ALLTEC

Operating Instructions

AllprintBasic

Version 1.x

Keep for future use!

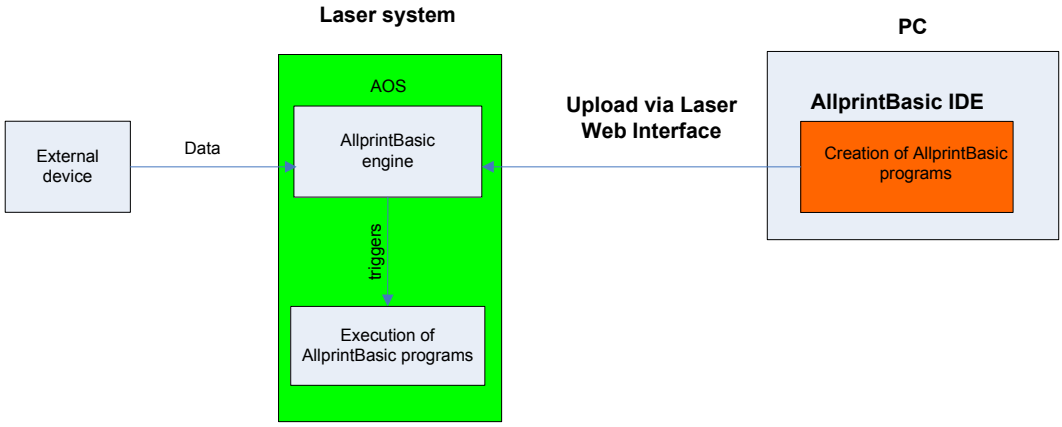
CE • Date: March 2008• Part No.: AL-71703 • Index: AA

ALLTEC International Service Centre
An der Trave 27-31 | 23923 Selmsdorf | Germany
Phone 0049.(0)38823.55-360 | Fax 0049.(0)38823.55-301
Email: service@alltec.org | www.alltec.org

1 Introduction

1.1 General

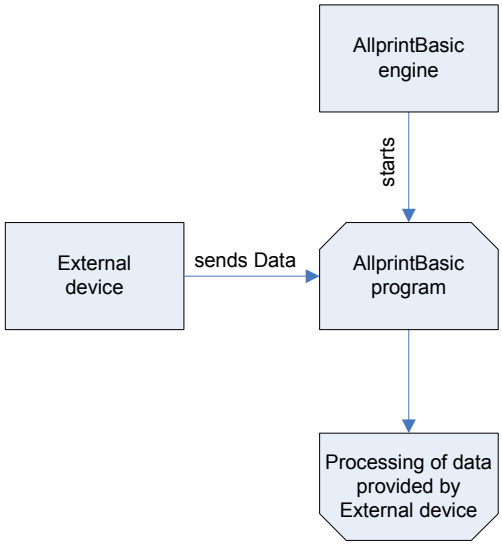
With the Allprint Basic software or IDE (Integrated Development Environment), so-called AllprintBasic programs are created which are uploaded onto the laser system via the Laser Web Interface. On the laser system (in the AOS), the Allprint Basic engine starts the Allprint-Basic program which has been activated on the laser system. This program receives data from an external device and is executed as long as it is supplied with input. In the following figure, a general overview of the interaction between the different components is displayed:



1.2 Program types

The data from the external device can be sent to the laser via the serial interface, the digital I/O interface or Ethernet. As these interfaces are different, the AllprintBasic programs, which are used to process the received information, are also different.

Generally, these programs consist of configuration elements and instructions defined with a Basic-like programming language - the AllprintBasic software - which determine the reaction of the laser to events or messages. When being started by the AllprintBasic engine, the AllprintBasic program, that has been activated in the laser database, is executed. When receiving input data from an external device, these are processed continuously by the AllprintBasic program. This process is displayed in the following figure:



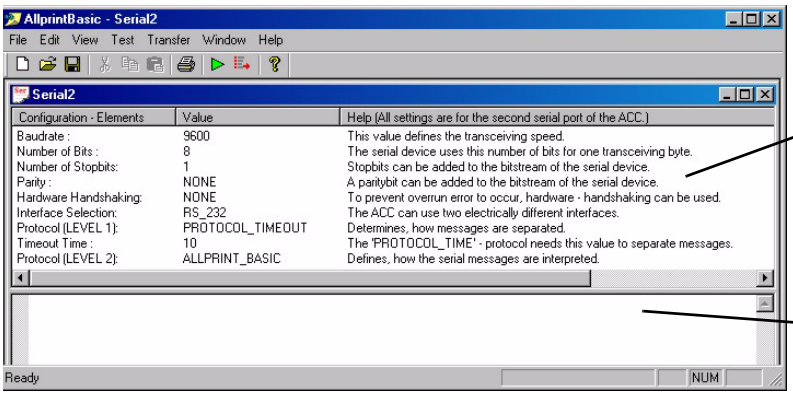
There are two different types of programs:

- Serial programs
- Digital programs

The differences of these program types are described in the following sections.

1.2.1 Serial programs

Before the serial interface of the laser can be used to receive data from the external device, this has to be configured in a serial AllprintBasic program.



The following standard configurations are available:

- Baud rate (600, 1200, 2400, 4800, 9600, 19200, 38400)
- Number of data bits (8, 7)
- Parity (NONE, ODD, EVEN)
- Number of stop bits (1, 2)
- Hardware handshaking (NONE, RTS_CTS)
- Interface selection (RS_232, RS_422)

Additionally, different protocols (see page 8) can be selected which describe how the laser should react when receiving data.

1.2.1.1 Change serial configuration

In the upper part of a serial program, the configuration of the serial interface of the laser system can be changed by clicking on one of the configuration elements with the mouse.

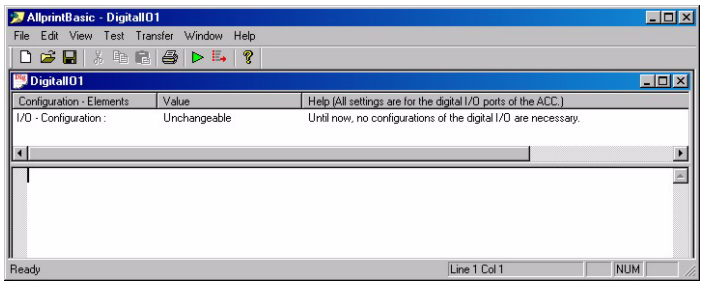
1.2.2 Serial connection to the external device

The serial cable from the external device has to be connected to the interface inside the laser system. For more information about the respective interface of your laser system and the connections to be established, please refer to the operating instructions of your laser system.

1.2.3 Digital programs

Digital programs are used to define the reaction of the laser to signals of the digital port. Normally, the input pins generate a one byte message on the strobe signal. This message byte will be interpreted by the AllprintBasic instruction string to execute the corresponding programm.

At the digital port, no special configurations are necessary. Therefore, there are no changeable configuration elements in the upper part of a digital program.



In the lower part of a digital program, an AllprintBasic instruction string can be inserted.

1.2.4 Digital connection to the external device

The digital cable from the external device has to be connected to the interface inside the laser system. For more information about the respective interface of your laser system and the connections to be established, please refer to the operating instructions of your laser system.

1.3 Protocols

In the configuration settings part of the serial AllprintBasic program, different protocols can be selected which describe the reaction of the laser when receiving data.

1.3.1 Low level communication protocol

The task of this protocol is to separate a stream of characters, which are received by the respective interface, into individual messages. Furthermore, some predefined characters are interpreted in a special way.

There are different protocols available:

- The timeout protocol
- The EOTSUM protocol
- The UDPSequence protocol

These protocols can be selected in the upper part of a serial program by clicking on the configuration element »Protocol (Level 1)«.

1.3.1.1 The timeout protocol

This protocol separates data streams by using timeouts. The timeout can bet set in units of 20 ms in the upper part of a serial program by clicking on the configuration element »Timeout Time«.

