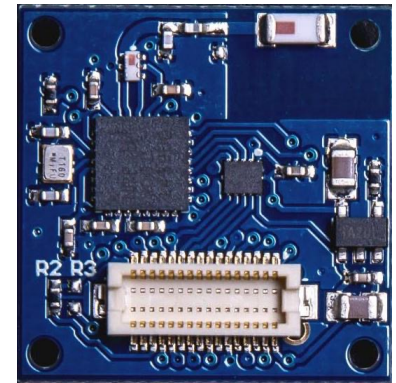
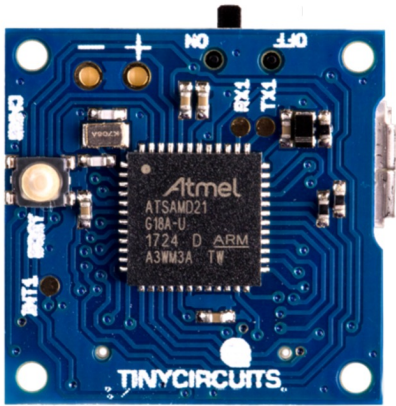


CSE190 Fall 2023

Lecture 4

I/O



Wireless Embedded Systems

Aaron Schulman

Input and Output (I/O)

Input Devices:



keyboard



mouse



other pointing devices



speech

**Direct
manipulation**

Output Devices:

monitor



projector



other displays



audio output



Input Peripherals are common on embedded systems (e.g., sensors)

- Keyboard, mouse, microphone, scanner, video/photo camera, etc.
- Large diversity
 - Many widely differing device types
 - Devices within each type also differs
- Speed
 - varying, often slow access & transfer compared to CPU
 - Some device-types require very fast access & transfer
- Access
 - Sequential VS random
 - read, write, read & write

What operations does software need to perform on I/O peripherals?

1. Get and set parameters
2. Receive and transmit data
3. Enable and disable functions

How can we imagine providing an interface to hardware from software?

1. Specialized CPU instructions (x86 in/out)

Port I/O

- Devices registers mapped onto “ports”; a separate address space



- Use special I/O instructions to read/write ports
- Protected by making I/O instructions available only in kernel/supervisor mode
- Used for example by IBM 360 and successors

How can we imagine providing an interface to hardware from software?

1. Specialized CPU instructions (x86 in/out)
2. Treating devices like they are memory (MMIO)

Memory Mapped IO

- Device registers mapped into regular address space



- Use regular move (assignment) instructions to read/ write a device's hardware "registers"
- Can use memory protection mechanism to protect device registers

MMIO is used for embedded systems

- Why not Ports I/O?
 - special I/O instructions would be instruction set dependent (x86, ARM Thumb, MIPS)
 - Bad if there are many different instruction sets out there
 - Need special hardware to execute and protect instructions
- Memory mapped I/O:
 - Can use all existing memory reference instructions for I/O
 - Can reuse code for reading and writing (e.g., memcpy)
 - Memory protection mechanism allows greater flexibility than protected instructions (protect specific registers)
 - Can reuse memory management / protection hardware to interface with hardware (saves space and power)

Reading and writing with MMIO is not like talking to RAM

- MMIO reads and writes hardware device registers
- Read and write to registers can cause peripherals to begin or end an operation
 - Reading is not a passive operation; it can make the hardware to do something!
 - E.g., Read to clear an interrupt flag, or to advance
 - Writing often starts operations
 - E.g., Send this data over the UART bus

