Microprocessors and Microcomputers

All pictures used in the presentation come from “Microprocessors and Microcomputers” book by Prof. Ing. Jiří Pinker, CSc.

Ing. Petr Krist, Ph.D.

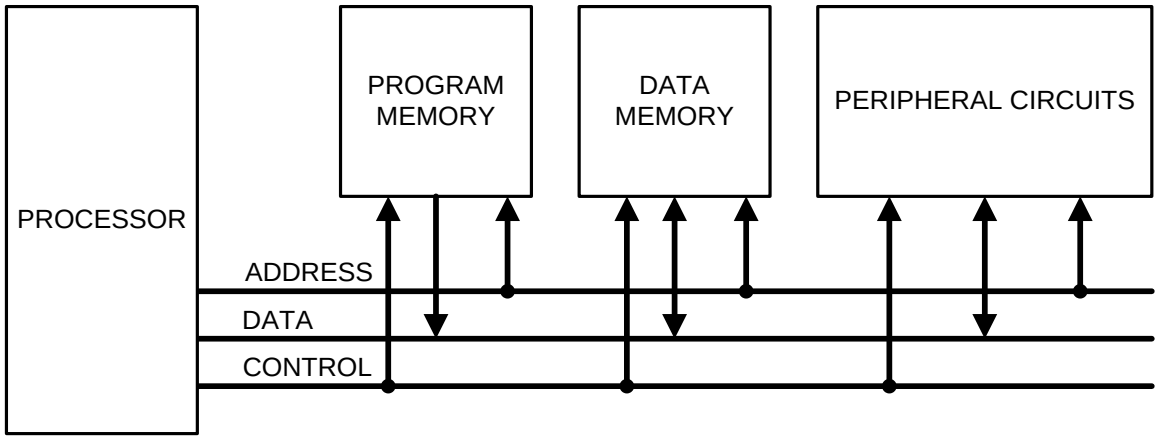
Department of Applied Electronics and Telecommunications

Faculty of Electrical Engineering, UWB in Pilsen

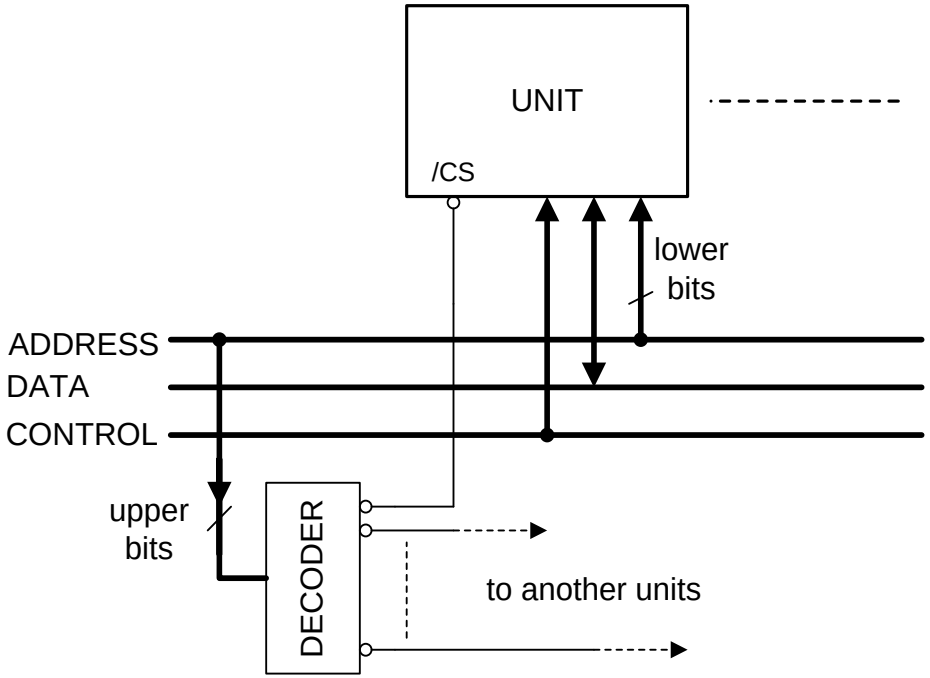
Contents

- Basic computer parts and functions
- Processor
- ALU – instruction overview
- Computer circuits
- Memory
- Interrupts
- Timers - counters
- Input and output circuits

Basic Computer Parts

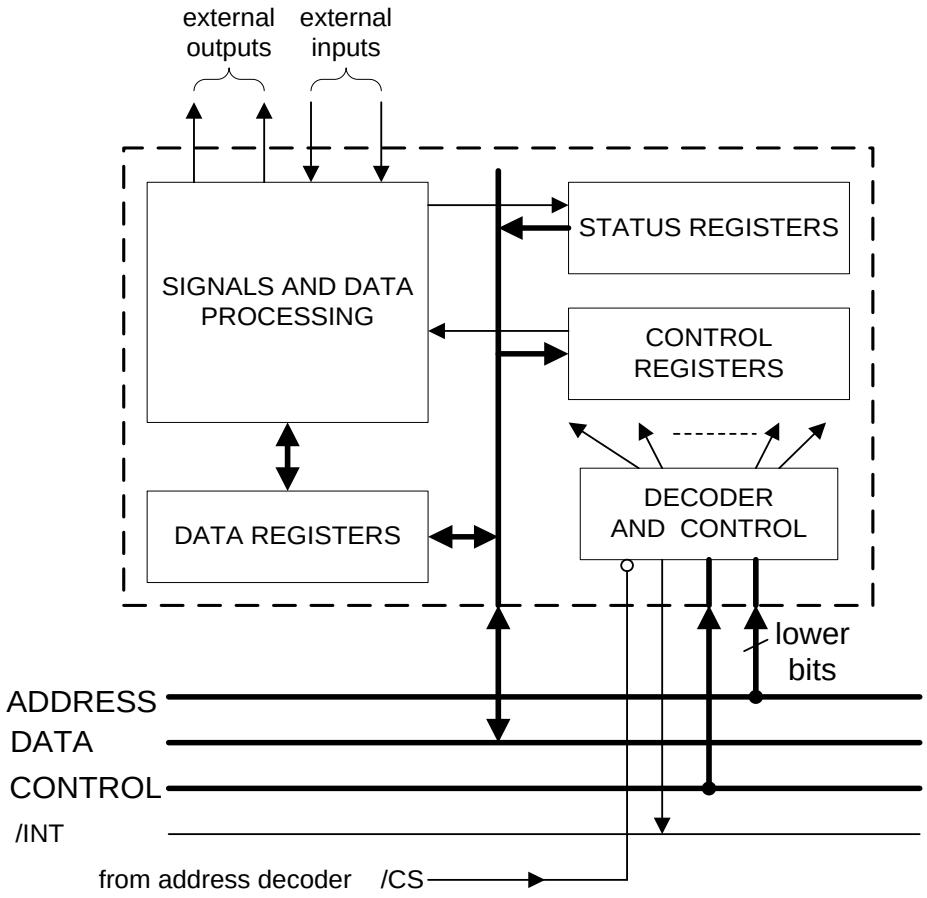


Attachment unit to the bus



Control signals: **/RD** and **/WR** – usually active in the low level „L“

Peripheral circuit structure

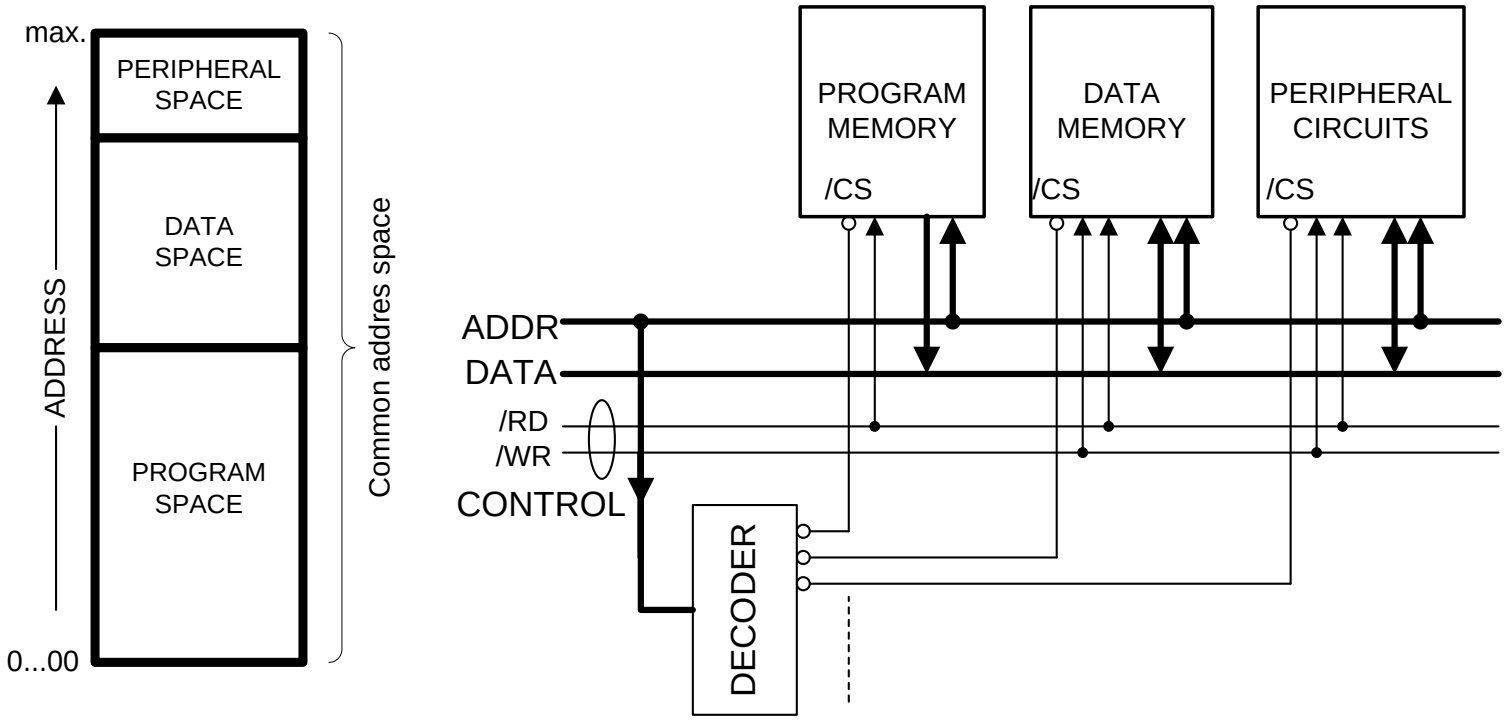


Address space

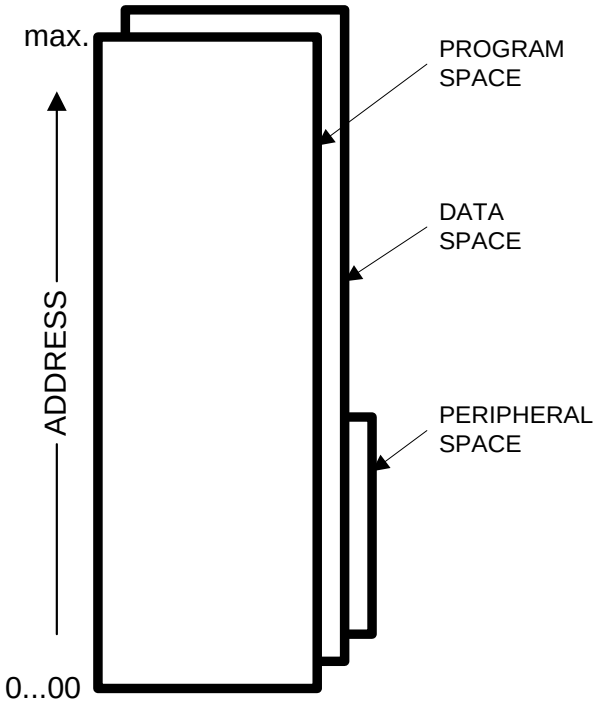
Set of addresses accessible by one type of the instruction cycle

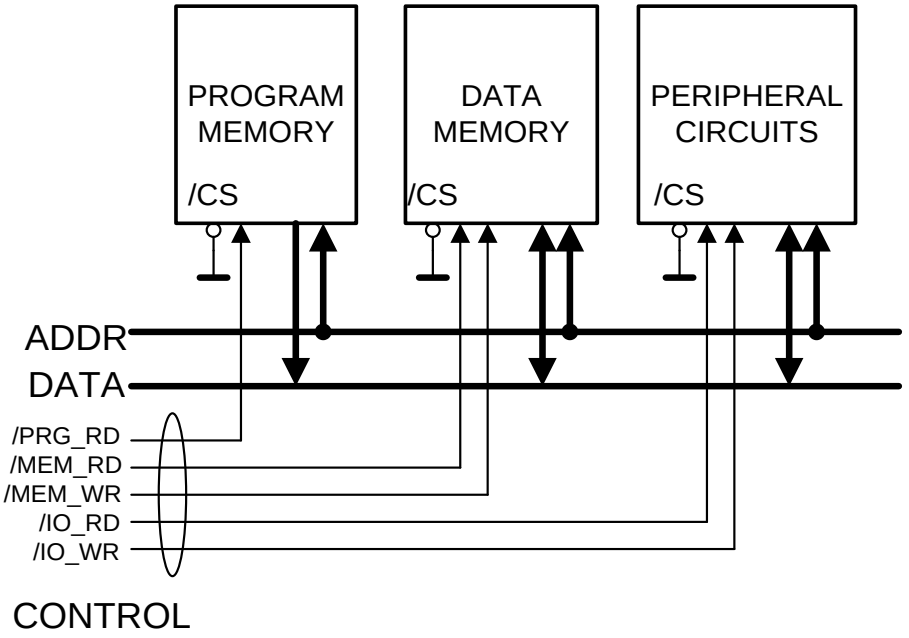
- **von Neumann architecture** – the only one address space
- **Harvard architecture** – several address spaces: program, data, peripheral I/O

von Neumann architecture

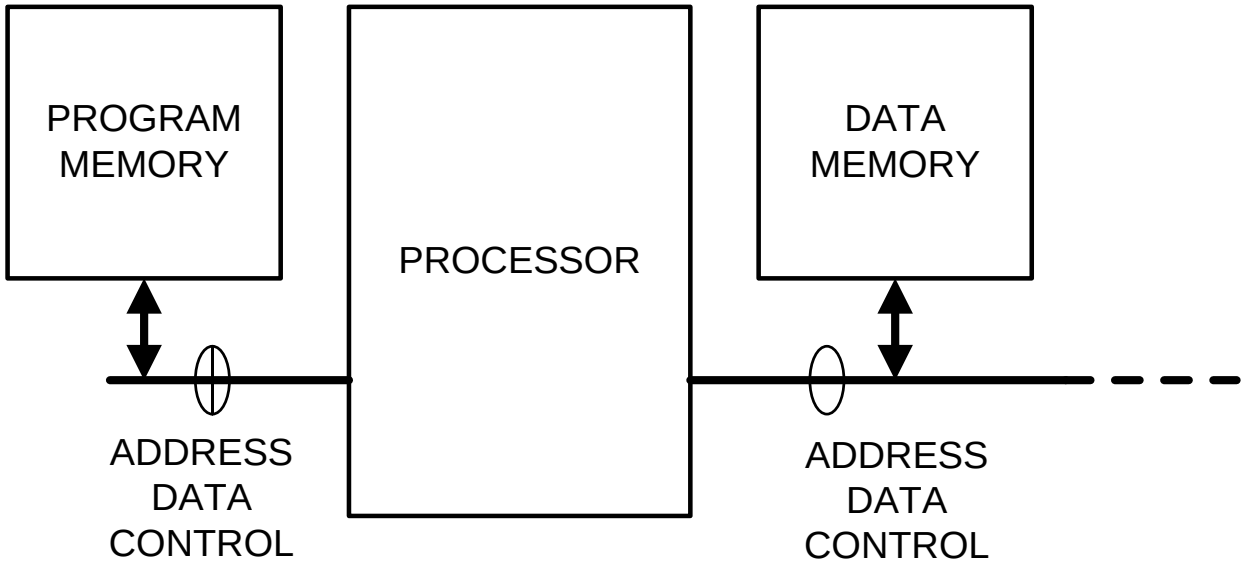


Harvard architecture





Two-bus system architecture



CISC and RISC processors

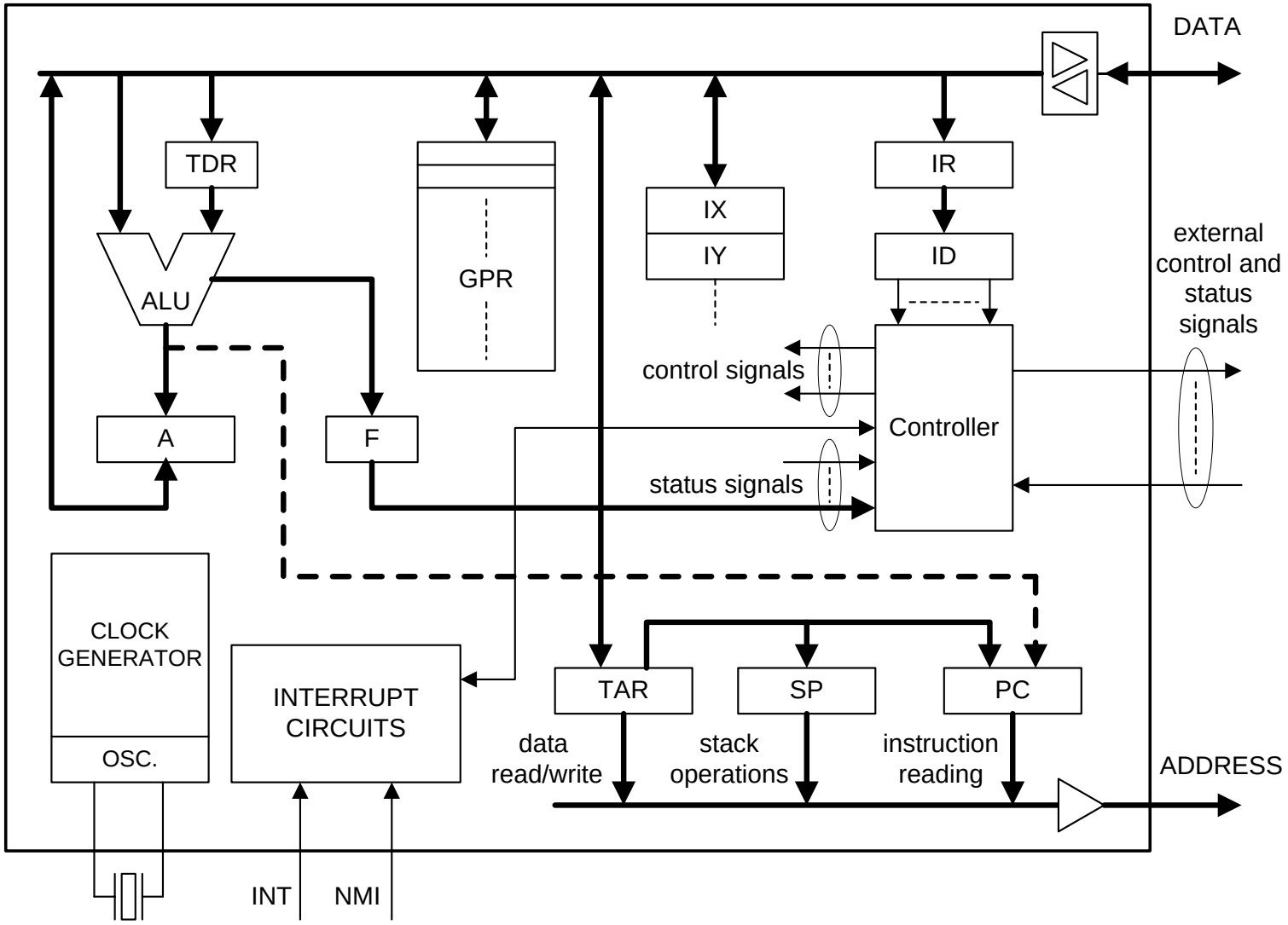
CISC - Complex Instruction Set Computer

Instruction sets of modern processors are very large. The result of this fact is very complex controller of the processor always implemented in a micro program machine manner.

RISC - Reduced Instruction Set Computer

Processor circuits optimized in order to maximize execution acceleration. The controller is designed like a typical sequence circuit with the emphasize on the maximal speed.

Basic CISC processor circuits



Internal processor structure attributes

PC – Program Counter – holds current program memory address

IR – Instruction register – instruction code storage

TAR - Temporary Address Register – auxiliary address register – address part of instruction storage

TDR - Temporary Data Register – auxiliary data register – data part of instruction storage

IX, IY – Index Registers

ID – Instruction Decoder

GPR – General Purpose Registers – user data registers

F – Flag register – CCR – Condition Code Register

SP – Stack Pointer – LIFO structure

ALU – Arithmetic-logical unit

Arithmetic operations:

addition, subtraction, sign inversion, multiplication and division

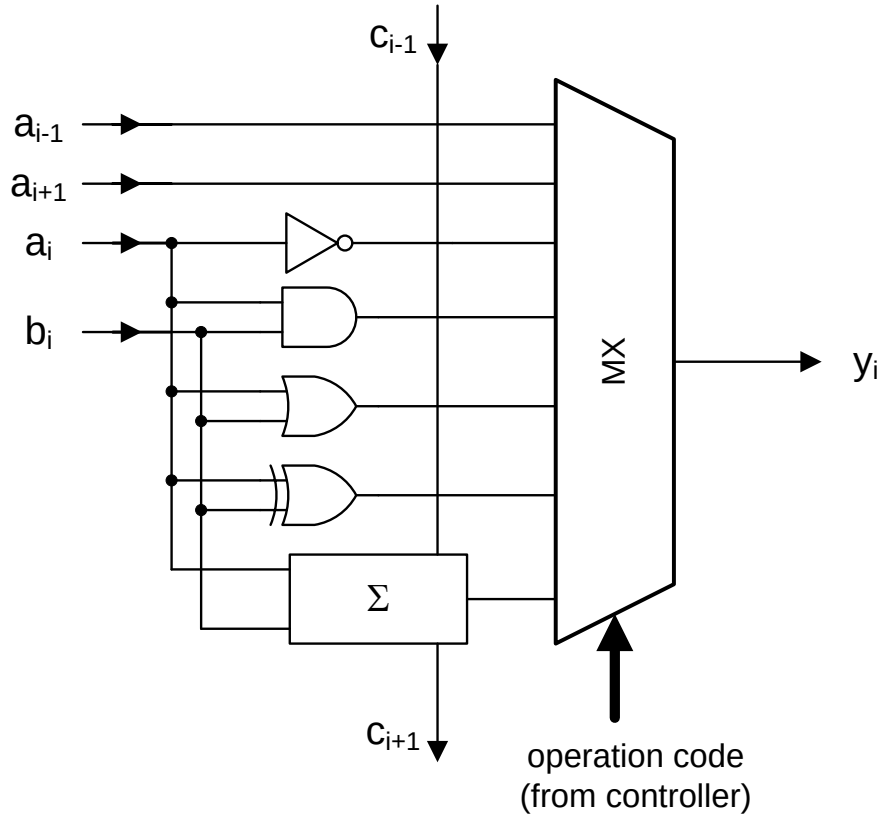
Logical operations:

AND, OR, EX-OR, negation, shift and rotation – left, right

Operands – data registers, memory

As a result of an operation is flag bits **F** settings – usually part of **PSW** – Program Status Word

ALU –1bit implementation example



Flag bits

Affected by a result of arithmetic or logical operation, eventually by the content of accumulator – instruction and processor architecture dependent.

