Modul Praktikum Jaringan Komunikasi 06

Protokol Pengiriman Data Synchronous berbasis Mikrokontroler

Tujuan:

1. Mengenal dan memahami protokol komunikasi data synchronous berbasis mikrokontroler
2. Mengenal dan memahami prinsip kerja protokol komunikasi I2C dan SPI pada mikrokontroler

Kompetensi Dasar

1. Rangkaian Listrik
2. Dasar Informatika
3. Elektronika Analog
4. Pengolahan Data
5. Elektronika Digital
6. Jaringan Komunikasi

Latar Belakang

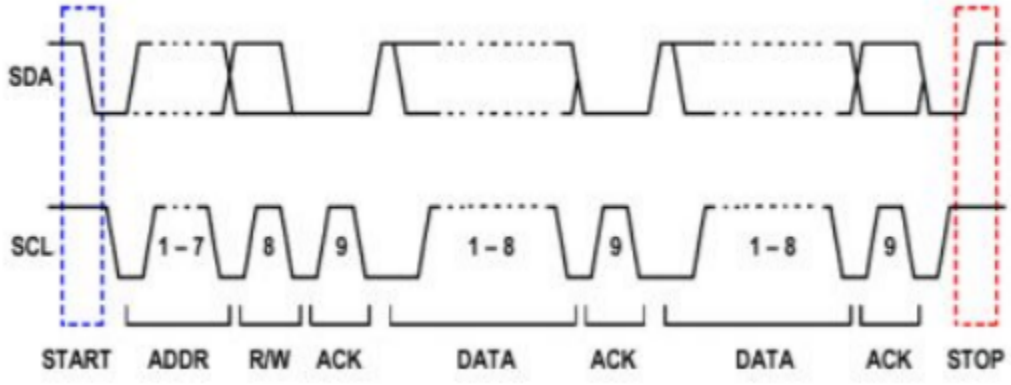
Port serial yang umum, jenis yang memiliki jalur TX dan RX, disebut "asinkron" (tidak sinkron) karena tidak ada kontrol kapan data dikirim atau jaminan bahwa kedua sisi berjalan pada kecepatan yang sama. Karena komputer biasanya mengandalkan segala sesuatu yang disinkronkan ke satu "jam" (kristal utama yang terpasang pada komputer yang menggerakkan segalanya), ini bisa menjadi masalah ketika dua sistem dengan jam yang sedikit berbeda mencoba berkomunikasi satu sama lain.

Untuk mengatasi masalah ini, koneksi serial asinkron menambahkan bit start dan stop tambahan pada setiap byte untuk membantu receiver melakukan sinkronisasi terhadap data yang datang. Kedua belah pihak juga harus menyetujui kecepatan transmisi (seperti 9600 bit per detik) sebelumnya. Perbedaan kecil dalam kecepatan transmisi tidak menjadi masalah karena receiver melakukan sinkronisasi ulang pada awal setiap byte.

Protokol serial sering kali mengirimkan bit yang paling tidak signifikan terlebih dahulu (least significant bits first), sehingga bit terkecil berada di paling kiri.

Inter Integrated Circuit

Inter Integrated Circuit biasa disebut sebagai I2C, Bus I2C atau IIC Bus. pada awalnya dikembangkan sebagai bus kontrol untuk menghubungkan mikrokontroler dan peripheral IC [1]. Inter-Integrated Circuit atau sering disebut sebagai komunikasi I2C adalah komunikasi yang dikembangkan oleh Philips SemiConductors, yang hanya menggunakan dua jalur komunikasi (2-wire) yaitu Synchronous Data (SDA) yang digunakan untuk mengirim dan menerima data (bi-directional) dan Synchronous Clock (SCL) yang digunakan untuk mengirim sinyal sinkronisasi. Protokol I2C untuk pengiriman satu byte data dapat dilihat pada Gambar 3.1.

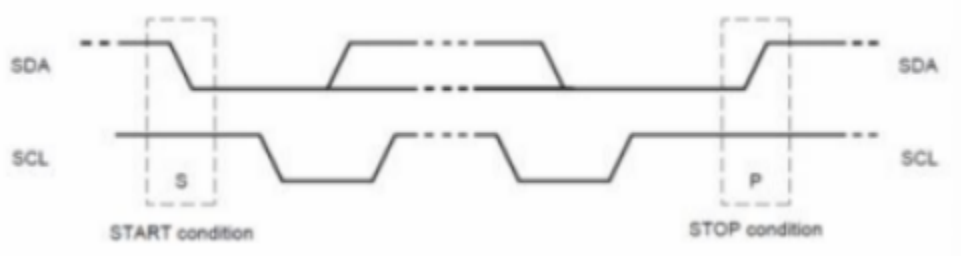


Gambar 3.1. Dataframe Protokol Komunikasi I2C

Perangkat I2C menggunakan 2 buah pin open-drain dua arah dengan memberikan pull-up resistor untuk setiap garis bus sehingga berlaku seperti AND menggunakan kabel. Piranti yang dihubungkan dengan sistem I2C Bus dapat dioperasikan sebagai Master dan Slave. Master adalah piranti yang memulai transfer data pada I2C Bus dengan membentuk sinyal Start, mengakhiri transfer data dengan membentuk sinyal Stop, dan membangkitkan sinyal clock. Slave adalah piranti yang dialamati Master.

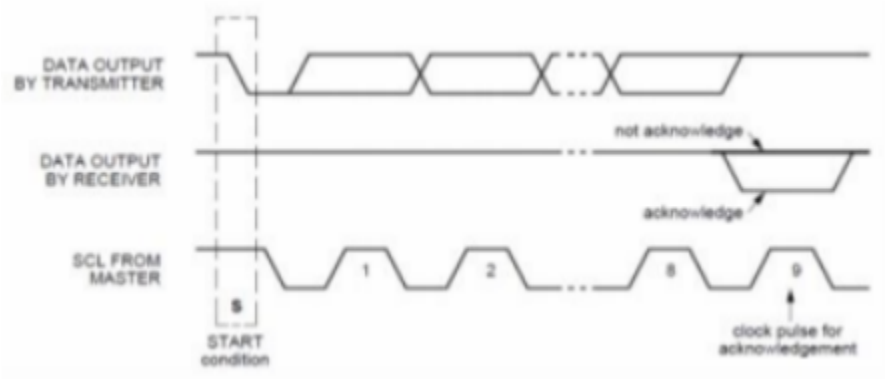
Sinyal Start merupakan sinyal untuk memulai semua perintah, didefinisikan sebagai perubahan tegangan SDA dari “1” menjadi “0” pada saat SCL “1”. Sinyal Stop merupakan sinyal untuk mengakhiri semua perintah, didefinisikan sebagai perubahan tegangan SDA dari “0”

menjadi “1” pada saat SCL “1”. Kondisi sinyal Start dan sinyal Stop seperti tampak pada Gambar 3.2 [3].



Gambar 3.2. Kondisi **s**inyal *start* dan *stop*

Sinyal dasar yang lain dalam I2C Bus adalah sinyal acknowledge yang disimbolkan dengan ACK Setelah transfer data oleh master berhasil diterima slave, slave akan menjawabnya dengan mengirim sinyal acknowledge, yaitu dengan membuat SDA menjadi “0” selama siklus clock ke 9. Ini menunjukkan bahwa Slave telah menerima 8 bit data dari Master. Kondisi sinyal acknowledge seperti tampak pada Gambar 3.3 [3].



