The Embedded Muse 130

Editor: Jack Ganssle (jack@ganssle.com) May 15th, 2006

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

I won't present my public seminar "Better Firmware Faster" till November or so. Locations and dates will be set in August. Meanwhile I can do the class at your facility, for your engineers. See http://www.ganssle.com/classes.htm .

The Embedded Muse will be on hiatus during the (northern hemisphere) summer.

Computer History and Tidbits

For a number of interesting links to the story of the Eniac, see http://ftp.arl.army.mil/~mike/comphist/ . Do check out http://ftp.arl.army.mil/~mike/comphist/46eniac-report/index.html , which is a 1946 technical report on the computer. It includes more about how the machine works than most of us want to know.

Harold Hallikainen is scanning old broadcast equipment manuals and posting them at http://www.hallikainen.org/BroadcastHistory. That page has links to the manuals he has scanned, plus a lot of other sites that document the early days of electronics. The computer history links are interesting; one takes you to a product (Catweasel) that reads old floppy disks (all PC formats, Amica, Atari, early Mac, Commodore and more). If you've got data stashed on one of these apparently write-only media, the product could be a lifesaver.

Jim Ziegler and others pointed out I misspelled a computer in the last issue; the correct spelling is LGP-30. He recommended a book by Montgomery Pfister: "Logical Design of Digital Computers." This 1958 tome had a lot of info about designing computers using serial memory systems such as drums, and mercury delay lines for registers, program counter, etc. Based on his recommendation I bought a used copy, and am astonished at how much of this material is still totally relevant for digital designers today. Though technology changes, Boolean Algebra hasn't.

Failure Stories

A lot of folks responded to the failure story in the last issue. Here are some readers' thoughts:

Harold Hallikainen wrote with one approach to optical measurements: Your smoke detector reminded me of a recent project. It's a reflective color sensor. We have "dark compensation" where we subtract out the sensor values when the illumination LED is off. We do not get enough stray light into the unit to need to worry about sensor, amplifier, or ADC saturation (we're using a sensor with built in current to voltage converter). In doing this project, I found that the temperature coefficient of LEDs is pretty wide, so the amount of illumination would vary with temperature (and the self heating of the LED). We solved this by having two sensors, one sensing the illumination, and one sensing the reflection. We established a dark compensated reflection ratio for RGB, then normalized this to the values for white (giving us a white balance and compensating for other variations in the system). The resulting normalized values are compared to values in a table to identify the color. Worked out nicely!

Steve Litt of Troubleshooters.com added another lesson learned from the failure: Design documents should be passed down the production process, and code should be commented.

Amen! But design docs are all-too-often limited to start with, and then discarded or lost. Check them into the version control system.

Scott Nowell seconded these thoughts: Generate clear documentation. We are used to at least hearing this for software, but the same holds true for hardware. Make sure you describe it so that the next guy can understand it. And read the documentation, if you're lucky enough to inherit some from an older project. Read it!

Tormod Tjaberg wrote with several thoughts. I liked his parting comment: Be vigilant!

Bill Auerbach of Softools wrote about the documentation issue: There is IMO a very important 6th cause: Not having an accurate and up-to-date spec on the firmware/product. Left on there own, engineerings implementing a product will do as they think it should be done, not as the customer has told marketing/sales what they want done. This leads to last minute changes which take away often from testing time.

Bill recommended the book "Joel on Software" by Joel Spolsky. I agree; it's a collection of the always thought-provoking essays Joel is known for.

Finally, Mark Dobrosielski contribute his own tale of woe:

I used to work for PRI Automation (now Brooks PRI Automation or something), a maker of semiconductor fab automation robotic systems. One of my projects was a system that automated the loading and unloading of diffusion furnaces. It consisted of two robots: The first was a 5-axis arm robot that transferred the silicon wafers between plastic cassettes and silicon carbide or quartz boats. The other was a 4-axis elevator robot that moved the cassettes and boats between the furnace, storage positions, and operator ports.

