HAND HELD® PRODUCTS
a Welch Allyn® affiliate

TT Advantage

Programmer's Guide

Disclaimer

Table of Contents

Introduction

TTAdvantage is a Visual Basic add-in tool that is distributed on the Transaction Team (TT) 3100 Series Software Suite CD. Add-ins are time and labor saving automation tools for the Visual Basic programming environment. Add-ins ease development time by customizing and extending the Visual Basic environment.

TTAdvantage provides two important benefits to developers building interactive point-of-sale applications for Hand Held Products TT3100 Series terminals. It speeds programming and eases the task of integrating TT3100 series terminals into existing software applications.

Functions

Based on the popular Visual Basic programming environment, TT Advantage provides simple tools to integrate command buttons, signature capture boxes, graphics, MSR data capture, text boxes, list boxes and check boxes onto Hand Held Products TT3100 Series screens. TT Advantage enables developers to quickly create, modify and add new screens and capabilities to TT3100 Series terminals.

Benefits

Designing the appearance of a TT3100 Series screen is as simple as using a drawing program. The graphical nature of the TT Advantage programming environment saves development time. Elements such as boxes, buttons, pictures and lines can be created easily with Visual Basic controls to compose ads and graphics. Controls added to Visual Basic by TT Advantage enable developers to create applications that capture signatures, read credit cards, and display forms for user input. With the ease of creating and programming display screens using TT Advantage, developers can build and update their systems in a timely manner.

Developers writing in C, C++ and other languages benefit especially when using TT Advantage to integrate TT3100 Series terminals with existing applications. With TT Advantage, developers can create applications called “scripts” that reside and run on the TT3100 Series terminals. Existing C and C++ programs trigger interactive point of sale resident applications. When the interactive point of sale application runs to completion, it can return information such as signature or magnetic stripe values to the triggering application. TT Advantage allows creation of TT3100 Series interactive point of sale resident scripts that minimize disruption to existing host code.

Modes

TT Advantage offers two modes of programming: **Script Mode** and **VB Mode**. Both modes rely on the Visual Basic Integrated Development Environment for code generation.

Script Mode

Programmers typically employ Script mode when TT3100 Series terminals and programs must integrate into existing programs written in C, C++ or other languages. One common example of that is retail point of sale, where new interactive applications must integrate with proprietary retail sales applications and dated equipment. Script mode creates new applications that reside and execute on the TT3100 Series terminals and are triggered by existing retail applications.

VB Mode

Programmers typically employ VB mode for building stand-alone Visual Basic applications that control the TT3100 Series terminal or integrating with existing Visual Basic applications that reside and execute on a host such as a Windows 95/98/NT PC. A common Visual Basic stand-alone application is security sign-in.

The flexibility inherent in TT Advantage allows developers to place application logic where it's most appropriate, either on a host or on the TT3100 Series terminal. Such flexibility allows Hand Held Products transaction applications to be created for practically any hardware and software configuration.

Why use TT Advantage?

Use TT Advantage to efficiently write applications for TT3100 Series terminals. The most important uses for TT Advantage are the following:

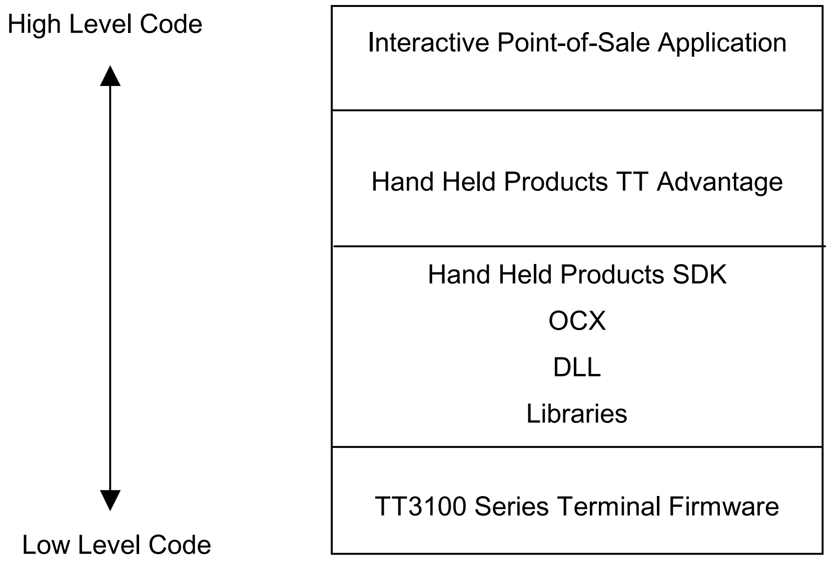
- As a tool for creating interactive point of sale resident scripts running on TT3100 Series terminals
- As a tool for integrating TT3100 Series terminals and transaction software into existing systems.
- As a tool to quickly modify projects and produce stable code for Hand Held Products transaction applications
- As a tool to quickly create interactive point of sale proof of concept demonstrations

Other Hand Held Products Development Software

Hand Held Products Software Development Kit (SDK) is included with TT Advantage on the TT3100 Series Software Suite CD. The SDK contains other development components such as static libraries and ActiveX components that interface with the TT3100 Series terminal.

For further information on the components that make up the SDK, see the SDK Roadmap document which is available upon installation of the SDK.

TT Advantage is a shell that insulates developers from lower level coding that is required if using the Hand Held Products SDK directly. The following diagram shows the relationship between TT Advantage and the SDK.


High Level Code
Low Level Code
Interactive Point-of-Sale Application
Hand Held Products TT Advantage
Hand Held Products SDK
OCX
DLL
Libraries
TT3100 Series Terminal Firmware

Developers of stand-alone Visual Basic PC applications may only need TT Advantage and not the SDK. Many developers use both the SDK and TT Advantage. The SDK supplies developers with special software components and libraries while TT Advantage provides an environment for rapid application development.

TT Advantage Users

The intended users are application developers and integrators building systems that contain TT3100 Series terminals, or extending systems to include TT3100 Series terminals. Users of TT Advantage need to be familiar with Visual Basic.

System Requirements

The system requirements for TT Advantage are the following:

- Windows PC with 100 MHz or faster Pentium microprocessor.
- 16 MB of RAM (32 MB recommended) and 10 MB free hard disk space
- TT3100 Series terminal
- Visual Basic 5.0 with Service Pak 3

Installation of TT Advantage

TT Advantage is a component on the Transaction Team 3100 Series Software Suite CD. The installation program is a simple to use InstallShield® application. Please refer to the Startup Guide you received with your hardware for instructions on software installation.

Installed Files

The files installed on the developmental PC are listed in the following table:

Type	Item	Description	Location
Application	mxVisual.dll		Program Directory
DLLs	mxSptool.dll	Downloads scripts and supports graphics	System Directory
	mxScript.dll	Script commands	System Directory
	mxVAhost.dll	Host required DLL for host mode	System Directory
OCX	Sigbox.ocx	Host mode connectivity	System Directory
	VisualControls.ocx	MSR + Sig Controls	System Directory
Text Files	scriptdescriber.txt	Definitions for script commands	Program Directory
VB Mode Templates	MDIMain.frm (read only)	MDI form that holds all screens	Program Directory\Templates\TT Advantage\VB
	TT AdvantageHost.vbp	Project files template	Program Directory\Templates\ TT Advantage \VB
	frmSplash.frm	Application specific splash screen	Program Directory\Templates\ TT Advantage \VB
	padStart.frm	Initial Screen	Program Directory\Templates\ TT Advantage \VB
	VAevents.cls	User can code these TT Advantage generated events	Program Directory\Templates\ TT Advantage \VB
	modMain.bas	Main code module that connects classes	Program Directory\Templates\ TT Advantage \VB
SCRIPT Mode Templates	TT Advantage Script.vbp	Template project for Script mode	Program Directory\Templates\ TT Advantage \SCRIPT
	MainScript.bas	Creates the main TOOL class and user code area	Program Directory\Templates\ TT Advantage \SCRIPT
	padStart.frm	First screen	Program Directory\Templates\ TT Advantage \SCRIPT
FORM Template	TT Advantage.frm	Form used in both modes	Program Directory\Templates\FORMS
	Vapinpad.frm	Form used to obtain PIN data in script mode	Program Directory\Templates\Forms
Output Files	script.bin	Temp BIN file for Script loading, Script file	...\TEMP or C:\
Admin	Admin.exe	This file does the following:	Program Directory\Templates

Type	Item	Description	Location
Program		Creates the TT Advantage Assistant entry in the VB Addin.ini file. Copies the TT Advantage Templates to the VB Templates directory. This program can be run at any time.	

Note: All template files copy automatically to the VB program template directory.

Developer Environment

TT Advantage leverages the powerful Visual Basic facilities to provide a programming environment for generating applications that incorporate TT3100 Series terminals.

Note: Familiarity with Visual Basic is recommended before attempting to use TT Advantage.

All Visual Basic applications, including TT Advantage applications, take the form of projects. A project is the master file for an application. It can contain three other files: a file containing forms, a file containing modules and a file containing classes. A TT Advantage project can be started in one of two modes: VB or Script.

Starting TT Advantage Projects

TO START BUILDING A NEW TT ADVANTAGE PROJECT:

1. In Visual Basic, open a blank project.
2. From the Add-Ins menu, select the Add-in Manager, double click on the Hand Held Products TT Advantage Assistant then click OK. The TT Advantage Assistant appears.
3. Select one of the two TT Advantage modes available from TT Advantage Message box: Visual Basic EXE Mode or Script Mode. Double-click the forms folder in the Project window. The Forms folder opens.
4. Double-click padStart form. The padStart form appears in the Designer window. All projects begin with the padStart form.

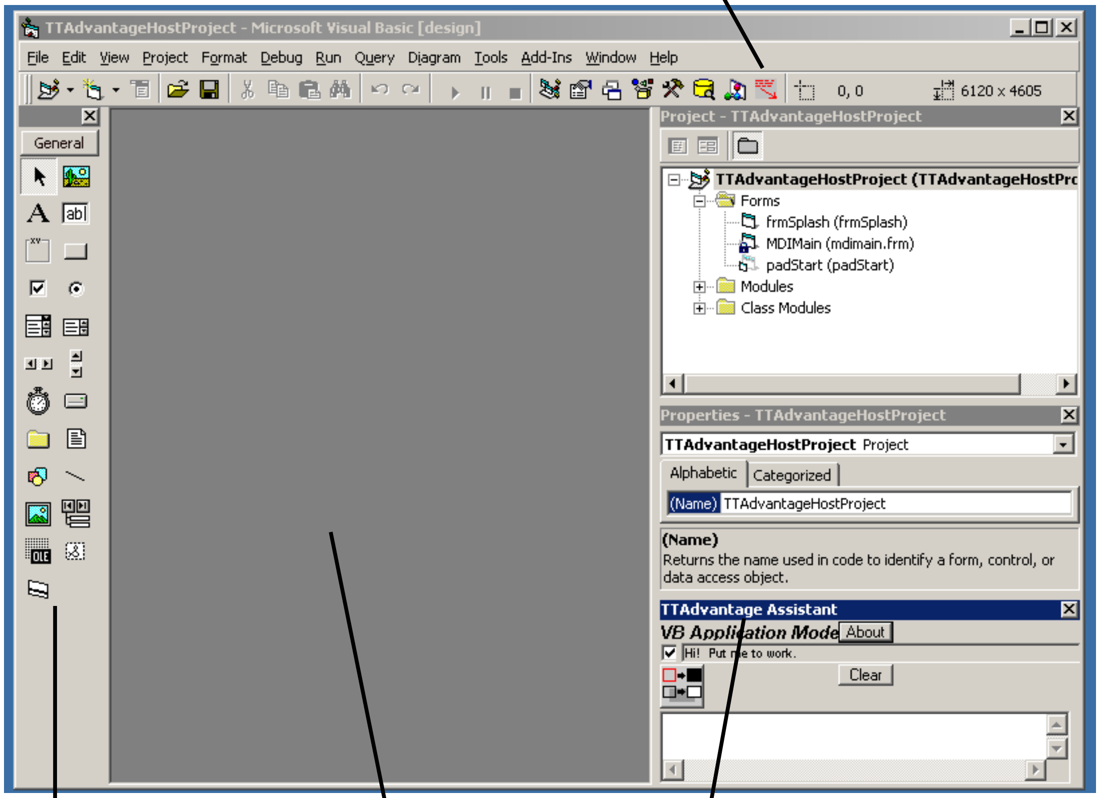
TO CONTINUE AN EXISTING TT ADVANTAGE PROJECT:

1. In Visual Basic click on the File menu and select Open. The Open message box appears.
2. Select the file from the list and click OK. The file opens.
3. From the Add-Ins menu, double click on the Hand Held Products TT Advantage Assistant. The TT Advantage Assistant appears.

Note: It is not necessary to activate the TT Advantage Assistant when opening an existing VB mode project.

The TT Advantage Project Window

Assistant Display button. If the TT Advantage Assistant tool window has been closed, open it by clicking on the red button


TTAdvantageHostProject - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
0, 0
6120 x 4605
General
Project - TTAdvantageHostProject
TTAdvantageHostProject (TTAdvantageHostPro
Forms
frmSplash (frmSplash)
MDIMain (mdimain.frm)
padStart (padStart)
Modules
Class Modules
Properties - TTAdvantageHostProject
TTAdvantageHostProject Project
Alphabetic
Categorized
(Name) TTAdvantageHostProject
TTAdvantage Assistant
VB Application Mode
About
Hi! Put me to work.
Clear

(Name)
Returns the name used in code to identify a form, control, or data access object.

