Penulis : Avinanta Tarigan

BAB 1
PENDAHULUAN

1.1. Latar Belakang Masalah

Dewasa ini, *MIDI* telah menjadi bagian penting dalam perkembangan teknologi musik dan komputer. Dalam dunia musik, MIDI merupakan perangkat penunjang dalam proses perekaman lagu dan permainan musik di panggung. Bagi pengguna komputer, MIDI dipakai sebagai salah satu perangkat multimedia dan hiburan.

Kombinasi antara seni dan teknologi ini telah membuat para ahli komputer semakin berdedikasi untuk mengembangkan teknologi MIDI. Bukti hasil dedikasi mereka adalah lebih digunakannya salah satu *MIDI Message* yaitu *System Exclusive Message* atau *Sys-Ex*.

Sys-Ex lahir dari kebutuhan untuk mengakses parameter internal dan kontrol internal dari suatu alat musik yang mempunyai kemampuan MIDI (piranti MIDI). Kemampuan ini dibutuhkan karena diterapkannya teknologi mikroprosesor pada piranti MIDI dalam membangkitkan suara.

Unit pembangkit suara pada piranti MIDI membutuhkan sejumlah parameter untuk membangkitkan suara yang spesifik. Seringkali pemusik membutuhkan beberapa perubahan nilai parameter suara untuk membangkitkan suara yang diinginkannya. Parameter suara tersebut dapat disunting melalui panel yang terdapat pada setiap piranti MIDI. Biasanya panel tersebut terdiri dari layar *LCD* yang kecil dan beberapa tombol.

Keterbatasan panel dengan layar yang kecil dan menu yang rumit menjadi kendala dalam menyunting nilai parameter suara tersebut. Apalagi pemusik sering membutuhkan banyak konfigurasi suara, yang tentu saja membutuhkan media penyimpanan yang relatif lebih besar dibandingkan dengan media penyimpanan yang terdapat pada piranti MIDI.

Dengan memanfaatkan Sys-Ex, parameter suara dapat disunting melalui sebuah komputer pribadi yang dilengkapi dengan antar muka MIDI dan program penyunting.

Tetapi perbedaan teknologi yang diterapkan pada masing-masing piranti MIDI menyebabkan format Sys-Ex yang diterapkan juga berbeda. Hal ini menyebabkan program penyunting didisain hanya untuk menangani penyuntingan parameter suara satu jenis piranti MIDI saja.

Salah satu produsen piranti MIDI terkemuka *Roland Corporation* membuat suatu standar teknologi baru yang diterapkan pada beberapa piranti MIDI buatannya. Standar teknologi ini dinamakan *Roland GS Sound Source* atau *Roland GS*. Standar teknologi ini memungkinkan semua MIDI Messages yang dihasilkan oleh satu piranti MIDI Roland GS dapat dijalankan pada piranti MIDI lain yang juga berstandar Roland GS. Standarisasi ini bertujuan agar suara yang dihasilkan oleh semua piranti MIDI Roland GS menjadi identik. Standar teknologi ini juga membakukan parameter suara dan Sys-Ex yang diterapkan pada piranti MIDI Roland GS.

Format Sys-Ex yang berstandar Roland GS ini memungkinkan dibuatnya suatu program penyunting suara *generic*, yang dapat digunakan untuk menyunting parameter suara semua piranti MIDI Roland GS.

1.2. Batasan Masalah

Tulisan ini membahas tentang pemanfaatan Sys-Ex untuk menyunting parameter suara piranti MIDI Roland GS. Maka pembahasannya meliputi : MIDI Messages secara umum, format Sys-Ex secara umum, Sys-Ex Roland GS, sekilas tentang pemrograman antar muka MIDI yang digunakan, dan implementasi program penyunting parameter suara.

1.3. Tujuan Penulisan

Tujuan yang ingin dicapai adalah memanfaatkan Sys-Ex untuk menyunting parameter suara piranti MIDI Roland GS dan diwujudkan dalam bentuk program penyunting parameter suara piranti MIDI Roland GS.

1.4. Metode Penulisan

Penelitian dilakukan dengan studi literatur pada buku-buku yang relevan dengan topik penulisan, yaitu buku-buku referensi yang berkaitan dengan MIDI, Sys-Ex Roland GS, dan pemrograman antar muka MIDI.

Untuk merealisasikan program penyunting ini dibutuhkan bahasa pemrograman terstruktur yang dapat dengan mudah mengakses perangkat keras antar muka MIDI. Oleh karena itu dipilih bahasa pemrograman Pascal dengan kompilator Turbo Pascal 7.0.

Selain itu dibutuhkan sebuah perangkat komputer pribadi yang dilengkapi dengan antar muka MIDI dan sebuah piranti MIDI Roland GS.

Sound Blaster dari *Creative Inc.* terpilih sebagai antar muka MIDI dalam penulisan ini, karena Sound Blaster merupakan antar muka MIDI yang populer dan paling banyak digunakan di kalangan pengguna multimedia berbasis *IBM PC*, dan juga relatif mudah dalam pemrogramannya.

1.5. Sistematika Penulisan

Penulisan ilmiah ini terdiri dari beberapa bab dan sub-bab yang memaparkan tentang landasan teori dan pemrograman aplikasi penyunting. Sistematika penulisan dijabarkan di bawah ini sebagai berikut :

- **BAB 1. PENDAHULUAN**
  Bab ini membahas tentang latar belakang penulisan, tujuan penulisan, batasan masalah, metode penulisan, dan sistematika penulisan.

- **BAB 2. LANDASAN TEORI**
  Membahas tentang landasan teori yang digunakan, antara lain adalah MIDI, klasifikasi MIDI Messages, format Sys-Ex secara umum, dan bagaimana memprogram antar muka MIDI Sound Blaster dalam aplikasi penyunting ini.

- **BAB 3. SYS-EX ROLAND GS DAN PARAMETER SUARA PIRANTI ROLAND GS**
  Bab ini membahas tentang Sys-Ex Roland GS, teknik pengalamatan *Address Mapped Data Transfer*, penanganannya dengan *One Way Transfer Procedure* , dan parameter suara piranti Roland GS.

- **BAB 4. DISAIN DAN IMPLEMENTASI PROGRAM**
  Berisi pembahasan disain dan algoritma program secara global, struktur data yang dipakai oleh program, penjelasan struktur, fungsi, dan prosedur program, serta pengujian program aplikasi penyunting parameter suara piranti MIDI Roland GS.

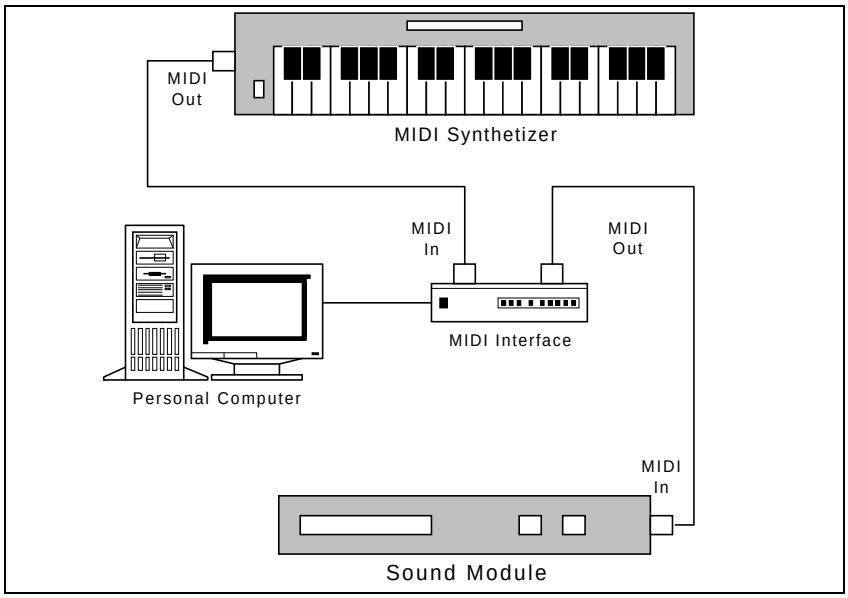
- **BAB 5. PENUTUP**

  Berisi tentang kesimpulan penulisan dan saran-saran untuk pengembangan program penyunting ini lebih lanjut.

BAB 2
LANDASAN TEORI

2.1. MIDI

MIDI adalah akronim dari *Musical Instrument Digital Interface* (Penfold, 1991 : 136). MIDI dibuat oleh gabungan perusahaan *Sequential Circuit* , *Oberheim*, dan *Roland Corporation* (Brian Heywood, 1991 : 6). MIDI adalah antar muka dijital standar yang menghubungkan *synthesizer*, *sequencer*, komputer pribadi, *rythim machine*, dan piranti MIDI yang lain sehingga dapat berkomunikasi dan bertukar data satu sama lain (Penfold, 1991 : 136).



Gambar 2.1. Salah satu cara menggabungkan piranti MIDI dengan menggunakan MIDI

Semua perangkat yang dilengkapi dengan kemampuan MIDI disebut piranti MIDI (*MIDI Device*). Setiap piranti MIDI mempunyai terminal *MIDI Out* atau terminal *MIDI In* atau kedua-duanya. Terminal MIDI Out piranti MIDI harus dihubungkan dengan terminal MIDI In piranti MIDI lain agar keduanya dapat saling berkomunikasi. Salah satu contoh cara menggabungkan piranti MIDI dapat dilihat pada gambar 2.1.

Terminal MIDI In suatu piranti menerima data yang dikirim oleh terminal MIDI Out piranti MIDI lain dalam bentuk rangkaian serial data dijital. MIDI memakai metode transfer data serial yang protokol dan konfigurasi perangkat kerasnya telah dibakukan di dalam *MIDI Specification 1.0*. Karena itu semua piranti MIDI yang menggunakan MIDI terstandarisasi dapat langsung berkomunikasi satu sama lain.

