JOURNAL OF APPLIED BEHAVIOR ANALYSIS 2003, **36,** 271–284 NUMBER 2 (SUMMER 2003)

CREATING A PORTABLE DATA-COLLECTION SYSTEM WITH MICROSOFT® EMBEDDED VISUAL TOOLS FOR THE POCKET PC

MARK R. DIXON

SOUTHERN ILLINOIS UNIVERSITY

This paper describes an overview and illustrative example for creating a portable data-collection system using Microsoft® Embedded Visual Tools for the Pocket PC. A description of the Visual Basic® programming language is given, along with examples of computer code procedures for developing data-collection software. Program specifications, strategies for customizing the collection system, and troubleshooting tips are also provided.

DESCRIPTORS: data collection, data analysis, computer, Visual Basic®, software, Pocket PC

There have been numerous attempts to reduce the amount of time and effort required to collect behavioral data. Methods have included using alarms or timers to prompt recorders (Sulzer-Azaroff & Mayer, 1991), videotaping sessions for post hoc recording (Miltenberger, Rapp, & Long, 1999), and purchasing a computerized system for automated recording (Bellack & Hersen, 1998). As technology advances and prices continue to decline, computerized systems are being developed for a greater variety of applications.

There are a number of reasons why one might wish to consider a computerized system of recording over traditional pencil and paper or other low-tech forms of collection. First, pencil-and-paper data collection is often time consuming. Requiring staff to write out descriptions of contingencies or remember the specific moment to record an instance of behavior are difficult endeavors with high response costs (Emerson, Reeves, & Felce, 2000). Likewise, traditional systems are prone to problems such as incompleteness, inaccuracies, and poor reliability (Bellack & Hersen, 1998).

Computer-based data-collection methods represent a useful advancement in the area of observational data collection. Kahng and Iwata (1998) reviewed many of the existing forms of behaviorally oriented computerized data-collection systems that have claimed to ease the job of data collection for the service provider and provide more accuracy and reliability for the researcher. Although the systems described by Kahng and Iwata offer the consumer a great advantage over pencil-and-paper systems, they have some limitations. First, although 3 of the 15 systems reviewed were available free, others cost from $50 to $1,740 plus the purchase of the necessary hardware, which may preclude their use in some situations. A second and potentially more problematic limitation is that the systems reviewed are static. To illustrate, when a computerized data-collection system is purchased, the consumer is investing in an existing technology and software that may meet the majority of their data-collection needs (e.g., frequency or interval recording). However, if unusual circumstances develop that require a novel type of data collection

This paper was made possible in part by additional equipment funding provided by the College of Education and Human Services, Southern Illinois University–Carbondale. Special thanks to Henry Roane for his continued technical assistance during the editorial process of this paper.

Address correspondence to Mark R. Dixon, Behavior Analysis and Therapy Program, Rehabilitation Institute, Southern Illinois University, Carbondale, Illinois 62901 (e-mail: mdixon@siu.edu).

(e.g., collecting data in discrete trials), a standardized system may not be adaptable. Third, the system may operate on obsolete technology. For example, all of the programs reviewed by Kahng and Iwata run in DOS or Microsoft Windows. Since the time of that publication, Microsoft has introduced four newer versions of its operating system (98, 2000, ME, and XP). In such cases, a program's operating system may not be compatible with more recent software (e.g., word-processing software) and may require the consumer to retrofit a computer for the data-collection software, and then perhaps to transfer the data into more contemporary programs for other functions (e.g., analysis or graphing).

A potential solution to the limitations of preexisting computerized data-collection systems is to develop a computerized data-collection system that is individualized to the user's specific needs. If a consumer can develop, modify, and expand his or her own data-collection system, it should yield greater utility than other existing methods of collection.

Designing Software on the Pocket PC

A Pocket PC is a small handheld computerized device that measures roughly 8 cm by 12 cm and costs $150 to $600. The machines are available from a number of manufacturers (e.g., Casio, Dell, Hewlett-Packard), and all use the Microsoft Windows CE operating system. Windows CE is Microsoft's portable version of the well-known Windows 98, 2000, XP operating systems and accommodates modified versions of many familiar Microsoft desktop software programs such as Word, Excel, and Outlook. Assuming a user has a Microsoft-based desktop computer, having similar programs on a portable device such as the Pocket PC allows easy communication and data transfer between devices.

Of all the software-sharing capabilities between the desktop and Pocket PC device, the most practical for developing a data-collection system is Microsoft's Embedded Visual Tools (eVT). The eVT software allows a relatively novice computer programmer to design his or her own software on a desktop computer and then download and run it on a Pocket PC. Furthermore, this software is available for free download at http://www.microsoft.com/mobile/developer/downloads/default.asp.

According to Microsoft, the system requirements for eVT include a computer with Pentium processor at 150 MHz or higher, a recent version of Microsoft Windows (Windows 98 2nd ed., Windows NT Version 4.0 with Service Pack 5 or later, Windows 2000, or Windows XP professional ed.), 48 MB of RAM, hard-drive space of 360 MB for minimum installation (720 MB for complete installation), a CD-ROM drive, a VGA or higher resolution monitor, and a mouse or compatible pointing device. This author strongly recommends the following upgrades for smoother operation: a Pentium III or better processor, Windows 2000, XP professional ed. or higher, 128 MG of RAM, and 1 gigabyte of hard-drive space.