Gambar 3.3. Sinyal ACK dan NACK pada I2C

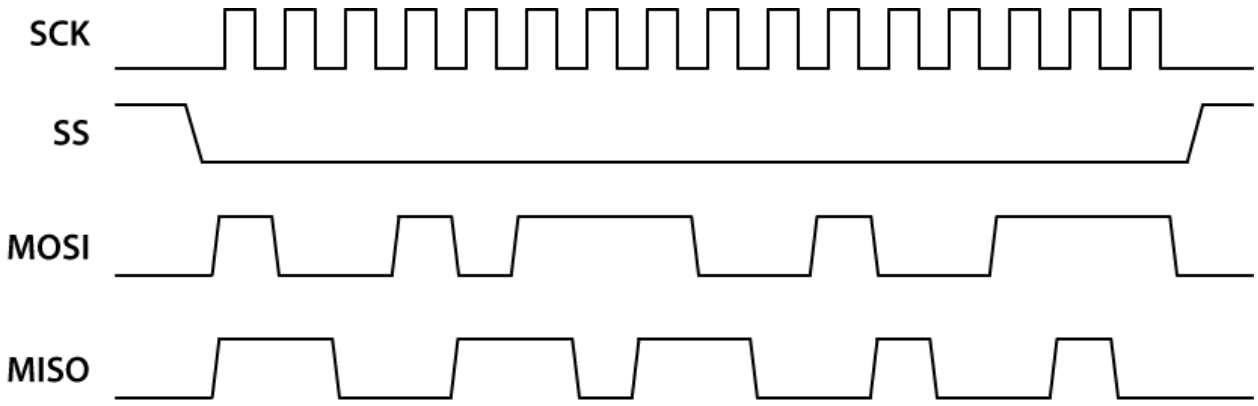
SPI (Serial Peripheral Interface)

Serial Peripheral Interface (SPI), protokol yang dibuat oleh Motorola pada tahun 1980-an, utamanya digunakan untuk komunikasi serial tersinkronisasi (synchronized serial communication) antara prosesor (master) dan IC yang dituju (slave). Empat buah jalur sinyal digunakan dalam komunikasi ini : Chip Select (CS), Serial Data Input (SDI), Serial Data Output (SDO), Serial Clock (SCKL). CS dan SCKL merupakan keluaran dari piranti master. Piranti

slave menerima sinyal clock dan masukan chip select dari piranti master. Apabila sebuah piranti SPI tidak dipilih, jalur SDO dari piranti tersebut memiliki status high impedance state. Jumlah bit yang dikirim ke piranti slave bervariasi antar piranti. Setiap piranti slave memiliki sebuah internal shift register yang digunakan untuk pengiriman data.

Ada dua tipe koneksi antar piranti master dan slave. Pada koneksi tipe pertama, seluruh slave berbagi satu jalur CS. Jalur SDO dari piranti master dihubungkan ke jalur SDI dari piranti slave pertama. Jalur SDO dari piranti slave pertama dihubungkan ke jalur SDI piranti slave kedua. Begitu seterusnya hingga hubungan terakhir ke jalur SDI piranti master. Pada koneksi tipe kedua, setiap jalur SDI piranti slave terhubung pada jalur SDO piranti master. Setiap piranti slave memiliki jalur CS masing-masing. Setiap jalur SDO piranti slave terhubung ke jalur SDI piranti master.

Jenis komunikasi ini juga mendukung mode multiple master. Laju serial clock mempunyai jangkauan dari 30 kHz hingga 3 MHz, bergantung pada piranti yang digunakan. Beberapa literatur menyebutkan CS sebagai SS, SDI sebagai MOSI, dan SDO sebagai MISO.

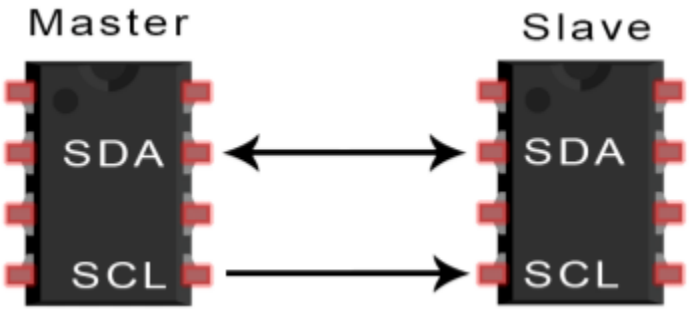


Gambar 3.4 . Contoh dataframe protokol komunikasi SPI

Langkah Percobaan

I2C

1. Buat rangkaian seperti pada blok diagram di bawah ini dengan menghubungkan SDA dan SCL pada *device master* dan *device slave*.



Gambar 4.1. Blok diagram protokol komunikasi I2C

Percobaan 1: I2C pada Arduino

1. Buka Arduino IDE
2. Buat program untuk pengiriman data via protokol komunikasi I2C seperti

```
#include <Wire.h>
char x[] = "A";
void setup() {
  wire.Begin();
  //Serial.begin(9600);
}
void loop() {
  wire.BeginTransmission(25);
  wire.setclock(100000);
  //Wire.setClock(400000);
  wire.Write(x);
  wire.EndTransmission();
  //Serial.println("I2C Master, UART Rx");
  //Serial.print("Data: ");
  //Serial.print((char)Serial.read());
  delay(500);
}
```

Gambar 4.2. Contoh program I2C *master* pada Arduino

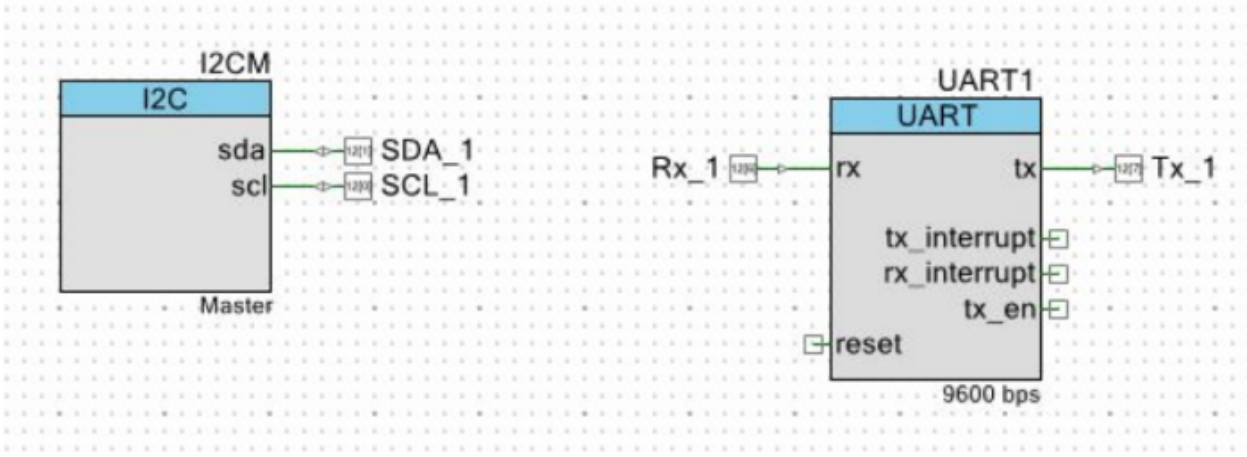
```
#include <Wire.h>
char x;
void setup() {
  wire.Begin(8);
  serial.Begin(19200);
  wire.OnReceive(receiveEvent);
}
void receiveEvent(int bytes) {
  x = wire.Read();
}
void loop() {
  serial.Print("Data: ");
  serial.Println(x);
  delay(200);
}
```

