トランジスタ技術SPECIAL増刊

Java & AKI-80

パソコンでつくるマイコン組み込みMIDI関連機器の製作

長嶋 洋一 著

CQ出版社

トランジスタ技術スペシャル　好評発売中

トランジスタ技術 SPECIAL No.59

エレクトロニクスの基礎と実用技術を濃縮したフィールド・ワーク・マガジン

B5判/176頁/定価1,840円

特集 RISCライクなZ80互換プロセッサKC80を詳解する

新世代Z80CPUで学ぶマイコン入門

8ビット・ワンボード・コンピュータに採用されているTMPZ80C015は，1970年代にZilog社が開発したZ80CPUとZ80周辺LSIを集積した使いやすいLSIです．しかし，8ビット処理のCPUであるため，素人目には処理スピードが遅いと思われることがありました．また昨年には，製造中止するのではという観測も流れ，ほかのCPUを物色する向きも現れてきました．ここで，H8やPIC，TLCS90，TLCS900などにまじって川崎製鉄のKC80の採用が検討されました．KC80はASICのコアとして開発されたRISCライクなCPUをZ80機械語コンパチブルにして動作速度を4倍にしたものです．現在，ボード・コンピュータに採用しているメーカも多々あります．今回はいまだ解説本などが出ていないKC80をとりあげ，その動作から使い方までを紹介します．

① マイコン・システムのあらまし，②アセンブラ・プログラミング，③Z80マイクロプロセッサとKC80アーキテクチャ，④Z80の割り込み機構とKL5C80A16の割り込みコントローラ，⑤カードCPU基板の設計とZビジョン・モニタの移植，⑥KL5C80A16のパラレル・インターフェース，⑦KL5C80A16のシリアル・インターフェース，⑧KL5C80A16のタイマ/カウンタ・インターフェース

既刊好評発売中　●B5判●定価：1〜33 1,540円　34〜45 1,600円　46〜49 1,690円　50〜57 1,800円　58以降1,840円

1 個別半導体素子活用のすべて
4 C-MOS標準ロジックIC活用マニュアル
5 画像処理回路技術のすべて
6 Z80ソフト&ハードのすべて
8 データ通信技術のすべて
9 パソコン周辺機器インターフェース詳解
11 フロッピ・ディスク・インターフェースのすべて
13 シミュレータによる電子回路理論入門
14 技術者のためのCプログラミング入門
16 A-D/D-A変換回路技術のすべて
17 OPアンプによる回路設計入門
22 ディジタル回路のノイズ対策技術のすべて
23 回路デザイナのためのPLD最新活用術
25 最新マイコン・メモリ・システム設計法
27 ハードディスクとSCSI活用技術のすべて
28 最新・電源回路設計技術のすべて
29 マイコン独習Z80完全マニュアル
31 基礎からのビデオ信号処理技術
32 実用電子回路設計マニュアル
33 オプト・デバイス応用回路の設計・製作
34 つくるICエレクトロニクス
35 C言語による回路シミュレータの製作
36 基礎からの電子回路設計ノート
37 実用電子回路設計マニュアルII
38 Z80システム設計完全マニュアル
39 A-Dコンバータの選び方・使い方のすべて
40 電子回路部品の活用ノウハウ
41 実験で学ぶOPアンプのすべて
42 高速ディジタル回路の測定とトラブル解析
43 Cによるマイコン制御プログラミング
44 フィルタの設計と使い方
45 PC98シリーズのハードとソフト
46 アナログ機能ICとその使い方
47 高周波システム&回路設計
48 作れば解るCPU
49 徹底解説 Z80マイコンのすべて
50 フレッシャーズのための電子工学講座
51 データ通信技術基礎講座
52 ビデオ信号処理の徹底研究
53 パソコンによる計測・制御入門
54 実践パワー・エレクトロニクス入門
55 作ってわかる電子回路製作入門
56 電子回路シミュレータ活用マニュアル
57 最新 スイッチング電源技術のすべて
58 基本 C-MOS標準ロジック活用マスタ

CQ出版社　〒170 東京都豊島区巣鴨1-14-2　販売部 ☎(03)5395-2141　振替 00100-7-10665

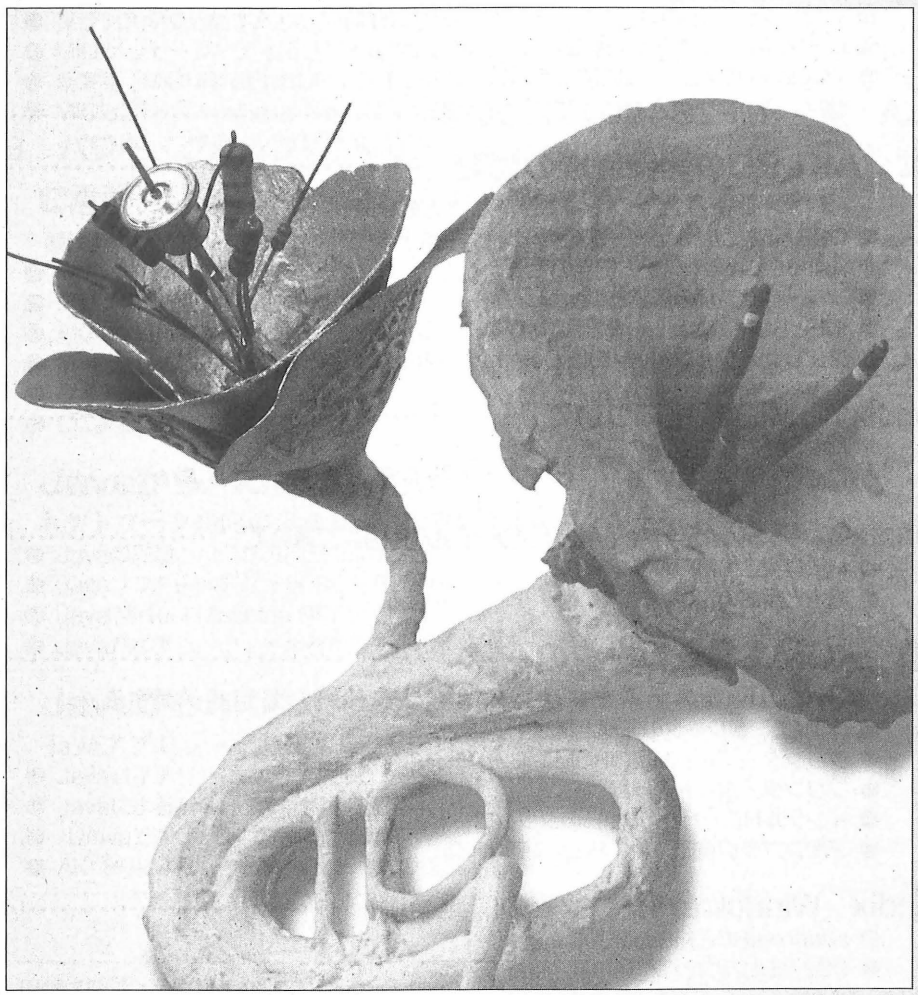
※定価は消費税込みの価格です

CQ出版社のホームページのアドレス　http://www.cqpub.co.jp

Java & AKI-80

パソコンでつくるマイコン組み込みMIDI関連機器の製作

長嶋 洋一 著



CQ出版社

トランジスタ技術SPECIAL 増刊
パソコンでつくるマイコン組み込みMIDI関連機器の製作

Java & AKI-80

長嶋 洋一 著

contents

表紙・扉デザイン/アイドマ・スタジオ　表紙イラスト/伊部 喜久雄　本文イラスト/神崎 真理子

第1章　なぜJava & AKI-80か

21世紀のソフト技術+20世紀後半のエレクトロニクス技術

Javaの登場

コンピュータが登場したころは，まだ夢のような先のことと思われていた「21世紀」まで，あとわずかとなりました．間違いなく21世紀をリードする情報テクノロジーは，その主役がハードウェアからソフトウェアに，さらにネットワークへと変貌しています．そして1996年から1997年にかけて，コンピュータ・エレクトロニクス技術のもっともホットな話題といえばインターネット，さらに端的にはJava（ジャバ）がその焦点となっています．

これまでにも数多くの「新しいコンピュータ言語」が登場しては消えていったにもかかわらず，Javaがこれほどまでに注目され，急速に普及しているのはなぜでしょうか．従来の出版物やパッケージ・ソフト（Windows95の販売手法がまさにこの方法）による「古典的」な普及手法にくらべて，インターネットによる世界同時的な情報供給手法というのも理由の一つではありますが，筆者はなにより，Java自身のもつユニークな可能性を指摘したいと思います．

これまでのすべてのコンピュータ言語，さらには広く言えばコンピュータ・ソフトウェアというのは，あくまでコンピュータ本体というハードウェアの存在を前提にしたものでした．「コンピュータ，ソフトなければただの箱」という言葉も，この前提を基本としています．ところがJavaは，初めてソフトがハードを忘れて独自の世界を歩み始めるという記念すべき新段階をもたらしたのです．

Javaは特定のコンピュータ・ハードウェアを想定せず，Virtual Machine（バーチャルマシン）という仮想的なプラットフォーム上で稼動します．WindowsでもMacでもUnixでも，ソース・プログラムをコンパイルし直すのでなく，どこかでコンパイルされたバイナリ・コードが，そのまま走るというのは驚異的なことでしょう．これは，今後どんどん新しいコンピュータ・ハードウェアが登場してくる時代にとって，「ソフトウェアの陳腐化」「使っていたソフトの走るハードが消える」などという現在の問題点を画期的に改善してくれるものです．

また，Javaは言語仕様そのものにインターネットを組み込んでいるため，「素材だけを自分で用意して，プログラムはインターネットから呼び込んで使う」という新しいソフトウェア利用形態を実現しました．これまではユーザが自分でプログラミングしなければならなかったのにくらべれば，この点でも画期的な変化が訪れていることになります．21世紀のプログラミングでは，完全に自分ですべてのソース・コードを開発する必要がなくなるかもしれません．

電子技術の空洞化

このように新しいコンピューティング・パラダイムを提供するJavaは歓迎すべきものですが，エンジニアリングの世界では大きな問題を引き起こす懸念があります．

この傾向はすでに大学理工系などで始まっているのですが，技術者・研究者のハードウェア離れという深刻な問題があります．いくらJavaがハードを選ばないといっても，とにかく何らかのハードウェアが存在しない限り，絶対になにもしてくれません．私たちの周囲の現実世界とコンピュータ・ソフトウェアとの間のインターフェースとして，コンピュータというハードウェア，あるいはコンピュータと外界とのやりとりを行うためのシステム，たとえばエレクトロニクス技術やメカトロニクス技術というのは普遍的に必要なのです．Javaプログラムがいくら「入力しろ」「表示しろ」「操作しろ」と命令するように記述されていても，手足となるハードウェアが存在しなければ，まったく意味をもたないのです．

そして，システムが次第に巨大で複雑化し，さらに多数がネットワークによって協調的に機能するようになると，ハードウェアの技術，あるいはソフトウェアと結びついたハードウェアとしてのシステムの理解が不可欠になってきます．たとえば，地球環境問題に関

Java　Sun Microsystems社が提唱したインターネット時代の新しいプログラミング言語．世界中で採用されてきている．

して環境汚染の計測をしようとした場合，いくら優れた計測・分析ソフトウェアをJavaで開発しても，それだけでは現状のレベルを超えることはできません．新しい計測センサ，センサ処理システムのハードウェア，多点観測データの計測インターフェースの開発，インターネットによる無人計測・集計システムの構築などがあって，はじめて画期的な進展があるのです．ここでは，「私はソフトウェアだけがわかります」という姿勢では，あまり役に立たないのは明らかです．

もちろん，20世紀後半のコンピュータ・エレクトロニクス技術の急速な発展は，回路，デバイス，ソフト，システムなど広範な技術領域を生み出し，いずれかの分野で専門家になるだけでも精一杯という状況となりました．若いエンジニアがこの世界に飛び込んでまず戸惑うのは，アナログやディジタルの回路技術，CPU技術(いわゆるマイコン技術)，周辺LSI，メモリ，DSP，インターフェース，ネットワーク，ソフトウェアなどのあまりにも多くのマスターすべきことがらです．

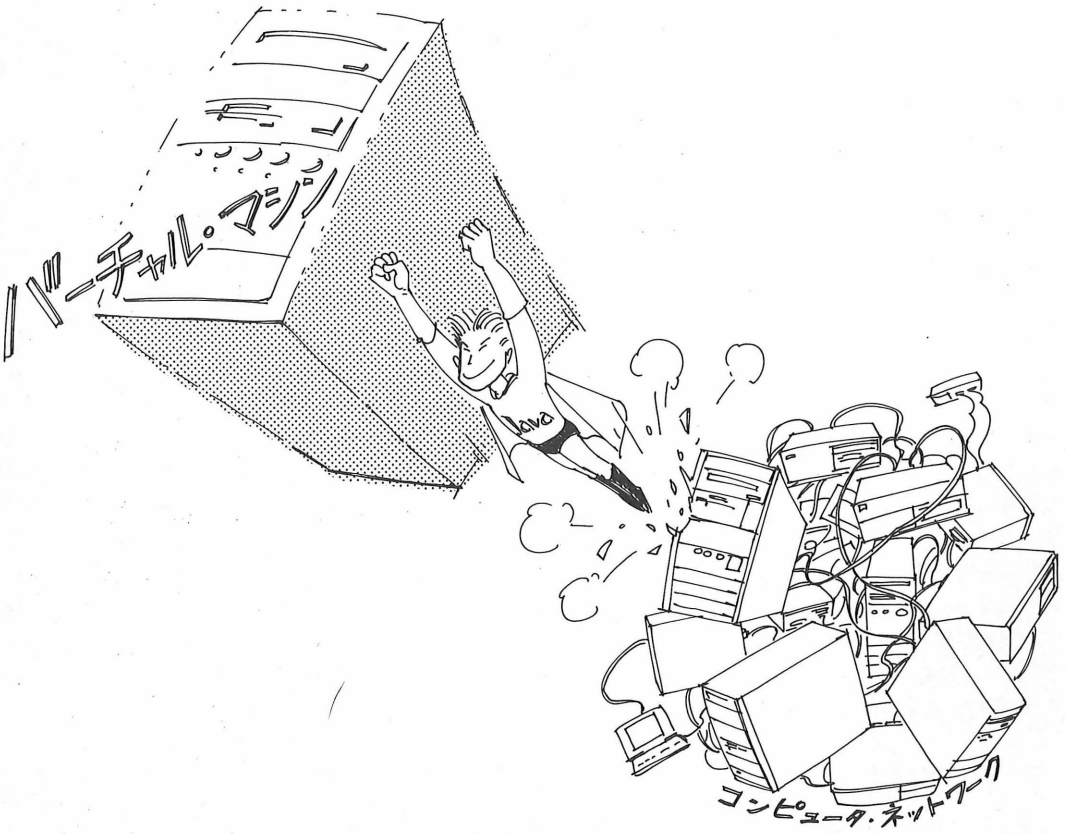
しかし，事態は少しずつ進展しています．昔のエンジニアが時間をかけて修得しなければならなかった技術項目のかなりの部分が，いろいろな規模のIC，LSI，モジュールなどの汎用部品としてブラック・ボックス化され，ノウハウを必要とせずに活用できるようになりました．

たとえば，かつては新しいマイコン・システムを開発しようとすると，CPU周辺から設計して所望の仕様を実現するためには，かなりの回路技術や経験を必要としていました．しかし現在であれば，部品として汎用のCPUカード・マイコンを使い，周辺機能に特化した専用LSIを組み合わせるだけで，システム開発の主力をソフトウェアに注げるようになっています．この事態を嘆くベテラン技術者もいますが，ある程度はやむをえない，むしろ積極的に採用すべきことだろうと筆者は思います．

本書の目標

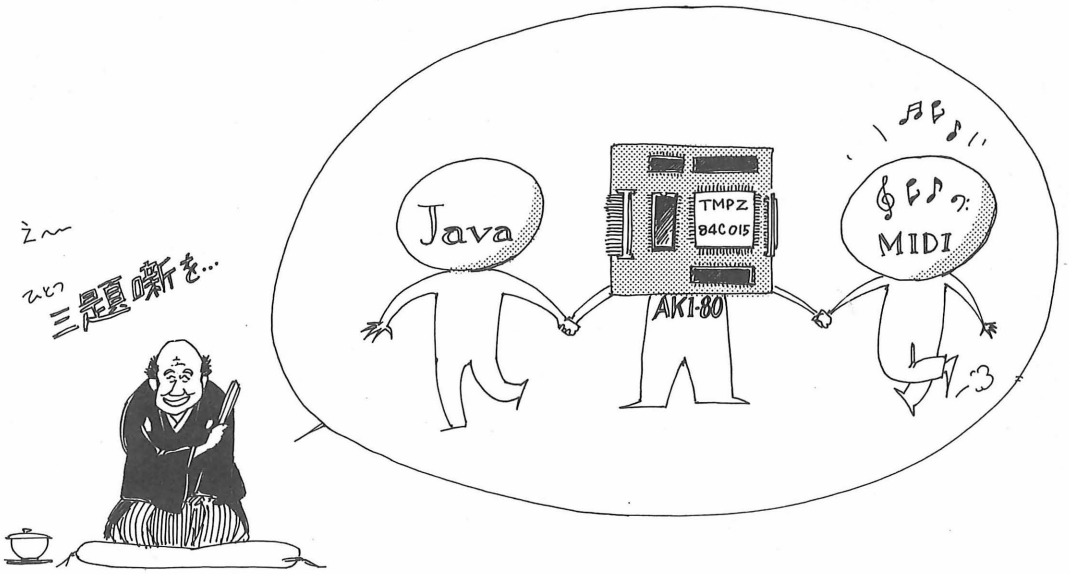
さて，そこでようやく本書の目標の項に到達しました．筆者は持論として，「21世紀型のエンジニアはハ

〈ソフトがハードを忘れて独自の世界を歩み始める〉



Virtual Machine　「Java言語によるプログラムが走る」と想定された仮想的なコンピュータのこと．特定のプラットフォームに限定されないJavaならではの発想．

〈本書の目標〉



ードもソフトも両方をカバーできなければならない」と主張してきました(＊脚注)．そしてこれはJavaの登場してきた現在，ますます必要になっているものだと思います．1997年版として言い換えるなら，「これからの技術者は，ハード，ソフト，システム，ネットワークをすべて理解して，自分でゼロから構築できなければならない」とでもしておきましょう．これはなにも，たった一人でいつもシステムを設計するということではありません．現代のシステム開発というのは，多人数によるプロジェクト・ワークとなっています．しかし，実際にはその一部を担当するとしても，上記のような視点をもっているのといないのとでは，まるで見通しが違ってくるのです．

本書の対象となるのは，「ある程度，ソフトのわかってきた人」ということを前提にしています．まったくエンジニアリングの初心者は，まずは基礎的な勉強をしていきましょう．ただし，本書は「読み物」として読み流しても面白いように考慮していますので，文系寄りで興味のある方も歓迎です．ある程度ははんだ付けの経験もあったほうがよいのですが，最近は大学の電気電子系を卒業したのに，はんだごてを握ったことのない新人も多い時代ですから，これは本書を読み進めるのと一緒に練習していきましょう．

はんだ付けの技術(技能)は，決してむだでないどころか，エンジニアとして技術領域を広げるためには必須のスキルなのです．たとえば，ケーブルの断線を突きとめて修理するという簡単なこともできないエンジニアなんて，誰だって信頼できませんから．

具体的には，もっともシンプルなマイコン・ボードとして，東京・秋葉原の(有)秋月電子通商が販売している「AKI-80(アキ・エイティ)」を対象とします．詳しい内容は追って紹介しますが，これは現代のマイコン技術の一つの結晶ともいえるものです．そして，もう一方の対象としてJavaを取り上げます．1980年代の主役ですでに殿堂入りしたといわれている「Z80 CPU」の流れのAKI-80と，1995年に登場したJavaとが，はたしてどのように結びついてくるのか，これが本書の最大のポイントなのです．

そして，「JavaでAKI-80」という目標に向けて，筆者は本書のためにほとんどすべての材料を新規に用意しました．情報が不足していれば自分で解析し，その解析ツールも自分で手作りしました．開発ツールがなければ自分でハードもソフトも作りました．「道具がなければ道具を作る」というこの姿勢は，それ自体が自分を向上させてくれるスキルアップの秘訣なのです．

前半は準備作業のような内容で，ややスローペースで要素技術の確認をしていきます．しかし，この部分は，読者それぞれのレベルに応じた技術的なポイントの確認，あるいは勉強の材料となるよう配慮しています．初心者は一つ一つ実験しながらハードとソフトを

(*)筆者は「マイコン技術者スキルアップ事典」(CQ出版社，1992年)で，これからプロの技術者を目指す「エンジニアの卵」を元気にしようと，コンピュータ・エレクトロニクス技術全般を概観しました．そして続編「プロ電子技術者のコモンセンス」(CQ出版社，1993年)では，入社2～3年の中堅エンジニアの視野を広げる200項目を紹介しました．本書は，ある意味でこの路線の延長線上にあるものです．

ソース・コード　プログラミング言語で書かれた元のプログラムのこと．ソース・コードをアセンブラやコンパイラなどの変換ツールによって，コンピュータが実際に使用するオブジェクト・コードに変換する．

体験的に修得し，ベテランの方は一瞥するだけで読み流しながらも，ポイントの整理として活用していただければと思います．

筆者はAKI-80を提供している秋月電子通商(「信越電気商会」という名前の頃の小学生時代からのファンです)の姿勢に共感して，基本的にアマチュアイズムを前面に押し出しています．プロ用の高価な道具や部品は使わず，アマチュアのホビーとして可能なところから実験していける題材ばかりです．具体的な製作テーマには，「技術者・研究者」とともに筆者のもう一つの顔である「音楽家・作曲家」として「MIDI(Musical Instruments Digital Interface)」を利用した音楽関連システムが並んでいますが，応用分野としてはどこにでも展開できるものばかりです．

