Application Note

USING THE BAR CODE SCANNER FOR PROGRAM SELECTION

Dynalab
TEST SYSTEMS

Using the Bar Code Scanner for Program Selection

This document explains how to use PASS 6.0 software to program the Dynalab Analyzer to scan bar-coded product identification information from a harness label, then select and execute the associated test program.

This document has the following main sections:

1. A list of assumptions – knowledge required to perform the tasks outlined in this document.
2. An explanation of the problem
3. An explanation of the solution
4. An overview of the Dynalab solution to the problem, including example Sequences.

Assumptions

To successfully use this document, the following knowledge is required:

- basic knowledge of how to enter harness data using PASS® 6.0
- knowledge of how to use the Sequence table to create a Sequence

For assistance on how to use features of PASS® 6.0, see the PASS® 6.0 Help file.

Problem

The Dynalab Circuit Analyzer is capable of storing several test programs in memory. Generally, it is the Operator's responsibility to manually select the correct program for the harnesses to be tested. Sometimes, the Operator selects the wrong program. A related problem is that it takes too long for the Operator to manually select the program.

Solution

The Analyzer prompts the Operator to scan the bar-coded product label on the harness. The Analyzer then selects and executes the associated test program. This method of program selection reduces errors caused by manual selection and increases productivity.

Solution Overview

The Dynalab Bar Code Laser Scanner is used to input bar code data directly into XL Series Analyzers using one of the RS-232 serial ports. The Analyzer can be programmed to use the data input from the scanner to select and execute the correct test program.

Serial Port Connection

Each scanner is equipped with a serial cable. The serial cable connector should be connected to the Analyzer's Serial Port 1, since this is the default port for serial data input to the Analyzer. If this is not suitable, the scanner's serial cable connector may be connected to Serial Port 2. This would require that the SERIAL2 Sequence item be inserted at the beginning of the program Sequence to instruct the Analyzer to monitor Serial Port 2 for input.

Programming the Analyzer to read bar coded product ID and select the correct PASS program

The Analyzer can be programmed to use one of the following two approaches to select and execute the associated test program:

1. Find a program in the Analyzer's memory whose name is an exact match with the scanned Product ID.
2. Find a program in the Analyzer's memory that is associated with the scanned Product ID.

A detailed description of each approach follows.

Find and execute a program whose name is an exact match with the scanned Product ID

The SELECT Sequence item is used to accomplish this. When SELECT is executed, the Analyzer monitors the serial port until a valid character string is received. Once a character string is input from the bar code scanner, SELECT looks for a program in the Analyzer's memory whose name is an exact match with the character string. If an exact match is found, the program is selected and executed. If no program is found in the Analyzer's memory whose name matches the scanned character string, execution continues with the Sequence item following SELECT.

There are several approaches to solving this problem. Perhaps the most obvious approach is to have one program that will prompt the operator to scan the product code and then employs the SELECT Sequence item to execute the correct program. The problem with this approach is that the Operator must manually load this program to get things started. This is not a comprehensive solution to the problem: to automatically select the correct program based on the scanned product code.

A better approach would be to design every program on the Analyzer to always prompt the Operator to scan the product code first and then to select the correct program to run. The following example illustrates this approach. The bold-highlighted program lines should be included in each Analyzer program for this approach to work.

Example Sequence to find and execute a program whose name matches scanned Product ID

Line	Command	Parameter	Application Effect
1	**HOLDING=**	**0**	Set value of Holding Register
2	**RCOUNT>**		If RCOUNT > Holding Register, Set Holding Register to 1
3	**BHR**	**9**	Branch on non-zero Holding Register value
4	**HOLDING=**	**99**	Set value of Holding Register
5	**RCOUNT=**		Set RCOUNT equal to Holding Register value
6	TEST	MAIN	Complete Netlist Scan
7	REPORT		Display Summary Report
8	KWAIT		Wait for START button
9	**RCOUNT=0**		Set RCOUNT equal to zero
10	**MESSAGE**	10	Display message 10: “SCAN Bar Code”
11	**SELECT**		Read program name from serial port and execute
12	**HOLDING=**	99	Set value of Holding Register
13	**RCOUNT=**		Set RCOUNT equal to Holding Register value
14	**KMESSAGE**	20	Display message 20: “Program Not Found Press START to continue”
15	**GOTO**	9	Go to Line Number 9

Line 1 HOLDING= 0: Sets the value of the holding register to zero

Line 2 RCOUNT>: Compares the value of RCOUNT to the holding register. If RCOUNT is greater than the value in the holding register, then the holding register is set to 1.

