



DESIGNING WITH PC/104 — A TUTORIAL

Rick Lehrbaum
Vice President, Strategic Programs
Ampro Computers, Inc.

Over the past ten years, the PC architecture has seen growing use as a platform for dedicated and embedded microcomputer applications. PCs are beginning to be found everywhere! They're being used as embedded controllers within vending machines, laboratory instruments, communications devices, portable and mobile systems, and medical equipment, to name a few examples. Why?

WHY THE TREND TOWARDS EMBEDDED-PCs?

From a computer architect's perspective, the PC "platform" is not the most highly efficient or sophisticated, with its 8088-based origins and inherently segmented world view. Why, then, turn the world's favorite *desktop* system into an *embedded* microcomputer standard? Why not continue using Z80s, 68HC11s, 8051s, and the like?

Regardless of its particular implementation, from 4- and 8-bit single-chip microcontrollers to high performance 32-bit RISC processors, the embedded microprocessor is merely a means to an end — not an end in itself. The purpose of an embedded microcomputer is, after all, to run the application software. Increasingly, it's the application-specific *software*, not the embedded *computer*, that makes the embedded system what it is. The ideal embedded computer is, therefore, one that minimizes risks, costs, and development time — provided it runs the application software acceptably.

Microprocessor architectures vary greatly. Each new microprocessor or microcontroller brings with it new development tools, including emulators, compilers, and debuggers. Every time you employ a new microprocessor in a project, you should expect to invest thousands of dollars and weeks (or even months) of time acquiring — and learning to use — the required development tools. It's no wonder system developers are looking for an alternative to always using the latest new microprocessor in every new project. Furthermore, It isn't uncommon for older products, based on earlier microprocessors, to become difficult or impossible to maintain, as familiarity with the older architectures and their associated development tools fades or as the engineers who designed them move on.

For these reasons, hardware and software engineers are beginning to migrate away from proprietary, project-specific microcontroller architectures towards well-defined hardware and software standards whenever possible. From the software perspective, this involves using structured languages like C and C++, employing object-oriented programming methods, working

within familiar and broadly supported operating system environments like DOS or UNIX, and interfacing to software “APIs” (application program interfaces) like Windows (for graphics) or TCP/IP (for communications).

The situation is not so clear on the hardware side, where the tremendous diversity of microprocessor and microcontroller architectures, from the lowly 8051 to the high-end RISC CPUs, has prevented the emergence of any real standards for embedded system hardware. Only the industrial computer buses such as VME, Multibus, and STD offer a degree of consistency — but their use is confined to systems which are larger, more complex, and less cost-sensitive than the typical *embedded* system.

Meanwhile, the highly multisourced PC-compatible 386/486 CPUs, chipsets, and associated peripherals have been making the PC architecture increasingly interesting as an alternative to traditional embedded controllers in systems requiring low to medium performance. With an estimated 200+ million desktop PCs in use worldwide, and nearly a million new ones *each week*, the PC architecture has even become known as the "Industry Standard Architecture" ("ISA").

And why not! The tremendous popularity of the PC architecture has spawned an entire industry of hardware, software, and publications specifically targeting the PC platform. PC-oriented hardware products include CPUs, chipsets, peripheral controllers, and even PC-specific peripherals (displays, disk drives, modems, pointing devices, etc.). PC-oriented software products include numerous real-time OS's, device drivers, function libraries, and application programs. Countless books and magazines document all aspects of the PC's hardware, software, and applications. Development tools for the PC platform are plentiful, cost-effective, and easy to use. And most importantly, most engineers and programmers are already knowledgeable and experienced in working with the PC's hardware and software.

Using the PC architecture in an embedded application can therefore mean big savings in development time and money, lower product costs (due to less expensive chips and peripherals), and fewer headaches in maintenance and support. That's why the PC architecture is increasingly being used as an embedded microcomputer standard.

MAKING THE PC FIT

After you decide to use the PC architecture in your next embedded system project, you'll need to find a way to satisfy your application's likely constraints on factors like size, power consumption, reliability, and ruggedness. You'll soon realize that normal PC system components (boards, peripherals, and even the BIOS software), which are aimed at the extremely price-sensitive desktop *personal computing* market, don't meet these typical constraints of embedded systems. Don't blame the desktop PC manufacturers for this situation; they are merely serving their customers, whose buying decisions are based mainly on price. Desktop PC buyers aren't opposed to ruggedness, quality, and reliability — they just don't want to pay extra for it.

