

WINBUILD 2000 PROGRAM DEVELOPMENT

- 1** → Read through this document first to understand how Winbuild 2000 projects are constructed, and how they run.
- 2** → Proceed through the Winbuild 2000 Introductory Labs to build a simple program, and learn the basic features in the software. Continue with more labs as needed.
- 3** → After going through the labs, complete the Project Worksheet to help plan your project.

WINBUILD 2000

- **What is Winbuild 2000?**
- **How does Winbuild 2000 run?**
- **What are the parts to a Winbuild 2000 project?**
- **How do I get started using it?**

TERMS:

- **Application** – The program you create that will be compiled and downloaded into the Eason unit.
- **BASIC** – Beginners All-Purpose Symbolic Instructional Code. A popular high-level programming language that is easy to use and learn. Eason's BASIC uses standard ANSI language, and adds some object-oriented capabilities.
- **Driver** - a piece of software that allows the Eason HMI unit to talk to another device. Eason supplies the drivers for just about any device to which you might connect.
- **HMI** – Human-Machine Interface or MMI (Man/Machine Interface) Essentially the screens and objects that the operator will interact with.
- **Object** - An item you might place on your HMI display screen. Typically these are buttons, indicator lights, data entry items, etc. A special feature of Winbuild is that BASIC programs can be written and included within your objects for additional functionality.
- **Pseudocodes** – a line of BASIC code that performs commonly needed task (such as sending information out a serial port or getting data from an external motion controller). These are already written and supplied by Eason. You merely have to select the pseudocode from the list in order to add it to your program.
- **Tag** - A data variable you might use in your application program. A typical variable would be "speed" which might contain the desired speed of a motor.

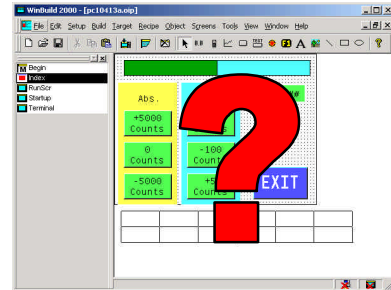
Tags can be set up to contain numbers or strings and you get to set up the tag names. Special tags called arrays are also available.

- **Task** – a program that you can write that runs “in the background”. A typical task might control I/O or command an external motor controller to start and stop. Your task(s) can run independently of the screen display program or can be linked in with it.

WHAT IS WINBUILD 2000?

Winbuild 2000 is a software package that gives you the ability to create an HMI program with motion and control capabilities for an Eason 2000 Family Intelligent Interface.

- Windows Based Graphical Programming Tool
- Object Property Based Programming
 - Place Objects on the Screen and Assign Properties
 - Create Variables and Communications Links Graphically
- BASIC and Pseudocodes Can be Attached to Any Object
- Multi-Tasking BASIC for Control
- Specifically Designed for the 2000 Family



Winbuild 2000 has integral support for:

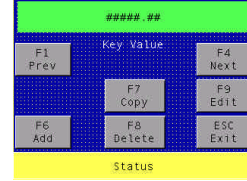
- Multi-Tasking
- Recipes
- PLC's
- Serial Based Devices
- Serial and Internal Motion Controllers
- Eason PC104 Cards

There are several components to a Winbuild 2000 Application:

1. Device drivers to aid communication.
2. Tags (or variables) that will hold your data.
3. HMI components that the user will see/interact with.
4. BASIC code to customize the application.
5. Tasks and Subroutines to further increase functionality.

HOW DOES IT DO IT?

Winbuild 2000 is a software package installed onto your PC, and allows you to build your HMI screens in a point & click graphic environment. All of your screen layouts, drivers and code are programmed and configured at your PC.