Visual Basic Toolbox

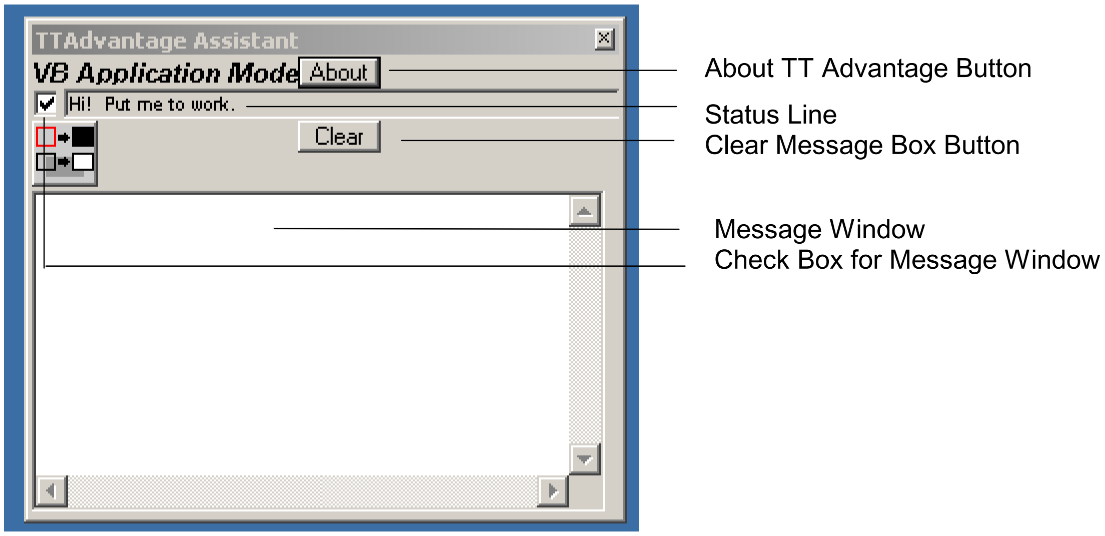
Form Designer

TT Advantage Assistant tool window is dockable and can be placed where convenient. When first activated, TT Advantage appears in the upper left of the form designer.

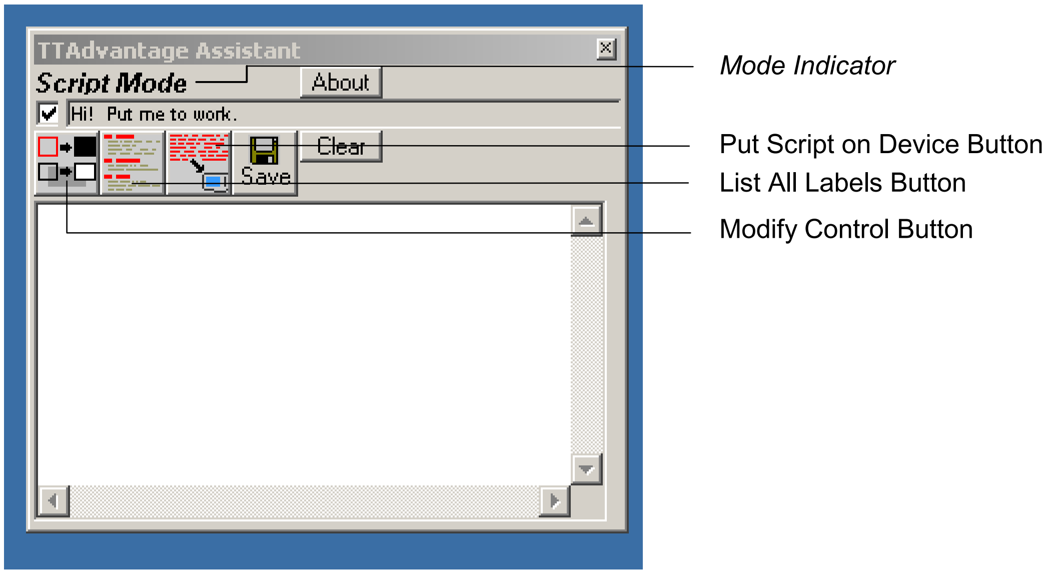
The TT Advantage Assistant

The TT Advantage Assistant takes two different forms depending on the operating mode as shown below.

VB MODE



SCRIPT MODE



The Controls on TT Advantage Assistant

The controls and indicators needed to manipulate TT Advantage Assistant capabilities are the following:

About Button

The About button works in both modes and when clicked, calls the About TT Advantage message box. The message box shows the version number of TT Advantage and presents a button that calls the Microsoft System Information message box.

Mode Indicator

The Mode Indicator shows the mode (either Script or VB) in which the TT Advantage Assistant is working. (See Programming Modes Overview, page 2-7.)

Status Line

The Status line shows process steps that TT Advantage is taking as they are occurring.

Check Box

The check box toggles the status line and error sound on and off.

Message Window

The Message window summarizes the results of processes, such as compilation errors or the successful loading of a script.

Clear Button

The button erases the Message Window and Status line.

List All Labels Button

The List All Labels Button only appears in the Script mode. It lists user-defined sub-routines and interactive point of sale screens in the Message Window.

Put Script on Device Button

The Put Script on Device Button only appears in the Script mode. This button loads compiled scripts onto the TT3100 Series terminal.

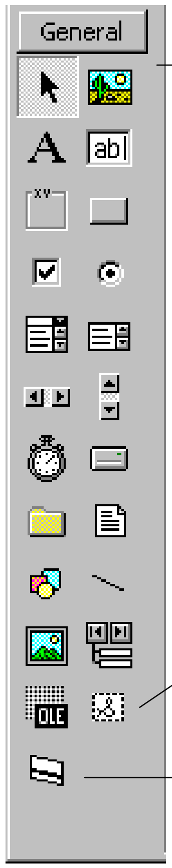
Modify Control Button

The Modify Control Button appears in both modes Script and VB mode. It changes the appearance of a limited number of controls from visible to transparent.

Controls for TT Advantage Forms

TT Advantage adds two special controls to the Toolbox: Signature Capture (SigControl) and Magnetic Stripe Reader (MSRControl).

Note: Only a selected subset of Visual Basic controls displays and works on TT3100 Series terminals. See Script Mode on page 3-1 and VB Mode on page 4-1.



Visual Basic Toolbox

Signature Capture Control (SigControl) allows developers to put a signature capture box on TT3100 Series terminals and handle signature information.

Magnetic Stripe Reader Control (MSRControl) activates magnetic stripe reader to capture credit or debit card information.

TT Advantage Forms

The special forms that TT Advantage adds to Visual Basic behave similarly to other Visual Basic forms but with some important differences. All TT Advantage projects start with a padStart form. Additional forms can be added as needed.

Adding TT Advantage Forms

Do the following to add a TT Advantage form.

1. Starting from an open TT Advantage project, click on the Project menu. The Project menu appears.
2. Select Add Forms from the Project menu. The Add Forms message box appears.

3. Double-click on the TT Advantage form in the Add Forms window. The TT Advantage form appears in the Designer Window.
4. Name the new form. All TT Advantage forms need to start with the word ***pad***, such as padSignature or padSwipe.

Pad Form Events

Each TT Advantage mode has an associated code window. The code window for the VB mode form contains four subroutines while the code window for Script mode form contains two subroutines. The subroutines are listed in the following table:

VB Mode Subroutines	Script Mode Subroutines
ScreenInitialize()	ScreenInitialize()
AfterScreenDraw()	AfterScreenDraw()
GoBack()	
GoNext()	

TT Advantage executes these routines automatically. TT Advantage executes any code added to the routines. For example: if code is placed inside the AfterScreenDraw routine, it executes after all the controls on the interactive point of sale screen are displayed. See the comments in the TT Advantage Form code window for detailed information.

Pinpad form

This PinPad form only supports Script mode. This screen is used to get the PIN from the user and saves the PIN in the PIN variable. The user does not need to define the PIN variable; TT Advantage adds it. This form code consists of three routines:

Script Mode Subroutines
ScreenInitialize()
cmdCancel_Click()
cmdEnter_Click()

TT Advantage executes these routines automatically. TT Advantage executes any code added to the routines. TT Advantage automatically adds the coding needed to save the PIN. As an example: if Tool.SCRIPT.GotoScreen padScreen command is added to the cmdEnter_Click sub, TT Advantage saves the PIN and goes to padScreen.

Note:

If the property Tag of the fldPIN text box is set to "dukpt", PinPad uses the dukpt encryption method. If the property is blank, PinPad uses Master Session encryption method.

TT Advantage Form Properties

The TT Advantage pad form generates an optimum display on TT3100 Series terminals. Most changes to properties of TT Advantage forms have no effect on the resulting screen display. Only five form properties modify the resulting screen. Of the five properties, only two should be changed during application development: the Caption property and the Name property. Do not change the other three properties: ScaleMode, WidthMode and HeightMode. Changes to these properties degrade screen display appearance and function.

Programming Modes Overview

Applications written for TT3100 Series terminals are developed with TT Advantage in one of two modes: Script or VB. These applications are event driven. TT Advantage creates event driven software for TT3100 Series terminals as scripts or Visual Basic executables. Both modes have unique advantages depending upon the TT3100 Series host and existing software.

Comparison of TT Advantage Modes

Intelligence built into TT3100 Series terminals causes TT Advantage derived applications to fall naturally into two categories: applications consisting of internal scripts that control TT3100 Series terminal functions, and applications in which external programs control TT3100 Series terminal functions. Both types of applications can be developed with TT Advantage. Applications residing on TT3100 Series terminals are written in TT Advantage Script mode. Applications residing on an external host are written in VB mode.

Deciding Which TT Advantage Mode to Use

Each of the two TT Advantage modes addresses an important class of applications for TT3100 Series terminals.

Script Mode

Developers integrating TT3100 Series terminals into legacy software and hardware systems should use the TT Advantage scripting mode. Sensitive systems such as financial transaction networks need to be thoroughly tested after they are modified. The less such code is disturbed, the more easily it can be readied for re-use. By inserting script activating “hooks” into legacy host software, interactive point of sale functionality can be added to a transaction system with ***light*** modifications to existing software. Light coding changes mean efficient upgrades of legacy transaction systems.

VB Mode

Developers integrating TT3100 Series terminals into systems that consist of a Windows Operating System and an application written in Visual Basic should use TT Advantage VB mode. VB mode can also be used as a tool to create proof of concept demonstrations.

Script Mode

The Script mode generates TT3100 Series-resident programs (scripts) that are activated from existing transaction software. Programmers creating new e-transaction systems, where the host controls all the interactive point of sale functions, should use the VB mode, while programmers upgrading legacy systems should use the Script mode.

What are Scripts?

Scripts are compiled programs written in the Visual Basic environment using the TT Advantage Assistant. With the customized controls and commands resident in the TT Advantage Assistant add-in, programmers construct semi-conventional Visual Basic programs that are compiled to machine code by a TT Advantage associated DLL.

Once compiled, developers use TT Advantage to download scripts into a TT3100 Series terminal attached to the COM port of a host. The TT3100 Series terminal holds the script in non-volatile memory. The speed and ease with which TT Advantage compiles and generates scripts allows developers to rapidly iterate and optimize TT3100 Series terminal applications.

Operating a Script

To activate a script, power down then power up the TT3100 Series terminal by disconnecting and reconnecting the power cable. The firmware resident operating system then runs the script. The operating system and the script work semi-autonomously, communicating and receiving commands through the RS-232 and RS 485 ports of the TT3100 Series terminal.

If a script contains a Sub Main(), the program starts execution with Sub Main; otherwise the program starts execution with padStart form.

Screen Creation Process in Script Mode

Screens on the TT3100 Series terminals are created control-by-control. Although it is not necessary to understand how interactive points of sale screens are created, knowledge about the process helps with troubleshooting. For example, when using the command "Tool.Display.Screen", the following occurs in the listed order.

Steps	Explanation
Clear Screen	Freezes the operation of interactive point of sale screen and deletes any objects on the screen
Process ScreenIniti-zation event on forms	Executes "ScreenInitilization" code in a form.
Draw screen	Places objects on the screen in the following order: 1. Signature areas 2. Magnetic Stripe Readers 3. TextBoxes 4. All other controls in reverse order of creation except for Command Buttons. 5. Command Buttons
Process After-ScreenDraw event on form	Executes "AfterScreenDraw" code in a form
Process timer routine if part of code	Note: Use only one timer per form. The timer code only executes once
Wait for user input	Script stops to wait for user or host initiated event.

Note: The TT3100 Series terminals ignore all host commands and events while in steps 1 - 4.

Script Mode Software Components

Script mode uses certain components in projects. The TT Advantage Assistant automatically adds those components to projects, so this information is presented for detailed understanding of TT Advantage Script mode.

The following components are needed to operate TT Advantage in the Script mode:

Controls

Name	Description	Additional Notes
VisualControls.ocx	Provides the SigArea and MSRArea controls	DO NOT MODIFY

Forms

Name	Description	Additional Notes
padStart.frm	TT Advantage defaults to padStart on starting in script mode unless submain() is present in mainScript. The first screen must always be padStart.	Modifiable component. To add more forms to the project, see TT Advantage Forms on page 2-5.

