TraceSystemsInc

Secrets of Successful USB Developers

Dr. Bob Miller, Trace Systems Inc

The Intention of This Guide

We have always felt that USB design work was harder than it needed to be. We have worked for over a decade to make our users more powerful -- by providing them with extraordinarily productive tools, knowledge, and personal help when they need it most.

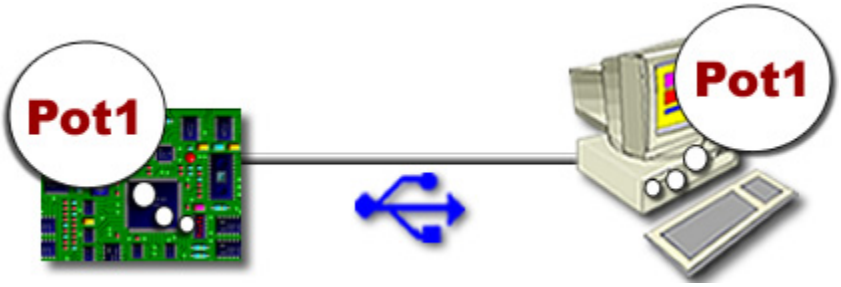
This guide aims to do that same sort of thing: to inform you of resources, tips, and a mindset that will make YOU more powerful too.

One of the first things that people ask us about is "what sort of commercial demo board and processor should I get started with?" A big part of this guide provides you with detailed information on boards that we have used in our work, and which have helped our customers as well.

Another section of this Guide tells you about some optional tools that we and our customers highly recommend -- tools that will help you in your USB development work, and in the rest of your work as well.

The last section in this guide is about tuning up the most important tool you already have: the one between your ears. Our customers have shown that sometimes, a little bit of change in their thinking can give them huge gains in productivity. We'll share those tips with YOU.

Known Good USB Demo Boards



Please understand that HIDmaker FS 2 and HIDmaker 32 are NOT limited to use with just certain demo boards. Both of our HIDmaker products are designed for engineering professionals who are designing custom hardware products of their own, so **HIDmaker projects can work on *any* board that functions properly over USB**, and which contains one of the USB-capable PIC processors that the product supports.

However, since early revisions of custom boards often have problems, **we always recommend that our customers use HIDmaker for the first time on a known-good commercial demo board.** That way, you can be sure that your first experience with HIDmaker FS 2 or HIDmaker 32 will be as simple as possible, and will allow you to concentrate on learning about HIDmaker and USB -- without the extra complication of trying out and possibly having to debug an untried PC board at the same time.

This is NOT meant to be an exhaustive list: this is NOT a list of the "only demo boards that will work with HIDmaker" - far from it! Just about any properly designed USB demo board will work with HIDmaker - as long as that board is properly designed and is in good working condition.

Generally, if you can get any of the Microchip USB demo projects to run on your board, AND the board contains one of the USB processors that HIDmaker FS supports, then that board should work just fine with HIDmaker FS. That's our definition of the term "known good."

The list below is a list of the USB boards that we currently have in our lab, and which we have actually used to test HIDmaker FS and HIDmaker 32 projects. If you are a registered HIDmaker FS 2 or HIDmaker 32 user who is eligible for Technical Support, and you are having a problem that can be demonstrated by running your project on one of these boards, then we can test your code right in our lab right away. That's a really strong reason for you to get one of these boards: so we can support you better!

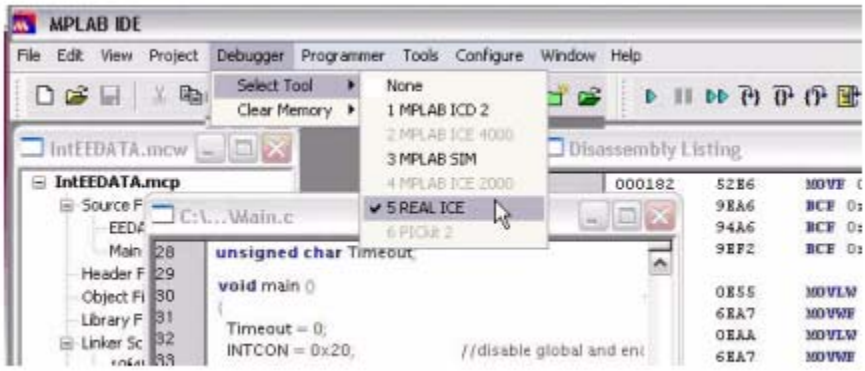
Microchip 8-Bit USB Demo Boards

PICDEM FS USB - Microchip

The full speed **PicDem FS USB board** (MIcrochip part number **DM163025**, shown above) contains an 18F4550 in a 44 pin surface mount package. The interactive controls that your project can use on this board include 4 LEDs, 2 pushbuttons, and a pot. (The extra pushbutton is connected to the /MCLR line, so you can reset the processor.)

The black RJ-11 modular connector on the upper left of the board is the interface to full sized Microchip programmer / debugger devices, like the circular ICD3:

or the more powerful **MPLAB REAL ICE In-Circuit Emulator** (Part Number: **DV164035**) shown below:



PIC18F87J50 Plug In Module - Microchip

This little 2" x 2" board (Microchip **MA180021**) contains a PIC18F87J50, which is one of the largest 8-bit USB processors that Microchip makes since it contains 128 KB of program memory and 3904 bytes of data memory in an 80-pin package.

The **PIC18F87J50 Plug In Module** board shown above can be used either as a standalone demo board (providing only 1 pushbutton and 1 LED that you can interact with when used as a standalone board), or it can be plugged in to several larger Microchip demo boards, as we'll show presently.

The **6-pin header located on the lower right corner of the Plug In Module is a programming connector** that works directly with Microchip's PICkit3 In Circuit Debugger:

If you have one of the more powerful Microchip debuggers like the ICD3 or Real ICE, you need to use the following adapter to connect the debugger to the inline 6-pin programming connector on the **PIC18F87J50 Plug In Module** :

The **PIC18F87J50 Plug In Module** is called a "Plug In Module" because it can be plugged into a larger demo board like the **PIC18 Explorer board** (Microchip product number DM183032) shown below:

When the **PIC18F87J50 Plug In Module** is plugged into the larger **PIC18 Explorer Board**, *the combination provides you with more resources to interact with:* 8 LEDs, 2 pushbuttons, a pot, and a temperature sensor. You can also plug in several PIC Tail daughter boards, so you can interact with a large variety of other resources.