Included in the Program Status Word register –**PSW**

C - carry – carry from the MSB bit (unsigned operations)

Z - **zero**

S - **sign** (or N – negative)

OV - overflow – result overflow (signed operations)

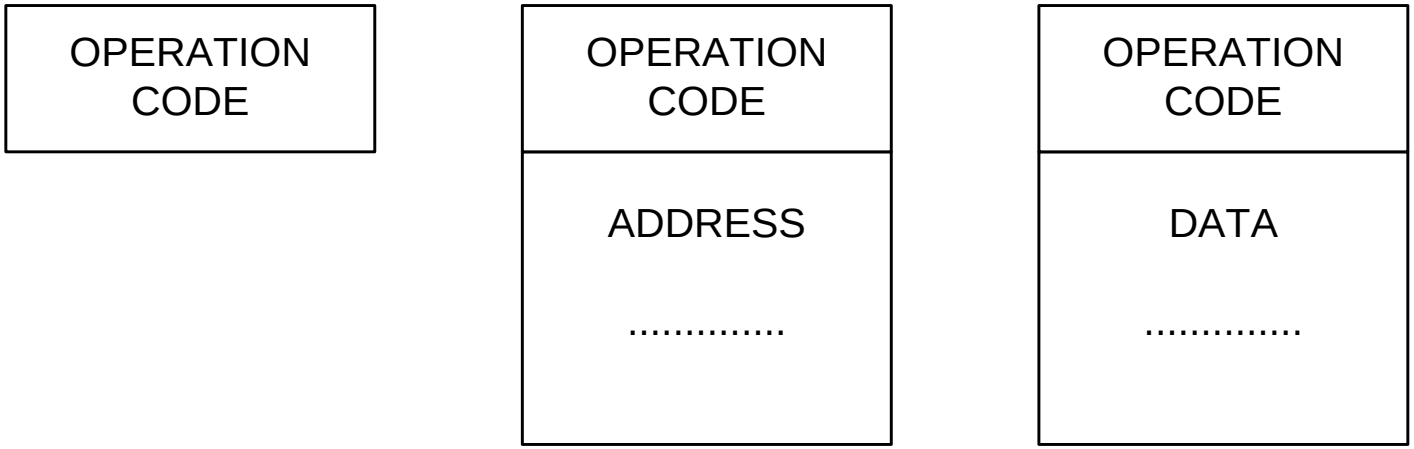
AC - **auxilliary carry** – half carry between two nibbles (4 bits)

P - **parity**

Basic instruction types

- **Data transfer – move instructions**
- **Arithmetic instructions**
- **Logical instructions**
- **Bit operations instructions**
- **Shift and rotation instructions**
- **Stack operations instructions**
- **Jump instructions**
- **Functions calling and return instructions**
- **Input and Output instructions**
- **Processor control instructions**

Basic instruction parts



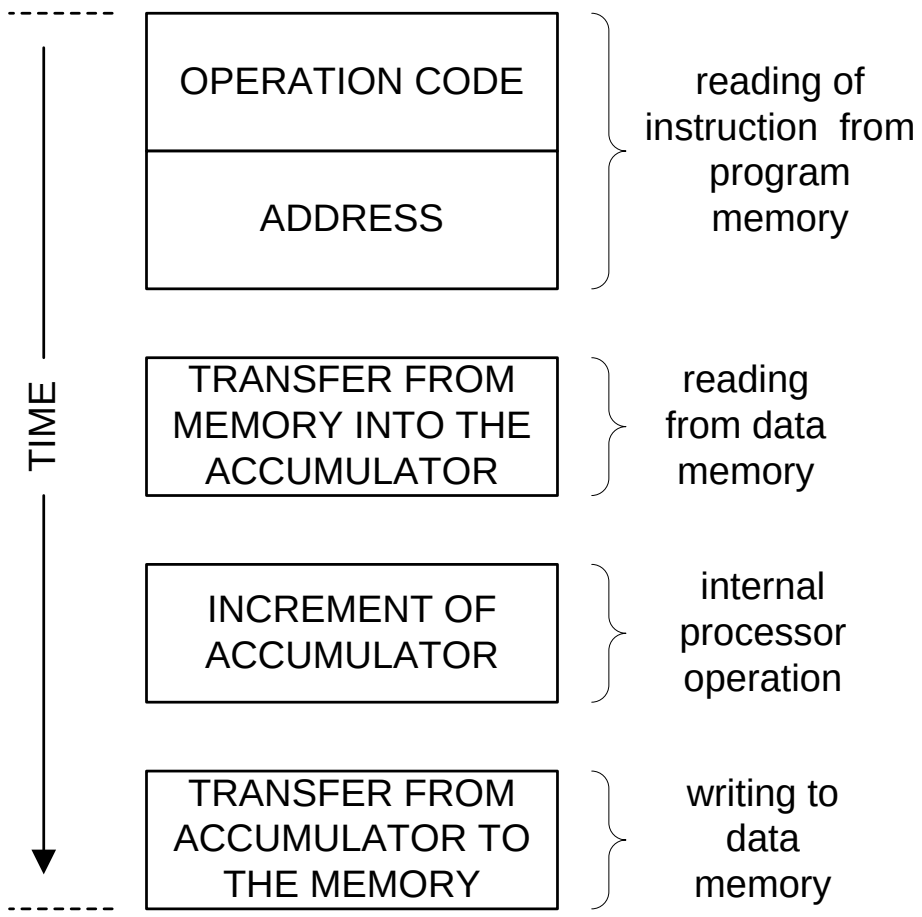
INC R0

ST R0, address

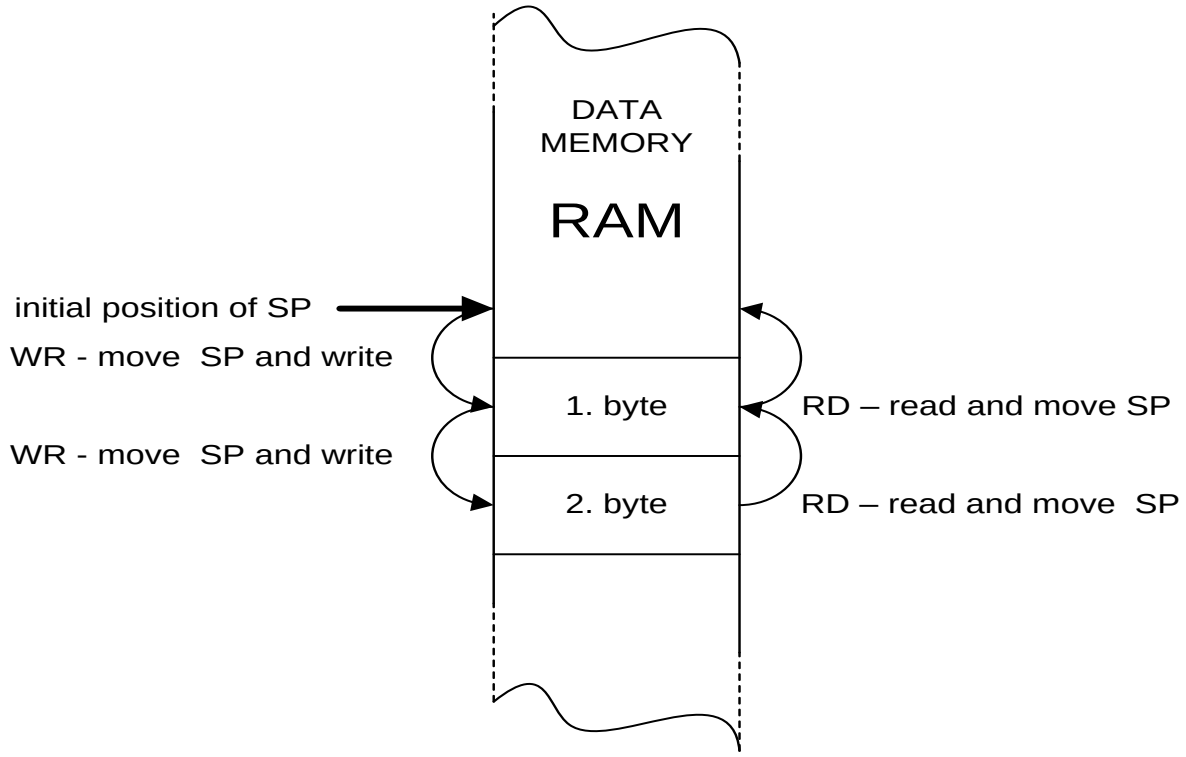
LD R1,#data

Instruction execution

„increment contents of the memory at the address..“



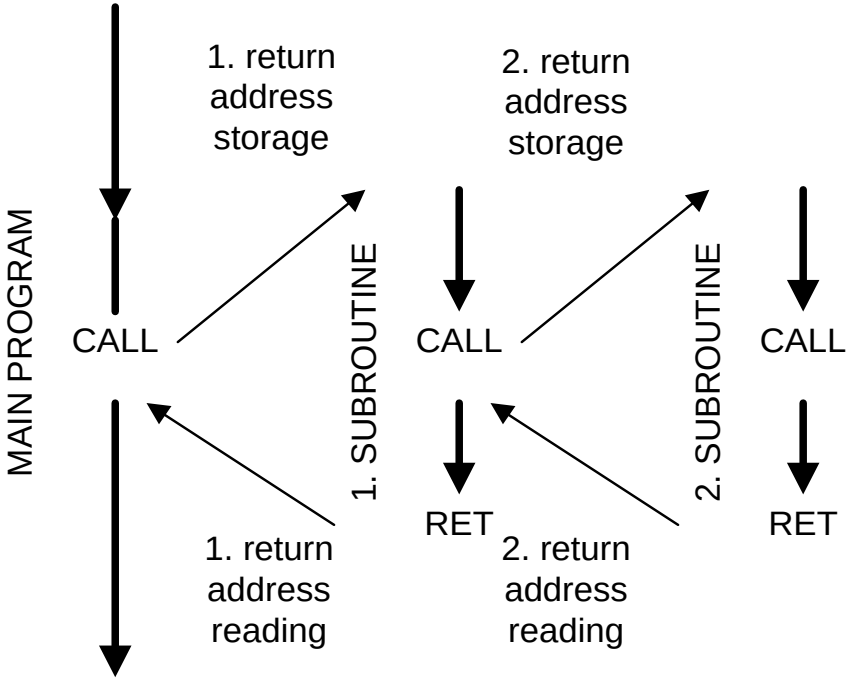
Stack pointer - SP



PUSH **data** **POP**

Saving of subroutine and ISR (Interrupt Service Routine) return addresses

Stack Pointer - SP



- SP **must** be initialized prior the first usage !!!
- **Look-out** stack overflow - underflow !!!

Processor execution acceleration

OP code reading from ROM	decoding	reading of address operand from ROM	RAM address generation	operand reading from RAM	arithmetic/ logic operation execution	next instruct.

Simple processor – execution sequence

← instruction cycle →				
reading + decoding of instruction 1	instruction 1 execution	reading + decoding of instruction 3	instruction 3 execution	
	reading + decoding of instruction 2	instruction 2 execution	reading + decoding of instruction 4	instruction 4 execution

Two-phase instruction overlapping - pipeline

Computer circuits

Multichip computer

All necessary function units are external with regard to processor

- design complexity, larger space
- HW configuration flexibility

Single-chip microcomputer

All components integrated on the chip. Predefined internal structure

- compactness, space savings.
- configuration and initialization by using large amount of control registers
- number of I/O pins of the device limited – shared I/O and peripheral functions

Computer circuits interconnection

Bus system is used to interconnect processor to the other computer circuits – address, data and control bus

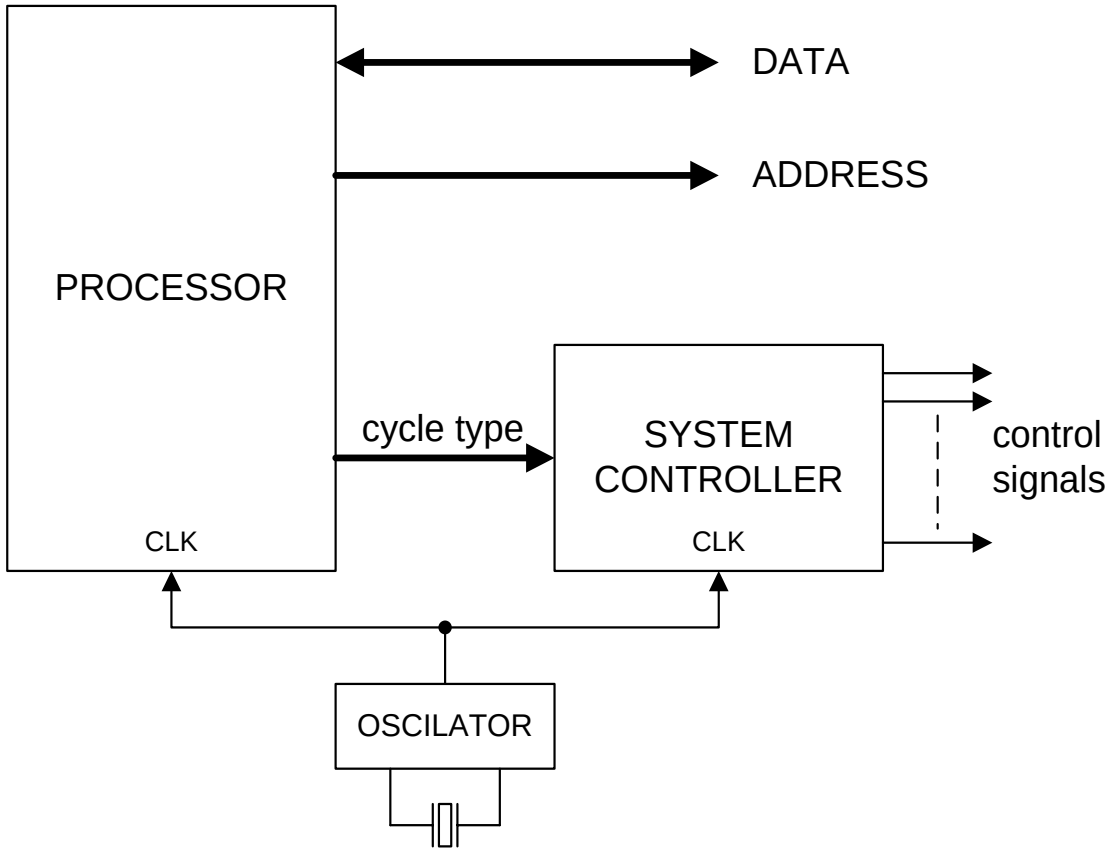
External bus signals and their timing knowledge is necessary to design system of external circuits.

Data exchange provided by **bus cycles**:

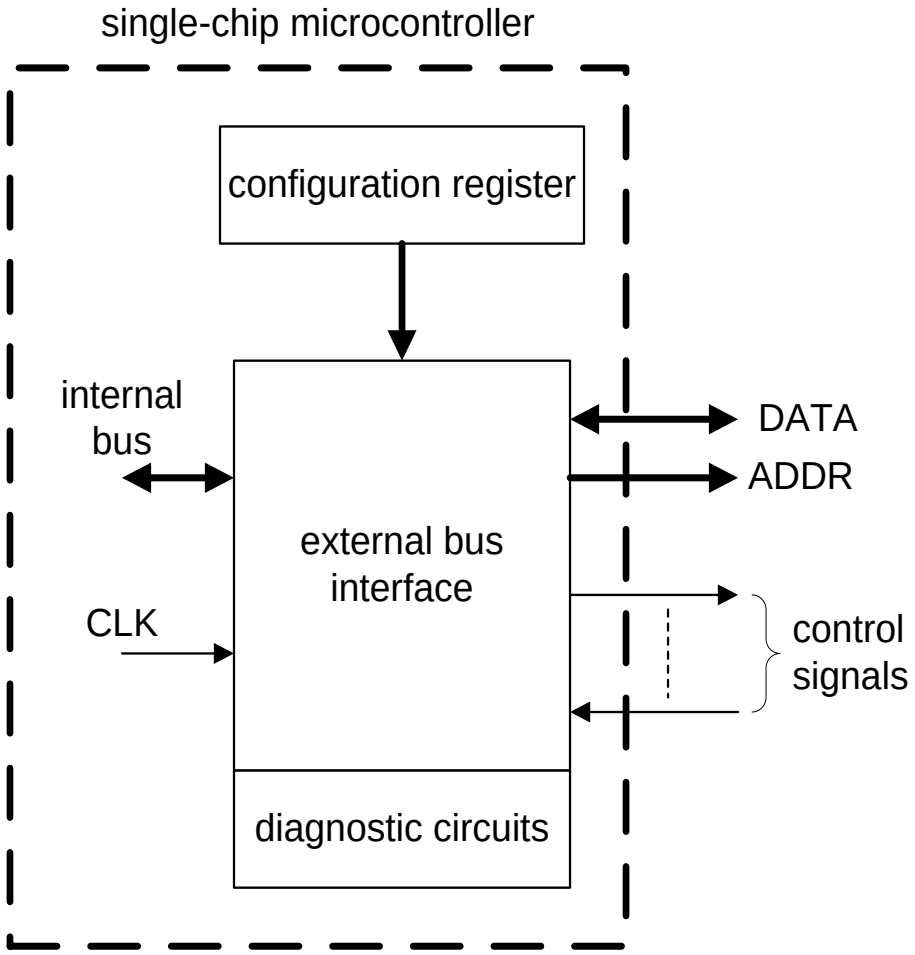
- memory read cycle
- memory write cycle

Control bus signals generated by system controller

External system controller



Internal system controller of the single-chip microcontroller



Any possible external circuit extension is solved using address, data and control bus pins.

