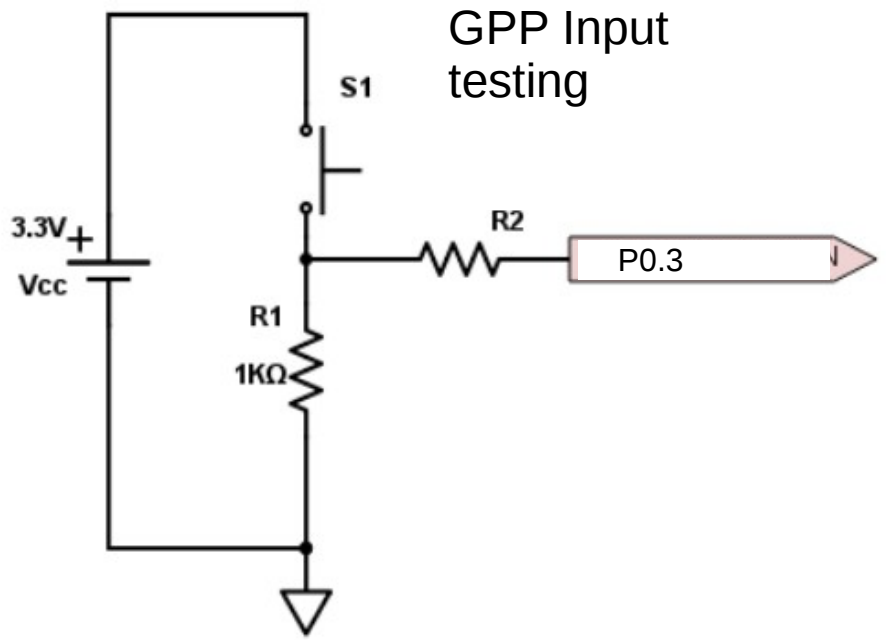
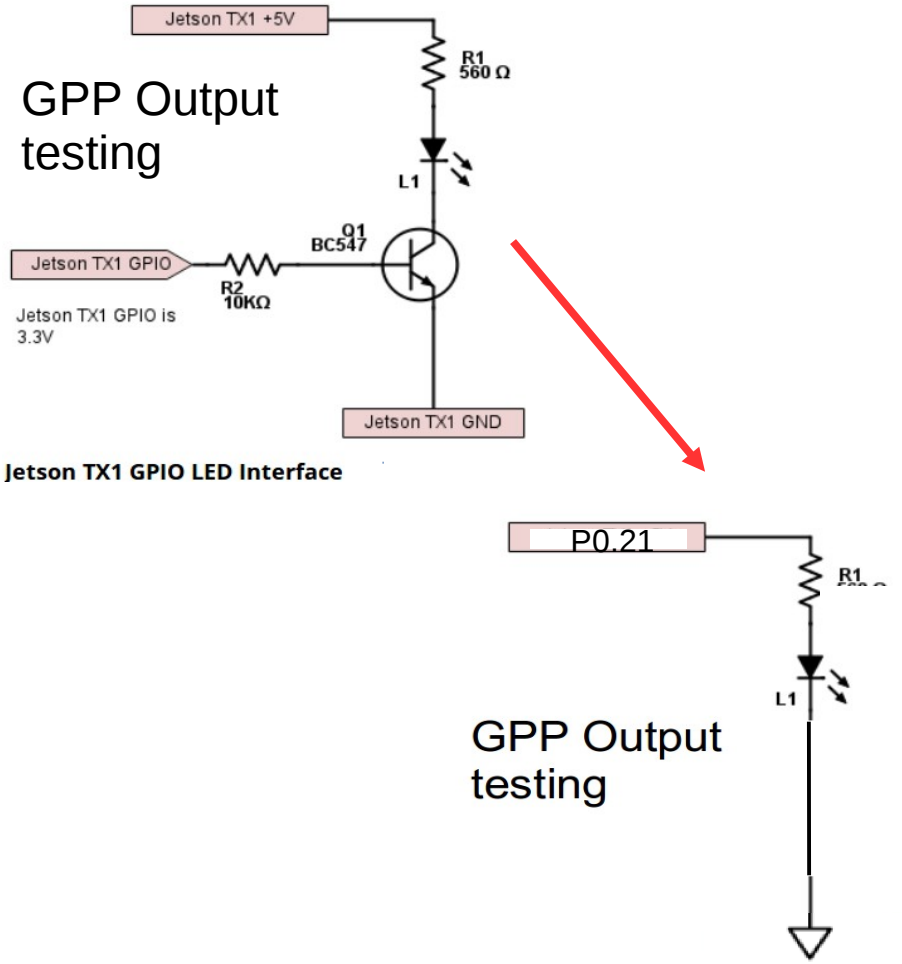
GPP I/O Testing Reference Design



