模块将控制器运行相关功能封装起来，使上层能获取到控制器当前运行状态并进行初始化、掉电等操作。系统管理功能中共包含三个模块：Afterrun 就绪模块、初始化模块、唤醒源信号模块。

4.5.1 Afterrun 就绪模块 Afterrun Readiness



当系统唤醒源全部无输入时，控制器会进入 Afterrun 状态，即掉电前的预备过程。Afterrun 过程分为上层和底层两部分，上层首先执行掉电前流程，如关闭或复位执行器、向其他模块发送关闭通知等动作；在该部分动作执行完毕后，上层算法停止不再工作，控制器完全由底层接管。在执行完 EEPROM 数据保存、故障及运行信息存储等操作后，控制器掉电。

Afterrun 就绪模块用于向控制器报告上层 Afterrun 流程已执行完毕，控制器可以由底层接管并掉电。该模块有一个输入端口，只接收数据类型为 uint8 的输入，当该模块输入为 0 时，控制器正常运行；当该模块输入为 1 时，控制器掉电。

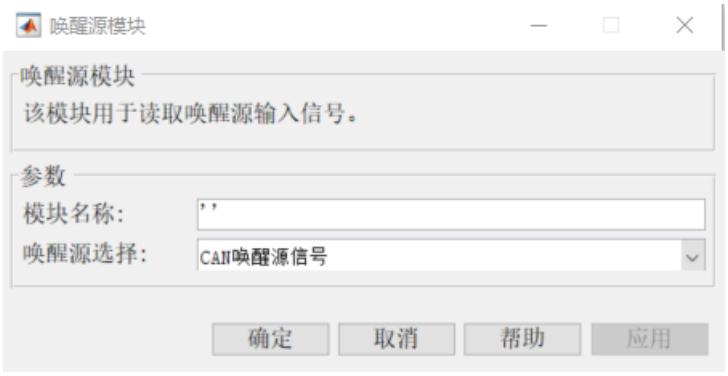
注意：一旦该模块输入为 1，上层算法将在下一 step 不再运行。

4.5.2 初始化模块 Initial Trigger

Simulink 中，模型生成代码通常随调度任务周期执行；但是对于 Triggered Sub-system，子系统中内容由 trigger 信号独立触发（详见 4.1.7 Function-call Triggered 子系统）。

在建模过程中，有一些初始化操作希望在模型整体周期执行之前完成，且只进行一次。初始化模块即提供一个 function-call trigger 信号，用于触发子系统，使子系统中内容仅在初始化时执行一次。

4.5.3 唤醒源信号模块 Wakeup Signal



唤醒源信号模块用于读取控制器唤醒源输入状况，各参数说明如下：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 唤醒源选择：模块当前读取的唤醒源信号。

该模块有一个输出端口，输出数据类型为 uint8 的信号，当输出为 0 时表示唤醒源未激活；当输出为 1 时表示唤醒源激活。

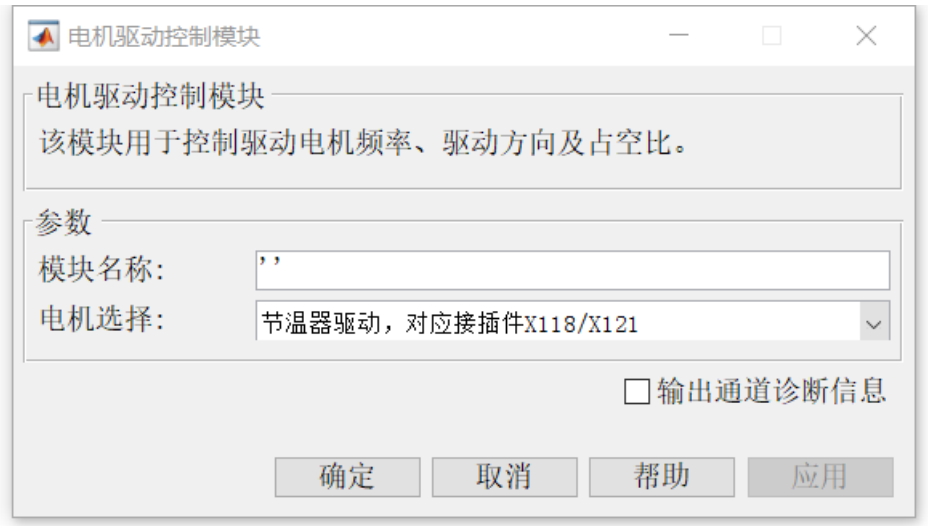
该模块只能读取一个唤醒源信号，对于有多个唤醒源的控制器，单个唤醒源

状态不代表控制器唤醒状态，如果需要确认控制器当前唤醒状态，需要添加多个模块选择不同通道后进行逻辑处理。以整车控制器为例，供包含 CAN 唤醒和三个接插件输入唤醒（其中一个为钥匙开关），四个唤醒源中只要有一个为激活状态控制器即开始工作。如需获取控制器整体唤醒状态，需要将四个唤醒源信号通过逻辑或进行处理。

4.6 复杂驱动模块

复杂驱动模块负责控制某些控制器上的特殊部件的驱动。与前几类模块不同的是，复杂驱动模块仅支持部分平台，使用之前需确定当前平台是否支持复杂驱动模块。

4.6.1 直流电机驱动模块

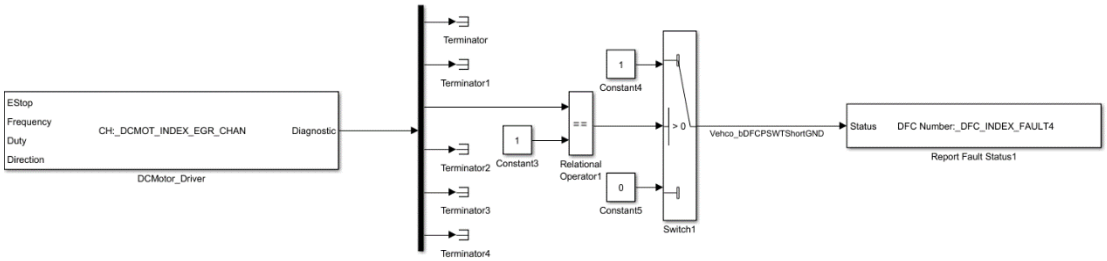


直流电机驱动模块用于控制电机驱动芯片，可通过输入信号来控制电机驱动信号频率、占空比、方向，以及强制停止电机。参数说明如下：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 电机选择：当前控制电机，与实际驱动芯片对应；
- 输出通道诊断信息：勾选该选项后，模块会增加一个输出端口，用于获取驱动通道诊断信息。诊断信息为数据类型为 uint8 的六维数组，所代表内容如下表所示：

信号序号	诊断内容	值含义
0	开路 *DSM_DIAG_OPENLOAD*	0x00:无故障 0x01:有故障 0xFF:故障状态未知或无效
1	短路 *DSM_DIAG_SHORTCIRCUIT*	
2	短路到地 *DSM_DIAG_SHORTTOGND*	
3	短路到电源 *DSM_DIAG_SHORTTOPWR*	
4	电源故障 *DSM_DIAG_PWRFAIL*	
5	温度超限 *DSM_DIAG_OVERTEMP*	

表中内容详见 HardwareLib.h 中枚举类型 *DSM_idxDiagTypeE_TYPE* 与 *DSM_stDiagE_TYPE*，电机驱动输出通道诊断功能函数为 *DCMotDrv_GetDiagStatus*。



可参考上述结构对电机驱动输出进行诊断。

模块共有四个输入：

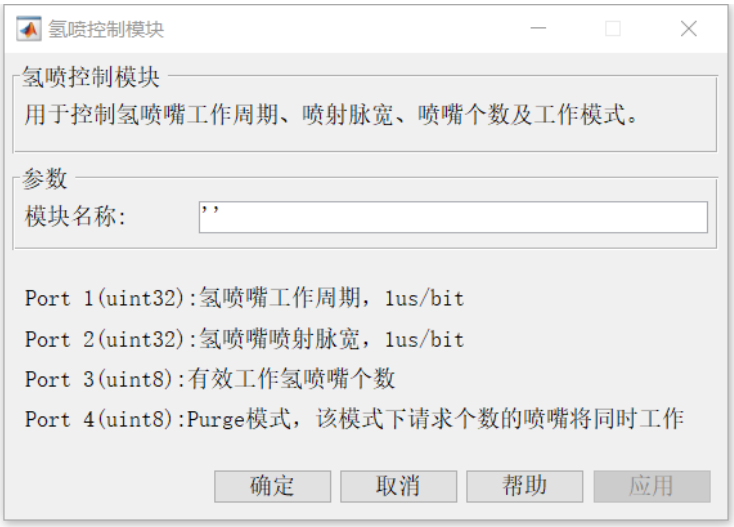
- EStop：紧急停机，数据类型 uint8。当该端口置 1 时，对输出通道进行紧急停机操作，端口为其他值时电机正常工作；
- Frequency：电机驱动信号频率，数据类型 uint32。频率精度为 0.1Hz/bit；
- Duty：电机驱动信号占空比，数据类型 uint16。占空比精度为 0.01%/bit；
- Direction：电机转动方向，数据类型为 uint8。正向为 0，反向为 1。

模块支持平台：F31_FCU（121pin）

EC3664CEJ001_FCVSH

EC3616CEF001_GAC

4.6.2 氢喷控制模块



氢喷控制模块用于控制氢喷嘴喷射，可通过模块输入控制氢喷嘴工作周期、喷射脉宽、喷嘴个数以及是否开启 Purge 模式。该模块仅有一个参数：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复。

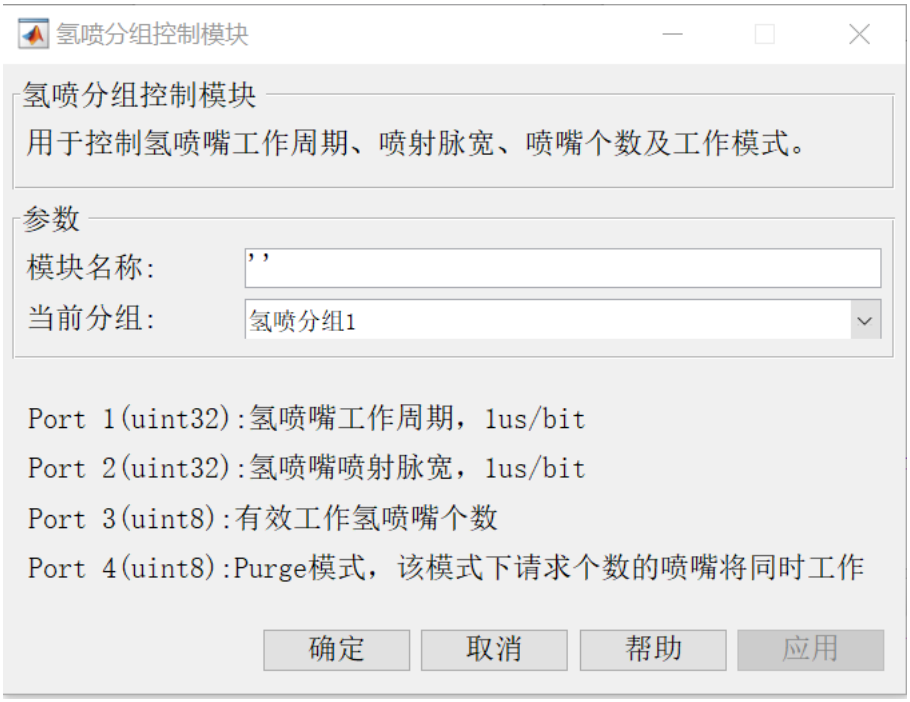
模块有四个输入端口：

- Period：工作周期，数据类型 uint32。精度为 1us/bit；
- Pulse：喷射脉宽，数据类型 uint32。精度为 1us/bit；
- InjectorNum：有效工作喷嘴个数，数据类型 uint8。最大个数为 6 个；
- Purge：Purge 模式，数据类型 uint8。该模式下请求个数的喷嘴将同时工作。输入为 1 时模式开启，输入为其他时模式关闭。