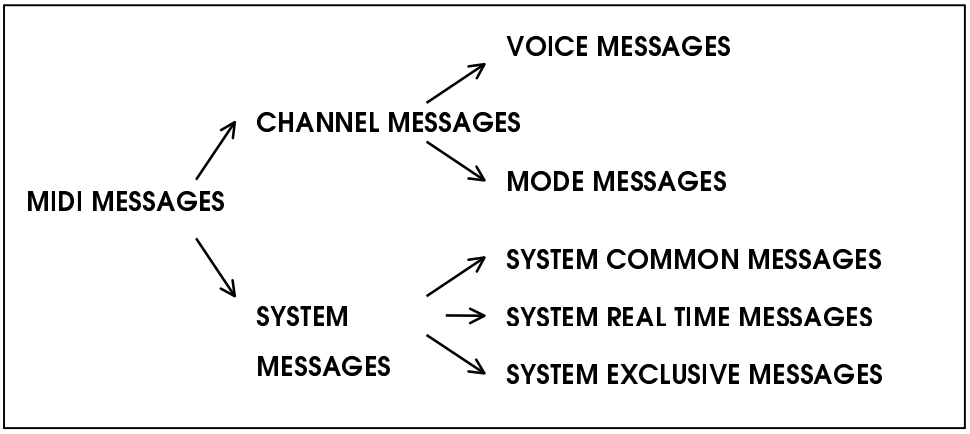
Jadi data yang ditransfer melalui MIDI bukanlah sinyal suara, tetapi berupa sebentuk informasi yang dirangkai dari perintah dan data dalam bentuk sinyal dijital. Informasi tersebut mendeskripsikan apa yang harus dilakukan oleh piranti MIDI yang menerima informasi tersebut lewat terminal MIDI In. Misalnya informasi *note-on* merupakan perintah untuk menyuarakan not tertentu. Informasi note-on ini juga akan dipancarkan dari terminal MIDI Out piranti MIDI, ketika tuts not tertentu piranti MIDI tersebut ditekan.

Semua informasi yang dibawa oleh MIDI disebut *MIDI Messages* yang merupakan bagian dari standar MIDI itu sendiri. MIDI Messages ini mempunyai format perintah dan data tertentu. Format MIDI Messages beserta klasifikasinya dibahas pada sub-bab selanjutnya.

2.2. Klasifikasi dan Format MIDI Messages

Semua informasi yang dikomunikasikan dalam MIDI mempunyai format *multi-byte* yang selalu diawali oleh *status byte* dan diikuti oleh beberapa *data byte*. Status byte bisa diartikan sebagai pengenal tentang data apa yang dibawa oleh MIDI messages tersebut atau yang mengidentifikasi tipe MIDI messages. Status byte terdiri dari delapan *bit* yang MSB-nya (*Most Significant Bit*) selalu bernilai 1. Sedangkan data byte terdiri dari delapan bit yang MSB-nya selalu bernilai 0.

MIDI messages dibagi menjadi dua kategori utama, yaitu *Channel messages* dan *System messages*. Gambar 2.2. menunjukkan ringkasan klasifikasi MIDI messages.

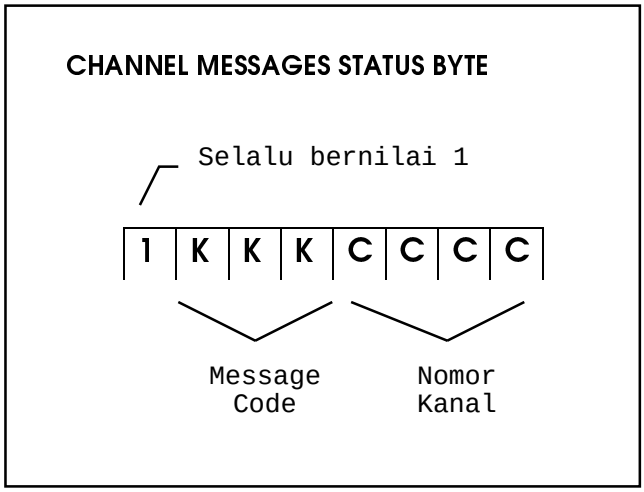


Gambar 2.2. Ringkasan Klasifikasi MIDI Messages

Channel messages merupakan informasi MIDI yang selalu memuat nomor kode kanal atau *Channel*. Kanal adalah fasilitas yang disediakan oleh piranti MIDI agar MIDI messages dapat menyuarakan suara jenis alat musik yang berbeda. Sehingga piranti MIDI dapat menyuarakan suara

beberapa jenis alat musik dalam waktu yang bersamaan. Piranti MIDI standar menyediakan enambelas kanal.

Status byte channel messages terdiri dari dua *nibble* yang terdiri dari empat bit data. Tiga bit rendah dari nibble tinggi menunjukkan *Message Code* , sedangkan empat bit dari nibble rendah menunjukkan nomor kanal dimana data akan dikirim. Gambaran tentang channel messages status byte digambarkan pada gambar 2.3.



Gambar 2.3. Channel Messages Status Byte

Status byte ini dilanjutkan dengan deretan beberapa data byte yang ditetapkan dalam *MIDI Implementation Chart* masing-masing piranti MIDI. Setiap piranti MIDI yang berbeda memiliki kemampuan yang berbeda dalam mengenali MIDI Message.

Voice messages adalah informasi MIDI yang mengontrol suara (voice) alat musik yang dipilih pada kanal. Informasi ini dikirimkan ke seluruh kanal. *Mode*

messages adalah informasi MIDI yang mendefinisikan bagaimana kanal merespon terhadap voice messages.

Contoh dari channel messages adalah informasi *note-on* yang berisi informasi perintah untuk menyuarakan not tertentu. Misalnya ada informasi penekanan not di kanal 6, nomor not 64, dan besar tekanan pada tuts (*velocity*) 127. Maka menurut tabel implementasi MIDI, MIDI messages yang dikirimkan dapat dilihat pada gambar 2.4.

Byte 1	**Byte 2**	**Byte 3**
95h	**40h**	**7Fh**
status byte	*nomor note*	*velocity*

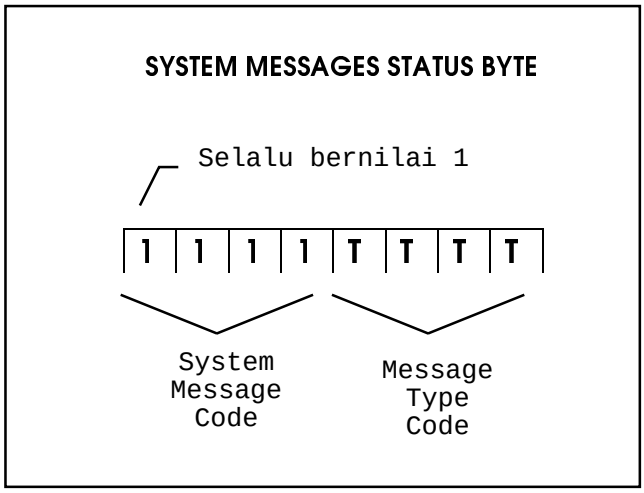
Gambar 2.4. Contoh informasi note-on channel messages

Nibble tinggi status byte bernilai 9h atau 1001b. Tiga bit nibble tinggi status byte bernilai 001b yang menunjukkan code messages untuk note-on. Sedangkan empat bit nibble rendah status byte bernilai 1001b atau 5h yang menunjukkan kanal yang dituju. Nilai kanal dalam channel messages selalu berbasis nol, jadi kanal enam mempunyai nomor 5.

System messages adalah informasi MIDI yang tidak mengkodekan nomor kanal. Informasi ini dikirimkan ke seluruh bagian dari sistem piranti MIDI dan apabila informasi tersebut sesuai, maka bagian sistem yang dituju akan merespon informasi tersebut. Biarpun system messages tidak mengkodekan nomor kanal, tidak menutup kemungkinan bahwa bagian dari sistem yang dimaksud dalam pernyataan di atas adalah kanal. Hal ini bisa dibuktikan pada

pembahasan tentang parameter suara Roland GS di bab selanjutnya.

Status byte system messages terdiri dari dua nibble yang terdiri dari empat bit data. Semua bit pada nibble tinggi diset pada nilai 1 yang mengidentifikasi status byte system message. Sedangkan nibble rendah menunjukkan tipe dari system messages. Penggambaran status byte system messages dijabarkan pada gambar 2.5.



Gambar 2.5. System Messages Status Byte

System messages terdiri dari tiga tipe, yaitu *System Common Messages*, *System Realtime Messages* , dan *System Exclusive Messages*. Sistem common messages digunakan untuk membawa informasi pointer lagu bagi piranti MIDI yang mempunyai peralatan *sequencer*. Sistem commond messages mempunyai status byte yang bernilai **F2h**.

System realtime messages adalah informasi yang dikirim secara kontinyu dalam jeda waktu, untuk mereset semua *MIDI Controller* pada piranti MIDI. Format informasi

MIDI ini merupakan pengecualian, karena hanya terdiri dari status byte saja yang bernilai **FEh** dan tidak mempunyai data byte.

System exclusive message yang akan dibahas dalam tulisan ini mempunyai status byte yang bernilai **F0h**. Format Sys-Ex secara umum akan dibahas pada sub bab selanjutnya.

2.3. Format System Exclusive Message Secara Umum

System exclusive message, seperti yang disebutkan dalam pendahuluan, adalah salah satu MIDI messages yang dikembangkan untuk mengakses parameter internal dan kontrol internal suatu piranti MIDI. Format Sys-Ex secara umum digambarkan pada gambar 2.6.

Byte	Deskripsi
F0h	Sys-Ex Status Byte (SOX)
MFI	Manufacturer ID
DEV	Device ID
D1h	Data Byte
.	.
.	.
.	.
Dnh	.
F7h	End Of Exclusive (EOX)

Gambar 2.6. Format Sys-Ex secara umum

Sys-Ex diawali dengan status byte yang disebut juga *Start Of Exclusive* (SOX) yang bernilai **F0h**. Kemudian diakhiri oleh byte *End Of Exclusive* yang bernilai **F7h**. Status byte diikuti dengan byte *Manufacturer-ID* yang berisi nomor kode perusahaan yang memproduksi piranti

MIDI. Byte ke-tiga menunjukkan *Device ID* yang berisi nomor kode piranti MIDI.

