Section		Page
1	**Presentation**	7
	1.1 Purpose of the TP UNI-TE Software	8
	1.2 Product Composition	10
	1.3 Operation Principle	11
	1.3-1 Definition of Terms and Features	11
	1.3-2 Operation	12
	1.3-3 Additional Operation Information for OS/2	13
	1.4 Description of the Main Functions	14
	1.4-1 Communication Context Management	14
	1.4-2 Device Management	15
	1.4-3 Access to Data	15
	1.4-4 Unsolicited Data	15
	1.4-5 Generic Function	15
	1.4-6 Response Retrieval	15
	1.5 Function Sumary Tables	16
2	**Installation**	19
	2.1 Installation under DOS	20
	2.2 Installation under OS/2	21
	2.3 Installation under WINDOWS	23
3	**Operation under DOS**	25
	3.1 Developing the Application under DOS	26
	3.2. Compiling and Linkage Editing	26

Section **Page**

3.3 Debugging and Operation 26

3.4 Application Example under DOS 27
3.4-1 Program Description 27
3.4-2 "Example.C" Source Program 28

4 Operation under OS/2 31

4.1 Developing the Application under OS/2 32

4.2. Compiling and Linkage Editing 32

4.3. Installing the Driver and Connecting 32

4.4 Debugging and Operation 33

4.5 Application Example under OS/2 33
4.5-1 Layout Corresponding to the Example 34
4.5-2 Program Description 34
4.5-3 "Example.C" Source Program 35

5 Operation under WINDOWS 39

5.1 Developing the Application under Windows 40

5.2. Compiling and Linkage Editing 40

5.3. Debugging and Operation 40

5.4 Application Example under Windows 40
5.4-1 Program Description 41
5.4-2 "Example1.C" Source Program 41

Section		Page
6	**Function Syntax**	47
6.1	Functions: Managing the Communication Context	49
	6.1-1 UNITEInitDriver	49
	6.1-2 UNITECloseDriver	50
	6.1-3 UNITEOpenConnection	51
	6.1-4 UNITECloseConnection	52
	6.1-5 UNITEOpenUnsolicitedData	53
	6.1-6 UNITECloseUnsolicitedData	54
	6.1-7 UNITETimeOut	55
	6.1-8 UNITEGetSystemError (OS/2 only)	56
	6.1-9 UNITEGetUsExtInfo (WINDOWS only)	56
6.2	Functions: Managing Devices	57
	6.2-1 UNITEMirror	57
	6.2-2 UNITEIdentification	58
	6.2-3 UNITEReserve	59
	6.2-4 UNITERelease	60
	6.2-5 UNITEIAmAlive	61
	6.2-6 UNITERun	62
	6.2-7 UNITEStop	63
6.3	Functions: Accessing Variables	64
	6.3-1 UNITEReadInternalBit	64
	6.3-2 UNITEReadSystemBit	65
	6.3-3 UNITEWriteInternalBit	66
	6.3-4 UNITEWriteSystemBit	67
	6.3-6 UNITEReadInternalWord	68
	6.3-6 UNITEReadConstantWord	69
	6.3-7 UNITEReadSystemWord	70
	6.3-8 UNITEWriteInternalWord	71
	6.3-9 UNITEWriteSystemWord	72
	6.3-10 UNITEReadCommonWord	73
	6.3-11 UNITEWriteCommonWord	74
	6.3-12 UNITEReadInternalDWord	75
	6.3-13 UNITEReadConstantDWord	76
	6.3-14 UNITEWriteInternalDWord	77

Section		Page
	6.3-15 UNITEReadObject	78
	6.3-16 UNITEWriteObject	79
	6.3-17 UNITEReadWordArray	80
	6.3-18 UNITEWriteWordArray	81
6.4	Functions: Domain Management	82
	6.4-1 UNITETransferTsxPC	82
	6.4-2 UNITETransferPCTsx	84
6.5	Function: Unsolicited Data	86
	6.5-1 UNITEUnsolicitedData DOS	86
	6.5-2 UNITEUnsolicitedData OS/2	87
6.6	Function: Generic Function	88
	6.6-1 UNITERequest	88
6.7	Function: Response Retrieval	90
	6.7-1 UNITEResponse	90
	6.7-2 UNITEResponseMult (OS/2 only)	91
7	**Appendix**	**93**
7.1	Glossary	94
7.2	List of Error Codes under DOS and Windows	95
	7.2-1 Error Codes Common to all Functions	95
	7.2-2 Error Codes Relative to File Transfer Functions	96
	7.2-3 Error Codes Relative to the usExtinfo Variable	97
7.3	List of Error Codes under OS/2	98
	7.3-1 Error Codes Common to all Functions	98
	7.3-2 Error Codes Relative to File Transfer Functions	98
7.4	UNI-TE Requests	99
	7.4-1 Read Objects	99
	7.4-2 Write Objects	104
	7.4-3 Application Initialization	107

Contents of the User's Manual

Section 1: **Presentation**

The use and composition of the product, the list of available functions, the principle of use of a function, the description of the main functions, the characteristics and performance level.

Section 2: **Installation**

The operating environment required, the installation procedure, the organization of the resulting files, how to modify configuration parameters.

Sections 3, 4 and 5: **Operation**

The various development steps for an application that uses TP UNI-TE library services, an example of an application that can be used to test connection with a remote device.

Section 6: **Function Syntax**

The syntax to follow to call-up a function.

Section 7: **Appendix**

Glossary of terms used, list of error codes.

Intended users

This manual is intended for C++ language application developers.

Prerequisites

The user is assumed to have prior knowledge of:
- C language,
- DOS or OS/2 operating systems
- UNI-TE objects used in the devices

Related Telemecanique documentation

TSX DM FPP E	: FIPIO Bus Interface for TSX/PMX Model 40 V6 level PLCs User's Manual.
TSX DM PCX V52E	: PL7-3 Coprocessor User's Manual.

Section		Page
1.1	**Purpose of the TP UNI-TE Software**	8
1.2	**Product Composition**	10
1.3	**Operation Principle**	11
1.3-1	Definition of Terms and Features	11
1.3-2	Operation	12
1.3-3	Additional Operation Information for OS/2	13
1.4	**Description of the Main Functions**	14
1.4-1	Communication Context Management	14
1.4-2	Device Management	15
1.4-3	Access to Data	15
1.4-4	Unsolicited Data	15
1.4-5	Generic Function	15
1.4-6	Response Retrieval	15
1.5	**Function Summary Tables**	16

1.1 Purpose of the TP UNI-TE Software

The TP UNI-TE software allows the connection of PC compatible microcomputers running under DOS, OS/2 or Windows operating systems equipped with a PL7-3 coprocessor.

It allows simple and reliable implementation of communication between a PC and Telemecanique automation control systems without requiring any specialist knowledge of these systems.

Integration of applications into the X-WAY communication layout

The purpose of ISO/OSI standardized communication networks is to allow the creation of system layouts combining devices of various types and functions, from production planning and management, automation systems such as PLCs or numerical controlled machine tools and robots to man-machine interface systems, etc.

The X-WAY communication layout, which complies with the OSI model, comprises a set of Industrial Local Area Networks used to set-up distributed automation structures, for the simplest to the most complex applications.

The TP UNI-TE software package is used with microcomputers that execute, for example:

- Standalone production or data processing applications,
- Gateway applications between automated control systems and production management and planning systems,
- etc.

Reminders on UNI-TE:

The Telemecanique's industrial messaging system supported by the X-WAY communication layout is called UNI-TE.
The UNI-TE protocol works on a Client/Server, Request/Response basis (services confirmed).
A device supporting the UNI-TE protocol can be a:

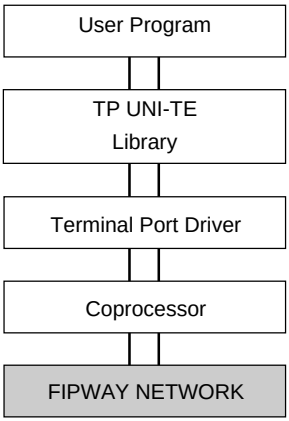
- CLIENT: the device initiates the communication with another SERVER device, it collects (read mode) or transmits (write mode) information or sends out an order (Run, Stop),
- SERVER: the device serves the CLIENT as required and sends a post-execution response.

A device may send a message on its own initiative, to another device without the message being confirmed. The data sent out is called unsolicited data.

The microcomputer equipped with the TP UNI-TE software is considered a UNI-TE CLIENT device.

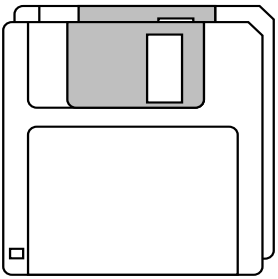
The TP UNI-TE software is a "library" of functions written in C. It allows user programs to easily access communication as no specific knowledge is required for implementing the various software layers it comprises.

This library is inserted between the user program and the TSX PC Driver, whose communication capabilities it uses.

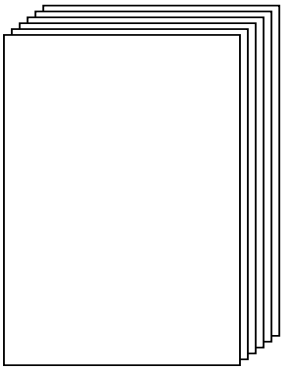


The main services provided by TP UNI-TE are:
- Management of the communication context,
- Management of the devices,
- Access to the variables,
- Reception of unsolicited data,
- Domain management,
- Generic function,
- Response retrieval.

1.2 Product Composition



Two 3 1/2" diskettes



This documentation package and the licence

The TP UNI-TE software supplied was developed with Microsoft Visual C++ language.

1.3 Operation Principle

1.3-1 Definition of Terms and Features

A few terms should be defined to assist in understanding the library operation principles.

UNI-TE Client: A device able to send out UNI-TE requests.

UNI-TE Server: A device responding to service requests from a client device.

The user program is a UNI-TE client, i.e. it accesses, at its own initiative, UNI-TE server devices on the network, using the library services listed in the tables in Section 1.5.

Due to the CLIENT/SERVER organization, installing two licensed copies of TP UNI-TE software on two different PCs does not make intercommunication possible.

5-LEVEL XWAY ADDRESSING: In the Telemecanique addressing system, the source and target addresses are encoded in 5 bytes:

- **network:** network number,
- **station:** device number on the network,
- **gate:** identification of the station's logical entities.
  If the device is a programmable controller and the gate number is equal to 5:
- **module:** physical location of the interface in the PLC,
- **Channel:** number of a logical device or address of a UNI-TELWAY slave + 100.

To access the UNI-TE server in a PL7-3 coprocessor:

(network) = 0
(station) = 254
(gate) = 0
(module) = 0
(channel) = 0

To access a TxT block n in the coprocessor PL7-3 application:

(network) = 0
(station) = 254
(gate) = 16 + 11
(module) = 0
(channel) = 0

To gain access to one or more remote devices it is necessary to open one or more communication channels:

Communication channel: a resource for sending out function-related requests to a remote device.

The TP UNI-TE program can handle up to 3 communication channels.

A communication channel will send out requests to one server device only.

Several channels can be opened for communication with a single device.

A request cannot be sent out through a communication channel until the response to the previous request is received.

Several requests may be sent out simultaneously through different channels.

1.3-2 Operation

To communicate with a device from any PC application the user program should follow the steps listed below:

1 . Initialize the communication context:
Using the *UNITEInitdriver* function

2 . Open one or more channels for communication to one or more devices.
Using the *UNITEOpenConnection* function which returns a *hEquip* channel number.

3 . Control the following sequence for each access made:
- Using a *UNITE_xx_xx* function specifying the dedicated channel (hEquip), the response storage location to use, and the usable data corresponding to the function.
- The function returns a Request identifier *hReq.*
- Continue processing.
- Send out the response request related to the previously used function, using the UNITEResponse function with the *hReq* identifier.
- If the data is ready, the function returns the response OK which means that the data stored at the predefined address is valid.

On completion of the connection sequence:

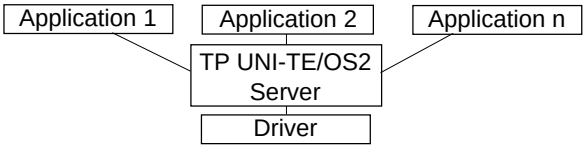
4 . Close the communication channel(s).
Using the *UNITECloseConnection* function.