Schematic GPIO



Jetson TX1 GPIO LED Interface

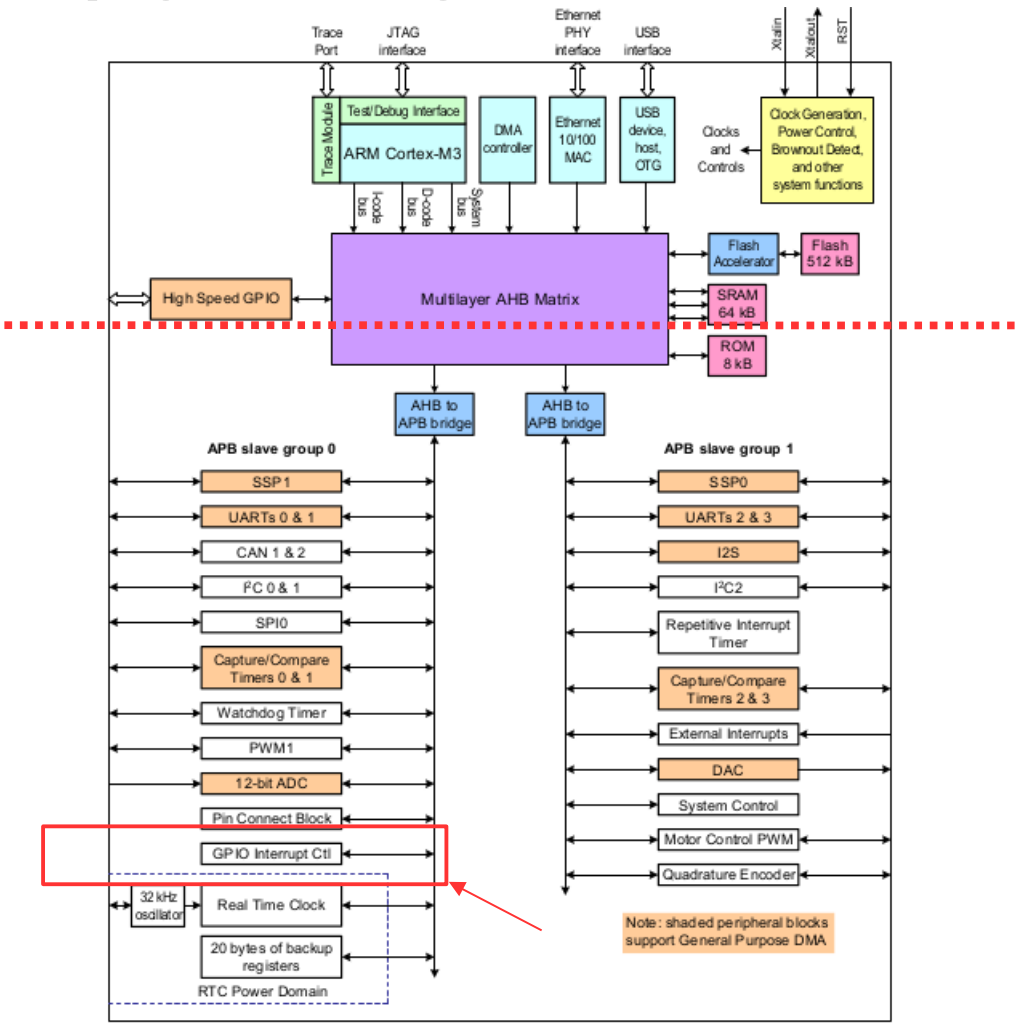
http://www.jetsonhacks.com/2015/12/29/gpio-interfacing-nvidia-jetson-tx1/

