Real-World Bug [Location: A hostile planet, millions of miles from Earth, deep in space] A small rover sent by sentient beings lumbers up over a rock and begins taking scientific samples. Suddenly, it dies. The home team receives intermittent spurts of garbled data. The firmware locks up, seeming to doom this $820-million mission to failure.

(Analysis of this Real-World Bug appears at the end of this chapter. Each Real-World Bug mirrors the chapter mystery in symptom or root cause.)

Chapter 1

The Case of the Irate Customer: Debugging Other People's Code, Fast

It was an odd-looking line of code, awkward in its form and syntax, dovetailed between well-formatted lines that marched up his computer screen. The pleasant left-and-right rhythm of indentation was marred by this single line, positioned brazenly flush with the left margin.

Not appropriate at all.

It was the offending line's placement that first caught his attention, as if it had been cut-and-pasted by mistake. Closer inspection added to his unease. The original author of this code was not the author of this line - a hack interloper had destroyed the beauty of this software. Oscar raked a hand through his hair as he pulled his focus away from the individual characters and syntax and let awareness of the code's function flood his brain.

It was a command to store a block of data into memory.

He scanned the comment section of the function and found no reference to the change. He wasn't surprised; someone writing sloppy code generally didn't pause to add comments.

But could this line be the source of the emergency, the reason why he'd been summoned back to work at 10 p.m. last night? And then spent the day alternately

hunched over a lab computer and being dragged into various managers' offices to estimate when he could fix a bug that he hadn't yet had time to understand?

Three days before the final hardware and software were to be finished and delivered to manufacturing, the display on the Friend-Finder Communicator device suddenly turned red.

For no apparent reason.

Red.

He closed his eyes and rubbed them, thinking about symptoms of the bug and when it first manifested. The software was officially done and the device had been through System Test with no errors. Then the manufacturing test run was a resounding success and the devices had all passed inspection. The first production devices were packaged and shipped to the customer, who pulled one out of the box, turned it on and found the failure. A second exhibited the same failure. And then, one after another, each device powered up the wrong color.

Everyone had been stunned.

Normally the Communicators powered up an intense blue; a ring of LEDs cast a hazy glow around the edges of the display, making the words appear to float above the screen. With the lights out, animated graphics on the display seemed to rise out of a heavy mist.

Pretty cool.

But the customer's first Communicator had not powered up blue. Last night, Bright Lights' vice president had emailed a digital picture of five Communicators lined up on the table, all glowing an angry, accusing red.

The 7-million-dollar project was barely on schedule. Last-minute changes to the requirements had triggered a cascade of software problems and the lead developer on the project had suddenly resigned.

Oscar imagined the panic following that email. An initial management powwow, followed by realization that the lead programmer was no longer available, and then confirmation of the looming delivery deadline. Soon after, a frantic call to Oscar.

He picked up a Communicator and turned it on. The device happily powered up with the ring of blue LEDs around the display. What was going on?

Oscar turned his attention back to the software. The badly integrated and undocumented block-memory-copy code didn't appear to be related, but he launched the source-code version control program just to be sure, wondering when this software had been changed last. Each Change Request, or CR, to the file was tagged with the date, the reason for the change, the actual software change, and the

version of official software that first included the change. He isolated the sloppy change to a software build over five weeks ago.

Since that change had been integrated, at least one premanufacturing run had been completed successfully. All the devices glowed blue. He made up his mind that the change was unrelated, closed the file and decided to look elsewhere.

This latest problem didn't make sense. The display ring could be configured to glow in several different colors, but those software routines were relatively old and unchanged. Oscar quickly typed the file name and the source code appeared; black characters on a white background. The font size was microscopic, a preference that allowed him to squeeze several pages of code on the screen at once.

The menu to select a display-ring color was pretty straightforward. He picked another of the bad Communicators, still glowing red, from the bench and selected the menu, and then scrolled to "LED Colors" and pressed the SELECT button. He was presented with a list of colors: BLUE, RED, YELLOW, GREEN, and WHITE. Blue was already highlighted at the top of the list. After he pressed SELECT, the ring of LEDs changed from red to blue and the device was happy.

Oscar was not.

Although he had done it several times already, he selected the "LED Colors" menu again and randomly changed the LED colors from blue to red, and then to green and white before changing the color back to blue. All of the menu options worked correctly. Just like they did every time System Test performed the automated test scripts for this product. Even when he power-cycled the device after changing the color, the bug did not reassert itself. Once the color was changed from the original offending red, the bug seemed to be gone for good.

He was losing his patience. As he stood up to get a can of soda, he saw Randy walk into the lab. Oscar sighed to himself and stood in place, wondering how many more management interruptions he'd have to deal with before he could get back to work. If only his cube had a door, he'd close it against the distractions and try to immerse himself in the problem.

This afternoon's meeting with the customer had not gone well and Randy still looked tense. "So, Oscar. Any progress?"

"No. I can change the color of this thing a hundred times and it works correctly. When I turn the power off and on again, it always starts up with the blue LEDs glowing."

"I've found that as well. Is it storing the new settings correctly? Is there any other way to change the color?"

Oscar kept his face expressionless to discourage conversation. His boss didn't write software and his attempts to provide debugging advice left Oscar feeling as if

Randy didn't have faith in him. "I haven't seen any other method yet." He paused to think about how the ring color data was stored. "I can use the debugger to set a breakpoint on a memory access of the color variable and see what function called it. That would reveal the call structure and nesting."

Oscar turned back to his computer and started to type, hoping that Randy would lose interest.

"Oscar, we need to talk about what this thing could cost us. Shipping this software late isn't going to be free. After you left the meeting, we talked about the financials. Bright Lights Corporation wrote penalty clauses into the contract, and they start kicking in if we don't get the manufacturing line on board for the product launch. And, by the way, Kenneth Anders didn't appreciate your feedback about his product."

"Penalty clauses? You signed a contract with penalty clauses in it and didn't say anything to me?" He tightened his grip on the Communicator and envisioned it exploding against the floor. He'd never met this Anders character before today, and he had immediately disliked him.

"I didn't sign this contract, Oscar. I got this over-budget project tossed on me just like you think it was tossed on you. You said the device wasn't that complicated. What's going on that you can't fix this? Can you have some of the people on your team work on this?"

Oscar felt his face getting warm. He didn't like his technical abilities questioned.

He knew he was tired and he pursed his lips to keep from saying something he would regret.

"How much is the penalty?"

"$10,000. Per day."

Oscar felt his stomach drop. "Per day? That's absurd!"

"I agree, but right now I'd rather not be negotiating the clause. Just avoiding it." Randy pulled out a lab stool and sat, staring at Oscar. After a moment, his voice softened slightly as he continued, "No one is going to be taking the money out of your paycheck, but my budget will feel the impact if we get hit with this, so you need to think about a different way to handle this. As you are fond of making clear to me, I am not an embedded developer."

Oscar's fingers froze over the keys. Without looking up, he said, "Well, Josie is between assignments and I asked her to work with the new employee tomorrow. She deserves a little down time to get caught up with email and whatever." He gestured offhandedly at the piles of trade journals stacked against the wall. "Maybe she'll have some time to brainstorm when she gets in tomorrow."

"Good. She seems to be making her mark. You've got a good engineer in her; don't screw that up." Randy stood. "Get this thing working."

Hours later, Oscar pushed back from the bench and expelled the breath he hadn't realized he'd been holding. It was 10:30 p.m. and he was alone in the lab, which was silent save for the periodic hum and pause of test equipment running automated scripts in the background. Closing his eyes, he pinched the skin at the bridge of his nose for a moment and then adjusted his glasses.

Red.

Perhaps foolishly, he'd asked in the meeting if *red* was really such a bad thing. After all, the requirements specified that the display should glow in a handful of colors. Did it truly matter if the device powered up red, and then the user changed it to blue using the menu?

Anders had crossed his long arms, protruding from monogrammed sleeves, and looked down his nose. *Clearly* Oscar did not understand.