5 . Release the communication context:
Using the *UNITECloseDriver* function.

The user program can receive unsolicited data. A preset communication channel (gate 3) can be opened in receive mode to accept this type of data.
The application will send unsolicited data by initializing a TERminal type text block (TXT) with its Txti,t parameter set to 3 by the application program.

1.3-3 Additional Operating Information for OS/2

OS/2 is a multi-task operating system allowing several application programs to access library services asynchronously.



1.4 Description of the Main Functions

1.4-1 Communication Context Management

The communication context management functions allow TP UNI-TE software to be used from one or more applications.

UNITEInitDriver:	Initialize the communication context
UNITECloseDriver:	Release the communication context

Both of these functions should be activated for opening (***UNITEInitDriver***) and closing (***UNITECloseDriver****)* each communication application.

UNITEOpenConnection:	Open a communication channel
UNITECloseConnection:	Close a communication channel

Prior to sending out requests to a remote device, one or more communication channel must be opened.
More than one channel can be opened for a single device.
Only one function per channel may be activated at one time.
Close the channel when it is no longer used.

UNITEOpenUnsolicitedData:	Open the unsolicited data reception channel
UNITECloseUnsolicitedData:	Close the unsolicited data reception channel

Prior to receiving unsolicited data from a remote device, a reception channel must be opened. Devices sending unsolicited data to the PC should use the following XWAY target address: Default Network and Station address (0: Net, 254: station), Gate equal to 3, Module and Channel set to 0 by TER LOCAL type text block (TXT), with Txti,t = 3 in the program and Module and Channel set to 0.

UNITETimeOut:	Sets the maximum time-out value

Each time the TP UNI-TE software sends a request to a remote device it initiates a time-out of the defined duration.
If the ***UNITEResponse*** function is activated in the application program and the time-out expires before a response is received, then UNITEResponse returns a TIME_OUT error code.

1.4-2 Device Management

The functions available allow the user to test the communication path used to a remote device, to obtain identification of a remote device, to start or stop it, to control the reservation-release mechanisms, etc.

1.4-3 Access to Data

These functions allow the read and write access to the UNI-TE objects located in a remote device.

1.4-4 Unsolicited Data

This function allows the user to retrieve unsolicited data sent by a remote device or the coprocessor.

1.4-5 Generic Function

This function allows the user to generate any UNI-TE request.

1.4-6 Response Retrieval

The ***UNITEResponse*** function is used for retrieving the response to a previously sent request.
The response may be polled or awaited with an unlimited time-out.

1.5 Function Summary Tables

The following tables summarize the functions available, arranged by type of service supported.

FUNCTION NAME	COMMUNICATION CONTEXT MANAGEMENT
UNITEInitDriver	Initializes the communication context
UNITECloseDriver	Releases the communication context
UNITEOpenConnection	Opens a communication channel
UNITECloseConnection	Closes a communication channel
UNITEOpenUnsolicitedData	Opens the unsolicited data reception channel
UNITECloseUnsolicitedData	Closes the unsolicited data reception channel
UNITETimeOut	Sets the maximum time-out
UNITEGetSystemError (OS/2 only)	Retrieves an OS/2 system error

FUNCTION NAME	DEVICE MANAGEMENT
UNITEMirror	Tests device connection
UNITEIdentification	Identifies a device
UNITEReserve	Reserves a device
UNITERelease	Releases device reservation
UNITEIAmAlive	Maintains reservation
UNITERun	Starts a device
UNITEStop	Stops a device

FUNCTION NAME	ACCESS TO VARIABLES
UNITEReadInternalBit	Read internal bit type variable
UNITEReadSystemBit	Read system bit type variable
UNITEWriteInternalBit	Write internal bit type variable
UNITEWriteSystemBit	Write system bit type variable
UNITEReadInternalWord	Read internal word type variable
UNITEReadConstantWord	Read constant word type variable
UNITEReadSystemWord	Read system word type variable
UNITEWriteInternalWord	Write internal word type variable
UNITEWriteSystemWord	Write system word type variable
UNITEReadCommonWord	Read common word type variable
UNITEWriteCommonWord	Write common word type variable
UNITEReadInternalDWord	Read internal double word type variable
UNITEReadConstantDWord	Read constant double word type variable
UNITEWriteInternalDWord	Write internal double word type variable
UNITEReadObject	Read object type variable
UNITEWriteObject	Write object type variable
UNITEReadWordArray	Read word array type variable
UNITEWriteWordArray	Write word array type variable

FUNCTION NAME	DOMAIN MANAGEMENT
UNITETransferTsxPC	Remote downloading of the PLC program
UNITETransferPCTsx	Remote uploading of the PLC program

FUNCTION NAME	UNSOLICITED DATA
UNITEUnsolicitedData	Retrieves unsolicited data

FUNCTION NAME	GENERIC FUNCTION
UNITERequest	Function used for sending any type of UNI-TE request

FUNCTION NAME	RESPONSE RETRIEVAL
UNITEResponse	Retrieves the response to a request
UNITEResponseMult (OS/2 only)	Retrieves the first response from a list of requests sent

FUNCTIONS	VALUES
Number of communication channels	3
Number of simultaneous requests	3
Number of PCX boards in the PC	1
Number of unsolicited data	1

Section		Page
2.1	**Installation under DOS**	20
2.2	**Installation under OS/2**	21
2.3	**Installation under WINDOWS**	23

2.1 Installation under DOS

Procedure

Prior to installing TP UNI-TE software on the hard disk, it is recommended that the user:

- Read the licence and guarantee certificate that details the restrictions that apply when copying and installing the software.
- Make a back-up copy of the diskette required for installing the software to avoid any accidental damage to the original.

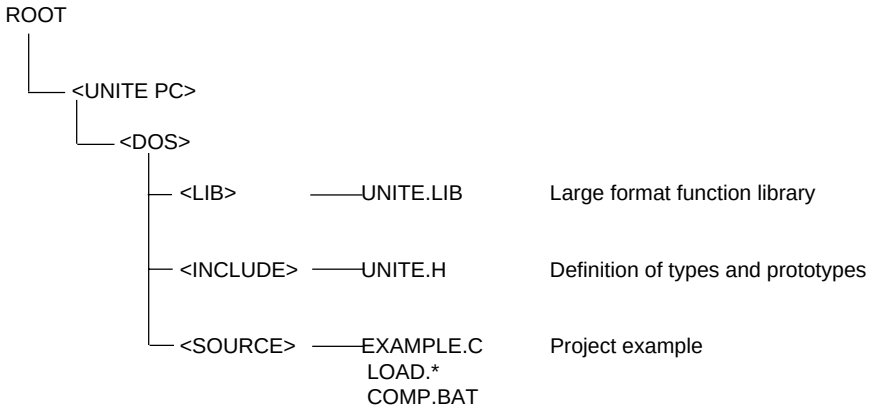
The installation is performed following the steps listed below (the commands to enter from the keyboard are shown in bold face and italic type):

1. Insert the diskette containing the TP UNI-TE library software into drive A: on a microcomputer running under DOS,
2. Type the command: ***a:*** then press <Return>,
   [a:\] is displayed on-screen.
3. The "install destdrive" command allows the library files to be installed on the selected target disk.
   For example, to install the files on disk C, type the command:
   install c, then press <Return>.

   The modifications shown below should be made to the C:\AUTOEXEC.BAT file:

   SET INCLUDE = (... existing configuration ...); Drive:\UNITEPC\DOS\INCLUDE
   SET LIB = (... existing configuration ...); Drive:\UNITEPC\DOS\LIB
4. Restart the system.

Organization of the resulting files



2.2 Installation under OS/2

Required environment
A PC compatible ISA (or EISA) bus microcomputer with a 386 or better processor running under OS/2 version 2.1.

Procedure
Prior to installing TP UNI-TE software on the hard disk, it is recommended that the user:
- Read the licence and guarantee certificate that details the restrictions that apply when copying and installing the software.
- Make a back-up copy of the diskette required for installing the software to avoid any accidental damage to the original.

The installation is performed following the steps listed below (the commands to enter from the keyboard are shown in bold face and italic type):

1. Insert the diskette containing the TP UNI-TE/OS2 library into drive A:

2. Open an OS/2 full screen window

3. Type the command: ***a:*** then press <Return>,
[a:\] is displayed on-screen.

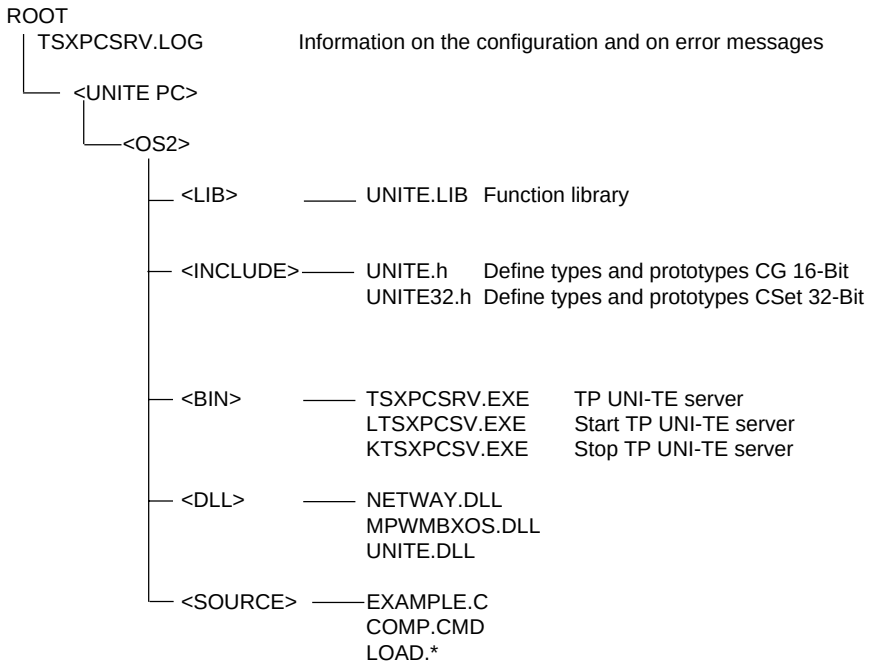
4. The "install destdrive" command allows the library files to be installed on the selected target disk.
For example, to install the files on disk C, type the command:
install c, then press <Return>.

The following modifications should be made to the C:\CONFIG.SYS file:
IOPL = yes
LIBPATH = (... existing configuration ...); Drive:\UNITEPC\OS2\DLL
SET PATH = (... existing configuration ...); Drive:\UNITEPC\OS2\BIN
SET INCLUDE = (... existing configuration ...); Drive:\UNITEPC\OS2\INCLUDE
SET LIB = (... existing configuration ...); Drive\UNITEPC\OS2\LIB

5. Restart the system. The program configuration parameters take the following default values:
NP: number of OS/2 applications that can be handled: 32
NU: number of unsolicited data elements buffered: 10
To change these parameters, refer to the "Modifying configuration parameters" heading on page 22.

6. Double-click on the "UNI-TE PCX OS/2 Library" program group, then double-click on the "Run UNI-TE PCX OS/2 Server" icon.

Organizing the resulting files



Modifying configuration parameters

For example to manage 50 communication channels and two interface boards.

1. Double-click on the "Stop UNITE PCX OS/2 server" icon,
2. Click with the right mouse button on the "Start UNITE PCX OS/2 server" icon,
3. Click on the arrow in the "Open" item,
4. Click in the box at the top left of the "Start UNITE PCX OS/2 Server - Parameter" window,
5. Click on the "Stop/Close" item,
6. Double-click on the "Start UNITE PCX OS/2 Server" icon.

2.3 Installation under WINDOWS

Required environment
A PC compatible microcomputer with ISA (or EISA) bus and 386 or better processor, running under WINDOWS 3.1x or WINDOWS 95.

Procedure
Prior to installing TP UNI-TE software on the hard disk, it is recommended that the user:
- Read the licence and guarantee certificate that details the restrictions that apply when copying and installing the software.
- Make a back-up copy of the diskette required for installing the software to avoid any accidental damage to the original.