この機会に本書をきっかけとしてComputer Musicの世界に興味を持っていただければ，これもうれしいことです．

本書を読み進めることで，「これまでソフトウェアを作ったことがある」程度だった人は，コンピュータ技術のハードウェア的な理解，アセンブラやコンパイラとコンピュータ言語の本質的な理解，そしてマイコン・システムの開発/実現というものがマスターできることでしょう．さらに，たんにインターネットのWebホームページにサンプル改編のJavaアプレット・プログラムを載せるだけの技術でない，本質的なJavaの理解をもったエンジニアとなることでしょう．これは間違いなく21世紀の技術につながる重要なスキルであると思います．筆者とともに，大きな実りに向けて，楽しんで頑張っていきましょう(^_^)．

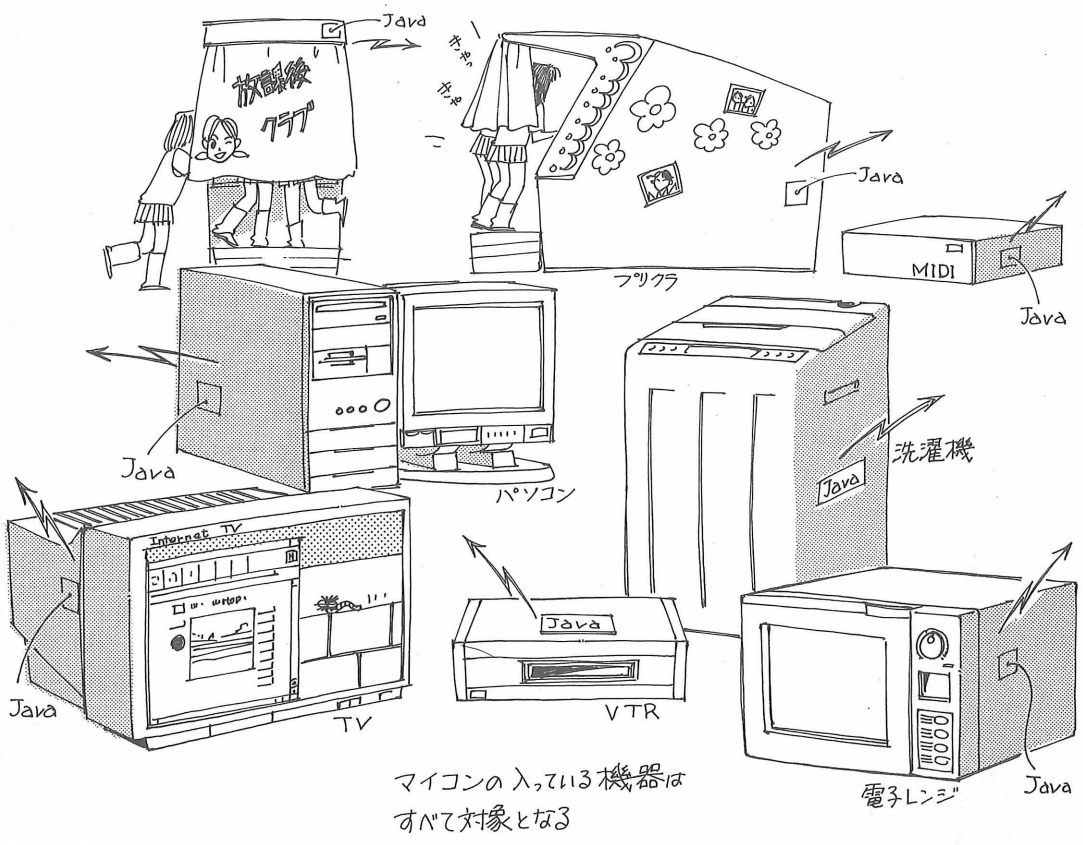
リストの入手方法について

本書の中に登場するすべてのリストについてはインターネットのホームページで誰でも入手できるようになっています．それぞれのリストの一覧メニューとなっているページのURLは，筆者のホームページの中の，

http://www.kobe-yamate.ac.jp/~nagasm/java-aki/

にあります．ここからリスト名のところをクリックすると，画面にそれぞれのリストが表示されるので，ブラウザの「名前をつけて保存」というメニューで自分のパソコンにダウンロードしてください．

〈Javaは組み込み機器開発用言語としてつくられた〉



Javaアプレット・プログラム　Javaによって作られるプログラムのうち，Netscapeなどの Webブラウザ内でのみ動作するように指定されたプログラムをアプレットという．

第2章　Javaへの旅立ちの準備

中古PC98＋MS-DOS,AKI-80,アセンブラ,
Cコンパイラ，テキスト・エディタ，…

ハードとソフトの環境と方針

AKI-80というカード・マイコンを中心にハードウェアのシステムを作り，ここにアセンブラ，Cコンパイラによって開発したソフトウェアを載せ，さらにはJavaを用いてAKI-80ソフトを作ってみよう，というのが本書の目標です．

そこで，まずは開発に必要なハードとソフトの環境を整備し，さらにシステム開発に必要なツールのうち，一部の重要なものは技術的な実験を兼ねて自作して揃えていくことにします．

本書を執筆している1996年なかばの時点で，世の中はWindows95パソコンでなければパソコンにあらずという状況になっていました．しかし，現在でもテキスト・ベースのコンピューティングは「開発」の主役だし，AKI-80のようなコンパクトでプリミティブなシステムの開発に大げさなパソコンは似合いません．そこで筆者は，あえて「古典的MS-DOS環境での開発」を採用することにしました．これは，ハードの基本的なところからシステムのすべてを理解していくという本書の目的にとって重要なことであり，とくに教育的にはベストのツールだからです．もちろん，手元にWindowsパソコンしかないという読者のための対応方法については，この章の最後の節にまとめています．Windowsの中の「**MS-DOS窓**」を使っても同様に開発できるのでご安心ください．

また，本書の読者としては「ある程度のソフトを書いた経験がある」というレベルを前提としているので，MS-DOSのバッチ・ファイル，テキスト・エディタ，C言語コンパイラなどはすでに使ったことがあるものとしています．

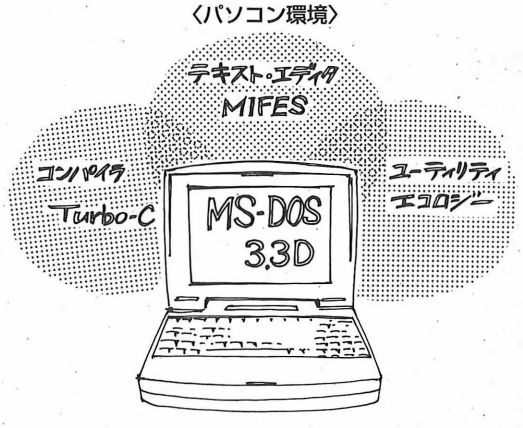
ただし，AKI-80のCPUプログラムを開発するためのツール(アセンブラとコンパイラ)については，まったく初めてという状況を想定しています．このために，プロ仕様の高価なツールでなく，「秋月電子でそれぞれ2,500円で入手したアセンブラとCコンパイラ」だけを使うという過酷な条件を課しています．プロの使うツールはパソコン版でも数十万円するのですが，アマチュア価格の道具だけでどこまで開発できるかというのも面白いテーマでしょう．

なお，Javaについてはインターネットにアクセスしなければ何も調べることができませんから，本書ではAKI-80開発用のパソコンとはまったく分離して考えることにします．最近ではインターネット・カフェや公共の図書館を含めて，インターネットへのアクセスはどこでも可能だからです．

ちなみに，Javaに関連して筆者が本書のために調査，解析，開発した環境は，大学や研究所と自宅にあるSilicon Graphics社のIndyとApple社のMacintoshがほとんどです．しかし，これらの環境で開発したJavaのバイナリ・プログラムをAKI-80開発用のパソコンにフロッピ経由やネットワーク経由で移してしまえば，あとは「素のMS-DOS」だけでも可能な開発の世界となるのです．

パソコン環境と中古98

まず最初に必要となるのが，AKI-80開発用のパソコン環境です．日常的にはWindows環境で仕事をしている場合でも，トラブルに遭遇したときやシステムをカスタマイズしていくときには，Windowsマシンを「た



MS-DOS窓　Windows95の環境で「MS-DOSプロンプト」と呼ばれるツールのこと．従来のMS-DOSアプリケーションはこのウインドウの中で動くことができる(できないものもある)．

だのMS-DOSパソコン」として眺める視点は重要です．また，本書の流れにしたがってAKI-80応用システムを製作していくと，開発用パソコンはそのまま専用マシンとして使用したくなってきます．

そこで，開発パソコンとしては「中古のPC-9801」という，現在では笑える環境を採用することにしました．

最近では，東京・秋葉原でも大阪・日本橋でも，中古パソコン・ショップのフロア構成が変化してきました．「Windows95マシン」「Mac」「PC互換機（Win3.1）」「その他」という分類が大部分で，かつて巨大勢力だったPC98シリーズは富士通のFM-Townsやシャープのx68000と並んで，「その他」になってしまいました．

しかし，これは逆に考えると，アマチュアにとって大きなチャンスです．じつは筆者も入手して使っているのですが，まだまだ十分に仕事をしてくれる強力なPC98が数千円，CRT一体型でも1万円台，ノートでも3万円で入手できます．分解しても壊してもかまわない格好のオモチャといえるでしょう．

それでは，具体的に筆者が本書のために用意した環境を紹介しておきます．基本的には，これらの環境は揃っていることが前提となるので，必要なものは秋葉原などでなるべく安く揃えましょう．

▶ **パソコン**：PC9801N(ノート・タイプ)

別にデスクトップでも，PC互換機でもかまいません．後述するように，プリンタ・ポートを解析して使うので，どんなパソコンでも大丈夫です．**ROMエミュレータ**としてプリンタ・ケーブルを切断して使うので，これも中古で入手します．メモリは640 Kバイトで十分です．

▶ **OS**：MS-DOS 3.3D

たまたま筆者のパソコンはこの環境であるというだけで，どのバージョンでもかまいません．エディタとCコンパイラが走れば十分です．

▶ **テキスト・エディタ**：MIFES

VZエディタでもEDLINでもなんでもかまいません．

▶ **Cコンパイラ**：Turbo-C

筆者はこのほかにも，MS-CとLSI-Cを使っています．どれでもかまいませんが，ポート出力の関数名だけ，該当するコンパイラに準拠して書き換えてください．

▶ **その他ユーティリティ：SYMDEB**，エコロジー，DUMPなど

解析の時にSYMDEBを使いましたが，何かほかのデバッガでもかまいませんし，PC98であればすでに情報があるので解析は不要です．エコロジーはデータを見たりファイル名を変えたりするのに便利ですが，フリーウェアのFDでも十分です．

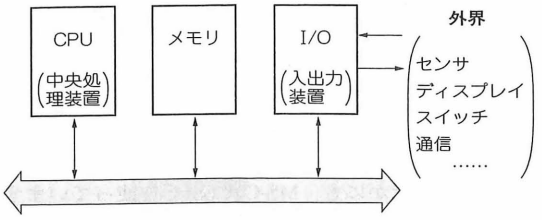
カード・マイコン「AKI-80」

さて，それではいよいよ「AKI-80」(**写真2-1**)について紹介します．これは秋月電子の「キット」となっているので，システムの中枢として使用するためには，まずはんだ付けをして製作しなければなりません．この作業がいわゆる「ソフト屋さん」には壁となるようです．AKI-80にはRAM容量に応じて「ゴールド」「シルバー」という2種類の価格のものがありますが，本書では廉価版のシルバーキットを対象としています．初めての人は，とりあえず通信販売でAKI-80のシルバ

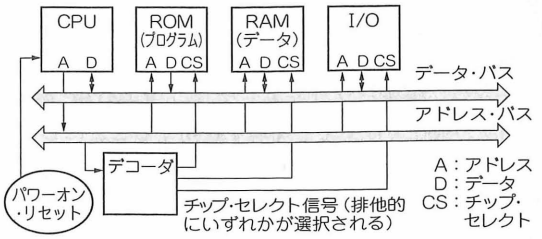
〈写真2-1〉
AKI-80のキット内容
(写真はゴールド・キット)

ただのMS-DOSパソコン　WindowsでなくMS-DOSのみで動作させているパソコン．

〈図2-1〉マイコン・システムの基本



〈図2-2〉組み込みマイコン・システムの基本

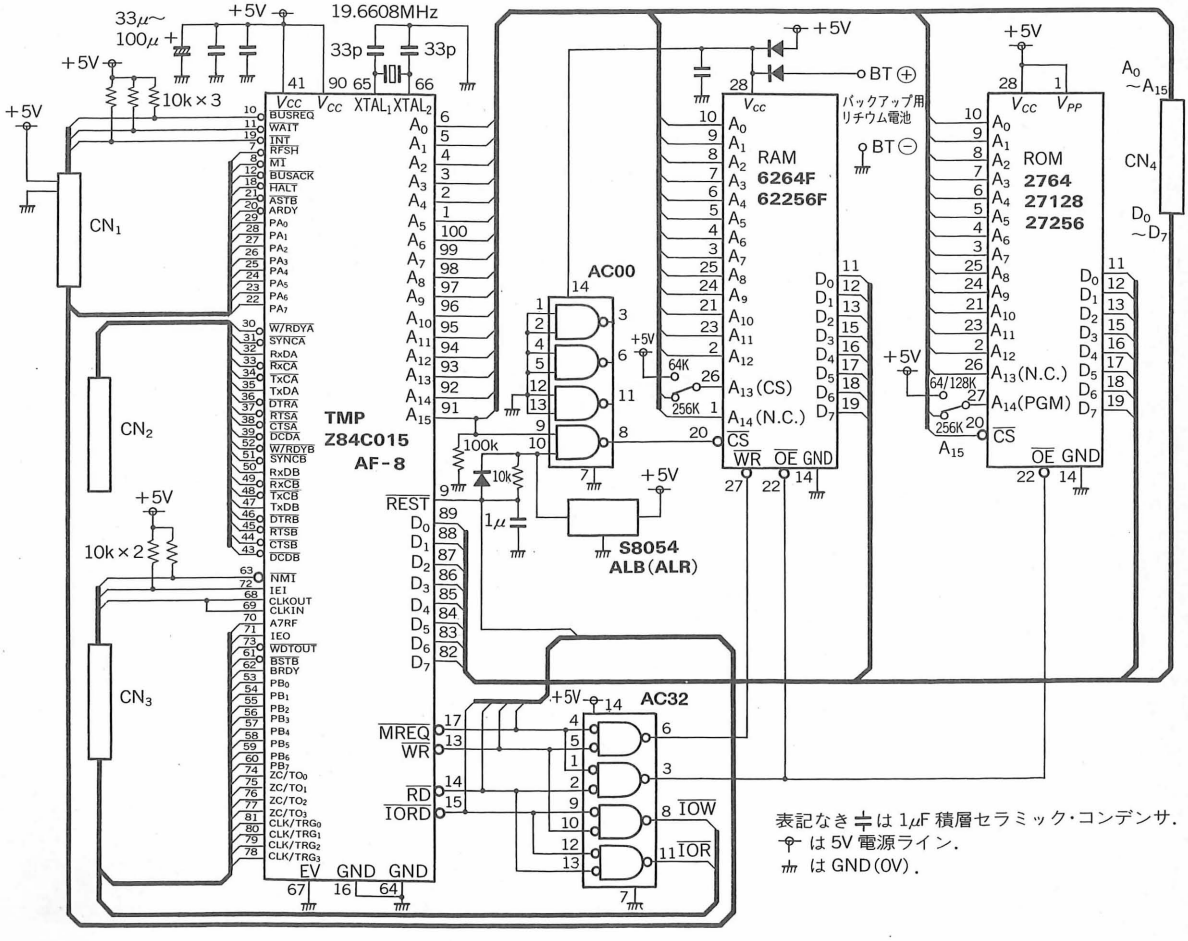


ーキットを入手して，同梱されているマニュアルにしたがって，まずは1台作ってみましょう．ただし電子工作キットと違って，完成したAKI-80のチェックは，まだしばらく先のこととなります．我慢してください．

さて，それでは基本の確認から始めましょう．マイコン・システムというのはリモコン内蔵のワンチップCPUからパソコンの心臓部まで，基本的に図2-1のような構成となっています．CPUはメモリに置かれたプログラムを順に読み出して実行し，外界とのやりとりのためにI/Oを使用します．「ノイマン式のコンピュータ」という意味で，カード・サイズではあるものの，AKI-80もまさにコンピュータ・システムそのものです．

パソコンのように，メイン・メモリがRAMになっていて，ここに補助記憶媒体(フロッピ，ハードディ

〈図2-3〉AKI-80の全体回路図（秋月電子通商）



ROMエミュレータ　マイコン・システムのプログラムを格納するEPROMのソケットにコネクタを接続し，これが内部的にはパソコンからロードされるRAMになっている装置．マイコン・システムはROMだと思って動作するので，開発段階でエミュレータとして使用する．

スク，CD-ROMなど)からプログラムを読み込むのではなく，電源を入れると特定の機能のシステムとして稼動するタイプのマイコン・システムを，一般に「組み込みマイコン・システム」などと呼んでいます．

ここでは**図2-2**のように，プログラムがROMに固定されているために，メモリはおもにプログラム用のROMとデータ用のRAMに分かれています．また，メモリやI/OはCPUのデータ・バスを共用するために，「どの素子がバスを使用するか」を許可するチップ・セレクト信号が必要となり，アドレス・バスからこの信号を生成するデコーダも必須となります．

そして，AKI-80の回路は**図2-3**のようになっています．CPUである東芝のTMPZ84C015（**コラムA**）については後述しますが，これを大きくCPUと見てみると，まさに**図2-2**そのままの回路となっています．基板上にあるIC/LSIとしては，CPU，ROM，ROMソケット下に埋もれたRAM，そしてデコーダの74AC32があり，あとはリセットICとメモリ・バックアップのための74AC00という二つのICがあるだけという，非常にシンプルな回路です．これだけで何ができるのかと思ってしまうほどです．

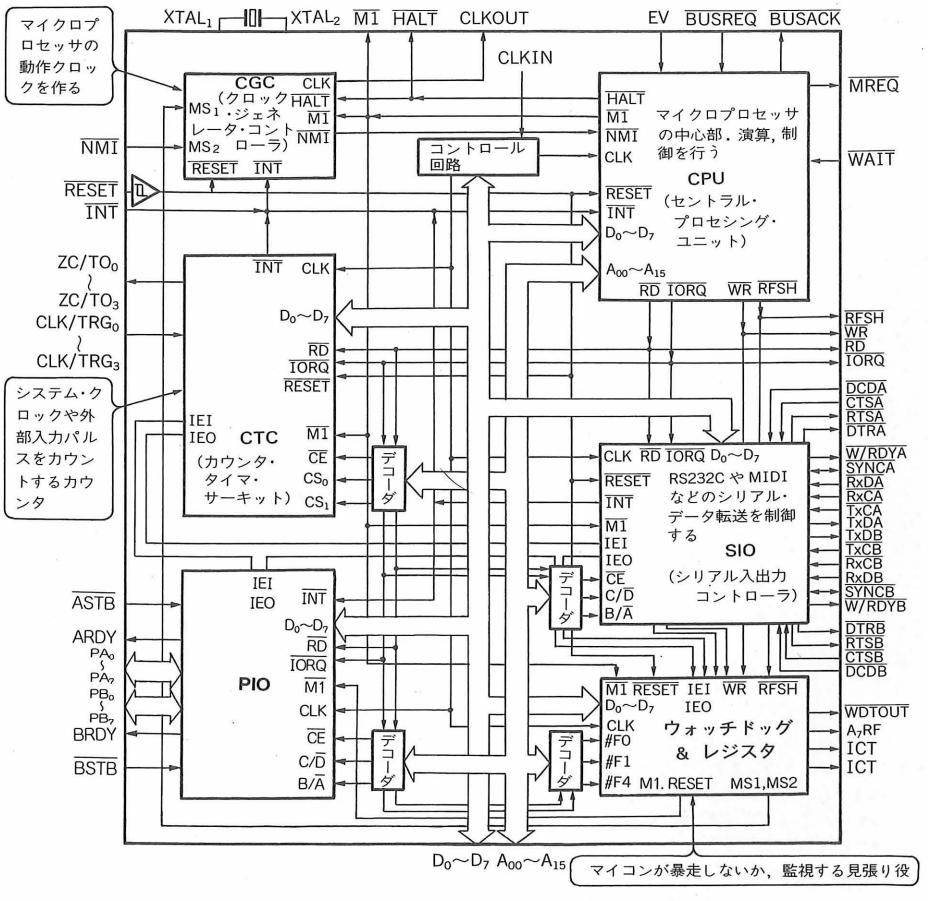
ところがあまく見てはいけません．**図2-4**はAKI-80の心臓部，東芝のTMPZ84C015の内部ブロック図です．ここには，いわゆるZ80CPUに相当するMPUだけでなく，

・水晶発振回路とシステム・クロック・ジェネレータ　:CGC
・4本のプログラマブル・タイマ/カウンタ　:CTC
・2本の8ビット入出力ポート　:PIO
・2本のシリアル通信ポート　:SIO
・ウォッチドッグ・タイマ(暴走検出回路)　:WDT