Line 3 BHR 9: Examines the value of the Analyzer’s holding register and branches to the specified line number if the value of the holding register is greater than zero. In this case, execution branches to line 9 if the value of the holding register is greater than 0 or continues with the next line if the value of the holding register is equal to 0. In this example, this means that execution branches to line 9 if value of RCOUNT was equal to zero in lines 1 and 2.

Line 4 HOLDING=99: Sets the value of the holding register to 99

Line 5 RCOUNT=: Sets RCOUNT equal to the holding register.

Line 6 TEST: Performs a complete test scan of the entire harness.

Line 7 REPORT: Displays a summary report

Line 8 KWAIT: Waits for the Operator to press the START button

Line 9 RCOUNT=0: Sets the value of RCOUNT to zero.

Line 10 MESSAGE 10: Displays the message defined as message number 10 in the Messages table. This message is defined as “SCAN Bar Code Label”

Line 11 SELECT: Monitors the serial port, and waits for a character string to be received from the scanner. Once a character string is received, SELECT looks for a program in the Analyzer’s

memory whose name is an exact match. If found, the program is executed. If not found, execution continues with the next line.

Line 12 HOLDING= 99: Sets the value of the holding register to 99.

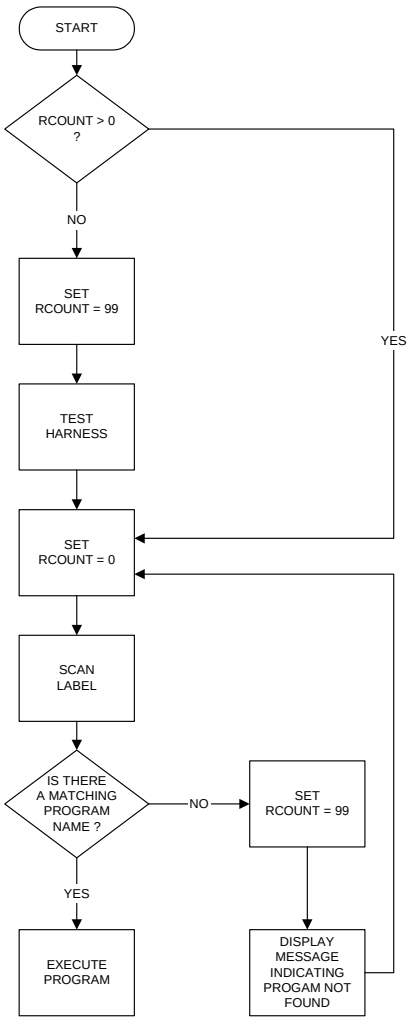
Line 13 RCOUNT=: Sets RCOUNT equal to the holding register.

Line 14 KMESSAGE 20: Displays the message defined as message number 20 in the Messages table. This message is defined as "PROGRAM NOT FOUND Press START to continue"

Line 15 GOTO 9: Instructs the Analyzer to go to line 9

The bold lines shown in the example Sequence above should be included in every program on the Analyzer that could be selected based upon reading the bar-coded product identification. The non-bold lines show the default test sequence. These can be substituted with any suitable test sequence.

If every program on the Analyzer is constructed in this way, no matter which program the Operator manually runs, the program will always first prompt the Operator to scan the bar code label and will automatically select the correct program based upon the scanned product identification.



There is a programming "trick" employed here to provide logic which determines if a program has been "selected" - in which case testing should immediately proceed, or if the program was Operator-initiated, - in which case the bar code label must be scanned. The trick relies uses the Report Counter (RCOUNT) as a flag. Here is the logical rule that is employed in the Sequence:

If RCOUNT = zero, then the program has been "selected" as a result of a bar code scan. In this case, testing must proceed.

If RCOUNT > zero, then the program was Operator-initiated, in which case the bar code must be scanned.

The flowchart at left illustrates the logical flow.

Find and execute a program in the Analyzer's memory that is associated with the scanned Product ID

This approach is used when the Analyzer's program names do not match the harness product codes. Each harness product code is however, associated with an Analyzer test program.