The installation is performed following the steps listed below (the commands to enter from the keyboard are shown in bold face and italic type) :

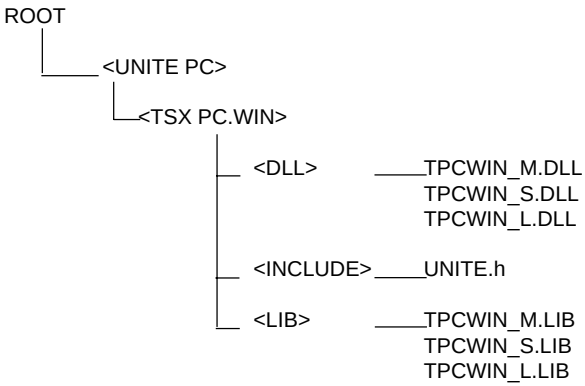
1. Get into the WINDOWS environment.

2. Insert the diskette containing the TP UNI-TE/WINDOWS library into drive A.

3. Then under WINDOWS, EXECUTE ***a: setup*** and follow the procedure.

4. If the TERMINAL PORT driver is not already installed, insert the TSX LF TP diskette, execute ***a: setup,*** then restart the PC.

Organization of the resulting files



Section		Page
3.1	**Developing the Application under DOS**	26
3.2.	**Compiling and Linkage Editing**	26
3.3	**Debugging and Operation**	26
3.4	**Application Example under DOS**	27
	3.4-1 Program Description	27
	3.4-2 "Example.C" Source Program	28

3.1 Developing the Application under DOS

The application should be written in C or any language accepting routines written in Microsoft C.

It is possible to use Microsoft Visual C++.

Each service corresponds to a "function" in the C language sense. Using a service amounts to a simple call with argument passing.

The user program organizes the management and consistency of the various services required.

3.2 Compiling and Linkage Editing

Any C program using the TP UNI-TE/DOS library must close the "include" list with:

 # include <UNITE.h>

The application must be compiled in LARGE format.

3.3 Debugging and Operation

Test the link to the remote SERVER device(s) by validating only a simple portion of the application:

- Use the *UNITEMirror* function

or

- Use the example described in the next section.

3.4 Application Example under DOS

The "EXAMPLE.C" application provided with the software package and described in this Section can be used for testing the installation and the correct operation of the data link to any remote TSX7 PLC SERVER device.
To do this, simply change the address (*UNITEOpenConnection*) and make it correspond to that of the device to access.
As this application is provided in the form of a "source" file, it requires compiling prior to use.
A compilation/linkage editor command file is provided with the example, type the following command to obtain an executable object file:
comp example for compiling in Large format.

3.4-1 Program Description

The program performs the following operations:
- Initialization of the communication environment,
- Opening a communication channel with the coprocessor,
- Reading words W10 to W50 using the UNITEReadObject function,
- Reading words W60 to W260 using the UNITEReadWordArray function,
- Closing the communication channel,
- Releasing the communication context.

3.4-2 "Example.C" Source Program

```
/                                                                        /
****************************************************************************
//
//          Example of PC-DOS library usage
//
//
/                                                                        /
****************************************************************************

/*  Objectives:

            - Examples of variables read
    Log:

           Created 19 January 1994
    Test algorithm:
            - Initialize driver
            - Open a communication channel
            - Read internal words 10 to 50 from the station
            - Read internal words 60 to 260 from the station
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unite.h>   //Prototype of the UNITE function + constants

void main(void)
{
//  INTERNAL VARIABLES
    HEQUIP hEquip;
    HREQ hrequete;
    UNITE_RC rc;
    short buffer1[41],buffer2[201];
    USHORT i;
    DEVICEADD  Device;

// INITIALIZING ADDRESSES
Device.uchBoard   = 1;
Device.uchNetwork  = 0;
Device.uchStation  = 254;
Device.uchGate    = 0;
Device.uchModule   = 0;
Device.uchDevice   = 0;
```

```
// OPENING THE DRIVER
rc = UNITEInitDriver(1);
if (rc < 0 ) {
      printf("\nINITIALIZATION PROBLEM status= %d ",rc);
      exit(0);
      }


// OPENING THE COMMUNICATION CHANNEL WITH THE STATION
hEquip = UNITEOpenConnection(&Device);
if (hEquip<0) {
      printf("\nPROBLEM OPENING CONNECTION status= %d ",hEquip);
      if (hEquip == ENOK)
      printf("\n usExtinfo = %d ",usExtinfo);
      exit(0);
      }

// Reading internal words 10 to 50 from the station
hrequete = UNITEReadObject(hEquip,0x68,0x07,10,41,(PCHAR)buffer1);
if(hrequete<0) {
      printf("\n PB REQUEST Read Object hrequete =%d  ",hrequete);
      if (hrequete == ENOK)
      printf("\n usExtinfo = %d ",usExtinfo);
      exit(0);
      }

//    READ RESPONSE
rc=UNITEResponse(hrequete,WAIT);
if(rc<0) {
      printf("\n PB RESPONSE Read Object rc =%d",rc);
      if (rc == ENOK)
      printf("\n usExtinfo = %d ",usExtinfo);
      exit(0);
      }

//    Display the values read
for (i=0;i<41;i++) {
      printf("\n  value[%d] = %d",i,buffer1[i]);
      }
```

```
// Reading internal words 60 to 260 from the station
hrequete=UNITEReadWordArray(hEquip,60,201,buffer2);
if(hrequete<0) {
     printf("\n PB REQUEST Read Word Array hrequete[0] =%d
     ",hrequete);
if (hrequete == ENOK)
     printf("\n usExtinfo = %d ",usExtinfo);exit(0);
     }

//   Displaying the values read
for (i=0;i<201;i++) {
     printf("\n  value[%d] = %d",i,buffer2[i]);
     }


//  CLOSING THE COMMUNICATION CHANNEL
rc = UNITECloseConnection(hEquip);
if (rc != OK) {
    printf("\nError UNITECloseConnection n¯%d : %d",i,rc);
if (rc == ENOK)
     printf("\n usExtinfo = %d ",usExtinfo);
    }

rc = UNITECloseDriver(1);
if (rc != OK) {
     printf("\nError UNITECloseDriver : %d",rc);
if (rc == ENOK)
     printf("\n usExtinfo = %d ",usExtinfo);
     }
}
```

Section		Page
4.1	**Developing the Application under OS/2**	32
4.2.	**Compiling and Linkage Editing**	32
4.3	**Installing the Driver and Connecting**	32
4.4	**Debugging and Operation**	33
4.5	**Application Example under OS/2**	33
	4.5-1 Layout Corresponding to the Example	34
	4.5-2 Program Description	34
	4.5-3 "Example.C" Source Program	35

4.1 Developing the Application under OS/2

The application should be written in C or any language accepting routines written in Microsoft C.

Possible languages:

. Microsoft C Version 6.0 (16-Bit)
. IBM C/2 Version 1.3 (16-Bit)
. IBM C Set Version 2.0 (32-Bit)

The software program follows a client-server pattern and supports OS/2 multi-task management. Therefore, a number of services can be used on an asynchronous basis by several applications.

Each service corresponds to a "function" in the C language sense. Using a service amounts to a simple call with argument passing.

The user program organizes the management and coherence of the various services it requires.

Before activating an application, ensure that the TP UNI-TE/OS2 server is activated first.

4.2 Compiling and Linkage Editing

Any C program using the TP UNI-TE/OS2 library must always begin with:

 # include <UNITE.h> (16-bit environment)

or

 # include <UNITE32.h> (32-bit environment)

The application should be compiled in LARGE format (16-bit environment).

4.3 Installing the Driver and Connecting

Refer to the PL7-3 coprocessor documentation.

4.4 Debugging and Operation

Test the data link with the remote SERVER device(s) by validating only a simple portion of the application:

- Using the *UNITEMirror* function

or

- Using the example described in the next section.

4.5 Application Example under OS/2

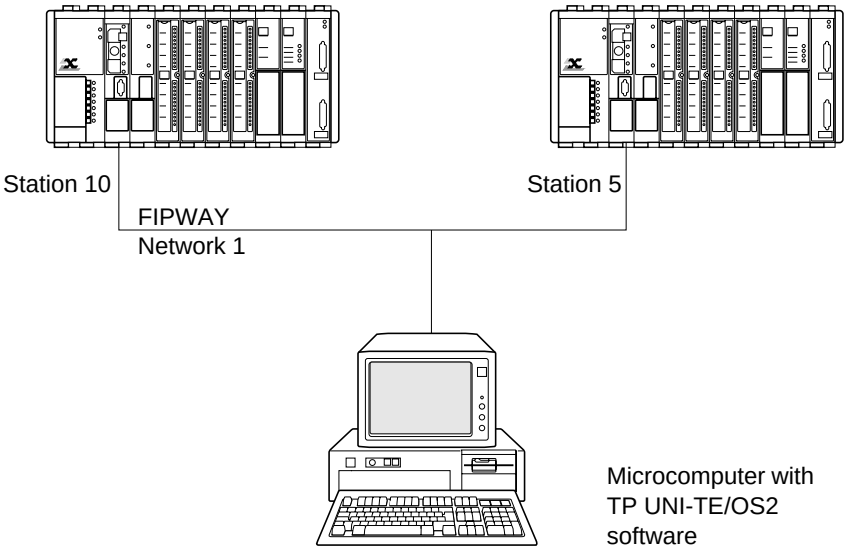
The "EXAMPLE.C" application supplied with the software package and described in this section may be used for testing the installation and the correct operation of the data link to any remote TSX7 PLC SERVER device.

To do this, change the address (*UNITEOpenConnection*) and make it correspond to the address of the device to access.

As this application is supplied in the form of a "source" file, it requires compilation prior to use.

A command file comes with the example, type "comp example" to obtain an executable object file (in the 16-bit development environment).

4.5-1 Layout Corresponding to the Example



4.5-2 Program Description

The program performs the following operations:

- Initializing the communication environment,
- Opening a communication channel,
- Reading internal word W50 from the PLC,
- Response retrieval,
- Writing internal word W100 to the PLC,
- Response retrieval
- Transferring the TSX.APP program from the TSX PLC to the PC,
- Response retrieval,
- Closing the communication channel,
- Releasing the communication context.

