Zipher Text Communications Protocol

Abstract

This document describes the text communications protocol that can be used with Zipher machines. The protocol is intended to allow simple communication between Zipher machines and other devices.
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Author</th>
<th>Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Ben Yeomans</td>
<td>21st January 2003</td>
<td>Original</td>
</tr>
<tr>
<td>1.01</td>
<td>Ben Yeomans</td>
<td>21st March 2003</td>
<td>Removed reference to RS485</td>
</tr>
<tr>
<td>1.02</td>
<td>Ben Yeomans</td>
<td>16th June 2004</td>
<td>Added syntax for Price fields</td>
</tr>
<tr>
<td>1.03</td>
<td>Matthew Faithfull</td>
<td>12th July 2005</td>
<td>Added syntax for SLI, JDI, GPC and SPC</td>
</tr>
<tr>
<td>1.04</td>
<td>Ben Yeomans</td>
<td>07th July 2006</td>
<td>Added reference to TCP/IP support</td>
</tr>
<tr>
<td>1.05</td>
<td>Matthew Faithfull</td>
<td>23rd August 2006</td>
<td>Added Line Select commands</td>
</tr>
<tr>
<td>1.06</td>
<td>Matthew Faithfull</td>
<td>11th May 2007</td>
<td>Added IJS Interactive Job Selection</td>
</tr>
<tr>
<td>1.07</td>
<td>Matthew Faithfull</td>
<td>3rd September 2007</td>
<td>Added CQI Clear Queue Item</td>
</tr>
<tr>
<td>1.08</td>
<td>Phil Hart</td>
<td>9th July 2009</td>
<td>Documented CLN and SLN commands that were added in August 2006 but not documented.</td>
</tr>
<tr>
<td>1.09</td>
<td>Phil Hart</td>
<td>17th May 2010</td>
<td>Added FMP command</td>
</tr>
<tr>
<td>1.10</td>
<td>Mat Tivey</td>
<td>18th February 2011</td>
<td>Updated description of JDI command to reflect the actual implementation</td>
</tr>
<tr>
<td>1.11</td>
<td>Mat Tivey</td>
<td>25th February 2011</td>
<td>Fixed example for LDU command (returns id not ACK)</td>
</tr>
<tr>
<td>1.12</td>
<td>Kerri Brown</td>
<td>9th May 2011</td>
<td>Added GTD and TAD commands</td>
</tr>
<tr>
<td>1.13</td>
<td>Jeff Froggatt</td>
<td>4th January 2013</td>
<td>Added Async message support and GFT and GWN commands. Added QSZ command.</td>
</tr>
<tr>
<td>1.14</td>
<td>Jeff Froggatt</td>
<td>11th February 2013</td>
<td>Corrections to Async message documentation.</td>
</tr>
<tr>
<td>1.15</td>
<td>Jeff Froggatt</td>
<td>19th February 2013</td>
<td>More corrections to Async messages.</td>
</tr>
<tr>
<td>1.16</td>
<td>Jeff Froggatt</td>
<td>16th May 2013</td>
<td>Added GJL, GJF and GJD commands.</td>
</tr>
<tr>
<td>1.17</td>
<td>Jeff Froggatt</td>
<td>11th June 2013</td>
<td>Updated GJL, GJF and GJD commands.</td>
</tr>
<tr>
<td>1.18</td>
<td>Jeff Froggatt</td>
<td>10th October 2013</td>
<td>Removed reference to QUE command in table of commands as this was previously replaced with the OUT command.</td>
</tr>
<tr>
<td>1.19</td>
<td>Jeff Froggatt</td>
<td>31st October 2013</td>
<td>Added CMD commands for Bark (8610) project</td>
</tr>
<tr>
<td>1.20</td>
<td>Helcio Mayor</td>
<td>9th July 2014</td>
<td>Added SPD and GPD commands for 2300</td>
</tr>
<tr>
<td>1.21</td>
<td>Jeff Froggatt</td>
<td>19th September 2014</td>
<td>Modified SPD and GPD commands for 2300</td>
</tr>
<tr>
<td>1.22</td>
<td>Daniel Varley</td>
<td>23rd June 2015</td>
<td>Added serialisation commands SHD, SHO, SDO, SCF, SRC, SCB, SID, SNI, SLI, SMR, SGM and SFS. Also added SED, GED, GCL and DFS commands.</td>
</tr>
<tr>
<td>1.23</td>
<td>Helcio Mayor</td>
<td>23rd October 2015</td>
<td>Added JDAS, JDIS and JDUS commands.</td>
</tr>
<tr>
<td>1.24</td>
<td>Daniel Varley</td>
<td>28th October 2015</td>
<td>Changed serialisation SLI command to SLR.</td>
</tr>
<tr>
<td>1.25</td>
<td>S. Penfold</td>
<td>4th March 2016</td>
<td>Added Appendix 2: ‘CCFG Settings’</td>
</tr>
</tbody>
</table>
# Table of Contents

Abstract ................................................................................................................ 1  
Table of Contents ................................................................................................. 3  
Introduction ......................................................................................................... 5  
Conventions Used in this Document .................................................................. 5  
Running the Text Communications system ....................................................... 6  
Communications Sessions .................................................................................. 7  
- The Job Select Commands (SEL and SLA/SLI) .................................................. 10  
- The Line Assignment Command (LAS) ............................................................. 11  
- The Clear Line Command (CLN) ..................................................................... 12  
- The Stop Line Command (SLN) ...................................................................... 12  
- Interactive Job Selection (IJS) ........................................................................ 12  
- The Job Data Update Commands (JDU and JDA/JDI) .................................... 13  
- The Job Data Update with Save Commands (JDUS and JDAS/JDIS) ............ 14  
- Line Data Update Command (LDU) ................................................................ 15  
- Line Select Command (LSL) ......................................................................... 15  
- The Print Command (PRN) ........................................................................... 16  
- Force Micro Purge command (FMP) ............................................................... 16  
- The Set State Command (SST) ..................................................................... 16  
- The Get State Request (GSR) ...................................................................... 17  
- The Get All Faults Command (GFT) .............................................................. 18  
- The Get All Warnings (GWN) ...................................................................... 19  
- The Clear All Faults Command (CAF) ........................................................... 19  
- The Clear All Warnings Command (CAW) ................................................... 20  
- The Clear Single Error (CEM) ....................................................................... 20  
- The Get Counts Command (GPC) ................................................................. 21  
- The Set Counts Command (SPC) .................................................................. 21  
- The Clear Queue Item Command (CQI) ......................................................... 22  
- The Get Time and Date Command (GTD) ....................................................... 22  
- The Set Date and Time Command (TAD) ....................................................... 22  
- The Get Job Name Command (GJN) .............................................................. 23  
- The Queue Size Command (QSZ) ................................................................ 23  
- The Get Job List Command (GJL) ................................................................ 24  
- The Get Job Fields Command (GJF) ............................................................. 24  
- The Get Current Job Data Command (GJD) ................................................... 26  
- The Device Specific Commands and Requests (CMD) .................................... 26  
- The Set Print Density command (SPD) ......................................................... 27  
- The Get Print Density command (GPD) ......................................................... 28  
- The Serialisation Header and Data command (SHD) ....................................... 29  
- The Serialisation Header Only command (SHO) ............................................ 30  
- The Serialisation Data only command (SDO) ................................................ 30  
- The Serialisation Change Field Data (SCF) .................................................... 30  
- The Serialisation Record Count command (SRC) .......................................... 31  
- The Serialisation Clear Buffer command (SCB) ............................................ 31  
- The Serialisation Indexed Data command (SID) ............................................ 32  
- The Serialisation Free Space command (SFS) .............................................. 32  
- The Serialisation Next Record Index command (SNI) .................................... 32  
- The Serialisation Last Record Index command (SLR) .................................... 33  
- The Serialisation Set Maximum Records command (SMR) ......................... 33  
- The Serialisation Get Maximum Records command (SGM) ......................... 34  
- The Set Encoder Direction command (SED) ............................................... 34  
- The Get Encoder Direction command (GED) .............................................. 34  
- The Get Consumable Levels command (GCL) .......................................... 35  
- The PackML Commands and Requests (PML) ............................................ 36  
- Async Notifications ...................................................................................... 39  
- The Overall State change Notification (STS) ............................................... 39  
- The Print Start Notification (PRS) ................................................................. 39  
- The Print complete Notification (PRC) ......................................................... 40
The I/O Output changes Notification (OUT) ................................................................. 40
The Error state change Notification (ERS) ................................................................ 40
The Current Job changed Notification (JOB) .............................................................. 41
The Set Async Notification command (SAN) .............................................................. 41
The Get Async Notification command (GAN) ............................................................ 42
The Enable All Notifications commands (EAN) ......................................................... 42
The Disable All Notifications command (DAN) ......................................................... 42
The Disable Print Notifications command (DPN) ....................................................... 43
The Set Single Notification command (SNO) ............................................................. 43
Example Code ........................................................................................................ 46
Appendix 1: State Transition Diagram ................................................................. 48
Introduction

This document describes the text communications system that can be used to communicate with a Zipher printer. The text protocol works over RS232 connections and (in software releases from June 2006) TCP/IP connections. The protocol is master-slave with the host machine considered as the master, and responsible for initiating any communications.

Conventions Used in this Document

All character values described in this document are text values. ANSI (1 byte per character) strings only are supported by the default text communications protocol, although a Unicode version is also available (see below).

