



[Help](#)


DCS800  
DC Drives



**CoDeSys Exercise**  
**Control the drive**  
**with a logic**  
**G562e\_a\_b Part 11**

eLearning





© Copyright 11/8/2021 / ABB. All rights reserved.  
CODESYS\_11R0101 page 1

Welcome to the CoDeSys training module for the DCS800, ABB DC Drives.


If you need help navigating this module, please click the Help button in the top right-hand corner. To view the presenter notes as text, please click the Notes button in the bottom right corner.

[Help](#)

## Basics

- Before starting the exercise, it is necessary to know the following parts:
  - Create a new program
  - Task configuration
  - Important functions and icons in CoDeSys
  - Communication with the drive
  - Memory Card handling
  - Drive basics and commissioning (e.g. Wizard)
  - Local control and IO handling

© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 2



Attention, this exercise is much more complex than the other parts. It's a complete project which demands that the user have experience with the commissioning of DCS800 and the local control of a drive.

It is necessary that the participant knows the approach to implement an application and configuration.

## Objectives

**After completing this module, you will be able to**

- Create a complete project with CoDeSys

**ABB**


After completing this module, you will be able to create a complete project with CoDeSys.

Help

## Exercise

- We build a complete project with the following themes:
  - We design a new function block
  - We integrate this block in a program
  - We are reading and writing parameters
  - We define a second program for new user events
  - We control the drive with an application
  - We start the application and test the response of the drive

© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 4



The exercise includes several jobs which must be solved to get a whole project. So we design a new function block and integrate them into a program. Also reading and writing parameters from the standard firmware is a part of this project. If the program is ready and implemented it is necessary to test it.

Help

## Description

- The project is a start logic for a motor
- It controls the speed reference and the actual speed
- Release of the drive is only possible if the speed reference is lower than minimum speed (new parameter)
- That means the drive will be controlled over speed reference potentiometer

© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 5

ABB


The project is a start logic for a motor. The motor logic controls the speed reference and the actual speed of the motor. Idea of this logic is that the release of the drive is only possible if the speed reference is lower than minimum speed. That means the drive will be controlled over speed reference potentiometer.

[Help](#)

## Overview

- Step 1: Open a new program
- Step 2: Design of the new function block  
*MotorLogic*
- Step 3: Insert function blocks
- Step 4: Insert Events
- Step 5: Define new parameters
- Step 6: Commissioning step by step and  
fault tracing
- Step 7: Test and Documentation

© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 6



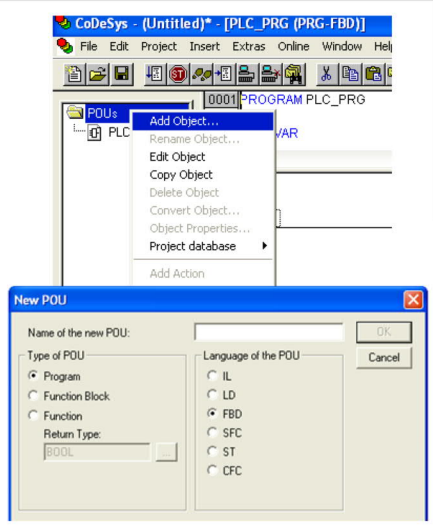
This is the overview about the whole project.

In step 1 the new program must be created. Step 2 deal with the design of the new function block called "MotorLogic". After this the other function blocks must be implemented and connected. Step 4 and 5 gives you information about new user events and parameters.


Commissioning step by step is the next step in our program. Here it is important to find faults and errors in the software. Last step is the test and documentation of the project.

## Step 1: Add a POU

- Start a new program
- Select *DCS800* target
- Choose type *Program* and language *FBD*
- POU name: *PLC\_PRG*
- Click with right mouse button to *POU*
- Select *Add Object*
- A new window opens



The screenshot shows the CoDeSys software interface. A right-click context menu is open over the 'POU' folder in the project tree, with 'Add Object...' selected. Below it, the 'New POU' dialog box is displayed. In the dialog, 'Name of the new POU:' is 'PLC\_PRG'. Under 'Type of POU', 'Program' is selected. Under 'Language of the POU', 'FBD' is selected. The 'Return Type' is set to 'BOOL'. The 'OK' button is highlighted.



© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 7

Start a new project and select the DCS800 target. For this program we use the programming language “Function Block Diagram”. This “POU” is used to build the program.  
But we need a second “POU” for the new function block called “Motor Logic”.

