General Information: “How to use CCP integration of the IOLib_TTC580 and CANape”

Limitation: The CCP integration into the IOLib includes the code generation of the interface functions for the CCP stack (v2.1) from Vector. The A2L file generation by Mathworks Embedded Coder and the functionality of CANape itself are not included in the support given by “*CCP integration of the IOLib_TTC580*”

1) Parameters that need to be tuned have to be defined in the Model Workspace within the Model Explorer with the storage class "ExportedGlobal".

2) Signals that need to be monitored as measurements have to be defined in the Signal Properties with the signal object class "Simulink.Signal" and storage class "Exported Global".

3) Please make sure that directories for proper a2l-file generation are part of MATLAB’s current search path :
```
addpath([matlabroot,'\toolbox\rtw\targets\asap2\asap2\CANape_user'])
addpath([matlabroot,'\toolbox\rtw\targets\asap2\asap2\CANape_user\templates'])
addpath([matlabroot,'\toolbox\CANape'])
```
Note, if you do not find the above mentioned directories, please resume installation of CANape with activated option "MATLAB Simulink MC Add on 64Bit".
Note further, that the generation of a2l-files is a feature of Mathwork’s Embedded Coder. Questions related to this need to be addressed to Mathwork’s support services.

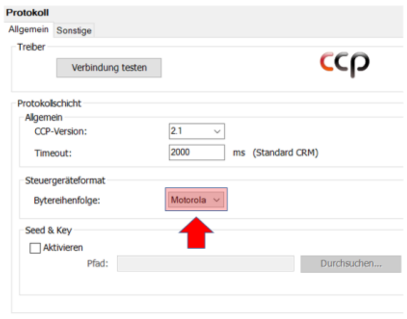
4) It is important that endianness is defined as “Motorola” within CANape, otherwise signals and parameter receive weird values. The proper setting may be achieved by definition in “Database settings” within CANape. Alternatively, you may edit the file *<MATLAB_ROOT>\toolbox\rtw\targets\asap2\asap2\CANape_user\asap2setup.tlc* and change the MSB-setting to be: `%assign ASAP2BYTE_ORDER = "MSB_FIRST"`. By doing so the byte order is set by default to "Motorola" in the generated *.a2l file by Mathwork’s Embedded Coder.

5) The file *C:\TTTech\Environment_TTC580\env\settings.mk* needs to be adapted to generate source-level debugging information during the build-procedure into *app.out*. To do so, identify in the file settings.mk the snippet:
```
...
 ifeq ($(OPT_STR), RELEASE)
     # optimize for release
     OPT_SYMDEBUG   := none
     OPT_LEVEL      := 2
     OPT_ADDITIONAL := -dRELEASE
 endif
...
```
and change this into:
```
...
ifeq ($(OPT_STR), RELEASE)
     # optimize for release
     OPT_SYMDEBUG   := dwarf
     OPT_LEVEL      := 2
     OPT_ADDITIONAL := -dRELEASE
 endif
...
```

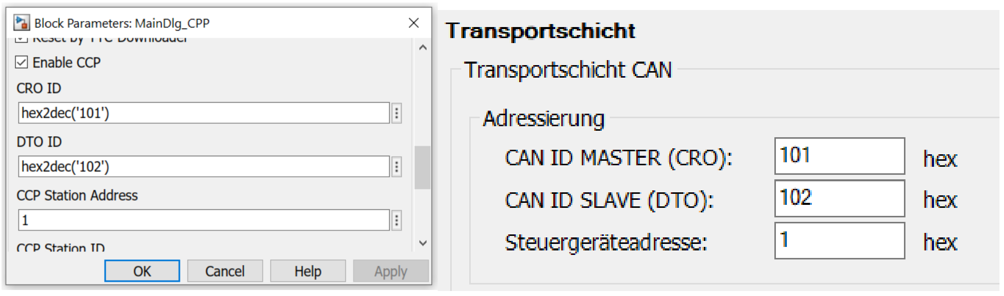
Note, that the same executable code is generated regardless of whether generation of debug information is turned on or off.

