Control units

Digital technology furnishes an extensive array of options for open and closed-loop control of automotive electronic systems. A large number of parameters can be included in the process to support optimal operation of various systems. The control unit receives the electrical signals from the sensors, evaluates them, and then calculates the triggering signals for the actuators. The control program, the "software", is stored in a special memory and implemented by a microcontroller. The control unit and its components are referred to as hardware. The Motronic control unit contains all of the algorithms for open and closed-loop control needed to govern the engine-management processes (ignition, induction and mixture formation, etc.).

Operating conditions

The control unit is subjected to very high demands with respect to

- Extreme ambient temperatures (in normal vehicle operation from –40 to +60...+125 °C)
- Extreme temperature changes
- Indirect materials and supplies (oil, fuel etc.)
- The effects of moisture and
- Mechanical stress such as vibration from the engine

The control unit must operate reliably when the vehicle is started with a weak battery (e.g. cold start) and with high charge voltages (vehicle electrical system fluctuations).

Other requirements arise from the need for EMC (ElectroMagnetic Compatibility). The requirements regarding immunity to electromagnetic interference and limitation of high-frequency interference signal emission are extremely stringent.

Design

The printed circuit board with the electrical components (Fig. 1) is installed in a housing of plastic or metal. A multiple plug connects the control unit to the sensors, actuators and electrical power supply. The high-performance driver circuits that provide direct control of the actuators are specially integrated within the housing to ensure effective heat transfer to the housing and the surrounding air.

The majority of the electrical components are of the surface-mounted device technology type. This concept provides extremely efficient use of space in low-weight packages. Only a few power components and the connectors use push-through assembly technology.

Hybrid versions combining compact dimensions with extreme resistance to thermal attack are available for mounting directly on the engine.

Data processing

Input signals

In their role as peripheral components, the actuators and the sensors represent the interface between the vehicle and the control unit in its role as the processing unit. The electrical signals of the sensors are routed to the control unit via a wiring harness and the connector plug. These signals can be of the following type:

Analog input signals

Within a given range, analog input signals can assume practically any voltage value. Examples of physical quantities which are available as analog measured values are intake-air mass, battery voltage, intake-manifold and boost pressure, coolant and intake-air temperature. They are converted into digital values by an analog-digital converter in the microcontroller of the control unit and used for calculations by the microcontroller CPU. The maximum resolution of these analog signals is 5 mV. This translates into roughly 1,000 incremental graduations based on an overall measuring range of 0 to 5 V.

Digital input signals

Digital input signals only have two states. They are either “high” or “low” (logical 1 and logical 0 respectively). Examples of digital input signals are on/off switching signals, or digital sensor signals such as the rotational-speed pulses from a Hall generator or a magnetoresistive sensor. Such signals are processed directly by the microcontroller.

Pulse-type input signals

The pulse-shaped input signals from inductive-type sensors containing information on rotational speed and reference mark are conditioned in their own control unit stage. Here, spurious pulses are suppressed and the pulse-shaped signals converted into digital rectangular signals.

Signal conditioning

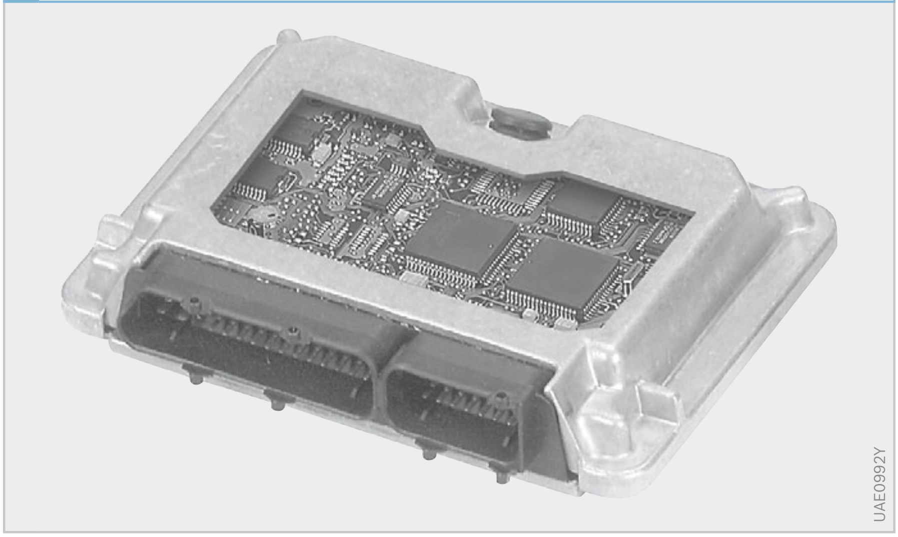
Protective circuits limit the voltages of input signals to levels suitable for processing. Filters separate the useful signal from most interference signals. When necessary, the signals are then amplified to the input voltage required by the microcontroller (0 to 5 V).

