Build-A-Board

The Next Generation of Onscreen Keyboards

Version 2.20 Release 7

User's Guide

Build-A-Board: The Next Generation of Onscreen Keyboards; Version 2.20 Release 7; 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.
Build-A-Board, Version 2.20 Release 7, 6/22/2022

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

Notice of Rights

Trademarks

Notice of liability

ISBN 978-0-557-96952-4

Table of Contents

Part I. Getting Started

General information about this guide, the product, installation, and getting started (how to get Build-A-Board running).

Chapter 1 - Quick Start contains details on the fastest way to install & begin using

Chapter 2 - Getting Started has more information about this guide, Build-A-Board features, Installing / Un-Installing, Licensing Information, Standard Settings, using Build-A-Board Setup, Commonly Asked Questions, and information about Customer Support.

Chapter 1. Quick Start

Install Build-A-Board

There are various ways to obtain IMG software, including (but not limited to):

- Web based / Internet download from IMG's web site (http://www.imgpresents.com/demo.htm)
- Shipped CD / DVD physical media - IMG Product Disc
- Download ISO image file and burn IMG Product Disc CD / DVD
- USB Flash Drive (pre-installed or installed from USB media web download)
- Single file install (from local network / internet / other media)
- IMG on-line account

Note: IMG Software is built by product and major version with minor releases. There is only 1 unique build, but can be packaged / delivered / obtained in various ways. When unlicensed, the software acts as a demonstration / evaluation copy, and operation will be limited in length or run-time or capabilities. Once licensed (typically via License Key and Serial Number, Internet / Account licensing, or OEM / Company-wide / Enterprise licensing), the software will operate with no limitations.

Recommendation: The quickest and easiest method for obtaining IMG software is to download the current version from IMG's web site (http://www.imgpresents.com/demo.htm). Download, then run the file to install. Test, review, play with, and verify the software

meets your needs and requirements. When ready to purchase, go to the IMG License Manager, and click on the "Purchase License Now" button. If working on a machine without internet access, use the Standard single file download, or use a Product Disc option. Separately, purchase the license for the product directly on the website - once you have your License Key and Serial Number, you can install and license the software.

The following outlines a standard retail Product Disc based install. For other options, refer to notes and details available at the source. In general, the approach is to install the software, then license the software. The Product Disc Installation Assistant provides an easy way to accomplish these two tasks.

- In Windows, insert the CD or DVD - the AutoRun feature will load the Installation Assistant - you may Install a licensed product , Install other product demos, or view Release Information. **If you have a Certificate of Authenticity, enter your License Key, Serial Number, and Name to Install and automatically License.**
- If AutoRun is not enabled:
- In Windows, Click on the Start Button
- Select Run
- Select D:SETUP, or type D:SETUP (or E:SETUP if CD/DVD drive E:, etc.)
- Press (ENTER) or click on OK
- In some versions of Windows, you may not have the Run Option - select Computer, your CD/DVD drive, and open Setup
- Answer the questions and follow the

instructions on your screen

> **Note:** You may also Un-Install Build-A-Board by running SETUP.EXE After Build-A-Board has been properly installed. (Build-A-Board Setup will ask you if you wish to Un-Install.) This has been provided as a convenience to the user. It is recommended that you use the Control Panel | Add/Remove Programs Icon to remove Build-A-Board.

Start Build-A-Board

Click on the Start Button, and open the Start Menu.

Select (All) Programs, then Select Build-A-Board. Build-A-Board menu will have selections corresponding to the icons in the group. Select Build-A-Board to begin operation.

Build-A-Board will typically show the Program Files folder after install. Select the Build-A-Board icon to launch/run Build-A-Board. Refer to the on-line help, and the User's guide for features and capabilities of Build-A-Board.

Chapter 2. Getting Started

Build-A-Board User's Guide

Version 2.20 Release 7

6/22/2022

The Next Generation of OnScreen Keyboards

this agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. The purchaser may be allowed to make a back-up copy. No part of this manual or guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchasers personal use, without the express written permission of Innovation Management Group, Inc.

This manual and product represent over 30 years of on-going development, testing, and support. Licensed users of the software are the most important aspect of the entire process that brings this manual and product into existence. Please be respectful of all parties involved.

Trademarks:

Using this Guide

This guide is a comprehensive collection of

details, notes, and information about Build-A-Board. Portions are incorporated within the product itself, and it is also available in various forms (e.g. printed, on-line, PDF, etc.).

Important User Information

This guide is for users who are familiar with Windows, its basic concepts, and how to operate within Windows. If you are not, the information you may need to fully utilize Build-A-Board and this guide may be limited. You may wish to review Windows help, tutorials, and other available information on using and operating Windows before continuing using this guide.

Conventions used within this guide

> **Note:** Notes and other additional information will be indicated in this way

Warning

Special and other important details to pay close attention to will appear this way

What is Build-A-Board?

My-T-Soft® Build-A-Board is a suite of utilities that provide tools to create and operate on-screen keyboards, panels, and buttons. These enhancements allow touchscreen, hand-held, wearable, wireless, ruggedized, tablet and pen-based computer users to operate these systems without the need for a physical keyboard. This results in space saving; reduced

hardware costs; quick & low-cost user-training; improved security; reduced operator errors; cleaner machine interfaces; enabling legacy equipment retrofits; and realization of new & innovative approaches for computer users.

Build-A-Board is the evolving culmination of years of experience working with the needs of end-users and system providers & their use of IMG's successful My-T-Soft® family of on-screen keyboards. Developed to meet the needs of manufacturers, integrators, developers, and end-users of touchscreen & pen-based systems, it is the Ultimate Tool for anyone with a need for virtual on-screen keyboards & keypads.

Why do I need

Build-A-Board?

The on-screen keyboard has become a readily recognized tool for working with touchscreen and pen/stylus based systems. When the need for security, customization, flexibility, or unique and special applications creates a situation where a custom keyboard (or interface) is required, Build-A-Board provides the tools and capabilities necessary to create and deploy a solution on all major computer platforms and operating systems.

Features

- Drag and drop keys with modifiable labels, actions, views
- Multiple target platforms - Windows,

Android, Linux, Mac OS X, etc.

- Create any on screen keypad, keyboard, or membrane layout
- Replace old legacy membrane panels with virtual onscreen panel replicas
- Select Colors of Text, Keys, and Panels
- Use High Resolution 3D key images on keys
- Drag & drop images / add images to keys and panels
- Select Fonts
- Build & Test within the Builder Environment locally (does not require Target system)
- Cut/Copy/Paste Keys
- Align keys - Top/Left/Bottom/Right/Horizontal Center/Vertical Center

- Evenly Space Keys
- Size Keys to match Width/Height/Both Width & Height
- Center Key or Keys
- Create keystrokes along with full-featured macros in Key Action
- Built-in Commands: Close, Minimize, Save Position
- Open different layouts from user-accessible keys or manage programmatically
- Play MIDI files (on supported platforms)
- Play Sounds (Wave files) (on supported platforms)
- Run External programs, Execute Shortcuts, Use File Associations to launch host application
- Save and Manage projects

- Upload and Download projects from your Build-A-Board.com Account
- Integrates with Build-A-Board.com online database of layouts

What You Need

To run Build-A-Board you need the following equipment (hardware requirements):

- IBM 80386 or higher or compatible
- 100 MB hard disk space available
- 1 GB memory or higher
- Windows 11 / 10 / 8.1 / 8 / 7 / Vista
- VGA or SVGA recommended
- Supported Run-Time Target device

Installing / Un-Installing Build-A-Board

There are various ways to obtain IMG software, including (but not limited to):

- Web based / Internet download from IMG's web site (http://www.imgpresents.com/demo.htm)
- Shipped CD / DVD physical media - IMG Product Disc
- Download ISO image file and burn IMG Product Disc CD / DVD
- USB Flash Drive (pre-installed or installed from USB media web download)
- Single file install (from local network / internet / other media)
- IMG on-line account

Note: IMG Software is built by product and major version with minor releases. There is only 1 unique build, but can be packaged / delivered / obtained in various ways. When unlicensed, the software acts as a demonstration / evaluation copy, and operation will be limited in length or run-time or capabilities. Once licensed (typically via License Key and Serial Number, Internet / Account licensing, or OEM / Company-wide / Enterprise licensing), the software will operate with no limitations.

Recommendation: The quickest and easiest method for obtaining IMG software is to download the current version from IMG's web site (http://www.imgpresents.com/demo.htm). Download, then run the file to install. Test, review, play with, and verify the software

meets your needs and requirements. When ready to purchase, go to the IMG License Manager, and click on the "Purchase License Now" button. If working on a machine without internet access, use the Standard single file download, or use a Product Disc option. Separately, purchase the license for the product directly on the website - once you have your License Key and Serial Number, you can install and license the software.

The following outlines a standard retail Product Disc based install. For other options, refer to notes and details available at the source. In general, the approach is to install the software, then license the software. The Product Disc Installation Assistant provides an easy way to accomplish these two tasks.

- In Windows, insert the CD or DVD - the AutoRun feature will load the Installation Assistant - you may Install a licensed product , Install other product demos, or view Release Information. **If you have a Certificate of Authenticity, enter your License Key, Serial Number, and Name to Install and automatically License.**
- If AutoRun is not enabled:
- In Windows, Click on the Start Button
- Select Run
- Select D:SETUP, or type D:SETUP (or E:SETUP if CD / DVD drive E:, etc.)
- Press [Enter] or click on OK
- In some versions of Windows, you may not have the Run Option - select Computer, your CD/DVD drive, and open Setup
- Answer the questions and follow the

instructions on your screen

Note: You may also Un-Install Build-A-Board by running SETUP.EXE After Build-A-Board has been properly installed. (Build-A-Board Setup will ask you if you wish to Un-Install.) This has been provided as a convenience to the user. It is recommended that you use the Control Panel | Add/Remove Programs Icon to remove Build-A-Board.

Starting Build-A-Board

Click on the Start Button, and open the Start Menu. Select (All) Programs, then Select Build-A-Board. Build-A-Board menu will have selections corresponding to the icons in the

group. Select Build-A-Board to begin operation.

The following icons will also be available in Build-A-Board group:

Build-A-Board - runs Build-A-Board

Build-A-Board Help - opens Build-A-Board Help (opens this document)

Keystroke Macro Recorder - opens Keystroke Macro Recorder for use to generate formatted macros from keystrokes

Licensing Information - Displays current license status of Build-A-Board, allows instant licensing

Licensing Information

Build-A-Board uses the IMG License Manager

to manage the licensed use of this product. If unlicensed, the operation will be as an evaluation - you can work with the Builder, but will not be able to build licenseable layouts - you can still build and work with samples, but will not be able to generate licensed Run-Time targets. Evaluation built layouts will operate as limited run-time versions, even on separately licensed run-time targets. In the evaluation (unlicensed) mode, Build-A-Board will upon exit display the license manager (announcing that it is unlicensed). It is recommended that a free account/free license be obtained for almost all Build-A-Board users. For extended evaluation, testing, and special purposes, please contact IMG Customer Service for license options. Once licensed, the operation will not be limited in any way.

Note: The only way to generate licensed Run-Time Targets will be from a licensed

version of Build-A-Board. However, systems running Build-A-Board generated layouts may also license the Run-Time software separately (system license).

The most common methods of licensing are electronic (web/web account/e-mail based) and by certificate (Certificate of Authenticity). With licenses using a registered serial number, a license key will be made available once a license has been purchased - these need to be entered into the IMG License Manager to activate a valid license. For further details, refer to the IMG License Manager. There is also USB storage device licensing, so Build-A-Board can be run from the USB device, simply by inserting the licensed device into any system.

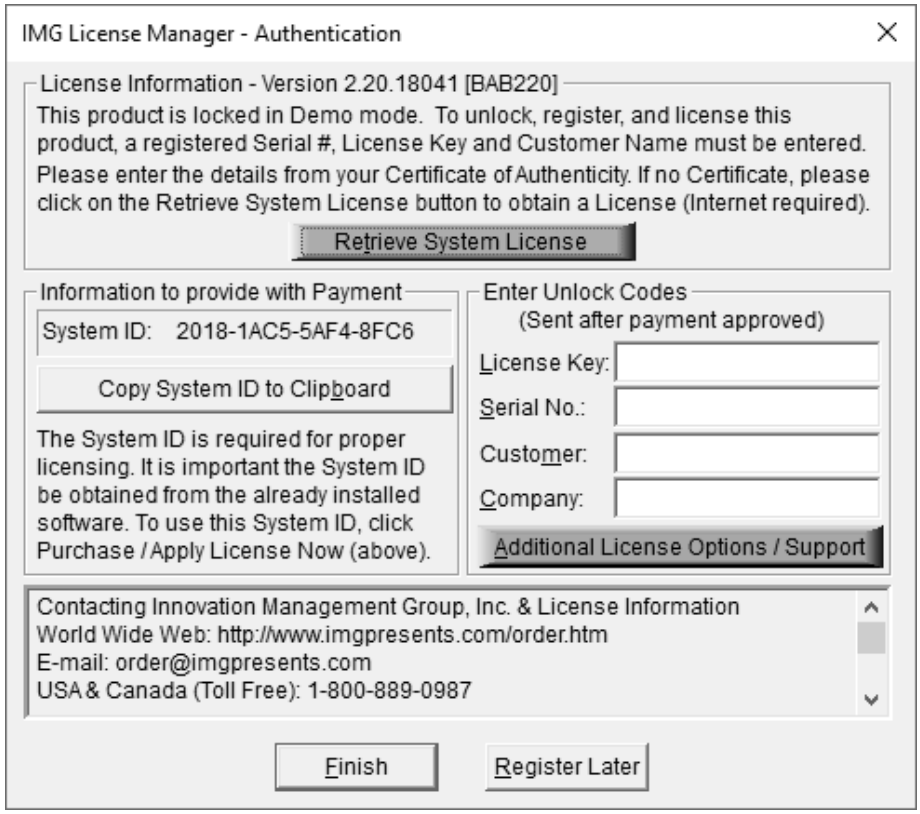
Note: There are numerous license

schemes (including OEM, site, & enterprise licenses) available to meet the needs of all our customers. If you have any licensing questions, please contact Innovation Management Group, Inc. directly.

License Manager

The IMG License Manager will operate with an IMG Personal License (Basic/Standard/Professional). This is a system based license that can license any supported IMG product per platform. Licensing is done through the internet and your Build-A-Board.com account, accessed via the "Retrieve System License" button.

IMG License Manager - Authentication

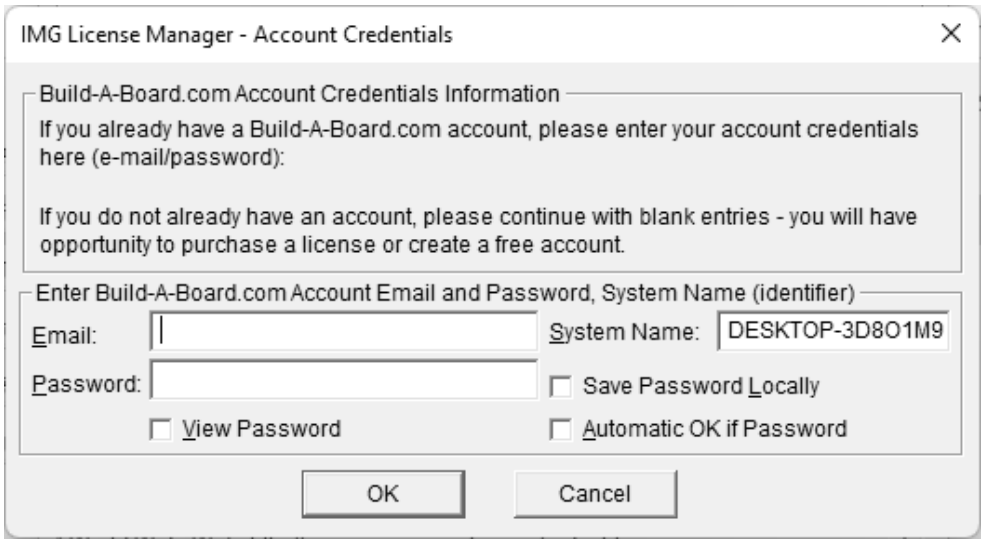


When the "Retrieve System License" button is used (if applicable to the product), you will have the option of using an existing Build-A-Board.com account or selecting the type of License you wish to purchase.

The IMG Personal License

Purchase of a new IMG Personal License creates a Build-A-Board.com account, and the system will be licensed automatically via the "Retrieve System License". If you already have a Build-A-Board.com account, you can Apply a license to your system when you use the "Retrieve System License" button.

IMG License Manager - Account Credentials



After selecting the "Retrieve System License"

button, you will be shown the Account Credentials dialog to enter e-mail and password, as well as several other options. If you do not have an existing Build-A-Board.com account, simply click OK to go to Build-A-Board.com for options on purchasing a license or creating an account.

If already have a Build-A-Board.com account, simply enter your existing account credentials with e-mail and password. By default, the password is obscured - to see in plain text, check on the View Password option. The password is not saved locally, and would need to be re-entered if system is unable to retrieve a valid license. If you check on the Save Password Locally, the entered password will be saved, and used to populate the field. Note the password is saved in an encrypted form (not plain text). If there is a saved password, and you wish to bypass this Account Credentials

dialog, the Automatic OK if Password when checked on will skip the display of the dialog, and automatically retrieve a license based on the e-mail/password credentials provided. For automation options, see below.

Automation Options - ACCOUNT.TXT and AUTOLICENSE

ACCOUNT.TXT

ACCOUNT.TXT is a simple text file that can contain account credentials and options that LICENSE.EXE (IMG License Manager) will read in and update the System Registry. Note for security purposes, if this file exists in the same folder as LICENSE.EXE, the file is read, parsed, updates the registry, and then is deleted. The ACCOUNT.TXT file may be saved with ANSI or UNICODE encoding. This creates a mechanism to bypass the user interface required with the Account Credentials dialog, and can be used to enable one-click

licensing, or to facilitate automatic licensing.

Below are the available entries for ACCOUNT.TXT and structure to enable correct reading and parsing of the file. Note that the password is plain text here, but saved in the registry in an non-plain text format. Also note that ACCOUNT.TXT is deleted once parsed, so the account credentials will be secure once LICENSE.EXE is run. In general, LICENSE.EXE should be run soon after whatever step drops ACCOUNT.TXT into the installation folder. Also, because Administrator privileges are required to copy ACCOUNT.TXT into a Program Files location, and LICENSE.EXE requires the same privileges, treating these as a dual-step process is recommended (i.e. copy in ACCOUNT.TXT, then run LICENSE.EXE). If no user interaction desired when running LICENSE.EXE, the POSTINSTALL command line parameter will

run the IMG License Manager without any user interaction needed - no dialogs, no screen display (e.g. LICENSE.EXE POSTINSTALL[Enter]). For practical purposes, this converts ACCOUNT.TXT entries into System Registry and removes ACCOUNT.TXT. See AUTOLICENSE below for no user interaction with License retrieval from Build-A-Board.com account.

[License Manager]
BABEmail=your.name@example.com
BABPassword=password
BABSystemName=MySystem
BABSavePassword=Yes
BABViewPassword=No
BABAutoOK=Yes

The following is the structure of data saved into the Registry Key as outlined below (example data):

