for Windows Computers

Application Builder™ for DuraTrax®/LaserLite®/LaserLite Pro/ LaserLite Mx

Table of Contents

Introduction

Welcome to Application Builder for DuraTrax, LaserLite, LaserLite Pro, and LaserLite Mx data collectors. With this software you can design application programs specific to your data collection needs. Application Builder makes it easy by prompting you to describe the steps of the application and then creating an entire data collection program based on your descriptions.

With Application Builder you can create any type of application from a simple "Scan a bar code" application to a complex application with a variety of branching steps and specific requirements.

Application Builder …

- Graphically represents and defines the steps in your application program
- Designs the data collector's display for each step
- Connects the steps in the order you want the data entered
- Builds an entire application program based on the path you create when connecting these steps

Applications created with Application Builder have a number of built-in features including:

- Scrolling through data
- Delete last entry
- Low-battery voltage warnings
- Display current battery voltage
- Power-loss warning messages
- Display remaining memory
- Low memory warning messages

Applications created with Application Builder are capable of generating BASIC source code. Users that are familiar with BASIC can modify the BASIC source code for advanced customization of their application.

Application Builder for Windows consists of this manual and one CD.

This manual consists of four chapters and three appendixes.

Chapter 1 is an introduction to Application Builder and its associated files. It includes a description of the files and two quick-start sections: one that steps you through installing and using a sample application, and one that steps you through building a cross-reference file.

Chapter 2 contains descriptions of the Application Builder menu commands and windows.

Chapter 3 steps you through building a simple application.

Chapter 4 steps you through building more complex applications using some of the more advanced features of Application Builder.

Appendix A contains descriptions and bar codes for the sample applications.

Appendix B contains information on resetting the data collector.

Appendix C contains information on using multiple data collectors and Base Stations, and changing a data collector's ID.

Chapter 1

Application Builder

This chapter provides:

- A description of the Application Builder program
- A description of the contents of the Application Builder disks
- A discussion about applications created with Application Builder
- A quick-start that steps you through installing the Application Builder files, transferring a sample application to the data collector, using the application to enter data, and viewing the data file
- A quick-start that steps you through using the New CRF command to build a cross-reference (CRF) file

Discussion

Application Builder creates data collection applications and transfers them to your data collector. The primary purpose of an application is to collect data and create a text file of the collected data. What distinguishes one application from another is the *way* it collects data. When an application is running on the unit, there is a single record of the data file being constructed in the unit's memory. This record is known as the **current record**. The application does its job by collecting data from the user and appending it to the end of the current record. When the current record is complete, it is added to the end of the data file and a new current record is started.

When you create an application, you need to specify at least three things:
1. The type of input that is acceptable.
2. The portions of the input that are added to the current record.
3. What the data collector's display should look like.

As you create and define the application, you are actually describing the application's path.

The path for a security application might look like the application in Figure 1-1.

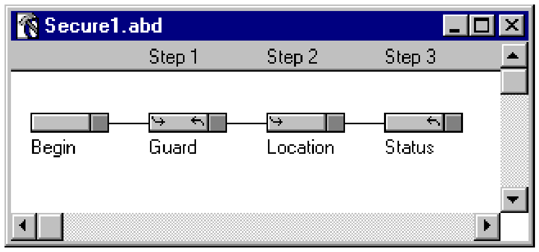


Figure 1-1 Security Application

Each box in the window (Begin, Guard, Location, and Status) is a stop along the path. The boxes are called **Input Handlers**. The lines between the boxes indicate where the application is allowed to go.

When creating an application program, there needs to be <u>something unique about the bar codes or entry for each Input Handler</u>; this allows the data collector to know that the criterion has been met to advance to the next Input Handler.

The application moves from one Input Handler to another in response to input from the user. Input may be in the form of a bar code scan, a button touch, or a keypad entry. There is always one Input Handler that is the "**current**" **Handler**. The "current" Handler defines what the display looks like, what type of data is acceptable, what data gets added to the current record, and so on.

The "Begin" Input Handler (the first box on the left) is automatically placed as the first Input Handler whenever a new application is opened; it cannot be moved or deleted, but it can be renamed. The "Begin" Handler is always the "current" Handler when you start an application. When the user enters data, the application attempts to move to another Input Handler (that is, tries to make another Input Handler the "current" Handler). It generally tries to take one step to the right in response to user input.

For example, in our security example, the application (Secure1.abd) has Input Handlers for Guard, Location, and Status. The application is designed to expect the following types of entries to meet the match criteria of the Input Handlers:

Guard:	Entry must start with a **G** followed by three digits
Location:	Entry must start with a **D**
Status:	Entry must be contained in the cross-reference file **Scstat.crf**. **Scstat.crf** is a cross-reference file in the **Samples** folder that contains five records (S91 = Locked, S92 = Unlocked, S93 = Called Police, S94 = Passed, S95 = Maintenance Required)

When the application starts, "Begin" is the "current" Handler. When the user makes an entry, the application steps to the right and asks the "Guard" Handler if it will accept the user input. If the user input matches the "Guard" Handler match criterion (starts with **G** followed by three digits), the entry is accepted and the "Guard" Handler becomes the "current" Handler.

When the next entry is made, the application again steps to the right and asks the “Location” Handler if it will accept the user input. If the user input matches the “Location” Handler criterion (starts with **D**), the “Location” Handler accepts the entry and becomes the “current” Handler. When the next entry is made, the application again steps to the right and asks the “Status” Handler if it will accept the user input. If the user input is acceptable (contained in Scstat.crf), the “Status” Handler accepts the entry and it becomes the “current” Handler.

Now, when the “Status” Handler is the “current” Handler and another entry is made, the application can no longer step to the right, so it loops back a column and asks the “Location” Handler if it will accept the user input. If, for example, the entry were G234, the “Location” Handler would reject the entry. The application would then loop back another column and see if the “Guard” Handler would accept the entry. Since the entry matches the “Guard” Handler criterion, the “Guard” Handler accepts the entry and becomes the “current” Handler.

Another security application might look like Figure 1-2.

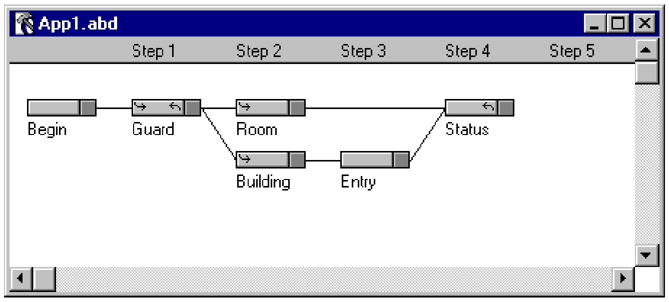


Figure 1-2 Security Example

In this application, suppose that “Guard” Handler is the “current” Handler, and the user scans the bar code B204. The application steps to the right and asks the “Room” Handler if it will accept the user’s input, B204. If the “Room” Handler rejects the input, the application then looks to see if there are any other Input Handlers in the same column. It locates the “Building” Handler and asks if it will accept the user input, B204. If the “Building” Handler accepts the input, then it becomes the “current” Handler. The application in Figure 1-2 would then require user input for the “Entry” Handler and then the “Status” Handler.

If the application cannot move to the right in response to user input, it attempts to loop back to the left. The basic idea behind looping is that the application steps back through the Input Handlers to see if any of them will accept the user input.

In order for a loop to occur, the “current” Handler must be allowed to end a loop, and the Input Handler accepting the user input must be allowed to start a loop. In Figure 1-2, the Input Handlers “Guard,” “Room,” and “Building” can start a loop (indicated by a curved arrow on the left side) and the Input Handlers “Guard” and “Status” can end a loop (indicated by a curved arrow on the right side). It would be possible to loop back from “Status” to “Building,” “Room,” or “Guard.” It would also be possible to loop back from “Guard” to itself.

When a loop occurs, the current record is complete, and it is committed to the data file. After the loop is finished, there is a new current record in memory ready to be constructed. If the loop did not go all the way back to Step 1, the new current record will already be partially constructed; it will contain the data previously entered up to the start of the loop.

Following is a specific example using the application in Figure 1-2.

Example

Start State: Current Handler: Begin
Current Record: empty

User Enters: G101

Application Response: “Guard” Handler accepts the user input and writes the input, date, and time to the current record.

State: Current Handler: Guard
Current Record: “G101”, “06/26/96”, “4:01 PM”

User Enters: B204

Application Response: Application steps to right and asks “Room” Handler if it will accept input; “Room” Handler refuses. The application looks for any other Input Handlers in the same column and asks the

"Building" Handler if it will accept the user input. "Building" Handler accepts the input and writes it to the current record.

State: Current Handler: Building
Current Record: "G101", "06/26/96", "4:01 PM", "B204"

User Enters: E34

Application response: "Entry" Handler accepts the user input and writes the input to the current record.

State: Current Handler: Entry
Current Record: "G101", "06/26/96", "4:01 PM", "B204", "E34"

User Enters: S91

Application Response: "Status" Handler accepts the user input and writes the input to the current record.

State: Current Handler: Status
Current Record: "G101", "06/26/96", "4:01 PM", "B204", "E34", "S91"

User Enters: R19

Application Response: Since it cannot step forward, the application tries to loop back. It first loops back to the Step 3 column to see if any Input Handlers in this column can start a loop. Since they cannot, the application loops back to the Step 2 column and looks for any Input Handlers that can start a loop. Both the "Room" and "Building" Handlers in this column can start a loop, but the application would ask the "Room" Handler first since it is at the top of the column. The "Room" Handler accepts the input and the application loops back to this Handler. When the application loops back to the left, the current record is committed to the data file and a new current record is begun.

State: Current Handler: Room
Current Record: "G101", "06/26/96", "4:01 PM", "R19"

Notice that when the application looped back, it assumed that the new current record should still have the same data as the previous record, up to the "Room" Handler (which started the loop).

Communication with the Computer

DuraTrax, LaserLite, LaserLite Pro, and LaserLite Mx data collectors communicate with the computer by means of an IR link, utilizing IrDA® technology. The data collector’s IR link is the IR transmitter on the end cap. The computer’s IR link may be either a Videx Base Station, or an external IrDA transceiver attached to your computer’s serial port. The data collectors establish two-way communication with the host computer using these IR links.

To communicate with the computer using a Base Station, insert the data collector into one of the Base Station’s slots (Figure 1-3). (Note: See your Hardware manual for information on connecting the Base Station to your computer. The Base Station should be placed *at least* three inches away from your computer and monitor to prevent high levels of electromagnetic interference from hindering communications.)

The DuraTrax/LaserLite Base Station has two slots; you may use either slot. The LaserLite Pro/Mx Base Station has one slot. If you are using more than one data collector, refer to Appendix C for information on using multiple units.

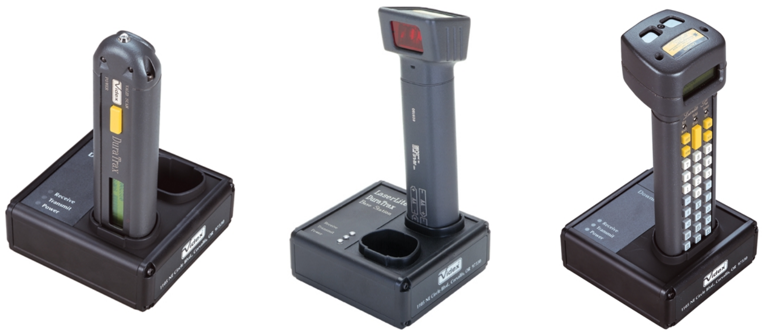


Figure 1-3 Videx Base Stations

To communicate with a computer using an external IR station or a built-in IrDA transceiver, point the data collector's IR transmitter (located in the end cap) towards the computer's IR station. Space the data collector approximately four inches from the IR device during communication. (The optimum range for communication is approximately 3–12 inches. The range varies depending upon the type of IR station used and the ambient light conditions.)

Use the Application Builder software or one of the download programs for communication.

Quick Start

The following directions step you through:

- Installing the files from the Application Builder CD to the computer.
- Using Application Builder to transfer a sample application to the data collector.
- Using the sample application to collect data with the data collector.
- Transferring the data from the data collector to the computer.
- Viewing the application program's data file.

Install IR Device and Application Software

1. Shut down your computer and install an external IR station or a Base Station to your computer's serial port. Consult the external IR station manual for instructions on installing an external IR station (Caution: If using a JetEye, do not install the JetEye drivers included with the JetEye). See your data collector's hardware manual for instructions on installing a Base Station.

2. Restart your computer and insert the Application Builder CD into your computer's CD drive.

3. The CD will automatically open; follow the on-screen instructions to transfer the files from the Application Builder CD to your computer.

Open an Application

4. Choose **Programs> Videx> Videx Application Builder** from the Windows **Start** menu or double-click the **Appbuild.exe** icon (Figure 1-4) in the **Videx** folder to run the Application Builder program.

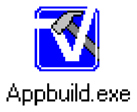


Figure 1-4 Appbuild.exe Icon

5. The Application Builder menu titles and toolbar appear (Figure 1-5). The menu commands and toolbar buttons are described in detail in Chapter 2.

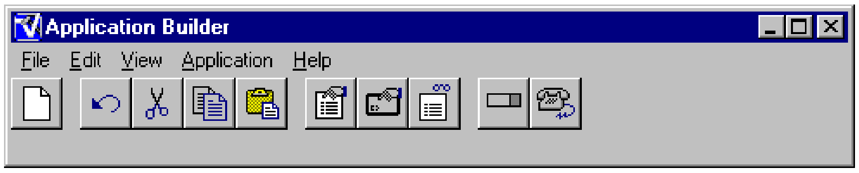


Figure 1-5 Application Builder Menu Titles and Toolbar

6. Choose **Open** from the **File** menu (Figure 1-6) or type **Ctrl+O**.

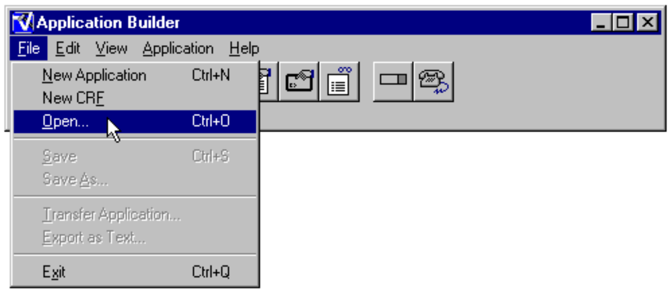


Figure 1-6 File Menu – Open Application

7. Open the **Secure1.abd** document located in the **Samples** folder (Figure 1-7).

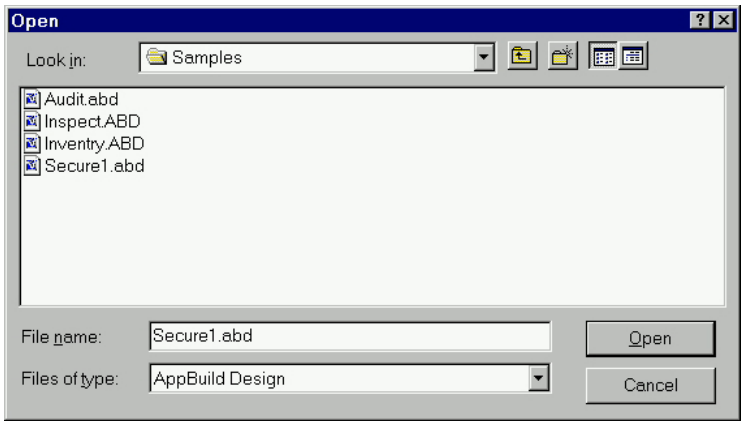


Figure 1-7 Open Secure1.abd

8. The sample application is displayed (Figure 1-8). The application is a three-step security application named **Secure1.abd**.

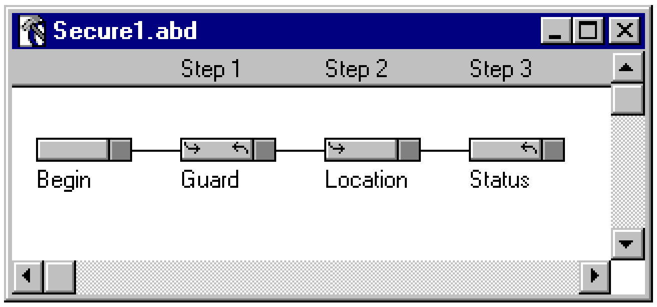


Figure 1-8 Secure1.abd Application

This application contains Input Handlers for Guard, Location, and Status. The application is designed to expect the following types of user input to meet the match criteria for the Input Handlers:

Guard:	User input must start with the letter **G** followed by three digits
Location:	User input must start with the letter **D**
Status:	User input must be contained in the cross-reference file **Scstat.crf**. **Scstat.crf** is a cross-reference file in the **Samples** folder that contains five records: S91 = Locked, S92 = Unlocked, S93 = Called Police, S94 = Passed, S95 = Maintenance Required

Pages 19–20 or page 194 contain bar codes for the application. Chapter 3 steps you through building a similar application.

9. Insert your data collector into a Base Station or point the data collector’s IR transmitter (located in the end cap) towards the computer’s IR station. The distance between the data collector and the IR station should be approximately four inches.

Transfer an Application

10. Click the **Transfer Application** button on the toolbar (Figure 1-9) or choose **Transfer Application** from the **File** menu (Figure 1-10).

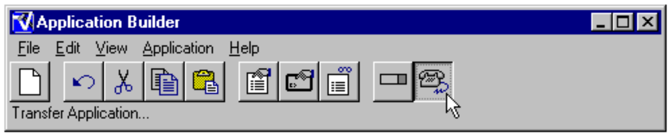


Figure 1-9 Application Builder Toolbar – Transfer Application

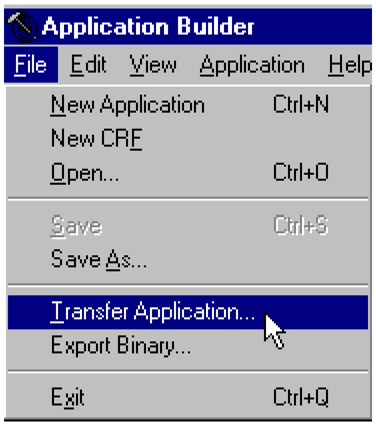
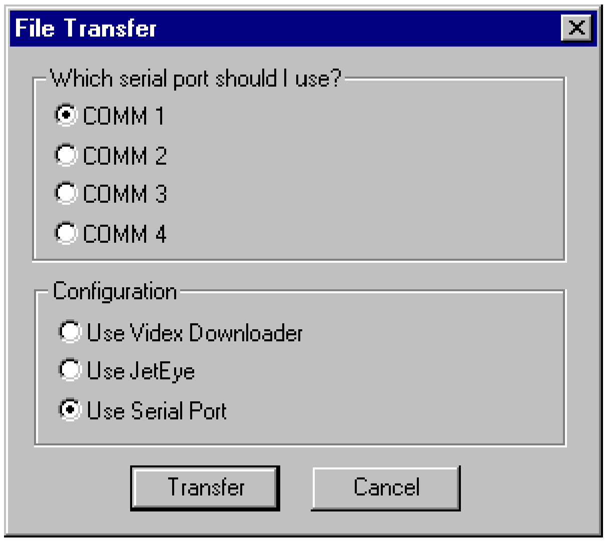


Figure 1-10 File Menu – Transfer Application

11. The **File Transfer** window (Figure 1-11) is displayed. If you are using a Base Station, select the serial port it is connected to and the **Use Videx Downloader** configuration. If you are using an external IR station, select the serial port it is connected to and the **Use Serial Port** configuration.


File Transfer

Which serial port should I use?

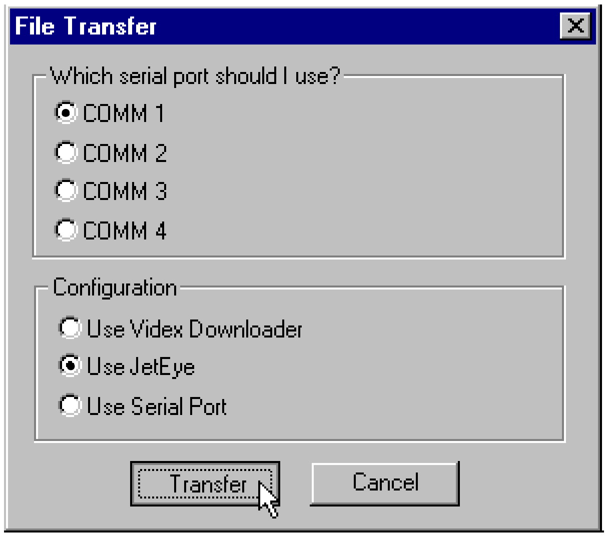
● COMM 1
○ COMM 2
○ COMM 3
○ COMM 4

Configuration

○ Use Videx Downloader
○ Use JetEye
● Use Serial Port

Figure 1-11 File Transfer Window

12. Click **Transfer** (Figure 1-12).


File Transfer

Which serial port should I use?

- (●) COMM 1
- ( ) COMM 2
- ( ) COMM 3
- ( ) COMM 4

Configuration

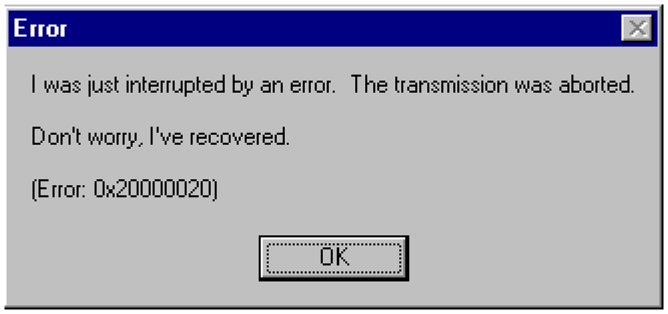
- ( ) Use Videx Downloader
- (●) Use JetEye
- ( ) Use Serial Port

Figure 1-12 File Transfer Window

13. Press any key on the data collector; the application and its cross-reference file transfer to the unit.

14. A transfer progress message reports the status of the transfer. If the transfer is successful, go to Step 16; if the transfer is not successful, go to Step 15.

15. Figure 1-13 shows the error message that is displayed if the file does not transfer correctly. To clear the message, press <Enter> or click **OK**.


Error

I was just interrupted by an error. The transmission was aborted.

Don't worry, I've recovered.

(Error: 0x20000020)

Figure 1-13 Transfer Not Successful Message

Try these troubleshooting tips if the application does not transfer:

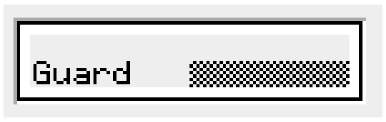
- Are the correct computer serial port and IR device configuration selected at the **File Transfer** window?
- Is the Base Station or IR Station correctly connected to your computer?
- If using a Base Station, is the station at least 3 inches away from the computer and monitor?
- If using a Base Station, reset it by unplugging it from the electric outlet and plugging it back in again.
- If using an external IR station, is the data collector's IR transmitter (located in the end cap) pointing towards the IR station at a distance of approximately four inches?

Reattempt to transfer the application; if you are still not able to transfer the application, contact the Videx Technical Support Department for assistance.

Use the Application

16. To run the sample application, press any button on the data collector. You will hear an upscale tone indicating an application is installed and is starting to run.

17. The data collector displays the beginning display for the application (Figure 1-14).



```
Guard   ▒▒▒▒▒▒▒▒
```

Figure 1-14 Secure1 Application Begin Display

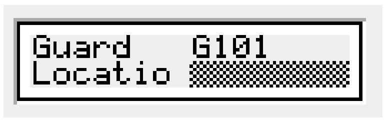
18. Scan the guard ID bar code in Figure 1-15.

G101

G101

Figure 1-15 Guard ID Bar Code

19. The display shows the guard ID and asks for the location (Figure 1-16). *(Note: **Location** appears as **Locatio** in the display because the names are truncated to the first seven characters if Application Builder generates the displays. We will show you how to edit the display in Chapters 2 and 3.)*



```
Guard   G101
Locatio ▒▒▒▒▒▒▒▒
```

Figure 1-16 Secure1 Application Display

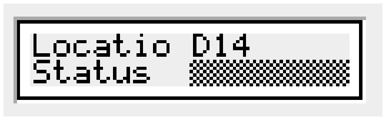
20. Scan one of the location bar codes in Figure 1-17.

D14　　　*D17*

D14　　　*D17*

Figure 1-17 Location Bar Codes

21. The display shows the location and asks for the status of the location (Figure 1-18).



```
Locatio D14
Status  ▒▒▒▒▒▒▒▒▒▒
```

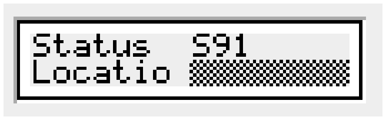
Figure 1-18 Secure1 Application Display

22. Scan one of the status bar codes in Figure 1-19.

S91	Locked
S92	Unlocked
S93	Called Police
S94	Passed
S95	Maintenance Required

Figure 1-19 Status Bar Codes

23. The display shows the status and asks for the next location (Figure 1-20).



```
Status  S91
Locatio
```

Figure 1-20 Secure1 Application Display

24. Continue to enter locations and status, or enter a new guard ID. If you wish to delete a scan, scan the bar code in Figure 1-21. The bar code in Figure 1-21 is composed of five spaces.

*   *

Figure 1-21 Delete Scan Bar Code

25. The data is saved in the unit's memory. To view the data, it must be transferred to the computer.

Transfer the Data File to the Computer with Vxcom

26. Unless you are using a Base Station attached to com port 1, you must set the parameters for **Vxcom** before transferring the file to the computer for the first time. To set the parameters for **Vxcom**, choose the **Run** command from the Windows **Start** menu.

27. Click **Browse** at the **Run** window (Figure 1-22).

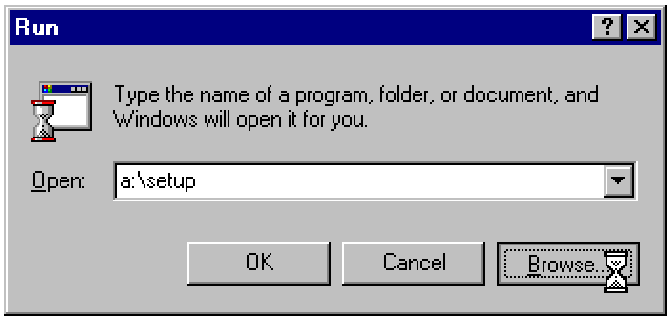

Run
Type the name of a program, folder, or document, and Windows will open it for you.
Open: a:\setup
OK
Cancel
Browse...

Figure 1-22 Run Window - Click Browse

28. Locate and open the **Vxcom** program. The **Vxcom** filename is **Vxcomm.exe**. If you used the Setup defaults while installing the software, the **Vxcom** program should be located in the **Videx** folder.

29. Add the parameters to the end of the path name (Figure 1-23).

    The syntax is: [**-p**<port number>]

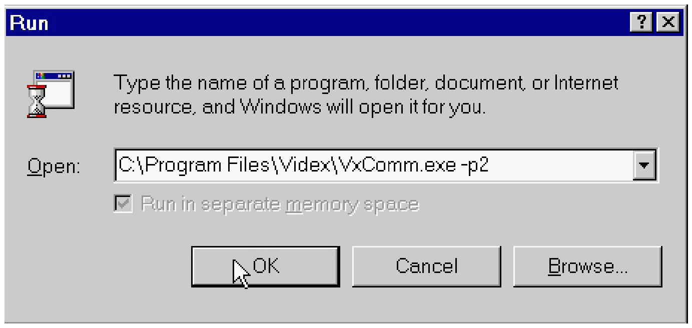

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open: C:\Program Files\Videx\VxComm.exe -p2
Run in separate memory space
OK
Cancel
Browse...

Figure 1-23 Add Vxcom Parameters

The first parameter (**-p**) is the serial port; the program uses 1 by default. If you are using serial port 1, you do not need to enter this parameter. If you are using serial ports 2, 3, or 4, you must enter a **-p** followed by the serial port number (**-p2** for serial port 2, **-p3** for serial port 3, **-p4** for serial port 4).

30. Click **OK** to close the **Run** window and save the parameter entry. (Note: The transfer parameters only need to be entered the first time **Vxcom** is run. You do not need to enter these parameters again unless the settings are changed.)

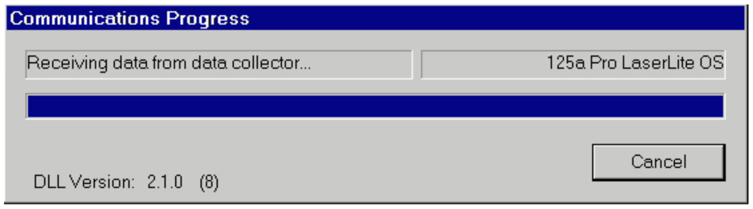
31. Insert the data collector into the Base Station or point the data collector's IR transmitter (located in the end cap) towards the external IR transceiver.

32. Double-click the **Vxcom** icon (Figure 1-24).

Figure 1-24 Vxcom Icon

33. The program begins looking for a data collector. Press any key on the data collector to begin the transfer.

34. A window opens during the data transfer process (Figure 1-25) and reports the progress of the transfer.



Communications Progress

Receiving data from data collector...

125a Pro LaserLite OS

DLL Version: 2.1.0 (8)

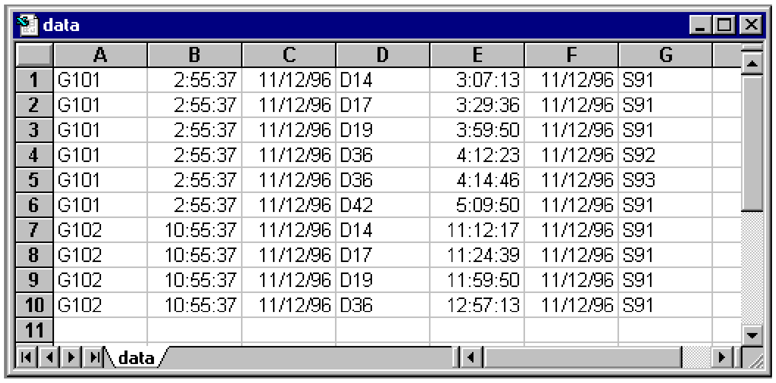
Figure 1-25 Communications Progress Window

View the Data File

35. The file is transferred to the **Videx** folder and saved as a text file named **data.txt** (by default). Double-click the **data.txt** icon (Figure 1-26) to view the data.

Figure 1-26 Data.txt Icon

36. You can also use an application program to open the data file (such as a word processing, spreadsheet, or database program). Figure 1-27 shows an example of the file displayed in Excel™. The file will differ in appearance depending on the date, time, and bar codes scanned.



	A	B	C	D	E	F	G
1	G101	2:55:37	11/12/96	D14	3:07:13	11/12/96	S91
2	G101	2:55:37	11/12/96	D17	3:29:36	11/12/96	S91
3	G101	2:55:37	11/12/96	D19	3:59:50	11/12/96	S91
4	G101	2:55:37	11/12/96	D36	4:12:23	11/12/96	S92
5	G101	2:55:37	11/12/96	D36	4:14:46	11/12/96	S93
6	G101	2:55:37	11/12/96	D42	5:09:50	11/12/96	S91
7	G102	10:55:37	11/12/96	D14	11:12:17	11/12/96	S91
8	G102	10:55:37	11/12/96	D17	11:24:39	11/12/96	S91
9	G102	10:55:37	11/12/96	D19	11:59:50	11/12/96	S91
10	G102	10:55:37	11/12/96	D36	12:57:13	11/12/96	S91
11							

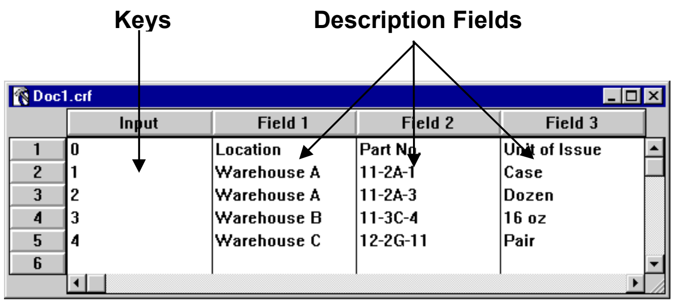
Figure 1-27 Sample Data File

This completes the Application Builder quick start. The next section is a quick start for creating a cross-reference (CRF) file.

Cross-Reference File Quick Start

Discussion

A cross-reference (CRF) file can be used to verify input and make entries to the data file. A CRF file is a list of keys and their corresponding description fields. A *key* represents the user input. A description field describes the key. You can use one or more description fields to describe a key.



	Input	Field 1	Field 2	Field 3
1	0	Location	Part No	Unit of Issue
2	1	Warehouse A	11-2A-1	Case
3	2	Warehouse A	11-2A-3	Dozen
4	3	Warehouse B	11-3C-4	16 oz
5	4	Warehouse C	12-2G-11	Pair
6				

Figure 1-28 Sample CRF File

If an Input Handler's **Matching** property specifies that it accepts input from a CRF file, the user input is compared to the keys in the CRF file. If the user input matches a key, the user input is accepted; either the user input or one of the matching key's description fields can be added to the data file.

A CRF file with a key field and up to three description fields can be created with the **File** menu's **New CRF** command. The following quick-start section shows you how to use the **New CRF** command to create a CRF file. See pages 88–90 for complete information on CRF files.

Quick Start

In this section, we will use the **New CRF** command to build a CRF file of bar codes and corresponding location descriptions. We will use the following information to build the CRF file:

Bar Code	Location
D14	Library
D17	Cafeteria
D26	Administration
D34	Accounting
D47	R&D Lab

1. Choose **New CRF** from the **File** menu (Figure 1-29).

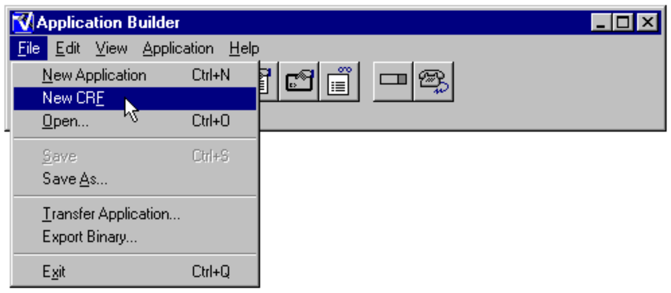


Figure 1-29 File Menu - New CRF Command

2. A new CRF window (Figure 1-30) opens. Keys are entered in the Input column and descriptions are entered in the Field columns. In this example, we will use one description field per key.

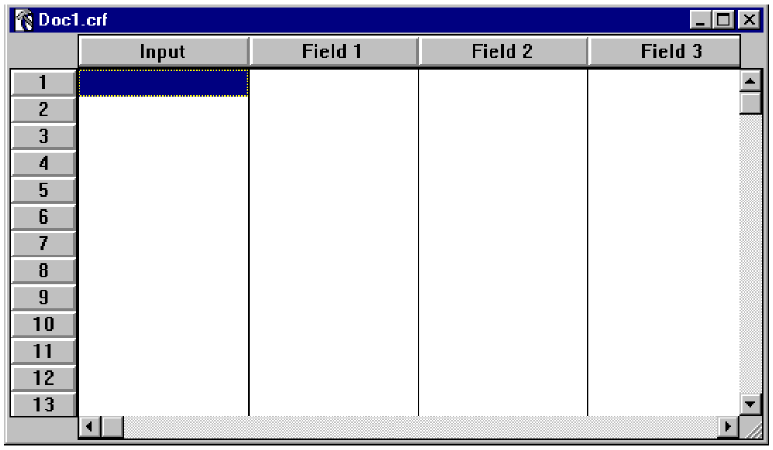
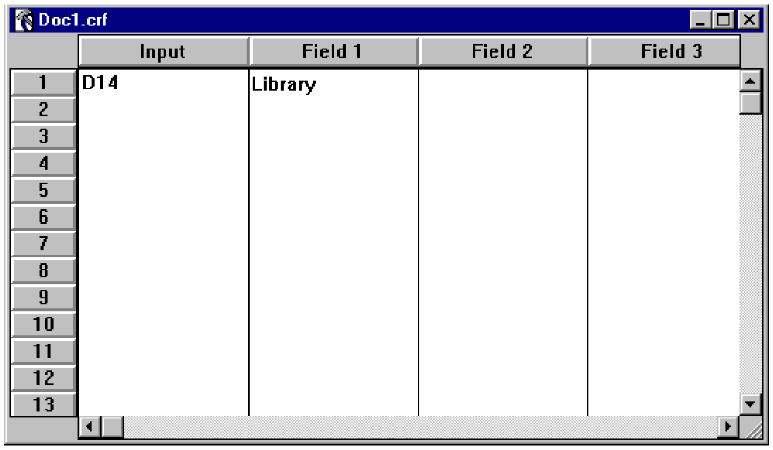


Figure 1-30 CRF Window

To insert or delete records in a CRF file, choose **Insert Record** or **Delete Record** from the **Edit** menu. If you are in the last cell of the CRF file (Field 3 of the last record) and you press the <Tab> key, a new record is automatically inserted at the end.

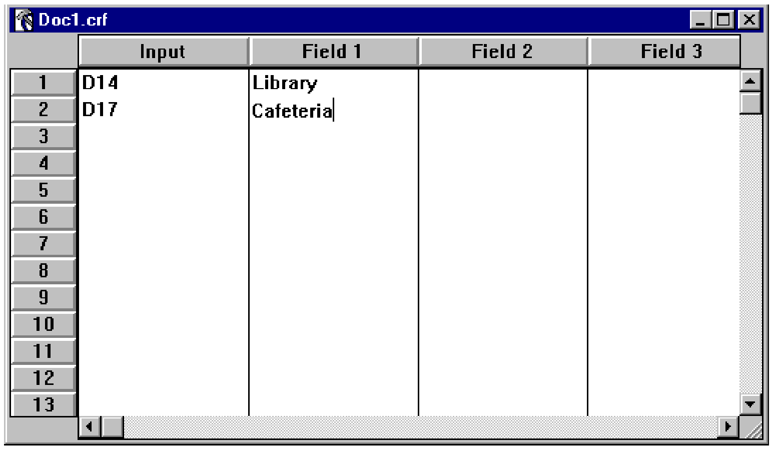
3. Click the first cell in the Input column. The cursor appears; type **D14**. Press the <Tab> key to move the cursor to the Field 1 cell and type **Library** (Figure 1-31).



	Input	Field 1	Field 2	Field 3
1	D14	Library		
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				

Figure 1-31 Enter First Key and Description

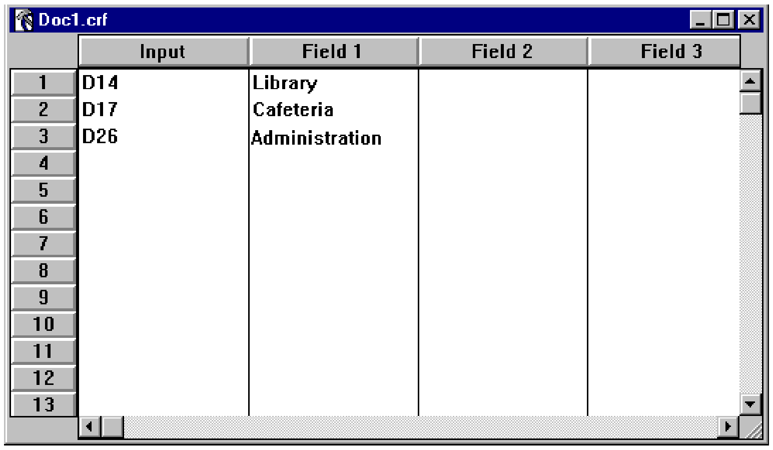
4. Go to the second cell in the Input column by clicking the cell, or by pressing the <Tab> key three times to move the cursor through Fields 2, 3, and back to Input. Type **D17**, press <Tab>, and type **Cafeteria** (Figure 1-32).



	Input	Field 1	Field 2	Field 3
1	D14	Library		
2	D17	Cafeteria		
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				

Figure 1-32 Enter Second Key and Description

5. Go to the third cell in the Input column by clicking the cell, or by pressing the <Tab> key until the cursor reaches the cell. Type **D26**, press <Tab>, and type **Administration** (Figure 1-33).

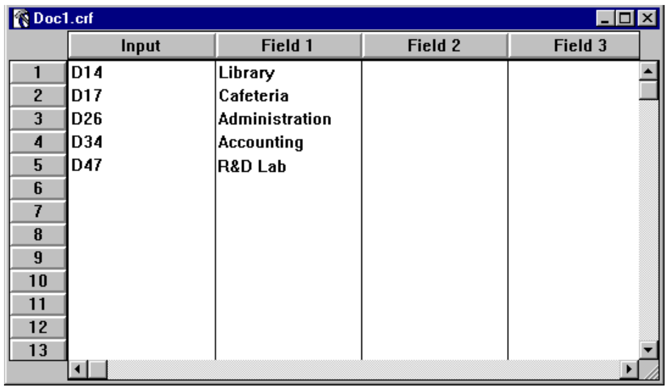


	Input	Field 1	Field 2	Field 3
1	D14	Library		
2	D17	Cafeteria		
3	D26	Administration		
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				

Figure 1-33 Enter Third Key and Description

6. Move the cursor to the fourth cell in the Input column. Type **D34**, press <Tab>, and type **Accounting**.

7. Move the cursor to the fifth cell in the Input column. Type **D47**, press <Tab>, and type **R&D Lab** (Figure 1-34).



	Input	Field 1	Field 2	Field 3
1	D14	Library		
2	D17	Cafeteria		
3	D26	Administration		
4	D34	Accounting		
5	D47	R&D Lab		
6				
7				
8				
9				
10				
11				
12				
13				

Figure 1-34 CRF File - Keys and Descriptions Entered

8. Choose **Save As** from the **File** menu (Figure 1-35).

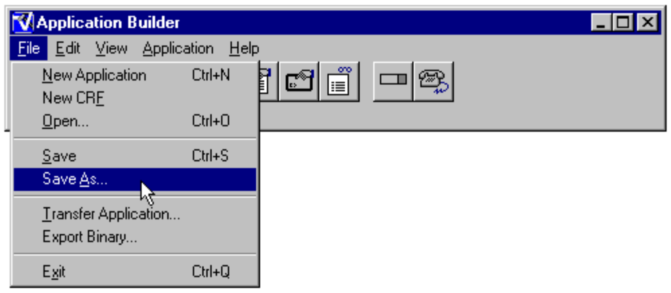
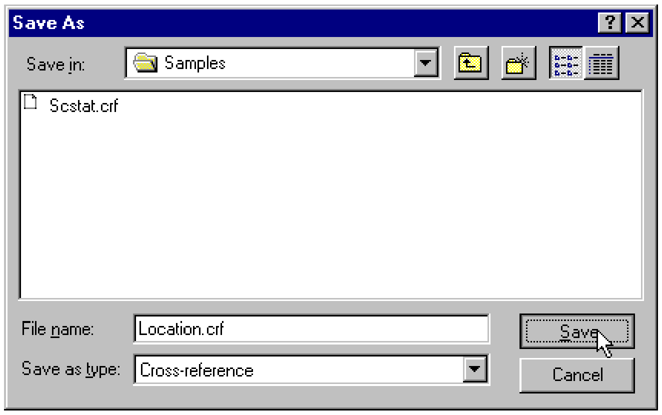


Figure 1-35 File Menu - Save As Command

9. Navigate to the folder you want to save the file in (we are using the **Samples** folder). Name the file **Location.crf** and click **Save** (Figure 1-36).

