# Lastic GUI
# User Guide

Version V6.1
Jan 2013

# PREFACE

Lastic GUI provides a licence and royalty free facility to operate MS-Windows* software on PCs communicating with Host "M" systems over a wide variety of hardware platforms and "M" implementations.

The PC operates a native MS-Window routine which communicates with the host system running the Lastic Drivers.

In addition, the software permits the host application to perform multiple DDE conversations with PC applications transparent to the user thus any "M" application can send and receive information to any compatible PC application.

*MS-Windows is a trademark of Microsoft Corporation

# CONTENTS

# 1. INTRODUCTION

Lastic GUI provides an MS-Windows environment for your M applications without the need for an M database to be resident on the PC. In fact, the PC requires only minimal memory and disk space to operate Lastic GUI. This enables any PC using MS-Windows to run Lastic GUI.

By using a compressed, synchronised messaging mechanism, windows are built from a skeleton containing predefined control types.

## 1.1 CONTROL TYPES

Currently supported controls include:

| | |
|---|---|
| BitButton | Text and Glyph |
| Button | (Command button) |
| Check box | |
| Combo box | (Drop down list or editable) |
| Document | (multi-line scrollable control- optionally editable) |
| Frame | (encapsulate controls in frame or panel format with optional heading) |
| Grid | (scrollable display and optionally editable) |
| Hypertext | Multi-line, scrollable text with "clickable" links |
| Input | (Single line Text) |
| List Box | (single and multiple selection) |
| Meter | (Bar meter to show percentage processed) |
| Outline | (Tree structure data display/selection) |
| Output | (Uneditable single line variable text) |
| Picture | (icons, bitmaps and metafiles) |
| Radio button | (options buttons) |
| Select | ("Clickable" Output) |
| Splitter | Enables areas of the form to be resized |
| Text | (single line Label) |
| Workbench | (designer control for layouts etc.) |

In addition, the product also supports:

| | |
|---|---|
| DDE text | (up to 9 concurrent DDE conversations using text transfers) |
| Graph generation | (up to 99 concurrent graphical forms) |
| Printing | (direct to PC printers) |
| File transfers | (direct from Mumps to PC files or reverse - text or binary) |

## 1.2 BENEFITS OF THE LASTIC GUI

Minimal resources are utilised on the PC.
No application distribution/version control problems.
No licence requirements for PCs.
Faster than using DOS + Windows tasks.
Can incorporate a wide range of GUI controls.
Can hold up to 9 concurrent DDE conversations (send and/or receive).
Can link and invoke other Windows tasks.
Emulator option to perform legacy tasks in same session.
Direct Windows logon and program activation capability through script file.

# 2. PC INSTALLATION

The PC software consists of the routines and files

| | |
|---|---|
| LG.EXE | The primary run-time routine |
| LGSETUP.EXE | The set-up routine |
| LG.HLP | Help text (Emulator information); |
| LGSETUP.HLP | Help text (Setup information) |
| | |
| LGRAPH.EXE | Lastic Graphics routine |

## 2.1 FILE TRANSFER

Create a directory on your PC or server which will be the working directory for Lastic GUI.
From the installation diskette:

Copy LG.EXE
LGSETUP.EXE
LGRAPH.EXE
LG.HLP
LGSETUP.HLP

## 2.2 INSTALLING ON DESKTOP

### 2.2.1 Lastic GUI

Lastic GUI can be set up one or more times as an iconed application or menu entries where each instance can point to a different INI file or a different server within the same INI file thus enabling multiple concurrent operations using a network or multiple serial lines.
To install Lastic GUI:

i) Create a short cut to lg.exe with the working directory the same as where lg.exe resides.
ii) If it is to use the default INI file and the "preferred" server then no further change is required. Additional parameters, in any order may be added to the command line separated by a space.
iii) To use an alternative INI file append /P=xxx.ini to the file reference where xxx.ini is the INI file name to use.
iv) To use an alternative server to the "preference" append /S=server name which must be defined in the INI file being used.

To initially set up Lastic GUI run the set up routine (LGSETUP) and select the "NEW" option. This will create, in memory, a default setup which can then be modified and saved as an INI file.
Lastic GUI makes no use of your Registry thus to remove the routine merely delete LG.EXE etc from your system.

### 2.2.2 Lastic GUI Set up

The routine lgsetup.exe is the set-up routine which creates and modifies lg.ini.

## 2.3 INI FILE UPDATE

You should run the set-up routine should any new parameters require set-up. These will be notified in the release notes.

# 3. LASTIC GUI SET UP

The set up parameters can be modified using the set up routine lgsetup.exe.

Initially, a working set-up can be achieved by selecting FILE|NEW and saving the initialised lg.ini file. This initial set-up is configured for your PC's resolution and contains a winsocket connection to localhost (127,0,0,1 IP address).

When invoked a window is displayed with various pull down menu options. These are described below.

## 3.1 FILE

This menu provides options for existing parameters to be opened, saved or saved as another ini file.

## 3.2 COMMUNICATIONS

This option shows two sub-menu options: Servers and Preferences. The server option should initially be used to define connection requirements.

### 3.2.1 SERVERS

Selecting this option invokes a window showing a list of existing server connections by connection type and with the facility to edit those or to add new connections. Existing ones can be selected from the combobox and edited as required. New connections must be given a name and then edited for the required parameters.
Server connection types include Winsocket (TCP/IP) [Note Serial line and Telephony connection options have been removed from new versions].

#### 3.2.1.1 WINSOCKETS

For MS Windows connections through TCP/IP using the Microsoft standard API to the network stacks, the details for such a server connection are requested in this window. Lastic GUI enables various parameters to be defined for your connectivity requirements. Lastic GUI utilises the SOCK_STREAM socket type and operates in non-blocking mode.

**IP Address**
The TCP/IP address of the host system to be connected. This must be entered in the dot format.

**Port**
The port number should reference either the telnet service which is port number 23 or a server port (>2000) which corresponds to the master server port defined in the Lastic GUI Mumps server.

**Version Number**
Lastic GUI currently utilises version 1.1 of the WinSock API. However, since only a sub-set of the facilities are used it is unlikely that future versions will affect this operation. Enter 1.1 for the version number. This is the wVersionRequested item in the WinSock API.

**Address Format**
To allow for future extensions to the Winsock API options. Currently, only one format exists and this should be set to a value of 2. This is the Internet address format (PF_INET in the WinSock API which is currently the same as the family as described below).

**Socket Type**
This should be set to a value of 1 indicating "SOCK_STREAM".

**Protocol**
A setting of 0 (zero) indicates no specific protocol requirements and is the recommended value for this item.

**Family**
This must be a value of 2 corresponding to the WinSock API AF_INET constant defining the internet family.

**Sync Start, End & Text Separator characters**
These correspond to the definitions previously given above. They are not WinSock parameters but are the controls used by Lastic GUI within this server link and must correspond to those defined on the Host system defined by this connection. Lastic recommends using ASCII 28, 29 and 30 respectively. Do not use any printable character nor ASCII 127.

**Script File Name**
If a logon script file is to be used, enter the path and file name here. Details of script files are given in section 3.7.

**Auto Negotiate Telnet Options**
If you want Lastic GUI to perform TELNET options negotiations automatically, check this box. If not checked it assumes any options settings will be incorporated into a script file or that the logon does not require options negotiations.

### 3.2.2 PREFERENCES

This sub-menu option permits overall definition of communication parameters.

**Default Server**
This may be changed by selecting available server connections.

**Time Out**
This is the number of seconds LASTIC GUI will lock out windows messages when a focus change occurs and invokes an event on the host system. After this period, windows messages are enabled if the server (mumps) process has not yet responded to the GUI message. It is recommended that this be set to between 5 and 10 seconds.

**Auto Open**
If this is checked, the server connection is automatically opened when the application is started. If not checked then the user must select the "Link Open" option from the Communications menu in the emulator window before any interaction with the host is available.

**Monitor**
If this is checked then a communications monitor window is invoked displaying both send and received data. The monitor can be switched on and off via the emulator window Communications menu.

**Show script**
If checked and a script file is in use the script dialog is displayed, as it is actioned, in a document control within the logon banner form.

**Terminate on Disconnect**
Check this box to cause Lastic GUI, when in emulator mode, to terminate when the connections to the server is disconnected.

**Disable Emulator Close**
If Terminate on disconnect is checked then you can check this box to stop the emulator from closing while a connection is active. This can be used to stop users closing the emulator while applications are running.

**Application Title**
Up to 32 characters may be entered to replace "Lastic GUI" as the application title at run-time. This will replace the default title in the About box, emulator heading etc.

**Icon File Name**
To replace the Lastic GUI icon with one of your own choice, enter the full pathname of the icon file required.

## 3.3 FONTS

This window provides the means of specifying font requirements for each control type. The font names listed are those available on your PC. This also includes the default font for your printer when using the print to PC printer facility.

**Object**
Select the object type from the list box. The current details for this object type are displayed. Font names, sizes, bold and italic properties may be defined.

**Font Name**
The list shows those fonts available on your PC. You should be aware that MS-Windows could change your font selection depending upon the size selected at run-time thus consider the font size in relation to the control sizing (See SUNDRY below).

**Font Size**
This font size (point equivalent). Character pitch is automatically determined when the size (height) is changed.

**Bold & Italic**
Check these boxes as required.

When any changes are made the text sample is re-displayed using the selected values.

## 3.4 HIGHLIGHTING

The option enables RED, BLUE, GREEN combinations to be defined (foreground and background) for each of the Lastic highlight codes (0-49). Colour selection is performed using "slider" bars for each of the colours.

**Highlight Code**
Select the required highlight code from the list box. The current highlighting values are displayed in code sequence (INI parameters), visual example and slider position forms.

**Beep**
Check this box if a beep is required when this highlight code is used

**Current Control Highlight Code**
If this highlight code is to be used for the current control (to enable easier identification) check this box.

**Foreground**
Move the sliders as required to increase/decrease the levels of the foreground colours. The code sequence and visual example change as the sliders are moved.

**Background**
Move the sliders as required to increase/decrease the levels of the background colours. The code sequence and visual example change as the sliders are moved.

**Password Character**
Enter the character to be used for hidden input fields.

**Default Colours**
Check this box for Lastic GUI to utilise Windows default desktop colours for highlight codes 0, 4 and 5. This will cause Lastic GUI to automatically change colours when the desktop colours scheme changes

**Use Windows backgrould colour in editable grid cell**
If checked then the current Scheme/Theme value for clWindow wil be used. If unchecked then the gid backgound colour is used.

## 3.5 SUNDRY

This includes both controls sizing and function key utilisation.

### 3.5.1 CONTROL SIZING

**Cell Height**
Within LASTIC GUI each control is positioned as a multiple of the cell height. A cell is the equivalent of a "line" on a character terminal. Typically, a cell height of 20 pixels on a screen of 480 pixels vertical resolution will enable 24 "lines" to be displayed.

**Cell Width**
Within LASTIC GUI each control is positioned as a multiple of the cell width. A cell is the equivalent of a column on a character terminal. Typically a cell width of 8 pixels on a screen of 640 pixels horizontal resolution will enable 80 "columns" to be displayed.

**Control Height**
If controls were the same height as the cell then each control would merge into the one below. The control height parameter enables the height of a control to be defined - typically three to five pixels less than the cell height.

**Grid Cell Height**
Within a grid control, the individual cells may be structured according to taste. This parameter defines the number of pixels for the height of an individual row.

**Grid Cell Width**
This defines the number of pixels to hold a single character. Cells are defined in terms of default character widths. Thus a column of 10 characters would be given a physical width of 10 * this parameter.

**Centre Forms**
If checked then forms will be aligned horizontally in the centre of the screen when created. If not checked they will be positioned as designed.

### 3.5.2 FUNCTION KEYS

This form provides a grid showing the function key action and the key number to invoke this action.

**Help**
To invoke the help system.

**Find**
To invoke the look-up facility.

**Reset**
To reset the current input field's value.

**Refresh**
To refresh the current window's fields.

**Jump to**
To move to the "default" or jump-to control.

**Application**
To invoke the form level application call-back.

**Status**
To invoke the status form.

**Grid Edit**
To enter character edit mode within a grid (default is replace mode)

## 3.6 EMULATOR

This window defines parameters used within the emulator. They include font details, answerback message and function key generated escape sequences.

### 3.6.1 FONT (80 and 132 columns)

**Text "Select"**
This button may be "clicked" to invoke a dialog form to select/change the font for representing normal text (Graphics characters "off").

**Line Draw**
Select the font face from the combobox for use in line drawing (Graphics "on"). Note these two fonts should be of the same style to ensure correct spacing.

If you are using English text then you can set both to LasticGUI font.

### 3.6.2 ANSWERBACK MESSAGE

Enter a sequence of characters as an answerback message.
NB. Carriage Return is automatically appended thus it is not to be entered.

### 3.6.3 EMULATOR PARAMETERS

**Hide Communications Menu**
Check this box if the emulator form is not to show the communications menu thus disabling user access to the open connection facility and the monitor form.

**Respond to Mouse Up**
If checked then the Emulator will send an escape sequence back to the host system when a right button Mouse-Up event occurs in the emulator. The escape sequence will consist of:
<27> <91> line_number ";" column_number <72>
to indicate which line and column number the mouse "clicked" on.

**Use 2 Colour Palettes**
Checking this box will cause the operation or 2 levels of colour palettes to correspond with normal and high intensity as set by programmed escape sequences. The "Normal" palette being darker colours than the high intensity. This will operate for both foreground and background colours.

**Reverse Black**
If the 2 colour palettes are used then, by default, "normal" colours are lower intensity with black shown as dark grey. If you require black to appear under normal intensity and dark grey under high intensity then check this box. No other colours are affected by this setting

**Force Bold Foreground**
If using 2 colour palettes then checking this box will cause the high intensity (Bold) colours to be used for the foreground and low intensity (normal) colours for the background regardless of the setting of high intensity via escape sequences.

