

ECE445M/ECE380L.12

Embedded and Real-Time Systems/ Real-Time Operating Systems

Lecture 1: Introduction, TM4C123, Software Development

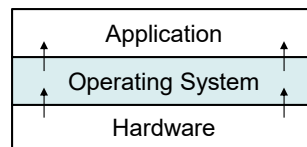
Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

1

Operating System

- Manage, provide abstractions for hardware



- CPU -> Parallel tasks, threads (Labs 2-3)
- Storage -> Files (Lab 4)
- Memory -> Heap, processes (Lab 5)
- I/O -> Hardware-specific (Lab 6)

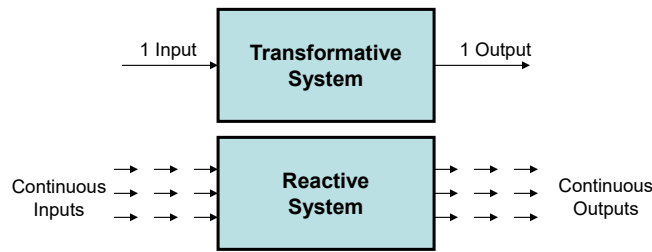
Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

2

Reactive Systems

- Interactive, event-driven
 - Parallel input/output events/streams
 - Some are real-time (limits on reaction time)



➤ Embedded systems, I/O, GUIs

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

3

Parallel Processing (1)

- Distributed systems
 - Multiple computers, separate memory, I/O or network link
 - Simultaneous execution of two or more software tasks
 - E.g. Lab 6 (CAN), Internet

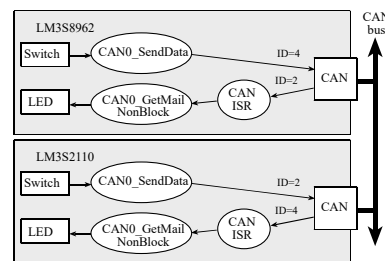


Figure 9.6. Simple CAN network.

- Multi-processing
 - Multiple processors, shared memory
 - Simultaneous execution of two or more software programs (tasks)
 - E.g. multicore CPU, GPU

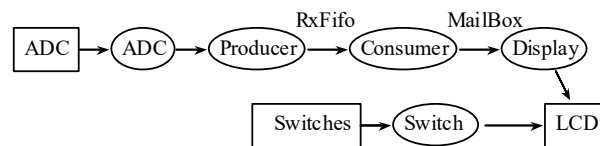
Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

4

Parallel Processing (2)

- Multi-threading / Multi-tasking
 - Single processor/core
(in a distributed and/or multi-processor system)
 - One foreground and multiple background threads (interrupt-driven)
 - Multiple foreground threads using a thread scheduler (*operating system, OS*)



- Multiple independent programs (tasks)

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

5

Class Setup

- Class web page
 - http://www.ece.utexas.edu/~gerstl/ece445m_s23/
- Canvas
 - Announcements, lab report upload, grades
- Communication
 - Piazza for general class discussion
 - Gradescope for exam grading and feedback
 - Mailing list: s23_ece445m@utlists.utexas.edu (all Professor & TAs)
- Office hours
 - Prof. (EER 5.882), TAs (EER 1.806):
See online posted Weekly Schedule

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

6

Course Overview

- Labs (50%)
 - Lab 1: UART, display, ADC (ECE445L review)
 - Lab 2: RTOS kernel
 - Lab 3: Scheduling
 - Lab 4: File & disk I/O
 - Lab 5: Memory, process loader
 - Lab 6: Networking, robot interfaces
 - Lab 7: Robot racing (last week of classes)
- Exams (50%)
 - Midterm: Thu, 3/2, in class (tentative)
 - Final: Fri, 4/28, 3:30-5:30pm (regularly scheduled)
- Graduate project (20%)
 - Independent RTOS project (proposal by mid February)

Teams of 2

Teams of 3-5

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

7

Announcements

- Labs
 - TA demos & partner selection this week
 - Lab 1 starts next week
- Equipment to get (recommended: each student)
 - TM4C123 LaunchPad board (Lab 1)
 - ST7735 LCD display (Lab 1) – requires soldering
 - ESP-01 Wifi module (Lab 6) – ESP-01S required clipping pin
 - Optional: USB logic analyzer (Lab 2), multimeter (Lab 6/7)
- We will provide (on loan, return at end of class)
 - Sensor & motor board, robot parts
- Setup laptop to be able to work independently
 - ARM environment: Keil μ Vision 5.x (not 4.7x!) or gcc
 - Starter and driver code: ValvanoWareTM4C123v5
 - Putty or HyperTerminal (serial terminal)

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

8

Lab Access

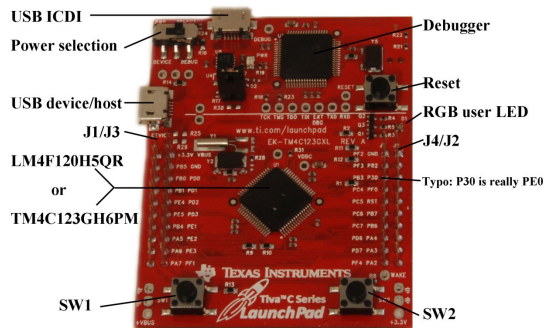
- Lab space: EER 1.806 (Embedded Systems Lab)
 - Used for checkouts/demos, TA office hours
 - PCs, soldering stations, scopes, logic analyzers, ...
 - Shared with ECE445L & Senior Design
- MakerSpace
 - Free for any student
 - Laser cutters, PCB mill, 3D printers, ...