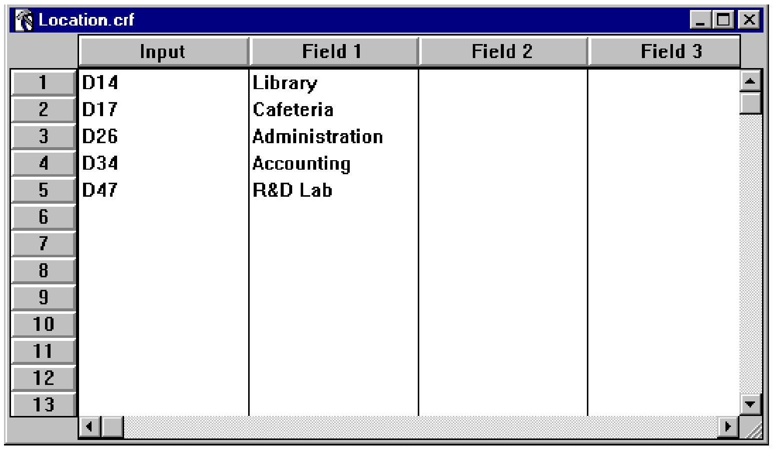


File name: Location.crf

Save as type: Cross-reference

Figure 1-36 Name and Save CRF File

10. The CRF file you have created is saved as **Location.crf** (Figure 1-37). *(Note: The* ***.crf*** *extension is automatically added to the end of the filename when the file is saved.)*



	Input	Field 1	Field 2	Field 3
1	D14	Library		
2	D17	Cafeteria		
3	D26	Administration		
4	D34	Accounting		
5	D47	R&D Lab		
6				
7				
8				
9				
10				
11				
12				
13				

Figure 1-37 Location.crf CRF File

11. Click the close window box to close the CRF window.

This completes the quick-start sections. Chapter 2 describes the Application Builder toolbar buttons, menu commands, and windows. Chapter 3 steps through the process of building an application, including using the CRF file you just created (*Chapter 3, step 86*).

Chapter 2

Application Builder — Menus & Windows

This chapter contains:

- Descriptions of the Application Builder menu commands
- Descriptions of the Application Builder windows
- Additional notes on using Application Builder

Application Builder Opening Screen

When you first open Application Builder, its menu titles and toolbar are displayed.

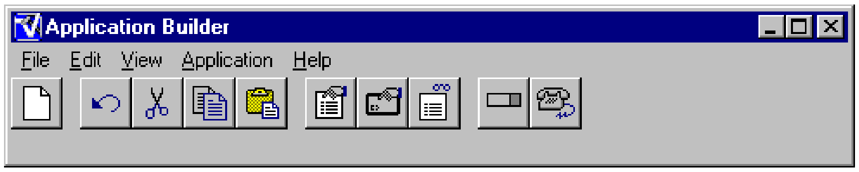


Figure 2-1 Application Menu Titles and Toolbar

The menus **File**, **Edit**, **View**, **Application**, and **Help** contain commands for building the application. The toolbar provides quick access to the most commonly used menu commands.

Toolbar Buttons

Following are the toolbar buttons and their functions:

New Application Button

Opens new application design window. This window is where you build the application; when it first opens it contains a "Begin" Handler and 64 steps or columns.

Undo Button

Undoes last action. *(Ten levels of undo.)*

Cut Button

Removes the selection from the screen and copies it to the clipboard.

Copy Button
Copies the selection to the clipboard; the selection remains on the screen.

Paste Button
Pastes the clipboard contents to the screen.

Properties Window Button
Opens the **Properties** window. The **Properties** window defines the Input Handlers located in the design window. The **Properties** window is discussed on pages 51–69.

Display Window Button
Opens the **Display** window. The **Display** window is where the data collector's LCD display is designed for applications. The **Display** window is discussed on pages 70–83.

Summary Window Button
Opens the **Summary** window. The **Summary** window summarizes the properties of the selected Input Handler. The **Summary** window is discussed on pages 84–87.

New Input Handler Button
Adds a new Input Handler to the application.

Transfer Application Button
Opens the **File Transfer** window and transfers the current application and its CRF files to the data collector.

The following sections list the menu commands and their functions.

File Menu

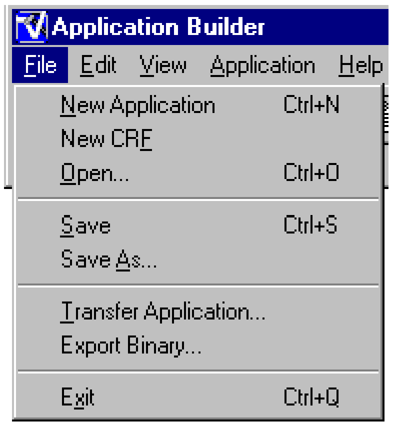


Figure 2-2 File Menu

The **File** menu (Figure 2-2):

- Opens a new application design window
- Opens a new cross-reference (CRF) file window
- Opens an existing application or CRF file
- Saves the current application or CRF file
- Transfers the current application, along with its CRF files, to the data collector
- Saves the application as a BASIC source code file
- Quits the program

Following is a list of the Application Builder's **File** menu commands and their functions. If the command has a keyboard equivalent, it is listed within parentheses next to the command name.

New Application (Ctrl+N)

Opens a new untitled design window.

New CRF

Opens a new untitled CRF window and allows you to create a CRF file. A quick-start section for this command begins on page 26. CRF files are discussed on pages 26–31 and 88–90.

Open… (Ctrl+O)

Opens the **Open** dialog box (Figure 2-3) and allows you to open an existing application or cross-reference file. To select a file, you can either click the file or you can enter the file's name in the **File name** area. To open the file, click **Open** or press <Enter>. Applications are displayed in an application window; cross-reference and any other files are displayed in a CRF window.

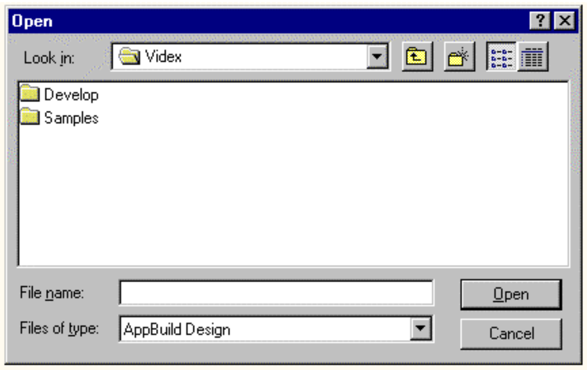


Figure 2-3 Open Dialog Box

Towards the bottom of the **Open** dialog box is a **Files of type** pop-up menu (Figure 2-4). This menu determines the types of files that are displayed.

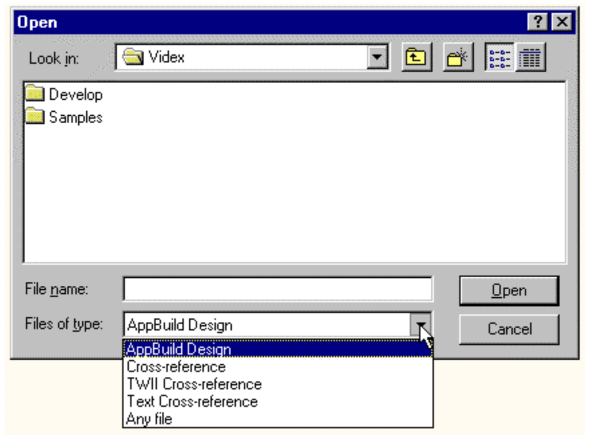


Figure 2-4 Open Dialog Box - Files of Type Pop-up Menu

By default, whenever the **Open** dialog box appears, **AppBuild Design** is the selected file type and only folders and applications with a **.abd** extension are displayed. If you wish to display other files, you must go to the pop-up menu and select one of the cross-reference file choices or **Any file**.

The **Files of type** pop-up menu choices are:

Pop-up Menu Choice	**Function**
AppBuild Design (default)	Displays any applications created with Application Builder (*.abd extension*). The file opens in a titled application design window.
Cross-reference	Displays any CRF files created with Application Builder (*.crf extension*). The file opens in a titled CRF window.
TWII Cross-reference	Displays any TimeWand II cross-reference files (*.xrf extension*). The file opens in an untitled CRF window.
Text Cross-reference	Displays any text files (*.txt extension*). The file opens in an untitled CRF window.
Any file	Displays all of the files.

Save (Ctrl+S)

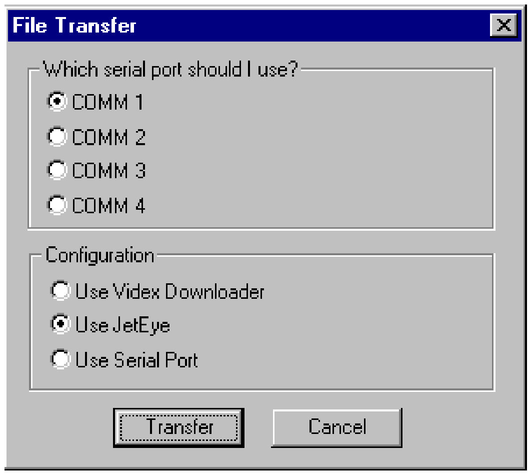
Saves the current application or CRF file. Documents keep track of any changes made to them; their status affects whether the **Save** menu command is enabled.

Save As…

Names and saves the current application or CRF file.

Transfer Application…

Displays the **File Transfer** window (Figure 2-5) and transfers the current application and its CRF files to the data collector.



File Transfer

Which serial port should I use?

[✓ COMM 1] COMM 2 COMM 3 COMM 4

Configuration

Use Videx Downloader [✓ Use JetEye] Use Serial Port

Figure 2-5 File Transfer Window

The **File Transfer** window indicates which computer serial port and download configuration are being used. Before transferring an application, verify that the settings are correct and click **Transfer**. Click **Cancel** to close the **File Transfer** window without transferring the application.

Export Binary…

This command is intended for BASIC programmers and developers and assumes the user has a fundamental knowledge of BASIC.

This command saves the created application as a BASIC source file (**.B** extension) and a compiled application file (**.S** extension).

Changes can be made to the BASIC source file with any editor; however, if any changes are made to the source file it must be recompiled into an application with **Vxbasicw.exe** or **Vxbasic.exe**. See the *Videx BASIC Manual* for information on using **Vxbasicw.exe** and **Vxbasic.exe** to compile a BASIC source file.

If you plan to make changes to the BASIC source file, you can save only the BASIC source file by holding down the <Ctrl> key before accessing the **File** menu. The command appears as **Export Source Only** in the menu and only the BASIC source file (**.B** file) is saved. Remember that the BASIC source file must be compiled into a **.S** file with either **Vxbasicw.exe** or **Vxbasic.exe** before it can be used in a data collector.

See the *Developer's Reference Manual* for more information on the source file. See the *Videx BASIC Manual* for information on the BASIC commands.

Exit (Ctrl+Q)

Quits the program.

Edit Menu

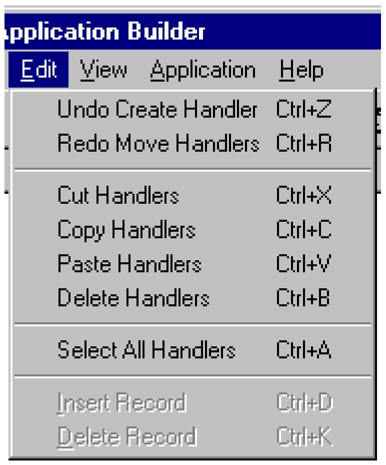


Figure 2-6 Edit Menu

The **Edit** menu (Figure 2-6):

- Undoes the last action
- Redoes the last undo
- Cuts, copies, pastes, and deletes items
- Selects all Handlers
- Inserts CRF record
- Deletes CRF record

Following is a list of the Application Builder's **Edit** menu commands and their purpose. If the command has a keyboard equivalent, it is listed within parentheses next to the command name.

Undo (Ctrl+Z)

Undoes the last action. (Ten levels of undo.)

Redo (Ctrl+R)

Redoes the last undo. (Ten levels of redo.)

Cut (Ctrl+X)

Removes the selection from the screen and copies it to the clipboard.

Copy (Ctrl+C)
Copies the selection to the clipboard; the selection remains on the screen.

Paste (Ctrl+V)
Pastes the clipboard contents to the screen.

If a paste operation includes a “Begin” Handler, it completely replaces any existing “Begin” Handler. All connections between copied Handlers are retained for pasting. Handlers are always pasted in the same column from which they were copied. Application Builder attempts to paste them in the same horizontal position, but will automatically move them down to avoid pasting one Handler on top of another.

Display objects are copied from the **Display** window, but pasted into all selected Handlers. This makes it possible to paste a particular display object into several Handlers with dissimilar displays. If the **Display** window is not open when display objects are pasted, it is automatically opened.

Delete (Ctrl+B)
Deletes the selection from the screen. This command can be used to delete Handlers and their names from the design window.

Select All Handlers (Ctrl+A)
Selects all of the Handlers in the application design window.

Insert Record (Ctrl+D)
Inserts a record into the CRF file. The new record is inserted before the currently selected record.

Delete Record (Ctrl+K)
Deletes the currently selected record from the CRF file.

View Menu

The **View** menu (Figure 2-7):

- Shows or hides the Detour connections
- Opens the **Properties** window
- Opens the **Display** window
- Opens the **Summary** window
- Lists the currently open design windows and identifies the active window with a checkmark

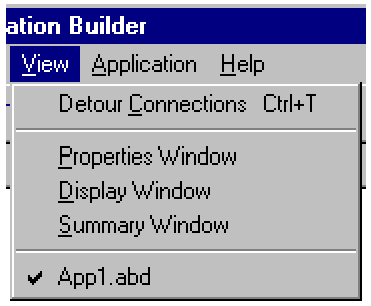


Figure 2-7 View Menu

Following is a list of the **View** menu commands and their purpose. If the command has a keyboard equivalent, it is listed within parentheses next to the command name.

Detour Connections (Ctrl+T)

Shows or hides the connections between Detour Handlers and Input Handlers. A checkmark before the command indicates detour connections are shown; no checkmark indicates they are hidden. Detour Handlers are discussed on page 69.

Properties Window

Opens the **Properties** window. A checkmark indicates the window is open; no checkmark indicates the window is closed. The **Properties** window is discussed on pages 51–69.

Display Window

Opens the **Display** window. A checkmark indicates the window is open; no checkmark indicates the window is closed. The **Display** window is discussed on pages 70–83.

Summary Window

Opens the **Summary** window. A checkmark indicates the window is open; no checkmark indicates the window is closed. The **Summary** window is discussed on pages 84–87.

<Application name>.abd

List of open design windows; there is a checkmark before the active window.

Application Menu

The **Application** menu (Figure 2-8):

- Adds a new Input Handler
- Displays the **Data File Format** window
- Displays the **Other Details** window
- Generates displays for the application

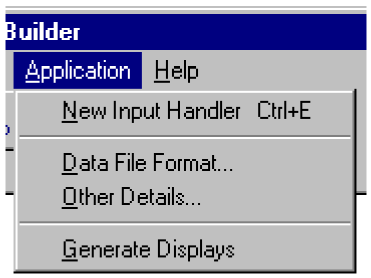


Figure 2-8 Application Menu

Following is a list of the **Application** menu commands and their purpose. If the command has a keyboard equivalent, it is listed in parentheses next to the command name.

New Input Handler (Ctrl+E)

Adds a new Input Handler to the application. If a Handler is not selected when you choose **New Input Handler**, the new Input Handler is added to the end of the application. If any Handlers are selected when the command is chosen, a connection is automatically made from all of the selected Handlers to the new Input Handler (if all connections are legal).

Working with Input Handlers

Connecting Handlers

A connection can only be made from an Input Handler to another Input Handler that is at least one step to the right, but not to an Input Handler in the same or preceding column.

You can connect an Input Handler to a Detour Handler (as long as the Detour Handler is at least one step to the right), but a connection cannot be made from a Detour Handler to an Input Handler.

To make or remove a connection between two Handlers, click the first Handler's gray box and drag a line to the other Handler. If the two Handlers are not connected, clicking and dragging a line between them makes a connection (if the connection is legal). If the two Handlers are already connected, clicking and dragging a line between them removes the connection.

To make all legal connections to a Handler, hold down the <Ctrl> key and click a Handler. All legal connections between that Handler and the selected Handlers are made.

To delete multiple connections to a Handler, first select the Handlers that are connected. To select multiple Handlers, either Shift-click each Handler or use the mouse to drag a rectangle around the Handlers. Then hold down the <Shift>+<Ctrl> keys, and click the original Handler. All connections between that Handler and any selected Handlers are deleted.

Moving Handlers

To move a Handler or a selection of Handlers, click and drag it to another location. You cannot move a Handler on top of another Handler, out of the design window, or into the same column as the "Begin" Handler. The "Begin" Handler cannot be moved from its location.

If you move a Handler into a location where any of its connections are not legal, the connection is deleted. If you delete a Handler, any connections to it are automatically deleted.

Caution: It is possible to move an Input Handler into a position where it appears that the Input Handler is connected to the Input Handler in the previous step, but it may not actually be connected.

For example, in Figure 2-9 it looks as if the Input Handler in Step 2 is attached to the Input Handler in Step 1. However, in Figure 2-10, when the Input Handler in Step 2 is moved, you can see that it is actually attached to the “Begin” Handler and not to the Input Handler in Step 1.

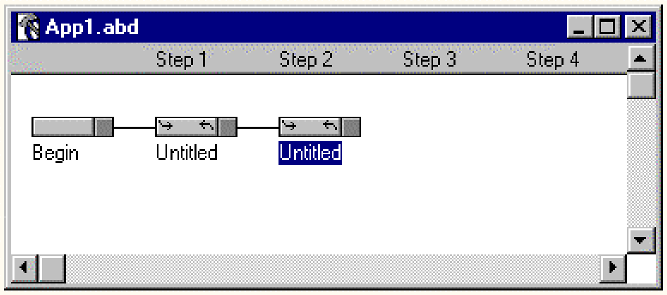


Figure 2-9 Input Handler Connections

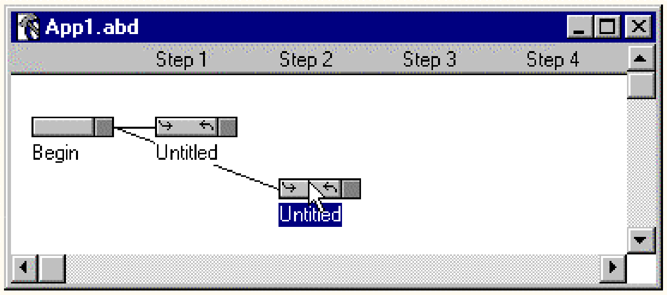


Figure 2-10 Input Handler Connections

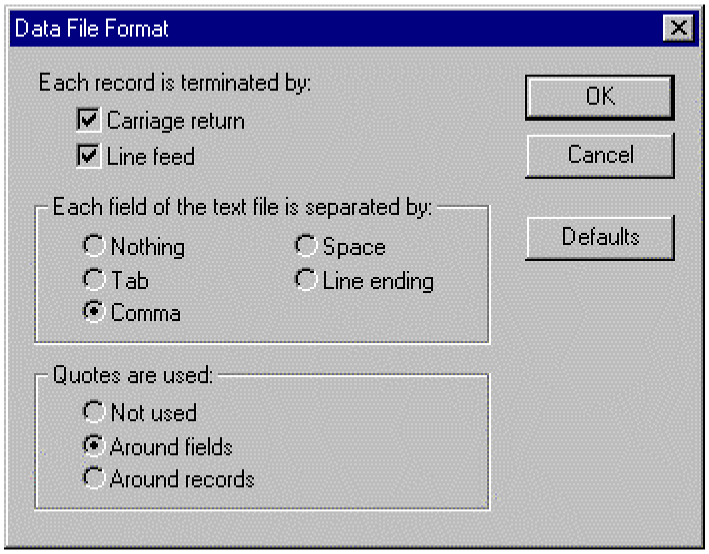
If your application is not stepping to an Input Handler as you expect, click and drag the Input Handler to visually verify the connections.

Data File Format

Displays the **Data File Format** window (Figure 2-11). The **Data File Format** window determines the format of the data file.

Selections are available to end a record in the text file with a carriage return, a line feed, or both; to separate each field in the record with a tab, comma, space, line ending, or nothing; and to decide if quotes are used around fields, records, or not used.

Click **OK** or press <Enter> to save the settings and close the window. Click **Cancel** to close the window without making any changes. Click **Defaults** to reset the settings to the default settings.



Data File Format

Each record is terminated by:
- [✓] Carriage return
- [✓] Line feed

Each field of the text file is separated by:
Nothing, Tab, [✓ Comma], Space, Line ending

Quotes are used:
Not used, [✓ Around fields], Around records

Figure 2-11 Data File Format Window

Other Details

This option assumes that you have a fundamental knowledge of BASIC.

The **Other Details** command opens the **Other Details** dialog box (Figure 2-12). This option, when selected, generates the input <<ESCAPE>> instead of quitting the application when you attempt to transfer the data from a unit. This gives you control over which Input Handler quits the application. (Note: You must include **Software** as an **Input Device** property for the Input Handler that accepts the <<Escape>> input.)

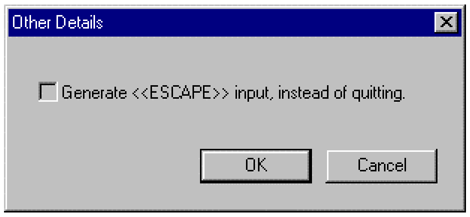


Figure 2-12 Other Details Dialog Box

This option is especially useful if you want an application to perform certain actions before quitting, or if you want to be able to intercept or manage exit events.

When using this command, you must quit the application by using the BASIC code plug-in of **running%=false%** as the last action for the Input Handler quitting the application. If you select the **Other Details** command, without following it up with the BASIC code plug-in **running%=false%**, the data collector will not be able to transfer its data to the computer. (Note: **running%=false%** is the BASIC code for quitting an Application Builder application.) For an example of the use of this feature, refer to the discussion of the Inspect.abd sample application in Appendix A.

Generate Displays

Automatically designs the data collector's display for the selected Handler(s). This command uses the first seven characters of the Input and Detour Handlers' names to design the display. (If you prefer, you can design the displays yourself; see pages 70-83.)

Help Menu

The **Help** menu (Figure 2-13) displays the **About Application Builder** command.

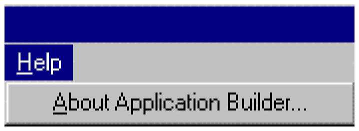
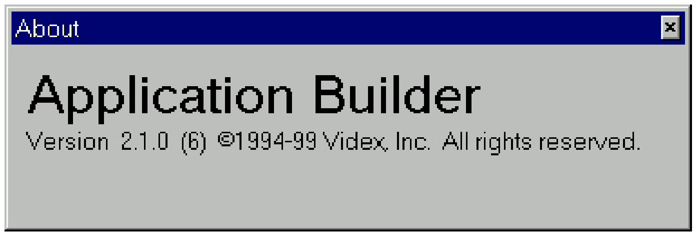


Figure 2-13 Help Menu

About Application Builder

Opens the **About** window (Figure 2-14).

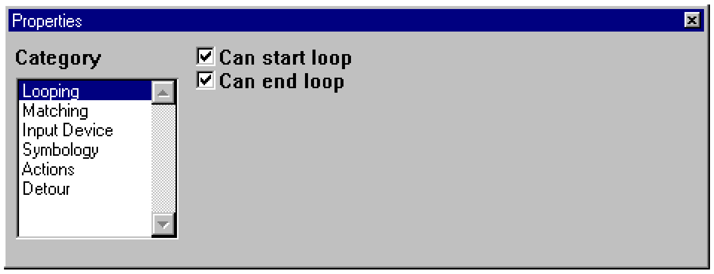

About

Application Builder

Version 2.1.0 (6) ©1994-99 Videx, Inc. All rights reserved.

Figure 2-14 About Application Builder Window

Properties Window



☑ Can start loop
☑ Can end loop

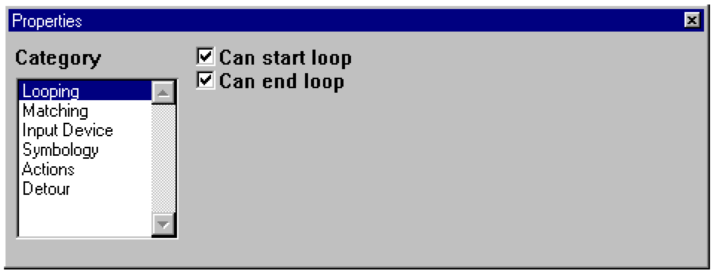
Figure 2-15 Properties Window

The **Properties** window (Figure 2-15) defines the selected Input Handler's properties. To open the window, choose **Properties Window** from the **View** menu or click the **Properties Window** toolbar button. To close the window, click the close window box.

The properties include **Looping**, **Matching**, **Input Device**, **Symbology**, **Actions**, and **Detour**. The following sections describe the properties.

Looping

The **Looping** property (Figure 2-16) determines if the selected Input Handler can start or end a loop. The default for the **Looping** property is the Handler can both start and end a loop. A checkmark in the box indicates that the option is enabled; no checkmark indicates the option is disabled. If an Input Handler can start a loop, it has a curved arrow on its left side; if it can end a loop, it has a curved arrow on its right side.



☑ Can start loop
☑ Can end loop

Figure 2-16 Looping Property

The basic idea behind looping is the application loops back through the Input Handlers that were previously visited (that is, have been "current" Handlers), looking for one that can accept the current user input.

An application's normal tendency is to try to move to the right. If the application cannot move to the right, and if the "current" Handler can end a loop (curved arrow on right side), the application will loop back a step at a time and ask the first Input Handler it finds that starts a loop (curved arrow on left side) if it can accept the input. If it can, the current record is committed to the data file; the input is added to the new current record; the application steps to the accepting Input Handler; and the application again continues to the right. If it cannot, the application asks any other Input Handlers in the same column (that start a loop) before stepping back another column and repeating the procedure.

In order for a loop to occur, the "current" Handler must be allowed to end a loop, and the Input Handler we are looping back to (the one accepting the user input) must be allowed to start a loop.

In the application in Figure 2-17, the "Guard" Handler can start and end a loop (arrows on right and left), the "Location" Handler can start a loop (arrow on left), and the "Status" Handler can end a loop (arrow on right). This means that it is possible to loop back from "Status" to either "Guard" or "Location," and it is also possible to loop back from "Guard" to itself.

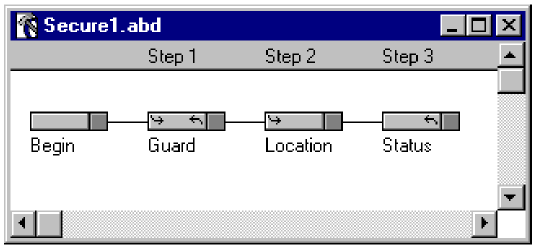


Figure 2-17 Security Application - Looping

When the application loops back to the left, the current record is complete, and is written to the data file. At this point, the application is ready to start constructing a new current record. The new current record may already be partially constructed. If the loop does not go all the way back to step 1, then the application uses the previous data in the new record, up to the start of the loop. (See the example on pages 7–8.)

We will use Figure 2-18 to illustrate the path a loop takes.

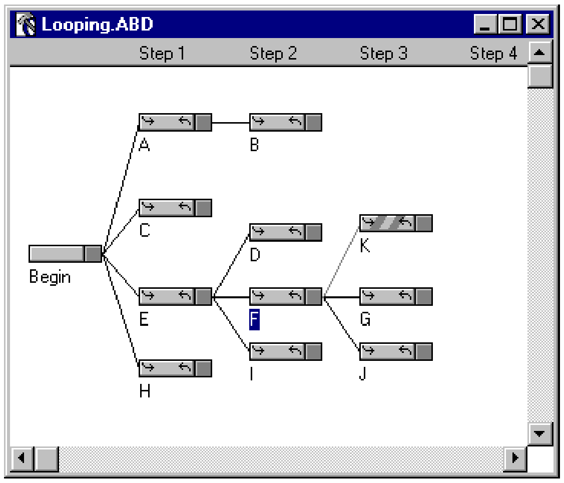


Figure 2-18 Loop Example

Suppose that in this application:

1. All of the Input Handlers (except **Begin**) can both start and end a loop and Input Handler (**K**) is a Detour Handler.
2. The Input Handlers **Begin**, **E**, and **F** have been visited.
3. The **F** Handler is the “current” Handler.

If the **F** Handler is the “current” Handler and a user entry is made, the application would search for an accepting Handler in the following order:

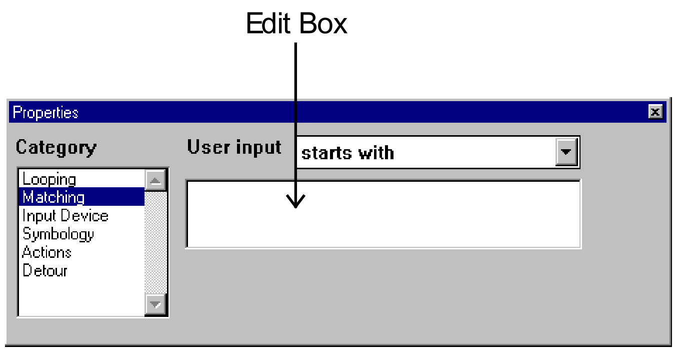
G, **J**, **D**, **F**, **I**, **A**, **C**, **E**, **H**, and if none of the normal Input Handlers can accept the input, the application then checks Detour Handler, **K**.

Note 1: The **B** Handler is not searched, because it cannot be directly reached from a previously visited Input Handler.

Note 2: All normal Input Handlers are searched before a Detour Handler is searched.

Note 3: An application can never loop back all the way to the Begin Handler. The only time the application is in the Begin Handler is when it starts running, before any data has been entered.

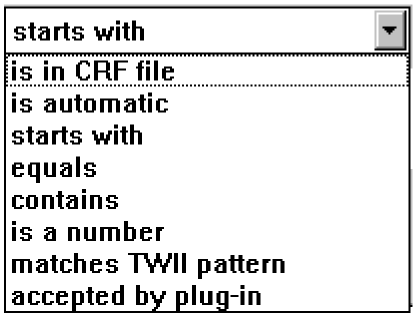
Matching



User input: starts with

Figure 2-19 Matching Property

The **Matching** property (Figure 2-19) determines whether an Input Handler will accept an entry. An Input Handler will only accept an entry if it matches the **Matching** property. The **Matching** property is the only property that does not have a default setting; you must identify the **Matching** property for each Input Handler in order for the software to build the application. There are several **Matching** property choices (Figure 2-20); they are described in the following sections.

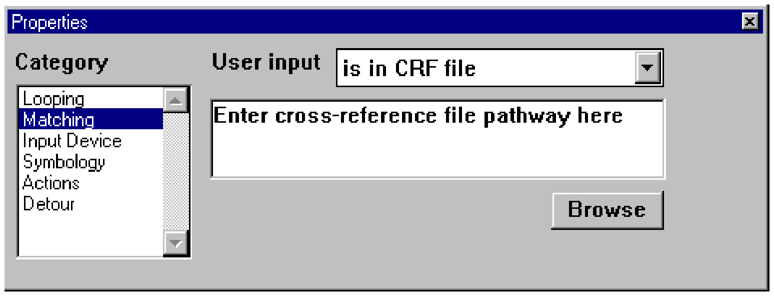


starts with

is in CRF file
is automatic
starts with
equals
contains
is a number
matches TWII pattern
accepted by plug-in

Figure 2-20 Matching Property Choices

is in CRF file
The entry must match one of the selected cross-reference (CRF) file's keys. *(Keys are the entries in the CRF file's Input column.)* The selected CRF file will automatically transfer to the data collector along with the application when **Transfer Application** is chosen from the **File** menu.

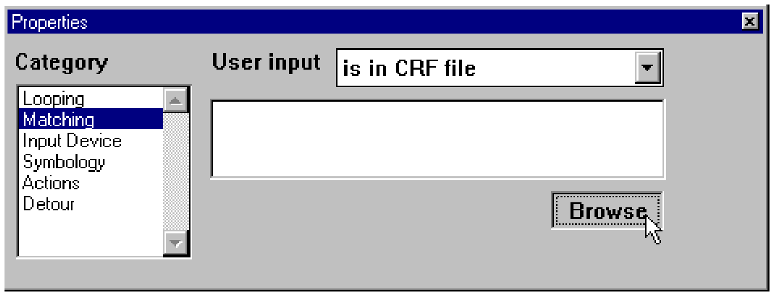


User input: is in CRF file

Enter cross-reference file pathway here

Figure 2-21 Matching Property - Is In CRF File

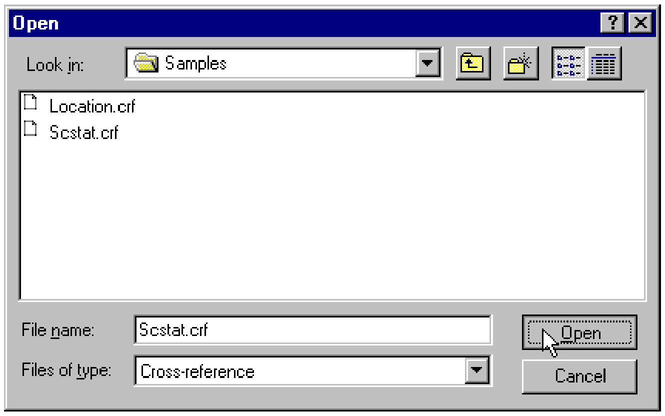
To select a CRF file, type the file's name and pathway in the **Properties** window edit box or click **Browse** to navigate to the file (Figure 2-22).



User input: is in CRF file

Figure 2-22 Is In CRF File - Click Browse

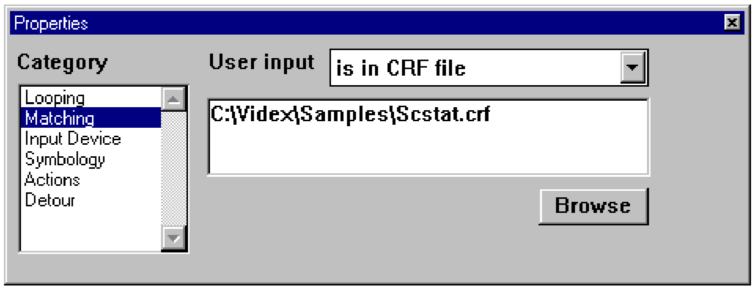
If you click **Browse** to navigate to the file, the **Open** window appears. To view the CRF cross-reference files, choose **Cross-reference** or **Any file** from the **File of type** menu at the bottom of the window. Select the CRF file and click **Open** (Figure 2-23) or double-click the file's icon; the name of the CRF file and its pathway are then displayed in the **Properties** window's edit box (Figure 2-24).



File name: Scstat.crf

Files of type: Cross-reference

Figure 2-23 Open CRF File Window

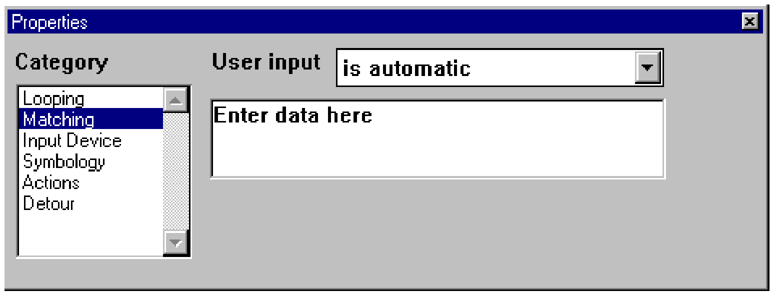


User input: is in CRF file

C:\Videx\Samples\Scstat.crf

Figure 2-24 Matching Property - Is In Scstat.crf File

is automatic
The entry is automatically entered by the application without any activity by the user at all. In other words, the application *pretends* that the user made an entry, and the Input Handler accepted it, even though the user didn't do anything. Type the data you want automatically entered in the edit box (Figure 2-25).

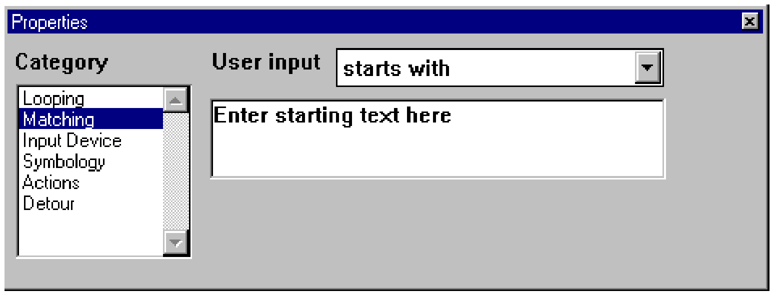


User input: is automatic

Enter data here

Figure 2-25 Matching Property - Is Automatic

starts with
The entry must start with the entered text. Type the text that the user input must start with in the edit box (Figure 2-26).

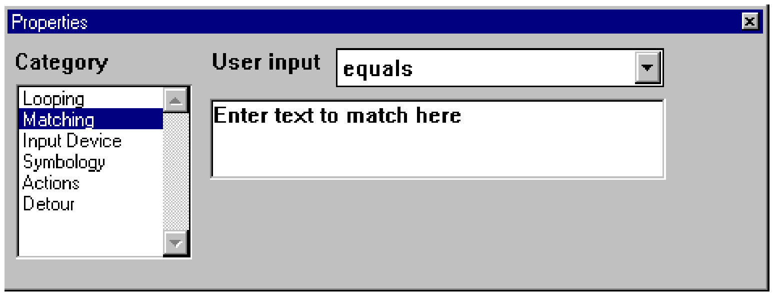


User input: starts with

Enter starting text here

Figure 2-26 Matching Property - Starts With

equals
The entry must exactly match the text typed in the edit box (Figure 2-27).

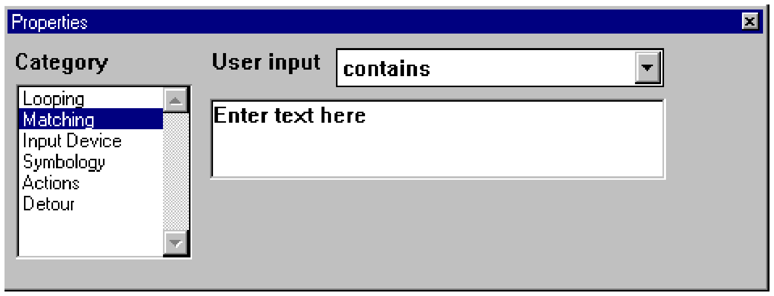


User input: equals

Enter text to match here

Figure 2-27 Matching Property - Equals

contains
The entry must contain the text typed in the edit box somewhere within the user input (Figure 2-28).

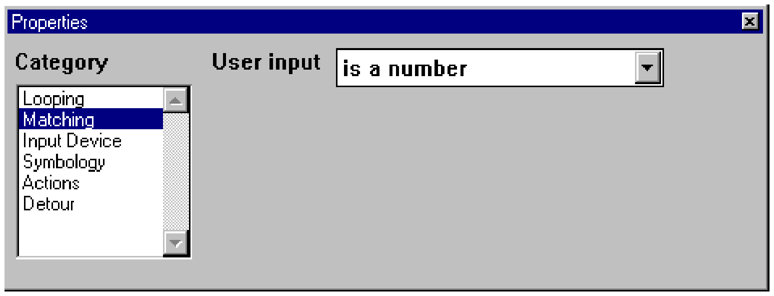


User input: contains

Enter text here

Figure 2-28 Matching Property - Contains

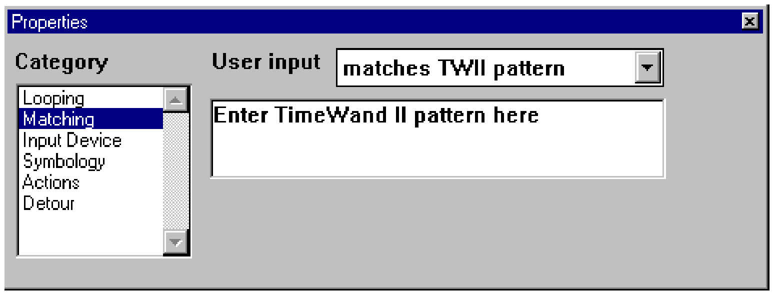
is a number
The entry must be a positive number (Figure 2-29). The number must be composed of the digits 0–9. The number may not contain any symbols such as commas, decimals, or percentage marks.



User input: is a number

Figure 2-29 Matching Property - Is a Number

matches TWII pattern
The entry must satisfy the TimeWand II (TWII) match pattern typed in the edit box (Figure 2-30). If you are familiar with the TimeWand II bar code reader and its match patterns, this command will be familiar to you.



User input: matches TWII pattern

Enter TimeWand II pattern here

Figure 2-30 Matching Property - Matches TWII Pattern

A TWII match pattern is a pattern you create by combining letters, numbers, symbols, and/or wild card characters to describe the input acceptable to the Input Handler.

When specifying a TWII match pattern, the following five characters are used as wild cards to create the pattern:

#	a number [0-9] must be in this position
@	a letter [a..z, A..Z] must be in this position
&	any character can be in this position
=	the rest of the string can be any characters or no character
>	the rest of the string must be numbers [0..9] or no character

These are the actual letters, numbers, and symbols the match pattern can contain:

A..Z 0..9 space + - / % . $

A TWII match pattern can be used in a variety of situations to provide strict control *(for example, the match pattern* ***VIDEX-7##*** *would require an entry that begins with VIDEX-7 followed by two numbers*) or to allow flexibility *(for example, the match pattern* ***&=*** *would accept any entry*).

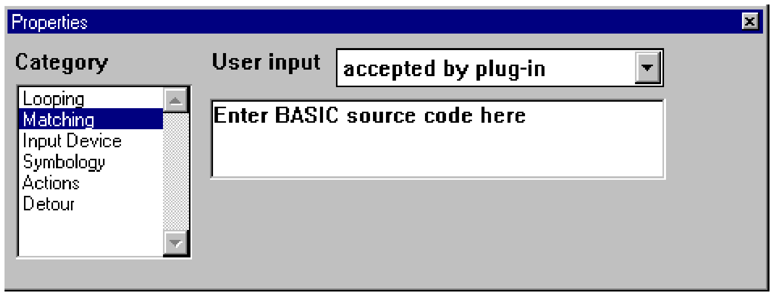
Following are examples of TWII match patterns:

&=	any entry is acceptable
#>	any entry consisting of numbers
E=	any entry that starts with the letter E
E>	any entry that starts with the letter E and is followed only by numeric digits
E0003	entry must match E0003 exactly
AAA=	any entry that begins with AAA
AA#A	a four-digit long entry with A in the first, second, and fourth positions and a number in the third position
AA@A	same as previous entry, but with a letter in the third position
AA&A	same as previous entry, but with any character in the third position

Application Builder version 2.1.0 and higher implements a feature which allows “Does **not** match a TimeWand II pattern” matching criterion. To do this, choose the matching criterion “matches TWII pattern” but precede the pattern with a “ < ”. For example, if barcodes which begin with “A” are to be rejected by an input handler, the TimeWand II pattern would be “<A&=”.

accepted by plug-in
This feature provides for more complex pattern matching by using BASIC source code to define the entry. This feature assumes the user has a fundamental knowledge of BASIC.