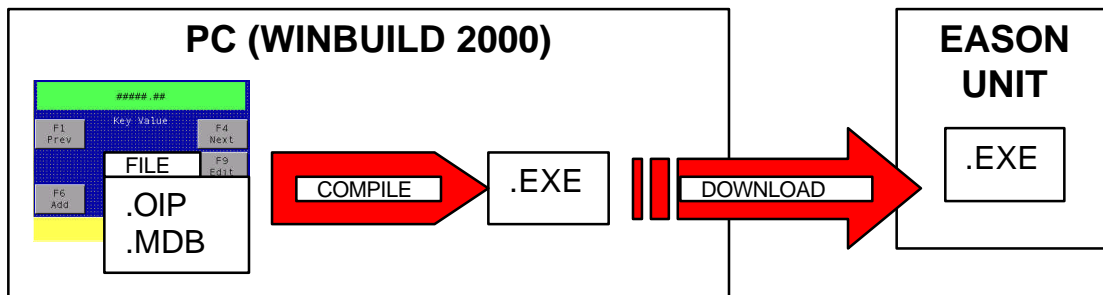


The program is saved into two files- an .OIP and .MDB file that contain all the information about your program.

The .OIP file contains the graphical screens and the BASIC program portion of your application and the .MDB file contains all the data information

When it comes time to load your program into the display unit, Winbuild 2000 will COMPILE the program into a compact, executable file.

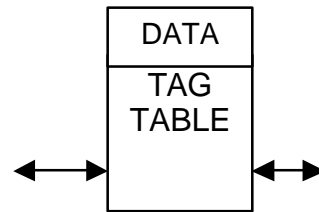
This file is then downloaded to the Eason unit, replacing the previously existing application on the unit's flash drive. Upon reboot, the Eason unit will then execute the updated executable files.



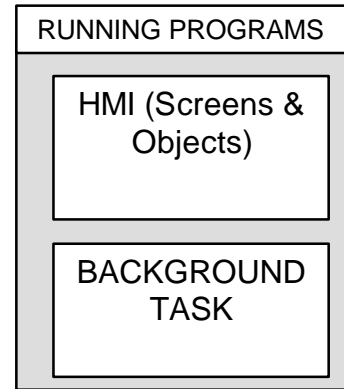
WINBUILD 2000 PROGRAM EXECUTION MODEL (HOW DOES IT ALL RUN)

Below is an execution diagram for a typical Winbuild 2000 program.

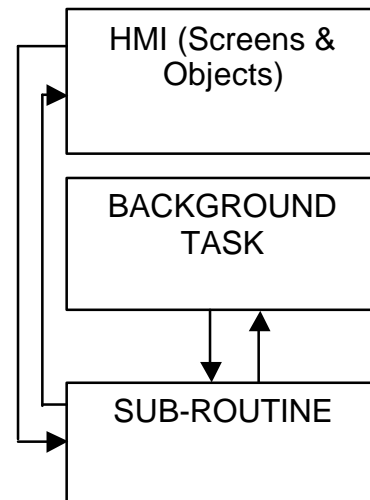
- The TAG TABLE serves as the central place where all tags (variables) are stored.



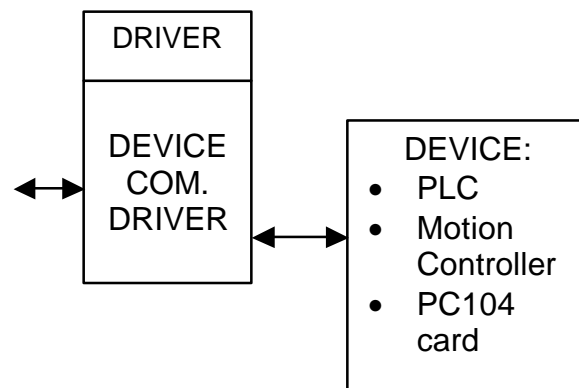
- The HMI and background tasks execute simultaneously. (Like separate programs.)

