CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

SILICON UPDATE ONLINE

Tom Cantrell

16-Bits or Bust

If 8-bit chips are the MCU compact pickup truck and 32-bit chips are the Corvettes, what does that make the 16-bit chips? This month, Tom does some homework and hopes to find out exactly where 16-bit chips fit in the MCU market.

I must say, when it comes to dissecting the comings and goings in the MCU market, I've always had a bit of trouble figuring out how 16-bit chips fit in. If 8-bit chips are the MCU compact pickup and 32-bit chips are the Corvettes, what's that make a 16-bit chip? Is it something that would only appeal to a teenager whose lust for the souped-up is bigger than his budget?

In trying to come up with an answer, part of my problem is simply figuring out what a 16-bit MCU is. Are we talking ALU-width, data bus pins, or programming model? Taking the automotive analogy all the way under the hood, you could think of it as a motor with 16 valves. But, am I talking about a compact pickup's four-valves-per-cylinder four-cylinder engine or a Corvette's two-valves-per-cylinder V8?

The issue has proved a headscratcher for ranks of micro marketers and an opportunity for some creative pitching. Even as I peck away, I notice on the wall a late-70s Intel 8088 ad reprint, which crowned that chip the 8-bit Champion over the Zilog Z80 and Motorola 6809. Of course, nowhere does it mention that this particular 8-bit chip, unlike those it planned to vanquish, happened to have a 16-bit ALU.

UNLIKELY SOURCE

I gave old pal and MCU-numbers guru Tom Starnes of Dataquest a call to get an update. Turns out he's been struggling with the whole issue of terminology as well (see the sidebar "Is a 32-bit MCU really an MPU?"). After a fun discussion of that subject, I got down to business—what's the story with 16 bits?

Bet you didn't know that something on the order of 500 million 16-bit MCUs were shipped in 1998! That's a lot of units, many more than 32-bit chips and practically one quarter of the everyone-knows-it's-huge 8-bit market. Which raises the question: if 16-bit MCUs are such a big deal, how come they're so stealthy when it comes to showing up on designer's radar?

A close look at Tom's numbers reveals one of the answers: 16-bit MCUs are somewhat of a Japanese phenomenon. For instance, four of the top five 16-bit MCU suppliers are Japanese. And, if you look at demand as well as supply, you'll see the same geocentricity. In Japan, the yen volume is highest for 16-bit MCUs, even though it represents a much smaller fraction in all other markets.

Perhaps it's because the U.S., having kicked off the MCU craze with the original 8-bit '51s, '68s, and PICs got more locked in to 8-bits, while Japan's later start left more of an opening for

Photo 1—*Though wrapped in an econo-box 56-pin package, the M16C/20 has a 16-bit engine under the hood.*

16-bit chips. I'm reminded of the way developing countries often go straight to wireless for their telecommunications infrastructure, leapfrogging past the need to run a bunch of wire like those who started sooner. Or maybe, it's the fact that 8-bit ASCII isn't enough to deal with the thousands of Kanji characters. And, perhaps, you could attribute some of the difference to the rather convoluted fits and starts of the 16-bit efforts by U.S. suppliers.

So who's number one in 16-bit MCUs? It may come as a surprise that Mitsubishi wins the title with 18% of the market, besting Motorola, the overall MCU leader, who stands at 14% and way ahead of the better-known Japanese IC suppliers NEC, Hitachi, and Fujitsu, who all hover around 10%.

