COEN-4720 Embedded Systems

Lecture 10
Memory

Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

Outline

- Registers
- Memory map
- Memory protection unit (MPU)
- Direct memory access (DMA)
- Program memory model
- Memories – basic concepts

Cortex-M Processors

- Cortex-M processors use a load/store architecture with three basic types of instructions
  1. **Register-to-register** operations for processing data
  2. **Memory operations** which move data between memory and registers
  3. **Control flow** operations enabling programming language control flow such as if and while statements and procedure calls

3

Processor "Register Set"

- 16 user-visible registers
  - R0 to R15
  - All processing takes place in these registers
- Three of these registers have dedicated functions
  - R15 is the Program Counter (PC) - holds the address of the next instruction to execute
  - R14 is a register called Link Register (LR) - holds the address from which the current procedure was called
  - R13 is the Stack Pointer (SP) - holds the address of the current stack top

4

4

Registers

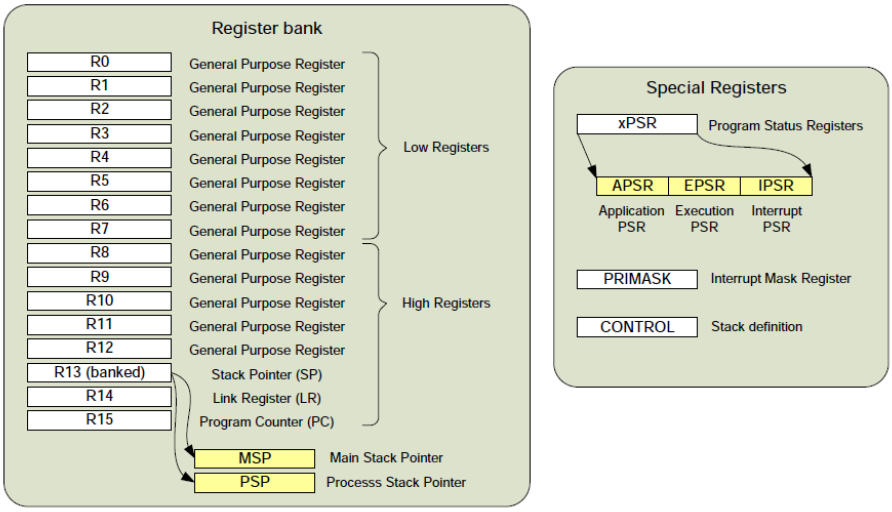


Figure 4.3
Registers in the Cortex®-M0 and Cortex-M0+ processors.

5

Cortex-M0+ Processor: Memory Addressing

- 32-bit addressing supporting up to 4 GB of memory space.
- The system bus interface is based on an on-chip bus protocol called (Advanced High-performance Bus) AHB-Lite, supporting 8-bit, 16-bit, and 32-bit data transfers.
- The AHB-Lite protocol is pipelined, support high operation frequency for the system.
- Peripherals can be connected to a simpler bus based on APB protocol (Advanced Peripheral Bus) via an AHB to APB bus bridge.
- Cortex-M0+ processor does not contain memories and peripherals (chip designers need to add these components to the MCU designs).

6

Example of MCU that uses Cortex-M0+ Processor

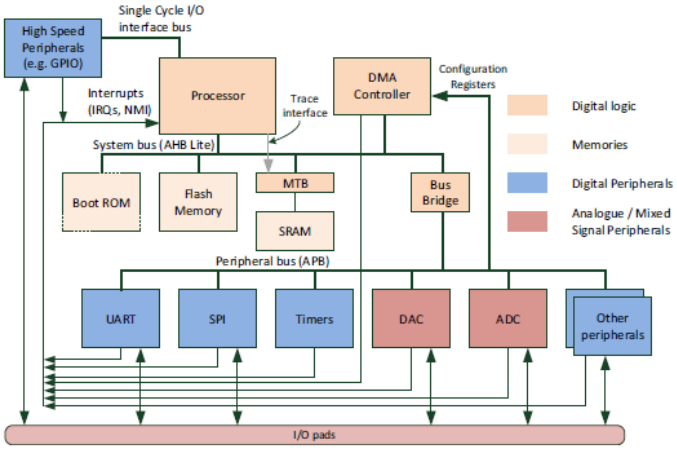


Figure 2.6
A system with the Cortex®-M0+ Processor and a DMA Controller.

Source: [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2). 7

7

Separation of main system bus and peripheral bus

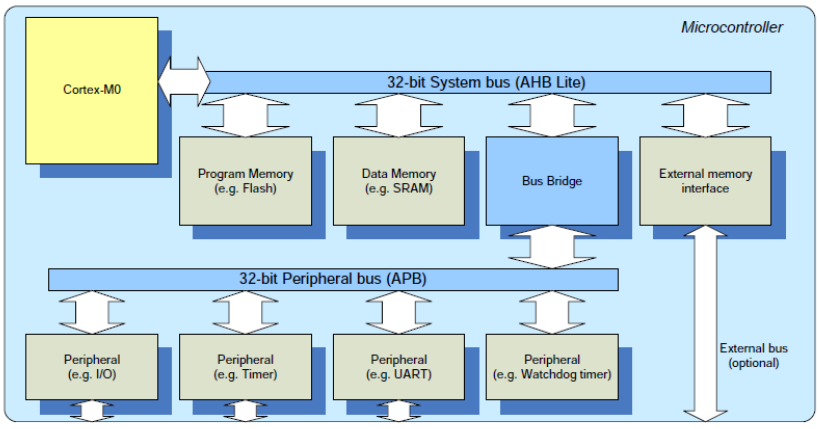


Figure 7.1
Separation of system and peripheral bus in a simple 32-bit microcontroller.