"Throughout the movie," Anders informed him, "the color *blue* is synonymous with victory and camaraderie. The movie will end with the group of children defeating the alien using their Friend-Finder Communicators in the blue mode, signifying that they have met their challenges and successfully joined their power to defeat the invasion. *Red* means something else entirely, and not anything appropriate for the initial presentation of the Communicator to the potential audience." Then Anders had opened a folder and removed a thick document, ticking off the contract commitment dates for final software delivery and the start of manufacturing.

Oscar could still feel the narrow hard gaze Anders had directed at him. "The Friend-Finder Communicator *will* be available in stores on the day the movie is released nationally, and when it is powered up for the first time out of the box, it will glow blue. You may think this just a toy, but this is a multimillion-dollar product launch with a very hard deadline. So I suggest you go back and identify which of your colorblind employees has corrupted your software and is placing your job in jeopardy."

Since the meeting, exactly whose job should be in jeopardy was a question that periodically stole into his thoughts. His anger flared at the thought that they could be charged a penalty because the display was the wrong color. Picking up one of several Friend-Finders strewn across the lab bench, he pressed the power button. The display immediately glowed blue and the Friend-Finder logo scrolled across the display.

This one was working correctly.

He had no idea why.

After the heat of the meeting had dissipated, Oscar had to admit that it was actually a neat little gadget - sort of an organizer for young kids who used the Communicator to locate their friends when they wanted to play. The Communicator he was holding showed no other "kids" in his "neighborhood" although it constantly searched for other active Communicators within its area. The Communicators exchanged information about who was available, and the names of those friends were displayed. If Eduardo's Communicator was on, he could send Eduardo a smiley-face. Or, Oscar thought dejectedly, an invitation to go catch a much-needed beer at Molly's Pub.

He could also invite Eduardo to participate in a Challenge: some Hollywood thing where kids used the Communicators to save the earth. The Challenges were games that friends could play when several Communicators came into range. He recalled groups of engineers running through the halls, trying to form a perfect circle around a rogue Communicator. The rogue Communicators flashed and vibrated, spitting out confusing messages to the other devices. When a Challenge was successfully completed, all of the Communicators glowed blue and displayed a victory graphic from the upcoming movie. Cheers had echoed in the halls as the System Test group completed each challenge to validate the software.

So blue was actually a special color of sorts.

Despite his reluctance to admit it to that twit Anders, Oscar was more anxious to solve the mystery of the red display for the technical challenge, rather than for a fear that he would lose his job.

He would nail this bug.

Hudson Technologies was not going to pay penalties on his watch.

Josie grabbed her bag and swung the door of her car closed, locking it behind her as she headed through the parking lot to the employee entrance. The trees the company planted last year were blooming and the sun was shining. Another cold northeast winter was finally over and the "Garden" in Garden State had emerged from hibernation. It was an apt analogy for her own mental state. She'd finished a major project yesterday, and at Oscar's prodding had taken the rest of the day off to "see a movie or something." An appealing thought, but she'd spent the day lounging in front of a stack of DVDs eating sushi and popcorn, and then sleeping like a log through the night with one cat at her head and the other tucked behind her knees.

She felt significantly more clearheaded this morning than she had the last several months. She began ticking through the to-do's when she realized that she'd nearly

forgotten that today was an auspicious day. Today was the new engineer's first day of work, and Josie was to be the Welcome Buddy for the week, making sure Li Mei was introduced around, given a tour of the building, and hosted for lunch the first day. That was more appealing than cleaning out the 521 emails in her inbox.

She smiled, thinking about her first nervous day at Hudson Technologies. She'd accidentally parked in the visitor's lot by the front of the building. Since they locked the front door at 5 p.m., she'd had to walk in the dark all the way around to the front from the employees' lot. Better make sure Li Mei doesn't make the same mistake. Deep in her thoughts, she swiped her ID badge to unlock the door and started up the long hallway to the cafeteria for the first cup of coffee.

Why am *I* nervous, she mused. This isn't *my* first day of work.

With coffee in hand, she made her way through the halls to her cubicle and stashed her bag under the desk, then wiggled the mouse to wake up her computer. She had long fallen into the habit of getting to work early to take advantage of the best debugging time of the day: the quiet before 9 a.m. She began scanning subject lines and deleting old mail and junk. "Daily bug report . . . Friday Lunch-and-Learn cancelled . . . Embedded Systems Newsletter . . . Fake Rolexes!" Ack! Now all she needed were some cheap pharmaceuticals and the morning email would be complete.

She took a long drink from her cup, and saw amid the electronic clutter an email from Oscar time-stamped late last night and set to high priority. Not good. It was addressed to the entire development staff, and the subject line said it all: "CODING STANDARDS ARE NOT A SUGGESTION." Ouch. Oscar was on the warpath again. Instinctively, she scrolled through her latest software in her head, checking for violations and hoping Oscar wasn't venting about something she had just submitted. The email continued with a terse, controlled rant about #defines, comments, and tab spacing, and concluded with an attachment containing the corporate coding standards document.

Probably best to steer clear of Oscar today.

As people began to arrive, Hudson Technologies came alive with typical development-environment chatter and shortly the front desk called. In the front lobby, she found a petite Asian woman who turned to her and smiled. Li Mei wore slacks and a button-down shirt and carried a red backpack. Josie felt herself smile back and introduced herself.

"Good morning, I'm Josie O'Neil. You're Li Mei? Did you find the place okay?"

"Yes, I am happy to meet you. I am Li Mei Cheng. The directions were fine. I arrived too early so I found a place to buy tea and waited for Hudson Technologies to open."

"Good. There are a lot of good places around for food and drink." She gestured for Li Mei to follow her out of the lobby and began pointing out various areas of the company as they walked. "Nearly half of the building is open cubical area. In the middle of the cubicle farm are the labs; that's where we do most of the development, although you can do a lot of software design, coding and documentation in your cube. We'll get your ID card programmed for our lab."

Josie introduced Li Mei around and they dropped off her backpack in her cube before continuing through the halls. Although the one-story building was not large, it took nearly half an hour to make the rounds from cubical to cafeteria, copier room to vending machines. Most importantly, Josie introduced her to the team's administrative assistant, Kathy. "This lady is a goddess," Josie gushed, making room for Li Mei to shake Kathy's hand. "Anything you need, she can do it or get it for you." After invoking a promise from Li Mei to stop by later, Kathy wished her good luck, and they continued on their tour.

As they chatted about Josie's projects, Li Mei asked her, "What work will I be doing? I have my degree in electrical engineering, but I like embedded software better." Then she quickly amended, "Although I am happy to work on anything."

Josie laughed. "Don't worry; you'll be doing different types of projects with a mix of hardware and software. We do a lot of contract work, but we also have a few of our own products that we design and sell, and some spec projects as well." She swiped her ID card and opened a lab door for Li Mei to follow her inside.

"I think Oscar is in our lab. He's a little swamped with a project, but I want him to know you're here. He'll probably meet with you today or tomorrow."

They found Oscar hunched over his bench with his chin cupped in one hand, using the other to step through code with a debugger.

"Hi, Oscar. Li Mei is here and I am giving her the grand tour."

Oscar startled and looked back, and then stood up and brushed the crumpled front of his shirt. Josie thought he looked exhausted.

"Hi, Li Mei. It's good to have you join our team." He shook her hand and then gestured around the room. "This is the embedded software lab; you probably remember from your interview. You'll have your own bench here with a computer, debugger and emulator . . . " His voice trailed off as he gestured haphazardly around the room. "Josie will help you get set up."

"Yes, I remember this lab." Li Mei looked around. The L-shaped room had high ceilings and several rows of benches that continued around the corner.

Oscar seemed at a loss for words, as if he were ready to say something else but couldn't make up his mind. Josie thought maybe it was just a bad time, and started to leave when he spoke quietly to her.

"Have you heard any new rumors about the Communicator project?"

She shook her head. "You sent me home early yesterday. What happened? Is this about the LEDs?"

He dropped his head and grunted. "We had a meeting with the customer yesterday. Idiot named Anders came in waving the contract and ranting about the colors. There's a penalty clause of $10,000 a day for late delivery so this thing is available for sale on the day the movie comes out.

"I know you need some down time, but I'd like to brainstorm with you this morning." He glanced at Li Mei for a moment, and continued, "Li Mei's first assignment is going to be the Meter Magic software that Benjamin didn't finish. I found another of his coding bombs in the software last night."

