OBSERVASI KOMUNIKASI DATA MIKROKONTROLLER DENGAN METODE SPI BERBASIS MIKROKONTROLLER ATMEGA 8535

Hasil Penelitian / Pemikiran yang tidak dipublikasikan
Disusun sebagai salah satu syarat untuk
Kenaikan Angka Kredit Jabatan Fungsional Lektor

Oleh
Pipit Anggraeni
197908242005012001

JURUSAN TEKNIK OTOMASI MANUFAKTUR
DAN MEKATRONIKA
POLITEKNIK MANUFAKTUR NEGERI BANDUNG BANDUNG
2009

OBSERVASI KOMUNIKASI DATA MIKROKONTROLER DENGAN METODE SERIAL PERIPERAL INTERFACE BERBASIS MIKROKONTROLER AT MEGA 8535

Pipit Anggraeni
Jurusan Teknik Otomasi Manufaktur dan Mekatronika – POLMAN Bandung, Jl. Kanayakan 21 Bandung,
email : pipit_anggraeni@polman-bandung.ac.id

Abstrak

Komunikasi data pada dasarnya adalah sebuah sistem yang dibangun untuk menyampaikan pesan atau data dari pihak pengirim kepada pihak penerima pesan atau data. Pada system komunikasi data dituntut untuk memiliki kecepatan transmisi data yang cepat bahkan mendekati real time.

Dilihat dari jenisnya, komunikasi data secara serial terdiri atas I^2C (Inter-IC Bus), RS-485 dan SPI (Serial Pheripheral Interfac).

Komunikasi SPI dapat digunakan sebagai interface mikrokontroler AT Mega 8535 dengan display LCD atau menghubungkan lebih dari dua mikrokontroler dalam suatu jaringan. Waktu pengiriman 1 byte data dengan SCK sebesar f_{osc}/128 pada SPI adalah 92 µs, hal ini yang menjadikan komunikasi SPI lebih cepat dari pada komunikasi RS-485 yang waktu pengiriman 1 byta datanya terjadi selama 48 µs. Selain itu protokol pengiriman dan penerimaan data lebih sederhana dibandingkan dengan I^2C dan RS-485. Akan tetapi komunikasi SPI masih memiliki kekurangan dibandingkan dengan I^2C dan RS-485 yaitu memiliki pengkabelan yang banyak.

Abstract

The basis of data communication is a system was build for transmit message or data from transmitter to receiver. At data communication must have fast transmission and to approach real time.

Kind of data communication at serial there is I^2C (Inter-IC Bus), RS-485 and SPI (Serial Peripheral Interface).

SPI communication can used as interface microcontroller AT Mega 8535 with display LCD or connect two of more microcontroller to connect something network. Time to send 1 byte data with value of SCK f_{osc}/128 is 92 µs, so that SPI communication as fast as RS-485 communication, the time to send 1 byte at RS-485 communication is 48 µs. Except that to send and receive data, SPI communication have a simply protocol, but SPI communication have a
Lack, that is have many wiring.

1. Pendahuluan

Komunikasi data pada dasarnya adalah sebuah sistem yang dibangun untuk menyampaikan pesan atau data dari pihak pengirim kepada pihak penerima pesan atau data. Pada pengkomunikasian data dituntut untuk memiliki kecepatan *transmisi* data yang cepat bahkan mendekati *real time*.

Metode komunikasi data yang banyak digunakan adalah komunikasi data secara serial karena kecepatan transmisi datanya lebih cepat daripada komunikasi data secara paralel serta pengkabelannya pada saat akan digunakan lebih mudah dirangkai, sehingga memperkecil faktor kesalahan.

Dilihat dari jenisnya, komunikasi data secara serial terdiri atas I^2C (*Inter-IC Bus*), RS-485 dan SPI (*Serial Pheripheral Interface*), maka dapat dibandingkan metode komunikasi data I^2C (*Inter-IC Bus*) dan RS-485 dengan metode komunikasi data SPI (*Serial Pheripheral Interface*).

Hal ini dilakukan agar dapat diketahui metode komunikasi serial yang efektif dan efisien antara berbagai *device* seperti antara mikrokontroler dengan mikrokontroler lainnya atau dengan *display* LCD.

2. Dasar Teori

2.1. Jenis Komunikasi Data

2.1.1. *Simplex*

Sistem komunikasi ini, terjadi satu arah dari pengirim *(transmitter)* menuju penerima *(receiver)*, hal ini terjadi dengan catatan bahwa penerima tidak dapat mengirim ke pengirim. Blok diagram gambar 1 akan memperjelas komunikasi *simplex* :

Gambar 1. Komunikasi *simplex*

Contoh : CPU mengirim data ke monitor, sedangkan monitor tidak mengirim data ke CPU dan *remote* TV

2.1.2. *Half Duplex*

Sistem komunikasi ini, terjadi secara searah dengan bergantian. Antara dua *device* dapat saling mengirim atau menerima data tetapi tidak dalam waktu yang bersamaan. Blok diagram gambar 2. akan memperjelas komunikasi *half duplex* :