Format Sys-Ex secara umum di atas sudah menjadi standar dan dipakai oleh semua perusahaan pembuat piranti MIDI. Nomor kode perusahaan harus unik karena dianggap sebagai kunci yang menjamin hanya piranti MIDI yang dimaksud yang dapat menterjemahkan Sys-Ex yang dikirimkan. Karena itu nomor kode perusahaan sudah dibakukan dan didaftar oleh asosiasi MIDI internasional dalam MIDI Specification 1.0. Spesifikasi MIDI ini memuat perjanjian internasional tentang spesifikasi standar MIDI.

Sys-Ex memberikan kebebasan pada perusahaan untuk mendefinisikan format Sys-Ex dan cara transfernya masing-masing. Yaitu format deretan data byte dalam Sys-Ex sebelum diakhiri dengan byte EOX. Dalam deretan data byte itulah letak perbedaan format Sys-Ex satu perusahaan dengan perusahaan yang lain, bahkan antara satu piranti MIDI dengan piranti MIDI yang lain meskipun dibuat oleh perusahaan yang sama. Sys-Ex Roland GS dan pemanfaatannya sebagai pokok pembahasan dalam tulisan ini akan dibahas pada bab III.

2.4. Memprogram Sound Blaster Sebagai Antar Muka MIDI

Sound Blaster adalah *sound card* yang diproduksi oleh perusahaan *Creative Labs Inc.* Sound card ini dilengkapi dengan antar muka MIDI yang dalam pemakaiannya harus digabung dengan *Sound Blaster MIDI Cable*.

Alamat port yang digunakan untuk memprogram dan menangani data berdasar pada alamat 2x0 heksa. Nilai x bisa dipilih dari dua atau empat, jadi nilai dasar alamat port Sound Blaster adalah 220h atau 240h.

Sound Blaster mempunyai piranti untuk menangani macam-macam aplikasi suara, FM musik, dan *CD ROM* . Diantara piranti itu adalah *DSP* (*Digital Sound Processor*) yang menangani aplikasi suara dan MIDI. Untuk dapat memakai fasilitas antar muka MIDI, maka perlu diketahui alamat *port* DSP yang dijabarkan pada tabel 2.1.

Alamat Port DSP	Fungsi Alamat Port DSP
2x6h	DSP Reset Port (Write)
2xAh	DSP Read Data Port (Read)
2xCh	DSP Write Data Or Command Port (Write)
2xCh	DSP Write Buffer Status Port *Bit 7* (Read)
2xEh	DSP Data Available Status Port *Bit 7* (Read)

Tabel 2.1. Alamat port DSP dan fungsinya

DSP Reset Port digunakan untuk menginialisasi DSP sebelum dapat digunakan. *DSP Read Data Port* digunakan untuk mengambil data yang masuk ke DSP. *DSP Write Data Or Command Port* digunakan untuk menulis perintah kepada DSP sekaligus untuk menuliskan data kepada DSP. *DSP Write Buffer Status Port* digunakan untuk mengetahui apakah DSP write data or command port siap untuk ditulis. *DSP Data Available Status Port* digunakan untuk mengetahui adanya data yang siap dibaca di DSP read data port.

Sebelum dapat digunakan, DSP harus diinialisasi dahulu. Algoritma prosedur inialisasi DSP dijabarkan sebagai berikut :

```
1. Begin
2.    Write 01h To DSP Reset Port
3.    Delay 3 mikrodetik
4.    Write 00h To DSP Reset Port
5.    Do While DSP Available Status Port and 80h<>80h
6.    End Do
7.    Read Data From DSP Read Data Port
8.    If Data = AAh
9.       then DSP Tereset
10.     else DSP Tidak Tereset
11.   End If
12. End
```

Proses inialisasi DSP memerlukan waktu sekitar 100 milidetik. Bila data terakhir dari DSP read data port belum menunjukkan nilai AAh maka langkah 5 sampai 11 dapat diulang sampai 40 kali. Jika sampai batas ulang tersebut nilai data belum menunjukkan AAh maka dapat dipastikan bahwa Sound Blaster gagal dalam inialisasi.

Untuk menulis perintah atau data pada DSP harus dilakukan algoritma sebagai berikut :

```
1. Begin
2.     Do While DSP Write Buffer Status Port and 80h=80h
3.     End Do
4.     Write Command atau Data To DSP Write Data Or
       Command Port
5. End
```

Jadi untuk menuliskan perintah atau data harus melewati prosedur pemeriksaan terhadap status di DSP write buffer status port. Apabila bit ke-7 bernilai 0 berarti DSP write data or command port dapat ditulis dengan data atau perintah.

Untuk mengirim data dari terminal MIDI Out Sound Blaster dituliskan perintah *MIDI Write* (**38h**) kepada DSP write, kemudian diikuti dengan menuliskan data yang hendak dikirim kepada piranti MIDI.

Untuk menerima data dari terminal MIDI In, Sound Blaster menyediakan dua modus pembacaan. Modus *polling*

menyebabkan CPU secara terus-menerus memantau adanya data MIDI yang masuk. Modus *interrupt* menyebabkan Sound Blaster akan melakukan interupsi ke CPU ketika ada data MIDI yang diterima. Pada program penyunting ini dipilih pemakaian modus interrupt dalam pembacaan data MIDI, dengan pertimbangan bahwa CPU tak perlu terus menerus memantau pemasukan data di terminal MIDI In Sound Blaster. Hal ini meringankan beban CPU dan memudahkan proses penanganan data.

Sound Blaster memakai salah satu interupsi hardware yang disediakan dan dapat dipilih dari *Interrupt Request* (IRQ) 3, 5, atau 7. Agar data dapat diterima, alamat IRQ tersebut dibelokkan pada suatu alamat prosedur yang berisi rutin untuk membaca data MIDI di DSP read data port. Untuk melayani interrupt Sound Blaster, maka *Programmable Interrupt Controller* (PIC) yang mengontrol interupsi hardware ke CPU harus diprogram sesuai dengan IRQ yang digunakan. Prosedur untuk mempersiapkannya adalah sebagai berikut :

1. Begin
2. Menset interrupt vector IRQ yang digunakan ke alamat prosedur pembaca
3. Write perintah *READ Interrupt Mode* (35h) to DSP
4. Menset Bit Interrupt Mask Register dengan menset OCW1 (port 21h) sesuai dengan IRQ yang digunakan
5. Read DSP data available status port
6. end

Prosedur pembacaan DSP data available status port harus dilakukan sebagai *aknowledge* atau persetujuan agar Sound Blaster menangani data yang masuk dengan melakukan interupsi.

Jika Sound Blaster melakukan interupsi akibat adanya data MIDI yang masuk di terminal MIDI In Sound

Blaster, maka prosedur pembaca akan diaktifkan. Prosedur ini harus segera membaca isi DSP read data port yang berisi data MIDI, melakukan prosedur persetujuan agar Sound Blaster dapat menerima data MIDI selanjutnya, serta melakukan prosedur *End Of Interrupt* (EOI) agar PIC dapat melayani interupsi selanjutnya.

BAB 3
SYS-EX ROLAND GS
DAN PARAMETER SUARA PIRANTI MIDI ROLAND GS

3.1. Format Sys-Ex Roland GS

Secara umum format System Exclusive Message Roland GS dijabarkan pada gambar 3.1.

Byte	Deskripsi
F0h	Sys-Ex Status Byte (SOX)
41h	Manufacturer ID (Roland)
DEV	Device-ID
42h	Model ID Untuk Roland GS
CMD	Command ID
[Body]	Main Data
	.
	.
	.
F7h	End Of Exclusive byte (EOX)

Gambar 3.1 Format Sys-Ex Roland GS secara umum (Roland, 1992 : 59)

Sys-Ex Roland GS selalu didahului dengan Sys-Ex status byte yang bernilai **F0h**, kemudian diikuti dengan nomor kode *Roland Manufacturer ID* yang bernilai **41h** dan nomor kode piranti MIDI (Device-ID). Nomor kode piranti ini mengandung nilai unik yang mengidentifikasi piranti MIDI tertentu. Nomor kode ini biasanya berkisar antara 00h-0Fh dan dapat ditetapkan pada piranti MIDI.

Model-ID mengandung nilai unik untuk mengidentifikasikan model data yang dikandung oleh Sys-

Ex. Model data Roland GS diidentifikasikan dengan Model-ID yang bernilai **42h**. *Command-ID* mengindikasikan tipe fungsi dari Sys-Ex yang membawa nilai Command-ID tersebut. Tipe fungsi ini dijelaskan pada sub bab berikutnya.

Bagian *Main data* berisi deretan data yang akan ditransfer lewat MIDI. Besarnya data, format data, dan isi dari data bervariasi menurut Model-ID dan Command-ID. Hal ini akan dibahas pada sub-bab berikutnya yang membahas teknik pengalamatan dan teknik pentransferan Sys-Ex Roland GS.

3.2. Address Mapped Data Transfer Dengan One Way Transfer Procedure.

Data yang dikandung Sys-Ex akan dikirimkan ke seluruh bagian dari sistem piranti MIDI. Untuk menspesifikasikan bagian dari sistem yang diharapkan akan merespon Sys-Ex yang dikirimkan, Roland mengembangkan teknik pengalamatan data dalam Sys-Ex.

Address Mapped Data Transfer (AMDT) adalah teknik pengalamatan data pada Sys-Ex Roland GS, agar data yang ditransfer lewat MIDI sampai pada lokasi yang spesifik dari bagian sistem yang dituju. Data ini dapat berupa isi memory tertentu dari piranti MIDI, parameter-parameter suara, status kontrol internal, dan parameter-parameter lainnya.