CPU GPP I/O Pins

LPCXpresso			Dual row holes (2x27), 100 mil spacing
GND		GND	J2-1
VIN (4.5-5.5V)		EXT_VIN	J2-2
VB (battery supply)		VBAT	J2-3
RESET_N		TARGET_RESET	J2-4
P0.9	MOSI1	P0.9	J2-5
P0.8	MISO1	P0.8	J2-6

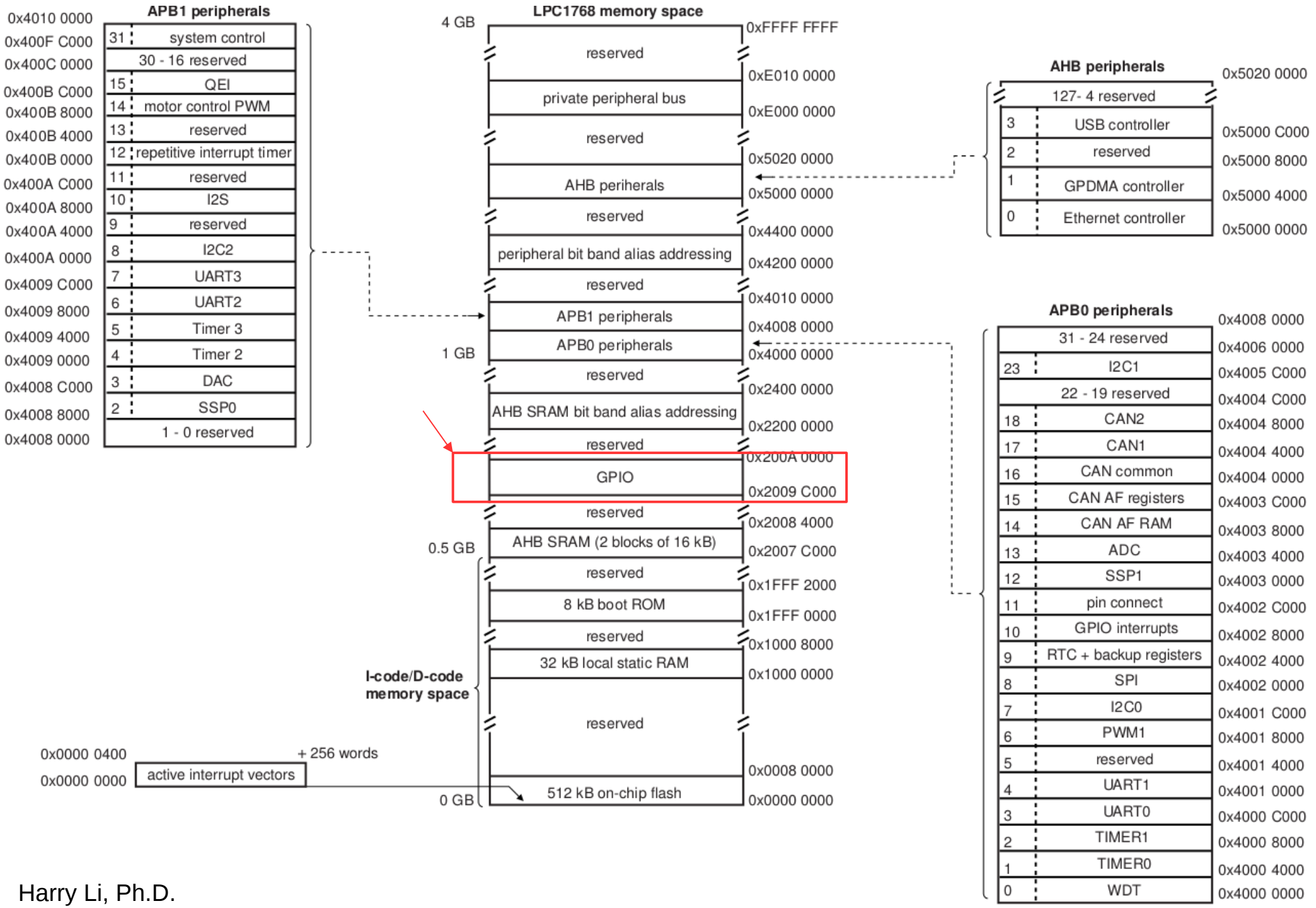
P1.30	AD0.4	P1.30	J2-19
P1.31	AD0.5	P1.31	J2-20
P0.2		P0.2	J2-21
P0.3		P0.3	J2-22
P0.21		P0.21	J2-23
P0.22		P0.22-RED_LED	J2-24
P0.27		P0.27-I2C_SDA	J2-25
P0.28		P0.28-I2C_SCL	J2-26
P2.13		P2.13	J2-27

