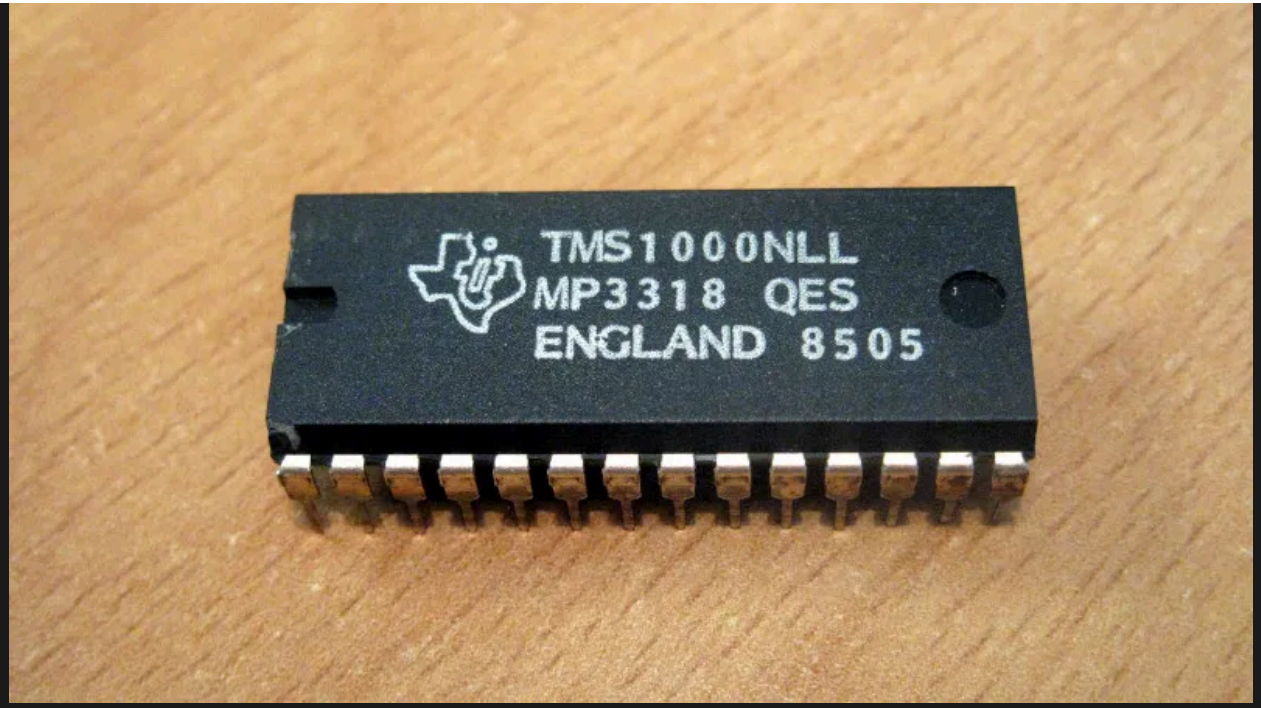
THE TMS1000: THE FIRST COMMERCIALLY AVAILABLE MICROCONTROLLER

by: Jenny List

81 Comments

February 18, 2020



We use a microcontroller without a second thought, in applications where once we might have resorted to a brace of 74 logic chips. But how many of us have spared a thought for how the microcontroller evolved? It's time to go back a few decades to look at the first commercially available microcontroller, the Texas Instruments TMS1000.

IMAGINE A WORLD WITHOUT MICROCONTROLLERS

It's fair to say that without microcontrollers, many of the projects we feature on Hackaday would never be made. Those of us who remember the days before widely available and easy-to-program microcontrollers will tell you that computer control of a small hardware project was certainly possible, but instead of dropping in a single chip it would have involved constructing

an entire computer system. I remember Z80 systems on stripboard, with the Z80 itself alongside an EPROM, RAM chips, 74-series decoder logic, and peripheral chips such as the 6402 UART or the 8255 I/O port. Flashing an LED or keeping an eye on a microswitch or two became a major undertaking in both construction and cost, so we'd only go to those lengths if the application really demanded it. This changed for me in the early 1990s when the first affordable microcontrollers with on-board EEPROM came to market, but by then these chips had already been with us for a couple of decades.

The Texas Instruments Speak & Spell from 1978 was a typical use for the TMS1000. FozzTexx (CC-SA 4.0)

It seems strange to modern ears, but for an engineer around 1970 a desktop calculator was a more exciting prospect than a desktop computer. Yet many of the first microcomputers were designed with calculators in mind, as was for example the Intel 4004. Calculator manufacturers each drove advances in processor silicon, and at Texas Instruments this led to the first all-in-one single-chip microcontrollers being developed in 1971 as pre-programmed CPUs designed to provide a calculator on a chip. It would take a few more years until 1974 before they produced the TMS1000, a single-

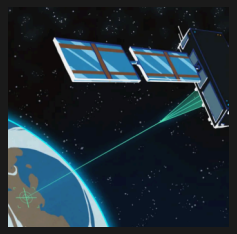
chip microcontroller intended for general purpose use, and the first such part to go on sale.

It's worth taking a moment to consider some of the terminology involved, because in 1974 the some of our current vocabulary was not necessarily in common usage. TI marketed the TMS1000 as a microcomputer because they saw it as an all-in-one computer without the need for extra peripherals. Today we'd consider a microcomputer to be an all-in-one general purpose computer such as the one you're probably reading this on, following an unbroken line stretching back to the Altair 8800 in the same year, but back then the vocabulary was like the technology; in its infancy. The word microcontroller was in use by then for a computer with self-contained I/O, the *Oxford English Dictionary* has a citation from 1971 in an IBM technical bulletin, but it seems not to have settled as the universal definition. By comparison the more modern "System on chip" or SoC refers to a general purpose all-in-one computer chip that presents its internal bus to the world rather than a set of I/O lines or peripherals as you would find in a microcontroller.

JUST HOW SIMPLE DOES A MICROCONTROLLER NEED TO BE?