Even if quality, reliability, and ruggedness weren't an issue, mechanical and environmental constraints are likely to prevent you from using *desktop* PC motherboards, expansion cards, and

peripherals in most embedded applications. Standard PC boards and peripherals are simply too big to fit within the tight space and power consumption budgets of most of embedded systems.

To circumvent these problems, why not design your own specialized, chip-level embedded PC directly onto a circuit board containing the application-specific electronics required by your embedded system? This way, you could create a more robust, reliable, rugged implementation of the standard PC — more like the traditional embedded microcontrollers you've used in the past. With this approach, you would hope to reduce the costs, risks, and elapsed time of your development project by taking advantage of standard PC hardware and software components and development tools, yet produce a robust, reliable, and rugged embedded computer.

Unfortunately, there's an inherent problem with this strategy. Because you still need to design and debug CPU, memory, and I/O subsystems, as well as license and port a PC BIOS, you really haven't eliminated many of the costs and risks you were hoping to eliminate by adopting the "off-the-shelf" PC architecture. Once again, you'll find yourself back on the "microcomputer development treadmill"! Be prepared to buy or rent an in-circuit emulator and to invest \$25,000 or more in a BIOS source license. What's worse, unless your specialized embedded-PC turns out to be very nearly 100% PC-compatible, you'll be disappointed when you discover, after all your effort, that many of those PC-compatible development tools, operating systems, device drivers, and application programs you hoped to take advantage of don't work as planned.

You are faced with a dilemma: you want to use the popular PC architecture to gain use of all the PC-compatible hardware and software components, technology, development tools, and know-how; but standard desktop PCs and peripherals aren't well suited to the unique requirements of embedded systems. Actually, you don't really want to embed a *Personal Computer* — you're trying to incorporate an embedded *PC architecture*. Seen from this perspective, the term "embedded-PC" is somewhat of an oxymoron, because the main requirements of *embedded systems* (high quality, robustness, minimal size and power, fail-safe operation) run counter to the top priority of *personal* computer buyers (low price).

This leads you to the thought: "if only there were a version of the PC specifically tailored to the needs of embedded systems . . ." This desire is what inspired the creation and rapid proliferation of the *PC/104 Embedded-PC Modules* standard.

WHAT IS PC/104?

PC/104 offers full hardware and software compatibility with the standard desktop PC (and PC/AT) architecture, but in an ultra-compact (3.6" x 3.8"), self-stacking, modular format. Basically, PC/104 standardizes a way to repackage desktop PC functions in order to satisfy the ruggedness, reliability, and size constraints of embedded systems. Consequently, PC/104 provides an attractive PC-compatible alternative to traditional embedded microcontrollers.

Although PC/104 modules have been around since 1987 (in Ampro's "MiniModules"), it was not until 1992, when Ampro released the formal PC/104 specification and established the PC/104 Consortium, that interest in PC/104 skyrocketed. Since then, hundreds of PC/104 modules of many types have been announced by members of the 140-company PC/104 Consortium. In

1994, PC/104 achieved a significant milestone, when Intel became a member of the PC/104 Consortium and endorsed PC/104 as a recommended way to expand designs based on its new embedded-386 CPUs.

In 1992, the IEEE Microcomputer and Microprocessor Standards Committee formed a working group with the objective of standardizing a small form-factor version of the PC/AT bus, based on PC/104. That resulting IEEE "P996.1" draft standard, which conforms closely to the current PC/104 Specification, is now nearing final IEEE approval.

WHAT'S IN THE PC/104 STANDARD?

The differences between PC/104 and the "normal" PC are mainly mechanical. There are no software differences. Here is a summary of what is contained within the PC/104 specification:

Miniature Form-factor

Instead of the usual PC or PC/AT expansion card form-factor (12.5" x 4.8") each PC/104 module's size is 3.550 by 3.775 inches. There are two bus formats, for 8- and 16-bit modules. However, unlike the 8- and 16-bit expansion cards of desktop PCs, both forms of PC/104 are the same size. Figure 2 shows the basic mechanical dimensions of the 16-bit PC/104 module format. To make an 8-bit module, just omit the P2/J2 bus connector. (It's also permissible to include the P2/J2 bus connector on 8-bit modules, to allow stacking them between 16-bit modules.)

Self-stacking Bus

To reduce the complexity, cost, and bulk of conventional motherboards, backplanes, and card cages, PC/104 provides a unique self-stacking ("stackthrough") bus connector. Multiple modules stack directly with each other (see Figure 3). Stacked modules are spaced 0.6 inches apart and are normally securely attached to each other by four metal or nylon standoffs.

