

Motion Terminal VTEM

Festo_VTEM_DevCon
Versions _._.37

Library for the control (Device Control) of the Motion
Terminal VTEM from Festo
from firmware version 4.22.x

FESTO

Description

TIA Portal
from V19

Title..... Festo_VTEM_DevCon
Version of the document1.07
Originalde
Author Festo

Last saved on 2026/05/06

Version information and library history

=====

Festo SE & Co. KG, Esslingen
Copyright © 2026, all rights reserved

Siemens TIA Portal function block library for VTEM
Version 3.7

Designed for use with
- Festo Motion Terminal VTEM
- Firmware version 4.22.x

Created
- with TIA V19
- 2026-02

=====

H I S T O R Y

=====

Version 3.7

object	version	change
eRespondingReadMalfunctionList	1.00	- no changes
eResponseTeachInRun	1.06	- no changes
eValveMode	1.02	- no changes
eValveState	1.00	- no changes
eResponseToValveModeSet	1.01	- no changes
eValveType_MA_01_02	1.00	- no changes
eMA100OperatingMedium	1.00	- no changes
eSavingResponse	1.01	- no changes
eTransferResponse	1.03	- no changes
stMalfunction	1.01	- no changes
stTimeStamp	1.00	- no changes
stTransferPackage	1.00	- no changes
FB_Read_Bits_From_Byte	1.01	- no changes
FB_Read_Bits_From_Word	1.02	- no changes
FB_Write_Value_To_Word	1.01	- no changes
FB_AppOptionGenerator_MA_02	1.01	- optimized valve access enabled
FB_TeachInRun	1.02	- no changes
FB_ValveControl	2.04	- no changes
FB_StateInterpreter_MA_01	1.02	- optimized valve access enabled
FB_StateInterpreter_MA_02	1.03	- optimized valve access enabled
FB_StateInterpreter_MA_03	1.02	- optimized valve access enabled
FB_StateInterpreter_MA_04	1.02	- optimized valve access enabled
FB_StateInterpreter_MA_05	1.03	- optimized valve access enabled
FB_StateInterpreter_MA_06	1.02	- optimized valve access enabled
FB_StateInterpreter_MA_07	1.02	- optimized valve access enabled
FB_StateInterpreter_MA_08	1.02	- optimized valve access enabled
FB_StateInterpreter_MA_10	1.01	- optimized valve access enabled
FB_StateInterpreter_MA_11	1.03	- optimized valve access enabled

FB_StateInterpreter_MA_12	1.02	- optimized valve access enabled
FB_StateInterpreter_MA_33	1.01	- optimized valve access enabled
FB_StateInterpreter_TeachInRun	1.02	- optimized valve access enabled
FB_Download_Multiple_Parameters	2.02	- no changes
FB_Download_Single_Parameter	2.02	- no changes
FB_ReadMalfunctionList	1.02	- optimized valve access enabled
FB_Save_Configuration_Persistent	2.03	- no changes
FB_Upload_Multiple_Parameters	2.02	- optimized valve access enabled
FB_Upload_Single_Parameter	2.02	- optimized valve access enabled
FB_Read_config_AI_Module	2.01	- optimized valve access enabled
FB_Read_config_DI_Module	2.01	- optimized valve access enabled
FB_Read_config_MA_01	2.01	- optimized valve access enabled
FB_Read_config_MA_02	2.01	- optimized valve access enabled
FB_Read_config_MA_03	2.01	- optimized valve access enabled
FB_Read_config_MA_04	2.01	- optimized valve access enabled
FB_Read_config_MA_05	2.01	- optimized valve access enabled
FB_Read_config_MA_06	2.01	- optimized valve access enabled
FB_Read_config_MA_07	2.01	- optimized valve access enabled
FB_Read_config_MA_08	2.01	- optimized valve access enabled
FB_Read_config_MA_10	2.01	- optimized valve access enabled
FB_Read_config_MA_11	2.01	- optimized valve access enabled
FB_Read_config_MA_12	2.01	- optimized valve access enabled
FB_Read_config_MA_33	2.01	- optimized valve access enabled
FB_Write_config_AI_Module	2.03	- no changes
FB_Write_config_DI_Module	2.03	- optimized valve access enabled
FB_Write_config_MA_01	2.01	- optimized valve access enabled
FB_Write_config_MA_02	2.02	- optimized valve access enabled
FB_Write_config_MA_03	2.02	- optimized valve access enabled
FB_Write_config_MA_04	2.01	- no changes
FB_Write_config_MA_05	2.01	- optimized valve access enabled
FB_Write_config_MA_06	2.01	- optimized valve access enabled
FB_Write_config_MA_07	2.01	- optimized valve access enabled
FB_Write_config_MA_08	2.02	- optimized valve access enabled
FB_Write_config_MA_10	2.02	- optimized valve access enabled
FB_Write_config_MA_11	2.01	- optimized valve access enabled
FB_Write_config_MA_12	2.01	- optimized valve access enabled
FB_Write_config_MA_33	2.02	- optimized valve access enabled

Version 3.6

object	version	change
eRespondingReadMalfunctionList	1.00	- no changes
eResponseTeachInRun	1.06	- no changes
eValveMode	1.02	- no changes
eValveState	1.00	- no changes

eResponseToValveModeSet	1.01	- no changes
eValveType_MA_01_02	1.00	- no changes
eMA100OperatingMedium	1.00	- no changes
eSavingResponse	1.01	- no changes
eTransferResponse	1.03	- no changes
stMalfunction	1.01	- no changes
stTimeStamp	1.00	- no changes
stTransferPackage	1.00	- no changes
FB_Read_Bits_From_Byte	1.01	- removed AT construct - optimized valve access enabled
FB_Read_Bits_From_Word	1.02	- removed AT construct - optimized valve access enabled
FB_Write_Value_To_Word	1.01	- removed AT construct - function block numbering set to Automatic - optimized valve access enabled
FB_AppOptionGenerator_MA_02	1.00	- no changes
FB_TeachInRun	1.02	- optimized valve access enabled
FB_ValveControl	2.04	- optimized valve access enabled
FB_StateInterpreter_MA_01	1.01	- no changes
FB_StateInterpreter_MA_02	1.02	- no changes
FB_StateInterpreter_MA_03	1.01	- no changes
FB_StateInterpreter_MA_04	1.01	- no changes
FB_StateInterpreter_MA_05	1.02	- no changes
FB_StateInterpreter_MA_06	1.01	- no changes
FB_StateInterpreter_MA_07	1.01	- no changes
FB_StateInterpreter_MA_08	1.01	- no changes
FB_StateInterpreter_MA_10	1.00	- no changes
FB_StateInterpreter_MA_11	1.02	- no changes
FB_StateInterpreter_MA_12	1.01	- no changes
FB_StateInterpreter_MA_33	1.00	- no changes
FB_StateInterpreter_TeachInRun	1.01	- no changes
FB_Download_Multiple_Parameters	2.02	- function block numbering set to Automatic - optimized valve access enabled
FB_Download_Single_Parameter	2.02	- optimized valve access enabled
FB_ReadMalfunctionList	1.01	- no changes
FB_Save_Configuration_Persistent	2.03	- optimized valve access enabled
FB_Upload_Multiple_Parameters	2.01	- no changes
FB_Upload_Single_Parameter	2.01	- no changes
FB_Read_config_AI_Module	2.00	- no changes
FB_Read_config_DI_Module	2.00	- no changes
FB_Read_config_MA_01	2.00	- no changes
FB_Read_config_MA_02	2.00	- no changes
FB_Read_config_MA_03	2.00	- no changes
FB_Read_config_MA_04	2.00	- no changes
FB_Read_config_MA_05	2.00	- no changes
FB_Read_config_MA_06	2.00	- no changes
FB_Read_config_MA_07	2.00	- no changes

FB_Read_config_MA_08	2.00	- no changes
FB_Read_config_MA_10	2.00	- no changes
FB_Read_config_MA_11	2.00	- no changes
FB_Read_config_MA_12	2.00	- no changes
FB_Read_config_MA_33	2.00	- no changes
FB_Write_config_AI_Module	2.03	- optimized valve access enabled
FB_Write_config_DI_Module	2.02	- no changes
FB_Write_config_MA_01	2.00	- no changes
FB_Write_config_MA_02	2.01	- no changes
FB_Write_config_MA_03	2.01	- no changes
FB_Write_config_MA_04	2.01	- function block numbering set to Automatic - optimized valve access enabled
FB_Write_config_MA_05	2.00	- no changes
FB_Write_config_MA_06	2.00	- no changes
FB_Write_config_MA_07	2.00	- no changes
FB_Write_config_MA_08	2.01	- no changes
FB_Write_config_MA_10	2.01	- no changes
FB_Write_config_MA_11	2.00	- no changes
FB_Write_config_MA_12	2.00	- no changes
FB_Write_config_MA_33	2.01	- no changes

Version 3.5

object	version	change
eRespondingReadMalfunctionList	1.00	- no changes
eResponseTeachInRun	1.06	- no changes
eValveMode	1.02	- no changes
eValveState	1.00	- no changes
eResponseToValveModeSet	1.01	- no changes
eValveType_MA_01_02	1.00	- no changes
eMA100OperatingMedium	1.00	- no changes
eSavingResponse	1.01	- no changes
eTransferResponse	1.03	- added value 31 (#3872358)
stMalfunction	1.01	- added timestamp (#4417523)
stTimeStamp	1.00	- created (#4417523)
stTransferPackage	1.00	- no changes
FB_Read_Bits_From_Byte	1.00	- no changes
FB_Read_Bits_From_Word	1.01	- no changes
FB_Write_Value_To_Word	1.00	- no changes
FB_AppOptionGenerator_MA_02	1.00	- no changes
FB_TeachInRun	1.01	- no changes
FB_ValveControl	2.03	- no changes
FB_StateInterpreter_MA_01	1.01	- no changes
FB_StateInterpreter_MA_02	1.02	- no changes
FB_StateInterpreter_MA_03	1.01	- no changes
FB_StateInterpreter_MA_04	1.01	- no changes

FB_StateInterpreter_MA_05	1.02	- no changes
FB_StateInterpreter_MA_06	1.01	- no changes
FB_StateInterpreter_MA_07	1.01	- no changes
FB_StateInterpreter_MA_08	1.01	- no changes
FB_StateInterpreter_MA_10	1.00	- no changes
FB_StateInterpreter_MA_11	1.02	- no changes
FB_StateInterpreter_MA_12	1.01	- no changes
FB_StateInterpreter_MA_33	1.00	- no changes
FB_StateInterpreter_TeachInRun	1.01	- no changes
FB_Download_Multiple_Parameters	2.01	- no changes
FB_Download_Single_Parameter	2.01	- no changes
FB_ReadMalfunctionList	1.01	- added timestamp information (#4417523)
FB_Save_Configuration_Persistent	2.02	- no changes
FB_Upload_Multiple_Parameters	2.01	- no changes
FB_Upload_Single_Parameter	2.01	- no changes
FB_Read_config_AI_Module	2.00	- created (#3872358)
FB_Read_config_DI_Module	2.00	- created (#3872358)
FB_Read_config_MA_01	2.00	- created (#3872358)
FB_Read_config_MA_02	2.00	- created (#3872358)
FB_Read_config_MA_03	2.00	- created (#3872358)
FB_Read_config_MA_04	2.00	- created (#3872358)
FB_Read_config_MA_05	2.00	- created (#3872358)
FB_Read_config_MA_06	2.00	- created (#3872358)
FB_Read_config_MA_07	2.00	- created (#3872358)
FB_Read_config_MA_08	2.00	- created (#3872358)
FB_Read_config_MA_10	2.00	- created (#3872358)
FB_Read_config_MA_11	2.00	- created (#3872358)
FB_Read_config_MA_12	2.00	- created (#3872358)
FB_Read_config_MA_33	2.00	- created (#3872358)
FB_Write_config_AI_Module	2.02	- no changes
FB_Write_config_DI_Module	2.02	- no changes
FB_Write_config_MA_01	2.00	- no changes
FB_Write_config_MA_02	2.01	- added application parameter iOperatingMode (#4414285)
FB_Write_config_MA_03	2.01	- no changes
FB_Write_config_MA_04	2.00	- no changes
FB_Write_config_MA_05	2.00	- no changes
FB_Write_config_MA_06	2.00	- no changes
FB_Write_config_MA_07	2.00	- no changes
FB_Write_config_MA_08	2.01	- added tuning parameters (#4510540)
FB_Write_config_MA_10	2.01	- no changes
FB_Write_config_MA_11	2.00	- no changes
FB_Write_config_MA_12	2.00	- no changes
FB_Write_config_MA_33	2.01	- no changes

Version 3.4

object	version	change
eRespondingReadMalfunctionList	1.00	- no changes
eResponseTeachInRun	1.06	- added new values 10 and 11 (#3800496)
eValveMode	1.02	- added value 10 (#3873894)
eValveState	1.00	- no changes
eResponseToValveModeSet	1.01	- no changes
eValveType_MA_01_02	1.00	- created (#3920106)
eMA100operatingMedium	1.00	- created (#3873393)
eSavingResponse	1.01	- no changes
eTransferResponse	1.02	- no changes
stMalfunction	1.00	- no changes
stTransferPackage	1.00	- no changes
FB_Read_Bits_From_Byte	1.00	- no changes
FB_Read_Bits_From_Word	1.01	- no changes
FB_Write_Value_To_Word	1.00	- no changes
FB_AppOptionGenerator_MA_02	1.00	- created (#3919901)
FB_TeachInRun	1.01	- integrated new app state return values #10 and #11 (#3800496)
FB_ValveControl	2.03	- added code to reset iErrorCode if ValveState is not „Failure“ or „NotReady“ (#2245880)
FB_StateInterpreter_MA_01	1.01	- no changes
FB_StateInterpreter_MA_02	1.02	- added outputs for indicating meaning of actual values (#3799739)
FB_StateInterpreter_MA_03	1.01	- no changes
FB_StateInterpreter_MA_04	1.01	- no changes
FB_StateInterpreter_MA_05	1.02	- added outputs for indicating meaning of actual values (#3799739)
FB_StateInterpreter_MA_06	1.01	- no changes
FB_StateInterpreter_MA_07	1.01	- no changes
FB_StateInterpreter_MA_08	1.01	- no changes
FB_StateInterpreter_MA_10	1.00	- created (#3805731)
FB_StateInterpreter_MA_11	1.02	- no changes
FB_StateInterpreter_MA_12	1.01	- no changes
FB_StateInterpreter_MA_33	1.00	- no changes
FB_StateInterpreter_TeachInRun	1.01	- added new values 10 and 11 (#3800496)
FB_Download_Multiple_Parameters	2.01	- no changes
FB_Download_Single_Parameter	2.01	- no changes
FB_ReadMalfunctionList	1.00	- no changes
FB_Save_Configuration_Persistent	2.02	- no changes
FB_Upload_Multiple_Parameters	2.01	- no changes
FB_Upload_Single_Parameter	2.01	- no changes
FB_Write_config_AI_Module	2.02	- added new sensor types within comments (#3787601)
FB_Write_config_DI_Module	2.02	- no changes
FB_Write_config_MA_01	2.00	- no changes
FB_Write_config_MA_02	2.00	- no changes