**Bold Colour**
If using 2 colour palettes then if a colour (other than "None") is selected from this combobox then the selected colour will always be used as the foreground colour when a high intensity (bold) "On" escape sequence is sent to the emulator. Background colours are not affected.

**Default Colours**
Press this button to invoke a form to set default colours for background and foreground.

### 3.6.4 FUNCTION KEY ESCAPE SEQUENCES

Enter or amend the character stream to be sent when the corresponding keys are pressed. The escape sequence consists of a series of ASCII values of characters separated by commas. No comma is to be entered at the end of the list.

## 3.7 SCRIPT FILE

It is possible to attach, through the set-up routine, a “script” file to a server’s parameters such that, upon opening the server connection, the script file is obeyed.
This script can be used for “waking up” the server, sending and receiving option negotiation data (“terminal capability”), logon details etc. and can also be used to bypass using the emulator window.
The script file must be held on a disk drive directly accessible to MS-Windows.
Typically, the file will be created and amended using a text editor such as notepad.

### 3.7.1 FILE STRUCTURE

All script lines are terminated with carriage return though the carriage return itself is not utilised.
The first line in the script file must be:

| | |
|---|---|
| Lastic GUI Options | This is used as an identifier to ensure that a suitable file is being presented to the GUI software. |

The final line of the script file must be:

| | |
|---|---|
| end | This identifies that all commands have been listed. |

A script file will appear in the form:

```
Lastic GUI Options
conn:.....
ask:.....
erm:.....
erm:.....
wake:.....
  “
recv:.....
send:.....
  “
disc:.....
end
```

Details of the individual commands are given below. Empty lines are permitted to separate sections. Comment lines may be inserted by entering the semi-colon as the first character of such a comment line.

### 3.7.2 COMMANDS

Individual commands are listed below and may be upper or lowercase and are always followed by a colon (:). Command arguments follow the colon. The commands are not case sensitive but certain sub-parameters are. These are defined individually below.

#### conn:

This is used to define the connection requirements and should be the first command. Optional sub-parameters, delimited by colons, are:

| | |
|---|---|
| auto | signifying automatic connection without using the emulator window. If the server “auto-open” setting is “off” then this parameter is ignored. (Not case sensitive). |
| wait=n | defining the number of seconds to pause between sending wake-up character strings. (Not case sensitive). |
| msg=x...x | defines a message to appear in a window while connection is made to the server. (Not case sensitive). |

Example formats are:

CONN:auto:wait=2:msg=connecting to the database - please wait
conn:MSG=Trying to connect:WAIT=3
conn:AUTO:wait=3

It does not matter what order the sub-parameters occur.
If the conn: parameter is omitted or individual parameters are not present, defaults will be:

| | |
|---|---|
| Non-auto | The system presents the emulator window though may concurrently run the script file if the set-up is set for auto connect. |
| Wait=1 | The system will wait for 1 second between transmissions of the wakeup characters. |
| msg=“” | The default message 'connecting to database' is displayed in the form. |

#### ask:

This command causes a logon window to be displayed to request “values” for the items listed in the command. It can be used for requesting such as username, password etc. The format of the argument to the command is a series of parameters, delimited by a colon (:) with each indicating the “prompt” name for the item and optionally indicating if the response is to be “hidden”(h/H) and if the entry is to be fixed at lower or upper case (l/u). The order of entry is the order displayed. Only a single ask command is permitted and up to 9 requests can be made. The ask command, where ever placed in the script, will be performed first before the communication 'port' is opened. If you want to perform requests during the operation of the script use the 'req:' command below.
For each parameter, the “text” is used for the prompt.

If the sub-parameter field contains ‘h’ or ‘H’ the input is hidden (password protected).
If the sub-parameter contains ‘l’ or ‘L’ the input is fixed at Lowercase.
If the sub-parameter contains ‘u’ or ‘U’ the input is fixed at Uppercase.

In order to use the entered values, the send command references entered values using the “?” option (see below).

Example: ask:Username,L:Password,hL

## erm:

Following the ask parameter, the “erm:” parameter can be listed as many times as required. Each entry will contain the text or partial text of an error message from the log in procedure. Partial texts should be unique and will, if encountered, by displayed as the received error thus they should be meaningful to the user. Do not just test for the word “invalid” since the user will not know what is invalid.
Messages from the database system will be checked against this list of errors and if the message contains such an error text then this will be reported in a message box followed by an abort of the log in procedure. The order of the message list is irrelevant.
Parameters to this command, delimited by ':' (colon), are:

| | |
|---|---|
| message from host | The text of the message to check (mandatory). |
| label name | The label to 'goto' if such a message is received (optional). If not present the logon procedure will terminate. |
| message to display | The message to display to the user instead of the message from host (optional). If a label name is also present this message will be displayed before proceeding to the specified label. If this is not present though a label is defined, no message is displayed. |

Examples: erm:invalid password
erm:unknown username::Your username is not recognised
erm:not available:namespace:This namespace is not available

## wake:

This parameter, which should follow the conn: command, defines the character(s) used to “wake up” the server. These typically are either carriage-return (ASCII 13) or ctrl-C (ASCII 3). For non-printable characters the ASCII values must be given. To identify ASCII values rather than characters, the first character after the colon must be # and character values must be separated by commas. For printable characters, these are entered directly and sent “as-is”.
The “wake” characters are sent to the server and the GUI listens for some response. If no response is received within the wait time period, the characters are sent again. This is repeated until a response is received.

| | | |
|---|---|---|
| Examples: | wake:#13 | send carriage return |
| | wake:#0 | send break |
| | WAKE:#13,3 | send carriage return, ctrl-C |

## label:

This command defines a marker point in the script though does not, itself, perform any action. It is used by the 'goto:' command and the label field of the 'erm:' command to cause a 'jump' to the command following this label.

Example:

| | | |
|---|---|---|
| | label:askname | mark this point in the script as 'askname' |

## goto:

This command causes an unconditional jump to the label defined in the command.

Example:

| | | |
|---|---|---|
| | goto:askname | jump to the point in the script marked as 'askname' |

## req:

Request for user input. The format of this command is the same as for the 'ask' command above. However, while the 'ask:' command, where ever placed, will be performed first (before the communication link is opened) the 'req:' command will be performed when encountered in the script. For format details, see the 'ask:' command above.

## send:

This command defines a series of characters to be sent to the server. As with the wake command, characters may be entered directly or ASCII values may be defined. Carriage return must always be defined in ASCII value terms. Send command parameters are not repeated as with the wake command. Send commands may follow each other to generate a stream of data.
If it is required to send a value entered via the 'ask:' or the latest 'req:' command then the format must be send:?n where n is the order number of the requested items starting at 1. You cannot reference user entries from other than the last request.

Do not start a character string with “?” unless the data to be sent is from the 'ask:' or latest 'req:' command entry.

| | | |
|---|---|---|
| Examples: | SEND:jones_a | send the characters ‘jones_a' |
| | send:#13 | send <carriage return> |
| | send:?2 | send the user entered data from the second question |
| | send:#13 | |

## recv:

The recv command defines a series of characters to be received from the server. In addition to defining printable and non-printable characters as for the send command, it is possible to ignore characters by using the “*” wildcard. The wildcard character must be entered on a separate line and enables a series of characters of unknown or variable quality to be “skipped” from checking.
A typical use is where the server issues a welcome text or notice board before asking for the user id. This would simply be coded as:

```
recv:*
recv:Username
```

Telling the system to ignore any incoming characters until the word “Username” (case sensitive) is received.

## disc:

Disconnection at the end of a session may be performed by the host server but may also require some form of disconnection by the client. An example of this is where a terminal server is used. In this situation the wake-up will be targeted at the terminal server rather than the host database server and the script will issue a connect command for the terminal server. It will then proceed to log on to the database server and operate the application. Upon completion, the application will terminate and the connection to the database server will be broken. However, the client will still have a session with the terminal server which also needs to be broken. This can be performed using the disc: command.
The format options are the same as the send: command above and can consist of multiple “lines” of printable and non-printable characters. However, the client will not check for any response to the characters sent and will discard any received.
The disconnect command only operates where the connect command was “auto”. Where connection is made through the emulator window, the disconnect command is ignored.

| | |
|---|---|
| Examples: | disc:logoff |
| | disc:#13 |

### 3.7.3 ABORTING THE CONNECTION

If the “msg=“ option is used within the conn: command, a window will be displayed containing that message and also a cancel button. If no response is received after a suitable time the user may press the cancel button to abort the connection attempt. If the script file contains error messages then, if such a message is received it will be displayed in a message box with an "ok" button. Pressing this button will terminate the task.

### 3.7.4 LINKING THE SCRIPT FILE TO THE SERVER PARAMETERS

Invoke the set-up routine and select the server option from the communication menu.
Select the appropriate server from the list and press the edit button. This displays the current parameters for the selected server. If a script is to be actioned enter the full path and filename of the script file and then press the OK button remembering to save the file upon exit from the set-up routine.
The same script file can be linked to a number of server parameters.

### 3.7.5 OPTIONS NEGOTIATIONS

For networks and certain terminal systems, the client and server have an initial dialogue to ascertain the nature of each other.
Lastic GUI can perform such negotiations automatically if you check the auto negotiations checkbox for that server. Alternatively, you can handle such negotiations within the script file if you are aware of the options required. These “negotiations” vary between networks and systems but the information should be available from your supplier. Using the send: and recv: commands it is possible to trap incoming requests and to return the required information to the server.

### 3.7.6 Example Script File

```
Lastic GUI script                                   Header
conn:auto:wait=0:msg=Connecting...                  Line 1
erm:User authorization failure                      Line 2
erm:namespaceerror:nspace:No Such Namespace         Line 3
ask:Username,:Password,h                            Line 4
recv:*                                              Line 5
recv:Username:                                      Line 6
send:?1                                             Line 7
send:#13                                            Line 8
recv:*                                              Line 9
recv:Password:                                      Line 10
send:?2                                             Line 11
send:#13                                            Line 12
recv:*                                              Line 13
recv:>                                              Line 14
label:nspace                                        Line 15
req:Namespace                                       Line 16
send:w $$new^changenamespace(                       Line 17
send:"                                              Line 18
send:?1                                             Line 19
send:")                                             Line 20
send:#13                                            Line 21
recv:*                                              Line 22
recv:ok                                             Line 23
recv:*                                              Line 24
recv:>                                              Line 25
send:d ^rmenu                                       Line 26
send:#13                                            Line 27
end                                                 Line 28
```

The script file above performs the following:

| | |
|---|---|
| Line 1 | Identifies an Auto connect with a message of “Connecting…” |
| Line 2 | Identifies a message resulting from a logon failure and will cause termination of the logon |
| Line 3 | identifies an error message “namespaceerror” resulting from a function call to change the user’s namespace. In this instance, it will return to the position of label “nspace” in the script after displaying a messagebox stating “No Such Namespace”. |
| Line 4 | will cause a dialog form to request username and password (with passwird hidden) |
| Line 5 | will permit a stream of unknown text to be received until:- |
| Line 6 | “Username:” is received |
| Line 7 | Sends the user’s response to the Username input (ask: Question 1) |
| Line 8 | Sends a <CR> character |
| Line 9 | permits a stream of unknown text to be received until:- |
| Line 10 | “Password:” is received |
| Line 11 | Sends the user’s response to the Password input (ask: Question 2) |
| Line 12 | Sends a <CR> character |
| Line 13 | will permit a stream of unknown text to be received until:- |
| Line 14 | a “>” character is received |
| Line 15 | Identifies a label “nspace” which matches error message “namespaceerror” |
| Line 16 | will trigger a dialog box to request the namespace |
| Lines 17-21 | will call a function “new^changenamespace” with the entered namespace as a parameter.<br>This function will attempt to change the namespace and will return “ok” if successful otherwise “namespaceerror” if not successful. |
| Line 22 | will permit a stream of unknown text to be received until:- |
| Line 23 | “ok” is received. If “ok” is not received then it checks for an “ask:” error – e.g. “namepaceerror”. If “namspaceerror” is returned then this will be trapped by the error identifed in Line 3 and will thus generate a message of “No Such Namespace” and will then reset to line 15 (and ask again). |
| Line 24 | will permit a stream of unknown text to be received until:- |
| Line 25 | a “>” character is received |
| Line 26 | Sends command to start routine ^rmenu appended with |
| Line 27 | A <CR> character |
| Line 28 | End of script file. |

# 4. M SYSTEM SET UP

This chapter describes the set up requirements to be performed on the host M system.
The M driver routines are supplied within the Lastic distribution file lzsdist.glo and should be installed using the normal Lastic software installation procedure.

## 4.1 TERMINAL MODE

The Lastic Screen Handler identifies Lastic GUI when the terminal type (%lztt) has a terminal mode of 30-39. A terminal mode of 30 is provided with the software. A terminal mode of 37 indicates a Lastic GUI Server type.

## 4.2 TERMINAL TYPE

You can create a number of terminal types associated with Lastic GUI though typically the supplied “lw” type will be sufficient. Using the Set up facility (available from the menu when ^lgmenu is run from the emulator) and select "GUI Client". A default type of "lw" is provided with the software distribution.

### 4.2.1 Default Message

A suitable message should be created using the message maintenance facility (msgm in lzmenu) describing the function keys available which can be linked into the terminal type.

### 4.2.2 Sync Start Char

This is the ASCII character value for start of message and must correspond to that set on the PC set-up as described in section 3.2.1.1 above. Recommend using ASCII 28

### 4.2.4 Sync End Char

The ASCII character value for the end of message and must correspond to that set on the PC set-up as described in section 3.2.1.1 above. Recommend using ASCII 29

### 4.2.5 Message Sep Char

The ASCII character value used as a replacement for Carriage Return/Line Feed when transferring documents between the Host and the PC. It must correspond to that set on the PC set up as described in section 3.2.1.1 above. Recommend using ASCII 30.

