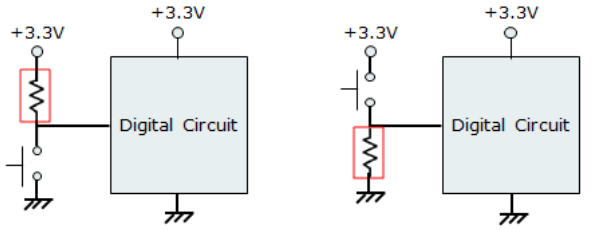
20190611 2일차 수업

http://d.pr/n9p21YB+

풀업/풀다운 저항과 회로

- 디지털 회로의 입력 단자는 내부 임피던스가 높기 때문에 입력 단자가 어디에도 연결되어 있지 않으면 근처의 정전기나 전자기 유도에 의하여 예상할 수 없는 전압이 인가될 수 있다
- 외부로 연결되는 입력 단자는 커넥터가 잘못 연결되어 비정상적인 전압이 인가되지 않도록 해야 한다. 즉, 플로팅 되지 않도록 해야한다



풀업 저항 회로 구성　　풀다운 저항 회로 구성

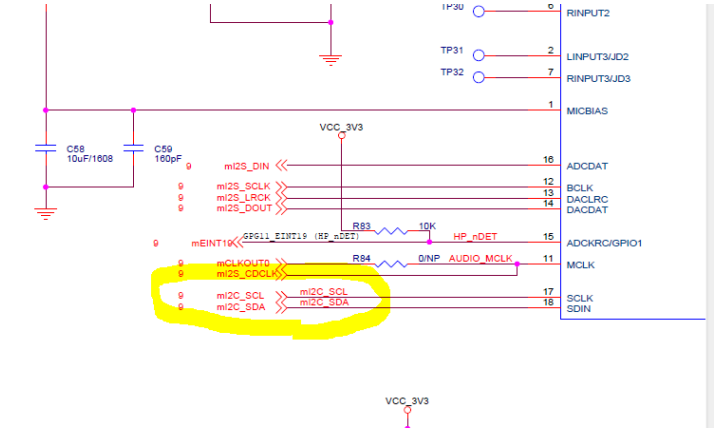
입력단자 에 저항이 붙어있다.

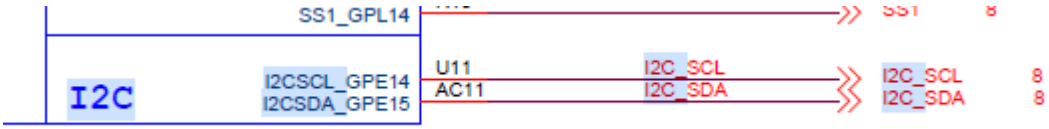
I/O PORTS　　S3C2450X RISC MICROPROCESSOR

GPGDAT	Bit	Description
Reserved	[31:16]	Reserved
GPG[15:0]	[15:0]	When the port is configured as an input port, the corresponding bit is the pin state. When the port is configured as an output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.
GPGUDP	**Bit**	**Description**
GPGUDP15 ~ GPGUDP0	[31:30] ~ [1:0]	[CPU:CPD] 00 = pull-up/down disable 01 = pull-down enable 10 = pull-up enable 11 = not-available

SOC들은 풀업저항이 대부분 들어가있다.

GPGUDP로 제어할수있다.