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

9

Texas Instruments LaunchPad



Debug connections:

- Female-male connectors (attach to top)
 - <https://www.adafruit.com/products/826>
 - DigiKey H1505-ND (Hirose DF11-2428SCA)

Reference material

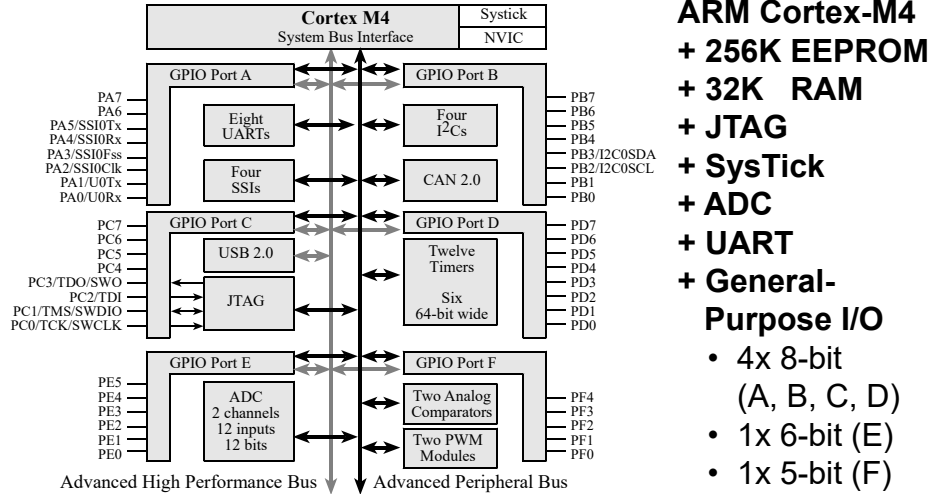
- http://www.ece.utexas.edu/~gerst/ece445m_s23/resources.html
- <http://www.ece.utexas.edu/~valvano/arm/> (starter files, example projects)
- TI manuals
 - <http://www.ti.com/lit/ug/spmu296/spmu296.pdf> (LaunchPad User's Guide)
 - <http://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf> (TM4C123 data sheet)
 - <http://www.ti.com.cn/cn/lit/ug/spmu159a/spmu159a.pdf> (Cortex-M4 instruction set)
- ARM manuals
 - https://static.docs.arm.com/ddi0439/b/DDI0439B_cortex_m4_r0p0_trm.pdf (Cortex-M4 technical reference)

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

10

Texas Instruments TM4C123



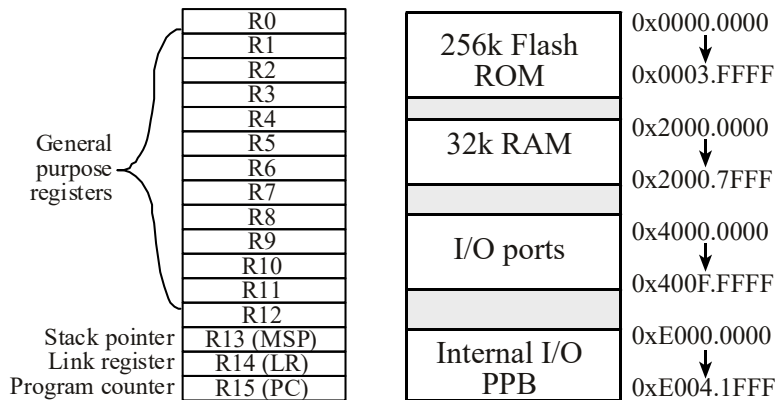
- ARM Cortex-M4**
- + 256K EEPROM**
- + 32K RAM**
- + JTAG**
- + SysTick**
- + ADC**
- + UART**
- + General-Purpose I/O**
 - 4x 8-bit (A, B, C, D)
 - 1x 6-bit (E)
 - 1x 5-bit (F)

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

11

Registers, Memory-map



<u>Condition Code Bits</u>		<u>Indicates</u>
N	negative	Result is negative
Z	zero	Result is zero
V	overflow	Signed overflow
C	carry	Unsigned overflow