Gambar 4.3. Contoh program I2C *slave* pada Arduino

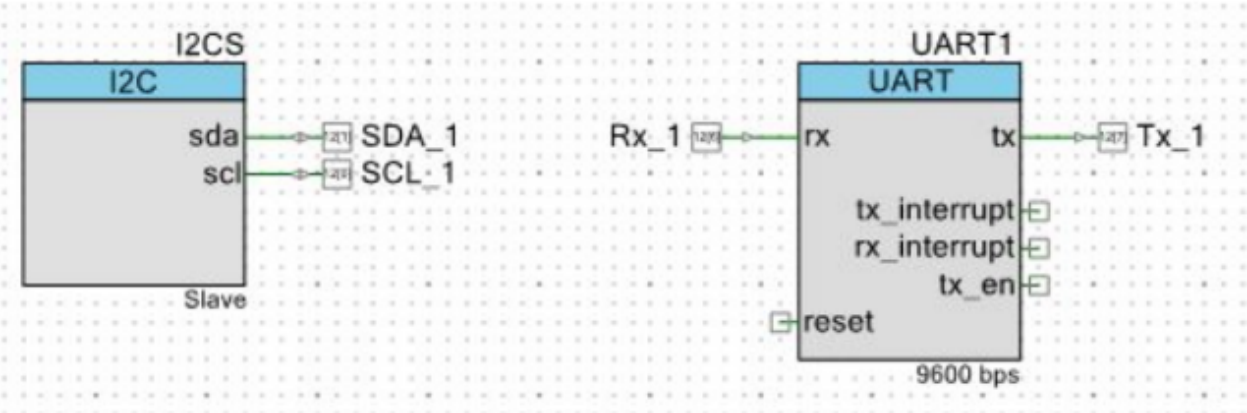
3. Siapkan 2 buah arduino. Arduino ke-1 sebagai I2C *master* dan Arduino ke-2 sebagai I2C *slave*.
4. *Verify* dan *build* program yang telah dibuat pada masing-masing Arduino 1 dan 2
5. Hubungkan pin Tx Arduino *master* pada osiloskop
6. Variasikan karakter, clock, dan address I2C sesuai tabel percobaan
7. Amati dataframe hasil variasi pada osiloskop dan serial monitor Arduino *slave*

Percobaan 2: I2C pada PSoC

7. Buka perangkat lunak PSoC Creator
8. Klik tab File > New > Project
9. Susun komponen blok digital pada PSoC Project sesuai Gambar 4.4 untuk *master* dan Gambar 4.5 untuk *slave*.