THE VACUUM TUBE'S LAST STAND(S)

122 Comments



THERE'S MORE TO GLOBAL POSITIONING THAN JUST GPS

18 Comments

AI ON EVERY MACHINE: THE LLM YOU PROBABLY DIDN'T WANT

72 Comments

More from this category

CATEGORIES

Select Category

OUR COLUMNS

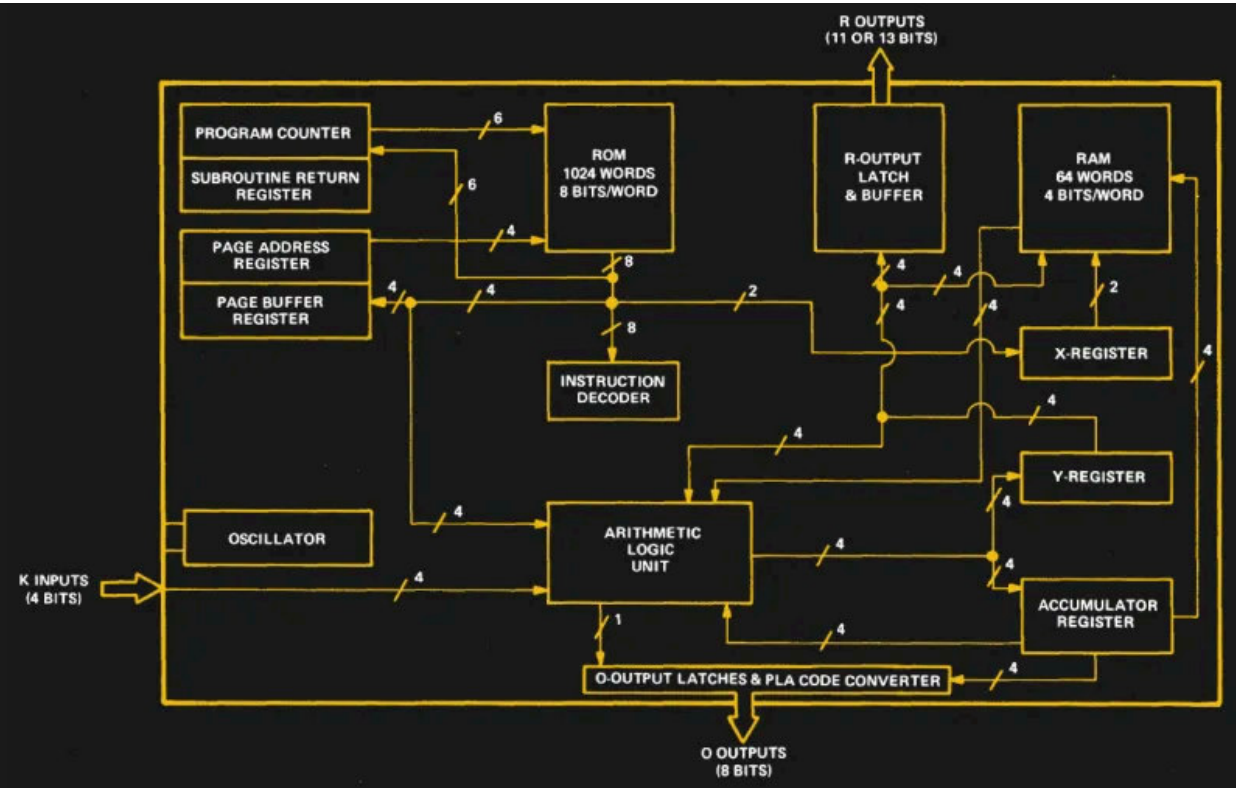


FIGURE 3 – TMS 1000/1200 LOGIC BLOCKS

The internal architecture of the TMS1000.

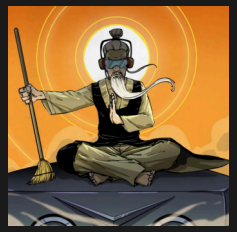
The TMS1000 then was the first commercially available microcontroller. But what kind of chip was it? There were four variants in the original range, all sharing the same 4-bit processor with a Harvard architecture, and sporting different numbers of I/O lines and ROM and RAM sizes. The TMS1000 and TMS1200 families had 8192 bits of program ROM and 265 bits of RAM, while the TMS1100 and 1300 families had double those figures. There were versions with high-voltage-tolerant outputs for driving vacuum fluorescent displays, and they were available in 28-pin and 40-pin packages. Internally it sports an extremely simple architecture by today's standards, without the banks of registers or pipelining you'd expect from more recent designs. It

ASK HACKADAY: DO WE NEED A 21ST CENTURY CALCULATOR?

77 Comments

HACKADAY LINKS: MAY 10, 2026

13 Comments



COPY OR REDESIGN?

5 Comments

HACKADAY PODCAST EPISODE 369: IR, E-INK, AND AVGAS

6 Comments

THIS WEEK IN SECURITY: ANOTHER LINUX EXPLOIT, UBUNTU KNOCKED OFFLINE, FINALS INTERRUPTED, AND BACKDOORED TOOLS

14 Comments

has none of the vast array of peripherals that you'd find on a modern microcontroller, but its I/O latches are supplemented by a simple programmable logic array. This would have been used as an encoder or decoder, an example in the data sheet uses it as a 7-segment display decoder.

You would not have found this chip for sale in single quantities for experimenters, because its on-board mask ROM could only be programmed at the point of manufacture by TI. Thus all coding was performed in a simulator on a time-sharing mainframe operated by TI. This would generate a deck of punch cards which after a very complex debugging and testing process would be used to generate the masks for ROM coding. The microcode could even be modified to order by TI, resulting in possible extensions to the device's 43 instructions. The mask-programming also means that any TMS1000 you find today will still contain whatever software it was manufactured with; without the original surrounding hardware for context they are of little use beyond a historical curiosity. This doesn't seem to stop some vendors attaching eye-watering prices to them,

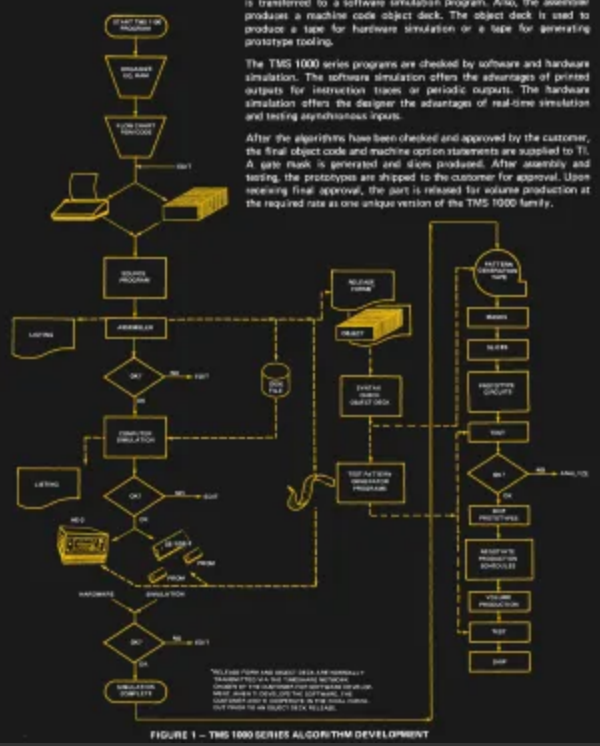


FIGURE 1 – TMS 1000 SERIES ALGORITHM DEVELOPMENT

The complex process of TMS1000 software development.