BASIC source is inserted in the application exactly as it is entered in the edit box (Figure 2-31).



Enter BASIC source code here

Figure 2-31 Matching Property - Accepted By Plug-in

For example, the following line of BASIC code:

```
If right$(evt_data$,1) = "T" then accepts% = true%
```

uses the Videx BASIC **RIGHT$** function to define a match pattern that accepts any entry that ends with a **T**. EVT_DATA$ is the string that contains the current entry and ACCEPTS% must be set equal to true to accept the entry.

What this line of code does is look at the EVT_DATA$ string, take the rightmost character, and see if the character is equal to **T**; if the character is a **T,** then ACCEPTS% is set equal to true and the current input is accepted. If the character is not a **T**, then the current input is not accepted.

In this example, the matching criterion is simple enough that the BASIC code fits into a single plug-in. Often, the desired matching criterion is too complex to fit into a single line of BASIC code. In that situation, the text of the plug-in should be:

gosub myroutine

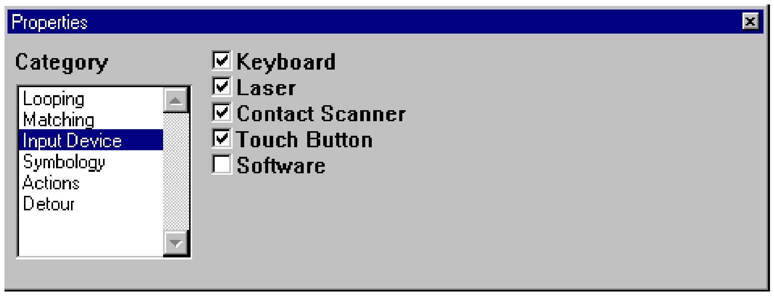
where **myroutine** is a subroutine containing the BASIC code which implements the matching criterion:

```
myroutine:
    •
    •
    •
    if (condition) then accepts% = true%
return
```

The subroutine myroutine can be run as a series of plug-ins executed by the “Begin” handler. Alternatively, myroutine can be incorporated into the application using the “Include” command, which is discussed on page 190.

See Chapter 4 for more information on plug-ins. See the *Videx BASIC Manual* for more information on the BASIC commands.

Input Device


Properties

Category

- Looping
- Matching
- Input Device
- Symbology
- Actions
- Detour

[✓] Keyboard
[✓] Laser
[✓] Contact Scanner
[✓] Touch Button
[ ] Software

Figure 2-32 Input Device Property

The **Input Device** property determines what methods of data entry are acceptable to the Input Handler. An Input Handler can only accept user input from devices with a checkmark in the adjacent box. The **Input Device** property has a default of **Keyboard**, **Laser**, **Contact Scanner**, and **Touch Button** (Figure 2-32). The **Input Device** property choices are described in the following sections.

Keyboard

The user input must be entered by the LaserLite Pro or LaserLite Mx keypad or by a bar code scanpad.

A bar code scanpad (Figure 2-33) is a card that contains printed Code 3 of 9 bar codes assigned to numbers, letters, and symbols. A scanpad is included with your data collector. You can scan the individual scanpad characters to enter any bar code. This allows manual entry of data and is especially useful if you encounter a bar code that is torn, damaged, or otherwise unreadable.

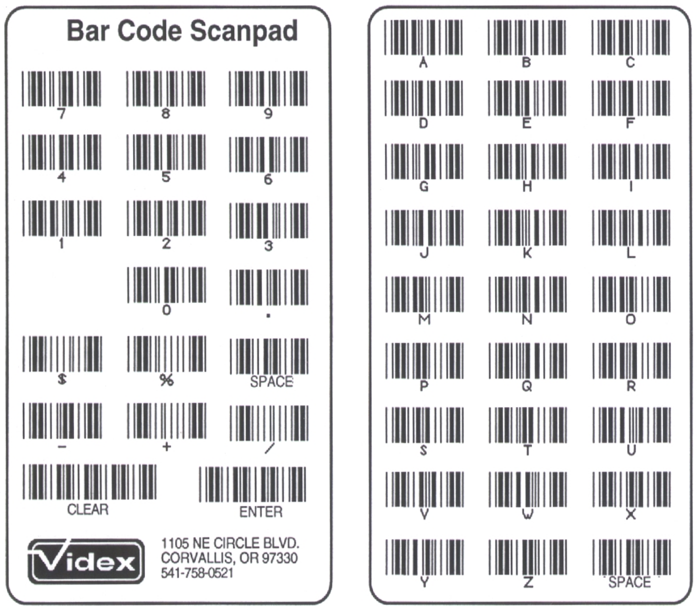


Figure 2-33 Bar Code Scanpad

Laser
The user input must be entered by the LaserLite, LaserLite Pro, or LaserLite Mx laser.

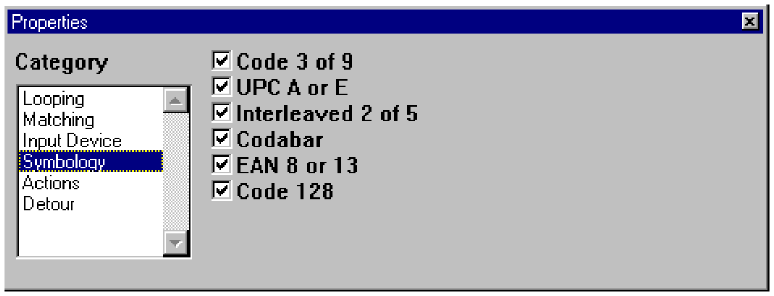
Contact Scanner
The user input must be entered by the DuraTrax contact scanner.

Touch Button
The user input must be entered by the DuraTrax, LaserLite Pro, or LaserLite Mx Touch Memory button reader.

Software
This feature assumes the user has a fundamental knowledge of BASIC. In some cases the software simulates data entry, and when it does **Software** must be used as the input device. For example, the **Audit** application in the **Samples** folder uses **Software** as the Input Device when the “Loc & Part#” Handler steps through the cross-reference file of locations and part numbers.

This feature must also be used when you use the **Other Details** command. See the **Other Details** command section on page 49 for more information.

Symbology



☑ Code 3 of 9
☑ UPC A or E
☑ Interleaved 2 of 5
☑ Codabar
☑ EAN 8 or 13
☑ Code 128

Figure 2-34 Symbology Property

If the user input is a bar code scan, the **Symbology** property (Figure 2-34) determines if the bar code symbology is acceptable to the Input Handler. A checkmark in the box indicates the Input Handler accepts the bar code symbology; no checkmark indicates the symbology is not acceptable. The default setting for the **Symbology** property is to accept all of the listed symbologies.

DuraTrax, LaserLite, LaserLite Pro, and LaserLite Mx can read Code 3 of 9, UPC, Interleaved 2 of 5, Codabar, EAN, and Code 128 bar code symbologies. Figure 2-35 describes the different symbologies.

Bar Code	**Uses**	**Description**
Code 3 of 9	Auto parts, labels for sorting, Dept. of Defense, purchasing, and supply control	The most popular industrial and government bar code symbology. Reliable. Variable length bar code with both numeric and uppercase letters.
Interleaved 2 of 5	Industry, cartons of food, labels for sorting, and supply control	A popular industrial bar code symbology. This is a numbers only code and requires an even number of digits to make a legal code.
Codabar	Medical sample control, photo processing, and libraries	Used for Red Cross blood banking. This is a variable length code.
EAN	European product code	European retail code symbology.
UPC	Universal Product Code; retail products, grocery, drug, and medical goods	A food and drug retail market symbology. This is a variable length code.
Code 128	Shipping/receiving, inventory	A popular high-density bar code symbology. Variable length with upper and lowercase letters, all ASCII control characters, and all numeric entries.

Figure 2-35 Bar Code Symbologies

Actions

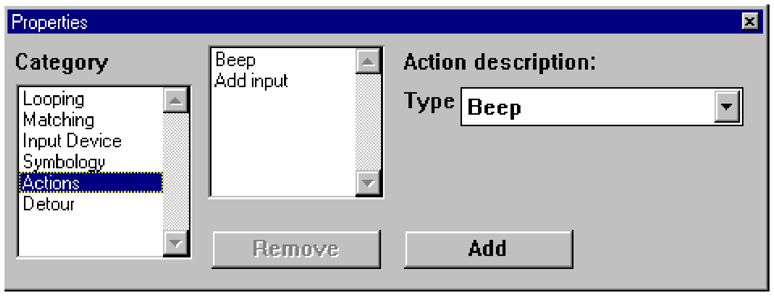


Figure 2-36 Actions Property

The **Actions** property (Figure 2-36) provides a list of actions that can be performed by the Input Handler when it accepts the user input and becomes the “current” Handler. The center section of the **Properties** window is the action list window. The action list window lists the actions to be performed by the Input Handler when it becomes the “current” Handler. The **Actions** property default is to **Beep** and **Add input to record**.

At the right side of the window is the action description pop-up menu. Click the pop-up menu indicator to display the menu (Figure 2-37).

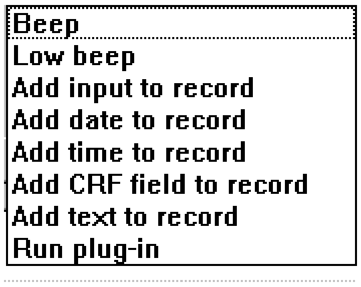


Figure 2-37 Action Description Pop-up Menu

To add an action to the action list, select the action in the pop-up menu and click **Add**. To reorder the actions in the action list window, click and drag them around. To delete an action from the action list, select the action in the action list window and click **Remove**.

The actions are described in the following sections.

Beep
Beeps with a high pitch. This should normally be used as the first action an Input Handler performs to alert the user that the input was accepted.

Low beep
Beeps with a low pitch. Same beep used to signify "bad" input. (Bad input is input which is not accepted by any Input Handler.) This action can be used in a Detour Handler to signal an error, or you can alternate the **Beep** and **Low Beep** actions to make a siren-type sound.

Add input to record
Adds user input to the current record.

Add date to record
Adds current date to the current record.

Add time to record
Adds current time to the current record.

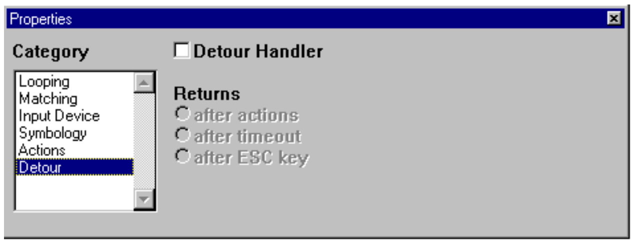
Add CRF field to record
Adds cross-reference (CRF) file description to the current record. If the user input matches a key in the CRF file, then any of the description fields associated with the key may be added to the current record.

Add text to record
Adds arbitrary text to the current record.

Run plug-in
Inserts BASIC source code in the application. This feature assumes the user has a fundamental knowledge of BASIC. See Chapter 4 for information on using plug-ins.

Detour

The **Detour** property determines if the selected Input Handler is a standard Input Handler or a Detour Handler. A Detour Handler lets you pause an application, perform another task with the data collector, and then resume the application. The default for the **Detour** property is no detour, as shown in Figure 2-38. A Detour Handler is typically used to display a message (such as instruction, error, and so forth) to the user.


Properties

Category

Looping
Matching
Input Device
Symbology
Actions
Detour

☐ Detour Handler

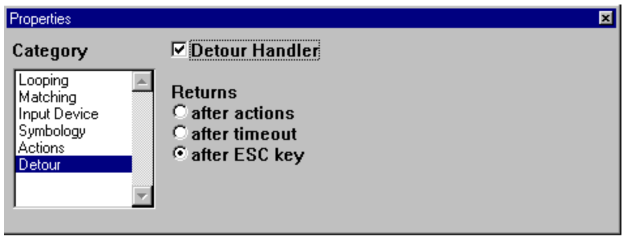
Returns
○ after actions
○ after timeout
○ after ESC key

Figure 2-38 Detour Property - No Detour (Default)

To make the selected Input Handler a Detour Handler, click the **Detour Handler** check box. A checkmark in the box indicates the Input Handler is a Detour Handler (Figure 2-39); no checkmark in the box indicates the Input Handler is not a Detour Handler.

When an application returns from a Detour Handler, it always returns to the previous Input Handler, and resumes where it left off. The application may return from a Detour Handler in one of three ways:

- After the Detour Handler’s actions are performed.
- After three seconds or a button press (whichever is first).
- After the ESC key is pressed on the LaserLite Pro or LaserLite Mx.


Properties

Category

Looping
Matching
Input Device
Symbology
Actions
Detour

☑ Detour Handler

Returns
○ after actions
○ after timeout
● after ESC key

Figure 2-39 Detour Property

Display Window

The **Display** window (Figure 2-40) is where the data collector's display is designed for each Input Handler. To open the **Display** window, choose **Display Window** from the **View** menu or click the **Display Window** toolbar button. To close the **Display** window, click the close window box.

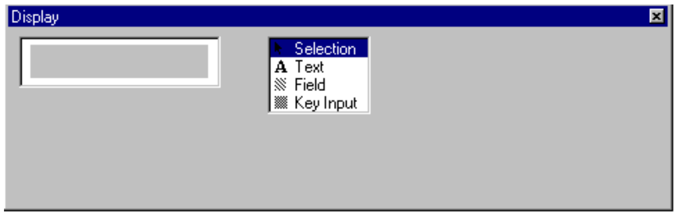


Figure 2-40 Display Window

The "current" Input Handler is responsible for the display. You can specify what the display should look like when an Input Handler is the "current" Handler by selecting the Input Handler and designing the display with the **Display** window. The **Display** window contains the display area and the tool palette. The display area is a graphic representation of the data collector's display (2 x 16 characters) and the tool palette contains tools for designing the display. The tool palette contains the following tools:

Selection Tool	Selects an object.
Text Tool	Allows text entry.
Field Tool	Inserts a field object. A field object marks the area where data from the current or a previously visited Input Handler can be displayed; the displayed data may be either the user input, the date, the time, or a CRF file field description. (The **Field** tool can also be used to display the value of any variable in the application; this may be useful when using plug-ins (see pages 189–193).

Key Input Tool	Inserts a key input area; the key input area marks the area where keypad and scanpad entries are displayed.

The **Display** window is a small draw program. Like most draw programs:

- You can create objects by selecting the appropriate tool, and placing the object in the display area by dragging a rectangle with the mouse.
- You can move objects by selecting the **Selection** tool, and clicking and dragging the object to the new location.
- You can resize objects by selecting the **Selection** tool, clicking the object, and then clicking and dragging one of the object's size handles.
- You can bring an object forward by selecting the **Selection** tool, holding the <Ctrl> key down and clicking the object.
- You can send an object back by selecting the **Selection** tool, holding the <Shift>+<Ctrl> keys down and clicking the object.
- You can delete an object by selecting the **Selection** tool, clicking the object and choosing **Delete Objects** from the **Edit** menu.

The following sections describe the tools contained in the **Display** window's tool palette.

Selection Tool

The **Selection** tool selects objects in the data collector's display area. To select an object, click it or drag a rectangle around the object. Shift-click to select multiple objects or drag a rectangle around the objects.

To delete an object in the display area, click the object with the Selection tool, then choose **Delete Objects** from the **Edit** menu or press **Ctrl+B**.

To cut an object from the display area and place it in the clipboard, click the object with the **Selection** tool, then choose **Cut Objects** from the **Edit** menu or press **Ctrl+X**.

To copy an object from the display area to the clipboard, click the object with the **Selection** tool, then choose **Copy Objects** from the **Edit** menu or press **Ctrl+C**.

To paste an object from the clipboard to the display area, choose **Paste Objects** from the **Edit** menu or press **Ctrl+V**.

Text Tool

The **Text** tool allows you to enter text when designing the data collector's display. The text to be displayed is typed into the edit box and then the <Enter> key is pressed to accept the entry. The text is then displayed in the display area.

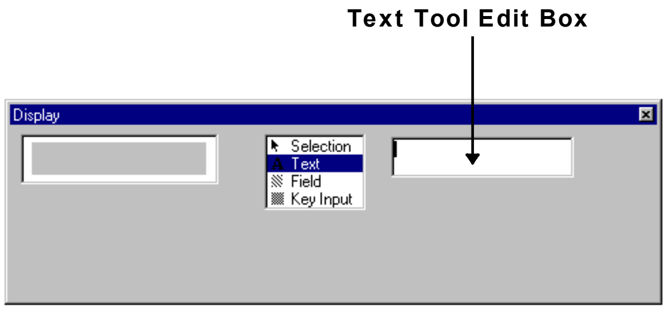


Figure 2-41 Display Window - Text Tool

To use the Text tool

1. Click the **Text** tool to select it (Figure 2-42).

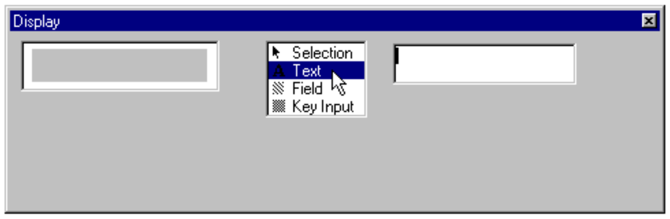


Figure 2-42 Text Tool Selected

2. Type **Security**; the typed text is displayed in the edit box (Figure 2-43).

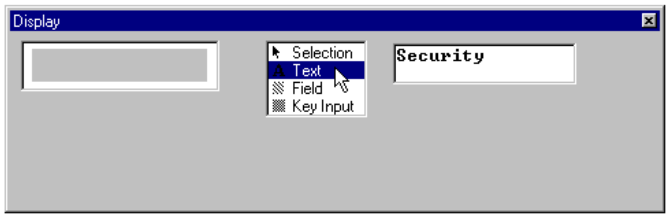


Figure 2-43 Display - Type Security

3. Press <Enter>; **Security** appears in the display area (Figure 2-44).

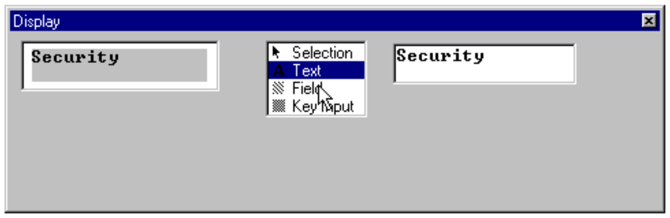


Figure 2-44 Display - Security Displayed

4. Click at the beginning of the second line in the edit box (Figure 2-45).

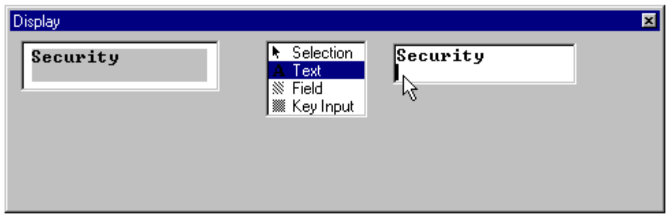


Figure 2-45 Display - Second Line

5. Type **Guard ID:** (Figure 2-46).

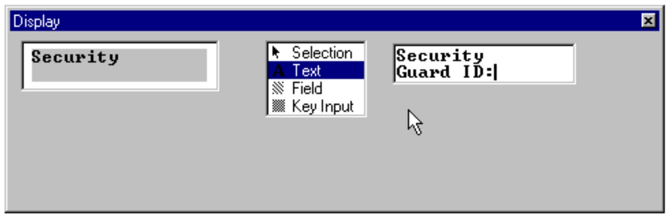


Figure 2-46 Display - Type Guard ID:

6. Press <Enter>; **Guard ID:** is shown on the second line in the display area (Figure 2-47).

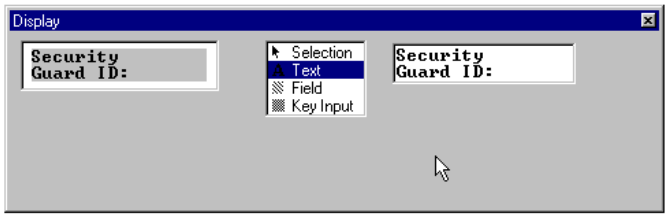


Figure 2-47 Display - Guard ID: Displayed

Field Tool

The **Field** tool places a field object in the display area. A field object displays information from the current or a previously visited Input Handler. The types of information that can be displayed as field objects are: **User Input**, **Date**, **Time**, **CRF Description**, **Text**, and **Variable**.

User Input
Displays the user input from the current or a previously visited Input Handler. This choice is indicated by a lowercase **i** in the field object.

Date
Displays the date from the current or a previously visited Input Handler. This choice is indicated by a lowercase **d** in the field object.

Time
Displays the time from the current or a previously visited Input Handler. This choice is indicated by a lowercase **t** in the field object.

CRF Description
Displays a cross-reference (CRF) file description (Fields 1, 2, or 3 only) from the current or a previously visited Input Handler. This choice is indicated by a lowercase **c** in the field object.

Text
Displays arbitrary text.

Variable
Displays the value of a named variable. This feature assumes that the user has a fundamental knowledge of BASIC. The variable must be a BASIC string or a string expression.

To use the Field tool

1. Click the **Field** tool (Figure 2-48).

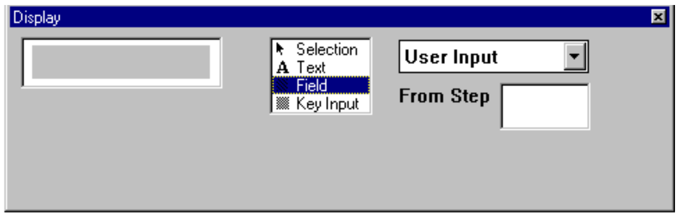


Figure 2-48 Field Tool Selected

2. There are two ways to use the **Field** tool: 1) drag a rectangle in the display area, then select the type of information and Input Handler column; or, 2) select the type of information and Input Handler column, and drag a rectangle in the display area.

3. Click and drag a field rectangle in the display area where you want the data to appear (Figure 2-49).

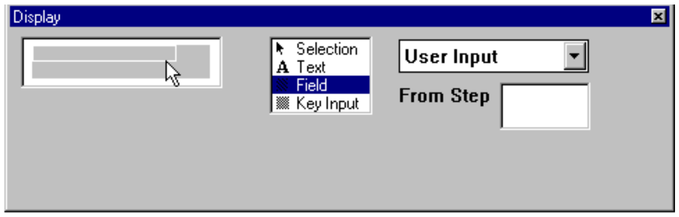


Figure 2-49 Field Marker Area Defined

4. Release the mouse button; a field object appears (Figure 2-50).

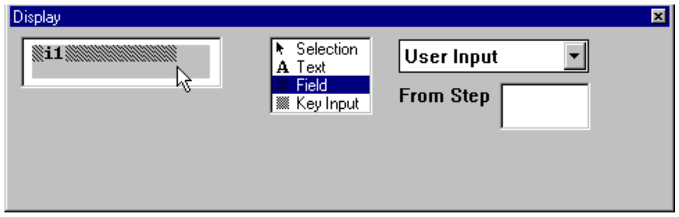


Figure 2-50 Display - Field Object

5. The field object defaults to **User Input** from **Step 1**; this is indicated by the **i1** in the field object. When the field object is selected, its information is displayed to the right of the tool palette. (Figure 2-51 shows **User Input** from **Step 1** as the information. This means that the user input from the Input Handler in Step 1 will be displayed in this area when this Input Handler is the “current” Handler.)

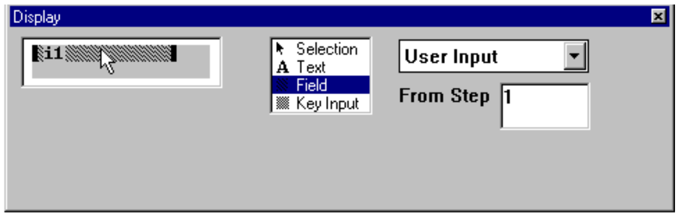


Figure 2-51 Display - Default Field Object

6. To change the type of data the field object displays, click the pop-up menu indicator next to **User Input**, to display the **Field** pop-up menu (Figure 2-52).

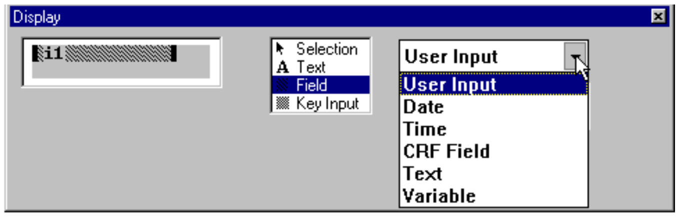


Figure 2-52 Display - Field Pop-up Menu

7. Select another option from the **Field** pop-up menu. We will select **Time** in this example (Figure 2-53).

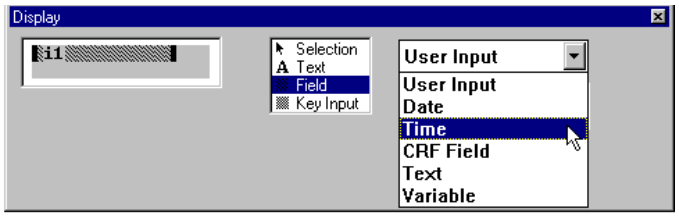


Figure 2-53 Field Pop-up Menu

8. The field object indicator changes from **i1** to **t1** (Figure 2-54). A **t1** indicates that the time from the Step 1 Input Handler will be displayed here.

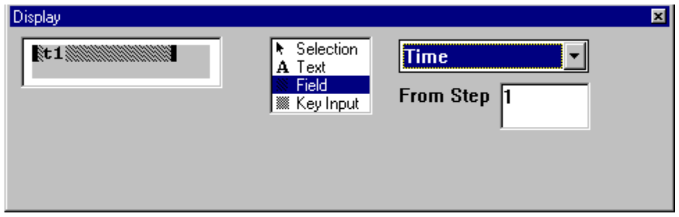
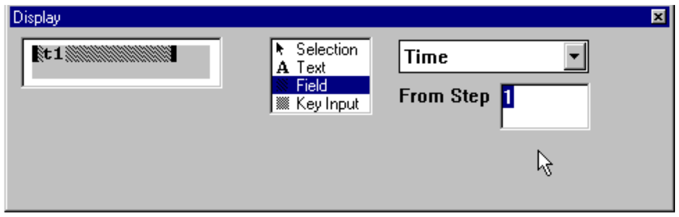


Figure 2-54 Display - Time Field Object

9. To change which Input Handler the information is taken from, enter the Input Handler's step number (located above the Input Handler in the design window) in the **From Step** section. (Note: If there is more than one Input Handler under the step number, the software displays the information from the last visited Input Handler.)



Time

From Step 1

Figure 2-55 Display - Field Object Input Handler Selected

For example, if you are designing a display for the **Secure1** application shown in Figure 2-56:

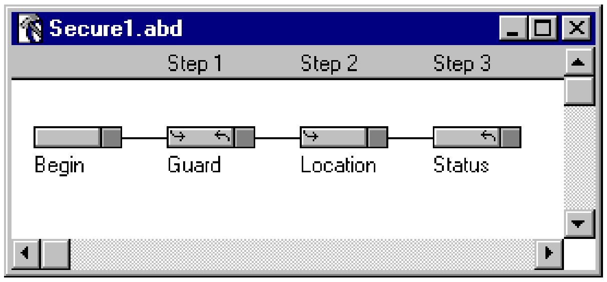
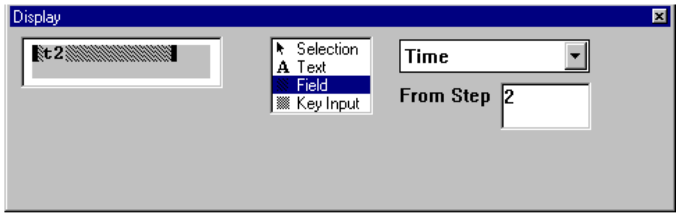


Figure 2-56 Secure1 Application

- If you want the time from the "Location" Handler, choose **Time** from the pop-up menu and enter a **2** in the **From Step** section (Figure 2-57).

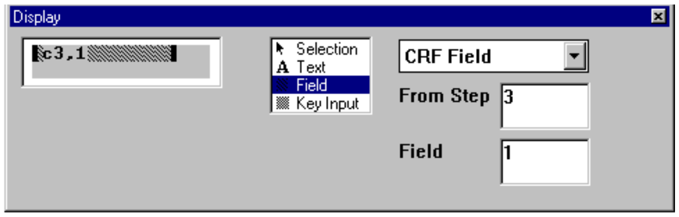


t2

From Step 2

Figure 2-57 Display - Field Object for "Location" Handler's Time

- If you want the CRF file description from the "Status" Handler, choose **CRF Field** from the pop-up menu and enter a **3** in the **From Step** section and a **1** in the **Field** section.



c3,1

From Step 3

Field 1

Figure 2-58 Display - Field Object for "Status" Handler's CRF Description

Key Input Tool

The **Key Input** tool displays a gray rectangle with a blinking cursor for scanpad entry of more than one character. You must specify where the key input area should appear in the display. If a display does not have a key input area, then each scanpad entry is considered a complete bar code entry.

To use the Key Input tool

1. Click the **Key Input** tool (Figure 2-59).

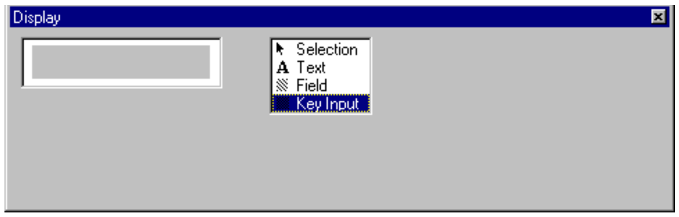


Figure 2-59 Key Input Tool Selected

2. Click in the display area where you want the key input area to begin (Figure 2-60).

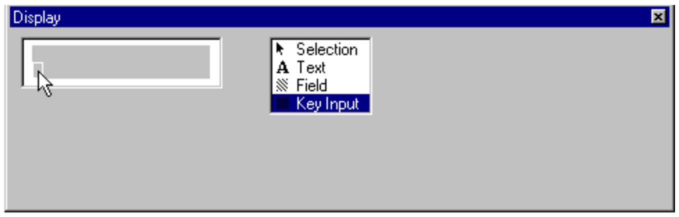


Figure 2-60 Select Where Key Input Area Begins

3. Drag the cursor to where you want the key input area to end (Figure 2-61).

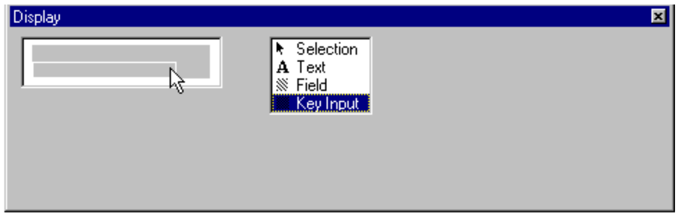


Figure 2-61 Select Where Key Input Area Ends

4. Release the mouse button; the key input area appears in the display area (Figure 2-62).

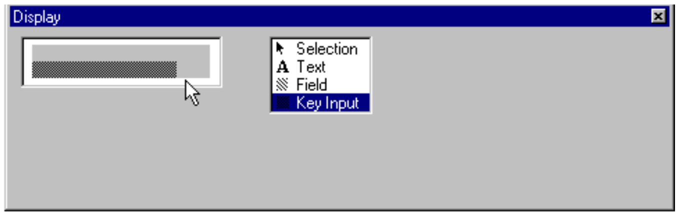


Figure 2-62 Key Input Area Displayed

Summary Window

The **Summary** window summarizes the selected Input Handler's properties. To open the **Summary** window, choose **Summary Window** from the **View** menu or click the **Summary Window** toolbar button. To close the **Summary** window, click the close window box. If an Input Handler is not selected, the **Summary** window is empty as in Figure 2-63.

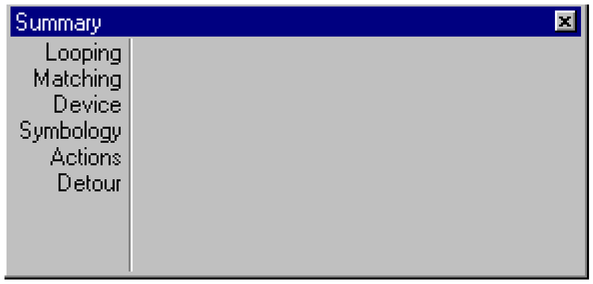
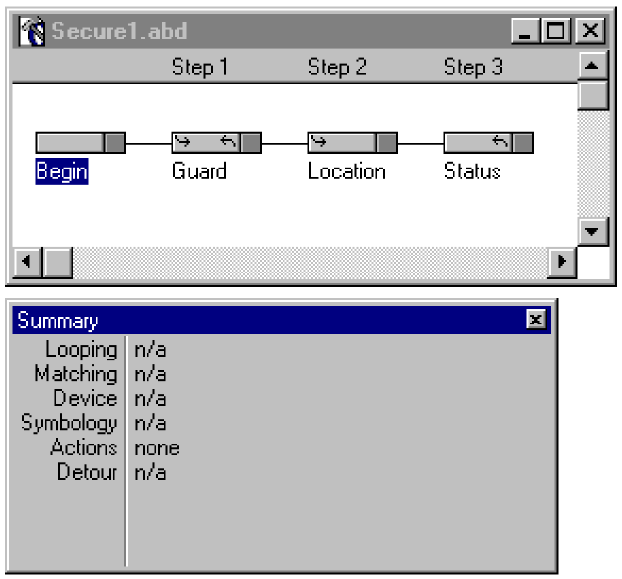


Figure 2-63 Summary Window - No Input Handler Selected

The **Summary** window reports:

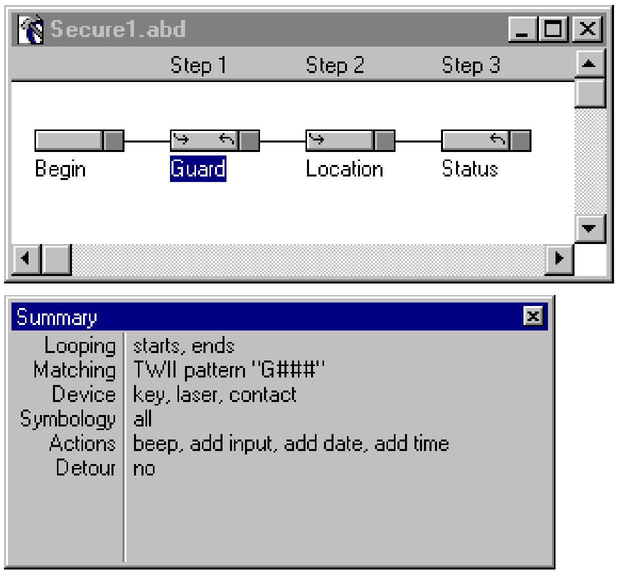
- If a Handler begins and/or ends a loop
- What the matching criterion for the Handler is
- Which devices the Handler accepts input from
- What bar code symbologies the Handler accepts
- What actions the Handler will perform if the input is accepted
- Whether or not this Handler is a Detour Handler

Figure 2-64 shows the **Summary** window of the **Secure1** application's "Begin" Handler. The **Summary** window lists **n/a** (not applicable) for the properties not used by the "Begin" Handler. Figure 2-65 shows the **Summary** window for the **Secure1** application's "Guard" Handler. Figure 2-66 shows the **Summary** window for the **Secure1** application's "Location" Handler. Figure 2-67 shows the **Summary** window for the **Secure1** application's "Status" Handler.



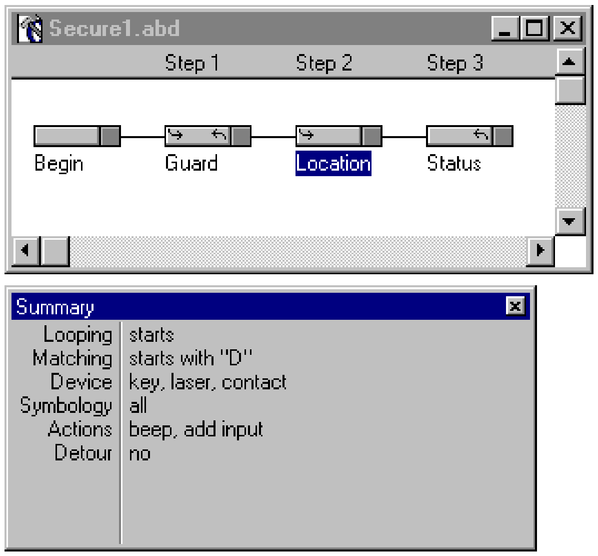
Looping	n/a
Matching	n/a
Device	n/a
Symbology	n/a
Actions	none
Detour	n/a

Figure 2-64 Secure1 Application - "Begin" Handler and Summary Window



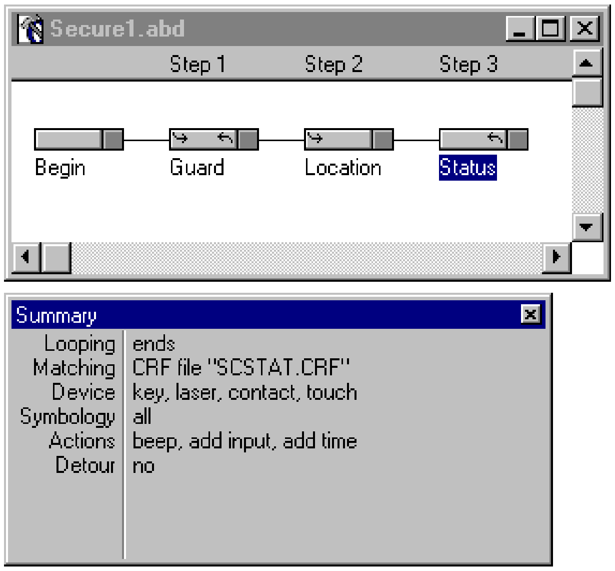
Looping	starts, ends
Matching	TWII pattern "G###"
Device	key, laser, contact
Symbology	all
Actions	beep, add input, add date, add time
Detour	no

Figure 2-65 Secure1 Application - “Guard” Handler and Summary Window



Looping	starts
Matching	starts with "D"
Device	key, laser, contact
Symbology	all
Actions	beep, add input
Detour	no

Figure 2-66 Secure1 Application - “Location” Handler and Summary Window

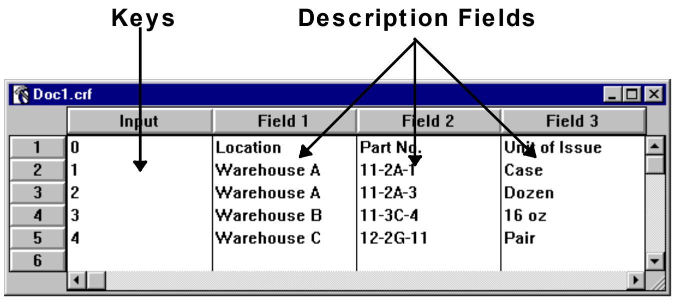


Looping	ends
Matching	CRF file "SCSTAT.CRF"
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add input, add time
Detour	no

Figure 2-67 Secure1 Application - “Status” Handler and Summary Window

Cross-Reference Files

A cross-reference (CRF) file can be used to verify input and make entries to the data file. A CRF file is a list of keys and corresponding description fields. A key represents the user input and the description fields describe the key. Each key can have one or more description fields. Figure 2-68 shows a sample CRF file.



	Input	Field 1	Field 2	Field 3
1	0	Location	Part No.	Unit of Issue
2	1	Warehouse A	11-2A-1	Case
3	2	Warehouse A	11-2A-3	Dozen
4	3	Warehouse B	11-3C-4	16 oz
5	4	Warehouse C	12-2G-11	Pair
6				

Figure 2-68 Sample CRF File

If an Input Handler's **Matching** property specifies that it accepts input from a CRF file, the user input is compared to the keys (Input column entries) in the CRF file. If the user input matches one of the keys, the input is accepted, and either the user input or one of the matching key's fields is added to the data file.

To add the user input to the data file, the **Add input to record** action must be in the Input Handler's action list; to add one of the fields to the data file, the **Add CRF field to record** action must be in the Input Handler's action list and the desired field number must be entered in the Field edit box. You can only enter one field number per **Add CRF field to record** action, but you can have multiple **Add CRF field to record** actions in an Input Handler's action list.

With Application Builder, you can create a CRF file with up to three description fields by choosing the **New CRF** command from the **File** menu (Figure 2-69).

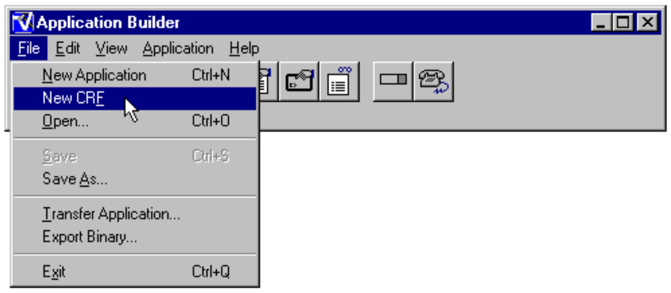


Figure 2-69 File Menu - New CRF Command

When the **New CRF** command is chosen, an empty CRF window opens (Figure 2-70). Type the keys in the **Input** column and the descriptions in the **Field** columns and save the file. See page 27 for a quick-start section on creating a new CRF file.

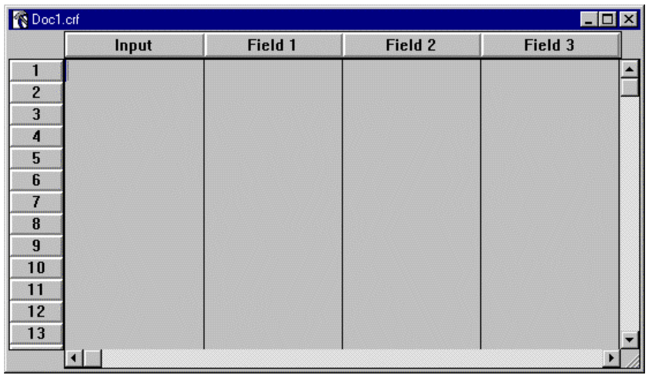


Figure 2-70 CRF Window

You can also use an existing database file, but the database file must be saved as a text file and converted into a CRF file before the data collector can use it.

If the existing database file has 0 to 3 description fields besides the key field, it can be converted into a CRF file by opening and saving the file in the CRF window or by converting the file with the **Vxcrfw.exe** or the

Vxcrf.exe convert programs. If the existing database file has 4 to 16 description fields (16 is maximum) besides the key field, it must be converted with the DOS program **Vxcrf.exe**. **Vxcrfw.exe** is the Windows program for converting a text file into a CRF file. **Vxcrf.exe** is the DOS program for converting a text file into a CRF file. See the *Developer's Reference Manual* for information on converting the files with **Vxcrfw.exe** or **Vxcrf.exe**.

