Data Communication Between MMI (Man Machine Interface) Module and Scrambler/Descrambler Module of Scrambler Telephone

Komunikasi Data Antara Modul MMI (*Man Machine Interface*) dan Modul *Scrambler*/*Descrambler* Pada Telepon *Scrambler*

Herlan *,[1], Ade Ramdan[1]

[1] Pusat Penelitian Informatika
Lembaga Ilmu Pengetahuan Indonesia
Gedung 20 lt 3, Jln Sangkuriang 154 D, Bandung
Indonesia

Abstract

This paper proposes a data communication system between MMI module with scrambler/descrambler module. The system is designed and implemented on a scrambler phone. The trasmitted data is needed to determine the speed of randomization sound signal processes and patterns. The data is transmitted through microcontroller based serial communication. A data communication protocol is designed to perform data communication between two modules. The experiment shows serial data communication between the MMI module and the scrambler/ descrambler module with 1200 bps data transmission speed is working properly without errors by a proof that the scrambler/descrambler module sends a "OK" after the MMI module sends data of the speed and pattern of the sound signal randomization that ends with a "AA".

Key Words: Microcontroler, serial data communication , communication protocol

Abstrak

Makalah ini mengusulkan sebuah sistem komunikasi data antara modul MMI dan modul *scrambler/descrambler*. Sistem komunikasi dirancang dan diimplementasikan untuk telepon *scrambler*. Data yang dikirimkan dari modul MMI ke modul *scrambler/descrambler* merupakan data untuk menentukan kecepatan pada saat proses pengacakan dan data pola-pola pengacakan sinyal suara. Data-data tersebut dikirimkan melalui komunikasi serial yang dilakukan oleh mikrokontroler. Untuk melakukan komunikasi data antara dua buah modul tersebut diperlukan suatu aturan tertentu yaitu protokol komunikasi data. Hasil pengujian komunikasi data serial antara modul MMI dan modul *scrambler/descrambler* dengan kecepatan pengiriman data 1200 bps dapat berjalan dengan baik tanpa terjadi kesalahan di mana modul *scrambler/descrambler* dapat mengirimkan tanda OK setelah modul MMI mengirimkan data-data kecepatan dan pola pengacakan sinyal suara yang diakhiri dengan tanda AA.

Kata kunci: Mikrokontroler, komunikasi data serial, protokol komunikasi

1. PENDAHULUAN

Pada penelitian sebelumnya sudah dibuat sepasang Pesawat Telepon *Scrambler* yang digunakan untuk menanggulangi terjadinya proses penyadapan pembicaraan pada saluran telepon oleh pihak lain. Prinsip kerja dari telepon *scrambler/descrambler* adalah pengacakan sinyal suara pada bagian pengirim yang akan ditransmisikan lewat saluran telepon, kemudian dibagian penerima sinyal yang telah diacak tersebut disusun kembali seperti sinyal suara aslinya. Pengacakan sinyal suara terdiri dari 32 pola pengacakan, di mana pola pengacakan tersebut sudah tertanam di dalam mikrokontroler pada Pesawat Telepon *Scrambler*, sehingga *user* tidak bisa melakukan pemilihan kombinasi dari pola-pola pengacakan sinyal suara tersebut [1].

Adanya kebutuhan *user* untuk dapat melakukan perubahan kombinasi pola-pola *scrambler* tersebut, maka dibutuhkan suatu perangkat tambahan yaitu perangkat MMI (*Man Machine Interface*). MMI ini merupakan perangkat sistem interface yang berfungsi sebagai penghubung antara manusia (*user*)

*Corresponding Author. Tel: +6222-2504711
Email: herlan@informatika.lipi.go.id
Received: 7 Sep 2012; revised: 8 Oct 2012; accepted: 22 Oct 2012
Published online: 26 Nov 2012

dan mesin (telepon *scrambler*) yang digunakan pada perubahan kombinasi pola-pola pengacakan informasi suara dan kecepatan proses pengacakan. Melalui MMI ini *user* dengan mudah dapat mengubah kombinasi pola-pola pengacakan suara dan juga mengubah kecepatan proses pengacakan dengan cara memasukan kode-kode pengacakan melalui keypad dan prosesnya ditampilkan melalui LCD.