FB_Write_config_MA_03	2.01	- included new application parameters (#3800158)
FB_Write_config_MA_04	2.00	- no changes
FB_Write_config_MA_05	2.00	- no changes
FB_Write_config_MA_06	2.00	- no changes
FB_Write_config_MA_07	2.00	- no changes
FB_Write_config_MA_08	2.00	- no changes
FB_Write_config_MA_10	2.01	- created (#3898591)
FB_Write_config_MA_11	2.00	- no changes
FB_Write_config_MA_12	2.00	- no changes
FB_Write_config_MA_33	2.01	- included new application parameters (#4102822)

Version 3.3

object	version	change
eRespondingReadMalfunctionList	1.00	- no changes
eResponseTeachInRun	1.05	- no changes
eValveMode	1.01	- no changes
eValveState	1.00	- no changes
eResponseToValveModeSet	1.01	- no changes
eSavingResponse	1.01	- no changes
eTransferResponse	1.02	- no changes
stMalfunction	1.00	- no changes
stTransferPackage	1.00	- no changes
FB_Read_Bits_From_Byte	1.00	- no changes
FB_Read_Bits_From_Word	1.01	- no changes
FB_Write_Value_To_Word	1.00	- no changes
eResponseToValveModeSet	1.01	- no changes
FB_TeachInRun	1.00	- no changes
FB_ValveControl	2.02	- no changes
FB_StateInterpreter_MA_01	1.01	- no changes
FB_StateInterpreter_MA_02	1.01	- no changes
FB_StateInterpreter_MA_03	1.01	- no changes
FB_StateInterpreter_MA_04	1.01	- no changes
FB_StateInterpreter_MA_05	1.01	- no changes
FB_StateInterpreter_MA_06	1.01	- no changes
FB_StateInterpreter_MA_07	1.01	- no changes
FB_StateInterpreter_MA_08	1.01	- no changes
FB_StateInterpreter_MA_11	1.02	- no changes
FB_StateInterpreter_MA_12	1.01	- no changes
FB_StateInterpreter_MA_33	1.00	- created (#3625800)
FB_StateInterpreter_TeachInRun	1.00	- no changes
FB_Download_Multiple_Parameters	2.01	- no changes
FB_Download_Single_Parameter	2.01	- no changes
FB_ReadMalfunctionList	1.00	- no changes

FB_Save_Configuration_Persistent	2.02	- no changes
FB_Upload_Multiple_Parameters	2.01	- no changes
FB_Upload_Single_Parameter	2.01	- no changes
FB_Write_config_AI_Module	2.02	- no changes
FB_Write_config_DI_Module	2.02	- no changes
FB_Write_config_MA_01	2.00	- no changes
FB_Write_config_MA_02	2.00	- no changes
FB_Write_config_MA_03	2.00	- no changes
FB_Write_config_MA_04	2.00	- no changes
FB_Write_config_MA_05	2.00	- no changes
FB_Write_config_MA_06	2.00	- no changes
FB_Write_config_MA_07	2.00	- no changes
FB_Write_config_MA_08	2.00	- no changes
FB_Write_config_MA_11	2.01	- no changes
FB_Write_config_MA_12	2.01	- no changes
FB_Write_config_MA_33	1.00	- created (#3625895)

Version 3.2

object	version	change
eRespondingReadMalfunctionList	1.00	- created (#2256314)
eResponseTeachInRun	1.05	- created (#2256460)
eValveMode	1.01	- created
eValveState	1.00	- no changes
eResponseToValveModeSet	1.01	- no changes
eSavingResponse	1.01	- added value 25 „FB_needs_rising_edge_xExecute“ (#2487339)
eTransferResponse	1.02	- added value 8 (#3104465)
stMalfunction	1.00	- created (#2256314)
stTransferPackage	1.00	- no changes
FB_Read_Bits_From_Byte	1.00	- no changes
FB_Read_Bits_From_Word	1.01	- no changes
FB_Write_Value_To_Word	1.00	- no changes
eResponseToValveModeSet	1.01	- no changes
FB_TeachInRun	1.00	- created (#2256460)
FB_ValveControl	2.02	- FB detect a falling edge of xManualAcknowledge while input xEnable is FALSE
FB_StateInterpreter_MA_01	1.01	- no changes
FB_StateInterpreter_MA_02	1.01	- no changes
FB_StateInterpreter_MA_03	1.01	- no changes
FB_StateInterpreter_MA_04	1.01	- no changes
FB_StateInterpreter_MA_05	1.01	- no changes
FB_StateInterpreter_MA_06	1.01	- no changes
FB_StateInterpreter_MA_07	1.01	- no changes
FB_StateInterpreter_MA_08	1.01	- no changes

FB_StateInterpreter_MA_11	1.02	- no changes
FB_StateInterpreter_MA_12	1.01	- no changes
FB_StateInterpreter_TeachInRun	1.00	- created
FB_Download_Multiple_Parameters	2.01	- adjusted range for iChannel from 255 to 31 (#2515542) - validity check shows first wrong element and stops the procedure (#2526252) - adapted functionality for transfer mode on slots without valves (#2957510) - added new enum value (#3104465)
FB_Download_Single_Parameter	2.01	- adjusted range for iChannel from 255 to 31 (#2515542) - validity check shows first wrong element and stops the procedure (#2526252) - adapted functionality for transfer mode on slots without valves (#2957510) - added new enum value (#3104465)
FB_ReadMalfunctionList	1.00	- created
FB_Save_Configuration_Persistent	2.02	- redesign to state machine model - corrected iResponse when calling the FB for the first time with xExecute = FALSE (#2510159) - set output xSavingDone and iResponse when the FB stops sending busdata (#2534862)
FB_Upload_Multiple_Parameters	2.01	- adjusted range for iChannel from 255 to 31 (#2515542) - validity check shows first wrong element and stops the procedure (#2526252) - reintegrated sourcecode (#2521445) - adapted functionality for transfer mode on slots without valves (#2957510) - added new enum value (#3104465)
FB_Upload_Single_Parameter	2.01	- adjusted range for iChannel from 255 to 31 (#2515542) - reintegrated sourcecode (#2521445) - adapted functionality for transfer mode on slots without valves (#2957510) - added new enum value (#3104465)
FB_Write_config_AI_Module	2.02	- corrected iResponse when calling the FB for the first time with xExecute = FALSE (#2510159) - iResponse changes to „FB_needs_rising_Edge_xExecute“ when changing the inputs while the FB is running (#2515568) - iResponse changes always to „invalid_module_number“ when the input iInputModuleNr is wrong(#2515421)
FB_Write_config_DI_Module	2.02	- corrected iResponse when calling the FB for the first time with xExecute = FALSE (#2510159) - iResponse changes to „FB_needs_rising_Edge_xExecute“ when changing the inputs while the FB is running (#2515568) - iResponse changes always to „invalid_module_number“ when the input iInputModuleNr is wrong(#2515421)
FB_Write_config_MA_01	2.00	- no changes
FB_Write_config_MA_02	2.00	- no changes
FB_Write_config_MA_03	2.00	- no changes
FB_Write_config_MA_04	2.00	- no changes
FB_Write_config_MA_05	2.00	- no changes
FB_Write_config_MA_06	2.00	- no changes

FB_Write_config_MA_07	2.00	- no changes
FB_Write_config_MA_08	2.00	- no changes
FB_Write_config_MA_11	2.01	- no changes
FB_Write_config_MA_12	2.01	- no changes

Version 3.1

object	version	change
eValveState	1.00	- no changes
FB_Read_Bits_From_Byte	1.00	- no changes
FB_Read_Bits_From_Word	1.01	- Changes iValue, xError from InOut to Out (#2222801)
FB_Write_Value_To_Word	1.00	- no changes
eResponseToValveModeSet	1.01	- added new Value „FB_not_enabled := 1051“ (#2417793)
FB_ValveControl	2.01	- integrated new Value „FB_not_enabled := 1051“ (#2417793) - updated enable = FALSE behaviour - changed output awBusDataToVTEM to VAR_IN_OUT (#2406661) - added new Output sResponseToValveModeSet (#2475841)
FB_StateInterpreter_MA_01	1.01	- no changes
FB_StateInterpreter_MA_02	1.01	- no changes
FB_StateInterpreter_MA_03	1.01	- no changes
FB_StateInterpreter_MA_04	1.01	- no changes
FB_StateInterpreter_MA_05	1.01	- no changes
FB_StateInterpreter_MA_06	1.01	- no changes
FB_StateInterpreter_MA_07	1.01	- no changes
FB_StateInterpreter_MA_08	1.01	- no changes
FB_StateInterpreter_MA_11	1.02	- added outputs xPistonDetectRet and xPistonDetectAdv (#2248584)
FB_StateInterpreter_MA_12	1.01	- no changes
eSavingResponse	1.00	- no changes
eTransferResponse	1.01	- added items 27, 28, 29, 30
stTransferPackage	1.00	- no changes
FB_Write_config_MA_01	2.00	- created (#2256987)
FB_Write_config_MA_02	2.00	- added parameters for the sensor port mapping (#2256299) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Write_config_MA_03	2.00	- added parameters for the sensor port mapping (#2256299) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Write_config_MA_04	2.00	- added parameters for the sensor port mapping (#2256299) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Write_config_MA_05	2.00	- added parameters for the sensor port mapping (#2256299) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Write_config_MA_06	2.00	- created (#2408095)
FB_Write_config_MA_07	2.00	- added parameters for moment of inertia used for swivel drives (#2307857)

		- added parameters for the sensor port mapping (#2256299) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Write_config_MA_08	2.00	- added parameters for moment of inertia used for swivel drives (#2307857) - added parameters for the sensor port mapping (#2256299) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Write_config_MA_11	2.00	- added parameters for the sensor port mapping (#2256299) - removed tuning parameters (#2408215) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Write_config_MA_12	2.00	- added parameters for moment of inertia used for swivel drives (#2307857) - changed awBusDataToVTEM into VAR_IN_OUT (#2406661)
FB_Download_Multiple_Parameters	2.00	- changed awBusDataToVTEM into VAR_IN_OUT (#2406661) - fixed bug that caused „exit transfer mode“ even if xExecute is FALSE (#2246210) - added value 7 to list of invalid commands (#2318899)
FB_Download_Single_Parameter	2.00	- changed awBusDataToVTEM into VAR_IN_OUT (#2406661) - fixed bug that caused „exit transfer mode“ even if xExecute is FALSE (#2246210) - added value 7 to list of invalid commands (#2406621)
FB_Save_Configuration_Persistent	2.01	- changed awBusDataToVTEM into VAR_IN_OUT (#2406661) - fixed bug that caused „exit transfer mode“ even if xExecute is FALSE (#2458152)
FB_Upload_Multiple_Parameters	2.00	- changed awBusDataToVTEM into VAR_IN_OUT (#2406661) - fixed bug that caused „exit transfer mode“ even if xExecute is FALSE (#2246210) - added value 7 to list of invalid commands (#2406621) - fixed a bug regarding the stability of the function block (#2262058)
FB_Upload_Single_Parameter	2.00	- changed awBusDataToVTEM into VAR_IN_OUT (#2406661) - fixed bug that caused „exit transfer mode“ even if xExecute is FALSE (#2246210) - added value 7 to list of invalid commands (#2406621)
FB_Write_config_AI_Module	2.01	- created
FB_Write_config_DI_Module	2.01	- created

Version 2.0

-- First official release --

=====

Copyright notice

This documentation is the intellectual property of Festo SE & Co. KG, who also holds exclusive copyrights. Any modification of the content, duplication or reprinting of this documentation as well as distribution to third parties can only be made with the express consent of Festo SE & Co. KG.

Festo SE & Co. KG reserves the right to make modifications to this document in whole or in part. All brand and product names are trademarks or registered trademarks of their respective owners.

Legal notice

Hardware, software, operating systems and drivers may only be used for the applications described and only in conjunction with components recommended by Festo SE & Co. KG.

Festo SE & Co. KG does not accept any liability for damages arising from the use of any incorrect or incomplete information contained in this documentation or any information missing therefrom.

Defects resulting from the improper treatment of devices and modules are excluded from the warranty.

The data and information specified in this document should not be used for the implementation of safety functions related to the protection of personnel and machinery.

No liability is accepted for claims for consequential damages arising from failure or malfunction. Otherwise, the clauses concerning liability included in the terms and conditions of delivery, payment and use of software of Festo SE & Co. KG shall apply, which can be found at www.festo.com or provided upon request.

None of the data contained in this document represent guaranteed features in the legal sense, in particular with regard to functionality, condition or quality.

The respective operating instructions for products from Festo can be found at www.festo.com/sp.

Users of this document (function and application) must themselves verify that all functions described herein also work correctly in their own applications. Even after examining this document and while using the specifications contained herein, users nevertheless remain solely responsible for their own applications.

Table of contents

1	Important information	16
1.1	Intended use	16
1.1.1	Target group	16
1.1.2	Service	16
1.2	Safety instructions	16
1.3	Marking special information.....	16
1.3.1	Pictograms	16
1.3.2	Text symbols and markers	16
1.3.3	Further conventions	17
2	Introduction	18
2.1	Fundamentals.....	18
2.1.1	Structure of the library	19
2.1.1.1	Data types	19
2.2	Requirements.....	20
2.2.1	Connection of the CPX terminal to the controller	20
2.2.2	Integrating the library.....	20
2.3	General information on Motion Terminal VTEM.....	21
2.3.1	Documentation for Motion Terminal VTEM	21
2.4	Communication protocol of the Motion Terminal VTEM	21
2.5	Connecting I/O Data to the function blocks.....	22
3	FB_ValveControl – Control of a valve by operating a Motion App	24
3.1	Functional description	24
3.2	Inputs and outputs.....	24
3.2.1	Data type eResponseToValveModeSet	26
3.2.2	Data type eValveType_MA_01_02	27
4	FB_AppOptionGenerator_MA_02 – Generation of the app option value for MA#02	28
4.1	Functional description.....	28
4.2	Inputs and outputs.....	28
5	FB_StateInterpreter_... – Interpretation of the AppState.....	29
5.1	Functional description.....	29
5.2	Inputs and outputs.....	29
5.2.1	Motion App #01 (Directional control valve functions).....	30
5.2.2	Motion App #02 (Proportional directional control valve)	30
5.2.3	Motion App #05 (Supply and exhaust air flow control)	31
5.2.4	Motion App #10 (Flow control)	31
5.2.5	Motion App #11 (Soft Stop).....	31
5.2.6	Motion App #12 (Leakage diagnostics)	32
5.2.7	Motion App #33 (Positioning).....	32
5.2.8	Teach-in run.....	32
6	FB_Read_config_MA_... – Reading parameterisation of a Motion App	33
6.1	Functional description.....	33
6.2	Inputs and outputs	33
6.2.1	Data type eTransferResponse.....	34
7	FB_Write_config_MA_... – Writing parameterisation of a Motion App	35
7.1	Functional description	35
7.2	Inputs and outputs.....	35
7.2.1	Example of assignment of inputs.....	36

8	FB_Read_config_DI/AI_Module – Reading parameterisation of the input modules	38
8.1	Functional description	38
8.2	Inputs and outputs	39
9	FB_Write_config_DI/AI_Module – Writing parameterisation of the input modules	41
9.1	Functional description	41
9.2	Inputs and outputs	42
10	FB_ReadMalfunctionList – Read out malfunction list	44
10.1	Functional description	44
10.2	Inputs and outputs	44
10.2.1	Data type eResponseReadMalfunctionList	45
10.2.2	Data type stMalfunction	46
10.2.3	Data type stTimeStamp	46
11	FB_TeachInRun – Parameterise and execute teach-in run	47
11.1	Functional description	47
11.2	Inputs and outputs	47
11.2.1	Data type eResponseTeachInRun	48
12	FB_Save_Configuration_Persistent – Save parameterisation persistent.....	50
12.1	Functional description	50
12.2	Inputs and outputs	50
12.2.1	Data type eSavingResponse	50
13	FB_Download_Single_Parameter – Transfer single parameters to the Motion Terminal.....	52
13.1	Functional description	52
13.2	Inputs and outputs	52
14	FB_Download_Multiple_Parameters – Transfer multiple parameters to the Motion Terminal	53
14.1	Functional description	53
14.2	Inputs and outputs	53
14.2.1	Data type stTransferPackage	54
15	FB_Upload_Single_Parameter – Reading single value from the Motion Terminal	55
15.1	Functional description	55
15.2	Inputs and outputs	55
16	FB_Upload_Multiple_Parameters – Reading multiple values from the Motion Terminal	56
16.1	Functional description	56
16.2	Inputs and outputs	56
17	Auxiliary function blocks	57
17.1	FB_Read_Bits_From_Byte – Read bit area from byte	57
17.1.1	Functional description	57
17.1.2	Inputs and outputs	57
17.2	FB_Read_Bits_From_Word – Read bit area from word	58
17.2.1	Functional description	58
17.2.2	Inputs and outputs	58
17.3	FB_Write_Value_To_Word – Write value within a word	58
17.3.1	Functional description	58
17.3.2	Inputs and outputs	58
A	Technical appendix	59
A.1	Technical data	59
A.1.1	General.....	59
B	Index.....	60

1 Important information

1.1 Intended use

The function blocks (FBs) described in this library are used to control and parameterise the Motion Terminal VTEM from Festo.