6) For the MAP file definition within Vector’s CANape use *app.out* which is generated by the linker during the make process. The format of the MAP file needs to be defined as `ELF/DWARF 32/64 Bit.` Assignment of the addresses to the identifier in the a2l-file needs to be performed by an appropriate tool (e.g. A2L-Editor within CANape)

7) In the Device Configuration within Vector's CANape set the ECU format to byte order "Motorola" and for the transport layer use the settings you defined in the mask of MainDlg within your Simulink model.


Protokoll
Allgemein | Sonstige
Treiber
Protokollschicht
Allgemein
CCP-Version: 2.1
Timeout: 2000 ms (Standard CRM)
Steuergeräteformat
Byterehenfolge: Motorola
Seed & Key
Aktivieren
Pfad:

Endianness needs to be set to “Motorola” (big-endian)

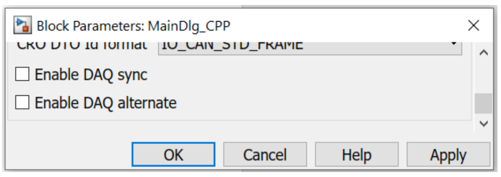

Block Parameters: MainDlg_CPP
Enable CCP
CRO ID: hex2dec('101')
DTO ID: hex2dec('102')
CCP Station Address: 1
CCP Station ID

Transportschicht
Transportschicht CAN
Adressierung
CAN ID MASTER (CRO): 101 hex
CAN ID SLAVE (DTO): 102 hex
Steuergeräteadresse: 1 hex

Definition of transport layer of the MainDlg-block corresponding to CANape settings

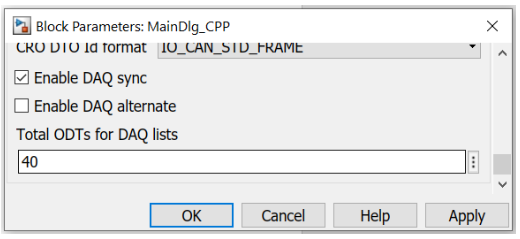
8) Activate CCP within the MainDlg block only for development purposes and deactivate it again for the final version of the SW, as the CCP-stack from Vector has the potential to access memory and may corrupt the execution, if not used with caution.

9) CCP Configuration without DAQ enabled:
In this type of configuration the CCP-master (CANape) sends out CRO commands to set or retrieve variables on the ECU and the ECU will answer with a corresponding DTO as an answer (e.g. SHORT_UPLOAD). This Communication is processed in the remainder of processing time between the end of the execution of the application and start of the next cycle.


Block Parameters: MainDlg_CPP
CRO DTO Id format: IO_CAN_STD_FRAME
Enable DAQ sync
Enable DAQ alternate

Setup without DAQ enabled

10) CCP Configuration with one DAQ enabled:
In this configuration one CRO-DTO handshake communication is possible in each cycle. Beside establishing the communication between CANape and the ECU, the setup of the DAQ-list is done via regular CRO-DTO. Once all ODTs of the DAQ are setup the synchronous transmission of measurements may be initiated by the CCP-master (CANape).


Block Parameters: MainDlg_CPP

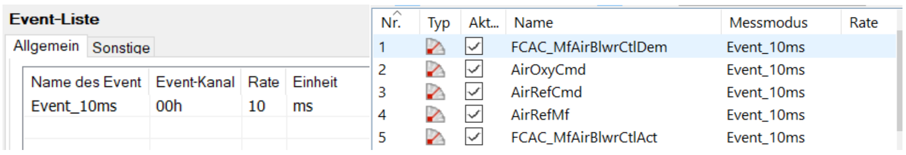
CRO DTO Id format: IO_CAN_STD_FRAME

☑ Enable DAQ sync

☐ Enable DAQ alternate

Total ODTs for DAQ lists: 40

Single DAQ-setup with the IOLibrary


Event-Liste

Allgemein | Sonstige

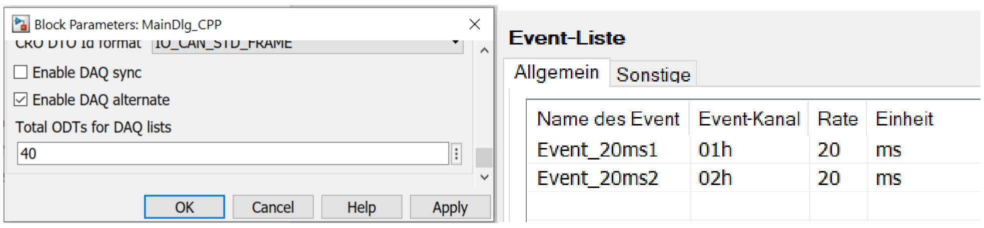
Name des Event	Event-Kanal	Rate	Einheit
Event_10ms	00h	10	ms