Teknik pengalamatan ini terdiri dari tiga byte alamat (*Address*) dengan tiga byte besar data (*Size*) yang masing-masing MSB-nya bernilai nol. Teknik AMDT ini memungkinkan satu byte data mempunyai satu alamat yang unik. Jika ada suatu deretan data yang banyaknya direpresentasikan dalam Size, maka alamat yang

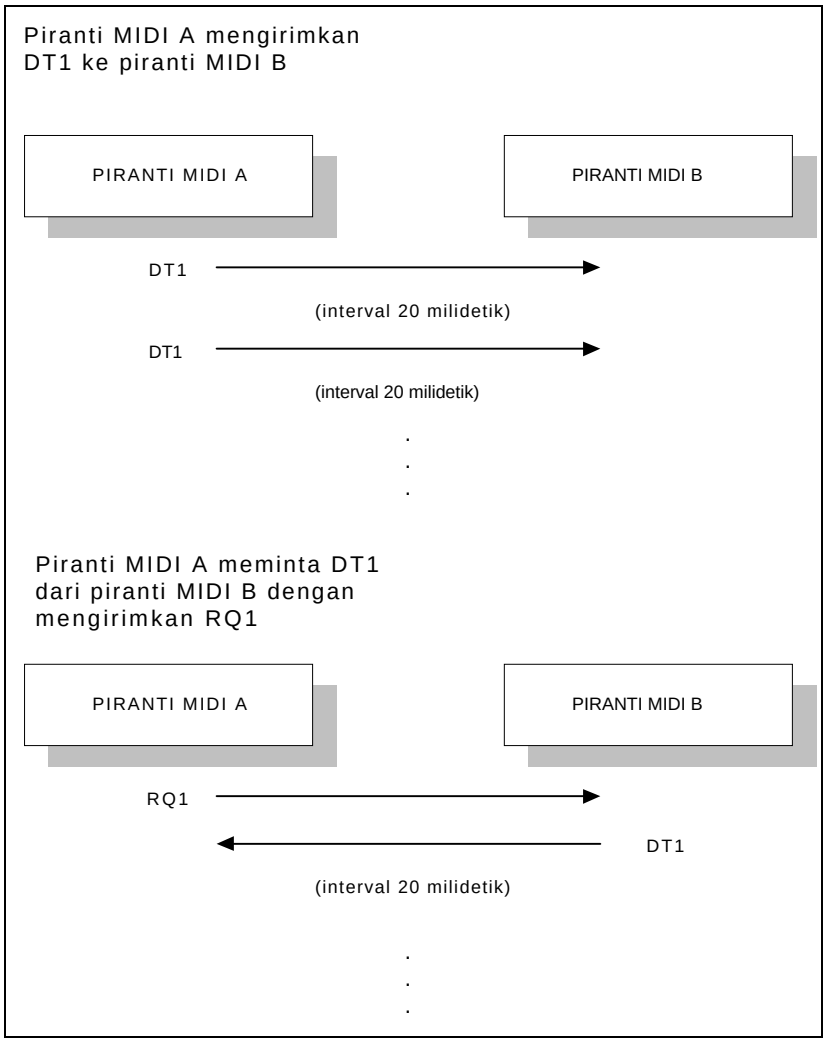
dicantumkan pada Sys-Ex hanya data yang ada di urutan pertama, dengan syarat bahwa deretan data yang mengikutinya mempunyai alamat yang berurutan dan dalam satu jenis lokasi yang ditetapkan pada implementasi Sys-Ex Roland GS. Penggunaan Address dan Size ini dijelaskan pada pembahasan paket Sys-Ex RQ1 dan DT1.

Teknik pengalamatan AMDT memungkinkan digunakannya dua prosedur transfer data, *One-Way Transfer Procedure* dan *Handshake Transfer Procedure* . Handshake Transfer Procedure memungkinkan terjadinya jabat-tangan (handshake) antara piranti MIDI sebelum transfer data dilakukan. Prosedur ini menjamin keandalan data yang dikirimkan, apalagi bila bagian Main Data pada Sys-Ex terdiri dari deretan data yang panjang.

Tetapi tidak semua piranti MIDI Roland GS mempunyai kemampuan Handshake Transfer Procedure. Oleh karena itu dalam penulisan ini digunakan One-Way Transfer Procedure untuk mengirimkan Sys-Ex Roland GS. Apalagi parameter suara yang akan digunakan nanti hanya terdiri dari satu byte saja untuk setiap pengiriman. Jadi masih relatif aman dan handal jika digunakan One-Way Transfer Procedure untuk mentransfernya. Disamping itu prosedur transfer ini memang ditujukan untuk mengirim data yang jumlahnya relatif sedikit. One-Way Transfer Procedure mentransfer semua data Sys-Ex dari awal data (SOX) sampai akhir data (EOX).

Ada dua tipe Sys-Ex dalam One-Way Transfer Procedure menurut fungsinya, yaitu Sys-Ex *Request Data 1* (RQ1) dan Sys-Ex *Data Set 1* (DT1). Sys-Ex DT1 berfungsi untuk mengirimkan data kepada piranti MIDI. Data beserta alamatnya ada dalam bagian Main Data Sys-Ex DT1 ini. Sedangkan Sys-Ex RQ1 berfungsi untuk memerintahkan

piranti MIDI agar mengirimkan data yang sudah dispesifikasikan alamat dan besar datanya pada bagian Main Data Sys-Ex RQ1. Penggambaran mekanisme One-Way Transfer Procedure ini digambarkan pada gambar 3.2.



Gambar 3.2 One-way Transfer Procedure untuk paket Sys-Ex RQ1 dan DT1. (Roland, 1992 : 60)

Setelah menerima Sys-Ex RQ1, maka piranti MIDI akan melakukan pemeriksaan internal terhadap alamat dan

besar data yang diminta. Jika sesuai, maka piranti MIDI tersebut akan mengirimkan data yang diminta dengan Sys-Ex DT1. Format Sys-Ex DT1 dan RQ1 dijabarkan pada gambar 3.3 dan gambar 3.4

Byte	**Deskripsi**
F0h	Sys-Ex status byte
41h	Manufacturer ID (Roland)
DEV	Device ID
42h	Model ID untuk Roland GS
12h	Command ID untuk DT1
AAh	Address MSB
.	.
.	.
.	LSB
DDh	nData
.	.
.	.
.	.
SUM	Check SUM
F7h	End of Exclusive Byte

Gambar 3.3 Format Sys-Ex Data Set 1
(Roland, 1992 : 60)

Bagian Main Data dalam Sys-Ex DT1 terdiri dari tiga byte alamat (address MSB-LSB), deretan data (nData), serta sebuah byte *Check Sum*. Penggunaan check sum disini adalah sebagai teknik pemeriksaan kesalahan data dalam pengiriman. Byte ini mempunyai pola bit tertentu yang disusun dengan tujuan apabila tiga byte alamat, deretan data, dan byte check sum dijumlahkan akan membentuk pola 7-bit 0000000b. Apabila byte check sum ini tidak membentuk pola bit seperti yang dijelaskan di atas maka

data yang dikirimkan melalui Sys-Ex DT1 adalah tidak sah atau terjadi kesalahan selama pengiriman.

Byte	Deskripsi
F0h	Sys-Ex status byte
41h	Manufacturer ID (Roland)
DEV	Device ID
42h	Model ID untuk Roland GS
11h	Command ID untuk RQ1
AAh	Address MSB
.	.
.	.
.	LSB
SSh	Size MSB
.	.
.	.
.	LSB
SUM	Check SUM
F7h	End of Exclusive Byte

Gambar 3.4 Format Sys-Ex Requset Data 1
(Roland, 1992 : 59).

Bagian main data pada Sys-Ex RQ1 terdiri dari tiga byte alamat data (address MSB-LSB), tiga byte besar data yang diminta (Size MSB-LSB), serta sebuah byte check sum. Byte check sum pada Sys-Ex RQ1 mempunyai pola bit yang disusun dengan tujuan agar apabila ketiga byte alamat, ketiga byte besar data, dan byte check sum dijumlahkan akan menghasilkan pola 7-bit 0000000b.

Yang perlu diperhatikan dalam pertukaran data dengan Sys-Ex RQ1 maupun DT1 adalah waktu jeda (*time interval*) diantara pengiriman. Waktu jeda ini memberikan

piranti MIDI kesempatan untuk siap melayani lagi masuknya Sys-Ex.

3.3. Parameter Suara Piranti MIDI Roland GS

Ada delapan jenis parameter suara yang dapat disunting pada piranti MIDI Roland GS. Masing-masing terdiri dari satu byte data saja dan dispesifikasikan pada alamat **40h 11h 30h** sampai alamat **40h 15h 37h** . Parameter suara beserta alamat dan deskripsinya dijabarkan pada tabel 3.1.

Alamat (Heksa) MSB - LSB	Size (Heksa) MSB - LSB	Range (heksa)	Deskripsi	Default (heksa)
40 1N 30	00 00 01	0E - 72	Vibrato Rate	40
40 1N 31	00 00 01	0E - 72	Vibrato Depth	40
40 1N 32	00 00 01	0E - 50	TVF CutOff Frequency	40
40 1N 33	00 00 01	0E - 72	TVF Resonance	40
40 1N 34	00 00 01	0E - 72	TVF & TVA Envelope Attack	40
40 1N 35	00 00 01	0E - 72	TVF & TVA Envelope Decay	40
40 1N 36	00 00 01	0E - 72	TVF & TVA Envelope Release	40
40 1N 37	00 00 01	0E - 72	Vibrato Delay	40

Tabel 3.1. Tabel parameter suara piranti MIDI Roland GS.

Nilai **N** disini adalah nomor kanal dimana lokasi parameter suara tersebut berada. Berbeda dengan channel messages nomor kanal dalam AMDT dimulai dengan satu (01h-0Fh). Yang perlu diperhatikan disini adalah pengecualian penomoran kanal di atas kanal 10. Menurut standar Roland GS, kanal 10 ditujukan untuk suara alat musik *drum* dan perkusi yang memiliki parameter suara tetap, sehingga parameter suara pada kanal 10 tidak dapat disunting. Dengan pengecualian tersebut di atas, kanal di atas kanal 10 diberi nomor N-1 (N adalah kanal).