Help

## Step 1: POU “Motor Logic”

- Select type *Function block*
- Type in a Name of POU *MotorLogic*
- Choose language *FBD*
- Click OK
- The new POU can be found in the root directory

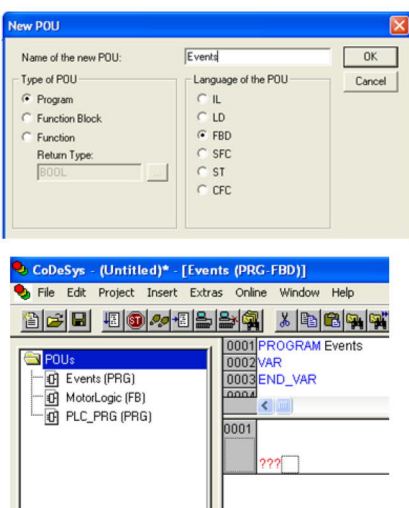
© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 8


Select the type “Function Block” for this “POU” and programming language “Function Block Diagram”. All “POUs” of the project are shown in the root directory on the left side. In CoDeSys it is possible to define many “POUs” which can be programs, function blocks or functions.



## Step 1: POU “Events”

- Add the third POU (see slides before)
- Select *Program*
- Type in the name of the POU *Events*
- Choose language *FBD*
- Click OK
- The new POU is added





© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 9

Now we need a third “POU” which is used to work with user events. It’s from type program which work with programming language “Function Block Diagram”. So, the project is split into 2 parts with a defined function block.

## Step 2: New function block

- With CoDeSys it is possible to create a new function block
- First it is important to define all inputs and outputs with variables

Text input:  
iN\_Ref: INT;

Definition Type:  
i = integer  
b = boolean



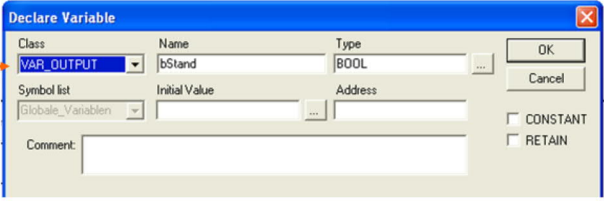
If you will define new function blocks it is an important questions which inputs and outputs are necessary. You can see after the variables the type of the variable. This is also an important information for the design of the block.


Define all input and output variables like in the picture below. For good documentation, add a comment, please.

[Help](#)

## Step 2: Declaration window

- There are 3 types of classes:
  - VAR\_INPUT:               input of function blocks
  - VAR\_OUTPUT:            output of function blocks
  - VAR:                    local variables
- Define the class in **Declare Window**



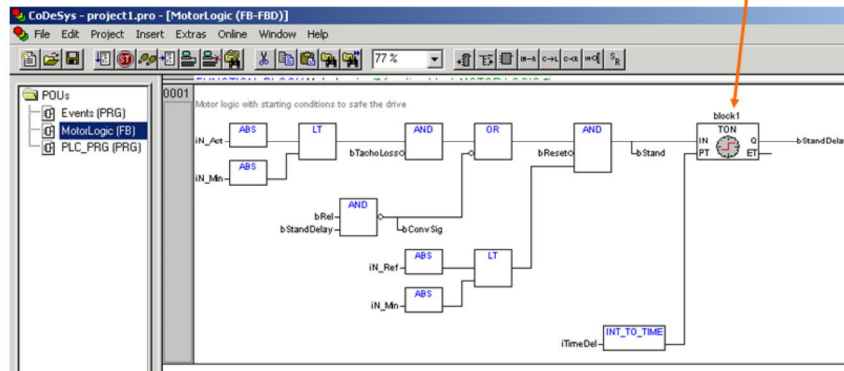


© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 11

If you type in variables, you get the declaration window in which the class and the type has to be defined. The class “VAR” accord to local variables which are only available in the function block. Class “VAR INPUT and OUTPUT” defines connection variables on the function block.

## Step 2: Build FB “Motor Logic”

- Build the program with function blocks
- Start with the function block on the right



**ABB**


Next step is to create the function block “Motor Logic”. Insert all function blocks in network 1 which are shown on the picture. Start with the last function block ease the approach to build the program.

Help

## Step 2: Build FB “Motor Logic”

- If the actual speed *InAct* of the motor is lower than a defined minimum value
- **&-Connection**
- If the speed reference (AI) *InRef* is greater than minimum value
- The motor start if the condition is fulfilled
- That means the motor will be controlled via analog input
- If the motor is standing still a special delayed signal will be generated
- The other inputs are features

© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 13



The new function block controls the motor. If the actual speed “InAct” of the motor is lower than a defined minimum value. This function is connected with an AND-element. If the speed reference “InAct” is greater than minimum value. The motor start if the condition is fulfilled. That means the motor will be controlled via analog input. If the motor is standing still a special delayed signal will be generated. The other inputs are features.

Help


## Step 2: FB “MotorLogic”

- After development of a new block, it can be used in the main program PLC\_PRG
- The name *MotorLogic* is the name of this block and must be called in PLC\_PRG
- In PLC\_PRG you can see the function block in the visualization with inputs and outputs

MotorLogic

iN_Ref	bStand
iN_Act	bConvSig
iN_Min	bStandDelay
bTachoLoss	
iTimeDel	
bReset	
bRel	

© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 14



After development of a new block, it can be used in the main program “PLC\_PRG”. The name “Motor Logic” is the name of this block and must be called for using in main program “PLC\_PRG”. If you use a graphical programming language the new function block is shown as box with inputs and outputs. The function of this block is hidden inside.

