Lecture 16
Nonvolatile Memory

CE346 – Microprocessor System Design

Branden Ghena – Spring 2021

Some slides borrowed from:
Josiah Hester (Northwestern), Prabal Dutta (UC Berkeley)

Administrivia

- Friday
  - Will do last checkoffs for people who need them on Lab 6
  - Also available to discuss projects
    - Will put a sign-up form online

- Moving forward
  - Only two lectures left!!
  - Time to put some serious effort into projects (two weeks remaining)
  - Week after next is project demos
    - Details to come

Today's Goals

- Discuss uses of memory, especially nonvolatile memory, in embedded systems

- Introduce protocols for interacting with non-volatile memory
  - Internal Flash
  - External SD Card

Outline

- **Embedded Memories**
- nRF52 NVMC
- SD Cards

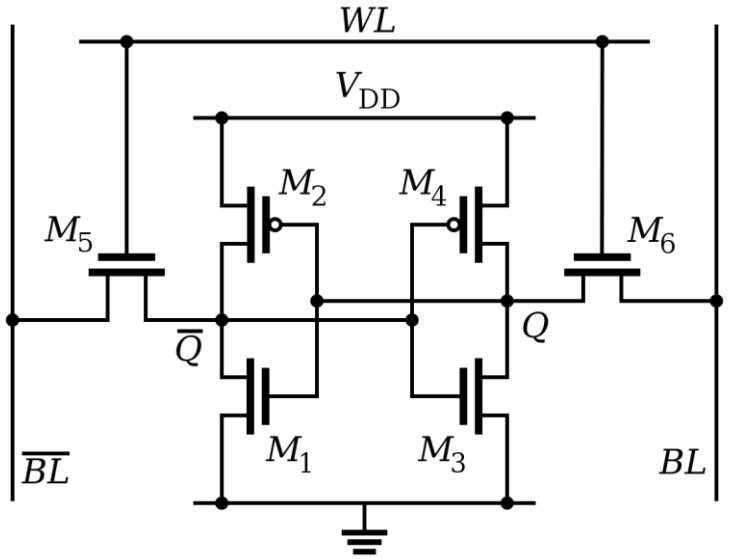
Memory in computing

- Various different memories serve different purposes in computing

- Needs
  - Fast, infinite-lifetime memory to keep things like stack memory
  - Nonvolatile memory that can be read from

- Desires
  - Fast, infinite-lifetime nonvolatile memory

Register technology: SRAM



- Static RAM (SRAM)
  - Each cell stores a bit in a bi-stable circuit, typically a six-transistor circuit
  - Static – no need for periodic refreshing; keeps data while powered
  - Relatively insensitive to disturbances such as electrical noise
    - Energetic particles (alpha particles, cosmic rays) can flip stored bits

- Fastest memory on computer
  - Also most expensive and takes up most space per bit
  - Typically used for registers and cache memories

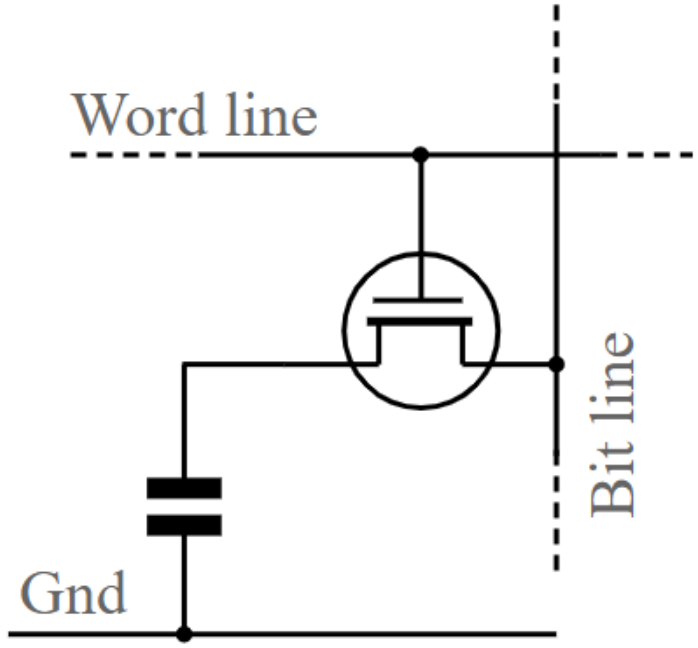
SRAM can be used a permanent memory in a pinch

- Gameboy and Gameboy Color used batteries to save state
- Gameboy Advanced games used batteries for an internal clock
- PSA: your old Gameboy games have likely lost their save files

