EXOR
Tech-note

CANopen HMI profile for UniOP

This Technical Note contains all the information required to connect UniOP panels as devices in a CANopen network and to take advantage from the advanced data access features supported by the UniOP CANopen HMI communication driver.
The communication driver implements a custom profile developed to support HMI applications, based on PDO communication and a dedicated application layer.

***Important**: this Technical Note applies to the CANopen communication driver identified by the name 'CANopen HMI' and included in the Designer files D32Uplc200.DLL and D32Uplc200.ini. Both files must be present in the Designer installation directory. This communication driver requires an operator panel of hardware type –0045 or newer equipped with a communication module type TCM09.*

Contents

Tn206
Ver. 1.01

www.exor-rd.com

1 Introduction

Using the UniOP “CANopen HMI” communication driver you can add one or more UniOP operator panels as devices in a CANopen network.
A new device communication profile has been developed for UniOP that takes advantage from the advanced user interface features of the UniOP products, while retaining the simple networking concept supported by the CANopen network.
The basic idea is create a client/server communication structure where the HMI is the client and the CANopen controller is the server.
This Technical Note will describe in detail how to implement this connection.

2 UniOP CANopen HMI Profile

In this communication model the HMI initiates the communication sessions, acting as a source of messages.
The basic messages are PDO messages with the standard size of 8 bytes.
The COB-ID of the messages is defined in a way that makes clear, from the well-known CANopen rules, what is the target of the PDO message.
The format of the PDO message has been defined according to a custom application layer protocol.
This application layer protocol defines a device-independent communication profile optimized for HMI applications.
When the CANopen master controller receives the PDO message, it will interpret its contents and produce a PDO message with the response addressed to the HMI panel.

The definition of this client/server relationship is independent of the CANopen Master in the sense that it can easily be supported in any particular CANopen master system. The resulting solution is easily portable to any CANopen master.
UniOP Designer offers a user interface that adapts itself to show the typical addressing model of CANopen master controller where the panel is going to be connected.
Adapting to different masters is possible using a profile customization file that may contain data definitions for different controller types.
The profile customization file is called D32Uplc200.INI and must be present in the Designer directory.

2.1 Profile details

This chapter provides the specification of the HMI profile and describes the subset of the request/response formats used by this implementation of the protocol.

UniOP generates PDO messages initiating communication request sessions as soon as the UniOP firmware requires data from the protocol.
The panel is using the first transmit PDO identified by the COB-ID 0x180 combined with the Node Number assigned to the panel.

The communication profile uses only one transmit PDO and one receive PDO; the limited number of bytes available in standard PDO message maybe limiting, in some cases, the driver capabilities especially in terms of performance.

2.1.1 Request format: HMI to Controller (Transmit PDO)

The PDO message delivered by UniOP is formatted according to Table 1.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Offset Low	Offset High	Data 0	Data 1	Data 2	Data 3	Data Length and Job Number	Operation Type and Controller ID

Table 1

The request frame includes the following elements:

Offset Low Low byte of the offset (16 bits address) for the requested block of data

Offset High High byte of the offset (16 bits address) for the requested block of data

Data 0 … Data 3 Data for Write Operations; not used in Read Operations

Data Length and Job Number Contains:
- number of requested bytes
- job Number indicator;

Table 2 shows the details of the bit assignment

Operation Type and Controller ID Contains:
- type of operation requested
- the Controller ID that identifies the target of the message;

Table 3 shows the details of the bit assignment

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data Length [1]	Data Length [0]	Job Number [5]	Job Number [4]	Job Number [3]	Job Number [2]	Job Number [1]	Job Number [0]

Table 2

The “Data Length” parameter is coded in 2 bits and takes values between 1 and 4 according to the following rules:

00	1 bytes
01	2 bytes
10	3 bytes
11	4 bytes

Note that the elementary size of each data item depends on the Controller memory organization.

The “Job Number” occupies 6 bits and can have values between 0 and 63; the “Job number” parameter is placed as last element in the PDO to ensure data consistency; the PLC program running the controller should constantly monitor the value of the “Job Number” parameter and consider the received message as valid only when detecting a change in the value of the “Job Number” field. “Job Number” is automatically increased at each new communication session (new request frame).

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Operation Type	Controller ID [6]	Controller ID [5]	Controller ID [4]	Controller ID [3]	Controller ID [2]	Controller ID [1]	Controller ID [0]

Table 3

The “Operation Type” uses one bit with the following definition:

0	Read	data is transferred from controller to UniOP
1	Write	data is transferred from UniOP to controller

The “Controller ID” uses 6 bits; it represents the Node Number in the CANopen network of the master controller addressed by the current request.
This parameter is required in case the CAN network has more than one master controller; the CANopen standard defines in fact the COB-ID of the messages in a way that all the partners of the bus known the originator. In case more than one master device is present in the same network, the “Controller ID” field will specify the target of each individual request message. Only the master controller that recognizes in this field its own Node ID will consider the message and process the PDO contents.

