Perfection in Automation
www.br-automation.com

Cognex Vision System

Library User Guide

Author: Stephan Stricker, B&R USA Atlanta
Revision: 1.0

1 Demo B&R hardware

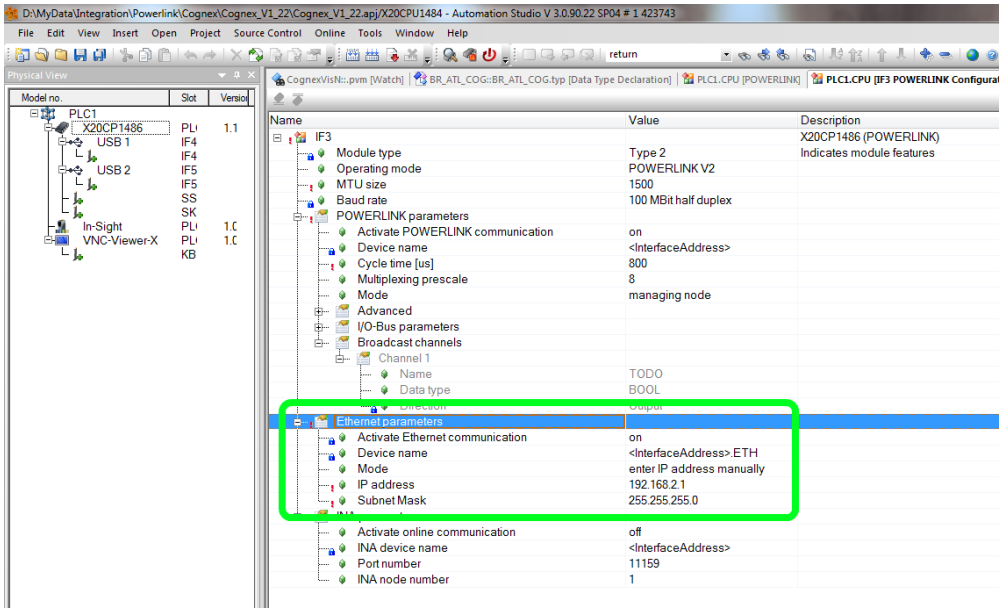
This document describes how to use the library “**BR_ATL_COG**”. The library connects to the Cognex telnet interface via TCPIP over Powerlink. The telnet supports a large variety of functions that can be called directly from the PLC. For more details use the InSight help file and search for “Native Mode”. All functions can be called while the camera is connected to the Powerlink interface. The only exception is first time setup where the camera is not configured for Powerlink. In this case the camera has to be connected to the onboard Ethernet interface.

1.1 Requirements

Automation Studio 3.0.90 SP4 or higher
Runtime V3.08 or higher
SG4 target

1.2 Setup

To use TCP over Powerlink it is important that Powerlink port has a valid IP address.



TIP The IP address cannot be in the same subnet as the standard Ethernet port.

1.3 Functions

1.3.1 NativeModeLogin

This function creates a connection to the Cognex camera telnet interface. The function will login automatically and does not require a user name or password.

Inputs

enable	The function is only executed when enable is = 1. If function responds with an error set enable = 0 and call the function once to reset the internal state machine.
ip	Camera IP address. When connected via Powerlink this address is 192.168.101.x where x is the Powerlink node number.

Outputs

ident	This is the reference number for this connection. This number has be connected to all other function blocks.
status	Status of function. Call the function as long as status is BUSY (65535).

1.3.2 NativeModeLogout

This function creates a connection to the Cognex camera telnet interface. The function will login automatically and does not require a user name or password.

Inputs

enable	The function is only executed when enable is = 1. If function responds with an error set enable = 0 and call the function once to reset the internal state machine.
ident	Reference from the NativeModeLogin command.

Outputs

status	Status of function. Call the function as long as status is BUSY (65535).

