MCL-*Link Lite* User Guide

What is MCL-*Link Lite*

MCL-Link Lite from Symbol Technologies is a secure communication protocol for P460/P360 Memory and P470/P370 Cordless Scanners only. This free software enables two-way data communication between the scanner and a host application. MCL-Link Lite is typically used in conjunction with applications created on MCL-Designer. Note that MCL-Link Lite does not need a license to operate.

MCL-Link Lite is a subset of the full MCL-Link product.

MCL-Link Lite provides:

Guaranteed secure two-way communication between the scanner and host

Automatic error detection and retransmission of corrupt/lost data

Secure communication = Higher baud rates = Faster downloads

Two-way communication between the scanner and host application

The following restrictions apply to MCL-Link Lite:

- Communicates with Symbol Phaser Units only
- Supports of Phaser ID 001 and 002 only
- Only the following commands are implemented:
    - Send File
    - Send packet
    - Receive File
    - Receive Packet

For advanced communication features such as fully automated execution of task and application upgrades, ODBC connectivity, keyboard wedge support, Windows DLL support and much more consider upgrading to the full "MCL-Link" version.

This manual describes all the functionalities of the full MCL-Link product.
If you are using MCL-Link Lite, disregard the commands and options that are not available in the MCL-Link Lite version.

Contents

Chapter 1 – Getting Started

1.1. About This Guide

The MCL-Link User Guide provides general information about operating the MCL-Link application, configuring the software, and using MCL-Link commands.

The concept of 'Terminal'

The word 'Terminal' as used in this manual covers all the devices that can support the MCL-Link protocol. This can be a desktop terminal, a fixed wall mounted terminal, a printer, etc.

This manual does not refer to any specific terminal.

Notational Conventions

The following conventions are used in this document:

- "Operator" and "User" refer to anyone using the MCL-Link software.
- "PC" refers to the IBM personal computer or compatible system that you are using to develop applications.
- "Terminal" refers to various types of terminals.
- "You" refers to the administrator or person who is using this guide as a reference aid to install, configure, and/or operate the software.
- Keystrokes in bold type indicate non-alphanumeric keystrokes.
- For example: Select the <**F1**> key on the terminal to access on-line help.
- **Bold** type identifies menu items and input or text fields on a terminal screen.
- *Italics* are used:
  - for the names of parameters in function prototypes and variable names in usage
  - and syntax descriptions to highlight specific items in the general text
  - to identify chapters and sections in this and related documents.
- Square brackets [ ] in a command line enclose optional command line parameters.
- The piping symbol | is used to separate inline parameters on a command line.
- Bullets (•) indicate:
  - action items
  - lists of alternatives
  - lists of required steps that are not necessarily sequential
- Sequential lists (e.g., those that describe step-by-step procedures) appear as numbered lists.

1.2. Introduction

MCL-Link is Windows NT/ 95 / 98 / 2000 and XP batch communication server designed to support terminals running batch applications created using MCL-designer.

MCL-Link is the software tool that ensures access to and delivery of both programs and data essential to the smooth operation of an enterprise that relies on batch data collection.

MCL-Link communicates with your terminals and scanners either through a simple RS232 direct connection, for remote access, using a modem, or via an internal Ethernet connection.
Either the server or the batch device can initiate communications.

MCL-link provides the ability to handle all the complex tasks needed in a batch communications environment. Users can easily and quickly upload or download files and programs either to or from a batch device. In addition, MCL-Link enables users to:

- synchronize the terminal with the host computer
- query the terminal's status
- query terminal directories and files structures
- perform file operations, such as file lookup, save, file append, file copy, file rename, and program execution.

MCL-Link's powerful instruction set gives you the ability to create complex communication scripts with virtually no need for human intervention on either the device or the server. This is due to MCL-Link's ability to perform actions based upon instructions it receives from the device (through the serial line or modem) or from a program on the server.

MCL-Link includes an SQL engine that provides the ability to access and perform SQL request on database using ODBC.

1.3. Installing MCL-Link

Prerequisites

Before you attempt to install MCL-Link, make sure that your system conforms to the following system requirements:

Windows 98 SE, 2000, XP are all supported.

Intel PII-compatible processor, with 500 MHz minimum processor speed

128 MB RAM.

Installing the software

Proceed as follows to install MCL-Link:

Insert the MCL-Collection CD into your CD-ROM drive.

Click Start on the Windows task bar.

Select Run....

Type "D:\setup.exe", where D is the drive letter of your CD-ROM drive, and then click OK.

Follow the instructions in the MCL-Collection Installation Wizard.

You can also download the installation file from our website: http://www.mcl-collection.com.

Activating MCL-Link

For the Activation procedure instructions, please refer to the "MCL product Activation Manual".

1.4. MCL-Link Operation

MCL-Link Operating modes

MCL-Link can receive commands:

- From the script file MCLLINK.CMD.

Any host application can write commands in the script file. Every second, MCL-Link checks if the script file exists, and executes the command(s) stored in it. At the end of the file, the file is renamed MCLLINK.BAK. The MCLLINK.CMD file must be created in the MCLLINK.EXE directory (See *About MCL-Link Files* on page 1-21).

- From the user buttons.

On the supervisor's screen, buttons activate specific commands like:

  - " request terminal status
  - " send date and time to the terminal
  - " send programs and data files to the terminal
  - " request programs and data files from terminal.

- From the terminal.

MCL-Link is able to receive commands from terminal. The program running in the terminal controls the communication process. (See *Commands Received From the Terminal* on page 2-28).

- From the DLL.

MCL-Link is able to receive commands from the DLL via named pipes. (See *MCLLink DLL* on page 2-39).

1.5. Starting MCL-Link

MCL-Link can be started one or several times (instance) on the same PC following the type of the license used.

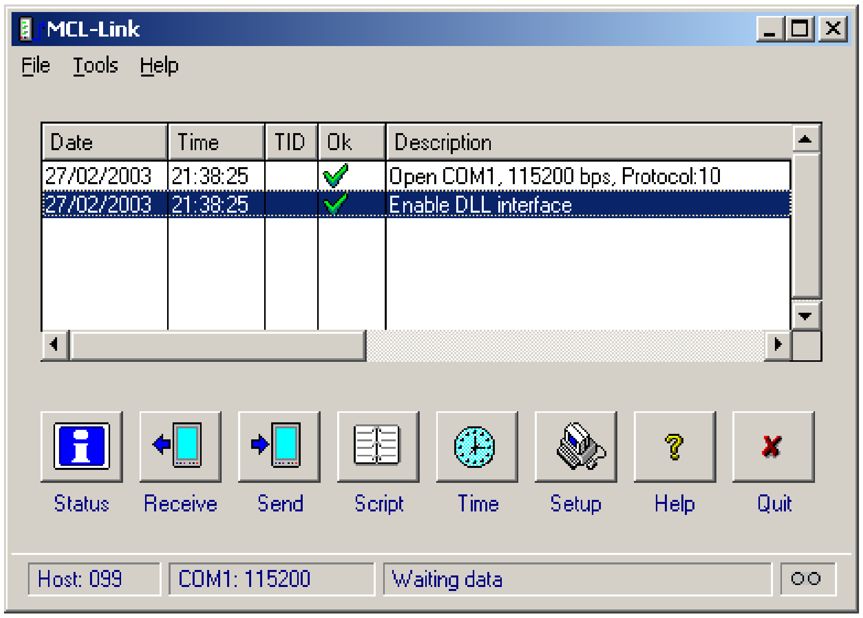
Each instance of the MCL-Link can manage one serial communication port of the PC or an Ethernet connection.

1.5.1. Starting MCL-Link – Single Instance

There are three ways to start the MCL-Link program:

- double-clicking on the MCL-Link icon on the desktop,
- from the standard Run window,
- from an application.

The MCL-Link window displays



Date	Time	TID	Ok	Description
27/02/2003	21:38:25		✓	Open COM1, 115200 bps, Protocol:10
27/02/2003	21:38:25		✓	Enable DLL interface