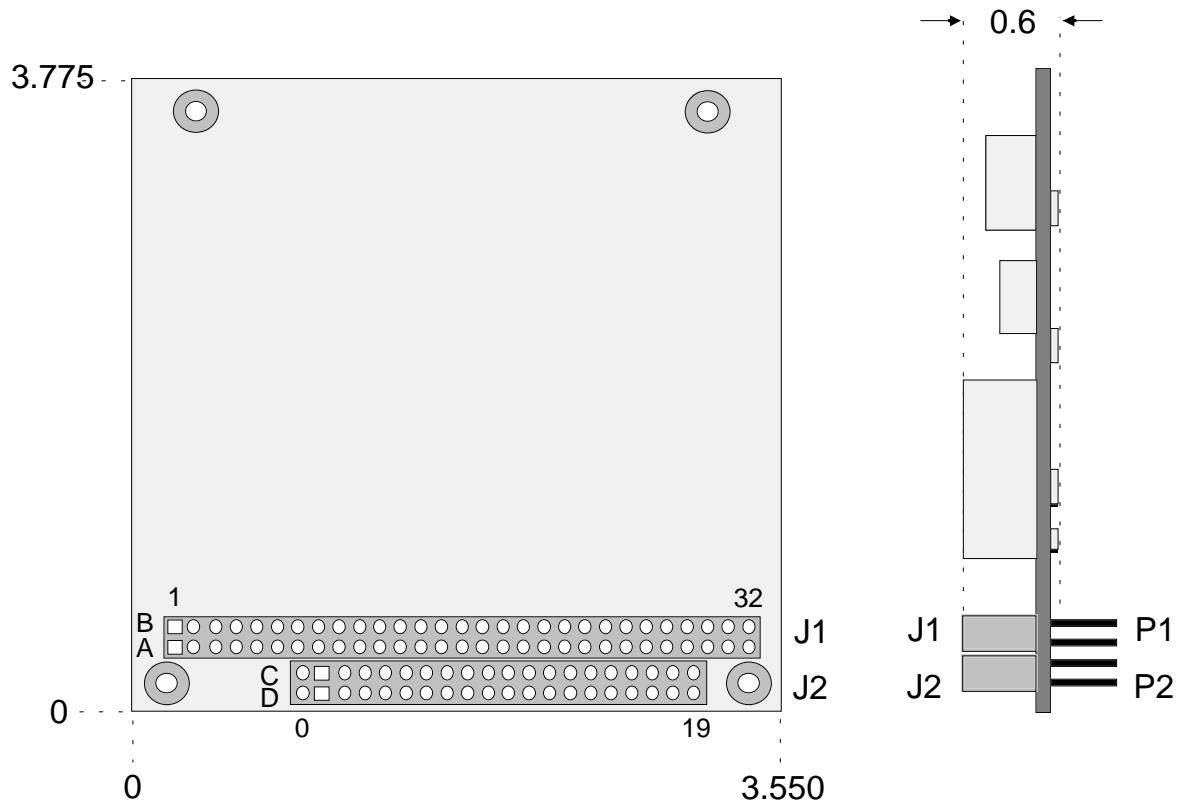


Figure 2. Basic mechanical dimensions, 16-bit PC/104 module.

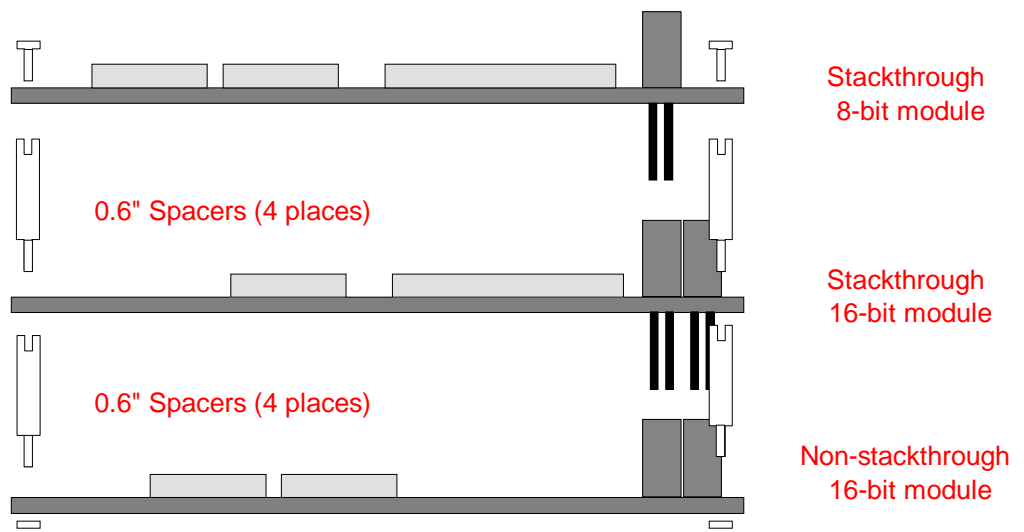


Figure 3. How PC/104 modules stack.