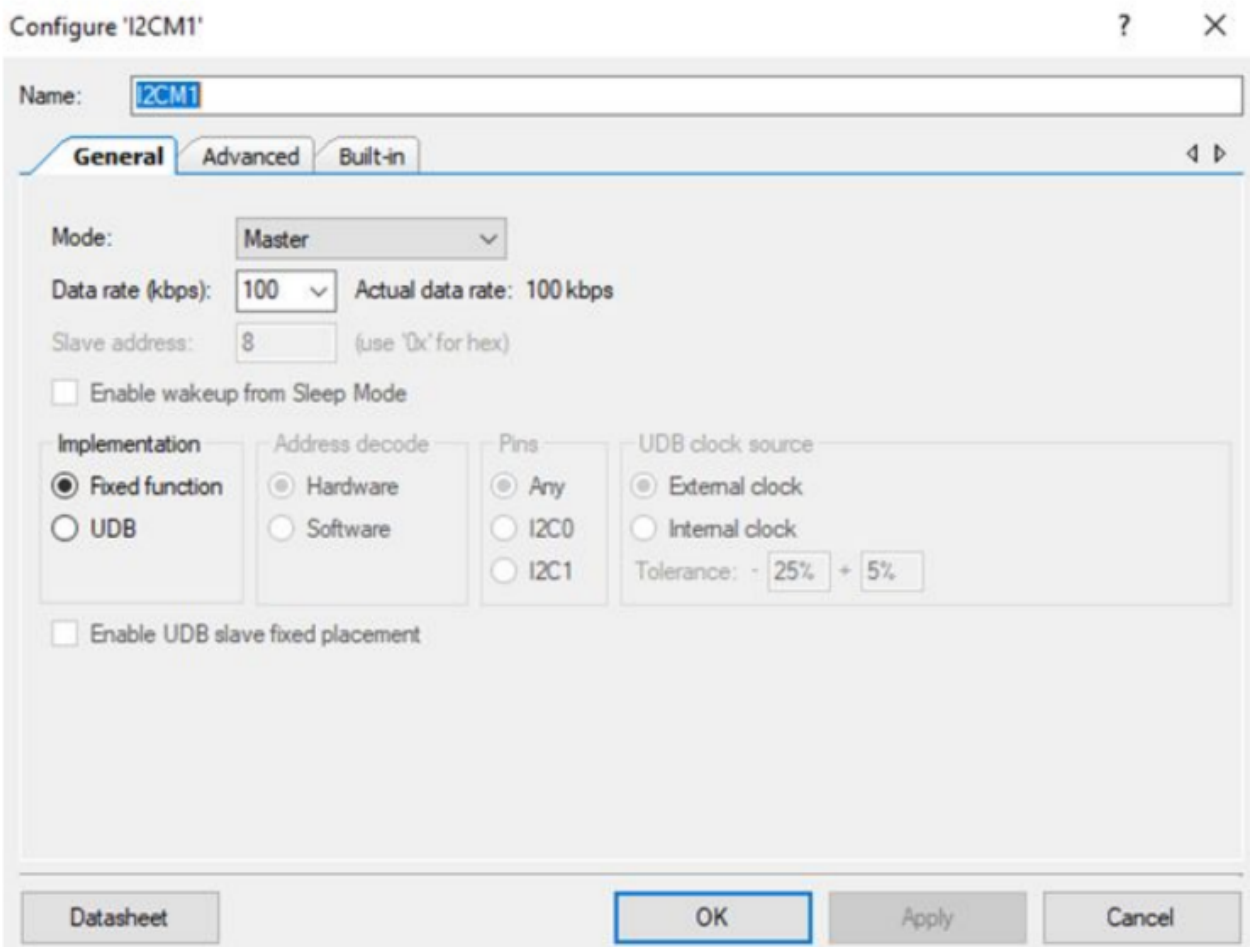


Gambar 4.4. Komponen blok yang digunakan pada PSoC *Master*



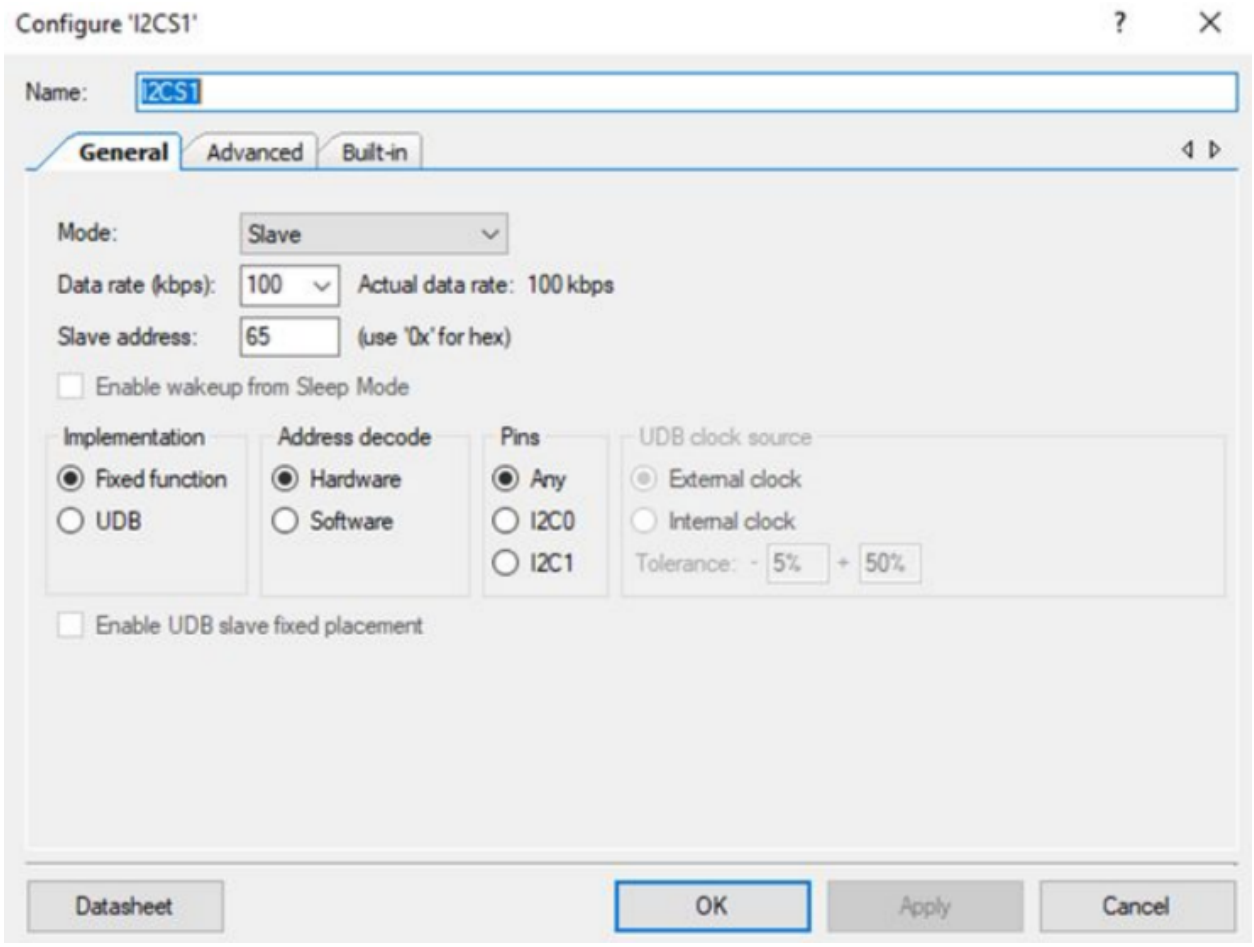
Gambar 4.5. Komponen blok yang digunakan pada PSoC *Slave*

10. Atur blok I2C master seperti pada Gambar 4.6.



Gambar 4.6. Konfigurasi komponen I2C *master*

11. Atur komponen I2C *slave* seperti pada Gambar 4.7



Gambar 4.7. Konfigurasi komponen I2C *slave*

12. Buat kode program untuk I2C *master* seperti pada Gambar 4.8.

```
#include <project.h>
#include <stdio.h>
#include <stdlib.h>
#define I2C_SLAVE_ADDRESS (8u)
#define WR_BUFFER_SIZE (255)
char send[3];
int main()
{
    uint8 data='a';
    uint16 temp;
    I2CM_Start();
    UART1_Start();
    CyGlobalIntEnable;
    for(;;){
        sprintf(send, "%d", data);
        do{
            temp = I2CM_MasterWriteBuf(I2C_SLAVE_ADDRESS, (uint16 *)send,
            WR_BUFFER_SIZE, I2CM_MODE_COMPLETE_XFER);}
        while (temp != I2CM_MSTR_NO_ERROR);
        while(I2CM_MasterStatus() & I2CM_MSTAT_XFER_INP);
        temp = I2CM_MasterClearStatus();
        if(temp & I2CM_MSTAT_ERR_XFER){
            UART1_PutString("Master I2C ERROR");
            UART1_PutString("\n");
        }
        else{
            UART1_PutString(send);
            UART1_PutString("\n");
        }
        CyDelay(500);}
}
```

Gambar 4.8. Contoh kode program I2C *master* pada PSoC

13. Buat kode program I2C *slave* seperti pada Gambar 4.9.

```
#include "project.h"
#include <stdio.h>
#include <stdlib.h>
#define WR_BUFFER_SIZE (255)
int main()
{
    int i;
    uint8 rec[4];
    rec [0] = 0;
    uint8 wrBuf[WR_BUFFER_SIZE];
    uint8 byteCount = 0u;
    I2CS_Start();
    UART1_Start();
    I2CS_SlaveInitWriteBuf((uint8 *)wrBuf, WR_BUFFER_SIZE);
    CyGlobalIntEnable;
    for(;;)
    {

        if(I2CS_SlaveStatus() & I2CS_SSTAT_WR_CMPLT)
        {
            byteCount = I2CS_SlaveGetWriteBufSize();
            I2CS_SlaveClearWriteStatus();
            if(byteCount == WR_BUFFER_SIZE)
            {
                for(i=0; i < byteCount; i++)
                {
                    rec[i] = wrBuf[i];
                }
                UART1_PutString(rec);
                UART1_PutString("\n");
                I2CS_SlaveClearWriteBuf();
            }
            else
            {
                UART1_PutString("Slave I2C Error");
                UART1_PutString("\n");
            }
        }
    }
}
```

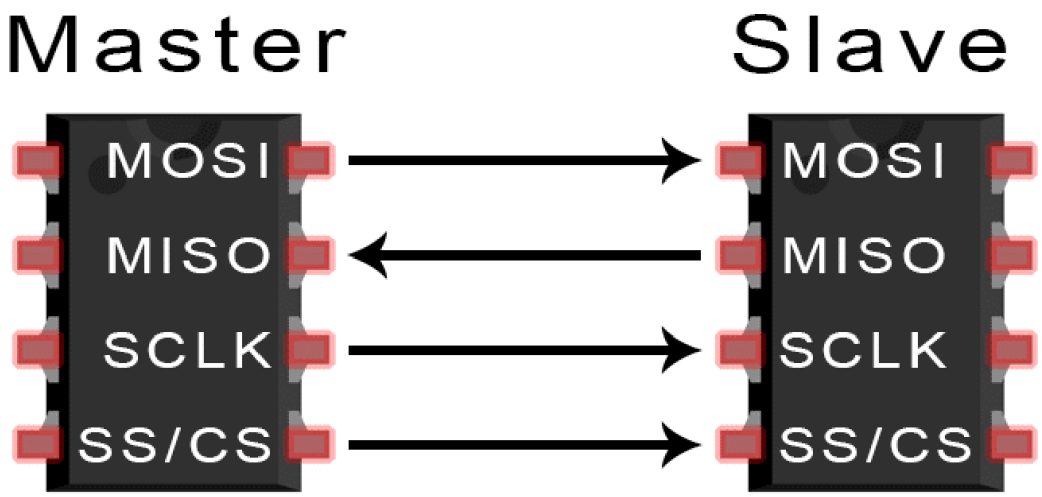
Gambar 4.9. Contoh kode program I2C *slave* pada PSoC