Signal conditioning can take place completely or partially in the sensor depending upon the sensor’s level of integration.

Signal processing

The control unit is the switching center governing all of the functions and sequences regulated by the engine-management system. The closed and open-loop control functions are executed in the microcontroller. The input signals that are provided by the sensors and the interfaces to other systems (such as the CAN bus) are used as input variables. and are subjected to a further plausibility check in the computer. The control unit program supports generation of the output signals used to control the actuators.

1 Design of a control unit using the example of an ME Motronic (sectional view through housing cover)



Output signals

The microcontroller uses the output signals to control output stages that usually provide enough power for connecting the actuators directly. It is also possible to actuate certain output stage relays for consumers that use up a great deal of power (e.g. motor fans).

The output stages are proof against short circuits to ground or battery voltage, as well as against destruction due to electrical or thermal overload. Such malfunctions, together with open-circuit lines or sensor faults are identified by the output-stage IC as an error and reported to the microcontroller.

Switching signals

Actuators can be switched on and off using the switching signals (e.g. motor fans).

PWM signals

Digital output signals can be in the form of PWM (Pulse-Width Modulated) signals. These are constant-frequency rectangular signals with variable on-times (Fig. 2), Various actuators can be moved to various operating positions using these signals (e.g. exhaust-gas recirculation valve, boost-pressure actuator).

2 PWM signals

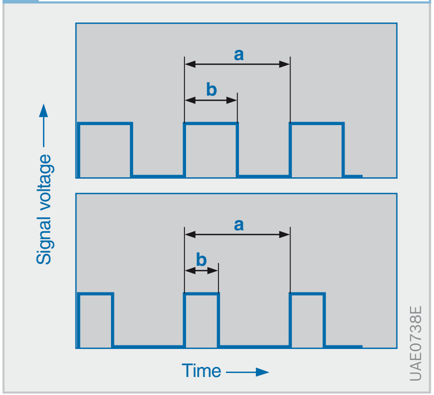


Fig. 2
a Period duration (fixed or variable)
b Variable on-time

Control unit-internal communication

In order to be able to support the microcontroller in its work, the peripheral components must communicate with it. This takes place using an address/data bus which, The microcontroller outputs the RAM address whose contents are to be read (for example) via the address bus. The data bus is then used to transmit the relevant data. For former automotive applications, an 8-bit bus topology sufficed. This meant that the data bus comprised 8 lines which together could transmit 256 values simultaneously. 65,536 addresses can be accessed using the 16-bit address bus in this system. Presently, more complex systems demand 16 bits, or even 32 bits, for the data bus. In order to save on pins at the components, the data and address buses can be combined in a multiplex system, i.e. addresses and data are dispatched through the same lines but offset from each other with respect to time.

Serial interfaces with just a single data line are used for data that does not have to transmitted extremely quickly (e.g. fault memory data).

EOL programming

The extensive variety of vehicle variants with differing control programs and data records, makes it imperative to have a system which reduces the number of control unit types needed by a given manufacturer. To this end, the Flash-EPROM's complete memory area can be programmed at the end of production with the program and the variant-specific data record (this is the so-called End-of-Line, or EoL, programming).

Another way of reducing the type diversity is to store several data variants (e.g. transmission variants) in the memory, which are then selected using coding at the end of the production line. This coding is stored in an EEPROM.

Performance of electronic control units

The performance of electronic control units goes hand-in-hand with advances achieved in the field of microelectronics. The first gasoline injection systems were still analog – with limited flexibility in the implementation of control functions. These functions were constrained by the hardware.

Progress advanced in quantum leaps with the arrival of digital technology and the microcontroller. The entire engine management system was taken over by the universally applicable semiconductor microchip. The actual control logic in microcontroller-controlled systems is in a programmable semiconductor memory.

From systems that initially simply controlled fuel injection, complex engine-management systems were then developed. They controlled not only fuel injection but also the ignition system including knock control, exhaust-gas recirculation and a whole variety of other systems. This continuous process of development is bound to continue in a similar vein over the next decade as well. The integration of functions and, above all, their complexity are constantly increasing. This pattern of development is only possible because the microcontrollers used are also undergoing a similar process of improvement.

