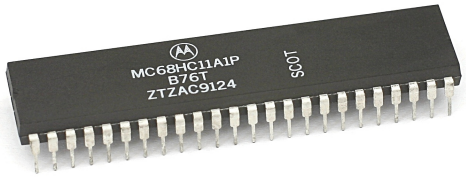
Contents

Chapter 1

Freescale 68HC11



Motorola MC68HC11 in a 48-pin dual in-line package (DIP)

The **68HC11** (**6811** or **HC11** for short) is an 8-bit microcontroller (µC) family introduced by Motorola in 1985.[1] Now produced by Freescale Semiconductor, it descended from the Motorola 6800 microprocessor. It is a CISC microcontroller. The 68HC11 devices are more powerful and more expensive than the 68HC08 microcontrollers, and are used in barcode readers, hotel card key writers, amateur robotics, and various other embedded systems. The MC68HC11A8 was the first MCU to include CMOS EEPROM.[2]

1.1 Architecture

Internally, the HC11 instruction set is upward compatible with the 6800, with the addition of a Y index register. (Instructions using the Y register have opcodes prefixed with the byte 0x18). It has two eight-bit accumulators, A and B, two sixteen-bit index registers, X and Y, a condition code register, a 16-bit stack pointer, and a program counter. In addition, there is an 8 x 8-bit multiply (A x B), with full 16-bit result, and Fractional/Integer 16-bit by 16-bit Divide instructions. A range of 16-bit instructions treat the A and B registers as a combined 16-bit D register for comparison (X and Y registers may also be compared to 16-bit memory operands), addition, subtraction and shift operations, or can add the B accumulator to the X or Y index registers. Bit test operations have also been added, performing a logical AND function between operands, setting the correct conditions codes, but not modifying the operands.

Different versions of the HC11 have different numbers of external ports, labeled alphabetically. The most common version has five ports, A, B, C, D, and E, but some have as few as 3 ports (version D3). Each port is eight-bits wide except for D, which is six bits (in some variations of the chip, D also has eight bits). It can be operated with an internal program and RAM (1 to 768 bytes) or an external memory of up to 64 kilobytes. With external memory, B and C are used as address and data bus. In this mode, port C is multiplexed to carry both the lower byte of the address and data.

1.2 Implementations

52-pin plastic leaded chip carrier (PLCC)

In the early 1990s Motorola produced an evaluation board kit for the 68HC11 with several UARTs, RAM, and an EPROM. The cost of the evaluation kit was $68.11.

The standard bootloader for the HC11 family is called BUFFALO, “Bit User Fast Friendly Aid to Logical Operation” (a BUFFALO prompt seen on the serial port at bootup is a sign that a board’s flash memory has been erased). Not all HC11 models come with the BUFFALO bootloader. The 68HC11A0 and A1 do not but the A8 does.

1.3 Other versions

The Freescale 68HC16 microcontroller family is intended as a 16-bit mostly software compatible upgrade of the 68HC11.

The Freescale 68HC12 microcontroller family is an enhanced 16-bit version of the 68HC11.

The Handy Board robotics controller by Fred Martin is based on the 68HC11.[3]

A MC68HC24 port replacement unit is available for the HC11. When placed on the external address bus, it replicates the original functions of B and C. Port A has input capture, output compare, pulse accumulator, and other timer functions; port D has serial I/O, and port E has an analog to digital converter (ADC).

1.4 References

[1] Hambley, Allan R.(1839). *Electrical Engineering: Principles and Applications*, Pearson Higher Education. p. 417. Digitized by Google. Retrieved on May 17, 2010.

[2] M68HC11 Reference Manual

[3] Handy Board Hardware

1.5 External links

- Freescale 68HC11 (Legacy) Part Info
- Wytec 68HC11 Development Board
- A fully synthesizable VHDL implementation of the HC11 CPU
- Digital Core Design 68HC11E - HDL IP Core
- Digital Core Design 68HC11F - HDL IP Core
- Digital Core Design 68HC11K - HDL IP Core
- Win/Linux-based freeware macro cross-assembler (ASM11)
- 4MHz-bus 68HC11F1-based board

Chapter 2

Intel 8085

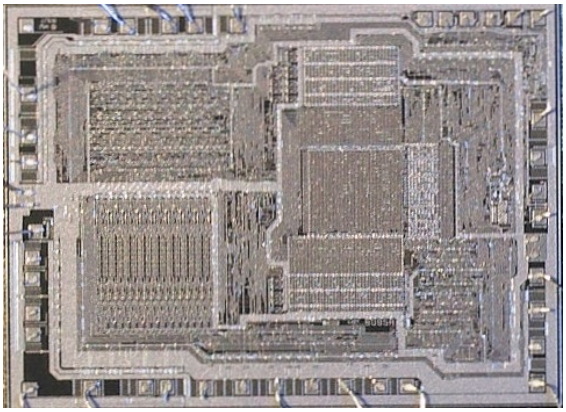
The **Intel 8085** (extquotedbl*eighty-eighty-five* extquotedbl) is an 8-bit microprocessor introduced by Intel in 1977. It was backward binary compatible with the more-famous Intel 8080 (only adding a few minor instructions) but required less supporting hardware, thus allowing simpler and less expensive microcomputer systems to be built.

The "5" in the model number came from the fact that the 8085 requires only a +5-Volt (V) power supply by using depletion mode transistors, rather than requiring the +5 V, −5 V and +12 V supplies the 8080 needed. This is similar to the competing Z80 (also 8080-derived) introduced the year before. These processors were sometimes used in computers running the CP/M operating system.

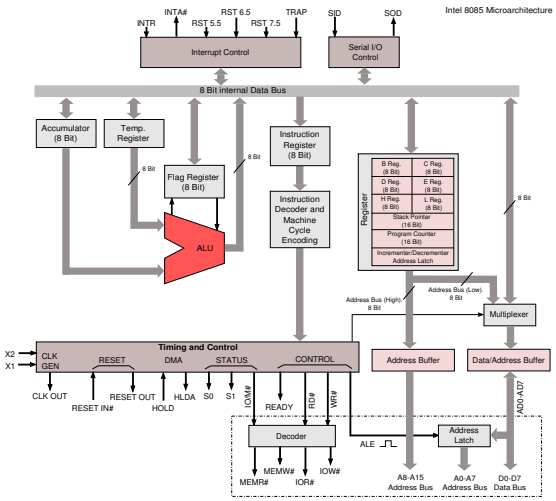
The Intel 8085 required at least an external ROM and RAM and an 8 bit address latch (both latches combined in the Intel 8755 2Kx8 EPROM / 2x8 I/O, Intel 8155 256-byte RAM and 22 I/O and 14 bit programmable Timer/Counter) so cannot technically be called a microcontroller.

Both designs (8080/8085) were eclipsed for desktop computers by the compatible Zilog Z80, which took over most of the CP/M computer market as well as taking a share of the booming home computer market in the early-to-mid-1980s.

The 8085 had a long life as a controller. Once designed into such products as the DECtape controller and the VT100 video terminal in the late 1970s, it served for new production throughout the life span of those products (generally longer than the product life of desktop computers).



Intel 8085A CPU Die



i8085 microarchitecture.

2.1 Description

The 8085 is a conventional von Neumann design based on the Intel 8080. Unlike the 8080 it does not multiplex state signals onto the data bus, but the 8-bit data bus was instead multiplexed with the lower part of the 16-bit address bus to limit the number of pins to 40. Pin No. 40 is used for the power supply (+5 V) and pin No. 20 for ground. Pin No. 39 is used as the hold pin. Pins No. 15 to No. 8 are generally used for address buses.

The processor was designed using nMOS circuitry and the later "H" versions were implemented in Intel's enhanced nMOS process called HMOS, originally developed for fast static RAM products. Only a 5 volt supply is needed, like competing processors and unlike the 8080. The 8085 uses approximately 6,500 transistors.[1]

The 8085 incorporates the functions of the 8224 (clock generator) and the 8228 (system controller), increasing the level of integration. A downside compared to similar contemporary designs (such as the Z80) was the fact

that the buses required demultiplexing; however, address latches in the Intel 8155, 8355, and 8755 memory chips allowed a direct interface, so an 8085 along with these chips was almost a complete system.

The 8085 has extensions to support new interrupts, with three maskable interrupts (RST 7.5, RST 6.5 and RST 5.5), one non-maskable interrupt (TRAP), and one externally serviced interrupt (INTR). The RST n.5 interrupts refer to actual pins on the processor, a feature which permitted simple systems to avoid the cost of a separate interrupt controller.

Like the 8080, the 8085 can accommodate slower memories through externally generated wait states (pin 35, READY), and has provisions for Direct Memory Access (DMA) using HOLD and HLDA signals (pins 39 and 38). An improvement over the 8080 was that the 8085 can itself drive a piezoelectric crystal directly connected to it, and a built in clock generator generates the internal high amplitude two-phase clock signals at half the crystal frequency (a 6.14 MHz crystal would yield a 3.07 MHz clock, for instance).

The 8085 is a binary compatible follow up on the 8080, using the same basic instruction set as the 8080. Only a few minor instructions were new to the 8085 above the 8080 set.

2.1.1 Programming model

The processor has seven 8-bit registers accessible to the programmer, named A, B, C, D, E, H, and L, where A is the 8-bit accumulator and the other six can be used as independent byte-registers or as three 16-bit register pairs, BC, DE, and HL, depending on the particular instruction. Some instructions use HL as a (limited) 16-bit accumulator. As in the 8080, the contents of the memory address pointed to by HL could be accessed as pseudo register M. It also has a 16-bit program counter and a 16-bit stack pointer to memory (replacing the 8008's internal stack). Instructions such as PUSH PSW, POP PSW affected the Program Status Word (Accumulator and Flags).

2.1.2 Commands/instructions

As in many other 8-bit processors, all instructions are encoded in a single byte (including register-numbers, but excluding immediate data), for simplicity. Some of them are followed by one or two bytes of data, which could be an immediate operand, a memory address, or a port number. Like larger processors, it has CALL and RET instructions for multi-level procedure calls and returns (which can be conditionally executed, like jumps) and instructions to save and restore any 16-bit register-pair on the machine stack. There are also eight one-byte call instructions (RST) for subroutines located at the fixed addresses 00h, 08h, 10h,...,38h. These were intended to be supplied by external hardware in order to invoke a corresponding interrupt-service routine, but are also often employed as fast system calls. The most sophisticated command was XTHL, which is used for exchanging the register pair HL with the value stored at the address indicated by the stack pointer.

8-bit instructions

Most 8-bit operations work on the 8-bit accumulator (the A register). For two operand 8-bit operations, the other operand can be either an immediate value, another 8-bit register, or a memory cell addressed by the 16-bit register pair HL. Direct copying is supported between any two 8-bit registers and between any 8-bit register and a HL-addressed memory cell. Due to the regular encoding of the MOV-instruction (using a quarter of available opcode space) there are redundant codes to copy a register into itself (MOV B,B, for instance), which are of little use, except for delays. However, what would have been a copy from the HL-addressed cell into itself (i.e., MOV M,M) instead encodes the HLT instruction, halting execution until an external reset or interrupt occurred (providing interrupts were enabled).

16-bit operations

Although the 8085 is an 8-bit processor, it has some 16-bit operations. Any of the three 16-bit register pairs (BC, DE, HL or SP) could be loaded with an immediate 16-bit value (using LXI), incremented or decremented (using INX and DCX), or added to HL (using DAD). LHLD loaded HL from directly-addressed memory and SHLD stored HL likewise. The XCHG operation exchanges the values of HL and DE. Adding HL to itself performs a 16-bit arithmetical left shift with one instruction. The only 16 bit instruction that affects any flag was DAD (adding HL to BC, DE, HL or SP), which updates the carry flag to facilitate 24-bit or larger additions and left shifts (for a floating point mantissa for instance). Adding the stack pointer to HL is useful for indexing variables in (recursive) stack frames. A stack frame can be allocated using DAD SP and SPHL, and a branch to a computed pointer can be done with PCHL. These abilities make it feasible to compile languages such as PL/M, Pascal, or C with 16-bit variables and produce 8085 machine code.

Subtraction and bitwise logical operations on 16 bits is done in 8-bit steps. Operations that have to be implemented by program code (subroutine libraries) included comparisons of signed integers as well as multiply and divide.

2.1.3 Input/output scheme

The 8085 supported up to 256 input/output (I/O) ports, accessed via dedicated Input/Output instructions—taking

port addresses as operands. This Input/Output mapping scheme was regarded as an advantage, as it freed up the processor's limited address space.

2.1.4 Development system

Intel produced a series of development systems for the 8080 and 8085, known as the MDS-80 Microprocessor System. The original development system had an 8080 processor. Later 8085 and 8086 support was added including ICE (in-circuit emulators). It was a large and heavy desktop box, about a 20" cube (in the Intel corporate blue colour) which included a CPU, monitor, and a single 8 inch floppy disk drive. Later an external box was available with two more floppy drives. It ran the ISIS operating system and could also operate an emulator pod and an external EPROM programmer. This unit used the Multibus card cage which was intended just for the development system. A surprising number of spare card cages and processors were being sold, leading to the development of the Multibus as a separate product .

The later iPDS was a portable unit, about 8" x 16" x 20", with a handle. It had a small green screen, a keyboard built into the top, a 5¼ inch floppy disk drive, and ran the ISIS-II operating system. It could also accept a second 8085 processor, allowing a limited form of multiprocessor operation where both processors ran simultaneously and independently. The screen and keyboard could be switched between them, allowing programs to be assembled on one processor (large programs took awhile) while files were edited in the other. It had a bubble memory option and various programming modules, including EPROM and Intel 8048 and 8051 programming modules which were plugged into the side, replacing stand-alone device programmers. In addition to an 8080/8085 assembler, Intel produced a number of compilers including PL/M-80 and Pascal languages, and a set of tools for linking and statically locating programs to enable them to be burnt into EPROMs and used in embedded systems.

A lower cost SDK-85 System Design Kit board was provided with an 8085 CPU, 8355 ROM containing a debugging monitor program, 8155 RAM and 22 I/O, 8279 hex keypad and 8-digit 7-segment LED, TTY (Teletype) 20 mA current loop serial interface. Pads were available for one more 2Kx8 8755 EPROM and another 256 byte RAM 8155 I/O Timer/Counter could be optionally added. All data, control and address signals were available on dual pin headers and a large prototype area was provided.

2.2 Applications

For the extensive use of 8085 in various applications, the microprocessor is provided with an instruction set which consists of various instructions such as MOV, ADD, SUB, JMP, etc. These instructions are written in the form of a program which is used to perform various operations such as branching, addition, subtraction, bitwise logical and bit shift operations. More complex operations and other arithmetic operations must be implemented in software. For example, multiplication is implemented using a multiplication algorithm.

The 8085 processor was used in a few early personal computers, for example, the TRS-80 Model 100 line used an OKI manufactured 80C85 (MSM80C85ARS). The CMOS version 80C85 of the NMOS/HMOS 8085 processor has several manufacturers. Some manufacturers provide variants with additional functions such as additional instructions. The rad-hard version of the 8085 has been in on-board instrument data processors for several NASA and ESA space physics missions in the 1990s and early 2000s, including CRRES, Polar, FAST, Cluster, HESSI, the Sojourner Mars Rover,[2] and THEMIS. The Swiss company SAIA used the 8085 and the 8085-2 as the CPUs of their PCA1 line of programmable logic controllers during the 1980s.

Pro-Log Corp. put the 8085 and supporting hardware on an STD Bus format card containing CPU, RAM, sockets for ROM/EPROM, I/O and external bus interfaces. The included Instruction Set Reference Card used entirely different mnemonics for the Intel 8085 CPU, as the product was a direct competitor to Intel's Multibus card offerings.

2.3 MCS-85 family

The 8085 CPU was one part of a family of chips developed by Intel, for building a complete system. Many of these support chips were also used with other processors. The original IBM PC based on the Intel 8088 processor used several of these chips; the equivalent functions today are provided by VLSI chips, namely the extquotedblSouthbridge extquotedbl chips.

- 8085-CPU
- 8155-RAM+ 3 I/O Ports+Timer
- 8156-RAM+ 3 I/O Ports+Timer
- 8185-SRAM
- 8355-16,384-bit (2048 x 8) ROM with I/O
- 8604-4096-bit (512 x 8) PROM
- 8755-EPROM+2 I/O Ports
- 8202-Dynamic RAM Controller
- 8203-Dynamic RAM Controller
- 8205-1 Of 8 Binary Decoder

- 8206-Error Detection & Correction Unit
- 8207-DRAM Controller
- 8210-TTL To MOS Shifter & High Voltage Clock Driver
- 8212-8 Bit I/O Port
- 8216-4 Bit Parallel Bidirectional Bus Driver
- 8218/8219-Bus Controller
- 8226-4 Bit Parallel Bidirectional Bus Driver
- 8231-Arithmetic Processing Unit
- 8232-Floating Point Processor
- 8237-DMA Controller
- 8251-Communication Controller
- 8253-Programmable Interval Timer
- 8254-Programmable Interval Timer
- 8255-Programmable Peripheral Interface
- 8256-Multifunction Support Controller
- 8257-DMA Controller
- 8259-Programmable Interrupt Controller
- 8271-Programmable Floppy Disk Controller
- 8272-Single/Double Density Floppy Disk Controller
- 8273-Programmable HDLC/SDLC Protocol Controller
- 8274-Multi-Protocol Serial Controller
- 8275-CRT Controller
- 8276-Small System CRT Controller
- 8275-Programmable Key Board Interface
- 8279-Key Board/Display Controller
- 8282−8-bit Non-Inverting Latch with Output Buffer
- 8283−8-bit Inverting Latch with Output Buffer
- 8291-GPIB Talker/Listener
- 8293-GPIB Transceiver
- 8294-Data Encryption/Decryption Unit+1 O/P Port
- 8295-Dot Matrix Printer Controller

2.4 Educational use

In many engineering schools[3] [4]the 8085 processor is used in introductory microprocessor courses. Trainer kits composed of a printed circuit board, 8085, and supporting hardware are offered by various companies. These kits usually include complete documentation allowing a student to go from solder to assembly language programming in a single course. Also the architecture of this and the associated instruction set is easy for a student to understand.

2.5 Simulators

Some of the simulators available for the 8085 microprocessor are listed below:

- GNUSim8085 - It consists of a simulator, assembler and a debugger. It is available for both Windows and Linux operating systems.
- Win85 - Open source (under the MIT license) simulator/debugger for Windows [5]
- 8085 simulator - It includes a simulated keypad, an assembler and a simulator.
- Intel 8085 Simulator for Android. [6]
- ENVI85 - It was written by professors Stefan Fedyschyn and Edwin Kay. This and the above simulator are provided on the CD that accompanies the book, *Microprocessor Architecture, Programming and Applications with the 8085* by Ramesh Gaonkar.

2.6 See also

- GNUSim8085 – An open source multi-platform simulator software for the 8085 processor.
- IBM System/23 Datamaster gave IBM designers familiarity with the 8085 support chips used in the IBM PC

2.7 References

[1] The history of the microcomputer-invention and evolution, S Mazor - Proceedings of the IEEE, 1995

[2] A Description of the Rover Sojourner

[3]

[4] Микропроцесорски системи

[5] Win85 project homepage

[6]

- William Stallings *Computer Organization and Architecture: Designing for Performance 8th Ed.* Prentice Hall, 2009 ISBN 0-13-607373-5
- Abhishek Yadav *Microprocessor 8085, 8086* Firewall Media, 2008 ISBN 81-318-0356-2
- Ramesh Gaonkar *Microprocessor Architecture, Programming and Applications with the 8085* Penram International Publishing ISBN 81-87972-09-2
- Bill Detwiler *Tandy TRS-80 Model 100 Teardown* Tech Republic, 2011 Web

2.8 External links

- Pin diagram and pin description of 8085
- Function of IC's Used in 8085 Microprocessor

Chapter 3

Intel 8086

The **8086**[1] (extquotedbl*eighty-eighty-six* extquotedbl, also called **iAPX 86**)[2] is a 16-bit microprocessor chip designed by Intel between early 1976 and mid-1978, when it was released. The Intel 8088, released in 1979, was a slightly modified chip with an external 8-bit data bus (allowing the use of cheaper and fewer supporting ICs[note 1]), and is notable as the processor used in the original IBM PC design, including the widespread version called IBM PC XT.

The 8086 gave rise to the x86 architecture which eventually turned out as Intel's most successful line of processors.

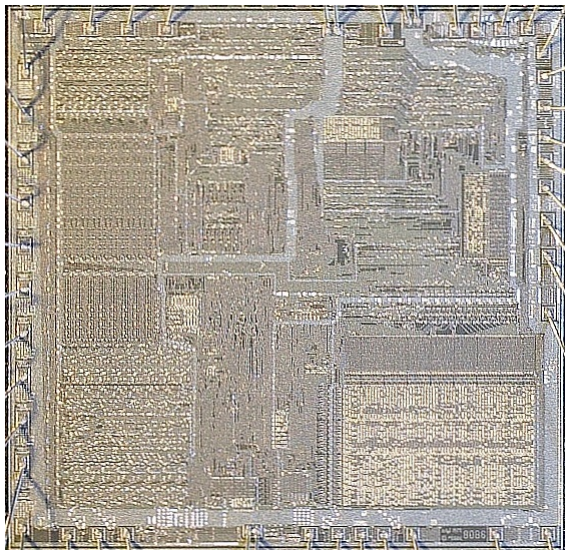
3.1 History

3.1.1 Background

In 1972, Intel launched the 8008, the first 8-bit microprocessor.[note 2] It implemented an instruction set designed by Datapoint corporation with programmable CRT terminals in mind, that also proved to be fairly general purpose. The device needed several additional ICs to produce a functional computer, in part due to it being packaged in a small 18-pin "memory-package", which ruled out the use of a separate address bus (Intel was primarily a DRAM manufacturer at the time).

Two years later, Intel launched the 8080,[note 3] employing the new 40-pin DIL packages originally developed for calculator ICs to enable a separate address bus. It had an extended instruction set that was source- (not binary-) compatible with the 8008 and also included some 16-bit instructions to make programming easier. The 8080 device, often described as the first truly useful microprocessor, was eventually replaced by the depletion-load based 8085 (1977) which could cope with a single 5V power supply instead of the three different operating voltages of earlier chips.[note 4] Other well known 8-bit microprocessors that emerged during these years were Motorola 6800 (1974), General Instrument PIC16X (1975), MOS Technology 6502 (1975), Zilog Z80 (1976), and Motorola 6809 (1978).

3.1.2 The first x86 design



Intel 8086 CPU Die Image

The 8086 project started in May 1976 and was originally intended as a temporary substitute for the ambitious and delayed iAPX 432 project. It was an attempt to draw attention from the less-delayed 16 and 32-bit processors of other manufacturers (such as Motorola, Zilog, and National Semiconductor) and at the same time to counter the threat from the Zilog Z80 (designed by former Intel employees), which became very successful. Both the architecture and the physical chip were therefore developed rather quickly by a small group of people, and using the same basic microarchitecture elements and physical implementation techniques as employed for the slightly older 8085 (and for which the 8086 also would function as a continuation).

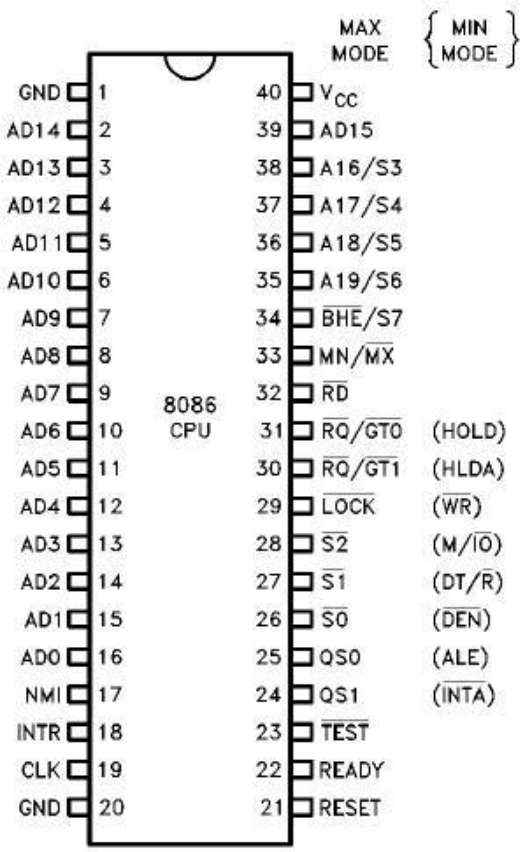
Marketed as source compatible, the 8086 was designed to allow assembly language for the 8008, 8080, or 8085 to be automatically converted into equivalent (sub-optimal) 8086 source code, with little or no hand-editing. The programming model and instruction set was (loosely) based on the 8080 in order to make this possible. However, the 8086 design was expanded to support full 16-bit processing, instead of the fairly basic 16-bit capabilities of the 8080/8085.

New kinds of instructions were added as well; full support for signed integers, base+offset addressing, and self-repeating operations were akin to the Z80 design[3] but were all made slightly more general in the 8086. Instructions directly supporting nested ALGOL-family languages such as Pascal and PL/M were also added. According to principal architect Stephen P. Morse, this was a result of a more software centric approach than in the design of earlier Intel processors (the designers had experience working with compiler implementations). Other enhancements included microcoded multiply and divide instructions and a bus-structure better adapted to future co-processors (such as 8087 and 8089) and multiprocessor systems.

The first revision of the instruction set and high level architecture was ready after about three months,[note 5] and as almost no CAD-tools were used, four engineers and 12 layout people were simultaneously working on the chip.[note 6] The 8086 took a little more than two years from idea to working product, which was considered rather fast for a complex design in 1976–1978.

The 8086 was sequenced[note 7] using a mixture of random logic[4] and microcode and was implemented using depletion-load nMOS circuitry with approximately 20,000 active transistors (29,000 counting all ROM and PLA sites). It was soon moved to a new refined nMOS manufacturing process called HMOS (for High performance MOS) that Intel originally developed for manufacturing of fast static RAM products.[note 8] This was followed by HMOS-II, HMOS-III versions, and, eventually, a fully static CMOS version for battery-powered devices, manufactured using Intel's CHMOS processes.[note 9] The original chip measured 33 mm^2 and minimum feature size was 3.2 µm.

The architecture was defined by Stephen P. Morse with some help and assistance by Bruce Ravenel (the architect of the 8087) in refining the final revisions. Logic designer Jim McKevitt and John Bayliss were the lead engineers of the hardware-level development team[note 10] and William Pohlman the manager for the project. The legacy of the 8086 is enduring in the basic instruction set of today's personal computers and servers; the 8086 also lent its last two digits to later extended versions of the design, such as the Intel 286 and the Intel 386, all of which eventually became known as the x86 family. (Another reference is that the PCI Vendor ID for Intel devices is 8086_h.)



The 8086 pin-assignments in min and max mode

3.2 Details

3.2.1 Buses and operation

All internal registers, as well as internal and external data buses, were 16 bits wide, firmly establishing the "16-bit microprocessor" identity of the 8086. A 20-bit external address bus gave a 1 MB physical address space (2^{20} = 1,048,576). This address space was addressed by means of internal 'segmentation'. The data bus was multiplexed with the address bus in order to fit a standard 40-pin dual in-line package. 16-bit I/O addresses meant 64 KB of separate I/O space (2^{16} = 65,536). The maximum **linear** address space was limited to 64 KB, simply because internal registers were only 16 bits wide. Programming over 64 KB boundaries involved adjusting segment registers (see below) and remained so until the 80386 introduced wider (32 bits) main registers (the memory management hardware in the 286 did not help in this regard, as registers were still 16 bits).