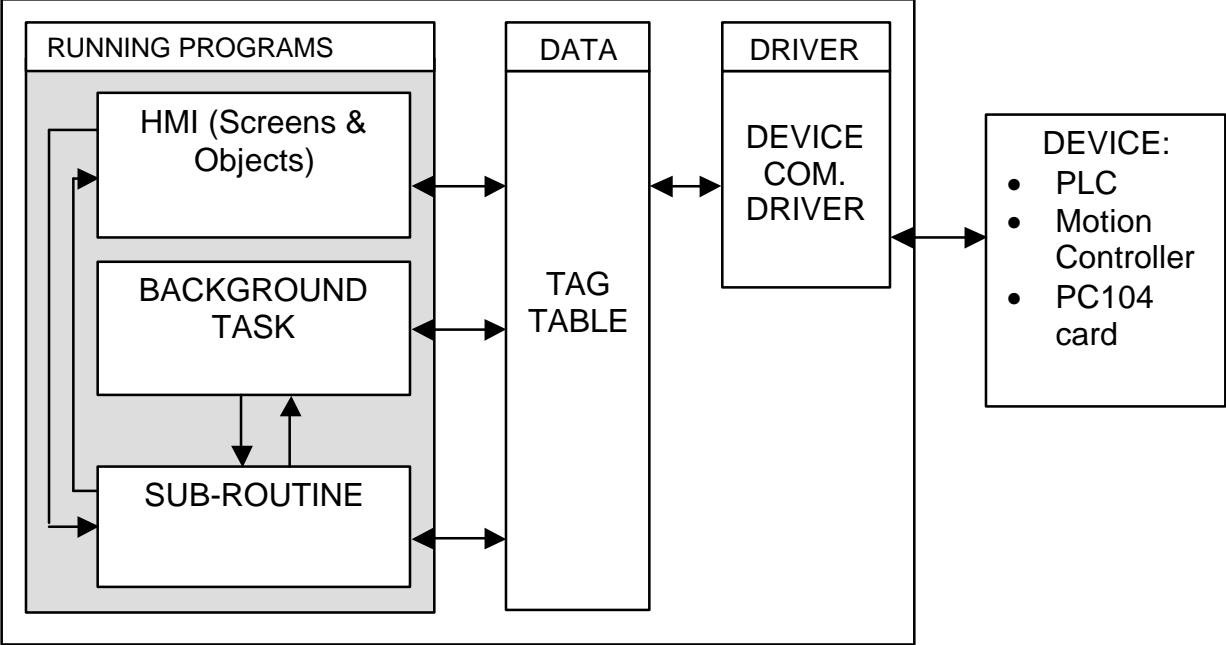


- The HMI and background tasks can call subroutines, which will execute and then return back to the point where they were called.



- The Device Drivers take care of taking values in/out of the Eason unit to your devices. (Including your Motion Controller, PLC, serial devices, and PC104 cards.)





DRIVERS

Winbuild 2000 has many drivers to aid building your HMI and control program:

- Internal Volatile
 - Used for temporary storage
 - Resets to specifiable default value on program reset

- Internal Non-Volatile
 - Permanent retentive storage
 - Can be initialized to a default value by WinBuild

- Recipe Drivers
 - Collections of Non-Volatile Data

- Motion Controller and PLC's
 - Specify PLC type and communications parameters

- PC104 Expansion Cards
 - Configure Hardware and Tags for Internal I/O and PC104 Cards

TAGS

Tags are variables that will be used for data entry/display and control in your program. When constructing your program, you should identify what kinds of data you want the interface to handle, and what variables (or constants) it will hold. These items are put into the TAG TABLE. Note that some variables will be stored in the Eason, and thus their source will be INTERNAL however some other variables may come from your PLC, or Motion Controller, and will be sourced to them.

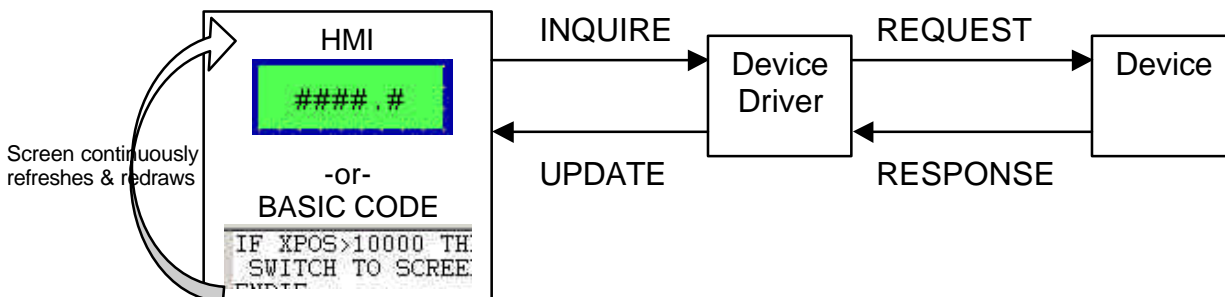
For example if you want to store a variable in battery-backed memory on the Eason, its TAG TABLE entry would look like this: (you can even store a DEFAULT value)

