IMG Developer's Kit

Developer's Kit Guide and Technical Documentation

Version 1.90

User's Guide

IMG Developer's Kit: Developer's Kit Guide and Technical Documentation; Version 1.90; User's Guide

IMG Real World Press

179 Niblick Road #454
Paso Robles, CA 93446
1-800-889-0987 (US & Canada)
+1-818-701-1579
Website: http://www.imgpresents.com
To report errors, please send a note to ts@imgpresents.com
IMG Real World Press is a division of Innovation Management Group, Inc.
IMG Developer's Kit, Version 1.90, 5/31/2022

Production/Editing/Composition/Indexing/Publishing: IMG Real World Press

ISBN ???-?-???-?????-?

Table of Contents

Part I. Introduction

Introduction of the IMG Developer's Kit and Developer's Corner (for use on IMG's website).

Chapter 1 - Introduction contains welcome, release notes, file and folder details, and Frequently Asked Questions (FAQ).

Chapter 2 - Developer's Corner is the website interface into this entire collection.

Chapter 1. IMG Developer's Kit

Welcome to the IMG Developer's Kit

Welcome to the IMG Developer's Kit!

Innovation Management Group, Inc. has been producing on-screen keyboards and software utilities since 1995, and from the beginning we have been asked to control, manipulate, and perform unique and special tasks with our user interface tools. This Developer's Kit has grown from a handful of utilities to a comprehensive collection of software, examples, code, and documentation. Most of the developer utilities have come from specific customer requests, or expanding on themes from customer requested capabilities. As a Developer's Kit, it is focused on integrating on-screen keyboards with applications and controlling on-screen keyboards, as opposed to designing and developing on-screen keyboards themselves. Although the original My-T-Soft is flexible and customizable, IMG's Build-A-Board is the preferred tool for customizing and modifying the on-screen keyboard.

As the IMG's Developer's Kit collection has grown, it now covers many products, and addresses many different needs. To make managing, maintaining, and updating the Developer's Kit as straightforward as possible, all IMG products that have external developer type utilities have been merged into a single, inclusive IMG Developer's Kit. Please refer to the Release Notes below for further details on the history of this Developer's Kit.

As a quick overview, this is a collection of compiled utilities, DLLs, source code, and documentation to address many various requests and requirements brought to us by our customers. For a more detailed overview, see here. It has grown from a single folder of simple utilities (i.e. the DEVKIT folder), to a huge collection of very broad and very specific examples that covers multiple products & versions. We recommend you read through these first few sections before diving in, as it will address many issues and answer common questions for interested parties new to this material.

The collection has undergone numerous re-organizations, and will undoubtedly go through more. Please be aware that most of focus is on the My-T-Soft family of onscreen keyboards. If your interest is in a specific product that is not an onscreen keyboard, you will want to look for the specific area that addresses the product you are interested in, rather than review the entire kit.

> **Important:** Developer's Kit Maintenance - some of this material dates back to the mid-1990's, and ongoing maintenance and relevance to the current release is not always desirable (for example, we have customers on 20 year old releases). Additionally, technology rapidly changes, and the wants & needs of customers is dynamic, so maintaining old (and possibly not in demand) code is not the most efficient use of resources. Therefore, if a certain section seems out of date, or does not operate as expected, please contact IMG for guidance.

OnScreen, My-T-Mouse, My-T-Touch, My-T-Pen & My-T-Soft are part of the My-T-Soft family of products that provide onscreen keyboards, programmable macros, and other utility panels to provide an alternate approach to character entry in a Windows based system. The primary application is to replace the physical keyboard as the text input device. Because of the user-interface aspect (i.e. virtual keyboards) and the universal operation across platforms and within all Windows applications, the My-T-Soft family has uses only limited by the imagination of the developers that implement these tools.

Because of customization & third-party naming, the term "My-T-Soft" will be used throughout this document as a generic term for the software. The default directory for My-T-Mouse is \Program Files\MYTMOUSE, the default directory for My-T-Touch is \Program Files\MYTTOUCH, the default directory for My-T-Pen is \Program Files\MYTPEN, and the default directory for OnScreen is \Program Files\ONSCREEN.

The Developer's Kit is no longer automatically installed due to its size. It is included on the IMG Product Disc, and may be in the installation directory (\Program Files\[PRODUCT ID] e.g. \Program Files\MYTSOFT). The file may also be found on distribution media (DevKit or SDK folder) or on the IMG Website "http://www.imgpresents.com", in the Developer's Corner Download area (http://www.imgpresents.com/corner/imgdevd1.htm). For The Magnifier and Joystick-To-Mouse, you will need to simply download and unzip the Kit, and look for the appropriate folder.

For PDF versions of this document, use the links in the Developer's Corner Download area (http://www.imgpresents.com/corner/imgdevd1.htm). Refer to IMG's website for purchasing as a book.

Almost all of the notes & included source code and documentation assume the interested party has experience and is knowledgeable of the platforms & environments covered. Since the source code is present and available in many cases, this is the ultimate reference for any developer.

Innovation Management Group, Inc.
179 Niblick Road #454
Paso Robles, CA 93446
USA
1-800-889-0987 (US & Canada)
+1-818-701-1579
<cs@imgpresents.com>
http://www.imgpresents.com

Release Notes

Release Notes

Please note some of the resources available here & on-line:

- The most frequently asked questions & implementation of the frequent uses of the DevKit are on-line in our Developer's Corner at IMG's Developer's Corner (http://www.imgpresents.com/imgdev.htm). This document is also used as the on-line information, and a more current version may be available on-line. This is the link for the Developer's Corner within this document.
- For quick questions, feel free to e-mail us at devkit@imgpresents.com - We can quickly point you in the right direction, or give you a handful of options available.
- If you run into something you can't do, also e-mail us at devkit@imgpresents.com, or call and ask for developer support - we may have some nugget of information that can save you a lot of time & effort...

- IDE Notes: In general, we tend to use older IDEs (e.g. Microsoft Visual Studio 6 was used more often then .NET 2003, or Visual Studio 2005/2008/2010/2012/2013/2015/2017/2019/2022). This is done for the simple reason that you can move forward to newer IDEs without too much trouble, but trying to go backwards can be difficult (if not impossible). The other aspect is that Microsoft moves forward with little care towards backwards compatibility, and often it is easiest to work with the IDE concurrent with the oldest version of Windows supported. Because our customers just want the software to work, priorities are on a solid base and stability over cutting edge items that have no relevance to the software.
- You can also refer to older and historical notes here.

Developer's Kit or Developer's Kit? A Brief History

The original Developer's Kit is what is now found in the DEVKIT folder. Certain comments, notes, and documentation refer to this as the "Developer's Kit." The current Developer's Kit includes the original Developer's Kit plus numerous add-ons, notes, utilities, modifications, and other tools. Please be aware of this historical context when reading notes and documents that may reference the original.

What is now called the IMG Developer's Kit evolved from the My-T-Soft Developer's Kit. This Developer's Kit originally contained utilities & folders focused on this family of products. Now it is a collection of ALL developer utilities, for all IMG products that have developer information, source code, and utilities.

OnScreen and the Developer's Kit

OnScreen (formerly My-T-Soft AT) has always been built on the same source as My-T-Soft. Because OnScreen is focused on the individual user, and not truly meant as a tool for developers or integrators, IMG does not recommend nor support using OnScreen with these Developer's Kit tools & utilities. With that said, OnScreen has its own set of tools & utils included with this Developer's Kit, and many of the DevKit utilities will operate with OnScreen. You can refer to the WCREMAP utility in the DEVKIT folder, and the OnScreen folder, that includes specially modified basic DevKit Utilities optimized for OnScreen. This is ONLY done as a courtesy to technical users of OnScreen or technicians / technically savvy specialists who are using OnScreen. To re-iterate, there are many assumptions taken in OnScreen, and the primary assumption is that it is used by an individual user (i.e. imagine an individual and their computer - shouldn't they be in complete control of all software that is being used?). So having external utilities that can manipulate this tool make no practical sense. However, the capabilities of the Developer's Kit in conjunction with the commercial software, combined with the special capabilities of OnScreen has also created a desire to use OnScreen and the Developer's Kit tools. The specific tools available provide a basic toolset that has satisfied various parties.

Build-A-Board and the Developer's Kit

The My-T-Soft executable (MYTSOFT.EXE) included with Build-A-Board is the next generation of the software. It is built from its own code base, and many of the Developer's Kit utilities, source code, and documentation may not be relevant (or operate correctly or at all) with the next generation of My-T-Soft (i.e. any version greater than 2.00). As Build-A-Board evolves, the Developer's Kit will evolve with it. If you have particular questions about a specific utility, or are having difficulties while working with Build-A-Board and the Developer's Kit, please feel free to contact IMG.

Because many users of Build-A-Board will eventually move the run-time files to a different system, the Developer's Kit is installed into its own SDK folder. Reference \Program Files\Build-A-Board\[Project Name]\[Target System Folder] By including relevant Developer's Kit utilities into that Target System folder, the Make Run-Time Files or ActiveSync options will also include these utilities.

In the 1.77 release of My-T-Soft/My-T-Touch/My-T-Pen, a Build-A-Board run-time license is included with the run-time license for the 1.77 release. By manipulating the KEYBOARD.KBF file, the desired layout can be brought up as required. Note that many of the Developer's Kit utilities work with both run-time programs, but not all. Refer to the support notes for Version 2 supported functions.

The Magnifier Developer's Kit

You will want to unzip the IMG Developer's Kit, and refer to the Magnifier folder, and the section covering The Magnifier Developer's Kit.

Joystick-To-Mouse Developer's Kit

You will want to unzip the IMG Developer's Kit, and refer to the JTMUTIL folder, and the section covering the Joystick-To-Mouse Developer's Kit.

Version 1.90 Notes

Developer's Kit 1.90 (DK190001.ZIP)

This was released in May 2022 for the 1.90 Release 5. There never was a 1.80 release. Over the past 10 years, the working updated Dev Kit was made available to customers, but was never publicly released until this release. Includes 64-bit versions of all DEVKIT utilities and code-signed EXEs with different manifests to assist integrating in different systems with different permissions/security levels and/or with User Account Control.

Version 1.79 Notes

Developer's Kit 1.79 (DK179001.ZIP)

This was released in September 2012 for the 1.79 Release. Includes the Themes, MultiTouch DLL, Zip updates, and first public release of the HTML version of help.

Version 1.78 Notes

Developer's Kit 1.78 Release 5 (DK178005.ZIP)

This was configured in April 2011 for 1.78 Release 5, but the 1.78 R5 was never publicly released, and all changes roll into the 1.79 release. There was never a public Release 5 or 4. The Developer's Kit documentation has been reworked to be available as HTML and PDF, and available on IMG's website in its entirety.

Developer's Kit 1.78 Release 3 (DK178003.ZIP)

This was released in October 2007 in conjunction with the 1.78 Release 2, and adds 3 new options. PAINTDLL allows external painting of keys, WORDSDLL to integrate with OnScreen and the Word lists/Word Complete candidates, and MTSAppBar which provides an AppBar (taskbar like app that attaches to top or bottom of screen and limits desktop/application area). The DLL capabilities require the 1.78 Release 2 versions of My-T-Soft.

In a continuing effort to streamline things, this release also includes the Developer tools for The Magnifier 1.50 and Joystick-To-Mouse 2.60. The intent is to bring all development tools & documentation into one easy to update and maintain structure.

Developer's Kit 1.78 Release 2 (DK178002.ZIP)

This was released in May 2007, and we have reworked all of the Developer's Kit examples, MTSDLL, Visual Basic, C#, J#, Java, and Internet Explorer examples to work with the new 1.78 location changes in a much better way. This also includes source for KeyboardSync, WebSync, EditSync, OnScreen samples, and keyboard layout (KMF) sources.

Developer's Kit 1.78 Release 1 (DK178001.ZIP)

There are various updates, and the following new examples: C Sharp (C# / .NET integration example & source);J Sharp (J#) example; Java integration example; AddOnDLL example; and a specific usage of the Add-On DLL approach with LOGDLL (when used, logs usage details). We have kept the samples in Visual Studio 6 (rather than Visual Studio 2005) for maximum compatibility - you should have no problem importing and updating these samples if you are using later MS tools.

IMPORTANT!! The 1.78 version & Vista versions moved configuration files such as the INI and KEYBOARD.KBF files out of the installation folder into personal / user locations. Refer to the ConfigPath entry in the INI documentation for options & details. In general, this will cause problems with many assumptions in the Developer's Kit. Use ConfigPath=0 for direct compatibility with the early 1.78 releases of the Developer's Kit. See below for other Developer's Kits notes regarding this.

Developer's Kits Updates: With the 1.78 release, the Developer's Kit is no longer included by default with the release of the software - check your distribution media, or go on-line to download the latest version. Because the on-line update is now a default option, there will be numerous updates to address Windows Vista and changes to the release that affect the Developer's Kit. There will be update releases to address & update the Developer's Kit to work with the new configuration files locations and changes to the current & future releases of the My-T-Soft software. If the current available Developer's Kit approach does not work with the current My-T-Soft software, please check for more recent updates and let us know your situation. Note that we provide source code, documentation, and assume we are dealing with capable developers. While we attempt to make our software easy to use, and would like to make everyones life easier, being forced to change the location of our configuration files and dropping 14 years of consistency wasnt our first choice. However, there are valid reasons and other benefits to providing flexibility in the location of the configuration files, so during the transition period, the developers kit and any developers running into these issues will need to be somewhat adaptable.

Version 1.77 Notes

Updates to DEVKIT (CONFGMTS for Toolbar panel manipulation), CPYCNMTS, MOVEWMTS, KybdSync, MTSDLL, updated MS-Access example, updates to the Internet Explorer examples, a Visual C example added, sample logos with minimize button shown, and all relevant notes are now in this single Help file.

Version 1.75 Notes

A DEVKIT2 Folder has been added to customize the Developer's Kit to work with Version 2.00+ versions of My-T-Soft.

CLOSEMTS, MOVEWMTS, MINMZMTS, FWCTLMTS, OPEN_MTS, REPOSMTS, and SVPOSMTS are included.

Version 1.74 Notes

A DEVKITCE Folder has been added to bring basic DevKit executables & source for the Windows CE platforms that Build-A-Board supports. CLOSEMTS, MOVEWMTS, MINMZMTS, OPEN_MTS, REPOSMTS, and SVPOSMTS are included.

- KEYBOARD folder includes KEYENTER.EXE, KEYBACK.EXE, KEYTAB.EXE, KEYESC.EXE - simple executable that generate the virtual keystroke as indicated by the file name.

Version 1.73 Notes

More of a minor update & maintenance release than anything else.

These are the relevant additions that may be of interest:

- MTSISTRT has been added in the DEVKIT folder - allows for a clean implementation of opening My-T-Soft/My-T-Touch/My-T-Pen as an icon (or icon in the tray (Shell Icon)). Note that if you tag the shortcut with the Run property as Minimized, the keyboard will open in the minimized state as a button. There is no way to alter this, so this utility gives you all three choices of opening in a minimized state.

- STRESS.EXE (DEVUTIL folder) has been added to script various developer kit commands and verify the integrity of the commands and the keyboard software. There was a minor memory leak when the 3D keys were in use and the CONFGMTS utility was used, and this little utility helped nail down that there was indeed a problem.

- EDITSYNC.EXE/ESLIB.DLL (UserUtil folder) has been added that allows 2 operation options - physical keyboard monitoring for OnScreen (to work in conjunction with WordComplete), and auto-positioning for text caret position (using Active Accessibility)

Version 1.72 Notes

Some of the redundant documentation in each individual folder has been removed - this document is the current & best source of info for the Developer's Kit.

The CPYCNMTS/FWCTLMTS have been redone to use only Microsoft Win32 APIs to allow proper operation in the NT/2000/XP platforms.

There have been some "Read-Only" file issues with Microsoft Source Safe within the Visual Basic examples, and a problem within the CPYCNMTS/FWCTLMTS utility on NT/2000/XP.

Files tagged as "Read-Only" will not be removed by the 1.70 Uninstall, and any files & folders will have to be removed manually.

Finally a handful of new utilities have been added:

- TurnAway for Head/Eye "mouse" users (OnScreen)

- WCRemap (WordComplete remap - external utility to trigger completions) (OnScreen)

- ShowInfo - a nice utility to show Window Names & Classes, Handles & Parent

- WebSync - Internet Explorer utility for client control of My-T-Soft via VBScript / Client Scripting

- WaitRun - a utility to execute a command after a set period of time

Version 1.71 Notes

In Version 1.71, the Win32 compiler is Microsoft Visual C++ Version 6 - the DevKit, CtAltDel, and MTSDLL folders include project information for this compiler.

Version 1.50 Notes

In Version 1.50, there is an icon in the Program Group created for My-T-Soft - Run Install Developer's Kit, and then select Extract to create the files & folders on your system.

Files And Installation Notes

Files And Installation Notes

The Developer's Kit is a single file delivered as a Zip file - it is recommended that you use the DVKTINST.EXE (DevKit Install executable to extract the files & folders, or use WinZip, or another

decompressing utility that can handle Zip files with long file names). The Zip does include long file names, so using PKUNZIP may result in some loss of information within the names (& projects containing these files).

By default, these folders & files will be removed in you select Un-install from the Control Panel | Add/Remove programs for the IMG product.

The following is the folder structure created when unzipped above. For My-T-Soft the default INSTALLFOLDER is: \Program Files\MYTSOFT (My-T-Pen replace MYTSOFT with MYTPEN) (My-T-Touch replace MYTSOFT with MYTTOUCH) For Build-A-Board, the default INSTALLFOLDER is \Program Files\Build-A-Board\SDK

(The following listing is in the order these utilities were added into the current DevKit. Over time, various utilities have been revisited & updated, so this list is just a quick reference of what is included - refer to the contents & other documentation details for a more useful way to access specific utilities)

INSTALLFOLDER\DEVKTDOC - This Document - start at home.html

INSTALLFOLDER\DEVKIT - My-T-Soft Developer Kit

INSTALLFOLDER\NTUTILS - NT Utilities

INSTALLFOLDER\WIN3UTIL - Win 3.x Utilities

INSTALLFOLDER\CNTRLMTS - Control My-T-Soft Button Utility

INSTALLFOLDER\LOGOS - Logos for Custom Logo Option

INSTALLFOLDER\MSECLKS - Mouse click utilities

INSTALLFOLDER\VBASIC - Visual Basic Examples

INSTALLFOLDER\CTALTDEL - Ctrl-Alt-Del Emulation

INSTALLFOLDER\MTSEDIT - COM Edit control

INSTALLFOLDER\MTSDLL - MTS DLL w/source (with Access 97 Example)

INSTALLFOLDER\KEYBOARD - Keyboard based utilities

INSTALLFOLDER\IEXPLORE - Internet Explorer utilities

INSTALLFOLDER\USERUTIL - User utilities

INSTALLFOLDER\DEVUTIL - Developer utilities

INSTALLFOLDER\DEVKITCE - My-T-Soft Developer's Kit for Windows CE

INSTALLFOLDER\DEVKIT2 - My-T-Soft Developer's Kit for Build-A-Board / My-T-Soft 2.x

INSTALLFOLDER\VISUALC - Visual C++ Examples (Microsoft Visual C++ Version 6 Project)

INSTALLFOLDER\MSACCESS - Microsoft Access integration (MTSDLL)

INSTALLFOLDER\CSHARP - C# (.NET) example

INSTALLFOLDER\JSHARP - J# (J sharp) example

INSTALLFOLDER\JAVA - Java example

INSTALLFOLDER\ADDONDLL - Add-On DLL template, example

INSTALLFOLDER\LOGDLL - Log DLL usage logging DLL based on ADDONDLL capability

INSTALLFOLDER\ONSCREEN - OnScreen specific Developer's Kit Utilities

INSTALLFOLDER\INCLUDE - MYTSOFT.H header file & Update.bat

INSTALLFOLDER\MODAL - Modal Input utility for integration (locks out non-keyboard mouse clicks)

INSTALLFOLDER\LAYOUTS - Source files for keyboard layouts & build tools

INSTALLFOLDER\PAINTDLL - Source files for DLL that allows external painting of keys

INSTALLFOLDER\WORDSDLL - Source files for DLL that integrates with OnScreens Word complete

INSTALLFOLDER\MTSAppBar - Uses a taskbar like appbar to resize desktop for OnScreen

INSTALLFOLDER\Magnifier - Tools for external control of The Magnifier

INSTALLFOLDER\JTMUtil - Tools for external control & working with Joystick-To-Mouse

INSTALLFOLDER\THEMES - Source for examples THEMES used with My-T-Soft releases

INSTALLFOLDER\MultiTouchDLL - Source for Multi-Touch/Gesture/Flicks interfaces

INSTALLFOLDER\ZIPUNZIP - Source for ZIP32.DLL/UNZIP32S.DLL

Application notes and various other helpful hints can be found at IMG's web site / IMG's Developer's Corner (http://www.imgpresents.com/imgdev.htm). This is the link for the Developer's Corner within this document.

If your distribution software did not come with the Developer's Kit diskette or files, you may download the latest version from the web site.

Because of the constantly changing nature of software development, this document may not be as current as the web site. Additionally, the wants & needs of developers depend on external factors, and timely information relevant to a particular application may only be available on-line.

The Developer's Kit contains utilities for implementing My-T-Soft within application development environments (such as Microsoft Access, Borland Delphi, Visual Basic, Oracle, and other visual application design environments), and high-level applications that provide basic external program control with a Shell / WinExec / Spawn function (such as LabView, WonderWare, and other third-party software systems).

Frequently Asked Questions

> **Note:** This is a general FAQ section on the IMG Developer's Kit - you may be also interested in the specific "How Do I..." section.

Why do I need the IMG Developer's Kit?

Implementing Touchscreen & Pen based input driven applications often require limited keyboard input. Even though the pointing device can operate a well written application, there are often times when quantities, names, files, passwords, or other character based input is required. By NOT including a physical keyboard with these systems, the manufacturer / integrator often sees lower production costs, and allows a seamless operation from the more intuitive user interface. By implementing the various keyboards & panels in the My-T-Soft software and using this developer's kit, the application developer /

systems integrator can provide all of the capabilities of a physical keyboard. The ability to bring up a specific panel for the user, then sending it away when not needed (to restore valuable screen real-estate), allows the developer / integrator to implement a complete & fully functional application on these types of systems.

My time is limited - where should I start?

Most people new to the Developer's Kit want to get going quickly, and don't want to review items that don't apply to them. Because there is a wide spectrum of experience for people new to My-T-Soft, and a wide spectrum of requirements, it is difficult to generalize. Here is an attempt (from our experiences) at addressing this question.

Most uses of the Developer's Kit revolve around moving and configuring My-T-Soft when appropriate for the user. Key utilities are MOVEWMTS (My-T-Soft MoveWindow), CONFGMTS (Configure My-T-Soft), and CPYCNMTS (Copy new Configuration to My-T-Soft). Note that these utilities are available in both the DEVKIT folder, and as part of the MTSDLL DLL example.

These utilities accomplish the following:

MOVEWMTS - by moving the My-T-Soft window (i.e. moving the keyboard panels around, or moving off-screen when not needed) a clean interface to the application from a touchscreen or pen interface can be accomplished quickly. Usually the call to MOVEWMTS is triggered by an event, such as Field Enter (e.g. OnEnter), or Field Exit (e.g. OnExit). It is recommended that My-T-Soft be moved off-screen when not needed, and on-screen when appropriate. This utility easily handles the manipulation of where the keyboard is displayed.

CONFGMTS - this lets you change size and panels available. Not as powerful as CPYCNMTS, but basic & familiar to most developers. Only real trade-off is display speed - on faster machines, this may not matter much.

CPYCNMTS - This utility requires pre-configured layouts, but once these are assembled, the seamless changeover from one configuration to another is a developers dream.

Programming language? Whats that? (or they call me a Developer, so I better act like one) Review the Hints & Comments below, and review the My-T-Soft Developer's Kit - the pre-compiled executables can be run from most high-level environments, or from a windowed command prompt.

Start with MOVEWMTS and CONFGMTS to get a feel for what is possible.

I use C and/or C++

The source for the Developer's Kit and the MTSDLL is in C. Picking up the relevant parts is easy for Windows developers. If not familiar with the Windows API, start with MTSDLL. Also refer to the Visual C++ Examples (Microsoft Visual C Version 6 Project)

I'm using .NET and/or C#

Refer to the C# (.NET) examples & source code.

I'm using JAVA or J#

Refer to the JAVA example and/or the J# examples

I'm using Visual Basic, Visual FoxPro, MS Access, or other high-level development environment

Refer to the Visual Basic example , or the MS Access example in MTSDLLMS, or review the MTSDLL functions.

I'm using Internet Explorer, or building a Web based kiosk Refer to the IEXPLORE.HTM (INSTALLFOLDER\IEXPLORE\IEXPLORE.HTM) file in the IEXPLORE folder

NOTE: The current directory depends on where you started this help file Run WebSync from INSTALLFOLDER (Run: \iexplore\websync.exe), then Open HTML file from INSTALLFOLDER (Open:.\iexplore\iexplore.htm)

Why isn't this an OCX (ActiveX/COM control)?

These types of controls imply that there is a container to hold them, or some other software available to communicate to the control. Since the keyboard software is used by all types of users in all types of situations, forcing this kind of constraint of the software is too limiting. Allowing both direct user control (in cases where technicians or users are modifying the names of desktop icons, or changing file names, or logging on to a network, etc.), and allowing programmatic control (via events in edit controls, form activations, etc.) with the same exact piece of software is more powerful than limiting it only to development environments capable of working with COM controls. Since the same results can be achieved with the current implementation, but not with an COM control, there is little reason to re-work the software as an COM Control. See the MTSEDIT COM control included in the Developer's Kit as an example of a useful approach using COM controls. Also note that the keyboard software is written at the same level as the Windows operating system, making it an important part of any system where there is only one interface (i.e. the pointing device).

Hints and Comments for Developers (My-T-Touch / My-T-Pen / My-T-Soft Developer's Kit)

This section is available at Hints and Comments.

Chapter 2. Developer's Corner

Developer's Corner

> **Note:** This section represents the Developer's Corner page on the IMG website, but is also represented in the Manual and Help provided with the IMG Developer's Kit itself. This section is intended primarily to be accessed via a web browser interface, direct from IMG's website. In order to maintain all information pertaining to the IMG Developer's Kit in a single source, certain aspects of this section may seem inconsistent or out of place depending on how and where it is accessed. The website information may be concurrent with this document, or newer.

IMG Developer's Kit Version 1.90 (Includes everything available for Developers!) Download Here (http://www.imgpresents.com/corner/imgdevd1.htm)

Click for File and Installation Notes, Release Notes, and/or Support Notes

See "How do I...?" Section Below...

Most Requested Information

- Overview of Developer's Kit
- Hints and Comments for Developers (My-T-Touch / My-T-Pen / My-T-Soft Developer's Kit)
- Downloads (http://www.imgpresents.com/corner/imgdevd1.htm)
- General Information on IMG's Software

For PDF versions of this document, use the links in the Developer's Corner Download area (http://www.imgpresents.com/corner/imgdevd1.htm). Refer to IMG's website for purchasing as a book.

How Do I...?

- **Configuration**
- How Do I Secure My-T-Touch / My-T-Pen / My-T-Soft for controlled applications (Operator Mode - where the user cannot change configuration)?
- How Do I Setup the Custom Logo in My-T-Touch / My-T-Pen / My-T-Soft?
- How Do I redefine keystrokes on the keyboard (How Do I use Key Options)?
- How Do I implement minimize on Enter key in My-T-Touch / My-T-Pen / My-T-Soft?
- **Developer's Kit**
- How Do I Open My-T-Touch / My-T-Pen / My-T-Soft Minimized?
- How Do I use the Copy and Configure Utility (CPYCNMTS) with My-T-Touch / My-T-Pen / My-T-Soft?
- How Do I toggle between 2 different layouts and have My-T-Touch / My-T-Pen / My-T-Soft synchronize the displayed keyboard (How do I use KYBDSYNC)?

- **Visual Basic, MS Access, Compatible Environments**
- How Do I integrate "on-demand" use with Visual Basic / MS-Access?
- How Do I register / move the DLL?
- **Internet Explorer**
- How Do I integrate "on-demand" use within Internet Explorer?

General Information on IMG's Software

General

All of IMG's products are applications within Windows, and provide enhancements to the standard interface options including keyboard, mouse, trackball, touchscreen, pen (or any pointing device), and joystick.

There are also other platforms supported, including Linux, Raspberry Pi, Android, and Mac OS X. Not all Developer Kit's tools have matching options on different platforms (sometimes due to platform differences, sometimes due to demand). If there are specific requirements that you can't find a solution to, be sure to reach out to us.

Product Information

Refer to each product's manual for all Initialization file options and product capabilities. See How Do I...? Solutions and other common Developer needs and review entire Developer's Corner for frequently requested information.

Hints & Comments

Hints and Comments for Developers (My-T-Touch / My-T-Pen / My-T-Soft Developer's Kit)

> **Note:** The Developer's Kit started with the set of utilities now found in the DEVKIT folder. It has grown to include numerous executables, source code, and helpful information - some older notes may refer to the Developer's Kit and only mean the utilities in the DEVKIT folder - newer notes refer to the whole set of utilities, source, documents, notes, etc.

(1) Instead of an open (launch) & close approach OR an Open & Minimize, IMG recommends to move the My-T-Soft window off-screen (i.e. at an X position of 1281 for a 1280 x 1024 layout) and use the Copy & Configure utility to "restore" the appropriate layout when required (or just MoveMTS back on-screen at the desired location if the layout does not need to change). This seamless changeover reduces repaint time, and is virtually instantaneous. In multiple monitor configurations, you must be aware of entire virtual space.

(2) Using a windowed Command Prompt allows you to use / test the utilities on demand, allowing you to learn them, save configurations, etc. Configure My-T-Soft, then use SVSETMTS.EXE to save configuration; Move window off screen with MOVEWMTS x:2000, then SVPOSMTS to save position off-screen. Use CLOSEMTS to shut down My-T-Soft. Add shortcut to StartUp group (with NoSplash

option), and your users will not know it is running until your application moves My-T-Soft window (MOVEWMTS) back on-screen!

(3) The -NoSplash command line can be used, or the NoSplash option is available via the INI file. Set NoSplash=1 (within the [Configuration] section of the MYTSOFT.INI in the installation directory) to disable the opening introduction (splash) screen. Note: First time painting delays may be noticeable depending on system speed, video memory, and system configuration.

(4) Sometimes the default reaction of the software to move off of dialog boxes (and other predefined windows) is not desired. Under Operation Options, the "Move off Dialogs..." setting sets Contention=0 (within [Configuration] section) to disable the default monitoring of other windows in the system. Note that this global setting can affect other advanced options. Be sure to test implementation if this change is made.

(5) Note the 2 save options (Settings & Position). Both must be done to "save" the current configuration. Settings refers to the displayed panels, and position refers to the screen position (upper left of window). The distinction is a legacy holdover from original design in 1993, and has specialized uses. Use utilities/batch to automate the two steps, or rework source of SVSETMTS and SVPOSMTS.

(6) When integrating with a high-level program, refer to the code examples for MS-Access, Visual Basic, or play around at a windowed command prompt (Item 2 above).

