

Computer Architecture License: <https://creativecommons.org/licenses/by-nc-nd/4.0/>

3 Input Output (I/O) Organization and Bus Operations

This organization manages the data transfer between inside world (registers, memory) and outside world (keyboard, mouse, hard disk, network interface).

3.1 CPU - I/O Interface Connections

A) Isolated I/O Map: Two separate address spaces for memory and I/O

- Two separate busses (address and data), one for memory and the other for I/O can be used.
- A common bus can be used both for memory and I/O units.

A control line express the access type of the CPU: memory or I/O.

Example:
Intel x86 processors

Valid address, address strobe, R/W' clock etc.

Memory and I/O address ranges can be of different sizes.

There are distinct instructions for memory and I/O operations. Memory: LOAD, STORE, I/O: IN, OUT

www.faculty.itu.edu.tr/buzluca
www.buzluca.info 2013-2018 Feza BUZLUCA 3.1

Computer Architecture

B) Memory-Mapped I/O:

Same address space is shared by memory and I/O

- Common address and data busses both for memory and I/O units with common control lines
- Same instructions are used both for memory and I/O operations. MOVE, LOAD, STORE

Example:
Motorola 68XX, 68K processors

www.faculty.itu.edu.tr/buzluca
www.buzluca.info 2013-2018 Feza BUZLUCA 3.2

Computer Architecture

Separate Busses for Memory and I/O units:

Common bus for Memory and I/O:

www.faculty.itu.edu.tr/buzluca
www.buzluca.info 2013-2018 Feza BUZLUCA 3.3

Computer Architecture

3.2 I/O Interface Module

Peripherals are connected to CPUs over an I/O interface unit.

Functions of an interface:

- The data transfer rates of peripherals and the CPU are different. Checking the status of the peripheral, data buffering
- Data conversion: Coding, encoding, different formats
- Error detection
- Signal conversion: Magnetic, electromechanic, electronic

Example I/O Interface:

www.faculty.itu.edu.tr/buzluca
www.buzluca.info 2013-2018 Feza BUZLUCA 3.4

Computer Architecture

3.3 Data Transfer Modes between I/O Interfaces and Memory

1. Programmed I/O (software polling):

It is the responsibility of the processor

- periodically to check the status of the I/O interface (ready/busy, complete)
- to perform the data transfer between memory (registers) and I/O interface.

Read from the I/O Interface:
The CPU runs a program to check the status of the I/O interface.

If the I/O interface has received data from a peripheral it sets the "READY" flag.

The CPU reads data from the I/O interface and writes to the memory.

```

graph TD
    ReadStatus[Read status] --> DataReady{Data Ready?}
    DataReady -- No --> ReadStatus
    DataReady -- Yes --> ReadData[Read data from I/O interface]
    ReadData --> WriteMem[Write data to memory]
  
```

Write to the I/O Interface:
If the CPU has data to send, it checks the status of the I/O interface.

If the I/O interface has finished the previous operation it sets the "COMPLETE" flag.

The CPU reads data from the memory and writes it to the I/O interface.

```

graph TD
    ReadStatus2[Read status] --> Complete{Complete?}
    Complete -- No --> ReadStatus2
    Complete -- Yes --> ReadData2[Read data from memory]
    ReadData2 --> WriteIO[Write data to I/O Interface]
  
```

www.faculty.itu.edu.tr/buzluca
www.buzluca.info 2013-2018 Feza BUZLUCA 3.5

Computer Architecture

1. Programmed I/O cont'd:

Disadvantage:
The main disadvantage of this technique is the **busy-waiting** of the CPU while checking the status of the I/O units.
During checking the status, the CPU cannot run other programs.
Data transfer is also performed by the CPU (The data flows over the CPU).

Advantage:
This technique is **simple**. Additional hardware units are not necessary.
When the CPU does not have any tasks other than performing I/O operations or if the CPU cannot execute another program without performing the I/O operation, then busy-waiting is not a problem.
For such systems programmed I/O is a simple and suitable technique for I/O operations.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info 2013-2018 Feza BUZLUCA 3.6

Computer Architecture License: <https://creativecommons.org/licenses/by-nc-nd/4.0/>

2. Interrupt-Driven I/O:

In the interrupt-driven technique the CPU sets the I/O interface to send an interrupt request if it is ready.

Advantage: The CPU does not need to check the status continuously. The "busy-waiting" problem does not exist.