Tag Table (double-click on column headings to sort)					
	Tag Name	Source	Address	Type	Default
1	JogSpeed	Internal Nonvolatile using directory 'B:'	R1	Real	1
2	IndexLength	Internal Nonvolatile using directory 'B:'	R2	Real	3

In another example if you are reading the x-axis position from a Galil motion controller, the tag entry might look like this:

Tag Table (double-click on column headings to sort)					
	Tag Name	Source	Address	Type	Default
1	XPOS	Galil DMC-1000/1500 [0] on 1	TPX.DW	Double Word	0

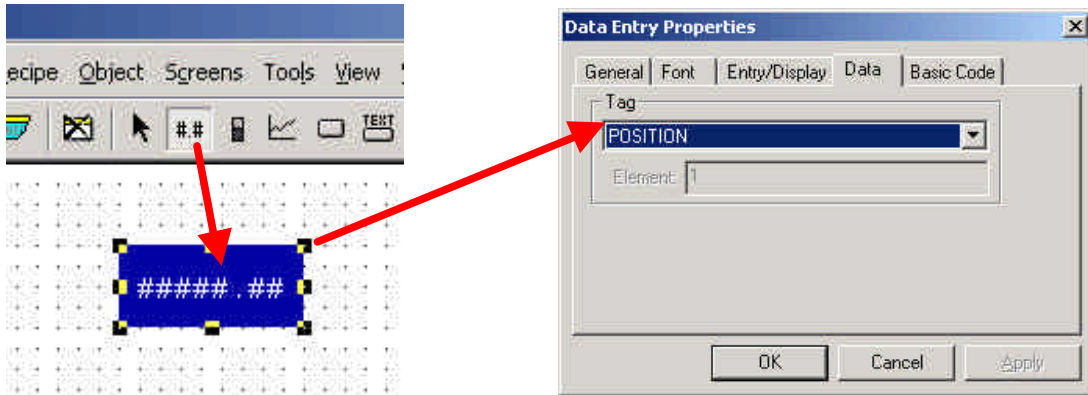
The device driver will now take care of requesting the variable from the Galil, and making it available in your Eason program. You can now place the tag "XPOS" onto the screen of the Eason, or use it in some BASIC code.



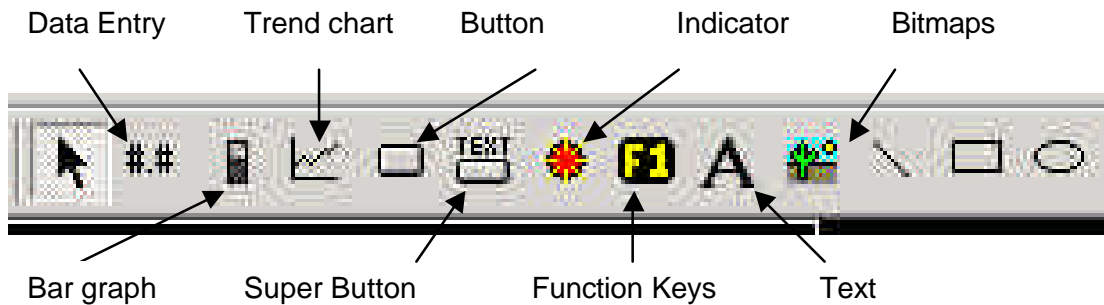
HMI

After having added drivers and created variables for your project, the next step is to create the Human-Machine Interface (HMI) part of the project. Winbuild 2000 has many HMI tools to use to create a clear, informative and dynamic interface.

The most common of these elements is the DATA DISPLAY object. This allows you to place your variables onto the screen of the Eason.