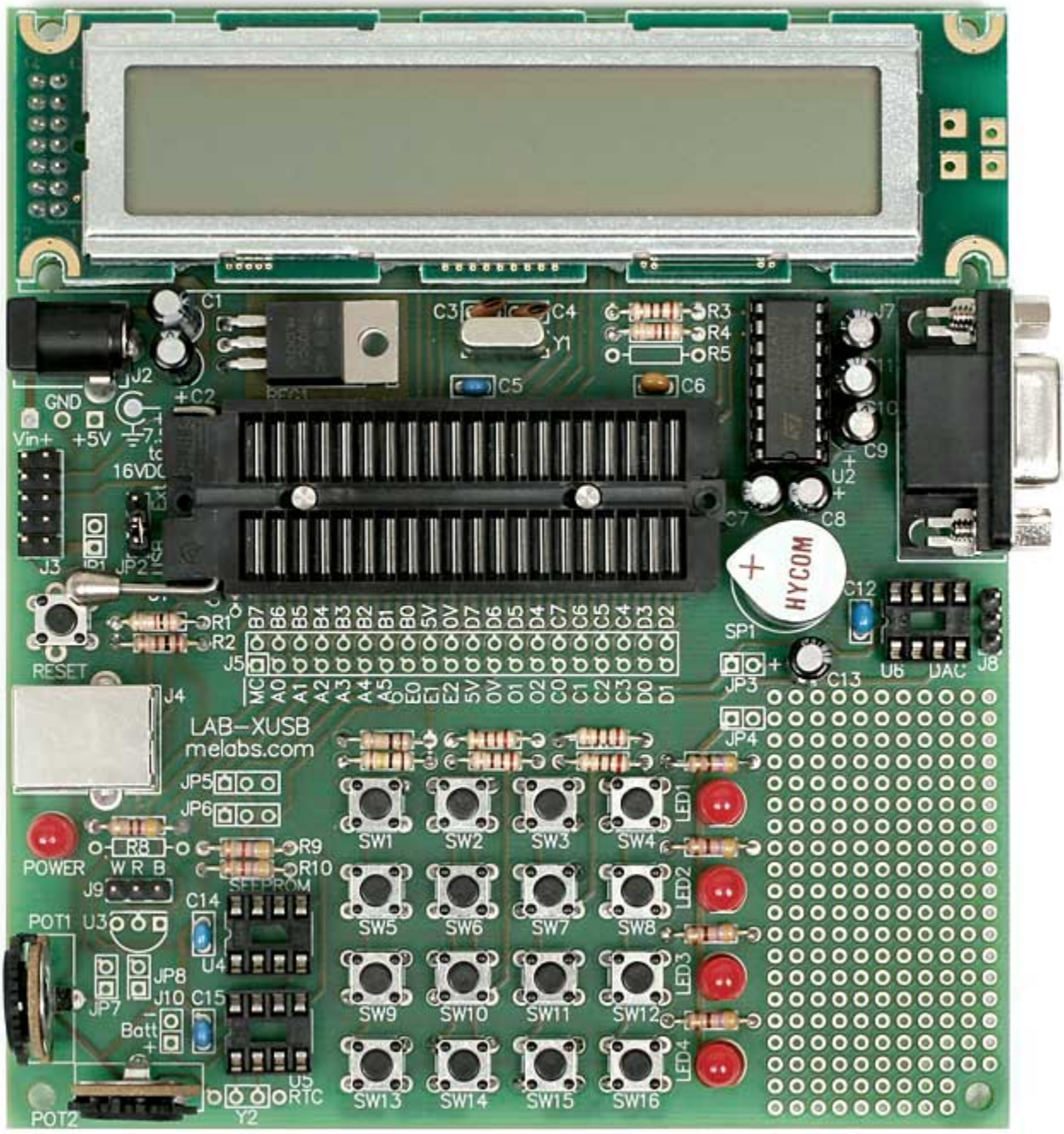
However, even if the **PIC18F87J50 Plug In Module** is plugged into another board, you can only program the PIC processor on it through the inline 6-pin programming connector on the Plug In Module itself.

The Plug In Module can also be used with the older (and officially obsolete, but still available on the web) **HPC Explorer Board** (Microchip product number **DM183022**), shown below:

As with the **PIC18 Explorer Board**, plugging the **PIC18F87J50 Plug In Module** into the **HPC Explorer Board** gives you more resources to interact with.

microEngineering Labs USB Demo Boards for 8-bit PIC18F Processors

LABX USB - microEngineering Labs



www.tracesystemsinc.com

The **LABX-USB**, shown above, is sold assembled and tested, and contains a 16 key keypad, 4 LEDs, 2 Pots, an LCD display, and lots of other goodies. This is our favorite USB test board, because it has so many things on it that you can interact with. You can find complete documentation (and even a schematic) online at the microEngineering Labs web site.

What we especially like about this board is that it has a lot of controls that you can interact with, so you can test all sorts of projects, and see how well your project works in real time.

* Notice the 2 small trim pots at the lower right corner of the board. You can easily read those pots in your HIDmaker PIC code and send the values to the PC in real time. If you want, you can even use this setup to make and test an equivalent of a joystick project. With a few changes to the code, you can make the board think it is a mouse instead.

* Also, notice the 4x4 array of pushbuttons. This forms a 16-key keypad, with row - column scanning connections like you will find inside the keyboard for your PC. If you want, you can make this board behave like an official USB keyboard (although with fewer keys).

* This board also has sockets for serial EEPROM, and even a D/A converter, so you could make it create analog waveforms under USB control!

Note that this board, and the PIC PROTO USB board (which we describe next in our list) is intended by melabs to be used with a 10-pin programming header that is compatible with microEngineering Labs programmers, like their popular U2 Programmer (which, by the way, is a HIDmaker device). The picture below shows the U2 Bundle, which includes the actual U2 Programmer (on the left, with the USB connector on it) and a programming adapter board that contains a Zero Insertion Force socket.

But the connection that we *really* recommend for use with this LABX-USB board is this next little gem from MicroEngineering Labs:

microEngineering Labs offers this handy adapter called the **ICD to 10-pin Ribbon Adapter** (shown above), that bridges the Microchip RJ-11 modular cable programming interface to the microEngineering Labs 10-pin header programming interface. With this adapter, you can connect a real Microchip Programmer / Debugger to your LABX-USB board -- like Microchip's **MPLAB® ICD 3 In-Circuit Debugger System** (Microchip product number **DV164035**), shown below:

Using the ICD3 will allow you to be able to set breakpoints to stop your PIC code when it reaches a certain point, and let you inspect the values of important variables and data structures. **If you are serious about developing real products that customers will use, you really need to have a real debugger in your lab.**

PIC PROTO USB - microEngineering Labs

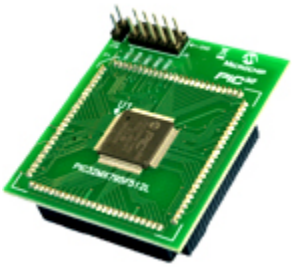
The low cost **PICPROTO USB board from MicroEngineering Labs** provides 2 pushbuttons, 2 LEDs, and 2 Pots that you can interact with, and is sold as an inexpensive bare board for you to assemble yourself. You can find complete documentation (and even a schematic) online at the microEngineering Labs web site.