14. Clean and build program masing-masing project ke masing-masing PSoC sesuai perannya masing-masing
15. Hubungkan Tx pada PSoC *master* dengan probe osiloskop
16. Variasikan karakter, address, dan buffer size pada bagian main.c *master* sesuai tabel percobaan

17. Variasikan frekuensi clock pada PSoC master dan slave sesuai dengan tabel percobaan.
18. Buka Arduino IDE
19. Pilih port yang tersambung dengan PSoC *Slave* lalu buka Serial Monitor
20. Amati dan catat hasil yang diperoleh pada serial monitor

SPI (Serial Peripheral Interface)

- Buat rangkaian pada Arduino dan PSoC seperti pada .. dengan menghubungkan SCLK, MISO, MOSI, dan SS pada Arduino dan PSoC *Master* dan *Slave*



Gambar 4.10. Blok diagram protokol komunikasi SPI

Percobaan 3: SPI pada Arduino

1. Buka Arduino IDE
2. Buat kode program Arduino *master* seperti pada Gambar 4.11.

```
#include <spi.h>

void setup(void) {
  // put your setup code here, to run once:
  serial.Begin(38400);
  digitalWrite(SS,HIGH);
  spi.Begin();
  spi.SetClockDivider(SPI_CLOCK_DIV8);

}

void loop(void) {
  // put your main code here, to run repeatedly:
  char c;
  digitalwrite(SS,LOW);
  for (const char * p = "JarKom Hore\r"; c = *p; p++)
  {
    spi.Transfer(c);
    serial.Print(c);
  }
  digitalwrite(SS,HIGH);
  delay(200);
}
```

Gambar 4.11. Contoh kode program Arduino untuk SPI *master*

3. Buat kode program untuk Arduino sebagai SPI *slave*, seperti pada Gambar 4.12.

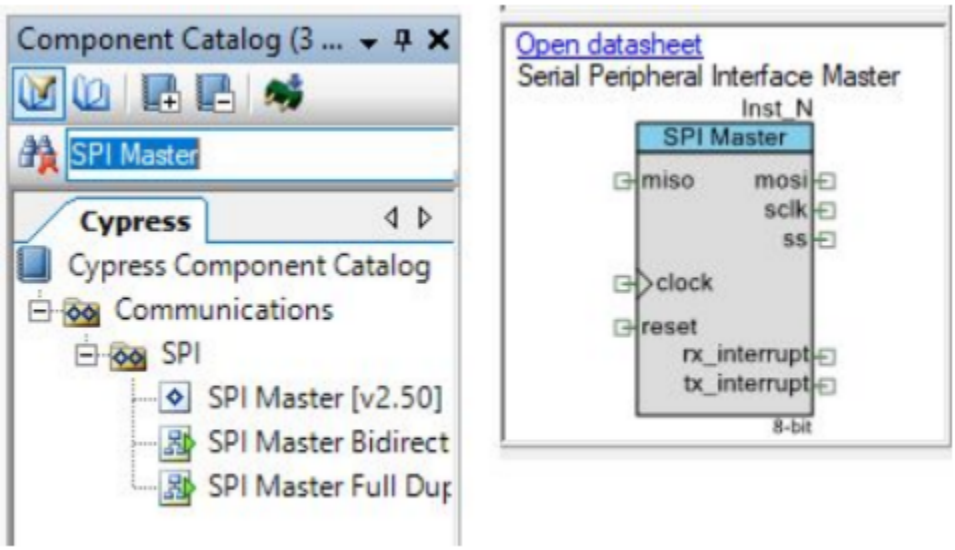
```
#include <spi.h>

char buff [50];
volatile byte indx;
volatile boolean process;


void setup(void) {
  // put your setup code here, to run once:
  serial.Begin(9600);
  PinMode(MISO,OUTPUT);
  SPCR |= _BV(SPE);
  indx = 0;
  process = false;
  spi.aAtachInterrupt();
}

ISR (SPI_STC_vect)
{
  byte c = SPDR;
  if (indx < sizeof buff){
    buff [indx++] = c;
    if (c == 'e')
    process = true;
  }
}

void loop(void) {
  // put your main code here, to run repeatedly:
  if (process){
    process = false;
    serial.Println(buff);
    indx = 0;
  }
}
```

Gambar 4.12. Contoh kode program Arduino SPI *slave*

4. Siapkan 2 buah Arduino. Arduino 1 sebagai SPI *master* dan Arduino 2 sebagai SPI *slave*.
5. Verify dan build kode program yang telah dibuat pada masing-masing Arduino
6. Variasikan karakter, clock, mode master, dan mode *slave* sesuai dengan tabel percobaan
7. Buka Arduino IDE dan sambungkan dengan Arduino *Slave*
8. Pilih port yang sesuai dan serial monitor pada Arduino IDE
9. Amati data yang diterima Arduino *slave* pada serial monitornya.