模块支持平台：F31_FCU（121pin）

EC3616CEF001_GAC

4.6.3 氢喷分组控制模块



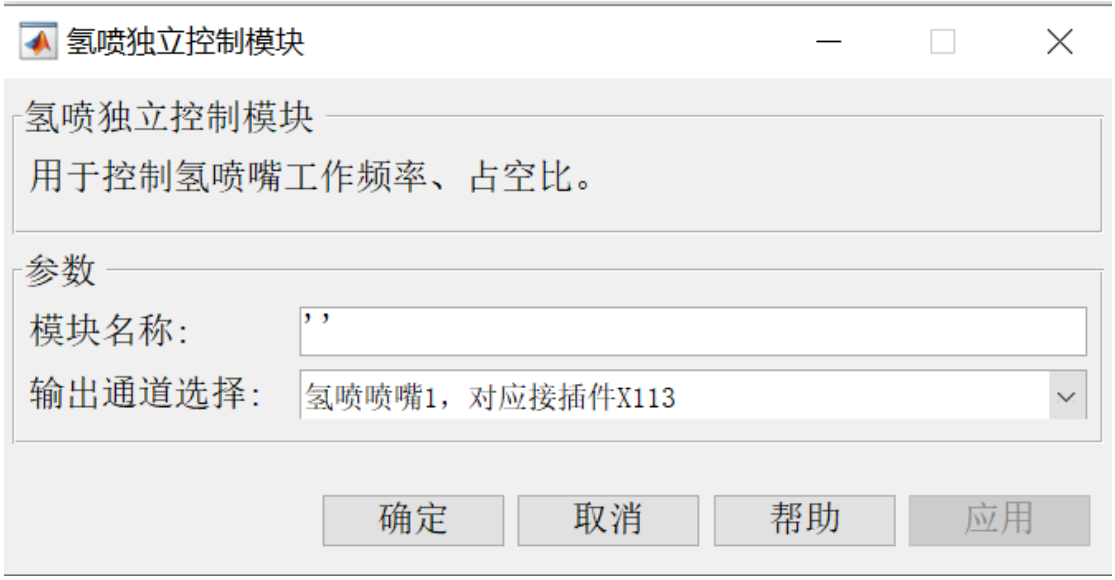
氢喷控制模块用于分组控制氢喷嘴喷射，可通过模块输入控制氢喷嘴工作周期、喷射脉宽、喷嘴个数以及是否开启 Purge 模式。该模块有两个参数：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复。
- 当前分组：选择目前氢喷模块控制的分组。

模块有四个输入端口：

- Period：工作周期，数据类型 uint32。精度为 1us/bit；
- Pulse：喷射脉宽，数据类型 uint32。精度为 1us/bit；
- InjectorNum：有效工作喷嘴个数，数据类型 uint8。最大个数为 6 个；
- Purge：Purge 模式，数据类型 uint8。该模式下请求个数的喷嘴将同时工作。输入为 1 时模式开启，输入为其他时模式关闭。

4.6.4 氢喷独立控制模块



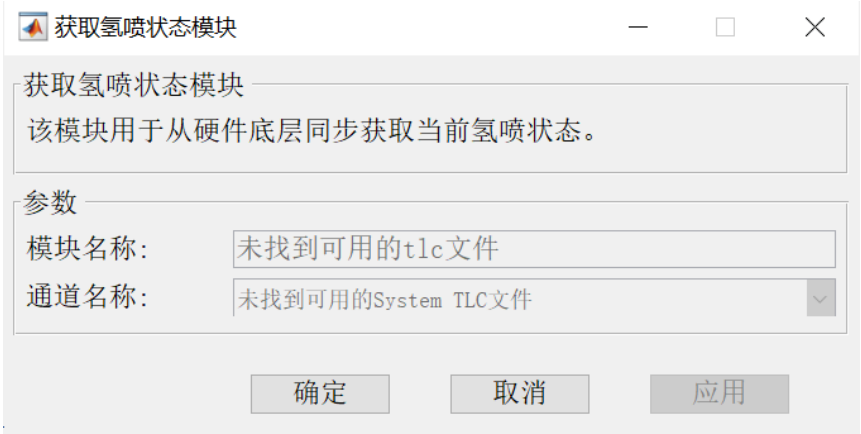
氢喷独立控制模块用于独立控制单个氢喷嘴喷射，可通过模块输入控制氢喷嘴频率和占空比。该模块有两个参数：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复。
- 输出通道选择：选择当前模块控制的氢喷通道。

模块有两个输入端口：

- Frequency：工作频率，数据类型 uint32。精度为 0.1Hz/bit；
- Duty：喷射占空比，数据类型 uint16。精度为 0.01%/bit。

4.6.5 氢喷获取喷嘴状态模块



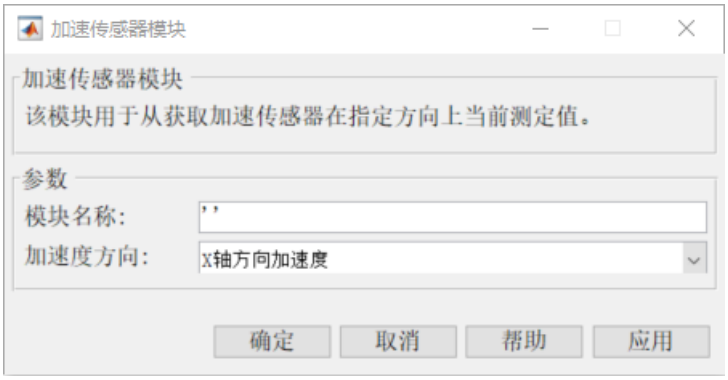
氢喷获取喷嘴状态模块用于实时获取当前喷嘴开启/关闭状态。该模块有两个参数：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复。
- 输出通道选择：选择当前模块获取的氢喷通道。

模块有一个输出端口：

- State：工作状态，数据类型 uint8，0 为关闭状态，1 为开启状态，0xFF 为通道号错误等原因导致的无效查询状态。

4.6.6 加速传感器模块



该模块用于从加速传感器获取当前选定方向的加速度。模块参数包括：

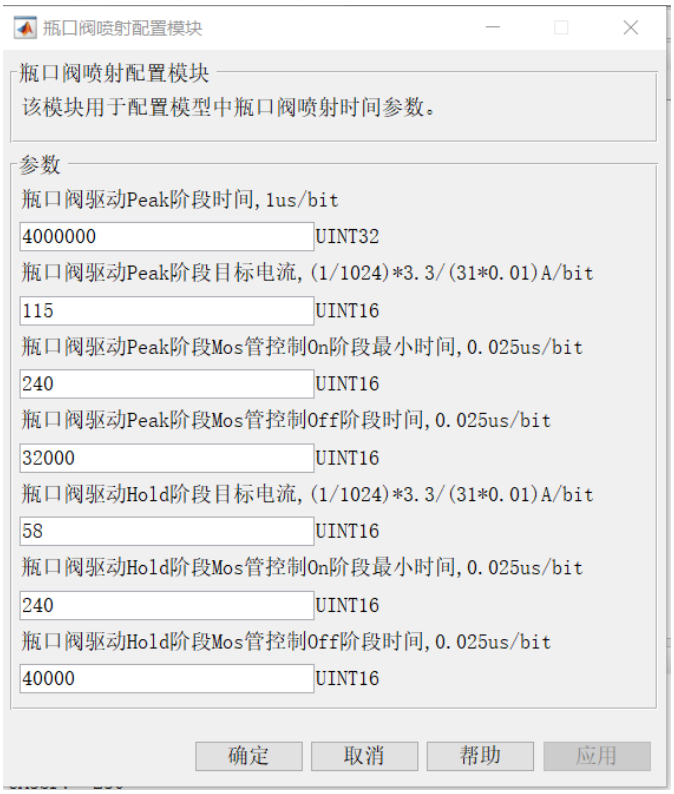
- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复。
- 加速度方向：当前采集的加速度值所对应方向。

模块有一个输出端口，用于传递采集到的信号值。信号值数据类型为 sint16，符号表示加速度方向，精度为 1/4096 g/bit。

模块支持平台：EC3600CBV008_HT(TC234_154pin)

4.6.7 瓶口阀配置模块/瓶口阀喷射控制模块

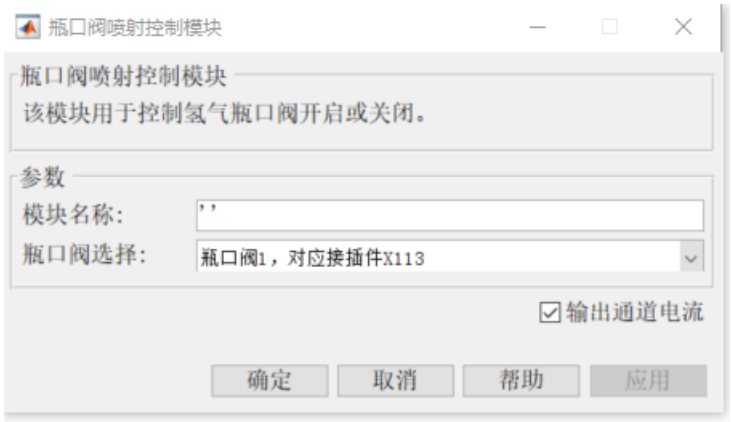
该组模块用于驱动氢瓶口阀及配置电流和时间参数。如需使用氢瓶阀驱动功能，首先需要向模型中添加氢瓶阀配置模块。



该模块参数包括：

- 瓶口阀驱动 Peak 阶段时间,1us/bit；
- 瓶口阀驱动 Peak 阶段目标电流,(1/1024)*3.3/(31*0.01)A/bit；
- 瓶口阀驱动 Peak 阶段 Mos 管控制 On 阶段最小时间,0.025us/bit；
- 瓶口阀驱动 Peak 阶段 Mos 管控制 Off 阶段时间,0.025us/bit；
- 瓶口阀驱动 Hold 阶段目标电流,(1/1024)*3.3/(31*0.01)A/bit；
- 瓶口阀驱动 Hold 阶段 Mos 管控制 On 阶段最小时间,0.025us/bit；
- 瓶口阀驱动 Hold 阶段 Mos 管控制 Off 阶段时间,0.025us/bit。

配置时需注意数据不要超限。



驱动瓶口阀开关，需使用瓶口阀喷射控制模块，该模块参数如下：

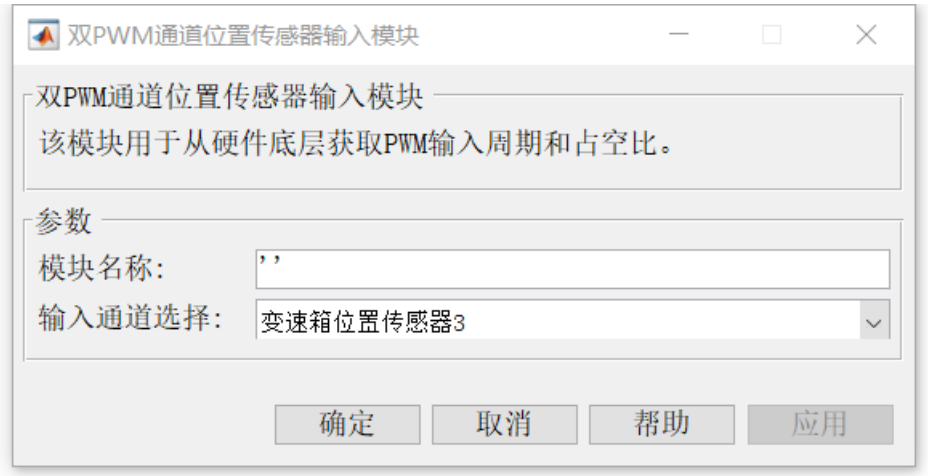
- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 瓶口阀选择：选择要驱动的瓶口阀通道，不同通道可单独控制；
- 输出通道电流：勾选该选项后，模块会产生一个输出端口用于采集驱动通道电流值。

模块分别各有一个输入和输出端口：

- State：驱动状态，数据类型 uint8。0 表示关闭，1 表示开启；
- Current：驱动电流，仅在勾选“输出通道电流”后显示，数据类型 uint16。精度为(1/1024)*3.3/(31*0.01)A/bit。

模块支持平台：EC3664CEJ001_FCVSH

4.6.8 TCU 位置传感器模块



该模块通过双通道 PWM 获取 TCU 当前位置信号。模块参数包括：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复。
- 输入通道选择：当前采集的传感器通道。