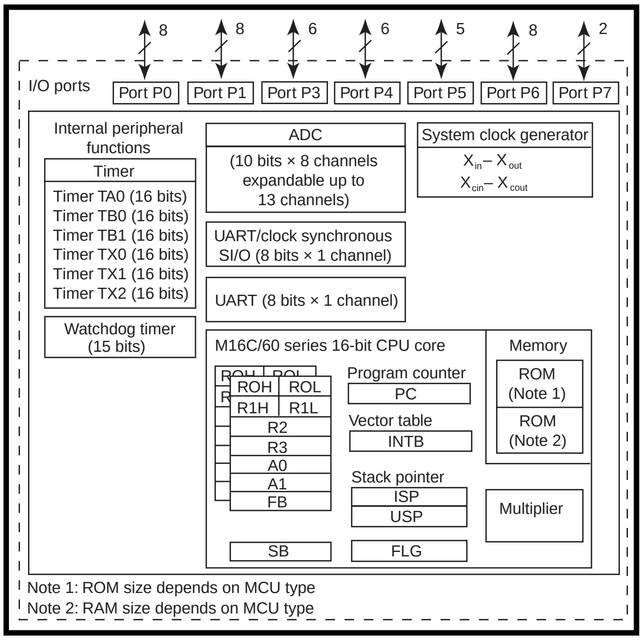


Figure 1—*Bits shmitz! It's the copious memory (up to 48 KB of ROM or flash memory and 2 KB of RAM) and functional peripherals onboard, not the 16-bit ALU, that makes the M16C/20 such a good fit for mainstream embedded designs.*

UNDER THE HOOD

The M16C family consists of a variety of parts based on a common architecture. Though the marketing collateral naturally claims the chips are RISC-like, in fact they're quite CISCy with 91 semantically rich, variable-length instructions and a dozen-plus addressing modes. Most of the instructions execute in three cycles or less and even MULtiply takes only five cycles.

As odd as it may seem, the architecture strikes me as kind of a cross between a 68k and a Z80. For instance, like the 68, the eight 16-bit registers are partitioned into data and address roles. And like the Z80, there are two banks of register that are software selectable.

Mini-pickup with a Corvette motor? Actually, I think it sounds weirder than it is. In practice, the CISCy nature yields real benefits in terms of code density and, if I may be so blasphemous, ease of assembly-language programming.

The lineup covers a broad range. At the high-end, the M16C/6x and M16C/8x incorporate plenty of memory (the M16C/62 features a whopping 256 KB of ROM or flash memory and 20 KB of RAM), high pin count (e.g., 100 pin) packages with external buses for connecting lots of DRAM, and so on.

However, though impressive and quite capable, I think such high-end 16-bit chips are most vulnerable to the challenge of similarly priced low-end 32-bit chips from outfits like ARM, MIPS, Hitachi, and Motorola.

Although the M16C gives it a good try, chips with 16-bit programmer models invariably stumble up against the 64-KB barrier and, despite oh-so-clever segment and banking schemes, there's fundamentally no elegant way around it. If you're looking at huge programs and lots of pins to hang a bunch of DRAM on, then 32 bits is the way to go.

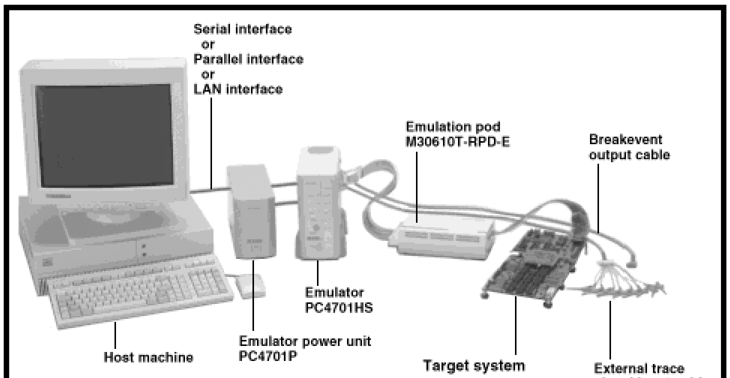
The brand new M16C/20 is another story (see Photo 1). Here's a chip that's a good fit with mainstream em-bedded apps and is even a practical alternative to 8-bit MCUs.