Pin-and-socket Connectors

Rugged and reliable 64- and 40-position male/female “header” connectors replace the standard PC's 62- and 36-position (P1 and P2) edgecard bus connectors. The PC/104 bus connectors feature two pin-and-socket rows on 0.1 inch centers and normally have gold-plated contacts. Two connector companies (Samtec and Astron) currently offer alternate sources for the required PC/104 stackthrough bus connectors.

Bus Signal Function And Pin Assignment

All PC/104 bus signal functions are identical to their counterparts on the “normal” PC/AT bus. Their assignments to the one hundred and four locations of the PC/104 bus connectors is given in the *PC/104 Specification*.

Reduced Bus Drive

To reduce power consumption (to around 1- 2W per module) and also to minimize chip count, bus drive was reduced from 24 mA (for the “normal” PC) to 4 mA. This permits “HCT” logic and many VLSI ICs to drive the bus directly, without additional buffer chips. An additional benefit of the reduced bus drive is that a PC/104-based system tends to generate reduced radiated emissions (EMI) relative to a normal PC bus.

A common question is, “How many modules can you put on a single PC/104 bus?” The reduced value of bus drive (4 mA) does not automatically imply a small number modules. Since the maximum input load spec is 0.4 mA per bus signal, the 4 mA bus drive can theoretically drive ten bus loads. However, factors like signal trace lengths and connector impedance transitions generally reduce the number of modules you can use reliably to between six and eight. The precise limit for a given system depends on total bus length, number of stacked connectors, environmental issues, and the characteristics of the specific PC/104 modules to be used. Also, don't forget to analyze the accumulated voltage drops on the “power” signals of the bus when multiple PC/104 modules are stacked.

Bus Termination Option

Most embedded systems with one to three PC/104 modules won't need bus termination. With more modules, or depending on the configuration of the system, you may want to terminate the PC/104 bus to ensure maximum signal integrity. If so, use the special method of “AC termination” recommended in the PC/104 specification. Traditional resistive termination (usually 220/330 ohms from each signal to power/ground) exceeds the available bus current. The recommended AC termination, which consists of a series R/C pair between each signal and ground, draws no static current and provides a good impedance match. If you aren't sure whether your system needs termination, provide a way to add it later, in case you decide it's required. Several “PC/104 terminators” are available, which implement the AC termination recommended in the PC/104 spec. These can be added at any PC/104 module stacking location. Or, you can include positions for tiny “SIP” termination networks directly on the PC/104 modules or PC/104-expandable boards you design.

Interrupt Sharing Option

When you use the PC architecture in embedded applications, you may find a shortage of bus interrupts. This is particularly the case when the system contains a lot of byte-oriented (8-bit) interfaces, such as serial ports, because the 8-bit subset of the PC bus contains just six interrupt lines, which are mostly dedicated to standard PC interfaces. Unfortunately, since the bus interrupt lines are active high, you can't "wire-OR" multiple interrupt requests on a single bus interrupt channel as is done on many other buses. Instead, PC/104 defines a mechanism by which multiple interrupting sources (on one or more modules) can share a single bus interrupt channel. The interrupt sharing technique defined by PC/104 requires a polling algorithm in the interrupt service routine. (The required polling function is included as an option in many software libraries, especially for serial communications and networking.)

HOW IS PC/104 USED IN REAL APPLICATIONS?

Although configuration and application possibilities for PC/104 modules are practically limitless, there are three basic ways the modules tend to be used in actual embedded systems.

Standalone Module Stacks

As shown in Figure 4, PC/104 modules can be used like ultra-compact bus boards, except they form compact stacks without requiring the usual backplanes and card cages. Such PC/104 module stacks can be “bolted” inside an embedded system’s enclosure, in an otherwise empty space. In this manner, an entire PC can often be *embedded* directly within a system that would otherwise require an external, *attached* PC for its operation.

To support this method of using PC/104 modules, several PC/104 vendors offer off-the-shelf PC/104 stack enclosures that can accommodate three to six PC/104 modules. PC/104 stacks in enclosures like these can be used as self-contained systems; or they can be used to package subsystems within larger systems. These PC/104 system/subsystem enclosures support a variety of system environments and applications, including commercial, industrial, and vehicular, and they are offered with options like PC/104 form-factor power supplies (for 8 to 80V AC or DC inputs), shock mounts, and quick release mechanisms.