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Inno
agement Group\License Manager]
"BABEmail"="your.name@example.com"
"BABPassword"="J,246,V,195,F,233,E,232,I,239
"BABSystemName"="MySystem"
"BABSavePassword"="Yes"
"BABAutoOK"="Yes"
"BABViewPassword"="No"
```

Note the SavePassword, AutoOK, and ViewPassword can hold values of "Yes" or "No".

AUTOLICENSE

If LICENSE.EXE is ran with AUTOLICENSE added at the command line as a parameter (e.g. LICENSE.EXE AUTOLICENSE[Enter]), the LICENSE.EXE will perform certain tasks, and if the credentials are correct, the options are correct, the Build-A-Board.com Account

Exists and is able to issue licenses for the account, the system will automatically license.

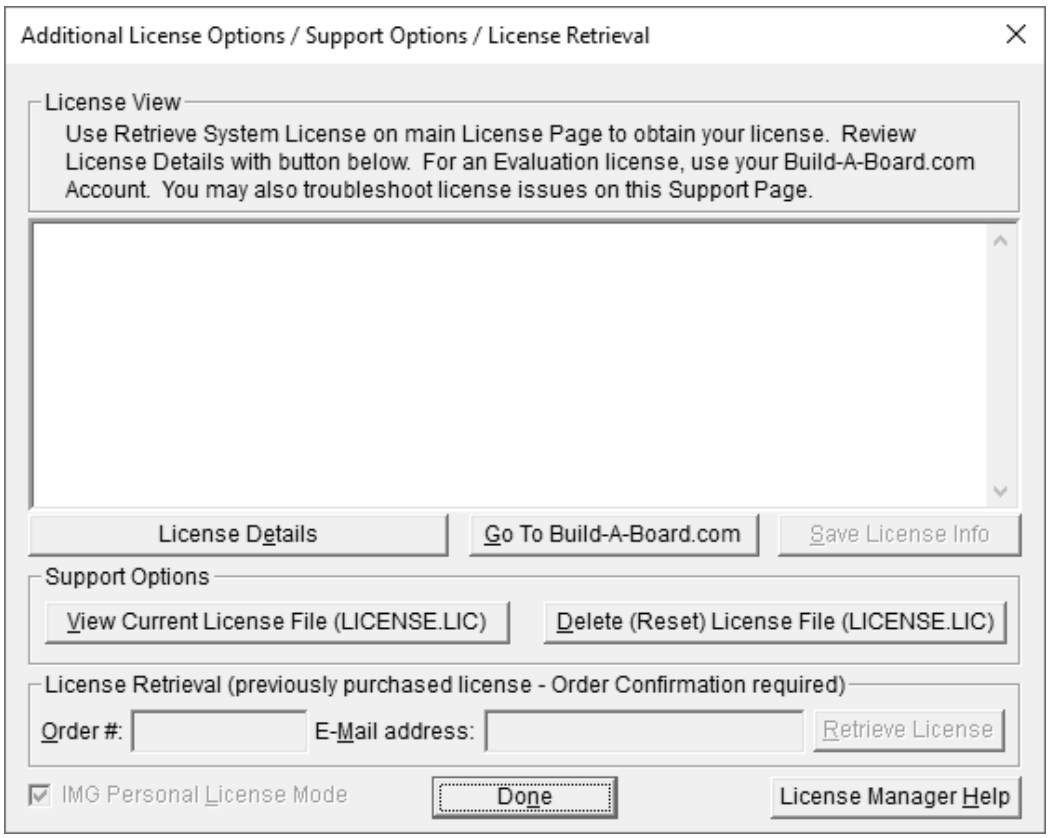
Steps Required to Automatically License Build-A-Board using AUTOLICENSE

1. Build-A-Board.com Account Exists - you can verify this by going to Build-A-Board.com/accounts/ and logging in with e-mail and password (account credentials).
2. Licenses available - once logged in to Build-A-Board.com account, click on License Manager tab to view licenses used and licenses available. Purchase Licenses if needed. Also note that Account Type can limit license issuance - Basic will only issue licenses for IMG Personal Products - Standard or Professional required for Build-A-Board

3. ACCOUNT.TXT - create text file ACCOUNT.TXT with correct credentials and options. As Administrator, drop ACCOUNT.TXT file into \Program Files\Build-A-Board.
4. LICENSE.EXE AUTOLICENSE - As Administrator, run LICENSE.EXE with AUTOLICENSE command line parameter. Note this can be automated via batch file or shell script.

IMG License Manager -

Additional License Options



The IMG Personal License is a system based license, and can be retrieved at the main license page with the "Retrieve System License" button. Most license issues can be dealt with online at your Build-A-Board.com account. There are some advanced and support options that can be managed at the Additional License

Options page.

The "License Details" button will check the System License and report the issue or issues why the license is not valid. In some cases, these details may explain the situation, and point to a corrective action.

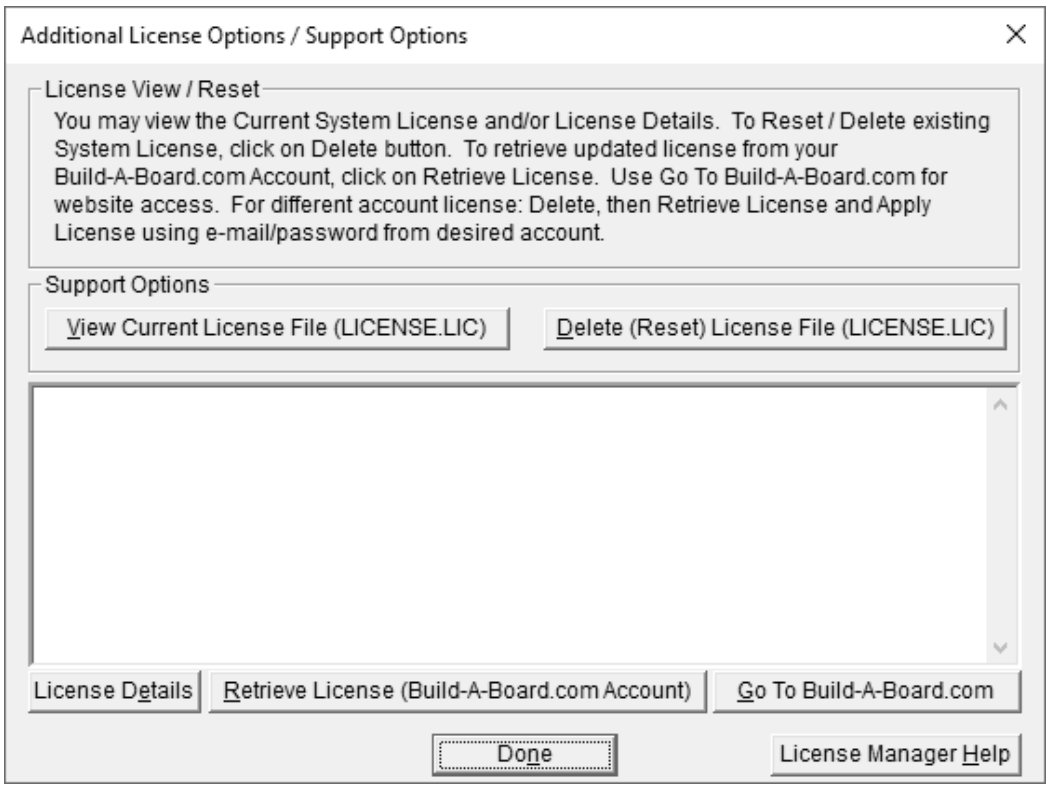
The "Go To Build-A-Board.com" button will open a browser and go directly to the login page for your Build-A-Board.com Account.

Use "View Current License File (LICENSE.LIC)" to read the System License, and display the License details. In some cases, these details may explain the situation, and point to a corrective action.

Use "Delete (Reset) License File (LICENSE.LIC)" to remove the System License and Reset the System ID. This action is not recommended and requires 2 confirmations, because all previous license

information is destroyed, and a new System License will be required.

Additional License Options - Licensed View



The Additional License Options changes slightly when the software is properly licensed. The buttons already described above move positions, and there is an additional button option (Retrieve License).

Use "Retrieve License (Build-A-Board.com

Account)" to reload and obtain the current license from your Build-A-Board.com account. This could be necessary if you've added a platform license, purchased a license while operating on an evaluation license, added a serial number and been advised to renew your license, or other license change that requires a refresh/renew from the current license available tied to the current Build-A-Board.com account. Note that if you need a license from a different Build-A-Board.com account, you must Delete / Reset the license, and Apply the license to the current system with your e-mail/password from the correct Build-A-Board.com account.

IMG License Manager - Licensed Display

The IMG Personal License

When properly Licensed with an IMG Personal License, a screen similar to this will show the License Information for the product. This display indicates that the software is Licensed. For help, support, and other options, you can click on the "Go to My Build-A-Board.com Account" to open a browser and access your account online.

If you single left click on the text display box

at the bottom, it will rotate through 3 different views - License Information, IMG Personal License details for the system, and results from IMG Personal License evaluation.

The second view shows License Details from the System License.

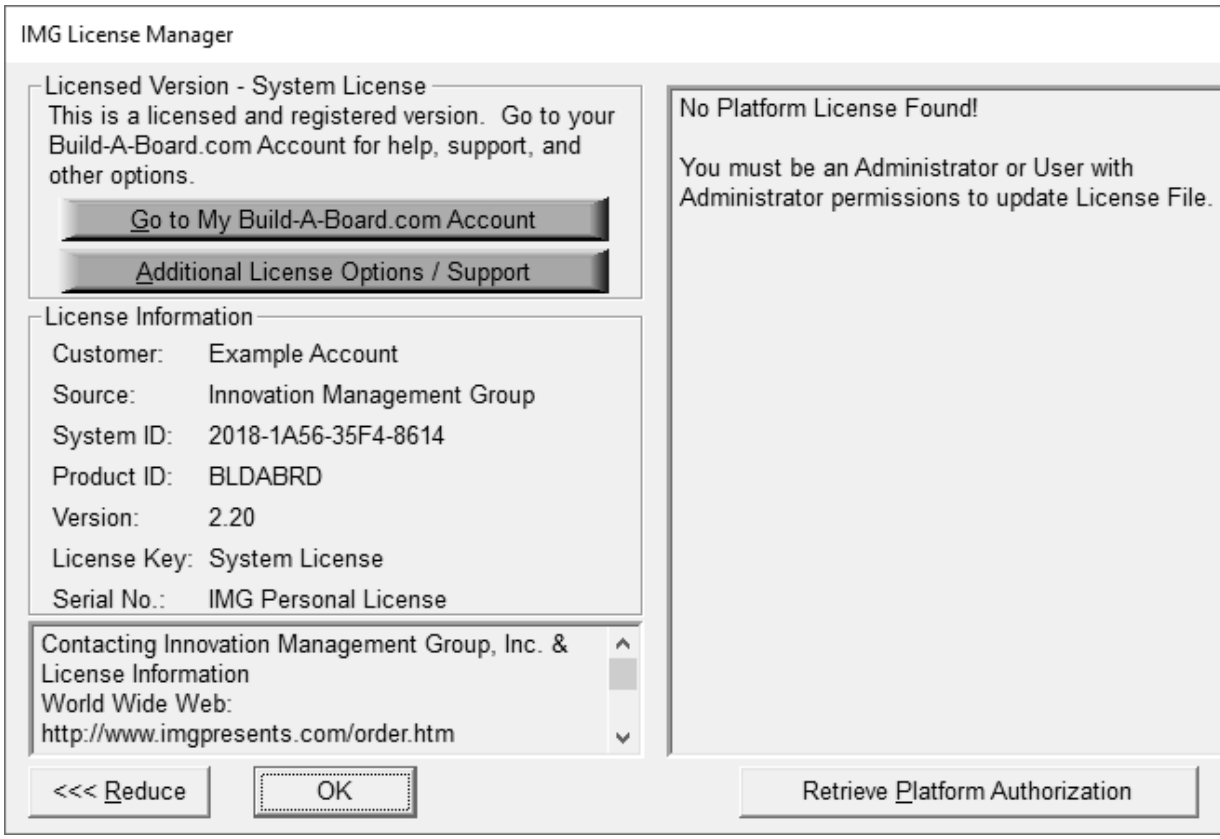
The third view shows details on how the System License was evaluated by the License Manager.

Technical Note: Professional / Support Licenses are licensed per product and the LICENSE.LIC file is located in the installation folder. The IMG Personal License is a System License, and the LICENSE.LIC is located in the Common

Files area of Program Files in the Innovation Management Group License Manager folder.

Build-A-Board Platform Authorization

If you expand the IMG License Manager, you will see the Platform License details. For Free, Personal Licenses, and Professional Licenses, this area will appear as below.

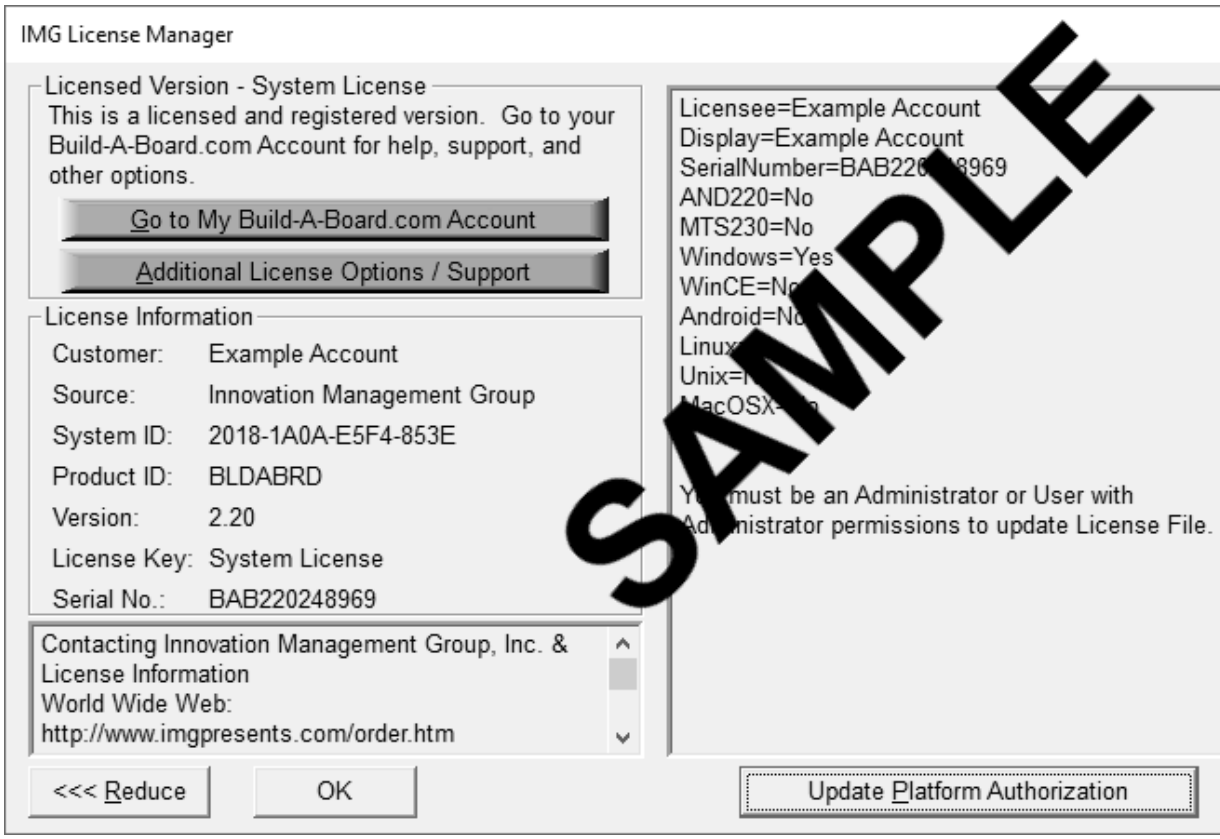


The Platform License provides a mechanism so all Keyboard Layouts (KBF Files) created from a properly licensed Build-A-Board are already licensed when deployed on a licensed platform. These KBF files can then be deployed with no additional license steps on the platforms indicated as licensed (i.e. Windows=Yes)

The main purpose of the platform license is so layout files built from a licensed

Build-A-Board can then be deployed directly on each licensed platform and no other licensing steps are required. KBF Layouts are licensed for operation on different platforms, and this is indicated with the Platform Authorization section of the Licensed Display in Build-A-Board. For each platform that indicates Yes, a built KBF can run on that platform (with the Platform Run-Time Target version of My-T-Soft) and will be licensed for target operation with no other action - i.e. licensed operation for the platform is embedded in the KBF generated from Build-A-Board as indicated in the Platform Authorization display.

> **Note:** To obtain additional platform licenses, please contact IMG Customer Service.



On platforms that are indicated with a No, your built layout will still operate, but the license status will be unlicensed, and the Run-Time target will operate as an evaluation version (demo only). After 100 keystrokes (or as defined by the platform), the Run-Time target program will close (and if available, a DEMO.KBF will be shown). On some platforms, the Run-Time target may be licensed separately (system type license, tied to

the system, allowing KBFs generated from a licensed Build-A-Board to operate, even if not licensed for that particular platform).

In general, as long as you have a proper license for Build-A-Board, and the platforms you wish to generate KBF layouts for are indicated as Yes, everything will work seamlessly. Referring to this Licensed Display will verify the actual license(s) available.

> **Note:** There are numerous license schemes (including OEM, site, & enterprise licenses) available to meet the needs of all our customers. If you have any licensing questions, please contact Innovation Management Group, Inc. directly.

Commonly Asked Questions

Compatibility

1. Does the My-T-Soft Keyboard work with all Windows Applications?

Yes.

2. Does the My-T-Soft Keyboard for Linux work with GNOME, or KDE, or only certain distributions?

The run-time software for Linux is written at the XLib level, which means just about all newer Linux Distributions (and many older distros), running GNOME / KDE / or other similar Window Manager, will have no issues with the run-time My-T-Soft for Linux.

Build-A-Board Operation

3. What is the difference between the builder and the run-time keyboards?

The builder is the development tool that allows you to customize layouts, and choose targets. For actual End-Users that just need to use the keyboard layout, only the run-time software (installed on the target system) is required. Your layout will be a .KBF file (KeyBoard File with a file extension of KBF), and if only one layout is used, the default KEYBOARD.KBF needs to be on the run-time target system. For multiple layouts, each layout's name needs to be available, along with the default opening layout (named KEYBOARD.KBF). Currently the Builder can only run within Windows, while the keyboards (as targets) can run in any supported platform, including Windows, Android, Linux, and Mac OS X.

4. Why would I need/want a platform license?

When working with the builder, layouts built and deployed may contain license information.

The type of license contained in the layout (.KBF file) determines the operation on different platforms (e.g. Windows, Android, Linux, etc.). If there is no license (or a free license) in place when a layout is created/deployed, the run-time target software will need to be licensed to allow normal operation with that layout (.KBF). By obtaining and building layouts with a platform license, operation on specified platforms (e.g. Android) will be licensed and no additional steps/licensing is required for those specified licensed platforms. In other words, layouts built with a platform license can be deployed to any device on that platform and will run in a licensed fashion. For individuals with a few systems, a platform license may not be needed, but if you are building for your company to deploy on dozens or hundreds of devices, a platform license will be highly desirable.

Customer Support

Build-A-Board Software is backed by a support staff trained to provide you with fast, courteous service. Over the years, IMG has astounded individuals at the quality of it software and its support. So that we can continue to focus our resources on development and providing high-quality products & support, we do appreciate your assistance in reviewing the help, manual, and support information available at our website to see if the problem or question has already been addressed. However, if you need assistance beyond what the manual, tutorial, help files, and on-line support database provide, please contact IMG Customer Service:

Innovation Management Group, Inc.
Customer Service
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

To open a Technical Support case and create a support ticket, please refer to https://www.imgpresents.com/orders/support/tech

Please provide, or have the following information ready when you ask for assistance:

- Build-A-Board version number, update level.
- Registered serial number (or if running demo)
- Make and Model of your computer.

- Windows version number, any Service Packs or major updates.
- A description of the problem.
- If possible, a list of the steps required to recreate the problem.
- If you have seen an error code, record and report the number.
- Additional information may be required, such as monitor type, type of pointing device, amount of RAM in your system, other software running (anti-virus, spyware, virtual machine, etc.).

Product Catalog

Innovation Management Group, Inc.'s Products

Commercial Division Products...

Indestructible Keyboards & Indispensable Utilities!

My-T-Soft® Build-A-Board

The Ultimate Tool for creating and modifying On-Screen Keyboards, buttons, and Panels. My-T-Soft Family, plus Cross-Platform Support

My-T-Pen® for Windows

On-Screen Keyboards & Utilities for Pen Based Systems

My-T-Touch® for Windows

On-Screen Keyboards & Utilities for Touchsreens

My-T-Soft® Basic for Windows

Basic On-Screen Keyboards and access to online database of boards. Free Edition, or IMG Personal License.

My-T-Soft® Professional for Windows

On-Screen Keyboards & Utilities for any pointing device Built-in Layouts, Logon Utilities, Developer Tools, and Support.

My-T-Soft® TS for Terminal Services

On-Screen Keyboards for Terminal Server / Terminal Services

My-T-Soft® for Android

Custom layouts and special features for Android based devices

TouchRight Utilities

Right Click Access for Pens & Touchscreens

Assistive Technology

Division Products...

Enabling Tools for Special Needs

AT Accessibility Suite

IMG's Assistive Technology Software with site license options

Joystick-To-Mouse

The Software That Lets You Run Windows With A Joystick!

My-T-Mouse®

The Software That Makes Your Mouse a Mouse That Types! Indispensable & Utilities for any Mouse or Trackball

OnScreen

Special Features for disabled & impaired users Word Prediction / Word Completion / Window Control / Scanning

OnScreen with CrossScanner

Complete control of Windows from a single switch! Support for Keyboard, Mouse, Joystick interfaces

SmartClick

Operate Windows without the need to Press/Click a Button

The Magnifier

Area and Full Screen Magnifier, Cursor Locator, Visual Aids

WordComplete

Word Completion, adaptive word prediction, Word List Management, etc. Type Better, Type Less - we do the rest!

For further information...

Contact your Local Software Dealer, your Hardware Vendor, your Information Technology Department

or

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
http://www.my-t-mouse.com
http://www.my-t-pen.com
http://www.my-t-soft.com
http://www.my-t-touch.com
http://www.onscreen-keyboard.com
http://www.build-a-board.com
http://www.joystick-to-mouse.com
http://www.themagnifier.us

For International Contacts, please see Web Site...

Part II. Build-A-Board Builder

Description of how to operate and configure the Build-A-Board Builder - the management and

design portion of creating and building custom on-screen keyboards and user interface panels.

Description of how to create, manage, and work with Build-A-Board source Projects.

Chapter 3 - Build-A-Board Operation contains general operation information, along with an overview about each section within the Builder.

Chapter 4 - Advanced Builder Information

additional information, configuration options, and advanced notes on Build-A-Board.

Chapter 3. Build-A-Board Operation

Build-A-Board Quick Usage Notes

TERMINOLOGY - the following outlines words and terms used throughout this manual and defines the specific meaning used within the context of Build-A-Board.

- **KEY** = used interchangeably with button, the user-interface control (often displayed as graphical button with up/down states). Pressing (clicking) a key triggers an action

(Key Action).

- **PANEL** = Keys are arranged in groups onto Panels.
- **BOARD** = the actual layout of keys (buttons) in a configuration, as one or more Panels.
- **RUN-TIME TARGET** = this is the device, operating system, or platform where a board will be displayed on and presented to a user for operation (typically as an on-screen keyboard or user-interface component).
- **PROJECTS** = contains the details and definitions to manage and work with boards for one or more run-time targets.
- **KBF** = the single-file output that defines a board for a run-time target. This is the file extension to identify the file, and it is derived from KeyBoard File (.KBF).

- **SOURCE** = the actual SOURCE folder that contains the source files for projects and boards, or used as reference back to the Builder (which can manipulate, edit, and modify the Source of a board).
- **TARGET** = the actual TARGET folder that contains built run-time targets and KBF files for each project. Each project also gets a TEST target that is used during development to show the built board
- **BOARDS** = the BOARDS folder contains built run-time targets and KBF files for use by the My-T-Soft run-time software. Each built project places its KBF into the local system's BOARDS folder.
- **BUILDER** = Application where boards can be created, modified, and KBF Run-Time data files are created. Manages projects (Source files) and Run-Time Targets (Target files). Provides a "rich-application"

environment for Developers and Users working with Board & Key Properties.

Quick Usage Notes

> **Note:** For those users that prefer to jump right in, this is a quick step-by-step approach to build your first board.

1) Select Project (Board)

Select existing sample or existing project, Default Size, or Set Board Size project. Details at Project Selection.

2) Add Keys

Add New Key (Right-click | Add New Button, Edit | Add New Key, Insert key, Add Tool, click & drag on empty board area, drag & drop from Keys window (F8)). For additional

information, see working with the Panel Display.

3) Move & Size Keys

Click & Drag - Key area, moves key; frames, resizes key

Arrow keys - Move Key; with Ctrl & Shift, resizes frames

See Panel Display and Cursor Tools and Menus & Settings.

4) Modify Keys

Key area (Double-click, Right-click | Properties). See Key Properties for details.

5) Build & Execute

Build Board (Build menu | Build, Tool bar | Build, F9 key). Refer to Building Boards.

Execute My-T-Soft with Board layout from with Builder for testing (TARGET TEST

Folder) (Build | Execute, Tool bar | Run, F10 key).

6) Copy to Target System / Deployment / Upload to Build-A-Board.com Account.

Other information at Run-Time Options

Windows Systems:

Run-Time | View Project Run-Time Targets Folder

The appropriate folder contains the run-time My-T-Soft software and Keyboard layouts (e.g. MSWIN, MSWIN32, MSWIN64, etc.). This folder can be copied/transferred to the matching Windows system and the MYTSOFT.EXE can be run directly. On systems with a My-T-Soft family product installed, the KBF file can be copied to the [Shared Documents] BOARDS folder.

Windows CE Systems:

By selecting an ActiveSync Synchronized folder in Run-Time | Select Output (ActiveSync) folder, after a successful Build, My-T-Soft Run-Time files will be copied to the selected folder, and ActiveSync will synchronize with the Target System.

Android Systems:

The Android run-time must be installed/configured on your Android device. By selecting an Output folder in Run-Time | Setup Output (ActiveSync) folder, after a successful Build, My-T-Soft Run-Time files will be copied to the selected folder, and the layout files can be transferred to your Android device.

Linux/Unix Systems:

You must install the appropriate build onto the target system, then copy required KBF files to the target system (folder with mytsoft

executable). Use Run-Time | View Project Run-Time Targets folder to view/access Target KBFs from built project(s).

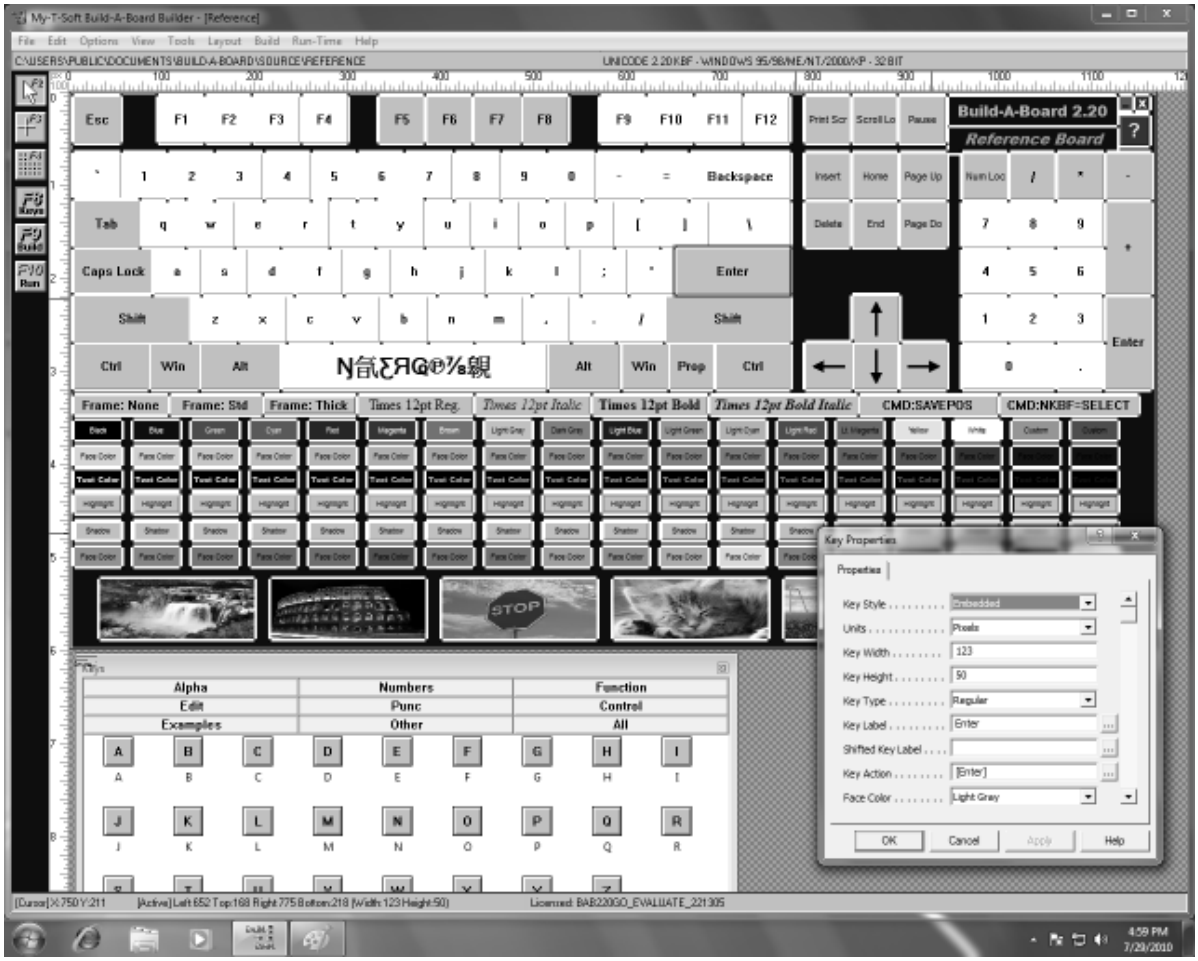
Mac OS X Systems:

You must install the Mac OS X run-time onto the target system, then copy required KBF files to the target system (My-T-Soft application folder). Use Run-Time | View Project Run-Time Targets folder to view/access Target KBFs from built project(s).

Using Build-A-Board Builder

The My-T-Soft® Build-A-Board Builder is the creation / design tool that allows you to create boards / panels / windows that contain keys & user-interface components, and manage your projects. This chapter outlines and describes

the various aspects and features available within the Builder. The next chapter covers additional details and reference information about working with the builder and creating boards.



Projects contain Source files, and build Targets - in most cases, the output Target folder also contains the run-time Program (or install files

for the run-time target selected). The data file is the KeyBoard File (.KBF) - this KBF contains the run-time ready layout designed in the builder.

Projects are saved in the Source folder (default [Shared Documents]\Build-A-Board\Source, e.g. \Users\Public\Documents\Build-A-Board\SOURCE). You may Build, Execute, and manage Run-time file sets from the Builder.

Projects can be configured to build for a selected Target (within any limitations imposed by the run-time target selected). These are saved as sub-folders in the Project folder in the TARGET folder (default [Shared Documents]\Build-A-Board\TARGET, e.g. \Users\Public\Documents\Build-A-Board\TARGET).

Note: All projects will have a TEST folder

that contains the Windows based run-time target files to display and operate a built board from within the Builder - when asked to "Run" a built target, the MYTSOFT.EXE in this folder will be launched, and display and operate with the current project's built board

The Builder displays several optional windows, along with the main Panel Display where you can manipulate a My-T-Soft Panel - adding/deleting/modifying keys, their contents & actions.

Project Selection

Panel Display

File Names

Toolbar

Rulers

Status

Cursor Tools

Menus & Settings

In the next chapter, detailed usage and reference material is covered.

Build-A-Board Overview & KBF Architecture

Building Boards

Run-Time Options

Key Properties

Window Properties

Global Settings

Project Properties

Projects & Files

Build-A-Board Text Compiler

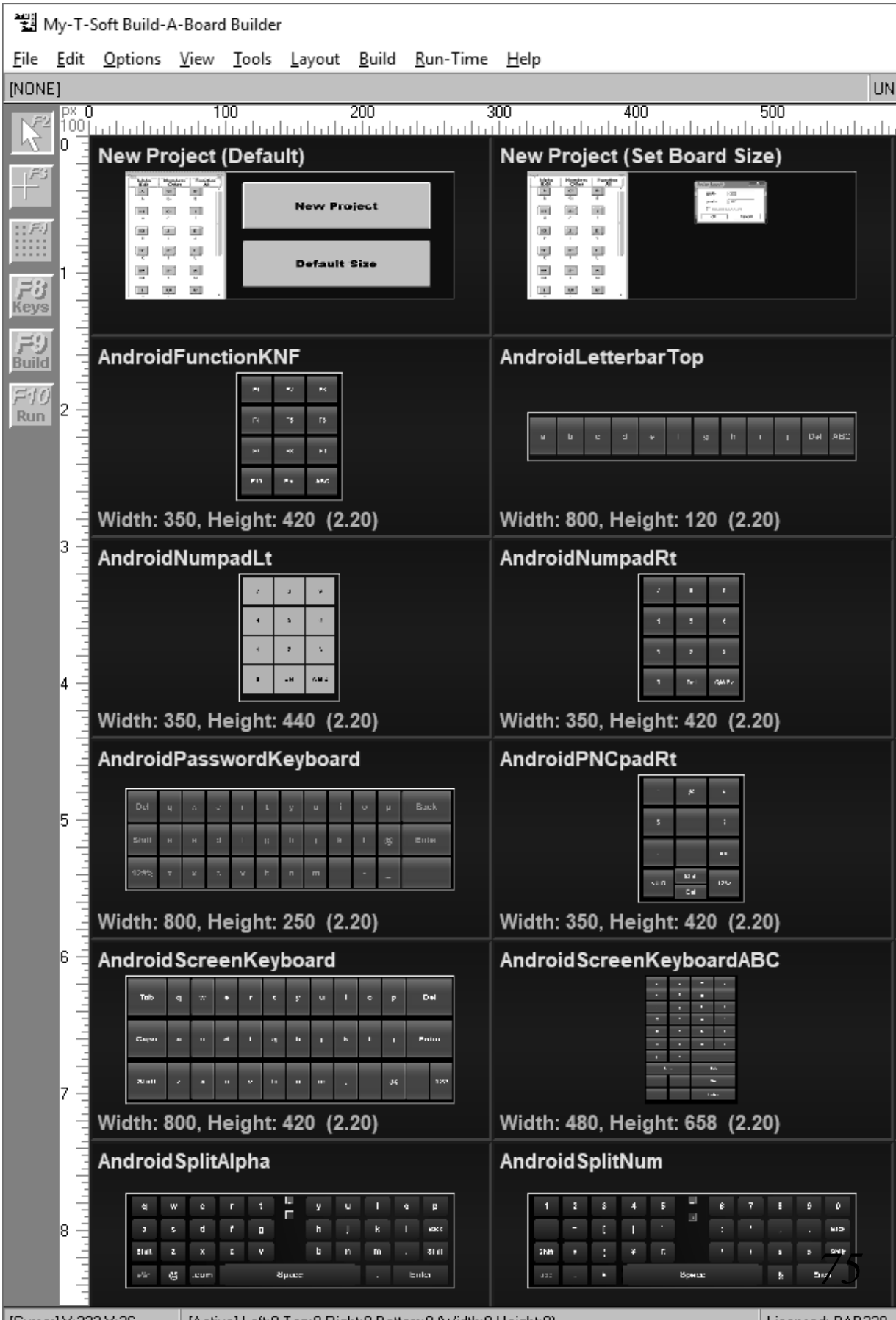
Keyboard Macro File (KMF) Notes

KeyBoard File (KBF) Notes

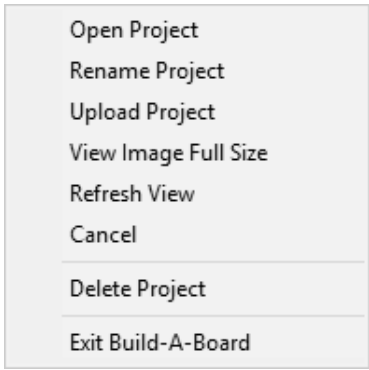
Macrobat Macro Reference

Project Selection

The Project Selection is the default window shown when not working on an actual project (when the Builder is started). There are 2 fixed entries - New Project (Default Size) and New Project (Set Board Size). Then all other existing projects in the current SOURCE folder location (see Global Settings) are shown. If managed and native to the Builder, an thumbnail image will be displayed of the Board in the listed project, along with width/height of the board.


My-T-Soft Build-A-Board Builder
File Edit Options View Tools Layout Build Run-Time Help
[NONE]
F2
F3
F4
F8 Keys
F9 Build
F10 Run
New Project (Default)
New Project
Default Size
New Project (Set Board Size)
AndroidFunctionKNF
Width: 350, Height: 420 (2.20)
AndroidLetterbarTop
Width: 800, Height: 120 (2.20)
AndroidNumpadLt
Width: 350, Height: 440 (2.20)
AndroidNumpadRt
Width: 350, Height: 420 (2.20)
AndroidPasswordKeyboard
Width: 800, Height: 250 (2.20)
AndroidPNCpadRt
Width: 350, Height: 420 (2.20)
AndroidScreenKeyboard
Width: 800, Height: 420 (2.20)
AndroidScreenKeyboardABC
Width: 480, Height: 658 (2.20)
AndroidSplitAlpha
AndroidSplitNum
[Cursor] X:332 Y:36
[Active] Left:0 Top:0 Right:0 Bottom:0 (Width:0 Height:0)
Licensed: BAB220

The mouse is used to select your project choice. Left-Click to select and open a particular project. You can scroll up or down with the right-hand scroll bar, or use Home/End, PgUp/PgDn. You can also right-click on a project, and access a context menu with several options.



You can Open Project, View Image Full Size, Refresh View, Cancel, Delete Project, or Exit Build-A-Board.

- **Open Project** - this opens the project for editing (same as a mouse left click).
- **Rename Project** - this opens a dialog where

you can enter a new name for the project.

- **Upload Project** - this uploads the selected project to your Build-A-Board.com account. The system must have a valid license obtained from your Build-A-Board.com account to correctly operate this feature. A separate process will open a status window during the upload process.
- **View Image Full Size** - the opens a window that displays the current project's image in full width/height. This window can be closed by clicking on it, or moving to another project selection.
- **Refresh View** - this will requery the source folder and redisplay the Projects available.
- **Cancel** - cancels the context menu.
- **Delete Project** - this is the only way to remove and delete a project from within the Builder. You must verify this action, and be

aware that the source files and folders, and all data pertaining to the project are deleted, and there is no way to recover this data once a project is deleted!

- **Exit Build-A-Board** - Exits and closes the Builder application.

The available project selection is a special view into the SOURCE folder of projects. If you are unable to view projects, be sure to check the Global Settings of the properties page (F7) and verify the SOURCE location is correct.

When a project is opened, or saved, an image bitmap file is created in the SOURCE folder - this .BMP file name must match the Project name (also the Project's folder name) to be shown in the Project Selection window. If a folder exists that does not have an image associated with it, an "Unknown/Question mark keyboard layout" will be displayed - to update/create the image shown in the Project

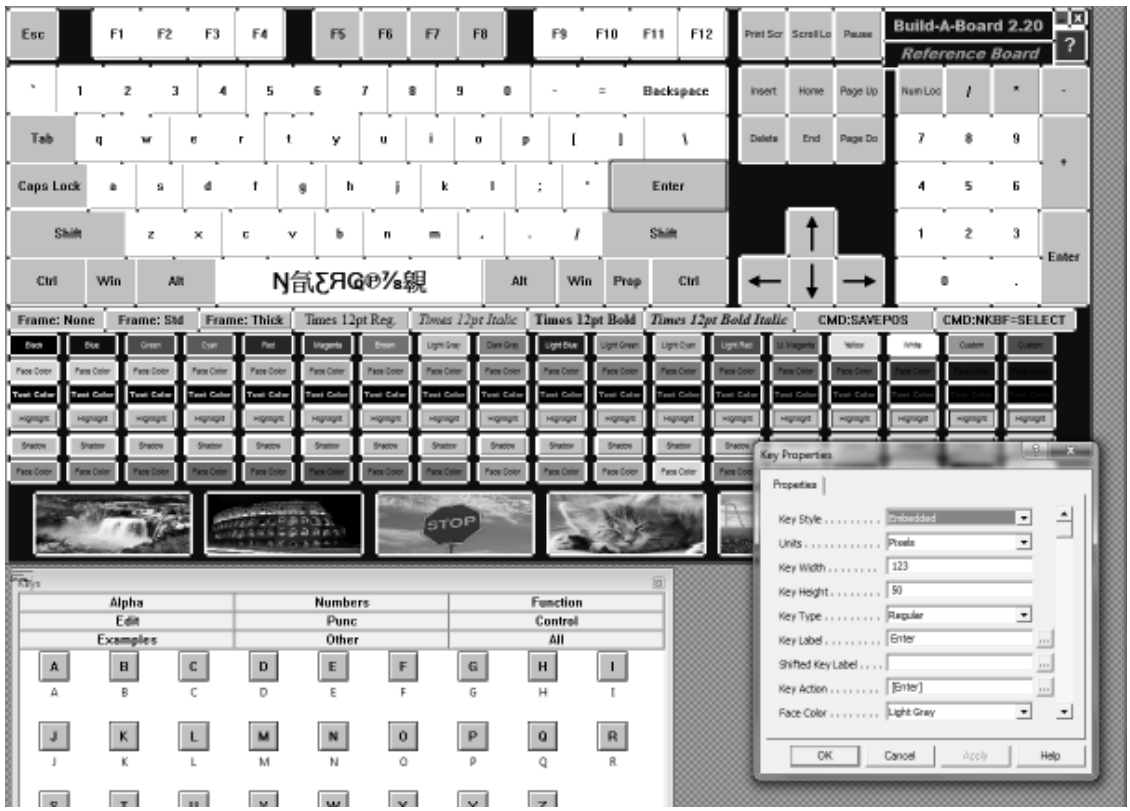
Seletion window, you will need to open the project.

> **Note:** The images and source files are managed within the Builder - users should not work with or manipulate these files externally unless they are familiar with files and structures (i.e. a developer or technical support personnel).

Panel Display

The panel display is the largest portion of the Builder - it may not be removed. A My-T-Soft Panel may contain keys & have certain properties (Frame type, background color, etc.). The Panel Display area contains the Panel being worked on - if the Panel is smaller

than the work area, a gray cross-hatched background is displayed.



There are many options for working with the Panel and Keys in this main window within the Builder. Most of the detail and reference information is in the next chapter.

Panel

Note: In Versions 2.00, 2.10, 2.20 the

Panel displayed in the Build-A-Board IS the My-T-Soft window display. The terms panel and board are interchangeable. The support of multiple windows, and multiple panels per window is managed at a run-time level, and is not handled within the Builder.