When receiving data using this protocol, the laser system awaits the first character of a message. A timeout timer is restarted every time a character is received. If the timeout timer elapses without receiving a character, all characters received so far are logically grouped into one message which is delivered to the AllprintBasic engine for subsequent processing by the current program.

Outgoing messages are sent immediately without any delay and without modification.

1.3.1.2 The EOTSUM protocol

This protocol manages a transfer of ASCII characters between the external device and the laser and offers framing and a simple checksum.

The external device frames each telegram with a »Start of Text« and »End of Transmission« character. The message is completed by the transmission of the checksum *cc*. The protocol follows the rule [STX] "mydata" [EOT] cc.

If the laser receives the telegram and computes the same cc, it replies with an acknowledge [ACK], otherwise with a negative acknowledge [NACK].

If the cc was correct, the laser can reply to the external device using the command »Send«. The response is framed with [EOT] and cc.

Example:

```
<STX>"mydata"<EOT>cc #sent by the host PC

<ACK>#generated automatically by the protocol

126<EOT>cc #if the AllprintBasic program reacts with send("126") to
the incoming "mydata"
```

The checksum is a simple 8-bit addition including all control characters.

1.3.1.3 The UDPSequence protocol

This proctocol operates on top of the UDP (IP) protocol via the Ethernet connection of the laser, i.e. when selecting this protocol, the »Interface Selection« entry is automatically changed to »Ethernet«. Each datagram is preceded by a sequence number. This sequence number is used to detect missing datagrams or the ones which were received in the wrong order. If the order of numbers is discontinuous, the laser system changes into the »Error« state.

1.3.2 High level communication protocol

After the characters from the serial interface have been combined to a complete message by the low level communication protocol, the messages are relayed to this protocol level. Here, the messages are interpreted and assigned to the commands in the AllprintBasic program.

There is one protocol available:

- The AllprintBasic protocol

This protocol is selected in the upper part of a serial program by clicking on the configuration element »Protocol (Level 2)«.

1.3.2.1 The AllprintBasic protocol

This protocol provides the greatest flexibility to adapt to the customer's specifications. With a special AllprintBasic instruction string the reaction of the laser system to a message of the customer can be programmed.

Example:

The customer wants the laser system to start if an »A« is sent to the laser and to stop if any other character is sent. This can be implemented by using the following AllprintBasic instruction string:

```
If (Left(1) == "A") Then Start; Else Stop;
```

For an overview of the commands, please refer to page 23.

1.4 AllprintBasic instruction string

In the lower part of an AllprintBasic program, the instruction string can be edited. This string defines the reaction of the laser system to a message from the external device.

Therefore, a Basic-like language was developed. With some string manipulation functions, the message of the external device can be interpreted in several ways.

The notation of this language is as follows:

[]:means: optional

():means: use one of

{}:means: repeat optional even no time

{}1+:means: repeat optional but minimum one time

| :means: or

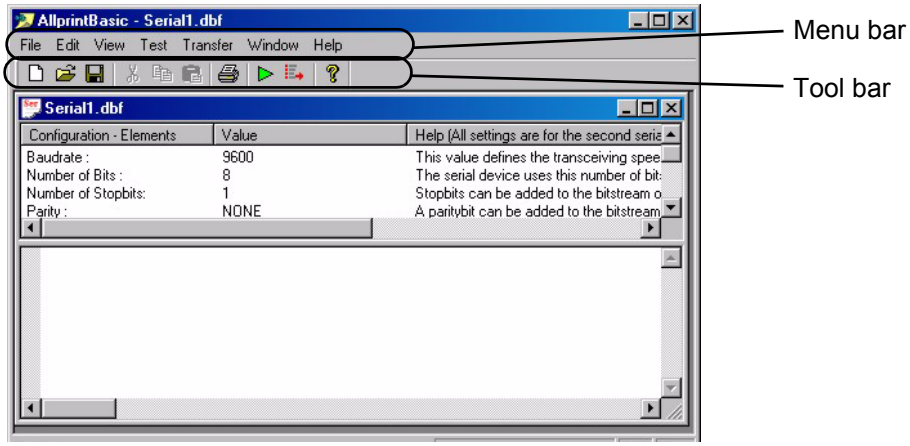
For more information about the commands to be used in the AllprintBasic instruction string, please refer to page 23.

2 Overview of the AllprintBasic software

In this chapter, a menu overview of the AllprintBasic software is given, which is used to create and/or edit the AllprintBasic programs.

2.1 AllprintBasic software menus

After having started the software on your PC and created or opened an AllprintBasic program, the following user interface is displayed:



2.1.1 The File menu

New	Creates a new program (serial or digital).
Open	Opens an AllprintBasic program.
Close	Closes the active AllprintBasic program. If modifications have been made to the file, a prompt appears asking whether you want to save the changes before closing the file.
Save	Saves the program and modifications (if any).
Save as	Saves the program and modifications (if any) under a different name.
Print	Prints the active program.
Print Preview	Shows a print preview of the active program.
Print Setup	Here, print setting can be made.
[Name of opened files]	Displays the name(s) of the opened program.
Exit	Closes the AllprintBasic program.

2.1.2 The Edit menu

Undo	Makes any number of steps undone.
Redo	Repeats any number of undone steps.
Cut	Cuts the selected element from the program. The cut object is moved to the clipboard and can be pasted at a different position (see Paste).
Copy	Copies the selected element from the program. The copied object is moved to the clipboard and can be pasted at a different position (see Paste).
Paste	Pastes the contents of the clipboard into the program.
Find	Displays the searched text element.
Replace	Substitutes the searched text element with the text element which has been entered in the appearing dialog.

2.1.3 The View menu

Toolbar	If a checkmark is set, the toolbar is displayed.
Status Bar	If a checkmark is set, the status bar is displayed.

2.1.4 The Test menu

Simulate serial/ digital program	Tests the reaction of the laser system to the serial or digital program.

2.1.5 The Transfer menu

Import/Export/ Delete	*These menu entries are not implemented any more.*

2.1.6 The Window menu

New window	The active program is opened in a new window.
Cascade	All opened programs are arranged in a cascaded way.
Tile	All opened programs are displayed next to each other.
Arrange icons	The icons representing the minimized windows are rearranged within the application window if the IDE.
[Names of the currently open programs]	Displays the names of the currently opened programs. The active program is indicated by a checkmark.

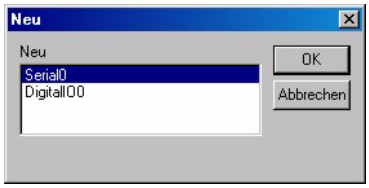
3 Getting started

3.1 Creating programs

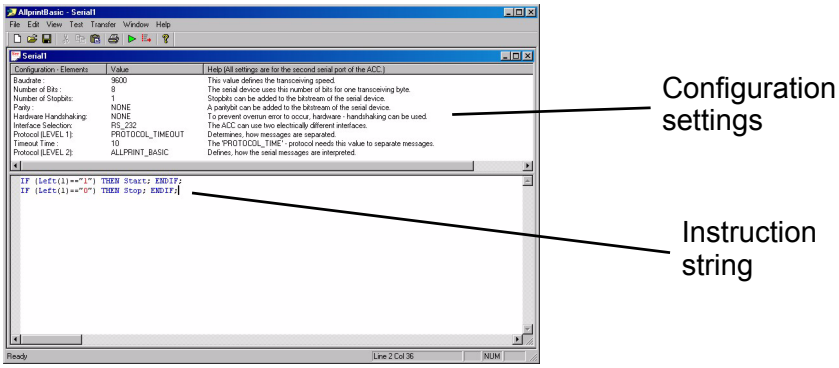
With the AllprintBasic software, two different types of AllprintBasic programs can be created.

3.1.1 Creating a serial program

1. Start the AllprintBasic software.
2. Click on the »New« icon in the toolbar or choose »New« from the File menu.
   The following dialog appears.