To avoid excessive number of device pins these external buses often share pins with other peripheral functions or general purposed parallel inputs/outputs - GPIOs

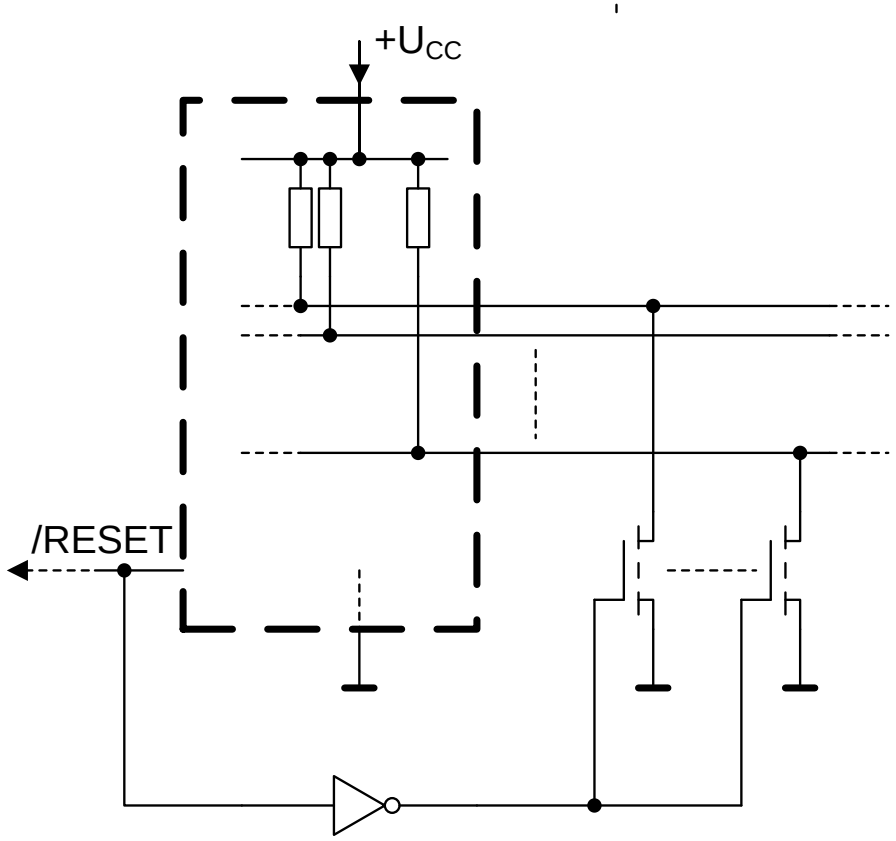
Configuration register - single-chip microcontroller

Operation mode - 16-bit or 8-bit width data bus, address and data bus multiplex, timings, signals polarity, wait states…

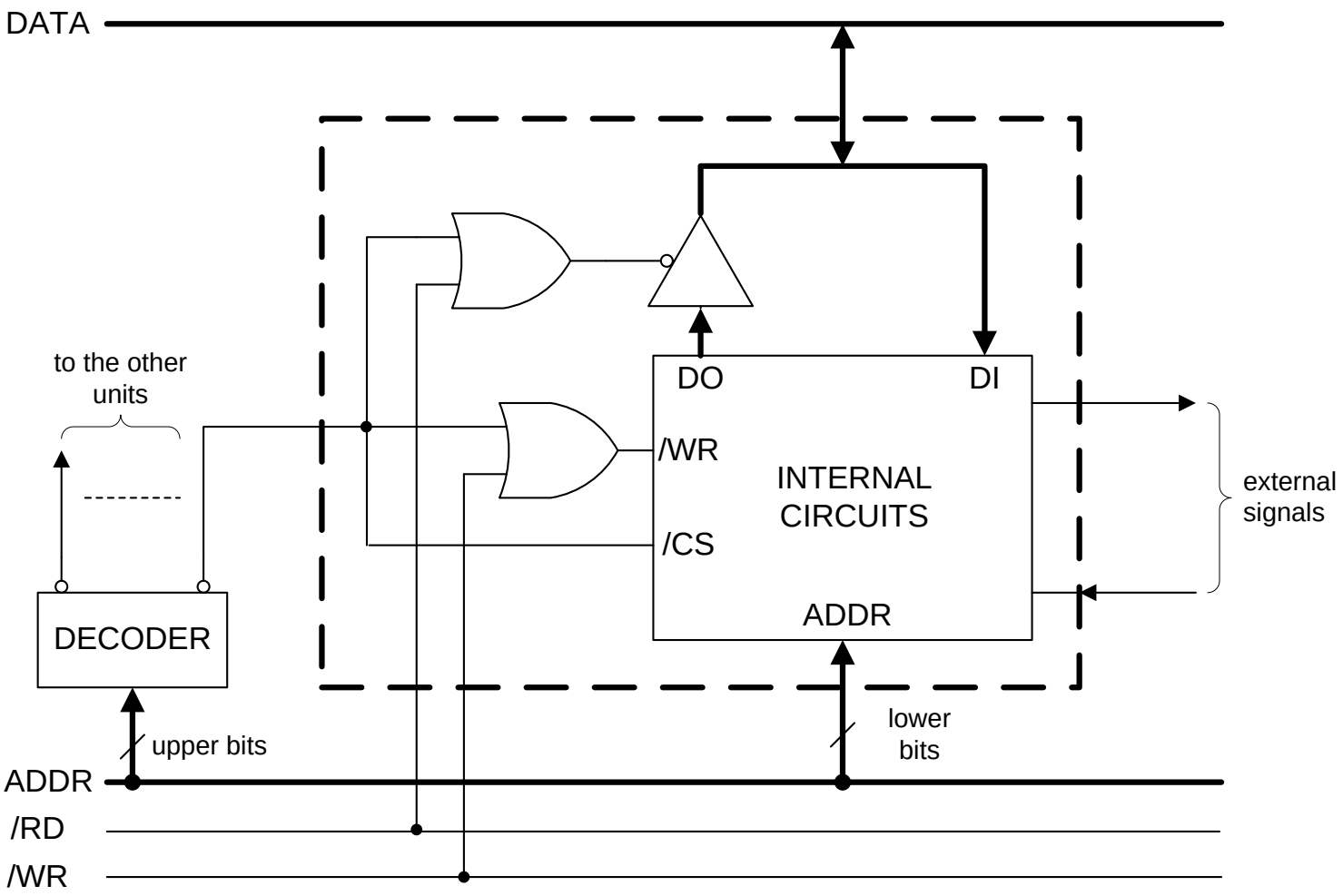
External bus operation mode is possible to be changed through the system controller programming - equipped with **configuration register** for these purposes.

Configuration register – initiated and affected by the sampled status of particular pins during reset on power up (/EA), eventually consecutively by the initiation program routine.

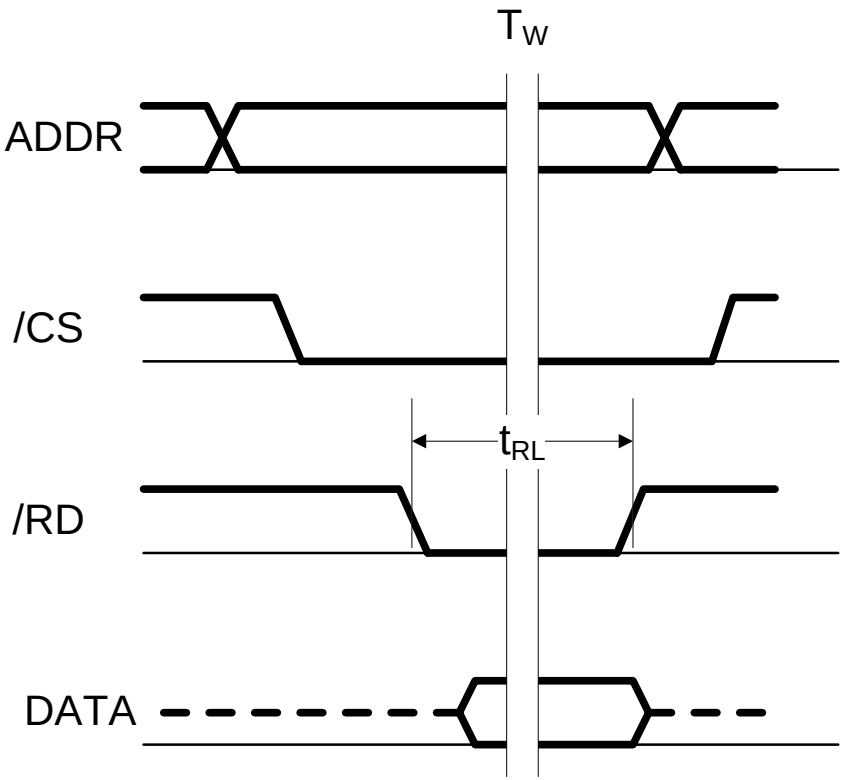
Example – configuration settings using I/0 pins



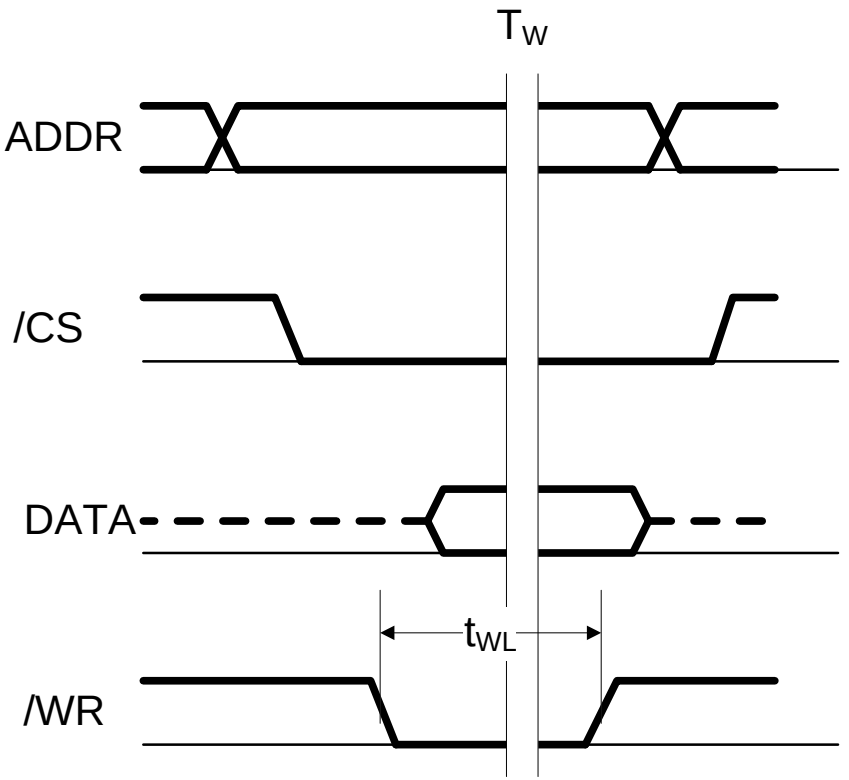
Unit connection to external buses



Basic bus cycles



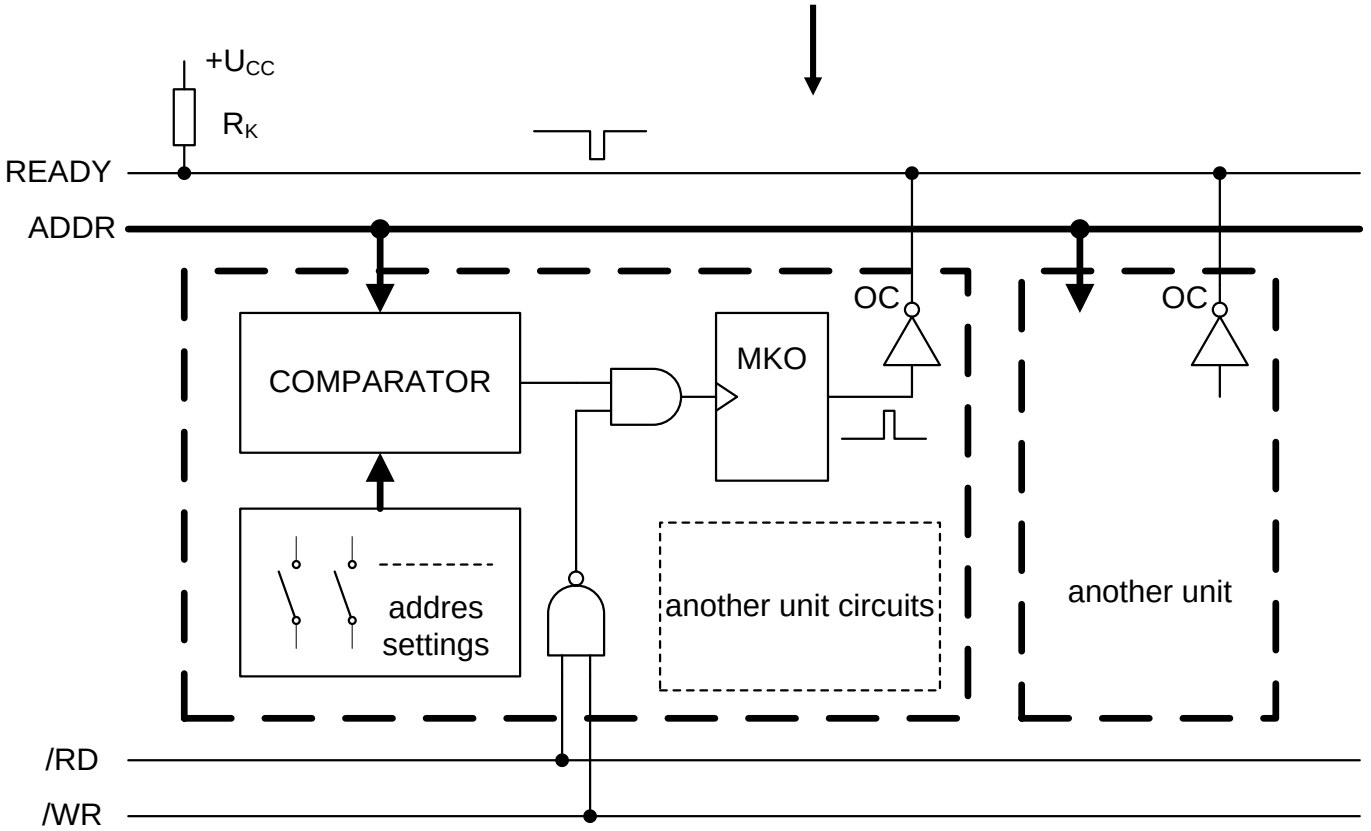
Read cycle



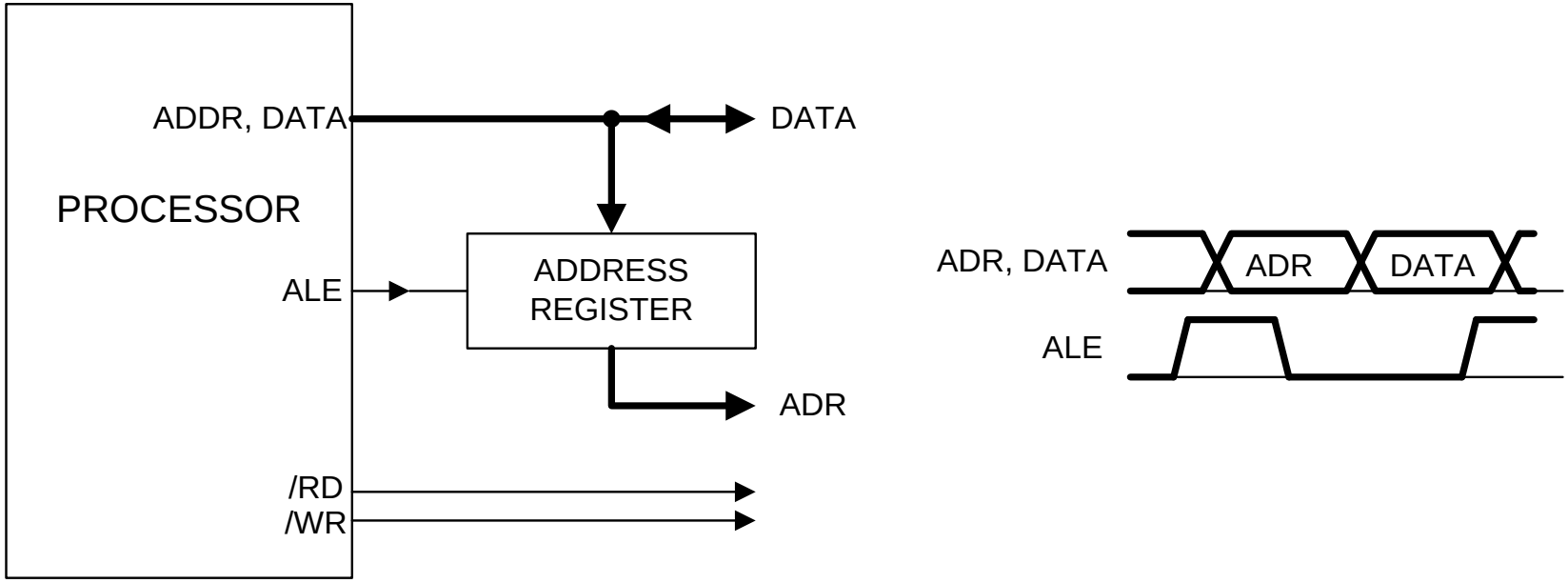
Write cycle

Wait states - READY- WAIT

- **Central control – Wait-State Generator**
- **Decentralized timing circuits**

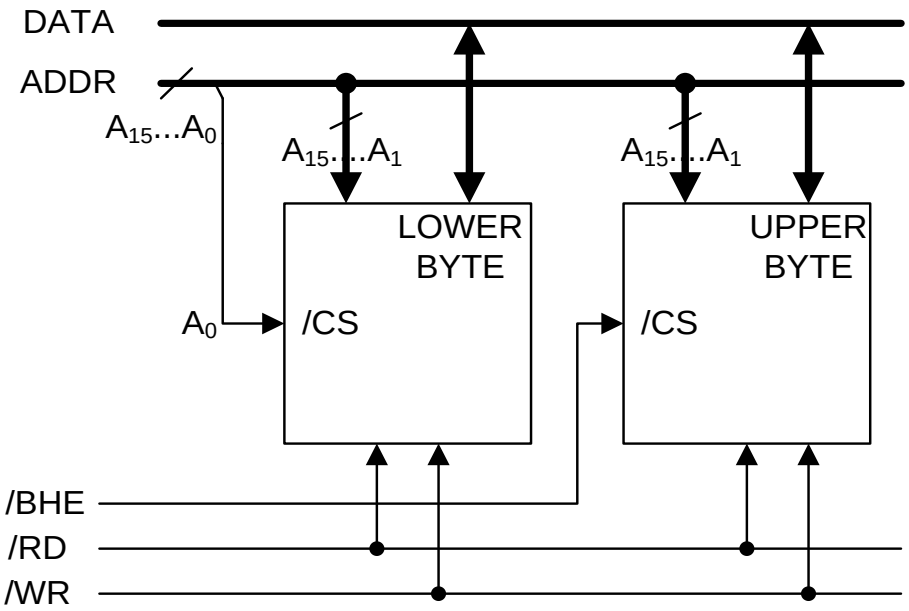


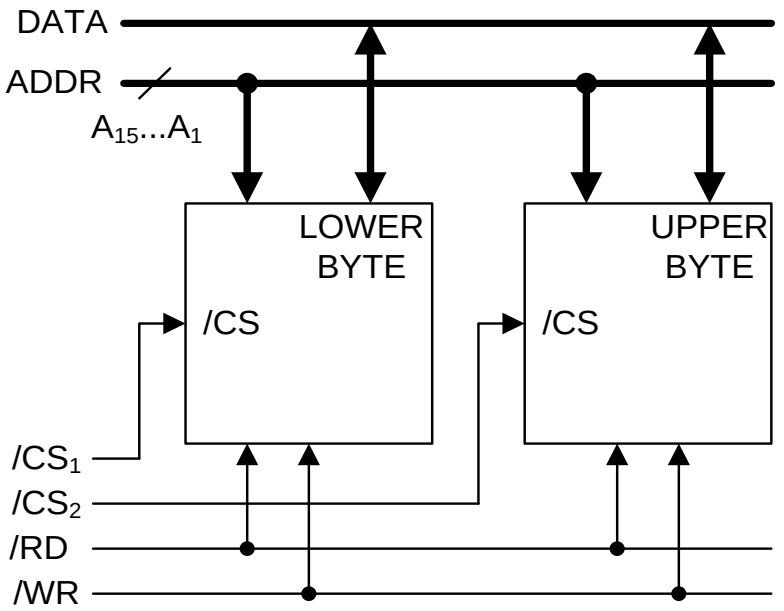
Multiplexed bus



- To save number of device pins
- Bus cycle slows down as a result of address latching

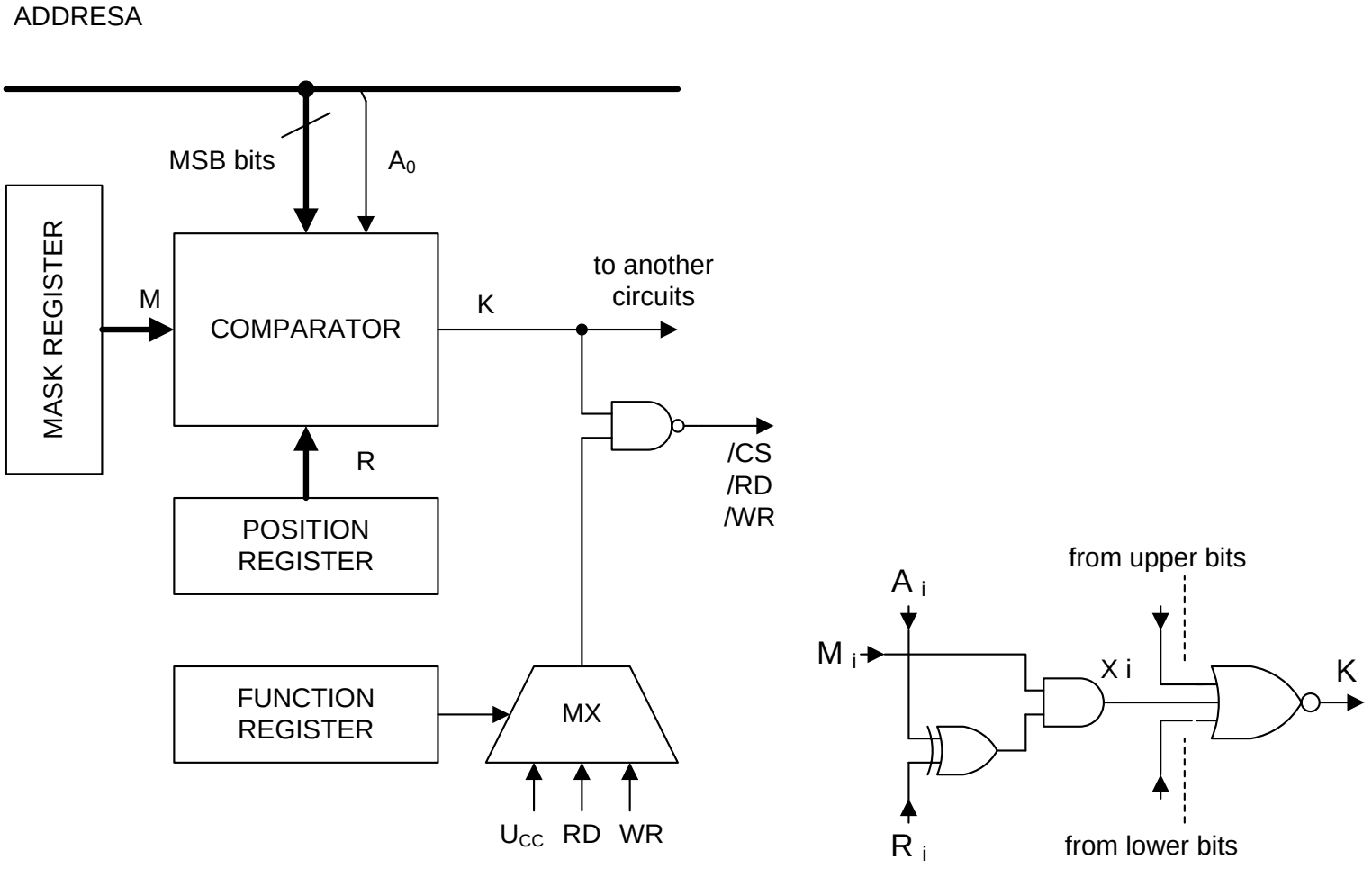
External memory connection - byte, word



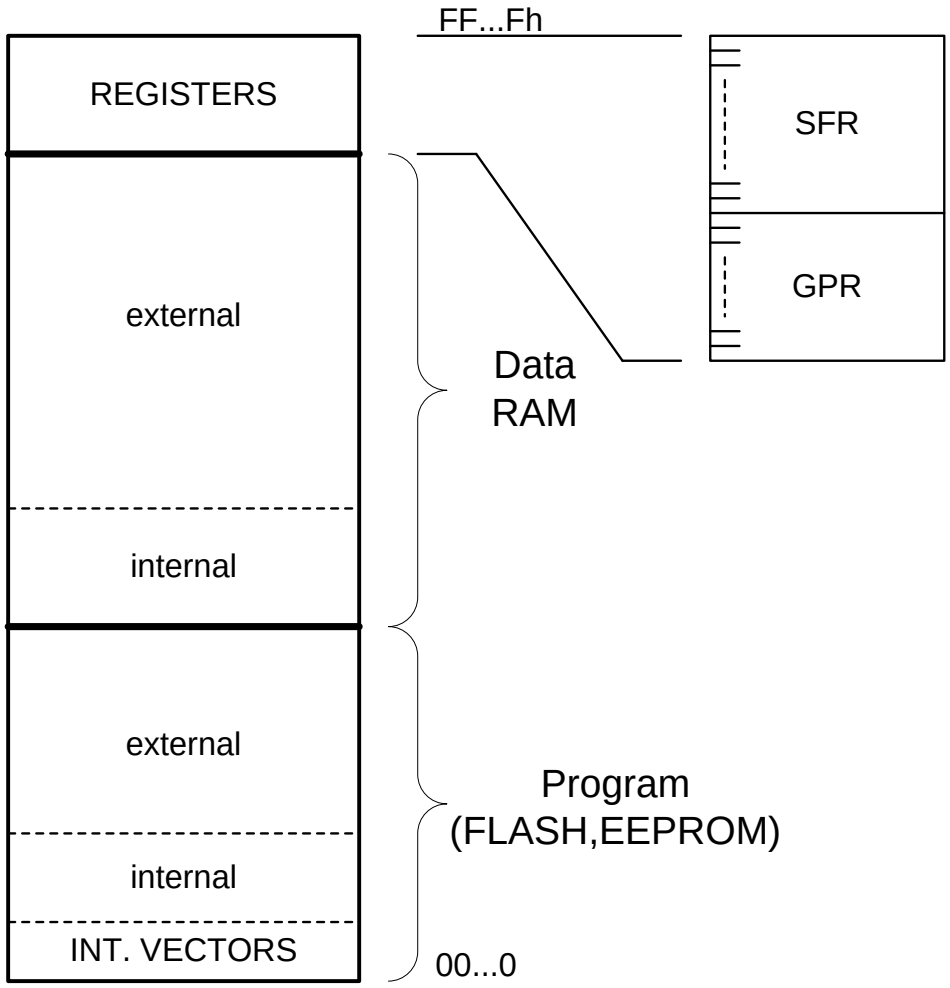


- In the case of **16-bit data bus** the external units can operate either full data width or 8-bit data width
- Possible transfer of **upper byte, lower byte**, or both of them at the same time – **complete word**