More from this category

RECENT COMMENTS

Dude on Mixapps: The Mixtape Of The Internet Age

Dude on Mixapps: The Mixtape Of The Internet Age

oozeBot Support on PreFlight Slicer Brings Added Part Strength Feature, And Many More

Sean on Restoring A 3DO Blaster Card From The Early 90s

Hunter Irving on Mixapps: The Mixtape Of The Internet Age

Hunter Irving on Mixapps: The Mixtape Of The Internet Age

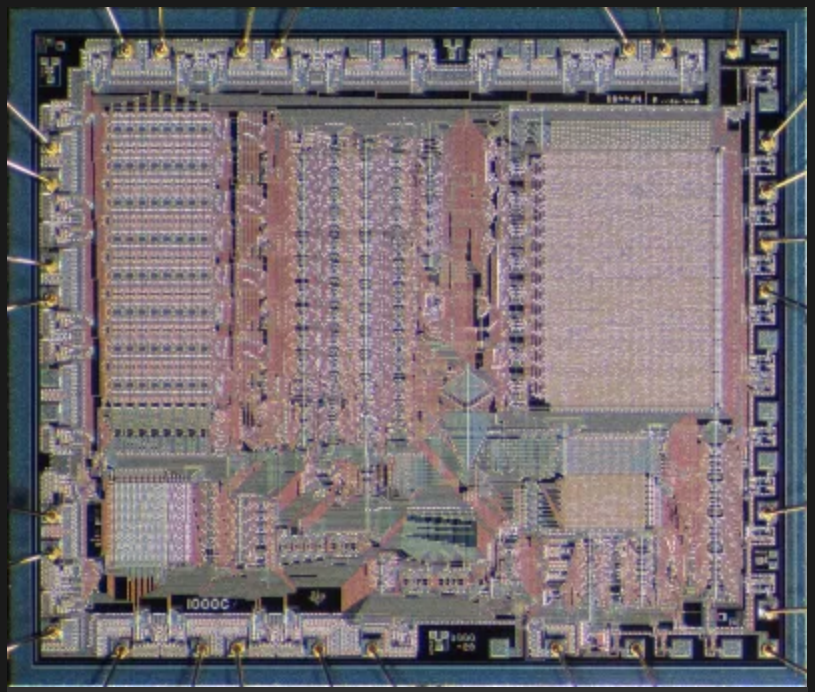
bitsquirrel on Mixapps: The Mixtape Of The Internet Age

Tomot on PreFlight Slicer Brings Added Part Strength Feature, And Many More

chaynz on Restoring A 3DO Blaster Card From The Early 90s

but thankfully if you really need a TMS1000 in your collection they can still be picked up for not a lot.

IF IT WAS EVERYWHERE IN THE 1970S, WHY IS IT NOWHERE NOW?



A die shot of a CMOS TMS1000. Pauli Rautakorpi [CC BY 3.0]

By the 1980s the world belonged to 8-bit and 16-bit microprocessors and microcontrollers, so other than surviving unseen for a few years as cores for TI's calculator chips, the TMS1000 series was eventually retired and has since slipped away unnoticed into electronics history. It's interesting to note that some of its contemporaries are very much still with us, you can still buy plenty of PIC, 8051, Z80, and even 6502 derivatives, yet there were no direct successors to the 4-bit TI processors. The onward march of technology is one culprit for this, but perhaps also the arcane software development played a part in its demise. Those 8-bit CPUs are still with us because anyone could pick up a dev board and an EPROM programmer and start coding, so they attracted a core of developers well-versed in their architectures. By comparison TMS1000 developers must have been few and far between, and

Tomot on PreFlight Slicer Brings Added Part Strength Feature, And Many More

certainly not enough to demand successor silicon. There were some special development versions of the processor added to the range that took an external ROM, but by then the market had shifted upwards by four bits.

If you encounter a TMS1000 today it's most likely that you'll have a late-70s electronic game in your hands, such as TI's own Speak & Spell, or Milton Bradley's Simon. Even these games survived longer than their original processor; you can still buy a modern version of Simon from Mattel, and TI's line of speech-enabled learning toys kept going into the 1990s. The legacy of this processor is immense though, and can be seen today in every electronic device containing a microcontroller. If you have one, it's a real piece of history!

Header image: Antonio Martí Campoy [CC BY-SA 4.0]

Posted in History, Microcontrollers, Parts, Slider

Tagged microcontroller, texas instruments, TMS1000

← OPEN-SOURCE NEUROSCIENCE HARDWARE HACK CHAT

FOXIE CLOCK WORKS IN TWO WAYS →

81 THOUGHTS ON "THE TMS1000: THE FIRST COMMERCIALLY AVAILABLE MICROCONTROLLER"

jackcrenshaw says:

February 18, 2020 at 10:57 am

Like millions of other folks in the digital world, I longed to build a computer for my very own use. Interestingly enough, I chose the TMS1000 (which choice only goes to prove how ignorant I was about building digital circuitry). For all the reasons Jenny mentioned, I didn't buy any hardware, but that didn't stop me from thinking about how I'd write software for it.

My first project was to write a floating-point arithmetic package. I actually got something that (I think) might have worked.

But then the Altair 8800 came along, and that changed everything.

Reply Report comment

Bernie Walker says:
February 18, 2020 at 12:06 pm

I think you will find that Intel 4004 beat the tms1000 by 3 years.

Reply Report comment

macsimski says:
February 18, 2020 at 1:27 pm

maybe because the 4004 was a processor, not a microcontroller?

Reply Report comment

Jenny List says:
February 18, 2020 at 1:40 pm

Indeed. Similar, but different.

Reply Report comment

Brian says:
February 18, 2020 at 1:38 pm

From Wikipedia:

“The Intel 4004 is a 4-bit central processing unit (CPU) released by Intel Corporation in 1971. It was the first commercially available microprocessor, and the first in a long line of Intel CPUs.”

“The TMS1000 is a family of microcontrollers introduced by Texas Instruments in 1974. It combined a 4-bit central processor unit, read-only memory (ROM), read/write memory (RAM), and input/output (I/O) lines as a complete “computer on a chip””

Seems you and Wikipedia agree.

Reply Report comment

Brian says:
February 18, 2020 at 1:40 pm

But wait…

https://en.wikichip.org/wiki/ti/tms1000

In September of 1971, TI finished the design for their TMS0100 single-chip calculators. Designs where done by the Texas Instruments engineers Gary Boone and Michael Cochran. Based on their design of the TMX1795, Gary patented the invention for a single-chip processing machine on Aug 31, 1971. On Sep 4 1973, he was awarded U.S. Patent 3,757,306. Building on top of their experiences with the TMS0100 and Boone’s 8-bit microprocessor prototype they went on to design the 4-bit TMS1000 microcontroller series. Boone was later awarded U.S. Patent 4,074,351 for the modern microcontroller.

After being slightly refined, the chip was released to general market in 1974.