Microcontrollers in the Intel 8051 family were used quite some time until they were replaced with 80515 derivatives with additional I/O facilities for timer-controlled signals and an integrated analog-digital converter at the end of the 1980's. It was then possible to create relatively powerful systems. Figure 3 shows a comparison between the performance of a fuel-injection system (LH3.2) and an ignition system (EZ129K) – equipped with 80C515 controllers – and that of the succeeding Motronic systems. The ME7 has approximately 40 times the performance capability of the LH/EZ combination with a clock frequency of 40 MHz. With the benefit of a new generation of microcontrollers and a further increase in clock frequency on the ME9, this figure will increase to a factor of well over 50.

In the foreseeable future microcontrollers will process more than just digital control sequences. Signal processors are integrated that can also directly process the signals provided by knock sensors, for example.

Advances in the development of semiconductor memory chips are also worthy of note. Complex control programs require an enormous amount of memory space. The capacity of memory chips at the start of the 1980s was still only 8 kilobytes. The ME7 now uses 1-megabyte chips and soon memory capacities of 2 megabytes will be required. Figure 3 shows this pattern of development and likely future trends.

3 Development of electronic control units

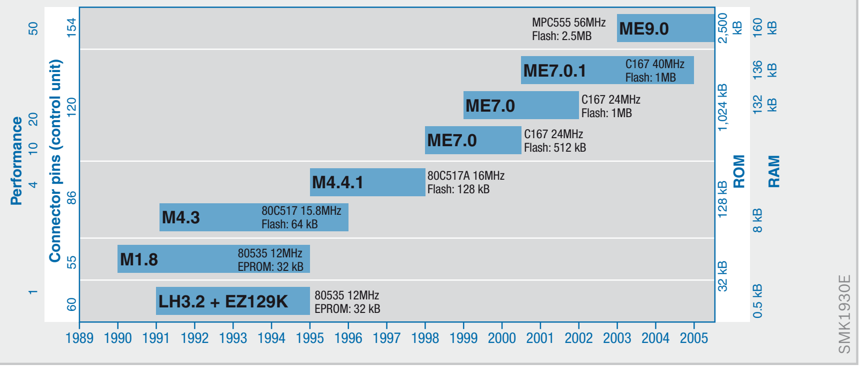


Fig. 3
Chart illustrating
- Performance capability of engine-management systems
- Number of control unit connector pins
- Program memory capacity
- Data memory capacity (RAM)

By way of comparison: The performance capability of a state-of-the-art engine-management system far exceeds that of Apollo 13.

Digital modules in the control unit

Microcontroller

Structure

A microcontroller consists of the following interacting components (Fig. 1):

- *Central processing unit* (CPU): this contains the control unit and the arithmetic and logic unit. The control unit executes the instructions from the program memory, whereas the arithmetic and logic unit performs arithmetical and logical operations.
- *Input and output devices* (I/O, Input/Output), which handle the exchange of data with peripheral devices. Peripheral devices include input and output devices and external data storage media.
- *Program memory*, in which the operating program (user program) is permanently stored (ROM, PROM, EPROM or flash EPROM).
- *Data memory*, which is accessed for reading and writing (RAM). This contains the data that is currently being processed. Non-volatile memory (EEPROM) is used for data that must not be deleted when the supply voltage is switched off.
- The *bus system* connects the individual elements of the microcontroller.
- A *clock generator* (oscillator) ensures that all operations in the microcontroller take place within a defined timing pattern.
- *Logic circuits* are modules with specialized tasks such as program interrupts. They are integrated in individual I/O units.

The chief components of a microcomputer are generally separate modules connected to one another on a printed-circuit board. The microprocessor within such as system – the CPU – is not functional on its own: it is always part of a microcomputer.

In a microcomputer, however, the above-mentioned functions are integrated on a silicon wafer (system-on-a-chip). This is not functional on its own (standalone) and is therefore referred to as a single chip microcomputer.

The microcontroller is used to control self-regulating systems such as an engine-management system. Depending on the application, they may also have expansion modules connected to them (e.g. additional memory for data and program code).

The user program is fixed and is not replaced for different applications. This is the difference between a microcontroller system, for example, and a PC.