Main memory technology: DRAM

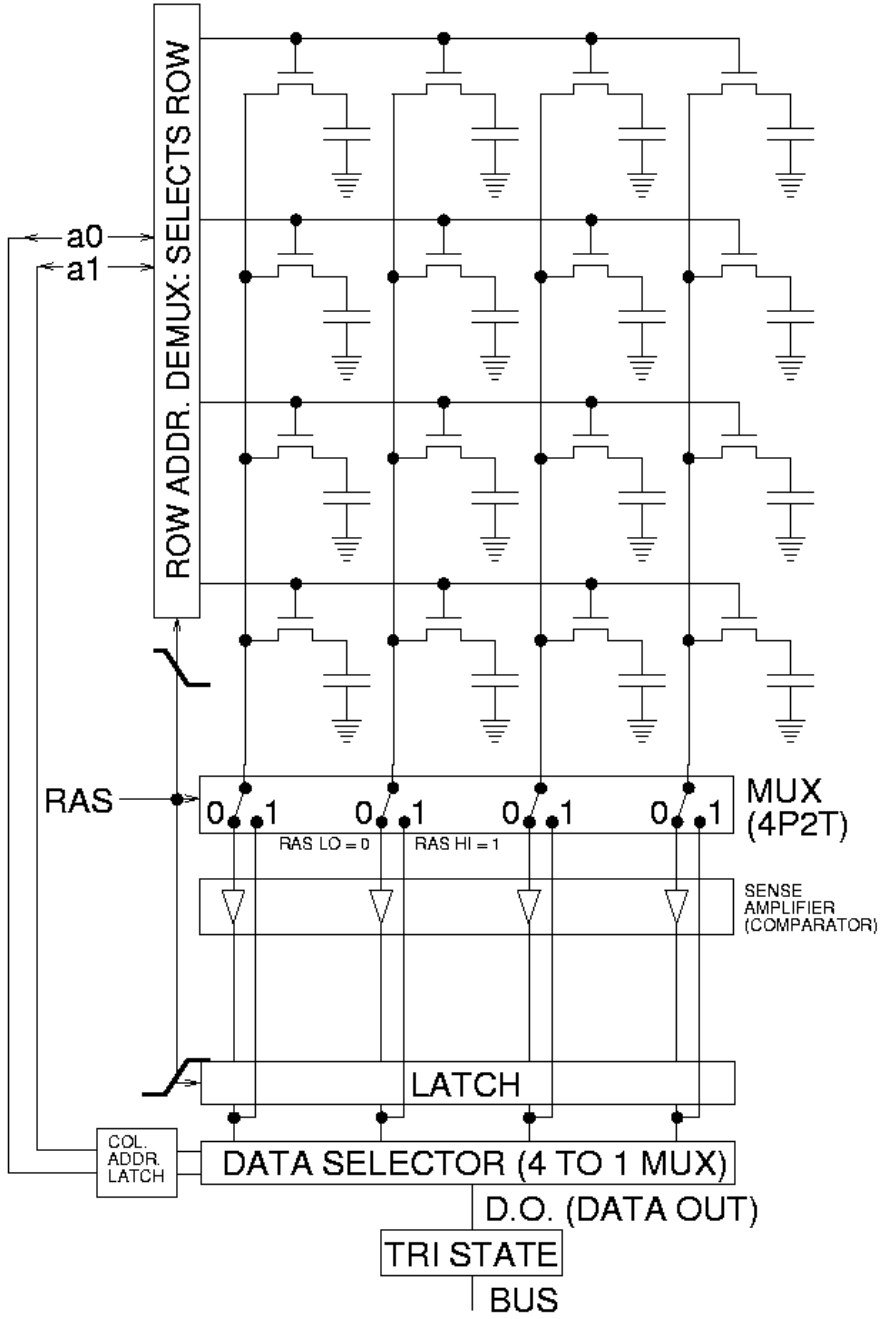
- Dynamic RAM (DRAM)
  - Each cell stores a bit as a charge in a capacitor
  - Capacitors lose charge; each cell must be refreshed every 10-100 ms
  - More sensitive to disturbances (EMI, radiation, …) than SRAM