Reply Report comment

Jenny List says:
February 18, 2020 at 1:42 pm

Wikipedia is absolutely right.

But if you take a look carefully at the wording, you'll see that one is a CPU and the other is a microcontroller. A microcontroller contains a CPU, but is a different component.

Reply Report comment

Brian Poole says:
February 20, 2020 at 12:41 am

The author mentions this CPU as well.

Reply Report comment

Chuckz says:
February 18, 2020 at 12:59 pm

And now kids are learning from the 32 bit Micro Bit.

Reply Report comment

Shannon says:
February 19, 2020 at 2:45 am

Yeah! It's pretty cool, right?

Reply Report comment

k says:
February 19, 2020 at 3:22 am

No they are not!
Unfortunately they are learning Arduino/ abstraction and have no clue what instruction pipeline is or assembly optimization is. They just grab a bunch of libraries cobble then together and scratch their head when something goes wrong during one specific execution scenario.
The abstraction is taking away the inner workings which sadly only a handful of kids put an effort to learn. But hey! all of them have "worked" with the same micro controllers!

Reply Report comment

CW says:
February 19, 2020 at 8:25 am

How dare people use something without knowing all the inner workings? … Really?

Reply Report comment

Dwight Day says:
February 19, 2020 at 12:04 pm

People who are more than casually interested in computers should have at least some idea of what the underlying hardware is like. Otherwise the programs they write will be pretty weird. (Donald Knuth)

Reply Report comment

Laurens says:
February 20, 2020 at 3:18 am

For people who do it for fun, it's just fine. For the professionals – they'll get deeper knowledge anyway. Don't complain about those gosh darn kids not doing it the old way anymore!
When the transistor arrived, the old folks complained about how they didn't know how to design tube circuits. And those who used tubes, were bashed by the spark gap crowd etc etc.

Reply Report comment

Martijn says:
February 20, 2020 at 4:21 am

Beware: Micro Bits are gateway microcontrollers.

Reply Report comment

JM says:
February 21, 2020 at 9:05 pm

I agree – look how much damage to the computing brought C64 with it's Basic! No kid had to learn about pipelines and assembly! Just type some command and scratch their heads why it's so slow? And than just to speed things up some of them peek and than poke and use those monitor programs and look where we are now? People using spreadsheets for their home budget instead of writing nice and efficient code in FORTRAN (or at least C)! Madness I tell you...

Reply Report comment

Greg Lőrincz says:
February 24, 2020 at 4:53 am

"The abstraction is taking away the inner workings which sadly only a handful of kids put an effort to learn." I'm a trained teacher and you have absolutely no idea of the inner working of people. It's a good thing that kids learn skills. Period. I despise this way of thinking that children should learn programming in assembly or punch cards BECAUSE THAT'S HOW WE USE TO DO IT. Old techies are some of the dumbest, most narrow-minded people I have ever come across.

Reply Report comment

NewCommentor1283 says:
February 25, 2020 at 8:45 pm

so you ADMIT that it is THE PEOPLE that are WRONG!

they should be wired to accept all programs that can work properly and without error, the first time around and without ANY guesses, glitches, or slowdowns.

this INCLUDES ASSEMBLER SOURCE-CODE CREATION software.

fix your “people” and turn them into computers, after-all; they sub-conciously learn analog spacial triginometery at age three just to walk; i think they can be forced to manage 64 commands and 64 registers.

if an old-fashioned rotary-dial telephone can pause a torrent download on windows then your kids can learn assembler.

Reply Report comment

eekee says:
April 28, 2020 at 5:43 am

I certainly like to “get down to the bare metal,” but having given that topic some serious consideration recently, I realise that programming in assembly language or even machine code (hand-

assembled) is *nothing like* getting down to the lowest level. There are layers upon layers of hardware abstractions below that, some of them hiding complex details. I'm talking about 70s/80s CPUs here, I don't even really understand late-80s stuff.

Libraries... I don't know. I love to write the code myself, but thinking practically, I think it's all right if the libraries are well-designed, well-written, and most of all well-documented. I know I'm going to end up "scratching my head" over instructions not working the way I expect when I get down to writing my Forth, so I can't logically see why having the same problem with libraries would be worse.

What makes more sense is the idea that many libraries aren't well written and/or have poor interfaces and worse documentation. That's true enough, especially in the open source freebie world. That's solved by teaching good algorithms and techniques, but teaching people to write low-level code seems to lead them to reinvent poor techniques. I'm saying this as a low-level coder; I see the disadvantages of what I've learned and *feel* the mental blocks it places on learning how to do things right.

I also *see* the mental blocks in others all over the place. So many low-level coders are just completely incapable of lifting their heads to see the big picture. Sometimes, it's frightening to see! Sometimes they can if educated, but many resist that education so hard. It's nice when someone doesn't, such as a very experienced, very good programmer my friend told me about, who insisted something was impossible until my friend showed him how. The big thing here is my friend was just a consultant; "a simple computer educator" in his own words.

Reply Report comment

Lucy says:
August 5, 2021 at 11:54 am

LOL, if people really wanted to learn how CPUs truly work they could study transistors and gates then learn how to assemble those into higher level components such as the ALU, instruction decoder, flip-flops for memory, etc. But for modern kids its pointless unless they are planning to be a low level EE or semiconductor designer.

One day I might still build one from scratch if I have plenty of time and money when I retire and I may even get it etched into silicon and diced into a full packaged ceramic chip. But truly, modern kids are better off sticking to higher level abstractions (learning Java, object orientated languages, functional programming, neural networks, and big data).

Reply Report comment

yash vardhan meena says:
August 5, 2025 at 4:03 am

i am 21 reading these comments made my mind hurt with interest to learning . i never knew this world also exist.... and after listening all of you . i see you guyss fighting over these stufff why young gen not also following or learning about chip,...low ..LL.. EE ..... we never found these ..... like u guys told me these exist i would never FOUND these on my own . these are cool stuff . which only we can like cause . maybe we are emotionaless and more logical people we cool ..... les go we unbreakable as CRT

Reply Report comment

andy4set says:
February 18, 2020 at 1:14 pm

Worked at a place in the early 80s that used tms9900 to make industrial controllers in applications like a laundry, a main signal box train describer, and an airport terminal flight display/baggage handler. You could get a lot out of a 16 bit microprocessor back then.

Reply Report comment

Philip S. Crosby says:
February 18, 2020 at 2:36 pm

The TMS9900, like the IBM 360 had its registers in memory. A task switch involved a BLWP (Branch and Load Workspace Pointer) and R0 .. R15 would have all new contents. I led a project at Tektronix to build the 1980, a video analyzer that was entirely programmable, The incoming video was digitized at 4X the color subcarrier rate and state machines controlled hardware functions in real time.
Later products with much faster processors followed, but much of the measurement algorithm development still occurred on the 1980 because it used an interactive form of BASIC (Tek Answer Basic).
Fun days.