(7) Developer's Kit Source notes - Structure: The "action" code is contained in the WM_TIMER case (message) of the switch construct. In general, this source code is NOT meant to be dropped into your application code. Because some high-level environments DO NOT support the Windows API, they are forced to use the compiled executables (and, in fact, it was assumed that a C developer would not need source for these simple calls). Windows C and C++ developers should have no problem sifting through the code for the relevant sections. The reason the WM_TIMER message is used (rather than simply using WinMain or WM_CREATE) is to allow an easy addition of a delay by increasing the SetTimer milliseconds. The extra structure and variables were left in place for flexibility in case there is the need for debugging, displaying a window, or some structure or variable is required. If you do not have familiarity with legacy Windows code and C only based Windows 3.1 structures, refer only to the code in WM_TIMER.

(8) The most useful utilities are the MOVEWMTS and CPYCNMTS (with SVSETMTS and SVPOSMTS to "lock" down a configuration) - an incredible amount of flexibility can be obtained easily be saving configurations, and using the MOVEWMTS to move My-T-Soft off-screen, and then revert to an already saved configuration based on your application's/user's needs.

(9) The SDSTRMTS allows for complete remapping of existing keyboard layouts within the Setup | Configuration | Key Options dialog. Note that complete macro strings can be embedded, so F1 could be your name, F2 could be your Address, etc. Only 1 executable with command line is available per key.

Overview of Developer's Kit

My-T-Soft Developer utilities (Developer's Kit)

In addition to the many programmable functions and features already found in OnScreen, My-T-Mouse, My-T-Pen, My-T-Soft, and My-T-Touch, IMG has had numerous requests for a way to control its programs externally without a lot of programming expertise. The Developer's Kit includes features & programs requested by our customers.

These files have been created by our programmers for calling My-T-Pen, My-T-Soft, & My-T-Touch keyboards, numeric, calculator, and programmable macro panels from within application development environments such as Access, ActiveX, Delphi, HTML, Intellution, Java, LabView, Oracle, RSView32, Visual Basic, Wonderware, or other high level development application design tools. They do not require knowledge of, or the use of Windows Messaging.

Each executable also comes with the C source module, a Resource Script (RC), and a Definition file (DEF) for programmers who wish to modify or customize functions. They are easily compiled into their executable form by anyone familiar with Visual C++, C++ Builder, or any other standard C compiler with Windows support.

These utilities were developed in response to customer suggestions, wants, needs, and by reviewing the capabilities of various high-end application development tools. The executable approach allows use of Shell, WinExec, or other API (Application Programming Interface) calls common to any robust environment. The inclusion of the source code allows easy integration into applications developed closer to the Windows API. The instructional information is also useful in allowing developers to create functional applications quickly.

There are numerous different utilities in the Developer's Kit. They are available on the web or included in both licensed and demo versions of My-T-Pen, My-T-Touch, and My-T-Soft. Some pair up as opposites, some are more powerful than others. Be sure to read the details outlined in the appropriate Kit to gain a solid understanding of the ability of these tools to assist you in your efforts. Please feel free to contact us with other ideas or needs.

EXECUTABLE FUNCTION DESCRIPTIONS (see Developer's Kit Documentation for additional features)

Open / Close - Keyboard, Panel, or Component

Show / Hide - Keyboard, Panel, or Component

Minimize / Maximize - Keyboard, Panel, or Component

Move - (to x: y: co-ordinates) - Keyboard, Panel, or Component

Get Position - Retrieves current position (x: y:) of Keyboard or Generic Window

MouseClick Utilities - generate hardware mouse click or double-click

SetCursor - Moves cursor to x: y: co-ordinates

Configure (close or open keyboard, edit, numeric, calculator, macro, windows, or tool panel(s) and choose size)

Copy and Configure (store multiple configurations and call up a specific configuration)

Save Position / Save Settings & Restore Position / Restore Settings

Button Utility - place up to 4 buttons floating on the caption bar to let users launch different configurations

Control Panel Logos - templates for building custom logos to cover and lock-out the IMG Control Panel

Toggle - move off-screen if on-screen and toggle / reconfigure panels off-screen if on-screen

SendString - send keystrokes or string of keystrokes (used for programming keys)

KeyWatch - This is a "debugging" program for monitoring keystrokes - works with all input methods

KeyboardSync - When multiple keyboard layouts are loaded, synchronizes displayed layout to current selection.

WaitRun - This allows you to insert a 1-30 second delay prior to executing the specified program/utility.

WCRemap - WordComplete Remap (for OnScreen only) Allows external triggering of WordComplete buttons.

NOTICE / DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

Visual Basic Forms / Developer's Kit DLL / MS Access Example

Sample Visual Basic forms are included for developers familiar with Microsoft Visual Basic. Code is included that interfaces with a Dynamic Link Library (DLL [Source included!]) that includes all of the Developer's Kit functionality accessible as function calls in C / Visual Basic, or any other high-level development environment that can interface with a standard DLL. A sample Microsoft Access 97 application is included with source to show how to integrate the DLL into Access, and some useful functions are implemented to illustrate "on-demand" display.

When run in Visual Basic (VB), the presented form (window) illustrates the power of the Developer's Kit by calling out to My-T-Soft / My-T-Pen / My-T-Touch in an easy to use GUI (Graphical User Interface) button driven application. Examples of Opening / Closing / Minimizing / Moving / Saving & Restoring: Settings & Position / Sending Characters through My-T-Soft / are shown, along with showing on-the-fly configuration from both pre-defined layouts to programmable selections. The Visual Basic form includes the source for integrating the DLL directly into VB (which can be referenced for integration into other environments). Written in VB 4 for maximum compatibility.

MTSEDIT COM Control

A sample Edit COM Control with source code & detailed instructions for building is included with the Developer's Kit. The example shows how to create an edit COM control (which can be integrated into any environment capable of working with COM controls). The new edit control automatically positions the keyboard at the desired location when the control receives the keyboard (input) focus, and removes the keyboard from the screen when it loses the input focus.

Logon Utilities

These files have been created by our programmers in response to the requests of customers & systems integrators. Each has a different function. Some are provided as stand-alone executable files (EXE). Although primarily meant for the Window NT platform, being standard Win32 32-bit windows Applications, the Reboot, Restart, Shutdown, and NT (Logoff) Logon will also operate in the Windows 95/98/Me environments.

Ctrl/Alt/Del Workstation/Desktop Logon

The feature permits a user to unlock the Ctrl/Alt/Del logon with a single touch on any pen or touchscreen, launching the workstation's logon window and a My-T-Soft Onscreen Keyboard. All three entry fields (User, Password, Domain) are accessible. Multi-user and domain logons are supported. The default user and domain is preserved.

Auto Logon

Utility used to enable the Auto Admin Logon capability already built in to Windows. Provides a front end user interface to allow enable or disable the capability, and to enter the user name and password of the user that will auto logon to the workstation when Windows starts up.

Reboot - Shuts down the PC, and restarts the computer (Cold-Boot)

Restart - Restarts Windows (Warm-Boot)

Shutdown - Shuts down Windows (Power-off if supported by hardware)

(Logoff) Logon - Shuts down Windows and logs on as another user

Joystick-To-Mouse

Contains pre-built utility EXEs that can externally communicate and control Joystick-To-Mouse.

The Magnifier

Contains a DLL with source that can externally communicate and control The Magnifier, along with a C based command line utility that links to the DLL, and Visual Basic integration example.

How do I secure My-T-Touch for controlled applications?

Securing My-T-Touch / My-T-Pen / My-T-Soft for controlled applications

My-T-Touch used as an example

(1) Configure My-T-Touch with the proper Panels open & set the appropriate size.

(2) Make sure the Control Panel is NOT open, and that if you are using the Macro Panels, that no accessible link can lead to other applications.

(3) Position My-T-Touch where desired, and then select from the My-T-Touch Menu: Current Settings, Save Current Settings and Position, Save Current Position.

(4) Close My-T-Touch.

(5) Open My-T-Touch Setup. Select Configuration, Special, & Check On Enable Operator Security. In 1.42, you can also disable minimize, and enable the Custom Logo (See Custom Logo below). You may also wish to turn off Action Button Move (My-T-Touch Setup | Configuration | Operation Options) to prevent repositioning of My-T-Touch if important for your application.

(6) Use Show & Hide Keys to eliminate any potential troublesome keys if keyboard panel is used (e.g. Ctrl, Alt, Function keys, etc.).

(7) Review Mouse Buttons to make sure 2nd or 3rd button (if applicable) are set properly, or set to Ignore Button Press.

(8) Close My-T-Touch Setup.

(9) Configure system for My-T-Touch launch, e.g. Windows StartUp group, MS Access Shell command, WinExec, CreateProcess, etc.

(10) You may wish to remove My-T-Touch Setup entirely, or place in secure directory, or set file attributes (check with network / system administrator). Note: A password will be required to enter My-T-Touch Setup, and this may be secure enough for some applications.

(11) My-T-Touch Repositioning, My-T-Touch reactions to other Windows, other special features: In general, you will want to make Contention=0 instead of Contention=1. You may need to review Advanced User Notes in Help for coverage of Initialization file settings, such as Contention, MyTClass????, etc.

How do I Setup the Custom Logo?

Custom Logo in My-T-Touch / My-T-Pen / My-T-Soft

My-T-Touch used as an example

(1) With My-T-Touch Closed, Open My-T-Touch Setup, Select Configuration, Special, & Check On Use Custom Logo.

(2) Close My-T-Touch Setup.

(3) Reference the On-Line manual for bitmap sizes, or use the templates available in the 12 Logo Bitmaps in LOGOS.ZIP (On the distribution disk, or available for download below).

(4) The appropriate bitmap for each size must be located in the configuration folder (see My-T-Touch Help | File menu | Show Config File Location -or- refer to ConfigPath setting in the MYTTOUCH.INI file)

(5) The naming convention is LOGO??.BMP where ?? is 01 through 12. For larger sizes, LOGO12.BMP is scaled to fit.

(6) If there is a problem locating or loading the bitmap image, a black rectangle will appear on My-T-Touch.

(7) This feature make the most sense when used in conjunction with the Operator Mode, and disabling all Tool Bar buttons, otherwise the Control Panel toggle, Menu, and Minimize buttons will still function even with the Custom Logo in use.

(8) Read LOGOS.TXT in LOGOS.ZIP for other information & implementation issues.

How Do I redefine Key Options

Steps for remapping the F1 & F10 keys (as a specific example)

My-T-Soft used as an example

Map F1 key on My-T-Soft to open an HTML Web Page instead of default keystroke of F1

(1) Configure My-T-Soft with the Keyboard Panel and any other Panels open & set the appropriate size. Save Settings, Save Position, then Close My-T-Soft.

(2) Run My-T-Soft Setup.

(3) Select Configuration | Key Options.

(4) Select the F1 key in the "Key" list on the left. Do not select any modifiers "Key with..." (Ctrl/Shift/Alt).

(5) Select Disable Keystroke(s) (so the F1 key is not sent as a keystroke), and Select Launch Key EXE.

(6) Now enter the path / file name for Explorer (or Internet Explorer) or use Browse to find the appropriate file. (In Windows 2000, it would be C:\WINNT\EXPLORER.EXE).

(7) Now enter the path / file name for page to be displayed after the exe (e.g. "C:\Program Files\Common Files\Microsoft Shared\Citrus punch.htm") with whole name as [C:\WINNT\EXPLORER.EXE C:\Program Files\Common Files\Microsoft Shared\Citrus punch.htm]

(8) Close Setup, Run My-T-Soft. When the F1 key is clicked, the web page will appear in Explorer.

Map F10 key on My-T-Soft to type web site instead of default keystroke of F10

(1) Configure My-T-Soft with the Keyboard Panel and any other Panels open & set the appropriate size. Save Settings, Save Position, then Close My-T-Soft.

(2) Run My-T-Soft Setup.

(3) Prepare for Send String - The Send String MUST be in the same directory as the Keyboard Executable (i.e. MYTSOFT.EXE). Therefore, we need to copy the SDSTRMTS.EXE file from the DEVKIT folder down a level into the default install (C:\Program Files\MYTSOFT) folder. Use Explorer to open WINNT, MYTSOFT, DEVKIT, Highlight SDSTRMTS.EXE, Right-click, select Copy, click on MYTSOFT, Right-click on blank area, Select Paste.

(4) Select Configuration | Key Options.

(5) Select the F10 key in the "Key" list on the left. Do not select any modifiers "Key with..." (Ctrl/Shift/Alt).

(6) Select Disable Keystroke(s) (so the F10 key is not sent as a keystroke), and Select Launch Key EXE.

(7) Now enter the path / file name for SDSTRMTS.EXE (Send String through My-T-Soft) or use Browse to find. (In Windows 2000, after copy in step 3, it would be C:\WINNT\MYTSOFT\SDSTRMTS.EXE).

(8) Now enter the keystrokes to send through My-T-Soft (e.g. "www.My-T-Soft.com[Enter]") with Executable name as the Key EXE as in [C:\WINNT\MYTSOFT\SDSTRMTS.EXE www.My-T-Soft.com[Enter]]

(9) Close Setup, Run My-T-Soft. When the F10 key is clicked on My-T-Soft, the text will be typed, effectively remapping the F10 key to a custom sequence of keystrokes.

How do I implement minimize on Enter?

Steps for setting up the Minimize on Enter Key / Key Options

My-T-Soft used as an example

It is recommended that developer's use a move off-screen when not needed, move on-screen when needed approach for "putting" My-T-Soft away when not desired. This gives more control of the user-interface, quicker display times, and is the preferred method for manipulating My-T-Soft.

(1) Configure My-T-Soft with the proper Panels open & set the appropriate size. Save Settings, Save Position, then Close My-T-Soft.

(2) The MINMZMTS.EXE should be available in the Installation directory, or in the default directory (Install Developer's Kit).

(3) Run My-T-Soft Setup.

(4) Select Configuration | Key Options.

(5) Select the Enter key in the "Key" list on the left. Do not select any modifiers "Key with..." (Ctrl/Shift/Alt).

(6) Leave Enable Key selected (so the Enter key is sent as a keystroke), and Select Launch Key EXE.

(7) Now enter the path / file name for MINMZMTS.EXE or use Browse to find the file. Verify Key EXE indicates MINMZMTS.EXE, then click OK.

(8) Close Setup, Run My-T-Soft. When the Enter key is clicked, the "Enter" keystroke will be sent, and the MINMZMTS.EXE will be launched, resulting in "Minimize on Enter key."

(9) NOTE: Using MOVEWMTS and other Dev Kit utilities can create other options. It is preferable to have these interactions controlled in the "Typed into" application, but there are times when this is NOT a possibility, and these key actions allow some customization.

(10) NOTE: Do not be surprised when the keyboard "disappears" after pressing Enter!!

How do I open minimized?

Begin operation minimized / StartUp Group / Developer Kit example

Version 1.70 of My-T-Touch/My-T-Pen/My-T-Soft support the Shell properties of the shortcut Icon (right-click on shortcut, select properties). Set the Run: option to Minimized. The default Minimize mode is to minimize as a button. If you need one of the alternate minimize modes, refer to the My-T-Soft Setup | Configuration section - minimize to task bar icon, shell tray icon, floating Window, or caption/title bar button. Also in 1.79 there is a command line option to open in the minimize state (without using the Run: shortcut link properties).

How Do I use the Copy & Configure utility

Steps for using the Copy & Configure utility / Integrating with Visual Basic / MS-Access

Visual Basic Developers: Review the samples in the VBASIC folder (Installed Developer's Kit) for Form Event Examples, and Module Load information - This is the starting point for integrating into VB! (Note: MTSDLL is the future of this DLL (1.71 Release))

Developer's Kit 1.71 includes the MTS DLL (MTSDLL folder from extracted Developer's Kit) with an sample MDB database for MS-Access 97 showing how to tie the DLL into Access, and the event settings on a sample form - the steps & examples included in the MTSDLL folder may be what you are looking for. The same concepts apply, but the implementation is more integrated than the steps listed below. Note: The steps to create preset configuration still apply.

My-T-Soft used as an example / Integrated with MS-Access

(1) Configure My-T-Soft with the proper Panels open & set the appropriate size. The necessary utilities are expected to be in current directory for MS-Access.

(2) Use the Menu to select Save Current Settings, Save Current position (or use utilities).

(3) The KEYBOARD.KBF in the installation directory contains the current saved configuration. Copy this file to a DIFFERENT file name, e.g. KEYPAD.CFG, NUM.CFG, etc. (For this example, this is assumed to be the installation directory) Note: Do not use .KBF extension for these extra files.

(4) Repeat Steps 1 - 3 as necessary to build up a set of configuration files.

(5) The Copy & Configure utility takes the file name as a single parameter.

(6) Example: In Microsoft Access - Field (Text Box) Property Enter (OnEnter). x = Shell("CPYCNMTS C:\WINNT\MYTSOFT\KEYPAD.CFG",1)

(7) Description: The example in Step 6 would restore My-T-Soft from an off screen position and reconfigure to match the KEYPAD (previously saved) configuration.

(8) Example: In Microsoft Access - Field (Text Box) Property Exit (OnExit). x = Shell("MOVEWMTS X:641 Y:320",1)

(9) Description: The example in Step 8 moves My-T-Soft off screen when the field is exited (Keyboard Enter, Tab, etc.)

How do I toggle between keyboard layouts

Steps for setting up a keystroke macro behind F12 to toggle keyboard layouts, and using KYBDSYNC to automatically update My-T-Soft

My-T-Soft used as an example

(1) You will need to extract Developer's Kit (Install Developer's Kit Icon in Program Group) to get to the KEYBOARD\KYBDSYNC.EXE and DEVKIT\SDSTRMTS.EXE

(2) Use the KYBDSYNC (just run EXE directly from Windows Explorer) - this will tie My-T-Soft display to current keyboard layout selected in Windows. You can run this in the Windows StartUp group to make it always available if required.

(3) Run Windows Control Panel and select the Keyboard icon. Click on Language Tab and verify the 2 language layouts desired are available. Your Windows CD may be required if you need to add a new layout.

(4) Copy the DEVKIT\SDSTRMTS.EXE to the MYTSOFT folder (e.g. to C:\WINDOWS\MYTSOFT from C:\WINDOWS\MYTSOFT\DEVKIT)

(5) Use F12 or one of the other function keys as the "trigger" to toggle keyboard layouts

(6) Go to My-T-Soft Setup | Configuration | Key Options

(7) Select F12, Disable Keystroke, Enable Key EXE

(8) Browse and select SDSTRMTS.EXE

(9) Add command line, e.g. "...\SDSTRMTS.EXE [Alt-Down][Shift-Down][Shift-Up][Alt-Up]"

(10) Now when F12 is clicked, the keyboard will toggle - with KeyboardSync running (KYBDSYNC.EXE), the display will toggle

How do I implement "on-demand" use with Visual Basic/MS Access?

Steps for using the Copy & Configure utility / Integrating with Visual Basic / MS-Access

Developer's Kit 1.71 includes the MTS DLL (MTSDLL folder from extracted Developer's Kit) with an sample MDB database for MS-Access 97 showing how to tie the DLL into Access, and the event settings on a sample form - the steps & examples included in the MTSDLL folder may be what you are looking for. The same concepts apply, but the implementation is more integrated than the steps listed below. Note: The steps to create preset configuration still apply.

My-T-Soft used as an example / Integrated with MS-Access

(1) Configure My-T-Soft with the proper Panels open & set the appropriate size. The necessary utilities are expected to be in current directory for MS-Access.

(2) Use the Menu to select Save Current Settings, Save Current position (or use utilities).

(3) The KEYBOARD.KBF in the installation directory contains the current saved configuration. Copy this file to a DIFFERENT file name, e.g. KEYPAD.CFG, NUM.CFG, etc. (For this example, this is assumed to be the installation directory) Note: Do not use .KBF extension for these extra files.

(4) Repeat Steps 1 - 3 as necessary to build up a set of configuration files.

(5) The Copy & Configure utility takes the file name as a single parameter.

(6) Example: In Microsoft Access - Field (Text Box) Property Enter (OnEnter). x = Shell("CPYCNMTS C:\WINNT\MYTSOFT\KEYPAD.CFG",1)

(7) Description: The example in Step 6 would restore My-T-Soft from an off screen position and reconfigure to match the KEYPAD (previously saved) configuration.

(8) Example: In Microsoft Access - Field (Text Box) Property Exit (OnExit). x = Shell("MOVEWMTS X:641 Y:320",1)

(9) Description: The example in Step 8 moves My-T-Soft off screen when the field is exited (Keyboard Enter, Tab, etc.)

How Do I register / move DLL

How Do I register / move the DLL?

Visual Basic Developers: Review the samples in the VBASIC folder (Installed Developer's Kit) for Form Event Examples, and Module Load information - This is the starting point for integrating into VB! (Note: MTSDLL is the future of this DLL (1.71 Release))

Question: How do I Register the DLL?

Answer: You don't - the DLL is a Dynamic Link Library of Compiled (Native C) code - by declaring the Module "Lib" statements, you are indicating to your VB Project where to find the DLL. This question arises because VB developer's are familiar with COM Components located in a DLL which need to register the Server that handles the interface to these COM objects. The MTSDLL is not like a COM Component in a DLL. The C-based DLL is a lot more direct, quicker, with no overhead, and what

optimized computer software is all about - do the job intended with minimal impact on system performance.

Several quick notes:

(1) You have the source code

(2) Think from your programs point of view: Calling the function (with the "Lib" statement required in the module) is more straightforward than reviewing all of the registered system components to find if a particular capability is available - if you've ever waited for Windows to load, you may appreciate the speed, power, and directness of using only what you need.

(3) If you move the DLL, you will need to update the "Lib" statements to indicate its location - also, see below.

(4) There are advantages and disadvantages of using COM components and advantages and disadvantages of using C-based DLL's. This is a C-based DLL.

(5) If you want to build this functionality as a COM interface that adds layers of complexity, takes time, taps system performance, and provides no further functionality, you have the source code.

Question: How do I move the DLL?

Answer:You may move the DLL anywhere on the system, but please note the following items:

(1) If you move the DLL, you will need to update the "Lib" statements to indicate its location.

(2) If "hard coding" the location is not desirable, you may simply enter the name of the DLL, e.g. "MTSDLL.DLL" and move the DLL into a location indicated in your environment's path. Although not recommended, you can drop the DLL into the "\WINDOWS\SYSTEM" ("\WINNT\SYSTEM32") folder. If unfamiliar with these issues, please refer to Microsoft documentation on Windows and the Visual Basic reference.

(3) If you don't know your project's location, you can use the "\Program Files\Common Files" area and create your own spot for your program's required files.

(4) Refer to the "Load" event of the example projects - use the Dir command to verify the DLL is located where you expected it, or exit gracefully.

(5) Refer to Microsoft Help for VBA on DLL's and locating them.

(6) Refer to Microsoft Knowledge Base Q106553 for notes on integrating C based DLL's and Visual Basic.

How do I Integrate with Internet Explorer

Integrating with Internet Explorer

My-T-Soft used as an example / Integrated with Internet Explorer

Option 1

You can trigger events within My-T-Soft by using WebSync and VBScript (or other scripting method that can create a file on the client system).

HREF="./webdev.exe">Download WEBDEV.EXE (39K)

(1) WEBDEV contains 2 HTM files and WebSync (EDITPAGE.HTM, OVERVIEW.HTM, WEBSYNC.EXE)

(2) Download this file and using Windows Explorer copy to the default folder (\WINDOWS\MYTSOFT or \WINNT\MYTSOFT)

(3) Double-click on WEBDEV.EXE to extract the 3 files (Note: You may need to close the DOS window after it runs and extracts files) (or use WinZip)

(4) Double-click on WEBSYNC.EXE to run WebSync

(5) Make sure you have My-T-Soft running, it is not minimized, and you have a keyboard open at about size 8

(6) Double-click on EDITPAGE.HTM - this should open Internet Explorer - if you are on Windows 98, My-T-Soft, you should see the keyboard position itself on EDITPAGE, and hide on OVERVIEW - the 2 pages are linked to illustrate this approach of using the Devevloper's Kit & VBScript with WebSync

(7) All the details about the VBScript approach and WebSync are in the OVERVIEW.HTM - refer to this for notes & implementation details

(8) If you are running Windows NT/2000 or My-T-Pen/My-T-Touch, you will have to modify EDITPAGE.HTM and OVERVIEW.HTM to view example

(9) Review OVERVIEW.HTM and the Web page "source" to see details.

(10) Quick Note: This example uses VBScript to communicate externally to WebSync, which uses the pre-compiled Developer's Kit executables to manipulate My-T-Soft.

Option 2

You can trigger events within My-T-Soft by using the My-T-Soft Setup | Configuration | Special Options' Run command. This works best if you only have a small number of pages where you need a keyboard configuration, and/or you aren't using an advanced browser. The following example assumes you have 1 page with a form on it (where the keyboard is required), and a second page indicating proper submission.

(1) You need the window names (the "TITLE" tag for the HTML page) - assuming you are in control of these, you should create a small, but unique title to each page you want to trigger on.

(2) Run My-T-Soft Setup | Configuration | Special Options.

(3) Enter the text for the "form" window in the drop-down combo edit box (Application Window Name), e.g. "Please fill out form"

(4) Click on Run EXE radio button, Browse to select Developer's Kit EXE (e.g. C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE)

(5) Add command line parameters (C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE x:20 y:344)

(6) Click OK

(7) Repeat for second window, click on Special Options

(8) Enter the text for the "submitted" window in the drop-down combo edit box (Application Window Name), e.g. "Thank you!"

(9) Click on Run EXE radio button, Browse to select Developer's Kit EXE (e.g. C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE)

(10) Add command line parameters (C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE x:2000 y:344)

(11) Click OK

(12) Test with web pages

(13) Notes: You need at least 2 windows to make operation work as you'd expect. To prevent an infinite loop, the Run EXE runs, and then is locked out until a different Run EXE event is triggered.

(14) The "React to Dialogs..." in My-T-Soft Setup | Configuration | Operation Options must be checked on to allow monitoring active windows.

(15) Only 64 characters from the Window Text (Window Name) are checked, and there is a max of 32 characters allowed in Special Options

(16) The name entered is matched against the whole name, so "Word" will match WordPad, Microsoft Word, WordPerfect, etc.

(17) This will drag on system resources, and may cause problems if you get a large number of entries in Special Options - this will depend on the speed of the system, memory, other applications running, etc., etc. so the exact number your particular system can handle before you notice speed issues will vary. This is really meant for just a handful of specific windows, not a 100 different configurations.

Support Notes

> **Important:** This is a collection of support notes that date back to the original releases, and could be helpful to maintainers of older versions.

This is a list of updates and details about releases since 1.70.

1.90 > Updated for 1.90 Release 5, included 64-bit builds of DEVKIT utils, also compiled/code-signed versions with different manifests, various other updates.

1.79 > Updated for 1.79 release, added Themes, MultiTouch DLL source, Zip updates, HTML help.

1.78R5 > DEVKTDOC moved to manual/HTML/PDF DocBook formatting with single source for all Developer's Kit information, and website Developer's Corner is Developer's Kit documentation

1.78R5 > MultiTouchDLL - source for Multi-Touch/Gesture integration DLL

1.78R5 > Inclusion of THEMES (PaintDLL interface for retail products)

1.78R5 > Zip/Unzip - Info-Zip's source files for zip/unzip capability

1.78R3 > Developer tools for working with Joystick-To-Mouse

1.78R3 > Developer tools, examples in C, C#, Visual Basic for integrating The Magnifier

1.78R3 > Words DLL for external word list control for OnScreen

1.78R3 > Paint DLL for external control of painting My-T-Soft

1.78 > .Net (C#), Java, and J-Sharp examples of integrating with My-T-Soft

1.78 > KBF Dump utility to extract project source files from Build-A-Board KBF's (KeyBoard Files)

1.78 > Keyboard Layouts as modifiable ASM source files (ability to modify Keyboard layouts)

1.78 > AddOnDLL, Logging DLL examples that integrate with running My-T-Soft

1.78 > OnScreen - DevKit basic utilities

1.77 > Documentation now in Help format

1.77 > Update to KeyboardSync (KYBDSYNC)

1.77 > Updates to MS-Access examples

1.77 > Updates to Visual Basic example

1.77 > Updates & new functions to MTSDLL (Support functions as DLL)

1.77 > Updates to DevKit for Build-A-Board layouts (Copy & Configure - CpyCnMTS)

1.77 > Ability to control Tool Bar with Configure MTS (ConfgMTS)

1.77 > New TRIGGER utility for monitoring Windows

1.75 > Developer's Kit for My-T-Soft 2.x (Build-A-Board support)

1.74 > Developer's Kit for My-T-Soft CE (Build-A-Board support)

1.74 > Keystroke executables for Enter, Tab, Back, Escape

1.73 > MTSISTRT Utility for My-T-Soft/My-T-Touch/My-T-Pen - 3 minimized states (button, Icon, Tray Icon) at StartUp

1.73 > EDITSYNC - Handles Physical Keyboard monitoring for OnScreen, and auto-positioning based on text caret

1.73 > STRESS.EXE handles scripted Devkit utilities

1.72 > ShowInfo utilities added - useful tool to show Window Class, Name, Handle of window pointed at & parent info

1.72 > WaitRun added to Developer's Kit - allows delayed execution of DevKit utilities

1.72 > WebSync (Web Page control example & HTML Pages) included

1.72 > (Open from Icon) Open_MTS updated to handle all 3 icon types

1.72 > MTSStart updated with comments to be a good DevKit API usage example

1.72 > (Copy & Configure) CpyCnMTS and (Toggle off-screen & configure) FWCtlMTS utilities have been updated to work properly on NT/2000/XP

1.72 > TurnAway - utility for Head/Eye mouse users (included with OnScreen 1.70)

1.72 > Developer's Corner Web Pages & Info included with DevKit

1.71 > The Devkit folder has a VC6 Workspace that contains all the Developer's Kit utilities.

1.71 > SDSTRMTS has been updated for the Win32 versions - passing a global handle is no longer supported, so a file based "string" is used.

1.71 > New additions include the CTALTDEL (Ctrl-Alt-Delete software emulation for the Secure Attention Sequence (SAS) in Windows NT/2000, restart system in 95/98/Me) - Source code is included.

1.71 > The MTSVBDLL has been generalized and placed in MTSDLL - this also adds the GetXYMTS & GetXYWnd which can retrieve top-left, bottom-right, and width-height for My-T-Soft or a generic window. Also includes an MS-Access MDB with example code for integrating the DLL.

1.71 > The MTSEdit COM control that can manipulate My-T-Soft based on properties set within the control.

1.71 > KeyboardSync has been added to automatically synchronize the My-T-Soft display with the current selected keyboard layout (locale) within Windows.

1.71 > There is also a KeyWatch utility to display scan codes and keystroke information - originally developed for comparison checking & debugging.

The 1.70 release of My-T-Soft / My-T-Pen / My-T-Touch ends the support of Windows 3.x. 1.61 is the last release of My-T-Pen & My-T-Touch that supports 16-bit windows / Windows 3.x. In making the break, the Developer's Kit has been completely updated to support only the Win32 platforms.

Important Support Notes with the 1.78 Release

> **Note:** The Developer's Kit 1.78 Release 2 has been reworked to address many of the shortcomings outlined below. We still recommend you read the following, and be aware that there are areas in the current Developer's Kit that may still refer to the original implementation. Please advise us if you run into sections that need to be updated.