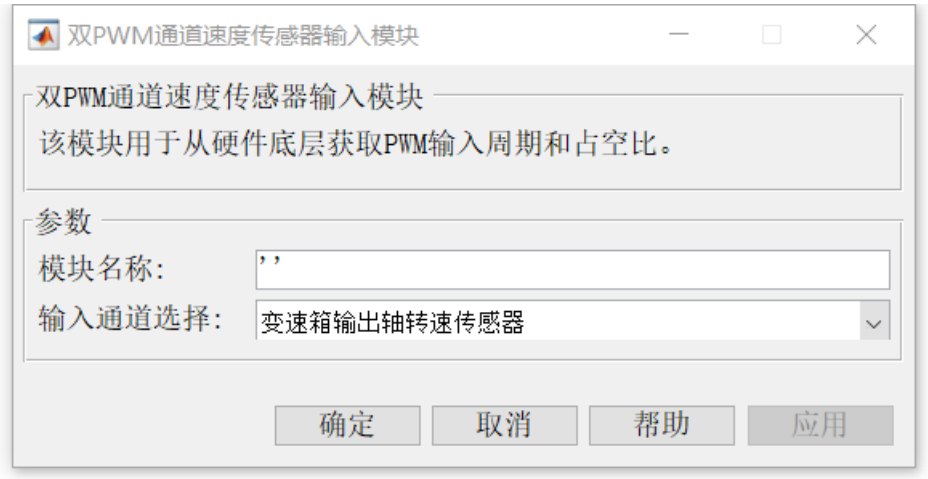
模块有两个输出端口，用于传递采集到的信号值：

- 周期 Period：PWM 信号周期，数据类型 uint32，单位:us，转换关系:1<->1us；
- 占空比 Duty：PWM 信号占空比，类型:uint16，单位:%，转换关

系:0~10000<->100%。

模块支持平台：EC6206MV00_YC(TC275_154pin)

4.6.9 TCU 速度传感器模块



该模块通过双通道 PWM 获取 TCU 当前速度信号。模块参数包括：

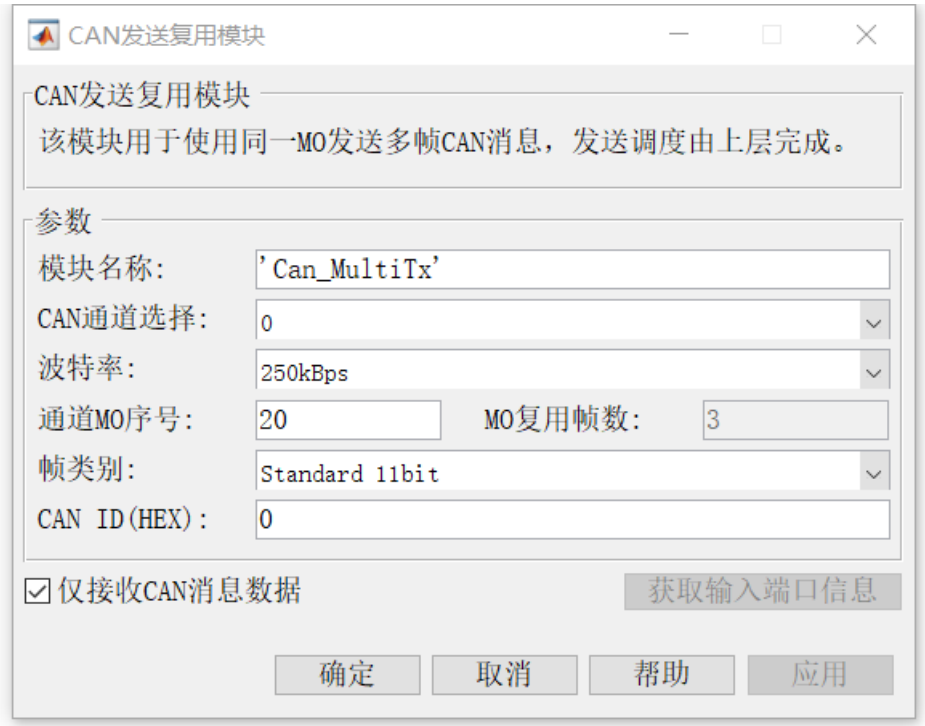
- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复。
- 输入通道选择：当前采集的传感器通道。

模块有三个输出端口，用于传递采集到的信号值：

- 周期Period：PWM信号周期，数据类型uint32，单位:us，转换关系:1<->1us；
- 占空比 Duty：PWM 信号占空比，类型:uint16，单位:%，转换关系:0~10000<->100%；
- 转速方向 Direction：数据类型 uint8，0 表示正转，1 表示反转。

模块支持平台：EC6206MV00_YC(TC275_154pin)

4.6.10CAN 发送复用模块



为解决 MO 模式下某些控制器 MO 数量不满足客户需求的问题，ECCoder 提供了 CAN 发送复用模块，使用户可以使用同一 MO 发送不同 ID 报文。该模块使用方法和 CAN 发送模块基本一致，包含参数如下：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- CAN 通道选择：硬件支持的 CAN 通道列表；
- 波特率：当前选定通道的波特率，共有 125/250/500/1000 kbps 四档可选。该值也可在 CAN 配置模块中统一修改，二者互相实时更新；
- 通道 MO 号：该接受模块所使用的 MO 序号，数值越低则代表其优先级越高。和 CAN 发送模块不同的是，发送复用模块中 MO 号必须手动指定，MO 号和 CAN 发送/接收模块 MO 号不可重复，但是发送复用模块直接的 MO 号可以重复。MO 复用帧数显示了当前 MO 所包含的复用帧数量，如何和发送/接收模块重复则显示 Used；
- 帧类别：接收消息的 ID 格式，有标准帧”Standard”和扩展帧”Extended”可选。
- CAN ID：接收消息的 CAN ID，用十六进制数表示；

- 仅接收 CAN 消息数据：非勾选情况下输入端口为 CAN Message 格式，输出包含 ID、数据长度、帧格式、消息数据等信息的结构体类型。接收的消息不需用 can pack 模块进行打包。勾选状态下只接收维度为 8 的 CAN 消息数据；

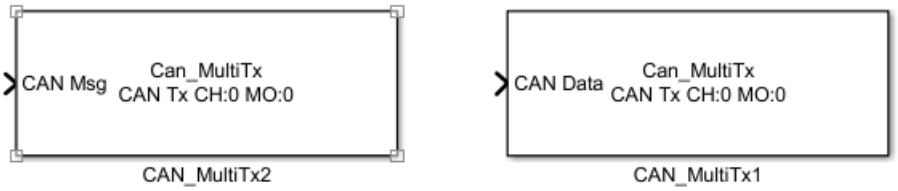


图.不同输入格式下的 CAN 发送复用模块

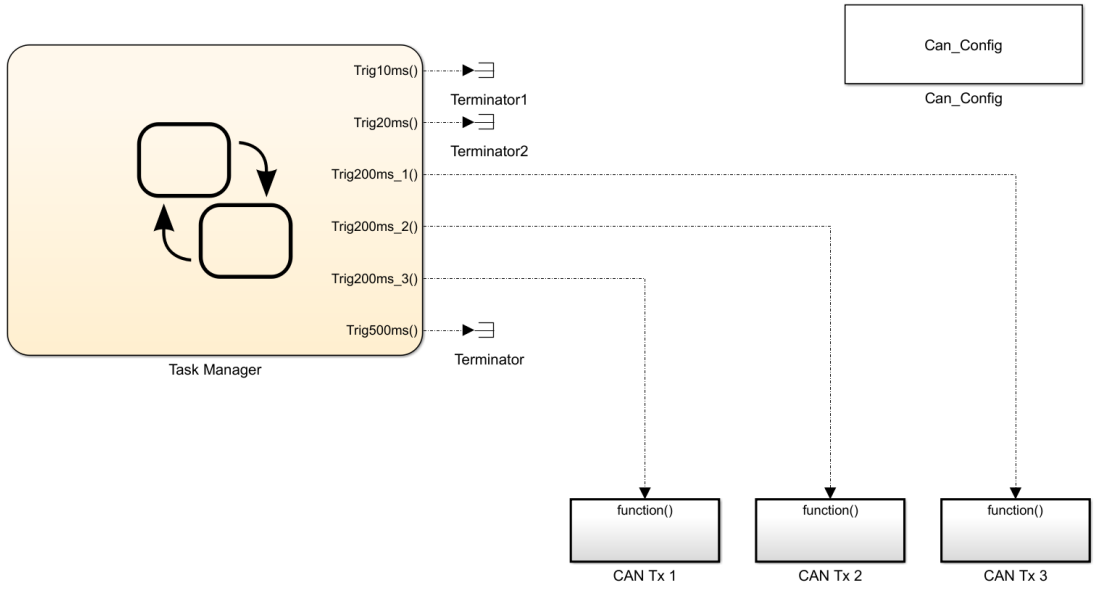
- 获取输入端口信息：在未勾选“仅接收 CAN 消息数据”情况下可用。如果模块输入端口已连接 can pack 模块，点击该按钮则自动将 pack 模块中的 CAN ID 和帧格式信息填入该模块，简化操作（仅在信号线直连情况下可用，如中间包含其他模块或被子系统分隔则无法获取）。

CAN 发送复用模块有一个输入端口：

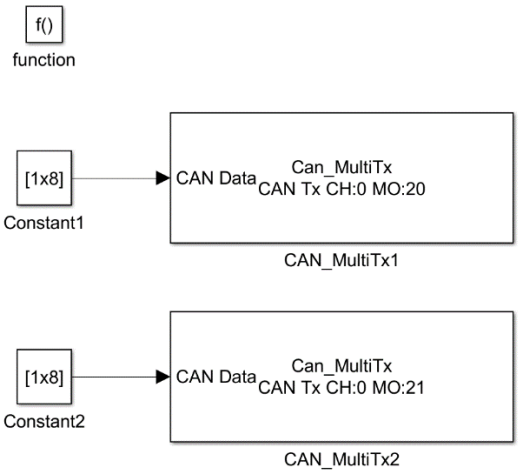
- 输入端口 1 – CAN Msg/CAN Data：输入 CAN Message 或者 CAN 消息数据用于发送。

CAN 发送复用模块没有输出端口，不支持发送回调函数。

使用 CAN 发送复用模块时，需要上层算法对发送进行调度，确保同一 MO 不可在一个周期内多次发送。如同一 MO 连续发送，由于上一帧报文没有发完，会导致后续报文丢帧的情况。用户可通过软件实现分频，将同一 MO 发送任务分散在不同周期内。



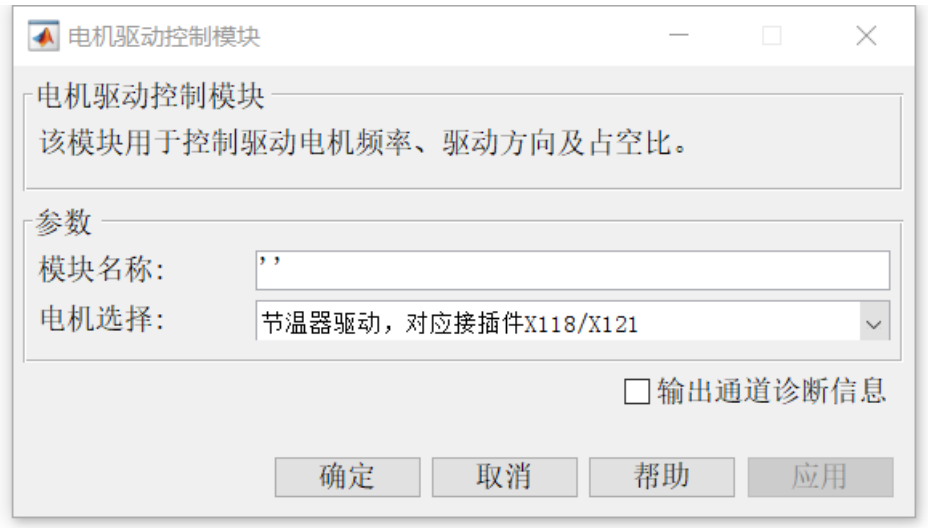
如图中示例，利用 Stateflow 生成了三个 200ms Trigger。三个 Trigger 周期均为 200ms，但不在同一 step 中触发，以此确保报文正常发送。



子系统中内容如图所示，不同 MO 的发送复用模块可在同一周期发送，相同 MO 的发送复用模块需分散在不同子系统中。该模块需手动分配 MO，使用起来时间成本较高，为解决 MO 数量不够问题的折衷方案，在 MO 资源足够的情况下请使用 CAN 通讯类别中 CAN 发送模块。上述样例可输入 *eccoder.demo('mulcan');* 打开。