Modules

Name	Description	Additional Notes
mainScript.bas	Module contains: TT Advantage internal function routines User routines A user modifiable command for com port selection and baud rate A user modifiable command for specifying command pre-fixes	In a script, all execution stops at the end of a routine. To continue execution past the end of a routine, add a GoToRoutine command at the end of the code. Script mode in TT Advantage does not have subroutines. A routine normally halts execution at the end of code. Communication port is selected with a line in MainScript.bas Command prefixes are also defined in MainScript.bas

References

Name	Description	Additional Notes
mxSptool.dll	Enables the downloading of scripts and graphics to the TT3100 Series terminal.	DO NOT MODIFY
mxScript.dll	Enables the “autolist” in Visual Basic that helps displays the TT Advantage commands.	DO NOT MODIFY

Downloading the Designed Script to the TT3100 Series Terminal

Once the port number is entered, be certain the TT3100 Series terminal is attached to that port then press the “Put Script on Device” button (SeeThe TT Advantage Assistant on page 2-3.) The status area indicates the progress of the download and the TT3100 Series terminal display reads “Downloading.” The Status indicator signifies when the download is completed.

Operational Host Communication with Scripts

For the host to communicate with a Script on a TT3100 Series terminal, the mainScript.bas component of TT Advantage needs to be configured to use the port attached to the TT3100 Series terminal. The example shows where to change the port in the **mainScript.bas** code.

```
'-----------------------------------------
' Start of required code for scripting
'  *** THIS MUST BE THE FIRST DIM or DECLARE IN THIS
MODULE!!!!!
Public Tool As New clsTool
'      ^
'      You can change this Command Indicator name.
'
Const PortNumber = 1
'                  ^
'                  Change this to the port number
where
'                  the POS device is connected
'
Const BaudRate = 57600
'                   ^
'                   Change this to the baud rate
which the
'                   POS device to be communicated
'
' End of required code for scripting
'-----------------------------------------
```

Host to Script Communication

With the script on the TT3100 Series terminal, power up the terminal. Once it is started, the script functions independently, however, a host terminal is required to send and receive information.

Communications between TT3100 Series terminals and hosts takes place using three commands, GoToLabel, SetTextVar, and GetTextVar.

GoToLabel LabelName

Executes the code starting at the label parameter. **LabelName** can be a routine name.

SetTextVar VarName, StringValue

Allows the host to set a string variable. For example, the host might set the variable “Data: to “12/01/1999”, to display a date on an attached TT3100 Series terminal.

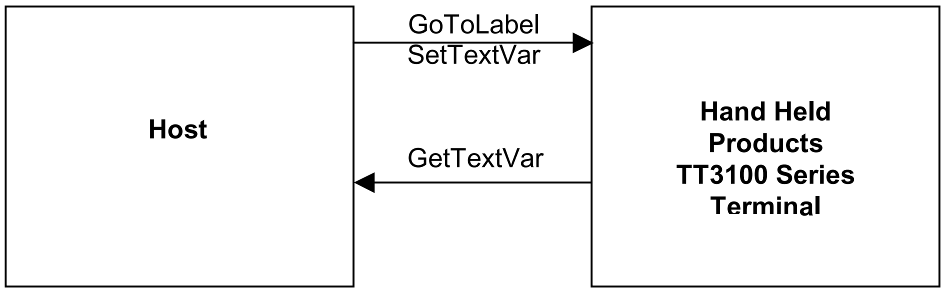
Note: All variables are case sensitive.

GetTextVar (VarName) as String

Allows the host to retrieve a string value from the TT3100 Series terminal for use in a host application. If script variable, “CredCardNum”, holds credit card data from an MSR read event, the operational host can request the variable information from the script by issuing a “GetTextVar” command to the script.

Any text (string) variable in a script can be read or be written to by using SetTextVar or GetTextVar commands. These commands are located on the mxsptool.dll and used by the host machines. See the Sample Host application installed in the program directory/Samples directory.

Host-TT3100 Series Terminal Command Traffic

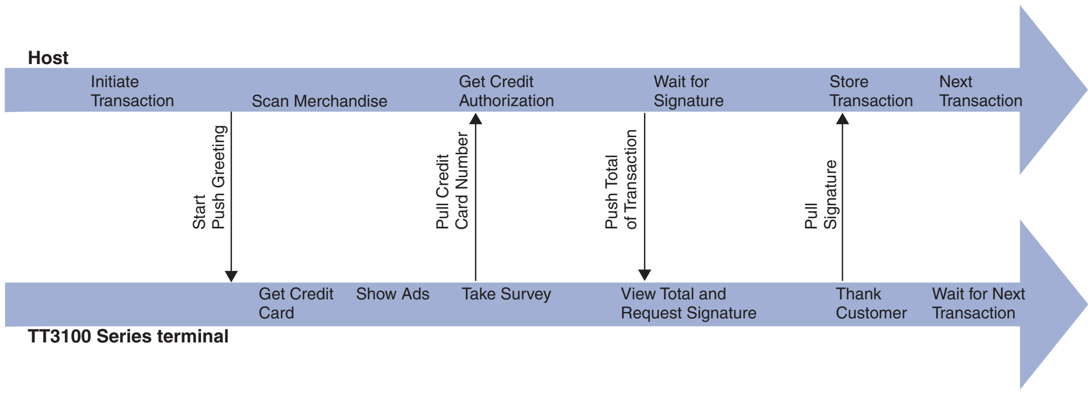


On start-up, the TT3100 Series terminal runs the loaded script. The Script can be designed to operate independent from the host. Communication between the host and the TT3100 Series terminal for a transaction can be as simple as:

1. Start transaction (GoToLabel)
2. Get credit card information (GetTextVar)
3. Set and display the purchase price (SetTextVar)
4. Get Signature (GetTextVar).

Transaction Events

Communication between the host and TT3100 Series terminals are simple in script mode. A light communications demand by TT3100 Series terminals of the host relieves the host for other tasks.



Script Mode Controls

TT Advantage scripts run in an environment unique to TT3100 Series terminals. The special demands of the TT3100 Series terminal environment requires a different set of controls than the Windows environment that is native for Visual Basic. Thus, while many of the controls function in TT Advantage as they do in Visual Basic, some do not.

Control Font Sizes

Font characters displayed on TT3100 Series terminals come in six sizes. All characters in each font size have the same pixel dimensions. The TT Advantage font families are mapped onto the Courier New font families in the property boxes of TT Advantage controls. The following table shows the correspondence between the TT Advantage font sizes and Courier New font sizes.

TT3100 Series Terminal Font Sizes in Pixels	**Courier New Font Sizes in Points**
6 x 8	10 to 11 pt
8 x 8	12 pt
8 x 12	13 to 16 pt
12 x 16	17 to 22 pt
16 x 16	23 to 26 pt
16 x 24	27 to 30 pt

COMMAND BUTTON

TT Advantage command buttons operate the same as Visual Basic command buttons.

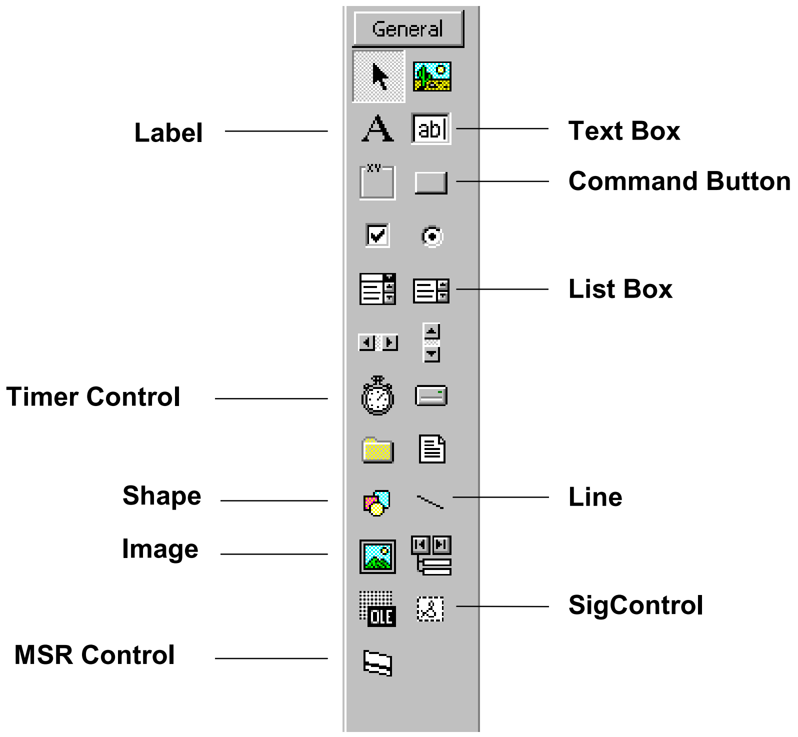
Property	**How property is used**
Font-Size	Controls the size on the font displayed on the TT3100 Series terminal
Caption	The text displayed inside the button
Style	Property can be set to normal or hidden

USING THE CONTROL

Use TT Advantage buttons in the same way as Visual Basic buttons.

TT Advantage Tool Box

The eight controls shown in the following diagram work in TT Advantage Script mode. The MSRControl and SigControl are specialized controls added to the toolbox by TT Advantage. Descriptions of the controls follow below:



Image

The image sizes on a TT3100 Series terminal depend on the number of pixels they contain. The images display with the same number of pixels as the images on the host monitor. The size of images can be adjusted once placed on a TT Advantage form, but look better if adjusted in the creating application. Keep all the image edges within the boundaries of the screen forms for best results.

Property	**How property is used**
Properties are same as in Visual Basic	Images can not be larger than 320 x 240 pixels

Using the Control

All Script mode images must be 2-color, black-and-white bitmaps. Adjust the image colors with the picture property. Always use the picture property to add or change the picture; do NOT cut and paste pictures.

Label

Label controls in Script mode work like Visual Basic label controls, but DO NOT wrap. Multiple lines of text require multiple labels.

Property	How property is used
Font-Size	Controls the size of the font on the pad

USING THE CONTROL

After putting the code under the _Click event for a label, TT Advantage automatically creates a hidden button in front of the label and assigns the _Click event code to the hidden button.

Textbox

Textbox controls in script mode work like a label. Also it works like a variable for script mode. Text property act likes name of the variable. Multiple lines of text require multiple labels.

Property	How property is used
BorderStyle	"0 - None" displays no border. "1 - Fixed Single" displays a border. This is the default setting.
Font-Size	Controls the size of the font on the pad

Line

The line control displays a line on the TT3100 Series terminal just as it appears on the developer host screen.

ListBox

Script mode ListBoxes behave very differently than Visual Basic ListBoxes. Script mode ListBoxes operate as a "serial marquee." A line of text appears one word at a time in the ListBox window. Create Script mode ListBoxes are one text line in height. Any other height produces unpredictable results.

Property	How property is used
List	Place a line of text directly into this property so the box displays one line at a time.
Font-Size	Sets the size of the font on the pad

USING THE CONTROL

Enter the lines of text for display in the List property. It is important to manually adjust the height of the ListBox until only one line of text displays.

MSRControl

Place the MSRControl anywhere on the form to give the MSR capabilities to TT3100 Series terminals. Though the control is visible on the developer screen, it is invisible on the TT3100 Series terminal. When a form containing a MSR control activates, the MSR also activates. Once activated, the MSR on the TT3100 Series terminal can accept information from credit card swipes. If a good swipe occurs, the code under the **GoodSwipe** event is called.

EVENTS:

GoodSwipe - fires after a good swipe.

BadSwipe - fires after a bad swipe or after no swipe after timeout.

Property	How property is used
There are no configurable control properties	

USING THE CONTROL

Draw this control on the developer screen. The MSRControl is hidden on the TT3100 Series terminal.

SigControl

The SigControl creates a signature capture area on the TT3100 Series terminal.

Put code under the “SignTimer” event to handle timeouts.

Property	How property is used
BorderStyle	“0 - None” displays no border “1 - Fixed Single” displays a border. This is the default setting.
MaxSize	This property limits the number of points collected per signature. Once a user produces MaxSize pixels, the TT3100 Series terminal stops taking signature data.
Timeout	This property sets the number of seconds, after the user lifts the pen from the TT3100 Series terminal, until the “SignTimer” event fires.

USING THE CONTROL

Use this control to draw a signature area. To save the signature into a string variable, use the **Tool.Sig.SaveVar** command. To clear a signature area, use the **Tool.Sig.Clear** command.

Shape

The shape tool can only be used to draw rectangles on the device, so be sure the control is set to the rectangle shape. After placing a Shape on a form, a dialog box appears asking if the object should be a “Box (Filled)” or a “Frame (Transparent).”

Property	How property is used
FillStyle	“0 - Solid” fills the rectangular area with the current fill color. “1 - Transparent” draws a transparent rectangle.

Timer

Place the Timer control anywhere on a form to give timer capabilities to a screen. The Timer control is visible on the developer screen but hidden on the TT3100 Series terminal. In Script mode the timer control does not act like a normal Visual Basic timer. It acts like a “countdown” timer. It waits for the number of seconds specified in the “internal” property, then executes the code under the timer event. The countdown occurs only once. Displaying a second screen after 15 seconds is an example of this timer’s usefulness.

Property	How property is used
Interval	TT Advantage parameters for timer intervals are expressed in seconds. Parameters for timers in Visual Basic are expressed in milliseconds.

USING THE CONTROL

Enter the countdown time in the Interval box of the timer’s properties.

Add the control on the form, then enter the interval to be timed. Double-click the control; add script commands to be executed after the interval elapses. Since Script mode timers are countdown timers, enter the amount of time to be counted down in the Interval property. The timer waits for the number of seconds specified with the “Interval” property, then executes the code under the timer event. A great use for the timer is to display a sequence of screens.