Close to 1 million years ago (or at least a very long time ago) when the predecessor of the current My-T-Soft family of products was first introduced, there was no "Program Files" location. For various reasons, the product installation was placed in the WINDOWS directory as a sub-directory (now a folder). For consistency & compatibility, this installation location was preserved through all the various incarnations of Windows. Now, however, for compatibility with Windows Vista, and to bring the installation location more in line with user expectations, the install location for the program files has been moved, and is now in the "Program Files" folder. Also, for the first time, the user configuration files are separated from the actual installation location (as the new default). This change, in and of itself, was not too major for the product, but did require modifications to various portions of the product code. Because the Developer's Kit is so broad, the release of the product was not delayed to bring the Developer's Kit in line with these changes. So as of the time of the 1.78 release, the Developer's Kit is more in tune with the 1.77 release than the new 1.78 release. Please check for updates, and additions to the Developer's Kit that address these changes. In order to accommodate these expected updates, a release number has been incorporated in the Developer's Kit and its naming. As of the 1.78 release, the Developer's Kit 1.78 is tagged Release 1. Look for future releases in this area as time rolls on. Finally, if there are other areas that you wish to be addressed by the Developer's Kit, please let us know.

Support notes for Kits prior to 1.72

Make sure to check the Technical Support pages on the IMG Web Site for the particular product for any general support issues.

CPYCNMTS.EXE and FWCTLMTS.EXE utilities used C based File functions, and created Read-Only files in NT/2000. The 1.72 update uses only Win32 API calls to resolve this issue. Quality testing was done on Win98, so this made it out the door in 1.71.

The following are notes for Kits prior to 1.71

MOVEWMTS.EXE, MoveWindow issues - In All Developer's Kits prior to 1.50, the final parameter in MoveWindow, the "repaint" flag, is FALSE. For My-T-Pen and My-T-Touch 3.x/95/98 versions, 1.53 or earlier and My-T-Soft NT 1.47 or earlier, this parameter did not affect visual appearances. For My-T-Soft NT 1.49, 1.50 and all versions of My-T-Touch, My-T-Pen, and My-T-Soft 1.60 or greater, for proper

repainting (and refreshing the desktop (or any window uncovered via MOVEWMTS)), the parameter should be set TRUE (this is done in DevKit 1.50).

SDSTRMTS.EXE, 1.51 required for escape sequences (%%x) keystrokes embedded in string.

SDSTRM32.EXE, 32-bit version required for "cooked" keys embedded in string (i.e. [Enter]).

Using the CPYCNMTS / CONFGMTS utilities in Windows 95/98/Me

The source files reference the WINNT location instead of WINDOWS for these platforms (the original My-T-Soft was only for the NT/2000 platforms). In order to fix this, either

A) Modify the source files & recompile (Windows C Compiler required)

B) Create a CONFGMTS.INI file in the \WINDOWS directory (Reference Developer Kit docs regarding this Override)

The file must contain:

[Location]

MTSINIFILE=C:\WINDOWS\MYTSOFT\MYTSOFT.INI

This assumes a default install in Windows 95/98/Me. This issue will be addressed in Dev Kit versions greater than 1.70.

Originally in the Hints and Comments, now moved to here as a historical note:

In Windows 95, an improper shutdown or lockup can force Scandisk to run - because a keystroke from a physical keyboard is necessary, the following workaround resolves this issue. (Note: In Windows 98, there are additional options in the System Configuration via Control Panel for options on Scandisk, etc.)

Edit the AUTOEXEC.BAT and add the line: SCANDISK C: /AUTOFIX /NOSAVE /NOSUMMARY

This will force Scandisk to run everytime Windows 95 begins, and bypass the need for a keystroke if Windows was improperly shutdown - from DOS (i.e. not the command prompt under Windows 95), use SCANDISK /? for other options (and reference SCANDISK.INI) for other details.

Originally in the "How do I open minimized?", now moved to here as a historical note:

Note: DevKit 1.70 adds MTSSTART which may be what you are looking for!

My-T-Touch used as an example / Developer's Kit / Win 95 or newer

(1) Follow steps 2-5 to Open My-T-Touch, then Minimize My-T-Touch (Verify that My-T-Touch is configured for desired minimized state - button or icon)

(2) Create a batch file, e.g. C:\WINDOWS\OPENMTT.BAT:
C:\WINDOWS\MYTTOUCH\MYTTOUCH.EXE[Enter]
C:\WINDOWS\MYTTOUCH\DEVKIT\MINMZMTS.EXE[Enter] [EOF] [EOF]=End Of File, assumes Developer's Kit unzipped in default directory C:\WINDOWS\MYTTOUCH

(3) Add Shortcut to Windows StartUp group - open StartUp, then New|Shortcut|Browse - select C:\WINDOWS\OPENMTT.BAT, name it as desired, select icon, then Finish

(4) Edit Icon in StartUp Group - right-click, properties - change / verify the following: Program tab|Run is set to Minimized|"Close on exit" is On; Screen tab|Usage is set on Window, not Full-screen

(5) If you do not want opening splash screen, add -NoSplash option after MYTTOUCH.EXE in first line of batch file

(6) Many more tricks can be accomplished with simple batch files & Developer's Kit EXE files if control of application is not possible

My-T-Touch / My-T-Pen 1.60 Feature Notes

UserNameNotPassword setting in INI (Version 1.60)

This setting only available in INI file

Forces Windows 95/98 Logon to start (keyboard focus) at User Name field (not Password field).

Set UserNameNotPassword=1 to change default operation. Default is 0 - normally User Name is preserved, and Windows expects password entry.

AllowDomainEdit setting in INI (Version 1.60)

This setting only available in INI file

Original customer specs for Windows 95/98 logon required user lockout of Domain. Recent customer requests asked for Domain edit as an option.

Set AllowDomainEdit=1 to change default operation. Default is 0 - Domain Edit locked from user.

DisableEnter setting in INI (Version 1.53)

This setting only available in INI file

This is ONLY required if there are conflicting programs or virus scans that cause a hang during logon sequence!

The Windows 95 Logon must be checked on in Setup | Configuration

Setting DisableEnter disables the Enter Key from operating during Logon ONLY. Set DisableEnter=1 to force user to logon by Clicking OK.

Review Manual for notes regarding other settings

Force Upper / Lower case for Windows Logon

The Windows 95 Logon must be checked on in Setup | Configuration

In the INI file [Configuration] section: ForcePasswordUpper=1 is required for Uppercase, ForcePasswordLower=1 is required for lowercase

Mixing of cases is not possible (implemented as per customer requests regarding security issues)

Key Enable / Disable & Executable Launch

All settings should be configured in Setup | Configuration | Key Options

Timing can be an issue with EXE Launch - only using Dev Kit EXEs prevents sending keystrokes to wrong application - not an issue if Key is Disabled, and Key press is only used for launching

If changing operations drastically, it may be a good idea to edit INI file and remove all [Key????] settings (or restore INI back to Original Configuration)

Active Window used to Run EXE in Special Options

IMPORTANT: To prevent infinite looping, EXE is only run once UNTIL another Special Setting forces another EXE Launch

Typically you MUST have at least 2 Run EXE Special Settings to have a practical configuration

Run EXE with Calculator Send button

Used to buffer input prior to sending numeric string to application with Send key

Use with Dev Kit EXEs to move MTT/MTP off screen after operation, etc.

Similar to Key Options, BUT no User front-end in Setup

Requires modification of INI file, [CalculatorTape] section

Modify / add CalcSendWithEXE=1

Modify / add Program= to reference EXE to run when Send key used

e.g. Program=C:\WINDOWS\MYTTOUCH\DEVKIT\MOVEWMTS.EXE X:801 Y:400

Timing can be an issue with EXE Launch - using Dev Kit EXEs only prevents sending keystrokes to wrong application

Windows 95/98 Network Logon with My-T-Touch / My-T-Pen

Due to Popular Demand, we have added this capability into our 1.50 release of My-T-Touch & My-T-Pen!! We have also added a Custom Logo option, and enhanced the security features.

Part II. My-T-Soft Developer's Kit

Details specific to IMG's My-T-Soft family of products

Details, samples, code, utilities, and Developer information.

Chapter 3 - My-T-Soft Developer's Kit for Windows contains the core / original Developer's Kit utilities for manipulating and controlling My-T-Soft.

Chapter 4 - My-T-Soft Developer's Kit for Windows CE contains the Developer's Kit utilities for manipulating and controlling My-T-Soft in Windows CE.

Chapter 5 - My-T-Soft Developer's Kit for Linux contains the Developer's Kit utilities for manipulating and controlling My-T-Soft in Linux.

Chapter 6 - OnScreen Developer's Kit for Windows contains Developer's Kit utilities for manipulating and controlling OnScreen (Assistive Technology version of My-T-Soft).

Chapter 3. My-T-Soft Developer's Kit for Windows

My-T-Soft Developer's Kit for Windows Overview

FOLDER: DEVKIT

TYPE: Utilities / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

64-bit IDE: Microsoft Visual Studio 2013

Utilities for implementing My-T-Soft within application development environments (Microsoft Access, Borland Delphi, Visual Basic, and other visual application design environments)

My-T-Soft is used as a generic term for My-T-Touch, My-T-Pen, My-T-Soft, or private label versions of IMG's My-T-Soft family.

Overview

Each utility comes with a C source module, a Resource Script (RC), and a Module Definition file (DEF).

The pre-compiled executables can be integrated into any environment that can launch (i.e. shell, spawn, WinExec, CreateProcess, etc.) an EXE executable program. Use a windowed command prompt (MS-DOS prompt/CMD/Command) to manually work with these executables.

These utilities were developed in response to customer suggestions, wants, needs, and by reviewing the capabilities of various high-end application development tools. The executable approach allows use of Shell, WinExec, CreateProcess, or other API (Application Programming Interface) calls common to any robust environment. The inclusion of the source code allows easy integration into applications developed closer to the Windows API. The instructional information is also useful in allowing developers to create functional applications quickly.

The order of the utilities is in the order they were created.

Be sure to read the Hints & Comments for Developers, or in the Frequently Asked Questions (FAQ) section.

There are various utilities. Some pair up as opposites, some are more powerful than others. Be sure to read through the following details to gain a good understanding of the ability of these tools to assist you in your development efforts. Please feel free to contact us with other ideas or needs.

Version Release Notes

The 1.90 release adds 64-bit versions, and a complete solution with all utilities based in Visual Studio 2013. There are pre-built EXEs that have been code-signed (both 32-bit/64-bit), as well as built with different manifests to address permissions, security-levels, and User Account Control.

The 1.77 release updates CONFGMTS to handle All panels, including the tool bar panel. MOVEWMTS now opens a minimized My-T-Soft before executing move. A new utility called TRIGGER waits on a window to disappear before triggering a command.

The 1.72 updates FWCTLMTS and CPYCNMTS to only use Microsoft Win32 APIs for file manipulation. The 1.71 version was quality tested under Win98, but these same (Visual C++ Version 6, SP3) compiled EXE files did not operate properly under NT/2000/XP. 1.72 adds WaitRun, WCRemap, and updates MTSStart & Open_MTS.

The 1.71 release is compiled for Win32 - the #define MYTWIN32 is added in the MYTSOFT.H, and each C source file uses this to build the appropriate target. The 1.71 also includes the MTSDLL Dynamic Link Library with examples for close integration with VBA (Visual Basic for Applications) - see the VBASIC projects, or the MDB in MTSDLL for coding samples.

The 1.70 release adds the StartUp as minimized utility. This resolves taskbar icon artifacts in Win 95/98 when opened for logon, along with minimizing to an icon. (NOTE: In the 1.70 release of My-T-Soft, the StartupInfo properties are referenced, and the shortcut can be set to Open as a minimized window - the button will be used) Also note that Size 12 is a virtual barrier for re-sizing on the fly from the larger sizes (i.e. greather than size 12) to the smaller sizes (i.e. 12 or less). Do not cross this barrier one way or the other without first establishing a painted keyboard at size 12. The larger sizes are all keyed off of the dimensions of size 12, so corruption of the valid KBF file is possible (probable!) if crossing size 12!

The 1.50 release adds the FindWindow & Control My-T-Soft that allows the toggling of the window off-screen, and on-the-fly configuration when called up by the user. Goes well with the Control My-T-Soft button utility (also included). The MoveWindow (MOVEWMTS) utility changes a parameter for cleaner operation with the My-T-Soft 1.50 release.

The 1.40 Release adds the Send String through My-T-Soft, but this is only fully functional in Version 1.52 of My-T-Touch / My-T-Pen.

The 1.30 release combines the previous My-T-Soft, My-T-Touch, and My-T-Pen utility kits. This was done to allow addition of customized versions, and to allow developers to extend the capabilities of the executables to other windows in the system. My-T-Soft is used throughout, but please note that the appropriate product name (window name) must be in the FindMyTSoft function provided internally in the source code.

DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

Support info:

These codes indicate which utility is available for which platforms

My-T-Soft 1.x = My-T-Soft / My-T-Touch / My-T-Pen versions prior to 2.00

- In DEVKIT folder

My-T-Soft 2.x = My-T-Soft versions for Build-A-Board - In DEVKIT2 folder

My-T-Soft CE = My-T-Soft versions for Windows CE - In DEVKITCE folder, in processor sub-folder

Developers Kit Utilities & Quick Description>

Close My-T-Soft Window

This sends the WM_CLOSE Window Message to shut down the software

Show window options for My-T-Soft

Example of using ShowWindow API call

Minimize My-T-Soft to Icon, Button, or Tray icon

Based on minimize settings (Setup | Configuration), minimizes window

Open My-T-Soft from minimized state

Restores to open window from minimized state

Move My-T-Soft Window

Position window at X/Y location

Set Cursor Position for My-T-Soft

Uses SetCursorPos API to move mouse cursor

Configure My-T-Soft Panels & Size

Open/close panels, and resizes from command line

Configure My-T-Soft on the fly (from pre-existing configurations)

Restores saved files for immediate configuration changes

Toggle My-T-Soft off-screen or on-screen in specified configuration

One command to move off-screen, or restore from file based on position

Send String (type) through My-T-Soft

Externally (programmatically) send keystrokes through My-T-Soft

Save Position of My-T-Soft

Saves current position to KEYBOARD.KBF

Save Settings of My-T-Soft

Saves panels/size configuration to KEYBOARD.KBF

Restore Position of My-T-Soft

Restores saved X/Y location from KEYBOARD.KBF

Restore Settings of My-T-Soft

Restores panels/size from KEYBOARD.KBF

My-T-Soft Startup Options

Manipulating My-T-Soft at Startup example

Wait and then Run Program/Utility

Utility to delay execution of another program

Wait and then Close Window Utility

Utility to close an arbitrary window

Trigger - Wait while Window, then Run utility

Utility to monitor a window, then run a command if window closes

WordComplete Remapping Utility (OnScreen only)

Trigger WordComplete selections in an alternate way

My-T-Soft Startup as Icon

Another Startup Utility example

What I really want (My-T-Soft 1.x)

Overview of how to work on-the-fly changes

Close My-T-Soft Window

Name: Close My-T-Soft

Executable name: CLOSEMTS.EXE

Command Line Arguments: NONE

Description: Executing this file will close the My-T-Soft Executable.

Notes: Requested by an MS-Access Developer.

Support: My-T-Soft 1.x, My-T-Soft CE, My-T-Soft 2.x

Show Window Options for My-T-Soft

Name: ShowWindow My-T-Soft

Executable name: SHOWWMTS.EXE

Command Line Arguments: HIDE or SHOW or MINIMIZE or MAXIMIZE

Example: SHOWMTS MINIMIZE

Description: Executing this file will affect the My-T-Soft Executable by calling the Windows API

ShowWindow call with the indicated parameter.

Notes: Provided for instructional use - for various actions within ShowWindow there are different internal calls that are preferred.

Minimize My-T-Soft to Icon, Button, or Tray icon

Name: Minimize My-T-Soft

Executable name: MINMZMTS.EXE

Command Line Arguments: NONE

Description: Minimizes My-T-Soft using configured method.

Notes: Use OPEN_MTS.EXE to return My-T-Soft to original state.

Support: My-T-Soft 1.x, My-T-Soft CE, My-T-Soft 2.x

Open My-T-Soft from Minimized State

Name: Open My-T-Soft

Executable name: OPEN_MTS.EXE

Command Line Arguments: NONE

Description: Opens My-T-Soft from minimized state using My-T-Soft API.

Notes: Opposite of MINMZMTS.EXE

1.72 - Modifies source to accommodate all 3 icon types (Button, Icon, Icon on Taskbar (Shell Tray)).

Support: My-T-Soft 1.x, My-T-Soft CE, My-T-Soft 2.x

Move My-T-Soft Window

Name: Move My-T-Soft

Executable name: MOVEWMTS.EXE

Command Line Arguments: X:??? Y:???

Example: MOVEWMTS x:200 y:20

Description: Moves My-T-Soft window to indicated x & y position.

Notes: Extremely useful for complex applications. Syntax requires no spaces between x: and y: and position. Some error checking in utility, upper or lower case OK. You may wish to always position My-T-Soft so the Tool bar panel (logo / menu / minimize) are mostly or all the way off the screen. Remember that by turning on the Operator mode, you can disable everything, and the minimize button can be disabled separately. 1.77 Notes - at the request of a developer, forcing the window open if it is in an iconized mode has been incorporated in the MOVEWMTS code (i.e. from OPEN_MTS). 1.77 Notes - now that you can turn off the Toolbar, the Operator mode is not as critical, but still may be useful to lock users out of Setup.

Support: My-T-Soft 1.x, My-T-Soft CE, My-T-Soft 2.x

Set Cursor Position for My-T-Soft

Name: SetCursor for My-T-Soft

Executable name: STCURMTS.EXE

Command Line Arguments: X:??? Y:???

Example: STCURMTS x:200 y:20

Description: Moves cursor to indicated x & y position.

Notes: Focus issues and shell or exec implementation of My-T-Soft Utilities can sometimes result in the wrong window having the focus. By moving the cursor over My-T-Soft, code in My-T-Soft can resolve some focus issues. Syntax requires no spaces between x: and y: and position. Some error checking in utility, upper or lower case OK.

Support: My-T-Soft 1.x

Configure My-T-Soft Panels & Size

Name: Configure My-T-Soft

Executable name: CONFGMTS.EXE

Command Line Arguments: /K[-] /E[-] /N[-] /C[-] /M[-] /W[-] /T[-] /B[-] /I[-] /Q[-] /G[-] /S:??

Example: CONFGMTS /k-/e-/w/m/s:10 Closes Keyboard & Edit panel, opens Windows Control & Macro Panel, sizes to 10.

K = Keyboard (alphanumeric) panel

E = Edit panel

N = Numeric panel

C = Calculator panel

M = Macro panel

W = Windows control panel

T = Tool / Control panel

B = Toolbar (3 icon panel)

G = Magnifier panel

I = System Info panel

Q = QuickHelp panel

- = after any of the above indicates the panel will be closed

S:?? = Size from 1 - 12

Description: Gives complete control over appearance. Used in conjunction with MOVEWMTS allows different looks / interfaces for complex applications.

All parameters can be used at the same time, so quick configuration changes can be accomplished. Use MOVEWMTS after CONFGMTS to place in new position. You may wish to change Setup option to Snap-Out panels to decrease change-over time. Alternatively, you may wish to CLOSEMTS, then CONFGMTS, then re-open MYTSOFT.EXE. For preset configurations that are fixed, see "What I really want..." below. Also see the Copy & Configure utility (next).

Support: My-T-Soft 1.x

Configure My-T-Soft on the fly (from pre-existing configurations)

Name: Copy and Configure My-T-Soft

Executable name: CPYCNMTS.EXE

Command Line Arguments: [drive:][path]FileName

Examples: CPYCNMTS NUM.CFG, CPYCNMTS D:\PANELS\KEYPAD.CFG

Allows easy selection of multiple configurations without need for a batch file (see "What I really want...")

Description: Gives complete control over appearance - cleaner switch over than CONFGMTS. Does not require separate batch file. If no path is given within the FileName specification, the default My-T-Soft directory is used - however, for guaranteed operation, a well formed path (with short file names) is required. The KEYBOARD.KBF file that is overwritten defaults to the default

My-T-Soft directory. You must use the CONFGMTS.INI file to override the default install directory. This utility is a shortcut to the batch approach described in detail below. 1.72 notes - uses ONLY Win32 API calls to manipulate files to resolve "Read-Only" problems in NT/2000/XP.

1.77 notes - to resolve issues with 101 & 104 layouts, there is support for matching INI files with this utility. Since some configurations require changes in both the KBF and the INI file, the following approach is used:

For a particular specified KBF file, a matching INI will be looked for using the INI extension instead of the KBF extension. For example, for NUM.CFG, CPYCNMTS will look for NUM.INI.

If the INI file exists, it will overwrite the current INI file, and CPYCNMTS will send a message to My-T-Soft to refresh from the INI file. Then the switch over to the new layout configuration will occur.

Things such as Show & Hide keys, layout language, and other things specific to the INI file will be preserved. The 1.80 release will support a comprehensive single file configuration, breaking free of these original design issues.

Support: My-T-Soft 1.x

Toggle My-T-Soft off-screen or on-screen in specified Configuration

Name: FindWindow & Control My-T-Soft

Executable name: FWCTLMTS.EXE

Command Line Arguments: [drive:][path]FileName

Example: FWCTLMTS NUM.CFG, FWCTLMTS D:\PANELS\KEYPAD.CFG

Description: Gives complete control over appearance with the added ability to move the displayed panels off-screen if already on-screen. This is a modified version of CPYCNMTS that has the following logic: If My-T-Soft is visible, move it off-screen and do not process command line.

If My-T-Soft is not visible (off-screen), then process command line.

Notes: This allows a toggle selection for the user using the same action (button on application, CNTRLMTS buttons, etc.). Bring up the keyboard, then put it away. Note if My-T-Soft is off-screen, then this acts exactly as the CPYCNMTS utility. If My-T-Soft is visible, then its added feature of moving the software off-screen occurs. 1 monitor assumed (for multiple monitor configurations, modification of the source is probably required).

1.72 notes - uses ONLY Win32 API calls to manipulate files to resolve "Read-Only" problems in NT/2000/XP.

1.75 note: For My-T-Soft 2.x, the "visible" position will be the saved position within the KBF file. There is no command line supported with this version.

Support: My-T-Soft 1.x, My-T-Soft 2.x

Send String (type) through My-T-Soft

Name: Send String Through My-T-Soft

Executable name: SDSTRMTS.EXE / SDSTRM32.EXE

Command Line Arguments: Text String (with internal key escapes as text)

Description: This communicates with My-T-Soft and hands off a text string to be processed and typed.

Notes: Knowledge of internal key escapes required for full keyboard capability through text string. See %%???? options in User Guide. (In the 32-bit version, the ability to use the user-friendly codes such as [Enter], [Down], etc. is available)

With the 1.5x versions that have the key enable/disable/launch capability, the entire alpha & edit panels can be remapped.

32-bit versions require SDSTRM32.EXE (Note: In DevKits 1.71 and later, SDSTRMTS.EXE is the same as SDSTRM32)- this file MUST reside in the same directory as MYTSOFT.EXE for proper operation! Also, the 32-bit version does not read the raw formats. Use the Save option in Build-A-Macro, Keystroke Macro, Zoom as the string format for SDSTRM32.EXE.

1.71 Note: The conversion of all devkit utilities to 32-bit means that the default SDSTRMTS.EXE is the same as SDSTRM32.EXE.

Support: My-T-Soft 1.x

Save Position of My-T-Soft

Name: Save Position of My-T-Soft

Executable name: SVPOSMTS.EXE

Command Line Arguments: NONE

Description: Saves current position, for use with REPOSMTS, or upon re-opening of My-T-Soft.

Notes: Opposite of REPOSMTS.EXE. Same as Menu Option, Save Current Position.

1.75 Note: For My-T-Soft 2.x versions, the saved position should be used only to affect the opening of the layout - SVPOSMTS, then CLOSEMTS. Trying to coordinate multiple positions with SVPOSMTS & REPOSMTS will not seem to work as intended - use MOVEWMTS for handling multiple screen positions of a layout.

Support: My-T-Soft 1.x, My-T-Soft 2.x, My-T-Soft CE

Save Settings of My-T-Soft

Name: Save Settings of My-T-Soft

Executable name: SVSETMTS.EXE

Command Line Arguments: NONE

Description: Saves current settings, for use with RESETMTS, or upon re-opening of My-T-Soft.

Notes: Opposite of RESETMTS.EXE. Same as Menu Option, Save Current settings.

Support: My-T-Soft 1.x

Restore Position of My-T-Soft

Name: Restore Position of My-T-Soft

Executable name: REPOSMTS.EXE

Command Line Arguments: NONE

Description: Restore saved position.

Notes: Opposite of SVPOSMTS.EXE. Same as Menu Option, Restore Current Position.

1.75 Note: For My-T-Soft 2.x versions, the saved position should be used only to affect the opening of the layout - SVPOSMTS, then CLOSEMTS. Trying to coordinate multiple positions with SVPOSMTS & REPOSMTS will not seem to work as intended - use MOVEWMTS for handling multiple screen positions of a layout.

Support: My-T-Soft 1.x, My-T-Soft 2.x, My-T-Soft CE

Restore Settings of My-T-Soft

Name: Restore Settings of My-T-Soft

Executable name: RESETMTS.EXE

Command Line Arguments: NONE

Description: Restores saved settings.

Notes: Opposite of SVSETMTS.EXE. Same as Menu Option, Restore Current settings.

Support: My-T-Soft 1.x

My-T-Soft Startup Options

1.71 Note: This really no longer applies because of the changes to the 1.70 release of the software - left intact for reference.

1.72 Note - reworked to match changes to 1.70 My-T-Soft, added comments to illustrate usage of Developer's Kit API.

Name: Start up, clean icon, and minimize to button

Executable name: MTSSTART.EXE

Command Line Arguments: NONE, or ms delay (250 - 10000)

Description: Minimizes to icon on taskbar, opens off screen, minimizes to button

Notes: The default value is set at 750 ms. Adding a numeric command line to the startup shortcut will modify the delay. The range of 250 to 10000 is accepted by the software (see source). This expects to be in the startup group, and was originally implemented to configure the keyboard for Win 95 / Win 98 after logon use. The minimize to icon step was used to remove icon artifact on taskbar.

Support: My-T-Soft 1.x

Wait and then Run Program/Utility

Name: Wait & then Run Utility

Executable name: WAITRUN.EXE

Command Line Arguments: [Wait time],[path][file]"Program Name"

Description: Wait 5 seconds, or specified Wait time, then execute given program name

Notes: Command Line notes: The comma is used as a delimiter. If it is not present, then the default 5000 milliseconds (ms) are used, and the complete command line is treated as the program to execute. If the comma is present, the first number is converted into a value, and this value is checked, then treated as the wait time valid wait time values are: 1000-30000.

Security Notes (NT/2000/XP): Default security rights are used, and if you try to execute a full path name, you may get Access Denied. Ideally you should run WaitRun in the same directory as the desired utility. If you experience security errors, we have put default security descriptors in the source, and you may need to modify these for your particular application, or work with the user's permissions.

Support: My-T-Soft 1.x (Note: not tied to version)

Wait and then Close Program/Utility

Name: Wait & then Close Utility

Executable name: WAITCLOSE.EXE

Command Line Arguments: [Wait time]

Requires: WAITCLOSE.INI

Description: Wait optional specified Wait time, then Close other windows based on entries in WAITCLOSE.INI

Notes: The below is a copy of the WAITCLOSE.INI file that has comments embedded in it. The INI must be in the same folder as the EXE file, and there must be at least some text for the title match or class match. CloseAll is default off, so only 1 window will be closed. Note that a simple match for PartofWindowsTitle with CloseAll=1 may match a lot of windows and cause other problems

The utility uses the Windows API GetWindowText and GetClassName to obtain the class names while enumerating all normal (application type) windows.

WAITCLOSE.INI

This WAITCLOSE.INI is used by the My-T-Soft Developer's Kit WAITCLOSE to identify a caption and/or class of a window to close

If the entry is empty (e.g. PartofWindowTitle=), then it will not be used

If both entries are empty, then WaitClose will issue a warning!

The window title is a substring search, while the class must match exactly!

For help finding out further info about actual values in running window, refer to the ShowInfo.exe in DevUtil folder (or Windows Spy utilities)

[Settings]

PartOfWindowTitle=

PartOfWindowTitle if it contains some text will be used in a case insensitive match based on the Window title (caption bar text) from GetWindowText

Word would match "WordPad","Microsoft Word","WordPeferect","E-mail to WordSmith", etc.

Examples:

PartOfWindowTitle=Document

PartOfWindowTitle=Microsoft Word

WindowClass=

The WindowClass must be a complete (case insensitive) match to a window class

This can be used to ensure that only the window(s) you are interested in will be matched

Examples:

WindowClass=IEFrame

WindowClass=Notepad

CloseAll=0

If CloseAll=1, then all windows will be enumerated and any matches will be send a WM_CLOSE message. If CloseAll=0 then the first match will stop the enumeration of top level windows.

Support: My-T-Soft 1.x (Note: not tied to version)

Trigger - Wait While Window, then Run - Also Launch / Activate App

This utility has various options - originally it was meant to wait while a window was active, then trigger a DevKit utility once the window was dismissed. This is part of the common situations that arise when trying to integrate My-T-Soft with existing applications. Because of the structure of the utility, it also is used to launch an Application, but if the Application is already running, it will bring the running Application to the foreground (Active window).

Name: Trigger - Wait while Window, then Run Utility

Executable name: TRIGGER.EXE

Command Line Arguments: [option character]{[Class],[Text]}![command to run]!

Command Line notes: Requires Command Line of form {Class,Window Text}!command.exe! Start with ~ to display debug info, e.g. ~{Class,Window Text}!command.exe!

Start with @ to run command (launch application) if no Window found, or activate if Window found, e.g. @{Class,Window Text}!command.exe!

Start with * to close any running TRIGGER Windows, e.g. *Close existing processes

Start with + to close any running TRIGGER Windows, then start new TRIGGER

Examples:

TRIGGER {#32770,Enter Name}!MOVEWMTS x:10000 y:10000!

(When Enter Name dialog is closed, move My-T-Soft off-screen)

TRIGGER ~{#32770,Error}!MOVEWMTS x:100 y:400!

(Show debug info, When Error dialog is closed, move My-T-Soft on-screen)

(Can use Debug with other special characters, but must be second position, e.g. @~ or +~)

TRIGGER *Close Existing TRIGGERS

(Any TRIGGERs waiting on a window will be sent a close message)

TRIGGER +{#32770,Error}!MOVEWMTS x:100 y:400!

(Close any existing TRIGGER processes, then When Error dialog is closed, move My-T-Soft on-screen)

When waiting on a window, there is buffer time!

TRIGGER waits 20 seconds for specified Window to appear

If this FAILSAFEDELAY entry runs out, TRIGGER will close.

This allows a holding period for the desired window to appear.

Notes:

Must include comma between {} to record class/text!

To use NULL value for Class/Text, use character '0', e.g. {SciCalc,0}, or {0,Document}

Description: If specified Window is found, will wait until window no longer exists, or if not an exact match, will match text in Caption, and wait until Window is no longer the foreground window, then it will execute the specified command. There is a 20 second delay to wait for a window match, otherwise TRIGGER will close if no match occurs.