Modul MMI dibuat tersendiri agar tidak merubah dan mengganggu modul *scrambler/descrambler* yang sudah jadi sebelumnya. Dengan begitu waktu dan cost untuk pengembangan Telepon *Scrambler* dapat kita efesienkan. Adanya dua modul dalam Telepon *Scrambler* tersebut maka dibutuhkan komunikasi diantara kedua modul tersebut. Komunikasi data antara modul MMI dan modul *scrambler/descrambler* yang digunakan adalah melalui media komunikasi serial.

Pada komunikasi serial pengiriman data dilakukan berdasarkan waktu, artinya pengiriman data n-bit dilakukan secara sekuansial per bit sehingga membentuk satu data yang berupa informasi [2].

Format data komunikasi serial terdiri dari parameter-parameter yang dipakai untuk menentukan bentuk data serial yang dikomunikasikan, di mana elemen-elemenya terdiri dari kecepatan mobilisasi data per bit (*baud rate*), jumlah bit data per karakter (*data length*), *parity* yang digunakan dan jumlah stop bit dan start bit [3].

Fitur komunikasi serial ini sudah terdapat pada mikrokontroler yang terpasang pada masing-masing modul, sehingga kita tinggal menggunakan media komunikasi tersebut serta membuat protokol komunikasi sendiri antara modul MMI dengan modul *scrambler/descrambler* pada Telepon *Scrambler*.

2. LANDASAN TEORI

2.1 Operasi *Serial Port*

Mikrokontroler AT89S51/52 atau mikrokontroler standar lainnya mempunyai On Chip *Serial Port* yang dapat digunakan untuk komunikasi secara serial melalui pin RXD dan TXD, di mana tingkat level tegangan komunikasi kedua pin serial menggunakan tingkat tegangan TTL. Level TTL di atas 2 Volt sampai 5 Volt dianggap sebagai level 1, sedangkan tegangan di bawah 0.8V dianggap sebagai level 0 TTL [4].

Pada prinsipnya, komunikasi serial adalah komunikasi di mana transmisi data dilakukan per bit. *Interface* serial hanya membutuhkan jalur yang sedikit (umumnya hanya 2 jalur), sehingga lebih menghemat pin jika dibandingkan dengan interface parallel. Komunikasi serial ada 2 macam, asynchronous serial dan synchronous serial:

(1) *Synchronous* serial adalah komunikasi di mana hanya ada satu pihak (pengirim dan penerima) yang menghasilkan clock dan mengirimkan clock tersebut bersama-sama dengan data. Contoh penggunaan synchronous serial terdapat pada transmisi data keyboard.

(2) *Asynchronous* serial adalah komunikasi di mana kedua pihak (pengirim dan penerima) masing-masing menghasilkan clock namun hanya data yang ditransmisikan, tanpa clock. Agar data yang dikirim sama dengan data yang diterima, maka kedua frekuensi clock harus sama dan harus terdapat sinkronisasi. Setelah ada sinkronisasi, pengirim akan mengirimkan datanya sesuai dengan frekuensi clock penerima. Contoh penggunaan asynchronous serial adalah pada Universal *Asynchronous* Receiver Transmitter (UART) yang digunakan pada serial port (COM) computer. MCS-51 mendukung komunikasi secara asinkron, bahkan 3 dari 4 serial mode yang dimiliki MCS-51 kompatibel dengan UART [5].

Keluarga MCS-51 mempunyai sebuah *register* yaitu SBUF yang terletak pada alamat 99H di mana *register* ini berfungsi sebagai buffer sehingga pada saat mikrokontroler ini membaca data yang pertama dan data kedua belum diterima secara penuh, maka data ini tidak akan hilang.

Pada kenyataannya *register* SBUF terdiri dari dua buah *register* yang memang menempati alamat yang sama yaitu 99H. Register tersebut adalah *Transmit* Buffer Register yang bersifat write only (hanya dapat ditulis) dan *Receive* Buffer Register yang bersifat read only (hanya dapat dibaca). Pada proses penerimaan data dari *Port Serial*, data yang masuk ke dalam *Port Serial* akan ditampung pada *Receive* Buffer Register terlebih dahulu dan diteruskan ke jalur bus internal pada saat pembacaan *register* SBUF sedangkan pada proses pengiriman data ke *Port Serial*, data yang dituliskan dari bus internal akan ditampung pada *Transmit* Buffer Register terlebih dahulu sebelum dikirim ke *Port Serial* [6] .