The robot was fairly large (about 15 feet high, 4 feet wide, and 20 feet long, approximately). Parts of it were enclosed by "skins," plastic or fiberglass panels that kept operators from grabbing material out of the storage locations without telling the robot. There were IR optical sensors at every location, so it would immediately see that material disappeared, and since it couldn't know where it went, it had to shut down and call for help.

The problem? During burn-in testing, the robot would often fail overnight. It would start to think that material was spontaneously appearing and disappearing from its storage locations. Our hardware guy was tearing his hair out trying to make the sensors more reliable. He'd make a change, test it for a few hours, and declare it fixed. He'd put the skins back on, and the robot would burn in just fine. Then we'd take the skins off to work on something else, and we'd fail the next overnight burn in.

Do you see it coming yet? We didn't but should have. The robots failed at about the same time early every morning. The logs would have told us this, but the software guys didn't notice it.

As management panic set in, we worked some long hours. I was there overnight one time with a few other guys . I had dozed off under a lab bench, and woke up just as the sun started to come up. The sun started streaming through the windows in the assembly bay. The windows faced east. The robot storage area faced east. The skins were off. As the sun started creeping up the robot, the material sensor alarms started going off. The sunlight was triggering the sensors, and as soon as people saw it, they realized it. That sunrise was greeted with a mixture of laughter and fairly creative cursing.

As embarrassing as it was, this bug wasn't a bug at all (in actual use, the system wouldn't be located next to a window and would never operate without its skins). It did require a change in procedures and the purchase of the some window shades, though. The hardware guy was vindicated - after thinking he was losing his mind - and the software guys learned to pay closer attention to what their logs told them (for example, when the robots failed, they always did so shortly after sunrise) . That they did so only on clear days with the skins off is not noted in the logs, so we had to forgive them not noticing that.

Tools

A lot of folks have been kindly submitting information about their favorite (and ones they hate) tools. Please continue! I'll post your thoughts to http://www.ganssle.com/tools.htm .

Cliff Brake has a hex calculator written in Ruby: http://bec-systems.com/web/content/view/42/9/ .

Leland Hamilton contributed a cornucopia of thoughts: I have used Dev-C++ (http://www.bloodshed.net), an IDE for C and C++ on Windows. It allows one to build a project with associated files, automates the compile & link functions along with program execution with parameters, and profile analysis. Apparently uses the Mingw compiler suite. Has a search function that can work on the current module, all open files, or all files associated with the project. Includes interactive debugger that uses gdb under the hood. The editor does syntax highlighting (bold keywords, italic blue comments, green preprocessor # statements, purple numbers, and red quoted text. Parses the project file and provides a class view on the side that allows one to jump to the code associated with a function or structure. (Have not used it for C++). Includes CVS integration (but it does not work for :local:", so I use WinCVS instead). Project setup tool includes Windows application, Console application, Static library, DLL, and empty project that select the build and libraries required for the project. I have only used the console application mode as I generally use Dev-C++ to develop embedded applications and frequently do unit and multi-unit testing on the PC platform before moving to the embedded platform, and for quick homebrew utilities.

Dev-C++ allows other tools to be added to a tool dropdown. I use the following:
- a pretty print (code highlighting, etc) n-up utility (prfile http://www.lerup.com/printfile/) configured to print a source listing of the current file,
- a homebrew utility that prints lines over 80 characters in length with line numbers (One customer does not want any more that 80 characters per line. Even though Dev-C++ displays a right margin line at column 80, it is possible to continue typing beyond that.),

- a cygwin bash script that to run the program, use less to scroll through the output and diff to see changes in the output, and - gcov for code coverage analysis using a cygwin bash script to capture the log output that summarizes coverage by function

The bloodshed homepage includes links to other resources including listings of free compilers, learning information, a list of downloadable C/C++ source file and useful links.

My two gripes with Dev-C++ are:
- only one file is visible at a time, although it is easy to switch between multiple open files
- this and several other editors use a separate search dropdown menu instead of including the find and replace commands under the edit dropdown menu like MSWord and another tool that I was using. I frequently used the wrong keyboard sequence one or more times after switching between MSWord, Dev-C++ and another tool.