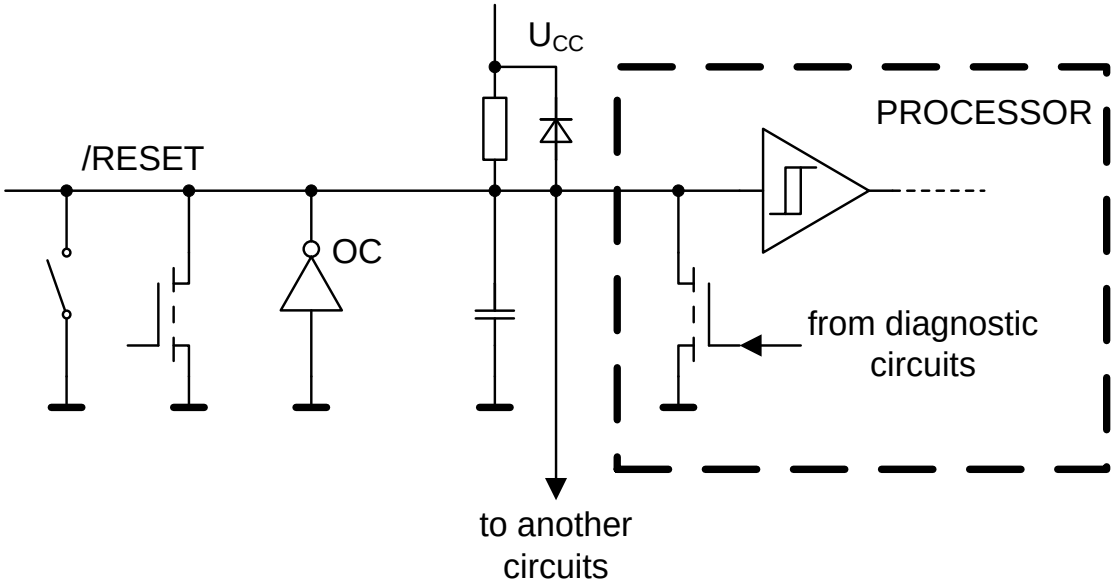
Address decoder and select logic



Memory map


FF...Fh
REGISTERS
external
internal
external
internal
INT. VECTORS
SFR
GPR
Data
RAM
Program
(FLASH,EEPROM)
00...0

Computer initialization - RESET



PC = „reset vector“ initial program address

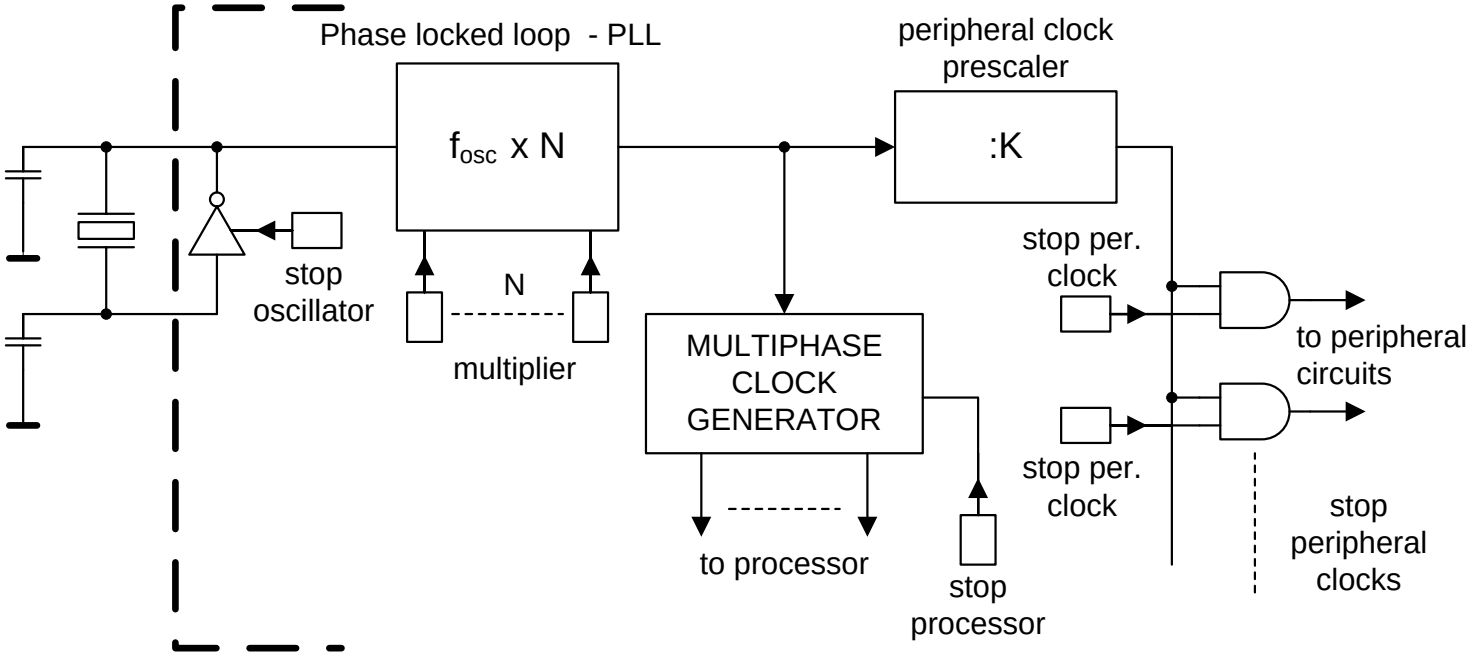
IE = 0 disable interrupts

GPIO set registers to input mode

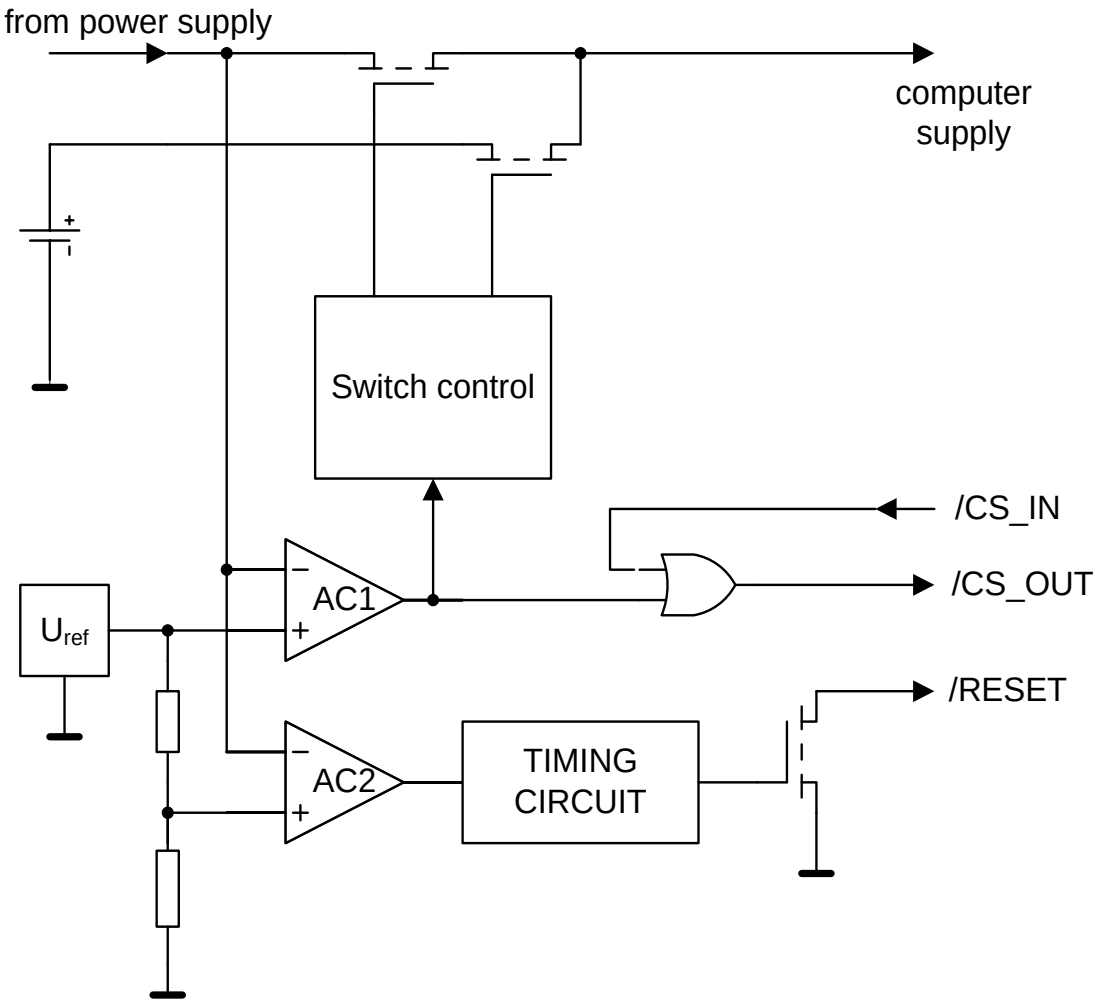
SP initialization (µP dependent)

GPR, SFR, WDT, BCU (wait, /CSi,addresses) initialization (µP dependent)

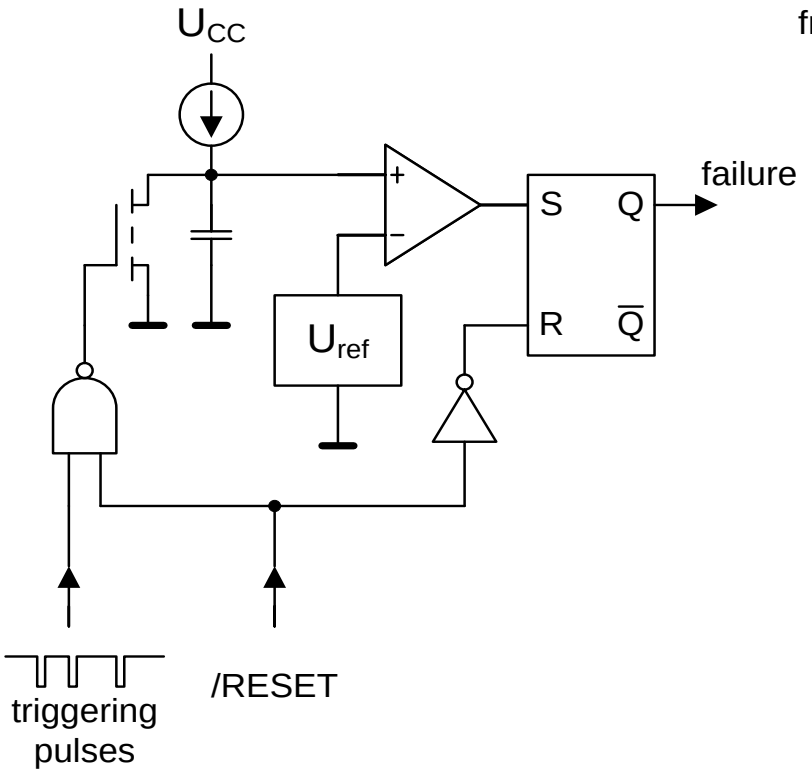
µP clocks

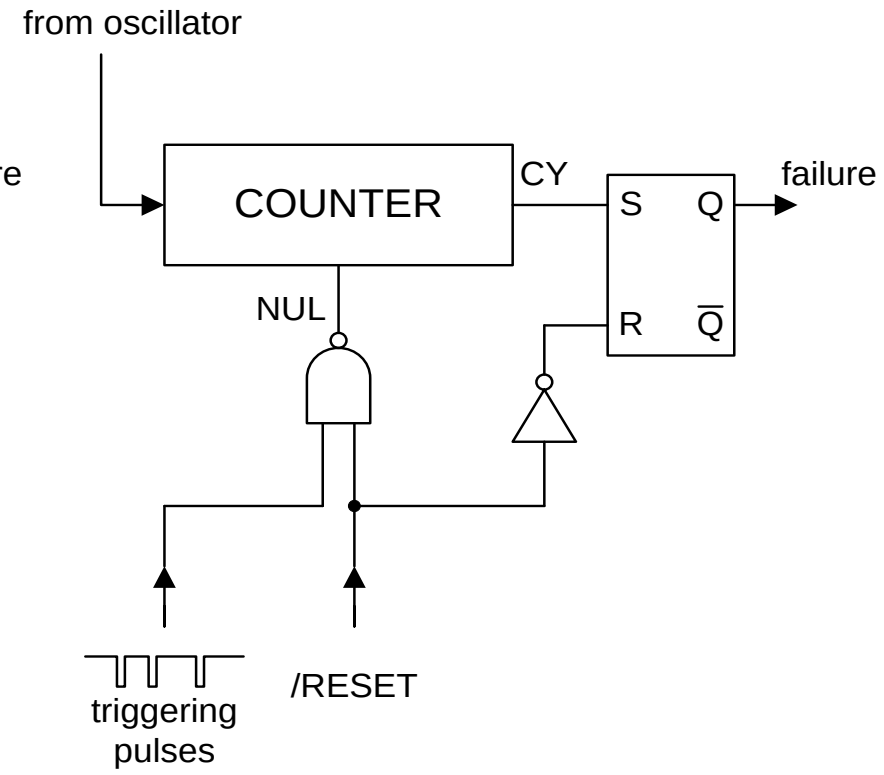

Phase locked loop - PLL
peripheral clock prescaler
f_{osc} x N
:K
stop oscillator
N
multiplier
MULTIPHASE CLOCK GENERATOR
stop per. clock
to peripheral circuits
stop per. clock
stop peripheral clocks
to processor
stop processor

Supervisor – control circuit



Diagnostic tools of computer
Watchdog Timer

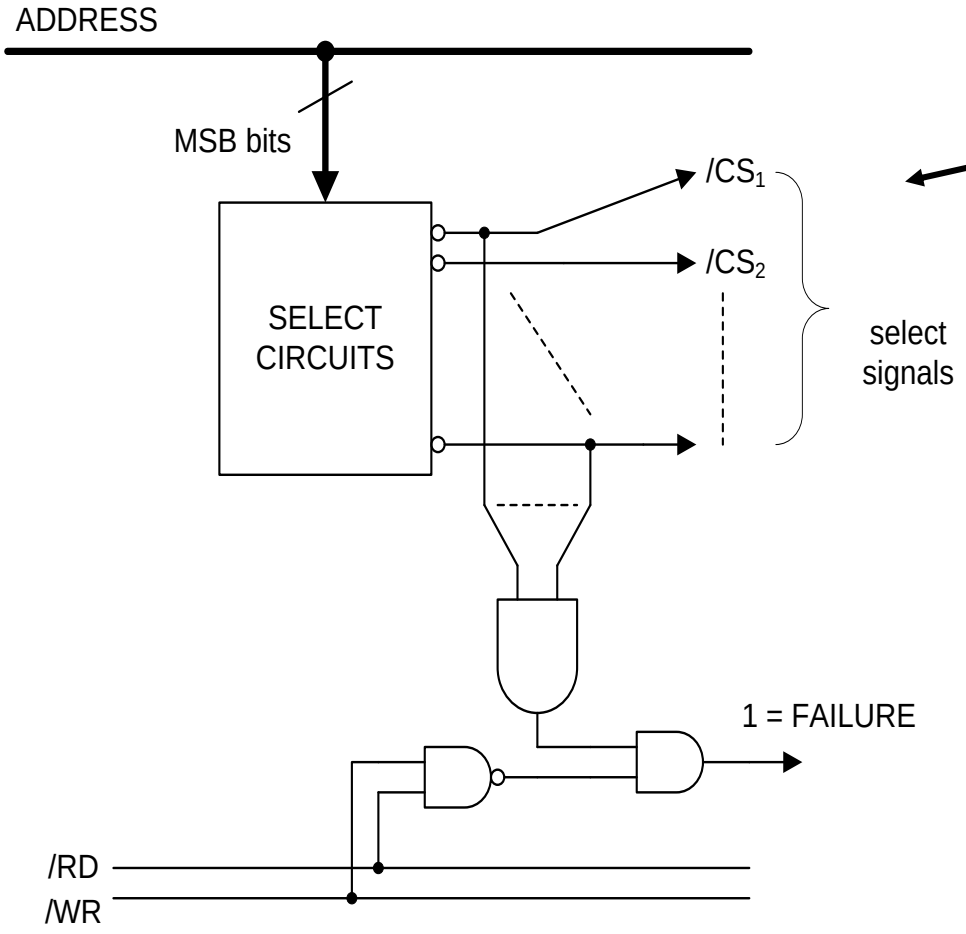




Asynchronous

Synchronous

Diagnostic tools of computer



- **Valid address check**
- **Unimplemented instruction codes**
- **Stack overflow, underflow**

Interrupts

Unexpected external event – it can happen any time:

- **asynchronous** signal derived from external processor pin – logical signal
- **asynchronous** request for data processing, coming from external or internal peripheral
- **asynchronous** request coming from processor diagnostic circuits at the exceptions (for example zero division, instruction read error, unimplemented codes…)
- software call – special instruction SWIi – debugging and diagnostic purposes

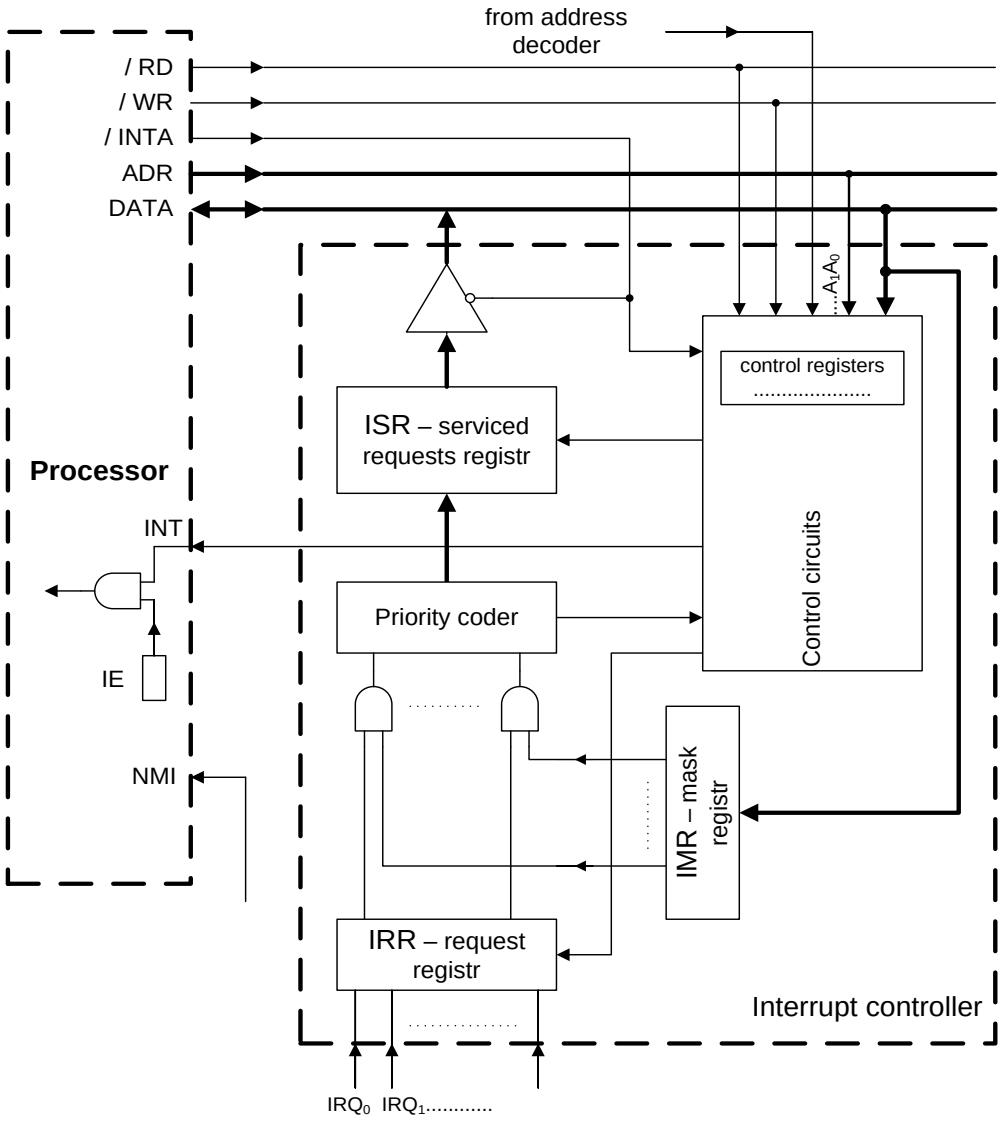
Interrupts

- As a result of unexpected external event processor immediately modifies its activity – stops the main program execution and jumps to the interrupt service routine corresponding the source interrupt vector
- Return address from the Interrupt service routine ISR and important processor's registers are saved onto the stack
- After ISR processing are these ones restored from the stack and program flow returns to the original point