By means of the function blocks, the various functions of this device can be comfortably incorporated into the PLC program. No additional functions are made available by the function blocks, but the operation of the functions implemented on the device is made much easier.

The function blocks are called up cyclically using a separate instance for each valve of the Motion Terminal. Simultaneous design of several function blocks for controlling the same valve is not permitted. It is therefore recommended to bring about the implementation via a status machine, to prevent the status of simultaneous access.



Unsafe status of the system at parallel call-up of function blocks

The simultaneous access of several function blocks to the output data of a valve (wBusDataToVTEM...) can lead to the unforeseeable behaviour of the Motion Terminal, as well as the connected periphery.

- Ensure that never more than one function block instance can gain access to the output data of a valve at the same time (wBusDataToVTEM...).

To enable an adjustment of function blocks by the user, the function blocks are made available unprotected in the source code. Changes to the source text of the function blocks should only be made by expert personnel and can have an effect on the support services.

Observe the “Safety instructions” and instructions on the intended use of the respective devices, components and modules. When connected with commercially available components, such as sensors and actuators, the specified critical limits for pressures, temperatures, electrical data, torques etc. must be observed.

1.1.1 Target group

This description is intended exclusively for technicians trained in control and automation technology, who have experience in installing, commissioning, programming and diagnosing positioning systems and the relevant fieldbuses.

1.1.2 Service

Please consult your local Festo service or the central service department via the following e-mail address if you have any technical problems:

service_international@festo.com

1.2 Safety instructions

When commissioning and programming pneumatic systems, you must observe the safety regulations in the descriptions and operating instructions for the components used. The user must ensure that nobody has access to the sphere of influence of the connected actuators. Access to the potential danger area must be prevented by suitable measures such as barriers and warning signs.

1.3 Marking special information

1.3.1 Pictograms

The following pictograms mark passages in the text containing special information:



Information

Recommendations, tips and references to other sources of information

1.3.2 Text symbols and markers

1. Figures denote activities that must be carried out in the order specified.
- Bullet points denote activities that can be carried out in any order.
 - The dash heads indicate general lists.

1.3.3 Further conventions

“FBs”	In some places the abbreviation “FB” is used for “function block” (also “FBs”)
“OK”	Names of windows, dialogs and buttons such as “Message Window”, “Dearchive Project”, “OK” as well as designations are shown in inverted commas.
CTRL	Names of keys on the PC keyboard are represented in the text with uppercase letters (e.g. ENTER KEY, CTRL, C, F1 etc.).
CTRL+C	For some functions, two keys must be pressed at the same time. Example: The CTRL key together with the “C” key produces the CTRL+C character.
	Where it says “click” or “double-click”, this always refers to the left mouse button. If the right-hand mouse button is to be used, this will be explicitly mentioned.

2 Introduction

2.1 Fundamentals

The library Festo_VTEM_DevCon is an internal library for Siemens TIA Portal from version 19 onwards.

The function blocks of the library support the user in the parameterisation and operation of Motion Apps on the Motion Terminal VTEM from Festo. Here the Motion Terminal VTEM can be controlled and parameterised via a suitable fieldbus interface of the CPX terminal.



Information

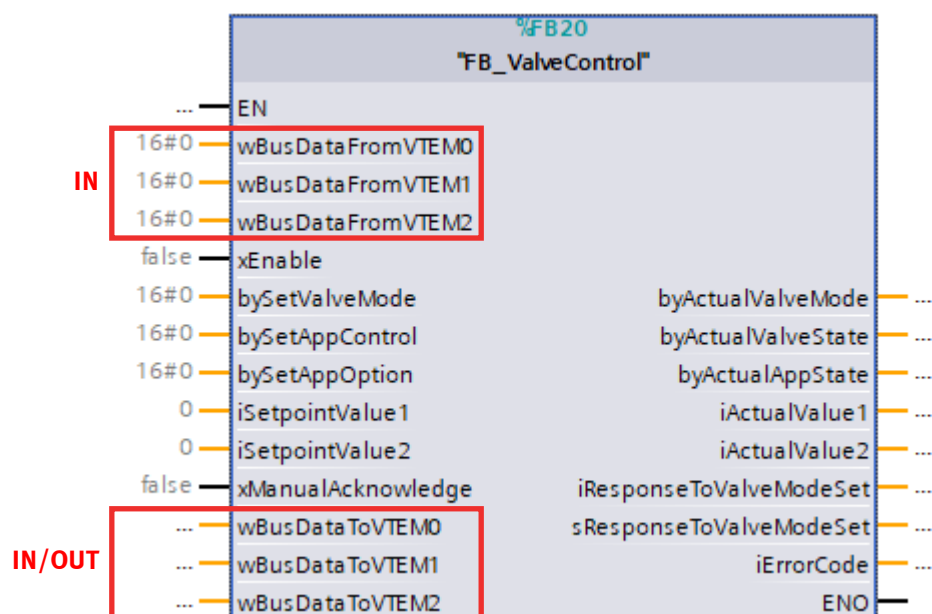
The function blocks represent a simplified operation of the protocol which is described in the user documentation of the Motion Terminal VTEM. All functions of the function blocks described here can also be implemented with the corresponding effort in the user program directly via the cyclic process data of the Motion Terminal.

Detailed information about the function, operation and parameterisation is available in the user documentation of the Motion Terminal VTEM (→ [Documentation for Motion Terminal VTEM](#)).

Architecture

The Motion Terminal VTEM is controlled on the valve level via the Motion Apps and with the aid of the function blocks. This means that the function blocks control or parameterise one Motion App respectively on one valve. The function blocks are connected to the valves via the input and output data of the CPX terminal.

All control and parameterisation function blocks each have three inputs and outputs for allocating the I/O data of the valve:



- wBusDataFromVTEM...: Input data of the valve (3 words)
- wBusDataToVTEM...: Output data of the valve (3 words)



Unsafe status of the system at parallel call-up of function blocks

The simultaneous access of several function blocks to the output data of a valve (wBusDataToVTEM...) can lead to the unforeseeable behaviour of the Motion Terminal, as well as the connected periphery.

- Ensure that never more than one function block instance can gain access to the output data of a valve at the same time (wBusDataToVTEM...).



It is recommended to use a separate instance of the function blocks for each valve.

2.1.1 Structure of the library

The library comprises different data types and function blocks with different functions and contents. Some data types are used within the function blocks as a list of numeric values, so that these values can be addressed by an identifier name (“enumeration”).

Example for using the data types `eValveState` and `eResponseToValveModeSet`

```
IF #byActualValveState = INT_TO_BYTE(#eValveState.Failure) THEN
    #iResponseToValveModeSet := #eResponseToValveModeSet.failure;
```

Function blocks

The function blocks are described in detail in the following chapters.

2.1.1.1 Data types

`eValveMode`

The data type is used as enumeration and contains the descriptions of the individual valve operating modes. The integer values between 1 and 59 correspond to the respective Motion App ID.

`eValveState`

The data type is used as enumeration and contains the possible states of the valve. The integer values correspond to the decimal value of the corresponding bits in the process data.

`eResponseToValveModeSet`

The data type is used as enumeration and contains feedback signals to the values currently in the process output data. The integer values correspond to the decimal values in area “App State”, extended by the additional feedback signals of function block “FB_ValveControl”.

`eValveType_MA_01_02`

The data type is used as enumeration and contains the values for the valve types of the Motion Apps #01 and #02, which are transferred via the “App Option” area in the process output data.

`eMA10OperatingMedium`

The data type is used as enumeration and contains the values for the “Operating Medium” parameter for Motion App #10 (flow control).

`eSavingResponse`

The data type is used as enumeration and contains feedback signals for command “Persistent saving”, which can also be started via function block “FB_Save_Configuration_Persistent”.

`eTransferResponse`

The data type is used as enumeration and contains feedback signals about started transfer mode commands and is used by the function blocks to control the transfer mode.

`eResponseReadMalfunctionList`

The data type is used as enumeration and contains the responses used by the function block “FB_ReadMalfunctionList”.

`eResponseTeachInRun`

The data type is used as enumeration and contains the response of the function block “FB_TeachInRun” as well as the response of the teach-in run itself.

`stTransferPackage`

The data type is used to take up parameterisations into a list, which can be transferred with function blocks “FB_upload_Multiple_Parameters” and “FB_download_Multiple_Parameters”.

`stMalfunction`

The data type maps the layout of an entry in the malfunction list and contains the malfunction code, the malfunction subcode and the classification of the malfunction.

2.2 Requirements

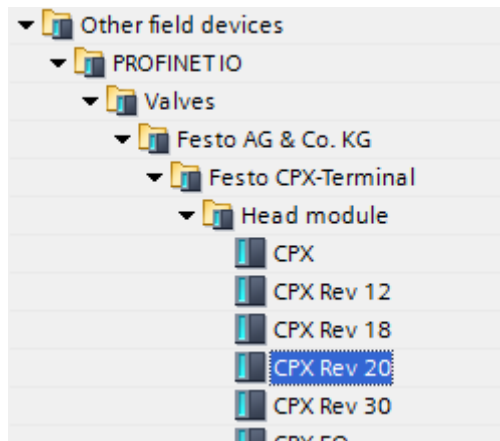
2.2.1 Connection of the CPX terminal to the controller



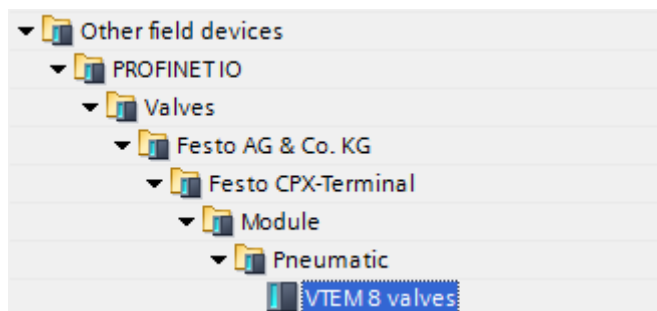
Device description file

To use the Motion Terminal VTEM, an up-to-date device description file of the CPX system for PROFINET (bus node CPX-FB33) must be integrated into the control project.

The device description file is available at the Festo Support Portal (→ www.festo.com/sp).

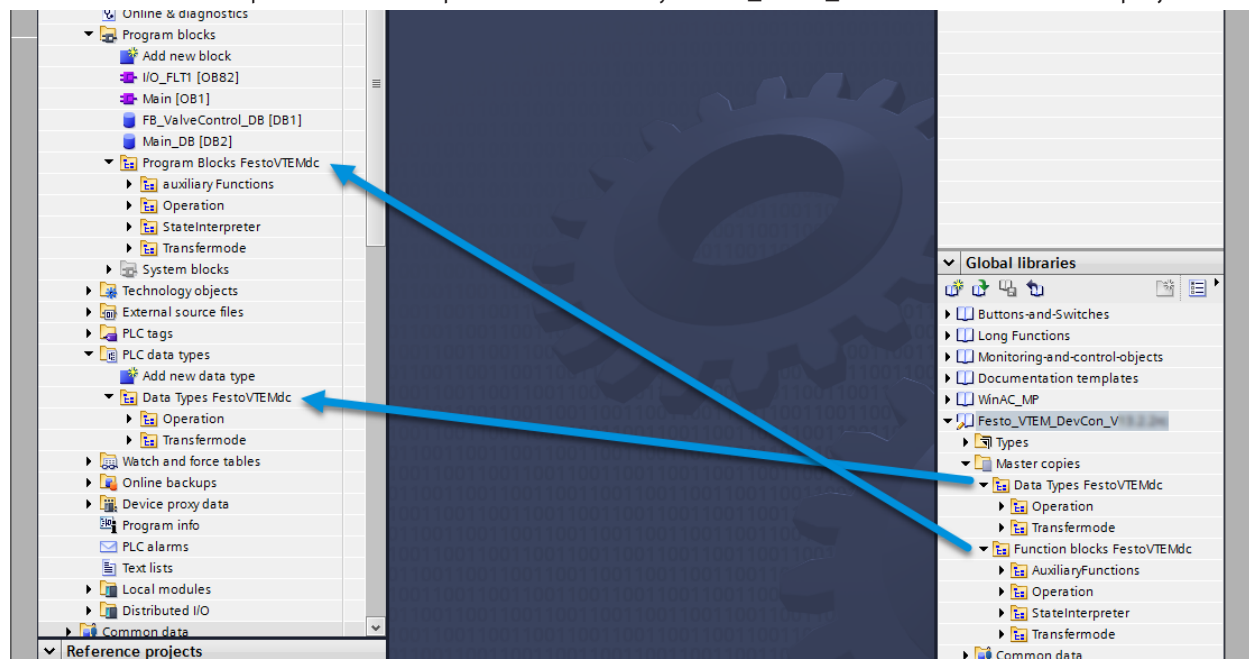


The Motion Terminal VTEM is integrated as a submodule of the CPX terminal:



2.2.2 Integrating the library

All function blocks required must be copied from the library “Festo_VTEM_DevCon” into the control project.



2.3 General information on Motion Terminal VTEM

For commissioning and handling of the Motion Terminal, a safe understanding of the function of the product is absolutely essential.