For instance, the '16C/20 is single-chip only. In other words, all the memory (you can get 32-KB ROM/flash + 1-KB RAM or 48-KB ROM/flash + 2-KB RAM versions) is on-chip. Without the 20–30+ pins required for external expansion, the chip fits in a tiny 56-pin QFP package and, surprise, even a DIP package—albeit, a rather off-the-wall 52-pin SDIP.

I suspect the popularity of the M16C family may have less to do with the ar-chitectural whizziness of the CPU core and more to do with the stuff surrounding it. Take a look at Figure 1. You can see the '16C/20's huge selection of peripherals, including six 16-bit timers (plus a watchdog), two UARTs (one also handles clock serial mode) and an 8-channel (up to 13 channels by sacrificing some shared pins) 10-bit ADC.

In case you were wondering, these peripherals aren't stripped down models by any means. For instance, the timers (see Figure 2) feature a myriad of modes, including event counter, one-shot, pulse-width measurement, and PWM. There are plenty of internal and external clock-source options individually programmable for each timer and a half a dozen pins to work with as well. The UARTs have their own transfer-rate generators, allowing completely independent operation without using any of the six timers.

The A/D includes a variety of repeat-sweep modes that cycle automatically through a programmer-defined list of channels. Unlike similar,



Serial interface or Parallel interface or LAN interface
Emulation pod M30610T-RPD-E
Breakevent output cable
Emulator PC4701HS
Emulator power unit PC4701P
Host machine
Target system
External trace signal input cable

Figure 3—*The Mitsubishi emulator may be a bit more than you need or can afford. Fortunately, they're working to expand the tool offerings with popular third-party suppliers, like HP, Orion, and Nohau.*

but simpler, schemes found on other chips, the ’16C/20 can emphasize (i.e., convert more frequently) a particular channel. For instance, the sweep might pay more attention to channel 0 with a sequence like ch.0, ch.1, ch.0, ch.2, ch.0, ch.3, and so on.

Even the parallel I/O is gussied up with bit-by-bit direction setting, nibble-by-nibble pull-up selection, and eight pins offer programmable drive up to 30 mA for directly connecting to LEDs or other high current loads or whatever else might come in handy.

The ’16C/20 also pays attention to details important to real-world embedded apps. For instance, there are two separate os-cillators. One delivers the main clock (up to 10 MHz) that’s used during normal operation and includes a programmable divider chain (1,2,4,8,16), while the other runs off a 32-kHz watch crystal. The neat thing is that you can switch CPU operation to either one via software and disable the unused oscillator to cut power.