Gambar 2. Komunikasi *half duplex*

Contoh : *Hand-talkie*

2.1.3. *Full Duplex*

Sistem komunikasi ini, merupakan komunikasi data dua arah yang dapat dilakukan secara bersamaan. Pada saat *device* 1 mengirim data, *device* 2 yang dituju dapat juga mengirimkan data ke *device* 1. Blok diagram ini akan memperjelas komunikasi *full duplex* :

Gambar 3. Komunikasi *full duplex*

Contoh : Telepon

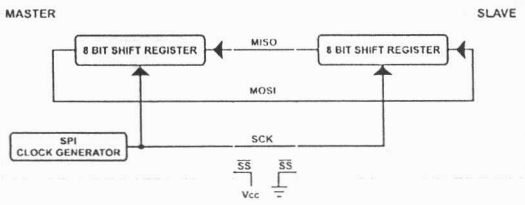
2.2. Cara Kerja Komunikasi SPI (*Serial Peripheral Interface*)

Inti dari komunikasi SPI adalah register geser 8-bit pada *master* dan *slave* serta sinyal *clock* yang dibangkitkan oleh *master* untuk menggeser bit-bit itu sendiri.

Komunikasi SPI membutuhkan empat jalur, tiga diantaranya :

- SCK *(Serial Clock)*
  Merupakan sinyal *clock* yang berfungsi untuk menggeser bit yang yang ada pada register geser.
- MOSI *(Master Out Slave In)*
  Sinyal bit data serial yang akan dikirim dari *master* menuju *slave*
- MISO *(Master In Slave Out)*
  Sinyal bit data serial yang diterima oleh *master* dari *slave*

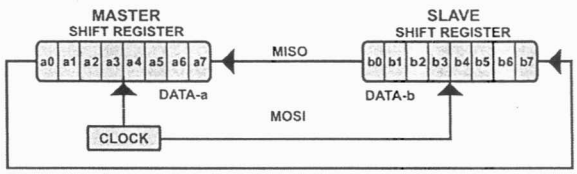
Satu jalurnya lagi yaitu SS` *(Slave Select)* sinyal untuk mengaktifkan *slave*. Jika SS` diset pada logika *high*, maka SPI *slave* akan berfungsi normal dan tidak akan menerima data SPI yang akan masuk maupun mengirim data ke *master*. Jika SS` diset pada logika *low* maka SPI akan aktif sehingga dapat menerima data dan mengirim data SPI ke *master*.



Gambar 4. Koneksi *master-slave*

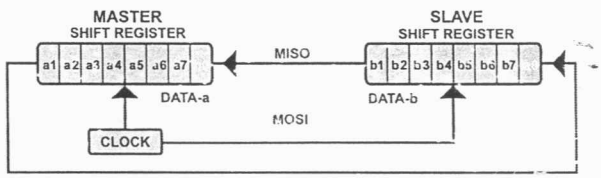
Jika *master* ingin mengirimkan data-a ke *slave* dan dalam waktu yang sama *master* juga menerima data-b dari *slave*. Sebelum memulai komunikasi SPI, *master* meletakkan data-a ke register gesernya dan *slave* juga meletakkan data-b ke register gesernya. Selanjutnya, *master* membangkitkan 8 pulsa *clock* sehingga data pada register geser *master* ditransferkan ke register geser *slave*, dan sebaliknya. Pada akhir pulsa *clock, master* telah menerima data-b dan *slave* telah menerima data-a. Oleh karena data diterima pada saat yang sama, maka komunukasi SPI termasuk komunikasi *full duplex* (komunikasi searah yang dapat saling bertukar data pada saat yang bersamaan).

Kondisi awal *master*



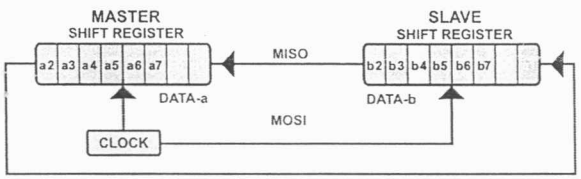
Gambar 5. Kondisi awal *master*

Master menghasilkan pulsa pertama



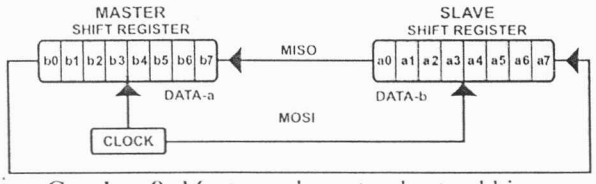
Gambar 6. *Master* pada pulsa pertama

Master menghasilkan pulsa kedua



Gambar 7. *Master* pada pulsa kedua

Master telah menghasilkan pulsa terakhir



Gambar 8. *Master* pada saat pulsa terakhir

2.3. Mikrokontroller AT Mega 8535

Untuk memulai komunikasi SPI pada AT Mega 8535 langkah yang paling penting adalah mengatur *register* SPCR (SPI *Control Register*) dan *register* SPSR (SPI *Status Register*).

SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 9. Konfigurasi SPI *Control Register*– SPCR

SPCR *(*SPI *Control* Register*)* digunakan untuk mengontrol operasi pada SPI seperti mengaktifkan komunikasi SPI, menentukan *device* sebaga *master* atau *slave* dan sebagai penentu sinyal *clock.*

Penjelasan bit pada SPCR :

1. SPIE (*SPI Interrupt Enable*), jika bernilai "1" akan membangkitkan interupsi SPI setelah transfer data selesai.
2. SPE (*SPI Enable*), untuk mengaktifkan SPI.
3. DORD (*Data Order*), jika bernilai "1" maka LSB akan dikirim terlebih dahulu, dan jika bernilai "0" maka MSB akan dikirim terlebih dahulu.
4. MSTR (*Master/Slave Select*), jika bernilai "1" maka AVR sebagai *master*, dan jika bernilai "0" maka AVR sebagai *slave.*
5. CPOL (*Clock Polarity*), digunakan untuk menentukan kondisi diam *clock* atau kondisi pada saat tidak bekerja. Jika CPOL bernilai "1" maka kondisi diam *clock* adalah *high* dan Jika CPOL bernilai "0" maka kondisi diam *clock* adalah *low.*
6. CPHA (*Clock Phase*), digunakan untuk menentukan waktu pengambilan data. Jika CPHA bernilai "1" pengambilan data dilakukan pada transisi turun clock sedangkan jika CPHA bernilai "0" pengambilan data dilakukan pada transisi naik clock.
7. SPR0 dan SPR1 (*SPI Clock Rate*), digunakan untuk menentukan frekuensi dari sinyal SCK.

	Leading Edge	Trailing Edge	SPI Mode
CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)	0
CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)	1
CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)	2
CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)	3

Tabel 1. Mode SPI

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Tabel 2. Konfigurasi frekuensi SCK

SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 10. Konfigurasi SPI *Status Register*– SPSR

SPSR (SPI *Status* Register) digunakan sebagai :

1. SPIF (*SPI Interrupt Flag*), digunakan untuk mengetahui bahwa proses pengiriman data 1 byte sudah selesai dan SPIF akan bernilai "1".
2. WCOL (*Write Collision Flag*), digunakan untuk mengetahui apakah transfer data sedang berlangsung dan jika sedang berlangsung WCOL bernilai "1".
3. SPI2X (*Double SPI Speed Bit*), digunakan untuk melipat gandakan kecepatan SCK menjadi 2 kali. Jika SPI2X bernilai "1" maka kecepatan SCK menjadi 2 kali. Untuk konfigurasi SPI2X dapat dilihat pada tabel 2.2.

SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

Gambar 11. Konfigurasi SPI *Data Register*SPDR

SPDR (*SPI Data* Register) Merupakan register baca/tulis yamg digunakan untuk transfer data.

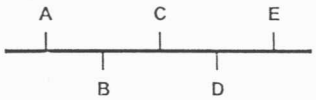
2.4. Topologi Jaringan

Topologi adalah cara yang digunakan untuk menyambung sejumlah terminal dan piranti pemroses data seperti komputer atau mikrokontroller ke dalam suatu jaringan. Terdapat beberapa jenis dalam topologi jaringan.

2.4.1. Topologi *Bus*

Topologi Bus mempunyai karakteristik:

- Merupakan satu kabel yang kedua ujungnya ditutup, dimana sepanjang kabel terdapat *node-node*
- Paling sederhana dalam instalasi
- Masalah terbesar yaitu jika salah satu segmen kabel utama putus, maka seluruh jaringan akan berhenti

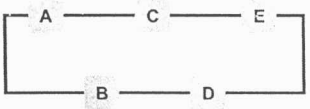


Gambar 12. Topologi bus

2.4.2. Topologi *Ring*

Topologi Ring mempunyai karakteristik:

- Lingkaran tertutup yang berisi *node-node*
- Sederhana dalam *Layout*
- Masalah sama seperti dalam toplogi bus, dimana jika salah satu segmen kabel putus, maka seluruh jaringan akan berhenti

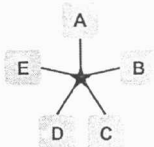


Gambar 13. Topologi *ring*

2.4.3. Topologi *Star*

Topologi Star mempunyai karakteristik:

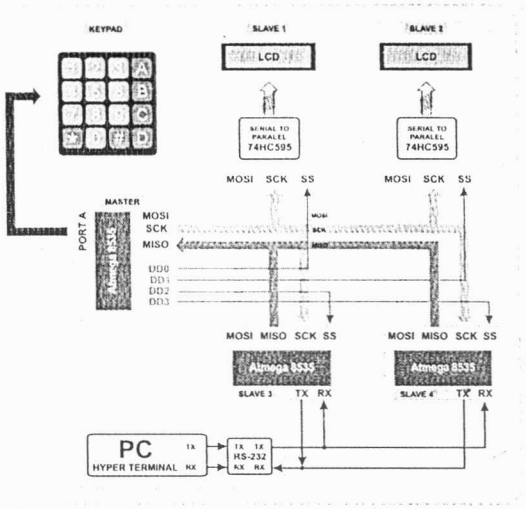
- Setiap node berkomunikasi langsung dengan central node, jalur data mengalir dari node ke *central node* dan kembali lagi
- Mudah dikembangkan karena setiap node hanya memiliki kabel yang langsung terhubung ke *central node*
- Keunggulannya yaitu jika satu kabel *node* terputus maka yang lainnya tidak terganggu



Gambar 14. Topologi *star*

3. Rancangan Sistem

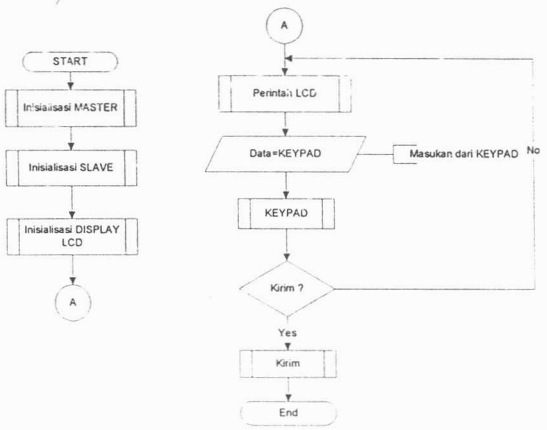
Blok sIstem pada gambar 15. merupakan gambaran keseluruhan sistem yang dirancang untuk komunikasi SPI.



Gambar 15. Gambaran Umum Sistem

Mikrokontroler *master* akan mengirim data ke mikrokontroler *slave* 3 dan *slave* 4 melalui bus MOSI, data yang dikirim berupa bilangan yang terdiri dari beberapa data dalam sekali kirim. *Slave* 1 berfungsi menampilkan data yang dikirim oleh *master*, sedangkan *slave* 2 menampilkan data control register (SPCR), data status register (SPSR) dan data register (SPDR) yang dikirim oleh *slave* 3 dan *slave* 4 melalui bus MISO menuju *master*. Untuk manampilkan data pada *slave* 3 dan *slave* 4 digunakan fitur *hyper terminal* yang ada pada PC. Data yang ditampilkan pada *hyper terminal* adalah data yang dikirim oleh *master* ,serta menampilkan data kontrol register (SPCR) dan data status register (SPDR) pada *slave* 3 dan *slave* 4. Diagram alir dari blok sistem dapat dilihat pada gambar 16.

Jenis jaringan yang digunakan pada sistem ini adalah topologi *bus*. Hal ini dilakukan karena SPI hanya mendukung topologi bus. Jaringan ini juga paling sederhana dalam instalasinya dibanding jenis jaringan lainnya.



Gambar 16. Diagram alir dari *blok* sitem

Konfigurasi port dan pin yang digunakan pada Atmega 8535 sebagai *master* dapat dilihat pada tabel 3

Port	Pin	Keterangan
PORT B	5	Bus MOSI
	6	Bus MISO
	7	SCK
PORT A	0	Untuk mengaktifkan *slave* 0
	1	Untuk mengaktifkan *slave* 1
	2	Untuk mengaktifkan *slave* 2
	3	Untuk mengaktifkan *slave* 3

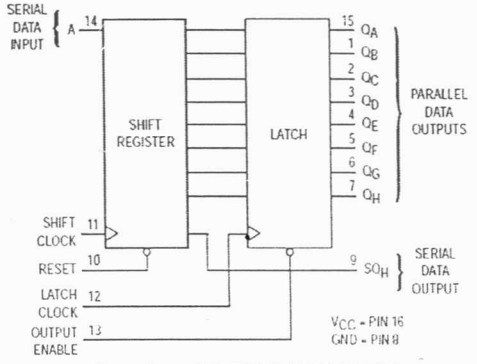
Tabel 3. Konfigurasi port dan pin padai AT Mega 8535 sebagai *master*

Konfigurasi port dan pin yang digunakan pada Atmega 8535 sebagai *slave* dapat dilihat pada tabel 4.

Port	Pin	Keterangan
PORT B	4	SS'
	5	Bus MOSI
	6	Bus MISO
	7	Bus SCK
PORT D	0	RX
	1	TX

Tabel 4. Konfigurasi port dan pin padai Atmega 8535 sebagai *slave*

Untuk *slave* 1 dan 2 karena berupa *display* LCD maka menggunakan 74HC595 sebagi converter untuk mengubah data parallel 8-bit menjadi data serial. Konfigurasi kaki-kaki IC 74HC595N terdapat pada gambar 3.2

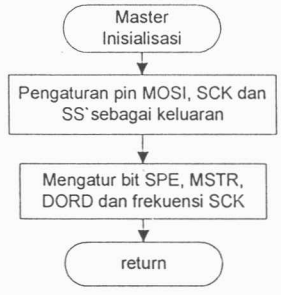


Gambar 17. IC 74HC595N

Inisialisasi SPI harus dilakukan agar komunikasi *master* dengan *slave* dapat berfungsi. Proses inisialisasi meliputi inisialisasi *master* dan *slave*.