Nr.	Typ	Akt...	Name	Messmodus	Rate
1		✓	FCAC_MfAirBlwrCtlDem	Event_10ms	
2		✓	AirOxyCmd	Event_10ms	
3		✓	AirRefCmd	Event_10ms	
4		✓	AirRefMf	Event_10ms	
5		✓	FCAC_MfAirBlwrCtlAct	Event_10ms	

Corresponding setup in CANape

It is necessary to define an event with a rate of the cycle time (in this example: *10ms*). In order to fill the ODTs in the DAQ-list accordingly the measurements need to be assigned to the corresponding event (in this example: *Event_10ms*).

11) CCP Configuration with two DAQs enabled:
In this configuration again one CRO-DTO handshake communication is possible in each cycle. Beside establishing the communication between CANape and the ECU, the setup of two DAQ-lists is done via regular CRO-DTO. Each DAQ-list receives half of total ODTs defined in the mask of the IOLib-MainDlg block. (e.g. total ODTs = 40, resulting in 20 for each DAQ-list)
Once all ODTs of the DAQs are setup the synchronous transmission of measurements may be initiated by the CCP-master (CANape). These lists are processed alternating meaning that in odd cycles one list is processed while in even cycles the other will be processed. Doing so, will double the number of signals that may be viewed on the expense of the time resolution.
It is necessary to define two events with rates of doubled cycle time (in this example: *20ms*). In order to fill the ODTs in the DAQ-list accordingly, the measurements need to be assigned to one of the alternating events (in this example: *Event_20ms1 or Event_20ms2*).


Block Parameters: MainDlg_CPP

CRO DTO Id format: IO_CAN_STD_FRAME

☐ Enable DAQ sync

☑ Enable DAQ alternate

Total ODTs for DAQ lists: 40

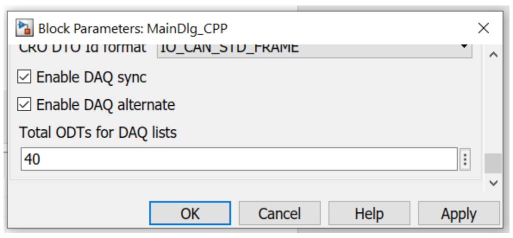
Event-Liste

Allgemein | Sonstige

Name des Event	Event-Kanal	Rate	Einheit
Event_20ms1	01h	20	ms
Event_20ms2	02h	20	ms

Definition of 2 DAQ list in the IOLibrary with corresponding events in CANape

12) CCP Configuration with one fast DAQ and two alternate DAQs enabled:
    In this configuration again one CRO-DTO handshake communication is possible in each cycle. Beside establishing the communication between CANape and the ECU, the setup of three DAQ-lists is done via regular CRO-DTO. Each DAQ-list receives a third part of total ODTs defined in the mask of the IOLib-MainDlg block (e.g. total ODTs = 40, resulting in 13 for each DAQ-list)
    Once all ODTs of the DAQs are setup the synchronous transmission of measurements may be initiated by the CCP-master (CANape). These lists are processedin the way, that first the fast DAQ is processed with the basic cycle time (e.g. 10ms) after this alternating processing of the other two DAQs continues. Alternating means that in odd cycles one list is processed while in the even cycles the other will be processed. By this definition one list is processed in 10ms while the other two are alternatingly processed in 20ms cycles.