The CPU can run other programs while the I/O interface is receiving data from or sending to a peripheral.

The I/O interface will then interrupt the processor to request service when it is ready to exchange data with the CPU.

The processor interrupts its current program, runs the interrupt service routine in which the data transfer is executed, and then resumes its former processing.

In this technique the CPU does not check the status but it is still the responsibility of the processor to perform the data transfer.

Disadvantage: Interrupt processing has its own overhead (saving return address, program status, registers and performing some other operations). This operations are performed in the interrupt cycle for each interrupt; and while returning return-address and program status are read.

Interrupt-driven I/O is not suitable for applications where I/O operations are performed very frequently.

Interrupts are explained in Section 4.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.7

Computer Architecture

3. Direct Memory Access (DMA):

In the programmed and interrupt-driven techniques the CPU is responsible for transferring data between memory and I/O interfaces.

The CPU must execute a number of instructions for each I/O transfer.

The direct memory access (DMA) technique involves an additional hardware module on the system bus called the DMA controller (DMAC).

The DMAC is capable of acting as the CPU and of taking over control of the system bus from the processor.

When the CPU needs to read or write a block of data, it initializes the DMAC by sending the necessary information (address, size, transfer mode etc.).

So, it has delegated the I/O operation to the DMAC.

The CPU can continue with its other programs during the transfer of data.

The data do not flow over the CPU.

The DMAC uses the system bus only when the processor does not need it, or it must force the processor to suspend the bus operations temporarily.

The DMA technique is suitable for applications where large volumes of data are transferred and I/O operations are performed very frequently.

An additional hardware module (DMAC) is necessary.

DMA is explained in Section 5.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.8

Computer Architecture

Summary of Data Transfer Modes:

The table below shows which device is responsible for checking the status of the I/O Interface and transferring the data.

Task Method	Check the status of the I/O Interface	Data transfer between I/O Interface and memory
Programmed I/O:	CPU (Program)	CPU (Program)
Interrupt driven I/O:	Interrupt Mechanism	CPU (ISR)
DMA:	DMAC	DMAC

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.9

Computer Architecture

3.4 Asynchronous Data Transfer:

Problems:

- Has the sender sent the data (Is the data on the data bus valid)?
- Has the receiver received the data (Is the receiver busy)?

1. Strobe Control:

a) Source-initiated strobe :

The duration of valid data is predetermined according to the speed of the destination. The source does not know whether the destination has really received the data.

b) Destination-initiated strobe:

Time to transfer (sample) the data from the bus is predetermined according to the speed of the source. The destination does not know whether the source has really sent the data.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.10

Computer Architecture

2. Handshaking:

a) Source-initiated:

The source waits for the "Data accepted" signal. To avoid "infinite waiting" when the "Data accepted" signal does not respond due to an error, a **time-out** mechanism must be used.

b) Destination-initiated:

The destination waits until the "Data valid" signal is received. To avoid infinite waiting a **time-out** mechanism is necessary.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.11

Computer Architecture

3.5 Data Transfer between the CPU - Memory (or I/O Interface)

CPUs also use synchronous or asynchronous data transfer mechanisms to access the memory.

3.5.1 Synchronous bus operation with strobe

Read cycle of MC6802:

For example, MC 6802 uses **strobe mechanism** in memory access.