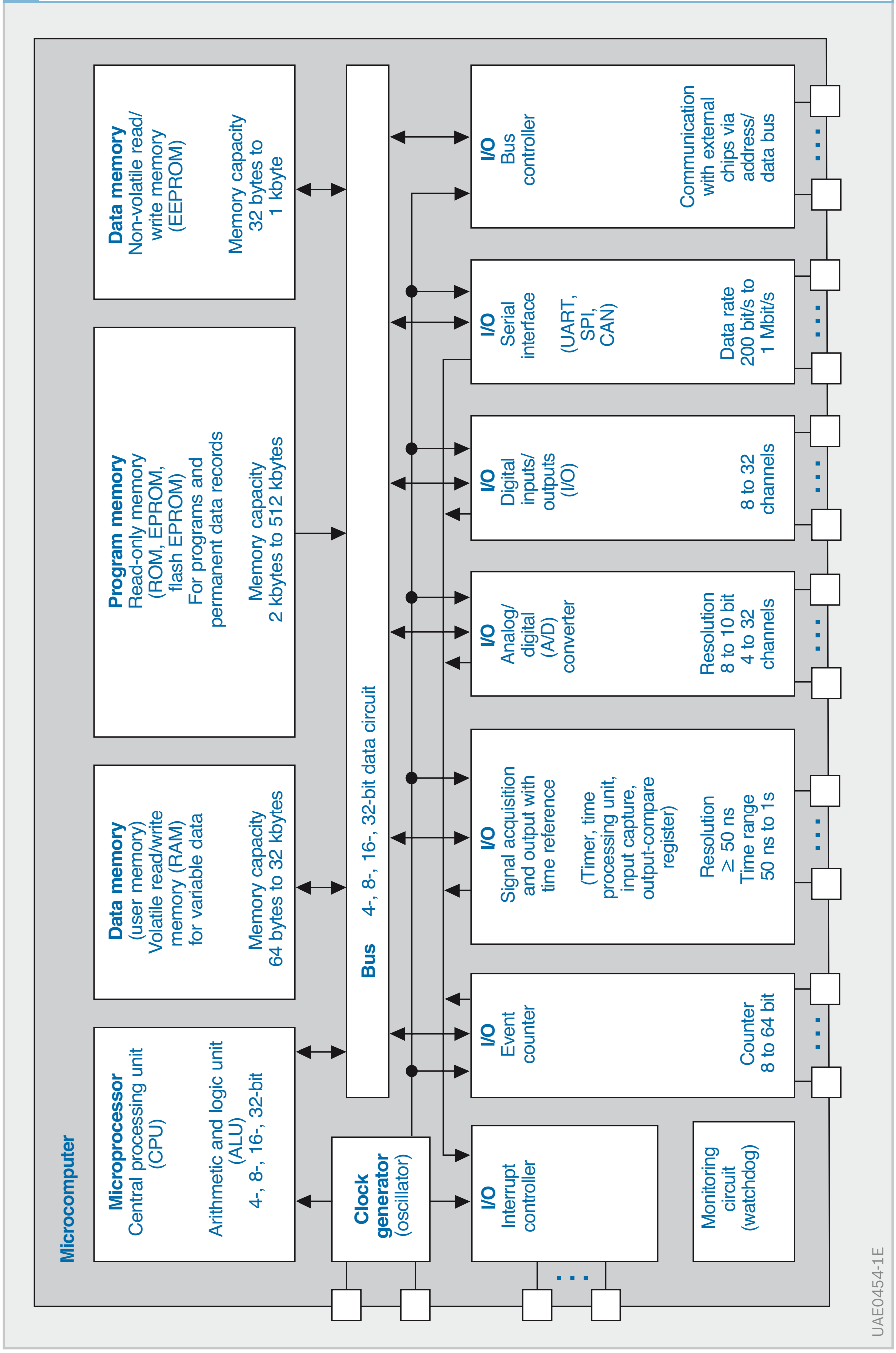
Programming

The only command form capable of direct interpretation by a microprocessor is a bit pattern, i.e. the binary representation of a number. Since, however, this form of instruction is not easy to work with for a programmer, and is therefore susceptible to errors, easily memorable abbreviations (mnemonics) are used. These are automatically translated by an assembler program into bit patterns (machine code) that can be understood by the microprocessor.

For more complex systems and programs, high-level programming languages such as C are needed, as otherwise it would be impossible to keep extensive programs manageable and free of errors. Such languages require sophisticated translation programs (compilers) which convert the text of the high-level language into a form that can be processed by the microcontroller.

The machine code is stored in the program memory, where it remains permanently. The CPU accesses these components via the bus system, reads the numerically coded commands and executes then.