To convert the text file by opening it in the CRF window, save the file as an ASCII text file *(.txt extension, tab or comma delimited, optional quotes around fields, no quotes around records, 0–3 description fields*). Open the text file by choosing **Open** from the **File** menu; the **Open** window appears. At the bottom of the **Open** window is a **Files of type** pop-up menu; choose **Text Cross-reference** or **Any file** from the menu and open the text file. The text file will be displayed in an untitled CRF window. When you save the file, it is converted and saved as a CRF file.

If you are converting a CRF file from an existing database, there are three limitations to be aware of. First, the entire CRF file must be able to fit into Application Builder's memory. Second, there is an absolute maximum number of keys or inputs, which is about 26,000. Third, Application Builder works much faster on CRF files with 6,000 keys or less. If you have a very large CRF file, it would be wise to break it up into several 6,000 record files.

When using a CRF file in an application, the CRF filename (including the .crf extension) must be assigned as the **Matching** property. The assigned CRF file is transferred to the data collector along with the application when you choose **Transfer Application** from the **File** menu.

To insert or delete records in a CRF file, choose **Insert Record** or **Delete Record** from the **Edit** menu. If you are in the last cell of the CRF window (Field 3 of the last record), and you press the <Tab> key, a new record is automatically inserted at the end.

Notes:

Chapter 3

Building an Application with Application Builder

This chapter provides instructions on:

- Creating an application
- Transferring the application to the data collector
- Transferring the application data file from the data collector to the computer
- Using cross-reference files in an application

Getting Started

The following Building an Application section steps you through:

- Using Application Builder to build a security application
- Transferring the application to the data collector
- Entering a sample security round on the data collector
- Using **Vxcom** to transfer the data from the data collector to the computer
- Viewing the data file on the computer
- Using CRF files in the application

Building an Application

1. If you have not already transferred the files from the Application Builder CD to your computer, do so now.

2. Double-click the **Appbuild.exe** icon to run the Application Builder program. The Application Builder menu titles and toolbar appear.

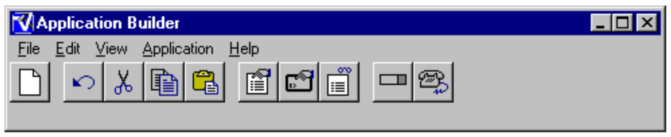


Figure 3-1 Application Builder Menu Titles and Toolbar

3. Click the **New Application** button on the toolbar (Figure 3-2) or choose **New Application** from the **File** menu (Figure 3-3) or type **Ctrl+N**.

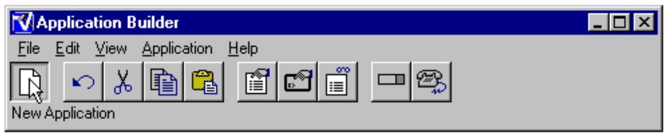


Figure 3-2 Application Builder Toolbar - New Application Button

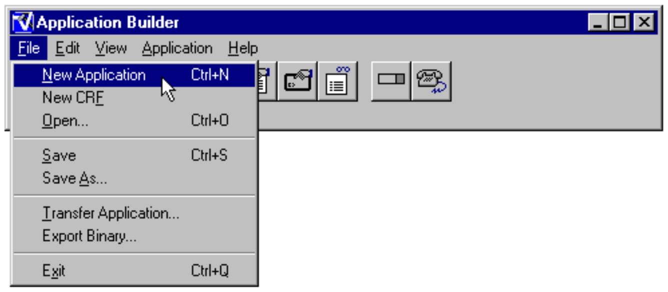


Figure 3-3 File Menu - New Application Command

4. A new application design window opens (Figure 3-4) with one Input Handler ("Begin" Handler) and 64 empty steps.

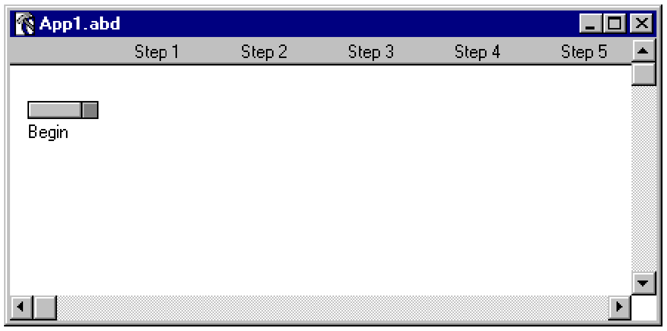


Figure 3-4 New Application Design Window

Security Example

For the security application, we want to know:

- Who the guard is
- What time the security round begins
- What time each location is checked
- The status of the location

This application can be built with three steps. The first step identifies the guard and the time the security round begins, the second step identifies the location and the time the location was checked, and the third step gives the status of the location.

5. Choose **Save As…** from the **File** menu (Figure 3-5).

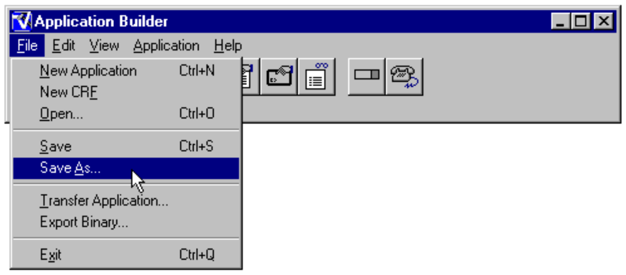
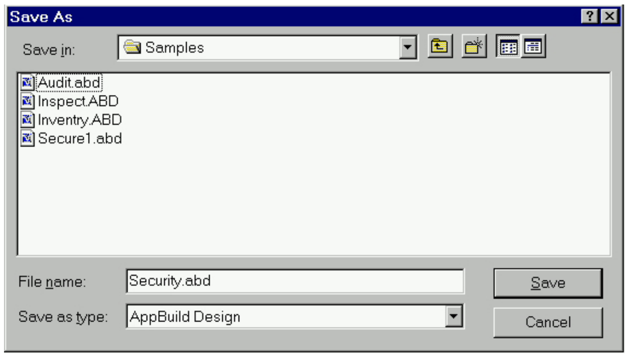


Figure 3-5 File Menu - Save As…

6. Enter **Security.ABD** and click **Save** (Figure 3-6).



File name: Security.abd

Save as type: AppBuild Design

Figure 3-6 Name Application

7. This names and saves the application (Figure 3-7). If you don't enter the .ABD extension, it is automatically added to the end of the filename.

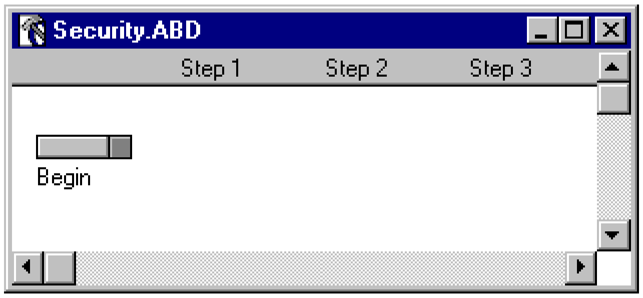


Figure 3-7 New Application Named

8. Click the **New Input Handler** button on the toolbar (Figure 3-8) or choose **New Input Handler** from the **Application** menu (Figure 3-9) or type **Ctrl+E**.

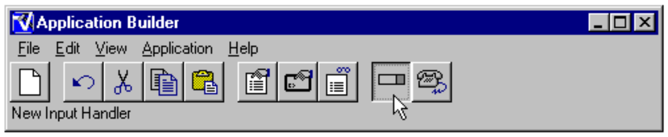


Figure 3-8 Application Builder Toolbar - New Input Handler Button

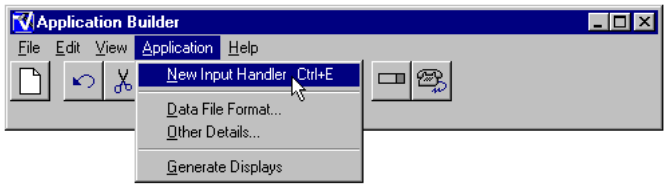


Figure 3-9 Application Menu - New Input Handler Command

9. An untitled Input Handler is added to the application (Figure 3-10).

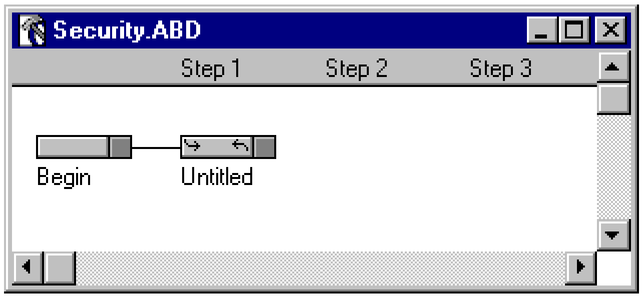


Figure 3-10 New Input Handler Added

10. Since our application needs Input Handlers in Steps 1, 2, and 3, click the **New Input Handler** button on the toolbar <u>two more times</u>. Your application window should appear similar to Figure 3-11.

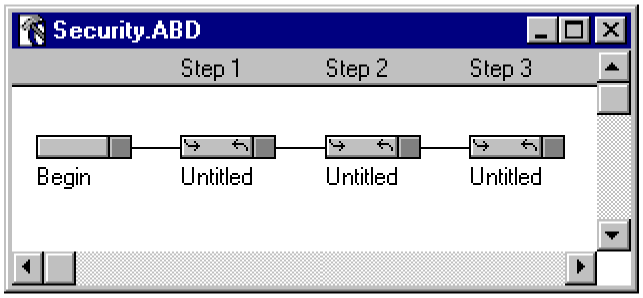


Figure 3-11 Security Application

11. Double-click the word **Untitled** under Step 1 (Figure 3-12). This opens the Input Handler name box so you may edit or change the name.

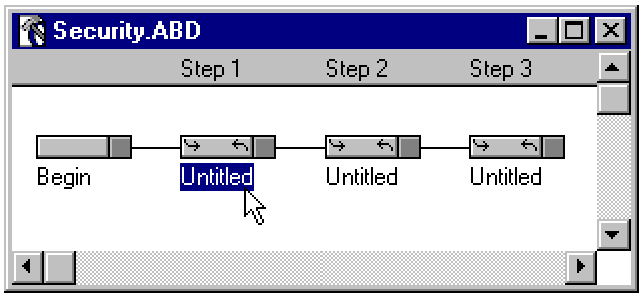


Figure 3-12 Step 1 Field Description Name Box

12. Type **Guard** in the name box. Press the computer's <Enter> key to close the name box. The Input Handler is now named **Guard** (Figure 3-13).

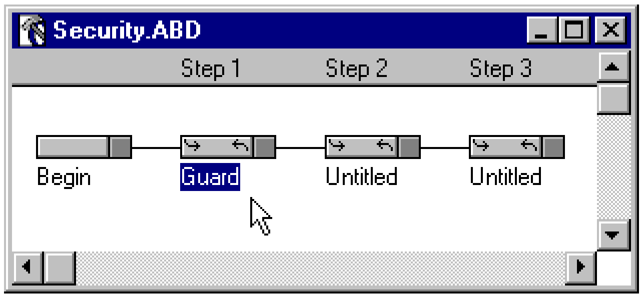


Figure 3-13 Step 1 Input Handler Named

13. Double-click **Untitled** under Step 2 to open the name box. Type **Location** and press <Enter> (Figure 3-14).

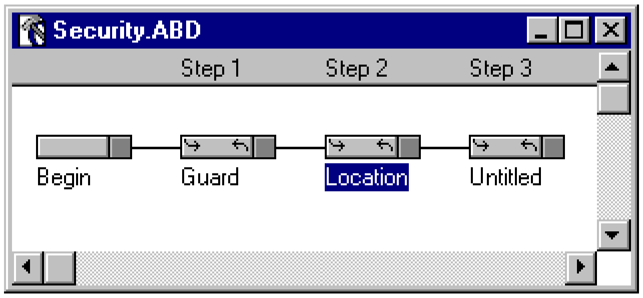


Figure 3-14 Step 2 Input Handler Named

14. Double-click **Untitled** under Step 3 to open the name box; type **Status** and press <Enter> (Figure 3-15).

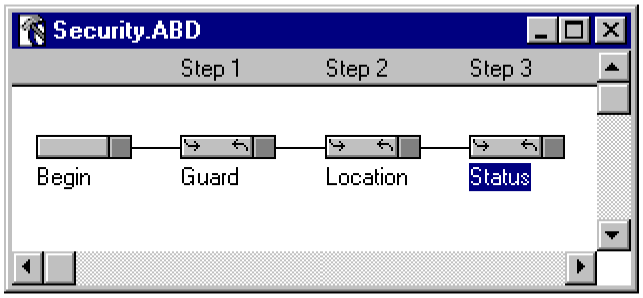


Figure 3-15 Step 3 Input Handler Named

15. Click the **Properties Window** button on the toolbar (Figure 3-16) or choose **Properties Window** from the **View** menu (Figure 3-17).

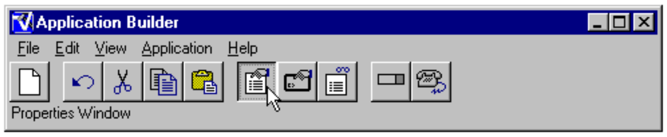


Figure 3-16 Application Builder Toolbar - Properties Window Button

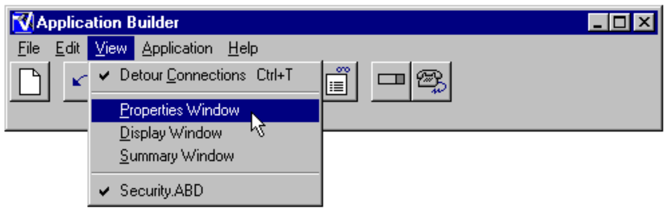


Figure 3-17 View Menu - Properties Window Command

16. The **Properties** window opens (Figure 3-18).

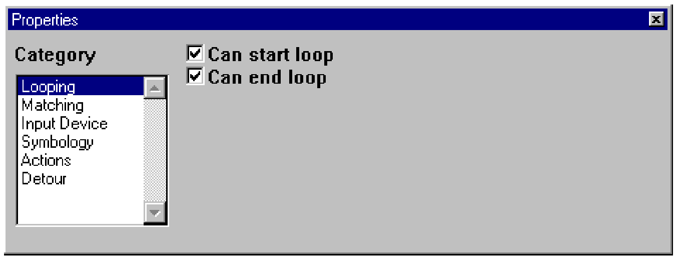


Figure 3-18 Properties Window

17. Click the "Guard" Handler to select it (Figure 3-19).

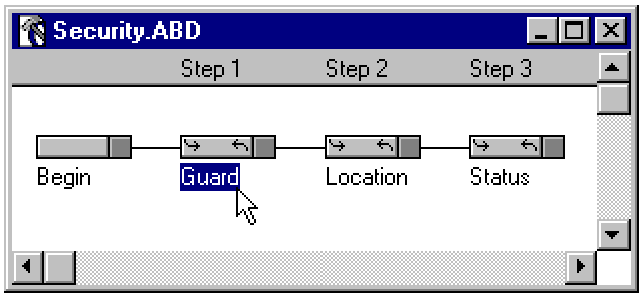
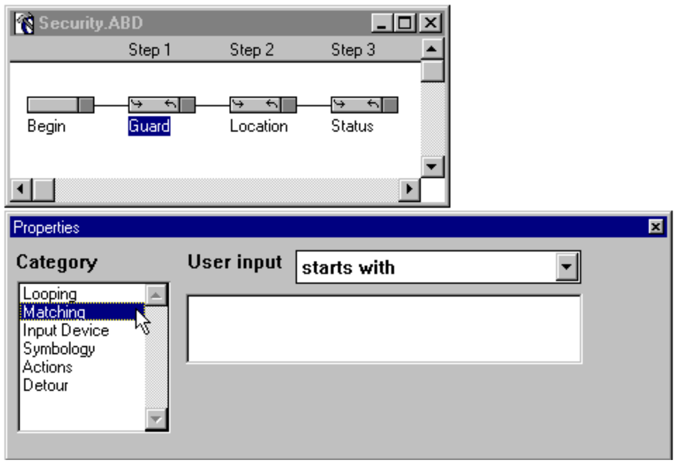


Figure 3-19 Select "Guard" Handler

In this example, we will use the default values for all of the properties except **Actions** and **Matching** (which do not have defaults).

18. Select **Matching** from the **Properties** window (Figure 3-20).



User input: starts with

Figure 3-20 "Guard" Handler - Matching Property

When creating an application program, there needs to be something unique about the bar codes or entry for each Input Handler; this allows the data collector to know that the match criterion has been met for the current Handler and that it can advance to the next Input Handler.

19. Choose **starts with** from the **Matching** property pop-up menu (Figure 3-21). For the "Guard" Handler, we will use bar codes that start with the letter G. So, we will use **starts with G** for our match criterion.

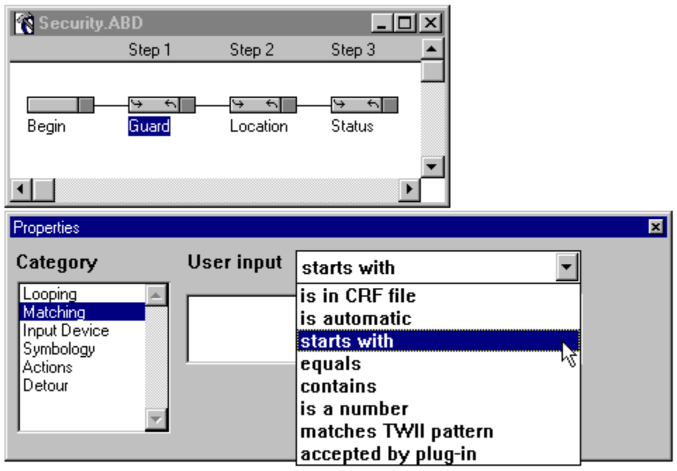
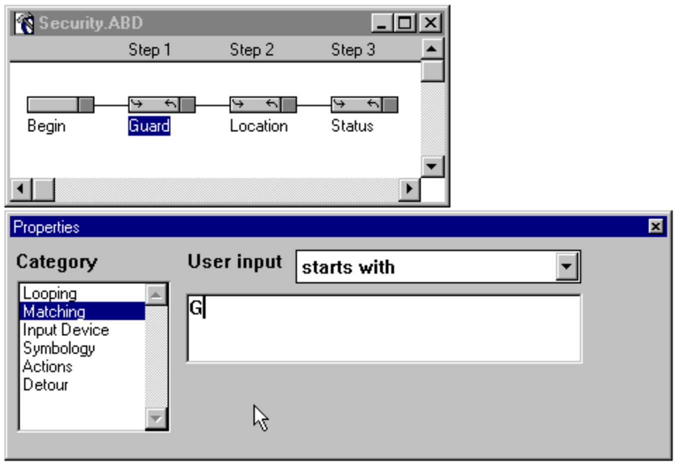


Figure 3-21 Matching Property - Starts With

20. Click in the edit box and enter a **G** (Figure 3-22). Any entry that starts with a **G** is now an acceptable entry for the "Guard" Handler.



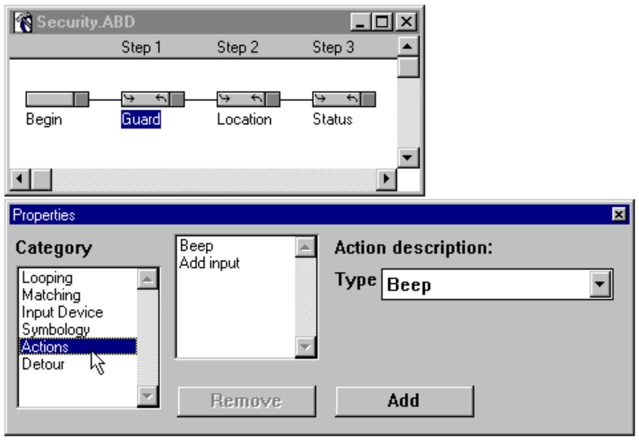
User input: starts with

G

Figure 3-22 Matching Property - Starts With G

21. Select **Actions** from the **Properties** window (Figure 3-23).

For the "Guard" Handler, we want the date and time to be recorded with the entry. To do this, we will include the **Add time to record** and the **Add date to record** actions to the action list.



Beep
Add input

Action description:

Type: Beep

Figure 3-23 "Guard" Handler - Actions Property

22. Select **Add time to record** from the **Action description** pop-up menu (Figure 3-24).

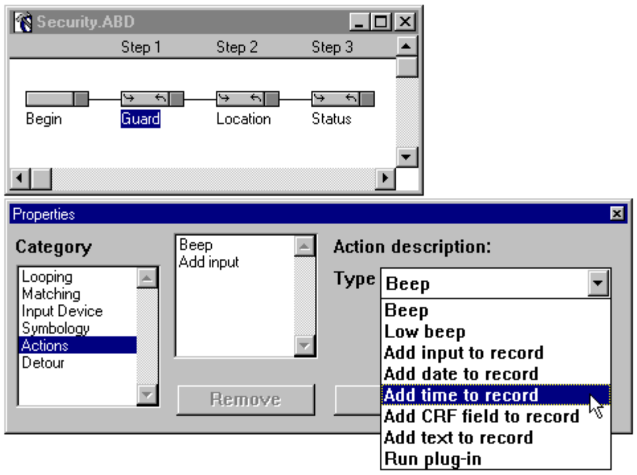


Figure 3-24 Actions Property - Add Time to Record

23. Click **Add** (Figure 3-25) to add the **Add time to record** action to the action list (Figure 3-26).

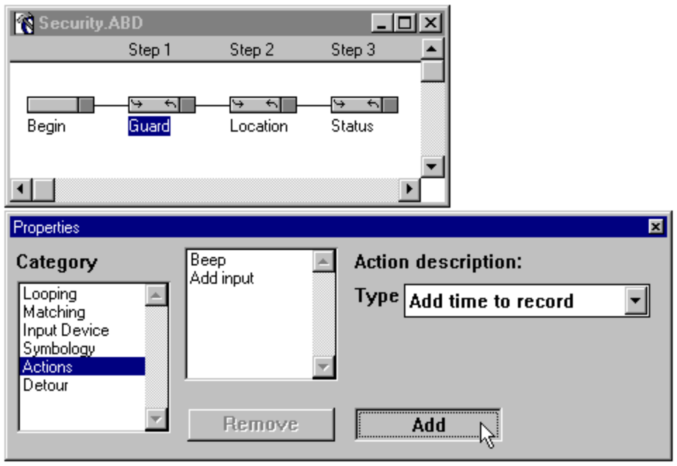


Figure 3-25 Actions Property - Add Action to Action List

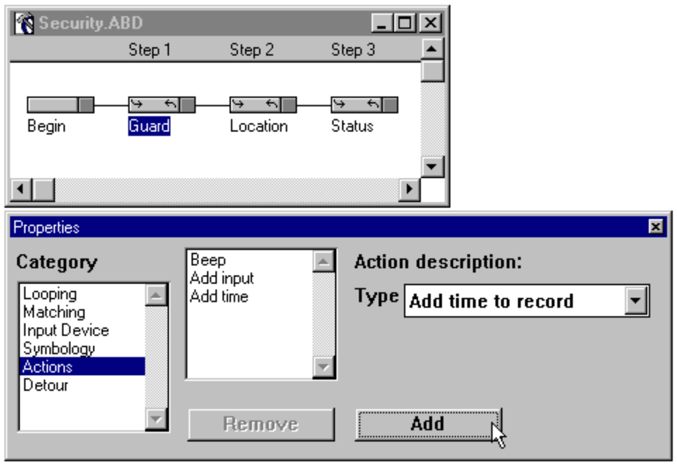


Figure 3-26 Actions Property - Action Added to Action List

24. Select **Add date to record** from the **Action description** pop-up menu (Figure 3-27).

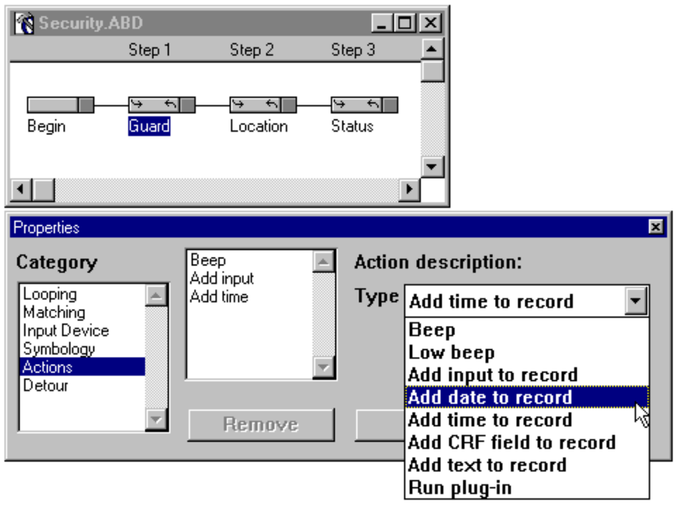


Figure 3-27 Actions Property - Add Date to Record

25. Click **Add** (Figure 3-28) to add the **Add date to record** action to the action list (Figure 3-29).

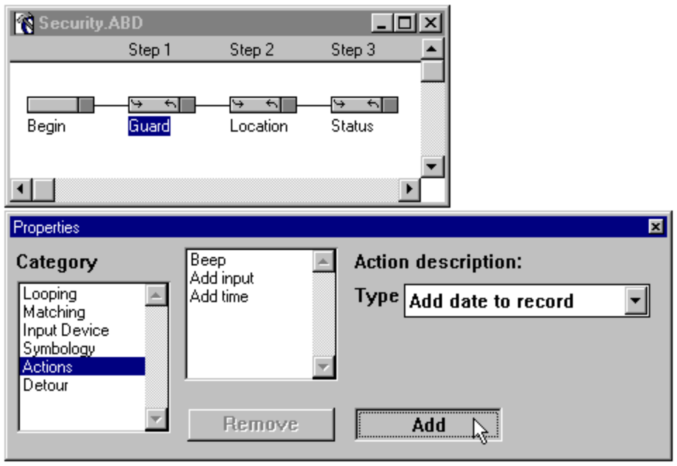


Figure 3-28 Actions Property - Add Action to Action List

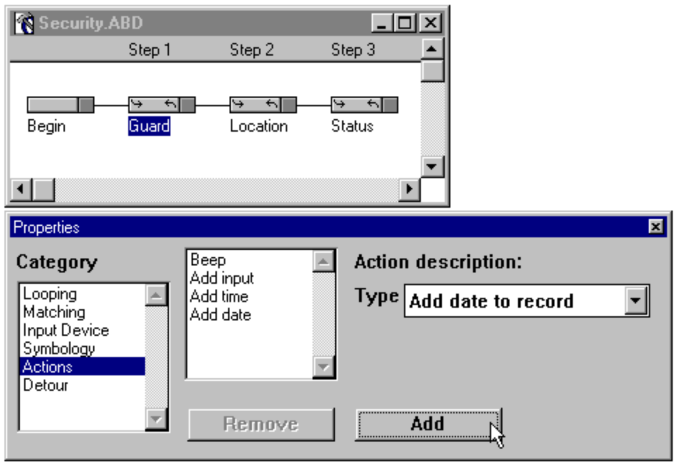
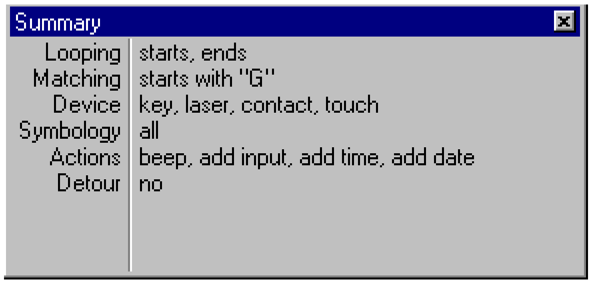


Figure 3-29 Actions Property - Action Added to Action List

26. The **Summary** window for the “Guard” Handler is shown in Figure 3-30.



Looping	starts, ends
Matching	starts with "G"
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add input, add time, add date
Detour	no

Figure 3-30 Summary Window - “Guard” Handler

“Guard” Handler Summary Window:
Looping - Can start and end a loop.
Matching - Entry must start with the letter G.
Device - Must be entered by keypad, laser, contact, or touch.
Symbology - Any of the following bar code symbologies are acceptable: Code 3 of 9, Interleaved 2 of 5, Codabar, UPC, EAN, Code 128.
Actions - When the input matches the above requirements, the data collector will perform these actions: **Beep**, **Add input to record**, **Add time to record**, and **Add date to record**.
Detour - This Input Handler is not a Detour Handler.

We will use these settings for the “Guard” Handler in this example. Next, we will set the properties for the “Location” Handler.

27. Click the “Location” Handler to select it (Figure 3-31).

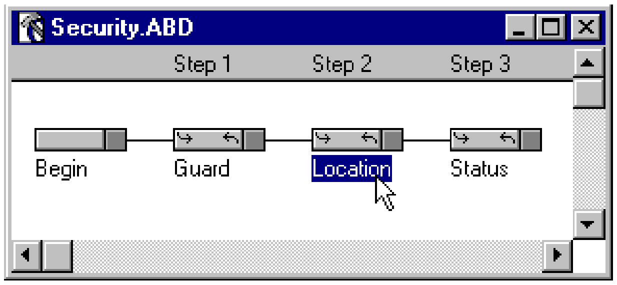
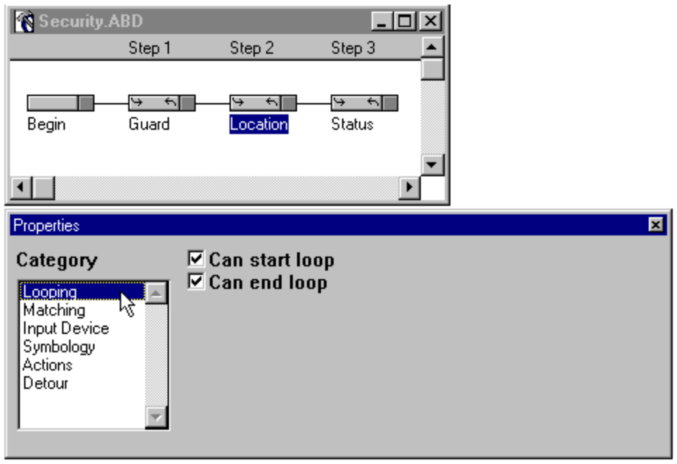


Figure 3-31 Select “Location” Handler

For the “Location” Handler we will use the default properties, except for **Looping**, **Matching**, and **Actions**.

For this security application, we want various locations checked and the status of each location to be entered. We want the application to require a status entry if a location entry is made. To do this, we will not let the “Location” Handler “end a loop.” By doing this, the application will not let the “Location” Handler be the final entry; it demands that another entry (Status) be made before the current record is completed and added to the data file.

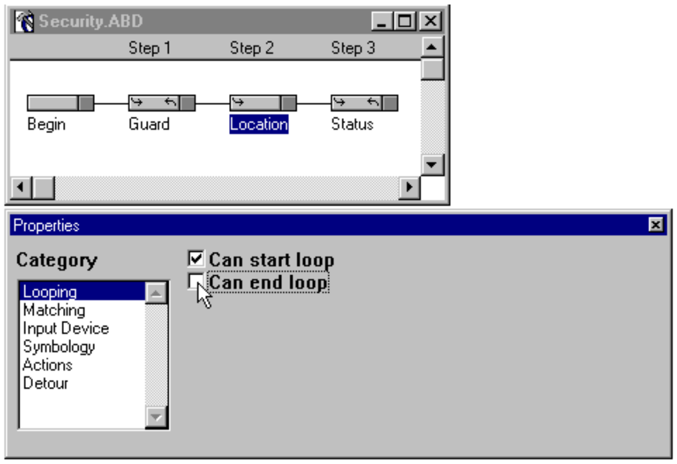
28. Select **Looping** from the **Properties** window **Category** list (Figure 3-32).



☑ Can start loop
☑ Can end loop

Figure 3-32 “Location” Handler - Looping Property

29. Click the **Can end loop** check box to deselect the option (Figure 3-33). *(A checkmark should not appear in the box.)*



☑ Can start loop
☐ Can end loop

Figure 3-33 “Location” Handler - Looping
(Can Start Loop - Cannot End Loop)

30. Select **Matching** from the **Properties** window (Figure 3-34).

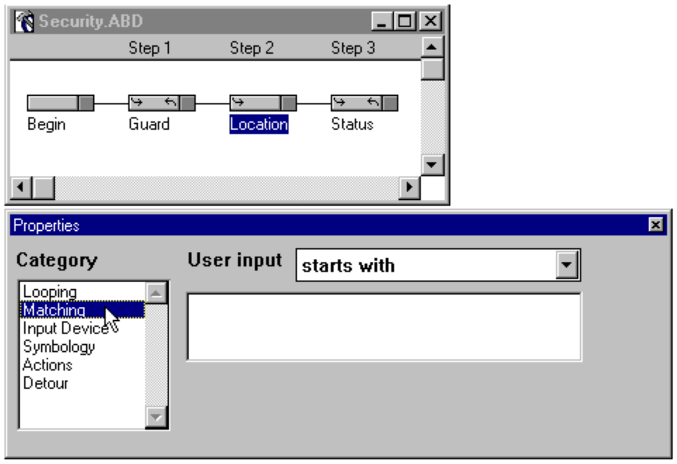


Figure 3-34 "Location" Handler - Matching Property

31. The unique match criterion that we have selected for the Location bar codes is they start with the letter D. So, we will use **starts with D** for our "Location" Handler match criterion.

    Choose **starts with** from the **Matching** property pop-up menu (Figure 3-35).

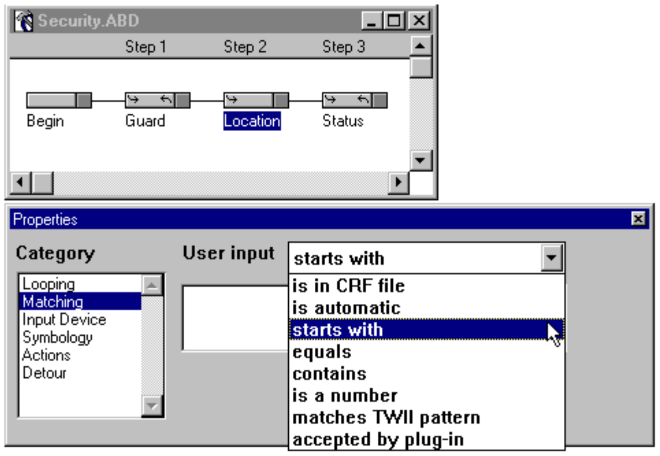
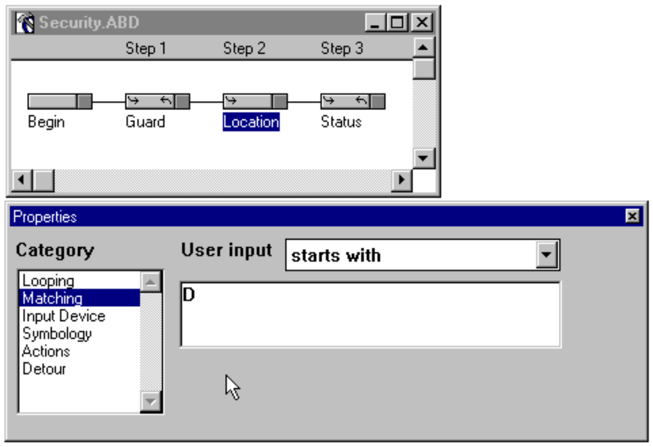


Figure 3-35 Matching Property - Starts With

32. Click in the edit box and enter a **D** (Figure 3-36). Any entry that starts with a **D** is now an acceptable entry for the “Location” Handler.



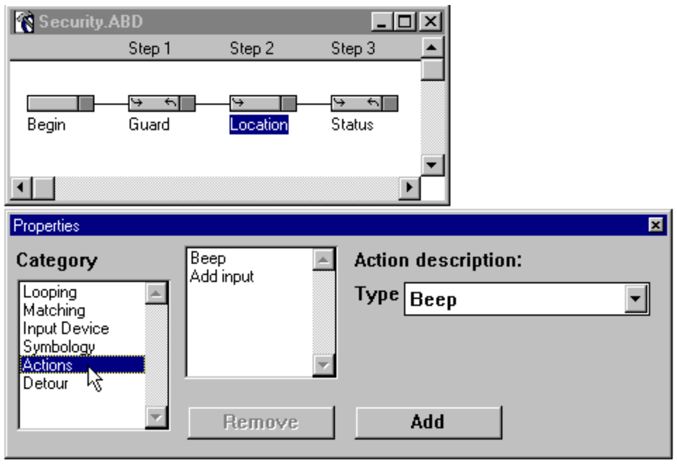
User input: starts with

D

Figure 3-36 Matching Property - Starts With D

33. Select **Actions** from the **Properties** window (Figure 3-37).

For the “Location” Handler, we also want the date and time recorded with the entry. To do this, we will include the **Add time to record** and the **Add date to record** actions to the action list.



Beep
Add input

Action description:
Type: Beep

Figure 3-37 “Location” Handler - Actions Property

34. Select **Add time to record** from the **Action description** pop-up menu (Figure 3-38).

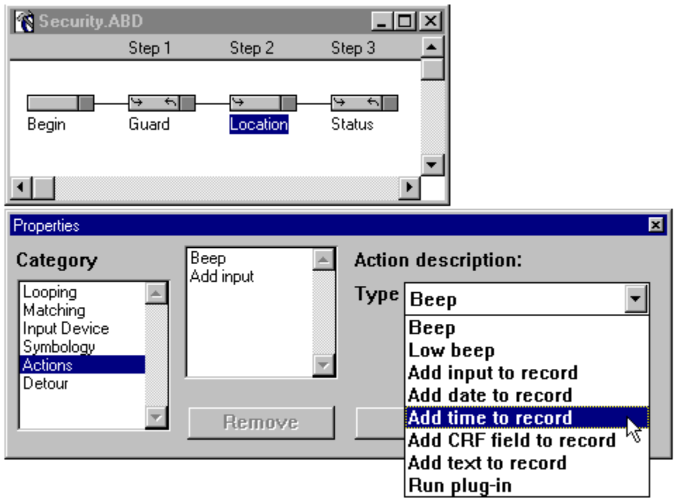
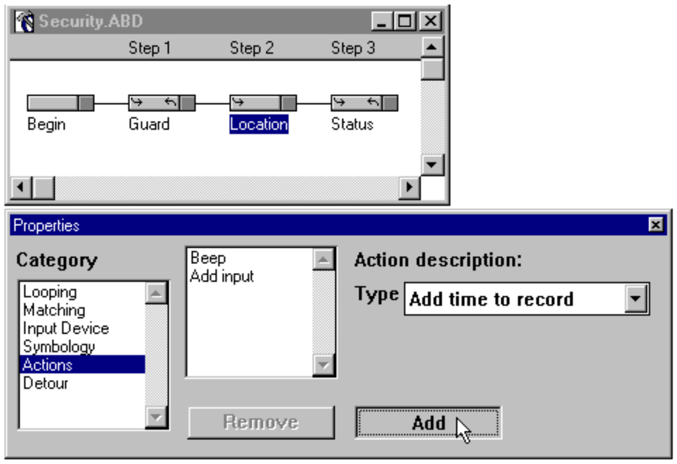


Figure 3-38 Actions Property - Add Time to Record

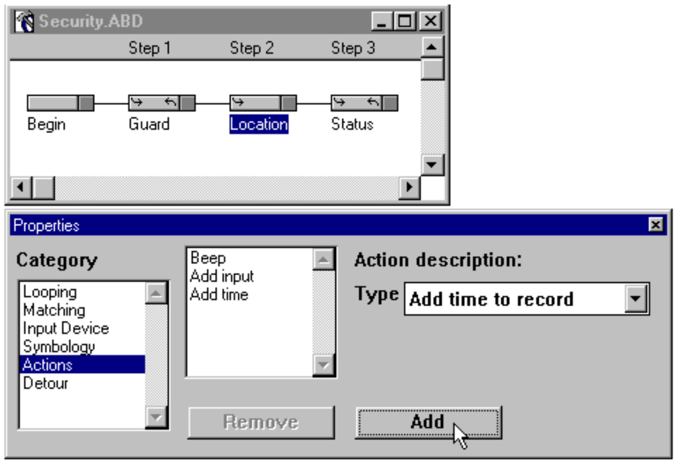
35. Click **Add** (Figure 3-39) to add the **Add time to record** action to the action list (Figure 3-40).



Action description:

Type Add time to record

Figure 3-39 Actions Property - Add Action to Action List



Action description:

Type Add time to record

Figure 3-40 Actions Property - Action Added to Action List

36. Select **Add date to record** from the **Action description** pop-up menu (Figure 3-41).

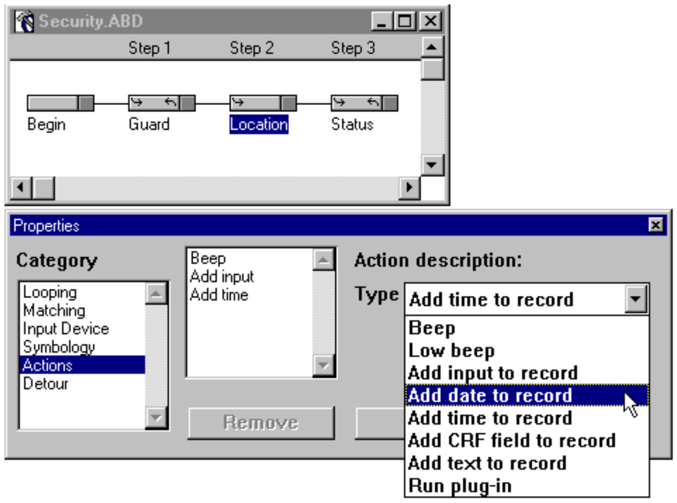


Figure 3-41 Actions Property - Add Date to Record

37. Click **Add** (Figure 3-42) to add the **Add date to record** action to the action list (Figure 3-43).

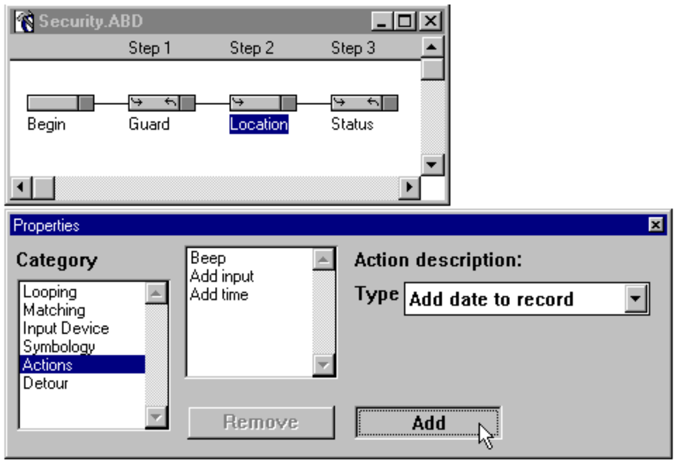


Figure 3-42 Actions Property - Add Action to Action List

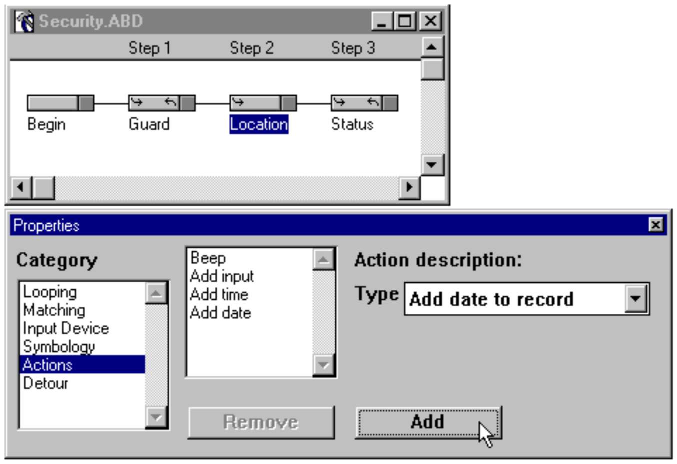
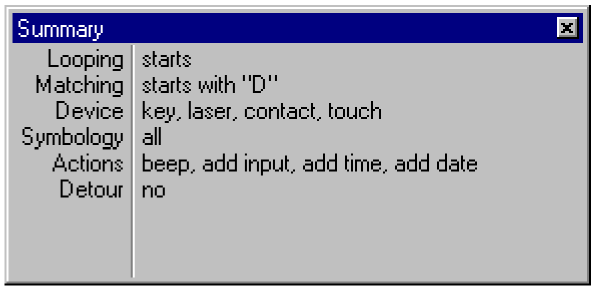


Figure 3-43 Actions Property - Action Added to Action List

38. The **Summary** window for the “Location” Handler is shown in Figure 3-44.



Looping	starts
Matching	starts with "D"
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add input, add time, add date
Detour	no

Figure 3-44 Summary Window - “Location” Handler

“Location” Handler Summary Window:
Looping - Can start a loop; cannot end a loop.
Matching - Entry must start with the letter D.
Device - Must be entered by keypad, laser, contact, or touch.
Symbology - Any of the following bar code symbologies are acceptable: Code 3 of 9, Interleaved 2 of 5, Codabar, UPC, EAN, Code 128.
Actions - When the input matches the above requirements, the data collector will perform these actions: **Beep**, **Add input to record**, **Add time to record**, and **Add date to record**.
Detour - This Input Handler is not a Detour Handler.