Most currently available Pocket PCs will meet the minimum requirements for programming in eVT, which include 32 MG of RAM and the Windows CE 2.0 or higher operating system. Newer operating systems may require an additional package of files (called a System Development Kit or SDK) to be downloaded at the Microsoft Web site noted above.

Once the eVT software is downloaded and installed on a compatible system, the user can begin programming. It is assumed that the user will have previously installed the communication software package included with his or her specific brand of Pocket PC.

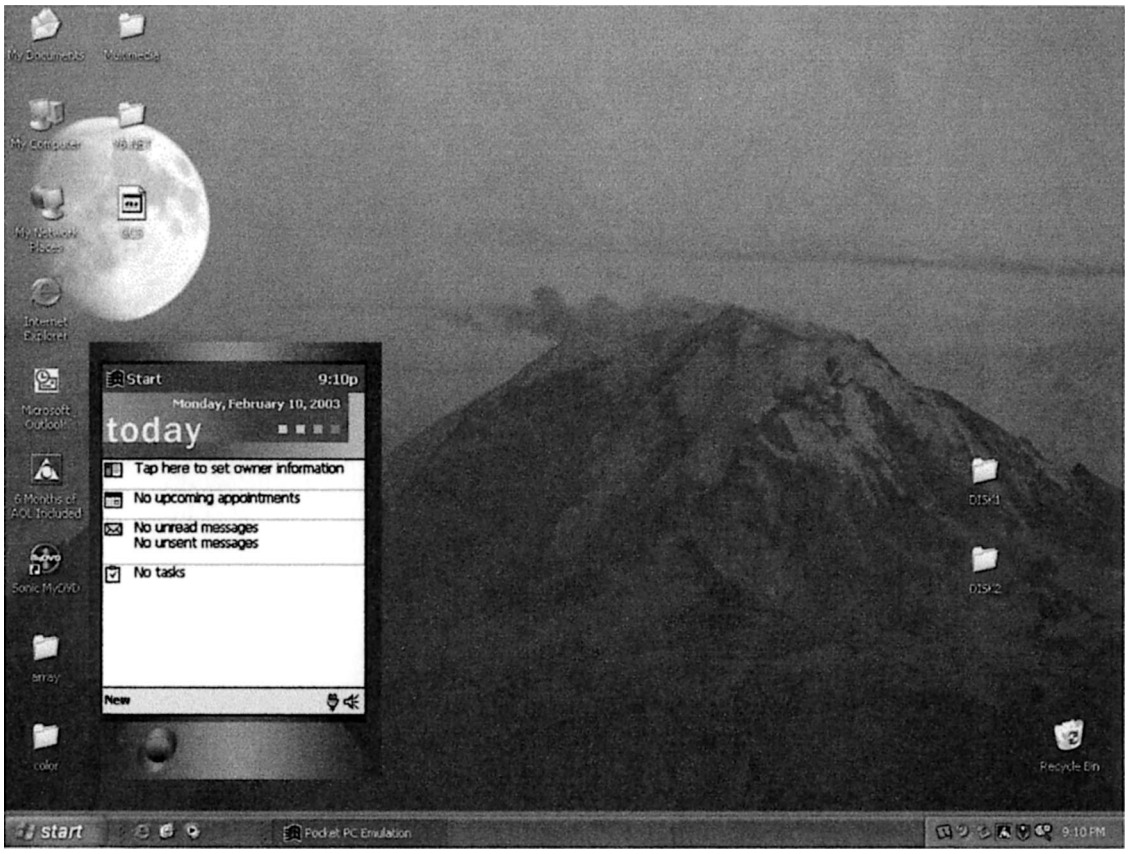

Figure 1. Graphic display of the PDA emulator.

PDA Emulator

For users who are hesitant about the viability of a self-created data-collection program, the eVT download also includes an emulator or virtual PDA (personal digital assistant) component. The PDA emulator allows the user to test the program before downloading it to an actual Pocket PC. This function also permits data collection using the emulator alone, although the user would require a portable computer to run the emulator. Figure 1 shows a graphic display of the PDA emulator.

To explore the emulator prior to programming, the user should click on the "start" button in the bottom left corner of the screen, click on "Microsoft Windows Platform SDK for Pocket PC," and finally on "Desktop Pocket PC Emulation." A blue image (7.6 cm by 12.7 cm) will appear on the screen. Another "start" button is located on the emulator to guide navigation within the emulator. It should be noted that the PDA emulator is not compatible with some versions of Windows (see Table 1 for details). When the eVT program is being downloaded, users who do not have systems that are compatible with the PDA emulator will be prompted to continue installation although the emulator will not work. Continuing installation allows all other components of eVT to work properly, even though the emulator has not been installed.