Host: 099 | COM1: 115200 | Waiting data | OO

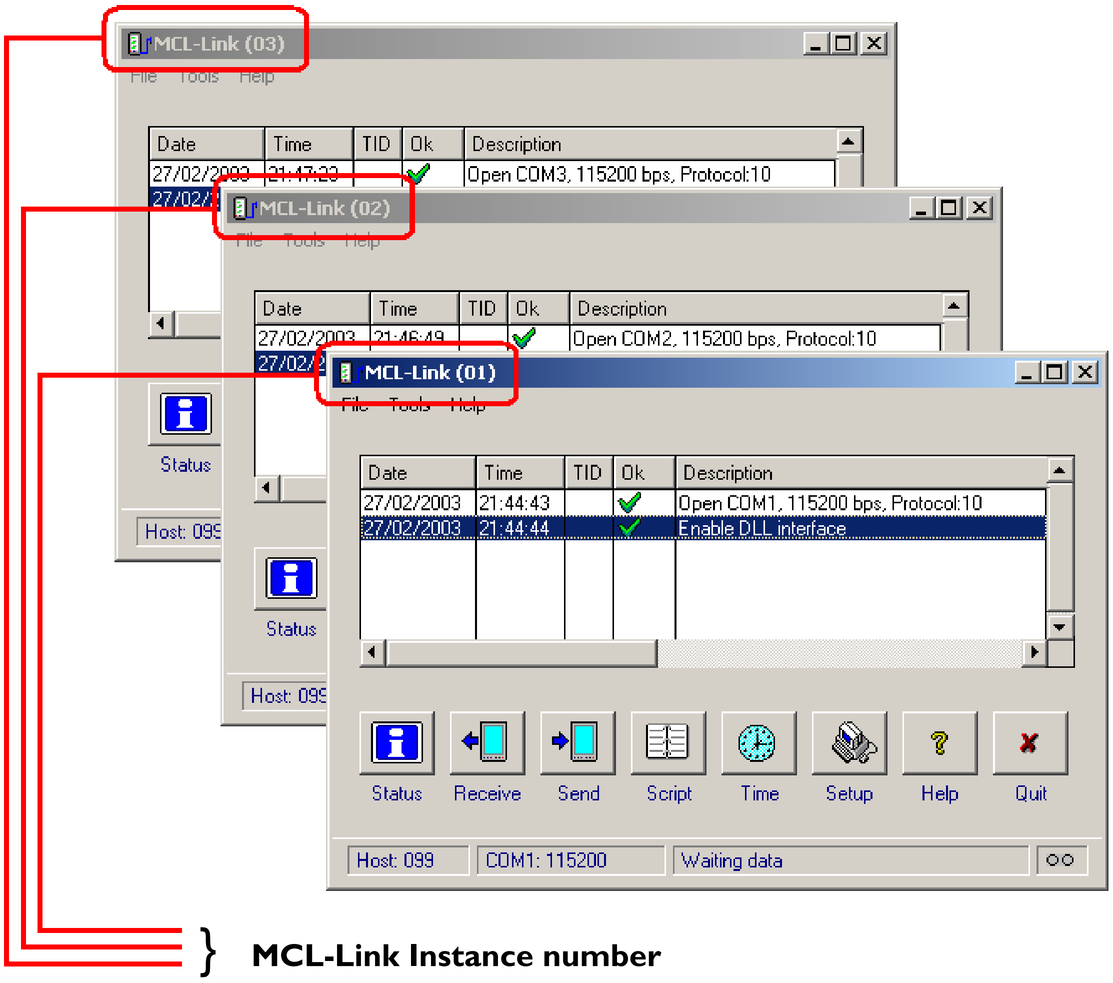
Figure 1-1. MCL-Link Window

Button Descriptions

Status	Requests terminal status.
Receive	Receives data from the terminal.
Send	Sends data to the terminal.
Script	Associates a specified Script file to a terminal.
Time	Enables MCL-Link to synchronize the terminal's time.
Setup	Sets the host communication parameters.
Help	Opens the MCL-Link on-line help application.
Quit	Terminates the MCL-Link program.

1.5.2. Starting MCL-Link – Multicom

To define an instance of MCL-Link, create a new MCLLINK.EXE shortcut on the Windows desktop. Click right on the new shortcut, select Properties and select shortcut tab. Modify the Target entry to add the instance argument (**-1** starts instance 1, **-2** starts instance 2, …**-8** starts instance 8 of MCLLINK.EXE).



MCL-Link (03)

Date	Time	TID	Ok	Description
27/02/2003	21:47:23		✓	Open COM3, 115200 bps, Protocol:10

MCL-Link (02)

Date	Time	TID	Ok	Description
27/02/2003	21:46:49		✓	Open COM2, 115200 bps, Protocol:10

MCL-Link (01)

Date	Time	TID	Ok	Description
27/02/2003	21:44:43		✓	Open COM1, 115200 bps, Protocol:10
27/02/2003	21:44:44		✓	Enable DLL interface

```
Host: 099 | COM1: 115200 | Waiting data | oo
```

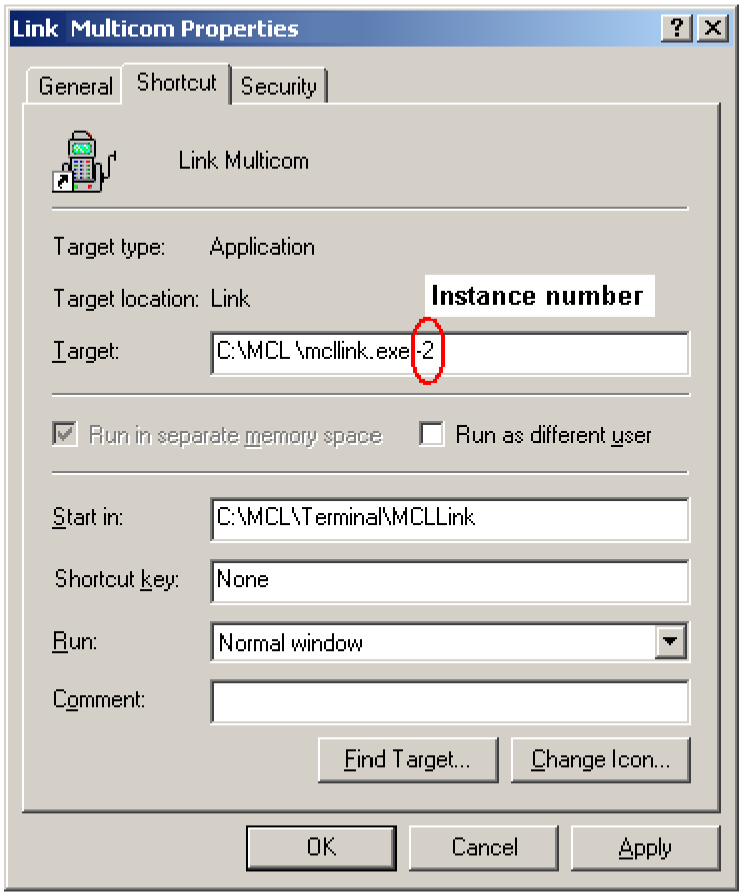
Figure 1-2. MCL-Link Multicom Window

Communication Port Assignment

When MCL-Link Multicom port is used, by default, the argument defines the PC communication serial port that MCL-Link uses.

Example: Argument –2 = MCL-Link uses PC communication serial port 2.

The user is able to modify the communication port and its settings using the setup button in the MCLLink main window.



Target type: Application

Target location: Link

Target: C:\MCL \mcllink.exe -2

☑ Run in separate memory space ☐ Run as different user

Start in: C:\MCL\Terminal\MCLLink

Shortcut key: None

Run: Normal window

Comment:

Figure 1-3. MCL-Link Shortcut Tab

Each instance of MCL-Link has its own configuration file (INI) and its command file (CMD).

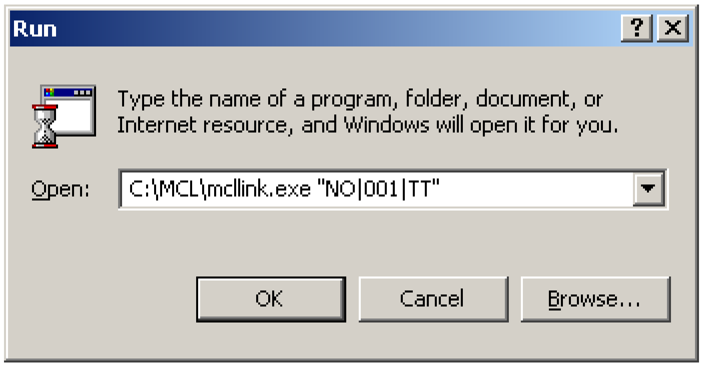
Example: Instance 2 of MCL-Link has its corresponding MCLLNK02.INI file, where all settings defined by the user are stored for this instance.

1.5.3. Starting MCL-Link with Arguments

Commands can be sent to a terminal from a command line.

The command is added to the MCL-Link program name as an argument:

1. Click on the `Start` button in the Task bar.
2. Select **Run**.
3. In the *Open* field, type the path and name of the MCL-Link program and the command (see Figure 1-4).
4. Each command must be sent as an argument so that it executes directly at MCL-Link start. The arguments must be embedded within double quotes characters.


Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
OK
Cancel
Browse...

```
C:\MCL\mcllink.exe "NO|001|TT"
```

Figure 1-4. Run MCL-Link with Arguments

1. Click on the OK button to run the MCL-Link program.
2. The MCL-Link window displays (Figure 1-1).

1.6. Communication Setup

Introduction

The communication setup is dependent on the terminal connected to the PC and the type of connection used.

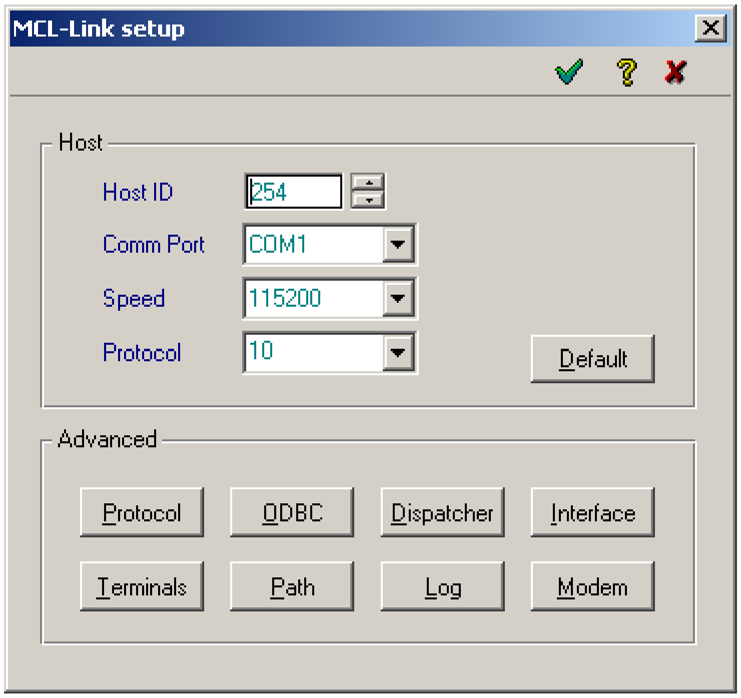
The setup includes different settings like: the protocol type, the speed of the communication line, the timeout value etc…

Default	1.6.1. Host and default protocol settings
Protocol	1.6.2. Set the different protocol settings
ODBC	1.6.3. Define the ODBC connections
Dispatcher	1.6.4. Define the route of the packet received
Interface	1.6.5. Define the MCL-Link interface
Terminal	1.6.6. Define the terminal's configuration
Path	1.6.7. Setup the path for the data and system files
Log	1.6.8. Define the Log files settings
Modem	1.6.9. Define the modem initialization commands

1.6.1. Host and Default Settings

To access the different setup functions, click on the "*Setup*" button.

The "*MCL-Link Setup*" window appears:



Host ID: 254
Comm Port: COM1
Speed: 115200
Protocol: 10

Figure 1-5. MCL-Link Setup window

Setup the Host ID

The "Host and Default settings" define the Host ID and the communication port used.

The Host ID is used to identify the host during the communication with the terminals. All terminals and host using MCL-Link must have a defined ID.

The Host ID, like the terminal Ids can have any value between 001 and 254.