1 Microcontroller


Microcomputer
Microprocessor
Central processing unit (CPU)
Arithmetic and logic unit (ALU)
4-, 8-, 16-, 32-bit
Data memory
(user memory)
Volatile read/write memory (RAM)
for variable data
Memory capacity 64 bytes to 32 kbytes
Program memory
Read-only memory (ROM, EPROM, flash EPROM)
For programs and permanent data records
Memory capacity 2 kbytes to 512 kbytes
Data memory
Non-volatile read/write memory (EEPROM)
Memory capacity 32 bytes to 1 kbyte
Bus 4-, 8-, 16-, 32-bit data circuit
Clock generator (oscillator)
I/O Interrupt controller
Monitoring circuit (watchdog)
I/O Event counter
Counter 8 to 64 bit
I/O Signal acquisition and output with time reference
(Timer, time processing unit, input capture, output-compare register)
Resolution ≥ 50 ns
Time range 50 ns to 1s
I/O Analog/digital (A/D) converter
Resolution 8 to 10 bit
4 to 32 channels
I/O Digital inputs/outputs (I/O)
8 to 32 channels
I/O Serial interface
(UART, SPI, CAN)
Data rate 200 bit/s to 1 Mbit/s
I/O Bus controller
Communication with external chips via address/data bus
UAE0454-1E

Semiconductor memories

Applications

Memories are used to store large volumes of

- Digital signals representing data (I/O data, statuses, intermediate results involving frequent and rapid reading and writing)
- Program code (usually permanently stored) and
- Constants (permanently stored)

Storage involves

- Recording (writing)
- Permanent retention (actual storage) and
- Location and retrieval (reading) of information

Memories utilize physical effects that clearly and easily create and show two different states (e.g. conducting/non-conducting or loaded/not loaded). The information that is to be stored must therefore exist in binary form, i.e. encoded as a series of "yes or no" statuses (logical "1" or logical "0"). Such a "yes or no" unit of information is called a bit (binary digit).

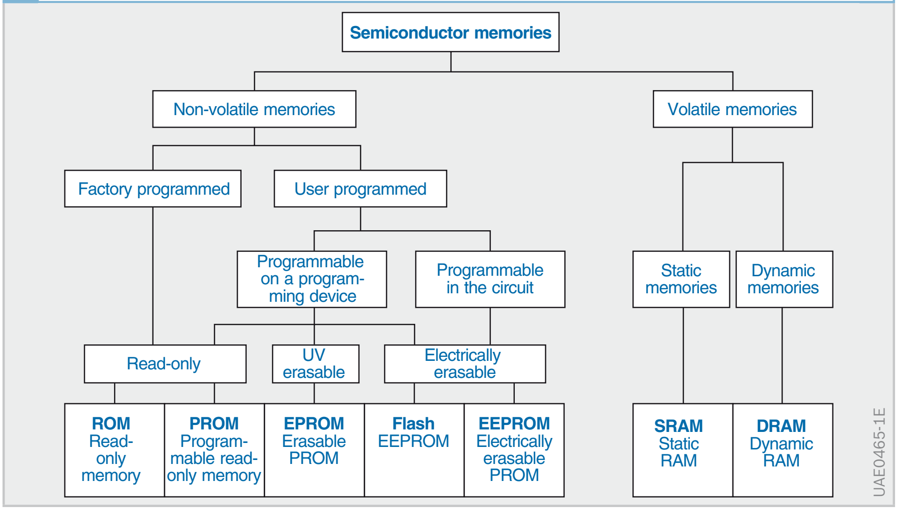
Memory modules are organized on a bit or word basis, depending on the application. A "word" is a group of bits that can be processed as a single unit. The word length is equal to the number of bits processed as a single unit. Eight bits are referred to as a byte.

Memories can be organized on the basis of different word lengths. An 8 M x 8-RAM, for example, has a memory capacity of 8 million times 8 bits (64 Mbits). The data is organized into bytes (8 bits), making the memory capacity 8 Mbytes.

Word lengths of 4, 8, 16 and 32 bits are common in microcontroller systems.
The word length is one of the factors that determines the performance capability of the system. The word length that is used depends on the performance capability requirements of the system.

The most important terms are explained below according to their standardized definitions, where applicable, or their most common usage (see Figure 2 for overview).

2 Overview of semiconductor memories



Random-access memory (RAM)

Random-access memory or RAM is a short-term memory that allows direct access to any storage location. Information can be read/written from/to the memory any number of times.

