Compsee

# M.A.T

## Mobile Application Terminal


## Custom Controls
## Reference Guide

***Version 2.0.0***

# Table of Contents

# Introduction

This document provides design details for Compsee's Mobile Application Terminal Custom Controls. Compsee's M.A.T. Controls are distributed as part of the Mobile Application Terminal Developer's Tool Kit and are available from the Visual Studio .NET IDE.

# M.A.T Toolbox

Once the Custom Controls are installed, the following group of tools is available from the toolbox. To take advantage of these controls, simply select the control from the toolbox by dragging and dropping it onto a windows form. Set the appropriate properties and call the appropriate methods.


# Compsee Controls

The following sections of this Guide discuss each custom control and its components.

## *Audio Control*

File Name: Compsee.Controls.Audio.dll
Version 2.0.0.0

The Audio Control Class Library provides access to the mobile device Microphone. It also allows playback of recorded sound files (wav).

**Components Table**

| Section | Property | Description |
|---|---|---|
| Constructs | Recorder | Initializes a new instance of Recorder Class |
| Constructs | Player | Initializes a new instance of Player Class |
| Property | Recorder.Recording | True if the recorder is presently recording |
| Property | Player.Playing | True if the player is currently playing |
| Property | Player.Volume | Gets or sets playback volume |
| Method | Recorder.RecordFor() | Record data for a specified number of seconds |
| Method | Recorder.Stop() | Stop recording currently in progress |
| Method | Player.Pause() | Pause play |
| Method | Player.Play() | Plays waveform in the given stream |
| Method | Player.Restart() | Restart a paused wave file |
| Method | Player.Stop() | Stop play |
| Event | Recorder.DoneRecording | Event fired when recording is stopped |
| Event | Recorder.WaveClose | Event fired when wave device is closed |
| Event | Recorder.WaveOpen | Event fired when wave device is opened |
| Event | Player.DonePlaying | Event fired when device has finished playback |
| Event | Player.WaveClose | Event fired when the wave device is closed |
| Event | Player.WaveOpen | Event fired when wave device is opened |

**Descriptions**

- **Recorder Constructor**
  Creates Recorder object and attaches it to the default wave device
  public Recorder();

- **Player Constructor**
  Constructs Player object on the default wave device
  public Player();

- **Recorder.Recording Property**
  True, if the Recorder is currently recording.
  [Visual Basic]
  Public ReadOnly Property Recording As Boolean

- **Player.Playing Property**
  True, if the Player is currently playing. False otherwise
  [Visual Basic]
  Public ReadOnly Property Playing As Boolean

- **Payer.Volume Property**
  Gets or sets playback volume on the current wave device
  [Visual Basic]
  Public Property Volume As Integer

- **Recorder.RecordFor Method**
  Record sound data for a specified number of seconds at 11025 sps, and 1 channel. The stream will be a properly formatted RIFF file.
  public void RecordFor(Stream,short);

- **Recorder.Stop Method**
  Stop recording operation currently in progress. Throws an error if no recording operation is in progress
  [Visual Basic]
  Public Sub Stop()

- **Player.Pause Method**
  Pause Play
  [Visual Basic]
  Public Sub Pause()

- **Player.Play Method**
  Plays waveform contained in the given stream. Stream is expected to contain full riff header
  [Visual Basic]
  Public Sub Play( ByVal *playStream* As Stream )

- **Player.Restart Method**
  Restart a paused wave file
  [Visual Basic]
  Public Sub Restart()

- **Player.Stop Method**
  Stop Play
  [Visual Basic]
  Public Sub Stop()

- **Recorder.DoneRecording Event**
  Handles the event that is fired when recording is stopped (on timer or by calling Stop method
  [Visual Basic]
  Public Event DoneRecording As WaveFinishedHandler

- **Recorder.WaveClose Event**
  Handles the event that is fired when wave device is successfully closed
  [Visual Basic]
  Public Event WaveClose As WaveCloseHandler

- **Recorder.WaveOpen Event**
  Handles the event that is fired when wave device is successfully opened
  [Visual Basic]
  Public Event WaveOpen As WaveOpenHandler

- **Player.DonePlaying Event**
  Raised when the wave device has finished playback
  [Visual Basic]
  Public Event DonePlaying As WaveDoneHandler

- **Player.WaveClose Event**
  Raised when the wave device is closed
  [Visual Basic]
  Public Event WaveClose As WaveCloseHandler

- **Player.WaveOpen Event**
  Raised when the wave device is opened
  [Visual Basic]
  Public Event WaveOpen As WaveOpenHandler

**Sample Reference**

Refer to the Audio Utility sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# Battery Status

File Name: BatteryStat.dll
Version 2.0.0.0

The Battery Status Custom Control retrieves status information from the mobile device power source.