Binary values are not used in this communications system; sending the string “65” means sending the character for ‘6’ followed by the character ‘5’.

The term “PackML” in this document refers to the “Automatic Mode Machine States Definition Version 2.1” by the OMAC Motion for Packaging Working Group, PackML Subcommittee. This document is available from http://www.omac.org/.
Running the Text Communications system

From CLARiTYConfig, Upload the parameters from the CLARiTY unit and open the System collection. There will be collections for “ComPort1” and “ComPort2”. Within these collections is a “Usage” entry which can be set as follows:

0. Deactivated  
1. Text Communications System  
2. Binary Communications System

Care should be taken if CLARiTYConfig is connected via a serial cable, as it is possible to shut out the CLARiTYConfig connection by changing it to “Deactivated” or “Text Communications System”. If this happens and it is not possible to reconnect by other means (Ethernet or the other serial port) the serial port settings can be restored to their default values (ComPort1 = Binary Communications System, ComPort2 = Deactivated) via the CLARiTY user interface. To do this, enter the **Tools** screen, then press the **Setup** button followed by **Control**, **Communications** and finally **Reset serial ports**.

Running the text communications system on both serial ports at the same time is not supported and attempting to do so may cause problems with CLARiTY.

When connecting a device to the text communications system, the first thing to ensure is that the serial port of the device to be connected is configured to match the configuration of the serial port on the CLARiTY unit. The settings should be:

- Bits per second (Baud rate): As configured in CLARiTYConfig
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow Control: As configured in CLARiTYConfig

If necessary, the Hyperterminal program, which ships with most versions of Microsoft Windows, can be used to test the text communications system.

The text communications system works best as a request/response protocol, where the device connected to the CLARiTY unit sends a command to the unit, and then waits for a response before sending another command. In some cases it is possible to send a command before the previous command has completed, but in most cases this will result in the second command failing.

To ensure that the commands you need are available and that the correct protocol mode is in use enter the **Tools** screen, the press the **Diagnostics** button followed by **Control**, **Communications** and finally select the **Communication port** that is set to use Text Communications.

The **Text Comms Protocol** item will now display the mode in use and installed version of the Text Comms Protocol. For example **Zipher ASCII Comms**, the version 1 ASCII protocol or **Zipher UNICODE Comms v2**, the version 2 UNICODE protocol. To determine which commands are available in the installed version of the protocol see the table below.

To switch between ASCII and UNICODE modes use CLARiTY Configuration Manager to set **System/TextCommsParameterFile** to the file name for the protocol definition file, either ZipherASCIICommsProtocol.xml or ZipherUNICODECommsProtocol.xml.

More recent software releases (from June 2006) also support the Text Communications Protocol via TCP/IP. This can be activated by using CLARiTYConfig to set the **System/TCPIP/TextCommunications/NetworkPort** parameter to a non-zero value. The Text Communications system will then wait for connections on the TCP port specified. Take care to avoid using the same port number as other services that may be in operation, such as Binary Communications (usually port 3001) or Master/Slave mode (usually port 3002).
Communications Sessions

A communications session comprises a set of commands or requests and responses. The carriage return character (ANSI character code 13 decimal / 0x0D hexadecimal) terminates each command or request, and also has the function of resetting the parser. This means that the carriage return character will stop parsing the current data even if the data has partially matched a command or request. The Carriage Return-Line Feed sequence (0x0D 0x0A) is also accepted as a terminator; the line feed character will simply be ignored.

Henceforth in this document, the carriage return character will be written <CR>.

Most commands return either the default success response or the default failure response depending on whether or not the command was successfully executed.

The default success response is:

ACK<CR>

The default failure response is:

ERR<CR>

The examples given below use the character `>` to indicate data sent to the printer, and `<` to indicate data being received from the printer.

For example:

>`CAF<CR>`

<ERR<CR>

means “the string ‘CAF<CR>’ was sent to the printer, which responded with the string ‘ERR<CR>’”.

Because the carriage return character resets the parser, it is recommended to send the carriage return character before each communications session in order to clear any unwanted data which may be in the parser’s buffer.

This example sends the parser reset character before sending two commands to the printer:

>`<CR>`

>`JDU|data|<CR>`

<ACK<CR>`

>`PRN<CR>`

<ACK<CR>`
The commands and requests accepted by the Zipher Text Communications Protocol are as follows:

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEL</td>
<td>Job Select (data inserted into consecutive fields)</td>
</tr>
<tr>
<td>SLA</td>
<td>Job Select (data inserted into named fields)</td>
</tr>
<tr>
<td>JDU</td>
<td>Job Data Update (data inserted into consecutive fields)</td>
</tr>
<tr>
<td>JDA</td>
<td>Job Data Update (data inserted into named fields)</td>
</tr>
<tr>
<td>PRN</td>
<td>Print</td>
</tr>
<tr>
<td>SST</td>
<td>Set printer state</td>
</tr>
<tr>
<td>GST</td>
<td>Get printer state</td>
</tr>
<tr>
<td>CAF</td>
<td>Clear All Faults</td>
</tr>
<tr>
<td>PML</td>
<td>PackML commands and requests</td>
</tr>
</tbody>
</table>

**The following commands require Version 2 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLI</td>
<td>Job Select with Allocation (data inserted into named fields)</td>
</tr>
<tr>
<td>JDI</td>
<td>Job Data Update with Allocation (data inserted into named fields)</td>
</tr>
<tr>
<td>GPC</td>
<td>Get Counts</td>
</tr>
<tr>
<td>SPC</td>
<td>Set Counts</td>
</tr>
</tbody>
</table>

**The following commands require Version 3 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>Line Assignment (data inserted into named fields)</td>
</tr>
<tr>
<td>LDU</td>
<td>Line Data Update (data inserted into named fields)</td>
</tr>
<tr>
<td>LSL</td>
<td>Line Select</td>
</tr>
<tr>
<td>CLN</td>
<td>Clear Line (Line set to Non-Printing Job)</td>
</tr>
<tr>
<td>SLN</td>
<td>Stop Line (Line set to Production Disabled)</td>
</tr>
</tbody>
</table>

**The following commands require Version 4 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJS</td>
<td>Interactive Job Selection</td>
</tr>
<tr>
<td>CQI</td>
<td>Clear Queue Item</td>
</tr>
</tbody>
</table>

**The following commands require Version 5 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTD</td>
<td>Get Time And Date</td>
</tr>
<tr>
<td>TAD</td>
<td>Set Time And Date</td>
</tr>
</tbody>
</table>

**The following commands require Version 6 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAN</td>
<td>Get async notification enable map</td>
</tr>
<tr>
<td>SAN</td>
<td>Set async notification enable map</td>
</tr>
<tr>
<td>DPN</td>
<td>Disable print notifications and enable all others</td>
</tr>
<tr>
<td>SNO</td>
<td>Enable/Disable a Single Notification</td>
</tr>
<tr>
<td>CEM</td>
<td>Clear error (Fault or Warning)</td>
</tr>
<tr>
<td>GFT</td>
<td>Gets the current Faults</td>
</tr>
<tr>
<td>GWN</td>
<td>Gets the current Warnings</td>
</tr>
<tr>
<td>EAN</td>
<td>Async send of current overall status</td>
</tr>
<tr>
<td>ERS</td>
<td>Async send of current error status</td>
</tr>
<tr>
<td>JOB</td>
<td>Async send of current job details</td>
</tr>
<tr>
<td>PRS</td>
<td>Async send of print start</td>
</tr>
<tr>
<td>PRC</td>
<td>Async send of print complete</td>
</tr>
<tr>
<td>OUT</td>
<td>Async send of various output changes</td>
</tr>
<tr>
<td>QSZ</td>
<td>Gets the size and status of the update queue</td>
</tr>
<tr>
<td>GJN</td>
<td>Gets the selected job name and line selection</td>
</tr>
</tbody>
</table>

**The following commands require Version 7 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD</td>
<td>Perform printer specific command</td>
</tr>
</tbody>
</table>

**The following commands require Version 8 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPD</td>
<td>Set Print Density (2300 only)</td>
</tr>
<tr>
<td>GPD</td>
<td>Get Print Density (2300 only)</td>
</tr>
</tbody>
</table>

**The following commands require Version 9 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDL</td>
<td>Get Job List</td>
</tr>
<tr>
<td>GJF</td>
<td>Get Job Field List</td>
</tr>
<tr>
<td>GJD</td>
<td>Get Current Job Data</td>
</tr>
</tbody>
</table>

**The following commands require Version 10 or greater of the Text Communications Protocol**

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPD</td>
<td>Set Print Density (2300 only)</td>
</tr>
<tr>
<td>GPD</td>
<td>Get Print Density (2300 only)</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>SPD</td>
<td>Set Print Density (Master and Slave version)</td>
</tr>
<tr>
<td>GPD</td>
<td>Get Print Density (Master and Slave version)</td>
</tr>
<tr>
<td><strong>The following commands require Version 10 or greater of the Text Communications Protocol</strong></td>
<td></td>
</tr>
<tr>
<td>SHD</td>
<td>Serialisation Header and Data</td>
</tr>
<tr>
<td>SHO</td>
<td>Serialisation Header Only</td>
</tr>
<tr>
<td>SDO</td>
<td>Serialisation Data Only</td>
</tr>
<tr>
<td>SCF</td>
<td>Serialisation Change Field Data</td>
</tr>
<tr>
<td>SRC</td>
<td>Serialisation Record Count</td>
</tr>
<tr>
<td>SCB</td>
<td>Serialisation Clear Buffer</td>
</tr>
<tr>
<td>SID</td>
<td>Serialisation Indexed Data</td>
</tr>
<tr>
<td>SFS</td>
<td>Serialisation Free Space</td>
</tr>
<tr>
<td>SNI</td>
<td>Serialisation Next Record Index to be printed</td>
</tr>
<tr>
<td>SLR</td>
<td>Serialisation Last Record Index printed</td>
</tr>
<tr>
<td>SMR</td>
<td>Serialisation Set Maximum number of Records</td>
</tr>
<tr>
<td>SGM</td>
<td>Serialisation Get Maximum number of Records</td>
</tr>
<tr>
<td>SED</td>
<td>Set Encoder Direction</td>
</tr>
<tr>
<td>GED</td>
<td>Get Encoder Direction</td>
</tr>
<tr>
<td>GCL</td>
<td>Get Consumable Levels</td>
</tr>
<tr>
<td>DFS</td>
<td>Determine Free Space</td>
</tr>
</tbody>
</table>

These are now described in detail.
The Job Select Commands (SEL and SLA/SLI)

These commands cause the selection of a job on the printer. Values for variable fields may also be specified. These commands are not available when Line Selection is enabled. Please see Line Assignment below.