Description (@ option): If specified Window is found, the Windows API SetForegroundWindow will be called to bring the specific app/window to the foreground (Active window). If the specified Window is not found, the command is executed (i.e. launch the application). This mode could also be used to trigger some other utility for the case when the specified window is not running, but was designed to launch the application associated with the window class/text.

Notes: This utility was requested by a DeltaV developer, and has been modified to handle more options. If run with no command line, a message box will be displayed indicating command line options.

Support: My-T-Soft 1.x (Note: not tied to version)

WordComplete Remapping utility (OnScreen only)

Name: WordComplete External Button Trigger

Executable name: WCREMAP.EXE

Command Line Arguments: b:1 or b:2 or b:3 or b:4 or b:5

Description: Triggers internal action to emulate a WordComplete button press

Notes: Customer wanted to trigger WordComplete buttons via function keys. Using Key Options, disable Key stroke, enable KEY Exe, and enter appropriate WordComplete button (numbered 1 thru 5, top to bottom).

Support: My-T-Soft 1.x

My-T-Soft Startup as Icon

1.77 Note - IMPORTANT: You should configure the startup icons properties, and use the Run: property as Minimized for best results. My-T-Soft supports the STARTUPINFO ShowWindow entry that comes from a shortcuts properties.

1.73 Note - MTSSTART reworked to match changes to 1.70 My-T-Soft, and allow clean opening as an icon, or as an icon in the tray.

Name: Start up as icon

Executable name: MTSISTRT.EXE

Command Line Arguments: NONE, or ms delay (50 - 10000)

Description: Visually opens as an icon - several tricks used to accomplish this (Force open off-screen with no-splash, iconize, then stay around to force back to correct screen position (KEYBOARD.KBF))

Notes: The default value is set at 750 ms. Adding a numeric command line to the startup shortcut will modify the delay. The range of 50 to 10000 is accepted by the software (see source).

This expects to be in the startup group, and the shortcut to the keyboard should NOT be used (this utility calls WinExec to create keyboard process).

There is also a second compile to handle the shell icon (in tray) - the source code included has these lines commented out at line 333.

Name: Start up as icon in tray (Shell Icon)

Executable name: MTSISTR2.EXE

Command Line Arguments: NONE, or ms delay (50 - 10000)

Description: Visually opens as an icon in the tray

Notes: The default value is set at 750 ms. Adding a numeric command line to the startup shortcut will modify the delay. The range of 50 to 10000 is accepted by the software (see source).

This expects to be in the startup group, and the shortcut to the keyboard should NOT be used (this utility calls WinExec to create keyboard process).

Support: My-T-Soft 1.x

What I really want... (My-T-Soft 1.x)

1.72 note - This is dated - it is only included because we have customers running older versions - we recommend you use the MTSDLL or the CPYCNMTS/FWCTLMTS utilities.

For developers who need several different configurations, and desire a seamless changeover, the following steps will enable a clean change-over from one configuration to another. It is implemented as a batch file, and the steps required to setup for this must be followed in exact detail.

Also see new Copy and Configure utility above. (CPYCNMTS & FWCTLMTS)

This batch method, the CPYCNMTS and FWCTLMTS utilities expect saved configurations. Use Steps 1 through 3 below to obtain these configuration files.

1) Configure My-T-Soft with the desired panels.

2) Save Settings & Position (either via the menu, or with SVSETMTS & SVPOSMTS with a windowed command prompt).

3) Copy the KEYBOARD.KBF (My-T-Soft Installation directory) to a distinctive name (e.g. NUMERIC.CFG, KEYBOARD.CFG, or MACRO.CFG).

4) Repeat steps 1 through 3 as required for as many different configurations as needed.

5) Create BATCH files (or script, or code) for each configuration.

EXAMPLE:

FILE: NUMERIC.BAT

copy \windows\mytsoft\numeric.cfg \windows\mytsoft\keyboard.kbf

resetmts.exe

FILE: MACRO.BAT

copy \windows\mytsoft\macro.cfg \windows\mytsoft\keyboard.kbf

resetmts.exe

FILE: KEYBOARD.BAT

copy \windows\mytsoft\keyboard.cfg \windows\mytsoft\keyboard.kbf

resetmts.exe

6) Run Batch files at appropriate times to change layouts.

Version 1.77, July 7, 2003

Chapter 4. My-T-Soft Developer's Kit for Windows CE

My-T-Soft Developer's Kit for Windows CE

FOLDER: DEVKITCE

TYPE: Utilities / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Embedded Visual C++ 3.00

The Windows CE executables were compiled with Microsoft's Embedded Visual C++ 3.00 compiler. As with the Win32 versions, the source code is included. The following platforms have been pre-compiled and are available as executables:

Processor Folder

ARM ARMRel

MIPS MIPSRel

SH3 SH3Rel

SH4 SH4Rel

X86 Emulation X86EmRel

The executables can be found under the DEVKITCE folder, the Utility Folder, and then the appropriate processor's folder.

The following utilities have been reworked to work with My-T-Soft 2.00 or higher:

CLOSEMTS - Close My-T-Soft

MOVEWMTS - Move My-T-Soft Window

MINMZMTS - Minimize My-T-Soft

OPEN_MTS - Open My-T-Soft from minimized state

REPOSMTS - Reposition My-T-Soft (to Saved Position)

SVPOSMTS - Save current position of My-T-Soft

For more details about these utilities, see My-T-Soft Developer's Kit

Version 1.74 - April 3, 2002

Chapter 5. My-T-Soft Developer's Kit for Linux

My-T-Soft Developer's Kit for Linux Overview

FOLDER: DEVKITLINUX

TYPE: Utilities / Stand-alone Executable

SOURCE: Not included

LANGUAGE: C / Windows API

IDE: None

The Linux executables were compiled for 32-bit Intel x86 instruction set - they will work on both 32-bit and 64-bit platforms (x86/x86_64/AMD64).

These are a few utilities that are commonly used to manipulate the My-T-Soft keyboard window from external applications or tools. The executables can be found under the DEVKITLINUX folder.

closemts - sends [CMD:CLOSE] command to close the running software - Close My-T-Soft

movewmts - moves the window based on 2 parameters x:??? y:??? - Move My-T-Soft Window

minmzmts - sends [CMD:MINIMIZE] command to minimize the open software - Minimize My-T-Soft

open_mts - sends a MapWindow event to the icon to simulate opening the icon - Open My-T-Soft from minimized state

svposmts - sends [CMD:SAVEPOS] command to save the current position for the current keyboard - Save current position of My-T-Soft

typefile - sends the command line, or the text from within a specified file

Notes/Examples

The files are executable, binary 32-bit files (gcc target i486-linux-gnu) If they do not get extracted with the executable bit, in the devkitlinux directory use 'chmod +x *' to set them so they will run.

Version 2.20 - Jan 20, 2012

closemts - Close My-T-Soft

closemts

Can be used whether the keyboard is open or minimized

Example - close My-T-Soft: ./closemts[Enter]

movewmts - Move My-T-Soft Window

movewmts

Must specify X: or x: or Y: or y: on the command line for any action. If command line is blank, nothing will occur. If X: or Y: is used without the other, 0 will be used for the missing parameter.

Example - move My-T-Soft to 540,295: ./movewmts x:540 y:295[Enter]

Example - move My-T-Soft to 0,295: ./movewmts y:295[Enter]

Example - move My-T-Soft to 540,0: ./movewmts x:540[Enter]

svposwmts - Save Position of My-T-Soft

svposmts

Used to save position of current keyboard in KBF. Sends [CMD:SAVEPOS] which updates current KBF file. This can be used multiple times if new KBFs are loaded using [CMD:NKBF=??]. Can be used in conjunction with movewmts to position, then save.

Example - save My-T-Soft's current position: ./svposmts[Enter]

minmzmts - Minimize My-T-Soft Window

minmzmts

Sends [CMD:MINIMIZE] command to minimize the open software. This should only be used when the keyboard window is visible.

Example - minimize open My-T-Soft: ./minmzmts[Enter]

open_mts - Open (Restore) My-T-Soft Window from minimized state

open_mts

Sends a MapWindow event to the icon to simulate opening the icon. This should only be used when the keyboard is iconized. Requires Expose event to repaint and properly open keyboard window and remove icon. Window manager may affect operation depending on keyboard size, position, active windows, and possibly other factors.

Example - open minimized My-T-Soft: ./open_mts[Enter]

typefile - sends command line or file through My-T-Soft

typefile

Sends the command line, or the text from within a specified file. The logic takes the command line and sees if a file exists. If a file exists, it will be opened, read, and then sent to MacroBat to type. If there is not an existing file, the command line will be sent to MacroBat to type. The file should contain the macro to type in a single line. The current limit is 255 characters.

Example - type Hello world with typefile: ./typefile [Shift-Down]h[Shift-Up]ello world[Enter]

Example - change layout via text in file: ./typefile keyboard1.txt[Enter]

Example - change layout via text in file: ./typefile keyboard2.txt[Enter]

Where keyboard1.txt contains:

[CMD:NKBF=FULL_LAYOUT.KBF]

keyboard2.txt contains:

[CMD:NKBF=NUM_LAYOUT.KBF]

Chapter 6. OnScreen Developer's Kit for Windows

OnScreen Developer's Kit for Windows

FOLDER: ONSCREEN

TYPE: Utilities / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

Refer to the OnScreen and the Developer's Kit in the Release Notes section for important information about OnScreen.

The executables and source can be found under the ONSCREEN folder, the utilities can be built via the ONSCREEN.DSW Visual Studio 6 workspace.

The following utilities have been reworked to work with OnScreen 1.75 or higher:

CLOSEMTS - Close My-T-Soft

MINMZMTS - Minimize My-T-Soft

OPEN_MTS - Open My-T-Soft from minimized state

SHOWONS is a modified version of OPEN_MTS that will open OnScreen if the software is not running, or do the default open_mts functionality if it is running.

For more details about these utilities, see My-T-Soft Developer Kit

Version 1.78 - May 4, 2007

Part III. My-T-Soft Developer Tool and Utilities

Developer Tools, and Utilities for Developers, Users

Details, samples, code, utilities, and Developer information.

Chapter 7 - Language and Platform Examples has specific examples based on a particular programming language, or based on certain applications.

Chapter 8 - Developer Tools and Examples outlines available utilities specific to Developers.

Chapter 9 - User Utilities covers utilities that may be useful for end-users, but are not necessarily development oriented.

Chapter 10 - System Utilities contains system specific utilities.

Chapter 7. Language and Platform Examples

Visual Basic Examples / DLL of DevKit

FOLDER: VBASIC

TYPE: Visual Basic project integrated with C based DLL

SOURCE: INCLUDED

LANGUAGE: Visual Basic/Forms, and C / Windows API for DLL

IDE: Microsoft Visual Basic 6 / Microsoft Visual C++ 6

Welcome to the Visual Basic Example set for Developers!

1.77 Notes

Due to various changes in the DevKit utilities, and updates to the MTSDLL.DLL, this example is dated - refer to MTSDLL.DLL for code & functionality and the MS-Access examples for up-to-date VB examples.

Refer to VBASIC6 for a folder with the project converted to Visual Basic 6 and updated to work with MTSDLL.DLL. The original VBASIC folder has been preserved for compatibility with older versions of Visual Basic.

If there are any specific questions, please refer to the source code provided, and the original notes below.

This is an add-on to the Developer's Kit for IMG's My-T-Soft, My-T-Touch, and My-T-Pen to allow Visual Basic Developer's to access & Use the Developer's Kit directly from Visual Basic.

MicroSoft Visual Basic Version Notes

This was built in VB 4.0 for maximum compatibility if you open the project in later versions of Visual Basic, just say OK, OK, OK, OK, etc. to convert to the current version you are running.

Installation

The files are delivered in the Zip format (VBASIC.ZIP).

Default Install Locations:

My-T-Soft = C:\WINNT\MYTSOFT

Extract VB Examples to: C:\WINNT\MYTSOFT\VBASIC

My-T-Touch = C:\WINDOWS\MYTTOUCH

Extract VB Examples to: C:\WINDOWS\MYTTOUCH\VBASIC

My-T-Pen = C:\WINNT\MYTPEN

Extract VB Examples to: C:\WINDOWS\MYTPEN\VBASIC

To Install, do the following:

1) Windows Explorer to default install location

2) Create New Folder called "VBASIC (Right-click, New, Folder, then change "New Folder" to "VBASIC")

3) Copy VBASIC.ZIP (download file) to the new VBASIC folder

4) PKUNZIP or WinZip to extract files from VBASIC.ZIP

Some of the features assume that the Developer's Kit has already been "installed". If the "Send Text Thru" or the "External Control" examples are not operating properly, then install the DevKit:

Start Menu | Programs | My-T-???? | Install Developer's Kit

Note: Only My-T-Soft Version 1.50 & above, or NT/2000 Versions supports advanced Send Text Feature

Operation

The appropriate Project name (MYTSOFT.VBP, MYTTOUCH.VBP, or MYTPEN.VBP) should be run from a "VBasic" folder under the install directory of matching product.

****

DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

****

There is a fair amount of error checking, but every possibility has not been addressed. If the default installation of all components has been done, then the operation should be straightforward & error free. If there are operational problems, then it is assumed that as a Visual Basic developer, fixing the error will not be a problem. You may contact IMG at "devkit@imgpresents.com" or check our website "http://www.imgpresents.com" for any news or updates, but technical support is not offered for the Developer's Kit utilities - source code is provided, and should be referenced.

Once in the Visual Basic Environment, simply "Run" to engage the form & start Operation. Click on buttons to see operational features, "Stop" operation and review Visual Basic Form / Properties / Controls & Properties / Module code, etc. See MTSVBDLL.C for source code of C based DLL.

1.71 Note - this DLL has been generalized - see the MTSDLL for the "future" of this DLL.

Operation Notes

Load Event

If the MTSVBDLL.DLL is not found in the current directory, and InputBox will appear to "ask" for its location.

Buttons / Text Fields

Launch - This will launch the appropriate product based on the project loaded. By default (See DLL Source code) My-T-Soft is attempted, if it fails, then My-T-Touch is attempted, if it fails, then My-T-Pen is attempted. If these fail, then it is assumed the default install locations have not been used, and the LaunchFromCurrentMTS function is called with the current directory, and the parent folder (i.e. the folder that contains the VBASIC folder) is used as the location of the product. If the My-T-Soft keyboard is NOT launched, either:

a) Modify the source code to fit your requirements

-or-

b) Run the My-T-Soft product directly...

Close - this will Close the running keyboard software

Minimize to Button - will send the message to minimize to button

Open from Button - will send the message to open from the button state

MoveWindow / text fields X & Y - enter appropriate values in the text fields, and click on button to move keyboard window to that (Top/Left) X, Y location

Set Cursor Position / text fields X & Y - this will move the cursor (mouse cursor) to X, Y location

Save Settings - Saves the current configuration / panels open on the keyboard software

Save Position - Saves the Top/Left X, Y position of the keyboard software

Restore Settings - restores the saved panels & position

Restore Position - restores X, Y location (Top/Left) of keyboard software with current configuration preserved

Send Text Through My-T-Soft - This is an advanced feature of the My-T-Soft version - the VB example does not back support My-T-Touch or My-T-Pen, although this could work with some modification - this is left as an exercise for the developer. The compatibility issue revolves around the way the keystrokes are sent - the example here uses the embedded codes ([Tab], [Enter], [F1]), and this is only supported in the SDSTRM32.EXE, which only operates with My-T-Soft Version 1.50 and above (or NT/2000 versions).

This is grayed out in the My-T-Touch and My-T-Pen versions (Enabled = False) earlier than 1.70. In order to operate properly, the Developer's Kit must have been installed (to ensure the SDSTRM32.EXE file is in the Install directory). By clicking on the Send button, the text in the left-hand text field will be sent through to My-T-Soft - the default [Tab] moves the keyboard focus to the right text field, and the rest of the text is typed. In order for My-T-Soft to type via this mechanism, the cursor must have previously been over the keyboard window, or must be moved over the keyboard window to "engage" the appropriate window & keyboard focus code in My-T-Soft. See Developer notes / source code for SDSTRMTS for other details. Note that using the Set Cursor Position option, this can be forced with a FindWindow / GetWindowRect with the Windows API or MoveWindowMTS and SetCursorPosMTS.

Configuration On-the-fly - Select the appropriate panels and/or size, click on Configure and the keyboard software will reconfigure itself based on the checked panels / size. Note that the position is dependent on current position, and how the new configuration will fit on screen.

See External Control & notes on CPYCNMTS.C for a much stronger level of control which includes positioning.

External Control - For proper operation, the Developer's Kit MUST be extracted (installed).

The files KYBD.CFG, EDIT.CFG, NUM.CFG, and MACRO.CFG must be in the installation directory. If you want to modify the default "configurations", refer to the notes of CPYCNMTS (Copy & Configure My-T-Soft) in the Developer's Kit. Essentially, the configuration & position must be saved - Save Position / Save Settings. Then the KEYBOARD.KBF can be copied to the "saved" configuration (e.g. COPY KEYBOARD.KBF NEW.CFG). Call 'CopyConfigureMTS("NEW.CFG")' to establish this configuration instantly...

Keyboard Config - Uses KYBD.CFG

Edit Config - Uses EDIT.CFG

Num Config - Uses NUM.CFG

Macro Config - Uses MACRO.CFG

Toggle off-screen or reconfigure - this uses the FWCTLMTS approach. Logic is: if "on-screen", move "off-screen" if "off-screen" used CopyConfigure approach to establish new configuration.

Files & Notes

MTSVBDLL.DLL - compiled DLL that contains C based functions as called by Visual Basic

MTSVBDLL.C - MTSVBDLL.DLL Source code

MTSVBDLL.RC - MTSVBDLL.DLL Resource Script

MTSVBDLL.DEF - MTSVBDLL.DLL Exports / Module Definition file

MTSUTIL.ICO - MTSVBDLL.DLL Icon

MYTSOFT.H - Header file MTSVBDLL.DLL source

RESOURCE.H - Visual C Resource Header

MYTSOFT.VBP - Visual Basic Project (v4) for My-T-Soft

MYTSOFT.FRM - Form for My-T-Soft

MYTSOFT.BAS - Module for My-T-Soft

MYTPEN.VBP - Visual Basic Project (v4) for My-T-Pen

MYTPEN.FRM - Form for My-T-Pen

MYTPEN.BAS - Module for My-T-Pen

MYTTOUCH.VBP - Visual Basic Project (v4) for My-T-Touch

MYTTOUCH.FRM - Form for My-T-Touch

MYTTOUCH.BAS - Module for My-T-Touch

My-T-Soft Visual Basic Examples

Version 1.70 - 5/11/2000

Visual C++ Examples (Microsoft Visual C++ Version 6 Project)

FOLDER: VISUALC

TYPE: Example integration in Visual C based program

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

This is an example showing off a few uses of integrating My-T-Soft within a Visual C program. The MTSDLL.C file is included so it can easily be referenced within the project. The key thing to understand with the external control of My-T-Soft is the ability to bring up the appropriate panel when required for your application.

The examples shown are:

When opened, the program checks for the My-T-Soft window if not running, you are given an option of launching the program. In general, it is recommended that My-T-Soft is run either at startup or with the start of the application using the NoSplash option (and perhaps saving a configuration where the keyboard is opened off-screen), the keyboard can be opened & ready for operation as required.

Using a timer, the program puts the keyboard window at the bottom of the window.

Using the default File Open dialog, the file KYBDX.CFG is used to bring up the keyboard panel only by default, with React to Dialogs On (Setup | Configure | Operation Options), the keyboard will move to the bottom of the dialog.

The Edit Find uses a similar approach, but after the dialog is closed, the Edit panel configuration is set.

The Options | My-T-Soft dialog gives you a method for manipulating the displayed panels/size via the dialog.

The Options | Embedded Window shows a sample window that embeds the keyboard into the window.

The example is in C (for familiarity with C & C++ developers), using a basic legacy Windows framework & approach. The source code is provided, along with a built example in the VISUALC folder.

Version 1.77 - July 7, 2003

C# (.NET) Example

C# (.NET) Example

FOLDER: CSHARP

TYPE: Example integration in C# / .NET

SOURCE: INCLUDED

LANGUAGE: C# / .NET Framework

IDE: Visual Studio .NET 2003

C# Examples / DLL of DevKit

Welcome to the C# Example set for Developers!

This is an add-on to the Developer's Kit for IMG's My-T-Soft to allow C# Developer's to access & Use the Developer's Kit directly from C#.

Microsoft C# Version Notes

This was built in Microsoft Visual Studio.NET 2003, .Net Framework version 1.1 for maximum compatibility if you open the project in later versions of Visual Studio.NET, just say OK, etc. to convert to the current version you are running.

NOTE: The C# project cannot be run in older versions of Visual Studio.NET.

Files & Notes

MYTSOFT.csproj - Microsoft Visual Studio.NET C# project file

MYTSOFT.csproj.user - Microsoft Visual Studio.NET Project User Options

Form1.resx - resource file

InputBox.resx - resource file

App.ico - icon file

Form1.cs - main form class which can all the controls and event handling for the controls

InputBox.cs - class which create a dialog that allows the user to enter the path of a file into a textbox and returns the path to the caller

AssemblyInfo.cs - class used to control general information about the assembly

C# Developers:

Running the application under Windows XP, Windows 2000 Professional

Prerequisites: .NET Framework version 1.1 or later.

a. You should find the folder CSHARP in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 1. Navigate to .\CSHARP\MYTSOFT\bin\Release

b. Click on MYTSOFT_CS.exe to run the application

Running the code under Windows XP, Windows 2000 Professional

Prerequisites: Visual Studio.NET 2003 or later.

a. You should find the folder CSHARP in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 1. Navigate to .\CSHARP\MYTSOFT

b. Click on MYTSOFT.csproj to open the project

c. Click on Debug->Start from the menu to run the application.

Operation

The appropriate Project name (MYTSOFT.csproj) should be run from a "CSHARP" folder under the install directory of matching product.

****

DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

****

There is a fair amount of error checking, but every possibility has not been addressed. If the default installation of all components has been done, then the operation should be straightforward & error free. If there are operational problems, then it is assumed that as a C# developer, fixing the error will not be a problem. You may contact IMG at "devkit@imgpresents.com" or check our website "http://www.imgpresents.com" for any news or updates, but technical support is not offered for the Developer's Kit utilities - source code is provided, and should be referenced.

Once in the Visual Studio.NET Environment, simply click Debug then Start from the menu to engage the form & start Operation. Click on buttons to see operational features, "Stop" operation and review C# Form / Properties / Controls & Properties / Class code, etc. See MTSDLL.csproj for source code of C# based DLL.

Note - this DLL has been generalized - see the MTSDLL for the "future" of this DLL.

Operation Notes

Load Event

If the MYTSDLLCS.DLL is not found in the current directory or in the application path where MT??, is installed, a InputBox will appear to "ask" for its location, enter the full path including the filname into the textbox and hit Enter.

Buttons / Text Fields

Launch - This will launch the appropriate product based on the project loaded. By default (See DLL Source code) My-T-Soft is attempted, if it fails, then My-T-Touch is attempted, if it fails, then My-T-Pen is attempted. If these fail, then it is assumed the default install locations have not been used, and the LaunchFromCurrentMTS function is called with the current directory, and the parent folder (i.e. the folder that contains the CSHARP folder) is used as the location of the product. If the My-T-Soft keyboard is NOT launched, either:

a) Modify the source code to fit your requirements

-or-

b) Run the My-T-Soft product directly...

Close - this will Close the running keyboard software

Minimize to Button - will send the message to minimize to button

Open from Button - will send the message to open from the button state

MoveWindow / text fields X & Y - enter appropriate values in the text fields, and click on button to move keyboard window to that (Top/Left) X, Y location

Set Cursor Position / text fields X & Y - this will move the cursor (mouse cursor) to X, Y location

Save Settings - Saves the current configuration / panels open on the keyboard software

Save Position - Saves the Top/Left X, Y position of the keyboard software

Restore Settings - restores the saved panels & position

Restore Position - restores X, Y location (Top/Left) of keyboard software with current configuration preserved

Send Text Through My-T-Soft - This is an advanced feature of the My-T-Soft version - the C# example does not back support My-T-Touch or My-T-Pen, although this could work with some modification - this is left as an exercise for the developer. The compatibility issue revolves around the way the keystrokes are

sent - the example here uses the embedded codes ([Tab], [Enter], [F1]), and this is only supported in the SDSTRM32.EXE, which only operates with My-T-Soft Version 1.50 and above (or NT/2000 versions).

This is grayed out in the My-T-Touch and My-T-Pen versions (Enabled = False) earlier than 1.70. In order to operate properly, the Developer's Kit must have been installed (to ensure the SDSTRM32.EXE file is in the Install directory). By clicking on the Send button, the text in the left-hand text field will be sent through to My-T-Soft - the default [Tab] moves the keyboard focus to the right text field, and the rest of the text is typed. In order for My-T-Soft to type via this mechanism, the cursor must have previously been over the keyboard window, or must be moved over the keyboard window to "engage" the appropriate window & keyboard focus code in My-T-Soft. See Developer notes / source code for SDSTRMTS for other details. Note that using the Set Cursor Position option, this can be forced with a FindWindow / GetWindowRect with the Windows API or MoveWindowMTS and SetCursorPosMTS.

Configuration On-the-fly - Select the appropriate panels and/or size, click on Configure and the keyboard software will reconfigure itself based on the checked panels / size. Note that the position is dependent on current position, and how the new configuration will fit on screen.

See External Control & notes on CPYCNMTS.C for a much stronger level of control which includes positioning.

External Control - For proper operation, the Developer's Kit MUST be extracted (installed). The files KYBD.CFG, EDIT.CFG, NUM.CFG, and MACRO.CFG must be in the installation directory. If you want to modify the default "configurations", refer to the notes of CPYCNMTS (Copy & Configure My-T-Soft) in the Developer's Kit. Essentially, the configuration & position must be saved - Save Position / Save Settings. Then the KEYBOARD.KBF can be copied to the "saved" configuration (e.g. COPY KEYBOARD.KBF NEW.CFG). Call 'CopyConfigureMTS("NEW.CFG")' to establish this configuration instantly...

Keyboard Config - Uses KYBD.CFG

Edit Config - Uses EDIT.CFG

Num Config - Uses NUM.CFG

Macro Config - Uses MACRO.CFG

Toggle off-screen or reconfigure - this uses the FWCTLMTS approach. Logic is: if "on-screen", move "off-screen" if "off-screen" used CopyConfigure approach to establish new configuration.

My-T-Soft C# Examples

Version 1.78 - August 9, 2005

C# (.NET) DLL Example

FOLDER: CSHARP\MYTSDLL

TYPE: Example integration in C# / .NET

SOURCE: INCLUDED

LANGUAGE: C# / .NET Framework

IDE: Visual Studio .NET 2003

MTS DLL with Source in C#

My-T-Soft Developer's Kit DLL - This is a reworked version of the MTSDLL written in C#, for inclusion with C# projects. For more details on the MTSDLL, see the original MTS DLL with Source.

Microsoft C# Version Notes

This was built in Microsoft Visual Studio.NET 2003, .Net Framework version 1.1 for maximum compatibility if you open the project in later versions of Visual Studio.NET, just say OK, etc. to convert to the current version you are running.

NOTE: The C# project cannot be run in older versions of Visual Studio.NET.

C# Developers:

Compiling the Dll code under Windows XP, Windows 2000 Professional

Prerequisites: .NET Framework version 1.1 or later.

a. You should find the folder CSHARP in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 1. Navigate to .\CSHARP\MYTSDLL

b. Click on MYTSDLL.csproj to open the project

c. Click on Build->Build Solution from the menu to create the dll. The dll can be found in the .\CSHARP\MYTSDLL\bin\Release folder.

The DLL utilities comes with the following C# classes:

MYTSDriver.cs - main functionality for MTSDLL.DLL

Util.cs - utility class used for conversions.

Win32API.cs - contains the dllimport declarations needed to perform p/invoke statements between win32 api and the common language runtime

AssemblyInfo.cs - class used to control general information about the assembly Along with Project and solution files for Microsoft Visual Studio.NET:

MYTSDLL.csproj - Microsoft Visual Studio.NET C# project file

MYTSDLL.sln - Microsoft Visual Studio.NET solution object file

MYTSDLL.suo - Microsoft Visual Studio.NET Studio Solution User Options

MYTSDLL.csproj.user - Microsoft Visual Studio.NET Project User Options

MYTSDLLCS.DLL - compiled DLL that contains C# based functions called by the C# Application

Overview

These utilities were developed in response to customer suggestions, wants, needs, and by reviewing the capabilities of various high-end application development tools. The DLL approach allows use of these function calls directly by any environment capable of integrating a DLL. The inclusion of the source code allows easy integration into applications developed closer to the Windows API. The instructional information is also useful in allowing developers to create functional applications quickly.

There are various utilities. Some pair up as opposites, some are more powerful than others. Be sure to read through the following details to gain a good understanding of the ability of these tools to assist you in your development efforts. Please feel free to contact us with other ideas or needs.

See My-T-Soft Developer's Kit for additional info & examples, including how to enable seamless change-overs between configurations.

For usage examples of the C Dll version, please refer to the following: See the MS Access samples integrating the MTSDLL.DLL

Developer's Corner on website "http://www.imgpresents.com/imgdev.htm"

How To's in Developer's Corner

The Visual Basic Example (VBASIC folder in DevKit)

Note that Size 12 is a virtual barrier for re-sizing on the fly from the larger sizes (i.e. greater than size 12) to the smaller sizes (i.e. 12 or less). Do not cross this barrier one way or the other without first establishing a painted keyboard at size 12. The larger sizes are all keyed off of the dimensions of size 12, so corruption of the valid KBF file is possible (probable!) if crossing size 12!

DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

Utility Description / Function Call Interface

This is a list of each function in the DLL. See below for descriptions & interface:

GetXYMTS

GetXYWnd

MoveWindowMTS

SetCursorPosMTS

LaunchMTS

LaunchFromCurrentMTS

CloseMTS

MinimizeMTS

OpenMTS

SaveSettingsMTS

SavePositionMTS

RestoreSettingsMTS

RestorePositionMTS

ConfigureMTS

SendStringMTS

CopyConfigureMTS

ToggleConfigureMTS

SetInputWindowMTS

Utility Name: Get My-T-Soft Position (Top/Left, Width/Height, Bottom/Right)

Function name: GetXYMTS

vFunction Arguments: WhichType as long

//In MYTSDriver.cs

public const long TOPLEFT = 0x01;

public const long LEFTTOP = 0x01;

public const long BOTTOMRIGHT = 0x02;

public const long RIGHTBOTTOM = 0x02;

public const long WIDTHHEIGHT = 0x04;

Returns: long with low word as X, high word as Y

Example: GetXYMTS(TOPLEFT)

Description: This utility uses the GetWindowRect and returns an XY pair composed in a long

return value using the MAKELONG(x,y) macro.