Static RAM (SRAM)

Static RAMs use bistable switching elements as the data storage cells. Their functionality is similar to that of a flip-flop, a simple circuit with two transistors, of which either the one (logical "1") or the other (logical "0") conducts at any one time. In SRAM, the information remains stored until the storage cell concerned is addressed and overwritten, or the operating voltage is switched off.
SRAM is therefore volatile memory.

Dynamic RAM (DRAM)

Unlike SRAM, the information is stored as an electrical charge in the gate capacity of a CMOS transistor in dynamic RAM (DRAM). As such capacitors are susceptible to leakage, the charge is gradually lost. In order to retain the information, the charge has to be refreshed at regular intervals (every few ms).

Read-only memory

Read-only memory (ROM) is permanent-storage memory that allows any memory location to be accessed directly but – as the name indicates – allows the information only to be read and not modified.

A ROM is nonvolatile memory, i.e. the information it contains is retained even when the operating voltage is switched off. It is usually used to store program code (control programs) and fixed data (function tables, encoding rules, engine characteristic data maps) that need to be retrievable at any time. The information may be indelibly entered in the memory by the manufacturer or the user by means of appropriate programming of specially prepared memories (PROMs or programmable ROMs).

Erasable ROM

There are also ROMs whose contents can be erased and reprogrammed as outlined below.

EPROM (Erasable PROM)

This type of erasable read-only memory can have its contents completely wiped by irradiation with UV light and can then be reprogrammed using a programming device.

EEPROM (Electrical EPROM)

The EEPROM (also known as E^2PROM) can be electrically erased and reprogrammed. Every storage cell of an EEPROM be individually overwritten. For that reason, this type of memory module can also be used as nonvolatile data memory (e.g. for learned information in engine management systems).

Flash EEPROM

A more sophisticated variant of the EPROM and EEPROM is flash EEPROM. In this case, electrical flash pulses are used to erase specific storage areas or the entire contents of the memory. The erased areas can subsequently be reprogrammed.

The flash memory can be reprogrammed on a programming station. However, the advantage of flash EEPROM is that is can also be reprogrammed while still inside the sealed control unit.

Flash EEPROM is used in cases where relatively large quantities of data need to be stored, but must also be modifiable (e.g. program memory in vehicle control units).

Control unit software

Real-time capability

One of the requirements on electronic systems is real-time capability. This means that control procedures must react to input signals within an extremely short time. For example, a wheel that has a tendency to lock must be detected so quickly that the ABS control algorithm in the control unit can reduce the brake pressure via the hydraulic modulator quickly enough. Brake slip is therefore reduced before the wheel can lock. Engine management systems make considerable real-time capability demands so that crankshaft angles can be adhered to with extreme accuracy at fast engine speeds for injection and ignition timing purposes.

The complexity of an electronic system therefore makes extremely high demands of the software that is developed. The software structure is explained below on the basis of an example.

Software structure

The microcontroller in the control unit executes commands sequentially. The command code is obtained from the program memory. The time taken to read in and execute the command depends on the microcontroller that is used and the clock frequency. The microcontrollers that are currently used in vehicles can execute up to 1 million commands per second.

Because of the limited speed at which the program can be executed, a software structure with which time-critical functions can be processed with high priority is required.

The engine-management program has to react extremely quickly to signals from the speed sensor, which records the engine speed and the crankshaft position. These signals arrive at short intervals that can be a matter of milliseconds, depending on the engine speed. The control unit program has to evaluate these signals with high priority. Other functions such as reading in the engine temperature are not as urgent, since the physical variable only changes extremely slowly in this case.

Interrupt control

As soon as an event occurs that requires an extremely rapid response (e.g. speed sensor pulse), the program that is currently running must be interrupted. This can be done using the microcontroller's interrupt control facility. Events can trigger a program execution interrupt, whereupon the program jumps and executes the “interrupt routine”. When this routine has been executed, the program resumes at the point at which it was interrupted (Fig. 1).

An interrupt can be triggered by an external signal, for example. Other interrupt sources are timers integrated in the microcontroller, with which timed output signals can be generated (e.g. ignition signal: microcontroller ignition output is switched at a point in time that is calculated beforehand). However, the timer can also generate internal time frames.