Bagaimana parameter-parameter suara tersebut ditransfer melalui MIDI, akan dicontohkan sebagai berikut. Misalnya parameter suara *Vibrato Depth* di kanal 4 piranti MIDI Roland GS JV-30 (Device-ID 10h) hendak dirubah dengan nilai 76h, maka dikirimkan Sys-Ex DT1 dengan byte berturut-turut sebagai berikut :

• F0h	SOX
• 41h	Roland Manufacturer ID
• 10h	Nomor Piranti JV-30
• 42h	Model Data Roland GS
• 12h	Command-ID DT1
• 40h	Alamat AMDT Vibrato Depth
• 14h	di channel 4
• 31h	MSB - LSB
• 76h	Nilai Parameter Vibrato Depth
• 05h	Nilai Check SUM
• F7h	EOX

Nilai check sum disini bernilai 05h yang didapat dengan perhitungan tertentu sedemikian rupa sehingga apabila byte alamat (40h+14h+31h), data (76h), dan chek sum (05h) dijumlahkan akan membentuk pola 7-bit 0000000b.

Untuk mengambil parameter suara dari piranti MIDI, pertama harus dikirimkan Sys-Ex RQ1, yang apabila nilai check sum valid maka piranti MIDI akan mengirimkan Sys-Ex DT1 pembawa parameter yang diminta.

Misalkan ingin diketahui nilai parameter *TVA Cutoff Frequency* piranti MIDI Roland GS JV-30 pada kanal 15. Maka akan dikirimkan Sys-Ex RQ1 dengan byte berturut-turut sebagai berikut :

• F0h	SOX
• 41h	Roland Manufacturer ID
• 10h	Nomor Piranti JV-30
• 42h	Model Data Roland GS

• 11h	Command-ID RQ1
• 40h • 14h • 32h	Alamat AMDT TVA Cutoff Freq. di kanal 15 MSB - LSB
• 00h • 00h	Besar (Size) TVA Cutoff Freq. MSB - LSB
• 01h	
• 79h	Nilai Check SUM
• F7h	EOX

Dalam urutan byte di atas terlihat bahwa pengecualian penomoran kanal di atas kanal 10, yaitu nomor kanal 15 direpresentasikan dengan nilai 14. Nilai check sum 79h didapat dari perhitungan tertentu sedemikian rupa sehingga apabila byte alamat (40h+14h+32h), Size (00h+00h+01h), dan byte check sum (79h) dijumlahkan akan membentuk pola 7-bit 0000000b.

Dengan bekal landasan teori MIDI, Sys-Ex Roland GS, dan pemrograman Sound Blaster sebagai antar muka MIDI, pemanfaatan Sys-Ex untuk menyunting parameter suara piranti MIDI Roland GS dapat diwujudkan dalam bentuk suatu program penyunting. Desain dan implementasi program tersebut akan dibahas pada bab selanjutnya.

BAB 4
DISAIN DAN IMPLEMENTASI PROGRAM

4.1. Desain dan Algoritma Program Secara Global.

Dalam mendisain program penyunting ini, hal yang menjadi pertimbangan sebagai tujuan dari desain program adalah :

- Penyuntingan dapat dilakukan dengan mudah. Hal ini dikarenakan Roland GS menetapkan 8 parameter di setiap kanal sehingga parameter yang disunting berjumlah 8 x 15 atau 120 parameter
- Perubahan nilai parameter suara dapat segera diketahui dengan segera mengirimkan data ketika terjadi penyuntingan dan dilengkapi dengan fasilitas untuk memperdengarkan contoh suara.

Karena jumlah parameter suara yang hendak disunting relatif banyak, maka program didisain agar penyuntingan dilakukan per-kanal pada layar penyunting. Setiap pergantian kanal yang aktif dalam layar penyunting dilakukan pengambilan nilai parameter suara dari piranti MIDI, sehingga setiap perubahan nilai parameter suara dapat diketahui.

Berikut ini adalah penjabaran algoritma program secara global :

```
1. Begin
2.    Lakukan Deteksi Sound Blaster
3.    If Sound Blaster Tidak Ada
4.    then Hentikan Program
5.    end if
6.       Reset DSP Sound Blaster
7.       Begin
         { Transfer Parameter Suara dari Piranti MIDI
           ke PC dengan Sys-Ex RQ1 }
         loop
```

```
                    kirim Sys-Ex RQ1 per byte
                end loop
                Terima parameter suara dalam Sys-Ex DT1
                If Error then Tampilkan Pesan
                end if
             end
 8.     Lakukan Prosedur Peyuntingan Nilai Parameter
 9.     If terjadi penyuntingan
                then
                Begin
                { Transfer parameter suara tersunting
                  dengan Sys-Ex DT1 kepada piranti MIDI }
                  loop
                      kirim Sys-Ex DT1 per byte
                  end loop
                end
10.     end if
11. end.
```

Pertamakali program harus mendeteksi keberadaan Sound Blaster beserta alamat port yang digunakan. Jika Sound Blaster terdeteksi keberadaannya, program harus melakukan reset pada DSP Sound Blaster sebelum dapat digunakan. Sebelum dilakukan proses penyuntingan, parameter suara yang hendak disunting harus diambil dari piranti MIDI terlebih dahulu. Pengambilan parameter suara dilakukan dengan metoda One-Way Transfer Procedure untuk Sys-Ex RQ1 pada piranti MIDI yang segera akan mengirimkan parameter suara yang diminta dalam Sys-Ex DT1. Setelah penyuntingan dilakukan, nilai parameter yang dirubah dikirimkan kembali ke piranti MIDI dengan Sys-Ex DT1.

Dalam implementasi algoritma tersebut nantinya, pengiriman DT1 segera dilakukan setiap terjadi perubahan nilai parameter suara agar efek penyuntingan bisa segera didengar. Sedangkan pengambilan parameter dilakukan setiap penggantian kanal yang aktif pada layar penyunting.

4.2. Implementasi Program.

Program penyunting ini ditulis dalam bahasa pemrograman terstruktur Pascal dengan memakai kompilator Turbo Pascal 7.0. Program penyunting ini dibagi dalam modul program berbentuk *unit* dan satu program utama penyunting. Modul program untuk menangani Sound Blaster sebagai antar-muka MIDI ditulis dalam unit “SBMIDI.PAS”, sedangkan program utama ditulis dalam program “GS_EDIT.PAS”. Selain itu dibutuhkan modul program untuk menangani tampilan pada layar penyunting yang ditulis dalam unit “WINOOP.PAS”.

4.2.1. Struktur Data

Pada program penyunting utama dibutuhkan beberapa variable dan konstanta global untuk memuat semua data yang diperlukan dalam proses penyuntingan. Beberapa variabel yang membutuhkan ruang memori besar, seperti parameter suara dan daftar nama suara Roland GS, menggunakan lokasi *heap* di memory yang terbentuk secara dinamis. Hal ini ditujukan untuk mengatasi keterbatasan segment data yang disediakan hanya sebesar 64kb saja untuk variabel statis.

Untuk menyimpan semua parameter suara beserta tone yang aktif pada setiap kanal diperlukan variabel dinamik **DataTones** yang dideklarasikan dengan tipe pointer **pDataTones**. Struktur data variabel ini dideklarasikan sebegai berikut :

```
Const
    cChanMax          = 16;
    cMDFMax           = 8;
Type
   tDataTone          =  Record
                          TNum : byte;
```

```
                                Modi : array[1..cMDFMax] of
                                       byte;
                               end;
    pDataTone          = ^tDataTones;
    tDataTones         = array[1..cChanMax] of tDataTone;
Var
    DataTones          : pDataTone;
    FileParam          : file of tDataTone;
```

Struktur data dari variabel dinamik ini adalah berupa tipe *array* **tDataTones** yang memuat tipe *record* **tDataTone** untuk setiap kanal. Record ini terdiri dari **Tnum** yang mengandung nomor tone yang aktif pada setiap kanal dan **Modi** dengan struktur data array yang memuat kedelapan nilai parameter suara. Semua perubahan nilai parameter suara dan nomor tone yang aktif pada setiap kanal disimpan pada variabel ini. Variabel berkas **FileParam** dipakai untuk mendefinisikan berkas yang digunakan untuk menyimpan nilai parameter hasil penyuntingan pada tiap kanal pada berkas dengan struktur data yang sama pada **tDataTone**, untuk setiap recordnya.

Alamat AMDT parameter suara beserta nama, nilai terbesar, dan terendahnya dideklarasikan dalam konstanta ToneMDF sebagai berikut :

```
Type
    tToneMod           = Record
                           Nama          : string[25];
                           ADM, ADL,
                           VMin, VMax    : byte;
                         end;
Const
     cMDFMax           = 8;
     ToneMDF           : array[1..cMDFMax] of tToneMod =

                        (( Nama : 'Vibrato Rate' ;
                           ADM  : $10 ; ADL  : $30 ;
                           VMin : $0E ; VMax : $72 ),

                         ( Nama : 'Vibrato Depth' ;
                           ADM  : $10 ; ADL  : $31 ;
                           VMin : $0E ; VMax : $72 ),
```

```
( Nama : 'TVF Cutoff Freg.' ;
  ADM  : $10 ; ADL  : $32 ;
  VMin : $0E ; VMax : $50 ),

( Nama : 'TVF Resonance' ;
  ADM  : $10 ; ADL  : $33 ;
  VMin : $0E ; VMax : $72 ),

( Nama : 'TVA+TVF Env. Attack' ;
  ADM  : $10 ; ADL  : $34 ;
  VMin : $0E ; VMax : $72 ),

( Nama : 'TVA+TVF Env. Decay' ;
  ADM  : $10 ; ADL  : $35 ;
  VMin : $0E ; VMax : $72 ),

( Nama : 'TVA+TVF Env. Release' ;
  ADM  : $10 ; ADL  : $36 ;
  VMin : $0E ; VMax : $72 ),

( Nama : 'Vibrato Delay' ;
  ADM  : $10 ; ADL  : $37 ;
  VMin : $0E ; VMax : $72 ) );
```

Konstanta **ToneMDF** ini berstruktur data array yang memuat delapan record parameter suara **tToneMod**. Record ini terdiri dari item **ADM** dan **ADL** yang memuat alamat AMDT parameter suara, **Nama** yang memuat nama dari parameter suara, **Vmax** dan **Vmin** yang memuat nilai terbesar dan terkecil setiap parameter suara, serta **ADM** dan **ADL** yang memuat nilai alamat tengah dan rendah dari parameter suara.