The following additional descriptions are required for a complete understanding:

2.3.1 Documentation for Motion Terminal VTEM

Documentation	Contents
Quick Start Guide	Examples of the description of the WebConfig interface
Quick Reference PLC programming	Compact overview of all functions and parameters of the Motion Terminal that can be used via PLC.
“Instructions for using Motion Terminal VTEM” (VTEM)	Instructions on safe use and important notes on handling, installation, commissioning and maintenance of the Motion Terminal VTEM.
“Description of Motion Terminal VTEM, function and parameterisation” (VTEM-BES-...)	Detailed description of the Motion Terminal VTEM as well as its function and parameterisation
“Motion Terminal VTEM description, Motion App ...” (VTEM-MA-... / GAMM-A...-...)	Detailed description of the Motion Apps for the Festo Motion Terminal VTEM
“CPX system description” (CPX-SYS-...)	Information on the complete system of the CPX terminal

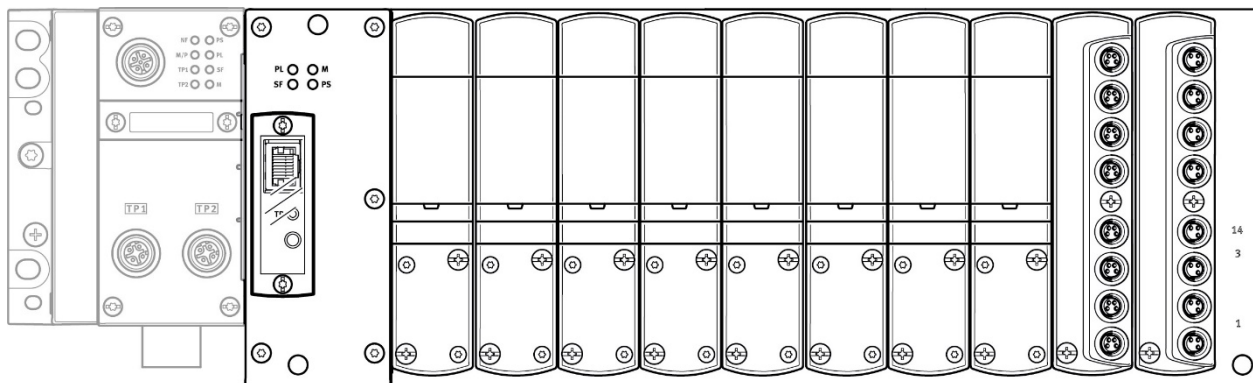


All available documents for the product → www.festo.com/pk

Always observe the information and safety instructions in the documentation!

2.4 Communication protocol of the Motion Terminal VTEM

The communication between the higher-order controller (PLC) and the Motion Terminal controller is based on input and output data of in each case 8×6 bytes from the CPX terminal. Each of the maximum 8 valve slots on a Motion Terminal is assigned 6 bytes of input data (Process Data Input, PDI) and 6 bytes of output data (Process Data Output, PDO), regardless of the actual number of valves present.



Valve on slot 0	Valve on slot 1	Valve on slot 2	Valve on slot 3	Valve on slot 4	Valve on slot 5	Valve on slot 6	Valve on slot 7
6 bytes PDI	6 bytes PDI	6 Bytes PDI	6 Bytes PDI	6 bytes PDI	6 bytes PDI	6 bytes PDI	6 bytes PDI
6 bytes PDO	6 bytes PDO	6 bytes PDO	6 bytes PDO	6 bytes PDO	6 bytes PDO	6 bytes PDO	6 bytes PDO



The exact structure of the input and output data is described in detail in “Motion Terminal VTEM function and parameterisation”.

The information in this description is assumed as being known.

2.5 Connecting I/O Data to the function blocks

In the program, which use the function blocks of this library, the 6 bytes of input data and 6 bytes of output data must be maintained per valve as 3 variables of type WORD.

It is also recommended to create a separate instance of the function blocks used for each valve.



The specified start address for the variables of the input and output data must match the valve with which communication is to take place (→ [Communication protocol of the Motion Terminal VTEM](#)). Modules of the CPX terminal must be taken into account.

The examples below refer to a Motion Terminal with a bus node CPX-FB33 as master in the CPX terminal.

Device overview								
Module	...	Rack	Slot	I address	Q address	Type	Article number	Firmware
fb33atvtem1of8		0	0	2042*		CPX Rev 20	TN 197330	V3.2.24
PN-IO Interface		0	0 X1	2041*		CPX		
VTEM 8 Valves		0	2	2...49	2...49	VTEM 8 valves	TN 8047502	

The input and output variables for the function blocks can be created in the standard variables table:

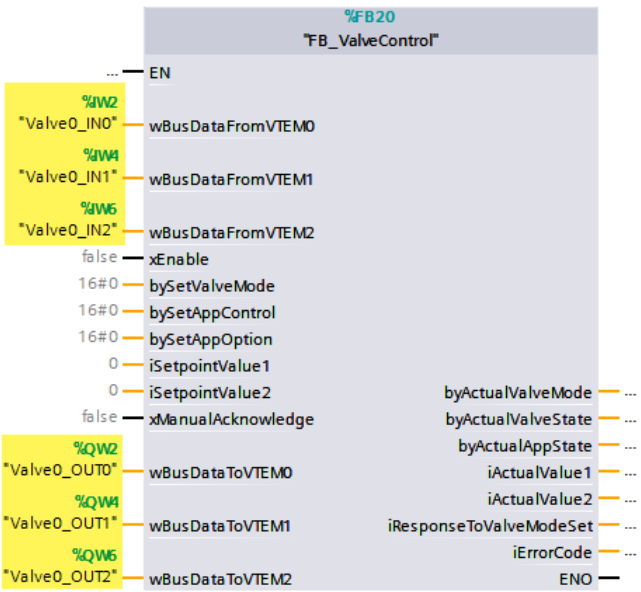
Standard tag table							
	Name	Data type	Address	Retain	Visible in HMI	Accessible from ...	Comment
1	Valve0_IN0	Word	%IW2		☐	☐	Data from Valve 0
2	Valve0_IN1	Word	%IW4		☐	☐	Data from Valve 0
3	Valve0_IN2	Word	%IW6		☐	☐	Data from Valve 0
4	Valve0_OUT0	Word	%QW2		☐	☐	Data to Valve 0
5	Valve0_OUT1	Word	%QW4		☐	☐	Data to Valve 0
6	Valve0_OUT2	Word	%QW6		☐	☐	Data to Valve 0
7	Valve1_IN1	Word	%IW8		☐	☐	Data from Valve 1
8	Valve1_IN2	Word	%IW10		☐	☐	Data from Valve 1
9	Valve1_IN3	Word	%IW12		☐	☐	Data from Valve 1
10	Valve1_OUT1	Word	%QW8		☐	☐	Data to Valve 1
11	Valve1_OUT2	Word	%QW10		☐	☐	Data to Valve 1
12	Valve1_OUT3	Word	%QW12		☐	☐	Data to Valve 1
13	Valve2_IN1	Word	%IW14		☐	☐	Data from Valve 2
14	Valve2_IN2	Word	%IW16		☐	☐	Data from Valve 2
15	Valve2_IN3	Word	%IW18		☐	☐	Data from Valve 2
40	Valve6_OUT1	Word	%QW38		☐	☐	Data to Valve 6
41	Valve6_OUT2	Word	%QW40		☐	☐	Data to Valve 6
42	Valve6_OUT3	Word	%QW42		☐	☐	Data to Valve 6
43	Valve7_IN1	Word	%IW44		☐	☐	Data from Valve 7
44	Valve7_IN2	Word	%IW46		☐	☐	Data from Valve 7
45	Valve7_IN3	Word	%IW48		☐	☐	Data from Valve 7
46	Valve7_OUT1	Word	%QW44		☐	☐	Data to Valve 7
47	Valve7_OUT2	Word	%QW46		☐	☐	Data to Valve 7
48	Valve7_OUT3	Word	%QW48		☐	☐	Data to Valve 7

When calling up a function block, the variables are transferred along with the input data to the function block inputs “wBusDataFromVTEM...”. The data at function block in-output “wBusDataToVTEM...” are assigned to the variables with the output data.

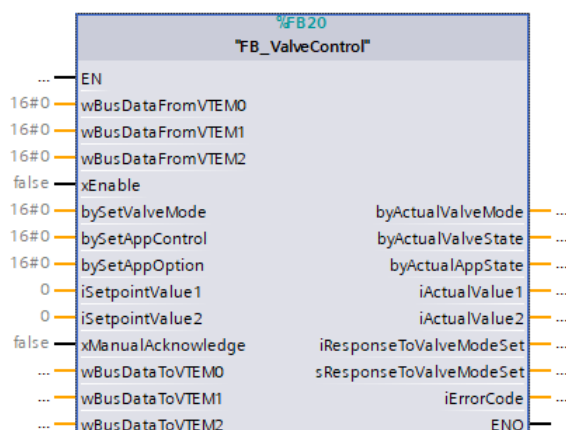
Example in SCL

```
"FB_ValveControl_DB"(wBusDataFromVTEM0:="Valve0_IN0",
wBusDataFromVTEM1:="Valve0_IN1",
wBusDataFromVTEM2:="Valve0_IN2",
xEnable:=,
bySetValveMode:=,
bySetAppControl:=,
bySetAppOption:=,
iSetpointValue1:=,
iSetpointValue2:=,
xManualAcknowledge:=,
byActualValveMode=>,
byActualValveState=>,
byActualAppState=>,
iActualValue1=>,
iActualValue2=>,
iResponseToValveModeSet=>,
iErrorCode=>,
wBusDataToVTEM0:="Valve0_OUT0",
wBusDataToVTEM1:="Valve0_OUT1",
wBusDataToVTEM2:="Valve0_OUT2");
```

Example in FBD

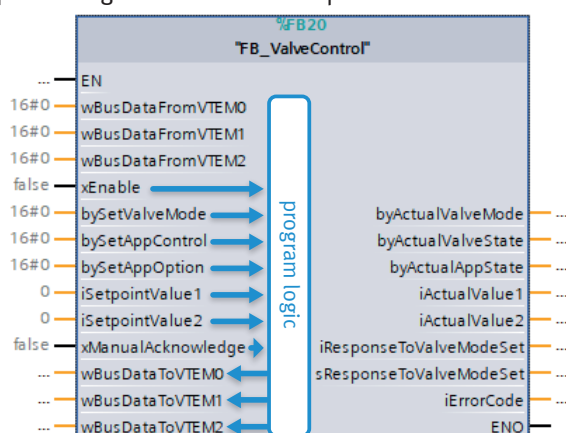


3 FB_ValveControl – Control of a valve by operating a Motion App

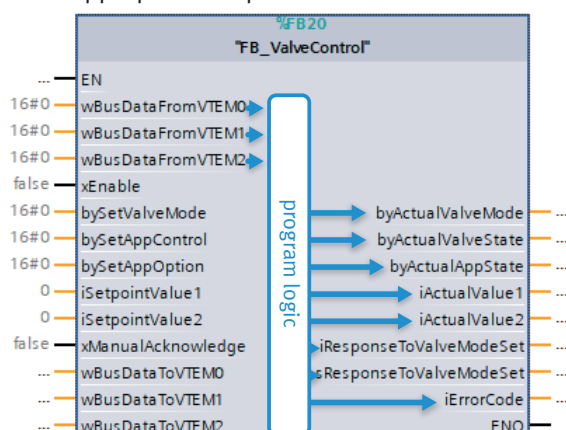


3.1 Functional description

The function block FB_ValveControl serves to operate and control the Motion Apps. For this the function block interprets the values at the inputs and generates the in-output data that is sent to the Motion Terminal.



In parallel, the function block interprets the input data of a valve received from the Motion Terminal, classifies the information in it and issues it to the appropriate outputs.



This means that the status of the function block outputs takes place at least one cycle after the change of the input values.

3.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (➔ Connecting I/O Data to the Add-On instructions).

Designation	Variable type	Data type	Description
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xEnable	VAR_INPUT	BOOL	Input must be TRUE to be able to use the function block. At a change to FALSE, the function block stops any operation running on the valve (Valve Mode = “valve inactive”). The output iResponseToValveModeSet shows the value 1051 = “FB_not_enabled” after the function block was stopped successfully.
bySetValveMode	VAR_INPUT	BYTE	Input for setting the valve mode. Permissible values are: 1 ... 59 (ID Motion App) 60 (Teach-in run) 61 (End Motion App) 62 (Acknowledge inactive error), alternatively use input “xManualAcknowledge”
bySetAppControl	VAR_INPUT	BYTE	Input for setting the Motion App control. The meaning is specific to the Motion App.
bySetAppOption	VAR_INPUT	BYTE	Input for setting the Motion App setting. The meaning is specific to the Motion App. For Motion Apps #01 and #02, the data type eValveType_MA_01_02 can be used. For Motion App #02 there is a function block for composing the App Option (FB_AppOptionGenerator_MA_02)
iSetPointValue1	VAR_INPUT	INT	Input for setpoint value 1: The meaning is specific to the Motion App.
iSetPointValue2	VAR_INPUT	INT	Input for setpoint value 2: The meaning is specific to the Motion App.
xManualAcknowledge	VAR_INPUT	BOOL	Inactive errors in valve status “not ready” can be acknowledged with a rising edge at this input (corresponds to specification “bySetValveMode := 62” → Valve status becomes “configurable”).
byActualValveMode	VAR_OUTPUT	BYTE	Returned mode of the valve
byActualValveState	VAR_OUTPUT	BYTE	Returned status of the valve
iActualAppState	VAR_OUTPUT	INT	Returned status of the Motion App
iActualValue1	VAR_OUTPUT	INT	Returned actual value 1
iActualValue2	VAR_OUTPUT	INT	Returned actual value 2
iResponseToValveModeSet	VAR_OUTPUT	INT	Response to the sent ValveMode (Input bySetValveMode). Meaning: → Data type eResponseToValveModeSet .
sResponseToValveModeSet	VAR_OUTPUT	STRING	Corresponding text to value at iResponseToValveModeSet
iErrorCode	VAR_OUTPUT	INT	In case of an error (valve state = “failure” or “not ready”), the error code of the last error is at this output.

The use of the inputs depends on the Motion App currently running.

3.2.1 Data type eResponseToValveModeSet

The data type is used as an enumeration and serves for the output of feedbacks to the ValveMode sent (Value at input “SetValveMode”). A part of the stored return values correspond to the value of output “AppState” in case of faulty output data (wBusDataToVTEM...). There are also function-block-specific return values (in grey) that cannot be imaged via the output “AppState” (e.g. via the status of the acknowledge function).

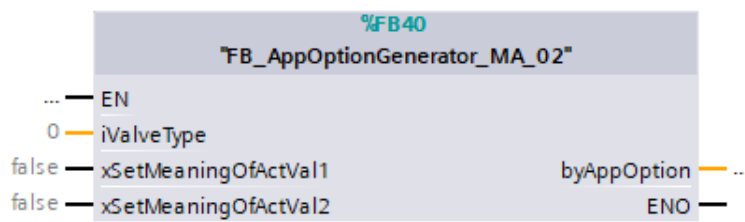
Meaning	Data type	Return value
invalid_SetValveMode	INT	1
invalid_SetAppControl	INT	2
invalid_SetAppOption	INT	3
invalid_SetPointValue1	INT	4
invalid_SetPointValue2	INT	5
invalid_configuration	INT	6
access_denied_WebConfig	INT	10
access_denied_saving_parmetrisation	INT	11
no_license_for_this_MA	INT	12
all_licenses_in_use	INT	13
invalid_license_file	INT	14
no_valve_detected	INT	15
valve_self_calibration_running	INT	16
no_TransferMode_with_this_FB	INT	1020
preparing_for_start	INT	1031
starting	INT	1032
MA_running	INT	1033
Transfermode_running	INT	1034
stopping	INT	1035
all_MAs_stopped	INT	1036
acknowledge_needed	INT	1037
failure	INT	1038
preparing_for_acknowledge	INT	1040
acknowledging	INT	1041
all_errors_acknowledged	INT	1042
failure_still_active	INT	1043
no_Communication	INT	1050
FB_not_enabled	INT	1051

3.2.2 Data type eValveType_MA_01_02

The data type is used as an enumeration and contains the values for the valve types that can be selected for Motion App #01 or Motion App #02.