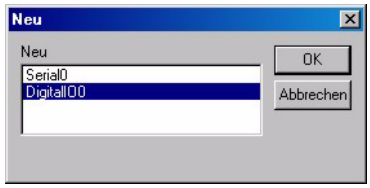
3. Select »Serial0« and click on the »OK« button.
   The following dialog appears.



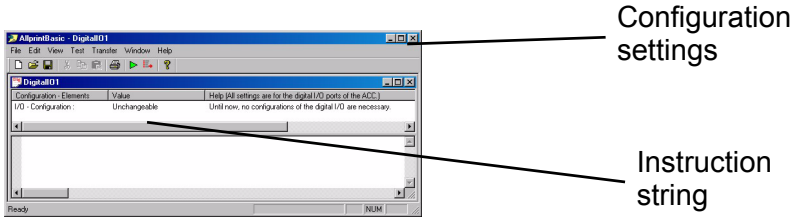
4. Choose the appropriate configuration settings for the serial interface of your laser.
   For more information about the configuration settings, refer to page 7.
5. Enter an instruction string, which defines the reaction of the laser to a message from the customer, into the lower part of the program.
   For more information about the commands to be used in the AllprintBasic instruction string, please refer to page 23.

3.1.2 Creating a digital program

1. Start the AllprintBasic software.
2. Click on the »New« icon in the toolbar or choose »New« from the File menu.
   The following dialog appears.



3. Select »Digital00« and click on the »OK« button.
   The following dialog appears.



4. Enter an instruction string, which defines the reaction of the laser to a message from the customer, into the lower part of the program.

3.2 Testing programs

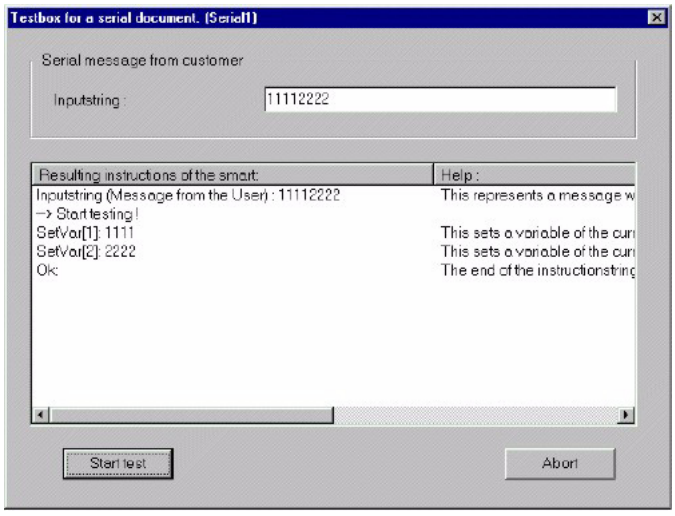
An AllprintBasic instruction string (see page 10) can be tested by a special dialog: The Test Box.

Press the key »F5« or select »Simulate...« from the Test menu to display the test dialog.

3.2.1 Testing serial programs

The test dialog can simulate the reaction of the laser system to serial message strings. For that purpose, the serial message string must be entered under »Serial message from customer«. After clicking on the button »Start test«, a commented execution trace of the program is displayed.

If the Testbox command needs data from a running laser system, a window will appear prompting the customer to type in the required data.

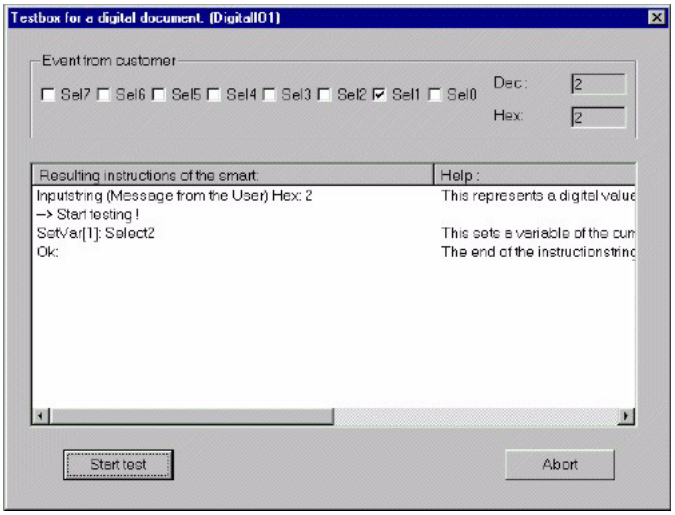


Click on »Abort« to exit the test dialog.

3.2.2 Testing digital programs

The test dialog can simulate the reaction of the laser system to digital events. Therefore, the digital one byte message string must be entered under »Event from customer«. The eight lines of the digital interface can be changed. After clicking on the button »Start test«, a commented execution trace of the program is displayed.

If the Testbox command needs data from a running laser system, a window will appear prompting the customer to type in the required data.



Click on »Abort« to exit the Test dialog.

3.3 Tutorial (Serial program)

This tutorial shows how to configure and use the serial interface of the laser system in order to adapt it to the customer's demands.

Example:
The customer wants to mark two independent words at a predefined position of his product. The words are always four characters long and the marking shall be changed every second via a serial device which is configured as follows:

- 9600 Baud
- 8 Bit
- 1 Stop bit
- No parity
- No handshaking
- Standard RS232 interface

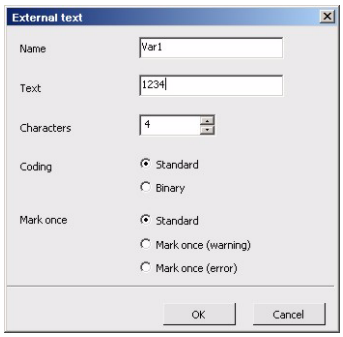
The customer sends the following string: **`'AAAABBBB'`**, time gap: >500 ms. The first four characters are the first word and the last four character are the second word to be marked on the product.

3.3.1 Creating a template for the laser system

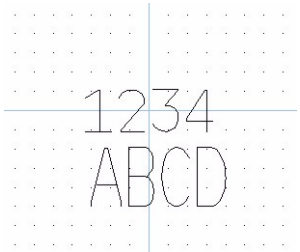
First of all, a template for the laser which contains two variables has to be created with the *Smart* Graph software. The Variable 1 contains the initiation text »1234« and variable 2 contains the text »ABCD«.

1. Open the »External text« dialog in the *Smart* Graph software.
   For more information about the *Smart* Graph software, please refer to the *Smart* Graph documentation.

2. Fill in the dialog as shown in the figure below for creating the first variable.



3. Create the second variable (Var2Text) with the text ABCD.
   The template in the *Smart* Graph software is displayed as follows:



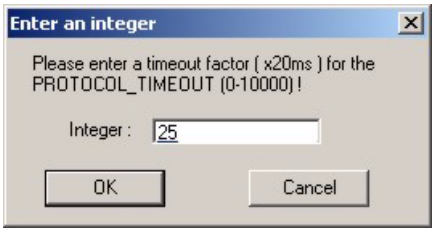
3.3.2 Creating the AllprintBasic instruction string

The next step is to generate the serial program and the AllprintBasic instruction string.

1. Create a new serial program.
   For more information about the creation of a serial program, refer to page 11.

2. Click on the »Baud rate« configuration element in the Configuration settings part.
   This will adjust the Baud rate to 9.600 Baud.

3. Click on the »Timeout Time« configuration element in the Configuration settings part.
   The following dialog is displayed.

