EE 3171 Lecture 6

I/O Programming

Lecture 6 Concepts

The General I/O Situation

And its problems.

Describing the I/O ports of the Tiva C

Explaining the basic parts of I/O programming

Giving examples of I/O device drivers

Understanding simple I/O programs

What Are We Doing Here?

Life inside the chip is dreadfully boring.

We need to talk to the outside world to make life interesting.

Hence, I/O programming.

Hello, World

An embedded system uses input/output devices to connect with the real world.

Input devices get information from the external world, often through sensors.

Output devices

control physical systems or

display information.

Why is I/O Hard?

Four Difficulties

- Analog vs. Digital
- Synchronous vs. Asynchronous
- Speed—Computer vs. Real World
- Throughput vs. Response Time

One Additional Design Decision

- Parallel vs. Serial

Analog vs. Digital

The real world is analog. Computers are digital.

We must convert from one to the other.

Going from digital to analog is easy.

- A digital value always corresponds to an analog value.
- Whether or not it's a meaningful value, we can't guarantee.

Going from analog to digital is harder.

- Most analog values do not correspond exactly to digital values.
- You know this from your DSP class(es).

Synchronous vs. Asynchronous

As you learned in EE 2174,

synchronous means “clocked”

asynchronous means “not clocked”

Computers are clocked.

Well, mostly.

The real world is unclocked.

Again, mostly.

This leads to synchronization issues.

Bottom Line: Asynchronous interfaces are hard.

Speed—Computer vs. Real World

Computers are very fast and getting faster rapidly.

Mechanical devices are much slower and getting faster slowly.

People are slow and not getting faster (or are getting faster *very,very slowly*).

A few characters per second output and maybe tens of characters per second input

Throughput vs. Response Time

Throughput—average amount of data moved over time

- Like bandwidth

Response time—time to do an I/O operation

- Such as inputting a single character

The difficulty here is that response time is not always proportional to throughput.

- A processor might have a very small response time compared to its throughput. (That means it waits a lot.)
- Or vice versa.

Speed—Computer vs. Real World

Clearly there is a very large (many orders of magnitude) mismatch between computers and the real world -- and it's growing.

Should computers:

Slow themselves down to the speed of the real world?

Do something else while waiting for reaction from the real world?

Stay tuned!

Parallel vs. Serial

Parallel means *n* bits at a time, which is:

faster

more expensive (but usually a minor factor)

Serial means one bit at a time, which is:

slower

cheaper

probably older

Parallel vs. Serial

Parallel means *n* bits at a time, which is:

faster

more expensive (but usually a minor factor)

Serial means one bit at a time, which *was*:

slower

cheaper

probably older

Unless it's faster, cheaper and newer

Think: FireWire, USB, InfiniBand, PCI Express (and so on…)

Ports and Drivers

Ports are the parts of a microcomputer that connect to its environment.

A device driver is software that lets higher level software use an I/O device.

> You're probably familiar with device drivers for your home computer.
>
> You shouldn't be though. (My editorial.)

In this lecture we look at ports and programs.

In the next slides we'll look at interfacing—connecting the microcomputer to specific electrical or mechanical gadgets.

Port Addresses

The most common way of accessing ports is called *memory-mapped I/O*.

This means that each I/O port has one or more dedicated “memory” addresses.

Data can then be written to the address for sending to the I/O device (output) or

Data can be read from the address for input to the processor.

These are normal load/store instructions to the normal address space.

The frame of reference for direction is the Tiva.

Input is to the computer from a device.

Output is from the computer to a device.

Other Port Addresses

The other, much less common option goes by lots of names:

Isolated I/O, separate I/O, port-mapped I/O.

The I/O devices have a completely separate address space.

They also only respond to special I/O versions of the load/store instructions.

For example, memory address `0x0000.0016` would be accessed via `ldr r10,0x0000.0016`.

Please note that ALL of this syntax is made up.

I/O address `0x0016` would be accessed via `ldrio r10, 0x0016`.