Consider the following situation. There are five different harness product codes. The table below shows the name of the associated Analyzer test program for each product code.

Harness Product Code	Analyzer Program Name
HPN-1001-R1	HPN-1001
HPN-1001-R2	HPN-1001
HPN-1001-R3	HPN-1001
A7568903	ASERIES
X5A79	5A79

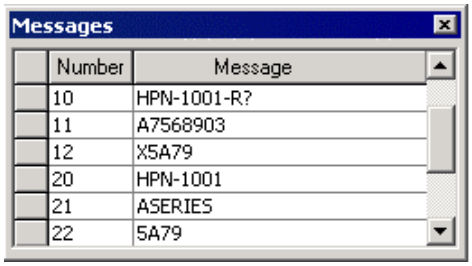
The Operator scans the bar-coded product code on the harness label. The Analyzer finds and executes the test program associated with the scanned harness product code.

This approach is implemented in the PASS program by first placing entries in the Messages table. One entry must exist for each valid harness product code, and one entry must exist for each associated program name.

The Analyzer will then be able to compare the scanned character string with each entry in the Messages table corresponding to a valid harness product code. If a match is found, the Analyzer will then execute the program whose name is contained in the associate entry in the Messages table.

Note that the first three valid harness product codes are exactly alike except for the last character: (HPN-1001-R1, HPN-1001-R2, and HPN-1001-R3). This will allow for the use of the wildcard character: '?'. Whenever the '?' character appears, it means that any character is valid in that position. So, instead of three entries in the Messages table, only one is needed using the wildcard character: H-1001-R?

The Messages table should have the following entries:



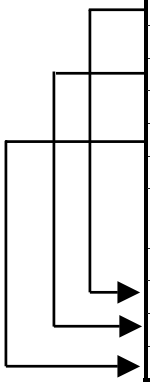
Messages 10, 11, and 12 contain the harness product codes.

Messages 20, 21, and 22 contain the associated Analyzer test program names.

Note that Message 10 will match any of the first three harness product codes, since it uses the wildcard character in the last position.

Using the Messages table described above, the following example Sequence illustrates how to program the Analyzer to execute a program that is associated with the scanned Product ID. This program prompts the Operator to scan the bar code label on the harness. Once the bar code character string is received, the string is compared with each of the five entries in the messages table that contain the valid harness product codes. If a match is found, the associated program is executed. If no match is found, the Operator is informed.

Example Sequence to execute a program that is associated with the scanned Product ID



Line	Command	Parameter	Application Effect
1	MESSAGE	1	Displays message 1: “SCAN Bar Code Label”
2	STRING		Reads string variable from serial port
3	STRCMP	10	Compares input string to message 10: HPN-1001-R?
4	BHR	12	Branch to line 12 if the holding register is greater than zero
5	STRCMP	11	Compares input string to message 11: A7568903
6	BHR	13	Branch to line 13 if the holding register is greater than zero
7	STRCMP	12	Compares input string to message 12: X5A79
8	BHR	14	Branch to line 14 if the holding register is greater than zero
9	SOUND	12	Plays a sound indicating that a matching program name is not found.
10	KMESSAGE	2	Displays message 2: Program Not Found for |x Press START to continue
11	REPEAT		Repeats the Sequence, starting at line 1
12	RUN	20	Executes the program whose name is contained in message 20
13	RUN	21	Executes the program whose name is contained in message 21
14	RUN	22	Executes the program whose name is contained in message 22

Line 1 MESSAGE 1: Displays the message defined as message number 1 in the Messages table. This message is defined as “SCAN Bar Code Label”.

Line 2 STRING: reads a string of characters from the Analyzer’s serial port. In this example, STRING is reading characters from the Bar Code Scanner. After STRING is invoked, the Analyzer will wait until the bar code is scanned. After the bar code is scanned, the string of characters is stored in the Analyzer’s string buffer and program execution continues with the next Sequence item. (By default, the Analyzer monitors Serial Port 1 for input from a scanner. If this is not suitable, the SERIAL2 Sequence item may be inserted at the beginning of the Sequence to instruct the Analyzer to monitor Serial Port 2 for input.)