시리얼 통신 데이터를 보내 받을 때 쓰는 것 I2C

하나는 클럭 하나는 데이터

I2C는 꼭 풀업저항을 달아야되는 통신장치

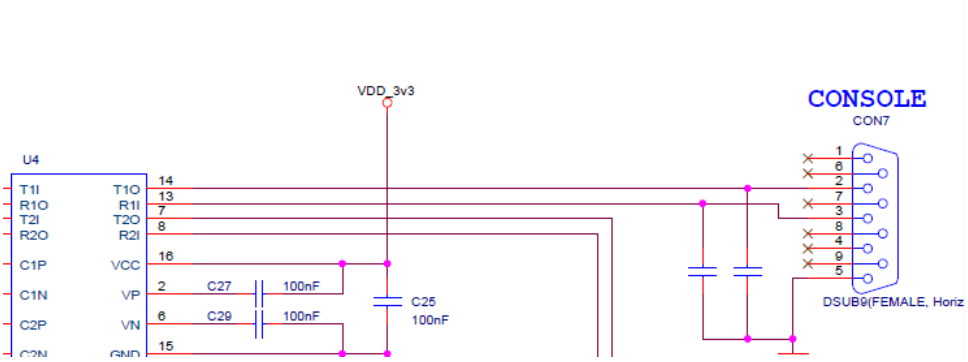
풀업안붙여놓으면 작동안됨

GPECON	Bit	Description
GPE15	[31:30]	00 = Input 10 = IICSDA 01 = Output 11 = Reserved
GPE14	[29:28]	00 = Input 10 = IICSCL 01 = Output 11 = Reserved
GPE13	[27:26]	00 = Input 10 = SPICLK0 01 = Output 11 = Reserved

10 IICSCL 얼터네이티브

==========

P91 교제



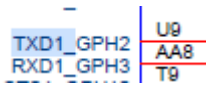
2번 3번

rs232c 트랜시버 보내고 받는 장치

12볼트로 보내고 sp3232Ecy에서 받는데 전압을 낮춰 3볼트로 데이터가 깨지지않게 한다.

케이블 길이가 길기 때문에 세게 보냄

rs232c 사용하려면 있어야함!



기능이 2개잇는걸 얼터네이티브 라함 조심히 셋팅해야됨

저 두개가 UART임 SOC안에 들어잇음
데이터를 보내고 받고를 시리얼로 하는 것이 UART이다.

==========

실습 리눅스 printf해서 보기
d.pr/i/KAQLxD+
user@linux:~/Desktop/m2450/lab/student/04_DeviceControl_Student$ make

user@linux:~/Desktop/m2450/lab/student$ subl 04_DeviceControl_Student/
main에서

UART_TEST부분

==========
헥사덤프
바이너리 파일에서 16진수로 보여주지만 모든 것이 다 16진수는아니며
아스키코드의 문자일수도 있다.
34가 숫자 34일지 아스키코드 4일지 모른다.
user@linux:~/Desktop/m2450/lab/student/04_DeviceControl_Student$ hexdump -C exception.o |more

아스키코드0x0a 는 '₩n'이다.

데이터를 받는 방식인 동기와 비동기. 이 둘의 개념에 대해 설명하는 게시물은 매우 많은데 프로그래밍적으로 생각했을 때 이해가 가지 않아서 쉽게 이해를 할 수 있는 동기와 비동기의 예가 어떤것들이 있는지 검색을 해봤습니다.

동기(synchronous : 동시에 일어나는)
- 동기는 말 그대로 동시에 일어난다는 뜻입니다. 요청과 그 결과가 동시에 일어난다는 약속인데요. 바로 요청을 하면 시간이 얼마가 걸리던지 요청한 자리에서 결과가 주어져야 합니다.
  -> 요청과 결과가 한 자리에서 동시에 일어남
  -> A노드와 B노드 사이의 작업 처리 단위(transaction)를 동시에 맞추겠다.

비동기(Asynchronous : 동시에 일어나지 않는)
- 비동기는 동시에 일어나지 않는다를 의미합니다. 요청과 결과가 동시에 일어나지 않을거라는 약속입니다.
  -> 요청한 그 자리에서 결과가 주어지지 않음
  -> 노드 사이의 작업 처리 단위를 동시에 맞추지 않아도 된다.

**
동기와 비동기는 상황에 따라서 각각의 **장단점**이 있습니다.
동기방식은 설계가 매우 간단하고 직관적이지만 결과가 주어질 때까지 아무것도 못하고 대기해야 하는 단점이 있고,
비동기방식은 동기보다 복잡하지만 결과가 주어지는데 시간이 걸리더라도 그 시간 동안 다른 작업을 할 수 있으므로 자원을 효율적으로 사용할 수 있는 장점이 있습니다.