Layout of the eVT Interface

To use the emulator during actual programming, the user must change the drop-down tool menu in embedded Visual Basic

Table 1
A Summary of Possible Problems and Possible Solutions to Programming in eVT

Problem	Possible solutions
Software is unable to transfer over to Pocket PC	Check communications cable, check to make sure Pocket PC is placed firmly in docking cradle, reboot system
Software takes unusually long time to compile onto Pocket PC	Free additional memory space on Pocket PC, upgrade version of Microsoft CE (if needed), free desktop space
eVT takes long time to download from Web site	Use a non-dial-up modem (cable or DSL), request a CD copy from Microsoft
PDA emulator does not work or cannot be found	Make sure you are using Windows 2000 Professional or XP professional; Windows 98 or ME are not compatible
Data output file cannot be found	Check "My Device" directory on PDA, or use "Find Files" option on desktop if using emulator
User makes errors when recording data	Create a command button named "Error," have user click to note error, and link response to data file
Programmer wants to note time and date	Add the code "File1.LinePrint Time & Date" under a command button
Programmer wants to terminate program during run time	Add timer object from toolbar and change the interval property to desired duration (in ms). Add the code "App.End" under Private Sub Timer1_Timer().
Programmer wants to track frequency of behavior	Declare variable in Option Explicit; add the code, "(variable name) = (variable name) + 1" under command button; display value in text box
Variable values do not transfer across multiple forms	Add a module, declare variable in Option Explicit on module by the code "Public (variable name); delete all declarations on individual forms
PDA screen is too small for desired data collection	Use multiple forms, resize objects, explore the use of ComboBoxes
Gray screen color undesirable	Select "BackColor" from properties window, and select the color of your choice
Border style of form or objects undesirable	Select one of the various options of "BorderStyle" from properties window
Message appears "Unable to contact host device" when using the "Run" . . . "Start Debug" sequence	Retry the testing sequence; reboot desktop if necessary (Note: this is a common problem)
Message appears "Insufficient memory to transfer files" when using the "Run" . . . "Start Debug" sequence	Retry the testing sequence; reboot desktop if necessary (Note: this is a common problem)
Cannot find the data file to transfer into Excel	Use the "Find Files" option; make sure Excel's file types option "All files" is selected
Data in Excel all appear in one row	Make sure to import data using the "delimited" and "comma" options in Excel

(eVB) from "Pocket PC (default device)" to "Pocket PC emulation." This drop-down menu is located directly underneath the "window" and "help" options on the main toolbar.

To begin programming, the user must first click on the "start" button located at the bottom left corner of the desktop computer screen. Next the user should click on "programs," then on "Microsoft Embedded Visual Tools," and finally on "Embedded Visual Basic 3.0." eVB is automatically installed in the download of eVT, which is the specific programming language that is used to create the data-collection program. There are many similarities between both the interface and code of Embedded Visual Basic 3.0 and Microsoft Visual Basic 6.0/Visual Basic.NET that were designed for desktop program development. (For information on

programming desktop computers with Visual Basic.NET, readers should refer to Dixon & MacLin, 2003.) Once eVB is opened, a "new project" menu will appear and the user can select from four options. At this point, the user should select "Windows CE for the Pocket PC" and then click "open."

The top panel of Figure 2 displays the graphic interface of the program. Four smaller windows appear within the program's main window. One of these smaller windows is labeled "Project1" and another is labeled "Form1." Each program created is technically called a "project." The "Form" is essentially the display that will be projected on the Pocket PC. Each project may have more than one form, although the default setting is for one.

The other two windows are named "Project-Project1" and "Properties-Form1." The Project-Project1 window contains a file folder with a file entitled "Form1." It serves as a visual directory of the contents of the program or project. If more forms or files are added to the project, these would be listed here as well. The Properties-Form1 window lists a variety of properties of Form1, including the form's name, color, and size.

An Illustrative Example: Time-Based Collection Routines

Graphic display set-up. The simplest form of data collection to program is time based. Using the Pocket PC, an observer can record time-based measures such as response duration, response latency, session duration, and intertrial intervals. The program can also be written to prompt the user to record responses at pre-specified intervals (e.g., every 10 s), which may be useful for time-sampling data-collection procedures.

To the left of the project and form is a vertical toolbox. The label corresponding to each tool will appear when the user holds the pointer over the objects in the toolbox. To add an object from the toolbox to the form, the user must click on the type of object in the toolbox (e.g., a text box), and then click on the form. Once the object is on the form, it may be dragged and its dimensions stretched by moving the mouse pointer to the desired dimensions. For the purpose of writing a data-collection program, one of the more useful objects in the toolbox is called the "command" button (represented as a gray rectangle four objects down on the left side of the toolbox). Command buttons are useful because they are objects that, when clicked while a program is running, will trigger events such as starting or ending a timer, writing data to file, or selecting a specific item from a list of options.

To place a command button (or any other object) onto the form, the user must first click on the command button and then move the cursor onto the form. The mouse pointer will turn into a small cross. Next, the user should depress the left mouse button and hold it down while moving the mouse to the left or right. The outline of a box will appear. Moving the mouse (while still depressing the left button) in any direction allows adjustment for the size of the box. When this outline is approximately the desired size of the command button, the mouse button should be released. This process will add a command button to the form. Once inserted, the command button can be resized or moved within the form.