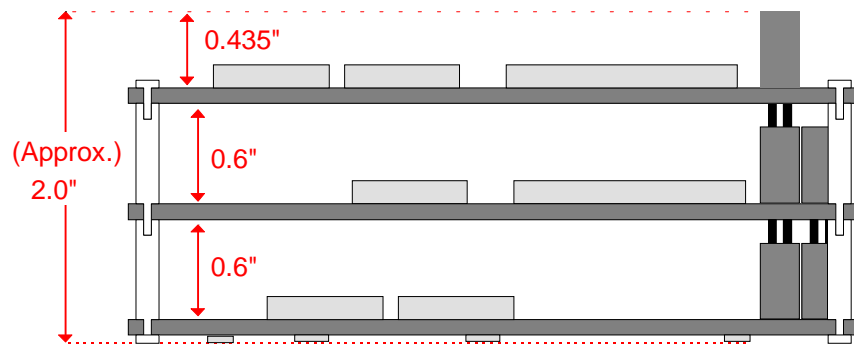


Figure 4. Using PC/104 modules in “standalone” stacks.

“Macrocomponent” Applications

You’re probably familiar with the image of a stack of several PC/104 modules, containing the equivalent functions of a complete desktop PC, in the palm of someone’s hand. Although this picture is visually appealing, most real PC/104-based system designs don’t actually use the modules this way. Increasingly, PC/104 modules are being used like multi-chip “macrocomponents”, plugged into custom, application-specific “baseboards”. This approach to using PC/104 is illustrated in Figure 5.

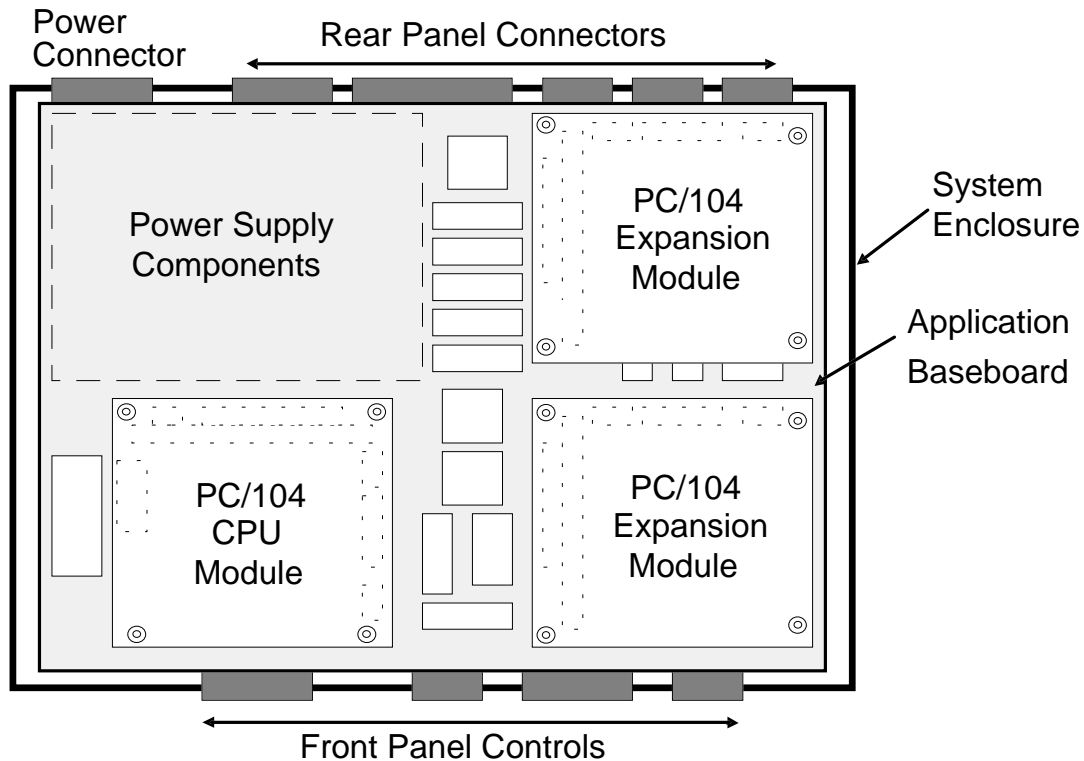


Figure 5. Using PC/104 “macrocomponents” on an application baseboard.