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

12

ARM Thumb Instruction Set

- Arithmetic/Logic
 - AND R1, R2, R3 ; register
 - EOR/ORR R1,R2,#1 ; immediate, 12-bit
 - LSR R1,R1,#4 ; logic shift
 - ADD{S} R1,R2,R3, LSL #2 ; $R1=R2+R3*4$ {set condition codes}
 - SUB{S} R1,R3, ASR #2 ; $R1=R1+R3/4$ {set condition codes}
 - CMP R2,R3 ; compare
- Data movement
 - MOV R0,#100 ; immediate
 - ADR R0,Label ; load address
 - LDR R0,=Label ; uses PC-relative
 - STR{H} R1,[R0] ; indexed {16-bit halfword}
 - LDR{{S}H} R1,[R0,#n] ; offset indexed {{signed} halfword}
- Control
 - B Target ; unconditional
 - BEQ/BNE Target ; (in)quality
 - BLO/BLS/BHI/BHS Target ; unsigned <,<=,>,>=
 - BLT/BLE/BGT/BGE Target ; signed <,<=,>,>=

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

13

Function calls

```
void delay (int cnt){
  while (cnt--);
}
```

```
void main(void) {
  delay(10);
}
```

```
delay
SUB    R0,R0,#0x01
BNE    delay
BX     LR
```

```
main
MOV    R0,#0x0A
BL     delay
```

AAPCS: Parameters in R0-R3, return in R0

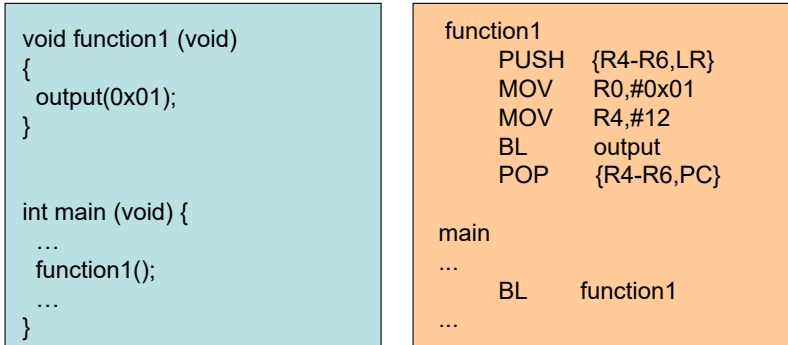
Follow the link register LR

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

14

Stack

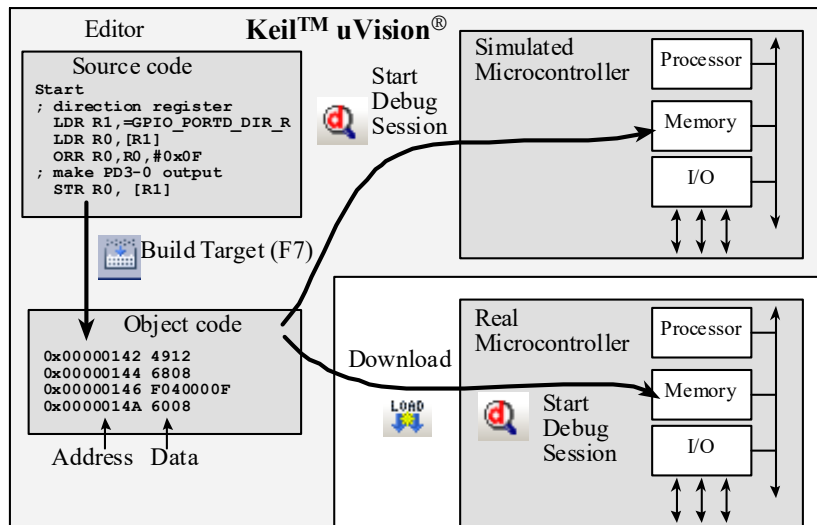


R4-R11 must be saved

Draw a stack picture

The accesses happen in order of decreasing (push)/increasing (pop) register numbers, with the lowest numbered register using the lowest memory address (top of stack) and the highest number register using the highest memory address

SW Development Environment



General-Purpose I/O (GPIO)

Address	7	6	5	4	3	2	1	0	Name
40F.E608	-	-	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGCGPIO_R
xxxx.x3FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTx_DATA_R
xxxx.x400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTx_DIR_R
xxxx.x420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTx_AFSEL_R
xxxx.x510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTx_PUR_R
xxxx.x51C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTx_DEN_R

- **Initialization**
 1. Turn on clock in **SYSCTL_RCGCGPIO_R**
 2. Wait two bus cycles (two **NOP** instructions)
 3. Set **DIR** to 1 for output or 0 for input
 4. Clear **AFSEL** & **AMSEL** bits to 0 to select regular I/O
 5. Set **DEN** bits to 1 to enable data pins
- **Input/output from pin**
 6. Read/write **GPIO_PORTx_DATA_R**