4.5-3 "Example.C" Source Program

```
/                                                                              /
*******************************************************************************
//          Example.C
/                                                                              /
*******************************************************************************

/*  Objectives:

          - Examples of reading and writing variables and
transferring programs

    Log:

          Created 10 December 1993

    Test algorithm:
          - Initializing the driver
          - Opening the communication channel
          - Reading internal word 50
          - Writing internal word 100
          - Transferring the program from TSX -> PC */

/*IMPORTED FILES */

#include <stdlib.h>
#include <stdio.h>
#include <os2.h>
#include <unite.h>


/*  GLOBAL VARIABLES */

  static DEVICEADD Equip1Adress = {1,0,1,10,0,0,0};
  static DEVICEADD Equip2Adress = {1,0,1,5,0,0,0};
  HEQUIP           hequip[10]   = {1,0,1,5,0,0,0};
  HREQ             hrequete[10];
  UNITE_RC         rc;
  short            valeur;
  USHORT           i;
void main (void)
{
```

```
/                                                                        /
*************************************************************************
// CLIENT CONTEXT INITIALIZATION//
*************************************************************************

rc = UNITEInitDriver();
if(rc<0)
     {
     printf("\nINITIALIZATION PROBLEM status=%d.",rc);
     exit(0);
     }
else
     printf("\n INIT DRIVER OK.\n");

/                                                                        /
*************************************************************************
// OPEN A CONNECTION WITH THE STATION 10//
*************************************************************************
hEquip[1] = UNITEOpenConnection(&Equip1Adress);
if (hEquip[1]<0)
     {
     printf("\nPROB. OPENING CONNECTION 1 status= %d ",hequip[1]);
     exit(0);
     }
else
     printf ("\n 1.re CONNECTION OPEN hequip[1]=%d.\n",hequip[1];

/                                                                        /
*************************************************************************
// OPEN A CONNCTION WITH THE STATION 5//
*************************************************************************
hEquip[2] = UNITEOpenConnection(&Equip2Adress);
if (hEquip[2]<0)
     {
     printf("\nPROB. OPENING CONNECTION 2 status= %d ",hequip[2]);
     exit(0);
     }
else
     printf ("\n 2.me CONNECTION OPEN hequip[2]=%d.\n",hequip[2];
```

```
*****************************************************************************
// READ WORD W50 FROM THE STATION 10/
*****************************************************************************
hrequete[1] = UNITEReadInternalWord(hequip[1],50, &valeur;
if(hrequete[1]<0)
      {
      printf("\n REQUEST PROB. Read hrequete[1] =%d.",hrequete[1]);
      exit(0);
      }
else
      printf("\n REQUEST READ W50 OK.\n");

*****************************************************************************
// READ RESPONSE W50/
*****************************************************************************
rc=UNITEResponse((HREQ)hrequete[1],WAIT);
if(rc<0)
      {
      printf("\n RESPONSE PROB. Read status=%d.",rc); exit(0);
      }
else
      printf("\n RESPONSE W50 OK value = %d.\n",valeur);

*****************************************************************************
// WRITE WORD W100 TO THE STATION 10/
*****************************************************************************

hrequete[1]=UNITEWriteInternalWord(hequip[1], 100, value);
if (hrequete[1]<0
      {
      printf("\n PB REQUEST Read hrequete[1]=%d, value=%d; ",
            hrequete[1],value); exit(0);
      }
else
      printf("\n REQUEST READ W100 OK\n");

*****************************************************************************
// WRITE RESPONSE W100/
*****************************************************************************
rc = UNITEResponse (hrequete[1],WAIT);
if (rc<0)
      {
      printf("\n RESPONSE PROB. Write status=%d.",rc); exit(0);
      }
else
      printf("\n Reponse W100 OK\n");
```

```
*****************************************************************************
// TRANSFER THE PROGRAM FROM THE STATION 10/
*****************************************************************************
hrequete[1]=UNITETransferTsxPC(hequip[1],"TSX.APP");
if (hrequete[1]<0)
     {
     printf ("\n Err. UNITETransferTsxPC:%d",hrequete[1];exit(0);
     }
else
     printf ("\n REQUEST Transfer TEST.APP OK.\n");

*****************************************************************************
// TRANSFER RESPONSE/
*****************************************************************************
rc = UNITEResponse ((HREQ)hrequete[1],WAIT);
if (rc<0)
     {
     printf("\n RESPONSE problem with status=%d\n",rc); exit(0);
     }
else
     printf("\n TRANSFER RESPONSE OK.\n");

*****************************************************************************
// CLOSE PLC CONNECTIONS/
*****************************************************************************
rc = UNITECloseConnection(hequete[1]);
if (rc<0)
     {
     printf("\n  Error  UNITECloseConnection  status=%d\n",rc);
exit(0);
     }
else
     printf("\n 1.st CONNECTION CLOSED.\n");
     printf("\n 2.nd CONNECTION CLOSED.\n");

*****************************************************************************
// CLOSE DRIVER/
*****************************************************************************
rc = UNITECloseDriver ();
if (rc<0)
     {
     printf("\n Error UNITECloseDriver:%d\n",rc); exit(0);
     }
else
     printf("\n CLOSE DRIVER OK.\n");
     }
```

Section		Page
5.1	**Developing the Application under Windows**	40
5.2.	**Compiling and Linkage Editing**	40
5.3	**Debugging and Operation**	40
5.4	**Application Example under Windows**	40
	5.4-1 Program Description	41
	5.4-2 "Example1.C" Source Program	41

5.1 Developing the Application under Windows

The application should be written in C or any language accepting routines written in Microsoft C.

It is possible to use Microsoft Visual C++.

Each service corresponds to a "function" in the C language sense. Using a service amounts to a simple call with argument passing.

The user program organizes the management and consistency of the various services required.

5.2 Compiling and Linkage Editing

Any C program using the TP UNI-TE/WINDOWS library must close the "include" list with :

 # include <UNITE.h>

The application can be compiled in SMALL, MEDIUM or LARGE format.

5.3 Debugging and Operation

Test the link to the remote SERVER device(s) by validating only a simple portion of the application:

- Use the *UNITEMirror* function

or

- Use the example described in the next section.

5.4 Application Example under Windows

The "EXAMPLE1.C" application provided with the software package and described in this section can be used for testing the installation and the correct operation of the data link to any remote TSX7 PLC SERVER device.
To do this, simply change the address (*UNITEOpenConnection*) and make it correspond to that of the device to access.

5.4-1 Program Description

The program performs the following operations :

- Initialization of the communication environment
- Opening a communication channel (with station 5)
- Reading internal words 10 to 50 using the UNITEReadObject function
- Response retrieval
- Reading internal words 60 to 260 using the UNITEReadWordArray function
- Closing the communication channel (with station 5)
- Releasing the communication context.

5.4-2 "Example1.C" Source Program

```
/                                                                        /
*************************************************************************
//          Example of DLL XWAY WINDOWS usage
/                                                                        /
*************************************************************************

/*
Objective : example of variables read

Test algorithm :
     - Initialize driver when window opens
     - Open a communication channel
     - Read internal words 10 to 50 with UNITEReadObject request
     - Read internal words 60 to 260 with UNITEReadWordArray
     request
     - Close the communication channel
*/

#include <windows.h>
#include <string.h>
#include "example.h"
#include "resource.h"
#include <stdio.h>
#include <unite.h>   prototype of the DLL functions
                     and definition of constants */
```

```
/*  INTERNAL VARIABLES */

HANDLE     hlnst;
char       buffer[150];
int        status;
HANDLE     hLib;
HEQUIP     equip[2];

/*  INITIALIZING DEVICE STRUCTURE

Drivers :   1
Unused :    0
Network :   1
Station :   5
Gate :      0
Module :    0
Channel :   0

*/
DEVICEADD  device = {1,0,1,5,0,0,0};

HWND hEditWnd;  /* handle to edit windows  */
HWND hwnd;      /* handle to main windows  */

*************************************************************************
// FUNCTION: WinMain(HANDLE, HANDLE, LPSTR, int)
   PURPOSE: calls initialization function, processes message
loop//
*************************************************************************

int PASCAL WinMain(HANDLE hInstance,
                HANDLE hPrevInstance,
                LPSTR IpCmdLine,
                int nCmdShow)
{
MSG        msg;
FARPROC    IpfnInit; // pointer on UNITEInit function
FARPROC    IpfnOpen; // pointer on UNITEOpenConnection function

/*loading DLL UPCWIN_M in medium format */
hLib = LoadLibrary;
if (hLib<32)
     return FALSE;
```

```
// Initializing driver
IpfnInit = GetProcAddress(hLib, "UNITEInitDriver");
status=(*IpfnInit)(1);

// Opening communication channel
IpfnOpen = GetProcAddress(hLib, "UNITEOpenConnection");
equip[0]=(*IpfnOpen)((PDEVICEADD)&device);

if (!hPrevInstance)
if (!InitApplication(hInstance))
     return (FALSE);

if (!InitInstance(hInstance, nCmdShow))
     return (FALSE);

while (GetMessage(&msg, NULL, NULL, NULL)) {
     TranslateMessage(&msg);
     DispatchMessage(&msg);
}
return (msg.wParam);
}


/****************************************************************
     FUNCTION: MainWndProc(HWND, unsigned, WORD,LONG)
     PURPOSE: Processes messages
     MESSAGES:
     WM_COMMAND - application menu (About dialog box)
     WM_DESTROY - destroy window
****************************************************************************/
long FAR PASCAL MainWndProc(HWND hWnd,
                unsigned message,
                WORD wParam,
                LONG IParam)
{
FARPROC         IpfnReadWordArray;
FARPROC         IpfnResponse;
FARPROC         IpfnReadObject;
FARPROC         IpfnClose;
FARPROC         IpfnReadCloseC;
static int      Req[2];
WORD            IpuBufobj[41];
WORD            IpuBufArray[201];
```

```
switch (message) {
case WM_COMMAND:
     switch (wParam) {
     /* file menu commands */

case ID_WORDLIST_START
     // Using request Read word list: synchronous request
     IpfnReadWordArray = GetProcAddress(hLib, "UNITEReadWordArray");
     Req[0]=(*IpfnReadWordArray)     (HEQUIP)equip[0],
                                     (USHORT)60,//start read
                                     (USHORT)201,//quantity
                                     (PSHORT)IpuBufArray;
     break;

case ID_OBJECTREQUEST_START
     // Using Read object request: asynchronous request
     IpfnReadObject = GetProcAddress(hLib, "UNITEReadObject");
     Req[1]=(*IpfnReadObject)  (HEQUIP)equip[0],
                               (UCHAR)104,
                               (UCHAR)7,
                               (USHORT)10,//start read
                               (USHORT)41,//quantity
                               (PUCHAR)IpuBufobj;
if (Req[1]>=0
{
     /*retrieval of response to request Req[1]*/
     IpfnResponse = GetProcAddress(hLib,"UNITEResponse");
     status=(*IpfnResponse)(Req[1],WAIT);
}
break;

case IDM_EXIT:
     DestroyWindow(hWnd);
     break;
}
break;
```

```
Case WM_DESTROY:
      // Closing communication channel
      IpfnCloseC=GetProcAdress(hLib, "UNITECloseConnection");
      status=(*IpfnCloseC)(equip[0]);

      /* Closing driver */
      IpfnClose=GetProcAdress(hLib, "UNITECloseDriver");
      status=(*IpfnClose)(1);

      // downloading DLL
      FreeLibrary(hLib);

      PostQuitMessage(0);
      break;

default:
return (DefWindowProc(hWnd, message, wParam,Iparam));
}
return (NULL);
}
```

Section		Page
6.1	**Functions: Managing the Communication Context**	49
6.1-1	UNITEInitDriver	49
6.1-2	UNITECloseDriver	50
6.1-3	UNITEOpenConnection	51
6.1-4	UNITECloseConnection	52
6.1-5	UNITEOpenUnsolicitedData	53
6.1-6	UNITECloseUnsolicitedData	54
6.1-7	UNITETimeOut	55
6.1-8	UNITEGetSystemError (OS/2 only)	56
6.1-9	UNITEGetUsExtInfo (WINDOWS only)	56
6.2	**Functions: Managing Devices**	57
6.2-1	UNITEMirror	57
6.2-2	UNITEIdentification	58
6.2-3	UNITEReserve	59
6.2-4	UNITERelease	60
6.2-5	UNITEIAmAlive	61
6.2-6	UNITERun	62
6.2-7	UNITEStop	63
6.3	**Functions: Accessing Variables**	64
6.3-1	UNITEReadInternalBit	64
6.3-2	UNITEReadSystemBit	65
6.3-3	UNITEWriteInternalBit	66
6.3-4	UNITEWriteSystemBit	67
6.3-5	UNITEReadInternalWord	68
6.3-6	UNITEReadConstantWord	69
6.3-7	UNITEReadSystemWord	70
6.3-8	UNITEWriteInternalWord	71
6.3-9	UNITEWriteSystemWord	72
6.3-10	1UNITEReadCommonWord	73
6.3-11	UNITEWriteCommonWord	74
6.3-12	UNITEReadInternalDWord	75

Section		Page
6.3-13	UNITEReadConstantDWord	76
6.3-14	UNITEWriteInternalDWord	77
6.3-15	UNITEReadObject	78
6.3-16	UNITEWriteObject	79
6.3-17	UNITEReadWordArray	80
6.3-18	UNITEWriteWordArray	81
6.4	**Functions: Domain Management**	82
6.4-1	UNITETransferTsxPC	82
6.4-2	UNITETransferPCTsx	84
6.5	**Function: Unsolicited Data**	86
6.5-1	UNITEUnsolicitedData DOS	86
6.5-2	UNITEUnsolicitedData OS/2	87
6.6	**Function: Generic Function**	88
6.6-1	UNITERequest	88
6.7	**Function: Response Retrieval**	90
6.7-1	UNITEResponse	90
6.7-2	UNITEResponseMult (OS/2 only)	91

6.1 Functions: Managing the Communication Context

6.1-1 UNITEInitDriver

• Under DOS and WINDOWS

Description:

Initializes the communication context. A client application must start with UNITEInitDriver.