To Access Panel Properties (My-T-Soft Run-Time Window Properties) & Global Settings, right-click on the panel (area clear of any keys), then select Properties from the menu. This is also accessible with the F7 key.

You can resize the panel (board) and can either resize the keys with the board, or just resize the board (key size and positions do not change).

Keys in Panel

Key Properties

To Access Key Properties, right-click on the Key face, then select Properties from the menu. You can also double-click on a key to go directly to the Key Properties.

Key Frames

Keys may be either selected, selected as the current selection, or unselected. The Key state is indicated by the Frame around it in the Builder (View menu). When frames are shown, the following indicates the Key state. You may Show/Hide frames with F5 or View Menu | View Frames. When View Frames is selected, you can limit the view to only selected keys by selecting View Menu | View Frames - Selected only.

Frame color - Key state

All White - unselected

Dark borders - selected (but not current selection)

Blue borders - selected as current selection

Key Sizing / Key Moving

Keys can be sized / moved by using the frames (click & drag on frame segments). There are 8 frame segments per framed key, and can be used for left/right/up/down, or diagonals up/left, up/right, down/left, down/right. You can also move selected keys with the cursor keys - Arrows Left/Right/Up/Down. Frame moves are selected by holding the Shift key down while using the Arrow keys.

Key Alignment / Spacing / Center / Match Size

Keys can be Aligned, Spaced, Centered, size-matched via the Layout menu, and there are various keyboard short-cuts (see menu for details), e.g. the Ctrl/Arrow combinations does alignment of keys when there are multiple selections. For example, you can align a row of

keys by dragging a selection rectangle around the row (selecting all keys), then using Ctrl-Up arrow to align the top of the keys (always referenced to the current selection (blue border)).

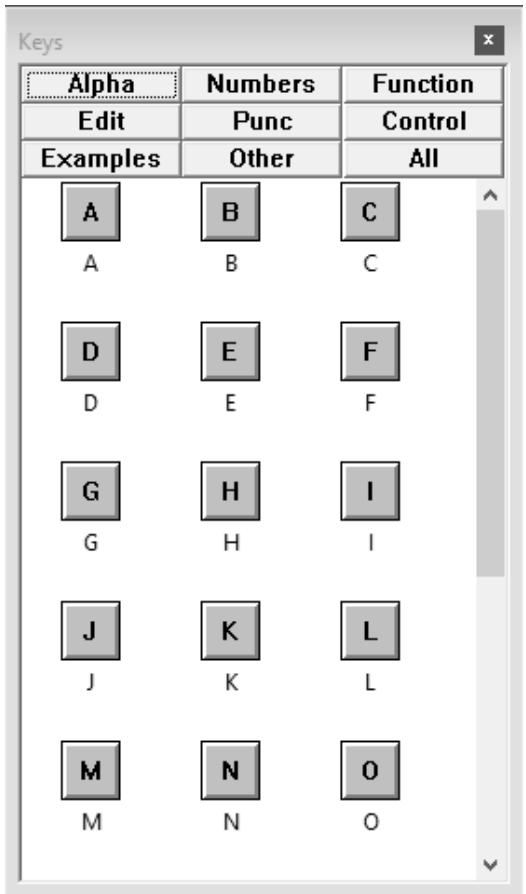
Grid

You can toggle the view of a Grid to help you align keys with the F4 key, or Tools Menu | Grid. When the Grid is shown, keys will snap to grid points (top/left of key when dragging key, or frame segment being moved). You can increase/decrease the Grid size with the Tools Menu | Grid + and Tools Menu | Grid -, or the keyboard shortcuts Shift-F4/Shift-F3. Size ranges from 2 to 20, and current grid point size is shown on the top / left of the grid and the tool bar grid icon.

Keys Window

The Keys Window is a quick selection tool to

drag and drop keys onto a board - there are various categories (Alpha, Numeric, Edit, All, etc.) You can toggle the view of the Keys window with the F8 key, the View menu | View Keys, or the Tool bar.



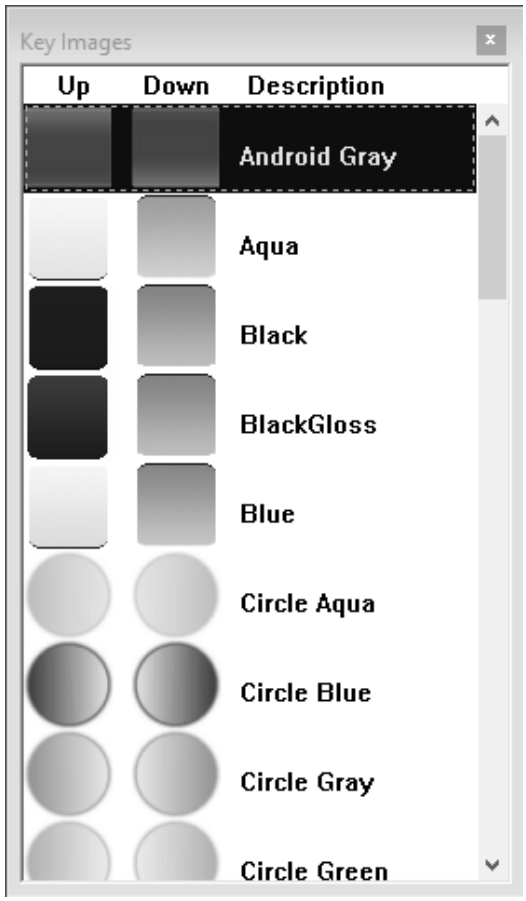
Technical Note: The Keys window is managed by the KEYS.INI file in the BIN folder - the actual Keys shown, and the

default labels and actions are contained within this file, and is relatively easy to modify for the technically minded - syntax notes and details are listed as comments in the file itself. Note that the 9 sections are fixed within the code itself, so the sections themselves cannot be added or subtracted.

Key Images Window

The Key Images Window is a quick selection tool to select key images for a key (or keys) - Key Type must be HiRes, and have the Frame Type set as Key Images. All keys on the open project that have these attributes will be affected by the selected Key Image (there is only 1 Key Image per board). You can select the current Key Image by highlighting and pressing Enter, or double-clicking on the desired key image.

Note: For Android targets, the actual image selected here will not be carried through to the run-time device, but only if Key Images are enabled in Build-A-Board will the option of selecting from the available Key Images in settings operate.

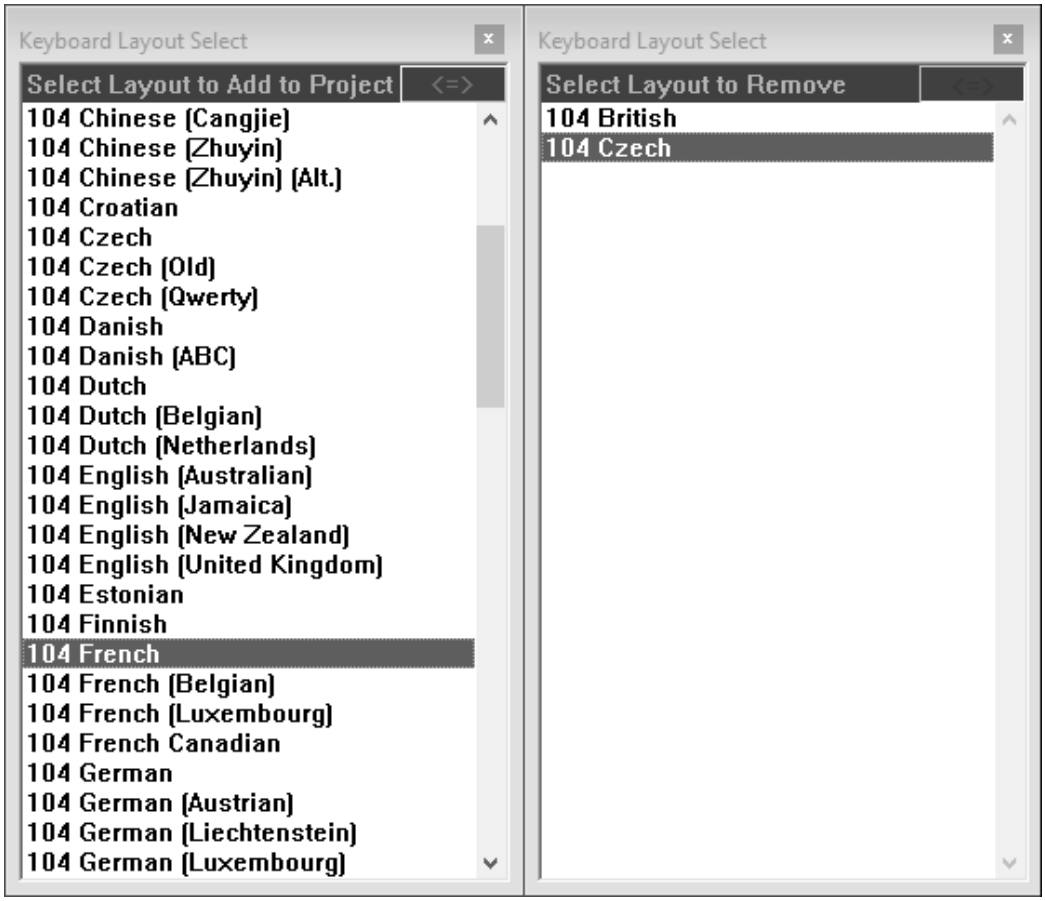


Keyboard Layout Window

The Keyboard Layout Window shows Version

1.xx KMF layout files that contain different labels / actions from the older 1.xx software. The Keyboard Macro Files are essentially lookup tables that can be referenced when the correct reference option is put into the Key Label and Key Action key properties. As an example, look at the MTSBASIC layouts, e.g. MTSBASIC_0800_AO_BASE. If you look at the A key, you will see it shows the Key Label as {KMF:41}, the Shifted Key Label is {KMF:Shift-41}, and the Key Action is [KMF:41]. These overrides look at the current selected KMF to display the Key Labels, and perform the Key Action. The key Change Language uses the [CMD:NEXTKMF] to scroll through the available KMF files (as selected in the Keyboard Layout Window). To remove layouts, click on the <=> to toggle between available layouts and current layouts - select from available layouts to add, select from current layouts to remove. To see Key

number reference for the KMF files, refer to the Build-A-Macro Notes in Advanced User Notes.



File Names

The File Names window is used to display the current project and target system.

The Left-hand portion of the File Names window shows the current project. If it indicates [None], then there is no current project in use. By default, a project will be saved as Untitled if no named project is used. Use File | Save As to save the current project as a different named project.

The Right-hand portion of the File Names window shows the current selected Target System. To change the current Target System, use Run-Time | Select Target System.

The distinction between ANSI 2.10 and UNICODE 2.20 is important, especially when moving between different target types for the same project, or working with older 2.10 targets. The first portion of the Target indicates what mode the Builder is in. If the actual target system is different than the Builder Mode, you may get warnings, or be unable to properly

work with the project or target. You should always convert and work with the UNICODE 2.20 versions, unless you are targeting an older system that only can support the older ANSI 2.10. For older projects, and projects that target ANSI 2.10, you should not select UNICODE targets, or convert to 2.20 in the File menu, as projects targeting UNICODE 2.20 cannot be converted back to ANSI 2.10 without losing key information.

To Show or Hide the File Names window, use View | View File Names. You may also right-click on the File Names window & select Hide to remove the window from the Build-A-Board workspace.

Toolbar

The Toolbar window is used to access various

tools with the pointing device.

All of the Toolbar options are available via Menu Selections, or direct by keyboard (Function key). The Toolbar is just an aid to quickly access common options.

Select [F2]

The Select option shifts the Cursor Tool to the Select mode, where keys can be selected, or groups of keys can be selected.

Add [F3]

The Add option shifts the Cursor Tool to the Add mode, where keys can be added just by

clicking and dragging, or single-clicking. When the Grid mode is On, multiple keys can be added by clicking and dragging - key size is controlled by the default key size (key property menu / reset in Options menu), and spacing is controlled by the current grid spacing.

Grid [F4]

The Grid option toggles the grid mode on and off. When in Grid mode, keys snap to grid points. Grid points can quickly be expanded/contracted with Shift-F3 and Shift-F4.

Keys [F8]

The Keys option toggles the Keys window visible or hidden.

Build [F9]

The Build option initiates the Build process for the current project with the currently selected

target.

Run [F10]

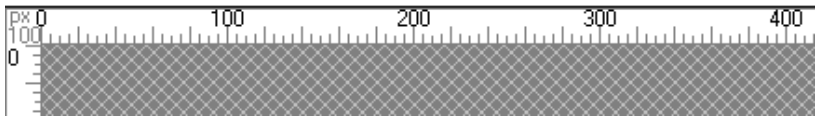
The Run option runs the Test (within the Builder) run-time target (Windows run-time) to display and operate the most currently built target KBF.

To Show or Hide the Toolbar window, use View | View Toolbar. You may also right-click on the Toolbar window & select Hide to remove the window from the Build-A-Board workspace.

Note: The Toolbar is not user-modifiable

Rulers

The Rulers windows are used to display a guide in the units of the current panel.



The Rulers position themselves to the left & top of the Panel Display work area.

To Show or Hide the Rulers windows, use View | View Rulers. You may also right-click on the Ruler window & select Hide to remove the window from the Build-A-Board workspace.

Status

The Status window is used to display the cursor position, Active key information, and

License information.

The Cursor position is in units of the current panel, in reference to the panel. The upper-left corner is position 0,0.

The Active key (Selected & highlighted blue) details show size, position, width & height.

To Show or Hide the Status window, use View | View Status. You may also right-click on the Status window & select Hide to remove the window from the Build-A-Board workspace.

Cursor Tools

My-T-Soft Build-A-Board has several selections & uses for the mouse cursor during the creation phase to assist the user while creating layouts.

Select Mode (Tools | Select)

Left-click on a key: Select key and make active

Left-click & drag on key: Moves key

Left-Double-click on key: Opens Key Properties page

Left-Double-click on board (panel) background: Opens resize board dialog

Left-click & drag on board, then release: Opens selection rectangle from original anchor point - selects any keys within boundary

Left-click on board (panel) background - clear current key selections

Add Mode (Tools | Add)

Left-click & drag, then release: Opens rectangle from original anchor point, adding new key at location & sized to rectangle upon release of mouse left button. If the Grid mode is on, this will add multiple keys (in a grid

layout) based on the current default size, separated by the current grid size spacing.

Left-click & on blank area: Add key with default size.

In either mode, the Right-click acts as follows:

Right-click: Opens context menu for pointed to object (Board, Key, etc.)

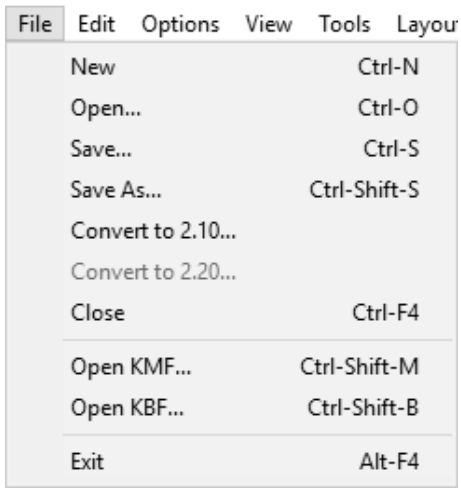
Right-click & drag on board: If board is larger than display area, positions board within display area (Moves board directly, quicker than using scroll bars)

Menus & Settings

The following lists detailed information about the Build-A-Board menu selections, along with the settings they control.

Note: Menu options will be grayed (disabled) when they are not appropriate, or don't apply based on various selections & settings. Ensure you have the correct/appropriate Key (or Keys) Selected, or check selected options if you are trying to perform a menu option and it is not available.

File Menu



New (Keyboard Shortcut = Ctrl-N)

- Creates a New Project (Untitled). You will

be asked to save the current project if necessary.

Open... (Keyboard Shortcut = Ctrl-O)

- Opens an existing Project. You will be asked to save the current project if necessary prior to opening another.

Save... (Keyboard Shortcut = Ctrl-S)

- Saves the current project - if no name selected, it will be saved as Untitled.

Save As... (Keyboard Shortcut = Ctrl-Shift-S)

- Saves the current project with the name specified.

Convert to 2.10...

- This will only be enabled for 2.20 / Level 2 projects. The Key Properties for the layout

are updated, and saved in the Level 1 format. You will be prompted for a new Project name (Save As...). Converting to 2.10 will lose extended key information, and will not be recoverable if the 2.10 project is converted back to 2.20. This conversion is only recommended for projects that were created as 2.20 and the 2.10 layout is needed due to legacy run-time software.

Convert to 2.20...

- This will only be enabled for 2.10 / Level 1 projects. The Key Properties for the layout are updated, and saved in the Level 2 format. You will be prompted for a new Project name (Save As...). Defaults are used for 2.20 / Level 2 values that are not in a 2.10 / Level 1 project.

Close (Keyboard Shortcut = Ctrl-F4)

- Will ask to save if necessary. This will compress all project files into 1 "Closed Project File" and remove the current project from the Build-A-Board workspace.

Open KMF... (Keyboard Shortcut = Ctrl-Shift-M)

- This will open a KMF file from pre 2.00 versions. There is no editing available for KMF files, and for display purposes, a matching pre 2.00 KBF file must be available. For more information on KMF Files and details on this option, see KMF File Notes.

> **Note:** More advanced capabilities for this option will be available in the future.

Open KBF... (Keyboard Shortcut = Ctrl-Shift-B)

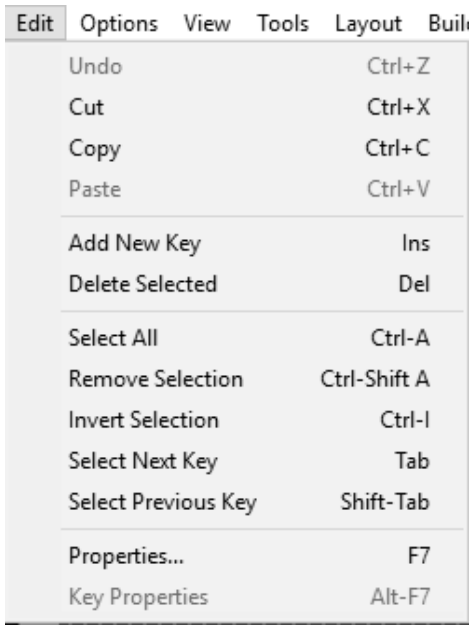
- This will open an existing KBF file from pre 2.00 versions. Limited Editing is available - the ability to hide keys (Key Properties), and resize keys can be handled within the current version. For more information on KBF Files and details on this option, see KBF File Notes.

> **Note:** More advanced capabilities for this option will be available in the future, including the ability to automatically extract KBF files to Source projects.

Exit (Keyboard Shortcut = Alt-F4)

- This will Exit & Close Build-A-Board - you will be asked if you wish to save the current project (if necessary).

Edit Menu



Undo (Keyboard Shortcut = Ctrl+Z)

- Reverses the previous action(s) - up to 25 undo levels are supported

Cut (Keyboard Shortcut = Ctrl+X)

- Copies the selected key(s) to the clipboard, and deletes them from the current project.

Copy (Keyboard Shortcut = Ctrl+C)

- Copies the selected key(s) to the clipboard.

Paste (Keyboard Shortcut = Ctrl+V)

- Pastes the current clipboard contents onto the Panel.

Add New Key (Keyboard Shortcut = Ins)

- This Adds a key onto the panel. The default size is 20 pixels wide, 30 pixels high.

Delete Selected (Keyboard Shortcut = Del)

- This deletes any selected keys.

Select All (Keyboard Shortcut = Ctrl-A)

- Selects all keys on the Panel.

Remove Selection (Keyboard Shortcut = Ctrl-Shift A)

- Removes selection from all keys.

Invert Selection (Keyboard Shortcut = Ctrl-I)

- Selects all currently non-selected keys, and removes the selection from currently selected keys.

Select Next Key (Keyboard Shortcut = Tab)

- Selects and highlights the next key in the internal order.

Select Previous Key (Keyboard Shortcut = Shift-Tab)

- Selects and highlights the previous key in the internal order.

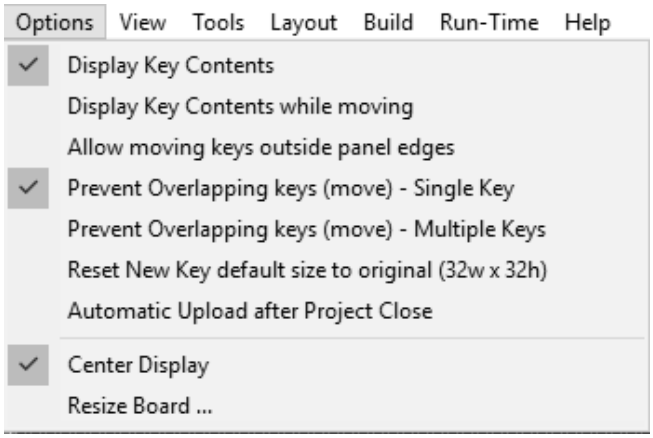
Properties... (Keyboard Shortcut = F7)

- Opens the Property page for the Panel - you may also access Global Settings from this page (Global Settings tab)

Key Properties... (Keyboard Shortcut = Alt-F7)

- Opens the Key Property page for the current actively selected Key (if available).

Options Menu



Display Key Contents

- When selected On, this will show the Key Contents. This is default On, you would only want this off if you have a slower system with a slow video card, or notice paint delays on the keys while working within Build-A-Board.

Display Key Contents while moving

- This is default Off, as it repaints the keys while you are moving them. This should only be On for fast systems with high-speed video capabilities.

Allow moving keys outside panel edges

- This allows keys to move outside the edges of the panel. It is default Off, and limits keys from moving outside the edges of the panel.

Prevent Overlapping keys (move) - Single Key

- This forces a moving key to position itself in a clear area of the board. It is default On - while moving a key, you will notice it "hop" around as it tries to find a clear area to position itself.

Prevent Overlapping keys (move) - Multiple Keys

- This monitors all keys during multiple key moves. It is default Off, because often there is not enough free space to position all keys being moved to a clear area. If On, Keys can position constantly if there is not enough free space to position they keys. This should be used with caution, and the awareness that there may not be a possible non-conflict during a multiple key move.

Reset New Key default size to original (32w x 32h)

- When working with Keys, you can select the default size for New Keys based on the current Key size. This Option allows you to reset to the 32 pixel wide by 32 pixel high default size.

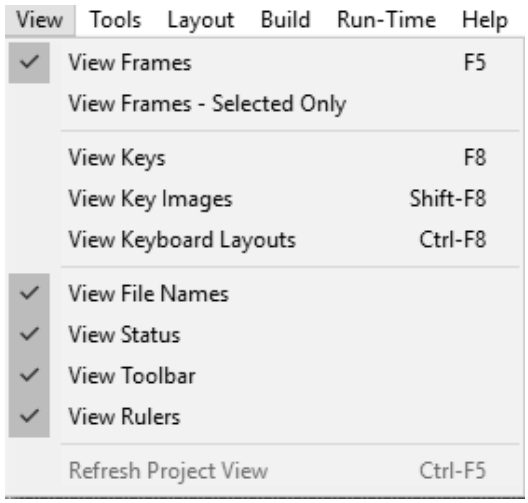
Center Display

- This is default On, and positions the panel in the center of the Build-A-Board work area. When Off, the panel will be positioned in the upper-left of the work area.

Resize Board ...

- This opens a dialog to resize the Panel (Board) - you also have the option to resize the keys in proportion to the new sized board.

View Menu



View Frames (Keyboard Shortcut = F5)

View Frames - Selected Only

- When View Frames is On, the frames around the keys is shown - for more details on the frames, see Panel Display, Keys.

- When viewing frames, you may also select the option to only view the frames of the Selected Keys.

- To Show the keys without the selection frames, toggle View Frames to Off. Note

that you may still select a key or keys in the same way, but there will not be any visual indication of the current selection. Sizing will need to be done via the keyboard cursor keys (arrows), or the Key property page.

View Keys (Keyboard Shortcut = F8)

- This toggles the Window showing the Drag & Drop Keys for the 9 categories - Alpha, Numeric, Edit, All, etc.

View Key Images (Keyboard Shortcut = Shift-F8)

- This toggles the Window showing the available Key Images that can be assigned to a board and will be used on all keys that have the HiRes Key Type and have Key Images as the Frame Type.

View Keyboard Layouts (Keyboard Shortcut = Ctrl-F8)

- This toggles the Window showing the available Keyboard Layouts and currently selected Keyboard Layouts (KMF Files) that are included with the project, and can be used for {KMF:??} images and [KMF:??] Key Actions. Keyboard layouts can be added and removed from the included layouts with the project / built KBF.

View File Names

View Status

View Toolbar

View Rulers

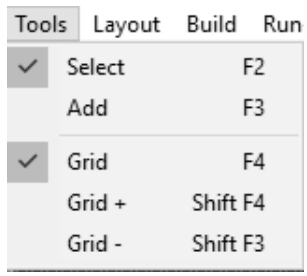
- These 4 items on the View Menu allows you to toggle on & off the display of the various

windows within the Build-A-Board workspace.

Refresh Project View

- This View option is only available when in the Project Select view - if selected, it will reload the current Source folder, and update the Project View..

Tools Menu



Select (Keyboard Shortcut = F2)

Add (Keyboard Shortcut = F3)

- Select & Add are the 2 cursor tools in Build-A-Board - they are mutually exclusive. See Cursor Tools for more details.

Grid (Keyboard Shortcut = F4)

- The Grid toggles on & off the grid display within the Panel.

Grid + (Keyboard Shortcut = Shift-F4)

- This increases the current Grid size by 1, up to the maximum grid size.

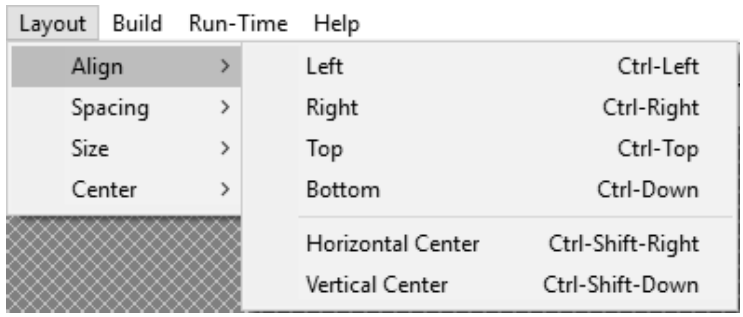
Grid - (Keyboard Shortcut = Shift-F3)

- This decreases the current Grid size by 1, down to the minimum grid size.

Layout Menu

The Layout Menu has several sub-menus with options to align, space, size, and center selected keys.

Layout Menu | Align



The Align commands all use the active selected key (blue-border) as the alignment anchor.

Left

- Positions all selected keys to have the same left-hand position as the active key.

Right

- Positions all selected keys to have the same right-hand position as the active key.

Top

- Positions all selected keys to have the same top position as the active key.

Bottom

- Positions all selected keys to have the same bottom position as the active key.

Horizontal Center

- Positions all selected keys to have the same horizontal-center position as the active key.

Vertical Center

- Positions all selected keys to have the same vertical-center position as the active key.

Layout Menu | Spacing

The Spacing command uses the minimum & maximum (Horizontal or Vertical) position of currently selected keys, and evenly spaces keys (with 3 options - spacing wins or keys win or fixed pixels).

Horizontal (Even Spacing, Max Key may move)

- Forces even spaces between keys, may move the right-hand maximum key to accommodate.

Vertical (Even Spacing, Max Key may move)

- Forces even spaces between keys, may move the bottommost maximum key to accommodate.

Horizontal (Min/Max Keys fixed, spacing may be uneven)

- The far-left & far-right keys remain fixed, so spacing may be uneven if units do not evenly divide.

Vertical (Min/Max Keys fixed, spacing may be uneven)

- The topmost & bottommost keys remain fixed, so spacing may be uneven if units do not evenly divide.

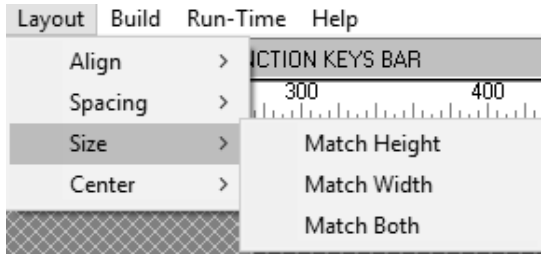
Horizontal (Pixels / Even Spacing, Max Key may move)

- Opens a pixel sub-menu to select between 1-10 pixels, forces even spaces between keys, may move the right-hand maximum key to accommodate.

Vertical (Pixels / Even Spacing, Max Key may move)

- Opens a pixel sub-menu to select between 1-10 pixels, forces even spaces between keys, may move the bottommost maximum key to accommodate.

Layout Menu | Size



The Size command changes the selected key(s) size to match the active selected key (blue-border).

Match Height

- The height of all selected key(s) will match the active selected key.

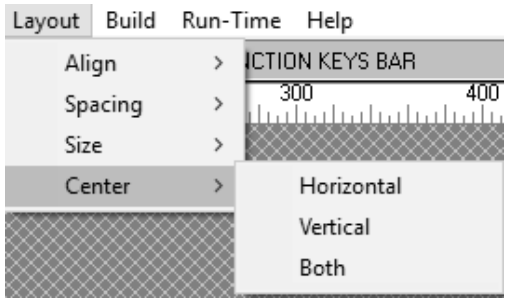
Match Width

- The width of all selected key(s) will match the active selected key.

Match Both

- The width & height of all selected key(s) will match the active selected key.

Layout Menu | Center



The Center command will center a selected key or key(s) horizontally, vertically, or both.

Horizontal

- Center the selected key or key(s) within the width of the panel.

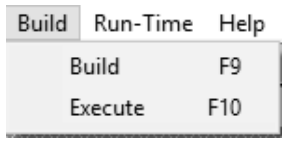
Vertical

- Center the selected key or key(s) within the height of the panel.

Both

- Center the selected key or key(s) within the width & height of the panel. (Center selected key(s) in panel).

Build Menu



Build (Keyboard Shortcut = F9)

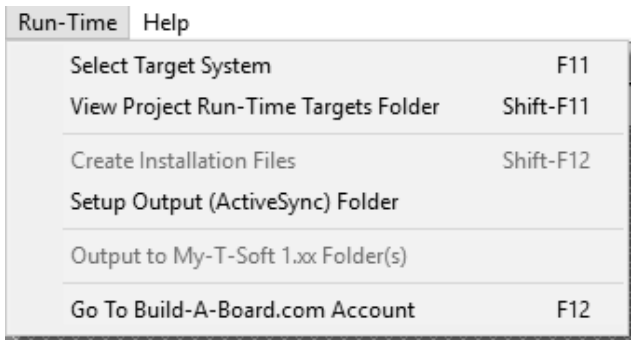
- The Build command will save the project, compile the project files, and create the run-time files for the selected target system. By default, a test configuration will be copied into the Target folder for the project to allow you to run & test the current layout. After a successful build, you will be asked if

you wish to execute My-T-Soft (from the test configuration). Note: This will close any running instance of My-T-Soft in order to prepare for a new launch.

Execute (Keyboard Shortcut = F10)

- This will execute My-T-Soft with the keyboard layout from the last successful build for the project. Note: This will close any running instance of My-T-Soft in order to prepare for a new launch.

Run-Time Menu



Select Target System (Keyboard Shortcut = F11)

- This will open a dialog to let you select the target system from the available target systems.

View Project Run-Time Targets Folder (Keyboard Shortcut = Shift-F11)

- This opens Windows Explorer and displays the Project folder in the Target area, which displays all built targets, and provides access to the built KBF file and available Targets for the current project.

Create Installation Files (Keyboard Shortcut = F12)

- Creates Installation Files is not enabled in this version. (This is an older option when target platforms were limited and security did not impose as many restrictions on building installation packages - it may be

used for certain platforms at some point in the future)

Setup Output (ActiveSync) Folder

- This allows you to choose a folder to output the current build after the build is complete. This can be used as the ActiveSync synchronization folder to allow Build-A-Board to copy successfully built target files into (which allows ActiveSync (or any synchronizing utility) to automatically copy to the target system). This also may be used for convenience or as another option for placing built layouts (.KBF files).

Output to My-T-Soft 1.xx Folder(s)

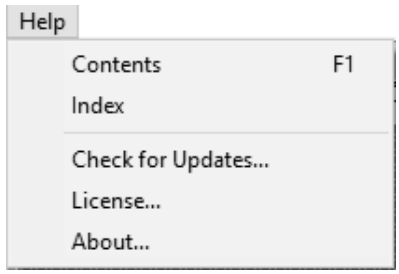
- This will be enabled if a suitable member of the My-T-Soft family 1.xx is installed (My-T-Soft / My-T-Touch / My-T-Pen /

My-T-Mouse / OnScreen), and there is a built project KBF available. When selected, the [Project Name].KBF will be copied to all available product folders, and will be available from the Custom Build-A-Board keyboard selection option in Setup | Keyboards. You must be sure to target the correct version ANSI 2.10 (1.7x) or UNICODE 2.20 (1.8x +).

Go To Build-A-Board.com Account

- If properly licensed, this will open the default browser and go direct to the Licensed User's Build-A-Board.com account for accessing options and layouts from the Build-A-Board.com website. If there is no account information with the license, this will go to the login page.

Help Menu



Contents (Keyboard Shortcut = F1)

- Opens this Help File from within Build-A-Board to the Table of Contents page.

Index

- Opens this Help File from within Build-A-Board to the Index page.

Check for Updates...

- Opens the IMG Download Manager and can check for and install product updates.

License...

- Opens the IMG License Manager to display current license details and provide tools from the License Manager.

About...

- Displays version & contact information for Build-A-Board.

Chapter 4. Building Boards & Reference

Build-A-Board Overview & KeyBoard File (KBF) Architecture