Script Commands

TT Advantage code lines use either ***statement*** or ***function*** syntax. An example of statement syntax is:

```
Tool.Sound.Bell Alarm
```

while an example of function syntax is:

```
Tool.Sound.Bell(Alarm).
```

Use function syntax with parentheses for sub commands such as a **Script.IfTrue** command.

Command Prefixes

Every Script mode command begins with a prefix. The default prefix is **Tool**. The Script command prefix can be changed by substituting a new word in for **Tool** in the class declaration statement in the mainScript.bas file at the indicated location.

```
'-----------------------------------------
' Start of required code for scripting
'  *** THIS MUST BE THE FIRST DIM or DECLARE IN THIS MODULE!!!!!
Public Tool As New clsTool
'      ^
'      You can change this Command Indicator name.
'
Const PortNumber = 1
'                  ^
'                  Change this to the port number where
'                  the POS device is connected
'
' End of required code for scripting
'-----------------------------------------


Sub main()
```

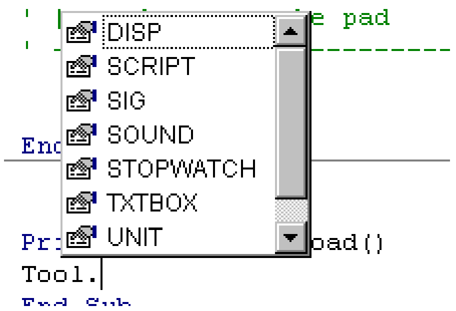
Declaring Variables

In Script mode, there are four types of variables: String and three types of numeric. Variables are created with the **Var** command and sub-commands to **SetVar**.

Script Command Generation

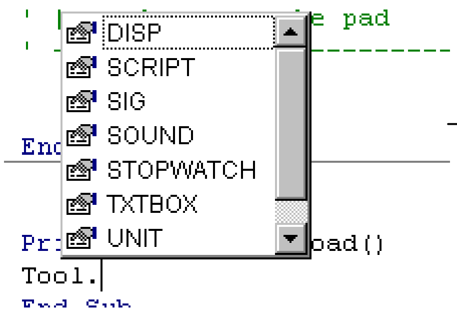
TT Advantage uses the Visual Basic interface to provide command options to developers as they construct Script commands. The illustration depicts a typical sequence:

1. Type Tool. A scroll box with available command options appears:

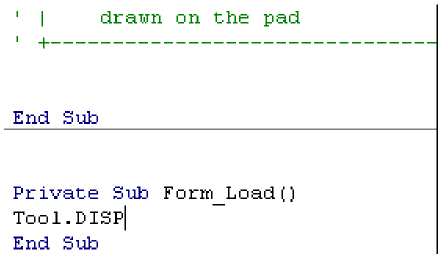


```
Tool.
```

2. Double click one of the command options. The selected command option appends to the end of the command under construction.



```
Tool.
```

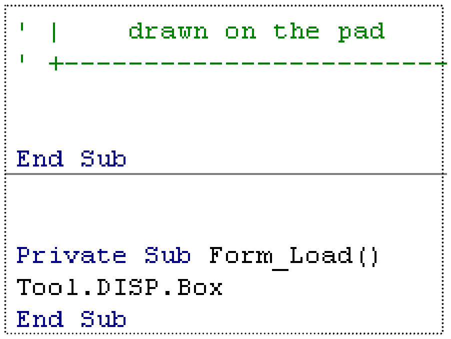


```
' |     drawn on the pad
' +------------------------------

End Sub

Private Sub Form_Load()
Tool.DISP
End Sub
```

3. Repeat process until typing a period brings up no new sub-options.

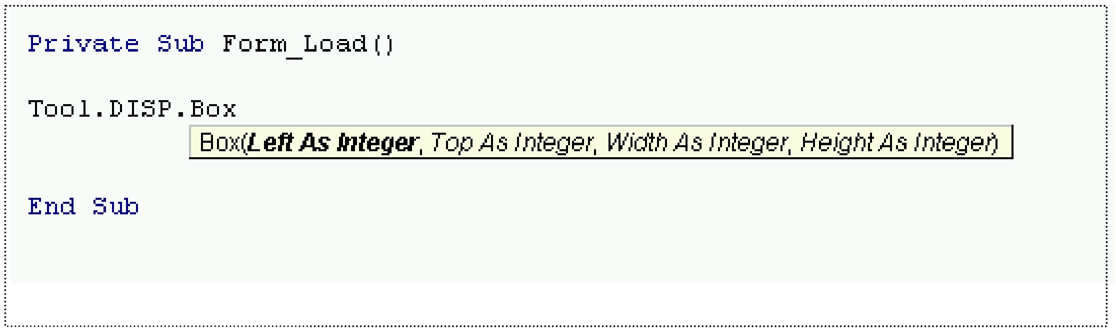


```
' |     drawn on the pad
' +-------------------------

End Sub

Private Sub Form_Load()
Tool.DISP.Box
End Sub
```

4. Type a space after the last sub-option to make parameters easier to see. A parameter list for the command, if one exists, appears with the first parameter highlighted.

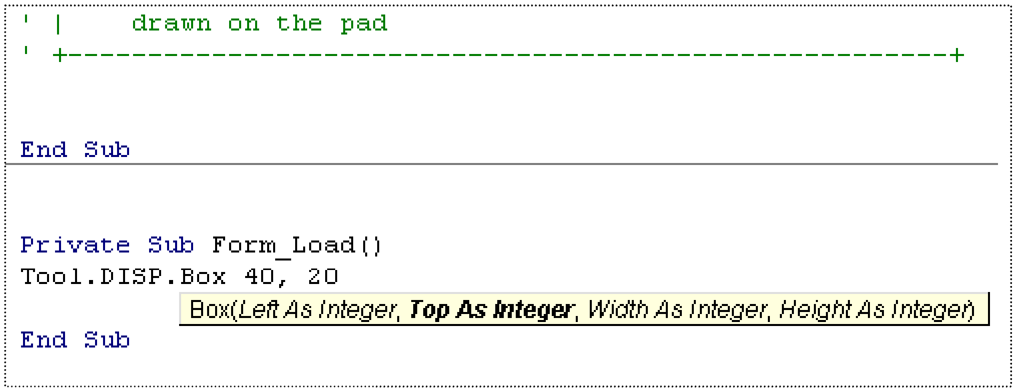


```
Private Sub Form_Load()

Tool.DISP.Box
         Box(Left As Integer, Top As Integer, Width As Integer, Height As Integer)

End Sub
```

5. Type in the parameters until the parameter list is exhausted.



```
' |     drawn on the pad
' +--------------------------------------------------------+


End Sub


Private Sub Form_Load()
Tool.DISP.Box 40, 20
          Box(Left As Integer, Top As Integer, Width As Integer, Height As Integer)
End Sub
```

TT Advantage Objects, Properties and Methods (Script Mode)

The organization of the objects, properties and methods for TT Advantage are displayed in the chart below:

TOOL

Disp

The DISP commands relate to the visual aspects of the interactive point of sale scripts. The DISP commands execute changes in the visual displays that occur during script operations.

Disp.Box(Left, Top, Width, Height)

Disp.Box draws a filled box on the screen.

Parameters	Description	Values
Left	Integer, distance from left edge of screen	Range: 0 – 319
Top	Integer, distance from top edge of screen	Range: 0 – 239
Width	Integer, width of box	Range: 0 – 319
Height	Integer, height of box	Range: 0 – 239

RETURN VALUES
None

SEE ALSO
Disp.Frame

EXAMPLE
```
Tool.Disp.Box 10, 10, 200, 140
```

Disp.Clear

Disp.Clear makes all objects on the screen invisible

Parameters	Description	Values
None		

RETURN VALUES
None

EXAMPLE
```
Tool.Disp.Clear
```

Disp.DeleteAllControls

Disp.DeleteControls removes all objects from the screen.

Parameters	Description	Values
None		

RETURN VALUES
None

Example
```
Tool.Disp.DeleteAllControls
```

Disp.DrawLine(X1, Y1, X2, Y2)

Disp.DrawLine draws a line on the screen.

Parameters	Description	Values
X1, Y1	Integers representing coordinates at the start of a line	X1: 0 to 319 Y1: 0 to 239
X2,Y2	Integers representing coordinates at the end of a line	X2: 0 to 319 Y2: 0 to 239

Return Values
None

Example
```
Tool.Disp.DrawLine 20, 20, 200, 20
```

Disp.Frame(Left, Top, Width, Height)

Disp.Frame draws unfilled rectangles on screen.

Parameters	Description	Values
Left	Integer, distance from left edge of screen	Range: 0 - 319
Top	Integer, distance from top edge of screen	Range: 0 - 239
Width	Integer, width of frame	Range: 0 - 319
Height	Integer, height of frame	Range: 0 - 239

Return Values
None

See Also
Disp.Box

Example
```
Tool.Disp.Frame 10,10,50,50
```
```
This will draw a small frame on top left hand corner of the
PadScreen
```

Disp.SetColor(Color)

Disp.SetColor sets the current color.

Parameters	Description	Values
Color	PadColors	White or Black

RETURN VALUES
None

EXAMPLE
```
Tool.Disp.SetColor Black
```

Disp.SetFont(FontType)

Disp.SetFont sets the current font.

Parameters	Description	Values
FontType		

RETURN VALUES
None

EXAMPLE
```
Tool.Disp.SetFont Font6x8
```

Disp.Text(Left, Top, Text)

Disp.Text displays text beginning at coordinates Left(X), Top(Y).

Parameters	Description	Values
Left	Integer, distance from left edge of screen	0 - 319
Top	Integer, distance from top edge of screen	0 - 239
Text	String	

RETURN VALUES

None

EXAMPLE
```
Tool.Disp.Text 20, 40, "Hello"
```

Disp.Getpin (Title, Account)

Disp.Getpin displays a DUKPT PIN entry prompt and returns a standard binary DUKPT PIN.

Parameters	Description	Values
Title	Optional, title message	String value to display as the title of the screen
Account	Required, account number as a string of ASCII digits.	

Return Values

This command encrypts a pin returned as a binary value. If an error occurs zero length result is returned. A canceled entry also returns a zero length result.

Example

```
Tool.Var.SetVar "PIN", Tool.DISP.GetPin("Enter Pin", "764012345678909")
```

Disp.Mkeypin (Title, Account,Skey,MkeyID)

Disp.Mkeypin displays a PIN entry prompt and returns a standard Masterkey PIN.

Parameters	Description	Values
Title	Optional, title message	String value to display as the title of the screen
Account	Required, account number as a string of ASCII digits.	9-19 ASCII digits
Skey	Transaction/Session key	16 ASCII hex digits
MkeyID	Stored master key ID	0-9

Return Values

This command encrypts a pin returned as a binary value.

Example

```
      Tool.Var.SetVar "PIN", Tool.DISP.Mkeypin("Enter Pin",
"764012345678909", "0123456789ABCDEF", "0")
```

Script

SCRIPT commands handle special logic that deals with scripts, such as navigation, and branching commands.

Script.DoNothing

Script.DoNothing is a placeholder. This command is mainly used with “If” statements when the developer wants the “Else” clause to do nothing.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
Script.IfTrue, Script.IfFalse

EXAMPLE
```
Tool.Script.IfTrue Tool.Object.IsEmpty(padScreen.SigArea1), _

       Tool.Sound.Bell(Alarm), Tool.Script.DoNothing()
```

This sample code checks SigArea1 to find out if the customer has signed in. If the user has not signed, the TT3100 Series terminal sounds an alarm, otherwise it does nothing.

Script.GotoRoutine(RoutineName) as Boolean

Script.GotoRoutine executes RoutineName.

Note: This command is not a subroutine. It will not continue processing after the line that called it. Execution stops at the end of RoutineName unless the last line is a **Script.Goto** command.

In the TT Advantage Assistant, clicking on the “View Labels” button causes TT Advantage to list available routine names.

Parameters	Description	Values
RoutineName	Name of user defined routine.	Must be in the format of module or form routine. (case sensitive)

RETURN VALUES
Boolean - True if successful

SEE ALSO
Script.GotoScreen

EXAMPLE
```
Tool.Script.GotoRoutine CheckIdFormat
```

This command transfers script control to the **CheckIdFormat** routine in a module of the project.

Note: This command completely transfers execution of the script. Execution does not return automatically to the calling code.

Script.GotoScreen(ScreenName)

Script.GotoScreen displays ScreenName. On the TT Advantage Assistant panel, click the “View Labels” button to get a list of available screen names to use.

Parameters	Description	Values
ScreenName	A name of a form.	case sensitive

RETURN VALUES
None

SEE ALSO
Script.GotoRoutine

EXAMPLE
```
Tool.Script.GotoScreen padGetMSR
```

Note: This command displays the **padGetMSR** form on the device.

Script.GoToVar(VarName)

Script.GotoVar is a typical goto statement. This command executes the routine VarName.

Note: This is not a subroutine; in other words, this will not continue processing after the line that called it. Program routines stop processing at routine end unless the routines are ended with **Script.Goto** commands.

In TT Advantage Assistant, clicking on the “View Labels” buttons produces a list of valid routine names.

Note: Insert routine and form names as the parameters.

Parameters	Description	Values
VarName	Variant: variable that contains the name of user defined subroutine.	Must be in the format of module or form subroutine or form name (case sensitive).

RETURN VALUES
Boolean - True if successful

SEE ALSO
Script.GotoScreen, Script.GotoRoutine

EXAMPLE
```
Tool.Var.Str.SetVar “NextScreenName”, “padGetSig”

...

Tool.Script.GotoVar “NextScreenName”
```

Note: This example will display the **padGetSig** form on the device.

