

Embedded Software

CMPED44

1/

Sept. 29 (Wed) 4:10-8:10pm.

Zoom Link To Be Used for the Entire Semester

Harry Li, Office: Engg. Building
Rm 261A

Email: hua.li@sjtu.edu.

Text messages (650) 490-1116.

Grading Policy:

1^o Projects & Assignment 30%

2 mandatory Projects, $10\% \times 2 = 20\%$

1 Semester-long Project 10%

2^o Midterm: 30%

3^o Final: 40%

Organization of the Course

1. CPU Architecture Memory map.

Special Purpose Registers for the init & config of Peripheral Controller. Firmware Development. (~ 3 weeks)

2. Kernel (OS) Source Distribution

I.D.E (Integrated Development

Environment), To be able to

Optimize Kernel image, to be able

to modify existing Device Drivers.

To write your own Device Driver.

(~ 3 weeks)

3. Integration & Development of O.S. Kernel + Device Driver + Sensors / Actuators

Stepper Motor Drive

Sensors LSM303

P.I.D controller.

Fourier Transform.

Web Server (HTTP)

OpenCV, OpenGL

Introduction

1. Development Setup

Target Board

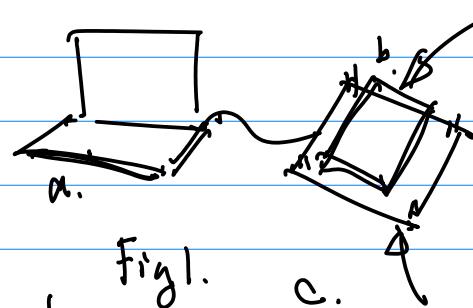


Fig 1.

Wire Wrapping Board

a. Host PC/Laptop, Linux
Ubuntu 18.04

(Virtual Box Installed, then
install Linux on top of it.)

b. Target Platform (To Be
Determined)

c. Wire Wrapping Board
① $\sim 3\frac{1}{2} \times 4$ " physical dimension
through holes with metal coating.

(2) 4 mounting holes @ the corners, 4 stand-offs (legs)

(3) LED Red/Green Current $\leq 10\text{mA}$
Resistors $V_{LED} \approx 1.8\text{VDC}$

$200 \sim 500\Omega$



A

B

(4) Toggle switch



2. OS Architecture

3. Selection & Evaluation of A Target platform

Linux D.S. Support

ARM (RISC — Reduced Instruction Set Computer)

{ Most Efficient Computation
Density per Unit Power
Reduced Instruction Set
(Smaller collection of Machine Code Instructions
→ Better Optimization of Compiler)

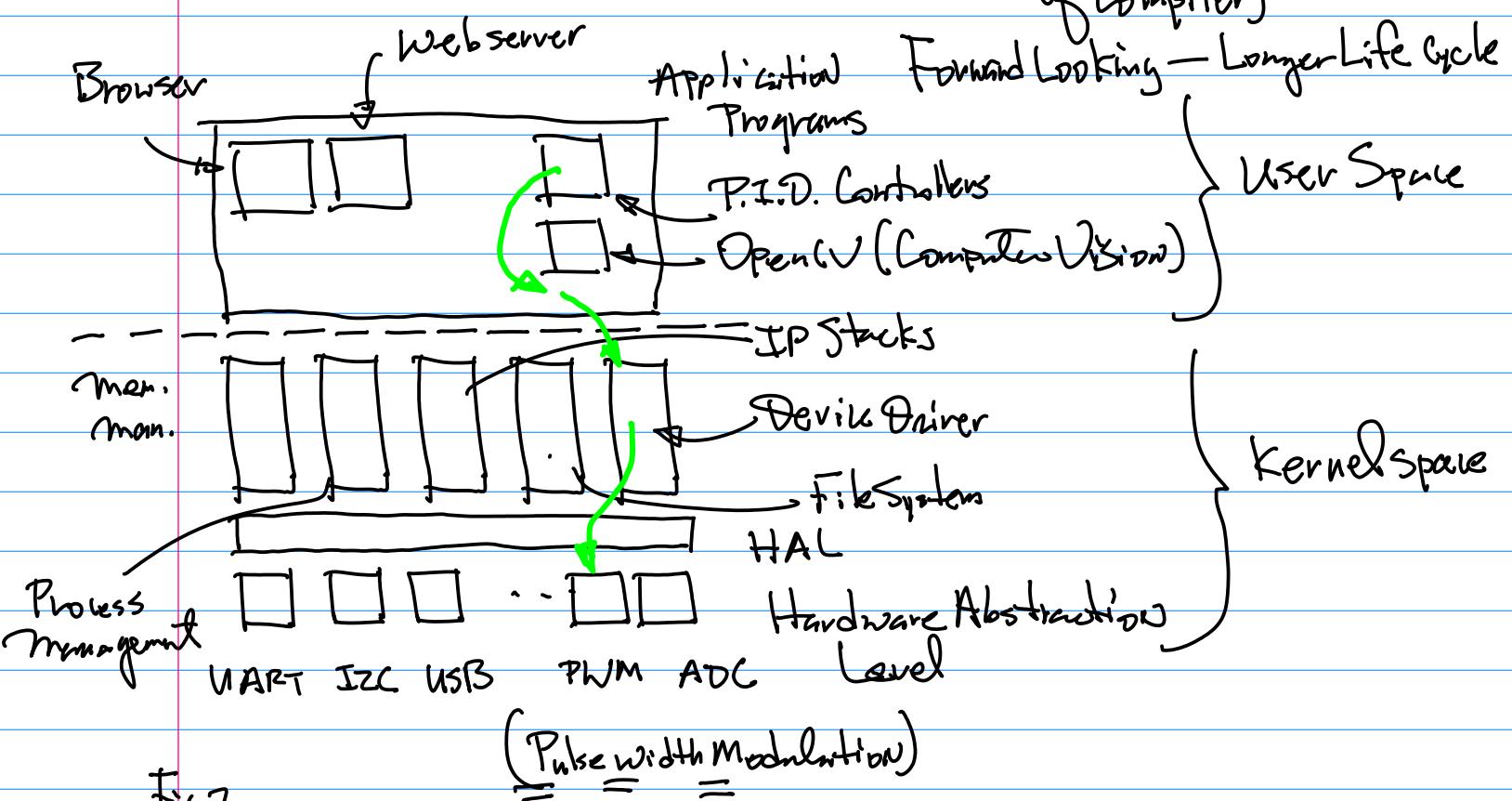


Fig.2

Target Platforms To Consider

1. NXP LPC17xx, 1769

Clock Rate: ~20MHz - 400MHz

RTOS But Not Unity OS.

No Linux

Rich I/O Interface

Ref: git ~ 2021F-107b - ..

Example: Schematic of LPC1769.

LPCXpresso	
GND	
VIN (4.5-5.5V)	
VB (battery supply)	
RESET_N	
P0.9	MOSI1
P0.8	MISO1
P0.7	SCK1
P0.6	SS1
P0.0	TxD3/SDA1
P0.1	RxD3/SCL1
P0.18	MOSI0
P0.17	MISO0
P0.15	TxD1/SCK0
P0.16	RxD1/SSEL0
P0.23	AD0.0
P0.24	AD0.1
P0.25	AD0.2
P0.26	AD0.3/AOUT
P1.30	AD0.4
P1.31	AD0.5
P0.2	
P0.3	
P0.21	
P0.22	
P0.27	
P0.28	
P2.13	

"3+1" pin: SPI1

(Serial Peripheral Interface)

} UART3 { Tx Transmission / multiplexed
} Rx Receiving { SDA: Serial Data
} SPI1 / UART1 { SCL: Serial CLK

} 6 ADC (AD - Analog to Digital conversion) I2C

} GPIO (GPIO: General Purpose Port I/O)

CmPE244

4

	+3V3
J2-28	
J2-29	
J2-30	
J2-31	
J2-32	ETH_RXN
J2-33	ETH_RXP
J2-34	ETH_TXN
J2-35	ETH_TXP
J2-36	USB-DM
J2-37	USB-DP
J2-38	P0.4
J2-39	P0.5
J2-40	P0.10
J2-41	P0.11
J2-42	P2.0
J2-43	P2.1
J2-44	P2.2
J2-45	P2.3
J2-46	P2.4
J2-47	P2.5
J2-48	P2.6
J2-49	P2.7
J2-50	P2.8
J2-51	P2.10-ISP_EN
7 J2-52	P2.11
13 J2-53	P2.12
19 J2-54	GND

LPCXpresso

- VOUT (+3.3V out) if self powered, else +3.3V input
- not used
- not used
- not used
- RD-
- RD+
- TD-
- TD+
- USB-D-
- USB-D+
- P0.4 CAN_RX2
- P0.5 CAN_TX2
- P0.10 TXD2/SDA2
- P0.11 RXD2/SCL2
- P2.0 PWM1.1
- P2.1 PWM1.2
- P2.2 PWM1.3
- P2.3 PWM1.4
- P2.4 PWM1.5
- P2.5 PWM1.6
- P2.6
- P2.7
- P2.8
- P2.10
- P2.11
- P2.12
- GND

J2-32, J2-33, J2-34, J2-35 grouped under Ethernet (TxN, RxP; TxN, RxP)
 J2-36, J2-37 grouped under USB
 J2-38, J2-39 grouped under CAN-Bus
 J2-40, J2-41 grouped under UART2/I2C
 J2-42 through J2-54 grouped under 6 ports PWM

Option Target platform: FPGA. Future Electronics.

FPGA IP Core: RISC-V.

Superset of ARM Architecture

IP Core: Open Source. Supports RTOS

Limitations: No Unix/Linux O.S.

Smaller Gate Counts, less

Computational Capability.

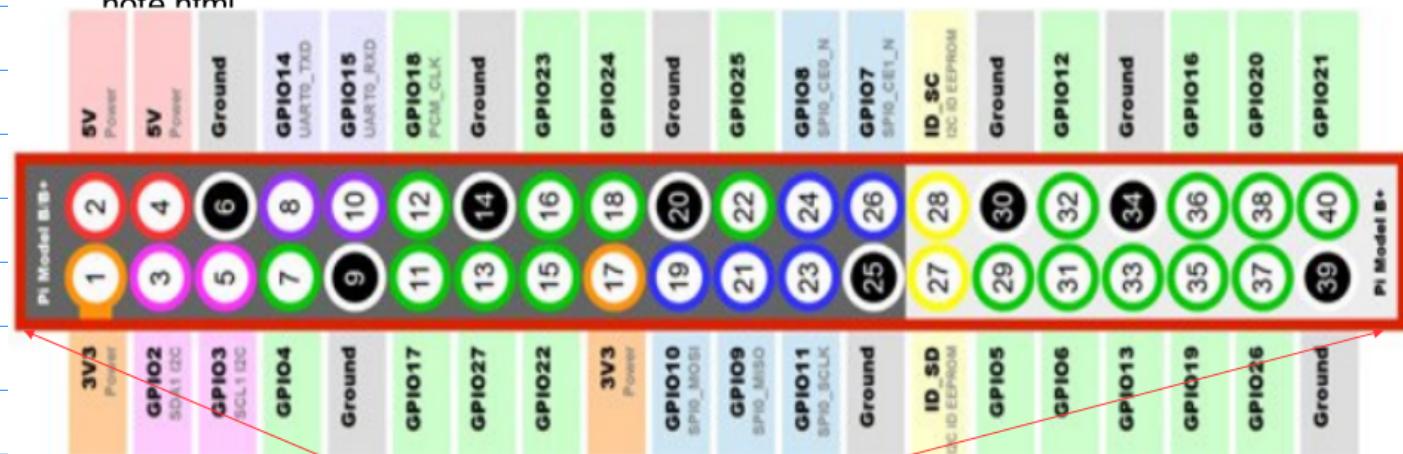
Pie. Broadcom BOM

Features: Support Linux/Ubuntu.

Provides Machine / Computer Vision Capability ; OpenCV.
Yolo (You Only Look Once) — Deep Learning , No ADC

Pie-3 Version B GPIO Pins

<https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html>



Plus: CPU Datasheet

Device Driver Development.

CPU Architecture

Memory Map

Peripheral Controllers, G.E. (Graphics Engine)

Option: ARM-11 Samsung C111: S3C6410X

CBU Worksheet: Architecture Block Diagram — well Document
born Smartz memory map, well Designed, documented

Support Linux, well Document / Sample code for Driver Development.

State-of-the-Art Feature: Graphics Processing Engine — GPU

LPC17xx, NO; FPGA RISC-V, NO; BCM-Pie G.E. yes
ARM-11. Video Codec, Marginal

Option: NVDA → Jetson TX2

6 CPUs + 256 GPUs in a
Single Package

CmpE244

Supports Linux/Ubuntu, I/O Interface is limited; (LPC17xx has the Best I/O I/F Support); ~\$7W Dev. Kit. Well Documented Datasheet.

Sign up as a developer at nxp website www.nxp.com
6
MCU Xpresso

Option: NVIDIA Jetson Nano. Ubuntu Linux Oct 6 (Thursday).
Multiple GPUs + 128GB GPU Support.
I/O (Limited)
Datasheet — Not As Detailed as other platform

Topics:

1° Architectural Aspects of Embedded System for Software Implementation

Developer forum is very active and it gives a good references.