Like the LABX-USB above, the PICPROTO USB uses the microEngineering Labs 10-pin programming header, which can be used with Microchip programmers and debuggers by using the microEngineering Labs ICD to 10-pin Ribbon Adapter which we showed above.

Microchip 32-Bit USB Demo Boards

Explorer 16 (100-Pin Version) - Microchip

The Explorer 16 100-pin demo board shown above (Microchip product number DM240001) can be used with Microchip's 16-bit and 32-bit processors. You purchase and install a Microchip Plug In Module (PIM) containing the processor you want to work with. A typical Plug In Module looks like this:



The Explorer 16 100-pin demo board comes with 2 Plug In Modules included: the PIC24FJ128GA010 and the dsPIC33FJ256GP710. (Both of these processors work fine with TCPmaker, but they are useless with HIDmaker 32, because neither of these processors have USB capability.) The Plug In Module shown above is the PIC32MX795F512L processor, which does have full speed USB.

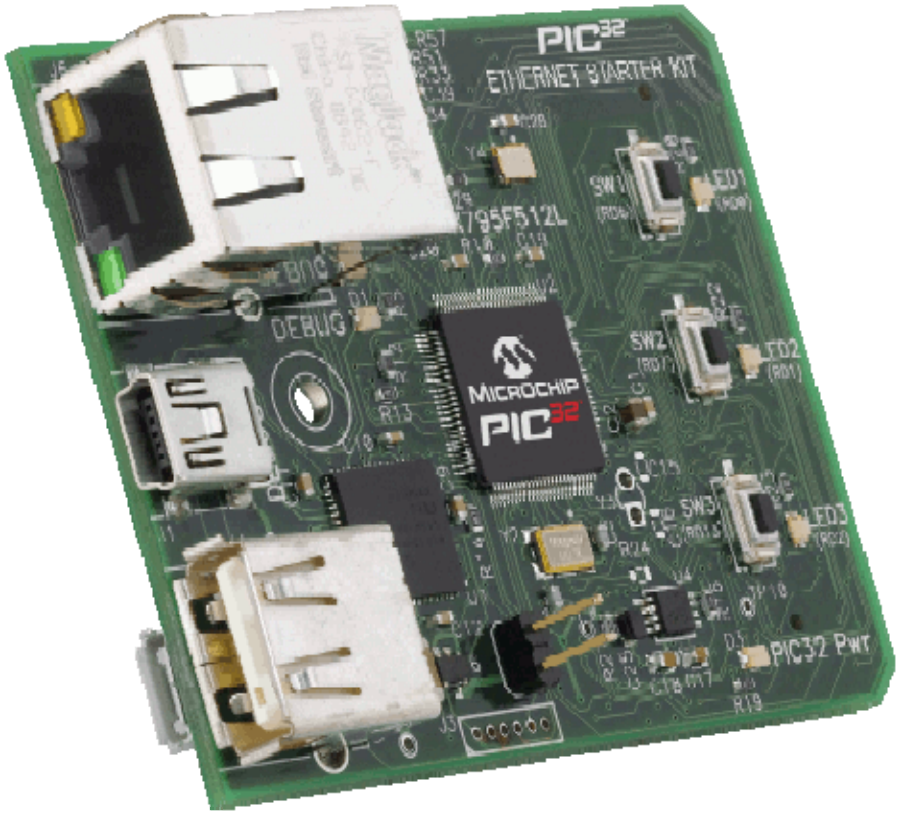
Although the picture of the Explorer 16 board does have a USB connector on it, that connector goes to a PIC18F4550 processor on the board that Microchip had originally intended to be used for programming the processor on whatever Plug In Module was being used. Microchip abandoned this approach, so that '4550 isn't really used for anything on the Explorer 16 board these days.

To actually use the USB module in your 32 bit or 16 bit plugin module, you need this **USB PICtail Plus Adapter** (Part number **AC164131**):

Here's how you use this with HIDmaker 32 and an Explorer 16 with a Plug In Module for a USB capable processor:

- Install the USB capable Plug in Module in the main Explorer 16 board
- Connect the black edgeboard socket on the left side of the **USB PICtail Plus Adapter** to the edgeboard connector on the right side of the Explorer 16.
- Connect a USB mini-B cable into the left-most small USB connector along the bottom edge of the **USB PICtail Plus Adapter** board. The dashed line shows that the components in this zone are for "Device" (= peripheral device) mode of the USB module of the processor. And of course, connect the other end of that USB cable to your PC.

PIC32 Ethernet Starter Kit (or PIC32 USB Starter Kit)



We consider the PIC32 Ethernet Starter Kit or "ESK" for short (Microchip product number DM320004), and its main processor, the PIC32MX795F512L, to be and extremely important development.

This processor has just about everything that you might ever want, and in huge amounts, all on one chip:

- a fast and powerful 32 bit processor,
- 512KB of Flash program memory,
- a hitherto unheard of 128 KB of DATA memory (a big change from earlier PIC devices),
- built in Ethernet MAC circuitry (but still needs an external PHY chip),
- USB peripheral device circuitry,
- USB host circuitry,
- USB On The Go (OTG) circuitry,
- CAN circuitry,
- 6 RS-232 serial ports,
- multiple synchronous serial modules (SPI and I2C),

and the list just goes on and on and on. Yet this processor only costs around $10 in single quantities, and about $7 in large quantities.