Source: [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2). 8

8

Cortex-M0+ Processor: Memory Map

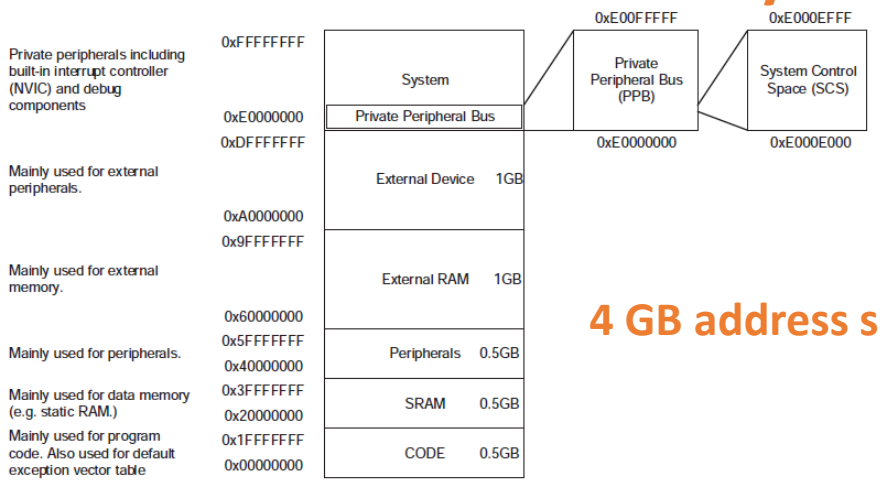


Figure 4.10
Memory map.

4 GB address space

Source: [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2).

9

Cortex-M0+ Processor: Memory Map

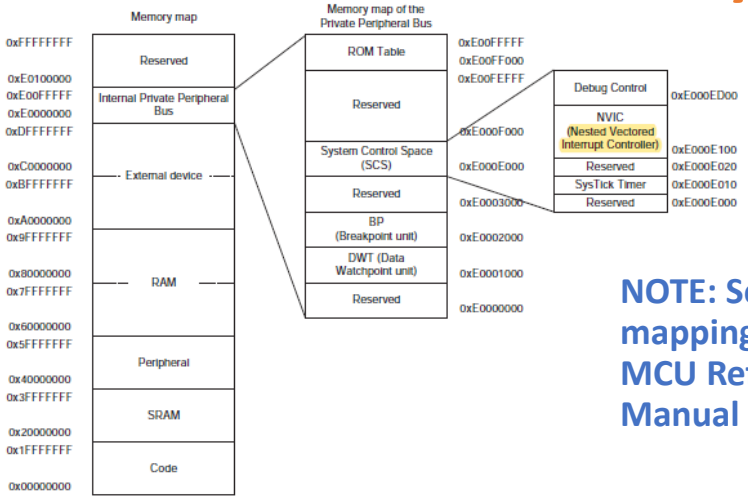


Figure 7.2
Architecturally defined memory map of the Cortex®-M0/M0+ processor.

NOTE: See detailed mapping info in MCU Reference Manual (Ch.3)

Source: [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2). 10

10

STM32L053R8 - Datasheet

life.augmented

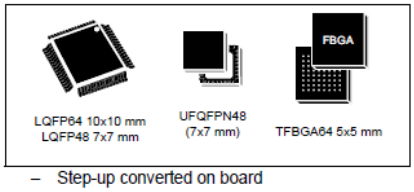
STM32L053C6 STM32L053C8
STM32L053R6 STM32L053R8

Ultra-low-power 32-bit MCU Arm®-based Cortex®-M0+, up to 64KB Flash, 8KB SRAM, 2KB EEPROM, LCD, USB, ADC, DAC

Datasheet - production data

Features

- Ultra-low-power platform
  - 1.65 V to 3.6 V power supply
  - -40 to 125 °C temperature range
  - 0.27 µA Standby mode (2 wakeup pins)
  - 0.4 µA Stop mode (16 wakeup lines)
  - 0.8 µA Stop mode + RTC + 8-Kbyte RAM retention
  - 88 µA/MHz in Run mode
  - 3.5 µs wakeup time (from RAM)
  - 5 µs wakeup time (from Flash memory)
- Core: Arm® 32-bit Cortex®-M0+ with MPU
  - From 32 kHz up to 32 MHz max.
  - 0.95 DMIPS/MHz
- Memories
  - Up to 64-Kbyte Flash memory with ECC
  - 8-Kbyte RAM
  - 2 Kbytes of data EEPROM with ECC
  - 20-byte backup register
  - Sector protection against R/W operation



  - Step-up converted on board
- Rich Analog peripherals
  - 12-bit ADC 1.14 Msps up to 16 channels (down to 1.65 V)
  - 12-bit 1 channel DAC with output buffers (down to 1.8 V)
  - 2x ultra-low-power comparators (window mode and wake up capability, down to 1.65 V)
- Up to 24 capacitive sensing channels supporting touchkey, linear and rotary touch sensors
- 7-channel DMA controller, supporting ADC, SPI, I2C, USART, DAC, Timers
- 8x peripheral communication interfaces

11

NUCLEO-L053R8

- Package pin count: 64 pins
- Flash memory size: 64 KB

Table 2. Codification explanation

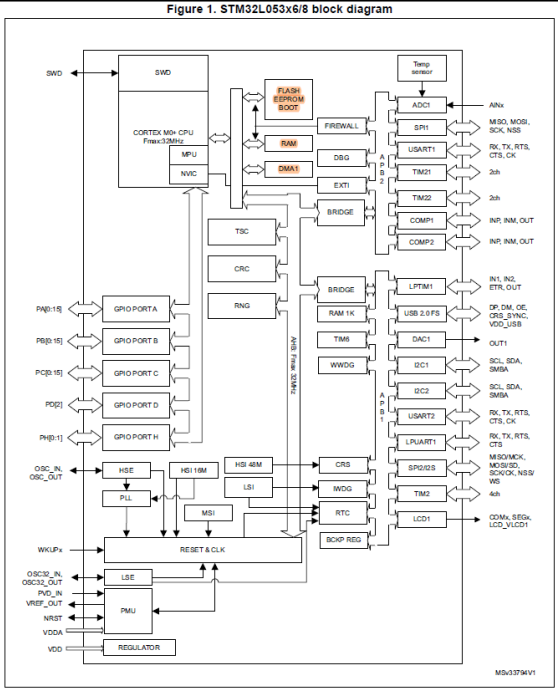
NUCLEO-XXYYRT	Description	Example: NUCLEO-L452RE
XX	MCU series in STM32 Arm Cortex MCUs	STM32L4 Series
YY	STM32 product line in the series	STM32L452
R	STM32 package pin count	64 pins
T	STM32 Flash memory size: – 8 for 64 Kbytes – B for 128 Kbytes – C for 256 Kbytes – E for 512 Kbytes – G for 1 Mbyte – Z for 192 Kbytes	512 Kbytes