Datasheet + Board Sch. +

Special Purpose Register + IDE (Compiler and Flash Tool)

Homework: 1° Form 2-4 person team

By Next week;

2° Choose Target Platform

3° Bring Wirewrapping Board /

Prototype Board to the Class for show & Tell ;

4° Sign up @ Nvidia Website

as a developer → kernel (Ubuntu)

Source Distribution
Jetpack

NXP, mcu
Xpresso

www.nxp.com

Example: LPC1769 ARM Cortex M3

CPU Datasheet

Simpler Architecture

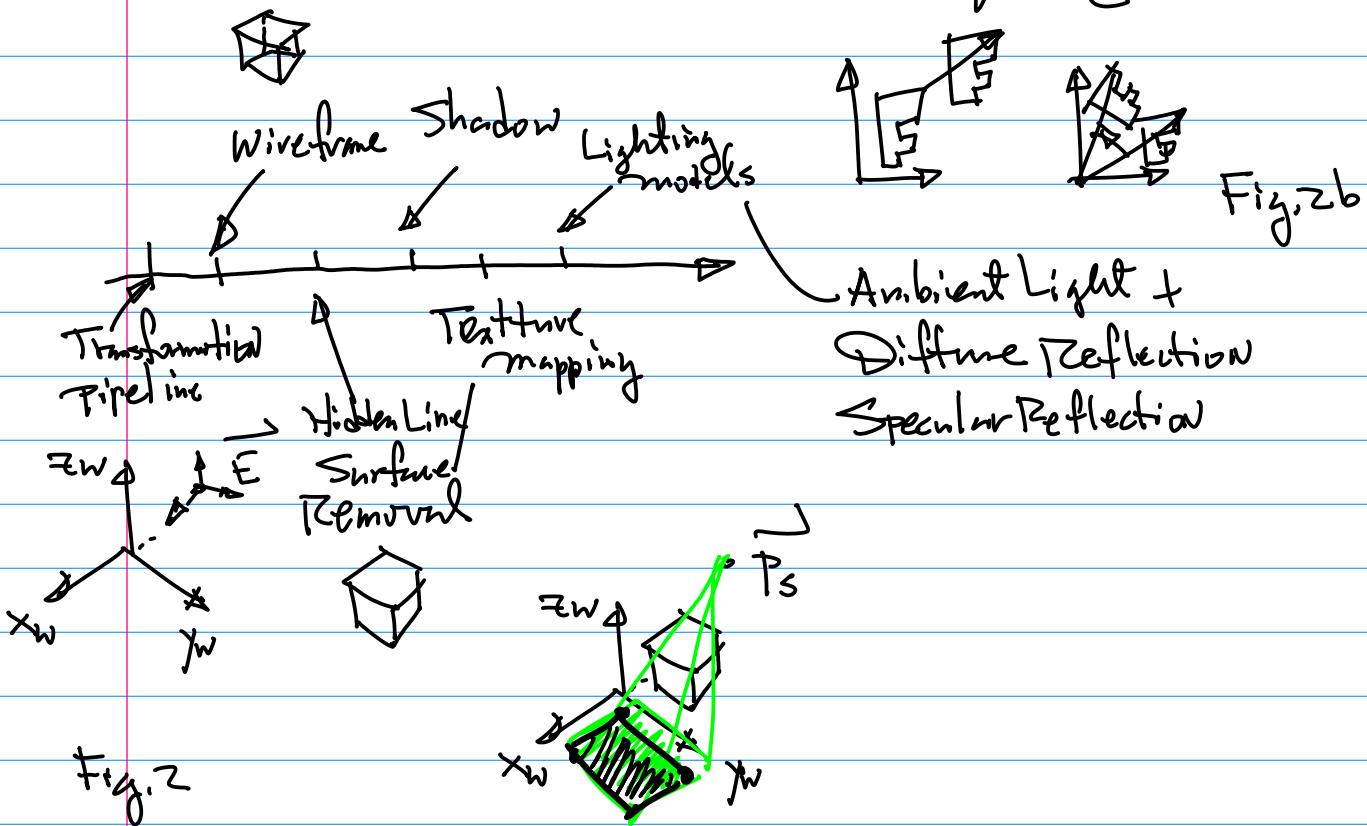
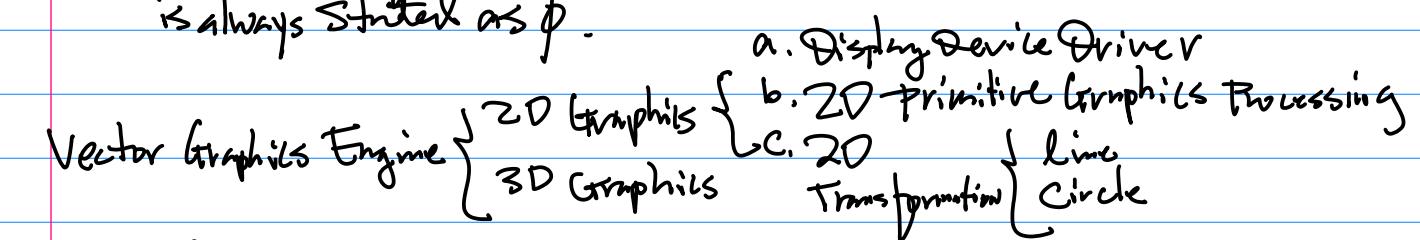
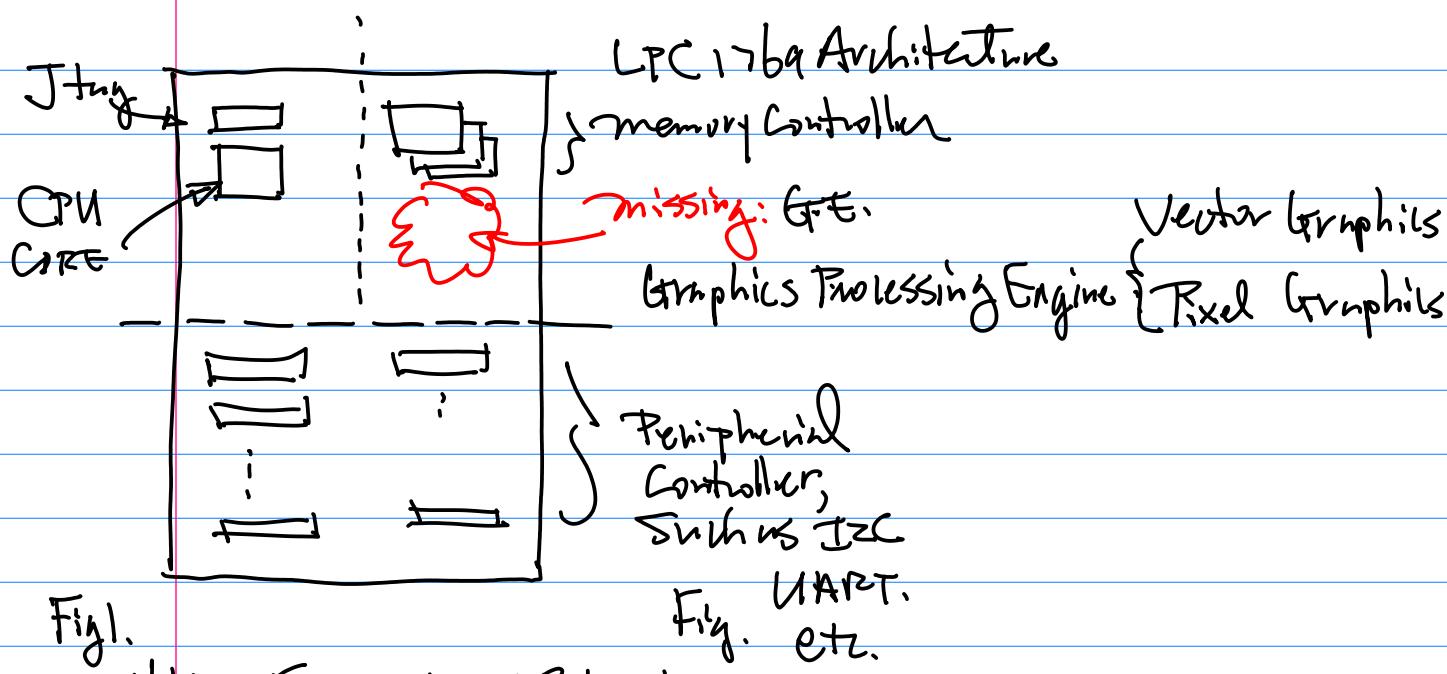
- (1) NXP LPC1769 Baseline
- (2) Samsung ARM11 Datasheet

CPU Block Diagram

Memory Map

Special Purpose Register

Tool Chain / Software Design, Implementation



Memory Map:

1. RISC: Reduced Instruction Set Architecture

1979 David Patterson

1982 John Hennessy

{ Uniformity
Regularity
Orthogonality

2. 32 Bit RISC Architecture

Data Bus: 32 bit, Bi-directional

$D[m:n]$ Vector Notation
most Significant Bit Least Significant Bit
 $D[31:\phi]$

Endian "Little Endian"

Address Bus: 32 bit, Uni-Directional

ALU (Arithmetic/Logic Unit) 32 bits

Register File: 32 bit

General Purpose Registers: 32 bit
GPRs

Special Purpose Registers: 32 bit
SPRs

GPRs: Those that can participate
Any meaningful Arithmetic
Logic Operations.

SPRs: Those Registers to
fulfill special functions,
such as init & config-for
Peripheral controllers.

3. Memory Map

a. Byte Addressable machine

The Smallest Memory Cell

With an Unique Address is

a Single Byte — "Byte

Addressable Machine".

Question: What is the
max. address for the memory
map of a given CPU?

Single 32 → Addr.
Bit Bus

Architecture 32 bits

$$\downarrow$$

$$2^{32} = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^2$$

$$2^{10} \dots 1K$$

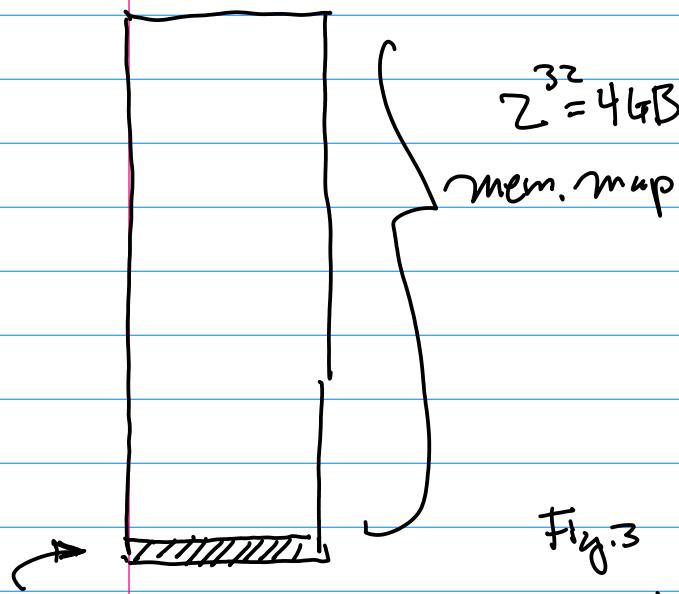
$$2^{20} \dots 1K \times 1K = 1 \text{ Meg.}$$

$$2^{30} \dots 1M \times 1K = 1 \text{ G.}$$

4 GByte (Byte Addressable)

CmPE244

9



0x0000_0000 System Power-Up Address:

The Address when CPU is powered

up, it will go to this memory location to fetch the 1st instruction to execute.

BANKS: A Block of memory.

Divide memory Map into 8

Equal Banks.

1st Bank: BANK 0

2nd Bank: BANK 1

:

8th BANK: BANK 7.

Question: How many Address bits do we need to define each memory bank? 3 bits

Question: Which 3 bits?

Addr. [3]: ϕ

$a_3, a_3, a_2, \dots, a_2, a_0$

Addr[3]:29] = a_3, a_3, a_2, a_2

Question: Find the Starting Address

of Each memory Bank?

SD:

a_3, a_3, a_2, a_1, a_0

0 0 0 : 0 0X0000_0000 BANK0

0 0 1 : 0 0X2000_0000 BANK1

0 1 0 : 0 0X4000_0000 BANK2

:

1 1 1

Datasheet, pp.14 from NXP LTC1769

Memory map, SSP ϕ as an example.

then, starting Addr. for Memory

Banks,

Starting Address SSP ϕ :

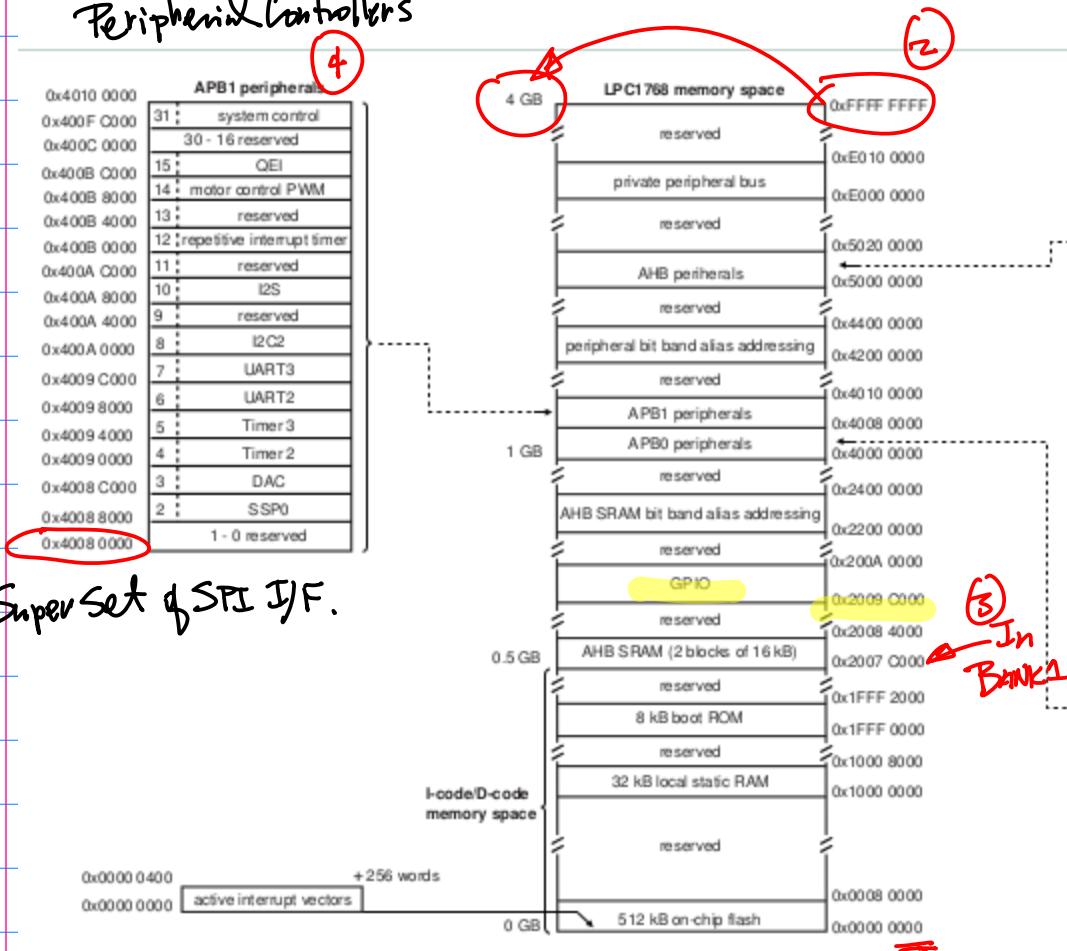
0x4f08_8000

Question: How much memory does SSP ϕ need?

Peripheral Controllers

APB1 peripheral	
0x4010 0000	31 : system control
0x400F C000	30 - 16 reserved
0x400C 0000	15 : QEI
0x400B C000	14 : motor control PWM
0x400B 8000	13 : reserved
0x400B 4000	12 : repetitive interrupt timer
0x400B 0000	11 : reserved
0x400A C000	10 : I2S
0x400A 8000	9 : reserved
0x400A 4000	8 : I2C2
0x400A 0000	7 : UART3
0x4009 C000	6 : UART2
0x4009 8000	5 : Timer3
0x4009 4000	4 : Timer2
0x4009 0000	3 : DAC
0x4008 C000	2 : SSP0
0x4008 8000	1 - 0 reserved
0x4008 0000	

SSP: SuperSet of SPI I/F.