## 4.3 Answerback

If you are operating a mixed terminal type environment it is necessary to determine what type of device is in operation so that the correct terminal type can be assigned. It is suggested that the easiest means of accomplishing this is to use the answerback facility. Character terminals also provide this facility. You could either set the answerback to the terminal type code directly or use a cross reference table to associate the answerback received to the terminal type required. The request for the answerback is made by sending ASCII 5 (ENQ) and then performing a read. When the terminal type is identified this should be set into the system variable %lztt before invoking any Lastic Screen Handler activity.
The ENQ facility is only available when operating though the emulator.

## 4.4 LASTIC GUI SERVER

You can also use Lastic GUI in server mode (in windows mode only - i.e. not providing emulator services). This mechanism enables automated background processes to run your applications. You will need to implement your own logon facility since such connections will automatically connect between your PC and the Mumps server. The Cache function $system.Security.login provides such a capability which would typically be embedded within the start application (see 4.4.1 below).
To operate this in a multi-namespace environment the routine ^lgserver (shipped as ^lzsr3srv) should be stored and operated within it's own directory/namespace which is mapped (for routine ^lgserver and global ^lgservctr) from the operational namespaces.

Server processes may be run concurrently with Telnet started processes and typically we would recommend IT aplications to be run via Telnet thus enabling them to operate while the server is shut down.

### 4.4.1 SETUP

To set this up, having installed your M software, transfer the routine lgserver to the namespace from which it is to operate.

Run **setup^lgserver**. The program will ask the following, displaying current values if you are amending the setup.

**Start Directory** — The name of the UCI/Directory in which the following application is to commence.

**Start Application** — The entrypoint of the routine which is to be started up - typically a logon application.

**Master TCP Port** — The initial connecting port as set in the Lastic GUI setup. Must be greater than 2000. Note, each process activated will increment this port number so use a base number which has sufficient range above it to accommodate all possible user logons. We would recommend port numbers of 50,000 or above.

**Max clients** — The maximum number of concurrent server processes to be permitted (= max number of logged on users).

**Terminal type** — A Lastic GUI terminal type set up with mode 37.

**Cache buffer size** — An optional override of the default TCP buffer size (in bytes). (applied to both input and output)

You can amend the server parameters as required though you must shut down the server first (see below).

### 4.4.2 STARTING THE SERVER

You can start the server manually by typing **d start^lgserver** from any mapped namespace or you can make it start automatically be adding **"d startup^lgserver"** from your customised auto startup facility. Both entrypoints will start the server as a background job.
For automated startup the script must locate to a mapped namespace for ^lgserver (see shutdown below).

**NB please note the entrypoints are different for starting the server manually or automatically.**

If the server was stopped manually without terminating current jobs then the server parameters may be amended and then the server manually restarted. This may be used, for example, to dynamically extend the range of available ports though the number of ports will not be dynamically reduced – it requires a full shutdown since active jobs may be using ports which would no longer be within the revised range.

### 4.4.3 SHUTTING DOWN THE SERVER

The server mechanism can be shut down manually by typing **d stop^lgserver** from within a mapped namespace.
The server routine will respond with the following:

| | |
|---|---|
| **Stop server logons <Y>** | Enter "Y" or <CR> only to stop further logons using the server (raw TCP/IP) facility. This does not disable TELNET logons which can be disabled using the Cache manager facilities.<br><br>If "N" is entered the facility terminates without any action. |
| **Stop current jobs <N>** | Entering "N" or <CR> only will leave current processes running thus merely disabling further connections.<br><br>Enter "Y" to cause current Cache jobs started by the server mechanism to terminate when inactive for 30 seconds.<br>The Cache processes will be terminated and the GUI users will, upon their next attempted communication activity, receive a message box informing them that the network has disconnected. Clicking the OK button will then terminate the PC process on their PC. |

To shut down via your automated closedown process add the following to your Cache command stream: -
d **shutdown^lgserver**
You may also need to swap to a mapped namespace for this command and then return to your original namespace:-

```
s x=zu(5)
zu(5,target_namespace)
d shutdown^lgserver
zu(5,x)
```

Note, the shutdown method will always cause current jobs to be terminated.

### 4.4.4 Restoring a job's state

Should a job terminate without it closing down correctly (for example your code performs a Hang after an error rather than letting the process continue down through it's stack (where it will be closed down cleanly be the server process) it may be necessary to re-set that port and job within the server details. To perform this, type:

d resjob^lgserver
Reset server job number > [enter job number to be re-set ]

If such a job is defined in the server then it will be "cleaned" up such that it's current details will be removed and the port returned to the available pool.

# 5. DYNAMIC DATA EXCHANGE

DDE conversations must be performed while within a call back ( this includes function calls) - the DDE control is loaded and assigned to the current window. If a window is terminated while assigned DDE sessions are in operation those sessions are terminated without informing the application. Upto 9 DDE conversations may be in concurrent operation. The conversations allow both send and receive modes - currently for "text" type items only (i.e. not pictures).

## 5.1 API

A DDE conversation may be started, data sent or received, commands executed and the session terminated via a single call if required. Alternatively, it may be repeated through a number of calls. No special controls are required - they are automatically created and remain invisible to the user.

A DDE call is made using the format: s x=$$dde^lzsr(id,action,.inf)

Note the .inf denoting a "by reference" parameter.
The value returned in x is either "0" denoting all ok or a numeric code indicating the error - see later.

id = conversation number range 1-9
action = activity(ies) required in this call
- 1 = initiate DDE conversation
- 2 = send/get information
- 4 = execute command(s)
- 8 = terminate conversation

values may be added such that 15 will perform all in one call

.inf is an array in the following format

Node (1,... used for conversation initiation

inf( 1, 1) = 0 = send data, 1=receive data :Default mode
1, 2) = Application name (up to 8 chars - e.g. EXCEL)
1, 3) = Link time out (seconds) - max. 60
1, 4) = Auto start application mode(0=No, >0=mode)
mode 1= Normal window, activated
2= Normal window, not activated
3= Minimised, not activated
4= Minimised, activated
5= Maximised & activated
1, 5) = Start up program name (i.e. winword.exe)
Can optionally include path and start up parameters
(NB if only program name specified the application must be in a path for DOS/Windows)
1, 6) = Topic (i.e. name of document)
1, 7) = 1 to send large data as string list. Optional setting for communicating with Delphi DDE server applications for data values >256 bytes long.
May be omitted if data item<256 bytes or Server process is not written in Delphi.

NB the system puts items 2 and 6 together as the link topic.

| | | | | |
|---|---|---|---|---|
| | Node (2,...used for data send/receive | | | |
| inf( | 2) | | = | 0:send, 1:receive to override default (else null) |
| inf( | 2, | x) | = | value to send or value returned<br>NB x = item "name"<br>Where the order of items is not important<br>a "data-set" can be sent or received<br>within a single call. |
| inf( | 2, | x, n) | = | additional data to be sent or received where n=1..n.<br>A request for data will determine it's size and will automatically generate lower nodes as required. It should be considered as a single stream. Carriage returns are substituted in this data with ASCII 127. Similarly, when sending data with carriage returns ASCII 127 should be used instead in the M strings. |
| | Node(3,...used for link execute commands | | | |
| inf( | 3, | n) | = | command string<br>where n is sequence number thus a series of commands can be invoked within a single call. |

## 5.2 ERROR CODES

The function returns a value which denotes the status of the performed function

| | | |
|---|---|---|
| 0 | = | OK |
| 1 | = | Auto start - application is not running and cannot be started. |
| 2 | = | Non-auto start -application is not running. |
| 3 | = | Conversation id is already running when trying to start it. |
| 4 | = | Conversation not operating (sending actions 2, 4, 8 without 1). |
| 5 | = | Not enough memory for DDE. |
| 6 | = | Data mismatch between source and destination. |
| 7 | = | Link execute failure (i.e. command not recognised/permitted). |
| 8 | = | DDE update failure. |
| 9 | = | Unknown record type (possible data corruption). |
| 10 | = | Using DDE when not in Lastic GUI mode. |
| 11 | = | Application is not currently in an event. |
| 12 | = | Conversation id not valid. |
| 13 | = | Action mode not valid. |

## 5.3 WINDOW CAPTION

Certain PC operations require the window caption (title) to be used to reference the destination application by the source application. Since the Lastic GUI caption can be made up of text and output items the exact representation of this item may be difficult to determine. To overcome this, a function is provided to return the caption or title of a window.

```
s x=$$lzwcap^lzsr(system_code,window_code)
```

will return the exact value of the nominated window caption. If the designated window does not exist or the function is performed while not running Lastic GUI then the returned value will be an empty string.

## 5.4 COMMAND LINE EXECUTION

To invoke another application call gexec^lzsr.

s result=$$gexec^lzsr(commandline, mode)

Where commandline contains application and parameters – typically as would be entered in the Windows “run” form, mode is as per DDE, namely one of:

1= Normal window, activated
2= Normal window, not activated
3= Minimised, not activated
4= Minimised, activated
5= Maximised & activated

result = 1 (application has started) or 0 (application failed to start).

# 6. ADDITIONAL FACILITIES

Lastic GUI is provided with an additional “exe” routine lgraph.exe to provide graphics. It is communicated with using DDE though this is hidden from the application by specific function calls.

## 6.1 Graph Facilities

Lastic GUI provides facilities for generating multiple graphs of differing styles directly from your M system while within an event of any form.
The system utilises DDE conversation numbers 8 and 9.
The GUI software is issued with additional routines to support this facility (listed earlier).

### 6.1.1 M Function

The graphics facility is invoked, from within any event, using the M function call $$graph^lzsr.

The format of the function call is: s x=$$graph^lzsr(.paras) where the function returns values of:

| | | |
|---|---|---|
| 0 | = | action successful |
| -1 | = | parameter error |
| 1-n | = | DDE errors |

The parameters, passed in an array by reference, consist of the following:

| **Paras Reference** | **Content (see action below)** |
|---|---|
| paras(“number”) | a graph number in the range 1-99 |
| paras(“type”) | 1=pie chart, 2=histogram (vertical), 3=line, 4=histogram (horizontal) |
| paras(“style”) | 0=normal (2D), 1=3D |
| Paras(“pie”) | Applicable to Pie charts only.<br>Fields: Initial legends (0/1)~explode slices %~donut hole %~rotation angle~multi-line labels (0/1)~slice frame [frame visible (0/1), colour, width (pixels) ]~Marks [marks visible(0/1),colour, length(pixels)] ~pie3D[horizontal offset(0-100%),vertical offset(0-100%),elevation (0-90deg,zoom(-100 to 500)].<br>By default all settings are null/zero. |
| paras(“linewidth”) | Width (in pixels) of 2D line joining plot points |
| paras(“percent3D”) | Optional to change the percentage of 3D effect (e.g. 3 = 3% shadow) |
| paras("name") | Window caption text |
| Paras(“title”) | Graph title text |
| Paras(“tittext”) | Font for Graph title– name~size~style~colour. If item blank uses the default setting for that item. Style is a combination of B,I and U as required. |
| paras("sets") | the number of plot sets (series) - max. 10<br>(only the first is applicable for pie charts) |
| paras("setlegends") | Legends for each set delimited by ~ character (max 10) |
| Paras(“legfont) | Font for legends – name~size~style~colour. If item blank uses the default setting for that item. Style is a combination of B,I and U as required. |
| Paras(“setcolours”) | For graph types 2-4, RGB values of sets delimited by commas. For type 1 RGB values of individual segments of pie chart. Optional. If not set colours will be auto generated. |
| Paras(“backcolour”) | The RGB value for the background colour of the graph. |
| Paras(“axtitles”) | titles for each axis (Left, Right, Top, Bottom) delimited by ~ |
| Paras(“axfont”) | Font for axes titles – name~size~style~colour. If item blank uses the default setting for that item. Style is a combination of B,I and U as required. |
| paras(“xlabangle”) | To angle the X axis labels in a line graph. 0 = horizontal, 90/270 for vertical, 315 for 45 degree from vertical. If not set defaults to zero. |
| Paras(“labfont”) | Font for labels – name~size~style~colour. If item blank uses the default setting for that item. Style is a combination of B,I and U as required. |
| paras(“lalign”) | Set to T, L, R or B to define location of legend table (def = R) |

| Paras Reference | Content (see action below) |
|---|---|
| paras(“vmargin”) | Set to define the %age of the value Axis range to extend past the extent of the plot points. |
| paras(“cmargin”) | To define the chart margins as a %age of the client area of the form. Default is 2%. If used, set to values for Top,Left,Right,Bottom in the form: 2,3,4,5 (comma delimited) all values must be provided if parameter is set. |
| paras("setminy") | set 1 if Y axis origin is to be the lowest plot value else null/0 |
| paras("points") | the number of points plotted per set |
| Paras("pagepoints") | The max number of points per page. Setting this will enable scrolling via scrollbars on the right or bottom of the form as appropriate for that graph. |
| paras("text") | Plot point labels: 0= none, 1= Value, 2= percent, 3= label, 4= Label+percentage, 5= label+value, 6= Legend, 7=Xvalue |
| paras("values",set,point) | plot value~label/legend text |
| paras("x") | Form margin from left of screen (in terms of character screen columns) |
| paras("y") | Form margin from top of screen (in terms of character screen lines) |
| paras("w") | Form width in terms of character screen columns |
| paras("h") | Form height in terms of character screen lines |
| paras(“file”) | File name & path for savetofile command |
| paras(“scrollpoints”) | Set to a number to ‘page’ that number of points using the ‘scroll’ command |

Notes
Colours are passed as integer values calculated from RGB values (i.e.colour= R*256+G*256+B)

action options and parameters are:

| **Action** | **use** | **resultant state** | **parameters** |
|---|---|---|---|
| open | open graphics window | window open but invisible | number |
| draw | pass graph parameters | parameters loaded | all* |
| show | make window visible | visible | number |
| hide | make window invisible | invisible | number |
| close | close graph window | window removed | number |
| end | terminate graphics | all windows removed | none |
| filehide | Hide file menu | File menu invisible | number |
| fileshow | Show file menu | File menu is visible | number |
| copymeta | Copy to Clipboard | Metafile on clipboard | number |
| copybmap | Copy to Clipboard | Bitmap on clipboard | number |
| clearcb | to clear clipboard | clipboard cleared | number |
| maximize | Maximise the window | Maximised window | number |
| savetofile | Save as file | Copy of graph as file | number, file |
| scroll | scroll a graph | graph scrolls plot ‘points’ | number, scrollpoints |

Note, the parameters column indicates which of the parameters (paras) are used for that command.