1.3.3 Sample Code

```
(* ---------------------------------------------------------------------------------------------------- *)
(* Connect to camera
(* ---------------------------------------------------------------------------------------------------- *)
IF(EDGEPOS(vkNativeModeConnect)) THEN
            NewLogEntry("Connecting to camera...", ADR(logbook), 20)
ENDIF
IF(vkNativeModeConnect = TRUE) THEN
            IF(NativeModeLoggedIn = FALSE) THEN
                        NativeModeLogin_0.enable = 1
                        NativeModeLogin_0.ip = adr("192.168.101.1")
                        NativeModeLogin_0 FUB NativeModeLogin()
                        IF(NativeModeLogin_0.status <> 65535) THEN
                                    IF(NativeModeLogin_0.status = 0) THEN
                                                NewLogEntry("Loggged into camera", ADR(logbook), 20)
                                                NativeModeLoggedIn      = TRUE
                                                NativeModeStatus        = VIS_ACTIVE
                                    ELSE
                                                NewLogEntry("Failed to log into camera", ADR(logbook), 20)
                                                vsErrorReset = VIS_ACTIVE
                                    ENDIF
                                    vkNativeModeConnect = FALSE
                        endif
            ELSE
(* ---------------------------------------------------------------------------------------------------- *)
(* Disconnect from camera
(* ---------------------------------------------------------------------------------------------------- *)
                        NativeModeLogout_0.enable = 1
                        NativeModeLogout_0.ident = NativeModeLogin_0.ident
                        NativeModeLogout_0 FUB NativeModeLogout()
                        IF(NativeModeLogout_0.status <> 65535) THEN
                                    NewLogEntry("Loggged out of camera", ADR(logbook), 20)
                                    NativeModeLogin_0.enable = 0
                                    NativeModeLogin_0 FUB NativeModeLogin()
                                    NativeModeStatus        = VIS_INACTIVE
                                    NativeModeLoggedIn      = FALSE
                                    vkNativeModeConnect     = FALSE
                        ENDIF
            ENDIF
ENDIF
```

1.3.4 NativeModeCommand

This function provides access to all commands that do not respond with large amounts of data (ex. file transfer). The function needs multiple cycles to execute.

Inputs

enable	The function is only executed when enable is = 1. If function responds with an error set enable = 0 and call the function once to reset the internal state machine.
ident	Reference from the NativeModeLogin command.
command	Native mode command in string format. (ex. GI for general information, see Cognex documentation for details)
out_data	Pointer to respond data, typically byte field or string. (Also see function NaticeModeSeperateData)
out_size	Size of respond buffer.

Outputs

in_size	Number of received bytes. (out_size is maximum size of data that can be received, this is the amount that actually was received)
status	Status of function. Call the function as long as status is BUSY (65535).

1.4 NativeModeSeperateData

Most commands will respond with a byte stream of information's that are separated with carriage return line feed. This function will automatically split the data and convert it into a string array.

Inputs

in_data	Pointer to respond data from NativeModeCommand.
in_size	Received bytes from NativeModeCommand.
out_string	Pointer to a string array where the converted data will be stored.
out_len	The size of a single string
out_size	The size of the complete string array