Utility Name: Get Window Position (Top/Left, Width/Height, Bottom/Right)

Function name: GetXYWnd

Function Arguments: hWnd as IntPtr, WhichType as long

//In MYTSDriver.cs

public const long TOPLEFT = 0x01;

public const long LEFTTOP = 0x01;

public const long BOTTOMRIGHT = 0x02;

public const long RIGHTBOTTOM = 0x02;

public const long WIDTHHEIGHT = 0x04;

Returns: LONG with low word as X, high word as Y

Example: GetXYWnd(hWnd,TOPLEFT)

Description: This utility uses the GetWindowRect and returns an XY pair composed in a long return value using the MAKELONG(x,y) macro.

Utility Name: Launch My-T-Soft

Function name: LaunchMTS

Function Arguments: NONE

Returns: Nothing

Description: Executing this will execute the My-T-Soft Executable.

Notes: This assumes a standard install - for non-standard installs, see LaunchFromCurrentMTS.

Utility Name: Launch My-T-Soft from Current Path

Function name: LaunchFromCurrentMTS

Function Arguments: FileName path

Example: LaunchFromCurrentMTS("C:\TEMP\DEVELOP\")

Returns: Nothing

Description: Executing this will execute the My-T-Soft Executable.

Notes: This assumes the My-T-Soft files have been copied into the C:\TEMP\DEVELOP folder, and MYTSOFT.EXE & MYTSOFT.INI are included along with supporting DLL's, KBF's, KMF's, etc., etc. Locks in default INI location for other calls to DLL functions.

Utility Name: Close My-T-Soft

Function name: CloseMTS

Function Arguments: NONE

Returns: Nothing

Description: Executing this will close the My-T-Soft Executable.

Notes: Requested by an MS-Access Developer. Sends a WM_CLOSE message to the My-T-Soft Window.

Utility Name: Minimize My-T-Soft

Function name: MinimizeMTS

Function Arguments: NONE

Returns: Nothing

Description: Minimizes My-T-Soft using configured method.

Notes: Use OpenMTS to return My-T-Soft to original state.

Name: Open My-T-Soft

Function name: OpenMTS

Function Arguments: NONE

Returns: bool

Description: Opens My-T-Soft from minimized state using My-T-Soft API.

Notes: Opposite of MINMZMTS.EXE

Name: Move My-T-Soft Window

Function name: MoveWindowMTS

Function Arguments: X as int, Y as int

Returns: Nothing

Example: MoveWindow(200,20)

Description: Moves My-T-Soft window to indicated x & y position.

Notes: Extremely useful for complex applications. You may wish to always position My-T-Soft so the Tool bar panel (logo / menu / minimize) are mostly or all the way off the screen.

Name: SetCursor for My-T-Soft

Function name: SetCursorPosMTS

Function Arguments: X as int, Y as int

Example: SetCursorPos(200,20)

Returns: Nothing

Description: Moves cursor to indicated x & y position.

Notes: Focus issues and shell or exec implementation of My-T-Soft Utilities can sometimes result in the wrong window having the focus. By moving the cursor over My-T-Soft, code in My-T-Soft can resolve some focus issues.

Name: Configure My-T-Soft

Function name: ConfigureMTS

Function Arguments: /K[-] /E[-] /N[-] /C[-] /M[-] /W[-] /T[-] /B[-] /I[-] /Q[-] /G[-] /S:??

Returns: Nothing

Example: ConfigureMTS("/k-/e-/w/m/s:9")

Closes Keyboard & Edit panel, opens Windows Control & Macro Panel, sizes to 9.

K = Keyboard (alphanumeric) panel

E = Edit panel

N = Numeric panel

C = Calculator panel

M = Macro panel

W = Windows control panel

T = Tool / Control panel

B = Toolbar (3 icon panel)

G = Magnifier panel

I = System Info panel

Q = QuickHelp panel

- = after any of the above indicates the panel will be closed

S:?? = Size from 1 - 12

Description: Gives complete control over appearance. Used in conjunction with

MoveWindowMTS allows different looks / interfaces for complex applications.

Notes: This modifies the INI file, and then sends the update command to My-T-Soft. The UpdateDefINI routine is used to find installation folder and location of INI file.

Name: Copy and Configure My-T-Soft

Function name: CopyConfigureMTS

Function Arguments: [drive:][path]FileName

Example: CopyConfigureMTS("NUM.CFG"), CopyConfigureMTS("D:\PANELS\KEYPAD.CFG")

Returns: bool

Description: Gives complete control over appearance - cleaner switch over than ConfigureMTS.

Notes:

If no path is given within the FileName specification, the default My-T-Soft directory is used - however, for guaranteed operation, a well formed path (with short file names) is required.

The KEYBOARD.KBF file that is overwritten defaults to the default My-T-Soft directory.

Name: FindWindow & Control My-T-Soft

Function name: ToggleConfigureMTS

Function Arguments: [drive:][path]FileName

Example: ToggleConfigure("NUM.CFG,") ToggleConfigure("D:\PANELS\KEYPAD.CFG")

Returns: Nothing

Description: Gives complete control over appearance with the added ability to move the displayed panels off-screen if already on-screen. This is a modified version of CopyConfigureMTS that has the following logic: If My-T-Soft is visible, move it off-screen and do not process file. If My-T-Soft is not visible (off-screen), then process file.

Notes: This allows a toggle selection for the user using the same action (button on application, CTRLMTS buttons, etc.). Bring up the keyboard, then put it away. Note if My-T-Soft is off-screen, then this acts exactly as the CopyConfigureMTS utility. If My-T-Soft is visible, then its added feature of moving the software off-screen occurs. 1 monitor assumed (for multiple monitor configurations, modification of the source is probably required).

Name: Send String Through My-T-Soft

Function name: SendStringMTS

Function Arguments: Text String (with Build-A-Macro cooked format)

Returns: Nothing

Description: This communicates with My-T-Soft and hands off a text string to be processed and typed.

Notes: Version does not read the raw formats. Use the Save option in Build-A-Macro, Keystroke Macro, Zoom as the string format for this command. There is a file created (SDSTRMTS.KMF), and then the message is sent to My-T-Soft. The location is keyed off of the INI file, and must reside in the same folder as the executable.

Name: Save Position of My-T-Soft

Function name: SavePositionMTS

Function Arguments: NONE

Returns: Nothing

Description: Saves current position, for use with RestorePositionMTS, or upon re-opening of My-T-Soft.

Notes: Opposite of RestorePositionMTS. Same as Menu Option, Save Current Position.

Name: Save Settings of My-T-Soft

Function name: SaveSettingsMTS

Function Arguments: NONE

Returns: Nothing

Description: Saves current settings, for use with RestoreSettingsMTS, or upon re-opening of My-T-Soft.

Notes: Opposite of RestoreSettingsMTS. Same as Menu Option, Save Current settings.

Name: Restore Position of My-T-Soft

Function name: RestorePositionMTS

Function Arguments: NONE

Returns: Nothing

Description: Restore saved position.

Notes: Opposite of SavePositionMTS. Same as Menu Option, Restore Current Position.

Name: Restore Settings of My-T-Soft

Function name: RestoreSettingsMTS

Function Arguments: NONE

Returns: Nothing

Description: Restores saved settings.

Notes: Opposite of SaveSettingsMTS. Same as Menu Option, Restore Current settings.

Name: Interface to SetForegroundWindow API via MTSDLL

Function name: SetInputWindowMTS

Function Arguments: IntPtr to Window

Returns: bool result of SetForegroundWindow call

Description: Used to set the input focus (keyboard focus) to the specified window

Notes: There are various limitations in the various versions of Windows with this API and the ability for programs to control the input focus. This is just a basic interface in the DLL so an environment that doesnt have access to the Windows API can access SetForegroundWindow.

Refer to MSDN (Microsoft Developer Network) resources for limitations on this API call.

Version 1.78 - August 9, 2005

Java Example

FOLDER: JAVA

TYPE: Example integration in Java

SOURCE: INCLUDED

LANGUAGE: Java

IDE: Eclipse 3.00

Welcome to the Java Example set for Developers!

Java Examples / DLL of DevKit. This is an add-on to the Developer's Kit for IMG's My-T-Soft to allow Java Developer's to access & Use the Developer's Kit directly from Java.

Java Version Notes

This was built in Eclipse 3.0, java version 1.4.2_08.

NOTE: jre version 1.4 or later is needed to run code from Eclipse 3.0.

Files & Notes

JNIMTSDLL.dll - Dll that interfaces with My-T-Soft

MTSDIR.dll - finds the My-T-Soft installation path

MYTSOFT.java - the main java application.

swt-win32-3064.dll - dll used for swt gui applications.

Running the application under Windows XP, Windows 2000 Professional

Prerequisites: J2RE version 1.4.2_08 or later.

J2SDK version 1.4.2_08 or later (needed to create C++ header files for jni).

Update the environment variable Path to include the bin folder of both the J2RE and J2SDK.

Eclipse 3.0

a. You should find the folder JAVA in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 3. Navigate to .\JAVA\MYTSOFT\exec

b. Click on MYTSOFT.jar to run the application

Running the code under Windows XP, Windows 2000 Professional

Prerequisites: J2RE version 1.4.2_08 or later.

J2SDK version 1.4.2_08 or later (needed to create C++ header files for jni).

Eclipse 3.0

Note: Update the environment variable Path to include the bin folder of both the J2RE and J2SDK.

a. You should find the folder JAVA in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 3. Navigate to .\JAVA

b. Open Eclipse.

c. Copy the MYTSOFT folder from .\JAVA to the Eclipse workspace

i.e. C:\eclipse\workspace

d. In Eclipse, click on Window->Open Perspective->Java

e. In the Package Panel on the left side, right click in the white space and click on Import... from the menu. In the Import Dialog box click on Existing Project into Workspace and click Next

f. Click Browse and choose the MYTSOFT folder that was copied to the eclipse workspace directory

g. Click OK

h. Click Finish

i. In the Package Panel expand (default package) to open the Java file.

j. To Run the application, click Run->Run As->Java Application.

Compiling the MTSDll Dll code under Windows XP, Windows 2000 Professional

Prerequisites: J2RE version 1.4.2_08 or later.

J2SDK version 1.4.2_08 or later (needed to create C++ header files for jni).

Microsoft Visual C++ 6.0

a. You should find the folder JAVA in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 3.

b. Open Microsoft Visual C++ 6.0

c. Click File->Open Workspace and navigate to the location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen) then to folder Java\MTSDLL and click on the file MTSDLL.dsw

d. To compile the code click Build->Build JNIMTSDLL.dll.

The dll can be found in the .\JAVA\MTSDLL\Release folder.

Note: jni_md.h, jni.h must be copied from the j2sdk 1.4 (or later) folder and pasted in the Visual Studio Include folder i.e "%installed path%\Microsoft Visual Studio\VC98\Include".

Compiling the MTSDIR Dll code under Windows XP, Windows 2000 Professional

Prerequisites: J2RE version 1.4.2_08 or later.

J2SDK version 1.4.2_08 or later (needed to create C++ header files for jni).

Microsoft Visual C++ 6.0

a. You should find the folder JAVA in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 3.

b. Open Microsoft Visual C++ 6.0

c. Click File->Open Workspace and navigate to the location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen) then to folder Java\MTSDIR and click on the file MTSDIR.dsw

d. To compile the code click Build->Build MTSDIR.dll.

The dll can be found in the .\JAVA\MTSDIR\Debug folder.

Some of the features assume that the Developer's Kit has already been "installed". If the "Send Text Thru" or the "External Control" examples are not operating properly, then install the DevKit:

Start Menu | Programs | My-T-???? | Install Developer's Kit

Notes on the two dll's used in this project

NOTE:

MTSDIR Folder contains the dll that is used obtains the directory where the My-T-Soft product was installed.

MTSDLL Folder contains the dll utilities themselves

NOTE: jni_md.h, jni.h must be copied from the j2sdk 1.4 (or later) folder and pasted in the Visual Studio Include folder i.e "%installed path%\Microsoft Visual Studio\VC98\Include".

****

DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

****

There is a fair amount of error checking, but every possibility has not been addressed. If the default installation of all components has been done, then the operation should be straightforward & error free. If there are operational problems, then it is assumed that as a JAVA developer, fixing the error will not be a problem. You may contact IMG at "devkit@imgpresents.com" or check our website "http://www.imgpresents.com" for any news or updates, but technical support is not offered for the Developer's Kit utilities - source code is provided, and should be referenced.

Once workspace is setup in Elcipse, simply click Run, Run As then click Java Application from the menu to engage the form & start Operation. Click on buttons to see operational features, "Stop" operation and review SWT Properties / Controls & Class code, etc. See MTDLL.dsw for the source code of C based DLL.

Note - the MTSDLLJNI DLL has been modified to implement the Java Native Interface (JNI) to allow interoperablity between Java and C.

Operation Notes

Load Event

If the MTSDLL.DLL is not found in the current directory or in the application path where MT??, is installed, a InputBox will appear to "ask" for its location, enter the full path including the filname into the textbox and hit Enter.

Buttons / Text Fields

Launch - This will launch the appropriate product based on the project loaded. By default (See DLL Source code) My-T-Soft is attempted, if it fails, then My-T-Touch is attempted, if it fails, then My-T-Pen is attempted. If these fail, then it is assumed the default install locations have not been used, and the LaunchFromCurrentMTS function is called with the current directory, and the parent folder (i.e. the folder that contains the CSHARP folder) is used as the location of the product. If the My-T-Soft keyboard is NOT launched, either:

a) Modify the source code to fit your requirements

-or-

b) Run the My-T-Soft product directly...

Close - this will Close the running keyboard software

Minimize to Button - will send the message to minimize to button

Open from Button - will send the message to open from the button state

MoveWindow / text fields X & Y - enter appropriate values in the text fields, and click on button to move keyboard window to that (Top/Left) X, Y location

Set Cursor Position / text fields X & Y - this will move the cursor (mouse cursor) to X, Y location

Save Settings - Saves the current configuration / panels open on the keyboard software

Save Position - Saves the Top/Left X, Y position of the keyboard software

Restore Settings - restores the saved panels & position

Restore Position - restores X, Y location (Top/Left) of keyboard software with current configuration preserved

Send Text Through My-T-Soft - This is an advanced feature of the My-T-Soft version - the Java example does not back support My-T-Touch or My-T-Pen, although this could work with some modification - this is left as an exercise for the developer. The compatibility issue revolves around the way the keystrokes are sent - the example here uses the embedded codes ([Tab], [Enter], [F1]), and this is only supported in the SDSTRM32.EXE, which only operates with My-T-Soft Version 1.50 and above (or NT/2000 versions).

This is grayed out in the My-T-Touch and My-T-Pen versions (Enabled = False) earlier than 1.70. In order to operate properly, the Developer's Kit must have been installed (to ensure the SDSTRM32.EXE file is in the Install directory). By clicking on the Send button, the text in the left-hand text field will be sent through to My-T-Soft - the default [Tab] moves the keyboard focus to the right text field, and the rest of the text is typed. In order for My-T-Soft to type via this mechanism, the cursor must have previously been over the keyboard window, or must be moved over the keyboard window to "engage" the appropriate window & keyboard focus code in My-T-Soft. See Developer notes / source code for SDSTRMTS for other details. Note that using the Set Cursor Position option, this can be forced with a FindWindow / GetWindowRect with the Windows API or MoveWindowMTS and SetCursorPosMTS.

Configuration On-the-fly - Select the appropriate panels and/or size, click on Configure and the keyboard software will reconfigure itself based on the checked panels / size. Note that the position is dependent on current position, and how the new configuration will fit on screen.

See External Control & notes on CPYCNMTS.C for a much stronger level of control which includes positioning.

External Control - For proper operation, the Developer's Kit MUST be extracted (installed). The files KYBD.CFG, EDIT.CFG, NUM.CFG, and MACRO.CFG must be in the installation directory. If you want to modify the default "configurations", refer to the notes of CPYCNMTS (Copy & Configure My-T-Soft) in the Developer's Kit. Essentially, the configuration & position must be saved - Save Position / Save Settings. Then the KEYBOARD.KBF can be copied to the "saved" configuration (e.g. COPY KEYBOARD.KBF NEW.CFG). Call 'CopyConfigureMTS("NEW.CFG")' to establish this configuration instantly...

Keyboard Config - Uses KYBD.CFG

Edit Config - Uses EDIT.CFG

Num Config - Uses NUM.CFG

Macro Config - Uses MACRO.CFG

Toggle off-screen or reconfigure - this uses the FWCTLMTS approach. Logic is: if "on-screen", move "off-screen" if "off-screen" used CopyConfigure approach to establish new configuration.

My-T-Soft Java Examples

Version 1.78 - August 9, 2005

J# (J Sharp) Example

FOLDER: JSHARP

TYPE: Example integration in J# (J Sharp)

SOURCE: INCLUDED

LANGUAGE: J#

IDE: Visual Studio .NET 2003

Welcome to the J# Example set for Developers!

J# Examples / DLL of DevKit - This is an add-on to the Developer's Kit for IMG's My-T-Soft to allow J# Developer's to access & Use the Developer's Kit directly from J#.

Microsoft J# Version Notes

This was built in Microsoft Visual Studio.NET 2003, .Net Framework version 1.1 for maximum compatibility if you open the project in later versions of Visual Studio.NET, just say OK, etc. to convert to the current version you are running.

NOTE: The J# project cannot be run in older version of Visual Studio.NET.

Files & Notes

MYTSOFTJSHARP.vjsproj - Microsoft Visual Studio.NET J# project file

MYTSOFTJSHARP.vjsproj.user - Microsoft Visual Studio.NET Project User Options

Form1.resx - resource file

Form1.jsl - main form class which can all the controls and event handling for the controls

InputBox.jsl - class which create a dialog that allows the user to enter the path of a file into a textbox and returns the path to the caller

AssemblyInfo.jsl - class used to control general information about the assembly

Running the application under Windows XP, Windows 2000 Professional

Prerequisites: .NET Framework version 1.1 or later with the Visual J# .NET Redistributable Package 1.1

a. You should find the folder JSHARP in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 2. Navigate to .\JSHARP\MYTSOFTJSHARP\bin\Release

b. Click on MYTSOFTJSHARP.exe to run the application

Running the code under Windows XP, Windows 2000 Professional

Prerequisites: Visual Studio.NET 2003 or later with the Visual J# .NET Redistributable Package 1.1

a. You should find the folder JSHARP in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 2. Navigate to .\JSHARP\MYTSOFTJSHARP

b. Click on MYTSOFTJSHARP.vjsproj to open the project

c. Click on Debug->Start from the menu to run the application.

Compiling the Dll code under Windows XP, Windows 2000 Professional

a. You should find the folder JSHARP in default install location of the My-T-Soft product (My-T-Soft, My-T-Touch, My-T-Pen). If folder is not found then follow the installations instructions under Sec. 2. Navigate to .\JSHARP\MYTSDLL

b. Click on MYTSDLL.csproj to open the project

c. Click on Build->Build Solution from the menu to create the dll. The dll can be found in the .\JSHARP\MYTSDLL\bin\Release folder.

Some of the features assume that the Developer's Kit has already been "installed". If the "Send Text Thru" or the "External Control" examples are not operating properly, then install the DevKit:

Start Menu | Programs | My-T-???? | Install Developer's Kit

Operation

The appropriate Project name (MYTSOFTJSHARP.vjsproj) should be run from a "JSHARP" folder under the install directory of matching product.

****

DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

****

There is a fair amount of error checking, but every possibility has not been addressed. If the default installation of all components has been done, then the operation should be straightforward & error free. If there are operational problems, then it is assumed that as a J# developer, fixing the error will not be a problem. You may contact IMG at "devkit@imgpresents.com" or check our website "http://www.imgpresents.com" for any news or updates, but technical support is not offered for the Developer's Kit utilities - source code is provided, and should be referenced.

Once in the Visual Studio.NET Environment, simply click Debug then Start from the menu to engage the form & start Operation. Click on buttons to see operational features, "Stop" operation and review J# Form / Properties / Controls & Properties / Class code, etc. See MTSDLL.csproj for source code of C# based DLL.

Note - this DLL has been generalized - see the MTSDLL for the "future" of this DLL.

Operation Notes

Load Event

If the MYTSDLLCS.DLL is not found in the current directory or in the application path where MY-T??, is installed, a InputBox will appear to "ask" for its location, enter the full path including the filname into the textbox and hit Enter.

Buttons / Text Fields

Launch - This will launch the appropriate product based on the project loaded. By default (See DLL Source code) My-T-Soft is attempted, if it fails, then My-T-Touch is attempted, if it fails, then My-T-Pen is attempted. If these fail, then it is assumed the default install locations have not been used, and the LaunchFromCurrentMTS function is called with the current directory, and the parent folder (i.e. the

folder that contains the JSHARP folder) is used as the location of the product. If the My-T-Soft keyboard is NOT launched, either:

a) Modify the source code to fit your requirements

-or-

b) Run the My-T-Soft product directly...

Close - this will Close the running keyboard software

Minimize to Button - will send the message to minimize to button

Open from Button - will send the message to open from the button state

MoveWindow / text fields X & Y - enter appropriate values in the text fields, and click on button to move keyboard window to that (Top/Left) X, Y location

Set Cursor Position / text fields X & Y - this will move the cursor (mouse cursor) to X, Y location

Save Settings - Saves the current configuration / panels open on the keyboard software

Save Position - Saves the Top/Left X, Y position of the keyboard software

Restore Settings - restores the saved panels & position

Restore Position - restores X, Y location (Top/Left) of keyboard software with current configuration preserved

Send Text Through My-T-Soft - This is an advanced feature of the My-T-Soft version - the J# example does not back support My-T-Touch or My-T-Pen, although this could work with some modification - this is left as an exercise for the developer. The compatibility issue revolves around the way the keystrokes are sent - the example here uses the embedded codes ([Tab], [Enter], [F1]), and this is only supported in the SDSTRM32.EXE, which only operates with My-T-Soft Version 1.50 and above (or NT/2000 versions).

This is grayed out in the My-T-Touch and My-T-Pen versions (Enabled = False) earlier than 1.70. In order to operate properly, the Developer's Kit must have been installed (to ensure the SDSTRM32.EXE file is in the Install directory). By clicking on the Send button, the text in the left-hand text field will be sent through to My-T-Soft - the default [Tab] moves the keyboard focus to the right text field, and the rest of the text is typed. In order for My-T-Soft to type via this mechanism, the cursor must have previously been over the keyboard window, or must be moved over the keyboard window to "engage" the appropriate window & keyboard focus code in My-T-Soft. See Developer notes / source code for SDSTRMTS for other details. Note that using the Set Cursor Position option, this can be forced with a FindWindow / GetWindowRect with the Windows API or MoveWindowMTS and SetCursorPosMTS.

Configuration On-the-fly - Select the appropriate panels and/or size, click on Configure and the keyboard software will reconfigure itself based on the checked panels / size. Note that the position is dependent on current position, and how the new configuration will fit on screen.

See External Control & notes on CPYCNMTS.C for a much stronger level of control which includes positioning.

External Control - For proper operation, the Developer's Kit MUST be extracted (installed).

The files KYBD.CFG, EDIT.CFG, NUM.CFG, and MACRO.CFG must be in the installation directory. If you want to modify the default "configurations", refer to the notes of CPYCNMTS (Copy & Configure My-T-Soft) in the Developer's Kit. Essentially, the configuration & position must be saved - Save Position / Save Settings. Then the KEYBOARD.KBF can be copied to the "saved" configuration (e.g.

COPY KEYBOARD.KBF NEW.CFG). Call 'CopyConfigureMTS("NEW.CFG")' to establish this configuration instantly...

Keyboard Config - Uses KYBD.CFG

Edit Config - Uses EDIT.CFG

Num Config - Uses NUM.CFG

Macro Config - Uses MACRO.CFG

Toggle off-screen or reconfigure - this uses the FWCTLMTS approach. Logic is: if "on-screen", move "off-screen" if "off-screen" used CopyConfigure approach to establish new configuration.

My-T-Soft J# Examples

Version 1.78 - August 9, 2005

MS Access samples integrating the MTSDLL.DLL

FOLDER: MSACCESS

TYPE: Example integration in Microsoft Access

SOURCE: INCLUDED

LANGUAGE: Visual Basic / MS Access

IDE: Microsoft Access

This example was originally in the MTDLL example in previous releases of the Developers Kit. It has been updated and improved to show off more functionality, and it has been updated to handle all versions of Windows, along with My-T-Touch/My-T-Soft/My-T-Pen without the need for editing.

Located in the MSACCESS Folder, select the installed product, then select the WINDOWS or WINNT Windows folder (WINNT for NT/2000 systems). The sample will walk you through some examples.

Here are some notes on what is included:

The Module illustrates the correct VB syntax for making the DLL functions visible to the MS Access VBA (Visual Basic for Applications) code.

The Forms show off several approaches on how people integrate My-T-Soft - using a minimize/open approach, using a Move Window (on/off screen) approach, or using the Copy & Configure approach. A configuration form is also shown illustrating how to call out & change the displayed panels and current size as necessary.

Version 1.77 - July 7, 2003

Internet Explorer utilities

FOLDER: IEXPLORE / IEXPLORE/WebSync

TYPE: Web Pages, utility / stand-alone executable with source

SOURCE: INCLUDED

LANGUAGE: C

IDE: Microsoft Visual C++ 6

Note: Option 2 will work with My-T-Soft "Out of the box" and the Developer's Kit (DEVKIT)

Name: WebSync

Program name: WEBSYNC.EXE

Program Arguments: NONE

Requires: 1.70 version

Description: WebSync monitors for the MTSCMD.TXT file based on its delay - if it finds the file, it will process it and "execute" the command. Review OVERVIEW.HTM in a browser for VBScript, and further details.

The IEXPLORE Folder contains:

WebSync - WEBSYNC.EXE

IEXPLORE.HTM

EXAMPLES.HTM

These HTML files are used as a starting point for the appropriate examples. The 1.77 version updates these with folders for My-T-Pen/My-T-Soft/My-T-Touch so modifications are not required to show off the examples.

Integrating with Internet Explorer

My-T-Soft used as an example / Integrated with Internet Explorer

Option 1

You can trigger events within My-T-Soft by using WebSync and VBScript (or other scripting method that can create a file on the client system).

(1) IEXPLORE contains folders for each product with 2 sets of 2 HTM files for the appropriate version of windows (EDTPAGEW.HTM, OVRVIEWW.HTM for use with versions of Windows that use a WINDOWS folder, and EDTPAGEN.HTM, OVRVIEWN.HTM for use with versions of Windows that use a WINNT folder)

(2) Run WEBSYNC.EXE to run WebSync

(3) Make sure you have My-T-Soft running, it is not minimized, and you have a keyboard open at about size 8

(4) Select the appropriate folder & file for your version of Windows and installed product. (EDTPAGE? will be referred to as EDITPAGE and OVRVIEW? Will be referred to as OVERVIEW) Double-click on EDITPAGE.HTM - this should open Internet Explorer - if you are on Windows 98, My-T-Soft, you should see the keyboard position itself on EDITPAGE, and hide on OVERVIEW - the 2 pages are linked to illustrate this approach of using the Devevloper's Kit & VBScript with WebSync

(5) All the details about the VBScript approach and WebSync are in the OVERVIEW.HTM - refer to this for notes & implementation details

(6) Review OVERVIEW.HTM and the Web page "source" to see details.

(7) Quick Note: This example uses VBScript to communicate externally to WebSync, which uses the pre-compiled Developer's Kit executables to manipulate My-T-Soft.

Option 2

You can trigger events within My-T-Soft by using the My-T-Soft Setup | Configuration | Special Options' Run command. This works best if you only have a small number of pages where you need a keyboard configuration, and/or you aren't using an advanced browser. The following example assumes you have 1 page with a form on it (where the keyboard is required), and a second page indicating proper submission.

(1) You need the window names (the "TITLE" tag for the HTML page) - assuming you are in control of these, you should create a small, but unique title to each page you want to trigger on.

(2) Run My-T-Soft Setup | Configuration | Special Options.

(3) Enter the text for the "form" window in the drop-down combo edit box (Application Window Name), e.g. "Please fill out form"

(4) Click on Run EXE radio button, Browse to select Developer's Kit EXE (e.g. C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE)

(5) Add command line parameters (C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE x:20 y:344)

(6) Click OK

(7) Repeat for second window, click on Special Options

(8) Enter the text for the "submitted" window in the drop-down combo edit box (Application Window Name), e.g. "Thank you!"

(9) Click on Run EXE radio button, Browse to select Developer's Kit EXE (e.g. C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE)

(10) Add command line parameters (C:\WINDOWS\MYTSOFT\DEVKIT\MOVEWMTS.EXE x:2000 y:344)

(11) Click OK

(12) Test with web pages

(13) Notes: You need at least 2 windows to make operation work as you'd expect. To prevent an infinite loop, the Run EXE runs, and then is locked out until a different Run EXE event is triggered.

(14) The "React to Dialogs..." in My-T-Soft Setup | Configuration | Operation Options must be checked on to allow monitoring active windows.

(15) Only 64 characters from the Window Text (Window Name) are checked, and there is a max of 32 characters allowed in Special Options

(16) The name entered is matched against the whole name, so "Word" will match WordPad, Microsoft Word, WordPerfect, etc.

(17) This will drag on system resources, and may cause problems if you get a large number of entries in Special Options - this will depend on the speed of the system, memory, other applications running, etc.,

etc. so the exact number your particular system can handle before you notice speed issues will vary. This is really meant for just a handful of specific windows, not a 100 different configurations.

Version 1.77 July 7, 2003

MTS DLL with Source

FOLDER: MTSDLL

TYPE: DLL with Source

SOURCE: INCLUDED

LANGUAGE: C

IDE: Microsoft Visual C++ 6

My-T-Soft Developer's Kit DLL

Utilities for implementing My-T-Soft within application development environments (Microsoft Access, Borland Delphi, Visual Basic, and other visual application design environments), implemented as a stand-alone Dynamic Link Library with source code.

My-T-Soft is used as a generic term for My-T-Touch, My-T-Pen, My-T-Soft, or private label versions of IMG's My-T-Soft family.

Overview

The DLL utilities comes with a C source module, a Resource Script (RC), and a Module Definition file (DEF), along with Project & Workspace information for Microsoft Visual C++ Version 6.

These utilities were developed in response to customer suggestions, wants, needs, and by reviewing the capabilities of various high-end application development tools. The DLL approach allows use of these function calls directly by any environment capable of integrating a DLL. The inclusion of the source code allows easy integration into applications developed closer to the Windows API. The instructional information is also useful in allowing developers to create functional applications quickly.

There are various utilities. Some pair up as opposites, some are more powerful than others. Be sure to read through the following details to gain a good understanding of the ability of these tools to assist you in your development efforts. Please feel free to contact us with other ideas or needs.

The 1.71 release modified the DLL included with the VBASIC examples. All future add-ons to the DLL portion of the Developer's Kit will be done in this project. Most of these are alternate implementations of the base Developer's Kit - see I. My-T-Soft Developer's Kit for additional info & examples, including how to enable seamless change-overs between configurations.

For usage examples, please refer to the following:

See the MS Access samples integrating the MTSDLL.DLL

Developer's Corner on website "http://www.imgpresents.com/imgdev.htm"

How To's in Developer's Corner

The Visual Basic Example (VBASIC folder in DevKit)

Note that Size 12 is a virtual barrier for re-sizing on the fly from the larger sizes (i.e. greather than size 12) to the smaller sizes (i.e. 12 or less). Do not cross this barrier one way or the other without first establishing a painted keyboard at size 12. The larger sizes are all keyed off of the dimensions of size 12, so corruption of the valid KBF file is possible (probable!) if crossing size 12!