Setup the Comm. Port

In the *Comm Port* list box, select the communication port used to connect the terminals. The communication port can be one of the serial com. ports from Com.1 up to Com.9 or the Ethernet connection.

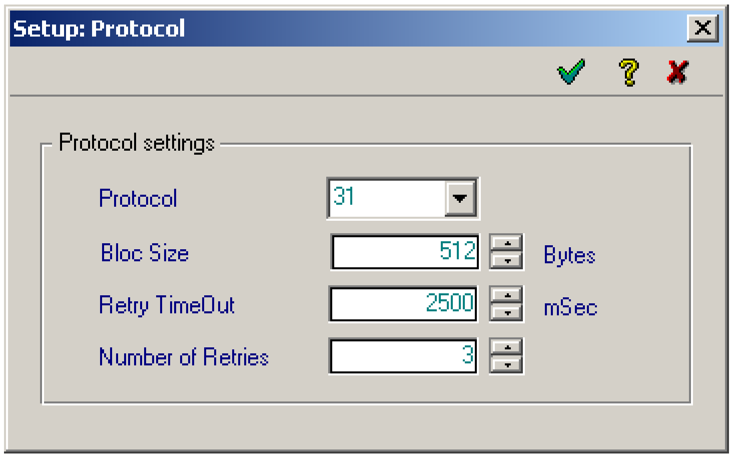
Setup the Default Protocol

The "Default settings" are dependent on the terminal type selected. The "Default settings" includes the following: the protocol, the speed, the timeout value and the number of retries.

1.6.2. Protocol

To set up the specific Protocol of the connection, click on the Setup – Protocol button in the MCL-Link window.

The *Setup: Protocol* window appears.



Setup: Protocol

Protocol settings

Protocol	31	
Bloc Size	512	Bytes
Retry TimeOut	2500	mSec
Number of Retries	3	

Figure 1-6. MCL-Link Setup: Protocol window

The Protocol settings

The “Protocol” type defined in the MCL-Link setup must match the protocol used by the terminal during the communication.

The “Bloc Size” defines the maximum bloc size that MCL-Link will send to the terminal during the communication. The standard default value is 512 bytes.

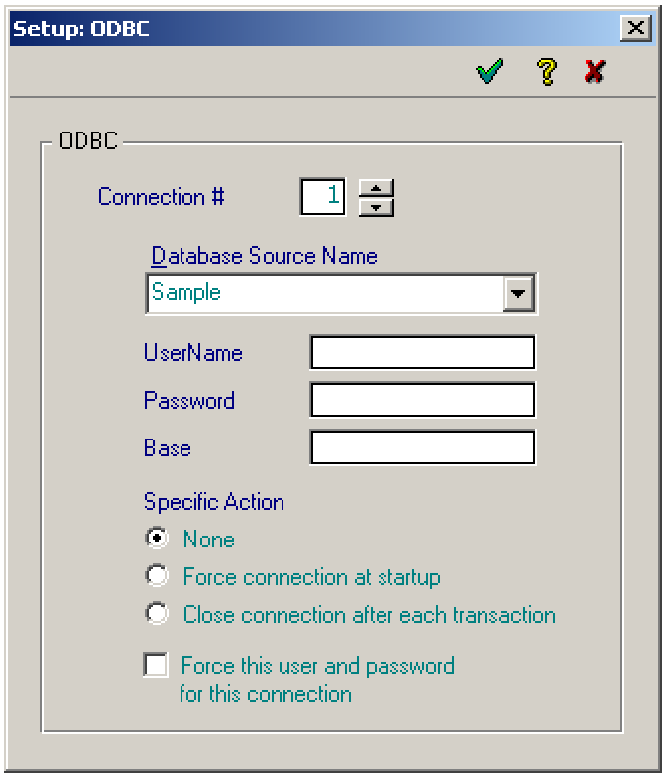
The “Retry Timeout” is used by MCL-Link when a reply from the terminal must be received. This timeout represent the number of milliseconds MCL-Link will wait for the acknowledgement of the bloc. The default value is generally 2.500 mSec.

The “Number of Retries” indicates how many retries MCL-Link will send to the terminal before declaring that the transmission failed. The default value is generally set to 3 retries; this means 4 attempts to send the bloc.

1.6.3. ODBC

This setup screen allows the definition of the MCL-Link ODBC connection.

Click on the "ODBC" button in the Setup screenan the following window appears:


Setup: ODBC

ODBC

Connection # 1

Database Source Name: Sample

UserName

Password

Base

Specific Action

- [✓ None]
- Force connection at startup
- Close connection after each transaction

☐ Force this user and password for this connection

Figure 1-7. MCL-Link Setup: ODBC window

ODBC connection setup

This window allows you to define up to 9 ODBC connections with different database or with different User Name and password. The "Database Source Name" list box present the different database DSN defined with in the Windows ODBC manager. When the connection requires a User name and a Password, this information may be defined here. Some database also requires an additional entry like a Base definition.

A specific action for each database connection may also be defined here:

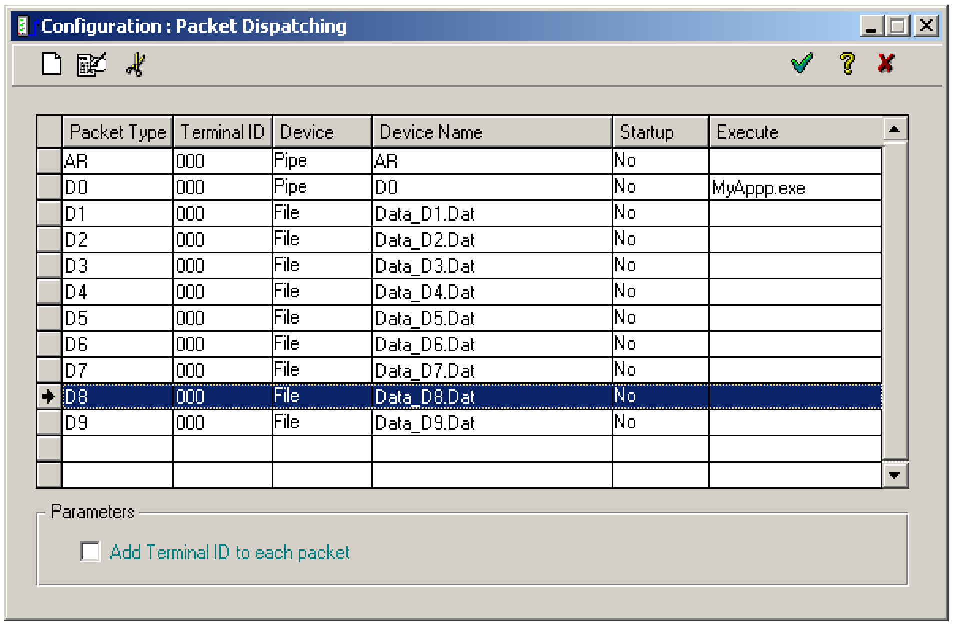
- Force the connection defined at startup of MCL-Link
- Close the connection after each transaction. This makes the connection available for other process.
- None of the above

The option "Force this user and password for this connection" defines MCL-Link as the User during the connection instead of the application within the terminal.

1.6.4. Dispatcher

The Configuration Packet Dispatcher table shows to the user the definition of the different packet setting.

For every packet type/terminal ID association, the configuration table define the action to perform: send the data to a file or send the packet to a pipe to have an interactive connection with a user defined application.



Configuration : Packet Dispatching

Packet Type	Terminal ID	Device	Device Name	Startup	Execute
AR	000	Pipe	AR	No	
D0	000	Pipe	D0	No	MyAppp.exe
D1	000	File	Data_D1.Dat	No	
D2	000	File	Data_D2.Dat	No	
D3	000	File	Data_D3.Dat	No	
D4	000	File	Data_D4.Dat	No	
D5	000	File	Data_D5.Dat	No	
D6	000	File	Data_D6.Dat	No	
D7	000	File	Data_D7.Dat	No	
D8	000	File	Data_D8.Dat	No	
D9	000	File	Data_D9.Dat	No	

Parameters

☐ Add Terminal ID to each packet

Figure 1-8. Packet Dispatching window

Packet Dispatching Table

The Dispatching Table contain the following information:

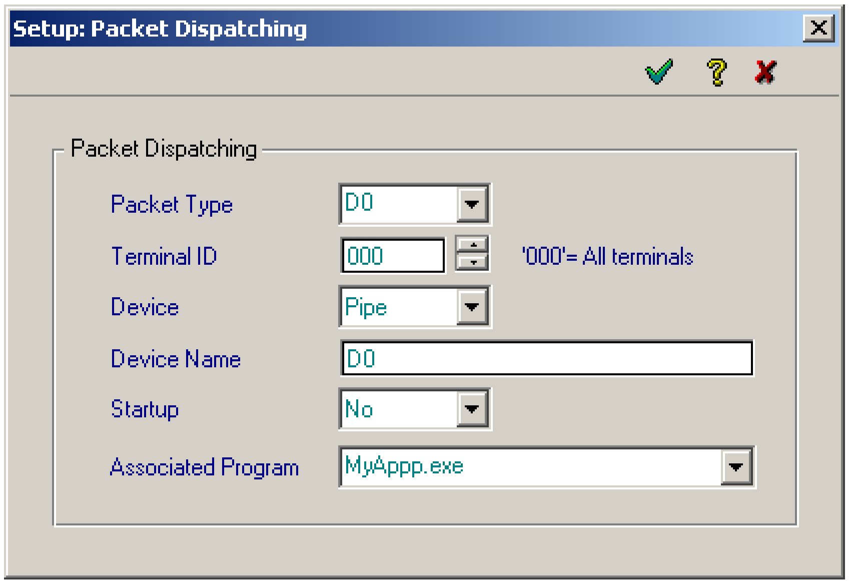
- The Packet type and the Terminal Id that dispatcher will use to route the packets. When the terminal ID is set to “000” this means “all terminals”.
- The device identifies whether the data will be stored in a file or send to a pipe
- The Device Name is the name of the destination file or the destination pipe.
- Startup and Execute indicate if the file defined in the Execute column must be started automatically at MCL-Link startup.

When necessary the dispatcher settings may be modified or new definitions can be added.