などの，全部で数個の周辺LSIに相当する回路ブロックと，それらの制御のためのデコーダや制御回路までが搭載されているのです．

つまり，周辺チップや周辺回路をいちいち増設しなくても，このCPUだけで基本的なシステムがそのまま実現できてしまいます．このCPUが東芝から発表された当初は，筆者も「東芝も結局はザイログのZ80から抜け出せないのかな」と思っていましたが，やがて逆にザイログが東芝からライセンスを受けてこのチップのセカンド・ソースを生産するようになったところを見ると，東芝の戦略もなかなか素晴らしかったのかな

〈図2-4〉
TMPZ84C015の
内部ブロック図



SYMDEB　MS-DOSに付属しているシンボリック・デバッガのこと．

と感心してしまいます．

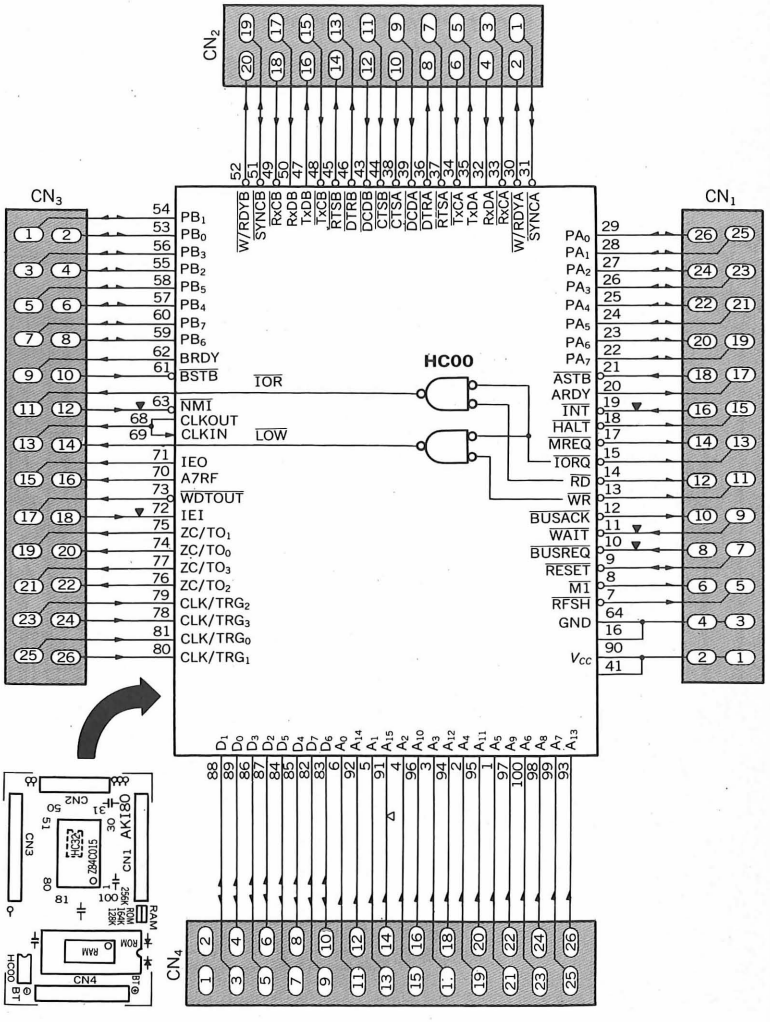
そして，**図2-5**が秋月電子のマニュアルにあるAKI-80のコネクタ・マップです．このキットでは，実際には100ピンの高密度な東芝のCPU，RAMと2個のTTLの計3個のミニフラットICは，はんだ付けされた状態の基板が入っていますから，ユーザは受動部品だけ注意してはんだ付けすればいいようになっています．筆者の経験からすれば，昔の個別部品(CPU,ROM,RAM,PIO,CTC,SIOなど)をはんだ付けして同等のシステムを製作するのにくらべて，体感として一桁以上(10倍～20倍)は楽になっているように思います．

筆者はいつもAKI-80を多めに通販で購入しておいて，暇なときにはんだ付けを行い，「キットの完成品」の形で何個ものAKI-80をストックしています．なお，筆者の場合，後述するようにMIDI関連機器を製作することがほとんどなので，キットに付属してくる水晶を使わずに，8MHzの水晶を取り付けます．そして，なにか新しいシステムをいざ作ろうというときには，このストックをまさに「部品」感覚で使用するのです．

すると，たいていのシステムの場合，ケースの穴あけに1日，AKI-80を使ったシステムの配線に1日，そしてCPUソフト開発に1日という，2～3日仕事でそこそこのものが完成してしまいます．この軽快さは，AKI-80が登場するまでは体験できなかったものです．

実際には，システムを設計，製作するときには**図2-3**も**図2-4**も必要ありません．筆者の場合，たいていは回路図も書かずにいきなり**図2-5**だけを見ながらはんだ付けを行い，ソフトを書いて動くところまでもっていってしまうので，あとから回路を思い出すというのがたいへんです．本書でこれから紹介するいくつかの実例システムは，すでにこのように完成していたものなので，控えの残っているソフトを参考にしたり，実際に分解して配線を追いかけて回路図を復元した

〈図2-5〉
AKI-80のコネクタ・マップ
(秋月電子通商)



▼ はプルアップされている．
△ はプルダウンされている．
→ は入力・出力の方向を示す．
⇀ は入力・出力の方向かつ3ステート端子．
↔ は入出力端子を示す．
⇄ は入出力端子かつ3ステート端子．

ノイマン式コンピュータ　順番にプログラムを読み出して実行していくという，ほとんどすべてのコンピュータの基本となっている方式．フォン・ノイマン博士の名前に由来する．

(というか初めて書いた)ものもあります.

その程度で済んでしまうのも，AKI-80が極限まで「中枢システムの部品化」に成功しているからなのです.

アセンブラ「XA80」

AKI-80を使ったシステムというのは，**図2-2**のように電源ON時にリセットがかかって，CPUの仕様にしたがってROMアドレスの0000h番地からスタートするというものです．そこで，このようなプログラムの書き込まれたROMを作成して，ROMソケットに挿入することが「ソフトウェア開発」という作業になります（実際にROMを作る部分をどうするかについては後述）.

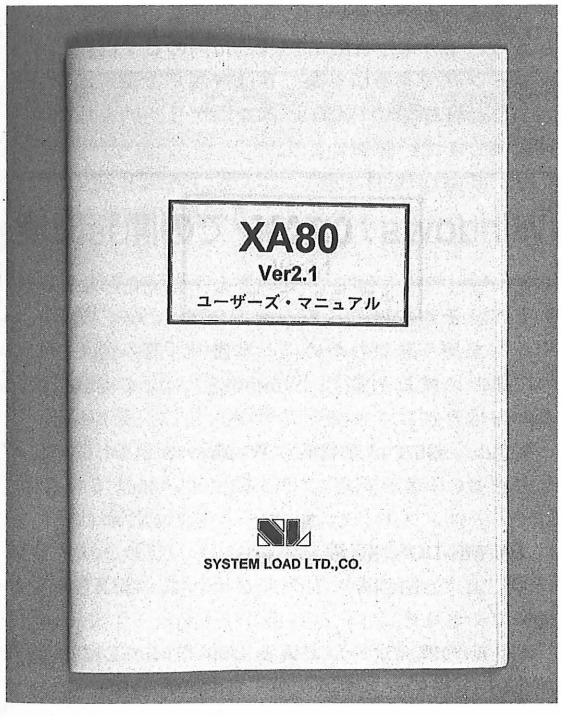
そこで登場するのが，「**アセンブラ**」というツールです．秋月電子では，システムロード社のXA80というアセンブラを販売しています(**写真2-2**)．これは2,500円と安いだけに，筆者がこれまで使ってきた他のアセンブラとは別格の，「何もない」ツールです．たとえば，マニュアルから関係した記述を紹介すると，

「XA80.EXEはMS-DOS上で走るZ80用アブソリュート・クロス・アセンブラです.」
→リンカ不要，絶対番地限定，マクロ使えず(^_^;)
「出力ファイルは，インテルHEX形式で出力されます.」
→他の選択肢はナシ(これはたいしたデメリットではない)
「XA80で使用できる疑似命令には以下の命令があります」
DB,DEFB,DEFM,DC,DS,DEFS,DW,DEFW,END,EQU,ORG
IF,ELSE,ENDIF,COND,ELSE,ENDC,INCLUDE,PAGE,TITLE
→ほとんど何もないのに等しい(^_^;)

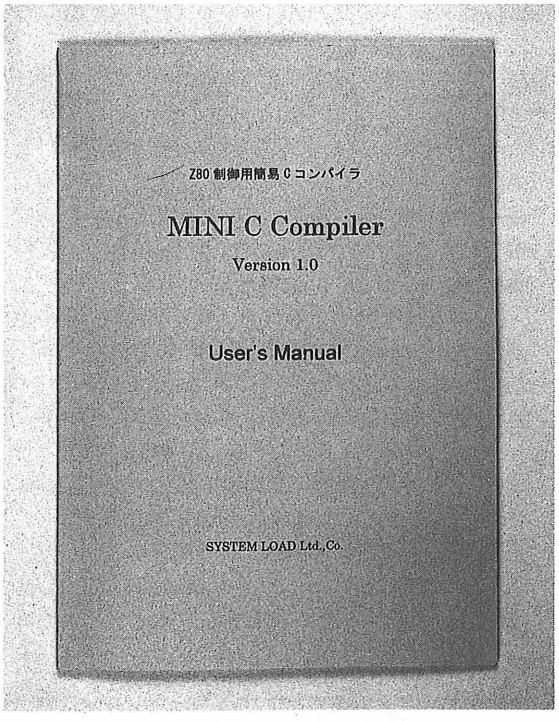
ということで，とくに筆者にとっては「マクロが使えない」というのが非常にひっかかりました．かつて「6502のアセンブラでZ80のソフトを開発」という，マクロの鬼のような体験までしたことのある筆者にとって，マクロでアセンブラの使い勝手を拡張できないのは，かなり厳しい条件といえるのです(この「6502アセンブラでZ80を開発」という意味を本当に理解できる人はもう本書の後半にスキップできます).

ただし，今回はアセンブラだけでなく，このXA80と親和性のあるCコンパイラ(Mini-C)を組み合わせることになっているので，可能性はあります．つまり，従来なら「マクロ技(わざ)」として完備していた各種のサービス・ルーチンをCのライブラリ関数として用意して，その中身をスタック渡しの関数展開でなく，マク

〈写真2-2〉Z80用アブソリュート・アセンブラXA80



〈写真2-3〉Z80制御用簡易コンパイラMini-C体験版



アセンブラ　もっとも簡単な英語に近い「ニーモニック記号」を用いた「アセンブリ言語」によるソース・プログラムを，コンピュータが実行できるための機械語プログラムに変換するツールのこと.

ロ的にそれぞれ**インライン・アセンブラ**で記述してしまうというテクニカルな方法です．

これはマクロアセンブラのマクロ定義とほぼ同等ですから，あえて何もないアセンブラを使うという面白さが生きるようにも思います．

Cコンパイラ「Mini-C体験版」

久しぶりに秋葉原に買い出しに行って，デフォルトで秋月電子に立ち寄り，掘り出しもののパーツを探していた筆者は，前述のXA80とともに，「AKI-80用Cコンパイラ，2,500円」というパックを発見して，迷わず衝動買いしてしまいました．

アセンブラならともかく，Cがこんなに安いとは凄いなぁと思ってよく読んでみると，これは同じシステムロード社の「Mini-C」という15,000円のコンパイラの機能限定「体験版」だったのです(**写真2-3**)．

マニュアル(一部抜粋)によると，

▶ ファイル構成
注意 限定版には*印がついているものは含まれていません．
mc.exe……Mini-Cコンパイラ
kl5c.h ……ライブラリなどを含むインクルード・ファイル
x2list.exe……Cソース・デバッグ情報を組み込むためのユーティリティ
xalst.exe ……アセンブラ・ソース・デバッグ情報のためのユーティリティ
iolibz.lib ……コンパイラ本体が使用するライブラリ・ファイル
startup.h ……ROM化等に必要なmain関数起動用のヘッダ・ファイル
*xa80.exe …アセンブラ
*alllib.h ……すべてのライブラリ・ヘッダ・ファイル
*minic.lib …Mini-Cライブラリ・ソース・ファイル本体
*sample.c …Mini-Cサンプル・プログラム・ソース・ファイル
*math.h ……整数変換関連のライブラリ・ヘッダ・ファイル
*ctyp.h……文字属性を調査するライブラリ・ヘッダ・ファイル
*stdlib.h ……文字変換関連のライブラリ・ヘッダ・ファイル
*string.h ……文字列操作に関連のライブラリ・ヘッダ・ファイル
*stdo.h……バッファ/画面出力に関連のライブラリ・ヘッダ・ファイル
*stdi.h ……コンソール入力に関連したライブラリ・ヘッダ・ファイル

ということで，なんだか肝心なものは何も入っていない，まったく使えないもののように思えます．ところが，試しにちらちらといじってみると，これが驚くほどスグレモノだったのです．ある意味では，本書の企画はこの「Mini-C限定版」との出会いがなかったら実現しなかったかもしれません．

筆者はここで，Mini-Cのフルセットを購入することをあえて我慢することにしました．何もない「素の」Cコンパイラだからこそ，後述するように解析してみればみるほど味が出てきました．そして，XA80と組み合わせてAKI-80のシステム開発環境を構築することで，マイコン技術の全般に関する重要な教材となることを確信しました．

たまたま同時進行でハマっていたJavaとの結びつきも，ここからスタートしたのです．Mini-Cはこの点で，本書のツボともいえる存在です．「コンパイラを制するものはソフトを制する」といわれた時代もありましたが，もっともシンプルなコンパイラの教材として，Mini-C限定版は重要な意義があると思っています．

Mini-CはCソース・プログラムを読み込んで，XA80のソースとなるアセンブラ・ソース・プログラムを出力します．Cの関数のなかにそのままインラインでアセンブラ記述もできます．ただし，かなりC言語としては「方言のキツイ」ものです．Cプログラマは，おそらくあれもできない，これもできないと嘆くことでしょう．しかし，AKI-80を2,500円のCで自在にプログラミングできるいう楽しさは最高だと筆者は思います．プロ仕様のコンパイラにないゲリラ的な攻略を一緒に楽しんでいきましょう．

Windowsパソコンでの開発環境

さて，それではここで，中古PC98でなくWindowsマシンを使う読者のために，本書の「読み替え」についてまとめておきます．Windowsマシンを実験に使わない方はとばしてもけっこうです．

Windows95では悪名高きWindows3.1にくらべてかなり「DOS窓が安定している」といわれています．「DOSプロンプト」というツールを起動すればその中はほぼMS-DOSの世界ですから，ハードウェアを直接に叩くような野蛮なソフトでなければ，ほぼちゃんと動いてくれます．

AKI-80の開発ツールであるXA80やMini-CはDOS汎用(MS-DOSのシステム・コールしか使っていない)で

インライン・アセンブラ C言語のプログラム中に，特定の予約語とともに，一部の処理をC言語でなくアセンブリ言語として記述できること．

すから，問題なく動きます．

Windowsといっても中身はDOSマシンですから，基本的に開発に必要なツールは[*.EXE]という実行ファイルとバッチ・ファイルです．ここで問題となるのは，DOS環境で呼び出すテキスト・エディタ[EDIT.EXE]が貧弱なことと，一連の処理をバッチとしてどう記述して実行するかというところです．

筆者はとりあえず，テキスト・エディタとしては定番の「**秀丸エディタ**」(シェアウェア)を使用しています．つまり，ターゲットのソース・ファイルについては，デスクトップに秀丸エディタのショートカットを置いて，ここにエコロジー代わりのエクスプローラからソース・ファイルのアイコンをドラッグ＆ドロップして秀丸エディタを起動して編集することにしました．

またバッチについては，これもシェアウェアのWinbatchというツールを使ったり，従来のMS-DOSと同様にたんなるバッチ・ファイルを書いて使えることを確認しました．

アセンブラやCコンパイラ，そしてエディタやバッチのソフト環境が揃ってくると，あと必要となるのは，次章で紹介する，自作したROMエミュレータとのインターフェース部分です．

ここには，

- ROMエミュレータとの物理的な接続
- DOS/Vマシンでのプリンタ・ポート処理
- ファイル転送プログラムの開発

という具体的な課題があります．

プリンタ・ポートとの接続については，筆者はROMエミュレータとして最初に98Noteの特別な20pinミニ・コネクタの専用ケーブルで製作したので，「20pinミニのメスと36pinミニのオス」とを変換するアダプタを分解して，36pin側を取り去り，ここに25pinD-subケーブルを接続して，ROMエミュレータとDOS/Vマシンを接続する変換ケーブルを製作しました．テスタで慎重に対応するピンを調べてはんだ付けする必要がありますが，8本の信号と1本のGNDの，基本的には単純な作業です．

また「DOS/Vマシンのパラレル・ポートのI/Oアドレス」が378hであるという情報だけを頼りに，実験的に作ったCプログラムでこのポートを直接叩いてみると，なんとWindowsマシンとはいえ，ちゃんとポートの信号が上下してくれることを確認できました．

そして，第4章で紹介する「インテルHEXファイルを自作のROMエミュレータに転送するためのプログラム」ですが，これもWindowsの「DOS窓」の中で，MS-CもTurbo-Cもちゃんと動作できることを確認できました．そこで，本書に記載された内容は，そのままWindows版でもアドレスの置き換えだけで実施できます（**第4章Appendix**参照）．

実際に筆者は，本書を執筆していた当初は98Noteをメインとして開発していたのですが，今では上記の変換ケーブルを使って，Windowsノートを持ち歩いて開発することがほとんどになっています．

●コラムA● 東芝TMPZ84C015について

1996年の秋になって，AKI-80の心臓部である「東芝のTMPZ84C015が製造中止になる」という噂が広がりました．心臓部のCPUが消えてしまってはAKI-80が使えませんから心配ですが，どうやら調べてみたところ，それほど慌てることはなさそうです．

東芝のこの「Z80コア＋周辺」(TMP-015)は，発表以来，おそらくかなり儲かってきたと思われます．なんせ，Z80の本家ザイログでさえ，東芝からライセンスを受けてセカンド・ソースを作ったぐらいですから．

しかし最近，川崎製鉄の4倍速Z80，日立のSH，NECのVなど，新しいCPU技術が出てきていて，いつまでも東芝だけZ80というのはブランド・イメージとしてよろしくないという判断のようです．

実は東芝は東芝オリジナルのCPUコアとしてCISCもRISCも出しているので，今後の新しい話(電機メーカなどの巨大ユーザからの引き合い)では，こちらを薦めて，新規のビジネスとしてはTMP-015では受けないという方針のようです．

ただし，当初は乱暴にイキナリ製造中止と言ったところ業界からの反発が凄かったので，新規の話はそちらに振るとして，当面はこれまでの製品に対する製造を続けるということらしいので，まぁ4，5年はなんとか続きそうです．

ということで，AKI-80の余命はあと4，5年は持ちそうです．この間には，当然，川崎製鉄のAKI版，JavaチップのAKI版など，いろいろと出てくるでしょうから，結果として，私たちはなんとか「AKI-80手作り」は続けて行けるものと思っています．

秀丸エディタ　Windows95ではもっとも多くのユーザに使われている，超強力なテキスト・エディタ．シェアウェアである．

第3章　AKI-80の開発環境の構築

パソコンを使ったソフト開発の方法とROM化の道具立て

DOSとC言語によるツールの自作

一般にプロの電子技術者の世界では，仕事に必要なツールは専業のツール・メーカの製品を購入して使用します．開発ツール，計測ツール，最近では実験支援ツールやデータ収集，処理ツールもパソコン・ソフトの形態で揃ってきています．

ところが，このようなツールにあまりに頼りすぎる弊害も指摘されるようになりました．つまり，測定器の較正を怠っていて，まったく意味のない誤データを仕事に使い続けていたり，バグのあるソフトをツールとして使用することによって品質の低下が起きたりといったトラブルも起きています．

これに対して，「必要な道具は自分で作る」というアプローチがあります．必要な仕事に本当に使える道具がないからという場合もありますが，専用ツールはあまりに高価だから自作するとか，自分でツールを設計することで本質的な理解が得られるというメリットを指摘する人も少なくありません．

そして，マイコン・システムの開発においても，これは一つの真実なのです．最近ではあまり聞かなくなりましたが，昔は「CPUメーカの提供する開発ツールにはかならずバグがあるので，まずツールを疑え」「LSIメーカはLSI自身の設計ミスを指摘されるとそのユーザには対応方法を教えるが，一般には公開せずに次のロットでさりげなくバグ対策をする」などという金言が通用していました．このような場合，市販のツールを使っているから安心などということはなく，最後に頼りになるのは自分だけなのです．

そこで，本書でも，必要なツールとしてハードだけでなく，ソフトもCでプログラムすることにします．マイコン技術の周辺では「ハード屋」も「ソフト屋」もなく，両方をカバーできなければ一人前の仕事はできません．ハード屋がCADやシミュレータを駆使し，場合によっては簡単なソフトやマクロを自作したり，PLDなどのプログラマブル・ロジックをプログラミングして設計するのは当然の流れです．

そして逆に，ソフト屋がC言語のようなハードウェア記述言語によって，まったくICやゲートの知識がなく数万ゲートのASIC(究極のハードウェア)を設計してしまう時代となっています．話題のJavaも，Cプログラミング経験者には馴染みのものですから，これを機会にCをマスターするのはいいことでしょう．

AKI-80によるシステム開発の手法

さて，ここからは実際にAKI-80と秋月電子通商で入手したツールを前にしての作業となります．まず，以前に教材のために製作した簡単なハードウェアのためのソフトウェアを題材として，アセンブラXA80を使って，どのようにROMプログラムが開発されていくのかを紹介していきましょう．