Syntax

SEL|<jobname>|<data>|...|<CR>
SLA|<jobname>|<fieldname>=<data>|...|<CR>
SLI|<jobname>|<allocation>|<fieldname>=<data>|...|<CR>

Return Values

On success, SEL and SLA return the default success response, SLI returns an ID for the item placed in the job queue by this command. On failure, the default failure response is returned. If the command succeeds, the response is sent immediately.

Details

The difference between SEL and SLA lies in the way values for variable fields are matched to field names. SEL assigns consecutive values to fields named “VarField00”, “VarField01”, “VarField02” etc.; SLA allows the field name to be specified in the command, useful if the job has fields which are not named “VarField00” etc. or if only a few fields need to be updated and using SEL would require the inclusion of values for fields that do not need updating. If the SEL command is to be used, care should be taken when designing the job that the fields to be updated have the correct names. SLI is similar to SLA except for the additional allocation parameter and the return id value.

Date Fields

Date fields are updated as for normal fields, but the correct format must be used. In the case of a User Entered Date, the correct format is:

```
dd/mm/yyyy
```

In the case of a Calculated Date, the correct format is:

```
ddd/mmm/yy
```

with negative values permitted for any of the components.

Please note that a Calculated Date depends on an underlying date calculation, and that it is this that will be updated by the command. If the job contains several Calculated Dates which obtain their value from the same date calculation, all of these will be updated.

Counter Fields

These are handled as normal fields.

Price Fields

These should be entered as decimal values. The decimal separator may be the point (.), comma (,) or centre dot (•). The value will be interpreted according to the number of decimal places specified for that field. For example, if the field has the default number of decimal places (2), the value “15” is interpreted as “15.00”, the value “15.5” as “15.50”, the value “15.076” as “15.07”, the value “.99” as “0.99” etc.

Allocation

This integer value will determine the number of prints allocated for the selected job.

The most likely reason for a failure on this command is trying to select a job which does not exist in the database. Specifying an invalid field name for a variable field or using an inappropriate format for a date field is not considered an error; the value will simply be ignored.

Examples

```
»SEL|Default 4 Line Text|<CR>
«ACK<CR>
```
Selects the job “Default 4 Line Text” without specifying any data for variable fields.

»SEL|Default 4 Line Text|First Field|Second Field|<CR>
«ACK<CR>
Selects the job “Default 4 Line Text” and sets the field “VarField00” to the value “First Field” and the field “VarField01” to the value “Second Field”.

»SLA|Default 4 Line Text|<CR>
«ACK<CR>
Selects the job “Default 4 Line Text” without specifying any data for variable fields.

»SLA|Default 4 Line Text|Line 1=First Line|Line 4=Fourth Line|<CR>
«ACK<CR>
Selects the job “Default 4 Line Text” and sets the field “Line 1” to the value “First Line” and the field “Line 4” to the value “Fourth Line”.

»SLA|DateTest|15000|<CR>
«2<CR>
Selects the job “Default 4 Line Text” with an allocation of 1500 prints and without specifying any data for variable fields. An id e.g. 2 is returned.

»SLA|DateTest|3000|Line 1= First Line|Line 4=Fourth Line|<CR>
«3<CR>
Selects the job “Default 4 Line Text” with an allocation of 3000 and sets the field “Line 1” to the value “First Line” and the field “Line 4” to the value “Fourth Line”. An id e.g. 3 is returned.

The Line Assignment Command (LAS)

When selection between multiple production lines (Line Selection) is enabled on the printer the process of Job Selection is replaced by the assignment of a Job to a numbered Line (Line Assignment).

Syntax

LAS|<jobname>|<line>|<fieldname>=<data>|…|<CR>

Return Values

On success LAS returns the queue item identifier for the line assignment item generated by the command. On failure the default failure response is returned. The response is sent immediately.

Details

The line value is a number between 1 and the number of lines for which the printer is configured. This has an absolute maximum of 16. Out of range values will cause a default failure response. Allocations are not currently supported in Line Select mode. The LAS command is otherwise identical to SLA in terms of its field names and data values.

Examples

»LAS|DateTest|8|OffsetDate=-010/002/00| <CR>
«ACK<CR>
Assigns the job “DateTest” to line 8 and sets the field “OffsetDate” to be two months minus ten days after its reference date.

**The Clear Line Command (CLN)**

When selection between multiple production lines (Line Selection) is enabled the clear line command allows the Line to have a Non Printing Job assigned to it.

**Syntax**

```
CLN|<line>|<CR>
```

**Return Values**

On success, returns the default success response. On failure, returns the default failure response.

**Details**

The line value is a number between 1 and the number of lines for which the printer is configured. This has an absolute maximum of 16. Out of range values will cause a default failure response.

**Examples**

```
»CLN|1|<CR>
«ACK<CR>
Assigns the Non Printing job to line 1.
```

**The Stop Line Command (SLN)**

When selection between multiple production lines (Line Selection) is enabled the clear line command allows the Line to have Production Disabled assigned to it.

**Syntax**

```
SLN|<line>|<CR>
```

**Return Values**

On success, returns the default success response. On failure, returns the default failure response.

**Details**

The line value is a number between 1 and the number of lines for which the printer is configured. This has an absolute maximum of 16. Out of range values will cause a default failure response.

**Examples**

```
»SLN|1|<CR>
«ACK<CR>
Assigns Production Disabled job to line 1.
```

**Interactive Job Selection (IJS)**

This command enables manual job selection to be remotely triggered by Text Comms. The Job Name and any Line Assignment (see LAS) and Allocation are set by the command. Variable data entry and preview confirmation are carried out on screen by the user.
Syntax

IJS|<line>|<jobname>|<allocation>|<CR>

Return Values

On success an interactive job selection will be initiated. Failure modes may be indicated by a message on the printer screen or by the standard error result or by no response and no interactive job selection being triggered.

Details

The line value is a number between 1 and the number of lines for which the printer is configured. If no lines are configured this values should be 0 or left blank e.g. IJS||<jobname>|<allocation>|<CR>.

Out of range values for line assignment will give the default failure response.

Examples

»IJS||TestJob||<CR>

Triggers Interactive Job Selection of the Job ‘TestJob’, with no allocation limit, when not in Line Select Mode.

»IJS|1|TestJob|100|<CR>

Triggers Interactive Job Selection of the Job ‘TestJob’, with an allocation of 100, allocated to the first line (Line 0), in Line Select mode.

The Job Data Update Commands (JDU and JDA/JDI)

These commands cause the variable fields on the currently selected job to be updated. These commands are not available when Line Selection is enabled. Please see Line Data Update below.

Syntax

JDU|<data>|...|<CR>
JDA|<fieldname>=<data>|...|<CR>
JDI|<allocation>|<fieldname>=<data>|...|<CR>

Return Values

On success JDU and JDA return the default success response JDI returns a queue item identifier for the job update item generated by the command. On failure the default failure response is returned. If the command succeeds, the response is sent immediately.

Details

The commands JDU, JDA and JDI work in the same way as SEL, SLA and SLI respectively, except that they do not select a new job, but update variable fields and allocations in the currently selected job.

The most likely reason for a failure on this command is trying to update fields when no job is selected. Specifying an invalid field name for a variable field is not considered an error; the value will simply be ignored.

Examples
»JDU|First Field|Second Field|<CR>
«ACK<CR>
Updates variable fields on the current job; sets the field “VarField00” to the value “First Field” and
the field “VarField01” to the value “Second Field”.

»JDA|Line 1=First Line|Line 4=Fourth Line|<CR>
«ACK<CR>
Updates variable fields on the current job; sets the field “Line 1” to the value “First Line” and the
field “Line 4” to the value “Fourth Line”.

»JDI|400|Line 1=First Line|Line 4=Fourth Line|<CR>
«3<CR>
Updates variable fields on the current job; sets the field “Line 1” to the value “First Line” and the
field “Line 4” to the value “Fourth Line”. The allocation for the job is reset to 400. An id e.g. 3 is
returned.

The Job Data Update with Save Commands (JDUS and JDAS/JDIS)

These commands cause the variable fields on the currently selected job to be updated and
corresponding changes to be saved permanently. These commands are not available when Line
Selection is enabled. Please see Line Data Update below.