Meaning		Data type	Return value
Valve_4_3_G	4/3 G (normally closed)	INT	1
Valve_4_3_B	4/3 B (normally open)	INT	2
Valve_4_3_E	4/3 E (normally exhausted)	INT	3
Valve_2x3_2_O	2 × 3/2 O (normally open)	INT	4
Valve_2x3_2_G	2 × 3/2 G (normally closed)	INT	5
Valve_3_2_O_and_3_2_G	3/2 O + 3/2 G	INT	6
Valve_4_2_bistable	4/2 bistable	INT	7
Valve_2x2_2_G	2 × 2/2 G (normally closed)	INT	8
Valve_4_2_Mono	4/2 mono	INT	9
Valve_4_3_G_MA02	4/3 G (normally closed)	INT	14
Valve_2x3_3_G_MA02	2 × 3/3 G (normally closed)	INT	15

4 FB_AppOptionGenerator_MA_02 – Generation of the app option value for MA#02



4.1 Functional description

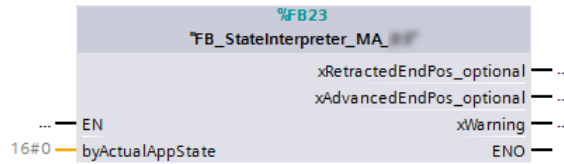
The function block FB_AppOptionGenerator_MA_02 takes over the generation of the app option value for the Motion App #02 (directional control valve function), which in the case of this app contains several pieces of information. The corresponding information can be applied to one input of the function block in each case. The function block then generates the appropriate app option value at the output.

4.2 Inputs and outputs

The following table contains the variables of the output and inputs of the function block.

Designation	Variable type	Data type	Description
iValveType	VAR_INPUT	INT	Value for the valve type to be simulated by the Motion App (→).
xSetMeaningOfActVal1	VAR_INPUT	BOOL	Setting for the meaning of actual value 1 0: Actual value 1 shows valve position 1: Actual value 1 shows pressure (relative)
xSetMeaningOfActVal2	VAR_INPUT	BOOL	Setting for the meaning of actual value 2 0: Actual value 2 shows valve position 1: Actual value 2 shows pressure (relative)
byAppOption	VAR_OUTPUT	BYTE	Byte for passing on to FB_ValveControl

5 FB_StateInterpreter_... – Interpretation of the AppState

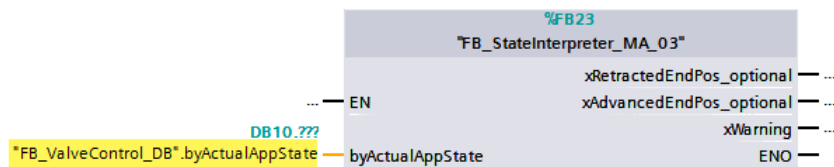


5.1 Functional description

The State Interpreter function blocks make available the information for the state of the running Motion App or the teach-in run from the input data (Byte 1) at separate outputs. For this, the input “byActualAppState” is directly connected with output “byActualAppState” of function block “FB_ValveControl”.

Example

```
"FB_StateInterpreter_MA_03_DB_1" (byActualAppState:="FB_ValveControl_DB".byActualAppState,
xRetractedEndPos_optional=>,
xAdvancedEndPos_optional=>,
xWarning=>);
```



The State Interpreter function blocks may only be used while the corresponding Motion App resp. the teach-in run is running on the relevant valve (the status of the valve (Valve State) = 2 (“running”). Otherwise, the values at the outputs are not correct.

5.2 Inputs and outputs

The State Interpreter function blocks are all structured according to the same principle (Division of the App-State-Byte into the individual information), the outputs, however, are Motion-App-specific. Outputs “xRetractedEndPos_optional”, “xAdvancedEndPos_optional” and “xWarning” are available for most function blocks.

Designation	Variable type	Data type	Description
byActualAppState	VAR_INPUT	BYTE	Input for area “App State” in the input data (Byte 1), which are received by the Motion Terminal. Function block “FB_ValveControl” outputs this area to a separate output (byActualValveState).
xRetractedEndPos_optional	VAR_OUTPUT	BOOL	The output corresponds to the state at the optionally assigned input of the digital input module for the retracted end position (➔ Description of Motion Terminal VTEM, function and parameterisation). The assignment is made with the aid of the parameterisation function blocks of the Motion Apps (➔ Example of assignment of inputs).
xAdvancedEndPos_optional	VAR_OUTPUT	BOOL	The output corresponds to the state at the optionally assigned input of the digital input module for the advanced end position (➔ Description of Motion Terminal VTEM, function and parameterisation). The assignment is made with the aid of the parameterisation function blocks of the Motion Apps (➔ Example of assignment of inputs).

Designation	Variable type	Data type	Description
xRetractedEndPos	VAR_OUTPUT	BOOL	The output corresponds to the state at the assigned input of the digital input module for the retracted end position (→ Description of Motion Terminal VTEM, function and parameterisation). The assignment is made with the aid of the parameterisation function blocks of the Motion Apps (→ Example of assignment of inputs).
xAdvancedEndPos	VAR_OUTPUT	BOOL	The output corresponds to the state at the assigned input of the digital input module for the advanced end position (→ Description of Motion Terminal VTEM, function and parameterisation). The assignment is made with the aid of the parameterisation function blocks of the Motion Apps (→ Example of assignment of inputs).
xWarning	VAR_OUTPUT	BOOL	The output is TRUE as long as the running Motion App issues a warning.

The following sections describe, if available, the Motion App-specific output variables of the corresponding function blocks:

5.2.1 Motion App #01 (Directional control valve functions)

Designation	Variable type	Data type	Description
bySwitchingPositionAt2	VAR_OUTPUT	BYTE	Switching position at the air supply port (2): 0: Closed 1: Pressurised 2: Exhausted 3: Error
bySwitchingPositionAt4	VAR_OUTPUT	BYTE	Switching position at the air supply port (4): 0: Closed 1: Pressurised 2: Exhausted 3: Error

5.2.2 Motion App #02 (Proportional directional control valve)

Designation	Variable type	Data type	Description
xMeaningOfActVal1	VAR_OUTPUT	BOOL	Information about content of actual value 1 (air supply port (2)): 0: Valve position (1 digit = 0.01 %) 1: Pressure (relative) (1 digit = 1 mbar)
xMeaningOfActVal2	VAR_OUTPUT	BOOL	Information about content of actual value 1 (air supply port (4)): 0: Valve position (1 digit = 0.01 %) 1: Pressure (relative) (1 digit = 1 mbar)

5.2.3 Motion App #05 (Supply and exhaust air flow control)

Designation	Variable type	Data type	Description
xMeaningOfActVal1	VAR_OUTPUT	BOOL	Information about content of actual value 1: 0: Opening degree exhaust flow (1 digit = 0.01 %) 1: Pressure (relative) at air supply port (2) (1 digit = 1 mbar)
xMeaningOfActVal2	VAR_OUTPUT	BOOL	Information zum Inhalt von Istwert 2: 0: Opening degree supply flow (1 digit = 0.01 %) 1: Pressure (relative) at air supply port (4) (1 digit = 1 mbar)

5.2.4 Motion App #10 (Flow control)

Designation	Variable type	Data type	Description
x2or4outOfSpec	VAR_OUTPUT	BOOL	Output is TRUE if operation of (2) or (4) is out of specification. The control accuracy may be limited.
xFlow2FromMeasurement	VAR_OUTPUT	BOOL	Output is TRUE if the actual value indication for the flow at (2) is based on measurement (higher accuracy).
xFlow4FromMeasurement	VAR_OUTPUT	BOOL	Output is TRUE if the actual value indication for the flow at (4) is based on measurement (higher accuracy).
xRatioSetToMaxOverLimit2	VAR_OUTPUT	BOOL	Output is TRUE if the ratio of target flow at (2) to maximum adjustable flow is greater than the specified limit (95% + value of parameter "Offset target flow rate limitation info for (2)").
xRatioSetToMaxOverLimit4	VAR_OUTPUT	BOOL	Output is TRUE if the ratio of target flow at (4) to maximum adjustable flow is greater than the specified limit (95% + value of parameter "Offset target flow rate limitation info for (4)").

5.2.5 Motion App #11 (Soft Stop)

Designation	Variable type	Data type	Description
xContactPressure	VAR_OUTPUT	BOOL	Output is TRUE if the contact pressure calculated internally by the Motion App has been reached in the end position.
xPistonDetectRet	VAR_OUTPUT	BOOL	Output is TRUE if the piston is within the detection range of the assigned partial stroke sensor for the retracted end position. The assignment is made with the help of the parameterization blocks of the Motion Apps (→ Example of assignment of inputs).
xPistonDetectAdv	VAR_OUTPUT	BOOL	Output is TRUE if the piston is within the detection range of the assigned partial stroke sensor for the extended end position. The assignment is made with the help of the parameterization blocks of the Motion Apps (→ Example of assignment of inputs).

5.2.6 Motion App #12 (Leakage diagnostics)

Designation	Variable type	Data type	Description
byStateLeakageAt2	VAR_OUTPUT	BYTE	State of the leakage at air supply port (2): 0: Defective (red) 1: Critical (orange) 2: Warning (yellow) 3: Good (green)
byStateLeakageAt4	VAR_OUTPUT	BYTE	State of the leakage at air supply port (4): 0: Defective (red) 1: Critical (orange) 2: Warning (yellow) 3: Good (green)
byStateProcedure	VAR_OUTPUT	BYTE	State of leakage diagnostics: 0: Diagnostics inactive 1: Diagnostics active 2: Diagnostics cancelled 3: Diagnostics partially completed ((2) valid) 4: Diagnostics partially completed ((4) valid) 5: Diagnostics completed

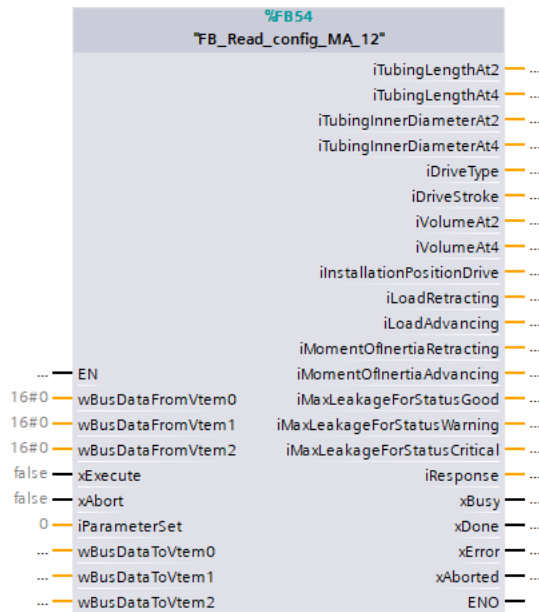
5.2.7 Motion App #33 (Positioning)

Designation	Variable type	Data type	Description
xTargetPositionReached	VAR_OUTPUT	BOOL	Output is TRUE if the absolute deviation of the actual position from the target position does not exceed the value defined in parameter “Tolerance for target position reached” for at least the duration defined in parameter “Damping time until ‘target position reached’”.
xOvershootInPlannedPath	VAR_OUTPUT	BOOL	Output is TRUE if the planned path contains an overshoot over the target position.
xControlledStop	VAR_OUTPUT	BOOL	Output is TRUE if a controlled stop was triggered by an end position violation (only active if the end position monitoring was not deactivated by the corresponding application parameter)

5.2.8 Teach-in run

Designation	Variable type	Data type	Description
iTeachInRunState	VAR_OUTPUT	INT	Current state of the teach-in run as number (→ Data type eResponseTeachInRun)
sTeachInRunState	VAR_OUTPUT	STRING	Current state of the teach-in run as string

6 FB_Read_config_MA... – Reading parameterisation of a Motion App



6.1 Functional description

The function blocks are used to read out the Motion App-specific parameter values and make each value available at a corresponding output. The parameter set whose values are to be read is defined via its own input. The transfer of the parameter values to the PLC takes place with a rising edge at the “xExecute” input. The corresponding valve is set to transfer mode and the parameter values are transferred one after the other. The status of the transfer is displayed at the outputs.

After completion of the transfer (“xDone” output = TRUE), the values at the outputs remain until the function block is reset by setting the “xExecute” input to FALSE. The transfer mode is ended and the corresponding valve is set to valve mode “inactive”.



Internally, the function blocks call the function block “FB_Upload_Multiple_Parameters” and transfer the values from the array of the data type “stTransferPackage” to the outputs.

6.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function blocks.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed, the function block is reset after the transmission is completed and the valve is set to the “inactive” state (valve mode = 61).
iParameterSet	VAR_INPUT	INT	Input for selecting the parameter set whose values are to be read.

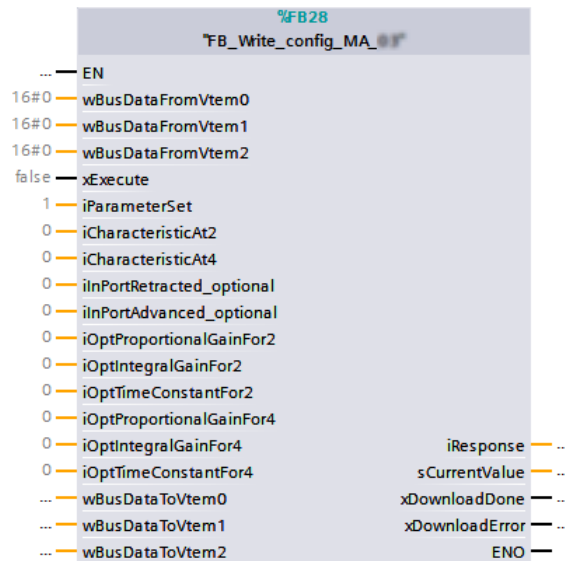
Designation	Variable type	Data type	Description
<ParameterName>	VAR_OUTPUT	<data type>	Outputs for parameter values that are to be transmitted. The outputs are specific to the Motion App. Their function is described in the documentation of the respective Motion App and in the comments of the outputs.
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: → Data type eTransferResponse .
xBusy	VAR_OUTPUT	BOOL	Output is TRUE as long as a process is active in the function block.
xDone	VAR_OUTPUT	BOOL	Output is TRUE when the transfer could be concluded without an error.
xError	VAR_OUTPUT	BOOL	Output is TRUE when errors occurred during the transfer.
xAborted	VAR_OUTPUT	BOOL	Output is TRUE when the transmission was terminated by the input “xAbort”. Output is reset when xExecute and xAbort are FALSE at the same time.

6.2.1 Data type eTransferResponse

The data type is used as an enumeration and serves for the output of feedbacks to the current status of the transfer.