Untuk inisialisasi *master* yang harus dilakukan adalah :

1. Register SPE berlogika "1", untuk mengaktifkan komunikasi SPI.
2. Register MSTR berlogika "1", maka mikrokontroler bertindak sebagai *master*.
3. Register DORD berlogika "1", maka MSB dikirim terlebih dahulu.
4. Register SPR0 dan SPR1 berlogika "1", maka frekuensi SCK sebesar $f_{OSC}/128$.
5. DDR *(Data Direct Register)* MOSI dan SCK berlogika "1", karena bertindak sebagai keluaran, sedangkan SS` berlogika "1", karena bertindak sebagai *master*.



Gambar 18. Diagram alir penginisialisasian *master*

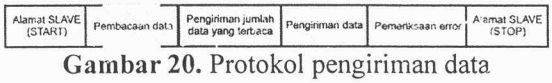
Untuk inisialisasi *slave* yang harus dilakukan adalah :

1. Register SPE berlogika "1",untuk mengaktifkan komunikasi SPI.
2. Register MSTR berlogika "0", mak mikrokontroler bertindak sebagai *slave*.
3. DDR *(Data Direct Register)* MISO berlogika "1", karena sebagai masukkan untuk menerima data dari *master,* sedangkan untuk MOSI, SCK dan SS` berlogika "0" karena bertindak sebagai masukan.

Gambar 19. Diagram alir penginisialisasian *slave*

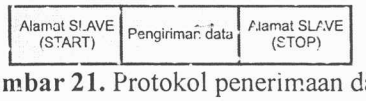
Untuk memilih *slave* dengan memberi logika "0" ke pin SS` yang terdapat pada *slave*, karena *slave* akan aktif jika berlogika "0" (aktif *low*). Sedangkan *slave* tidak akan aktif jika diberi logika "1" oleh *master*, hal ini dapat dimanfaatkan untuk memulai *(start)* dan mengakhiri *(stop)* proses pengiriman data dari *master* ke *slave*.

Untuk proses pengiriman data dari *master* ke *slave* data yang akan dikirimkan disimpan di *buffer* yaitu pada register SPDR dan pengiriman akan berakhir jika status SPIF (SPI *Interrupt Flag*) pada register SPCR berlogika "1". Protokol pengiriman data terdiri dari pengaktifan *slave* (*start)*, pembacaan data, penyimpanan data pada *buffer,* setelah itu pengiriman data dilakukan secara serentak, pemerikasaan *error* dan penon-aktifan *slave* (*stop*). Jika terdapat *error* dalam proses pengiriman data, maka pengiriman data akan diulang kembali.



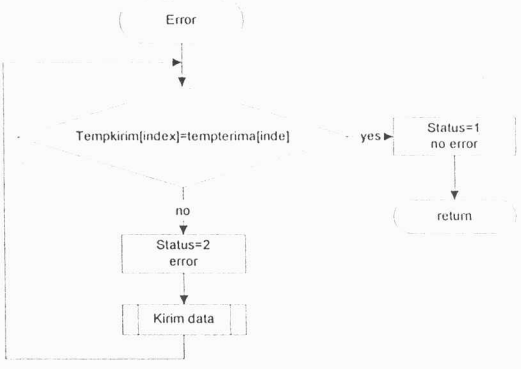
Gambar 20. Protokol pengiriman data

Untuk penerimaan data di *slave* dari *master,* dikarenakan program dalam pengeksekusiannya berjalan secara berurutan dan *slave* tidak mengetahui kapan *master* mengirim data, maka diperlukan sebuah interupsi. Dengan adanya interupsi, ketika *master* mengirimkan data maka rutinitas program akan berhenti dan akan langsung menunjuk alamat interupsi SPI untuk dieksekusi. Program interupsi ini akan dijalankan jika status SPIF (SPI *Interrupt Flag*) pada register SPCR (SPI *Control Register*), SPIE (SPI *Interrupt Enable*) dan bit *Global Interrupt* pada register SREG berlogika "1". Data yang diterima oleh *slave* dari *master* disimpan di *buffer* yaitu pada register SPDR.