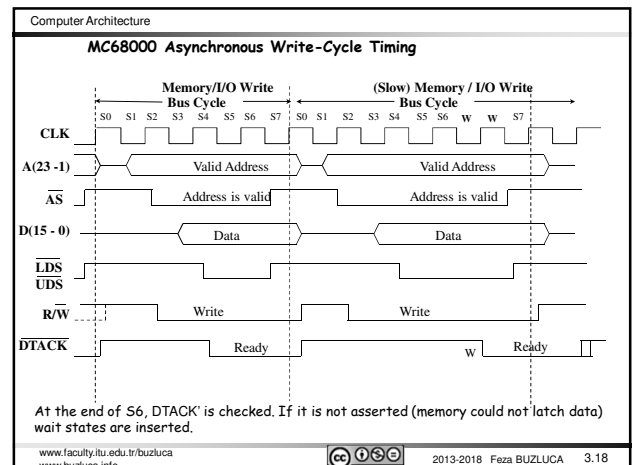
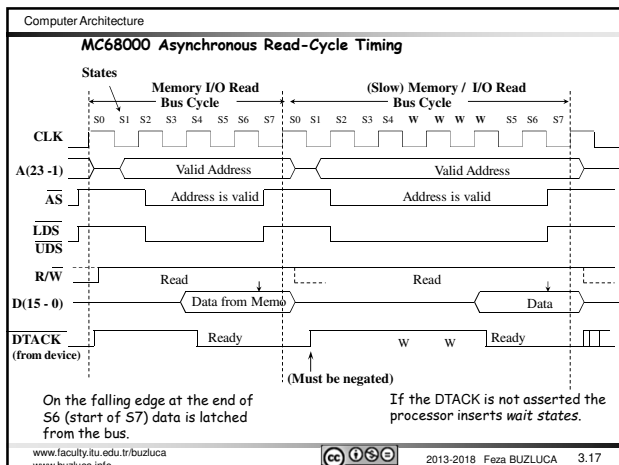
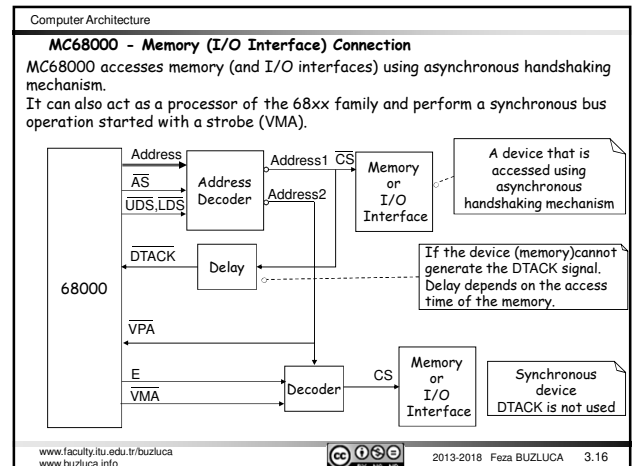
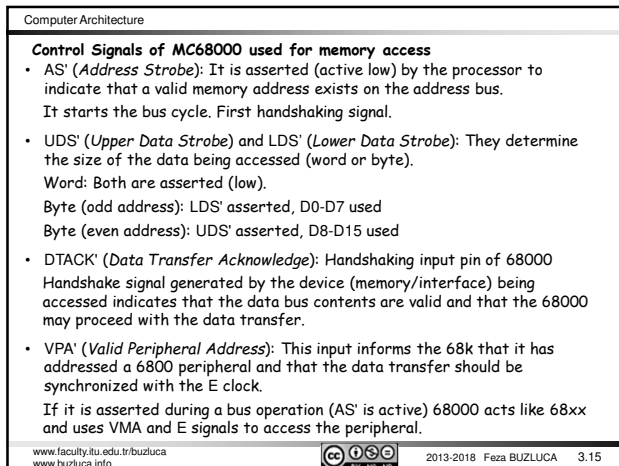
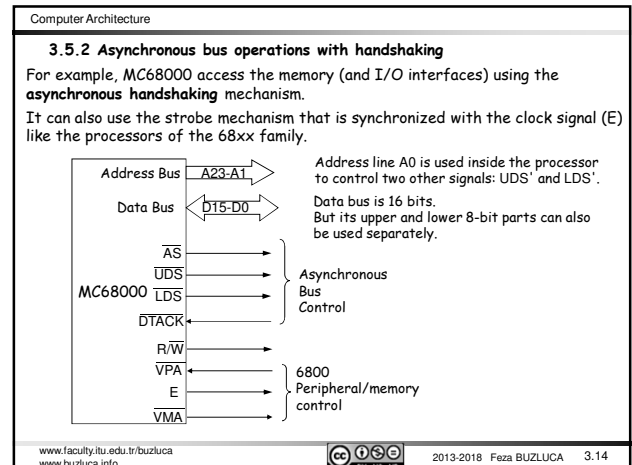
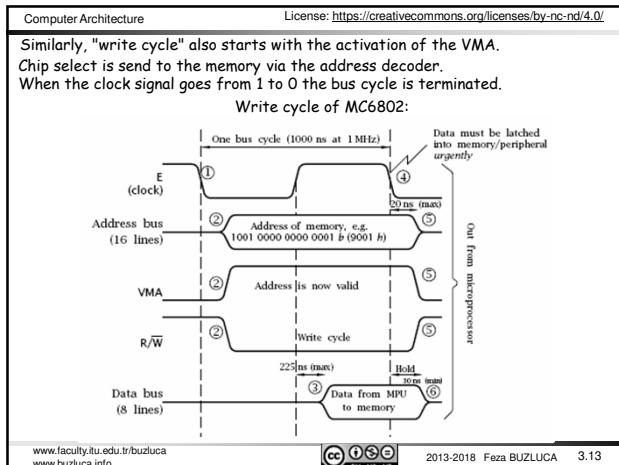
This operation is also **synchronized** with the clock signal (E) of the processor.