A typical application baseboard might contain all of the interfaces and logic that aren’t available on — or, for whatever reason, aren’t wanted on — the system’s PC/104 modules. These baseboards often include power supply components, signal conditioning, specialized interfaces, “real-world” I/O connectors, etc. Some devices on the baseboard may not even interface to the PC/104 bus, but may instead be included there for convenience (e.g. to eliminate additional electronic assemblies from the system).

This approach to using PC/104 modules is especially interesting, in that instead of plugging the application-specific I/O into an embedded computer, you plug the embedded computer (in the form of a PC/104 macrocomponent) into the application-specific I/O! This represents a paradigm shift in the design of embedded systems. By using this strategy, you are free to focus more of your time and energy on the application’s *unique* requirements — typically software,

application-specific electronics, and system packaging — rather than on reinventing the embedded-microcomputer. With this approach, the embedded electronics becomes, in effect, a *hybrid* of out-sourced modules (the PC/104 modules) and custom circuitry (the baseboard).

What size and shape should the application baseboard be? Since it typically takes the shape of the desired end *system*, it can be square, round, rectangular — or any shape! Often, an application baseboard provides multiple PC/104 stack locations. This can serve two purposes: it allows distributing the required PC/104 modules horizontally, producing a lower system profile; it also can provide a spare module location for future upgrades or expansion. It's usually a good idea to provide an extra 0.6 inches of vertical clearance (above the top PC/104 module), to allow use of the self-stacking PC/104 bus for unanticipated future requirements.

“Mezzanine Bus” Use

An increasingly popular use for PC/104 modules is as “daughter modules” on PC-compatible single-board computers (SBCs). Such an approach, illustrated in Figure 6, is called a PC/104 “mezzanine bus”. A PC/104 mezzanine bus is now included on nearly every new PC-compatible SBC, including both “standalone” (proprietary form-factor) SBCs and “passive backplane” (PC expansion card form-factor) industrial PCs. This is not surprising, since PC/104 was actually created by Ampro as a mezzanine bus for its Little Board SBC family.

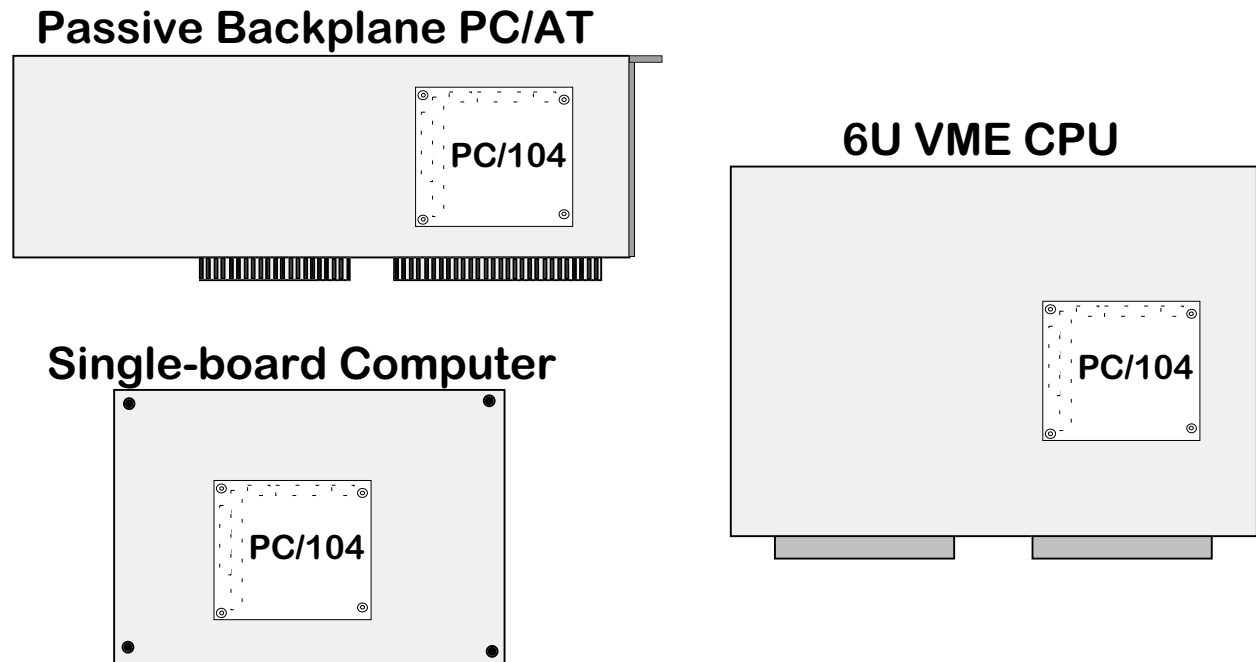


Figure 6. Using PC/104 as a “mezzanine bus”.