- Slower than SRAM, but cheaper and denser
  - ~100x slower than registers



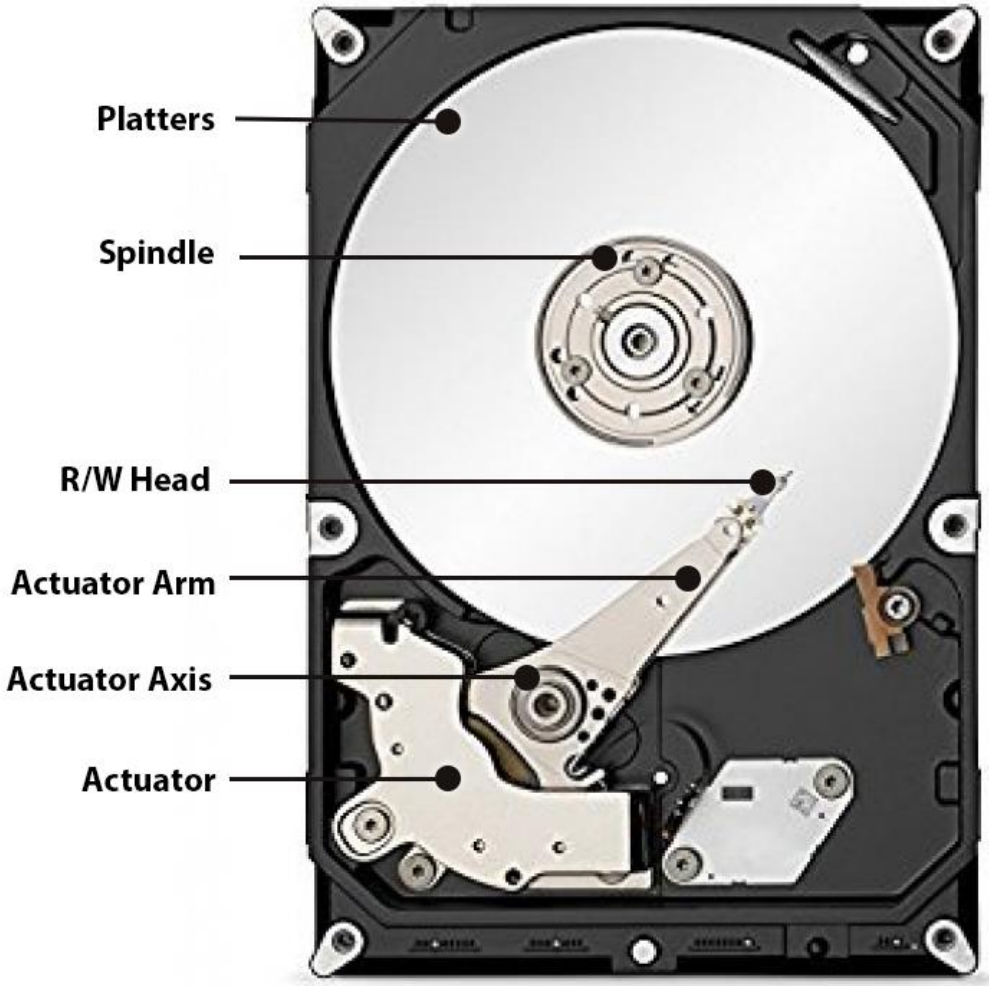
Accessing DRAM

- Read entire row of data at a time
  - Large in practice, kilobytes
- Select actual bytes that are wanted
  - Possibly modifying those bits
- Write row back to memory
  - Must always happen!
  - Reading is destructive
- Typically used for main memory in traditional computing systems
  - Constant refresh makes it untenable for low-power embedded systems



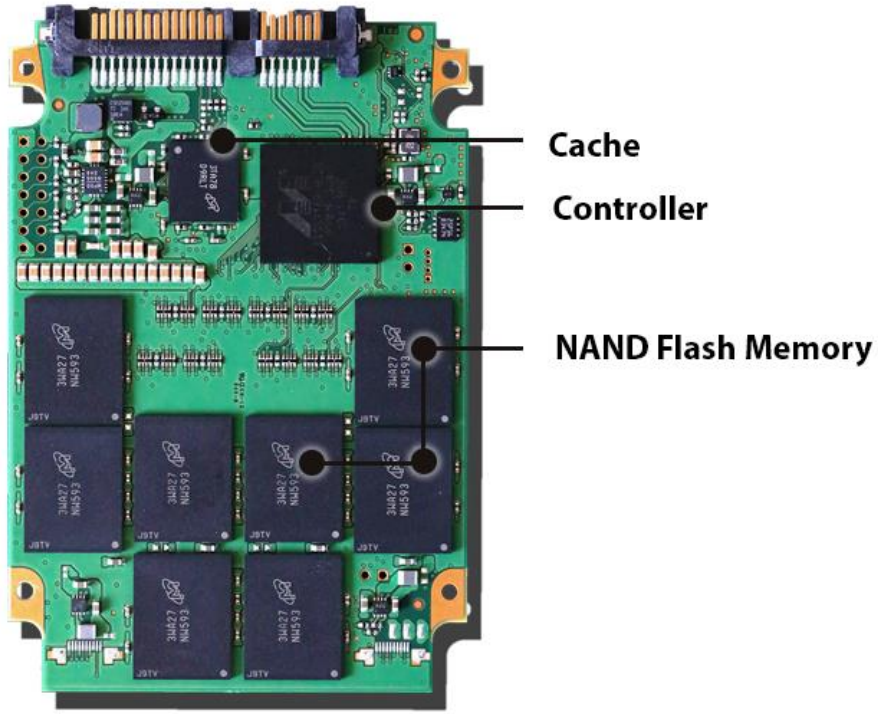
Disk drive storage

HDD
3.5″



Shock resistant up to 55g (operating)
Shock resistant up to 350g (non-operating)

SSD
2.5″



Shock resistant up to 1500g
(operating and non-operating)