At this point, the Properties-Form1 window (seen in the lower right corner of the screen, bottom panel of Figure 2) will have changed names to "Properties-Command1" because a command button has been added. As before, the size and location of the command button can be changed, along with many other features, through the "properties" button. Potential changes include the name, the caption (what text is written directly on the button), and the color. For example, to change the caption to "begin," the

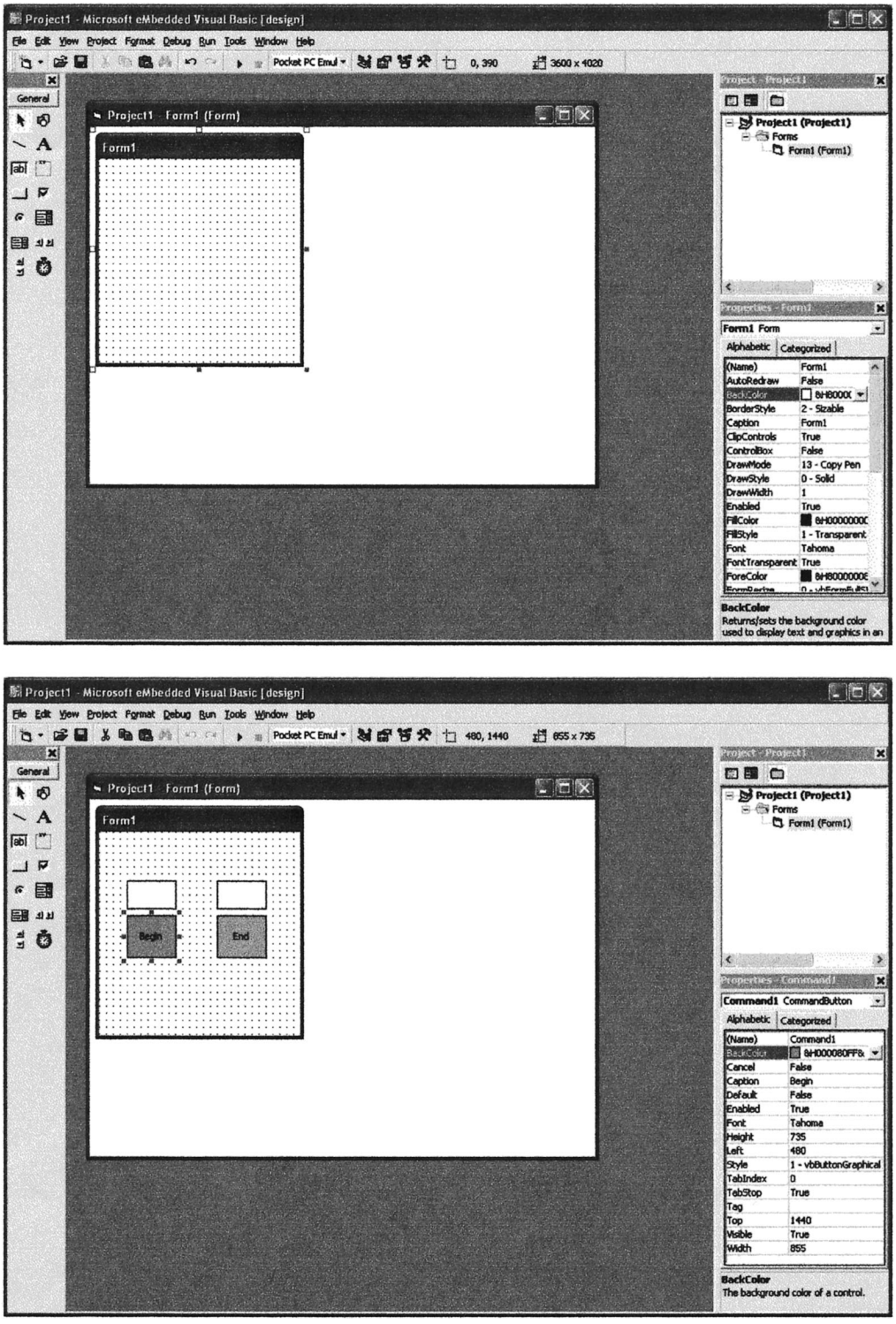

Figure 2. Screen shots of the graphic interface of eVB and the command buttons and text box for display of time-based data collection.

user would click on the "caption" option, highlight the existing text (which should read "Command1" at this point), retype the word "begin," and press return. To change the color of the command button, the user would first click on "backcolor" and then on the downward arrow that appears to the right of this description. Once the desired color is selected, the user must scroll down in the Properties-command1 window until the "style" property appears. To the right of this text, the user should select the second option "vbButtonGraphical" and the color will change. These steps are also used for adding additional command buttons. To illustrate, the user could insert a second command button and change its caption to "end."

Also in the toolbox is an object called a "text box" (shown in the toolbox as a white box with the letters "ab" inside). A text box can be inserted into the form using the method described above (for adding a command button). Once inserted, a rectangular box with the word "Text1" will appear in the form. Again, the name of the properties-command window will change. In this case, it will read "Properties-Text1." If the properties window displays something other than "Properties-Text1," the user may use the drop-down menu at the top of this window and select the object that should be modified. As with the command button, features of the text box (e.g., name, color, border style, and font) can be changed. For the purpose of this example, the user should change the text to a blank space by clicking on the Properties-Text1 window, highlighting the "Text1" text next to the "text" property, and deleting "Text1." This process will result in an empty text box on the form. This process may be repeated to add a second text box. The lower panel of Figure 2 displays the resulting layout of the form.