Reference: from SCH pdf document, CPU module rev. D



CPU architecture, from CPU datasheet, pp. 9

Memory Map


APB1 peripherals
0x4010 0000
0x400F C000
0x400C 0000
0x400B C000
0x400B 8000
0x400B 4000
0x400B 0000
0x400A C000
0x400A 8000
0x400A 4000
0x400A 0000
0x4009 C000
0x4009 8000
0x4009 4000
0x4009 0000
0x4008 C000
0x4008 8000
0x4008 0000
31 system control
30 - 16 reserved
15 QEI
14 motor control PWM
13 reserved
12 repetitive interrupt timer
11 reserved
10 I2S
9 reserved
8 I2C2
7 UART3
6 UART2
5 Timer 3
4 Timer 2
3 DAC
2 SSP0
1 - 0 reserved
LPC1768 memory space
4 GB
0xFFFF FFFF
reserved
0xE010 0000
private peripheral bus
0xE000 0000
reserved
0x5020 0000
AHB periherals
0x5000 0000
reserved
0x4400 0000
peripheral bit band alias addressing
0x4200 0000
reserved
0x4010 0000
APB1 peripherals
0x4008 0000
APB0 peripherals
1 GB
0x4000 0000
reserved
0x2400 0000
AHB SRAM bit band alias addressing
0x2200 0000
reserved
0x200A 0000
GPIO
0x2009 C000
reserved
0x2008 4000
AHB SRAM (2 blocks of 16 kB)
0.5 GB
0x2007 C000
reserved
0x1FFF 2000
8 kB boot ROM
0x1FFF 0000
reserved
0x1000 8000
32 kB local static RAM
0x1000 0000
I-code/D-code memory space
reserved
0x0008 0000
512 kB on-chip flash
0 GB
0x0000 0000
0x0000 0400
+ 256 words
active interrupt vectors
0x0000 0000
AHB peripherals
0x5020 0000
127- 4 reserved
3 USB controller
0x5000 C000
2 reserved
0x5000 8000
1 GPDMA controller
0x5000 4000
0 Ethernet controller
0x5000 0000
APB0 peripherals
0x4008 0000
31 - 24 reserved
0x4006 0000
23 I2C1
0x4005 C000
22 - 19 reserved
0x4004 C000
18 CAN2
0x4004 8000
17 CAN1
0x4004 4000
16 CAN common
0x4004 0000
15 CAN AF registers
0x4003 C000
14 CAN AF RAM
0x4003 8000
13 ADC
0x4003 4000
12 SSP1
0x4003 0000
11 pin connect
0x4002 C000
10 GPIO interrupts
0x4002 8000
9 RTC + backup registers
0x4002 4000
8 SPI
0x4002 0000
7 I2C0
0x4001 C000
6 PWM1
0x4001 8000
5 reserved
0x4001 4000
4 UART1
0x4001 0000
3 UART0
0x4000 C000
2 TIMER1
0x4000 8000
1 TIMER0
0x4000 4000
0 WDT
0x4000 0000