For example, to save an open and drawn graph to file set paras='savetofile', paras("file") = filename and paras("number")=graph number.

When saving the graph as a file through the savetofile command, the extension determines the type of resultant file. The extensions (upper or lower case) possible are :BMP, WMF, EMF

* Unless you wish to force a placement of this graph the parameters x,y,w and h should not be passed to a window after the first time it is drawn otherwise the graph window will be re-positioned after a user has moved it.
Parameters not used by an action are ignored thus it is not necessary to "clear" the array before issuing an action function call.
It is necessary to show the window the first time but not again unless you have hidden it. Issuing a draw command to a visible window will re-draw that graph.
It is not necessary to close each window before terminating the action. The end command will automatically close all graph windows.

### 6.1.2 Operation

The M application must issue an open command before a draw, show, hide or close command for a window number.
Once a window has been opened and "shown" repeated draw commands will re-draw the graph with the designated type, style and values without requiring a hide or show command.
The show and hide commands enable graphs to be removed from the screen and then re-displayed without having to close and re-open the windows.
The release includes a distribution for routine "graphdem" which generates the types of graphs with random values.

# 7 PC FACILITIES

This chapter describes the functions for:

- Printing through the PC (Windows) API
- Transferring data from the Mumps database as PC files
- Transferring PC files into the Mumps Database
- Locating Files and directories
- Copy PC files
- Moving, PC files
- Renaming PC files
- Deleting PC files

## 7.1 PRINTING TO PC PRINTERS

Lastic GUI provides a mechanism for printing directly to printers attached to the PC or the PC's network. A singular function is provided and is called in the form:

```
s x=$$gprt^lzsr(mode,.params)
```

where mode = 1 (open), 2 (data), 3 (close), 4(abort).
and params consists of an array, passed by reference and containing:

| | | |
|---|---|---|
| params("collate") | = | "Y" or "N" (default status in printer dialog) |
| params("data") | = | for single line output instead of using "varref" |
| params("fontsel") | = | "Y" or "N" (Y enables font selection dialog box) |
| params("fontsize") | = | NN (font size -max. 99 if using default font) |
| params("orient") | = | "L" or "P" (landscape/portrait) |
| params("pmode") | = | "T"=Text (default), "R"=Raw (for PCL reports) |
| params("type") | = | T (text - default), R=Raw or B (Binary) |
| params("pages") | = | "Y" or "N" ("Y" indicates page range selection available) |
| params("pages","min") | = | min page number default |
| params("pages","max") | = | max. page number default |
| params("selectonly") | = | Set to 'Y' to select a printer without opening it. |
| params("strend") | = | optional line terminator characters - e.g. $c(13,10) |
| params("title") | = | title for print manager list |
| params("varref") | = | variable reference for data (can be global and subscripted) |

If the "varref" parameter is defined this will be used as the source of the output. Otherwise, a single line of the contents of the "data" parameter will be used. The "data" option is more likely to be used for sending print lines where data is read from a file or global one line at a time and sent.
The "pmode" parameter (if set to "R") enables raw printer characters to be sent such as text with embedded PCL code. If not set, the default 'Text' mode is used.

The "type" parameter enables binary data, such as bitmaps to be encoded in HEX and transferred to the print device. Note, to utilise binary transfer the pmode parameter should be set to "R" to allow the transfer directly to the printer. If sending Raw data, the type parameter should also be set to either 'R' or 'B'. If the "type" parameter is not set then the "Text" default is assumed.

If type = 'T' then ASCII 127 will be converted to CR/LF
If type = 'R' then no conversions will be performed
If type = 'B' the data will be converted from Hex to Binary.

For Non-Text data the "strend" parameter is ignored.

When called the function return is a response code and is one of:
-1 = parameter error or not using GUI, 0 = ok, 1 = Not open, 2 = already open (mode 1 only)

Mode 1, if a printer is opened or selected, will update the array in respect of:

| | | |
|---|---|---|
| params("collate") | = | "Y" or "N" |
| params("copies") | = | nnn |
| params("device") | = | Windows printer device name |
| params("fontsize") | = | nnn |
| params("orient") | = | "P" or "L" |

| | | |
|---|---|---|
| params("pages") | = | "A"(all) or "R"(range) |
| params("pages","min"/"max") | = | selected range if "R" |
| params("pageheight") | = | The number of lines of text available on the physical page. (does not allow for margins). |

To open the printer, create the array and: s x=$$gprt^lzsr(1,.params)
To print data: s x=$$gprt^lzsr(2,.params)
To close and assign to the printer: s x=$$gprt^lzsr(3,.params)
To abort the print job: s x=$$gprt^lzsr(4,.params)

The parameter params("varref") enables as much data as required to be printed from a single call. If, say, the reference is set to "^PRINT($j)" then the data mode will transfer all lower level nodes of this global to the printer within a single function call. This enables multi lines or multi pages to be processed from a single call.
Line throws are created either by appending CR/LF or ASCII 127 or setting params("strend")=$c(13,10) to cause every node of "varref" to be terminated with CR+LF automatically. For page throws use ASCII 12.

A typical piece of M code for printing, from within a form event such as a button click, would be:

```
       s params(.....                              ;create params array
       s x=$$gprt^lzsr(1,.params)                  ;open printer
       if x>0 q                                    ;terminate event
       s lcount=params("pageheight")\10*9          ; line count adjusted for vert margins (say 10 %)
lab    d build                                     ;create page of data
       d form^lzsr("system","printmon","000000")       ;show "printing in progress" form
       i %lzsex=1 s x=$$gprt^lzsr(4,.params) q ;aborted
       s x=$$gprt^lzsr(2,.params)
       g lab:more                                  ;another page?
       s x=$$gprt^lzsr(3,.params)                  ;close print job
       q                                           return to form
```

Note, to provide a “printing in progress” form as exampled, you should set the form timeout as zero seconds and provide a cancel button on the form which, if clicked, sets %lzskip=”-“. Each page will then be constructed and, before printing, the form “activated” meaning that it will detect the clicking of a “cancel” button.

## 7.2 TRANSFERRING DATA TO PC FILES

Lastic GUI provides a mechanism for transferring global data to PC files.
Files can be opened, data transferred, the file closed or the transfer aborted, from a single M function call in the form:

s x=$$gsfile^lzsr(mode,.params)

where mode = 1 (open), 2 (data), 3 (close), 4(abort).
and params consists of an array, passed by reference and containing:

| | | |
|---|---|---|
| params("data") | = | for single line output instead of using "varref" |
| params("dialog") | = | "Y" or "N" (Y enables the user to select the file) |
| params("overwrite") | = | "Y" or "N" (Y enables existing file to be replaced) |
| params("type") | = | "T"/null=text file, ”R”=Raw text, "B"=Binary file |
| params("title") | = | title for dialog box |
| params("file") | = | file name and path |
| params("filter") | = | dialog box file extensions filter |
| params("strend") | = | optional line terminator characters - e.g. $c(13,10) |
| params("varref") | = | variable reference for data (can be global and subscripted) |

If the "varref" parameter is defined this will be used as the source of the output. Otherwise, a single line of the contents of the "data" parameter will be used. The "data" option is more likely to be used for sending files where data is read from the local file one line at a time and sent.
Text files type ‘T’ will also convert any ASCII 127 characters to CR/LF whereas ‘Raw’ text files will not change any sent text.
For binary files, (type="B") the strings must already be in Hex format and the "strend" parameter is ignored.

When called the function return is a response code and is one of:

-1 = parameter error or not using GUI
0 = open ok
1 = Not opened
2 = already open
3 = no such directory
4 = file exists

Mode 1, if a file is opened, will update the array in respect of:

params("file") = the file name and path actually opened

| | |
|---|---|
| To open a file, create the array and: | s x=$$gsfile^lzsr(1,.params) |
| To send data: | s x=$$gsfile^lzsr(2,.params) |
| To close the file: | s x=$$gsfile^lzsr(3,.params) |
| To abort the transfer & close the file: | s x=$$gsfile^lzsr(4,.params) |

The parameter params("varref") enables as much data as required to be transferred from a single call. If, say, the reference is set to "^XFER($j)" then the data mode will transfer all lower level nodes of this global to the PC file within a single function call. To force CR+LF after each data node set params("strend")=$c(13,10).

**NOTES**

**Do not set the "strend" parameter for binary files.**
**Raw text files required CR/LF characters rather than ASCII 127 to generate a new paragraph.**

A typical piece of M code for data transfer, from within a form event such as a button click, would be:

```
       s params(.....                          ;create params array
       s x=$$gsfile^lzsr(1,.params)            ;open file
       if x>0 q                                ;terminate event
lab    d build                                 ;create set of data
       d form^lzsr("system","xfermon","000000")       ;show "file transfer in
progress" form
       i %lzsex=1 s x=$$gsfile^lzsr(4,.params) q      ;aborted
       s x=$$gsfile^lzsr(2,.params)
       g lab:more                        ;another set?
       s x=$$gsfile^lzsr(3,.params)            ;close file
       q                                 return to form
```

The filter parameter will be set in the form: "Text files (*.txt)|*.txt|Any file (*.*)|*.*"
containing pairs of fields, delimited by the character "|" .
For each pair of fields the first field = text to appear in combo box. while the second field is the actual search value to be used. The first filter specified will be the default.

## 7.3 FINDING A PC FILE OR DIRECTORY

A facility is provided within Lastic GUI for determining the status of a PC file. This could be used, transparent to the user, to determine is a file exists. It could also be used, with a user's interaction, to determine a file name and path for a later transfer.

The function is called in the form:

```
s x=$$gffile^lzsr(.params)
```

where the array params is a parameter passed by reference.

The params array consists of:

| | | |
|---|---|---|
| params("check") | = | "F" or "D" indicating check file or directory |
| params("dialog") | = | "Y" or "N" (Y=show dialog box) |
| params("file") | = | name and path of file.<br>If directory only then include final "\" |
| params("filter") | = | dialog box file extensions filter |
| params("title") | = | title of dialog box |

The return code will be one of:

-1 = Not GUI
0 = dialog box abort
1 = No such file - no such directory
(The file cannot exist because there is no such directory)
2 = No such file
(directory exists but file does not)
3 = directory exists (when only directory check performed)
4 = file exists

If the user selected another file then the params("file") will be updated
If the file exists then the array node params("attributes") will be created and will contain a four character string in the form: ABCD where

A:= "r" (readonly) or space
B:= "h" (hidden) or space
C:= "s" (system) or space
D:= "a" (archive) or space

## 7.4 TRANSFERRING DATA FROM PC FILES

Lastic GUI provides a mechanism for transferring PC files into Mumps data structures.
Files can be opened, data transferred and the file closed from a single M function call in the form:

```
s x=$$grfile^lzsr(mode,.params)
```

where mode = 1 (open), 2 (data), 3 (close).
and params consists of an array, passed by reference and containing:

| | | |
|---|---|---|
| params("dialog") | = | "Y" or "N" (Y enables the user to select the file) |
| params(“display”) | = | 0 = no display,<br>1 = display only,<br>2 = display + cancel facility. |
| params("type") | = | "T"/null=text file, "B"=Binary file |
| params("title") | = | title for dialog box |
| params("file") | = | file name and path |
| params("filter") | = | dialog box file extensions filter |
| params("varref") | = | variable reference for received data<br>(can be global and subscripted) |
| params(“maxstring”) | = | An optional parameter to define the maximum line length expected (until a carriage return is encountered). |

The maxstring parameter is only relevant if you are transferring in data records and you want to ensure that each record in held in a separate, sequential node.
If the parameter is not set then lines will be limited to 255 bytes and subsequent text will be stored on the next global node (i.e. it will be wrapped).

The display parameter, if set to 1 or 2, will cause a form to be shown detailing the status of the transfer. Setting the value to 2 will cause a cancel button to be included. If the parameter is omitted it will be defaulted to 0.

All files will be transferred as hexadecimal. If the file type is set to "T" or not defined (text file) then the inward stream is converted back to characters before storing into the "varref" with <CR> being removed and <LF> being converted to ASCII 127.

**NOTE Text strings may contain embedded control characters, tabs, ASCII 127 etc after conversion. It is the developer's responsibility to re-structure the data as required upon receipt. The wpr^lzslib routine may be used to separate text records.**

Received data will be stored into the "varref" structure at the next node with sequentially ordered subscripts.