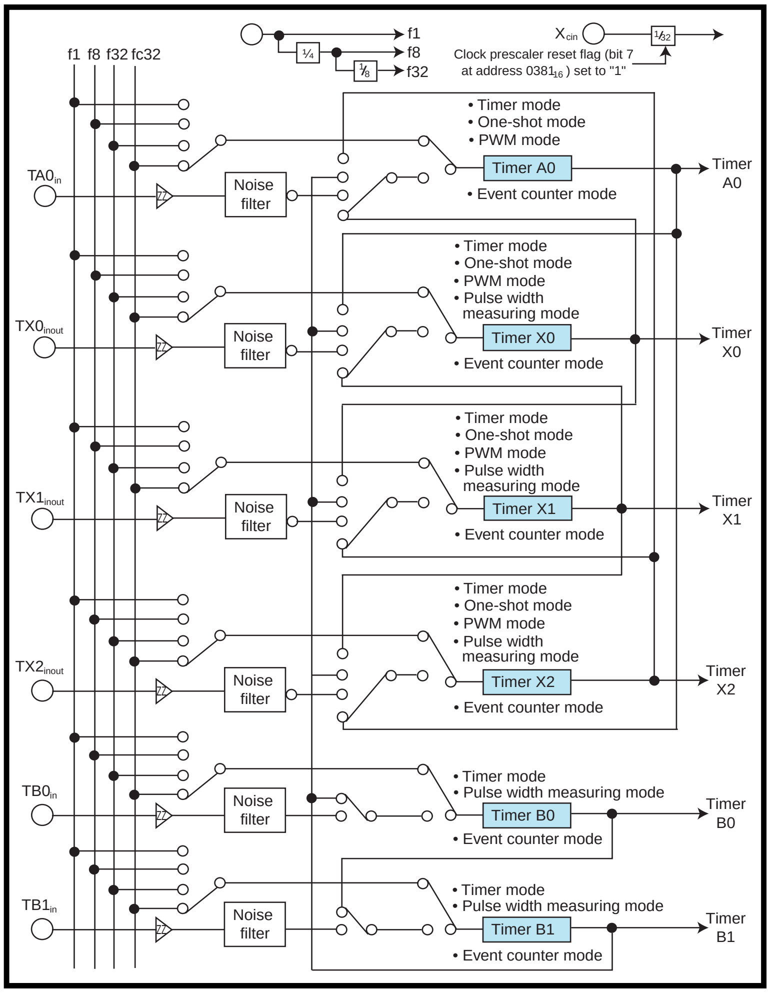


Figure 2—*M16C/20 offers a competitive array of timers—far more than the typical 8-bit MCU—six channels, six pins, and a bunch of modes. Notice the flexible triggering options, including the chip’s dual clocks (Xin and XCin).*

For instance, running off the 32-kHz clock and shutting the main clock down, the chip typically consumes as little as 4.0 mA, which is competitive with a typical MCU all-clocks-off Sleep mode. Both flash memory and ROM versions run at the full 10-MHz speed between a wide 4.0–5.5-V operating range, while ROM versions can go all the way down to 2.7 V, albeit at a reduced clock rate. RAM contents are retained all the way down to 2.0 V.

NIHON-NO MICON?

Frankly, there was a time when going with a Japanese MCU was a decision not to be taken lightly. It had nothing to do with technology but that Japan was far away, had a different language, and the business style was quite different.

Old-timers know what I mean. Indecipherable datasheets, over-engineered and over-expensive tools, disinterested reps who didn’t want to hear about anything but your next DRAM order, phone calls unreturned, unless you worked for IBM, HP, or GM.

These concerns were valid, and I’m in no position to say they’ve been totally eliminated. For instance, the official Mitsubishi emulator seems a rather arcane and expensive lash-up (see Figure 3) and third-party support, though growing, is nowhere near the level of a Motorola or Microchip.

On the other hand, Mitsubishi and the other Japanese suppliers have been trying to adapt. They’re working hard to improve the traditionally somewhat rocky relationship with distributors, recognizing that the channel is critical for a product like the ’16C/20 in the U.S. market.

Also, I’ve been pleasantly surprised with the quality and quantity of documentation. In addition to the usual datasheets, Mit-subishi offers a software application note that includes more than 50 useful routines, everything from the simple (block move) to the sublime (trig functions). Notably, each example religiously adheres to a rigorous format, in-cluding description, flow chart, register, and memory usage, restrictions, copious comments, and so on. The document is a good lesson for anybody who programs, not to mention all other MCU suppliers.

Bottom line: the ’16C/20 is an impressive chip and, after looking at its support and tools, I didn’t run across anything I would consider a show-stopper. Best of all, at only $6 (32-KB ROM) to $12 (48-KB flash memory), the price is definitely right. In fact, I seem to recall those clever 8088 marketers of yore also touted, “16-bit performance at 8-bit price,” a tag line that seems to match the ’16C/20.

Tom Cantrell has been working on chip, board and system design and marketing in Silicon Valley for more than ten years. You may reach him by e-mail at tom.cantrell@circuitcellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

SOURCE

M16C/xx 16-bit MCU
Mitsubishi Electronics America, Inc.
(408) 730-5900
Fax: (408) 730-4972
www.mitsubishichips.com