模块支持平台：TC234 平台

4.6.11H 桥驱动模块



H 桥驱动模块用于控制 H 桥驱动芯片，可通过输入信号来控制驱动信号频率、占空比、方向。参数说明如下：

- 模块名称：为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- 通道选择：当前控制通道，与实际驱动芯片对应；
- 输出通道诊断信息：勾选该选项后，模块会增加一个输出端口，用于获取驱动通道诊断信息。诊断信息为数据类型为 uint8 的六维数组，所代表内容如下表所示：

信号序号	诊断内容	值含义
0	开路 *IOHWAB_CHAN_OPENLOAD*	0x00:无故障 0x01:有故障
1	短路 *IOHWAB_CHAN_SHORTCIRCUIT*	
2	短路到地 *IOHWAB_CHAN_SHORTTOGND*	
3	短路到电源 *IOHWAB_CHAN_SHORTTOPWR*	

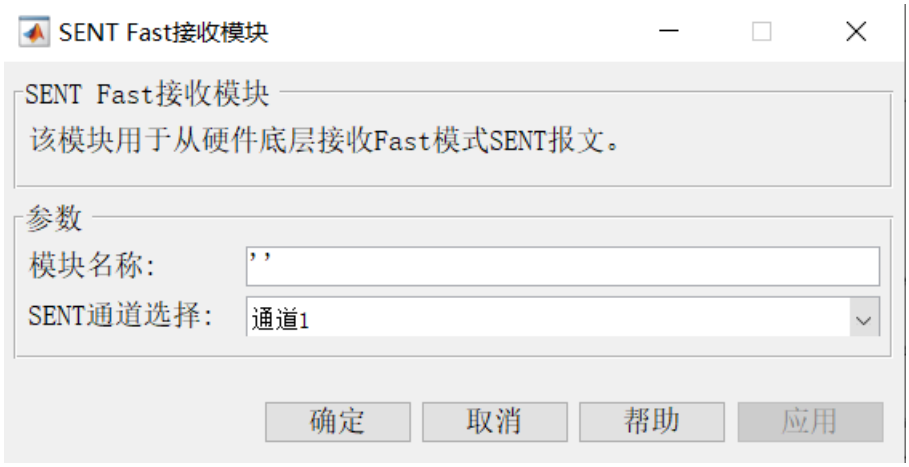
表中内容详见 Rte_Types.h 中枚举类型 *Rte_idxChanDiagTypeE_TYPE* 与 *Rte_stDiagE_TYPE*，H 桥驱动输出通道诊断功能函数为 *Rte_Call_IoHwAb_GetChanDiagInfo*。

模块共有三个输入：

- Direction：电机转动方向，数据类型为 uint8。正向为 0，反向为 1。
- Frequency：电机驱动信号频率，数据类型 uint16。频率精度为 1Hz/bit；
- Duty：电机驱动信号占空比，数据类型 uint16。占空比精度为 0.01%/bit；

模块支持平台：FCCU(TC275_154pin)

4.6.12SENT Fast 接收模块



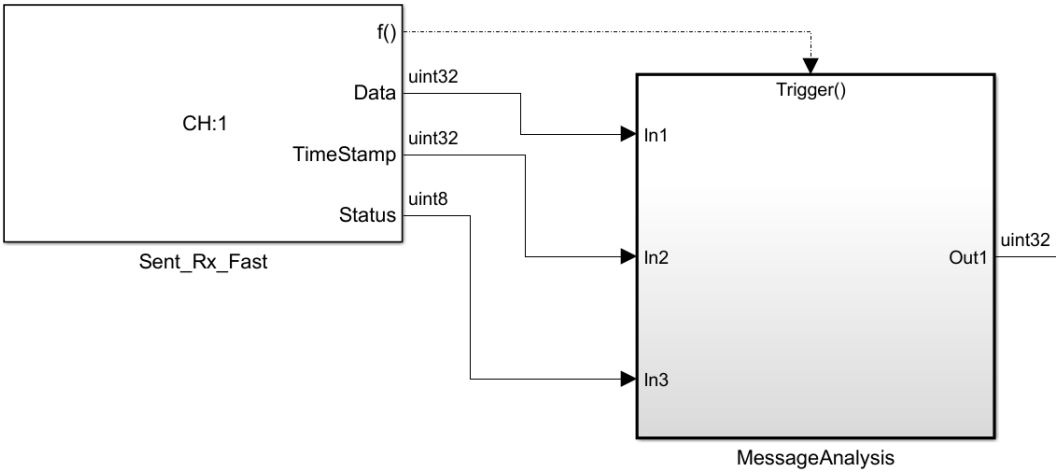
SENT Fast 接收模块用于接收 Fast 模式下的 SENT 消息。

- **模块名称：**为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- **SENT 通道选择：**硬件支持的 SENT 通道列表。

模块共有四个输出端口：

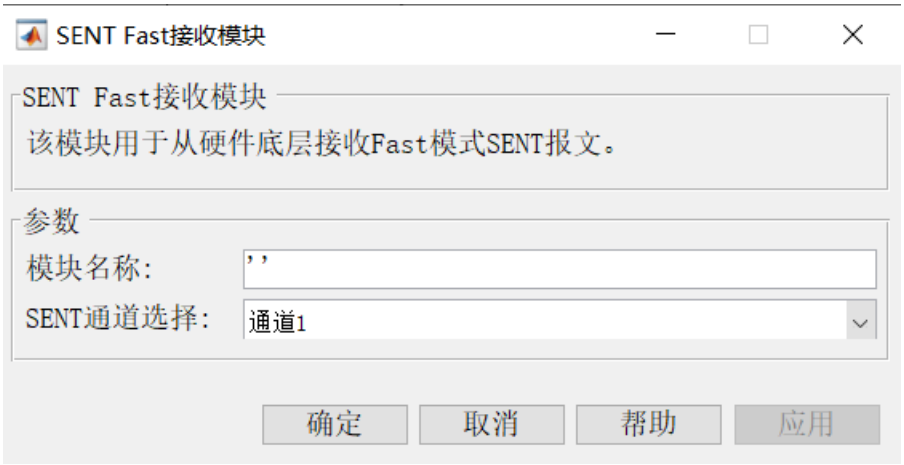
- f() – Trigger 信号，在成功接收到 SENT 消息后触发；
- Data – SENT 数据信息，数据类型 uint32；
- Time Stamp – 时间戳信息，数据类型 uint32；
- Status – 状态信息，数据类型 uint8。

SENT 接收模块必须触发子系统，f()端不可悬空或接 terminator。其他输出端口建议直接连接至被触发的子系统中。可参考下图中方式实现接收功能。



支持平台：玉柴 FCCU

4.6.13SENT Slow 接收模块



SENT Slow 接收模块用于接收 Slow 模式下的 SENT 消息。

- **模块名称：**为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- **SENT 通道选择：**硬件支持的 SENT 通道列表。

模型共有五个输出端口：

- f() – Trigger 信号，在成功接收到 SENT 消息后触发；
- Crc – Crc 校验，数据类型 uint8；
- MessageId – 信息 ID，数据类型 uint8；

- SerialData – 串行数据，数据类型 uint16；
- ConfigBit – 配置位，有 0/1 两种状态，数据类型 uint8。

SENT Slow 接收模块使用方式与 Fast 相同。

支持平台：玉柴 FCCU

4.6.14转速传感器模块

转速传感器模块用于获取 TCU 当前转速传感器返回值。

- **模块名称：**为模块指定名称，用于标识模块功能，在生成代码中以注释形式体现。该参数仅用于说明功能，不能作为索引，可以重复；
- **通道选择：**硬件支持的转速传感器通道列表。

模型共有三个输出端口：

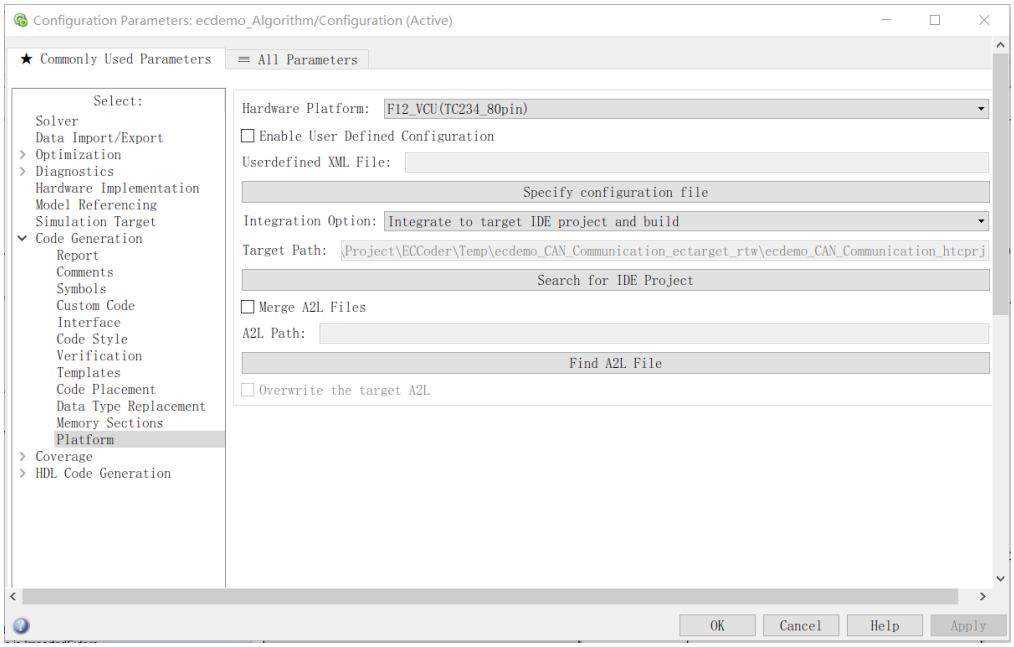
- f() – Trigger 信号，在转速传感器触发 TimeStamp 时调用回调函数；
- Period – 信号周期，数据类型 uint32，转换关系:1<->1us；
- Duty – 信号占空比，数据类型 uint16，转换关系:0~10000,<->100%。

支持平台：海易 TCU

5 ECCoder 用户自定义配置说明

ECCoder 支持自定义配置，配置内容包括故障诊断中 DFC/FID 的名称和其在代码生成中的宏定义、IO 硬件通道的自定义名称等内容。

5.1 打开配置文件



在模型 Configuration – Code Generation – Platform 界面中，用户可以通过勾选 Enable User Defined Configuration 选框来加载自定义配置文件；如果未勾选，则采用工具箱目录下对应平台的 UserDefine.xml 作为配置文件。

如果一个控制器中需要添加多个模型，要保证多个模型采用同一个自定义配置文件。自定义配置中的内容与底层库声明文件 HardwareLib.h 相关，如果配置文件不一致会造成编译错误。

在命令行中输入 eccoder.packagemanager 打开包管理器，点击 Package Path 按钮打开平台支持包所在路径，复制其中的 UserDefine.xml 文件即可。

5.2 编辑配置文件

配置文件使用 xml 格式进行编写，格式要求参照 w3c 对于 xml 文件的标准（https://www.w3.org/XML/）。编辑前请选择适合 xml 的文本编辑器（截图中所

用编辑器为 Sublime Text3）。

5.2.1 硬件通道配置

硬件通道配置中包含三个标签，分别为 adin（模拟量输入）、din（数字量输入）、pswtout（高低边驱动输出）。

```
<!--硬件通道配置：硬件通道不可增加/删除条目，只能修改左侧文字描述，右侧通道索引为标识作用，不可修改-->
<!--显示菜单（可自定义）                    通道号索引（不可修改）-->
<!--模拟量输入配置-->
<adin>
    BAK1信号，对应接插件X64;                BAK1/X64,
    BAK2信号，对应接插件X26;                BAK2/X26,
    BAK3信号，对应接插件X27;                BAK3/X27,
    BAK4信号，对应接插件X35;                BAK4/X35,
    BAK5信号，对应接插件X30;                BAK5/X30,
    BAK6信号，对应接插件X31;                BAK6/X31,
    DI9信号，对应接插件X15;                 DI9/X15,
    DI8信号，对应接插件X11;                 DI8/X11,
    DI10信号，对应接插件X4;                 DI10/X4,
    制动1，对应接插件X58;                   BRAKE1/X58,
    加速踏板2，对应接插件X63;               ACCPED2/X63,
    DI7信号，对应接插件X5;                  DI7/X5,
    DI1信号，对应接插件X10;                 DI1/X10,
    DI6信号，对应接插件X9;                  DI6/X9,
    制动2，对应接插件X61;                   BRAKE2/X61,
    加速踏板1，对应接插件X62;               ACCPED1/X62,
    DI5信号，对应接插件X14;                 DI5/X14,
    DI4信号，对应接插件X13;                 DI4/X13,
    DI3信号，对应接插件X18;                 DI3/X18,
    DI2信号，对应接插件X17;                 DI2/X17,
    蓄电池电压，内部连接;                   UBATTERY/INTERNAL
</adin>
```