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

17

Bit-Specific Port I/O

- Bit-specific addressing is used to access port data register
 - Define address offset as $4 \cdot 2^b$, where **b** is the selected bit position
 - 256 possible bit combinations (0-8)
 - Add offsets for each bit selected to base address for the port
 - Other bits masked during access
 - **DATA_R @ base+\$3FC** equals all bits

If we wish to access bit	Constant
7	0x0200
6	0x0100
5	0x0080
4	0x0040
3	0x0020
2	0x0010
1	0x0008
0	0x0004

Example: PF4 and PF0

Port F = 0x4005.D000

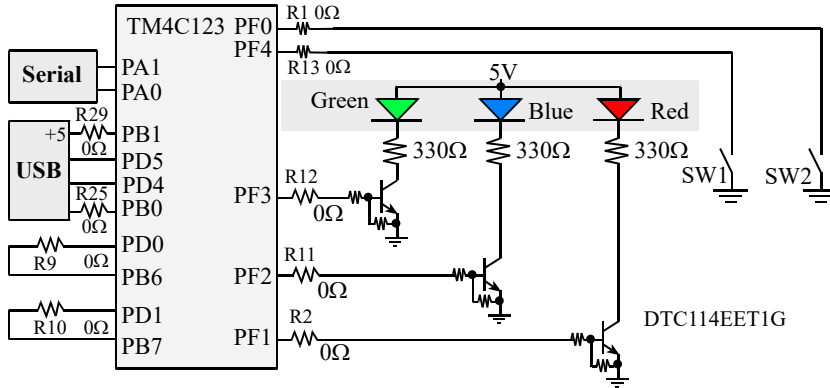
 $0x4005.D000 + 0x0004 + 0x0040$ $= 0x4005.D044$ *Provides friendly and atomic access to port pins*

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

18

LaunchPad Switches and LEDs



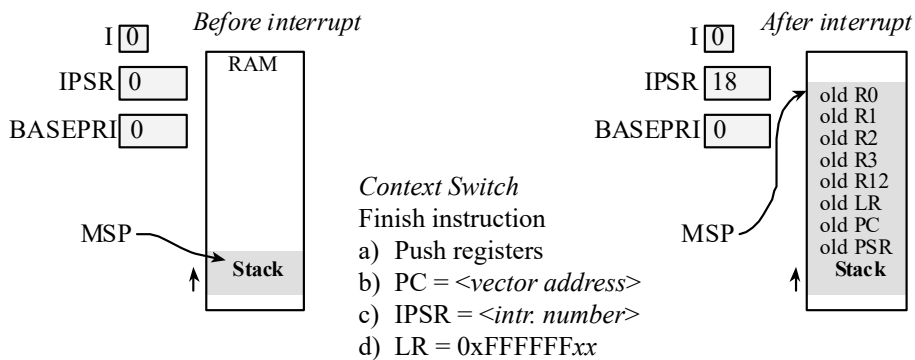
- The switches on the LaunchPad
 - Negative logic, require internal pull-up (set bits in PUR)
- The PF3-1 LEDs are positive logic