Josie flushed with relief; Oscar's email *hadn't* been directed at her.

Oscar turned to Li Mei and spoke with a little more determination. "Li Mei, I need to borrow Josie for a while. The Meter Magic project has some documentation that should give you an overview. How about Josie giving you everything and you start going through it?" Li Mei nodded. "You'll own all of that software, so you may as well get started on that."

Glancing back to the Communicator in his hands and fiddling with the buttons, he added, "And don't catch any bad habits from the code you're inheriting. I don't tolerate bad coding habits well."

Li Mei looked between them and stood up straighter. "I can do that. I will review everything and learn it and try not to bother you." She looked suddenly serious.

"Don't worry, Li Mei, we don't lock new employees in their cubes the first week," Oscar said. "You get a little ramp-up time before the torture begins." He gave her a tired smile and turned to Josie. "Would you get her set up?"

"Yes, let me do that now." She grabbed a notebook off a nearby bench and motioned to Li Mei. "Ready, Li Mei?"

"Okay." She stood quietly. "It will be good to start working on something today."

Li Mei's quiet response made Josie pause until she reminded herself how scary the first day at work can be, especially when your new boss looks rumpled and strung out. She promised herself to help Li Mei feel that she was a welcomed member of the team.

Josie returned to the lab a little later and relayed to Oscar that she had gotten Li Mei set up and occupied. Oscar pulled up a stool for her and gave her an overview of the software design. He hoped that her nods meant she understood his quick

explanations. She looks so wide awake, he thought. He dreaded the idea of another long night in the lab if he couldn't get this resolved quickly.

Josie must have sensed his stress. "What is the timeline for this thing?"

He sighed. "That's the problem. The manufacturing line starts running the final software on Friday morning. That's a hard deadline. Working backwards, System Test needs 15 hours to run the test suite, so that must start first thing Thursday morning. The final software build with documentation takes three hours. No one is available to start a build at 4 a.m., but Mahesh will start one as late as 9 p.m. tonight and will stay overnight to baby-sit it. That gives us about 10 hours."

"Tonight?" Josie pulled back and stared at him. "The deadline is tonight and you let me go home early yesterday? I would have stayed."

"You needed a break, and it's probably better that you had a night to de-stress. This is my mess and I didn't want to drag anyone else into it." He didn't add that he thought he'd have it resolved by now.

"Well, what can I do?" She crossed her legs and rested her elbows on the bench. "What have you checked so far?"

He gathered his papers and tried to put his mind back into debugging mode. It felt like fuzz, but he forced himself to think back to the beginning. Staring at a point over her shoulder, without really focusing, he began. "First, I tried to verify the bug. I got one of the units straight from manufacturing that hadn't been turned on yet. When I powered it up, it glowed red. Bad." He smacked the bench with his palm to emphasize his frustration.

"I scrolled through the menus, and the color BLUE was highlighted. When I selected that, the LEDs turned blue. After I repowered the unit, the LEDs stayed blue. It only fails on first power up. I can't get it to fail otherwise." Josie nodded, and he took it as a sign to continue.

"Next, I checked all the latest changes to the software. The only Change Requests involved the graphics and menu options. All straightforward and none related to LED color." He explained the software changes in detail, but all had been code reviewed and tested before they were submitted.

Josie asked, "Do all the new units have this problem? Could they be using a bad programming unit at the factory to load the software?"

"All of the units glow red the first time. A bad device programmer; that's a good thought." He turned to the keyboard and tapped out a quick email. "They halted the manufacturing line, but I'll ask them to try another reprogrammer."

"What about the LED variable? The LED color must be stored in a variable somewhere, I imagine. Did you check accesses to the variable? Is it being over-written? Memory corruption?"

"All good thoughts. I checked the call-tree display already to see what software changes that variable. There are only three places. One is the menu system where the user can manually change the color using the menu option, one is a system test command that automatically changes the color using one of their test scripts, and the third is part of a routine that initializes the entire memory map. The initialization method isn't related to this, but I checked that all three places access the memory map correctly."

He went on to explain that each wrote the value to nonvolatile memory so changes would be saved after the device was turned off. FLASH memory on these portable devices was similar to storing data long term on a computer hard drive. "And before you ask," he added, "the variable is an unsigned char and every reference to it in the code is using an unsigned char, so there is no truncation or data loss."

Oscar wasn't sure if he was overwhelming her with information, but he continued anyway, plugging one end of a cable into the debugging port on the bottom of the Communicator, and the other to a port on his computer. "What I was about to do next was run the debugger with the Communicator connected, and explicitly check the LED color variable as the program runs to see if it's overwritten accidentally. I would have gotten to this sooner if I wasn't interrupted so many times by freaked-out managers. We might find that another array is accessed with a pointer out of bounds and that corrupts our variable in memory."

Josie queried him, "Pointer out of bounds - that's when you declare an array that's 10 elements long and then you accidentally write into the eleventh element?" She printed "array [10]" on her notepad. Oscar leaned over to look at her notes and nodded.

"If the array starts from zero, yes, that's a common bug. C language lets you get away with that very easily. I want to check for things like that." With Josie leaning in to see his monitor, he scrolled to the breakpoint menu. "If this is memory corruption, then the best way to detect it is using a watch point or breakpoint on the memory location itself."

"I use source breakpoints all the time."

Oscar wasn't sure she understood, so he explained as he typed. "This is a little different from a source breakpoint. When you set a source breakpoint, the program runs to that line of source code and then completely stops. Like using an axe." He looked at her. "And then you probably look at the value of that variable and then execute the software one line at a time while checking to see if that variable changes, right?"

Josie nodded.

"And then after a while, when you don't find anything, you let the program run again, and eventually it fails and you don't know why." As he expected, Josie was silent, but she seemed interested as she followed his keystrokes. "Using a watch point is less crude, although it will slow down the execution of the entire program as it runs." In the debugger, he selected "Add breakpoint" and configured it as a watch point on the variable **`LED_Color`**.

"The debugger will check after every line executed to see if this memory address has been accessed. That takes time, but it's thorough and the program doesn't stop running." He started the debugger, and then picked up the Communicator and watched it complete its initialization, or power-on-reset software.

When it finished, he scrolled through the menus to change the LED color and highlighted WHITE. As soon as he pressed SELECT, the debugger stopped and the breakpoint window was displayed. The **`LED_Color`** variable had been changed.

Oscar smiled and raised a finger. "Observe. This is proper behavior. We just verified that the debugger will stop when a new value is written to the LED color variable. Now that we know the debugger is set up correctly, we will try to get that watch point to fire when we're *not* trying to change the color. That way, if the debugger halts, we know something went wrong. Then we can look back in the debugger trace history to find out what part of the software was running when the improper memory access occurred." He reset the program and the debugger. The Communicator restarted and he randomly scrolled through the menu options. As Josie watched, he began changing every user-configurable setting in the device except LED color. The debugger continued to run.

Soon Josie picked up another Communicator and dragged the stool to her work bench, next to his. From the corner of his eye, he watched with satisfaction as she searched through the debugger options and found the watch-point settings to repeat what he was doing. After twenty minutes, she looked over. "It isn't tripping. That means that none of the user controls is causing this bug, right?"

"So it appears. And as I suspected. Otherwise, we would have found the bug before this. But it was important that we checked this for completeness."

She set the Communicator back on the bench. "Okay, what next?"

"Time to interview System Test. They were the first to see the bug." He checked the time on his cell phone. "The Late Risers should be in their lab by now."

The System Test lab was down the hall, but Oscar detoured to the vending machines for a soda.

"I don't know how you can drink soda in the morning." Josie grimaced, then intoned, "Coffee in the morning, no soda until after lunch."

"A little addiction you got there, eh?" Oscar said, unscrewing the bottle. Either coffee or soda tended to sit, forgotten, on the bench as he worked. He detested cold coffee, but warm soda wasn't that bad.

After swiping into the System Test lab, they found Eduardo fiddling with a piece of test equipment.

"Hey, Eddie, we have some questions about the Communicator. You made it go red first. What were you doing right before that?"

Eduardo turned. "I should be on the news; you're the third one to come and ask me that in the last two days. You're the big Technical Manager now. I give you interview number three, you give me the afternoon off?"