The Valid Memory Address (VMA) signal indicates that the address on the bus is valid and initiates the bus cycle (strobe).

When the clock signal goes from 1 to 0 data is latched and the bus cycle is terminated.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.12



Computer Architecture

Avoiding Infinite Waiting

MC68000 has an exception input called **BERR'** (Bus Error) that can be asserted by an external logic if an error in the current bus cycle is detected.

If this input is asserted (active 0) the 68000 terminates the current bus cycle, saves the current status into the stack (accessed address, current instruction etc.), and jumps to an exception handler program.

BERR' will be explained in the chapter "Exceptions".

To avoid infinite waiting a counter can be connected to the BERR' as shown below:

If the bus takes (AS' is active) longer than expected, BERR' is asserted.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.19

Computer Architecture

8/16-bit Bus operations in MC68000 (access to odd/even addresses)

In MC68000, the width of the data bus is 16 bits. Instructions can operate on 8, 16 or 32-bit operands (bus operations). Therefore, 8-bit memories are connected to 16-bit data bus in parallel.

To specify which memory is being accessed the MC68000 has two outputs: **UDS'** (Upper Data Strobe) and **LDS'** (Lower Data Strobe). The value of internal address line A0 determines the values of UDS' and LDS'.

UDS'	LDS'	D15-D8	D7-D0	Description:
H	H	---	---	No bus operation
H	L	---	Data	Byte access to an odd address A0=1
L	H	Data	---	Byte access to an even address A0=0
L	L	Data	Data	Word access to an even address A0=0

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.20

Computer Architecture

Example:

	Address: (A ₀ is in the processor)	UDS	LDS
MOVE.B (\$000000),D1	A ₂₃ A ₂₂ A ₁ A ₀	0 0	0 1
MOVE.B (\$000001),D1	0 0 0 1	1 0	1 0
MOVE.W (\$000000),D1	0 0 0 0	0 0	0 0
MOVE.W (\$000001),D1	Exception: Address error		

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.21

Computer Architecture

Example:

In the example on the right following units are connected to the MC68000:

- One serial communication interface (MC6850), a total of 4Kx8 ROM (2x 2716) and total of 4Kx8 SRAM (2x 4016).
- Only two address lines A13 and A12 are used for address decoding.
- The serial communication interface (MC6850) is connected as a synchronous unit (VMA and E are used).
- Note: It communicates with its peer interface using the asynchronous mechanism.
- Between the CPU and memories data transfer occurs according to the asynchronous handshaking mechanism (AS and DTACK are used).
- Since ROMs are not written to, UDS/LDS lines are not used to select these units.

3.22

Computer Architecture

Function Code Outputs in MC68000

MC68000 has 3 outputs that indicate the type of the operations:

Function Codes Outputs: FC2, FC1, FC0.

These outputs get valid values in each bus cycle (when AS' is asserted) and indicate the type of the operation.

FC2	FC1	FC0	Description:
0	0	0	Undefined (Reserved)
0	0	1	User Mode, Data access (User Data)
0	1	0	User Mode, Program access (User Program)
0	1	1	Undefined (Reserved)
1	0	0	Undefined (Reserved)
1	0	1	Supervisor Mode, Data access (Supervisor Data)
1	1	0	Supervisor Mode, Program access (Supervisor Program)
1	1	1	Interrupt Acknowledge

These outputs can be used in address decoding.

- Access to specific devices and memory addresses can be restricted. These addresses can be accessed only in supervisor mode.
- Separate memory spaces can be created for programs and data.

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.23

Computer Architecture

Example:

Placing separate program and data memories to the same address space

In this example, by using the FC0 output of the MC68000 two separate memory modules are placed into the same address space.

One module is selected when FC0 = 0 (program access). The second one is selected if FC0 = 1 (data access).

www.faculty.itu.edu.tr/buzluca
www.buzluca.info

2013-2018 Feza BUZLUCA 3.24