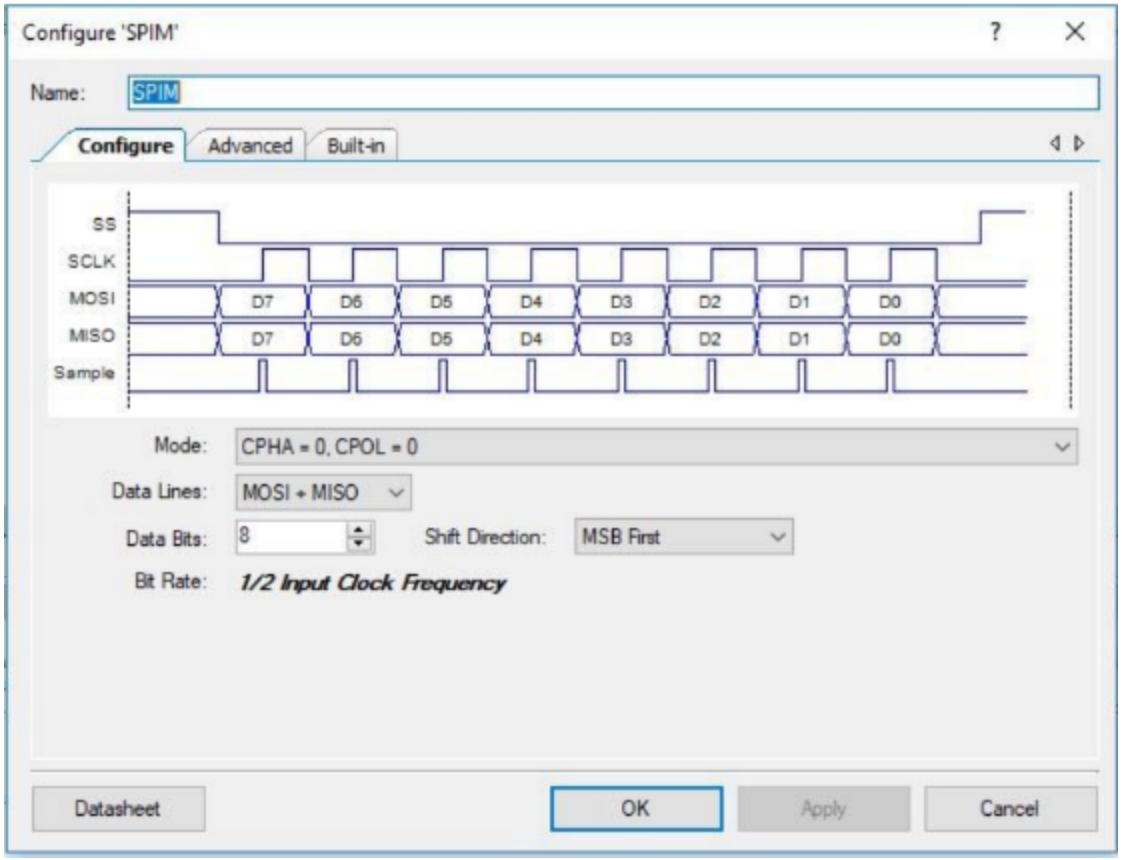
Percobaan 4: SPI pada PSoC

1. Buka PSoC Creator
2. Buat 2 (dua) project pada PSoC, dengan nama SPIMaster dan SPISlave
3. Di dalam project SPIMaster, pada bagian TopDesign.cysch masukkan blok komponen SPI master dari Component Catalog.



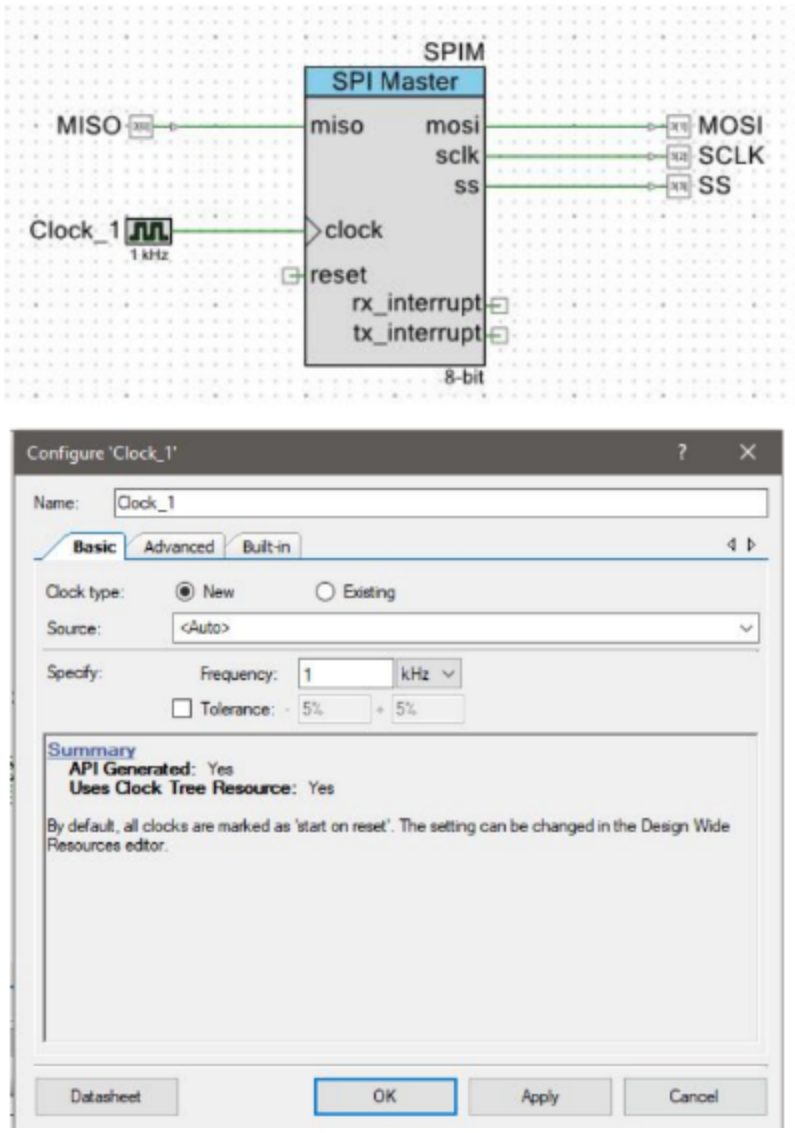
Gambar 4.13. Komponen SPI Master pada Component Catalog PSoC

4. Atur komponen SPI Master, gunakan pengaturan komponen sesuai Gambar 4.14. Pemberian nama komponen akan mempengaruhi nama pada program.



Gambar 4.14. Pengaturan komponen SPI *Master*

5. Tambahkan pin digital input pada MISO, digital output pada MOSI, SCLK, dan SS. Logic Low pada reset dan Clock sebesar 1kHz pada clock dan pada bagian Tolerance seperti pada Gambar 4.15.



Gambar 4.15. Konfigurasi komponen clock SPI *master* pada PSoC

6. Buat kode program SPI *master* seperti pada Gambar 4.16.

```
#include <project.h>
int main() {
    // Edit only this part
    uint8 message = 'a';
    uint32 delayTime = 100;
    // End of Edit only this part
    uint8 status;
    SPIM_Start();
    for (;;) {
        SPIM_ClearTxBuffer();
        status = SPIM_ReadTxStatus();
        if (status & (SPIM_STS_SPI_DONE | SPIM_STS_SPI_IDLE)) {
            SPIM_WriteTxData(message);
        }
        CyDelay(delayTime);
    }
}
```

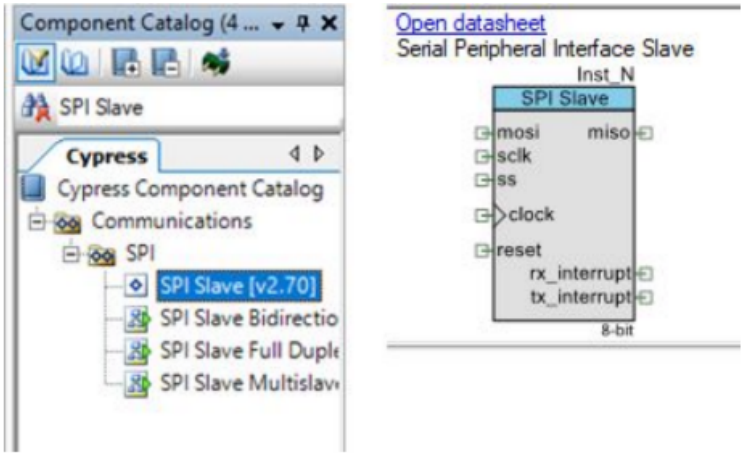
Gambar 4.16. Contoh kode program PSoC SPI *master*