Syntax

JDUS|<data>|...|<CR>
JDAS|<fieldname>=<data>|...|<CR>
JDIS|<allocation>|<fieldname>=<data>|...|<CR>

Return Values

On success JDUS and JDAS return the default success response. JDIS returns a queue item identifier
for the job update item generated by the command. On failure the default failure response is
returned. If the command succeeds, the response is sent immediately.

Details

The commands JDUS, JDAS and JDIS work in the same way as JDU, JDA and JDI respectively, except
that they save the job changes permanently. The printer will remember the values of the fields
updated by this command upon re-start at the printer.

The most likely reason for a failure on this command is trying to update fields when no job is
selected. Specifying an invalid field name for a variable field is not considered an error; the value
will simply be ignored.

Notes:

This command should be used judiciously as excessive use can drastically reduce the life of the
internal memory card.

Examples

»JDUS|First Field|Second Field|<CR>
«ACK<CR>
Updates variable fields on the current job; sets the field “VarField00” to the value “First Field” and
the field “VarField01” to the value “Second Field” and save these changes to the internal memory
card.

»JDAS|Line 1=First Line|Line 4=Fourth Line|<CR>
«ACK<CR>
Updates variable fields on the current job; sets the field “Line 1” to the value “First Line” and the
field “Line 4” to the value “Fourth Line” and save these changes to the internal memory card.
Updates variable fields on the current job; sets the field “Line 1” to the value “First Line” and the field “Line 4” to the value “Fourth Line” and save these changes to the internal memory card. The allocation for the job is reset to 400. An id e.g. 3 is returned.

**Line Data Update Command (LDU)**

When selection between multiple production lines (Line Selection) is enabled on the printer the process of Job Data Update is replaced by the updating of fields in the job assigned to a numbered Line (Line Data Update).

**Syntax**

LDU|<line>|<fieldname>=<data>|…|<CR>

**Return Values**

On success LDU returns a queue item identifier for the line update item generated by the command. On failure the default failure response is returned. The response is sent immediately.

**Details**

The line value is a number between 1 and the number of lines for which the printer is configured. This has an absolute maximum of 16. Out of range values will cause a default failure response. Allocations are not currently supported in Line Select mode. The LAS command is otherwise identical to JDA in terms of its field names and data values.

**Examples**

```plaintext
LDU|8|OffsetDate=-020/003/00|<CR>
<1<CR>
```

Updates variable fields on the job assigned to line 8; sets the field “OffsetDate” to be three months minus twenty days after its reference date.

**Line Select Command (LSL)**

When selection between multiple production lines (Line Selection) is enabled on the printer the Line Select command becomes available.

**Syntax**

LSL|<line>|<CR>

**Return Values**

On success LSL returns the default success response, and on failure it returns the default failure response. The response is sent immediately.

**Details**

The line value is a number between 1 and the number of lines for which the printer is configured. This has an absolute maximum of 16. Out of range values will cause the default failure response.

**Examples**

```plaintext
LSL|5|
```

```plaintext
ACK<CR>
```
Sets line 5 to be the active line. A Job Select for the job assigned to Line 5 will be added to the job selection queue.

**The Print Command (PRN)**

This command causes the currently selected job to be printed once.

**Syntax**

```
PRN<CR>
```

**Return Values**

On success, returns the default success response. On failure, returns the default failure response. If the command succeeds, the response is sent after the job has been printed.

**Details**

The PRN command attempts to print the current job.

*Note: This command is intended for use only with Intermittent TTO printers.*

**Examples**

```
»PRN<CR>
«ACK<CR>
```

The current job was successfully printed.

**Force Micro Purge command (FMP)**

This command causes the currently printers that have the functionality to Micro Purge. Micro purging ejects a small amount of ink from the printhead. It can only be performed when the printer is either Online or Offline. Attempting to perform it in any other state will result in an ERR command being returned. If the printer is Online and printing the Micro Purge will occur when the print completes. The result is only returned when the command completes so the ACK response can be delayed if the printer is printing.

**Syntax**

```
FMP<CR>
```

**Return Values**

On success, returns the default success response. On failure, returns the default failure response. If the command succeeds, the response is sent after the micro purge has completed.

**Details**

The FMP command attempts to carry out a Micro Purge.

**Examples**

```
»FMP<CR>
«ACK<CR>
```

The micro purge was successfully completed.

**The Set State Command (SST)**

This command attempts to set the overall state of the printer.
Syntax

\[ \text{SST}|<\text{desiredstate}|<\text{CR}\] 

Return Values

On success, returns the default success response. On failure, returns the default failure response. If the command succeeds, the response is sent after the state transition has taken place.

Details

The SST command attempts to set the overall state. Valid values for \text{desiredstate} are:

- 0  Shut down
- 1  Starting up
- 2  Shutting down
- 3  Running
- 4  Offline

The most likely reason for a failure on this command is trying to perform a state transition which is not supported. Please refer to Appendix 1 for the state transition diagram showing the permitted state transitions.

Examples

\[\text{SST}|1|<\text{CR}>\]  
\[\text{ACK}|<\text{CR}>\]  
Starts up the printer

\[\text{SST}|0|<\text{CR}>\]  
\[\text{ERR}|<\text{CR}>\]  
An attempt to shut down the printer. The error is caused because it is not possible to enter state 0 directly, the printer must be moved into state 2 to shut down the printer.

The Get State Request (GST)

This request retrieves various state values from the printer.

Syntax

\[ \text{GST}|<\text{CR}>\] 

Return Values

\[ \text{STS}|<\text{overallstate}|<\text{errorstate}|<\text{currentjob}|<\text{batchcount}|<\text{totalcount}|<\text{CR}\] 

Returns immediately.

Details

The GST request retrieves the following values:

- \text{overallstate}  
  The overall state of the printer, as described for the SST command.

- \text{errorstate}  
  The error condition of the printer. This will be one of the following:
  - 0  No errors
  - 1  Warnings present
  - 2  Faults present

- \text{currentjob}  
  The job selected in the printer. This will be empty if no job is selected.

- \text{batchcount}  
  The number of batches in the printer.
The printer’s batch count.

**totalcount**

The printer’s total count.

**Examples**

```plaintext
»GST<CR>
«STS|3|0|Default 4 Line Text|4345|8253|<CR>
```

The printer is running, there are no errors present, the currently selected job is “Default 4 Line Text”, the batch count is 4345 and the total count is 8253.

```plaintext
»GST<CR>
«STS|4|2||0|8253|<CR>
```

The printer is offline with a fault, there is no job selected, the batch count is 0 and the total count is 8253.

**The Get All Faults Command (GFT)**

This command lists all current faults in the printer.

**Syntax**

GFT<CR>

**Return Values**

On success, returns the total count of the number of faults followed by a list of current faults. On failure, returns the default failure response.

```plaintext
FLT|<count>|[<nnnn>|<clearable>|<title>|]<CR>
```

Part within square brackets repeats for each fault present.

**Details**

- **count**: The total number of faults.
- **nnnn**: The error number (the 4 digit “E” number as displayed on the printer message area)
- **clearable**: Indicates if the fault is clearable
  - 0. Not Clearable
  - 1. Clearable
- **title**: The textual title of the fault as displayed on the printer message area.

Note that the repeating part of the return string (fault number, clearable and title) is of fixed format and cannot be changed via the protocol definition file, although the separator ‘|’ may be changed as required.

**Examples**

```plaintext
»GFT<CR>
«FLT|3|5308|0|Printhead 1 - Printhead Disconnected|5307|0|Printhead 1 - No Cartridge|1005|0|Print Limit Exceeded|<CR>
```

Indicates there are 3 faults at the printer all of which are not as yet clearable.
The Get All Warnings (GWN)

This command lists all current warnings in the printer.

Syntax

GWN<CR>

Return Values

On success, returns the total count of the number of warnings followed by a list of current warnings. On failure, returns the default failure response.

WRN|<count>|[<nnnn>|<clearable>|<title>]|<CR>

Part within square brackets repeats for each warning present.

Details

- **count**: The total number of warnings.
- **nnnn**: The error number (the 4 digit “E” number as displayed on the printer message area)
- **clearable**: Indicates if the warning is clearable
  - 0. Not Clearable
  - 1. Clearable
- **title**: The textual title of the warning as displayed on the printer message area.

Note that the repeating part of the return string (fault number, clearable and title) is of fixed format and cannot be changed via the protocol definition file, although the separator ‘|’ may be changed as required.

Examples

```plaintext
GWN<CR>
WRN|0|<CR>
```

Indicates that there are no warnings at the printer.

The Clear All Faults Command (CAF)

This command attempts to clear all fault conditions present in the printer.

Syntax

CAF<CR>

Return Values

On success, returns the default success response. On failure, returns the default failure response. If the command succeeds, the response is sent after all faults have been cleared.

Details
The CAF command attempts to clear all faults present in the printer. Each fault will only be cleared if it is user-clearable (i.e. if it could be cleared using the user interface). The most likely reason for this command failing is the presence of a non-user-clearable fault. If there are no faults present, the success response is immediately returned.

Examples

`CAF<CR>`
`ACK<CR>`

All faults successfully cleared

**The Clear All Warnings Command (CAW)**

This command attempts to clear all warning conditions present in the printer.

**Syntax**

`CAW<CR>`

**Return Values**

On success, returns the default success response. On failure, returns the default failure response. If the command succeeds, the response is sent after all warnings have been cleared.