Example: Find memory Needed for SSP.

Starting Addr: 0x4008_8000

End Addr: 0x4008_C000 - 1

$$= 0x4008_bFFF$$

8000-bfff Block of memory

For what purpose? Employed
By Special Purpose Registers
for init & configuration of SSP,

(1) PWR-map Add

And to perform Data
Input/Output Operation

Special Purpose Registers
Design.

focus on GPP (General Purpose
Port, e.g. GPIO)

3 Common Types of SPI

Fig 4

Control Register : Init & Config.

Data Register : Data I/O

Operation

Pull up/Down : Electrical characteristic
of the Controller

C. Typical Number of
GPPs :

LPC17ba P4, P1, P2, P3.

Samsung ARM II:

17 GPPs

S3C6410 includes 187 multi-functional input/o

4. Naming Convention of Special Purpose Registers.

Let's Design / Define Naming Convention.

follow RISC Design Guidelines

Prefix + Root + Postscript

3 letters 3 letters 3 letters

Control Register : GPx CON

GPx : General Purpose Port x

Meaning we have more than one general purpose ports.

PortName	Number of Pins.	
GPA port	8	UAI
GPB port	7	UAI
GPC port	8	SPI
GPD port	5	PCI
GPE port	5	PCI
GPF port	16	CAI
PGP port	7	SDI
GPH port	10	SDI
GPI port	16	LCI
GPJ port	12	LCI
GPK port	16	Hos
GPL port	15	Hos
GPM port	6	Hos
GPN port	16	EIN
GPO port	16	Mer
GPP port	15	Mer
GPQ port	9	Mer

Fig 5.

LPCXpresso	
GND	
VIN (4.5-5.5V)	
VB (battery supply)	
RESET_N	
P0.9	MOSI1
P0.8	MISO1
P0.7	SCK1
P0.6	SSEL1

① Negative "Active Low"
② Reset

1st Pin
P0.9
Dual row ho
J2-1

Part (GPP) 1

b Theoretically, pins can

be as many as 32

Fig 6.

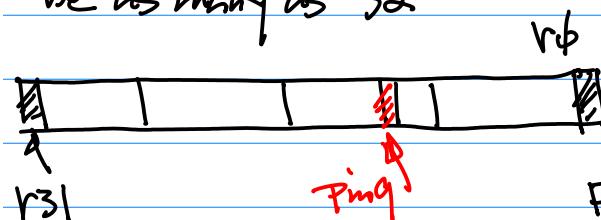
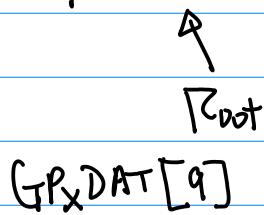


Fig 7

c. GPx DAT Ping:



d. Physical Connector pin to GPxDAT[9] is given from SCH Design. Example, LPC1769

Pb,9 → J2-5 (Connector J2, Pin 5)

e. GPx PUD (Pull-up/Down) SPR.

f. Question: How many control functions can we have for a single control register?

Homework: Download Image And Bring up your Target.

Oct. 13 (Wed)

Ref: 1^o 202F-112 - Homework
(Need Submission to CANVAS)

2^o 202F-113 - LPC17xx.h

(For NXP LPC17xx platform)

Homework, Note GPIO pins selection
Select one for Output, one
for the Input, for "Hello, the

world" program.

Turn on/off LED via
(Input) GPIO I/O
Read from
= GPIO pin for "1" or "0"
when switch is toggled:

Note: for NXP Xpresso IDE
download, you would have to
become a developer at NXP
website, www.nxp.com.
Then, download NXP Xpresso,
and finish the installation.

Once installed, download

LPC1769 pitch from class
github, import it into your
IDE.

Example: Continued from GPx CON
Number of Possible control
functions?

From CPU Datasheet (ARM), SPRs

	Register	Address	R/W
①	GPBCON	0x7F008020	R/W
	GPBDAT	0x7F008024	R/W
	GPBPUD	0x7F008028	R/W
	GPBCONSLP	0x7F00802C	R/W
	GPBPUDSLP	0x7F008030	R/W

②
Memory (Address)

Define GPB0 as an input pin.

GPBCON	Bit	Description
GPB0	[3:0]	0000 = Input 0010 = UART RXD[2] 0100 = IrDA RXD 0110 = Reserved
GPB1	[7:4]	0000 = Input 0010 = UART TXD[2] 0100 = IrDA TXD 0110 = Reserved

Table 1

Note 1: 32 Bit Architecture

32 Bit SPURS (Special Purpose Register)

Address 4 Bytes Apart.

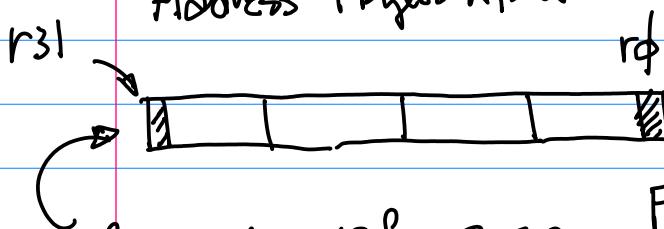


Fig 1. GPBCON 0x7f00-8020

Connector Type, DB-9

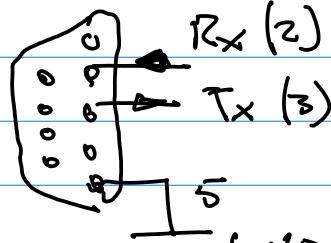


Fig. 2

No. of possible functions for Control register:

$$2^{32} = 4 \text{ G.}$$

GPB ϕ → pin ϕ → CPU, GPBDAT[ϕ]GPBCON[3: ϕ] = 0000 // Define GPB ϕ inputGPBCON[3: ϕ] = 0001 // Define GPB ϕ outputGPBCON[3: ϕ] = 0010, UART Rx (Receiving)

UART Serial Communication

"2+1" pins minimum Requirements
to establish UART
Communication

Rx (Receiving)
Tx (Transmitting)
GND

Example: Implement / Design

"Hello, the world" program.

Steps Involved for this is

1. Identify the pin(s)

from a connector
of a target Board;

2. Find Driver Program

to allow us to config
GPP for Input/Output
function.

14.

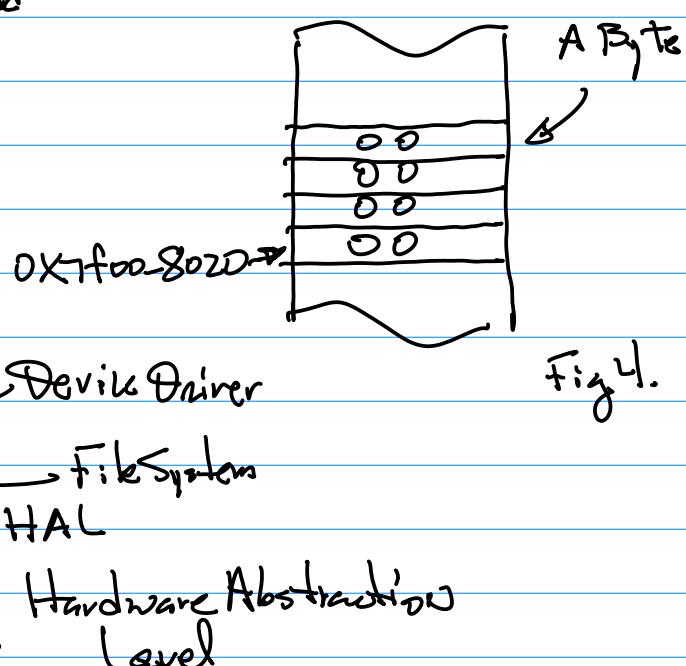
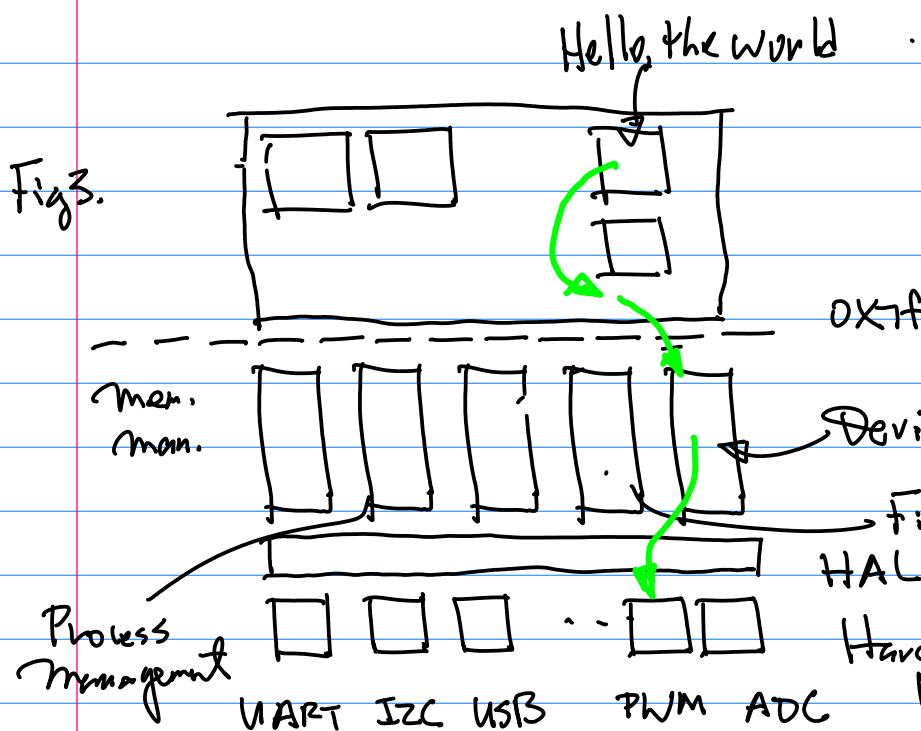
Note Driver Program in the environment where OS is installed, can be accessed in a user space.

From software.

Write $0x0000_0000$

to define GPB0 as an input

to the following memory location



A program will access to GPIO Devices as if it access to a file

open the file

read from the file

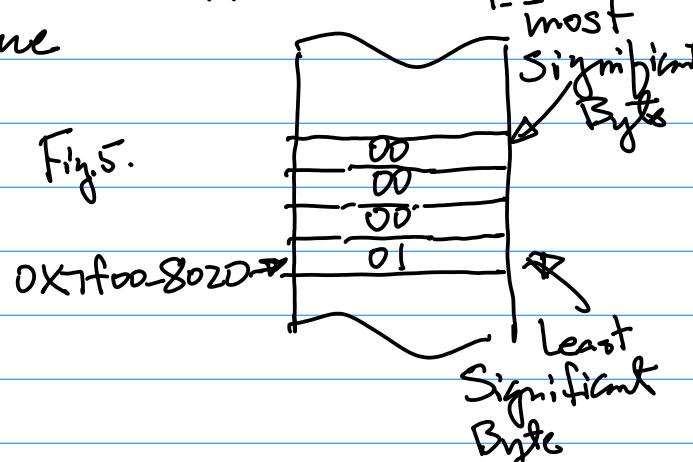
write to the file

In user space

Now, Define GPB0 as an output, from CPU Datasheet

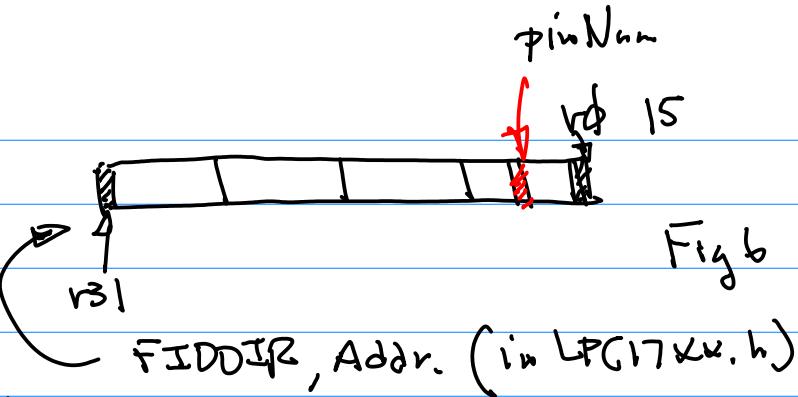
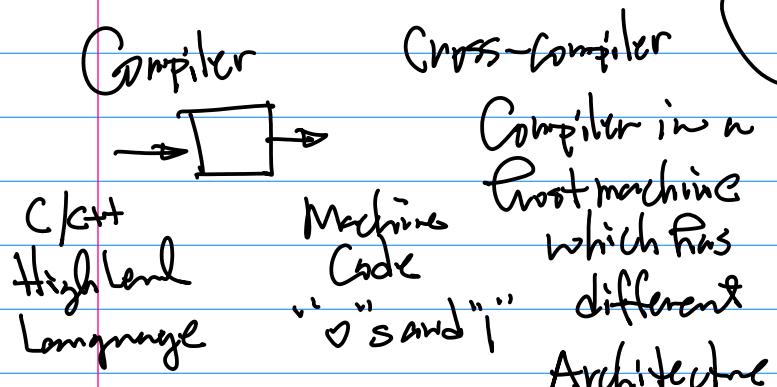
$GPBCON[3:0] = 0001$ Hex

Write $0x0000_0001$



In the kernel space, SPRs, like GPxCON, GPxDAT will fulfill the I/O I/F function.

Now, Discussion on IDE (MCU Xpresso)



#define FIDDIR 0x0000_0000
LPC_GPIOD → FIDDIR
 $I = (1 \ll \text{pinNum})$
for example make P0.3 as output pin. From SCH.

CPU Architecture (Memory Mapping) → Cross Compiler
"Port" Cross Compiler to a target CPU

= Machine code for ARM
↳ fits to the target CPU

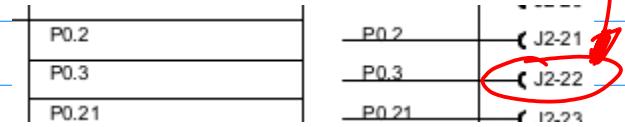


Fig 7.

Example: MCU Xpresso

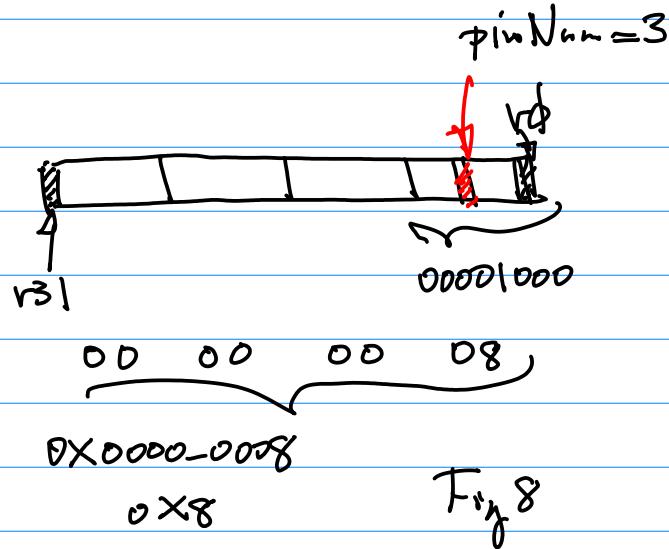
GPIO program (Project) →
firmware program into O.S.