**Components Table**

| Section | Property/Method | Description |
|---|---|---|
| ***Main Battery*** | | |
| Property | .BatteryAvgCurrent | Battery Average Current in milliAmps (1/1000) |
| Property | .BatteryAvgCurrentTime | Battery Average Current Time Constant in milliSeconds (1/1000) |
| Property | .BatteryChargeStatus | Battery Charge Status |
| Property | .BatteryChemistry | Battery Type (Alkaline,Nicad, etc.) |
| Property | .BatteryCurrent | Battery Current in milliAmps (1/1000) |
| Property | .BatteryLifeFull | Battery Full Charge Life (seconds) |
| Property | .BatteryLifeRemaining | Battery Life Remaining (seconds) |
| Property | .BatteryMAmpHour | Battery Discharge in milliAmp Hours (mAH) |
| Property | .BatteryPercent | Battery Percent of Full Charge |
| Property | .BatteryTemperature | Battery Temperature in .1 C |
| Property | .BatteryVoltage | Battery Voltage in milliVolts (1/1000 v) |
| ***Backup Battery*** | | |
| Property | .BackupBatteryVoltage | Backup Battery voltage in millivolts |
| Property | .BackupBatteryChargeStatus | Backup Battery Charge Status |
| Property | .BackupBatteryPercent | Backup Battery Percent of Full Charge |
| Property | .BackupBatteryLifeRemaining | Backup Battery Life Remaining in seconds |
| Property | .BackupBatteryLifeFull | Backup Battery Charge Life in seconds |
| ***AC Plug*** | | |
| Property | .ACLine | AC Line Status |
| | | |
| Method | .GetStatus | Populates Properties with Battery Info (now) |

**Descriptions**

- **.ACLine [Byte]**
  AC Power Status. It has one of the following values:
  - AC_LINE_OFFLINE = 0
  - AC_LINE_ONLINE = 1
  - AC_LINE_BACKUP_POWER = 2
  - AC_LINE_UNKNOWN = 255

- **.BatteryChargeStatus [Byte]**
  Battery charge status. It is a combination one of the following values:
  - HIGH = 1
  - LOW = 2
  - CRITICAL = 4
  - CHARGING = 8
  - NO_BATTERY = 128
  - UNKNOWN = 255 (-1)

- **.BatteryPercent [Byte]**
  Percentage of full battery charge remaining. The number must be in the range of 0 to 100, or 255 if percentage of battery life remaining is unknown.

- **.BatteryLifeRemaining [String]**
  Number of seconds of battery life remaining or ffffffff (hex) if the number of seconds of battery life remaining is unknown.

- **.BatteryLifeFull [String]**
  Number of seconds of battery life when at full charge or ffffffff (hex) if full lifetime of battery is unknown.

- **.BatteryVoltage [Integer]**
  Number of millivolts (mV) of battery voltage. The number can range from 0 to 65535.

- **.BatteryCurrent [Integer]**
  Number of milliamps (mA) of instantaneous current drain. The number can range from 0 to 32767 for charge and 0 to –32768 for discharge.

- **.BatteryAvgCurrent [Integer]**
  Average number of milliamps of short term device current drain. The number can range from 0 to 32767 for charge and 0 to –32768 for discharge.

- **.BatteryAverageCurrentTime [Integer]**
  Number of milliseconds (mS) that is the time constant interval used in reporting BatteryAverageCurrent.

- **.BatteryMAmpHour [Integer]**
  Average number of milliamp hours (mAh) of long-term cumulative average discharge. It can range from 0 to –32768. This value is reset when the batteries are charged or changed.

- **.BatteryTemperature [Float, single precision]**
  Battery temperature reported in 0.1 degree Celsius increments. It can range from –3276.8 to 3276.7.