2.2 *Serial* Register

Serial Register yang digunakan untuk mengatur komunikasi serial terdapat pada *Serial* Control (SCON) [7]. *Port* serial mikrokontroler AT89C51 mempunyai empat buah mode operasi yang diatur oleh bit ke 7 dan bit ke 5 dari Register SCON (*Serial* Control) seperti terlihat pada Gambar 1.

SM0: *Serial Port Mode* bit 0, bit pengatur mode serial.

MSB LSB

SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Gambar 1. Susunan Bit dalam Register SCON

SM1: *Serial Port Mode* bit 1, bit pengatur mode serial.
SM2: *Serial Port Mode* bit 2, bit untuk mengaktifkan komunikasi multiprosesor pada kondisi set.
REN: *Receive Enable*, bit untuk mengaktifkan penerimaan data dari *Port Serial* pada kondisi set. Bit ini diset dan clear oleh perangkat lunak.
TB8: *Transmit* bit 8, bit ke 9 yang akan dikirimkan pada mode 2 atau 3. Bit ini diset dan clear oleh perangkat lunak
RB8: *Receive* bit 8, bit ke 9 yang akan diterima pada mode 2 atau 3. Pada *Mode* 1 bit ini berfungsi sebagai stop bit.
TI: *Transmit Interrupt Flag*, bit yang akan set pada akhir pengiriman karakter. Bit ini diset oleh perangkat keras dan di clear oleh perangkat lunak.
RI: *Receive Interrupt Flag*, bit yang akan diset pada akhir penerimaan karakter. Bit ini diset oleh perangkat keras dan di clear oleh perangkat lunak [8]

2.3 *Mode* Operasi

MCS-51 memiliki 4 mode komunikasi serial. *Mode* 0 berupa synchronous serial (shift *register*), sedangkan tiga mode yang lain yaitu *Mode* 1, 2, dan 3 berupa asynchronous serial (UART). Pada semua mode, pengiriman dilakukan jika ada instruksi yang mengisi nilai *register* SBUF. Dari keempat mode tersebut, satu mode diantaranya bekerja secara sinkron dan 3 lainnya bekerja secara asinkron. Keempat mode kerja tersebut adalah:

(1) *Mode* 0 adalah 8-bit shift *register* di mana data dikirimkan dan diterima melalui pin RXD sedangkan clock dikirimkan dan diterima melalui pin TXD. Kecepatan pengiriman data (baut rate) adalah sebesar 1/12 dari frekuensi kristal.
(2) *Mode* 1, jumlah data yang dikirimkan/diterima sebanyak 10 bit yang terdiri dari start bit, 8 bit data (LSB terlebih dahulu), dan stop bit. Pada proses penerimaan, nilai stop bit akan dimasukkan ke RB8 secara otomatis. Pada proses pengiriman, stop bit akan diberi nilai 1 secara otomatis. Baud rate yang digunakan dapat diatur melalui Time 1.
(3) *Mode* 2, jumlah data yang dikirimkan/diterima sebanyak 11 bit yang terdiri dari start bit, 8 bit data (LSB terlebih dahulu), bit ke-9, dan stop bit. Pada proses pengiriman, nilai bit ke-9 dapat diatur dengan mengisi nilai TB8. Pada proses penerimaan, bit ke-9 akan dimasukkan ke RB8 secara otomatis. Baud rate yang dapat digunakan bisa dipilih antara 1/32 atau 1/64 frekuensi kristal yang digunakan.
(4) *Mode* 3 hampir sama dengan mode 2. Perbedaannya terdapat pada *baud rate* yang digunakan. Jika mode 2 menggunakan *baud rate* yang pasti, mode 3 menggunakan *baud rate* yang dihasilkan oleh Timer 1 [9].

2.4 Kecepatan Pengiriman Data (Baud Rate)

Baud rate adalah frekuensi clock yang digunakan dalam pengiriman dan penerimaan data. Satuan *baud rate* pada umumnya adalah bps (bit per second), yaitu jumlah bit yang dapat ditransmisikan per detik. Baud rate untuk mode 0 bernilai tetap dengan rumus yang terdapat pada persamaan berikut:

$$BaudRate_{Mode_0} = \frac{Freq_{Kristal}}{12} \quad (1)$$

Baud rate untuk mode 2 memiliki 2 variasi tergantung pada nilai bit SMOD pada *register* PCON. Jika SMOD=0, baut rate-nya 1/64 frekuensi kristal, jika SMOD=1 maka baut rate-nya 1/32 frekuensi kristal. Baut rate *Mode* 2 ini mengikuti persamaan berikut:

$$BaudRate_{Mode_2} = \frac{2^{SMOD}}{64} \quad (2)$$

Baud rate untuk mode 1 dan 3 dihasilkan oleh Timer 1 berdasarkan laju limpahan (overflow rate) Timer 1 dan nilai SMOD dengan persamaan berikut:

$$BaudRate_{Mode_{1,3}} = \frac{2^{SMOD}}{32} \times Timer_{1,Overflow} \quad (3)$$