Some of the control pins, which carry essential signals for all external operations, had more than one function depending upon whether the device was operated in *min* or *max* mode. The former was intended for small single processor systems while the latter was for medium or large systems, using more than one processor.

3.2.2 Registers and instructions

The 8086 has eight more or less general 16-bit registers (including the stack pointer but excluding the instruction pointer, flag register and segment registers). Four of them, AX, BX, CX, DX, could also be accessed as twice

as many 8-bit registers (see figure) while the other four, BP, SI, DI, SP, were 16-bit only.

Due to a compact encoding inspired by 8-bit processors, most instructions were one-address or two-address operations which means that the result was stored in one of the operands. At most one of the operands could be in memory, but this memory operand could also be the *destination*, while the other operand, the *source*, could be either *register* or *immediate*. A single memory location could also often be used as both *source* and *destination* which, among other factors, further contributed to a code density comparable to (and often better than) most eight bit machines.

Although the degree of generality of most registers were much greater than in the 8080 or 8085, it was still fairly low compared to the typical contemporary minicomputer, and registers were also sometimes used implicitly by instructions. While perfectly sensible for the assembly programmer, this made register allocation for compilers more complicated compared to more regular 16- and 32-bit processors such as the PDP-11, VAX, 68000, 32016 etc. On the other hand, it was more regular and orthogonal than ubiquitous but rather minimalistic 8-bit microprocessors such as the 6502, 6800, 6809, 8085, MCS-48, 8051 and other contemporary accumulator based machines. It was significantly easier to construct an efficient code generator for the 8086 design.

Another factor for this was that the 8086 also introduced some new instructions (not present in the 8080 and 8085) to better support stack based high level programming languages such as Pascal and PL/M; some of the more useful ones were **push** *mem-op*, and **ret** *size*, supporting the "pascal calling convention extquotedbl directly. (Several others, such as **push** *immed* and **enter**, would be added in the subsequent 80186, 80286, and 80386 processors.)

The 8086 had a 64 KB of 8-bit (or alternatively 32 K-word of 16-bit) I/O space. A 64 KB (one segment) stack growing towards lower addresses is supported in hardware; 2-byte words are pushed to the stack and the stack top is pointed to by SS:SP. There are 256 interrupts, which can be invoked by both hardware and software. The interrupts can cascade, using the stack to store the return addresses.

3.2.3 Flags

8086 has a 16-bit flags register. Nine of these condition code flags are active, and indicate the current state of the processor: Carry flag (CF), Parity flag (PF), Auxiliary carry flag (AF), Zero flag (ZF), Sign flag (SF), Trap flag (TF), Interrupt flag (IF), Direction flag (DF), and Overflow flag (OF).

3.2.4 Segmentation

See also: x86 memory segmentation

There are also four 16-bit segment registers (see figure) that allow the 8086 CPU to access one megabyte of memory in an unusual way. Rather than concatenating the segment register with the address register, as in most processors whose address space exceeded their register size, the 8086 shifts the 16-bit segment only four bits left before adding it to the 16-bit offset (16×segment + offset), therefore producing a 20-bit external (or effective or physical) address from the 32-bit segment:offset pair. As a result, each external address can be referred to by 2^{12} = 4096 different segment:offset pairs.

Although considered complicated and cumbersome by many programmers, this scheme also has advantages; a small program (less than 64 KB) can be loaded starting at a fixed offset (such as 0000) in its own segment, avoiding the need for relocation, with at most 15 bytes of alignment waste.

Compilers for the 8086-family commonly support two types of pointer, *near* and *far*. Near pointers are 16-bit offsets implicitly associated with the program's code or data segment and so can be used only within parts of a program small enough to fit in one segment. Far pointers are 32-bit segment:offset pairs resolving to 20-bit external addresses. Some compilers also support *huge* pointers, which are like far pointers except that pointer arithmetic on a huge pointer treats it as a linear 20-bit pointer, while pointer arithmetic on a far pointer wraps around within its 16-bit offset without touching the segment part of the address.

To avoid the need to specify *near* and *far* on numerous pointers, data structures, and functions, compilers also support "memory models" which specify default pointer sizes. The *tiny* (max 64K), *small* (max 128K), *compact* (data > 64K), *medium* (code > 64K), *large* (code,data > 64K), and *huge* (individual arrays > 64K) models cover practical combinations of near, far, and huge pointers for code and data. The *tiny* model means that code and data are shared in a single segment, just as in most 8-bit based processors, and can be used to build *.com*-files for instance. Precompiled libraries often came in several versions compiled for different memory models.

According to Morse et al., the designers actually contemplated using an 8-bit shift (instead of 4-bit), in order to create a 16 MB physical address space. However, as this would have forced segments to begin on 256-byte boundaries, and 1 MB was considered very large for a microprocessor around 1976, the idea was dismissed. Also, there were not enough pins available on a low-cost 40-pin package for the additional four address bus pins.[5]

In principle, the address space of the x86 series *could* have been extended in later processors by increasing the shift value, as long as applications obtained their seg-

ments from the operating system and did not make assumptions about the equivalence of different segment: offset pairs.[note 11] In practice the use of "huge" pointers and similar mechanisms was widespread and the flat 32-bit addressing made possible with the 32-bit offset registers in the 80386 eventually extended the limited addressing range in a more general way (see below).

Intel could have decided to implement memory in 16 bit words (which would have eliminated the BHE signal along with much of the address bus complexities already described). This would mean that all instruction object codes and data would have to be accessed in 16-bit units. Users of the 8080 long ago realised, in hindsight, that the processor makes very efficient use of its memory. By having a large number of 8-bit object codes, the 8080 produces object code as compact as some of the most powerful minicomputers on the market at the time.[6]:5-26

If the 8086 is to retain 8-bit object codes and hence the efficient memory use of the 8080, then it cannot guarantee that (16-bit) opcodes and data will lie on an even-odd byte address boundary. The first 8-bit opcode will shift the next 8-bit instruction to an odd byte or a 16-bit instruction to an odd-even byte boundary. By implementing the BHE signal and the extra logic needed, the 8086 has allowed instructions to exist as 1-byte, 3-byte or any other odd byte object codes.[6]:5-26

Simply put: this is a trade off. If memory addressing is simplified so that memory is only accessed in 16-bit units, memory will be used less efficiently. Intel decided to make the logic more complicated, but memory use more efficient. This was at a time when memory size was considerably smaller, and at a premium, than that which users are used to today.[6]:5-26

Porting older software

Small programs could ignore the segmentation and just use plain 16-bit addressing. This allowed 8-bit software to be quite easily ported to the 8086. The authors of MS-DOS took advantage of this by providing an Application Programming Interface very similar to CP/M as well as including the simple *.com* executable file format, identical to CP/M. This was important when the 8086 and MS-DOS were new, because it allowed many existing CP/M (and other) applications to be quickly made available, greatly easing acceptance of the new platform.

3.2.5 Example code

The following 8086/8088 assembler source code is for a subroutine named _memcpy that copies a block of data bytes of a given size from one location to another. The data block is copied one byte at a time, and the data movement and looping logic utilizes 16-bit operations.

; _memcpy(dst, src, len) ; Copy a block of memory from one location to another. ; ; Entry stack parameters ; [BP+6] = len, Number of bytes to copy ; [BP+4] = src, Address of source data block ; [BP+2] = dst, Address of target data block ; ; Return registers ; AX = Zero 0000:1000 org 1000h ; Start at 0000:1000h 0000:1000 _memcpy proc 0000:1000 55 push bp ; Set up the call frame 0000:1001 89 E5 mov bp,sp 0000:1003 06 push es ; Save ES 0000:1004 8B 4E 06 mov cx,[bp+6] ; Set CX = len 0000:1007 E3 11 jcxz done ; If len=0, return 0000:1009 8B 76 04 mov si,[bp+4] ; Set SI = src 0000:100C 8B 7E 02 mov di,[bp+2] ; Set DI = dst 0000:100F 1E push ds ; Set ES = DS 0000:1010 07 pop es 0000:1011 8A 04 loop mov al,[si] ; Load AL from [src] 0000:1013 88 05 mov [di],al ; Store AL to [dst] 0000:1015 46 inc si ; Increment src 0000:1016 47 inc di ; Increment dst 0000:1017 49 dec cx ; Decrement len 0000:1018 75 F7 jnz loop ; Repeat the loop 0000:101A 07 done pop es ; Restore ES 0000:101B 5D pop bp ; Restore previous call frame 0000:101C 29 C0 sub ax,ax ; Set AX = 0 0000:101E C3 ret ; Return 0000:101F end proc

The code above uses the BP (base pointer) register to establish a call frame, an area on the stack that contains all of the parameters and local variables for the execution of the subroutine. This kind of calling convention supports reentrant and recursive code, and has been used by most ALGOL-like languages since the late 1950s. The ES segment register is saved on the stack and replaced with the value of the DS segment register, so that the MOV AL instructions will operate within the same source and destination data segment. Before returning, the subroutine restores the previous value of the ES register.

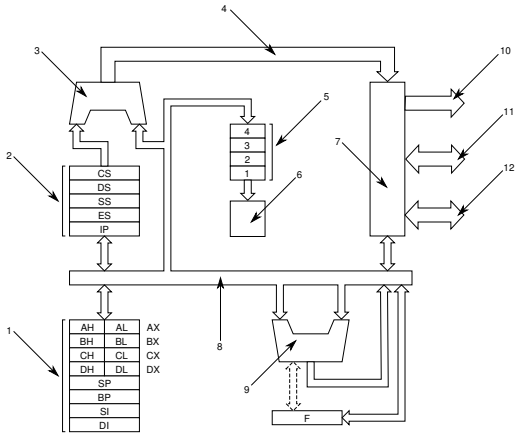
The above routine is a rather cumbersome way to copy blocks of data. Provided the source and the destination blocks reside within single 65,536 byte segments (a requirement of the above routine), advantage can be taken of the 8086's block MOV instructions. The loop section of the above can be replaced by:

0000:1011 F2 loop rep ; Repeat until CX=0 0000:1012 A5 movsw ; Move the data block

This copies the block of data one word at a time. The REP instruction causes the following MOVSW to repeat until CX=0, automatically incrementing SI and DI as it repeats. Alternatively the MOVSB or MOVSD instructions can be used to copy single bytes or double words at a time. Most assemblers will assemble correctly if the REP instruction is used as a prefix to MOVSW as in REP MOVSW.

This routine will operate correctly if interupted, because the program counter will continue to point to the REP instruction until the block copy is completed. The copy will therefore continue from where it left off when the interupt service routine returns control.

3.2.6 Performance



Simplified block diagram over Intel 8088 (a variant of 8086); 1=main registers; 2=segment registers and IP; 3=address adder; 4=internal address bus; 5=instruction queue; 6=control unit (very simplified!); 7=bus interface; 8=internal databus; 9=ALU; 10/11/12=external address/data/control bus.

Although partly shadowed by other design choices in this particular chip, the multiplexed address and data buses limited performance slightly; transfers of 16-bit or 8-bit quantities were done in a four-clock memory access cycle, which was faster on 16-bit, although slower on 8-bit quantities, compared to many contemporary 8-bit based CPUs. As instructions varied from one to six bytes, fetch and execution were made concurrent and decoupled into separate units (as it remains in today's x86 processors): The *bus interface unit* fed the instruction stream to the *execution unit* through a 6-byte prefetch queue (a form of loosely coupled pipelining), speeding up operations on registers and immediates, while memory operations unfortunately became slower (four years later, this performance problem was fixed with the 80186 and 80286). However, the full (instead of partial) 16-bit architecture with a full width ALU meant that 16-bit arithmetic instructions could now be performed with a single ALU cycle (instead of two, via internal carry, as in the 8080 and 8085), speeding up such instructions considerably. Combined with orthogonalizations of operations versus operand-types and addressing modes, as well as other enhancements, this made the performance gain over the 8080 or 8085 fairly significant, despite cases where the older chips may be faster (see below).

- EA = time to compute effective address, ranging from 5 to 12 cycles.
- Timings are best case, depending on prefetch status, instruction alignment, and other factors.

As can be seen from these tables, operations on registers and immediates were fast (between 2 and 4 cycles), while memory-operand instructions and jumps were quite slow; jumps took more cycles than on the simple 8080 and 8085, and the 8088 (used in the IBM PC) was additionally hampered by its narrower bus. The reasons why most memory related instructions were slow were threefold:

- Loosely coupled fetch and execution units are efficient for instruction prefetch, but not for jumps and random data access (without special measures).
- No dedicated address calculation adder was afforded; the microcode routines had to use the main ALU for this (although there was a dedicated *segment* + *offset* adder).
- The address and data buses were multiplexed, forcing a slightly longer (33~50%) bus cycle than in typical contemporary 8-bit processors.

However, memory access performance was drastically enhanced with Intel's next generation chips. The 80186 and 80286 both had dedicated address calculation hardware, saving many cycles, and the 80286 also had separate (non-multiplexed) address and data buses.

3.2.7 Floating point

The 8086/8088 could be connected to a mathematical coprocessor to add hardware/microcode-based floating point performance. The Intel 8087 was the standard math coprocessor for the 8086 and 8088, operating on 80-bit numbers. Manufacturers like Cyrix (8087-compatible) and Weitek (*non* 8087-compatible) eventually came up with high performance floating point co-processors that competed with the 8087 as well as with the subsequent, higher performing Intel 80387.

3.3 Chip versions

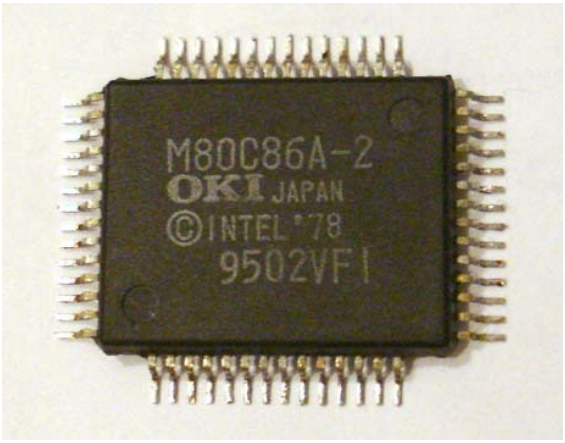
The clock frequency was originally limited to 5 MHz (IBM PC used 4.77 MHz, 4/3 the standard NTSC color burst frequency), but the last versions in HMOS were specified for 10 MHz. HMOS-III and CMOS versions were manufactured for a long time (at least a while into the 1990s) for embedded systems, although its successor, the 80186/80188 (which includes some on-chip peripherals), has been more popular for embedded use.

The 80C86, the CMOS version of the 8086, was used in the GRiDPad, Toshiba T1200, HP 110, and finally the 1998-1999 Lunar Prospector.

3.3.1 Derivatives and clones

Compatible—and, in many cases, enhanced—versions were manufactured by Fujitsu, Harris/Intersil, OKI,

Soviet clone KP1810BM86.



OKI M80C86A QFP-56.

NEC μPD8086D-2 (8MHz) from 1984year 19week JAPAN (clone of Intel D8086-2)

Siemens AG, Texas Instruments, NEC, Mitsubishi, AMD. For example, the NEC V20 and NEC V30 pair were hardware compatible with the 8088 and 8086 even though NEC made original Intel clones μPD8088D and μPD8086D, respectively, but incorporated the instruction set of the 80186 along with some (but not all) of the 80186 speed enhancements, providing a drop-in capability to upgrade both instruction set and processing speed without manufacturers having to modify their designs. Such relatively simple and low-power 8086-compatible processors in CMOS are still used in embedded systems.

The electronics industry of the Soviet Union was able to replicate the 8086 through both industrial espionage and reverse engineering. The resulting chip, K1810BM86, was binary and pin-compatible with the 8086.

i8088 and i8086 were respectively the cores of the Soviet-made PC-compatible EC1831 and EC1832 desktops (EC1831 is the EC identification of IZOT 1037C and EC1832 is the EC identification of IZOT 1036C, developed and manufactured in Bulgaria). However, EC1832 computer (IZOT 1036C) had significant hardware differences from its authentic prototype, and the data/address bus circuitry was designed independently of Intel products. EC1832 was the first PC compatible computer with dynamic bus sizing (US Pat. No 4,831,514). Later some of the ES1832 principles were adopted in PS/2 (US Pat. No 5,548,786) and some other machines (UK Patent Application, Publication No. GB-A-2211325, Published June. 28, 1989).

3.4 Hardware modes

The 8086 and 8088 support two hardware modes: maximum mode and minimum mode. Maximum mode is for large applications such as multiprocessing and is also required to support the 8087 coprocessor. The mode is usually hard-wired into the circuit and cannot be changed by software. Specifically, pin #33 (MN/MX) is either wired to voltage or to ground to determine the mode. Changing the state of pin #33 changes the function of certain other pins, most of which have to do with how the CPU handles the (local) bus. The IBM PC and PC/XT use an Intel 8088 running in maximum mode, which allows the CPU to work with an optional 8087 coprocessor installed in the math coprocessor socket on the PC or PC/XT mainboard. (The PC and PC/XT may require Max mode for other reasons, such as perhaps to support the DMA controller.)

3.5 Peripherals

- Intel 8237: direct memory access (DMA) controller
- Intel 8251: USART
- Intel 8253: programmable interval timer
- Intel 8255: programmable peripheral interface
- Intel 8259: programmable interrupt controller
- Intel 8279: keyboard/display controller
- Intel 8282/8283: 8-Bit latch
- Intel 8284: clock generator
- Intel 8286/8287: bidirectional 8-Bit driver
- Intel 8288: bus controller
- Intel 8289: bus arbiter

3.6 Microcomputers using the 8086

- The Xerox NoteTaker was one of the earliest portable computer designs in 1978 and used three 8086 chips (as CPU, graphics processor, and i/o processor), but never entered commercial production.
- Seattle Computer Products shipped S-100 bus based 8086 systems (SCP200B) as early as November 1979.
- The Norwegian Mycron 2000, introduced in 1980.
- One of the most influential microcomputers of all, the IBM PC, used the Intel 8088, a version of the 8086 with an eight-bit data bus (as mentioned above).
- The first Compaq Deskpro used an 8086 running at 7.14 MHz, (?) but was capable of running add-in cards designed for the 4.77 MHz IBM PC XT.
- An 8 MHz 8086 was used in the AT&T 6300 PC (built by Olivetti), an IBM PC-compatible desktop microcomputer. The M24 / PC 6300 has IBM PC/XT compatible 8-bit expansion slots, but some of them have a proprietary extension providing the full 16-bit data bus of the 8086 CPU (similar in concept to the 16-bit slots of the IBM PC AT, but different in the design details, and physically incompatible).
- The IBM PS/2 models 25 and 30 were built with an 8 MHz 8086.
- The Amstrad/Schneider PC1512, PC1640, PC2086, PC3086 and PC5086 all used 8086 CPUs at 8 MHz.
- The NEC PC-9801.
- The Tandy 1000 SL-series and RL machines used 8086 CPUs.
- The IBM Displaywriter word processing machine[8] and the Wang Professional Computer, manufactured by Wang Laboratories, also used the 8086.
- NASA used original 8086 CPUs on equipment for ground-based maintenance of the Space Shuttle Discovery until the end of the space shuttle program in 2011. This decision was made to prevent software regression that might result from upgrading or from switching to imperfect clones.[9]
- KAMAN Process and Area Radiation Monitors[10]

3.7 Notes

[1] Fewer TTL buffers, latches, multiplexers (although the amount of TTL logic was not drastically reduced). It also permitted the use of cheap 8080-family ICs, where the 8254 CTC, 8255 PIO, and 8259 PIC were used in the IBM PC design. In addition, it made PCB layout simpler and boards cheaper, as well as demanding fewer (1- or 4-bit wide) DRAM chips.

[2] using enhancement load PMOS logic (demanding 14V, achieving TTL-compatibility by having VCC at +5V and VDD at −9V)

[3] using non-saturated enhancement load NMOS logic (demanding a higher gate voltage for the load transistor-gates)

[4] made possible with depletion load nMOS logic (the 8085 was later made using HMOS processing, just like the 8086)

[5] Rev.0 of the instruction set and architecture was ready in about three months, according to Morse.

[6] Using rubylith, light boards, rulers, electric erasers, and a digitizer (according to Jenny Hernandez, member of the 8086 design team, in a statement made on Intel's web-page for its 25th birthday).

[7] 8086 used less microcode than many competitors' designs, such as the MC68000 and others

[8] Fast static RAMs in MOS technology (as fast as bipolar RAMs) was an important product for Intel during this period.

[9] CHMOS is Intel's name for CMOS circuits manufactured using processing steps very similar to HMOS.

[10] Other members of the design team were Peter A.Stoll and Jenny Hernandez.

[11] Some 80186 clones did change the shift value, but were never commonly used in desktop computers.

3.8 See also

- Transistor count
- iAPX, for the iAPX name

3.9 References

[1] "Microprocessor Hall of Fame". Intel. Archived from the original on 2007-07-06. Retrieved 2007-08-11.

[2] Official Intel iAPX 286 programmers' manual (page 1-1)

[3] Birth of a Standard: The Intel 8086 Microprocessor. Thirty years ago, Intel released the 8086 processor, introducing the x86 architecture that underlies every PC-Windows, Mac, or Linux-produced today, PC World, June 17, 2008

[4] Randall L. Geiger, Phillip E. Allen, Noel R. Strader *VLSI design techniques for analog and digital circuits*, McGraw-Hill Book Co., 1990, ISBN 0-07-023253-9, page 779 "Random Logic vs. Structured Logic Forms", illustration of use of "random" describing CPU control logic

[5] Intel 8008 to 8086 by Stephen P. Morse et al.

[6] Osborne 16 bit Processor Handbook (Adam Osborne & Gerry Kane) ISBN 0-931988-43-8

[7] *Microsoft Macro Assembler 5.0 Reference Manual*. Microsoft Corporation. 1987. "Timings and encodings in this manual are used with permission of Intel and come from the following publications: Intel Corporation. iAPX 86, 88, 186 and 188 User's Manual, Programmer's Reference, Santa Clara, Calif. 1986." (Similarly for iAPX 286, 80386, 80387.)

[8] Zachmann, Mark (August 23, 1982). "Flaws in IBM Personal Computer frustrate critic". *InfoWorld* (Palo Alto, CA: Popular Computing) **4** (33): 57–58. ISSN 0199-6649. "the IBM Displaywriter is noticeably more expensive than other industrial micros that use the 8086."

[9] For Old Parts, NASA Boldly Goes ... on eBay, May 12, 2002.

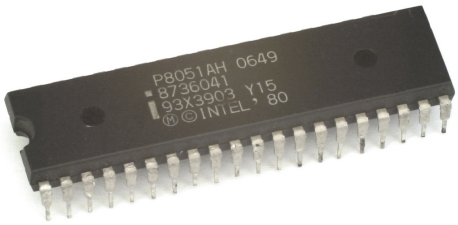
[10] Kaman Tech. Manual

3.10 External links

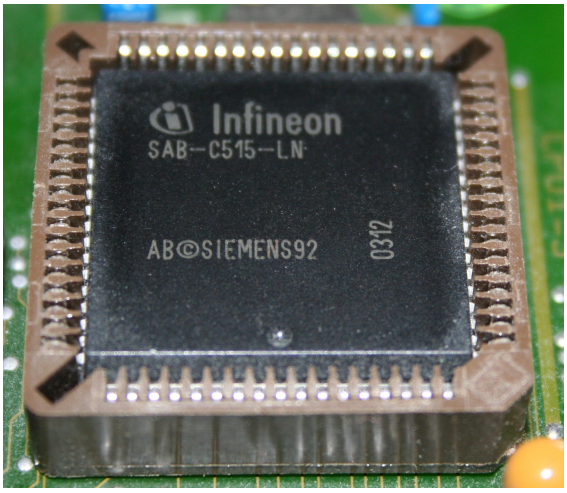
- Architecture-Of-8086 and pin at scanftree.com
- Intel datasheets
- List of 8086 CPUs and their clones at CPU-world.com
- 8086 Pinouts
- Maximum Mode Interface
- The 8086 User's manual October 1979 INTEL Corporation (PDF document)
- 8086 program codes using emu8086 (Version 4.08) Emulator
- Intel 8086/80186 emulator written in C, this file is part of a larger PC emulator

Chapter 4

Intel MCS-51



Intel P8051 microcontroller.



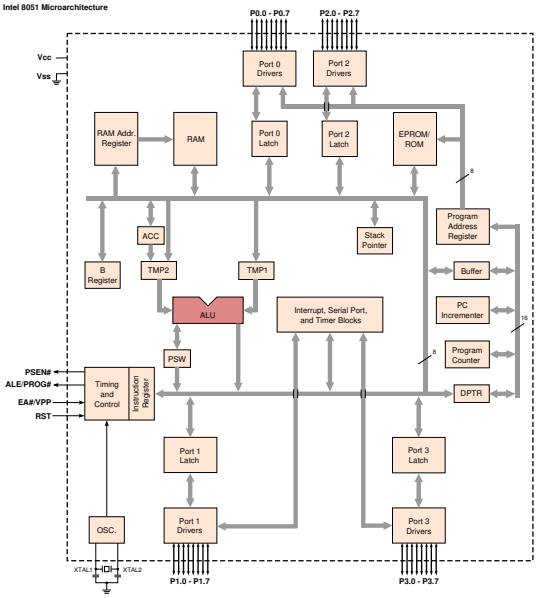
SAB-C515-LN by Infineon is based on the 8051

The **Intel MCS-51** (commonly referred to as **8051**) is a Harvard architecture, CISC instruction set, single chip microcontroller (µC) series which was developed by Intel in 1980 for use in embedded systems.[1] Intel's original versions were popular in the 1980s and early 1990s and enhanced binary compatible derivatives remain popular today.

Intel's original MCS-51 family was developed using NMOS technology, but later versions, identified by a letter C in their name (e.g., 80C51) used CMOS technology and consume less power than their NMOS predecessors. This made them more suitable for battery-powered devices.

The family was continued in 1996 with the enhanced 8-bit MCS-151 and the 8/16/32-bit MCS-251 family of binary compatible microcontrollers.[2] While Intel no longer manufactures the MCS-51, MCS-151 and MCS-251 family, enhanced binary compatible derivatives made by numerous vendors remain popular today. Some derivatives integrate a digital signal processor (DSP). In addition to these physical devices, several companies also offer MCS-51 derivatives as IP cores for use in FPGAs or ASICs designs.

4.1 Important features and applications



i8051 microarchitecture.

The 8051 architecture provides many functions (CPU, RAM, ROM, I/O, interrupt logic, timer, etc.) in a single package

- 8-bit ALU and Accumulator, 8-bit Registers (one 16-bit register with special move instructions), 8-bit data bus and 2×16-bit address bus/program

counter/data pointer and related 8/11/16-bit operations; hence it is mainly an 8-bit microcontroller

- Boolean processor with 17 instructions, 1-bit accumulator, 32 registers (4 bit-addressable 8-bit) and up to 144 special 1 bit-addressable RAM variables (18 bit-addressable 8-bit)[3]
- Multiply, divide and compare instructions
- 4 fast switchable register banks with 8 registers each (memory mapped)
- Fast interrupt with optional register bank switching
- Interrupts and threads with selectable priority[4]
- Dual 16-bit address bus – It can access 2 x 2^{16} memory locations – 64 KB (65,536 locations) each of RAM and ROM
- 128 bytes of on-chip RAM (IRAM)
- 4 KiB of on-chip ROM, with a 16-bit (64 KiB) address space (PMEM). Not included on 803X variants
- Four 8-bit bi-directional input/output port
- UART (serial port)
- Two 16-bit Counter/timers
- Power saving mode (on some derivatives)

One feature of the 8051 core is the inclusion of a boolean processing engine which allows bit-level boolean logic operations to be carried out directly and efficiently on select internal registers and select RAM locations. This feature helped cement the 8051's popularity in industrial control applications because it reduced code size by as much as 30%. Another feature is the inclusion of four bank selectable working register sets which greatly reduce the amount of time required to complete an interrupt service routine. With a single instruction the 8051 can switch register banks as opposed to the time consuming task of transferring the critical registers to the stack or designated RAM locations. These registers also allowed the 8051 to quickly perform a context switch.

