The Virtual Shed

Robotics and Embedded Systems

Saturday, 16 March 2013

PlayStation 2 Controller Interface

The PlayStation2 controller has two analog joysticks and 14 buttons, making it an excellent tool for remotely controlling a robot (such as for example a hexapod). This entry describes how a PlayStation2 controller can be interfaced to a STM32 microcontroller and draws heavily on the information provided here and here.

The SPI peripheral on the STM32F4 will be used to communicate to the PS2 controller. Those of you who have a LeafLab maple can find a library that allows communication with the PS2 controller here. This library also uses SPI to communicate with the PS2 controller.

The following should be noted when interfacing the PS2 controller to the microcontroller

- The controller appears to require a minimum of 3.3V on its GPIO. I used a CMOS hex buffer (part number CD4050B) as a logic converter because the STM32F4 GPIOs operate at 3V.
- The PS2 has open collector outputs, which requires the use of pull-up resistors (10K resistors seem to work well).
- Wiring should be as follows
  - Data (PS2) to the SPI MISO (microcontroller)
  - Command (PS2) to SPI MOSI (microcontroller)
  - Clock (PS2) to SPI SCLK (microcontroller)
  - Attention (PS2) to whichever GPIO will be configured for this purpose (microcontroller)
  - Data, Command, Clock and Attention are the absolute minimum connections required
- The PS2 controller pulls down and releases the Acknowledge signal when it is ready to receive and process bytes in a multi-byte package. This was accommodated with a 15us delay between bytes. Whilst this is the simplest solution, it is not the best solution.

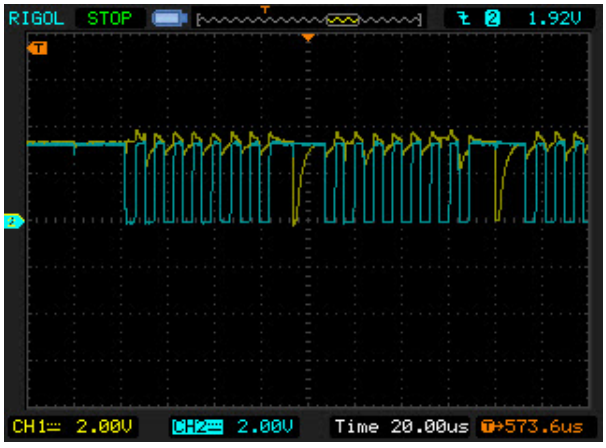

Figure1: SCLK in blue and Acknowledge from PS2 controller in yellow

The STM32F4 SPI parameters were configured as follows

- The PS2 controller operates at a maximum 500kHz. A clock rate of 125kHz worked well.
- CPOL bit was set to SPI_CPOL_High i.e. the clock will be logical state 1 when idle

Labels

Algorithms
Discovery Board
Hexapod
Python

Blog Archive

▼ 2013 (5)
► November (1)
▼ March (2)
PlayStation 2 Controller Interface
Python Excel Tips and Tricks
► February (1)
► January (1)
► 2012 (8)

About Me

Anonymous

View my complete profile

- CPHA bit was set to SPI_CPHA_2Edge i.e. the second clock transition is the first data capture edge
- 8 bit data size was used in full duplex mode with as the STM32F4 the master

The main.c file is as shown below

```
#include "stm32f4xx.h"
#include "Peripheral_Init.h"

unsigned char PS2_InitialPoll[5] = {0x01, 0x42, 0x00, 0x00, 0x00};

void PS2_send(unsigned char commands[], unsigned char data[], int leng
void delay_us(int delay_time);

int main()
{
 PeriphClk_config();
 GPIO_config();
 Timer3_config();
 SPI_config();
 unsigned char Data[5] = {0x00, 0x00, 0x00, 0x00, 0x00};

 while(1)
 {
  PS2_send(PS2_InitialPoll, Data, 5);
  GPIOD->ODR = GPIOD->ODR & ~(GPIO_Pin_15 | GPIO_Pin_14 | GPIO_Pin_13
//  Data[3] = ~Data[3];
//  if((Data[3] == 0b1)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_12; //sele
//  if((Data[3] == 0b10)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_13;
//  if((Data[3] == 0b100)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_13 | GPI
//  if((Data[3] == 0b1000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14; //s
//  if((Data[3] == 0b10000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14 | G
//  if((Data[3] == 0b100000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14 |
//  if((Data[3] == 0b1000000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14 |
//  if((Data[3] == 0b10000000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_15;
  Data[4] = ~Data[4];
  if((Data[4] == 0b1)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_12; //L2
  if((Data[4] == 0b10)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_13; //R2
  if((Data[4] == 0b100)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_13 | GPIO_
  if((Data[4] == 0b1000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14; //R1
  if((Data[4] == 0b10000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14 | GPI
  if((Data[4] == 0b100000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14 | GP
  if((Data[4] == 0b1000000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_14 | G
  if((Data[4] == 0b10000000)) GPIOD->ODR = GPIOD->ODR | GPIO_Pin_15; /

  delay_us(300); //0.3ms delay
 }
 return 0;
}

void PS2_send(unsigned char commands[], unsigned char data[], int leng
{
 int i = 0;
 GPIO_ResetBits(GPIOA, GPIO_Pin_3); //set CS pin low to select PS2
 delay_us(15); //15us delay before sending commands

 for(i = 0; i < length; i = i + 1)
 {
  while(SPI_I2S_GetFlagStatus(SPI1, SPI_FLAG_TXE) == RESET){}
  SPI_I2S_SendData(SPI1, commands[i]);
  while (SPI_I2S_GetFlagStatus(SPI1, SPI_FLAG_RXNE) == RESET){}
  data[i] = SPI_I2S_ReceiveData(SPI1);
  delay_us(15); //15us delay before sending next command
 }
 GPIO_SetBits(GPIOA, GPIO_Pin_3); //set CS pin high to deselect PS2
}

void delay_us(int delay_time)
{
 //can delay to a max of 60000us or 60ms
 int time = TIM3->CNT; //get initial time
 while ((TIM3->CNT - time) < delay_time);
}
```