Although there were a few side references to Version Control Systems (VCS) in the discussion of other tools, VCS probably deserves a section of its own.

I prefer to place software in some sort of VCS system as it is being developed, mostly to keep a backup in case I clobber a file, but it also helps to see the development progression. I once consulted for a small company for a few days to solve a problem they were having with some embedded firmware. Once we got the software working I suggested they update their VCS repository before making any more changes so we could recover the working copy if necessary, but they insisted on making a few small changes first. They ended up with a non-working version and could not reconstruct the working version.

To start off, I suggest CVS with WinCVS GUI front end. WinCVS provides a graphical tree structure on the left similar to Windows Explorer, and a display on the right of files that are and are not in CVS, whether the CVS tracked files are modified Includes capability for almost every CVS command you can think of, along with the ability to type any command in a command line window.

CVS files are stored in a central repository that can be on your computer or another computer running a CVS server. Also uses a local subdirectory file structure that identifies the repository and current workspace file information. Tagging specific revision of files, comments for each update, branching (for those cases where you need to continue new product development and perform maintenance updates on a prior release. Commands include creating a repository, importing and exporting a directory tree, tagging and branching, checkout, update working file with repository changes, committing a file to the repository, adding text, binary and Unicode files, difference of working file versus repository and other combinations, annotated listing showing the version number associated with each line, module status listing, graphical branch structure diagram with ability to select specific tree locations, option for file locking editing, watch notification.

Subversion and TortoiseSVN (http://tortoisesvn.tigris.org/svn/tortoisesvn): a modern day VCS system that includes integration in Windows Explorer. Prefers to work with directory trees, but using its repo[sitory]-browser, it is possible to create a subfolder and put a single file into Subversion. I found it a little difficult to get right the first time, but once set up it is easy to work with.

TortoiseSVN is a free open-source client for the Subversion version control system. That is, TortoiseSVN manages files and directories over time. Files are stored in a central repository. The repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to recover older versions of your files and examine the history of how and when your data changed.

TortoiseSVN adds Shell integration, Icon overlays for file status, explorer context submenu.

Some of the many Subversion features that may be of particular interest:

Atomic commits: A commit either goes into the repository completely, or not at all. This allows developers to construct and commit changes as logical chunks.

Directory versioning: CVS only tracks the history of individual files, but Subversion implements a virtual versioned file system that tracks changes to whole directory trees over time. Files and directories are versioned. As a result, there are real client-side move and copy commands that operate on files and directories.

Usable locally and over networks, versioned metadata (properties associated with a file), binary differencing algorithm for both text and binary files.

Also RCS probably deserves mention, although I think CVS and Subversion trump RCS.

Pretty print (code highlighting) n-up file printing utility (prfile http://www.lerup.com/printfile/): very useful for printing 1 and 2 up source listings with line numbers, code highlighting (bold keywords, italic comments), and long line wrapping. Can select various output formats, such as 131 columns wide landscape text printing, and save the setups with names for recall. You can chose either column width or lines per page, optional pretty printing code highlighting, optional line numbering, specify tab width, heading and footer formats, number of pages per sheet (n-up) horizontal and/or vertical with either landscape or portrait orientation. Also there is support for Postscript file printing using the ghostscript and gsview programs, and a print spooler capability. Also offers a command line mode. I set up prfile as a Windows Explorer right click SendTo with the default to do 2 up line numbered source listings, and a tool in my IDE, Dev-C++, to print the current file.

Mark Bereit says: I use Microsoft Visual Studio 6 as my all-purpose IDE, for both desktop and embedded development. The abilities of this tool are greatly enhanced using Visual Assist from Whole Tomato Software (http://www.wholetomato.com). This does better source code coloring, great autocompletion and suggestion, and great source code browsing (that doesn't depend upon Microsoft's browse database, so it works just fine on your cross-platform C/C++). Works for both Visual Studio 6 and Visual Studio .Net. I've used several versions over the years and find it to be a great help.