We will use these settings for the “Location” Handler in this example. Next, we will set the properties for the “Status” Handler. We will use the property defaults, except for the **Looping** and **Matching** properties.

39. Click the “Status” Handler to select it (Figure 3-45).

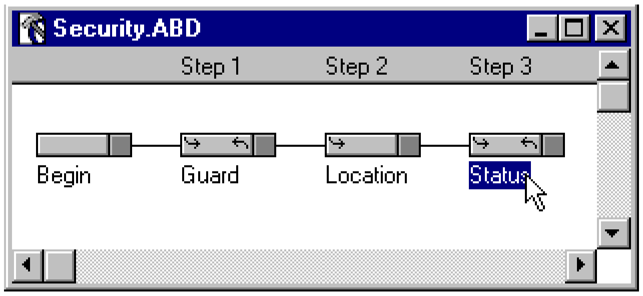
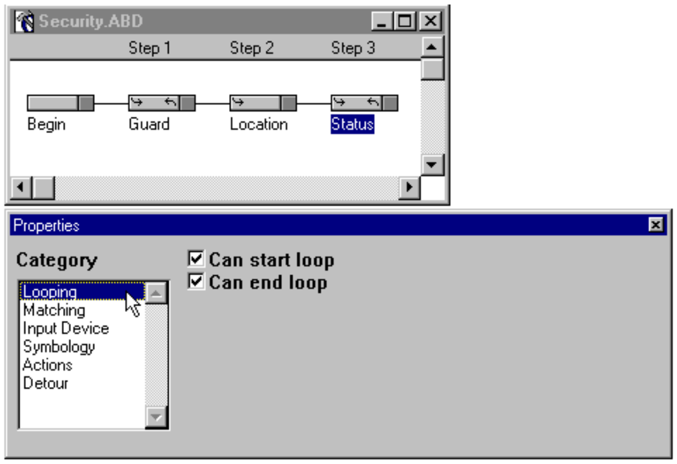


Figure 3-45 Select “Status” Handler

For the “Status” Handler we will change the **Looping** and **Matching** properties.

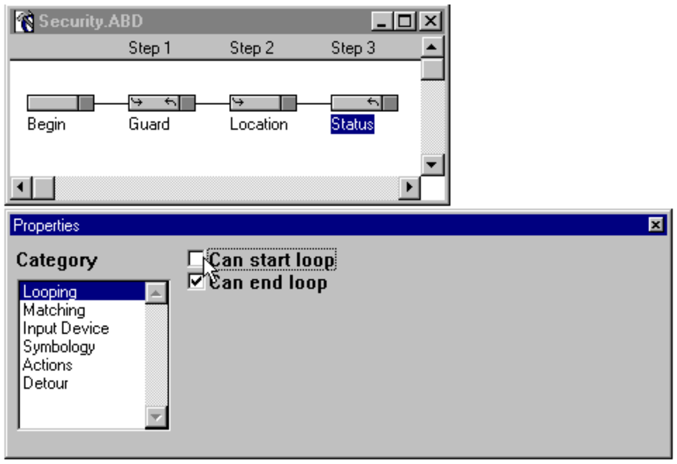
40. Select **Looping** from the **Properties** window **Category** list (Figure 3-46). Since we only want a Status entry if a Location has been entered, we will not allow the “Status” Handler to begin a loop; this will require that a Location entry be made before a Status entry can be made.



☑ Can start loop
☑ Can end loop

Figure 3-46 “Status” Handler - Looping Property

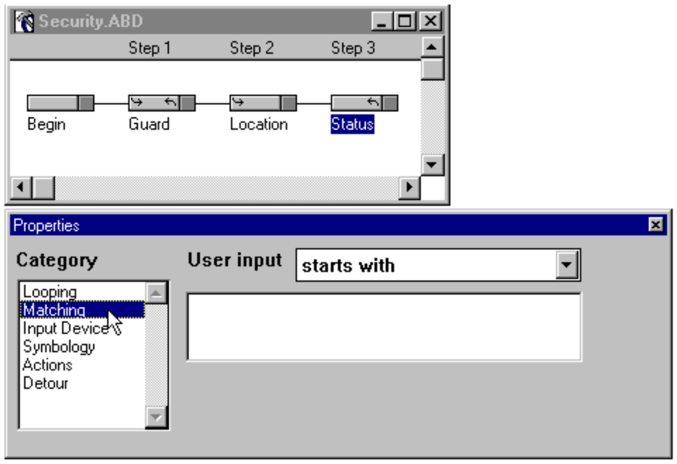
41. Click the **Can start loop** check box to deselect the option (Figure 3-47). *(A checkmark should not appear in the box.)*



- [ ] Can start loop
- [✓] Can end loop

Figure 3-47 "Status" Handler - Looping
(Cannot Start Loop - Can End Loop)

42. Select **Matching** from the **Properties** window (Figure 3-48).



User input: starts with

Figure 3-48 "Status" Handler - Matching Property

43. The unique match criterion that we have selected for the Status bar codes is they start with the letter **S**. So, we will use **starts with S** for our "Status" Handler match criterion.

    Choose **starts with** from the **Matching** property pop-up menu (Figure 3-49).

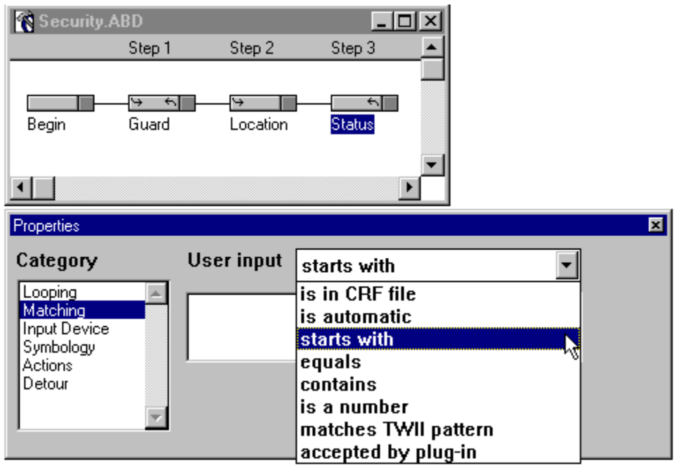
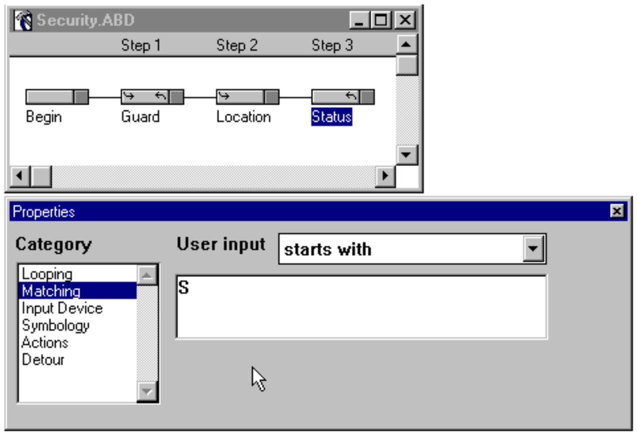


Figure 3-49 Matching Property - Starts With

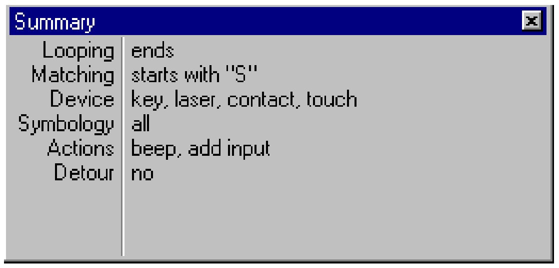
44. Click in the edit box and enter an **S** (Figure 3-50). Any entry that starts with an **S** is now an acceptable entry for the "Status" Handler.



Category: Matching
User input: starts with
S

Figure 3-50 Matching Property - Starts With S

45. The **Summary** window for the "Status" Handler is shown in Figure 3-51.



Summary

Looping	ends
Matching	starts with "S"
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add input
Detour	no

Figure 3-51 Summary Window - "Status" Handler

“Status” Handler Summary Window:
Looping - Cannot start a loop; can end a loop.
Matching - Entry must start with the letter S.
Device - Must be entered by keypad, laser, contact, or touch.
Symbology - Any of the following bar code symbologies are acceptable: Code 3 of 9, Interleaved 2 of 5, Codabar, UPC, EAN, Code 128.
Actions - When the input matches the above requirements, the data collector will perform these actions: **Beep** and **Add input to record**.
Detour - This Input Handler is not a Detour Handler.

Next, we will create the data collector's display for each Input Handler. There are two ways to do this: you can either select an Input Handler and create your own display design with the **Display** window; or, you can let the program automatically generate the display.

In this example, we will let the program automatically generate the displays for the Input Handlers.

46. Select all of the Input Handlers by choosing **Select All Handlers** from the **Edit** menu (Figure 3-52), typing **Ctrl+A,** Shift-clicking each Input Handler, or dragging a rectangle around all of the Input Handlers.

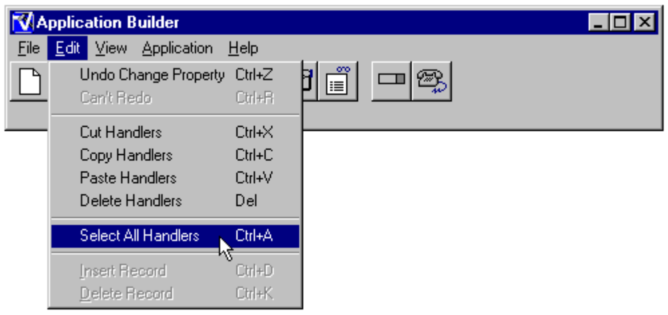


Figure 3-52 Edit Menu - Select All Handlers

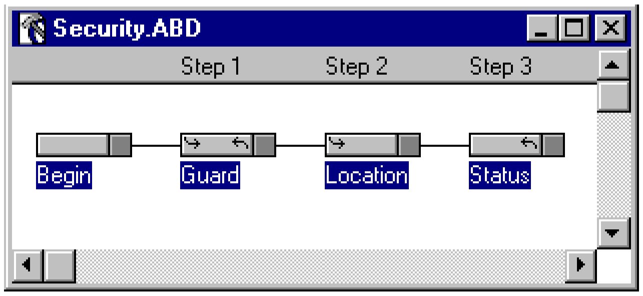


Figure 3-53 All Handlers Selected

47. Choose **Generate Displays** from the **Application** menu (Figure 3-54).

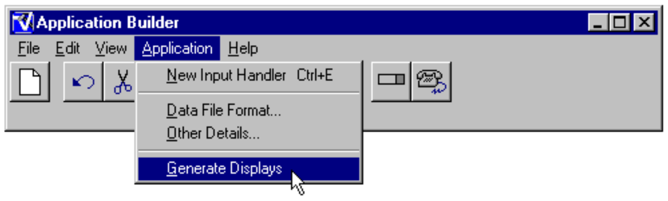


Figure 3-54 Application Menu - Generate Displays Command

48. The message window in Figure 3-55 appears.

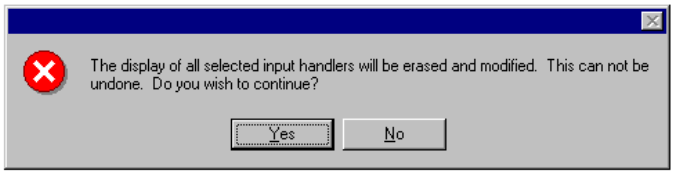

The display of all selected input handlers will be erased and modified. This can not be undone. Do you wish to continue?

Figure 3-55 Generate Displays Warning Message Window

49. Click **Yes** or press <Enter> to automatically generate the displays (Figure 3-56). *(Clicking* ***No*** *would close the window without generating displays.)*

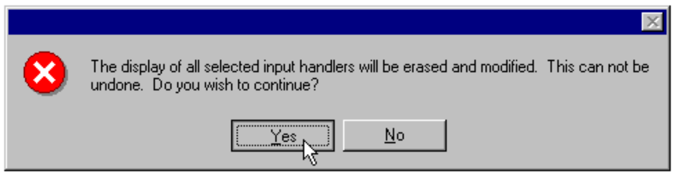

The display of all selected input handlers will be erased and modified. This can not be undone. Do you wish to continue?

Figure 3-56 Generate Displays

50. Click the **Display Window** button on the toolbar (Figure 3-57) or choose **Display Window** from the **View** menu (Figure 3-58).

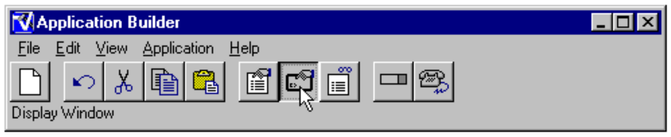


Figure 3-57 Application Builder Toolbar - Display Window Button

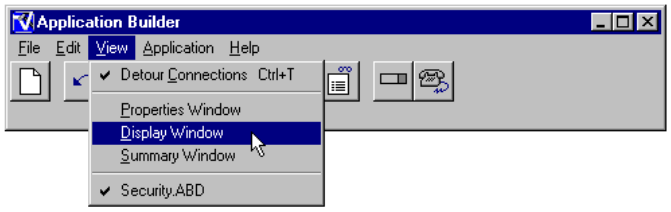


Figure 3-58 View Menu - Display Window Command

51. The **Display** window opens (Figure 3-59).

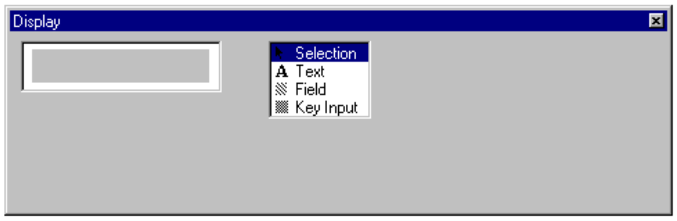


Figure 3-59 Display Window

52. Click the “Begin” Handler (Figure 3-60) to select it. The generated display is shown in the **Display** window (Figure 3-61). The generated display asks for a guard entry on the second line.

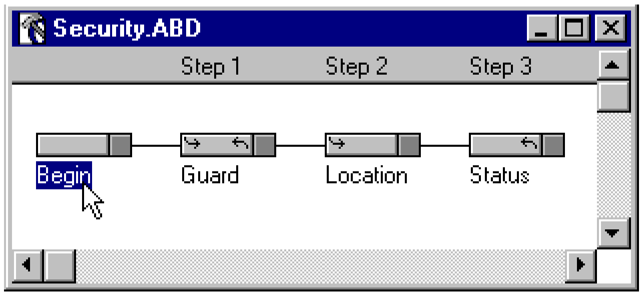


Figure 3-60 Security Application - “Begin” Handler

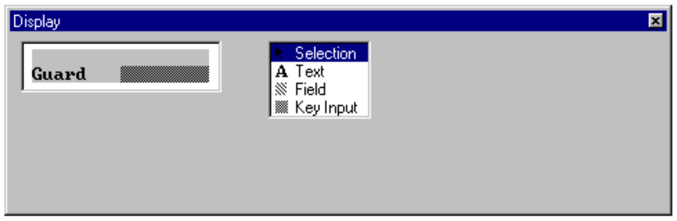


Figure 3-61 “Begin” Handler Display

53. Click the "Guard" Handler (Figure 3-62) to select it. The generated display is shown in the **Display** window (Figure 3-63). The generated display shows the guard entry on the first line and asks for a location entry on the second line.

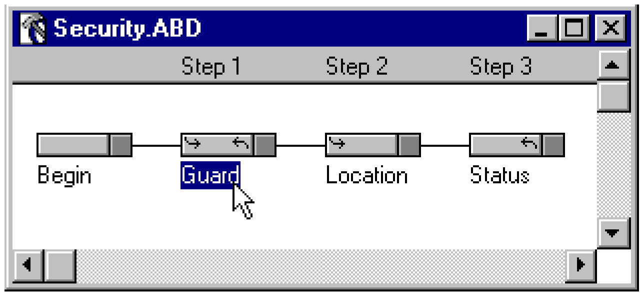


Figure 3-62 Security Application - "Guard" Handler

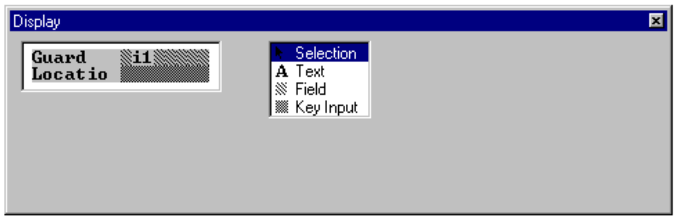


Figure 3-63 "Guard" Handler Display

54. The **Generate Displays** command automatically truncates the Handler names to seven characters in the display; this is why **Location** appears as **Locatio** in the display. To correct this, select **Text** from the **Display** window tool palette. The text entries are shown in the edit box (Figure 3-64).

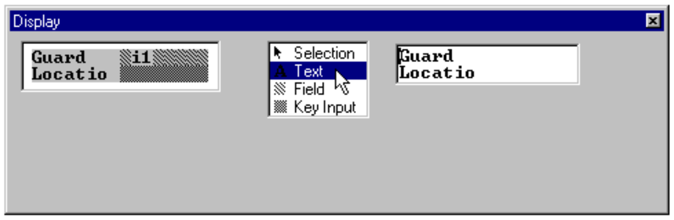


Figure 3-64 "Guard" Handler Display - Text Tool Selected

55. Click at the end of **Locatio** in the edit box (Figure 3-65).

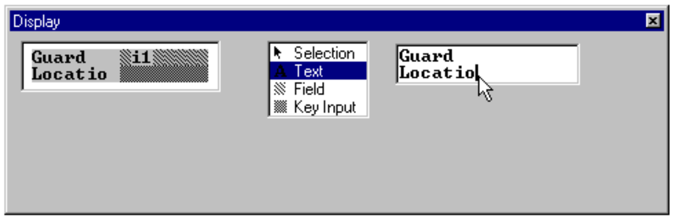


Figure 3-65 Editing “Guard” Handler Display

56. Type an “**n**” (Figure 3-66).

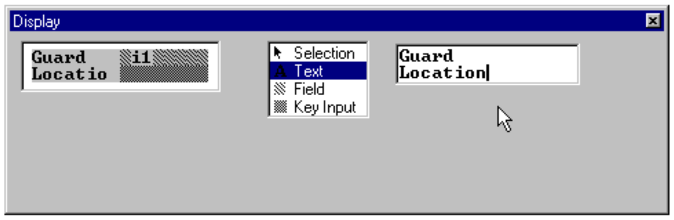


Figure 3-66 “Guard” Handler Display - Location Corrected in Edit Box

57. Press <Enter>; the display now shows **Location** on the second line (Figure 3-67).

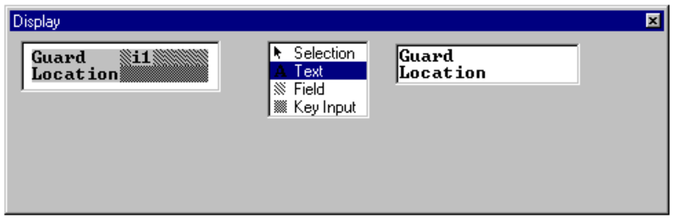


Figure 3-67 “Guard” Handler Display - Location Corrected in Display

58. Click the “Location” Handler (Figure 3-68) to select it. The generated display is shown in the **Display** window (Figure 3-69). The generated display shows the location on the first line and asks for the status of the location on the second line.

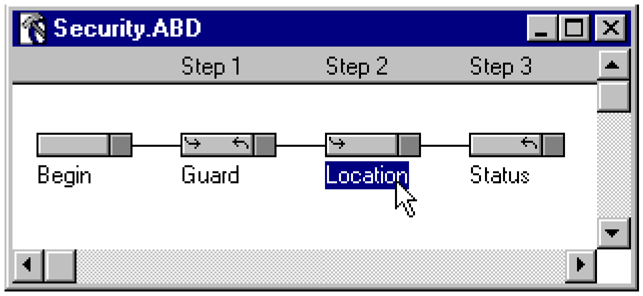


Figure 3-68 Security Application - “Location” Handler

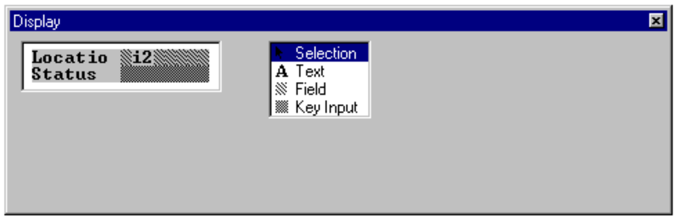


Figure 3-69 “Location” Handler Display

59. To correct **Locatio** in the display, select **Text** from the **Display** window tool palette. The text entries are shown in the edit box (Figure 3-70).

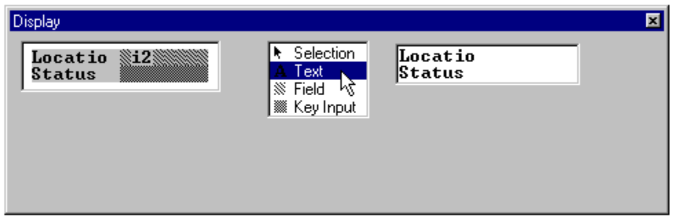


Figure 3-70 “Location” Handler Display - Text Tool Selected

60. Type an “**n**” at the end of **Locatio** in the edit box (Figure 3-71).

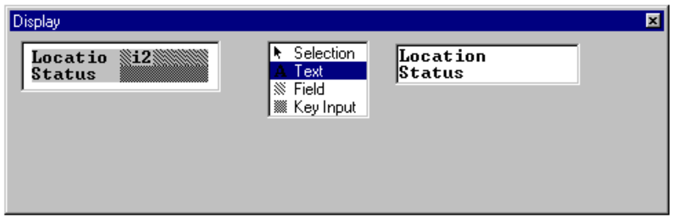


Figure 3-71 “Location” Handler Display - Location Corrected in Edit Box

61. Press <Enter>; the display now shows **Location** on the first line (Figure 3-72).

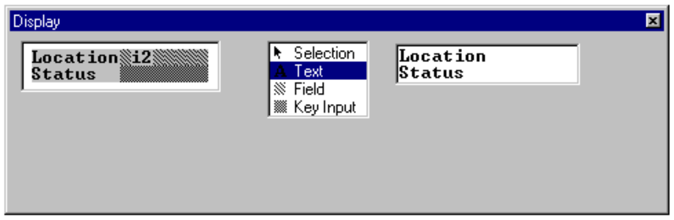


Figure 3-72 “Location” Handler Display - Location Corrected in Display

62. Click the “Status” Handler (Figure 3-73) to select it. The generated display is shown in the **Display** window (Figure 3-74). The generated display shows status of the location on the first line and asks for the next location on the second line.

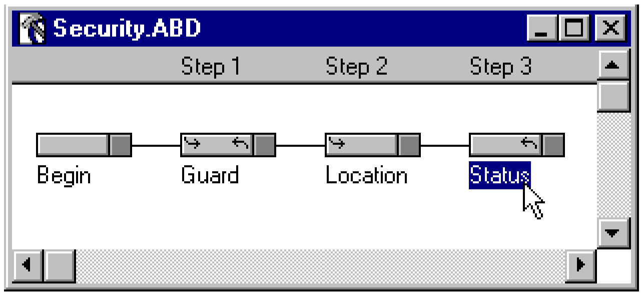
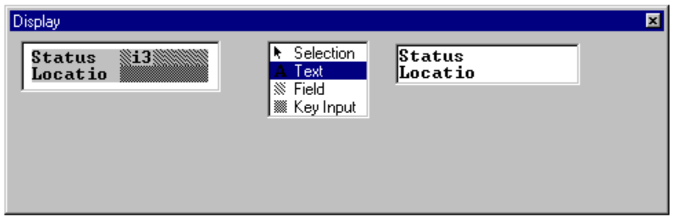


Figure 3-73 Security Application - “Status” Handler

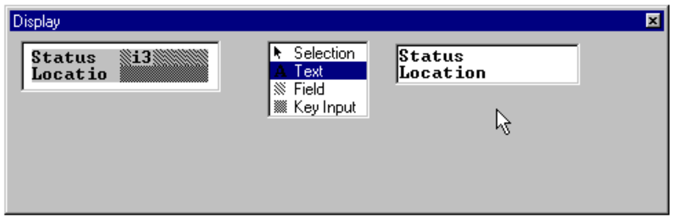


```
Status  i3
Locatio
```

```
Status
Locatio
```

Figure 3-74 “Status” Handler Display

63. To correct the display, type an “**n**” at the end of **Locatio** in the edit box (Figure 3-75).



```
Status  i3
Locatio
```

```
Status
Location
```

Figure 3-75 “Status” Handler Display - Location Corrected in Edit Box

64. Press <Enter>; the display now shows **Location** on the second line (Figure 3-76).

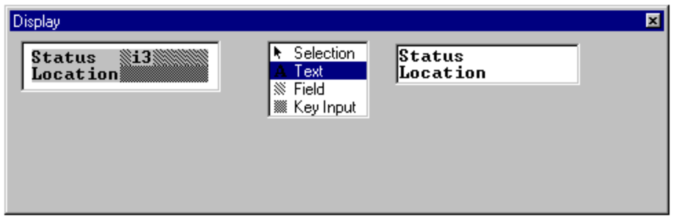


Figure 3-76 "Status" Handler Display - Location Corrected in Display

65. Choose **Save** from the **File** menu (Figure 3-77) or type **Ctrl+S**. This saves the security application you have just created.

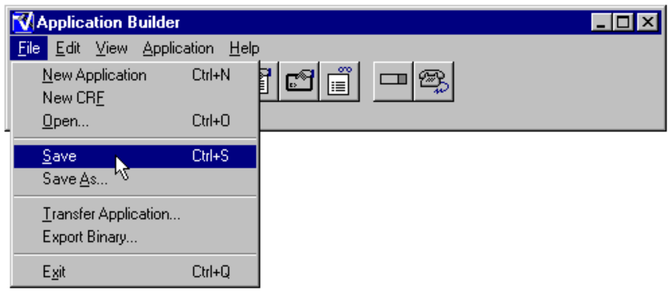


Figure 3-77 File Menu - Save Command

66. To transfer the application to the data collector, insert the data collector into the Base Station or point the data collector's IR transmitter (located in end cap) towards the computer's IR station at an approximate four-inch distance.

67. Click the **Transfer Application** button on the toolbar (Figure 3-78) or choose **Transfer Application** from the **File** menu (Figure 3-79).

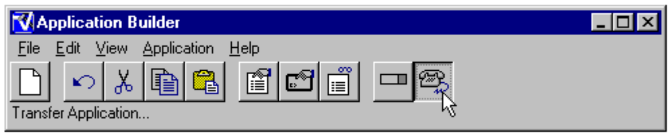


Figure 3-78 Application Builder Toolbar - Transfer Application

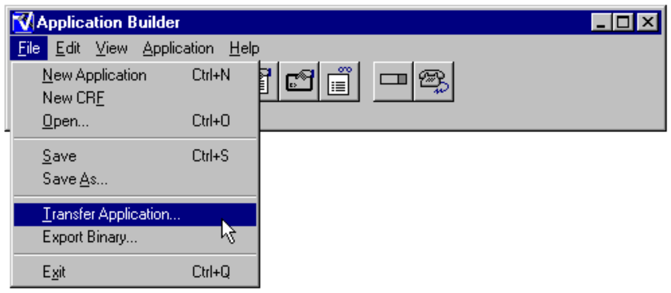


Figure 3-79 File Menu - Transfer Application Command

68. The **File Transfer** window is displayed (Figure 3-80). If you are using a Base Station, select the serial port it is connected to and the **Videx Downloader** configuration. If you are using an external IR transceiver, select the serial port it is connected to and **Use Serial Port** configuration.

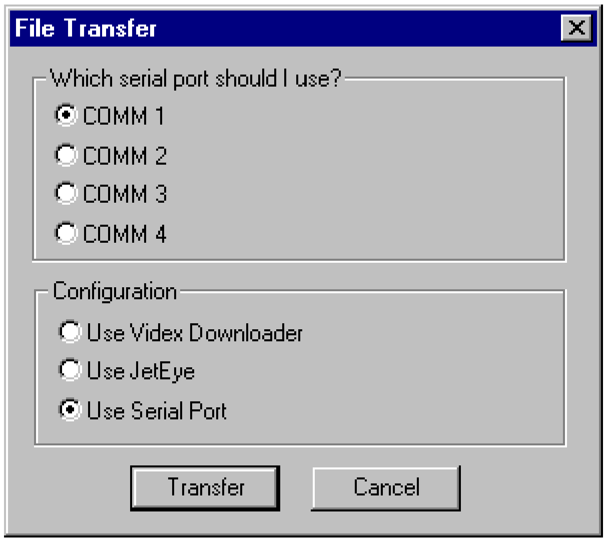
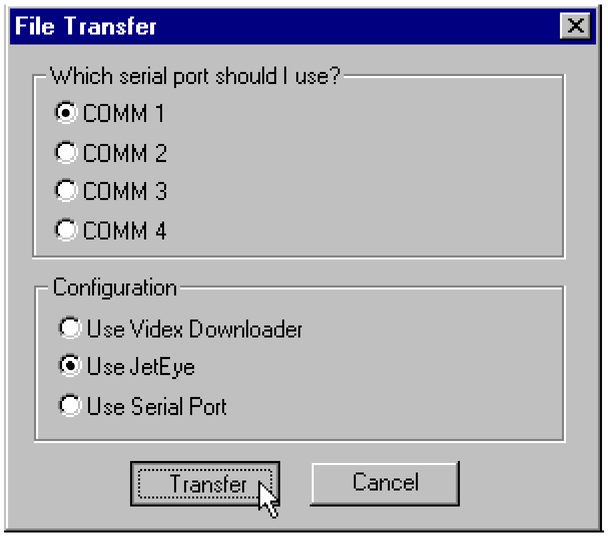


Figure 3-80 File Transfer Window

69. After you have selected the serial port and configuration, click **Transfer** at the **File Transfer** window (Figure 3-81).


File Transfer

Which serial port should I use?

(● COMM 1) COMM 2 COMM 3 COMM 4

Configuration

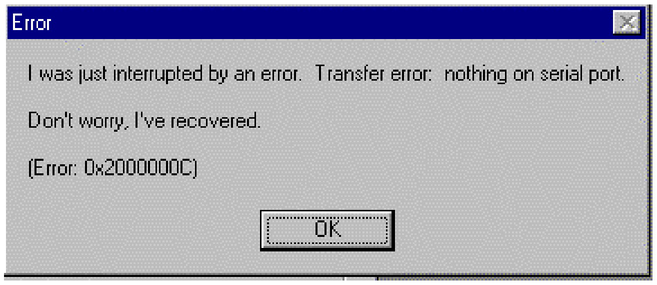
Use Videx Downloader (● Use JetEye) Use Serial Port

Figure 3-81 File Transfer Window

70. Press any key on the data collector to begin the transfer. The application transfers to the data collector.

71. A transfer progress message appears showing the status. The message is removed after the transfer process is complete. If the transfer is successful, go to Step 73; if the transfer is not successful, go to Step 72.

72. The message shown in Figure 3-82 is displayed if the file does not transfer successfully. Click **OK** to clear the message.



```
I was just interrupted by an error.  Transfer error:  nothing on serial port.

Don't worry, I've recovered.

(Error: 0x2000000C)
```

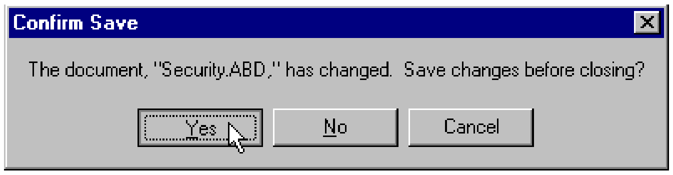
Figure 3-82 Transfer Not Successful Message

Try these troubleshooting tips if the application does not transfer:

- Are the correct computer serial port and IR device configuration selected at the **File Transfer** window?
- Is the Base Station or IR Station correctly connected to your computer?
- If using a Base Station, is the station at least 3 inches away from the computer and monitor?
- If using a Base Station, reset it by unplugging it from the electric outlet and plugging it back in again.
- If using an external IR transceiver, is the data collector's IR transmitter (located in the end cap) pointing towards the IR station, at an approximate 4 inch distance?

Reattempt to transfer the application; if you are still not able to transfer the application, contact the Videx Technical Support Department for assistance.

73. Choose **Exit** from the **File** menu. The program asks if you want to save the document “Security.ABD” before closing; click **Yes** (Figure 3-83).



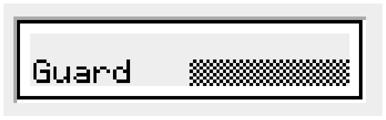
Confirm Save

The document, "Security.ABD," has changed. Save changes before closing?

Figure 3-83 Closing Application Dialog

74. To run the sample application, press any key on the data collector. You will hear an upscale tone; this tone indicates an application is installed and ready to be used.

75. The data collector shows the application’s beginning display (Figure 3-84).



Guard

Figure 3-84 Security Application Begin Display

76. Use the bar codes in Figure 3-85 to enter a security round. Scan a guard ID bar code first. You may then scan another guard ID bar code or a location bar code followed by a status bar code.

Guard ID Bar Codes:

G101 *G102*

G101 *G102*

Location Bar Codes:

D14 *D17*

D14 *D17*

D26 *D34*

D26 *D34*

D47

D47

Status Bar Codes:

S91 Locked *S92* Unlocked

S93 Called Police *S94* Passed

S95 Maintenance Required

Figure 3-85 Security Example Bar Codes

77. The data is saved in the data collector's memory. To view the data, it can be transferred to the computer or viewed on display by using the scroll up and scroll down keys.

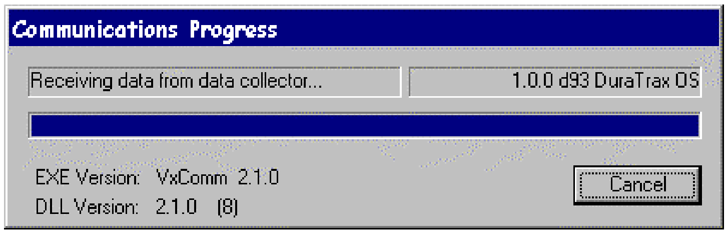
78. To transfer the data file to the computer, point the data collector's IR transmitter towards the computer's IR station.

79. Double-click the **Vxcom** icon. (Please note: If this is the first time you are using **Vxcom**, you must first set its parameters. See Steps 26–31 on pages 21–23 for instructions.)

Figure 3-86 Vxcom Icon

80. The download program looks for the data collector. Press any key on the data collector to start the transfer.

81. A window opens to report the progress of the transfer (Figure 3-87).


Communications Progress

Receiving data from data collector...

1.0.0 d93 DuraTrax OS

EXE Version: VxComm 2.1.0

DLL Version: 2.1.0 (8)

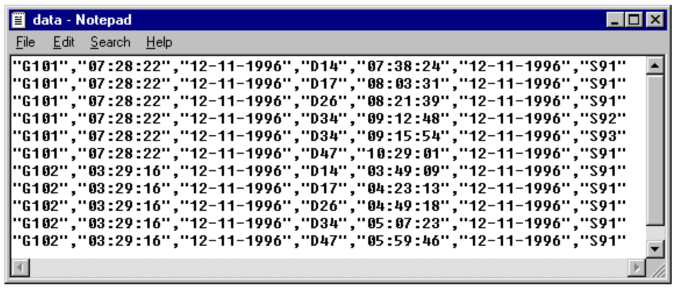
Figure 3-87 Communications Progress Window

Viewing the Data File

82. The file transfers to the **Videx** folder and is saved as a text file named **data.txt** (Figure 3-88). Double-click the **data.txt** icon to open the file.

Figure 3-88 Data.txt Icon

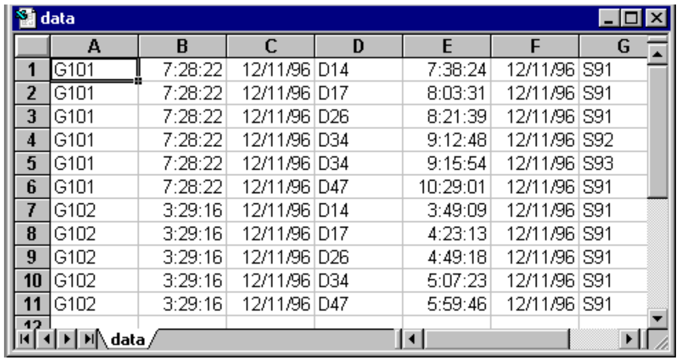
83. The data file opens (Figure 3-89). The data file shows the guard ID entry followed by the time and date; then the location entry followed by the time and date; and finally, the status entry.



```
"G101","07:28:22","12-11-1996","D14","07:38:24","12-11-1996","S91"
"G101","07:28:22","12-11-1996","D17","08:03:31","12-11-1996","S91"
"G101","07:28:22","12-11-1996","D26","08:21:39","12-11-1996","S91"
"G101","07:28:22","12-11-1996","D34","09:12:48","12-11-1996","S92"
"G101","07:28:22","12-11-1996","D34","09:15:54","12-11-1996","S93"
"G101","07:28:22","12-11-1996","D47","10:29:01","12-11-1996","S91"
"G102","03:29:16","12-11-1996","D14","03:49:09","12-11-1996","S91"
"G102","03:29:16","12-11-1996","D17","04:23:13","12-11-1996","S91"
"G102","03:29:16","12-11-1996","D26","04:49:18","12-11-1996","S91"
"G102","03:29:16","12-11-1996","D34","05:07:23","12-11-1996","S91"
"G102","03:29:16","12-11-1996","D47","05:59:46","12-11-1996","S91"
```

Figure 3-89 Data File Displayed

84. You can also use an application program (such as a word processing, spreadsheet, or database program) to open the data file. Figure 3-90 shows an example of the file displayed in Excel. The file will differ in appearance depending on the date, time, and bar codes scanned.



data

	A	B	C	D	E	F	G
1	G101	7:28:22	12/11/96	D14	7:38:24	12/11/96	S91
2	G101	7:28:22	12/11/96	D17	8:03:31	12/11/96	S91
3	G101	7:28:22	12/11/96	D26	8:21:39	12/11/96	S91
4	G101	7:28:22	12/11/96	D34	9:12:48	12/11/96	S92
5	G101	7:28:22	12/11/96	D34	9:15:54	12/11/96	S93
6	G101	7:28:22	12/11/96	D47	10:29:01	12/11/96	S91
7	G102	3:29:16	12/11/96	D14	3:49:09	12/11/96	S91
8	G102	3:29:16	12/11/96	D17	4:23:13	12/11/96	S91
9	G102	3:29:16	12/11/96	D26	4:49:18	12/11/96	S91
10	G102	3:29:16	12/11/96	D34	5:07:23	12/11/96	S91
11	G102	3:29:16	12/11/96	D47	5:59:46	12/11/96	S91

Figure 3-90 Data File Displayed in Excel

You can create an application so that it displays a CRF file description field in the data file, instead of the user input. Steps 85 and 86 will show you how to use CRF files for the "Status" and "Location" Handlers. Step 85 shows you how to change the "Status" Handler to use CRF files. Step 86 shows you how to change the "Location" Handler to use CRF files. Steps 85 and 86 are optional.

85. To use a CRF file for the "Status" Handler:
a) Click to select the "Status" Handler in the design window.

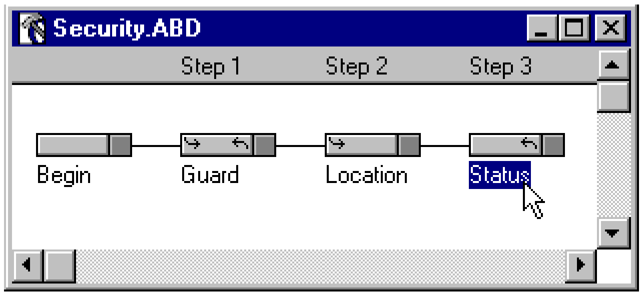


Figure 3-91 Select "Status" Handler

b) Select **Matching** in the **Properties** window.

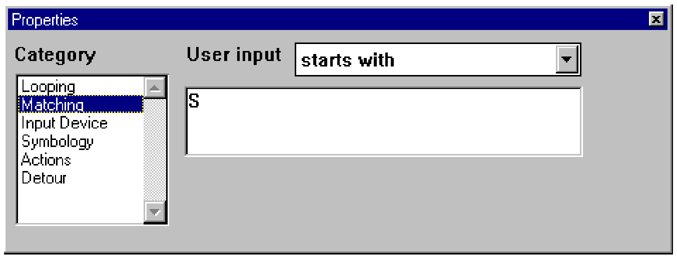


Figure 3-92 Select Matching from Properties Window

c) Choose **is in CRF file** from the pop-up menu.