Umumnya Timer 1 dioperasikan pada mode 2 (8-bit Auto Reload) yang hanya menggunakan *register* TH1 saja, sehingga didapat persamaan berikut [10]:

$$BaudRate_{Mode_{1,3}} = \frac{2^{SMOD}}{32} \times \frac{Freq_{Kristal}}{12 \times [256 - TH]} \quad (4)$$

Nilai TH1 bisa didapat pada tabel 2.4, di mana nilai isi-ulang tersebut digunakan untuk menentukan *baud rate*. Pada Tabel 2.4 ditunjukkan *baud rate* yang sering digunakan dengan menggunakan Timer1 [10].

2.5 Pemograman Assembly

Bahasa pemrograman assembly adalah bahasa pemrograman dasar yang pertama dikenal untuk menghubungkan programmer dengan mesin. Sifatnya yang mendekati bahasa mesin memberikan

Tabel I. Baud rate yang sering digunakan dengan Timer1

Baud rate	Fosc (MHz)	SMOD	Timer 1		
			C/T	*Mode*	Nilai isi-ulang
Mode 0 Max:1MHz	12	X	X	X	X
Mode 2 Max:375K	12	1	X	X	X
Mode 1&3:					
62,5K	12	1	0	2	FFH
19,2K	11,059	1	0	2	FDH
9,6K	11,059	0	0	2	FDH
4,8K	11,059	0	0	2	FAH
2,4K	11,059	0	0	2	F4H
1,2K	11,059	0	0	2	E8H
137,5K	11,986	0	0	2	1DH
110	6	0	0	2	72H
110	12	0	0	2	FEEBH

keistimewaan tersendiri, yaitu kecepatan akses. secara umum, semakin dekat bahasa pemrograman dengan mesin, semakin tinggi pula kecepatannya. Pemrograman bahasa assembly menawarkan kelebihan ini. Walaupun saat ini sudah banyak bahasa pemrograman tingkat tinggi (high level language), bahasa pemrograman assembly masih tetap banyak digunakan terutama pada pemrograman perangkat mikrokontroler [11]. Program bahasa assembly berisikan instruksi-instruksi mesin seperti:

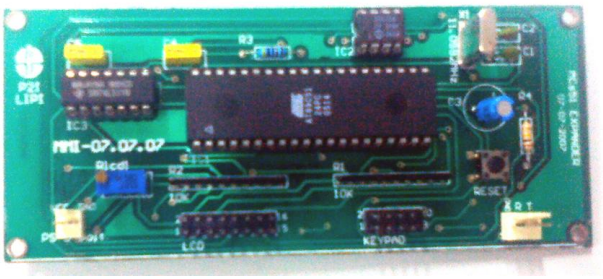
(1) Instruksi MOV merupakan instruksi penyalinan data dari satu lokasi ke lokasi lain.
(2) Instruksi ADD merupakan instruksi akumulator.
(3) Instruksi LCALL merupakan instruksi lompatan atau panggil subrutin dan kembali lagi dari suatu subrutin dengan menggunakan instruksi RET.
(4) Instruksi JNB merupakan instruksi uji bit yang akan mengerjakan lompatan jika bit yang diperiksa isinya 1.
(5) Instruksi SJMP merupakan instruksi lompatan (jump) tanpa syarat.
(6) Instruksi DJNZ merupakan instruksi lompat jika tidak nol digunakan sebagai kontrol loop [12].

3. METODOLOGI PENELITIAN

Dalam melaksanakan penelitian ini, tahap awal yang dilakukan adalah melakukan analisa-analisa terhadap karakteristik media komunikasi, jarak komunikasi dan jenis data yang akan dipergunakan. Langkah analisa ini digunakan untuk menentukan cara menghubungkan Modul MMI dengan Modul *Scrambler/Descrambler* melalui media komunikasi serial. Tahap selanjutnya adalah perancangan software untuk proses pengiriman data serial dengan membuat *flowchart* untuk membuat program dalam bahasa assembler dan di-compile ke dalam bentuk hex untuk ditanamkan ke dalam mikrokontroler. Tahap terakhir adalah proses pengujian untuk mengetahui bahwa secara keseluruhan proses komunikasi antara modul MMI dan modul *Scrambler/Descrambler* dapat berjalan dengan baik sesuai dengan yang diharapkan atau tidak.