- **.BatteryChemistry [Byte]**
  Type of battery. It can be one of the following values:
  - ALKALINE = 1
  - NICKEL CADMIUM = 2
  - NICKEL METAL HYDRIDE = 3
  - LITHIUM ION = 4
  - LITHIUM POLYMER = 5
  - ZINC AIR = 6
  - BATTERY UNKNOWN = 255

- **.BackupBatteryChargeStatus [Byte]**
  Battery charge status. A combination one of the following values:
  - HIGH = 1
  - LOW = 2
  - CRITICAL = 4
  - CHARGING = 8
  - NO_BATTERY = 128
  - UNKNOWN = 255 (-1)

- **.BackupBatteryPercent [Byte]**
  Percentage of full battery charge remaining. Must be in the range 0 to 100, or 255 if the percentage of battery life remaining is unknown.

- **.BackupBatteryLifeRemaining [String]**
  Number of seconds of battery life remaining, or ffffffff (hex) if number of seconds of battery life remaining is unknown.

- **.BackupBatteryLifeFull [String]**
  Number of seconds of battery life when at full charge, or ffffffff (hex) if full lifetime of battery is unknown.

- **.BackupBatteryVoltage [Integer]**
  Number of millivolts (mV) of battery voltage. It can range from 0 to 65535.

**Methods**

- **.GetStatus [Returns Integer 1= OK, 0 = Error]**
  Call this function to get Battery Information that is current (now).
  This function will populate the Properties.

**Sample Reference**

Refer to the BatteryStat sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# Brightness

File Name: Brightness.dll
Version: 2.0.0.0

The Brightness Custom Control provides access to the LCD backlight brightness setting of the mobile device.

### Components Table

| Section | Property/Method | Description |
|---|---|---|
| Property | .LcdReadBrightness | Gets the LCD Backlight Brightness 0 to 100 (%) |
| Property | .LcdSetValue | Value used to set LCD Backlight Brightness 0 to 100 (%) |
| | | |
| Method | .SetBrightness | Sets the LCD Backlight Brightness of the Hardware |

### Properties

- **.LcdReadBrightness**
  Read this property to get the current LCD backlight brightness value. Value is 0 to 100.

- **.LcdSetValue**
  Set the value of this property to adjust the LCD backlight brightness. The value is used in combination with the .SetBrightness method. Value is 0 to 100.

### Methods

- **.SetBrightness**
  Call this method to set the mobile device LCD backlight brightness to the new brightness value defined by the .LcdSetValue property.

### Sample Reference

Refer to the LCD Brightness sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# Keypad Remap

File Name: KeypadRemap.dll
Version 2.0.0.0

The Keypad Remap Custom Control is used to map a key on the keypad, to a different virtual key code.

### Components Table

| Section | Property | Description |
|---|---|---|
| Property | .Keypad_A | Keycode for Keypad Key A |
| Property | .Keypad_B | Keycode for Keypad Key B |
| Property | .Keypad_C | Keycode for Keypad Key C |
| Property | .Keypad_D | Keycode for Keypad Key D |
| Property | .Keypad_E | Keycode for Keypad Key E |
| Property | .Keypad_F | Keycode for Keypad Key F |
| Property | .Keypad_G | Keycode for Keypad Key G |
| Property | .Keypad_H | Keycode for Keypad Key H |
| Property | .Keypad_I | Keycode for Keypad Key I |
| Property | .Keypad_J | Keycode for Keypad Key J |
| Property | .Keypad_K | Keycode for Keypad Key K |
| Property | .Keypad_L | Keycode for Keypad Key L |
| Property | .Keypad_M | Keycode for Keypad Key M |
| Property | .Keypad_N | Keycode for Keypad Key N |
| Property | .Keypad_O | Keycode for Keypad Key O |
| Property | .Keypad_P | Keycode for Keypad Key P |
| Property | .Keypad_Q | Keycode for Keypad Key Q |
| Property | .Keypad_R | Keycode for Keypad Key R |
| Property | .Keypad_S | Keycode for Keypad Key S |
| Property | .Keypad_T | Keycode for Keypad Key T |
| Property | .Keypad_U | Keycode for Keypad Key U |
| Property | .Keypad_V | Keycode for Keypad Key V |
| Property | .Keypad_W | Keycode for Keypad Key W |
| Property | .Keypad_X | Keycode for Keypad Key X |
| Property | .Keypad_Y | Keycode for Keypad Key Y |
| Property | .Keypad_Z | Keycode for Keypad Key Z |