[Help](#)

## Step 3: Build main program

■ Insert the function blocks and the variables in POU PLC\_PRG like in the picture below:

0001 read parameter 2.01 (SpeedRef2) physical value  
 block1  
 2-byGroup wVal — SPEED\_REF  
 1-byIndex iErrCode

0002 read parameter 1.04 (ProcessSpeed) physical value  
 block2  
 1-byGroup wVal — SPEED\_ACT  
 4-byIndex iErrCode

0003 read digital inputs  
 block5  
 8-byGroup wOutput — REL  
 5-byIndex iErrCode  
 FALSE-blndirect wPointerpar

0004

Variables:  
 SPEED\_REF: INT; (VAR\_GLOBAL)  
 SPEED\_ACT: INT; (VAR\_GLOBAL)  
 REL: INT; (VAR\_GLOBAL)

Description:  
 Block1: Read speed reference 2.01 physical value in rpm  
 Block2: Read actual speed 1.04 physical value in rpm  
 Block5: Read digital inputs 8.05

© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 15

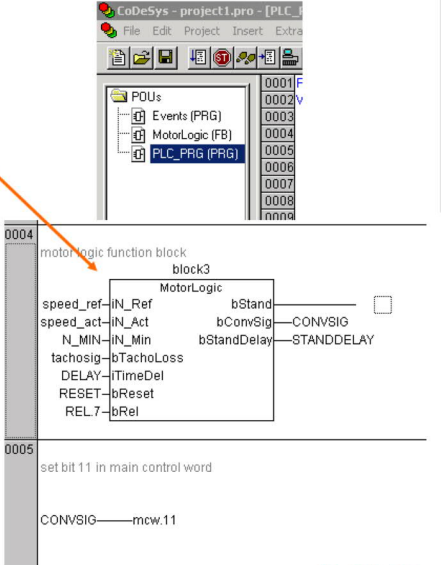
Next step is to build the main program “PLC\_PRG”. Put in the function blocks like in the picture and define the variables. We need global variables for the variables speed reference, speed actual and “REL” because they should be connected later to another POU.

The functionality of this three blocks are the directly reading of parameters.

## Step 3: Insert FB “Motor Logic”

- Enlarge the program with FB *Motor Logic*
- Type in the variables

Variables:  
N\_MIN: INT; (VAR\_GLOBAL)  
tachosig: BOOL; (VAR)  
DELAY: INT; (VAR\_GLOBAL)  
RESET: BOOL; (VAR\_GLOBAL)  
REL: INT; (VAR\_GLOBAL)  
CONVSIG: BOOL; (VAR\_GLOBAL)  
STANDDELAY: BOOL; (VAR\_GLOBAL)



© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 16

ABB

Now the main program must be enlarged with the new function block “Motor Logic”. Connect the the variables like in the picture and write comments in the several networks. Network 5 shows a conversion between a boolean and a word type. With this operation it is possible to set a bit in a word with a boolean value. In this exercise we set bit 11 in main control word.



Help

## Step 3: Bit selection

- **Select a Bit of a Word:**
  - This is a typical conversion in CoDeSys
  - A Boolean value set a bit in a word (bit 11 in mcw)
- **The other way:**
  - *REL* is a word variable
  - Bit 7 of this word should be set input *bRel* of FB *MotorLogic*

The diagram illustrates two methods for bit selection. The top method shows a variable 'mcw' with bit 11 set, labeled 'set bit 11 in main control word'. The bottom method shows a 'MotorLogic' function block (block3) with inputs 'REL.7' and 'bRel', and outputs 'REL' and 'bRel'. The 'REL' output is connected to the 'wOutput' input of a 'ParGet' block (block5), which also has inputs 'byGroup' (8) and 'byIndex' (5). The 'bRel' output is connected to the 'bStand' input of the 'MotorLogic' block. The 'MotorLogic' block also has inputs 'speed\_ref', 'speed\_act', 'N\_MIN', 'tachosig', 'DELAY', 'RESET', and 'REL.7', and outputs 'bStand', 'bConvSig', 'bStandDelay', 'CONVSIG', and 'STANDELAY'.

ABB

© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11RD101 page 17

The picture on the top explained again the conversion between bits and words. Another way to do this is the possibility to use a function block from the special library which is available from the CD.

This approach is from a boolean value to word. But with CoDeSys it is possible to do this in the inverse direction. Take a word variable and write it to a boolean connection. Select the needed bit after the dot.

The picture on the bottom shows the connection of variable "REL" to function block "Motor Logic". We need for this application only bit 7. So we have to set the notation "REL dot 7" to input "REL" of function block "Motor Logic".