4. HASIL DAN PEMBAHASAN

Pada telepon *scrambler* yang telah dibuat tersebut terdapat dua buah modul elektronik yaitu modul MMI seperti pada Gambar 2 dan modul *scrambler/descrambler* seperti pada Gambar 3.



Gambar 2. Modul MMI

Gambar 3. Modul *Scrambler/Descrambler*

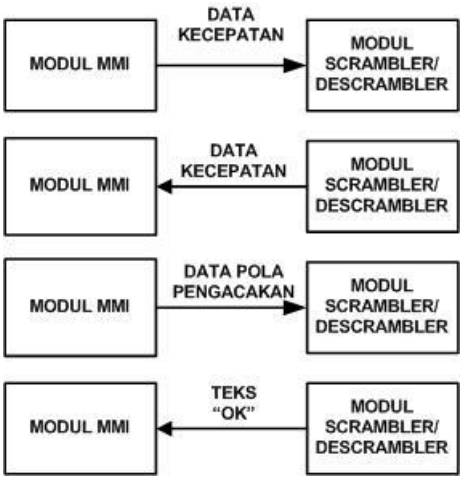
Modul MMI berfungsi untuk mengisi data-data kecepatan dan data pola yang dilakukan oleh *user*, sedangkan modul *scrambler/descrambler* berfungsi untuk melakukan penyambungan telepon dan proses pengacakan data saat terjadi pembicaraan. Modul MMI akan mengirimkan data kecepatan dan data pola pengacakan ke modul *scrambler*.

Hasil dari analisa didapat data yang dikirimkan dari modul MMI ke Modul *Scrambler/Descrambler* adalah data untuk proses kecepatan pengacakan dan data-data pola pengacakan sinyal suara dalam

bentuk bilangan hexadesimal. Data kecepatan proses pengacakan tersebut terdiri dari satu byte data dan data pola pengacakan panjangnya maksimum sebanyak 32 byte yang dapat ditentukan oleh *user*. Jarak komunikasi antara kedua modul adalah 10 cm maka data tersebut dapat dikirimkan langsung melalui pin TXD dan RXD dari mikrokontroler yang digunakan, di mana tingkat level tegangan komunikasi kedua pin serial tersebut menggunakan tingkat tegangan TTL dengan kecepatan pengiriman data yang digunakan adalah 1200 bps.

Urutan pengiriman data yang terjadi antara modul MMI dengan modul secrambler/descrambler pada telepon *scrambler* dapat dilihat pada Gambar 4. Pertama kali yang dikirimkan adalah data untuk menentukan kecepatan pada proses pengacakan yang panjangnya terdiri dari satu byte. Data tersebut akan diterima oleh modul *scrambler/descrambler* dan di simpan di memori. Proses kedua adalah pengembalian data kecepatan dari modul *scrambler/descrambler* untuk diperiksa kembali oleh modul MMI, hal ini dilakukan untuk meyakinkan bahwa data kecepatan proses yang dikirimkan dari modul MMI sudah diterima dengan baik oleh modul *scrambler* dan tidak terjadi kesalahan. Selanjutnya modul MMI akan mengirimkan data-data pola pengacakan yang panjangnya ditentukan oleh pemakai (*user*). Maksimum data pola pengacakan yang dapat dikirimkan adalah 32 byte dan ditambah 1byte data sebagai penunjuk bahwa data sudah selesai dikirimkan semuanya. Data yang digunakan sebagai penunjuk data terakhir tidak boleh sama dengan data-data pola pengacakan maupun data pemilihan kecepatan pengacakan. Pada kegiatan ini ditentukan datanya berupa angka hexadecimal AA. Proses terakhir adalah pengiriman tanda OK dari modul *scrambler/descrambler* yang memberitahukan bahwa semua data sudah diterima. Tanda OK tersebut akan diterima oleh modul MMI.