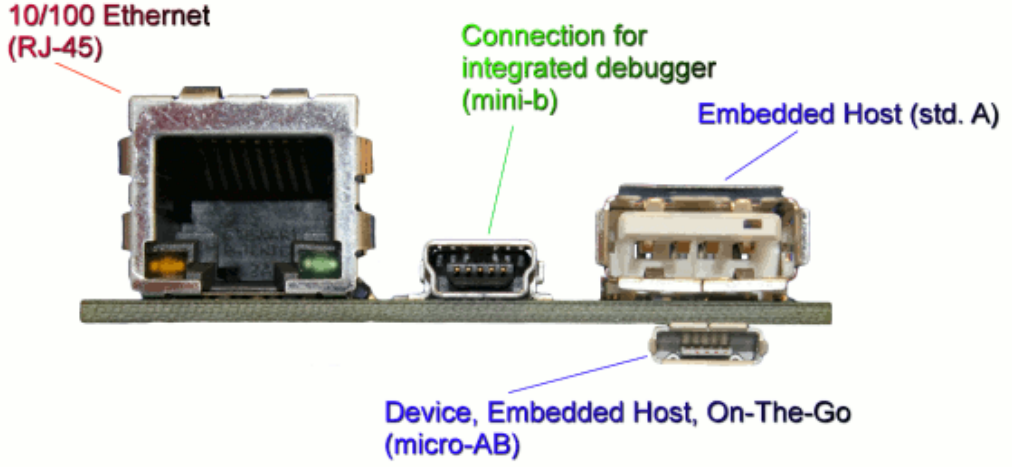
We have customers (who admittedly don't make cost-sensitive applications) that essentially tell us:

> **"We're done. We are going to standardize our whole product line on this one part. We know that we'll never run out of memory, we can do anything we want with this chip, and we won't have to keep figuring out new chips all the time anymore. We can design all our products to one single common processor, and build up a unified code base that can work in all of our products."**

That's a pretty powerful statement.

The ESK board shown above has USB connectors for USB Host and Device / OTG, and an Ethernet connector. You can interact with the 3 LEDs and 3 pushbuttons on the ESK board. (Though I sure wish there was a pot on this thing...) You can operate this board as a standalone board, or you can plug it into other Microchip development boards, as we will explain shortly.

You may notice that the ESK board actually contains 2 PIC32 processors. The secondary processor is only used for programming and debug when you are operating the ESK board as a standalone - the second PIC32 processor ("integrated debugger") is NOT needed in your own design!



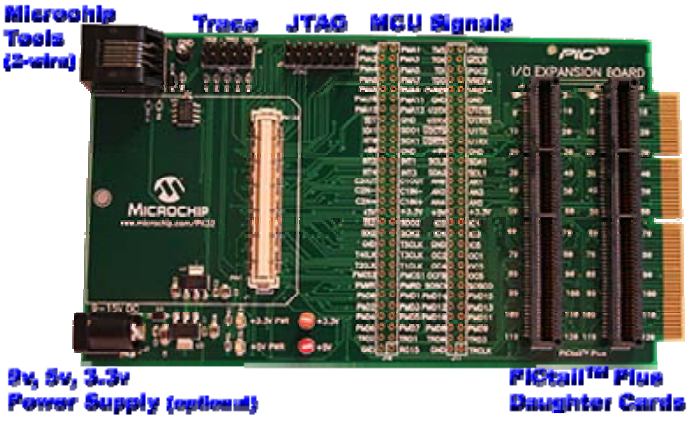
To use the PIC32 Ethernet Starter Kit board with HIDmaker 32, you need a USB On-The-Go cable connected to the one USB connector on the bottom side of the board -- the one labeled **"Device, Embedded Host, On-The Go (micro-AB)"** in the photo above.

Also please note that, when operating the board as a standalone, the board takes its power from the USB cable that is connected to the secondary PIC32 processor.

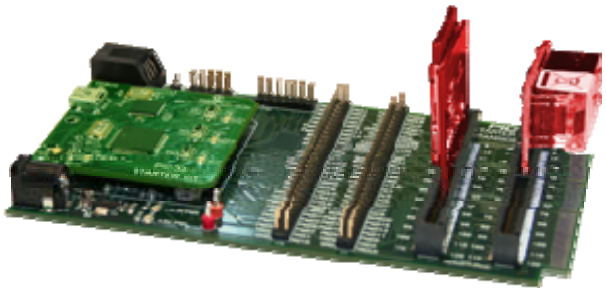
Finally, and most importantly, please understand that, when operated as a standalone board, there is no way to connect your own device programmer directly to the main PIC32MX795F512L processor on the ESK. Programming and debugging can ONLY be done through the USB connection to the secondary PIC32 processor on this board. This "integrated debugger" is very limited, and is easy to confuse. (You cannot use it with a bootloader at all.)

We prefer to use the PIC32 Ethernet Starter Kit board with a real debugger like the Real ICE. To be able to do that, or If you need to be able to interface with more stuff than is present on the PIC32 Ethernet Starter Kit board itself, you can plug the ESK into this little motherboard, called:

Microchip PIC32 I/O Expansion Board



As you can guess from the photograph above (and confirm from the photograph below), you plug the PIC32 Ethernet Starter Kit board into the white connector on the left side of the board, and you can connect various other PICtail Plus boards into the sockets on the right side of the **PIC32 I/O Expansion Board.**