| Section | Property | Description |
|---|---|---|
| Property | .Keypad_0 | Keycode for Keypad Key 0 |
| Property | .Keypad_1 | Keycode for Keypad Key 1 |
| Property | .Keypad_2 | Keycode for Keypad Key 2 |
| Property | .Keypad_3 | Keycode for Keypad Key 3 |
| Property | .Keypad_4 | Keycode for Keypad Key 4 |
| Property | .Keypad_5 | Keycode for Keypad Key 5 |
| Property | .Keypad_6 | Keycode for Keypad Key 6 |
| Property | .Keypad_7 | Keycode for Keypad Key 7 |
| Property | .Keypad_8 | Keycode for Keypad Key 8 |
| Property | .Keypad_9 | Keycode for Keypad Key 9 |
| Property | .Keypad_BACKSPACE | Keycode for Keypad Key Bksp |
| Property | .Keypad_NUMLOCK | Keycode for Keypad Key NumLck |
| Property | .Keypad_RIGHT_ENTER | Keycode for Keypad Key Rt. Entr |
| Property | .Keypad_LEFT_ENTER | Keycode for Keypad Key Lft. Entr |
| Property | .Keypad_LeftSCAN | Keycode for Keypad Key Lft Scan |
| Property | .Keypad_RightSCAN | Keycode for Keypad Key Rt. Scan |
| Property | .Keypad_TRIGGER | Keycode for Keypad Key Trigger |
| Property | .Keypad_PERIOD | Keycode for Keypad Key Period |

**Methods**

- **.SetRemapKey()**
  Call this method to set the changes made to the above key properties into the Keypad controller.

- **.SetKeypadDefaults()**
  Call this method to reset to the default keycodes in the Keypad controller.

- **.GetCurrentKeys()**
  Call this method to populate the properties with the current keypad keycodes.