Script.IfFalse(Expression, ThenClause, ElseClause)

Script.IfFalse is an If Not True statement. If the expression is false, the ThenClause executes, otherwise the ElseClause executes.

Note: Every sub-command must be enclosed in parenthesis in function syntax fashion.

Parameters	Description	Values
Expression	A script command that returns a Boolean value.	Must be a script command in function syntax.
ThenClause	The command executed if the expression returns False.	Must be a script command in function syntax.
ElseClause	The command executed if the expression returns True.	Must be a script command in function syntax.

RETURN VALUES
None

SEE ALSO
Script.IfTrue

EXAMPLE
```
Tool.Script.IfFalse Tool.Object.IsEmpty(padScreen.SigArea1),
```
Tool.Sound.Bell(Alarm), Tool.Script.DoNothing()

If the Signature Area (SigArea1) on padScreen has not been signed, then an alarm sounds.

Script.IfTrue(Expression, ThenClause, ElseClause)

Script.IfTrue evaluates the Boolean Expression. If Expression is true, the ThenClause executes. If Expression is false, ElseClause executes.

Note: Every command used as a parameter must be enclosed in parenthesis in function syntax fashion.

Parameters	Description	Values
Expression	A Script command that returns a Boolean value.	Must be a script command in function syntax.
ThenClause	The command executed if the expression returns True.	Must be a script command in function.
ElseClause	The command executed if the expression returns False.	Must be a script command in function syntax.

RETURN VALUES
None

SEE ALSO
Script.IfFalse

EXAMPLE

```
Tool.Script.IfTrue Tool.Object.IsEmpty(padScreen.SigArea),
```

Tool.Sound.Bell(Alarm), Tool.Script.DoNothing()

Script.Pause(Seconds)

Script.Pause halts script execution for the indicated number of seconds.

Note: When a script is halted with **Script.Pause**, the command allows most actions to occur and events to be handled such as button clicks. Use for displaying messages and splash screens.

Parameters	Description	Values
Seconds	Integer, the number of seconds for delay.	1- 600

RETURN VALUES
None

SEE ALSO
Script.StopScript

EXAMPLE

```
Tool.Script.Pause 30
```

Script.StopScript

Script.StopScript completely halts script execution.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
Script.Pause

EXAMPLE

```
Tool.Script.StopScript
```

Sig

The SIG commands are commands that work with signature data. In conjunction with these commands, use the **Sig.Save** command to save signatures into a binary variable.

Sig.Clear(ControlName)

Sig.Clear clears the signature area.

Parameters	Description	Values
ControlName	The name of SigControl to be cleared.	

RETURN VALUES
None

SEE ALSO
TxtBox.Clear

EXAMPLE
```
Tool.Sig.Clear padStart.SigControl1
```

Sig.Convert(VarName1, Format) as VarName

Sig.Convert changes signatures from one format to another.

Parameters	Description	Values
VarName1	String variable containing signature information for conversion to another format.	Signature variable

Format	Integer format for saving signature	Points = 1
		Packet = 5
		COTF = 7

RETURN VALUES
Variable containing string data for signature in new format.

EXAMPLE
```
Tool.Var.SetVar “SIGNATURE2” , Tool.Sig.Convert (“SIGNATURE1”,
CONF)
```

Sig.GetCount(VarName) as Integer

Sig.GetCount returns the number of points for a signature stored in VarName.

Parameters	Description	Values
VarName	String variable contains signature.	Signature variable

RETURN VALUES
Integer

EXAMPLE
```
Tool.Var.SetVar “Num” , Tool.Sig.GetCount(“Sig”)
```

Sig.Height(VarName) as Integer

Sig.Height returns the signature height in pixels for a signature that is stored in the VarName.

Parameters	Description	Values
VarName	String, variable name that contains signature	Signature variable

RETURN VALUES
Integer

SEE ALSO
Script.Inches, Script.Width

EXAMPLE
```
Tool.Var.SetVar "Height" , Tool.Sig.Height("Sig")
```

Sig.InchesHigh(VarName) as Integer

Sig.InchesHigh returns the signature height in inches x 100 for signature that is stored in the VarName.

Note: This does not return fractions, so 2.32 inches returns as 232.

Parameters	Description	Values
VarName	String variable name that contains signature	Signature variable

RETURN VALUES
Integer

SEE ALSO
Sig.InchesWide, Sig.Width, Sig.Height

EXAMPLE
```
Tool.Var.SetVar "InchesHigh" , Tool.Sig.InchesHigh("Sig")
```

Sig.InchesWide(VarName) as Integer

Sig.InchesWide returns the signature width in inches x 100 for signature that is stored in the VarName.

Parameters	Description	Values
VarName	String variable name that contains signature	Signature variable

RETURN VALUES
Integer

SEE ALSO
Sig.InchesHigh, Sig.Height, Sig.Width

EXAMPLE

```
Tool.Var.SetVar "InchesWide" , Tool.Sig.InchesWide("Sig")
```

Sig.IsEmpty(ControlName) as Boolean

Sig.IsEmpty.determines if ControlName holds a signature.

Parameters	Description	Values
ControlName	Name of SigControl	Control name in Visual Basic format

RETURN VALUES

Boolean - True if SigControl has no signature in it.

EXAMPLE

```
Tool.SCRIPT.IfTrue Tool.Sig.IsEmpty(SigControl1),_
Tool.SOUND.Bell(Success), Tool.SOUND.Bell(Fail)
```

Sig.Save (ControlName, VarName) as Boolean

Sig.Save moves signature data in a control to a binary variable.

Parameters	Description	Values
ControlName	Name of SigControl	Control name in Visual Basic format
VarName	String variable stores signature	Signature variable

RETURN VALUES

Boolean - True if successful

EXAMPLE

```
Tool.Sig.Save padStart.SigControl1, "Sig1"
```

Sig.Width(VarName) as Integer

Sig.Width returns the width of a signature in pixels for a signature stored in VarName.

Parameters	Description	Values
VarName	String, variable contains signature	Signature variable

RETURN VALUES

Integer

SEE ALSO

Sig.Height, Sig.InchesWide, InchesHigh

EXAMPLE

```
Tool.Var.SetVar "Width" , Tool.Sig.Width("SIG")
```

Sound

The SOUND commands control all audio output for the TT3100 Series terminals.

Sound.Bell(BellType)

Sound.Bell causes a TT3100 Series terminal to make the bell sound specified by BellType.

Parameters	Description	Values
BellType	BellTypes	Alarm Fail Normal_Bell Success

RETURN VALUES

None

SEE ALSO

Sound.Tone

EXAMPLE

```
Tool.Sound.Bell Alarm
```

Sound.Tone(FreqHZ, TempoBeats)

Sound.Tone causes a TT3100 Series terminal to make the tone specified by the FreqHZ and TempoBeats.

Parameters	Description	Values
FreqHZ	Integer specifying the frequency of the tone generated.	
TempoBeats	Integer specifying the duration of the tone generated	

RETURN VALUES
None

SEE ALSO
Sound.Bell

EXAMPLE
```
Tool.Sound.Tone 220, 100
```

Stopwatch

The STOPWATCH commands control the timer. Use Start, Stop, Continue and GetTimer commands to manipulate the timer and use **Var.Num** commands to program decisions based on timer information.

Stopwatch.Continue

Stopwatch.Continue re-activates the internal interactive point of sale timer from the last **Pause** command.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
Stopwatch.Pause

EXAMPLE
```
Tool.Stopwatch.Continue
```

Stopwatch.GetTime as Integer

Stopwatch.GetTime returns the number of seconds in the elapsed since timer activation. Execute this command before pausing the timer.

Parameters	Description	Values
None		

RETURN VALUES
Integer

SEE ALSO
Stopwatch.Start

EXAMPLE
```
Tool.Var.SetVar "Seconds" , Tool.Stopwatch.GetTime
```

Stopwatch.Pause

Stopwatch.Pause halts the internal timer of a TT3100 Series terminal.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
Stopwatch.Continue, Stopwatch.GetTime

EXAMPLE
```
Tool.Stopwatch.Pause
```

Stopwatch.Start

Stopwatch.Start initializes and starts the internal timer of a TT3100 Series terminal. Use **GetTime** to get the number of seconds since the **Stopwatch.Start** command was executed. **Important:** Read the value of the timer before pausing.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
Stopwatch.Continue, Stopwatch.Time

EXAMPLE
```
Tool.Stopwatch.Start
```

TxtBox

The TXTBOX commands provide the ability to manipulate text boxes.

TxtBox.IsEmpty(ControlName) as Boolean

TxtBox.IsEmpty determines if TextBox control contains data. Not the Text property of the TextBox control. Because of Text property acts as a variable, this method checks whether any values assign to the variable (i.e., TextBox). By default TextBox controls contain “VAR not defined” as their values, so initially it is not empty.

Parameters	Description	Values
ControlName	Name of TextBox control	Must be a screen control name in the format of Form.Control

RETURN VALUES
Boolean - True if successful

SEE ALSO
Sig.IsEmpty

EXAMPLE
```
Tool.TxtBox.IsEmpty(padScreen1.txtResult)
```

TxtBox.Load(ControlName, VarName) as Boolean

TxtBox.Load copies the contents of the VarName into ControlName.

Parameters	Description	Values
ControlName	Name of the TextBox control	Must be a screen control name in the format of form control.
VarName	Variable name contains string value	String

RETURN VALUES
Boolean - True if successful

SEE ALSO
TxtBox.Save

EXAMPLE

```
Tool.TxtBox.Load PadScreen1.TextBox1, "CustName"
```

This example takes the value in the variable **CustName** and copies it into the **padScreen1** form textbox control **TextBox1**.

TxtBox.Save(ControlName, VarName) as Boolean

TxtBox.Save copies the contents of the ControlName into the VarName.

Parameters	Description	Values
ControlName	Name of TextBox	Must be a screen control name in the format of Form.Control
VarName	Variable name stores string value	String

RETURN VALUES
Boolean - True if successful

SEE ALSO
TxtBox.Load

EXAMPLE

```
Tool.TxtBox.Save padScreen1.txtCustName, "CustName"
```

This command copies the contents of the TextBox control **txtCustName** on the form **padScreen1** into **CustName**.

TxtBox.SendKeys(ControlName, Keys) as Boolean

TextBox.SendKeys sends text to a control then refreshes the control image. This command appends text to the contents of a Control.

Parameters	Description	Values
ControlName	Name of the TextBox control	Must be a screen control name in the format of form control
Keys	String	Characters to append

RETURN VALUES
Boolean - True if successful

EXAMPLE
```
Tool.TxtBox.SendKeys padScreen1.txtTotal, "110.99"
```

This command appends the text "110.99" to the end of the TextBox control, (txtTotal), on the form, (padScreen)1.

Unit

The UNIT commands provide information about a TT3100 Series terminal the script is running.

Unit.Model as Integer

Unit.Model returns the model number of the TT3100 Series terminal running the script.

Parameters	Description	Values
None		

RETURN VALUES
Integer

SEE ALSO
Unit.Version

EXAMPLE
```
Tool.Var.SetVar "NumModel" , Tool.Unit.Model
```

Unit.Version as Integer

Unit.Version returns the version number of the firmware running on a TT3100 Series terminal.

Parameters	Description	Values
None		

RETURN VALUES
Integer

SEE ALSO
Unit.Model

EXAMPLE
```
Tool.Var.SetVar “NumVersion” , Tool.Unit.Version
```

Var

The VAR commands handle variables. There are subcommands under the VAR command. These subcommands are Bin, Num, and Str. They control the following:

Command	Data Used
Var.Bin	Binary data
Var.Num	Numeric data (integers)
Var.Str	String data

Var.DeleteVar(VarName)

Var.DeleteVar removes the variable name and frees its memory resources.

Parameters	Description	Values
VarName	The name of a variable	Case sensitive

RETURN VALUES
None

EXAMPLE
```
Tool.Var.DeleteVar “IsCardSwiped”
```

Var.FindVar(VarName) as Boolean

Var.FindVar indicates if VarName exists.

Parameters	Description	Values
VarName	The name of a variable	Case sensitive

RETURN VALUES
Boolean - True if the variable exists.

EXAMPLE
```
Tool.SCRIPT.IfTrue Tool.Var.FindVar("CustName"),
Tool.SOUND.Bell(Success), Tool.SOUND.Bell(Fail)
```

Var.SetVar(VarName, Command) as Boolean

Var.SetVar assigns the result of a command into VarName

Parameters	Description	Values
VarName	Name of a variable (case sensitive).	Case sensitive
Command	A command that returns a value.	Script command with function syntax

RETURN VALUES
Boolean - True if VarName was successfully set to the value that Command returned.

EXAMPLE
```
Tool.Var.SetVar"FullName", Tool.Var.Str.Concat("FirstName",
"Space", "LastName")
```

Var.Bin

Var.Bin commands handle binary data.

Var.Bin.SetVar(VarName, Data) as Boolean

Var.BinSetVar sets VarName to Data (binary).

Parameters	Description	Values
VarName	Name of a variable	Case sensitive
Data	Raw binary data	Binary data

RETURN VALUES
Boolean - True if successful.

SEE ALSO
Var.Num.SetVar, Var.Str.SetVar

EXAMPLE
```
Tool.Var.SetVar "IsSet" , Tool.Var.Bin.SetVar "Mode", 2
```

This line of code sets the variable, "Mode**",** to 2.

Var.Bin.ToBase64(VarName) as String

Var.Bin.ToBase64 converts binary data in VarName to Base64 format.

