

LESSONS FROM THE TRENCHES

George Martin

Embed This PC

Part 4: Designing Peripherals In

George has covered the CPU options and selected DRAM and BIOS flash memory devices for the embedded '486 project. In the final part of this series, he ties it all together and takes a look at what kinds of peripherals you might want to add on.



This is the last part of my four-part series about designing an embedded '486-class CPU. We've looked at several CPU options and selected DRAM and BIOS flash devices. I pointed you to schematics and other resources to help you get your design off the ground. This month, I'd like to wrap up the series by taking a look at some peripherals you might like to add.

WHAT TO HOOK ON

With many '486 designs, you have to plan for a big chunk of memory whether you use flash memory for program storage or an EEPROM or static RAM (SRAM) for data storage. The CPU provides hardware interfaces for DRAM and BIOS EPROM as part of its basic design. Built into its logic is refresh control and wait-state generation.

Some also provide other types of memory, and many offer interface support for the different cards and buses (e.g., PCMCIA, ISA VL, and

PCI). All this comes with the PC keyboard, IRDA, and legacy devices, with the exception of a tape deck. If you have a standard card, such as a PCMCIA, the manufacturer has details regarding how to hook that up.

Be careful, though. Not all of these interfaces are available at the same time. Selecting one type of interface often precludes another. You probably can't have PCMCIA and ISA interfaces both supported by the CPU as a result of pin limitations.

THE SQUARE PEGS

What about unusual or custom interfaces? Many highly integrated devices offer general-purpose pins, which can be used as chip selects or have their address, control, and timing individually configured. Again, you need to be careful. There are never enough of these pins, and the range of address and control is not exactly what you need for your design. I've come across this issue several times.

Most chips let you attach memory devices in one of three or four standard ways. If your need is not standard, built-in pins aren't the answer. Most CPUs also let you access an external bus with all the control signals. With PAL programming tools, you can create your own logic.

I've combined CPU pins and customized PAL logic. With this arrangement, the CPU does the major decoding over a large address block. The CPU then controls the bus direction and timing, while the PAL details the interface to a particular device.

I keep referring to a laser light show as a typical design. To run lasers, you need D/A converters that you load with data as fast as possible. If you use D/A converters connected to the bus and mapped in the I/O

address space, the design looks a lot like an ISA interface. Assuming that you're programming in C, the CPU would perform an out instruction, *out(address, data)*, with 16 bits for the I/O address and 16 bits for the I/O data. The compiler translates that command into just a few assembly language instructions, which first loads dx with the address, ax with the data, and then performs an out instruction. It's all simple and straightforward.

THE HARDWARE INTERFACE

The hardware interface requires the CPU data bus to connect to the D/A data bus. The D/A converter has chip select (CS\ and write (WR\ inputs, which are typically active low. However, don't make any assumption about the timing of the D/A converter. It needs to be looked at in detail. Devices with no setup and hold-time requirements can interface using a PAL without much worry. Devices with timing requirements on CS\ and WR\ need more care and a register type of design. You will find requirements such as CS\ must be low, 15 ns after WR\ goes high.

This is the stuff that gives you gray hair—or makes you lose your hair. If the timing requirements of your devices differ too much from what the CPU is generating, you'll need to make the PAL a clocked device. If you add a high-speed clock, you can capture and hold the values of the offending pins.

Listing 1 outlines the logic for decoding the '486 bus signals and connecting a dual D/A converter to the bus. The PAL decodes and controls the bus signals. Keep in mind that this is only an example and doesn't necessarily give you the best, cheapest, or fastest way of doing things. You'll find there are a number of ways to connect memory and peripherals in an embedded '486 design. Just decide which way suits your application best.

THE ARTWORK

Finally, let's talk about the artwork for your design. If you attempt to put as much circuitry on one

printed-circuit board (PCB) as possible, you'll undoubtedly end up in surface-mount technology (SMT). That's not so bad, if you keep the use and implementation of that technology under control.

SMT mounts parts on the pads of the PCB surface instead of using through holes for components to be inserted into. Through-hole spacing is 0.100", which lets you put component leads on a 0.100" grid. Look at any