Peripheral_Init.c is as shown below

```
//PA5 - SPI1_SCLK
//PA4 - SPI1_NSS
//PA6 - SPI1_MISO
//PA7 - SPI1_MOSI
//PA3 - CS (GPIO)

//PD12 - LED4 (green)
//PD13 - LED3 (orange)
//PD14 - LED5 (red)
//PD15 - LED6 (blue)

```

```
#include "Peripheral_Init.h"

void PeriphClk_config()
{
 //AHB1 clock runs at SystemCoreClock/AHB1_Prescaler = 64/1 = 64Mhz
 RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
 RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

 //APB1 clock runs at SystemCoreClock/APB1_Prescaler = 64/2 = 32MHz
 //Max speed of APB1 clock is 42Mhz
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

 //APB2 clock runs at SystemCoreClock/8 = 8MHz
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);
}

void Timer3_config()
{
 //Tim3Clk = 2* PCLK1
 //PCLK1 = (HCLK)/(APB1 Prescaler) = SystemCoreClock/2
 //Get the APB1 Prescaler from system_stm32f4xx.c
 //Tim3Clk = SystemCoreCLock

 //CONFIGURE TIMER3 CHANNEL 3 as timer only
 TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
 uint16_t PrescalerValue = (uint16_t) ((SystemCoreClock) / 1000000) -

 TIM_TimeBaseStructure.TIM_Period = 60000; //final frequency = clock f
            //count from 0-60000 at 1Mhz
 TIM_TimeBaseStructure.TIM_Prescaler = PrescalerValue;
 TIM_TimeBaseStructure.TIM_ClockDivision = 0;
 TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
 TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

 TIM_Cmd(TIM3, ENABLE); //enable timer TIM3
}

void GPIO_config()
{
 GPIO_InitTypeDef GPIO_InitStructure;

 //configure PD12-15 as outputs
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
 GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14
 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;
 GPIO_Init(GPIOD, &GPIO_InitStructure);

 //configure pins associated with SPI
 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
 GPIO_Init(GPIOA, &GPIO_InitStructure); //configure CS pin
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
 GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;
 GPIO_Init(GPIOA, &GPIO_InitStructure);
 GPIO_PinAFConfig(GPIOA, GPIO_PinSource5, GPIO_AF_SPI1); //sclk
 GPIO_PinAFConfig(GPIOA, GPIO_PinSource6, GPIO_AF_SPI1); //MISO
 GPIO_PinAFConfig(GPIOA, GPIO_PinSource7, GPIO_AF_SPI1); //MOSI
 GPIO_SetBits(GPIOA, GPIO_Pin_3); //set CS high to deselect PS2
}

void SPI_config()
{
 SPI_InitTypeDef  SPI_InitStructure;
 SPI_I2S_DeInit(SPI1);

 //configure the SPI
 SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_64; /
 SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge; //sample on second edge
 SPI_InitStructure.SPI_CPOL = SPI_CPOL_High; //clock high when idle
 SPI_InitStructure.SPI_CRCPolynomial = 0;
 SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b; //send 8 bits at a
 SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex; //
 SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_LSB; //least significar
 SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
 SPI_InitStructure.SPI_Mode = SPI_Mode_Master;

 SPI_Init(SPI1, &SPI_InitStructure);
 SPI_Cmd(SPI1, ENABLE);
}
```

and Peripheral_Init.h is as follows

```
#ifndef PERIPHERAL_INIT_H
#define PERIPHERAL_INIT_H

```

```
#include "stm32f4xx.h"

//FUNCTION PROTOTYPES
void PeriphClk_config(); //function to initiate peripheral clocks
void GPIO_config(); //function to initiate required GPIO
void Timer3_config(); //configure timer 3
void SPI_config(); //configure SPI1

#endif
```

The code above was developed for an STM32F4 Discovery board, changes will need to be made as per individual circumstances. Furthermore the program only polls the PS2 controller in its default digital mode. Additional commands would need to be sent to configure the controller to obtain joystick analog values etc. The links provided at the start of this discussion are an excellent resource to learn how to fully exploit the PS2 controller. Finally whilst this guide was aimed at specifically the STM32F4, the general principles regarding SPI setup should by universally transferable to other microcontrollers.

Posted by Anonymous at 18:55

Labels: Discovery Board, Hexapod, SPI, STM32F4

5 comments:

vbo 26 August 2013 at 06:44

I have a question, i have ps1 controller, can i make it interface with stm32 with the same routine, 0x01, 0x42,0,0,0?

Reply

Replies

Anonymous 5 October 2013 at 23:47

In short I have no idea. However this site http://www.instructables.com/id/Extracting-the-PS1-Controller-Joysticks/ would suggest that PS1 uses UART and not SPI, so I would not expect the code provided here to work with a PS1 controller.

Reply

Joshua Green 8 December 2014 at 02:40

Hi, I'm attempting to interface a ps2 controller using your code, and a simple but frustrating error message is occuring at the start of the polling lines for Data[4]. Lines 31 - 38 in your main.c file. Its asking for a ')' symbol. Any idea why? I'm using uVision btw.

Reply

Replies

Anonymous 4 September 2016 at 08:17

Hi, I have too problem, how can I fix that thing?

Reply

ARDIANSYAH PERSIKAL 9 February 2017 at 23:20

hi, can i use wireless ps2 for this tuorial?

Reply

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)