Packet Dispatching definition

To create a new entry in the dispatcher table, or to modify an existing entry, click on the New, Edit or Delete icons in the upper left corner of the Dispatcher table.

The "*Setup:Packet Dispatching*" window will then appear:



Packet Type	D0
Terminal ID	000 '000'= All terminals
Device	Pipe
Device Name	D0
Startup	No
Associated Program	MyAppp.exe

Figure 1-9. Setup: Packet Dispatching window

Packet Dispatching

The Packet Type must be selected from the selection list.

The Terminal ID can be either a specific terminal ID or "000". The "000" defines that all terminals are considered.

The Device entry can be

- "Pipe" : The data will be passed directly to a user application,
- "File" : The data will be stored sequentially in a file on the host,
- "None" : The data will be disregarded.

The Device Name defines the name of the Pipe or host File

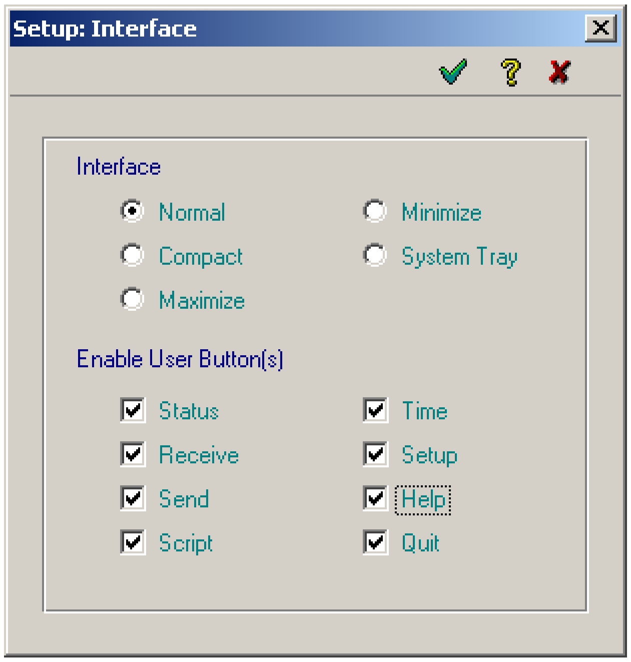
The Startup option determines if the Associated Program must be started automatically when MCL-Link is started or not.

The Associated Program is the name of the user application that will receive the data send to the pipe by Dispatcher.

1.6.5. Interface

The Interface dialog box allows the user to define how the MCL-Link window will be displayed.

This interface setup also allows the user to Enable or Disable the function buttons at the bottom of the MCL-Link window.



[✓ Normal] Compact Maximize Minimize System Tray

Enable User Button(s)

[✓ Status] [✓ Receive] [✓ Send] [✓ Script] [✓ Time] [✓ Setup] [✓ Help] [✓ Quit]

Figure 1-10. Setup: Interface window

Interface setup

The Interface setup determines the size of the window and buttons.

When the "Compact" interface is selected, a only small buttons in a very small window appear on the screen.

When "Maximize" Interface is chosen, the MCL-Link window will work in use the full screen

The "Minimize" interface will remove the MCL-Link window from the screen. It will run minimized in the Taskbar.

The "System Tray" option start MCL-Link without any window on the screen and only a small icon is displayed in the system tray.

Enable User Button(s)

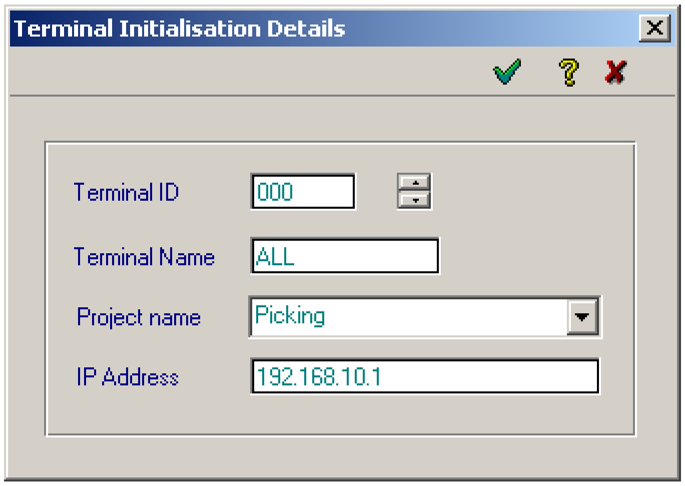
This part of the setup Enable or Disable the checked buttons.

Deselect the check boxes to disable the corresponding buttons.

1.6.6. Terminal

The Terminal Setup window allows the user to define the project to download to the terminal when the terminal sends the special "project request" command to MCL-Link. This setup defined the name of the project to download.

Click on the `Terminal` button and the *Terminal* dialog box appears.



Terminal ID	000
Terminal Name	ALL
Project name	Picking
IP Address	192.168.10.1

Figure 1-11. Setup: Terminal Configuration Dialog Box

Field Description

The Terminal ID identifies the terminal to receive the project. If Terminal ID "000" is defined, this means that all terminals not defined in another Terminal Configuration line will receive the project defined here.

The Terminal Name is only mentioned here as a help to remember the terminal groups. This entry is not used by MCL-Link.

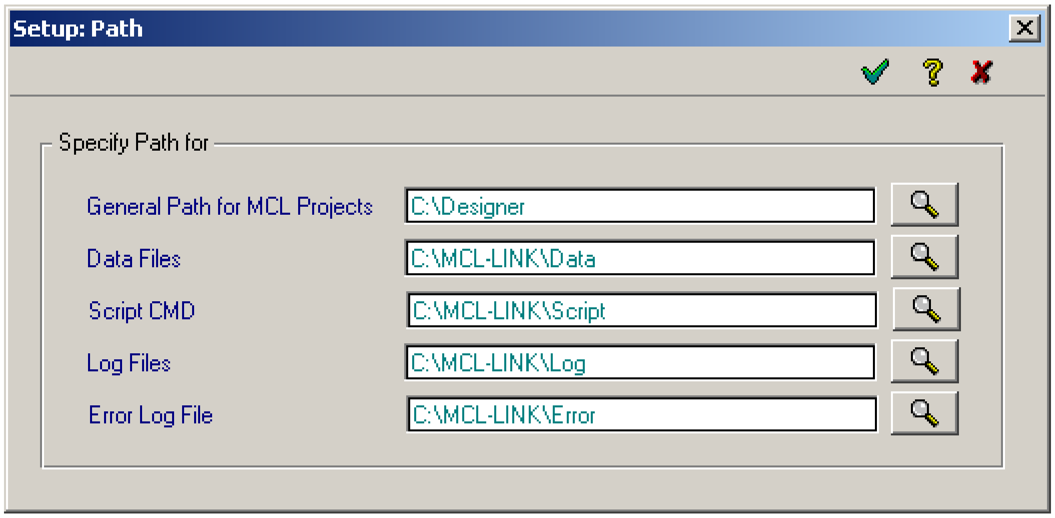
The Project Name allows the definition of the project to download to the terminal.

The IP Address must be defined here when the Ethernet connection is used.

1.6.7. Path

The Setup Path window allows you to define the different paths used by MCL-Link.

Click on the *"Setup – Path"* button, the following window appear:



General Path for MCL Projects	C:\Designer
Data Files	C:\MCL-LINK\Data
Script CMD	C:\MCL-LINK\Script
Log Files	C:\MCL-LINK\Log
Error Log File	C:\MCL-LINK\Error

Figure 1-12. Setup: Path Dialog Box

General Path for MCL Projects

To access an SQL database through ODBC, the terminal sends a specific request («DR» data packet) that contains the project name, the SQL query number and the data to MCL-Link. The project name is used to locate the project.mqd file where all the SQL command definitions are stored.

MCL-Link transfers this request to the MCL-Link OBDC/SQL engine that extracts the right SQL query definition from the project.mqd file. The project.mqd contains all queries generated by MCL-Designer that allows you to access data in a specified database for a specific project. The project.mqd file is stored in the specified project directory. When the MCL-Link ODBC engine finds (or does not find) the corresponding data in the database, it sends a return code and data to the terminal through the MCL-Link. The return code indicates the number of records that are found in the database. Selecting the ODBC connection at startup automatically connects MCL-Link to the specified database.

Path for Data Files

This path indicates where the data files received from the terminals or requested by the terminal will be located, if no specific path is defined.

Path for Script Files

This path indicates where the script files used by MCL-Link are stored. The script can be launched either by the "Start Script" button on the User interface or directly under the control of a terminal sending a "Run Script" command to MCL-Link.

Path for Log Files

This path indicates where the log files created by MCL-Link will be located. The Log file definition are setup in the “Setup – Log” dialog box.

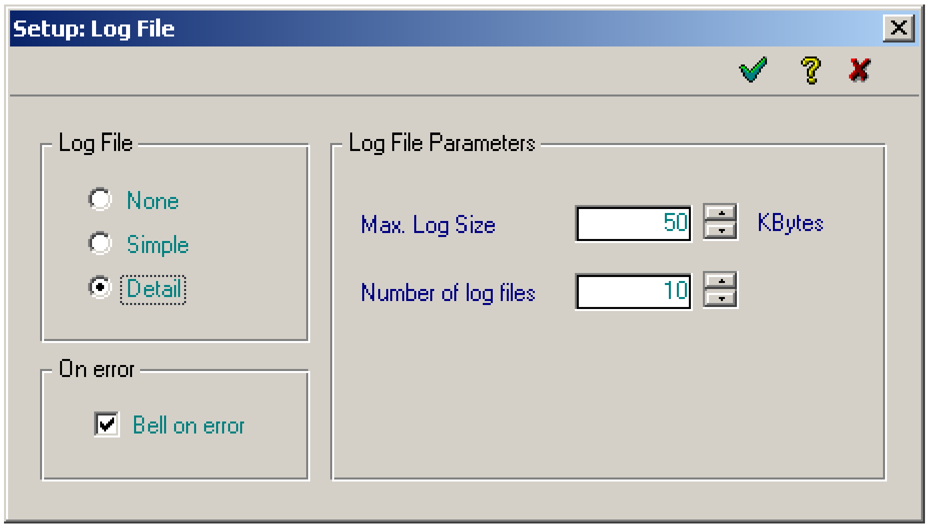
Path for Error Files

This path indicates where the Error files generated by MCL-Link will be stored. The Error files contain all the command that where not executed correctly by MCL-Link.