1 CPU power distribution principle

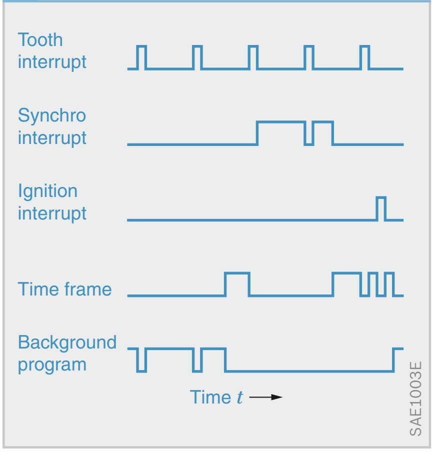


Fig. 1
Depiction of several program levels on the example of the software from a Motronic system

The control unit program reacts to several of these interrupts. An interrupt source can therefore request an interrupt while another interrupt routine is currently being executed. Every interrupt source therefore has a fixed priority assigned to it. The priority controller decides which interrupt is allowed to interrupt another interrupt.

Tooth interrupt

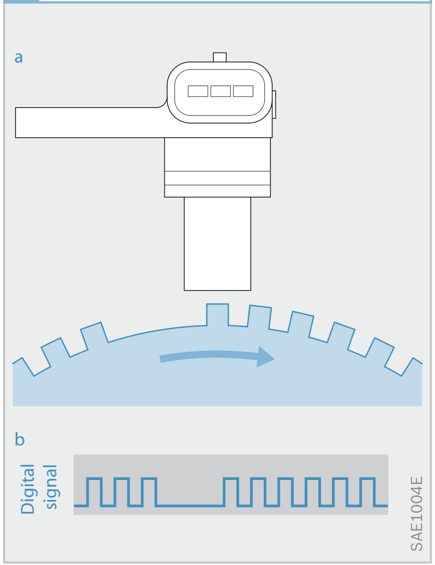
The crankshaft is equipped with a pulse wheel (Fig. 2a) that has a certain number of teeth on its circumference. The teeth are scanned by the speed sensor. This allows the crankshaft position to be recorded. The typical distance between a pair of teeth on the crankshaft sensor wheel is 6°. In order to determine the crankshaft position, the control unit program must execute certain routines as each tooth is detected. At 6,000 rpm the detection time between two teeth is approximately 300 µs. Every command in these routines must be executed within this time. This requires a rapid response to the speed sensor signal. For this purpose, the engine-speed signal is connected to a microcontroller interrupt input. Every falling signal edge at this input interrupts the current calculations that are in progress and forces a branch to the interrupt routine. After executing the commands in the interrupt routine, the program continues execution at its point of origin.

In order to perform certain operations the control unit program requires the time taken for the crankshaft to travel between one tooth and the next. This calculation is performed by an internal timer. This is a freewheeling 16-bit counter (Fig. 3) that increments at a certain rate, depending on the microcontroller oscillator clock cycle. This time frame amounts to about 0.5 µs. When the falling tooth flank occurs, the current counter status is recorded. The difference (and therefore the tooth interval) is calculated using the stored counter from the previous tooth.

Example: crankshaft position calculation

The engine-management system (Motronic for gasoline engines, EDC for diesel engines) must know the crankshaft position at any given point in time. This is a prerequisite for injecting into the right cylinder at the right time and ensuring that ignition takes place at the calculated ignition angle (Motronic systems). In order to

2 Crankshaft sensor ring with speed sensor



3 Time calculations via internal timer

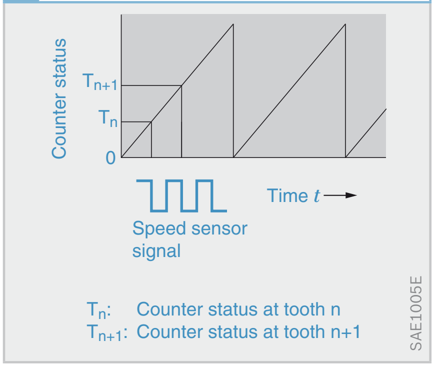


Fig. 2
a Design
b Speed sensor signal

detect the engine position and the engine speed, the control unit evaluates the speed sensor signal (Fig. 2b).

There is gap in the crankshaft sensor wheel in which two teeth are missing. The tooth space has a defined position in relation to the top dead center (TDC) of cylinder no. 1. The control unit program has to synchronize itself with this tooth space. This is done by measuring the times between two consecutive falling tooth flanks. The time for the tooth space is considerably greater than the time before and after the gap. Following a "short – long – short" sequence the last thing to be scanned was the falling flank of the second tooth after the space.

The crankshaft has rotated by 6° for each falling tooth flank that has been detected by the control unit program. This is how the control unit program knows the crankshaft position within this time frame.