Help

## Step 3: Complete PLC\_PRG

- Put in the last function block *ParSet* and check your declaration window:

The screenshot shows the CoDeSys software interface. The main window displays the PLC\_PRG program editor with the following code:

```

0001 PROGRAM PLC_PRG (* main program *)
0002 VAR
0003   block1: ParRead; (* parameter get = read a parameter value from firmware *)
0004   block2: ParRead; (* --|-- *)
0005   block3: MotorLogic; (* call function block MOTOR LOGIC *)
0006   block4: ParSet; (* set a selected parameter *)
0007   block5: ParGet; (* parameter get *)
0008   mcw: INT; (* variable type integer *)
0009   tachosig: BOOL; (* tacho loss signal *)
0010 FND_VAR

```

The declaration window (FND\_VAR) is open, showing the declaration of the *ParSet* function block. The declaration is as follows:

```

block4
ParSet
7-byGroup
1-byIndex
FALSE-blindirect
0-wAND
mcw-wOR
iErrCode
wPointerpar

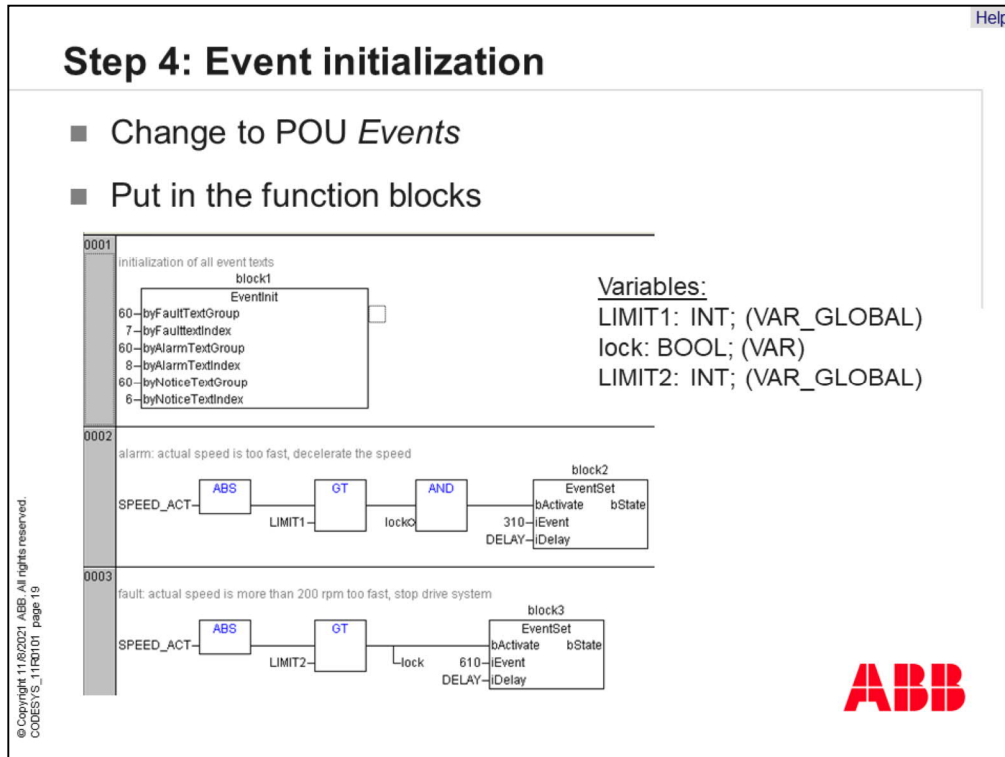
```

The ABB logo is visible in the bottom right corner of the software interface.

© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 18

Now we have to complete the main program with network 6. This network includes function block parameter set because we will write to parameter 7.01, the main control word of the drive.

The picture shows also the declaration window with the documentation. Correct your application that it looks like the picture. If you have done this the main program is ready and we can go to the next step.



Our next job is to define the POU “Events”. Here we will define the trigger signals for all events and the initialization.

First function block in network 1 is the event initialization. With this block the event texts are allocated to the events. In this exercise we save the event messages in parameters 60.06, 60.07 and 60.08.

Network 2 gives information about the trigger signals. Event 310 should be set if the actual speed of the drive is greater than the limit. The same procedure is in network 3. There event 610 will be triggered.