Interrupt types:

- **INT** – mask able – it is possible to disable through global mask
- **NMI** – non mask able – the highest priority

Interrupt controller - external



IRR – holds input requests IRQi for necessary time prior their processing by the next circuits

IMR – mask interrupt register

ISR – serviced requests registr – it determines interrupt type

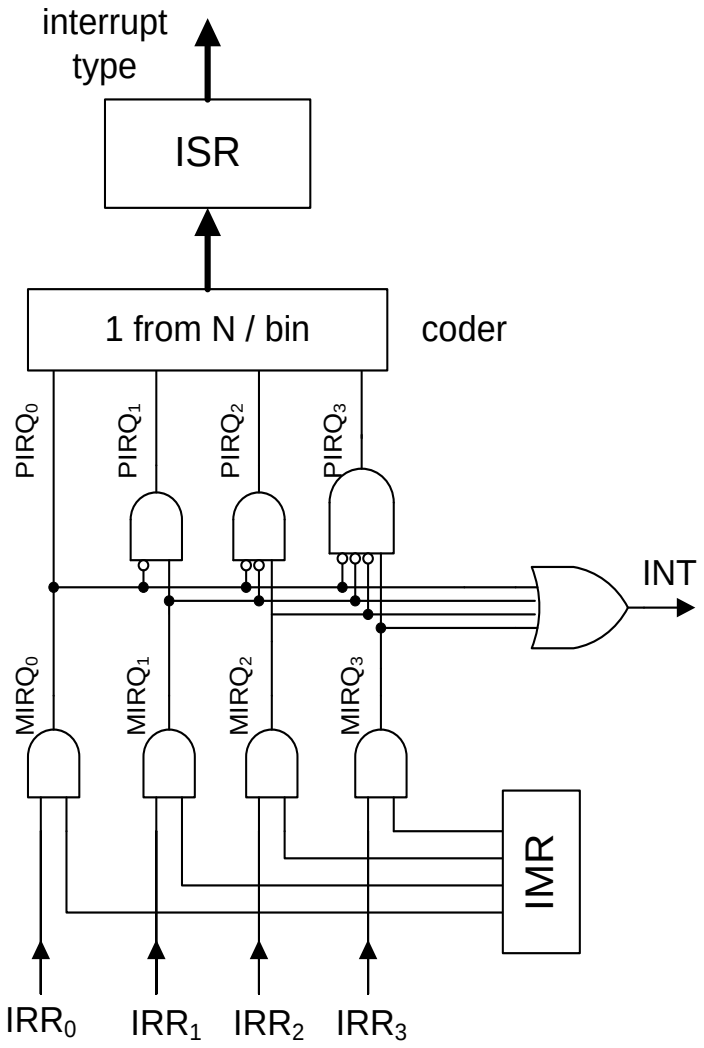
IRQi – level, edge

IE – interrupt global mask (part of PSW, CCR register)

/INTA - Interrupt Acknowledge

Interrupt priority

- **Fixed**
- **Cyclic**
- **Random**
- **Level based**



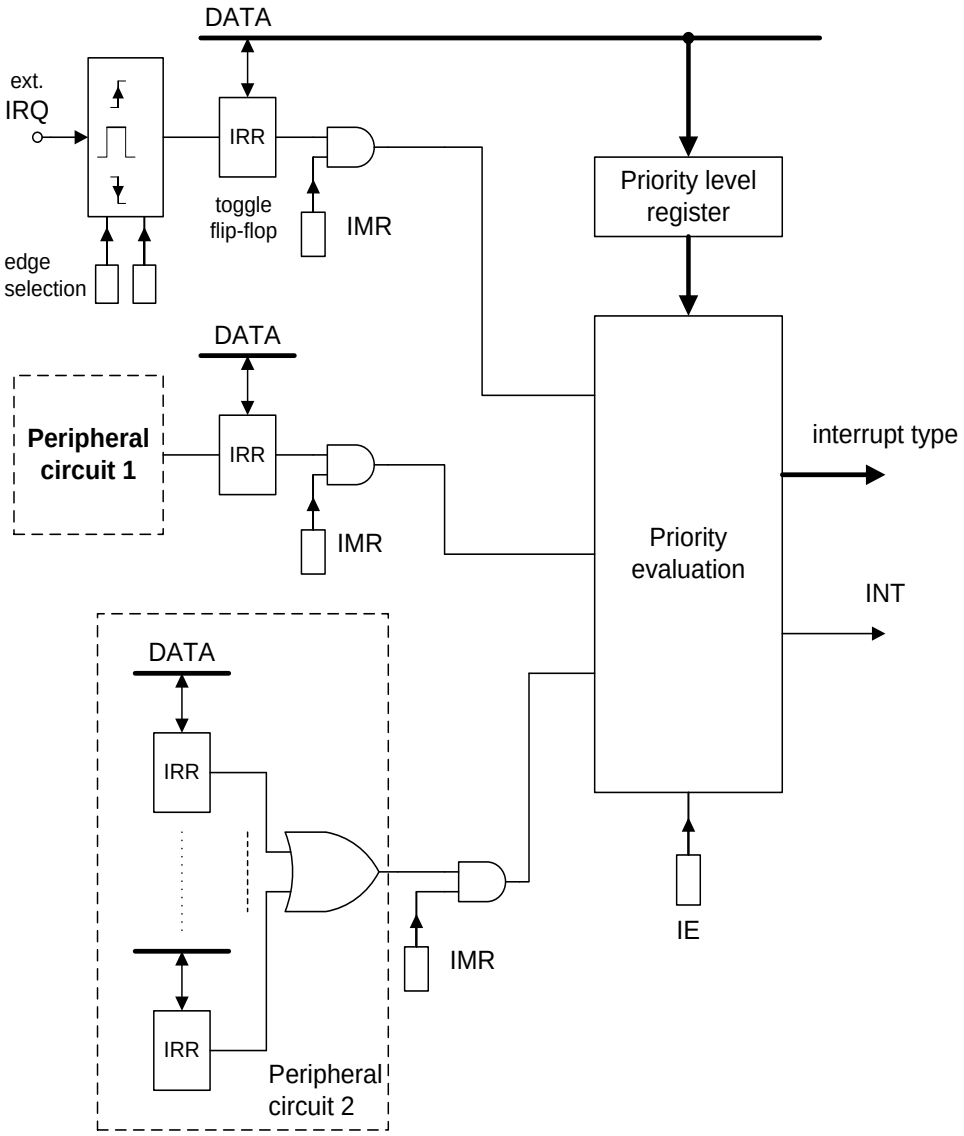
Interrupt level register

level H				IRQ_3	IRQ_4		IRQ_6	IRQ_7
level L	IRQ_0	IRQ_1	IRQ_2			IRQ_5		
	← higher ..... **priority** ..... lower →							

Interrupt sequence

- Provided **IRQi** is not masked and concurrently has got the highest priority, the interrupt request **INT** is generated to processor from interrupt controller
- If global mask **IE** is enabled, interrupt service is started
- Current processing instruction is finished
- The address of instruction following current instruction is saved onto the stack as the return address
- Interrupt type is read by the processor from the interrupt controller's register **ISR** – through signal **INTA/**
- Corresponding flip-flop at the register IRR is cleared
- Global interrupt mask **IE** is cleared
- Address assignment - corresponding interrupt service routine address is assigned – ISR vector table, usually in the ROM (FLASH) memory. Control is passed to this routine.
- PSW, accumulator, important registers are saved onto the stack – ISR software is responsible for this activity!!! –**PUSH** instruction
- Global mask **IE** cold be optionally enabled for **IRQi** of higher priority processing
- Interrupt service
- All saved registers (PSW, accumulator …) are restored from the stack – **POP instruction**
- Return to the main program – **RETI** instruction

Single-chip microcomputer interrupts

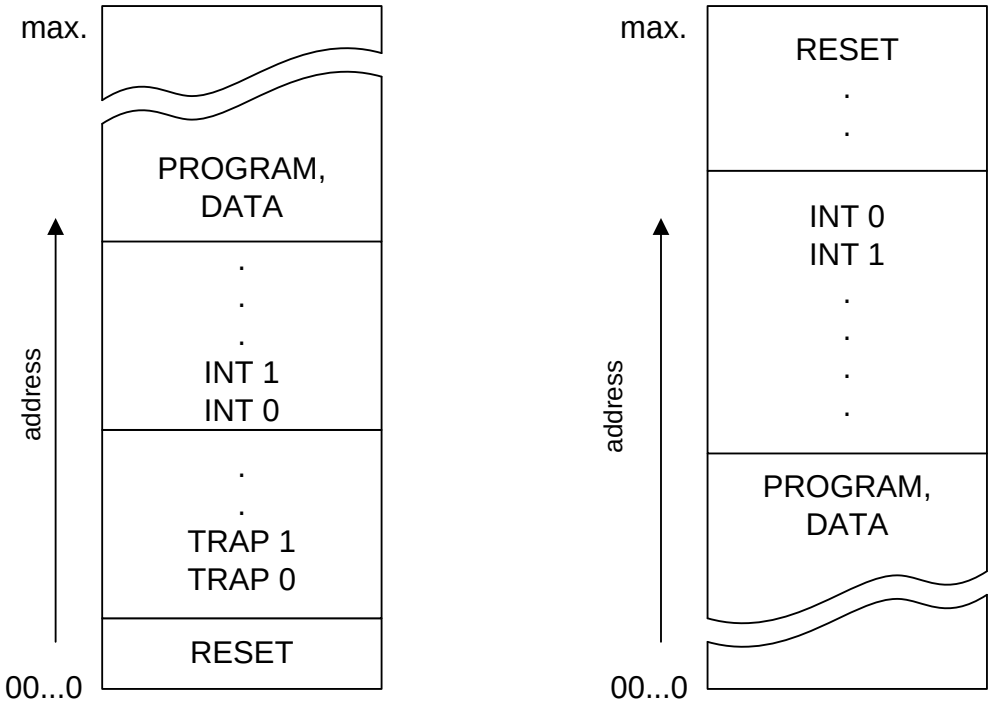


Interrupt circuits connected directly to the peripheral circuits

Processor has got the access to all interrupt controller circuits – not necessary to communicate through the external bus. IRR can be tested by program.

IRR and IMR bits are aggregated either in the common registers or they are included as a part of corresponding control peripheral registers.

Interrupt vectors table



Unused vectors are suitable to service !!!

Interrupt latency

The time from the occurrence of the IRQ request till the time of the first instruction of interrupt service routine execution. Typically several microseconds.

Tl_max = Tinst_max + Tproc

Tinst_max – the time of execution of the longest uninterruptable instruction

Tproc - the time for acceptance and processing of the interrupt

Memory

ROM

Mask programmable. The contents is defined during production. Big series

EPROM

Programming - programmer, erasing - UV ultraviolet radiation, reprogramming in order **10^1 x**

PROM

EPROM without window – not possible to erase – **OTP** memory

EEPROM

Programming - programmer, in system – per unique addresses. Slow programming (compared to EPROM or FLASH). **$10^4 \div 10^5$ x**

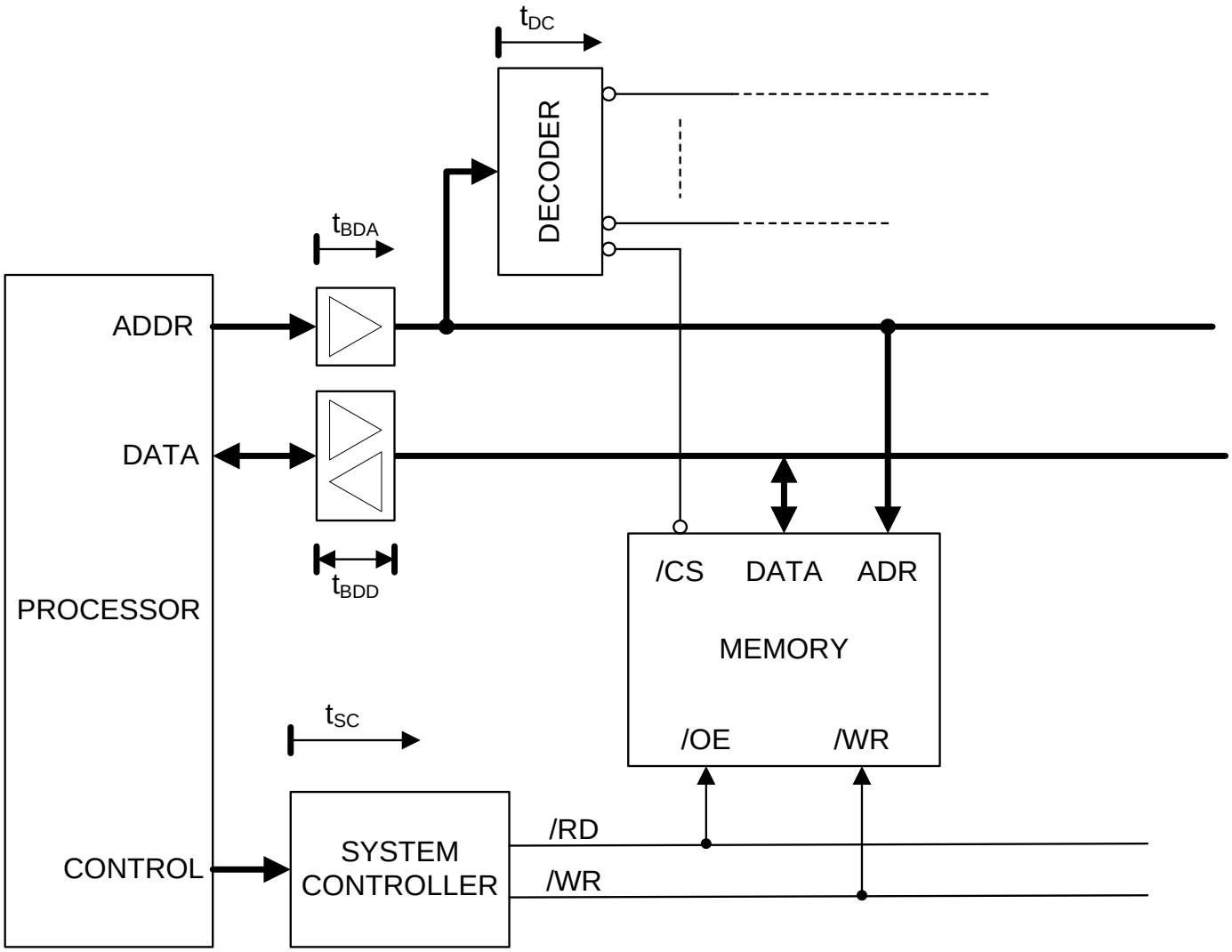
FLASH

Organized per sectors. Programming - programmer, in system – per unique addresses. Erasing – whole memory or per sector. Higher capacity.**$10^4 \div 10^5$ x**

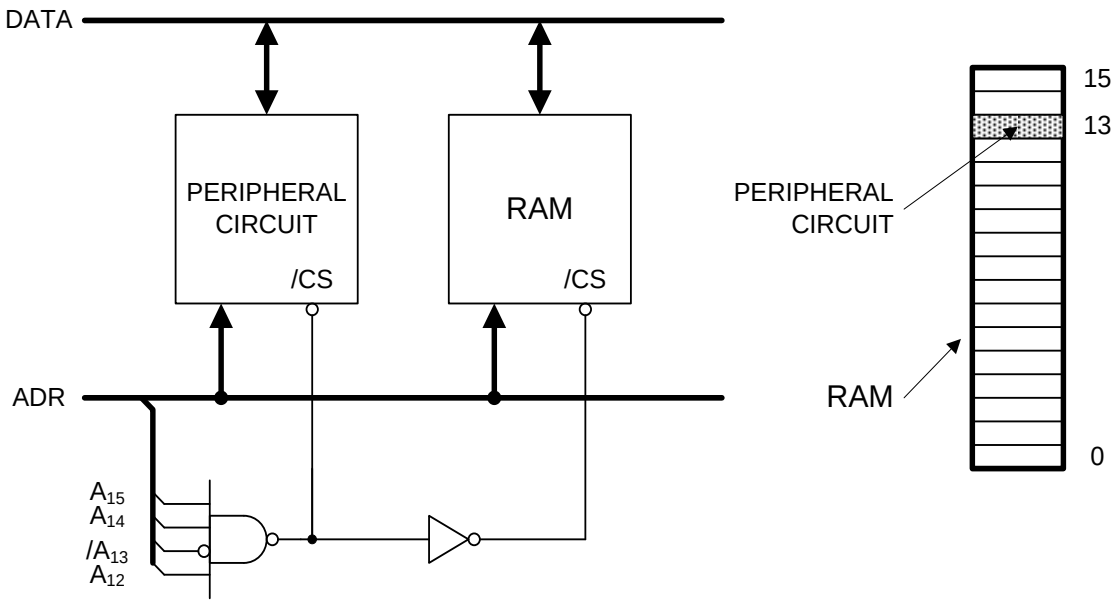
RAM

Fast nonvolatile memory. Data and variable storage. Static – CMOS, dynamic - PC

External memory

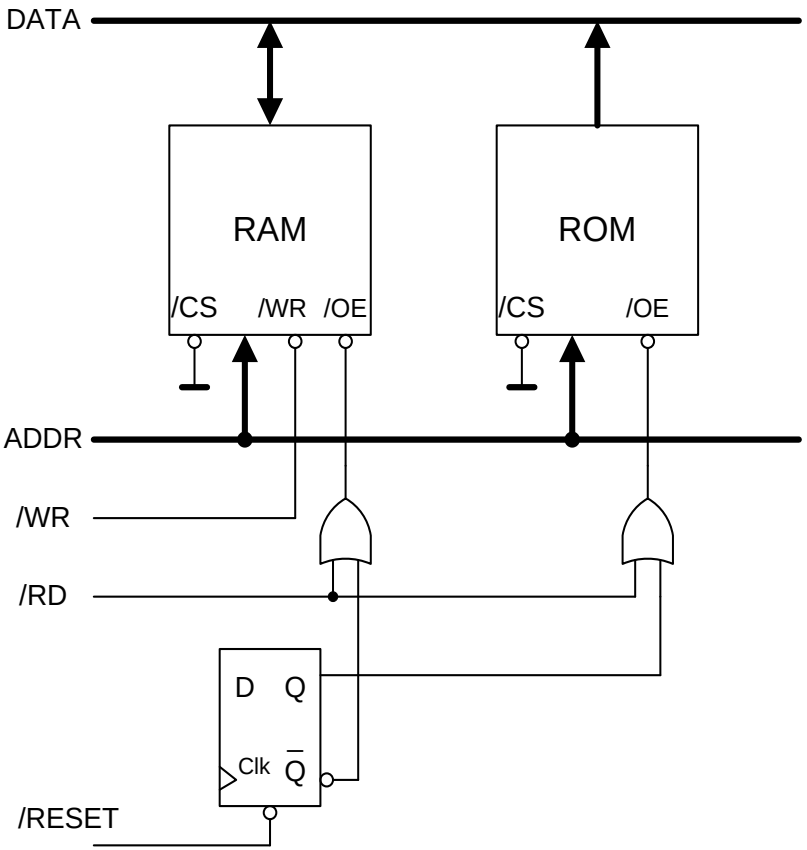


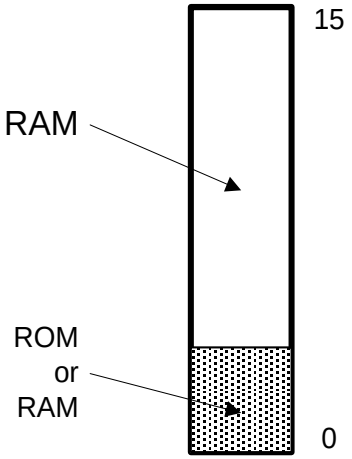
Memory overlapping



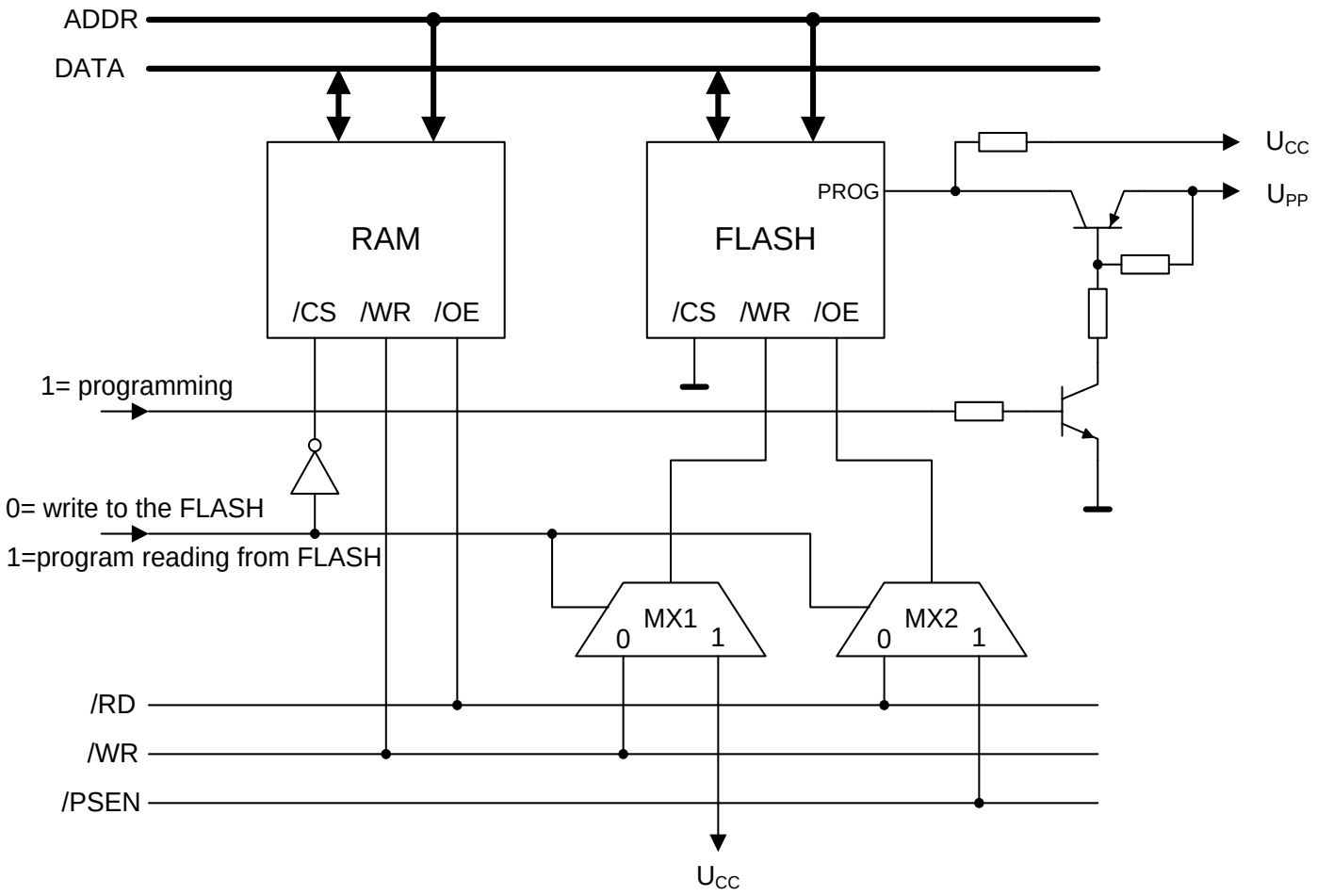
1101 XXXX XXXX XXXX$_B$ **Peripheral address space**