Syntax:

UNITE_RC rc = UNITEInitDriver (USHORT usNb_drv);

Input:

USHORT usNb_drv : Number of drivers 1

Information returned:

- Either an OK code if no problem was found,
- Or a negative error code:

EBORNES	The **usNb_drv** parameter is invalid.
EDRVAOPEN	Driver(s) already open.
ENOK	There is a general error returned by the driver. See the **usExinfo** global driver to find the cause (see Error Tables).

• Under OS/2

Description:

Initializes the communication context. A client application must start with UNITEInitDriver.

Syntax:

UNITE_RC rc = UNITEInitDriver ();

Information returned:

- Either an OK code if no problem was found,
- Or a negative error code:

NORESAIVAIBLE	No more resources available.
MPV_INITDONE	Client initialization already done.

6.1-2 UNITECloseDriver

• Under DOS and WINDOWS

Description:

Releases the communication context. Use this function to terminate any client application.

Syntax:

UNITE_RC rc = UNITECloseDriver(USHORT usNb_drv);

Input:

USHORT usNb_drv : Number of drivers to close = 1.

Information returned:

- Either an OK code if no problem was found,
- Or a negative error code:

EBORNES	The **usNb_drv** input parameter is invalid.
EDRVAOPEN	Driver(s) not opened.
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

• Under OS/2

Description:

Releases the communication context. Use this function to terminate any client application.

Syntax:

UNITE_RC rc = UNITECloseDriver();

Information returned:

- Either an OK code if no problem was found,
- Or a negative error code:

PBINIT	Driver not initialized.

6.1-3 UNITEOpenConnection

Description:

Assigns a communication channel between the driver and a remote device.

Syntax:

HEQUIP Hequip = UNITEOpenConnection(PDEVICEADD pDeviceAdd);

Input:

PDEVICEADD pDeviceAdd : Remote device address

with typedef struct {

```
UCHAR   uchBoard;    :   Driver number
UCHAR   uchMode;     :   Not used, set to 0
UCHAR   uchNetwork;  :   \ Corresponding to
UCHAR   uchStation;  :   | the XWAY target
UCHAR   uchGate;     :   | address as it is
UCHAR   uchModule;   :   | defined in the
UCHAR   uchDevice;   :   / UNI-TE protocol.
} DEVICEADD,  * PDEVICEADD;
```

Information returned DOS and WINDOWS:

Either a code (>=0) if no problem occurred.
The hEquip code identifies a device.
It will be passed as an argument on each function call to the device.
Or a negative error code:

EBORNES	The **uchBoard** input parameter is invalid.
EDRVAOPEN	Driver(s) already open.
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).
ENOMORESOCK	All available communication channels opened.
EMEMFULL	No more free memory available.
ESRCEADR	Invalid source address.
EDSTADR	Invalid remote device address.

Information returned OS/2:

Either a code (>=0) if no problem occurred.
The hEquip code identifies a device.
It will be passed as an argument on each function call to the device.
Or a negative error code:

NORESAVAILABLE	No more resources available.
PBINITSESSION	Communication context not initialized.
MPW_NO_ok	Data link problem.

6.1-4 UNITECloseConnection

Description:

Releases a driver to device communication channel.

Syntax:

UNITE_RC rc = UNITECloseConnection (HEQUIP hEquip);

Input:

hEquip : Identifier of the device assigned to the channel to release.

Information returned DOS and WINDOWS:

- Either an OK code if no problem was found,
- Or a negative error code:

EBORNES	The **hEquip** input parameter is larger than the maximum number of connections available.
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).
ENOTOPENSOCK	The communication channel specified by the **hEquip** parameter was not opened.

Information returned OS/2:

- Either an OK code if no problem was found,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.

6.1-5 UNITEOpenUnsolicitedData

Description:

Applications run on the coprocessor or on devices connected to the coprocessor via FIPWAY/FIPIO address their unsolicited data to the communication channel opened for reception (network and station addresses are those of the coprocessor, the target gate number is 3). By TXT block; LOCAL or NET TXT block (LOCAL for a coprocessor application, NET on FIPWAY), TER type, the Txti,T parameter will be set to 3 in the application program prior to sending.

Syntax:

HEQUIP hEquip = UNITEOpenUnsolicitedData(UCHAR uchNumDriver);

Input:

uchNumDriver : Driver number: 1 for DOS and WINDOWS, 0 for OS/2

Information returned DOS and WINDOWS:

- Either a number (>=0) identifying the communication channel opened for receiving unsolicited data if no problem occurs.
  The hEquip number will be passed as an argument each time unsolicited data is read.
- Or a negative error code.

EBORNES	The **uchNum_drv** input parameter is invalid.
EDRVNOTOPEN	The driver was not opened.
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).
EMEMFULL	There is no more free memory available.
EUNSOOPEN	The communication channel assigned to unsolicited data is already open.

Information returned OS/2:

- Either a number (>=0) identifying the communication channel opened for receiving unsolicited data if no problem occurs.
  The hEquip number will be passed as an argument each time unsolicited data is read.
- Or a negative error code.

MPW_OPENED	Unsolicited data channel already opened.
PBINIT	Driver not initialized.
NORESAVAILABLE	No more resources available.

6.1-6 UNITECloseUnsolicitedData

Description:

Releases the communication channel used to receive unsolicited data.

Syntax:

UNITE_RC rc = UNITECloseUnsolicitedData(UCHAR uchNumDriver);

Input:

uchNumDriver : Driver number: 1 for DOS and WINDOWS, 0 for OS/2.

Information returned DOS and WINDOWS:

- Either an OK code if no problem was found,
- Or a negative error code:

EBORNES	The **uchNum_drv** input parameter is invalid.
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).
ENOTOPENSOCK	The communication channel specified by the **hEquip** parameter was not opened.

Information returned OS/2:

- Either an OK code if no problem was found,
- Or a negative error code:

NORESAVAILABLE	No more resources available.
PBINITSESSION	Communication context not initialized.
PBIN	Driver not initialized.
MPW_NO_OK	Data link problem.

6.1-7 UNITETimeOut

Description:

Sets the maximum time-out before request execution.
When the TP UNI-TE program sends a request to a remote device, it starts a time-out equal to this value.
If ***UNITEResponse*** is activated in the application program and the time-out expires before any response is received, UNITEResponse will send back a TIME_OUT error code.

Syntax:

UNITE_RC rc = UNITETimeOut(ULONG ulTimeOut);

Input:

ulTimeOut: Time in milliseconds.

Information returned DOS and WINDOWS:

- Either an OK code if no problem was found,
- Or a negative error code:

EBORNES	The ulTimeOut parameter is less than -1

Remark
On time-out the transaction is aborted.

Information returned OS/2:

- Either an OK code if no problem was found,
- Or a negative error code:

MPW_NO_OK	Data link problem

Remark
On time-out the transaction is aborted.

6.1-8 UNITEGetSystemError (OS/2 only)

Description:

Retrieving a system error when a function returns the MPW_ERR_SYSTEM code.

Syntax:

USHORT rc = UNITEGetSystemError();

Information returned:

The last system error number.

6.1-9 UNITEGetUsExtInfo (WINDOWS only)

Description :

Retrieves the UsExtInfo variable in order to find out the details of an error (ENOK) returned by a function.

Syntax :

UNITE-RC rc = UNITEGetUsExtInfo (void);

Input :

None.

Information returned :

The UsExtInfo variable : the error codes linked to this variable are shown in the Appendix.

6.2 Functions: Managing Devices

6.2-1 UNITEMirror

Description:

Enables testing connection with a device.
All UNI-TE server devices support this request and return the same data as received.

Syntax:

```
HREQ    hReq = UNITEMirror (HEQUIP    hEquip,
                            UCHAR uchQty);
```

Input:

hEquip : Identifier of the request target.
uchQty : Number of bytes sent (from 1 to 126).

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBORNES	The **ucQty** input parameter value exceeds 126.
EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.2-2 UNITEIdentification

Description:

Remote device identification.
All UNI-TE server devices support this request.

Syntax:

```
HREQ    hReq = UNITEIdentification (HEQUIP    hEquip,
                                    PUCHAR    pBuffer_Ident);
```

Input:

hEquip : Identifier of the request target.

Output:

pBuffer_Ident : Pointer of the identification response storage buffer.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.2-3 UNITEReserve

Description:

Reservation of the server device for communication purposes.
The reservation starts with the *UNITEReserve* function and ends with the *UNITERelease* function.
The reservation is automatically halted after 60s if not kept alive by means of the *UNITEIAmAlive* function.
The reservation may be kept alive by any UNI-TE request.
If there is no communication, this is achieved through the *UNITEIAmAlive* function.

Syntax:

HREQ hReq = UNITEReserve (HEQUIP hEquip);

Input:

hEquip : Identifier of the request target.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

Remark

The critical services which require reservation (RUN, STOP, File transfer, etc.), reservation maintenance and reservation release should use the same communication channel as that implemented at the time of reservation.

6.2-4 UNITERelease

Description:

Release of a device.

Syntax:

HREQ hReq = UNITERelease(HEQUIP hEquip);

Input:

hEquip : Identifier of the request target.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

Remark

The critical services which require reservation (RUN, STOP, File transfer, etc.), reservation maintenance and reservation release should use the same communication channel as that implemented at the time of reservation.

6.2-5 UNITEIAmAlive

Description:

Device reservation maintenance request. This reservation should be periodically maintained (approx. every 60 s), else the device is automatically released.

Syntax:

HREQ hReq = UNITEIAmAlive(HEQUIP hEquip);

Input:

hEquip : Identifier of the request target.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

Remark

The critical services which require reservation (RUN, STOP, File transfer, etc.), reservation maintenance and reservation release should use the same communication channel as that implemented at the time of reservation.

6.2-6 UNITERun

Description:

Used for setting a device to "Run".

Remark

Requires the preliminary reservation of the TSX7 PLC by the same communication channel.

Syntax:

HREQ hReq = UNITERun (HEQUIP hEquip);

Input:

hEquip : Identifier of the request target.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.2-7 UNITEStop

Description:

Used for setting a device to "Stop".

Remark

Requires the preliminary reservation of the TSX7 PLC by the same communication channel.

Syntax:

HREQ hReq = UNITEStop(HEQUIP hEquip);

Input:

hEquip : Identifier of the request target.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3 Functions: Accessing Variables

6.3-1 UNITEReadInternalBit

Description:

Reading an internal bit Bi.

Syntax:

```
HREQ    hReq = UNITEReadInternalBit (HEQUIP    hEquip,
                                     USHORT    usNbit,
                                     PBITVAL   pValue);
```

Input:

hEquip : Identifier of the request target.
usNbit : Number of the internal bit to read.

Output:

pValue : Response structure pointer.
with typedef struct {

```
        BYTE    bValue;      : Value of the bit read
        BYTE    bOverride;   : Forcing state of the bit read
        }BITVAL,  * PBITVAL;
```

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-2 UNITEReadSystemBit

Description:

Reading a system bit SYi.

Syntax:

```
HREQ    hReq = UNITEReadSystemBit (HEQUIP    hEquip,
                                   USHORT    usNbit,
                                   PBITVAL   pValue);
```

Input:

hEquip : Identifier of the request target.
usNbit : Number of the system bit to read.

Output:

pValue : Response structure pointer.
with typedef struct {

```
BYTE    bValue;
BYTE    bOverride;
}BITVAL,  * PBITVAL;
```

Only the bValue field (value of the bit read) is significant.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-3 UNITEWriteInternalBit

Description:

Writing an internal bit Bi.

Syntax:

```
HREQ    hReq  =  UNITEWriteInternalBit ( HEQUIP    hEquip
                                         USHORT   usNbit,
                                         BOOL     boValue);
```

Input:

hEquip : Identifier of the request target.
usNbit : Number of the internal bit to write.
boValue : Value to write (0 or 1)

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-4 UNITEWriteSystemBit

Description:

Writing a system bit SYi.

Syntax:

```
HREQ    hReq  =  UNITEWriteSystemBit( HEQUIP    hEquip
                                      USHORT   usNbit,
                                      BOOL     boValue);
```

Input:

hEquip : Identifier of the request target.
usNbit : Number of the system bit to write.
boValue : Value to write (0 or 1).

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-5 UNITEReadInternalWord

Description:

Reading an internal word Wi.