Help

## Step 4: Event activation

- Complete the program with network 4
- Check your declaration window

CoDeSys - project1.pro - [Events (PRG-FBD)]

File Edit Project Insert Extras Online Window Help

100 %

POUs

- Events (PRG)
- MotoLogic (FB)
- PLC\_PRG (PRG)

```

0001 PROGRAM Events
0002 VAR
0003   block1: EventInit;      (* event initialization *)
0004   block2: EventSet;       (* activate alarm message 310 *)
0005   block3: EventSet;       (* activate fault message 610 *)
0006   block4: EventSet;       (* activate notice 810 *)
0007   lock: BOOL;            (* lock signal *)

```

0003 SPEED\_ACT—ABS—LIMIT2—lock

0004 notice: motor is standing still; drive start is possible

STANDDELAY—bActivate—bState

810—iEvent—DELAY—Delay

block3 EventSet

bActivate—bState

610—iEvent—DELAY—Delay

ABB

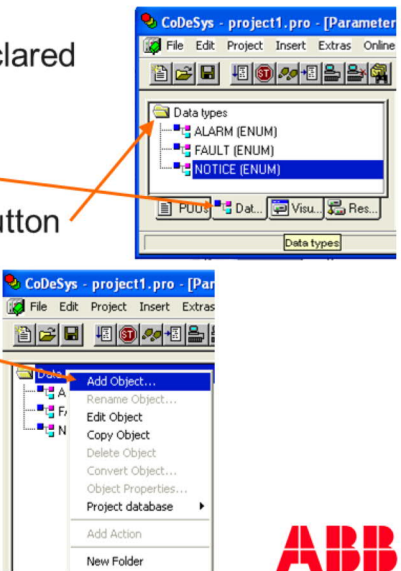
Last event is the notice message 810. Check the declaration window so that it looks like the picture.

Now the event program is ready.

20

## Step 4: Data types for events

- Parameter texts must be declared as data type *Enum*
- Select *Data Types*
- Click with the right mouse button to Data types
- Select *Add Object*

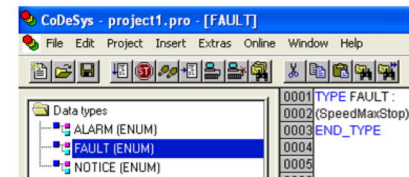
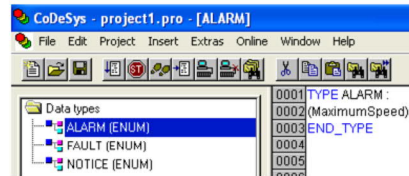
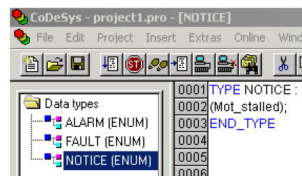


© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 21

Before we defined the events. Now we must set the message texts for this project. Message texts have a own data type in CoDeSys. So, we must change to item "Data types" and add new objects. Each object is a new data type.

## Step 4: Event Texts

- Type in *FAULT* and click OK
- In the same way add *NOTICE* and *ALARM*
- Click to *ALARM* and type in the text ...



We need a name for the new data types, and it will be avoided to choose a name which associate with the event. So, we use names like "Fault, Alarm and Notice". In this new data types, we must set the event messages.

Delete the structure and write a message text into brackets. Note, only 12 characters are allowed! If you need more than 1 text, set a comma between the texts in the bracket. The first string in the bracket accord to the first event number.

[Help](#)

## Step 5: Global variables

- Connection of variables and parameters are possible if the variable is a global one!
- Check your global variables!

CoDeSys - project1.pro - [Globale\_Variablen]

File Edit Project Insert Extras Online Window Help


Resources

- Global Variables
- Globale\_Variablen
- Variablen\_Configuration (VAR\_CO
- library DCS800lib.lib 2.3.06 09:37:44
- library lecSfc.lib 26.11.02 10:23:26
- library SysLibTime.lib 8.10.03 17:33:40
- library SysTaskInfo.lib 8.10.03 16:33:40
- Alarm configuration
- Library Manager
- Log
- Parameter Manager
- PLC Configuration

```

0001 VAR_GLOBAL
0002 SPEED_REF: INT; (* speed reference: parameter 60.01 *)
0003 SPEED_ACT: INT; (* actual speed: parameter 60.02 *)
0004 N_MIN: INT; (* minimum speed: parameter 61.01 *)
0005 DELAY: INT; (* time delay: parameter 61.02 *)
0006 RESET: BOOL; (* reset activation: parameter 61.03 *)
0007 REL: INT; (* main control word: parameter 60.03 *)
0008 LIMIT1: INT; (* alarm limit: parameter 61.04 *)
0009 LIMIT2: INT; (* fault limit: parameter 61.05 *)
0010 CONVSIG: BOOL; (* converter signal: parameter 60.04 *)
0011 STANDDELAY: BOOL; (* stand signal: parameter 60.05 *)
0012 appnotice: NOTICE; (* notice texts: parameter 60.06 *)
0013 appfault: FAULT; (* fault texts: parameter 60.07 *)
0014 applalarm: ALARM; (* alarm texts: parameter 60.08 *)
0015 END_VAR