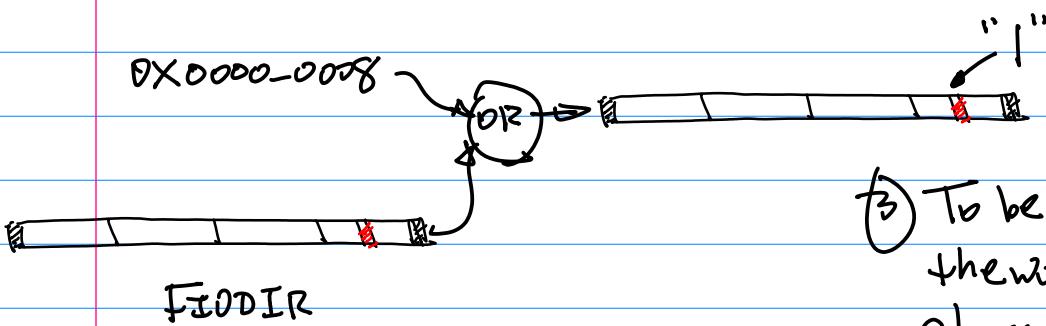
Sample code for GPIO operation.

HelloWorld.c code

LPC_GPIOD → FIDDIR
Target CPU Peripheral Controller
Fast I/O Direction

Write Binary Pattern in H/w. to set P0.3 as an output.





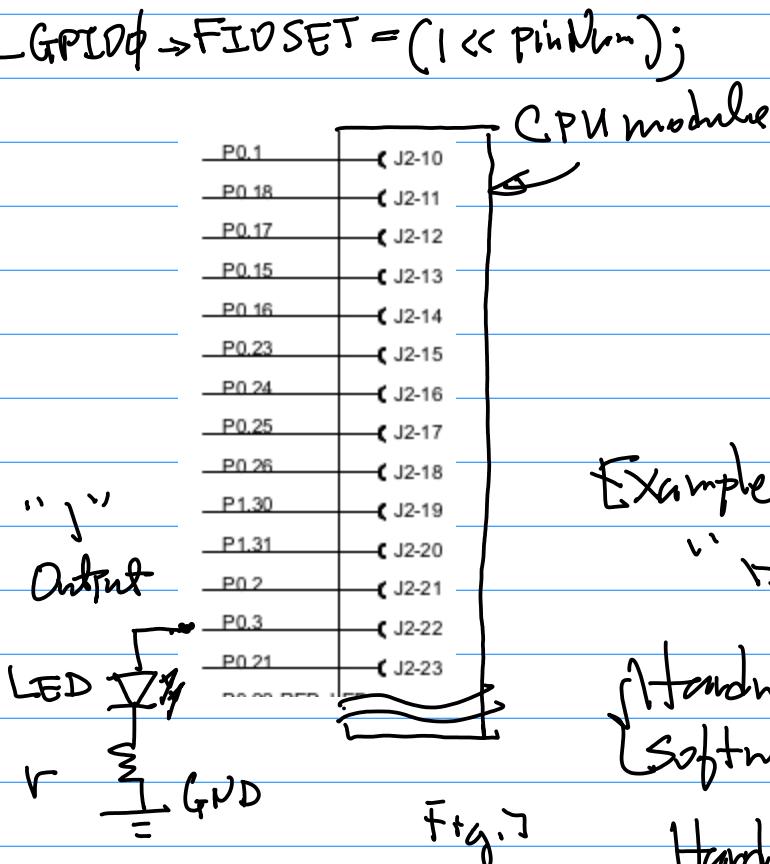
LPC_GPIODφ → FIOSET
 Special purpose
 Register:
 "DAT"
 Register

③ To be able to implement "Hello, the world" program on your chosen platform.

Note: $\stackrel{a}{\rightarrow}$ CPU Datasheet { ARM11 GPP
LPC GPP }

$\stackrel{b}{\rightarrow}$ SPIs { ARM11
GPPxCON
GPPxDAT
GPPxPWD
LPC
FIODIR
FIRSET }

\subseteq Sample code
`LPC17xx.h`
 Peripheral Control
 mem. mapping
 GPIO (Last 2~3 pages)



Example: Design/Implement
 "Hello, the world" program.

Hardware Design
 Software Design.

Hardware Design.

Step 1. Identify Hardware platform, Identify

Requirements: Define Init & Config
 ① Pattern Based on CPU Datasheet
 ② Analyze SPRs Responsible for GPIO operations. GPIO pins.

To start the Design with the set of a chosen target platform.

Note: 1° CMOS $[0, 3.3V]$

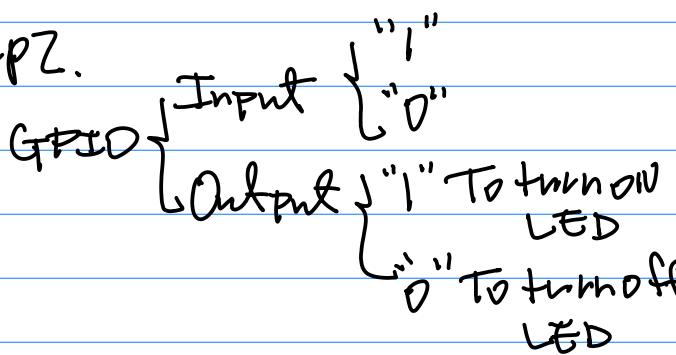
Find Connector(s) information

$$I_{CPU} \approx 10mA$$

Identify GPIO pins

For S/W connected to "A"

Step 2.

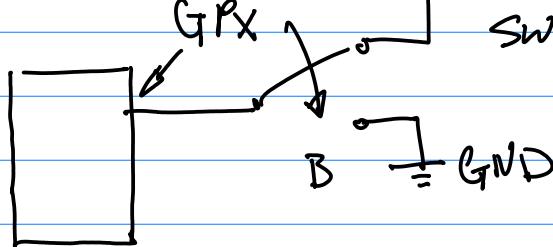


$$\frac{V_{CC}}{R_1} = I_{CPU} \dots (2)$$

$$V_{CC} = 3.3V, I_{CPU} = 10mA$$

$$\therefore R_1 = \frac{V_{CC}}{I_{CPU}} = \frac{3.3}{10 \times 10^{-3}}$$

$$\text{Input CKT: } = 3.3 \times 10^2 = 330\Omega$$



CPU (Connector)

Fig.8a

For S/W @ B
Assume GPx is output high

$$V_{GPX} = 3.3V$$

$$I_{CPU} = \frac{V_{GPX}}{R_2} \dots (3)$$

$$R_2 = \frac{V_{GPX}}{I_{CPU}} = 3.3 / 10 \times 10^{-3} = 330\Omega$$

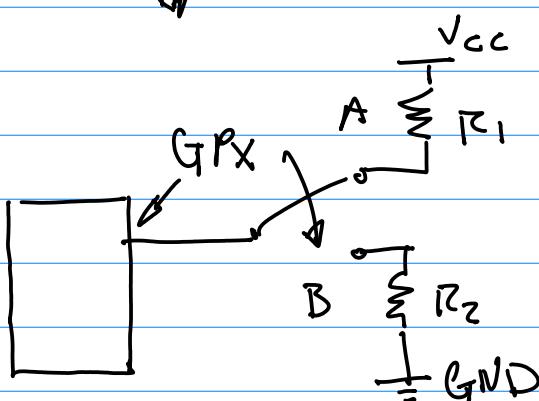


Fig.8b

$$R_1 = R_2 \approx 1k\Omega$$

Output Test. GPx

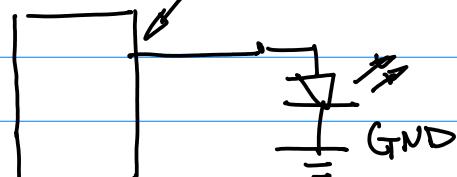


Fig.9a

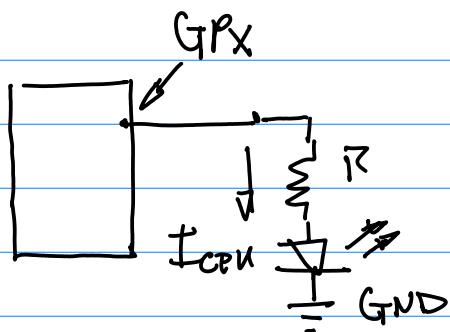


Fig. 9.b.

Let $I_{CPU} = 10 \text{ mA}$;

$$V_{LED} = 1.2 \text{ VDC}$$

$$I_{CPU} \cdot R = V_{GPX} - V_{LED} \dots (4)$$

$$R = \frac{V_{GPX} - V_{LED}}{I_{CPU}}$$

$$= \frac{3.3 - 1.2}{10 \times 10^{-3}} = 2.1 \times 10^2$$

$$= 210 \Omega$$

Bring your Prototype Board for quick Demo.

(Target Board + Carrier Board)

with I/O Testing Ckt)

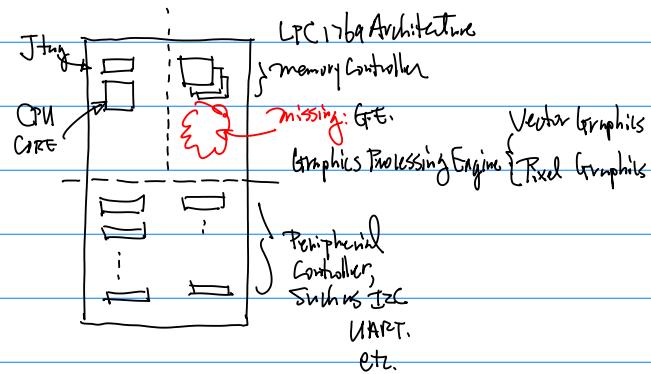
Oct. 20th (Wed)

Today's Topics: 1° GPIO Interface Design

On LTC/ARM11. 2° GPIO Implementation

On target platforms (Jetson NAND, Pie)

Note: Homework Discussion



CMPE 244
Homework CPU Architecture and Sample IDE

1. Use LPC1769 CPU datasheet as an example to create a CPU block diagram based on the discussion in the class, e.g.,
- 1.1. Find CPU or SOC (System on Chip) block diagram of you chosen platform (Jetson NANO, Nvidia Jetson Tx2, or Pi) and copy it into word document.
- 1.2. Identify the GPP port controller (general purpose port block, e.g., GPIO block) in the block diagram; ① pinAssignment ② Controller from Architecture
- 1.3. Find the target board connector diagram and place this diagram in your word document. Then choose 2 GPIO pins, one for output and one for input, for the coming GPIO hello-the-world assignment.

2. Install NXP MCU Expresso and import LPC1769 patch. You can find the LPC1769 patch download from here

<https://github.com/hualili/CMPE240-Adv-Microprocessors/blob/master/1769%20patch.zip>

Zip to be imported to the IDE.

Project Panel, Import Project into your IDE

The screenshot shows the NXP MCU Expresso IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Config, Tools, and Help. The Project panel on the left lists several projects, with 'DrawLine <Debug>' currently selected and highlighted in orange. Other projects listed include 1769 patch.zip_expanded, 1769template, CMSIS_CORE_LPC17xx, freertos_blinky, iperf_server, lpc_board_nxp_lpcxpresso_1769, lpc_chip_175x_6x, LPC1769_EINT_TIMER2016-4-19.zip_expanded, LPC1769_GPIO, lpcusbllib_KeyboardHost, lpcusbllib_MassStorageHost, lwip_tcpecho_freertos, lwip_tcpecho_sa, periph_adc, periph_blinky, and periph_bod. The code editor on the right displays a portion of the 'core_cm3.h' header file, specifically the SSP1 initialization code:

```

216 ** Returned
217 **
218 ****
219 void SSP1Init(
220 {
221     uint8_t i,
222
223     /* Enable SSPI */
224     LPC_SC->PCI
225
226     /* Further */
227     LPC_SC->PCI
228     /* P0.6~0.1 */
229     LPC_PINCON
230     LPC_PINCON
231
232     //if !USE_C
233     LPC_PINCON
234     LPC_GPIO0-
235 //endif
236
237     /* Set DSS */
238     LPC_SSP1->
239

```

Import Sample project (.zip) into the IDE.

<https://github.com/hualili/CMPE240-Adv-Microprocessors>

Configuration of IDE to work with Class example on
GPIO Interface.

CmpE244

20

Note: this IDE will allow us to analyze special purpose registers and their init & config, as well as mapping CPU architecture to memory map, and then making the connection to IDE cross compiler.

Once IDE + LFC1719 patch Import is accomplished, then
a+b will be posted on Class github together
with New Homework Assignment.
Jumper

Note: Please Purchase wires, LEDs (2~5 PCS),
Resistors (10Ω ~ 100kΩ). Need them
for Next week Homework ("Hello, the world"
On your target platform).

Homework Due Oct 27th, Submission to
CANVAS.

1. Bring up the target platform.

e.g. Jetson NANO or Pi.

(Submission of a photo. Showing
System Setup and Screen of the
target platform;

2. photo of your GPIO Circuit

a. Input CKT: (S/W or Jumper

wire to allow input "0" or
"1")

b. Output CKT: ^{with} LED on.

3. Submission

3.1. photos in Z.

3.2. Source code + Binary
(Executable)

3.3 Readme file.

please zip them into One file,
Submit to SJSLU CANVAS.

↳ Import GPIO Simple project
↳ Analyze + Read (Code
Walk Through the GPIO
Code Handout)

Example: GPIO Implementation on
NANO.

Ref: Ref. Linux { NANO
Pi }



Page Discussion

RPi GPIO Code Samples

The Raspberry Pi GPIOs can be controlled using

Main Page
Community portal
Current events
Recent changes
Help
Volunteering
Bug Tracker

Where else to find us
Twitter (@elinux)
#elinux on Libera.Chat
Facebook (@elinux.org)
Mailing Lists

Contents [t]

- 1 C
 - 1.1 Direct register access
 - 1.2 WiringPi
 - 1.3 sysfs**
 - 1.4 bcm2835 library
 - 1.5 pigpio
 - 1.6 I2C (local /dev/gpiochip I/F)
 - 1.7 rgpio (local & remote /dev/gpiochip I/F)

Ref2. from Nvidia Developer forum.

