

PYXIS 2

USER GUIDE



skewworks

Copyright ©2010-2011 Thomas W Holtquist

www.skewworks.com

Version 2.2

Table of Contents

[1.0 About this Document](#)

[1.1 Change Log](#)

[1.2 Intended Audience](#)

[2.0 Introduction](#)

[2.1 Advantages](#)

[3.0 Supported Devices](#)

[3.1 Boards](#)

[3.2 Screens](#)

[3.3 Components](#)

[4.0 Getting Started](#)

[4.1 Installing Pyxis 2](#)

[4.2 Installing the Updater](#)

[5.0 Using Pyxis 2](#)

[5.1 The Desktop](#)

[5.2 The App Store](#)

[5.3 File Finder](#)

[5.4 Settings](#)

[5.5 System Updates](#)

[5.6 Installing Applications](#)

[6.0 Developing Applications](#)

[6.1 Required Structure](#)

[6.2 Creating an Icon](#)

[6.3 Creating an Install Package](#)

1.0 About This Document

1.1 CHANGE LOG

The following changes are from Beta 2.2 to the first production release.

- Applications now accept start-up parameters
- Multilevel modal support fixed
- Proper casing is now used on control names
- App Store added
- File Finder and Picture Viewer have been incorporated into the kernel
- The kernel has been split: Drivers, AppearanceManager, ApplicationKey, Colors, FontManager, and Settings manager are now in Kernel.EXT
- Forms now automatically register themselves in applications; you no longer need to call `API.CreateForm()`
- An application installer utility has been created
- You can now add file associations through `API.MyFiles.AssociateFile`
- Pyxis 2 will now check for updates on start-up (if set)
- Forms now support auto-scrolling

1.2 INTENDED AUDIENCE

This document is intended for those with a basic working knowledge of C#, .NET Micro Framework and Microsoft Visual Studio. It is intended to get users of, and developers for, Pyxis 2 up and running quickly.

2.0 Introduction

Pyxis 2 is an Operating Environment (OE) written in C# on the .NET Micro Framework (NETMF). The aim of this project is to provide a means for developers to create robust NETMF applications with greater speed and ease while giving users a common interface to launch applications from several different vendors.

2.1 ADVANTAGES

- Over a dozen built-in controls
- Managers for common tasks (Drives, Files, Fonts, Networking, etc)
- AppDomain management
- Menuing
- 100% WPF free rendering for greater speed
- Multiple device support
- 1 file installs for applications
- Much more

3.0 Supported Devices

One a whole NETMF supports a wide variety of devices and can be ported to various chips. Pyxis 2 natively supports the [FEZ Cobra](#) and [ChipworkX](#) systems from [GHI Electronics](#). Since Pyxis 2 is based on the .NET Micro Framework it can easily be modified to work with any other device that supports LCDs and has sufficient RAM/EEPROM.

3.1 BOARDS

- [FEZ Cobra](#) – A mid-level device that supports 3 LCD sizes and has a custom enclosure available.
- [ChipworkX](#) – This high-level device comes with a 480x272 LCD attached, built-in MP3 decoder, stereo speakers, RLP access and much more. Significantly faster than the smaller FEZ Cobra; ChipworkX is the recommended device for Pyxis 2.

3.2 SCREENS

While Pyxis 2 has built-in support for all 3 screen sizes from [GHI Electronics](#) only the 320x240 and 480x272 flavors are recommended as the 800x480 screen eats up plenty of memory and is only supported on the smaller of the 2 boards.

3.3 COMPONENTS

- **VS1053** – While not actively used by Pyxis 2 this MP3 decoder is built into the [ChipworkX](#) module and a driver is present for your use.
- **Wii Controller Interface** – (No direct link available at time of writing documentation) This component allows you to connect a Wii Nunchuck controller to your FEZ device.

4.0 Getting Started

While Pyxis 2 works best with a physical device, you can test it out inside Visual Studio's emulator. This way you can see if Pyxis 2 is right for you before investing in a physical device. The steps provided below are for working with an actual device, to use the emulator instead simply select Emulator instead of USB when you go to deploy.

4.1 INSTALLING PYXIS

In order to begin using Pyxis you must already have Visual Studio installed (free C# version: <http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express>).

You will also need the latest SDK drives from both GHI & Microsoft.