```



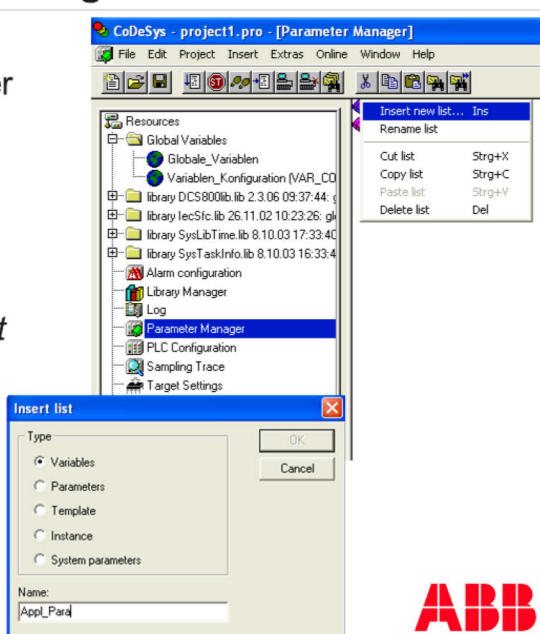
© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 23

Global variables are used to connect programs and parameters. So, we defined before parameters as global ones which are shown in the list. The last 3 rows shows the new data types of the events which are set 1 slide before. Fill in the global variables for this events so that they can be connected to parameters. This variables are needed later in the parameter manager.

Now the variable definitions are complete, and we can do the next step.

## Step 5: Parameter Manager

- Change to parameter manager
- Click with the right mouse button to the white window
- Select *Insert new list*
- Type in *Appl\_Para*



© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 24

Change to the parameter manager in folder resources and insert a new list. Select variables and set the name “Application Parameters” in a short notation to the list. A new list is created!



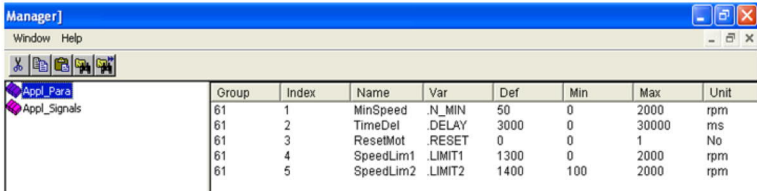
[Help](#)

## Step 5: Application parameters


■ Fill in the variables and numbers!

- Name: This name is the parameter name
- Var: This is the connection to a variable
  - Don't forget the dot! Press F2 to select a variable!
- Def: This is the default value
- Min/Max: Minimum / maximum value
- Unit: This unit is shown in the parameter list

© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 25



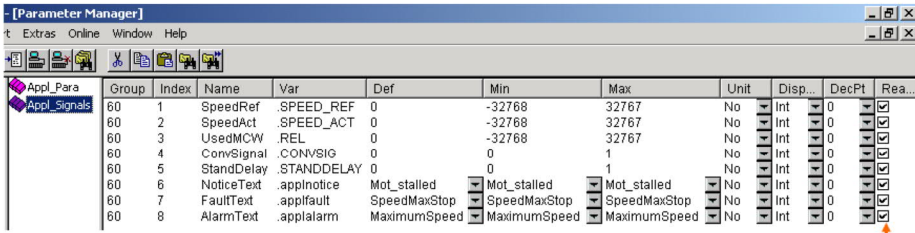
Group	Index	Name	Var	Def	Min	Max	Unit
61	1	MinSpeed	.N_MIN	50	0	2000	rpm
	2	TimeDel	.DELAY	3000	0	30000	ms
	3	ResetMot	.RESET	0	0	1	No
	4	SpeedLim1	.LIMIT1	1300	0	2000	rpm
	5	SpeedLim2	.LIMIT2	1400	100	2000	rpm



There are several fields to declare parameters. Fill in the variables and numbers from the picture on the bottom. The field “Name” is the parameter name and “Var” the connection to the variable in CoDeSys. Range of the data is given by the minimum and maximum fields. All fields are parameters that means the tick in field “Read Only “ isn’t set.

## Step 5: Application signals

- Add a second list named *Appl\_Signals*
- Fill in the variables and numbers!



**Further Settings:**  
 These are signals. That means you cannot write to this parameters. In this case select **Read Only!!!**  
 Parameter 60.6, 60.7 and 60.8 includes the texts for the events. Select the first row in the fields!

**ABB**

© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 26

Add a second list named “Application Signals” and add lines to the list. Fill in the variables and numbers like in the picture. Set a tick in the fields “Read Only” because these are signals, which aren’t writeable.

Now the application is ready and we can go to the next step.


[Help](#)

## Step 6: Fault Tracing

- Build your program (F11)
- If you see this message, you can be happy: The program is ok!
- Otherwise you have to find the errors in the program. This is not easy!

Hardware-Configuration  
POU indices:130 (25%)  
Size of used data: 462 of 32768 bytes (1.41%)  
Size of used retain data: 0 of 512 bytes (0.00%)  
Size of GlobalInit: 0 bytes  
Code Size: 6912, Data Size: 45208, Used global data 462 of 32768  
Code size: 6912 bytes  
0 Error(s), 0 Warning(s).

