Connecting UniOP to Datalogic Barcode Readers

This Technical Note contains the information needed to connect UniOP to Datalogic Barcode Scanners.

Contents

1. Introduction 1
2. Designer setup 1
2.1 Controller Setup 1
3. Define Field Dialog Box 2
4. Datalogic Data Types 3
5. Data Transfer 4
5.1 The Fast Data Transfer 5
6. Sending commands to Barcode Scanner 5
7. Advanced Programming 6
Appendix A. Communication Error Codes 8

1. Introduction

The Datalogic protocol is included in the Designer file UPLC174.DLL.

The Datalogic protocol will communicate with a Datalogic Barcode Scanner and will store received barcode strings into UniOP's internal memory. Stored strings can then be displayed and/or transferred to other controller(s).

2. Designer setup

To create a Designer project for a Datalogic Barcode Reader, select the driver "Datalogic" from the list of available communication drivers in the "Project\Change Controller Driver…" dialog box.

2.1 Controller Setup

Figure 1 below shows the "Controller Setup..." dialog box for the Datalogic protocol.

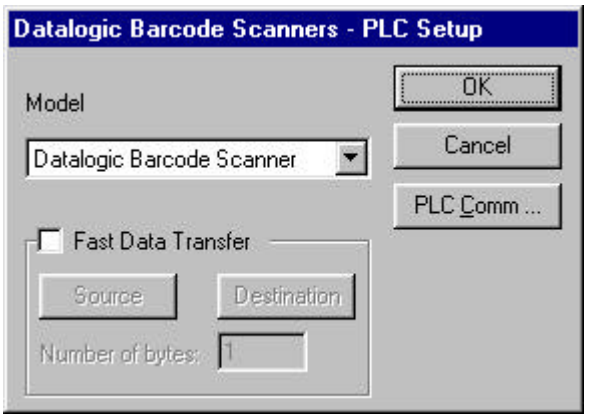

Figure 1 - Controller Setup dialog box

As the Datalogic is a very simple protocol, only communication parameters are to be configured. If needed, it is also possible to configure the Fast Data Transfer here.

Pressing the “PLC Comm...” button the dialog as shown below in Figure 2 will appear.

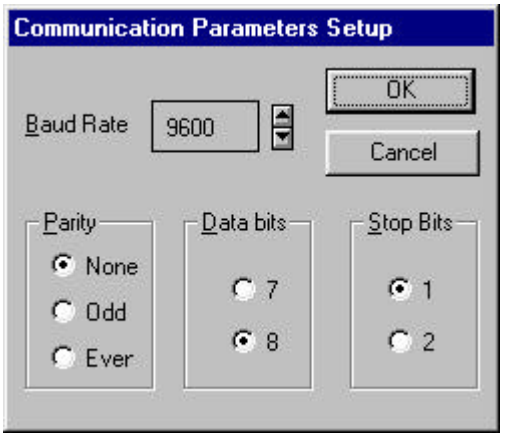

Figure 2 – Communication Parameters Setup dialog box

Default communication Parameters are: 9600,N,8,1. They should match the communication parameters programmed for the scanner.

3. Define Field Dialog Box

When defining a field the Data Field Properties dialog box will pop up. The dialog box is shown in Figure 3 below.