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

19

Interrupts **



** More details in Lecture 2

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

20

Interrupt Vectors

Vector address	Number	IRQ	ISR name in Startup.s	NVIC	Priority bits
0x00000038	14	-2	PendSV_Handler	NVIC_SYS_PRI3_R	23-21
0x0000003C	15	-1	SysTick_Handler	NVIC_SYS_PRI3_R	31-29
0x00000040	16	0	GPIOPortA_Handler	NVIC_PRI0_R	7-5
0x00000044	17	1	GPIOPortB_Handler	NVIC_PRI0_R	15-13
0x00000048	18	2	GPIOPortC_Handler	NVIC_PRI0_R	23-21
0x0000004C	19	3	GPIOPortD_Handler	NVIC_PRI0_R	31-29
0x00000050	20	4	GPIOPortE_Handler	NVIC_PRI1_R	7-5
0x00000054	21	5	UART0_Handler	NVIC_PRI1_R	15-13
0x00000058	22	6	UART1_Handler	NVIC_PRI1_R	23-21
0x0000005C	23	7	SSI0_Handler	NVIC_PRI1_R	31-29
0x00000060	24	8	I2C0_Handler	NVIC_PRI2_R	7-5
0x00000064	25	9	PWMFault_Handler	NVIC_PRI2_R	15-13
0x00000068	26	10	PWM0_Handler	NVIC_PRI2_R	23-21
0x0000006C	27	11	PWM1_Handler	NVIC_PRI2_R	31-29
0x00000070	28	12	PWM2_Handler	NVIC_PRI3_R	7-5
0x00000074	29	13	Quadrature0_Handler	NVIC_PRI3_R	15-13
0x00000078	30	14	ADC0_Handler	NVIC_PRI3_R	23-21
0x0000007C	31	15	ADC1_Handler	NVIC_PRI3_R	31-29
0x00000080	32	16	ADC2_Handler	NVIC_PRI4_R	7-5
0x00000084	33	17	ADC3_Handler	NVIC_PRI4_R	15-13
0x00000088	34	18	WDT_Handler	NVIC_PRI4_R	23-21
0x0000008C	35	19	Timer0A_Handler	NVIC_PRI4_R	31-29
0x00000090	36	20	Timer0B_Handler	NVIC_PRI5_R	7-5
0x00000094	37	21	Timer1A_Handler	NVIC_PRI5_R	15-13
0x00000098	38	22	Timer1B_Handler	NVIC_PRI5_R	23-21
0x0000009C	39	23	Timer2A_Handler	NVIC_PRI5_R	31-29
0x000000A0	40	24	Timer2B_Handler	NVIC_PRI6_R	7-5
0x000000A4	41	25	Comp0_Handler	NVIC_PRI6_R	15-13
0x000000A8	42	26	Comp1_Handler	NVIC_PRI6_R	23-21
0x000000AC	43	27	Comp2_Handler	NVIC_PRI6_R	31-29
0x000000B0	44	28	SysCtl_Handler	NVIC_PRI7_R	7-5
0x000000B4	45	29	FlashCtl_Handler	NVIC_PRI7_R	15-13
0x000000B8	46	30	GPIOPortF_Handler	NVIC_PRI7_R	23-21
0x000000BC	47	31	GPIOPortG_Handler	NVIC_PRI7_R	31-29
0x000000C0	48	32	GPIOPortH_Handler	NVIC_PRI8_R	7-5
0x000000C4	49	33	UART2_Handler	NVIC_PRI8_R	15-13
0x000000C8	50	34	SSI1_Handler	NVIC_PRI8_R	23-21
0x000000CC	51	35	Timer3A_Handler	NVIC_PRI8_R	31-29
0x000000D0	52	36	Timer3B_Handler	NVIC_PRI9_R	7-5
0x000000D4	53	37	I2C1_Handler	NVIC_PRI9_R	15-13
0x000000D8	54	38	Quadrature1_Handler	NVIC_PRI9_R	23-21
0x000000DC	55	39	CAN0_Handler	NVIC_PRI9_R	31-29
0x000000E0	56	40	CAN1_Handler	NVIC_PRI9_R	7-5
0x000000E4	57	41	CAN2_Handler	NVIC_PRI10_R	15-13
0x000000E8	58	42	Ethernet_Handler	NVIC_PRI10_R	23-21
0x000000EC	59	43	Hibernate_Handler	NVIC_PRI10_R	31-29
0x000000F0	60	44	USB0_Handler	NVIC_PRI11_R	7-5
0x000000F4	61	45	PWM3_Handler	NVIC_PRI11_R	15-13
0x000000F8	62	46	uDMA_Handler	NVIC_PRI11_R	23-21
0x000000FC	63	47	uDMA_Error	NVIC_PRI11_R	31-29

Startup.s

Lecture 1

21

Nested Vectored Interrupt Controller (NVIC)

- Priorities (global level in **BASEPRI**)

Address	31-29	23-21	15-13	7-5	Name
0xE00E400	GPIO Port D	GPIO Port C	GPIO Port B	GPIO Port A	NVIC PRI0 R
0xE00E404	SSI0, Rx Tx	UART1, Rx Tx	UART0, Rx Tx	GPIO Port E	NVIC PRI1 R
0xE00E408	PWM Gen 1	PWM Gen 0	PWM Fault	I2C0	NVIC PRI2 R
0xE00E40C	ADC Seq 1	ADC Seq 0	Quad Encoder	PWM Gen 2	NVIC PRI3 R
0xE00E410	Timer 0A	Watchdog	ADC Seq 3	ADC Seq 2	NVIC PRI4 R
0xE00E414	Timer 2A	Timer 1B	Timer 1A	Timer 0B	NVIC PRI5 R
0xE00E418	Comp 2	Comp 1	Comp 0	Timer 2B	NVIC PRI6 R
0xE00E41C	GPIO Port G	GPIO Port F	Flash Control	System Control	NVIC PRI7 R
0xE00E420	Timer 3A	SSI1, Rx Tx	UART2, Rx Tx	GPIO Port H	NVIC PRI8 R
0xE00E424	CAN0	Quad Encoder 1	I2C1	Timer 3B	NVIC PRI9 R
0xE00E428	Hibernate	Ethernet	CAN2	CAN1	NVIC PRI10 R
0xE00E42C	uDMA Error	uDMA Soft Tfr	PWM Gen 3	USB0	NVIC PRI11 R
0xE00ED20	SysTick	PendSV	--	Debug	NVIC SYS PRI3 R

- Interrupt enable
 - **NVIC_EN0_R** and **NVIC_EN1_R**

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

22

SysTick Timer

Address	31-24	23-17	16	15-3	2	1	0	Name
\$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
\$E000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
\$E000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