Reply Report comment

Gregg Eshelman says:

February 18, 2020 at 7:45 pm

What would have been nice is if TI had continued production of the Home Computer's Peripheral Expansion Box as a box for building industrial control and other embedded stuff. The PEB was already built like a tank with a massively overbuilt, unregulated, linear power supply and a thick steel chassis.

The Expansion cards each had to have their own onboard power regulation to bring the voltages down to the normal 12V and 5V of the day. An "open" power source like the PEB would easily allow for hardware using any voltage, even quite a bit over the typical 12V and 5V.

The backplane was a simple, dumb, bus. All lines connected straight through, with pin definitions determined by the interface card connecting to the computer console. That would've given a PEB system designer essentially complete freedom for the bus design, to make cards compatible with another system, or deliberately incompatible.

TI could have switched to a lighter, regulated power supply for the PEB and designed a 9900 CPU card and other peripherals to have a successor or companion to their 960, 980, and 990 mini computers.

Reply Report comment

Ostracus says:
February 18, 2020 at 4:08 pm

Games too if memory serves.

Reply Report comment

Ian King says:
February 20, 2020 at 1:17 am

I used the SBP9900, in a military project in the 1980's, this was an integrated injection logic version of the TMS9900 and they cost a fortune! The development kit filled a room! I recently saw the prototype of the unit in an air museum, aah, happy days.

Reply Report comment

gregg4 says:
February 18, 2020 at 2:04 pm

Let us not forget the COP402 and its relatives. They were 4 bit parts who found themselves into a lot of things. Not surprisingly I have a batch of them, programmed of course, and the COP420 part who is wearing the name of the company it was made for. It has the part number on its back.

Reply Report comment

RW ver 0.0.1 says:
February 18, 2020 at 3:19 pm

What are they programmed for? You could swap one into your coffeemaker and see if it makes toast or washes the dishes instead.

Reply Report comment

gregg4 says:
February 18, 2020 at 7:42 pm

Well the two I have are the COP411 and the COP420. And the COP420 for a 4bit part is slightly more complex and almost too difficult for a hobbyist to use. The technology salvage place was completely clueless as to what they were.

Reply Report comment

Peter says:
February 18, 2020 at 2:20 pm

I remember the days when we had to design a “TTL Grave” to get things done. Not to mention the manufacturing of standard PCBs that was totally out of reach of the hobbyist. Perfboards and a Verowire Pen were the tools of the time.

You kids can’t imagine how much the more senior crowd appreciate the immense capabilities we have at hand today. You could hardly dream this up in

the eighties.

Reply Report comment

Jenny List says:
February 18, 2020 at 4:27 pm

It's nice that someone thinks I'm part of the younger crowd. :)

Done all of those things. And laid PCBs by hand with crepe paper tape on acetate.

Reply Report comment

Mog says:
February 18, 2020 at 3:25 pm

The TMS1000 is also emulated well in MAME, which emulates the majority of TI Speak and Spell, Speak and Math, etc. titles, as well as a whole host of games which used the TMS1000, such as Simon.

Reply Report comment

jawnhenry says:
February 18, 2020 at 4:59 pm

Stop, already, with the stupid, infantile word games.

This horse was beaten to death when people argued about the fact that the original IBM PC was not a computer, but a ***micro***computer. Do any of you remember *that* particular piece of tech-wordplay, "I'm-smarter-than-you-are", insanity? Well, that's exactly what this is.

All of everyones' "appeals to authority" (wikipedia; a particular manufacturer' nomenclature; data sheets; etc.) in order to justify what you consider to be the correct terminology all boils down to *what some INDIVIDUAL* ***THOUGHT*** sounded "...about right...". Absolutely nothing more.

To use a characterization we used to use in the semiconductor industry–I don't care if it's "...a piece of hammered horse-shit...",
if it can be programmed, it's a computer: mini-, micro-, peta-; CPU; you name it, but don't expect me to be impressed by your erudition in claiming that something is NOT a micro***computer*** because it's a–harUMPHHH–micro***CONTROLLER***.

Pure, simple unadulterated obfuscation and posturing–completely unnecessary, and absolutely, one-hundred-percent confusing to the very people who look to this venue for clarification, elucidation, and education.

Clarification, elucidation, and education this ain't.

Reply Report comment

daveboltman says:
February 19, 2020 at 3:56 am

I think the distinction between XXXController and XXXProcessor is clear enough, and widely enough accepted – An XXXController has built in peripherals e.g. general purpose I/O pins, counter, timers etc, whereas an XXXProcessor does not (including the "Central" type of Processing Unit).

No word games – every XXXController has a built in XXXProcessor, but not the other way round. An XXXController can form the heart of an XXXWave Oven, DVD player, etc. etc. _without_ any additional address bus decoding logic, data bus latches etc. exactly because of the built in GPIO pins, timers etc. An XXXProcessor cannot.

Reply Report comment

MvK says:
February 19, 2020 at 4:16 pm

There has been a semantic shift on many EE terms. For example, a 6502 wouldn't be called a microcontroller today. But on introduction it was described and intended as such. The term didn't imply "integrated program memory". Only later a distinction was made. You see similar, sometimes subtle, shifts in terms as "TTL", "RISC", "PC", "emulator".

Reply Report comment

Michael Black says:
February 19, 2020 at 5:21 pm

Yes. I was going to say something.

The history is well known, the Intel 4004 designed to be used in a calculator, the 8008 intended for use in a terminal. I had a National book about using one of the early microprocessors to replace logic.

I don't know about industry, but hobbyists looked at them and saw "computer" rather than controller. All of the

microprocessor companies sold single board computers, which was mostly to introduce people to tye microprocessor, and thus sell the ICs, just incidentally giving us simple computers.

In parallel with the home computers were lots of “embedded controllers” . Thet weren’t general purpose computers, but dedicated to controller. Some surely used the tms1000, some used microprocessors that included ram and some I/O, as they became available.

But others just used 8080s and 6502s and whatever. In 1979 or early 1980 I knew someone who got a contract to update a local record pressing plant, and he used the intersil 6100, which was compatible with the PDP-8. Maybe because it was CMOS, but likely because of the PDP-8 angle. I was never sure if he hired someone to write software first, or chose the cpu first.

So a lot saw “microcontroller” use, even if they weren’t as dense as what came later.

Reply Report comment

Ren says:

February 20, 2020 at 11:12 am

” I was never sure if he hired someone to write software first, or chose the cpu first.”

When I worked in aviation weather research (1990’s) the FAA was building a huge ($1+Bn) Air Traffic control system using 1980’s hardware. (80286 procs and the

like).
Our group was one of those tasked with writing software for part of the system.