Controlled overlapping – shadow ROM

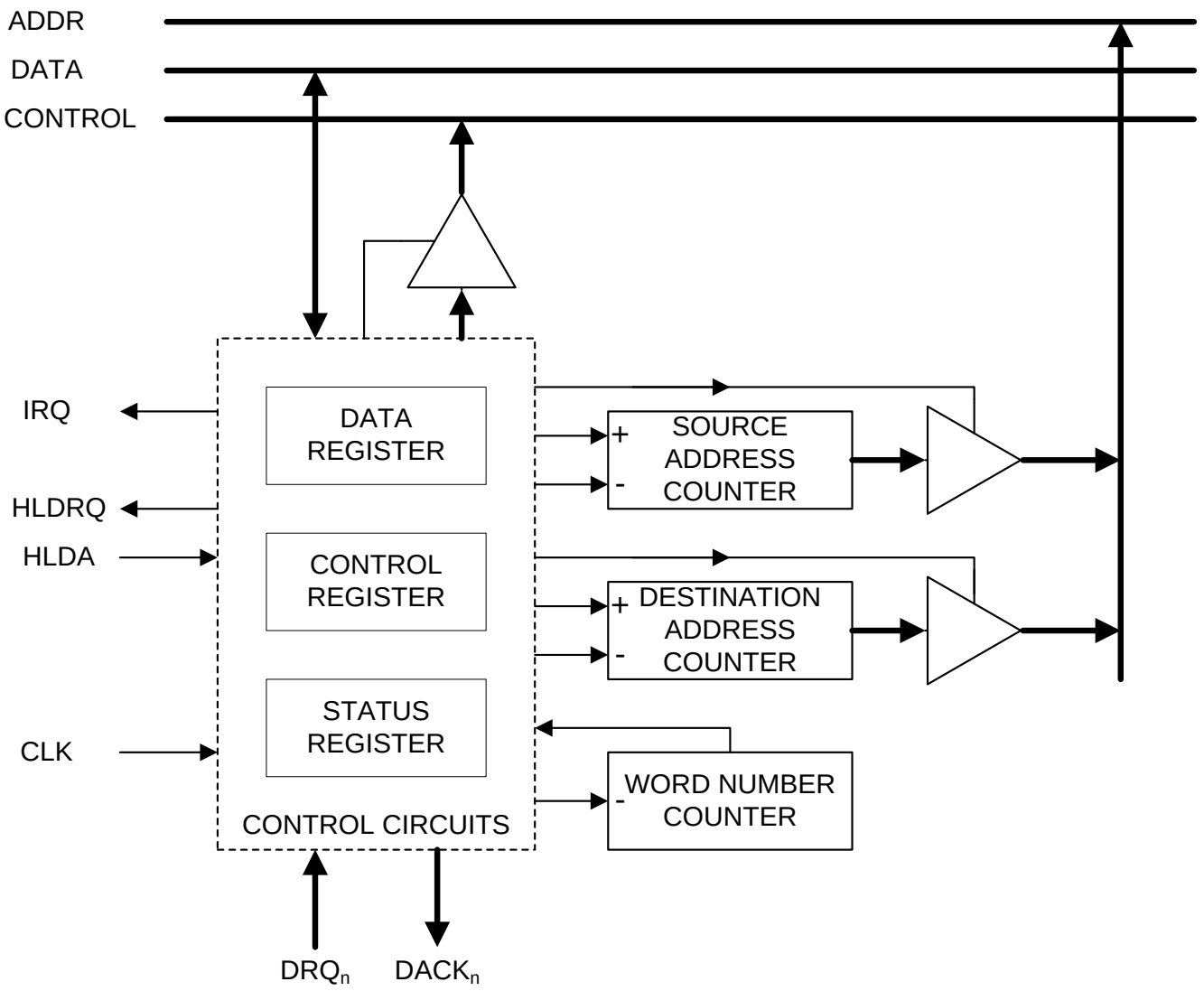




In system FLASH memory programming



Direct memory access - DMA



Timers - counters

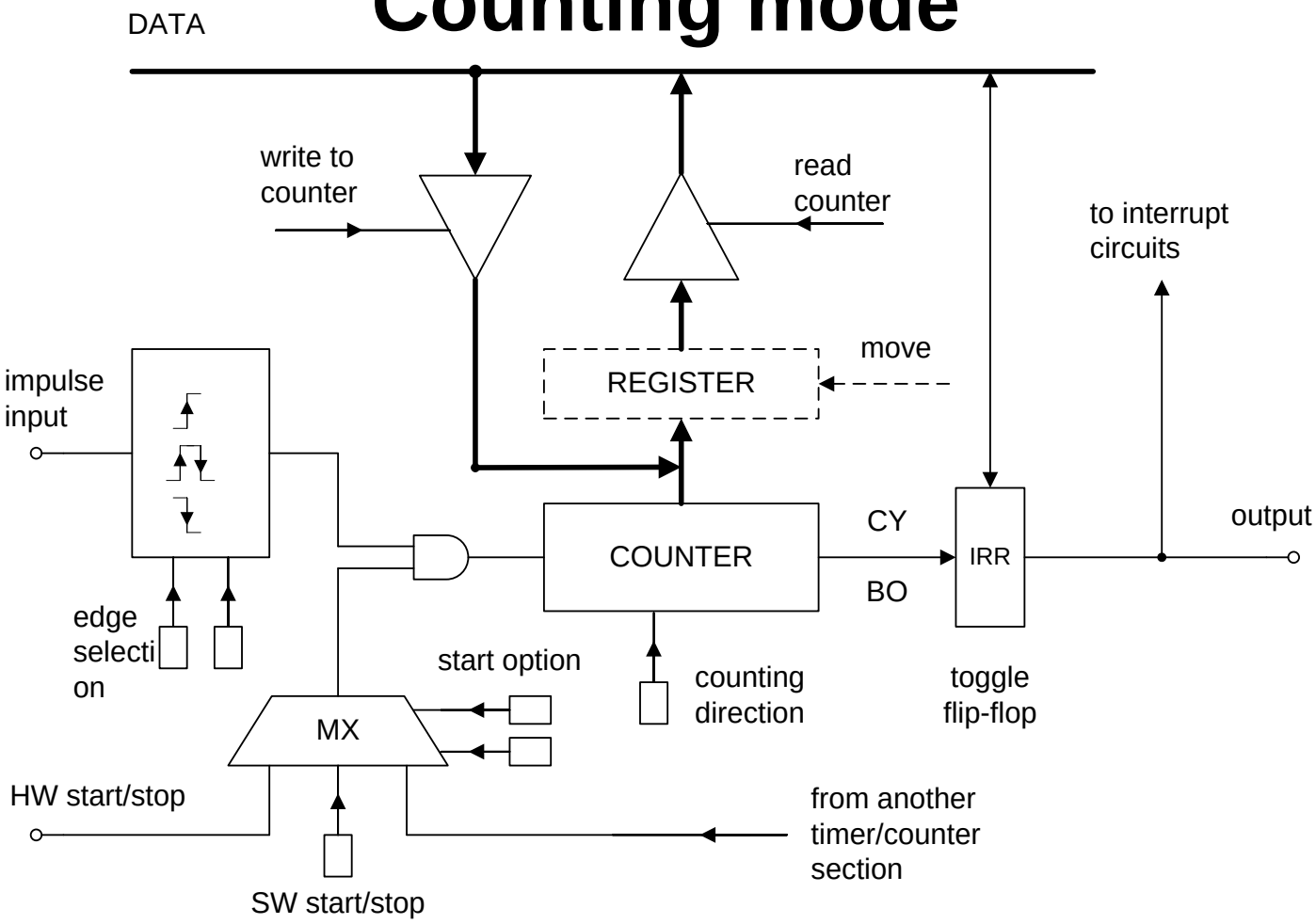
- Important and necessary for real-time control applications
- Time functions generation, precision timing signals, external events counting, program synchronization on external events …

- **Multi-chip solution** – HW configuration flexibility
- **Single-chip microcomputer** – internal structure configurable through SFRs

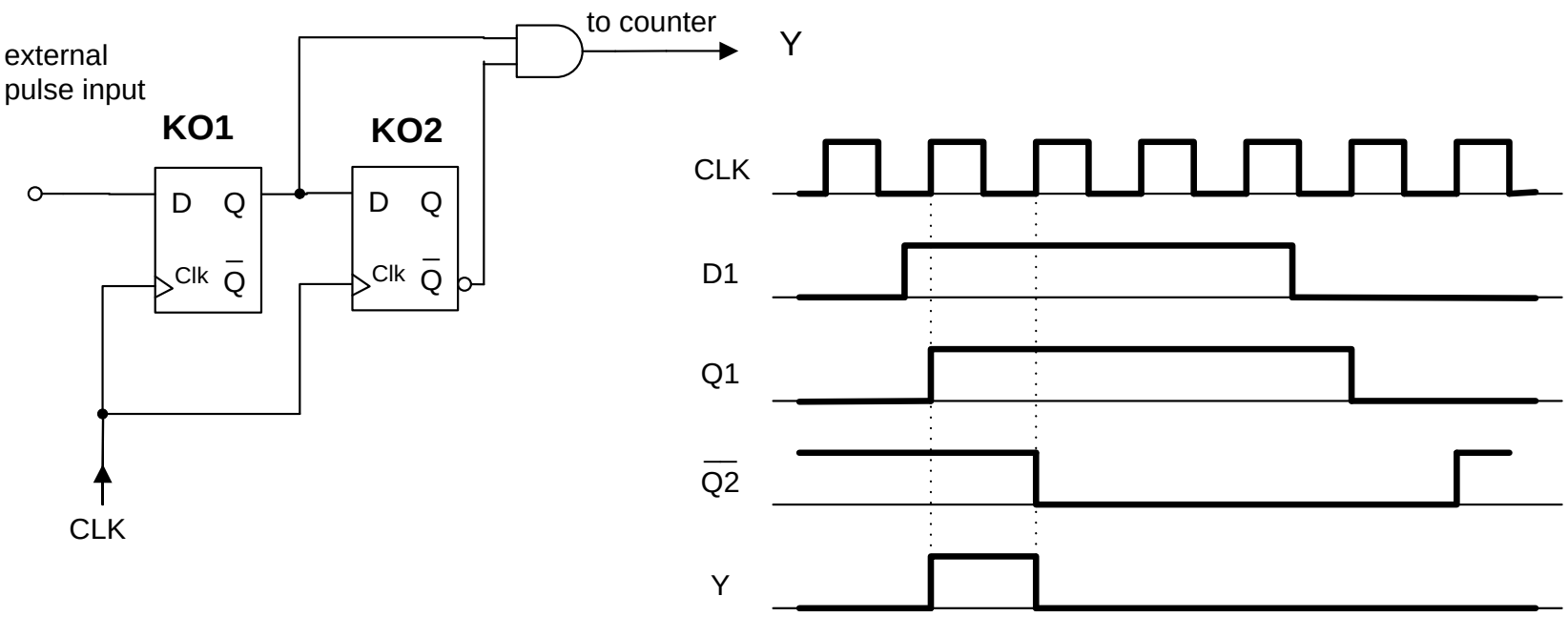
- Usually 16-bit
- More timers-counters aggregated into the one configurable system
- Connection to the internal interrupt system of microprocessor

Generally purposed timer-counter (GPT)

Counting mode

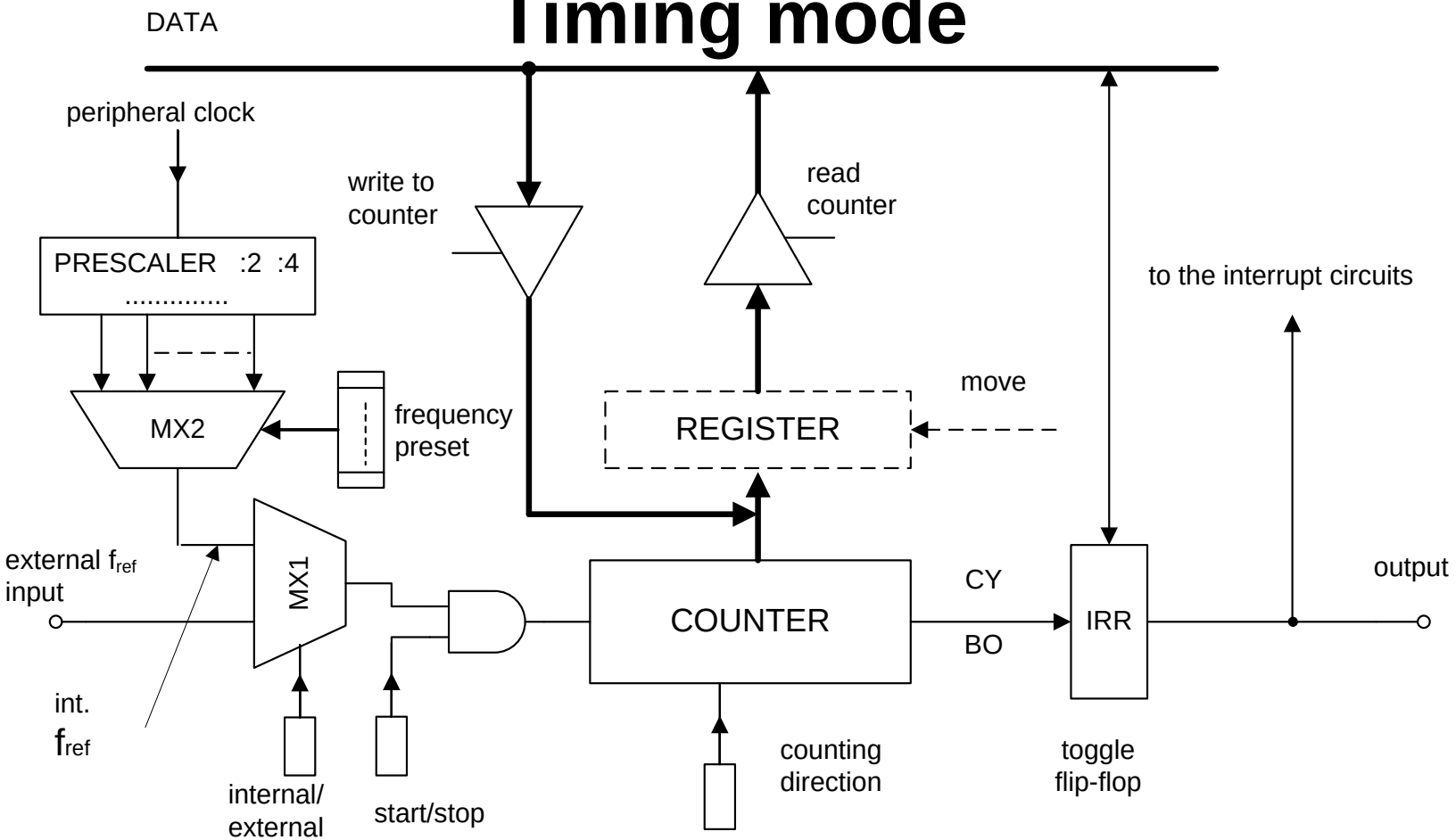


Synchronous edge detector

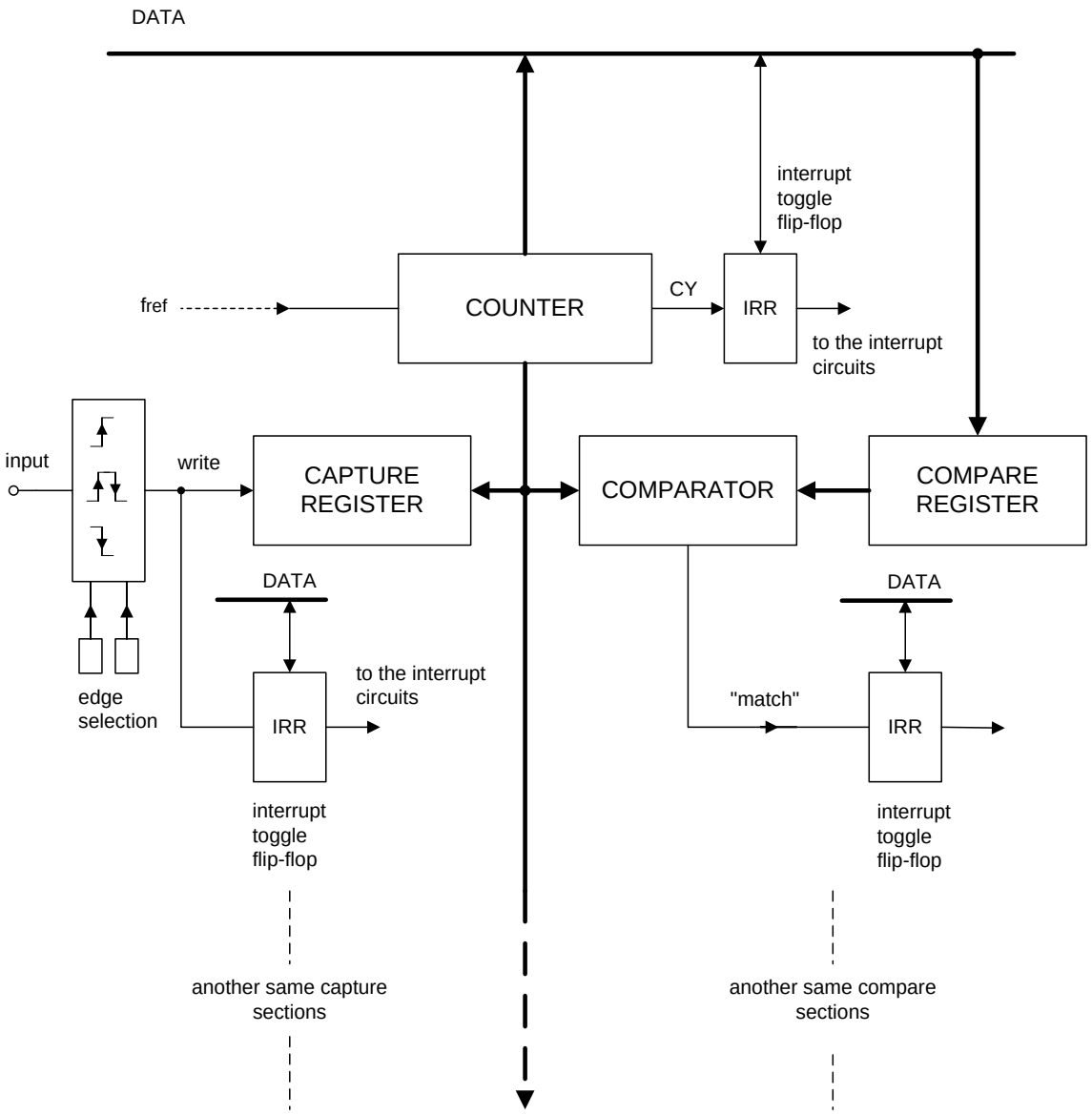


Generally purposed timer-counter (GPT)