- **Timer/Counter**
 - 24-bit counter decrements at bus clock frequency
 - With 80 MHz bus clock, decrements every 12.5 ns
 - Counting is from $n \rightarrow 0$
 - Setting n appropriately will make the counter a modulo $n+1$ counter:
 - $\text{next_value} = (\text{current_value} - 1) \bmod (n+1)$
 - Sequence: $n, n-1, n-2, n-3, \dots, 2, 1, 0, n, n-1, \dots$
- **Initialization**
 1. Clear *ENABLE* to stop counter
 2. Specify the *RELOAD* value
 3. Clear the counter via *NVIC_ST_CURRENT_R*
 4. Set *CLK_SRC*=1 and specify interrupt action via *INTEN*

PeriodicSysTickInts_4C123

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

23

System Tick

```
#define GPIO_PD0 (*(volatile unsigned long *) 0x40007004)
```

```
void SysTick_Handler(void) {
    GPIO_PD0 = GPIO_PD0^0x01;
    Counts = Counts + 1;
}
```

```
void main(void){
    ...
    SysTick_Init(50000);    // period: 1ms = 50,000 clock ticks
                          // assuming 50 MHz bus clock
    ...
}
```

Reset debugger:
- stop in ISR and
- single step through ISR
- look at assembly code

PeriodicSysTickInts_4C123

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

24

Other Peripherals (Lab 1)

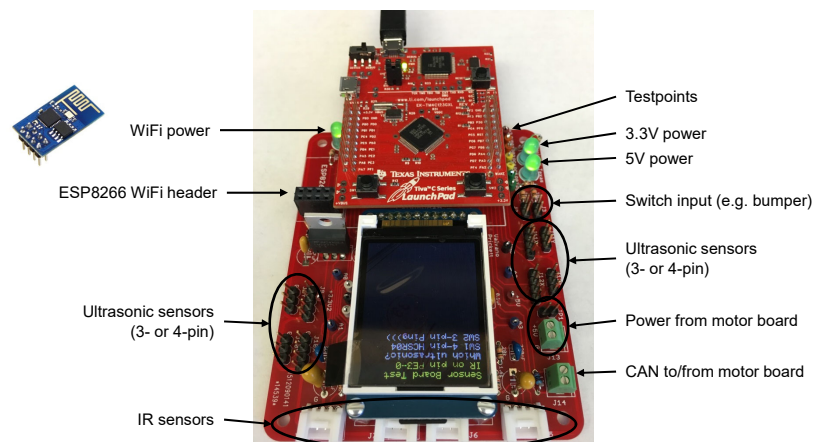
- Serial I/O (UART)
 - `UARTInts_4C123`
- Analog-to-digital conversion (ADC)
 - `ADCSWTrigger_4C123`
- LCD display (ST7735) via GPIO
 - `ST7735_4C123`
- General-purpose timers
 - `PeriodicTimer2AInts_4C123` (Timer2A)

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

25

Sensor Board (Labs 1-7)



- **Reference material**

- Schematic: http://www.ece.utexas.edu/~gerstl/ee445m_s16/resources/Robot_Sensor_v3.pdf
- PCB layout: http://www.ece.utexas.edu/~gerstl/ee445m_s16/resources/sensor_top3.png

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

26

Terminal I/O

- UART0 connected to USB serial port
- How to do terminal input/output?
 1. Write your own, like `UARTInts`
 2. Use `sprintf()` to create strings then output string
 3. Retarget and link to standard library
 - Output using stdlib function `printf()`
 - Input using stdlib function `getchar()`

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

27

Stdio Retargeting

```
int fputc(int ch, FILE *f){
    UART_OutChar(ch);
    return (1);
}

int fgetc (FILE *f){
    return (UART_InChar());
}

int ferror(FILE *f){
    /* Your implementation of ferror */
    return EOF;
}
```

retarget.c in C:\Keil_v5\ARM\Startup

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

28

Starter Code and Driver Lib

- How much code to reuse?
 - Starter files (ValvanoWare) & **driverlib** (TI)
 - Will have fewer bugs than any you or I write
 - You will have to certify all code working in parallel environment (critical sections)
 - Most students will want to fit code into 32k
 - All students must understand everything

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

29

Lab 1 Starter Project

- **RTOS_Lab1_Interpreter** project
 - Application/user code (**Lab1.c**)
- **RTOS_Labs_common** code
 - Command line (**Interpreter.c**) to be developed
 - OS kernel (**OS.c**, **osasm.s**) developed in labs
 - Drivers to be developed (**ST7735.c**, **ADC.c**)
 - Custom copies of standard drivers (**UART0Int.c**)
- Standard drivers (from **inc** directory)
 - Timers, ADC, ...