<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=cff5a7b7-c21c-4127-ac65-5516384da3a0>

<http://www.ghielectronics.com/downloads/NETMF/GHI%20NETMF%20v4.1%20SDK.zip>

Once you have the studio and drivers installed it is time to grab the latest code from CodePlex: <http://pyxis2.codeplex.com/releases/view/57612>.

Extract the files and folders into your Projects directory and open the Pyxis2.sln file.

From inside Visual Studio select Project -> Pyxis2 Properties... from the menubar. Next select the ".NET Micro Framework" tab. Here you can select the device you wish to program with Pyxis 2. Once you have your device selected press F5 to deploy.

Once your project deploys to the device it will automatically reboot and disconnect from studio (even though the studio will show you're still connected). This happens because the first time you enable SystemUpdates on a GHI board it

splits the data section into two sectors; one for the bootloader and the other for your application.

Press the stop button and then the play button to redeploy Pyxis 2 to your device. This time you will see the Pyxis boot screen. If you are using an LCD other than the 320x240 this screen *will* look a little strange, it's nothing to worry about.

After a few moments a screen will appear asking you to select your screen size; do so and click "Set Resolution and Reboot". Your device will once again reboot and become disconnected from the debugger.

When your device finishes resetting you will be shown a blue screen with a crosshair, tap on the crosshair and subsequent crosshairs to calibrate your touch screen.

(Note: These steps do not occur for the ChipworkX system).

Congratulations, Pyxis 2 is now installed properly on your device!

4.2 INSTALLING THE UPDATER

The very first time you attempt to apply a system update to Pyxis 2 your device will reboot and stay on the bootloader icon. This is because it does not have the updater installed yet.

Open Pyxis2.sln in Visual Studio, right-click on "Pyxis2Updater" and select "Set as StartUp Project". Now press F5 or the green play button to deploy.

Once you have deployed the project to your device the updater will apply your updates and reboot.

At this point you should *never* need to program Pyxis 2 on your device again. The updater is capable of not only updating itself and Pyxis 2 but also applying new firmware updates to the device itself.

5.0 Using Pyxis 2

Pyxis 2 is an intuitive icon-based operating environment. Navigation is performed through use of the touch screen. Pyxis 2 is also capable of performing with or without an attached drive. However, the functionality of Pyxis 2 without a drive is severely limited and not recommended.

5.1 THE DESKTOP

The first time you start Pyxis 2 you will likely be greeted by a prompt asking you to prepare your attached drive (if one or more is present). Selecting “Yes” **will not** cause you to lose any information already stored on your drive; instead Pyxis simply adds a few new folders and directories that it requires to function to its fullest.

After you have chosen to prepare your drive (or not) you’ll see a screen much like the one below:



This is your desktop, from here you’ll be able to launch applications or access various parts of Pyxis through the menu system.

5.2 THE APP STORE

From the App Store you'll be able to browse and download Pyxis 2 applications from the web (internet connection required).

When you first launch the App Store you'll be presented with a list of available applications sorted by date added. You can also search for a specific application by tapping App Store -> Search.

Once you have found an app that interests you tap it to bring up the information/download screen (see below). From here you can either choose to download the application or go back to the previous screen.