"Yeah, right, nice try. For all I know, you seeded that bug." Oscar leaned against the bench and rapped a Communicator against it. "We're not having a lot of luck, so we are trying to reconstruct what happened right when the bug first hit."

"Well, we did the same tests we've been doing all along. Everything is automated. The test scripts include loading the program into a Communicator, and then sending commands through the Communicator port to initialize and test the memory, like the LED color. More tests verify that the units can transmit and receive signals from other units, and simulate the user pressing keys to make sure the menu options work correctly for storing and recalling the Friends database information. You know, the information stored about each friend - name, favorite sound, friend color. After the test completed, we repowered up the unit and it came up red instead of blue." Eduardo shrugged. "Did you change the software? Because we test the blue and it works A-OK during our testing."

"We don't see any software change with the last load that could cause this," Josie said. "Can you make it happen again?"

"After the testing, we found it always happens just once. But we didn't see it until now because we don't turn the devices back on when the testing finishes without failure. The tests pass."

Josie began to doodle on her notepad. "If the color works at the beginning of your test but not after power cycle, then something in the test scripts is causing this. Can we run some of the scripts and see if we can narrow down when it happens?"

Eduardo's face clouded and he shrugged. "We can try. But we're in the middle of an automated test for another product. When that's finished, I'll jump you next in line. When I get your hardware jig and test scripts loaded, I'll call you."

Oscar nodded. "Thanks, Eddie. Just remember that we're in a crunch."

When Josie returned from her welcome lunch with Li Mei, she told Oscar that Li Mei had the documentation and that Ravi offered to take her to security and get her badge programmed for the lab doors. She seemed satisfied that Li Mei was occupied for now.

Oscar looked at her squarely. "Eduardo's not so happy that you implied his test scripts are the problem."

Josie replied defensively, "Well, I didn't mean for him to take it personally. But it could help us narrow down how it happens."

"That's true," he admitted. "Just be aware he's a little paranoid that he will be blamed for this, and with good reason.

"But I had an idea." He changed gears. "If our code is fine and this is a System Test script thing, then one solution is to power on every device from manufacturing and just change the LED color to blue using the menus. Then each device should never have the error again. On the other hand," his enthusiasm quickly dissipated, "someone actually has to do it. Labor costs might be more than the penalty clause until we identify the real bug. I gave Randy a heads-up and he's running the numbers."

"Ack - I sure as heck don't want to be that person!" Josie winced. "You ought to make that Anders guy from Bright Lights do it."

They shared a laugh, and Josie took a drink from her post-lunch diet soda. "Will you show me the Communicator software that System Test uses for the test scripts?"

Oscar turned back to the computer and pulled up a file. "System Test uses the Communicator hardware port as a back door to the software so they can simulate user keypresses and some system behavior. The commands are processed one at a time and ultimately get funneled down to a function called **`process_commands()`**. The Communicator code and System Test scripts both use the same function." He pointed to the code in Figure 1-1.

Reader Instructions: Before continuing, review the function in Figure 1-1. Be forewarned - the code is not well written, but it works. Figure out the overall functionality first. Do you see anything suspicious? Do the #defines help or hurt your understanding? Don't give up - Oscar and Josie will step through the important issues.

"The original author could have used a few more comments and #defines for clarity, but it isn't too hard to understand if you look at the big picture. The function receives a command, **`cmd`**, and the switch uses it to control what code is executed. The function also takes a pointer to an array of characters that contains data to be read or stored, an offset, and a return code for status of the operation."

```
/* Friend-Finder Communicator Software Code Listing (Partial)              */
#define MAX_NUM_FRIENDS       10
#define MAX_FIRSTNAME_LEN     15
#define MAX_LASTNAME_LEN      15
#define SIZEOF_FRIEND         (MAX_FIRSTNAME_LEN + MAX_LASTNAME_LEN + 3)
#define MEM_START             0x00
#define FRIEND_AREA_START     0x00
#define PARAMETER_AREA_START  (FRIEND_AREA_START + MAX_NUM_FRIENDS*SIZEOF_FRIEND)
#define FIRSTNAME             0x00
#define LASTNAME              (FIRSTNAME + MAX_FIRSTNAME_LEN)
#define FRIEND_TYPE           (LASTNAME + MAX_LASTNAME_LEN)
#define FRIEND_COLOR          (FRIEND_TYPE + 1)
#define FRIEND_SOUND          (FRIEND_COLOR + 1)
#define DISPLAY_LEDS          PARAMETER_AREA_START
#define TRANSMIT_PARAMS       (DISPLAY_LEDS + 1)
#define BATT_CHARGING_PARAMS  (TRANSMIT_PARAMS + 10)
#define POWER_CONTROL_PARAMS  (BATT_CHARGING_PARAMS + 6)

enum Colors {LED_RED, LED_BLUE, LED_YELLOW, LED_GREEN, LED_WHITE};

/* Structure for Friend-Finder Database Entry */
struct friend_data_entry_type {
   char first_name[15];
   char last_name[15];
   char friend_type;
   char friend_color;
   char friend_sound;
   char friend_available;
} friend_data;

struct friend_data_entry_type Friends[MAX_NUM_FRIENDS];
char LED_color;
char RxTx;
char Battery;
char Power;

void process_commands(char cmd, int offset, char *data_ptr, char *return_code)
{
   char status;
   int  len;
   switch (cmd)
   {
      case 0x01 :  len = (strlen(data_ptr) <= (MAX_FIRSTNAME_LEN-1))
                      ? strlen(data_ptr) : (MAX_FIRSTNAME_LEN-1);
                   status = store_data (FRIEND_AREA_START
                      + (offset*SIZEOF_FRIEND) + FIRSTNAME, data_ptr, len+1);
                   break;
      case 0x02 :  len = (strlen(data_ptr) <= (MAX_LASTNAME_LEN-1))
                      ? strlen(data_ptr) : (MAX_LASTNAME_LEN-1);
                   status = store_data (FRIEND_AREA_START
                      + (offset*SIZEOF_FRIEND) + LASTNAME, data_ptr, len+1);
                   break;
      case 0x03 :  status = store_data (FRIEND_AREA_START + (offset*SIZEOF_FRIEND)
                     + FRIEND_TYPE, data_ptr, CHAR_SIZE);
                   break;
      case 0x04 :  status = store_data (FRIEND_AREA_START + (offset*SIZEOF_FRIEND)
                     + FRIEND_COLOR, data_ptr, CHAR_SIZE);
                   break;
      case 0x05 :  status = store_data (FRIEND_AREA_START + (offset*SIZEOF_FRIEND)
                     + FRIEND_SOUND, data_ptr, CHAR_SIZE);
                   break;
      case 0x06 :
store_data (offset,data_ptr, sizeof(friend_data));  break;
      case 0x10 :  status = store_data (DISPLAY_LEDS, data_ptr, CHAR_SIZE);
                   break;
      case 0x20 :  status = store_data (TRANSMIT_PARAMS, data_ptr, 10*CHAR_SIZE);
                   break;
      case 0x21 :  status = store_data (BATT_CHARGING_PARAMS, data_ptr, 6*CHAR_SIZE);
                   break;
      case 0x22 :  status = store_data (POWER_CONTROL_PARAMS, data_ptr, 3*CHAR_SIZE);
                   break;
      default :    break;
   }
   *return_code = ERROR;
   if (status == 1)      *return_code = OK;
}
```

Figure 1-1 Software Listing to Process Commands.

Oscar interrupted himself. "Oh, and the factory also returned my email - the programming devices are not the problem. Apparently happens on every one of them."

"Too bad," Josie sighed.

"Anyway, each command stores some data. They look confusing but they're similar. The first several compute the length of the incoming data string first, and then use the conditional operator '**? :**' to truncate the length to the max allowable length if it's too long for that database entry. That's shorthand for an if-then-else statement."

She nodded her understanding and he pointed to the first several. "Look at the #defines. These write entries into the Friends database, stuff like first and last name. Each command is a different database entry or parameter. At the bottom of the switch are commands for parameters like battery operation and transmit and receive power. **LED_color** is the 8-bit variable that we're interested in."

He straightened on the stool and turned to her. "Notice anything interesting, without knowing more about the code?"