Meaning	Data type	Return value
invalid_Channel	INT	1
invalid_TransferControl	INT	2
invalid_AddressedTarget	INT	3
invalid_Index	INT	4
invalid_TransferValue	INT	5
invalid_Combination	INT	6
access_denied_DataInUse	INT	7
no_data_empty_slot	INT	8
invalid_ParameterSet	INT	9
access_denied_WebConfig	INT	10
access_denied_saving	INT	11
no_valve_detected	INT	15
reading_transfermode	INT	20
ready_for_new_transfer	INT	21
preparing_transfer	INT	22
transferring	INT	23
transfer_successful	INT	24
FB_needs_rising_edge_xExecute	INT	25
Valve_Mode_is_not_61_inactive	INT	26
Invalid_firmware_version	INT	27
communication_failed	INT	28
invalid_module_number	INT	29
resetting	INT	30
aborting	INT	31

7 FB_Write_config_MA... – Writing parameterisation of a Motion App



7.1 Functional description

The function blocks for transmitting the parameters required for the operation of a Motion App make an input available for each parameter to which the desired parameter value can be applied. An additional input can be used to specify the parameter set into which the values are to be written.

The transmission of the parameter values to the Motion Terminal takes place at a rising edge at input “xExecute”. The corresponding valve is put into the transfer mode and the parameter values are transmitted sequentially. The status of the transmission is depicted at the outputs.

When ending the function block with a falling edge at input “xExecute”, the transfer mode is ended and the valve is set to “inactive” (valve mode = inactive).



Internally the function blocks call up function block “FB_Download_Multiple_Parameters” and transfer the values to the inputs as array of the data type “stTransferPackage”.

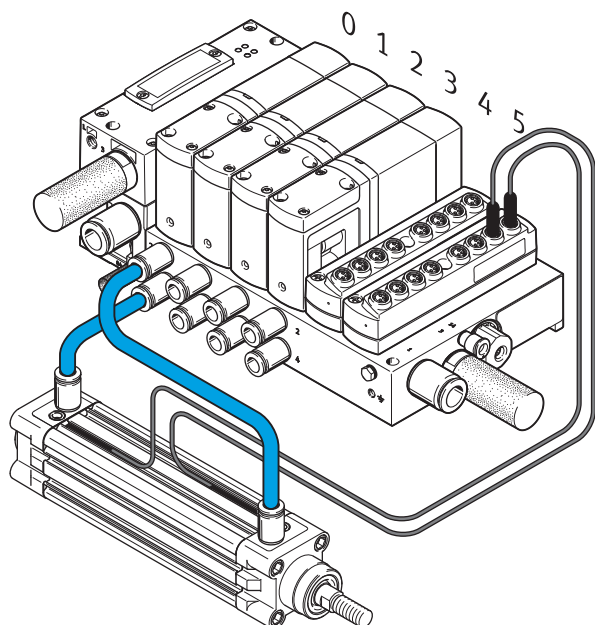
7.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function blocks.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	Rising edge at the input starts the transmission. The transfer mode ends with the removal of the “TRUE” signal and an unfinished transfer is aborted.
xAbort	VAR_INPUT	BOOL	TRUE signal at the input breaks off the procedure in the function block. The transfer mode is ended if required and the valve is set to the status “inactive” (valve mode = 61).
iParameterSet	VAR_INPUT	INT	Input for selecting the parameter set into which the values are to be taken over.

Designation	Variable type	Data type	Description
<ParameterName>	VAR_INPUT	<data type>	Inputs for parameter values that are to be transmitted. The values must be constant for the duration of the transmission. The inputs are specific to the Motion App. Their function is described in the documentation of the respective Motion App and in the comments of the inputs.
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: ➔ Data type eTransferResponse .
sCurrentValue	VAR_OUTPUT	STRING	Information on which value is currently transferred. This information can be used for error identification when the transfer is terminated.
xDownloadDone	VAR_OUTPUT	BOOL	Output becomes TRUE when the transfer could be concluded without an error.
xDownloadError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

7.2.1 Example of assignment of inputs



For this example, the Motion App 6 (ECO drive) should be parameterised on the valve on slot 0.

A piston rod cylinder is connected at the air supply ports of slot 0.

The proximity sensors SMT-8M-A-PS-24V... at both end positions of the piston rod cylinder are connected at the inputs 0 and 1 of the digital input module.

For the configuration of the input module (connected sensor type and alignment of sensor for analogue input modules), the function blocks “FB_Write_config_AI_Module” and FB_Write_config_DI_Module can be used (➔ [VTEM_AOI_Write_config_DI/AI_Module – Writing parameterisation of the input modules](#)).

The assignment of inputs to the valve must be parameterised via the system parameters 70 and 71 of the valve (➔ Descriptions of the Motion Apps).

When using the function blocks “FB_Write_config_MA...”, the inputs “iInPortRetracted” or “iInPortRetracted_optional” and “iInPortAdvanced” or “iInPortAdvanced_optional” are used. (The inputs “..._optional” are a component of the function blocks for Motion Apps that do not need sensors for operation.)

Using the function block “FB_Write_config_MA_06”

The values for the parameters are calculated from the position of the input module (slot), multiplied by a factor of 10 and then added to the number of the input on this input module (0 ... 7).

iInPortRetracted: Number of the slot of the digital input module (5) × 10 + number of input at which the proximity sensor for the “Retracted” position is connected (0).
 $5 \times 10 + 0 = 50$

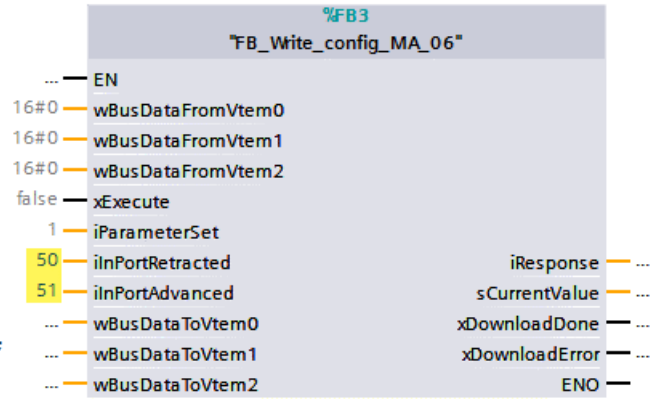
iInPortAdvanced: Number of the slots of the digital input module (5) × 10 + number of input at which the proximity sensor for the “Advanced” position is connected (1).
 $5 \times 10 + 1 = 51$

Example in SCL

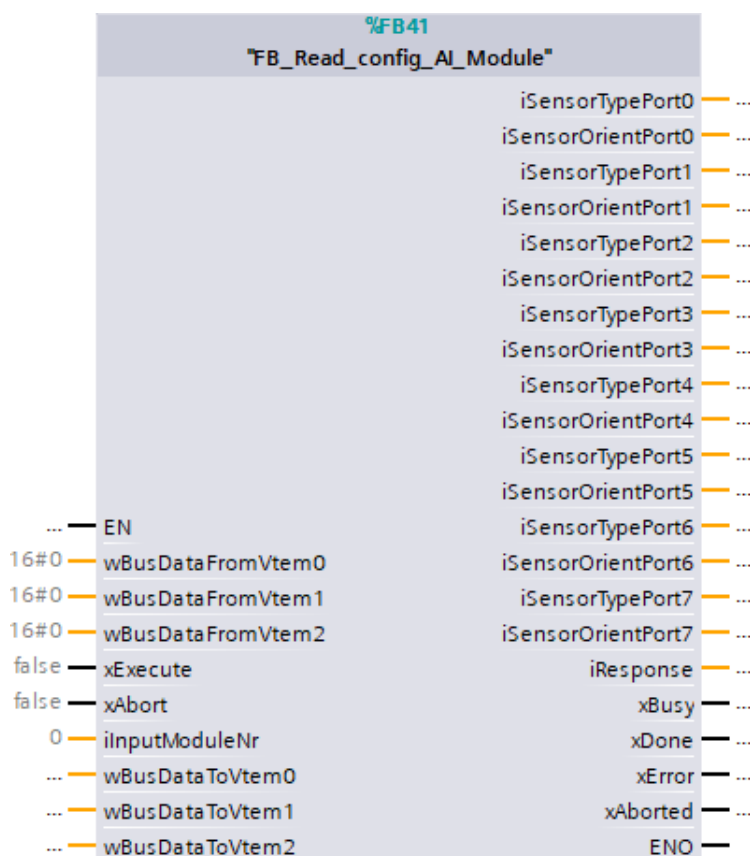
```

"FB_Write_config_MA_06_DB" (wBusDataFromVtem0:=16#0,
                           wBusDataFromVtem1:=16#0,
                           wBusDataFromVtem2:=16#0,
                           xExecute:=false,
                           iParameterSet:=1,
                           iInPortRetracted:=50,
                           iInPortAdvanced:=51,
                           iResponse=>_int_out_,
                           sCurrentValue=>_string_out_,
                           xDownloadDone=>_bool_out_,
                           xDownloadError=>_bool_out_,
                           wBusDataToVtem0:=_word_inout_,
                           wBusDataToVtem1:=_word_inout_,
                           wBusDataToVtem2:=_word_inout_);

```

Example in FBD

8 FB_Read_config_DI/AI_Module – Reading parameterisation of the input modules



8.1 Functional description

The function blocks transmit the parameter values of the digital and analogue input modules and make each value available at a separate output. The number of the input module whose parameters are to be read can be defined via an input (starting from the left with the first input module with the number 1).

The transfer of the parameter values to the PLC starts when the “xExecute” input is TRUE. The corresponding valve is set to transfer mode and the parameter values are transferred one after the other. The status of the transfer is displayed at the outputs of the function block.

The valve is set to “inactive” mode (valve mode = 61 (inactive)) for the following events:

Event	iResponse	xBusy	xDone	xError	xAborted
Successful transmission of all parameters	24 (transfer_successful)	FALSE	TRUE	FALSE	FALSE
Error during the transmission of parameters	Cause of error (→ Data type eTransferResponse)	FALSE	FALSE	TRUE	FALSE
Termination of transmission via „xAbort“ = TRUE	31 (aborting)	FALSE	FALSE	FALSE	TRUE

After one of this events the function block is reset if there is a FALSE signal at the inputs “xAbort” and “xExecute”.



Internally, the function blocks call the function block “FB_Upload_Multiple_Parameters” and transfer the values from the array of the data type “stTransferPackage” to the outputs.

8.2 Inputs and outputs

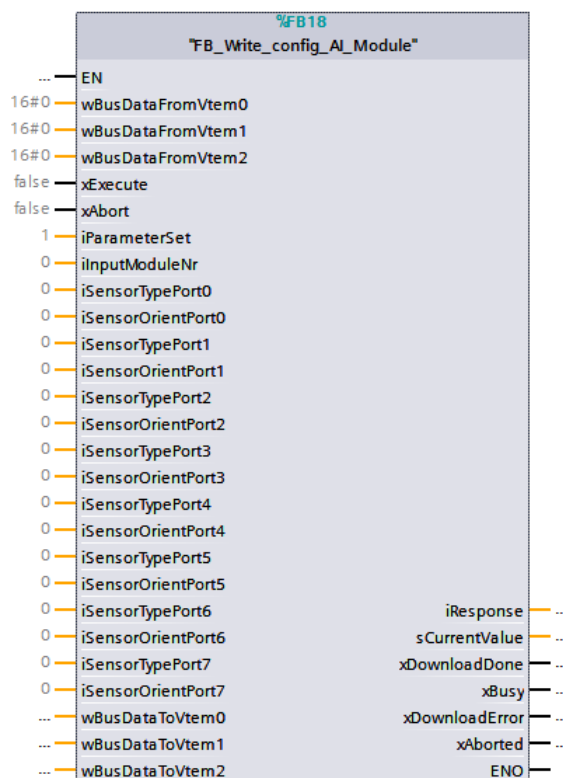
The following table contains the variables of the outputs and inputs of the function blocks.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	A rising edge at the input starts the transmission. When the “TRUE” signal is removed, the function block is reset after the transmission is completed and the valve is set to the “inactive” state (valve mode = 61).
xAbort	VAR_INPUT	BOOL	TRUE signal at the input breaks off the procedure in the function block. The transfer mode is ended if required and the valve is set to the status “inactive” (valve mode = 61).
iInputModuleNr	VAR_INPUT	INT	Number of the input module whose parameters are to be read, starting with 1 at the first input module from the left.
iSensorTypePort0 ... 7	VAR_OUTPUT	INT	ID of the sensor type connected to the respective input. 0: no sensor 1001: SMT-8M-A-PS-24V... (digital) 2001: SDAP-MHS-M50 (position sensor) 2002: SDAP-MHS-M100 (position sensor) 2003: SDAP-MHS-M160 (position sensor) 2901: User-definded position sensor 1 ¹⁾ 2902: User-definded position sensor 2 ¹⁾ 2903: User-definded position sensor 3 ¹⁾ 3001: SFAB-50U-...-2SA-... (flow sensor) 3002: SFAB-200U-...-2SA-... (flow sensor) 3003: SFAB-600U-...-2SA-... (flow sensor) 3004: SFAB-1000U-...-2SA-... (flow sensor) 3101: SFAH-50U-...-PNVBA-... (flow sensor) 3102: SFAH-100U-...-PNVBA-... (flow sensor) 3103: SFAH-200U-...-PNVBA-... (flow sensor) 3901: User-defined flow sensor 1 ¹⁾ 3902: User-defined flow sensor 2 ¹⁾ 3903: User-defined flow sensor 3 ¹⁾
iSensorOrientPort0 ... 7 ²⁾	VAR_OUTPUT	INT	Signal behaviour of the sensor connected to the respective input of the analogue input module. Position transmitter 0: Output signal increases when moving out ³⁾ 1: Output signal falls when moving out ³⁾ Flow sensor 0: Positive flow values (“pushing”) 1: Negative flow values (“sucking”)
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: → Data type eTransferResponse .

Designation	Variable type	Data type	Description
xBusy	VAR_OUTPUT	BOOL	Output is TRUE as long as a process is active in the function block.
xDone	VAR_OUTPUT	BOOL	Output is TRUE when the transfer could be concluded without an error.
xError	VAR_OUTPUT	BOOL	Output is TRUE when errors occurred during the transfer.
xAborted	VAR_OUTPUT	BOOL	Output is TRUE when the transmission was terminated by the input “xAbort”. Output is reset when xExecute and xAbort are FALSE at the same time.

- 1) User-defined sensor have to be defined via the transfer mode (→ [Documentation for Motion Terminal VTEM](#))
- 2) Only for analogue input modules
- 3) Advanced: air supply port (4) pressurised, air supply port (2) exhausted

9 FB_Write_config_DI/AI_Module – Writing parameterisation of the input modules



9.1 Functional description

The function blocks transfer the parameters for the digital and analogue input modules and make an input available for each parameter to which the desired parameter value can be applied. Via another input, the number of the input module to be parameterised can be specified (beginning from left for the first input module with the number 1).

The transmission of the parameter values to the Motion Terminal starts if the input “xExecute” is true. The corresponding valve is put into the transfer mode and the parameter values are transmitted sequentially. The status of the transmission is depicted at the outputs of the function block.

The valve is moved to “inactive” (valve mode = 61 (inactive) for the following events:

Event	iResponse	sCurrentValue	xBusy	xDownload Done	xDownload Error	xAborted
Successful transmission of all parameters	24 (transfer_successful)	„AllParametersWritten“	FALSE	TRUE	FALSE	FALSE
Error during the transmission of parameters	Cause of error (→ Data type eTransferResponse)	Name of the input value for which the error occurred during transmission.	FALSE	FALSE	TRUE	FALSE
Termination of transmission via „xAbort“ = TRUE	Value upon termination (→ Data type eTransferResponse)	Name of the input value for which the termination occurred during transmission.	FALSE	FALSE	FALSE	TRUE

After one of this events the function block is reset if there is a FALSE signal at the inputs “xAbort” and “xExecute”.