When called the function return is a response code and is one of:

-1 = parameter error or not using GUI
0 = open ok
1 = Not opened
2 = already open
3 = no such directory
4 = file does not exist
5 = transfer has been aborted

Mode 1, if a file is opened, will update the array in respect of:
params("file") = the file name and path actually opened

To open a file, create the array and: `s x=$$grfile^lzsr(1,.params)`
To receive data: `s x=$$grfile^lzsr(2,.params)`
To close the file: `s x=$$grfile^lzsr(3,.params)`

A typical piece of M code for data transfer, from within a form event such as a button click, would be:

```
       s params(.....                          ;create params array
       s x=$$grfile^lzsr(1,.params)            ;open file
       if x>0 q                                ;terminate event
lab    s x=$$grfile^lzsr(2,.params)
       i x=5 k ^work(data)
       s x=$$grfile^lzsr(3,.params)            ;close file
       q                                       ;return to form
```

The filter parameter will be set in the form

"Text files (*.txt)|*.txt|Any file (*.*)|*.*"

containing pairs of fields, delimited by the character "|" .
For each pair of fields the first field = text to appear in combo box. while the second field is the actual search value to be used. The first filter specified will be the default.

## 7.5 OTHER FILE FACILITIES

In addition to creating files on the PC and reading a file from a PC into the database, various other facilities are provided. The function $$gmfile^lzsr provides the facilities for: Copying a file, Moving a file, Renaming a file and Deleting a file.

The function is called as follows:

**$$gmfile^lzsr(mode, from file, to file)**

where the file names must also include the path.

| Mode | Action | From file | To File |
|---|---|---|---|
| 1 | Copy | Original file name & path | New file name & path |
| 2 | Move | Initial file name & path | New file name & path |
| 3 | Rename | Original file name & path | New file name & path |
| 4 | Delete | File name & path | Ignored. |

**Response**

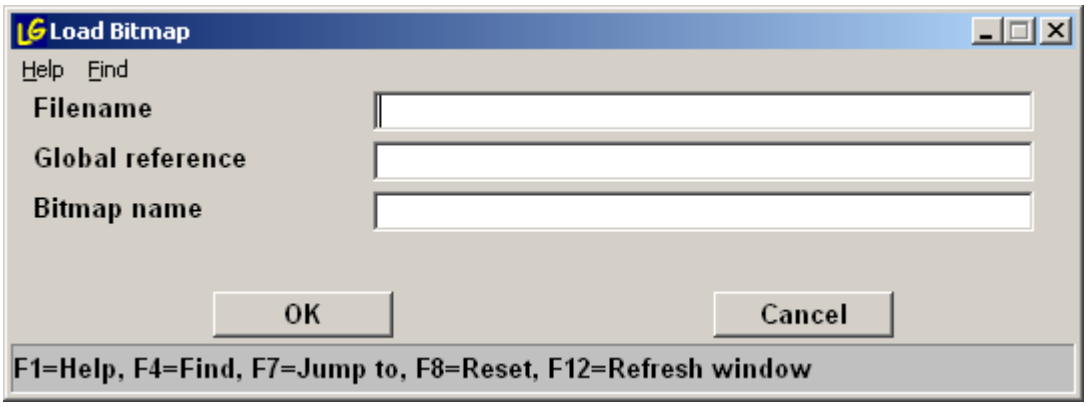
The function will return a response code being one of the following:

-1 = Not called from Lastic GUI
0 = OK
1 = No such from file
2 = To file already exists
3 = To file directory does not exist
4 = Action refused (e.g. sharing violation)

## 7.6 USING BITMAPS

### 7.6.1 Loading into the M database

This routine is available from ^lgmenu and has been provided to enable bitmap PC files to be loaded into the Mumps database. It uses the file transfer facilities described above but also generates "header" information for the bitmap for subsequent loading into a Lastic GUI control.
When run the following form is displayed.




**Filename**
Either enter the full path and filename or use the look-up facility to identify the required file.

**Global reference**
This should include the subscripting requirements and must be the full reference -e.g. ^BMP("A",2). It is possible to move the data to another global subsequently.

**Bitmap name**
Enter a description for this bitmap. For information purposes only.

Bit maps a transferred and held, within the M database, in Hex format to avoid any possible problems with global save and restore utilities. When loaded into Lastic GUI again the Hex format is transferred and changed back to Binary within Lastic GUI.

### 7.6.2 Loading Bitmaps into Lastic GUI

Lastic GUI provides a facility to load bitmaps into the PC routine once and to be used by any number of controls on different forms. Up to 1000 bitmaps can be held and referenced by index number (0-999).
To load a bitmap, call the function $$ldbmp^lzsr(index_no, varref, tp) where:

| | |
|---|---|
| index_no | is a value in the range 0-999 |
| varref | is a global reference to a bitmap loaded through ^lzsm025 (see above). |
| tp | optional param to indicate transparency of background.<br>If tp="A" it sets colour of bottom left pixel as transparency colour<br>If tp="H"nn It sets highlight code nn background colour as transparency colour. |

for example: s x=$$ldbmp^lzsr(123,"^bmp(43)", "A")

Using the same index number a second time will cause the new bitmap to replace the old one though individual controls using that bitmap will not be changed unless a re-display call is made for that control.
The function call returns a value of:

| | |
|---|---|
| -1 | Not running Lastic GUI |
| 0 | Loaded OK |
| 1 | Index out or range |
| 2 | Cannot locate bitmap specified. |

## 7.7 Obtaining IP Information

When connected via a TCP/IP socket, the host M system application can obtain connection information using the function **getip^lzsr**.

S x=$$getip^lzsr() will return, in the variable x, the following, separated by ASCII(127) characters.

- PC client IP address
- PC client host name
- PC client port number
- M server IP address
- M server host name
- M server port number

If the connection is not TCP/IP or there is no current connection then an empty string is returned.

## 7.8 Obtaining user name

To obtain the user name of the logged on PC user, call the function: $$getwinuser^lzsr().

## 7.9 Obtaining Environment Variables

To obtain an environment variable call the function: $$getenvvariable^lzsr(var_name).
The variable name (var_name) is not case sensitive. If it does not exist then the functon returns null.
The function returns the user level variable value if it overrides the system level.

## 7.10 Obtaining Locale information

To obtain information about the PC locale settings of the current logged on user, calling routine:
d getlocaleinfo^lzsr(.var) will return an array in variable ‘var’ as follows:

| | |
|---|---|
| (“LOCALE_ILANGUAGE”) | = language code (e.g. “0809” = English UK) |
| (“LOCALE_SLANGUAGE”) | = language name (e.g. “English”) |
| (“LOCALE_IDEFAULTCOUNTRY”) | = Country code (e.g. “44” = UK) |
| (“LOCALE_IDEFAULTCODEPAGE”) | = Code Page (e.g. 850) |
| (“LOCALE_IDEFAULTANSICODEPAGE”) | = ANSI Code Page (e.g. 1252) |
| (“LOCALE_SISO639LANGNAME”) | = ISO Language name (e.g. “en”) |
| (“LOCALE_SISO3166CTRYNAME”) | = IS) Country name (e.g. “GB”) |

7.11

# 8. PROGRAMMING FOR GUIs

In this section we will identify certain Dos and Don'ts applicable to the M programmer (and designer) when developing applications which could be operated both using character terminals and Lastic GUI.

## 8.1 FORM DESIGN

**DO**
Include the form "title" in line 0 of the form (text and/or output items). This will then appear in the title bar of the GUI form. Include at least one focussable type item on the form for the user to respond. The GUI will give focus to the first enabled control in the form unless overridden. If no focussable control is provided the form itself will receive focus and, unless a time-out is defined, will have to be terminated by the user.

Ensure that buttons which terminate a window set %lzskip="-" within their click event processing.

## 8.2 EVENT PROCESSING

**DO**
Provide an "output" item on the window into which messages can be sent (using the d disp^lzsr call) to inform the user of the processing taking place if the activity is likely to take more than, say 1 second - you can use the message variable (%lzser) for this with a type 2 message (non-hold).
Consider the use of disabled (skipped) controls where, within MS-Windows, the user can move between items. Either control this navigation using event validation and %lzfocus or set the appropriate skips and only enable controls when the user can viably access them.

**DO NOT**
Invoke forms from change or focus change events (gotfocus and lost focus). Forms should be invoked from command buttons or from menu entries.
Do not use d disp^lzsr to display items from within focus change events - set %^lzsdisp array instead. This is due to MS Windows stacking events (such as gotfocus and click events). performing a disp^lzsr will cause lastic GUI to yield in order to display those items which will also cause any click event to be fired while within another event. Lastic GUI will reject this click event though the user may not be aware of this.
Alter the value of a control while within it's own gotfocus event. MS-Windows will activate that control upon receiving focus - i.e. before the application is notified of that focus. It is therefore sensible to update such controls from within the event processing of those controls which cause such changes.
As a general rule do not invoke processes or other windows from within events of input fields. Such processing should normally be associated with a button or menu item.

## 8.3 APPLICATION TERMINATION

**DO**
When the last application has terminated (e.g. a main menu form) and a return is required to a character format or full termination, perform the "close^lzsr" call. This will close the last open form and will return to the emulator or terminate the GUI application..

**DO NOT**
Merely "quit" the environment. This will leave the GUI driver awaiting further messages from the host and will require the user to terminate the application forcibly.

## 8.4 MONITORING MESSAGES

During development it is possible to log messages from and to the PC on the host M database. To achieve this, set ^lzs("lg","mon",$J)=1. This indicates that all messages for your job will be logged. To stop logging of messages kill ^lzs("lg","mon",$J).

Messages are recorded into ^lwmlog for your job. This log is reset each time Lastic GUI is initialised for that job number (i.e. if the partition is "killed" before restarting).

At run-time, if a message is received from the PC which cannot be evaluated (say due to a network error), it is logged to this global even if monitoring is not turned on.

# 9. Reporting

Lastic GUI provides a reporting facility enabling applications to construct an XML file containing reporting requirements and then invoking a function to print that report.

## 9.1 Facilities and mechanisms

The report file includes:

- Report header information
- Page definitions
- Control definitions

Report header information includes:

- Device scaling (allowing for specific devices which assume a specific resolution to be negotiated)
- Orientation (Picture/Landscape)
- Copies
- Page margins (defining offsets for pages)

Page Definition information defines the individual controls appearing on that page.

Control information contains:

Text Control (a multi-line auto wrapping control)

- Position on page
- Optional frame thickness and colour
- Font details (name, size, style, colour)
- Alignment (Left, Right and Centre)
- Line spacing (variable as a fraction of the text height)

Table Control (multi-columns of single line text)

- Position on page
- Optional frame (thickness and colour)
- Optional Vertical column lines
- Optional Horizontal row lines
- Column width and alignment (left, right & centre)
- Cell data (text and optional colour - override of control level font colour)

Image Control (Bitmap, Metafile)

- Position on page
- Source filename
- Source file type

## 9.2 The DTD

The DTD used to control the XML file is shown below.

```
<!ELEMENT Report (Output, Page+)>
<!ELEMENT Output EMPTY>
<!ATTLIST Output
        DeviceScaling CDATA #IMPLIED
        Orientation (L|P) #REQUIRED
        Copies CDATA #REQUIRED
        PageMarginLeft CDATA #IMPLIED
        PageMarginTop CDATA #IMPLIED
        >
<!ELEMENT Page (Text*,Image*,Table*)>
<!ELEMENT Text (Font, TextLine)>
<!ATTLIST Text
        Position CDATA #REQUIRED
        FrameSize CDATA #REQUIRED
        FrameColour CDATA #IMPLIED
```

```
        BackColour CDATA #IMPLIED
        Alignment (L|R|C) #REQUIRED
        LineSpacing CDATA #IMPLIED
        TextMarginSide CDATA #IMPLIED
        TextMarginTop CDATA #IMPLIED
        >
<!ELEMENT TextLine (Font* ,#PCDATA)>
<!ELEMENT Image EMPTY>
<!ATTLIST Image
        Position CDATA #REQUIRED
        FileName CDATA #REQUIRED
        FileType (BMP|WMF|EMF) #REQUIRED
        >
<!ELEMENT Table (Font, Column+, Row+)>
<!ATTLIST Table
        Position CDATA #REQUIRED
        FrameSize CDATA #REQUIRED
        FrameColour CDATA #IMPLIED
        BackColour CDATA #IMPLIED
        ColumnLineSize CDATA #IMPLIED
        RowLineSize CDATA #IMPLIED
        LineSpacing CDATA #IMPLIED
        TextMarginSide CDATA #IMPLIED
        TextMarginTop CDATA #IMPLIED
        >
<!ELEMENT Column EMPTY>
<!ATTLIST Column
        Width CDATA #REQUIRED
        Alignment (L|R|C) #REQUIRED
        >
<!ELEMENT Row (Cell+)>
<!ELEMENT Cell EMPTY>
<!ATTLIST Cell
        Colour CDATA #IMPLIED
        BackColour CDATA #IMPLIED
        FontStyle CDATA #IMPLIED
        Text CDATA #REQUIRED
        >
<!ELEMENT Font EMPTY>
<!ATTLIST Font
        FontName CDATA #REQUIRED
        FontSize CDATA #REQUIRED
        FontStyle CDATA #IMPLIED
        FontColour CDATA #IMPLIED
        >
```

## 9.3 Attribute Definitions

| Attribute name | Description, format |
|---|---|
| Alignment | Text alignment L(eft), R(ight) or C(entre) |
| Colour | A colour name e.g. Black, Red etc. as understood by the PC,<br>A Hex value – e.g. #B2FF56 or<br>An RGB value as red,green,blue (e.g. 56,255,173) |
| ColumnLineSize | The thickness of the line in 10ths of a millimetre. |
| Copies | Number of report copies – becomes the default in the printer dialog. |
| BackColour | Optional background colour overriding the default White. See Colour above |
| DeviceScaling | A string containing printer device names and decimal values to increase or decrease the position values for specific output devices. E.g. “printerA,9.2354*printerB,0.625”.<br>The report facility automatically scales the report based upon the LogPixelsX value for the print device (via GetDeviceCaps call) based around a default of 600 pixels per inch. Thus this printer need only be added to the DeviceScaling parameter if this is to be overridden. |
| FileName | File path and name of graphic image |
| FileType | BMP (Bitmap), WMF (Windows MetaFile) or EMF (Enhanced MetaFile) |
| FontName | Name of font in font selection dialogs |
| FontSize | Size of font as per dialogs (numeric) |
| FontStyle | Combination of B(old), I(talic) and U(nderline) |
| FontColour | See Colour above. Indicates “foreground” colour |
| FrameColour | See Colour above. Indicates colopur of lines in frame |
| FrameSize | The thickness of the line in 10ths of a millimetre. |
| LineSpacing | A decimal fraction of the text height. For example, 1.25 = 1 and ¼ text line spacing |
| Orientation | P(ortrait) or L(andscape) |
| Position | The control position, in millimetres, in the form:<br>Left, Top, Right, Bottom (e.g. 120.5,45,200.125,90) |
| PageMarginLeft | Left margin of page in millimetres |
| PageMarginTop | Top margin of page in millimetres |
| Text | The Table cell text. |
| TextMarginSide | The margin, in millimetres, between the frame edge and the text. |
| TextMarginTop | The margin, in millimetres, between the frame edge and the text. |
| Width | The table column width in millimetres |

**Spelling must be correct though the attribute name is not case sensitive.**

## 9.4 Notes

In order to simplify the operations and handling of the XML file the following conventions are applied.