まず，アセンブラ環境の解析ツールとして，**リスト3-1**のようなバッチ・プログラム[ASM.BAT]を作成しました．たった4行のプログラムですが，ここには本書を通じて幾度となく登場する重要なツールが勢揃いしています．

まず1行目ですが，これは簡単です．テキスト・エディタ**MIFES**を起動して，サンプル・ソース・ファイル[TEST.ASM]を編集するというそれだけのものです．よく，このバッチにはターゲットのファイル名を記述せず，バッチ変数(%1とかいうもの)で汎用にするという手も使われますが，筆者は**リスト3-1**のように固定した名前を使うことを好みます．

それは，複数のプロジェクトで同時に開発をしていると，ターゲット・ファイル名が多数乱立して混乱してしまうためです．

そこで，アセンブル処理の対象はつねに「予約名」

〈リスト3-1〉　解析実験用バッチ・プログラム ASM.BATの内容

```
mifes test.asm                ←  実験用ソース・ファイルの編集
xa80 test,,,;                 ←  アセンブラ［XA80］で処理
copy test.hex transfer.hex ←  予約ファイル名にコピー
trans                         ←  ターゲット・システムにロード
```


MIFES　MS-DOSではVZエディタとともに多くのユーザに使われていた，超強力なテキスト・エディタ．製品ソフトである．

のように固定しておいて，あとはエコロジーのようなファイラで頻繁にファイル名を変更したり，プロジェクトごとのディレクトリに移動を繰り返すようにしています．

2行目では，ここでの核心であるアセンブラXA80にターゲット・ファイルである[TEST.ASM]を指定して起動します．カンマの並ぶオプションは不要な関連ファイル名の入力を省略しているもので，ここではデバッガを使わないので，最終出力の「**インテルHEX形式ファイル**」である[TEST.HEX]が得られれば十分です．

3行目は，このアセンブラの出力結果である[TEST.HEX]を単純に[TRANSFER.HEX]という名前のファイルとしてコピーしているだけのMS-DOS標準コマンドです．この予約名ファイルは，次行で使用されるものです．

4行目は，[TRANS.EXE]というオリジナル・ソフトを起動しているだけです．このソフトについては次章で詳しく紹介しますが，開発したCPUのプログラム(ここでは[TRANSFER.HEX]となっている)を，AKI-80の組み込まれたシステムで稼動させるための「ROMエミュレータ」に転送して，ターゲット・システムをリセット(スタート)させるためのツールです．この部分は次章のメイン・テーマです．

さて，このような解析用バッチで最初に使用したサンプルは，**リスト3-2**のようなプログラムです．アセンブラに馴染みのない方には読みにくいかもしれませんが，何度も登場するうちに，そしてなによりAKI-80とともに使っているうちに親しんでいくと思いますので，最初は我慢して眺めてください．ちなみにこのプログラムは，もともとは筆者のもっていた**BASM80**というアセンブラ用に書いたものなのですが，マクロ定義を削除するなどのほんのわずかの変更で，XA80でも問題なく処理できました．

それでは，AKI-80の中枢である東芝CPUのそのまた中核にある，Z80CPUの簡単なおさらいを兼ねて，**リスト3-2**のプログラムを解説していきましょう．なお，ここでは「パソコンでのCプログラミング経験がある人」にとってヒントになるような，パソコン環境との違いについても触れていきます．

まず最初に「RAM Map」として，アドレス8000h(****hは16進数，Cでは0x****のこと)から，4種類の変数を定義しています．Z80CPUというのは，予約された数少ないレジスタ以外の変数については，通常，基本的にこのようにRAM領域に静的に定義して使います．「ds」で定義した変数は8ビット幅で，Cなら「unsigned char」というところです．

つぎの「I/O Map」という部分では，東芝CPU内部に提供されている8ビット汎用入出力ポートのアドレスを定義しています．これは定義といっても，メーカが予約しているアドレスを参照するためのラベルとして定義したものであって，勝手には決められません．

Z80CPUでは，I/O空間とメモリ空間というのはまったく別になっていて，ハードウェア的にそれぞれのアクセスに対応した別々の選択信号($\overline{\text{MREQ}}$，$\overline{\text{IOREQ}}$)が出力されます．これを「I/Oマップド I/O」といい，メモリもI/Oも区別なく統一のメモリ空間としてアクセスする「メモリ・マップド I/O」のCPUと大きく違っているところです．

つぎの「RESET」の部分は，「org 0000h」とあります．Z80CPUはリセットされるとかならずここからスタートします．このあたりの細かい命令は，本当はすべてが重要なものですが，ここではオマジナイとして常にこのようにしておくという程度で先に進むことにします．

そしていよいよ，「main」という部分にジャンプしてきます．

ここではまず，「pio_a」と「pio_b」と名付けた二つのパラレル・ポートを「初期化」しています．具体的には，pio_aを8ビットすべて出力ポートに，pio_bは上位3ビットを入力用に，下位5ビットを出力用に設定しています．このようなビット単位の設定が行えるというのはシステム設計の上ではとても便利です．命令はロード(ld)とアウト(out) だけですが，Z80CPUではこのように，いちいち**アキュムレータ**(Aレジスタ)を介在させなければならないので，ちょっとほかのCPUに慣れた人には目障りなようです．アセンブラにマクロ機能がある場合には，このようなAレジスタを経由する処理を1行のマクロとしてまとめて定義しておいて，たとえば「mov src,dst」(メモリsrcの内容をメモリdstに転送)などと記述してしまうことも可能です．

ソフトの主要な部分はあとわずかです．初期化の最後にcounterという変数をゼロ・クリアしていますが，「xor a」というのは，Aレジスタ自身でエクスクルーシブOR(排他的論理和)をとるという(結果はかならずゼロ)定石です．このあとの5行で，ソフトは無限ループになっています．

まず「wait_long」というダミー・ウェイトのサブルーチンをコールして，つぎにcounterをインクリメント(オーバフローすればゼロに勝手に戻る)して，つぎに「led_display」というLED表示のためのサブルーチンをコールしています．

つまり，このプログラムは，刻々とインクリメントされるcounterの値を，LEDで表示するというだけのプログラムなのです．

リスト3-2のこれ以降の部分については，詳しい解説を省略します．本書とともにAKI-80プログラミング

インテルHEX形式ファイル　マイコン・システムのプログラムを格納するEPROMに機械語として書き込む「PROMライタ」という装置に，プログラムを開発したパソコンからデータを転送する際に一般的に使用されるデータ形式の一種．テキスト・ファイルである．

〈リスト 3-2〉 サンプル・ソースTEST.ASMの内容

```
;##### RAM Map #####
            org     8000h           ←  RAM領域に変数を定義
disp        ds      1
sw          ds      1
timer       ds      2
counter             ds      1

;##### I/O Map #####
pio_a               equ     001ch ←  内蔵PIOのアドレス定義
pio_b               equ     001eh

;##### RESET #####
            org     0000h           ←  リセットするとここから始まる
            ld      sp,09fffh       ←  スタック・ポインタをセット
            di                      ←  割り込みを禁止
            jp       main           ←  メインに飛ぶ

;##### INT / NMI #####
            org      0066h
            retn

;##### Main #####
main:
            ld      a,0cfh          ; Mode 3
            out     (pio_a+1),a
            ld      a,00000000b     ; 0:Out / 1:In
            out     (pio_a+1),a
            ld      a,0cfh          ; Mode 3
            out     (pio_b+1),a
            ld      a,11100000b     ; 0:Out / 1:In
            out     (pio_b+1),a
            ld      a,11111111b
            out     (pio_b+0),a
            ld      a,11111000b     ←  3個のLEDにデータをセット
            out     (pio_b+0),a
            ld      a,11111111b
            out     (pio_b+0),a
            xor     a
            ld      (counter),a
loop:
            call    wait_long       ←  ソフトウェア・タイマ
            ld      a,(counter)
            inc     a               ←  インクリメントしたデータを
            ld      (counter),a
            call    led_display     ←  3桁のLEDで表示
            jr      loop

;##### Subroutines #####
wait_long:
            ld      a,50
            ld      (timer+0),a
_w_loop_1:
            call    sw_scan         ←  ウェイト中にスイッチを見る
            ld      a,(sw)
            cp      0
            jr      z,_pass
            xor     a
            ld      (counter),a     ←  スイッチがONなら表示をクリア
_pass:
            call    wait_short      ←  足踏みルーチンを呼ぶ
            ld      a,(timer+0)
            dec     a
            ld      (timer+0),a
            cp      0
            jr      nz,_w_loop_1
            ret
wait_short:
            ld      a,10
            ld      (timer+1),a
_w_loop_2:
            nop                     ←  NOPを5回、それを10回ループ
            nop
            nop
            nop
            nop
            ld      a,(timer+1)
            dec     a
            ld      (timer+1),a
            cp      0
            jr      nz,_w_loop_2
            ret
sw_scan:
            in      a,(pio_b)       ←  ポートから入力
            and     11100000b
            srl     a
            srl     a
            srl     a
            srl     a
            srl     a
            xor     11111111b
            and     00000111b
            ld      (sw),a          ←  スイッチ状態を記憶する
            ret
led_disp_0:
            ld      a,(disp)
            ld      c,a
            ld      b,0
            ld      hl,table_0      ←  7セグメントの配置テーブル
            add     hl,bc
            ld      a,(hl)
            out     (pio_a),a
            ld      a,11111110b
            out     (pio_b+0),a     ←  [1]の位のデータをラッチする
            ld      a,11111111b
            out     (pio_b+0),a
            ret
led_disp_1:
            ld      a,(disp)
            ld      c,a
            ld      b,0
            ld      hl,table_0
            add     hl,bc
            ld      a,(hl)
            out     (pio_a),a
            ld      a,11111101b
            out     (pio_b+0),a     ←  [10]の位のデータをラッチする
            ld      a,11111111b
            out     (pio_b+0),a
            ret
led_disp_2:
            ld      a,(disp)
            ld      c,a
            ld      b,0
            ld      hl,table_0
            add     hl,bc
            ld      a,(hl)
            out     (pio_a),a
            ld      a,11111011b
            out     (pio_b+0),a     ←  [100]の位のデータをラッチする
            ld      a,11111111b
            out     (pio_b+0),a
            ret
table_0:
            db      11110101b,00000101b,11010011b,01010111b ;
            db      00100111b,01110110b,11110110b,01100101b ;
            db      11110111b,01110111b,00000000b
led_display:
            ld      a,(counter)
            ld      c,a
            ld      b,0
            ld      hl,table_1      ←  (data%10)を求めるテーブル
            add     hl,bc
            ld      a,(hl)
            ld      (disp),a
            call    led_disp_0      ←  1の位を表示
            ld      a,(counter)
            ld      c,a
            ld      b,0
            ld      hl,table_2      ←  ((data/10)%10)を求めるテーブル
            add     hl,bc
            ld      a,(hl)
            ld      (disp),a
            call    led_disp_1      ←  10の位を表示
            ld      a,(counter)
            ld      c,a
            ld      b,0
            ld      hl,table_3      ←  (data/100)を求めるテーブル
            add     hl,bc
            ld      a,(hl)
            ld      (disp),a
            call    led_disp_2      ←  100の位を表示
            ret
table_1:
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ;  0 -  19
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ;  20 -  39
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ;  40 -  59
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ;  60 -  79
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ;  80 -  99
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ; 100 - 119
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ; 120 - 139
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ; 140 - 159
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ; 160 - 179
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ; 180 - 199
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ; 200 - 219
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9          ; 220 - 239
            db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5                  ; 240 - 255
table_2:
            db      10,10,10,10,10,10,10,10,10,10,1,1,1,1,1,1,1,1,1,1 ;0 -  19
            db      2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3               ;  20 -  39
            db      4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5           ;  40 -  59
            db      6,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,7,7           ;  60 -  79
            db      8,8,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,9,9           ;  80 -  99
            db      0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1           ; 100 - 119
            db      2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3           ; 120 - 139
            db      4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5           ; 140 - 159
            db      6,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,7,7           ; 160 - 179
            db      8,8,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,9,9           ; 180 - 199
            db      0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1           ; 200 - 219
            db      2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3           ; 220 - 239
            db      4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5                   ; 240 - 255
table_3:
            db      10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
            db      10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
            db      10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
            db      10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
            db      10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
            db      1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1           ; 100 - 119
            db      1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1           ; 120 - 139
            db      1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1           ; 140 - 159
            db      1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1           ; 160 - 179
            db      1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1           ; 180 - 199
            db      2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2           ; 200 - 219
            db      2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2           ; 220 - 239
            db      2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2                   ; 240 - 255

end
```


BASM80　ベータ・システム社のZ80用アセンブラの名称.

に慣れていけばわかってくる程度の簡単なものです．ポイントとしては，定数テーブルもソフトの中に置いてしまえるというところでしょう．つまり，7セグメントLEDの各ドットを表示すべき数字と対応させるところで，いちいち比較とジャンプを使うのでなく，入力の数字をアドレスとする配列(テーブル)の内容をすぐに引き出すという常套的なテクニックです．

このようなアセンブラ・ソース・ファイルを**リスト3-1**にあるようにXA80に与えると，エラーがあって停止しない場合には，最終的にいくつかのファイルが作られます．**リスト3-3**(p.20)はアセンブル作業の結果を示した「アセンブル・リスト」と呼ばれるファイル[TEST.LST]の内容です．ここでは，CPUが呼び出すメモリ・アドレスがリストの左端に順に表示されています．当然のことですが，連続した命令は連続したアドレスに並んでいて，これを順に処理していくことがわかります．

アドレスの隣にある16進数の羅列が，いわゆる「機械語プログラム」そのもので，実際にプログラム・メモリに書き込まれるのはこのデータ(本当はさらに“1”と“0”の2進数)なのです．アセンブラというツールのなかった時代のプログラマは，開発に際してこれだけをプログラムとして記述したというのですがちょっと信じられませんね．

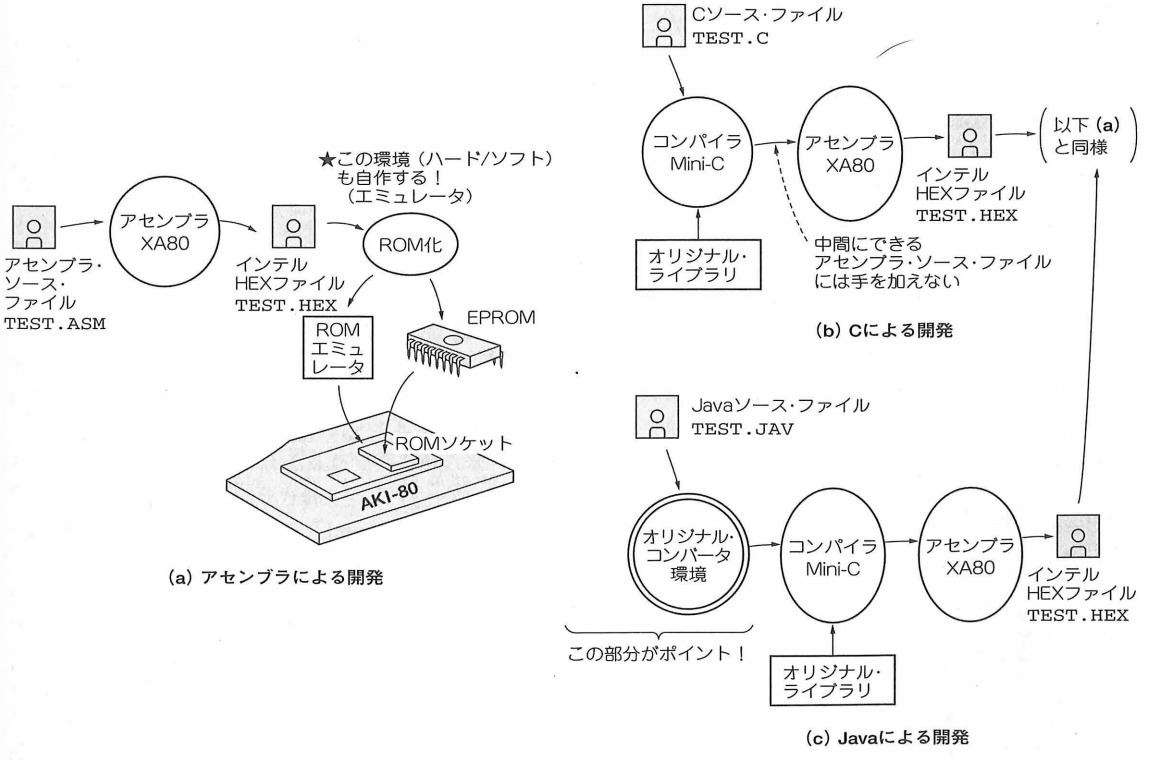
また，**リスト3-4**(p.20)の[TEST.SYM]のようなシンボル・ファイルというものも作られます．これは，人間のプログラマが理解しやすいように勝手に定義した「名前」と対応するデータとの参照表のようなものです．

このファイルは，デバッガというツールで使用されますが，今回はまず使うことはありません．

そして最後に，XA80は**リスト3-5**(p.21)の[TEST.HEX]のような「インテルHEXフォーマット」ファイルを生成します．これは，16進文字ばかりが並ぶテキスト・ファイルとなっています．情報は完全にバイナリの情報ですが，ファイルとしてはテキスト・ファイルであるというところをチェックしておきましょう．このファイルは，1行ごとに**リスト3-6**(p.20)のような意味をもつと定義されていて，マイコン屋の世界ではこの形式のデータをROM化するというのが常識となっています．

このインテルHEXファイルの処理については，Java関連ツールの開発とも似たところがあるので，ツール製作のところで詳しく検討します．

〈図3-1〉 AKI-80システムのソフト開発環境



アキュムレータ　CPUの中心で演算処理を行うブロック．加算，ビット操作，シフト処理などを行う．

〈リスト3-4〉 シンボル・ファイルTEST.SYMの内容

```
00AA _pass          009C _w_loop_1      00BE _w_loop_2
8004 counter        8000 disp           00E5 led_disp_0
00FB led_disp_1     0111 led_disp_2     0132 led_display
0088 loop           0068 main           001C pio_a
001E pio_b          8001 sw             00CF sw_scan
0127 table_0        0166 table_1        0266 table_2
0366 table_3        8002 timer          0097 wait_long
00B9 wait_short
```