Source: Board user manual

12

12

Figure 1. STM32L053x6/8 block diagram



13

3.8 Memories

The STM32L053x6/8 devices have the following features:

- 8 Kbytes of embedded SRAM accessed (read/write) at CPU clock speed with 0 wait states. With the enhanced bus matrix, operating the RAM does not lead to any performance penalty during accesses to the system bus (AHB and APB buses).
- The non-volatile memory is divided into three arrays:
  - 32 or 64 Kbytes of embedded Flash program memory
  - 2 Kbytes of data EEPROM
  - Information block containing 32 user and factory options bytes plus 4 Kbytes of system memory

The user options bytes are used to write-protect or read-out protect the memory (with 4 Kbyte granularity) and/or readout-protect the whole memory with the following options:

- **Level 0**: no protection
- **Level 1**: memory readout protected.

  The Flash memory cannot be read from or written to if either debug features are connected or boot in RAM is selected
- **Level 2**: chip readout protected, debug features (Cortex-M0+ serial wire) and boot in RAM selection disabled (debugline fuse)

The firewall protects parts of code/data from access by the rest of the code that is executed outside of the protected area. The granularity of the protected code segment or the non-volatile data segment is 256 bytes (Flash memory or EEPROM) against 64 bytes for the volatile data segment (RAM).

The whole non-volatile memory embeds the error correction code (ECC) feature.

Source: MCU Datasheet

14

3.9 Boot modes

At startup, BOOT0 pin and nBOOT1 option bit are used to select one of three boot options:

- Boot from Flash memory
- Boot from System memory
- Boot from embedded RAM

The boot loader is located in System memory. It is used to reprogram the Flash memory by using SPI1(PA4, PA5, PA6, PA7) or SPI2 (PB12, PB13, PB14, PB15), USART1(PA9, PA10) or USART2(PA2, PA3). See STM32™ microcontroller system memory boot mode AN2606 for details.

Source: MCU Datasheet

15

Memory Attributes and Memory Access Permission

- To make porting of software between different devices easier, a number of memory attribute settings are available for each regions in the memory map.
- Memory attributes are characteristics of the memory accesses; they can affect data and instruction accesses to memory as well as accesses to peripherals.

16

16

Memory Attributes

Executable—The executable attribute defines whether program execution is allowed in that memory region. If a memory region is defined as nonexecutable, in ARM documentation it is marked as eXecute Never (XN).

Bufferable—When a data write is carried out to a bufferable memory region, the write transfer can be buffered, which means the processor can continue to execute next instruction without waiting for the current write transfer to complete.

Cacheable—If a cache device is present on the system, it can keep a local copy of the data during a data transfer, and reuse it next time the same memory location is accessed to speed up the system. The cache device can be a cache memory unit, or could be a small buffer in a memory controller.

Shareable—The shareable attribute defines whether a memory region can be accessed by more than one processor. If a memory region is shareable, the memory system needs to ensure coherency between memory accesses by multiple processors in this region.

17

17

Memory Attributes

- Memory attributes used to define what type of devices could be used in each memory region

Normal memory—Normal memories can be shareable or nonshareable, and can be either cacheable or noncacheable. For memories with cacheable, the caching behavior can be further divided into Write Through (WT) or Write Back Write Allocate (WBWA).

Device memory—Device memories are noncacheable. They can be shareable or nonshareable.

Strongly Ordered (SO) memory—A memory region that is nonbufferable, noncacheable and transfer to/from SO region takes effect immediately. Also, the orders of SO transfers on the memory interface must be identical to the orders of the corresponding memory access instructions (i.e., no access reordering for speed optimization—please note that the Cortex-M0 and Cortex-M0+ processors do not have such access reordering feature anyway). SO memory regions are always shareable in terms of architectural definition.

18

18

Table 7.3: Default memory attribute map defined by the architecture

Address	Region	Memory type	Cache	XN	Shareable	Descriptions
0x00000000–0x1FFFFFFF	CODE	Normal	WT	–	–	Memory for program code including vector table
0x20000000–0x3FFFFFFF	SRAM	Normal	WBWA	–	–	SRAM, typically used for data and stack memory
0x40000000–0x5FFFFFFF	Peripheral	Device	–	XN	–	Typically used for on-chip devices
0x60000000–0x7FFFFFFF	RAM	Normal	WBWA	–	–	Normal memory with Write Back, Write Allocate cache attributes
0x80000000–0x9FFFFFFF	RAM	Normal	WT	–	–	Normal memory with Write Through cache attributes
0xA0000000–0xBFFFFFFF	Device	Device	–	XN	S	Shareable device memory
0xC0000000–0xDFFFFFFF	Device	Device	–	XN	–	Nonshareable device memory
0xE0000000–0xE00FFFFF	PPB	Strongly ordered	–	XN	S	Internal Private Peripheral Bus
0xE0100000–0xFFFFFFFF	Reserved	Reserved	–	–	–	Reserved (Vendor-specific usage)

Source: [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2). 19

19

Memory access permission for regions

Table 7.4: Memory access permission

Memory region	Default permission	Note
CODE, SRAM, Peripheral, RAM, Device	Accessible for both privileged and unprivileged code.	Access permission can be overridden by MPU configurations
System Control Space including NVIC, MPU, SysTick	Accessible for privileged code only. Attempts to access these registers from unprivileged code result in HardFault exception.	Cannot be overridden by MPU configurations

Source: [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2). 20

20

Memory Protection Unit (MPU)

3.3 Arm® Cortex®-M0+ core with MPU

The Cortex-M0+ processor is an entry-level 32-bit Arm Cortex processor designed for a broad range of embedded applications. It offers significant benefits to developers, including:

- a simple architecture that is easy to learn and program
- ultra-low power, energy-efficient operation
- excellent code density
- deterministic, high-performance interrupt handling
- upward compatibility with Cortex-M processor family
- platform security robustness, with integrated Memory Protection Unit (MPU).

