

# CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

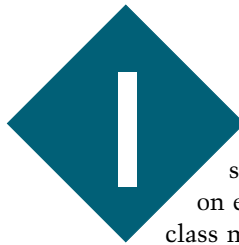
## LESSONS FROM THE TRENCHES

George Martin

## Embed This PC

### Part 2: Emulator and EPROM Basics

Continuing with his latest project, George shows us how to use emulators in an embedded '486 project to monitor the design. There are a few options to choose from, he'll show you how to make the best choice.



Last month, I started a discussion on embedding a '486-class machine. I defined the controller for a laser light show as a typical application, gave you several sources of background information, and investigated one particular DRAM device. Your assignment was to write up your system requirements and search through the literature for a CPU and DRAM.

When you build your system, you'll experience what it feels like to bring up your very first system. You're not sure if anything is correct—the design, artwork, boards, or soldering.... In fact, the entire development process will probably have felt like the foolproof method for sculpting an elephant—you got a huge block of marble and chipped away the things that didn't look like an elephant.

Well, you get the picture, and you've probably been there before. When you're in this place, you need a plan to check out your design. Which

brings us to—what options do you have with emulators?

### EMULATOR OPTIONS

Of course, we could purchase the classic emulator that replaces the CPU and operates at full speed. Although this is the best solution both for hardware and software development, at \$20,000–50,000, the cost is high, very high.

I could write, compile, link, and locate the code, creating an image of the program that we could download into the EPROM. The EPROM is then plugged into the unit and powered up. Although this is the least expensive approach, it is perhaps the most time consuming.

These examples represent both ends of the spectrum. Either way, you spend money up front in equipment or at the end in labor. But, are there other choices?

I could monitor the external bus signals with a logic analyzer. After capturing the bus cycles, a logic analyzer converts the bus activity into CPU instruction activity. Unfortunately, this method has a significant weakness since CPU activity is not always displayed on the bus and not necessarily at the time you expect. With instruction prefetch queues and internal memory caching, you soon realize that the external signals make little sense. Of course, I could disable some features, but then I'm not running the CPU at full speed. The cost of the logic analyzer must be taken into account.

Most highly integrated CPUs have JTAG ports, which that and write internal registers. These registers can be set to clock-cycle execute with the results being read out on the same JTAG port. It offers the classic control and observability that you want in testing devices. Because JTAG ports

Signal	DRAM 1		DRAM 2		DRAM 3		DRAM 4		
	Bank 0		Bank 1		Buffer				
	Low	High	Low	High	Low	High			
MA0	17	17	17	17	18	18	74LVTH244ADW	2	
MA1	18	18	18	18	16	16	74LVTH244ADW	4	
MA2	19	19	19	19	14	14	74LVTH244ADW	6	
MA3	20	20	20	20	12	12	74LVTH244ADW	8	
MA4	23	23	23	23	9	9	74LVTH244ADW	11	
MA5	24	24	24	24	7	7	74LVTH244ADW	13	
MA6	25	25	25	25	5	5	74LVTH244ADW	15	
MA7	26	26	26	26	3	3	74LVTH244ADW	17	
MA8	27	27	27	27	18	18	74LVTH244ADW	2	
MA9	28	28	28	28	16	16	74LVTH244ADW	4	

**Table 1**—Here are the DRAM connections for the Elan SC400. As you can see, DRAM 1 and 2 share Bank 0, while DRAM 3 and 4 share Bank 1.

can perform emulation functions, a set of emulators focuses on this approach, which is called BDM (Background Debug Mode). Although this is a good choice, it's not available on all CPUs.

An onboard monitor EPROM can be used to load and run programs, insert breakpoints, single step, and all the other features you desire in an emulator. It's a good choice, but I really want quick loading and a single step in the high-level language (at least C source code) that I might use. Mind you, monitor EPROMs have grown into complete solutions that include loading, single step, breakpoints, and source-level debugging. The only thing lacking is trace capability. If a monitor EPROM works, it is both low cost and easy to use.

## MAKING THE CHOICE

Before I select an emulation plan, I want to look into the CPU a bit more. First, I'm going to assume the CPU is in a surface-mount package, which I think is reasonable. Because there are 250–400 pins, finding a connector that lets me remove the CPU and plug in an emulator will be difficult at best.

Second, I'm assuming that I've selected a highly integrated device. If I apply power and a clock, all I need is memory and an LED or serial port to determine activity. The CPU is permanently attached to the board and, with power and a clock, it's going to run.