**Virtual Keycodes**

Use these keycodes to set the above properties to remap the keys.

```
byte VK_LBUTTON        = 0x01;
byte VK_RBUTTON        = 0x02;
byte VK_CANCEL         = 0x03;
byte VK_MBUTTON        = 0x04;
byte VK_BACK           = 0x08;
byte VK_TAB            = 0x09;
byte VK_CLEAR          = 0x0C;
byte VK_RETURN         = 0x0D;
byte VK_SHIFT          = 0x10;
byte VK_CONTROL        = 0x11;
byte VK_MENU           = 0x12;
byte VK_PAUSE          = 0x13;
byte VK_CAPITAL        = 0x14;
byte VK_KANA           = 0x15;
byte VK_JUNJA          = 0x17;
byte VK_FINAL          = 0x18;
byte VK_HANJA          = 0x19;
byte VK_KANJI          = 0x19;
byte VK_ESCAPE         = 0x1B;
byte VK_CONVERT        = 0x1c;
byte VK_NOCONVERT      = 0x1d;
byte VK_SPACE          = 0x20;
byte VK_PRIOR          = 0x21;
byte VK_NEXT           = 0x22;
byte VK_END            = 0x23;
byte VK_HOME           = 0x24;
byte VK_LEFT           = 0x25;
byte VK_UP             = 0x26;
byte VK_RIGHT          = 0x27;
byte VK_DOWN           = 0x28;
byte VK_SELECT         = 0x29;
byte VK_PRbyte         = 0x2A;
byte VK_EXECUTE        = 0x2B;
byte VK_SNAPSHOT       = 0x2C;
byte VK_INSERT         = 0x2D;
byte VK_DELETE         = 0x2E;
byte VK_HELP           = 0x2F;
```

```
byte VK_0               = 0x30;
byte VK_1               = 0x31;
byte VK_2               = 0x32;
byte VK_3               = 0x33;
byte VK_4               = 0x34;
byte VK_5               = 0x35;
byte VK_6               = 0x36;
byte VK_7               = 0x37;
byte VK_8               = 0x38;
byte VK_9               = 0x39;

byte VK_A               = 0x41;
byte VK_B               = 0x42;
byte VK_C               = 0x43;
byte VK_D               = 0x44;
byte VK_E               = 0x45;
byte VK_F               = 0x46;
byte VK_G               = 0x47;
byte VK_H               = 0x48;
byte VK_I               = 0x49;
byte VK_J               = 0x4A;
byte VK_K               = 0x4B;
byte VK_L               = 0x4C;
byte VK_M               = 0x4D;
byte VK_N               = 0x4E;
byte VK_O               = 0x4F;
byte VK_P               = 0x50;

byte VK_Q               = 0x51;
byte VK_R               = 0x52;
byte VK_S               = 0x53;
byte VK_T               = 0x54;
byte VK_U               = 0x55;
byte VK_V               = 0x56;
byte VK_W               = 0x57;
byte VK_X               = 0x58;
byte VK_Y               = 0x59;
byte VK_Z               = 0x5A;

byte VK_LWIN            = 0x5B;
byte VK_RWIN            = 0x5C;
byte VK_APPS            = 0x5D;
byte VK_SLEEP           = 0x5F;
```

```
byte VK_NUMPAD0         = 0x60;
byte VK_NUMPAD1         = 0x61;
byte VK_NUMPAD2         = 0x62;
byte VK_NUMPAD3         = 0x63;
byte VK_NUMPAD4         = 0x64;
byte VK_NUMPAD5         = 0x65;
byte VK_NUMPAD6         = 0x66;
byte VK_NUMPAD7         = 0x67;
byte VK_NUMPAD8         = 0x68;
byte VK_NUMPAD9         = 0x69;
byte VK_MULTIPLY        = 0x6A;
byte VK_ADD             = 0x6B;
byte VK_SEPARATOR       = 0x6C;
byte VK_SUBTRACT        = 0x6D;
byte VK_DECIMAL         = 0x6E;
byte VK_DIVIDE          = 0x6F;
byte VK_F1              = 0x70;
byte VK_F2              = 0x71;
byte VK_F3              = 0x72;
byte VK_F4              = 0x73;
byte VK_F5              = 0x74;
byte VK_F6              = 0x75;
byte VK_F7              = 0x76;
byte VK_F8              = 0x77;
byte VK_F9              = 0x78;
byte VK_F10             = 0x79;

byte VK_F11             = 0x7A;
byte VK_F12             = 0x7B;
byte VK_F13             = 0x7C;
byte VK_F14             = 0x7D;
byte VK_F15             = 0x7E;
byte VK_F16             = 0x7F;
byte VK_F17             = 0x80;
byte VK_F18             = 0x81;
byte VK_F19             = 0x82;
byte VK_F20             = 0x83;
byte VK_F21             = 0x84;
byte VK_F22             = 0x85;
byte VK_F23             = 0x86;
byte VK_F24             = 0x87;
byte VK_NUMLOCK         = 0x90;
byte VK_SCROLL          = 0x91;
```

```
byte VK_HYPHEN          = 0xBD;
byte VK_PERIOD          = 0xBE;
byte VK_SLASH           = 0xBF;
byte VK_COMMA           = 0xDC;

byte VK_L_SCAN          = 0xE9;
byte VK_R_SCAN          = 0xEA;
byte VK_C_SCAN          = 0xEB;
byte VK_TRIGGER         = 0xEC;
byte VK_SOFT_SCAN       = 0xED;

byte VK_MUTE            = 0xF1;
byte VK_BACKLIGHT       = 0xF2;
byte VK_LCD_UP          = 0xF3;
byte VK_LCD_DOWN        = 0xF4;
```

**Sample Reference**

Refer to the KeyPad Remap sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# Key Send

File Name: KeySend.dll
Version 2.0.0.0

The KeySend transmits a virtual key code to Windows.

### Components Table

| Section | Property/Method | Description |
|---|---|---|
| Property | .VirtualKey | Virtual Key Code to Simulate |
| | | |
| Method | .PressVirtualKey | Simulate a Key Press |
| Method | .ReleaseVirtualKey | Simulate a Key Release |

### Properties

- **.VirtualKey**
  Set this property to the virtual key code you want sent to Windows.

### Methods

- **.PressVirtualKey**
  Call this function to send the VirtualKey to Windows. Acts like a user pressed a key.

- **.ReleaseVirtualKey**
  Call this function to send the VirtualKey to Windows. Acts like a user released a key.

### Sample Reference

Refer to the Scan Demo sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# LEDs

File Name: Leds.dll
Version 2.0.0.0

The LEDs Custom Control provides access to the Decode LED on the device.