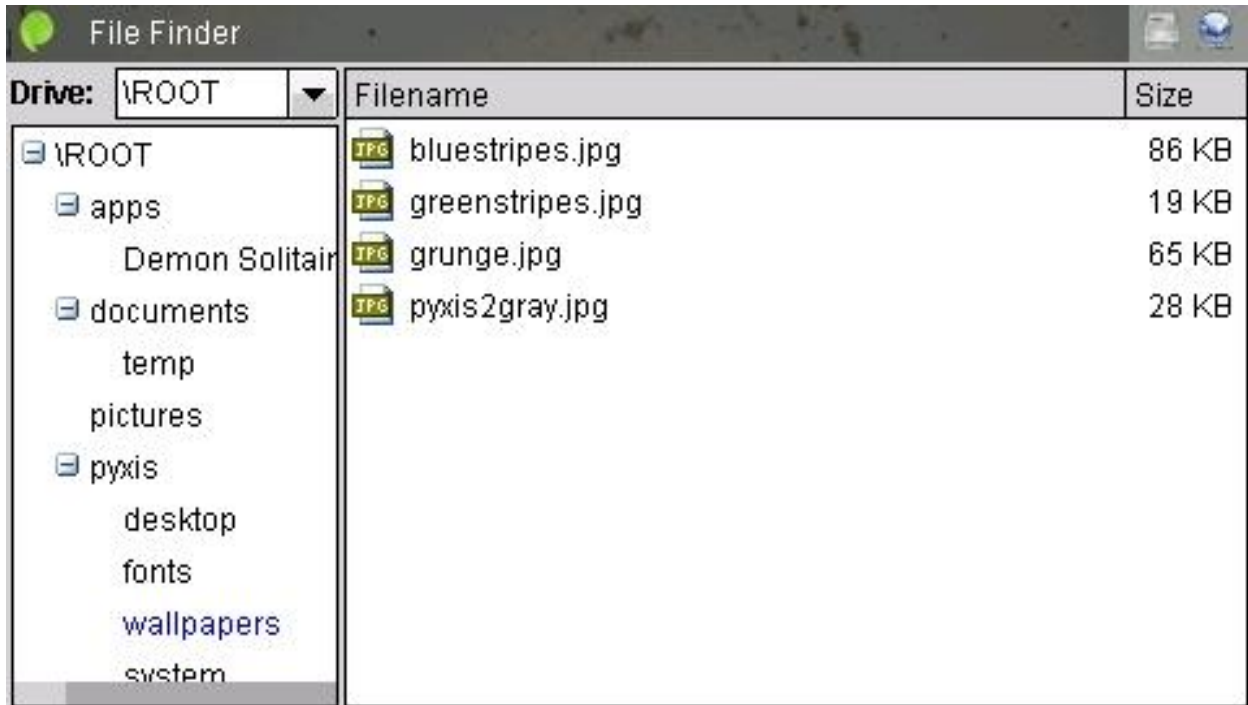
FREE Download

Back

If you choose to download the application Pyxis 2 will download and install it. After installation of the application is completed you'll be asked if you wish to create a shortcut on the desktop. If you select "Yes" the next time you view your desktop there will be a new icon there that points to the application you just installed.

5.3 FILE FINDER

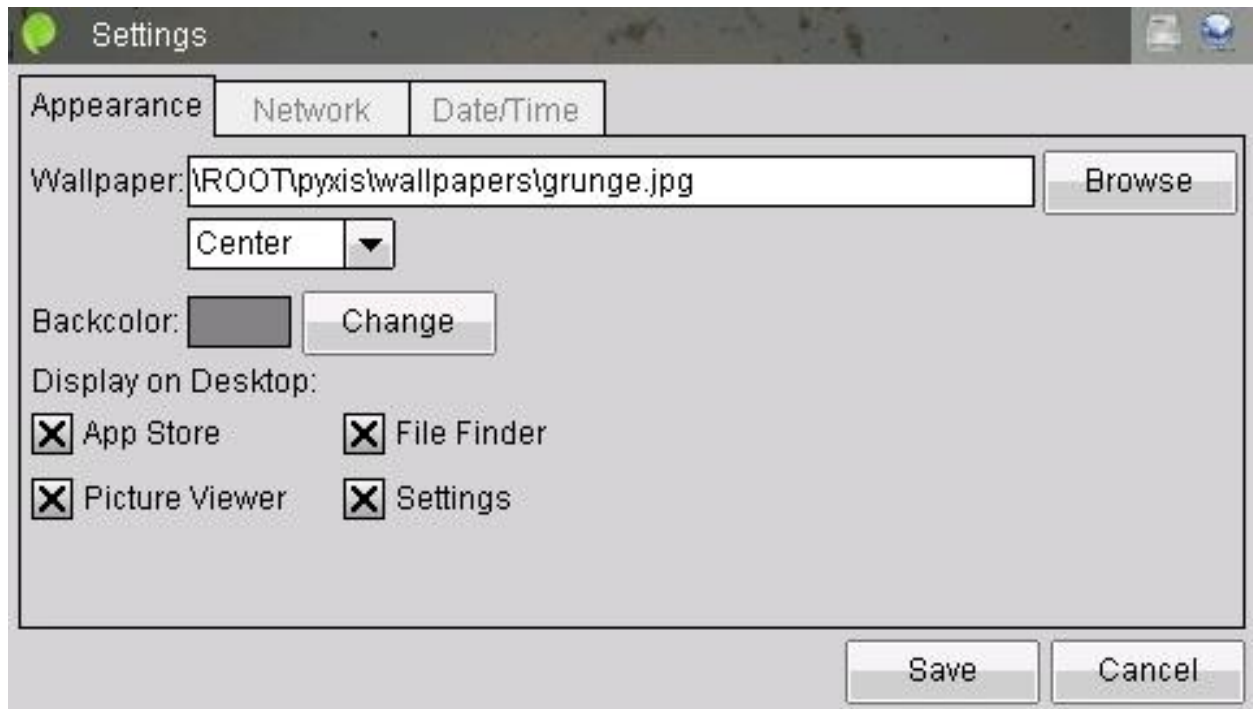
An essential application, File Finder, is built directly into the Pyxis 2 kernel. This application allows you to browse your drive(s) and perform various functions such as creating directories, copying/renaming/moving files, and the like.