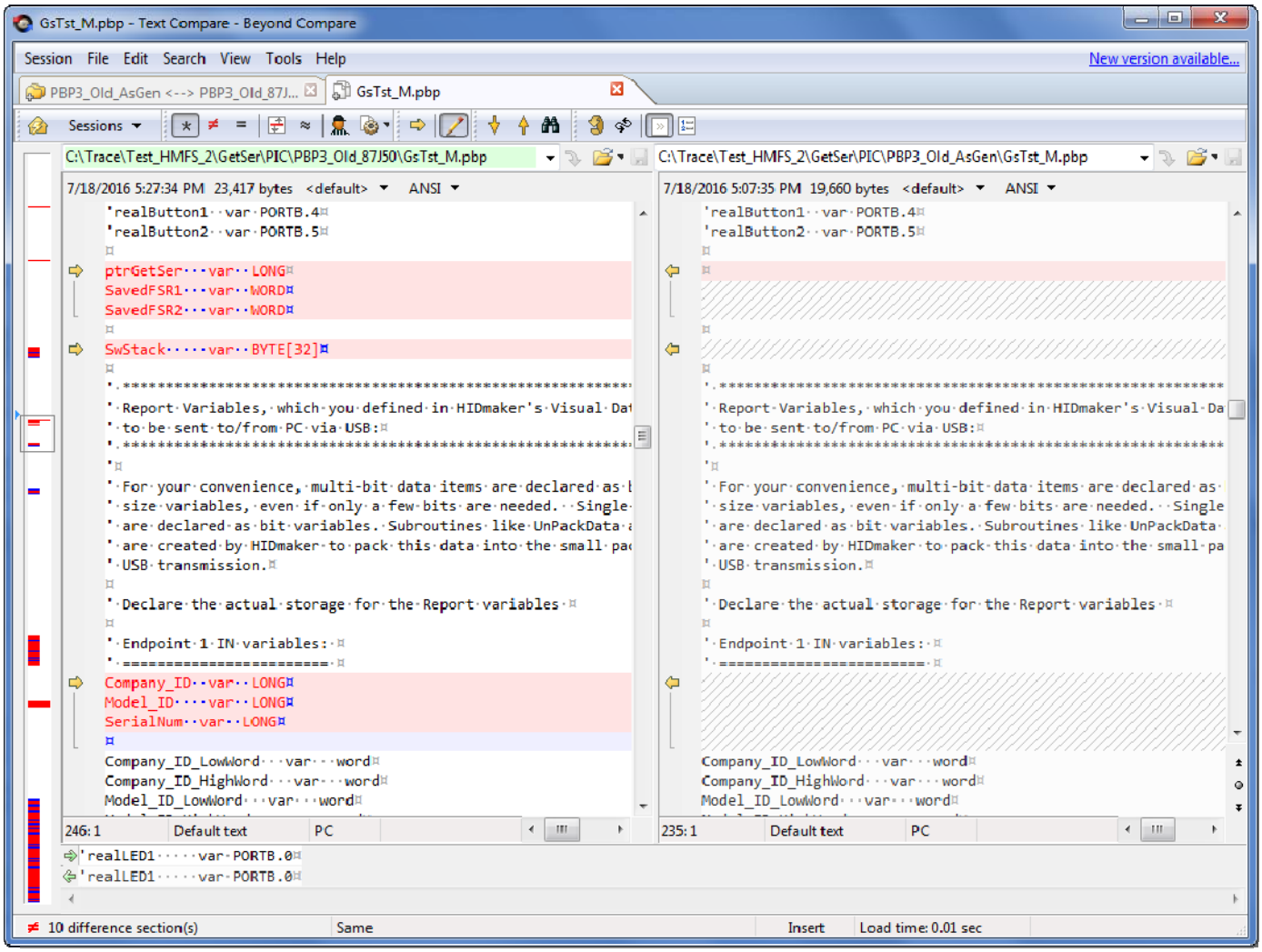


You power the board from the same kind of Microchip wall transformer that powers other boards like the Explorer 16. You connect the modular cable from an ICD3 or Real ICE debugger into the modular connector at the upper left corner of the **PIC32 I/O Expansion Board.**

Software Tools We Love

These are optional tools that WE use a lot, and have recommended them to our customers, who have also found these tool to be really helpful. These tools will really turbocharge your work, and not just your USB work.

Beyond Compare



Beyond Compare is an inexpensive, modern differencing program that has amazing display, search, copy, editing, and reporting capabilities. That is, it doesn't just show

you the differences between 2 files, it will allow you to quickly copy multi-line code snippets from one file to the exact corresponding position in the other file.

Here is the classic scenario where you, as a HIDmaker user, would want to use Beyond Compare.

Suppose you have created a HIDmaker project for a customer or consulting client. You got the generated code running and communicating, and then you added code of your own to the code that HIDmaker generated for you. You added "real I/O code" on the PIC side, to read A/D converters, I/O pins, and other PIC peripherals to connect the USB data to actual hardware on your device. And on the PC side, you completely redesigned the graphical user interface -- away from the generic UI that HIDmaker had generated for you, and toward a UI that does, and displays, exactly what YOU want for your design.

All was going smoothly, and you were 95% done, when your client told you that he wanted you to add "just one more thing." You discover that, in order for you to be able to satisfy your client's wishes, you will have to add a few more USB variables. At first, you would think that this leaves you with 2 really bad choices:

1. Try to add those new variables into your existing, mostly done program, by hand. (Trust me, you don't want to do that. There are lots of things that need to get connected to those new variables...)
2. OR, generate new code with HIDmaker, adding those new variables in the process. This will definitely get you something working, but now you have to add in all that code you have been writing in the other version.

Option (2) is the right choice, but with Beyond Compare, it's no sweat -- you can handle this problem in minutes!

Here's what you do:

- Make a clean, empty new directory to generate new HIDmaker code into.
- From your old project directory tree, copy the HIDmaker project file *.hps into the new directory you just made. (Copy any .rdi files you saved from the Visual Data Designer to that new dir as well.)
- Optionally, also copy the file ProjectSettings.ini to that new directory as well
- Fire up HIDmaker, and open the *.hps project file in that new dir.
- Open the Visual Data Designer, and add those new variables you need -- to whatever USB Interfaces needed.
- Have HIDmaker generate new code in that new directory, which now becomes a directory tree that looks just like your old one did.

- When you exit HIDmaker, use Beyond Compare to compare the two entire directory trees -- you can to that in Windows Explorer. Select your old directory (tree) first, and then select your newly generated directory (tree).
- Beyond Compare will show you those directory trees side by side. You'll be able to click on directories, and see that corresponding subdirectories will line up exactly, and corresponding files within those directories will also line up exactly. Your old directory will show up on the left side, because you selected that one first.
- Double click on the directory that contains the old and newly regenerated versions of your PIC code, side by side
  - The names of files that are identical on both sides will be colored black
  - The names of files that are different on the two sides will be red
  - Any files that only exist on one side, will be blue
- Double click on one of the red files into which you had added customization code. Beyond Compare will show you the contents of the old file (the one WITH the customization code you had added, but WITHOUT the new USB variables) on one side, and the newly generated file (the one WITHOUT the customization code you had added, but WITH the new USB variables) on the other side.
- Beyond Compare will show you the lines of code in those 2 files, side by side, using the same color scheme: black chars on lines that are the same on both sides, red chars on lines that are different, and blue chars on lines that only exist on one side
- If you scroll down through the file, you will see that corresponding lines that exist on both sides, will line up perfectly. But the groups of lines of customization code in your old file (left side) will be blue, because they don't exist in your new file -- YET. The right side will show an empty gap, showing where that group of lines should go.
- You can quickly fix that. In a gutter on the left side, you will see that groups of code lines are bracketed together, with an arrow pointing to the other side. Click that arrow, and Beyond Compare will copy that group of lines to that exact corresponding gap on the right side.
- So all you need to do is to click those arrows and copy your groups of customization code into the exact places they should go in the newly generated code.
- When you finish updating this first file, go on to other files shown in red, and repeat the same process on them, copying lines of customization code from your old files to the newly generated ones.
- At first you may think that there are a lot of files to update, but some of those file differences are caused by the timestamp that HIDmaker places in the comment block of every file it generates.

    - If the only difference is in one of those time stamps, you needn't bother copying those old timestamp lines over the new ones.
- Repeat the process in the files for your PC code as well.

We find that the whole process can be done in about 15 minutes or so. Hardly enough work to break a sweat!