Writing code for data collection. Continuing with the example described above, the user can write the program to create a basic data-collection program. First, the user will double-click on the "begin" command button. This will result in a new window opening over the form that is titled "Project1-Form1 (code)." This window shows the actual programming syntax that the computer reads when the program is run. Initially, the curser will be flashing underneath a line of text entitled "Private Sub Command1_Click( )." This refers to a subroutine that the computer will perform when an observer clicks on the "begin" command button (developed above). To write a program that can be used to calculate response duration, the user should type the following line of code on the line directly below "Private Sub Command1_Click( )":

```
vTime1 = Timer
```

This code creates a variable entitled "vTime1" (the name of the variable is arbitrary) that will be equal to the computer's internal clock time (i.e., "timer" in the above code). Once written, this code will activate the time when the observer clicks on the "begin" command button.

Additional code is required to deactivate the timer. This code should be written under the "end" command button. When the user double-clicks on the "end" command button, the "Project1-Form1 (code)" window will appear, showing the cursor flashing underneath a line of text entitled "Private Sub Command2_Click( )." This refers to a subroutine that the computer will perform when an observer clicks on the "end" command button. To deactivate the timer, the following lines of code must be written directly below "Private Sub Command2_Click( )":

```
vTime2 = Timer − vTime1
Text1.Text = vTime2
```

This first line of code creates another variable called "vTime2" (again the name of the

variable is arbitrary). However, this variable must be equal to the difference between the current computer clock time and the computer clock time when the observer clicked on the "begin" command button. That is, the computer must subtract the current computer time from the variable "vTime1." The second line of code presents the time-difference calculation in the empty text box on the form.

To collect a response-latency measure, additional timers are required. Specifically latency will measure the time from the observer clicking the "end" button until the observer clicks the "begin" button again. To accomplish this, the user should write the following line of code under the "end" command button below the previously written code:

```
vTime3 = Timer
```

This code allows the program to capture the current computer time and store it in a variable named "vTime3." This value may be displayed in the second text box by typing an additional two lines of code under the "begin" button below the previously written code:

```
vTime4 = Timer − vTime3
Text2.Text = vTime4
```

At this point, response latencies will appear in the second text box each time the "begin" button is clicked.

Before testing the program, the variables must be "declared." This process allows the computer to recognize the text (e.g., vTime1, vTime2) as variables rather than text. To the right of the text "Command2" directly underneath the window heading of the "Project1-Form1 (code)" window is a downward arrow. Clicking on that arrow allows the user to select the option "(general)." Once this option is selected, the following four lines of code should be written under the newly displayed text "option explicit":

```
Dim vTime1
Dim vTime2
Dim vTime3
Dim vTime4
```

The "option explicit" statement allows the computer to recognize the variables that will be used throughout the form. The "dim" statement is short for "declare." Thus, for each variable created, the user must have a corresponding "dim" statement followed by the variable name under the "(general)" section of the program.

Once the variables are declared, the user can close the "Project1-Form1 (code)" window and the form will return. The program is now ready for testing. To test, the user should click on the "run" option at the top of the screen and then on the "start debug" option. The program will be transferred to the Pocket PC (or the emulator software will start) and the program can be tested.

Data files. For the time-based data to be written to a data file for later analyses, another object and additional code must be added to the form. First, a "file" must be added for data storage. However, this object is not automatically placed in the toolbox when the program is initially opened. To add the file option to the toolbox, the user must right click the mouse on the vertical toolbox on the left of the screen. Next, the user will click on "components," select "Microsoft CE File System Control," and click "OK." This causes two new objects to appear in the toolbox. One looks like a file, and the other looks like a file cabinet. The left object is the file, and should be added to the form. The process creates a rectangular box on the form entitled "file."

To open the file automatically when the program begins, code must be written in the Project1-Form1(code) window. To access this window, the user must double-click on the form instead of one of the command buttons. The blinking cursor will appear,

and the following line of code will be displayed:

```
Private Sub Form_OKClick( ).
```

The user must move the cursor to the bottom of this section, to the line after "end sub." Next, the user must move the mouse to the top of the window, click on the downward arrow to the right of the text "OKClick," and select the "activate" option. This will add several new lines to the code. They are:

```
Private Sub Form_Activate( )
End Sub
```

Between these two lines of text, the following line of additional code should be written:

```
File1.Open "data.txt", fsModeAppend
```

This line of code allows the program to open a file (arbitrarily named "data") that is a text file (.txt) and to write the data to this file in an append fashion. Append means that the newly collected data will appear in the data file directly after the previous data. This storage method allows a running record of all sessions unless older files are deleted. From this text file record, the user can cut or copy the desired selection of data and paste it in a new file specific for a participant, date, or session. If the user attempts to locate this data file on the Pocket PC, it will be found directly in the "my documents" directory. However, it may require more searching to locate the file on a PC if the user is running the PDA emulator. To locate this file, the user should look in the following directory: C:\Windows CE Tools\wce300\MS Pocket PC\emulation\palm300 directory. If the eVT software has been placed in a different directory during installation, the user must search in that specific directory.