- Element and attribute names may be upper or lowercase (or a mixture).
- Tab characters are not allowed. All tabs will be stripped from the lines when read in.
- Attribute values may contain the various special escapes (e.g. &quote) since the attribute value will be assumed to be all that is enclosed between a pair of double quote characters.
- PCDATA paragraph terminators can be both <CR><LF> and ASCII(127) characters.
- PCDATA will be terminated upon receipt of </element>.

# APPENDICES

# A MESSAGE FORMATS

## A.1.1 CONTROL IDs

Each window can accept up to 900 controls. Control IDs are automatically generated and are never referenced by the application.

Messaging control IDs are:

| | |
|---|---|
| 000 | Window caption |
| 001-900 | User defined controls |
| 901-909 | DDE conversations (the user specifies DDE numbers 1-9) |
| 990 | Messagebox |
| 997 | Timer (for window time out control) |
| 998 | Clipboard |
| 999 | Message line |

**Control Types**

| | |
|---|---|
| A | Bitbutton |
| B | Command Button |
| C | Combobox |
| D | Document (Multiline memo) |
| E | Tab |
| F | Frame |
| G | Grid |
| H | Hypertext |
| I | Input (single line – Tedit) |
| J | Date/Time Picker |
| K | |
| L | Listbox |
| M | Meter |
| N | Outline |
| O | Output |
| P | Picture |
| Q | Splitter |
| R | Radiobutton |
| S | Select |
| T | Text (Label) |
| U | |
| V | |
| W | Workbench |
| X | Checkbox |
| Y | |
| Z | |

## A.1.2 MESSAGE STRUCTURE FROM HOST TO WORKSTATION

**Header - all messages start with this block**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 1 | 3 | Message length | 999 | Length of message |
| 4 | 2 | Message type | 99 | Message type |
| 6 | 2 | Win ID | 99 | Level of form (starts at 1) |

### A.1.2.1 FORM BUILD MESSAGES

**Type=10 Form header**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Form type | 1Byte | ASCII 64 + binary value<br>x0xx No System menu<br>x1xx System menu<br>xx00 No Border<br>xx01 Single line border<br>xx10 Fixed (dialog) border<br>xx11 Resizeable boarder<br>0xxx Not Document form<br>1xxx Document form |
| 9 | 1 | Events | 1 Byte | 1 = Pre-window<br>2 = application<br>4 = enable right mouse click |
| 10 | 1 | Page up/down | 1 byte | 0 = None<br>1 = page up<br>2 = Page down<br>3 = Both |
| 11 | 1 | Close on exit | 1 Byte | 0 = No, 1 = Yes |
| 12 | 1 | List/Select | 1 Byte | 0 = No, 1 = Yes |
| 13 | 3 | Start $Y position | 999 | |
| 16 | 3 | Start $X position | 999 | |
| 19 | 2 | Highlight code | 99 | 0-49 |
| 21 | 2 | Height | 99 | |
| 23 | 3 | Width | 999 | |
| 26 | 1 | Resize control type | 1 Byte | W, D, G, H, N,# (space if none) |
| 27 | 1 | Form type | 1 byte | Space, B (Banner) or L(Look-up) |
| 28 | 1 | Control/form resizing | 1 Byte | Space, V or H. Space indicates both if char 26 is not space. |
| 29 | 2 | Spare | 2 Bytes | Set to zeros |
| 31 | V | Form Caption | Var. | Initial caption text |

**Type=12 Control set up**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Control type | X | Alpha letter |
| 9 | 3 | Control ID | 9(3) | Tab order number |
| 12 | 3 | Start $Y position | 9(3) | Lines |
| 15 | 3 | Start $X position | 9(3) | Cell widths |
| 18 | 3 | Width | 9(3) | Cell width |
| 21 | 2 | Height | 9(2) | Lines |
| 23 | 1 | Flags | 9 | 0 = None set<br>1 = Read only<br>2 = Hide<br>4 = Multiselect (listbox)<br>Autoselect (Input)<br>Editable (Combobox) |
| 24 | 1 | Look-ups | 9 | 0=None, 1=Look-up, 2=Autolist |
| 25 | 1 | Alignment<br><br><br>Picture properties | 9 | 0 = left justify, 1 = Right justify<br>2 = Centered<br>4 = Wordwrap<br>0= none, 1=stretch,<br>2 = proportional<br>3 = stretch + proportional |
| 26 | 1 | Radiobutton: set number<br>Frame | 9<br>9 | 0-9<br>0= normal offset<br>1=full width, 2=full height<br>3=both |
| 27 | 2 | Highlight code | 9(2) | 0-49 |
| 29 | 1 | Action flags | 9 | 0= normal, 1= jump to, 2= cancel |
| 30 | 1 | Event markers | 1 byte | ASCII 32 + sum of<br>1 = gotfocus<br>2 = click<br>4 = Double Click<br>8 = Change<br>NB host determines events 5&6 from next control |
| 31 | 3 | Max chars | 9(3) | if zero use cell width |
| 34 | 1 | Case | 9 | 0 = as entered<br>1 = Uppercase<br>2 = Lowercase |
| 35 | 1 | Borderstyle/<br>Glyph for Bitbutton<br>Splitter details | 9 | 0 = lowered, 1 = raised, 2= none<br>0=top, 1=bottom, 2= left, 3=right<br>0=Vert, 1=Horz, +2=bringtofront |
| 36 | 1 | Anchors | 1 Byte | ASCII 32 +<br>1 = Anchor left<br>2 = Anchor right<br>4 = Anchor top<br>8 = Anchor Bottom |
| 37 | 1 | Spare | 1 Byte | |
| 38 | V | Caption | Var. | for T, X & R control types |

**Type=13 Control amend**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Control type | 1 byte | |
| 9 | 3 | Control ID | 9(3) | tab order |
| 12 | 3 | Start $Y position | 9(3) | |
| 15 | 3 | Start $X position | 9(3) | |
| 18 | 3 | Width | 9(3) | |
| 21 | 2 | Height | 9(2) | |
| 23 | 1 | Sundry flags | 1 byte | 0 = none, 1 = read only, 2 = hide<br>4 = Multiselect (listbox)<br>Autoselect (input) |
| 24 | 1 | Case | 1 byte | 0 = as entered, 1 = upper, 2=lower |
| 25 | 1 | Alignment | 1 byte | 0 = left, 1 = right, 2 = centered |
| 26 | 3 | Max chars | 9(3) | |
| 29 | 1 | Control position flags = composite values | 1 byte | 0 = no changes<br>1 = position change (T/L)<br>2 = Width change<br>4 = height change |
| 30 | V | Control caption | Var | for T, R, & X controls |

**Type=14 Control Hint amendment**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Control Type | 1 byte | |
| 9 | 3 | Control ID | 9(3) | Tab order |
| 12 | V | Text | X(V) | Hint text (max 250 bytes) |

**Type=16 Window menu sub-menu setup**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 2 | menu ID | 9(2) | ref menu numbers 1 – 99 |
| 10 | 3 | Sub menu ID | 9(3) | 0 = menu header, 1-999 items |
| 13 | V | Sub menu code | X(V) | Terminates with msgsep |
| | | Separator | X | Message separator |
| | V | Parent sub menu code | X(V) | If child menu.<br>Terminates with msgsep |
| | | Separator | X | Message separator |
| | V | Menu caption | X(V) | Menu Text |
| | | Separator | X | Message separator |
| | V | Bitmap number | 9(V) | Bitmap number (optional) |

**Type=19 Window setup end**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|

NB This consists of a 7 byte message only.

### A.1.2.2 WINDOW POPULATE MESSAGES

**Type=21 Window control values**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | |
| 11 | 1 | Value type | 9 | 0 = first/only<br>1 = Subsequent<br>2 = Item update<br>4 = End of update |
| 12 | V | Value | Var | For details see notes below |

The value element of the message is variable depending upon the control type and value type.

Control ID = 0 Value = window caption
Control ID = 999 Value type = message type, value = highlight code + message
For multiple message update (listbox, document, outline etc) value types will be 0, 1...1, 4

For Listbox controls

Update of text item, value type = 2 and value consists of:

sub-type/ listnumber / text where / = text separator
(sub type = null (amend text),
+ (insert item ),
– (delete item)

For grid controls:

if value type is 0 then value consists of either:

0, num rows, row 1 fixed, num cols, col 1 fixed, editable, lines, exactrows, colsizing, lastcolresize, rowselect, highlighting, fixed row lines, body row lines, line separator.

Or col number, col width, alignment (0, 1, 2 equates to L,R,C), [U/L – to force case entry], [Y = allow button on this column]

if value type is 1 then value consists of:

row number, col number/value[/highlighting[/graphic index]] where / = text separator

if value type is 2 then value is the cell selection and consists of:

from row, from col, to row, to col/ top, left/ canedit, maxchars, lookup available/case

Where "/" is the text separator character

If value type = 3 then value is the row to be added (positive) or deleted (negative)

If value type = 4 this is for column sizing and positions 12 onwards contain:

| Position | Value | Description |
|---|---|---|
| 12 | 0 | Indicates request for number of columns in grid. Return message. |
| 13 + | 0 | Return message contains count of grid columns |
| 12 | 1 | Request column size<br>Pos 13 + = col number |
| 12 | 2 | Set column size<br>Pos 13 + = Col number + text separator + column width (pixels) |

If value type = 5 then message(s) define extended editing using eidth a drop down list selection or invoking another form via an embedded button. To specify this type of editing the parameters are stored in the grid array subscripted by “celleditoptions” – see programmer guide.

| Position | Value | Description |
|---|---|---|
| 12 | 0 | First record:<br>Indicates initialisation record then fields (delimted by “commas”) from position 13 onwards contain:<br>Fld 1: Type of extended editing:<br>“ddl” (drop down list) or<br>“but” (Button – ellipsis)<br>Fld 2: Can modify result (0/1)<br>Fld 3: Button width<br>(as fraction of grid cell width<br>e.g. 1.5 = 150%)<br>Fld 4: Dropdown list visible count<br>(0=use default of 10) |
| 12 | 1 | For Drop down list only:<br>Positions 13 onwards=List item text |
| 12 | 2 | For Drop down list only:<br>End of list. Causes list to be assigned to cell and cell info set. |

For Workbench controls

if value type=0 then value (workbench header) consists of:
logwth, loghgt, ruler, startx, starty, stepx, stepy, grid, scaling, zoom

if value type=1 then value (control header) consists of:
subtype, controlname, left, top, width, height, hlcode, caption, align, borderstyle, fixed, resizeh, resizev, limits, fontstyle, fontscale, foreground colour, background colour, gridrows, gridcolumns.

if value type=2 then value (control selection) consists of subtype, controlname.

For Meter controls

the value consists of a 2 byte highlight code (the "selected highlight code") for background display followed by a numeric value (0-100) + optional "~"_display text

For Outline controls:

if value type = 0 then value(item detail) consists of:
open bmp index/ closed bmp index/ leaf bmp index
(null field = unchanged, / = text separator)

if value type =1 then value (item detail) consists of
level number/ parent index number/ expanded status/ text

if value type = 2 then value (item detail) consists of:
null/ index number/ text – amend text
“+”/ index number/ expanded status/ text/action - add item
“-“/ index number - delete
“S”/ index number - reset selection
Where / = text separator

action = 1 Insert at index
2 add child to index number
3 Add child to index number
4 Add item (index = 0)

if value type =4 then value consists of
Node 0 status /current selected item index where / = text separator

For Date/Time Controls

If value type = 0 then value of the date/time is in the message remainder
If value type = 1 then the message details contain, delimited by the message separator:
Type 0 = Time, 1 = Date (Using 'spin' buttons), 2=Date using DropDown List for calendar.
Format As defined in the programmers guide
Range Min, Max dates – not applicable for times.