Source: MCU Datasheet

21

Memory Protection Unit (MPU)

- Memory Protection Unit (MPU) is a programmable block inside the processor that defines memory attributes and memory access permissions.
- MPU is used to detect problems in the system
  - e.g., when an application task behaves erroneously by trying to access a memory location which is invalid or disallowed
- MPU can be used to make an embedded system more robust, and in some cases make the system more secure by:
  - Preventing application tasks from corrupting stack or data memory used by other tasks
  - Preventing unprivileged tasks from accessing certain peripherals
  - Defining SRAM or RAM space as nonexecutable to prevent code injection attacks
- MPU is disabled by default

22

22

Direct Memory Access (DMA)

3.10 Direct memory access (DMA)

The flexible 7-channel general-purpose DMA is able to manage memory-to-memory, peripheral-to-memory and memory-to-peripheral transfers. The DMA controller supports circular buffer management, avoiding the generation of interrupts when the controller reaches the end of the buffer.

Each channel is connected to dedicated hardware DMA requests, with software trigger support for each channel. Configuration is done by software and transfer sizes between source and destination are independent.

The DMA can be used with the main peripherals: SPI, I^2C, USART, LPUART, general-purpose timers, DAC, and ADC.

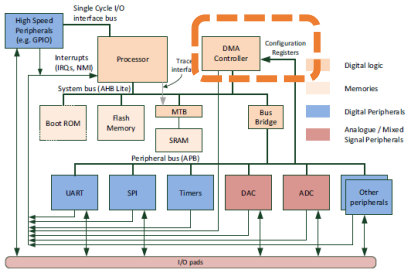


Figure 2.6
A system with the Cortex®-M0+ Processor and a DMA Controller.

Source: MCU Datasheet

Need for DMA

```
uint8_t buf[20];
...
HAL_UART_Receive(&huart2, buf, 20, HAL_MAX_DELAY);
```

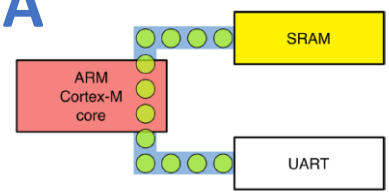


Figure 9.1: the flow of data during a transfer from peripheral to SRAM

- CPU will be involved during these operations, even if its role is "limited" to move data from peripheral to SRAM
- This simplifies design of the hardware, but introduces performance penalties
- Cortex-M core is "responsible" to load data from SRAM to UART peripheral - this is a **blocking operation**
- Prevents the CPU from doing other activities
- It also requires the CPU to wait for "slower" units completing their job
- **This is the reason why high-performance MCUs provide DMA controllers**

Source: [Book 1] Carmine Noviello, Mastering STM32, Second Edition, 2022.

DMA

- Direct memory access (DMA) controller is a bus master and system peripheral.
- The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.
- The DMA controller features a single AHB master architecture.
- There is one instance of DMA with 7 channels.
- Each channel is dedicated to managing memory access requests from one or more peripherals.
- The **DMA includes an arbiter** for handling the priority between DMA requests.

25

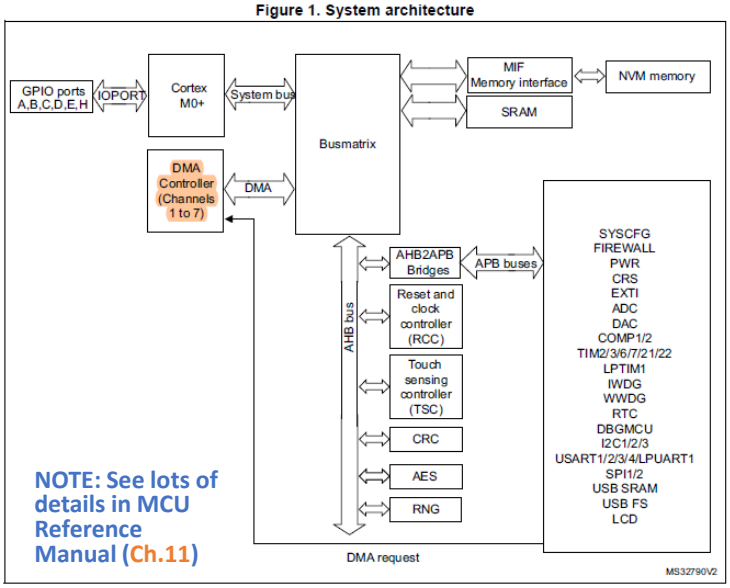
25

System architecture

The main system consists of:

- Two masters:
  - Cortex®-M0+ core (AHB-lite bus)
  - GP-DMA (general-purpose DMA)
- Three slaves:
  - Internal SRAM
  - Internal Non-volatile memory
  - AHB to APB, which connects all the APB peripherals

Figure 1. System architecture



NOTE: See lots of details in MCU Reference Manual (Ch.11)

Source: MCU Reference Manual

26

26

DMA Channels

- A **channel** is used to exchange data between two memory regions in the 4GB address space.
- Peripherals are slave units: they cannot access the bus independently.
- A master is always needed to start a transaction.
- A way to notify that the peripherals are ready to exchange data => dedicated number of DMA requests lines are available from peripherals to the DMA controller.

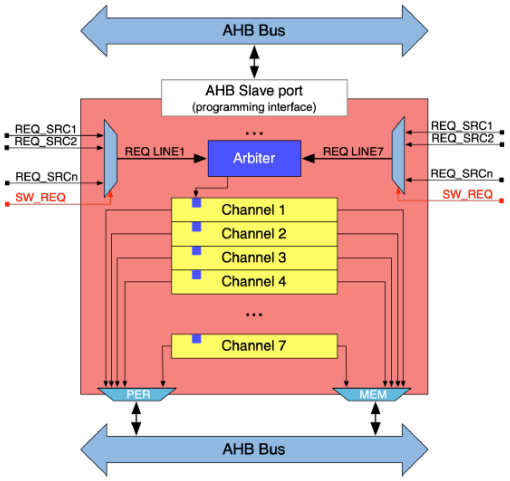


Figure 9.3: A representation of the DMA structure in F0/F1/F3/L0/L1/L4 MCUs

Source: [Book 1] Carmine Noviello, Mastering STM32, Second Edition, 2022.