Build-A-Board has been under constant development since its beginning as a developer's tool, primarily focused on creating on-screen keyboards and deploying user interface components on various platforms, all while being influenced from customer demands, new technology, and changing uses of computing devices. Being software, and implementing a flexible, yet solid core of

well-designed software, while targeting multiple operating systems and platforms (which often go through considerable changes on a regular basis) is an on-going challenge. This is a quick overview of the structure & design of Build-A-Board.

Distinct components

- **Builder**
- **Source Files**
- **Compiler**
- **KBF (Data)**
- **Run-Time Targets (Program)**

Builder

The Builder is the user interface component that gives flexibility to the keyboard layout designer. It provides certain functions and capabilities that may not be desirable for an

operator using the on-screen keyboard. In practice, it is often used by a developer, and 1 instance of the Builder may be responsible for thousands of run-time keyboard deployments. It is treated as a "rich" application, and is constrained differently than the run-time aspects. Also, there are multiple builder tools - Windows based application, and a web-based application.

> **Note:** The web-based application is not yet publicly available as of this release

Source Files

The Source Files (or SOURCE project) is the definition of the keyboard layout as managed by the Builder. It exists in different ways - internal memory structure within the Builder, Text files in the Project SOURCE folder,

database records, with translation methods available to manage these. Note also that using the utility KBFDUMP, a Data file KBF can be extracted to Source files.

Compiler

The Build-A-Board Text Compiler (BABTC) is used to read and condense the information in the Source Text files into the binary representation. This approach is used to ensure the Data component can be in as small a footprint as possible, and the translation from Source to Target usable representation forces design discipline & also provides data validation/verification.

KBF (Data)

The KeyBoard File (KBF) structure can be viewed as a flat file database, or as multiple files joined with an index table. Once built, the file size is fixed - any user-modifiable

options/variables/areas are constrained to existing data size locations in the KBF. This approach is a result of various factors:

- For embedded systems, or for secure operator situations, having a read-only, fixed Keyboard File (as similar as software can be to a physical hardware based keyboard in this context) was deemed an important aspect during the original design, especially for hardware engineers implemented an on-screen keyboard as a replacement to a physical keypad. This "fixed" aspect adds a level of comfort for hardware engineers when working with the software. Also, the software must also operate correctly in situations where the data is ROM (Read-Only Memory).
- A single Data file ensures easier handling for system administrators, integrators, and

end-users.

- Add-ons are easy to manage - additional files can be added to the single-file structure with very little effort.
- In practical usage, once a fixed layout is accepted, changing it can have adverse affects on standard and limited operators, so treating the KBF as a fixed end-result works best (and it can always be changed with the Builder, so there is no down-side to this approach, however, any other approach may be unacceptable, i.e. secure operator situations where the operator must have NO configuration options).
- Multiple KBFs can be used with the New KBF (CMD:NKBF) and Load KBF (CMD:LKBF) commands
- Conceptualizing the Keyboard layout AS the KBF itself works better from experience

than working with the separate layout and labels/actions aspects (see the KMF/KBF structure in the IMG My-T-Soft Family (1.xx))

The current information stored within the KBF File:

- File signature and version, along with reference/lookup (index) into the rest of the file.
- Window data - location, size, frame type, etc.
- Panel data - location, size, frame type, etc.
- Key (Button) reference data - location, sequence.
- Key data - size, labels, actions, colors, etc.
- State data - reference information on possible operational states, etc.

- Image data - Image scripts used for Keys, Panels, Windows.
- Action data - Action scripts used for Keys, Panels, Windows.
- License data - License for KBF / Platform License if applicable.
- INI data - Initialization file (INI), includes General Program settings, User modifiable settings.
- KMF data - Keyboard Macro Files, default key mappings, etc.
- Image files - Image files as PNGs.

Run-Time Targets (Program)

The original design goal was to have a Program/Data type approach - i.e. My-T-Soft (Program) presented the keyboard layout contained in the KBF (Data) file. Depending on the system, there are sometimes other

support files (Libraries or data files), and the Program is often 2 processes (one handling user interface & the other key actions). As much as possible, limiting the required Run-Time files is the preferred approach. Certain aspects (i.e. image support/library calls/executables) are treated as "non-core" capabilities of the software, and these are handled with additional files when required (e.g. PNG images are handled natively in the Mac OS X API, but require libpng in Linux, and interfacing with GDIPlus.dll in Windows via the gdikit.dll add-on).

The run-time target approach (on most platforms) is to separate the user display / user interface, and the resultant key actions as separate processes. The MYTSOFT/My-T-Soft/mytsoft process is responsible for the user interface, and the Macrobat/macrobat process is responsible for

the macro batch processing (key actions). This is done for various reasons:

- Interoperability - different operating systems handle virtual input differently, so keeping this operating system specific portion as a separate process reduces potential conflicts.
- Flexibility - by having the input handled by a virtual input server, this process can provide these services to other processes in the system.
- Historical reasons / future extensibility - The Macrobat (Macro Batch) processing syntax and handling as a separate process is integral in various IMG products, and this is seen as an important architectural design for the future. For example, having a specific computing devices as a user-input device (e.g. a separate handheld device) and then requiring the keyboard input on a different,

main device, i.e. **REALLY** separating the user interface from the virtual input process!

- Approach - from experience, having 2 distinct aspects managed in separate ways provides a much more robust approach. For example, some MIDI (Musical Instrument Digital Interface) devices use a controller / synthesizer design approach, where the user based controller (e.g. piano type keyboard, or guitar) can run the synthesizer directly, but the device's synthesizer may also be controlled via external MIDI. Designing the run-time approach in this way provides other capabilities and options that would not be available if the user-interface was tightly integrated with the virtual input server.
- Arbitrary Key Action - initiating a Key Action based on a user input can also be extended to other arbitrary events, handled by the Macrobat process, or handled by a

completely different process (or contained in a library), or even handled by system or other available services. This provides even more flexibility to a keyboard, i.e. providing non-keyboard based actions.

In other words, the design approach used is that the user-interface is just a way for the user to tell the computer to perform an action. Since this action could theoretically be anything imaginable, it is unlikely that every possible action could be embedded into one process. Therefore, treating the action as an event to be handled by some other process is the best design approach to use, resulting in a separate process to handle the keyboard actions (i.e. Macrobat). So even though the use as a keyboard is the default Key Action, the underlying design just treats this as one of many possible Key Actions.

Android / Input Method type interface - newer

platforms treat the keyboard and text input in an integrated approach, which provides its own set of benefits and limitations. Because the text processing / input is more of an extension of a system edit control/field (vs. a separate, independent text input device), certain aspects must be handled in a system compatible fashion. Because this system interface has various aspects dictated by the system itself, features and capabilities typically have to be implemented based on the available options provided by the platform. Because this inherently results in its own platform-based implementation, features and operation may be different than other platforms, and not all features may be available.

Building Boards

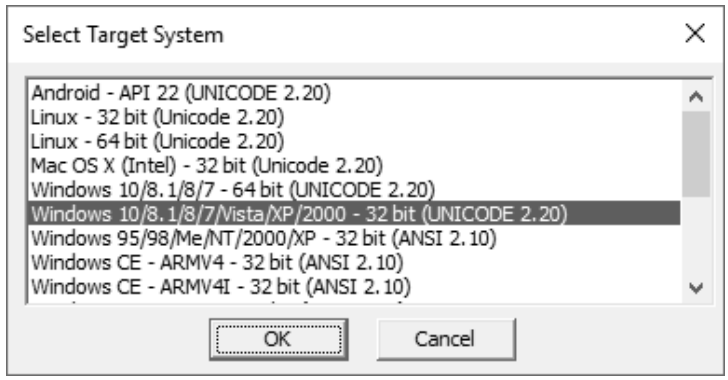
The process of building boards is fairly straightforward - start with an existing sample, or start from an empty board. Add keys and modify keys as necessary to meet your requirements. Build & Test to work with keyboard layout. Finally, deploy on target systems.

An iterative design/build/modify/build process is the most effective, as a large part of the actual use involves humans - no matter how well you think it through, actually letting users operate with a built board will often provide important information as to what additional changes are necessary.

Build Process Overview

The Build process within the Builder transforms the source files into a single file KBF (KeyBoard File) that contains all layout,

key, label, action, image, and run-time option information. Each Build process builds for the currently selected Target.



Note: The currently selected Target can affect features and capabilities within the Builder. For example, when working on pre 1.80 KBF files, ANSI 2.10 KBF targets, or Unicode 2.20 KBF targets, different Key properties will be available.

IMPORTANT NOTE: For older projects, and projects that target ANSI 2.10 that will continue to target 2.10 targets, you should

not select UNICODE targets, or convert to 2.20 in the File menu, as projects targeting UNICODE 2.20 cannot be converted back to ANSI 2.10 (there is no mechanism, as data loss is likely).

Once built, results will be placed in a folder in the TARGET folder - See Global Settings. This Sub-folder will be named after the Project name in use. As each Target is built, a sub-folder under the Project name containing the Run-Time Target Program, and KBF Data will be created / updated. In all builds, there will be a TEST sub-folder created / updated that contains a Windows run-time version and the currently build KBF, so the layout can be quickly and easily tested from withing the Builder (F10, Build Menu | Run). The last built KBF will be created in the Project name folder.

For each build, the KBF will always be saved

as 2 distinct files - 1 named KEYBOARD.KBF, and 1 named as [Project Name].KBF (e.g. MyProject.KBF). The KEYBOARD.KBF is always created, as this is the default layout displayed if no other option is used when running My-T-Soft. When multiple layouts are integrated, having these distinct names resolves naming conflicts, while the KEYBOARD.KBF is the "as first displayed when run" layout (i.e. Default layout).

Board Tools

When building layouts, the Grid tools, and Layout / Alignment tools are the most helpful in providing capabilities to build well-organized layouts. See Panel Display and Cursor Tools and Menus & Settings.

Add Keys

Add New Key (Right-click | Add New Button, Edit | Add New Key, Insert key, Add Tool,

click & drag on empty board area, drag & drop from Keys window (F8)). For additional information, see working with the Panel Display.

Move & Size Keys

Click & Drag - Key area, moves key; frames, resizes key

Arrow keys - Move Key; with Ctrl & Shift, resizes frames

See Panel Display and Cursor Tools and Menus & Settings.

Modify Keys

Key area (Double-click, Right-click | Properties). See Key Properties for details.

Build & Execute

Build Board (Build menu | Build, Tool bar | Build, F9 key).

Execute My-T-Soft with Board layout from with Builder for testing (TARGET TEST Folder) (Build | Execute, Tool bar | Run, F10 key).

Typical Approach / Final Notes

Typically the board is modified, built, and then tested from within the Builder in an iterative process, up until the developer/user feels it is ready to deploy and test in its final/target environment. See the next section Run-Time Options for deployment and target options.

Run-Time Options

Run-Time Options

Once a Board is Built, there are various options available - you are always asked if you would like to Run My-T-Soft (to Test) after a

succesful build. The following discusses other options.

View Project Run-Time Targets Folder (Keyboard Shortcut = Shift-F11)

- This opens Windows Explorer and displays the Project folder in the Target area, which displays all built targets, and provides access to the built KBF file and available Targets for the current project.

Setup Output (ActiveSync) Folder

- This allows you to choose a folder to output the current build after the build is complete. This can be used as the ActiveSync synchronization folder to allow Build-A-Board to copy successfully built target files into (which allows ActiveSync to automatically copy to the target system).

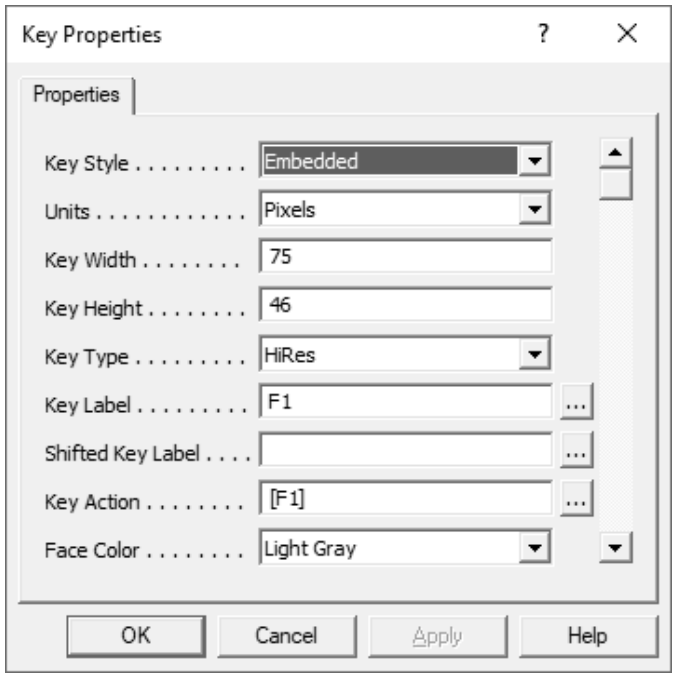
Output to My-T-Soft 1.xx Folder(s)

- This will be enabled if a suitable member of the My-T-Soft family 1.xx is installed (My-T-Soft / My-T-Touch / My-T-Pen / My-T-Mouse / OnScreen), and there is a built project KBF available. When selected, the [Project Name].KBF will be copied to all available product folders, and will be available from the Custom Build-A-Board keyboard selection option in Setup | Keyboards. You must be sure to target the correct version ANSI 2.10 (1.7x) or UNICODE 2.20 (1.8x +).

Deployment

Refer to the Run-Time Targets for notes and details on deploying Run-Time Targets on various platforms.

Build-A-Board Key Properties



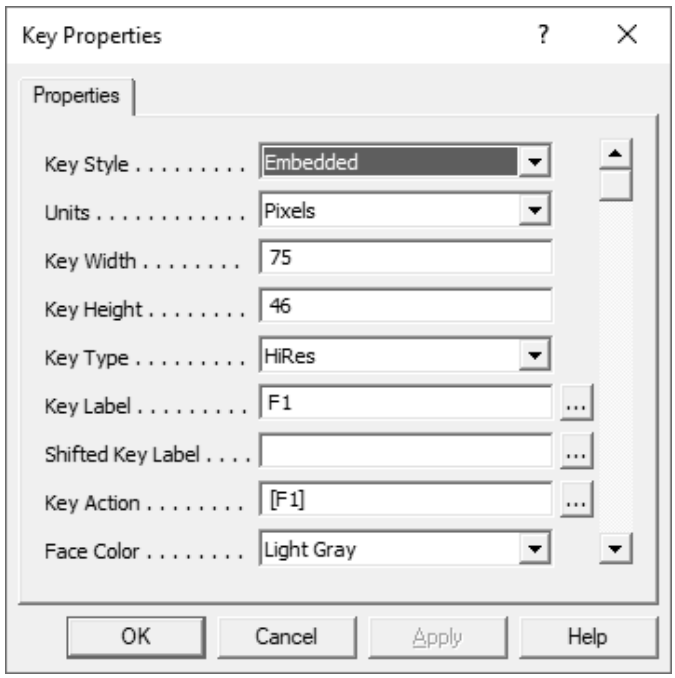
The Key Properties available depends on the project target, and the internal Mode (Level) of the Builder. Level 1 targets ANSI 2.10 KBFs, while Level 2 targets UNICODE 2.20 KBFs. This section is divided into 3 areas for addressing the different properties for each level, and the Key Actions available

Key Properties - Level 2 (UNICODE 2.20)

Key Properties - Level 1 (ANSI 2.10)

Key Properties - Key Actions

Key Properties - Level 2 (UNICODE 2.20)



The following lists each available Key Property. Important notes & limitations for the

Key Properties may only be included here.

Key Style

Currently the only Key Style in Build-A-Board is Embedded.

Units

Currently all measurements in Build-A-Board is in Pixels.

Key Width, Key Height

This specifies the Key Width & Key Height in Units.

Key Type

Supported types are Simple, Regular, & HiRes.

Simple will only convert the displayed Key Label into a Keystroke.

Regular can handle different actions, separate from the Key Label

HiRes uses a high-resolution background for a "Regular" key. HiRes is also required for Key Images / key backgrounds.

The Caps Aware identifier has been used in custom versions, and will be part of the Caps Lock solution. It is currently unused.

Key Label

This can be up to 63 characters long.

Built-In (Internal) Images

There is an override lable {IMG:*} that will display alternate images. Currently supported images (drawn via internally handled low-level graphic calls) are:

Label Image

{IMG:MIN} Minimize (Windows) bar

{IMG:A_UP} Up Arrow

{IMG:A_DN} Down Arrow

{IMG:A_LT} Left Arrow

{IMG:A_RT} Right Arrow

{IMG:A_DUP} Double Up Arrow

{IMG:A_DDN} Double Down Arrow

{IMG:A_DLT} Double Left Arrow

{IMG:A_DRT} Double Right Arrow

{IMG:BOXFL} Box, Filled

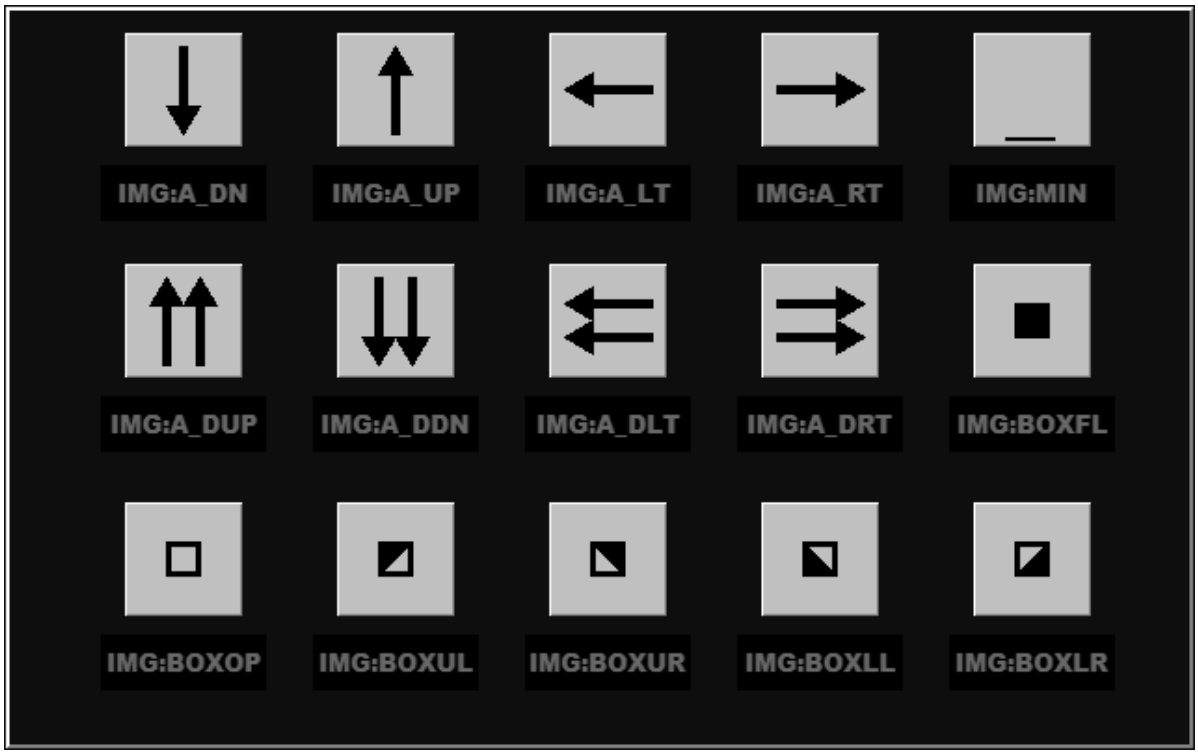
{IMG:BOXOP} Box, Open

{IMG:BOXUL} Box, Upper Left (Filled)

{IMG:BOXUR} Box, Upper Right (Filled)

{IMG:BOXLL} Box, Lower Left (Filled)

{IMG:BOXLR} Box, Lower Right (Filled)



External Images

When images are placed on Keys, the syntax used is {IMG:@ccccccc:nnnnn}, where the cccccc is a character based file name identifier, and the nnnnn is a 5 digit, 0 based image number (i.e. 00000, 00001, 00002, etc.). The actual number is internally calculated during a save/build, so the number after the : should not be modified. The name, e.g. {IMG:@somepic:00001} means there is an IMGsomepic.BMP in the Source folder, that

will be converted to a sized PNG, assigned an ID number, and then included in the KBF. While this could be done manually, it is recommended you simply use the "Place Image File" option, available in the right-click Key menu, or via the "..." option next to the Key Label. When selected, a system File Open dialog will be displayed at the Pictures location for the current user. When an image is selected (or dragged to the key via Windows Explorer), the file will be converted to a Bitmap, properly named, placed in the source folder, and the Key label will be automatically updated based on the above structure.

> **Panel Image Notes:** In 2.20, the background panel image requires an existing key image (and referenced within the MYTSOFT.INI, edited via the Window Properties Edit Keyboard Run-Time Settings). See notes in the file itself. For

best results, the image should be placed by dragging an image onto the panel - this will create a key off-board (to the right) at the same size as the panel, and then tag the MYTSOFT.INI with the image number. If necessary, resize the board (without resizing keys) to see this key container of the panel image. Note that this is a temporary work-around approach to the panel image for the 2.20 release, since only keys can arbitrarily carry image files in this release.

Shifted Key Label

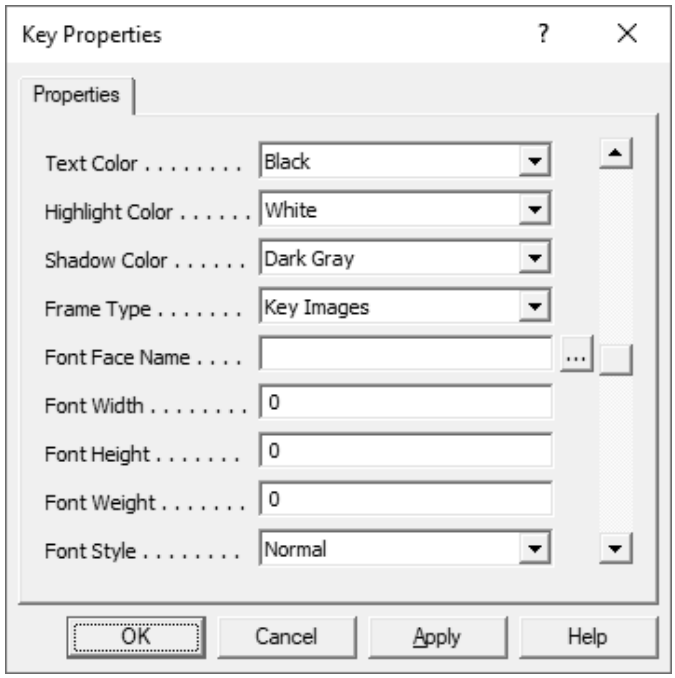
This is the image that will be displayed on the Key when the Shift state is triggered via a Key with a [Shift-Down] Action. If the Shifted Key Label is blank, the regular Key Label will remain when in the Shifted state. Also 63 characters long.

Key Action

For Regular (& HiRes) keys, this is the Macro command to generate Keystrokes. It can be up to 127 characters long. There is the Build-A-Board Macro Builder available by clicking on the Build button (...). - See Key Actions for commands.

Face Color, Text Color, Highlight Color, Shadow Color

There are 16 basic colors, and then the option to use custom colors for each element (24-bit color (8-bit Red, 8-bit Green, 8-bit Blue)).



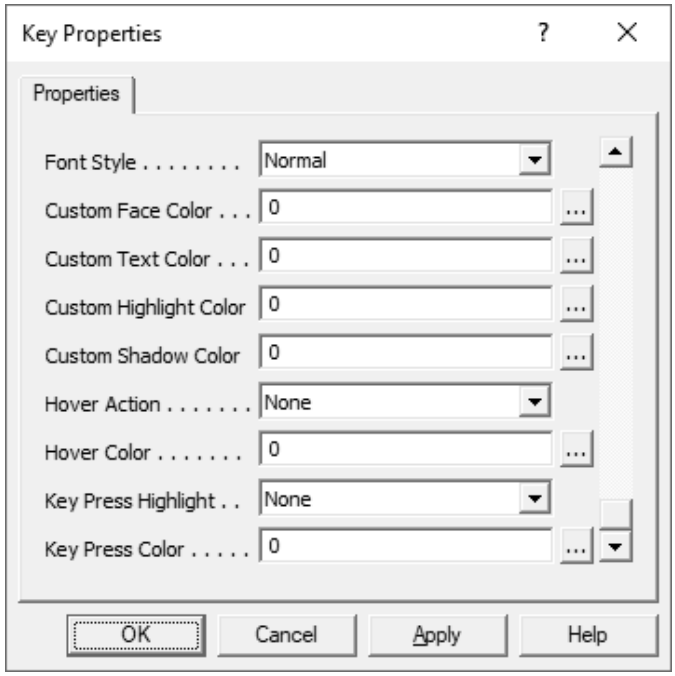
Frame Type

You can select None (no frame at all (all background), Standard (single pixel), and Thick (double pixel).

Font Face Name, Font Width, Font Height, Font Weight, Font Style

These are the font characteristics that define a Font per key - it is recommended you use the "..." button to use the Font dialog to easily select the font you wish, which will fill in these values - please be aware font support on the

Run-Time target may be limited - see Fonts.



Custom Face Color, Custom Text Color, Custom Highlight Color, Custom Shadow Color

When set, these are override colors to the basic Face Color, Text Color, etc. 0 means no custom color is in use. Values 1-15 are reserved for the basic colors. The representation is decimal, but the internal value is an RGB value of 0x00xxxxxx, where the RGB values are 0x00BBGGRR.

Hover Action, Hover Color

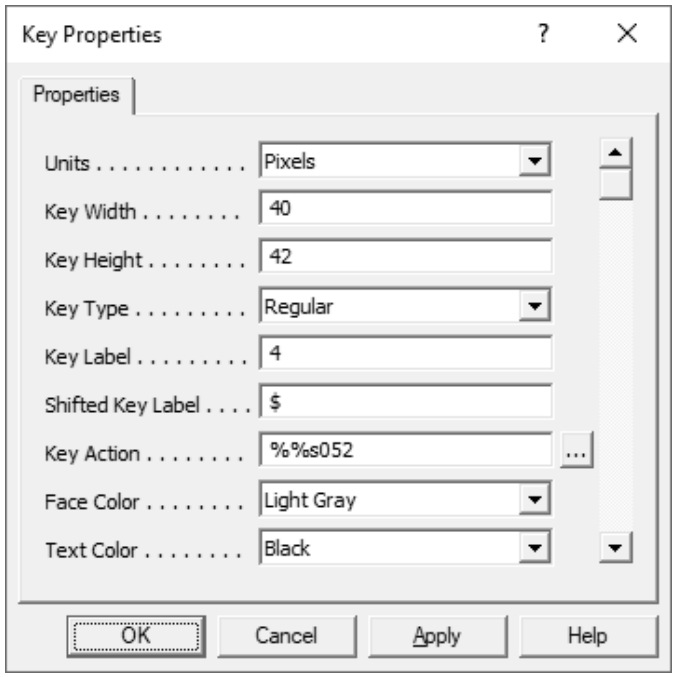
If Hover Action is None, the key will remain unchanged when the mouse pointer is over the key. If Hover Action is Highlight, the background will change to the Hover Color when the mouse pointer (Mouse Over) is over the key. IMPORTANT: This only affects Regular keys with color backgrounds - HiRes keys and image labels do not highlight. Windows targets, not implemented on all platforms.

Key Press Highlight, Key Press Color

If Key Press Highlight is None, the key color will remain unchanged when the key is pressed. If Short, Medium, or Long, the key will change color to the Key Press Color when pressed for a time based on the selection. IMPORTANT: This only affects Regular keys with color backgrounds - HiRes keys and image labels do not highlight. Windows

targets, not implemented on all platforms.

Key Properties - Level 1 (ANSI 2.10)



The following lists each available Key Property. Important notes & limitations for the Key Properties may only be included here.

Units

Currently all measurements in Build-A-Board is in Pixels.

Key Width, Key Height

This specifies the Key Width & Key Height in Units.

Key Type

Supported types are Simple, Regular, & HiRes.

Simple will only convert the displayed Key Label into a Keystroke.

Regular can handle different actions, separate from the Key Label

HiRes uses a high-resolution background for a "Regular" key.

Key Label

This can be up to 11 characters long.

There is an override lable {IMG:*} that will display alternate images. Currently supported

images (drawn via internally handled low-level graphic calls) are:

Label Image

{IMG:MIN} Minimize (Windows) bar

{IMG:A_UP} Up Arrow

{IMG:A_DN} Down Arrow

{IMG:A_LT} Left Arrow

{IMG:A_RT} Right Arrow

{IMG:A_DUP} Double Up Arrow

{IMG:A_DDN} Double Down Arrow

{IMG:A_DLT} Double Left Arrow

{IMG:A_DRT} Double Right Arrow

{IMG:BOXFL} Box, Filled

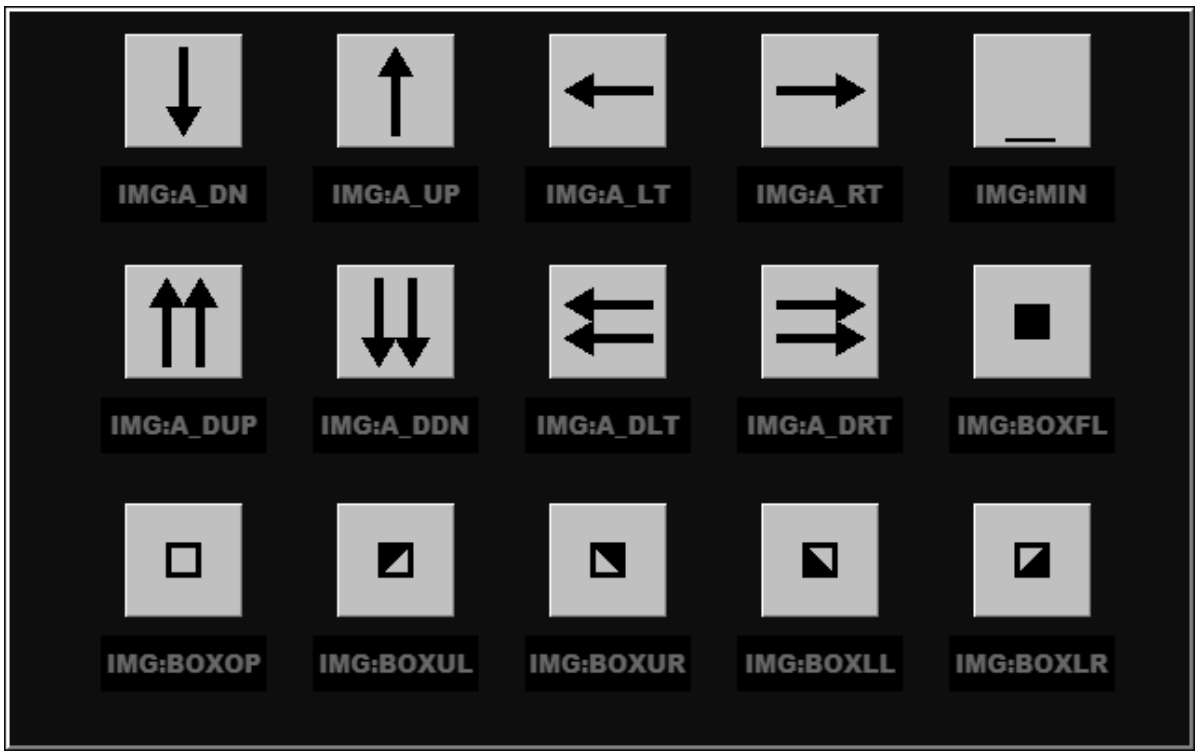
{IMG:BOXOP} Box, Open

{IMG:BOXUL} Box, Upper Left (Filled)

{IMG:BOXUR} Box, Upper Right (Filled)

{IMG:BOXLL} Box, Lower Left (Filled)

{IMG:BOXLR} Box, Lower Right (Filled)

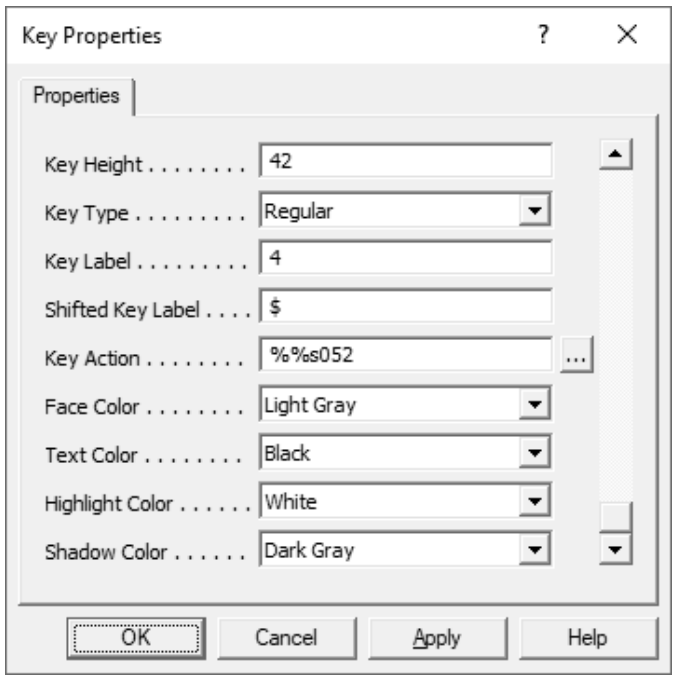


Shifted Key Label

This is the image that will be displayed on the Key when the Shift state is triggered via a Key with a [Shift-Down] Action. If the Shifted Key Label is blank, the regular Key Label will remain when in the Shifted state.

Key Action

For Regular (& HiRes) keys, this is the Macro command to generate Keystrokes. It can be up to 31 characters long. There is the Build-A-Board Macro Builder available by clicking on the Build button (...).



Face Color, Text Color, Highlight Color, Shadow Color

There are 16 basic colors supported.

Key Properties - Key Actions

These are the various commands available for the targets. Because each platform runs different code bases for each run-time target, support for a particular platform may be limited or not available. Check for updates, and with IMG technical support if there is a need beyond CLOSE, MINIMIZE, SAVEPOS, and NKBF on non-Windows platforms.

There are various commands (available in the Macro Builder):

[CMD:CLOSE] - When this is the Key Action, My-T-Soft will close. (Also shortcut **[CMD:CL]** Windows 32/64)

[CMD:MINIMIZE] - When this is the Key Action, My-T-Soft will Minimize to the Tray (Shell icon).(Also shortcut **[CMD:MN]** Windows 32/64)

[CMD:SAVEPOS] - When this is the Key Action, My-T-Soft will Save the current position to the KBF file (This will not work if the Window, Panel, Button, and Key file option is selected in Global Settings). Once the position is saved, the saved top, left setting will be used when this layout is opened in the future. (Also shortcut **[CMD:SP]** Windows 32/64)

[CMD:NKBF=???.KBF] - When this is the Key Action, the ??? must be replaced by a valid Keyboard File (KBF File). For Example, the Command might be [CMD:NKBF=NUM.KBF]. If there is a NUM.KBF file in the same folder as My-T-Soft, this will change the currently displayed keyboard layout to the specified "new" keyboard layout.

[CMD:LKBF=???.KBF] - When this is the Key Action, the ??? must be replaced by a

valid Keyboard File (KBF File). For Example, the Command might be [CMD:LKBF=NUM.KBF]. The distinction between NKBF (New KBF) and LKBF (Load KBF) depends on the target platform - for Windows 32/64, the Load manipulates the displayed layout window, and can overlay or position based on the current display - for full details, see this tech item: Tech Item PU2011120670 (http://www.imgpresents.com/imgfaq.htm?keywo

[CMD:HIDE] - When this is the Key Action, this will hide or close the keyboard layout (as if text input is no longer needed) (Android) (Windows CE / (Software Input Panel) SIP type interface only - the input panel (keyboard) will be hidden (removed from screen)).

[CMD:NEXTKMF] - When this is the Key Action, this will select the Next KMF file for Key Labels/Key Actions, if multiple KMFs

associated with KBF (when available for platform). (Also shortcut **[CMD:NK]** Windows 32/64)

[CMD:GETMYKBF] - When this is the Key Action, this will download favorites and defaults from the associated Build-A-Board.com Account (when licensed with a valid Build-A-Board.com Account). (Also shortcut **[CMD:GK]** Windows 32/64)

[CMD:GETMYBOARDS] - When this is the Key Action, this will download favorites and defaults from the associated Build-A-Board.com Account (when licensed with a valid Build-A-Board.com Account). (Also shortcut **[CMD:GB]** Windows 32/64)

[CMD:BOARDS] - When this is the Key Action, this will open the Select Boards screen (when available for platform). (Also shortcut **[CMD:BD]** Windows 32/64)