Konstanta **cSyxMsg** berikut ini diperlukan untuk menampung nilai tetap Sys-Ex seperti byte SOX, Roland manufacturer-ID, model-ID, command-ID RQ1, command-ID DT1, dan EOX.

```
Type
    tSetSYX             = ( SOX, Roland_ID, Model_ID,
                            RQ1, DT1, EOX );
Const
     cSyxMsg            : array[tSetSYX] of byte =
                          ($F0, $41, $42, $11, $12, $F7);
```

Struktur data **cSyxMsg** adalah array dengan dengan tipe index yang didefinisikan sedemikian rupa sehingga mudah diingat dalam pembuatan program.

Untuk menampung data nama-nama suara piranti MIDI Roland GS diperlukan variabel dinamis **GSToneName** yang dideklarasikan dengan tipe pointer **pGSToneName** yang dijabarkan sebagai berikut :

```
Const
     cMaxTone           = $7F;
     cGSFileName        = 'GS_TONE.RLD';
Type
     tGSFToneName        = text;
    pGSToneName         = ^tGSToneName;
    tGSToneName         = array[1..cMaxTone] of string[28];

Var
    GSToneName          : pGSToneName;
    FileToneName        : tGSFToneName;
```

Struktur data **GSToneName** dideklarasikan sebagai array **tGSToneName** dengan tipe elemen data *string* dengan panjang 28 karakter. Nama-nama suara piranti MIDI Roland GS disimpan dalam berkas teks “GS_TONE.RLD” yang didefinisikan dengan variabel berkas **FileToneName** dan dimuat pada saat program dijalankan.

Selain variabel dan konstanta yang dijelaskan di atas ada beberapa variabel global yang lain dalam program utama. Variabel-variabel ini akan dijelaskan bersama dengan penjelasan struktur dan prosedur program.

4.2.2. Struktur Program.

Program terdiri dari sebuah program utama, modul program untuk menangani Sound Blaster sebagai antarmuka MIDI, serta modul penunjang untuk penanganan tampilan di layar.

Struktur program utama terdiri dari judul program, bagian deklarasi, serta bagian utama program. Judul program didefinisikan seperti nama program sumbernya yaitu “Program GS_EDIT;”. Bagian deklarasi mendeklarasikan variabel dan konstanta global program serta prosedur dan fungsi yang digunakan dalam program. Program utama memakai tiga unit yang dideklarasikan pada bagian ini dengan kata cadangan *Uses*. Ketiga unit tersebut adalah *Crt*, *SBMIDI*, dan *WINOOP*. Unit Crt adalah unit standar Turbo Pascal yang berisi prosedur dan fungsi untuk menangani rutin masukan dan keluaran program.

Bagian utama program utama memanggil empat prosedur penting dalam program :

```
Begin
     ProsedurToEnter;
     InitAllWindow;
     Editor;
     ProsedurToExit;
end.
```

Prosedur **ProsedurToEnter** adalah prosedur yang harus dilaksanakan sebelum masuk ke dalam tahap penyuntingan. Cuplikan prosedur ini dijabarkan sebagai berikut :

```
Procedure ProsedurToEnter;
Begin
     If Not SBAda then
        Begin
             Writeln('Error : Sound Blaster Tidak Terpasang
...');
             Halt;
        end;
    AmbilNamaTone;
    DefChannel:=cDefChannel;
    Device_ID:=cDevice_ID;
    New(DataTones);
    InitAllParam;
    RefreshSimpan;
    Delay(1000);
```

```
        ProgramChange(DefChannel,1);
    end;
```

Pertamakali prosedur mendeteksi keberadaan Sound Blaster dengan fungsi **SBAda**, jika Sound Blaster tidak terdeteksi maka pesan kesalahan ditampilkan dan program dihentikan dengan prosedur *Halt*. Variabel GSToneName dimuati dengan nama-nama suara dari berkas “GS_TONE.RLD” dengan memanggil prosedur AmbilNamaTone. Prosedur InitAllParam menginisialisasi isi item **Modi** variabel **DataTones** dengan nilai 40h yang merupakan nilai tetap dari seluruh parameter. Sebelumnya variabel dinamik **DataTones** dialokasikan dengan *New*. **DefChannel** merupakan variabel yang menunjukkan kanal yang aktif pada layar penyunting. Variabel ini diset dengan nilai satu pada awal program. **Device_ID** merupakan variabel yang menunjukkan nomor kode piranti MIDI Roland GS yang hendak disunting, pada awal program nomor kode ini diset dengan nilai 10h yang merupakan nomor kode piranti MIDI Roland JV-30. Selanjutnya variabel ini dapat dirubah untuk nomor kode yang diinginkan.

Setelah memanggil prosedur **ProsedurToEnter**, bagian utama program memanggil prosedur **InitAllWindow** yang menginialisasi jendela yang diperlukan dalam layar tampilan penyunting.

Letak proses penyuntingan parameter suara terdapat pada prosedur **Editor** yang dipanggil oleh bagian utama program utama penyunting. Prosedur ini mempunyai sebuah loop yang di dalamnya dilakukan deteksi penekanan kunci papan ketik. Cuplikan prosedur **Editor** dijabarkan dibawah ini :

```
Procedure Editor;
Var ASCII,Scode,
    MDFN,
    YPOS,I      : byte;
Begin
MDFN:=1;
Repeat
      asm
         mov ah,0
         int 16h
         mov ASCII,al
         mov Scode,ah
       end;
Case ...
.
.
Until (SCode=$1) and (ASCII=$1B);
end;
```

Interrupt 16h fungsi 0 adalah interupsi yang melayani pendeteksian penekanan kunci papan ketik. Hasil dari penekanan kunci diletakkan pada *register* AL dan AH, isi register ini disimpan pada variabel lokal **ASCII** dan **Scode**. Jika register AL bernilai 0 berarti terjadi penekanan *extended-key* yang nilainya ada di register AH. Jika register AL tidak sama dengan 0 berarti terjadi penekanan kunci yang bisa diterjemahkan dalam nilai ASCII. Ketika kunci yang ditekan sesuai dengan yang ditentukan dalam penyuntingan, maka proses penyuntingan tertentu akan dijalankan. Proses loop ini akan berhenti apabila kunci *Escape* ditekan. Akhir proses loop juga akan mengakhiri prosedur Looping ini untuk kembali ke bagian utama program utama.

Sebelum keluar dari program, bagian utama memanggil prosedur **ProsedurToExit** yang berisi pelepasan semua variabel dinamis dari memory dan penutupan semua jendela yang telah dipakai.

Proses penyuntingan dalam prosedur **Editor** memanggil beberapa prosedur penting baik untuk

mentransfer nilai parameter suara dari piranti MIDI dengan Sys-Ex.

Prosedur **AmbilParamChan** adalah prosedur untuk mengambil nilai parameter suara dari piranti MIDI untuk setiap kanal yang ditentukan pada parameter **Chan**. Prosedur ini memanggil fungsi **AmbilParameter** untuk mengambil parameter suara dari piranti MIDI secara individual. Fungsi ini mengambil parameter tertentu di kanal tertentu yang ditentukan melalui parameter **MDFNN** dan **ChanNN**. Cuplikan fungsi **AmbilParameter** ini dijabarkan sebagai berikut :

```
Function AmbilParameter(ChanNN,MDFNN : byte):boolean;
Var C,SUM,Try,UEox : byte;
Begin
     Try:=0;
     Repeat
           If ChanNN > 10 then dec(ChanNN);
           C:=cSyxMSB+ToneMDF[MDFNN].ADM+
              ChanNN+ ToneMDF[MDFNN].ADL;
           KirimSyxRequest(cSyxMSB, ToneMDF[MDFNN].ADM +
                           ChanNN, ToneMDF[MDFNN].ADL, 1);
           AktifkanIntr;
           UEox:=0;
           Repeat
                 Delay(1);
                 Inc(UEox);
           Until (BacaBuffer(11)=cSyxMsg[EOX])
                  or (UEox>100) ;
           NonAktifkanIntr;
           SUM:=C+BacaBuffer(9)+BacaBuffer(10);
           If ChanNN >= 10 then inc(ChanNN);
           If SUM = $0 then
             DataTones^[ChanNN].Modi[MDFNN]:=BacaBuffer(9);
           SBAda;
           Delay(20);
           Inc(Try);
     Until (Try>=cTry) or (SUM = $0) or (UEox>30);
     If (Try>=CTry) or (UEox>100)
        then AmbilParameter:=False
     else AmbilParameter:=True;
end;
```

Sesuai dengan urutan cara pengambilan parameter suara, dikirimkan Sys-Ex RQ1 yang sesuai dengan parameter yang diminta. Sys-Ex RQ1 ini dikirimkan dengan prosedur **KirimSyxRequest**. Setelah Sys-Ex RQ1 ini dikirimkan maka pembacaan data Sys-Ex yang masuk segera dilakukan dengan memanggil prosedur **AktifkanIntr** yang mengaktifkan Sound Blaster untuk menerima data Sys-Ex. Data Sys-Ex yang masuk ditampung dalam sebuah penyangga (*buffer*) yang dideklarasikan di unit "SBMIDI". Setelah data Sys-Ex diakhiri dengan EOX pada buffer indeks ke-11, penerimaan data dihentikan dengan memanggil prosedur **NonAktifkanIntr**. Pengecekan keabsahan data dalam Sys-Ex DT1 yang dikirimkan oleh piranti MIDI dilakukan dengan memeriksa nilai chek sum dari nilai yang didapat. Apabila chek sum cocok maka nilai parameter yang didapat dari fungsi **BacaBuffer(9)** diletakkan pada variabel dinamis **DataTones** dan nilai fungsi **AmbilParameter** adalah TRUE. Sebaliknya apabila check-sum tidak cocok maka proses pengambilan dilakukan lagi sebanyak yang ditentukan pada konstanta **cTry**. Ketika proses pengambilan berikutnya gagal, maka nilai fungsi AmbilParameter adalah FALSE.

Nilai dari fungsi **AmbilParameter** digunakan untuk memberitahukan kesalahan dengan menampilkan pesan kesalahan pada layar penyunting.