DISCLAIMER

The source code & derived executable are provided at no cost as useful & instructional materials to developers & system integrators incorporating IMG software. Because no licensing fee has been tendered, there is no technical support offered for this software, source code, or its capabilities to solve a particular problem.

Utility Description / Function Call Interface

This is a list of each function in the DLL. See below for descriptions & interface:

GetXYMTS

GetXYWnd

MoveWindowMTS

SetCursorPosMTS

LaunchMTS

LaunchFromCurrentMTS

CloseMTS

MinimizeMTS

OpenMTS

SaveSettingsMTS

SavePositionMTS

RestoreSettingsMTS

RestorePositionMTS

ConfigureMTS

SendStringMTS

CopyConfigureMTS

ToggleConfigureMTS

QueryPanelsMTS

QuerySizeMTS

SetInputWindowMTS

Utility Name: Get My-T-Soft Position (Top/Left, Width/Height, Bottom/Right)

Function name: GetXYMTS

Function Arguments: WhichType as long

//In MYTSOFT.H

#define TOPLEFT 0x01

#define LEFTTOP 0x01

#define BOTTOMRIGHT 0x02

#define RIGHTBOTTOM 0x02

#define WIDTHHEIGHT 0x04

Returns: LONG with low word as X, high word as Y

Example: GetXYMTS(TOPLEFT)

Description: This utility uses the GetWindowRect and returns an XY pair composed in a long return value using the MAKELONG(x,y) macro.

Utility Name: Get Window Position (Top/Left, Width/Height, Bottom/Right)

Function name: GetXYWnd

Function Arguments: hWnd as Long, WhichType as long

//In MYTSOFT.H

#define TOPLEFT 0x01

#define LEFTTOP 0x01

#define BOTTOMRIGHT 0x02

#define RIGHTBOTTOM 0x02

#define WIDTHHEIGHT 0x04

Returns: LONG with low word as X, high word as Y

Example: GetXYWnd(hWnd,TOPLEFT)

Description: This utility uses the GetWindowRect and returns an XY pair composed in a long return value using the MAKELONG(x,y) macro.

Utility Name: Launch My-T-Soft

Function name: LaunchMTS

Function Arguments: NONE

Returns: Nothing

Description: Executing this will execute the My-T-Soft Executable.

Notes: This assumes a standard install - for non-standard installs, see LaunchFromCurrentMTS.

Utility Name: Launch My-T-Soft from Current Path

Function name: LaunchFromCurrentMTS

Function Arguments: FileName path

Example: LaunchFromCurrentMTS("C:\TEMP\DEVELOP\")

Returns: Nothing

Description: Executing this will execute the My-T-Soft Executable.

Notes: This assumes the My-T-Soft files have been copied into the C:\TEMP\DEVELOP folder, and MYTSOFT.EXE & MYTSOFT.INI are included along with supporting DLL's, KBF's, KMF's, etc., etc.

Locks in default INI location for other calls to DLL functions.

Utility Name: Close My-T-Soft

Function name: CloseMTS

Function Arguments: NONE

Returns: Nothing

Description: Executing this will close the My-T-Soft Executable.

Notes: Requested by an MS-Access Developer. Sends a WM_CLOSE message to the My-T-Soft Window.

Utility Name: Minimize My-T-Soft

Function name: MinimizeMTS

Function Arguments: NONE

Returns: Nothing

Description: Minimizes My-T-Soft using configured method.

Notes: Use OpenMTS to return My-T-Soft to original state.

Name: Open My-T-Soft

Function name: OpenMTS

Function Arguments: NONE

Returns: Nothing

Description: Opens My-T-Soft from minimized state using My-T-Soft API.

Notes: Opposite of MINMZMTS.EXE

Name: Move My-T-Soft Window

Function name: MoveWindowMTS

Function Arguments: X as long, Y as long

Returns: Nothing

Example: MoveWindow(200,20)

Description: Moves My-T-Soft window to indicated x & y position.

Notes: Extremely useful for complex applications. You may wish to always position My-T-Soft so the Tool bar panel (logo / menu / minimize) are mostly or all the way off the screen.

Name: SetCursor for My-T-Soft

Function name: SetCursorPosMTS

Function Arguments: X as long, Y as long

Example: SetCursorPos(200,20)

Returns: Nothing

Description: Moves cursor to indicated x & y position.

Notes: Focus issues and shell or exec implementation of My-T-Soft Utilities can sometimes result in the wrong window having the focus. By moving the cursor over My-T-Soft, code in My-T-Soft can resolve some focus issues.

Name: Configure My-T-Soft

Function name: ConfigureMTS

Function Arguments: /K[-] /E[-] /N[-] /C[-] /M[-] /W[-] /T[-] /B[-] /I[-] /Q[-] /G[-] /S:??

Returns: Nothing

Example: ConfigureMTS("/k-/e-/w/m/s:10")

Closes Keyboard & Edit panel, opens Windows Control & Macro Panel, sizes to 10.

K = Keyboard (alphanumeric) panel

E = Edit panel

N = Numeric panel

C = Calculator panel

M = Macro panel

W = Windows control panel

T = Tool / Control panel

B = Toolbar (3 icon panel)

G = Magnifier panel

I = System Info panel

Q = QuickHelp panel

- = after any of the above indicates the panel will be closed

S:?? = Size from 1 - 12

Description: Gives complete control over appearance. Used in conjunction with

MoveWindowMTS allows different looks / interfaces for complex applications.

Notes: This modifies the INI file, and then sends the update command to My-T-Soft. The UpdateDefINI routine is used to find installation folder and location of INI file.

Name: Copy and Configure My-T-Soft

Function name: CopyConfigureMTS

Function Arguments: [drive:][path]FileName

Example: CopyConfigureMTS("NUM.CFG"), CopyConfigureMTS("D:\PANELS\KEYPAD.CFG")

Description: Gives complete control over appearance - cleaner switch over than ConfigureMTS.

Notes:

If no path is given within the FileName specification, the default My-T-Soft directory is used - however, for guaranteed operation, a well formed path (with short file names) is required. The KEYBOARD.KBF file that is overwritten defaults to the default My-T-Soft directory.

Name: FindWindow & Control My-T-Soft

Function name: ToggleConfigureMTS

Function Arguments: [drive:][path]FileName

Example: ToggleConfigure("NUM.CFG,") ToggleConfigure("D:\PANELS\KEYPAD.CFG")

Returns: Nothing

Description: Gives complete control over appearance with the added ability to move the displayed panels off-screen if already on-screen. This is a modified version of CopyConfigureMTS that has the following logic: If My-T-Soft is visible, move it off-screen and do not process file. If My-T-Soft is not visible (off-screen), then process file.

Notes: This allows a toggle selection for the user using the same action (button on application, CTRLMTS buttons, etc.). Bring up the keyboard, then put it away. Note if My-T-Soft is off-screen, then this acts exactly as the CopyConfigureMTS utility. If My-T-Soft is visible, then its added feature of moving the software off-screen occurs. 1 monitor assumed (for multiple monitor configurations, modification of the source is probably required).

Name: Send String Through My-T-Soft

Function name: SendStringMTS

Function Arguments: Text String (with Build-A-Macro cooked format)

Returns: Nothing

Description: This communicates with My-T-Soft and hands off a text string to be processed and typed.

Notes: Version does not read the raw formats. Use the Save option in Build-A-Macro, Keystroke Macro, Zoom as the string format for this command. There is a file created (SDSTRMTS.KMF), and then the message is sent to My-T-Soft. The location is keyed off of the INI file, and must reside in the same folder as the executable.

Name: Save Position of My-T-Soft

Function name: SavePositionMTS

Function Arguments: NONE

Returns: Nothing

Description: Saves current position, for use with RestorePositionMTS, or upon re-opening of My-T-Soft.

Notes: Opposite of RestorePositionMTS. Same as Menu Option, Save Current Position.

Name: Save Settings of My-T-Soft

Function name: SaveSettingsMTS

Function Arguments: NONE

Returns: Nothing

Description: Saves current settings, for use with RestoreSettingsMTS, or upon re-opening of My-T-Soft.

Notes: Opposite of RestoreSettingsMTS. Same as Menu Option, Save Current settings.

Name: Restore Position of My-T-Soft

Function name: RestorePositionMTS

Function Arguments: NONE

Returns: Nothing

Description: Restore saved position.

Notes: Opposite of SavePositionMTS. Same as Menu Option, Restore Current Position.

Name: Restore Settings of My-T-Soft

Function name: RestoreSettingsMTS

Function Arguments: NONE

Returns: Nothing

Description: Restores saved settings.

Notes: Opposite of SaveSettingsMTS. Same as Menu Option, Restore Current settings.

Name: Query Currently Displayed Panels on My-T-Soft

Function name: QueryPanelsMTS

Function Arguments: NONE

Returns: Long value containing bits set to indicate panel displayed

Description: Queries running My-T-Soft program for current panels displayed

Notes: The bit values can be checked to examine which panels are visible (opened). The following lists the panels & their corresponding bit value - refer to the source code for particular syntax.

Keyboard (Alpha) panel, bit 1 0x00000001

Edit panel, bit 2 0x00000002

Numeric (Numpad) panel, bit 4 0x00000004

Calculator (Calc) Panel, bit 8 0x00000008

Magnifier panel, bit 16 0x00000010

System Information (Info) panel, bit 32 0x00000020

Windows Controls panel, bit 64 0x00000040

Macro panel, bit 128 0x00000080

QuickHelp panel, bit 256 0x00000100

Control panel, bit 512 0x00000200

ToolBar panel, bit 1024 0x00000400

Name: Query Current Size of My-T-Soft

Function name: QuerySizeMTS

Function Arguments: NONE

Returns: Long value current size (number)

Description: Queries running My-T-Soft program for current size

Notes: The return value is the current (internal) size of My-T-Soft

Name: Interface to SetForegroundWindow API via MTSDLL

Function name: SetInputWindowMTS

Function Arguments: Handle to Window

Returns: Result of SetForegroundWindow call

Description: Used to set the input focus (keyboard focus) to the specified window

Notes: There are various limitations in the various versions of Windows with this API and the ability for programs to control the input focus. This is just a basic interface in the DLL so an environment that doesnt have access to the Windows API can access SetForegroundWindow.

Refer to MSDN (Microsoft Developer Network) resources for limitations on this API call.

COM Edit Control Example

FOLDER: MTSEDIT

TYPE: COM control Project

SOURCE: INCLUDED

LANGUAGE: C++

IDE: Microsoft Visual C++ 6

There is a lengthy document in the MTSEDIT folder (MTSEDIT.DOC (Wordpad / Word 6.0)) that should be referenced if you are going to review the code.

Overview

This is a step-by-step example of building a COM control based on the default Edit control with calls to manipulate the My-T-Soft keyboard window. The result is an Edit control (which can be used in any environment capable of adding a COM control) that makes the keyboard window visible when it receives the keyboard input focus, and "hides" the window when it loses the input focus. Meant as a useful & educational guide for accessing the power of COM controls, it can be dropped into Visual Basic (or other high-level environment) and be used directly. For programmers familiar with COM controls, adding additional intelligence, properties, or modifying it to meet a specific set of requirements will be straightforward.

Note that all of this COM control’s capabilities can also be accomplished using the base developer’s kit compiled executables (or Windows API source code), or using the MTSDLL DLL (see Access 97 Example).

Version 1.71 - 3/17/2001

Chapter 8. Developer Tools and Examples

Keyboard based utilities

FOLDER: KEYBOARD, KEYBOARD/KYBDSYNC

TYPE: Utilities / Stand-alone Executable with Source

SOURCE: INCLUDED with KYBDSYNC

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

2 utilities are included: KeyboardSync, and KeyWatch:

Name: KeyboardSync

Program name: KYBDSYNC.EXE

Program Arguments: NONE

Requires: 1.70 version

Description: Automatically synchronizes My-T-Soft to current Windows layout

Notes: When running, it will place a small icon in the system tray (Task bar tray - Shell icon)

If My-T-Soft/My-T-Pen/My-T-Touch is running, it will monitor the current Keyboard layout, and select the appropriate layout for My-T-Soft and update the layout automatically, matching the My-T-Soft visual layout with the current Windows layout. This is most helpful when the user has 2 or more layouts that get used often.

1.77 Notes

The KeyboardSync utility has been updated to be more flexible. If the file KYBDSYNC.TXT is in the same folder as the KYBDSYNC.EXE, it will be read and used to manage the synchronization between the current input locale and the displayed My-T-Soft keyboard. The input locale information comes from Microsoft, based on the settings for Windows for more details, refer to their knowledge base, Q224804.

The logic used is as follows:

Using the GetKeyboardLayout API call, KeyboardSync based on the delay, queries the foreground window (keyboard focus/input window), and establishes the locale ID for the window. Then it attempts to match the local ID against the list read in from KYBDSYNC.TXT, and if there is a match, the My-T-Soft keyboard layout is updated, and My-T-Soft is refreshed to match.

The structure of the KYBDSYNC.TXT is important.

Lines beginning with a ; (semi-colon) are treated as comments.

Lines with local information must be structured as follows: 0x????,??,Description

The locale MUST start with 0x and follow with the 4 digit hex code (exactly 4 characters!). E.g. 0x0409,1,English

The commas are used to separate the values, and spaces should not be included for the first 2 entries. The description currently is not used by KeyboardSync, so it may contain any characters it is used for reference.

The second entry is the My-T-Soft keyboard layout value refer to the list in the My-T-Soft Help for more details.

Name: KeyWatch

Program name: KEYWATCH.EXE

Program Arguments: NONE

Requires: No special requirements, will also work with physical keyboard

Description: Displays keystroke information, scan code, virtual codes, extended, etc.

Notes: Useful for debugging and comparing input approaches to the keyboard. Runs as a stand-alone application, displays last keystroke, and streams keystroke display, automatically clearing when screen if full. Really a programmer debugging tool, it has helped to resolve some issues regarding how My-T-Soft generates shift sequences, etc.

Name: KeyEnter

Program Name: KEYENTER.EXE

Program Arguments: NONE

Requires: Nothing

Description: Generates an Enter Keystroke

Notes: Asked for by a customer to handle a particular application.

Name: KeyBack

Program Name: KEYBACK.EXE

Program Arguments: NONE

Requires: Nothing

Description: Generates an Backspace key Keystroke

Notes: Built along with KEYENTER

Name: KeyTab

Program Name: KEYTAB.EXE

Program Arguments: NONE

Requires: Nothing

Description: Generates an Tab key Keystroke

Notes: Built along with KEYENTER

Name: KeyEscape

Program Name: KEYESC.EXE

Program Arguments: NONE

Requires: Nothing

Description: Generates an Escape key Keystroke

Notes: Built along with KEYENTER

Version 1.74 - 4/3/2002

Modifying Keyboard Layouts

FOLDER: LAYOUTS

TYPE: Keyboard Layout source for My-T-Soft

SOURCE: INCLUDED

LANGUAGE: ASM (Assembler files)

IDE: Text editor / NASM

The Keyboard Macro File (KMF) - (e.g KYBD0001.KMF) controls the display & operation of the particular layout for My-T-Soft (My-T-Touch/My-T-Pen), etc.

The layout itself is a binary file, which is generated from an ASM (e.g. ASeMbler). The reason for this is the file layout structure - it is a set of offsets & links, so that all of the data can be variable length. Over time, certain areas have become fixed, while other areas are changed to create the different layouts.

The following documents and provides an overview to the sections within the ASM file. Before getting into all the detail, here is a quick example of modifying one of the files:

Quick Example

You want to change the # symbol to a + symbol in the normal US 101 layout. Edit KEYBRD01.ASM with your favorite text editor, find the LBL000: section, find the # symbol (e.g. LBL003: DB '#',0)

And change it to LBL003: DB '+',0

Save the file

Now rebuild the KMF file:

nasmw -f bin KEYBRD01.ASM -o KYBD0001.KMF

Overwrite the existing KYBD0001.KMF in \Program Files\[ProductDir], and when this layout is used (e.g. Keyboard=1 in the INI file), the shift-3 key will show a + symbol instead of the # symbol.

A few notes on the example:

In general, it is recommended that you preserve the existing files (unless there is a fundamental problem that you are trying to resolve), and use numbers in the 3000 range (e.g. KYBD3001.KMF). You should also modify the layout name in the ENDDATA at the end of the file (or with LAYOUT_NAME in those that have the LAYOUT_NAME define), so there is a unique entry in the Keyboard layout selection in Setup.

Modifying KMF Source files (ASM files)

A quick lesson in assembler syntax:

The EQUs are equates, or macro replacements (similar to #define in C) These should not be touched.

DB means Define Byte (or create a single Byte of data based on the following human readable entry)

DW means Define Word (or double byte, 16-bits)

DD means Define Double word (or 32-bit value)

A label is a non-reserved word followed by a colon, e.g. KEY000:

Text can be represented with single-quotes, e.g. This is some text,0 (always end text with a null terminating byte, in case it us used as a string internally!) Decimal values are OK, but hex is used often. To represent hexadecimal, use a leading 0 and a trailing H, e.g. 020H is 2x16+0, or 32 decimal - these are all the same when compiled to binary:

DB 32 ;this is a decimal 32, or a byte with a decimal value of 32 will be placed into the file

DB 020H ; hexadecimal representation of 32 decimal, e.g. a 2 in the 16s place and a 0 in the ones place

DB ;this is a text - the character for the space is decimal 32

ASM File Overview

For all practical purposes, these are the 12 areas to be familiar with:

1) The header (after the equates (EQU is like #define in assembler)). This is followed by a long listing of offsets, handled as fixed length data objects, with various offsets into this table kept in the header.

2) The KEY scan codes, e.g. KEY000:

3) The Label pool,e.g. LBL000:

4) The IDKeys section KMF001: (normal display)

5) The IDShiftKeys section KMF002: (Shifted display)

6) The IDShiftAltGrKeys section KMF003: (Shift AltGr display)

7) The IDAltGrKeys section KMF004: (AltGr display)

8) (* NOT RELEVANT IN ALL LAYOUTS!) The IDKeysCAPS section KMF005: (normal display - CAPS overrides)

9) (* NOT RELEVANT IN ALL LAYOUTS!) The IDShiftKeysCAPS section KMF006: (Shifted display - CAPS overrides)

10) (* NOT RELEVANT IN ALL LAYOUTS!) The IDShiftAltGrKeysCAPS section KMF007: (Shift AltGr display - CAPS overrides)

11) (* NOT RELEVANT IN ALL LAYOUTS!) The IDAltGrKeysCAPS section KMF008: (AltGr display - CAPS overrides)

12) The Layout Name (ENDDATA)

In most layouts, only the labels need to change (and in some international layouts, occasionally the KEY scan codes may require some changes).

This is how the file is laid out:

Header

Fixed Table of offsets into actual data

Actual data

This file structure allows variable actual data, while the reference (or offset/pointer) to it is always in the same location. By keeping the raw data in Assembler form, it can be edited fairly easily by humans (once

you become familiar with the areas & the implementation). Because the format is compact in its binary form, does not need to be changed that often, the format has stuck.

ASM File Details

Here are the details on the sections:

```
#########################################################################
### (1) HEADER EXCERPT ##################################################
```

The first 32 bytes are a header

Note the KeyLabelOffset coded here, and used in the Key Label "lookup" below

```
DB 'MK77' ;signature, subject to change BYTE 0,1,2,3
DB 32 ;Length of header (32) BYTE 4
DD OFFSET ENDDATA ;File Pointer to next entry (5 bytes) BYTE 5,6,7,8,9
DB 0
DD 65536 ;File ID BYTE 10,11,12,13
DW 0 ;Reserved BYTE 14,15
DW 0 ;ScancodeOffset BYTE 16,17
DW 256 ;QuickHelpOffset BYTE 18,19
DW 476 ;BaseMacroOffset BYTE 20,21
DW 542 ;KeyLabelOffset BYTE 22,23
DW 772 ;MacroPanelOffset BYTE 24,25 ;Base KMF look up tables
DW 0 ;PanelKeyOffset BYTE 26,27
DW 0 ;Char Set BYTE 28,29
DW 0 ;Attributes BYTE 30,31
```

The Char Set (28/29), and Attributes (30,31) are the only items that should be changed. The Char Set matches the ANSI code page, as defined by Microsoft The relevant ones are documented in the My-T-Soft help under Fonts.

The Attributes are bits that signify certain internal operation.

Low Byte is bits for specific handling

Internal Header #defines:

```
#define KMFHDR_TRACKALTGR 0x0001
#define KMFHDR_NOLOWERCASEDISPLAY 0x0002 (NOT USED ANYMORE)
/* Asian Keyboards */
#define KMFHDR_CODEPAGEINENDDATA 0x0004
/* Localized KMF files / keys over 104 (ScaleAFont) 1.78*/
#define KMFHDR_LOCALIZEDKMFFILE 0x0008
/* Weird handling (in code) - Use 16 values in low nybble of high byte */
```

#define KMFHDR_HEBREWHANDLING 0x0100

#define KMFHDR_FRENCHHANDLING 0x0200 (Old, could use CAPSTABLES)

#define KMFHDR_MULTIBYTEHANDLING 0x0400

#define KMFHDR_CAPSTABLES 0x0800

##

KEY EXCERPT

The KEY entries are the actual scan codes

This is the scan code pool (lookup table) for the KMF000 entries below

KEY000: DW 0001BH

KEY001: DW 00070H

KEY002: DW 00071H

KEY003: DW 00072H

##

Label Pool -LBL EXCERPT

These are the labels used on the displayed keys

This is the Label pool (lookup table) for the different keyboard states KMF001, KMF002, KMF003, KMF004

e.g. Regular, Shifted, Shift Alt-Gr, and Alt-Gr

LBL000: DB '~',0

LBL001: DB '!',0

LBL002: DB '@',0

LBL003: DB '#',0

LBL004: DB '$',0

SPECIAL LABEL NOTES:

Use Unicode character - 255,'U',D1,D2,D3,D4 where D1-D4 is hex "text" representation, e.g. DB 255,'U','0','6','D','E' - or use Unicode value 06DEH, (or 0x06de) (or unicode value 1758 (in decimal))

Also DB 0xFF,'U','0','6','D','E'

Note the ending 0 is not assumed or required, but ALL 4 Unicode hex digits are required!

The unicode character makes the most sense with KMFHDR_CODEPAGEINENDDATA, as these characters most likely will require a special code page for the WideCharToMultiByte to pull them up & display correctly. In general, this really isn't the unicode character, as the code page limits what will really be displayed.

Use Multiple Unicode characters - 255,'M',D1,D2,D3,D4,D1,D2,D3,D4,0 or 255,'M',D1,D2,D3,D4,D1,D2,D3,D4,D1,D2,D3,D4,0

This works very similar to above (with same issues with code pages) The internal logic always assumes at least 2 characters (e.g. if there was only 1, use the 'U' above, not the 'M'!).

The third character is optional, and the check for the ending 0 (null value) is used to see if only 2 (or if there is a third). Internally, the M is changed to a U and the code falls through into the handling for U automatically. The characters 2 and 3 may have spacing issues, depending on the actual character. Although this was implemented & tested, it does not fully handle the spacing for the 2nd (or 3rd) characters. In general, this shouldn't be used, but was left in the code in case it needed to be improved at a later time.

Override character length as 1 - 255,'1',C1,[C2,][C3,][C...,]0, this allows multiple characters to be used, but the font size will be used internally as if only 1 character is used.

If the KMFHDR_CAPSTABLES attribute is used (0800H), there is also an "override" label available. By specifying a 3 character entry, e.g. 0FFH,0FFH,0BDH, (or 255,255,[some other char value]), this type of entry will use the third character without any modification (since some of the internal caps logic says if it is a 1 character label, use AnsiUpper/AnsiLower if LowerCaseDisplay=1 in the INI) This forced upper/lower case change for single character displays can cause some issues on international layouts, and if it affects a normal or shifted state (e.g. KMF001/KMF002), using a label with the 3 character override (leading 255,255/0FFH,0FFH) will allow selection of the unmodified character value for display.

```
########################################################################
### IDKeys - KMF EXCERPT ###############################################
IDKeys
This is the regular layout
Note the 542 offset to refer to the LBL pool
KMF001: DW 542+21 ;Esc
DW 542+22 ;F1
DW 542+23 ;F2
########################################################################
### IDShiftKeys - KMF EXCERPT ##########################################
IDShiftKeys
This is the shifted layout
0 means no change from the regular, rather than no display
The 542 offset is used to refer to the LBL pool
KMF002: DW 0 ;Esc
DW 0 ;F1
DW 0 ;F2
########################################################################
### IDShiftAltGrKeys - KMF EXCERPT #####################################
IDShiftAltGrKeys
This is the shifted Alt-Gr layout
0 means no display
The 542 offset is used to refer to the LBL pool
```

KMF003: DW 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ;20

DW 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ;20

##

IDAltGrKeys - KMF EXCERPT

IDAltGrKeys

This is the Alt-Gr layout

0 means no display

The 542 offset is used to refer to the LBL pool

KMF004: DW 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ;20

DW 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ;20

##

CAPS SECTION - SIMILAR TO ABOVE###################################

0 means no change from the current display, rather than no display

KMF005 is CAPS override to KMF001

KMF006 is CAPS override to KMF002

KMF007 is CAPS override to KMF003

KMF008 is CAPS override to KMF004

As overrides, they should be 0, unless the CAPS state for that key label is wrong. If it is wrong, then you need to specify the correct label, e.g. 542+123 - the internal logic, says if this is not 0, then lookup the specified label & use that rather than what would have been displayed.

##

Layout Name (ENDDATA)

ENDDATA:

DB ’104 United States Standard’,0

_TEXT ENDS

Building KMF Files from ASM files

This section defines how to build the asm files

These examples are for the Borland Turbo Assembler (i.e. TASM), used by IMG. These notes are for IMGs developers only - see below for using NASM.

This is for 101 layouts (%1 = 23 for example):

TASM KEYBRD%1

TLINK /3/x/t KEYBRD%1,KYBD00%1.KMF

DEL KEYBRD%1.OBJ

This is for 104 layouts (%1 = 123 for example):

TASM KYBD0%1

TLINK /3/x/t KYBD0%1,KYBD0%1.KMF

DEL KYBD0%1.OBJ

The layouts included in the Developers Kit have been modified slightly to work with nasm (GPL assembler). The included zip nasm-0.98.39-win32.zip was downloaded on April 27, 2007 from http://nasm.sourceforge.net - (you may wish to go there and obtain the latest stable version). The COPYING file describes the license for this software. The modified KMF layout ASM files contain a %include nasm.inc to bypass & ignore certain entries from the original TASM source files.

File: nasm.inc

%idefine offset

%idefine _TEXT

%idefine SEGMENT

%idefine PUBLIC

%idefine BYTE NOTHING EQU

%idefine ENDS

To build a KMF file for use with My-T-Soft, you can do the following with the nasm binaries in the nasm-0.98.39-win32.zip file.

nasmw -f bin KEYBRD01.ASM -o KYBD0001.KMF

The binary file can be copied to the \Program Files\[ProductDir] folder, and can be used directly.

There is a BUILD.BAT file that will make the typing a bit easier, e.g. you just specify the number (Build 01 for Keyboard 1, Build 105 for Keyboard 105, etc.). Here is the BUILD.BAT text:

@echo off

if %1XX == XX goto NOGOOD

if exist KEYBRD%1.ASM goto BUILD101

if exist KYBD0%1.ASM goto BUILD104

goto NOMATCH

:BUILD101

nasmw -f bin KEYBRD%1.ASM -o KYBD00%1.KMF

ECHO Built KYBD00%1.KMF

GOTO END

:BUILD104

nasmw -f bin KYBD0%1.ASM -o KYBD0%1.KMF

ECHO Built KYBD0%1.KMF

GOTO END

:NOMATCH

Echo Problem finding an ASM file!!

ECHO "%1"

ECHO See following build specs:

goto NOGOOD

:NOGOOD

ECHO No ASM FILE SPECIFIED! You must specify a specific

ECHO keyboard layout ASM file, e.g.

ECHO build 01[Enter]

ECHO -or-

ECHO build 102[Enter]

goto END

:END

Example & instructions making changes between KMF & testing

Test procedures in XP Pro

Some things to know about keyboards - AltGr, Dead Keys

AltGr - some international layouts provide a whole new mode to access even more characters.

The right-hand Alt key becomes AltGr (which I believe was originally for Alt Graphics), and it creates 2 possible new states - AltGr, and Shift-AltGr.

Dead Key - is an accenting key, used mostly for accent characters. The way this works in practice, is you type the key, and NOTHING shows up !!! (hence, dead key), but when you type the next character that can be accented, e.g. e, you will see ï¿½instead of e. Dead Keys (when typed on 2 times, will generate 2 accent characters - this is important when testing, since you can type a key & get nothing, and this could mean 1 of 2 things - it is a blank/not used key, or it is a dead key!)

To test any layout, you must be in the appropriate "Locale" with the correct keyboard layout selected.

For example, in Turkey, there is a Type F/Type Q layout. If you have Type F in My-T-Soft, but Windows thinks it is Type Q, you'll have lots of problems.

This is the best way to test in Windows XP

Load Wordpad

Load the Locale

Set the font fairly large, e.g. 24 or 26

Type with the physical keyboard & My-T-Soft keyboard to verify everything matches.

On All keys, you must press the key 2 or 3 times, to verify it is not a dead key (dead keys will not generate any visible output if only pressed once!)

For reference to layouts, you can try: http://www.microsoft.com/globaldev, then when loaded, on the left-hand column under the GlobalDev Home, find References | Keyboard Layouts. Once at the layout page, you can select from the available layouts - this will load a popup that shows the layout/dead-keys, etc.

Some helpful hints:

Show the locale / internationalization / layout selection bar in a window - if floats towards the top-right

Compare physical keyboard to My-T-Soft results first, e.g. type with the physical keyboard, then type with My-T-Soft - they should match, then verify the display in My-T-Soft matches the results

These are 8 possible states to be aware of:

Normal, Shifted, Caps & Caps-Shifted

If the keyboard supports AltGr, then you have:

Normal, Shifted, Caps, Caps-Shifted, AltGr, Shift-AltGr, Caps-AltGr, Caps-Shift-AltGr

Some advanced notes:

Setting KeyWatch=1 in the INI file, shows you the number of the key internal to My-T-Soft KeyWatch.exe (from Developer's Kit/KEYBOARD folder) shows virtual key/actual results for comparing physical keyboard/My-T-Soft generated keystrokes

Version 1.78 Release 2 - 5/4/2007

Add-On DLL Template

FOLDER: ADDONDLL

TYPE: External Capabilities for My-T-Soft

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

Overview:

The Add-On DLL capability provides the mechanism for an external DLL to integrate tightly with My-T-Soft, allowing various add-ons to operate in conjunction with the keyboard executable program during run-time. This is a fairly advanced topic, and the examples provided gives the developer all they need to use this effectively.

Examples:

- Communicating with another process in the system (through a shared DLL)

- Setting and Removing a system hook during the run of My-T-Soft

- Monitoring My-T-Soft for Developer Kit manipulation during its run via a DLL (rather than another process in the system)

- Logging Start/End/Run times of My-T-Soft

- Integrating with an Application to handle whether My-T-Soft is available or not

There are 3 integration options, with 9 different parameter options:

- Initialization (Startup)

- Run Time (System Timer controlled)

- Termination (Cleanup)

To enable this capability, the INI (MYTSOFT.INI/MYTTOUCH.INI//MYTPEN.INI) must be updated to include the relevant information (see below for further details).