Once a UART, and a timer if necessary, has been configured, the programmer needs only write a simple interrupt routine to refill the *send* shift register whenever the last bit is shifted out by the UART and/or empty the full *receive* shift register (copy the data somewhere else). The main program then performs serial reads and writes simply by reading and writing 8-bit data to stacks.

4.1.1 Derivate features

As of 2013, new derivates are still developed by many major chipmakers, and major compiler suppliers such as IAR Systems, Keil and Altium Tasking continuously release updates.

MCS-51 based microcontrollers typically include one or two UARTs, two or three timers, 128 or 256 bytes of internal data RAM (16 bytes of which are bit-addressable), up to 128 bytes of I/O, 512 bytes to 64 KB of internal program memory, and sometimes a quantity of extended data RAM (ERAM) located in the external data space. The original 8051 core ran at 12 clock cycles per machine cycle, with most instructions executing in one or two machine cycles. With a 12 MHz clock frequency, the 8051 could thus execute 1 million one-cycle instructions per second or 500,000 two-cycle instructions per second. Enhanced 8051 cores are now commonly used which run at six, four, two, or even one clock per machine cycle, and have clock frequencies of up to 100 MHz, and are thus capable of an even greater number of instructions per second. All Silicon Labs, some Dallas and a few Atmel devices have single cycle cores.

8051 variants may include built-in reset timers with brown-out detection, on-chip oscillators, self-programmable Flash ROM program memory, built-in external RAM, extra internal program storage, bootloader code in ROM, EEPROM non-volatile data storage, I^2C, SPI, and USB host interfaces, CAN or LIN bus, ZigBee or Bluetooth radio modules, PWM generators, analog comparators, A/D and D/A converters, RTCs, extra counters and timers, in-circuit debugging facilities, more interrupt sources, extra power saving modes, etc.

In many engineering schools the 8051 microcontroller is used in introductory microcontroller courses.

4.2 Memory architecture

The MCS-51 has four distinct types of memory – internal RAM, special function registers, program memory, and external data memory.

Internal RAM (IRAM) is located from address 0 to address 0xFF. IRAM from 0x00 to 0x7F can be accessed directly. IRAM from 0x80 to 0xFF must be accessed indirectly, using the @R0 or @R1 syntax, with the address to access loaded in R0 or R1. The 128 bits at IRAM locations 0x20–0x2F are bit-addressable.

Special function registers (SFR) are located in the same address space as IRAM, at addresses 0x80 to 0xFF, and are accessed directly using the same instructions as for the lower half of IRAM. They can *not* be accessed indirectly via @R0 or @R1. 16 of the SFRs are also bit-addressable.

Program memory (PMEM, though less common in usage than IRAM and XRAM) is up to 64 KiB of read-only memory, starting at address 0 in a separate address space. It may be on- or off-chip, depending on the particular model of chip being used. Program memory is read-only, though some variants of the 8051 use on-chip flash memory and provide a method of re-programming the memory in-system or in-application. In addition to code, it is possible to store read-only data in program memory, accessed by the MOVC A, @DPTR instruction. Data is fetched from the address specified in the 16-bit special function register DPTR.

External data memory (XRAM) is a third address space, also starting at address 0. It can also be on- or off-chip; what makes it "external" is that it must be accessed using the MOVX (Move eXternal) instruction. Many variants of the 8051 include the standard 256 bytes of IRAM plus a few KB of XRAM on the chip.

The 8051 is designed as a strict Harvard architecture. The 8051 can only execute code fetched from program memory. The 8051 does not have any instruction to write to program memory. Most 8051 systems respect this distinction, and so are unable to download and directly execute new programs. The strict Harvard architecture has the advantage of making such systems immune to most forms of malware. Some 8051 systems have (or can be modified to have) some "dual-mapped" RAM, making them act somewhat more like Princeton architecture. This (partial) Princeton architecture has the advantage of making it possible for a Forth boot loader running on the 8051 to write new native code to RAM and then execute it, leading to faster incremental and interactive programming cycles than strict Harvard systems.[5][6]

4.3 Registers

The only register on an 8051 that is not memory-mapped is the 16-bit program counter PC. This specifies the address of the next instruction to execute. Relative branch instructions supply an 8-bit signed offset which is added to the PC.

The following registers are memory-mapped into the special function register space:

- (0x81) Stack pointer SP. This is an 8-bit register used by subroutine call and return instructions. The stack grows upward; the SP is incremented before pushing, and decremented after popping a value.
- (0x82–83) Data pointer DP. This is a 16-bit register that is used for accessing PMEM and XRAM.
- (0xD0) Program status word PSW. This contains important status flags:
  - --- PSW.0: P Parity. Gives the parity (modulo-2 sum of the bits of) the most recent ALU result.
  - --- PSW.1: UD User Defined. For general software use, not otherwise used by hardware.
  - --- PSW.2: OV Overflow flag. Set when addition produces a signed overflow.
  - --- PSW.3: RS0 Register select 0. The low-order bit of the register bank. Set when banks at 0x08 or 0x18 are in use.
  - --- PSW.4: RS1 Register select 1. The high-order bit of the register bank. Set when banks at 0x10 or 0x18 are in use.
  - --- PSW.5: F0 Flag 0. For general software use, not otherwise used by hardware.
  - --- PSW.6: AC auxiliary carry. Set when addition produces a carry from bit 3 to bit 4.
  - --- PSW.7: C Carry bit.
- (0xE0) Accumulator A. This register is used by most instructions.
- (0xF0) B register. This is used as an extension to the accumulator for multiply and divide instructions.

In addition, there are 8 general purpose registers R0–R7, mapped to IRAM between 0x00 and 0x1F. Only 8 bytes of that range are used at any given time, determined by the bank select bits in the PSW.

256 single bits are directly addressable. These are the 16 IRAM locations from 0x20–0x2F, and the 16 special function registers 0x80, 0x88, 0x90, …, 0xF8.

Note that the PSW does not contain the common N (negative) and Z (zero) flags. Instead, because the accumulator is a bit-addressible SFR, it is possible to branch on individual bits of it, including the msbit. There is also an instruction to jump if the accumulator is zero or non-zero.

4.4 Instruction set

Instructions are all 1 to 3 bytes long, consisting of an initial opcode byte, followed by up to 2 bytes of operands.

There are 16 basic ALU instructions that operate between the accumulator and a second operand, specified using one of the following addressing modes:

- Register direct, R0–R7 (opcodes *x*8–*x*F)
- Register indirect, @R0 or @R1 (opcodes *x*6 and *x*7)
- Memory direct, specifying an IRAM or SFR location (opcodes *x*5, followed by 1 byte of address)
- Immediate, specifying an 8-bit constant (opcodes *x*4, followed by 1 byte of data)

The instructions are as follows. Not all support all addressing modes; the immediate mode in particular is sometimes nonsensical:

- 0*y* INC *operand*: Increment the specified operand. Opcode 04 specifies "INC A"
- 1*y* DEC *operand*: Decrement the specified operand. Opcode 14 specifies "DEC A"
- 2*y* ADD A,*operand*: Add the operand to the accumulator A.
- 3*y* ADDC A,*operand*: Add the operand, plus the C bit, to the accumulator.
- 4*y* ORL A,*operand*: Logical OR the operand into the A register.
- 5*y* ANL A,*operand*: Logical AND the operand into the A register.
- 6*y* XRL A,*operand*: Logical exclusive-OR the operand into the A register.
- 7*y* MOV *operand*,#*data*: Move immediate data to the operand. Opcode 74 specifies "MOV A,#*data*.
- 8*y* MOV *address*,*operand*: Move data to an IRAM or SFR register.
- 9*y* SUBB A,*operand*: Subtract the operand from the accumulator, with borrow. Note there is no subtract *without* borrow.
- A*y* MOV *operand*,*address*: Move data from an IRAM or SFR register. Opcodes A4 and A5 are not used.
- B*y* CJNE *operand*,#*data*,*offset*: Compare *operand* to the immediate *data*, and branch to PC+*address* if not equal. Opcodes B4 and B5 perform CJNE A,*operand*,*offset*, for memory direct and immediate operands. Note there is no "compare and jump if equal" instruction.
- C*y* XCH A,*operand*: Exchange (swap) the accumulator and the operand. Opcode C4 is not used.
- D*y* DJNZ *operand*,*offset*: Decrement the operand, and branch to PC+*offset* if the result is non-zero. Opcodes D4, D6, and D7 are not used.
- E*y* MOV A,*operand*: Move operand to the accumulator. Opcode E4 is not used. (Use opcode 74 instead.)
- F*y* MOV *operand*,A: Move accumulator to the operand. Opcode F4 is not used.

Only the ADD, ADDC and SUBB instructions set PSW flags. The INC, DEC, and logical instructions do not. The CJNE instructions modify the C bit only, to the borrow that results from *operand1*−*operand2*.

The 32 opcodes 0x00–0x3F, plus the few opcodes not used in the above range, are used for other instructions with more limited operand-specification capabilities.

One of the reasons for the 8051's popularity is its range of operations on single bits. Bits are always specified by absolute addresses; there is no register-indirect or indexed addressing. Instructions that operate on single bits are:

- SETB *bit*, CLR *bit*, CPL *bit*: Set, clear, or complement the specified bit
- JB *bit*,*offset*: Jump if bit set
- JNB *bit*,*offset*: Jump if bit clear
- JBC *bit*,*offset*: Jump if bit set, and clear bit
- MOV C,*bit*, MOV *bit*,C: Move the specified bit to the carry bit, or vice-versa
- ORL C,*bit*, ORL C,/*bit*: OR the bit (or its complement) to the carry bit
- ANL C,*bit*, ANL C,/*bit*: AND the bit (or its complement) to the carry bit
- XRL C,*bit*, XRL C,/*bit*: Exclusive-OR the bit (or its complement) to the carry bit

Although most instructions require that one operand is the accumulator or an immediate constant, it is possible to perform a MOV directly between two internal RAM locations.

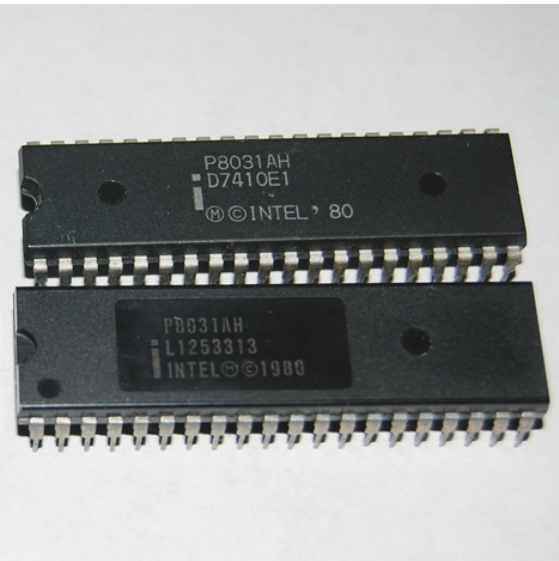
4.5 Programming

Main article: 8051 compiler

There are various high-level programming language compilers for the 8051. Several C compilers are available for the 8051, most of which allow the programmer to specify where each variable should be stored in its six types of memory, and provide access to 8051 specific hardware features such as the multiple register banks and bit manipulation instructions. There are many commercial C compilers. SDCC is a popular open source C compiler. Other high level languages such as C++, Forth, BASIC, Pascal/Object Pascal, PL/M and Modula-2 are available for the 8051, but they are less widely used than C and assembly.

Because IRAM, XRAM, and PMEM (**read only**) all have an address 0, C compilers for the 8051 architecture provide compiler-specific pragmas or other extensions to indicate where a particular piece of data should be stored (i.e. constants in PMEM or variables needing fast access in IRAM). Since data could be in one of three memory spaces, a mechanism is usually provided to allow determining to which memory a pointer refers, either by constraining the pointer type to include the memory space, or by storing metadata with the pointer.

4.6 Related processors



Intel 8031 processors

The 8051's predecessor, the 8048, was used in the keyboard of the first IBM PC, where it converted keypresses into the serial data stream which is sent to the main unit of the computer. The 8048 and derivatives are still used today for basic model keyboards.

The *8031* was a cut down version of the original Intel 8051 that did not contain any internal program memory (ROM). To use this chip, external ROM had to be added containing the program that the 8031 would fetch and execute. An 8051 chip could be sold as a ROM-less 8031, as the 8051's internal ROM is disabled by the normal state of the EA pin in an 8031-based design. A vendor might sell an 8051 as an 8031 for any number of reasons, such as faulty code in the 8051's ROM, or simply an oversupply of 8051s and undersupply of 8031s.

The *8052* was an enhanced version of the original 8051 that featured 256 bytes of internal RAM instead of 128 bytes, 8 KB of ROM instead of 4 KB, and a third 16-bit timer. The *8032* had these same features except for the internal ROM program memory. Most modern "8051-compatible" microcontrollers include these features.

Intel discontinued its MCS-51 product line in March 2007;[7] however, there are plenty of enhanced 8051 products or silicon intellectual property added regularly from other vendors.

The 80C537 and 80C517 are CMOS versions, designed for the automotive industry. Enhancements mostly include new peripheral features and expanded arithmetic instructions. The 80C517 has fail-safe mechanisms, analog signal processing facilities and timer capabilities and 8 KB on-chip program memory. Other features include:

- 256 byte on-chip RAM
- 256 directly addressable bits
- External program and data memory expandable up to 64 KB
- 8-bit A/D converter with 12 multiplexed inputs
- Arithmetic unit can make division, multiplication, shift and normalize operations
- Eight data pointers instead of one for indirect addressing of program and external data memory
- Extended watchdog facilities
- Nine ports
- Two full-duplex serial interfaces with own baud rate generators
- Four priority level interrupt systems, 14 interrupt vectors
- Three power saving modes

4.6.1 Derivate vendors

Current vendors of MCS-51 compatible processors include more than 20 independent manufacturers including Atmel, Infineon Technologies (formerly Siemens AG), Maxim Integrated Products (via its Dallas Semiconductor subsidiary), NXP (formerly Philips Semiconductor), Microchip Technology, Nuvoton (formerly Winbond), ST Microelectronics, Silicon Laboratories (formerly Cygnal), Texas Instruments, Ramtron International, Silicon Storage Technology, Cypress Semiconductor and Analog Devices.[8]

ICs or IPs compatible with the MCS-51 have been developed by

- Acer Labs
- Actel
- Aeroflex UTMC
- Altium
- Analog Devices
- ASIX
- Atmel
- AustriaMicroSystems
- AXSEM
- California Eastern Laboratories (CEL)
- Cast
- CML Microcircuits

- CORERIVER
- Cybernetic Micro Systems
- CybraTech
- Cypress
- Daewoo
- Dallas Semiconductor
- Digital Core Design
- Dolphin
- Domosys
- easyplug
- EnOcean
- Evatronix
- Fairchild
- Genesis Microchip
- Genesys Logic
- Goal Semiconductor
- Handshake Solutions
- Honeywell
- Hynix Semiconductor
- Infineon
- InnovASIC
- Intel
- ISSI
- Lapis Semiconductor
- Maxim (Dallas Semiconductor)
- Megawin
- Mentor Graphics
- Micronas
- Microsemi
- MXIC
- Myson Technology
- Nordic Semiconductor
- Nuvoton (Winbond)
- NXP (founded by Philips)
- OKI
- Oregano Systems
- PalmChip
- Prolific
- Radio Pulse
- Ramtron
- RDC
- RDC Semiconductor
- Sanyo
- Sharp
- Sigma Designs
- Silicon Laboratories (Cygnal)
- Siliconians
- SMSC
- SST
- STMicroelectronics
- SyncMOS
- Synopsys
- Syntek Semiconductor
- Tekmos
- Teridian Semiconductor
- Texas Instruments
- Tezzaron Semiconductor
- Triscend
- Vitesse
- Yitran
- Zensys
- Zilog
- Zylogic Semiconductor

4.7 Use as intellectual property

Today, 8051s are still available as discrete parts, but they are mostly used as silicon intellectual property cores. Available in high-level language source code (VHDL or Verilog) or FPGA netlist forms, these cores are typically integrated within embedded systems, in products ranging from USB flash drives to washing machines to complex wireless communication systems on a chip. Designers use 8051 silicon IP cores, because of the smaller size, and lower power, compared to 32 bit processors like ARM M series, MIPS and BA22. Modern 8051 cores are faster

than earlier packaged versions. Design improvements have increased 8051 performance while retaining compatibility with the original MCS 51 instruction set. The original Intel 8051 ran at 12 clock cycles per machine cycle, and most instructions executed in one or two machine cycles. A typical maximum clock frequency of 12 MHz meant these old 8051s could execute one million single-cycle instructions, or 500,000 two-cycle instructions, per second. In contrast, enhanced 8051 silicon IP cores now run at one clock cycle per machine cycle, and have clock frequencies of up to 450 MHz. That means an 8051-compatible processor can now execute 450 million instructions per second.

4.8 MCU based on 8051

- Atmel: AT89C51, AT89S51, AT83C5134
- Infineon: XC800
- NXP: NXP700 and NXP900 series
- Silicon Labs: C8051 series
- Texas Instruments CC111x, CC24xx and CC25xx families of RF SoCs

4.9 Digital signal processor (DSP) variants

Several variants with an additional 16-bit digital signal processor (DSP) (for example for MP3 or OGG coding/decoding) with up to 675 million instructions per second (MIPS)[9] and integrated USB 2.0 interface[10] or as intellectual property[11] exist.

4.10 Enhanced 8-bit binary compatible microcontroller: MCS-151 family

1996 Intel announced the MCS-151 family, an up to 6 times faster variant.[2] 8051 fully binary and instruction set compatible, but with pipelined CPU, 16 bit internal code bus and 6x speed. The MCS-151 family was also discontinued by Intel, but is widely available in binary compatible and partly enhanced variants.

4.11 8/16/32-bit binary compatible microcontroller: MCS-251 family

The 80251 8/16/32-bit microcontroller with 16MB (24-bit) address-space and 6 times faster instruction cycle was introduced by Intel in 1996.[2][12] It can perform as an 8-bit 8051, has 24-bit external address space which is 16-bit wide segmented and 32-bit ALU with mostly 8/16/32-bit wide data instructions (also Boolean processor with special registers/memory) and a large CISC instruction set, 40 8/16/32-bit registers with 8 8-bit registers in 4 times fast switching memory banks (maximum 512 addressable 8-bit special registers).

It features extended instructions[13] - see also the programmer's guide[14] - and later variants with higher performance,[15] also available as intellectual property (IP).[16] It is 3-stage pipelined. The MCS-251 family was also discontinued by Intel, but is widely available in binary compatible and partly enhanced variants from many manufacturers.

4.12 See also

- SDK-51 System Design Kit
- DS80C390

4.13 References

[1] John Wharton: *An Introduction to the Intel MCS-51 Single-Chip Microcomputer Family*, Application Note AP-69, May 1980, Intel Corporation.

[2] Intel MCS 151 and MCS 251 Microcontrollers

[3] John Wharton: *Using the Intel MCS-51 Boolean Processing Capabilities* Application Note AP-70, May 1980, Intel Corporation.

[4] 8051 Tutorial: Interrupts

[5] Bradford J. Rodriguez. "CamelForth/8051".

[6] Brad Rodriguez. "Moving Forth Part 7: CamelForth for the 8051".

[7] Intel bows out, discontinues MCS-51.

[8] http://www.analog.com/static/imported-files/data_sheets/ADUC832.pdf

[9] TI Delivers new low-cost, high-performance audio DSP for Home and Car w/ 8051

[10] Atmel AT85C51SND3 Audio DSP Data Sheet with USB 2.0

[11] Integration of 8051 With DSP in Xilinx FPGA

[12] The 8051 microcontroller By Kenneth J Ayala Google books

[13] Temic TSC80251 Architecture

[14] Atmel TSC80251 Programmers Guide

[15] DQ80251 32bit Microcontroller DCD

[16] R80251XC 32bit Microcontroller Evatronix

4.14 Further reading

Books

- *The 8051 Microcontroller : A Systems Approach*; Mazidi, McKinlay, Mazidi; 648 pages; 2012; ISBN 978-0135080443.
- *C and the 8051*; 4th Edition; Thomas Schultz; 464 pages; 2008; ISBN 978-0978399504.
- *The 8051/8052 Microcontroller : Architecture, Assembly Language, and Hardware Interfacing*; Craig Steiner; 348 pages; 2005; ISBN 978-1581124590.
- *8051 Microcontrollers : Hardware, Software and Applications*; Calcutt, Cowan, Parchizadeh; 329 pages; 2000; ISBN 978-0340677070.
- *The Microcontroller Idea Book : Circuits, Programs, and Applications featuring the 8052-BASIC Microcontroller*; Jan Axelson; 277 pages; 1994; ISBN 978-0965081900.
- Payne, William (December 19, 1990) [1990]. *Embedded Controller FORTH for the 8051 Family* (hardcover). Boston: Academic Press. p. 528. ISBN 978-0-12-547570-9.

Intel

- *MCS-51 Microcontroller Family User's Manual*; Intel; 1994; publication number 121517.
- *MCS-51 Macro Assembler User's Guide*; Intel; publication number 9800937.
- *8-Bit Embedded Controllers*; Intel; 1991; publication number 270645-003.
- *Microcontroller Handbook*; Intel; 1984; publication number 210918-002.
- *8051 Microcontroller Preliminary Architectural Specification and Functional Description*; Intel; 44 pages; 1980.

4.15 External links

- Complete tutorial for 8051 microcontrollers
- Instruction set of 8051 microcontroller
- the source website for tutorials and simulator for 8051
- Basic 8051 Interfacing Circuits
- Open source VHDL 8051 implementation (Oregano Systems)

Chapter 5

Motorola 6800

Motorola MC6800 microprocessor

The **6800** (extquotedbl*sixty-eight hundred* extquotedbl) was an 8-bit microprocessor designed and first manufactured by Motorola in 1974. The MC6800 microprocessor was part of the M6800 Microcomputer System that also included serial and parallel interface ICs, RAM, ROM and other support chips. A significant design feature was that the M6800 family of ICs required only a single five-volt power supply at a time when most other microprocessors required three voltages. The M6800 Microcomputer System was announced in March 1974 and was in full production by the end of that year.[1][2]

The 6800 architecture and instruction set were influenced by the then popular Digital Equipment Corporation PDP-11 mini computer.[3][4] The 6800 has a 16-bit address bus that could directly access 64 KB of memory and an 8-bit bi-directional data bus. It has 72 instructions with seven addressing modes for a total of 197 opcodes. The original MC6800 could have a clock frequency of up to 1 MHz. Later versions had a maximum clock frequency of 2 MHz.[5][6]

In addition to the ICs, Motorola also provided a complete assembly language development system. The customer could use the software on a remote timeshare computer or on an in-house minicomputer system. The Motorola EXORciser was a desktop computer built with the M6800 ICs that could be used for prototyping and debugging new designs. An expansive documentation package included datasheets on all ICs, two assembly language programming manuals, and a 700-page application manual that showed how to design a point-of-sale computer terminal.[7]

The 6800 was popular in computer peripherals, test equipment applications and point-of-sale terminals. The MC6802, introduced in 1977, included 128 bytes of RAM and an internal clock oscillator on chip. The MC6801 and MC6805 included with RAM, ROM and I/O on a single chip were popular in automotive applications.

5.1 Motorola's history in semiconductors

Main article: Motorola § History

Galvin Manufacturing Corporation was founded in 1928;

Motorola began making semiconductors in the 1950s.

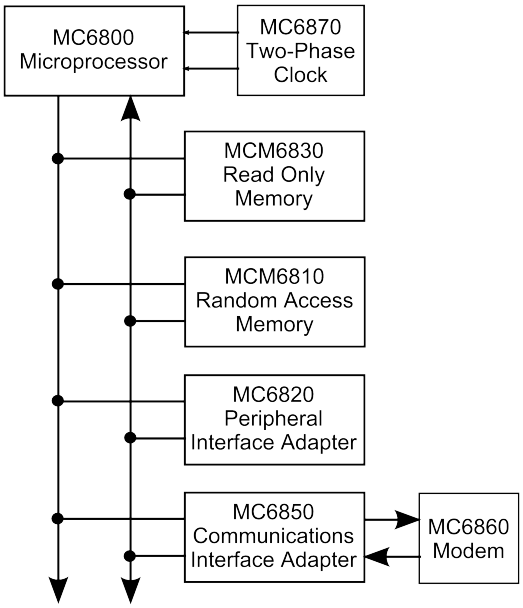
the company name was changed to Motorola in 1947. They began commercial production of transistors at a new US$1.5 million facility in Phoenix in 1955.[8]

Motorola's transistors and integrated circuits were used in-house for their communication, military, automotive and consumer products and they were also sold to other companies. By 1973 the Semiconductor Products Division (SPD) had sales of $419 million and was the second largest semiconductor company after Texas Instruments.[9]

In the early 1970s Motorola started a project that developed their first microprocessor, the MC6800. This was followed by single-chip microcontrollers such as the MC6801 and MC6805.

In 1999 Motorola spun off their analog IC, digital IC and transistor business to ON Semiconductor of Phoenix Arizona. In 2004 they spun off their microprocessor business to Freescale Semiconductor of Austin, Texas.

5.2 Development team



Block diagram of a M6800 microcomputer system

Motorola did not chronicle the development of the 6800 microprocessor the way that Intel did for their microprocessors. In 2008 the Computer History Museum interviewed four members of the 6800 microprocessor design team. Their recollections can be confirmed and expanded by magazine and journal articles written at the time.

The Motorola microprocessor project began in 1971 with a team composed of designer Tom Bennett, engineering director Jeff LaVell, product marketer Link Young and systems designers Mike Wiles, Gene Schriber and Doug Powell.[10] They were all located in Mesa, Arizona. By the time the project was finished, Bennett had 17 chip designers and layout people working on five chips. LaVell had 15 to 20 system engineers and there was another applications engineering group of similar size.[11]

Tom Bennett had a background in industrial controls and had worked for Victor Comptometer in the 1960s designing the first electronic calculator to use MOS ICs, the Victor 3900.[12] In May 1969 Ted Hoff showed Bennett early diagrams of the Intel 4004 to see if it would meet their calculator needs. Bennett joined Motorola in 1971 to design calculator ICs. He was soon assigned as the chief architect of the microprocessor project that produced the 6800.[13] Others have taken credit for designing the 6800. In September 1975 Robert H. Cushman, *EDN* magazine's microprocessor editor, interviewed Chuck Peddle about MOS Technology's new 6502 microprocessor. Cushman then asked "Tom Bennett, master architect of the 6800," to comment about this new competitor.[14] After the 6800 project Bennett worked on automotive applications and Motorola became a major supplier of microprocessors used in automobiles.