Line 3 STRCMP 10: compares the contents of the Analyzer’s string buffer to the entry in the Messages table referenced by the parameter. In this case, the parameter is 10 – so STRCMP will compare the scanned bar code characters with the contents of message 10 in the Messages table. In this example, message 10 is defined as “HPN-1001-R?”. STRCMP will declare a successful match if the scanned bar code string is of exactly the same length as message 10, and if all the characters match exactly except last character. If STRCMP finds a match, the value of the Analyzer’s holding register is set to 1. If STRCMP

determines that the two character strings do not match, the value of the holding register is set to zero.

Line 4 BHR 12: Examines the value of the Analyzer's holding register and branches to the specified line number if the value of the holding register is greater than zero. In this case, execution branches to line 12 if the value of the holding register is greater than zero or continues with the next line if the value of the holding register is equal to zero. In this example, execution branches to line 12 if the scanned bar code matches the contents of message 10 (HPN-1001-R?), meaning that the scanned bar code was either HPN-1000-R1, or HPN-1000-R2, or HPN-1000-R3.

Line 5 STRCMP 11: compares the contents of the Analyzer's string buffer to message 11: A7568903. If STRCMP finds a match, the value of the Analyzer's holding register is set to 1. If STRCMP determines that the two character strings do not match, the value of the holding register is set to zero.

Line 6 BHR 13: Examines the value of the Analyzer's holding register and branches to line 13 if the value of the holding register is greater than zero

Line 7 STRCMP 12: compares the contents of the Analyzer's string buffer to message 11: X5A79. If STRCMP finds a match, the value of the Analyzer's holding register is set to 1. If STRCMP determines that the two character strings do not match, the value of the holding register is set to zero.

Line 8 BHR 14: Examines the value of the Analyzer's holding register and branches to line 14 if the value of the holding register is greater than zero

Line 9 SOUND 12: Plays a sound to indicate that a matching program name was not found (none of the STRCMP operations above were successful).

Line 10 KMESSAGE 2: Displays message 2: "Program Not Found for |x Press START to continue". The Analyzer will continue to display this message until the operator presses the START button. (the "|x" format symbol will cause the scanned characters to be displayed as part of the message).

Line 11 REPEAT: Instructs the Analyzer to go to line 1 which repeats execution of the Sequence

Line 12 RUN 20: Executes the program whose name is the same as the contents of message 20: HPN-1001

Line 13 RUN 21: Executes the program whose name is the same as the contents of message 21: ASERIES

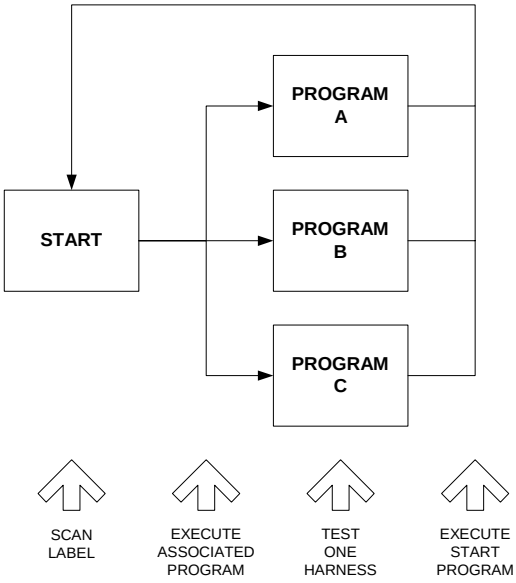
Line 14 RUN 22: Executes the program whose name is the same as the contents of message 22: 5A79

Program Structure

The previous example illustrates how the bar code scanner can be used to by the Analyzer for program selection. In each example, once a valid selection is made, the desired program is executed. Of course, once the desired program is executed, the original program is no longer running.

The programs should be structured as follows for this method to work.

A program named START is initially executed. This program provides the same functionality as illustrated in the previous example Sequence. The START program prompts the operator to scan a label, then selects and executes the associated test program. The test program then runs to test one harness. After one harness is tested, the test program executes the START program, and the process repeats.



To implement this method, each test program's Sequence must execute the RUN Sequence item after a test is completed. RUN requires a parameter: the message number of the entry in the Messages table that contains the name of the program to execute. Therefore, each test program's Messages table must have an entry that contains the name of the START program. This entry is then referenced by the RUN Sequence item after a harness has been tested.