Conceptually, this is how this works:

When the My-T-Soft executable begins, it reads the INI & if these [AddOnDLL?] section entries are correct, it will load the library, obtain the function address & call the function at the appropriate time (at init, via timer, or at termination). The DLL MUST be written in C, and have proper exports, handle the parameters correctly, etc. This isn't meant to be fancy or complicated it is done as straightforwardly as possible.

This is a My-T-Soft Developer's Kit example file for Windows, written in C. The project is in the ADDONDLL folder, created in Microsoft Visual C Version 6. For best results, play with the ADDONDLL.EXE & the INI first to understand how this works, then refer to the LOGDLL to see another example.

The example provided in the ADDONDLL folder is purely a template and an example of how to do the actual integration it only shows what is possible, it does not actually accomplish anything useful. You can refer to the LOGDLL example for a simple logging integration example.

To integrate and see the example, you need to update the INI file with the [AddonDLL1] section below, then run ADDONDLL.EXE to see how this works.

The following is an excerpt from the ADDONDLL.EXE Window display:

This window is simply a container to show the results of DLL calls to the ADDONDLL.DLL code, when integrated with the AddOnDLL entries in the .INI file.

This is a fairly advanced aspect of the software, and the developer should be familiar with DLLs and calling them dynamically using LoadLibrary and GetProcAddress Windows API calls. Furthermore, this approach presupposes a complex task that requires the use of a DLL to begin with (rather than externally from an application, or using other Developer Kit approaches).

This example was created to test the various options and ensure proper operation of the Initialization, Cleanup, and Timer based function calls. For a particular application of this approach, the unused functions can be removed if desired.

The following is an example entry from the .INI file to illustrate the use of this DLL. In the example, the InitFuncType7 function in the DLL will be called with the Value1 entry at Initialization of My-T-Soft, then during the run of My-T-Soft, the TimerFuncType9 in the DLL (with all supported parameters) will be called every second (1000 ms), and when My-T-Soft is closed, CleanupFuncType8 in the DLL will be called with the 3 value entries.

[AddOnDLL1]

DLLName=%PRODDIR%ADDONDLL\ADDON.dll

InitializeFunctionName=InitFuncType7

InitializeFunctionType=7

TimerFunctionName=TimerFuncType9

TimerFunctionType=9

TimerDelay=1000

CleanupFunctionName=CleanupFuncType8

CleanupFunctionType=8

Value1=1

Value2=201

Value3=3302

Notes:

There are 3 sections available - [AddOnDLL1], [AddOnDLL2], [AddOnDLL3]. Each section can have the Init, Timer, and/or Cleanup functions specified independently. By default, the FreeLibrary will be called at Cleanup, whether or not there is a Cleanup function specified. The [Type]FunctionName entries are ANSI (1 Byte Characters), and expect to be loaded in a C based DLL. A C++ DLL must export the function names as C (e.g. extern "C"), so any name mangling from C++ compilers does not change the actual exported name of the function. The 9 function types cover various aspects and address various needs. Most are of a C Declaration type (default "cdecl" assumed), but the WINAPI entries (similar to a CALLBACK entry) resolves to a FAR PASCAL, so these must carry the same type of declaration if built in another DLL. It is best to start with the declarations, or even start with the ADDON.C file, and strip out / rename pieces as desired, being sure to keep the rest of the declarations and definitions intact.

The TimerDelay entry is in milliseconds, and is used in a call to the Windows API SetTimer internally if there is a non-0 value for TimerDelay, TimerFunctionType, and some text in TimerFunctionName.

Error checking is basic, and error conditions are not reported. If the DLL is not loaded, or the function name not found, My-T-Soft will continue normally, but never call the DLL function.

When integrating an external DLL using this mechanism, verify path, file name, and function name entries carefully. You may want to verify your DLL function is being called, before implementing any real functionality in the DLL function.

If memory errors occur, verify the calling conventions, and verify you are not returning values for void functions, or different types than specified for the pre-defined function types.

Version 1.78 - August 9, 2005

Log DLL (ADDONDLL Example)

FOLDER: LOGDLL

TYPE: External Capabilities for My-T-Soft

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

Located in the LOGDLL folder, this pre-built DLL (source code & Microsoft Visual C++ Version 6 project & workspace included- available with the Developers Kit extracted) will log start & end times during runs of My-T-Soft. The INI entry must be:

[AddOnDLL1]

DLLName=%PRODDIR%LOGDLL\LOGDLL.DLL

InitializeFunctionName=StartLog

InitializeFunctionType=2

TimerFunctionName=TimerLog

TimerFunctionType=2

TimerDelay=1000

CleanupFunctionName=EndLog

CleanupFunctionType=2

Value1=0

Value2=0

Value3=0

Example of Log (My-T-Touch) (C:\WINDOWS \MYTTOUCH\MYTTOUCH.LOG):

*** START RUN OF C:\WINDOWS\MYTTOUCH\MYTTOUCH.EXE

*** User: Username

*** Timestamp: 12:14:44 on 9 Aug 2005.

END RUN (4 seconds), Timestamp: 12:14:49 on 9 Aug 2005.

***

*** START RUN OF C:\WINDOWS\MYTTOUCH\MYTTOUCH.EXE

*** User: Username

*** Timestamp: 12:14:50 on 9 Aug 2005.

END RUN (1 seconds), Timestamp: 12:14:52 on 9 Aug 2005.

***

*** START RUN OF C:\WINDOWS\MYTTOUCH\MYTTOUCH.EXE

*** User: Username

*** Timestamp: 12:14:54 on 9 Aug 2005.

END RUN (7 seconds), Timestamp: 12:15:01 on 9 Aug 2005.

***

*** START RUN OF C:\WINDOWS\MYTTOUCH\MYTTOUCH.EXE

*** User: Username

*** Timestamp: 15:16:46 on 9 Aug 2005.

END RUN (10 seconds), Timestamp: 15:17:05 on 9 Aug 2005.

***

Version 1.78 - August 9, 2005

Paint DLL and external painting interface

FOLDER: PAINTDLL

TYPE: External Capabilities for My-T-Soft via DLL

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

The PaintDLL.DLL with source code outlines what is necessary to integrate external painting to customize and control more of the visual display of My-T-Soft.

The ExternalPaint settings allow you to control the key painting in several ways:

Paint the key background (PaintKeyBackground)

Modify the default key background (PaintKeyModBackground)

Paint the key labels (PaintKeyLabel)

Paint the entire background of the panel (PaintFrame)

Completely Paint the key, bypassing internal painting (PaintKey)

By using internal key values & keyboard info, you can also do small scale modifications, or large scale modifications to the visual display of My-T-Soft!

IMPORTANT NOTE: The ConfigPath determines the location of the INI where you need to modify and add the PaintDLL example below. Refer to the My-T-Soft Setup | File | Show Config File Location for the location, or use My-T-Soft Setup | Configuration | Special Options | Edit My-T-Soft Initialization file.

Here are the relevant settings from the MYTSOFT.INI to achieve examples seen on website:

In [Configuration], the transparency is on, e.g.

Transparency=1

TransparencyLevel=161

The default PaintDLL.DLL is built from the Developer's Kit sample, and this is the MYTSOFT.INI settings:

[PaintDLL]

DLLName=%PRODDIR%\PaintDLL\Release\PaintDLL.DLL

PaintFrame=1

PaintFrameFunc=PaintDLLPaintFrame

PaintKey=0

PaintKeyFunc=PaintDLLPaintKey

PaintKeyBackground=1

PaintKeyBackgroundFunc=PaintDLLPaintKeyBackground

PaintKeyModBackground=0

PaintKeyModBackgroundFunc=PaintDLLPaintKeyModBackground

PaintKeyLabel=1

PaintKeyLabelFunc=PaintDLLPaintKeyLabel

The key down & up bitmaps were modified to be gray-scale and copied into the Release folder for PaintDLL

If you refer to the Sounds / Advanced notes in the My-T-Soft Manual, you can find a key # reference that can be used within the PaintDLL code to handle specific keys.

This is the sequence the keyboard panels are painted.

The frame (entire panel) is painted (PaintFrame can be used to paint your own frame) Each key is painted in sequence based on its key number

For each key, this is the sequence:

The key background is painted (PaintKeyBackground can be used to paint your own background, or PaintKeyModBackground can be used to modify the default (internal) painted background) The key label (or image) is painted (PaintKeyLabel can be used to paint your own label).

Finally, PaintKey can be used to completely replace or modify a fully painted key

This documents the settings used with the external PaintDLL.DLL

[PaintDLL]

All settings must be in a section headed by [PaintDLL]

DLLName=%PRODDIR%\PaintDLL\Release\PaintDLL.DLL

The DLLName entry must contain the full path and name of the DLL to be used as the external interface. %PRODDIR% expands to the current location of MYTSOFT.EXE, and

%WINDIR% uses GetWindowsDirectory

PaintFrame=1

PaintFrameFunc=PaintDLLPaintFrame

see below for general usage - PaintFrame paints the background of the entire panel - see

source for panel reference

PaintKey=0

PaintKeyFunc=PaintDLLPaintKey

see below for general usage - PaintKey can be used to modify the normal painted key (i.e. adding items or modifying the normal key as it is painted) This is called after the entire key has been painted, e.g. the background and the lable has been painted.

PaintKeyBackground=1

PaintKeyBackgroundFunc=PaintDLLPaintKeyBackground

see below for general usage - PaintKeyBackground can be used to paint the background of the key, leaving the key label to be painted by My-T-Soft

PaintKeyModBackground=0

PaintKeyModBackgroundFunc=PaintDLLPaintKeyModBackground

see below for general usage - PaintKeyModBackground can be used to take the default key background painted by My-T-Soft and modify it prior to painting the label.

PaintKeyLabel=1

PaintKeyLabelFunc=PaintDLLPaintKeyLabel

see below for general usage - PaintKeyLabel can be used to paint the text in a different style, or use other labels

General Usage

The settings are listed as a pair - an entry that is used internally by My-T-Soft to determine if the function should be used - if 1, the named function is used, and if 0, the function is not used. If the function is to be used, the Func label entry lists text that is used internally using the GetProcAddress to find the appropriate function in the PAINTDLL.DLL

Refer to the PAINTDLL.C for some examples that paint a custom look keyboard and further notes.

Developers Kit Version 1.78 Release 3 - 9/22/2007

WordsDLL and external Word List interface

FOLDER: WORDSDLL

TYPE: External Capabilities for OnScreen via DLL

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

The WordsDLL.DLL with source code offers 2 callback type functions that allow external integration of word list / WordComplete candidates for display & usage within OnScreen.

This allows an external DLL to supply the words displayed by OnScreen in the WordComplete panel. The example provided does no specific work, it only shows the interface, and gives the source code to build a DLL that can interface with OnScreen.

As a general overview, OnScreen calls GetWordList every time it needs to fill the WordComplete candidates (see below for details on the parameters used). As a way to provide feedback, OnScreen calls TypedWord to tell the word list when the user selects a word on the WordComplete panel (or when the user presses a space or punctuation after a buffer bas been built (i.e. a new word has been typed in OnScreen)).

This is the C prototype for the GetWordList function called by OnScreen as specified in the WordsDLL entry for the WordsGetWordListFunc entry.

BOOL WINAPI GetWordList(LPTSTR lpCurrentBuffer,DWORD dwReturnThisManyWords, DWORD dwStartListAt,LPTSTR lpCommaDelimitedList,DWORD dwSizeOfList,LPTSTR lpLetterAssistList);

Parameters:

[in] LPTSTR lpCurrentBuffer - the current letter/letters typed, e.g. "th" would return "the,that,their,they,these," when the first character is a - (dash), suffixes should be returned.

[in] DWORD dwReturnThisManyWords - number of words to return (typically 5 from OnScreen)

[in] DWORD dwStartListAt - start List At - will be 0, 5, 10 from OnScreen (based on the More key used by the user) As an example, if there are 23 possible words than match th, then the first list will StartAt 0 - if the user presses More, then the next 5 in the list will be returned.

[out] LPTSTR lpCommaDelimitedList - return list of words, comma delimited, ending with a comma

[in] DWORD dwSizeOfList - max buffer size for lpCommaDelimitedList

[out] LPTSTR lpLetterAssistList - comma delimited list of possible "next" letters (max size Windows API MAX_PATH), pass in NULL if not required - must return UPPER-CASE list, e.g. "A,C,E," this is used by OnScreen to highlight or allow letters that can build words in the current word list based on the current buffer.

TypedWord - external call to tell DLL what word was typed

BOOL WINAPI TypedWord(LPTSTR lpWord);

[in] LPTSTR lpWord - The word / suffix that was typed (suffixes have leading '-')

The following entry must be in ONSCREEN.INI to use the WordsDLL.DLL

[WordsDLL]

DLLName=%PRODDIR%\WordsDLL\Release\WordsDLL.DLL

WordsGetWordListFunc=GetWordList

WordsTypedWordFunc=TypedWord

This documents the settings used with the external WordsDLL.DLL

[WordsDLL]

All settings must be in a section headed by [WordsDLL]

DLLName=%PRODDIR%\WordsDLL\Release\WordsDLL.DLL

The DLLName entry must contain the full path and name of the DLL to be used as the external interface. %PRODDIR% expands to the current location of ONSCREEN.EXE, and %WINDIR% uses

GetWindowsDirectory

WordsGetWordListFunc=GetWordList

This lists the name of the GetWordList function that is used internally with GetProcAddress :to obtain the function name from the specified DLL.

WordsTypedWordFunc=TypedWord

This lists the name of the TypedWord function that is used internally with GetProcAddress :to obtain the function name from the specified DLL.

IMPORTANT NOTE: The ConfigPath determines the location of the INI where you need to modify and add the PaintDLL example below. Refer to the My-T-Soft Setup | File | Show Config File Location for the location, or use My-T-Soft Setup | Configuration | Special Options | Edit My-T-Soft Initialization file.

Because of the complexity of integrating something this complicated, there may be integration issues when working with this capability. If you experience problems, limitations, or need further assistance with this DLL, please contact developer support.

Developers Kit Version 1.78 Release 3 - 9/22/2007

Ctrl-Alt-Delete Emulation Example

FOLDER: CTALTDEL

TYPE: Utility / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C

IDE: Microsoft Visual C++ 6

This is a utility that contains 2 modes of operation based on the windows platform that executes it. In Windows 95/98/Me, the utility will restart the system. In Windows NT/2000/XP, if the IMGLOGON.DLL is present & in use, will popup the security dialog that allows the user to lock the workstation, logoff, run the task manager, etc. The source code is included, along with project information for MS VC++ Version 6. The example includes the source code to generate a "soft" SAS (Secure Attention Sequence) to allow access to system maintenance functions.

The default action in 95/98/Me is to restart the system, but this could be modified to bring up the Task Manager (TASKMAN.EXE). The "Close Program" dialog that appears in 95/98/Me is tied to the physical interrupt created by using the Ctrl-Alt-Del combination on the physical keyboard.

Version 1.72 - August 28, 2001

Modal Input Utility

FOLDER: MODAL

TYPE: Utility / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C / Windows API

IDE: Microsoft Visual C++ 6

This is a simple utility (source included) that provides a mechanism to produce modal input using My-T-Soft. In general, the on-screen keyboard is modeless, in that the user can select at will what they want to have the keyboard focus, and use My-T-Soft to type anywhere they choose. A customer desired to make the keyboard modal, i.e. that your choice was to type with the keyboard, or exit the field and get out of that mode. In keeping with the terminology for dialogs, they wanted a Modal keyboard, i.e. the user must respond and enter some text, or cancel out of that situation (by leaving the input field). After discussing the situation thoroughly, it was determined that modifying My-T-Soft to somehow create this user-input situation made little or no sense - the software has been, and always will be modeless, and adding this feature to the software could only cause trouble or confusion for the user. For example, what would turn this on, and how/when? Plus the point of modal input is that this situation should be application driven, so this capability should not be a feature or option in a virtual (or physical) keyboard. There are just too many variables, and requires too much intelligence on part of the keyboard software - a keyboard is just an input device for the user, not some intelligent software that should be allowed to control where & when or how the user can operate the system.

The Modal utility is ideal for the following situation - the developers kit utilities are being used to bring the keyboard up as needed, and moved off-screen when not needed. When the keyboard is visible, you do not want the user to do anything else with your full screen application - just enter the text and finish with the field. At field exit, the keyboard software is dismissed, and the modal utility is stopped.

For modal input, the real solution was not allowing the user (touchscreen based) a chance to do anything but click on the keyboard. So bringing up a transparent window (WS_EX_TRANSPARENT), that was topmost (WS_EX_TOPMOST), and could not be activated (WS_EX_NOACTIVATE), painting it with the NULL_BRUSH, and covering the whole screen, then positioning it right after My-T-Soft window results in making My-T-Soft Modal. Mouse (touchscreen) clicks to any other part of the screen result in nothing (these are ignored by the Modal transparent window), so only clicking on My-T-Soft does anything useful for the user. Nothing changes the input / keyboard focus when Modal is run, so its presence changes nothing except prevent unwanted mouse input. To dismiss this modal window, running Modal.exe again triggers a message to close the existing process (close window), and the second process does not continue (it simply returns FALSE from WinMain).

To use this effectively, you run Modal.exe at field enter, and run it again at field exit, and you will have My-T-Soft Modal input while the field has the input / keyboard focus.

This example is not only useful, but shows an ideal solution using the developers kit utilities to create a very effective application interface for the end-user.

Version 1.78 Release 2 - 5/4/2007

MTSAppBar

FOLDER: MTSAppBar

TYPE: Utility / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C++ / Windows API

IDE: Microsoft Visual C++ 6

MTSAppBar interface is primarily for OnScreen - not tested with other builds.

The MTSAppBar source code was modified from the AppBar example available from Microsoft showing and integrated with OnScreen to trap OnScreen on the AppBar. The AppBar shell interface within Windows provides something similar to the Auto-Arrange windows in OnScreen - a way to limit Applications screen area and separate the OnScreen display from the Application being typed into. If actually used, turning off Auto-Arrange in OnScreen Setup | User Options | Operation Options, and clearing items like Allow Action Button move, etc. may make using OnScreen with the AppBar more effective.

The MTSAppBar executable creates a top or bottom based "AppBar" or a special window that changes the desktop window size.

This has a timer running (internal ID: IDT_MONITOR) that watches the OnScreen keyboard window, and reacts if the window is sized or moved - this resets the appbar to go to the top or bottom & size to the current keyboard size.

The code was modified from Microsoft sample code that created a more customizable appbar, similar to the Windows taskbar - so certain modifications were done to limit what could be done with this implementation of the AppBar.

Most of the code remains untouched, but there were minor modifications throughout. Here are notes on the changes:

Comments were modified throughout

UTIL.C - removed (slidewindow)

PROPSHT.C - removed (property sheet dialog)

GLOBALS.H - IDT_MONITOR ADDED

APPBAR.H - comments only

MYTSOFT.H - added, but slightly modified from normal Developer's Kit MYTSOFT.H (DEVKIT folder)

MAIN.C - code was added to the bottom for various utility aspects required, mainly coordinating with the My-T-Soft code - there were also add-ons in WinMain to prepare operation.

The window was created with out a thick frame (and the sizing/moving code modified - only menu commands can move window, and only to top & bottom)

APPBAR.C - Minor modifications (force desktop window redraw whenever _SetSide called

WNDPROC.C - Various modification throughout to accommodate changes to implementation (mostly OnTimer, OnCommand, OnCreate, OnPaint code removed)

APPBAR.RC - the about dialog modified, and menu modified, along with icon & bitmap

Comments:

The WinMain prep (MAIN.C) and OnTimer code (WNDPROC.C) is the most significant, along with the call to GetMTSAppBarHeight (MAIN.C) to establish correct window height.

The PrepareMyTSoft works with DevKit tools & messages to establish the correct INI file, and verify the MTSAppBar entry is in the DoNotArrangeClasses (Very IMPORTANT for correct operation).

Usage Notes:

Expects OnScreen to be running before it is launched (otherwise messagebox shown). Option to add a "WAIT" command line option to run in a "Sleep" (pause) loop to wait for OnScreen to open & paint (in case both are started at the same time (i.e. Windows StartUp Group). Run in a shortcut with "...\MTSAppBar.exe" WAIT

The code was built & tested in Microsoft Visual Studio 6 with Windows 2000.

Developers Kit Version 1.78 Release 3 - 9/22/2007

MTSDock (MTSAppBar)

FOLDER: MTSAppBar\MTSDock

TYPE: Utility / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C++ / Windows API

IDE: Microsoft Visual C++ 6

MTSDock interface is primarily for My-T-Soft 2.20 (Build-A-Board Run-Time MSWIN32) - not tested with other builds. When run with the example Keyboard Files (KBFs) and the MSWIN32 target, there is a side panel (right-side) and bottom full keyboard that can be toggled. By using this with a maximized application (i.e. a browser based interface/maximized) a clean and workable entire interface can be handled with specialized keystrokes and access to a full keyboard.

The MTSDock source code was modified from the MTSAppBar example (parent folder)

The MTSDock executable creates a side or bottom based "AppBar" or a special window that changes the desktop window size. There are some sample 2.20 KBF files that can show off an 800x600 resolution, and respond to portrait or landscape changes (WM_DISPLAYCHANGE message/TypeFile.exe)

This has a timer running (internal ID: IDT_MONITOR) that watches the My-T-Soft keyboard window, and reacts if the window is sized or moved - this resets the appbar to go to the side or bottom & size to the current keyboard size.

The code was modified from Microsoft sample code that created a more customizable appbar, similar to the Windows taskbar - so certain modifications were done to limit what could be done with this implementation of the AppBar.

Comments:

See README.TXT and MAIN.C for specifics and changes from MTSAppBar.

Usage Notes:

Expects My-T-Soft 2.20 to be running before it is launched (otherwise messagebox shown). Option to add a "WAIT" command line option to run in a "Sleep" (pause) loop to wait for My-T-Soft to open & paint (in case both are started at the same time (i.e. Windows StartUp Group). Run in a shortcut with "...\MTSDock.exe" WAIT

The code was built & tested in Microsoft Visual Studio 6 with Windows Vista.

AppBar example Copyright Microsoft Corporation.

Developers Kit Version 1.79 - 8/14/2012

Control My-T-Soft Button Utility

FOLDER: CNTRLMTS

TYPE: Utility / Stand-alone Executable

SOURCE: NONE

LANGUAGE: C / Windows API

IDE: N/A

OVERVIEW

The CNTRLMTS (Control & Select My-T-Soft) utility allows a selection of 1-4 buttons that float in the caption bar, and allow a selection of different configurations of My-T-Soft / My-T-Pen / My-T-Touch.

The default settings in CNTRLMTS.INI assume that there are saved configurations in the installation directory.

This little utility is meant primarily as a front-end for the other developer kit utilities, and a method for the user/operator of selecting other previously saved configurations.

Review Developer's Kit Section for operation of specific DevKit utilities...

Note that by default ([Options], Operator=0) a right-click will open a menu that allows modification of the utility. Use Operator=1 to disable this default action.

If you try and turn off the last button, all buttons will be made visible.

CNTRLMTS.INI

[Buttons]

Keyboard=1

Edit=1

Numeric=1

Macro=1

The [Buttons] section defines which "buttons" are displayed

Use 1 to have the button displayed, 0 to hide

[Options]

Operator=0

Operator=0 enables right-click menu

Operator=1 disables right-click menu, and essentially locks out any configuration via the operator - direct editing of this INI is then required.

Editor=NOTEPAD.EXE

This allows a different selection for utility to edit the INI file when selected via the right-click menu

[Keyboard]

Program=C:\WINDOWS\MYTTOUCH\FWCTLMTS C:\WINDOWS\MYTTOUCH\DEV1.CFG

[Edit]

Program=C:\WINDOWS\MYTTOUCH\FWCTLMTS C:\WINDOWS\MYTTOUCH\DEV5.CFG

[Numeric]

Program=C:\WINDOWS\MYTTOUCH\FWCTLMTS C:\WINDOWS\MYTTOUCH\NUM.CFG

[Macro]

Program=C:\WINDOWS\MYTTOUCH\FWCTLMTS C:\WINDOWS\MYTTOUCH\MACRO2.CFG

Each particular button may have its own executable program / command line options. Use CPYCNMTS for immediate change over, FWCTLMTS to toggle on-screen / off-screen display with different configuration.

LICENSE INFORMATION & DISCLAIMER

Innovation Management Group, Inc. (IMG) provides these utilities as aids to individuals and companies configuring systems for use with other licensed products of IMG. For any other use, licensing fees must be arranged for directly with IMG. The use of these utilities is solely the responsibility of the user. IMG provides these utilities for the use of its customers, and ALL issues of system security must be managed by the end-user of these utilities.

Version 1.00

August 1, 1999

Logos for Custom Logo Option

FOLDER: LOGOS

TYPE: Bitmap images

SOURCE: NONE

LANGUAGE: N/A

IDE: N/A

This information briefly describes the steps required to implement your custom logo on the My-T-Touch / My-T-Pen / My-T-Soft products.

Overview:

The LOGOS.ZIP file has been provided for customers that use the standard retail product for controlled applications. For the custom logo to work in the most effective manner, the Supervisor / Operator mode should be Enabled (Setup | Configuration | Special), with Operator Minimize disabled. This hides the configuration / Tool Bar buttons from the operator, displays a suitable logo, and allows for a final implementation that reduces questions & confusion.

1.77 Note

The LOGOSMIN.ZIP file has been added that includes a minimize image on the lower part of the bitmaps. This would apply if you are in Operator mode, but you still enable the Minimize button capability, so the operator can minimize the keyboard. By using these images, the custom logo image does not need additional modification to make the display match the operation.

Basic Steps:

For these steps, My-T-Touch is used as an example, but the sames steps apply to My-T-Pen, My-T-Soft, etc.

1) Configure My-T-Touch to the size & Panel configuration required for the end user. Position where required on the screen.

2) From the My-T-Touch Menu (Middle button / Tool Bar), select Setting | Save Current Settings, Position | Save Current Position.

3) Close My-T-Touch.

4) Go into My-T-Touch Setup, select Configuration, then Special.

5) Check "Enable Operator Security" On. Check "Disable Operator Minimize" On. Check "Use Custom Logo" On. OK & Close Setup.

6) Copy the LOGO??.BMP files to the installation directory (e.g. C:\WINDOWS\MYTTOUCH). You will want to modify the bitmaps to include your logo. The button shading is included on the examples, but the entire bitmap image is at your disposal. Only 16-color bitmaps are guaranteed to work, but 256-color bitmaps worked in limited testing. (If you are only using 1 or 2 sizes, only those bitmaps need to be modified e.g. LOGO09.BMP and LOGO10.BMP).

That's it! From now on, when My-T-Touch is run, the custom logo will appear over the Tool Bar, and the buttons on the Tool Bar will no longer work. These steps should only be done as the final, final steps before implementation, as the lack of configuration control can be annoying during the development stages.

Notes:

Name, Width in pixels, Height in pixels

LOGO01.BMP, 12, 46

LOGO02.BMP, 15, 58

LOGO03.BMP, 18, 70

LOGO04.BMP, 21, 82

LOGO05.BMP, 24, 94

LOGO06.BMP, 27, 106

LOGO07.BMP, 30, 118

LOGO08.BMP, 33, 130

LOGO09.BMP, 36, 142

LOGO10.BMP, 47, 190

LOGO11.BMP, 62, 250

LOGO12.BMP, 83, 322

If the required bitmap is not available, a black rectangle is displayed over the tool bar if CustomLogo=1 is set in the INI file.

This option is not really meant for "on-the-fly" changes in Setup. It is suggested that all configuration & testing be done and the My-T-Soft software closed prior to setting the Custom Logo on.

If the user clicks the Custom logo, "flickers" of the original button can occur on some displays. This does not affect the overriding operation of the "Operator" settings in Setup.

Without the Operator Security on, the "Tool Bar" buttons can be customized using this Custom Logo feature, however the full Up-Down button display will not appear to the user, since only 1 bitmap is in use for all three buttons. For example, if you leave minimize available, then the top 2/3 of the bitmap can be used for the logo, with the bottom 1/3 being used as an indication for the operator of the minimize capability.

For higher-color depth bitmaps, the current resolution & palette issues come into play. It is suggested (again) that only 16-color bitmaps be used.

In Windows 3.1, resizing the software when the Custom Logo is in use can cause display problems. It is strongly recommended that only 1 size be used when using the Custom Logo in Windows 3.1.

Bitmap 12 is stretched for larger sizes.

July 7, 2003

Developer utilities

FOLDER: DEVUTIL

TYPE: Utilities / Stand-alone Executables

SOURCE: N/A

LANGUAGE: C / Windows API

IDE: N/A

These are a few utilities that were asked for to accomplish developer based tasks.

Display Window information for Window being pointed at

Name: Show Info

Program name: SHOWINFO.EXE

Program Arguments: NONE

Requires: Nothing

Description: A topmost window that displays the Window Name/Class/Handle along with the same info for the parent window. Basically a view into various API calls for determining a Window's class, along with other helpful info. There is a Shell Tray Icon for manipulating the window, including update delay, etc. You may close the window from the icon.

Stress Tester (script manager for DevKit utilities)

Name: Stress Tester / script manager

Program name: STRESS.EXE

Program Arguments: NONE

Requires: My-T-Soft must be running, STRESS.TXT file must be in same location as STRESS.EXE

Description: Reads STRESS.TXT and processes text file line by line. The logic requires Stress to be Enabled (right-click menu from Shell icon), and My-T-Soft or My-T-Touch or My-T-Pen to be running (and open). Comments can be included (or lines temporarily removed by inserting a semicolon as the first character in the line). Lines in STRESS.TXT all affect timing cycle, so comments also effectively induce pauses. There is also a full cycle pause between the end of the script and starting over (to allow closing or disabling Stress Tester).

Notes: Settings in registry at:

HKEY_CURRENT_USER\Software\Innovation Management Group, Inc.\Stress\Settings

Make sure keyboard software is not minimized.

Set delay and make sure enabled.

Originally developed to run through CONFGMTS cycle to test for memory leak verified bitmap memory leak with 3D keys.

Build-A-Board KBF Dump (Extract)

FOLDER: KBFDUMP

TYPE: Utility / Stand-alone executable

SOURCE: N/A

LANGUAGE: C / Windows API

IDE: N/A

KBFDUMP.EXE is a utility that can extract the data from the KBF file, and optionally, extract the data and re-create source files, usable by the Build-A-Board builder. This is for the 2.10 KBF files created by Build-A-Board 2.10 only - it will not operate with KBF files prior to 2.00, or later than 2.10.