Untuk mengirim kembali parameter yang telah disunting kepada piranti MIDI, dilakukan pemanggilan terhadap prosedur **KirimParam**. Prosedur ini mengirimkan parameter tertentu di kanal tertentu yang ditentukan pada parameter **MDFNum** dan **Chan**. Kemudian prosedur ini memanggil prosedur **KirimSyxDataSet** yang mengirimkan satu-

persatu byte Sys-Ex DT1. Cuplikan prosedur **KirimSyxDataSet** dijabarkan debagai berikut :

```
Procedure KirimSYXDataSet( AD_MSBM, AD_MSBL,
                           AD_LSB, Data : byte);
Var SUM : byte;
Begin
     SUM := AD_MSBM + AD_MSBL + AD_LSB + Data;
     SUM := $100 - SUM;
     SUM := SUM and $7F;
     TulisMIDI(cSyxMsg[SOX]);
     TulisMIDI(cSyxMsg[Roland_ID]);
     TulisMIDI(Device_ID);
     TulisMIDI(cSyxMsg[Model_ID]);
     TulisMIDI(cSyxMsg[DT1]);
     TulisMIDI(AD_MSBM);
     TulisMIDI(AD_MSBL);
     TulisMIDI(AD_LSB);
     TulisMIDI(Data);
     TulisMIDI(SUM);
     TulisMIDI(cSyxMsg[EOX]);
end;
```

Prosedur ini mengirimkan Sys-Ex DT1 dengan One Way Transfer Procedure, yaitu mengirimkan data Sys-Ex satu-persatu sampai selesai. Chek sum dihitung dari pengurangan nilai alamat ditambah data dari nilai $100 yang akan membentuk pola 7-bit tertentu sedemikian rupa sehingga apabila nilai alamat, data, dan check-sum dijumlahkan akan bernilai 0.

Untuk mengirimkan Sys-Ex RQ1 pada prosedur pengambilan parameter tadi dipanggil prosedur **KirimSyxRequest** yang dijabarkan sebagai berikut :

```
Procedure KirimSYXRequest( AD_MSBM, AD_MSBL, AD_LSB,
                           Size_LSB : byte);
Var SUM : byte;
Begin
     SUM := AD_MSBM + AD_MSBL + AD_LSB + Size_LSB;
     SUM := $100 - SUM;
     SUM := SUM and $7F;
     TulisMIDI(cSyxMsg[SOX]);
     TulisMIDI(cSyxMsg[Roland_ID]);
```

```
            TulisMIDI(Device_ID);
            TulisMIDI(cSyxMsg[Model_ID]);
            TulisMIDI(cSyxMsg[RQ1]);
            TulisMIDI(AD_MSBM);
            TulisMIDI(AD_MSBL);
            TulisMIDI(AD_LSB);
            TulisMIDI(0);
            TulisMIDI(0);
            TulisMIDI(Size_LSB);
            TulisMIDI(SUM);
            TulisMIDI(cSyxMsg[EOX]);
        end;
```

Di sini terletak perbedaan antara pengiriman Sys-Ex DT1 dengan Sys-Ex RQ1 yaitu bahwa pada Sys-Ex RQ1 alamat diikuti dengan jumlah data yang diminta, sedangkan Sys-Ex DT1 alamat diikuti dengan deretan data yang akan dikirim. Perbedaan juga terletak pada cara perhitungan chek sum dan tentu saja command-ID yang dikirim.

Pada unit SBMIDI prosedur-prosedur yang akan dipakai dideklarasikan pada bagian *Interface* dan didefinisikan pada bagian *Implementation*.

Unit SBMIDI mempunyai variabel penyangga **DataBuffer** berstrukturdata array yang terdiri dari kumpulan byte terindeks. Variabel ini akan terisi apabila pelayanan interupsi Sound Blaster untuk menerima data MIDI diaktifkan dengan prosedur **AktifkanIntr** dan ada data MIDI yang masuk. *Interrupt vector* Sound Blaster yang dibelokkan pada prosedur **UARTIntr** akan aktif apabila ada data MIDI yang masuk pada terminal MIDI In Sound Blaster. Pemanggilan prosedur ini karena adanya interrupt ini akan menaikkan indeks array DataBuffer dan mengisinya dengan data MIDI yang masuk.

Untuk mengirimkan data MIDI ke piranti MIDI disediakan prosedur **TulisMIDI** yang akan memanggil prosedur TulisSB untuk memerintahkan pengiriman data

lewat terminal MIDI Out Sound Blaster dengan perintah DSP Write MIDI (38h).

Listing program penyunting “GS_EDIT.PAS” ini dapat dilihat pada lembar lampiran bersama dengan listing unit “SBMIDI.PAS” dan “WINOOP.PAS.

4.2.3. Uji Coba Program Program Penyunting.

Uji coba program penyunting ini dilakukan dengan menggunakan konfigurasi sebagai berikut :

- Perangkat PC AT 486 SX
- Antar Muka MIDI Sound Blaster Pro 2.0 + MIDI Cable / MIDI BOX
- Piranti MIDI Roland JV-30 (Standar Roland GS)
- Perangkat pengeras suara

Sound Blaster Card ditancapkan pada salah satu *slot-bus* yang tersedia pada *motherboard* komputer. Kemudian kabel terminal MIDI Out Sound Blaster dikoneksikan ke terminal MIDI In piranti MIDI Roland JV-30, demikian juga dengan terminal MIDI In Sound Blaster dikoneksikan dengan terminal MIDI Out piranti MIDI Roland JV-30. Sedangkan perangkat pengeras suara dikoneksikan dengan keluaran sinyal suara piranti MIDI Roland JV-30.

Setelah diujicoba, secara keseluruhan program penyunting dapat berjalan dengan cukup baik. Ke-delapan parameter suara waktu dapat diambil dengan baik pada waktu pergantian kanal yang aktif. Penyuntingan dapat dilakukan dengan mudah dan dapat didengar langsung karena pengiriman Sys-Ex dilakukan segera setelah terjadi perubahan parameter suara.

Pemakaian kunci-kunci papan ketik yang tepat dan terlokasi di satu tempat pada papan ketik memudahkan proses penyuntingan. Perubahan nilai suara akibat pengiriman Sys-Ex juga segera dapat didengar dengan menekan kunci spacebar atau kunci F1, F2, dan F3.

Penanganan kesalahan dapat dikatakan telah berjalan cukup baik, hal ini terbukti ketika kabel MIDI piranti yang terhubung dilepaskan atau piranti MIDI dimatikan, maka akan tampil pesan kesalahan akibat pengambilan data dengan Sys-Ex RQ1 yang tidak sah. Tetapi ada satu kekurangan program penyunting ini yaitu ketidakberesan pentransferan data tidak dapat dideteksi waktu nilai parameter suara dirubah pada saat pengiriman Sys-Ex DT1, kesalahan akan terdeteksi waktu pengambilan parameter suara saja, yang berarti hal ini terjadi waktu pergantian kanal yang aktif pada layar penyunting.

Penanganan berkas juga berjalan dengan cukup baik. Setiap pengambilan kembali nilai parameter suara yang tersimpan dalam berkas, maka program akan secara otomatis mengirimkan semua parameter-parameter tersebut ke piranti MIDI. Kesalahan akibat kesalahan dalam penyimpanan atau pengambilan kembali berkas dapat tereliminasi.

4.2.4. Petunjuk Penggunaan Program.

Dalam prosedur **Editor** dalam program utama terdapat proses utama penyuntingan dengan mendeteksi penekanan kunci papan ketik. Kunci-kunci papan ketik yang digunakan dalam penyuntingan berikut fungsinya antara lain adalah :

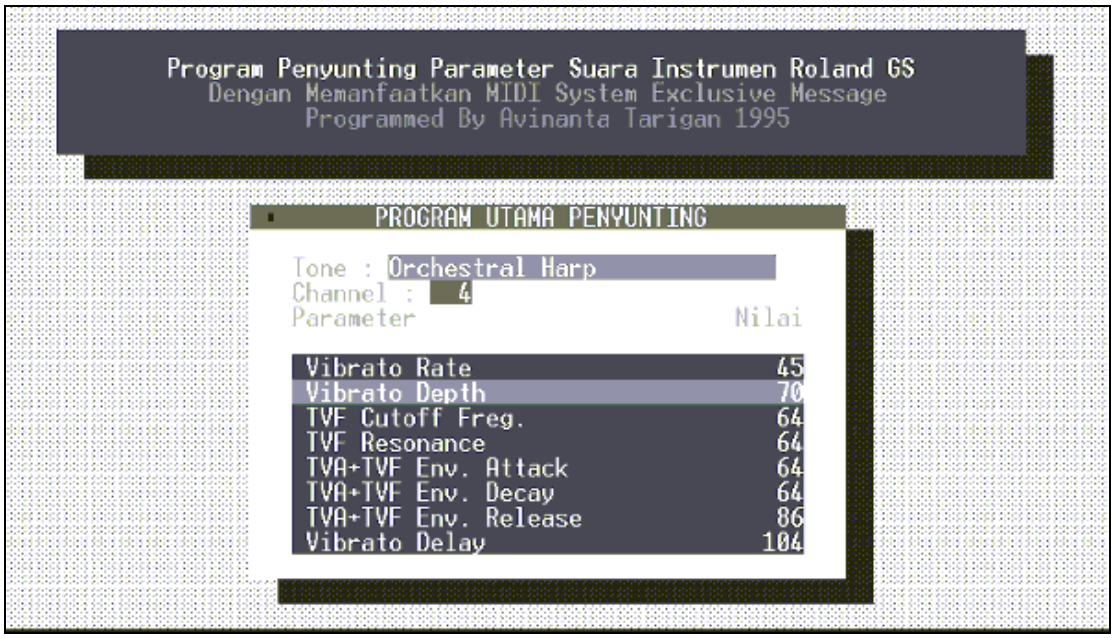
- *Arrow Left* dan *Arrow Right*. Untuk memilih saluran yang aktif pada layar penyunting. Perubahan saluran diikuti dengan pengambilan kedelapan

parameter suara dalam kanal yang aktif dari piranti MIDI.