**Details**

The CAW command attempts to clear all warnings present in the printer. Each warning will only be cleared if it is user-clearable (i.e. if it could be cleared using the user interface). The most likely reason for this command failing is the presence of a non-user-clearable warning. If there are no warnings present, the success response is immediately returned.

Examples

`CAW<CR>`
`ACK<CR>`

All warnings successfully cleared

**The Clear Single Error (CEM)**

This command attempts to clear a single numbered error present in the printer.

**Syntax**

`CEM|<number>|<CR>`

Where `<number>` is the 4 digit number of the error.

**Return Values**

On success, returns the default success response. On failure, returns the default failure response.

**Details**

The CEM command attempts to clear a specific error (fault or warning) present in the printer. If an error with the given number is not pre-set then it returns the default failure response. The error will only be cleared if it is user-clearable (i.e. if it could be cleared using the user interface). The most likely reason for this command failing is the presence of a non-user-clearable error or an incorrect error number. If there are no errors present, the success response is immediately returned.
Examples

»CEM|1005|<CR>
«ACK<CR>

The Print Limit Exceeded fault is successfully cleared.

The Get Counts Command (GPC)

This command gets a snapshot of various performance counters from the printer

Syntax

GPC<CR>

Return Values

PCS|<success prints>|<fail prints>|<missed prints>|<remaining prints>|<CR>
Returns immediately.

Details

The GPC request retrieves the following values:

- **success prints**: The number of successful prints since start-up
- **fail prints**: The number of failed prints since start-up
- **missed prints**: The number of missed print signals since start-up
- **remaining prints**: The number of prints left in the current allocation

The Set Counts Command (SPC)

This command sets the values of various performance counters on the printer

Syntax

SPC|success prints|fail prints|missed prints|<CR>

Return Values

PCS|<success prints>|<fail prints>|<missed prints>|<remaining prints>|<CR>
Returns immediately.

Details

The SPC request sets the following values:

- **success prints**: The number of successful prints since start-up
- **fail prints**: The number of failed prints since start-up
- **missed prints**: The number of missed print signals since start-up
The Clear Queue Item Command (CQI)

This command removes items from the job queue that have not yet become active

Syntax

CQI|<item number>|<CR>

Return Values

On success, returns the default success response. On failure, returns the default failure response.

Details

The CQI request removes the numbered item from the job queue. The item number to use is that returned by the command that added the job to the queue.

If the command is issued with no queue item number all items are removed from the queue.

It is possible that a queue item existing at the time when the command is issued will have reached selection or been removed from the queue by other means before the CQI command is processed causing it to fail. This is expected behaviour.

Examples

»CQI<CR>
«ACK<CR>
All queue items successfully cleared

»CQI|3|<CR>
«ACK<CR>
Item with id 3 (Not necessarily the third item) successfully removed from the queue.

The Get Time and Date Command (GTD)

Syntax

GTD<CR>

Return Values

On success, returns the date and time of the printer in the following format:

TAD|DD/MM/YYYY HH:MM:SS|<CR>

On failure, returns the default failure response.

Details

The GTD command gets the current date and time of the printer.

Examples

»GTD<CR>
«TAD|09/05/2011 10:00:00|<CR>

The Set Date and Time Command (TAD)

Syntax

TAD|DD/MM/YYYY HH:MM:SS|<CR>
Return Values

On success, sets the printer time and date and returns the default success response. On failure, sends the default failure response.

Details

The TAD command sets the time and date on the printer to the time specified. If the time is sent in any other format the command will return the default failure response.

Examples

```plaintext
»TAD|09/05/2011 10:00:00|<CR>
«ACK<CR>

»TAD|09/05/2011|<CR>
«ERR<CR>

»TAD|09/05/2011 10:00|<CR>
«ERR<CR>
```

The Get Job Name Command (GJN)

Syntax

GJN<CR>

Return Values

On success, returns the name of the current job in the printer in the following format :-

`JOB|<job name>|<line number>|<CR>`

Line number is set to a dash “-“ if the printer is not currently in line select mode.

On failure, returns the default failure response.

Details

The GJN command gets the name of the job current set on the printer and the selected line if in line select mode.

Examples

```plaintext
»GJN<CR>
«JOB|Counter_Test|-|<CR>

Indicates that the current job is “Counter_Test”

In line select mode

»GJN<CR>
«JOB|Counter_Test|2|<CR>

Indicates that the job called “Counter_Test” has been loaded from line 2.

The Queue Size Command (QSZ)

Syntax

QSZ<CR>
Return Values

On success, returns the size of the update queue at the printer and the queue status in the following format:-

QSZ | <nn> | <s> | <CR>

Where

<nn> is the current queue size
<s> is the queue status

0. queue empty
1. queue low
2. queue high
3. queue full
4. queue state intermediate

On failure, returns the default failure response.

Details

The QSZ command returns the size of the update queue at the printer and the queue status. A queue status of high is only returned if the queue sizes rises above the pre-set high threshold, it is not returned when the queue size is falling. Similarly, a queue status of low is only returned if the queue size is falling. The queue state intermediate indicates that the queue is neither empty, full or at the pre-set low or high thresholds.

Examples

»QSZ<CR>
«QSZ|0|0|<CR>
Indicates that the queue is empty

»QSZ<CR>
«QSZ|15|3|<CR>
Indicates that the queue is full

»QSZ<CR>
«QSZ|2|1|<CR>
Indicates that the queue size is 2 and has triggered a queue low indication.

»QSZ<CR>
«QSZ|7|4|<CR>
Indicates that the queue size is 7 but this is not low enough to have triggered a queue low indication or high enough to have triggered a queue high indication.

The Get Job List Command (GJL)

Syntax

GJL<CR>
Return Values

On success, returns the total count of the number of Jobs followed by a list of the names of the Jobs in the printer in the following format :-

\[ JBL|<\text{count}>|[[<\text{jobname}>]|]<\text{CR}> \]

Part within square brackets repeats for each job present.

On failure, returns the default failure response.

Details

\textbf{count} \\
\hspace{1cm} The total number of jobs.

\textbf{jobname} \\
\hspace{1cm} The name of the job.

Note that the repeating part of the return string is of fixed format and cannot be changed via the protocol definition file, although the separator ‘|’ may be changed as required.

Examples

\[ GJL<\text{CR}> \]
\[ JBL|3|\text{ApplesJob}|\text{OrangesJob}|\text{BananasJob}|<\text{CR}> \]

Indicates that there are 3 jobs in the database called ApplesJob, OrangesJob and BananasJob.

\section*{The Get Job Fields Command (GJF)}

Syntax

\[ \text{GJF}|<\text{jobname}|<\text{CR}> \]

Where \textit{<jobname>} is the name of a Job at the printer (obtained from a GJL command for example)

Return Values

On success, returns the total count of the number of fields in the Job followed by a list of field names in the following format :-

\[ JFL|<\text{count}>|[[<\text{fieldname}>]|]<\text{CR}> \]

Part within square brackets repeats for each field present. The list of fields are not returned in any particular order.

On failure, returns the default failure response.

Details

\textbf{count} \\
\hspace{1cm} The total number of fields in the job.

\textbf{fieldname} \\
\hspace{1cm} The name of the job field.

Note that the repeating part of the return string is of fixed format and cannot be changed via the protocol definition file, although the separator ‘|’ may be changed as required.
Examples

»GJF|OrangesJob|<CR>
«JFL|2|Field00|PriceField01|<CR>

Indicates that there are 2 fields in the job called “OrangesJob” which are called “Field00” and “PriceField01”.

The Get Current Job Data Command (GJD)

Syntax

GJD<CR>

Return Values

On success, returns the total count of the number of fields in the currently loaded Job followed by a list of field name and value pairs in the following format :-

JDL|<count>|[[<fieldname>=<value>|]<CR>

Part within square brackets repeats for each field present.

On failure, returns the default failure response.

Details

count  The total number of fields in the job.

fieldname  The name of the job field.

value  The value of the job field.

The values will be those supplied when the user selected the job (SEL, SLA, SLI command) but will be unaffected by changes to the printing fields made by JDU, JDA or JDI commands. Also real time incrementing fields such as counters and elapsed times will remain as entered at job select.

Note that the repeating part of the return string (fieldname, assigner symbol, value) is of fixed format and cannot be changed via the protocol definition file, although the separator ‘|’ and assigner “=” may be changed as required.

Examples

»GJD<CR>
«JDL|2|Field00=Apples|PriceField01=$2.00|<CR>

Indicates that there are 2 fields in the current job, “Field00” which is text field having a value of “Apples” and “PriceField01” which has a value of “$2.00”.

The Device Specific Commands and Requests (CMD)

This set of commands allows control of certain printer functions.

Syntax

CMD|<devcommand>|<CR>
Return Values

The CMD commands all return the default success response if the syntax is correct and the printer supports the command. The default failure parameter is returned if the syntax is invalid or the printer does not support the command.

Details

Send a CMD command. The command is specified in the devcommand parameter; valid values for this parameter for the videojet 8610 printer are:

- **COP**: Causes printers which implement this command to open the cap covering the print head.
- **CCL**: Causes printers which implement this command to close the cap covering the print head.
- **BWE**: Causes printers which implement this command to enable warming of the print head.

Examples

```plaintext
»CMD|COP|<CR>
«ACK<CR>
The caps are opened

»CMD|CCL|<CR>
«ACK<CR>
The caps are closed

»CMD|XXX|<CR>
«ERR<CR>
The CMD sub command XXX is not defined.
```

The Set Print Density command (SPD)

This is only currently available for Videojet 2300 series printer.

Set the “Printer Density” parameter to a specified value.

Syntax

```
SPD|<d>|<CR>
```

or

```
SPD|<d>|<ms>|<CR>
```

where `<d>` is a numeric value between 0 and 100% and is the required print density

where `<ms>` is the Master and Slave selector with a value from 1 to 15 as follows:

- **1**: Update the printer (or the Master only in a Master/Slave configuration)
- **3**: Update the Master and the first Slave (if enabled)
- **7**: Update the Master and the first 2 Slaves (if enabled)
- **15**: Update the Master and the all 3 Slaves (if enabled)

<ms> values other than 1 can only be sent to a Master printer. If <ms> is omitted then it is equivalent to an <ms> value of 1 i.e. just the local or master printer.

Other numerical values of `<ms>` can be used where the Master or Slaves affected is determined by the binary equivalent of the number e.g. 9 will update the Master and 3rd Slave only.

For a non master slave configuration the second parameter must be 1 or omitted.
The command will set the Print Density to the nearest supported value on the master and slaves selected. For the 2300 series the supported values are 100%, 70%, 50%, 33% and 25%.

The command will return an error if the values are out of range or an update is requested on a slave printer which is either not currently configured or connected.

Return Values
On success, returns the default success response. On failure, returns the default failure response.

Examples

```
»SPD|33|<CR>
«ACK<CR>
```
Sets the print density to 33% on the printer.