1.6.8. Log

The Log file dialog box allows you to define the options for the capture of the traffic between the terminal and MCL-Link.

Click on the *"Setup – Log"* button and the following window appear:


Setup: Log File

Log File
- [ ] None
- [ ] Simple
- [✓] Detail

On error
- [✓] Bell on error

Log File Parameters

Max. Log Size: 50 KBytes

Number of log files: 10

Figure 1-12. Setup: Log Dialog Box

Log File setup

The Log file setup defines the format of the captured data in the log files.

The log file can stored be "Simple" or "Detailed" information about the traffic between the terminal and MCL-Link. The "Simple" option only stores the commands that are executed by MCL-Link. The "Detailed" option stores the detail of every packet exchanged between the terminal and MCL-link.

Log File Parameters

This option defines the number of log files and the size of each file created in the Log directory during the communication.

The On error setup

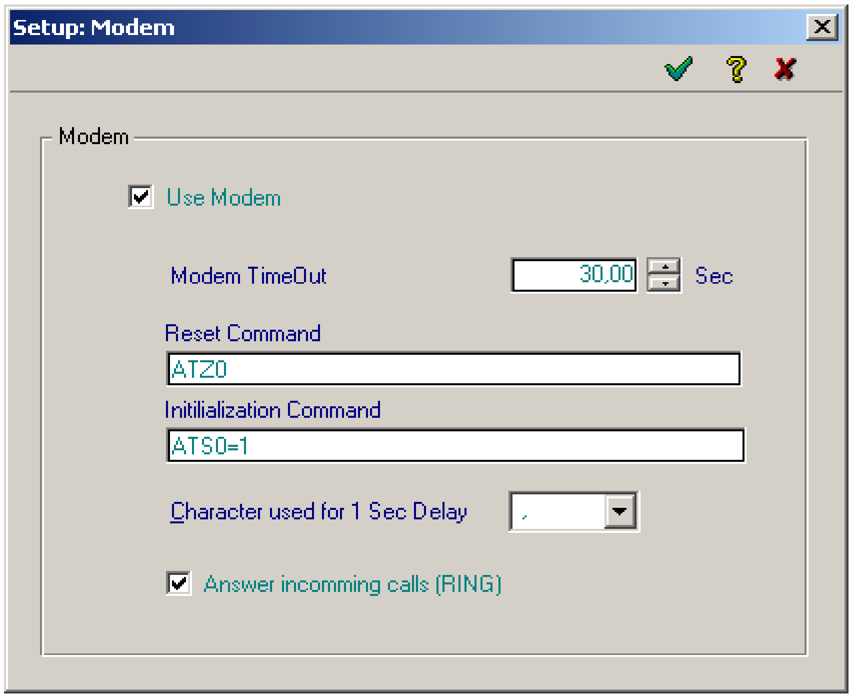
The Bell on error defines whether MCL-Link will activate the bell when an error occurs.

1.6.9. Modem

The Modem setup window defines if a line modem is used between the terminal and MCL-Link.

This window also defines the modem initialization command that MCL-Link will send at startup.

Click *"Setup – Modem"* and the following window appear:



☑ Use Modem

Modem TimeOut 30,00 Sec

Reset Command
ATZ0

Initilialization Command
ATS0=1

Character used for 1 Sec Delay ,

☑ Answer incomming calls (RING)

Figure 1-13. Setup: Modem Dialog Box

Modem setup

Select the "Use Modem" box if a modem is connected to the com.port between the terminal and MCL-Link. If a modem is used, the Modem Timeout must be defined. This timeout is used whenever a command is sent to the modem and MCL-Link is waiting for the status on the command.

The "Reset" command must be used to ensure that the modem is correctly setup and is waiting for the initialization command.

The Initialization command is generally used to place the modem in "Auto Answer" mode at startup.

The "Character used for 1 sec delay" allows the definition of the character to use within the modem command when the modem must perform a 1 second pause before continuing the command.

The "Answer Incoming calls" defines whether the modem or MCL-Link will reply to the calls.

1.7. Using MCL-Link

The MCL-Link main window contains buttons to perform all the different functions necessary for maintaining the terminals.

The buttons can be “enabled” or “disabled” by the use of the “Setup – Interface” option.

The buttons allows the user to activate the selected MCL-Link function.

Status	1.7.1. Send a status request to the terminal
Receive	1.7.2. Send a File request to the terminal
Send	1.7.3. Send a Project, Program or File
Script	1.7.4. Start a MCL script on the host
Time	1.7.5. Send a Time synchronization command

1.7.1. The Status button

Click the Status button to initiate the status request operation for a specific terminal.

The *"Terminal Status"* dialog box appears.

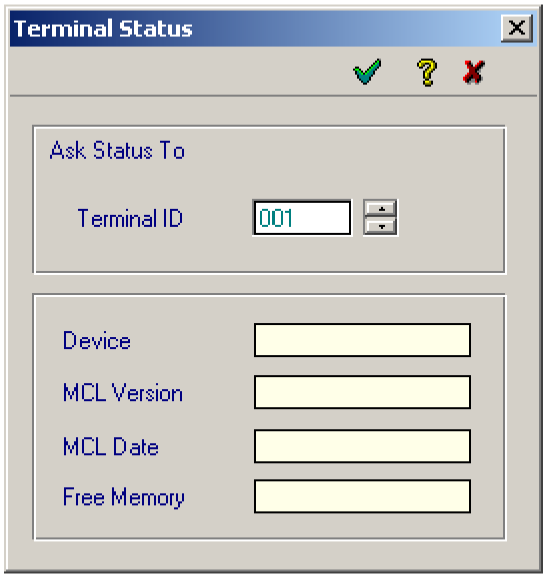


Figure 1-14. Terminal Status window

The Status request

Using this button, the operator request MCL-Link to send a "Status Request" command to the terminal.

The operator selects the terminal ID and click on the ✓ button to execute the command.

The terminal answers to the status request with a status packet.
The status packet contains:

- The device type
- The MCL version running in the terminal
- The current date
- The memory available in the terminal.

1.7.2. The Receive button

Click the Receive button to initiate a File Request to be send to the terminal.

The *"Receive"* dialog box appears.

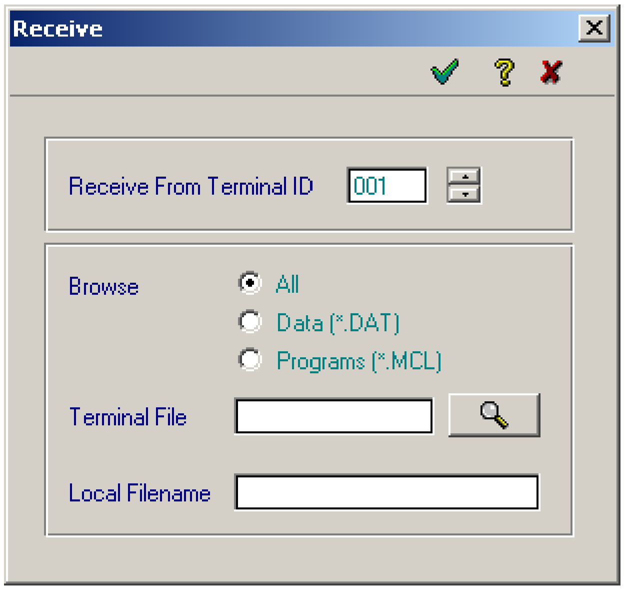


Figure 1-15. Receive File window

File Request

Enter or select the terminal ID of the terminal connected to the com. Port.

The *Magnification* icon requests a directory status from the terminal and allows the user to select the desired file to be uploaded. Only the files with data will be presented. Empty files are ignored by this command.

The *Browse* field determines which files will be presented in the requested directory. If "All" is selected, all files will be presented.

If "Data" is selected, only the .dat files will be presented. The .mcl files will not be part of the directory.

If "Programs" is selected, only the .mcl files will be presented. The .dat files will not be part of the directory.

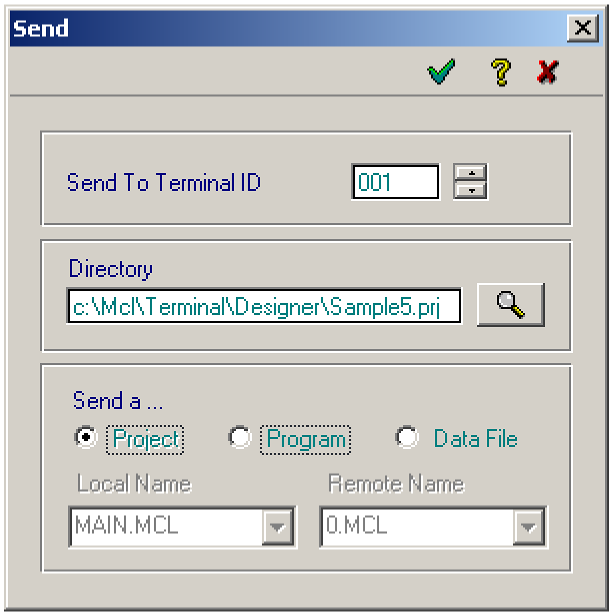
The "Terminal File " is the source file.

The "Local Filename" is the name that MCL-Link will assign to the received file on the PC. The file received will be placed in the directory for the Data files as defined in the "Path" setup screen.

1.7.3. The Send button

Click the Send button to initiate a send File or Project to the terminal.

The *"Send"* dialog box appears.


Send

Send To Terminal ID 001

Directory c:\Mcl\Terminal\Designer\Sample5.prj

Send a ...

[✓ Project] Program Data File

Local Name MAIN.MCL

Remote Name 0.MCL

Figure 1-16. Send File or Project window

Send File or Send Project Request

Enter or select the terminal ID of the terminal connected to the com. Port.

The Magnification button opens a local (computer) directory structure and allows the user to select another directory to locate the file(s).

The Send a... field determines what type of file to send to the terminal.