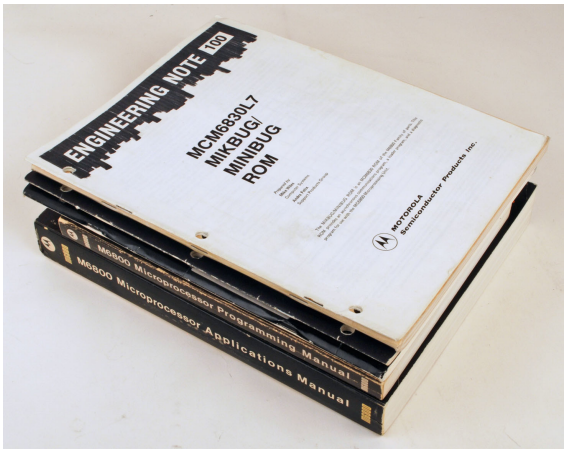
Jeff LaVell joined Motorola in 1966 and worked in the computer industry marketing organization. Jeff had previously worked for Collins Radio on their C8500 computer that was built with small scale ECL ICs. In 1971 he led a group that examined the needs of their existing customers such as Hewlett Packard, National Cash Register, Control Data Corporation (CDC), and Digital Equipment Corporation. They would study the customer's products and tried to identify functions that could be implemented in larger integrated circuits at a lower cost. The result of the survey was a family of 15 building blocks; each could be implemented in an integrated circuit.[11] Some of these blocks were implemented in the initial M6800 release and more were added over the next few years. To evaluate the 6800 architecture while the chip was being designed, Jeff's team built an equivalent circuit using 451 small scale TTL ICs on five 10 by 10 inch (25 by 25 cm) circuit boards. Later they reduced this to 114 ICs on one board by using ROMs and MSI (medium scale integration) logic devices.[15]

John Buchanan was a memory designer at Motorola when Bennett asked him to design a voltage doubler for the 6800. Typical n-channel MOS IC's required three power supplies: −5 volts, +5 volts and +12 volts. The M6800 family was to use only one, +5 volts. It was easy to eliminate the −5 volt supply but the MOS transistors needed a supply of 10 to 12 volts. This on-chip voltage doubler would supply the higher voltage and Buchanan did the circuit design, analysis and layout for the 6800 microprocessor. He received patents on the voltage doubler and the 6800 chip layout.[16][17] Rod Orgill assisted Buchanan with analyses and 6800 chip layout. Later Orgill would design the MOS Technology 6501 microprocessor that was socket compatible with the 6800.

Bill Lattin joined Motorola in 1969 and his group provided the computer simulation tools for characterizing the new MOS circuits in the 6800. Lattin and Frank Jenkins had both attended UC Berkeley and studied computer

circuit simulators under Donald Pederson, the designer of the SPICE circuit simulator.[18] Motorola's simulator, MTIME, was an advanced version of the TIME circuit simulator that Jenkins had developed at Berkeley. The group published a technical paper, "MOS-device modeling for computer implementation" in 1973 describing a "5-V single-supply n-channel technology" operating at 1 MHz. They could simulate a 50 MOSFET circuit on an IBM 370/165 mainframe computer.[19] In November 1975, Lattin joined Intel to work on their next generation microprocessor.[20]

Bill Mensch joined Motorola in 1971 after graduating from the University of Arizona. He had worked several years as an electronics technician before earning his BSEE degree. The first year at Motorola was a series of three-month rotations through four different areas. Mensch did a flowchart for a modem that would become the 6860. He also worked the application group that was defining the M6800 system. After this training year, he was assigned to the 6820 Peripheral Interface Adapter (PIA) development team. Mensch was a major contributor to the design of this chip and received a patent on the IC layout[21] and was named as a co-inventor of seven other M6800 system patents.[22] Later Mensch would design the MOS Technology 6502 microprocessor.

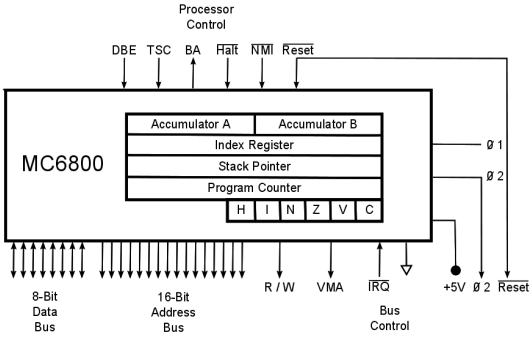


MIKBUG was part of the extensive M6800 microcomputer support developed by Motorola's Application Engineering Group.

Mike Wiles was a design engineer in Jeff LaVell's group and made numerous customer visits with Tom Bennett during 6800 product definition phase. He is listed as an inventor on eighteen 6800 patents but is best known for a computer program, MIKBUG.[23] This was a monitor for a 6800 computer system that allowed the user to examine the contents of RAM and to save or load programs to tape. This 512 byte program occupied half of an MCM6830 ROM.[24] This ROM was used in the Motorola MEK6800 design evaluation kit and early hobby computer kits.[25] Wiles stayed with Motorola, moved to Austin and helped design the MC6801 microcontroller that was released in 1978.[26]

Chuck Peddle joined the design team in 1973 after the 6800 processor design was done but he contributed to overall system design and to several peripheral chips, particularly the 6820 (PIA) parallel interface.[27] Peddle is listed as an inventor on sixteen Motorola patents, most have six or more co-inventors.[28] Like the other engineers on the team, Peddle visited potential customers and solicited their feedback. Peddle and John Buchanan built one of the earliest 6800 demonstration boards.[29] In August 1974 Chuck Peddle left Motorola and joined a small semiconductor company in Pennsylvania, MOS Technology. There he led the team that designed the 6500 microprocessor family.

5.3 MC6800 microprocessor design



A Motorola MC6800 microprocessor registers and I/O lines

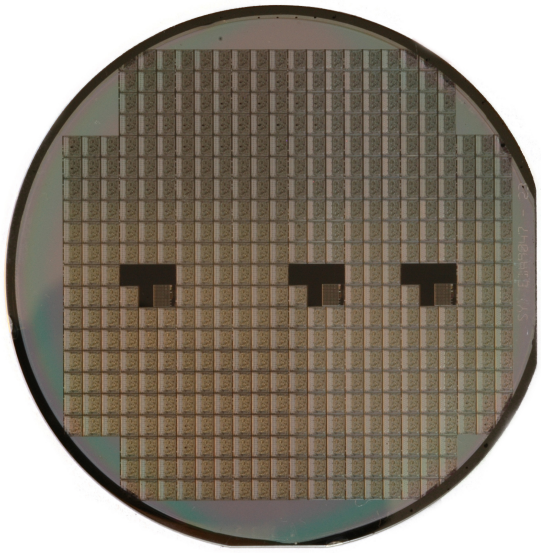
The Motorola 6800 and the Intel 8080 were designed at the same time and were similar in function. The 8080 was a superset of the Intel 8008, which was based on the Datapoint 2200 processor. The 6800 architecture was modeled after the DEC PDP-11 processor.[4] Both were TTL compatible, had an 8-bit bidirectional data bus, a 16-bit stack pointer, a 16-bit address bus that could address 64 KB of memory, and came in a 40-pin DIP package. The 6800 had two accumulators and a 16-bit index register. The direct addressing mode allowed fast access to the first 256 bytes of memory. I/O devices were addressed as memory so there were no special I/O instructions. The 8080 had more internal registers and instructions for dedicated I/O ports. When the 8080 was reset, the program counter was cleared and the processor started at memory location 0000. The 6800 loaded the program counter from the highest address and started at the memory location stored there.[30] The 6800 had a three-state control that would disable the address bus to allow another device direct memory access. A disk controller could therefore transfer data into memory with no load on the processor. It was even possible to have two 6800 processors access the same memory.[31] However, in practice systems of such complexity usually required the use of external bus transceivers to drive the system bus; in such circuits the on-processor bus control was disabled entirely in favor

of using the similar capabilities of the bus transceiver.[32] In contrast, the 6802 dispensed with this on-chip control entirely in order to free up pins for other functions in the same 40-pin package as the 6800, but this functionality could still be achieved using an external bus transceiver.

MOS ICs typically used dual clock signals (a two-phase clock) in the 1970s. These were generated externally for both the 6800 and the 8080.[33] The next generation of microprocessors incorporated the clock generation on chip. The 8080 had a 2 MHz clock but the processing throughput was similar to the 1 MHz 6800, since the 8080 required more clock cycles to execute a processor instruction than the 6800. The 6800 had a minimum clock rate of 100 kHz, while the 8080 had no lower limit and could be halted (effectively a 0 Hz clock speed). Higher-speed versions of both microprocessors were released by 1976.[34]

Other divisions in Motorola developed components for the M6800 family. The Components Products Department designed the MC6870 two-phase clock IC, and the Memory Products group provided a full line of ROMs and RAMs. The CMOS group's MC14411 Bit Rate Generator provided a 75 to 9600 baud clock for the MC6850 serial interface. The buffers for address and data buses were standard Motorola products. Motorola could supply every IC, transistor, and diode necessary to build an MC6800-based computer.

5.4 MOS ICs



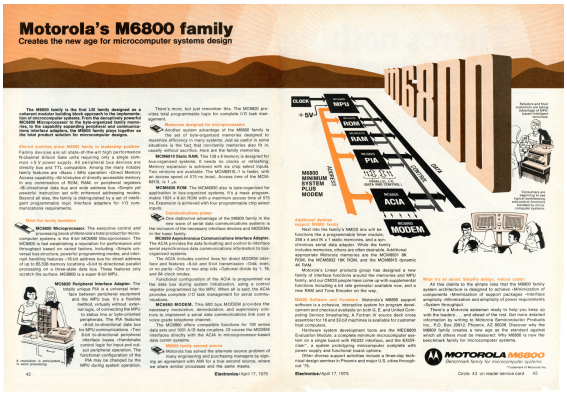
A silicon wafer holding many integrated circuit chips

The first-generation metal–oxide–semiconductor (MOS) chips used p-channel field-effect transistors, known as p-channel MOSFETs (p-channel describes the configuration of the transistor). These ICs were used in calculators and in the first microprocessor, the Intel 4004. They were easy to produce but were slow and difficult to interface to the popular TTL digital logic ICs. An n-channel MOS integrated circuit could operate two or three times faster and was compatible with TTL. They were much more difficult to produce because of an increased sensitivity to contamination that required an ultra clean production line and meticulous process control.[35] Motorola did not have an n-channel MOS production capability and had to develop one for the 6800 family.

Motorola's n-channel MOS test integrated circuits were complete in late 1971 and these indicated the clock rate would be limited to 1 MHz. These used "enhancement-mode" MOS transistors. There was a newer fabrication technology that used "depletion-mode" MOS transistors as loads, which would allow smaller and faster circuits (this was also known as depletion-load nMOS). The "depletion-mode" processing required extra steps so Motorola decided to stay with "enhancement-mode" for the new single-supply-voltage design. The 1 MHz clock rate meant the chip designers would have to come up with several architectural innovations to speed up the microprocessor throughput.[13] These resulting circuits were faster but required more area on the chip.[36]

In the 1970s, semiconductors were fabricated on 3 inch (75 mm) diameter silicon wafers. Each wafer could produce 100 to 200 integrated circuit chips or dies. The technical literature would state the length and width of each chip in "mils" (0.001 inch). The Intel 8080 microprocessor chip size was 164 mils x 191 mils (4.1 mm by 4.9 mm).[37] The current industry practice is to state the chip area so the size of the 8080 chip would be 19.7 mm^2.

Processing wafers required multiple steps and flaws would appear at various locations on the wafer during each step. The larger the chip the more likely it would encounter a defect. The percentage of working chips or yield began to decline for chips larger than 160 mils (4 mm) on a side. The target size for the 6800 was 180 mils (4.6 mm) on each side but the final size was 212 mils (5.4 mm) with an area of (29.0 mm^2). At 180 mils, a 3-inch (76 mm) wafer will hold about 190 chips, 212 mils reduces that to 140 chips. At this size the yield may be 20% or 28 chips per wafer.[38][39] The Motorola 1975 annual report highlights the new MC6800 microprocessor but has several paragraphs on the "MOS yield problems."[9] The yield problem was solved with design revision started in 1975 to use depletion mode in the M6800 family devices. The 6800 die size was reduced to 160 mils (4 mm) per side with an area of 16.5 mm^2. This also allowed faster clock speeds, the MC68A00 would operate at 1.5 MHZ and the MC68B00 at 2.0 MHz. The new parts were available in July 1976.[26][40]



An early advertisement for the Motorola's M6800 family microcomputer system

5.5 M6800 family introduction

The March 7, 1974 issue of *Electronics* had a two-page story on the Motorola MC6800 microprocessor along with the MC6820 Peripheral Interface Adapter, the MC6850 communications interface adapter, the MCM6810 128 byte RAM and the MCM6830 1024 byte ROM.[1] This was followed by an eight-page article in the April 18, 1974 issue authored by the Motorola design team.[41] This issue also had an article introducing the Intel 8080[42]

The Intel 8080 and the Motorola MC6800 processors both began layout around December 1972. The first working 8080 chips were produced January 1974[43] and the first public announcement was in February 1974.[37] The 8080 used same three voltage N-channel MOS process as Intel's existing memory chips allowing full production to begin that April.

The first working MC6800 chips were produced in February 1974 and engineering samples were given to select customers. Hewlett Packard in Loveland, Colorado wanted the MC6800 for a new desktop calculator and had a prototype system working by June.[44][45] The MC6800 used a new single-voltage N-channel MOS process that proved to be very difficult to implement. The M6800 microcomputer system was finally in production by November 1974. Motorola matched Intel's price for single microprocessor, $360.[46][47] (The IBM System/360 was a well-known computer at this time.) In April 1975 the MEK6800D1 microcomputer design kit was offered for $300. The kit included all six chips in the M6800 family plus application and programming manuals.[48] The price of a single MC6800 microprocessor was $175.

Link Young was the product marketer that developed the total system approach for the M6800 family release. In addition to releasing a full set of support chips with the 6800 microprocessor, Motorola offered a software and hardware development system. The software development tools were available on remote time-sharing computers or the source code was available so the customer could use an in-house computer system. The software that would run on a microprocessor system was typically written in assembly language. The development system consisted of a text editor, assembler and a simulator.[49] This allowed the developer to test the software before the target system was complete. The hardware development was a desktop computer built with M6800 family CPU and peripherals known as the EXORcisor.[41] Motorola offered a three to five-day microprocessor design course for the 6800 hardware and software.[50] This systems-oriented approach became the standard way new microprocessor were introduced.[51]

5.6 Design team breakup

The principal design effort on the M6800 family was complete in mid-1974, and many engineers left the group or the company. Several factors led to the break-up of the design group.

Motorola had opened a new MOS semiconductor facility in Austin Texas. The entire engineering team was scheduled to relocate there in 1975.[52] Many of the employees liked living in the Phoenix suburb of Mesa and were very wary about moving to Austin. The team leaders were unsuccessful with their pleas to senior management on deferring the move.[53]

A recession hit the semiconductor industry in mid-1974 resulting in thousands of layoffs. A November 1974 issue of *Electronics* magazine reports that Motorola had laid off 4,500 employees, Texas Instruments 7,000 and Signetics 4,000.[54] Motorola's Semiconductor Products Division would lose thirty million dollars in the next 12 months and there were rumors that the IC group would be sold off. Motorola did not sell the division but they did change the management and organization.[55] By the end of 1974 Intel fired almost a third of its 3,500 employees.[56] The MOS IC business rebounded but job security was not taken for granted in 1974 and 1975.

Chuck Peddle (and other Motorola engineers) had been visiting customers to explain the benefits of microprocessors. Both Intel and Motorola had initially set the price of a single microprocessor at $360. Many customers were hesitant to adopt this new microprocessor technology with such a high price tag. (The actual price for production quantities was much lower.) In mid-1974 Peddle proposed a simplified microprocessor that could be sold at a much lower price. Motorola's "total product family" strategy did not focus on the price of MPU but on reducing the customer's total design cost.[57][58] Their immediate goal was to get their completed system into production and they would work on improvements in 1975.

Peddle continued working for Motorola while looking for investors for his new microprocessor concept.[59] In August 1974 Chuck Peddle left Motorola and joined a small semiconductor company in Pennsylvania, MOS Technol-

Introductory advertisement for the MOS Technology MCS6501 microprocessor in August 1975

ogy. He was followed by seven other Motorola engineers: Harry Bawcum, Ray Hirt, Terry Holdt, Mike James, Will Mathis, Bill Mensch and Rod Orgill.[27] Peddle's group at MOS Technology developed two new microprocessors that were compatible with the Motorola peripheral chips like the 6820 PIA. Rod Orgill designed the MCS6501 processor that would plug into a MC6800 socket and Bill Mensch did the MCS6502 that had the clock generation circuit on chip. These microprocessors would not run 6800 programs because they had a different architecture and instruction set. The major goal was a microprocessor that would sell for under $25. This would be done by removing non-essential features to reduce the chip size. An 8-bit stack pointer was used instead of a 16-bit one. The second accumulator was omitted. The address buffers did not have a three-state mode for Direct Memory Access (DMA) data transfers.[60] The goal was to get the chip size down to 153 mils x 168 mils (3.9 mm x 4.3 mm).[14]

Chuck Peddle was a very effective spokesman and the MOS Technology microprocessors were extensively covered in the trade press. One of the earliest was a full-page story on the MCS6501 and MCS6502 microprocessors in the July 24, 1975 issue of *Electronics* magazine.[61] Stories also ran in *EE Times* (August 24, 1975),[62] *EDN* (September 20, 1975), *Electronic News* (November 3, 1975) and *Byte* (November 1975). Advertisements for the 6501 appeared in several publications the first week of August 1975. The 6501 would be for sale at the WESCON trade show in San Francisco, September 16–19, 1975, for $25 each.[63] In September 1975 the advertisements included both the 6501 and the 6502 microprocessors. The 6502 would only cost $20.[64]

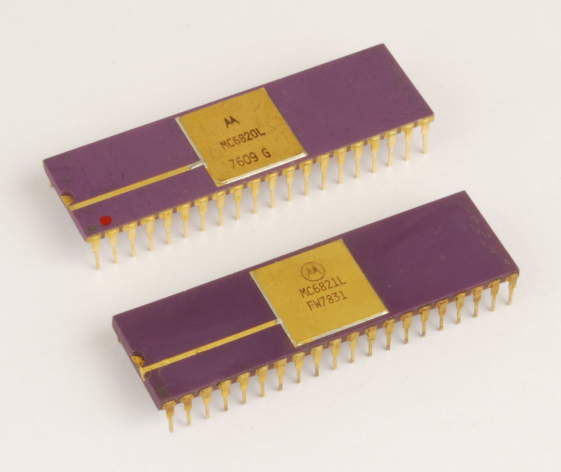
Motorola responded to MOS Technology's $20 microprocessor by immediately reducing the single-unit price of the 6800 microprocessor from $175 to $69[65] and then suing MOS Technology in November 1975.[66] Motorola claimed that the eight former Motorola engineers used technical information developed at Motorola in the design of the 6501 and 6502 microprocessors. MOS Technology's other business, calculator chips, was declining due to a price war with Texas Instruments so their financial backer, Allen-Bradley, decided to limit the possible losses and sold the assets of MOS Technology back to the founders.[27] The lawsuit was settled in April 1976 with MOS Technology dropping the 6501 chip that would plug into a Motorola 6800 socket and licensing Motorola's peripheral chips.[67][68] Motorola reduced the single-unit price of the 6800 to $35.[35][69]

The MOS Technology vs. Motorola lawsuit has developed a David and Goliath narrative over the years. One point was the Motorola did not have patents on the technology. This was technically true when the lawsuit was filed in late 1975. On October 30, 1974, before the 6800 was released, Motorola filed numerous patents applications on the microprocessor family and was granted over twenty patents. The first was to Tom Bennett on June 8, 1976 for the 6800 internal address bus.[13] The second was to Bill Mensch on July 6, 1976 for the 6820 chip layout.[21] Many of these patents named several of the departing engineers as co-inventors. These patents covered the 6800 bus and how the peripheral chips interfaced with the microprocessor.[70] (Intel had a similar incident. Federico Faggin, who had led the development of the Intel's first microprocessor, the 4004, and it latest, the 8080, grew restless under the management changes at Intel. Faggin and another Intel engineer, Ralph Ungermann, began talking about starting up their own microprocessor company. Faggin and Ungermann left Intel and started Zilog in November 1974. Masatoshi Shima, the designer of the Intel 8080, joined Zilog in February 1975 and they obtained funding from Exxon's venture capital group in June 1975. Zilog decided to make a superset of the Intel 8080 that also incorporated features from the 6800 and others. The Z80 only required a single 5 volt power supply and a single-phase clock input. It was the first microprocessor to offer built-in support for dynamic RAM.[30][71][72][73])

5.7 Move to Austin

Gary Daniels was designing ICs for electronic wristwatches when Motorola shut down their Timepiece Electronics Unit. Tom Bennett offered him a job in the microprocessor group in November 1974. Bennett did not want to leave the Phoenix area so Gary Daniels managed the microprocessor development in Austin. (Daniels was the microprocessor design manager for the next ten years before he was promoted to a vice president.)

The first task was to redesign the 6800 MPU to improve the manufacturing yield and to operate at a faster clock. This design used depletion-mode technology and was known internally as the MC6800D. The transistor count went from 4000 to 5000 but the die area was reduced from 29.0 mm^2 to 16.5 mm^2. The maximum clock rate for selected parts doubled to 2 MHz. The other chips in the M6800 family were also redesigned to use depletion-mode technology. The Peripheral Interface Adapter had



The M6800 family chips were redesigned to use depletion-mode technology. The MC6820 PIA became the MC6821.

a slight change in the electrical characteristics of the I/O pins so the MC6820 became the MC6821.[74] These new IC were completed in July 1976.

A new low-cost clock generator chip, the MC6875, was released in 1977. It replaced the $35 MC6870 hybrid IC. The MC6875 came in a 16-pin dip package and could use quartz crystal or a resistor capacitor network.[75]

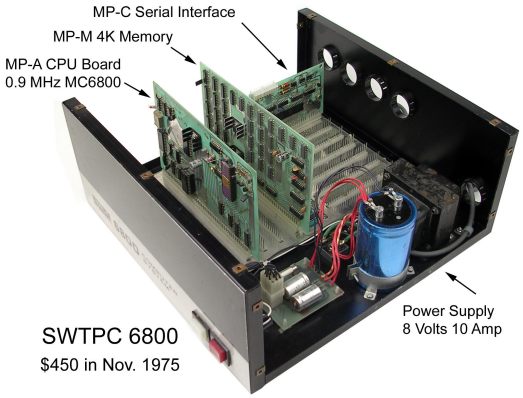
Another project was incorporating 128 bytes of RAM and the clock generator on a single 11,000-transistor chip. The MC6802 microprocessor was released in March 1977. The companion MC6846 chip had 2048 byte ROM, an 8-bit bidirectional port and a programmable timer. This was a two-chip microcomputer. The 6802 has an on-chip oscillator that uses an external 4 MHz quartz crystal to produce the two-phase 1 MHz clock. The internal 128 byte RAM could be disabled by grounding a pin and devices with defective RAM were sold as a MC6808.[76]

A series of peripheral chip were introduced by 1978. The MC6840 programmable counter had three 16-bit binary counters that could be used for frequency measurement, event counting, or interval measurement. The MC6844 Direct Memory Access Controller could transfer data from an I/O controller to RAM without loading down the MC6800 microprocessor. The MC6845 CRT Controller provided the control logic for a character based computer terminal. The 6845 had support for a light pen, an alternative to a computer mouse. This was a very popular chip and was even used in the original IBM PC Monochrome Display Adapter with the Intel 8088 16-bit microprocessor in 1981, and in the follow-up IBM Color Graphics Adapter for the original PC and successors; the IBM Enhanced Graphics Adapter card contained custom IBM chips that emulated the Motorola 6845, with minor differences.

The MC6801 was a single-chip microcomputer with a 6802 CPU with 128 bytes of RAM, a 2 KB ROM, a 16-bit timer, 31 programmable parallel I/O lines, and a serial port. It could also use the I/O lines as data and address buses to connect to standard M6800 peripherals. The 6801 would execute 6800 code but it had ten additional instructions and the execution time of key instructions was reduced. The two 8-bit accumulators could act as a single 16-bit accumulator for double precision addition, subtraction and multiplication.[77] It was initially designed for automotive use with General Motors as the lead customer. The first application was a trip computer for the 1978 Cadillac Seville.[78] This 35,000 transistor chip was too expensive for wide-scale adoption in automobiles so a reduced function MC6805 single-chip microcomputer was designed.

The MC6809 was the most advanced 8-bit microprocessor Motorola produced. It had a new instruction set that was similar to the 6800 but abandoned op-code compatibility for improved performance and high-level language support; the two were software compatible in that assemblers could (and generally did) generate code which was equivalent to 6800 opcodes the 6809 did not directly emulate. In that sense, the 6809 was upward compatible with the 6800. The 6809 had many 16-bit operations, including the first 8-bit multiply instruction (generating a 16 bit product) in a microprocessor, and two 16-bit index registers and stack pointers.[79]

5.8 Personal computers



The SWTPC 6800 computer system, introduced in November 1975, was based on the MEK6800 design evaluation kit chip set.

The MITS Altair 8800, the first successful personal computer, used the Intel 8080 microprocessor and was featured on the January 1975 cover of *Popular Electronics*.[80] The first personal computers using the Motorola 6800 were introduced in late 1975. Sphere Corporation of Bountiful, Utah ran a quarter-page advertisement in the July 1975 issue of *Radio-Electronics* for a 650 USD computer kit with a 6800 microprocessor, 4 kilobytes of RAM, a video board and a keyboard. This would display 16 lines of 32 characters on a TV or monitor.[81]

The Sphere computer kits began shipping in November 1975.[82] Southwest Technical Products Corporation of San Antonio, Texas, officially announced their SWTPC 6800 Computer System in November 1975. Wayne Green visited SWTPC in August 1975 and described the SWTPC computer kit complete with photos of a working system in the October 1975 issue of *73*. The SWTPC 6800 was based on the Motorola MEK6800 design evaluation kit chip set and used the MIKBUG ROM Software.[25] The MITS Altair 680 was on the cover of the November 1975 issue of *Popular Electronics*. The Altair 680 used a 6800 microprocessor and also had a front panel with toggle switches and LEDs. The initial design had to be revised and first deliveries of the Altair 680B were in April 1976.[83]

Sphere was a small startup company and had difficulties delivering all of the products they announced. They filed for a Chapter 11 bankruptcy in April 1977.[84] The Altair 680B was popular but MITS focused most of the resources on their Altair 8800 computer system and they exited the hobby market in 1978. The Southwest Technical Products computer was the most successful 6800 based personal computer.[85][86] Other companies, for instance, Smoke signal Broadcasting (California), Gimix (Chicago), Midwest Scientific (Olathe, Kansas), and Helix Systems (Hazelwood, Missouri), started producing SWTPC 6800 compatible boards and complete systems. The 8080 systems were far more popular than the 6800 ones.[87]

The Tektronix 4051 graphics computing system used a 6800 microprocessor.

The Tektronix 4051 Graphics Computing System was introduced in October 1975. This was a professional desktop computer that had a 6800 microprocessor with up to 32 KB of user RAM, 300 KB magnetic tape storage, BASIC in ROM and a 1024 by 780 graphics display. The Tektronix 4051 sold for $7000, rather higher than the personal computers using the 6800.[88]

By 1977 personal computers were fully assembled and ready to use, not do-it-yourself kits. The Apple II and Commodore PET were based on the MOS Technology 6502 microprocessor designed by former Motorola engineers. The Radio Shack TRS-80 used the Zilog Z80 microprocessor designed by former Intel engineers Federico Faggin and Masatoshi Shima.

The 6800 processor was also used in the APF Imagination Machine game console.