```
»SPD|75|1|<CR>
«ACK<CR>
```
Sets the print density to 70% on the printer.

```
»SPD|70|15|<CR>
«ACK<CR>
```
Sets the print density to 70% on the Master and all 3 Slave printers.

```
»SPD|150|1|<CR>
«ERR<CR>
```
Fail as the density value is out of range.

```
»SPD|100|20|<CR>
«ERR<CR>
```
Fail as the master/slave selector value is out of range.

```
»SPD|100|3|<CR>
«ERR<CR>
```
Fail because the printer is either not in master-slave configuration or the slave is not configured or disconnected.

The Get Print Density command (GPD)

This is only available for Videojet 2300 series printer.

Get the “Printer Density” value from the printer.

Syntax

```
GDP<CR>
```

or

```
GDP|<p>|<CR>
```

where <p> is a numeric value between 0 and 3 as follows

- 0 Read the printer value (or the Master value only in a Master/Slave configuration)
- 1 Read the value at the 1st Slave (if enabled)
2  Read the value at the 2\textsuperscript{nd} Slave (if enabled)
3  Read the value at the 3\textsuperscript{rd} Slave (if enabled)

If \texttt{<p>} is not present then the value at the local printer is returned.

**Return Values**

On success, returns

\texttt{DEN|\langle d\rangle|\langle CR\rangle}

Where

\(<d>\) is a numeric value between 0 and 100\% and is the current print density. Or -1 if a Slave printer is not currently connected.

On failure, returns the default failure response

**Examples**

\texttt{»GDP<CR>}
\texttt{«DEN|50|<CR>}

Indicates that the current Print Density for the Master printer is is 50\%.

\texttt{»GDP|2|<CR>}
\texttt{«DEN|100|<CR>}

Indicates that the current print density at the first Slave is 100\%.

\texttt{»GDP|2|<CR>}
\texttt{«ERR<CR>}

Indicates that the second Slave is not enabled.

\texttt{»GDP|2|<CR>}
\texttt{«DEN|-1|<CR>}

Indicates that the second Slave is disconnected.

**The Serialisation Header and Data command (SHD)**

This commands sends a single record with both field names and data.

**Syntax**

\texttt{SHD|\langle fieldname\rangle=\langle data\rangle|...|<CR>}

Where

\(<\texttt{fieldname}>\) is the name of the field to be updated.
\(<\texttt{data}>\) is the data that will be used to update the \(<\texttt{fieldname}>\) field.

**Return Values**

On success, returns the default success response. On failure, returns the default failure response

**Example**

\texttt{»SHD|fieldName1=myField1|fieldName2=myField2|fieldName3=myField3|<CR>}
\texttt{«ACK<CR>
The Serialisation Header Only command (SHO)

This command sends field names only.

Syntax

```
SHO|<fieldname>|...|<CR>
```

Return Values

On success, returns the default success response. On failure, returns the default failure response.

**Details**

`fieldname`

The name of the job field.

**Example**

```
»SHO|myField1|myField2|myField3|<CR>
«ACK<CR>
```

The Serialisation Data only command (SDO)

This command sends field data only.

Syntax

```
SDO|<data>|...|<CR>
```

Return Values

On success, returns

```
SFS|<s>|<CR>
```

Where

`<s>` is the number of available bytes in the serialisation buffer.

On failure, returns the default failure response.

Details

`data`

The field data supplied in the order of field names supplied in SHO command.

**Example**

```
»SDO|myData1|myData2|myData3|<CR>
«SFS|951744|<CR>
```

The Serialisation Change Field Data (SCF)

This command is used to update non-serialisation fields when in serialisation mode.
Syntax

SCF|<fieldname>=<data>|…|<CR>

Where
- `<fieldname>` is the name of the non-serialisation field to be updated.
- `<data>` is the data that will be used to update the `<fieldname>` field.

Return Values

On success, returns the default success response. On failure, returns the default failure response.

Example

```plaintext
»SCF|fieldName1=myField1|fieldName2=myField2|fieldName3=myField3|<CR>
«ACK<CR>
```

The Serialisation Record Count command (SRC)

This command gets the number of records currently in the serialisation buffer.

Syntax

SRC<CR>

Return Values

On success, returns

```plaintext
SRC|<c>|<CR>
```

Where
- `<c>` is the number of records in the serialisation buffer.

On failure, returns the default failure response.

Example

```plaintext
»SRC<CR>
«SRC|25|<CR>
```

The Serialisation Clear Buffer command (SCB)

This command clears the serialisation buffer.

Syntax

SCB<CR>

Return Values

On success, returns the default success response. On failure, returns the default failure response.

Example