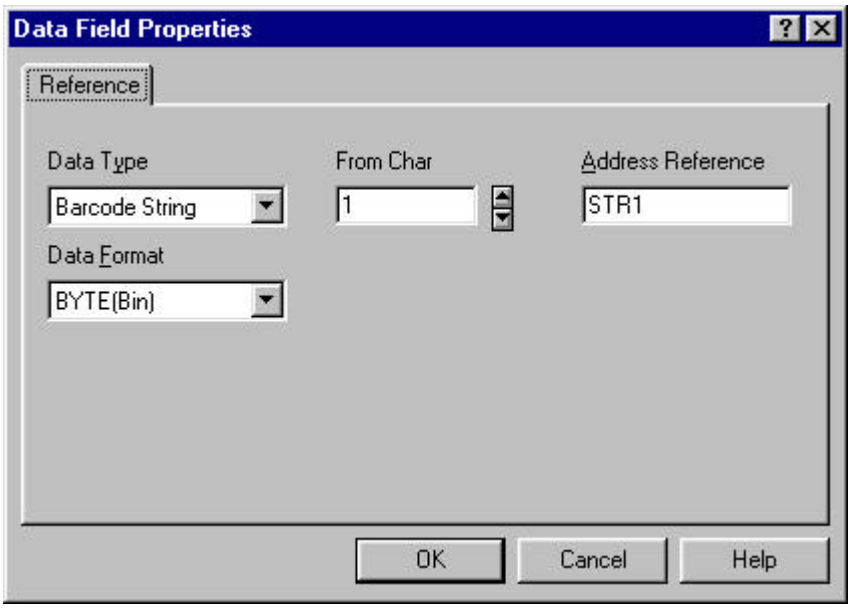

Figure 3 – The Define Field dialog box

For the Barcode String and the Memory Buffer data types, it is possible to select ASCII as display format.

4. Datalogic Data Types

The following data types can be accessed:

Data type	Description
Barcode String	Represents the barcode string received from the Datalogic Barcode Scanner.
String Length	Holds the length (in characters) of the received barcode string.
Error Code	Holds the error code of the string receiving communication session. 1 – no error, 2 – line error
String Identifier	Holds the unique number (byte) – in fact a modulo-265 counter – dedicated to the received barcode string. It can be useful for identification.
Memory Buffer	550 bytes of the Internal UniOP memory.

The Memory Buffer data type deserves a more detailed explanation. Most of it can freely be used for various purposes (Interlock Mailbox, Data Transfers, RDA...), but there are some areas dedicated to some special purposes. These regions are:

1. **The Storage Record** (at the very beginning of the Memory Buffer) – this is the place where the following information: Barcode String, String Length, Error Code and String Identifier, are actually placed. The Storage Record is described in the table below:

Offset	Data
0	String Identifier
1	Error Code
2	String Length

3	First character of the Barcode String
4	Second character of the Barcode String
...	...

The length of the Storage Record is not strictly defined. It depends on how long barcode strings are expected to be. If no strings longer than 20 bytes are expected, then the length of the region used for the Storage Record is for instance no longer than 23 bytes and thus first 23 bytes of the Memory Buffer shall not be used for other purposes.

2. **Offset range from 420 to 469**. These 50 bytes are used for the special mailbox (see the Data Transfers chapter of this tech-note) and for the advanced programming (see the Advanced Programming section of this tech-note).

5. Data Transfer

The use of the Data Transfer will be useful when the Datalogic communication driver is used in combination with a second protocol in a dual-driver architecture.
In order to transfer the received barcode strings and other data to the controller(s) that are using them it is required to program at least one data transfer job. To program it, choose the "Panel Controller Interface..." menu option in the Designer and then select the "Data Transfer" tab.