Flowchart pengiriman data dari modul MMI dapat dilihat pada Gambar 5. Proses pertama kali dalam pengiriman data adalah melakukan inisialisasi serial yaitu untuk menentukan mode serial yang digunakan dan juga untuk menentukan kecepatan pengiriman data. Kecepatan pengiriman data pada penelitian ini dipilih 1200 bps. Selanjutnya mikrokontroler pada modul MMI akan mengirimkan data untuk memilih kecepatan proses pengacakan. Setelah itu akan menunggu jawaban dari modul *scrambler*, apabila dalam waktu 200ms tidak ada jawaban atau data yang dikirim kembali dari modul *scrambler* tidak sama dengan data yang ada di modul MMI maka pengiriman data akan diulangi. Selanjutnya mikrokontroler pada modul MMI akan mengirimkan data-data pola pengacakan yang sudah ditentukan



Gambar 4. Urutan pengiriman data antara modul MMI dengan Modul *Scrambler*

oleh pemakai (*user*). Setelah semua data dikirimkan proses selanjutnya adalah menunggu tanda OK dari modul *scrambler*. Tanda OK tersebut menandakan bahwa data sudah diterima semuanya oleh modul *scrambler*.

Program yang dibuat pada penelitian ini menggunakan bahasa assembler untuk mikrokontroler Atmel AT89C51. Rutin pengiriman data pada modul MMI adalah sebagai berikut:

Rutin inisialisasi serial:

```
ORG 100H
INIT_SER: MOV SCON,#52H
MOV TMOD,#20H
MOV TH1,#0E8H
MOV TCON,#40H
MOV PCON,#00H
```

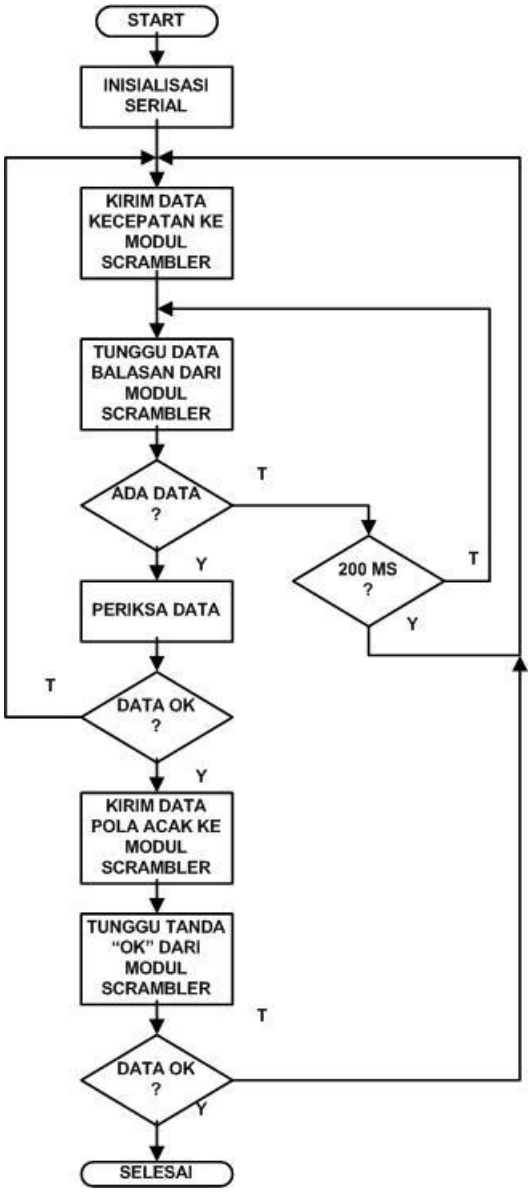
Rutin pengiriman data kecepatan:

```
KIR_HOP MOV R6,#00H
MOV R7,#01H
LCALL BC_1DATA
MOV DT_HOP,A
LCALL SERIAL _OUT
```

Rutin tunggu balasan dari modul *scrambler*:

```
CLR RI
MOV R5,#02H
WDT1: MOV R6,#0C8H
WDT2: MOV R7,#0FAH
WDT3: JNB RI,WDT4
SJMP WDT5
WDT4: DJNZ R7,WDT3
DJNZ R6,WDT2
DJNZ R5,WDT1
SJMP KIR_HOP
WDT5: MOV A,SBUF
CJNE A,DT_HOP,KIR_HOP
```