### Components Table

| Section | Property/Method | Description |
|---|---|---|
| Property | .BlinkOnTime | Blink On Time |
| Property | .BlinkOffTime | Blink Off Time |
| Property | .NumberOnBlinks | Number of ON Blinks |
| Property | .NumberOffBlinks | Number of OFF Blinks |
| | | |
| Method | .Led_CAPSLock_Toggle() | Toggles Caps Lock LED (and Caps State) |
| Method | .Led_NAVLock_Toggle() | Toggles NAV Lock LED (and Nav State) |
| Method | .Led_Decode_On() | Turns On Decode LED Continuously |
| Method | .Led_Decode_Off() | Turns Off Decode LED |
| Method | .Led_Decode_Blink() | Blinks Decode LED Continuously |

### Properties

- .**BlinkOnTime (Integer)**
  Blink On time in microseconds.

- .**BlinkOffTime (Integer)**
  Blink Off time in microseconds.

- **.NumberOnBlinks (Integer)**
  Number of ON blinks.

- **.NumberOffBlinks (Integer)**
  Number of OFF blinks.

**Methods**

- **.Led_CAPSLock_Toggle**
  Call this function to toggle Caps Lock LED (and Caps State) between On/Off State.

- **.Led_NAVLock_Toggle**
  Call this function to toggles NAV Lock LED (and Nav State) between On/Off State.

- **.Led_Decode_On**
  Call this function to turn On Decode LED Continuously.

- **.Led_Decode_Off**
  Call this function to turn Off Decode LED.

- **.Led_Decode_Blink**
  Call this function to blink Decode LED Continuously at 1 sec on and 1 sec off.

**Sample Reference**
Refer to the LED Test sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# MultiApp

File Name: MultiApp.dll
Version 2.0.0.0

The MultiApp Custom Control is used to check the run status of an application on the device.

### Components Table

| Section | Property/Method | Description |
|---|---|---|
| Property | .FindResult | Status of the FindAppWindow. |
| | | |
| Method | .FindAppWindow (AppTitle) | Check the run status of an Application. |
| Method | .ShowAppWindow () | Sets focus to the Application Window |

### Properties

- .**FindResult (Boolean)**
  FindResult returns a TRUE if the FindAppWindow() found the application and a FALSE if the application was not currently running on the device.

### Methods

- **.FindAppWindow(AppTitle as String)**
  Call this function to check the run status of an application. The AppTitle is the name of the Application Window Title to search.
- **.ShowAppWindow()**
  Call this function to set the focus to the application.

### Sample Reference

Refer to the Scan Demo sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# Signature Capture

File Name: SignatureCapture.dll
Version 2.0.0.0

The Signature Capture Custom Control will follow a pen signature on a display box to be saved as a bitmap file.

### Components Table

| Section | Property | Description |
|---|---|---|
| Property | .BitmapPath | Sets the bitmap file path and name |
| Property | .SignText | Sets the “sign here” to a custom text |
| | | |
| Method | .Reset | Resets and clears the Signature Capture box. |
| Method | .SaveBitmap | Saves the Signature to a bitmap file. |

### Methods

- **.Reset**

Resets and clears the Signature Capture box.

- **.SaveBitmap**

Saves the Signature to a bitmap file with the name and path set in the .BitmapPath property.

### Sample Reference

Refer to the Signature Capture sample, installed with the Mobile Application Terminal Developer’s Kit, to view sample code for this control.

## *SystemParms*

File Name: SystemParms.dll
Version 2.0.0.0

The SystemParms Custom Control provides access to the system and platform information on the mobile device.

**Components Table**

| Section | Property/Method | Description |
|---|---|---|
| ***OEM Info*** | | |
| Property | .CEBuildDate | OS Build Date |
| Property | .CpldDate | CPLD Date |
| Property | .CpldTime | CPLD Time |
| Property | .EbootMajor | Boot Loader Major Version |
| Property | .EbootMinor | Boot Loader Minor Version |
| Property | .KeyPadId | Keypad Id |
| Property | .KeyPadStr | Keypad Type |
| Property | .TotalFlash | Total Flash |
| Property | .FBank0 | Flash Bank 0 |
| Property | .FBank1 | Flash Bank 1 |
| Property | .FBank2 | Flash Bank 2 |
| Property | .TotalRam | Total RAM |
| Property | .RBank0 | RAM Bank 0 |
| Property | .RBank1 | RAM Bank 1 |
| Property | .OEMString1 | OEM Supplied String |
| Property | .OEMString2 | OEM Supplied String |
| Property | .PlatformString | Platform String |
| Property | .UnitIdString | Unit Id |
| | | |
| Method | .GetOEMInfo() | Retrieves the Device Specific information. |
| ***Platform Info*** | | |
| Property | .PlatformType | Platform Type |
| | | |
| Method | .GetPlatformType() | Retrieves the Platform Type. |

**Properties**

- .**CEBuildDate**
  OS Build Date returned by the .GetOEMInfo method.