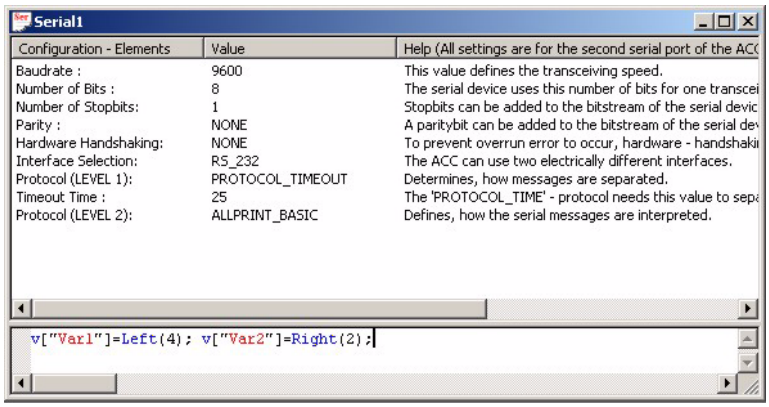


4. Type »25« to set a timeout time of about 500 ms, because the customer wants the marking to change only every second.
   Time gaps between two characters of more than 500 ms will then separate messages which are sent from the customer to the laser system.

   Note The update of the laser system variables is also delayed by 500ms!

   The rest of the configuration elements do not have to be adjusted for this example.

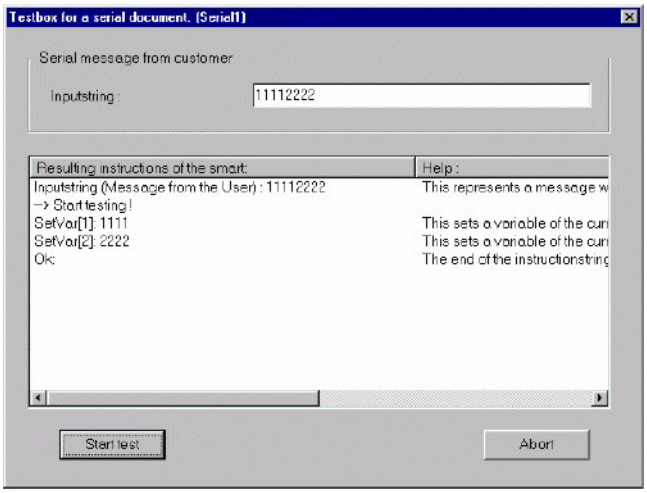
5. Enter the following AllprintBasic instruction string in the lower part of the program: **`V["Var1"]=Left(4); V["Var2"]=Right(4)`**;
   The first four characters of the customer's message are assigned to variable 1 and the last four characters are assigned to variable 2.
   The program should now look like this and can be saved:



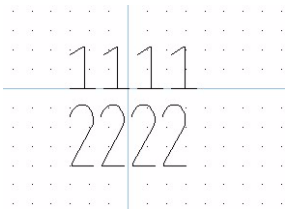
3.3.3 Test the AllprintBasic instruction string

Now the AllprintBasic instruction string can be tested.

1. Press the button »F5« or access the test dialog by selecting »Simulate...« from the Test menu..



The message »11112222« from the customer is divided correctly into two parts and assigned to the correct variables. If the message was sent to the laser system, the marking would look like this:



3.3.4 Upload the serial program to the laser

After the AllprintBasic instruction string has been tested, the serial program can be uploaded onto the laser system. For more information about uploading the AllprintBasic program file, please refer to page 40.

3.4 Tutorial (Digital program)

This tutorial shows how to configure and use the digital interface of the laser system in order to adapt it to the customer's demands.

Example:

The customer wants to mark a text at a predefined position of his product. He will provide two digital signals (SEL0, SEL1) and a strobe signal. For every strobe signal the two 'data lines' determine which text has to be marked on the product.

SEL0 = low, SEL1 = low➔Mark the text "Select0"

SEL0 = high, SEL1 = low➔ Mark the text "Select1"

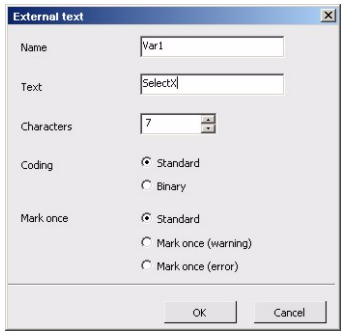
SEL0 = low, SEL1 = high➔ Mark the text "Select2"

SEL0 = high, SEL1 = high➔ Mark the text "Select3"

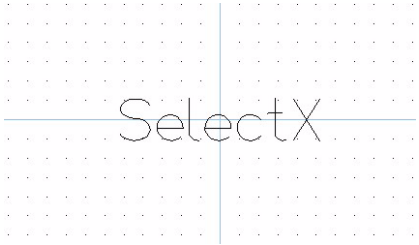
3.4.1 Creating a template for the laser (digital)

First of all, a template for the laser which contains one variable has to be created with the *Smart* Graph software. The Variable 1 contains the initiation text »SelectX«.

1. Open the »External text« dialog in the *Smart* Graph software.
   For more information about the *Smart* Graph software, please refer to the *Smart* Graph documentation.
2. Fill in the dialog as shown in the figure below for creating the variable.



3. The template in the *Smart* Graph software is displayed as follows:



3.4.2 Generate the AllprintBasic instruction string (digital)

The next step is to generate the digital program and the AllprintBasic instruction string.

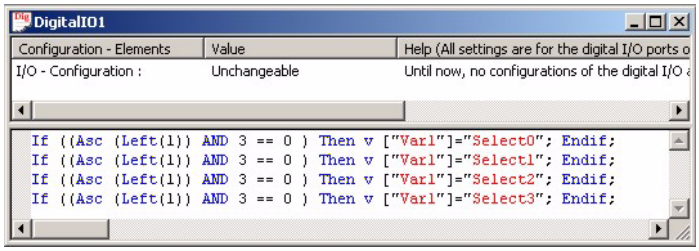
1. Create a new digital program.
   For more information about creating digital programs, please refer to page 11.
   Until now no configuration elements for the digital interface must be set.

2. Enter the following AllprintBasic instruction string in the lower part of the program:

```
If ( (Asc(Left(1)) AND 3) == 0 ) Then v["Var1"]="Select0"; Endif;
If ( (Asc(Left(1)) AND 3) == 1 ) Then v["Var1"]="Select1"; Endif;
If ( (Asc(Left(1)) AND 3) == 2 ) Then v["Var1"]="Select2"; Endif;
If ( (Asc(Left(1)) AND 3) == 3 ) Then v["Var1"]="Select3"; Endif;
```

With every strobe signal this AllprintBasic instruction string is interpreted. The message from the customer is a one byte string that can be accessed by the instruction: 'Left(1)'. This byte is built from all digital lines: SEL0(Bit 0) to SEL7 (Bit 7). Therefore the instructions above select which text is assigned to variable one of the current job, which is running on the laser system.

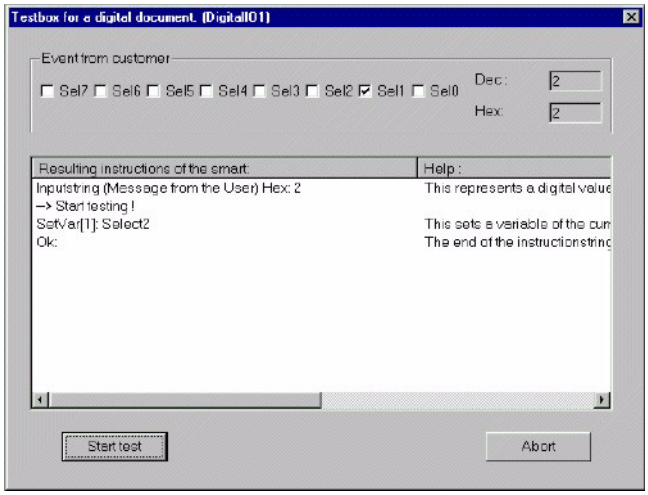
The program should now look like this and can be saved.



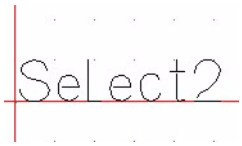
3.4.3 Test the AllprintBasic instruction string (digital)

Now the AllprintBasic instruction string can be tested.