Parameters	Description	Values
VarName	Name of variable containing binary data	Case sensitive

RETURN VALUES
String that contains data in Base64 format

SEE ALSO
Var.Bin.ToHex

EXAMPLE
```
Tool.Var.SetVar "Mode_Base64" , Tool.Var.Bin.ToBase64 ("Mode")
```

Var.Bin.ToHex(VarName) as String

Var.Bin.ToHex converts binary data in VarName to hexadecimal format.

Parameters	**Description**	**Values**
VarName	Name of variable that contains binary data	Case sensitive

RETURN VALUES
String that contains data in hexadecimal format

SEE ALSO
Var.Bin.ToBase64

EXAMPLE
```
Tool.Var.SetVar "ID_Hex" , Tool.Var.Bin.ToHex ("ID")
```

Var.Num

VAR.NUM commands handle numeric information in the form of integers.

Var.Num.Dec(VarName)

Var.Num.Dec decrements the number in VarName by one.

Parameters	**Description**	**Values**
VarName	Name of variable that contains numeric data	Case sensitive

RETURN VALUES
None

SEE ALSO
Var.Num.Inc

EXAMPLE
```
Tool.Var.Num.Dec "Count"
```

Var.Num.Diff(VarName1, VarName2) as Integer

Var.Num.Diff subtracts the integer stored in VarName2 from the integer stored in VarName2 and returns the difference between the variables.

Parameters	Description	Values
VarName1	Name of variable that contains numeric value	Case sensitive
VarName2	Name of variable that contains numeric value	Case sensitive

RETURN VALUES
Integer

EXAMPLE
```
Tool.Var.SetVar "Difference" , Tool.Var.Num.Diff("Total",
"SubTotal")
```

Var.Num.GetVar(VarName) as Integer

Var.Num.GetVar returns the numeric value assigned to the variable.

Parameters	Description	Values
VarName	Name of variable that contains numeric value	Case sensitive

RETURN VALUES
Integers

SEE ALSO
Var.Str.GetVar

EXAMPLE
```
Tool.Var.SetVar "NumMode" , Tool.Var.Num.GetVar ("Mode")
```

Var.Num.Inc(VarName)

Var.Num.Inc increments the integer stored in VarName by one.

Parameters	Description	Values
VarName	Name of variable that contains numeric value	Case sensitive

RETURN VALUES
None

SEE ALSO
Var.Num.Dec

EXAMPLE

```
Tool.Var.Num.Inc"Count"
```

Var.Num.IsEqual(VarName, Value) as Boolean

Var.Num.IsEqual determines if an integer stored in VarName is identical to Value.

Parameters	Description	Values
VarName	Name of variable that contains numeric data	Case sensitive
Value	An integer that is compared to VarName	Integer

RETURN VALUES

Boolean - True if the variable is equal to the value.

SEE ALSO

Var.Str.IsEqual

EXAMPLE

```
Tool.Var.Num.IsEqual("Mode", 2)
```

Var.Num.IsGreater(VarName, Value) as Boolean

Var.Num.IsGreater indicates if data in VarName is greater than Value.

Parameters	Description	Values
VarName	Name of a variable that contains numeric data	Case sensitive
Value	An integer that is compared to VarName	Integer

RETURN VALUES

Boolean - True if the variable is greater than Value

EXAMPLE

```
Tool.Var.Num.IsGreater("Total", 10000)
```

Var.Num.IsLess(VarName, Value) as Boolean

Var.Num.IsLess indicates if an integer in VarName is less than the integer Value.

Parameters	Description	Values
VarName	Name of a variable that contains numeric data	Case sensitive
Value	An integer that is compared to VarName	Integer

RETURN VALUES

Boolean - True if the variable is less than Value

EXAMPLE

```
Tool.Var.Num.IsLess("Total", 100)
```

Var.Num.SetVar(VarName, Value) as Boolean

Var.Num.SetVar assigns Value to the VarName.

Parameters	Description	Values
VarName	Name of a variable that contains numeric data	Case sensitive
Value	An integer assigned to VarName	Integer

RETURN VALUES

Boolean - True if the Value is assigned to VarName.

EXAMPLE

```
Tool.Var.Num.SetVar "Mode", 4
```

Var.Num.Sum(VarName1, VarName2) as Integer

Var.Num.Sum sums integers contained in VarName1 and VarName2.

Parameters	Description	Values
VarName1	Name of variable that contains numeric data	Case sensitive.
VarName2	Name of variable that contains numeric data	Case sensitive

RETURN VALUES

Integer - The sum of all parameters

EXAMPLE

```
Tool.Var.SetVar "Sum" , Tool.Var.Num.Sum("LenFirstName",
"LenLastName")
```

Var.Str

Var.Str commands handle string data.

Var.Str.Concat(VarName1, VarName2) as String

Var.Str.Concat concatenates the string contained in VarName2 to the end of the string contained in VarName1.

Parameters	Description	Values
VarName1	Name of variable that contains string data	Case sensitive
VarName2	Name of variable that contains string data	Case sensitive

RETURN VALUES
String

EXAMPLE
```
Var.Str.SetVar "Comma", ","
```

Tool.Var.SetVar “Name” , Tool.Var.Str.Concat(“LastName”, “Comma”, “FirstName”)

Var.Str.GetVar(VarName) as String

Var.Str.GetVar returns the string stored in VarName.

Parameters	Description	Values
VarName1	Name of variable that contains string data	Case sensitive

RETURN VALUES
String

EXAMPLE
```
Tool.Var.SetVar "Name" , Tool.Var.Str.GetVar("LastName")
```

Var.Str.IsEqual(VarName, Value) as Boolean

Var.Str.IsEqual determines if string data contained in VarName is identical to Value.

Parameters	Description	Values
VarName	Name of variable that contains string data to compare with Value	Case sensitive
Value	A string that is compared to VarName	

RETURN VALUES
Boolean - True if equal

EXAMPLE
```
Tool.Var.Str.IsEqual "LastName", "Jones"
```

Var.Str.Left(VarName, Count) as String

Var.Str.Left returns left number of characters from string data contained in VarName.

Parameters	Description	Values
VarName	Name of variable that contains string data	Case sensitive
Count	Integer number of characters for extraction from the left edge of string data in VarName	

RETURN VALUES
String

EXAMPLE
```
Tool.Var.SetVar "Left" , Tool.Var.Str.Left("LastName", 3)
```

Var.Str.Right(VarName, Count) as String

Var.Str.Right returns right most specified number of characters from the string data contained in VarName.

Parameters	Description	Values
VarName	Name of variable that contains string data	Case sensitive
Count	Integer number of characters for extraction from the right edge of string data in VarName.	

RETURN VALUES
String

EXAMPLE
```
Tool.Var.SetVar "Right" , Tool.Var.Str.Right("Name", 2)
```

Var.Str.SetVar(VarName, Value) as Boolean

Var.Str.SetVar assigns string data to VarName.

Parameters	Description	Values
VarName	Name of variable created for storing string data	Case sensitive
Value	String data for storing in VarName	

RETURN VALUES
Boolean - True if successful

EXAMPLE
```
Tool.Var.Str.SetVar "IsCardSwiped", "Yes"
```

VB Mode

TT Advantage VB mode generates PC-resident applications that control TT3100 Series terminals. Programmers creating electronic transaction systems that consist of a Visual Basic application in a Windows operating system should use the VB mode, while programmers upgrading legacy systems should use the Script mode.

VB mode applications are programs in which the entire application resides on the host. That is opposed to Script mode applications in which screen generation and data collection is handled by a script or program residing on the TT3100 Series terminal.

When to Use VB Mode

TT Advantage VB mode is especially useful to developers who need to quickly develop interactive point of sale applications or prepare mock-ups and demos for project proposals.

TT Advantage supplies the tools to quickly implement new applications for TT3100 Series terminals. TT Advantage extends the Visual Basic programming environment to include special forms, controls, and classes for creating TT3100 Series interactive point of sale applications. With TT Advantage, the need to generate machine or C code is eliminated. All of the low level details are taken care of by the TT Advantage Assistant.

Developing Applications in the VB Mode

The visual programming environment of TT Advantage provides a high degree of parallelism between the host environment and interactive point of sale environment. Objects appearing on host forms are mirrored on interactive point of sale screens except those that should be hidden on screen. Both platforms operate simultaneously from the same program. Most program responses triggered by actions at the host closely match the program responses triggered by actions to the TT3100 Series terminal.

Honing interactive point of sale applications proceeds quickly. As soon as the host runs the TT Advantage application, the form appearing on the host screen appears on the TT3100 Series terminal. The close coupling between host and TT3100 Series terminals enables rapid iteration of application designs. The responsive TT Advantage interface allows cause-and- effect between application changes and application performance to remain clear.

Because developer hosts can run the VB mode application, it is possible to test many aspects of this application without having to hook a TT3100 Series terminal to the host. To test a VB mode application, set the project into test mode by setting the global variable, TEST_MODE, to “True.” TEST_MODE is found at the top of the ModMain module.

Communication

The port number and baud rate can configure communication with a TT3100 Series terminal. The maximum number of ports to be used can be changed accordingly. If the port number specified is beyond the range, COM1 is selected as a default.

```
    SIGBOX.PORTS = 4  ◄──────────────
    '              ^
    '              CHANGE THIS TO DEFINE MAXIMUM NUMBER OF PORTS TO BE
USED.
    '


    SigBox.Port = 0  ◄──────────────
    '             ^
    '             change this to connect to necessary port
    '             "0" is for auto select
    '
    SigBox.BaudRate = Baud9600  ◄────────────
    '                   ^
    '                   change this to select required baud
rate
    '                   if necessary
    '

    V.A.start                       'VA Required
    '---TT Advantage required code ABOVE
```

Distributing TT Advantage Applications

After completing a TT Advantage VB mode application, create an exe program using the normal Visual Basic “Make Exe” process.

To distribute applications to other machines, do one of the following:

1. Copy the “Distribution Setup folder” to the target machine. Run Setup.exe. This places all files normally required to run the program on the target machine. The executable file should now be able to run.

-or-

2. Use an Install creation application (such as Visual Basic’s Application Setup Wizard, or InstallShield) to include all the necessary files in a Setup program for your executable file. Depending on the project, additional files can be required. The additional files that might be required are the following: **mxVAhost.dll** and **VisualControls. ocx.** Both files are found in the system directory.

VB Mode Software Components

Several software components are required for TT Advantage to function properly:

Class

Name	Description	Additional Notes
VAevents.cls	This class allows code to be put behind certain VA events.	Code for this component can be modified in certain areas. See the code in the class module for more information.

Controls

Name	Description	Additional Notes
VisualControls.ocx	Provides the SigControl and MSRArea controls	DO NOT MODIFY. These function like normal OCX controls and must be listed in the controls windows.

Forms

Name	Description	Additional Notes
frmSplash.frm	Application specific splash screen.	A modifiable graphical form for announcing an application or function or for displaying logos on the TT3100 Series terminal. Modify the splash screen for the form under development.
padStart.frm	The first screen must always be padStart.	To add more forms, see TT Advantage Forms on page 2-5.
MDIMain.frm	This is the parent screen for all the displayed forms.	DO NOT MODIFY.

Modules

Name	Description	Additional Notes
modMain.bas	modMain.bas module connects TT Advantage classes.	Add functions and subroutines to this file in the areas designated inside the code.

References

Name	Description	Additional Notes
Sigbox.ocx	Sigbox.ocx provides connectivity between host and TT3100 Series terminals. Sigbox.ocx also supplies additional “non-standard” commands.	DO NOT MODIFY. Note: Sigbox.ocx is a control, but is used as a reference. Normally the SigBox control is not needed for a TT Advantage project, but it needs to exist in the project as reference. SigBox is accessed indirectly through TT Advantage.
MxVAhost.dll	Controls the communication with the TT3100 Series terminal	DO NOT MODIFY.

VB Mode Controls

TT Advantage forms generate screen displays that run in a special environment - on TT3100 Series terminals. The special demands of the Hand Held Products interactive point of sale environment requires a different set of controls than the Windows environment native to Visual Basic. Thus, while many of the controls function in TT Advantage as they do in Visual Basic alone, some do not.

Control Font Sizes

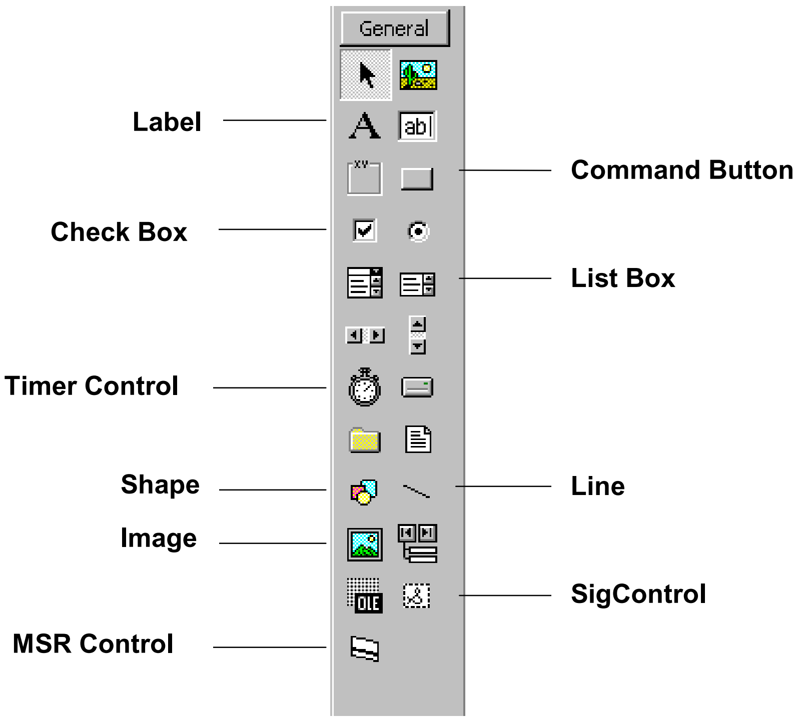
Font characters displayed on TT3100 Series terminals come in six sizes. All characters in each font size have the same pixel dimensions. The TT Advantage font families are mapped onto the Courier New font families in the properties boxes of TT Advantage controls. The following table shows the correspondence between the TT Advantage font sizes and Courier New font sizes.