Syntax:

```
HREQ    hReq = UNITEReadInternalWord (HEQUIP   hEquip,
                                      USHORT   usNword,
                                      PSHORT   pValue);
```

Input:

hEquip : Identifier of the request target.
usNword : Number of the internal word to read.

Output:

pValue : Pointer to the word read.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-6 UNITEReadConstantWord

Description:

Reading a constant word CWi.

Syntax:

```
HREQ    hReq = UNITEReadConstantWord ( HEQUIP   hEquip,
                                       USHORT  usNword,
                                       PSHORT  pValue);
```

Input:

hEquip : Identifier of the request target.
usNword : Number of the constant word to read.

Output:

pValue : Pointer to the word read.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-7 UNITEReadSystemWord

Description:

Read a system word SWi.

Syntax:

```
HREQ    hReq = UNITEReadSystemWord ( HEQUIP    hEquip,
                                     USHORT    usNword,
                                     PSHORT    pValue);
```

Input:

hEquip : Identifier of the request target.
usNword : Number of the system word to read.

Output:

pValue : Pointer to the word read.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-8 UNITEWriteInternalWord

Description:

Writing an internal word Wi.

Syntax:

```
HREQ    hReq  = UNITEWriteInternalWord( HEQUIP   hEquip,
                                        USHORT  usNword,
                                        SHORT   sValue);
```

Input:

hEquip	: Identifier of the request target.
usNword	: Number of the internal word to read.
sValue	: Value to write (in 16-bits)

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-9 UNITEWriteSystemWord

Description:

Write a system word SWi.

Syntax:

```
HREQ    hReq  = UNITEWriteSystemWord( HEQUIP    hEquip,
                                      USHORT   usNword,
                                      SHORT    sValue);
```

Input:

hEquip	: Identifier of the request target.
usNword	: Number of the system word to write.
sValue	: Value to write (on 16-bits)

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-10 UNITEReadCommonWord

Description:

Read a common word COMi,j.
With i = station number, and j = word number.

Syntax:

```
HREQ hReq  = UNITEReadCommonWord (HEQUIP          hEquip,
                                  UCHAR           uchStation,
                                  USHORT          usNWord,
                                  PCOMWORDVAL     pValue);
```

Input:

hEquip : Identifier of the request target.
uchStation : Station number.
usNWord : Number of the word to read.

Output:

pValue : Response structure pointer.
with typedef struct {

```
USHORT      usNbWord;      : Number of COM words
                             in the station.
SHORT       sValue;        : Value read.
}COMWORDVAL,  * PCOMWORDVAL;
```

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available.
EBUILD	Error building the datagram.
ENOK	General error returned by the driver. See global variable **usExtinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-11 UNITEWriteCommonWord

Description:

Writing a common word COMi,j .
With i = station number, and j = word number.

Syntax:

```
HREQ  hReq = UNITEWriteCommonWord ( HEQUIP    hEquip,
                                    UCHAR     uchStation,
                                    USHORT    usNWord,
                                    SHORT     sValue);
```

Input:

hEquip	: Request target identifier.
uchStation	: Target station number.
usNWord	: Number of the word to write.
sValue	: Value to write.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-12 UNITEReadInternalDWord

Description:

Read an internal double word DWi.

Syntax:

```
HREQ   hReq = UNITEReadInternalDWord ( HEQUIP   hEquip,
                                       USHORT   usNDword,
                                       PLONG    pValue);
```

Input:

hEquip : Request target identifier.
usNDword : Number of the internal double word to read.

Output:

pValue : Pointer to the double word read.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-13 UNITEReadConstantDWord

Description:

Read a double constant word DCWi.

Syntax:

```
HREQ    hReq = UNITEReadConstantDWord (HEQUIP    hEquip,
                                       USHORT    usNDword,
                                       PLONG     pValue);
```

Input:

hEquip : Request target identifier.
usNDword : Number of the double constant word to read.

Output:

pValue : Pointer to the double word read.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-14 UNITEWriteInternalDWord

Description:

Write an internal double word DWi.

Syntax:

```
HREQ hReq  = UNITEWriteInternalDWord ( HEQUIP    hEquip,
                                       USHORT    usNDword,
                                       LONG      lValue);
```

Input:

hEquip : Request target identifier.
usNDword : Number of the double word to write.
lValue : Value to write (on 32-bits).

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-15 UNITEReadObject

Description:

Read a string of objects.

Remark

The maximum data length (number of objects multiplied by the size of an object in bytes) depends on the interrogated UNI-TE server.
The length is 120 bytes for a TSX PLC or a NUM numerical controller and 30 bytes for TSX 17s.

Syntax:

```
HREQ  hReq = UNITEReadObject (HEQUIP   hEquip,
                              UCHAR    uchSegment,
                              UCHAR    uchType,
                              USHORT   usFirst,
                              USHORT   usQty,
                              PCHAR    pValue);
```

Input:

hEquip	: Request target identifier.
uchSegment	: Specifies the segment number. (see Section 6.4)
uchType	: Specifies the type of object (dbl. word, word, byte, etc.). (Section 6.4)
usFirst	: Address of the first object to read.
usQty	: Number of consecutive objects to read.

Output:

pValue : Address of the buffer where data read is stored.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available.
EBUILD	Error building the datagram.
ENOK	General error returned by the driver. See global variable **usExtinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-16 UNITEWriteObject

Description:

Write a string of objects.

Remark

The maximum data length (number of objects multiplied by the size of an object in bytes) depends on the interrogated UNI-TE server.
The length is 120 bytes for a TSX PLC or a NUM numerical controller and 30 bytes for TSX 17s.

Syntax:

```
HREQ    hReq = WriteObject (HEQUIP      hEquip,
                            UCHAR       uchSegment,
                            UCHAR       uchType,
                            USHORT      usFirst,
                            USHORT      usQty,
                            UCHAR       uchSize,
                            PUCHAR      pValue);
```

Input:

hEquip	: Request target identifier.
uchSegment	: Specifies the segment number. (see Section 6.4)
uchType	: Specifies the type of object (dbl. word, word, byte, etc.). (Section 6.4)
usFirst	: Address of the first object to read.
usQty	: Number of consecutive objects to read.
usSize	: Size of the string of objects to write (in bytes).
pValue	: Address of the buffer where the data to write is stored.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available.
EBUILD	Error building the datagram.
ENOK	General error returned by the driver. See global variable **usExtinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-17 UNITEReadWordArray

Description:

Read an internal word array (segment 104, specific byte 7).

Remark

The maximum array length is 16 Kwords.

Syntax:

```
HREQ    hReq  =  UNITEReadWordArray (HEQUIP    hEquip,
                                    USHORT   usFirst,
                                    USHORT   usQty,
                                    PSHORT   pValue);
```

Input:

hEquip	: Request target identifier.
usFirst	: Address of the first array word.
usQty	: Number of consecutive objects to read (less than 16000).

Output:

pValue	: Address of the buffer containing the words of the read array.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available.
EBUILD	Error building the datagram.
ENOK	General error returned by the driver. See global variable **usExtinfo** to find the cause (see Error Tables).

Remark

This function sends back its return code after execution is completed. It therefore does not require a UNITEResponse request. The assigned channel remains inhibited during this time.

Information returned OS2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.3-18 UNITEWriteWordArray

Description:

Write an internal word array (segment 104, specific byte 7).

Remark

The maximum array length is 16 Kwords.

Syntax:

```
HREQ    hReq =  UNITEWriteWordArray ( HEQUIP    hEquip,
                                      USHORT    usFirst,
                                      USHORT    usQty,
                                      PSHORT    pValue);
```

Input:

hEquip	: Request target identifier.
usFirst	: Address of the first array word.
usQty	: Number of words to write (less than 16000).
pValue	: Address of the buffer of values to write.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available.
EBUILD	Error building the datagram.
ENOK	General error returned by the driver. See global variable **usExtinfo** to find the cause (see Error Tables).

Remark

This function sends back its return code after execution is completed. It therefore does not require a UNITEResponse request. The assigned channel remains inhibited during this time.

Information returned OS2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.4 Functions: Domain Management

6.4-1 UNITETransferTsxPC

Description:

Downloading a TSX Model 40 program to a PC.

Syntax:

```
HREQ    hReq = UNITETransferTsxPC (HEQUIP    hEquip,
                                   PCHAR     pchFileName);
```

Input:

hEquip : Request target identifier.
pchFileName : Application file name with extension.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).
EFICH	Error opening the file.
ELECFICH	Error reading the file.
ENOTSXV4	The TSX processor version is prior to V6.
ETYPTSXDIFF	The type of TSX installed and described in the file are different.
ECOMPATINOK	The file and the TSX are not compatible.
EROMCART	The TSX cartridge is a ROM.
ENOVALIDCART	The TSX cartridge is not valid.
EAPIOUTMEM	The file size exceeds the cartridge size.
ETSXETATNOK	The TSX is not ready to upload or download.
EDATAORDER	Data coherence error.
ESEQ	Sequencing error.
ETSXWRITE	Cannot write.
EVERIFAPPNOK	Invalid application.

Remark

This function sends back its return code after execution is completed. It therefore does not require a UNITEResponse request. The assigned channel remains inhibited during this time.

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

ERRFICH	File error.
NOFICHAPP	The file is not an application file.
ERRECRFICH	Error writing the file.
ERRLECFICH	Error reading the file.
NAKRESERV	TSX reservation impossible.
NOTSXV4	Remote TSX is not a V4 type.
TYPTSXDIFF	The file and remote TSX are different.
TSXETATNOK	TSX status incompatible with transfer
COMPATINOK	TSX compatibility problem.
APIPROT	Protected application.
NOAPI	Non-existent application.
ROMCART	Cartridge is ROM.
NOVALIDCART	Invalid cartridge.
APIOUTMEM	Application too large for remote TSX.
NAKSTOP	Cannot stop remote TSX.
ERRSEQ	Sequencing error.
ERRTSXWRITE	Cannot write TSX.
ERRDATAORDER	Incoherent data.
NAKENTRESERV	Reservation maintenance error.
ENDSEQNAK	End of download sequence refused.
VERIFAPPNOK	Error after checking. Invalid application.
PBINITSESSION	Communication context not initialized.
PBSHUTDOWN	Shutdown problem.
NORESAVAIBLE	Terminal port driver problem.
NOK	No response received yet.
PBRECEPTRAME	Reception error.
NACK	Message received but not processed.
RESPREF	Response refused code received.
PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.4-2 UNITETransferPCTsx

Description:

Uploading a TSX Model 40 PLC program from a PC.

Syntax:

```
HREQ    hReq  =  UNITETransferPcTsx (HEQUIP      hEquip,
                                     PCHAR       pchFileName);
```

Input:

hEquip : Request target identifier.
pchFileName : Application file name with extension.

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).
EFICH	Error opening the file.
ELECFICH	Error reading the file.
ENOTSXV4	The TSX processor version is prior to V6.
ETYPTSXDIFF	The type of TSX installed and described in the file are different.
ECOMPATINOK	The file and the TSX are not compatible.
EROMCART	The TSX cartridge is a ROM.
ENOVALIDCART	The TSX cartridge is not valid.
EAPIOUTMEM	The file size exceeds the cartridge size.
ETSXETATNOK	The TSX is not ready to upload or download.
EDATAORDER	Data coherence error.
ESEQ	Sequencing error.
ETSXWRITE	Cannot write.
EVERIFAPPNOK	Invalid application.

Remark

This function sends back its return code after execution is completed. It therefore does not require a UNITEResponse request. The assigned channel remains inhibited during this time.