Josie peered at the screen. "Yes, **status** isn't initialized. If **cmd** isn't handled in the switch statement, the return code argument in **process_commands** will be indeterminate."

"Good catch. I didn't see that. But that's not what I was looking for."

She continued to inspect the function. "Command 0x06 is what prompted your hate email last night, right?"

Oscar grunted. "That would be the cause." He'd forgotten about the email he had dashed off in frustration early this morning. "Anything else?"

"The command after that is the one that writes the LED color. Coincidence?"

"It could be. But remember that those two chunks of code don't run consecutively. Only one command is executed each time through the function. The switch statement controls the execution based on the input command. But you're on the right track. Check out the #defines for the memory map. Don't get caught up in the details."

She scrolled to the top of the file and located the #define for **DISPLAY_LEDS**. "All of the parameters are located in memory right after the Friends database. So if anything overwrote the end of the database, it would overwrite the LED color."

"Bingo."

It was later in the afternoon by the time Eduardo finally found them. Oscar and Josie had walked through the code several times and he was reasonably certain that they both understood the calling structure. When they arrived back in System Test, Eduardo had the Communicator in the test jig and he started the automated test scripts. They watched the display scroll through menus and access data, seemingly controlled by invisible hands. Oscar's thumbs felt tired just looking at the rapidly changing screen. Shortly after testing started, the LEDs glowed blue, followed by red, green, yellow, white, and finally back to blue. After several minutes, the test completed and the device powered down. Eduardo removed it from the test jig and handed it to Oscar.

"You can do the honors."

Oscar pressed the power button.

The display graphic flickered and the LEDs glowed red. He groaned, but quickly reversed himself. "This is actually a good thing - if it's red now, then we can work backwards. Josie and I took a look at the software used to write the memory, and the Friends database is located right before the LED color variable. We'd like to run the test scripts one at a time, and stop just past the point where the LEDs are tested but before the database is written."

Eduardo gestured for the Communicator and placed it back in the jig. "Okay, honored debugging guru, we run to the database. You think the database is the problem?"

Oscar ignored the wisecrack. "I do. We'll check the variable right before the database is written, and then check it again afterwards. I suspect it will be changed."

Eduardo halted the scripts after the LED test was completed and checked the **`LED_color`** variable. Its value was 1; the LEDs were configured for blue. He restarted testing, and then rechecked the variable and turned to Oscar.

"The variable has changed to zero, but the LEDs are still glowing blue. What does this mean? Is it the bug?"

Oscar felt a smile appear on his face and the stress seep from his shoulders. That singular point when he knew he had a grasp on the tail of the bug was always an intense relief.

The value for red LEDs was 0.

"It surely is the smoking gun. And, no offense, but I am extremely pleased that your script pulled the trigger." Oscar saw Eduardo tense defensively, and he gestured for patience.

"Eddie, I still don't know where this bug is, but your test script flushed it out. This is a major victory, but we still haven't solved the problem. Don't get bent, okay?"

"System Test gets blamed for a lot, Oscar. Don't let my manager know about this until you know for sure where the problem is. I don't need more frustration."

"You're great, Eddie. Josie and I are going to go check the software that's used to store the database elements, and we'll get back to you."

"Okay, I got the System Test script manual for the database that describes the test. The script fills the database completely. It doesn't hold much - only ten friends' worth of data." Oscar showed her the list. "And the friend color is different from LED color. Each kid can pick a color for his name."

Each Friend Finger Database Entry:

First name	15 bytes
Last name	15 bytes
Friend Type	1 byte
Friend Color	1 byte
Friend Sound	1 byte

Josie spread the document on the bench and drew a diagram (Figure 1-2). "Let me understand - all commands get funneled down to **`process_commands()`** to change the LED color during normal operation, right?" He nodded, noting she'd also identified that the color could be changed by initialization. "So all the System Test script commands that come in through the test port ultimately end up using the same functions as the internal menu system."

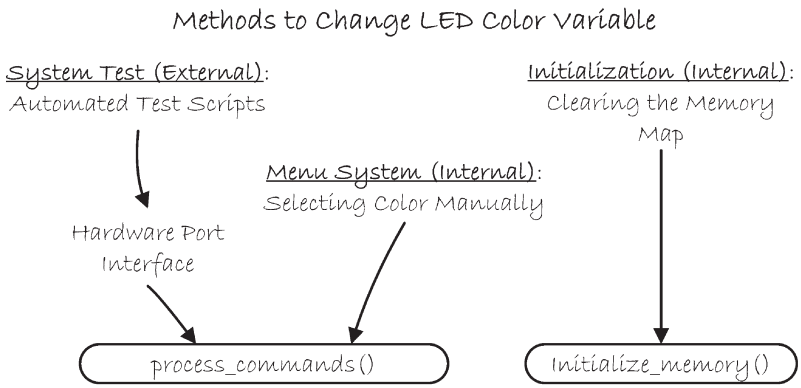

Figure 1-2 Three Methods to Change the LED Color Variable.

"Yes. The Communicator software that writes the LED variable shouldn't know the difference. We're all using code like this - it takes five commands to fill one database entry. I built a debugging function to fill the entire database, but this software doesn't cause the problem."

```
construct_cmmd(0x01, 0,"Kid1Firstname");
construct_cmmd(0x02, 0,"Kid1Lastname");
construct_cmmd(0x03, 0,"0");
construct_cmmd(0x04, 0,"0");
construct_cmmd(0x05, 0,"0");

construct_cmmd(0x01, 1,"Kid2Firstname");
construct_cmmd(0x02, 1,"Kid2Lastname");
construct_cmmd(0x03, 1,"1");
construct_cmmd(0x04, 1,"1");
construct_cmmd(0x05, 1,"1");
```

Oscar pulled down the front of his shirt to straighten it, noting despondently that the effort was futile. His earlier confidence was beginning to fade as the deadline approached. The building was nearly cleared out for the night, but he noted with relief that Josie didn't fidget to pack up and leave.

If everyone used the same command, he pondered, why couldn't they duplicate the problem from the internal Communicator interface? They'd have to do it differently.

"That code looks okay. Is there any easy way to know exactly what parts of memory got changed?" She picked up the soda bottle and started to drink.

"Well, we can dead beef it."

Josie nearly choked on the soda. "'Dead beef it'? What the heck does *that* mean?" She wiped her mouth with the back of her hand.

"Ah, you young ones. There is much you do not know." He smiled sublimely and assumed the crossed-arms pose of the wise instructor. Josie rolled her eyes, preparing for the lecture.

"Dead beef is one of the interesting phrases that can be formed using only the available hexadecimal characters A through F. DEADBEEF. A trick to checking what parts of memory have been written is to first lay down a recognizable fill pattern in memory. DEADBEEF is a historical pattern. After the program in question runs, it's easy to see where the pattern is broken." He smiled down his nose at her and said in his best pompous authority voice, "We shall DEADBEEF the memory and retest the software."

Josie laughed as he opened a file and began to code. "Okay, so how do we do this 'dead beef' thing?"

She seemed genuinely interested in the vintage method and he decided to teach her more about detecting memory corruption.

He asked, "You read a lot of sci-fi, don't you?" Seeing her nod, he continued, "Who's your favorite science-fiction author?"

The question caught her off guard, but she was quick to reply, “Marion Zimmer Bradley. No wait, Piers Anthony is good too, and Isaac Asimov.”

“Any others?” His eyes never left the screen as he typed.

“Well, how about Orson Scott Card?”

Oscar soon had another new debugging function. This one filled the entire memory with the repeating phrase DEADBEEF. He placed the call directly before his function to populate the database.

```
debug_init_mem_map_deadbeef();
debug_fill_friend_memory();
```

He commanded the debugger to run to the second function, and then showed Josie the memory filled with the repeating phrase (Figure 1-3). The memory viewer showed 20 columns and 24 rows of data in hex on the left, with the ASCII representation of each on the right. “See, the four ASCII characters for ‘DEAD’ are stored as 0x44, 0x45, 0x41, 0x44.” She nodded for him to continue.