TT3100 Series Terminal Font Sizes in Pixels	Courier New Font Sizes in Points
6 x 8	10 to 11 pt
8 x 8	12 pt
8 x 12	13 to 16 pt
12 x 16	17 to 22 pt
16 x 16	23 to 26 pt
16 x 24	27 to 30 pt

TT Advantage Tool Box

Controls in the following diagram work in TT Advantage VB mode, some with modified capabilities. The MSRControl and SigControl are specialized controls supplied by TT Advantage.

Controls that can be used in VB mode are shown and described below:



CheckBox

The TT Advantage VB mode Check Box operates just as the Visual Basic Check Box. In VB mode, the check box can also be used in an option group by setting the tag property.

Property	How property is used
Font-Size	Controls the size on the font on the pad. See Control Font Sizes, above.
Value	Sets the default value when displayed, depending on the status of the check box.
Caption	Controls the text displayed beside the check box.
Tag	Setting Tag to “hide” (lowercase) causes this control to be hidden. Setting Tag to “groupname” allows a check box to be used as part of an option group. When using multiple check boxes on the form, set the Tag property of each to the same value. Then, checking one box causes all others to uncheck. Groupname can be any string except the words “groupname” and “hide”.

USING THE CONTROL

Touching the CheckBox image on the TT3100 Series terminal checks and un-checks the control. Use the normal Visual Basic control properties for determining the status of the check box (checkbox1.value).

SEE ALSO

Commands**: V.A.PadFontSize, V.A.FontWidth, V.A.FontHeight**

Command Button

The TT Advantage command button functions the same way as the command button of the pure Visual Basic environment.

Property	How property is used
Font-Size	Controls the size on the font on the pad. See Control Fonts section in this chapter.
Caption	Controls the text displayed inside the button.
Value	Sets the default value when displayed. Setting to True activates the click event of the command button.
Tag	Setting Tag to “hide” (lowercase) makes the control invisible.
Style	Setting Style to “0 - Standard” displays a normal button. Setting Style to “1 - Graphical” displays a transparent button.

USING THE CONTROL

Use like a normal Visual Basic button. Put code under click_Event to execute commands when clicked.

Image

Image controls display a black-and-white bitmap graphics on interactive point of sale screens.

Images in VB mode stretch to fit the size of the image control. Images in VB mode can be in any VB compatible format (BMP, WMF, etc.).

Static images in VB mode can be stored in the non-volatile TT3100 Series terminal so they appear faster and don't have to be transmitted to the interactive point of sale. To store an image, command TT Advantage to pre-load the form where the image resides. Use the Load “formname” in the “A_PreLoadForms” event of the Class module “VAEvents.”

Note: For best image quality set the image control to the exact size of the imported image. Stretching the image can drastically alter the quality of the image displayed on the TT3100 Series terminal. Images should not be larger than 240 x 320 pixels and must not extend past the edges of the screen.

Property	How property is used
Tag	Setting Tag to “hide” (lowercase) causes the control to be invisible.
BorderStyle	Setting BorderStyle to “0 - None” causes the Image control to display no border. Setting BorderStyle to “1 - Fixed Single” causes the Image control to display a border around the picture.
ToolTipText	Text entered into ToolTipText property box shows temporarily until the image displays - in Web fashion.
Stretch	The stretch property is not really used by TT Advantage. Setting the property to “'True” allows TT Advantage to display a sized image. Leaving the property “False” causes Visual Basic to resize it back to the original image size when the program runs. If Stretch is “False”, the size of the image cannot change.

USING THE CONTROL
Use the TT Advantage image control like a normal Visual Basic image control. Set stretch property to “True” for correct functionality.

Label

The Label control in VB mode functions like a normal label and includes the capability to word wrap.

Property	How property is used
Tag	Setting Tag to “hide” (lowercase) makes this control invisible. Setting Tag to “clear” (lowercase) clears text from the label before it displays.
Font-Size	Font-Size sets the size on the font displayed on the form.
BorderStyle	Set BorderStyle to “0 - None” to make the label display with no border. Set BorderStyle to “1 - Fixed Single” to make the label display with a border.

USING THE CONTROL
Use TT Advantage labels just like Visual Basic labels.

SEE ALSO
Commands: **V.A.DisplayLabel**

Line

The Line control works the same in TT Advantage VB mode as in the pure Visual Basic environment.

ListBox

The ListBox control in TT Advantage VB mode works just like a ListBox control in pure Visual Basic. Use for displaying a list of selectable items that user can pick by touching the TT3100 Series screen. A ListBox on the TT3100 Series terminal adds a scroll bar and buttons when there are more lines in the ListBox than can be displayed all at once.

Property	**How property is used**
Tag	Setting Tag to “hide” (lowercase) hides this control. Setting Tag to “scroll” places a scroll bar on the right side even if there is no need to scroll. Setting Tag to “groupname” allows use of the ListBox as an option group. If using multiple ListBoxes on a form, set the tag properties of each ListBox to the same value. Then selecting an item on one ListBox, de-selects the item on all the others.
Font-Size	Sets the size on the font on the pad.
ToolTipText	After the ListBox displays on the TT3100 Series terminal, the ToolTipText property displays the number of selections the ListBox holds. For example, if a ListBox draws on a screen and this property contains “7”, this means that the ListBox displays 7 lines at a time.
Enabled	Setting this property to True, allows the TT Advantage code to accept a click event.

USING THE CONTROL
Use like a normal Visual Basic ListBox.

MSRControl

Place the MSR Control anywhere on the form to activate the MSR. This control is visible on the host screen yet invisible on the interactive point of sale display. Once activated, the MSR captures data from card swipes. If a good swipe occurs, TT Advantage fires the GoNext Event for the form. The GoBack event fires where the MSR times out. Such an event occurs where the interactive point of sale didn't get a swipe within the time limit programmed for the swipe.

In VB Mode, do not put code under the MSR GoodSwipe or BadSwipe events. TT Advantage ignores code in those events while in VB mode. Instead, put code under the GoNext event for the current form for good swipe. (Use timeout = 0 for no timout).

EVENTS:

- GoNext - fires after a good swipe.
- GoBack - fires after no swipe before timeout.

Property	How property is used
Timeout	The number of seconds to wait before firing the GoBack event on the form. Property defaults to "0" for no timeout.

USING THE CONTROL

When using the MSRControl to design screens, the control is visible on the host development screen but does not show on the interactive point of sale screen. After drawing this control on the form it will not appear on the device.

To obtain data from the MSR, call the **MSR.AssignTrackNames** method, then use the MSR object to get the properties from the MSR. See MSR object in this chapter.

Draw this control on the developer screen.

SEE ALSO

Commands:

SigControl

The Signature Control displays signature data captured from the TT3100 Series terminal as a series of dots within the signature area. Use the Line control to draw a signature line in the signature area.

Property	How property is used
BorderStyle	Setting BorderStyle to "0 - None" causes the SigControl to be drawn without a border. Setting BorderStyle to "1 - Fixed Single" causes a border to be drawn.

USING THE CONTROL

Use this control to draw a signature area. To save the signature into a string variable, use the command: **V.A.Signature.** See VB Mode Commands on page 4-10.

SEE ALSO

Commands: **V.A. Signature, V.A.Shrinkwrap**

Shape

The Shape tool only displays rectangles on the TT3100 Series terminal. After placing a Shape on a form, a dialog box appears asking if the object should be a “Box (Filled)” or a “Frame (Transparent)”. For correct functionality the shape must be set to rectangle.

Property	**How property is used**
BorderStyle	Setting BorderStyle to “0 - None” causes the rectangle to be drawn without a border. Setting BorderStyle to “1 - Fixed Single” causes the border to be drawn around the shape.
FillStyle	Setting FillStyle to “0 - Solid” fills the rectangular area with black. Setting FillStyle to “1-Transparent” draws a transparent rectangle.
FillColor	Setting FillColor to “0 -Solid” fills the rectangle with Black. Setting FillColor to non-zero fills the rectangle with White.

Note : If “FillStyle” is set to “1-Transparent,” FillColor property has no control to make the shape filled with black as described above.

TimerControl

The TT Advantage VB mode Timer Control does not operates like Visual Basic timer control.

The coding under the Timer1_Timer() event only executes after the given interval is reached after enabling the timer, and it executes only once. VA only recommends one timer control per form.

Property	**How property is used**
Interval	Controls the delay for the action follows the timer
Enabled	Activates/deactivates timer

USING THE CONTROL

TT Advantage VB mode timers do not act like Visual Basic timers. VB mode timers are ‘countdown’ timers. Enter the countdown time in the Interval box of the timer’s properties.

VB Mode Commands

All TT Advantage commands require a command prefix. This prefix depends on the TT Advantage mode. For VB Mode, there are 3 command prefixes: “V.A.”, “MSR” and “SigBox”. Use TT Advantage Command prefixes to bring up Visual Basic's auto coding drop-down lists. They help complete commands and walk the developer through required parameters. This section describes all the TT Advantage commands that are available.

V.A.

TT Advantage relies on the methods and properties of the SigBox object to function. Most of the properties and methods of SigBox are accessed indirectly through the TT Advantage interface. For developers wanting to use the large number of methods and properties of SigBox directly, information about them can be found in the Software Developers Kit (SDK) documentation.

V.A.

V.A. Commands relate specifically to an attached TT3100 Series terminal. Some of V.A. commands are used by TT Advantage to initialize communication with the terminal.

V.A.CurrentScreen

Command Type: Object

V.A.CurrentScreen is a form method holding information about the currently displayed form. **V.ACurrentScreen** gives access to all the properties and methods of a form.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
Visual Basic Manuals: Forms

EXAMPLE

```
Dim CurForm as Form

Set curForm = V.A.CurrentScreen
```

V.A.DisplayPreviousScreen

Command Type: Method

V.A.DisplayPreviousScreen navigates back to the previous screen. If the current screen is the first screen, TT Advantage terminates normally.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
None

EXAMPLE

```
V.A.DisplayPreviousScreen
```

V.A.DisplayScreen ScreenName

Command Type: Method

V.A.DisplayScreen displays ScreenName.

Parameters	Description	Values
ScreenName	The name of the screen to display	For any TT Advantage form name to display on the device; That form name should start with "pad".

RETURN VALUES
None

SEE ALSO
V.A. DisplayPreviousScreen, V.A.RedrawScreen

EXAMPLE
V.A. DisplayScreen, padThankYou

V.A.DrawLabel Lbl [, ClearScreenUnder Label]

Command Type: Method

V.A.DrawLabel Lbl draws a label on the current screen. The label does ***not*** have to come from the current screen. This allows the developer to change the contents, then display the new contents on the screen.

Updating a label on the TT3100 Series screen is the one of the main uses for this command. For example, a label on the screen with a caption of "Please Swipe your Card." can change to read "Thank you, Please wait"...**V.A.DrawLabel Lbl** changes the labels caption but does not display on the screen until a **V.A.DrawLabel** command executes or the **V.A.RedrawScreen** command is executed.

Error messages make a good application for the **DrawLabel** command. To do that:

1. Create a label on a screen containing a message for a user making an expected error.
2. Set the tag property of the label to "hide". This keeps the label from displaying.
3. If the user makes the anticipated mistake, update the caption then have the program execute a **DrawLabel** command so that the label text appears on the TT3100 Series terminal.

Parameters	Description	Values
Lbl	A label for display	Use full name of label. If the label resides on a different form prefix the label name with the label, form name. Example: **padScreen1.lblMessage**
ClearScreenUnderLabel (optional parameter)	Setting **ClearScreen UnderLabel** to “True” clears the screen under the label before drawing. This allows the label to overwrite other screen elements.	Optional parameter Boolean: True or False

RETURN VALUES
None

SEE ALSO
Controls: Labels

EXAMPLE

```
V.A.DrawLabel lblMessage, True
```

V.A.DrawScreenHotSpot

Command Type: Method

V.A.DrawScreenHotSpot creates an invisible button that covers then entire screen. When the screen is touched anywhere, the hidden button activates and the **GoNext** command of the screen fires.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
Forms: **GoNext** event

EXAMPLE

```
V.A.DrawScreenHotSpot
```

V.A.ErrorDescription

Command Type: Property

V.A.ErrorDescription displays error messages to the user. To display an error message, set **V.A.ErrorDescription** to the error message, then go to another screen with **V.A.DisplayScreen** or re-display the current screen with **V.A.RedrawScreen**. After the screen redraws, the error message appears along with an “OK” button. When the OK button is clicked, TT Advantage starts over at padStart.

Note: No variables lose information in the process.

Parameters	Description	Values
None		

SEE ALSO

V.A.ShowError

EXAMPLE

```
V.A.ErrorDescription = "Database connection lost."

V.A.RedrawScreen
```

V.A.FontHeight padFont

Command Type: Property

V.A.FontHeight pad returns the height of the padFont parameter in pixels.