My boss (among others) were trying to convince the FAA to design/write the software they needed _FIRST_, and once that was ready, go and buy modern equipment that would run it.
For such a humongous system, the software took multiple years to write, while the hardware was quickly getting outdated.

IIRC, it was the “FAA Modernization Program”, and it was a $1Bn project that was already $1Bn over budget and years behind schedule before DOT Secretary Federico Pen~a shut it down.
(IMO, it was the only worthwhile thing he accomplished.)

Report comment

Jenny List says:
February 20, 2020 at 12:35 am

I was idly thinking I’d like to filter comments that start with “I think you’ll find” :)

In this context I think it’s an important distinction to make, because as someone who’s made control computers using Z80s etc as well as PICs,

Atmels etc there is a HUGE difference in the task involved. They're different components, even if they can do a similar job.

My past isn't in the semiconductor industry, despite being an electronic engineer and lifelong hardware hacker I've spent most of my career in the more techie side of the publishing industry. The job I did before going freelance and then working for Hackaday was by far one of the coolest places I've worked because of the breadth of knowledge among my colleagues, the Oxford Dictionaries. There I encountered people for whom the history of language is as much a passion as tech is for us, and I gained an appreciation for the history of language when followed as a science. Everyone has their pet theories and hills they're prepared to die on in language, but what sets lexicographers apart is their willingness to work only in the light of evidence gathered by painstaking research. Thus mentioning an OED citation is entirely appropriate when giving a potted history of a word, which I think was worth devoting a paragraph to in order to set the scene.

Of far more interest is the device itself, which I hope everyone will agree is a somewhat unsung seminal component in the history of semiconductors.

Reply Report comment

jackcrenshaw says:
February 20, 2020 at 10:57 am

Hear, hear. Reminds me of the guys that insist that centrifugal force doesn't exist. There is only centripetal force. Bah. It may impress folks at cocktail parties, but it's 100% useless information.

Reply Report comment

Ed says:
November 21, 2020 at 9:58 am

I had that same physics teacher in the 80s. 🤣

Reply Report comment

just sayin says:
February 18, 2020 at 5:10 pm

So .. why doesn't some enterprising soul craft an emulator? Sure, in 197x you'd have to have your code masked into ROM (oh gawd that's practically as cumbersome as an ASIC), but in an emulator, that wouldn't be such a problem now, would it. Then we could devise our own alternate pasts ..

Reply Report comment

Gregg Eshelman says:
February 18, 2020 at 8:05 pm

Might be able to get all the Speak and Spell sound samples here https://www.speaknspell.co.uk/ How about a TMS1000 emulation in Minecraft? https://www.minecraftforum.net/forums/minecraft-java-edition/redstone-discussion-and/337920-tms1000-ti-cpu-emulator

Reply Report comment

Elliot Williams says:
February 19, 2020 at 2:12 am

There's one in MAME.

Reply Report comment

greenbit says:
February 18, 2020 at 5:23 pm

Kind of a tangent, has anyone ever seen a logic diagram for the insides of a TMC0981, the heart of the TI-30 of yore? I guess because TI still sells a '30, maybe they try to keep a lid on that, but surely they're not still putting 1970s chips in the fresh crop of 30s!

I've always been curious how the 30 worked on the inside. That and the TI-57. I understood gates and flip flops well enough that when I got hold of a 7400 data book, it became self evident how you could paste blocks of registers and buffers and what not together and produce something like a 6502 or 8080 – but I never had that same kind of ah-ha moment for the calculators. So it's for academic reasons, really, that I'd like to know how they did it. Haven't had any luck goggling it, hence my appeal for clue here

Reply Report comment

Claus Buchholz says:
February 18, 2020 at 6:32 pm

I wrote a couple articles on the TI-57 inner workings here:
http://www.rskey.org/CMS/the-library

They are based on an emulator which is based on TI patents, found here:
http://www.hrastprogrammer.com/ti57e/index.htm

Reply Report comment

RW ver 0.0.1 says:
February 18, 2020 at 7:23 pm

Here's the die...
https://seanriddle.com/ti30fulldie.jpg

Reply Report comment

profumple says:
February 18, 2020 at 5:38 pm

BigTrak was cooler than Simon by far. Wonderful battery sucking toy recently re-created by Zeon. Doubting that TI takes paper tape roll programs and would have to brush off the dies; probably doesnt have original MCU. Magnetic coupled differential was also very cool. Hack worthy toy.

Reply Report comment

Jenny List says:
February 19, 2020 at 12:55 am

I'd forgotten the MB BigTrak when I wrote this. Makes sense it might have carried a TMS1000.

Reply Report comment

profumple says:
February 21, 2020 at 2:17 pm

There's no might to it. Original Big Trak hasTMS1000. Theres a teardown in ur own backyard. There were several revisions but all had same MCU. The Stereo phono jack 'trailer hitch' looks alot like TI cheapo serial port doesnt it? Still havent got the new one.

Reply Report comment

Andy in Indy says:
February 18, 2020 at 6:43 pm

I have not thought about that microcontroller in years. I learned about it because the World of Wonder Lazer Tag from the mid 1980's used that chip for the StarLyte and StarSensor.

Reply Report comment

commander_cook says:
February 18, 2020 at 7:51 pm

I happen to have a Speak and Math sitting at my workbench awaiting repair.

Reply Report comment

Smartroad says:
February 18, 2020 at 11:11 pm

“after a very complex debugging and testing process” ahhh those were the days when things weren’t able to be day one patched LOL ;)

Reply Report comment

Jim says:
February 19, 2020 at 2:39 am

It all happened in Scotland first….. If you look at the patent info regarding single chip microprocessors on
http://xnumber.com/xnumber/microprocessor_history.htm
A side-note I used to work at Pico in the 80’s and loved the job.

Reply Report comment

Alan Hightower says:
February 19, 2020 at 8:10 am

The Science Fair Microcomputer Trainer – available at Radio Shack – has a TMS1100. My first introduction...

Reply Report comment

Rick Presley says:
February 20, 2020 at 12:30 pm

What's a Radio Shack? :)

Reply Report comment

Michael Black says:
February 21, 2020 at 5:32 pm

It's the place on the deck of a ship where the radio equipment is kept and used. Originally it was a retrofit, since radio first went into existing ships. Later, they became not a shack, but incorporated into a new design Of course by then the equipment wasn't spark gaps.

There is recent talk about sending a sub down to the Titanic, remove the top of the radio shack, and extract any equipment. I suspect the radio shack is chosen because it's accessible.

Reply Report comment

Commodore Z says:

February 22, 2020 at 1:08 am

Me too, and what a great platform the SFMT was. Sadly my original one is damaged now, but it was fun while it lasted.

Reply Report comment

jawnhenry says:

February 20, 2020 at 7:31 am

I'm curious–

Let's consider Hackaday's favorite "computer": the Raspberry Pi–

1) Does Hackaday consider the Raspberry Pi 4 a computer, a microprocessor, a microcontroller, some in-between chimera? What, precisely?