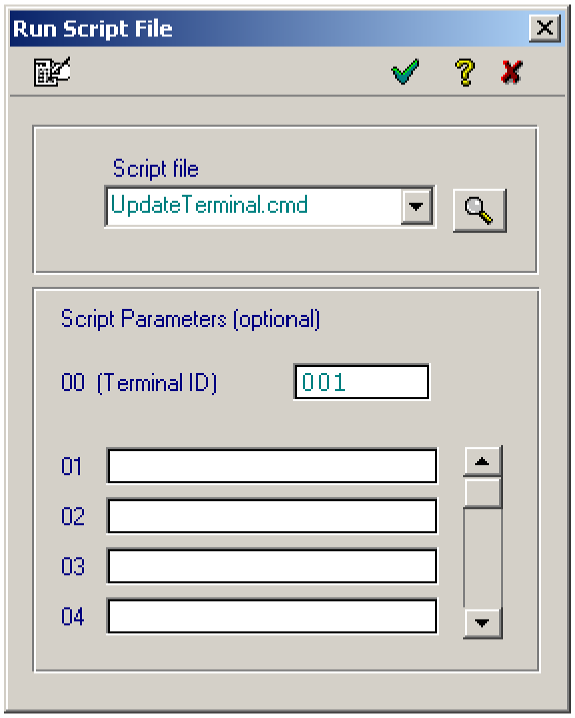
The Local Name field states the name of the file on the local computer, and Remote Name states the name that the file has on the terminal.

When the selected directory contains a project, the "Send Project" option is available.

1.7.4. The Script button

Selecting the Script button on the main window provides the ability to associate a script file (a list of commands) to a specified terminal.

The *"Script"* dialog box appears.



Script file: UpdateTerminal.cmd

00 (Terminal ID): 001

Figure 1-17. Start Script window

Start an MCL script

The Magnification button next to the *Script file* field opens a command window on a local computer and allows the user to select a file (.CMD).

Click the button to view the script file in a text editor such as Notepad. If a file is not selected using the Magnification button above, a window appears that allows the user to select the script file.

The Script parameters fields are optional. They are transferred to the script file as arguments.

For example, &00 written in script file represents the terminal ID, &01 represents the script parameter 01, etc.

1.7.5. The Time button

Selecting the Time button provides the ability to synchronize the terminal. The date and hour are updated.

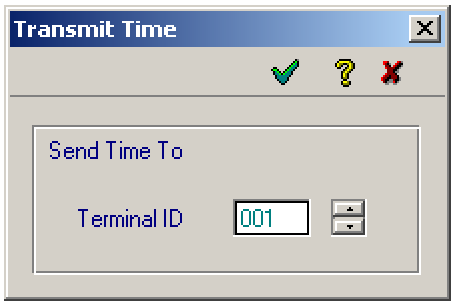


Figure 1-18. Transmit Time Window

Transmit date and Time

The transmit Time can be used to force the time synchronization with the host.

The command is handled directly by the MCL-Client on the terminal. The Time packet will not be passed to the application.

The Time and date of the terminal are automatically updated when the message is received by the terminal.

1.8. MCL-Link Files

The main MCL-Link files

MCLLINK.EXE	The executable program.
MCLLINK.HLP	The Windows Help file.
MCLLINK.INI	Contains the Setup information for MCL-Link.
MCLLINK.STA	Contains all status received from the terminals.
MCLLINK.ERR	Contains all transactions errors.
MCLLINK1.LOG	Contains the last Log sessions
MCLLINK2.LOG	
MCLLINK3.LOG	

When the size of MCLLINK1.LOG reaches the maximum size defined in the Setup - Log screen, it is copied into MCLLINK2.LOG which is itself copied into MCLLINK3.LOG. The data contained in the LOG files is dependent upon the parameter setup in the Setup – Log screen.

1.8.1. MCL-Link Log File

The MCL-Link Log file contains data depending on the Setup Log file settings.

The structure of the records in the MCLLINK.LOG file are explained hereafter:

```
2003/01/10 03:05:24.03  << ..  [STX] 05 63 DT 1|0[ETX] <1:0
2003/01/10 03:05:24.06  >> ..  [STX] 63 05 AK[ETX] 1?91
2003/01/10 04:06:33.08  CM .. NO|01|TT|20030110040633|5
2003/01/10 04:06:33.12  << ..  [STX] 01 63 TT 2|20030110040633|5[ETX] 5300
2003/01/10 04:06:35.66  << R1[STX] 01 63 TT 2|20030110040633|5[ETX] 5300
```

Column 1	=	date and time of the transfer of this record
Column 2	=	**<<** message send to the terminal **>>** message received from the terminal CM command received by MCL-Link – From the operator, From the script From the terminal R1 … R5 : send retries – up to "n" retries depending on the MCL-Link setup ER MR Error message – MR = Max retries reached
Column 3	=	[STX] Start of text at the beginning of all packets sent or received

Column 4 = Addresses, Packet type and sequence number

First address (2 charac. Hex value) destination ID

Second address (2 charac. Hex value) source ID

Packet Type :

- TT = Time Transmit
- AK = Acknowledgement of the packet transmitted
- RJ = Packet rejected. Packet received without error but Link cannot handle it.
  - Packet type not supported
  - No memory available to store the packet
  - ODBC: project.mqd file not found or SQL command not defined
  - Invalid sequence number
- D0 …. D9 = data packet D0 to D9
- TF = transmit file packet
- FR = File request
- DR = Database Request command (Terminal to Host)
- DT = Database transaction (Host to Terminal)
- Etc … (see MCL-Link manual for all the commands)

Sequence Number = used to detect duplicate packets

Column 5	=	Data transferred
Column 6	=	[ETX] End of text at the end of all packets sent or received and the CRC of the packet.

1.8.2. MCL-Link Error File

The MCLLINK.ERR file contains all transactions errors. The first two letters identify the error, followed by the date, time, terminal ID, and command.

Error When Initializing MCL-Link

CA	Cancel by User
DE	Demo mode (no communication)
PO	Open port comm. Error. Choose another comm. port or close the application that uses this comm. port (close the DOS Box if needed).
PI	Initialize port comm. Error. Choose another comm. port or close the application that uses this comm. port (close the DOS Box if needed).

Error In Command File

LB	Label not found in a Command file. Correct the Command file (refer to Chapter 2, *Command File*).
CM	Bad or unknown command in a Command file. Correct the Command file (refer to Chapter 2, *Command File*).

Error In Transaction - Error In Command When Files Are Implied

ID	Bad terminal identification. The command received from the terminal is incorrect. Correct the MCL program.
TO	Time Out error, no response from the terminal or the modem.
RJ	Reject received.
MR	Max retry.
XX	Unknown command received from the terminal. The command received from the terminal is incorrect. Correct the MCL program.
SS	Sub-directory creation error. The path of the file is incorrect.
FF	File not found.
FN	File Name incorrect.
FO	File open error.
FS	File error (bad structure, format, etc.).
FE	File Operation error during copy, rename, append, delete. Check file's existence, directory, disk space, etc.

Error Code When Executing Another Program (EX or NO|xxx|EX)

Error When OBDC is Used

01	Initialization error.
02	Program already finished.
03	Already in use.
04	Running too many programs (50 max).
05	Too much memory message (400 max).
06	Unknown destination.
07	No return message.
08	Not enough memory.
09	Unknown program or path.
10	Program is already running.
11	Initialization error.
12	Memory allocation error.
13	DDE initialization error.
14	Error when connecting to the server.
15	DDE transaction error.
16	DDE received error.
O1	ODBC source not defined.
O2	ODBC connection failed.
O3	ODBC error on 16 bits systems when using ODBC 32 bits.
O4	ODBC SQL command not defined.
O5	ODBC command error.
O6	ODBC error on file execution.

1.8.3. Troubleshooting

If you encounter any problems:

- Check that the data is sent correctly from your host (terminal ID, filename,…).
- Check that power is correctly applied to the cradle or PIM.
- Check that your terminal is powered ON and in MCL-Link mode.
- Check communication parameters on the terminal and host computer.
- Parameters must be set the same on both the terminal and the host computer.
- Check the RS232 parameters on host side (connected to good communication port, etc.).
- Check your RS232 cable.

Chapter 2 - Command File

2.1. Introduction

A command file can provide instruction to MCL-Link. This file contains a list of commands to execute. There can only be one command per line.

The command file MCLLINK.CMD must be created in the current MCLLINK directory. This file is checked by MCLLINK.EXE several times every second.

The commands are subdivided into REMOTE commands and LOCAL commands.

Table 2-1 lists the REMOTE commands that are sent from the host computer to the terminal.

Note: xxx is the terminal ID (from 001 to 254).

NO|xxx|TF	Transmit File
NO|xxx|FR	File Request
NO|xxx|TT	Transmit Time
NO|xxx|RZ	Reset
NO|xxx|SR	Status Request
NO|xxx|FC	File Copy
NO|xxx|FN	File Rename
NO|xxx|FA	File Append
NO|xxx|FD	File Delete
NO|xxx|QX	Quit MCL-Link

Table 2-1. Remote Commands

Table 2-2 lists the LOCAL commands that are performed on the host computer.

WT	Wait
QX	Quit the MCL-Link script
QT	Quit the MCL-Link script on timeout
FN	File Rename
FC	File Copy
FD	File Delete
FA	File Append
EX	Execute Program
SK	Skip
LB	Define a Label or Mark
IF	Test and Branch
MD	Modem command
**	Comment line

Table 2-2. Local Commands

2.2. Syntax of Commands

The syntax of the commands with the MCL-Link script files uses the following characters and field names.

| (pipe)	= Separator ASCII 124.
\	= Backslash character ASCII 92.
xxx	= Terminal ID or Host ID (range: 001 to 254).
Terminal_File_Name	= The File Name in the terminal
PC_File_Name	= The File Name on the PC Side. Contains the directory and file extension (if no directory is specified, the current one is used).
\mcllink\data	= Default Path for the data files.
.dat	= Default File Extension for data files
.mcl	= Default File Extension for MCL program file.

Note : Within the MCL commands, the file names (data files and program files) must always be encoded in lower case.