The architecture and instruction set of the 6800 were easy for beginners to understand and Heathkit developed a microprocessor course and the ET3400 6800 trainer. The course and trainer proved popular with individuals and schools.[89]

Motorola's next generation 8-bit microprocessor architecture, the MC6809 (1979), was used in the Radio Shack TRS-80 Color Computer and the compatible Dragon 32/64 which was sold in Europe. SWTPC also released a 6809 based system, the s/09, as did other SS-50 vendors. The 6809 and the 16/32 bit 68000 were incompatible with the instruction set of the 6800, but could use 6800-family peripheral chips.

An clone of the 6800 processor was used in the Bulgarian computer Pyldin-601. About 35000 of these computers were produced from 1988 to 1992. They were used mainly for educational and industrial purposes.

5.9 Example code

The following 6800 assembler source code is for a subroutine named memcpy that copies a block of data bytes of a given size from one location to another. The data block is copied one byte at a time, from lowest address to highest.

; memcpy -- ; Copy a block of memory from one location to another. ; ; Entry parameters ; cnt - Number of bytes to copy ; src - Address of source data block ; dst - Address of target data block cnt dw $0000 src dw $0000 dst dw $0000 memcpy public ldab cnt+1 ;Set B = cnt.L beq check ;If cnt.L=0, goto check loop ldx src ;Set IX = src ldaa ix ;Load A from (src) inx ;Set src = src+1 stx src ldx dst ;Set IX = dst staa ix ;Store A to (dst) inx ;Set dst = dst+1 stx dst decb ;Decr B bne loop ;Repeat the loop stab cnt+1 ;Set cnt.L = 0 check tst cnt+0 ;If cnt.H=0, beq done ;Then quit dec cnt+0 ;Decr cnt.H decb ;Decr B bra loop ;Repeat the loop done rts ;Return

5.10 Peripherals

List from "Motorola Microcomputer Components", November 1978

5.11 Second sources

A common requirement for manufacturing companies was to require two or more sources for every part in the products they made. This ensured they could get parts if a supplier had financial problems or a disaster. Initially Motorola selected American Microsystems Inc (AMI) as a second source for the M6800 family. Hitachi, Fujitsu, Fairchild, Rockwell and Thomson Semiconductors were added later.

- AMI S6800 MPU
- Fairchild F6802P and an AMI S6820 PIA

5.12 Oral histories

- "Intel 8080 Microprocessor Oral History Panel" Steve Bisset, Federico Faggin, Hal Feeney, Edward Gelbach, Ted Hoff, Stan Mazor, Masatoshi Shima, Computer History Museum, April 26, 2007, moderator: David House.
- "Zilog Z80 Microprocessor Oral History Panel" Federico Faggin, Masatoshi Shima, Ralph Ungermann, Ralph Ungermann. Computer History Museum, April 27, 2007, moderator: Michael Slater.
- "Motorola 6800 Oral History Panel" Thomas H. Bennett, John Ekiss, William (Bill) Lattin, Jeff Lavell. Computer History Museum, March 28, 2008, moderator: David Laws.
- Interview with William Mensch Stanford and the Silicon Valley Project, October 9, 1995. Transcript

5.13 References

[1] "Motorola joins microprocessor race with 8-bit entry". *Electronics* (New York: McGraw-Hill) **47** (5): pp. 29–30. March 7, 1974.

[2] "Microcomputer system runs on one 5-V supply". *Electronics* (New York: McGraw-Hill) **47** (26): pp.114–115. December 26, 1974. "Motorola's M6800 microcomputer system, which can operate from a single 5-volt supply, is moving out of the sampling stage and into full production." The small-quantity price of the MC6800 is $360. The MC6820 PIA cost $28.

[3] "The Digital Age". *Electronics* (New York: McGraw-Hill) **53** (9): p. 377. April 17, 1980. "It introduced its 6800 microprocessor in March 1974. The device needed only one +5-volt power supply, in contrast with the Intel 8080's three. And it had an untangled bus structure like the one the Digital Equipment Corp. put in its PDP-11 minicomputers."

[4] Ceruzzi, Paul E. (2003). *A History of Modern Computing*. Cambridge, MA: MIT Press. p. 244. ISBN 0-262-53203-4. "The microprocessor phenomenon passed the PDP-11 by, even though elements of its architecture turned up in microprocessor designs (especially the Motorola 6800)." - Author interviewed Gordon Bell, designer of the PDP-11

[5] *M6800 Microcomputer System Design Data*. Phoenix AZ: Motorola. 1976.

[6] Daniels, R. Gary; William C. Bruce (April 1985). "Built-In Self-Test Trends in Motorola Microprocessors". *IEEE Design & Test of Computers* (IEEE) **2** (2): pp. 64–71. doi:10.1109/MDT.1985.294865. extquotedbl… MC6800, which was introduced in 1974. The device was built in six-micron NMOS technology with about 4000 transistors."

[7] *M6800 Microprocessor Applications Manual*. Phoenix AZ: Motorola. 1975.

[8] *Motorola 1955 Annual Report*. Chicago: Motorola. 1956. p. 9.

[9] *Motorola 1975 Annual Report*. Chicago: Motorola. March 1976.

[10] Malone, Michael S. (1995). *The Microprocessor: A Biography*. New York: Springer-Verlag. pp. 141–147. ISBN 0-387-94342-0.

[11] Motorola 6800 Oral History (2008)

[12] "1964 - First Commercial MOS IC Introduced". Computer History Museum. 2007. Retrieved August 9, 2010.

[13] Bennett, Thomas H., "Split low order internal address bus for microprocessor", US Patent 3962682, issued June 8, 1976. Bennett is listed as an inventor on 18 M6800 family patents.

[14] Cushman, Robert H. (September 20, 1975). "2-1/2 Generation μP's -$10 Parts That Perform Like Low-End Mini's". *EDN* (Boston: Cahners Publishing) **20** (17): pp. 36–42. About the MOS Technology 6502 on page 40. "It measures just 168x183 mils now and will be shrunk 10% to 153x168 mils soon."

[15] *Electronics* April 18, 1974. Photo of boards on page 82, description of circuit on page 93.

[16] Buchanan, John K., "MOS DC Voltage booster circuit", US Patent 3942047, issued March 2, 1976.

[17] Buchanan, John K., "Chip topography for MOS integrated circuitry microprocessor chip", US Patent 3987418, issued October 19, 1976.

[18] Idleman, Thomas E.; Jenkins, Francis S.; McCalla, William .J.; Pederson, Donald. O (August 1971). "SLIC - A Simulator for Linear Integrated Circuits". *Solid-State Circuits, IEEE Journal of* (IEEE) **6** (4): pp. 188–203. doi:10.1109/jssc.1971.1050168. ISSN 0018-9200.

[19] Jenkins, Francis; Lane, E.; Lattin, W.; Richardson, W. (November 1973). "MOS-device modeling for computer implementation". *Circuit Theory, IEEE Transactions on* (IEEE) **20** (6): 649–658. doi:10.1109/tct.1973.1083758. ISSN 0018-9324. All of the authors were with Motorola's Semiconductor Products Division.

[20] Hoefler, Don (November 1, 1975). "Outer". *Microelectronics News* (Santa Clara, CA): p. 2. Bill Lattin leaves Motorola to join Intel.

[21] Mensch, William D., "Chip topography for MOS interface circuit", US Patent 3968478, issued July 6, 1976.

[22] Bill Memsch's is listed as an inventor on the following M6800 patents : 3979730, 4020472, 4086627, 4087855, 4145751, 4218740, 4263650

[23] Michael F. Wiles is listed as an inventor on the following US Patents on the Motorola 6800 system: 3979730, 4003028, 4004281, 4004283, 4010448, 4016546, 4020472, 4030079, 4032896, 4037204, 4040035, 4069510, 4086627, 4087855, 4090236, 4145751, 4218740, 4263650

[24] Wiles, Mike; Andre Felix (1974). *Engineering Note 100: MCM6830L7 MIKBUG/MINIBUG ROM*. Phoenix Arizona: Motorola Semiconductor Products.

[25] "SWTPC 6800: The Computer System You Have Been Waiting For". *Byte* (Peterborough MH: Green Publishing) **1** (3): Cover 2. November 1975. First advertisement for the SWTPC 6800 computer.

[26] Daniels, R. Gary (December 1996). "A Participant's Perspective". *IEEE Micro* (IEEE Computer Society) **16** (5): pp. 21–31. doi:10.1109/40.546562. Daniels, "My first assignment was to lead a small team to redesign the 6800 MPU to make it more manufacturable and so that higher speed versions could be selected."

[27] Bagnall, Brian (2006). *On the Edge: The Spectacular Rise And Fall of Commodore*. Winnipeg, Manitoba: Variant Press. pp. 9–12. ISBN 0-9738649-0-7. Chapters 1 and 2 cover Chuck Peddle's early life, his time at Motorola and the genesis of the 6501/6502 microprocessors.

[28] Charles Peddle is listed as an inventor on the following US Patents on the Motorola 6800 system: 3975712, 3979730, 4004283, 4006457, 4016546, 4020472, 4030079, 4032896, 4037204, 4040035, 4086627, 4087855, 4090236, 4145751, 4218740, 4263650. Most of these patents have six or more co-inventors.

[29] "Motorola 6800 prototype board". Computer History Museum. Retrieved July 5, 2010. Gift from Thomas H. Bennett, designer of the 6800 microprocessor. This 6800 prototype board was constructed by Chuck Peddle and John Buchanan.

[30] "Microprocessors: Designers gain new freedom as options multiply". *Electronics* (New York: McGraw-Hill) **49** (8): pp. 78–100. April 15, 1976. This was Electronics magazine annual microprocessor special edition

[31] Motorola 6800 Oral History (2008) pp. 15-16

[32] Clements, Alan (1982). *Microcomputer Design and Construction*. Prentice-Hall. pp. 70, 49. ISBN 0-13-580738-7.

[33] "How to drive a microprocessor". *Electronics* (New York: McGraw-Hill) **49** (8): p. 159. April 15, 1976. Motorola's Component Products Department sold hybrid ICs that included a quartz oscillator. These ICs produced the two-phase non-overlapping waveforms that the 6800 and 8080 required. Later, Intel produced the 8224 clock generator and Motorola produced the MC6875. The Intel 8085 and the Motorola 6802 processors included this circuitry on chip.

[34] "Intel's Higher Speed 8080 µP". *Microcomputer Digest* (Cupertino CA: Microcomputer Associates) **2** (3): p. 7. September 1975.

[35] Verhofstadt, Peter (June 1976). "Evaluation of technology options for LSI processing elements". *Proceedings of the IEEE* (IEEE) **64** (6): pp. 842–851. doi:10.1109/PROC.1976.10234.

[36] Motorola 6800 Oral History (2008), p. 27

[37] Masatoshi, Shima; Federico Faggin; Stanley Mazor (February 1974). "An N-Channel 8-Bit Single Chip Microprocessor". "Solid-State Circuits Conference. Digest of Technical Papers. 1974 IEEE International". IEEE Computer Society Press. pp. 56, 57, 229. doi:10.1109/ISSCC.1974.1155265. Table 2 on page 229 gives the 8080 chip size as 164 x 191 mils. The 8008 was 124 x 173 mils

[38] Wikes, W. E. (January 1977). "A Microprocessor Chip Designed with the User in Mind". *Computer* (IEEE) **10** (1): pp. 18–22. doi:10.1109/C-M.1977.217492. This paper describes the Electronic Arrays EA9002 microprocessor that was 200 by 200 mils and fabricated on a 3 inch silicon wafer.

[39] Elmasry, Mohamed I., ed. (1981). *Digital MOS integrated circuits*. IEEE Press. ISBN 978-0-87942-152-6. A 3-inch wafer can hold 200 dies of 160 by 160 mils. Total yield is Wafer yield x Assembly yield x Final test yield. In 1976 this was 40% x 80% x 85% or 26%. A 3 inch wafer with 200 die would yield 54 working microprocessors.

[40] "Electronics Newsletter: 6800 gains speed, lower prices by summer". *Electronics* (New York: McGraw-Hill) **49** (5): p. 25. March 4, 1976.

[41] Young, Link; Tom Bennett; Jeff LaVell (April 18, 1974). "N-channel MOS technology yields new generation of microprocessors". *Electronics* (New York: McGraw-Hill) **47** (8): pp. 88–95.

[42] Shima, Masatoshi; Federico Faggin (April 18, 1974). "In switch to n-MOS microprocessor gets a 2-µs cycle time". *Electronics* (New York: McGraw-Hill) **47** (8): pp. 95–100.

[43] House, Dave (April 26, 2007). "Oral History Panel on the Development and Promotion of the Intel 8080 Microprocessor". Mountain View, CA: Computer History Museum.

[44] Motorola 6800 Oral History (2008) pp. 9, 15

[45] "HP designs custom 16-bit uC chip". *Microcomputer Digest* (Cupertino CA: Microcomputer Associates) **2** (4): p. 8. October 1975. "The instrument is a companion to the firm's new 9815A calculator which uses a Motorola M6800 microcomputer and is priced at $2900."

[46] "Motorola microprocessor set is 1 MHz n-MOS". *Control Engineering* **21** (11): p. 11. November 1974. MC6800 microprocessor price was $360. The MC6850 asynchronous communications interface adaptor (ACIA) was slated for first quarter 1975 introduction.

[47] Intel Corporation; Glynnis Thompson Kaye (Editor) (1984). *A Revolution in Progress - A History to Date of Intel*. Intel Corporation. p. 14. Order number:231295. "Shima implemented the 8080 in about a year and the new device was introduced in April 1974 for $360."

[48] "Motorola mounts M6800 drive". *Electronics* (New York: McGraw-Hill) **48** (8): p. 25. April 17, 1975. "Distributors are being stocked with the M6800 family, and the division is also offering an introductory kit that includes the family's six initial parts, plus applications and programming manuals, for $300."

[49] *M6800 Microprocessor Programming Manual*. Phoenix AZ: Motorola Semiconductor Products. 1975. This book was the instruction manual for the development software. Some of the software listing examples have dates from 1973 and 1974.

[50] "It's Easy and Inexpensive.". *Electronics* (New York: McGraw-Hill) **49** (8): p. 27. April 15, 1976. The three-day design course cost $375 and included a copy of all the training materials. A company could schedule a course for 20 engineers at their own facility for $4000.

[51] Noyce, Robert N.; Marcian E. Hoff, Jr. (February 1981). "A History of Microprocessor Development at Intel". *IEEE MICRO* (IEEE Computer Society Press) **1** (1): pp. 8–21. doi:10.1109/MM.1981.290812. "Motorola also introduced a development system and four peripheral chips mated to the 6800. Motorola's systems-oriented approach influenced the industry; henceforth CPUs would be introduced with full support available rather than on a trailing schedule."

[52] "Semiconductor makers delay expansion". *Electronics* (New York: McGraw-Hill) **47** (23): pp. 82–85. November 14, 1974. Motorola's Austin MOS plant already in operation. "However, engineering and marketing won't move until 1975."

[53] Hoefler, Don (July 3, 1976). "Backfire". *Microelectronics News* (Santa Clara, CA): p. 3.

[54] "Semiconductor makers continue to trim employment". *Electronics* (New York: McGraw-Hill) **47** (24): p. 46. November 28, 1974.

[55] Waller, Larry (November 13, 1975). "Motorola seeks to end skid". *Electronics* (New York: McGraw-Hill) **48** (23): pp. 96–98. Summary: Semiconductor Products split into two parts, integrated circuits and discrete components. Semiconductor losses for the last four quarters exceeded $30 million. The sales organization lost its sensitivity to customer needs, "delays in responding to price cuts meant that customers bought elsewhere." Technical problems plagued IC production. The troubles are "not in design, but in chip and die yields." Problems have been solved. The MC6800 microprocessor "arrived in November 1974."

[56] Tedlow, Richard S. (2007). *Andy Grove: The Life and Times of an American Business Icon*. New York: Portfolio. p. 158. ISBN 1-59184-182-8. "By the end of the year [1974], Intel had fired fully 30 percent of its thirty-five hundred employees."

[57] "It's the total product family". *Electronics* (New York: McGraw-Hill) **48** (1): p. 37. January 9, 1975. Motorola advertisement emphasizing their complete set of peripheral chips and development tools. This shorten the customers product design cycle.

[58] Motorola 6800 Oral History (2008) p. 18

[59] Bagnall (2006), "On the Edge". Page 10, "While still employed at Motorola, Peddle tried raising money to fund his new microprocessor.

[60] Fylstra, Daniel (November 1975). "Son of Motorola (or the $20 CPU Chip) extquotedbl. *Byte* (Peterborough, NH: Green Publishing) **1** (3): pp. 56–62. Comparison of the 6502 and the 6800 microprocessors. Author visited MOS Technology in August 1975.

[61] "Microprocessor line offers 4, 8,16 bits". *Electronics* (New York: McGraw-Hill) **48** (15): p. 118. July 24, 1975. The article covers the 6501 and 6502 plus the 28 pin versions that would only address 4K of memory. It also covered future devices such as "a design that Peddle calls a pseudo 16."

[62] Sugarman, Robert (25 August 1975). "Does the Country Need A Good $20 Microprocessor? extquotedbl. *EE Times* (Manhasset, New York: CMP Publications): p. 25.

[63] "MOS 6501 Microprocessor beats 'em all". *Electronics* (New York: McGraw-Hill) **48** (16): pp. 60–61. August 7, 1975.

[64] "MOS 6502 the second of a low cost high performance microprocessor family". *Computer* (IEEE Computer Society) **8** (9): pp 38–39. September 1975. doi:10.1109/C-M.1975.219074.

[65] Motorola (October 30, 1975). "All this and unbundled $69 microprocessor". *Electronics* (McGraw-Hill) **48** (22): p. 11. The quantity one price for the MC6800 was reduced from $175 to $69. The previous price for 50 to 99 units was $125.

[66] "Motorola Sues MOS Technology". *Microcomputer Digest* (Cupertino CA: Microcomputer Associates) **2** (6): p. 11. December 1975.

[67] "MOS Technology Drops 6501". *Microcomputer Digest* (Cupertino CA: Microcomputer Associates) **2** (11): p. 4. May 1976.

[68] Teener, Mike (May 1976). "Politics and Intrigue". *SCCS Interface* (Los Angeles: Southern California Computer Society) **1** (6): p. 58. extquotedblSo Motorola sued and just recently won an out-of-court settlement that has MOS Technology paying $200,000 and stopping production on the 6501."

[69] "New 6800 Pricing". *SCCS Interface* (Los Angeles: Southern California Computer Society) **1** (6): p. 63. May 1976. The new prices for the Motorola 6800 were $35 for 1–9 units, $32.50 for 10–49 and $29.25 for 50–99.

[70] Motorola was awarded the following US Patents on the 6800 microprocessor family: 3962682, 3968478, 3975712, 3979730, 3979732, 3987418, 4003028, 4004281, 4004283, 4006457, 4010448, 4016546, 4020472, 4030079, 4032896, 4037204, 4040035, 4069510, 4071887, 4086627, 4087855, 4090236, 4145751, 4218740, 4263650

[71] Shima, Masatoshi; Federico Faggin; Ralph Ungermann (August 19, 1976). "Z-80 chip set heralds third microprocessor generation". *Electronics* (New York: McGraw-Hill) **49** (17): pp. 89–93.

[72] Slater, Michael (April 27, 2007). "Zilog Oral History Panel on the Founding of the Company and the Development of the Z80 Microprocessor". Mountain View, CA: Computer History Museum.

[73] Michalopoulos, D.A. (July 1976). "New Products: Zilog microcomputer". *Computer* (IEEE Computer Society) **9** (7): p. 69. doi:10.1109/C-M.1976.218651.

[74] *Advanced Information: 1.5 and 2.0 MHz Components for the M6800 Microcomputer System*. Austin, Texas: Motorola Semiconductor Products. April 1977. pp. 4–6. ADI-429. The MC6820 became the MC6821 because the electrical characteristic of PA0–7 and PB0–7 pins changed slightly. The typical Input High Current went from −250 μAdc to −400 μAdc and the Input Low Current went from 1.0 mAdc to 1.3 mAdc.

[75] "New Clock Chip for 6800 Systems". *Byte* (Peterborough NH: Byte Publications) **2** (12): p. 210. December 1977. Requiring only a 5 V supply and a quartz crystal or an RC network, the MC6875 provides buffered 2 phase clock outputs… $3.75 in 1000 piece quantities from Motorola Linear Products

[76] "Texas Instruments and Motorola pare down microprocessors for low end market". *Electronic* (McGraw-Hill) **50** (5): pp. 34, 36. March 3, 1977. MC6802 microprocessor has oscillator and 128 byte RAM. MC6846 has ROM Timer and I/O. Samples later this month.

[77] *Product Preview MC6801*. Austin, Texas: Motorola Semiconductor Products. August 1978. NP-93..

[78] Motorola 6800 Oral History (2008) pp. 21-22

[79] *Product Preview MC6809*. Austin, Texas: Motorola Semiconductor Products. December 1978. NP-98 R1..

[80] H. Edward Roberts; William Yates (January 1975). "Altair 8800 minicomputer". *Popular Electronics* (Ziff Davis) **7** (1): pp. 33–38.

[81] "Computer System $650". *Radio-Electronics* (New York: Gernsback Publications) **42** (7): p.88. July 1975.

[82] Anderson, Bruce (July 1976). "Assembling a Sphere". *Byte* (Peterborough NH: Byte Publications) **1** (11): pp. 18–20.

[83] Pollini, Steve (April 1976). "680-b ready for production". *Computer Notes* (MITS) **1** (11): p. 8. "MITS is now ready to begin full production of the Altair 680b"

[84] Norell, Melvin (May 31, 1977). "Dear Sphere Microcomputer User". *Programma News Letter* (Los Angeles: Programma Consultants): pp. 1–3.

[85] Ahl, David; Green, Burchenal (1980). *The Best of Creative Computing Volume 3*. Morristown, NJ: Creative Computing Press. pp. 106–108. ISBN 0-916688-12-7. Interview with Daniel Meyer at the "Personal Computing 77" conference at Atlantic City NJ in August 1977

[86] "SWTPC announces first dual minifloppy kit under $1,000". *Byte* (Peterborough NH: Green Publishing) **2** (10): Cover 2. October 1977.

[87] Wallace, Bob (December 1977). "Bob's Bits: Personal Computers in 1976". *Northwest Computer Club News* (Renton WA) **2** (12): p. 9.

[88] "Terminal Talks Basic". *Electronics* (New York: McGraw-Hill) **42** (22): p. 120. October 30, 1975. Ad for Tektronix 4051 in *Electronics* April 1976

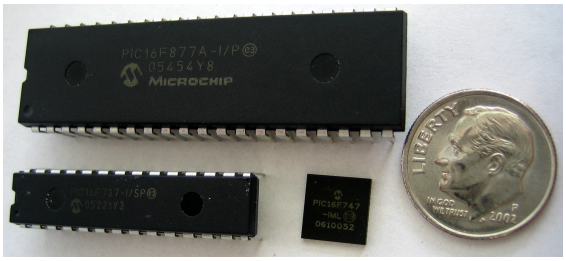
[89] "Heathkit Microprocessor Course". *Popular Science* (New York: Times Mirror Magazines) **211** (5): p. 133. November 1977. ISSN 0161-7370.

5.14 External links

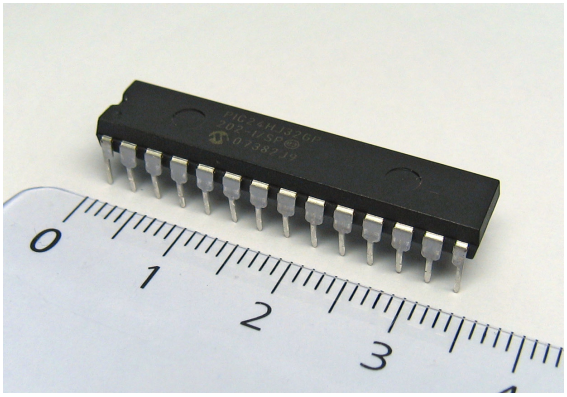
- MC6800 applications manual from 1975- lots of information
- MDOS User's Manual
- Motorola Exorciser Emulator for Windows
- Open source Motorola Exorciser and SWTPC emulator for Linux/Cygwin
- MIKBUG
- 680x images and descriptions at cpu-collection.de
- Instruction set summary
- Java Applet Simulator of a simplified M6800 Microprocessor

Chapter 6

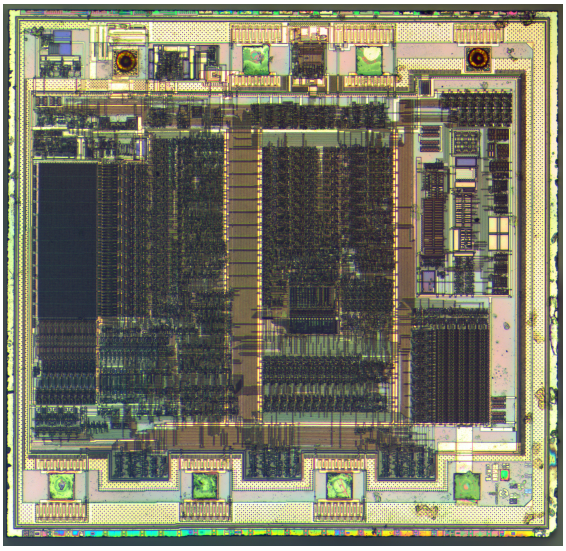
PIC microcontroller



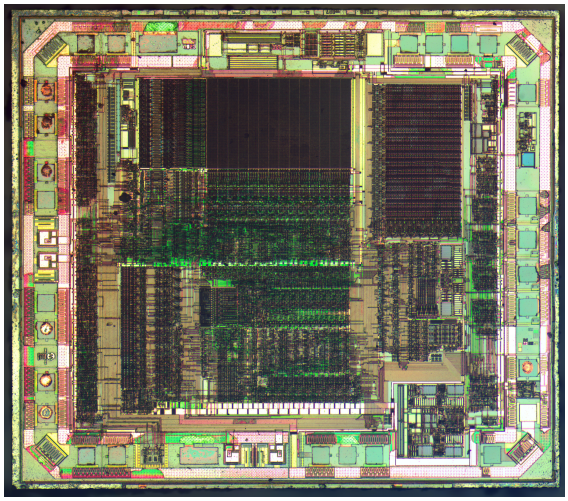
PIC microcontrollers in DIP and QFN packages



16-bit 28-pin PDIP PIC24 microcontroller next to a metric ruler



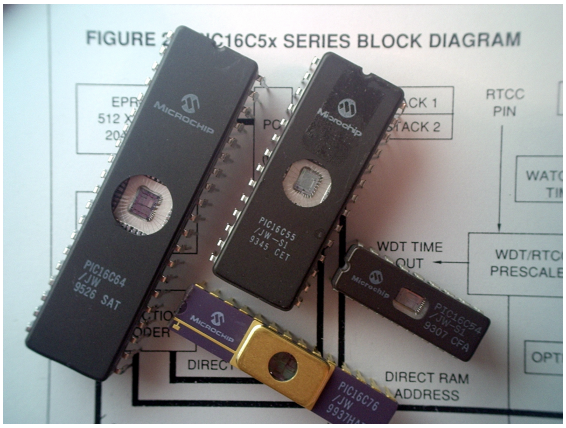
Die of a PIC12C508 8-bit, fully static, EEPROM/EPROM/ROM-based CMOS microcontroller manufactured by Microchip Technology using a 1200 nanometre process.



Die of a PIC16C505 CMOS ROM-based 8-bit microcontroller manufactured by Microchip Technology using a 1200 nanometre process.