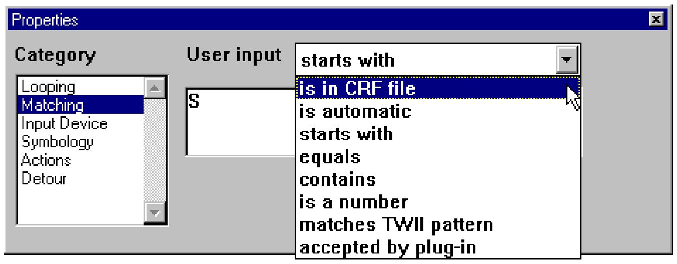


Figure 3-93 Choose **is in CRF File** from Menu

d) Click **Browse**.

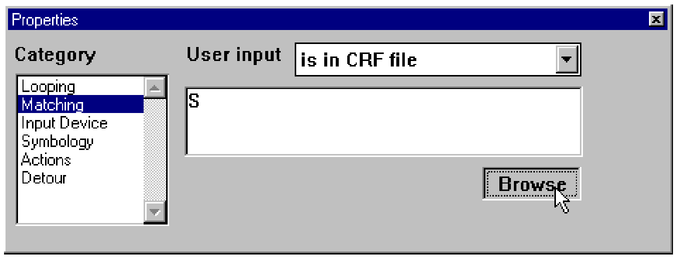


Figure 3-94 Click Browse

e) Select the **Scstat.crf** file (located in the **Videx** folder's **Samples** folder) and click **Open** (Figure 3-95).

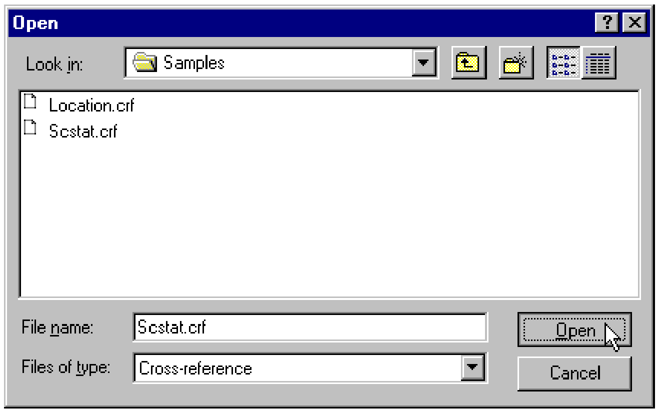
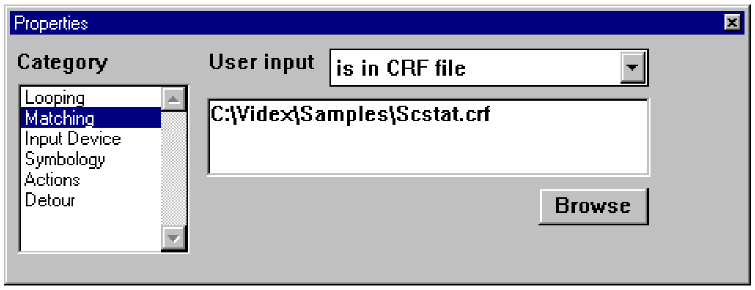


Figure 3-95 Open Scstat.crf File

f) The pathway to the CRF file is displayed in the edit window (Figure 3-96).



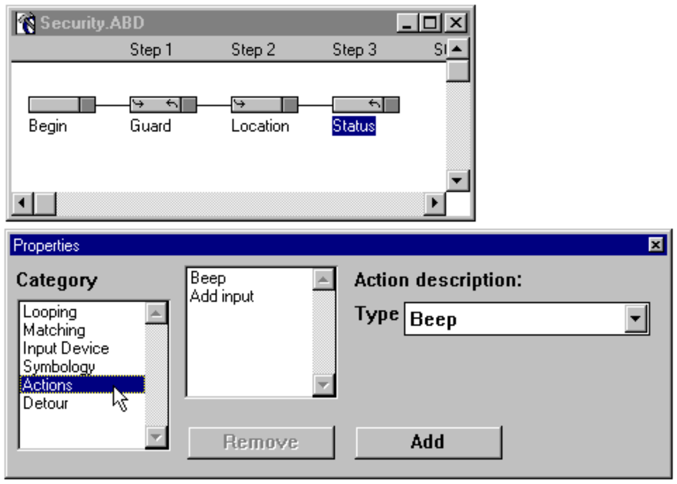
User input: is in CRF file

C:\Videx\Samples\Scstat.crf

Figure 3-96 “Status” Handler - Matching Criterion in Scstat.crf File

To display the CRF file description in the data file, we also need to change the **Actions** property to show the CRF file description.

g) Select **Actions** in the **Properties** window (Figure 3-97).



Type: Beep

Figure 3-97 “Status” Handler - Actions Property

h) Choose **Add CRF field to record** from the **Action description** pop-up menu (Figure 3-98).

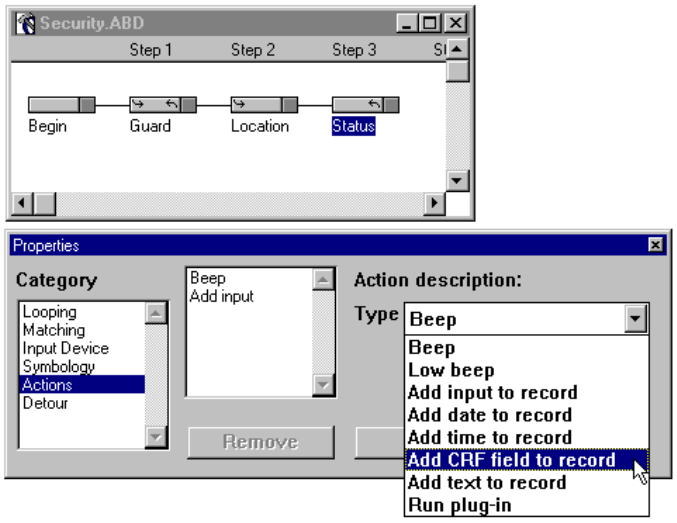
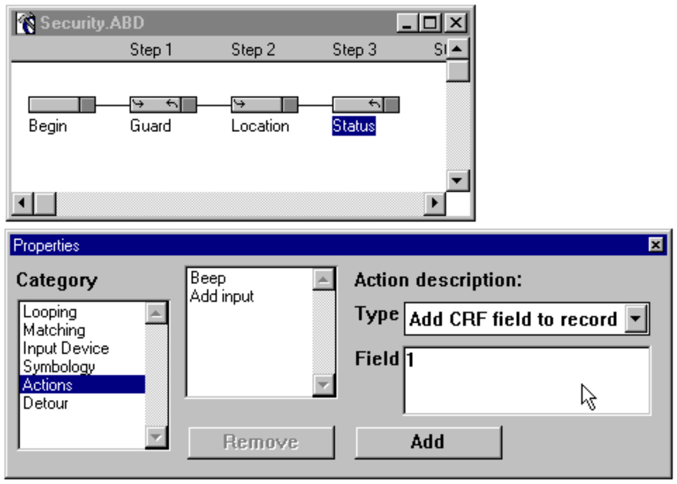


Figure 3-98 “Status” Handler - Actions Property

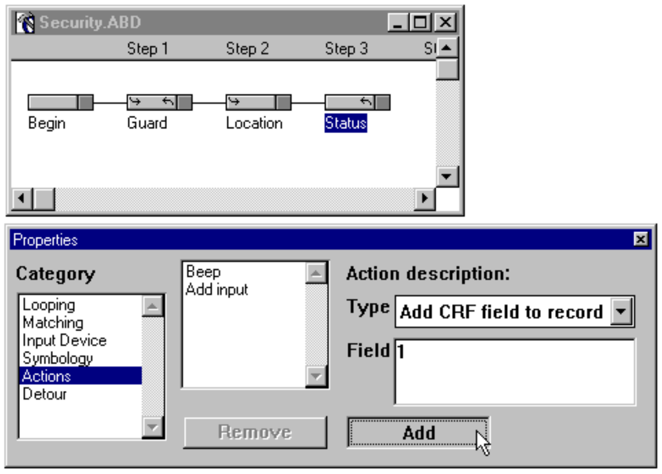
i) The **Properties** window then displays a Field edit box to enter which CRF description field to use. Click in the **Field** edit box and type a **1** (Figure 3-99).



Action description:
Type Add CRF field to record
Field 1

Figure 3-99 Actions Property - Add CRF Field to Record

j) Click **Add** (Figure 3-100) to add the new description to the action list.



Action description:

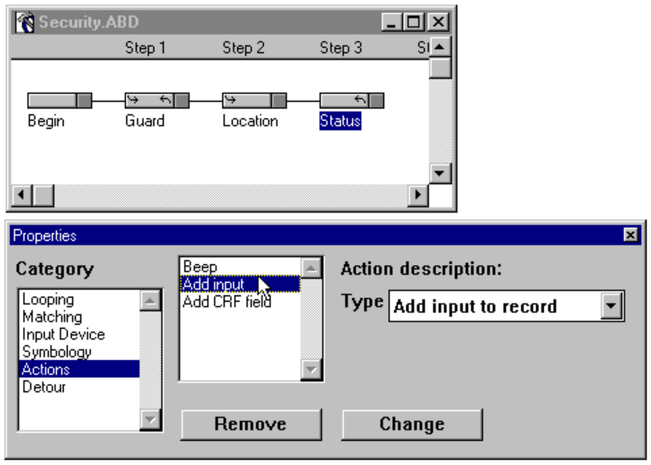
Type: Add CRF field to record

Field: 1

Figure 3-100 Actions Property - Adding CRF Field Description

To have the data file show only the "Status" CRF file description and not the actual user input, you need to remove the **Add input** action from the action list.

k) Select **Add input** in the action list (Figure 3-101).

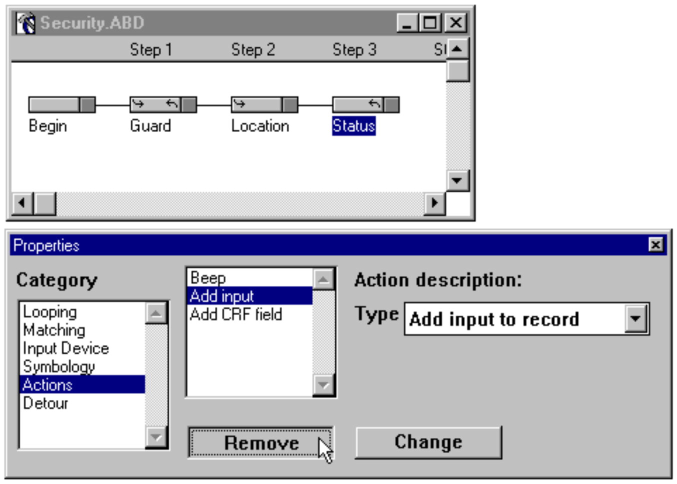


Action description:

Type: Add input to record

Figure 3-101 Actions Property - Select Add Input

l) Click **Remove** (Figure 3-102) and the **Add input** action is removed from the action list (Figure 3-103).



Action description:

Type: Add input to record

Figure 3-102 Actions Property - Click Remove

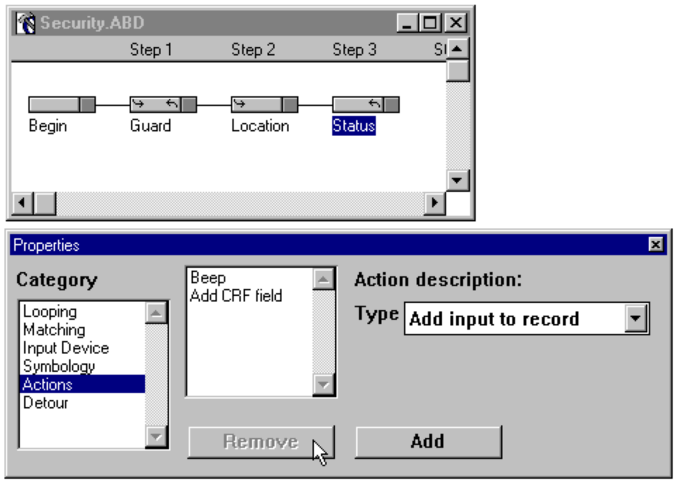
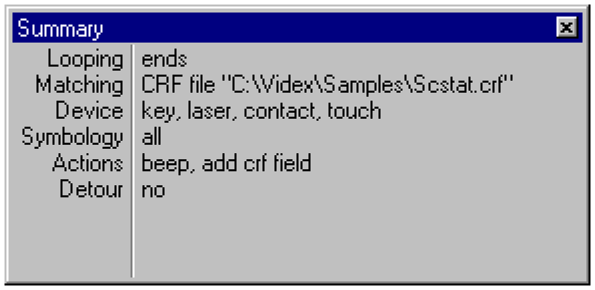


Figure 3-103 Actions Property - Add Input Action Removed

m) Choose **Save** from the **File** menu or type **Ctrl+S** to save the changes.

n) The “Status” Handler **Summary** window shows the changes to the **Matching** and **Actions** properties (Figure 3-104).



Summary

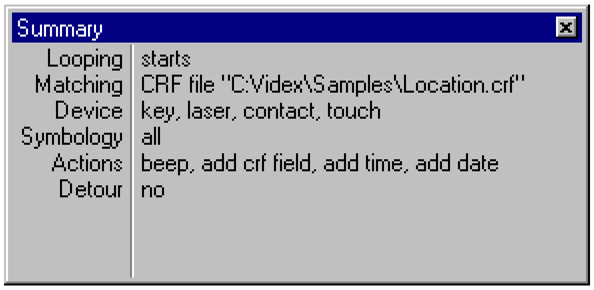
Looping	ends
Matching	CRF file "C:\Videx\Samples\Scstat.crf"
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add crf field
Detour	no

Figure 3-104 “Status” Handler Summary Window

86. If you built the Location.crf file in Chapter 1, you may use it for the “Location” Handler.
a) Select the “Location” Handler.
b) Change its **Matching** property to **is in CRF file**.
c) Open the **Location.crf** file.
d) Change the **Actions** property’s action list to include **Add CRF field to record** (type a 1 in the CRF Field edit box).

e) Remove **Add input** from the **Actions** property's action list.
f) Click and drag the **Add CRF field** action in the action list up to the **Add time** action (this will display the CRF field before the time and date in the data file).
g) Choose **Save** from the **File** menu or type **Ctrl+S** to save the change.

Figure 3-105 shows the "Location" Handler **Summary** window with the changes.

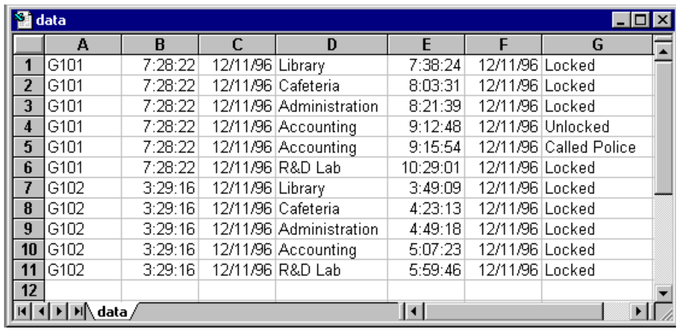


Looping	starts
Matching	CRF file "C:\Videx\Samples\Location.crf"
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add crf field, add time, add date
Detour	no

Figure 3-105 "Location" Handler Summary Window

87. After changing the Security application, transfer it to the data collector and use the Security application bar codes to enter a security round.

    Figure 3-106 shows an example of the data file when CRF files are used for the “Location” and “Status” Handlers.



	A	B	C	D	E	F	G
1	G101	7:28:22	12/11/96	Library	7:38:24	12/11/96	Locked
2	G101	7:28:22	12/11/96	Cafeteria	8:03:31	12/11/96	Locked
3	G101	7:28:22	12/11/96	Administration	8:21:39	12/11/96	Locked
4	G101	7:28:22	12/11/96	Accounting	9:12:48	12/11/96	Unlocked
5	G101	7:28:22	12/11/96	Accounting	9:15:54	12/11/96	Called Police
6	G101	7:28:22	12/11/96	R&D Lab	10:29:01	12/11/96	Locked
7	G102	3:29:16	12/11/96	Library	3:49:09	12/11/96	Locked
8	G102	3:29:16	12/11/96	Cafeteria	4:23:13	12/11/96	Locked
9	G102	3:29:16	12/11/96	Administration	4:49:18	12/11/96	Locked
10	G102	3:29:16	12/11/96	Accounting	5:07:23	12/11/96	Locked
11	G102	3:29:16	12/11/96	R&D Lab	5:59:46	12/11/96	Locked
12							

Figure 3-106 Data File Using CRF Files

This completes the *Building an Application* sections. The next chapter describes the advanced features of Application Builder.

Chapter 4

Advanced Application Building

This chapter provides instructions on:

- Using Detour Handlers
- Using Plug-Ins and Variables
- Using Custom Templates
- Advanced Application Building Strategies

Detour Handlers

A Detour Handler pauses an application, allows another task to be performed, and then resumes the application. When the application returns from a Detour Handler, it returns to the previous Input Handler and resumes where it left off. The application may return from a Detour Handler in one of three ways:

- After performing the Detour Handler's actions.
- After three seconds or a button press (whichever is first).
- After pressing the ESC key on a LaserLite Pro or LaserLite Mx.

Detour Handlers can be used to provide additional information to the user; for example, instructions, error messages, the amount of battery charge remaining, the amount of memory remaining, and so on.

In the following sections we will show examples that use Detour Handlers.

Example 1 will add a single Detour Handler that will be an error message. Example 2 will add Detour Handlers that will provide instructions to the user. For another example of the use of Detour Handlers, refer to the discussion of the "Inspect.abd" sample application in Appendix A.

We will use the Security application we built in the last chapter in the following examples. The Security application has three steps. The first step is the "Guard" Handler and it identifies the guard and the time the security round begins. The second step is the "Location" Handler and it identifies the location and the time it was checked. The third step is the "Status" Handler and it reports the status of the location.

Example 1: Error Message Detour Handler

In the following example, we will attach a single Detour Handler to the existing Input Handlers to inform the user if an error is made while scanning.

The following steps show you how to add a Detour Handler to an application:

1. Open the Security application that you built in Chapter 3 (Figure 4-1).

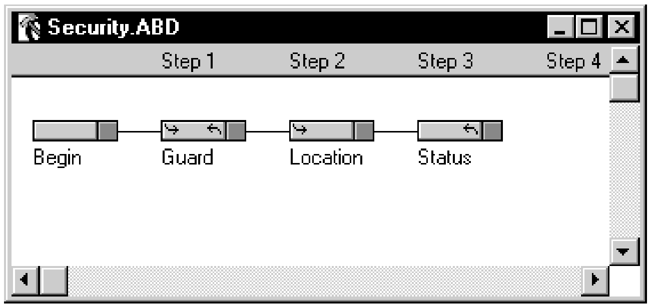


Figure 4-1 Security.abd Application

In this example, we want the error message Detour Handler to be accessible from each current Input Handler. The easiest way to do this is to select all of the Input Handlers before adding the new Input Handler. The new Input Handler will automatically be connected to each selected Input Handler.

2. Select all of the Input Handlers by dragging a rectangle around them (Figure 4-2), by choosing **Select All Handlers** from the **Edit** menu (Figure 4-3), or by Shift-clicking each Input Handler.

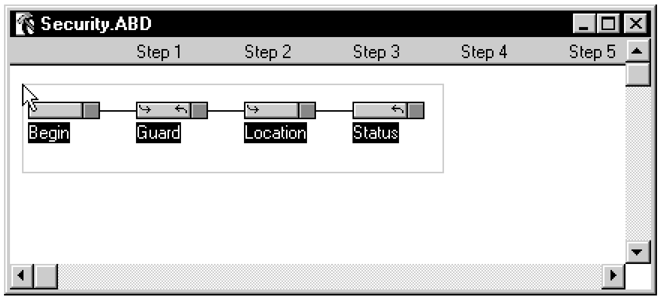


Figure 4-2 Drag Rectangle to Select Input Handlers

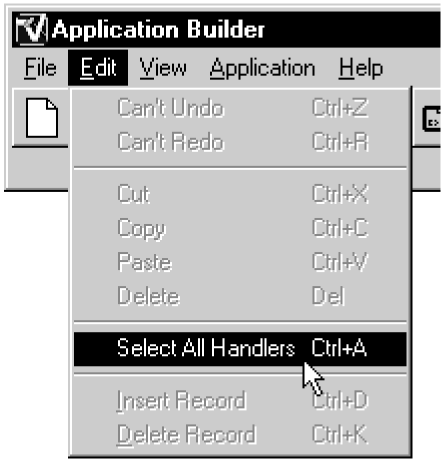


Figure 4-3 Edit Menu–Select All Handlers Command

3. The application window should look like Figure 4-4 when all of the Input Handlers are selected.

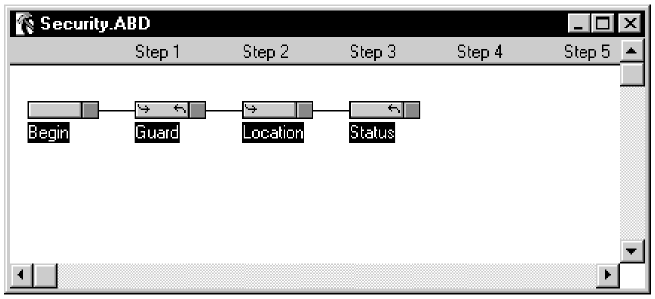


Figure 4-4 All Input Handlers Selected

4. Add a new Input Handler to the application by clicking **New Input Handler,** by choosing **New Input Handler** from the **Application** menu, or by typing **Ctrl**+**E**.

5. A new Input Handler is added to the application (Figure 4-5).

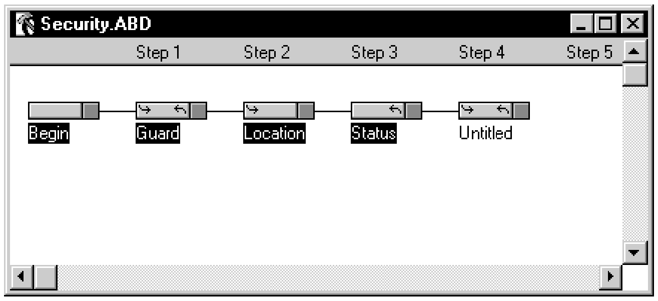


Figure 4-5 New Input Handler Added

6. To verify that the new Input Handler is attached to each of the other Input Handlers, click and drag the new Input Handler down the window (Figure 4-6). You should see connecting lines between the new Input Handler and each previous Input Handler.

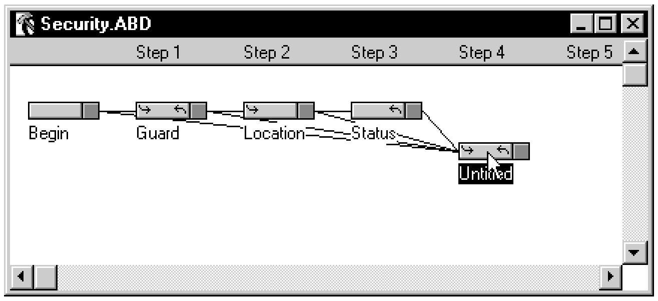


Figure 4-6 Click and Drag New Input Handler

7. Select the new Untitled Input Handler (Figure 4-7).

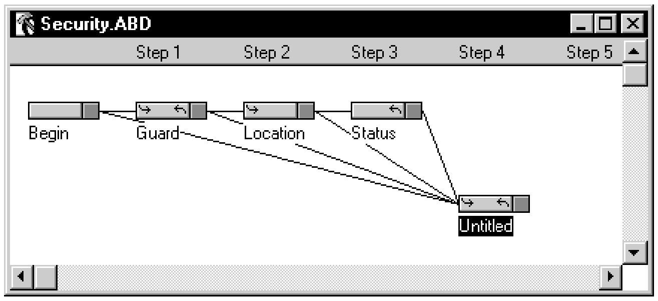
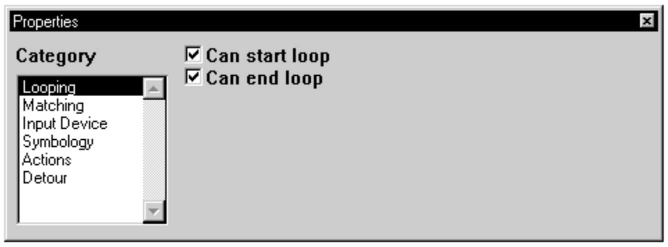


Figure 4-7 New Untitled Input Handler Selected

8. Open the **Properties** window by clicking **Properties Window** or by choosing **Properties Window** from the **View** menu.

9. The **Properties** window opens (Figure 4-8).

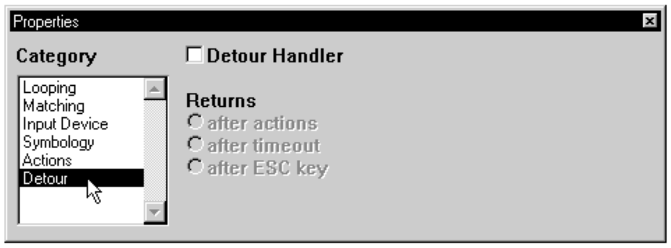

Properties

Category: Looping (selected), Matching, Input Device, Symbology, Actions, Detour

☑ Can start loop
☑ Can end loop

Figure 4-8 Properties Window

10. Click **Detour** from the **Category** list to select the Detour category (Figure 4-9).


Properties

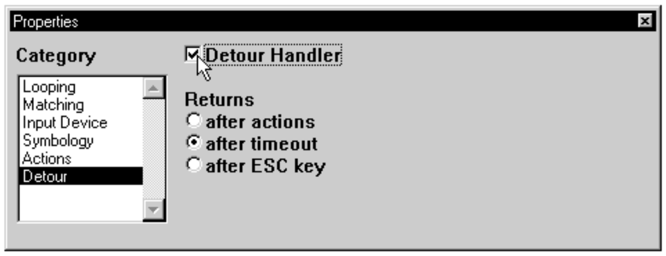
Category: Looping, Matching, Input Device, Symbology, Actions, Detour (selected)

☐ Detour Handler

Returns
○ after actions
○ after timeout
○ after ESC key

Figure 4-9 Properties Window - Detour Category

11. Click the Detour Handler selection box to make the selected Input Handler a Detour Handler (Figure 4-10).


Properties

Category: Looping, Matching, Input Device, Symbology, Actions, Detour (selected)

☑ Detour Handler

Returns
○ after actions
● after timeout
○ after ESC key

Figure 4-10 Properties Window - Detour Handler Selected

You can return to the application from a Detour Handler in one of three ways:
1) **after actions**: After the Detour Handler's actions are performed.
2) **after timeout**: After three seconds or if a button on the data collector is pressed (default).
3) **after ESC key**: After the ESC key is pressed on a LaserLite Pro or LaserLite Mx.

12. We will use the **Returns after timeout** default in this example.

13. The selected Input Handler now appears with stripes to indicate that it is a Detour Handler (Figure 4-11).

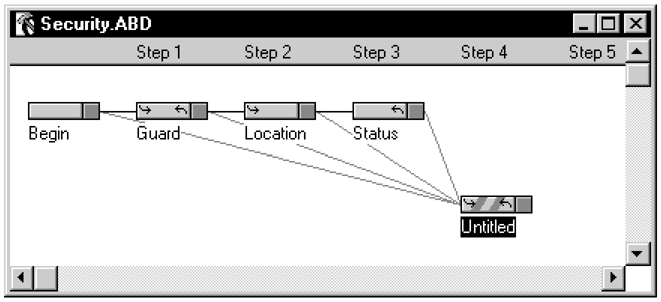


Figure 4-11 Selected Input Handler is Detour Handler

14. Click **Untitled** to edit the name of the Detour Handler (Figure 4-12). Since we are using the Detour Handler to report a scanning error, we will name the handler **Error**.

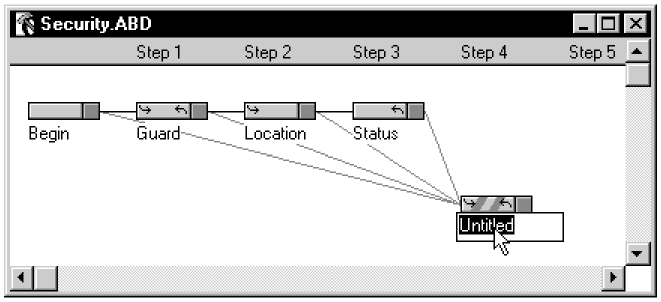


Figure 4-12 Edit Detour Handler Name

15. Type **Error** (Figure 4-13).

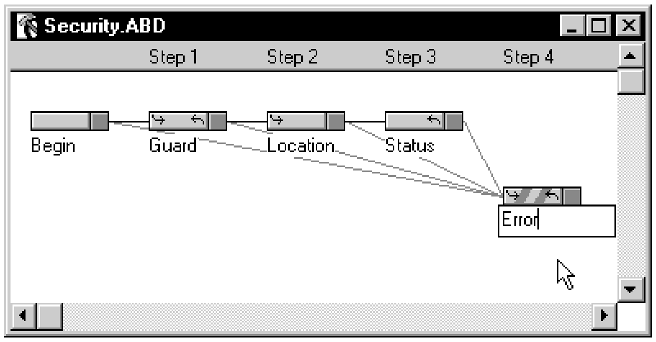


Figure 4-13 Edit Detour Handler Name

16. Press the computer's <Enter> key to enter the new name (Figure 4-14).

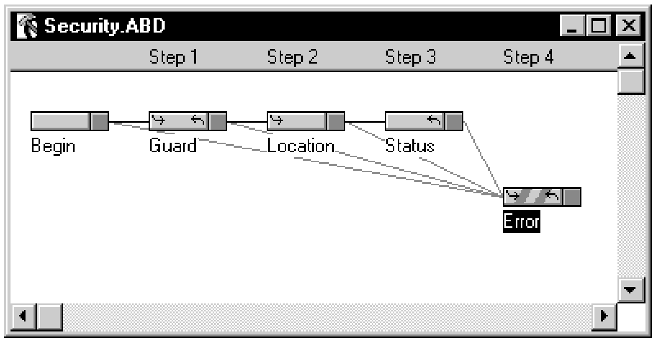


Figure 4-14 Detour Handler Named Error

17. At the **Properties** window click **Action** in the **Category** list (Figure 4-15).

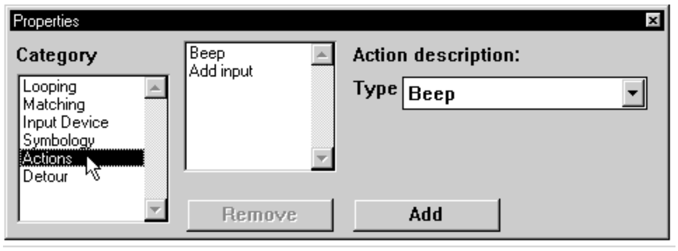


Figure 4-15 Properties Window - Action Category

By default the handler is given the actions of **Beep** and **Add input**. For the “Error” Handler we want to remove these default actions and add the **Low beep** action to signify that an error occurred.

18. Select **Beep** in the action list (Figure 4-16).

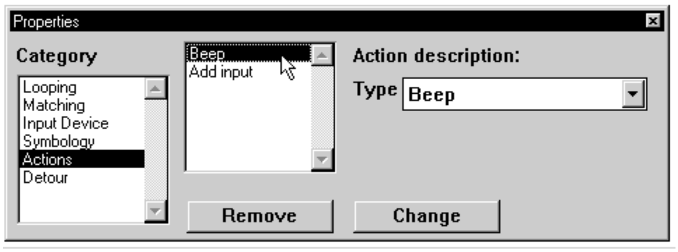


Figure 4-16 Beep Action Selected

19. Click **Remove** to delete the action from the list (Figure 4-17).

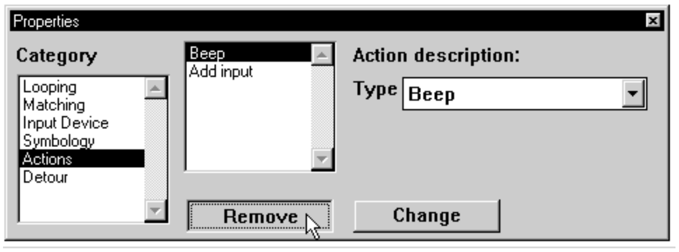


Figure 4-17 Remove Beep Action

20. Select **Add input** in the action list (Figure 4-18).

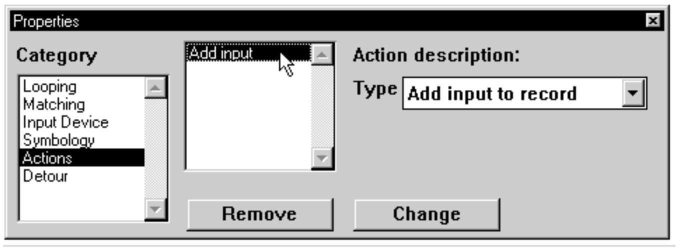


Figure 4-18 Add Input Action Selected

21. Click **Remove** to delete the action from the list (Figure 4-19).

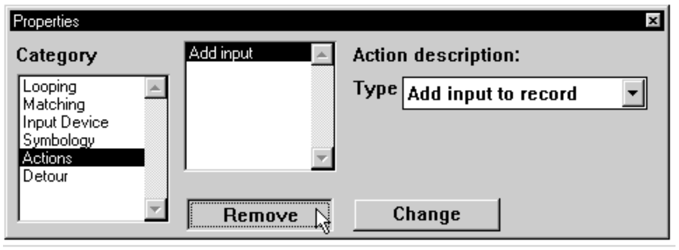


Figure 4-19 Remove Add Input Action

22. Choose **Low beep** from the **Action description** pop-up menu (Figure 4-20).

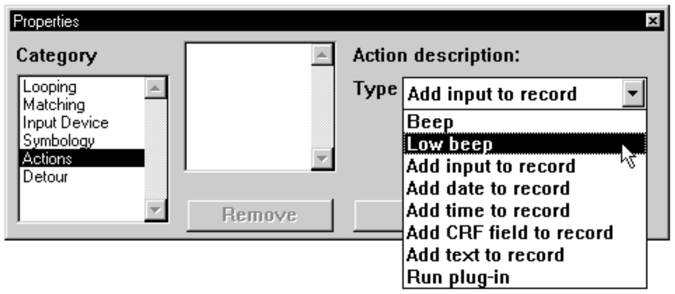
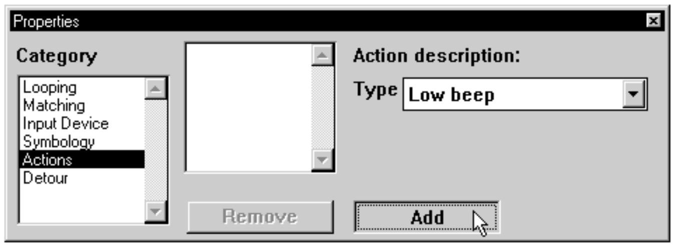


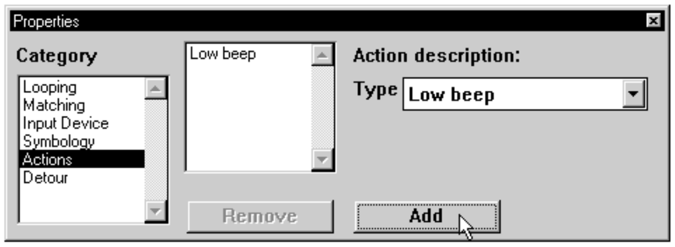
Figure 4-20 Choose Low Beep Action from Menu

23. Click **Add** (Figure 4-21) to add the **Low beep** action to the action list (Figure 4-22).



Action description:
Type Low beep

Figure 4-21 Click Add to Add Action to Action List



Low beep

Action description:
Type Low beep

Figure 4-22 Action Added to Action List

24. Open the **Display** window by clicking **Display Window** or by choosing **Display Window** from the **View** menu.

25. The **Display** window opens (Figure 4-23).

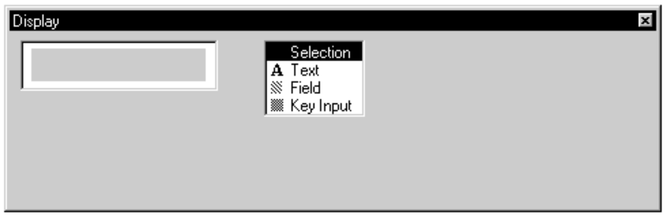


Figure 4-23 Display Window

26. Click **Text** from the tool palette (Figure 4-24).

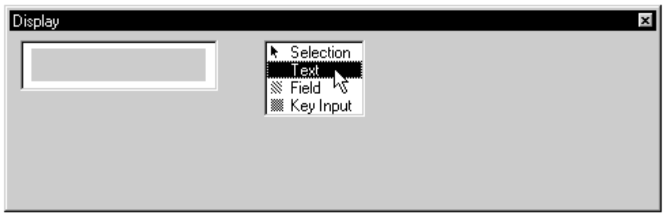


Figure 4-24 Display Window - Select Text Tool

27. The **Display** window shows the text entry screen (Figure 4-25).

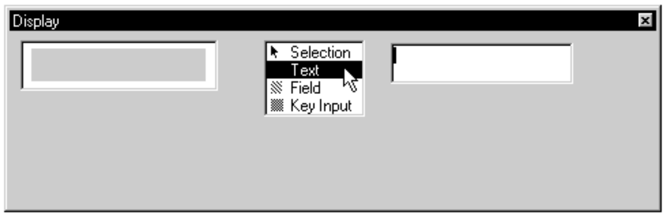
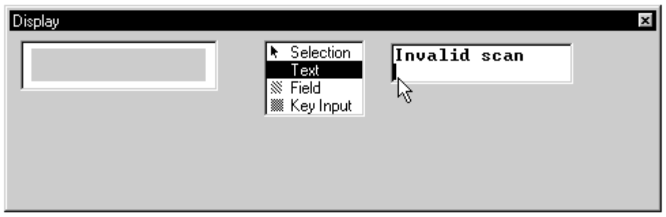


Figure 4-25 Display Window - Text Tool Selected

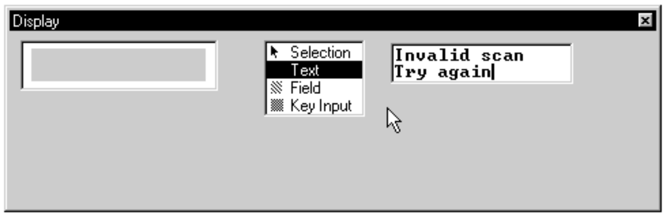
28. Type **Invalid scan** on the top line (Figure 4-26). DO NOT PRESS <ENTER> YET. Use your mouse to click at the beginning of the second line (Figure 4-26) or use the arrow keys on your computer's keyboard to move the cursor to the beginning of the second line. The <Enter> key will not act as a line return in the **Display** window; it is only used to accept the text entry.



```
Invalid scan
```

Figure 4-26 Display Window - First Line of Text Entered

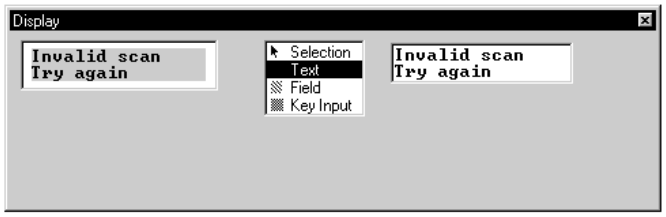
29. Type **Try again** on the second line (Figure 4-27).



```
Invalid scan
Try again
```

Figure 4-27 Display Window - Second Line of Text Entered

30. Press <Enter> to accept the text entry (Figure 4-28). The entered text appears in the display area at the left of the **Display** window.



```
Invalid scan
Try again
```

Figure 4-28 Display Window - Text Entered

31. Save the application by choosing **Save** from the **File** menu or by typing **Ctrl+S**.

32. Transfer the Security application to your data collector.

33. Use the bar codes on page 200 to enter a security round. To see how the Detour Handler operates, scan two Location bar codes or two Status bar codes in a row.

The Detour Handler in this example notifies you that an error has occurred, but does not offer any advice on how to correct the error. The following example will show you how to use Input Handlers to provide instructions to a user.

Example 2: Instruction Message Detour Handler

In the following example, a Detour Handler will be attached to each existing Input Handler. The Detour Handlers will instruct the user as to what action to perform if they do not make a valid scan.

In this example, let us assume: 1) the Guard ID bar code is on the Guard's ID badge; 2) the Location bar codes are on the door of the location; and 3) the Status bar codes are on a scanpad being carried by the guard.

The Detour Handler attached to the "Begin" Handler will be stepped to if a guard bar code is not scanned, so we want this Detour Handler to tell the user to scan the bar code on the Guard ID badge. The Detour Handler attached to the "Guard" Handler will be stepped to if the next bar code scanned is not a location bar code, so we want this Detour Handler to tell the user to scan the bar code on the door. The Detour Handler attached to the "Location" Handler will be stepped to if the next bar code scanned is not a status bar code, so we want this Detour Handler to tell the user to scan one of the status bar codes on the scanpad. The Detour Handler attached to the "Status" Handler will be stepped to if the next bar code scanned is not a guard or location bar code, so we want this Detour Handler to tell the user to scan the door or the badge.

1. Open the Security application that you built in Chapter 3 (Figure 4-29). Note: If you added the “Error” Handler in the previous example, delete the “Error” Handler by selecting it and either pressing **Delete** from the computer’s keyboard or choose **Delete Handler** from the **Edit** menu. The application window should appear similar to Figure 4-29.

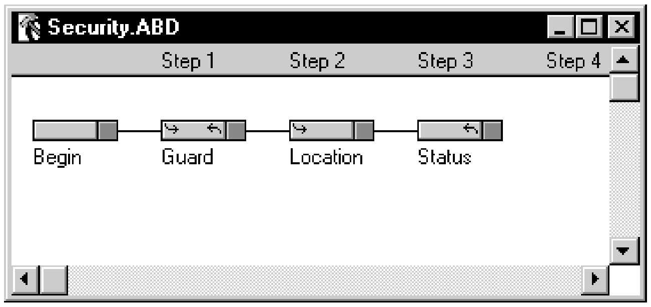


Figure 4-29 Security.abd Application

2. Select the “Begin” Handler (Figure 4-30).

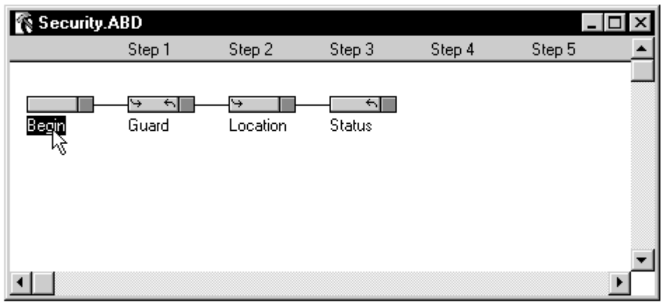


Figure 4-30 “Begin” Handler Selected

3. Insert a new Input Handler by clicking **New Input Handler** or choosing **New Input Handler** from the **Application** menu.