Need breeds creativity

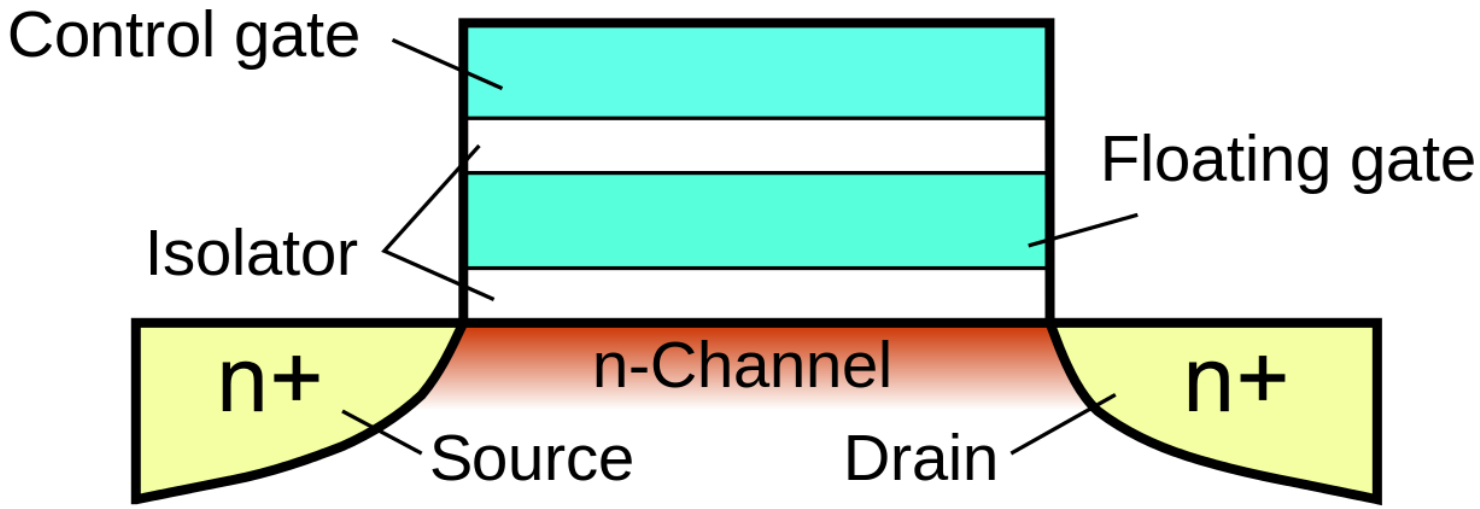
- Original iPod used a small disk drive

Floating-gate transistors

- Concept behind transistor-based non-volatile memory
  - EPROM, EEPROM, and Flash

  - High voltage on control gate creates charge on floating gate
  - Charge on floating gate activates/deactivates transistor

- High voltage degrades structure, leading it to eventually fail after enough writes



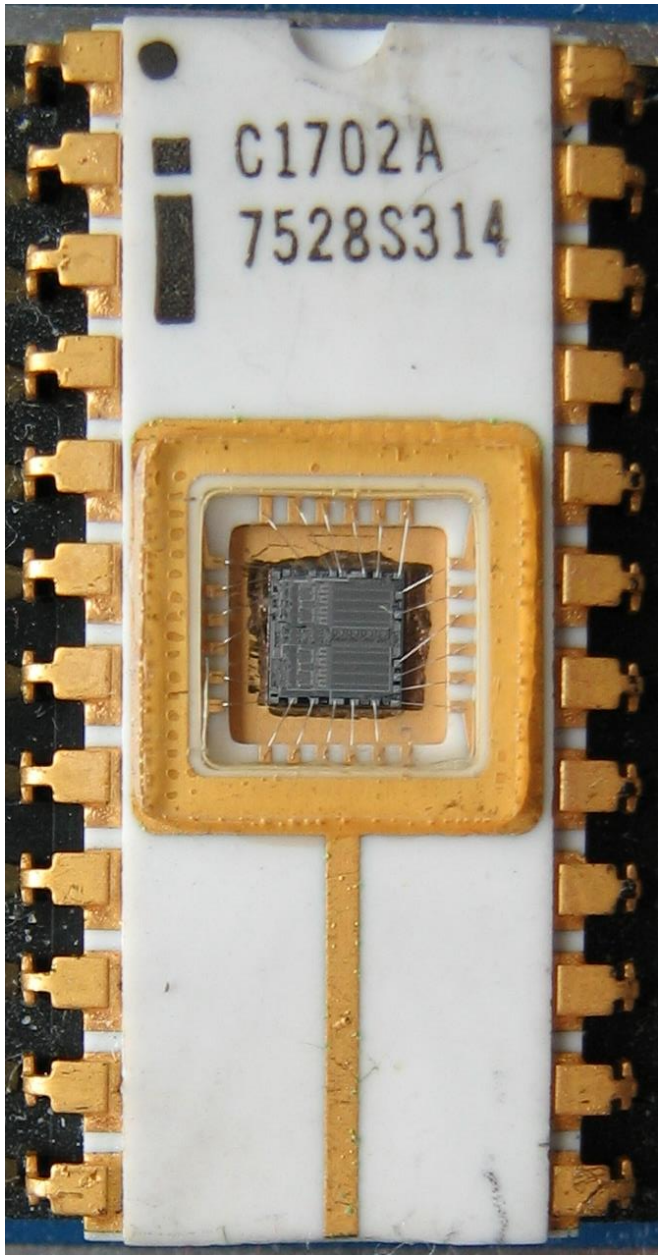
EPROM

- Erasable programmable read-only memory

- Erasable
  - If you shine UV light directly on the IC
  - Needed a window to expose the IC

- Programmable
  - With high voltage (25-50 volts)

- Typically acted as read-only memory in circuits

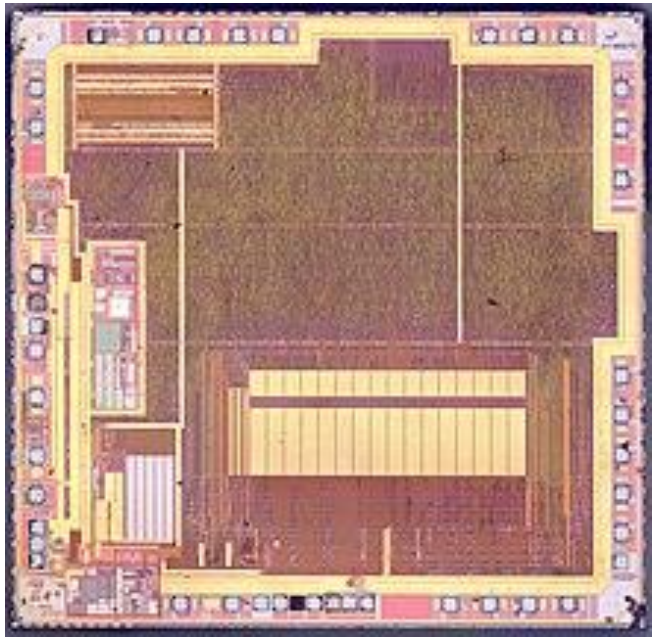


EEPROM

- Electrically-erasable programable read-only memory

- Same concept as EPROM, but includes internal circuitry to allow rewriting under normal conditions
  - Slow and high-power to write
  - Has a longer lifetime compared to flash, ~100k writes

- Can be built into other ICs
  - Example: AT90USB162 microcontroller (512 bytes)



Flash

- Similarly based on floating-gate transistors
  - But with a different design that allows for faster erase of entire blocks
  - More limited lifetime, ~1k-100k writes (10k common for embedded)

- Cannot erase individual bytes, must erase in units of blocks
  - Read can happen in units of bytes though

- Heavily used in commercial devices
  - Flash drives
  - SSDs
  - Smartphone storage
  - Microcontroller non-volatile storage!