Parameters	Description	Values
PadFont	A defined pad font size.	Font6x8 Font8x8 Font8x12 Font12x16 Font16x16 Font16x24

RETURN VALUES

Integer value

SEE ALSO

V.A.FontWidth, V.A.PadFontSize

EXAMPLE

```
FontYSize = V.A.FontHeight(Font8x8)
```

V.A.FontWidth padFont

Command Type: Property

V.A.FontWidth.padFont returns the width of the indicated font in pixels.

Parameters	Description	Values
PadFont	A defined pad font size	Font6x8 Font8x8 Font8x12 Font12x16 Font16x16 Font16x24

RETURN VALUES
Integer value

SEE ALSO
V.A.FontHeight, V.A.PadFontSize

EXAMPLE
```
FontXSize = V.A.FontWidth (Font16x16)
```

V.A.InvertControl ControlName

Command Type: Method

V.A.InvertControl reverses or inverts the colors of ControlName on the screen.

Parameters	Description	Values
ControlName	Any existing control on a pad screen that has Left, Top, Width, and Height properties	Examples of the controls are: Labels, buttons, and images.

RETURN VALUES
None

EXAMPLE
```
V.A.InvertControl lblMessage
```

V.A.MagOff

Command Type: Method

V.A.MagOff turns off the Mag Card reader (MSR Reader).

Parameters	Description	Values
None		

Return Values

None

See Also

`V.A.MagOn, CONTROLS: MSRcontrol`

Example

V.A.MagOff

V.A.MagOn

Command Type: Method

V.A.MagOn turns on the Mag Card reader (MSR Reader). This method provides the same functionality as the MSRControl.

To use **V.A.MagOn** put the command in the **ScreenInitialize** event on a form.

Parameters	Description	Values
None		

Return Values

None

See Also

V.A.MagOff, CONTROLS: MSRControl

Example

```
V.A.MagOn
```

V.A.PadFontSize

Command Type: Method

V.A.PadFontSize returns the size of the font in the defined pad font size, **SigBoxTextFont**.

Use **V.A.PadFontSize** in connection with **FontWidth** and **FontHeight** to determine the screen width of a character in a control. To learn how wide the characters are in a label. Pass the result of **PadFontSize** to **PadWidth** (see example below)

Parameters	Description	Values
	Any object that has a Visual Basic Font.	

RETURN VALUES
None

SEE ALSO
V.A.FontWidth, V.A.FontHeight

EXAMPLE
This command returns the width of each character in the label. Each character has the same width so only one value is returned.

```
CharWidth = V.A.FontWidth(V.A.PadFontSize(lblMessage))
```

V.A.RedrawScreen

Command Type: Method

V.A.RedrawScreen clears the current screen and redraws it.

Parameters	Description	Values
None		

RETURN VALUES
None

SEE ALSO
V.A.DisplayScreen

EXAMPLE

```
V.A.RedrawScreen
```

V.A.RefreshListBox

Command Type: Method

V.A.RefreshListBox redraws the list box control on the TT3100 Series terminal. You need this if you add new items to the list box or remove some.

Parameters	Description	Values
ListBoxControl	Valid List Control Object	

RETURN VALUES

None

SEE ALSO

Using the ListBox Control

EXAMPLE

```
V.A.RefreshListBox
```

V.A.ScreenStackClear

Command Type: Method

V.A.ScreenStackClear clears a stack of screens that loaded to the TT3100 Series terminal. If you need to go back to a previous screen, you must clear screen stack.

Parameters	Description	Values
None		

RETURN VALUES

None

SEE ALSO

V.A.DisplayScreen

EXAMPLE

V.A.ScreenStackClear

V.A.ShowError ErrorMsg

Command Type: Method

V.A.ShowError Error displays an error message on the interactive point of sale screen with an “OK” Button. When this button is clicked, TT Advantage starts over at padStart.

Parameters	Description	Values
ErrorMsg	Text message to display	String Value

RETURN VALUES
None

SEE ALSO
V.A.ErrorDescription

EXAMPLE

```
V.A.ShowError "Cannot write to file"
```

V.A.ShrinkWrap

Command Type: Property

Set **ShrinkWrap** to True to reduce the amount of memory storage a signature occupies. Setting **ShrinkWrap** to False causes the entire signature screen, 320 x 240 pixels, to save to memory. With **ShrinkWrap** set to "True", the extra pixels surrounding the signature are trimmed off before the save. A signature with dimensions of 50 x 200 pixels takes up more than ***seven*** times as much space in memory when **ShrinkWrap** is set to False than when it is set to True.

Parameters	Description	Values
= True = False	"True" shrink-wraps the signature. "False", is the default value.	

RETURN VALUES

None (Write only)

SEE ALSO

V.A.Signature

EXAMPLE

```
V.A.ShrinkWrap = True
```

V.A.Signature

Command Type: Property

V.A.Signature saves a signature into a string variable. This property is the recommended way to obtain signatures from TT3100 Series terminals. Set the property to a string variable, then write the variable to a database or other file.

Parameters	Description	Values
None		

RETURN VALUES

Signature in the form of a string

SEE ALSO

V.A.ShrinkWrap

EXAMPLE

```
Dim Sig as String

Sig = V.A.Signature
```

V.A.Smoothing

Command Type : Property

V.A.Smoothing Sets the level of smoothing applied to captured signatures. Smoothing is used to remove unwanted lines from the signature. Average smoothing is 5. More than 10 is not recommended because it causes loss of data in signature.

Parameters	Description	Values
None		

RETURN VALUES

None (Write only).

SEE ALSO

`V.A.Signature`

EXAMPLE

```
V.A.Smoothing = 5
```

V.A.Terminate

Command Type: Method

V.A.Terminate ends the current application.

Parameters	Description	Values
None		

RETURN VALUES

None

EXAMPLE

```
V.A.Terminate
```

MSR

The MSR object can be accessed at any point after a good swipe.

MSR.AccountNumber

Command Type: Property

MSR.AccountNumber returns the account number that was read off of Track 1. This property does not work with all Mag Stripe cards. The remaining data on track1 can still be accessed using MSR.Track1 property.

Parameters	Description	Values
None		

RETURN VALUES

String

EXAMPLE

```
Dim AcctNum as String

AcctNum = MSR.AccountNumber
```

MSR.AssignTrackNames

Command Type: Method

MSR.AssignTrackNames assigns all MSR information Track 1, Track 2 and Track 3 information: FirstName, LastName, and AccountNumber. Call **MSR.AssignTrackNames** before any property gets information. On a good swipe, this method gets called automatically.

Parameters	Description	Values
None		

RETURN VALUES

None

EXAMPLE

```
MSR.AssignTrackNames
```

MSR.FirstName

Command Type: Property

MSR.FirstName returns first name information from track 1. **MSR.FirstName** does not work with all Mag Stripe cards. The data on Track1 can still be accessed by using **MSR.Track1**.

Parameters	Description	Values
None		

RETURN VALUES

String

EXAMPLE

```
Dim FName as String
FName = MSR.FirstName
```

MSR.LastName

Command Type: Property

MSR.LastName returns the last name information from Track 1. **LastName** does not work with all Mag Stripe cards. The data on Track1 is accessed by using **MSR.Track1**.

Parameters	Description	Values
None		

RETURN VALUES

String

EXAMPLE

```
Dim FULLName as String

FULLName = Trim(MSR.FirstName) + " " + Trim(MSR.LastName)
```

MSR.Reset

Command Type: Method

MSR.Reset clears all MSR object properties (FirstName,Track1, etc).

Parameters	Description	Values
None		

RETURN VALUES

None

EXAMPLE

```
MSR.Reset
```

MSR.Track1

MSR.Track2

MSR.Track3

Command Type: Property

Returns the information from Track 1, Track 2 or Track 3.

Parameters	Description	Values
None		

RETURN VALUES

String

EXAMPLE

```
Dim TrackOne as String
TrackOne = MSR.Track1
```

SigBox

Normally TT Advantage, V.A. Commands, and the MSR object provide all the necessary interface and controls needed to create TT3100 Series applications. If some facet of TT3100 Series terminal function is beyond the properties and methods in the V.A. and MSR objects, use the methods and properties of the SigBox object.

The SigBox object contains over 140 properties and methods, and is not re-documented in detail here because of size considerations. Refer to those objects and properties in the Hand Held Products Software Developers Kit (SDK) included with TT Advantage.

VAEvent Class Module

VA Events are modifiable subroutines in the VAEvent Module governing the start-up and shut-down of TT3100 Series terminals.

A_ErrorBeforeStart Event

A_ErrorBeforeStart event fires if TT Advantage fails to start or find the TT3100 Series terminal.

A_PreLoadForms Event

A_PreLoadForms stores code for pre-loading graphics into the non-volatile memory of TT3100 Series terminals and fires upon start-up. Preloading images increases the speed with which the TT3100 Series terminals operate. Fetching images from memory is faster than downloading them from a host.

EXAMPLE

```
V.A.PreLoadImagesForms.Add padScreenWelcome
```

When a preload command executes, it loads ***all*** the images on the form into memory. In the example, the command PreLoadingImagesForms command loads the images on the form called **padScreenWelcome**.

Note: Images on a form cannot be selectively loaded. This event triggers before the splash screen is displayed.

A_PreSplashScreen Event

A_PreSplashScreen fires before a splash screen is displayed on the interactive point of sale screen.

A_PreTerminate Event

A_PreTerminate fires just before normal termination of a TT Advantage application. With the **A_PreTerminate** command, databases and files can be closed and objects freed up.

A_PreUnloadSplashScreen Event

A_PreUnloadSplashScreen fires just before the splash screen is removed. Place code in the event subroutine, to connect to databases, initialize variables, etc. With this line, status line displaying text information on the Splash screen, to reflect the activities of the code, can be implemented.

Error Messages

VB Mode Error Messages

TT Advantage provides error messages that depend on the TT Advantage mode. In VB mode the TT Advantage error messages display in the same way as Visual Basic error messages.

Error/Warning	Occurred while	Description	Remedy
TT Advantage will not be able to display this control	Putting new controls on a TT Advantage form	This control is not a control that TT Advantage can display on the screen.	This is just a warning. In VB Mode many controls do not display on a TT3100 Series screen, such as RDO controls, Timer, etc.
Timeout occurred	Downloading Script	The device might have been disconnected while downloading.	Check the connection of the RS232 cable.

Script Mode Error Messages

Script mode error messages display in the TT Advantage Assistant message box.

Error/Warning	Occurred while	Description	Remedy
Port error occurred	Downloading Script	This message usually means that the TT3100 Series terminal is not connected or VA is configured to use the wrong port.	Check the connection of the RS-232 cable and verify the PORT setting in modMain.
Timeout occurred	Downloading Script	The TT3100 Series terminal might have been disconnected while downloading.	Check the connections of the RS232 cable.
TT Advantage will not be able to display this control.	Putting new control on a VA form	This control is not a control that TT Advantage can display on the interactive point of sale screen.	This is just a warning. See page 3-2 to see a list of valid Script mode controls.
Only TT Advantage screens can be added…	Adding a new form to your project	In script mode, only TT Advantage forms, padScreens, can be used.	When adding a new form, be sure to add only TT Advantage forms. (See TT Advantage Forms on page 2-5.)
Only Modules and TT Advantage screens…	Adding a new component to your project	In script mode, only TT Advantage screens and normal modules can be added to a project.	See previous remedy.
padStart is a Fundamental component to…	Removing pad-Start form	The padStart form is a required form in script mode projects.	Re-attach the padStart form.

This form is not a valid Script Designer Form	Adding a form	The size of the form is incorrect. TT Advantage requires a form of a specific size.	If you must use this offending form, then change these properties for that form:
			ScaleWidth = 320
			ScaleHeight = 240
			Be sure you have the ScriptDe-scriber.txt file and it is not cor-rupted. Be sure it is located in the directory with mxVisual.dll

This is not a valid label. It is not the name of a valid label or routine	Trying to down-load script	The indicated parameter is not a label.	Use the “Show Labels” button to see a list of valid labels. Labels can only be screen or routine names.

Error/Warning	Occurred while	Description	Remedy
ERROR!: Script command PARAMETER:… is not a valid command.	Trying to down-load script	The error message indicates a script command entered as a parameter to another command is incorrect.	Be sure that ***every*** Script mode command begins with the command prefix (see Command Prefixes on page 3-10). For example, every command must begin with “TOOL”, even commands that are inside other commands, like: TOOL.Script.IfTrue _ Tool.Var.Find(“NAME”), _ Tool.Script.Bell(Success), _ Tool.Script.Bell(Fail)
ERROR!:Script command PARAMETER:… an invalid parameter type was found in the describer file.	Trying to down-load script	The ScriptDescriber.txt file might be corrupted.	Be sure the ScriptDescriber.txt file is present, is ***located*** in the directory with mxVisual.dll and is not corrupted.
ERROR!: Script command:… is not a valid com-mand	Trying to down-load script	The indicated command is not a valid script command.	Check the manual for the correct spelling and syntax of the indicated command.

Help Desk

If you need assistance installing or troubleshooting your software, please call your Distributor or the nearest Hand Held Products technical support office:

North America:

Telephone:	(315) 685-2476 (8 a.m. to 6 p.m. EST)
or, in the U.S.:	(800) 782-4263
Fax number:	(315) 685-4960

E-mail: support@handheld.com

Europe:

Telephone-

European Ofc:	Int+31 40 242 4486
U.K. Ofc:	Int+44 1925 240055

E-mail: support@handheld.com

Asia:

Telephone:	Int+852-2511-3050 *or* 2511-3132

E-mail: support@handheld.com

HAND HELD PRODUCTS®
a WelchAllyn® affiliate

4619 Jordan Road
P.O. Box 187
Skaneateles Falls, New York 13153-0187

TT Advantage/PG Rev A