2.1.2 Response Format: Controller to Panel (Receive PDO)

The PDO message returned to UniOP by the controller must be formatted as defined in Table 4.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Status Flag / Error Code	Dummy – Always 0	Data 0	Data 1	Data 2	Data 3	Data Length and Job Number	Operation Type and Controller ID

Table 4

The request frame consist of the following elements:

Status Flag / Error Code	Contains the information related to the execution of the operation type of the request; Table 5 shows the coding information
Data 0 … Data 3	Contain the data information returned to the panel in response to a Read request
Data Length and Job Number	It is the copy of the corresponding field of the request frame
Operation Type and Controller ID	It is the copy of the corresponding field of the request frame

	Status Flag / Error Code	
Operation Type in the Request Frame	**No Errors**	**Error**
Read	0x01	0x81
Write	0x02	0x82

Table 5

3 The .ini File

The D32Uplc200.DLL file requires a special initialization file for proper operation. The file must be present in the Designer installation directory.

Note: *If the initialization file is missing, the communication driver "CANopen HMI" will not be listed in the list of available drivers.*

The D32Uplc200.INI file is divided into a number of sections where specific controller models and the corresponding data can be configured.

The file is designed to ensure the maximum possible level of customization of the Designer user interface depending on the controller model and on its internal memory organization.

Each controller model has its how section in the .INI file marked with [ModelInfoX] where "X" is the model number that must be sequential for any new added model. The driver supports up to 10 controller models.

The user can define for each model the list of available Data Types and related Data Formats in the [ModelInfo] session.

The [ModelInfo] session contains the following entries:

ModelString Controller Model name to be displayed in the Designer "Controller Setup" dialog

DataTypeString Specifies the string to be displayed in the "Field Reference" dialog box; the string can be followed by a semicolon ";" and a switch that identifies the controller memory organization type according to the following list:

1 BYTE organization
2 WORD organization
4 DWORD organization

In case no value or an invalid value is specified, the communication driver will assume to work with a BYTE organized memory.
Current version of the driver supports ONLY ONE Data Type per each controller model

DataTypeShortString Specifies the string to be displayed in the "Address Reference" field of the "Data Field Properties" dialog box; this parameter can be followed by a semicolon ";" and an additional Boolean switch specifying if the string in the "Address Reference" should contain also the short name of the selected data type as specified later in the file; the Boolean switch has the following meaning:

FALSE The short string will always be displayed as specified later in the file

TRUE The short name string will be displayed in the extended format according to the following list:

BIT	No suffix is added to the Short Name String
BYTE	"B" suffix is added for Byte Data Type
WORD	"W" suffix is added for Word Data Type
DWORD	"D" suffix is added for Double Word Data Type

CustomizeUIOffset Specifies the available offsets in the user interface when using the spin buttons to adjust the reference:

FALSE	All the offsets are available
TRUE	Only a subset of the offsets are available depending on the selected Data Type; if for instance a Controller has memory organized in Bytes and the selected Data Format is Word, the available range of offset will be: 0, 2, 4, 6, …

4 Configuring Designer

Once the INI file has been properly configured, you will have to select the PLC model from the list presented by the Designer software in the Controller Setup dialog box as shown for example in Figure 1.

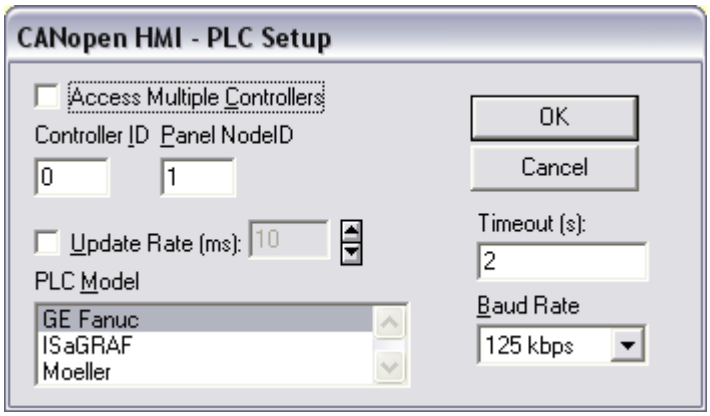

Figure 1

The “Controller Setup” dialog box provides several controls for the setup of the communication driver.