〈リスト3-6〉 インテルHEXファイル TEST.HEXのフォーマット

(通常データ部)

```
:      07     0000     00     31 FF 9F F3 C3 68 00     0C
```

コロン　バイト数　先頭アドレス　(無視)　データ本体　チェック・サム

(最終データ部)

```
:      00     0000      01     FF
```

コロン（ここは00 0000とする）　チェック・サム

これが最終の印

〈リスト3-3〉 アセンブル結果ファイルTEST.LSTの内容

```
                ;##### RAM Map #####
8000                    org     8000h
8000             disp   ds      1
8001             sw     ds      1
8002             timer  ds      2
8004             counter ds     1

                ;##### I/O Map #####
001C            pio_a   equ     001ch
001E            pio_b   equ     001eh

                ;##### RESET #####
0000                    org     0000h
0000  31 FF 9F          ld      sp,09fffh
0003  F3                di
0004  C3 68 00          jp      main

                ;##### INT / NMI #####
0066                    org     0066h
0066  ED 45             retn

                ;##### Main #####
0068            main:
0068  3E CF             ld      a,0cfh          ; Mode 3
006A  D3 1D             out     (pio_a+1),a
006C  3E 00             ld      a,00000000b     ; 0:Out / 1:In
006E  D3 1D             out     (pio_a+1),a
0070  3E CF             ld      a,0cfh          ; Mode 3
0072  D3 1F             out     (pio_b+1),a
0074  3E E0             ld      a,11100000b     ; 0:Out / 1:In
0076  D3 1F             out     (pio_b+1),a
0078  3E FF             ld      a,11111111b
007A  D3 1E             out     (pio_b+0),a
007C  3E F8             ld      a,11111000b
007E  D3 1E             out     (pio_b+0),a
0080  3E FF             ld      a,11111111b
0082  D3 1E             out     (pio_b+0),a
0084  AF                xor     a
0085  32 04 80          ld      (counter),a
0088                    loop:
0088  CD 97 00          call    wait_long
008B  3A 04 80          ld      a,(counter)
008E  3C                inc     a
008F  32 04 80          ld      (counter),a
0092  CD 32 01          call    led_display
0095  18 F1             jr      loop

                ;##### Subroutines #####
0097            wait_long:
0097  3E 32             ld      a,50
0099  32 02 80          ld      (timer+0),a
009C                  _w_loop_1:
009C  CD CF 00          call    sw_scan
009F  3A 01 80          ld      a,(sw)
00A2  FE 00             cp      0
00A4  28 04             jr      z,_pass
00A6  AF                xor     a
00A7  32 04 80          ld      (counter),a
00AA                  _pass:
00AA  CD B9 00          call    wait_short
00AD  3A 02 80          ld      a,(timer+0)
00B0  3D                dec     a
00B1  32 02 80          ld      (timer+0),a
00B4  FE 00             cp      0
00B6  20 E4             jr      nz,_w_loop_1
00B8  C9                ret
00B9                  wait_short:
00B9  3E 0A             ld      a,10
00BB  32 03 80          ld      (timer+1),a
00BE                  _w_loop_2:
00BE  00                nop
00BF  00                nop
00C0  00                nop
00C1  00                nop
00C2  00                nop
00C3  3A 03 80          ld      a,(timer+1)
00C6  3D                dec     a
00C7  32 03 80          ld      (timer+1),a
00CA  FE 00             cp      0
00CC  20 F0             jr      nz,_w_loop_2
00CE  C9                ret
00CF                  sw_scan:
00CF  DB 1E             in      a,(pio_b)
00D1  E6 E0             and     11100000b
00D3  CB 3F             srl     a
00D5  CB 3F             srl     a
00D7  CB 3F             srl     a
00D9  CB 3F             srl     a
00DB  CB 3F             srl     a
00DD  EE FF             xor     11111111b
00DF  E6 07             and     00000111b
00E1  32 01 80          ld      (sw),a
00E4  C9                ret
00E5                  led_disp_0:
00E5  3A 00 80          ld      a,(disp)
00E8  4F                ld      c,a
00E9  06 00             ld      b,0
00EB  21 27 01          ld      hl,table_0
00EE  09                add     hl,bc
00EF  7E                ld      a,(hl)
00F0  D3 1C             out     (pio_a),a
00F2  3E FE             ld      a,11111110b
00F4  D3 1E             out     (pio_b+0),a
00F6  3E FF             ld      a,11111111b
00F8  D3 1E             out     (pio_b+0),a
00FA  C9                ret
00FB                  led_disp_1:
00FB  3A 00 80          ld      a,(disp)
00FE  4F                ld      c,a
00FF  06 00             ld      b,0
0101  21 27 01          ld      hl,table_0
0104  09                add     hl,bc
0105  7E                ld      a,(hl)
0106  D3 1C             out     (pio_a),a
0108  3E FD             ld      a,11111101b
010A  D3 1E             out     (pio_b+0),a
010C  3E FF             ld      a,11111111b
010E  D3 1E             out     (pio_b+0),a
0110  C9                ret
0111                  led_disp_2:
0111  3A 00 80          ld      a,(disp)
0114  4F                ld      c,a
0115  06 00             ld      b,0
0117  21 27 01          ld      hl,table_0
011A  09                add     hl,bc
011B  7E                ld      a,(hl)
011C  D3 1C             out     (pio_a),a
011E  3E FB             ld      a,11111011b
0120  D3 1E             out     (pio_b+0),a
0122  3E FF             ld      a,11111111b
0124  D3 1E             out     (pio_b+0),a
0126  C9                ret
0127                  table_0:
0127  F5 05 D3 57       db      11110101b,00000101b,11010011b,01010111b
012B  27 76 F6 65       db      00100111b,01110110b,11110110b,01100101b
012F  F7 77 00          db      11110111b,01110111b,00000000b
0132                  led_display:
0132  3A 04 80          ld      a,(counter)
0135  4F                ld      c,a
0136  06 00             ld      b,0
0138  21 66 01          ld      hl,table_1
013B  09                add     hl,bc
013C  7E                ld      a,(hl)
013D  32 00 80          ld      (disp),a
0140  CD E5 00       call  led_disp_0
0143  3A 04 80          ld      a,(counter)
0146  4F                ld      c,a
0147  06 00             ld      b,0
0149  21 66 02          ld      hl,table_2
014C  09                add     hl,bc
014D  7E                ld      a,(hl)
014E  32 00 80          ld      (disp),a
0151  CD FB 00       call  led_disp_1
0154  3A 04 80          ld      a,(counter)
0157  4F                ld      c,a
0158  06 00             ld      b,0
015A  21 66 03          ld      hl,table_3
015D  09                add     hl,bc
015E  7E                ld      a,(hl)
015F  32 00 80          ld      (disp),a
0162  CD 11 01       call  led_disp_2
0165  C9                ret
0166                    table_1:
0166  00 01 02 03       db      0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9
016A  04 05 06 07
016E  08 09 00 01
0172  02 03 04 05
0176  06 07 08 09
017A  00 01 02 03    db    0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9
017E  04 05 06 07
0182  08 09 00 01
0186  02 03 04 05
018A  06 07 08 09

  (... 中略...)

0456  02 02 02 02    db    2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
045A  02 02 02 02
045E  02 02 02 02
0462  02 02 02 02

0466                    end
```

```
:0700000031FF9FF3C368000C
:20006600ED453ECFD31D3E00D31D3ECFD31F3EE0D31F3EFFD31E3EF8D31E3EFFD31EAF32AA
:2000860000480CD97003A04803C320480CD320118F13E32320280CDCF003A0180FE00280414
:2000A600AF320480CDB9003A02803D320280FE0020E4C93E0A320C8000000000003A038014
:2000C6003D320380FE0020F0C9DB1EE6E0CB3FCB3FCB3FCB3FCB3FEEFFE607320180C93AD0
:2000E60000804F0600212701097ED31C3EFED31E3EFFD31EC93A00804F06002127010976E3
:20010600D31C3EFDD31E3EFFD31EC93A00804F0600212701097ED31C3EFBD31E3EFFD31EA1
:20012600C9F505D3572776F665F777003A04804F0600216601097E320080CDE5003A048022
:200146004F0600216602097E320080CDFB003A04804F0600216603097E320080CD1101C93C
:2001660000010203040506070809000102030405060708090001020304050607080900010F1
:2001860002030405060708090001020304050607080900010203040506070809000102030CD
:2001A60004050607080900010203040506070809000102030405060708090001020304050A9
:2001C60006070809000102030405060708090001020304050607080900010203040506078 5
:2001E60008090001020304050607080900010203040506070809000102030405060708096 1
:2002060000010203040506070809000102030405060708090001020304050607080900015 0
:2002260002030405060708090001020304050607080900010203040506070809000102032C
:2002460004050607080900010203040506070809000102030405060708090001020304050 8
:20026600A0A0A0A0A0A0A0A0A0A0A0101010101010101010202020202020202020203030F0
:2002860003030303030303030404040404040404040505050505050505060606060CE
:2002A60006060606060607070707070707070708080808080808080809090909090948
:2002C60009090909000000000000000000000001010101010101010202020202020202DA
:2002E60002020303030303030303030404040404040404040505050505050505050057C
:2003060006060606060606060607070707070707070708080808080808080809090F3
:2003260009090909090900000000000000000000000001010101010101010202020250D
:2003460002020202020303030303030303030404040404040404040505050505050527
:20036600A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A037
:20038600A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A017
:2003A6000A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0AF7
:2003C6000A0A0A0A0101010101010101010101010101010101010101010101010101010D3
:2003E6000101010101010101010101010101010101010101010101010101010101010D7
:2004060001010101010101010101010101010101010101010101010101010101010101B6
:2004260001010101010101020202020202020202020202020202020202020202020202027E
:2004460002020202020202020202020202020202020202020202020202020202020202056
:00000001FF
```

〈リスト3-5〉
インテルHEXファイルTEST.HEXの内容

ソフト開発環境の展望とJava

さて，ここまでのところでは「ばりばりのアセンブラ開発で，十数年以上も昔と同じ」と思われるかもしれませんが，ここからがいよいよオリジナル路線となってきます．**図3-1**は本書でのAKI-80システムのソフト開発環境の流れを示したものです．

図(a)の「アセンブラによる開発」というのは，上述のツールXA80を中心としたものです．最終結果のインテルHEXファイルは，今回はオリジナルに製作する「ROMエミュレータ」によってターゲット・システムに組み込まれたAKI-80上に転送されてテストを行い，検証が済めばROMライタによってEPROMにプログラムされて完成となります．

また，Mini-C体験版を利用した「Cによる開発」が図(b)に示した流れです．ここでは，極限までカットされているCコンパイラを補充するために，かなりオリジナルのライブラリを完備する必要がありますが，努力に応じた内容を期待できそうです．また，Mini-Cは出力としてXA80のソース・ファイルを生成するのですが，このファイルにはまったく手を触れないでオートマチックに行くという方針とします．

そして，本書の目玉である「JavaによるAKI-80の開発」というのが図(c)なのですが，ここではあえて曖昧に書いてあります．つまり，「Javaで記述されたプログラムをMini-Cのソースであるに変換する」という方法には，実はいろいろな可能性があるのです．もっとも簡単には「Javaソースを構文解析と置換で単純にCソースに変換する」という方法がありますが，筆者はこれよりも凝った方法をとります．この点についてはMini-Cの解析のときに考察することと関係が深いので，ここでは伏せたままにしておくことにします．

ICEとROMエミュレータとROMライタ

図3-1の開発フロー最終段のところの「ROM化」という部分についても，実はいくつもの選択肢があります．大きく分けると「どうデバッグするか」という要素と「どうROMを焼くか」という要素になるのですが，まず本書では，「ROMライタとROMイレーサは自作しない」という方針をとります．後述するROMエミュレータを製作するのにROMライタを製作しないのはアンバランスのようにも思えますが，前者は標準的なディジタル回路として初めての読者でも簡単に製作できるのに対して，後者まで広げると，電源系統や信頼度などの，本書のテーマから外れた余分な項目がかなり出てくるためです．

本書のROMエミュレータを製作した上で，これをROMライタ機能まで拡張するのはそれほど難しいことではありませんから，興味のある方は「トランジスタ技術」誌にときどき載っている記事などを参考にして，トライしてみてください．

筆者の使用している環境は，ROMライタとして，

- PC98拡張ボード・タイプのもの(インターフェース社)
- ROMライタ・キット（秋月電子通商）

を活用し，ROMイレーサとしてはサンハヤト（株）の小型のものを使用しています．いずれも1万円ほどでプロ仕様のものが入手できるので，この部分をアマチ

PLD　Programmable Logic Deviceの略．専用のツールを使って，チップ上の回路機能を外部からプログラムできるIC．

ュアが自作するメリットがないほど「使える」ものです．最近は廉価なROMライタでも，PLDやFPGAをプログラムできる製品が出てきていますから，これを使うのもいいと思います．

そして，最終的にEPROMを焼く(プログラムする)までのデバッグで活用するのが，ROMソケットに挿入するタイプの「ROMエミュレータ」です．

本書では，パソコンのシステムとディジタル回路の仕組みを理解する意味からも，このツールの製作からスタートします．従来のマイコン・システムでは，CPUソケットに挿入するICE(In Circuit Emulator)がよく使われていたのですが，AKI-80のようなフラット・パッケージのCPUが増えてきたために，現在ではROMエミュレータが主流になりつつあるようです．この部分は，次章で詳しく検討します．

アプリケーション例：MIDIとComputer Music

さて，本書の導入部の最後として，実際のAKI-80応用システムのターゲットについて触れておきましょう．普通であれば，スイッチとLEDをパネルに並べた実例，つぎにRS-232-Cでパソコンのターミナルの実例，さらに温度センサを使った温度計などの製作例が並ぶというのが一般的でしょう．

しかし本書では，筆者の趣味に徹底して，もう少し「処理の厳しい」音楽への応用例を紹介していくことにします．

具体的なターゲットを一言で述べるなら，「MIDIとComputer Musicに関連した機器」ということになります．MIDIとは，世界中のメーカの電子楽器をお互いに接続して音楽演奏情報をやりとりできる国際規格で，非同期通信としてはまずまずの31.25　kbpsのスピードをもっています．そして，音楽演奏というのは，たとえば「ピアノの鍵盤を同時に10鍵，連続して刻む」という状況でわかるように，かなりのイベント密度にリアルタイムで対応できる情報処理が必要となります．いくら情報を漏らさず溜めていても，鍵盤を弾いてちょっとしてから音が出るのでは使えませんから，リアルタイム・システムの実験としてはいい対象です．

また，最近ではカラオケの音源としてMIDI規格の上位に位置する「スタンダードMIDIファイル(SMF)」規格，そしていろいろな音源の間で音色の互換性を求めた「ジェネラルMIDI(GM)」規格（たとえば1番の音色名はピアノなどと規定したもの）も普及して，インターネットのホームページでもMIDI形式の音楽データを自動演奏することが一般的になってきました．MIDI規格については，**Appendix1**と**Appendix2**にまとめてありますので参照してください．

そして，加速度や曲げゲージやジャイロ・センサなど，各種のセンサを利用した新しい「楽器」というものも研究されていて，AKI-80が活躍するにはなかなか面白いターゲットがたくさんあるのです．

そこで，本書では，筆者が実際に研究者，作曲家，技術者として開発し，音楽演奏の公演に利用してきた各種の機器への応用事例を紹介して，関連した技術について検討していきます．音楽に利用できるぐらいのリアルタイム性と**多重割り込みシステム**の設計ができれば，たいていの応用はできるといえるほどに奥が深いものであることが，だんだん理解できると思います．

多重割り込みシステム　割り込み要因が一つでなく多数あり，ある割り込みを処理している最中にさらに緊急度の高い割り込みの処理を実行できるシステム．

MIDI規格について

ここでは本書が「MIDI関係のリファレンス」としても活用できるように，MIDIに関する全体をまとめて紹介します．規格の細かい部分まですべて詳細に並べる誌面はありませんので，より詳しい情報の欲しい方は，たとえばICMA(International Computer Music Association)のホームページ，
`http://www.coos.dartmouth.edu/~rsn/icma/icma.html`
などを参照してみてください．

MIDI規格の基礎部分の階層

MIDI規格はもともと，「異なるメーカの電子楽器同士を接続して音楽演奏情報を伝達できる」というために規定されました．

● ハードウェアの規約

そのもっとも基礎となるハードウェア部分の規約は図5-12(p.53)にあるように，

- コネクタは5ピンのDINコネクタで，ピン番号も指定されている
- UARTの出力からオープン・コレクタで5 mA 電流ループで出力する（MIDI OUT）
- 高速フォトカプラで入力し，電気的に分離してUARTに入力する（MIDI IN）
- MIDI入力を上記のように入力して出力するスルー出力（MIDI THRU）

となっています．データ形式の階層では，

- 非同期シリアル通信のため8251のような**USART**を使う
- スタート1ビット，データ8ビット，ストップ1ビット
- データ伝送速度は31.25 kbps

となっています．つまり，1バイトの情報を伝送するのに，およそ320　μs，もっとわかりやすく言うと「1 msで最大3バイト」の伝送量があります．これは最近のコンピュータ・システムから考えると相当に遅い通信速度ですが，1983年から変わっていないもので，人間の音楽演奏情報の伝送としては，まずまず普通には問題なく「使える」ものなのです．

● ソフトウェア規約の階層

この電気的規約の階層の上にあるのが，ソフトウェア的な規約の階層です．データ形式としては，8ビットの「素の」データの最上位ビット(MSB)で大きく意味が分かれて，MSBが“1”のデータを「ステータス・バイト」，MSBが“0”のデータを「データ・バイト」といいます．

また，MIDIでは同時に16種類の異なる楽器の演奏情報を伝達するために，これを区別する4ビットの「チャネル」という概念があり，これをステータス・バイトの下4ビットに割り当てています．たとえば「発音開始」というステータス・バイトには「第*チャネルの」と言うチャネル情報が添えられているので，音源では異なったチャネルが間違って発音することはありません．

MIDIメッセージの階層

このように「ステータス」と「データ」を組み合わせて，複数バイトである音楽的な意味を規定した「塊」を「MIDIメッセージ」と言います．

MIDIは非同期信号ですから，あるメッセージを構成する一連のデータが揃って到達するとは限りません．たとえば，「発音第1チャネルの中央“C#”の音階の音が強度100で演奏された発音開始」という情報は3バイトからなるメッセージですが，これが一気に1 msの間に揃って到達しても，3バイトがそれぞれ10 secずつの間隔で届いても，MIDI受信側ではその情報を保存しておいて，メッセージとして完結した瞬間に発音しなければならないのです．

代表的な「ボイス・メッセージ」と呼ばれるMIDIメッセージには**表3-1**のような種類があります．

この意味は，たとえば16進で「90」というステータスで始まるメッセージは，ステータス・バイトの下4ビット，つまり「0」から第1チャネルの情報で，これに続く2バイトのデータ・バイトで音高(pitch)と強度(velocity)を表した「ノート・オン」（発音開始）という音楽演奏情報を意味するというものです．

すべてのボイス・メッセージが3バイトで構成されているわけではなく，たとえば音色の切り替えに使われる「プログラム・チェンジ」は2バイト構成ですから，MIDI受信のプログラムは，ステータス・バイトの上位4ビットの内容に応じて，あと何バイトのデータでメッセージが完結するのかを判定する必要があります．

また，7ビットのデータ（128段階）では表現できない微妙なピッチを表現するための「ピッチ・ベンド」

USART　Universal Synchronous Asynchronous Receiver Transmitterの略．同期/非同期のシリアル通信のいろいろな機能を搭載した専用LSI．

〈表3-1〉 MIDIメッセージ

status byte	meaning	data bytes
0x80-0x8f	note off	2 - 1 byte pitch, followed by 1 byte velocity
0x90-0x9f	note on	2 - 1 byte pitch, followed by 1 byte velocity
0xa0-0xaf	key pressure	2 - 1 byte pitch, 1 byte after-touch
0xb0-0xbf	parameter	2 - 1 byte parameter number, 1 byte setting
0xc0-0xcf	program	1 byte program selected
0xd0-0xdf	after touch	1 byte channel pressure
0xe0-0xef	pitch wheel	2 bytes gives a 14 bit value

〈表3-2〉 システム・メッセージ

byte	purpose	data bytes
0xf0	system exclusive	variable length
0xf1	time code	variable length
0xf2	song position	2 - 14 bit value
0xf3	song select	1 - song number
0xf4	undefined	
0xf5	undefined	
0xf6	tune request	0
0xf7	EOX (terminator)	0

〈表3-3〉 リアルタイム・メッセージ

0xf8	timing clock
0xf9	undefined
0xfa	start
0xfb	continue
0xfc	stop
0xfd	undefined
0xfe	active sensing
0xff	system reset

〈表3-4〉 ノート・オンとノート・オフ

0x80-0x8f	note off	2 - 1 byte pitch, followed by 1 byte velocity
0x90-0x9f	note on	2 - 1 byte pitch, followed by 1 byte velocity

〈表3-5〉 ノート・ナンバ

Octave#	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

〈表3-7〉 チャネル・モード・メッセージ

Status D7----D0	Data Byte(s) D7----D0	Description
1011nnnn	0ccccccc 0vvvvvvv	ローカル・コントロール （音源が独立するかどうかのスイッチ） c = 122, v = 0 : Local Control Off c = 122, v = 127 : Local Control On
		オール・ノート・オフ（非常用発音停止） c =123, v =0: All Notes Off
		発音モードの設定（詳細は省略） c=124, v=0: Omni Mode Off c=125, v=0: Omni Mode On c=126, v=M: Mono Mode On (Poly Off) c=127, v=0: Poly Mode On (Mono Off)

〈表3-6〉 チャネル・ボイス・メッセージ

Status D7----D0	Data Byte(s) D7----D0	Description
1000cccc	0nnnnnnn 0vvvvvvv	ノート・オフ (nnnnnnn) is the note number. (vvvvvvv) is the velocity.
1001cccc	0nnnnnnn 0vvvvvvv	ノート・オン (nnnnnnn) is the note number. (vvvvvvv) is the velocity.
1010cccc	0nnnnnnn 0vvvvvvv (nnnnnnn) (vvvvvvv)	ポリフォニックのアフター・タッチ 鍵盤を弾いたあとで押し込む圧力） (ccccccc) is the note number. (vvvvvvv) is the pressure.
1011cccc	0ccccccc 0vvvvvvv	コントロール・チェンジ （各種のコントローラの情報［後述］） (ccccccc) is the controller number. (vvvvvvv) is the new value.
1100cccc	0ppppppp	プログラム・チェンジ (ppppppp) is the new program number.
1101nnnn	0ccccccc	アフター・タッチ (ccccccc) is the channel number.
1110nnnn	01111111 0mmmmmmm	ピッチ・ベンド （14ビット形式での微調整データ） (111111) are the least significant 7 bits. (mmmmmm) are the most significant 7 bits.