1. Press the button »F5« or access the test dialog by selecting »Simulate...« from the Test menu.



The digital event '0x00000010' sets the lines SEL0 to low and SEL1 to high and the variable one is correctly set to the text »Select2«. If the message would be sent to the laser system the marking would look like this:



3.4.4 Uploading the digital program to the laser

After the AllprintBasic instruction string is tested, the digital program can be uploaded onto the laser system. For more information about uploading the AllprintBasic program file, please refer to page 40.

4 Commands

As already stated on page 10, the AllprintBasic instruction string is a sequence of commands which are described in detail in this chapter. These commands are separated by a ';' or a New-Line and are written in a Basic-like language the notation of which is as follows:

[]:means: optional

():means: use one of

{}:means: repeat optional even no time

{}1+:means: repeat optional but minimum one time

| :means: or

4.1 General commands

In the following, the general commands are described:

Remark

Notation:

```
'This is a remark
```

Lines starting with an apostrophe are ignored by the parser and can be used to enter remarks or comments.

'IF', 'THEN', 'ELSE', 'ENDIF'

Notation:

```
IF (ConditionChain) THEN Commandchain {ELSE Commandchain}ENDIF
```

The instructions if, then, else and endif can be used for conditional controlling.

Example: Set '1234' as current job because condition is true.

IF (1==1) THEN J = "1234"; ENDIF;

'Error'

Notation:

```
Error
```

This command sets the laser system to an error state.

See also: **MsgBox()**, page 24, **Warning**, page 27

'MsgBox()'

Notation:

```
MsgBox(TextOperandChain)
```

This command opens a message window on the connected user interface.

Example:

You want the user to select another job.

```
MsgBox( "Please select another job" )
```

See also: **Error**, page 24, **Warning**, page 27

'J' Set the current job

Notation:

```
J = TextOperandChain
```

This command sets the current job of the laser system. The database of the laser system has to contain the specified job. Otherwise, an error occurs.

Example:

The job '1234' should be selected as the current job.

```
J = "1234"
```

See also: **GetCurrentJob()**, page 30

'V' Set a job variable

V[ID]

V[ID,Jobname]

V[Variablename]

V[Variablename,Jobname]

V[Variablename, GLOBAL]

Notation:

```
V[ValueOperandChain[,TextOperandChain][ = TextOperandChain
```

```
V[TextOperandChain[,TextOperandChain][ = TextOperandChain Version 4.0 or higher
```

```
V[TextOperandChain[,GLOBAL][ = TextOperandChain Version 4.0 or higher
```

This command sets a variable - defined by an integer identifier (1....) or by the name within the template - of the current or specified job of the laser system to the specified string. If you use the constant GLOBAL for the job you change all variables with the specified name in the system.

This can be done only if the laser template contains a related variable. If there is no related variable, the laser system generates an error message.

Note: Please make sure that the elements which you create in the *Smart* Graph software exactly match the requirements of the external device, e.g. the length and the name of a variable which you create in the dialog »External text« in the *Smart* Graph software have to match with the variable names referred by the AllprintBasic program. Otherwise, an error occurs.

Examples:

Variable 1 of the current job is set to '1234'

```
V[1] = "1234"
```

Variable 1 of the current job is set to the value of variable 2

```
V[1] = V[2]
```

Variable(s) with the name 'Name' of the job 'Variable' is(are) set to 'Hello World'

```
V["Name", "Variable"] = "Hello World"
```

Variables with the name 'Name' in all jobs of the system are set to 'Hello World'

```
V["Name", GLOBAL] = "Hello World"
```

'SetVariable'

Notation:

`SetVariable(TextOperandChain, TextOperandChain )`

You can insert a variable text into the database. The text "aValue" is store under the name "akey". This text can be read with **`'GetVariable()'.`** Don't mix this command with the command **`'V[]'`**. The **`'SetVariable()'`** does not work with the variable of the current job.

See also: **`GetVariable()`**, page 31

'Start' Start laser marking

Notation:

`Start`

This command starts the marking process. If the instruction string contains only this instruction, the laser system starts the marking process with every message the customer sends.

See also: **`Stop`**, page 26

'Stop' Stop laser marking

Notation:

`Stop`

This command stops the marking process. If the instruction string contains only this instruction, the laser system stops the marking process with every message the customer sends.

See also: **`Start`**, page 26

'Trigger'

Notation:

`Trigger`

If the product registration of the laser is set to "program" and it is started then you can use this command to start the marking.

See also: **`Start`**, page 26

'Warning'

Notation:

`Warning`

This command sets the laser system to a warning state.

See also: **`MsgBox()`**, page 24, **`Error`**, page 24

'SetPos'

SetPos(Jobname, x, y)

Notation:

`SetPos(TextOperandChain,ValueOperandChain, ValueOperandChain)`

Sets the position of the specified job to the values x and y. The units are in mm.

See also: **`'SetExtent'`**, page 27 **`'SetRotation'`**, page 27 **`'GetCurrentJob()'`**, page 30

'SetExtent'

SetExtent(Jobname, width, height)

Notation:

`SetExtent(TextOperandChain,ValueOperandChain, ValueOperandChain)`

Sets the extent of the specified job to the values width and height. The units are in mm. If you specify "0, 0" the job will have the extent of the template.

See also: **`'SetPos'`**, page 27 **`'SetRotation'`**, page 27 **`'GetCurrentJob()'`**, page 30

'SetRotation'

SetRotation(Jobname, angle)

Notation:

`SetRotation(TextOperandChain,ValueOperandChain)`

Sets the rotation of the specified job to the value specified with angle. The units are in degree.

See also: **`'SetPos'`**, page 27 **`'SetExtent'`**, page 27 **`'GetCurrentJob()'`**, page 30

'Send()'

Notation:

```
Send(TextOperandChain)
```

This command allows you to send an ASCII string to the host PC.

Example:

You want to send the number of pulses (i.e. 1000) to the host.

Send("1000")

'SetMarkingCounter'

Notation:

```
SetMarkingCounter(jobname,value)
```

This command sets the marking counter for the job 'jobname' to the defined 'value'.

Example:

You want to set the marking counter of job ALLTEC to 1000.

```
SetMarkingCounter("ALLTEC","1000")
```

'GetMarkingCounter'

Notation:

```
GetMarkingCounter(jobname)
```

This command returns the marking counter value of the specified job.

Example:

You want to set the first variable to the counter value of job ALLTEC.

```
V[1] = GetMarkingCounter("ALLTEC")
```

'SetRTC'

Notation:

```
SetRTC(yyyy:mm:dd:hh:mm:ss)
```

This command sets the real time clock (RTC) on the ACC.

Make sure that the format is correct (year:month:day:hour:minute:second).

Example:

You want to set the RTC to April 10, 2006 at 12 o'clock.

```
SetRTC("2006:04:10:12:00:00")
```

'GetRTC'

Notation:

```
GetRTC()
```

This command returns the value the real time clock (RTC) on the ACC.

Example:

You want to set the first variable to the RTC value.

```
V[1] = GetRTC()
```

'WaitForMarking'

All commands after 'WaitForMarking' are executed only when the system is in state "Marking".

4.2 The text operands

Text operands are strings. The counting of the characters within the strings is displayed in the follow example:

```
"ABCD12"
```

Text operands are used e.g. to set the values of job variables.

Text operands can be added to a chain.

Notation:

```
TextOperandChain = TextOperand {+ TextOperand}
```

Simple Text

Notation:

```
"12345"
```

A simple text can be a text operand if it is between quotation marks.

'Chr()'

Notation:

```
Chr(ValueOperandChain)
```

This function returns the ASCII character represented by the value of the parameter.

See also: **Asc()**, page 34

'GetCurrentJob()'

Notation:

```
GetCurrentJob()
```

This function returns the name of the current job.