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9    01234567890123456789
 0: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
 1: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
 2: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
 3: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
 4: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
 5: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
 6: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
 7: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
 8: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
 9: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
10: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
11: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
12: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
13: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
14: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
15: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
16: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
17: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
18: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
19: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
20: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
21: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
22: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
23: 42454546444541444245454644454144424545 46    BEEFDEADBEEFDEADBEEF
24: 44454144424545464445414442454546444454144    DEADBEEFDEADBEEFDEAD
```

Figure 1-3 Hex and ASCII Dump of DEADBEEF on Patterned Memory.

After stepping over the **`debug_fill_friend_memory()`** function, the DEADBEEF pattern was overlaid with a hastily assembled Top 10 of Who’s Who in Science Fiction and Fantasy (shown in Figure 1-4).

Josie laughed as she scanned the right side of the memory viewer on Oscar’s monitor. “Pretty smooth! Oh, Dan Simmons - I forgot him!”

Each character of the friend’s names wiped out the underlying DEADBEEF character, leaving snippets of the repeating phrase to show through among the

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9    01234567890123456789
 0: 49736161630045464445414442454541736 96D6F   Isaac EFDEADBEEAsimo
 1: 760045464445414442453030305069657273 0046   v EFDEADBE000Piers F
 2: 444541444245454641 6E74686F6E790044454144   DEADBEEFAnthony DEAD
 3: 424545313131526F6265727400454144424545 46   BEE111Robert EADBEEF
 4: 4453696C766572626572670042454546323232 44   DSilverberg BEEF222D
 5: 6F75676C6173004442454546444541646 16D7300   ouglas DBEEFDEAdams
 6: 444541444245454644333333526179004445414 4   DEADBEEFD333Ray DEAD
 7: 424545464445414272616462757279004245454 6   BEEFDEABradbury BEEF
 8: 444534343450686 96C6970204B2E004644454144   DE444Philip K. FDEAD
 9: 4469636B004541444245454644454135353547 65   Dick EADBEEFDEA555Ge
10: 6E650044424545464445414442526F6464656E62   ne DBEEFDEADBRoddenb
11: 6572727900454144363636 4D6172696F6E205A69   erry EAD666Marion Zi
12: 6D6D6572004542726164 6C6579004546444454144  mmer EBradley EFDEAD
13: 4237373744616E004245454644454144424545 53   B777Dan BEEFDEADBEES
14: 696D6D6F6E730046444541444245383838 4F7273   immons FDEADBE888Ors
15: 6F6E2053636F747400454546436172640045454 6   on Scott EEFCard EEF
16: 444541444245453939390 1040505040305030200   DEADBEE999□□□□□□□□□
17: 000101020203030606064546444541444245454 6    □□□□□□□□□EFDEADBEEF
18: 444541444245454644454144424545464445414 4   DEADBEEFDEADBEEFDEAD
19: 424545464445414442454546444541444245454 6   BEEFDEADBEEFDEADBEEF
20: 444541444245454644454144424545464445414 4   DEADBEEFDEADBEEFDEAD
21: 424545464445414442454546444541444245454 6   BEEFDEADBEEFDEADBEEF
22: 444541444245454644454144424545464445414 4   DEADBEEFDEADBEEFDEAD
23: 424545464445414442454546444541444245454 6   BEEFDEADBEEFDEADBEEF
24: 444541444245454644454144424545464445414 4   DEADBEEFDEADBEEFDEAD
```

Figure 1-4 Hex and ASCII Dump of Good Code on Patterned Memory.

famous authors' names and the three friend's numbers following each entry. He showed her that each name was terminated with a NULL character, or zero, so the software could detect the end of each name string. When they reached the bottom of the list, his humor evaporated. Data from the last author was overlaid straight to the end of the Friends database memory area, but no further.

The **`LED_color`** variable remained unchanged.

Still 0x01.

Still blue.

Josie pointed to the printed squares on the right. "I don't understand. **`LED_color`** is a number but it's printed as a square, but numbers that are in the Friends database like Orson Scott Card's '999' are printed as real numbers."

"Remember, this is a debugging display - memory contents are not always in a format that is easily readable. Values less than Hex 20 are nonprintable characters, and that square you see is just a placeholder for a nonprintable character." He pointed to the corresponding data location on the left and circled the **`LED_color`** variable. "For these, you have to look at the actual hex value on the left side of the display. It is correct; the **`LED_color`** value is 0x01 and that means blue." The System Test script was not the source of the bug.

Josie turned to face Oscar. "Oh, I get it. You thought the software would overwrite the end and change that value to a zero."

"Yes. Back to Eduardo."

"Hey, Eduardo, thanks for staying late. We just duplicated what the System Test script sends to the Communicator to fill the Friends database. It doesn't corrupt memory like your script did, and it doesn't cause the LED color problem. Are you sure that's the test you are running?"

Eduardo tossed his hands in the air. "I know what tests are being run. I know this system inside and out! The only thing that changed with that test is that we requested an update to your software a couple months ago for a more efficient command to fill memory. That was tested and released. Last week we changed the contents of the Friends database to test another CR, but that has nothing to do with the code or the LED color."

Oscar stopped short.

"You changed something?"

"Not the software - just the data we load into memory."

"Show me." Oscar walked him to the computer and Eduardo pulled up the source code for the test script that successfully reproduced the failure (shown in Figure 1-5).

Like a well-shot arrow, Oscar's finger was already smudging the screen. "This constructs a test string to use command 0x06? We are not using that command. Is this test script calling the new Communicator code you mentioned?"

"Yeah, that's it. Before, we had to fill a friend's entry one field at a time and it took a lot of code, so we requested a command to write an entire entry with one line. Is that related to the problem?"

"I don't know, but we need to redo our debugging test using that command. From our end, it's poorly implemented." He didn't add that the system-test

```
/* System Test Automated Script (partial code listing)
   Product:   Friend Finder Communicator
   Test #12:  Fill database memory
   Function:  Send 10 commands with sample entries.                    */

void fill_database(void)
{
   int  i = 0;

   construct_cmmd(0x06, 33*i++, "First          Last          111");
   construct_cmmd(0x06, 33*i++, "First          Last          222");
   construct_cmmd(0x06, 33*i++, "First          Last          333");
   construct_cmmd(0x06, 33*i++, "First          Last          444");
   construct_cmmd(0x06, 33*i++, "First          Last          555");
   construct_cmmd(0x06, 33*i++, "First          Last          666");
   construct_cmmd(0x06, 33*i++, "First          Last          777");
   construct_cmmd(0x06, 33*i++, "First          Last          888");
   construct_cmmd(0x06, 33*i++, "First          Last          999");
   construct_cmmd(0x06, 33*i++, "First          Last          111");
}
```

Figure 1-5 New System Test Software to Write Friends Database Information.

implementation with hard coded "33s" was about the same level of quality as the 0x06 command. Instead, he spun on his heels and marched out of the lab. Josie shrugged at Eduardo and followed Oscar out.

Oscar ranted as he headed back to the embedded lab. "Did you see that code? That's another thing that gets me. Just because software running in System Test isn't shipped, it doesn't mean it shouldn't be well tested and documented. And now they are using custom commands to test our memory and bypassing the very commands that we are using. That defeats the entire purpose of testing!" He stomped to his bench.

While he fumed, Josie slid up to her own bench and keyed up the file containing the new command. The 0x06 command was similar to the others in the function. Rather than receiving one Friends parameter at a time, it appeared to receive and write an entire entry at once. **`Friends`** was the global variable containing the entire database; 10 entries of type **`friend_data_entry_type`**.

At first glance, it appeared to be okay.

Oscar tried to calm down as he changed his debugging program to fill memory using command 0x06, mimicking the system-test scripts as closely as he could. He wanted fiercely for this test program to cause the bug. The deadline was upon them, and it would be sweet to nail the author of this hack. He glanced at Josie, who was still scrolling through the code. After his change compiled, he ran the program and opened the output file (shown in Figure 1-6).