- .**CpldDate**
  CPLD Date returned by the .GetOEMInfo method.

- .**CpldTime**
  CPLD Time returned by the .GetOEMInfo method.

- .**EbootMajor**
  Eboot loader Major Version returned by the .GetOEMInfo method.
- .**EbootMinor**
  Eboot loader Minor Version returned by the .GetOEMInfo method.

- .**KeyPadId**
  Keypad Id returned by the .GetOEMInfo method.

- .**KeyPadStr**
  Keypad Type returned by the .GetOEMInfo method.

- .**TotalFlash**
  Total Flash available on the device returned by the .GetOEMInfo method.

- .**FBank0**
  Total Flash in Flash Bank0 returned by the .GetOEMInfo method.

- .**FBank1**
  Total Flash in Flash Bank1 returned by the .GetOEMInfo method.

- . **FBank2**
  Total Flash in Flash Bank2 returned by the .GetOEMInfo method.

- .**TotalRam**
  Total RAM available on the device returned by the .GetOEMInfo method.

- .**RBank0**
  Total RAM in RAM Bank0 returned by the .GetOEMInfo method.

- .**RBank1**
  Total RAM in RAM Bank1 returned by the .GetOEMInfo method.

- .**OEMString1**
  First OEM String identifying the device returned by the .GetOEMInfo method.

- .**OEMString2**
  Second OEM String identifying the device returned by the .GetOEMInfo method...

- .**PlatformString**
  Platform String identifying the device returned by the .GetOEMInfo method.

- .**PlatformType**
  Platform Type of the device returned by the .GetPlatformType method.

- .**UnitIdString**
  Unit Id of this specific device returned by the .GetOEMInfo method.

**Methods**

- .**GetOEMInfo**
  Gets the OEM Information and populates the corresponding properties.

- .**GetPlatformType**
  Gets the Platform Type information and populates the corresponding properties.

**Sample Reference**

Refer to the Scan Demo sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

## *Vibrate*

File Name: Vibrate.dll
Version 2.0.0.0

The Vibrate Custom Control provides access to the vibrator mechanism in the mobile device.

**Components Table**

| Section | Property | Description |
|---|---|---|
| Property | .PulseOnTime | Pulse On Time |
| Property | .PulseOffTime | Pulse Off Time |
| Property | .NumberOnCycles | Number of ON Blink Cycles |
| Property | .NumberOffCycles | Number of OFF Blink Cycles |
| | | |
| Method | .Vibrator_On | Turns On Device Vibrator for a set period |
| Method | .Vibrator_Off | Turn Off Device Vibrator. |
| Method | .Vibrator_Pulse | Pulses Vibrator On and Off Continuously. |

**Methods**

- **.Vibrate_On**
  Call this function to turn On device Vibrator for a set period.

- **.Vibrate_Off**
  Call this function to turn Off device Vibrator.

- **.Vibrate_Pulse**
  Call this function to Pulse device Vibrator On and Off continuously at 1 sec on and 1 sec off.

**Properties**

- **.PulseOnTime (Integer)**
  Pulse On time in microseconds.

- **.PulseOffTime (Integer)**
  Pulse Off time in microseconds.

- **.NumberOnCycles (Integer)**
  Number of ON blink cycles.

- **.NumberOffCycles (Integer)**
  Number of OFF blink cycles.

**Sample Reference**

Refer to the VibratorTest sample, installed with the Mobile Application Terminal Developer's Kit, to view sample code for this control.

# Technical Support

Technical Support on the MAT 203/204 is available from Compsee, Inc. through the following methods:

Via our Website:  www.compsee.com
E-mail:  support@compsee.com
Phone:  1 (800) 628-3888
Before calling please follow these guidelines:

- Have a complete description of the problem as well as any pertinent information on when it occurred.