More exotic memories

- FRAM and MRAM are both rising protentional Flash replacements
  - Non-volatile
  - Writable at the byte level
  - Very high to infinite write/erase cycles
  - Lower energy costs for writing and reading

- The two use unrelated magnetic techniques for data storage

- Starting to appear in microcontrollers
  - TI MSP430s have used 16 kB FRAM
  - Apollo4 (ARM Cortex-M4F) has 2 MB of MRAM

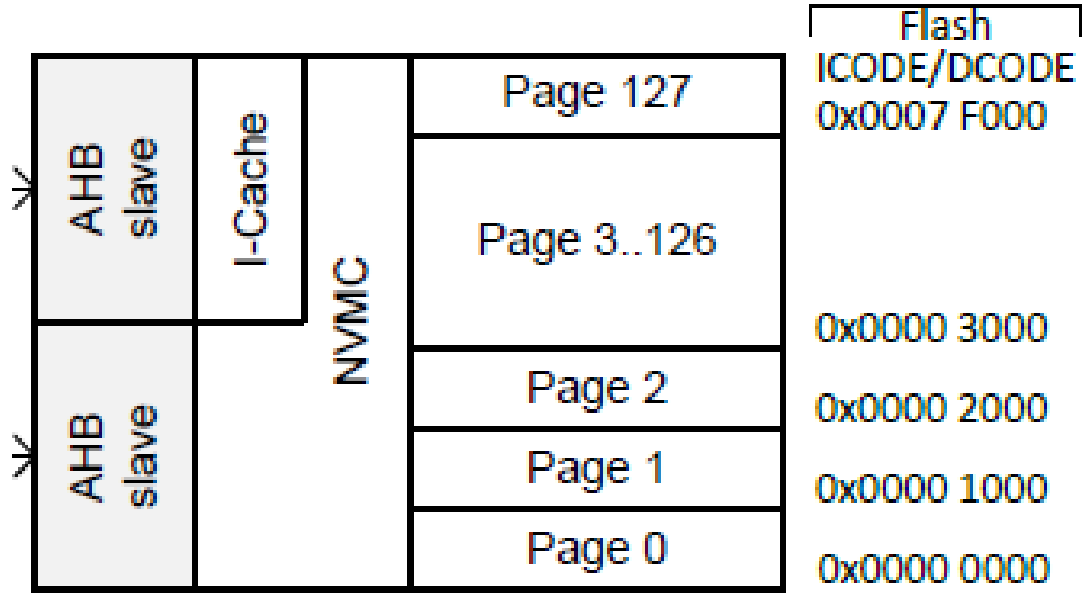
Outline

- Embedded Memories
- **nRF52 NVMC**
- SD Cards

Flash memory on the nRF52833

- 512 kB total Flash memory
  - 128 pages each 4 kB in size

- Non-Volatile Memory Controller (NVMC) controls access
  - Enables writing to flash
  - Enables erasing flash
  - Manages status of flash



Writing to Flash

- Configurable, disabled by default
  - Enable with configuration register

- Rules for writing to Flash
  - Must write word-aligned 32-bit values
  - Can only write 0 values, not ones
  - Can only write 2 times before erasing (even if there are still 1 bits)

- Takes 42.5 µs to write a 32-bit word
  - 64 MHz clock ⇨ 2720 cycles per 32-bit write

Erasing Flash

- Lifetime: 10000 erase cycles per page

- Options
  - Erase a single page (4 kB): 87.5 ms
  - Erase all of flash (512 kB): 173 ms

- CPU is halted if executing code from Flash during the erase
  - That’s 5.6 million cycles…
  - Code can execute from SRAM instead
  - Can also be split into a series of partial erases
    - Which must add up to a complete erase time before writing

Factory Information Configuration Registers

- Read-only memory

- Chip-specific information and configuration
  - Code size
  - Unique device ID
  - Production IDs
  - Temperature conversion functions

User Information Configuration Registers

- Additional Flash memory for non-volatile user configurations
  - Writable and erasable through NVMC processes described earlier
- 32 words of customer information (128 bytes total)
- Special configurations
  - Reset pin
  - NFC pin enable/disable
  - Debug configuration