Implementation of POU 'PLC\_PRG'  
**Error 4001: PLC\_PRG (2): Identifier 'BLOCK' not defined**  
**Error 4052: PLC\_PRG (2): 'block' must be a declared instance of function blo**  
Implementation of task 'EventTask'  
Implementation of the task configuration  
Implementation of task 'PLC\_Task'  
Check of the parameter configuration  
Hardware-Configuration  
2 Error(s), 0 Warning(s).



© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 27

An important thing in CoDeSys is fault tracing. In bigger projects it is normal that they are any errors in program. CoDeSys shows the user all errors with the applied line or network. Build your program and look to the message window. If you see the message “0 errors and 0 warnings” you can be happy. Otherwise, you must find the errors in the program.

Help

## Step 6: Find errors

■ A double click to the red error shows the FB

In this case the program cannot find identifier Block. If you look into declaration window, the name of the block must be BLOCK2. Other faults are not correct names (BLOCK↔BOLCK) or a missing identifier in the list.

Implementation of POU 'PLC\_PRG'

**Error 4001: PLC\_PRG (2): Identifier 'BLOCK' not defined**

**Error 4052: PLC\_PRG (2): 'block' must be a declared instance of function block 'ParRead'**

Implementation of task 'EventTask'

Implementation of the task configuration

Implementation of task 'PLC\_Task'

Check of the parameter configuration

Hardware-Configuration

2 Error(s), 0 Warning(s).

```

0001 PROGRAM PLC_PRG      (* main program *)
0002 VAR
0003   block1: ParRead;      (* parameter get = read a parameter value from firmware *)
0004   block2: ParRead;      (* --||-- *)
0005   block3: MotorLogic;   (* call function block MOTOR LOGIC *)
0006   block4: ParSet;       (* set a selected parameter *)
0007   block5: ParGet;       (* parameter get *)
0008   mcw: INT;             (* variable type integer *)
0009   tachosig: BOOL;       (* tachometer loss signal *)
0010 END VAR

```

Attention:  
This is only an example for an error. There are many errors in CoDeSys! If you have more than 10 errors, check the errors step by step.

ABB

© Copyright 11/8/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 28

Find the errors when you click to the error line and the editor window jump to the applied line or network.

If there are many errors check the correct declaration and the several characters and types to find the errors.


Help

## Step 6: Errors in parameter list

- The fields Def, Min and Max must be a value of the variable:
  - REL: INT; (-32768 < REL < 32767)
  - Conv: BOOL; (0 < Conv < 1)
  - Applnotice: ENUM; (Mot\_stalled == Mot\_stalled)

Group	Index	Name	Var	Def	Min	Max	Unit	Disp...	DecPt	Rea...
60	1	SpeedRef	SPEED_REF	0	-32768	32767	No	Int	0	
60	2	SpeedAct	SPEED_ACT	0	-32768	32767	No	Int	0	
60	3	UsedMCW	REL	0	-32768	32767	No	Int	0	
60	4	ConvSignal	CONVSIG	0	0	1	No	Int	0	
60	5	StandDelay	STANDDELAY	0	0	1	No	Int	0	
60	6	NoticeText	applnotice	Mot_stalled	Mot_stalled	Mot_stalled	No	Int	0	
60	7	FaultText	applfault	SpeedMaxStop	SpeedMaxStop	SpeedMaxStop	No	Int	0	
60	8	AlarmText	applalarm	MaximumSpeed	MaximumSpeed	MaximumSpeed	No	Int	0	

Display: don't care  
 DecPt: decimal point  
 Read only: This is only for signals  
 No Save: Save parameter to Flash (cycle saving is not allowed!)




© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 29

Errors in the parameter list are dangerous, because incorrect parameters will not be generated. Check the maximum and minimum of the selected variables. Boolean variables gets the character 0 and 1. In Enum variables it is necessary to select the first entry.

Help

## Step 6: Task configuration

- Select the task cycle like the picture below



PLC Task:  
- 5 ms cycle

Task attributes

Name:

Priority(0..31):

Type

☒ cyclic


☐ freewheeling

☐ triggered by event

☐ triggered by external event

Properties

Event:



Event Task:  
- 100 ms cycle

Task attributes

Name:

Priority(0..31):

Type

☐ cyclic


☐ freewheeling

☐ triggered by event

☒ triggered by external event

Properties

Event:



© Copyright 11/6/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 30

If the program is ready, we must configure the task configuration. In this application we need two different tasks. The main program PLC\_PRG work with the 5 ms cycle. The event program needs the 100 ms task.

Help


## Step 6: Drive configuration

- It is necessary that the drive is in Local Mode
- Change Parameter settings with the panel, DW or DWL

10.01: Local Mode	(drive controlled via DI's)
10.15: DI7	(main contactors ON)
10.16: MCW11	(release drive with application program)

Check also your speed reference settings:  
AI1 must be connected with a potentiometer

Check your speed feedback and the controller settings:  
e.g. Startup assistant (Wizard)

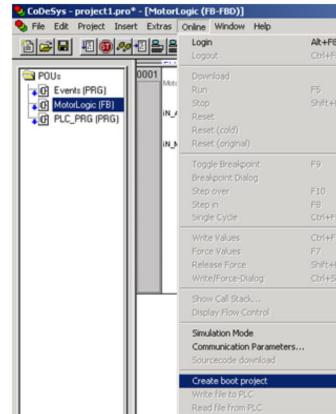


© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11R0101 page 31

It is necessary that the drive is in "Local Mode". Change parameter settings with the panel, DriveWindow and DriveWindow light.