NVIDIA Jetson NANO Adaptation and Bring Up

This is the most
comprehensive
reference source

https://docs.nvidia.com/jetson/4/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/adaptation_andBringUp_nano.html&wpID0E0RR0HA

Nano Boards

Jetson Nano™ devices	Jetson Nano (P3448-0000) Developer kit version	Jetson Nano Developer Kit (P3450-0000) †; includes P3448-0000 module
	Jetson Nano (P3448-0002)	Jetson Nano 2GB Developer Kit (P3541-0000, P3541-0001); includes P3448-0003 module

Board Configuration and Developer Kits
Jetson Nano
Jetson Nano 2GB
Board Naming
Placeholders in the Porting Instructions
Root Filesystem Configuration
Pinmux Changes
Updated on the Boardlevel Forum

Step1. Down + Install Kernel Image (OS.)
Copy Kernel image to SD Card, then
Bring
Step2. Identify connector(s) which provide

CmpE244

21

GPIO pins to Build Interface.

However GPIO Mapping is different

NVIDIA Jetson Nano J41 Header Pinout

(2) General Guideline

<https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/>

Note: I2C and UART pins are connected to hardware and should not be reassigned. By default, all other pins (except power) are assigned as GPIO. Pins labeled with other functions are recommended functions if using a different device tree.

	GND	25	26	SPI_1_CS1	gpio20
	I2C_1_SDA I2C Bus 0	27	28	I2C_1_SCL I2C Bus 0	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_P20	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio77
	GND	39	40	I2S_4_SDOUT	gpio78

Hardware Pin

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
3.3 VDC Power		1	2	5.0 VDC Power	
I2C_2_SDA I2C Bus 1		3	4	5.0 VDC Power	
I2C_2_SCL I2C Bus 1		5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1	
	GND	9	10	UART_2_RX /dev/ttyTHS2	
gpio50	UART_2_RTS	11	12	I2S_4_SCLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3 VDC Power	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_1_CS0	gpio19
	GND	25	26	SPI_1_CS1	gpio20

(3) physical pin number
from your Target Board, Even Number on One Row,
Odd Number on the other.

Enumeration of the pin Numbers
Starts from 1

	GND	25	26	SPI_1_CS1	gpio20
	I2C_1_SDA I2C Bus 0	27	28	I2C_1_SCL I2C Bus 0	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_P20	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51

(4) Left Part (Col)

GPIO mapping. e.g. gpio149 → Physical pin 29

\$echo 79 > /sys/class/gpio/export

Connector Corresponds to gpio79?

Which pin on my

from the Pin Assignment Diagram
Pin 12 is mapped to gpio79.

Step 3 Build the Hardware Prototype.

Circuit. Make Sure a 2 GPIO pins selected, one for Input, one for

b Select the pins with Clear
mapping information, e.g. CPU gpio v.s.
physical pin connector.

Example: See Sample code below:

Build Connectivity Table.

	I2C_2_SDA I2C Bus 1	3	4	5.0 VDC Power
	I2C_2_SCL I2C Bus 1	5	6	GND
gpio216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1
	GND	9	10	UART_2_RX /dev/ttyTHS1
gpio50	UART_2_RTS	11	12	I2S_4_SCLK gpio79
gpio14	SPI_2_SCK	13	14	GND
gpio194	LCD_TE	15	16	SPI_2_CS1 gpio232

Sample Code Part.2

```

28 // #define PIN 24 /* P1-18 */
29 // #define PIN 78 /* P1-40 */ ① GPIO 78, physical pin 40
30 // #define PIN 78 /* P1-40 */ INPUT
31 // HL: 2021-10-4 commented out #define POUT 4 /* P1-07 */
32 #define POUT 79 /* HL 2021-10-4 for 40 pin connector pin-12 */
33

```

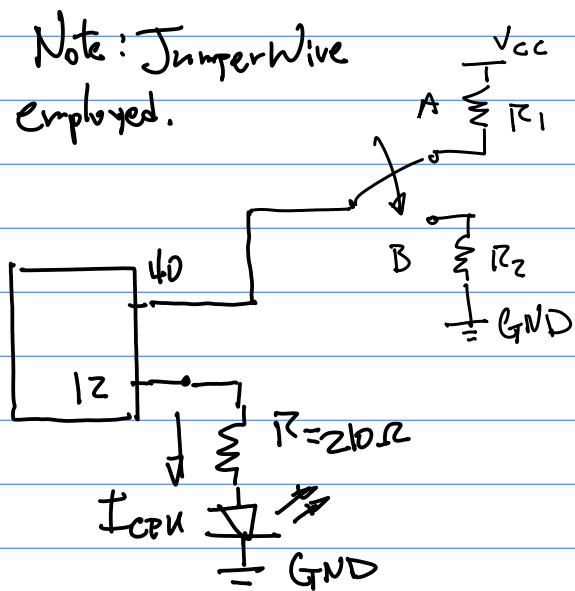
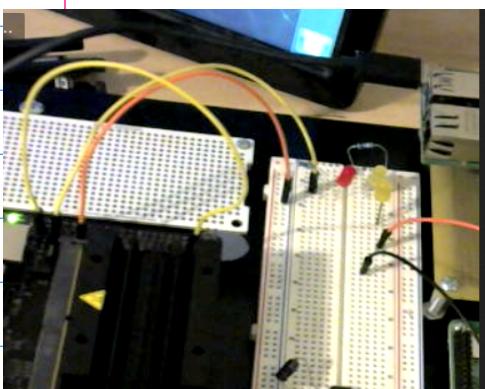
CPU	J41	Note
GPIO78	J41-40	line 30 / Input Conn. Topin 3P3V
GPIO79	J41-12	line 32 / Output Conn. LED+220Ω Resistor

Sample Code Part 1. (Header) I took pie code for NAND

```

1 ****
2 * blink.c
3 *
4 * Raspberry Pi GPIO example using sysfs interface.
5 * Guillermo A. Amaral B. <g@maral.me>
6 *
7 * This file blinks GPIO 4 (P1-07) while reading GPIO 24 (P1_18).
8 */
9 // https://elinux.org/RPi_GPIO_Code_Samples#sysfs
10 // HL: 2021-10-8
11 // build with:
12 // g++ -O1 -g -o mem gpiomem.cpp -Wall -std=gnu++17
13 // run with:
14 // sudo ./mem
15
16 #include <sys/stat.h>

```



Step 4. Software Implementation

Preliminary Testing. CLI (Command Line Instructions)

CMRE244

```
$echo 79 > /sys/class/gpio/export  
$ echo out > /sys/class/gpio/gpio79/direction  
$echo 1 > /sys/class/gpio/gpio79/value  
$echo 0 > /sys/class/gpio/gpio79/value  
$echo 79 /sys/class/gpio/unexport  
$cat /sys/kernel/debug/gpio
```

- ① Allocate option 79, used as an pair. (Export, 79/direction (c) unexport) Define it as output, /value direction /value

Note: please memorize these cities for the future use.

Once quick CI Testing Done, go to C/C++ Implementation.

Example: Code Walk-Through

Name of the program grid-p.c to
Be changed.

- 1° Line 1 ~ 14. Program Header (make a template)
Always provide program header
(For the homework as well as)

Program Name:

Coded by :

Date & Version:

Status: (Debug, Release)

Copyright:

Note: a Compilation + Build
is instruction

b Reference Source, such as
github, URL etc.

- Z' line 29-32 info: gpio mapping pins
Need Pin Assignment Diagram.
See P.P.T. (github) ... (... 114 ...)
pp.4.

① Static Declaration of a function module

```
3 static int
4 GPIOExport(int pin)
5 {
6     #define BUFFER_MAX 3
7     char buffer[BUFFER_MAX];
8     ssize_t bytes_written;
9     int fd;
10
11     fd = open("/sys/class/gpio/export", O_WRONLY);
12     if (-1 == fd) {
13         fprintf(stderr, "Failed to open export for writing!\n");
14         return(-1);
15     }
16
17     bytes_written = snprintf(buffer, BUFFER_MAX, "%d", pin);
18     write(fd, buffer, bytes_written);
19     close(fd);
20
21     return (0);
22 }
```

Consider Kernel Space Programming.
Device Driver Development, Debugging,
Improvement.

1. O.S. Source Distribution

Target platform dependent
distribution.

Jetson NAND --- Nvidia

Developer Forum

Tie ... Manufacturer Related

Site to Download

Kernel Source Distribution.

ARM11 as a baseline reference
for Device Driver Development.

Tools for Device Driver Development

IDF + CPU Architecture + Special

Purpose Registers + T.S. modules
as Device Driver.

Example: Introduction to Kernel
Space programming.

In Particular, Device Drivers.

2. Toolchain

Cross-Compiler plus other
Packages Provided by the
Vendor for other functions,
such as Computer Vision, GPU
graphics, ML, AI, etc.

Example: Introduction to Kernel
Space programming.

Example: (1) make menuconfig, Kernel Configuration (2) On x86 Host, for Linux kernel
4.4.197

harry@workstation: ~/nvidia/src/public_sources/kernel/kernel-4.4

File Edit View Search Terminal Help

.config - Linux/x86 4.4.197 Kernel Configuration

Linux/x86 4.4.197 Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in
[] excluded <M> module < > module capable

modify the
Kernel
Config Tool
By adding
→ ④

[*] 64-bit kernel
General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Executable file formats / Emulations --->
Networking support --->
Harry 2021-7-27 testing Device Drivers --->

① ② ③ → Built-In: Becomes a part
of the Bigger O.S. Kernel
Image.

Module: Built as a
Stand Alone module w/o
Compile & Build Entire O.S.

Discussion on Sensor Interface and P.I.D. Controller Design.

Sensor Interface :

- { X-Y-Z Acceleration Sensor
LSM303 ; for Robotics Applications, Such as ATVs, Autonomous Robot.
- Heart Beat/Rate Sensor,

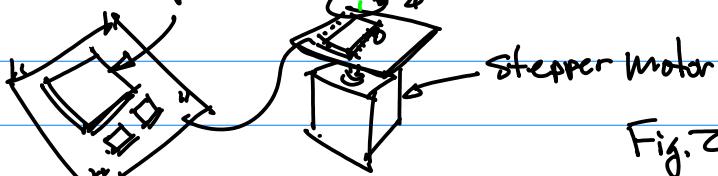


Fig.2

In Class, for Semester Long project



LSM303DLHC

Ultra compact high performance e-compass
3D accelerometer and 3D magnetometer module

Preliminary data

Features

- (1) ■ 3 magnetic field channels and 3 acceleration channels
- From ± 1.3 to ± 8.1 gauss magnetic field full-scale
- $\pm 2g/\pm 4g/\pm 8g/\pm 16g$ selectable full-scale
- 16 bit data output
- I²C serial interface

X-Y-Z
3 axis

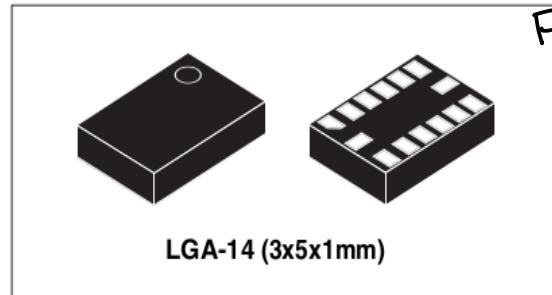


Fig.2

Note: In Fig.1. User Space program,

P.I.D. Controller (Proportional,

Integral, Derivative) \rightarrow Device Driver

(Kernel Space)

\cong Sensor: LSM303

\cong Activate Stepper motor.

Oct. 27 (Wed)

Today's Topics: 1^o GPIO Homework

Quick Review/Presentation of the homework. 2^o J.S. Source Distribution
& ToolChain, Example on Device Driver

Development: 3^o Sensors Needed for Semester Long Project.

Note: Homework on LFC Reference Platform.
IDE mCLXpresso, Requirements:

Requirements

- 1° Import LCC169 patch → Set the Development Environment
- 2° Import GRID project in Zip format, to exam/investigate Special Purpose Registers, and their roles in Init & Config → Connection to Datasheet (80+ pages)

CPU Architecture → P.13?

Memory Map

Special Purpose Register

- 3° Firmware Concept.

Note: Midterm Schedule

Nov. 17 (Wed) | 1 hr. Exam.

2~3 Question.

Example: Kernel Source Distribution & ToolChain Installation.

Objective: To be able to Build Kernel image, to Write/Debug Device Drivers.

What Do we need:

- 1° OS. Source Distribution from the Vendor of your target platform.

- a. Baseline Reference: ARM11
- b. NAND or C. Pie, Down

Down Kernel Source Distribution from the Company.

(Jetson Nano)

Please check the class github By the end of the Day today. Pie.

- 2° Tool Chain, Very often is packaged together with the Kernel Source distribution, for Example for Jetson Nano. (L4T) "Linux (D.S.) for Tegra (GPU platform)"

Step 1. Download Kernel Source And Tool Chain. And Install the Kernel Source & Tool Chain

(A Part of Homework Due Next Week Nov. 3rd)

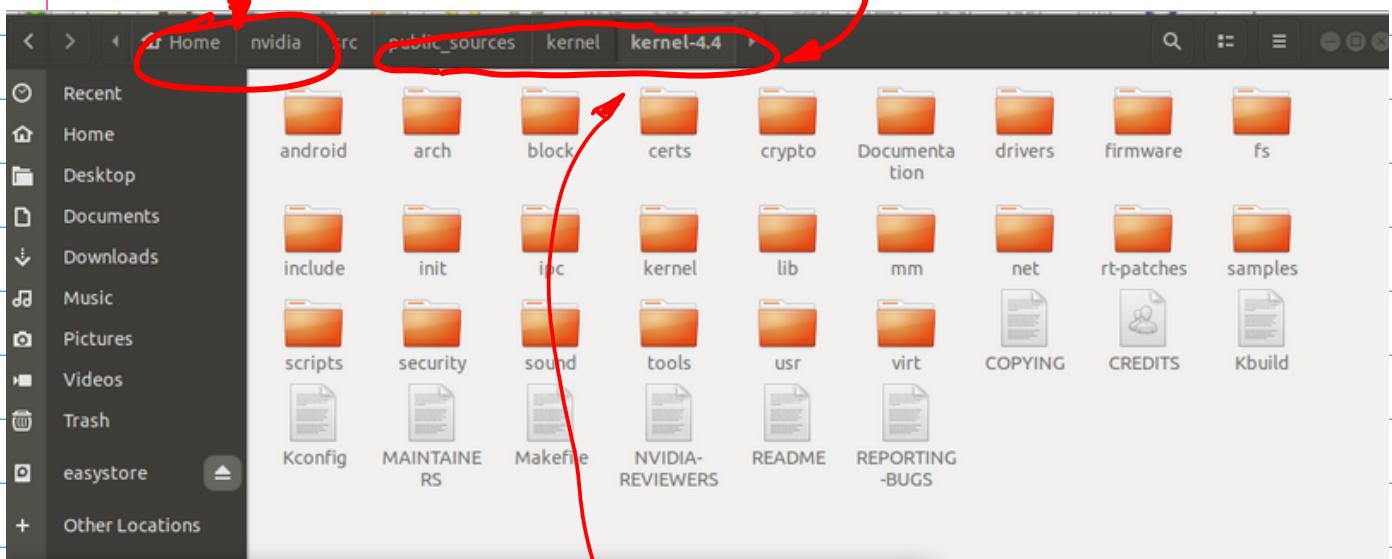
Download Instruction & Installation Guide are given by the vendor. for Linux (Ubuntu 18.04) Host—laptop.