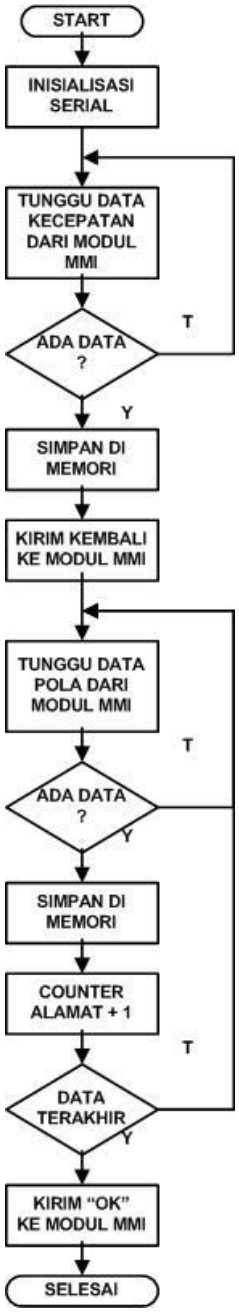
Rutin pengiriman data pola pengacakan:



Gambar 5. *Flowchart* Pengiriman data pada Modul MMI

```
MOV 42H,#08H
MOV R6,#00H
MOV R7,#10H
LCALL ADBC_RAM
LOOP: LCALL IN_8BIT
LCALL MASTER_ACK
LCALL SERIAL_OUT
CJNE A,#0AAH,LOOP
LCALL IN_8BIT
LCALL MASTER_NACK
LCALL STOP_BIT

Subroutin SERIAL_OUT :
SERIAL_OUT: CLR TI
MOV SBUF,A
WAIT1: JNB TI,WAIT1
RET
```



Gambar 6. *Flowchart* penerimaan data pada modul *scrambler*

Gambar 6 Merupakan *flowchart* program pada modul *scrambler* untuk proses penerimaan data yang diterima dari modul MMI. Yang dilakukan pertama kali adalah proses inisialisasi serial, pemilihan untuk *baud rate* harus sama dengan yang diprogram pada modul MMI yaitu 1200 bps supaya bisa terjadi komunikasi data dengan baik. Proses kedua adalah menunggu kiriman data kecepatan pengacakan dari modul MMI. Setelah data diterima data tersebut akan disimpan dimemori dan akan dikirimkan kembali ke modul MMI untuk diperiksa. Proses

selanjutnya adalah menerima data pola pengacakan sampai selesai yang ditandai dengan data terakhir yang berisi angka hexadesimal tertentu yaitu bilangan AA. Setelah semua data diterima program pada modul *scrambler* akan mengirimkan tanda OK ke modul MMI. Rutin program bahasa assembler penerimaan data pada modul *scrambler* adalah sebagai berikut:

Rutin menunggu data kecepatan pengacakan:

```
TRM_HOP: LCALL SERIAL_IN
MOV DT_HOP,A
```

Rutin menerima data pola pengacakan:

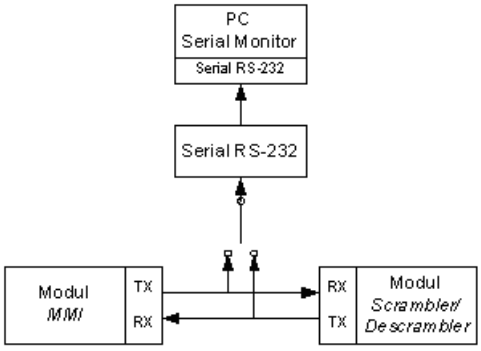
```
MOV R0,#40H
TRM_DT: LCALL SERIAL_IN
MOV @R0,A
INC R0
CJNE A,#0AAH,TRM_DT
```

Subroutin SERIAL_IN:

```
SERIAL_IN: CLR RI
WAIT2: JNB RI,WAIT2
MOV A,SBUF
RET
```

Untuk melihat berhasil tidaknya proses komunikasi data antara modul MMI dan modul *Scrambler/Descrambler* perlu dilakukan pengujian terlebih dahulu yaitu dengan mencoba secara langsung pada alat telepon *scrambler*.

Hasil pengujian dilihat menggunakan software serial monitor pada PC, dengan blok diagram pengujian seperti terlihat pada Gambar 7.