```plaintext
»SCB<CR>
```
The Serialisation Indexed Data command (SID)

This command adds another record with the supplied data and the supplied ID/serial number.

Syntax

SID|<recordindex>|<data>|...|<CR>

Where

<recordindex> is the ID or serial number of the record.
<data> is the field data to be used for each field.

Return Values

On success, returns the default success response. On failure, returns the default failure response.

Example

»SID|6|myData1|myData2|myData3|<CR>
«ACK<CR>

The Serialisation Free Space command (SFS)

This command gets the amount of free space in the serialisation buffer.

Syntax

SFS<CR>

Return Values

On success, returns

SFS|<s>|<CR>

Where

<s> is the number of available bytes in the serialisation buffer.

On failure, returns the default failure response

Example

»SFS<CR>
«SFS|952368|<CR>

The Serialisation Next Record Index command (SNI)

This command gets the next index to be printed.

Syntax

SNI<CR>
Return Values

On success, returns

\[ SNI|<i>|<CR> \]

Where

\[ <i> \] is the next index to be printed.

On failure, returns the default failure response

Example

\[ »SNI<CR> \]
\[ «SNI|7|<CR> \]

**The Serialisation Last Record Index command (SLR)**

This command gets the last index printed.

**Syntax**

\[ SLR<CR> \]

**Return Values**

On success, returns

\[ SLR|<i>|<CR> \]

Where

\[ <i> \] is the last index printed.

On failure, returns the default failure response

Example

\[ »SLR<CR> \]
\[ «SLR|7|<CR> \]

**The Serialisation Set Maximum Records command (SMR)**

This command sets the maximum number of records allowed in the serialisation buffer.

**Syntax**

\[ SMR|<r>|<CR> \]

Where

\[ <r> \] is the maximum number of records allowed in the serialisation buffer.

**Return Values**

On success, returns the default success response. On failure, returns the default failure response.

Example

\[ »SMR|120|<CR> \]
\[ «ACK<CR> \]
The Serialisation Get Maximum Records command (SGM)

This command gets the maximum number of records allowed in the serialisation buffer.

Syntax

\[ SGM<CR> \]

Return Values

On success, returns

\[ SGM|<r>|<CR> \]

Where

\(<r> \) is the maximum number of records allowed in the serialisation buffer.

On failure, returns the default failure response

Example

\>
\<SGM<CR>
\<ACK<CR>\>

The Set Encoder Direction command (SED)

This command sets the encoder direction. Thermal Inkjet printers only (8510, 8610).

Syntax

\[ SED|<d>|<CR> \]

Where

\(<d> \) is the encoder direction:

- N - Normal direction
- R - Reverse direction

Return Values

On success, returns the default success response. On failure, returns the default failure response.

Example

\>
\<SED|N|<CR>
\<ACK<CR>\>

\>
\<SED|R|<CR>
\<ACK<CR>\>

The Get Encoder Direction command (GED)

This command gets the encoder direction. Thermal Inkjet printers only (8510, 8610).

Syntax

\[ GED<CR> \]
Return Values

On success, returns

GED|<d>|<CR>

Where

<d> is the encoder direction:
N - Normal direction
R - Reverse direction

On failure, returns the default failure response or -1 if SED/GED are not supported on this device.

Example

»GED<CR>
«GED|N|<CR>

»GED<CR>
«GED|R|<CR>

The Get Consumable Levels command (GCL)

Syntax

GCL<CR>

Or

GCL|<id>|<CR>

Where

<id> is the slave id of the printer you wish to view the consumable levels of, for example:
0 gets the consumable levels of the master printer.
1 gets the consumable levels of the first slave printer.
2 gets the consumable levels of the second slave printer.
3 gets the consumable levels of the third slave printer.

Return Values

This command will return the values of the consumables currently in use for the printer specified.

GCL|<c>...|<CR>

Where

<c> is the consumable level

Details

This command will return all of the consumable levels that are in use for the connected printer, for example, an 8510 printer might return 1, 2, 3 or 4 different consumable levels depending on how many print heads are connected.

Example

»GCL<CR>
«GCL|20|50|<CR>
The PackML Commands and Requests (PML)

This set of commands and requests allow the printer to be controlled according to the PackML Automatic Mode Machine States Definition.

Syntax

PML|<packmlcommand>|<CR>

Return Values

The PackML commands all return the default success response on success and the default failure response if the command was rejected. The return values for the PackML requests are described below.

Details

Sends a PackML command or request. The command or request is specified in the `packmlcommand` parameter; valid values for this parameter are:

**PRE**

Sends a PackML Prepare command, which represents the PackML state transition from STOPPED to STARTING. In CLARiTY terms it is equivalent to pressing the button while the system is in the Shut Down state. The success response is sent when the system state reaches the target state. The failure response is sent if the state transition is rejected. If the system is already in the target state when the command is received, a success response is immediately sent without performing any action. If the system is in a state other than the source or target states for this transition, the command is immediately rejected.

**STA**

Sends a PackML Start command, which represents the PackML state transition from READY/HELD to STANDBY. In CLARiTY terms it is equivalent to pressing the button while the system is in the Offline state. The success response is sent when the system state reaches the target state. The failure response is sent if the state transition is rejected. If the system is already in the target state when the command is received, a success response is immediately sent without performing any action. If the system is in a state other than the source or target states for this transition, the command is immediately rejected. This command may fail if there is no job selected.

**STP**

Sends a PackML Stop command, which represents the PackML state transition from STARTING/HELD/READY/STANDBY/PRODUCING to STOPPING or ABORTED to STOPPED. In CLARiTY terms it is equivalent to clearing any faults and then pressing the button while the system is in the Offline, Starting up, or Running states, or equivalent to clearing any faults while the system is in the Shut Down state and a fault has occurred. The success response is sent when the system state reaches the target state. The failure response is sent if the state transition is rejected. If the system is already in the target state when the command is received, a success response is immediately sent without performing any action. If the system is in a state other than the source or target states for this transition, the command is immediately rejected. This command may fail if there are faults present that cannot be cleared.
**HLD**

Sends a PackML Hold command, which represents the PackML state transition from **STANDBY/PRODUCING** to **HOLDING**. In CLARiTY terms it is equivalent to pressing the button while the system is in the **Running** state. The success response is sent when the system state reaches the target state. The failure response is sent if the state transition is rejected. If the system is already in the target state when the command is received, a success response is immediately sent without performing any action. If the system is in a state other than the source or target states for this transition, the command is immediately rejected.

**ABT**

Sends a PackML Abort command, which represents the PackML state transition from **STARTING/HELD/READY/STANDBY/PRODUCING** to **ABORTING**. This command causes an **Abort** fault to occur, thus setting the error state to Fault Present. The success response is sent when the system state reaches the target state. The failure response is sent if the state transition is rejected. If the system is already in the target state when the command is received, a success response is immediately sent without performing any action. If the system is in a state other than the source or target states for this transition, the command is immediately rejected.

**GST**

Gets the PackML State as a decimal integer, with the following possible values:

- 0: **STOPPED**
- 1: **STARTING**
- 2: **READY**
- 3: **STANDBY**
- 4: **PRODUCING**
- 5: **STOPPING**
- 6: **ABORTING**
- 7: **ABORTED**
- 8: **HOLDING**
- 9: **HELD**

The command response string is:

```text
<packmlstate><CR>
```

**GSX**

Gets the PackML State as a string, with the following possible values:

- STOPPED
- STARTING
- READY
- STANDBY
- PRODUCING
- STOPPING
- ABORTING
- ABORTED
- HOLDING
- HELD

The command response string is:

```text
<packmlstatestring><CR>
```

**Examples**

```text
PML|GST|<CR>
9<CR>
The printer is in the HELD state.

PML|GSX|<CR>
HELD<CR>
The printer is in the HELD state.

PML|ABT|<CR>
ACK<CR>
```
The PackML Abort command was successfully executed.

»PML|GSX|<CR>
«ABORTED<CR>

Confirmation that the printer is now in the ABORTED state.

»PML|PRE|<CR>
«ERR<CR>

The PackML Prepare command was unsuccessful because the printer is in the ABORTED state.

»PML|STP|<CR>
«ACK<CR>

The PackML Stop command successfully cleared the Abort error and moved the printer to the STOPPED state.

»PML|PRE|<CR>
«ACK<CR>

The PackML Prepare command successfully moved the printer to the READY state.

»PML|STA|<CR>
«ACK<CR>

The PackML Start command successfully moved the printer to the STANDBY state.

»PML|HLD|<CR>
«ACK<CR>

The PackML Hold command successfully moved the printer to the HOLDING state.

»PML|GSX|<CR>
«HELD<CR>

The printer is in the HELD state.
Async Notifications

The Async Text Comms is can be provided on the same port as the existing Text Comms.

After upgrade to Version 6 of Text Comms or initial delivery of a printer all notifications are completely disabled and all parameter changes will be ignored.

The notifications can be enabled or disabled from Clarity config via the TextCommsAsyncEnables parameter. This parameter sets the default for the notifications. Upon restarting the printer or starting a new Text Comms session this default will be reloaded.

Various Text Commands are provided to temporarily change the enable or disable state of each of the notifications.

Each Asynchronous message will be separated by a separator character as determined in the protocol XML (usually a carriage return i.e. 0x0D)

The following values can be enabled for notification via the Async Text Comms

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS</td>
<td>Async send of current overall status</td>
</tr>
<tr>
<td>ERS</td>
<td>Async send of current error status</td>
</tr>
<tr>
<td>JOB</td>
<td>Async send of current job details</td>
</tr>
<tr>
<td>PRS</td>
<td>Async send of print start</td>
</tr>
<tr>
<td>PRC</td>
<td>Async send of print complete</td>
</tr>
</tbody>
</table>

The Overall State change Notification (STS)

This message indicates a change in the overall state of the printer as seen on the home screen.

Syntax

STS|<n>|<CR>

Details

The values of the states (<n>) are as defined in the SST (State Set Command) and GST (Get State Request) text commands. i.e.

0. Shutdown  
1. Starting Up  
2. Shutting Down  
3. Running  
4. Offline

The message will contain the value of the state i.e. 0 to 4.

Examples

«STS|0|<CR>  
the printer has changed state to idle.

The Print Start Notification (PRS)

This message indicates that the current print has commenced and that the next print can be loaded into the print queue of the printer.

Syntax
The Print complete Notification (PRC)

This message indicates that the current print is complete and that the next print can be sent to be printed at the printer.

Syntax

PRC<CR>

The I/O Output changes Notification (OUT)

A message is generated each time an output event occurs. A message will only be generated if enabled individually by the text commands given below. These are a sub-set of those which can be specified as a trigger to turn a physical output on or off, i.e.

0. Update Queue Empty (OutEvtUpdateQueueEmpty)
1. Update Queue Full (OutEvtUpdateQueueFull)
2. Update Queue High (OutEvtQueueHigh)
3. Update Queue Low (OutEvtQueueLow)

A queue status of high is only returned if the queue sizes rises above the pre-set high threshold, it is not returned when the queue size is falling. Similarly, a queue status of low is only returned if the queue size is falling.

Syntax

OUT|QEMPTY|<CR>
OUT|QFULL|<CR>
OUT|QHIGH|<CR>
OUT|QLOW|<CR>

The Error state change Notification (ERS)

This message indicates a change in the overall error state of the printer.

Syntax

ERS|<n>|<CR>

Details

The values of the states (\(<n>\)) are as defined in the existing GST (Get State Request) text commands. These are as given below.

0. No errors
1. Warnings present
2. Faults present

The message will contain the value of the state i.e. 0 to 2.

Examples

«ERS|2|<CR>

the printer has changed state to fault present.
The Current Job changed Notification (JOB)

This message indicates that a new job has been loaded into the printer.

Syntax

JOB|<name>|<line>|<CR>

Details

The name of the new job is sent (<name>) and if in line select mode, the name of the job and the new line number is sent (<line>).

Examples

«JOB|Cartridge Test Image|-|<CR>

A new job was loaded called “Cartridge Test Image”, while not in line select mode.

«JOB|Counter Job|2|<CR>

A new job was loaded called “Counter Job” for line select 2.

The Set Async Notification command (SAN)

The command enables or disables a set of notifications.

Syntax

SAN|<bbbbbbbbbb>|<CR>

Return Values

On success, returns the default success response. On failure, returns the default failure response.

Details

It takes a binary value to define a set of binary flags. Each bit is defined as follows:

0. Overall State change
1. Print Start (Message start)
2. Print complete (Message complete)
3. I/O Output changes
4. Error state change
5. Current Job changed
6. I/O Output change - Update Queue Empty
7. I/O Output change - Update Queue Full
8. I/O Output change - Update Queue High
9. I/O Output change - Update Queue Low

If Bit 3 is not present then all I/O Output change notifications are disabled, otherwise Bits 7 to 10 allow individual outputs to be enabled or disabled.

The default value for these flags will be stored as a parameter in the printer which will be remembered across a comms link restart and printer reboot. This parameter can be edited from CLARiTY Config.

Note the change is not saved across a comms link restart and printer reboot, the value will revert to that from the value set in CLARiTY Config.
Examples

`SAN|10101|<CR>
<ACK|<CR>`

enables reporting of error state, print complete and overall state.

**The Get Async Notification command (GAN)**

Returns a bit map of binary flags indicating currently enabled notifications.

**Syntax**

GAN<CR>

**Return Values**

On success, returns binary value to define a set of binary flags. Each bit is defined as for the SAN command.
On failure, returns the default failure response.

**Examples**

`GAN<CR>
SAN|000010001|<CR>`

Indicates that reporting of only error state and overall state is enabled.

**The Enable All Notifications commands (EAN)**

Switches on the Async comms mode and prevents the sending of all notifications.
Equivalent to SAN|111111111|

**Syntax**

EAN<CR>

**Return Values**

On success, returns the default success response. On failure, returns the default failure response.

**Examples**

`EAN<CR>
<ACK<CR>`

**The Disable All Notifications command (DAN)**

Switches off all Async comms messages and prevents the sending of all notifications.
Equivalent to SAN|0|

**Syntax**

DAN<CR>

**Return Values**
On success, returns the default success response. On failure, returns the default failure response.

Examples

`DAN<CR>`
`ACK<CR>`

**The Disable Print Notifications command (DPN)**

Switches off the print start and print complete Async comms messages and enables all other notifications.
Equivalent to `SAN|1111111001` |

**Syntax**

`DPN<CR>`

**Return Values**

On success, returns the default success response. On failure, returns the default failure response.

Examples

`DPN<CR>`
`ACK<CR>`

**The Set Single Notification command (SNO)**

The command enables or disables a single class of notification.

**Syntax**