To unzip the Download file.
(tar -xvf)

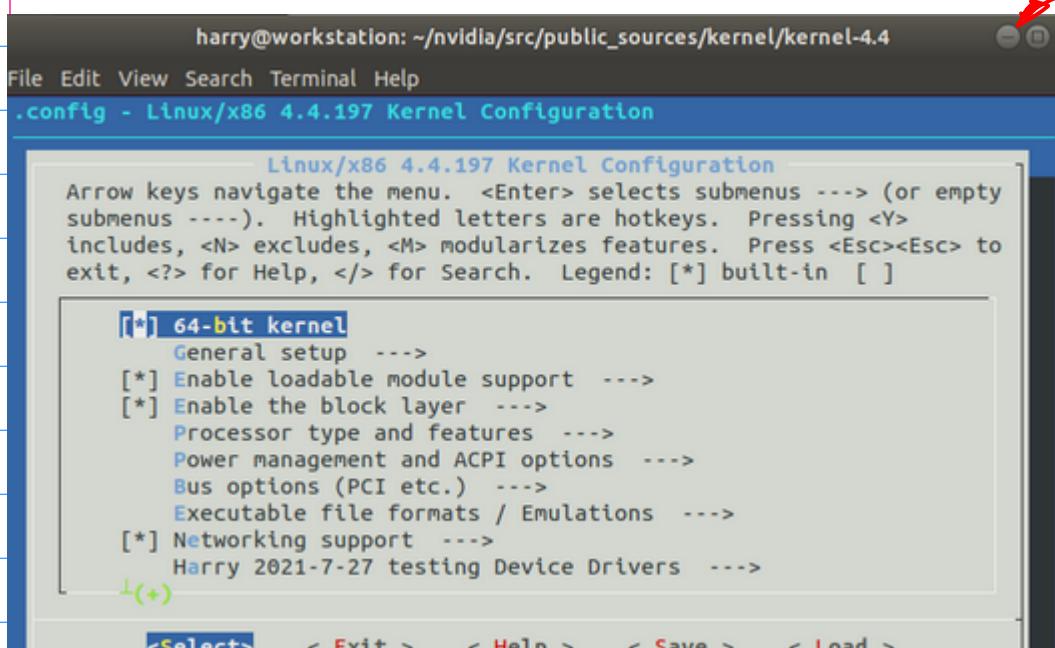
See Screen Capture Next Page.

OS. Source

① Installed O.S. Source + Toolchain (2) Toolchain, make menuconfig.



Step 2. Execute \$make menuconfig. Once the download is un-zipped, then go to the proper directory. for Example /kernel/kernel-4.4 (Depending on which version of the distribution you have). Then to start \$make menuconfig.



Homework : Due A week from today

(Nov.3rd) on CANVAS.

Submit : 1. Screen Capture of make menuconfig

2. Write Readme to Describe download, unzip, installation process

3. Provide Version, URL and clear description of the OS. Download.

Example: make menuconfig on ARM-1 platform.

Background: Programming in Kernel Space.

```
root@harry-laptop:/opt/FriendlyARM/mini6410/linux# ls
arm-qte-4.7.0    busybox-1.17.2  rootfs_qtokia_qt4      x86-qte-4.6.1
arm-qt-extended-4.4.3 examples        rootfs_qtokia_qt4-s   x86-qt-extended-4.4.3
arm-qtokia       linux-2.6.38     u-boot-mini6410      x86-qtokia
arm-qtopia
root@harry-laptop:/opt/FriendlyARM/mini6410/linux#
```

① Kernel Source ② u-boot for the Reference

Now, Consider User Space Program(s) which requires the utilization of Device Driver in Kernel Space.

④ Camera Interface

```
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples# ls
adc-test  camtest  i2c      lecSSBike  leds  readme.txt  threadTest  vfp-test  www
buttons   fft      tecPID   led-player  pwm   robotControl usbcam   Vision

```

③ User-Created Example

① GPIO LEDs

② PWM — pulse width modulation.

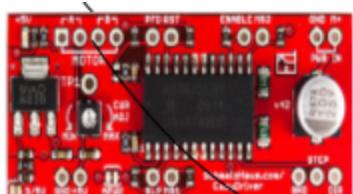
to Drive Stepper Motor

⑤ EEPROM (Electronic E. Progr. Read Only Memory)

~lwoma

Stepper motor Drive

~f10



Other Stepper motor Drive can be used as well.

Ads · Shop nema 17 stepper m...



[Amazon.com](#)

-

\$12.99

[Amazon.com](#)



[Amazon.com](#)

-

\$9.99

[Amazon.com](#)



[TexasInstruments.com](#)

-

\$2.19

[TexasInstruments.com](#)

PART II of the Homework: Purchase a

1. Stepper motor Drive (PWM Device Driver — PSD Controller User Program Implementation), And a
2. Stepper motor, NEMA17.

(Have them Ready By Next week Nov. 3rd)

User Space Program To Kernel Space (Device Driver), GPIO (program: leds.c)

```

15     printf("hello\n"); //Feb. 16, 2015
16
17     if (argc != 3 || sscanf(argv[1], "%d",
18                             on < 0 || on > 1 || le
19                             fprintf(stderr, "Usage: leds 1
20                             exit(1);
21
22
23     fd = open("/dev/leds0", 0);
24     if (fd < 0) {
25         fd = open("/dev/leds", 0);
26     }
27     if (fd < 0) {
28         perror("open device leds");
29         exit(1);
30
31
32     ioctl(fd, on, led_no);
33     close(fd);
34

```

Ready Build Kernel Image, which has this Driver

(1) fd (file descriptor) is defined as a file with open(" " , p) path(folder) and the device driver

Regined, please memorize!

(2) ioctl(fd, P1, P2); function to communicate to device driver defined by fd. Regined, Memorize!

③ Once done, close it. please!

Open-Close

In kernel Space.

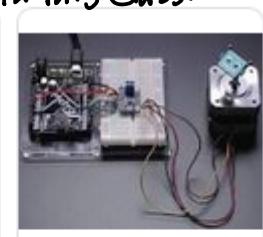
Note: The following Stepper motor Drive can be used in this class.



Qunqi L298N
Motor Drive ...
\$6.99
Amazon.com



STEPPERONI
CNC Stepper
\$28.99
Amazon.com
Free shipping



Adafruit
DRV8833
\$4.95
Adafruit Indu...
Amazon.com



Usongshine
Stepper Motor
\$10.99
Amazon.com

root@harry-laptop: /opt/FriendlyARM/mini6410/linux

.config - Linux/arm 2.6.38 Kernel Configuration

Linux/arm 2.6.38

Arrow keys navigate the menu. <Enter> selects hotkeys. Pressing <Y> includes, <N> excludes to exit, <?> for Help, </> for Search. Legend module capable

^(-) Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
Device Drivers ---> →
File systems --->
L(+) <Select>

Select
 (1) Go to Device Drivers
 (2) Select "Char" Device
 e.g. Character Device.

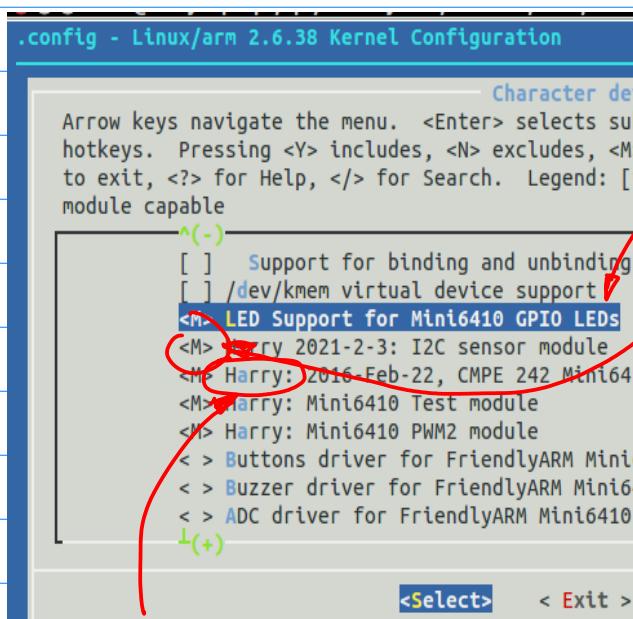
root@harry-laptop: /opt/FriendlyARM/mini6410/

.config - Linux/arm 2.6.38 Kernel Configuration

Dev

Arrow keys navigate the menu. <Enter> selects hotkeys. Pressing <Y> includes, <N> excludes to exit, <?> for Help, </> for Search. Legend module capable

^(-) SCSI device support --->
< > Serial ATA and Parallel ATA
[] Multiple devices driver support
< > Generic Target Core Mod (TC)
[*] Network device support --->
[] ISDN support --->
< > Telephony support --->
Input device support --->
Character devices ---> →
<*> I2C support --->
L(+) <Select>



③ GPIO Device Driver to turn on/off LED

Use Keyboard "SpaceBar" to select
Build it into An Entire OS Kernel Image
or Build stand-Alone module "M"
without Building the entire Kernel OS
And Later this module can be
installed as a part of a kernel
image.

or De-Select this module, e.g., Not
Build

④ Userdefined UI, By modifying the corresponding Kconfig file.

Now, for Kconfig at each level, now the
Kconfig @ Char Device level

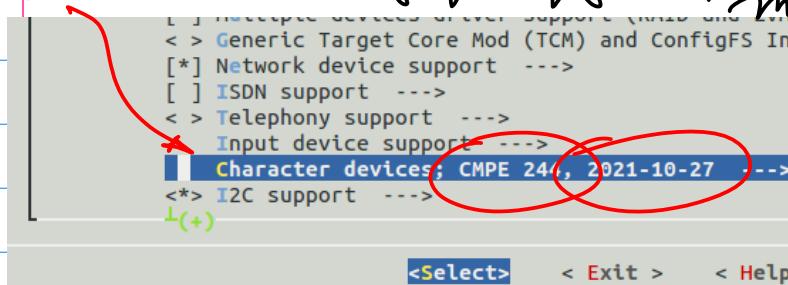
(Kernel)

Now, check Kconfig @ char told

```
root@harry-laptop: /opt/FriendlyARM/mini6410/linux
1 #
2 # Character device configuration
3 #
4
5 menu "Character devices"
6
7 config VT
8     bool "Virtual terminal" if EXPERT
9     depends on !S390
10    select INPUT
11    default y
12    ---help---
```

UI will show this
On the screen.

③ User can make change of Kconfig, to Add/Remove/
Modify ...



Defines select as a kernel, or select as "M"
 CMPED4 or De-Select.

Keywords (1) "Config" (4) "tristate" (5) "depends on" (6) "default"

```

100 config MINI6410_LEDS
101     tristate "LED Support for Mini6410 GPIO LEDs"
102     depends on CPU_S3C6410
103     default y
104     help
105         This option enables support for LEDs connected to GPIO lines
106         on Mini6410 boards.
107

```

(7) "help" Verbage for message.

Defines target 32 platform (with target CPU)

(2) Connection from "Kconfig" file to Actual Device Driver (char), note the folder (Path)

/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char

~ /drivers/char

(3) Device Driver for "LED" in the folder /char.

```

-rw-r--r-- 1 root root 10096 Oct 27 15:48 mini6410_hello_mod
-rw-r--r-- 1 root root 1920 Feb 22 2016 mini6410_leds.c
-rw-r--r-- 1 root root 76847 Oct 27 15:48 mini6410_leds.ko

```

Source Code

Compiled module, Ready to Be used.

Copy this ".ko" file, for example, onto USB-stick, plug the USB into the target platform (e.g. ARM-1)
 then, copy this file to your Working directory, use `insmod`

Name of the file

to make it a part of Kernel image ready to Be used.

The terminal window shows the following output:

```

root@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38
.config - Linux/arm 2.6.38 Kernel Configuration

CONFIG_MINI6410_LEDS:

This option enables support for LEDs connected to GPIO lines
on Mini6410 boards.

Symbol: MINI6410_LEDS [=m]
Type : tristate
Prompt: LED Support for Mini6410 GPIO LEDs
Defined at drivers/char/Kconfig:100
Depends on: CPU_S3C6410 [=y]
Location:
    -> Device Drivers
    -> Character devices

```

- Requirements:
- To be able to understand Vi/Vim and to locate Character Device Driver
 - To be able to modify "Kconfig" file at \drivers\char

(3) To be able to choose keywords, 5 of them, listed above.
 please memorize them!

"Kconfig @ \drivers

```

1 menu "Device Drivers"
2
3 source "drivers/base/Kconfig"
4
5 source "drivers/connector/Kconfig" ← Red arrow from here
6
7 source "drivers/mtd/Kconfig"
8
9 source "drivers/of/Kconfig"
10
11 source "drivers/parport/Kconfig"
12
13 source "drivers/pnp/Kconfig"
14
15 source "drivers/block/Kconfig"
--
```

① "Kconfig" Connector, Reports Process event

```

root@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/connector
menuconfig CONNECTOR
    tristate "Connector - unified userspace <-> kernelspace linker"
    depends on NET
    ---help---
        This is unified userspace <-> kernelspace connector working on top
        of the netlink socket protocol.

        Connector support can also be built as a module. If so, the module
        will be called cn.

if CONNECTOR
    config PROC_EVENTS
        boolean "Report process events to userspace"
        depends on CONNECTOR=y
        default y
        ---help---
            Provide a connector that reports process events to userspace. Send
            events such as fork, exec, id change (uid, gid, suid, etc), and exit.
endif # CONNECTOR
~
```

Now, "Kconfig" at Kernel Level.

root@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38

```

1 #
2 # For a description of the syntax of this configuration file,
3 # see Documentation/kbuild/kconfig-language.txt.
4 #
5 mainmenu "Linux/$ARCH $KERNELVERSION Kernel Configuration"
6
7 config SRCARCH
8     string
9     option env="SRCARCH"
10
11 source "arch/$SRCARCH/Kconfig"

```

Example: A Simple Device Driver, Does nothing. But "print"

ADC: Analog to
Digital Conversion

① Hello Driver

GPIO Drivers

PWM

```

root@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char# ls mini6410_*
mini6410_adc.c      mini6410_hello_module.c      mini6410_leds.c      mini6410_pwm2.c
mini6410_adc.mod.c  mini6410_hello_module.ko    mini6410_leds.ko    mini6410_pwm2.mod.c
mini6410_adc.o      mini6410_hello_module.mod.c  mini6410_leds.mod.c  mini6410_pwm.mod.c
mini6410_buttons.c   mini6410_hello_module.mod.o  mini6410_leds.mod.o  mini6410_pwmHarry.c
mini6410_buttons.o  mini6410_hello_module.o      mini6410_leds.o     mini6410_pwm.mod.c
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char#

```

Ex: INT: External Interrupts
Drivers

Note: "Hello" Driver. ① <linux/kernel.h>
<linux/module.h>

② Syntax: static int __init target(void)

root@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/d

```

① 1 #include <linux/kernel.h>
2 #include <linux/module.h>
3
4
5 (2) static int __init mini6410_hello_module_init(void)
6 {
7     printk("Hello, Mini6410 module is installed !\n");
8     return 0;
9 }
10
11 (3) static void __exit mini6410_hello_module_cleanup(void)
12 {
13     printk("Good-bye, Mini6410 module was removed!\n");
14 }
15
16 module_init(mini6410_hello_module_init);
17 module_exit(mini6410_hello_module_cleanup);
18 MODULE_LICENSE("GPL");

```

③ static void target
__exit Name

Steps to Test out this "Hello" module.