2.3. Remote Commands

2.3.1. Transmit File (TF)

Function

Transmits a file to the terminal. The file can be any file type like the data files, the image files or program files.

Syntax

NO|xxx|**TF**| PC_File_Name|Terminal_File_Name

where:

xxx	= Terminal ID (range: 001 to 254).
PC_File_Name	= Full name that contains directory and file extension (if no directory is specified, the current one is used).
Terminal_File_Name	= define the name of the file when downloaded in the terminal.

Example

NO|001|**TF**|c:\mcl\project.prj\datafile.dat|aa.dat

2.3.2. Data File Request (FR)

Function

Requests a file from the terminal. The file can be any file type like the data files, the image files or program files.

Syntax

NO|xxx|**FR**|Terminal_File_Name|PC_File_Name

where:

xxx	= Terminal ID (range: 001 to 254).
PC_File_Name	= Full name that contains directory and file extension (if no directory is specified, the current one is used).
Terminal_File_Name	= Name of the file in the terminal.

Example

NO|001|**FR**|aa.dat|c:\mcl\project.prj\data\users.dat

2.3.3. Transmit Current date and Time (TT)

Function

Transmits current time and date to the terminal.

Syntax

NO|xxx|**TT**

where:

xxx	= Terminal ID (range: 001 to 254).

Example

NO|001|**TT**

2.3.4. Reset Terminal (RZ)

Function

Resets the terminal.

Syntax

NO|*xxx*|**RZ**|*0*

NO|*xxx*|**RZ**|*1*|*x*

NO|*xxx*|**RZ**|*2*

NO|*xxx*|**RZ**|*2*|*Filename*

NO|xxx|**RZ**|*4*

where:

xxx	= Terminal ID (range: 001 to 254).
0	= warm re-boot.
1	= MCL program.
2	= all data files or A to P file.
4	= cold re-boot.
x	= program number (0 through 10).

Example

NO|001|**RZ**|2

.

2.3.5. Status Request (SR)

Function

Requests status from the terminal.

Syntax

NO|*xxx*|**SR**|*1*

NO|xxx|**SR**|*3*|*File_Name*

where:

xxx	= Terminal ID (range: 001 to 254).
File_Name	= a file in the terminal (A through P) for data files.
1	= terminal.
3	= a file.

Example

NO|001|**SR**|3|A

The response to an SR command is an ST status transmit command. The file MCLLINK.STA contains all the status (ST) frames received from the terminal(s).

2.3.6. File Copy (FC)

Function

Copies one file on the terminal into another file on the terminal.

Syntax

NO|xxx|**FC**|Old_file|New_file

where:

xxx	= Terminal ID (range: 001 to 254).
Old_file	= name of the file to be copied.
New_file	= name of the file old_file is copied into

Example

NO|001|**FC**|AA|BA

File **AA** is copied to file **BA**. Files **AA** and **BA** are identical after the copy

2.3.7. File Rename (FN)

Function

Renames a file on the terminal.

Syntax

NO|xxx|**FN**|*Old_file*|*New_file*

where:

xxx	= Terminal ID (range: 001 to 254).
Old_file	= name of the file to be renamed.
New_file	= new name for file old_file.

Example

NO|001|**FN**|AA|BA

File **AA** is renamed File **BA**

2.3.8. File Delete (FD)

Function

Deletes a file from the terminal.

Syntax

NO|xxx|**FD**|file_Name

where:

xxx	= Terminal ID (range: 001 to 254).
file_Name	= name of the file to be deleted

Example

NO|001|**FD**|AA

File **AA** is deleted in the terminal

.
.

2.3.9. File Append (FA)

Function

Adds a file to another file on the terminal.

Syntax

NO|xxx|**FA**|*File_1*|*File_2*

where:

xxx	= Terminal ID (range: 001 to 254).
File_1	= name of file to be added to File_2.
File_2	= name of file that File_1 is added to.

Example

NO|001|**FA**|AA|BA

File AA is added to File BA. File A remains unchanged

2.3.10. Exit MCL-Link (QX)

Function

Exits MCL-Link program on the terminal.

Syntax

NO|xxx|**QX**|1

NO|xxx|**QX**|2

where:

xxx	= Terminal ID (range: 001 to 254).
QX1	= Exit MCL-Link Client in the terminal and continue the program with the next process line.
QX2	= Exit MCL-Link Client in the terminal and restart MCL in the terminal.

Example

NO|001|**QX**|1

2.4. Local Commands

2.4.1. Wait (WT)

Function

Adds a delay to the command file.

Syntax

WT|*Time_in_second*

where:

Time_in seconds	= amount of time to wait.

Example

WT|10

Wait 10 seconds before continuing to the next command.

2.4.2. Quit (QX)

Function

Closes the MCL-Link program.

Syntax

QX

Example

QX

MCL-Link terminates immediately

2.4.3. Quit on Time Out (QT)

Function

Closes the MCL-Link program after a set time of no activity.

Syntax

QT|*Time_in_seconds*

where:

Time_in seconds	= amount of time with no activity before closing the MCL-Link program.

Example

QT|10

Close the MCL-Link program after 10 seconds of no activity with the terminal.

2.4.4. Label (LB)

Function

Defines a label in the command file.

Syntax

LB|*Label*

where:

Label	= name of the label.

Example

LB|START

Defines the label *START* in the MCLLINK.CMD command file.

2.4.5. Skip (SK)

Function

Goes to a label in the command file.

Syntax

SK|*label*

SK|+2, **SK**|-3

where:

label	= name of the label to go to or the number of lines above (-) or below (+) the current line.
+2, -3	= are used to jump directly to the corresponding number of lines.

Example

SK | START

Go to the label *START* in the command file. The command Skip and Label are used together to make branches and loops in a command file.

2.4.6. Test and Branch (IF)

Function

Tests the value of a variable and then branches to a label upon condition.

Syntax

IF|&99|=|0|*Label_if_ok*|*Label_if_not_ok*

IF|&98|=|*xxxx*|*Label_if_ok*|*Label_if_not_ok*

where:

xxxx	= value to test variable against.
Label_if_ok	= label to go to if the condition is true.
Label_if_not_ok	= label to go to if the condition is false.

Example

IF|&99|=|0|START|ERROR

IF|&98|=|CONNECT|+1|ERROR

After each command, &99 contains "0" if the instruction had terminated correctly otherwise &99 contains "1".

Script Example

LB|START	define de label "START"
NO|099|**TF**|AA|data.dat	send file "AA" to host 099 and name it data.dat on host.
IF|&99|=|0|+1|ERROR	test the status of the previous command. If variable 99 = 0 : continue, if not "0", branch to label "ERROR".
WT|10	wait 10 seconds before continuing
SK|START	skip to label START
LB|ERROR	define de label "ERROR"
QT|10	Close MCL-Link

2.4.7. Modem Commands (MD)

Function

Sends a modem command to the modem connected to the serial port.

Syntax

MD|*Modem_command*

where:

Modem_command = modem command sent to the terminal

Example

MD|ATZ0	(reset the modem)
MD|ATDT1234567	(dial a number)
MD|,+++,ATH0	(disconnect the line)

A comma in the modem command represents a 1 second delay. Time out on MD commands is 30 seconds. The MD Modem command sets &98 variable with the response of the modem (i.e., OK, 0, CONNECT, etc.)

Script Example

LB|START	define de label "START"
MD|ATZ0	reset the modem
MD|ATE0M0V0	initialize the modem
MD|ATDT123456787	dial the number 123456787
IF|&98|=|CONNECT|+1|ERROR	test if the line is connected
WT|10	wait 10 seconds before continuing
....	execute the transfer commands
SK|START	skip to label START
LB|ERROR	define de label "ERROR"
QT|10	Close MCL-Link

.

2.4.8. Comment Line (* *)

Function

Adds a comment to the command file.

Syntax

******| *xxx*

where:

xxxx	= the comment.

Example

******| THIS IS A COMMENT LINE

Defines the label *START* in the MCLLINK.CMD command file.

No action is performed when the MCL-Link command file interpreter meets this line.

2.4.9. Local File Copy (FC)

Function

Copies one file into another file on the host computer.

Syntax

FC|*Old_file*|*New_file*

where:

Old_file	= name of the file to be copied.
New_file	= name of the file Old_file is copied into.

Example

FC|ITEM.TXT|ITEM.BAK

The file ITEM.TXT is copied into ITEM.BAK

2.4.10. Local File Rename (FN)

Function

Renames a file on the host computer.

Syntax

FN|*Old_file*|*New_file*

where:

Old_file	= name of the file that is to be renamed.
New_file	= new name for file old_file.

Example

FN|ITEM.TXT|ITEM.BAK

The file ITEM.TXT is renamed into ITEM.BAK.

The file ITEM.BAK may not exist for this command to execute.

2.4.11. Local File Delete (FD)

Function

Deletes a file on the host computer.

Syntax

FD|*file_Name*

where:

file_Name = name of the file to be deleted.

Example

FD|ITEM.TXT

The file ITEM.TXT is deleted.

2.4.12. Local File Append (FA)

Function

Adds a file to another file.

Syntax

FA|*File_1*|*File_2*

where:

File_1	= name of file to be added to File_2.
File_2	= name of file that File_1 is added to.

Example

FA|ITEM.TXT|ITEM.BAK

The file ITEM.TXT is added to file ITEM.BAK. File ITEM.TXT remains unchanged.

2.4.13. Execute (EX)

Function

Executes a program on the host computer.

Syntax

EX|*program_file_Name_and_Its_arguments*

where:

program_file_Name_and_Its_arguments	= name of program to run and any arguments.

Example

EX|NOTEPAD.EXE MyNote.DOC

The program Notepad.exe is run and the argument MyNote.DOC is the file that opens.

2.5. Commands Received From the Terminal

The following commands are MCL packets sent by the terminal to the host computer.

2.5.1. Receive Data Packet (D0)

Function

Sends data to appropriate data file.

Syntax

D0|*data*

where:

data	= data added to the DATA_DX.DAT file

Example

NO|099|**D0**|19971010|231022|1111

D0 = Data is appended in DATA_D0.DAT File in the general path.

D1 = Data is appended in DATA_D1.DAT File in the general path.

. . .

D9 = Data is appended in DATA_D9.DAT File in the general path.