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

30

Lab Logistics

- Lab 1
 - Week of 1/9: TA demos, partnering & boards
 - Week of 1/16: Lab 1 Prep
 - Week of 1/23: Lab 1 Demo/Report, Lab 2 Prep
- Lab submissions
 - Preparation: software design (oral/paper)
 - Demo: software implementation (Github)
 - Submission and repository info will be mailed
 - Report: software documentation (Canvas)

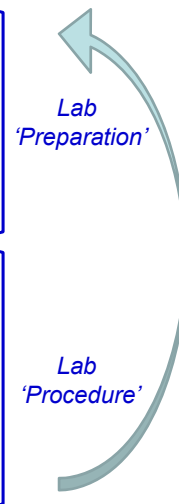
Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

31

Development Process

1. Analyze (Ask)
 - Requirements, specifications
2. Design (Think)
 - Dataflow/call graphs, flowcharts, OO
3. Implement (Do)
 - Code, coding styles, documentation
4. Debug and test (Check)
 - Verification (correctness)
 - Validation (performance)



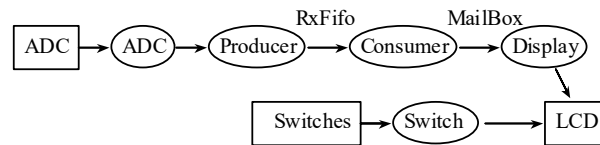
Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

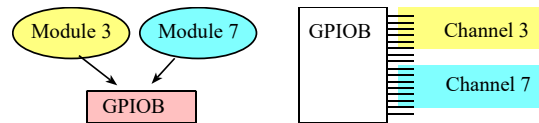
32

Software Design

- Dataflow graphs
 - Parallel or sequential execution



- Call graphs
 - Use to identify potential conflicts



Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

33

Modular Programming

- Encapsulation (private definitions, .c file)
 - Information hiding
 - Reduce coupling
- Abstraction (public declarations, .h file)
 - Well-defined external interfaces
 - Separate mechanism from policy

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

34

Object-Oriented Programming (OOP)

- Elevate modules into first-class citizens
 - Encapsulation of data & code
 - Member variables and functions (*methods*)
 - Dynamic vs. static scope & lifetime
 - Blueprint (*class*) and instances (*objects*)
 - Initialization & tear-down (*constructor/destructor*)
 - Plus inheritance, polymorphism, ...
 - Further increase opportunities for reuse

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

35

Objects in C

- “Class” declaration (header file)

```
#define AddFifo(NAME,SIZE,TYPE, SUCCESS,FAIL) \
unsigned long volatile PutI ## NAME; \
unsigned long volatile GetI ## NAME; \
TYPE static Fifo ## NAME [SIZE]; \
void NAME ## Fifo_Init(void){ \
    PutI ## NAME= GetI ## NAME = 0; \
} \
int NAME ## Fifo_Put (TYPE data){ \
    if(( PutI ## NAME - GetI ## NAME ) & ~(SIZE-1)){ \
        return(FAIL); \
    } \
    Fifo ## NAME[ PutI ## NAME &(SIZE-1)] = data; \
    PutI ## NAME ## ++; \
    return(SUCCESS); \
} \
int NAME ## Fifo_Get (TYPE *datapt){ \
    if( PutI ## NAME == GetI ## NAME ){ \
        return(FAIL); \
    } \
    *datapt = Fifo ## NAME[ GetI ## NAME &(SIZE-1)]; \
    GetI ## NAME ## ++; \
    return(SUCCESS); \
}
```

} “Member” variables

} “Methods”

- Object instantiation

```
AddFifo(Tx,32,unsigned char, 1,0)
```

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

FIFO_4C123.zip

36

Design Patterns

- Buffered I/O

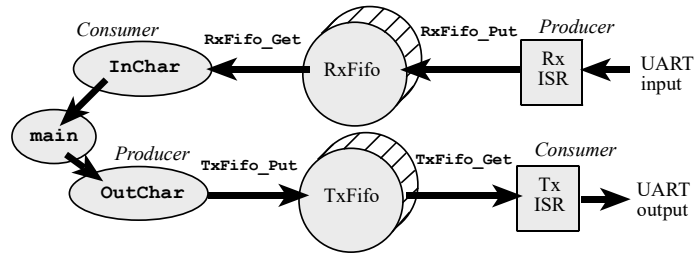


Figure 3.3. A data flow graph showing two FIFOs that buffer data between producers and consumers.

UARTInts_4C123.zip

Structured Programming

- Flowcharts

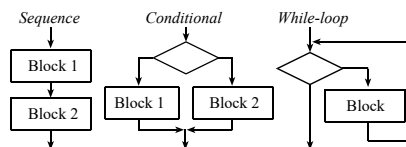


Figure 2.1. Flowchart showing the basic building blocks of structured programming.

- Parallel constructs

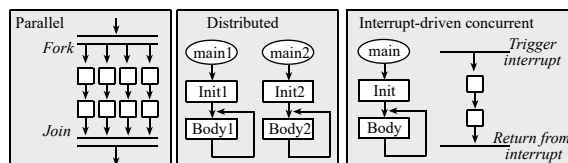


Figure 2.2. Flowchart symbols to describe parallel, distributed, and concurrent programming.