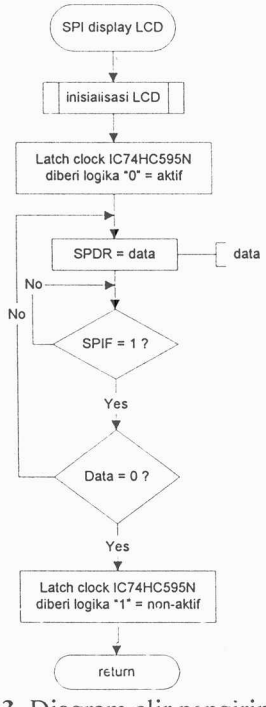
Gambar 21. Protokol penerimaan data

Untuk memeriksa keberhasilan komunikasi SPI dapat dilakukan dengan membaca data yang ada di *slave*, dikarenakan komunikasi SPI bersifat *full duplex* (komunikasi dua arah yang dapat dilakukan secara bersamaan) maka data dari *slave* akan dikembalikan ke *master* melalui bus MISO, sehingga hal tersebut dapat dimanfaatkan untuk memeriksa kembali data yang dikirimkan oleh *master*. Jika data tidak sama, maka dapat dikatakan komunikasi SPI tidak berjalan (*error).* Untuk mengatasi hal tersebut pengiriman data akan di ulang hingga data yang dikirim sama dengan data yang diterima oleh *master*.



Gambar 22. Diagram alir pemeriksaan *error*

Untuk proses pengiriman data dari *master* ke *display* LCD yang bertindak sebagi *slave*, hal yang pertama kali dilakukan adalah melakukan penginisialisasian *display* LCD. Pengiriman data ke *display* LCD bersifat *master transmitter.*



Gambar 23. Diagram alir pengiriman data ke *display* LCD

4. Hasil dan Kesimpulan

Terdapat 2-bit untuk mengendalikan nilai SCK *(Serial Clock),* yaitu SPR0 dan SPR1. Kecepatan SCK dapat digandakan menjadi dua kalinya dengan memberi logika "1" pada SPI2X. *Slave* tidak dapat membangkitkan nilai SCK, yang dapat membangkitkan nilai SCK adalah *master* karena pin SCK pada *master* diatur

sebagai keluaran sedangkan pin SCK pada *slave* diatur sebagai masukan. Untuk melihat nilai dan gelombang SCK yang dihasilkan oleh *master* dapat dilihat dengan menggunakan osiloskop digital terhadap *ground*. Hubungan SCK dan kristal (f_{osx}) dapat dilihat pada tabel 5.

SPI2X	SPR1	SPR0	SCK Frekuensi (secara perhitungan) (kristal) f_{osx} = 11059200	Nilai yang di dapat dari oscilloscope	
				T	f
0	0	0	f_{osx} / 4 = 2,764 MHz	380, 0 ns	2,632 MHz
0	0	1	f_{osx} / 16 = 691,2 KHz	1,46 0 µs	684,9 KHz
0	1	0	f_{osx} / 64 = 172,8 KHz	5,80 0 µs	172,4 KHz
0	1	1	f_{osx} / 128 = 86,40 KHz	11,6 0 µs	86,20 KHz
1	0	0	f_{osx} / 2 = 5,529 MHz	190, 0 ns	5,263 MHz
1	0	1	f_{osx} / 8 = 1,382 MHz	720, 0 ns	1,389 MHz
1	1	0	f_{osx} / 32 = 345,6 KHz	2,90 0 µs	344,8 KHz
1	1	1	f_{osx} / 64 = 172,8 KHz	5,80 0 µs	172,4 KHz

*Gambar gelombang terdapat pada lampiran

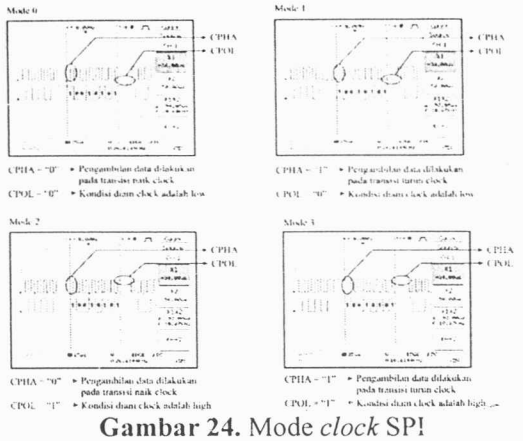
Tabel 5. Perhitungan SCK

Perbedaan SCK secara perhitungan dengan pengukuran tidak terlalu jauh karena masih dalam batas toleransi.

Dengan pengaturan mode SCK maka kecepatan pengiriman data pada komunikasi SPI dapat diatur sesuai dengan kebutuhan dan dengan adanya pengaturan, waktu pengiriman melalui pengaturan mode SCK maka data yang diterima oleh *slave* tidak akan mengalami *error* atau data yang dikirim sesuai dengan data yang diterima.