========== o

UART 제어

https://blog.naver.com/guile21c/52117520

amba bus 설명 참고

*AMBA 버스 규격*arm에서 만듬

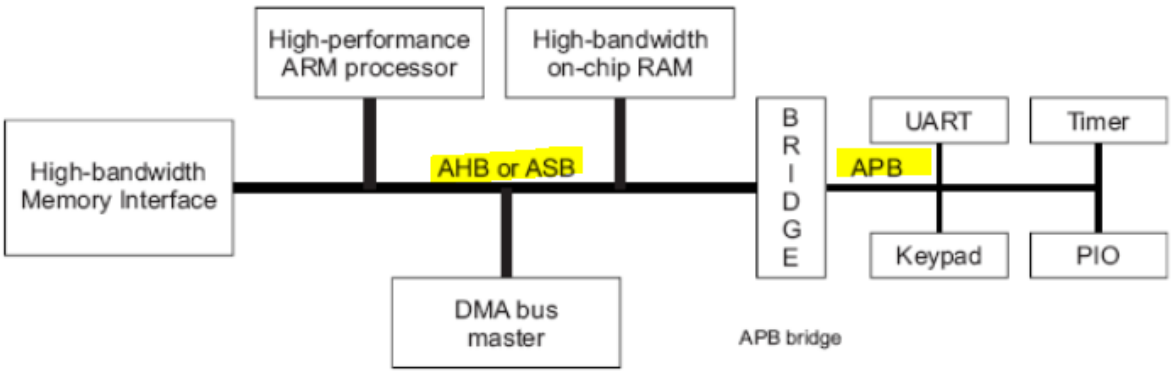
USB

PCI

I2C

버스 정보가 다니는 파이프

soc에서 규격을 지킴.



AHB는 고속도로 cpu가 연결되어잇음 HCLK

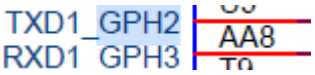
APB는 국도(low speed 버스) PCLK

Uart는 국도정도 속도

클락이란

Cpu(SOC)에 전원이 들어가야됨
전압은 조금씩다름
크리스탈이 달려서 클럭이 들어감
그래서 cpu의 스피드를 조절해준다.
빠르면 빠르게 계산함.

클럭은 다 버스마다 공유되어있어서 속도는 버스에 연결되어있는것마다
정해져잇다.
==========
Uart 이닛 uart.c 에서 설정

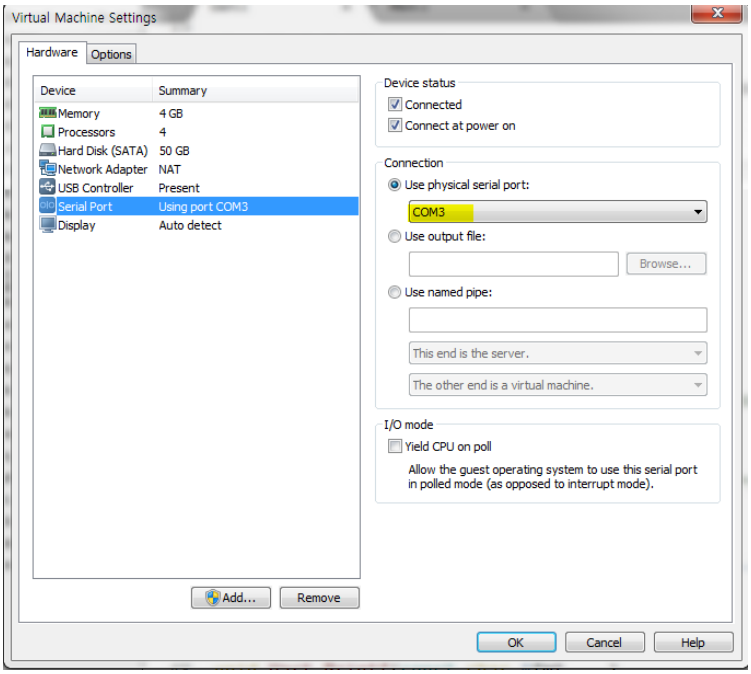


두개가 얼터네이티브를 수정해서 설정해야됨
rGPHCON = (rGPHCON & ~(0xF<<4))|(0xA<<4);

초기화는 전원이 들어오면 부트로더가 초기화셋팅을 한다.