PC/104 AS “OBJECT-ORIENTED” HARDWARE

The use of PC/104 modules as “macrocomponent” building blocks, described above, is very much like the “object-oriented” software methods used by today’s programmers. In object-oriented software, a program is constructed of software modules (building blocks) which are independently specified, developed, tested, and maintained. Object-oriented *software* has become widely used because it reduces the risks and complexity of software development and accelerates project schedules; at the same time it also results in more powerful, feature-rich, and maintainable software.

Similar benefits are gained by using PC/104 CPU and I/O “macrocomponents” as *object-oriented hardware* building blocks. Projects are completed more quickly and with reduced expense — yet with enhanced features. Plus, the resulting systems are easier to maintain due their inherent modularity.

MAKING THE MOST OF OBJECT-ORIENTED HARDWARE

Here are some of the ways you can gain maximum benefit from an object-oriented, PC/104-based approach to designing embedded systems:

Treat the PC Architecture as a “Macrocomponent”

The entire embedded-PC *architecture* can be a single plug-in component, including all motherboard functions, system RAM, memory, and BIOS. You shouldn’t need to be concerned with licensing or modifying a PC BIOS. The PC/104 CPU module can even include a “solid state disk”, so you won’t have to worry about “ROMing” your embedded application code.

Use Variable Performance To Your Advantage

Embedded systems often end up needing higher CPU performance than originally planned. When this happens, you can simply unplug the PC/104 CPU module you’re using and replace it with a faster one. (Can you imagine doing this with an 8051, 68HC11, or a discrete 80386SX?)

Also, keep in mind the option of kick-starting a project by initially using a faster PC/104 CPU module than required, to get the application up and running quickly. Later, “cost reduce” by optimizing the software and substituting a slower (and less expensive) CPU module. To allow such flexibility, select PC/104 CPU modules that are members of a product family that spans a broad range of CPU types and performances.

Offer Performance And/Or Feature Options

How often have you been faced with the challenge of achieving a lengthy “wish list” of performance and features, but within highly constrained costs? With a modular, PC/104-based architecture, you can satisfy conflicting requirements by offering a spectrum of price/performance/feature alternatives. This way, a single PC/104-based design can achieve multiple cost/performance goals. You can provide 486 performance at the high end, and 8088

economy at the low end. Communications options can easily range from serial to Ethernet. Display options can run the gamut, from extremely low cost LEDs, to mid-range monochrome LCDs, to expensive TFT color LCD displays. You can thus configure the system in multiple ways, using alternative or additional PC/104 modules to produce the desired alternative feature sets.

Take Advantage Of Sophisticated PC Functions

In contrast to traditional microcontroller-based designs, your PC/104-based embedded system designs need not be limited by what you can accomplish by yourself. You can draw on a rich set of predeveloped hardware and software technologies that have been developed for the enormous PC user base.

Key technologies waiting to be incorporated in embedded applications include: powerful and user-friendly graphical user interfaces (GUIs); a broad range of mass storage devices, including floppy, IDE hard disks, SCSI drives, or PCMCIA cards; Flash memory storage, based on readily available flash-file-systems; full-function LANs, including Ethernet, Arcnet, and Token Ring, plus cost-effective RS485 multidrop. In addition, there are many PC/104 “real-world” interface modules, for digital and analog I/O, motion control, etc. — all supplied with ready-to-use DOS or Windows drivers.

Maximize Your Product’s Life Expectancy

A PC/104-based product can have a much longer life span than one based on the traditional “monolithic” design approach, for two reasons:

- ▶ **Module-based products can evolve.** PC/104-based products can evolve to meet changing market needs, to counter competition, or to incorporate newly available technologies. When a “monolithic” system no longer meets the needs of the market, you’re faced with a complete redesign. Unlike the traditional monolithic system, a PC/104-based system can be upgraded to a higher performance CPU, alternative peripheral interfaces, and enhanced software.
- ▶ **Modules insure products against chipset obsolescence.** When you design a product that will be produced for several years, you must seriously consider the risks of component obsolescence. What will happen when one of the chips in the system is no longer offered by its manufacturer? This issue is especially serious in systems that contain PC-compatible chipsets, due to the extremely rapid obsolescence rate of desktop PC chipsets. For planning purposes, you should assume a “half-life” (i.e., the time until you receive an end-of-life notice) for most PC chipsets of about three “Comdexes” — where one “Comdex” equals six months! This frenzied pace of PC chipset evolution is driven by the consumer-driven desktop market’s pressure to constantly introduce new models with incremental features.