Example:

You want to set the name of the first variable in the job to the current job.

```
V[1] = GetCurrentJob()
```

See also: **'J' : Set current job**, page 24

'GetVariable()'

Notation:

```
GetVariable(TextOperandChain)
```

You can retrieve a variable text from the database. The function returns the text which is stored under "aKey". This text can be set with **'SetVariable()'**. Don't mix this command with the command **'V[]'**. The **'GetVariable()'** does not work with the variable of the current job.

Example:

Send the value of the variable **MyVariable** to the connected user interface.

```
MsgBox( "The Variable is : " + GetVariable("MyVariable")  )
```

See also: **SetVariable()**, page 26

'Left()'

Notation:

```
Left(ValueOperandChain)
```

The parameter of the instruction 'Left()' specifies the number of characters that are copied from the message string, starting at the left end of the string. The result is a text operand.

Example:

```
Message string: "ABBCCAB".

Left (4); ' Result = "ABBC"

Left (1); ' Result = "A"
```

See also: **Mid()**, page 31, **Right()**, page 32

'Mid()'

Notation:

```
Mid(ValueOperandChain [,ValueOperandChain])
```

The first parameter of the instruction 'Mid()' specifies the start point (1...) and the second parameter specifies the number of characters (0...) which are copied from the customer's message. If the second parameter is not entered, the rest of the customer's message, beginning at the start point, is copied. The result is a text operand.

Example:

```
Message string: "ABBCCAB"

Mid (4,2); ' Result = "CC"

Mid (1,3); ' Result = "ABB"
```

See also: **Left()**, page 31, **Right()**, page 32

'Right()'

Notation:

```
Right(ValueOperandChain)
```

The parameter of the instruction 'Right()' specifies the number of characters that are copied from the message string, starting at the right end of the string. The result is a text operand.

Example:

```
Message string: "ABBCCAB"

Right (4); ' Result = "CCAB"

Right (1); ' Result = "B"
```

See also: **Left()**, page 31, **Mid()**, page 31

'Str()'

Notation:

```
Str(ValueOperandChain)
```

The parameter (integer) of the instruction 'Str()' is converted into a string. The result is a text operand.

Example:

```
Str(123); ' Result = "123"
```

See also: **Val()**, page 36

'Swap()'

Notation:

```
Swap(TextOperandChain)
```

The parameter (string) of the instruction `'Swap()'` generates a new string with swapped characters. The result is a text operand.

Example:

```
Swap ("AB12"); 'Result = "21BA"
```

4.3 The value operands

Value operands are integers and may be used e.g. for text operand operations. Value operands can be added to a chain.

Notation:

```
ValueOperandChain = (+|-)ValueOperand{(+|-|And|Or|*|/)ValueOperand} |
"("ValueOperandChain")"
```

The operator precedence is as follows:

Operator Priority

unary minus, ( )High

```
/, *
```

```
+, -
```

```
And, Or Low
```

'123' Simple Value

Notation:

```
{(0..9)} 1+
```

A simple integer can be a value operand.

'Asc ()'

Notation:

```
Asc(TextOperandChain)
```

`'Asc()'` returns the ASCII value of the first character in the string. The result is a value operand.

Example:

```
Asc ("A"); ' Result = 65
```

See also: `Chr()`, page 30

'CountOccurrence ()'

Notation:

```
CountOccurrence(TextOperandChain)
```

With this instruction the occurrence of a string in the customer's message is counted. Every character of the parameter string is compared to the customer's message. If the case sensitive comparison succeeds the occurrence of the parameter string is counted. The result is a value operand.

Example:

Message string: `"ABBCCAB"`

```
CountOccurrence ("B"); ' Result = 3

CountOccurrence ("Z"); ' Result = 0
```

'InputLen'

Notation:

```
InputLen
```

'InputLen' is a constant integer which represents the length of the customer's message. The result is a value operand.

Example:

Message string: `"ABBCCAB"`

```
Left (InputLen); ' Result = "ABBCCAB"
```

See also: **Len()**, page 36

'InStr ()'

Notation:

```
InStr(ValueOperandChain, TextOperandChain)
```

'InStr()' searches for the first occurrence of the second parameter string in the customer's message, beginning at the position specified by the first parameter. The result is a value operand which defines the start position in the customer's message where an equal (case sensitive) string begins. If the second parameter string is not found, the function returns zero.

Example:

Message string: **"ABBCCAB"**

InStr(4,"C"); ' Result = 4

InStr(1,"CAB"); ' Result = 5

See also: **InStrRev()**, page 35

'InStrRev ()'

Notation:

```
InStrRev(ValueOperandChain, TextOperandChain)
```

'InStrRev()' works like **'InStr()'**, but search direction is inverted.

Example:

Message string: **"ABBCCAB"**

InStrRev (InputLen, "AB"); ' Result = 6

InStrRev (5, "AB"); ' Result = 1

See also: **InStr()**, page 35

'Len()'

Notation:

`Len(TextOperandChain)`

`'Len()'` returns the length of a string. The result is a value operand.

Example:

```
Len ("AB12"); ' Result = 4
```

See also: **`InputLen`**, page 34

'Val ()'

Notation:

`Val(TextOperandChain)`

`'Val()'` converts a string into an integer. The result is a value operand.

Example:

```
Val ("123AB"); ' Result = 123
```

See also: **`Str()`**, page 32

4.4 The conditional operands

This operand type is used for conditional instructions and it compares two value operands or two text operands. The result is boolean (TRUE, FALSE) and can only be used in conditional instructions. Condition operands can be added to a chain.

Notation:

```
Conditionchain = ConditionOperand {(&&| ||) ConditionOperand}
```

`&&` is a logical and.

`||` is a logical or.

Compare value operands

Notation:

```
Condition = ValueOperandChain (== | != | > | < | >= | <=) ValueOpe-
randChain
```

This condition operand compares two value operands. The result is a boolean.

Example:

`If (12 == 12) Then Start`; ' Result = The laser would be started, because condition is true.

`If (12 > 13) Then Start`; ' Result = The laser wouldn't be started, because condition is false.

Compare text operands

Notation:

`Condition = TextOperandChain (== | != | > | < | >= | <=) TextOperand-Chain`

This condition operand compares two text operands. The result is a boolean.

Example:

Message string: **`"ABBCCAB"`**

`If (Left(1) == "A") Then Start`; ' Result = The laser would be started, because condition is true.

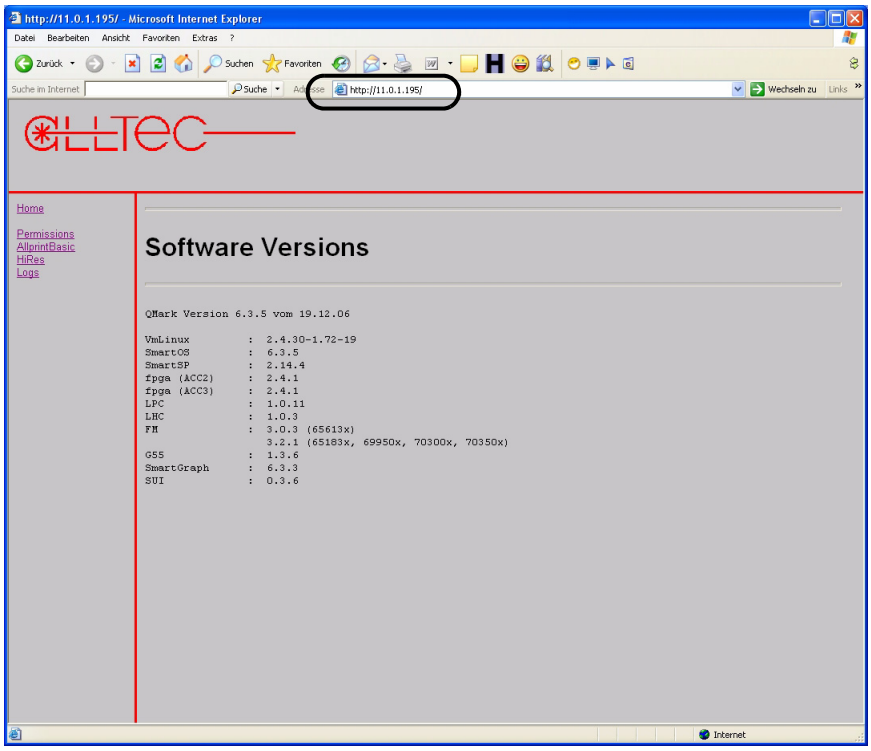
`If (Right(1) != "B") Then Start`; ' Result = The laser wouldn't be started, because condition is false.