I/O Hardware Model

I/O *port* —

A composite object that is part of the microcontroller.

It includes:

Set of pins for data transfer

A register attached to the pins

Other registers that define the control protocol for the pins

Command/Status Registers

The command/status register is part of the I/O port.

Usually one bit of it is a “Ready” flag.

The c/s register has its own address, unique from the data port itself.

The c/s register lets the processor

Read status info on the port

e.g. Ready, Error, etc

Write commands to the port

I/O Hardware Model

I/O Device --

A device attached to the I/O port

External to the computer

Generates data and sends it to a port (output) or

Reads data from the port and acts on it (input)

or both

I/O Hardware Model

Input Port

receives data from incoming pins

makes it available to the data bus

generally does not latch the data

Output Port

receives data from the data bus

makes it available on the outgoing pins

usually latches the data for the device

=> value remains constant until a new value is output

Notice that this is again defined from the point of view of the processor.

I/O Memory Mapped Addresses

The Tiva C uses memory-mapped I/O.

All built-in I/O and control registers are mapped into a 16K block from **`0x4000.0000`…** **`0x4400.0000`**

But not necessarily at contiguous addresses

And actually, a whole bunch of locations are simply “reserved”.

Tiva C GPIO Ports

Port A - 8 bits wide.

Special Consideration Pins: 1:0 are UART, 5:2 are SSI0.

Also CAN1, PWM.

Port B - 8 bits wide.

Special Consideration Pins: 3:2 are I^2C0.

Also some Timers, AIN10-11, SSI2, PWM Outputs, CAN0.

Port C - 8 bits wide.

Special Consideration Pins: 3:0 are JTAG.

Also UART, Timers, PWM Outputs.

Tiva C GPIO Ports

Port D - 8 bits wide.

Special Consideration Pins: 7 is the Non-Maskable Interrupt (NMI)

Also AIN7-4, SSI1, SS3, PWM Outputs, Quadrature Encoding, Timers

Port E - 6 bits wide

Special Consideration Pins: None

Also AIN3-0, 9, 8; I^2C2, PWM, CAN0.

