The Embedded Muse 7

Editor: Jack Ganssle (jack@ganssle.com) September 23, 1997

More Dumb Mistakes

From: Steve Litt, whose troubleshooters.com site is a great resource for folks interested in the arcane arts of making things work:

I wasn't winning any popularity contests that day. My client's client was cursing the programmer (me) for "not fixing the problem" that had intermittently (about 0.1% of the time) plagued them for two weeks. My client wasn't happy about sending their best programmer (who was also a part owner) with me to hunt down the problem.

I finally converted the intermittent to a reproducible by assembling a specific set of input files, and we watched in the debugger as a string magically changed for no reason. After watching it a few times, the other programmer gave me a dirty look and said "you can't do that!".

He pointed out that a C function was returning a pointer to a string assembled in a local character array variable inside the function. When the function returned, the character array went out of scope and the memory pointed to by the returned pointer was on the stack and was "fair game" for the next function wanting to change it. Usually nothing changed it before it was used by the calling function, but once in a while ...

There are some mistakes so costly you never make them again. After I paid a locksmith $75 to open my car trunk to retrieve my keys, I never again put keys down inside a car. And I never again passed back a pointer to a non-static local variable.

From Jon Ward of Keil Software

I few years ago I received a phone call with questions about a piece of code that I had written several years earlier. One of the engineers now assigned to the project had a question about the #define pre-processor directive. Their concern was that the Microsoft C compiler didn't properly handle #defines. The values the compiler assigned to them didn't jive with what the engineer expected. Naturally, it must be a compiler problem! :) I knew the particular section of code had been thoroughly tested and worked as expected, so I asked if they had made any changes. No changes...just a little re-formatting. So, I requested they fax me a copy of the offending code. The listing appeared similar to the following.

```
#define PARM1   00020
#define PARM2   01000
#define PARM3   01001
#define PARM4   01002
#define PARM5   01003
#define PARM6   01004
```

After studying the code for a while, I requested that they remove the leading zeros from in front of the DECIMAL constants. The leading zeros caused the compiler to (properly) interpret these numbers as OCTAL constants yielding an unexpected value. After the 5 seconds of stunned silence, I got a hurried thanks and never heard from them again. I guess they probably stopped re-formatting working code after that.

From Clyde Smith-Stubbs at HI-TECH Software

Jack, here's a dumb mistake for you - hardware again. I think the reason you don't get dumb software mistakes submitted is that they're so common I forget about them within minutes of fixing them - then spend just as much time tracking the same dumb mistake next time!

Anyway, I designed a PCB with a Hitachi H8/300H processor - it has a 16 bit data bus, but I only wanted to use an 8 bit ROM, so I set it up to run in 8 bit bus mode. I got PCBs made, built one up, it didn't work. No sweat, that's normal. Three days later I finally went back and read the databook properly - in 8 bit mode the data is read and written on the UPPER 8 data bits. Being used to little-endian chips, I had assumed that if only 8 bits were used, they would be the lower 8 bits! It required some clever cutting and jumpering to make the board work, but it finally did. At least the dynamic RAM I'd designed in (using 16 bit mode) worked fine.

From: John Stauffer

Although I've made my own share of dumb ones this is someone else's mistake I discovered it in an early 80's, 6800 based system that implemented a successive approximation register (SAR) A/D with a combination of discrete resistors, comparator, and software. For making readings the program would switch combinations of resistors which were summed together to produce a voltage, a comparator output would tell if this voltage was higher or lower than the input, resistor combinations would be changed for different & successively smaller voltages until a close match was found and the "A/D conversion" was complete.

This A/D input was used to read the position of a pointing system as it moved up and down, we were finding significant errors between the up readings and down readings. After several weeks I found the SAR A/D was not implemented as a subroutine but with "copies" of the code in the 'up' & 'down' routines. The one difference I found was that the combinations of resistors selected for the same voltage iteration was different. Because the resistors were 5% discrete parts, what the programmer thought were identical resistance values being selected were actually different, resulting in different A/D readings depending on which way the mechanical system was moving. We solved it by using the same code for all A/D readings.

From: Niall Murphy in Ireland

This mistake teaches a valuable lesson about any components that hold state through power cycles. In this case it was a real time clock. As part of power on self test, many of the components of the system were used in some trivial way to ensure that the system had some hope of running.

For the Real Time Clock (RTC), we simply wrote a time/date to it and read it back to ensure that it was operating. Before the write we read the current time of day and then restored it after the write. Some date had to be picked for the test, so I picked my birthday. During the trial period we got occasional complaints that the date was wrong. Because the user could change it back to the right date, it was no big deal, and at first we assumed that the dates were set wrong by some user messing around, and then a different user of the same system raised the complaint. After a while we noticed that all of the dates were late December 1968, shortly after my birthday. I smelled a rat, but could not figure out why the date would be jumping to my birthday, since the test should always restore the current date after the test write. I could not reproduce the problem, but we continued to get occasional reports form users, though the engineers never saw it on any of our systems - leading to a few 'Are you really, really sure no one else changed the date on you' conversations.

Eventually the penny dropped. Occasionally the user would power the system on and then off quickly. The self test would have got to just past the write of the time/date, and then lose power. Obviously this only happened once in every couple of thousand power cycles. The engineers never saw the problem because they were much more delicate with the machine than the trial users. The engineers would always look to see that the self-test was passing before doing anything else with the machine - including turning it off.

The lessons:
1. You can lose power any time, even just after power-on - use a power loss interrupt or some other mechanism to ensure that you do not lose it in an inconsistent state.
2. Engineers are not typical users.