オクターブの表現 どのオクターブ音域の音階であるかということ．楽器ごとに異なる．

というメッセージでは，2バイトのデータ幅を使って14ビット精度の表現を伝達するようになっています．

各種のパラメータを転送するための「コントロール・チェンジ」というメッセージの部分には，MIDI演奏する音源の動作モードを規定するなど複雑な規定があります．

また，多量の音楽演奏情報を高速に伝達するための工夫として，「ランニング・ステータス」という規定があります．これは，「同じステータス・バイトに続くデータ・バイトの組であれば，ステータス・バイトを省略して送信してもよい」というものです．送信側は省略してもしなくてもよいのですから，受信側はどちらであっても確実に判定して処理しなければなりません．

MIDIの受信というのは，たとえば「発音開始」を受けて音を鳴らしたのに「発音終了」を正しく解釈できなければ，「音が鳴り続いて止まらない」という，音楽にとって致命的な誤動作となりますから，送信側よりも受信側のほうが重大な責任があるのです．

ステータス・バイトの上位4ビットがすべて“1”，つまり「f*」となっている部分には，さらに**表3-2,表3-3**のような「システム・メッセージ」や「リアルタイム・メッセージ」と呼ばれる複雑な広がりをもつグループがあります．

これらの特別なメッセージは，それ以外のボイス・メッセージが完結していない途中に「割り込む」ことが可能です．つまり，受信側は割り込まれた情報があとで再開するということまで保持しておかなければならないのです．

● 「ノート・オン」と「ノート・オフ」

ボイス・メッセージのうち，**表3-4**で表現されるものが，MIDIによる音楽演奏情報のもっとも基本となるものです．これは「ノート・オン」と「ノート・オフ」で，2バイト目はそのpitch（音高）で，ノート・ナンバと言い，**表3-5**のように規定されています．

いわゆるピアノの「中央C」というのが10進で60となりますが，楽器というのは音色によっても音域によってもオクターブの表現は異なりますから，実際の周波数としてかならずしも同じになるとは限りません．

また，3バイト目のデータは個々の発音の音量に相当する「ベロシティ」という値で，鍵盤楽器の鍵盤を打鍵した速度から検出するので，この名前が付いています．「ノート・オフ」におけるベロシティは，「鍵盤から指を離す速度」とでもいうべきものですが，これに対応した電子楽器はほとんど存在していません．

そして，ランニング・ステータスのルールで情報を効率よく転送するために，「ノート・オフという情報には［ノート・オンでベロシティがゼロ］という表現を用いる」ことが，きわめて一般的に行われています．たとえば，普通に「中央Cを強度64で弾いて，その後発音を停止した」という演奏情報の場合，MIDIチャネルを第1チャネルすなわち「0」とすると，

```
90 3C 64 ― 80 3C 64
```

というメッセージとなります．ところがランニング・ステータスを使うと，

```
90 3C 64 ― 3C 00
```

と，データ量が少なくて済むわけです．受信側では，この両方について正しく対応しなければなりません．

● チャネル・ボイス・メッセージ

ここで，MIDIメッセージの定義を詳細にまとめると，つぎのようになります．ここでは，ステータス・バイトの下4ビットの「cccc」がMIDIチャネルです．

まず，個々のMIDIチャネルごとの情報である，「チャネル・ボイス・メッセージ」の部分です．ただし，個々の詳しい内容については省略します（**表3-6**）．

表3-7は，**表3-6**の「コントロール・チェンジ」の一部を使って特別に定義されている「チャネル・モード・メッセージ」の部分です．

● コントロール・チェンジ

上で述べた「B*」で始まるコントロール・チェンジについては，2バイト目で指定される「コントロール番号」によって，**表3-8**のような規定があります．

● システム・メッセージ

つぎは，「システム・コモン・メッセージ」の部分です．

まず最初は，メーカごとにまったく独自の情報規約を設けてよいという「エクスクルーシブ・メッセージ」です（**表3-9**）．これは，最初が「F0」，次のデータがメーカごとに指定されている「ID」になり，あとは各社が全体のデータ長まで含めてまったく独自に定義してよいというものです．このデータは最後が「F7」で終わることになっていますから，基本的にはMIDIの受信側では「F0からF7までは無視する」というルールで対応します．

基本的にメーカ独自に規定するというはずのエクスクルーシブの例外として，サンプラ音源のサンプリング波形データをこのエクスクルーシブの部分を使ってダンプするという以下の規定があります．

まず，ヘッダ部分では**表3-10**のような情報を送ります．

そして，これに続いてサンプリング波形データのパケットを，**表3-11**のように送ります．パケットは127

ID　MIDIエクスクルーシブ・メッセージ中に埋め込まれている，MIDI協議会に登録されたメーカごとの識別番号．

〈表3-8〉 コントロール・チェンジ

2nd Byte Value			Function	3rd Byte	
Binary	Hex	Dec		Value	Use
00000000=	00=	0	Continuous controller #0	0-127	MSB
00000001=	01=	1	Modulation wheel	0-127	MSB
00000010=	02=	2	Breath control	0-127	MSB
00000011=	03=	3	Continuous controller #3	0-127	MSB
00000100=	04=	4	Foot controller	0-127	MSB
00000101=	05=	5	Portamento time	0-127	MSB
00000110=	06=	6	Data Entry	0-127	MSB
00000111=	07=	7	Main Volume	0-127	MSB
00001000=	08=	8	Continuous controller #8	0-127	MSB
00001001=	09=	9	Continuous controller #9	0-127	MSB
00001010=	0A=	10	Continuous controller #10	0-127	MSB
00001011=	0B=	11	Continuous controller #11	0-127	MSB
00001100=	0C=	12	Continuous controller #12	0-127	MSB
00001101=	0D=	13	Continuous controller #13	0-127	MSB
00001110=	0E=	14	Continuous controller #14	0-127	MSB
00001111=	0F=	15	Continuous controller #15	0-127	MSB
00010000=	10=	16	Continuous controller #16	0-127	MSB
00010001=	11=	17	Continuous controller #17	0-127	MSB
00010010=	12=	18	Continuous controller #18	0-127	MSB
00010011=	13=	19	Continuous controller #19	0-127	MSB
00010100=	14=	20	Continuous controller #20	0-127	MSB
00010101=	15=	21	Continuous controller #21	0-127	MSB
00010110=	16=	22	Continuous controller #22	0-127	MSB
00010111=	17=	23	Continuous controller #23	0-127	MSB
00011000=	18=	24	Continuous controller #24	0-127	MSB
00011001=	19=	25	Continuous controller #25	0-127	MSB
00011010=	1A=	26	Continuous controller #26	0-127	MSB
00011011=	1B=	27	Continuous controller #27	0-127	MSB
00011100=	1C=	28	Continuous controller #28	0-127	MSB
00011101=	1D=	29	Continuous controller #29	0-127	MSB
00011110=	1E=	30	Continuous controller #30	0-127	MSB
00011111=	1F=	31	Continuous controller #31	0-127	MSB
00100000=	20=	32	Continuous controller #0	0-127	LSB
00100001=	21=	33	Modulation wheel	0-127	LSB
00100010=	22=	34	Breath control	0-127	LSB
00100011=	23=	35	Continuous controller #3	0-127	LSB
00100100=	24=	36	Foot controller	0-127	LSB
00100101=	25=	37	Portamento time	0-127	LSB
00100110=	26=	38	Data entry	0-127	LSB
00100111=	27=	39	Main volume	0-127	LSB

2nd Byte Value			Function	3rd Byte	
Binary	Hex	Dec		Value	Use
00101000=	28=	40	Continuous controller #8	0-127	LSB
00101001=	29=	41	Continuous controller #9	0-127	LSB
00101010=	2A=	42	Continuous controller #10	0-127	LSB
00101011=	2B=	43	Continuous controller #11	0-127	LSB
00101100=	2C=	44	Continuous controller #12	0-127	LSB
00101101=	2D=	45	Continuous controller #13	0-127	LSB
00101110=	2E=	46	Continuous controller #14	0-127	LSB
00101111=	2F=	47	Continuous controller #15	0-127	LSB
00110000=	30=	48	Continuous controller #16	0-127	LSB
00110001=	31=	49	Continuous controller #17	0-127	LSB
00110010=	32=	50	Continuous controller #18	0-127	LSB
00110011=	33=	51	Continuous controller #19	0-127	LSB
00110100=	34=	52	Continuous controller #20	0-127	LSB
00110101=	35=	53	Continuous controller #21	0-127	LSB
00110110=	36=	54	Continuous controller #22	0-127	LSB
00110111=	37=	55	Continuous controller #23	0-127	LSB
00111000=	38=	56	Continuous controller #24	0-127	LSB
00111001=	39=	57	Continuous controller #25	0-127	LSB
00111010=	3A=	58	Continuous controller #26	0-127	LSB
00111011=	3B=	59	Continuous controller #27	0-127	LSB
00111100=	3C=	60	Continuous controller #28	0-127	LSB
00111101=	3D=	61	Continuous controller #29	0-127	LSB
00111110=	3E=	62	Continuous controller #30	0-127	LSB
00111111=	3F=	63	Continuous controller #31	0-127	LSB
01000000=	40=	64	Damper pedal on/off (Sustain)	0=off	127=on
01000001=	41=	65	Portamento on/off	0=off	127=on
01000010=	42=	66	Sustenuto on/off	0=off	127=on
01000011=	43=	67	Soft pedal on/off	0=off	127=on
01000100=	44=	68	Undefined		?
			(この間はまだ未定義)		
01011111=	5F=	95	Undefined		?
01100000=	60=	96	Data entry +1		127
01100001=	61=	97	Data entry −1		127
01100010=	62=	98	Undefined		?
			(この間はまだ未定義)		
01111001=	79=	121	Undefined		?

バイトずつになります．それぞれにチェック・サムをとって信頼性を上げています．

また，コンピュータとサンプラとがハンドシェイクによってダンプするための通信プロトコルも，以下のように規定されています．

まず，ダンプをリクエストするには，**表3-12**を送ります．ハンドシェイクのメッセージとしては，

```
cc = channel number
pp = packet number
```

という定義に対して，**表3-13**が定義されていて，処理能力の異なる機器の間でも同期して通信できます．ここでは，ハンドシェイクの詳しい通信プロトコルの規定については省略します．

つぎに「F1」で始まるのは，「タイム・コード」の領域です．これは，MIDI機器と映像機器との同期をとるために定義されたもので，マルチメディアのスタジオでは一般的に利用されています．

タイム・コードの規定には，いくつかの種類があります．

まず「クォータ・フレーム・メッセージ」の場合に

通信プロトコル 通信に関する信号形式，データ形式，情報の表現形式などの規約の全般．

〈表3-9〉 エクスクルーシブ・メッセージ

Status D7----D0	Data Byte(s) D7----D0	Description
11110000	0iiiiiii 0ddddddd 0ddddddd 11110111	System Exclusive. This message makes up for all that MIDI doesn't support. (iiiiiii) is a seven bit Manufacturer's I.D. code. If the synthesizer recognizes the I.D. code as its own, it will listen to the rest of the message (ddddddd). Otherwise, the message will be ignored. System Exclusive is used to send bulk dumps such as patch parameters and other non-spec data.

〈表3-10〉 ヘッダ部情報

F0 7E cc 01 ss ss ee ff ff ff gg gg gg hh hh hh ii ii ii jj F7	
cc	channel number
ss ss	sample number (LSB first)
ee	sample format (number of significant bits; 8->28)
ff ff ff	sample period (1/sample rate) in nanoseconds (LSB first)
gg gg gg	sample length, in words
hh hh hh	sustain loop start point (word number) (LSB first)
ii ii ii	sustain loop end point (word number) (LSB first)
jj	loop type (00:forwards only; 01:alternating)

〈表3-11〉 サンプリング波形データのパケット

F0 7E cc 02 kk <120 bytes> mm F7	
cc	channel number
kk	running packet count (00->7F)
mm	checksum (XOR of 7E, cc, 02, kk <120 bytes>)

〈表3-12〉 ダンプのリクエスト

F0 7E cc 03 ss ss F7	
cc	channel number
ss ss	sample number requested (LSB first)

〈表3-13〉 ハンドシェイクの定義

ACK	F0 7E cc 7F pp F7
NAK	F0 7E cc 7E pp F7
CANCEL	F0 7E cc 7D pp F7
WAIT	F0 7E cc 7C pp F7

〈表3-14〉 クォータ・フレーム・メッセージ

F1 0nnn dddd
dddd = 4 bits of binary data for this Message Type
nnn = Message Type: 0 = Frame count LS nibble 1 = Frame count MS nibble 2 = Seconds count LS nibble 3 = Seconds count MS nibble 4 = Minutes count LS nibble 5 = Minutes count MS nibble 6 = Hours count LS nibble 7 = Hours count MS nibble and SMPTE Type

は，**表3-14**という2バイトで定義します．ここでは，映像機器に関する個々の用語の説明は省略します．上記のデータのさらに詳しい形式としては，**表3-15**などと決められています．そして，「フル・メッセージ」としての定義としては，**表3-16**というように10バイト形式の表現も規定されています．また，この他にユーザ独自のタイムコードを規定できる形式も定義されていますが，ここでは省略します．

また，SMPTEタイムコードに準拠した映像機器のセッティングに関するメッセージも，**表3-17**のように規定されています．ここでも，それぞれの用語ごとの定義については省略します．

この他のシステム・コモン・メッセージとしては，**表3-18**のものがあります．

そして最後に，時間的同期のための「システム・リアルタイム・メッセージ」として，**表3-19**のものがあります．これはすべて1バイトで，規約の上ではサンプル・ダンプを含むエクスクルーシブ・メッセージの中にでも，突然に埋め込まれても正しく認識されるべきであると規定されています．

〈表3-15〉 詳しいクォータ・フレーム・メッセージ

RAME COUNT: xxx yyyyy
xxx = undefined and reserved for future use. Transmitter must set these bits to 0 and receiver should ignore! yyyyy = Frame number (0-29)
SECONDS COUNT: xx yyyyyy
xx = undefined and reserved for future use. Transmitter must set these bits to 0 and receiver should ignore! yyyyyy = Seconds Count (0-59)
MINUTES COUNT: xx yyyyyy
xx = undefined and reserved for future use. Transmitter must set these bits to 0 and receiver should ignore! yyyyyy = Minutes Count (0-59)
HOURS COUNT: x yy zzzzz
x = undefined and reserved for future use. Transmitter must set this bit to 0 and receiver should ignore! yy = Time Code Type: 0 = 24 Frames/Second 1 = 25 Frames/Second 2 = 30 Frames/Second (Drop-Frame) 3 = 30 Frames/Second (Non-Drop) zzzzz = Hours Count (0-23)

〈表3-16〉 フル・メッセージ

F0 7F <chan> 01 <sub-ID 2> hr mn sc fr F7
F0 7F = Real Time Universal System Exclusive Header <chan> = 7F (message intended for entire system) 01 = <sub-ID 1>, 'MIDI Time Code' <sub-ID 2> = 01, Full Time Code Message hr = hours and type: 0 yy zzzzz yy = type: 00 = 24 Frames/Second 01 = 25 Frames/Second 10 = 30 Frames/Second (drop frame) 11 = 30 Frames/Second (non-drop frame) zzzzz = Hours (00->23) mn = Minutes (00->59) sc = Seconds (00->59) fr = Frames (00->29) F7 = EOX

〈表3-17〉 SMPTEタイムコード

F0 7E <chan> 04 <sub-ID 2> hr mn sc fr ff sl sm <add. info> F7
F0 7E = Non-Real Time Universal System Exclusive Header <chan> = Channel number 04 = <sub-ID 1>, MIDI Time Code <sub-ID 2> = Set-Up Type 00 = Special 01 = Punch In points 02 = Punch Out points 03 = Delete Punch In point 04 = Delete Punch Out point 05 = Event Start points 06 = Event Stop points 07 = Event Start points with additional info. 08 = Event Stop points with additional info. 09 = Delete Event Start point 0A = Delete Event Stop point 0B = Cue points 0C = Cue points with additional info 0D = Delete Cue point 0E = Event Name in additional info hr = hours and type: 0 yy zzzzz yy = type: 00 = 24 Frames/Second 01 = 25 Frames/Second 10 = 30 Frames/Second drop frame 11 = 30 Frames/Second non-drop frame zzzzz = Hours (00-23) mn = Minutes (00-59) sc = Seconds (00-59) fr = Frames (00-29) ff = Fractional Frames (00-99) sl, sm = Event Number (LSB first) <add. info.> F7 = EOX

〈表3-18〉 システム・コモン・メッセージ

11110010	01111111 0mmmmmmm	ソング・ポジション・ポインタ (曲データのどの部分かを指定)
11110011	0sssssss	ソング・セレクト
11110110		チューニングのリクエスト
11110111		エクスクルーシブの終わり

〈表3-19〉 システム・リアルタイム・メッセージ

11111000	タイミング・クロック
11111010	シーケンサのスタート
11111011	シーケンサの続行
11111100	シーケンサのストップ
11111110	MIDIケーブルの断線チェック
11111111	システム・リセット (一般には使ってはいけない)

マルチメディア テキスト情報，視覚的情報，聴覚的情報などを統合したメディアとして利用することの総称．

スタンダードMIDIファイル規格について

MIDI規格とSMF規格

MIDI規格とは，基本的に「ある音」「ある制御」「あるイベント」という個別の情報までしか規定していませんから，「MIDI情報が時間的に次々に送られる」という「楽曲」の情報はいっさい定義できません．そこで規定されたのが，「標準MIDIファイル」というファイル形式で，この定義にしたがったデータであれば，どのようなシーケンサ（自動演奏）ソフトでも装置でも，同じようなMIDI演奏を「再生」できるものです．

ここには，「時間をどう定義するか」「曲の構造をどう定義するか」「ファイルとしてどうデータをまとめるか」などの複雑な検討が必要となります．SMFはあくまで「自動演奏として再生する」という目的のためのものである，ということになります．

概念と用語の定義

ここでは，SMFで必要ないくつかの用語について簡単に定義しておきます．

● ［Sequence］シーケンス

時間的情報とともに一連のMIDI情報を並べたデータのことで，このデータにしたがって自動再生すると元の音楽演奏が再現されます．たとえば，楽譜で「一つの音符」と記述されている音楽演奏情報は，「ノート・オン，一定の時間，ノート・オフ」という三つの情報のシーケンスと等価というわけです．

● ［Track］トラック

楽曲というのは単一のパートから構成されていることはほとんどなく，メロディと伴奏，あるいはピアノとベースとドラム，あるいはオーケストラのように，複数のパートが同時に演奏されます．そこでこの各パートのかたまりを，トラックという概念で複数にわたって処理します．

● ［Chunk］チャンク

SMFのファイル構造を定義するために，シーケンスを構成する要素として「チャンク」という概念を用いています．シーケンス・ファイルは，ヘッダのチャンクと，複数のトラックに応じたデータをもつトラック・チャンクから，標準化された構造で構成されています(**表3-20**)．ここで「MThd」とか「MTrk」という4バイトのキャラクタ・データは，予約されたSMF独自の規定で，これに続くデータ長は32ビットと決められています．

このデータ長により，それぞれのチャンクのデータは必要に応じた可変長データとなります．このあたりはJavaのバイト・コードでも踏襲されています．

● ［デルタタイム］

MIDI規約では，「それぞれのMIDIイベントの間の相対時間」あるいは「起点から各MIDIイベントまでの絶対時間」という，MIDIイベントの集合から実際の楽曲を構成するための「時間」の概念がありません．そこでSMFでは，「次のMIDIイベントまでの相対時間」を定義するために，デルタタイムという概念を使用します．

これは，システムに固有のペースで刻まれるクロック数でいくつというカウント値として使用します．

時間データは，「同時」を表す最小クロックから「数分」以上にまで幅広いダイナミック・レンジをもっているために，あるビット幅のデータとして定義すると，小さい時間に対して無駄があまりにも大きいために，以下のような可変長形式の定義を用いています．

デルタタイムは，まず相対時間差データが7ビットで表現できる場合には，そのまま1バイトで済みます．そしてこれを超える場合には，MSBを繰り上げた形式で，7ビットだけを使用してデータを表現します．表現できる相対時間差の最大データは「0FFFFFFF」ですが，これは最小単位クロックが96分音符として「毎分500」という超高速指定でも「4日」にあたるので，現実的な音楽データとしては十分です．

デルタタイムと相対時間差データとの対応は，**表3-21**のようになります．

● トラック・データの形式

トラック・データの定義も構造的に標準化されています．シンタックスとしては**表3-22**のようになっています．

つまり，トラック・データは「トラック・イベント」が連結されて構成され，トラック・イベントは「デルタタイム」と「イベント」から成ります．

イベントは，「MIDIイベント」「エクスクルーシブ・イベント」「メタイベント」のいずれかから成ります．

SMF最小単位クロック　SMF形式のMIDI演奏データを再生演奏する際に，この最小単位クロックを何カウントしたら次のデータを再生するかということで音楽演奏のテンポを表現している．

〈表3-20〉
チャンクの構成

```
MThd  <length of header data>
<header data>
MTrk  <length of track data>
<track data>
MTrk  <length of track data>
<track data>
 ...
```

〈表3-22〉 トラック・データの定義

```
<track data> = <MTrk event>+

<MTrk event> = <delta-time> <event>

<event> = <MIDI event> | <sysex event> | <meta-event>
```

〈表3-23〉 エクスクルーシブ・イベント

```
F0 <length> <bytes to be transmitted after F0>
F7 <length> <all bytes to be transmitted>
```

〈表3-24〉 メタイベント

FF 00 02 ssss	Sequence Number
FF 01 len text	Text Event
FF 02 len text	Copyright Notice
FF 03 len text	Sequence/Track Name
FF 04 len text	Instrument Name
FF 05 len text	Lyric
FF 06 len text	Marker
FF 07 len text	Cue Point
FF 2F 00	End of Track
FF 51 03 tttttt	Set Tempo
FF 54 05 hr mn se fr ff	SMPTE Offset
FF 58 04 nn dd cc bb	Time Signature
FF 59 02 sf mi	Key Signature sf = −7: 7 flats sf = −1: 1 flat sf = 0: key of C sf = 1: 1 sharp sf = 7: 7 sharps mi = 0: major key mi = 1: minor key
FF 7F len data	Sequencer-Specific Meta-Event

「MIDIイベント」は，MIDI規格で規定されている「生の」情報そのものとなっています．つまり，SMFはMIDI規格の上にできているわけです．

デルタタイムで切り離されたイベントをまたいでランニング・ステータスのルールも有効ですから，このMIDIイベントの部分にはMSBの立たないデータだけがある，という場合もあります．

「エクスクルーシブ・イベント」の部分では表3-23のような形式で，MIDIのエクスクルーシブで規定するデータ・パケットを定義することができます．

「メタイベント」の部分は，

〈表3-21〉
デルタタイムと相対時間差データとの対応（16進表記）

相対時間差	SMFでのデルタタイム表現
00000000	00
00000040	40
0000007F	7F
00000080	81 00
00002000	C0 00
00003FFF	FF 7F
00004000	81 80 00
00100000	C0 80 00
001FFFFF	FF FF 7F
00200000	81 80 80 00
08000000	C0 80 80 00
0FFFFFFF	FF FF FF 7F

```
FF <type> <length> <bytes>
```

という形式で，MIDIで表現できないイベント情報を扱います．たとえば，表3-24のようなものが定義されていますが，ここでは個々の内容の詳細については省略します（楽譜のルール［楽典］を知っている人にとっては必須の概念ばかりです）．

● ヘッダーチャンク

ヘッダーチャンクは楽曲データ全体を表現するための情報ブロックで，

```
<chunk type> <length> <format> <ntrks> <division>
```