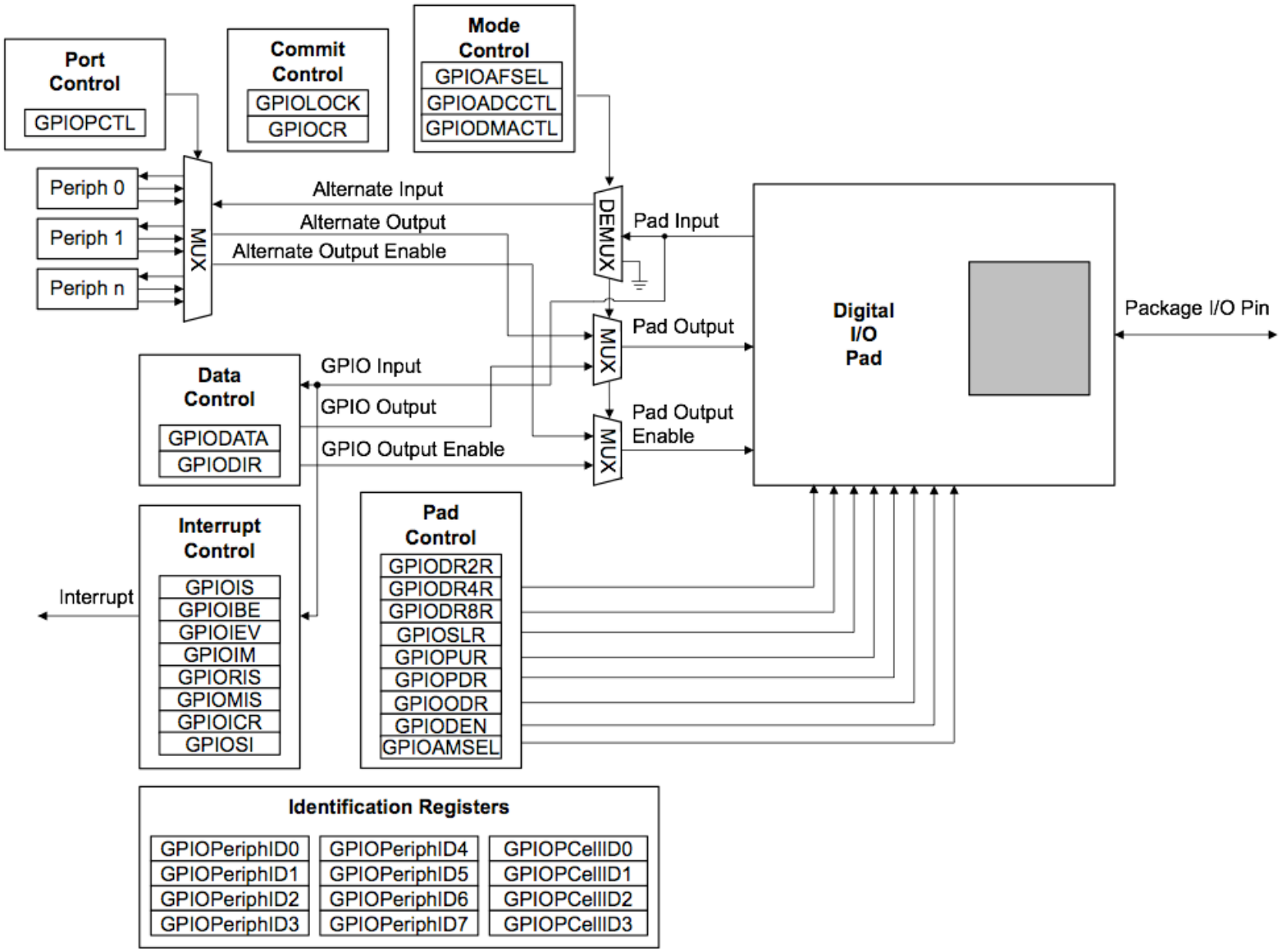
Port F - 5 bits wide

Special Consideration Pins: PF0 is NMI.