From: Ian Blythe in France

Ian makes this fascinating comment: " I really would recommend that engineers spend at least a few years in a sales environment. In this age of down-sizing, engineers need to learn how to sell their competances."

Agreed, Ian - we need to learn to communicate what we do better to our bosses and others in our organizations (and lives!)

My first job was as an electronics design engineer in an Avionics Company. My responsibility involved the main system architecture for the navigation computer and several CRT and LCD based display units.

Point one. If you are designing a display system, NEVER make mock-up display pages that change as keys are pressed. You'll have a real fun time then explaining to the management that there is still 18 man-months work to go on the *real* software...

Point two. Beware of exhibitions and demo models. Our system worked fine from the emulator, so the night before installation in the exhibition we swapped the emulator probe for a processor. Nothing worked...I personally checked the clock lines, yup 4MHz and 16MHz clocks where present and correct. Finally at 5am a decision was made to take the emulator with the unit to the exhibition and to find somewhere to hide the emulator. We also had to train the stand staff how to boot the emulator up. (I was actually back in the office to go with the kit at 8am...) Needless to say, when it all came back I checked it again, yes the 4MHz and 16MHz clocks were present, but they were swapped! I hadn't cross checked against the pin position on the backplane. Moral being: don't just check the signal, check where it is!

Point 3. If you are developing stuff for an exhibition, always check where it's going to go. At another exhibition, they had decided to place our display into a photographic mockup of a helicopter fight deck. When we got there, sure enough there was a hole for the display unit, but this hole was only the visible screen size, too small for the full unit to slide in... So while all the other stands were polishing off we had a small crowd watching us hacksawing a hole into the photograph. Needless to say, three hacksaw blades later, we managed to squeeze our box in and by the morning it looked really good.

From Derek Somerville in New Zealand

I once ran a project where we designed a system for a company here in New Zealand, who I will call XYZ. The first end user for the system was in England.

It was a good sized project - took four of us almost a year to complete. The first system was tested and then disassembled and packed for shipment by XYZ. I went to England with a person from XYZ and we unpacked and began reassembling everything. As we started to assemble the units, we ran into problems. Nothing would fit properly. We were up against a real problem then, because the client's whole business was based on this equipment.

Finally I clicked that the cases had been changed. When challenged, the dipstick from XYZ finally admitted that he had changed the case design at the last minute, and that these cases were new ones - not the ones we had tested the system in! The reason? They looked nicer.

End result - the client lost faith in XYZ and closed up. XYZ went bankrupt. So did the company I worked for about a year later mostly because of the amount of money we had lost on this project.

Moral - monitor anything that is not under your control, especially when the stakes are so high.

And, another from Derek:

Some years ago I worked in a small company that dealt in cctv equipment. We had in a motor drive amplifier, which took 220V-AC, rectified and filtered it to something just 400v-DC, and then produced 220V-AC again. This enabled equipment requiring stable mains supply to run off something like a portable petrol-driven generator or other supply with unstable frequency and voltage.

My boss had looked at this unit for some time without finding the fault, which was that it would run for some period of time and then start intermittently 'loosing' a couple of cycles here and there. Of course this did nasty things to the stability of video equipment running on the supply.

The only clue was that there was sometimes a small clicking, almost like an arcing sound coming from the unit. However, this was dismissed as not relevant, possibly from the fan or something else.

Anyway, the client came to see how it was going, as we had had the unit in for over a week by this time, and my boss was describing to him the problem and what he had done to try to find the cause. Suddenly, as they were leaning over the unit, there was an almighty explosion, and bits flew everywhere, along with a big cloud of smoke. Both my boss and the client did of course get quite a fright. One of the filter capacitors, several thousand microfarads at a working voltage of 440V, had exploded.

That was the clue that was missed - easy to see in hindsight that the clicking was the capacitor breaking down internally.

The moral - do not dismiss any symptom if you a problem you cannot get to the bottom of.

PS. My punishment for laughing so much was having to clean all the gunk and out the tiny bits of tin foil that had wrapped around everything on the board, which took me a couple of days of painstaking work!

Thought for the Week

In the beginning, God created the bit. And the bit was a zero. On the first day, he toggled the 0 to 1, and the Universe was. (In those days, bootstrap loaders were simple, and "active low" signals didn't yet exist.)

On the second day, God's boss wanted a demo, and tried to read the bit. This being volatile memory, the bit reverted to a 0. And the universe wasn't. God learned the importance of backups and memory refresh, and spent the rest of the day (and his first all-nighter) reinstalling the universe.

On the third day, the bit cried "Oh, Lord! If you exist, give me a sign!" And God created rev 2.0 of the bit, even better than the original prototype. Those in Universe Marketing immediately realized that "new and improved" wouldn't do justice to such a grand and glorious creation. And so it was dubbed the Most Significant Bit. Many bits followed, but only one was so honored.

On the fourth day, God created a simple ALU with 'add' and 'logical shift' instructions. And the original bit discovered that -- by performing a single shift instruction -- it could become the Most Significant Bit. And God realized the importance of computer security.

On the fifth day, God created the first mid-life kicker, rev 2.0 of the ALU, with wonderful features, and said "Forget that add and shift stuff. Go forth and multiply." And God saw that it was good.

On the sixth day, God got a bit overconfident, and invented pipelines, register hazards, optimizing compilers, crosstalk, restartable instructions, micro interrupts, race conditions, and propagation delays. Historians have used this to convincingly argue that the sixth day must have been a Monday.

On the seventh day, an engineering change introduced Windows into the Universe, and it hasn't worked right since.

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.