Outline

- Embedded Memories
- nRF52 NVMC
- **SD Cards**

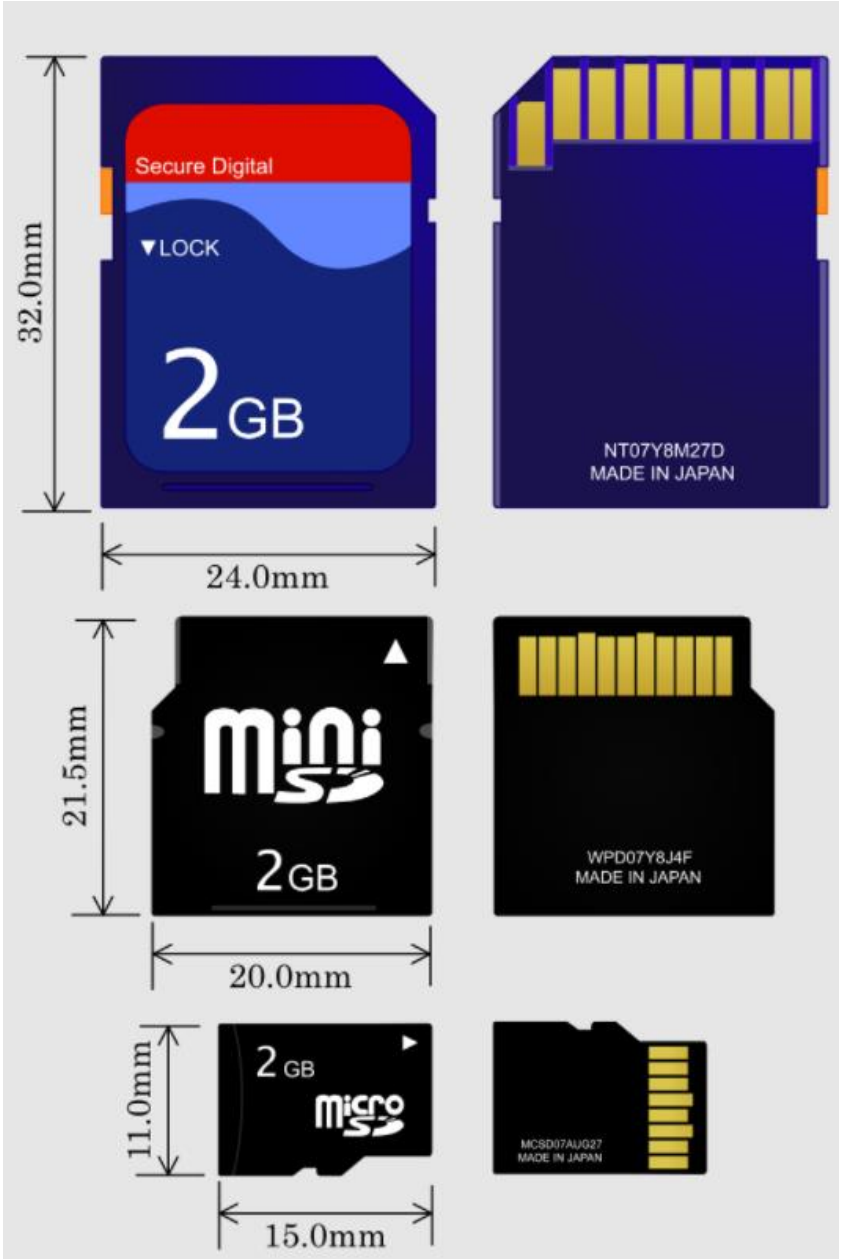
SD card references

- ChaN
  - Embedded systems engineer in Japan (and is amazing)
  - http://elm-chan.org/docs/mmc/mmc_e.html
  - http://elm-chan.org/fsw/ff/00index_e.html

- Various others
  - http://users.ece.utexas.edu/~gerstl/ee445m_s15/lectures/Lec08.pdf
  - http://alumni.cs.ucr.edu/~amitra/sdcard/Additional/sdcard_appnote_foust.pdf
  - https://luckyresistor.me/cat-protector/software/sdcard-2/
  - http://users.ece.utexas.edu/~valvano/EE345M/SD_Physical_Layer_Spec.pdf
  - https://github.com/tock/tock/blob/master/capsules/src/sdcard.rs