- *Arrow Up* dan *Arrow Down* . Untuk memilih jenis parameter suara yang hendak disunting. Paramater terpilih akan di*highlight* pada layar penyunting.
- *Page Up* dan *Page Down* . Untuk memilih suara piranti MIDI yang diinginkan pada kanal yang aktif pada layar penyunting. Perubahan suara ini dilakukan dengan mengirimkan channel messages *Program Change* yang berisi informasi nomor suara dan saluran yang aktif.
- *"+"* dan *"-"* . Untuk merubah nilai parameter ter*highlight*. Penekanan kunci "+" menyebabkan nilai akan bertambah, demikian pula sebaliknya apabila kunci "-" ditekan. Setiap perubahan nilai suara oleh penekanan kunci-kunci ini, program akan mengirimkan parameter tersebut ke piranti MIDI dengan Sys-Ex DT1.
- *Alt-S* dan *Alt-L*. Untuk mengaktifkan jendela dialog berkas. Penekanan Alt-L akan memanggil prosedur Load untuk mengambil parameter suara dari berkas. Dan penekanan Alt-S akan memanggil prosedur Save untuk menyimpan parameter suara dalam berkas.
- *SpaceBar*. Untuk menyuarakan contoh suara di kanal yang aktif dengan mengirimkan channel messages note-on dan note-off.
- *F1,F2,* dan *F3*. Untuk menyuarakan contoh suara dalam satu oktaf.
- *Alt-G*. Untuk mengirimkan Sys-Ex DT1 ke piranti MIDI agar melakukan GS Reset.

- *Escape*. Untuk keluar dari loop dan program.

Tampilan layar penyunting pada program aplikasi penyunting ini didisain sedemikian rupa agar memudahkan dalam penyuntingan parameter suara. Tampilan tersebut dibuat dengan bantuan unit pendukung "WINOOP" yang berisi rutin penanganan tampilan penyunting di layar. Tampilan program penyunting digambarkan di gambar 4.1 di bawah ini.



Gambar 4.1. Tampilan program penyunting parameter suara piranti MIDI Roland GS.

Instalasi program penyunting ini tidaklah rumit, karena hanya terdiri dari program "GS_EDIT.EXE" dan sebuah berkas penyimpan nama-nama suara "GS_TONE.RLD". Kedua berkas tersebut harus diletakkan dalam satu

direktori bersama dengan berkas-berkas hasil penyuntingan, misalnya dalam direktori “C:\GS_EDITOR\”

Untuk menjalankan program penyunting, hanya diketikkan “GS_EDIT” pada *prompt* sistem operasi, selanjutnya pengguna tinggal melakukan penyuntingan dengan melakukan penekanan kunci-kunci yang sudah dijabarkan di atas.

Beberapa pesan kesalahan akan muncul bila terjadi ketidakberesan. Pesan kesalahan tersebut adalah sebagai berikut :

- “Sound Blaster Tidak Terpasang”. Pesan ini berarti bahwa kartu Sound Blaster tidak terdeteksi oleh program.
- “File GS_TONE.RLD tidak ada”. Pesan ini berarti bahwa program tidak menemukan berkas “GS_TONE.RLD”
- “File GS_TONE.RLD tidak benar”. Program menemukan ketidakabsahan dalam berkas “GS_TONE.RLD”.
- “Data MIDI Error .... “. Program menemukan bahwa ada ketidakberesan saat proses transfer data dari piranti MIDI. Pesan ini bisa disebabkan oleh kesalahan pemeriksaan check sum atau tidak ada data yang masuk di terminal MIDI In Sound Blaster.
- “File (NAMA FILE) tidak ada ...”. Pesan ini berarti program tidak menemukan berkas parameter yang hendak diambil.
- “Sudah Ada ... Tumpangi ? “. Pesan ini berarti program menemukan nama berkas yang sama dengan nama berkas yang hendak disimpan. Program

meminta pernyataan apakah berkas tersebut akan ditumpangi dengan berkas yang hendak disimpan.

BAB 5
PENUTUP

Berdasarkan uraian dari Bab 1 sampai Bab 4 yang mengetengahkan pemanfaatan Sys-Ex untuk menyunting parameter suara piranti MIDI Roland GS dan diwujudkan dalam bentuk program aplikasi penyunting, maka dapat diambil kesimpulan dan saran sebagai berikut :

5.1. Kesimpulan

System Exclusive Message merupakan salah satu MIDI Message yang dapat mengakses parameter internal dan kontrol internal dari suatu piranti MIDI. Dengan memanfaatkan Sys-Ex dapat dibuat banyak program aplikasi, diantaranya adalah program penyunting parameter suara yang diketengahkan dalam penulisan ilmiah ini.

Asosiasi MIDI internasional telah membakukan sebuah standar MIDI yang dijabarkan dalam MIDI Spesification 1.0. Sys-Ex mendapat perhatian khusus di dalamnya, karena setiap perusahaan diberi kebebasan untuk mengembangkan format Sys-Ex dan cara pentransferannya. Hal ini perlu dilakukan karena adanya perbedaan teknologi dan struktur sistem perangkat keras yang diterapkan oleh masing-masing perusahaan terhadap piranti MIDI produksinya. Bahkan untuk setiap jenis piranti MIDI, satu perusahaan juga menerapkan format Sys-Ex yang berbeda.

Perusahaan piranti MIDI Roland Corporation telah mengembangkan standar teknologi untuk diterapkan pada satu kelompok jenis piranti MIDI produksinya. Standar teknologi tersebut dinamakan Roland GS Sound Source atau

Roland GS. Standar teknologi ini juga membakukan MIDI Message yang termasuk didalamnya adalah Sys-Ex.

Sys-Ex merupakan MIDI Message yang akan dikirimkan ke seluruh bagian dari sistem piranti MIDI. Agar data yang dikandung oleh Sys-Ex sampai ke lokasi spesifik yang dituju, Roland GS mengembangkan suatu teknik pengalamatan dan pentransferan dalam Sys-Ex Roland GS yang dinamakan Address Mapped Data Transfer. Dalam penulisan ini digunakan prosedur transfer Sys-Ex One-Way Transfer Procedure. Prosedur ini membagi dua tipe Sys-Ex yaitu RQ1 dan DT1. Sys-Ex RQ1 digunakan untuk memerintahkan piranti MIDI agar mengirimkan parameter yang diminta dalam RQ1. Sedangkan Sys-Ex DT1 merupakan tipe Sys-Ex yang memuat data yang sebenarnya. Teknik pengalamatan ini terdiri dari tiga byte Address yang menunjukkan lokasi data, dan tiga byte Size yang menunjukkan besar data.

Dengan Sys-Ex Roland GS yang teknik pengalamatan Address Mapped Data Transfer ini, dibuat program penyunting parameter suara. Dasar program penyunting ini adalah mengambil parameter suara dari Roland GS, menampilkannya di layar penyunting, dan mengirimkannya kembali apabila terjadi penyuntingan nilai parameter tersebut. Ada delapan jenis parameter suara yang berlokasi di setiap kanal, jadi penyuntingan dilakukan pada setiap kanal, hal ini menghindari proses penyuntingan yang rumit.

Program penyunting didisain agar pengguna dapat menyunting dengan mudah, pendisainan kunci-kunci papan ketik yang digunakan dalam proses penyuntingan serta tampilan layar penyunting menjadi salah satu faktor yang diperhatikan dalam pendisainan user interface program.

Ada satu kelemahan dalam program penyunting parameter suara ini, yaitu penanganan ketidakberesan waktu terjadi proses pengiriman Sys-Ex kepada piranti MIDI. One-Way Transfer prosedur tidak memungkinkan program untuk mengetahui apakah koneksi MIDI ke piranti MIDI sudah benar atau piranti MIDI sudah siap menerima Sys-Ex yang dikirimkan. Tetapi ketidakberesan tersebut dapat terdeteksi waktu program menerima Sys-Ex DT1 dari piranti MIDI, karena dapat dideteksi keabsahan data yang dikirimkan dengan memperhitungkan chek sum.

Setelah di uji coba, program peyunting ini telah berhasil untuk menyunting parameter suara piranti MIDI Roland GS. Dengan memanfaatkan Sys-Ex ternyata dapat dibuat program penyunting parameter suara yang merupakan tujuan dari penulisan ilmiah ini.

5.2. Saran

Seperti yang telah disebutkan di atas bahwa salah satu kelemahan program adalah tidak dapat mendeteksi kesiapan koneksi MIDI waktu pengiriman parameter suara tersunting dengan Sys-Ex DT1. Mungkin dengan teknik pentransferan yang lebih baik atau dengan menggunakan teknik-teknik lain, kelemahan tersebut dapat tereliminasi. Misalnya dengan mengambil kembali parameter suara yang telah dikirimkan dan dilakukan pencocokan nilainya.

Program penyunting ini memakai *Text-based User Interface*, alangkah baiknya apabila digunakan *GUI* atau *Graphic User Interface* dengan menggunakan *mouse* sebagai alat masukan dalam proses penyuntingan. Dengan melakukan operasi *"click and drag"* mouse pada grafik yang dibentuk dari parameter-parameter suara, proses penyuntingan

parameter suara dapat dilakukan dengan mudah dan interaktif. Karena selain dapat mendengar suara hasil suntingan, pengguna juga dapat melihat secara visual bagaimana parameter-parameter tersebut membentuk suara yang disuntingnya.

DAFTAR PUSTAKA

1. Brian Heywood et al. The PC Music Handbook. Brook Street, Tonbridge : PC Publishing, 1991.
2. R. A. Penfold. Advanced MIDI User's Guide. Brook Street, Tonbridge : PC Publising, 1991.
3. Roland. Roland JV-30 OWNER'S MANUAL. Tokyo, Japan : Roland Corporation, 1992.
4. Arianto Widyanto et al. Belajar Mikroprosesor-Mikrokontroler. Jakarta : PT Elex Media Komputindo, 1994.
5. Creative. Sound Blaster : The Official Book. Singapore : Creative Labs Inc., 1992.
6. Borland International. Turbo Pascal 6.0 Library Reference. Scotts Valley, CA : Borland International, Inc., 1990.