[CMD:EXEC=???.EXE] - When this is the Key Action, the ??? must be replaced by a valid executable file (Windows 32/64 Platforms). For Example, the Command might be [CMD:EXEC=NOTEPAD.EXE]. The following logic is used internally to run the named executable file: The current directory where the MYTSOFT.EXE is being run from is checked first - if the file is found, then it is run from that location. Otherwise, the file specified is handed off to Windows - Windows will search the the Windows folders, and then the path to find the file - if found, it will be run. If the file is not found, or there is an error, no other action will occur. The internal Windows API called is "ShellExecute", so a wide range of files can be run with this command - you can specify Bitmap files (.bmp), Documents (.doc), Shortcut files (.lnk) (which can contain URLs), etc. For command line entries, and other properties, use a shortcut, then specify the

shortcut as the EXEC file. Note that for Level 1 keys, there is a limit of 31 characters, so only a limited path could be used (e.g. [CMD:EXEC=C:\UTIL\ACTION.BAT]). When a long path is required, use a shortcut, and put the shortcut into the folder with MYTSOFT.EXE.

On the Android platform, the EXEC= must list a valid app package name or URL. Package names are typically a reverse domain type format (Java adopted this as a way to generate unique identifiers), so to launch the calendar as an example, you could specify [CMD:EXEC=com.android.calendar], or for Calculator, [CMD:EXEC=com.android.calculator2]. URLs specified should start with a http or https, so examples of this would be [CMD:EXEC=https://www.build-a-board.com] or [CMD:EXEC=http://mytsoft.com].

On the Linux platform, the EXEC= must list a valid executable. So an example of this to run the KDE Avanced Text Editor kate might be: [CMD:EXEC=/opt/kde3/bin/kate].

[CMD:TYPE=???.KMF] - When this is the Key Action, the ??? must be replaced by a valid text based Macro file File (KMF File). For Example, the Command might be [CMD:TYPE=KMF90000.KMF]. When the key action is processed, the KMF file will be read, and the text will be typed as a macro. This can enable extremely long text entries. The macro file must be placed in the source folder (e.g. the Public Documents / Build-A-Board / SOURCE / [Project] folder) so it will be included when the Target .KBF is built. (Windows 32/64)

[CMD:SND=???.WAV] - When this is the Key Action, the ??? must be replaced by a valid Wave file name. The following logic is

used internally to run the named wave file: The current directory where the MYTSOFT.EXE is being run from is checked first - if the file is found, then it is run from that location. Otherwise, it is handed off to Windows - Windows will search the Windows folders, and then the path to find the file - if found, it will be run. If the file is not found, or there is an error, no other action will occur. The internal Windows API called is "sndPlaySound" and any valid wave of multimedia file supported by that function will be played. Note that for Level 1 keys, there is a limit of 31 characters, so only a limited path could be used (e.g. [CMD:SND=C:\SNDS\ACTN.WAV]). (Windows 32/64)

[CMD:MIDI=???.MID] - When this is the Key Action, the ??? must be replaced by a valid MIDI file name. The current directory where the MYTSOFT.EXE is being run from

is checked for the file - if the file is found, then it is run from that location. If the file is not found, or there is an error, no other action will occur. The internal Windows API calls use the MCI commands with the MIDI device setup as the "sequencer". Note that for Level 1 keys, there is a limit of 31 characters, so only a limited path could be used (e.g. [CMD:MIDI=C:\SNDS\SQNC.MID]). (Windows 32/64)

[CMD:MIDI=STOP] - When this is the key action, the MCI command will be used to send a STOP command to the device previously used to Play a MIDI file (i.e. [CMD:MIDI=???.MID]). (Windows 32/64)

[CMD:SAY=Speak this] - this can use Text To Speech to echo text on a key press/key action (Windows 32/64)

[CMD:MACRO=Macro?] - this is used to reference up to 3 macros entered in the

Run-Time Settings (MYTSOFT.INI). [CMD:MACRO=Macro1] will then obtain the text of Macro1=Example Macro[Enter] from the Run-Time Settings, and type this macro, which will type "Example Macro" followed by an enter key. For reference, see the Run-Time Settings file Only Macro1, Macro2, Macro3 are supported. (Android/Windows)

[CMD:TOGGLEFORCESHOW] - this is an Android only command. As a key action, this enables/disables the current Force Show setting for Build-A-Board. If the Force Show feature is enabled, certain system actions are disabled, and it is possible to make the system unusable if the keyboard blocks all or key portions of the screen. By incorporating this Key Action on a layout button, the user can toggle the Force Show setting and revert to normal operation and re-enable when desired. (Android)

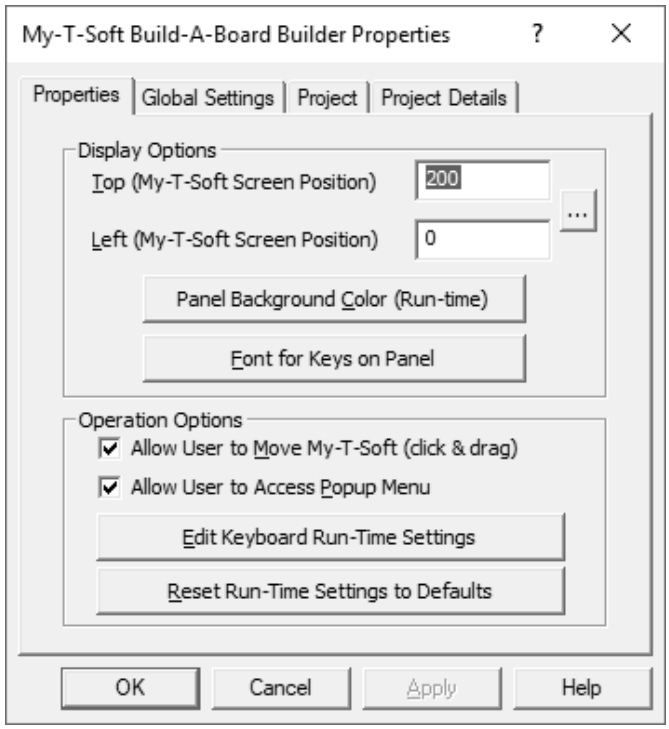
[CMD:KEYCODE=nnn] - this is an Android

only command. As a key action, this is supported as a way to generate virtual key codes outside of the normal text input process via the input method used by keyboards in Android. This approach is closer to what a physical keyboard would generate vs. the character approach of the input method interface. For function keys, arrow keys, etc. (i.e. non-character based keystrokes), this approach may be important for apps that are monitoring or looking for this type of input (vs. text field input characters). It is also important to note that certain keystrokes are managed internally, so a Key Action of [Left] will move the text caret one character to the left, where a [CMD:KEYCODE=21] will send a left arrow keystroke. The distinction is where the management of the user input is handled - in the Input Method, in the input field, or in the app itself. For a list of Android virtual key codes (so actual decimal values can be

determined), please reference the Android documentation here: Android KeyEvent KEYCODE reference (https://developer.android.com/reference/android/ The KEYCODE=nnn should be simple integer decimal values (e.g. [CMD:KEYCODE=22] or [CMD:KEYCODE=131]) There is also support for the older shorthand notation where %%k021 can be used a virtual key. For details on this notation, see Build-A-Macro Notes. (Android)

[KMF:??] - as a key action, this is a layer of abstraction of where the key action comes from. When the KMF entry is used, the currently selected KMF file (as included in the KBF file (Keyboard Layouts)) is referenced to obtain the Key Action. For Key Labels, the Image entry is {KMF:??} or {KMF:Shift-??}. Also see [CMD:NEXTKMF]. (Windows 32/64)

Build-A-Board Window Properties



Top, Left position of My-T-Soft

This setting changes the absolute screen position (in Pixels) for the Top-Left corner of the Panel displayed in the My-T-Soft window. For visual positioning and other position options, click on the ... button.

Panel Background Color (Run-Time)

This will open the default Color Selection Dialog Box and allow you to modify the Background color on the run-time version of My-T-Soft. Only the basic 16 colors (original IBM CGA Colors) are supported. If you select a different color, you will be warned, and the background color will be set to the default. The following table outlines the colors and their RGB (Red/Green/Blue) values of the supported colors in Version 2.00, 2.10, 2.20.

Color - R, G, B

Black - 0, 0, 0

Blue - 0, 0, 128

Green - 0, 128, 0

Red - 128, 0, 0

Cyan - 0, 128, 128

Magenta - 128, 0, 128

Brown - 128, 128, 0

Dark Gray - 128, 128, 128

Light Gray - 192, 192, 192

Light Blue - 0, 0, 255

Light Green - 0, 255, 0

Light Red - 255, 0, 0

Light Cyan - 0, 255, 255

Light Magenta - 255, 0, 255

Yellow - 255, 255, 0

White - 255, 255, 255

Font for Keys on Panel

You may set the internal, low-level settings that will interact with Windows to select a display font. This font will be used for all keys on the panel. The Common Font Selection Dialog is available to assist in selecting a font & and its

settings. Italics & other advanced font characteristics are not available.

Note that the fonts on the target system may not match the fonts on the system with the Build-A-Board Builder!

For projects that target 2.20 KBFs, fonts per key may be used (which will override the panel font) - see Key Properties - Level 2 (UNICODE 2.20)

Operation Options

Allow User to Move My-T-Soft (click & drag), if enabled, allows the end-user on the run-time keyboard to move the keyboard by clicking & dragging on any non-key portion of the panel.

Allow User to Access Popup Menu, if enabled (Windows targets), allows the end-user to open a menu to minimize, save, or close the My-T-Soft keyboard.

Edit Keyboard Run-Time Settings - this opens Notepad and allows run-time options to be set, such as the panel image. Refer to notes within the settings file. This gets embedded as MYTSOFT.INI into the built KBF (2.20 only).

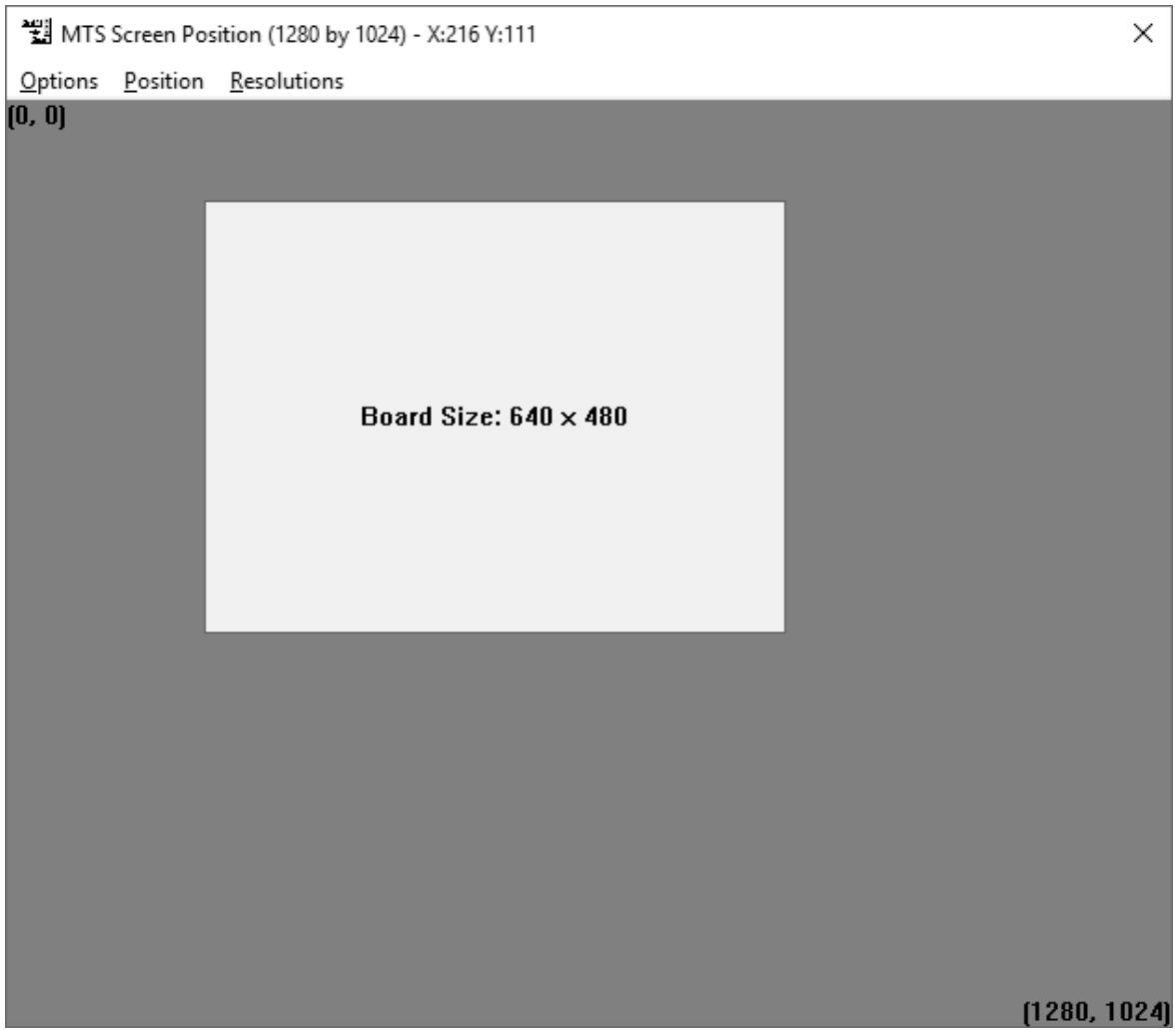
Advanced Settings in MYTSOFT.INI:
These settings cover user options / field configuration options that may need to be changed based on the user/system/situation and does not dictate rebuilding the KBF, along with other advanced settings. Some of these settings are special use, or specific to certain systems and situations. See MYTSOFT.INI Settings and Details in Advanced Notes for default file and to reference features and details.

Reset Run-Time Settings to Default - this will copy the current version default settings to the

project's MYTSOFT.TXT (embedded as MYTSOFT.INI) and overwrite any existing settings. This can be useful if older projects are being updated, but care must be taken in case there are custom settings in the existing file (you will be queried to confirm this operation).

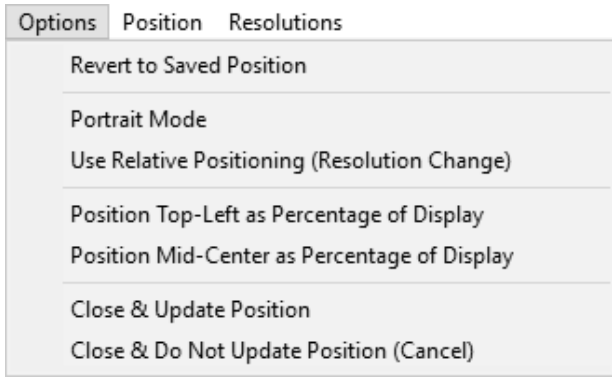
Screen Positioning Window (Top/Left)

When you click on the "..." next to the Top/Left position, a Screen Position window will open to provide a visual tool for setting the opening screen position. By default, the current screen resolution is used - for targets with different screen resolutions or layouts, use the Options and Resolutions menus.



In the Screen Position window, you can simply click & drag the white window (representation of keyboard layout relative to screen resolution) to the position on the screen where you want the keyboard layout to open. When you close the window, the current position will automatically update the Top, Left position in

the Window Properties dialog.



The Options menu provides various options available when working with the Screen Position window.

Revert to Saved Position

Selecting this will revert the layout to the original position (current position in the Window Properties settings).

Portrait Mode

This turns the screen 90 degrees, so the setting can be set for target systems that have a portrait oriented screen.

Use Relative Positioning (Resolution

Change)

This setting when enabled performs percentage calculation based on position relative to top/left of window - so the top/left of the layout is calculated and remains at the same relative position if you change resolutions while in the Screen Position window.

Position Top-Left as Percentage of Display

When this is selected, instead of using pixel position, the screen resolution will be represented as a percentage (with % symbol, e.g. 12%). This uses the Top-Left board location as the point where percentage display is used. Check with run-time target / platform support before using percentages.

Position Mid-Center as Percentage of Display

When this is selected, instead of using pixel position, the screen resolution will be

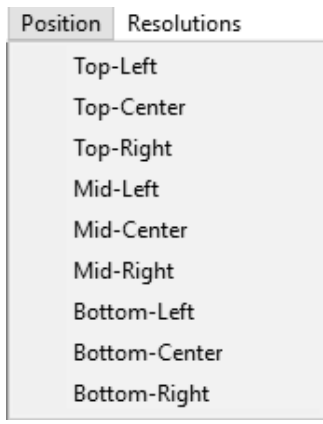
represented as a percentage - to indicate this is Mid-Center (vs. Top-Left), the %% notation will be used (e.g. 12%%). Check with run-time target / platform support before using percentages.

Close & Update Position

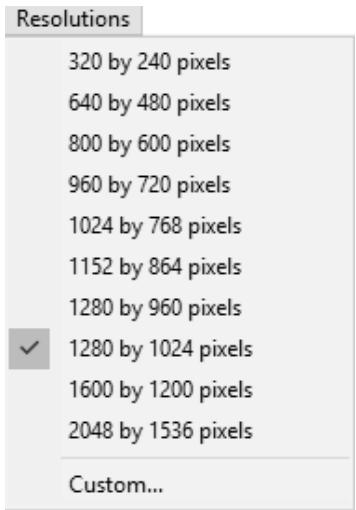
Closes the Screen Position window and updates the top / left values in the Window Properties settings.

Close & Do Not Update Position (Cancel)

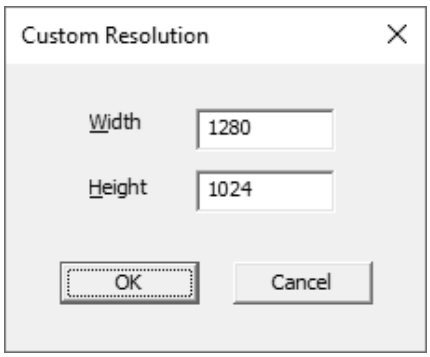
Closes the Screen Position window and does not change the top / left values in the Window Properties settings.



Select from the preset / most common screen positions.

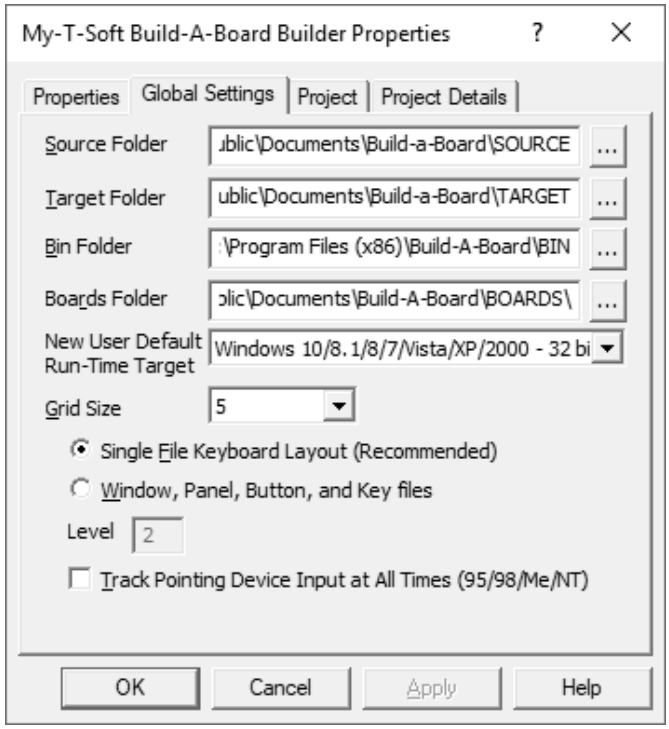


Select from the preset standard screen resolutions, or select Custom if your screen resolution is not available.



Custom - set your own custom screen resolution if there is not a preset available that matches your target screen resolution.

Build-A-Board Global Settings



Folder Locations

Source Folder - Location of Source files for all Panels. Any Projects/Panels you create will be stored in their source form here.

Target Folder - Location of Target files for all Panels. Any Builds will be created here under

the name of the Project (Panel), then the Target System subfolder. A "Test" set of files (Windows Win32 platform) will be location in the Project (Panel) folder, so you can view the Build results immediately (Build | Execute).

Bin Folder - Internal files for Build-A-Board

Boards Folder - Boards available for use on the Local System by run-time My-T-Soft

For flexibility, and future enhancements, the locations of the Build-A-Board files are included here. These are set during install as defaults, and should not be modified.

New User Default Run-Time Target

This sets the machine based default for the default Run-Time Target, and also defines the default Run-Time Target for New projects.

Grid Size

This lets you set the default Grid Size - note

that the most recent grid size will be saved when the Builder is closed. The Grid Size can be set via the Tools menu, Shift-F3/Shift-F4, or here.

Keyboard Layout Files

The 2 options are:

Single File Keyboard Layout

Window, Panel, Button, and Key files

The Single File Keyboard Layout is recommended for all uses of Build-A-Board. All of the layout information is stored in KEYBOARD.KBF.

The individual breakdown of the Window(s), Panel(s), Button(s) and Key(s) is supported for debugging and development purposes, but in use, this simply creates more small files, that will require more disk space than actually required. Use the Single File Keyboard Layout!

Level

In Versions 2.00, 2.10, only Level 1 was supported, while in Version 2.20 Level 2 and Level 1 are available. These levels control the size of the Keyboard Layout files - higher levels will result in larger files, but include more capabilities, such as graphic displays and support for longer macros. This is not a user-settable setting, as there are various issues that must occur to move between levels - in general, a project should stay within the level it was originally created - although conversion options are available on the File Menu.

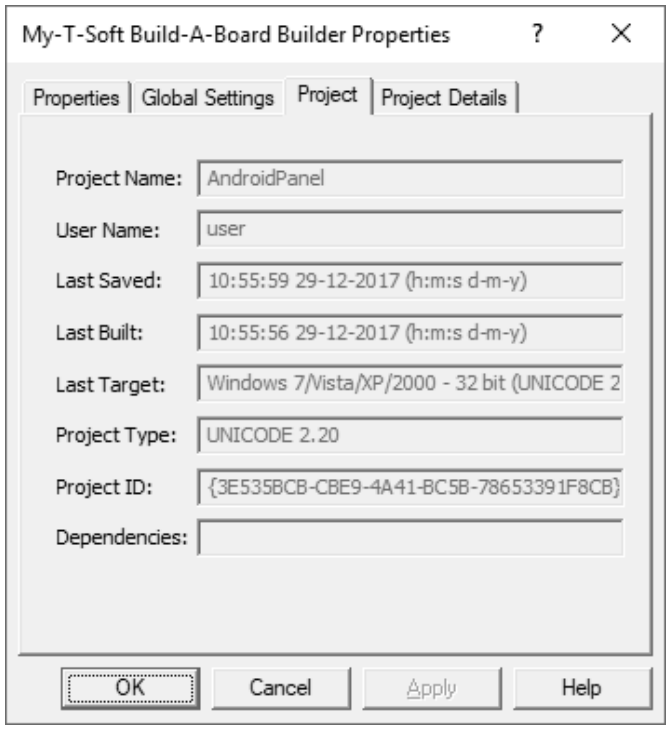
Track Pointing Device Input at All Times

This is essentially obsolete, but it remains for maintaining 2.10 projects and legacy support. For support of various touchscreen drivers (ELO, eTurboWare) in Windows 95/98/Me/NT this setting may need to be checked on. Certain drivers do not truly emulate a mouse pointer,

and you will find that the mouse will work with My-T-Soft, but the touchscreen will not. By setting this option On (and verifying that the touchscreen driver is in the correct mode (drag/double-click, Button mode, etc.), My-T-Soft will operate without changing the focus between the intended "type-into" application and the keyboard display.

> **IMPORTANT NOTE:** When using the Target system of Windows 95/98/Me/NT/2000/XP and creating Installation files via the Run-Time Menu, it is critical that the SETUP.EXE be used to install the run-time files. SETUP.EXE uses SETUP.INI to track this setting, and set the registry on the target system.

Build-A-Board Project Properties

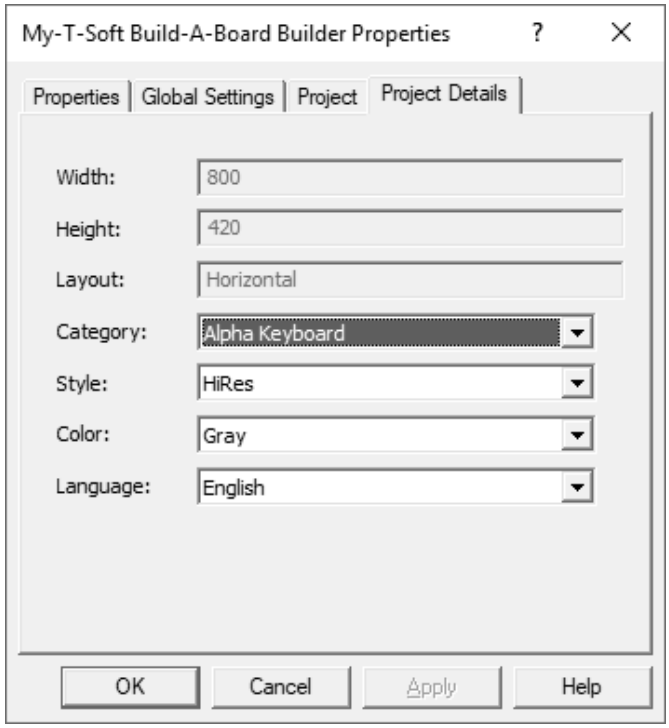


Project Properties

Various information about the current project is displayed on this tab.

Build-A-Board Project

Details Properties



Project Details Properties

The settings here are used in conjunction with the Build-A-Board online database to help categorize, sort, and manage any uploaded layouts. These settings should be set prior to uploading your project to the Build-A-Board.com.

Projects and Files

Projects are created in the Source folder (Global Settings (Property pages))

Compiled files are stored in the Target folder (Global Settings (Property pages))

Each Project is contained in a folder located within the Source folder.

A Closed project will contain a ZIP file named with the name of the project. The Zip file contains compressed files of an Open Project.

An Open Project contains the following files:

PROJECT.TXT

This file contains project information, and is required for Build-A-Board Builder to recognize the files as a valid Project.

The following text files are the "source" code for the boards created in each project. They are

stored as Text files so they are human readable outside of the Build-A-Board builder. In general, it is much easier & safer to edit the project using the Builder, but because these tools are to make life easier for people, the text files may be reviewed or edited directly. Please note that these files will be compiled prior to loading (opening a project). This ensures that they are properly understandable by the Builder.

My-T-Soft will only read compiled files - this is for 2 main reasons: 1) To ensure that the files are in the proper format, and 2) to reduce the target file size.

MWF00001.TXT

This is the My-T-Soft Window File - it contains information about the window size & position, and which panels are contained - it may contain multiple MPF?????.MPF files (Compiled from the MPF?????.TXT files).

MPF00001.TXT

These are the My-T-Soft Panel files - they contain information about the panel & which Button files are contained - there will be only 1 MBF?????.MBF file per panel (compiled from the MBF?????.TXT file).

MBF00001.TXT

This contains which Keys are associated with the button file

KEY00001.TXT

This contains information about the actual key (button) on My-T-Soft.

When Compiled, the following files may be seen:

MWF00001.MWF

This is a compiled MWF00001.TXT (My-T-Soft Window File)

MPF00001.MPF

This is a compiled MPF00001.TXT file (My-T-Soft Panel File)

MBF00001.MBF

This is a compiled MBF00001.TXT file (My-T-Soft Button File)

KEY00001.KEY

This is a compiled KEY00001.TXT file (Key file)

KEYBOARD.KBF

This is a single file containing MWF, MPF, MBF, and KEY file information.

The KEYBOARD.KBF is created in the Target folder, along with a ProjectName.KBF file (where ProjectName is the name of the current project (displayed in the Caption of the Build-A-Board Builder).

Additional File Notes (Advanced)

There are 3 folders (default \Program Files\Build-A-Board)

BIN Folder

See Files and Installation Information for more details on program and BIN files.

SOURCE Folder

The Source folder contains various Closed Projects in Sub-Folders for reference. Projects may be modified and saved as other projects if required. By default new projects are saved as sub-folders under this folder.

KEY1.TXT - This file contains the internal descriptions for Level 1 Keys & properties. It should not be modified.

KEY2.TXT - This file contains the internal descriptions for Level 2 Keys & properties. It should not be modified.

UNKNOWN.BMP - This file contains the default image for SOURCE project folders that do not have an associated bitmap image (for Project Display window).

TARGET Folder File Descriptions

The files within the Target folder are the Build-A-Board Test Files (files located in BIN\TEST).

Target Folder Notes

When a Project is built (Build | Build) in Build-A-Board, a folder of the same name as the Project is created in the Target folder. This folder gets a copy of the files in BIN\TEST, along with the result of the Build. Each selected Target (Run-Time | Select Target System), gets its own sub-folder under the Project's Target folder - upon a successful Build, the Target System sub-folder will get a copy of the appropriate sub-folder from BIN,

and the result of the Build (for that target system).

Since everything in the Target folder can be re-created at any time from the Projects in the Source folder, only the Source folder needs to be archived to ensure project integrity.

Production Notes: Since the Target folder's Project sub-folder contains the output used on the Target System, preserving the compiled boards & target system files may be important. There is no default mechanism in Build-A-Board to manage this aspect of the process. Note that every Build from the same project for the same selected target will overwrite any duplicate files in the Target folder's Project's Target System sub-folder.

Text Compiler

The Text Compiler / Edit window is the process that reads the source files and compiles them into binary files for maximum space conservation on the target system.

Technical Note: During the build process, each source file is compiled with a separate instance of the Text Compiler. Each process runs with a hidden window, unless there is a compile error. The separate process approach maximizes build speed when a multi-core or multi-processor system is available. However, system or file errors can result in numerous windows being shown in an error state due to this (typically hidden) aspect. If you experience this, be sure to check Global Properties (SOURCE path), system configuration (memory/hard drive space available), and project integrity

(have you done anything manual, or moved files, etc.?).

Keyboard Macro File (KMF) Notes

Keyboard Macro Files (KMF, or KMF Files) are data files that contain macros and key label information for use within the My-T-Soft family & Build-A-Board. One intended use of the Builder will be to modify and work with older (pre 2.00) KMF files, along with newer KMF files. In the current version, these editing capabilities are not available - KMF files can be opened and saved, and future capabilities will be added in later releases.

The KMF files can be thought of tables of data for use within the program. In the 2.00/2.10 versions, there are still distinct KMF files required for use along with the KBF. In 2.20 and later, the KMF files in use are included as part of the KBF file itself. For added information, See KBF Notes and KBF Architecture. The use of KMF files will include mapping tables from key scan codes and key symbols, along with override information and common keyboard macro data

Pre 2.00 KMF notes

In the Pre 2.00 version of My-T-Soft, KBF files defined the key and panel sizes, and KMF files defined the Key Labels and Key Actions. Refer to File Notes for the My-T-Soft family for further information. KEYBRD??.KMF files matched different keyboard layouts, and MAC?????.KMF matched macro panels.

KeyBoard File (KBF) Notes

KeyBoard Files (KBF, or KBF Files) are data files that define the on-screen keyboard layout and what is presented to the user. In the current version, key labels, key actions, configuration information, license information, and reference lookup data is also included in the KBF file. See KMF Notes and KBF Architecture for further details.

Pre 2.00 KBF notes

In the Pre 2.00 version of My-T-Soft, KBF files defined the key and panel sizes, and KMF files defined the Key Labels and Key Actions. The KBF itself defined the window size, button sizes, and panel configuration.

The actual rectangles that define the button layout for pre 2.00 KBF files can be modified using a special mode within the Builder (activated when an older KBF file is opened).

Only 1 panel at a time can be operated on, and Key Properties are limited to hiding keys, or resizing them - labels, etc. are based on the KMF files.

Macrobat Macro Reference

The My-T-Soft Macrobat process is the "Macro Batch Processor" that handles action requests from the user interface component (i.e. My-T-Soft). Full support is only available in a Windows system - for other platforms, many of the advanced options are not available.

Macrobat Notes

The Macrobat core is based on code that was part of the pre-2.00 releases of My-T-Soft. Because of this heritage, various advanced capabilities are available, some of which are

documented here. Not all supported pre-2.00 actions are available in 2.00 & later releases, and some capabilities may be added over time.

The Build-A-Macro operations, the following double-characters are reserved:

@@ - Signifies an Alt keystroke to follow

e.g. @@f = [Alt-Down]f[Alt-Up]

~~ - Signifies a Ctrl keystroke to follow

^^ - Signifies a Shift keystroke to follow

$$ - used internally for internal macro uses

%% - used to specify a virtual key, or a keyboard scan code

You may not use these character combinations in your macros, unless you use them as outlined. For example, you may quickly create a macro for File, New ([Alt]-F, N) by entering "@@fn" (do not include quote characters) and

clicking OK. However, using the Reserved words in brackets is the preferred method.

The %% sequence has 2 options, and must be formed correctly to be interpreted as a special entry. 4 characters must follow the 2 percent signs, spaces are not allowed. When this is used, it generates both the Down and Up keyboard messages, (press & release), similar to the entry of a specific character.