What It Is

- **Beyond Compare** is a modern differencing program with display, search, copy, editing, and reporting capabilities
- Not limited to text files. Can compare binary files like Word doc, Excel spreadsheet. PDF, binary

Why You Need It

- Compares 2 source code or text files and lets you copy code snippets very quickly
- Compares 2 directory trees, and then lets you compare individual files with those directories
- The Pro version can even do a 3-way compare, so you can merge changes from e.g. 2 different developers, into a single master version.

What We Use It For In Our USB Development

- Copying our code customizations to newly generated projects
- Comparing source code in different versions of projects
- Making sure that distribution images are up to date and correct

Other Uses

- Incremental backup to external hard drive
- Determining differences between old developmental versions of some of our files, so we could document the history of the products.
- Over FTP, Beyond Compare Pro version had even let us spot files that hackers had placed on our web site a few years ago.

What It Costs

- Standard version : $30

- Pro version : $60

Where You Can Get It

- http://www.scootersoftware.com/index.php

Inno Setup & Related Tools

If you are going to make a Windows program as part of your product, you will definitely need to be able to install it on your end user's machine. What everybody does is to just use a tool that can generate an installer for you, rather than spending a crazy amount of time trying to develop an install program of your own.

Microsoft used to provide a deployment package plugin with Visual Studio, which would make an install program for you, but Microsoft has discontinued that. The big name companies make installer packages that are VERY complex and VERY expensive ($1000's) -- far more than you would want or need to use for deploying your HIDmaker PC program and other files.

Luckily, Inno Setup is free and even open source. It has valuable 3rd party add-ons, many of which are also free. We have been able to make very nice looking installers for our programs, using only the free tools.

Inno Setup is script based, but it comes with a free GUI front end called Inno Script Studio -- a wizard that will make a script for you. If you need to, you can then modify or enhance the script, using Inno Setup's very good help files that document the entire scripting language. We also use a free 3rd party styling tool called VCL Styles Inno

For Visual Studio users, there are TWO 3rd party products that appear to be related to each other and can be used together.

- "Visual & Installer" (not a typo) is a €99 Visual Studio plugin that lets you edit the script in Visual Studio
- Graphical Installer, with its Graphical Installer Wizard, is a €49 tool that lets you substantially customize the look of the installer that will be created when the script is compiled.

What It Is

- **Inno Setup** is a special script compiler that generates install programs for your Windows programs.
- **Inno Script Studio** is a GUI wizard that will generate an Inno Setup script that Inno Setup can compile. Both are free, and you can install both from a single installer.

Why You Need It

- Trying to make an install program yourself, for your Windows software, is very complex, difficult, time consuming, and error prone. It's not something you would want to spend your time on.
- Microsoft has removed their Deployment Package generator from newer versions of Visual Studio, so you need something else now
- Commercial install program generators are VERY expensive, and MUCH more complex

- Especially with Microsoft languages, it can be difficult to even figure out WHICH files you need to install on your end user's PC

What WE Use It For

- Making the Setup program that installs our HIDmaker products, and its ActiveX files, on our end user's PC

What It Costs

- The main programs (Script generator and wizard) are FREE
- Styler add-ons are FREE (We use VCL Styles Inno)
- Add-ons for Microsoft Visual Studio are not free: 99 Euro

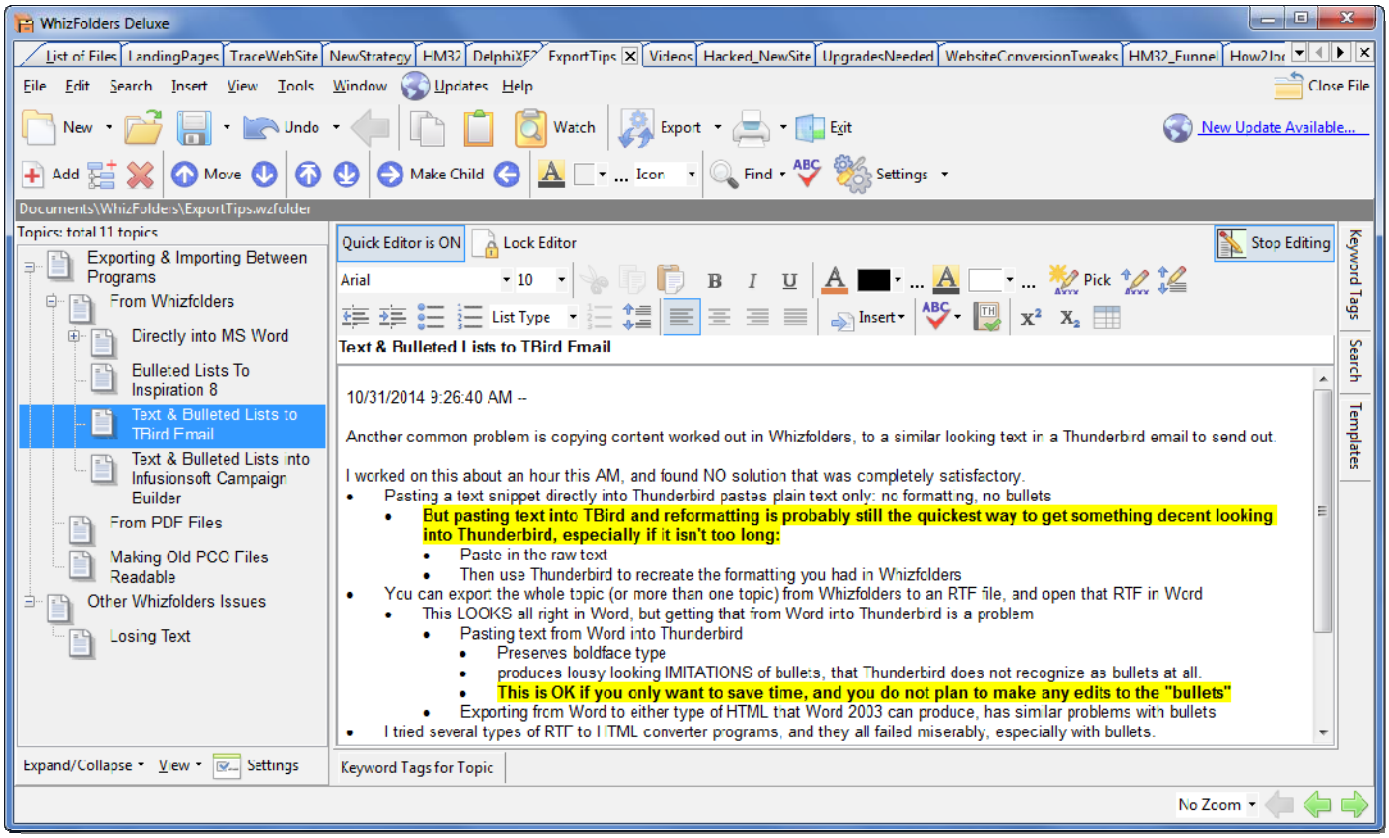
Where You Can Get the Free Inno Setup and Inno Script Studio

- http://jrsoftware.org/isinfo.php

Where You Can Get The Visual Studio Related Products

- "Visual & Installer" (not a typo) is available from http://www.visual-installer.com/
- Graphical Installer is available from http://www.graphical-installer.com/

Whizfolders



This next resource may seem a bit unnecessary to you, and we admit that it is not absolutely essential to your specific USB work. We're telling you about it because it is an inexpensive tool that will really surprise you.