PIC is a family of modified Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1650[1][2][3] originally developed by General Instrument's Microelectronics Division. The name PIC initially referred to extquotedbl**Peripheral Interface Controller** extquotedbl now it is extquotedbl**PIC** extquotedbl only.[4][5]

PICs are popular with both industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.



Various older (EPROM) PIC microcontrollers

6.1 History

The original PIC was built to be used with General Instrument's new CP1600 16-bit CPU. While generally a good CPU, the CP1600 had poor I/O performance, and the 8-bit PIC was developed in 1975 to improve performance of the overall system by offloading I/O tasks from the CPU. The PIC used simple microcode stored in ROM to perform its tasks, and although the term was not used at the time, it shares some common features with RISC designs.

In 1985, General Instrument spun off their microelectronics division and the new ownership cancelled almost everything — which by this time was mostly out-of-date. The PIC, however, was upgraded with an internal EPROM to produce a programmable channel controller. Today a huge variety of PICs are available with various on-board peripherals (serial communication modules, UARTs, motor control kernels, etc.) and program memory from 256 words to 64k words and more (a "word" is one assembly language instruction, varying from 8, 12, 14 or 16 bits depending on the specific PIC micro family).

PIC and PICmicro are registered trademarks of Microchip Technology. It is generally thought that PIC stands for **Peripheral Interface Controller**, although General Instruments' original acronym for the initial PIC1640 and PIC1650 devices was extquotedbl**Programmable Interface Controller** extquotedbl.[4] The acronym was quickly replaced with extquotedbl**Programmable Intelligent Computer** extquotedbl.[5]

The Microchip 16C84 (PIC16x84), introduced in 1993, was the first Microchip CPU with on-chip EEPROM memory. This electrically erasable memory made it cost less than CPUs that required a quartz "erase window" for erasing EPROM.

By 2013, Microchip was shipping over one billion PIC microcontrollers every year.[6]

6.2 Core architecture

The PIC architecture is characterized by its multiple attributes:

- Separate code and data spaces (Harvard architecture).
- A small number of fixed length instructions
- Most instructions are single cycle execution (2 clock cycles, or 4 clock cycles in 8-bit models), with one delay cycle on branches and skips
- One accumulator (W0), the use of which (as source operand) is implied (i.e. is not encoded in the opcode)
- All RAM locations function as registers as both source and/or destination of math and other functions.[7]
- A hardware stack for storing return addresses
- A small amount of addressable data space (32, 128, or 256 bytes, depending on the family), extended through banking
- Data space mapped CPU, port, and peripheral registers
- ALU status flags are mapped into the data space
- The program counter is also mapped into the data space and writable (this is used to implement indirect jumps).

There is no distinction between memory space and register space because the RAM serves the job of both memory and registers, and the RAM is usually just referred to as the register file or simply as the registers.

6.2.1 Data space (RAM)

PICs have a set of registers that function as general purpose RAM. Special purpose control registers for on-chip hardware resources are also mapped into the data space. The addressability of memory varies depending on device series, and all PIC devices have some banking mechanism to extend addressing to additional memory. Later series of devices feature move instructions which can cover the whole addressable space, independent of the selected bank. In earlier devices, any register move had to be achieved via the accumulator.

To implement indirect addressing, a "file select register" (FSR) and "indirect register" (INDF) are used. A register number is written to the FSR, after which reads from or writes to INDF will actually be to or from the register pointed to by FSR. Later devices extended this concept

with post- and pre- increment/decrement for greater efficiency in accessing sequentially stored data. This also allows FSR to be treated almost like a stack pointer (SP).

External data memory is not directly addressable except in some high pin count PIC18 devices.

6.2.2 Code space

The code space is generally implemented as ROM, EPROM or flash ROM. In general, external code memory is not directly addressable due to the lack of an external memory interface. The exceptions are PIC17 and select high pin count PIC18 devices.[8]

6.2.3 Word size

All PICs handle (and address) data in 8-bit chunks. However, the unit of addressability of the code space is not generally the same as the data space. For example, PICs in the baseline (PIC12) and mid-range (PIC16) families have program memory addressable in the same wordsize as the instruction width, i.e. 12 or 14 bits respectively. In contrast, in the PIC18 series, the program memory is addressed in 8-bit increments (bytes), which differs from the instruction width of 16 bits.

In order to be clear, the program memory capacity is usually stated in number of (single word) instructions, rather than in bytes.

6.2.4 Stacks

PICs have a hardware call stack, which is used to save return addresses. The hardware stack is not software accessible on earlier devices, but this changed with the 18 series devices.

Hardware support for a general purpose parameter stack was lacking in early series, but this greatly improved in the 18 series, making the 18 series architecture more friendly to high level language compilers.

6.2.5 Instruction set

A PIC's instructions vary from about 35 instructions for the low-end PICs to over 80 instructions for the high-end PICs. The instruction set includes instructions to perform a variety of operations on registers directly, the accumulator and a literal constant or the accumulator and a register, as well as for conditional execution, and program branching.

Some operations, such as bit setting and testing, can be performed on any numbered register, but bi-operand arithmetic operations always involve W (the accumulator), writing the result back to either W or the other operand register. To load a constant, it is necessary to load it into W before it can be moved into another register. On the older cores, all register moves needed to pass through W, but this changed on the "high end" cores.

PIC cores have skip instructions which are used for conditional execution and branching. The skip instructions are 'skip if bit set' and 'skip if bit not set'. Because cores before PIC18 had only unconditional branch instructions, conditional jumps are implemented by a conditional skip (with the opposite condition) followed by an unconditional branch. Skips are also of utility for conditional execution of any immediate single following instruction. It is possible to skip skip instructions. For example, the instruction sequence "skip if A; skip if B; C" will execute C if A is true or if B is false.

The 18 series implemented shadow registers which save several important registers during an interrupt, providing hardware support for automatically saving processor state when servicing interrupts.

In general, PIC instructions fall into 5 classes:

1. Operation on working register (WREG) with 8-bit immediate ("literal") operand. E.g. movlw (move literal to WREG), andlw (AND literal with WREG). One instruction peculiar to the PIC is retlw, load immediate into WREG and return, which is used with computed branches to produce lookup tables.

2. Operation with WREG and indexed register. The result can be written to either the Working register (e.g. addwf *reg*,w). or the selected register (e.g. addwf *reg*,f).

3. Bit operations. These take a register number and a bit number, and perform one of 4 actions: set or clear a bit, and test and skip on set/clear. The latter are used to perform conditional branches. The usual ALU status flags are available in a numbered register so operations such as "branch on carry clear" are possible.

4. Control transfers. Other than the skip instructions previously mentioned, there are only two: goto and call.

5. A few miscellaneous zero-operand instructions, such as return from subroutine, and sleep to enter low-power mode.

6.2.6 Performance

The architectural decisions are directed at the maximization of speed-to-cost ratio. The PIC architecture was among the first scalar CPU designs, and is still among the simplest and cheapest. The Harvard architecture—in which instructions and data come from

separate sources—simplifies timing and microcircuit design greatly, and this benefits clock speed, price, and power consumption.

The PIC instruction set is suited to implementation of fast lookup tables in the program space. Such lookups take one instruction and two instruction cycles. Many functions can be modeled in this way. Optimization is facilitated by the relatively large program space of the PIC (e.g. 4096 × 14-bit words on the 16F690) and by the design of the instruction set, which allows for embedded constants. For example, a branch instruction's target may be indexed by W, and execute a "RETLW" which does as it is named - return with literal in W.

Interrupt latency is constant at three instruction cycles. External interrupts have to be synchronized with the four clock instruction cycle, otherwise there can be a one instruction cycle jitter. Internal interrupts are already synchronized. The constant interrupt latency allows PICs to achieve interrupt driven low jitter timing sequences. An example of this is a video sync pulse generator. This is no longer true in the newest PIC models, because they have a synchronous interrupt latency of three or four cycles.

6.2.7 Advantages

- Small instruction set to learn
- RISC architecture
- Built in oscillator with selectable speeds
- Easy entry level, in-circuit programming plus in-circuit debugging PICKit units available for less than $50
- Inexpensive microcontrollers
- Wide range of interfaces including I^2C, SPI, USB, USART, A/D, programmable comparators, PWM, LIN, CAN, PSP, and Ethernet[9]
- Availability of processors in DIL package make them easy to handle for hobby use.

6.2.8 Limitations

- One accumulator
- Register-bank switching is required to access the entire RAM of many devices
- Operations and registers are not orthogonal; some instructions can address RAM and/or immediate constants, while others can use the accumulator only.

The following stack limitations have been addressed in the **PIC18** series, but still apply to earlier cores:

- The hardware call stack is not addressable, so preemptive task switching cannot be implemented
- Software-implemented stacks are not efficient, so it is difficult to generate reentrant code and support local variables

With paged program memory, there are two page sizes to worry about: one for CALL and GOTO and another for computed GOTO (typically used for table lookups). For example, on PIC16, CALL and GOTO have 11 bits of addressing, so the page size is 2048 instruction words. For computed GOTOs, where you add to PCL, the page size is 256 instruction words. In both cases, the upper address bits are provided by the PCLATH register. This register must be changed every time control transfers between pages. PCLATH must also be preserved by any interrupt handler.[10]

6.2.9 Compiler development

While several commercial compilers are available, in 2008, Microchip released their own C compilers, C18 and C30, for the line of 18F 24F and 30/33F processors.

As of 2013, Microchip offers their XC series of compilers, for use with MPLAB X. Microchip will eventually phase out its older compilers such as C18, and recommends using their XC series compilers for new designs.[11]

The easy to learn RISC instruction set of the PIC assembly language code can make the overall flow difficult to comprehend. Judicious use of simple macros can increase the readability of PIC assembly language. For example, the original Parallax PIC assembler ("SPASM") has macros which hide W and make the PIC look like a two-address machine. It has macro instructions like "mov b, a" (move the data from address *a* to address *b*) and "add b, a" (add data from address *a* to data in address *b*). It also hides the skip instructions by providing three operand branch macro instructions such as "cjne a, b, dest" (compare *a* with *b* and jump to *dest* if they are not equal).

6.3 Family core architectural differences

PICmicro chips have a Harvard architecture, and instruction words are unusual sizes. Originally, 12-bit instructions included 5 address bits to specify the memory operand, and 9-bit branch destinations. Later revisions added opcode bits, allowing additional address bits.

6.3.1 Baseline core devices (12 bit)

These devices feature a 12-bit wide code memory, a 32-byte register file, and a tiny two level deep call stack. They are represented by the PIC10 series, as well as by some PIC12 and PIC16 devices. Baseline devices are available in 6-pin to 40-pin packages.

Generally the first 7 to 9 bytes of the register file are special-purpose registers, and the remaining bytes are general purpose RAM. Pointers are implemented using a register pair: after writing an address to the FSR (file select register), the INDF (indirect f) register becomes an alias for the addressed register. If banked RAM is implemented, the bank number is selected by the high 3 bits of the FSR. This affects register numbers 16–31; registers 0–15 are global and not affected by the bank select bits.

Because of the very limited register space (5 bits), 4 rarely read registers were not assigned addresses, but written by special instructions (OPTION and TRIS).

The ROM address space is 512 words (12 bits each), which may be extended to 2048 words by banking. CALL and GOTO instructions specify the low 9 bits of the new code location; additional high-order bits are taken from the status register. Note that a CALL instruction only includes 8 bits of address, and may only specify addresses in the first half of each 512-word page.

Lookup tables are implemented using a computed GOTO (assignment to PCL register) into a table of RETLW instructions.

The instruction set is as follows. Register numbers are referred to as “f”, while constants are referred to as “k”. Bit numbers (0–7) are selected by “b”. The “d” bit selects the destination: 0 indicates W, while 1 indicates that the result is written back to source register f. The C and Z status flags may be set based on the result; otherwise they are unmodified. Add and subtract (but not rotate) instructions that set C also set the DC (digit carry) flag, the carry from bit 3 to bit 4, which is useful for BCD arithmetic.

6.3.2 ELAN Microelectronics clones (13 bit)

ELAN Microelectronics Corp. make a series of PICmicro-like microcontrollers with a 13-bit instruction word.[13] The instructions are mostly compatible with the mid-range 14-bit instruction set, but limited to a 6-bit register address (16 special-purpose registers and 48 bytes of RAM) and a 10-bit (1024 word) program space.

The 10-bit program counter is accessible as R2. Reads access only the low bits, and writes clear the high bits. An exception is the TBL instruction, which modifies the low byte while preserving bits 8 and 9.

The 7 accumulator-immediate instructions are renumbered relative to the 14-bit PICmicro, to fit into 3 opcode bits rather than 4, but they are all there, as well as an additional software interrupt instruction.

There are a few additional miscellaneous instructions, and there are some changes to the terminology (the PICmicro OPTION register is called the CONTrol register; the PICmicro TRIS registers 1–3 are called I/O control registers 5–7), but the equivalents are obvious.

*: Same opcode as 12-bit PIC
†: Instruction unique to EM78 instruction set with no PIC equivalent

Some models support multiple ROM or RAM banks, in a manner similar to other PIC microcontrollers.

6.3.3 Mid-range core devices (14 bit)

These devices feature a 14-bit wide code memory, and an improved 8 level deep call stack. The instruction set differs very little from the baseline devices, but the 2 additional opcode bits allow 128 registers and 2048 words of code to be directly addressed. There are a few additional miscellaneous instructions, and two additional 8-bit literal instructions, add and subtract. The mid-range core is available in the majority of devices labeled PIC12 and PIC16.

The first 32 bytes of the register space are allocated to special-purpose registers; the remaining 96 bytes are used for general-purpose RAM. If banked RAM is used, the high 16 registers (0x70–0x7F) are global, as are a few of the most important special-purpose registers, including the STATUS register which holds the RAM bank select bits. (The other global registers are FSR and INDF, the low 8 bits of the program counter PCL, the PC high preload register PCLATH, and the master interrupt control register INTCON.)

The PCLATH register supplies high-order instruction address bits when the 8 bits supplied by a write to the PCL register, or the 11 bits supplied by a GOTO or CALL instruction, is not sufficient to address the available ROM space.

6.3.4 Enhanced mid-range core devices (14 bit)

Enhanced mid-range core devices introduce a deeper hardware stack, additional reset methods, 14 additional instructions and ‘C’ programming language optimizations. In particular. there are two INDF registers (INDF0 and INDF1), and two corresponding FSR register pairs (FSR*n*L and FSR*n*H). Special instructions use FSR*n* registers like address registers, with a variety of addressing modes.

6.3.5 PIC17 high end core devices (16 bit)

The 17 series never became popular and has been superseded by the PIC18 architecture. It is not recommended for new designs, and availability may be limited.

Improvements over earlier cores are 16-bit wide opcodes (allowing many new instructions), and a 16 level deep call stack. PIC17 devices were produced in packages from 40 to 68 pins.

The 17 series introduced a number of important new features:

- a memory mapped accumulator
- read access to code memory (table reads)
- direct register to register moves (prior cores needed to move registers through the accumulator)
- an external program memory interface to expand the code space
- an 8-bit × 8-bit hardware multiplier
- a second indirect register pair
- auto-increment/decrement addressing controlled by control bits in a status register (ALUSTA)

6.3.6 PIC18 high end core devices (8 bit)

In 2000, Microchip introduced the PIC18 architecture. Unlike the 17 series, it has proven to be very popular, with a large number of device variants presently in manufacture. In contrast to earlier devices, which were more often than not programmed in assembly, C has become the predominant development language.[15]

The 18 series inherits most of the features and instructions of the 17 series, while adding a number of important new features:

- call stack is 21 bits wide and much deeper (31 levels deep)
- the call stack may be read and written (TOSU:TOSH:TOSL registers)
- conditional branch instructions
- indexed addressing mode (PLUSW)
- extending the FSR registers to 12 bits, allowing them to linearly address the entire data address space
- the addition of another FSR register (bringing the number up to 3)

The RAM space is 12 bits, addressed using a 4-bit bank select register and an 8-bit offset in each instruction. An additional "access" bit in each instruction selects between bank 0 (a=0) and the bank selected by the BSR (a=1).

A 1-level stack is also available for the STATUS, WREG and BSR registers. They are saved on every interrupt, and may be restored on return. If interrupts are disabled, they may also be used on subroutine call/return by setting the s bit (appending extquotedbl, FAST" to the instruction).

The auto increment/decrement feature was improved by removing the control bits and adding four new indirect registers per FSR. Depending on which indirect file register is being accessed it is possible to postdecrement, postincrement, or preincrement FSR; or form the effective address by adding W to FSR.

In more advanced PIC18 devices, an "extended mode" is available which makes the addressing even more favorable to compiled code:

- a new offset addressing mode; some addresses which were relative to the access bank are now interpreted relative to the FSR2 register
- the addition of several new instructions, notable for manipulating the FSR registers.

These changes were primarily aimed at improving the efficiency of a data stack implementation. If FSR2 is used either as the stack pointer or frame pointer, stack items may be easily indexed—allowing more efficient re-entrant code. Microchip's MPLAB C18 C compiler chooses to use FSR2 as a frame pointer.

6.3.7 PIC24 and dsPIC 16-bit microcontrollers

In 2001, Microchip introduced the dsPIC series of chips,[17] which entered mass production in late 2004. They are Microchip's first inherently 16-bit microcontrollers. PIC24 devices are designed as general purpose microcontrollers. dsPIC devices include digital signal processing capabilities in addition.

Although still similar to earlier PIC architectures, there are significant enhancements:[18]

- All registers are 16 bits wide
- Data address space expanded to 64 KB
- First 2 KB is reserved for peripheral control registers
- Data bank switching is not required unless RAM exceeds 62 KB
- "f operand" direct addressing extended to 13 bits (8 KB)

- 16 W registers available for register-register operations.

(But operations on f operands always reference W0.)

- Program counter is 22 bits (Bits 22:1; bit 0 is always 0)
- Instructions are 24 bits wide
- Instructions come in byte (B=1) and (16-bit) word (B=0) forms
- Stack is in RAM (with W15 as stack pointer); there is no hardware stack
- W14 is the frame pointer
- Data stored in ROM may be accessed directly ("Program Space Visibility")
- Interrupt vectors for different interrupt sources are supported.

Some features are:

- hardware MAC (multiply–accumulate)
- barrel shifting
- bit reversal
- (16×16)-bit single-cycle multiplication and other DSP operations
- hardware divide assist (19 cycles for 16/32-bit divide)
- hardware support for loop indexing
- Direct memory access

dsPICs can be programmed in C using Microchip's XC16 compiler (formerly called C30) which is a variant of GCC.

Instruction ROM is 24 bits wide. Software can access ROM in 16-bit words, where even words hold the least significant 16 bits of each instruction, and odd words hold the most significant 8 bits. The high half of odd words reads as zero. The program counter is 23 bits wide, but the least significant bit is always 0, so there are 22 modifiable bits.

Instructions come in 2 main varieties. One is like the classic PIC instructions, with an operation between W0 and a value in a specified f register (i.e. the first 8K of RAM), and a destination select bit selecting which is updated with the result. The W registers are memory-mapped. so the f operand may be any W register,

The other form, new to the PIC24, specifies 3 W register operands, 2 of which allow a 3-bit addressing mode specification:

The register offset addressing mode is only available to 2-operand instructions. 3-operand instructions use Ww as the second source operand, and use this encoding for an unsigned 5-bit immediate source. Note that the same Ww may be added to both Wd and Ws.

A few instructions are 2 words long. The second word is a NOP, which includes up to 16 bits of additional immediate operand.

6.3.8 PIC32 32-bit microcontrollers

In November 2007, Microchip introduced the new PIC32MX family of 32-bit microcontrollers. The initial device line-up is based on the industry standard MIPS32 M4K Core.[20] The device can be programmed using the Microchip MPLAB C Compiler for PIC32 MCUs, a variant of the GCC compiler. The first 18 models currently in production (PIC32MX3xx and PIC32MX4xx) are pin to pin compatible and share the same peripherals set with the PIC24FxxGA0xx family of (16-bit) devices allowing the use of common libraries, software and hardware tools. Today starting at 28 pin in small QFN packages up to high performance devices with Ethernet, CAN and USB OTG, full family range of mid-range 32-bit microcontrollers are available.

The PIC32 architecture brings a number of new features to Microchip portfolio, including:

- The highest execution speed 80 MIPS (120+[21] Dhrystone MIPS @ 80 MHz)
- The largest flash memory: 512 KB
- One instruction per clock cycle execution
- The first cached processor
- Allows execution from RAM
- Full Speed Host/Dual Role and OTG USB capabilities
- Full JTAG and 2 wire programming and debugging
- Real-time trace

An upcoming product from Microchip is the PIC32MZ family of microcontrollers.

6.4 Device variants and hardware features

PIC devices generally feature:

- Sleep mode (power savings)
- Watchdog timer
- Various crystal or RC oscillator configurations, or an external clock

6.4.1 Variants

Within a series, there are still many device variants depending on what hardware resources the chip features:

- General purpose I/O pins
- Internal clock oscillators
- 8/16/32 bit timers
- Internal EEPROM memory
- Synchronous/Asynchronous Serial Interface USART
- MSSP Peripheral for I^2C and SPI communications
- Capture/Compare and PWM modules
- Analog-to-digital converters (up to ~1.0 MHz)
- USB, Ethernet, CAN interfacing support
- External memory interface
- Integrated analog RF front ends (PIC16F639, and rfPIC).
- KEELOQ Rolling code encryption peripheral (encode/decode)
- And many more

6.4.2 Trends

The first generation of PICs with EPROM storage are almost completely replaced by chips with Flash memory. Likewise, the original 12-bit instruction set of the PIC1650 and its direct descendants has been superseded by 14-bit and 16-bit instruction sets. Microchip still sells OTP (one-time-programmable) and windowed (UV-erasable) versions of some of its EPROM based PICs for legacy support or volume orders. The Microchip website lists PICs that are not electrically erasable as OTP. UV erasable windowed versions of these chips can be ordered.

6.4.3 Part number suffixes

The F in a name generally indicates the PICmicro uses flash memory and can be erased electronically. Conversely, a C generally means it can only be erased by exposing the die to ultraviolet light (which is only possible if a windowed package style is used). An exception to this rule is the PIC16C84 which uses EEPROM and is therefore electrically erasable.

An L in the name indicates the part will run at a lower voltage, often with frequency limits imposed.[22]

Parts designed specifically for low voltage operation, within a strict range of 3 - 3.6 volts, are marked with a J in the part number. These parts are also uniquely I/O tolerant as they will accept up to 5 V as inputs.[22]

6.4.4 PIC clones

Third party manufacturers make compatible products, for example the Parallax SX.

6.5 Development tools

Microchip provides a freeware IDE package called MPLAB, which includes an assembler, linker, software simulator, and debugger. They also sell C compilers for the PIC18 and dsPIC which integrate cleanly with MPLAB. Free student versions of the C compilers are also available with all features. But for the free versions, optimizations will be disabled after 60 days.[23]

Several third parties make C language compilers for PICs, many of which integrate to MPLAB and/or feature their own IDE. A fully featured compiler for the PICBASIC language to program PIC microcontrollers is available from meLabs, Inc. Mikroelektronika offers PIC compilers in C, Basic and Pascal programming languages.

A graphical programming language, Flowcode, exists capable of programming 8 and 16 bit PIC devices and generating PIC compatible C code. It exists in numerous versions from a free demonstration to a more complete professional edition.

The only opensource compiler for the PIC16 and PIC18 family is the SDCC. It make use of GPutils for linker and assembler tools. A plugin is needed to install them in MPLAB or MPLABX.[24]

Development tools are available for the PIC family under the GPL or other free software or open source licenses.

6.6 Device programmers

Main article: PICKit

Devices called extquotedblprogrammers extquotedbl are traditionally used to get program code into the target PIC. Most PICs that Microchip currently sell feature ICSP (In Circuit Serial Programming) and/or LVP (Low Voltage Programming) capabilities, allowing the PIC to be programmed while it is sitting in the target circuit. ICSP programming is performed using two pins, clock and data, while a high voltage (12V) is present on the Vpp/MCLR pin. Low voltage programming dispenses with the high voltage, but reserves exclusive use of an I/O pin and can therefore be disabled to recover the pin for other uses

(once disabled it can only be re-enabled using high voltage programming).

There are many programmers for PIC microcontrollers, ranging from the extremely simple designs which rely on ICSP to allow direct download of code from a host computer, to intelligent programmers that can verify the device at several supply voltages. Many of these complex programmers use a pre-programmed PIC themselves to send the programming commands to the PIC that is to be programmed. The intelligent type of programmer is needed to program earlier PIC models (mostly EPROM type) which do not support in-circuit programming.

Many of the higher end flash based PICs can also self-program (write to their own program memory). Demo boards are available with a small bootloader factory programmed that can be used to load user programs over an interface such as RS-232 or USB, thus obviating the need for a programmer device. Alternatively there is bootloader firmware available that the user can load onto the PIC using ICSP. The advantages of a bootloader over ICSP is the far superior programming speeds, immediate program execution following programming, and the ability to both debug and program using the same cable.

Programmers/debuggers are available directly from Microchip. Third party programmers range from plans to build your own, to self-assembly kits and fully tested ready-to-go units. Some are simple designs which require a PC to do the low-level programming signalling (these typically connect to the serial or parallel port and consist of a few simple components), while others have the programming logic built into them (these typically use a serial or USB connection, are usually faster, and are often built using PICs themselves for control).

The following are the official PICkit programmer/debuggers from Microchip:

- Microchip PICkit1
- Microchip PICkit2
- Microchip PICkit3

6.6.1 PICKit 2 clones and open source

PICKit 2 has been an interesting PIC programmer from Microchip. It can program most PICs and debug most of the PICs (as of May-2009, only the PIC32 family is not supported for MPLAB debugging). Ever since its first releases, all software source code (firmware, PC application) and hardware schematic are open to the public. This makes it relatively easy for an end user to modify the programmer for use with a non-Windows operating system such as Linux or Mac OS. In the mean time, it also creates lots of DIY interest and clones. This open source structure brings many features to the PICKit 2 community such as Programmer-to-Go, the UART Tool and the Logic Tool, which have been contributed by PICKit 2 users. Users have also added such features to the PICKit 2 as 4MB Programmer-to-go capability, USB buck/boost circuits, RJ12 type connectors and others.

6.7 Debugging

6.7.1 Software emulation

Commercial and free emulators exist for the PIC family processors.

6.7.2 In-circuit debugging

Later model PICs feature an ICD (in-circuit debugging) interface, built into the CPU core. ICD debuggers (MPLAB ICD2 and other third party) can communicate with this interface using three lines. This cheap and simple debugging system comes at a price however, namely limited breakpoint count (1 on older pics 3 on newer PICs), loss of some IO (with the exception of some surface mount 44-pin PICs which have dedicated lines for debugging) and loss of some features of the chip. For small PICs, where the loss of IO caused by this method would be unacceptable, special headers are made which are fitted with PICs that have extra pins specifically for debugging.

6.7.3 In-circuit emulators

Microchip offers three full in-circuit emulators: the MPLAB ICE2000 (parallel interface, a USB converter is available); the newer MPLAB ICE4000 (USB 2.0 connection); and most recently, the REAL ICE. All of these ICE tools can be used with the MPLAB IDE for full source-level debugging of code running on the target.

The ICE2000 requires emulator modules, and the test hardware must provide a socket which can take either an emulator module, or a production device.

The REAL ICE connects directly to production devices which support in-circuit emulation through the PGC/PGD programming interface, or through a high speed connection which uses two more pins. According to Microchip, it supports “most” flash-based PIC, PIC24, and dsPIC processors.[25]

The ICE4000 is no longer directly advertised on Microchip’s website, and the purchasing page states that it is not recommended for new designs.