Access Multiple Controllers	Enables the possibility to configure the driver for access to more than one Controller
Controller ID	CANopen Node ID assigned to the Master device
Panel NodeID	CANopen Node ID to be assigned to the panel
Update Rate (ms)	Minimum interval time between two requests; it can be useful when the bus load needs to be properly controller and limited
Timeout (s)	Maximum allowed time the driver will wait for a response from the PLC before reporting a communication error
Baud Rate	Speed of the CANopen network
PLC Model	Mmodel of the Master controller

5 Connecting UniOP to GE Fanuc Controllers

This chapter describes all the steps you have to follow in order to establish a successful connection between UniOP and GE Fanuc CANopen master controller.
Functionality has been tested with the CANopen master module HE693CAN501D from Horner, used with a GE Fanuc SERIES 90-30 CPU 363 or higher.
The PLC support program has been developed with Versa-Pro programming software version 2.03.

The UniOP communication driver can access ONLY ONE data type; for GE Fanuc controllers the Register (%R) memory area has been chosen.

5.1 PLC Function Block Call

The server function running in the PLC program has been designed in the form of Function Block called “UnCanCl”, written using the “ST” programming language. Proper working example is available on demand.

The PLC program requires a reserved memory area from Register %R9948 to %R9999; this area MUST NOT be reserved to the UniOP support program and MUST NOT be addressed by the user’s application.
These registers contain the function block parameters and the internal temporary variables used by the function block.
The Function Block parameters are the following:

CANMasterID	**%R9948**	CANopen Master Node number; the ID number must be in the high byte of the word
CANSlaveID	**%R9949**	UniOP CANopen Slave ID matching the Controller Setup setting of the Designer project; the ID number must be written in the low byte of the word
INOffset	**%R9950**	Offset in the PLC memory where the PDO message received from the panel is mapped
OutOffset	**%R9951**	Offset in the PLC memory where the PDO message to be sent to the panel is mapped
RegLimitLow	**%R9952**	Lower limit of the PLC memory addressable (visible) by UniOP
RegLimitHigh	**%R9953**	Upper limit of the PLC memory addressable (visible) by UniOP

The registers %9937 and %9938 are used as pointers in the Function Block and must not be used.

The above parameters must be defined before the CAL instruction.

Considering for example the following parameters:
CanMasterID: 2
CanSlaveID: 3
InOffset: 9
OutOffset: 9
RegLimiLow 1
RegLimitHigh: 9947
the PLC Function Block call should be prefixed with the following assignments:

```
LD_INT      16#0200     // Master ID in high byte in the word
ST_INT      %R9948
LD_INT      3           // Slave ID
ST_INT      %R9949
LD_INT      9           // Input offset
ST_INT      %R9950
LD_INT      9           // Output offset
ST_INT      %R9951
LD_INT      1           // Low limit of Register visibility
ST_INT      %R9952
LD_INT      9947        // High limit of Register visibility
ST_INT      %R9953
```

```
CAL   UnCanCl          // Call to function block
```

The PLC Function block support the use of more than one panel simply repeating the call of the same function for all the additional units specifying before each call the proper calling parameters.

5.2 PLC Configuration

The CANopen master requires a dedicated hardware configuration. Please refer to Horner documentation for all related details.
The board configuration is contained in text file identified by the extension. RPT. the text file should be produced according to Horner instruction related to the syntax and transferred to the Master Controller via serial line.

We provide following only some basic information that should be used when compiling the UniOP part of the .RPT configuration file.

1. The .RPT configuration file contains basically the mapping f the Receive and Transmit PDO
2. UniOP should be mapped in the first PDO channel dedicated to real time operation
3. The Receive PDO (transmitted from UniOP) corresponds to the Object dictionary 0x1600 + NodeID, where NodeID is the Node Number assigned to the UniOP panel
4. The Transmit PDO (received by UniOP) corresponds to the Object dictionary 0x1A00 + NodeID, where NodeID is the Node Number assigned to the UniOP panel
5. The .RPT file should contain the information related to the CPU registers used to store incoming and outgoing PDOs

The PLC support program available as example includes already proper configuration for the PLC and the CANopen card configuration file for one or two UniOP panels.

Appendix A. Communication Error Codes

Current communication status is displayed on the system page of the UniOP. Beside the string, describing current state of the communication (OFF, ON, ERR), there is an additional error code representing the last (which may be not the current one) error encountered. The codes are:

Code	Description	Notes
00	No error	There are no communication errors and there have been no errors since start-up.
04	PLC address error	UniOP is requesting a register out of the addressable range
05	Timeout (receiving)	The PLC did not respond to the request within the timeout interval
06	Response error	Unexpected response format in application layer
07	PLC operation error	The PLC did not accept the execution of the last operation; the operation code in the response frame does not match the one specified in the request frame