4. A new untitled Input Handler is added to the “Begin” Handler (Figure 4-31).

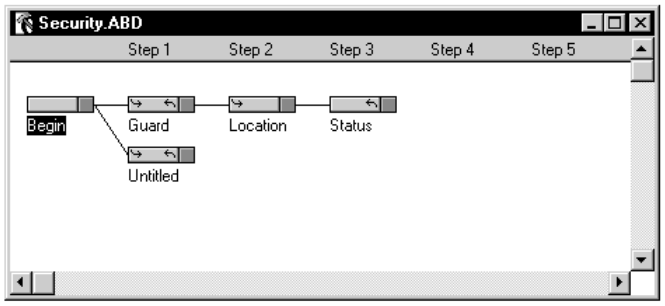


Figure 4-31 New Input Handler Added

5. Click the “Guard” Handler (Figure 4-32).

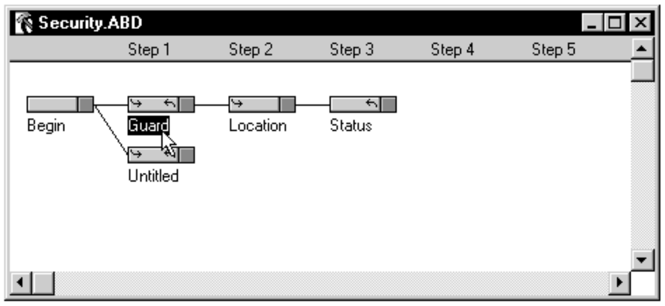


Figure 4-32 “Guard” Handler Selected

6. Click **New Input Handler** or choose **New Input Handler** from the **Application** menu.

7. A new untitled Input Handler is added to the “Guard” Handler (Figure 4-33).

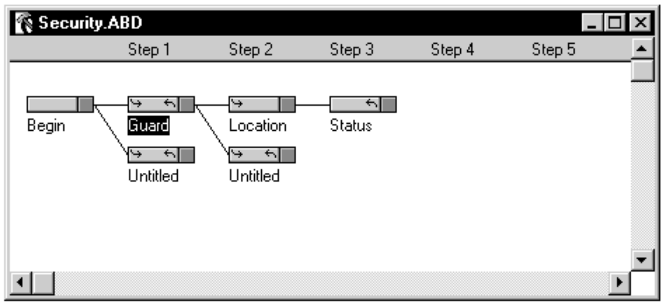


Figure 4-33 New Input Handler Added

8. Click the “Location” Handler (Figure 4-34).

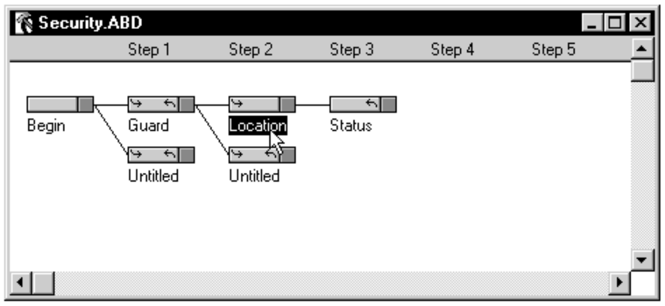


Figure 4-34 “Location” Handler Selected

9. Click **New Input Handler** or choose **New Input Handler** from the **Application** menu.

10. A new untitled Input Handler is added to the “Location” Handler (Figure 4-35).

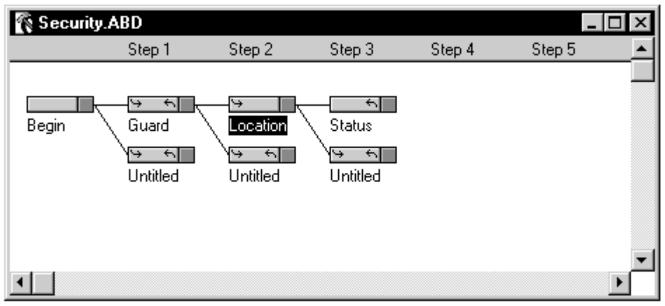


Figure 4-35 New Input Handler Added

11. Click the “Status” Handler (Figure 4-36).

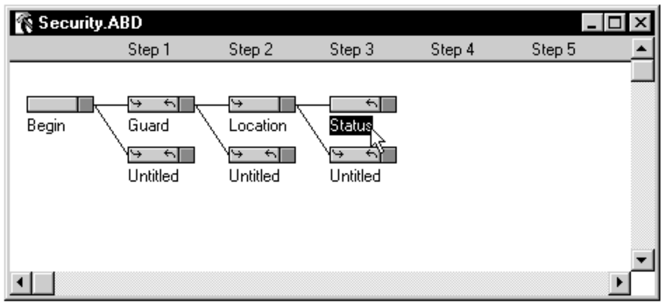


Figure 4-36 “Status” Handler Selected

12. Click **New Input Handler** or choose **New Input Handler** from the **Application** menu. A new Input Handler is added to the “Status” Handler (Figure 4-37).

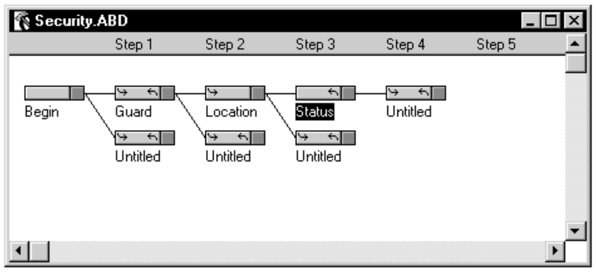


Figure 4-37 New Input Handler Added

13. Select the new Input Handlers. In the following diagrams, we select the new Input Handlers by dragging a rectangle around the three at the bottom of the window (Figure 4-38) and Shift-clicking the one attached to the "Status" Handler (Figure 4-39).

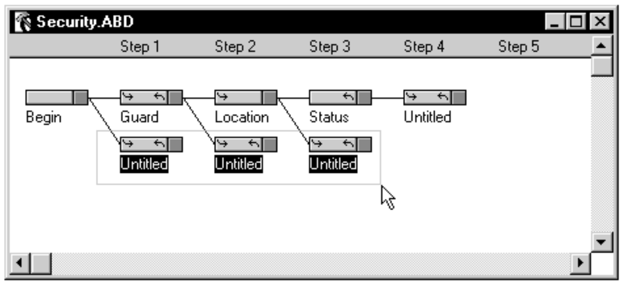


Figure 4-38 Drag Rectangle to Select Input Handlers

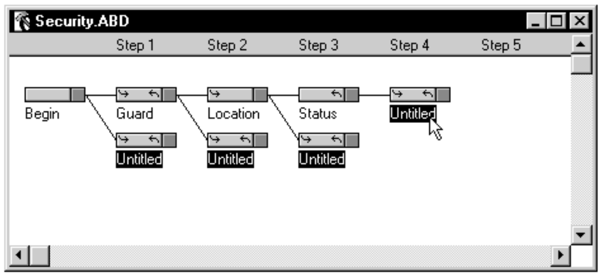


Figure 4-39 Shift-click to Select Input Handlers

14. Open the **Properties** window by clicking **Properties Window** (Figure 4-40) or by choosing **Properties Window** from the **View** menu (Figure 4-41).

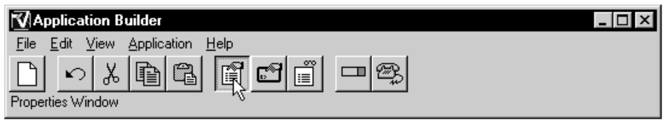


Figure 4-40 Click Properties Window

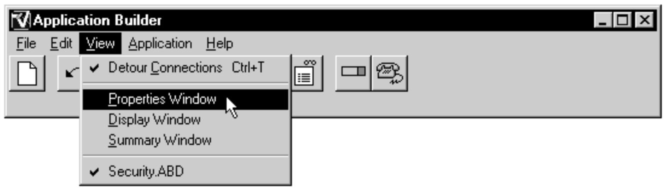
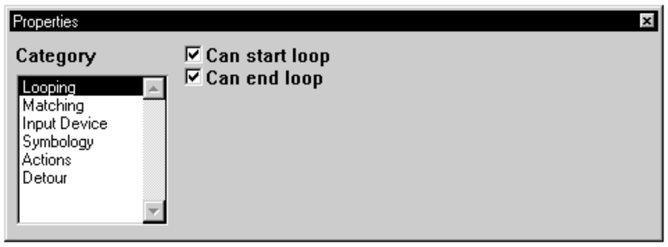


Figure 4-41 View Menu - Properties Window Command

15. The **Properties** window opens (Figure 4-42).



☑ Can start loop
☑ Can end loop

Figure 4-42 Properties Window

16. Click **Detour** from the **Category** list to select the Detour category (Figure 4-43).

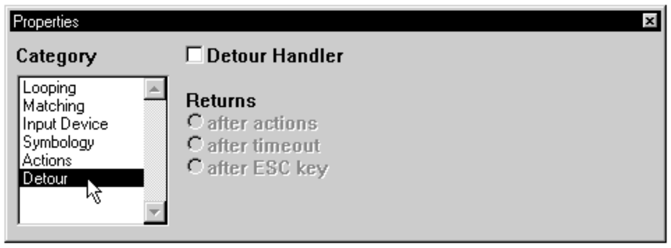


Figure 4-43 Properties Window - Detour Category

17. Click the Detour Handler selection box to make the selected Input Handlers into Detour Handlers (Figure 4-44).

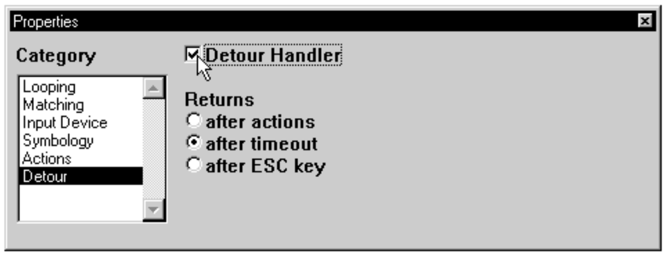


Figure 4-44 Properties Window - Detour Handler Selected

You can return to the application from a Detour Handler in one of three ways:
1) **after actions**: After the Detour Handler's actions are performed.
2) **after timeout**: After three seconds or if a key on the data collector is pressed (default).
3) **after ESC key**: After the ESC key is pressed on a LaserLite Pro or a LaserLite Mx .

18. We will use the **Returns after timeout** default in this example.

19. The selected Input Handlers now appear with stripes to indicate that they are Detour Handlers (Figure 4-45).

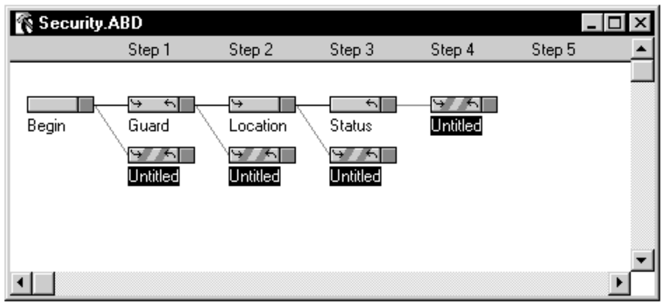


Figure 4-45 Selected Input Handlers Now Detour Handlers

20. With all four Detour Handlers still selected, click **Action** in the **Properties** window's **Category** list (Figure 4-46).

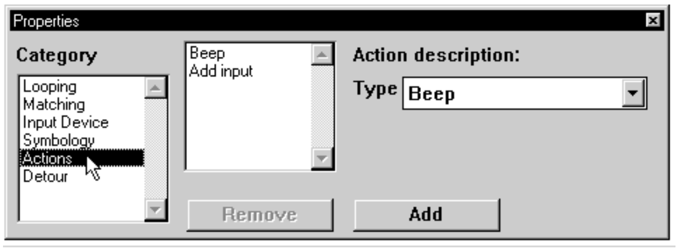


Figure 4-46 Properties Window - Action Category

By default the handlers are given the actions of **Beep** and **Add input**. For these Detour Handlers we want to remove the default actions and add the **Low beep** action to signify an invalid scan.

21. Select **Beep** in the action list (Figure 4-47).

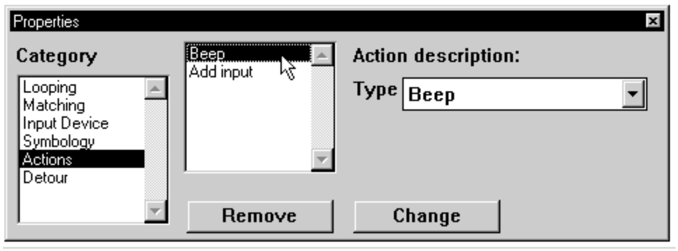


Figure 4-47 Beep Action Selected

22. Click **Remove** to delete the action from the list (Figure 4-48).

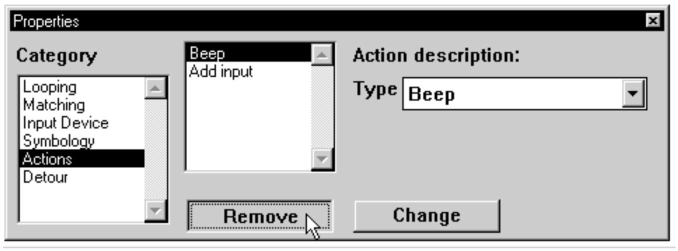


Figure 4-48 Remove Beep Action

23. Select **Add input** in the action list (Figure 4-49).

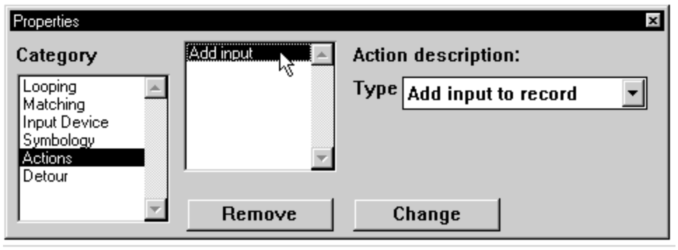


Figure 4-49 Add Input Action Selected

24. Click **Remove** to delete the action from the list (Figure 4-50).

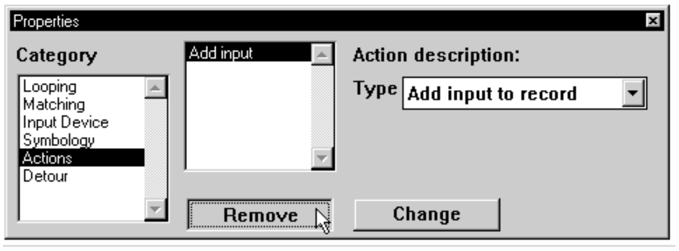


Figure 4-50 Remove Add Input Action

25. Choose **Low beep** from the **Action description** pop-up menu (Figure 4-51).

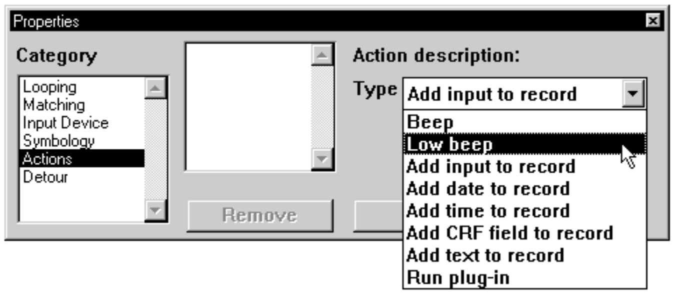


Figure 4-51 Choose Low Beep Action from Menu

26. Click **Add** (Figure 4-52) to add the **Low beep** action to the action list (Figure 4-53).

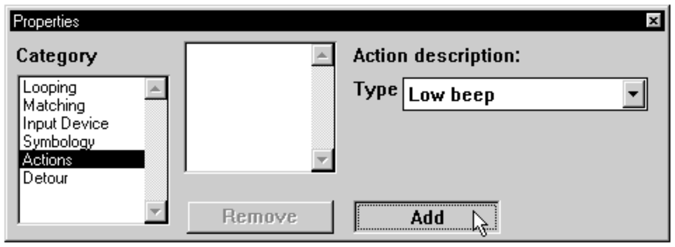


Figure 4-52 Click Add to Add Action to Action List

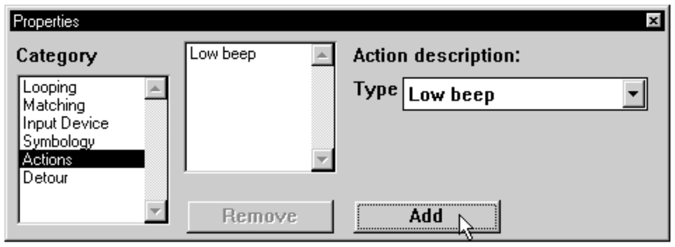


Figure 4-53 Action Added to Action List

27. Click away from the Detour Handlers to deselect them.

28. Select the Detour Handler connected to the "Begin" Handler (Figure 4-54).

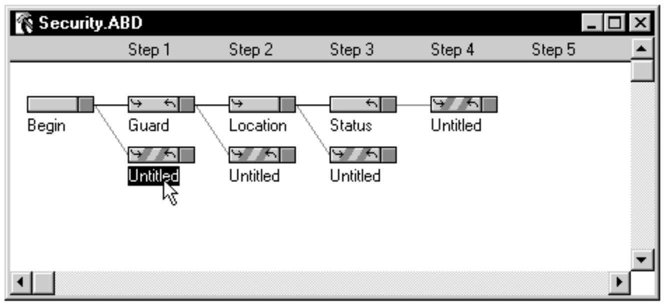


Figure 4-54 Select Detour Handler

29. Click the Detour Handler name to edit the name (Figure 4-55).

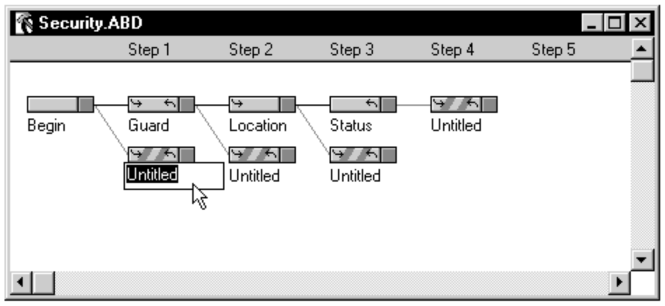


Figure 4-55 Edit Detour Handler Name

30. Type **Instruct1** (Figure 4-56). We will now refer to this Detour Handler as the “Instruct1” Handler.

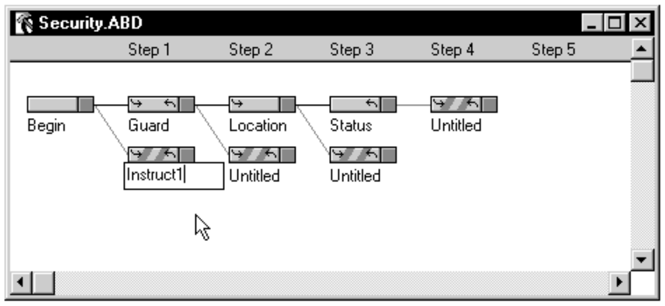


Figure 4-56 Detour Handler Named

31. Follow the same steps to name the other Detour Handlers. Name the one attached to the "Guard" Handler, **Instruct2**; the one attached to the "Location" Handler, **Instruct3**; and the one attached to the "Status" Handler, **Instruct4** (Figure 4-57).

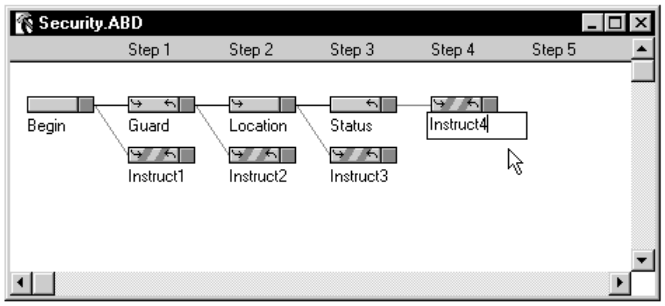


Figure 4-57 Detour Handlers Named

32. Your application window should appear similar to Figure 4-58.

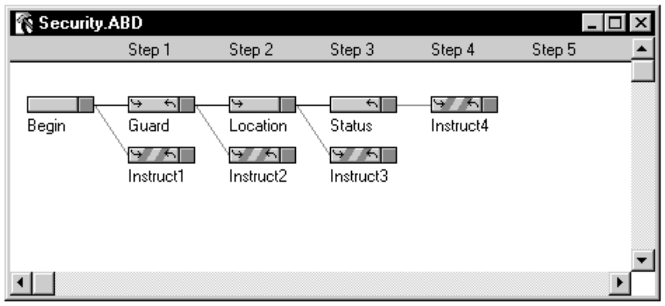


Figure 4-58 Security.abd with Detour Handlers Named

33. Next, we will add the display messages for each Detour Handler. Click the “Instruct1” Handler (Figure 4-59). The “Instruct1” Handler is stepped to if the user does not scan a valid guard bar code. In this example, we are assuming the guard bar code is on the badge. So, we will use **Scan bar code on badge.** as our instruction message.

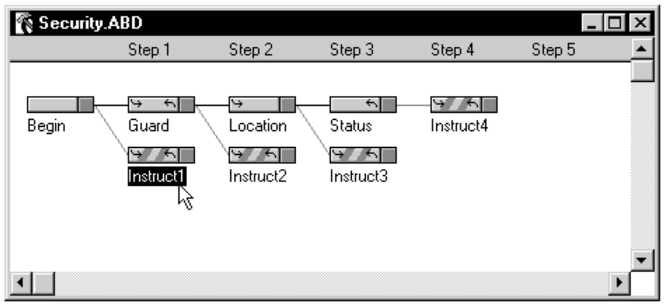


Figure 4-59 “Instruct1” Handler Selected

34. Open the **Display** window by clicking **Display Window** (Figure 4-60) or by choosing **Display Window** from the **View** menu (Figure 4-61).

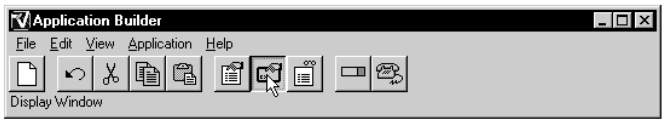


Figure 4-60 Click Display Window

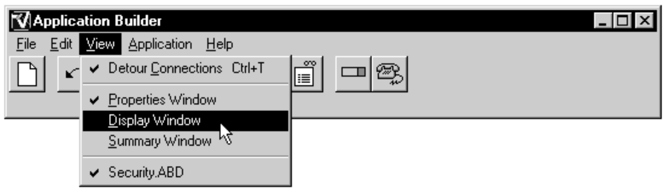


Figure 4-61 View Menu - Display Window Command

35. The **Display** window opens (Figure 4-62).

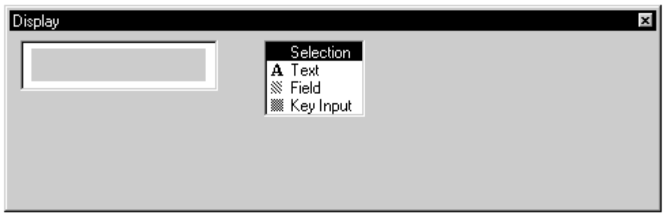


Figure 4-62 Display Window

36. Click **Text** from the tool palette (Figure 4-63).

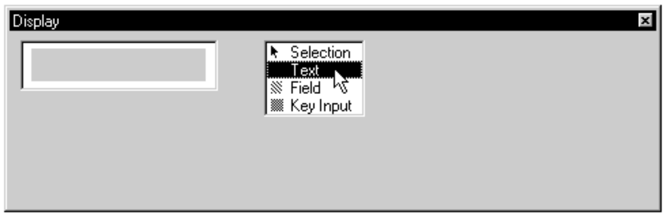


Figure 4-63 Display Window - Select Text Tool

37. The **Display** window shows the text entry screen (Figure 4-64).

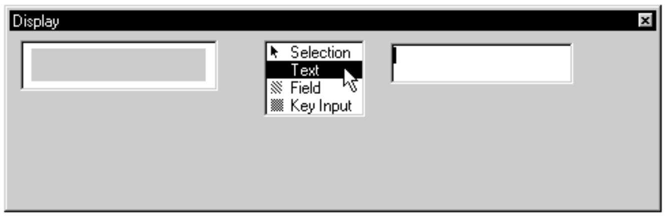
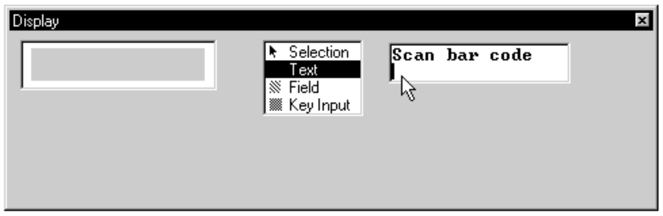


Figure 4-64 Display Window - Text Tool Selected

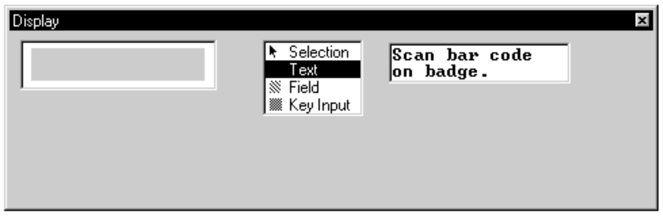
38. Type **Scan bar code** on the top line. DO NOT PRESS <ENTER> YET. Use your mouse to click at the beginning of the second line (Figure 4-65) or use the arrow keys on your computer's keyboard to move the cursor to the beginning of the second line. Note: The <Enter> key does not produce a line return in the **Display** window; it is only used to accept the text entry.



Scan bar code

Figure 4-65 Display Window - First Line of Text Entered

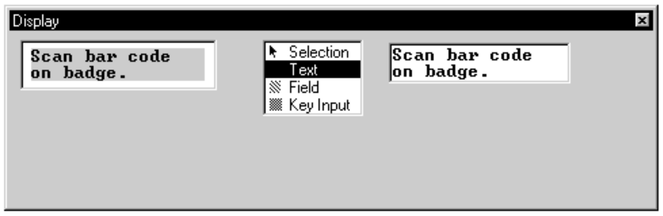
39. Type **on badge.** on the second line (Figure 4-66).



Scan bar code
on badge.

Figure 4-66 Display Window - Second Line of Text Entered

40. Press <Enter> to accept the text entry. The entered text appears in the display area at the left of the **Display** window (Figure 4-67).



Scan bar code
on badge.

Scan bar code
on badge.

Figure 4-67 "Instruct1" Handler Display Message Entered

41. Select the “Instruct2” Handler (Figure 4-68). The “Instruct2” Handler will be stepped to if the user does not scan a valid location bar code. In this example we are assuming the location bar codes are on the door of the location, so we will use **Scan bar code on door.** as our instruction message.

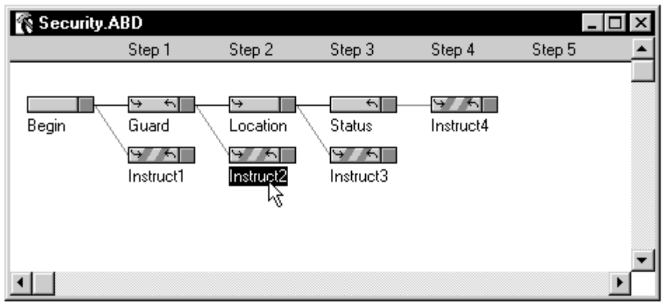
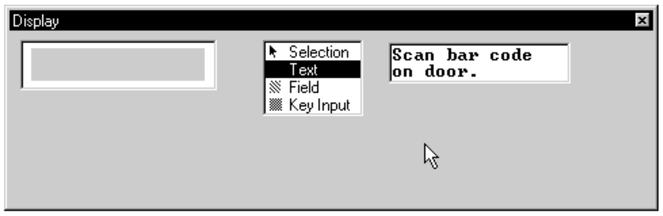


Figure 4-68 “Instruct2” Handler Selected

42. At the **Display** window, type **Scan bar code** on the top line, and type **on door.** on the second line (Figure 4-69).



```
Scan bar code
on door.
```

Figure 4-69 “Instruct2” Handler Message

43. Press <Enter> to accept the text entry. The entered text appears in the display area at the left of the **Display** window (Figure 4-70).

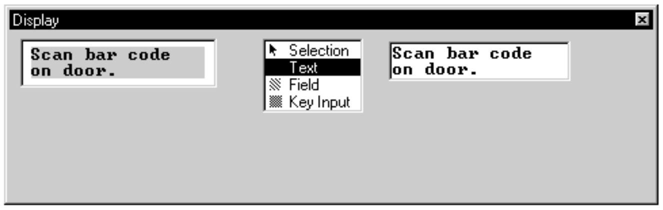


Figure 4-70 “Instruct2” Handler Message Entered

44. Select the “Instruct3” Handler (Figure 4-71). The “Instruct3” Handler will be stepped to if the user does not scan a valid status bar code. In this example we are assuming the status bar codes are on the scanpad being carried by the guard, so we will use **Scan Status bar code on scanpad.** as our instruction message.

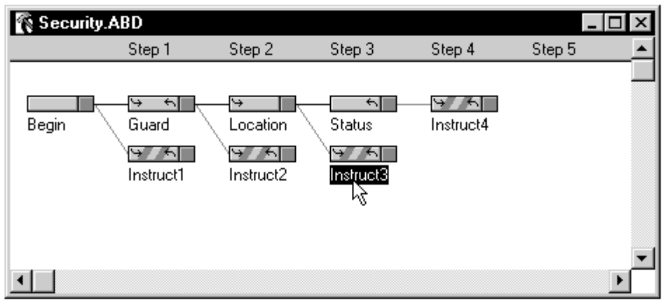
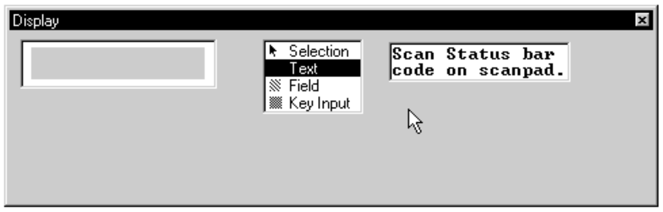


Figure 4-71 “Instruct3” Handler Selected

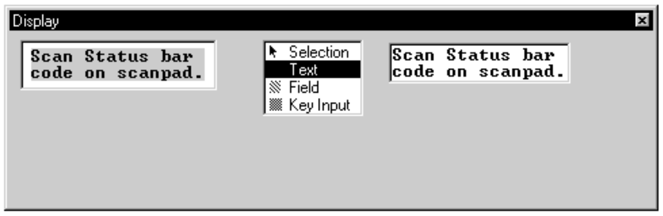
45. At the **Display** window, type **Scan Status bar** on the top line, and type **code on scanpad.** on the second line (Figure 4-72).



```
Scan Status bar
code on scanpad.
```

Figure 4-72 “Instruct3” Handler Message

46. Press <Enter> to accept the text entry. The entered text appears in the display area at the left of the **Display** window (Figure 4-73).



```
Scan Status bar
code on scanpad.
```

```
Scan Status bar
code on scanpad.
```

Figure 4-73 Instruct3 Message Entered

47. Select the “Instruct4” Handler (Figure 4-74). The “Instruct4” Handler will be stepped to if the user does not scan a valid location or guard bar code after scanning a status bar code. We will use **Scan bar code on door or badge.** as our instruction message.

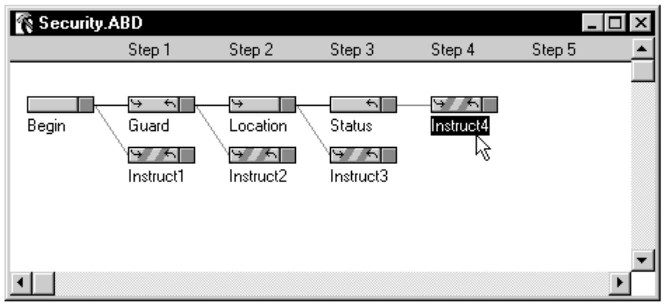
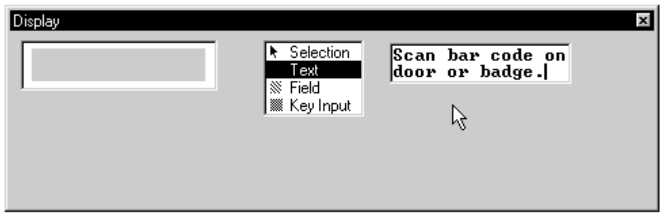


Figure 4-74 “Instruct4” Handler Selected

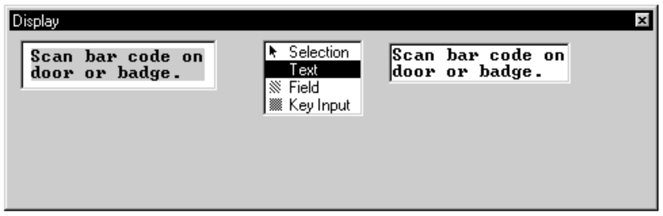
48. At the **Display** window, type **Scan bar code on** on the top line, and type **door or badge.** on the second line (Figure 4-75).



```
Scan bar code on
door or badge.
```

Figure 4-75 “Instruct4” Handler Message

49. Press <Enter> to accept the text entry. The entered text appears in the display area at the left of the **Display** window (Figure 4-76).



```
Scan bar code on
door or badge.
```

```
Scan bar code on
door or badge.
```

Figure 4-76 Instruct4 Message Entered

50. Save the application by choosing **Save** from the **File** menu or by typing **Ctrl+S**.

51. Transfer the Security application to your data collector.

52. Use the bar codes on page 200 to enter a security round. To see how the Detour Handlers operate, scan the bar codes out of order; the Detour Handler messages will instruct you on what should be scanned.

Plug-Ins

Application Builder uses a template (Describe.src) to generate source code in BASIC. This process is discussed in detail in the *Developer's Reference Manual*. A plug-in is BASIC source code that is inserted into the application by Application Builder. Plug-ins provide additional features when creating applications, but they require familiarity with BASIC programming. To use plug-ins effectively, one must also be familiar with the source code template, Describe.src, which is used by Application Builder.

The table on page 193 lists some common and useful plug-ins. ($ represents a string; % represents a number; *italics* represent a variable.) To add a plug-in to an application, open the application in Application Builder, and click the appropriate handler. Open the **Properties** window by clicking **Properties Window** or by choosing **Properties Window** from the **View** menu. Click **Actions** in the **Properties** window's **Category** list. Choose **Run plug-in** from the **Action description** pop-up menu; a text area will appear in the **Properties** window. Click within the text area and enter the text of the desired plug-in. Click **Add** and the plug-in will be added to the list of actions performed by that handler.

Application Builder supports up to 16 plug-ins per handler; each plug-in can be up to 63 characters long. It is possible to incorporate more than one line of Videx BASIC code in a single plug-in by pressing <Ctrl><Enter> at the end of each line of code. If more than one plug-in action is run on an input handler, those lines of Videx BASIC code are inserted consecutively into the source code for the application.

One of the most common reasons to use plug-ins is for applications with exceptional requirements for the format of the data file. For example, plug-ins can be used to modify the format of the time and date, to write fixed-width fields to the data file, or to strip the family code from iButton reads. In these applications, the general form of the plug-ins would be:

> var$ = *expression*
> print #0, *delimiter$*; var$;

where var$ is the information to be written to the data file and *delimiter*$ is the desired delimiter between fields in each record of the data file. For example, if the data file is delimited by commas, *delimiter$* = ",". If the data file is delimited by tab characters, *delimiter$* = chr$(9). If quotes are required around each field, they can be written to the data file as chr$(34). Refer to the Videx BASIC manual for more information on the print statement and the chr$( ) function.

Application Builder version 2.1.0 and higher implements a feature which allows text files to be included within the BASIC source code of an application. This feature is useful in situations in which several plug-ins are required. The syntax for including the Videx BASIC code contained in *myfile.txt* is:

INCLUDE "*myfile.txt*"

If this statement is run as a plug-in within an Application Builder application, the code contained in *myfile.txt* is inserted into the application at that point.

Customized Templates

Application Builder uses an extremely flexible method to generate BASIC source code. A template file is parsed and copied directly into the source file, until a tilde (~) character is reached. All text between ~'s is interpreted as code generation directives. The code generation directives use the information provided by the user via the Application Builder graphical interface to generate custom BASIC code for a particular application. This process is described in detail in the *Developer's Reference Manual*.

By default, Application Builder uses a file called describe.src as a template for generating source code. Some applications have exceptional requirements which cannot be accommodated within the Application Builder interface, even with the use of plug-ins. In these situations, use of a customized template will often provide the required functionality.

Application Builder version 2.1.0 and higher implements the ability to use a template file other than describe.src to generate the source code. If an alternative template is desired, add a plug-in action to the “Begin” handler using the syntax:

$TEMPLATE “*mytemp.src*”

where *mytemp.src* is the name of the template file to be used. The Application Builder software package (version 2.1.0 and higher) includes four template files:

- describe.src – the default template, appropriate for most applications
- timewand.src – similar to describe.src, except that a TimeWand-style raw scan file is produced
- mx.src – generates applications which utilize the LaserLite Mx memory card
- touch.src – generates applications which utilize the TouchMemory read/write capabilities of the DuraTrax and LaserLite Pro

Examples of applications in which use of a customized template is appropriate include modification of the functionality of the scrolling keys or changing the name of the data file which is downloaded from the data collector.

Use of Function Keys (LaserLite Pro and LaserLite Mx only)

Pressing a function key (F1 through F5) on the LaserLite Pro or LaserLite Mx generates an event which is returned by the INPUTEVT statement in Videx BASIC. The template files provided with Application Builder (describe.src, timewand.src, touch.src, and mx.src) include code which simulates a user input when these keys are pressed. For example, if the F1 key is pressed, a user input of “<<F1>>” is generated by the application. If there is a handler available which will accept this input, the application will step to that handler and perform its actions. Note that it would usually not be appropriate to include the action “add input” on this handler, because “<<F1>>” would be written to the data file.

Pressing the F2 key simulates a user input of “<<F2>>”; pressing the F3 key simulates a user input of “<<F3>>”; and pressing the F4 key simulates a user input of “<<F4>>”. Use of handlers which accept these inputs often provides a convenient means of controlling the flow of an application.

Pressing the F5 key is handled differently than the other four function keys. In Application Builder applications, pressing the F5 key invokes a routine which allows the user to delete the most recently scanned data. Repeated presses of the F5 key continues to delete data until the entire data file is cleared. It is possible to disable the F5 key by running a “plug-in” action on the “Begin” handler using the syntax: delete_flag% = false%.

Advanced Application Building Strategies

Application Builder supports four different levels of customization of an application. In order, from the simplest to the most complex, they are:

1) Generate the application within the Application Builder interface program, using the standard template (describe.src) and no plug-ins. This process is described in detail in Chapters 2 and 3 of this manual, and is suitable for the majority of end user applications.
2) Generate the application within the Application Builder interface program, using the standard template (describe.src) and plug-ins for enhanced functionality. Plug-ins are discussed in this chapter; effective use of plug-ins requires some familiarity with programming in BASIC. Most commonly, this level of customization is used when the application requires a data file format which is not supported directly through the Application Builder interface.
3) Generate the application within the Application Builder interface program, but use a customized template. Most commonly, this level of customization is used to modify the functionality of the scrolling, escape, or function keys on the data collector. Again, familiarity with programming in BASIC is required.

4) Generate the application within the Application Builder interface program, and export the source code (refer to the Export Binary… command on page 40). Modify the source code using a text editor and recompile (refer to Chapter 5 of the Videx BASIC manual). This level provides the highest level of customization, but also requires programming skill. Exporting source code from Application Builder is a powerful shortcut compared to writing the source code for an application "from scratch."

Contact Videx Customer Service at 541-758-0521 if additional assistance is needed in creating applications for the data collector.

Plug-In	Function
evt_data$ = *expression*	set input data to **expression**. (requires input device = software) (*refer to audit.abd sample application*)
gosub evt_input	processes **evt_data$** as if it had just been entered into the data collector (*refer to audit.abd sample application*)
gosub ring	wand makes a ringing sound
print #0, *expression*	write expression to the data file.
running% = false%	quit application (*refer to inspect.abd sample application*)
delete_flag% = false%	disables the ability to delete the most recent scan.
sound *frequency*%, *duration*%	wand generates a sound at ***frequency%*** Hz for a duration of ***duration%*** millisec. ***frequency%*** must be between 50-8000. ***duration%*** must be between 1-2000.
option (259) = 1	puts LaserLite Pro or LaserLite Mx into shift mode (to enable keypad entry of letters instead of numbers)
variable$ = environ$(0)	place current battery voltage in ***variable$***
variable$ = environ$(1)	place available RAM in ***variable$***
variable$ = environ$(2)	place operating system version in ***variable$***
variable$ = environ$(3)	place wand ID in ***variable$***

Guard ID Bar Codes located on Guard's badge:

G101	*G102*
G101	*G102*

Location Bar Codes located on doors:

D14	*D17*
D14	*D17*
D26	*D34*
D26	*D34*

Status Bar Codes on Scanpad

S91	Locked	*S92*	Unlocked
S93	Called Police	*S94*	Passed
S95	Maintenance Required		

BATT	*MEM*
Check Battery	Available Memory
ID	* *
Unit's ID	Delete Last Scan

Appendix A

Sample Applications: Descriptions and Sample Bar Codes

Inventory Applications

Secure1.abd

The sample security application is **Secure1.abd**. This application is a three-step security round application that asks for a Guard ID, location, and status of the location. This is the same application used in the Chapter 1 quick-start section and similar to the application built in Chapter 3.

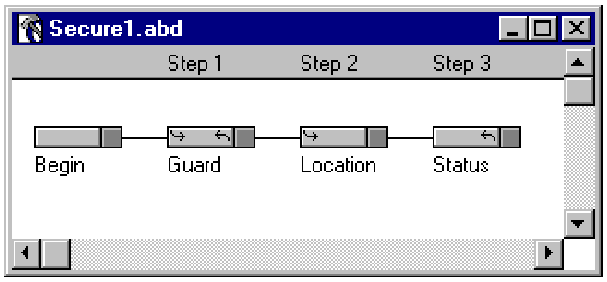
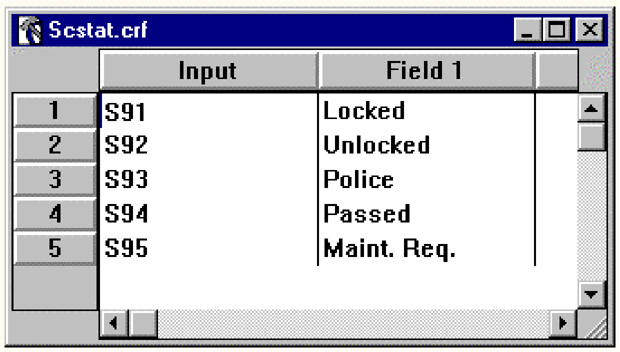


Figure A-1 Secure1.abd Application

The **Secure1.abd** application has Input Handlers for Guard, Location, and Status. The application is designed to expect the following types of entries to meet the match criteria of the Input Handlers:

Guard:	Entry must start with a **G** followed by three digits
Location:	Entry must start with a **D**
Status:	Entry must be contained in the cross-reference file **Scstat.crf** (Figure A-2). **Scstat.crf** is a cross-reference file in the **Samples** folder that contains five records (S91 = Locked, S92 = Unlocked, S93 = Police, S94 = Passed, S95 = Maintenance Required)

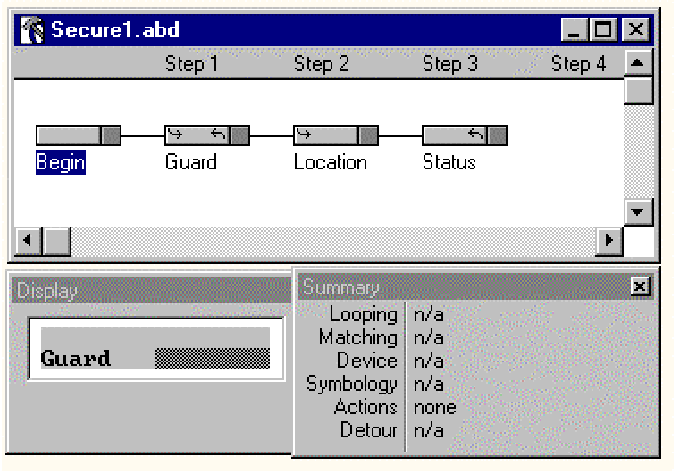
Sample bar codes for the application are on page 200.