Other objects that help you to build your HMI:



See the Winbuild 2000 Help system for more information regarding each of these components.

BASIC CODE

By using BASIC code in your application, you can customize functionality and greatly expand the capabilities of the Eason unit.

BASIC code can be put into any object in your program (button, data entry, text box, even screens). This adds new functionality to your program, and allows very customized actions to occur. Perform math, logic, string functions, control I/O and communicate to devices easily with the available codes.

BASIC code can even be used to change the properties of objects- say to change a text message or flip the color of a button.

See the "Winbuild 2000 Technical Reference" for the full listing of BASIC codes available in Winbuild 2000.

TASKS

A background task is code that runs independently in the unit, in a multi-tasking environment. The task always runs, regardless of what screen is currently being viewed or other code that is being run. Tasks can call subroutines, and the task shares variables with the Tag Table and also shares the device drivers you have added to your project.

- Run Multiple Programs at the Same Time
- Monitor or control a Process While Providing Reasonable GUI Responsiveness
- Share Drivers and Tags with Multiple Tasks
- Make Your Program Simpler

The background task runs in a continuous loop, and is great for creating code that performs:

- Alarm Process Monitoring
- Machine Control
- Linking I/O and Data to an On Board Motion Controller
- System diagnostics and error checking

Tasks give you benefit of:

- BASIC programs that can be very large.
- Update rates for small tasks are 30-50 msec.
- Sharing drivers, com ports, tags, I/O etc.
- Organizing and simplifying your project.

There are downsides to tasks:

- Can slow down user interface
- Improper use of communications can cause a bottle-neck
- Not “real-time”, multitasks are timesliced with hmi, tasks & other code.
- Limit to 3 tasks on 6” screen products and 5 tasks on 10” screen products (CPU dependent)

SUBROUTINES

Subroutines are smaller sections of code that can be called from anywhere in the program. Any object, button, screen or task can run a subroutine. Subroutines are useful in creating commonly used snippets of code, and only needing to program them once- and then call them from anywhere in the program as you need to. Subroutines need to completely finish before returning to the previous code that called them.

GUIDELINES (OPTIMIZING)

It is best to limit the number of background tasks in your project to 3 or less. Each task carries a certain amount of fixed overhead, and thus more tasks means more overhead. One large task is preferable to 4-5 small tasks.

Place timing critical code into a background task. This will assure the code gets run most efficiently without being affected by the HMI foreground activities.

Try combining tasks, or moving certain pieces of code into subroutines to be called occasionally from screens or from your other tasks when needed.

Tags also take up memory in the unit, and should be limited to less than 500 volatile tags.

Space on the Non-Volatile battery-backed drive is limited to 128kB, and thus when constructing your program be aware there is a limit to the number variables that can be stored. (See 'Supported Data Types' in the HELP system or 'Technical Reference'.)

GETTING STARTED (HOW DO I USE IT?)

After reading through this introduction, move onto the first lab. This lab will help you build a simple HMI program using several Winbuild 2000 features.

By moving through the labs, you will be familiar with all the features and capabilities of Winbuild 2000.

1. **Introductory Features**
2. **Advanced Topics (and BASIC codes)**
3. **Serial Communication**
4. **PLC Communications**
5. **Alarms**
6. **Recipes**
7. **I/O**
8. **Galil drivers**

Other resources to help you in your Winbuild 2000 programming:

1. **Winbuild 2000 User's Guide:**
Outlines how to install software, get started and explains some of the basic features.
2. **Winbuild 2000 Technical Reference:**
Documents driver features, settings, and wiring to devices. Also includes full BASIC reference, and other misc reference material.
3. **Sample Programs:**
In your c:\programfiles\Winbuild2000V4\Samples directory there are many programs to help show you
4. **Technical Notes:**
See www.eason.com to review technical notes- there may already be a document helping you through a common question or problem.
 - TechNote79: Ethernet Networking Setup
 - Technote72: TouchCover (TCVR3) Install Procedure
 - TechNote85: Battery Change Procedure

PLAN YOUR PROJECT

After you have completed the labs are, use the Project Planning Guide to help you outline your application.