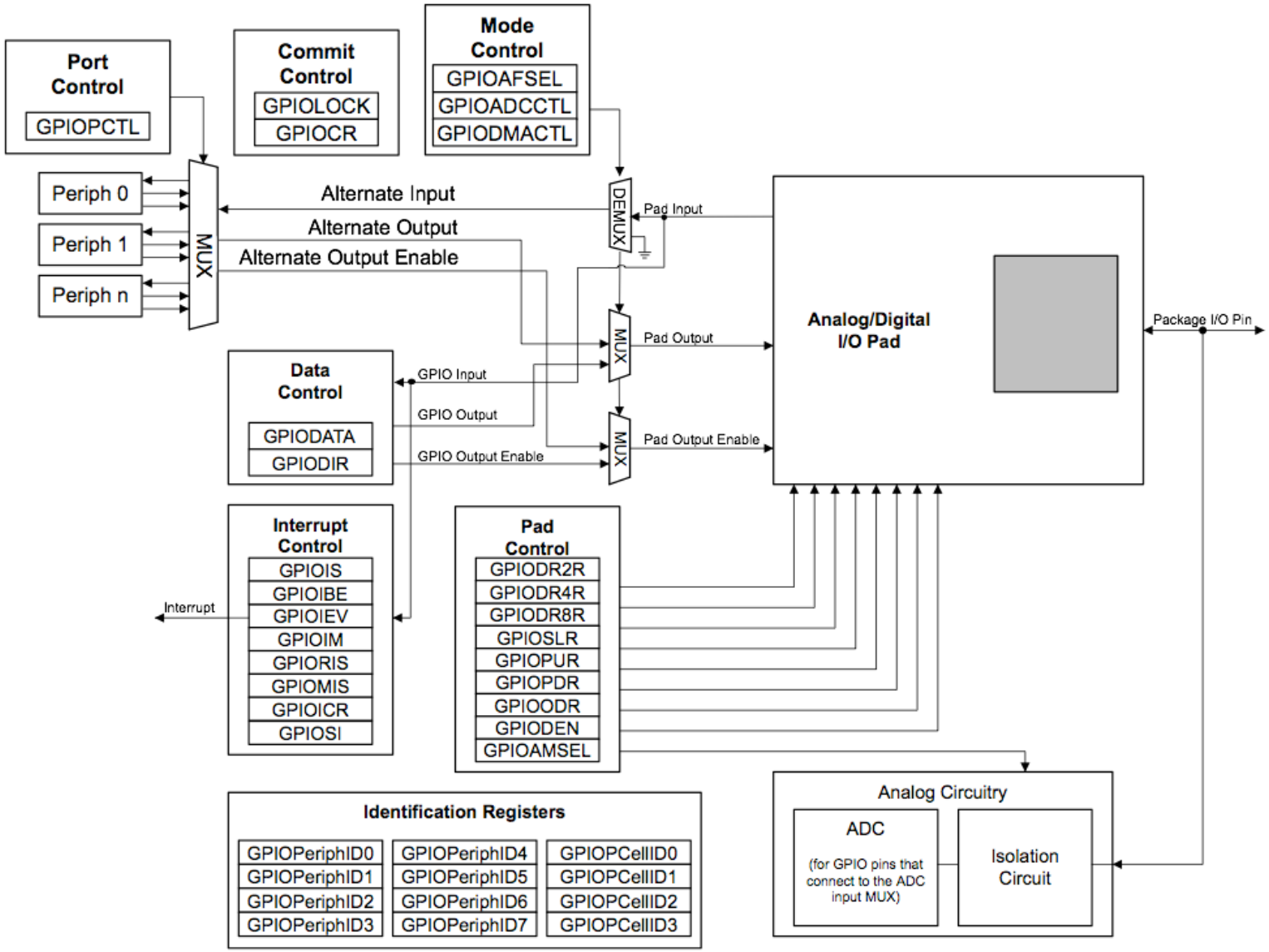
Also CAN0, SSI1, CAN0, PWM Outputs, USB, Timers, QEI, Analog Comparators,

Special Consideration Pins require unlocking in order to change their function. Be VERY careful.

Digital Pins



Analog Pins


Port Control
GPIOPCTL
Commit Control
GPIOLOCK
GPIOCR
Mode Control
GPIOAFSEL
GPIOADCCTL
GPIODMACTL
Periph 0
Periph 1
Periph n
MUX
Alternate Input
Alternate Output
Alternate Output Enable
DEMUX
Pad Input
MUX
Pad Output
MUX
Pad Output Enable
Analog/Digital I/O Pad
Package I/O Pin
Data Control
GPIODATA
GPIODIR
GPIO Input
GPIO Output
GPIO Output Enable
Interrupt Control
GPIOIS
GPIOIBE
GPIOIEV
GPIOIM
GPIORIS
GPIOMIS
GPIOICR
GPIOSI
Interrupt
Pad Control
GPIODR2R
GPIODR4R
GPIODR8R
GPIOSLR
GPIOPUR
GPIOPDR
GPIOODR
GPIODEN
GPIOAMSEL
Identification Registers
GPIOPeriphID0
GPIOPeriphID1
GPIOPeriphID2
GPIOPeriphID3
GPIOPeriphID4
GPIOPeriphID5
GPIOPeriphID6
GPIOPeriphID7
GPIOPCellID0
GPIOPCellID1
GPIOPCellID2
GPIOPCellID3
Analog Circuitry
ADC
(for GPIO pins that connect to the ADC input MUX)
Isolation Circuit

GPIO Registers

Each port here clearly has a bunch of registers

- 25 for each port

Let's look at each register's "big picture" purpose

- Not a bit-by-bit description

Base Addresses:

- GPIO Port A: 0x4000.4000
- GPIO Port B: 0x4000.5000
- GPIO Port C: 0x4000.6000
- GPIO Port D: 0x4000.7000
- GPIO Port E: 0x4002.4000
- GPIO Port F: 0x4002.5000