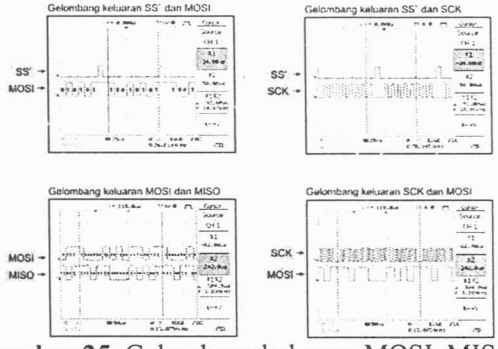
SPI memiliki 4 mode *clock* yang ditentukan oleh nilai CPOL *(Clock Polarity)* dan CPHA *(Clock Phase)*. Jika CPOL berlogika "1" maka kondisi diam *clock* adalah *high* dan jika CPOL berlogika "0" maka kondisi diam *clock* adalah *low*. Sedangkan jika CPHA berlogika "1" maka pengambilan data dilakukan pada transisi turun *clock* dan jika CPHA berlogika "0" maka pengambilan data dilakukan pada transisi naik *clock*.

Kondisi diam *clock* terjadi pada saat mulai dan berakhirnya pengiriman atau penerimaan data, hal ini terjadi karena pin SS` pada *slave* diberi logika "1" oleh *master* untuk menghentikan pengiriman data.



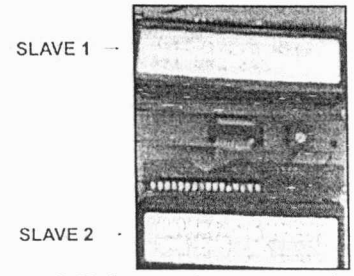
Gambar 24. Mode *clock* SPI

Untuk memeriksa gelombang pada bus MOSI, MISO dan SCK serta pin SS` dilakukan dengan mengirimkan data dari *master* ke *slave* menggunakan osiloskop digital yang diukur terhadap *ground*. Data yang dikirimkan bernilai : 81h, 99h dan E7h.



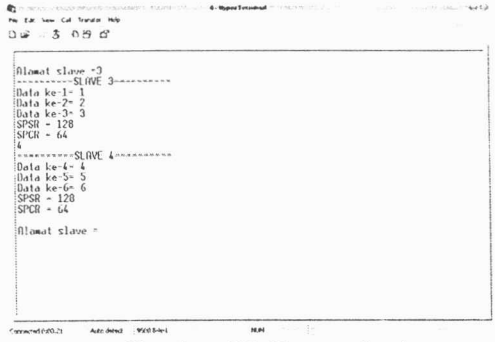
Gambar 25. Gelombang keluaran MOSI, MISO, SCK dan SS`

Untuk pemeriksaan *error* (kondsi data) dapat dilakukan dengan cara mengirimkan data dari *master* ke *slave 3* dan *slave 4*. Data yang terdapat di *master* ditampilkan pada *slave 1 dan slave 2* yaitu pada *display* LCD. *Master* mengirimkan data 1 ,2 dan 3 ke *slave 3* (mikrokontroler) dan data dengan nilai 4 ,5 dan 6 ke pada *slave 4* (mikrokontroler). Data yang dikirimkan tersebut akan dikirimkan kembali oleh *slave* ke *master* melalui bus MISO. Untuk mengetahui error yang terjadi yaitu dengan membandingkan data yang dikirimkan dengan data yang diterima pada *master*, jika data yang dibandingkan adalah sama maka *master* berhasil mengirimkan data. Komunikasi SPI pada sitem yang dibuat seperti pada gambar 15 tidak mengalami *error*, hal ini membuktikan bahwa komunikasi SPI sangat baik dalam pengiriman data maupun penerimaan data.



Gambar 26. Keluaran data error pada *master*

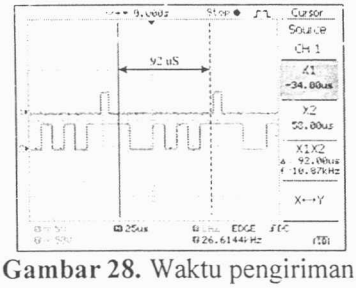
Untuk memantau data maupun memantau nilai status register (SPSR) dan kontrol register (SPCR) pada *slave 3* (mikrokontroler) dan *slave 4* (mikrokontroler) yang dikirimkan oleh *master* dapat dilihat datanya pada fitur *hyper terminal* yang ada pada PC.



Gambar 27. Data pada *slave*

Ketika data dari *master* diterima oleh *slave* maka nilai SPSR (SPI Status Register) adalah 128 (80h), hal ini menyebabkan SPIF (SPI *Interrupt Enable*) berlogika "1" yang artinya bahwa proses penerimaan data sudah selesai. Sedangkan nilai SPCR (SPI Control Register) adalah 64 (40h), hal ini menyatakan SPE (SPI *Enable*)aktif.

Dari hasil pengukuran menggunakan osiloskop didapatkan waktu pengiriman 1 byte data dengan SCK sebesar fosc/128 pada SPI mode 3 adalah 92 µs.