Block Parameters: MainDlg_CPP

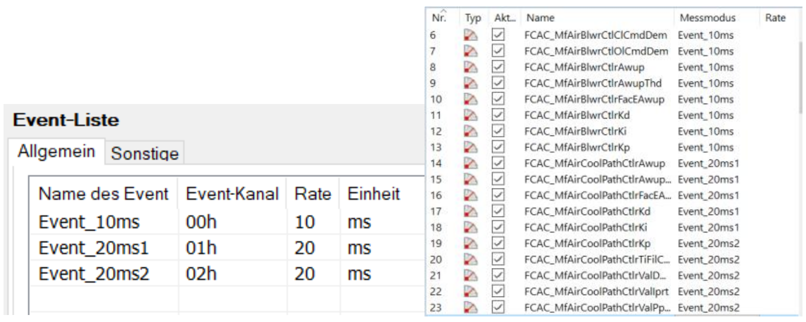
CRO DTO Id format: IO_CAN_STD_FRAME

☑ Enable DAQ sync

☑ Enable DAQ alternate

Total ODTs for DAQ lists: 40

Triple DAQ list setup with the IOLibrary


Event-Liste

Name des Event	Event-Kanal	Rate	Einheit
Event_10ms	00h	10	ms
Event_20ms1	01h	20	ms
Event_20ms2	02h	20	ms

Nr.	Typ	Akt...	Name	Messmodus	Rate
6		✓	FCAC_MfAirBlwrCtlClCmdDem	Event_10ms	
7		✓	FCAC_MfAirBlwrCtlOlCmdDem	Event_10ms	
8		✓	FCAC_MfAirBlwrCtlrAwup	Event_10ms	
9		✓	FCAC_MfAirBlwrCtlrAwupThd	Event_10ms	
10		✓	FCAC_MfAirBlwrCtlrFacEAwup	Event_10ms	
11		✓	FCAC_MfAirBlwrCtlrKd	Event_10ms	
12		✓	FCAC_MfAirBlwrCtlrKi	Event_10ms	
13		✓	FCAC_MfAirBlwrCtlrKp	Event_10ms	
14		✓	FCAC_MfAirCoolPathCtlrAwup	Event_20ms1	
15		✓	FCAC_MfAirCoolPathCtlrAwup...	Event_20ms1	
16		✓	FCAC_MfAirCoolPathCtlrFacEA...	Event_20ms1	
17		✓	FCAC_MfAirCoolPathCtlrKd	Event_20ms1	
18		✓	FCAC_MfAirCoolPathCtlrKi	Event_20ms1	
19		✓	FCAC_MfAirCoolPathCtlrKp	Event_20ms2	
20		✓	FCAC_MfAirCoolPathCtlrTiFilC...	Event_20ms2	
21		✓	FCAC_MfAirCoolPathCtlrValD...	Event_20ms2	
22		✓	FCAC_MfAirCoolPathCtlrVallprt	Event_20ms2	
23		✓	FCAC_MfAirCoolPathCtlrValPp...	Event_20ms2	

Corresponding setup in CANape

It is necessary to define an event with a rate of the cycle time (in this example: *10ms*) and two events with rates of doubled cycle time (in this example: *20ms*). In order to fill the ODTs in the DAQ-list accordingly the measurements need to be assigned to one of the events (in this example: Event_10ms, *Event_20ms1 or Event_20ms2*).

13) Processing of the DAQ is performed in the remainder of processing time between the end of the execution of the application and start of the next cycle. If too many signals need to be transferred via DAQ an overrun situation may occur. As rule of thumb one ODT (=7bytes of payload) will be processed within 0.25ms when running 500kBaud on the corresponding CAN-cannel. Due to performance reasons it is not recommended to run CAN for CCP communication lower than 250kBaud.

14) The file *ccp_interface.c* which shapes the communication between application and Vector's CCP-stack is ready to be used for the HY-TTC580 in general without any modification. There are two exceptions:

- If any changes in the memory layout apply, specially related to `stack_size` in the linker script file `mem_ttc500_bl.lsf` the `#define MIN_RAM_ADDR` in ccp_interface needs to be adapted to the lowest ram address given in `mem_ttc500_bl.lsf`. (Typically, no changes are made for the linker script file, so this point may be skipped)
- If using a variant of the HY-TTC500 family without external memory (e.g. HY-TTC510) the following setup in *ccp_interface.c* should be used:
  ```
  #define MIN_EXTRAM_ADDR  0x0
  #define MAX_EXTRAM_ADDR  0x0
  ```