1.4.1 Sample Code

```
(* ---------------------------------------------------------------------------------------------------------- *)
(* Use command interface for Cognex camera                                                                    *)
(* ---------------------------------------------------------------------------------------------------------- *)
IF(NaticeModeCommand = CMD_INFO) THEN
            NativeModeCommand_0.command = ADR("GI")
ENDIF
IF(NaticeModeCommand = CMD_STATUS) THEN
            NativeModeCommand_0.command        = ADR("EV GetMSBuffer(0)")
ENDIF
IF(NaticeModeCommand = CMD_DELETE_JOBID3) THEN
            NativeModeCommand_0.command        = ADR("DJ3")
ENDIF
IF(NaticeModeCommand <> NONE) THEN
            NativeModeCommand_0.enable                    = 1
            NativeModeCommand_0.ident                     = NativeModeLogin_0.ident
            NativeModeCommand_0.out_data       = ADR(data)
            NativeModeCommand_0.out_size       = SIZEOF(data)
            NativeModeCommand_0 FUB NativeModeCommand()
            NativeModeStatus = VIS_INACTIVE
            IF(NativeModeCommand_0.status <> 65535) THEN
                        IF(NativeModeCommand_0.status = 0) THEN
                                    NewLogEntry("Command finished", ADR(logbook), 20)
                                    (* Separate data stream by \r\n and split into strings *)
                                    NativeModeCommandStatus = NativeModeSeparateData(ADR(data), NativeModeCommand_0.in_size, ADR(data_string),
SIZEOF(data_string[0]), SIZEOF(data_string)/SIZEOF(data_string[0]))
                                    (* Check camera responds (1 = The command was executed successfully. 0=Unrecognized command. -2=The
command could not be executed.*)
                                    IF(NativeModeCommandStatus = 1) THEN
                                                NativeModeStatus = VIS_ACTIVE
                                    ELSE
                                                NewLogEntry("Camera responded with error", ADR(logbook), 20)
                                                vsErrorReset = VIS_ACTIVE
                                    ENDIF
                        ELSE
                                    NewLogEntry("Command failed", ADR(logbook), 20)
                                    vsErrorReset = VIS_ACTIVE
                        ENDIF
                        NaticeModeCommand = NONE
            ENDIF
ENDIF
```

1.4.2 NativeModeReadFile

This function will transfer files from the camera to and save it on the PLC flash card. Files can be camera jobs or configuration data.

Inputs

enable	The function is only executed when enable is = 1. If function responds with an error set enable = 0 and call the function once to reset the internal state machine.
ident	Reference from the NativeModeLogin command.
command	Native mode command in string format. (ex. RJ1 for read job ID1, see Cognex documentation for details)
file_name	Pointer to file name where the data will be stored.
device	Pointer to file device string. See Automation Studio help for library FileIO.

Outputs

progress	Read file progress in percent.
status	Status of function. Call the function as long as status is BUSY (65535).

```
(* ---------------------------------------------------------------------------------------------------------------- *)
(* Read file from Cognex camera
(* ---------------------------------------------------------------------------------------------------------------- *)
IF EDGEPOS(vkTransferRead = READ_CFG) THEN
          NativeModeReadFile_0.command  = ADR("RS")
          NativeModeReadFile_0.file_name          = ADR("settings.dat")
          NewLogEntry("Read job file ID1", ADR(logbook), 20)
ENDIF
IF EDGEPOS(vkTransferRead = READ_ID1) THEN
          NativeModeReadFile_0.command  = ADR("RJ1")
          NativeModeReadFile_0.file_name          = ADR("JobFileID1.job")
          NewLogEntry("Read job file ID1", ADR(logbook), 20)
ENDIF
IF(vkTransferRead <> NONE) THEN
          NativeModeReadFile_0.enable   = 1
          NativeModeReadFile_0.ident              = NativeModeLogin_0.ident
          NativeModeReadFile_0.device   = ADR("CognexFiles")
          NativeModeReadFile_0 FUB NativeModeReadFile()
          vsTransferProgress            = NativeModeReadFile_0.progress
          NativeModeCancelStatus        = VIS_ACTIVE
          NativeModeStatus              = VIS_INACTIVE
          IF(NativeModeReadFile_0.status <> 65535) THEN
                    IF(NativeModeReadFile_0.status = 0) THEN
                              NewLogEntry("Load file finished", ADR(logbook), 20)
                              NativeModeStatus = VIS_ACTIVE
                    ELSE
                              NewLogEntry("Load file failed", ADR(logbook), 20)
                              vsErrorReset = VIS_ACTIVE
                    ENDIF
                    NativeModeCancelStatus = VIS_HIDDEN
                    vkTransferRead = NONE
          ENDIF
ENDIF
```

1.4.3 NativeModeWriteFile

This function will transfer files to the camera from the PLC flash card. Files can be camera jobs or configuration data. **The camera has to be in offline mode to use this function.**

Inputs

enable	The function is only executed when enable is = 1. If function responds with an error set enable = 0 and call the function once to reset the internal state machine.
ident	Reference from the NativeModeLogin command.
command	Native mode command in string format. (ex. WJ1 for qrite job ID1, see Cognex documentation for details)
file_name	Pointer to file name where the data is stored.
device	Pointer to file device string. See Automation Studio help for library FileIO.