GPIOs are the general digital I/O device

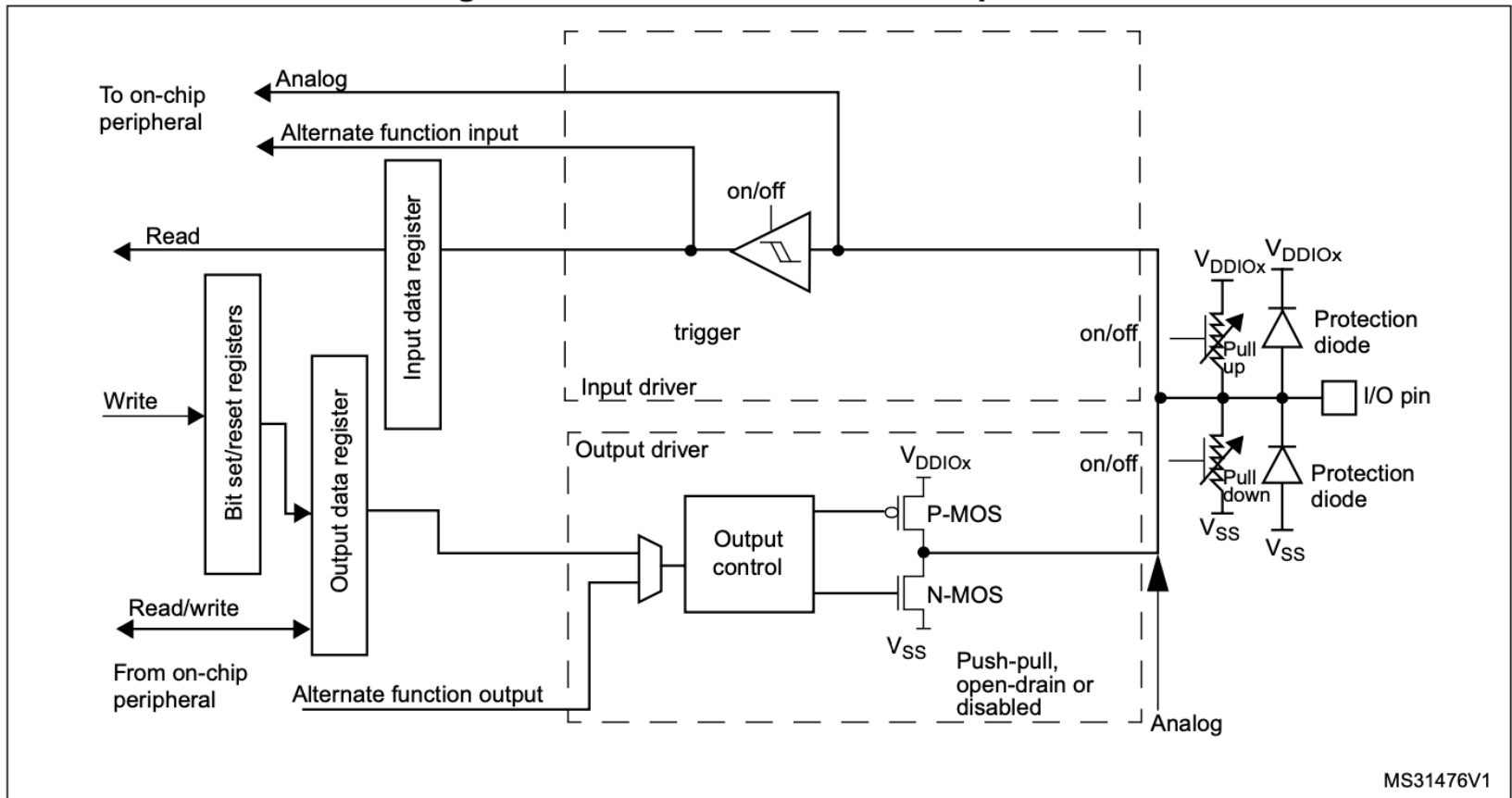
Each GPIO pin **represents one bit in memory**: if the pin is **on it's a 1**, **off it's a 0**

That bit can be an input or an output

- GPIOs can be used to control lights (light on or off), but even more
- Indicating that an event just happened
 - Interrupt the radio to tell it to transmit data
 - Interrupt the CPU to tell it a button was pressed
 - Read pin status to receive configuration messages
- Debugging
 - Did this one part of my code actually execute?
 - Is the timer firing at the interval that I expect it to fire (connect GPIO to oscilloscope)?
 - Why using GPIO?
 - GPIO ops are lightweight

Hardware Schematic of a GPIO pin

Figure 23. Basic structure of an I/O port bit



GPIO Output Configurations

Table 39. Port bit configuration table⁽¹⁾

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]	PUPD(i) [1:0]		I/O configuration		
01	0	SPEED [1:0]	0	0	GP output	PP	
	0		0	1	GP output	PP + PU	
	0		1	0	GP output	PP + PD	
	0		1	1	Reserved		
	1		0	0	0	GP output	OD
	1		0	1	1	GP output	OD + PU
	1		1	0	0	GP output	OD + PD
	1		1	1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]	0	0	AF	PP	
	0		0	1	AF	PP + PU	
	0		1	0	AF	PP + PD	
	0		1	1	Reserved		
	1		0	0	0	AF	OD
	1		0	1	1	AF	OD + PU
	1		1	0	0	AF	OD + PD
	1		1	1	1	Reserved	

GPIO Input configuration

Table 39. Port bit configuration table⁽¹⁾ (continued)

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.