Given these assumptions, I think an EPROM emulator is a good choice for my embedded '486 project. The development plan: write code, transfer

it to the EPROM emulator, and reset the CPU using the EPROM emulator. The unit should then run the code I loaded.

## LAYING OUT THE TOOLS

Now that some of the peripheral issues are covered let me look more closely at the CPU. I found two classes of devices—those intended for desktop or laptop systems and those intended for a more minimal existence. If you need desktop or laptop characteristics, then use the first type of CPU.

I'm focusing on the second group. I need to consider floating-point support (FPU), L1 and L2 cache, the bus interface, and speed. I'm going to use the AMD ELAN series of embedded devices, which consists of the SC300, SC400, and the just-announced SC500. These devices incorporate a lot, and I mean a lot, of the support circuitry from the desktop design into the more minimal, embedded CPU.

The SC300 family is a '386 machine, the SC400 is a '486, and the SC500 matches the Pentium. Each device uses different supports, which are built into the chip. For example, serial, parallel, DMA, keyboard, and LCD controller are built into these devices. Others have a FPU and PCI bus interface. Check AMD's web site for a more complete overview ([www.amd.com/products/lpd](http://www.amd.com/products/lpd)). You can get a list of their evaluation boards at [www.amd.com/products/lpd/prodsupp/20071.html](http://www.amd.com/products/lpd/prodsupp/20071.html). And, vendor support is available at [www.directories.mfi.com/embedded/](http://www.directories.mfi.com/embedded/)

amd-e86/. Keep in mind that if you select a different device for your project, most of the vendors and their products are applicable in some form.

The Elan family ranges from '386 to '486 without FPU, and to '586 with FPU. Clock speeds range from 25 to 133 MHz, and interfaces range from simple I/O to PCI support. I think you should be able to find a solution that fits your price/performance requirements.

In the web pages for the AMD (and other) CPUs, there's a list of development tools such as development boards, compilers, and emulators. I recommend finding a development board that's close to your design, so you can use that as the basis for your schematics and software development.

For language tools, I know I need an assembler and a C compiler. As a '486 machine, I could theoretically use anything that runs on the desktop. I'm a big fan of Borland's Turbo C V.3.0 (DOS), which I use to write C code that I can debug in the PC and then compile for the target machine. Of course, you could use Microsoft Visual C++ or a host of others. Keep in mind that these compilers are 16 bit. If you're wanting to work with protected mode, tune in next month.

A vendor that I have used in the past and one which works well with the Borland set of tools is Paradigm ([www.devtools.com](http://www.devtools.com)). Paradigm has taken Borland's compiler and wrapped it into an integrated environment that does the usual editing and compiling, while also offering downloading and emulation through its Remote Debug Utility. If I have a CPU running PDREM, then I can load code, single step, insert breakpoints, and examine variables. It's been around for years and is a stable product.

Now, for the last piece of the puzzle. Because I don't want to burn EPROMs to transfer and test the code, the obvious solution is an EPROM emulator. Some of my customers have used these in the past and reported good results. I tried a couple of years ago without much luck (I always suspected it was my fault). To successfully embed a CPU as complicated as a '486, I need an emulator or the com-

bination of a remote debug and EPROM emulator. Again, there are several manufactures, but the one I selected was Grammar Engine's Prom Ice ([www.gei.com](http://www.gei.com)).

## GETTING THE DEBUGGER GOING

I set up the debugger by plugging the EPROM emulator into the EPROM socket and attaching it to my PC using a serial or parallel cable. Loading code from powerup is a two-step process. First, I load the PDREM software into the EPROM Emulator and then my application. Once the PDREM software is loaded, I can just keep changing and reloading my application code as long as the application controls the CPU. If my CPU locks up or spins off into never-never land, I need to reload both the PDREM and the application.

With PDREM support, I can look at assembly language and single step the CPU, as well as keep an eye on the memory and I/O. In fact, I can even operate at the C source level, monitoring variables and stepping through complete C source statements. All in all, it's quite powerful.

## EPROM SELECTION

Although I described how this EPROM emulator connects, I skipped over the EPROM I'm emulating. Even though EPROMs come in all sorts of sizes and shapes, I typically think of them as the devices programmed and then erased using UV light. However, now that the IC's features are so small, UV light is no longer practical. In fact, many EPROMs are one-time programmable (OTP).

I also need to consider flash-memory devices for my EPROM. Flash is erasable and reprogrammable, even while it's in the circuit. The PC BIOS has used flash-memory technology for several years now.