Scstat.crf

	Input	Field 1
1	S91	Locked
2	S92	Unlocked
3	S93	Police
4	S94	Passed
5	S95	Maint. Req.

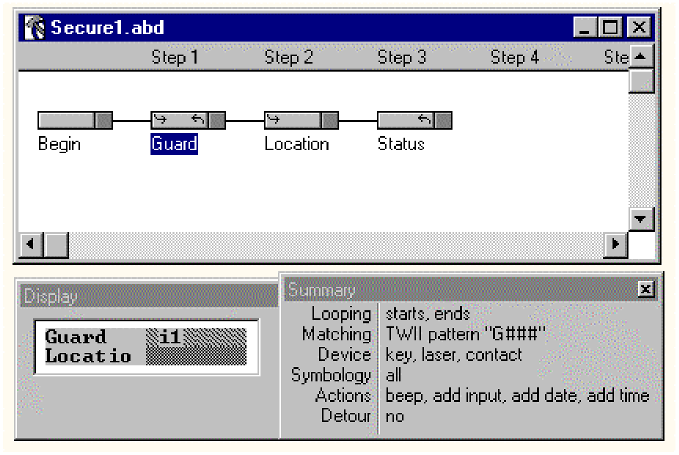
Figure A-2 Scstat.crf Cross-Reference File

Figures A-3 through A-6 show the **Summary** and **Display** windows for each of the **Secure1.abd** application's Input Handlers.
Figure A-3 shows the "Begin" Handler.
Figure A-4 shows the "Guard" Handler.
Figure A-5 shows the "Location" Handler.
Figure A-6 shows the "Status" Handler.



Looping	n/a
Matching	n/a
Device	n/a
Symbology	n/a
Actions	none
Detour	n/a

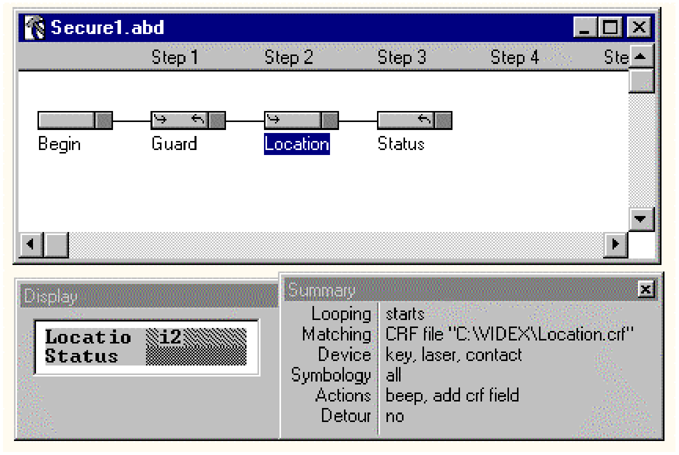
Figure A-3 Secure1.abd Application - "Begin" Handler's Display and Summary Windows



```
Guard     i1
Locatio
```

Looping	starts, ends
Matching	TWII pattern "G###"
Device	key, laser, contact
Symbology	all
Actions	beep, add input, add date, add time
Detour	no

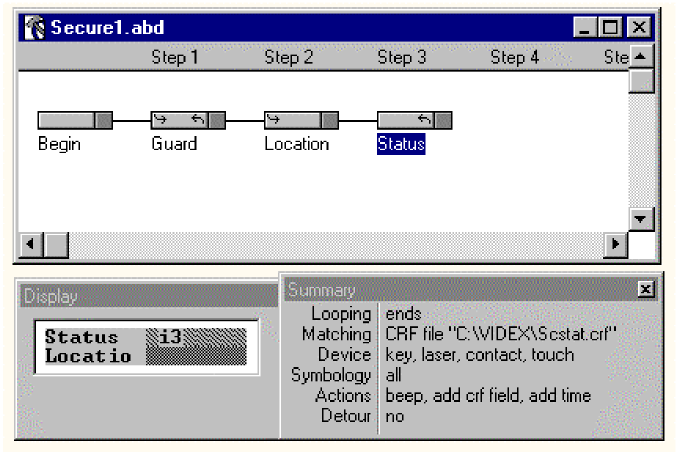
Figure A-4 Secure1.abd Application - “Guard” Handler’s Display and Summary Windows



```
Locatio   i2
Status
```

Looping	starts
Matching	CRF file "C:\VIDEX\Location.crf"
Device	key, laser, contact
Symbology	all
Actions	beep, add crf field
Detour	no

Figure A-5 Secure1.abd Application - “Location” Handler’s Display and Summary Windows



Looping	ends
Matching	CRF file "C:\VIDEX\Scstat.crf"
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add crf field, add time
Detour	no

Figure A-6 Secure1.abd Application - "Status" Handler's Display and Summary Windows

Security Application Bar Codes

Guard ID Bar Codes:

G101   *G102*

G101   *G102*

Location Bar Codes:

D14   *D17*

D14   *D17*

D26   *D34*

D26   *D34*

Status Bar Codes:

S91	Locked	*S92*	Unlocked
S93	Called Police	*S94*	Passed
S95	Maintenance Required		

Delete Last Entry Bar Code for any Application (*5 space bar code*):

*   *

DELETE LAST ENTRY

Audit.abd

The **Audit.abd** application (Figure A-7) automatically steps through the **Audit.CRF** cross-reference file, displaying its contents on the screen and prompting the user for a quantity. This application simulates an inventory audit in which the auditor is told to go to a location and enter the quantity of the item listed on the display. The **Audit.abd** application has Input Handlers for Index#, Loc & Part#, Quantity, and Done.

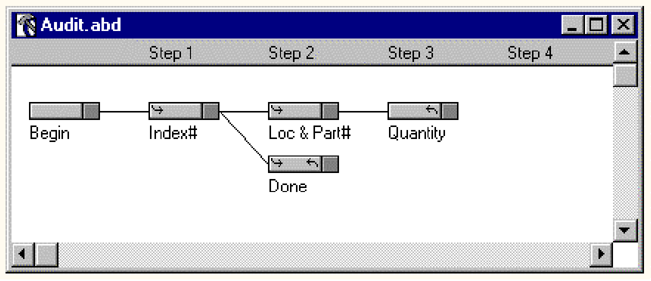
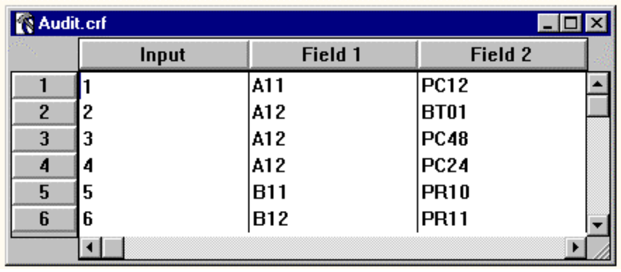


Figure A-7 Audit.abd Application

When you start the application, the “Begin” Handler runs a plug-in, setting a variable named *counter%* to 0. This variable is used by the application to step through the **Audit.CRF** cross-reference file (Figure A-8). The **Audit.CRF** Input column lists the numbers used by *counter%* to step through the file, Field 1 lists the locations of the parts being inventoried, and Field 2 lists the part numbers.



	Input	Field 1	Field 2
1	1	A11	PC12
2	2	A12	BT01
3	3	A12	PC48
4	4	A12	PC24
5	5	B11	PR10
6	6	B12	PR11

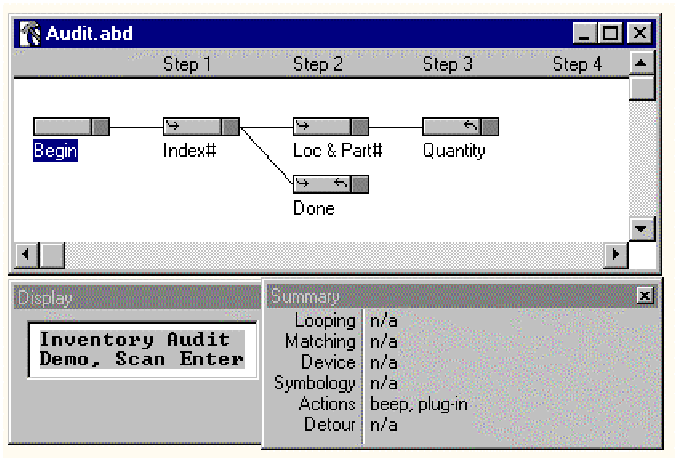
Figure A-8 Audit.crf Cross-Reference File

The “Begin” Handler’s display asks the user to scan an ENTER bar code. When the user scans the ENTER bar code, it is accepted by the “Index” Handler which runs three plug-ins. The first plug-in increments *counter*% by 1 (so *counter*% now equals 1), the second plug-in converts *counter*% to a string, and the third plug-in runs an input event using *counter*% as the input. This automatically generated input is accepted by the “Loc and Part#” Handler if it equals an entry in the **Audit.crf’s** Input column. The “Loc and Part#” Handler beeps and adds the input (*counter*%) and corresponding cross-reference fields to the data file. The user is then prompted to enter a quantity, which is accepted by the “Quantity” Handler. The “Quantity” Handler runs plug-ins incrementing *counter*% by one (so *counter*% now equals 2) and generating input to be accepted by the “Loc and Part#” handler. Each time the “Quantity” Handler is stepped to, *counter*% is again incremented by one. Eventually, *counter*% is incremented to a value that is not in the **Audit.crf** file. The application then steps to the “Done” Handler which stops the application, prompts the user to download the data, and runs a plug-in to reset *counter*% to 0 so that the process may begin again.

Sample bar codes for the application are on page 209.

Figures A-9 through A-23 show the **Summary** and **Display** windows for each of the **Audit.abd** application’s Input Handlers.

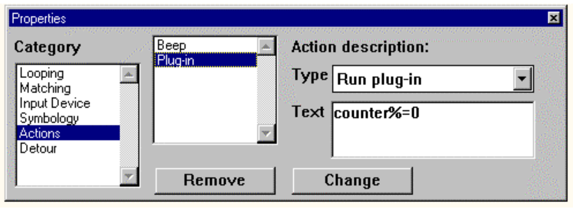
Figures A-9 and A-10 show the “Begin” Handler.
Figures A-11 through A-14 show the “Index#” Handler.
Figures A-15 through A-17 show the “Loc & Part#” Handler.
Figures A-18 through A-21 show the “Quantity” Handler.
Figures A-22 through A-23 show the “Done” Handler.



```
Inventory Audit
Demo, Scan Enter
```

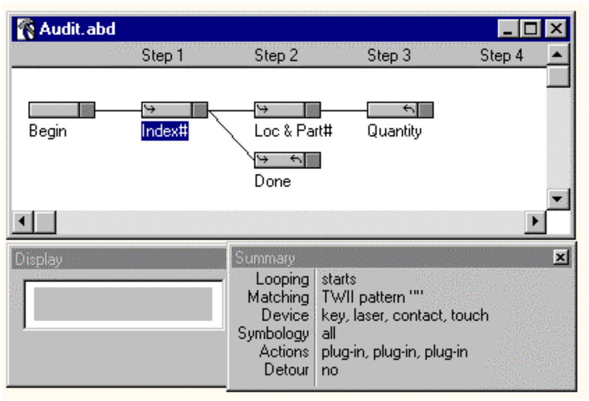
Looping	n/a
Matching	n/a
Device	n/a
Symbology	n/a
Actions	beep, plug-in
Detour	n/a

Figure A-9 Audit.abd Application - “Begin” Handler



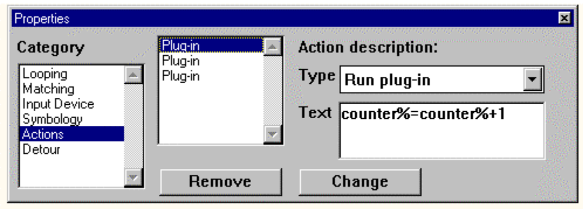
Action description:
Type: Run plug-in
Text: counter%=0

Figure A-10 Audit.abd Application - “Begin” Handler Plug-in



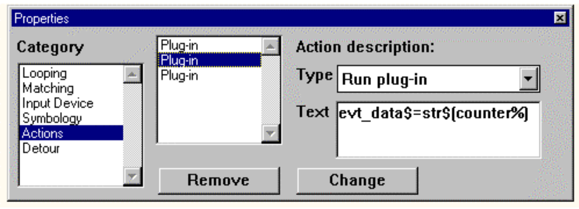
Looping	starts
Matching	TWII pattern ""
Device	key, laser, contact, touch
Symbology	all
Actions	plug-in, plug-in, plug-in
Detour	no

Figure A-11 Audit.abd Application - “Index#” Handler



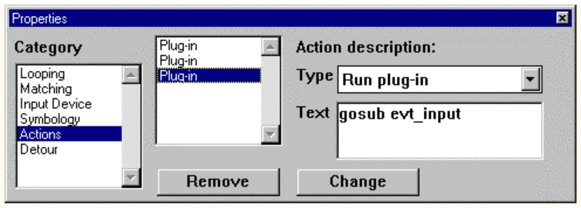
Type: Run plug-in
Text: counter%=counter%+1

Figure A-12 Audit.abd Application - “Index#” Handler First Plug-in



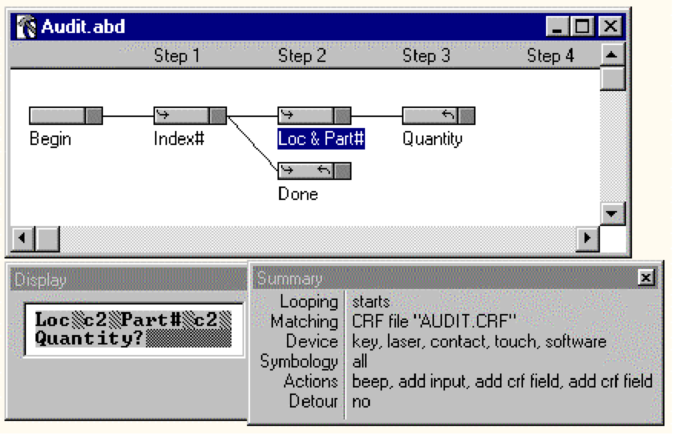
Type: Run plug-in
Text: evt_data$=str$(counter%)

Figure A-13 Audit.abd Application - “Index#” Handler Second Plug-in



Action description:
Type: Run plug-in
Text: gosub evt_input

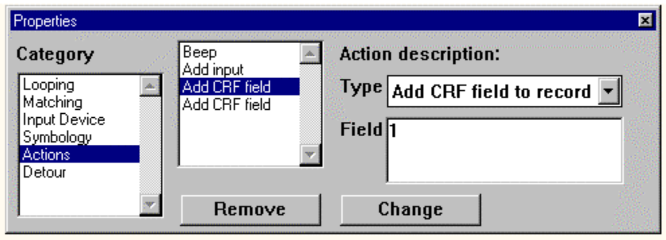
Figure A-14 Audit.abd Application - “Index#” Handler Third Plug-in



Loc▒c2▒Part#▒c2▒
Quantity?▒▒▒▒▒▒▒

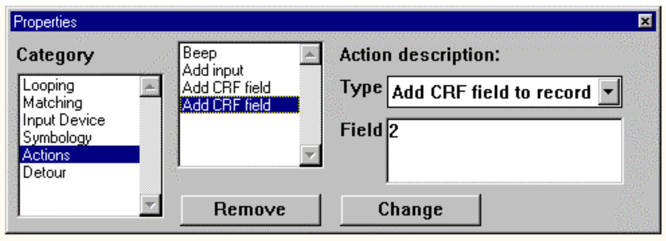
Looping	starts
Matching	CRF file "AUDIT.CRF"
Device	key, laser, contact, touch, software
Symbology	all
Actions	beep, add input, add crf field, add crf field
Detour	no

Figure A-15 Audit.abd Application - “Loc & Part#” Handler



Action description:
Type: Add CRF field to record
Field: 1

Figure A-16 Audit.abd Application - “Loc & Part#” Handler Actions


Properties

Category: Looping, Matching, Input Device, Symbology, Actions, Detour

Beep
Add input
Add CRF field
Add CRF field

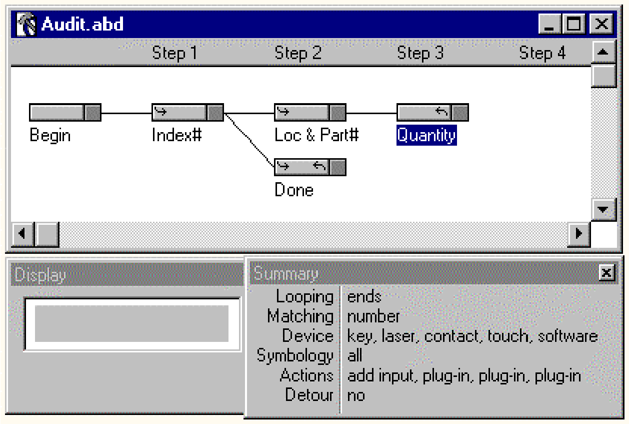
Action description:

Type: Add CRF field to record

Field: 2

Remove | Change

Figure A-17 Audit.abd Application - “Loc & Part#” Handler Actions


Audit.abd

Step 1 | Step 2 | Step 3 | Step 4

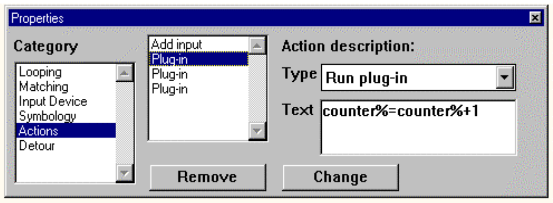
Begin, Index#, Loc & Part#, Quantity, Done

Display

Summary

Looping	ends
Matching	number
Device	key, laser, contact, touch, software
Symbology	all
Actions	add input, plug-in, plug-in, plug-in
Detour	no

Figure A-18 Audit.abd Application - “Quantity” Handler


Properties

Category: Looping, Matching, Input Device, Symbology, Actions, Detour

Add input
Plug-in
Plug-in
Plug-in

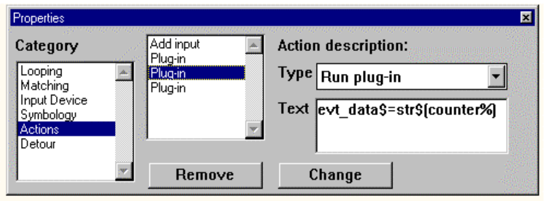
Action description:

Type: Run plug-in

Text: counter%=counter%+1

Remove | Change

Figure A-19 Audit.abd Application - “Quantity” Handler First Plug-in

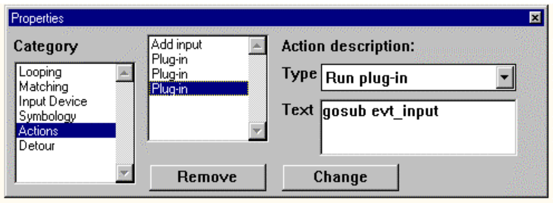


Action description:

Type: Run plug-in

Text: evt_data$=str$(counter%)

Figure A-20 Audit.abd Application - "Quantity" Handler Second Plug-in

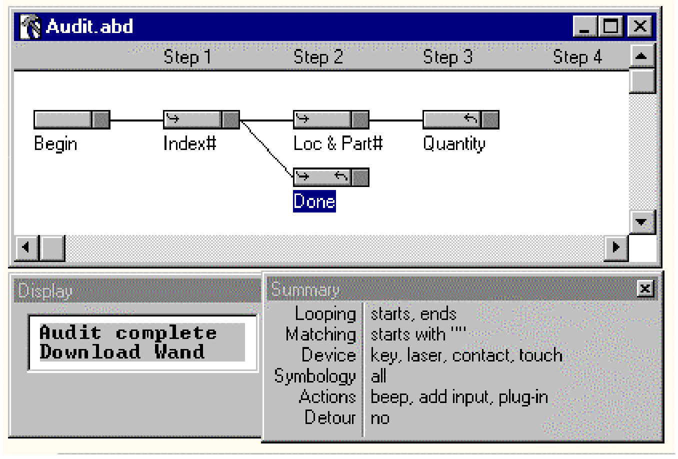


Action description:

Type: Run plug-in

Text: gosub evt_input

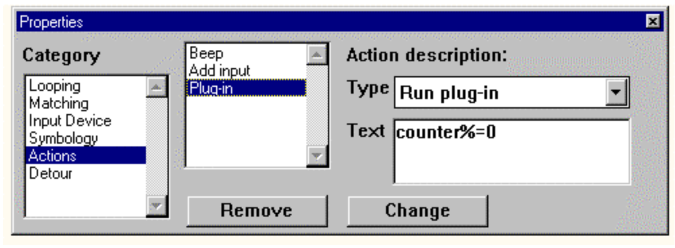
Figure A-21 Audit.abd Application - "Quantity" Handler Third Plug-in



```
Audit complete
Download Wand
```

Looping	starts, ends
Matching	starts with ""
Device	key, laser, contact, touch
Symbology	all
Actions	beep, add input, plug-in
Detour	no

Figure A-22 Audit.abd Application - "Done" Handler



Action description:

Type Run plug-in

Text counter%=0

Figure A-23 Audit.abd Application - “Done” Handler Plug-in

Audit.abd Application Bar Codes

For Audit.abd Sample Application:

1	*2*
1	*2*
3	*4*
3	*4*
5	*6*
5	*6*
7	*8*
7	*8*
9	*0*
9	*0*
* *	* *
Enter	**Clear**

Inspect.abd

Inspect.abd is a sample application which demonstrates some of the advanced features of Application Builder: Detour Handlers, plug-ins, and control over the data download.

The user is prompted to scan in a sequence of entries: “Inspector,” “Equipment Code,” “Model Number,” and “Condition.” In most applications, the data collector can be downloaded at any time. However, in this application, the user wants all of the records–even the last record in the data file–to be complete. Therefore, it is only possible to download after scanning in “Condition,” but not after any of the other steps.

To control the download process, from the “Application” menu, choose “Other details.” A dialog box appears with a check box which enables “Generate <<ESCAPE>> input instead of quitting.” In most applications, this box is not checked. If this box is not checked, when the data collector detects an “unlock” command at the serial port (as when the user is attempting to download it), it quits the application, returns to the operating system command line, and communicates with the computer. If this box is checked, when the data collector detects an “unlock” command, it does not quit the application. Instead, a user input of “<<ESCAPE>>” is simulated, with an input device of “software.”

In the “Inspect” application, the “Download” and “Attempt Download” Input Handlers accept a user input of “<<ESCAPE>>” with an input device of “software.” If a “Condition” has just been scanned, and the user attempts to download the data collector, the application will step to the “Download” handler. The action performed by this handler is to run a simple plug-in which quits the application (“running%=false%”) and allows the data collector to communicate with the computer.

The “Attempt Download” Input Handler is stepped to if the user attempts to download the data collector before completing the last record (scanning a “Condition”). This handler displays a message telling the user to “finish record before download.” It is a Detour Handler, so after a timeout, the application returns to the point it was at immediately before the user attempted to download the data collector.

Inventry.abd

Inventry.abd is a simple inventory application which demonstrates how to use the F1 key on the LaserLite Pro to control the flow of the data collection.

The user is prompted to scan “Location,” followed by prompts to repeatedly scan “Item” followed by “Quantity.” “Quantity” must be a number, but “Location” and “Item” will accept any bar code input. As a result, any bar code which is scanned after entering a numeric “Quantity” is assumed by the application to be an “Item,” not a “Location.”

To change to a new “Location,” the user presses the F1 key on the LaserLite Pro or LaserLite Mx. The application then steps to the “F1” handler, which prompts the user to enter a new “Location.” Pressing the F1 key simulates a user input of “<<F1>>”, which is used as a matching criterion by the F1 handler. The default action “add input” has been removed from this handler, because the input which would be written to the data file would be “<<F1>>”.

Inventry.abd Application Bar Codes

Location Bar Codes:

D14	*D17*
D14	*D17*

Item Bar Codes:

PC12

Fax Machine

PC14

Printer

PC24

Projector

Misc. Bar Codes:

MEM

Memory

BATT

Battery

*   *

DELETE LAST SCAN

Appendix B

Resetting the Data Collector

In the unlikely event that your data collector locks up and does not recover after sitting for two to three minutes, you must manually reset the unit and use **Vxcom** to resend the operating system software to the data collector.

The actual names of the operating system files will change as new versions of the software are released. The operating system software filename will always begin with **Trax** for a DuraTrax, with **Lite** for a LaserLite, with **Pro** for a LaserLite Pro, and with **Lmx** for a LaserLite Mx. The operating system software filename for all three units will always end with a **.OS** extension. (For example, **Trax125a.OS** for a DuraTrax, **Lite125a.OS** for a LaserLite, **Pro125a.OS** for a LaserLite Pro, and **Lmx123a.OS** for a LaserLite Mx.)

To manually reset the unit:

1) Remove the battery end cap.
2) Hold the scan button down while replacing the battery end cap.
3) The unit will beep three times, indicating a successful reset.
4) The DuraTrax or LaserLite will show **VX1 Monitor 1.07** on the first line of the display, and **Ready, bps=9600** on the second line. The LaserLite Pro will show **VX1 Monitor 1.11** or **1.34** on the first line, and **Ready, bps=9600** on the second line. The LaserLite Mx will show **VX1 Monitor 1.33** or **1.34** on the first line, and **Ready, bps=9600** on the second line.
5) Drag the operating system file onto the **Vxcom** icon to resend the operating system software to the unit.

Note: If you have not previously used **Vxcom** to transfer the data file, you will first need to set the **Vxcom** parameters. See Steps 26–31 on pages 21–23 for instructions.

See the *Developer's Reference Manual* for complete information on **Vxcom**.

Notes:

Appendix C

Using Multiple Data Collectors and Base Stations

Up to ten Videx Base Stations can be connected in series to a single serial port; this allows the user to transfer data from multiple data collectors in a single transfer process. If you are using LaserLite Pro or LaserLite Mx data collectors, you can transfer data from up to ten units at a time; and if you are using DuraTrax or LaserLite data collectors, you can transfer data from up to twenty units at a time.

Connecting the Base Stations

To connect multiple Base Stations:

1) Connect the first Base Station to the computer's serial port:
   a) Plug the Base Station's transformer into an electric outlet and connect the other end into the Base Station's **Power** connection.
   b) Use the serial port cable to connect the computer's serial port to the Base Station's **Computer** port.
   c) The Base Station should be placed at least 3 inches away from the computer and monitor.

2) To add additional Base Stations:
   a) Use an interconnect cable (TWC-006) to connect the **Extension** port of the first Base Station to the **Computer** port on the second Base Station. See Figure C-1.
   b) Continue the same pattern of connection for each additional unit.

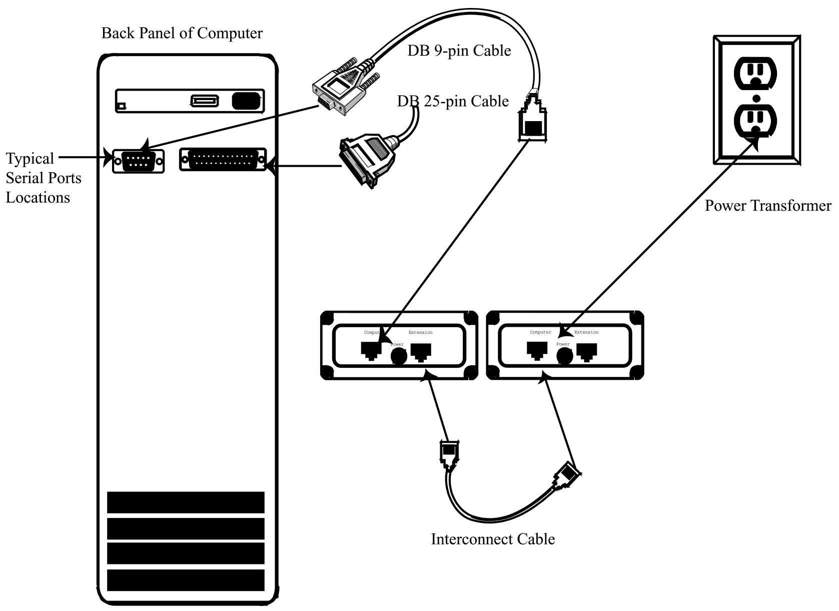


Figure C-1 Base Stations in Series

Exception to connecting ten Base Stations to a single serial port:
If you are using the charge feature on the LaserLite Pro/Mx Base Station and only one power transformer, then you can only connect up to three Base Stations to a single serial port. However, even if using the charge feature, you can still connect up to ten Base Stations if each one is plugged into its own power transformer.

Changing a Data Collector's ID

When using multiple Base Stations, you must give each data collector a unique ID; this allows the computer to access each data collector and verify that it has transferred the data from each one. To assign an ID, you must first create a commands text file, then click and drag the file onto the **Vxcom** icon. **Vxcom** then uses the commands file to unlock the unit and change the ID. (For complete information on **Vxcom** and using a commands text file, see Chapter 1 of the *Developer's Reference Manual*.)

During normal operation, a unit is given an ID of ten zeros. To change the ID, you must create a text file consisting of the following three command lines:

1) An **I** command, followed by a space, followed by ten zeros to unlock the unit.
2) A **C** command, followed by a space, followed by the new ID.
3) An **L** command to relock the unit.

I 0000000000	Sends an **UNLOCK** command to the data collector. (Note: Ten zeros will unlock any data collector.)
C *new ID#*	Changes the unit's ID to the given ID. The ID can be any alphanumeric combination of up to ten characters in length. For example, if you want an ID of 1, this line should be **C 1**; if you want an ID of BuildingD, this line should be **C BuildingD**; if you want an ID of 123#_AB, this line should be **C 123#_AB**, and so on.
L	Sends a **LOCK** command and puts the data collector to sleep. When the unit is awakened with a keypress, it immediately runs the current application.

For example, to change a data collector's ID to **12345**:

1. Use your word processing program to create a text file containing the following three lines:

   ```
   I 0000000000
   C 12345
   L
   ```

2. Save the file as a text file and quit the program. (Note: The text file must have a **.txt** extension.)

3. Place a data collector into the Base Station slot.

IMPORTANT: When you are sending a file to a data collector, it must be the ONLY data collector attached to the computer.

4. Click and drag the text file you created onto the **Vxcom** icon.

5. **Vxcom** executes the commands listed in the text file; it unlocks the unit, changes the ID to what is listed after the **C** command (in this case, 12345), and then relocks the unit.

6. Remove your data collector from the Base Station slot; your data collector is ready to use.

To continue changing other data collectors' IDs:

1. Reopen the text file.

2. Change the ID listed after the **C** command by selecting **12345** and typing in a new ID. (Note: There must be a space character between the **C** and the ID.)

3. Save and close the text file. Remember, the text file must have a **.txt** extension.

4. Insert a data collector into the Base Station slot.

5. Click and drag the edited text file to the **Vxcom** icon.

6. **Vxcom** executes the commands listed in the text file; it unlocks the unit, changes its ID to what is listed after the **C** command, and then relocks the unit.

7. Continue these same steps until you have assigned a unique ID to each of your units.

Note: Keep a list of the IDs you are using to identify your data collectors. You will use this list in the next section, *Transferring Data from Multiple Data Collectors*.

Transferring Data from Multiple Data Collectors

To transfer data from a group of data collectors in one transfer process, you must:

1. Create a commands text file that uses the **FOREVER**, **FOREACH**, or **FORALL** looping commands to transfer the data from the data collectors. (Note: The commands file must have a **.txt** extension. For complete information on using commands files, see Chapter 1 of the *Developer's Reference Manual*.)
2. Insert all the data collectors into their Base Stations.
3. Click and drag the commands text file onto the **Vxcom** icon.

The three looping commands provide transfer methods for three different situations. The **FOREVER** loop is useful if you have a computer dedicated to transferring the data from the data collectors. The **FOREACH** loop is useful if you will be transferring the data from only a portion of your data collectors. The **FORALL** loop is useful if you require that all of the data collectors be transferred during the process.

The following table describes the looping commands:

FOREVER	For use with a computer that is dedicated to transferring data. This loop will transfer the data from any data collector it locates. This loop attempts to execute commands within a loop indefinitely.
FOREACH	This loop will search for each data collector in the list of IDs. If it locates the data collector with the ID, it executes the commands within the loop once; if it does not locate the data collector with the ID, it goes on to the next ID. This loop attempts to execute commands within a loop once for each ID listed.
FORALL	This loop will look for each listed data collector's ID until it has located and transferred the data from each ID in the list. This loop attempts to execute commands within a loop indefinitely, until all listed IDs have been located.

FOREVER Looping Command

The following example shows how the **FOREVER** loop is used in a commands text file to continually look for data collectors to transfer.

Example FOREVER Commands File:

Example:

```
FOREVER                     'Begin the loop.
    I 0000000000
    'Unlock any unit.
    M1 Downloading Data     'Put message on unit’s display.
    S                       'Download data from unit.
    Z                       'Clear data from unit.
    T                       'Set time by computer’s time.
    G                       'Restart the application.
END                         'End loop, begin re-executing at top.
```

Note on the UNLOCK command (I id): No additional characters should be included on the same line as the **UNLOCK** command. Any characters after the unit ID (for example, space characters or comments) will cause this command to fail. In the above example, the comment associated with the **UNLOCK** command is on a separate line.

FOREACH Looping Command

The following example shows how the **FOREACH** loop is used in a commands text file to attempt to transfer the data from each data collector in the list once.

Example FOREACH Commands File:

You can either list the IDs individually as in the first line of Example 1, or you can make a text file of the IDs and use the filename in quotes as in the first line of Example 2.

Example 1 Commands Text File for Transferring Data from Multiple Data Collectors (IDs listed on command line):

```
FOREACH ID1, ID2, ID3       'Begin the loop. All IDs listed on line.
    I loop_id
'Unlock the unit using the special identifier. This identifier substitutes
'for all the units listed on the first line.
    M1 Downloading Data     'Put message on unit’s display.
    S                       'Download data from unit.
    Z                       'Clear data from unit.
    T                       'Set time by computer’s time.
    L                       'Lock the unit and put it to sleep.
END                         'End loop, begin re-executing at top.
```

Note on the loop_id argument: When the argument of the **UNLOCK** command is a specific ID (as in the previous **FOREVER** example), the program attempts to unlock the unit eight times at five second intervals. This ensures that a unit is unlocked regardless of its sleep cycle (a unit sleeps for 25 seconds, then wakes for 5 seconds). If the program does not unlock a unit, the program returns an error and aborts. On the other hand, when the argument of the **UNLOCK** command is loop_id (as in the **FOREACH** example), the program only makes one attempt to unlock the unit. If it is successful, it performs the remaining commands in the loop; if not successful, it moves on to the next value for loop_id without aborting the program. Given the unit’s sleep cycle, it is unlikely that one attempt will successfully unlock a unit. Therefore, it would generally not be reasonable to use a **FOREACH** loop by itself as in the above example. For examples showing appropriate use of a **FOREACH** loop, refer to the discussions of Scenarios 3 and 4 on pages 226 - 227.

Example 2 Commands Text File for Transferring Data from Multiple Data Collectors (IDs listed in **IDs.txt** file):

```
FOREACH "IDs.txt"           'Begins the loop. All IDs listed in
                            'the IDs.txt file.
    I loop_id
'Unlock the unit using the special identifier. This identifier substitutes
'for all the units listed on the first line.
    M1 Downloading Data     'Put message on unit's display.
    S                       'Download data from unit.
    Z                       'Clear data from unit.
    T                       'Set time by computer's time.
    L                       'Lock the unit and put it to sleep.
END                         'End loop, begin re-executing at top.
```

The **IDs.txt** file must contain a list of the IDs. The IDs in the text file must be separated by commas or spaces.

FORALL Looping Command

The following two examples show how the **FORALL** loop is used in a commands text file to transfer the data from each data collector in the list.

Example FORALL Commands File:

You can either list the IDs individually as in the first line of Example 1, or you can make a text file of the IDs and use the filename in quotes as in the first line of Example 2.

Example 1 Commands Text File for Transferring Data from Multiple Data Collectors (IDs listed on command line):

```
FORALL ID1, ID2, ..., IDN    'Begins the loop. All IDs are listed on
                             'this line.
    I loop_id
'Unlock the unit using the special identifier. This rotates through
'identifiers for each unit.

    M1 Downloading Data      'Put message on unit's display.
    S                        'Download data from unit.
    Z                        'Clear data from unit.
    T                        'Set time by computer's time.
    L                        'Lock the unit and put it to sleep.
END                          'End loop, begin re-executing at top.
```

Example 2 Commands Text File for Transferring Data from Multiple Data Collectors (IDs listed in **IDs.txt** file):

```
FORALL "IDs.txt"             'Begin the loop.
    I loop_id
'Unlock the unit using the special identifier. This identifier substitutes
'for all the units in the IDs.txt file.
    M1 Downloading Data      'Put message on unit’s display.
    S                        'Download data from unit.
    Z                        'Clear data from unit.
    T                        'Set time by computer’s time.
    L                        'Lock the unit and put it to sleep.
END                          'End loop, begin re-executing at top.
```

The **IDs.txt** file must contain a list of the IDs. The IDs in the list must be separated by commas or spaces.

In order to clarify the functioning of these different looping structures, it is helpful to consider the four following scenarios for downloading multiple units.

Scenario #1: The user has multiple data collectors to download, but only one at a time will be put in the Base Station or in front of the external IR transceiver. A computer is dedicated to downloading the data collectors as they become available. In this situation, the units do not need to be assigned individual IDs and a **FOREVER** loop can be used:

```
FOREVER
I 0000000000
M1 Downloading Data
S
Z
T
L
END
```

This loop will continue indefinitely, unlocking any data collector that is available. It is important that only one unit at a time be in the Base Station or in front of the external IR transceiver when using this looping structure. This loop will unlock any data collector; if more than one data collector is unlocked at a time, communications will fail.

Scenario #2: The user has multiple data collectors to download. Some or all of them may be in the Base Station at any time. Each of the units must be downloaded exactly one time. A computer is dedicated to downloading the units as they become available. In this situation, each unit must be assigned a unique ID and the **FORALL** loop can be used:

```
FORALL "IDs.txt"
        I loop_id
        M1 Downloading Data
        S
        Z
        T
        L
END
```

This loop will continue until each unit in the IDs.txt file has been downloaded once. Multiple units can be in the Base Station at one time because only one unit will be unlocked at a time.

Scenario #3: The user has multiple data collectors to download. Some or all of them may be in the Base Station at any time. Each unit is to be downloaded when it is placed in the Base Station; some of the units may be downloaded more than one time. A computer is dedicated to downloading the units as they become available. In this situation, each unit must be assigned a unique ID and a **FOREACH** loop nested inside a **FOREVER** loop can be used:

```
FOREVER
        FOREACH "IDs.txt"
                I loop_id
                M1 Downloading Data
                S
                Z
                T
                L
        END
END
```

This loop will continue indefinitely, continually attempting to unlock any units contained in the **IDs.txt** file. Unlike the **FORALL** loop in Scenario #2, this loop will continue attempting to download a data collector even after it has been successfully downloaded one time.

Scenario #4: The user has multiple data collectors to download; all of the units to be downloaded are in their Base Stations. However, some of the units (listed in the **IDs.txt** file) are not available to download. If the computer is unable to unlock a particular unit, the computer will stop trying to unlock that unit. In this situation, each unit must have a unique ID and a **FOREACH** loop can be used:

```
FOREACH "IDs.txt"
        I loop_id
        M1 Downloading Data
        S
        Z
        T
        L
END
```

As was explained in the earlier discussion of the **FOREACH** loop, when the argument of the **UNLOCK** command is **loop_id**, the program only makes one attempt to unlock the unit. Given the unit's sleep cycle, it is unlikely that one attempt will successfully unlock a unit. However, it is possible to ensure that each unit is awake when the program attempts to unlock it. This is done by placing six unused IDs at the beginning of the **IDs.txt** file (Figure C-2).

```
Dummy1
Dummy2
Dummy3
Dummy4
Dummy5
Dummy6
ValidID1
ValidID2
ValidID3
ValidID4
```

Figure C-2 Sample IDs.txt File

If a unit is awake and receives an **UNLOCK** command with the wrong ID, it will stay awake as long as it detects activity on the serial port. By placing six unused IDs at the beginning of the **IDs.txt** file, it is possible to ensure that each unit receives at least one **UNLOCK** command with an invalid ID and then stays awake. Then each unit is awake when the **FOREACH** loop attempts to unlock it and the unit is downloaded. The program only attempts to unlock each unit one time; if a unit is not present, the program moves on to the next ID. The program quits after attempting one time to unlock and download each unit.

For complete information on using the commands file, see Chapter 1 of the *Developer's Reference Manual*.

Notes on Using Multiple Data Collectors

To avoid problems when using multiple data collectors and Base Stations, you need to be aware of the following cautions:

- Each of the data collectors used in a multiple Base Station arrangement must have a unique ID. See the preceding section in this appendix for information on changing a data collector's ID.
- When the data transfer process begins, do not remove a data collector from its Base Station, quit the program, or shut down your computer until the transfer process is completed.

CAUTION:
Do not attempt to send an application to more than one data collector at a time in a multiple Base Station set up. The units will not be damaged, but they will not be properly programmed.
When transferring data from the data collectors, do not interrupt the process by removing the unit from its Base Station or aborting the program. The system is designed to avoid data loss, but observing this caution provides added protection.

If you should inadvertently remove a data collector while it is in the process of transferring data, the software displays a message that it was unable to communicate with the unit and then continues to try to transfer the data from all of the other data collectors. The data in the interrupted unit will not be cleared and can be transferred after the others have completed their data transfer.

Key Points on Using Multiple Data Collectors

- Each data collector must have a unique ID.
- When sending files to a data collector, it must be the only unit in a Base Station.
- Do not interrupt the data transfer process.
- Data is not cleared from a data collector until all of the data is correctly transferred to the computer.

Notes:

Index