5 Uploading of AllprintBasic programs

In the following, you will find an instruction how to upload and activate the created AllprintBasic programs via the laser web interface.

In order to open the respective site on your PC, proceed as follows.

2. Open your Internet Explorer and enter the IP address of your laser system.
   The corresponding home site is opened, displaying an overview about the current software versions which are installed on the laser system and a navigation column on the left hand side.

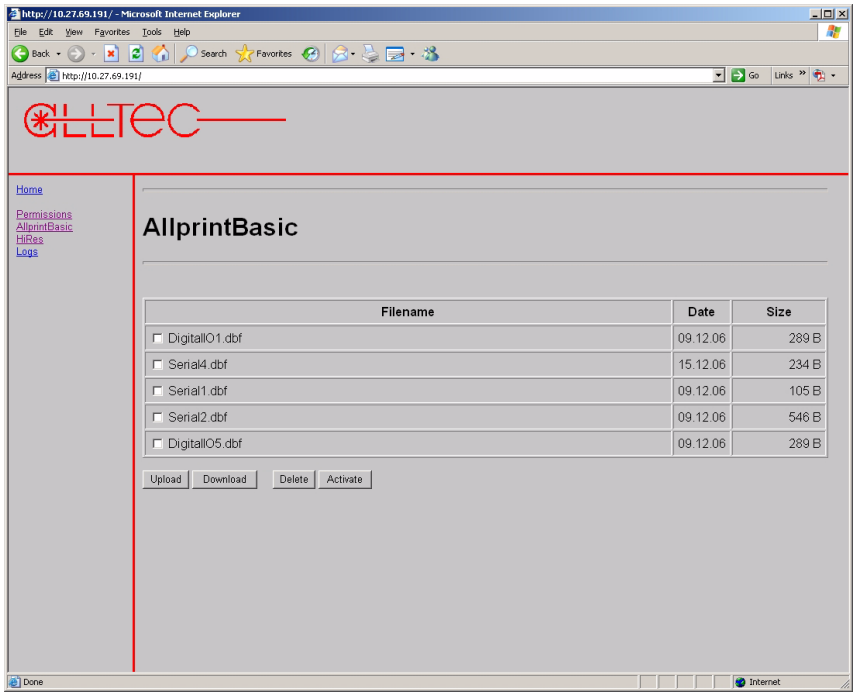


5.1 Uploading AllprintBasic programs via the laser web interface

By clicking on the AllprintBasic link in the column on the left side, the following page is displayed showing an overview about the AllprintBasic program files available on your laser system.

If no AllprintBasic program files are available on your laser system yet, the AllprintBasic page

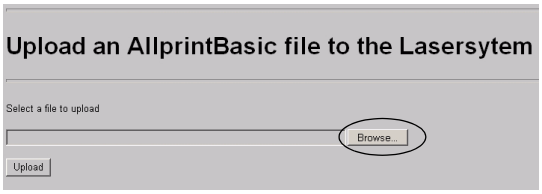
will be blank.



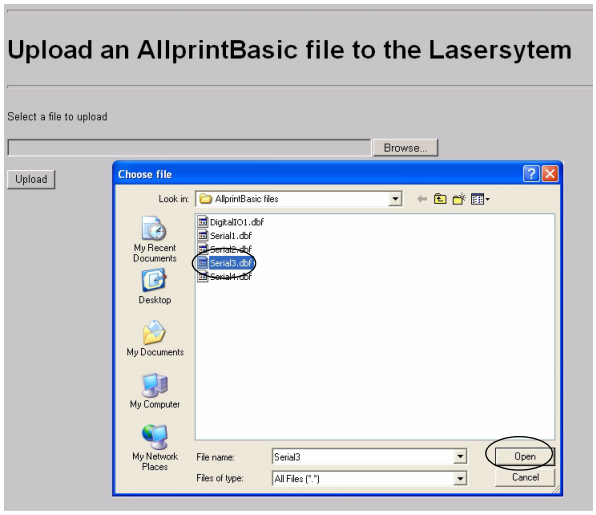
5.1.1 Uploading an AllprintBasic program to the laser system

In order to upload the AllprintBasic program from your PC to the laser system, proceed as follows:

1. Click onto »Upload« on the main »AllprintBasic« page.
2. In the following dialog, click onto »Browse«.

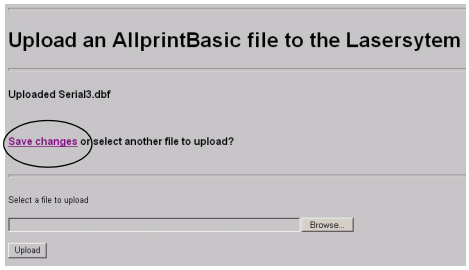


3. Select and open the respective AllprintBasic program file.

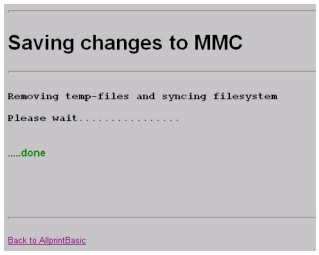


4. Click onto »Upload«.

5. In the following dialog, click onto »Save changes« if you want to save the changes to the MMC.

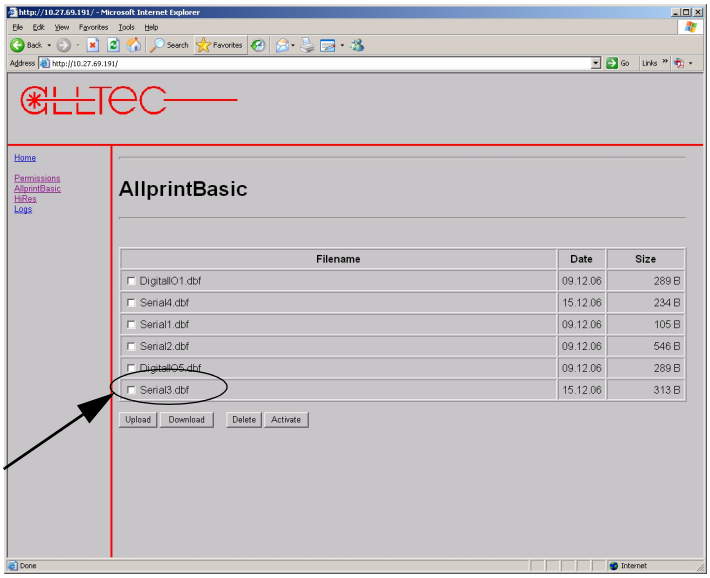


The following confirmation dialog is displayed.



Note: If you wish to upload various AllprintBasic program files, click again onto »Upload« in the dialog shown under point 3.

The selected AllprintBasic program file is uploaded to the laser system and is displayed (including the upload date and the file size) on the main AllprintBasic page.



Note: If the AllprintBasic program file which you want to upload, already exists on the laser system, you will be informed by the following dialog.

5.1.2 Activating an AllprintBasic program file

In order to activate one of the AllprintBasic program files available on your laser system, please proceed as follows:

1. Set a checkmark in front of the AllprintBasic program you wish to activate and click onto »Activate«.

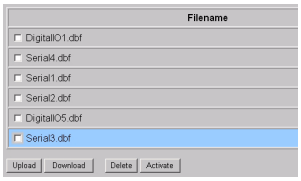
   The corresponding AllprintBasic program is activated on your laser system and highlighted by a blue bar.