7. Konfigurasi pin / ports untuk *input* dan *output* SPI *master* sesuai dengan Gambar 4.17.

	Name	Port	Pin	Lock
	MISO	P3[0]	29	☑
	MOSI	P3[1]	30	☑
	SCLK	P3[2]	31	☑
	SS	P3[3]	32	☑

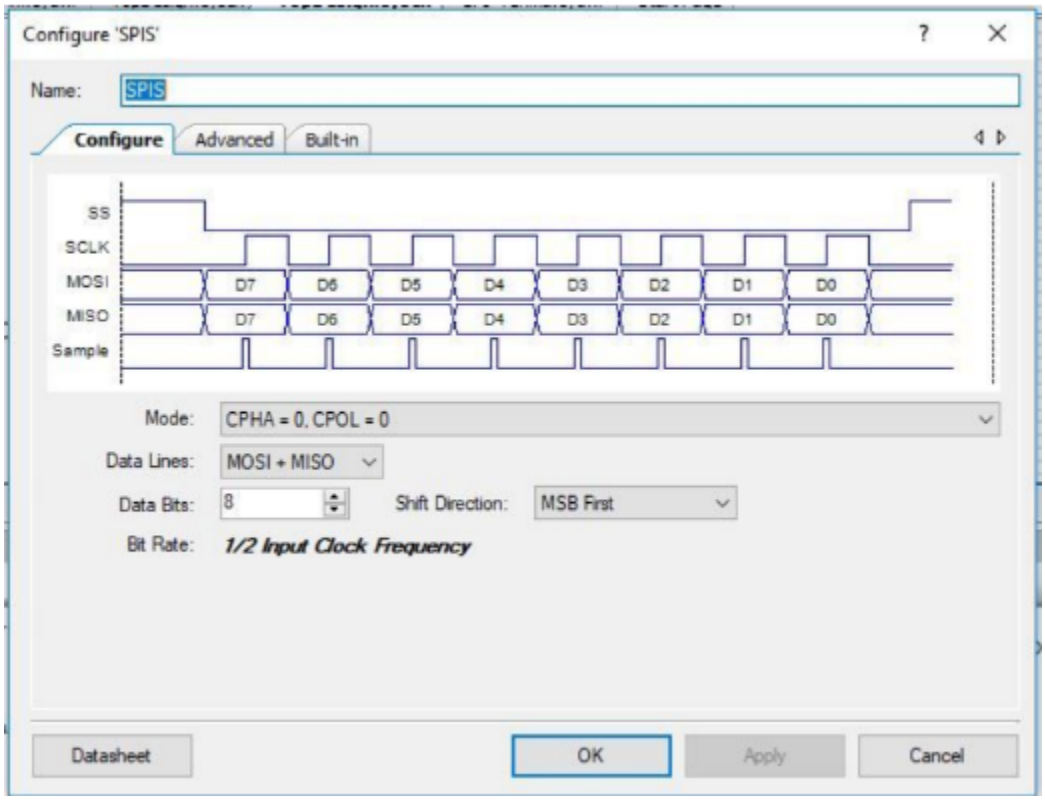
Gambar 4.17. Nomor pin *input* dan *output* pada SPI *master*

8. Clean dan build *program* lalu unggah program ke PSoC pertama yang berlaku sebagai SPI *master*.
9. Dalam project SPISlave, pada workspace TopDesign.cysch masukkan komponen SPI slave yang didapatkan dari Component Catalog seperti pada Gambar 4.18.



Gambar 4.18. Komponen SPI Slave pada Component Catalog PSoC

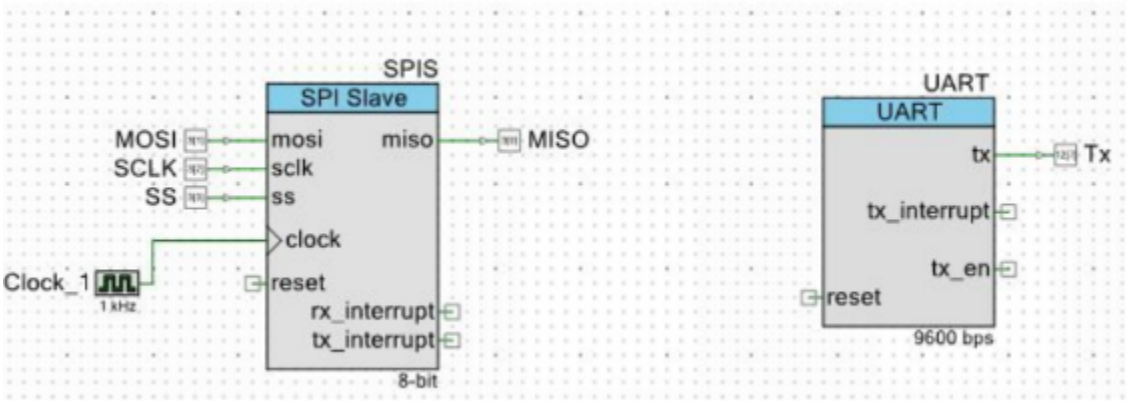
10. Atur SPI Slave pada komponen, gunakan pengaturan komponen seperti pada Gambar 4.19. Pemberian nama komponen akan mempengaruhi nama pada program.



Gambar 4.19. Pengaturan komponen SPI *Slave*

11. Tambahkan pin digital output pada MISO, digital output MOSI, SCLK, dan SS, lalu logic Low pada reset dan Clock sebesar 1 kHz. Pada bagian clock, bagian tolerance

di-nonaktifkan hingga sesuai dengan Gambar 4.20. Tambahkan komponen UART dan port keluaran Tx.



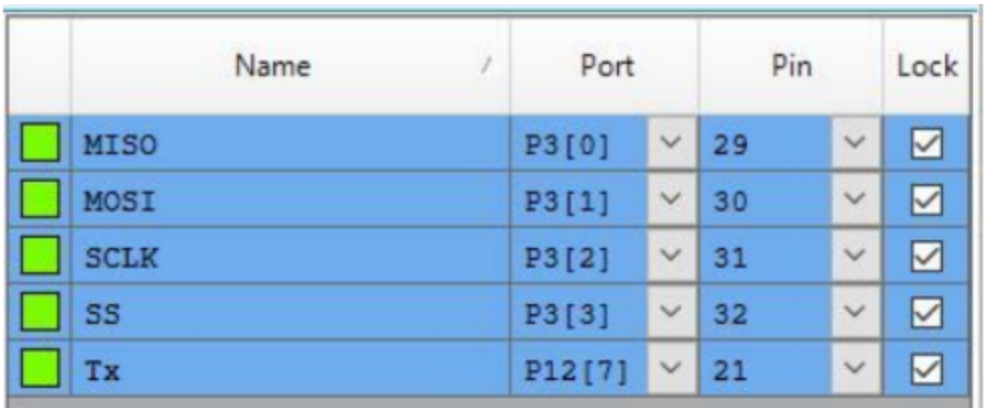
Gambar 4.20. Komponen SPI Slave dan UART

12. Buat kode program SPI *Slave* pada main.c untuk Project SPISlave seperti pada Gambar 4.21.

```
#include <project.h>
int main() {
    char message[2];
    message[1] = 0;
    SPIS_Start();
    UART_Start();
    for (;;) {
        if (SPIS_GetRxBufferSize()) {
            message[0] = SPIS_ReadRxData();
            // Pass to UART ? Tx: P12[7]
            UART_PutString(message);
            SPIS_ClearRxBuffer();
        }
    }
}
```