27

HAL DMA Module

```
typedef struct {
  DMA_Channel_TypeDef       *Instance;          /* Register base address           */
  DMA_InitTypeDef           Init;              /* DMA communication parameters    */
  HAL_LockTypeDef           Lock;              /* DMA locking object              */
  __IO HAL_DMA_StateTypeDef State;             /* DMA transfer state              */
  void                      *Parent;           /* Parent object state             */
  void                      (* XferCpltCallback)( struct __DMA_HandleTypeDef * hdma);
  void                      (* XferHalfCpltCallback)( struct __DMA_HandleTypeDef * hdma);
  void                      (* XferErrorCallback)( struct __DMA_HandleTypeDef * hdma);
  __IO uint32_t             ErrorCode;         /* DMA Error code                  */
} DMA_HandleTypeDef;

typedef struct {
  uint32_t Direction;
  uint32_t PeriphInc;
  uint32_t MemInc;
  uint32_t PeriphDataAlignment;
  uint32_t MemDataAlignment;
  uint32_t Mode;
  uint32_t Priority;
} DMA_InitTypeDef;
```

28

28

(1) Perform DMA Transfers in Polling Mode

- Once we have configured the DMA channel/stream, we must do few other things:
  1. to setup the addresses on the memory and peripheral port;
  2. to specify the amount of data we are going to transfer;
  3. to arm the DMA;
  4. to enable the DMA mode on the corresponding peripheral;

- First three points by using:
  - **HAL_StatusTypeDef HAL_DMA_Start(…);**
- Fourth point is peripheral dependent

29

29

Example 1

- Sending a string over UART2 peripheral using DMA mode
- Steps
  1. UART2 is configured using the HAL_UART module
  2. DMA1 channel is configured to do a memory-to-peripheral transfer
  3. Corresponding channel is armed to execute the transfer and UART is enabled in DMA mode
- **See demo in class (also see Ch.9 of textbook)**

30

30

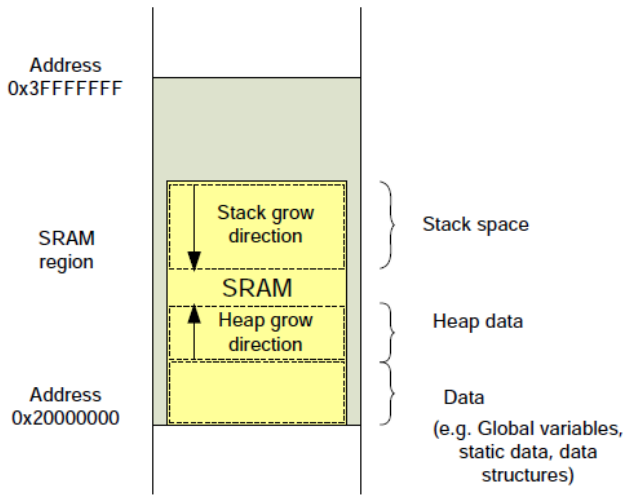
(2) Perform DMA Transfers in Interrupt Mode

- DMA can generate interrupts related to channel activities
- DMA can be enabled in interrupt mode following steps:
  1) Define three functions acting as callback routines and pass them to function pointers XferCplt-Callback, XferHalfCpltCallback and XferErrorCallback in a DMA_HandleTypeDef handler (it is ok to define only the functions we are interested in);
  2) Write ISR for the IRQ associated to the channel you are using and do a call to the HAL_DMA_IRQHandler() passing the reference to the DMA_HandleTypeDef handler;
  3) Enable the corresponding IRQ in the NVIC controller;
  4) Use function HAL_DMA_Start_IT(), which automatically performs all the necessary setup steps, passing to it same arguments of HAL_DMA_Start().
- Example 2
  - **See demo in class (also see Ch.9 of textbook)**

31

31

Program Memory Model



32

Program Memory Model

- RAM for an executing program is divided into three regions:
  1) **Data** in RAM are allocated during the **link process** and initialized by startup code at reset
  2) The (optional) **Heap** is managed at **runtime** by library code implementing functions such as the *malloc* and *free* which are part of the standard C library
  3) The **Stack** is managed at **runtime** by compiler generated code which generates per-procedure-call stack frames containing local variables and saved registers

33

Program Code

- Program code can be located in:
  - the Code region
  - the SRAM region
  - the External RAM region
- Program code typically stored in flash memory (i.e., code region)

34

References & Credits

- [Book 1] Carmine Noviello, Mastering STM32, Second Edition, 2022.
- [Book 2] Joseph Jiu, The Definitive guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2015.
- https://www.st.com/content/st_com/en/arm-32-bit-microcontrollers/arm-cortex-m0-plus.html
- STM32L053R8 MCU
  - Datasheet
  - User Manual
- NUCLEO-L053R8 Board
  - User Manual

35

Outline

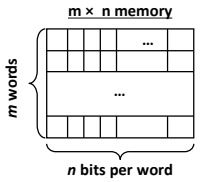
- Registers
- Memory map
- Memory protection unit (MPU)
- Direct memory access (DMA)
- Program memory model
- Memories – basic concepts

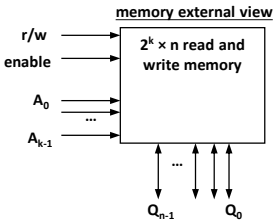
36

36

Memory: basic concepts

- Stores large number of bits
  - *m* x *n*: *m* words of *n* bits each
  - k = $\text{Log}_2(m)$ address input signals
  - or $m = 2^k$ words
  - e.g., 4,096 x 8 memory:
    - 32,768 bits
    - 12 address input signals
    - 8 input/output data signals
- Memory access
  - r/w: selects read or write
  - enable: read or write only when asserted
  - multiport: multiple accesses to different locations simultaneously





37

Memory: basic categories

Writable?

- **Read-Only Memory (ROM):**
  - Can only be read; cannot be modified (written) by the processor. Contents of ROM chip are set before chip is placed into the system.
- **Random-Access Memory (RAM)**:
  - Read/write memory. Although technically inaccurate, term is used for historical reasons. (ROMs are also random access.)

Permanence?

- **Volatile memories**
  - Lose their contents when power is turned off. Typically used to store program while system is running.
- **Non-volatile memories** do not.
  - Required by every system to store instructions that get executed when system powers up (boot code).