标签中，每一行表示一个条目，条目之间用英文逗号[,]分割，最后一行末尾没有逗号。每个条目中分两部分，前一部分为硬件通道自定义名称，显示在模块用户定义名称一栏，这部分内容可由用户修改；后一部分为对应底层硬件通道号，起标识作用，不建议修改。两部分之间用英文分号[;]分隔。

配置文件中空格会正常显示在模块中，制表符和换行符则不会被显示。如需对齐文本，请使用制表符[Tab]。

5.2.2 DSM（诊断系统）配置

DSM 配置共包含两个标签：dfc（故障检查项）配置，用于 Get DFC Error/Get DFC Status/Report Fault Status 模块；fid（功能禁用项）配置，用于 Get FID Status 模块。

```
<!--DSM配置：DSM配置项可自定义条目数，左侧为显示内容，右侧为代码生成宏定义，宏定义需符合C语言规范-->
<!--显示菜单（可自定义）                    宏定义（可自定义）-->
<!--故障检查项配置-->
<dfc>
    Accelerator Pedal APP1SRC too High;     '_DFC_INDEX_ACCPED_APP1SRC_HIGH',
    Accelerator Pedal APP2SRC too High;     '_DFC_INDEX_ACCPED_APP2SRC_HIGH',
    Accelerator Pedal APP1SRC too Low;      '_DFC_INDEX_ACCPED_APP1SRC_LOW',
    Accelerator Pedal APP2SRC too Low;      '_DFC_INDEX_ACCPED_APP2SRC_LOW',
    Accelerator Pedal APPNPL;               '_DFC_INDEX_ACCPED_APPNPL',
    Brake Pedal SRC Too High;               '_DFC_INDEX_BRKPED_SRC_HIGH',
    Brake Pedal SRC Too Low;                '_DFC_INDEX_BRKPED_SRC_LOW',
    Gear Logic Fault;                       '_DFC_INDEX_GEAR_LOGIC_ERR',
    Brake Switch NPL Fault;                 '_DFC_INDEX_BRKSWT_NPL_ERR',
    Air Pump Overload;                      '_DFC_INDEX_AIRPUMP_OVERLOAD',
    Air Pump NPL;                           '_DFC_INDEX_AIRPUMP_NPL',
    Battary Voltage too Low Fault 1;        '_DFC_INDEX_BATT_LOW1_ERR',
    Battary Voltage too Low Fault 2;        '_DFC_INDEX_BATT_LOW2_ERR',
    Accelerator NPL Fault;                  '_DFC_INDEX_ACCBRK_NPL_ERR',
    Battary Voltage Fault;                  '_DFC_INDEX_BATTVOL_ERR',
    DMCM1toHCU CAN Timeout Fault;           '_DFC_INDEX_DMCM1TOHCU_TIMEOUT',
    DMCM2toHCU CAN Timeout Fault;           '_DFC_INDEX_DMCM2TOHCU_TIMEOUT',
    AVL2toHCU CAN Timeout Fault;            '_DFC_INDEX_AVL2TOHCU_TIMEOUT',
    BMStoHCU1 CAN Timeout Fault;            '_DFC_INDEX_BMSTOHCU1_TIMEOUT',
    BMStoHCU2 CAN Timeout Fault;            '_DFC_INDEX_BMSTOHCU2_TIMEOUT',
    BMStoHCU3 CAN Timeout Fault;            '_DFC_INDEX_BMSTOHCU3_TIMEOUT',
    ISGCM1toHCU CAN Timeout Fault;          '_DFC_INDEX_ISGCM1TOHCU_TIMEOUT',
    IP2toHCU CAN Timeout Fault;             '_DFC_INDEX_IP2TOHCU_TIMEOUT',
    EAC2HCU CAN Timeout Fault;              '_DFC_INDEX_EAC2HCU_TIMEOUT',
    EEC2toHCU CAN Timeout Fault;            '_DFC_INDEX_EEC2TOHCU_TIMEOUT',
    AVL1toHCU CAN Timeout Fault;            '_DFC_INDEX_AVL1TOHCU_TIMEOUT',
    ETC1 CAN Timeout Fault;                 '_DFC_INDEX_ETC1_TIMEOUT',
    ICto24V CAN Timeout Fault;              '_DFC_INDEX_ICTO24V_TIMEOUT',
    CCVSGauge CAN Timeout Fault;            '_DFC_INDEX_CCVSGAUGE_TIMEOUT',
    Motor Sensor Fault;                     '_DFC_INDEX_MOTOR_SENSOR_ERR'
</dfc>
```

标签中，每一行表示一个条目，条目之间用英文逗号[,]分割，最后一行末尾没有逗号。每个条目中分两部分，前一部分为 dfc/fid 名称，显示在诊断模块通道一栏，这部分内容可由用户修改；后一部分为在代码中 dfc/fid 对应的宏定义，同样由用户自定义，命名需符合 C 语言宏定义规范，两端需要有英文单引号[‘]。两部分之间用英文分号[;]分隔。

配置文件中空格会正常显示在模块中，制表符和换行符则不会被显示。如需对齐文本，请使用制表符[Tab]。

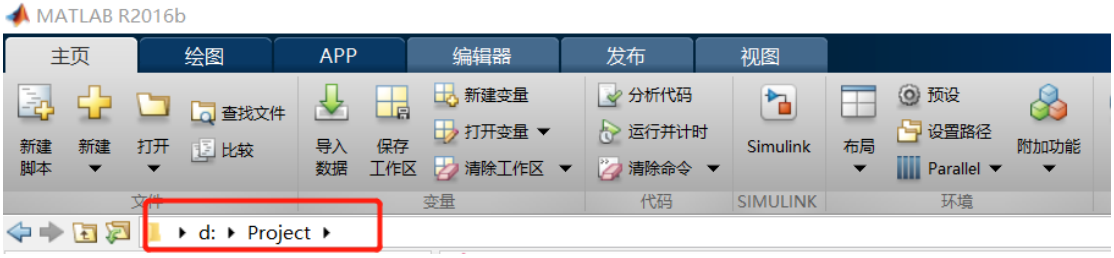
6 代码生成与集成

6.1 代码生成步骤

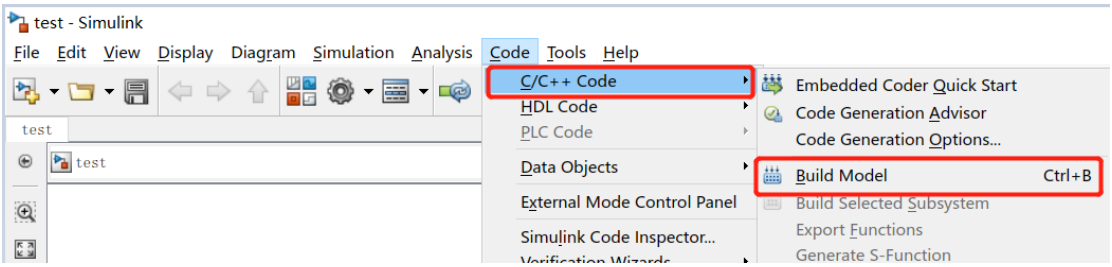
模型算法搭建完成后，可以进行代码生成。

在代码生成前，需要对模型进行仿真，只有当模型仿真无错误时才能进行代码生成。当模型中的错误排除后，首先需要选择代码生成路径。代码会生成在

matlab 当前路径中。



选择好路径之后，点击 Build Model 即会对模型编译并生成代码（或直接按 Ctrl+B）。



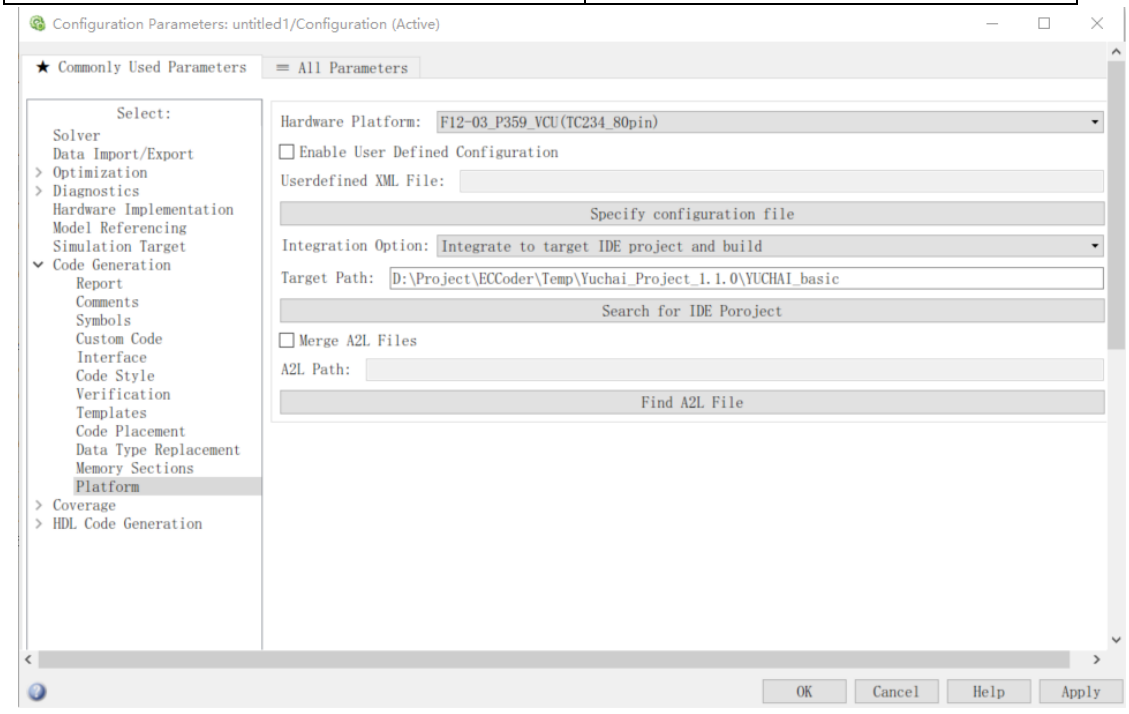
如生成成功，在选定路径下会生成代码及相关文件，目录下所有.c 和.h 文件均为有效文件。下一步需要将生成代码与 IDE（Hightec Eclipse）中底层代码进行集成，集成分为自动集成和手动集成两种方式。

6.2 自动代码集成及编译步骤

ECCoder 包含了一键集成及编译功能，该功能能在生成代码的同时完成集成和编译工作。ECCoder 在 Configuration－Code Generation－Platform 页面中 Integration Option 提供了六种不同的集成方式：

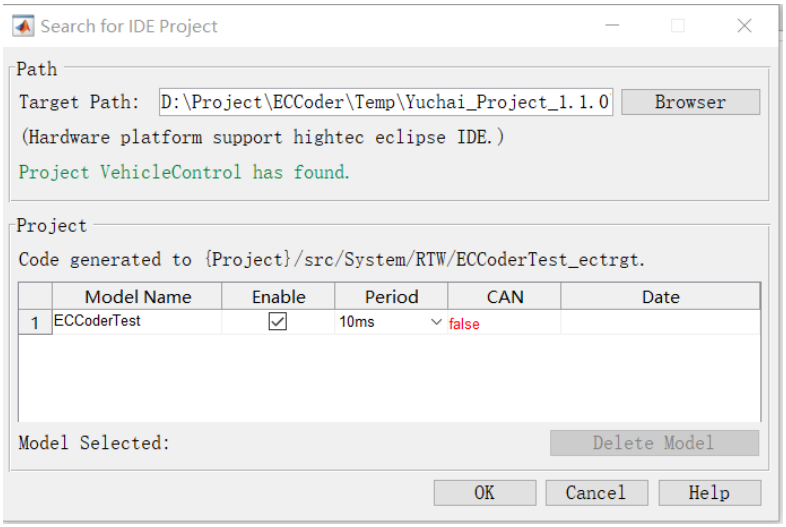
Generate Code Only	仅生成代码，不执行其他集成或编译操作。
Integrate to target IDE project	集成到目标工程，目标工程通过 Search for IDE Project 按钮选定，支持同一工程中包含多个 simulink 模型。适用于多模型存在于一个工程中，或需要对代码进行手工定制的场景。