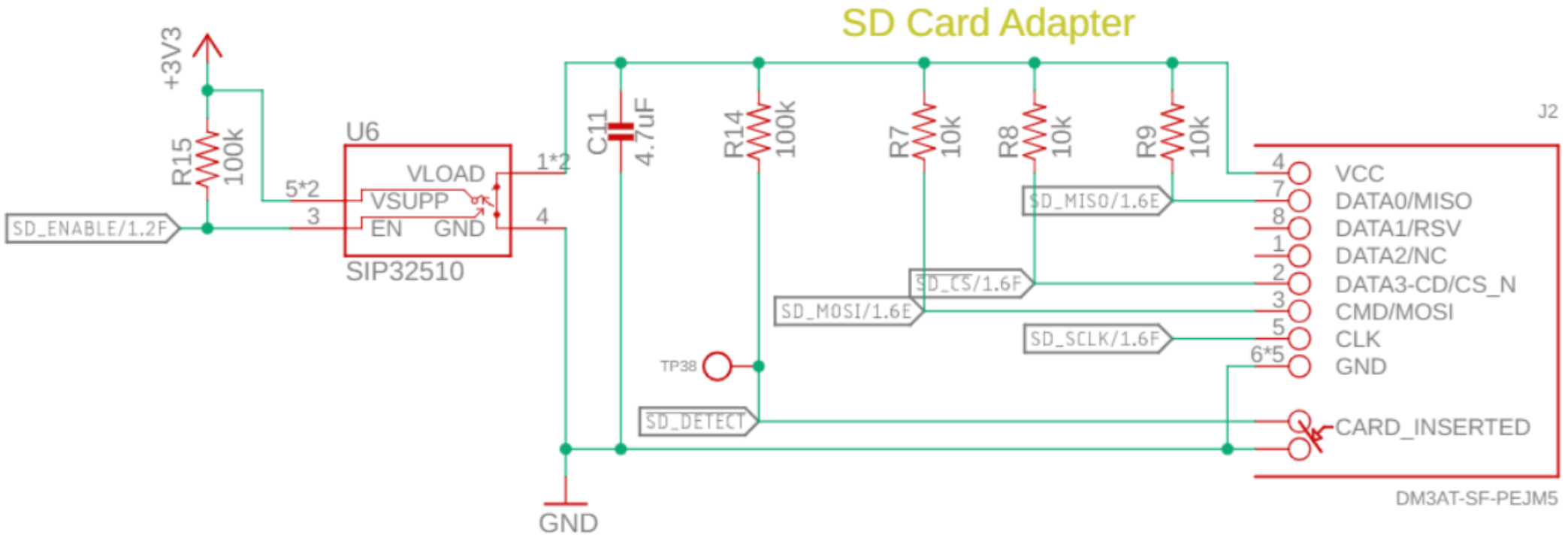
SD cards

- "Secure Digital" Card
  - Includes various formfactors
  - Flash memory
  - Capacities from 8 MB to 128 TB
    - 512 byte blocks
- Supports 1-bit SPI interface
  - As well as 4-bit SD bus protocol
- Easy to support in embedded systems
  - Cheap but high power



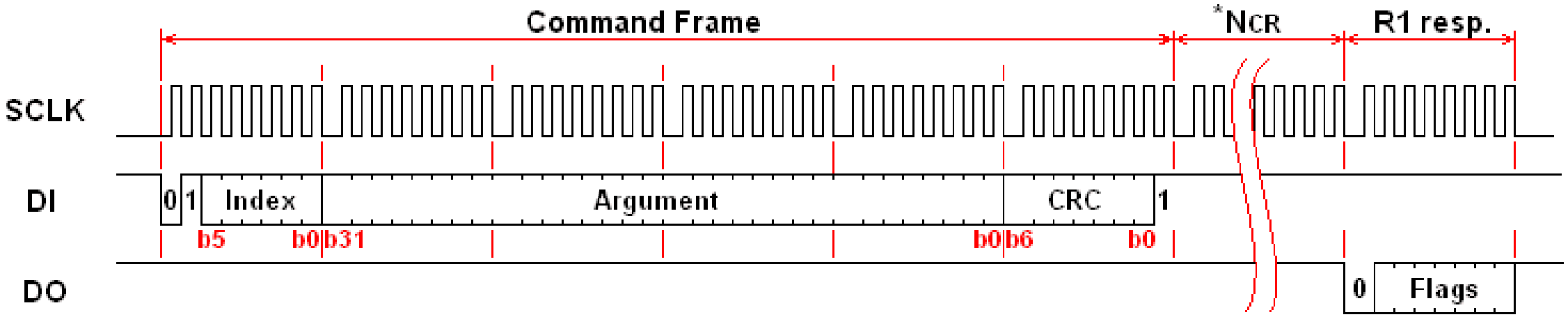
Electrical connections for an SD card

- SD Card connections
  - SPI CIPO, COPI, CS, SCLK
  - Plus a switch to enable/disable the SD card and a detect signal



Controlling the SD card

- Index: 6-bit value of command being sent
- Argument: 32-bit value that may be arguments to commands
- CRC: checks for bit errors
- Response (after delay)

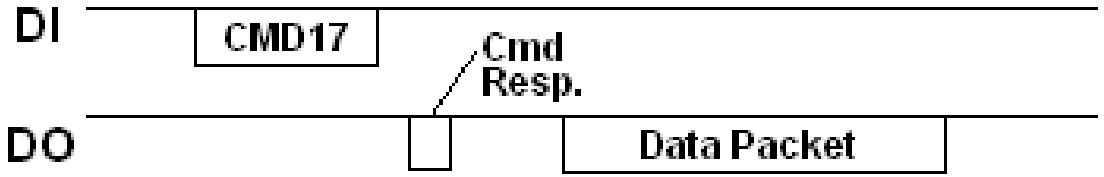


SD card SPI commands

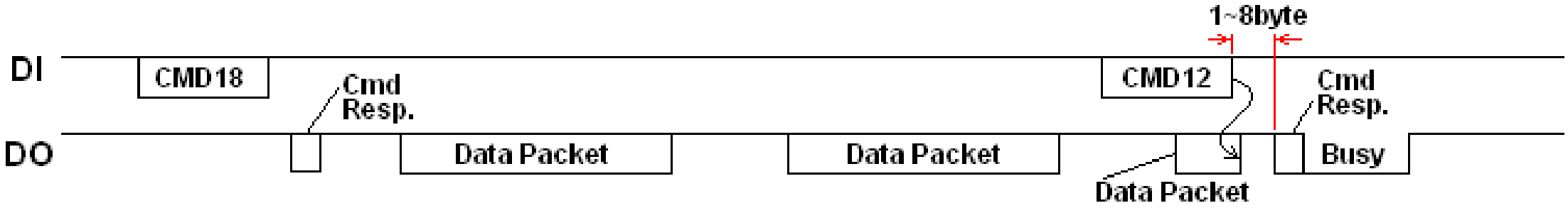
Command Index	Argument	Response	Data	Abbreviation	Description
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate initialization process.
CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Leading command of ACMD<n> command.
CMD58	None(0)	R3	No	READ_OCR	Read Operations Condition Register (OCR). Indicates supported working voltage range.