```
SNO|<name>|<b>|<CR>  or
SNO|<nn>|<b>|<CR>
```

**Return Values**

On success, returns the default success response. On failure, returns the default failure response.

**Details**

The set single notification command takes either a number `<nn>` of the bit as given for the `SAN` command or a name `<name>` corresponding to one of the bits as below

- **STS** - Overall State change
- **PRS** - Print Start (Message start)
- **PRC** - Print complete (Message complete)
- **OUT** - I/O Output changes
- **ERS** - Error state change
- **JOB** - Current Job changed
- **QEM** - I/O Output change - Update Queue Empty
- **QFU** - I/O Output change - Update Queue Full
- **QHI** - I/O Output change - Update Queue High
- **QLO** - I/O Output change - Update Queue Low

It also takes a second parameter `<b>` to indicate the required new value.

- 0. Disabled
- 1. Enabled
Examples

»SNO|4|1|<CR>
«ACK<CR>

enables reporting of error status changes

»SNO|ERS|0|<CR>
«ACK<CR>

disables reporting of error status changes

»SNO|OUT|1|<CR>
»SNO|QEM|1|<CR>
«ACK<CR>

enables reporting of Update Queue Empty.

If the value is changed or if the new value is the same as the current value then acknowledge (ACK) is returned otherwise an error (ERR) is returned.

Note the change is not saved across a comms link restart and printer reboot, the value will revert to that from the value set in CLARiTY Config (see below)
Example Code

The following code is a simple Visual Basic program to send a series of commands to the printer via COM1. The code assumes that there is a job in the printer’s database called “Default 4 Line Text” and that it has variable text fields named “Line 1”, “Line 2”, “Line 3” and “Line 4”.

```vbscript
Private Sub ZIPHERTextCommsTest()
    ' Use the MSComm object to interface with the serial port
    Dim serialport As Object
    Set serialport = CreateObject("MSCOMMLib.MSComm")
    ' Use COM1.
    serialport.comport = 1
    ' 115200 baud, no parity, 8 data, and 1 stop bit.
    serialport.Settings = "115200,N,8,1"
    ' Read the entire buffer
    serialport.InputLen = 0
    ' Open port
    serialport.PortOpen = True
    ' Send the commands
    SendZIPHERCommands serialport
    ' Close the serial port.
    serialport.PortOpen = False
End Sub

Private Sub SendZIPHERCommands(serialport As Object)
    Dim successResponse As String
    successResponse = "ACK" & Chr$(13)
    ' If the printer is shut down, start it up
    Dim state As Integer
    state = GetState(serialport)
    If state = 0 Then
        serialport.Output = "SST|1|" & Chr$(13)
        If WaitForResponse(serialport) <> successResponse Then Exit Sub
    End If
    ' Wait for it to get into a useful state
    Do Until state = 4 Or state = 3
        Sleep 1
        state = GetState(serialport)
    Loop
    ' Select the "Default 4 Line Text" job and set the Line 1,2,3,4 fields to empty
    serialport.Output = "SLA|Default 4 Line Text|Line 1=|Line 2=|Line 3=|Line 4=|" & Chr$(13)
    If WaitForResponse(serialport) <> successResponse Then Exit Sub
    ' Set the printer to Running if necessary
    If GetState(serialport) <> 3 Then
        serialport.Output = "SST|3|" & Chr$(13)
        If WaitForResponse(serialport) <> successResponse Then Exit Sub
    End If
    ' Wait for 1 second so that the printer can get the printhead into printing position
    Sleep 1
    ' Take the printer offline
    serialport.Output = "SST|4|" & Chr$(13)
    If WaitForResponse(serialport) <> successResponse Then Exit Sub
Next

Private Sub ZIPHERTextCommsTest()
    ' Use the MSComm object to interface with the serial port
    Dim serialport As Object
    Set serialport = CreateObject("MSCOMMLib.MSComm")
    ' Use COM1.
    serialport.comport = 1
    ' 115200 baud, no parity, 8 data, and 1 stop bit.
    serialport.Settings = "115200,N,8,1"
    ' Read the entire buffer
    serialport.InputLen = 0
    ' Open port
    serialport.PortOpen = True
    ' Send the commands
    SendZIPHERCommands serialport
    ' Close the serial port.
    serialport.PortOpen = False
End Sub
```

Private Sub SendZIPHERCommands(serialport As Object)
    Dim successResponse As String
    successResponse = "ACK" & Chr$(13)
    ' If the printer is shut down, start it up
    Dim state As Integer
    state = GetState(serialport)
    If state = 0 Then
        serialport.Output = "SST|1|" & Chr$(13)
        If WaitForResponse(serialport) <> successResponse Then Exit Sub
    End If
    ' Wait for it to get into a useful state
    Do Until state = 4 Or state = 3
        Sleep 1
        state = GetState(serialport)
    Loop
    ' Select the "Default 4 Line Text" job and set the Line 1,2,3,4 fields to empty
    serialport.Output = "SLA|Default 4 Line Text|Line 1=|Line 2=|Line 3=|Line 4=|" & Chr$(13)
    If WaitForResponse(serialport) <> successResponse Then Exit Sub
    ' Set the printer to Running if necessary
    If GetState(serialport) <> 3 Then
        serialport.Output = "SST|3|" & Chr$(13)
        If WaitForResponse(serialport) <> successResponse Then Exit Sub
    End If
    ' Wait for 1 second so that the printer can get the printhead into printing position
    Sleep 1
    ' Take the printer offline
    serialport.Output = "SST|4|" & Chr$(13)
    If WaitForResponse(serialport) <> successResponse Then Exit Sub
Next
```

Private Sub ZIPHERTextCommsTest()
    ' Use the MSComm object to interface with the serial port
    Dim serialport As Object
    Set serialport = CreateObject("MSCOMMLib.MSComm")
    ' Use COM1.
    serialport.comport = 1
    ' 115200 baud, no parity, 8 data, and 1 stop bit.
    serialport.Settings = "115200,N,8,1"
    ' Read the entire buffer
    serialport.InputLen = 0
    ' Open port
    serialport.PortOpen = True
    ' Send the commands
    SendZIPHERCommands serialport
    ' Close the serial port.
    serialport.PortOpen = False
End Sub
Private Function GetState(serialport As Object) As Integer
Dim statusString As String
Dim pos As Integer
serialport.Output = "GST" & Chr$(13)
statusString = WaitForResponse(serialport)
' Get to the character after the "STA|" part
statusString = Mid$(statusString, 5, 1)
GetState = CLng(statusString)
End Function

Private Function WaitForResponse(serialport As Object) As String
Dim outputbuffer As String
' Wait for data to come back to the serial port.
Do
    DoEvents
    outputbuffer = outputbuffer & serialport.Input
Loop Until InStr(outputbuffer, Chr$(13))
WaitForResponse = outputbuffer
End Function

Private Sub Sleep(seconds As Single)
    'I don't recommend this code, but I want to make everything simple to show the principles!
    start = Timer
    Do
        Loop Until Timer >= start + seconds
    End Sub
Appendix 1: State Transition Diagram

The following diagram shows the state transitions for CLARiTY:

Note that any state transition may be blocked by the printer if it would be unsafe or inappropriate in the current circumstances. For example, if no job is selected the printer cannot be placed in the Running state.
Appendix 2: CLARiTY Configuration manager Settings

In CLARiTY Configuration Manager (CCFG) there is a parameter ‘Job Selection Mode’ which has the options:

- Normal
- Selection Per Print

Normal is to be used as the default running mode. If you are not experiencing any issues running in this mode then this does not need to be changed.

Selection Per Print should be selected when SLI commands are being sent to the printer with an allocation of one, i.e. for each print there is a job selection. This mode switches off-loading and saving of job parameters.