# Software:  End User License Agreement (EULA)

You have acquired a device (“DEVICE”) that includes software licensed by Compsee from an affiliate of Microsoft Corporation (“MS”).  Those installed software products of MS origin, as well as associated media, printed materials and “online” or electronic documentation (“Software”) are protected by international intellectual property laws and treaties.  Manufacturer, MS and its supplies (including Microsoft Corporation) own the title, copyright, and other intellectual property rights in the SOFTWARE.  The SOFTWARE is licensed, not sold.  All rights reserved.

The EULA is valid and grants the end-user rights ONLY if the SOFTWARE is genuine and a genuine Certificate of Authenticity for the SOFTWARE is included.  For more information on identifying whether your software is genuine, please see http://www.microsoft.com/piracy/howtotell.

**IF YOU DO NOT AGREE TO THIS END USER LICENSE AGREEMENT (“EULA”), DO NOT USE THE DEVICE OR COPY THE SOFTWARE.  INSTEAD, PROMPLY CONTACT COMPSEE FOR INSTRUCTIONS ON RETURN OF THE UNUSED DEVICE(S) FOR A REFUND.  ANY USE OF THE SOFTWARE, INCLUDING BUT NOT LIMITED TO USE ON THE DEVICE, WILL CONSTITUTE YOUR AGREEMENT TO THE EULA (OR RATIFICATION OF ANY PREVIOUS CONSENT).**

**GRANT OF SOFTWARE LICENSE**.  This EULA grants you the following license:

- You may use the SOFTWARE only on the DEVICE.

- **Restricted Functionality.**  You are licensed to use the SOFTWARE to provide only the limited functionality (specific tasks or processes) for which the DEVICE has been designed and marketed by Compsee.  This license specifically prohibits any other use of the software programs or functions, or inclusions of additional software programs or functions that do not directly support the limited functionality on the DECOVE.  Notwithstanding the foregoing, you may install or enable on a DEVICE, systems utilities, resource management or similar software solely for the purpose of administration, performance enhancement and/or preventive maintenance of the DEVICE.

- If you use the DEVICE to access or utilize the services or functionality of Microsoft Windows Server products (such as Microsoft Windows Server 2003), or use the DEVICE to permit workstations or computing devices to access or utilize the services or functionality of Microsoft Windows Server products, you may be required to obtain a Client Access License for the DEVICE and/or each such workstation or computing device.  Please refer to the end user license agreement for you Microsoft Windows Server product for additional information.

- NOT FAULT TOLERANT.  THE SOFTWARE IS NOT FAULT TOLERANT.  COMPSEE HAS INDEPENDENTLY DETEREMINED HOW TO USE THE

SOFTWARE IN THE DEVICE, AND MS HAS RELIED UPON COMPSEE TO CONDUCT SUFFICIENT TESTING TO DETERMINE THAT THE SOFTWARE IS SUITABLE FOR SUCH USE.

- **NO WARRANTIES FOR THE SOFTWARE.** The SOFTWARE is provided "AS IS" and with all faults. THE ENTRIRE RISK AS TO SATISFACTORY QUALITY PERFROMANCE, ACCURACY, AND EFFORT (INCLUDING LACK OF NEGLIGENCE) IS WITH YOU. ALSO, THERE IS NO WARRANTY AGAINST INTERFERENCE WITH YOUR ENJOYMENT OF THE SOFTWARE OR AGAINST INFRINGEMENT. **IF YOU HAVE RECEIVED ANY WARRANTIES REGARDING THE DEVICE OR THE SOFTWARE, THOSE WARRANTIED DO NOT ORIGINATE FROM, AN ARE NOT BINDING ON, MS.**

- No liability for Certain Damages. EXCEPT, AS PROHIBITED BY LAW, MS SHALL HAVE NO LIABILITY FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL OR INCIDENTAL DAMAGES ARISING FROM OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SOFTWARE. THIS LIMITATION SHALL APPLY EVEN IF ANY REMEDY FAILS OF ITS ESSENTIAL PURPOSE. IN NO EVENT SHALL MS BE LIABLE FOR ANY AMOUNT IN EXCESS OF U.S. TWO HUNDRED FIFTY DOLLARS (U.S. $250.00).

- **Restricted Users.** The SOFTWARE is not designed or intended for use or resale in hazardous environments requiring fail-safe performance, such as n the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, or other devices or systems in which a malfunction of the SOFTWARE would result in foreseeable risk or injury or death to the operator of the device or system, or to others.