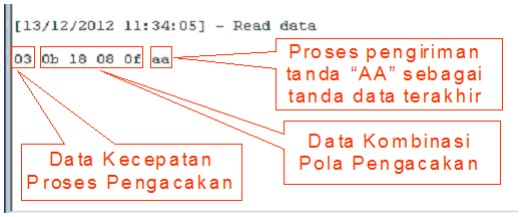


Gambar 7. Blok Diagram Pengujian

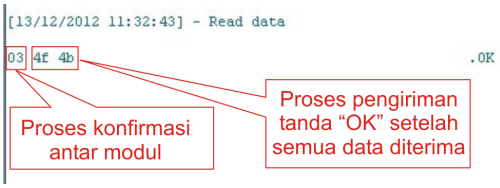
Adapun Pengujian dan hasil pengujian yang telah dilakukan adalah sebagai berikut:

(1) Pengujian modul MMI, yaitu dengan cara mengisi data-data kecepatan dan data pola pengacakan yang dilakukan langsung oleh *user*, lalu dikirimkan ke modul *Scrambler/Descrambler*. Hasil pengujian modul MMI dapat dilihat pada Gambar 8.

(2) Pengujian modul *Scrambler/Descrambler*, yaitu dengan melihat hasil proses konfirmasi antara modul MMI dengan modul *Scrambler/ Descrambler*. Hasil pengujian modul *Scrambler/ Descrambler* dapat dilihat pada Gambar 9.



Gambar 8. Hasil Pengujian pada Modul MMI



Gambar 9. Hasil Pengujian pada Modul *Scrambler/Descrambler*

Dari hasil pengujian yang dilakukan beberapa kali, terlihat proses komunikasi data antara modul MMI dan modul *Scrambler/Descrambler* sudah berjalan dengan baik tanpa terjadinya *error*.

5. KESIMPULAN

Untuk melakukan komunikasi data antara dua buah modul yang menggunakan mikrokontroler diperlukan suatu aturan tertentu yaitu protokol komunikasi data, sehingga data yang dikirimkan dari suatu modul ke modul yang lainnya dapat diterima dengan baik dan benar dan tidak terjadi kesalahan (*error*). Komunikasi data antara modul MMI dengan modul *Scrambler/Descrambler* dilakukan langsung melalui pin TXD dan RXD dari mikrokontroler pada masing-masing modul dengan kecepatan pengiriman data 1200 bps. Dari hasil pengukuran, data yang dikirim dari modul MMI berupa data-data kecepatan dan data pola pengacakan yang diakhiri dengan angka hexadesimal AA dapat diterima dengan baik oleh modul *Scrambler/Descrambler* yang ditandai dengan mengirimkan tanda OK ke modul MMI.

Daftar Pustaka

[1] Herlan, E. A. Gojali, A. Lukman, A. Ramdan, and D. Rianto, "Man machine interface untuk telepon scrambler," in *Prosiding Seminar Pemaparan Hasil Litbang Ilmu Pengetahuan Teknik*, Pusat Penelitian Informatika LIPI, Bandung, Maret 2008, pp. B161–B164.

[2] Sukarman, "Komunikasi perangkat keras menggunakan perangkat lunak matlab," in *Seminar Nasional II SDM Teknologi Nuklir*, December 2006, pp. 373–379.

[3] A. Najmurrokhman and T. Muslim, "Perancangan prototipe sistem pengaturan lampu dan pintu gerbang menggunakan sms (short message service) berbasis mikrokontroler atmega 8535," *Jurnal Tekno Insentif*, 2001.

[4] W. Budiharto, *Interfacing Komputer dan Mikrokontroler*. PT. Elex Media Komputindo, Jakarta, 2004.

[5] D. Susilo, *48 jam kupas tuntas Mikrokontroler MCS51 dan AVR*. Andi Publisher, Yogyakarta, 2010.

[6] P. A. Nalwan, *Teknik Antarmuka dan Pemrograman Mikrokontroler AT89C51*. Elex Media Komputindo, Jakarta, 2003.

[7] Suhata, *Aplikasi Mikrokontroler sebagai Pengendali Peralatan Elektronika via Line Telepone*. Elex Media Komputindo, Jakarta, 2005.

[8] *MCs51 Microcontroller Family Users Manual*, Order No: 272383-002, INTEL, 1994.

[9] D. Susilo, *48 Jam Kupas Tuntas Mikrokontroler MCs51 & AVR*. Andi Publisher, Yogyakarta, 2010.

[10] A. E. Putra, *Belajar Mikrokontroler AT89C51/52/55*. Gava Media, Yogyakarta, 2003.

[11] M. Abdurohman, *Pemograman Bahasa Assembly Konsep Dasar dan Implementasi*. Andi Publisher, Yogyakarta, 2010.

[12] J. Prestiliano, *Strategi Bahasa Asembler*. Gava Media, Yogyakarta, 2007.