The following general form is: %%cnnn

where c is a character signifying Keystroke or Scancode -

The only valid characters for c are k or s or e (case does not matter) (NOTE: The k in post 2.00 releases should not be used, unless otherwise documented). The nnn must be 3 decimal digits (values between 000 and 999 are valid - for Virtual key codes, only values between 1-254 are valid).

The s and e directives are translated into Virtual key codes, and given to the Windows low-level API calls. Windows CE targets have various limitations, and not all options may be supported. The %%snnn configuration will generate a scan code / keyboard event where nnn is the decimal representation of a Windows Virtual Key (see below). The same applies for the e directive, but the e generates an extended keystroke. (If some or all of these terms are not familiar, then a good review of physical keyboards may be required. For explanations of extended characters, scan codes, reference books that describe the PC hardware, the 84, 101 and 104 key IBM compatible keyboards - good sources are Microsoft references, older Peter Norton books, and books about PC compatible hardware).

Not all virtual key codes are supported for all platforms. Modifier keys such as Ctrl/Shift/Alt,

and toggle keys such as Caps Lock / Num Lock / Scroll Lock keys create situations where the down / up resultant keystroke may not be sufficient for your needs (when using these low-level overrides).

Windows CE Notes: For certain keystrokes, use of the %%snnn may be required, especially in the OEM range and for keystrokes after VK_OEM_1.

As an example, the Fujitsu PenCentra Windows CE system will not map function keys (from a physical keyboard) to the 112-123 range, but use OEM (Reserved(!)) keys in the range 193-204 (e.g. to get the F1 key on My-T-Soft to act as the F1 on a physical keyboard use %%s193 - see WCE_KBRD project). The Suspend function uses VK_OEM_8 (%%s223).

For a particular unit, you will either have to contact the manufacturer for their

implementation of these "keystrokes", or test the various codes to see what the result is (and if supported).

Windows #define Hex Value Decimal Value Description

VK_LBUTTON 01 1 Left mouse button

VK_RBUTTON 02 2 Right mouse button

VK_CANCEL 03 3 Control-break processing

VK_MBUTTON 03 3 Middle mouse button (3-button mouse)

VK_XBUTTON1 05 5 Windows 2000: X1 mouse button

VK_XBUTTON2 06 6 Windows 2000: X2 mouse button

07 7 Undefined

VK_BACK 08 8 BACKSPACE key

VK_TAB0 09 9 TAB key

0A-0B 10-11 Reserved

VK_CLEAR 0C 12 CLEAR key

VK_RETURN 0D 13 ENTER key

0E-0F 14-15 Undefined

VK_SHIFT 10 16 SHIFT key (Latching key)

VK_CONTROL 11 17 CTRL key (Latching key)

VK_MENU 12 18 ALT key (Latching key / System key)

VK_PAUSE 13 19 PAUSE key

VK_CAPITAL 14 20 CAPS LOCK key

VK_KANA 15 21 IME Kana mode

VK_HANGUEL 15 21 IME Hanguel mode (maintained for compatibility; use VK_HANGUL)

VK_HANGUL 15 21 IME Hangul mode

16 22 Undefined

VK_JUNJA 17 23 IME Junja mode

VK_FINAL 18 24 IME final mode

VK_HANJA 19 25 IME Hanja mode

VK_KANJI 19 25 IME Kanji mode

1A 26 Undefined

VK_ESCAPE 1B 27 ESC key

VK_CONVERT 1C 28 IME convert

VK_NONCONVERT 1D 29 IME nonconvert

VK_ACCEPT 1E 30 IME accept

VK_MODECHANGE 1F 31 IME mode change request

VK_SPACE 20 32 SPACEBAR

VK_PRIOR 21 33 PAGE UP key

VK_NEXT 22 34 PAGE DOWN key

VK_END 23 35 END key

VK_HOME 24 36 HOME key

VK_LEFT 25 37 LEFT ARROW key

VK_UP 26 38 UP ARROW key

VK_RIGHT 27 39 RIGHT ARROW key

VK_DOWN 28 40 DOWN ARROW key

VK_SELECT 29 41 SELECT key

VK_PRINT 2A 42 PRINT key

VK_EXECUTE 2B 43 EXECUTE key

VK_SNAPSHOT 2C 44 PRINT SCREEN key

VK_INSERT 2D 45 INS key

VK_DELETE 2E 46 DEL key

VK_HELP 2F 47 HELP key

"0" (ANSI 0) 30 48 0 key

"1" (ANSI 1) 31 49 1 key

"2" (ANSI 2) 32 50 2 key

"3" (ANSI 3) 33 51 3 key

"4" (ANSI 4) 34 52 4 key

"5" (ANSI 5) 35 53 5 key

"6" (ANSI 6) 36 54 6 key

"7" (ANSI 7) 37 55 7 key

"8" (ANSI 8) 38 56 8 key

"9" (ANSI 9) 39 57 9 key

3A-40 58-64 Undefined

"A" (ANSI A) 41 65 A key

"B" (ANSI B) 42 66 B key

"C" (ANSI C) 43 67 C key

"D" (ANSI D) 44 68 D key

"E" (ANSI E) 45 69 E key

"F" (ANSI F) 46 70 F key

"G" (ANSI G) 47 71 G key

"H" (ANSI H) 48 72 H key

"I" (ANSI I) 49 73 I key

"J" (ANSI J) 4A 74 J key

"K" (ANSI K) 4B 75 K key

"L" (ANSI L) 4C 76 L key

"M" (ANSI M) 4D 77 M key

"N" (ANSI N) 4E 78 N key

"O" (ANSI O) 4F 79 O key

"P" (ANSI P) 50 80 P key

"Q" (ANSI Q) 51 81 Q key

"R" (ANSI R) 52 82 R key

"S" (ANSI S) 53 83 S key

"T" (ANSI T) 54 84 T key

"U" (ANSI U) 55 85 U key

"V" (ANSI V) 56 86 V key

"W" (ANSI W) 57 87 W key

"X" (ANSI X) 58 88 X key

"Y" (ANSI Y) 59 89 Y key

"Z" (ANSI Z) 5A 90 Z key

VK_LWIN 5B 91 Left Windows key (Natural keyboard)

VK_RWIN 5C 92 Right Windows key (Natural keyboard)

VK_APPS 5D 93 Applications key (Natural keyboard)

5E 94 Reserved

VK_SLEEP 5F 95 Computer Sleep key

VK_NUMPAD0 60 96 Numeric keypad 0 key

VK_NUMPAD1 61 97 Numeric keypad 1 key

VK_NUMPAD2 62 98 Numeric keypad 2 key

VK_NUMPAD3 63 99 Numeric keypad 3 key

VK_NUMPAD4 64 100 Numeric keypad 4 key

VK_NUMPAD5 65 101 Numeric keypad 5 key

VK_NUMPAD6 66 102 Numeric keypad 6 key

VK_NUMPAD7 67 103 Numeric keypad 7 key

VK_NUMPAD8 68 104 Numeric keypad 8 key

VK_NUMPAD9 69 105 Numeric keypad 9 key

VK_MULTIPLY 6A 106 Multiply key

VK_ADD 6B 107 Add key

VK_SEPARATOR 6C 108 Separator key

VK_SUBTRACT 6D 109 Subtract key

VK_DECIMAL 6E 110 Decimal key

VK_DIVIDE 6F 111 Divide key

VK_F1 70 112 F1 key

VK_F2 71 113 F2 key

VK_F3 72 114 F3 key

VK_F4 73 115 F4 key

VK_F5 74 116 F5 key

VK_F6 75 117 F6 key

VK_F7 76 118 F7 key

VK_F8 77 119 F8 key

VK_F9 78 120 F9 key

VK_F10 79 121 F10 key

VK_F11 7A 122 F11 key

VK_F12 7B 123 F12 key

VK_F13 7C 124 F13 key

VK_F14 7D 125 F14 key

VK_F15 7E 126 F15 key

VK_F16 7F 127 F16 key

VK_F17 80 128 F17 key

VK_F18 81 129 F18 key

VK_F19 82 130 F19 key

VK_F20 83 131 F20 key

VK_F21 84 132 F21 key

VK_F22 85 133 F22 key

VK_F23 86 134 F23 key

VK_F24 87 135 F24 key

88-8F 136-143 Unassigned

VK_NUMLOCK 90 144 NUM LOCK key

VK_SCROLL 91 145 SCROLL LOCK key

92-96 146-150 OEM specific

97-9F 151-159 Unassigned

VK_LSHIFT A0 160 Left SHIFT key

VK_RSHIFT A1 161 Right SHIFT key

VK_LCONTROL A2 162 Left CONTROL key

VK_RCONTROL A3 163 Right CONTROL key

VK_LMENU A4 164 Left MENU key

VK_RMENU A5 165 Right MENU key

VK_BROWSER_BACK A6 166 Windows 2000: Browser Back key

VK_BROWSER_FORWARD A7 167 Windows 2000: Browser Forward key

VK_BROWSER_REFRESH A8 168 Windows 2000: Browser Refresh key

VK_BROWSER_STOP A9 169 Windows 2000: Browser Stop key

VK_BROWSER_SEARCH AA 170 Windows 2000: Browser Search key

VK_BROWSER_FAVORITES AB 171 Windows 2000: Browser Favorites key

VK_BROWSER_HOME AC 172 Windows 2000: Browser Start and Home key

VK_VOLUME_MUTE AD 173 Windows 2000: Volume Mute key

VK_VOLUME_DOWN AE 174 Windows 2000: Volume Down key

VK_VOLUME_UP AF 175 Windows 2000: Volume Up key

VK_MEDIA_NEXT_TRACK B0 176 Windows 2000: Next Track key

VK_MEDIA_PREV_TRACK B1 177 Windows 2000: Previous Track key

VK_MEDIA_STOP B2 178 Windows 2000: Stop Media key

VK_MEDIA_PLAY_PAUSE B3 179 Windows 2000: Play/Pause Media key

VK_LAUNCH_MAIL B4 180 Windows 2000: Start Mail key

VK_LAUNCH_MEDIA_SELECT B5 181 Windows 2000: Select Media key

VK_LAUNCH_APP1 B6 182 Windows 2000: Start Application 1 key

VK_LAUNCH_APP2 B7 183 Windows 2000: Start Application 2 key

B8-B9 174-185 Reserved

VK_OEM_1 BA 186 Windows 2000: For the US standard keyboard, the ’;:’ key

VK_OEM_PLUS BB 187 Windows 2000: For any country/region, the ’+’ key

VK_OEM_COMMA BC 188 Windows 2000: For any country/region, the ’,’ key

VK_OEM_MINUS BD 189 Windows 2000: For any country/region, the ’-’ key

VK_OEM_PERIOD BE 190 Windows 2000: For any country/region, the ’.’ key

VK_OEM_2 BF 191 Windows 2000: For the US standard keyboard, the ’/?’ key

VK_OEM_3 C0 192 Windows 2000: For the US standard keyboard, the ’‘~’ key

C1-D7 193-215 Reserved

D8-DA 216-218 Unassigned

VK_OEM_4 DB 219 Windows 2000: For the US standard keyboard, the ’[{’ key

VK_OEM_5 DC 220 Windows 2000: For the US standard keyboard, the ’\|’ key

VK_OEM_6 DD 221 Windows 2000: For the US standard keyboard, the ’]}’ key

VK_OEM_7 DE 222 Windows 2000: For the US standard keyboard, the ’single-quote/double-quote’ key

VK_OEM_8 DF 223 OEM specific

E0 224 Reserved

E1 225 OEM specific

VK_OEM_102 E2 226 Windows 2000: Either the angle bracket key or the backslash key on the RT 102-key keyboard

E3-E4 227-228 OEM specific

VK_PROCESSKEY E5 229 Windows 95/98, Windows NT 4.0, Windows 2000: IME PROCESS key

E6 230 OEM specific

VK_PACKET E7 231 Windows 2000: Used to pass Unicode characters as if they were keystrokes. The VK_PACKET key is the low word of a 32-bit Virtual Key value used for non-keyboard input methods. For more information, see Remark in KEYBDINPUT, SendInput, WM_KEYDOWN, and WM_KEYUP

E8 232 Unassigned

E9-F5 233-245 OEM specific

VK_ATTN F6 246 Attn key

VK_CRSEL F7 247 CrSel key

VK_EXSEL F8 248 ExSel key

VK_EREOF F9 249 Erase EOF key

VK_PLAY FA 250 Play key

VK_ZOOM FB 251 Zoom key

VK_NONAME FC 252 Reserved for future use

VK_PA1 FD 253 PA1 key

VK_OEM_CLEAR FE 254 Clear key

FF 255 Reserved

Part III. Build-A-Board Run-Time Targets

Description of Build-A-Board Run-Time Targets and Platform

Operation notes.

Details and reference information about Build-A-Board Run-Time Targets.

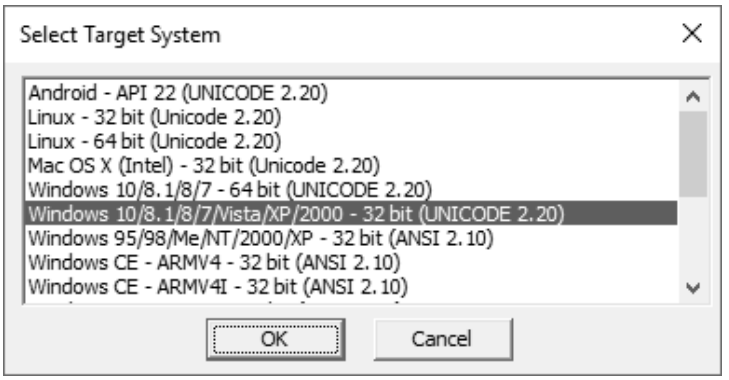
Chapter 5 - Build-A-Board Run-Time Targets contains general operation information, along with specific information about each of Build-A-Board's Run-Time Targets.

Chapter 6 - Build-A-Board Run-Time Targets Notes additional information, configuration options, and advanced notes on Build-A-Board Run-Time Targets.

Chapter 5. Build-A-Board Run-Time Targets

Run-Time Targets Overview

Build-A-Board Run-Time Targets Overview



Overview

Build-A-Board was designed from the ground-up to be a cross-platform, multiple

target custom keyboard design tool. The Build-A-Board Builder is the developer front-end that can manipulate keyboard layouts, look, and features. The Run-Time Targets are the actual software (Program) that runs on a target system to display and operate the keyboard layout (Data). Due to the various nature and aspects of the different run-time operating systems and environments, there are features and capabilities that may not translate from one system to another. Also, with over 30 years of providing keyboard solutions, and a half-dozen keyboard file data formats, there are also other constraints that may affect a particular target.

The actual code-base for the run-time targets is a combination of target specific code, shared core keyboard software code, and shared lower-level abstracted operating system API code. Because the foundation was designed to

be cross-platform, and extensible, the ability to run the same keyboard layout (Data) on multiple run-time targets (Programs) is the expected operation. However, due to a wide range of limitations & potential cross-platform compatibility issues, this cannot be guaranteed. Various aspects that may affect a particular run-time target will be addressed in this chapter and the next.

Available targets

Windows 95/98/Me/NT/2000/XP - 32 bit (ANSI 2.10)

Windows 2000/XP/Vista/7 - 32 bit (UNICODE 2.20)

Windows 11/10/8.1/8/7/Vista - 32 bit (UNICODE 2.20)

Windows 11/10/8.1/8/7/Vista - 64 bit (UNICODE 2.20)

Windows CE - SIP (Software Input Panel) and Free-Floating keyboard interfaces (ANSI 2.10)

Android - System Input Method (UNICODE 2.20)

Linux - 32 bit (UNICODE 2.20)

Linux - 64 bit (UNICODE 2.20)

UNIX - 32 bit (UNICODE 2.20)

Mac OS X (10.4 and higher) - 32 bit (UNICODE 2.20)

2.00, 2.10, and 2.20 Notes

Build-A-Board 2.00 was updated to 2.10, and no customer should be running any KBF (KeyBoard Files) (Data) from 2.00. The 2.00 version is not supported.

KBF Files based on the ANSI 2.10 data format can run under the My-T-Soft Family 1.7x/1.8x versions, and the older My-T-Soft 2.10 Run-Time software. Because there are multiple

hundreds of thousands (est. 250,000+) installs throughout the world, the 2.10 data format is supported in 2.20 as a Run-Time target option. In other words, Build-A-Board 2.20 can still generate 2.10 KBF files for these older, already installed systems. The license for these systems are handled by the run-time software. 2.10 KBF files do not have any license information within them.

KBF Files based on the UNICODE 2.20 data format are the preferred format for any new installs, or cross-platfrom layouts. The UNICODE support provides a much better way to handle key labels for the world's locales, and the image support, and extensibility of the expanded data format is also a marked improvement from the older data formats. Note that the 2.20 also supports embedded licensing, so a properly licensed Build-A-Board Builder can generate "licensed"

KBF files for licensed platforms. This license approach makes for easier deployments, especially when multiple targets are in use.

Platform Notes

It is important to remember that each platform (i.e. Windows, Windows CE, Android, Linux, Mac OS X, UNIX, etc.) has its own implementation, and features available in one platform may not be available in another, or have different operation, or cause different results from the same KBF. Syntax for file paths and file locations are also variable, so platform specific KBFs may be required.

The internal design of the software results in many aspects sharing the same source code, but due to specific implementation of input and interaction with the system, it is often the case

that platform specific code is used to implement the working end-result. Because of this, there is a high degree of probability that certain features, capabilities, or results may differ on different platforms. It is highly recommended that if you experience something that isn't what you expect, that you refer to the Run-Time Target documentation for the specific platform to review any specific notes for the command, label, or action issue you are experiencing - it is possible that the specific issue is noted for the platform. If you cannot resolve the issue with this help document, the platform README file, or on-line support info, please contact IMG technical support for assistance.

> **Note:** The goal of Build-A-Board is to provide a seamless, cross-platform way to build & deploy keyboards and user-interface components, and for each

supported label and action, to have similar results. However, due to the varied operating systems, font support, differing design philosophies, differing priorities & goals, along with constant revisions to the platforms, all of which is outside of the control of IMG, this goal is virtually impossible to achieve. In order to be responsive to specific needs and requirements, all users and distributors should be on IMG's annual support, annual maintenance, or under a license agreement so that any particular inconsistency or issue can be addressed in a timely fashion.

My-T-Soft® Macrobat (Macro

Batch server)

The My-T-Soft Macrobat process is the "Macro Batch Processor" that handles action requests from the user interface component (i.e. My-T-Soft). Full support is only available in a Windows system - for Windows CE versions, many of the advanced options are not available.

Macrobat Notes

The Macrobat core is based on code that was part of the pre-2.00 releases of My-T-Soft. Because of this heritage, various advanced capabilities are available, some of which are documented here. Not all supported pre-2.00 actions are available in 2.00 & later releases, and some capabilities may be added over time.

My-T-Soft®

My-T-Soft®is My Typing Software. The original on-screen keyboard for Windows was My-T-Mouse®, and the history of innovative approaches to user-interfaces continues. The My-T-Soft® component takes the compiled output from the design tool and presents the display to the user.

Target System support for My-T-Soft includes:

Windows 95/98/Me/NT/2000/XP - 32 bit (ANSI 2.10)

Windows 2000/XP/Vista/7 - 32 bit (UNICODE 2.20)

Windows 11/10/8.1/8/7/Vista - 32 bit (UNICODE 2.20)

Windows 11/10/8.1/8/7/Vista - 64 bit (UNICODE 2.20)

Windows CE - SIP (Software Input Panel) and Free-Floating keyboard interfaces

- Windows CE - ARM - 32 bit (various)
- Windows CE - MIPS - 32 bit (various)
- Windows CE - SH3 - 32 bit
- Windows CE - SH4 - 32 bit
- Windows CE - x86 - 32 bit

Android 10+ - API 29 (UNICODE 2.20)

Linux - 32 bit (UNICODE 2.20)

Linux - 64 bit (UNICODE 2.20)

- Linux - GNOME and KDE interfaces (and others)
- Linux Distribution support: Fedora
- Linux Distribution support: Red Hat
- Linux Distribution support: Debian

- Linux Distribution support: Ubuntu
- Linux Distribution support: LinuxMint
- Linux Distribution support: PC Linux OS
- Linux Distribution support: SuSE
- Linux Distribution support: Many others

Mac OS X (10.4 and higher) - 32 bit (UNICODE 2.20)

As a User Interface component, there are 2 main aspects incorporated in the Run-Time My-T-Soft - the visual contents of the button(s) & My-T-Soft window(s) displayed to the user, and the actions initiated once the user clicks upon a button.

Windows

Build-A-Board Run-Time Target: Windows

Notes on Run-Time versions available as Products

The installable / packaged as a product versions include the original My-T-Soft for Windows (My-T-Soft Professional) and also My-T-Soft Basic. These can be installed & licensed on Windows as complete self-contained components, and may be preferable to some customers.

My-T-Soft for Windows (My-T-Soft Professional) has supported Build-A-Board run-time layouts since version 1.76 in 2002. This is a full-featured run-time target, and provides additional options beyond just running custom boards, including logon options, etc. For details and help on this run-time target, see My-T-Soft (http://www.imgpresents.com/mytsoft/mytsoft.ht for product infor-
mation and My-T-Soft for Windows User Guide

(http://www.imgpresents.com/mytsoft/guide/html
for product help and documentation.

My-T-Soft Basic was created as an installable product that provides full support for custom boards, but no legacy options. Because per device licensing is included, for some customers this is the preferred option when deploying custom boards in small quantities. For details and help on this run-time target, see My-T-Soft Basic (http://www.imgpresents.com/mtsbasic/mtsbasic. for product information and My-T-Soft Basic User Guide (http://www.imgpresents.com/mtsbasic/guide/htm for product help and documentation.

The following details are for the My-T-Soft options included with Build-A-Board - these are meant for Platform licensing and customer controlled deployments.

Windows 95/98/Me/NT/2000/XP - 32 bit

(ANSI 2.10)

Target Files

MYTSOFT.EXE

This is the My-T-Soft executable that reads the KEYBOARD.KBF layout (or multiple file layouts), and displays the window to the user. If a shortcut is used, separate KBF files may be referenced via the command line (e.g. MYTSOFT.EXE NUM.KBF will run My-T-Soft with the NUM.KBF layout). The opening screen position is controlled by Build-A-Board builder. A right-click will open a menu with Minimize, Save Position, or Close. If a touchscreen or pen is in use without right-click ability (see IMG's TouchRight Utilities!), dragging the window to a position where any part of the window is off-screen will also open the menu.

MYTSOFT may be passed a command line

parameter to open a specific keyboard layout (????????.KBF file).

MTSLIB.DLL

This is a Support Dynamic Link library for Build-A-Board My-T-Soft.

BABDLL.DLL

This is a Support Dynamic Link library for Build-A-Board

IMGVERS.DLL

This is a Support Dynamic Link library for Build-A-Board

Windows Version tracking for support of appropriate features.

STOCK.DLL

This is a Dynamic Link Library of Images (HiRes images).

MACROBAT.EXE

This is the Macro Batch Processor. It is the support process for managing inter-process communication and handling low-level interface issues.

MAC00000.KMF

KEYBRD01.KMF

These are support & low-level interface files for MacroBat

KEYBOARD.KBF

WELCOME.KBF

The KBF file is the default display file for the My-T-Soft Board (from Build-A-Board). Any other KBF files are projects that have been built - With each successful build, a KEYBOARD.KBF and a ProjectName.KBF file are created in the Target folder.

Windows 2000/XP/Vista/7 - 32 bit (UNICODE 2.20)

Target Files

MYTSOFT.EXE

This is the My-T-Soft executable that reads the KEYBOARD.KBF layout (or multiple file layouts), and displays the window to the user. If a shortcut is used, separate KBF files may be referenced via the command line (e.g. MYTSOFT.EXE NUM.KBF will run My-T-Soft with the NUM.KBF layout). The opening screen position is controlled by Build-A-Board builder. A right-click will open a menu with Minimize, Save Position, or Close. If a touchscreen or pen is in use without right-click ability (see IMG's TouchRight Utilities!), dragging the window to a position where any part of the window is off-screen will also open the menu.

MYTSOFT may be passed a command line parameter to open a specific keyboard layout (????????.KBF file).

MACROBAT.EXE

This is the Macro Batch Processor. It is the support process for managing inter-process communication and handling low-level interface issues.

GDIKIT.DLL

This is a Dynamic Link Library of for interfacing with GDIPlus.DLL to handle Images.

KEYBOARD.KBF

WELCOME.KBF

The KBF file is the default display file for the My-T-Soft Board (from Build-A-Board). Any other KBF files are projects that have been built - With each successful build, a KEYBOARD.KBF and a ProjectName.KBF file are created in the Target folder.

Windows 11/10/8.1/8/7/Vista - 32 bit

(UNICODE 2.20)

Target Files

MYTSOFT.EXE

This is the My-T-Soft executable that reads the KEYBOARD.KBF layout (or multiple file layouts), and displays the window to the user. If a shortcut is used, separate KBF files may be referenced via the command line (e.g. MYTSOFT.EXE NUM.KBF will run My-T-Soft with the NUM.KBF layout). The opening screen position is controlled by Build-A-Board builder. A right-click will open a menu with Minimize, Save Position, or Close. If a touchscreen or pen is in use without right-click ability (see IMG's TouchRight Utilities!), dragging the window to a position where any part of the window is off-screen will also open the menu.

MYTSOFT may be passed a command line

parameter to open a specific keyboard layout (????????.KBF file).

MACROBAT.EXE

This is the Macro Batch Processor. It is the support process for managing inter-process communication and handling low-level interface issues.

GDIKIT.DLL

This is a Dynamic Link Library of for interfacing with GDIPlus.DLL to handle Images.

KEYBOARD.KBF

WELCOME.KBF

The KBF file is the default display file for the My-T-Soft Board (from Build-A-Board). Any other KBF files are projects that have been built - With each successful build, a KEYBOARD.KBF and a ProjectName.KBF

file are created in the Target folder.

Windows 11/10/8.1/8/7/Vista - 64 bit (UNICODE 2.20)

Target Files

MYTSOFT.EXE

This is the My-T-Soft executable that reads the KEYBOARD.KBF layout (or multiple file layouts), and displays the window to the user. If a shortcut is used, separate KBF files may be referenced via the command line (e.g. MYTSOFT.EXE NUM.KBF will run My-T-Soft with the NUM.KBF layout). The opening screen position is controlled by Build-A-Board builder. A right-click will open a menu with Minimize, Save Position, or Close. If a touchscreen or pen is in use without right-click ability (see IMG's TouchRight Utilities!), dragging the window to a position where any part of the window is off-screen will

also open the menu.

MYTSOFT may be passed a command line parameter to open a specific keyboard layout (????????.KBF file).

MACROBAT.EXE

This is the Macro Batch Processor. It is the support process for managing inter-process communication and handling low-level interface issues.

GDIKIT.DLL

This is a Dynamic Link Library of for interfacing with GDIPlus.DLL to handle Images.

KEYBOARD.KBF

WELCOME.KBF

The KBF file is the default display file for the My-T-Soft Board (from Build-A-Board). Any other KBF files are projects that have been

built - With each successful build, a KEYBOARD.KBF and a ProjectName.KBF file are created in the Target folder.

Windows CE

Build-A-Board Run-Time Target: Windows CE

Due to the nature of embedded and OEM Windows CE, Windows CE deployments are done either with the available pre-built run-times, or custom versions built directly for the customer (with some based on the customer's SDK as created specifically for the device). So it is often beneficial to discuss your particular needs with IMG.

Windows CE - SIP (Software Input Panel) and Free-Floating keyboard interfaces (ANSI 2.10)

Target Files

The .CAB file is an installation file set that carries the system interface DLL, the Option dialog, and the Keyboards. The SIP interface DLL is installed in the Windows folder, and the other supporting files are in \Program Files\My-T-Soft SIP and \Program Files\My-T-Soft SIP\KEYBOARDS.

Windows CE - 32 bit (ARM, MIPS, x86, etc.) (ANSI 2.10)

Target Files

MYTSOFT.EXE

This is the My-T-Soft executable that reads the KEYBOARD.KBF layout (or multiple file layouts), and displays the window to the user.