という形式をしています．最初のデータのチャンク・タイプには，

- ▶ 0：マルチチャネルのトラックがただ一つだけある
- ▶ 1：同時に演奏される複数のトラックがある
- ▶ 2：別々の複数のトラックがある（メドレーのようなイメージ）

の3種類がありますが，インターネットなどで一般的に使われるのは「フォーマット0」のタイプのSMFがほとんどです．これ以下のデータについては，本書の内容とも関係のない細かい内容なので省略します．

第4章　ROMエミュレータの開発

ターゲット・システムをプリンタ・ポートに接続してデバッグする

本章では，マイコン・システムの本格的な理解，とくにディジタル回路で構成されるハードウェアを実際に確認し，ソフトウェアとの結びつきを検証するための題材として，AKI-80システムの開発で重宝する「ROMエミュレータ」というツールを製作します．

ROMエミュレータの原理と構成

もっとも基礎の確認として，**図2-1**(p10)のCPUシステムの中の「プログラムROM」の部分に注目してみます．一般にROM(ここでは256KビットEPROMの27256)のピン配置と内部構成は，**図4-1**のようになっています．最終的に完成したプログラムは，このROMチップに書き込まれてAKI-80のROMソケットに挿入します．AKI-80上のCPUは，必要な命令を引き出すためにアドレス・バスに所定のアドレス信号を乗せると，ROMからは該当するアドレスに記憶されたデータがデータ・バスに出力されて，CPUはこれを命令として獲得(フェッチ)するという繰り返しです．

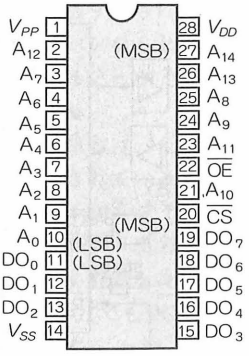
さて，そこでROMエミュレータとは，「CPUに対して，ROMソケットにプログラムROMが正常に挿入されていると思わせる」というのが第一の機能となります．つまり，CPUが自分のタイミングでアドレスを出すと，データ・バスにそのメモリ内容が出力されればいいわけです．

これはメモリがROMでなくRAMであってもまったくかまわないことです．そしてもう一方で，「ソフト開発パソコンからは，簡単にメモリ内容を書き換えられる」という要請があります．いちいち実験途中の未完成プログラムを時間のかかるROMライタでEPROMに焼きたくないのでROMエミュレータを使うのですから，メモリは**図4-2**のようなRAM(ここでは256KビットSRAMの84256)を使うことになります．EPROMとRAMの読み出し時間はほぼ同程度ですが，書き込み時間がおよそ1000倍から10000倍ほどRAMのほうが高速だからです．

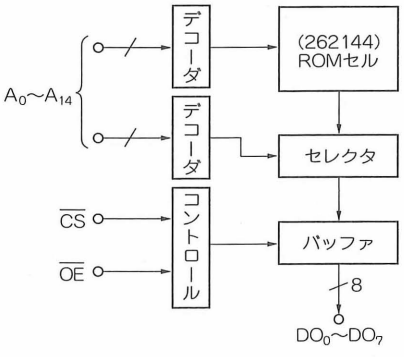
そして重要なのが，これらの「開発パソコンからのプログラムの書き込み(ダウンロード)」と「ターゲット・マシンでの実行(読み出し)」とは，絶対に同時には行われないという点です．プログラムを実行中にその一部が書き換えられてしまっては，CPUは暴走など予期せぬ動作を起こしますから，「ダウンロード中はCPUからのアクセスを禁止して，ダウンロード後にまずCPUをリセットする」という基本的な設計方針をとることにします．これは，DSPでは「動作しながら同時にプログラムを改訂する」というテクニックがあるのに対して，ずいぶんシステム設計がシンプルで済むことになり，助かります．

このように，ROMエミュレータの基本的原理の部

〈図4-1〉EPROM 27256のピン配置と内部構成



(a) ピン配置



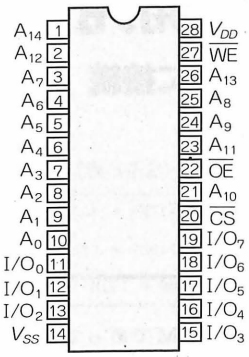
(b) ブロック図

入力		DO	動作
$\overline{CS}$	$\overline{OE}$		
H	X	High-Z	Stand by
L	H	High-Z	Operating
L	L	DO	Operating

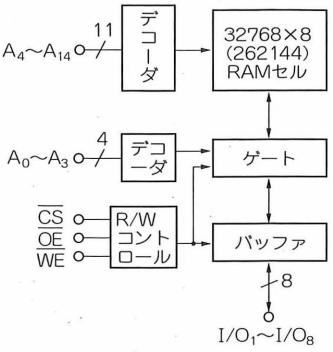
(c) 動作表

フェッチ　コンピュータが命令をメモリから取り出すこと．フェッチとその実行で一組のサイクルとなる．

〈図4-2〉SRAM 84256のピン配置と内部構成



(a) ピン配置



(b) ブロック図

入力			I/O		動作
$\overline{CS}$	$\overline{WE}$	$\overline{OE}$	DI	DO	
H	X	X	X	High-Z	非選択
L	H	L	High-Z	DO	Read
L	L	H	DI	High-Z	Write
L	L	L	DI	High-Z	Write
L	H	H	X	High-Z	選択

(c) 動作表

分だけを取り出してみたのが**図4-3**のブロック図です．

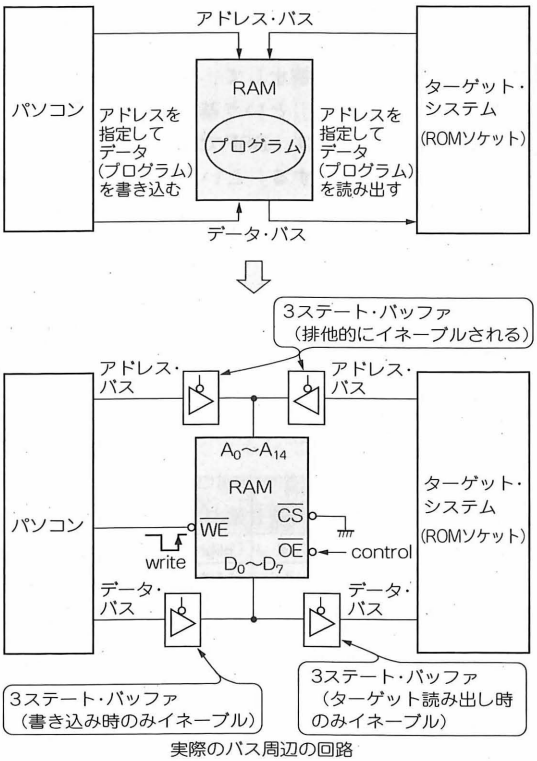
パソコンからは「何らかの方法で」RAMのためのアドレスとデータの信号を与えて，一方でターゲット・システムのROMソケットに挿入されるROMプローブ(ROMソケット内の全ピンの信号を外部に引き出すケーブルの付いた挿入ソケット)には，アドレスを受けてデータ・バスに出力する機構が必要になります．実際には，バス・ラインの信号と同期したチップ・セレクト($\overline{CS}$)と出力イネーブル($\overline{OE}$)に対する制御，RAMに書き込むためのライト・イネーブル($\overline{WE}$)，そしてターゲットCPUをリセットするためのリセット出力がさらに必要です．

図4-4は，これら動作に必要なタイミング信号の動作を示したものです．

プリンタ・ポート利用のROMエミュレータの製作

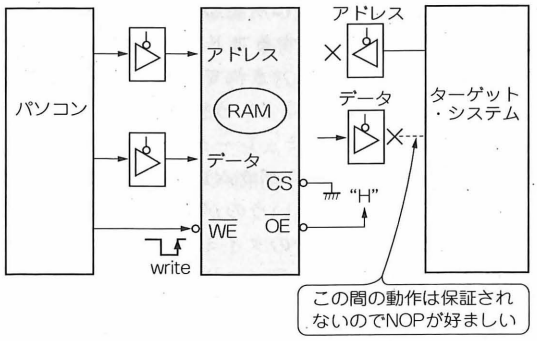
ROMエミュレータの基本機能からわかることは，この装置そのものはCPUをもたない，非インテリジェ

〈図4-3〉ROMエミュレータの構成

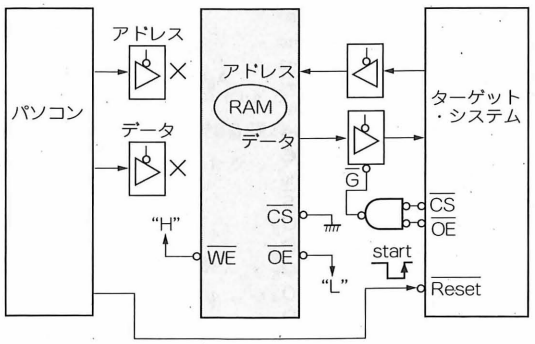


実際のバス周辺の回路

〈図4-4〉ROMエミュレータの動作と制御



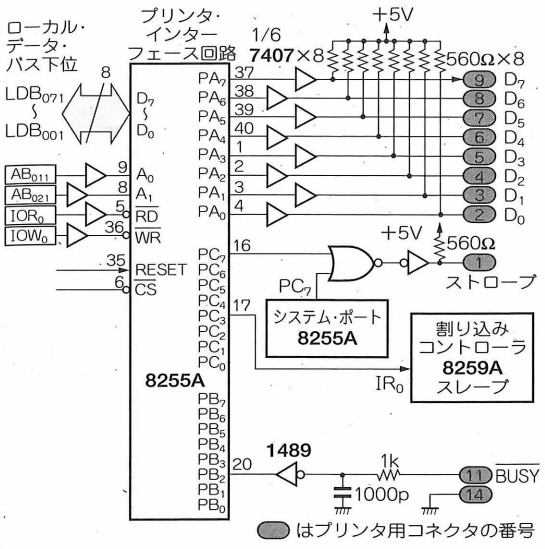
(a) パソコンからRAMにプログラムをダウンロード



(b) ターゲット・システムがRAMをプログラムROMとしてアクセス

ROMプローブ　マイコン・システムでプログラムの書かれたEPROMを挿入するROMソケットに差し込んで，ROMエミュレータに接続するコネクタ部分のこと．

〈図4-5〉プリンタ・インターフェース回路



ントなシステムで十分であるということです．もちろん，市販の製品にはCPUを搭載して，ホストのパソコンとシリアル通信をしたり，ターゲット・システムの動作を監視したりするものもありますが，本書の方針からはオーバスペックでしょう．そこで，AKI-80システムの製作に慣れるための実験としては，非インテリジェント型ROMエミュレータを製作していくことにします．つまり，インテリジェントな部分は，すべてホスト・パソコンのソフトウェア側で吸収することになります．

図4-3からわかることは，ターゲット・システムに対しては**ROMプローブ**を経由して接続するので，どうしてもほぼROMソケットのピンと同数の信号ケーブルが必要となります．ところで，開発パソコンとの接続の部分については，ここにも多数のケーブルを走らせるというのはあまり美しくありません．とくに，アドレス15ビットにデータ8ビット，そして制御ラインがさらに数本という，パソコンの外部ポートをこのために増設するのは，拡張スロットのあるデスクトップ・パソコンならともかく，筆者もフィールドに持ち歩いているノート・パソコンでは，あまりに一般性がありません．

そこで筆者は，どんなパソコンでももっていて，規格が統一されている「プリンタ・ポート」に注目しました．AKI-80システムを開発しているときにわざわざ同時にプリント出力することもありませんから，これはなかなか有効なコンピュータ資源の活用であり，さらにPC9801でもPC互換機でも共通に使えるテクニックなのです．

図4-5はPC9801のプリンタ・ポートの内部回路です．コネクタの形状や内部回路はともかく，プリンタ端子の外部仕様というのは世界共通です．したがって，ここで製作するROMエミュレータはほんのわずかの変更で，PC互換機でも，FM-Townsでも使えるものとなっています．筆者はこのROMエミュレータをまったくそのままで，WindowsマシンであるDynaBookでも使っています．

パソコンのプリンタ信号というのは，「セントロニクス仕様」という規格に従っていて，PC9801の場合にはとくに簡易型の仕様となっています．つまりパソコンから見て，データ・ラインとして8ビットの出力ライン，このデータをラッチするための1ビットの「ストローブ」出力信号，さらにプリンタの状態を見るための1ビットの「ビジー」入力信号があり，あとはGNDだけで，電源ラインは外部出力されていません．

さて，ここからはパズルとなります．たった9本の出力信号しかないプリンタ・ポートから，8ビット幅のデータを32768種類のアドレスに対して別個に設定して，さらに上記のようなシステムの制御も実現するにはどのような回路とすればよいでしょうか．ある程度，腕に自信のある方は，ここでちょっと時間をとって，ROMエミュレータの回路を設計するという設問に挑戦してみましょう．

実は，筆者は今回の製作ではプリンタ・ポートのデータとして8ビットの全部を使わずに，7ビットだけとしています．1ビット減で半分の情報量になりますから，そのくらいのハンディがあっても実現できるのです．

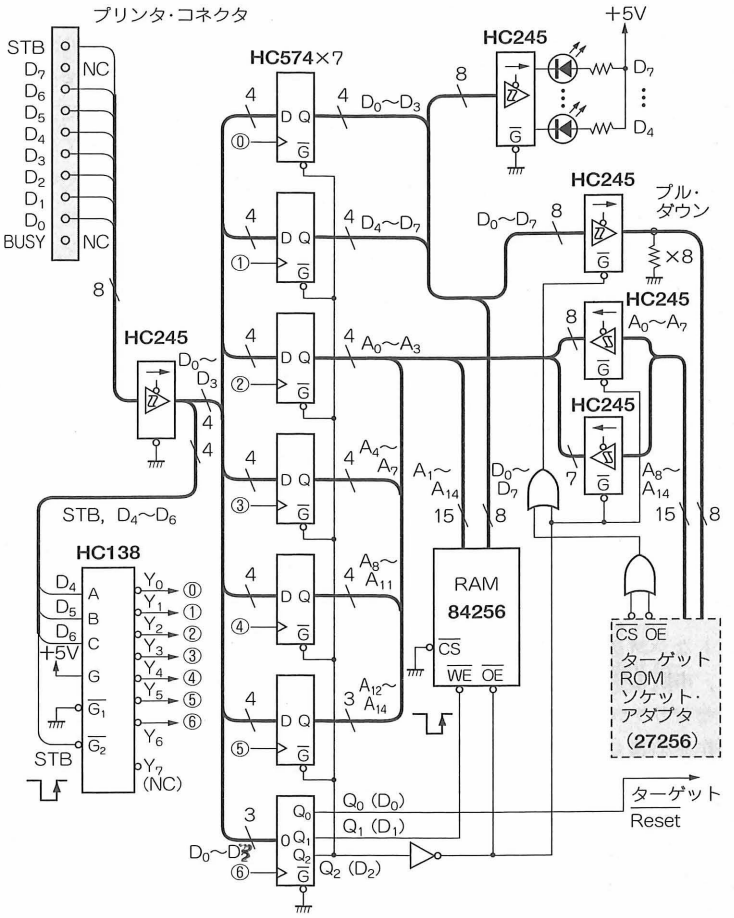
また，本書ではふれませんが，ROMエミュレータでなく「ICカード読み書き装置」として，同様にプリンタ・ポート(入力がたったの1ビットしかない)を使って，64KバイトまでのICカードを2枚差し込んで，その内容をパソコンから読み書きしたり，編集，コピーするマシンを製作したこともあります．ディジタル回路の設計課題としては格好の題材なので，興味のある方はこちらも考えてみてください．

図4-6(p.34)が今回製作した「プリンタ・ポート使用のROMエミュレータ」の回路図です．バス信号線をまとめて書いているので，初めての方は戸惑うかもしれませんが，マイコン・システムではふつうの書き方なので，慣れてください．プリンタ・コネクタにはデータのビット7とビジー入力の1ビットは接続していません．そこで，Windowsマシンの場合には，ここで「ストローブ」としている信号にビット7をそのまま使うことができます．

パソコンごとの違いは「プリンタ・コネクタのピン番号と信号の対応」だけなのです．筆者の場合，手持ちの98ノートの小型プリンタ・コネクタのピン配置は知りませんでしたから，パソコン側の特殊コネクタ端子と，プリンタ側の端子(こちらはデータブックに載

プルダウン抵抗　バス・ラインの信号がオープン状態でノイズを拾ったりしないように一定の抵抗で全ビットをGNDと接続しておくための抵抗．普通は数kΩから10kΩほど．

〈図4-6〉
プリンタ・ポート使用ROMエミュレータ



っていた)との対応をいちいちテスタで調べて信号の対応表を作り，そのあとでケーブルを切断して配線しました．実際には，切り落としたプリンタ側のコネクタというのはすべて共通ですから，このピン番号とケーブル内のそれぞれの信号線のカラーの対応表を作れば，「どの信号線がどのビットである」ということがわかって簡単に配線できます．

全体をざっと解説しておくと，プリンタ・コネクタのデータ・ラインの7ビットとストローブ信号の計8本は，とりあえずバッファ74HC245で受けてそのまま通過します．ここには，パソコンからプリンタ・ケーブルで延びてきた信号ラインとROMエミュレータ内部とを切り離してノイズなどの誤動作を避ける意味と，ROMエミュレータ内部で多数のICに対する信号を駆動するという二つの意味があります．

データ7ビットのうち，下位の4ビットが内部的に共通信号ラインとして使用されています．これらは内部RAMのデータ・バスの8ビットのために2個，アドレス・バスの15ビットのために4個，計6個のラッチ74HC574に供給され，その出力はRAMのバス・ラインとなっています．74HC574は8ビットですから半分はむだとなっているのですが，このICは出力が3ステートとなっているので，CPUシステムのようなバス・ラインには使いやすいのです．

データ7ビットの上位の3ビットはデコーダ74HC138のアドレスとして使用され，ストローブ信号を74HC138のイネーブルとして利用しています．これで，「データの上位3ビットを設定しつつ，ストローブを“L”にして，また“H”にする」という操作をソフト側で行うことで，該当する74HC138の出力ラインだけが「立ち上がりエッジのラッチ・パルス」を発生することになります．これらは，制御用の74HC574を含む全部で7個の74HC574のラッチ・パルスとなっています．

7個目の74HC574にはデータ7ビットの下位のうち3ビットがラッチされ，これがターゲット・システムとの調停機能を実現しています．データD_0はターゲット・システムのリセット信号，データD_1はRAMのWE信号として独立していますから，通常状態ではかならず“H”になっているようにします．

データD_2が「RAM書き込みか，ターゲットの読み出しか」を決定している信号で，RAMのバスをパソ

該当するI/Oアドレス　I/Oを選択するためのアドレス・デコード回路によってチップ・セレクト信号が与えられて許可されたアドレスのこと．

〈リスト4-1〉プリンタBIOS機能一覧(INT 1Ah)

AHレジスタ	機能
10h	初期化
11h	データの出力
12h	ステータスの取得
30h	データの連続出力

〈リスト4-3〉割り込みベクタの位置

```
BIOS：「INT 1A」→「1Ah」番目のベクタを参照する
                →1ベクタは4バイト(セグメント、アドレスに各2バイト)
                → 1Ah * 4 =「68h」の位置が「INT 1A」のジャンプ先

                アドレス(下位・上位の順)= 7B4Eh
                                ↓ ↓
0000:0060  68 0D 5B 24 34 04 A3 12-4E 7B 60 00 56 7B 60 00  h.[$4.#.N{`.V{`.
                                      ↑ ↑
                                セグメント(下位・上位の順)= 0060h

                →「INT 1A」のジャンプ先は「0060:7B4E」!!
```

〈リスト4-2〉
デバッガでの表示

```
Microsoft Symbolic Debug Utility
Version 3.01
(C)Copyright Microsoft Corp 1984, 1985

Processor is [80286]

-d 0:0
0000:0000  54 53 AF 08 36 09 80 FD-F0 08 80 FD 36 09 80 FD  TS/.6..}p..}6..}
0000:0010  36 09 80 FD 43 77 60 00-77 76 60 00 36 09 80 FD  6..}Cw`.wv`.6..}
0000:0020  3D 06 80 FD AC 0F 5B 24-37 09 80 FD 37 09 80 FD  =..},.[$7..}7..}
0000:0030  6C 05 A3 12 1E 00 00 DC-37 09 80 FD 37 09 80 FD  l.#....\7..}7..}
0000:0040  36 09 80 FD C0 0E 00 F6-1A 0E 00 D8 85 0E 00 D8  6..}@..v...X...X
0000:0050  37 09 80 FD 7D 09 DE 11-37 09 80 FD 37 09 80 FD  7..}}.^.7..}7..}
0000:0060  68 0D 5B 24 34 04 A3 12-4E 7B 60 00 56 7B 60 00  h.[$4.#.N{`.V{`.
0000:0070  00 05 80 FD 36 09 80 FD-EC 27 80 FD 24 00 00 D8  ...}6..}l'.}$..X
```

コン側とターゲットのいずれが使用するかを決定しています．ここでは，ターゲット・システムに異常な信号が出ないように，「$\overline{CS}$信号で許可されるときだけバスに出力する」「バスからの出力を許可しないときには，**プルダウン抵抗**によってデータ00[NOP]を返す」というように設計しています．

図4-6からそのままICのデータブックのピン配置図を見て製作できる人は，かなりのプロといえるかもしれません．じつは筆者の場合，**図4-6**ほど丁寧に書くこともなく，実際に製作するときに書いたのは**図4-4**に相当する制御部分のメモだけでした．ただし，あまりディジタル回路の製作に慣れていない人は，もう少しブレークダウンした回路図を自分で書いてから始めるようにしましょう．実際のはんだ付けは，慣れた人なら半日もあれば製作できてしまいます．