매뉴얼은 필할 때 부분만 추출해서 읽어라.
레지스터값이 0이면 사용하지않는다는 의미

미니컴
Player- file – preference- device 인에이블 체크확인



```
m
subl ~/.bashrc
source ~/.bashrc
```

Loopback 자기 자신에서 echo함

소프트웨어가 문제인지 하드웨어가 문제인지

소프트웨어 스스로 루프백으로 검증이가능함

컨트롤레지스터

UCONn	Bit	Description	Initial State
Transmit Mode (note 3)	[3:2]	Determine which function is currently able to write Tx data to the UART transmit buffer register. 00 = Disable 01 = Interrupt request (note 6) or polling mode 10 = DMA request(request signal 0) 11 = DMA request(request signal 1)	00
Receive Mode	[1:0]	Determine which function is currently able to read data from UART receive buffer register. 00 = Disable (note 4) 01 = Interrupt request or polling mode 10 = DMA request(request signal 0) 11 = DMA request(request signal 1)	00

NOTES:

```
void Uart_Send_Byte(int data)
{
    if(data=='\n')
    {
        while(!(rUTRSTAT1 & 0x2));
        WrUTXH1('\r');
    }

    while(!(rUTRSTAT1 & 0x2));
    WrUTXH1(data);
```

0x2는 10 두번재 비트

UTRSTAT1	0x50004010	R	UART channel 1 Tx/Rx status register	
UTRSTAT2	0x50008010	R	UART channel 2 Tx/Rx status register	0x6
UTRSTAT3	0x5000C010	R	UART channel 3 Tx/Rx status register	0x6

UTRSTATn	Bit	Description	Initial State
Transmitter empty	[2]	Set to 1 automatically when the transmit buffer register has no valid data to transmit and the transmit shift register is empty. 0 = Not empty 1 = Transmitter (transmit buffer & shifter register) empty	1
Transmit buffer empty	[1]	Set to 1 automatically when transmit buffer register is empty. 0 =The buffer register is not empty 1 = Empty (In Non-FIFO mode, Interrupt or DMA is requested. In FIFO mode, Interrupt or DMA is requested, when Tx FIFO Trigger Level is set to 00 (Empty)) If the UART uses the FIFO, users should check Tx FIFO Count bits and Tx FIFO Full bit in the UFSTAT register instead of this bit.	1
Receive buffer data ready	[0]	Set to 1 automatically whenever receive buffer register contains valid data, received over the RXDn port. 0 = Empty 1 = The buffer register has a received data (In Non-FIFO mode, Interrupt or DMA is requested) If the UART uses the FIFO, users should check Rx FIFO Count bits and Rx FIFO Full bit in the UFSTAT register instead of this bit.	0

관련된 레지스터의 첫데이터 시트 페이지는 꼭 읽는게 좋다.

0x2니까 1번비트인 트렌지스트버퍼 관련한것이다.

tx버퍼가 비엇는지 확인하고 보내야된다.

1번비트가 1이면 비엇다는 것

0이면 비어잇지않다는 것. (무언가 보내고있는중임.)

==========

https://ruddyscent.blogspot.com/2010/12/blog-post.html

리눅스와 윈도우의 줄바꿈 차이점

'₩n' 개행

'₩r' 캐리지리턴 커서를 홈으로 (여러 번해도상관없당)

	개행 표기
Windows	₩r₩n
Unix(Linux 및 Mac OS X 포함)	₩n
Classic Mac OS(Mac OS 9까지)	₩r

윈도우는 0d 0a
리눅스는 0a
이런차이를 극복하기위해서 캐리지리턴을 넣는다.