38

Memories Classification

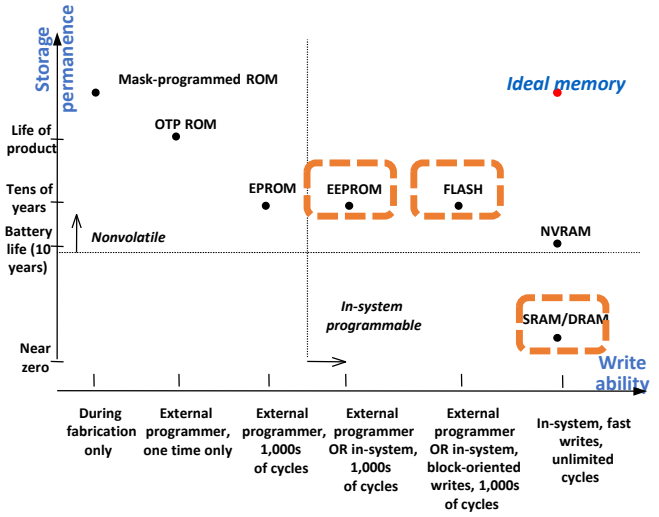
Read-Write Memory			Read-Only Memory
Volatile Memory		Non-volatile Memory	Mask-Programmed ROM (PROM) (nonvolatile)
Random Access	Sequential Access	EPROM EEPROM FLASH	
SRAM DRAM	FIFO LIFO Shift Register CAM		

- Volatile: need electrical power
- Nonvolatile: magnetic disk, retains its stored information after the removal of power
- Random access: memory locations can be read or written in a random order
- EPROM: erasable programmable read-only memory
- EEPROM: electrically erasable programmable read-only memory
- FLASH: memory stick, USB disk
- Access pattern: sequential access: (video memory streaming) first-in-first-out (buffer), last-in-first-out (stack), shift register, content-addressable memory
- Static vs. Dynamic: dynamic needs periodic refresh but is simpler, higher density

- Key Design Metrics:
  1. Memory Density (number of bits/mm^2) and Size
  2. Access Time (time to read or write) and Throughput
  3. Power Dissipation

39

Write-ability and Storage-permanence



Write ability and storage permanence of memories, showing relative degrees along each axis (not to scale)

40

Write-ability

- Ranges of write ability
  - High end
    - processor writes to memory simply and quickly
    - e.g., RAM
  - Middle range
    - processor writes to memory, but slower
    - e.g., FLASH, EEPROM
  - Lower range
    - special equipment, “programmer”, must be used to write to memory
    - e.g., EPROM, OTP ROM
  - Low end
    - bits stored only during fabrication
    - e.g., Mask-programmed ROM
- In-system programmable memory
  - Can be written to by a processor in the microcomputer system using the memory
  - Memories in high end and middle range of write ability

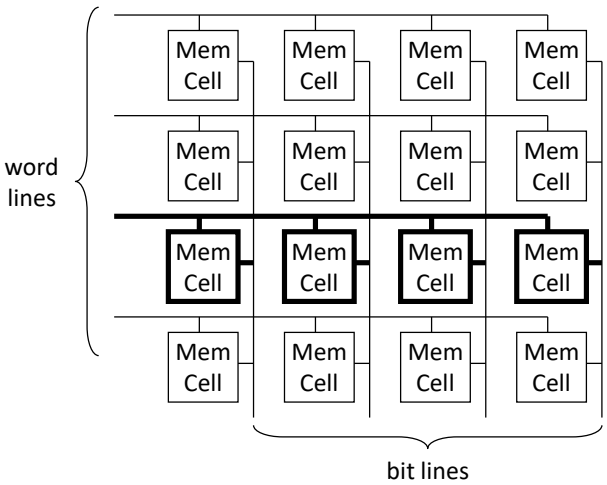
41

Storage-permanence

- Range of storage permanence
  - High end
    - essentially never loses bits
    - e.g., mask-programmed ROM
  - Middle range
    - holds bits days, months, or years after memory’s power source turned off
    - e.g., NVRAM
  - Lower range
    - holds bits as long as power supplied to memory
    - e.g., SRAM
  - Low end
    - begins to lose bits almost immediately after written – refreshing needed
    - e.g., DRAM
- Nonvolatile memory
  - Holds bits after power is no longer supplied
  - High end and middle range of storage permanence

42

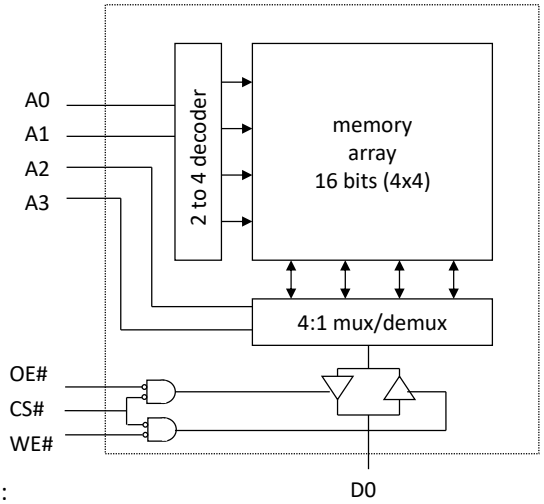
Memory array



Different memory types are distinguished by technology for storing bit in memory cell.

43

Support circuitry



Control signals:

- Control read/write of array
- Map internal physical array to external configuration (4x4 → 16x1)

44

Interface (1/2)

- Physical configurations are typically square.
  - Minimize length word + bit line → minimize access delays.
- External configurations are "tall and narrow". The narrower the configuration, the higher the pin efficiency. (Adding one address pin cuts data pins in half.)
  - Several external configurations available for a given capacity.
  - 64Kbits may be available as 64Kx1, 32Kx2, 16Kx4,…

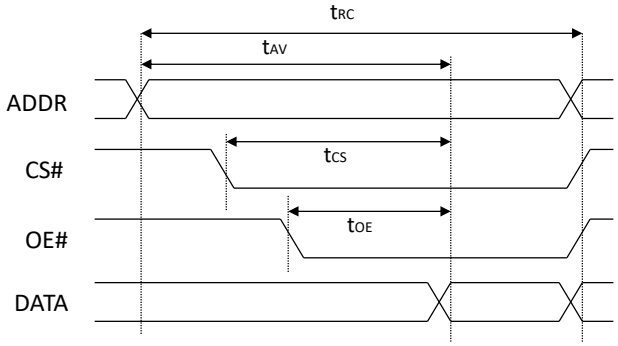
45

Interface (2/2)

- Chip Select (CS#): Enables device. If not asserted, device ignores all other inputs (sometimes entering low-power mode).
- Write Enable (WE#): Store D0 at specified address.
- Output Enable (OE#): Drive value at specified address onto D0.