You'll find that it will help you in everything you do, because it helps you THINK, it helps you SOLVE PROBLEMS (even really hard ones), and, over time, it helps you build up a Knowledge Base of all of your own Intellectual Property.

Pretty cool for 30 bucks...

Whizfolders is billed as a "two-pane outliner and organizer tool." As the screen shot shows, the left pane is a tree -- an outline of "topics" containing notes you write. Each "topic" is like a separate embedded Word document, that lets you type notes, using different fonts, boldface, colors, highlighting, bulleted or numbered outlines, tables, indenting, etc -- anything you need to organize your thoughts as you work, yet make

them clear when you come back to a project months or even years later. The right pane is a "topic", into which you type notes, add pictures, quickly add timestamps, and even add links to other topics.

So what? Why should you care ???

Because, if you are reading this document, you are someone who has to ***solve problems:*** really hard ones at times. A key fact about the way the brain works, is that when you start to figure out a difficult problem, ideas do not come to you in a linear, start-at-the-beginning kind of way. Instead, they tend to come in a jumble -- rapidly at first, and in random order. At the beginning, you don't even know which ideas are important yet. If you don't get each idea written down or typed up in a short time, a new idea will come along, and you'll forget the previous one.

We find that Whizfolders helps you organize your thoughts very quickly, before you lose them. Then, when the flow of ideas slows down and stops, you are looking at the screen where you have typed up and quickly organized the ideas that already came to you -- which will stimulate new ideas and connections. Which you can again record, and organize, and elaborate upon. And build up a solution.

And if you use it the way we suggest, you will find that the problem, and your solution to it, **gets automatically documented, by you, as you go along.** This does NOT slow you down. Rather, **it helps you make better use of your time while you *already ARE* slowed down** -- while you are trying to figure out what the problem IS, and how to SOLVE it.

This is a real sleeper -- I got it about 5 years ago as a helper for A FEW tasks, but I wound up using it ALL THE TIME. This one is ***waaay*** more important than you may think right now.

Years ago, when we developed the first version of HIDmaker, we documented the work in hand written notes that we mounted in 3-ring binders, each binder titled with a sticky label. In the ensuing years, which included some moves of our household and business, some important binders have gotten lost, partly because some of those labels have fallen off. Those binders are heavy, take up a lot of space, and cumbersome. It is also hard to determine which binder contains the import piece of "how did I DO that?" information we might need to find, years later.

Nowadays, during the development of HIDmaker 32, I used Whizfolders at all times, to document the entire development effort.

I can hit function key F5 to cause Whizfolders to instantly add the date and time wherever I am in the document -- **so everything gets time stamped.** When I have to track something down months or years later, those timestamps help me correlate file times on my hard drive with what I was working on at that time, so I can figure out the

context of why I was working on that. Was it for a client? Which one? What was I really working toward?

And all my notes are ***searchable*** now.

If I modify some important file, I can easily get Windows Explorer to copy the file path to the clipboard, so I can quickly paste it in to my Whizfolders notes -- and now, in a few seconds, I have recorded, for the future, a record of where that important file was located (and hopefully by that future time, still IS there). Let's face it -- file paths get pretty long these days, so when I was writing notes by hand, I didn't often write out those long file paths. It can be maddening to have look at those old handwritten notes nowadays, and have to wonder "where WAS that file, anyway?"

The point of Whizfolders is that everything you need to do is easy. And if it is easy, you will ***use*** it, and document things with it, ***while you work.***

What It Is

- Technically, **Whizfolders is a "two-pane outliner tool."**
- Left pane is a tree -- an outline of "topics" containing notes you write
  - Each "topic" is like a separate embedded Word document, that lets you type notes, using different fonts, boldface, colors, highlighting, bulleted or numbered outlines, tables, indenting, etc -- anything you need to organize your thoughts as you work, yet make them clear when you come back to a project months or even years later
- Right pane is a "topic", into which you type notes, add pictures, quickly add timestamps, and even add links to other topics

Why You Need It

- Helps you organize your thoughts, document and date stamp your work, and solve all sorts of difficult problems, while you work
- When you are starting to work on a problem
  - Your thoughts do not come to you in an orderly fashion.
  - And if you don't record a thought, it can be quickly lost as new thoughts pop into your head, so you forget the previous thought
  - Especially at the beginning of solving a problem, new thoughts will come to you quickly, but you don't yet know which of those thoughts are the important ones.

- Various KINDS of thoughts and ideas will occur to you, often in a completely random order
    - Some of these thoughts will be about the details of the problem you are facing
    - Other thoughts will be about things you could do to learn more about the symptoms of the problem
    - Still other thoughts will be about
- At a stage like this, if you can get those thoughts typed up as they occur to you, you can also ***organize*** those thoughts ***as they occur to you, at the same time***: simply place each thought at the appropriate place in bulleted outline
- When the flow of new thoughts eventually slows down, you now have a list of ideas or factors right in front of you, already organized in a meaningful way
    - As you look at this list, and think about your goal, new thoughts about what to do will occur to you -- so add them to your list
    - After awhile, you will at least have some ideas about what to try or information you still need to find
        - In other words, this process leads you to actionable ideas that will lead you to a solution
        - As you try out some of the things you wrote down, take notes in the same Whizfolders topic, to document:
            - What did or didn't happen
            - What you learned
            - WHERE (e.g. on the web) you got more information
            - And what you think might be worth trying next.
- The end result is that, as a natural part of the work you did to figure out and solve the problem, you have also documented completely, AT THE SAME TIME, the problem, its symptoms (so you can recognize it if it ever pops up again), and your solution
    - You don't need to go back and do a separate step of documenting what you did -- it is already there in your Whizfolders files.
    - If you need to formally document your solution for somebody else, you can copy parts of the text you wrote in Whizfolders, and paste it into your favorite word processor, as you build up your formal documentation
- IMPORTANT! Doing this does ***NOT*** SLOW YOU DOWN !
    - Rather, **it helps you make better use of your time while you are *already* slowed down** -- when you are trying to figure out what the problem IS, and how to SOLVE it
- Over time, you build up your own encyclopedia of everything you have done
- As you do work on important files on your computer, it is easy to document WHERE those files are located on your hard drive: just copy the directory and file name in My Computer (Windows Explorer) to the clipboard, and paste into your Whizfolders notes