Alejandro Weinstein suggests Bray's Terminal (http://bray.velenje.cx/avr/terminal/). From the webpage:

"Terminal is a simple serial port (COM) terminal emulation program. It can be used for communication with different devices such as modems, routers, embedded uC systems, GSM phones. It is very useful debugging tool for serial communication applications"

It is not an "Hyperterminal" kind of terminal. It is really oriented toward embedded developing. And it is free.

Tim Dahlin wrote about a number of tools: here are some programs that I use on a regular basis.

Tera Term Pro --> Terminal Program Old Version => http://hp.vector.co.jp/authors/VA002416/teraterm.html . New Version => http://www.ayera.com/teraterm/ .

xvi32 --> Free Hex Editor for Windows, http://www.chmaas.handshake.de/ .

Tiny Hexer --> Free Hex Editor for Windows, www.mirkes.de.

Advin Programming Tool --> www.advin.com, I use it to convert different program file formats. It does not require an Advin programmer. The DOS version can be scripted using batch files.

PCalc --> Programmer's Calculator from www.analogx.com .

VolumeID --> http://www.sysinternals.com/Utilities/VolumeId.html . Let's you modify your hard drive id.

Jobs!

Let me know if you're hiring firmware or embedded designers. No recruiters please, and I reserve the right to edit ads to fit the format and intents of this newsletter.

BJ Services Company in Tomball, Texas is looking for a Embedded Software Engineer. Check them out at www.bjservices.com . This position has the responsibility for firmware through the full product development life cycle from detailed analysis and design through verification and release.

Desired Skills/Experience:
* EE/CE/CS bachelor's degree
* 3-6 years experience
* Embedded C software development
* Real Time embedded operating systems and low level programming
* Software development on Windows and Linux platforms
* Ability to port Linux to new platforms and produce required drivers
* Experience with ARM & Coldfire processor development

Send resume to rrupe@bjservices.com or fax to (281) 357-2716

Ag Leader Technology Inc., Ames, Iowa, builds equipment for precision farming. They're looking for two people:

Electronics Design Engineer: Design of high-reliability electronics for precision farming products. 3+ years experience in design of complex, high speed digital circuits required. Experience designing leading edge PDA type products a plus.

Embedded Programmer: Entry level position to coordinate language translations of existing applications and continue development of existing demonstration and simulation programs. C++ experience required.

Send resume to: careers@agleader.com

Schweitzer Engineering Laboratories (Pullman, WA) is always looking for engineers. Firmware, software, electronics, etc.

Unfortunately the web site contains non-descriptive job titles. But Software Engineer usually has something about embedded.

See http://www.selinc.com/careers/index.html . They are planning to add an astonishing 200 to 500 people this year.

Dynojet Research, a leading designer, manufacturer & marketer of automotive, motorcycle, and power sports performance enhancement products has an immediate need for Electrical Engineers. These positions are responsible for all aspects of R&D, from conception to production, in engine management products. Skills required include analog/digital circuit design, schematic and circuit board design using PCAD, and embedded processor experience in Assembly and C. Freescale and ARM based processor experience is preferred.

Demonstrated knowledge in fuel injection and internal combustion engine operation is a plus. Entry level to experienced engineers encouraged to apply. Send your resume to Dianne@Dynojet.com or fax it to 406-388-6523.

Joke for the Week

The folks at Keil (vendors of compilers for Arm, 8051, etc) are a fun bunch. As an April Fool's joke one year they ran an ad for a Cobol compiler for the 8051. Check it out at www.keil.com/cobolad.pdf .

About The Embedded Muse

The Embedded Muse is an occasional newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to him at jack@ganssle.com.

To subscribe, send a message to majordomo@ganssle.com, with the words "subscribe embedded *your-email-address*" in the body. To unsubscribe, change the message to "unsubscribe embedded *your-email-address*".

The Embedded Muse is supported by The Ganssle Group, whose mission is to help embedded folks get better products to market faster. We offer seminars at your site offering hard-hitting ideas - and action - you can take now to ***improve firmware quality and decrease development time***. Contact us at info@ganssle.com for more information.