æProjekt:

Spi vs i2c

Her kommer jeg kun ind på de kritererier der relevante for os.

SPI pros:  
Full duplex  
Mindre strøm  
25Mbps  
SPI cons:  
rækkevide 20cm

I2C pros:  
rækkevide 1m  
I2C cons:  
Half duplex.  
1Mbps

Example 1:

*SPI kode:*

Raspberry:  
#include <stdio.h>

#include <wiringPi.h>

int main(void)

{

int pinNumber = 0; // Change this to the appropriate GPIO pin number

if (wiringPiSetup() == -1) {

printf("Error: Could not initialize wiringPi\n");

return 1;

}

pinMode(pinNumber, OUTPUT);

digitalWrite(pinNumber, HIGH); // Send a high signal

return 0;

}

*PSOC:*

#include "project.h"

#define BUFFER\_SIZE 4

int main(void) {

uint8\_t txBuffer[BUFFER\_SIZE] = {0};

uint8\_t rxBuffer[BUFFER\_SIZE] = {0};

CyGlobalIntEnable; /\* Enable global interrupts. \*/

/\* Start SPI communication \*/

SPIM\_Start();

for(;;) {

/\* Wait for a message to be received \*/

while (SPIM\_GetRxBufferSize() < BUFFER\_SIZE) {}

/\* Read the received data into the buffer \*/

SPIM\_ReadRxData(rxBuffer, BUFFER\_SIZE);

/\* Echo the message back to the Raspberry Pi \*/

SPIM\_WriteTxData(rxBuffer, BUFFER\_SIZE);

}

}

Example 2:

*Raspberry code:*

#include <iostream>

#include <fcntl.h>

#include <unistd.h>

#include <linux/i2c-dev.h>

using namespace std;

int main()

{

// Open the I2C bus

int file;

char \*filename = (char\*)"/dev/i2c-1";

if ((file = open(filename, O\_RDWR)) < 0) {

perror("Failed to open the I2C bus");

return 1;

}

// Set the address of the slave device

int addr = 0x42; // The address of the Cypress PSoC on the I2C bus

if (ioctl(file, I2C\_SLAVE, addr) < 0) {

perror("Failed to set the I2C address");

return 1;

}

// Send data to the slave device

char data[] = "Hello, Cypress PSoC!";

int len = sizeof(data);

if (write(file, data, len) != len) {

perror("Failed to write to the I2C bus");

return 1;

}

// Read data from the slave device

char buf[256];

if (read(file, buf, 256) != len) {

perror("Failed to read from the I2C bus");

return 1;

}

// Display the received data

cout << "Received data: " << buf << endl;

// Close the I2C bus

close(file);

return 0;

}

*Psoc code:*

#include "project.h"

#include <stdio.h>

#define BUFFER\_SIZE 256

uint8\_t buffer[BUFFER\_SIZE];

uint16\_t bufferIndex = 0;

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

/\* Start the I2C slave component \*/

I2CS\_Start();

for (;;)

{

/\* Check if there is data available to read from the I2C bus \*/

if (I2CS\_SlaveStatus() & I2CS\_SSTAT\_RD\_CMPLT)

{

/\* Read the data from the I2C bus \*/

uint8\_t len = I2CS\_SlaveGetReadBufSize();

I2CS\_SlaveReadBuf(buffer, len);

/\* Process the data \*/

printf("Received data: %s\n", buffer);

/\* Respond to the data \*/

char response[] = "Hello, Raspberry Pi!";

I2CS\_SlaveClearReadBuf();

I2CS\_SlaveWriteBuf(response, sizeof(response));

}

/\* Check if there is space available to write to the I2C bus \*/

if (I2CS\_SlaveStatus() & I2CS\_SSTAT\_WR\_CMPLT)

{

/\* Write data to the I2C bus \*/

uint8\_t len = I2CS\_SlaveGetWriteBufSize();

I2CS\_SlaveClearWriteBuf();

printf("Writing data: %s\n", buffer);

}

/\* Delay to prevent overloading the processor \*/

CyDelay(1);

}

}