## Step 6: Download

- Select your communication channel and download the program
- Click *Run*
- Select *Create Boot Project*
- Check the message that
  - the program is saved
  - the parameters are generated



Select your communication channel and download the program. Click “Run” and select “Create Boot Project” to save the program on the Memory Card. When the program is saved correctly you get the message that the program is saved and the user parameters are generated.




[Help](#)

## Step 7: Test the program

- Close the main contactor with DI7
- DI8 must be active; inactive DI8 disable the motor logic (bypass)
- Turn the potentiometer until the motor is running
- Add the speed until you see the alarm message
- More speed activate the fault
- Check also the notice if the drive is standing still
- Change the new parameters and play

© Copyright 11/6/2021 ABB. All rights reserved.  
CODESYS\_11RD101 page 33



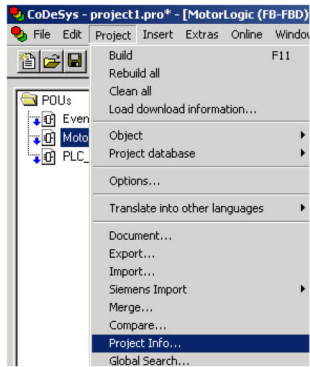
Now we have to test the program with the hardware. Close the main contactors with digital input 7. Digital input 8 must be active. An inactive digital input 8 disables the motor logic and close a bypass.

Turn the potentiometer until the motor is running. Increase the speed reference until you see the alarm message. If the motor speed increased more and more until a fault is activated. Check also the notice if the drive is standing still. Change the new parameters and play with the application.

Help

## Step 7: Documentation

- For all applications it is necessary to document it for other users and for your own protection



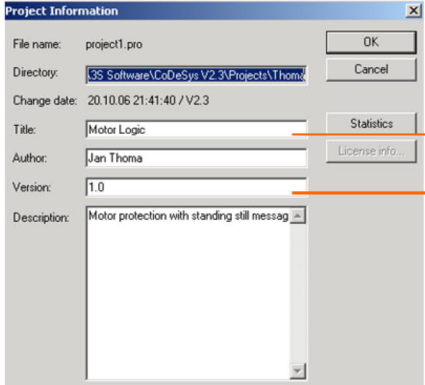
CoDeSys - project1.pro\* - [MotorLogic (FB-FBD)]

File Edit Project Insert Extras Online Window

Build F11  
Rebuild all  
Clean all  
Load download information...

PDU's  
Even  
Moto  
PLC

Object  
Project database  
Options...  
Translate into other languages  
Document...  
Export...  
Import...  
Siemens Import  
Merge...  
Compare...  
**Project Info...**  
Global Search...



Project Information

File name: project1.pro OK Cancel

Directory: C:\SS Software\CoDeSys V2.3\Projects\Thoma

Change date: 2010.06.21:41:40 / V2.3


Title: Motor Logic → **Parameter: 4.03**

Author: Jan Thoma

Version: 1.0 → **4.12**

Description: Motor protection with standing still message

Statistics License info



© Copyright 11/8/2021 ABB. All rights reserved.  
 CODESYS\_11R0101 page 34

For all applications it is necessary to document it for other users and for your own protection. You can find "Project Information" in menu Project. The field "Title" appears in parameter 4.03 and "Version" in parameter 4.12.


[Help](#)

## Summary

### Key points of this module

- Learn to create a complete project in CoDeSys

© Copyright 11/6/2021 ABB. All rights reserved.  
CoDeSys\_11R0101 page 35



Key point of this module is learning to create a complete project in CoDeSys.

## Additional information

- Further information:
  - Application Manual (3ADW 000 199)
  - Firmware Manual (3ADW 000 193)
  - Hardware Manual (3ADW 000 194)
- CoDeSys support:
  - [www.3s-software.com](http://www.3s-software.com)
- Drive support and libraries:
  - ABB DC-Supportline



## Glossary

- **CoDeSys**  
Controller Development System (software tool)
- **Memory Card**  
Flash memory
- **DriveWindow Light**  
Software Tool for commissioning and maintenance using AC/DC
- **Target**  
Interface between Drive and CoDeSys tool
- **Control Builder**  
Whole system with software and hardware
- **PLC\_PRG**  
Main program which is used in all applications
- **POU**  
Program Organization Unit
- **Library**  
It includes function blocks which are given or designed by other users





Power and productivity  
for a better world™

Thank you for your attention. You may now go ahead and move on to the next unit.