The DATA_Dx.DAT file is created if it doesn't exist.

2.5.2. File Look-Up (CR) from Terminal

Function

Looks for data in a file on the host.

Syntax

CR|*File_Name*|*Key_to_search*

where:

File_Name	= name of file to search.
Key_to_search	= data to search for

Example

NO|099|**CR**|parts.dat|123456789012

File_Name is a text file sorted on a key, the key must be the first field in the file and the records length must be constant.

MCL-Link sends to the terminal a **CT** command with data from the record.

CT|0 = key not found

CT|1|Data_from_file = key found, data are the rest of the line

CT|9 = file not found

2.5.3. Transmit Remote Data File (TF) from Terminal

Function

Transmits a data file to the host computer.

Syntax

TF|*Terminal_File_Name*|*PC_File_Name*

where:

Terminal_File_Name	= named of the file in the terminal.
PC_File_Name	= Full name that contains directory and file extension (if no directory is specified, the current one is used).

Example

NO|099|**TF**|AA|c:\data\datafile.dat

The File “AA” is send from the terminal to the host. The file on the host is renamed “datafile.dat”.

2.5.4. Remote Data File Request (FR) from Terminal

Function

Requests a data file from the host computer.

Syntax

FR|*PC_File_Name*|*Terminal_File_Name*

where:

PC_File_Name	= Full name that contains directory and file extension (if no directory is specified, the current one is used).
Terminal_File_Name	= The name of the file in the terminal.

Example

NO|099|**FR**|c:\data\datafile.dat|AA

The terminal sends a request for “datafile.dat”.

2.5.5. Host Status Request (SR) from Terminal

Function

Requests status from the host computer.

Syntax

SR|*1* query MCL-Link status

SR|*3*|*PC_File_Name* query a file status

where:

1	= MCL-Link.
3	= a file.
PC_File_Name	= a file on the host computer.

Example

NO|099|**SR**|3|c:\data\item.dat

Request the status of the file c:\data\item.dat.

2.5.6. Host File Copy (FC) from Terminal

Function

Copies a file on the host computer.

Syntax

FC|*Old_file*|*New_file*

where:

Old_file	= name of the file to be copied.
New_file	= name of the file Old_file is copied into.

Example

NO|099|**FC**|item.dat|item.bak

The file "item.dat" is copied to file "item.bak". The two files are identical after the copy.

2.5.7. Host File Rename (FN) from Terminal

Function

Renames a file on the host computer.

Syntax

FN|*Old_file*|*New_file*

where:

Old_file	= name of the file that is to be renamed.
New_file	= new name for file Old_file

Example

NO|099|**FN**|item.dat|item.bak

The file “item.dat” is renamed “item.bak”.

A file with the name “item.bak” may not exist on the host before the rename is executed.

2.5.8. Host File Delete (FD) from Terminal

Function

Deletes a file from the host computer.

Syntax

FD|*file_Name*

where:

file_Name	= name of the file to be deleted.

Example

NO|099|**FD**|item.txt

Deletes the file "item.txt" from the host computer.

2.5.9. Host File Append (FA) from Terminal

Function

Adds a file to another file on the host computer.

Syntax

FA|*File_1*|*File_2*

where:

File_1	= name of file to be added to File_2.
File_2	= name of file that File_1 is added to..

Example

NO|099|**FA**|item.dat|item.bak

The file "item.dat" is added to file "item.bak".

The "item..dat remains unchanged.

2.5.10. Quit MCL-Link (QX) from Terminal

Function

Closes the MCL-Link program on the host computer.

Syntax

QX

where:

File_1	= name of file to be added to File_2.
File_2	= name of file that File_1 is added to..

Example

NO|099|**QX**

Close the MCL-Link program.

2.5.11. Execute a Program (EX) from Terminal

Function

Executes a program on the host computer.

Syntax

EX|"*program_file_Name*" *[argument_1 argument_2 ….]*

where:

program_file_Name	= name of program to run with complete path The path+program name must be embedded within double quotes.
[argument_1 argument_2 ….]	any argument(s) for the program.

Example

NO|099|**EX**|"c:\notepad.exe" my_note.txt

The program Notepad.exe is run and the argument MyNote.txt is the file that opens.

2.6. MCL-Link DLL

The MCL-Link provides a DLL interface to host applications in order to facilitate a client/server relationship between industrial terminals and a host application.

The main goal of this DLL is to receive operational terminal transaction data from the MCLLink and send transaction-related data to a specific terminal via MCL-Link.

The DLL enables you to check if MCL-Link is running, and lets you start or stop MCL-Link on your Windows environment.

2.6.1. DLL Conventions

The following conventions are used:

- int and long represents a 32-bit signed integer (range -2E31 to 2E31- 1)
- char represents an 8 bit character (range 0 to 255)
- int * represents a near pointer to an array of 32 bit signed integer
- char * represents a near pointer to an array of characters

2.6.2. DLL Functions

Function	**Description**
MCLLink_Start	Starts MCL-Link instance
MCLLink_Stop	Stops one MCL-Link instance
MCLLink_Check	Checks if MCL-Link is running
MCLLink_OpenPipe	Opens a named pipe
MCLLink_ClosePipe	Closes a named pipe
MCLLink_WaitData	Receives data from an MCL-Link instance
MCLLink_SendData	Sends data to a specified terminal
MCLLink_CheckTerminal	Checks the state of a specific terminal using an MCL-Link instance

2.6.3. Return Code Standard Values

The return code is a 32-bit signed integer.

A return code lower than 0 means that an error occurred.

The nine significant error values are explained here:

-10 Error - Thread not found

-9 Too many processes/threads use the DLL (max 128)

-8 Error - Terminal not defined

-7 Error - Terminal not connected

-6 Error - Terminal connected but not reachable

-5 Error - A parameter is invalid

-2 Error - System error

-1 Error - Timeout

0 Error - MCL-Link not started

2.7. MCL-Link DLL Functions

2.7.1. MCLLink_Start

Function

Starts MCL-Link instance (if not already started)..

Syntax

int MCLLink_Start(char *server, char *service, int arg)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.

Interface mode:

0 = normal window

1 = maximized window

2 = minimized window

Return Values

A return code greater than 0 indicates that MCL-Link instance is started correctly.

DLL function ordinal number: 2

2.7.2. MCLLink_Stop

Function

Stops one MCL-Link instance.

Syntax

int MCLLink_Stop(char *server, char *service)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.

Return Values

A return code greater than 0 indicates that MCL-Link is stopped correctly

DLL function ordinal number: 3

2.7.3. MCLLink_Check

Function

Checks if MCL-Link is running.

Syntax

int MCLLink_Check(char *server, char *service)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.

Return Values

A return code greater than 0 indicates that MCL-Link is running.

A return code < = 0 indicates that MCL-Link is not running.

DLL function ordinal number: 1

Example

Win95/Win98:

MCLLink_Check(«.», MCLLINK01)

Windows NT, 2000, XP

MCLLink_Check(«SERVER», «MCLLINK01»)

2.7.4. MCLLink_OpenPipe

Function

Opens a named pipe.

Syntax

int MCLLink_OpenPipe (char *server, char *service, char * buffer)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.
buffer	= the name of the pipe (zero terminated string)

Return Values

A return code greater than 0 indicates that has no error to open the specified pipe.

DLL function ordinal number: 9

Note

The name of the pipe must be defined in the dispatcher and MCL-Link must be started. The name of the pipe may not exceed 60 bytes.

2.7.5. MCLLink_ClosePipe

Function

Close a named pipe.

Syntax

int MCLLink_ClosePipe (char *server, char *service, char * buffer)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.
buffer	= the name of the pipe (zero terminated string)

Return Values

A return code greater than 0 indicates that has no error to close the named pipe.

DLL function ordinal number: 10

Note

The name of the pipe may not exceed 60 bytes.

2.7.6. MCLLink_WaitData

Function

Receives data from a MCL-Link instance.

Syntax

int MCLLink_WaitData (char *server, char *service, int * Term, long Timeout,

char * Cmd, char * Buffer, int Maxlen)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.
Term	= terminal number (range: 1 to 254).
Timeout	= timeout in msec.
Cmd	= the pipe name (defined in the Dx Packet setup).
Buffer	= data input receive buffer.
Maxlen	= maximum length of input buffer.

Return Values

A return code greater than 0 indicates that data has been received from the terminal which is connected and reachable.

The return code, if greater than 0, indicates the number of received characters.

DLL function ordinal number: 6

Note

1. Buffer that receives data must be declared to the effective maximum data length +1 because a 0 (hexadecimal) is added as terminator.

2. The variable Term is written with the terminal number after the execution.

3. Received command (Packet Type) is copied in Cmd after the execution of the

function.

2.7.7. MCLLink_SendData

Function

Sends data to a specified terminal.

Syntax

int MCLLink_SendData (char *server, char *service, int * Term, long Timeout,

char * Cmd, char * Buffer, int Len)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.
Term	= terminal number (range: 1 to 254).
Timeout	= timeout in msec.
Cmd	= the pipe name (defined in the Dx Packet setup).
Buffer	= data to send.
Len	= maximum length of data.

Return Values

A return code greater than 0 means that data has been sent correctly to the terminal; which is connected and reachable.

DLL function ordinal number: 7

Note

Timeout must be long enough in case of file or MCL-Code programs transfer to a specified terminal.

If Timeout is set to 0, the function MCLLink_SendData is not waiting for acknowledgement from the terminal. This command exits immediately

2.7.8. MCLLink_CheckTerminal

Function

Checks the state of a specific terminal using a MCL-Link instance.

Syntax

int MCLLink_CheckTerminal(char *server, char *service, int Term, long Timeout)

where:

server	= the name of the server for Windows NT only. Use «.» for Win95 and Win98 server.
service	= the name of the service created by the MCL-Link instance Instance 1 of MCL-Link creates a service named MCLLNK01, instance 2 creates service MCLLNK02, and so on to instance 8, which creates service MCLLNK08. When MCL-Link is started in single instance, the name of the service is MCLLINK.
Term	= terminal number (range: 1 to 254).
Timeout	= timeout in msec.

Return Values

A return code greater than 0 indicates that the terminal is connected.

DLL function ordinal number: 4