MYTSOFT may be passed a command line parameter to open a specific keyboard layout (????????.KBF file).

MAC00000.KMF

KEYBRD01.KMF

These are support & low-level interface files for MacroBat

KEYBOARD.KBF

WELCOME.KBF

The KBF file is the default display file for the My-T-Soft Board (from Build-A-Board). Any other KBF files are projects that have been built - With each successful build, a KEYBOARD.KBF and a ProjectName.KBF file are created in the Target folder.

Android

Build-A-Board Run-Time Target: Android

The base keyboard interface in Android is built around an Input Method, and integrates with the system / Android platform in a specific way. Due to the compiled Java byte code type approach for this interface, much of the platform code had to be refactored to operate in this environment, and as such, specific feature support had to be addressed individually. Refer to current Android version support & features to identify capabilities available on the Android platform.

Android (InputMethodService / API 29) (UNICODE 2.20)

Target Files

The .APK file is an installation file set that carries the system interface and the Keyboards. For most up-to-date feature list and support, refer to the features and technical information at the
product/download page: My-T-Soft for Android

(http://www.imgpresents.com/bldabrd/babandroi

Only the most recent Android build is included with the install release. For other available Android targets, please see the download page above, or use the Help | Check for Updates... option.

Linux

Build-A-Board Run-Time Target: Linux

The run-time Linux files are built on 2 different targets - 32-bit, and 64-bit. If you need another target type, please contact IMG Technical Support.

Linux - 32 bit (UNICODE 2.20)

Linux - 64 bit (UNICODE 2.20)

Target Files

mytsoft

This is the My-T-Soft executable that reads the KEYBOARD.KBF layout (or multiple file layouts), and displays the window to the user.

macrobat

This is the Macro Batch Processor. It is the support process for managing inter-process communication and handling low-level interface issues.

KEYBOARD.KBF

WELCOME.KBF

The KBF file is the default display file for the My-T-Soft Board (from Build-A-Board). Any other KBF files are projects that have been built - With each successful build, a KEYBOARD.KBF and a ProjectName.KBF file are created in the Target folder.

Deploying My-T-Soft Build-A-Board custom

layouts onto Linux

There are 2 aspects - the My-T-Soft software (program), and then the custom KBF file (data).

Step 1: Get built KBF (& install files)

From Build-A-Board, after you have designed, then Built the layout, you can go to the Run-time menu | View Project Run-Time Targets folder (or use Shift-F11) to open explorer and view the target location. You will find a LINUX32 or LINUX64 folder - this can be copied to a new "target" Linux system.

Step 2: Install on Linux system

The pre-built binary files will extract into a my-t-soft sub-directory. Be sure to copy the tar.gz file into your home or Desktop folder (as appropriate). Copy the my-t-soft_????.tar.gz file onto the Linux system, and extract (at command line) with **tar zxf**

my-t-soft_????.tar.gz[Enter] - this will create a my-t-soft folder with the program, data, and support files. There will be run-time information in a README.txt file. If you run My-T-Soft (e.g. ./my-t-soft/mytsoft[Enter]), you will see the demo layouts.

Step 3: Drop in your custom KBF

From the Copied files, you will see a KEYBOARD.KBF and a [Your Project Name].KBF - for testing, if only 1 layout, just copy the KEYBOARD.KBF to the ./my-t-soft folder (make sure My-T-Soft is closed). Then run My-T-Soft, and you will see your custom layout.

Tips:

1. Get My-T-Soft on the target first - you can download the demo from the website, or copy the .tar.gz file from the Targets

location, and install.

2. To run My-T-Soft, just use the command line, or a file manager - open the My-T-Soft folder, then double-click on My-T-Soft - depending on the Window Manager, you may need to create an Application link or Desktop shortcut - also refer to the README.txt installed with the run-time binary software.

3. My-T-Soft is the program, the KBF file is the data. Currently, for simplicity, all data must be in the ./my-y-soft location - just copy KEYBOARD.KBF in (overwrite existing) and run My-T-Soft.

4. As you make updates, just copy in KEYBOARD.KBF from each new build. Once My-T-Soft is on the target, you just need to copy in your "new" data file, i.e. updated .kbf file. Think Word Processer (Program) and a Document File (Data) -

the .kbf is the .doc, and each revision (build) creates an updated .kbf data file.

5. The Target folder (View Run-Time Targets in Explorer) for your project will always have the most recently build KEYBOARD.KBF.

UNIX

Build-A-Board Run-Time Target: UNIX

UNIX - 32 bit (UNICODE 2.20)

As of the 2.20 release, there is not a pre-built, ready-to-run UNIX deployable target. The reason is there has been no customer demand for any particular UNIX version. However, due to the cross-platform design and support for the low-level XLib interface, and file & memory similarities between Linux and Mac OS X and

UNIX, and older UNIX development, this deployable target awaits an actual customer need. If you are reading this with this requirement, please contact IMG directly.

Mac OS X

Build-A-Board Run-Time Target: Mac OS X

The Mac OS X target is built under XCode for 10.4 Intel and will work with 10.5/10.6. If older support is required, please contact IMG Technical Support.

Mac OS X (10.4 and higher) - 32 bit (UNICODE 2.20)

Target Files

My-T-Soft.app

This is the My-T-Soft executable that reads the

KEYBOARD.KBF layout (or multiple file layouts), and displays the window to the user.

Macrobat.app

This is the Macro Batch Processor. It is the support process for managing inter-process communication and handling low-level interface issues.

SeeThru.app

This is the slider display that enables/disables transparency, and sets transparency level. My-T-Soft must be running to access this tool.

KEYBOARD.KBF

WELCOME.KBF

The KBF file is the default display file for the My-T-Soft Board (from Build-A-Board). Any other KBF files are projects that have been built - With each successful build, a KEYBOARD.KBF and a ProjectName.KBF

file are created in the Target folder.

Deploying My-T-Soft Build-A-Board custom layouts onto Mac OS X

There are 2 aspects - the My-T-Soft software (program), and then the custom KBF file (data).

Step 1: Get built KBF (& install files)

From Build-A-Board, after you have designed, then Built the layout, you can go to the Run-time menu | View Project Run-Time Targets folder (or use Shift-F11) to open explorer and view the target location. You will find a MACOSX folder - this can be copied to a new "target" Mac system.

Step 2: Install on Mac

You can open the DMG, and mount the disk, then drag the My-T-Soft folder to the Applications folder. This will put the run-time

software onto the target system, and create a /Applications/My-T-Soft folder - if you use Finder, and run My-T-Soft, you will see the demo layouts

Step 3: Drop in your custom KBF

From the Copied files, you will see a KEYBOARD.KBF and a [Your Project Name].KBF - for testing, if only 1 layout, just copy the KEYBOARD.KBF to the /Applications/My-T-Soft folder (make sure My-T-Soft is closed). Then run My-T-Soft, and you will see your custom layout.

Tips:

1. Get My-T-Soft on the target first - you can download the demo from the website, or copy the DMG file from the Targets location, and install.
2. The DMG is bare-bones, but it works -

when opened, just click on the My-T-Soft folder, and drag it into the shortcut for the Applications folder - it will do the copy for you.

3. To run My-T-Soft, just use Finder, open Applications, open the My-T-Soft folder, then double-click on My-T-Soft

4. My-T-Soft is the program, the KBF file is the data. Currently, for simplicity, all data must be in the /Applications/My-T-Soft location - just copy KEYBOARD.KBF in (overwrite existing) and run My-T-Soft.

5. As you make updates, just copy in KEYBOARD.KBF from each new build. Once My-T-Soft is on the target, you just need to copy in your "new" data file, i.e. updated .kbf file. Think Word Processer (Program) and a Document File (Data) - the .kbf is the .doc, and each revision (build) creates an updated .kbf data file.

6. The Target folder (View Run-Time Targets in Explorer) for your project will always have the most recently build KEYBOARD.KBF.

Chapter 6. Build-A-Board Run-Time Targets Notes

Build-A-Board Run-Time Targets Notes

Build-A-Board Run-Time Targets Notes

Each Run-Time Target is built with target specific code, shared core application code, and abstracted API interface code. Because of compile, system, and code differences it is important to understand that each Run-Time Target should be considered its own Program, and as such, could be different than other

Run-Time targets (due to completely different code/design/implementation). The following chapter covers areas that affect the Run-Time targets - sometimes on a particular platform, sometimes across a feature and its implementation, and sometimes on an aspect affecting one particular option or feature.

While at first glance, an on-screen keyboard seems like a relatively easy program to develop, the nature of low-level development, platform differences, and the various areas that require in-depth programming practices to create a seamless user-interface requires a great deal of effort. Most programs can be modular, and focus on low-level specific functions, or high-level, and focus on user interface aspects. Due to the nature of the on-screen keyboard, not only is a high-level user interface required, but so is a low-level interface into the system. For speed and responsiveness, access to video

calls at a system level is required, along with typically system or low-level calls to operate and work with interprocess communication and virtual keyboard input. As such, code must be written at a system level, and the effort to create an environment that compiles at a native level across multiple platforms is not trivial or easy. So while an on-screen keyboard is something you might be able to do in a few hours with a high-level tool such as Visual Basic, you will find that animal won't even get in the door to run in Linux or Mac OS X (unless you have the means to recreate that whole high-level environment). Alternatives such as Java or Qt provide cross-platform environments for certain aspects, but can't provide the power and flexibility of operating without this abstraction layer between the code and the system, and also add a requirement that may be a show-stopper in embedded, secure, or other less-than-full-featured environments.

The various constraints and goals of creating a native, system level tool that can not only provide keyboard and system actions, along with a flexible user-interface that is responsive and an effective tool has resulted in each Run-Time target being its own Program, while operating on a shared Data format (KBF File). Because it is not always the same code operating on the data, and the scope and complexity of entire Build-A-Board concept, there may be issues or unintended consequences that may result when certain combinations of features, program/data interaction, and user actions occur.

One concept in programming is "Design for Test" so that every possible input and output can be tested & verified. While that sounds good on paper, it is a virtual impossibility when applied to the scope of Build-A-Board (unless access to virtually unlimited resources

is made available). Due to the facts that the user can create an unlimited combination of layouts, labels, and actions; for each platform, a wide range of processes and services can interact with the system; each user can perform different actions and sequences of interaction to generate results; therefore, it is probable that some user, on some platform, with some feature may experience a result that isn't expected, or isn't correct.

In conclusion, one of the engineering constraints was not having unlimited resources to verify every combination of features for every application for every platform for every device in every configuration for every possible situation. Therefore, we strongly recommend customers continue with on-going support, for updates, fixes, and access to technical support staff.

Images

Images

The PNG format is used for carrying key and panel images within the KBF file. This format was chosen because it is lossless and compressed, and supported on all current target platforms. However, the support on each platform has various limitations, and the information below covers some of these aspects. For best results, use 24-bit color images (16 million colors) vs. 8-bit (256 colors).

If a key or panel contains a PNG image to display, and there is some problem on the run-time system to display the PNG image, a black background will be all that is shown on the key/panel. So an expected PNG image showing a black background indicates a PNG display issue, and you should reference the

information below for more details on a particular system.

PNG on Windows

To display PNG files, the system GDIPlus.dll must be available. This was added during the life of XP, so it may not be available on Windows 2000, or older, not up-to-date XP systems. There is a redistributable available from Microsoft (see WindowsXP-KB975337-x86-ENU.exe, knowledgebase KB975337) to add this to older systems.

MYTSOFT.EXE requires the GDIKIT.DLL file to be in the same folder as MYTSOFT.EXE - the dependency link is GDIKIT.DLL which needs MSVC80??.DLL which relies on GDIPLUS.DLL for PNG display. The most effective way to resolve a black background / no PNG display issue on Windows is as follows:

- Verify GDIKIT.DLL is in the same folder as MYTSOFT.EXE
- For 2000/XP, Verify GDIPLUS.DLL is on the system (\WINDOWS\SYSTEM32)
- For 2000/XP, GDIKIT.DLL may need C run-time (CRT) libraries - copy these files from \Program Files\Build-A-Board\BIN: Microsoft.VC80.CRT.manifest, msvcm80.dll, msvcp80.dll, msvcr80.dll into the same folder as GDIKIT.DLL

PNG on Linux

Linux requires the libpng to be on the system - when installed (e.g. Debian "apt-get install libpng") a libpng.so file (.so = shared object) is typically a symbolic link in the /usr/lib folder. The run-time mytsoft on Linux uses a library call (dlopen/dlsym/etc.) to access the library without requiring it (for cases where the system does not have libpng available). So the

following may be helpful in getting images displayed on a Linux system (if a KBF with images shows a black background where you would expect the image).

- Important requirement 1: libpng support must be available
- Important requirement 2: libpng.so is only file accessed, so symbolic link may be required.
- Verify the libpng is available - apt-get install, Synaptic, rpm, yast, yum, etc. dependent on the distribution
- The run-time mytsoft ONLY looks for libpng.so - so if this is not resolving, check for libpng.so in /usr/lib (or /usr/lib64, etc.) a symbolic link may be required if no actual libpng.so is there (for example, use "ls -l /usr/lib/libpng*[Enter]" for details on links -

to create libpng.so, use "ln -s /usr/lib/libpng.so.3.1.2.8 /usr/lib/libpng.so[Enter]" (root/superuser required))

- Actual lib location (lib vs. lib64, or /lib vs. /usr/lib vs. /usr/png/lib) - use ldd command, e.g. "ldd mytsoft[Enter]" to list dependencies of mytsoft (and where library locations are on the system) or use "man hier[Enter]" to see the system's manual page on the file hierarchy
- Run from command prompt for error details reported from libpng code - it may help identify a dependency, or specific issue that can be resolved.

PNG on Android

The PNG display code is available in the native system API, so there are no known issues with PNG display on this platform.

PNG on Mac OS X

The PNG display code is available in the native system API, so there are no known issues with PNG display on this platform.

Fonts

Fonts

Font support on the various platform is varied. One issue is the the development system for the Builder may have Fonts that are not on the target run-time system, and the matching algorithm (dependent on the run-time program) may not select an acceptable substitute. It is best to limit fonts to family (e.g. serif vs. san-serif), or work with known target available fonts. Fonts are also not necessarily system transferrable, and there may be licensing issues

with particular fonts. Also, with UNICODE support, some fonts may not be fully populated for all glyphs on the target system, resulting in inconsistent results.

The following outlines some known issues and implementation notes for Fonts on the target systems. Because there is no consistency or world standard on fonts, names, and support on a particular system, fonts can represent a challenge for implementers of cross-platform run-time targets.

Fonts on Windows

Because Windows fonts can typically be moved (or are available) for multiple Windows systems, the only real concern is matching the fonts on the development system (where the Builder is creating the board), and the run-time target system(s).

Fonts on Linux

Linux has a well designed font matching and labeling system, but due to licensing and varied support, not all fonts available on Windows may be available on a particular run-time Linux system. Also, full Unicode support for various glyphs can be limited, especially when mixing font styles and sizes. The default approach is to perform a match based on Linux font naming and try different sizes within that family. Trying different fonts, or working with more common fonts may yield thest best results when working on Linux run-times.

> **Note:** Additional support for overrides and manual matching will be added to Linux run-times, but is currently not available.

Fonts on Android

As of the 2.20.11 release, only the default

system font is used, and no KBF font support is available.

Fonts on Mac OS X

The font calling mechanism on Mac OS X is also different than Windows, although there is more font availability on Mac OS X. Trying different fonts, or working with more common fonts may yield thest best results when working on Mac OS X run-times.

> **Note:** Additional support for overrides and manual matching will be added to Mac OS X run-times, but is currently not available.

Caps Lock

Caps Lock

Caps Lock and Shift operation is different on different platforms, and also for different languages (e.g. French). Over time, the generally preferred operation has been to change the on-screen keyboard label to match the resultant key press for alphabetic & shifted keys. This is accomplished through Shifted Key Labels and run-time operation settings.

Caps Lock on Windows

The default operation is to have the shift toggle the alphabetic keys out of their capital (shifted) modes.

Caps Lock on Linux

The default operation is to have the shift toggle the alphabetic keys out of their capital (shifted) modes.

Caps Lock on Android

The default operation is to have the shift toggle

the alphabetic keys out of their capital (shifted) modes. This platform also has an optional shift-lock feature.

Caps Lock on Mac OS X

The default operation is to ignore the shift state when the Caps Lock state is set. This is implemented as a special case in the shared run-time code.

> **Note:** The Caps Aware key type available in the Builder is not implemented with this release. The approach will be to move the Caps Lock operation and state management as a layout option and user-settable. Because this aspect of an on-screen keyboard can be manipulated based on the layout, system, or user, moving towards the most flexible implementation by making this configurable will resolve various issues.

Part IV. Build-A-Board Technical Notes

Advanced, Technical, and additional Operation notes.

Details and information on the technical architecture, design, and use of Build-A-Board.

Chapter 7 - Build-A-Board Technical Notes

contains general technical information, and release notes.

Chapter 7. Advanced User Notes

Advanced User Notes & Information

The Advanced User Notes and Technical Documentation covers a wide-range of topics & information. Detailing all of the features, options, settings mixed in with the standard product help would overwhelm the majority of users, so the more technical and advanced portions are included here.

Note: For up-to-date information, and other specific technical issues, it will always be important to refer to the on-line

support database at
http://www.imgpresents.com/imgfaq.htm

Note: For developers, integrators, or technicians, the following information should be read in its entirety - also refer to the IMG Developer's Kit for even more options & other advanced information.

If you have been referred to this section, or have been unable to resolve your question(s) or problems within this manual, on-line help, and tutorial, or are an advanced user and wish to learn additional information about Build-A-Board not required for typical end-users, please read the following chapter. For developers, integrators, or technicians, the following information may be useful and

should be read in its entirety.

Build-A-Board Files & File Notes & Installation Information

Build-A-Board Files & File Notes

Files located in the \Program Files\Build-A-Board\BIN Installation Directory:

Note: Previous versions of Build-A-Board had SOURCE and TARGET folders in the \Program Files\Build-A-Board folder. After Microsoft released Vista, standard users no longer had permission to write files in the \Program Files area, so these folders are installed in the shared/public

Documents area. But for consistency, and future use, the single folder BIN remains as the only sub-folder.

The following files are REQUIRED for proper operation of

- BUILDERU.EXE - Build-A-Board executable - the Builder / Layout editor

- BABTCU.EXE - Build-A-Board Text Compiler executable - used by the Builder

- BABDLLU.DLL - Build-A-Board support library - used by the Builder

- IMGUTIL.EXE - used for installation and uninstallation of software

- IMGVERS.DLL - IMG Dynamic Link Library

- GDICONVERT.EXE - Image conversion program (GDIPlus required)

(BMP/JPG/GIF/PNG/TIF)

- GDIKIT.DLL - Interface DLL for GDIPlus / GDIConvert

- HELP*.html, HELP\IMAGES*.png - Build-A-Board Help

- KEYS.INI - Reference file for User selectable Keys window (Builder)

- LICENSE.EXE - IMG License Manager

- LICENSE2.EXE - IMG License verification only (read-only access)

- LICENSE.LIC - IMG License File (with current license)

- LICENSE.ORG - Original IMG License File (as released)

- README.TXT - Product Installation text file

- MANIFEST.TXT - Reference File from Build

Microsoft redistributable files (GDI support)

- Microsoft.VC80.CRT.manifest - manifest for MS VC run-time DLL (GDI support)

- MSVCM80.DLL - MS VC run-time DLL (GDI support)

- MSVCP80.DLL - MS VC run-time DLL (GDI support)

- MSVCR80.DLL - MS VC run-time DLL (GDI support)

- SEETHRU.EXE - Separate EXE for controlling transparency (Windows)

- STOCK.DLL - 2.10 support, HiRes graphics resource library

- UNZIP32S.DLL - Dynamic Link Library for installation and updates

- ZIP32.DLL - Dynamic Link Library for ZIP compression

- ZIPDLL.DLL - IMG interface into Zip/Unzip DLLs

- uninstall.exe - Used for un-installation link (MSI Installs)

KMF Files (Keyboard Macro Files) (default keyboard mapping tables):

- KEYBRD01.KMF - Windows default keyboard map

- MAC00000.KMF - Basic macro file

- MACOSX.KMF - Mac OS X default keyboard map

- XLIB.KMF - X Windows default keyboard map

Target Folders:

- ANDROID29 - Android - API 29 (2.20 UNICODE) (as .apk)

- LINUX32 - Linux 32-bit Intel binary (2.20 UNICODE) (as .tar.gz)

- LINUX64 - Linux 64-bit Intel binary (2.20 UNICODE)(as .tar.gz)

- MACOSX32 - Mac OS X 32-bit Intel binary (2.20 UNICODE) (as .DMG)

- MSWIN - Microsoft Windows (2.10 ANSI) 32-bit Intel binary

- MSWIN32 - Microsoft Windows (2.20 UNICODE) 32-bit Intel binary

- MSWIN64 - Microsoft Windows (2.20 UNICODE) 64-bit Intel binary

- TEST - Builder system run-time binary (Testing layouts from within Builder)

- WCE_ARMV4 - Microsoft Windows CE (2.10 ANSI) 32-bit ARMV4 binary

- WCE_ARMV4I - Microsoft Windows CE (2.10 ANSI) 32-bit ARMV4I binary

- WCE_MIPSII - Microsoft Windows CE (2.10 ANSI) 32-bit MIPSII binary

- WCE_MIPSIV - Microsoft Windows CE (2.10 ANSI) 32-bit MIPSIV binary

- WCE_SH3 - Microsoft Windows CE (2.10 ANSI) 32-bit SH3 binary

- WCE_SH4 - Microsoft Windows CE (2.10 ANSI) 32-bit SH4 binary

- WCE_X86 - Microsoft Windows CE (2.10 ANSI) 32-bit X86 binary

- WCESIP_ARM - Microsoft Windows CE SIP / DLL CAB (2.10 ANSI) 32-bit ARM binary

- WCESIP_MIPS - Microsoft Windows CE SIP / DLL CAB (2.10 ANSI) 32-bit MIPS binary

- WCESIP_X86 - Microsoft Windows CE SIP / DLL CAB (2.10 ANSI) 32-bit X86 binary

Other Folders:

- MANAGER\ESTABLISH.EXE - this is a post installation utility that establishes or

updates the IMG Download Manager and IMG License Manager files as outlined below.

Located in \Program Files\Common Files\Innovation Management Group\Download Manager directory:

- IMGCLEAN.EXE - used to complete uninstall of software, required for Control Panel Add/Remove Programs