For Tabs controls

If Value type is zero, existing tabs are cleared and message contains node 0 of control.
If value type is 1 then new tab text is message content
If value type is 4 = end of tabs

**Type=22 Window set focus**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | |

**Type=23 Window control list selections**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | |
| 11 | 1 | Value Type | 1 Byte | 0 = de-select, 1 = select |
| 12 | V | value | Var | List number |

**Type 24 Set control highlight code**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | |
| 11 | 2 | Highlight code | 9(2) | |
| 13 | 1 | FontStyle | 1 byte | F (fixed) P (proportional) or null (no change) |
| 14 | V | Font Size | Var | 999 set this as font size<br>999% set this as %age of original size<br>null no change |

**Type=25 Window control skip - Clear all skips**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|

**Type=26 Window control set status**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | |
| 11 | 1 | Status | 1 byte | 0 = visible, enabled<br>1 = visible, disabled<br>2 = invisible, disabled |

**Type=27 Menu enabled status For menubar items**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | set to 001 |
| 11 | 2 | Menu ID | 99 | 1-99 |
| 13 | 1 | Status | 9 | 0 = enabled,<br>1 = disabled,<br>2 = invisible |

**Type=27 Menu enabled status For menu sub-items**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | set to 000 |
| 11 | 2 | Menu ID | 99 | 1-99 |
| 13 | V | Status | Var | 0 = disabled, 1 = enabled, 2 = checked, 4 = invisible. Values can be summed. Byte position corresponds to menu item number. |

**Type=28 Bitmaps (for Picture and BitButton controls)**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | |
| 11 | 1 | Value type | 1 Byte | 0 = initialise<br>1 = data<br>4 = end of data<br>8 = load from index |
| 12 | V | Data | Var | if value type = 0 – size of BMP<br>if value type = 1 – Hex stream<br>if value type = 4 – no data<br>if value type = 8 – index number |

**Type=29 Bitmaps for index loading**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Index number | 9(3) | 0-999 |
| 11 | 1 | Value type | 1 byte | 0 = initialise,<br>1 = hex stream<br>4 = end |
| 12 | V | Data | Var | if value type = 0 – size of BMP<br>if value type = 1 – Hex stream<br>if value type = 4:<br>“A” = auto transparency<br>“H”nn – transparency colour = backgournd of H/L nn |

### A.1.2.3 DDE COMMUNICATIONS

**Type=41 DDE Conversation initiate**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | 900 + DDE channel no |
| 11 | 1 | Error code | 9 | Initialised to zero |
| 12 | 1 | Default mode | 9 | 0 = send, 1 = receive |
| 13 | V | Application name | Var | Name used within App\|Topic |
| | V | Use string list | Var | 0 = no, 1 = yes<br>set to 1 for Delphi servers and data strings >255 bytes |
| | V | | 9 | 0 = No<br>1 = Normal window, activated<br>2 = Normal window, nonactivated<br>3 = Minimized, not activated<br>4 = Minimized, activated<br>5 = Maximized, activated |
| | V | Program name | var | Application program name to start |
| | V | topic | Var | Name of Topic |

Variable field separated by text separator character.

**Type=42 DDE link items (send/receive data)**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | 900 + channel no |
| 11 | 1 | Error code | 9 | Initialised to zero |
| 12 | 1 | Mode | 9 | 0 = send, 1 = receive |
| 13 | 1 | Sequence | 9 | 0 = one & only<br>1 = First<br>2 = subsequent<br>4 = last |
| 14 | V | Data | Var | Item name<br>=<br>Value |

**Type=43 DDE link Execute commands**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | 900 + channel no |
| 11 | 1 | Error code | 9 | Initialised to zero |
| 12 | V | Command | V | Command string to execute |

**Type=49 DDE link terminate**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | 900 + channel no |
| 11 | 1 | Error code | 9 | Initialised to zero |

**Type=40 Command line execute**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Mode | 9 | Form opening mode |
| 9 | V | Command line | X(N) | As required |

### A.1.2.4 SUNDRY MESSAGES

**Type=50 Message box request**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | fixed value of 990 |
| 11 | 4 | Controls code | 9(4) | See Programmer manual |
| 15 | V | Message box title | X(40) | Form Caption – terminated by text separator |
| | | Message text | X(40) x 4 | Up to 4 lines of 40 characters delimited by the text separator |

**Type=51 Sundry request details**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Type | 9(1) | 1 = About box details |
| 9 | 3 | Spare | 9(3) | Initialised to zero |
| 12 | V | Data | X | if type =1 (delimiter = text sep)<br>Fld 1 = customer name |

**Type=52 Set Drag/Drop capability**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Control tab order |
| 11 | 1 | Seq number | 9 | 0= Start<br>1 = target setting<br>2= end of list |
| 12 | 1 | Target ctrl type | X(1) | one of I,D,P,H,W,L,N,G |
| 13 | 3 | Target control ID | 9(3) | Target control tab order |
| 16 | V | Target control parameter | V | If target ctrl = I,D,P,H,W – Null<br>if L<br>0 = Drop on Item<br>1 = Drop anywhere<br>if N<br>0 = drop on branch<br>1 = drop on leaf<br>2 = drop on either<br>if G<br>a^b^c^d^e where<br>a = 0 Cell must be empty<br>a =1 Cell can have value<br>b = null – any row<br>b = row number allowed<br>c = null – any column<br>c = column number allowed<br>d = Null -no row check<br>d = row number with cell not null<br>e = Null – no Column check<br>e = column no. with cell not null<br>NB ^ = text separator character |

**Type=53 Pop Up Menu populate**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Increments from 1 |
| 11 | 3 | Parent control ID | 9(3) | Zero if not child |
| 14 | 1 | Value type | 9 | 0 = initialise<br>1 = Item data<br>4 = Action |
| 15 | V | Item data | V | Caption<br>Checked (0/1)<br>Enabled (0/1)<br>Routine entrypoint<br>fields delimited by commas |

**Type=54 File manipulation**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Mode | 9(1) | 1 = file copy<br>2 = file move<br>3 = file rename<br>4 = file delete |
| 9 | 1 | Response char | 9(1) | 0 = OK, >0 = error code |
| 10 | 1 | Message separator char | 9(1) | Field separator |
| 11 | V | First file name & path | V | File name &path |
| V | 1 | Message separator char | 9(1) | Field separator |
| V | V | Second file name & path | V | Not applicable for mode 4 |

**Type=55 Printer messages to PC**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Action mode | 9(1) | 1 = open, 2 = data,<br>3 = close & print, 4 = abort |
| Mode = 1 | | | | |
| 9 | 1 | Print mode | X(1) | T = Text (default), R = Raw |
| 10 | 1 | Orientation | X(1) | P(ortrait), L(andscape) |
| 11 | 1 | Collate enabled | X(1) | Y or N |
| 12 | 1 | Page selection enabled | X(1) | Y or N (enables page range sel) |
| 13 | 1 | Font Selection | X(1) | Y or N (enables font dialog box) |
| 14 | 2 | Default Font Size | 9(2) | 01-99 |
| 16 | V | | V | fld 1 – first page default<br>fld 2 – Last page default<br>fld 3 – title for print manager<br>NB fields delimited by commas |
| Mode = 2 | | | | |
| 9 | 1 | Data type | X(1) | T = Text (default) , R = Raw<br>B = Binary (HEX encoded) |
| 10 | V | data | X(N) | Data stream.<br>If Text type then any ASCII 127 chars will be converted to CR+LF. Existing CR/LF chars will be passed through. |
| Mode 3/ 4 | | No more bytes used | | |

**Type=56 File transfer to PC messages**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Mode | 9(1) | 1 = Open, 3 = Data<br>3 = Close, 4 = Abort |
| 9 | 1 | File Type | X(1) | T = Text, B = Binary |
| | | | | |
| Mode = 1 | | | | |
| 10 | 1 | Overwrite enabled | X(1) | Y or N (used by dialog box) |
| 11 | 1 | Enable dialog form | X(1) | Y or N |
| 12 | V | Text fields delimited by text separator | X(N) | fld 1 = Title for dialog box<br>fld 2 = file name & path<br>fld 3 = Filters for dialog box |
| | | | | |
| Mode = 2 | | | | |
| 10 | V | Data string | X(N) | If file type is B data is Hex encoded |
| | | | | |
| Mode 3/4 | | No more bytes used | | |

**Type=57 Find File messages**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Type | X(1) | F = File, D= Directory |
| 9 | 1 | Dialog Box | X(1) | Y or N |
| 10 | V | Text fields delimited by text separator | X(N) | fld 1 = Title for dialog box<br>fld 2 = file name & path<br>fld 3 = Filters for dialog box |

**Type=59 File transfer from PC messages**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Mode | 9(1) | 1 = Open, 2 = Data<br>3 = Close, 4 = Abort |
| 9 | 1 | File Type | X(1) | T = Text, B = Binary |
| | | | | |
| Mode = 1 | | | | |
| 10 | 1 | Reserved | X(1) | Set to N |
| 11 | 1 | Enable dialog box | X(1) | Y or N |
| 12 | 1 | Display optione | 9(1) | 0 = no display<br>1 = Display<br>2 = display + cancel option |
| 13 | V | Text fields delimited by text separator | V | fld 1 = dialog box title<br>fld 2 = flie name & path<br>fld 3 = filters for dialog box |
| | | | | |
| Mode = 2 | | | | |
| 10 | 1 | Type | 9(1) | 0 = Data, 1 = End of Data |
| 11 | V | Data | V | data string.<br>if data type = “B” string is Hex |
| | | | | |
| Mode 3/4 | | No more bytes used | | |

**Type=64 Various functions**

**Colour function**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | | | 0 = colour function |
| 9 | 1 | Message type | 9 | 0 = Set Highlight code to colour(s)<br>1 = Get Highlight code colour(s)<br>2 = Get colour from dialog box. |
| 10 | 2 | Highlight code | 99 | 0-99. Ignored for message type 2. |
| 12 | V | Value | Var | For details see notes below |

Message type: 0 Set HL colour Value: rgb composite= foreground,background
e.g.1956724,29645624

1 Get HL colour Value: rgb composite= foreground~background
e.g. 1956724,29645624

2 Get Dlg colour Value: rgb composite (e.g. 29645624)

**Font function**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | | | 1 = Font Function |
| 9 | 3 | Control ID | 9(3) | Either control ID or Zeros |
| 12 | 1 | Message type | 9 | 0 = Get control font<br>1 = Set control font<br>2 = Get font from dialog box. |
| 13 | V | Value | Var | For details see notes below |

Value = Font Name, size, colour, Style e.g. Arial,10,2546893,BIU
Style can be any combinations of B, I and U (upper or lowercase)
For Message type 2 the control ID is ignored and can be zeros.

**Type=70 XML Report Print**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | fixed value of 000 |
| 11 | 1 | Message code | 9(4) | 0 = Application to GUI<br>1 = GUI to application |
| 12 | V | Message details | X(N) | For type 0 – filename (&path)<br>For type 1 - Response |

The message response field will contain:
0 If the report was produced
N:text If an exception occurred
1:Cannot open Report File
2:Print Cancelled
3:File Read Error
4:Cannot find xxxxxxxxxxx (x….x = image filename)

### A.1.2.5 SUNDRY INFORMATION MESSAGES

**Type=09 Windows information**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Zeros |
| 11 | 2 | request type | 99 | See below |
| 13 | V | Value | Var | For details see notes below |

| | | | | |
|---|---|---|---|---|
| Request type: | 00 | Get User logon ID | Value: | null |
| | 01 | Get environment var. | Value | Name of variable |

### A.1.2.6 FORM CONTROL MESSAGES

**Type=90 Window close action**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Action | 9(1) | 0/null = close form<br>1 = do not close form |

**The following messages are of 7 bytes only**

**Type=91 Message error detected by host**
**Type=92 Intermediate batch end (used for such as disp^lzsr calls)**
**Type=98 End of Host messages - return control to workstation**
**Type=99 Close Lastic GUI forms**

## A.1.3 MESSAGES FROM WORKSTATION TO HOST

**Header - all messages start with this block**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 1 | 3 | Message length | 999 | Length of message |
| 4 | 2 | Message type | 99 | Message type |
| 6 | 2 | Win ID | 99 | Level of form (starts at 1) |

**Type=09 Windows information**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Zeros |
| 11 | 2 | request type | 99 | See below |
| 13 | V | Value | Var | For details see notes below |

Request type: 00 Get User logon ID Value: User logon ID
01 Get environment var. Value: variable_name +separator+ value
e.g. varname_$c(30)_varvalue

**Type=30 Control gotfocus event**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Control Tab order |
| 11 | 1 | Type | 9 | 0 |

**Type=31 Control Change event**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Control Tab order |
| 11 | 1 | Type | 9 | 0 |
| 12 | V | Value | X(N) | Text for Input controls,<br>Index for Combobox controls |

**Type=32 Control lostfocus event**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Control Tab order |
| 11 | 3 | Next control ID | 9(3) | Next control tab order |
| 14 | 1 | Value type | 9(1) | 0 = singular<br>1 = first<br>2 = subsequent<br>4 = last |
| 15 | V | Control value | V | Control content |

**Type=33 Control click/Dblclick event**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Control Tab order |
| 11 | 1 | Type | 9(1) | 0 = de-select, 1 = select, 2=grid col button, 3=grid cell options 'but' seting |
| 12 | 1 | Click type | 9(1) | 0 = Click, 1 = Double click |
| 13 | V | Value | X(N) | Control content |
| | | | | |

**Control Contents:**

| Control type | Content |
|---|---|
| Input | Control text |
| Document | Line text |
| Combobox | Text (editable) or index (non-editable) |
| Listbox | index value |
| Radiotbutton,Checkbox | 0/1 |
| Select,Button | Null |
| Grid | $row_0$, $col_0$, $row_1$, $col_1$<ts><br>last cell row, last cell col <ts> last cell text <ts><br>toprow,leftcol<br>(subscript 0 = from, 1 = to)<br>Column number (if type = 2)<br>Edit button click row,col (if type =3) |
| Workbench | n<ts>left,top,right,bottom,multi-select<br>[n=zoom prop:0 -selected, 1=changed] |
| Outline | selected item index<ts>expanded status |
| Hyperlink | link sequence no<ts>link text |
| Date/Time Picker | date/time as numeric value |

**Type=34 Drag Drop**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Control Tab order |
| 11 | 1 | Type | 9 | 0 |
| 12 | 3 | Target Control ID | 9(3) | Control tab order |
| 15 | V | Target position | V | Ctrl types I, D, P, H – Null<br>Ctrl type L – null or List index<br>Ctrl type N – Index number<br>Ctrl type G – Row, Column<br>Ctrl type W – line, Column |

**Type=35 Function keys**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Control Tab order |
| 11 | 1 | Type | 9 | 0 = look-up, 1 = help,<br>2 = Application, 3 = repaint,<br>4 = Reset, 8 = Refresh |
| 12 | V | Value | X(N) | Current control's content for types 0 & 2 |

**Type=36 Menu select**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Current control tab order |
| 11 | 2 | Type | X(2) | Menu number (1-99) or PP for Popup menu |
| 13 | 3 | Value | 9(3) | Menu index (0 for header) |
| 16 | V | Menu code | X(V) | For form menus |
| | | Separator | X | Message separator |
| 15 | V | Routine entrypoint | X(N) | For pop-up menus only |

**Type=38 Notify**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Current control tab order |
| 11 | 3 | Message ID | X(3) | A unique number identifying the type of message. |
| 14 | V | Message data | X(V) | Variable content |

**Type=39 Window terminate**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Terminate code | 9(1) | 0 = close,<br>1 = page up,<br>2 = page down |

**Types 41 - 49 DDE responses**

As per types 41-49 above. If an error occurred the error code is set accordingly. This error code is passed back to the application as the response from the DDE function call.