Once the file option is open and the code has been written, data are capable of being stored. For illustrative purposes, the dependent measures are response duration and response latency. Because the response duration will be collected when the user clicks on the "end" button and response latency will be collected the next time the "begin" button is clicked, the code for writing data to file will be written under the "begin" button. This code is added by clicking on the downward arrow to the right of the word "form" in the upper left corner of the window. The user should scroll down until the "Command1" option appears and should select this option. Once this option is chosen, the cursor will appear under the "Private Sub Command1_Click( )" text. After the timer code (vTime4) and before the line "end sub," the following text should be typed:

```
File1.LinePrint vTime2 & "," & vTime4
```

This line of code writes the value of vTime2 (the response duration) to a text file followed by a comma, and then followed by the value of vTime4 (the response latency). The comma is necessary because it separates these two values in the data (text) file and allows the data to be separated into individual rows when they are imported into graphical and statistical programs.

To close the file automatically when the session ends, the following line of code should be typed:

```
File1.Close
```

This code may be placed under a separate command button called "exit," or it could be added to the Private Sub Form_OK Click( ) subroutine immediately before the line of code "App.End."

Minimizing collection errors. Although the program described above is designed to record parameters of time accurately, it is not completely error proof. For example, a user might accidentally click on the "begin" button twice, resulting in inaccurate time values. Another problem may be that the computer may write the data to file twice, dis-

rupting eventual data analysis. These problems can be eliminated easily.

First, the observer can be prompted as to which text box corresponds to which type of response data by adding a "label" object. To add a label, the user should select the second object on the right side of the toolbox labeled "label," which resembles a capital letter A. The label is added to the form using the same procedures described above (for adding command buttons and text boxes) and should be placed on the form immediately above "Text1." To assist the observer, the "caption" of the label may be changed to "duration" using the properties window. Following this process, another label (for latency) can be added directly above "Text2."

In addition to adding labels, the program can be written such that the "begin" button becomes temporarily inoperable once it has been clicked and can only be turned on again after clicking the "end" button. Likewise, the "end" button can be programmed to become temporarily inoperable. The process for writing this code is as follows. First, the user should double-click on the "begin" button and enter the following line of code below Private Sub Command1_Click( ):

```
Command1.Enabled = False
Command2.Enabled = True
```

This code will turn off or disable the Command1 button (the "begin" button) when it is clicked on and will enable the Command2 ("end") button. To reactivate the "begin" button and turn off the "end" button after it has been clicked, the following line of code must be added directly below Private Sub Command2_Click( ):

```
Command1.Enabled = True
Command2.Enabled = False
```

Finally, the user must click once on the "end" button, move to the Properties-Command2 window, and select the "enabled" property. This property must be changed from "true" to "false" and will result in the "end" button being disabled upon the initial start-up of the program. Figure 3 displays the resulting graphic interface, inputted code, and properties window.

Saving and transferring work to the Pocket PC. At this point, a basic time-based data-collection program has been developed. To save the collected data, the user must click on "file," click on "save project," and enter the directory and name that the project will be saved as. Afterwards, the user should again click on "file," click on "save form," and then enter the directory and name that the form should be saved as. The form can be saved under any name.

To transfer the completed data-collection program from the desktop computer to the Pocket PC, the user must again click on "file" and then on "Make Project1.vb." A window appears that prompts the user to select a project name and file directory to store the project in. Once the user makes these selections and clicks on "OK," the program is ready to be transferred to the Pocket PC. To transfer the program, the communication software that came with the Pocket PC should be opened, the "explore" or "navigate" feature should be selected, and the user will need to find the newly created project or program, copy it, change directories back to the Pocket PC, and paste the file into the Pocket PC's directory. This process creates a version of the program on the portable device that will be ready for data collection. If the user has tested the program along the way using the PDA emulator and now attempts to run it from the actual PDA, an error message may occur. This is because additional eVB support files (in addition to the above-created project) must be placed on the PDA prior to running. The easiest way to transfer these support files is for eVB to do it automatically. The user should test the program in eVB at least once through the