- IMGDLM.EXE - The IMG Download Manager - Installed by MANAGER\ESTABLISH.EXE

- IMGNET.DLL - The IMG Download Manager Library - Installed by MANAGER\ESTABLISH.EXE

Located in \Program Files\Common Files\Innovation Management Group\License Manager directory:

- LICENSE.DLL - library used by the IMG

MYTSOFT.INI Settings and Details

The listing below is the Version 2.20 Release 7 default listing/settings for MYTSOFT.INI (Builder file MYTSOFT.TXT). This text file is embedded in the built .KBF file and settings are platform dependent (as documented for the platform or in the file below).

[Configuration]

Info=This is MYTSOFT.TXT from Builder as Unicode.

;if PanelImage references a valid image (of form IMGnnnnn.PNG, e.g. IMG00002.PNG)

;then this image will be used at run-time as the panel background

PanelImage=

;The following 2 entries are created at build time, and are normally seen

;in the built KBF via KBFEdit (but not from the Builder)

;These entries should not be modified or removed

;ProjectName=

;GUID=

[Settings]

;Various global or user optional settings can go here.

;This file is embedded in the KBF, and is accessed at run-time.

;This provides another way for settings (especially user options) to travel with a KBF file.

XPosition=40%%

YPosition=38%%

;The XPosition / YPosition settings can be any supported format. If set, these automatically

;override the embedded pixel position as part of the KBF Window data structure

;Additionally, for these to be used as overrides, the Position type embedded in the

;KBF must be set to 2 - if default 1, these overrides will not be used. Internal Save Position

;and KBFEdit utility can save pixel position and clear the override.

;nn% (where nn can be 0-100) defines percentage of main display for Top/Left X/Y

;<left> <right> for X left side, middle of screen, or X right side of display

;<top><middle><bottom> for Y top, middle of screen, or Y bottom of display

;Example: <bottom><right> or <middle>>.

XParallax=0

YParallax=0

;The XParallax / Y Parallax settings can be a positive or negative integer value that

;directly modify the X/Y left click/mouse button position to adjust for an external

;or system offset induced by glass thickness / positioning / etc.

;The implementation is to adjust the actual coordinate by adding the integer indicated

;here - use XParallax=0 and yParallax=0 to bypass these settings

KeyBlockTime=0

;The KeyBlockTime is a per key blockout time in milliseconds where any additional

;key press via mouse click event is ignored (blocked) The setting is per key

;but is global in nature - there is one setting for any particular key, and that key

;cannot be "clicked" during the blockout time. However, another new key press will clear

;the current blockout and start a new blockout time for the new key.

;Use KeyBlockTime=0 to bypass this setting

;KeyBlockTime=100 means 100ms key block time out

;KeyBlockTime=500 means 500ms (1/2 second) key block time out

CursorOverride=Hand

;The CursorOverride allows selection of something other than the default hand type cursor

;None = blank, no cursor

;Blank = blank, no cursor

;Dot = small dot cursor

;SmallHand = small hand cursor

;Hand = regular hand cursor

;no entry, anything else - regular hand cursor

MenuOverride=

;The MenuOverride setting will bypass the built in "Access Popup Menu" setting in the KBF

;If MenuOverride=No or just blank, MenuOverride= then the KBF setting will use

;If MenuOverride=Menu the menu will be available

;If MenuOverride=NoMenu, then the menu will not be available

MenuOffScreen=

;The MenuOffScreen setting will enable the popup menu (if available) when a portion of

;the window is moved off-screen. This allows a limited pointing device to access

;the menu

;If MenuOffScreen=No or just blank, MenuOffScreen= then no menu will be shown

;If MenuOffScreen=Yes then the menu will be shown if a portion of the keyboard is off-screen

MenuLeftHold=

;The MenuLeftHold setting will enable the popup menu (if available) when the main

;button is held for several seconds. This allows a limited pointing device to access

;the menu

;If MenuLeftHold=No or just blank, MenuLeftHold= then no menu will be shown

;If MenuLeftHold=Yes then the menu will be shown if main button is held down

;The menu will be shown when the mouse button is released. The menu will not be

;shown if the release position is more than 5 pixels away from the pressed position,

;so any movement will typically prevent the menu from being shown.

MoveOverride=

;The MoveOverride setting will bypass the

built in "Allow User to Move My-T-Soft (click & drag)"

;setting in the KBF

;If MoveOverride=No or just blank, MenuOverride= then the KBF setting will use

;If MoveOverride=Move the user will be able to move the layout (click & drag to move)

;If MoveOverride=NoMove, then the layout will not be moveable

HoverEnable=

;The HoverEnable setting will enable the use of the key hover highlight option

;If HoverEnable=No or just blank, the key hover option will be disabled

;If HoverEnable=Yes or HoverEnable=Hover the key hover will be enabled

;If HoverEnable=NoHover, then the key hover option will be disabled

KeyPressEnable=

;The KeyPressEnable setting will enable the use of the key press highlight option

;If KeyPressEnable=No or just blank, the key press option will be disabled

;If KeyPressEnable=Yes or KeyPressEnable=KeyPress the key press option will be enabled

;If KeyPressEnable=NoKeyPress, then the key press option will be disabled

StretchToFitWidth=

;The StretchToFitWidth setting will enable the use of the automatic resizing to fit width of screen

;If StretchToFitWidth=No or just blank, the stretch to fit option will be disabled

;If StretchToFitWidth=Yes or the stretch to fit option will be enabled

;If StretchToFitWidth=WinCE, then the stretch to fit option will be only enabled or Windows CE platforms

KeyZoom=No

KeyZoomOffset=15

KeyZoomRegionAdjust=8

KeyZoomHideZoomTimeMS=250

KeyZoomMagFactor=2

KeyZoomLocation=1

KeyZoomImage=Down

KeyZoomWideKeyRatioNoZoom=1.5

;The KeyZoom setting will use a secondary window to display a magnified (zoomed) image of

;the key being pressed. If KeyZoom=Yes, the window will be used, and the other settings

;will be referenced. If KeyZoom=No (or blank/anything but yes), the window will not be

;used, and the other settings will not be referenced.

;KeyZoomOffset=15 - this is a pixel offset used for handling the display of the window

;off the cursor - larger values means further away from the current cursor position.

;KeyZoomRegionAdjust=8 - this is a value used for the rounded rectangle region used for

;the key zoom window. If the value is negative, no region is used, and a rectangle will

;be shown. Larger values mean less area for the key, and more rounded corners.

;KeyZoomHideZoomTimeMS=450 is the setting for the timer used to automatically hide

the

;zoomed image window - reasonable values range from 500-1500. Note that if the mouse cursor

;clicks or moves over the zoom window, it will instantly hide.

;KeyZoomMagFactor=2 - this is how much of a magnification factor will be used. It is an integer.

;Reasonable values are 2 or 3.

;KeyZoomLocation=1 - this setting determines which location the zoom window will be shown

;in reference to the mouse cursor / click point. Use 1 for above, 2, for below, 3 for left, 4 for right

;KeyZoomImage=Default - this setting determines which image(s) will be used for the

zoom/magnified key image

;If empty, or KeyZoomImage=Active, then the current/active painted image will be used. If

;KeyZoomImage=Default, then only the base/original/Up image will be used for the zoom image

;If KeyZoomImage=Down, then the down key image will be used for the zoom image

;KeyZoomWideKeyRatioNoZoom=0 - If KeyZoomWideKeyRatioNoZoom is 0, then ALL keys will be

;treated as equal, and the zoom window will be displayed for all keys.

;If KeyZoomWideKeyRatioNoZoom is non zero, then the ratio will determine which wide keys will NOT be

;displayed. For example, if KeyZoomWideKeyRatioNoZoom=2, then only

keys twice as wide as they are high

;will NOT be zoomed.
KeyZoomWideKeyRationNoZoom=1.5 is the lowest value you should use, as lower

;values could result in no keys being displayed, even though KeyZoom=Yes

[Region]

;This is 1280 settings

;Area1=Ellipse,0,2,70,70

;Area2=Rect,0,38,1280,229

;Area3=Ellipse,1210,2,1280,70

;Area4=Rect,38,2,1242,264

;Area5=Ellipse,0,194,70,264

;Area6=Ellipse,1210,194,1280,264

;The section [Region] can have Area1 and /or Area2 entries up to Area8

;The rectangle is 0 based, and has form "Area1=Rect,[Left],[Top],[Right],[Bottom]"

;e.g. Area1=Rect,10,12,100,200

;

;The Ellipse shape is the ellipse that fits in the rectangle defined by

;a specified "Ellipse" and the rectange coordinates

;0 based, and has form "Area1=Ellipse,[Left],[Top],[Right],[Bottom]"

;e.g. Area1=Ellipse,10,12,100,200

;

;If only Area1 exists, it will be the sole region

;If the coordinates or the region fails for any reason, no region action will occur

;

;If Area1 and Area2 exists, they will be combined with an OR logic function,

;resulting in a "Union of the 2 combined regions" This continues up to Area8

;

;If the regions are not connected, there will be multiple visible regions

;

;If any area is invalid, no region action will occur

;

;General approach / suggestions

;- Establish board size in Build-A-Board

;- Decide on final shape to be shown on target system

;- Use graph paper, or visualize within Build-A-Board the coordinates for

;the final shape

;- Enter Area1 and/or Area2-Area8 as required

;- Save [Region] section in Keyboard Run-Time Settings

;- Build board

;- Test

;- To disable, remove section, or comment out entries using semi-colon

;

;Example - rounded rect keyboard

;[Region]

;Area1=Ellipse,0,2,70,70

;Area2=Rect,0,38,1280,229

;Area3=Ellipse,1210,2,1280,70

;Area4=Rect,38,2,1242,264

;Area5=Ellipse,0,194,70,264

;Area6=Ellipse,1210,194,1280,264

;

;

;Examples - Board is 200 x 300

;Circle

;[Build-A-Board]

;Area1=Ellipse,0,0,200,200

;T shape

;[Build-A-Board]

;Area1=Rect,0,0,200,100

;Area2=Rect,75,100,125,300

;Reversed L shape

;[Build-A-Board]

;Area1=Rect,150,0,200,300

;Area2=Rect,0,200,200,300

[Macros]

Macro1=Example Macro[Enter]

Macro2=This is a sample sentence for Macro #2![Enter]

Macro3=[CMD:EXEC=Notepad.exe]

Build Process Notes

When opening or building board with many keys (est. 200+), it is possible that some systems will remain in loading or building state for a long period of time, or actually not complete the process. During the load and build process, the Build-A-Board Text Compiler gets initiated for each window, panel, key, etc., and when there are many keys, there can be hundreds of processes of the Text Compiler launched to compile each source file

- see Build-A-Board Text Compiler notes.

If you experience this on a system, be sure to check for proper system memory configuration, virtual memory configuration, and free disk space. You can refer to the Task Manager to see if there are any BABTCU.EXE processes running (and you can end these processes within the Task Manager to cancel the load or build process - note, however, that results will be incomplete, and Build-A-Board should be closed without saving any projects). You may wish to reset the system, and check system logs for any system issues that may affect memory or the ability of the system to launch and run processes.

Run-Time Log Files

In the Windows, Linux, and Mac OS X

run-time platforms, there is support for a run-time log file to identify KBF, License, and other run-time aspects during operation. The file itself is the flag to indicate logging should occur. The file must be named **mytsoft.log**, and it must reside in the same folder as the MYTSOFT.EXE/mytsoft/My-T-Soft.app (i.e. the location of the executable for My-T-Soft).

In the Android platform, there is an optional Logging setting in the Operation Settings.

You can create an empty file with a text editor, or at the command prompt, use the following approach: On Linux and Mac OS X, , or use the touch command, e.g. touch mytsoft.log[Enter]. On Windows, use copy con mytsoft.log[Enter]Ctrl-Z (or F6 key)[Enter].

> **Note:** It is recommended you delete this file when you are done working with the log file, as it will continue to grow if left in

existence.

Note: Macrobat also will look for the mytsoft.log file, and will log information in the same file (indicated as Macrobat).

Note: The logged information is currently fairly limited, and is primarily meant for debugging purposes to indicate how far into initialization sequence the program has gone, and for license info. If there are specific needs or requirements, please contact IMG technical support for further info. It is anticipated that more documentation and log options will be added in the future.

Build-A-Macro Notes

IMPORTANT NOTE: This section is from the 1.xx software, and is primarily included to show the KMF key numbers (below). When using {KMF:???} and [KMF:???] type overrides, the original keyboard location/key number may be needed - this section lists the key reference numbers from the 1.xx software. The macro specifics, in general, are not supported on platforms other than Windows, and even the Windows releases may not support all aspects outlined here. For Android platforms, only reference the construct of the %% type codes - only these are supported for Key Actions. If you are reading this section for something other than the key listings to reference KMF lookups, you may wish to contact IMG Technical Support to clarify your version of run-time software, and review what you are trying to accomplish.

The following double-characters are reserved in Build-A-Macro:

@@ - Signifies an Alt keystroke to follow

e.g. @@f = [Alt-Down]f[Alt-Up]

~~ - Signifies a Ctrl keystroke to follow

^^ - Signifies a Shift keystroke to follow

$$ - used internally for internal macro uses

%% - used to specify a virtual key, internal keystroke, or a keyboard scan code

You may not use these character combinations in your macros, unless you use them as outlined. For example, you may quickly create a macro for File, New ([Alt]-F, N) by entering "@@fn" and clicking OK. However, using the Reserved words in brackets is the preferred method.

The %% sequence has 4 options, and must be formed properly to be interpreted as an special

entry. 4 characters must follow the 2 percent signs, spaces are not allowed. When this is used, it generates both the Down and Up keyboard messages, (press & release), similar to the entry of a specific character.

The following general form is:

%%cnnn

where c is a character signifying Keystroke or Scancode or Virtual Key Code or ProcessKey keystroke - The only valid characters are k or or e or s or v or p (case does not matter)

The nnn must be 3 decimal digits (values between 000 and 999 are valid). Note that there are only 200+ keys on the panels and 255 Virtual keys - higher numbers may cause memory access errors!

The keystroke (k) numbers refer to the internal numbering of the keyboard - see below for a table of the standard 101 keyboard.

For example, to generate a F10 keystroke, you may use "%%k010" in the Build-A-Macro portion. In most cases, using the actual character will generate the appropriate scan codes internally, and this is only provided as an enhancement to provide more functionality in certain environments. Documented deficiencies include lack of support for the keys on the Numeric keypad panel. Additionally, by using the scancodes, macros can generate "keystrokes" for keys that are not present on the physical keyboard. Because of the hardware oriented nature of the actual scancode generated by the physical keyboard, and its interpretation by the system hardware and Windows, the scope of these issues requires that they be referenced in appropriate system manuals, and/or documentation on industry standards.

The Extended Key (e) is similar to k, but sends

key event as an extended key code.

The Virtual Key (v) passes the Windows based Virtual key code directly through Build-A-Board. How an application interprets the virtual key code is dependent on regional, language, keyboard layout, and possibly other issues.

The ProcessKey (p) approach uses the internal ProcessKey function which handles more internal processing beyond the AcceptCode function handled by the k, s, or v options. This was added to allow macros to interact with the WordComplete panel on the Assistive Technology version.

> **Technical Information:** The internal interface uses the AcceptCode approach which adds the keystroke to the buffer to be typed. This is managed in different ways as specified based on the k, v, or s

options. The p option calls the internal ProcessKey that does various tasks prior to calling the AcceptCode function to insert the keystroke into the output buffer. Because of the tightly interwoven aspects of ProcessKey, the keyboard panel, and special options such as WordComplete, there may be side-effects if the %%pnnn is used extensively. There may also be other interactions with various other features (Show & Hide Keys, LetterAssist, etc.) that the direct AcceptCode approach will completely bypass. The intended use is to allow a single macro button to type a letter, and interact with the WordComplete panel in the Assistive Technology version.

It is probably worthwhile to state that each of these approaches is important dependent on the context that you want a particular keystroke. Is the keystroke dependent on the physical layout or virtual layout selected? Is the keystroke a character typed into the application with

keyboard focus? Is the keystroke acting like it was typed on a the physical keyboard? Is the keystroke macro acting like it was typed on the virtual keyboard? Each can be valid and required depending on many factors. Supporting each possibility may explain one reason why the software has been around for 30 years.

Quick Reference:

%%knnn, e.g. %%k010 - uses the currently selected KMF (Build-A-Board layout) and looks up the virtual key / scan code information for this key, sends down/up keystrokes

%%snnn, e.g. %%s112 - sends the scancode, but processed internally. In most cases this will be the same virtual key code as specified, but it may be modified.

%%vnnn, e.g. %%v112 - sends the Windows

Virtual Key (refer to Microsoft documentation) unmodified (note decimal notation, hex is not supported).

%%pnnn, e.g. %%p041 - uses the currently selected KMF (Build-A-Board layout) and looks up the virtual key / scan code information for this key, sends down/up keystrokes, and processes the key as if typed via the keyboard directly.

Key# Description

0 Escape Key

1 F1

2 F2

3 F3

4 F4

5 F5

6 F6

7 F7

8 F8

9 F9

10 F10

11 F11

12 F12

13 Select key

14 1/!

15 2/@

16 3/#

17 4/$

18 5/%

19 6/^

20 7/&

21 8/*

22 9/(

23 0/)

24 -/_

25 =/+

26 Back space

27 Tab

28 Q

29 W

30 E

31 R

32 T

33 Y

34 U

35 I

36 O

37 P

38 [/{

39]/}

40 Caps Lock

41 A

42 S

43 D

44 F

45 G

46 H

47 J

48 K

49 L

50 ;/:

51 ’/"

52 Enter Key

53 Enter Key

54 Shift / Left Shift

55 Z

56 X

57 C

58 V

59 B

60 N

61 M

62 ,/<

63 ./>

64 //?

65 Shift / Right Shift

66 \/|

67 Control / Left Control

68 Alt / Left Alt

69 Space Bar

70 Alt / Right Alt / Alt-Gr

71 Control / Left Control

These are on the edit panel

72 Print Screen

73 Scroll Lock

74 Pause

75 Insert

76 Home

77 Page Up

78 Delete

79 End

80 Page Down

81 Up Arrow

82 Left Arrow

83 Down Arrow

84 Right Arrow

These are on the Numeric keypad panel

85 CL indicator / not processed as key

86 NL indicator / not processed as key

87 SL indicator / not processed as key

88 Num Lock

89 /

90 *

91 -

92 7

93 8

94 9

95 +

96 4

97 5

98 6

99 1

100 2

101 3

102 Enter / =

103 0

104 decimal point

Developer's Kit

All relevant & up-to-date Help information is part of the IMG Developer's Kit - See the DEVKTDOC folder. Also reference on-line

information at IMG's Developer's Corner (http://www.imgpresents.com/imgdev.htm).

2.20 Version

In the 2.20 Version, all Developer's Kit tools and information is in the IMG Developer's Kit.

Final Release Notes

Version 2.20 Release 7 (June 22, 2022)

License updates, documentation updates, Windows/Linux/Android platform updates, maintenance updates.

Version 2.20 Release 6 (September 14, 2018)

Ongoing updates, user interface updates to the Builder, documentation updates, Android platform update, maintenance updates.

Version 2.20 Release 5 (January 5, 2018)

Ongoing updates, features, and more capabilities to address customer needs and requirements. Includes Android platform, Key Images, embedded KMFs, new IMG License Manager, ties to Build-A-Board.com online database of layouts and Build-A-Board.com Accounts, maintenance updates.

Version 2.20 Release 4 (August 4, 2016)

Private release to select customers - includes Windows 10/8.1/8 support, maintenance updates.

Version 2.20 Release 3 (August 9, 2010)

Version 2.20 Release 3 is the first public, licensable version of Build-A-Board 2.20 (previous releases were Evaluation only)

Support for Linux/Unix/Mac OS X, updates for Windows Vista/7, Windows CE, and 32-bit and 64-bit support. The 2.20 KBF includes support for Unicode, modifiable key-mappings,

user options (MYTSOFT.INI), image support, and flexible license options.

Version 2.10 (March 21, 2002)

Many of the missing user-interface pieces in the 2.00 release have been addressed. Support for all major Windows CE releases has been included. Projects in 2.00 will be automatically converted to the 2.10 formats. See the Release Notes (RELEASE.WRI) for final feature list & known limitations.

Version 2.00 (April 9, 2001)

As the first commercially available release, we acknowledge that this is the very beginning... In order to address all our customers needs, and in response to our customers who are able to build satisfactory solutions given the limitations within the Builder and the various support components, we have released the software as the 2.00 version. See the Release

Notes (RELEASE.WRI) for final feature list & known limitations.

Closed Project Storage as Zip

Build-A-Board Projects contain numerous files. Most file storage approaches use file allocation schemes that allocate a minimum storage unit. Since many small files can use a disproportional amount of storage space, using a file compression approach to store closed projects helps conserve the system's storage space. Since the Zip format is common and there are numerous utilities available to manipulate these types of files, the use of this format was chosen. The ZIP32.DLL and UNZIP32S.DLL file used for Zip files was not developed by Innovation Management Group,

Inc. The following copyright & license information is included here as required.

==========

This is version 2000-Apr-09 of the Info-ZIP copyright and license. The definitive version of this document should be available at ftp://ftp.info-zip.org/pub/infozip/license.html indefinitely.

Copyright (c) 1990-2000 Info-ZIP. All rights reserved.

For the purposes of this copyright and license, "Info-ZIP" is defined as the following set of individuals:

Mark Adler, John Bush, Karl Davis, Harald Denker, Jean-Michel Dubois, Jean-loup Gailly, Hunter Goatley, Ian Gorman, Chris Herborth, Dirk Haase, Greg Hartwig, Robert Heath, Jonathan Hudson, Paul Kienitz, David Kirschbaum, Johnny Lee, Onno van der

Linden, Igor Mandrichenko, Steve P. Miller, Sergio Monesi, Keith Owens, George Petrov, Greg Roelofs, Kai Uwe Rommel, Steve Salisbury, Dave Smith, Christian Spieler, Antoine Verheijen, Paul von Behren, Rich Wales, Mike White

This software is provided "as is," without warranty of any kind, express or implied. In no event shall Info-ZIP or its contributors be held liable for any direct, indirect, incidental, special or consequential damages arising out of the use of or inability to use this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. Redistributions of source code must retain the above copyright notice, definition, disclaimer, and this list of conditions.

2. Redistributions in binary form must reproduce the above copyright notice, definition, disclaimer, and this list of conditions in documentation and/or other materials provided with the distribution.

3. Altered versions--including, but not limited to, ports to new operating systems, existing ports with new graphical interfaces, and dynamic, shared, or static library versions--must be plainly marked as such and must not be misrepresented as being the original source. Such altered versions also must not be misrepresented as being Info-ZIP releases--including, but not limited to, labeling of the altered versions with the names "Info-ZIP" (or any variation thereof, including, but not limited to, different capitalizations), "Pocket UnZip," "WiZ" or "MacZip" without the explicit permission of Info-ZIP. Such altered versions are further prohibited from

misrepresentative use of the Zip-Bugs or Info-ZIP e-mail addresses or of the Info-ZIP URL(s).

4. Info-ZIP retains the right to use the names "Info-ZIP," "Zip," "UnZip," "WiZ," "Pocket UnZip," "Pocket Zip," and "MacZip" for its own source and binary releases.

Index

A

B

C

D

E

F

G

H

K

L

M

O

P

Q

R

S

T

U

V

W

Z