Mapping Hardware to Software

From lpc17xx.h find the mapping architecture

```
/*****************************************************************************/
/*                         Peripheral memory map                             */
/*****************************************************************************/
/* Base addresses   */
#define LPC_FLASH_BASE      (0x00000000UL)
#define LPC_RAM_BASE        (0x10000000UL)
#define LPC_GPIO_BASE       (0x2009C000UL)
#define LPC_APB0_BASE       (0x40000000UL)
#define LPC_APB1_BASE       (0x40080000UL)
#define LPC_AHB_BASE        (0x50000000UL)
#define LPC_CM3_BASE        (0xE0000000UL)
/* APB0 peripherals                                                          */
#define LPC_WDT_BASE        (LPC_APB0_BASE + 0x00000)
#define LPC_TIM0_BASE       (LPC_APB0_BASE + 0x04000)
#define LPC_TIM1_BASE       (LPC_APB0_BASE + 0x08000)
#define LPC_UART0_BASE      (LPC_APB0_BASE + 0x0C000)
```

From lpc17xx.h, partial

```
/*General Purpose Input/Output (GPIO) */
typedef struct
{
  union {
    __IO uint32_t FIODIR;
    struct {
      __IO uint16_t FIODIRL;
      __IO uint16_t FIODIRH;
    };
    struct {
      __IO uint8_t  FIODIR0;
      __IO uint8_t  FIODIR1;
      __IO uint8_t  FIODIR2;
      __IO uint8_t  FIODIR3;
    };
  };
```

From lpc17xx.h, partial

Program GPP Port (1)

From lpc17xx.h find the mapping architecture

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>
#include <stdio.h>

//Initialize the port and pin as outputs.
void GPIOinitOut(uint8_t portNum, uint32_t pinNum)
{
	if (portNum == 0)
	{
		LPC_GPIO0->FIODIR |= (1 << pinNum);
	}
	else if (portNum == 1)
	{
		LPC_GPIO1->FIODIR |= (1 << pinNum);
	}
	else if (portNum == 2)
	{
		LPC_GPIO2->FIODIR |= (1 << pinNum);
	}
	else
	{
		puts("Not a valid port!\n");
	}
}
```

```
void setGPIO(uint8_t portNum, uint32_t pinNum)
{
	if (portNum == 0)
	{
		LPC_GPIO0->FIOSET = (1 << pinNum);
		printf("Pin 0.%d has been set.\n",pinNum);
	}
	//Can be used to set pins on other ports for future
	else
	{
		puts("Only port 0 is used, try again!\n");
	}
}