6.8 Operating systems

An open source project by Serge Vakulenko adapts 2.11BSD to the PIC32 architecture, under the name RetroBSD. This brings a familiar Unix-like operating system, including an on board development environment, to the microcontroller, within the constraints of the onboard hardware.[26]

6.9 See also

- PIC16x84
- Atmel AVR
- Arduino
- BASIC Atom
- BASIC Stamp
- OOPic
- PICAXE
- TI MSP430
- Maximite

6.10 References

[1] http://ww1.microchip.com/downloads/en/DeviceDoc/39630C.pdf

[2] http://www.datasheetarchive.com/dl/Databooks-1/Book241-407.pdf

[3] "PICmicro Family Tree", PIC16F Seminar Presentation http://www.microchip.com.tw/PDF/2004_spring/PIC16F%20seminar%20presentation.pdf

[4] "MOS DATA 1976", General Instrument 1976 Databook

[5] "1977 Data Catalog", Micro Electronics from General Instrument Corporation http://www.rhoent.com/pic16xx.pdf

[6] Microchip press release. "Microchip Technology Delivers 12 Billionth PIC® Microcontroller to Leading Motor Manufacturer, Nidec Corporation". 2013.

[7] http://ww1.microchip.com/downloads/en/DeviceDoc/35007b.pdf

[8] "AN869: External Memory Interfacing Techniques for the PIC18F8XXX". Retrieved 24 August 2009.

[9] Microchip Product Selector

[10] "PIC Paging and PCLATH"

[11] "MPLAB® XC: Compiler Solutions"

[12] *PIC10F200/202/204/206 Data Sheet*. Microchip Technology. 2007. p. 52.

[13] http://www.emc.com.tw/eng/products.asp

[14] ELAN Microelectronics Corp. (September 2005), *EM78P157N 8-bit microcontroller with OTP ROM Product Specification*, retrieved 2012-04-02

[15] http://www.microchipc.com/sourcecode/

[16] Microchip Technology, Inc. (2007), *PIC18F1220/1320 Data Sheet*, retrieved 2012-04-02

[17]

[18] "PIC24H Family Overview". Retrieved 23 September 2007.

[19] *dsPIC30F Programmer's Reference Manual*, Microchip Technology, 2008, DS70157C, retrieved 2012-07-02

[20] http://www.mips.com/products/processors/32-64-bit-cores/mips32-m4k/

[21] "32-bit PIC MCUs". Retrieved 13 October 2010.

[22] "3V Design Center". Retrieved 2 August 2011.

[23] "MPLAB C Compiler for PIC18 MCUs".

[24] "SDCC plugin for MPLABX".

[25] "MPLAB REAL ICE In-Circuit Emulator Product Overview". Retrieved 23 September 2007.

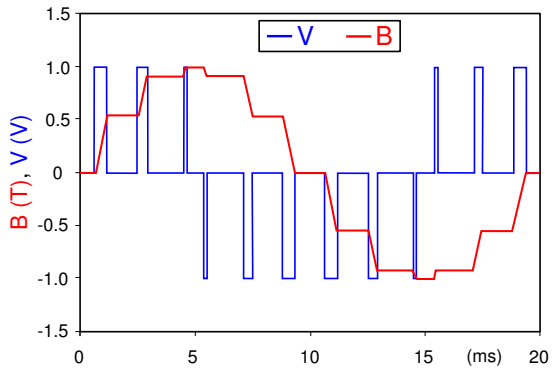
[26] RetroBSD start

6.11 External links

- PIC microcontroller at DMOZ.
- Official Microchip website
- PIC wifi projects website

Chapter 7

Pulse-width modulation



An example of PWM in an AC motor drive: the phase-to-phase voltage (blue) is modulated as a series of pulses that results in a sine-like flux density waveform (red) in the magnetic circuit of the motor. The smoothness of the resultant waveform can be controlled by the width (number) of modulated impulses (per given cycle)

Pulse-width modulation (**PWM**), or **pulse-duration modulation** (**PDM**), is a modulation technique that controls the width of the pulse, formally the pulse duration, based on modulator signal information. Although this modulation technique can be used to encode information for transmission, its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors. In addition, PWM is one of the two principal algorithms used in photovoltaic solar battery chargers,[1] the other being MPPT.

The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast pace. The longer the switch is on compared to the off periods, the higher the power supplied to the load.

The PWM switching frequency has to be much higher than what would affect the load (the device that uses the power), which is to say that the resultant waveform perceived by the load must be as smooth as possible. Typically switching has to be done several times a minute in an electric stove, 120 Hz in a lamp dimmer, from few kilohertz (kHz) to tens of kHz for a motor drive and well into the tens or hundreds of kHz in audio amplifiers and computer power supplies.

The term *duty cycle* describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on.

The main advantage of PWM is that power loss in the switching devices is very low. When a switch is off there is practically no current, and when it is on and power is being transferred to the load, there is almost no voltage drop across the switch. Power loss, being the product of voltage and current, is thus in both cases close to zero. PWM also works well with digital controls, which, because of their on/off nature, can easily set the needed duty cycle.

PWM has also been used in certain communication systems where its duty cycle has been used to convey information over a communications channel.

7.1 History

In the past, when only partial power was needed (such as for a sewing machine motor), a rheostat (located in the sewing machine's foot pedal) connected in series with the motor adjusted the amount of current flowing through the motor, but also wasted power as heat in the resistor element. It was an inefficient scheme, but tolerable because the total power was low. This was one of several methods of controlling power. There were others—some still in use—such as variable autotransformers, including the trademarked 'Autrastat' for theatrical lighting; and the Variac, for general AC power adjustment. These were quite efficient, but also relatively costly.

For about a century, some variable-speed electric motors have had decent efficiency, but they were somewhat more complex than constant-speed motors, and sometimes required bulky external electrical apparatus, such as a bank of variable power resistors or rotating converter such as Ward Leonard drive.

However, in addition to motor drives for fans, pumps and robotic servos, there was a great need for compact and low cost means for applying adjustable power for many devices, such as electric stoves and lamp dimmers.

One early application of PWM was in the Sinclair X10, a

10 W audio amplifier available in kit form in the 1960s. At around the same time PWM started to be used in AC motor control.[2]

7.2 Principle

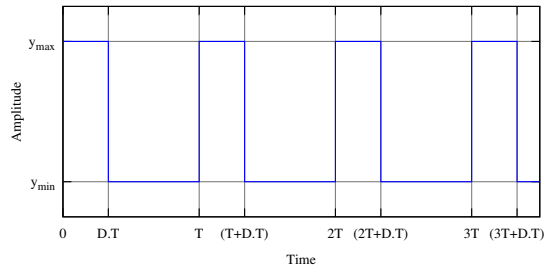


Fig. 1: a pulse wave, showing the definitions of y_{min} , y_{max} and D.

Pulse-width modulation uses a rectangular pulse wave whose pulse width is modulated resulting in the variation of the average value of the waveform. If we consider a pulse waveform $f(t)$, with period T , low value y_{min} , a high value y_{max} and a duty cycle D (see figure 1), the average value of the waveform is given by:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt.$$

As $f(t)$ is a pulse wave, its value is y_{max} for $0 < t < D \cdot T$ and y_{min} for $D \cdot T < t < T$. The above expression then becomes:

$$\begin{aligned}\bar{y} &= \frac{1}{T}\left(\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt\right) \\ &= \frac{D \cdot T \cdot y_{max} + T\,(1-D)\,y_{min}}{T} \\ &= D \cdot y_{max} + (1-D)\,y_{min}.\end{aligned}$$

This latter expression can be fairly simplified in many cases where $y_{min} = 0$ as $\bar{y} = D \cdot y_{max}$. From this, it is obvious that the average value of the signal ($\bar{y}$) is directly dependent on the duty cycle D.

The simplest way to generate a PWM signal is the intersective method, which requires only a sawtooth or a triangle waveform (easily generated using a simple oscillator) and a comparator. When the value of the reference signal (the red sine wave in figure 2) is more than the modulation waveform (blue), the PWM signal (magenta) is in the high state, otherwise it is in the low state.

7.2.1 Delta

Main article: Delta modulation

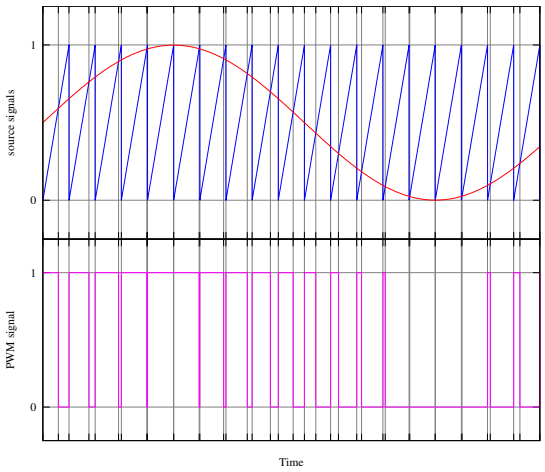


Fig. 2: A simple method to generate the PWM pulse train corresponding to a given signal is the intersective PWM: the signal (here the red sinewave) is compared with a sawtooth waveform (blue). When the latter is less than the former, the PWM signal (magenta) is in high state (1). Otherwise it is in the low state (0).

In the use of delta modulation for PWM control, the output signal is integrated, and the result is compared with limits, which correspond to a Reference signal offset by a constant. Every time the integral of the output signal reaches one of the limits, the PWM signal changes state. Figure 3

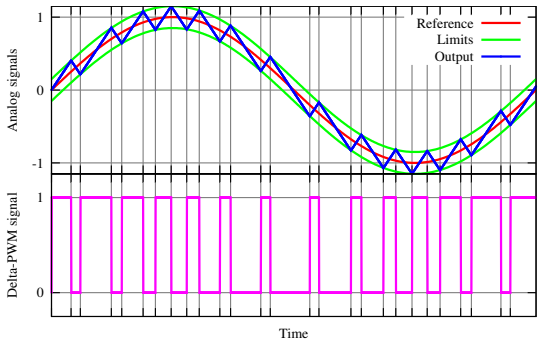


Fig. 3 : Principle of the delta PWM. The output signal (blue) is compared with the limits (green). These limits correspond to the reference signal (red), offset by a given value. Every time the output signal (blue) reaches one of the limits, the PWM signal changes state.

7.2.2 Delta-sigma

Main article: Delta-sigma modulation

In delta-sigma modulation as a PWM control method, the output signal is subtracted from a reference signal to form an error signal. This error is integrated, and when the integral of the error exceeds the limits, the output changes state. Figure 4

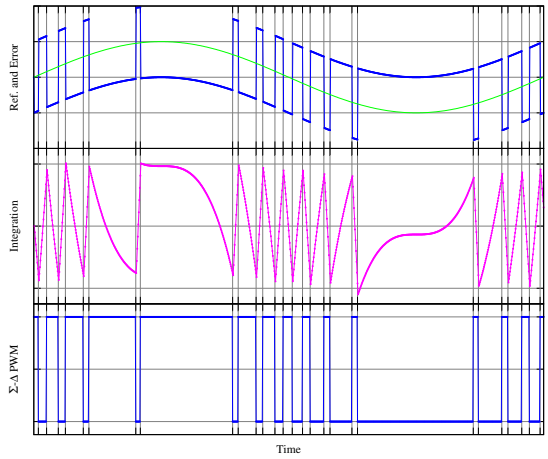


Fig. 4 : Principle of the sigma-delta PWM. The top green waveform is the reference signal, on which the output signal (PWM, in the bottom plot) is subtracted to form the error signal (blue, in top plot). This error is integrated (middle plot), and when the integral of the error exceeds the limits (red lines), the output changes state.

7.2.3 Space vector modulation

Main article: Space vector modulation

Space vector modulation is a PWM control algorithm for multi-phase AC generation, in which the reference signal is sampled regularly; after each sample, non-zero active switching vectors adjacent to the reference vector and one or more of the zero switching vectors are selected for the appropriate fraction of the sampling period in order to synthesize the reference signal as the average of the used vectors.

7.2.4 Direct torque control (DTC)

Main article: Direct torque control

Direct torque control is a method used to control AC motors. It is closely related with the delta modulation (see above). Motor torque and magnetic flux are estimated and these are controlled to stay within their hysteresis bands by turning on new combination of the device's semiconductor switches each time either of the signal tries to deviate out of the band.

7.2.5 Time proportioning

Many digital circuits can generate PWM signals (e.g., many microcontrollers have PWM outputs). They normally use a counter that increments periodically (it is connected directly or indirectly to the clock of the circuit) and is reset at the end of every period of the PWM. When the counter value is more than the reference value, the PWM output changes state from high to low (or low to high).[3] This technique is referred to as **time proportioning,** particularly as **time-proportioning control**[4] – which *proportion* of a fixed cycle time is spent in the high state.

The incremented and periodically reset counter is the discrete version of the intersecting method's sawtooth. The analog comparator of the intersecting method becomes a simple integer comparison between the current counter value and the digital (possibly digitized) reference value. The duty cycle can only be varied in discrete steps, as a function of the counter resolution. However, a high-resolution counter can provide quite satisfactory performance.

7.2.6 Types

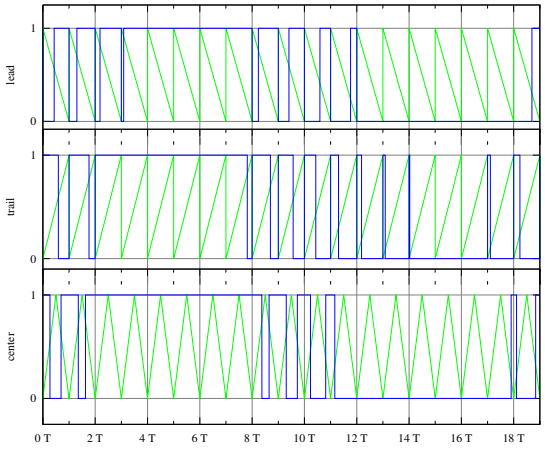


Fig. 5 : Three types of PWM signals (blue): leading edge modulation (top), trailing edge modulation (middle) and centered pulses (both edges are modulated, bottom). The green lines are the sawtooth waveform (first and second cases) and a triangle waveform (third case) used to generate the PWM waveforms using the intersective method.

Three types of pulse-width modulation (PWM) are possible:

1. The pulse center may be fixed in the center of the time window and both edges of the pulse moved to compress or expand the width.
2. The lead edge can be held at the lead edge of the window and the tail edge modulated.
3. The tail edge can be fixed and the lead edge modulated.

7.2.7 Spectrum

The resulting spectra (of the three cases) are similar, and each contains a dc component, a base sideband containing the modulating signal and phase modulated carriers at

each harmonic of the frequency of the pulse. The amplitudes of the harmonic groups are restricted by a $\sin x/x$ envelope (sinc function) and extend to infinity. The infinite bandwidth is caused by the nonlinear operation of the pulse-width modulator. In consequence, a digital PWM suffers from aliasing distortion that significantly reduce its applicability for modern communications system. By limiting the bandwidth of the PWM kernel, aliasing effects can be avoided.[5]

On the contrary, the delta modulation is a random process that produces continuous spectrum without distinct harmonics.

7.2.8 PWM sampling theorem

The process of PWM conversion is non-linear and it is generally supposed that low pass filter signal recovery is imperfect for PWM. The PWM sampling theorem[6] shows that PWM conversion can be perfect. The theorem states that "Any bandlimited baseband signal within ±0.637 can be represented by a pulsewidth modulation (PWM) waveform with unit amplitude. The number of pulses in the waveform is equal to the number of Nyquist samples and the peak constraint is independent of whether the waveform is two-level or three-level."

7.3 Applications

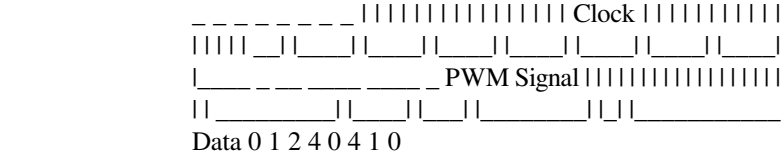
7.3.1 Servos

PWM is used to control servomechanisms, see servo control.

7.3.2 Telecommunications

In telecommunications, PWM is a form of signal modulation where the widths of the pulses correspond to specific data values encoded at one end and decoded at the other.

Pulses of various lengths (the information itself) will be sent at regular intervals (the carrier frequency of the modulation).



The inclusion of a clock signal is not necessary, as the leading edge of the data signal can be used as the clock if a small offset is added to the data value in order to avoid a data value with a zero length pulse.

7.3.3 Power delivery

PWM can be used to control the amount of power delivered to a load without incurring the losses that would result from linear power delivery by resistive means. Potential drawbacks to this technique are the pulsations defined by the duty cycle, switching frequency and properties of the load. With a sufficiently high switching frequency and, when necessary, using additional passive electronic filters, the pulse train can be smoothed and average analog waveform recovered.

High frequency PWM power control systems are easily realisable with semiconductor switches. As explained above, almost no power is dissipated by the switch in either on or off state. However, during the transitions between on and off states, both voltage and current are nonzero and thus power is dissipated in the switches. By quickly changing the state between fully on and fully off (typically less than 100 nanoseconds), the power dissipation in the switches can be quite low compared to the power being delivered to the load.

Modern semiconductor switches such as MOSFETs or Insulated-gate bipolar transistors (IGBTs) are well suited components for high efficiency controllers. Frequency converters used to control AC motors may have efficiencies exceeding 98%. Switching power supplies have lower efficiency due to low output voltage levels (often even less than 2 V for microprocessors are needed) but still more than 70–80% efficiency can be achieved.

Variable-speed fan controllers for computers usually use PWM, as it is far more efficient when compared to a potentiometer or rheostat. (Neither of the latter is practical to operate electronically; they would require a small drive motor.)

Light dimmers for home use employ a specific type of PWM control. Home-use light dimmers typically include electronic circuitry which suppresses current flow during defined portions of each cycle of the AC line voltage. Adjusting the brightness of light emitted by a light source is then merely a matter of setting at what voltage (or phase) in the AC halfcycle the dimmer begins to provide electrical current to the light source (e.g. by using an electronic switch such as a triac). In this case the PWM duty cycle is the ratio of the conduction time to the duration of the half AC cycle defined by the frequency of the AC line voltage (50 Hz or 60 Hz depending on the country).

These rather simple types of dimmers can be effectively used with inert (or relatively slow reacting) light sources such as incandescent lamps, for example, for which the additional modulation in supplied electrical energy which is caused by the dimmer causes only negligible additional fluctuations in the emitted light. Some other types of light sources such as light-emitting diodes (LEDs), how-

ever, turn on and off extremely rapidly and would perceivably flicker if supplied with low frequency drive voltages. Perceivable flicker effects from such rapid response light sources can be reduced by increasing the PWM frequency. If the light fluctuations are sufficiently rapid, the human visual system can no longer resolve them and the eye perceives the time average intensity without flicker (see flicker fusion threshold).

In electric cookers, continuously variable power is applied to the heating elements such as the hob or the grill using a device known as a Simmerstat. This consists of a thermal oscillator running at approximately two cycles per minute and the mechanism varies the duty cycle according to the knob setting. The thermal time constant of the heating elements is several minutes, so that the temperature fluctuations are too small to matter in practice.

7.3.4 Voltage regulation

Main article: Switched-mode power supply

PWM is also used in efficient voltage regulators. By switching voltage to the load with the appropriate duty cycle, the output will approximate a voltage at the desired level. The switching noise is usually filtered with an inductor and a capacitor.

One method measures the output voltage. When it is lower than the desired voltage, it turns on the switch. When the output voltage is above the desired voltage, it turns off the switch.

7.3.5 Audio effects and amplification

PWM is sometimes used in sound (music) synthesis, in particular subtractive synthesis, as it gives a sound effect similar to chorus or slightly detuned oscillators played together. (In fact, PWM is equivalent to the difference of two sawtooth waves with one of them inverted.) The ratio between the high and low level is typically modulated with a low frequency oscillator. In addition, varying the duty cycle of a pulse waveform in a subtractive-synthesis instrument creates useful timbral variations. Some synthesizers have a duty-cycle trimmer for their square-wave outputs, and that trimmer can be set by ear; the 50% point (true square wave) was distinctive, because even-numbered harmonics essentially disappear at 50%. Pulse waves, usually 50%, 25%, and 12.5%, make up the soundtracks of classic video games.

A new class of audio amplifiers based on the PWM principle is becoming popular. Called extquotedblClass-D amplifiers extquotedbl, they produce a PWM equivalent of the analog input signal which is fed to the loudspeaker via a suitable filter network to block the carrier and recover the original audio. These amplifiers are characterized by very good efficiency figures (≥ 90%) and compact size/light weight for large power outputs. For a few decades, industrial and military PWM amplifiers have been in common use, often for driving servo motors. Field-gradient coils in MRI machines are driven by relatively high-power PWM amplifiers.

Historically, a crude form of PWM has been used to play back PCM digital sound on the PC speaker, which is driven by only two voltage levels, typically 0 V and 5 V. By carefully timing the duration of the pulses, and by relying on the speaker's physical filtering properties (limited frequency response, self-inductance, etc.) it was possible to obtain an approximate playback of mono PCM samples, although at a very low quality, and with greatly varying results between implementations.

In more recent times, the Direct Stream Digital sound encoding method was introduced, which uses a generalized form of pulse-width modulation called pulse density modulation, at a high enough sampling rate (typically in the order of MHz) to cover the whole acoustic frequencies range with sufficient fidelity. This method is used in the SACD format, and reproduction of the encoded audio signal is essentially similar to the method used in class-D amplifiers.

7.3.6 Electrical

SPWM (Sine–triangle pulse width modulation) signals are used in micro-inverter design (used in solar or wind power applications). These switching signals are fed to the FETs that are used in the device. The device's efficiency depends on the harmonic content of the PWM signal. There is much research on eliminating unwanted harmonics and improving the fundamental strength, some of which involves using a modified carrier signal instead of a classic sawtooth signal [7][8][9] in order to decrease power losses and improve efficiency. Another common application is in robotics where PWM signals are used to control the speed of the robot by controlling the motors.

7.4 See also

- Delta-sigma modulation
- Pulse-amplitude modulation
- Pulse-code modulation
- Pulse-density modulation
- Pulse-position modulation
- Radio control
- RC servo
- Sliding mode control - produces smooth behavior by way of discontinuous switching in systems

- Space vector modulation
- Class-D amplifier

7.5 References

[1] http://www.homepower.com/articles/solar-electricity/design-installation/sizing-grid-tied-pv-system-battery-backup

[2] Schönung, A.; Stemmler, H. (August 1964). "Geregelter Drehstrom-Umkehrantrieb mit gesteuertem Umrichter nach dem Unterschwingungsverfahren". *BBC Mitteilungen* (Brown Boveri et Cie) **51** (8/9): 555–577.

[3] www.netrino.com – Introduction to Pulse Width Modulation (PWM)

[4] *Fundamentals of HVAC Control Systems,* by Robert McDowall, p. 21

[5] Hausmair, Katharina; Shuli Chi; Peter Singerl; Christian Vogel (February 2013). "Aliasing-Free Digital Pulse-Width Modulation for Burst-Mode RF Transmitters". *IEEE Transactions on Circuits and Systems I: Regular Papers* **60** (2): 415–427. doi:10.1109/TCSI.2012.2215776.

[6] J. Huang, K. Padmanabhan, and O. M. Collins, "The sampling theorem with constant amplitude variable width pulses", IEEE transactions on Circuits and Systems, vol. 58, pp. 1178 - 1190, June 2011.

[7] Hirak Patangia, Sri Nikhil Gupta Gourisetti, "A Harmonically Superior Modulator with Wide Baseband and Real-Time Tunability", IEEE International Symposium on Electronic Design (ISED), India, Dec.11.

[8] Hirak Patangia, Sri Nikhil Gupta Gourisetti, "Real Time Harmonic Elimination Using a Modified Carrier", CONIELECOMP, Mexico, Feb 2012.

[9] Hirak Patangia, Sri Nikhil Gupta Gourisetti, "A Novel Strategy for Selective Harmonic Elimination Based on a Sine-Sine PWM Model", MWSCAS, U.S.A, Aug 2012.