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9    01234567890123456789
 0: 49736161632020202020202020204173696D6F    Isaac           Asimo
 1: 76202020202020202020303030506965727320 0    v         000Piers
 2: 20202020202020416E74686F6E792020202020          Anthony
 3: 2020203131315 26F6265727420202020202020       111Robert
 4: 2053696C766572626572672020202032323244     Silverberg     222D
 5: 6F75676C6173202020202020204164616D7320     ouglas         Adams
 6: 20202020202020202033333352617920202020020          333Ray
 7: 20202020202020427261646275727920202020020         Bradbury
 8: 2020343434506869 6C6970204B2E202020202020    444Philip K.
 9: 4469636B20202020202020202020353535476 5     Dick            555Ge
10: 6E652020202020202020202020526F6464656E62    ne            Roddenb
11: 6572727920202020363636 4D6172696F6E205A69    erry    666Marion Zi
12: 6D6D6572202042726164 6C6579202020202020     mmer  Bradley
13: 2037373744616E2020202020202020202020 53      777Dan          S
14: 696D6D6F6E732020202020202020 3838384F7273    immons        888Ors
15: 6F6E2053636F747420202020436172642020202020    on Scott   Card
16: 2020202020202039393900405050403050302000          999□□□□□□□□□
17: 0001010202030306060645464445414442454546    □□□□□□□□□□EFDEADBEEF
18: 4445414442454546444541444245454644454144    DEADBEEFDEADBEEFDEAD
19: 4245454644454144424545464445414442454546    BEEFDEADBEEFDEADBEEF
20: 4445414442454546444541444245454644454144    DEADBEEFDEADBEEFDEAD
21: 4245454644454144424545464445414442454546    BEEFDEADBEEFDEADBEEF
22: 4445414442454546444541444245454644454144    DEADBEEFDEADBEEFDEAD
23: 4245454644454144424545464445414442454546    BEEFDEADBEEFDEADBEEF
24: 4445414442454546444541444245454644454144    DEADBEEFDEADBEEFDEAD
```

Figure 1-6 Hex and ASCII Dump of System Test Code on Patterned Memory.

Reader Instructions: What are the differences in the patterned memory? Can you conclusively verify the presence of the bug?

"Got it." He smacked the bench with his fist. A tiny smile crept onto his face. "Check it out - look at the difference in memory patterns for the good code and the bad code. What do you see?"

Josie leaned over to look at his display, where he had both memory patterns displayed side by side. He watched her take in the differences. Even though the System Test software completely overwrote the pattern in between the names, the first and last names still appeared at the same place in each memory map. That apparently didn't affect how the software ran.

She straightened. "Sheesh, I see it. After the '999' in the System Test case, there is a blank space before the DEADBEEF starts again. That's exactly where the **`LED_color`** variable is stored. Somehow that new command ends up stepping on memory one byte beyond where it is supposed to. That's the bug!" She furled her eyebrows. "But that doesn't explain why the LED didn't turn red when Eddie ran this piece of software for us."

"That's what we must discover." Now that he had nailed the bug down, it needed to be unraveled and fixed. Normally he enjoyed the process of understanding exactly how it worked and then implementing the fix, but they were out of time.

His computer beeped and launched an email notification. Just as they proved that they could invoke the bug, Mahesh was expecting the completed software change to launch the official build. How had it gotten to be 9 o'clock so quickly? They could probably delay Mahesh a short while, but they hadn't even proposed a fix or created a test build to verify it.

With a sigh, Oscar ignored the email and returned to the memory dump. As he pondered, Eduardo's unanswered question from that afternoon resurfaced. If the System Test scripts had changed the LED color variable but the LED still glowed blue, were they looking at the right variable? He glanced at the Communicator beside him; they had just changed the software using the System Test scripts and the LED glowed blue.

The symptom replayed in his brain. The LED color variable was changed to RED, but the LEDs still glowed blue.

Suddenly he remembered part of the original symptom. They had to power cycle before the LED color changed. He explained his train of thought to Josie as the Communicator rebooted. But instead of the red he anticipated, it again powered up blue.

Oscar was stunned. "I am getting a little crazy here - we need to stand back and be logical." He put the Communicator on the bench and raked his hands through his hair. "We have a couple of avenues to explore. First, why is command 0x06 overwriting the LED color variable? And second, why doesn't that reproduce the problem?

"Command 0x06 writes a string of data from System Test," he said, pointing to the **`fill_database()`** function from the System Test script code, "and the string is not too long. Look, the string ends with the three parameters."

Josie asked, "Since they are writing a string rather than a number, is the software automatically storing a string terminator at the end like the NULL character?"

"That's a thought, but look at the code for command 0x06. It doesn't use terminators; it explicitly sends the length of the data as the third argument. It sends the size of the **`friend_data`** structure." He started to think, and scrolled up to the declaration. "The Friend-Finder Database Entry structure is 15 plus 15 plus 4 bytes long. 34 bytes."

Suddenly Josie dipped her head backwards and exclaimed, "Oscar - I get it! I know what's happening! System Test is using the data structure to derive the memory size instead of using the #defines for size." She pointed to the #defines at the top of the file. "Each **`SIZE_OF_FRIEND`** entry is 33 bytes, not 34 bytes!"

He quickly scanned the file and found that she was right. Josie had seen the tiny clue that he had missed. "Check it out - the **`friend_data`** data structure is one byte larger - the last byte is a temporary byte that's only used when the program is running. It tracks if a friend is currently available. It's not a part of the permanent database in FLASH."

She continued in a rush. "The new command that System Test started using only supplies 33 bytes worth of data, like it's supposed to. But whoever coded command 0x06 used the **`sizeof(friend_data)`** to derive the size, and it accidentally told the software to write 34 bytes. Who knows what's in that last byte of data? It could be the mystery value for RED."

He stopped. None of this explained why a power cycle was needed to invoke the bug, and then his brain flooded with complete understanding. Suddenly everything fell into place and he pushed out his stool to pace with newfound energy.

"Okay, see if this makes sense. First, System Test requests the new 0x06 command and begins using it. For some reason, the extra garbage byte that gets written is a zero, just like we reproduced here. If the **`LED_color`** value is zero, the LED color is red, right?"

Josie was nodding. Everything had been working correctly, completely by accident. Her Communicator now lay forgotten on the bench as she followed his pacing.

"Next, Eddie said that they changed the contents of the Friends database. Even that shouldn't affect that final critical garbage byte, but who knows. They did something that changed RAM - whatever was stored in the variable directly after that command buffer." He continued walking back and forth between the rows of benches, trying to remember the rest of the System Test program, but then decided it didn't really matter. "And that single byte became mystery data byte number 34."

He stopped pacing and faced Josie, his face hot. "And the reason that we don't see the LED change color immediately is that these commands are writing the data to FLASH memory. They commanded a change to the database, so the database was updated in FLASH, and then the local database variables were updated. The command accidentally changed the LED color variable in FLASH, but not local memory. The critical variable, **`LED_color`**, didn't get changed immediately. The only time we saw the change is after a power cycle.

"And what happens after a power cycle?" He looked at her, expecting to see her sharing his enthusiasm, but she looked lost again. He willed himself to slow down.

"Remember what happens after power-on reset. The program initializes the hardware and peripherals, and then reads FLASH memory to populate all the local variables in RAM before the rest of the program runs. FLASH holds the data values while the power is off. When the power comes back on, everything in RAM is garbage until it's initialized from FLASH." He stopped and crossed his arms, thinking that he'd never really considered that the problem could be due to a mismatch between the volatile and nonvolatile memory storage locations for the same variable. He'd have to remember this the next time.

He continued, "That's when the LEDs turned RED. Why?"

Josie broke eye contact. He imagined her mental wheels turning, and then prompted her again. "That's when the bad data was read into RAM for the first time."

"Are you saying that the new data isn't stored in the RAM variables like **`LED_color`** at the same time it's stored to FLASH?"

He shook his head. "Not at all." He drew a picture on the whiteboard behind him, starting with a square labeled NONVOLATILE FLASH and another labeled VOLATILE RAM. Inside each he printed the words "Display LED color" and drew a line to connect them, and repeated the same with Friends database variables.

Oscar pointed to the squares. "When a variable is changed, both the RAM and FLASH locations are supposed to be written at the same time, but *only for the variables that are explicitly commanded*." He tried to be as clear as possible, knowing that juggling two memory locations for the same variable could be confusing. "When the bad command to update the Friends database was executed, the 34 bytes

of data were written to FLASH. But the program only commanded it to update the Friends variables, not the **`LED_color`** variable. So only the RAM version of the Friends database was read from the newly written FLASH and stored locally for use." He tapped on the bench for emphasis. "But there was no command to update the LED variable, so FLASH held the new corrupted value without copying it to the corresponding RAM variable. The RAM variable for **`LED_color`** didn't get written with the bad data until the power cycle when *everything in RAM* had to be written from scratch." He paused and waited for his explanation to sink in. "Does that make sense?"

Josie nodded. She had a serious look on her face, but he thought he could detect the light bulb warming up.

"Now, here's the plan." Oscar had some ideas about how to proceed now that he understood the problem. "While you're still thinking, I want you to fix command 0x06 in the Communicator code so that it only writes 33 bytes, as I believe it was intended to do, and then test the fix. Verify with the DEADBEEF memory-patterning function that the **`LED_color`** byte is no longer overwritten. I am going to find the entire set of System Test code for the Communicator and check their scripts."

Josie showed him her pad. "Wouldn't it just be this one line change to fix the last size argument and store the return status byte?"