Internally the function blocks call up function block “FB_Download_Multiple_Parameters” and transfer the values at the inputs as array of data type “stTransferPackage”.

9.2 Inputs and outputs

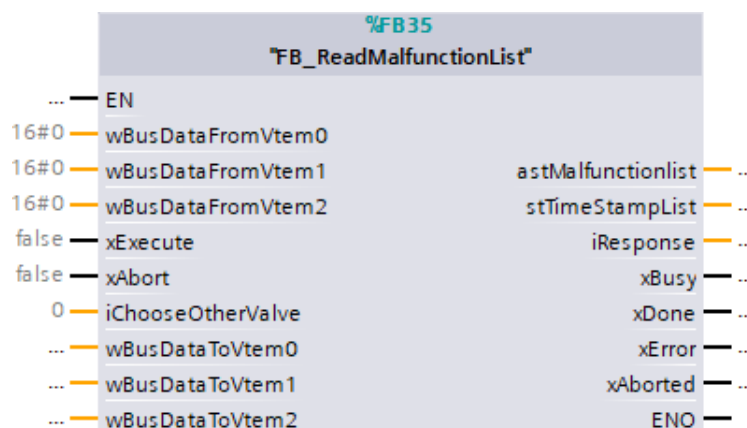
The following table contains the variables of the outputs and inputs of the function blocks.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	TRUE signal at the input starts the transmission. When the TRUE signal is removed, the transfer mode is terminated after the transfer is completed.
xAbort	VAR_INPUT	BOOL	TRUE signal at the input breaks off the procedure in the function block. The transfer mode is ended if required and the valve is set to the status “inactive” (valve mode = 61).
iInputModuleNr	VAR_INPUT	INT	Number of the input module to be parameterised, beginning from left to right with the number 1 for the first input module and going in increments.
iSensorTypePort0 ... 7	VAR_INPUT	INT	ID of the sensor type connected to the respective input. 0: no sensor 1001: SMT-8M-A-PS-24V... (digital) 2001: SDAP-MHS-M50 (position sensor) 2002: SDAP-MHS-M100 (position sensor) 2003: SDAP-MHS-M160 (position sensor) 2901: User-definded position sensor 1 ¹⁾ 2902: User-definded position sensor 2 ¹⁾ 2903: User-definded position sensor 3 ¹⁾ 3001: SFAB-50U-...-2SA-... (flow sensor) 3002: SFAB-200U-...-2SA-... (flow sensor) 3003: SFAB-600U-...-2SA-... (flow sensor) 3004: SFAB-1000U-...-2SA-... (flow sensor) 3101: SFAH-50U-...-PNVBA-... (flow sensor) 3102: SFAH-100U-...-PNVBA-... (flow sensor) 3103: SFAH-200U-...-PNVBA-... (flow sensor) 3901: User-defined flow sensor 1 ¹⁾ 3902: User-defined flow sensor 2 ¹⁾ 3903: User-defined flow sensor 3 ¹⁾
iSensorOrientPort0 ... 7 ²⁾	VAR_INPUT	INT	Signal behaviour of the sensor connected to the respective input of the analogue input module. Position transmitter 0: Output signal increases when moving out ³⁾ 1: Output signal falls when moving out ³⁾ Flow sensor 0: Positive flow values (“pushing”) 1: Negative flow values (“sucking”)
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: → Data type eTransferResponse .

Designation	Variable type	Data type	Description
sCurrentValue	VAR_OUTPUT	STRING	Information on which value is currently transferred. This information can be used for error identification when the transfer is terminated.
xBusy	VAR_OUTPUT	BOOL	Output is TRUE as long as a process is active in the function block.
xDownloadDone	VAR_OUTPUT	BOOL	Output is TRUE when the transfer could be concluded without an error.
xDownloadError	VAR_OUTPUT	BOOL	Output is TRUE when errors occurred during the transfer.
xAborted	VAR_OUTPUT	BOOL	Output is TRUE when the transmission was terminated by the input "xAbort". Output is reset when xExecute and xAbort are FALSE at the same time.

- 1) User-defined sensor have to be defined via the transfer mode (➔ [Documentation for Motion Terminal VTEM](#))
- 2) Only for analogue input modules
- 3) Advanced: air supply port (4) pressurised, air supply port (2) exhausted

10 FB_ReadMalfunctionList – Read out malfunction list



10.1 Functional description

The function block “FB_ReadMalfunctionList” automatically reads the contents of the malfunction list of a valve and passes it to the output “astMalfunctionlist”.

The output “astMalfunctionlist” passes an array of 40 elements of the type “stMalfunction”. Each element thereby contains a complete entry of the malfunction list consisting of “Code”, “Subcode”, “Classification” and timestamp („hours“, „minutes“, „seconds“ und „milliseconds“). A time stamp in the format “stTimeStamp” is output at the “stTimeStampList” output, which describes the time at which the fault list was read out. By comparing this time with the times of the fault list entries, their event time can be determined. The times refer to the operating time of the Motion Terminal since it was powered on.

Via the input „iChooseOtherValve“ it can be defined from which valve the malfunction list should be read. This way the process data of any other slot can be used for this purpose.

10.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	TRUE signal at the input starts the transmission. If the TRUE signal is removed, the transfer mode is terminated and the function block is reset as soon as the transfer process has been completed successfully (iResponse = “done” or “no_entries_in_malfuncrion_list”) or with an error (xError = TRUE).
xAbort	VAR_INPUT	BOOL	TRUE signal at the input terminates the process in the function block (Response = “aborting”). The transfer mode is terminated if necessary and the valve is set to the “inactive” state (valve mode = 61).
iChooseOtherValve	VAR_INPUT	INT	Select the valve whose malfunction list is to be read. 0: Read the malfunction list of the valve whose process data the function block uses. 100 ... 107: Read the malfunction list of the valve in slot 0 (100) ... 7 (107).

Designation	Variable type	Data type	Description
astMalfunctionlist	VAR_OUTPUT	ARRAY [1..40] OF stMalfunction	Content of the malfunction list (→ Data type stMalfunction). The content of the array is complete as soon as xDone = TRUE. The output is reset as soon as xAbort = FALSE and xEnable = FALSE.
stTimeStampList	VAR_OUTPUT	stTimeStamp	Creation time of the malfunction list Data type stTimeStamp .
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning:→ Data type eResponseReadMalfunctionList .
xBusy	VAR_OUTPUT	BOOL	Output is TRUE as long as the function block executes an action.
xDownloadDone	VAR_OUTPUT	BOOL	Output is TRUE when the transfer could be concluded without an error (even if the malfunction list is empty).
xDownloadError	VAR_OUTPUT	BOOL	Output is TRUE if an error occurred during the transmission process
xAborted	VAR_OUTPUT	BOOL	Output is TRUE if the transmission was aborted by the input “xAbort”. Output is reset if xExecute and xAbort are FALSE at the same time.

10.2.1 Data type eResponseReadMalfunctionList

The data type is used as an enumeration and serves for outputting module-specific feedback messages. It also contains the values of “AppState” in case of invalid output data with an offset of +100 and the return values of the transfer mode from the enumeration eTransferResponse with an offset of +200.

Meaning	Data type	Return value
idle	INT	0
setting_valve_to_inactive	INT	1
getting_malfunction_count	INT	2
transferring	INT	3
done	INT	4
no_entries_in_malfunction_list	INT	5
error_while_transmitting_data_to_vtem	INT	6
aborting	INT	7
resetting	INT	8
FB_needs_rising_edge_xExecute	INT	10
reserved	INT	100
invalid_SetValveMode	INT	101
invalid_SetAppControl	INT	102
invalid_SetAppOption	INT	103
invalid_SetPointValue1	INT	104
invalid_SetPointValue2	INT	105
invalid_configuration	INT	106
access_denied_WebConfig	INT	110
access_denied_saving_parameterisation	INT	111
no_license_for_this_MA	INT	112

Meaning	Data type	Return value
all_licenses_in_use	INT	113
invalid_license_file	INT	114
no_valve_detected	INT	115
valve_self_calibration_running	INT	116
invalid_Channel	INT	201
invalid_TransferControl	INT	202
invalid_iChooseOtherValve	INT	203
invalid_Index	INT	204
invalid_TransferValue	INT	205
invalid_Combination	INT	206
access_denied_DataInUse	INT	207
no_data_empty_slot	INT	208

10.2.2 Data type stMalfunction

The data type represents the content of an entry in the malfunction list, consisting of malfunction code, malfunction subcode, classification of the malfunction and time stamp of the entry.

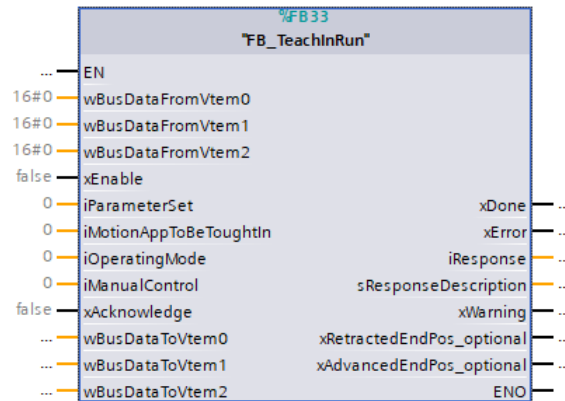
Designation	Data type	Description
iMalfunctionCode	INT	Code of malfunction list entry
iMalfunctionSubcode	INT	Subcode of malfunction list entry
iMalfunctionClassification	INT	Classification of malfunction list entry: 1: active error 2: inactive error 3: warning
diHours	DINT	Time stamp (hours) of the malfunction list entry
iMinutes	INT	Time stamp (minutes) of the malfunction list entry
iSeconds	INT	Time stamp (seconds) of the malfunction list entry
iMilliseconds	INT	Time stamp (milliseconds) of the malfunction list entry

10.2.3 Data type stTimeStamp

The data type represents the content of a timestamp, consisting of hours, minutes, seconds and milliseconds.

Designation	Data type	Description
diHours	DINT	Time stamp (hours)
iMinutes	INT	Time stamp (minutes)
iSeconds	INT	Time stamp (seconds)
iMilliseconds	INT	Time stamp (milliseconds)

11 FB_TeachInRun – Parameterise and execute teach-in run



11.1 Functional description

The function block FB_TeachInRun is used to create learning data with the help of the teach-in run (valve mode = 60).

The corresponding inputs of the function block can be used to determine for which motion app the learning data is to be created (iMotionAppToBeTaughtIn) and in which parameter set it is to be stored (iParameterSet).

The operating mode (iOperatingMode) and, for manual operation, the control (iManualControl) of the teach-in run can also be specified via the corresponding inputs of the function block.

In addition, the function block offers an input for acknowledging inactive errors (xAcknowledge). Acknowledgement takes place on a rising edge at the input.

On the output side, the function block provides a double output for the responses of the function block as integer value (iResponse) and as string (sResponseDescription) apart from the indicators for the successfully completed (xDone) or terminated due to an error (xError) execution of the function block.

Further information from the AppState of the teach-in run (status of the inputs, active warning) is also provided via separate outputs.

Executing the function block with iOperatingMode = 0 (Stop) results in the teach-in run being activated (valve mode = 60) but not started. This means that no valve positions are changed yet. In this case, the output “iResponse” and “sResponseDescription” contain the result of the last teach-in run executed. If no teach-in run has yet been performed for the Motion App to be taught-in, the outputs output the value “1” or “TeachInRun_unlearned”.

If write access via the WebConfig interface is activated during the active state of the function block, it returns the value “110” or “access_denied_WebConfig” at output iResponse.

11.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xEnable	VAR_INPUT	BOOL	TRUE: Start the function block FALSE: Stop the function block in any state. The valve and the function block are reset. Reset is finished when iResponse = 210 (“idle”) → Data type eResponseTeachInRun . Until then the function block must be called.

Designation	Variable type	Data type	Description
iParameterSet	VAR_INPUT	INT	0: active parameter set is used 1...5: active parameter set is changed to the value at this input for the duration of the teach-in run and then reset to the previous parameter set
iMotionAppToBeTaughtIn	VAR_INPUT	INT	ID of the Motion App, that is to be taught in.
iOperationMode	VAR_INPUT	INT	0: Stop 1: Auto 2: Manual (only Motion App #11)
iManualControl	VAR_INPUT	INT	0: Block 1: Advance 2: Retract 3: Exhaust
xAcknowledge	VAR_INPUT	BOOL	A rising edge acknowledges inactive errors in the malfunction list of the valve.
xError	VAR_OUTPUT	BOOL	Output becomes TRUE if an error occurs during execution of the function block or teach-in run.
iResponse	VAR_OUTPUT	INT	Response to the status of the function block or teach-in run as a numerical value. Meaning: ➔ Data type eResponseTeachInRun .
sResponseDescription	VAR_OUTPUT	STRING	Response to the status of the function block or teach-in run as text output according to data type eResponseTeachInRun.
xWarning	VAR_OUTPUT	BOOL	Output is controlled by bit 7 of the app state.
xRetractedEndPos_optional	VAR_OUTPUT	BOOL	The output corresponds to the state at the assigned input of the digital input module for the retracted end position (➔ Description of Motion Terminal VTEM, function and parameterisation). The assignment is made with the aid of the parameterisation function blocks of the Motion Apps (➔ Example of assignment of inputs).
xAdvancedEndPos_optional	VAR_OUTPUT	BOOL	The output corresponds to the state at the assigned input of the digital input module for the advanced end position (➔ Description of Motion Terminal VTEM, function and parameterisation). The assignment is made with the aid of the parameterisation function blocks of the Motion Apps (➔ Example of assignment of inputs).

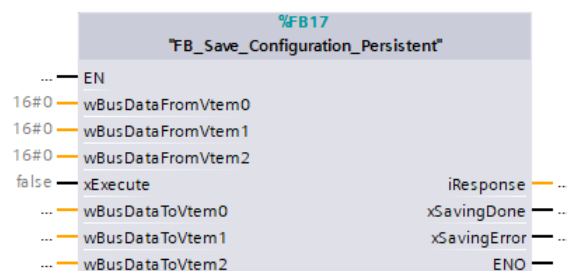
11.2.1 Data type eResponseTeachInRun

The data type is used as an enumeration and serves to output module-specific responses (e.g. the status of the function block) and also contains return values on the status of the teach-in run. Part of the stored return values corresponds to the value of the output “AppState” in case of faulty output data (wBusDataToVTEM...) with an offset of +100.