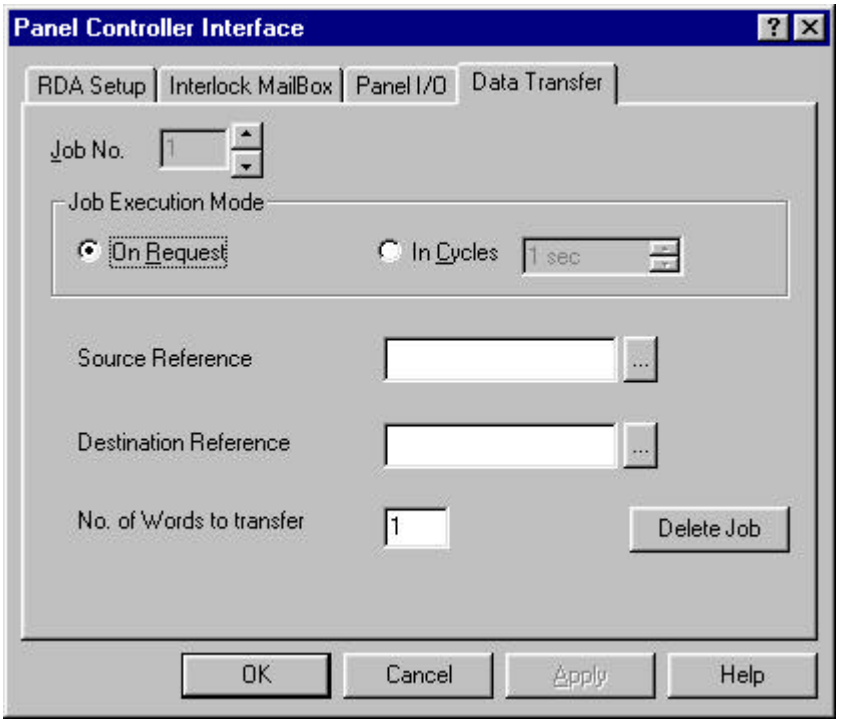

Figure 4 – The Data Transfer dialog box

Although it is possible to program separate data transfer jobs for barcode strings, string length, error code or string identifier, it is usually much more efficient to transfer all the data at once – in a single job. Due to the fact that all the data is stored consecutively at the beginning of the Memory Buffer (in the Storage Structure) it is possible to use, for example, the Unit Identifier (which is the first member

of the Storage Structure) as the Source Reference and then to transfer as many words of data to cover all the length of the Storage Structure.

To make it possible to use “On Request” data transfers, the Interlock Mailbox has to be activated. Typically the Interlock Mailbox would be placed in the memory of the controller that is receiving barcode data and then the controller will have to be programmed to handle the Mailbox.

The Datalogic protocol has the ability to handle the Mailbox itself (without any need for programming). To use this feature, the Mailbox must be placed in the Memory Buffer at offset of 420. If made so, the UniOP will execute the Job 1 data transfer just after it receives the barcode string from the Datalogic Barcode Scanner.

5.1 The Fast Data Transfer

In addition to “In Cycles” and “On Request” data transfers, the Datalogic driver offers an additional option to transfer received barcode strings to the other controller, which is communicating with the Panel. It is called the Fast Data Transfer.

When the Fast Data Transfer is used, barcode strings will be transferred to the other controller much faster than in the case of “In Cycles” or “On Request” data transfer.

To use this feature the standard Designer’s data transfer mechanism is not used. Instead, the Fast Data Transfer can be configured in the Controller Setup dialog box (see chapter 2.1).

If the Panel is already configured for dual-driver operation it will be possible to check the “Fast Data Transfer” checkbox enabling all other options.

As the data source it will only be possible to choose references offered by the Datalogic barcode scanner driver. Instead, for the data Destination address it will be possible to select references offered only by the second driver. Press the “Source” or the “Destination” buttons to configure these references.

Number of transferred bytes must also be defined in the “Number of bytes” text box.

Note: *At the moment, the Fast Data Transfer can only be used in dual-driver configuration. Over-UniNET transfers are not supported. Only single fast data transfer job is supported.*

6. Sending commands to Barcode Scanner

It is possible to send commands to the Barcode Scanner connected to Panel. To use this option, the “Send Command” macro can be used in the Macro Editor.
The Send Command feature can be useful whenever the connected device needs to be re-programmed in order to be compatible with current Datalogic-like implementation of the driver.

After pressing the “OK” button a simple data entry dialog box will popup.

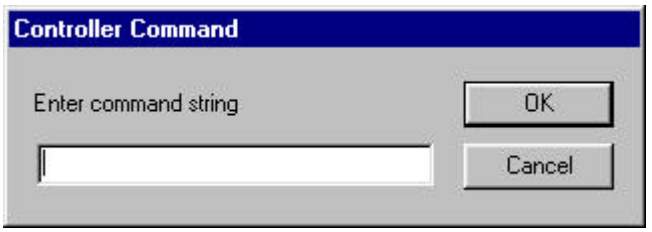

Figure 5 – Data entry dialog box for entering command string

Generally, the command string will be sent to the Barcode Scanner exactly as it is typed in this dialog box. However, when a non-printable character must be sent to the scanner, it is possible to use the following syntax (escape sequence) inside the command string:

<num>

where *num* represents decimal ASCII code of the character that must be placed instead of the escape sequence. For example, to send the "STOP" string followed by CR and LF characters to the scanner, a string like this must be typed: **STOP<13><10>**

There are predefined mnemonics for some often used non-printable characters. Refer to the following table:

ASCII code	Mnemonic	ASCII code	Mnemonic	ASCII code	Mnemonic
0	NUL	11	VT	22	SYN
1	SOH	12	FF	23	ETB
2	STX	13	CR	24	CAN
3	ETX	14	SO	25	EM
4	EOT	15	SI	26	SIB
5	ENQ	16	SLE	27	ESC
6	ACK	17	CS1	28	FS
7	BEL	18	DC2	29	GS
8	BS	19	DC3	30	RS
9	TAB	20	DC4	31	US
10	LF	21	NAK		

The above example thus could be also typed as: **STOP<CR><LF>**

There must be no spaces inside an escape sequence. If an escape sequence is not valid, the string will be sent to the scanner exactly as typed.

If there are two consecutive "**<**" characters found in typed string, only a single "<" character will be actually sent to the barcode scanner.

7. Advanced Programming

To use only the basic, above-mentioned functions it is not necessary to know anything about the advanced programming feature. However using these advanced functions the following is achievable:

1. The Data Buffering (storing more than just one barcode string into the UniOP memory).
2. Thanks to the Data Buffering it is possible to show a short history of received strings on UniOP display (not only the very last received string) without need for any special controller programming (or even without any controller at all).

The general idea is to have the possibility to place the Storage Record anywhere in the Memory Buffer – not only at the very beginning of the Memory Buffer. By placing newly received strings dynamically on different locations it is possible to keep old strings intact when a new string is received. Also, placing the Storage Record back to an old location, the Barcode String, String Length, Error Code and the String Identifier will be turned into old data.

In addition, it is possible to have the protocol itself automatically change the position of the Storage Record just before a new string is stored. This requires defining the minimum and maximum allowed location and the length of the Storage Record. Address range from 460 to 469 of the Memory Buffer is used for advanced programming.

Offset	Description	Default
460-461	**Memory valid indicator** (2-byte string) Fixed to string 'OK'. If this string is changed the protocol will clear the Memory Buffer at the next power up.	'OK'
462-463	**Storage Record Position** (word) Points to the beginning of the Storage Record. This identifies the position, in the Memory Buffer, where the Storage Record is placed.	0
464-465	**Min. Location** (word) If the Data Buffering is used, it points to the minimum offset address allowed for the first byte of the Storage Structure. Otherwise it has no meaning.	0
466-467	**Max. Location** (word) If the Data Buffering is used it points to the maximum offset address allowed for the first byte of the Storage Structure. Otherwise it has no meaning.	400
468	**Storage Record Length** (byte) It informs about how many bytes of the Storage Record must be cleared to zero (filled with "0") before the newly received string is placed over the old one. It allows to clearly display the new received string without overlapping with the previous one. If the Data Buffering is enabled, it also informs about how much the Storage Record Position pointer must be increased (if the Max. Location is not yet reached).	0
469	**Enable Data Buffering** (byte) 0 – Data Buffering disabled 1 – cyclical Data Buffering enabled. When the Storage Record Position reaches the Max. Location it is re-pointed to the Min. Location and starts filling the buffer from the beginning overwriting the old data. 3 – scrolling Data Buffering enabled. When the Storage Record Position reaches the Max. Location value, it stays fixed but every time a new string is received the old data is scrolled back and the oldest string is lost.	0

Appendix A. Communication Error Codes

Current communication status is displayed in the System Menu of the UniOP.
A message and a numeric error code describe the error status.
The message reports the current communication status. The number shows the code of the current communication error or, if the communication is correct, the code of the last error encountered. When the error code 0 is shown, it means there have been no communication errors since this system start-up.

Code	Description	Notes
00	**No error**	There are no communication errors and there have been no errors since start-up.
07	**General communication error**	Should never happen
11	**Line Error**	Parity error or similar