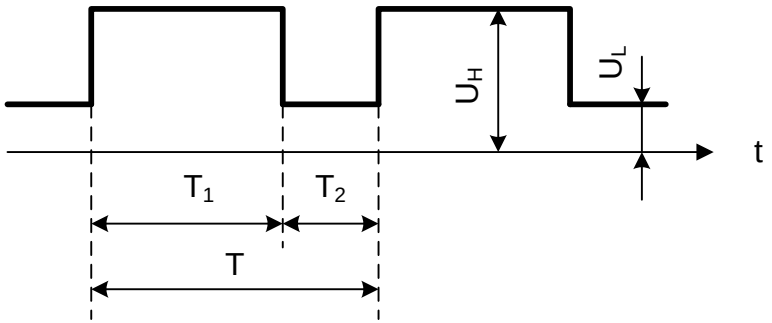
Timing mode



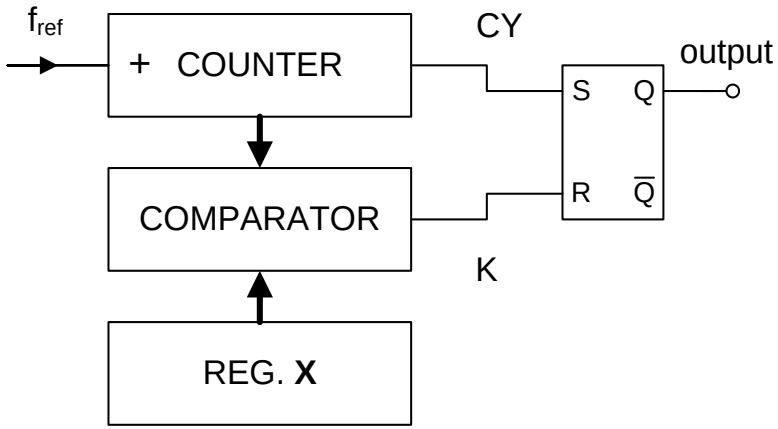
Capture – Compare Unit

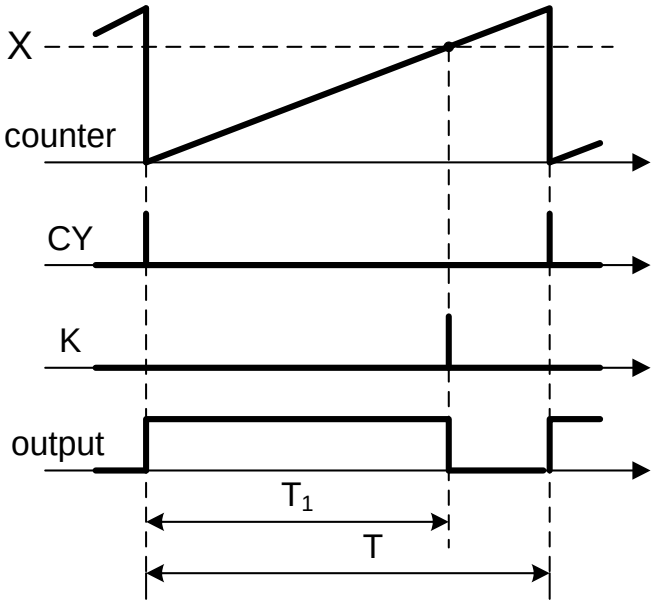


PWM modulation

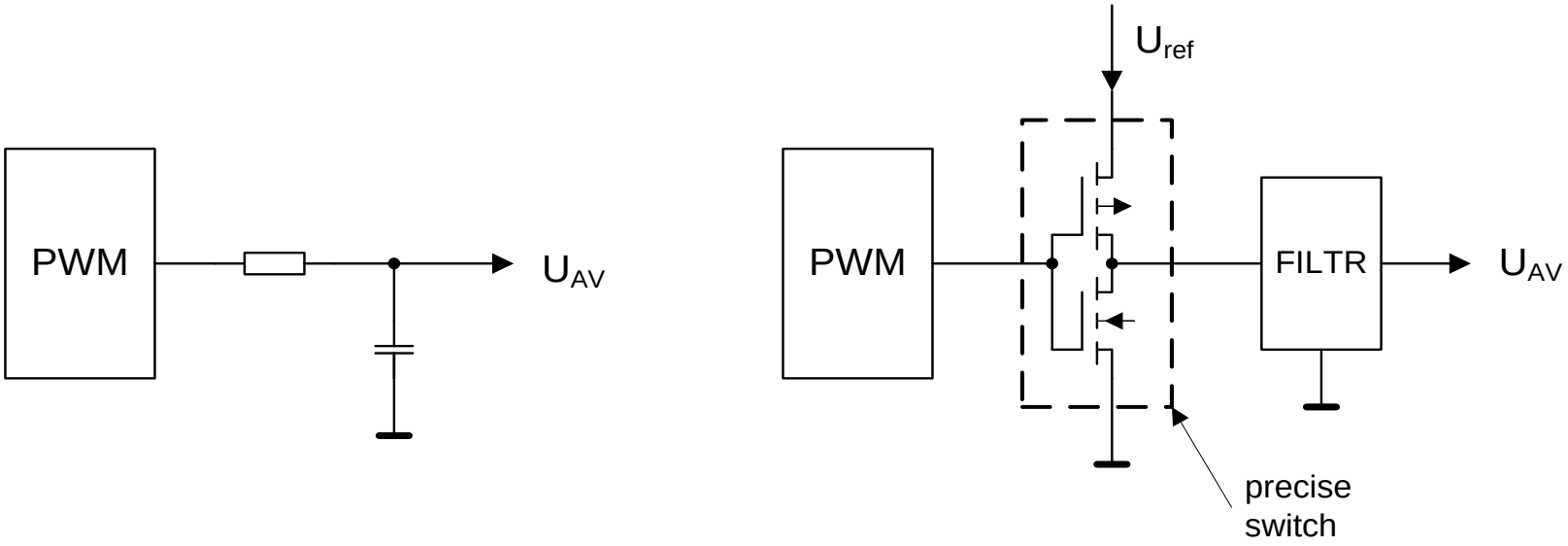


$$U_{STR} = \frac{T_1}{T}(U_H - U_L) + U_L$$

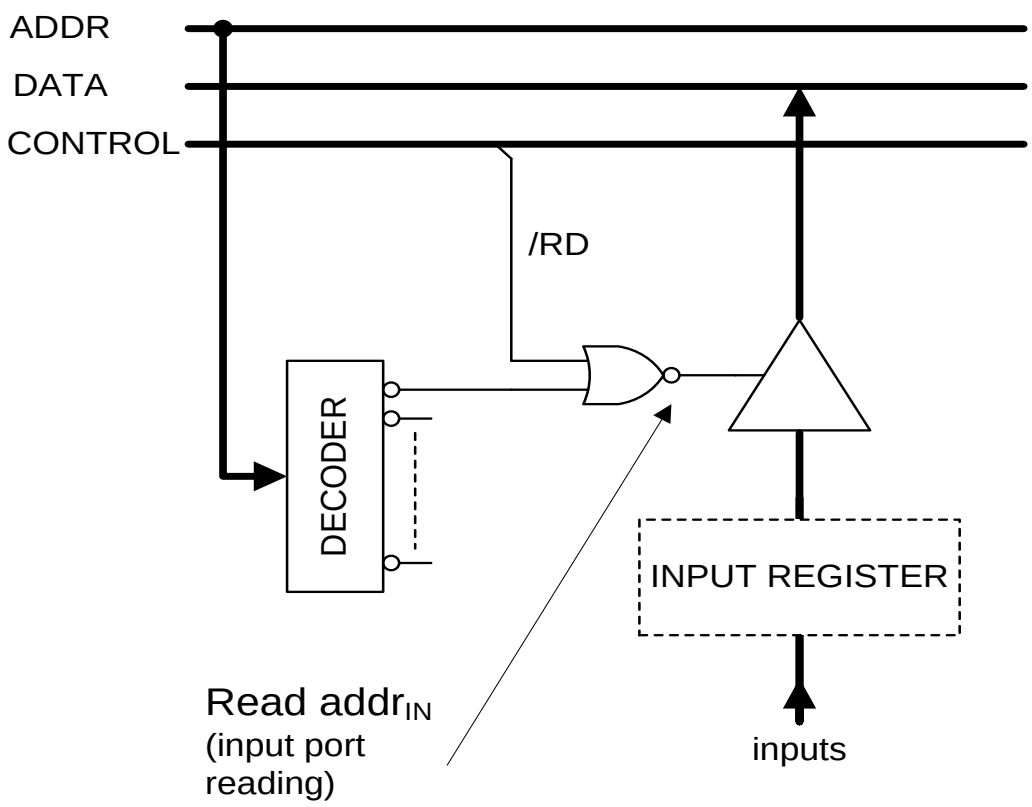




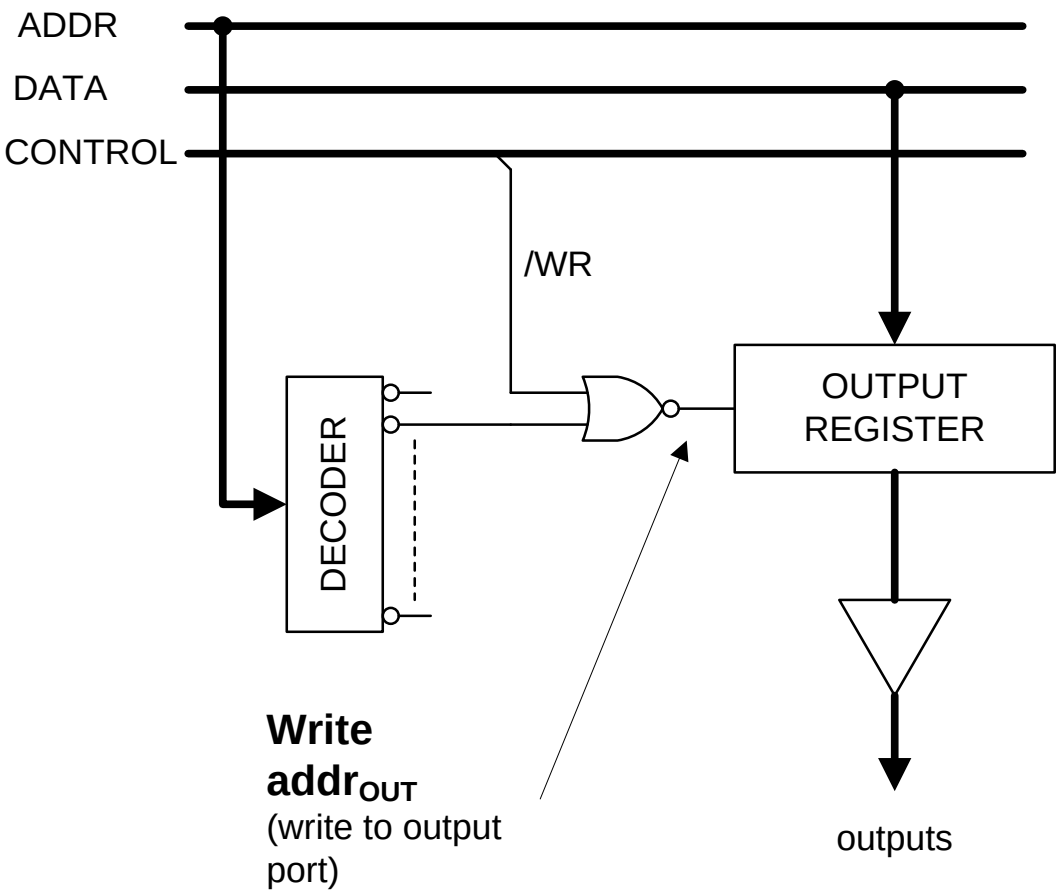
PWM demodulation



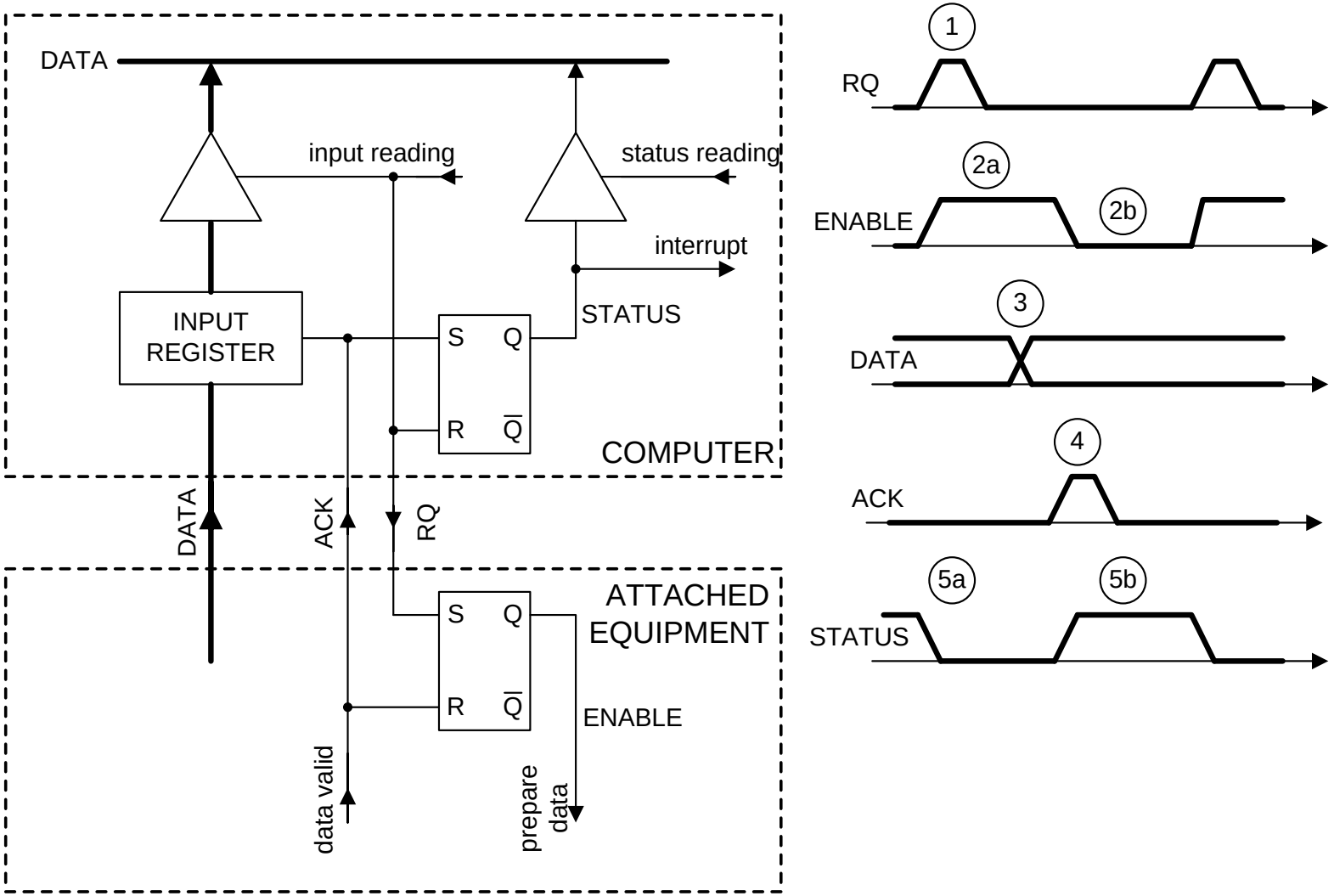
Parallel input circuit



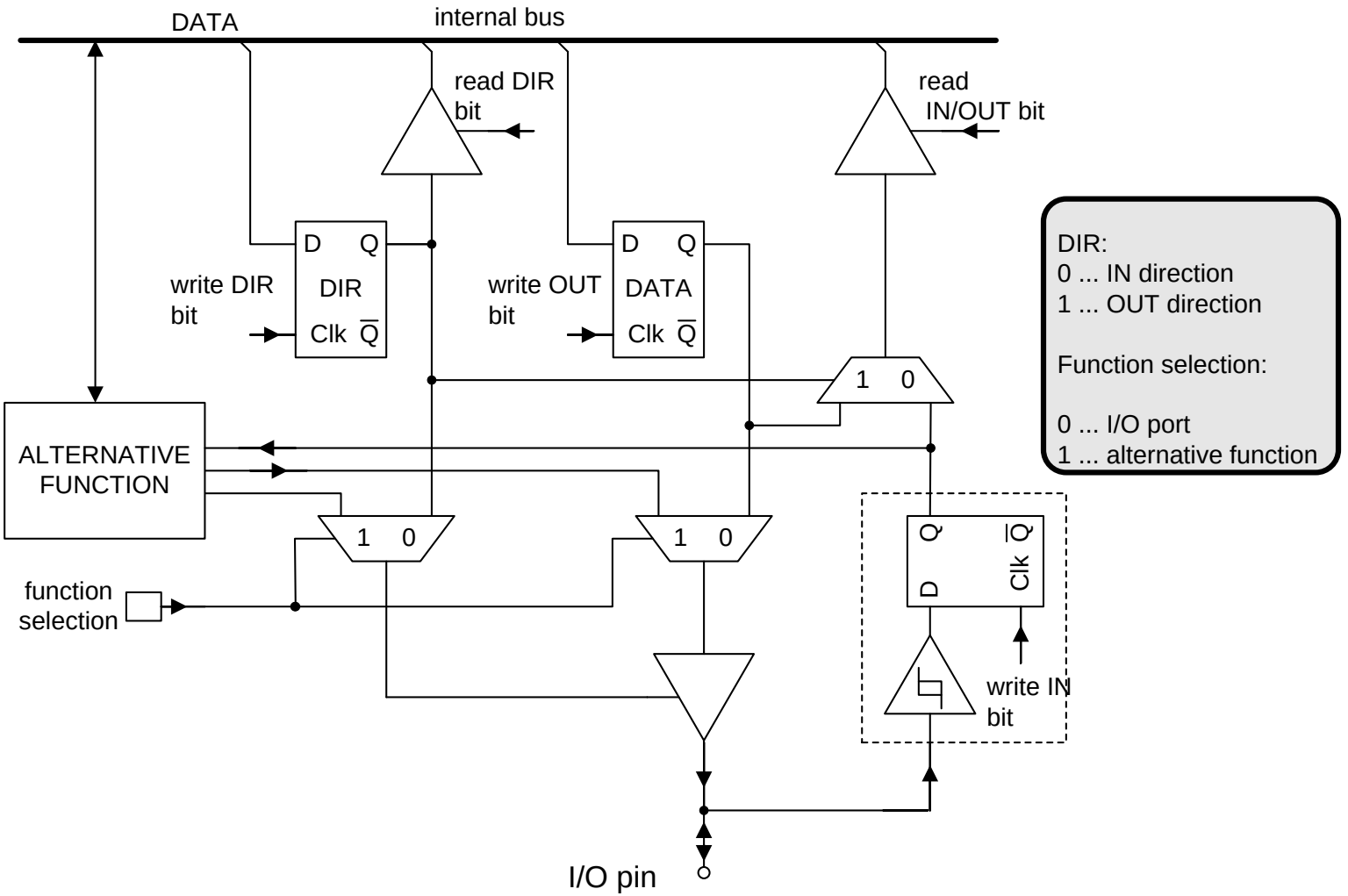
Parallel output circuit



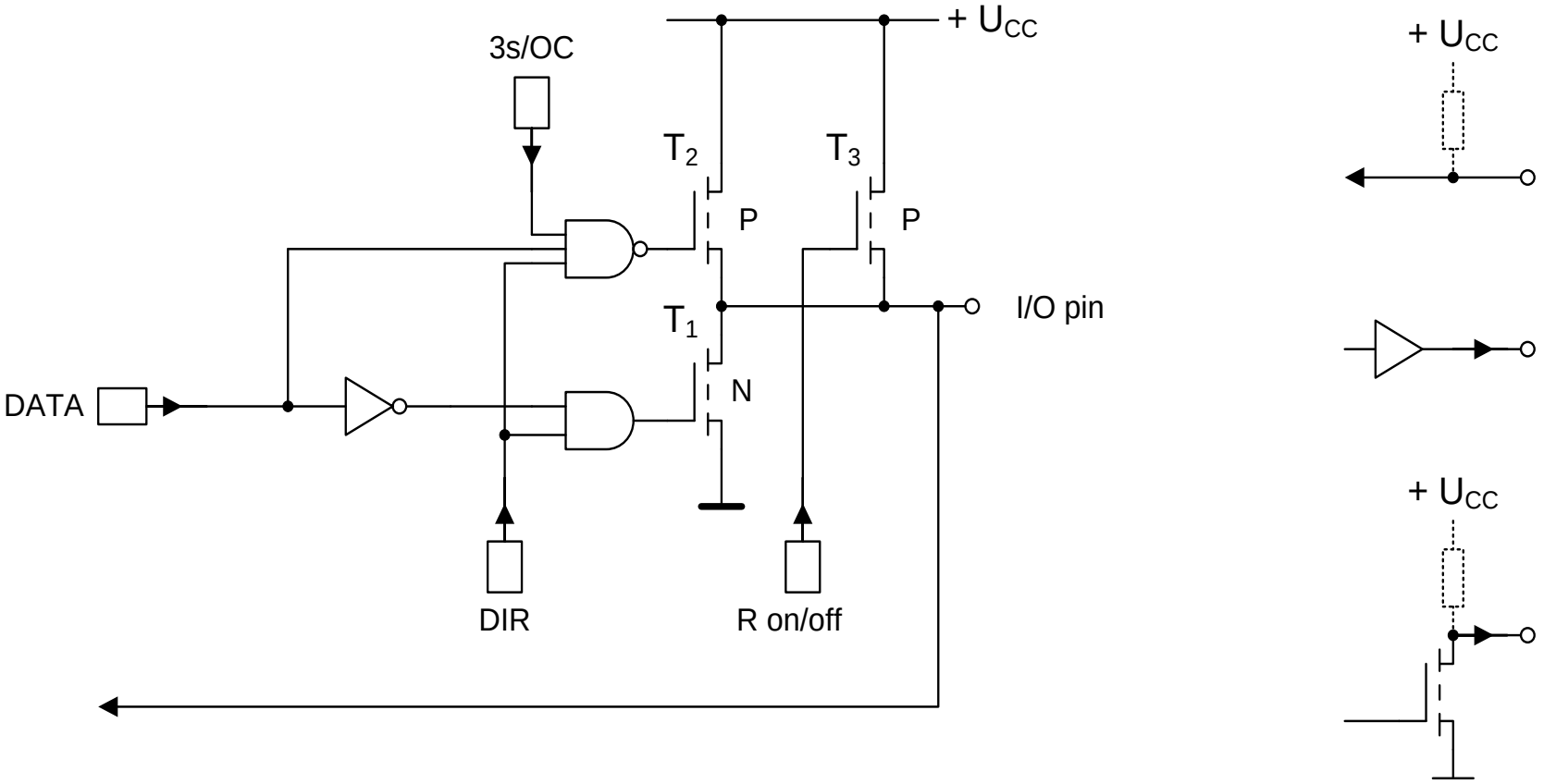
Handshake


DATA
input reading
status reading
interrupt
INPUT
REGISTER
S
Q
R
Q
STATUS
COMPUTER
DATA
ACK
RQ
ATTACHED
EQUIPMENT
S
Q
R
Q
ENABLE
data valid
prepare
data
1
RQ
2a
2b
ENABLE
3
DATA
4
ACK
5a
5b
STATUS