Uart 표준
115200n81
패리티 없음
데이터비트8
정지비트1

Cpu 속도 534000000
Uart 속도 115200

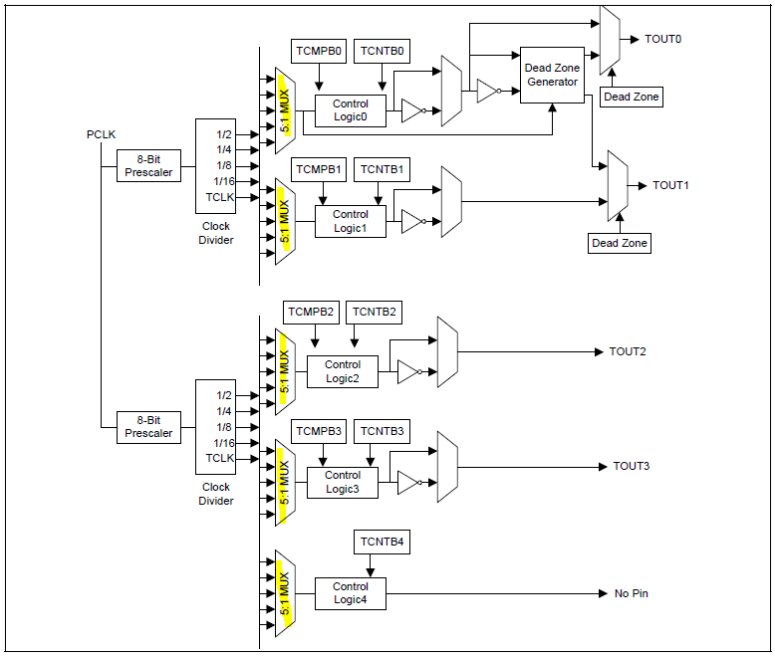
한글자 보낼때마다 기다린다. 버퍼에서
Cpu가 그동안 논다.

버퍼에 문자열을 다 집어넣고 보낸다 이게 피포모드
좋은것같지만 쓰지않는다.
글자를 뿌리는 것이 목적이기에 소프트의 호환성 때문에
피포모드를 쓰지않는다.
피포를 안쓰는 것이 일반적

프로젝트 피포 모드가 작동하도록 Uart를 구성하시오.
하이퍼터미널을 통해서 파일 보내기

루프백

==========



타이머 장치 응용

1) 시간지연(delay)
2) 시간측정
3) PWM 신호발생 (시간과는 이질적)

4번타이머는 시간지연과 시간측정만 쓸수있다.

0~3번은 pwm 핀이 잇다.

PCLK 66M

8bit(0~255) 프리스케일러는 나눈다

프리스케일은 작게 만드는 것이 목적이다.

타이머에 공급되는 클락의 수를 다운시킴

66을 6.6으로

이후에 1/2 , 1/4...나눠서 들어간다.

프리스케일과 잘나눠서 1kHz를 만들수 있다.

1000분의 1초마다 TCNB0의 값이 10일 때

다운카운팅하게함.

10,9,8,...1,0까지

0이되면 인터럽트가 발생하게 끔함.

레지스터의 표현할 수 있는 수의 범위가 있기 때문에

TCNTB0에 100000로 100초를 나누느것이 불가능할수있다.

10초를 구현하기위해서는 프리스케일 divider TCNTB0의 값을

조절하여 여러 경우로 만들수있다.

TCNTB0는 프리로드 카운터 레지스터

시간측정 방법은
65336(2의16승)(TCNTB0의 최대값)– 카운터 여기에 *주파수하면 시간을 잴수있다.

타임아웃이되면 0이된다. high에서 low로 TOUT0

먹스에서 뒤집혀서 반대로 나갈수잇다.
또한 반주기로 뒤집혀서 나갈수 있다.

TCMPB0에 5000을 주고 다운카운팅할 때 일치하는 경우가 발생
Match event (TCMPB0 == TCNTB0)
TCMPB0가 2000이면 반주기 비율이 달라졋다.
DUTY RATIO 가 5:5에서 2:8

Pulse width modulation

TOUT0에 LCD를 연결하면 켜고 꺼지는데
비율에따라 밝기가 차이가남
2:8이 더 밝음
이렇게 밝기 조절을 할수 있다.
모니터, 손전등 밝기조절의 원리이다.

데드존 제너레이터란
펄스신호가 있다. 위상(phase)의 차이가 180도가 나는 2개의 신호가 잇다.
1이고 하나는0일 때
동시에 1과0을 받아야할 때 데드존을 이용한다.
그래서 분리되서 받는다 TOUT0 와 데드존을 통한 TOUT1
==========ㅇ

Timer input clock Frequency = PCLK / {prescaler value+1} / {divider value}
{prescaler value} = 0~255
{divider value} = 2, 4, 8, 16

목표주파수를 먼저 설정하고 값을 정하는것이다.
엑셀메크로 추천함

TCNTO0 카운팅 읽기전용 레지스터임

```
rTCON |= (1<<1)|(0); //manual update
        rTCON &= ~(1<<1);

        rTCON |= 1;         //trigger timer start

        while(rTCNTO0 != 0); //rTCNTO0
```

타이머 0번 구현 끝~