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

ERRFICH	File error.
NOFICHAPP	The file is not an application file.
ERRECRFICH	Error writing the file.
ERRLECFICH	Error reading the file.
NAKRESERV	TSX reservation impossible.
NOTSXV4	Remote TSX is not a V4 type.
TYPTSXDIFF	The file and remote TSX are different.
TSXETATNOK	TSX status incompatible with transfer
COMPATINOK	TSX compatibility problem.
APIPROT	Protected application.
NOAPI	Non-existent application.
ROMCART	Cartridge is ROM.
NOVALIDCART	Invalid cartridge.
APIOUTMEM	Application too large for remote TSX.
NAKSTOP	Cannot stop remote TSX.
ERRSEQ	Sequencing error.
ERRTSXWRITE	Cannot write TSX.
ERRDATAORDER	Incoherent data.
NAKENTRESERV	Reservation maintenance error.
ENDSEQNAK	End of download sequence refused.
VERIFAPPNOK	Error after checking. Invalid application.
PBINITSESSION	Communication context not initialized.
PBSHUTDOWN	Shutdown problem.
NORESAVAIBLE	Terminal port driver problem.
NOK	No response received yet.
PBRECEPTRAME	Reception error.
NACK	Message received but not processed.
RESPREF	Response refused code received.
PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

Remark

Under OS/2, this function requires a UNITEResponse request. The transfer must be started when the TSX Model 40 PLC is stopped.

6.5 Function: Unsolicited Data

6.5-1 UNITEUnsolicitedData DOS and WINDOWS

Description:

Reception of unsolicited data. (Refer to UNITEOpenUnsolicitedData).

Syntax:

```
UNITE_RC rc = UNITEUnsollicitedData ( HEQUIP hEquip,
                                      PREADUNSOLLICITEDDATA pValue);
```

Input:

hEquip : Identifier of the communication channel opened for receiving unsolicited data.

Information returned:

```
pValue                  : Data reception buffer.
with typedef struct     {
                          UCHAR     uchBoard
                          UCHAR     uchNetwork;  :  \
                          UCHAR     uchStation;  :  | Source address
                          UCHAR     uchGate;     :  | defined in the
                          UCHAR     uchModule;   :  | XWAY protocol.
                          UCHAR     uchDevice;   :  /
                          UCHAR     uchQty              : Number of usable bytes
                          CHAR      chData[LG_MAX_UNSOL];
                        } READUNSOLLICITEDDATA, * PREADUNSOLLICITEDDATA;
#define LG_MAX_UNSOL    126
```

Information returned DOS and WINDOWS:

- OK : Unsolicited data available in the request buffer.
- Else, a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).
PENDING	No unsolicited data.

6.5-2 UNITEUnsolicitedData OS/2

Description:

Reception of unsolicited data. (Refer to UNITEOpenUnsolicitedData).

Syntax:

```
UNITE_RC rc = UNITEUnsollicitedData ( UCHAR uchNumDriver,
                                      PREADUNSOLLICITEDDATA pValue);
```

Input:

uchNumDriver: Driver number (0 for OS/2).

Output:

```
pValue                : Data reception buffer.
with typedef struct   {
                      UCHAR     uchBoard
                      UCHAR     uchNetwork;  :  \
                      UCHAR     uchStation;  :  | Source address
                      UCHAR     uchGate;     :  | defined in the
                      UCHAR     uchModule;   :  | XWAY protocol.
                      UCHAR     uchDevice;   :  /
                      UCHAR     uchQty       : Number of usable bytes
                      CHAR      chData[LG_MAX_UNSOL];
                    } READUNSOLLICITEDDATA, * PREADUNSOLLICITEDDATA;
#define LG_MAX_UNSOL      126
```

Information returned OS/2:

- OK : Unsolicited data available in the request buffer.
- WAIT : No unsolicited data.
- Else, a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.

6.6 Function: Generic Function

6.6-1 UNITERequest

Description:

Sends a generic request (any type of message).

Syntax:

```
HREQ    hReq  =  UNITERequest ( HEQUIP          hEquip,
                                PREQUESTIN      pIn,
                                PREQUESTOUT     pOut);
```

Input:

hEquip : Request target identifier.
pIn : Pointer of the buffer containing the request to send.

```
with  typedef struct   {
                       UCHAR      uchCode;     Request code
                       UCHAR      uchCat;      Category code
                       UCHAR      uchQty;      Number of bytes in the request
                       CHAR       chData[LG_MAX_PAR-1]; Byte detail.
                       }REQUESTIN,   * PREQUESTIN;
```

Output:

pOut : Pointer of the response retrieval buffer.

```
with  typedef struct  {
                       UCHAR      uchCode;     Response code.
                       UCHAR      uchQty;      Number of bytes in the response.
                       CHAR       chData[LG_MAX_PAR-1];  Byte detail.
                       } REQUESTOUT,   * PREQUESTOUT;

                       with LG_MAX_PAR = 126
```

Information returned DOS and WINDOWS:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

EBADPARAM	The **hEquip** input parameter is less than 0.
ECONNOTOPEN	The communication channel specified by the value of **hEquip** was not opened.
EBADBUFFER	The internal buffers were not assigned.
EMEMFULL	There is no more free memory available
EBUILD	Error building the datagram
ENOK	General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

- Either a request identifier (>=0) if no problems occurred,
- Or a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.7 Function: Response Retrieval

6.7-1 UNITEResponse

Description:

UNI-TE response request function.

Syntax:

```
UNITE_RC rc  =  UNITEResponse (      HREQ        hReq,
                                  short          sMode);
```

Input:

hReq : Request identifier.
sMode : Operating mode.

- WAIT : Response requested with no time-out.
- NOWAIT : Request result required without wait.

Information returned DOS and WINDOWS:

- OK : The response is available in the request buffer.
- PENDING : No response yet, try again later.
- Else, a negative error code:

  ENOK — General error returned by the driver. See global variable **usExinfo** to find the cause (see Error Tables).

Information returned OS/2:

OK	The response is available in the request buffer.
MPW_RETRY	No response yet, try again later.

Else, a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

6.7-2 UNITEResponseMult (OS/2 only)

Description:

Function used to request a report on a list of requests.

Syntax:

```
UNITE_RC rc  =  UNITEResponseMult   (PHREQ      phRequête,
                                    short       ulMod
                                    PUSHORT     pusReponse);
```

Input:

phReq : Pointer to the list of request identifiers (hReq) for which responses are required.
The list must end with HREQ_NULL (-1).

ulMode : Operating mode.

- WAIT : Response requested with no time-out.
- ABANDON : Request list aborted.
- other value : Response time-out (in ms.).

Output:

pusReponse : When the response from a request has been received, pusReponse points the hReq identifier of the corresponding function.
If a number of responses are received, pusReponse points the identifier of the first response received by the server.
Remark: The order in which responses are received by the server is independent of the order in which the UNITE_XX functions were called.

Information returned:

- OK : One of the responses from the request list has been received (its index in the list is pointed by pusReponse). To get the responses to the other requests, repeat the request after removing the terminated request from the array.
- WAIT : No response yet, try again later.
- Else, a negative error code:

PBINITSESSION	Communication context not initialized.
MPW_CLIENTNOK	Communication channel not assigned.
MPW_REQPENDING	Request pending on the communication channel.
MPW_SEQ	Server sequencing problem.
NORESAVAIBLE	No more resources available.
MPW_REQABORTED	Request aborted.
PBDRV	Terminal port problem.
RESPREF	Response code refused.
MPW_ERR_SYSTEM	System error.

Sub-section		Page
7.1	**Glossary**	94
7.2	**List of Error Codes under DOS**	95
7.2-1	Error Codes Common to all Functions	95
7.2-2	Error Codes Relative to File Transfer Functions	96
7.2-3	Error Codes Relative to the usExtinfo Variable	97
7.3	**List of Error Codes under OS/2**	98
7.3-1	Error Codes Common to all Functions	98
7.3-2	Values Relative to File Transfer Functions	98
7.4	**UNI-TE Requests**	99
7.4-1	Read Objects	99
7.4-2	Write Objects	104
7.4-3	Application Initialization	107

7.1 Glossary

Communication channel: A resource used for sending function-related requests to any remote device.

UNI-TE client: A device capable of sending UNI-TE requests.

Layers: see **OSI model**: A reference model that organizes all communications into 7 separate *layers* each of which communicate with their neighbors. The layers are:

- 7	APPLICATION	A window open between two application processes for significant data exchange purposes.
- 6	PRESENTATION	The representation of data flowing between the application processes.
- 5	SESSION	The organization and synchronization of the dialog between two application processes and the management of the related data.
- 4	TRANSPORT	The transparent and reliable transfer of data between two systems.
- 3	NETWORK	The routing of data between two devices.
- 2	DATA LINK	The transfer of data between two adjacent systems with error detection.
- 1	PHYSICAL	The transmission of binary elements between two systems via a communication medium.

UNI-TE: The client-server messaging system developed by Telemecanique.

UNI-TE server: A device responding to service requests received from a client device.

Minimum server: A server device supporting only the mandatory UNI-TE requests: *Mirror*, *Identification*, *Protocol_Version* and *Status*.

Transaction: The request and its response.

7.2 List of Error Codes in DOS and WINDOWS

A code is returned by every function in the library. This code is used for ensuring exchange integrity throughout the operation.
It is available from the status word.

7.2-1 Error Codes Common to all Functions

OK	(0)
ENOK	(-1) : General driver error. See usExtinfo)
PENDING	(-4) : Waiting
EREFUSEDFLAG	(-3) : Flag not valid
EBORNES	(-4) : Value out of range
EDRVNOTOPEN	(-5) : Driver not opened
EMEMFULL	(-6) : No more memory available
ESRCEADR	(-7) : Invalid source address
EDESTADR	(-8) : Invalid target address
ECONNOTOPEN	(-9) : Communication channel not opened
EBUILD	(-10) : Datagram building error
EBADMODE	(-11) : Mode parameter error
ERESPREF	(-12) : Negative UNI-TE response
EMIRROIR	(-13) : Mirror request error
ERQTIMEOUT	(-14) : Request time-out error
EBADADD	(-15) : Incorrect address error
ENOMORESOCK	(-16) : Max. nbr. of com. channels per driver exceeded
ENOTOPENSOCK	(-17) : Communication channel not opened error
EDRVAOPEN	(-18) : Driver already opened error
EBADBUFFER	(-19) : pSendRequest buffer not initialized error
EBADPARAM	(-20) : Invalid parameter error
ECLEPROT	(-21) : No protection key
EUNSOOPEN	(-22) : Unsolicited data channel already opened

7.2-2 Error Codes Relative to File Transfer Functions

ENOTSXV4	(-30)	: Not a V4 or higher version remote PLC
ETYPTSXDIFF	(-31)	: The processor configured in the file and the one in the remote PLC are different
ECOMPATINOK	(-32)	: The version level of the remote PLC and the processor configured in the file are not compatible
EROMCART	(-33)	: ROM cartridge in the PLC
ENOVALIDCART	(-34)	: Invalid cartridge in the PLC
EAPIOUTMEM	(-35)	: PLC memory size too small
ETSXETATNOK	(-36)	: The TSX PLC will not allow remote uploading
EDATAORDER	(-37)	: Data coherence error
ESEQ	(-38)	: Sequencing error
ETSXWRITE	(-39)	: Cannot write
EVERIFAPPNOK	(-40)	: Post-checking error. Invalid application
EFICH	(-41)	: Open file error
ELECFICH	(-42)	: File read error
ENOFICHAPP	(-43)	: This is not an .APP file
ENAKENTRESERV	(-44)	: Reservation maintenance error
EAPIPROT	(-45)	: PLC protected application error
ENOAPI	(-46)	: No application in the PLC
EECRFICH	(-47)	: File write error
ETYPEINVAL	(-48)	: Invalid file type
EPROGEXIST	(-49)	: Program already present
EMANIPMEMCN	(-50)	: Error loading numerical controller memory
ESATURMEM	(-51)	: Memory full
EPBFILE	(-52)	: Error in file
ECNNORAZ	(-53)	: Numerical controller not in reset state
ESIZE	(-54)	: Incoherent size error
ESTCNINCOMP	(-55)	: Incompatible numerical controller status error
ETIMEOUTUCCN	(-56)	: Coprocessor time-out error
EFICHFERME	(-57)	: File already closed
ECLE	(-58)	: Key error
EABORTOPER	(-59)	: Sequence aborted by the operator
EVERIF	(-60)	: Checking error
EENDFILE	(-61)	: End file error
ECRFICH	(-62)	: File write error

7.2-3 Error Codes Relative to the usExtinfo Variable

ENOERROR	(0)	:
ENETUNREACH	(1)	: Network or driver not available
EHDLOUTRANGE	(2)	: No connection for handle
ESOCKNOTOPEN	(3)	: Communication channel not allocated
ESOCKOUTOFR	(4)	: Channel number out of range
ESOCKNOMORE	(5)	: No channel available
ESOCKOPENED	(6)	: Communication channel already reserved
ESOCKNOEMPTY	(7)	: Message in communication channel queue
EPRNOSUPPORT	(8)	: Protocol and driver incompatibility
EINVLEN	(9)	: Buffer size not valid
ENOTCONN	(10)	: Connection not possible
EFULL	(11)	: No driver buffers available
EBADFLAG	(12)	: Asynchronous mode forbidden
ETIMEOUT	(13)	: Time-out
ENOASYNCH	(14)	: IOCB not allocated in asynchronous mode
ESEMOUTOFRANGE	(15)	: Flag address not valid
EINVALIDMODE	(16)	: Flag mode not valid
EATTEMPTINGLINKUP	(17)	: Connection is not yet established
EBADPARAMETER	(18)	: IOCB parameter error
ELOCALNACK	(19)	: No acknowledgment from the interface
EREMOTENACK	(20)	: No acknowledgment from remote device
EBADVERSION	(21)	: Network and driver incompatible
EUNKNOWNERROR	(22)	: Unknown error

7.3 List of Error Codes under OS/2

An error code is returned by every library function. The error codes are used to ensure exchange integrity throughout the operation. They are available from the status word.