Outputs

progress	Write file progress in percent.
status	Status of function. Call the function as long as status is BUSY (65535).

```
(* ---------------------------------------------------------------------------------------------------------------- *)
(* Write file to Cognex camera                                                                                      *)
(* ---------------------------------------------------------------------------------------------------------------- *)
IF EDGEPOS(vkTransferWrite = WRITE_ID1) THEN
          NativeModeWriteFile_0.command          = ADR("WJ1")
          NativeModeWriteFile_0.file_name = ADR("1_Powerlink.job")
          NewLogEntry("Write job file ID1", ADR(logbook), 20)
ENDIF
IF EDGEPOS(vkTransferWrite = WRITE_ID2) THEN
          NativeModeWriteFile_0.command          = ADR("WJ2")
          NativeModeWriteFile_0.file_name = ADR("2_Powerlink.job")
          NewLogEntry("Write job file ID2", ADR(logbook), 20)
ENDIF
IF EDGEPOS(vkTransferWrite = WRITE_ID3) THEN
          NativeModeWriteFile_0.command          = ADR("WJ3")
          NativeModeWriteFile_0.file_name = ADR("3_Powerlink.job")
          NewLogEntry("Write job file ID3", ADR(logbook), 20)
ENDIF
IF EDGEPOS(vkTransferWrite = WRITE_CFG) THEN
          NativeModeWriteFile_0.command          = ADR("WS")
          NativeModeWriteFile_0.file_name = ADR("settings.dat")
          NewLogEntry("Write configuration file", ADR(logbook), 20)
ENDIF
IF(vkTransferWrite <> NONE) THEN
          NativeModeWriteFile_0.enable  = 1
          NativeModeWriteFile_0.ident   = NativeModeLogin_0.ident
          NativeModeWriteFile_0.device  = ADR("CognexFiles")
          NativeModeWriteFile_0 FUB NativeModeWriteFile()
          vsTransferProgress            = NativeModeWriteFile_0.progress
          NativeModeCancelStatus        = VIS_ACTIVE
          NativeModeStatus              = VIS_INACTIVE
          IF(NativeModeWriteFile_0.status <> 65535) THEN
                    IF(NativeModeWriteFile_0.status = 0) THEN
                              NewLogEntry("Write file finished", ADR(logbook), 20)
                              NativeModeStatus = VIS_ACTIVE
                    ELSE
                              NewLogEntry("Write file failed", ADR(logbook), 20)
                              vsErrorReset = VIS_ACTIVE
                    ENDIF
                    NativeModeCancelStatus = VIS_HIDDEN
                    vkTransferWrite = NONE
          ENDIF
ENDIF
```

2 Error numbers

ERR_NO_NODE_NUM	10000	No Powerlink node number specified
ERR_NO_IDENT	10001	No ident specified
ERR_NO_CMD	10002	Command string is not specified
ERR_DUPLICATE_CMD	10003	Multiple commands called at the same time
ERR_SIZE_CMD	10004	Command exceeds maximum size of 100 bytes
ERR_RESP_TIMEOUT	10010	Cognex camera did not respond to request
ERR_RESP_UNEXPECTED	10011	Respond is not expected format
ERR_RESP_HEADER	10012	Response header is too small
ERR_RESP_DATA	10020	Respond does not contain status
ERR_CAM_CMD_UNKNOWN	10030	Unrecognized command
ERR_CAM_NAME_MISS	10031	The job filename is missing
ERR_CAM_NAME_UNKNOWN	10032	There is no job saved with the given name
ERR_CAM_ACCESS	10033	User does not have Full Access to execute the command
ERR_CAM_GENERAL	10039	General error message from camera
ERR_FILE_SIZE	10040	File exceeds expected file size
ERR_FILE_CRC	10041	The file CRC does not match the data transmitted

3 Version History

V1.0

- First release