GPIO Registers

`GPIODATA` (offset `0x000`)

Read and write data from/to the pins

`GPIODIR` (offset `0x400`)

Change the direction of the tristate pins (1 = output, 0 = input)

`GPIOIS` (offset `0x404`)

Change the trigger for interrupts to either edge-sensitive (0) or level-sensitive (1)

`GPIOIBE` (offset `0x408`)

Configure the port to interrupt on both edges (1) or a single edge (0)

`GPIOIEV` (offset `0x40C`)

Interrupts on rising edge (1) or falling edge (0) on pins

GPIO Registers

`GPIOIM` (offset `0x410`)

Ignore potential interrupts (0) or process the interrupt (1)

`GPIORIS` (offset `0x414`)

Indicates if an interrupt has been triggered on a pin (1) or not (0)

`GPIOMIS` (offset `0x418`)

Indicates if an interrupt has been masked (0) or has occurred (1)

`GPIOICR` (offset `0x41C`)

Write a 1 to clear an interrupt (declare it serviced)

`GPIOAFSEL` (offset `0x420`)

Configure the pin to be GPIO (0) or some other peripheral function (1)

GPIO Registers

`GPIODR2R` (offset `0x500`)

Drive a 2 mA output current (1) or some other current (0)

`GPIODR4R` (offset `0x504`)

Drive a 4 mA output current (1) or some other current (0)

`GPIODR8R` (offset `0x508`)

Drive an 8 mA output current (1) or some other current (0)

`GPIOODR` (offset `0x50C`)

Configure an output pin to function as open drain (1) or not (0)

`GPIOPUR` (offset `0x510`)

Configure a pin to have a weak pull-up resistor (1) or not (0)

One of these registers needs to configure every pin. By default, currents are all 2 mA.

No effect on input pins.

This register is locked with the `GPIOCR`.

GPIO Registers

`GPIOPDR` (offset `0x514`)

This register is locked with the GPIOCR.

Configure a pin to have a weak pull-down resistor (1) or not (0)

`GPIOSLR` (offset `0x518`)

Configure slew rate control for a pin (1) or disable it (0)

Can only use with 8 mA drive.

`GPIODEN` (offset `0x51C`)

Configure a pin to recognize digital inputs (1) or not (0)

All pins start up completely disabled.

`GPIOLOCK` (offset `0x520`)

Enables write access to the `GPIOCR` register. Must write `0x4C4F.434B`. All other values lock things up.

GPIO Registers

`GPIOPCR` (offset `0x524`)

Indicates whether writes to `GPIOAFSEL`, `GPIOPUR`, `GPIOPDR` or `GPIODEN` will be committed (1) or not (0)

`GPIOAMSEL` (offset `0x528`)

Enables (1) or disables (0) the analog functionality of a pin

`GPIOPCTL` (offset `0x52C`)

Used in conjunction with `GPIOAFSEL` to specify which alternate function is actually enabled

`GPIOADCCTL` (offset `0x530`)

Enables the pin to act as a trigger for ADC conversions (1) or not (0)

GPIO Registers

`GPIODMACTL` (offset `0x534`)

Enables the pin to act as a trigger for DMA operations (1) or not (0)

And then a whole host of ID registers.

Synchronization Woes

I/O Speed

I/O devices are typically much slower than processors (integer factor)

Can't read until the data ready in device

Can't write until device is ready to receive data

And the device is usually busy processing the previous data

Processor must synchronize with I/O

device sets a 1-bit flag when ready

Creatively named the "Ready" flag.

processor waits for flag before accessing the port

flag is cleared when processor accesses the port

Note that none of our ports had a "ready" flag.

We did have the Interrupt Status registers.

Lecture 6 Summary

I/O is important and fairly interesting, but hard.

For lots of reasons

Parallel vs. serial

Memory mapped vs. isolated I/O

Control and Status registers + Data register + pins = I/O Port

A whole bunch of ports on the Tiva C, all with some level of complexity or uniqueness.