File Finder is also aware of all your file associations and will automatically open a file with its associated application (if one is present). Developers can even set associations from within their applications.

5.4 SETTINGS

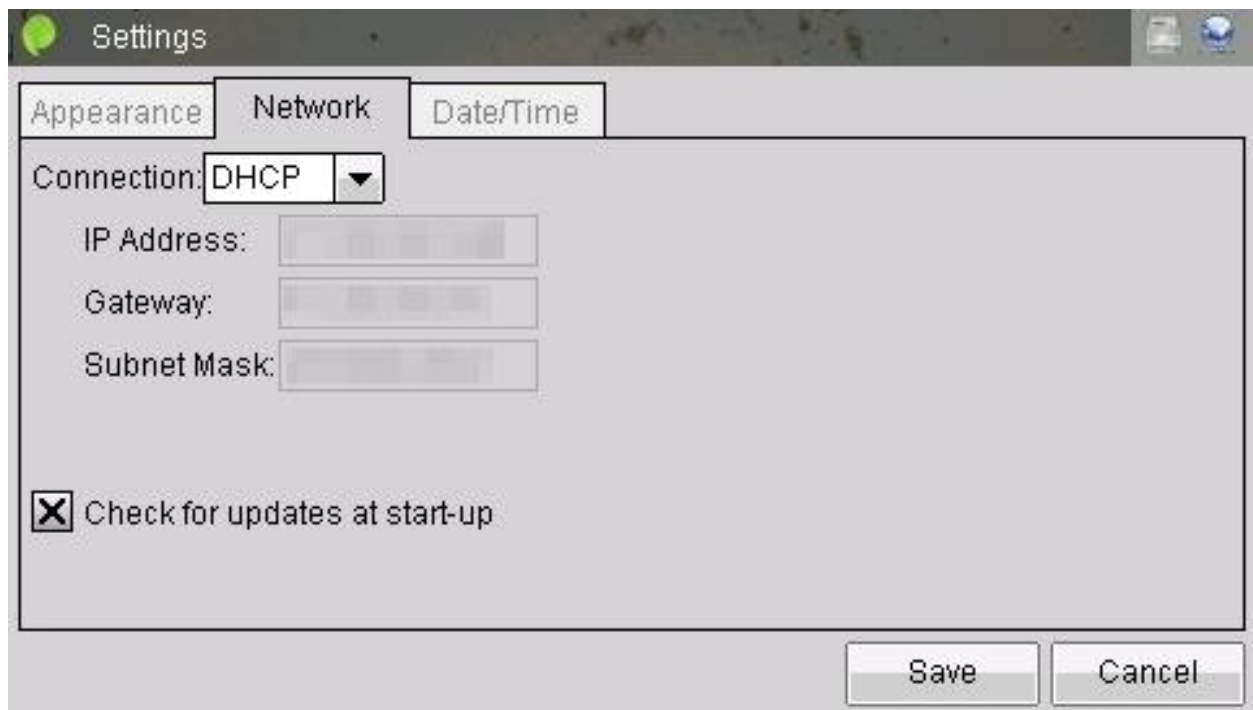
The Settings window allows you to control how Pyxis 2 looks and feels. From inside this window you can set your Desktop properties, network settings, system date & time, and even sound (sound available only on ChipworkX).