```
status = store_data(offset, data_ptr, SIZEOF_FRIEND);
```

"Yeah. Good catch on the status byte again. Test it."

It was past midnight when they packed up to go home. Before he left, Oscar sent Randy a short email, "No software fix tonight. Already sent Mahesh home."

Oscar was already parked at his bench in the lab when Randy arrived first thing in the morning. As he made his way through the benches, Oscar could tell that he had received the late-night cryptic email.

"Mahesh told me you never called him to do a build," Randy said.

"No, we didn't."

"You've passed the deadline. I'm on my way to tell the factory to reschedule now. Is that what I should be doing?"

"I'm not changing the code. There's nothing wrong with it."

Disbelieving, Randy stared back at him.

"Really, it's under control, we're not going to pay any late fees. Josie and I figured it out." Oscar explained everything they had discovered about the System Test procedures and the mysterious command 0x06.

"This was all a strange set of circumstances, but it turns out we don't need to change the Communicator software immediately. After we discovered what caused the failure, I modified the System Test automated scripts to send the Communicator the correct commands instead of the broken 0x06 command. Eduardo is checking my changes and running the new script now. If the Communicator passes using the modified scripts, then we will have verified that the source of the problem was the bad command."

Oscar paused to gauge Randy's reaction. Randy motioned him to continue.

"What's the better news is that we don't need a new version of Communicator software. Josie fixed the broken command, but to make this deadline we are still using the one that the factory already has approved and loaded."

Randy interrupted him, "I am missing something. You're describing a change that needs to be made, so what happened to all the build and System Test time you needed before release to manufacturing?"

"Oh, that's what saves this deadline! All that time is required if we rebuild the Communicator software to fix the 0x06 command. We are not doing that. Manufacturing normally runs an abridged version of the full System Test suite that we run here. We're just going to restore a previous version of those scripts that uses the correct commands before the bad command 0x06 was introduced. We should have that completed and verified in the next few hours."

Randy dropped onto a stool and let out a long breath. A series of emotions passed over his face, from relief to pleasure to exhaustion.

Oscar quickly added, "This isn't System Test's fault, either. The bad command in our code is the source of the problem." He wanted to complain about the new custom command, but Randy was off on his own mental checklist.

Oscar waited until Randy looked up again and asked, "Are you sure? Is this all going to work?"

"Josie code-reviewed everything I changed in the System Test script. She also verified that fixing the bad 0x06 command in our software also fixes the problem, but we don't want to go that route because it would require we recompile the software and we already missed the deadline for that." Oscar finally let a smile break onto his face. "If we take the path I have proposed and update the System Test script instead, we are still on schedule. No penalties."

Randy stood and tucked his hands under his arms. "Good. Get it done."

He silently held Oscar in his gaze for several moments before he spoke again.

"I hope you realize that you can't continue to act like the lone developer any longer. You have a good reputation for firefighting the emergency problems, but you've got a team that you're not leveraging. You were promoted to technical manager and it isn't clear to me that you are making that transition effectively."

Oscar paused. He thought about Josie's involvement, and realized that her help had probably saved the deadline. Just having someone to bounce his ideas off of. He amended the thought, admitting to himself that she had been much more than that; she had understood him and her attitude inspired him to explain things. He looked up to see Randy still studying him.

Oscar shook his head, thinking that a few hours of sleep and several cups of caffeine just weren't enough to overcome the sleep deprivation he felt. Tomorrow was another day to think about what all this meant. He followed Randy out of the lab to catch another can of soda on his way back to System Test to let Eddie off the hook.

This bug was most certainly nailed.

Chapter Summary: The Case of the Irate Customer (Difficulty Level: Moderate)

In this mystery, Oscar is called in to debug an emergency problem that appeared after all testing was successfully completed and the Friend-Finder Communicator was ready to ship. The device suddenly began behaving differently, even though no software or hardware had been changed. In addition, it only failed after first power-up, right out of the box. By collaborating with Josie and with Eduardo in System Test, they find that a combination of an old software change and the payload of a new automated script used in System Test and Manufacturing causes the problem.

The Problem Symptom(s):

- Device glowed red on first power-up rather than blue as required.
- All subsequent power-ups correctly glowed blue.
- Problem only occurred after manufacturing or system test, and could not be duplicated in the lab.

Targeted Search:

- Search in the code for all places the color could be changed.
- Review of recent software changes.

The Smoking Gun:

- The color variable was the first memory location in a logical block of memory, suggesting overrun from the previous data structure.
- Changing the value of the color variable did not immediately update the LED color.

The Bug:

- An incorrect structure length was used to write data into a database. This caused a 1-byte memory overrun that corrupted the color variable stored just after the database in FLASH. This bug was masked because the corrupted FLASH value was only used to update critical variables in RAM after first power-up.

The Debugging Method Used:

- Interviewing system test engineer.
- Reviewing recent changes in the defect tracking system.
- Searching the call structure for accesses to the **`LED_color`** variable.
- Using watch points to allow continued code execution while waiting for a memory value to change.
- Using patterned memory (i.e., “DEADBEEF”) to isolate memory writes.
- Using a team member as a sounding board to work through ideas.

The Fix:

- Changed block-memory copy command to use the correct database length (33 rather than 34 bytes).
- Used a #define rather than a hard-coded value for database length.

Verifying the Fix:

- Tested both the software fix and the system test script rollback using the patterned memory method.

Lessons Learned:

- Nonshipping software used in development and testing should be code-reviewed and placed under version control.
- Brainstorming with a team member can quickly generate a target list of ideas to explore.
- Breakpoints configured as watch points allow continued code execution.

Code Review:

The software in Figure 1-1 is pretty awful. The data are described twice: once as the **`friend_data`** structure and once as list of #defines, and the layouts are not identical. A coder might be tempted to simply copy into the data structure and the data would not end up in the right places. And the descriptive #defines at the top are positive, but why didn't the developer use #defines for array lengths and to clarify the confusing cases in the switch? The **`process_commands()`** function can be rewritten a number of ways to increase readability and maintainability. Oscar was rightfully upset at the software in Figure 1-5; the structure is accessed inappropriately and this test code does not duplicate the actual code.

What Caused the Real-World Bug? During the seven-month cruise to Mars, the Mars Expedition Rover Spirit collected science data to FLASH memory. Lots of it. Once on the planet's surface, even more data was stuffed into FLASH. But a flaw in the file system software prevented memory from being correctly deleted after the data had been sent to earth, allowing all available free memory to be quickly consumed. When more memory was requested, the system crashed. A watchdog timer faithfully rebooted the vehicle's code but on each start-up the code first tried to allocate yet more FLASH. So it rebooted again and again till the batteries nearly died, driving the system into a safe mode.

When engineers uploaded diagnostic software, they were able to free up memory, fixing this embedded application from 100 million miles away. The problem could have been caught in analysis - but wasn't. Testing,

too, showed evidence that had not been investigated. Ground tests never exercised the system in "a flight-like way," certainly nowhere near the mission goal of 10 months. [1]

References

[1] Reeves, G., Neilson, T., and Litwin, T. (2004), "Mars Exploration Rover Spirit Vehicle Anomaly Report," Jet Propulsion Laboratory Document No. D-22919, May 12, 2004.