    - Let's face it. If you were writing notes by hand, you probably wouldn't take the time to write out those long file paths, but if you can take 2 seconds to copy and paste them into your Whizfolders notes, you WILL do that!
- Your notes are searchable, so if you dimly remember that you solved some particular problem involving "widgets," but cannot remember what the solution was or where you documented the solution, just start searching for "widgets"
- This is the best tool we have found for figuring out how to solve problems, while at the same time documenting your solution

Other Uses

- We used this tool to create the content for our entire web site

What It Costs

- $35

Where You Can Get It

- http://whizfolders.com/

Delphi / RAD Studio

Look, we get it. Talking about different compilers is like talking about different religions: you're probably not gonna change anybody's mind, and you are very likely to offend someone. We promised to tell you, in this document, about the resources and tools that WE use, so we'll give it to you straight, and try not to offend.

You are perfectly free to disagree and ignore us.

I have been programming all of my professional life -- even in the old days when I was working in the field of Surface Acoustic Wave filters at Zenith. In those days, the

computer program was a means to design the end result: a SAW filter that had the desired frequency response.

Once I started my small company, Trace Systems Inc., TIME became really important -- not just solving a problem for a consulting client, or making a new product, **but doing it quickly enough** that it won't break the bank.

Delphi gives you true Rapid Application Development. Embarcadero, the makers of Delphi, say that it lets you develop "up to 5X faster." Based on my experience, that might even be an understatement.

What It Is

- I believe that Delphi is the most productive Windows programming tool on the planet
- Huge range of nice looking screen controls
- Code that is simple to read, simple to use, but does everything you need
- Highly reliable
- Recent versions are cross platform -- lets you compile same code base to Windows desktop, Macintosh desktop, iOS mobile devices, Android mobile devices This WORKS!
- Your Windows programs compile to simple .exe files, rather than gigantic, complex .NET assemblies

Why You Need It

- True Rapid Application Development
- Embarcadero says you can develop "up to 5X faster" -- I find it much faster as well
- Better Provisioned, Down Deep
- In my experience, when you run into one of those "How do I get Windows to do THAT?" problems, Delphi helps you far more. Google the problem and you will see the difference
- Usually, the Delphi solution is some small 1 or 2 line snippet that calls something already built in to Delphi
- Usually, the corresponding Microsoft solution is that you get to write 20-100 lines of extra code

- Every person who has ever taken our advice and gotten Delphi, has thanked us for telling them about it.

What We Use It For

- We CAN program in those other languages, but we CHOOSE to only develop our own software with Delphi

Other Uses

- Everything we do

What It Costs

- First time purchase is expensive: > $1000
- Upgrade price to new version is much more reasonable -- around $600, so keeping current is less painful, and is WELL worthwhile

Where You Can Get It

- www.embarcadero.com

Important Tips for USB PIC Developers

Here is a really illuminating quote from one of our customers:

> **"I just wanted to let you know that I just started using my copy of HIDMakerFS. I have read your entire manual and experimented with the sample code. I cannot believe how easy and intuitive this product is. You have really boiled down the development process; I could not be more pleased or impressed. I find the Visual Data Designer tool to be EXCELLENT.**
>
> **The most difficult challenge for me has been to retrain my thinking of RS232 communication methods to the easy process you have developed of passing direct variables from the desktop to the PIC. Everything is so much easier now; I don't feel I really need to fully understand USB communications to develop hardware/software applications. Kudos to you and your work for the development of such a powerful product, it's worth every penny and I could not be more pleased!" - Ryan Sheldon**

We highlighted the key passage to emphasize his important point: **he adjusted his thinking a little bit, and everything is much easier now.**

Sometimes we can make great gains in productivity by simply adjusting our thinking a little bit. It doesn't have to take a lot of time to change our thinking, but having done that can save a lot of time and effort going forward.

The following tips are meant to do just that: to adjust your thinking in ways that will help you:

- avoid common mistakes,
- so you can avoid wasted time and dead ends,
- make a project that works smoothly and efficiently,
- and get it done quickly.

Reminders of Basic but Important Facts:

- Remember that a modern PC is orders of magnitude faster than a microcontroller, with orders of magnitude more storage than a microcontroller.
- A USB project is a 2 processor system, so partition segments of the total amount of work to the right processor -- the one best suited for each part
- The microcontroller is good at doing things like this:
  - Low level bit banging
  - Responding rapidly to immediate needs of the hardware being controlled
  - Concentrating on one thing
  - Doing primitive operations
  - Controlling or monitoring hardware
  - Directly reading sensors
- The microcontroller is NOT good at doing things like this:
  - Mathematical calculations
  - Parsing string commands
  - Formatting data in human readable forms
- The PC is good at doing things like this:
  - Commanding the PIC side to do a series of primitive operations
  - Mathematical calculations
  - Interacting with the user
  - Storing large amounts of data
  - Displaying formatted data in human readable forms
  - Multitasking / multithreading
- The PC is NOT good at doing things like this

- Low level bit banging
- Responding rapidly to immediate needs of the hardware being controlled
- Concentrating on one thing
- Doing primitive operations
- Controlling or monitoring hardware
- Directly reading sensors

Helpful Tips To Adjust Your Thinking:

- Try to break up the work that the PIC has to do, into small, primitive operations that can be commanded separately
  - If you do this right, your PC program can command the device to do different combinations of those primitive operations, which will make your 2-processor system much more powerful and flexible -- and future proof
- The PC is where all the speed, power, graphics, and memory are, so make the best of those capabilities --
  - Your PC program can be updated by just sending the customer a new program, via email or downloading from your web site -- no special hardware needed
  - Do as many calculations on the PC side as you can, as opposed to doing calculations on the PIC side -- they will be done much quicker, and more powerful math functions and math hardware are available
  - Design the PC program using modern Object Oriented principles, with an object that acts as a facade of your device
    - Design the PC code so that the main program controls the facade object, and the facade object communicates with the device as part of executing the commands from the main program
- Think binary -- don't waste time trying to parse string commands in a slow PIC device.
- Don't bother sending formatted information over the USB -- this is another waste of time
- Don't bother calculating stuff on the microcontroller
- Think in terms of choreography -- if one side is going to be busy doing something for awhile, what can the other side be doing during that time?
- Repeat after me: "A PIC is not a PC, A PIC is NOT a PC, ..."