5.5 SYSTEM UPDATES

Pyxis 2 is a fully updatable system that is not only capable of updating its own code but the bootloader and firmware as well.

To check for available updates you can simply tap [Pyxis Icon] -> Software Update. If updates are available you'll be prompted to download them. Alternatively you can Pyxis 2 automatically check for updates every time you start the system (as long as an internet connection is available) from within the Settings window.



5.6 INSTALLING APPLICATIONS

In addition to automatically installing applications from the App Store you can install .PIE (Pyxis Installer Executable) files directly from File Finder. Simply select the file from inside File Finder and tap File Finder -> Open as seen below.



Just like with the App Store; after the application is installed you be given an opportunity to create a shortcut on the desktop.

6.0 Developing Applications

Application development for Pyxis 2 is very similar to the development of any other NETMF application. The only real difference is the addition of helpers and how your application is called.

Sample applications can be found on the Pyxis 2 CodePlex page:

<http://pyxis2.codeplex.com>

6.1 REQUIRED STRUCTURE

Applications for Pyxis 2 **must** be created as *Class Library* projects. Additionally there are 2 required methods for an application to run on Pyxis 2.

```
public static AppStartup PyxisApp(ref ApplicationKey key, ref PyxisAPI api, string[] parameters)
```

The above method can be looked at as the same as a Main() in a regular application. This is the entry point for your application and as you can see you are passed a few items.

The first item `ref ApplicationKey key` provides your application with a unique key within Pyxis 2. You need this key to properly terminate your application manually. This object also contains your applications AppDomain and the directory your application was called from.

The second item `ref PyxisAPI api` is a reference to the Pyxis 2 API object. You can use this object to access all sorts of Pyxis 2 features such as prompts, file selection, available drives, etc.

Finally you're provided with `string[] parameters`. This string array provides any parameters that were passed when launching your application and will be null if no parameters were passed.

You'll also notice that you are required to return an `AppStartup` object. This object tells Pyxis details about your application such as its name, menus and start-up form. If you have a lot of work to do when starting your application we suggest

creating a thread inside your `PyxisApp` function to prevent the callback from taking too long.

Of course part of any GUI application is Forms. Forms are created like any other control; however the visible property has little effect on forms. The main form for your application should be assigned to `PyxisApp.startupForm`. In this way Pyxis 2 will automatically display this form when your application launches.

If you wish to have multiple forms you can switch between them at anytime by calling `api.ActiveForm = [yourform]`.

The second required function returns your icon and is seen below:

```
public static AppIcon PyxisIcon()
```

The `AppIcon` object provides a title and icon for your application. This title will be used when naming any desktop links to your application. The `.icon` property is a byte array and **must** return bytes for a P2I image.

P2I images can be created using the Pyxis 2 Image Converter application.

5.2 CREATING AN ICON

Pyxis 2 applications are required to return the bytes for a P2I image; which is a custom image format that allows Pyxis 2 to display alpha blended icons with minimal processor delays. The Pyxis 2 Image Converter application is included in the source for Pyxis 2.

When creating icons; keep in mind that you're only allowed a limited number of alpha-blended pixels. If your icon exceeds the number of allotted alpha pixels the application will let you know and abort converting the image.

Once you have your image simply add the P2I image to your project's resources.

5.3 CREATING AN INSTALL PACKAGE

InstallMaker is an application that is included with the Pyxis 2 source download. This application allows you to take your application and create a single-file install. This is an optional step.

To begin open the InstallMaker and give your project a name. This name will be used in the filename as well as displayed in prompts to the user when installing your application.

Next you'll need to select your project's PE file. This can be found in your project's bin\release\le folder. There may be other PE files in this directory but you need to only select the file for your application.

Once you've selected your PE file you need to select an output folder for the PIE to be created in.

At this point most applications are ready for you to click "Build" in the menu. If, however, your project requires additional files or directories you can add them now.

All files and directories are added inside the PIE file and replicated on the device during install. This is useful for dependency PE files and resources such as images.

You can optionally add basic file associations here; however we recommend that you take care of associations inside of the application itself so you can dictate the target path and supply an icon as well. (Icons are required to be 24bit 16x16 1-layer GIF or BMP images).