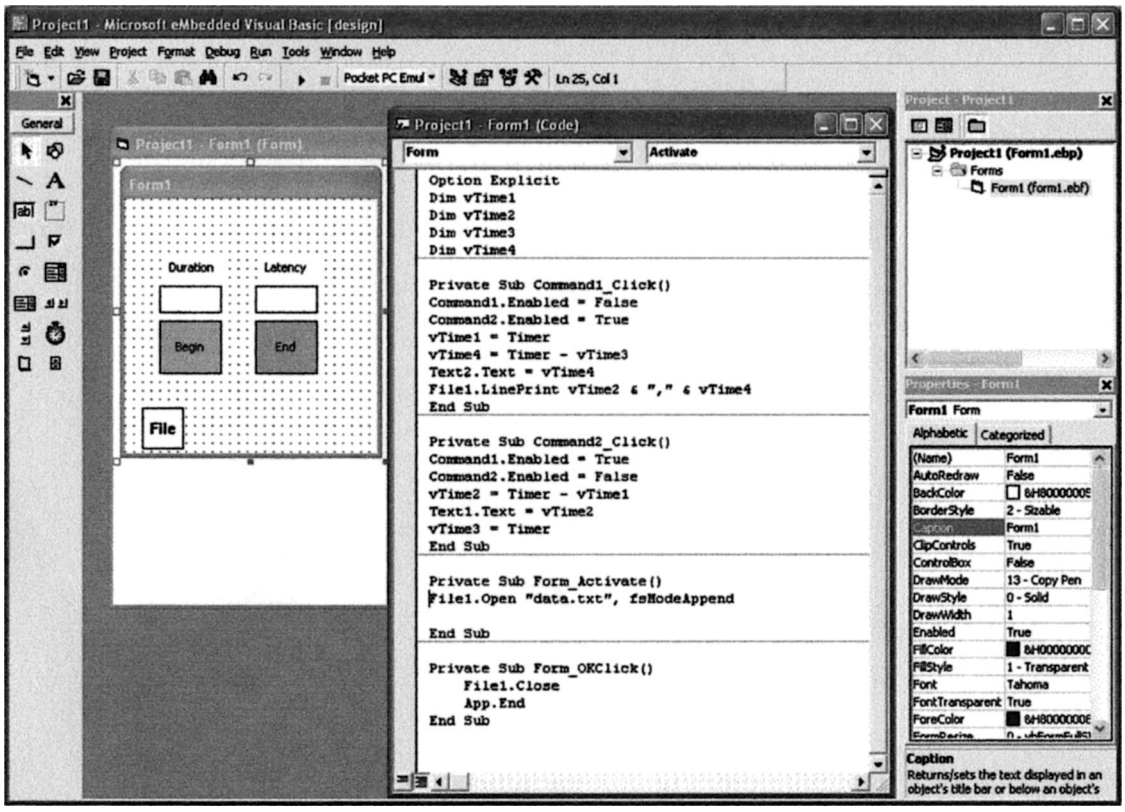

Figure 3. Final graphics and programming code for a time-based data-collection system.

PDA instead of the emulator. Afterwards, the finished project will run correctly.

Saving Time, Customizing, Troubleshooting, and a Program

As with other Microsoft products, eVT allows the user to use many editing shortcuts similar to those found in other Microsoft programs. For example, the user can copy and paste code using the CTRL+C and CTRL+V key combinations, drag and drop images, or use CTRL+S to save work. There are also many customization features for fonts, line type, text box borders, and so forth that are located in the "properties" window of eVT and that can be easily manipulated.

Although the above example illustrates how to create a data-collection system, programming a computer is often difficult. Many types of bugs or glitches in the process might occur. Table 1 displays a summary of potential problems and possible solutions that might occur when using the eVT system.

Downloading Data for Further Analysis

To retrieve the data that have been collected, the file must first be located on the Pocket PC. It will be entitled whatever it was named in the "File1.Open" code (File1.Open "data.txt," fsModeAppend in the earlier example). In this example, the user would locate the file named "data," which will most likely be placed in the "my device" directory of the Pocket PC. Although the text file can be previewed through Microsoft Word on the Pocket PC, copying the file to the desktop computer will allow the user to have more flexibility for

