The Embedded Muse 16

Editor: Jack Ganssle (jack@ganssle.com) February 17, 1998

Editor's Notes

Today the 4000th subscriber to this newsletter signed up. I appreciate everyone's continued interest and support! Please feel free to send embedded stories and hints that you'd like to share with your fellow readers.

Recent news indicates Sun is ready to introduce (reintroduce?) Embedded Java in late March. Though I'm convinced that current embedded languages (C, C++) serve us poorly, I have to admit to being skeptical about Java's future in this market. It is a fascinating departure in programming paradigms, in some ways drawing on the best from C, Forth, and from object-oriented practices. The embedded community is, however, conservative to the extreme and tends to adopt new languages only very slowly. Until Java solves its various technical woes, and until a cadre of trained embedded Java folks appear, I suspect it will be a technology without an application.

We must, however, keep a close eye on Java's progress. Who knows what as-yet-unknown force may drive us into adoption of embedded Java tomorrow?

Finally, a number of people have been asking for the Muse to publish a personal embedded history I worked on under a different venue some years ago. Here's part one of a rambling story, to be continued erratically as time permits. Feel free to send your own stories along for reposting if appropriate.

A Personal Embedded History - Part 1

In the blink of an eye our children grow into adults, middle age suddenly assaults us while we're still wrestling with becoming (more or less) responsible citizens, and the technology of yesterday ages, becomes obsolete, and then forgotten. Once in a while it's important to step back, take a breath, and remember where we've been.

The embedded industry lacks a coherent record of its history. Most of us are too busy designing systems to undertake the role of industry historian, but, still - we're often interested in where we've been.

In 1971 Intel shocked the electronics world with the announcement that the 4004, a 4 bit microprocessor, was now available as a component. A confirmed 18 yr old computer geek at the time, I remember the press hoopla well. Frankly, most engineers just did not believe the announcement. Computers cost tens of thousands of dollars. A computer on a chip? Clearly unlikely, if not impossible. (Now, in February 1998, a generation and more after Intel's announcement, postings in comp.arch.embedded reveal the formation of a "4004 Enthusiasts Group". Yesterday's impossible technology reverts to the stuff of nostalgia.)

Time proved that such an advance was indeed possible. A year or so later Intel's 8008 hit the market. This part generated much more enthusiasm than its predecessor. An 8 bit computer just might be able to do something useful. After all, DEC's PDP-8, a well-accepted "serious" computer, had a word 12 bits long.

The 8008 was an 18 pin chip that needed three different voltages to run. As I recall, +5, -9, and +12 were all needed, plus a two phase clock input. This didn't leave many pins for the address and data bus! Intel decided to multiplex data and address on the same pins, an approach later used on their 8085 and 8088 as well. It's only recently that high pin count surface mount devices from Intel have come with separate busses. The chip used PMOS technology, which has long since been supplanted by NMOS and finally CMOS.

In 1972 no one dreamed that large microprocessor programs would be important. The 8008 had only 14 address lines, limiting its address space to a measly 16 Kb. It would be tough to write a device driver in 16 Kb now, but back then we were thrilled to have so much memory. In fact, memory was so expensive that none of the 8008 systems I worked on ever used more than 12 kb. Typical static RAMs were 256 bits to 1k bits per part; dynamic devices weighed in with all of 4k. The 1702 EPROM (256 bytes) ran at an astonishing 1.3 microseconds.

The 8008 had a hardware stack, a 7 word-deep stack built into the silicon. You could push or call up to seven levels deep but that was it. Finding code that pushed just once too many times was a nightmare we fought constantly.

Through the luck of being in the right place at the right time I managed to land an engineering job while barely in college. No one had a clue how to program these beasts; I knew as little as anyone but was cheap and available. The company I worked for started a major project around the 8008.

Our development environment was an Intellec 8, a computer Intel built around the 8008. It had a modular bus with 18 slots, so given enough money you could populate the computer with a whopping 16 k of RAM. We built an extender to connect the Intellec's bus to the backplane in the system we were designing, building what in effect was a crummy bus-level emulator. Our product measured protein content in grains - wheat, soybeans - by looking at how the sample absorbed IR light.

Booting the Intellec 8 was one of those rare “pleasures” computer folks no longer have to hassle with. Its front panel was covered with switches. You’d key in a boot loader in binary, instruction by instruction, and then hit the EXECUTE button. If your fingers set all the switches perfectly the teletype would read in a paper tape loader program. A single bit error required reentering all of the hundreds of switch settings.

A later upgrade put the boot loader in ROM. Then all one had to do was enter the binary for a JMP 0000 to start the code. I still remember those codes: 01000100 00000000 00000000.

Our only I/O device, other than the lights and switches on the Intellec’s front panel, was a not-so-trusty ASR-33 teletype. There were no CRT monitors in those days. Operating at a blinding 10 characters per second, this mechanical beast was no doubt the cause of many ulcers and crimes of frustration. The teletype needed 8 seconds to print one line of output.

The ASR-33 included a paper tape punch and reader. Tape was our storage media - neither tape nor disks came along till much later. All of our programs, both binary and source, were stored on paper tape.

Needless to say, high level languages were just not feasible. We did have one brief flirtation with PL/M, cross compiling on a mainframe and downloading the resulting tape to the Intellec. The compiler was notoriously unreliable so we eventually switched to assembly language.

More later…

Thought for the Week

Jake is struggling through a bus station with two huge and obviously heavy suitcases when a stranger walks up to him and asks "Have you got the time?"

Jake sighs, puts down the suitcases and glances at his wrist. "It's a quarter to six," he says.

"Hey, that's a pretty fancy watch!" exclaims the stranger.

Jake brightens a little. "Yeah, it's not bad. Check this out" - and he shows him a time zone display not just for every time zone in the world, but for the 86 largest metropolis.

He hits a few buttons and from somewhere on the watch a voice says "The time is eleven 'til six" in a very West Texas accent. A few more buttons and the same voice says something in Japanese. Jake continues "I've put in regional accents for each city". The display is unbelievably high quality and the voice is simply astounding. The stranger is struck dumb with admiration.

"That's not all", says Jake. He pushes a few more buttons and a tiny but very high-resolution map of New York City appears on the display. "The flashing dot shows our location by satellite positioning," explains Jake. "View recede ten", Jake says, and the display changes to show eastern New York state.

"I want to buy this watch!" says the stranger.

"Oh, no, it's not ready for sale yet; I'm still working out the bugs", says the inventor. "But look at this", and he proceeds to demonstrate that the watch is also a very creditable little FM radio receiver with a digital tuner, a sonar device that can measure distances up to 125 meters, a pager with thermal paper printout and, most impressive of all, the capacity for voice recordings of up to 300 standard-size books, "though I only have 32 of my favorites in there so far" says Jake.

"I've got to have this watch!", says the stranger.

"No, you don't understand; it's not ready -"

"I'll give you $1000 for it!"

"Oh, no, I've already spent more than -"

"I'll give you $5000 for it!"

"But it's just not -"

"I'll give you $15,000 for it!" And the stranger pulls out a checkbook.

Jake stops to think. He's only put about $8500 into materials and development, and with $15,000 he can make another one and have it ready for merchandising in only six months.

The stranger frantically finishes writing the check and waves it in front of him. "Here it is, ready to hand to you right here and now. $15,000. Take it or leave it." Jake abruptly makes his decision. "OK", he says, and peels off the watch. They make the exchange and the stranger starts
happily away.

"Hey, wait a minute", calls Jake after the stranger, who turns around warily. Jake points to the two suitcases he'd been trying to wrestle through the bus station. "Don't forget your batteries."

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.