Note: Please note that only **one** AllprintBasic program can be activated at a time!

5.1.3 Deactivating AllprintBasic program files

In order to deactivate the AllprintBasic program files available on your laser system, please proceed as follows:

1. Make sure that **no** checkmark is set in front of any AllprintBasic program file and click onto »Activate«.



All AllprintBasic program files available on your laser system are deactivated.

5.1.4 Deleting an AllprintBasic program file from the laser system

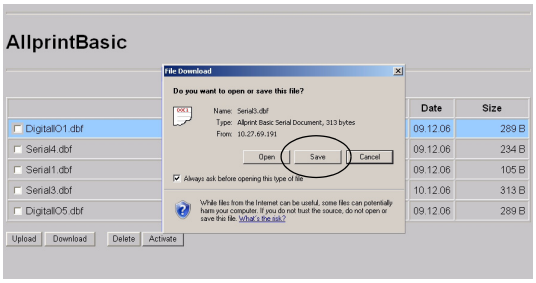
1. Set a checkmark in front of the AllprintBasic program you wish to delete and click onto »Delete« to remove the respective program file from your laser system.
   The program file is removed from your laser system.

Note: Please note that the activated AllprintBasic program file (highlighted by a blue bar) cannot be deleted, but has to be deactivated first (see “Deactivating AllprintBasic program files” on page 43.

5.1.5 Downloading an AllprintBasic program file onto a PC

It is possible to download the AllprintBasic program files, which are available on your laser system, onto a PC, e.g. to make a backup. Please proceed as follows:

1. Set a checkmark in front of the AllprintBasic program you wish to download and click onto »Download« to download the respective program file on your PC.
   The following Windows dialog is displayed:



2. Select the directory where you wish to save your program file and click onto »Save« to save the respective program file on your PC.
   The program file is saved into the selected directory.

6 Examples

In this chapter, some examples are given and an error list is explained.

6.1 Complex examples

6.1.1 Example 1

The laser system shall be started when it receives a "1" from the customer and stopped if it receives a "0".

Solution:

AllprintBasic instruction string:

```
IF (Left(1)=="1") THEN Start; ENDIF;

IF (Left(1)=="0") THEN Stop; ENDIF;
```

Result:

Message string: `"1"->Start;`

6.1.2 Example 2

The variable which is selected by the number before the ":" shall be set to the text operand after the ":".

Solution:

AllprintBasic instruction string:

```
V[Val(Left(InStr(1,":")-1))] = Right (InputLen-InStr(1,":"));
```

Result:

Message string: `"1:200" -> V[1] = "200";`

6.1.3 Example 3

Variable 3 shall be set to last character in the customer's message, variable 2 shall be set penultimate character and the rest of the messages shall be assigned to variable 1.

Solution:

AllprintBasic instruction string:

```
V[3] = Right(1); V[2] = Mid(InputLen-1,1);

V[1] = Left(InputLen-2);
```

Result:

Message string: `"1AMV:" -> V[3] = ":"; V[2] = "V"; V[1] = "1AM";`

6.1.4 Example 4

Variable 1 shall be set to the number of "A"s and variable 2 shall be set to the number of "B"s in the customer's message.

Solution:

AllprintBasic instruction string:

```
V[1] = Str(CountOccurrence("A"));

V[2] = Str(CountOccurrence("B"));
```

Result:

Message string: `"ABBA" -> V[1] = "2"; V[2] = "2";`

6.1.5 Example 5

The selection of the current template as well as the content of a variable of the current template is to be controlled remotely by the RS-232 interface of the laser.

It is specified that a message of the form:
Case 1:
`[STX]Job;JOB_NAME;[EOT][Checksum]`
switches the current template to the template named: JOB_NAME,

while a message of the form:
Case 2:
`[STX]Var;VAR_CONTENT;[EOT][Checksum]`
sets the content of the variable with the predefined name “Var1” to VAR_CONTENT.

The laser shall acknowledge the switching of the current template according to case “1”, by returning the name of the current template.

After setting the content of variable “Var1” according to case “2”, the laser shall acknowledge the subsequent mark by returning the literal “End of Mark!”.

`[STX]`: Start of Text character (Ascii 0x02)
`[EOT]`: End of Transmission character (Ascii 0x04)
`[Checksum]`: checksum byte, unsigned addition of all telegram characters including the control characters
`Job`: string “`Job`”, serves as command token to trigger the template selection
`Var`: string “`Var`”, serves as command token to trigger the variable setting
`;`: semicolon, serves as delimiter to terminate the individual items of the request telegram
`JOB_NAME`: string denoting the template to be selected
`VAR_CONTENT`: string which is to be assigned to the external text variable

Solution:

For the transmission of the telegrams, the EOTSUM protocol is chosen which is responsible for the separation of the serial character stream into individual telegrams according to the above protocol convention.

Allprint Basic instruction string:

```
' --- Template Selection

IF (Mid(1,3) == "Job" ) THEN

J = Mid(InStrRev(InStrRev(InputLen,";")-1,";")+1,↵
InStrRev(InputLen,";")-InStrRev(InStrRev(InputLen,";")-1,";")-1);

Send(GetCurrentJob())

ENDIF


' --- Variable Assignment

IF (Mid(1,3) == "Var" ) Then

V["Var1"] = Mid(InStr(1,";")+1,InStr(InStr(1,";")+1,";")-↵
(InStr(1,";")+1));

WaitForMarking

Send("End of Marking!")

ENDIF
```

Result:

```
Message String "Job;TheJob;" -> J = "TheJob"; send("TheJob")

Message String "Var;NewContent; -> V["Var1"] = "NewContent";
<Mark> -> send("End of Marking!")
```

6.2 Errorlist

While testing the AllprintBasic instruction string (see section »Testing programs« on page 14), errors may occur. This overview describes the possible causes of error messages.

The AllprintBasic instruction string is read from left to right until an error occurs or the hole string is processed. The displayed error location (line and column) is normally one character behind the instruction which has caused the error.

It is possible that errors only occur at special message strings.

Error	Description
Error 1	Undefined error! This error message is displayed every time the error cannot be specified in detail.
Error 2	Missing end mark! Every instruction must be separated from the following instruction by an apostrophe or a new line.
Error 3	Missing quotation mark! The simple text operand must be started and ended by a quotation.
Error 4	Illegal parameter! Some instructions or text operand functions need parameters in the defined range. Example: **`Left(-1)`**; ' Illegal parameter
Error 5	Missing bracket! Some instructions need parameters between brackets (open and close).
Error 6	Missing equation mark! An equation mark was expected.
Error 7	No memory! While interpreting the AllprintBasic instruction string the program needs to allocate some memory and the allocating failed.
Error 8	Illegal text operand! The parser expected a text operand, but there was no correct one.
Error 9	Illegal value operand! The parser expected a value operand, but there was no correct one.
Error 10	Missing comma! If there is more than one parameter in an instruction, it has to be separated by a comma.
Error 11	Illegal text chain! The parser did not find a correct text chain.
Error 12	Ilegal condition operand! The operand for the comparison is wrong.
Error 13	Missing **`'Then'`** identifier! Every 'IF' instruction needs a **`'Then'`** identifier.
Error 14	Missing **`'Endif`**' identifier! Every 'IF' instruction needs an **`'EndIf'`** identifier.
Error 15	Unknown condition! The following conditions are available: `==', `!=', `>', `<', `>=', `<='

Error	Description
Error 16	Type mismatch! For comparisons it is necessary that the parts which shall be compared are of the same type. Text operands can only be compared to text operands, and value operands to value operands.
Error 17	Missing **'EndIf'** or **'Else'**! The parser expects an **'EndIf'** or an **'Else'** identifier for the 'IF' instruction.
Error 18	Integer is too long! The entered integer should not have more than 10 digits.
Error 19	Division by zero!
Error	Unknown error appears! An error appeared which is not documented.