Integrate to target IDE project and build	同上，增加一键编译功能。(仅支持 Hightec V2.1.2 以上版本编译器)
Integrate to default IDE project	集成到默认工程（默认调度周期 10ms），默认工程包含了必需的驱动和服务（如故障诊断等），集成后的工程中仅包含当前模型。IDE 工程保存在代码生成路径下。适用于单模型工程，或由模型新建 IDE 工程的场景。
Integrate to default IDE project and build	同上，增加一键编译功能。(仅支持 Hightec V2.1.2 以上版本编译器)
Generate code and hex file	集成到默认工程后自动编译，并将生成的 hex 和 elf 文件保存在代码生成路径下。适用于通过硬件快速验证模型效果的场景。



选择 Integrate to target IDE project 选项时，点击 Search for IDE Project 按钮弹

出菜单。



点击浏览按钮选择 Hightec 工程所在目录（必须为工程根目录，即.project 文件所在目录）。选定路径后，ECCoder 会自动扫描路径下包含的自动生成的代码模块及其调用信息。

- Model Name：模块名称，即 Simulink 模型名称（不可修改）；
- Enable：模块使能，模块是否在程序运行中被调用；
- Period：调用周期，共有 1ms/10ms/50ms 三种可选；
- CAN：模型中是否含有 CAN 模块，IDE 包含的所有模型中必须有且只有一个模型中包含 CAN 通讯模块。如果有超过一个或不存在包含 CAN 模块的模型，编译会发生错误（自动检测，不可修改）。
- Date：模型代码最后修改日期（自动检测，不可修改）。

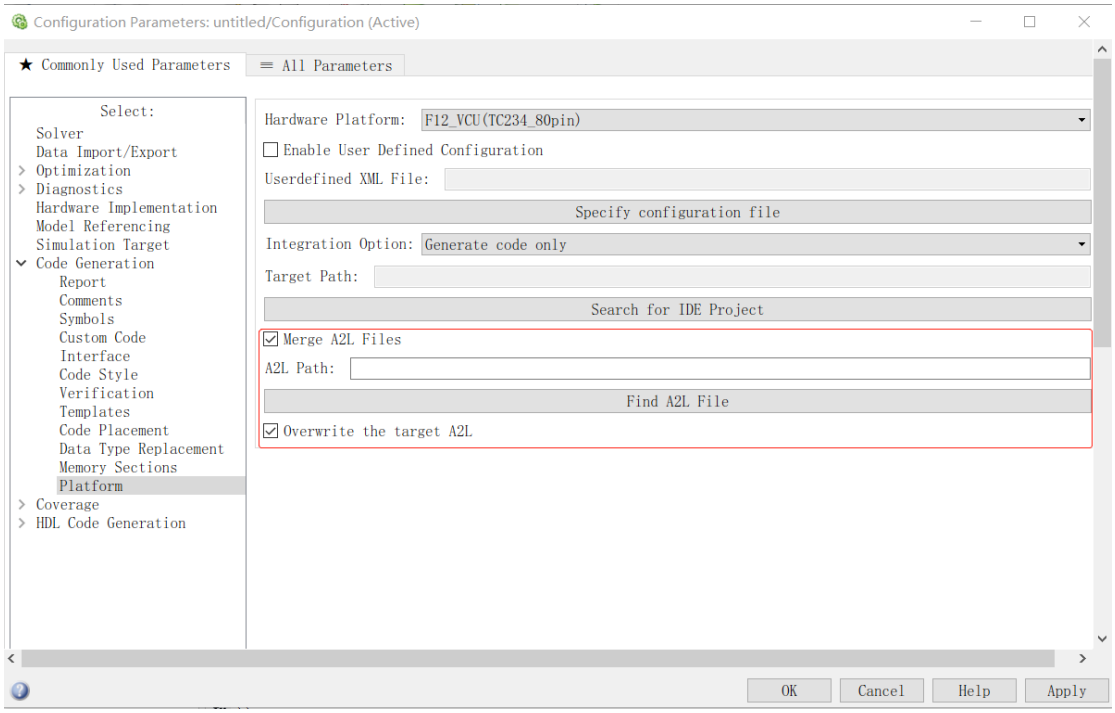
如需删除模型，在表格中选择模型后点击删除按钮，确认后即可删除，删除操作不可恢复。配置好集成信息后，编译模型，在生成代码后会自动将代码与工程进行集成。集成后的代码都保存在工程路径下 src/system/RTW 目录下，该目录中代码均为自动生成并由 ECCoder 统一管理，请勿手动修改。

6.3 A2L 自动合并及地址更新

模型生成的 A2L 文件只包含模型内部的变量及转换关系等信息，并不包含控制器底层变量；A2L 中也缺少对于监控软件与 ECU 通讯时的协议和配置信息(CCP/XCP、CAN 通讯等)；且 A2L 中变量地址均为空。在实际应用中，需要将带有

底层变量和配置信息的 A2L 和模型生成的 A2L 合并并更新地址。

6.3.1 集成阶段 A2L 自动合并



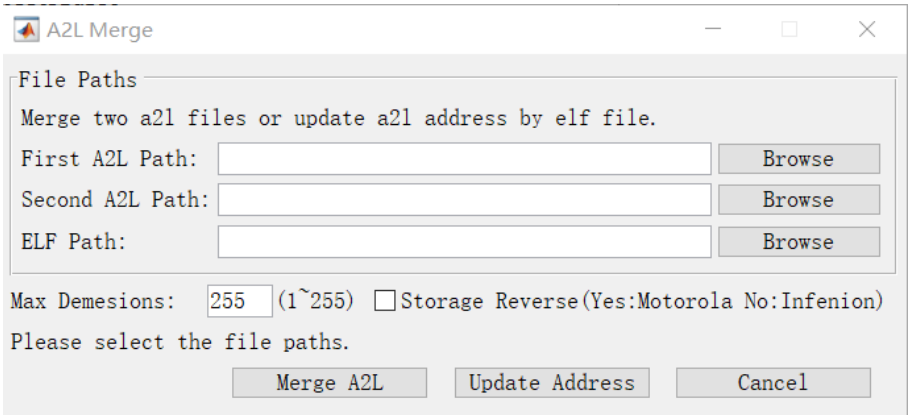
在配置界面中可以选择自动合并 A2L 选项。在 Code Generation – Platform 界面，勾选 Merge A2L Files 选项后，输入待合并的 A2L 文件路径或点击 Find A2L File 按钮浏览，在生成代码后即会自动将 A2L 合并。合并后的 A2L 文件会以 {ModelName}_merged.a2l 的名称保存在生成目录下。如果勾选了 Overwrite the target A2L 选项，合并后的文件会覆盖掉目标 A2L 文件（该操作不可逆，请注意备份）。如果在 Integration Option 中选择了 build 选项，且自动编译通过（生成了 hex 和 elf 文件），生成的 a2l 会根据 elf 文件自动更新地址。

6.3.2 A2L 自动合并工具

在命令行中输入

eccoder.a2lmerge

即可打开自动合并工具。



在该工具中分别输入两个 A2L 文件路径，点击 Merge A2L 按钮及可完成合并。如果同时指定了 ELF 文件地址，合并后会根据 ELF 对地址进行更新。在 First A2L Path 栏中输入 A2L 路径，同时指定 ELF 文件路径，点击 Update Address 按钮后会根据 ELF 对该 A2L 进行地址更新。

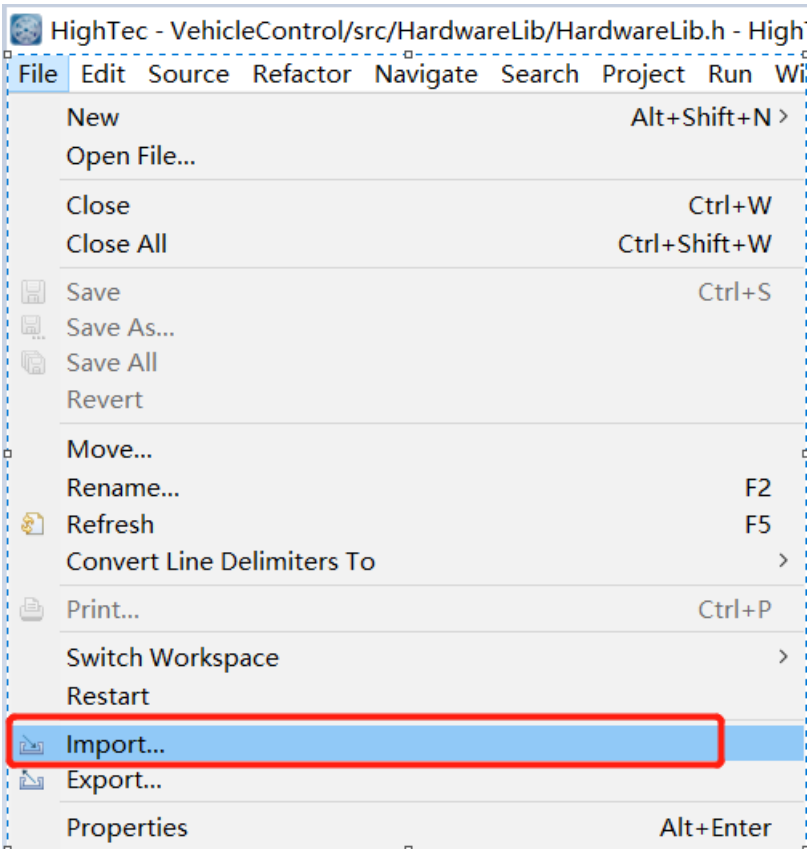
Max Demesions 指 A2L 中数组变量最大显示维度，超出该维度的数组元素在 A2L 中无法显示。Storage Reverse 选项指示了在数据转换时会不会将内存反转保存，例如采用摩托罗拉格式则需勾选该项，英飞凌则无需勾选。

6.4 手动代码集成步骤（Hightec IDE）

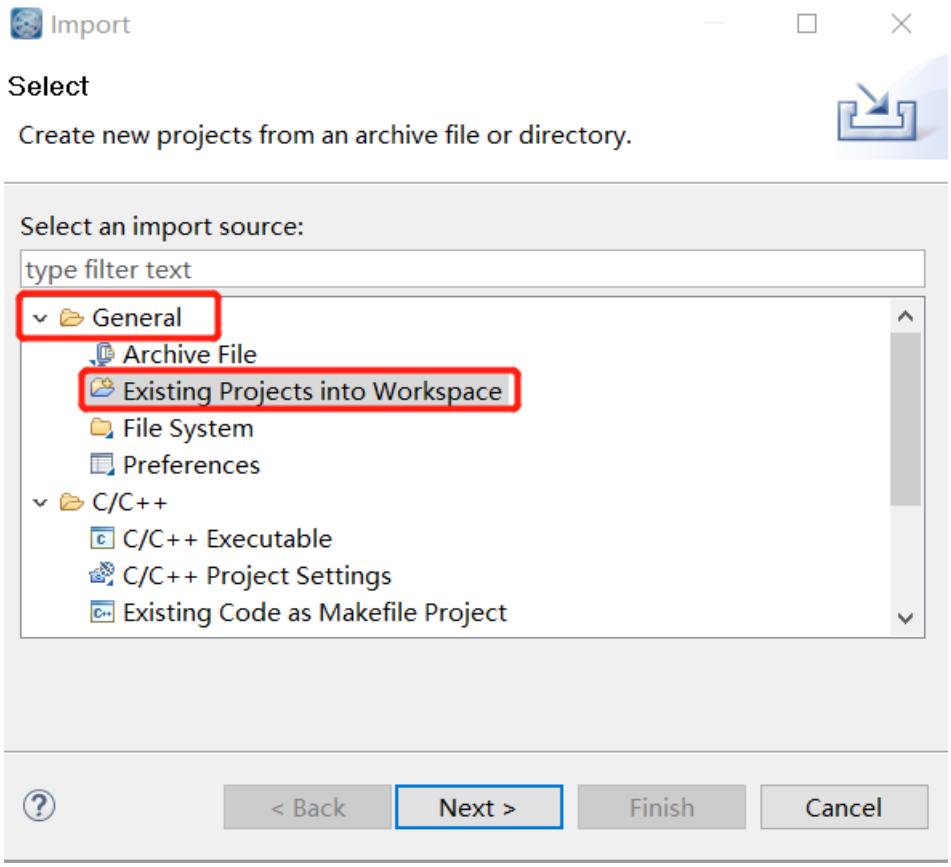
不推荐使用手动方式进行集成，可能会产生某些运行或兼容性错误，ECCoder 不会对这些问题进行检测和处理。如果模型需要进行特殊调用，可考虑使用手动集成。