2) Does Hackaday consider the BCM2711–used in the RPi4–to be a processor, a CPU, a computer, a microcomputer, a microcontroller? What, precisely, does Hackaday consider the BCM2711 SoC to be? What do *you* consider it to be?

3) If the BCM2711 is truly a microprocessor / microcontroller / CPU / full-blown processor, why can it not be programmed in assembly language (if it COULD be, one wouldn't be limited to bit-banging at a pathetically low rate), as can be any OTHER microprocessor / microcontroller / CPU / processor / mainframe? (If you even want to dare answer this question, then tell ALL of us where we might get a low-cost, or free, Assembler for the device).

Careful, now: *personal opinions* absolutely won't do–including completely wrong statements as to your ability to program the RPi in native assembly language. Only logical, technical, objective answers based on sound electrical engineering principles and precedent are allowed.

Reply Report comment

Greenaum says:
February 21, 2020 at 4:37 pm

YOU might not know how to program the Broadcom chip in assembly. I don't either! But the people who wrote the compiler surely do. Is the C compiler open-source? Perhaps there's some files there, or nearby, that tell a little bit about the workings. Or perhaps you could write a short program in C, then disassemble it. We know it's basically ARM, right? Try disassembling the drivers for some of /dev.

Yeah there's lots to bitch about the Ras Pi from an open source point on view. The Pi itself is a computer, the CPU is a SoC, and whatever HAD or you and me, or anyone else who isn't slinging millions of dollars around, thinks, doesn't matter.

Reply Report comment

jackcrenshaw says:
February 23, 2020 at 9:12 am

The problem with the RPi is not inept programmers or a broken compiler; it's about the fact that the boot code in the Broadcom chip is proprietary. They don't WANT anybody tinkering with it.

A good assembly-level programmer can absolutely write assembly-language code for his application code; the CPU is an ARM chip, and the instruction set is very straightforward. But you get no visibility into the chip, and must abide by the system calls Broadcom has defined.

Reply Report comment

Michael Black says:
February 21, 2020 at 5:38 pm

If you have to ask those questiins, I suspect you don't really know.

Reply Report comment

Martin says:
February 25, 2020 at 2:45 am

That is always true :-) If somebody knows something, he for sure does not have to ask questions about it.

Reply Report comment

Watchmaker says:
February 23, 2020 at 2:25 am

1, It's a computer.
2, It's a CPU with a built in MMU and a bunch of peripherals on an internal PCIe interface
3, ARM assembly. Look on the bare metal development section of the Raspberry Pi Foundation's forums for the IO addresses, or just have a peek in the Raspbian source code. A quick glance suggests the GPIO pins can be driven at about 12.5 MHz, but your coding may vary.

Reply Report comment

Julian Skidmore says:
February 21, 2020 at 4:59 am

4-bit processors are notoriously awkward. You'd think they'd have minimal and easy instruction sets, but for commercial reasons they're full of arcane op-codes and operations. Here's a summary of the TMS1000/1200 instruction set.

https://docs.google.com/document/d/1LW1kuvZIXua_MI4OHB8efh2I3jkhgfvNxr79-e2sh78/edit?usp=sharing

Some striking features are:

1. There's no bitwise logical operations (but you can clear or set individual bits), the ALU just adds and subtracts.
2. There's only an indexed addressing mode M(X,Y) which accesses 1 of 64 nybbles and X can only be loaded with a literal or inverted (though Y can be auto-incremented).
3. There's specific instructions for adding 1, 6 or 10 to the accumulator, but no other immediate adds!
4. Mnemonics are really hard to learn, you'd think they'd all be simple 3 letter things, but you get crazy stuff like: TAMZA, YNEA and A6AAC.

So, doing anything is pretty challenging which makes products like Speak and Spell all the more impressive!

Reply Report comment

Greenaum says:

February 21, 2020 at 7:00 pm

Speak And Spell must have had it's dicionary in a separate ROM. Actually that would be handy back then cos they could do international versions with just a ROM chip swap rather than a whole new run of TMS's needing masked up. Weird the best way to do things swaps around when you have certain limits imposed on the technology. Yeah, respect to everyone who had to program this bloody odd chip! Just the literature I've read online makes it out to be very odd. It has help for some things, like driving a 7-segment display, yet goes out of it's way to be weird for other things.

Is the "Add 6" instruction something to do with BCD? I can't quite see it but I suppose TI's chip designers are smarter than I am.

I thought programming the Atari 2600 was baroque, weird, and arbitrary. And it is! But the TMS makes the Atari's 6502, complete with a video chip you had to feed graphics to for each scanline, as it was being drawn, look like a nice day programming in LOGO to draw a lovely square.

Reply Report comment

jawnhenry says:

February 25, 2020 at 7:31 am

Actually, the TMS1000's instruction set is quite powerful, if not a bit arcane. This is a shame, since, as pointed out below, T.I. had it totally within their power to make the mnemonics much more 'remember-able'.

For example, there are actually fourteen immediate-add instructions (including "add 1"—"IA / IAC"); not just 'add 1', 'add 6', and 'add 10'. These are, respectively, "IA / IAC", and then "A2AAC" through "A14AAC".

These last 13 could have just as easily have been assigned the mnemonics of “A2” through “A14”, with no loss of clarity (they DO have the disadvantage of forcing the programmer to have to *actually remember* something. And as for things *similar* to “TAMDYN”—well. it—and those—are simple abominations which were completely avoidable and totally unnecessary.

see

http://www.nyx.net/~lturner/public_html/TMS1000ins.html

Reply Report comment

jawnhenry says:

February 21, 2020 at 7:12 am

Most people feel that there is something “magic” about the generation of the mnemonics which comprise the assembly-language instruction set of a computer (mainframe; mini-; mili-; micro-; nano-; femto-; atto-. Use whatever prefix makes you feel, and seem, important).
Nothing could be further from the truth; the mnemonics associated with a particular operation of a particular conglomeration of hardware—which comprise a new computer-hardware design—is strictly at the whim (quite literally) of the designer of that computer or, more than likely, the manager of the project.

When Lockheed Electronics designed their extremely powerful MAC-16 minicomputer at the same time the L-1011 wide-body jet was being designed and “pre-sold”, we accused LEC of pandering, because they assigned their assembly-language instruction for negative-binary arithmetic (a one cycle instruction to generate a negative number in the accumulator) the mnemonic

"TWA"—for "TWos-complement-the-Accumulator". You can't get a more memorable mnemonic than that—and that's precisely what "mnemonic" means.

TWA bought a lot of L-1011s; probably the first- or second largest buyer.