46

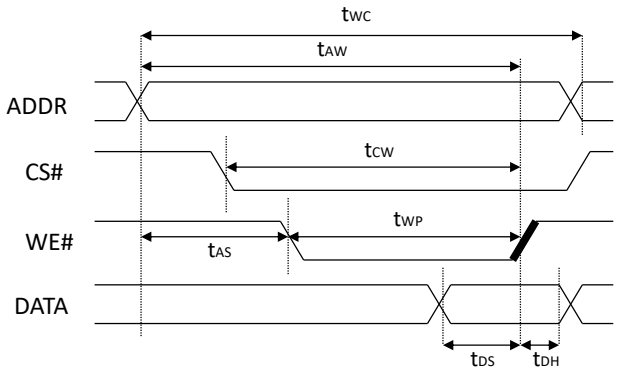
Memory timing: Reads



- **Access time**: Time required from start of a read access to valid data output.
  - Access time specified for each of the three conditions required for valid data output (valid address, chip select, output enable)
- Time to valid data out depends on which of these is on critical path.
- t_{RC}: Minimum time required from start of one access to start of next.
  - For most memories equal to access time.

47

Memory timing: Writes



- Write happens on **rising** edge of WE#
- Separate access times t_{AW}, t_{CW}, t_{WP} specified for address valid, CS#, WE#.
- Typically, $t_{AS} = 0$, meaning that WE# may **not** be asserted **before** address is valid.
- Setup and hold times required for data.
- Write cycle time t_{WC} is typically in the order of t_{AW}.

48

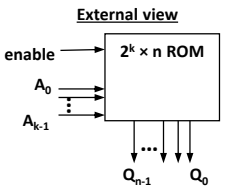
Memory Comparison

Memory Type	Read Speed	Write Speed	Volatility	Density	Power	Rewrite
SRAM	+++	+++	-	-		++
DRAM	+	+	- -	++	-	++
EPROM	+	-	+		+	-
EEPROM	+	-	+		+	+
Flash	+		+	+	+	+

49

ROM: "Read-Only" Memory

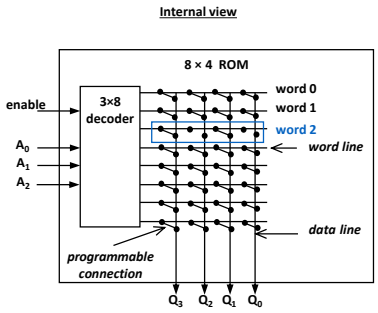
- Nonvolatile
- Can be read from but not written to
- Traditionally written to, "programmed", before inserting to microcomputer system
- Uses
  - Store software program for general-purpose processor
  - Store constant data (parameters) needed by system
  - Implement combinational circuits (e.g., decoders)



50

Example: 8 x 4 ROM

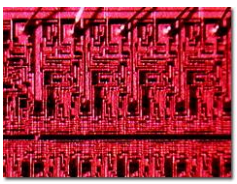
- Horizontal lines = words
- Vertical lines = data
- Lines connected only at circles
- Decoder sets word 2's line to 1 if address input is 010
- Data lines Q_3 and Q_1 are set to 1 because there is a "programmed" connection with word 2's line
- Word 2 is not connected with data lines Q_2 and Q_0
- Output is 1010



51

Mask-programmed ROM

- Connections "programmed" at fabrication
  - set of masks
- Lowest write ability
  - only once
- Highest storage permanence
  - bits never change unless damaged
- Typically used for final design of high-volume systems
  - spread out NRE (non-recurrent engineering) cost for a low unit cost



52

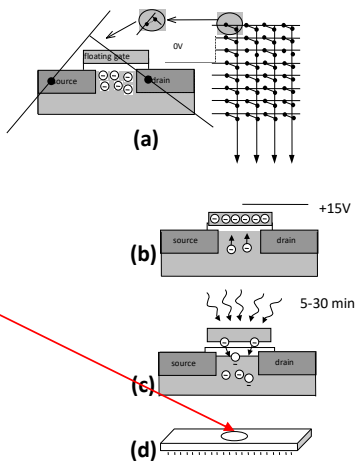
OTP ROM: One-time programmable ROM

- Connections "programmed" after manufacture by user
  - user provides file of desired contents of ROM
  - file input to machine called ROM programmer
  - each programmable connection is a fuse
  - ROM programmer blows fuses where connections should not exist
- Very low write ability
  - typically written only once and requires ROM programmer device
- Very high storage permanence
  - bits don't change unless reconnected to programmer and more fuses blown
- Commonly used in final products
  - cheaper, harder to inadvertently modify

53

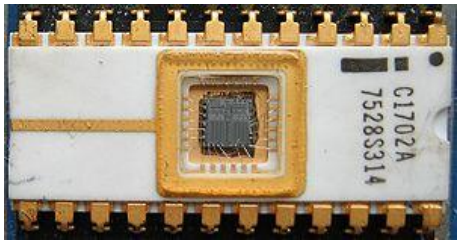
EPROM: UV Erasable programmable ROM

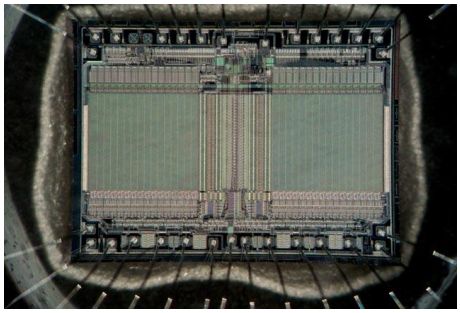
- **Programmable component is a MOS transistor**
  - Transistor has "floating" gate surrounded by an insulator
  - **(a)** Negative charges form a channel between source and drain storing a logic 1
  - **(b)** Large positive voltage at gate causes negative charges to move out of channel and get trapped in floating gate storing a logic 0
  - **(c)** (Erase) Shining UV rays on surface of floating-gate causes negative charges to return to channel from floating gate restoring the logic 1
  - **(d)** An EPROM package showing quartz window through which UV light can pass
- **Better write ability**
  - can be erased and reprogrammed thousands of times
- **Reduced storage permanence**
  - program lasts about 10 years but is susceptible to radiation and electric noise
- **Typically used during design development**