Coding Style

- Find your own conventions
 - Naming
 - Meaning, type of variables & functions
 - Readability
 - Indentation, white space
 - Documentation
 - Comments for every module, function, code block

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

39

Testing

- Unit vs. integration testing
 - Test function, module before integrating into next bigger system
- Black box vs. white box testing
 - Just inputs/outputs vs. can probe inside
 - Know what it does vs. know how it works
 - Have interfaces vs. have internals

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

40

Debugging

1. Stabilize the system (reproducibility)
 - Creating a test that fixes (or stabilizes) all inputs
 - Can reproduce the exact bug over and over again
 - Modify the program
 - Any change is a function of modification only

2. Debugging instruments (control, observability)
 - Software & hardware debuggers
 - Breakpoints, single-step, inspect
 - “Rough and ready” manual methods
 - Dumps, printf
 - Logic analyzer
 - Real-time observation of I/O signals

Intrusiveness ↑

<http://www.digilentinc.com/analogdiscovery/>
 Software: <http://www.digilentinc.com/waveforms/>
<http://www.usbee.com>
<http://www.saleae.com>

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

41

Debugging Real-Time Systems

- Events that are observable in real time
 - The input and output signals of the system
 - Observe using logic analyzer
 - Dumps
 - Record in real time, observe later off line
 - Extra output pins
 - Heart beats, monitors, profiling (logic analyzer)

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

42

Functional Debugging

- Verification of input/output behavior
 - What data is processed at specific points
 - What outputs are produced in response to input
- A static process where
 - inputs are supplied,
 - the system is run, and
 - the outputs are compared against expected results.

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

43

Functional Debugging Methods

- Intrusive methods
 - Single stepping or trace
 - Breakpoints
 - Instrumentation w/ print statements
 - Difficult in embedded systems
 - A standard output device may not be available
 - Output may be slow (relative to rest of system)
 - Output device used for normal operation
 - Send print output to special debug device
 - E.g. UART, see Lecture 1

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

44

Performance Debugging

- Verification of timing behavior of system
 - When do specific events occur
 - Measure dynamic efficiency of software
 - Delta time spent in pieces of code
- A dynamic process
 - System is run, and
 - dynamic behavior compared to expected results

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

45

Debugging Instruments (1)

- Dump into array with or without out filtering
 - Dumps strategic information into an array at run time
 - Observe the contents of the array at a later time
 - Use debugger to visualize when running

```
long DebugList[100];
unsigned int DebugCnt=0;
void RecordIt(long data){
    if(DebugCnt==100) return;
    DebugList[DebugCnt]=data;
    DebugCnt++;
}
```

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

46

Debugging Instruments (2)

- Monitor using output port (GPIO or LED)
 - Measure using scope or logic analyzer
 - Heartbeat, events at points in code
 - Measure time elapsed between two events

```
#define PF1 ((volatile unsigned long *)0x40025008)
#define GPIO_PORTF_DATA_R ((volatile unsigned long *)0x400253FC)
PF1 = 0x02; // atomic (specific for TM4C)
GPIO_PORTF_DATA_R |= 0x02; // not atomic
PF1 = 0x00; // atomic (specific for TM4C)
GPIO_PORTF_DATA_R &= ~0x02; // not atomic
```

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

47

Performance Instruments

- Use SysTick as independent hardware counter

```
unsigned long before, elapsed;

// ranges from 0 to NVIC_ST_RELOAD_R
unsigned long OS_Time(void) {
    return NVIC_ST_CURRENT_R; // 20ns
}

void main(void) {
    before = OS_Time(); // initialize
    // software to test, assume no interrupts
    ...
    elapsed = OS_TimeDiff(OS_Time(), before);
}
```

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

48

Profiling

- Dump with independent counter

```
long DebugList[100];
unsigned long TimeList[100];
unsigned int DebugCnt=0;
void RecordIt(long data) {
    if (DebugCnt==100) return;
    DebugList[DebugCnt]= data;
    TimeList[DebugCnt] = OS_Time();
    DebugCnt++;
}
```

- Monitor using an output port
 - Allocate different ports to different strategic events
 - Observe and measure in logic analyzer

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

49

Debugging Style

- Develop your own unique style
 - Place all print statements in a unique column
 - Specific pattern in their names
 - Test a run time global flag
 - Leaves a copy of the code in the final system
 - Simplifies “on-site” customer support
 - Use conditional compilation (#ifdef DEBUG)
 - Performance and effectiveness
 - For safety-critical systems leave instruments in

Lecture 1

J. Valvano, A. Gerstlauer
ECE445M/ECE380L.12

50

Summary

- Setup Laptop & Keil
- Learn ARM assembly language
- Get your board & display
- Get familiar with TM4C123 microcontroller
- Learn & practice how to debug
- Get started on Lab 1!
 - If you can finish Lab 1 by yourself, you will be fine in this class...