7.6 External links

- An Introduction to Delta Sigma Converters
- Introductory Tutorial on PWM and Quadrature Encoding
- Pulse Width Modulation using 555 Timer
- Pulse Width Modulation in PID control loop - free simulator

7.7 Text and image sources, contributors, and licenses

7.7.1 Text

- **Freescale 68HC11** *Source:* http://en.wikipedia.org/wiki/Freescale_68HC11?oldid=628185196 *Contributors:* Stephen Gilbert, Ellmist, Palnatoke, Stan Shebs, Wernher, Altenmann, PlatinumX, TobinFricke, 2fargon, James Foster, (aeropagitica), Wdfarmer, Brycen, Woohookitty, Graham87, SLi, Mirror Vax, YurikBot, Sverre, LanguidMandala, Unyoyega, Chris the speller, Bluebot, TimBentley, Can't sleep, clown will eat me, Cybercobra, Loadmaster, Dicklyon, Raidfibre, Pipatron, Tonymac32, Choppingmall, Calltech, RockMFR, Kyle the bot, Sarenne, Andy Dingley, SieBot, Dbryant 94560, Lightmouse, Frappucino, Alexbot, Joel Saks, Tonypdmtr, Legobot, Luckas-bot, Amirobot, Rfbb, Eumolpo, KuroiShiroi, Tomekdcd, GerbilSoft, Lumag Lumag, No1anyoneknows, Rocketrod1960, Iswantoumy, Rosso-Malpelo, Kahtar and Anonymous: 35
- **Intel 8085** *Source:* http://en.wikipedia.org/wiki/Intel_8085?oldid=624200412 *Contributors:* Khendon, Scipius, Mahjongg, Dave Farquhar, Ixfd64, Stan Shebs, Berteun, Wernher, AnonMoos, Donarreiskoffer, Vespristiano, Adam Sampson, Mboverload, Kusunose, Sam Hocevar, Moxfyre, Imroy, Roo72, Ttguy, Kghose, FyreFiend, Cmdrjameson, Polluks, Photonique, Hooperbloob, Jumbuck, Nazli, Guy Harris, Ashley Pomeroy, Pravs, Wtshymanski, Jannex, Wdyoung, Tabletop, Alecv, Sattish, Quale, Commander, StuartBrady, FlaBot, Toresbe, Dresdnhope, YurikBot, RobotE, Armistej, Yuhong, Rsrikanth05, Omniwriter, Lomn, Vlad, Uwezi, 2fort5r, Allens, KnightRider, SmackBot, Unyoyega, Eskimbot, Gilliam, NCurse, Thumperward, Frap, Wine Guy, Morio, H3g3m0n, Ohconfucius, Dmit, Loadmaster, Beetstra, Nwwaew, Onkarshinde, Manurastogi, Saulo35, HenkeB, Future Perfect at Sunrise, Viscious81, Thijs!bot, TheJosh, Vodomar, Escarbot, Guy Macon, Jj137, Smartse, Rforums, Rossaxe, Iwick, RastaKins, Vibhav Chauhan, Magioladitis, Bongwarrior, Diego bf109, PeterAStoll, Mustafa1702, Tracer9999, STBot, Kateshortforbob, J.delanoy, Useight, Jeff G., Linefeed, Silpsilp, WinTakeAll, Meters, Adam.J.W.C., Jpeeling, Spinningspark, Wjl2, SieBot, Milnivri, Lightmouse, ClueBot, Rilak, MikeVitale, Sumanthsr25, Niceguyedc, Kumar amit p, Jotterbot, DumZiBoT, Templarion, XLinkBot, Parallelized, Addbot, Mortense, JoshuaKuo, Fireaxe888, Lightbot, Matthew Anthony Smith, Cleberz, LuK3, Luckas-bot, Yobot, Amirobot, Muthuramanece, Junaidpv, Freezykid, Materialscientist, Ilija139, Kellogg257, Qorilla, LilHelpa, NSK Nikolaos S. Karastathis, Djomla88, JWBE, Victor Waiman, Prari, Thorenn, Mail shan, ZenerV, Kwiki, Arndbergmann, Karna079, John Elson, SpaceFlight89, Jm61288, DARTH SIDIOUS 2, Kbjayashankar, EmausBot, Dewritech, Ibbn, RenamedUser01302013, Cogiati, Bhanuvrat, A930913, Prerna maheshwari, Spnro, WIMIA, DASHBotAV, ClueBot NG, Matthiaspaul, Newyorkadam, Helpful Pixie Bot, Wbm1058, Jphill19, Barewires, Ranjithfs1, Vanangamudiyan, Wiki13, Pdesousa359, Vikram.gurve, Kokkkikumar, Morning Sunshine, Epicgenius, Pseudonymous Rex, Shreyaacharya1990, Comp.arch, Kahtar, Poepkop and Anonymous: 268
- **Intel 8086** *Source:* http://en.wikipedia.org/wiki/Intel_8086?oldid=629311538 *Contributors:* Matthew Woodcraft, Vulture, Bryan Derksen, Stephen Gilbert, Nate Silva, William Avery, RTC, Michael Hardy, Cprompt, Mahjongg, Ixfd64, Egil, Ahoerstemeier, Andres, Cimon Avaro, Hpa, Furrykef, Grendelkhan, Saltine, Wernher, BenRG, Donarreiskoffer, Robbot, Yarvin, Rursus, Trevor Johns, Wereon, SamB, Mintleaf, Levin, Ferkelparade, Brona, Fleminra, Neilc, Kevins, Bumm13, TheObtuseAngleOfDoom, Imroy, Slady, Discospinster, Rich Farmbrough, Jpk, Smyth, Thomas Willerich, Ivan Bajlo, Martpol, Djordjes, Ht1848, CanisRufus, Jaques O. Carvalho, Trixter, Agoode, Yonghokim, Shenme, Hooperbloob, Nkedel, Alansohn, Guy Harris, Arthena, Denniss, Wtshymanski, Kbolino, Thryduulf, Woohookitty, Mindmatrix, Timharwoodx, Jeff3000, Damicatz, GregorB, Isnow, Alecv, Azkar, Graham87, Qwertyus, NeonMerlin, FlaBot, Quuxplusone, Swtpc6800, Mathrick, Butros, Chobot, DVdm, YurikBot, Wavelength, Hairy Dude, Pigman, Yuhong, Thane, DragonHawk, Megapixie, Richardcavell, Iambk, KGasso, Fourohfour, SmackBot, Jagged 85, Eaglizard, Brianski, LostArtilleryman, Amatulic, Keegan, Sandycx, Jerome Charles Potts, Chisophugis, Idallen, Милан Јелисавчић, Frap, OrphanBot, Rrburke, Radagast83, Cybercobra, Shadow1, Morio, Vina-iwbot, SpareHeadOne, John, Natarajuab, Jeberle, Coredesat, Mahinthjoe, IronGargoyle, Loadmaster, Illythr, Camp3rstrik3r, DJMalone, Mdanh2002, DabMachine, Fan-1967, Asmpgmr, Sul4bh, CmdrObot, Memetics, Van helsing, HenkeB, Emilio Juanatey, TimmyRaa, Thijs!bot, Al Lemos, TheJosh, AntiVandalBot, Seaphoto, Uvaphdman, Edokter, MECU, Alphachimpbot, DOSGuy, Nthep, Bongwarrior, VoABot II, PeterAStoll, JaGa, GermanX, Gwern, Ginsengbomb, Aryoc, Jerry, MJStadler, Belovedfreak, Potatoswatter, Useight, DanielVerkamp, Imperator3733, PGSONIC, Kww, TheThiefMaster, Hqb, Sarenne, Nxavar, LiveOnAPlane, WinTakeAll, Lerdthenerd, Andy Dingley, Adam.J.W.C., Mpx, Fnagaton, Sonicology, Aesthetic.online, WereSpielChequers, Raffzahn, Lightmouse, Dust Filter, Treekids, ClueBot, Lonegroover, MikeVitale, Niceguyedc, TheSmuel, The 888th Avatar, Auntof6, Pointillist, Microprofessor, Excirial, Jotterbot, 8400sx, Callmejosh, Aitias, SoxBot III, Galzigler, PL290, Farmdogg, Thebestofall007, Addbot, Yousou, Dk pdx, Magus732, Saurabh desire, Debresser, AtheWeatherman, Tide rolls, Matthew Anthony Smith, Matt.T, Yobot, Bunnyhop11, Legobot II, Crispmuncher, AnomieBOT, RandomAct, Materialscientist, Atw1996, Лъчезар, Xqbot, JWBE, GrouchoBot, Iceman444k, IShadowed, Shadowjams, Endothermic, FrescoBot, ZenerV, Arndbergmann, Hoo man, Praveen.giluka, Alex146, FoxBot, Seahorseruler, Ybungalobill, Jfmantis, RjwilmsiBot, WinContro, EmausBot, Immunize, Ibbn, ZéroBot, Resh123, H3llBot, L Kensington, Avivanov76, Wallentis, Kevin J Chase, Kevindass, Mikhail Ryazanov, ClueBot NG, Naren2010, Matthiaspaul, CocuBot, Snotbot, DieSwartzPunkt, Helpful Pixie Bot, Wbm1058, Jphill19, Pdesousa359, Andrzej w k 2, Digital Brains, FootholdTechnology, ZaferXYZ, Rob491, Kahtar, Bert Freudenberg, Sofia Koutsouveli, Tshubham and Anonymous: 332
- **Intel MCS-51** *Source:* http://en.wikipedia.org/wiki/Intel_MCS-51?oldid=627801615 *Contributors:* Ellmist, Stan Shebs, Ronz, GRAHAMUK, Arteitle, Emperorbma, Wernher, Robbot, Tobias Bergemann, Roger Irwin, DavidCary, Wmahan, Chowbok, Sam Hocevar, Flex, Ljosa, Nabla, CanisRufus, Simon South, R. S. Shaw, Minghong, Alansohn, Corwin8, Wtshymanski, Cburnett, Kgrr, Isnow, MarkusHagenlocher, Rjwilmsi, Miha Ulanov, Wragge, Mirror Vax, Bgwhite, YurikBot, Matanya (renamed), Armistej, DragonHawk, Dhollm, David Biddulph, RunOrDie, Veinor, SmackBot, John Lunney, Nihonjoe, Maelwys, Rhondle, CSWarren, Chlewbot, OrphanBot, Nishkid64, Rait, Dicklyon, Riordanmr, Pfagerburg, Kimjoarr, ShelfSkewed, HenkeB, Shreyasjoshis, Wsmarz, Solidpoint, Thijs!bot, Racaille, Electron9, Modal, MichaelFrey, Guy Macon, Bobke, Makkwong, Rushikeshshinde, JAnDbot, RastaKins, Ljudina, JamesBWatson, Choppingmall, Mustafa1702, Sajupa, Calltech, Gwern, STBotD, Enivid, VolkovBot, Flyte35, Someguy1221, MauriceS, Don4of4, LeaveSleaves, SQL, Thunderbird2, Abhi3385, VVVBot, Jerryobject, Antzervos, Lightmouse, Nyelvmark, Frappucino, Ken123BOT, Tuxa, Niceguyedc, Pointillist, Plaes, Muro Bot, Amolhshah, Joel Saks, XLinkBot, Flipmode fly, Asaco, Addbot, Rogue780, Mortense, CactusWriter, MrOllie, Lightbot, Yobot, Crispmuncher, Raj591, AnomieBOT, Kushagraalankar, Citation bot, ArthurBot, MauritsBot, Xqbot, TheAMmollusc, Capricorn42, Armstrong1113149, JWBE, Victor Waiman, Edsim51, Aaditya 7, FrescoBot, Oldlaptop321, Plindemann, Glider87, Outback the koala, Tomekdcd, RedBot, MaxDel, Mohitjoshi999, Songsing77, Sarojlucky, Dead Horsey, Dewritech, AndroidX1, Dcirovic, Ebrambot, Kevjonesin, Sbmeirow, Spacetrucker23, Lazar.Elena, ClueBot NG, Kasirbot, Widr, Helpful Pixie Bot, Microheat, DBigXray, Tailor-tinker, Dzlinker, Srinathkr3, Flutte, Ndzervas, Fugmarsh, Tagremover, Surjansh, Dexbot, Ambiguous Furry Rocking Thing, Frosty, Justme8910, Nvtj, Comp.arch, Kahtar, Monkbot and Anonymous: 202
- **Motorola 6800** *Source:* http://en.wikipedia.org/wiki/Motorola_6800?oldid=628253470 *Contributors:* Stephen Gilbert, Maury Markowitz, Ellmist, Gregben, Mahjongg, Egil, Stan Shebs, Ww, Furrykef, Wernher, FlyByPC, Archivist, Korath, Merovingian, Mdrejhon, Leonard

G., Revth, Tooki, Hugh Mason, Wrp103, Pixel8, Moki80, Bender235, Jonathan Drain, Foobaz, Hooperbloob, WideArc, Sverde1, Wtshymanski, Suruena, Morkork, TreveX, Alecv, Marudubshinki, Graham87, Miq, Rjwilmsi, Pbnelson, FlaBot, Swtpc6800, Viznut, YurikBot, Armistej, Art Navsegda, Gaius Cornelius, JulesH, Pelladon, LanguidMandala, Cojoco, SmackBot, Chris the speller, McNeight, Милан Јелисавчић, RCX, Loadmaster, Eurodog, Oswald Glinkmeyer, Dicklyon, GhostInTheMachine, Simonmcc, TJ Spyke, Iridescent, HenkeB, Myasuda, Orion Blastar, Aldis90, Al Lemos, Electron9, Humble Scribe, Tedickey, Daibot, R'n'B, Nono64, Jhallenworld, STBotD, ChrisWar666, Urbancamo, Billinghurst, Andy Dingley, Plan10, Phe-bot, Jerryobject, Lightmouse, Renan S2, ClueBot, Rilak, Briangjob, Lambtron, Addbot, DOI bot, Magus732, Tothwolf, Leszek Jańczuk, Lightbot, Luckas-bot, KamikazeBot, AnomieBOT, Aneah, LucienBOT, Ballon845, Citation bot 1, Orenburg1, EmausBot, WikitanvirBot, AsceticRose, Floydvirginia, SeattleMurki, Helpful Pixie Bot, ChrisGualtieri, ל, Khazar2, Comp.arch, Monkbot and Anonymous: 68

- **PIC microcontroller** *Source:* http://en.wikipedia.org/wiki/PIC_microcontroller?oldid=627181377 *Contributors:* Maury Markowitz, Azhyd, Heron, Frecklefoot, Michael Hardy, Mahjongg, Ahoerstemeier, Stan Shebs, Mac, Ronz, Glenn, Wik, Zoicon5, Furrykef, Morwen, Val42, Omegatron, Wernher, Izx, Kizor, RedWolf, Chris Roy, Jrash, Brouhaha, DavidCary, Mat-C, Mintleaf, Ds13, Mboverload, Ferdinand Pienaar, Bo102010, Dan Gardner, Uzume, Cashimor, Chowbok, Hellisp, BrianWilloughby, Abdull, Moxfyre, Monkeyman, Imroy, Rich Farmbrough, Alistair1978, ChrisJ, Plugwash, CanisRufus, Adambro, Bobo192, Jfraser, Zetawoof, JesseHogan, Jason One, Alansohn, Redfarmer, Wdfarmer, Velella, Wtshymanski, Tony Sidaway, Dirac1933, Gene Nygaard, Talkie tim, Kinema, Mr z, Nuno Tavares, Firsfron, Tabletop, Stingraze, Gengiskanhg, Alecv, Toussaint, Marudubshinki, Liqk, Rjwilmsi, .digamma, Atomsmith, Avochelm, Vary, MZMcBride, Brighterorange, Toddsoc, FlaBot, Pip11, Intersofia, Alvin-cs, Thunderchild, Chobot, DVdm, Bgwhite, Gwernol, YurikBot, Wavelength, Pip2andahalf, Adam1213, RussBot, SpuriousQ, Gaius Cornelius, Ihope127, Wiki alf, Dogcow, Speedevil, Voidxor, Ma3nocum, Niggurath, Erpingham, Zzuuzz, Morcheeba, Mateo LeFou, Mikemurphy, GraemeL, Allens, Tom Morris, Attilios, SmackBot, Brammers, KnowledgeOfSelf, Bmearns, Jonathanwagner, Gilliam, Smt52, Chris the speller, Tree Biting Conspiracy, Peter todd, DHN-bot, Colonies Chris, A. B., Bsodmike, Frap, Hyvatti, JonHarder, Weopgon, Megamix, TechPurism, Dcamp314, Vic93, Bejnar, Nishkid64, ArglebargleIV, Kuru, Turbo852, RCX, CyrilB, Mr Stephen, Eplondke, Dicklyon, Warnockm, Mdenton, Lmblackjack21, JoeBot, Trialsanderrors, Angelpeream, CmdrObot, PSL, Shoez, Jburstein, HenkeB, Cydebot, Scalerobotics, Stjrna, Nabokov, Plaasjaapie, Hithisishal, Thijs!bot, Bradediger, Electron9, James086, X201, FourBlades, AntiVandalBot, MichaelFrey, Saimhe, Guy Macon, AlexOvShaolin, Gadlen, MortimerCat, JAnDbot, Gcm, Barek, Karlwk, BrotherE, Jahoe, Magioladitis, VoABot II, Ferritecore, Choppingmall, 28421u2232nfenfcenc, Allstarecho, Japo, Matt B., GermanX, Calltech, Gwern, PJohnson, Rettetast, Kateshortforbob, CommonsDelinker, Sbogdanov, Ckielstra, Skeeter2, NightFalcon90909, Javawizard, Reedy Bot, Rod57, Katalaveno, Jhallenworld, Plasticup, Ppapadeas, SJP, Cometstyles, S.riccardelli, WardXmodem, Reelrt, CSumit, Soliloquial, Philip Trueman, EvanCarroll, Una Smith, CanOfWorms, Gfutia, Fabero74, Lerdthenerd, Andy Dingley, Ramsey585, Dirkbb, DavesPlanet, Nat1192, Jd4x4, Insanity Incarnate, Sxpilot250, SieBot, SheepNotGoats, VVVBot, OpBanana, Hawk777, Pcbbc, Jerryobject, Taemyr, Randomblue, Smishek, Bombadier337, Ikmac, ClueBot, LizardJr8, Shjacks45, Sv1xv, Excirial, Davitenio, Jamodio, Devon Sean McCullough, XLinkBot, MohammadEbrahim, NobbiP, Addbot, Mortense, Kevin E Hawkins, TutterMouse, Johnpeterharvey, MrOllie, SamatBot, Tide rolls, Lightbot, CountryBot, Softy, Margin1522, Lrb13615, Washburnmav, Tempodivalse, AnomieBOT, Wjw1961, Materialscientist, Akilaa, Xqbot, Fransschreuder, Wikiwooroo, Browsem, DimychUA, Carveone, Knuckx, Kyxui, Shadowjams, Fixentries, Samwb123, FrescoBot, Miceduan, Goldzen, Outback the koala, Jon-ecm, Lraingele, A little insignificant, Velociostrich, PPA94, Eworldprojects, 10metreh, Strigoides, DiamondDevil, Yunshui, Fiducial, Lauri.pirttiaho, EmausBot, Heracles31, Bernard Teo, RenamedUser01302013, Betatester228, Jwortzel, Sbmeirow, MatthewJBennett, Greglecuyer, Bugfarmer, WimHeirman, Pazza pazza, ClueBot NG, Jaanus.kalde, Aleksandarbrain, Satellizer, Kubing, Matthewmadmad, Danim, Danara5, Gauravsangwan, BG19bot, Barefoottech, Cyanoir, LordSputnik, Dhx1, ZaferXYZ, Alexdzm90, Electracion, Onorai, Jonny wdrw, Comp.arch, Jianhui67, CyanRuby, Div2005, Monkbot and Anonymous: 522
- **Pulse-width modulation** *Source:* http://en.wikipedia.org/wiki/Pulse-width_modulation?oldid=628618587 *Contributors:* Damian Yerrick, PierreAbbat, SimonP, Heron, Lumpbucket, Michael Hardy, Tim Starling, Ixfd64, Glenn, GRAHAMUK, Selket, EpiVictor, Pingveno, Giftlite, DavidCary, BenFrantzDale, Everyking, Starx, Mschlindwein, Sonett72, TedPavlic, Jaberwocky6669, Adambro, Bobo192, Meestaplu, Hooperbloob, RJFJR, DV8 2XL, Kenyon, SCEhardt, RuM, Ademkader, Alll, Dermeister, Fish and karate, Ian Pitchford, Gurch, Krishnavedala, YurikBot, Toffile, Rohitbd, Deville, KnightRider, SmackBot, Sam8, Chris the speller, Nbarth, SundarBot, Nahum Reduta, Funky Monkey, S Roper, Johncatsoulis, Gobonobo, Ckatz, CyrilB, Dicklyon, Spook`, Politepunk, Wizard191, Twas Now, CapitalR, Nit vs atwiki, Porterjoh, Jaeger5432, Chrike, Circuit dreamer, Zureks, WeggeBot, Cydebot, MC10, Rsutherland, Nick Wilson, Simon Brady, Thijs!bot, Gerry Ashton, Vibhutesh, Xoneca, Magioladitis, Rivertorch, Nikevich, ML1986, Matt B., Glrx, JKN abb, R'n'B, Candleknight, GandalfDaGraay, Jeepday, Leopold Stotch, Broadbot, Andy Dingley, Spinningspark, AlleborgoBot, Deconstructhis, Yngvarr, SieBot, Allmightyduck, EnOreg, Denisarona, Asher196, ClueBot, DragonBot, Arjayay, SchreiberBike, Lambtron, DumZiBoT, Joel Saks, XLinkBot, Teslaton, MystBot, Addbot, Mortense, Tide rolls, OlEnglish, Legobot, Luckas-bot, Yobot, Mikey likes mountains, AnomieBOT, Zangar, RandomAct, Xzapro4, GB fan, ArthurBot, Xqbot, DSisyphBot, Shulini, Shfork, Omnipaedista, RibotBOT, DoostdarWKP, I dream of horses, Phoenix7777, TobeBot, DARTH SIDIOUS 2, DexDor, EmausBot, DMChatterton, Wikipelli, ZéroBot, Fæ, Squall line, Tolly4bolly, Mayflowerone, Yves86, Puffin, Sven Manguard, ClueBot NG, Matthiaspaul, Satellizer, Kubing, PoqVaUSA, Snotbot, Willardmcg, Widr, BG19bot, Ninney, Tungstic, BattyBot, Hebert Peró, Mrt3366, GoShow, عبدالرحمن مسلم, Ajv39, MadCowpoke, AK456, Liangjingjin, SFK2, Sihuapilapa, Buntybhai, Hello371882, Satwikmishravit, Megapod, Monkbot and Anonymous: 216

7.7.2 Images

- **File:Commons-logo.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* ? *Contributors:* ? *Original artist:* ?
- **File:Delta_PWM.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/dc/Delta_PWM.svg *License:* CC-BY-SA-3.0 *Contributors:*
- Delta_PWM.png *Original artist:* Delta_PWM.png: Cyril BUTTAY
- **File:Duty_cycle_general.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/6/68/Duty_cycle_general.svg *License:* CC-BY-SA-3.0 *Contributors:*
- Duty_cycle_general.png *Original artist:* Duty_cycle_general.png: Cyril BUTTAY
- **File:INTEL8031AH.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/a/af/INTEL8031AH.png *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Rhondle
- **File:Intel_8051_arch.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/cd/Intel_8051_arch.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Appaloosa

- **File:Intel_8085A_Die_CPU_Image.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/6/6c/Intel_8085A_Die_CPU_Image.jpg *License:* CC-BY-SA-3.0 *Contributors:* I took the picture with my own camera
  Previously published: none *Original artist:* Pdesousa359
- **File:Intel_8085_arch.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/bb/Intel_8085_arch.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Appaloosa
- **File:Intel_8086_CPU_Die.JPG** *Source:* http://upload.wikimedia.org/wikipedia/commons/a/a8/Intel_8086_CPU_Die.JPG *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Pdesousa359
- **File:Intel_8086_block_scheme.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/f7/Intel_8086_block_scheme.svg *License:* CC-BY-SA-3.0 *Contributors:* Własne opracowanie na podstawie różnych źródeł *Original artist:* Harkonnen2
- **File:KL_Intel_P8051.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/f0/KL_Intel_P8051.jpg *License:* CC-BY-SA-3.0 *Contributors:* CPU collection Konstantin Lanzet *Original artist:* Konstantin Lanzet (with permission)
- **File:KL_Motorola_68HC11.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/0/02/KL_Motorola_68HC11.jpg *License:* CC-BY-3.0 *Contributors:* CPU collection *Original artist:* Konstantin Lanzet
- **File:KL_USSR_KP1810BM86.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/d2/KL_USSR_KP1810BM86.jpg *License:* GFDL *Contributors:* CPU collection Konstantin Lanzet.
  Picture taken with Canon EOS 400D. *Original artist:* Konstantin Lanzet
- **File:M6800_Family_Block_Diagram.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/57/M6800_Family_Block_Diagram.png *License:* Public domain *Contributors:* self *Original artist:* Swtpc6800 en:User:Swtpc6800 Michael Holley
- **File:MC6800_Processor_Diagram.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/1b/MC6800_Processor_Diagram.png *License:* Public domain *Contributors:* self *Original artist:* Swtpc6800 en:User:Swtpc6800 Michael Holley
- **File:MC68HC11_microcontroller.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/b5/MC68HC11_microcontroller.jpg *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:MOS_6501_Ad_August_1975.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/1d/MOS_6501_Ad_August_1975.jpg *License:* Public domain *Contributors:* Scanned from August 7, 1975 issue of *Electronics* by Michael Holley Swtpc6800 *Original artist:* MOS Technology. Advertising agency: Henry S. Goodsett Advertising
- **File:Microchip_PIC24HJ32GP202.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/26/Microchip_PIC24HJ32GP202.jpg *License:* CC-BY-SA-3.0-2.5-2.0-1.0 *Contributors:* Self-taken *Original artist:* w:User:Acdx
- **File:Motorola_M6800_manuals.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/de/Motorola_M6800_manuals.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Swtpc6800 en:User:Swtpc6800 Michael Holley
- **File:Motorola_M6800_microcomputer_ad_April_1975.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/8/82/Motorola_M6800_microcomputer_ad_April_1975.jpg *License:* Public domain *Contributors:* Scanned from pages 42 and 43 of the April 17, 1975 issue of *Electronics* magazine by Michael Holley Swtpc6800 *Original artist:* Motorola Semiconductor Products Inc. Advertising agency, E.B. Lane and Associates. Both located in Phoenix Arizona.
- **File:Motorola_MC6800_microprocessor.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/5a/Motorola_MC6800_microprocessor.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Swtpc6800 en:User:Swtpc6800 Michael Holley
- **File:Motorola_MC6820L_MC6821L.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/33/Motorola_MC6820L_MC6821L.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Swtpc6800 en:User:Swtpc6800 Michael Holley
- **File:Motorola_Transistor_Radio_1960.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/cf/Motorola_Transistor_Radio_1960.jpg *License:* Public domain *Contributors:* Scanned from the May 23, 1960 issue of *Life* magazine by Michael Holley Swtpc6800 . *Original artist:* Motorola
- **File:Oki_80c86a.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/9/97/Oki_80c86a.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Alecv
- **File:PIC12C508-HD.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/d0/PIC12C508-HD.jpg *License:* CC-BY-3.0 *Contributors:* http://zeptobars.ru/en/read/open-microchip-asic-what-inside-II-msp430-pic-z80 *Original artist:* ZeptoBars
- **File:PIC16C505-HD.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/fc/PIC16C505-HD.jpg *License:* CC-BY-3.0 *Contributors:* http://zeptobars.ru/en/read/open-microchip-asic-what-inside-II-msp430-pic-z80 *Original artist:* ZeptoBars
- **File:PIC16CxxxWIN.JPG** *Source:* http://upload.wikimedia.org/wikipedia/commons/f/f8/PIC16CxxxWIN.JPG *License:* CC-BY-2.5 *Contributors:* ? *Original artist:* ?
- **File:PIC_microcontrollers.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/4/4c/PIC_microcontrollers.jpg *License:* Public domain *Contributors:* Photo taken by uploader *Original artist:* MikeMurphy
- **File:PWM,_3-level.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/8/8e/PWM%2C_3-level.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Zureks
- **File:Pwm.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/7/72/Pwm.svg *License:* CC-BY-SA-3.0 *Contributors:*
- Pwm.png *Original artist:* Pwm.png: CyrilB
- **File:Question_book-new.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* ? *Contributors:*
  Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:*
  Tkgd2007
- **File:SAB-C515-LN.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/9/9f/SAB-C515-LN.jpg *License:* CC-BY-2.5 *Contributors:* Own work *Original artist:* MichaelFrey
- **File:SWTPC6800_open.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/be/SWTPC6800_open.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Swtpc6800 en:User:Swtpc6800 Michael Holley

- **File:Sigma-delta_PWM.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/53/Sigma-delta_PWM.svg *License:* CC-BY-SA-3.0 *Contributors:*
- Sigma_delta.png *Original artist:* Sigma_delta.png: Cyril BUTTAY
- **File:Silicon_wafer.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/e2/Silicon_wafer.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Inductiveload
- **File:Tektronix_4051_ad_April_1976.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/c2/Tektronix_4051_ad_April_1976.jpg *License:* Public domain *Contributors:* Scanned from page 189 of the April 15, 1976 issue of *Electronics* magazine by Michael Holley Swtpc6800 *Original artist:* Tektronix
- **File:Text_document_with_red_question_mark.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/a/a4/Text_document_with_red_question_mark.svg *License:* Public domain *Contributors:* Created by bdesham with Inkscape; based upon Text-x-generic.svg from the Tango project. *Original artist:* Benjamin D. Esham (bdesham)
- **File:Three_PWM_types.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/0/06/Three_PWM_types.svg *License:* CC-BY-SA-3.0 *Contributors:*
- Three_types.png *Original artist:* Three_types.png: Cyril BUTTAY
- **File:UPD8086D-2_NEC_1984year_19week_JAPAN.JPG** *Source:* http://upload.wikimedia.org/wikipedia/commons/a/a0/UPD8086D-2_NEC_1984year_19week_JAPAN.JPG *License:* CC-BY-SA-4.0 *Contributors:* Own work *Original artist:* Andrzej w k 2
- **File:Wiki_letter_w_cropped.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/1c/Wiki_letter_w_cropped.svg *License:* CC-BY-SA-3.0 *Contributors:*
- Wiki_letter_w.svg *Original artist:* Wiki_letter_w.svg: Jarkko Piiroinen
- **File:Wikibooks-logo-en-noslogan.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/df/Wikibooks-logo-en-noslogan.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.
- **File:Wyprowadzenie_mikroprocesora_8086.JPG** *Source:* http://upload.wikimedia.org/wikipedia/commons/8/81/Wyprowadzenie_mikroprocesora_8086.JPG *License:* Public domain *Contributors:* Dokumentacja mikroprocesora Intel 8086 *Original artist:* Unknown

7.7.3 Content license

- Creative Commons Attribution-Share Alike 3.0