Listing 1—Sometimes, the best way to work out the hardware interface is with a little PAL logic. In this sample, I show you how to decode '486 bus signals and connect a dual D/A converter to the bus.

```

; PAL Descriptive file

Inputs:
MRD\      Memory Read ; line goes low when external memory is
read
MWR\      Memory Write ; line goes low when external memory is
written
AEN       Address Enable ; signals when the DMA controller has
the bus
IORD\     I/O Read ; line goes low when I/O device is read
IOWR\     I/O Write ; line goes low when I/O device is written
A25:A20   Address Lines A25 to A20
A11:A0    Address Lines A11 to A0

Outputs:
SRAMCS\   SRAM Chip Select
DEVICEA\  I/O Device A
DEVICEB\  I/O Device B
DEVICEC\  I/O Device C

Equations:
; Set the variable to not conflict with SC400 internal registers or
legacy I/O devices
; 11 10 9 8 7 6 5 4 3 2 1 0
; 0 1 0 0 0 0 x x x x x x

; I/O devices 0x400 to 0x43F

; when you see IOBASE, substitute the stuff between the ''
string IOBASE '(/A11 * A10 * /A9 * /A8 * /A7 * /A6 * /AEN)'

; Set up SRAM from 0x0800000 to 0x08FFFFFF
; RAM requires reads and writes, which excludes DAM activity
; with 512K x 16 memory, connect A19 and lower address lines to the
chip
; use A20 and above for the decode
/SRAM_CS = /A25 * /A24 * A23 * /A22 * /A21 * /A20 * /MRD * /AEN
+ /A25 * /A24 * A23 * /A22 * /A21 * /A20 * /MWR * /AEN

; Now some I/O devices
; 11 10 9 8 7 6 5 4 3 2 1 0
; 0 1 0 0 0 0 0 0 0 0 x x x Device A
; 0 1 0 0 0 0 0 0 0 1 0 0 0 Device B
; 0 1 0 0 0 0 0 0 0 1 1 0 0 Device C

; This equation generates a chip select for an I/O device with 8
internal registers.
; Address lines A0, A1, A2 go directly to the chip
; IOBASE includes upper address lines and excludes internal and
legacy devices (IBM/PC)
; chip select is generated for either an I/O read or write
/DEVICEA = IOBASE * /A5 * /A4 * /A3 * /IORD
+ IOBASE * /A5 * /A4 * /A3 * /IOWR

/DEVICEB = IOBASE * /A5 * /A4 * A3 * /A2 * /A1 * /A0 * /IORD
+ IOBASE * /A5 * /A4 * A3 * /A2 * /A1 * /A0 * /IOWR

/DEVICEC = IOBASE * /A5 * /A4 * A3 * A2 * /A1 * /A0 * /IORD
+ IOBASE * /A5 * /A4 * A3 * A2 * /A1 * /A0 * /IOWR

```

dual inline package (DIP), and you'll see what I'm talking about. With 0.100" pad spacing, a pad of 0.050" diameter, and a track of 0.012" thickness, you could route one track between pads and keep a 0.013" clearance.

Tracks can also be routed on a 0.025" grid with that same clearance maintained. But, this layout commonly uses 12 mil lines and 12 mil spaces. Engineering design practices extended this to 10 and 10 by switching to a 0.010" line and a 0.010" grid, which lets you put two lines between IC pads.

In contrast, SMT lead spacing can be 0.100", 0.050", 0.020", or even smaller. With the smaller spacing, you can't route tracks between the pads until the spacing gets to 0.020". Actually, it is possible, but 5 and 5 design rules are expensive when you get into production. Just so PCB costs don't get out of hand, it's usually best to keep to 10 and 10 or 8 and 8.

All the devices I talked about in this series are available in a DIP or 0.050" SMT package, with the SMT packages typically referred to as SOJ or PSOP. Be careful if you see a TSOP package type. That's the 0.020" category. But, the TSOP-2 (a.k.a., TSOP type 2 or TSOP-II) uses larger spacing and might work for your design.

You have the same headaches with resistors and capacitors. Package types, such as 1206, 0805, and 0603, will come flying across your desk. Just duck. Because the numbering convention indicates package size, the 0805 is approximately 0.080" by 0.050", and so on.

Pick shapes small enough to fit your design, but don't go so small that you drive your manufacturing costs out of sight. I chose the 0603 package size for a recent design because I thought I might need pull-up resistors on a bus. I was going to run a track between the IC pads, which had 0.050" centers. If each pad needed a pull-up resistor, then the resistor needed to be small enough to permit the track to pass. Resistors larger than 0603 block the routing lanes.

BALANCING DESIGN FACTORS

The point I'm trying to make about the artwork is that you need to bal-

ance the size with the cost of the parts, PCB, and manufacturing. There's no such thing as a great design if the design rules you used are so unnecessarily complicated that you can't afford the PCB.

I've run out of general guidance for embedding a '486. If you have questions about this series or specific issues, post a question in the *Circuit Cellar* news groups. Currently, I'm beta testing for 32-bit versions of various tools sets, and I'll write about those soon. This is an industrial/commercial grade design that can be customized and licensed. If you have any interest, please contact me.

Next month, let's embed the Internet—almost. ☒

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' new house. You can reach him at george.martin@worldnet.att.net.

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