If you are considering a particular computer and find that it has anything other than a rational set of assembly-language mnemonics, run like hell; that's an indication of hubris—and self-importance—on the part of the designer(s). It's VERY easy to make it EASY. It's also a very good indication of the types of interactions you're likely to have with the manufacturer, down the road.

Reply Report comment

Claus Buchholz says:
February 24, 2020 at 2:28 am

How about the 68HC12's mnemonic for Sign EXtend?

Reply Report comment

jawnhenry says:
February 24, 2020 at 6:25 am

I suppose that Motorola could be accused of pandering to that contingent which speaks—or understands—Latin.

Reply Report comment

BradLevy says:

February 22, 2020 at 1:51 am

It should be pointed out that, while the TI Speak & Spell used a member of the TMS1000 family (one with 2K words of instruction rom and 128 nibbles of ram, vs the 1K and 64 in the diagram of your article), that was only handling the keyboard and display. The voice synthesis was handled by a separate TMC0281 chip which was groundbreaking for its time, establishing the category we now refer to as DSP (Digital Signal Processor). The speech data was stored in two separate ROM chips (16K bytes each) connected to the synthesizer chip. The speech content was encoded using linear predictive coding, specifying several oscillator/sound sources and filter coefficients to model sound generation. The synthesizer chip implemented digital filters applied to the sources using these coefficients to generate the output stream to a D/A converter. The digital filtering involved on-chip multi-stage pipelined hardware multiplication, at a density not seen in consumer products to that point.
(I find it amusing that one of the google searches turned up “How To Pronounce TMC0281: TMC0281 pronunciation”)

Reply Report comment

Charles Harris says:

May 5, 2020 at 5:53 pm

The Texas Instruments SR-16 calculator was one TMS1000 that had test mode access. Anyone used the test mode ?

Where would the archives be of TMS 1000 product be now, in a library somewhere ?

Reply Report comment

LambdaMikel says:
April 6, 2024 at 4:50 pm

Check this out:

https://hackaday.io/project/194876-exploring-the-science-fair-microcomputer-trainer/log/227981-dumping-the-tms1100

as well as for the TMS 1600:

https://youtu.be/oasEa7iOIXE

Reply Report comment

Charles Harris says:
May 5, 2020 at 5:54 pm

Has anyone got the full list of Masked Programs for the TMS1000 ?

Reply Report comment

Charles Harris says:
May 5, 2020 at 6:24 pm

In the 1970s there would have been documents/guides supplied to consumers to assist in their design for masked programs on the TMS1000 series. (I have

the overall 1976 TMS1000 data sheets)

Games, microwaves etc were amongst the designs created.

For example Tiger Games and all other consumers would have been provided with a standard set of guidelines, example programs, etc, to assist in design of their games etc. Tiger would have produced their design and then sent back to TI for approval. The documentation to assist in design is what I am looking for.

Also would like to locate a list of all the Masked Programs allocated for the TMS1000 series. MPxxxx etc

Above as referred to in:

TMS_1000_Series_MOS_LSI_One-Chip_Microcomputers_1975.pdf

1.3

DESIGN SUPPORT

Through a staff of experienced appl ication pro·

grammers, Texas Instruments will, upon request,

assist customers in evaluating applications, in training

designers to program the TMS1000 series and in

simulating programs. TI will also contract to write

programs to customer’s specifications.

*******TI has developed an assembler and simulator for

aiding software designs. These programs are available

on nationwide time·sharing systems and at TI compu·

ter facilities.*******

A TMS1000 series program (see flowchart, Figure 3) is written in assembly language using standard mnemonics. The assembler converts the source code (assembly language program) into machine code, which is transferred to a software simulation pro· gram. Also the assembler produces a machine code object deck. The object deck is used to produce a tape for hardware simulation or a tape for generating prototype tool ing.

The TMS1000 series programs are checked by soft· ware and hardware simulation. The software simula· tion offers the advantages of printed outputs for instruction traces or periodic outputs. The hardware simulation offers the designer the advantages of real·time simulation and testing asynchronous inputs.

A software user's guide is available.

After the algorithms have been checked and approved by the customer, the final object code and machine

option statements are supplied to TI. A gate mask is

generated and slices produced. After assembly and

testing, the prototypes are shipped to the customer

for approval. Upon receiving final approval, the part

is released for volume production at the requ ired rate

as one unique version of the TMS1000 family.

Any reference to archived info and suitable contacts would be appreciated.

Thanks

Charles

Reply Report comment

Blue Chip says:
July 31, 2023 at 5:22 pm

Hey Charles,
Have some decapped TMS1000 chips: https://seanriddle.com/decap.html
Did you have any luck finding the software guide?
BC

Reply Report comment

sswcharlie says:

March 28, 2021 at 3:08 pm

Anyone able to help with my messages (3) of 5 May 2020 ? Attention jawn henry, ian king, jenny list, mog, etc

Reply Report comment

Lucy says:

August 5, 2021 at 12:24 pm

The Microcontroller vs CPU debate is interesting. I'm fully aware that a microcontroller is a SoC, so they have a CPU and other components jammed into the single die, whereas CPUs are not complete SoCs. But given that uCs are fundamentally intended for real-time systems and need to be trusted with reliable handling of hard-realtime firmware execution requirements (usually enforced by a basic RTOS), I am curious now whether a computer CPU can reliably be trusted with hard-realtime firmware execution? (e.g. mission critical pacemaker style functions). I suppose you could install an RTOS onto a computer CPU, but is it architected to support trusted hard real-time event management like a uC? Dunno .. perhaps that is the difference between the "CPU" and the "microcontroller" .. it's real-time support mechanism?

Reply Report comment

genixia says:

December 19, 2021 at 9:32 am

TMS 1000 was also the brains in the Hornby Zero 1, an early digital model train command and control system available in the 1980s.

https://dccwiki.com/Hornby_Zero_1

Reply Report comment

Charles Harris says:
April 10, 2023 at 8:52 pm

A very nice system too. The output from the TMS1000 was down a two wire bus to the track, one to each rail.
Would both these outputs be identical ? Could the output be fed into a Arduino for fun and try an insert data into the original stream.
Charles

Reply Report comment

zpekic says:
February 17, 2023 at 4:59 pm

Looks like there is still much passion and interest about these remarkable old 4-bit devices. While not a TMS1100, I tried to resurrect a very similar microcontroller from that era – including adapting a compiler I wrote to be the assembler. The internal guts are explained as well as the test programs to validate that it works.

Reply Report comment

zpekic says:
February 17, 2023 at 5:00 pm

Link: https://hackaday.io/project/188614-iskra-emz1001a-a-virtual-resurrection

Reply Report comment

Leave a Reply

Please be kind and respectful to help make the comments section excellent. (Comment Policy)

This site uses Akismet to reduce spam. Learn how your comment data is processed.

NEVER MISS A HACK

SUBSCRIBE TO NEWSLETTER

Enter Email Address SUBSCRIBE