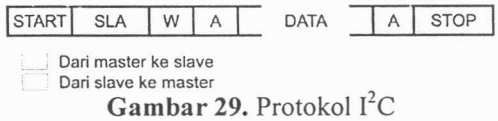
Gambar 28. Waktu pengiriman data

Setiap jenis komunikasi pasti memiliki kelebihan dan kekurangan. Untuk mengetahui kelebihan dan kekurangan komunikasi SPI maka dapat dilakukan dengan cara membandingkannya dengan komunikasi lainnya yaitu komunikasi serial secara I2C dan RS-485 yang telah dibahas pada karya tulis sebelumnya pada tahun 2007.

Waktu pengiriman 1 byte data pada frekuensi *oscilator* (frekuensi pada kristal) sebesar 11,0592 MHZ pada SPI dengan SCK sebesar $f_{osc}/128$ adalah 92 µs, dan pada komunikasi RS-485 waktu pengiriman 1 byta dengan frekuensi *osilator* (frekuensi pada kristal) sebesar 11,0592 MHZ terjadi salama 48 µs. sedangkan pada komunikasi I^2C waktu pengiriman 1 byta dengan frekuensi *osilator* (frekuensi pada kristal) sebesar 11,0592 MHZ terjadi salama 936 µs. Oleh karena itu komunikasi SPI memiliki waktu pengiriman data yang lebih cepat.

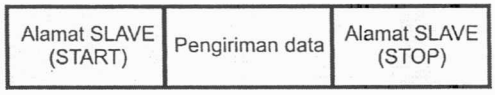
I^2C dan RS-485 memiliki jenis komunikasi data yang bersifat *half duplex,* sedangkan SPI memiliki jenis komunikasi data yang bersifat *full duplex.* Hal ini menjadikan komunikasi SPI lebih menguntungkan, karena dalam satu waktu *slave* dapat mengirimkan data ke *master* sehingga dapat dimanfaatkan untuk pengecekan *error.*

Protokol data secara umum SPI lebih unggul dibandingkan dengan I^2C dan RS-485, karena lebih sederhana. Selain itu RS-485 membutuhkan rangkaian untuk untuk pemilihan *slave.*



Gambar 29. Protokol I^2C

Gambar 30. Protokol RS-485



Gambar 31. Protokol SPI

Pengkabelan pada I^2C dan RS-485 lebih sederhana dari pada SPI, hal inilah yang menjadikan kekurangan SPI.

Jenis komunikasi	Pin	keterangan
I^2C	2 buah	SDA dan SCL
RS-485	2 buah	Tx dan Rx
SPI	3 + n slave	MOSI, MISO ,SCK dan SS`

Tabel 6. Pengkabelan komunikasi data

Berikut tabel perbandingan komunikasi SPI, I^2C dan RS-48

	SPI	RS-485	I^2C
Kec. Transfer data/byte	380 ns - 5,800 µs	48 µs	936 µs

Jenis komunikasi	*Full duplex*	*Full duplex / half duplex*	*half duplex*
Protokol	Pada gambar 31	Pada gambar 30	Pada gambar 29.
Jumlah kabel penghubung	3 + n *slave* buah	2 buah	2 buah

Tabel 7. Perbandingan komunikasi SPI, I^2C dan RS-485

5. Kesimpulan

komunikasi SPI dapat digunakan sebagai *interface* mikrokontroler AT Mega8535 dengan *display* LCD dengan menggunakan sebuah *converter* SPI to paralel. Protokol yang digunakan untuk menampilkan data ke *display* LCD adalah *master transmitter.*

SPI dapat menghubungkan lebih dari dua mikrokontroler dalam suatu jaringan. Waktu pengiriman 1 byte data adalah 380 ns - 5,800 µs, hal ini yang menjadikan komunikasi SPI lebih cepat dari pada komunikasi RS-485 yang waktu pengiriman 1 byta datanya terjadi salama 48 µs dan lebih cepat dari pada komunikasi I^2C 936 µs.

Protokol pengiriman dan penerimaan lebih sederhana dibandingkan dengan I^2C dan RS-485 dan yang keempat adalah kekurangan SPI dibandingkan dengan I^2C dan RS-485 memiliki pengkabelan yang banyak.

Daftar Acuan

1. Kusumayadi, Ibnu . (2007), *Aplikasi Sistem Komunikasi I2C dan RS-485 Berbasis ATMEGA8535 pada Robot Penjejak Garis.* Bandung : Polman Bandung : tidak diterbitkan.
2. Wardhana, Lingga. (2006). *Belajar Sendiri Mikrokontroler seri ATMega 8535.* ANDI: Yogyakarta.
3. Bejo, Agus. (2008). *C dan AVR Rahasia Kemudahan Bahasa C dalam Mikrokontroler ATMega 8535.* Graha Ilmu: Yogyakarta
4. AVR 8535,8535L, (2006). *ATMEL Datasheet AVR ATMEGA 8535,8535L*, rev 2006.
5. Atmel (September 2005). AVR151 : *Software SPI Master.* PDF. www.atmel.com.
6. Atmel (September 2005). AVR320 : Setup And Use of The SPI. PDF. www.atmel.com.