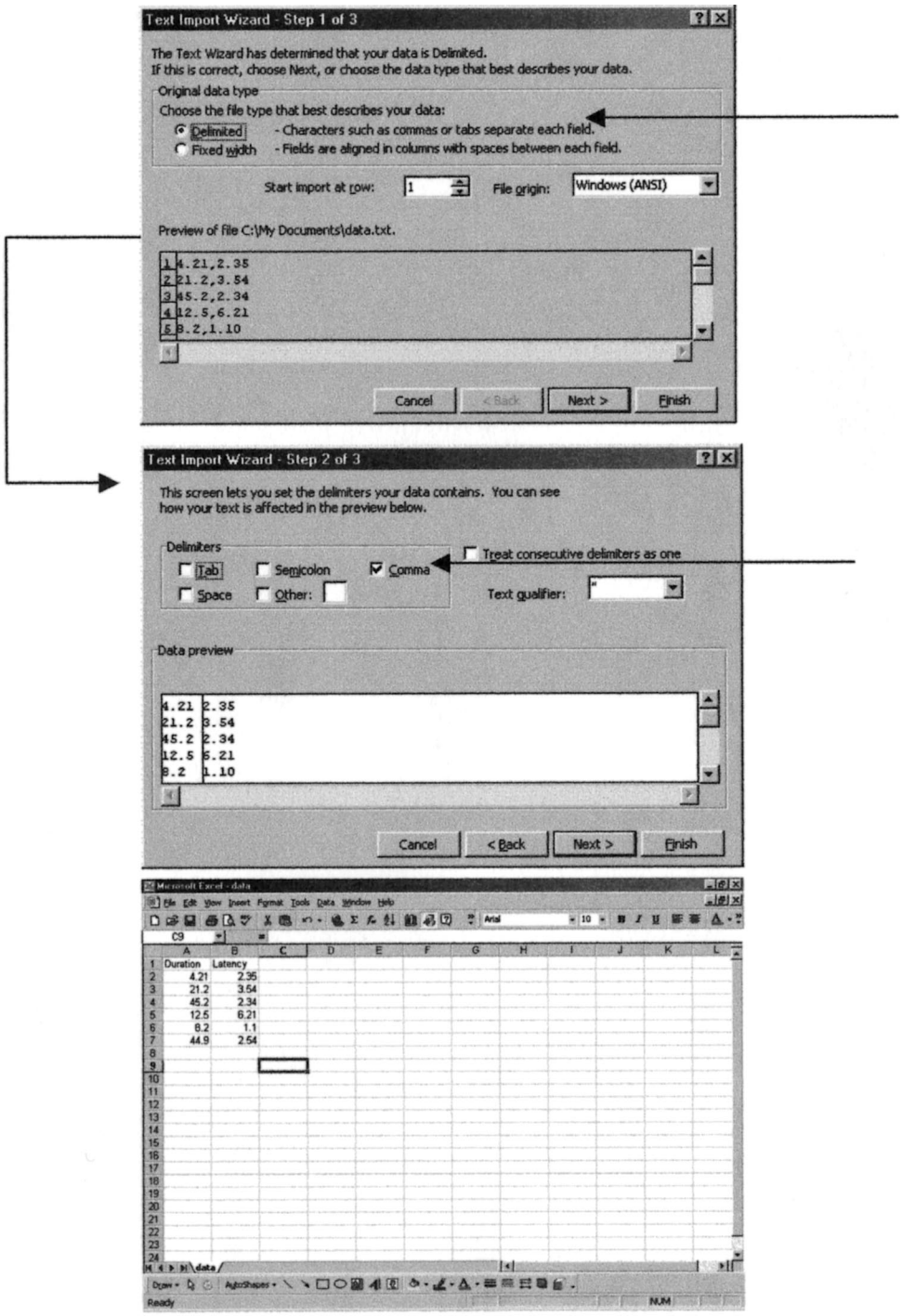

Figure 4. Screen shots of data-importing windows and the resulting spreadsheet in Microsoft Excel.

data analysis. To paste the file, the user must open the communication software on the desktop, click on the options necessary to locate the Pocket PC's documents, select the data file, and copy and paste it in a directory on the desktop computer.

Many data analyses and graphic generations can be accomplished using spreadsheets. Microsoft Excel is one such spreadsheet that has been previously noted by Carr and Burkholder (1998) as useful for behavior analysts. To import the data file into Microsoft Excel, the user must first open the Excel program. Next, the user should click on "file," then on "open," and then change the option "File of type: all Microsoft Excel files" to "Files of type: all files." It may be necessary to select the directory the data file is in and then select the data file. Figure 4 displays the Excel screens for data importing.

A window entitled "Text Import Wizard" will appear. Here the user can select the line of text in the data file that will be imported, along with the type of importing method (delimited or fixed width). The user should first select the desired first line of data, then check the "delimited" option, and then click "next." The next window will prompt the user to select the type of delimiter that will separate the data into individual cells in the Excel worksheet. The "comma" option should be selected (while deselecting the "tab" option), because a comma was used to separate the data in the written code (see above description of writing code for text files). Finally, the user should click on "finish." The data will then appear within the Excel worksheet for analyses and graphing (see Carr & Burkholder, 1998, for a guide to graphing single-subject data using Excel).

If the data require more sophisticated statistical procedures than a spreadsheet can compute, they may also be imported into various statistical programs (e.g., SPSS, SAS). The import method will be similar to that for Excel, whereby the user imports a text file that has comma delimiters. (Refer to the statistical package owner's manual for specific importing procedures of text files.)

SUMMARY

Data collection is a critical component for developing effective treatment strategies and demonstrating experimental control (Sulzer-Azaroff & Mayer, 1991). The use of computerized system that is customized to the user's needs may aid in the attainment of these goals. The programming routine codes that are presented in the current article are only a fraction of what is possible with eVB. However, these routines lay the foundation for what can be accomplished. Each user can further integrate the code in a meaningful way for his or her purposes.

The Pocket PC provides a portable and powerful platform for recording computerized data, and the Microsoft Embedded Visual Basic programming language provides an interface for designing the data-collection system. Although the eVT software is available free by download, potential users must purchase a PDA before using the software. PDAs differ in price, with the average cost of a PDA to run eVT being around $200. However, the combined costs of the software and the PDA are much less than several of the programs described by Kahng and Iwata (1998). Together, the information and the methods introduced in the current article provide a basis for designing, customizing, and using individualized handheld data-collection systems.

REFERENCES

Bellack, A. S., & Hersen, M. (1998). *Behavioral assessment* (4th ed.). New York: Allyn & Bacon.

Carr, J. E., & Burkholder, E. O. (1998). Creating single-subject design graphs with Microsoft Excel™. *Journal of Applied Behavior Analysis, 31,* 245–251.

Dixon, M. R., & MacLin, O. H. (2003). *Visual Basic*

for behavioral psychologists. Reno, NV: Context Press.

Emerson, E., Reeves, D. J., & Felce, D. (2000). Palmtop computer technologies for behavioral observation research. In T. Thompson, D. Felce, & F. J. Symons (Eds.), *Behavioral observation: Technology and applications in developmental disabilities* (pp. 47–60). Baltimore: Brookes.

Kahng, S., & Iwata, B. A. (1998). Computerized systems for collecting real-time observational data. *Journal of Applied Behavior Analysis, 31,* 253–261.

Miltenberger, R. G., Rapp, J. T., & Long, E. S. (1999). A low-tech method for conducting real-time recording. *Journal of Applied Behavior Analysis, 32,* 119–120.

Sulzer-Azaroff, B., & Mayer, G. R. (1991). *Behavior analysis for lasting change.* New York: Harcourt Brace Jovanovich.

Received January 18, 2002
Final acceptance February 19, 2003
Action Editor, Wayne Fisher