5-Step Process for Executing "Hello" module (In Kernel Space).

Sensor Selection for Semester Long, Please discuss with your team member,
And Selection By Next Wed

- 2021F-116b-heart-beat...
- 2021F-117-#lec4-Stepp...

3D Accelerometer and 3D Magnetometer LMS303

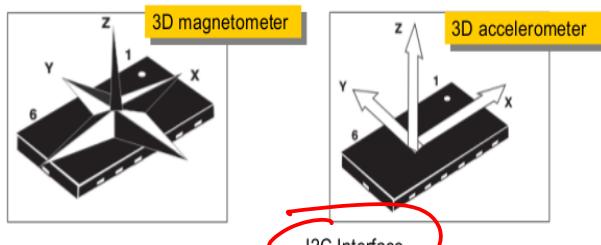


Table 9



Max30100 Sensor

max30100 pulse oximetry and heart-rate monitor sensor

I2C-Compatible Interface



MH-E
MAX:

I2C I/F

https://www.amazon.com/MAX30102-Detection-Concentration-Compatibility-Arduino/dp/B07ZQNC8XP/ref=asc_df_B07ZQNC8XP/?tag=hypprod-20

Nov. 3rd (Wed)

Nbs: Midterm Exam Scheduled
2 weeks from today. Nov. 17 (Wed).

1 hour Exam. On Line.

Option A: Lecture ($\sim 1\frac{1}{2}$ hr), then Exam.

Option B: Exam, then Break 30 min.
then lecture.

Topics: 1. Semester Long project
(Team Project) 2. Device Driver Development

Semester Long Project.

1. Team project. By End of the Semester,

Presentation, PPT, Project Demo.

2. Key elements for Project

a. Kernel space Programming, e.g.

Device Driver Development / Fix
Thinking;

b Integration of User Space Application

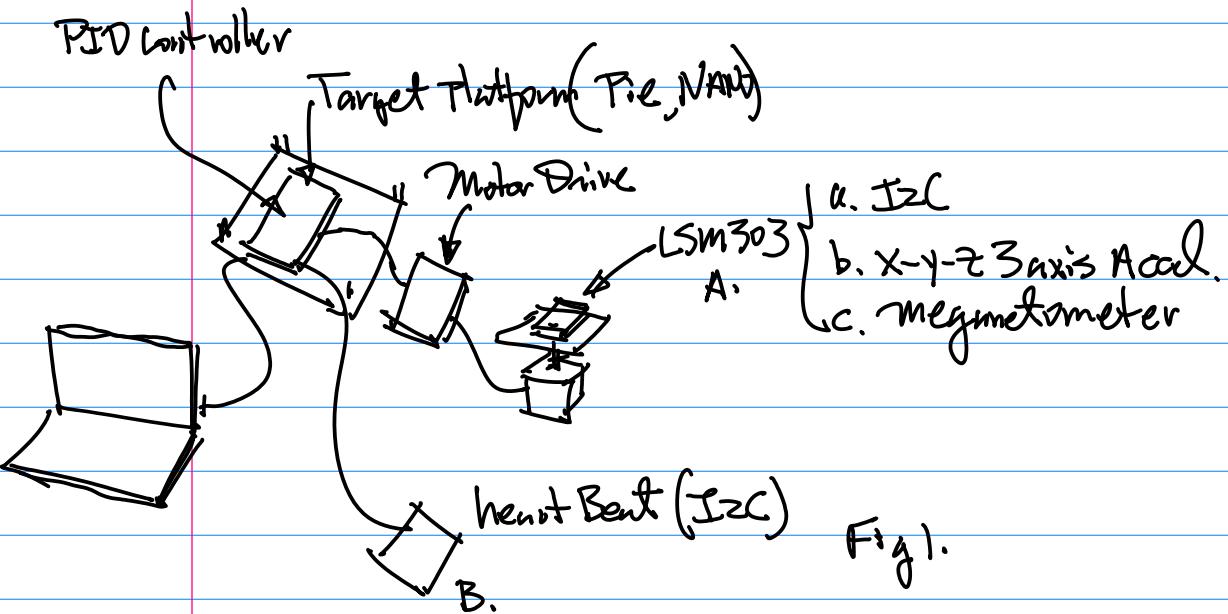
Program with the Kernel Space
Program(s), e.g. your device drivers

c. Embedded Nature of the class, requires

implementation that involves Sensor
interface (at least 2 I/F

Protocols, PWM, I2C on top of
GPIO, or ADC)

d. PID (Proportional, Integral,
Derivative) controller.



Option A: Program on the target

Platform to activate stepper

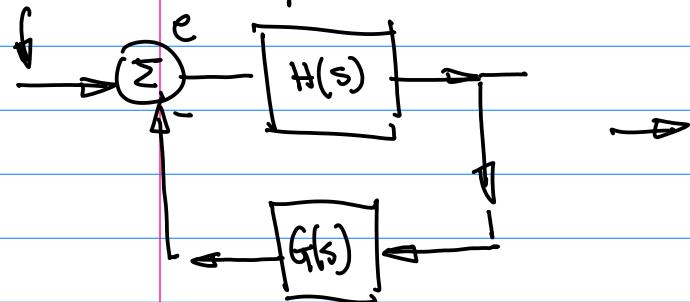
motor \rightarrow Capture Angular

Displacement information from LSM303

then feed the information into

PID controller,

Output (Nano) plant



feedBack

feedBack

$$K_d \frac{d^2 e(t)}{dt^2}$$

Design 3 controllers.

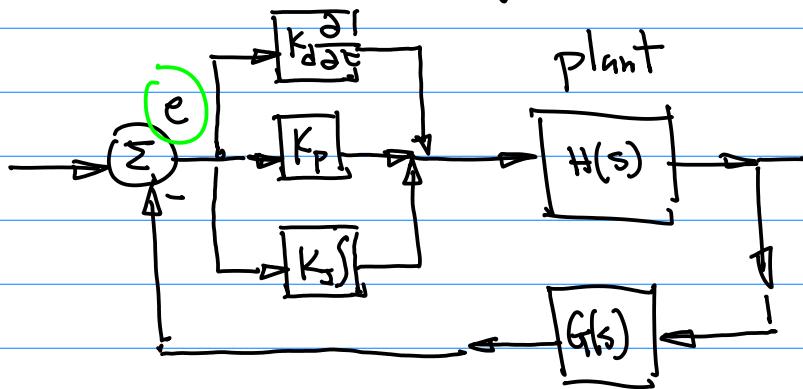


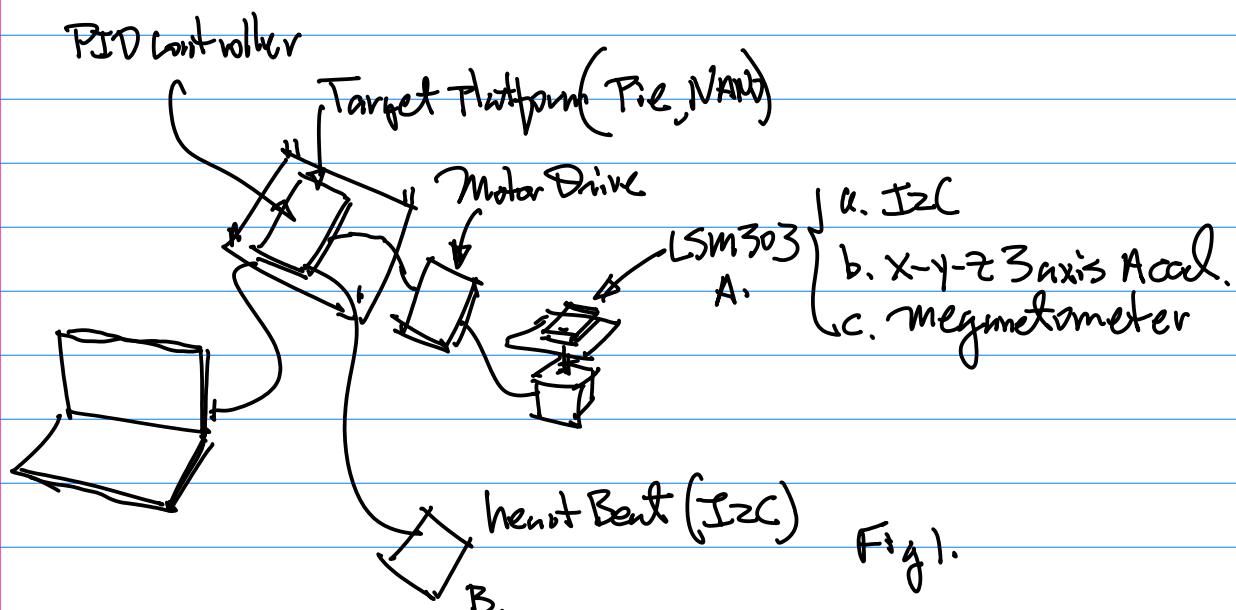
Fig 2-b. Implementation of this system is illustrated in Fig 1

Fig 1

Option B: Replace Sensor (Lsm303)

with HeartBeat Sensor. In this Case PID controller still has to be implemented with Angular Displacement from Simulation.

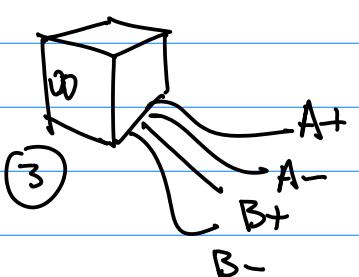
Example: Steppermotors & Steppermotor Drives.



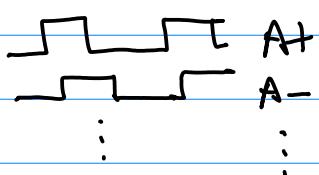
1. Steppermotors. NEMA ① 4 pins ($A+, A-, B+, B-$)

Dual Bi-Polar

② Current $\leq 1000 \text{ mA}$



③ Configuration of the motor is realized by coding / generating a combination of Square waves to connect to each pin, which in turn to drive the motor.



④ One Revolution 360° takes 200 Steps.

$$\frac{360}{200} = 1.8 \text{ Degree/Step, Full Step.}$$

- (b) Full Step 1.8 Degree
 $\frac{1}{2}$ Step 0.9 Degree
 $\frac{1}{4}$ Step 0.45 (in this class)
 $\frac{1}{8}$ " 0.225
 $\frac{1}{16}$ Step

2021F-117-#lec4-Stepp...

Stepper Motor Drive.

Define Driving Direction

Counter Clockwise — GPIO

Clockwise

How fast (what resolution) do you want to drive

the motor — Clock

(Pwm)

plus x pins for configuration

$$2^x \Big|_{x=1} = 2 \dots (1)$$

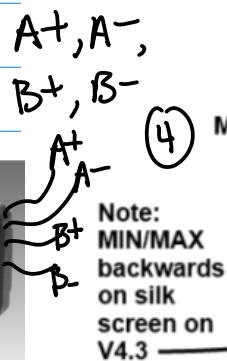
$$2^x \Big|_{x=2} = 4$$

R1
R2
SLEE
OUT:
LOI
SUPPL
GI

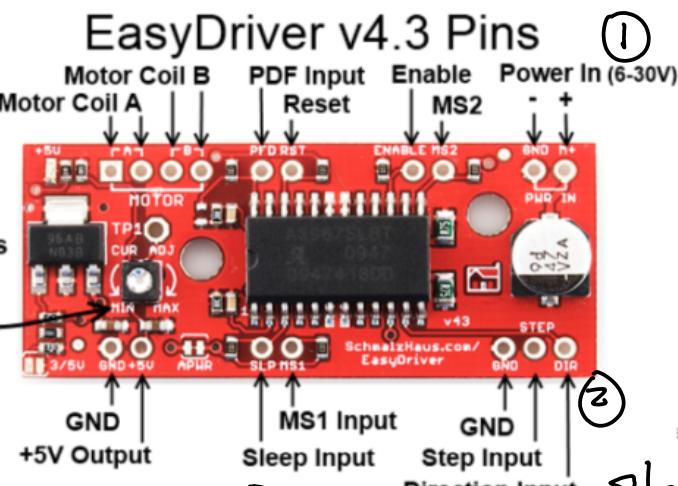
① Power input

StepInpt \rightarrow CLK (Pwm)

direction input \rightarrow GPIO



Configure the drive chip



Configuration pin

MS1 works together with

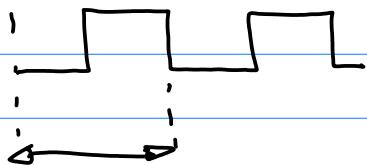
MS2. 2^x Config. pins $\rightarrow x=2$

$$2^x \Big|_{x=2} = 4. \text{ Full step, } \frac{1}{2}, \frac{1}{4}, \frac{1}{8} \text{ steps.}$$

Implementation of the Configuration Pins.

MS2	MS1	Configuration	
0	0	Config 1	Full
0	1	" 2	$\frac{1}{2}$
1	0	" 3	$\frac{1}{4}$
1	1	" 4	$\frac{1}{8}$

Note: PWM Signal Input to Motor Drive, set its duty cycle = 50%.



T : One period

t : Time Interval when the Output Signal is high.