Reading from the SD card

- Single block read

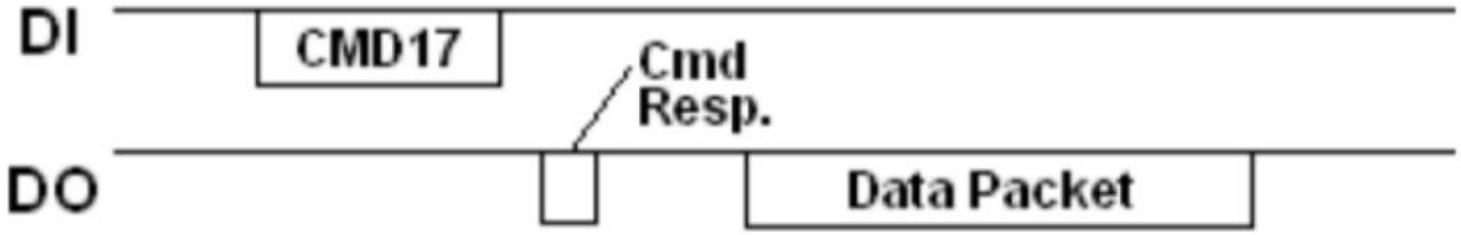


- Multiple block read (CMD12 – Stop Transmission)



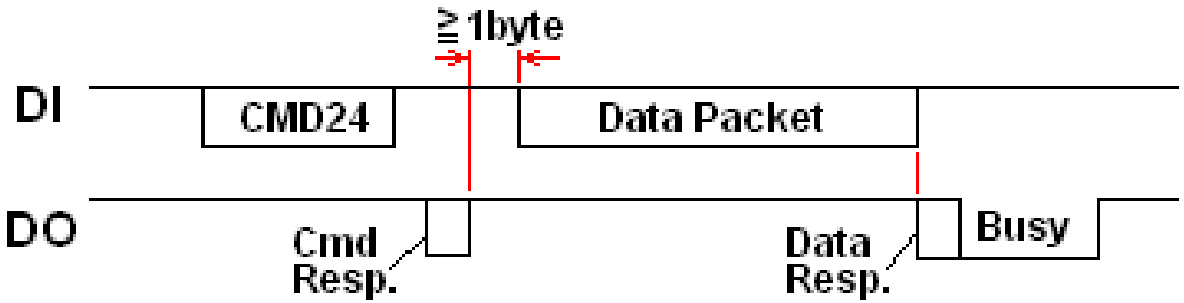
SD card delays can be significant

- Performing a single byte read
  - Almost 300 µs before the SD card *starts* sending data
  - ~200 µs additional time to send the 512 bytes (20 Mbps data, 8 Mbps total)

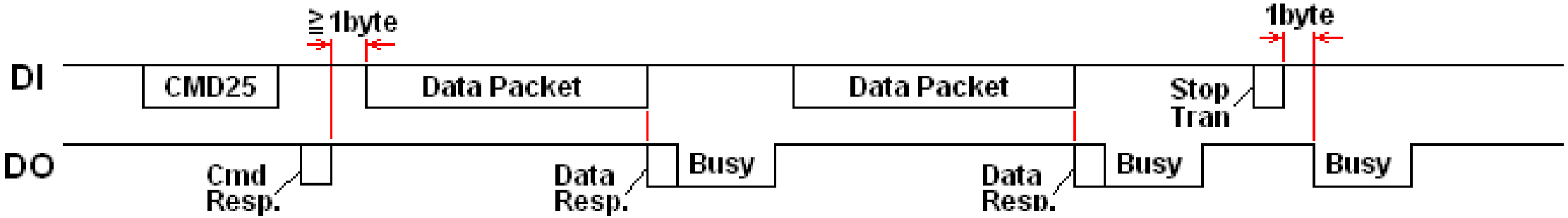


Writing to the SD card

- Single block write

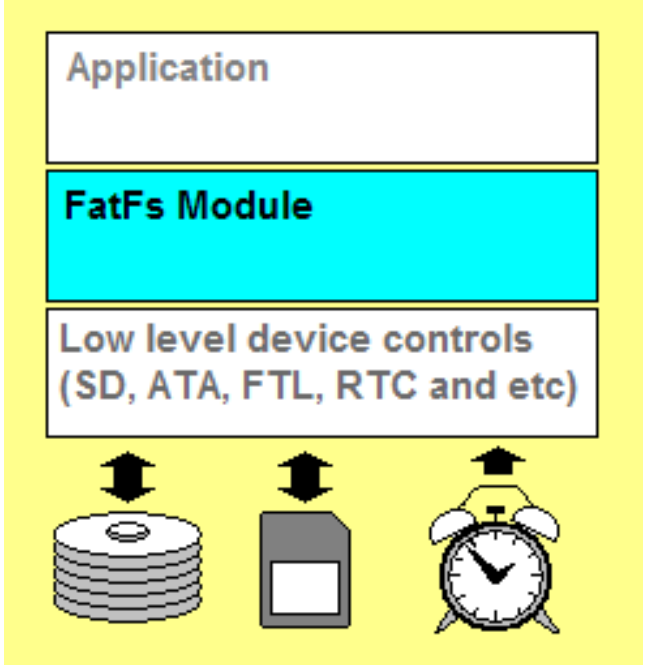


- Multiple block write



Layering a filesystem on top of an SD card

- FatFs library implements the filesystem agnostic of application and storage medium
- Enables the use of file system calls:
  - Open, Close, Read, Seek
- Connects to generic interface for low-level implementation
  - disk_status, disk_init, disk_read, disk_write



Outline

- Embedded Memories
- nRF52 NVMC
- SD Cards