As I sort through the devices, packaging again becomes one of the definitive parameters. EPROMs are available in a dual-inline package (DIP) of up to 32 pins. Although the package is large, it's easy to work with.

Its size is limited, though. The largest I've found is only 8-bit devices with a 512K x 8 memory con-

figuration. If you want a larger EPROM or a wider data path, then you have to use two devices in parallel or switch to a surface-mount package (SMT). If you use SMT, then 16-bit data paths and 2M x 16 flash-memory devices are readily available.

EPROM emulators come with connectors that plug into DIP sockets. Depending on the device I select, I'll need adapters to make the connection. Also, the 8-bit interface is standard, and if I use wider data paths, I'll need additional EPROM emulators connected in parallel. While all this is possible, it adds a whole lot of complexity to development, so I'll opt for the easier route.

For this project, I've selected the 512K x 8 flash-memory device as my EPROM, and I'll get it in a DIP package.

## PUTTING IT ALL TOGETHER

Now that I've selected a CPU, DRAM, and BIOS, I'll try to put together a design. Take a look at the AMD ELAN SC400. You can find the schematics for the development boards at the AMD web site. The CPU comes in a 292-pin ball-grid array (BGA) package.

I'll start by connecting the DRAM. I use OrCAD as my design-capture software, but I'm still in the 16-bit DOS version, so I'm hesitant about posting those schematics. And, unless your requirements exactly match mine, you'll be rolling your own schematics anyway. Hopefully, the AMD web site and this series will give you what you need to be able to generate your own schematics.

Let me try this approach. Table 1 illustrates the DRAM connections. Most are directly connected to the CPU, while some go through buffers. Depending on your specific application, you may not require the buffers. Remember my previous article on defects. Well, the type of mistake I tend to make is with errors in large tables of data—like that spreadsheet. So, check my work. Don't just copy it.

As you can see, it looks pretty straightforward. I connected two banks of DRAM (each bank is 1M x

32) to the CPU using the 1M x 16 DRAM devices I talked about earlier. This gives a total of 8 MB of DRAM.

Now, let me configure the CPU to use the DRAM. The ELAN SC400 has configuration registers—lots of them! Much of the classic PC architecture is preserved 100%, so the COM0: I/O addresses are unchanged. To find places for the new registers without causing a conflict with the existing code base, AMD found a few free I/O registers using a different technique. It adopted an index register into the new register set. For example, I/O addresses 0x22 and 0x23 are the register pair used to configure DRAM. In I/O address 0x22, I write the index, and in I/O address 0x23, I write the value:

```
Outb(0x22, 0x00);  
Outb(0x23, 0xAD);
```

The first instruction points the index register to index zero, while the second instruction sets the value. Index register zero is the DRAM Bank0 configuration register, and the value 0xAA does the following:

```
1 = Bank 0 Enabled  
0 = Reserved AMD  
1 = EDO type DRAM  
0 = Symmetrical Addressing  
1 = 32-bit DRAM interface (for speed!)  
010 = 1-Mb deep bank
```

These are the values required for this particular DRAM and design. You'll quickly notice that the CPU supports many more options than you'll ever come across in your design experience. But, it's good to know they're all there.

## LOOKING AHEAD

This should give you a start on your project. Next month, I'll go further into the software, taking a closer look at the EPROM emulator and real versus protected mode.

As you poke away at your initial design, remember indecision is the key to flexibility. ▣

*George started his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and cofounded a design and manufacturing firm. Typical systems that George designs include servo-motion control, graphical input and output, data acquisition, and remote control. George is a charter member of the Ciarcia Design Works Team and most recently, he's been working on the people-tracking system for Bill Gates's new house. You can reach him at [george.martin@worldnet.att.net](mailto:george.martin@worldnet.att.net).*

## SOURCES

### **SC300, SC400, SC500**

Advanced Micro Devices

[www.amd.com/products/lpd](http://www.amd.com/products/lpd)

### **Borland's Turbo C V. 3.0 (DOS)**

Borland

[www.borland.com/borlandcpp/turbosuite](http://www.borland.com/borlandcpp/turbosuite)

### **Remote Debug Utility**

Paradigm

[www.devtools.com](http://www.devtools.com)

### **Prom ICE**

Grammar Engine

[www.gei.com](http://www.gei.com)

Circuit Cellar, the Magazine for Computer Applications.  
Reprinted by permission. For subscription information,  
call (860) 875-2199, [subscribe@circuitcellar.com](mailto:subscribe@circuitcellar.com) or  
[www.circuitcellar.com/subscribe.htm](http://www.circuitcellar.com/subscribe.htm).