KBFDUMP.EXE Usage Notes

Usage: KBFDUMP [KBFFileName]

(If no file name specified, KEYBOARD.KBF is used)

To extract SOURCE projects from KBF files, use "-extract"

Usage: KBFDUMP -extract [KBFFileName]

(If no file name specified, KEYBOARD.KBF is used)

Extracted project files will be saved to ExtractedKBFProject[#]

These source files will be located in Build-A-Board\Source folder

Instructions

Copy the KBFDUMP.EXE utility, and place in the \Program Files\Build-A-Board\Bin folder

From the command-line (CMD project), in \Program Files\Build-A-Board\Bin, run KBFDUMP on the existing KBF file you have, e.g. KBFDUMP -extract "\My Documents\KEYBOARD.KBF"

Now go into the Build-A-Board Builder

Use File | Open, and select and open ExtractedKBFProject

From the opened ExtractedKBFProject, use File | Save As and select a new name for the project (Example: MyLostProjectt)

Use File | Close to close (& save) the project

Now copy the \Program Files\Build-A-Board\Source\MyLostProject\MyLostProject.zip to a safe place, and repeat "I will always backup my source files" as many times as necessary

Notes

By default, the utility also dumps to the screen the data values of the KBF file. You can use output redirection to save to a file, e.g. KBFDUMP > KBF.TXT

If an ExtractedKBFProject folder already exists, a new ExtractedKBFProject1 folder will be created, then ExtractedKBFProject2, ExtractedKBFProject3, etc.

Example

The KBF structure is fairly well documented here - this is an example of a one-key, one panel, one window layout from Build-A-Board 2.10 - in general the actual dump text isn't that useful, where the -extract option which recreates the source files has been requested numerous times.

==========

KBF HEADER

==========

BYTE Sig[2] = M*

BYTE Ver[3] = 210

DWORD Startoffset = 94 | 0x005E

BYTE Level = 1 | 0x0001

======================

KBF Files/Offsets 0

======================

BYTE FileName[13] = MWF00001.MWF

DWORD Startoffset = 0 | 0x0000

DWORD Length = 109 | 0x006D

======================

KBF Files/Offsets 1

```
======================
BYTE FileName[13] = MPF00001.MPF
DWORD Startoffset = 109 | 0x006D
DWORD Length = 37 | 0x0025
======================
KBF Files/Offsets 2
======================
BYTE FileName[13] = MBF00001.MBF
DWORD Startoffset = 146 | 0x0092
DWORD Length = 25 | 0x0019
======================
KBF Files/Offsets 3
======================
BYTE FileName[13] = KEY00000.KEY
DWORD Startoffset = 171 | 0x00AB
DWORD Length = 71 | 0x0047
==================
MWF structure
==================
BYTE Sig[2] = MW
BYTE Version = 1 | 0x0001
BYTE Level = 1 | 0x0001
DWORD PanelStructure = 1 | 0x0001
BYTE Measure = 1 | 0x0001
WORD Width = 640 | 0x0280
WORD Height = 488 | 0x01E8
WORD PosX = 200 | 0x00C8
WORD PosY = 200 | 0x00C8
BYTE Position = 1 | 0x0001
BYTE Frame = 4 | 0x0004
BYTE Background = 7 | 0x0007
BYTE Highlight = 15 | 0x000F
BYTE Shadow = 8 | 0x0008
BYTE Border = 1 | 0x0001
BYTE BorderColor = 0 | 0x0000
```

BYTE FontFaceName =

BYTE FontHeight = 0 | 0x0000

BYTE FontWidth = 0 | 0x0000

WORD FontWeight = 0 | 0x0000

DWORD dwStyle = 3 | 0x0003

==========

Panels List

==========

MPF00001.MPF

==========

MPF structure

==========

BYTE Sig[2] = MP

BYTE Version = 1 | 0x0001

BYTE Level = 1 | 0x0001

BYTE cMBFFILE[13] = MBF00001.MBF

BYTE Measure = 1 | 0x0001

WORD Top = 0 | 0x0000

WORD Left = 0 | 0x0000

WORD Width = 640 | 0x0280

WORD Height = 488 | 0x01E8

BYTE Position = 1 | 0x0001

BYTE Frame = 3 | 0x0003

BYTE Background = 1 | 0x0001

BYTE Highlight = 15 | 0x000F

BYTE Shadow = 8 | 0x0008

BYTE Border = 1 | 0x0001

BYTE BorderColor = 0 | 0x0000

UINT Keys = 0 | 0x0000

==========

MBF structure

==========

BYTE Sig[2] = MB

BYTE Version = 1 | 0x0001

BYTE Level = 1 | 0x0001

DWORD KeyStructures = 1 | 0x0001

==========

KEYS - KEY 0 (1 of 1)

==========

KEY00000.KEY

WORD X = 118 | 0x0076

WORD Y = 100 | 0x0064

==========

KEY Structure - KEY 0

==========

BYTE Sig[2] = MK

BYTE Version = 1 | 0x0001

BYTE Level = 1 | 0x0001

BYTE bDown = 0 | 0x0000

BYTE Measure = 0 | 0x0000

BYTE Width = 410 | 0x019A

BYTE Height = 290 | 0x0122

BYTE BtnType = 2 | 0x0002

BYTE BtnLabel = KeySymbol

BYTE BtnShift = ShiftSymbol

BYTE BtnAction = keyaction

BYTE BtnFace = 7 | 0x0007

BYTE BtnText = 0 | 0x0000

BYTE BtnHigh = 15 | 0x000F

BYTE BtnShad = 8 | 0x0008

Version 1.78 Release 2 - 6/5/2007

Themes with Key Images and PaintDLL Source

FOLDER: THEMES

TYPE: DLL with Source / images

SOURCE: INCLUDED

LANGUAGE: C

IDE: Microsoft Visual C++ 6

To allow different views of the keyboard, the PaintDLL can be used. In order to show off this capability, Themes were introduced in 1.79, using pre-defined looks (key images and coding in a specific PaintDLL). This is the folder that contains the source images and code for the predefined Themes released with the software. For further information on PaintDLL capabilities, refer to the PaintDLL section.

Each folder is named and listed back in the Initialization file for the software - see Theme, ThemeList, ThemesFolder. The base folder has a THEME.INI and a Release folder with a built PaintDLL.DLL interface. When a Theme is changed or selected, the THEME.INI file is referenced, and settings in this file are then used to update the Initialization file of the running software, which then becomes the current PaintDLL in use. The special RESET folder clears the Theme and defaults to the current default configured display.

Note the THEMES folder in the IMG Developer's Kit has the source for the PaintDLLs for each Theme. The Themes included in the release software only contain the THEME.INI and PaintDLL.DLL. Because the 101 keyboard have a large return key, these themes may be better on the 104 key layouts - the 101 keys layouts splits the return key, and uses the carriage return symbol on the upper "key".

To recap the THEMES folders, the THEMES folder installed / included with the software only contains the DLL and details necessary to allow operation within the software, and the THEMES folder here in the IMG Developer's Kit contains the source to build the DLL, and has the specific code to create the matching theme. The method used to create the installed THEMES.zip (for the as installed THEMES folder) was to test and finalize with the Developer's Kit THEMES folder, then run the cleanup.bat (which strips out everything but the necessary DLL/THEME.INI).

Themes can easily be added or removed, but do require some manual handling as based on the settings in the INI and the appropriate THEMES sub-folder(s). Refer to the existing samples and settings, along with notes on the INI settings Theme, ThemeList, and ThemesFolder.

Version 1.79 - September 5, 2012

MultiTouchDLL DLL Source Code

FOLDER: MultiTouchDLL

TYPE: DLL with Source

SOURCE: INCLUDED

LANGUAGE: C

IDE: Microsoft Visual C++ 6

This is an external DLL that handles Multi-Touch, Gesture and Flicks events and messages, and triggers reactions of the My-T-Soft keyboard window. Although some parts are tightly integrated with the My-T-Soft software, by having the source available for modification, recompilation, and replacement, new event handling or different interactions can be accomplished.

The Developer will want to reference and be familiar with the underlying Windows APIs and concepts utilized in the DLL before making any detailed changes.

Version 1.79 - September 5, 2012

Zip / Unzip DLL Source Code

FOLDER: ZIPUNZIP

TYPE: DLL with Source

SOURCE: INCLUDED

LANGUAGE: C

IDE: Microsoft Visual C++ 6 / Visual Studio 2005

This contains the original ZIP30 and UNZIP60 Source Zips from Info-Zip (http://www.info-zip.org). This is the source for the ZIP32.DLL and UNZIP32S.DLL used by the software in various places (Install/Settings/etc.) Please refer to the LICENSE information in the zip files for details. It is included in the IMG Developer's Kit as a convenient place to save/reference this third-party source.

Version 1.79 - September 5, 2012

Chapter 9. User Utilities

User utilities

FOLDER: USERUTIL

TYPE: Utilities / Stand-alone Executables

SOURCE: N/A

LANGUAGE: C / Windows API

IDE: N/A

Turn Away from the Screen for Head-Mouse/Eye-Mouse users

Name: TurnAway

Program name: TURNAWAY.EXE

Program Arguments: NONE

Requires: Nothing

Description: Displays empty screen with single-button, asked for by a Head-mouse user.

Also may be useful for eye-mouse or other assistive / specialty pointing devices. Prevents spurious clicks from being sent to the system when the user may turn away from the screen. Various options are available with a Right-Click / Context menu.

EditSync

- physical keyboard monitor (OnScreen)

- auto-positioning based on text caret

Name: Edit Sync Monitor

Program name: EDITSYNC.EXE

Program Arguments: NONE

Requires: ESLIB.DLL (same folder)

Description: A dual-purpose utility that monitors the current input (edit) state. With OnScreen, the physical keyboard monitor watches keystrokes from the physical keyboard and info is sent to synchronize the WordComplete word completion candidates within OnScreen. This functionality was originally in My-T-Soft AT, asked for by a customer with an upgrade to OnScreen 1.70.

In the second mode, the utility monitors events within applications that support Microsoft Active Accessibility and synchronizes the keyboard window to the position of the input text caret (text cursor). There are 2 options within this mode: 1 monitors the Y position of the caret, and makes sure the keyboard window is on the opposite side of the screen (vertically) of the input text caret. The second option keeps the keyboard window at a fixed offset from the input text caret when the caret is visible - if not visible, the keyboard window is moved off-screen (keyboard not visible when text caret not visible).

Notes: The caret is sometimes a system resource, and other times an application resource.

To support the Accessibility API monitoring, the program(s) being used must support the Active Accessibility events. In testing in Windows 98, various programs act inconsistently. Some hide the caret when moving, causing a lot of keyboard display activity. Other programs don't support the Active Accessibility events. Try Notepad for intended operation. Outside of the menu options available by right-clicking on the icon in the tray (Shell icon), some options are only available by modifying the registry see below for documentation of all the registry settings.

EditSync modifies the OnScreen INI settings to set KeyboardEnabled=1, to enable the coding internal to OnScreen for handling the messages from EditSync.

OnScreen Usage with EditSync:

To enable the physical keyboard monitoring, simply run EditSync.exe. An icon will appear in the tray (next to clock) - you can click on the icon to see the menu - verify "Physical Key Monitor" is checked on. OnScreen will populate the WordComplete panel with word completion candidates as keys are typed on the physical keyboard.

There will also be small numbers (upper-left of WordComplete buttons) shown to indicate which number to use to select the word from the Numeric keypad panel. For example, to select the 3rd WordComplete word, type the number 3 on the numeric keyboard panel (far right of physical keyboard). OnScreen will do an automatic backspace to remove the 3, then complete the word.

These are the methods for selecting the WordComplete words from the physical keyboard:

(For the Ctrl key combos, the numbers must be from the upper row on the keyboard)

Word 1 = Ctrl-1, or Ctrl-6, or Num panel 1

Word 2 = Ctrl-2, or Ctrl-7, or Num panel 2

Word 3 = Ctrl-3, or Ctrl-8, or Num panel 3

Word 4 = Ctrl-4, or Ctrl-9, or Num panel 4

Word 5 = Ctrl-5, or Ctrl-0, or Num panel 5

Settings: The following documents the various settings and where available for modification.

Registry Location:

HKEY_CURRENT_USER\Software\Innovation Management Group, Inc.\EditSync\Settings

String Values:

"StartCount"="2"

This is a scale factor for the delay (in 50ms) Available in the menu, and it should only be modified via the Delay Menu.

"Mode"="1"

1 = Watch Text Caret, 2 = Watch physical keys

"MenuCount"="40"

Not used in EditSync

"DisplayWindow"="0"

Not used in EditSync

"SplitScreen"="0"

When SplitScreen=1, the position of the caret is monitored against CenterScreenY pixel position. This should be only be modified from the menu.

"VisibleWithCaret"="1"

When VisibleWithCaret=1, the approach to be visible with the caret is used. When on, the OffsetX and OffsetY settings are used to position the visible keyboard. This should be only be modified from the menu.

"CenterScreenY"="400"

This decides the cutoff point for the flip between the upper part of the screen or the lower part of the screen when SplitScreen=1. This can only be modified in the registry (use RegEdit or RegEdt32).

"OffsetX"="40"

"OffsetY"="40"

These settings define the offset from the location of the caret when visible, and VisibleWithCaret=1. Positive X means to the left, negative X means to the right. Positive Y means down, and negative Y means up. These can only be modified in the registry (use RegEdit or RegEdt32).

Version 1.73 - January 9, 2002

Chapter 10. System Utilities

NT Utilities

FOLDER: NTUTILS

TYPE: Utilities / Stand-alone Executable

SOURCE: NONE

LANGUAGE: C / Windows API

IDE: N/A

Windows NT AutoLogon - Utilities for Windows NT

Information and other utilities for Windows NT Automatic control

NTAUTO.EXE - Assistance to enable Auto-Logon

NTREBOOT.EXE - Windows NT Reboot System Utility

NTRSTART.EXE - Windows NT Restart

NTSHUTDN.EXE - Windows NT Shut Down system

NTLOGON.EXE - Windows NT Logon as another user

Description

NTAUTO:

This is a utility that provides a front-end to the System Registry in Windows NT to allow an administrator a simple way to enter the Name & Password of a user who will "Auto Logon" into Windows NT.

This Logon will by-pass the Ctrl-Alt-Del / forced logon required typically by a Windows NT Workstation. Because of security issues, the use of this utility and management of the system and its users is entirely in the hands of the System Administrator.

NTREBOOT:

This is a utility that will reboot the system in Windows NT for external control from an application or other process that can initiate this executable file.

NTRSTART:

This is a utility that will restart Windows NT. Available for external control from an application or other process that can initiate this executable file.

NTSHUTDN:

This is a utility that will shut down / power down the system in Windows NT. Available for external control from an application or other process that can initiate this executable file. Useful as an Icon on the desktop to speed system shut down.

NTLOGON:

This is a utility that will logon to Windows NT as another user. Available for external control from an application or other process that can initiate this executable file.

Disclaimer

Innovation Management Group, Inc. (IMG) provides these utilities as aids to individuals and companies configuring Windows NT systems for use with other licensed products of IMG. For any other use, licensing fees must be arranged for directly with IMG. The use of these utilities is solely the responsibility of the user. IMG provides these utilities for the use of its customers, and ALL issues of system security must be managed by the end-user of these utilities.

Usage

To run these utilities, make sure the files are located in an appropriate directory. The files are contained in a self-extracting compressed executable file (NT_AUTO.EXE). Be sure to run this Executable file within the directory where you want the utility placed. The WINNT system directory or another Utility folder is preferred.

Using Explorer or the Run option of the Start Menu, run NTAUTO.EXE.

NTAUTO

Be Sure You Have Administrator Rights!

For proper operation, you must be logged on with Administrator Rights. If the Text fields are empty, then there is a strong probability that you do not have sufficient rights, and using this utility will accomplish nothing. Also, you may run into other troubles if you do not know the password for the Administrator (or other user with Administrator rights).

Be Sure You Have Administrator Rights!

When launched, you will see a dialog that has a check-box, and 2 text fields.

To Enable the Auto Logon capability, make sure there is a checkmark in the Enable Auto Logon.

To Disable the Auto Logon, clear the Enable Auto Logon check-box.

When Enabled, you have the option of entering a Default Logon Name, and the corresponding password for the Logon Name. Refer to the User Manager within the Administrative tools for setting up an appropriate user and a password. Issues of System Security are paramount, due to the lack of ANY security when the opening Ctrl-Alt-Del Logon / Password process is bypassed! (See Notes on the User Manager below)

Once the settings are entered, simply click OK to save the entries into the Registry, and close the Windows NT AutoLogon utility. Click on CANCEL at any time to stop the utility.

If you click OK, you will be notified appropriately and given the option of restarting Windows NT.

Troubleshooting

If something goes wrong with the Auto Logon Name & Password, note that you may Hold the Shift Key down to enable the default Ctrl-Alt-Del Logon process.

The following settings are managed by this utility:

HKLM = HKEY_LOCAL_MACHINE

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\AutoAdminLogon

Set to "1" for enable, "0" for disable

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\DefaultUserName

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\DefaultPassword

Set to appropriate User Name & Password for Auto Logon

Notes on the User Manager

Under the User Properties within the User Manager, the following settings are preferred. Other settings may produce undesirable results!

The _ indicates the setting is cleared, and the X indicates the setting is checked:

_ User Must Change Password at Next Logon

_ User Cannot Change Password

X Password Never Expires

_ Account Disabled

NTREBOOT

Execute to shut down NT and restart the system.

NTRSTART

Execute to shut down NT and restart Windows NT.

NTSHUTDN

Execute to shut down NT. Systems that support power down modes will shut off.

NTLOGON

Execute to logoff of NT and logon as another user.

IMPORTANT NOTES on NTREBOOT, NTRSTART, NTSHUTDN, NTLOGON

Depending on Logon settings, system settings, and system capabilities, these utilities may perform the same function(s) OR not operate.

Version 1.10

January 10, 1998

Mouse Click Utilities

FOLDER: MSECLKS

TYPE: Utilities / Stand-alone Executable

SOURCE: NONE

LANGUAGE: C / Windows API

IDE: N/A

There are 7 executables that generate mouse clicks (single click, double-click, chord) for each mouse button. These were asked for by a customer, and we have included them to enable features & options within other applications, and resolve keyboard focus issues that some applications create.

These are best used with the STCURMTS.EXE - move the cursor to the appropriate X,Y position, and then execute the appropriate mouse click.

MSLFTCLK.EXE - Generates Single Left Mouse Click-Down/Up

MSLFTDBL.EXE - Generates Double Left Mouse Click-Down/Up/Down/Up

MSRGTCLK.EXE - Generates Single Right Mouse Click-Down/Up

MSRGTDBL.EXE - Generates Double Right Mouse Click-Down/Up/Down/Up

MSMIDCLK.EXE - Generates Single Middle Mouse Click-Down/Up

MSMIDDBL.EXE - Generates Double Middle Mouse Click-Down/Up/Down/Up

MSLRCHRD.EXE - Generates Single Left / Single Right Mouse Click-Down/Down/Up/Up

The virtual hardware event will occur where the current mouse cursor is located.

November 1, 1999

Win 3.x Utilities

FOLDER: WIN3UTIL

TYPE: Utilities / Stand-alone Executable

SOURCE: NONE

LANGUAGE: C / Windows API

IDE: N/A

Windows 3.0, 3.1, 3.11, 16-bit utilities

OVERVIEW

These 3 utilities were requested by IMG's customers and are a subset of the NT utilities for 32-bit. These are 16-bit versions that will run under Windows 3.x.

DESCRIPTIONS

File Name: EXITWIN.EXE

Description: Exit Windows

Notes: When run in Windows, this program will force Windows to shut down and return to the DOS prompt.

File Name: RSTRTWIN.EXE

Description: Restart Windows

Notes: When run in Windows, this program will force Windows restart and reset Windows back to when it was first run.

File Name: RSTRTSYS.EXE

Description: Restart System (reboot computer)

Notes: When run in Windows, this program will force Windows to shut down and do a "Cold boot", and reset the entire system (reloading CONFIG.SYS, AUTOEXEC.BAT)

DISCLAIMER

Innovation Management Group, Inc. (IMG) provides these utilities as aids to individuals and companies configuring Window 3.x systems for use with other licensed products of IMG. For any other use, licensing fees must be arranged for directly with IMG. The use of these utilities is solely the responsibility of the user. IMG provides these utilities for the use of its customers, and ALL issues of system security must be managed by the end-user of these utilities.

Version 1.00

August 4, 1998

Part IV. Additional IMG Products

Developer information for other IMG products

Contains information, examples, code, and Developer notes for other IMG products

Chapter 11 - Joystick-To-Mouse developer information and available utilities.

Chapter 12 - The Magnifier developer information, code, and examples.

Chapter 11. Joystick To Mouse Utilities

Joystick-To-Mouse

FOLDER: JTMUTIL

TYPE: Utility / Stand-alone Executable with Source

SOURCE: INCLUDED

LANGUAGE: C++ / Windows API

IDE: Microsoft Visual C++ 6

Joystick-To-Mouse External speed control + Other Utilities

Overview

Joystick-To-Mouse now has predefined messages for controlling the internal speed variables and global acceleration setting, along with basic operation and buttons. This allows high-level development environments some basic control over the functionality of Joystick-To-Mouse.

Various other utilities have been added. These are primarily meant for true developer's, although there are a set of pre-built EXEs that can be used by a computer user that is familiar with the basic concepts of programs, messaging, and scripting.

The following "external" interfaces are available via these utilities:

Set Speed (1-10, 0, and Reset from current configuration)

On, Off, Close (Start, Stop, or Close Joystick-To-Mouse)

Buttons (1-32, 01-10 built as EXEs)

This is bare-bones code that implements the control of Joystick-To-Mouse.

There are pre-built EXEs that can be implemented via a WinExec, CreateProcess, Shell, Spawn, etc. function

Details about the source code are in the source code. This is developer material, and the source code is the explanation / details you need (if you are a developer).

These are the available EXEs with descriptions:

JTMSPD00.EXE - Set speed of 0, center mouse cursor

JTMSPD01.EXE - Set speed of 1, disable acceleration

JTMSPD02.EXE - Set speed of 2, disable acceleration

JTMSPD03.EXE - Set speed of 3, disable acceleration

JTMSPD04.EXE - Set speed of 4, disable acceleration

JTMSPD05.EXE - Set speed of 5, disable acceleration

JTMSPD06.EXE - Set speed of 6, disable acceleration

JTMSPD07.EXE - Set speed of 7, disable acceleration

JTMSPD08.EXE - Set speed of 8, disable acceleration

JTMSPD09.EXE - Set speed of 9, disable acceleration

JTMSPD10.EXE - Set speed of 10, disable acceleration

JTMRESET.EXE - reset from the last set of saved settings for speed and acceleration

JTMON.EXE - turn ON Joystick-To-Mouse if OFF

JTMOFF.EXE - turn OFF Joystick-To-Mouse if ON

JTMCLOSE.EXE - Close Joystick-To-Mouse

JTMBTN01.EXE - trigger button 1 if pressed (Macros not supported)

JTMBTN02.EXE - trigger button 2 if pressed (Macros not supported)

JTMBTN03.EXE - trigger button 3 if pressed (Macros not supported)

JTMBTN04.EXE - trigger button 4 if pressed (Macros not supported)

JTMBTN05.EXE - trigger button 5 if pressed (Macros not supported)

JTMBTN06.EXE - trigger button 6 if pressed (Macros not supported)

JTMBTN07.EXE - trigger button 7 if pressed (Macros not supported)

JTMBTN08.EXE - trigger button 8 if pressed (Macros not supported)

JTMBTN09.EXE - trigger button 9 if pressed (Macros not supported)

JTMBTN10.EXE - trigger button 10 if pressed (Macros not supported)

Implementation

STEP 1

Interfacing with Joystick-To-Mouse from an application.

Decide which method:

INTERNAL (using Windows API Messaging), or

EXTERNAL (run pre-built stand-alone executables via Shell or WinExec type commands).

To use the INTERNAL method, your development environment must support calls such as FindWindow and SendMessage.

To use the EXTERNAL method, you must have the ability to compile C based Windows Source code with a Windows Application as the target.

If the requirements are not clear, then this type of low-level program interfacing is not recommended.

Otherwise reference the JTMSPEED.C and JTMSPEED.RC files for reference & proceed on to STEP 2.

STEP 2 - INTERNAL (Overview)

Obtain the Window handle for the Joystick-To-Mouse window. As seen in the source code, only the Class of "Joystick-To-Mouse" is required for the FindWindow API call. Note that the Window Text (Window Name / Caption Bar Text) can change from "Joystick-To-Mouse" to "Joystick-To-Mouse ON" to "Joystick-To-Mouse OFF" - this can be dangerous to use because of these changes, however, it can also allow your application to sense the current state of Joystick-To-Mouse.

Once you have the window handle, you must then Send or Post a message to JTM. The SendMessage API is preferred because the action will be engaged immediately. All of the options return a TRUE (non-0) value.

Reference JTMSPEED.C for an example and also see the header info below.

If required to use decimal values, note the appropriate values under the HEADER DEFINES.

Also note that working between 16 & 32-bit apps may cause parameter issues.

For reference, the parameters expected within the JTM window function are:

HWND - Handle to JTM - Must be obtained within your app by some mechanism.

MESSAGE - Must be WM_COMMAND, or 273 decimal

WPARAM - Must be the appropriate value (see HEADER DEFINES) (256-267 decimal)

LPARAM - Not used - should be 0 to resolve 16/32 bit issues.

STEP 2 - EXTERNAL (Overview)

Most of the basic EXEs have been compiled for you, only the Buttons 10-32 have not. See Overview for details about each executable. Once these files are built, use the Shell or WinExec to launch the executable when triggered by the appropriate event.

STEP 3 - EXTERNAL APP CONSIDERATIONS

If using the MouseMove event to trigger the communication between your app and JTM, it is recommended to use a global boolean variable to indicate the status of the communication. If this is not implemented, you will end up sending numerous messages to JTM, and this can be especially undesirable if using the external method of communication.

In testing, the following method was used to make a clean interface:

2 Controls were created, a button, with a larger enclosing rectangle (static box) surrounding the button.

Both Controls were configured to have the MouseMove property event trigger communication with JTM, with a Global variable used to indicated which state JTM was in (Fast or Slow). If "Fast", the button would send the "Slow" speed to JTM & set the variable to "Slow". If "Slow", the rectangle would send the RESETSPEED ("Fast" in this example), and change the variable to "Fast".

Other Notes

The RESETSPEED command results in JTM re-reading the INI file for the values set there. If you have the capability of modifying the INI, you can use the SpeedX, SpeedY, and Multiply settings, and then issue the RESETSPEED message.

Multiply is the internal variable for Global acceleration effects. Using any of the SETSPEED? commands results in the Multiply variable to be set to 0, disabling acceleration. If you do not have acceleration enabled, then this means nothing, but if you do, you MUST call RESETSPEED to return JTM to its proper settings.

All SETSPEED? messages set X & Y speeds to the same values, and clears the global acceleration variable.

RESETSPEED message re-reads the values from the INI.

Final note: If you are working within JTM to configure settings, and working with an external app that can affect the internal variable settings, you may get strange results when the INI file is saved/re-read. It is strongly recommended that you stay out of the JTM setting dialogs while using these external messages.

Developers Kit Version 1.78 Release 3 - 9/26/2007

Chapter 12. The Magnifier

The Magnifier

FOLDER: Magnifier

TYPE: Utilities / Stand-alone Executables with Source

SOURCE: INCLUDED

LANGUAGE: C / Windows API / Visual Basic / C#

IDE: Microsoft Visual C++ 6 / Visual Basic / Visual Studio 2005

We have been asked if the Magnifier can be controlled externally for integration with other applications, and with the release of The Magnifier 1.50, some tools and capabilities have been created for developers interested in controlling The Magnifier window / software.

These are the capabilities available:

Open - Launches The Magnifier (up to 7 windowed magnifiers can be running at one time)

Close - Closes The Magnifier

Close All - Closes all instances of The Magnifier

Status - Obtains status from The Magnifier - is it running, is it a window or in full screen mode, are colors inverted or normal, is it visible or hidden.

Set Magnification factor - 1x-9x

Set Magnification point - this can reposition the mouse cursor (normal magnification point), force the point so the mouse/text caret/keyboard focus are NOT referenced, and clear the forced point

Move / Size - As a window, The Magnifier can be repositioned, resized as desired.

Get Position - obtain The Magnifiers current position (screen coordinates)

Get Size - obtain The Magnifiers windows width & height

Invert Colors - toggles inverted colors or normal colors

Show / Hide - toggle The Magnifiers visibility - if visible, hides, if hidden, shows

Full Screen - go to full screen mode

Window - go to windowed mode

Note that most utilities assume only 1 magnifier is running, as they use the API FindWindow, and the first instance / topmost magnifier window will be used.

There 4 main areas of interest:

Magnifier DLL (MAGDLL folder) - This is the core of these utilities. This DLL communicates with The Magnifier window. All other utilities interface with the DLL, which then interfaces with The Magnifier. This is a DLL written in C, has a workspace for Microsoft Visual C++ 6. It is built with exported functions seen by other apps as __stdcall (which is PASCAL or WINAPI).

MagControl (root Magnifier folder) - this is a C based console utility (Command Prompt utility) that can be driven by commands typed at the prompt. Integrating the C code with a C / C++ based utility from this example should be trivial.

Visual Basic - Built in VB 6, this has a single form with buttons that show off all the capabilities of the Magnifier DLL. There is a Module that shows the DLL integration required by Visual Basic.

C# / .NET - Built in Visual Studio 2005 from a sample C# application, a single form shows off all the capabilities of the Magnifier DLL (exactly the same as the Visual Basic example). There is the MagDLL class that shows the Magnifier DLL integration required by the C#/.NET environment.

The following assumes you have The Magnifier installed.

The Visual Basic and C# examples just need you to open the project in the appropriate IDE, and run them - the operation is straightforward and should be obvious - Open The Magnifier (if not already running) and click on the buttons. All the code is included and it should be easy to merge or integrate these capabilities with existing projects for any capable developer.

If you just want to explore these capabilities & dont have an IDE, the MagControl command line utility can be used. Also, this can be integrated within any high-level App (e.g. Microsoft Access, LabView, etc.) that can Shell or launch an external command.

For those that need or want some guidance, here is a quick step-by-step guide to see these capabilities in action.

Step 1 - Make sure The Magnifier is installed, download and unzip the Developers Kit (e.g. Check for Updates from the Magnifier Advanced Menu)

Step 2 - From the Start Menu, run CMD - e.g. in Vista, just type **CMD** and [Enter], or select Start Menu | Run, then type CMD and [Enter]

Step 3 - Change to the Magnifier developers kit folder, e.g. CD \Program Files\The Magnifier\Magnifier

Step 4 - Type these command (in bold) with descriptions as - description here

C:\Program...> **MagControl open[Enter]** - opens the magnifier

C:\Program...> **MagControl fullscreen[Enter]** - goes to full screen

C:\Program...> **MagControl setmag 4[Enter]** - 4x Magnification

C:\Program...> **MagControl setmag 2[Enter]** - 2x Magnification

C:\Program...> **MagControl window[Enter]** - to window mode

C:\Program...> **MagControl movexywh 100 300 400 100[Enter]** - moves window

C:\Program...> **MagControl showhide[Enter]** - hide magnifier

C:\Program...> **MagControl showhide[Enter]** - shows magnifier

C:\Program...> **MagControl invert[Enter]** - inverts color

C:\Program...> **MagControl status[Enter]** - shows magnifier status

C:\Program...> **MagControl currentxy[Enter]** - shows current screen position

C:\Program...> **MagControl currentwh[Enter]** - shows current window width/height

C:\Program...> **MagControl invert[Enter]** - inverts color

C:\Program...> **MagControl close [Enter]** - closes the magnifier

Index