**Type=50 Message box response**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Fixed at 990 |
| 11 | 1 | Response code | 9 | See programmer manual |

**Type=54 File manipulation responses**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Mode | 9(1) | 1 = file copy<br>2 = file move<br>3 = file rename<br>4 = file delete |
| 9 | 1 | Response char | 9(1) | 0 = OK, >0 = error code<br>1 = No such first file<br>2 = second file already exists<br>3 = second file dir does not exist<br>4 = action refused |
| 10 | 1 | Message separator char | 9(1) | Field separator |
| 11 | V | First file name & path | V | File name &path |
| V | 1 | Message separator char | 9(1) | Field separator |
| V | V | Second file name & path | V | Not applicable for mode 4 |

**Type=55 Printer messages from PC**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Action mode | 9(1) | 1 = open (only open messages are responded to) |
| 9 | 1 | Response code | 9 | 0 = open OK<br>1 = Not opened<br>2 = Already open |
| 10 | 1 | Collate | X(1) | Y or N |
| 11 | 1 | Page selection | X(1) | A (all) or R (range) or pages |
| 12 | V | string fields delimited by commas | V | fld 1 = number of copies<br>fld 2 = first page number<br>fld 3 = last page number |

**Type=56 Transfer to PC file responses only for mode 1**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Action type | 9(1) | 1 = Open, 2 = data<br>3 = close, 4 = abort |
| 9 | 1 | Response code<br>Applicable to Open mode only | 9 | 0 = open OK<br>1 = Not open /user abort<br>2 = already open<br>3 = no such directory<br>4 = file already exists |
| 10 | V | File name | V | File name & path if opened OK |

**Type=57 Find File messages from PC**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Response code | 9 | 0 = Dialog box abort<br>1 = No such file/Directory error<br>2 = No such file<br>3 = Directory exists<br>4 = file exists |
| 9 | 4 | File attribute codes | X(4) | r = read only<br>h = hidden<br>s = system<br>a = archive |
| 13 | V | File name | V | File name & path if opened OK |

**Type=59 File Xfer from PC - messages from PC**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | Action mode | 9(1) | 1 = Opening<br>2 = data |
| Mode = 1 | | | | |
| 9 | 1 | Response code | 9(1) | 0 = open OK<br>1 = Not open /user abort<br>2 = already open<br>3 = no such directory<br>4 = file does not exist |
| 10 | V | File name | V | File name & path if opened OK |
| | | | | |
| Mode = 2 | | | | |
| 9 | 1 | Response code | 9(1) | 0 = data attached<br>1 = file not open<br>4 = end of file transfer<br>8 = user aborted |
| 10 | V | Data | X(N) | Data (Hex encoded if Binary) |

**Types=60-69 General communication messages**

**Type=60 Clipboard**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Fixed at 998 for clipboard |
| 11 | 1 | Data type | 9(1) | 0 = Text |
| 12 | 1 | Action type | 9(1) | 0 = send data to PC clipboard<br>1 = Request data from PC clipboard<br>2 = data from clipboard |
| Action=0 | | | | |
| 13 | 1 | Stage | 9(1) | 1 = first message<br>2 = subsequent messages<br>4 = last message |
| 14 | V | Data | X(N) | Data to put on clipboard |
| | | | | |
| Action=1 | | No more bytes used | | |
| | | | | |
| Action=2 | | | | |
| 13 | 1 | response code | 9(1) | 0 = no text in clipboard<br>1 = first text message<br>2 = subsequent text lines<br>4 = end of clipboard text |
| 14 | V | Data | X(N) | Data from clipboard |

**Type=61 Form Metrics**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Form number. | 9(3) | form/screen number |
| 11 | 1 | message type | 9(1) | 0= request<br>1 = response from GUI<br>2 = Set metrics to GUI for forms |
| type = 0 | | NO more bytes | | |
| 12 | 3 | Control ID | 9(3) | 0 = form else control number |
| type = 1 | | | | |
| 15 | V | Form Metrics<br>fields delimited by text separator | V | fld 1 – Window State<br>fld 2 = Left<br>fld 3 = Top<br>fld 4 = Right<br>fld 5 = Bottom |

NB
If Form Number is 0 then request is for screen metrics (window state is null).
If Form Number >0 then this is the form number for metrics
If Control ID is zero then form metrics are sent/returned
If control ID is non-zero then control metrics are sent/returned.
(field 1 for control metrics (window state) is not relevant)
Cannot set screen metrics – only form & control level.

**Type=62 Document control functions**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | document control ID |
| 11 | 1 | message type | 9(1) | 1=find pos,len para N<br>2=find pos,len of text<br>3=get pos,len of selected text<br>4=set text selected for pos,len<br>5=get text in pos, length<br>6=replace text in pos, length<br>7=set selected text display mode |
| 12 | 1 | direction | 9(1) | 0=send to GUI<br>1=return from GUI |
| 13 | 1 | Record sequence | 9(1) | 0 = only<br>1=first, 2= subsequent, 4=last |
| | | | | |
| type = 1 | | | | |
| 14 | V | Send | 9(N) | Paragraph number |
| | | Return | 9^9 | pos, length, no paras, tot length |
| | | | | |
| type=2 | | | | |
| 14 | V | Send | 9^9<br>^<br>X(N) | Start position, ignore case<br>Text (excl <CR> or <LF>)<br>Delimited by msg sep |
| | | Return | 9^9 | Start, length (null if not found) |
| | | | | |
| type=3 | | | | |
| 14 | V | Send | 9(1) | 0 |
| | | Return | 9^9 | pos, length |
| | | | | |
| type=4 | | | | |
| 14 | V | Send | 9^9 | Pos, length |
| | | Return | | Null |
| | | | | |
| type=5 | | | | |
| 14 | V | Send | 9^9 | pos, length |
| | | Return | V | Selected text<br>Multiple messages to contain text |
| | | | | |
| type=6 | | | | |
| 14 | V | Send | 9^9^X | pos, length, text |
| | | | V | Replacement text<br>Multiple messages to contain text |
| Type=7 | | | | |
| 14 | 1 | mode | 9 | 0=show always<br>1=show when control has focus. |

^ represents the message separator character.

**Type=63 TCP/IP information**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 3 | Control ID | 9(3) | Fixed at 999 |
| 11 | 1 | Data type | 9(1) | 0 = Text |
| 12 | 1 | Action type | 9(1) | 0 = get info |
| Returned info | | | | Fields separated by message separator |
| 13 | V | Lastic GUI IP address | 9.9.9.9 | |
| | V | Lastic GUI Host name | X(N) | |
| | V | Lastic GUI Port number | 9(N) | |
| | V | Server IP address | 9.9.9.9 | |
| | V | Server Host name | X(N) | |
| | V | Server port number | 9(N) | |
| | | | | |

Called from Mumps function getip^lzsr(). Returns null if socket not open.

**Type=64 Various functions**

**Colour function**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | | | 0 = colour function |
| 9 | 1 | Message type | 9 | 0 = Set Highlight code to colour(s)<br>1 = Get Highlight code colour(s)<br>2 = Get colour from dialog box. |
| 10 | 2 | Highlight code | 99 | 0-99. Ignored for message type 2. |
| 12 | V | Value | Var | For details see notes below |

Message type: 0 Set HL colour Value: rgb composite= foreground,background
e.g.1956724,29645624

1 Get HL colour Value: rgb composite= foreground~background
e.g. 1956724,29645624

2 Get Dlg colour Value: rgb composite (e.g. 29645624)

**Font function**

| Start Pos | Length | Field type | Format | Description |
|---|---|---|---|---|
| 8 | 1 | | | 1 = Font Function |
| 9 | 3 | Control ID | 9(3) | Either control ID or Zeros |
| 12 | 1 | Message type | 9 | 0 = Get control font<br>1 = Set control font<br>2 = Get font from dialog box. |
| 13 | V | Value | Var | For details see notes below |

Value = Font Name, size, colour, Style e.g. Arial,10,2546893,BIU
Style can be any combinations of B, I and U (upper or lowercase)
For Message type 2 the control ID is ignored and can be zeros.

# B. TCP/IP ERROR CODES

The following table lists all Winsock error codes including those compatible with Berkeley Sockets. The majority of there errors will not apply to Lastic GUI as they are applicable to services and connectivity options not employed.
The most likely errors to occur are 10049-10054, 10060 due to network failure or excessive logon time.
Errors indicating address family, protocol or version problems will be associated with the connection parameters or the version of Winsock.dll in use.

| Code | Mnemonic | Meaning |
|---|---|---|
| 10004 | WSAEINTR | A blocking call has been cancelled while in operation. |
| 10009 | WSAEBADF | Standard C error |
| 10013 | WSAEACCES | Invalid attempt to send to a broadcast address |
| 10014 | WSAEFAULT | Connection address is incorrect. |
| 10022 | WSAEINVAL | The socket has not been bound, is already connected or the Winsock version specified is not supported by this DLL. |
| 10024 | WSAEMFILE | No more file descriptors are available. |
| 10035 | WSAWOULDBLOCK | The socket is marked as non-blocking and the connection has not been completed yet. |
| 10036 | WSAEINPROGRESS | A blocking Windows Sockets operation is in progress. |
| 10037 | WSAEALREADY | An asynchronous routine being cancelled has already completed. |
| 10038 | WSAENOTSOCK | The descriptor is not a socket. |
| 10039 | WSAEDESTADDRREQ | No destination address provided. |
| 10040 | WSAEMSGSIZE | A data packet was too long for the buffer and has been truncated. |
| 10041 | WSAEPROTOTYPE | The specified protocol is the wrong type for this socket. |
| 10042 | WSAENOPROTOOPT | The action required is not supported. |
| 10043 | WSAEPROTONOSUPPORT | The specified protocol is not supported. |
| 10044 | WSAESOCKTNOSUPPORT | The specified socket type is not supported in this address family. |
| 10045 | WSAEOPNOTSUPP | The referenced socket is not of the type that supports this service. |
| 10046 | WSAEPFNOSUPPORT | BSD error |
| 10047 | WSAEAFNOSUPPORT | The IP address family is not supported by the protocol specified. |
| 10048 | WSAEADDRINUSE | The specified address is already in use. |
| 10049 | WSAEADDRNOTAVAIL | The specified address is not available from the local machine. |
| 10050 | WSAENETDOWN | The Windows Sockets implementation has detected that the network sub-system has failed. |
| 10051 | WSAENETUNREACH | The network can't be reached from this host at this time. |
| 10052 | WSAENETRESET | Winsock has dropped the socket connection due to some unknown problem. |
| 10053 | WSAECONNABORTED | Connection has been aborted due to unknown failure. |
| 10054 | WSAECONNRESET | The connection has been reset by the remote side. |
| 10055 | WSAENOBUFS | No buffer space available. |
| 10056 | WSAEISCONN | The socket is already connected. |
| 10057 | WSAENOTCONN | The socket is not connected |
| 10058 | WSAESHUTDOWN | A read or write has been attempted on a socket that has been shut down. |
| 10059 | WSAETOOMANYREFS | BSD error |
| 10060 | WSAETIMEDOUT | The attempt to connect has timed out without establishing a connection. |

| Code | Mnemonic | Meaning |
|---|---|---|
| 10061 | WSAECONNREFUSED | The connection attempt has been forcibly rejected. |
| 10062 | WSAELOOP | BSD error |
| 10063 | WSAENAMETOOLONG | BSD error |
| 10064 | WSAEHOSTDOWN | The Host server network interface is down |
| 10065 | WSAEHOSTUNREACH | The Host server cannot be reached. |
| 10066 | WSAENOTEMPTY | BSD error |
| 10067 | WSAEPROCLIM | BSD error |
| 10068 | WSAEUSERS | BSD error |
| 10069 | WSAEDQUOT | BSD error |
| 10070 | WSAESTALE | BSD error |
| 10071 | WSAEREMOTE | BSD error |
| 10091 | WSASYSNOTREADY | The underlying network system is not ready for network communication or does not support TCP/IP Telnet service. |
| 10092 | WSAVERNOTSUPPORTED | The Winsock API version required is not supported by this particular Windows Sockets implementation. |
| 10093 | WSANOTINITIALISED | An attempt has been made to use Winsock though it has not been successfully started-up. |
| 11001 | WSAHOST_NOT_FOUND | Authoritative Answer Host not found. |
| 11002 | WSATRY_AGAIN | Non-Authoritative Host not found. |
| 11003 | WSANO_RECOVERY | Non-recoverable error has occurred. |
| 11004 | WSANO_DATA | Valid host name but no data record of the requested type. |