Since cylinder no. 1 is in the tooth space position in the vicinity of top dead center (TDC) or bottom dead center (BDC), an additional signal is required to determine the position. The camshaft sensor provides a different voltage level in both cases. The control unit is therefore able to uniquely assign the crankshaft and camshaft positions.

Combustion-synchronous interrupt

Some calculations have to be performed for every combustion cycle. For example, the ignition angle and the injection have to be recalculated synchronously with combustion for each cylinder. The program does this by branching to the "synchronization program" after certain teeth (Fig. 4). This interrupt takes place after every 30 teeth (ignition interval) for a four-cylinder engine, and after every 20 teeth for a six-cylinder engine.

The synchronization program is fixed to a certain tooth position and has to be executed with high priority. For this reason it is activated via an interrupt (triggered by a command in the tooth interrupt routine). Since the synchronization program runs over several teeth at fast engine speeds, it has to be interrupted by the tooth interrupt. The tooth interrupt is given higher priority than the synchronization program.

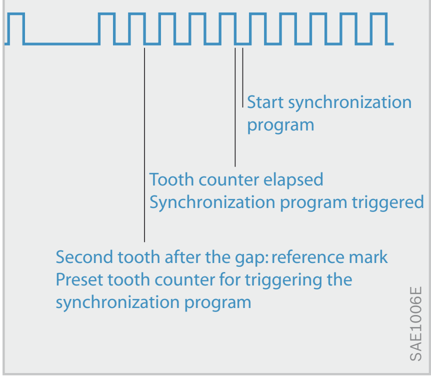
Ignition interrupt

The ignition output takes place within a certain crankshaft range, depending on the value from the ignition map. Since the specified ignition angle has to be adhered to exactly, the ignition output is controlled by an interrupt. Like the synchronization program, the ignition interrupt is also called up once per combustion cycle.

The control unit program is aware of the crankshaft position in the 6° framework. However, this framework is not accurate enough for ignition angle output. For this reason, accurate ignition output between two teeth must take place as well as this approximate counting for the last 0 to 6 crankshaft degrees. This is done using a timer (Fig. 5). Ignition angle output that was purely timer-controller would lead to an ignition angle output error at high engine speed dynamics.

Firstly, the ignition coil must be enabled for a defined time (the so-called dwell period). In order to do this, the program calculates the switch-on time by calculating

4 Triggering the interrupt synchronously with combustion

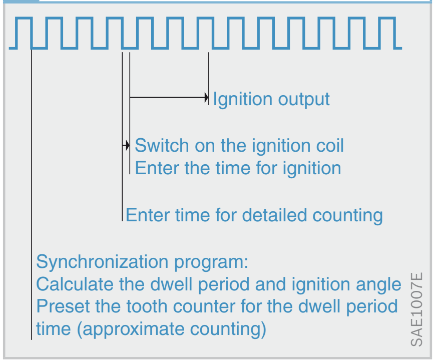

Start synchronization program
Tooth counter elapsed
Synchronization program triggered
Second tooth after the gap: reference mark
Preset tooth counter for triggering the synchronization program
SAE1006E

backwards from the ignition angle at which the ignition coil has to be switched off. This makes it possible to calculate the tooth after which the ignition coil has to be switched off (approximate counting in 6° time frame). The remaining angle (detailed counting 0 to 6°) is converted into an output time using the current engine speed. As soon as the specified tooth position has been reached using approximate counting, a time is loaded with the output value from the detailed counting. When this time period expires, the timer triggers an interrupt. The commands that switch on the ignition coil are programmed in this interrupt routine. Then the timer is preset to the dwell period value, which causes an interrupt to be triggered when the timer elapses, switching off the ignition coil and therefore initiating ignition.

Time frame

Many control algorithms have to run within a certain time frame. Lambda control, for example, has to be processed within a fixed time frame (e.g. 10 ms) so that the correcting variables are calculated quickly enough.

5 Dwell and ignition time output


Ignition output
Switch on the ignition coil
Enter the time for ignition
Enter time for detailed counting
Synchronization program:
Calculate the dwell period and ignition angle
Preset the tooth counter for the dwell period
time (approximate counting)
SAE1007E

Background program

All other activities that do not run in an interrupt routine or a time frame are processed in the background program. At fast engine speeds, the synchronization program and the tooth interrupt are called frequently, leaving little CPU time for the background program. The time taken for a complete run-through of the background program therefore increases rapidly with the increasing engine speed. The background program must therefore only contain low-priority functions.