6.4.1 向 workspace 中导入工程文件

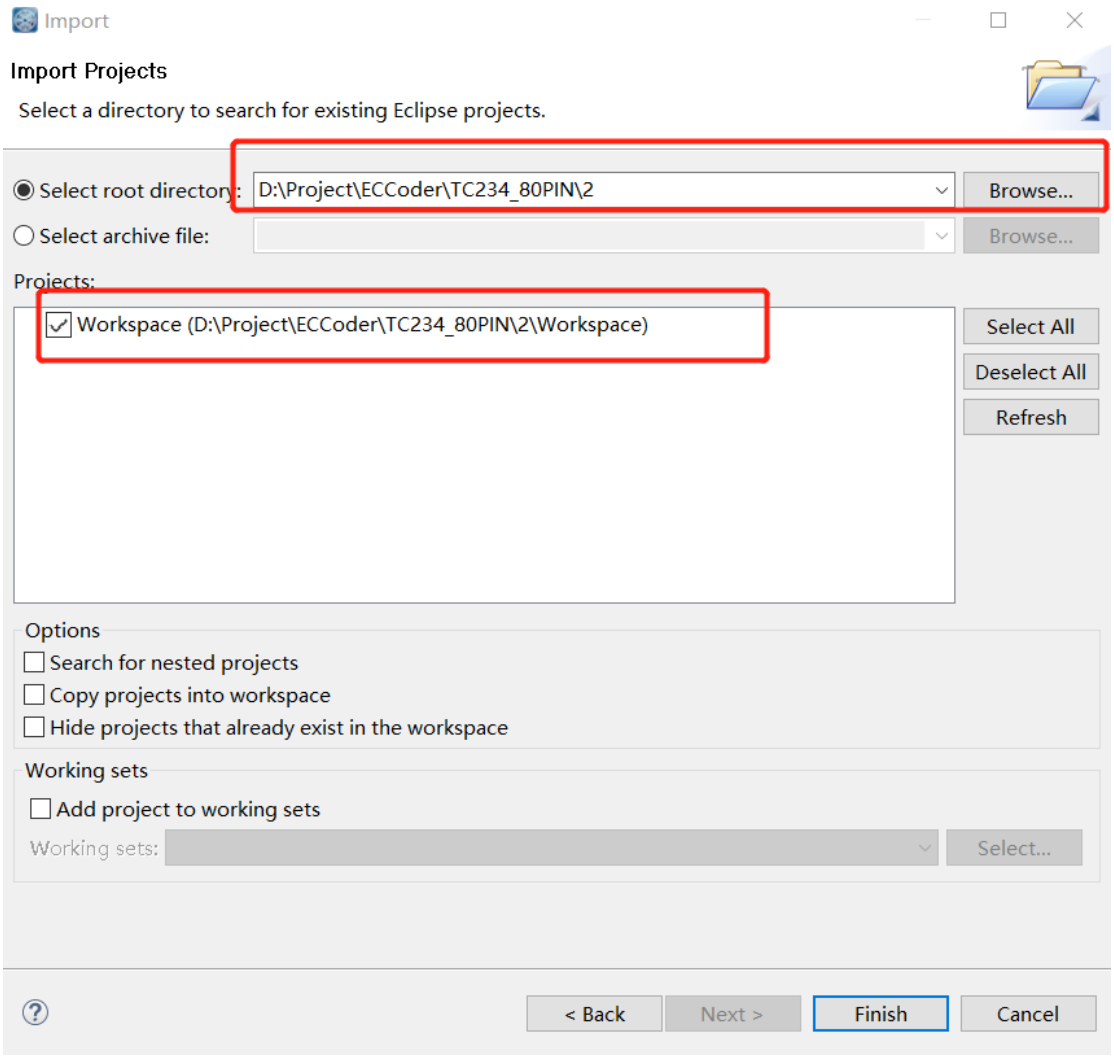
完成 Hightec 安装后，首次打开会提示选择 workspace 路径，指定路径之后即可将已有工程导入到 workspace 中。



点击 File-Import 按钮，在弹出菜单中选择 general-Existing Projects into Workspace，点击 next 将已有工程文件导入。