7.3-1 Error Codes Common to all Functions

MPW_OK	(0)	Everything is OK
MPW_NO_OK	(-1)	Data link problem
MPW_RETRY	(-2)	No response received yet
TIME_OUT	(-3)	Exchange time-out
NACK	(-4)	The message has been received but not processed by the target address due to lack of resources
BORNE	(-5)	Index out of range
PBINIT	(-6)	Driver not initialized
PBRECEPTRAME	(-7)	Problem on message reception
NORESAVAIBLE	(-8)	No more resources available
PBINITSESSION	(-9)	No channel initialized on this device
PBSHUTDOWN	(-10)	Queue destruction problem
RESPREF	(-11)	Response refused code received
INITNOK	(-12)	Init response refused code
MIRRORNOK	(-13)	Mirror response refused code
NOKEY	(-14)	No protection key

7.3-2 Error Codes Relative to File Transfer Functions

ERRFICH	(-20)	This file cannot be opened
NOFICHAPP	(-21)	This is not an application file
ERRECRFICH	(-22)	File write error
ERRLECFICH	(-23)	File read error
NAKRESERV	(-24)	TSX reservation impossible
NOTSXV4	(-25)	This is not a V4-type remote TSX
TYPTSXDIFF	(-26)	The file and remote TSX are different
TSXETATNOK	(-27)	TSX status incompatible with transfer
COMPATINOK	(-28)	TSX compatibility problem
APIPROT	(-29)	Protected application
NOAPI	(-30)	Non-existent application
ROMCART	(-31)	ROM cartridge
NOVALIDCART	(-32)	Cartridge not valid
APIOUTMEM	(-33)	Application too large for remote TSX
NAKSTOP	(-34)	The remote TSX cannot be stopped
ERRSEQ	(-35)	Sequencing error
ERRTSXWRITE	(-36)	TSX write impossible
ERRDATAORDER	(-37)	Incoherent data
NAKENTRESERV	(-38)	Reservation maintenance error
ENDSEQNAK	(-39)	End of transfer sequence refused
VERIFAPPNOK	(-40)	Post-checking error

7.4 UNI-TE Requests

7.4-1 Read Objects

This request reads simple objects (words or bit strings etc.) using the UNITEReadObject function.

Request format

Request Code Hex.	Category Code	Segment	Type of Object	Object Address		Number of Objects to Read	
36/54	0 → 7						

Segment: Specifies the addressing mode used for the objects to be read and the position where they are located (in hexadecimal notation). The segments accessible by TSX Series 7 PLCs are (in hexadecimal notation):

- 10: Common object space segment,
- 64: Internal bit space segment,
- 68: Internal word space segment,
- 69: Constant word space segment,
- 6C: Ctrl. user task space segment,
- 80: TSX Series 7 system object space segment,
- 81: Function block space segment,
- 82: I/O module space segment.

Type of object: Specifies the type of object to read:

- 0: Text block or module in the rack,
- 1: Ctrl. block,
- 5: Internal bits with forcing,
- 7: 16-bit signed integer,
- 8: 32-bit signed integer,
- 64: Task period.

Object address:

- The physical or logical address in the segment,
- The sequence number of the object in the segment:
  - 0: Current date and time, in the common segment,
  - 1: Stored time and date, in the common segment (last time the system was stopped),
  - 2: Current date and time (in hexadecimal notation) in the common segment.

Read objects (continued)

Confirm format

Positive confirm

Confirm Code Hex.	Type of Object	Data				
66/102						

Type of object: Returns the type of object selected when the request is sent.

Negative confirm

Confirm Code Hex.
FD/253

Causes for rejection:
- Unknown request,
- Inadequate access rights,
- Unknown segment or object,
- Address out of range,
- Too many objects for the reception buffer.

Request Examples

Read words or double words

Segment: 68 (internal word segment),
Type of object: 7 → Wi, or 8 → DWi,
Object address: index of the first Wi or DWi to read,
Response: array of n objects.

Read constant words or constant double words

Segment: 69 (constant word segment),
Type of object: 7 → CWi, or 8 → CDWi,
Object address: index of the first CWi or CDWi to read,
Response: array of n objects.

Read objects (continued)

Read Date and Time

Segment : 10 (common objects segment),
Type of object : 0 by default,
Object address : 0 → current date and time,
1 → saved date and time,
Quantity : 0 by default,
Response : object address = 0 (current date and time):
YYYYMMDDHHMMSS.TN
object address = 1 (saved date and time):
YYYYMMDDHHMMSS.TP
YYYY = year,
MM = month,
DD = day,
HH = hour,
MM = minute,
SS = second,
T = tenth of a second,
N = day of the week,
P = power-break code.

Read task period
Segment : 6C (user task segment Ctrl),
Type of object : 64 (task period),
Object address : 1 → Interrupt task,
2 → Fast task,
3 → Master task,
4 → Auxiliary task 0,
5 → Auxiliary task 1,
6 → Auxiliary task 2,
7 → Auxiliary task 3,
Quantity : 0 by default,
Response : Task period encoded in one byte (1 to 255) respecting the time base for each task (FAST = 1ms, MAST = 1 ms and AUXi = 10 ms). For the IT task, the response corresponds to the number of EXEC cycles triggered.

Read internal bits

Segment : 64 (internal bit segment),
Type of object : 5 (internal bit with forcing),
Object address : logical number of the first internal bit,
Quantity : number of bits to read, modulo 8,
Response : array of n bits containing the status of the bits followed by another array of n bits indicating whether the corresponding bit is forced or not.

Read objects (continued)

Read date and time (in hexadecimal notation)

Segment	:	10 (common object segment),
Type of object	:	0 by default,
Object address	:	2 → current date and time in hexadecimal notation,
Quantity	:	0 by default,
Response	:	Array of eight words indicating: milliseconds, seconds, minutes, hours, day, month, year, number of the day in the week.

Read text block parameters

Segment	:	81 (function block segment),
Type of object	:	0 (text block),
Object address	:	logical number of the first text block,
Quantity	:	number of consecutive text blocks to read,
Response	:	table of bits and words, giving for each text block:

TXTi,D	:	bit (1 = done),
TXTi,E	:	bit (1 = error),
Indirect	:	bit (1 = indirect text block),
Distant	:	bit (1 = remote text block),
Not defined	:	4 bits, not significant,
Type	:	0 = TXT, 1 = CPL, 2 = TER, 3 = SYS, 5 = TLG,
TXTi, A	:	word,
TXTi, M	:	word,
TXTi, T	:	word,
TXTi, C	:	word,
TXTi, R	:	word,
TXTi, S	:	word,
TXTi, L	:	word.

The text blocks updated in the IT or FAST tasks may be read when they contain apparently incoherent data. This is caused by this request being processed in the Master task, a task with a lower priority level than the IT or FAST tasks.

Read objects (continued)

Read a CTRL block

Segment : 81 (function block segment),
Type of object : 1 (Ctrl block),
Object address : logical number of the first Ctrl block,
Quantity : number of consecutive Ctrl blocks,
Response : byte array arranged as follows:

Task status		
not configured	:	0
STOP	:	1
RUN	:	2
breakpoint	:	3
software error	:	4
active task	:	Bit 0 = active, Bit 1 to 7 not significant,
period	:	0 to 255. For the IT task, this field corresponds to the number of times the task was activated since initialization of the application.

Read a rack mounted I/O module

Segment : 82 (I/O module segment),
Type of object : 0 (module in the rack),
Object address : module address defined as follows:

bits 8 to 11	:	station number,
bits 3 to 6	:	rack number,
bits 0 to 2	:	module number,

the other bits are not significant,
Quantity : 1,
Response : Array of bytes structured as follows:

- error byte: refer to the read I/O module memory image request,
- configuration byte: refer to the read I/O module memory image request,
- byte giving the configured extension code,
- byte giving the physical status (bit 0 = acknowledgment error, bit 1 = parity error, the other bits are not significant),
- byte giving the extension code of the physical module.

7.4-2 Write Objects

This request writes simple objects (words, word strings, etc.) using the UNITEWriteObject function.

Request format

Request Code Hex.	Category code	Segment	Type of Object	Object Address		Number of Objects to Write		Data
37/55	0 → 7							

Segment: Specifies the addressing mode and the addressing field (in hexadecimal notation):
10: Common object space segment,
64: Internal bit space segment,
68: Internal word space segment,
69: Constant word space segment,
6C: Ctrl user task space segment.

Type of object: Specifies the object to write:
5: internal bits,
7: 16-bit signed integer,
8: 32-bit signed integer,
64: Task period.

Object address:
- The physical or logical address in the segment,
- The sequence number of the object in the segment:
  - 0 : Current date and time, in the common segment,
  - 1 : Terminal port configuration in the system segment.

Confirm format

Positive confirm

Confirm Code Hex.
FE/254

Write objects (continued)

Negative confirm

Confirm Code Hex.
FD/253

Causes for rejection:
- Unknown request,
- Inadequate access rights,
- Unknown object,
- Address of the last object out of range.

Request Examples

Write words or double words

Segment : 68 (internal word segment),
Type of object : 7 → Wi, or 8 → DWi,
Object address : index of the first Wi or DWi to write,
Quantity : number,
Response : array of n objects.

Write constant words or constant double words

Segment : 69 (constant word segment),
Type of object : 7 → CWi, or 8 → CDWi,
Object address : index of the first CWi or CDWi to write,
Quantity : number,
Response : array of n objects.

Write Date and Time

Segment : 10 (common objects segment),
Type of object : 0 by default,
Object address : 0 → current date and time,
Quantity : 0 by default,
Data : 17 ASCII characters describing the date and the time:
YYYYMMDDHHMMSS.TN
YYYY = year,
MM = month,
DD = day,
HH = hour,
MM = minute,
SS = second,
T = tenth of a second,
N = day of the week.

Write task period

Segment	:	6C (Ctrl user task segment),
Type of object	:	64 (task period),
Object address	:	1 → Interrupt task, 2 → Fast task, 3 → Master task, 4 → Auxiliary task 0, 5 → Auxiliary task 1, 6 → Auxiliary task 2, 7 → Auxiliary task 3,
Quantity	:	0 by default,
Response	:	New task period respecting the time base for each task (FAST = 1ms, MAST = 1 ms and AUXi = 10 ms).

Write internal bits

Segment	:	64 (internal bit segment),
Type of object	:	5 (internal bits),
Object address	:	logical number of the first internal bit,
Quantity	:	number of bits to write, modulo 8,
Response	:	array of n bytes containing the status of the bits followed by the value of eight bits (bit forcing cannot be written).

7.4-3 Application Initialization

This request is used to Stop a PLC when a software error occurs, using the UNITERequest generic function.

Request format

Request Code Hex.	Category Code
EF/239	0 → 7

Warning: Depending on the type of product, prior reservation may be required.

Confirm format

Positive confirm

Confirm Code Hex.
FE/254

Negative confirm

Confirm Code Hex.
FD/253

Causes of rejection:
- Unknown request,
- Inadequate access rights,
- Non reservation.