54

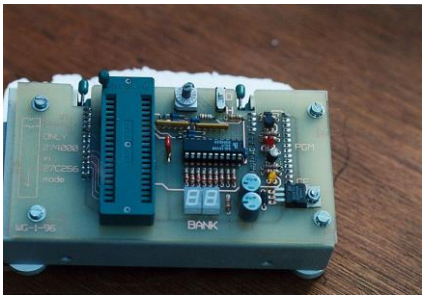
Sample EPROM components





55

Sample EPROM programmers



56

EEPROM: Electrically erasable programmable ROM

- Programmed and erased electronically
  - typically by using higher than normal voltage
  - can program and erase individual words
- Better write ability
  - can be in-system programmable with built-in circuit to provide higher than normal voltage
    - built-in memory controller commonly used to hide details from memory user
  - writes very slow due to erasing and programming
    - "busy" pin indicates to processor EEPROM still writing
  - can be erased and programmed tens of thousands of times
- Similar storage permanence to EPROM (about 10 years)
- Far more convenient than EPROMs, but more expensive

57

FLASH

- Extension of EEPROM
  - Same floating gate principle
  - Same write ability and storage permanence
- Fast erase
  - Large blocks of memory erased at once, rather than one word at a time
  - Blocks typically several thousand bytes large
- Writes to single words may be slower
  - Entire block must be read, word updated, then entire block written back

58

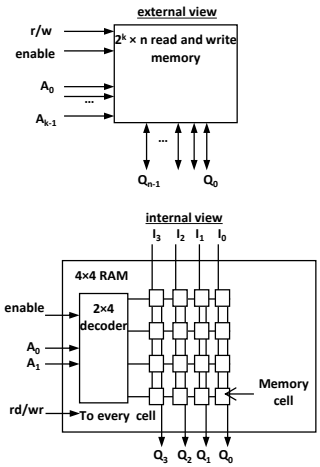
FLASH applications

- Flash technology has made rapid advances in recent years.
  - cell density rivals DRAM; better than EPROM; much better than EEPROM.
  - multiple gate voltages can encode 2 bits per cell.
  - many-GB devices available
- ROMs and EPROMs rapidly becoming obsolete.
- Replacing hard disks in some applications.
  - smaller, lighter, faster
  - more reliable (no moving parts)
  - cost effective
- PDAs, cell phones, laptops, iPods, etc…

59

RAM: Random-Access Memory

- **Typically volatile memory**
  - bits are not held without power supply
- **Read and written to easily by microprocessor during execution**
- **Internal structure more complex than ROM**
  - a word consists of several memory cells, each storing 1 bit
  - each input and output data line connects to each cell in its column
  - rd/wr connected to every cell
  - when row is enabled by decoder, each cell has logic that stores input data bit when rd/wr indicates write or outputs stored bit when rd/wr indicates read

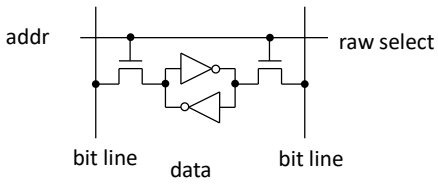


60

Basic Types of RAM: SRAM vs. DRAM

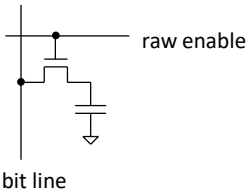
Primary difference between different memory types is the bit cell

- **SRAM Cell**



- **Larger cell ⇒ lower density, higher cost/bit**
- **No dissipation**
- **Read non-destructive**
- **No refresh required**
- **Simple read ⇒ faster access**
- **Standard IC process ⇒ natural for integration with logic**

- **DRAM Cell**



- **Smaller cell ⇒ higher density, lower cost/bit**
- **Needs periodic refresh, and refresh after read**
- **Complex read ⇒ longer access time**
- **Special IC process ⇒ difficult to integrate with logic circuits**

61

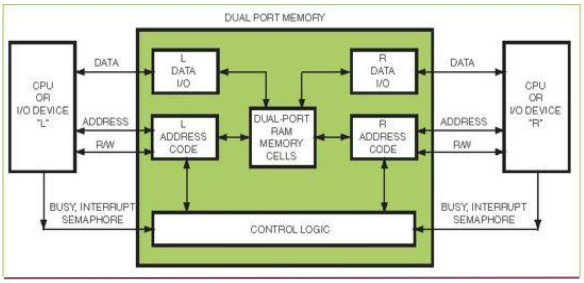
RAM variations

- PSRAM: Pseudo-static RAM
  - DRAM with built-in memory refresh controller
  - Popular low-cost high-density alternative to SRAM
- NVRAM: Nonvolatile RAM
  - Holds data after external power removed
  - Battery-backed RAM
    - SRAM with own permanently connected battery
    - writes as fast as reads
    - no limit on number of writes unlike nonvolatile ROM-based memory
  - SRAM with EEPROM or FLASH
    - stores complete RAM contents on EEPROM or FLASH before power turned off

62

Dual-port RAM (DPRAM)

- Usually a **Static RAM** circuit with two address and data bus connections
  - Shared RAM for two independent users
- Flexible communication link between two processors
  - Master/slave



63

DDR1 SDRAM, DDR2, …

- Double Data Rate synchronous dynamic random access memory (DDR1 SDRAM) is a class of memory integrated circuits **used in computers**.
- The interface uses double pumping (transferring data on both the rising and falling edges of the clock signal) to lower the clock frequency
- One advantage of keeping the clock frequency down is that it reduces the signal integrity requirements on the circuit board connecting the memory to the controller
- DDR2 memory is fundamentally similar to DDR SDRAM
- DDR2 SDRAM can perform four transfers per clock using a multiplexing technique

64