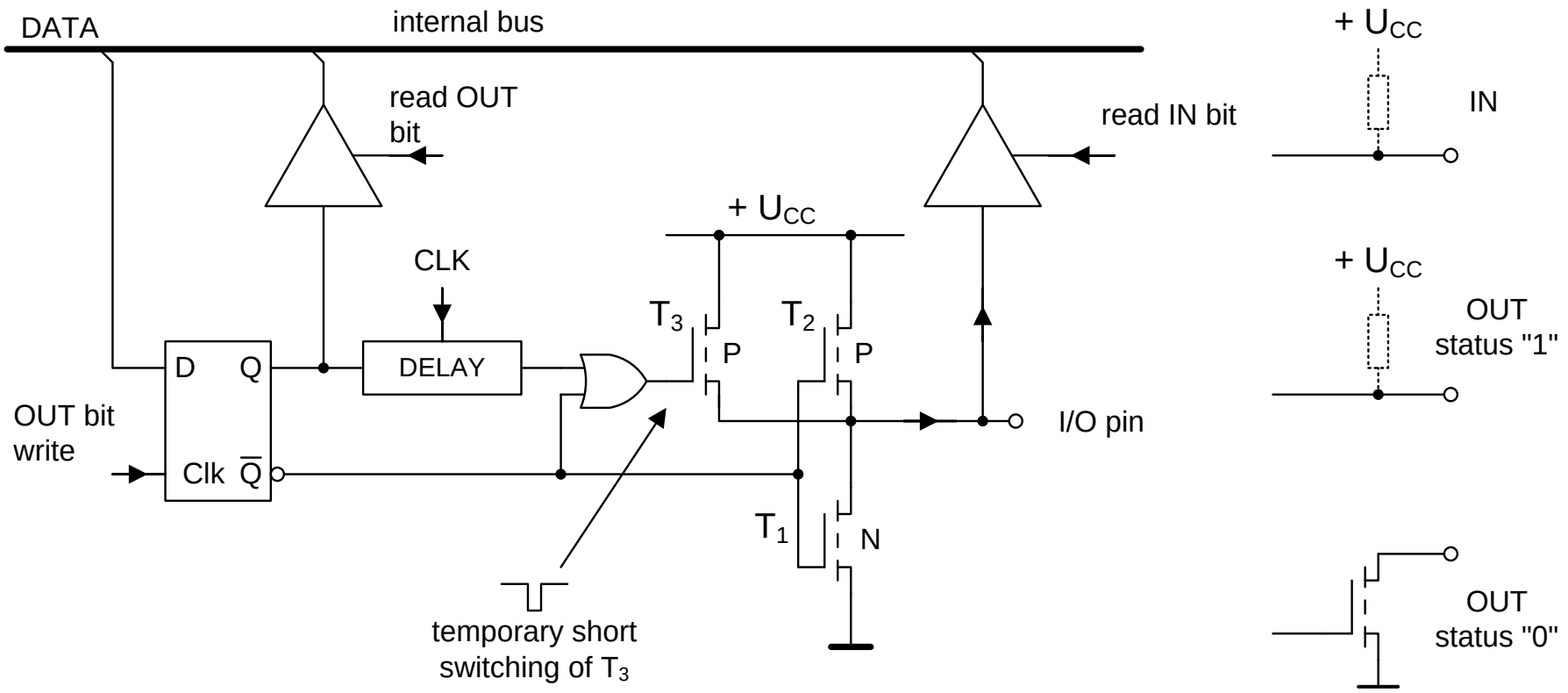
Bidirectional I/O port – 1bit



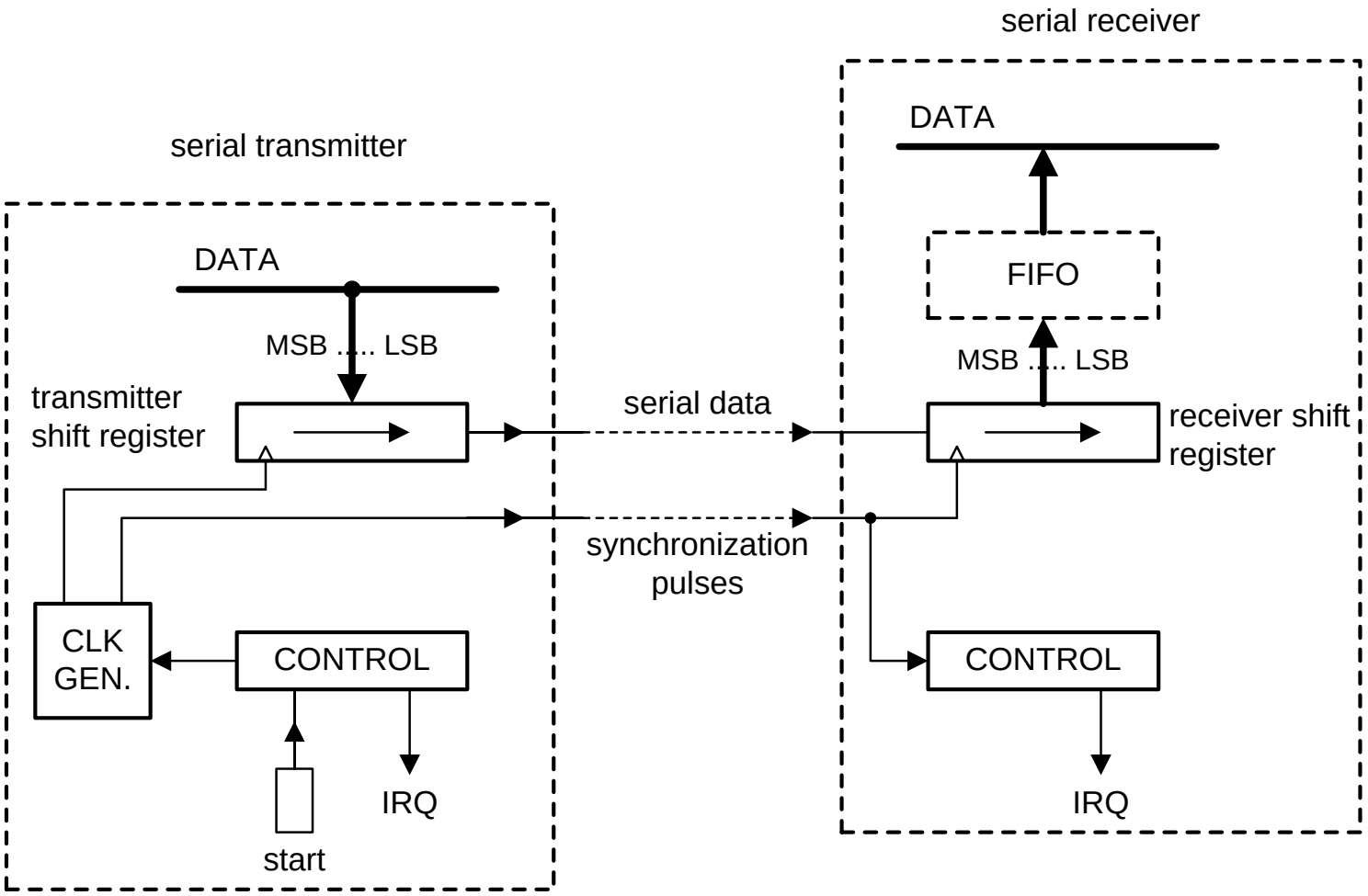
Output port circuit



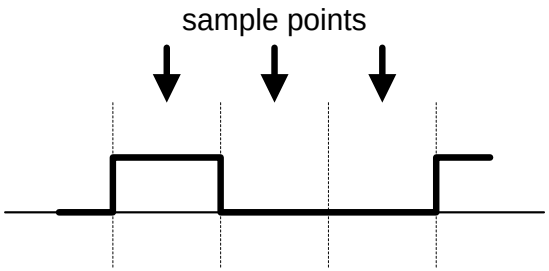
Quasi-bidirectional port

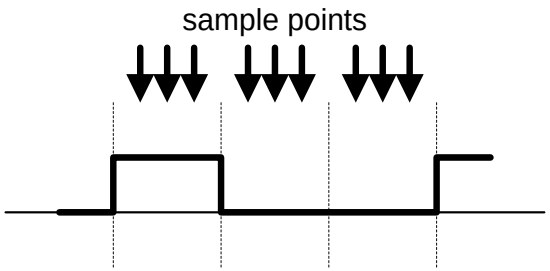


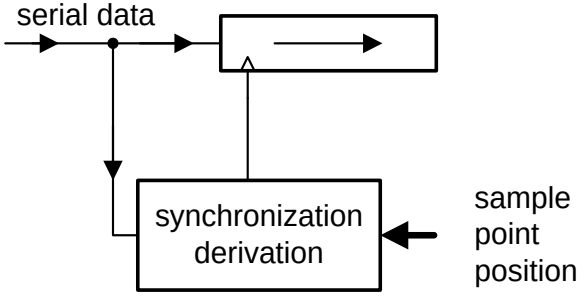
Serial communication

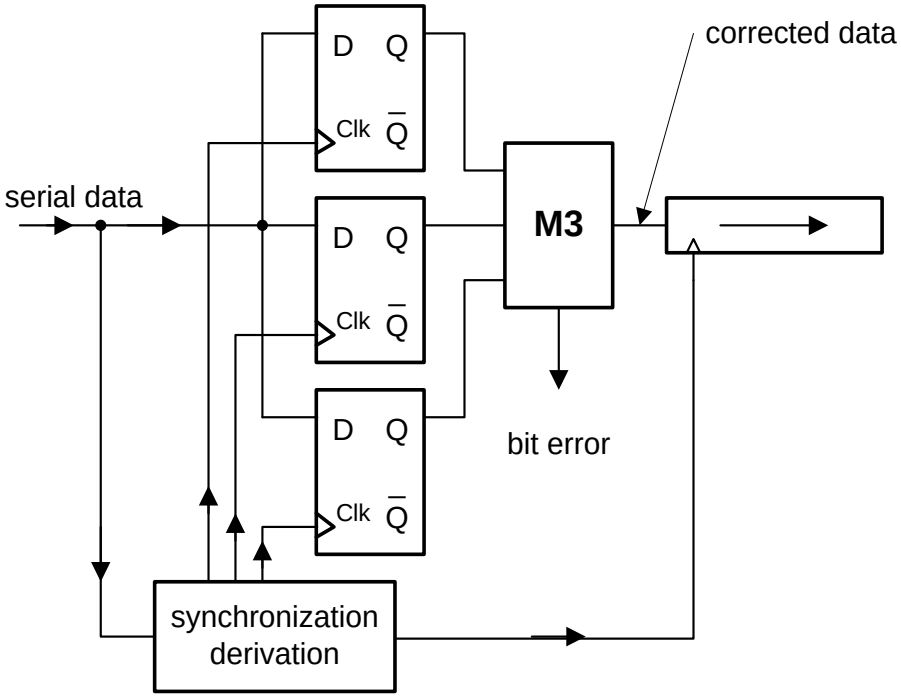


Serial data sampling

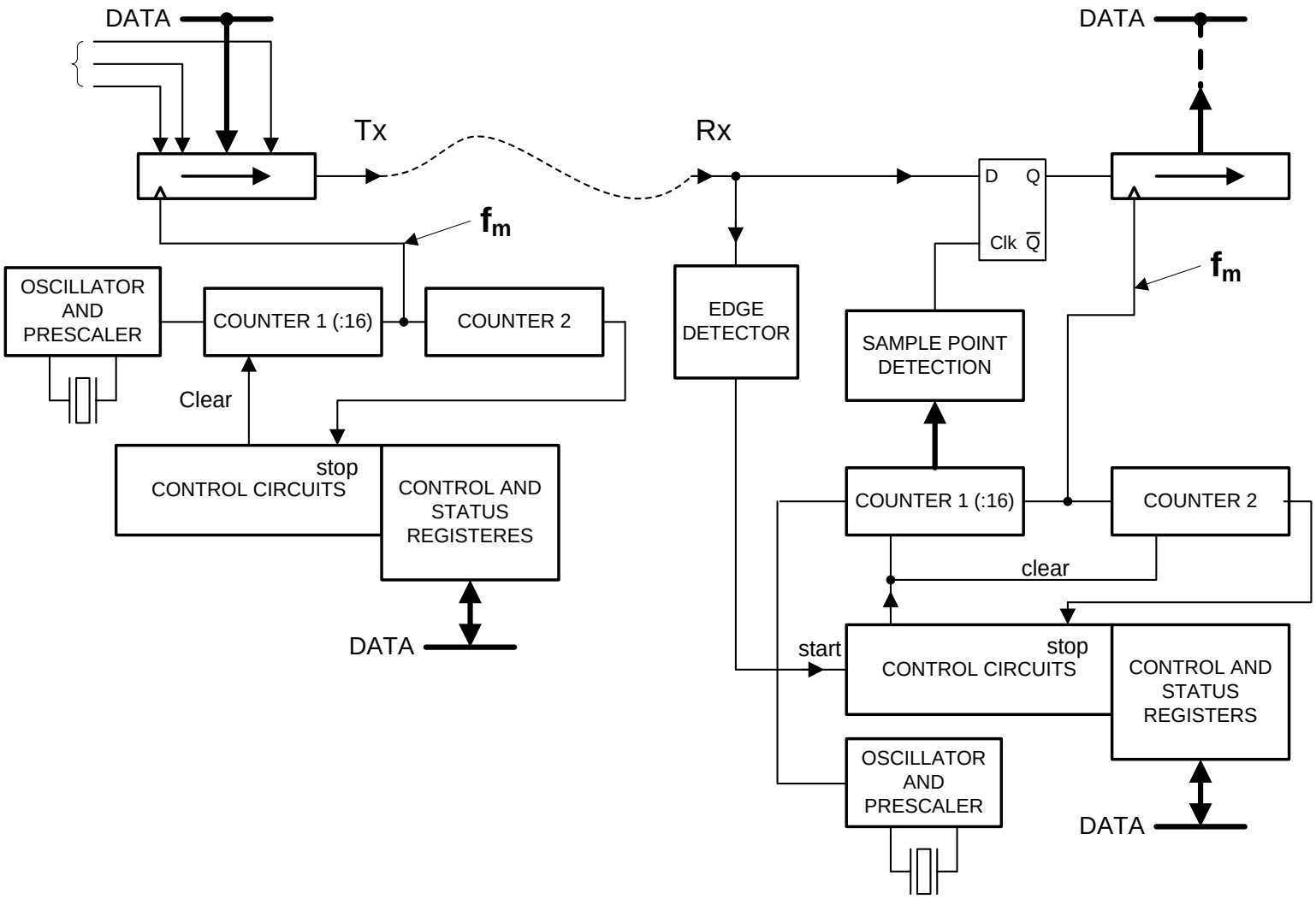






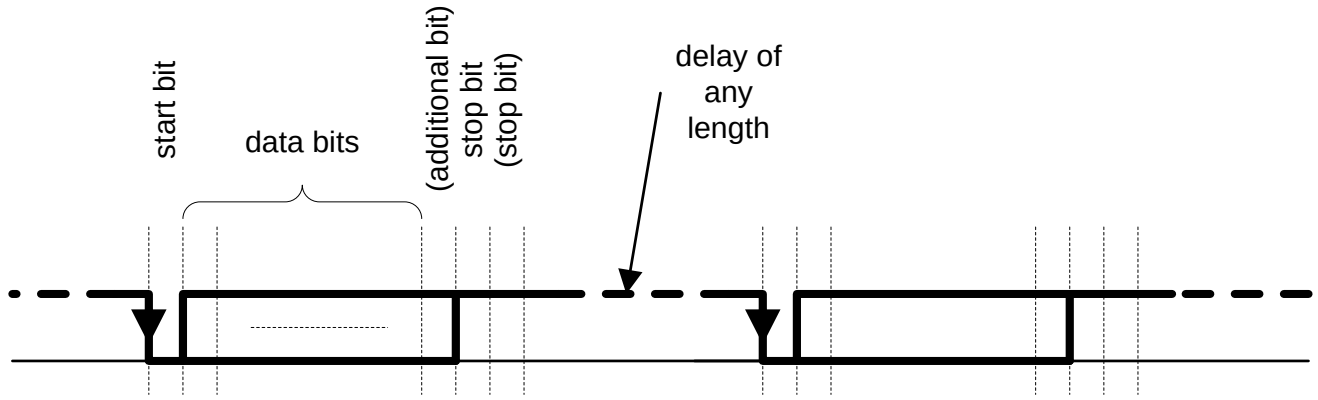


Character synchronization



UART

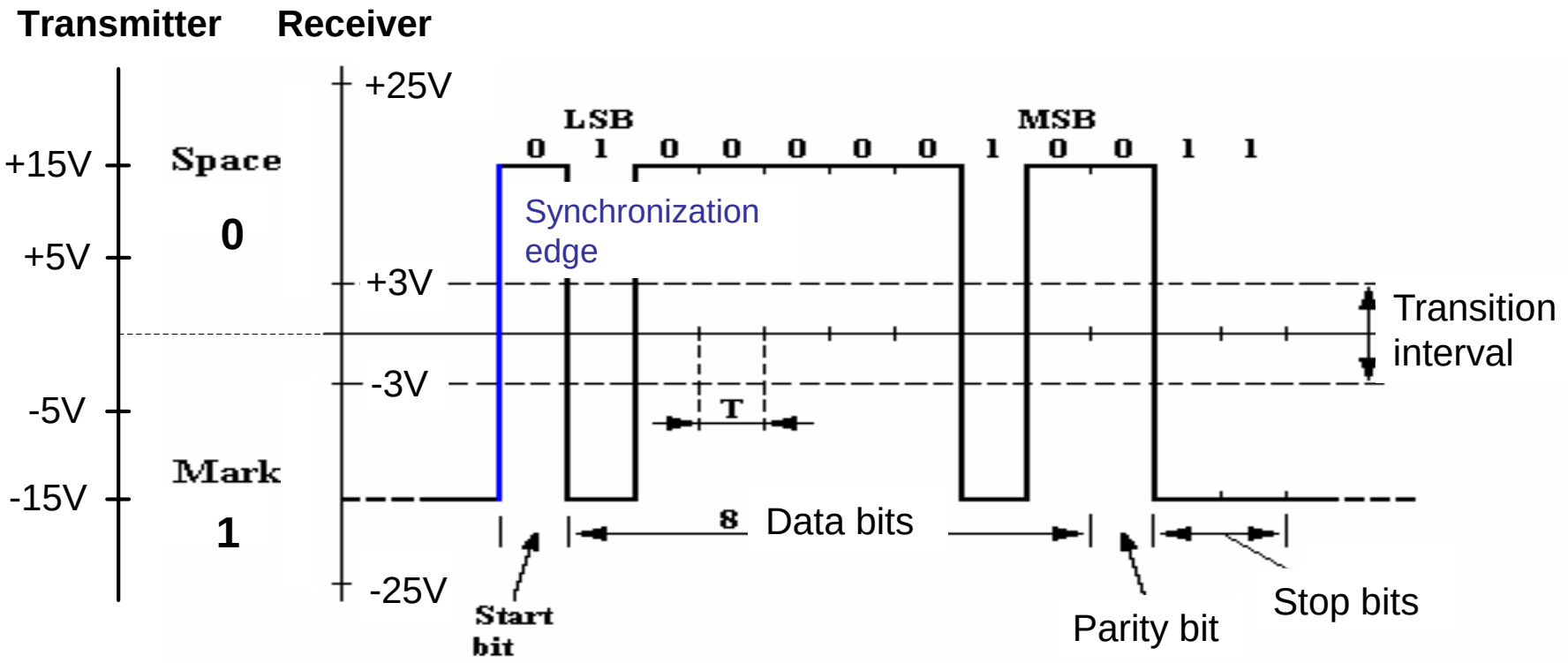
Universal Asynchronous Receiver - Transmitter



Number of data bits optional (typically 7 - 9) – LSB first
Parity bit – peripheral dependent
Number of stop bits optional (1 - 2)

Error detection	**- bit error**	**samples mismatch**
	- parity error	
	- frame error	**stop bit equals 0**
	- dat lost	**overrun error**

RS232


Transmitter
Receiver
+25V
+15V
Space
0
+5V
+3V
-3V
-5V
Mark
1
-15V
-25V
LSB
MSB
0 1 0 0 0 0 0 1 0 0 1 1
Synchronization edge
Transition interval
T
8 Data bits
Start bit
Parity bit
Stop bits

RS232 - features

- point to point duplex asynchronous communication
- asymmetric line
- negative logic
- input and output impedance of the transmitter and receiver in the range 3 ÷7 kΩ >> terminating line impedance
- Slew rate of the signal 30V/μs at maximum
- Baud rate determined by maximal load capacitance of the line 2 500 pF (166,7 pF/m):

  19,2 kbit/s at 15 m length

  115,2 kbit/s at 3 m length

Thank for your attention…