Gambar 4.21. Kode program SPI *Slave*

21. Konfigurasikan pin input-output SPI Slave seperti pada Gambar 4.22.



Gambar 4.22. Konfigurasi pin input-output SPI *Slave*

22. *Clean* dan *Build* program lalu unggah program ke PSoC kedua yang berperan sebagai SPI *Slave*.
23. Hubungkan dengan kabel port MOSI master ke MOSI Slave, MISO Master ke MISO Slave, dan berlaku seterusnya untuk SS, SCLK dan Ground
24. Buka Arduino IDE pada laptop yang terhubung dengan PSoC *Slave* dan pilih port yang sesuai.
25. Buka serial monitor
26. Catat dan amati data hasil percobaan melalui Serial Monitor

Percobaan 5: I2C antara board Arduino dan PSoC

1. Buka Arduino IDE
2. Buat kode program seperti pada Gambar 4.2.
3. Verify dan Build kode program pada Arduino Master
4. Buka PSoC Creator
5. Masukkan komponen blok I2C slave pada TopDesign.cysch seperti Gambar 4.5
6. Atur konfigurasi komponen I2C *slave* seperti pada Gambar 4.7
7. Buat kode program seperti pada Gambar 4.9 di bagian main.c PSoC Creator.
8. Atur ports PSoC P12[0] sebagai SCL, P12[1] sebagai SDA, dan P12[7] sebagai Tx.
9. Clean and build project PSoC kemudian program PSoC-nya
10. Buka Arduino IDE pada laptop yang tersambung dengan PSoC. lalu buka serial monitor
11. Amati dan catat data hasil percobaan melalui serial monitor.

Percobaan 6: I2C antara board PSoC dan PSoC

1. Buka PSoC Creator
2. Masukkan komponen blok I2C *master* pada TopDesign.cysch seperti pada Gambar 4.4
3. Atur konfigurasi komponen I2C *master* seperti pada Gambar 4.6.
4. Buat kode program seperti pada Gambar 4.8 pada bagian main.c PSoC Creator
5. Atur ports PSoC P12[0] sebagai SCL, P12[1] sebagai SDA, P12[6] sebagai Rx
6. Buka Arduino IDE
7. Buat kode program seperti pada Gambar 4.3.
8. Verify dan build kode program pada Arduino *Slave*, lalu buka serial monitor
9. Catat dan amati data hasil percobaan melalui serial monitor

Percobaan-07: SPI antar board Arduino dan PSoC

1. Buka Arduino IDE
2. Buat kode program seperti pada Gambar 4.11
3. *Verify* dan *upload* kode program pada Arduino SPI *master*
4. Buka PSoC Creator
5. Masukkan blok komponen SPI *slave* dan UART seperti pada Gambar 4.20.
6. Atur konfigurasi blok SPI seperti pada Gambar 4.19
7. Buat kode program seperti pada Gambar 4.21 di bagian main.c
8. Buka Arduino IDE pada laptop yang tersambung dengan PSoC SPI *slave*
9. Buka serial monitor
10. Catat dan amati data yang diterima pada serial monitor.

Percobaan-08: SPI antar board PSoC dan Arduino

1. Buka PSoC Creator
2. Masukkan komponen SPI *master* seperti pada Gambar 4.13.
3. Atur blok komponen SPI *master* seperti pada Gambar 4.14.
4. Buat kode program SPI *master* pada main.c seperti Gambar 4.16.
5. *Clean* and *Build* project, kemudian program pada PSoC SPI *master*
6. Buka Arduino IDE
7. Buat kode program Arduino SPI *slave* seperti Gambar 4.12.
8. Verify dan *upload* program pada Arduini SPI *slave*

9. Buka serial monitor
10. Catat dan amati hasil yang diterima Arduino *slave* pada Serial Monitor

Tabel Percobaan

Percobaan 1: I2C pada Arduino

Mikrokontroler	Data yang dikirim	Clock Master	Clock Slave	Address Master	Adddress Slave	Data di Serial Monitor
Arduino	A	100 kHz		8	8	
	C	1 MHz		21	19	
	D	100 kHz	400 kHz	65	65	
Kesimpulan						

Percobaan 2: I2C pada PSoC

Kata	Clock Master dan Slave	Clock pada Slave	Address pada Master	Address pada Slave	Buffer Master	Buffer Slave	Data di Serial Monitor
Jarkom Hore	400kHz		8		1	1	
Ada di	100kHz		13		10	2	
Lab. SSTK	1MHz		16		12	12	
Yaitu	127kHz	673kHz	33		6	3	
Badak	400kHz	127kHz	26		50	50	
Kesimpulan							

Percobaan 3: SPI pada Arduino

Pesan yang dikirim	Clock Divider Master	Data yang diterima
Tahun ini	2	
Jarkom	4	
Asyik	8	
Ada	16	
Responsinya	32	
Kesimpulan		

Percobaan 4: SPI pada PSoC

Karakter	Clock di Master	Clock di Slave	Mode Master	Mode Slave	Data di Serial Monitor
L	10kHz		CPHA = 0 CPOL = 0	CPHA = 0 CPOL = 0	
A			CPHA = 0 CPOL = 0	CPHA = 0 CPOL = 1	
B			CPHA = 0 CPOL = 0	CPHA = 1 CPOL = 0	
.			CPHA = 0 CPOL = 0	CPHA = 1 CPOL = 1	
S			CPHA = 0 CPOL = 1	CPHA = 0 CPOL = 0	
S			CPHA = 0 CPOL = 1	CPHA = 0 CPOL = 1	
T			CPHA = 0 CPOL = 1	CPHA = 1 CPOL = 0	
K			CPHA = 0 CPOL = 1	CPHA = 1 CPOL = 1	
A			CPHA = 1 CPOL = 0	CPHA = 0 CPOL = 0	
D			CPHA = 1 CPOL = 0	CPHA = 1 CPOL = 0	
A			CPHA = 1 CPOL = 0	CPHA = 1 CPOL = 1	
B			CPHA = 1 CPOL = 1	CPHA = 1 CPOL = 1	
A	1Mhz	3Mhz	CPHA = 0 CPOL = 0		
D	1Mhz	400kHz	CPHA = 0 CPOL = 0		
A	3Mhz	1Mhz	CPHA = 1 CPOL = 1		
K	10kHz	100khz	CPHA = 1 CPOL = 1		
Kesimpulan					

Percobaan 05: I2C antar board Arduino dan PSoC

Data yang dikirim	Address Master	Address Slave	Clock Master	Clock Slave	Data yang diterima

H					
Halo Dunia!					

Percobaan 06: I2C antar board PSoC dan Arduino

Data yang dikirim	Address Master	Address Slave	Clock Master	Clock Slave	Data yang diterima
H					
Hehe					

Percobaan 07: SPI antar board Arduino dan PSoC

Data yang dikirim	Address Master	Address Slave	Clock Master	Clock Slave	Data yang diterima
Y					
Yuk Bisa					

Percobaan 08: SPI antar board PSoC dan Arduino

Data yang dikirim	Address Master	Address Slave	Clock Master	Clock Slave	Data yang diterima
J					
Jarkom Asyik					