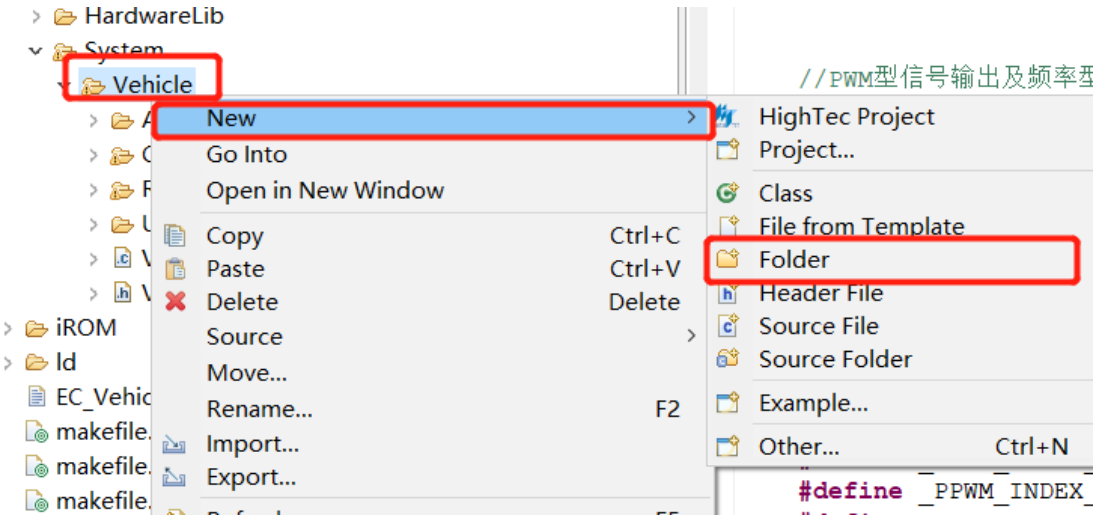


选择工程文件所在路径，下方会自动扫描路径内存在的工程文件，勾选需要导入的工程，点击 Finish 即完成导入。

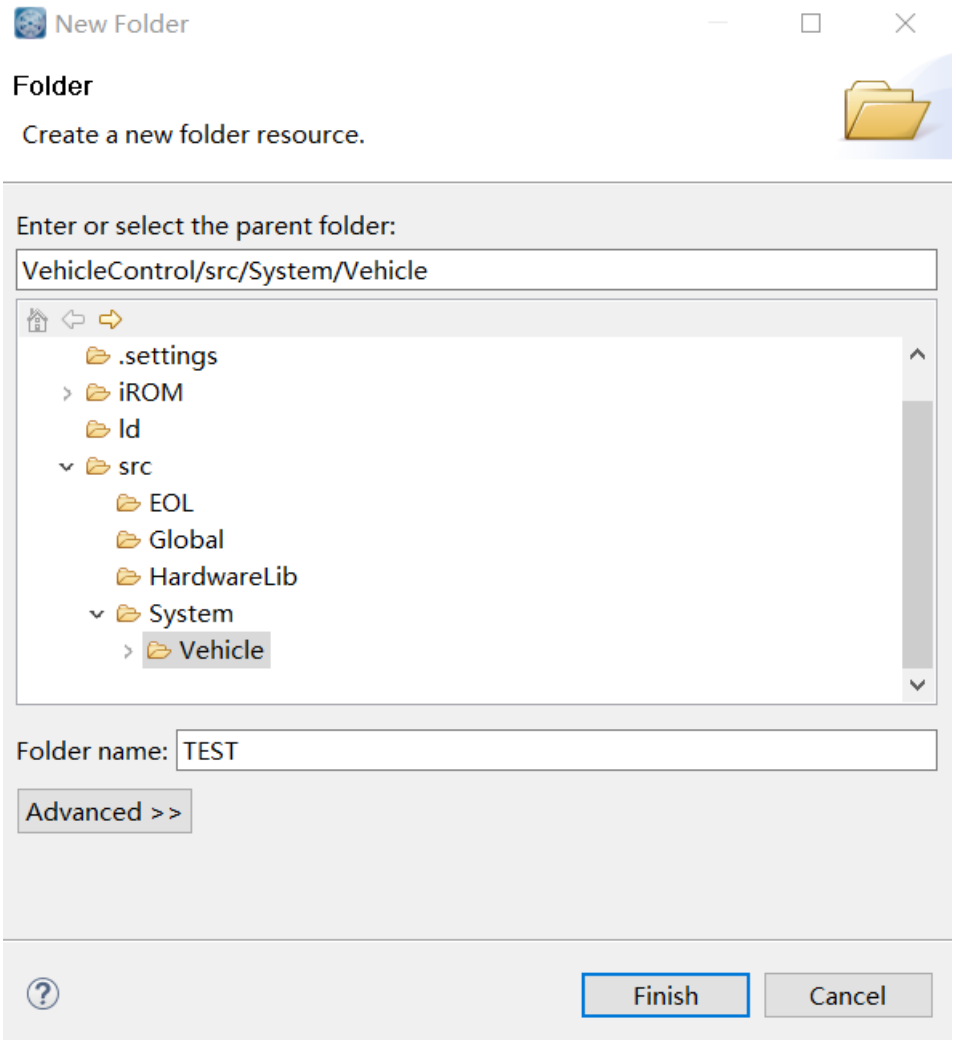


6.4.2 在工程文件中新建路径

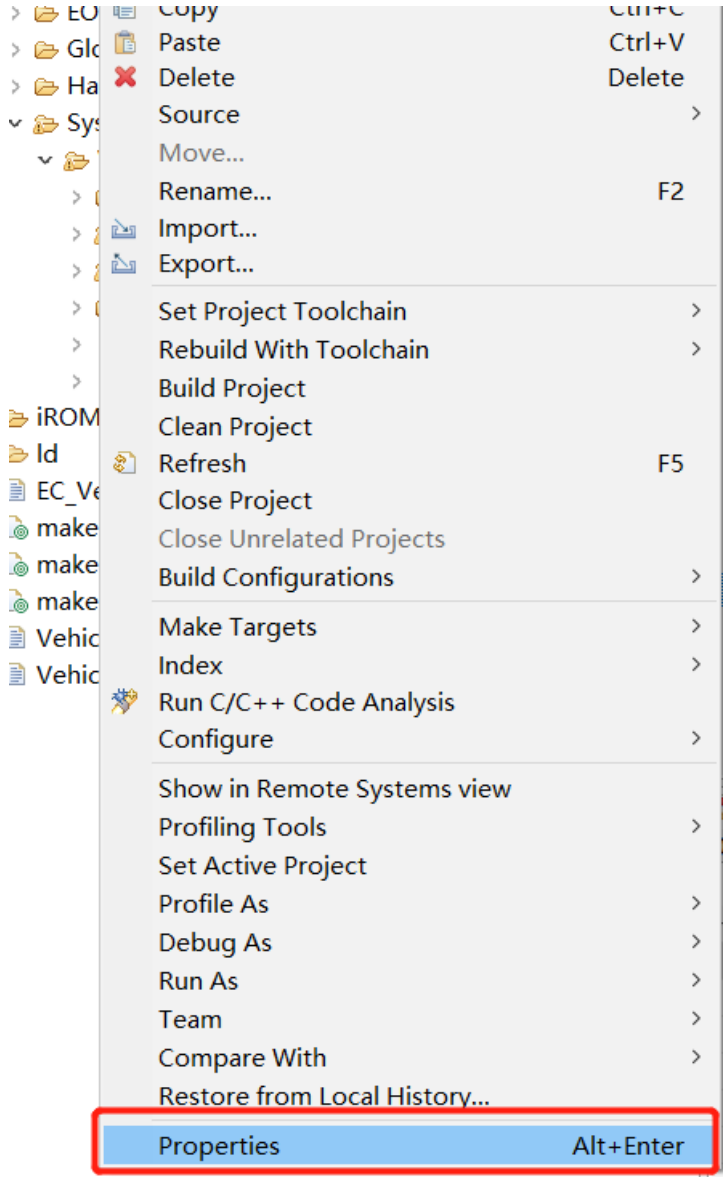
在想要新建路径的文件夹下点击右键 New-Folder。



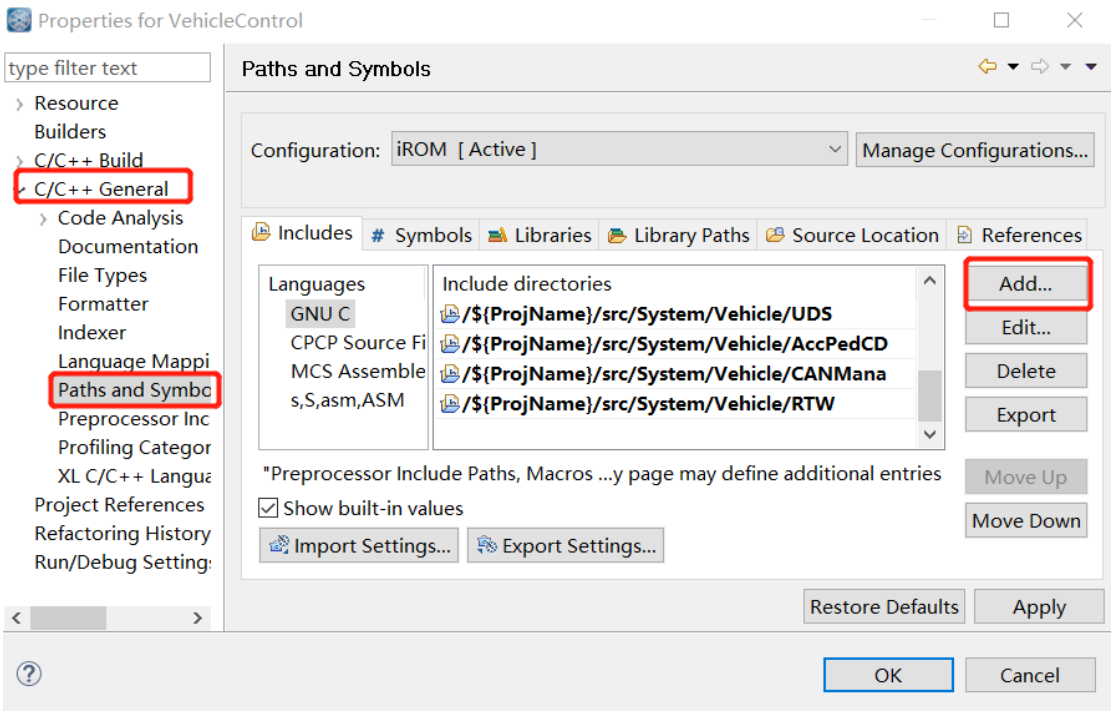
在对话框中输入中名称后即成功创建路径。



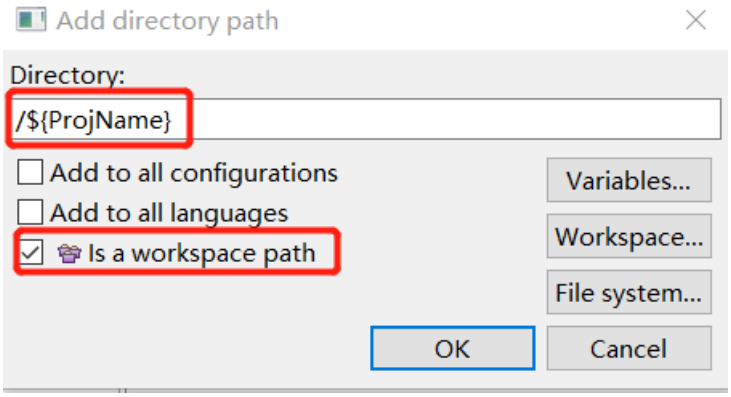
添加路径后，需要将其添加到 C builder 搜索路径后，其中的文件才会参与编译。右键单击工程文件，选择 properties。



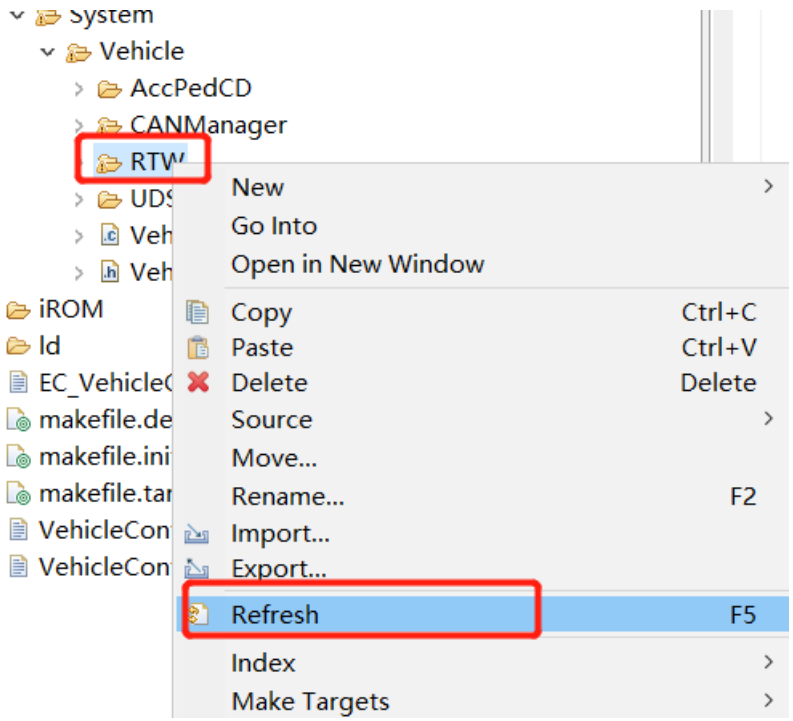
在弹出对话框中选择 Path and Symbols，点击 Add 按钮添加路径。



在目录中输入路径，以/${ProjName}开头表示工程文件目录，后面添加相对路径。同时勾选 Is a workspace path 选项。点击 OK 后即添加成功。



添加文件只需要在资源管理器中将文件添加至目录中即可。添加后，右键单击路径，点击 Refresh，即可更新目录下文件列表。



至此，打开工程文件及添加代码工作已经完成。

6.4.3 添加代码调用

在上述问题均解决之后，编译即可通过。但此时算法部分代码并没有被执行，因为没有入口函数调用它。算法入口函数为与算法同名的_step 函数，每次采样时 step 函数即执行一次。同时还有与算法同名的_initialization 函数，用以初始化模型中的变量。这两个函数均保存在和 slx 工程文件同名的.c 文件中。

如要添加对模型的调用，首先打开 VehCo.c 文件，该文件为整车控制器上层软件管理模块。在该模块开头添加算法模型头文件的调用：

#include "Can.h"

在该模块中找到整车初始化函数 VehCo_Init(),在该函数内调用算法初始化函数：

Can_initialize();

再在模块中找到任务管理模块，如模型采用 10ms 为采样周期，则找到 VehCo_Manage10ms()函数，在函数内部添加对 step 函数的引用：

Can_step();

step 函数也可在其他模块中进行调用，调用前需要在模块中添加对生成代码

头文件的 include 并执行初始化函数。至此，集成工作完成。算法代码可以在硬件设备上运行了。

6.5 生成代码结构及功能

6.5.1 英飞凌 TC234 平台

6.5.1.1 HardwareLib.h

HardwareLib.h 是底层库接口的声明文件，其中包含了 IO、诊断、CAN 通讯等接口函数的声明和相关数据类型及宏定义。文件中包含的函数如下表所示：

函数名	ECCoder 模块	功能
ATDDrv_GetChanResult	AD In	获取模拟量输入信号
DINDrv_GetChanState	DIn	获取数字量输入信号
PSwtDrv_Interface	PSWT Out	输出高低边驱动信号
MCANDrv_SendMsg	Can Transmit	发送 CAN 消息
DFC_ReportFaultLevel	Report Fault Status	报告故障状态
DFC_GetErrState	Get DFC Error	查询 DFC 是否处于故障中
DFC_GetDebState	Get DFC Status	查询 DFC 状态
DINH_GetFIDState	Get FID Status	查询功能禁用状态

表.HarewareLib.h 中函数列表

HardwareLib.h 中还包括了接口相关宏定义。IO 通道、DFC、FID 相关定义和用户自定义配置有关。当用户自定义配置 xml 文件发生变化时，HardwareLib.h 也会随之变化。在使用中，需要确保该文件和模型配置匹配。HardwareLib.h 存放在 Hightec 工程中 src/HardwareLib 路径下。

6.5.1.2 CANNet_Cfg.h

当模型中使用 CAN 相关功能时，即包含 CAN Config 模块时，会在代码中生成 CANNet_Cfg.h、CANNet_Cbk.c、CANNet_PBCfg.c 三个文件，用于保存 CAN 相关配置信息和回调函数。

上述三个文件在工程中只能唯一存在，所以针对同一控制器，涉及到 CAN 功

能的模块必须保存在同一模型中。

6.5.2 英飞凌 TC275 平台

6.5.2.1 Rte.h

Rte.h 是底层库接口的声明文件，其中包含了 IO、诊断、CAN 通讯等接口函数的声明和相关数据类型及宏定义。文件中包含的函数如下表所示：

函数名	ECCoder 模块	功能
Rte_Call_IoHwAb_Analog_GetChan	AD In	获取模拟量输入信号
Rte_Call_IoHwAb_Discrete_GetChan	DIn	获取数字量输入信号
Rte_Call_IoHwAb_Pwm_GetDutyCycle	PWN In	获取 PWM 输入周期和占空比
Rte_Call_IoHwAb_Discrete_SetLevel	PSWT Out	输出高低边驱动信号
Rte_Call_IoHwAb_Pwm_SetPeriod *Rte_Call_IoHwAb_Pwm_SetDuty*	PWM Out	输出 PWM 信号
Rte_Call_CanIf_Send	Can Transmit	发送 CAN 消息
Rte_Call_DFC_ReportFaultLevel	Report Fault Status	报告故障状态
Rte_Call_DFC_GetErrStatus	Get DFC Error	查询 DFC 是否处于故障中
Rte_Call_DFC_GetDebState	Get DFC Status	查询 DFC 状态
Rte_Call_DINH_GetFIDState	Get FID Status	查询功能禁用状态

表.Rte.h 中函数列表

Rte.h 中还包括了接口相关宏定义。IO 通道、DFC、FID 相关定义和用户自定义配置有关。当用户自定义配置 xml 文件发生变化时，Rte.h 也会随之变化。在使用中，需要确保该文件和模型配置匹配。Rte.h 存放在 Hightec 工程中 src/Run_Time_Environment 路径下。

6.5.2.2 Com.c/Com.h

当模型中使用 CAN 相关功能时，即包含 CAN Config 模块时，会在代码中生成 Can*_Cfg.h、Can*_PBCfg.c、Com.c、Com.h 文件（其中*表示 CAN 通道号），用于保存 CAN 相关配置信息和回调函数。

上述文件在工程中只能唯一存在，所以针对同一控制器，涉及到 CAN 功能的模块必须保存在同一模型中。CAN 通讯相关文件均存放在 Hightec 工程中 src/Com

路径下。

7 其他功能

7.1 模型样例

在命令行中输入以下代码来打开样例文件：

eccoder.demo;

执行该命令后会在 windows 资源浏览器中打开样例模型路径。该命令也可有输入参数，以直接打开样例模型，如

eccoder.demo('can');

表.eccoder.demo 函数接收参数列表

参数	模型
can	CAN 通讯模型，包含了 CAN 通道配置、CAN 收发、分频、超时处理等内容。（需与 algorithm 同时集成）
algorithm	算法模型，包含了简单的状态机、IO 通道、故障诊断、跨模型数据交互等内容。（需与 can 同时集成）
start	Get_Started 模型，ECCoder 入门指引 guideline。

7.2 ECCoder 支持命令一览

表.eccoder 支持命令列表

命令	功能
eccoder. canmgr	打开 CAN 通讯管理模块。
eccoder.datamgr	打开数据管理模块。
eccoder.activation	激活及 license 管理。
eccoder. backuplicense	将 license.dat 文件备份到当前路径下。
eccoder. systemconfig	系统管理（指定编译器路径）。
eccoder. packagemanager	包管理，添加/删除/更新支持包，查看包内容。
eccoder. a2lmerge	a2l 合并和更新地址工具。
eccoder.demo	打开 ECCoder 内置样例模型。