//Deactivate the pin
void clearGPIO(uint8_t portNum, uint32_t pinNum)
{
	if (portNum == 0)
	{
		LPC_GPIO0->FIOCLR = (1 << pinNum);
		printf("Pin 0.%d has been cleared.\n", pinNum);
	}
	//Can be used to clear pins on other ports for future
	else
	{
		puts("Only port 0 is used, try again!\n");
	}
}
```

Program GPP Port (2)

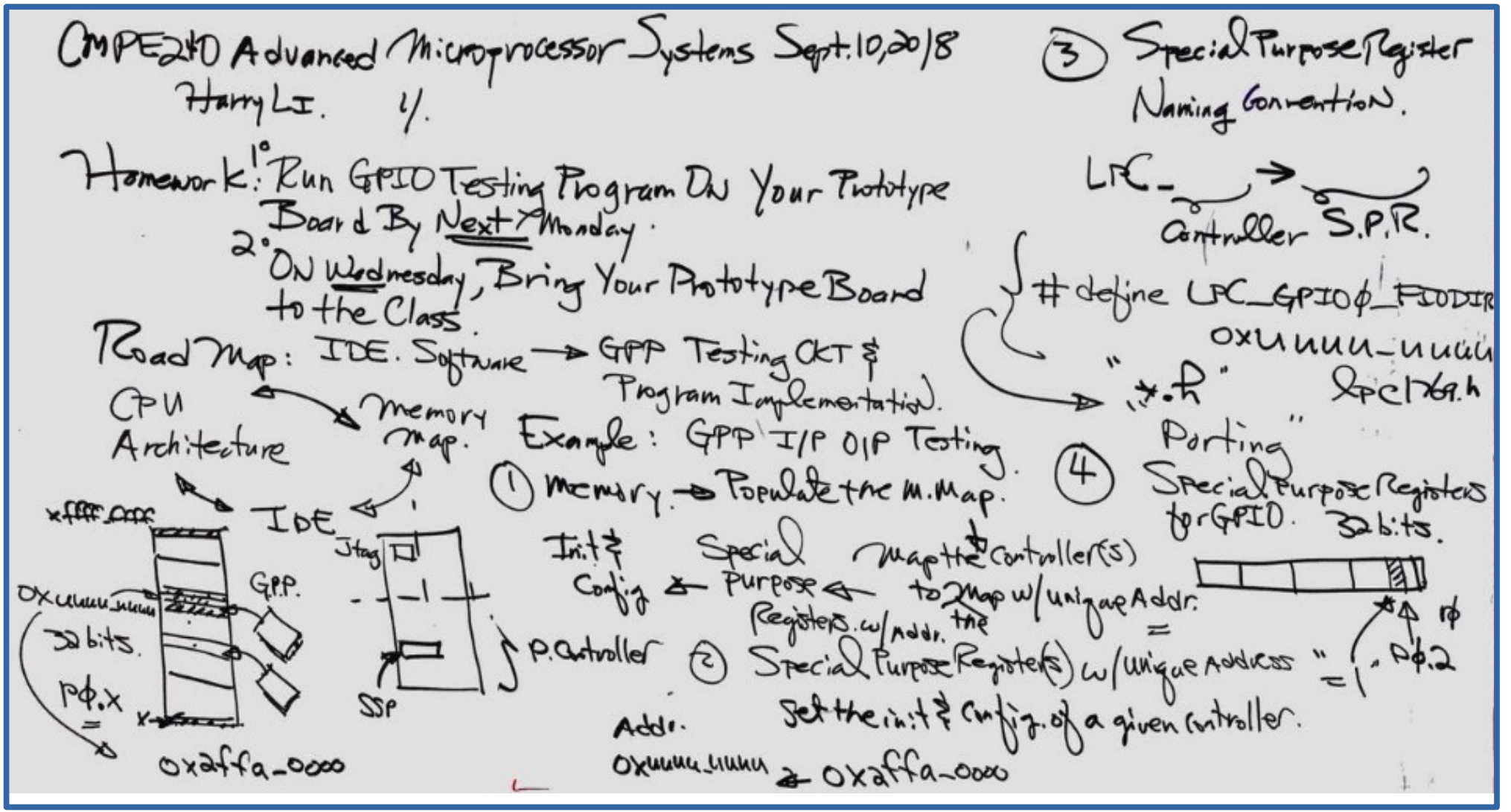
From lpc17xx.h find the mapping architecture

```
int main(void)
{
    // Force the counter to be placed into memory
    volatile static int i = 0 ;
    //Set pin 0.2 as output
        GPIOinitOut(0,2);
        //Set pin 0.3 as output
        GPIOinitOut(0,3);

    while(1)
    {
        printf("Enter a command to activate LED1(1)
                & LED2(2) or both(3).\n");
       scanf("%d", &i);

                if (i == 1)
                {
                        //Activate pin 0.2
                        setGPIO(0,2);
                }
                else if (i == 2)
                {
                        //Activate pin 0.3
                        setGPIO(0, 3);
                }
                else if (i == 3)
                {
                        //Activate both pins
                        setGPIO(0, 2);
                        setGPIO(0, 3);
                }
                else
                {
                        puts("Not a valid option!\n");
                }
                        //Deactivate pins
                        clearGPIO(0, 2);
                        clearGPIO(0, 3);
    }
    //0 should never be returned, due to infinite while loop
    return 0;
}
```

9-10-2018 GPP I/O Testing



9-10-2018 GPP I/O SPR FIODIR