With an object-oriented, PC/104-based architecture, you are “buffered” from having to constantly struggle with every IC’s pending obsolescence. When an IC is phased out by its supplier, you can expect your PC/104 module supplier to either incorporate a similar device into the existing module, or to create a new, but functionally equivalent, module. (Be sure you understand your module supplier’s policies in this regard!) If necessary, you also have the option to find an

alternate supplier who can provide the required PC/104 function. As you can see, a PC/104-based architecture protects you in multiple ways from the constant nightmare of having to do a product redesign each time an IC is discontinued.

KEEP YOUR OPTIONS OPEN

If you want to take full advantage of the *flexibility* PC/104 offers for future options, upgrades, and substitutions, follow this “golden rule” of object-oriented hardware design whenever possible:

Treat each PC/104 module as a “generic” function block.

This will ensure that some day, when you need to (and, inevitably, you will!), you’ll have a relatively easy job substituting equivalent new modules for the old ones. Here are some specific suggestions:

Avoid Vendor-Unique Functions In PC Chipsets

Unless a particular feature in a PC chipset is part of the PC “standard”, or at least part of a well-defined and multi-sourced superset, resist the temptation to use it! Only by building your application around “generic” PC functionality will you facilitate multi-sourcing of modules, high/low performance substitutions, and future backwards-compatible functional enhancements. In addition, try to keep the software CPU-independent. For example, avoid V20-specific instructions on an 8-bit CPU module, to retain the option of substituting *any* 8088-class PC/104 CPU module for a V20-based one..

Wrap Software Around Non-Standard Hardware

Despite your best efforts to keep things “generic”, there will probably be times when you need to use functions that aren’t part of the normal PC “standard”. In these cases, it’s important to keep a software layer between the application program and the nonstandard hardware, in the form of BIOS or device driver functions. This will give you the flexibility to alter the hardware in the future without having to rewrite the main body of application code, since differences in the hardware are masked by a software driver layer. In selecting PC/104 modules, look for ones that are supplied with BIOS or software drivers for any “non-standard” hardware functions. This will enhance your ability to maintain the system despite future module changes, whether required or desired.

Use Flexible I/O Cabling

Remember, the PC/104 standard confines its attention to the bus interface and the overall module dimensions. Although many I/O ports on the PC/104 modules you will use are functionally identical to those found in a standard PC (e.g. serial, printer, floppy, IDE, SCSI, VGA, etc.), the PC/104 standard does not define either their types or locations. Actually, since PC/104 modules are meant to be *embedded*, it often wouldn’t make sense to provide the same style of connectors that are located on the I/O bracket of standard PC expansion cards, since the latter are used for

direct external-world access. Furthermore, the interface connectors — including their mating connectors — must fit within the tight dimensions imposed by the PC/104 specification.

Therefore, flexible I/O “transition cables” are normally used to interface each PC/104 module with its intended peripheral devices or I/O connections. It’s best to consider the “object” to be *the module plus its I/O transition cables*. This will assure that the embedded systems you design will offer maximum module-independence. Later, when you change modules, you’ll probably be changing the module plus its transition cables.

DON’T FORGET THE “BOTTOM LINE”

Besides its many other benefits, an object-oriented PC/104-based product architecture helps you maximize your company’s profits and revenues. Here are two key reasons:

Shortened Time-To-Market

The use of off-the-shelf modules obviously shortens the product development cycle and therefore gets the product to market faster. This means realizing revenue sooner. Another important result is that “first of a kind” products normally achieve higher pricing margins than do late entries. The result of both these factors is increased near-term product revenue along with increased profitability.

Lengthened Time-In-Market

Mature products often contribute significant sales revenues while requiring minimal support. But when the lifespan of a product lives is shortened by changing market conditions, competition, or component obsolescence, this important “cash cow” phase of the product’s life cycle is lost. Since a PC/104-based product can evolve to adapt to these threats, its life-cycle can be extended to provide substantial incremental revenues without significant additional engineering costs.

CONCLUSION

PC/104 embedded-PC modules offer a highly efficient “building-block” approach to designing embedded systems based on the popular and “user friendly” IBM PC architecture. With more than 140 vendors currently offering off-the-shelf PC/104 modules, and additional hardware and software vendors announcing PC/104 products nearly every week, you can expect to see PC/104 modules used in an increasing number of embedded systems for a long time to come.

Consider using an “object-oriented” PC/104-based architecture in your next embedded system project, instead of “reinventing the wheel” with the usual microcontroller approach. The benefits include a quicker development cycle, reduced development tool costs, a more maintainable product, and increased corporate revenues and profits.