Meaning	Data type	Return value
TeachInRun_reserved	INT	0
TeachInRun_unlearned	INT	1
TeachInRun_check_tubes	INT	2
TeachInRun_choose_control	INT	3
TeachInRun_tune_trajectory	INT	4
TeachInRun_determine_reference_value	INT	5
TeachInRun_identification	INT	6

Meaning	Data type	Return value
TeachInRun_operable_with_reduced_performance	INT	10
TeachInRun_completed_with_limitations	INT	11
TeachInRun_partially_completed_only_2	INT	12
TeachInRun_partially_completed_only_4	INT	13
TeachInRun_completed	INT	14
TeachInRun_error	INT	15
invalid_SetValveMode	INT	101
invalid_SetAppControl	INT	102
invalid_SetAppOption	INT	103
invalid_SetPointValue1	INT	104
invalid_SetPointValue2	INT	105
invalid_configuration	INT	106
access_denied_WebConfig	INT	110
access_denied_saving_parameterisation	INT	111
no_license_for_this_MA	INT	112
all_licenses_in_use	INT	113
invalid_license_file	INT	114
no_valve_detected	INT	115
valve_self_calibration_running	INT	116
idle	INT	210
storing_first_cycle_input_data	INT	215
resetting_valve_to_61	INT	220
ReadingActiveParameterSet	INT	230
ChangingActiveParameterSet	INT	232
writing_parameter_MotionAppToBeToughtIn	INT	234
running_teachIn_run	INT	240
teachin_run_completed	INT	260
error_occured_check_DiagnosticMemory	INT	270
input_data_invalid	INT	271
FB_needs_rising_edge_xEnable	INT	272
acknowledge_needed	INT	273
Invalid_MotionAppToBeToughtIn	INT	274
resetting	INT	300

12 FB_Save_Configuration_Persistent – Save parameterisation persistent



12.1 Functional description

The function block serves for the simple call-up of function “Save configuration persistent”. The data for the valve currently being parameterised are saved permanently on the controller of the Motion Terminal.

12.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	Rising edge at the input starts a saving process. A new saving process therefore requires a new rising edge. If the “TRUE” signal is removed before the save operation is completed (xSavingDone = TRUE), the transfer mode is terminated.
iResponse	VAR_OUTPUT	INT	Response to the status of saving process. Meaning: → Data type eSavingResponse .
xSavingDone	VAR_OUTPUT	BOOL	The output becomes “TRUE” when the saving process is completed.
xSavingError	VAR_OUTPUT	BOOL	The output becomes “TRUE” when an error occurred during the saving process.

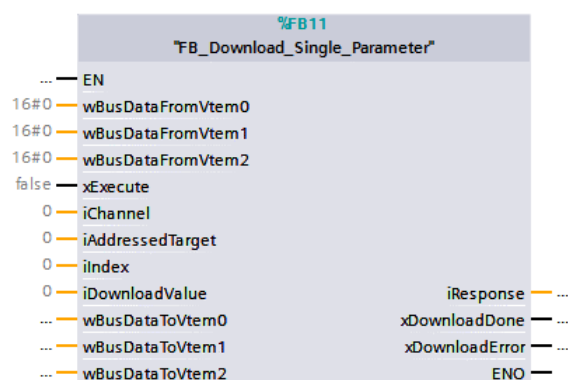
12.2.1 Data type eSavingResponse

The data type is used as an enumeration and serves for the output of feedbacks to the current status of the saving process.

Meaning	Data type	Return value
saving_in_progress	INT	1
saving_successful	INT	2
saving_not_possible	INT	3
saving_failed	INT	4
access_denied_WebConfig	INT	10
access_denied_saving	INT	11
no_valve_detected	INT	15
preparing_to_save	INT	20
ready_to_save	INT	21

Meaning	Data type	Return value
FB_needs_rising_edge_xExecute	INT	25

13 FB_Download_Single_Parameter – Transfer single parameters to the Motion Terminal



13.1 Functional description

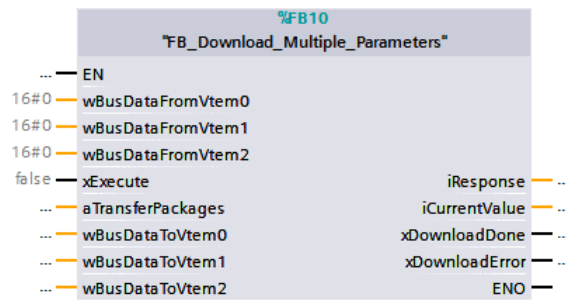
The function block Instruction transmits a value via the transfer mode from the PLC to the Motion Terminal. It can only be used for writable values.

13.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed before transmission is completed (xUploadDone = TRUE), the transfer mode is terminated.
iChannel	VAR_INPUT	INT	Transfer mode channel
iAddressedTarget	VAR_INPUT	INT	Channel-specific address of the target into which the value to be transmitted is to be written.
iIndex	VAR_INPUT	INT	Index of the parameter that is to be set on the transmitted value.
iDownloadValue	VAR_INPUT	INT	Value on which the parameter is to be set.
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: → Data type eTransferResponse in section VTEM_AOI_Write_config_MA ... – Writing parameterisation of a Motion App .
xDownloadDone	VAR_OUTPUT	BOOL	Output becomes TRUE when the transfer could be concluded without an error.
xDownloadError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

14 FB_Download_Multiple_Parameters – Transfer multiple parameters to the Motion Terminal



14.1 Functional description

The function block transmits a list with values via the transfer mode from the PLC to the Motion Terminal. The list consists of a one-dimensional array with the elements of data type “stTransferPackage”. Only writable values can be transferred.



A single invalid value in the list leads to the transfer process being aborted.

14.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed before transmission is completed (xUploadDone = TRUE), the transfer mode is terminated.
aTransferPackages	VAR_INPUT	ARRAY [0..100] OF stTransferPackage	List of values to be transferred (including channel, addressed target and index) (→ Data type stTransferPackage) After the last valid TransferPackage, an entry that is completely “0” must follow. This signals to the function block that no further transfers are necessary. This makes it possible to transfer up to 100 values. In such case entry 101 [100] must contain value “0” in all elements of the data type.
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: → Data type eTransferResponse in section VTEM_AOI_Write_config_MA_... – Writing parameterisation of a Motion App .
iCurrentValue	VAR_OUTPUT	INT	Number of the array element which is currently being transferred. This information can be used for identification of the time when the transfer is terminated.

Designation	Variable type	Data type	Description
xDownloadDone	VAR_OUTPUT	BOOL	Output becomes TRUE when the transfer could be concluded without an error.
xDownloadError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

14.2.1 Data type stTransferPackage

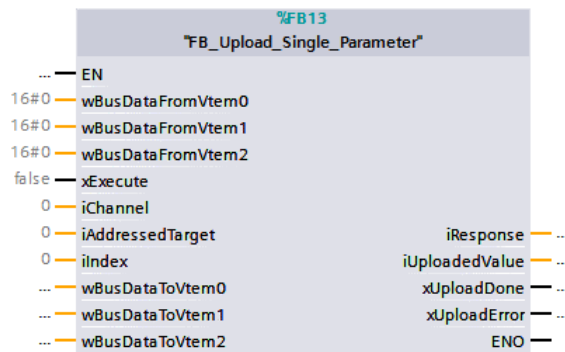
The data type type forms the pattern for a dataset of the list that is transferred with function blocks “FB_Download_Multiple_Parameters” and “FB_Upload_Multiple_Parameters”.

The data type contains one integer value each for the channel, the addressed target, the index and the value to be transferred.

The following table shows the elements of the data type.

Designation	Data type	Description
iChannel	INT	Transfer mode channel
iAddressedTarget	INT	Channel-specific address of the target whose value is to be written or read.
iIndex	INT	Index of the value to be transferred
iTransferValue	INT	Value to which the parameter should be set (download) or read value (upload).

15 FB_Upload_Single_Parameter – Reading single value from the Motion Terminal



15.1 Functional description

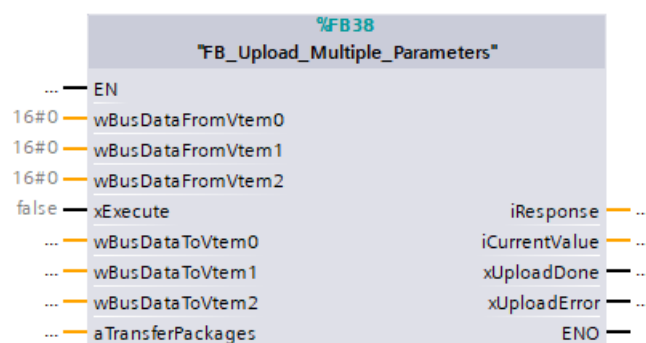
The function block transmits a value via the transfer mode from the Motion Terminal to the PLC.

15.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed before transmission is completed (xUploadDone = TRUE), the transfer mode is terminated.
iChannel	VAR_INPUT	INT	Transfer mode channel
iAddressedTarget	VAR_INPUT	INT	Channel-specific address of the target from which the value to be transmitted is to be read.
iIndex	VAR_INPUT	INT	Index for which the stored value is to be read out.
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: → Data type eTransferResponse in section VTEM_AOI_Write_config_MA_... – Writing parameterisation of a Motion App .
iUploadedValue	VAR_OUTPUT	INT	Read out value
xUploadDone	VAR_OUTPUT	BOOL	Output becomes TRUE when the transfer could be concluded without an error.
xUploadError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

16 FB_Upload_Multiple_Parameters – Reading multiple values from the Motion Terminal



16.1 Functional description

The function block transmits a list with values via the transfer mode from the Motion Terminal to the PLC. The list consists of a one-dimensional array from data type “stTransferPackage”.

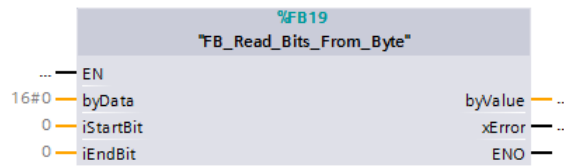
16.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
wBusDataFromVTEM0 wBusDataFromVTEM1 wBusDataFromVTEM2	VAR_INPUT	WORD	The 6 Bytes of input data received from the Motion Terminal must be applied to these inputs (→ Connecting I/O Data to the Add-On instructions).
wBusDataToVTEM0 wBusDataToVTEM1 wBusDataToVTEM2	VAR_IN_OUT	WORD	The 6 Bytes of output data that must be sent to the Motion Terminal are available at these outputs (→ Connecting I/O Data to the Add-On instructions).
xExecute	VAR_INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed before transmission is completed (xUploadDone = TRUE), the transfer mode is terminated.
aTransferPackages	VAR_IN_OUT	ARRAY [0..100] OF stTransferPackage	List of values to be transferred (including channel, addressed target and index) (→ Data type stTransferPackage) After the last valid TransferPackage, an entry that is completely “0” must follow. This signals to the function block that no further transfers are necessary. This makes it possible to transfer up to 100 values. In such case entry 101 [100] must contain value “0” in all elements of the data type.
iResponse	VAR_OUTPUT	INT	Response to the status of the transmission. Meaning: → Data type eTransferResponse in section VTEM_AOI_Write_config_MA_... – Writing parameterisation of a Motion App .
iCurrentValue	VAR_OUTPUT	INT	Number of the array element which is currently being transferred. This information can be used for identification of the time when the transfer is terminated.
xUploadDone	VAR_OUTPUT	BOOL	Output becomes TRUE when the transfer could be concluded without an error.
xUploadError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

17 Auxiliary function blocks

17.1 FB_Read_Bits_From_Byte – Read bit area from byte



17.1.1 Functional description

The function block simplifies the reading from areas within a byte. For this a byte is transferred to the function block, as well as the start bit and end bit of the area, whose value is to be output as byte.

Example

From output byte “byActualAppState” of instance “fbvalvecontrol_V0” of function block “FB_ValveControl”, bit 7 (Warning) is to be read and written to variable “byWarning”.



17.1.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Designation	Variable type	Data type	Description
byData	VAR_INPUT	BYTE	Byte from which the value of the area between start bit and end bit is to be read.
iStartBit	VAR_INPUT	INT	First bit (0 ... 7) of the area whose value is to be output as integer.
iEndBit	VAR_INPUT	INT	Last bit (0 ... 7) of the area whose value is to be output as integer.
byValue	VAR_OUTPUT	BYTE	Value of the specified area as byte.
xError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

17.2 FB_Read_Bits_From_Word – Read bit area from word



17.2.1 Functional description

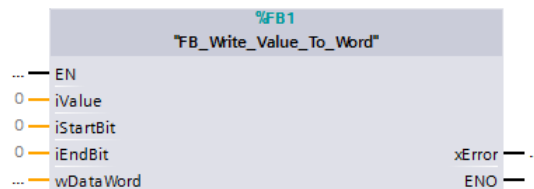
The function block simplifies the reading from areas within a word. For this a word is transferred to the function block, as well as the start bit and end bit of the area, whose value is to be output as integer.

17.2.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Bezeichnung	Variablentyp	Datentyp	Beschreibung
wData	VAR_INPUT	WORD	Word from which the value of the area between start bit and end bit is to be read.
iStartBit	VAR_INPUT	INT	First bit (0 ... 15) of the area whose value is to be output as integer.
iEndBit	VAR_INPUT	INT	Last bit (0 ... 15) of the area whose value is to be output as integer.
iValue	VAR_OUTPUT	INT	Value of the specified area as integer.
xError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

17.3 FB_Write_Value_To_Word – Write value within a word



17.3.1 Functional description

The function block simplifies the writing of values in an area of a word. For this the corresponding word is transferred to the function block, as well as the start bit and end bit of the area, into which the applied integer value is to be written.

17.3.2 Inputs and outputs

The following table contains the variables of the outputs and inputs of the function block.

Bezeichnung	Variablentyp	Datentyp	Beschreibung
iValue	VAR_INPUT	INT	Value to be written to the specified area.
iStartBit	VAR_INPUT	INT	First bit (0 ... 15) of the range in which the transferred integer value is to be written.
iEndBit	VAR_INPUT	INT	Last bit (0 ... 15) of the range in which the transferred integer value is to be written.
wDataWord	VAR_IN_OUT	WORD	Word to which the value between start bit and end bit is to be written.
xError	VAR_OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

A Technical appendix

A.1 Technical data

A.1.1 General

Technical data	
Version of the creation software	Siemens TIA Portal V19

B Index

A

App state	
interpretation	29

B

Bit area	
read from byte	57
read from word	58
write within word	58

C

Communication protocol	21
Connection to the controller	20
creation software	59

D

Data type	
eMA100OperatingMedium	19
eResponseReadMalfunctionList	19, 45
eResponseTeachInRun	19, 48
eResponseToValveModeSet	19, 26
eSavingResponse	19, 50
eTransferResponse	19, 34
eValveMode	19
eValveState	19
eValveType_MA_01_02	19
stMalfunction	19, 46
stTimeStamp	46
stTransferPackage	19, 54
Device description file	20

F

FB_Download_Multiple_Parameters	53
FB_Download_Single_Parameter	52
FB_Read_Bits_From_Byte	57
FB_Read_Bits_From_Word	58
FB_ReadMalfunctionList	44
FB_Save_Configuration_Persistent	50
FB_StateInterpreter	29
FB_TeachInRun	47
FB_Upload_Multiple_Parameters	56
FB_Upload_Single_Parameter	55
FB_ValveControl	24
FB_Write_config_DI/AI_Module	41
FB_Write_config_MA_...	35
FB_Write_Value_To_Word	58
Fundamentals	18

I

Inputs	
assignment	36
Intended Use	16
I/O data	22
transfer to the function blocks	22

L

library	
architecture	18
history	2
integrating	20
structure	19
version	2

M

Malfunction list	
read out	44
Motion App	
operate	24

P

Parameterisation	
input modules	
reading	38
writing	41
Motion App	35
save persistent	50
teach-in run	47
Parameter transfer to motion terminal	
multiple	53
single	52

R

Reading values from Motion Terminal	
multiple	56
single	55
Requirements	20

S

Safety instructions	16
save persistent	50
Service	16
State Interpreter	29

T

Target group	16
Technical data	59