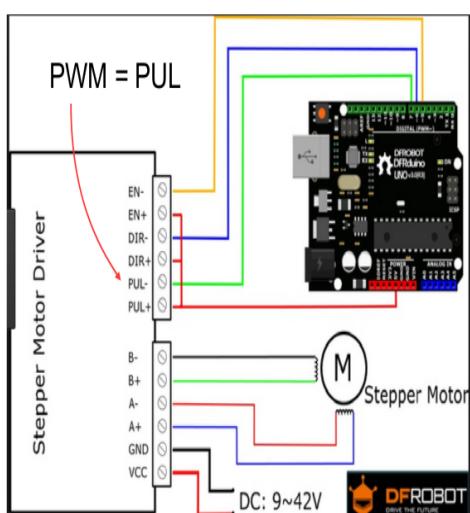
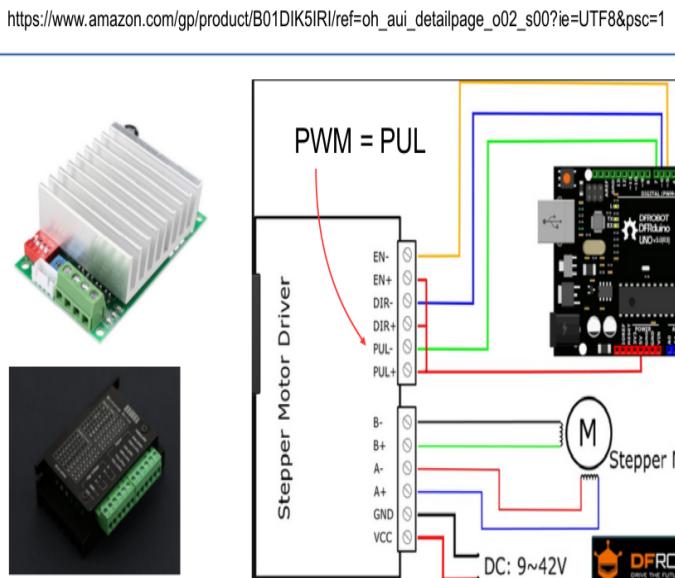
Hence, Duty Cycle $\equiv \frac{t}{T} \dots \%$

2021F-120-#lec16-0-2...

TB6600 4.5A CNC Single-Axis For LDR-40B

Stepper Motor Controller 1

15 USD



Nov. 10th.

Homework: Due a week from today

- Coordinate for each Team.
- Submit One Paragraph (21 words)

Proposal for your team for the semester long Project. Submit to CANVAS.

Consider Device Driver Development for the semester project

LPC1769 \rightarrow ARM11

CPU + Mem + Spr. Device Driver Development

+ IDE

- OS, Kernel Distribution
- Tool Chain
- make menuconfig.

Kconfig Script.



Kernel Space Programming

Example: GPIO Code
[PWM Code]

Example: ARM11 Device Driver Code.

Cmpe244

① Folder, OS Kernel
and ToolChain

40

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux$ ls
arm-qte-4.7.0          examples           u-boot-mi
arm-qt-extended-4.4.3   linux-2.6.38      x86-qte-
arm-qtopia              rootfs_qtopia_qt4  x86-qt-e>
busybox-1.17.2          rootfs_qtopia_qt4-s x86-qttopi
```

② examples

linux-2.6.38
rootfs_qtopia_qt4
rootfs_qtopia_qt4-s

u-boot-mi
x86-qte-
x86-qt-e>
x86-qttopi

C-Code Examples, in User Space, But Calling
DeviceDriver(s) in kernel space.

Now, Let's consider Device Driver. in this folder

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls
arch                      CREDITS
block                     crypto
Documentation             drivers
drivers                   Documentation
firmware                  fs
fs                        include
include                   init
init                      ipc
ipc                       Kbuild
Kbuild                   Kconfig
Kconfig                  kernel
kernel                   modules.builtin
modules.builtin           modules.order
modules.order              Module.symvers
Module.symvers            net
net                      README
README                   REPORTING-BUGS
REPORTING-BUGS            samples
samples                   scripts
scripts                   security
security                 sound
sound                    System.map
System.map                tools
```

③ drivers

Let's Look into Character Device "Char"

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers$ ls
accessibility  cpuidle  ieee802154  mmc          pps        telephony
acpi           crypto    infiniband  modules.builtin ps3        thermal
amba           dca      input       modules.order  rapidio    tty
ata            dio      isdn       mtd           regulator  uio
atm            dma      Kconfig     net           rtc       usb
auxdisplay    edac     leds       nfc           s390      uwb
base           eisa     lguest     nubus        sbus      vhost
block          firewire  macintosh  of            scsi      video
bluetooth     firmware  Makefile   oprofile    sfi       virtio
built-in.o     gpio      mca       parisc      sh        vlynq
cdrom          gpu       md        pci          sn        w1
char           hid       media     pcmcia    spi        watchdog
clk            hwmon    memstick  platform   ssb        xen
clocksource    i2c      message
```

④

⑤

Target CPU/Platform

```
(b) harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ ls
mini6410*
mini6410_adc.c          mini6410_hello_module.mod.c  mini6410_leds.o
mini6410_adc.mod.c       mini6410_hello_module.mod.o  mini6410_pwm2.c
mini6410_adc.o           mini6410_hello_module.o      mini6410_pwm2.mod.c
mini6410_buttons.c       mini6410_leds.c            mini6410_pwm.c
mini6410_buttons.o       mini6410_leds.ko           mini6410_pwmHarry.c
mini6410_hello_module.c  mini6410_leds.mod.c        mini6410_pwm.mod.c
mini6410_hello_module.ko mini6410_leds.mod.o
```

Simple "Hello, the world" Implementation

Kernel Space programming

```
(8) 1 #include <linux/kernel.h>
2 #include <linux/module.h>
3
4 Program Prototype Definition. __init name(void)
5 static int __init mini6410_hello_module_init(void)
6 {
7     printk("Hello, Mini6410 module is installed !\n");
8     return 0;
9 }
10
11 static void __exit mini6410_hello_module_cleanup(void)
12 {
13     printk("Good-bye, Mini6410 module was removed!\n");
14 }
15
16 module_init(mini6410_hello_module_init);
17 module_exit(mini6410_hello_module_cleanup);
18 MODULE_LICENSE("GPL");
```

b
Copy the build Device
Driver "Hello" into

Target platform, proton folder, such as
home/harry>Hello.ko

(Copy : tftpd, or USB Device, RS232 Serial
link)

\$ cp

(10) printf (Print in the
Kernel space)

a USE \$ make
menuconfig
+ To Build this
"Hello..."
module by
Selecting "M"
option.

Once it is copied into the
folder, then install it into
Kernel.

\$ insmod Hello.ko

The System will respond by printing
"Hello, ..."
the install is successful.

Cmpe244

42

Once we done with the module,
we can remove the module from
the kernel by

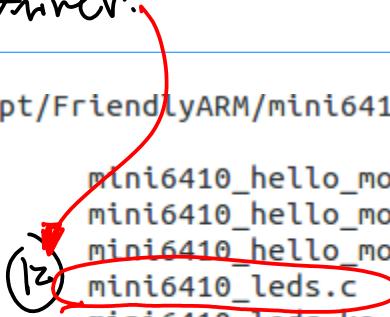
\$ rmmod hello.ko ↴

printed message to appear on
the terminal

"Good bye ... "

Example: Now, move from "Hello..."
to gpio device Driver.

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/lin
mini6410*
mini6410_adc.c          mini6410_hello_module.mod.c
mini6410_adc.mod.c      mini6410_hello_module.mod.o
mini6410_adc.o          mini6410_hello_module.o
mini6410_buttons.c       mini6410_leds.c
mini6410_buttons.o       mini6410_leds.Ko
mini6410_hello_module.c  mini6410_leds.mod.c
mini6410_hello_module.ko mini6410_leds.mod.o
```



'DEVICE_NAME' is defined here, note its connection to

(13) 32
33 #define DEVICE_NAME "leds0"
34

Kconfig file
which ties to
make menuconfig.

Function Prototype for "ioctl" module.

32 ioctl(fd, on, led_no);
 33 close(fd);
 34 (see pp. 4b)

```

(14) . DeviceID Userdefined Name      Par1          Par2
35 static long sbc2440_leds_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
36 {
37     switch(cmd) {
38         unsigned tmp;
39     case 0:
40     case 1:
41         if (arg > 4) {
42             return -EINVAL;
43         }
44         tmp = readl(S3C64XX_GPKDAT);
45         tmp &= ~(1 << (4 + arg));
46         tmp |= ( (!cmd) << (4 + arg) );
47         writel(tmp, S3C64XX_GPKDAT);
48         // printk (DEVICE_NAME": %d %d\n", arg, cmd);
49         return 0;
50     default:
51         return -EINVAL;
52     }
53 }
54
55 static struct file_operations dev_fops = {
56     .owner           = THIS_MODULE,
57     .unlocked_ioctl = sbc2440_leds_ioctl,
58 };
  
```

readl "long,32bits. Read General purpose
 Prt K, Data Register
 ↴ Need information of
 from the Control
 Registerisset.

Therefore, continues Line 66

Note the function prototype for init.

```

66 static int __init dev_init(void)
67 {
68     int ret;
69
70     {
71         unsigned tmp;
72         tmp = readl(S3C64XX_GPECON);
73         tmp = (tmp & ~0xffffU<<16))|(0x1111U<<16);
74         writel(tmp, S3C64XX_GPECON);
75
76         tmp = readl(S3C64XX_GPEDAT);
77         tmp |= (0xF << 4);
78         writel(tmp, S3C64XX_GPEDAT);
79     }
80     ret = misc_register(&misc);
81
82     printk (DEVICE_NAME"\nHarry: PGE initialized\n");
83
84     return ret;
85
86 }
  
```

Bitwise DR
Operation

Control Register of GPP, Part E.

(16) $\text{tmp} = (\text{tmp} \& \sim 0xffffU \ll 16) | (0x1111U \ll 16);$

Binary
 0001 0001 0001 0001

0000 ... 0000 1111

Bitwise "OR" Operation:

"~" Negation
 makes all 1s

Clear all bits

→ become all 0s.

Check GPP Port E, from GPU Datasheet, Lookup Control Register from the datasheet

01-Overview	60
02-MemoryM...	114
03-SystemCo...	120
04-MemoryS...	173
05-DRAMC	189
06-SROMCon...	205
07-OneNAND...	212
08-NFCON	245
09-CFCON	274
10-GPIO	306
11-DMAC	381
12-Vectored I...	409
13-Security S...	425
14-DisplayCo...	455
15-Post Proc...	536
16-TV Scaler	563

6410X_UM

10 GPIO

10.1 OVERVIEW

S3C6410 includes 187 multi-functional pins.

PortName	Number of Pins.
GPA port	8
GPB port	7
GPC port	8
GPD port	5
GPE port	5

Control Register for GPE

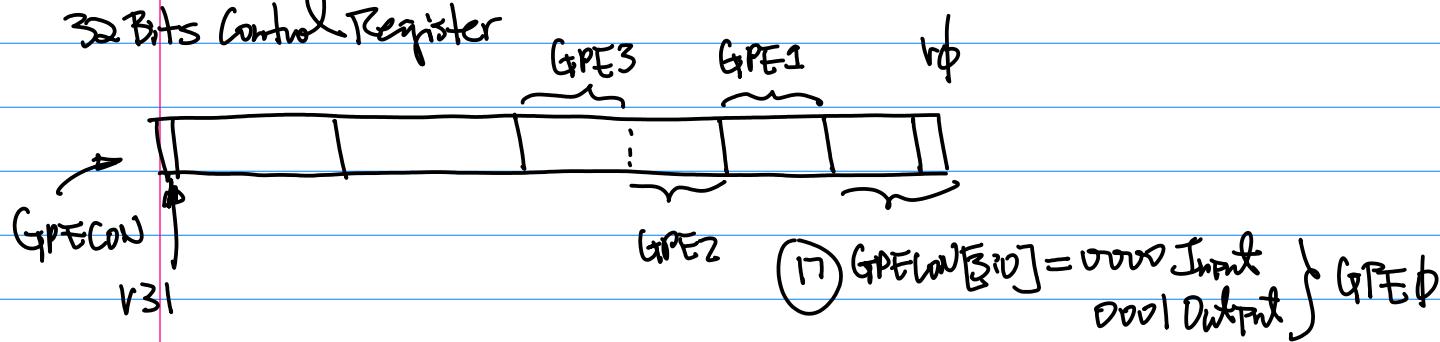
6410X_UM

10.5.5 PORT E CONTROL REGISTERS

There are five control registers including GPECON, GPEDAT, GPEPUD, GPECON: Port E Control Registers.

Register	Address	R/W	Description
GPECON	0x7F008080	R/W	Port E Configuration Register
GPEDAT	0x7F008084	R/W	Port E Data Register
GPEPUD	0x7F008088	R/W	Port E Pull-up/down Register
GPECONSLP	0x7F00808C	R/W	Port E Sleep mode Configuration
GPEPUDSLP	0x7F008090	R/W	Port E Sleep mode Pull-up/down

32 Bits Control Register



Related to (17)

GPDCON	Bit	Description	
GPE0	[3:0]	0000 = Input 0010 = PCM SCLK[1] 0100 = AC97 BITCLK 0110 = Reserved	0001 = Output 0011 = I2S CLK[1] 0101 = Reserved 0111 = Reserved
GPE1	[7:4]	0000 = Input 0010 = PCM EXTCLK[1] 0100 = AC97 RESETn 0110 = Reserved	0001 = Output 0011 = I2S CDCLK[1] 0101 = Reserved 0111 = Reserved
GPE2	[11:8]	0000 = Input 0010 = PCM FSYNC[1] 0100 = AC97 SYNC 0110 = Reserved	0001 = Output 0011 = I2S LRCLK[1] 0101 = Reserved 0111 = Reserved
GPE3	[15:12]	0000 = Input 0010 = PCM SIN[1] 0100 = AC97 SDI 0110 = Reserved	0001 = Output 0011 = I2S DI[1] 0101 = Reserved 0111 = Reserved
GPE4	[19:16]	0000 = Input 0010 = PCM SOUT[1] 0100 = AC97 SDO 0110 = Reserved	0001 = Output 0011 = I2S DO[1] 0101 = Reserved 0111 = Reserved

In User Space, Device Driver is defined here.

harry@harry-laptop: /opt/FriendlyARM/minibl

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/ioctl.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
8
9 int main(int argc, char **argv)
10 {
11     int on;
12     int led_no;
13     int fd;
14
15     printf("hello\n"); //Feb. 16, 2
16
17     if (argc != 3 || sscanf(argv[1])
18         on < 0 || on >
19         fprintf(stderr, "Usage:
20         exit(1);
21     }
22
23     fd = open("/dev/leds0", 0);

```

minibl-leds.c

from Device Driver code

```

29 #include <plat/gpio-cfg.h>
30 #include <mach/gpio-bank-e.h>
31 #include <mach/gpio-bank-k.h>
32
33 #define DEVICE_NAME "leds0"

```

46

From user program, the access to Device GPP for I/O operations
is realized by ioctl(fd, ^{Par1}Par2, [¶]Par3)

pointing to
the Device

```
32     ioctl(fd, on, led_no);  
33     close(fd);  
34
```