プリンタ・ポートの解析

さて，ROMエミュレータのハードウェアがうまく製作できたとしても，これはまだ「AKI-80システムの開発ツール」にはなっていません．

パソコンのプリンタ・ポートを使っていますが，実際のデータは7ビットを上下に分けて多重化していますから，パソコンの「プリント出力」という機能では役に立ちません．そこで，ROMエミュレータのためのソフトウェア開発の次のステップとして，まずプリンタ・ポート操作の部分を自力で解析しなければなりません．なお，この部分はPC9801を例としていますが，Windowsマシンの場合にも，「**該当するI/Oアドレス**」という情報は公開されています．

パソコンのシステム解析などというとたいへんなことのように思う方もいるかもしれませんが，慣れてしまえばたいしたことではありません．ツールとしてdebug,symdebなどのデバッガを使って，MS-DOSの下層で，ハードウェアと直接やりとりしているBIOSの処理をちょっと追いかけてみるだけのことです．手がかりとなるスタート・ラインとしては，**リスト4-1**のような「パソコン解説本」の「**プリンタBIOS仕様**」が便利です．このような資料がなければ，サンプルとして「プリンタから1文字だけ出力」というプログラムをアセンブラで作成して，これを解析(逆アセンブル)して追いかけていくことが必要となります．

もっとも，**リスト4-1**の記述というのもそっけないものです．ここで重要な情報というのは，「プリンタ処理は[INT　1Ah]というソフトウェア割り込みによって行う」という部分です．ここからはx86系のCPUの特性となってしまいますが，このCPUはソフトウェア割り込みのアドレスをメモリ空間の先頭セグメントのところに置いているという知識を利用すれば，まずは**リスト4-2**のように，デバッガでメモリ空間の先頭の部分を表示するところから始まります．

リスト4-2の例はたまたま筆者の使っているパソコンの情報ですから，実際にはパソコンの機種ごとに異なります．ただし，PC98シリーズに限らず，このように追跡することで，MS-DOSであればどんなパソコンでも通用するのです．**リスト4-2**でダンプしているのは，**セグメント**0のアドレス0の，ソフトウェア割

プリンタBIOS仕様　プリンタに対する処理のためのサービス・ルーチンを格納したパソコン内部のROMに組み込まれた仕様．ROMなので変更できない．

〈リスト4-4〉逆アセンブル結果①

```
-u 0060:7b4e

0060:7B4E FB          STI
0060:7B4F 1E          PUSH    DS
0060:7B50 52          PUSH    DX
0060:7B51 EA990680FD JMP      FD80:0699 → PRINT処理に
                                          無条件ジャンプ
                      ↑
          (セグメントがかなり高位なので、たぶんROM-BIOS領域)
```

〈リスト4-5〉逆アセンブル結果②

```
-u fd80:0699

FD80:0699 E82500      CALL 06C1   → さらに処理ルーチンに行く
FD80:069C 50          PUSH AX
FD80:069D B0FF        MOV  AL,FF
FD80:069F E640        OUT  40,AL
FD80:06A1 33C0        XOR  AX,AX
FD80:06A3 8ED8        MOV  DS,AX
FD80:06A5 FE0E5604    DEC  Byte Ptr [0456]
FD80:06A9 58          POP  AX
```

〈リスト4-8〉AH=10hの初期化ルーチン

AHレジスタ	機能
10h	初期化

```
FD80:0708 B00D      MOV     AL,0D
FD80:070A E637      OUT     37,AL     ←ポート[37h]に何かコマンドを書き込む
FD80:070C EB00      JMP     070E      ←(wait)
FD80:070E EB00      JMP     0710      ←(wait)
FD80:0710 B082      MOV     AL,82
FD80:0712 E646      OUT     46,AL     ← (8255Aのモードをセット)
FD80:0714 EB00      JMP     0716      ←(wait)
FD80:0716 EB00      JMP     0718      ←(wait)
FD80:0718 B00F      MOV     AL,0F
FD80:071A E646      OUT     46,AL     ←ストローブ信号を[High]にする
FD80:071C EB00      JMP     071E      ←(wait)
FD80:071E EB00      JMP     0720      ←(wait)
FD80:0720 B00C      MOV     AL,0C
FD80:0722 E637      OUT     37,AL     ←ポート[37h]に何かコマンドを書き
FD80:0724 EB00      JMP     0726      ←(wait)込む
FD80:0726 EB00      JMP     0728      ←(wait)
FD80:0728 E84300    CALL    076E      ←ステータスをチェックして
FD80:072B EBD9      JMP     0706      ←共通リターン出口へ戻る
```

〈リスト4-10〉AH=12hのステータス入力ルーチン

AHレジスタ	機能
12h	ステータスの取得

```
FD80:076E 86E0      XCHG    AH,AL
FD80:0770 E442      IN      AL,42     ←ステータスはポート[42h]から入手
FD80:0772 EB00      JMP     0774      (wait)
FD80:0774 EB00      JMP     0776      (wait)
FD80:0776 C0E802    SHR     AL,02
FD80:0779 2401      AND     AL,01
FD80:077B 86E0      XCHG    AH,AL     ←AHのLSBに[busy]を格納してリターン
FD80:077D C3        RET
```

り込みルーチンのアドレス(ベクタ)を並べたテーブルです．一つの割り込みベクタは，**リスト4-3**のように2バイトのセグメントと2バイトのアドレス，合計4バイト単位で並んでいますから，「1Ah」のベクタに相当するのは，4倍して「68h」の位置となります．この4バイトを，ルールにしたがって「下位，上位」の順序(あとで検討するJavaの場合には上位，下位の順序なので注意!)で解釈すれば，このジャンプ先は「0060:7B4E」ということになります．

そこで**リスト4-4**のように，この「0060:7B4E」のアドレスから，今度は「u」コマンド**(逆アセンブル)**で表示させてみると，「FD80:0699」というアドレスにさらにジャンプしています．この高いセグメント・アドレスは，MS-DOSパソコンでは「ROM-BIOS」と呼ばれる，システムROMに置かれたサービス・ルーチンのあたりなので，この部分がプリンタ処理に際してハードウェアを実際に叩いているところであると予想されます．この部分をさらに追ってみると，**リスト4-5**のように「FD80:06C1」のサブルーチンをコールしていて，この先が**リスト4-6**のようになっていました．つまり，AHレジスタの内容が30hでなければ，**リスト4-7**の「本命」のルーチンにジャンプするのです．

〈リスト4-6〉逆アセンブル結果③

```
-u fd80:06c1

FD80:06C1 50          PUSH  AX          ------
FD80:06C2 33C0        XOR   AX,AX               |
FD80:06C4 8ED8        MOV   DS,AX               | AXレジスタは
FD80:06C6 FE065604    INC   Byte Ptr [0456]     | そのまま
FD80:06CA 58          POP   AX          <------
FD80:06CB 80FC30      CMP   AH,30
FD80:06CE 7403        JZ    06D3
FD80:06D0 EB1B        JMP   06ED  → AHが30h以外のとき
FD80:06D2 90          NOP           ジャンプ
FD80:06D3 32E4        XOR   AH,AH
```

〈リスト4-7〉逆アセンブル結果④

```
-u fd80:06ed

FD80:06ED 51          PUSH  CX
FD80:06EE 33D2        XOR   DX,DX
FD80:06F0 8EDA        MOV   DS,DX
FD80:06F2 80E40F      AND   AH,0F       ←実際にはAHの下位4ビット
FD80:06F5 7411        JZ    0708        →AH = 10h の処理へ
FD80:06F7 FECC        DEC   AH
FD80:06F9 7432        JZ    072D        →AH = 11h の処理へ
FD80:06FB FECC        DEC   AH
FD80:06FD 7505        JNZ   0704
FD80:06FF E86C00      CALL  076E        →AH = 12h の処理へ
FD80:0702 EB02        JMP   0706
FD80:0704 32E4        XOR   AH,AH       →どれでもないとリターン
FD80:0706 59          POP   CX          ←共通リターン出口
FD80:0707 C3          RET
```

〈リスト4-9〉AH=11hのデータ出力ルーチン

AHレジスタ	機能
11h	データの出力（データはALに入っている）

```
FD80:072D 33C9      XOR     CX,CX
FD80:072F B211      MOV     DL,11
FD80:0731 F606800402 TEST   Byte Ptr [0480],02
FD80:0736 750B      JNZ     0743
FD80:0738 B206      MOV     DL,06
FD80:073A F606010580 TEST   Byte Ptr [0501],80
FD80:073F 7502      JNZ     0743
FD80:0741 B208      MOV     DL,08
FD80:0743 E82800    CALL    076E      ←ステータスOKをチェック
FD80:0746 F6C401    TEST    AH,01
FD80:0749 750B      JNZ     0756      ←ステータスがOKなら処理へ
FD80:074B E2F6      LOOP    0743
FD80:074D FECA      DEC     DL
FD80:074F 75F2      JNZ     0743
FD80:0751 80CC02    OR      AH,02
FD80:0754 EBB0      JMP     0706

FD80:0756 E640      OUT     40,AL     ←ALのデータをポート[40h]に出力
FD80:0758 EB00      JMP     075A      (wait)
FD80:075A EB00      JMP     075C      (wait)
FD80:075C B00E      MOV     AL,0E
FD80:075E E646      OUT     46,AL     ←ストローブを[Low]にする
FD80:0760 EB00      JMP     0762      (wait)
FD80:0762 EB00      JMP     0764      (wait)
FD80:0764 B00F      MOV     AL,0F
FD80:0766 E646      OUT     46,AL     ←ストローブを[High]にする
FD80:0768 EB00      JMP     076A      (wait)
FD80:076A EB00      JMP     076C      (wait)
FD80:076C EB98      JMP     0706      →共通リターン出口へ戻る
```

ここで初めて，**リスト4-1**で定義されていた「AHレジスタで与えるパラメータ」との関係が判明します．つまり，AHレジスタの下位4ビットだけを残してゼロと比較し，さらにデクリメントしながらゼロと比較することで，AHの値ごとに異なる処理に分岐しているわけです．

リスト4-8は「AH=10h」の初期化ルーチン，**リスト4-9**は「AH=11h」のデータ出力ルーチン，**リスト4-10**は「AH=12h」のステータス入力ルーチンの実際の逆アセンブル・リストです．今回のROMエミュレータでは，とりあえず**リスト4-9**から，「データそのものはポート40hに書き込む」という部分と，このデータをラッチ出力するためのストローブとしては「ポート46hのLSB(ビット 0)を下げて上げる」という処理が判明すれば，必要な情報が解析できたことになります．

セグメント　x86系CPUのアドレス指定に特有の概念で，64Kバイトのアドレスごとの単位をセグメントとして，このさらに上位のアドレスとして指定されるアドレスとセグメント単位のアドレスとを加算する．非常に美しくない仕様であり，時代とともに廃れた．

〈リスト4-11〉TRANS.EXEのソース・リスト[TRANS.C]

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

main(){
        int d;

        printf("\n\t########## [transfer.hex] --> ROM Emulator ##########\n");


        system_init();                  /* システムの初期設定 */

        if(hex_file_send()==1){         /* ターゲット・ファイルをロード */
             printf("\n\n\t !!! Target File Missing !!!\n");
             exit(0);
        }

        system_start();                 /* システムをスタート */

        printf("\n\n\t########## Let's Start ([r]eset/[s]end) ##########\n");

        while(1){                       /* 無限ループ */
             if(kbhit()!=0){
                        d=getch();                      /* キー入力をチェック */
                        if(d==0x1b) break;
                        else if(d=='r'){                /* [r]ならリセット */
                                system_init();
                                system_reset();
                                printf("\n\n\t !!! Reset !!!\n");
                        }
                        else if(d=='s'){                /* [s]なら再ロード */
                                system_init();
                                hex_file_send();
                                system_start();
                                printf("\n\n\t !!! Re-Send !!!\n");
                        }
             }
        }
}

system_init(){
        set_4bit(6,0x02);               /* 010 */       /* Bus Disable */
}

system_start(){
        int i,j=0;
        set_4bit(6,0x06);               /* 110 */       /* Bus Enable */
        for(i=0;i<10000;i++) j++;
        set_4bit(6,0x07);               /* 111 */       /* Go ! */
}

system_reset(){
        int i,j=0;
        set_4bit(6,0x06);               /* 110 */       /* Reset ! */
        for(i=0;i<30000;i++) j++;
        set_4bit(6,0x07);               /* 111 */       /* Go ! */
}

setting_1byte(unsigned int address, int data, int head){
        set_address(address);
        set_data(data);
        set_4bit(6,0x00);               /* 000 */       /* Write Pulse */
        set_4bit(6,0x02);               /* 010 */       /* Bus Disable */
        if(head!=0){
             printf("\n\t\tAddress = %04X , Data = %02X ",address,data);
             }
             else if(address%16==0) putchar('.');
             return(0);
}

set_address(unsigned int address){
        unsigned int add;
        add=address;
        set_4bit(2,add%16);             /* アドレスA3-A0のセット */
        add=add/16;
        set_4bit(3,add%16);             /*       A7-A4       */
        add=add/16;
        set_4bit(4,add%16);             /*       A11-A8      */
        add=add/16;
        set_4bit(5,add%16);             /*       A15-A12     */
}

set_data(int data){
        set_4bit(0,data%16);            /* データD0-D3のセット */
        set_4bit(1,data/16);            /*       D4-D7       */
}

set_4bit(int add, int data){
        outport(0x0040,16*add+data);    /* 7ビットのデータをセットして */
        outport(0x0046,0x0e);           /* ラッチ・パルスを下げて */
        outport(0x0046,0x0f);           /* 上げる。このエッジで書き込み */
}

conv(int a){
        if(a<'9'+1) return(a-'0');
        else return(a-'A'+10);
}

hex_file_send(){            /* インテルHEXファイルを読み込んでロードする */
        FILE *fds;
        unsigned int address,head=0;
        int data,d,a,dd[10],i,count;
        if((fds=fopen("transfer.hex","rb"))==NULL) return(1);
        while(1){
                d=fgetc(fds);
                if(d!=':') break;
                dd[0]=fgetc(fds); dd[1]=fgetc(fds);
                dd[2]=fgetc(fds); dd[3]=fgetc(fds);
                dd[4]=fgetc(fds); dd[5]=fgetc(fds);
                fgetc(fds); d=fgetc(fds); if(d=='1') break;
                a=dd[0]; count=16*conv(a);
                a=dd[1]; count=count+conv(a);
                a=dd[2]; address=4096*conv(a);
                a=dd[3]; address=address+256*conv(a);
                a=dd[4]; address=address+16*conv(a);
                a=dd[5]; address=address+conv(a);
                head=1;
                for(i=0;i<count;i++){
                    dd[0]=fgetc(fds); dd[1]=fgetc(fds);
                    a=dd[0]; data=16*conv(a);
                    a=dd[1]; data=data+conv(a);
                    head=setting_1byte(address,data,head);
                    address++;
                }
                fgetc(fds); fgetc(fds); fgetc(fds); fgetc(fds);
        }
        fclose(fds);
        return(0);
}
```

インテルHEXファイルからのダウンロード・ソフト

プリンタ・ポートに接続されるROMエミュレータのハードウェア(**図4-6**)が完成し，上記のようにソフトウェアから叩くための場所が解析できれば，あとはこの情報をもとに「ROMエミュレータにXA80で作成したインテルHEXファイルをダウンロードし，ターゲット・システムをリセットする」という制御プログラムを作成すればよいことになります．ここには大きく二つの重要なステップがあります．その第一は，製作したハードの確認を兼ねて，実際にソフトからROMエミュレータを制御する部分の製作，そして第二には，インテルHEX形式ファイルを読み込んで，規約にしたがって正しくRAMに書き込む部分です．

リスト4-11は，このような目的のためにパソコン上のCコンパイラで開発したプログラム[TRANS.EXE]のソース・リストです．今回はTurbo-Cで作りましたが，MS-CでもLSI-C[TRANS.C]でもほとんどそのままで使えます．

基本的に本書の読者は簡単なプログラムをCで書いたことはあるという前提で，ざっと解説しておきましょう．リスト内の順序とはやや前後しますが，トップダウンとボトムアップの両方の視点で，わかりやすいところから説明することにします．

まずmain()ですが，非常に単純な構造となっています．最初にsystem_init()という関数を呼んでシステム全体を初期設定し，次にターゲット・ファイル(ここでは[transfer.hex]という名前で予約)を探しに行って，ついでに読み込んでROMエミュレータにダウンロードしてしまうのがhex_file_send()という関数です．この名前のファイルが存在しなければ，これで終了です．そしてsystem_start()という関数で，ターゲット・システムのリセット信号を“L”にしてふたたび“H”にします．これでエミュレーションは開始されて，あとはほとんど意味のない無限ループを回っているだけです．

無限ループでは，再度のダウンロード，再度のターゲット・リセット，そしてプログラム終了のいずれかをキー入力で選択できるだけです．

ここでちょっと飛びますが，まずset_4bit(int add, int data)という関数を見ておきましょう．これは，ROMエミュレータ内の7個のラッチ74HC574に

逆アセンブル　機械語プログラムを解析して，元のアセンブラ・プログラム・リストを再構成すること．完全に実現することは不可能である．

対して，dataという4ビット・データを，addで指定されるラッチに書き込むという関数です．トップダウン式にmain()を書くと，筆者はこのように今度はボトムアップ的に「部品」となる関数をよく書きます．

実際にプリンタ・ケーブルで供給されるデータは，下位4ビットがここでのdataで，addはその上位に来ますから，パソコンから見たデータは「16*add+data」となります．そして，これを解析結果の情報から，「40h」のポートに書き込み，あとはポート「46h」のLSBを下げて上げてラッチするというものです．いちいちこのように書き込むのは手間のようですが，どうせパソコンは十分に速いので問題ありません．

このset_4bit()という関数を使って，システム制御のために，system_init()，system_start()，system_reset()という3種類の関数を作っています．これも中身は単純で，制御信号を設定する7番目のラッチの3ビット・データを作っているだけです．この部分は，中身としては単純なのですが，ビットの位置が異なったり，“L”と“H”の状態が反転していたりすると，システムがまったく動かないことになる重要なところです．開発のステップとしては，回路図と突き合わせてよく検討する必要のある，もっともクリチカルな部分でしょう．

一方，RAMにデータを書き込むためには，データの8ビットと，アドレスの15ビットを設定しなければなりません．そこでまず，set_4bit()という関数を使って，set_data(int data)というデータを2個のラッチにセットする関数，set_address(unsigned int address)というアドレスを2個のラッチにセットする関数を用意します．すると，この「このアドレスにこのデータ」という書き込むべき内容を添えて呼び出すsetting_1byte()という関数は簡単に記述できます．

一般にプログラムというのは連続したアドレスに入っていますから，アドレスの部分は初期値からインクリメントすることが多いのですが，この例では，スピードよりも単純さを重視して，いちいちアドレスを与えて設定するようにしています．スピード重視の製品版ROMライタなどでは失格の方法ですが，筆者の経験からすると，「単純なものほど確実に動く」というのも大切なことなのです．

さて，以上で「ROMエミュレータのRAM内の任意のアドレスに任意のデータをロードする」という部分まで完成しました．あとは，インテルHEXファイルの内容を正しく解釈して，連続してダウンロードすればよいことになります．

リスト4-11では，予備的な関数としてconv(int a)というものを置いています．これは，バイナリ・データを16進テキストとして表現したインテルHEXファイルを処理するための「16進文字から対応した10進数に変換する関数」です．

このような処理は，C言語ではこの例のほかに，少なくとも3，4種類は瞬時に考えつくようでないと困りますから，Cの経験者はぜひ考えてみてください．

さて，いよいよメインのhex_file_send()にやってきました．といっても，ここまでの準備によって，中身はとてもシンプルになっています．

まず最初にファイルの存在をチェックしてオープンし，あとは「データ終了」で抜け出すまでの無限ループとなっています．そして，今回のAKI-80システム開発では，アセンブラにXA80というソフトを使っていて，基本的に「アセンブラの出力したインテルHEXファイルには，形式違反などのバグはない」という前提でいるために，ソフトはとても単純になりました．もしアセンブラまで自作している場合には，インテルHEXファイル・フォーマットに対する形式チェックをかなり厳重にしなければなりません．ちなみに，Javaではセキュリティ(悪意に満ちた改造を紛れ込ませる不正の除去)のために，作成されたJavaプログラムを実行させる前に厳重な形式チェックを行っています．「規約通りであることのチェック」なんてつまらないという感覚でいた人は，インターネットの時代になって発想の転換が必要のようです．

リスト3-6(p.20)のルールにしたがって，各行のデータ数と先頭アドレスを読み込むと，あとは「データ数だけループ」というところで2文字ずつのデータを数値に変換しては，setting_1byte(address,data,head)を呼び出してRAMに転送します．最後の**チェック・サム**による検証も省略して，最終データまで回せば終了です．この関数が終了したところでバスを切り替えてターゲット・システムをリセットすれば，もうそれでエミュレーションの開始となるわけです．

アセンブラ環境でのAKI-80システム開発

このようなROMエミュレータとダウンローダによって，実際にAKI-80システムを開発する手法については，すでに**リスト3-1**(p.16)で簡単に紹介しました．ここで再度，確認しておきましょう．**リスト3-1**のバッチ・プログラム[ASM.BAT]によって，AKI-80システムのためのプログラム[TEST.ASM]を開発し，アセンブラXA80によってアセンブルしてインテルHEXファイル[TEST.HEX]を作り，これをツールの予約である[TRANSFER.HEX]にコピーして，最後にダウンローダの[TRANS.EXE]を起動してダウンロードします．

このループはきわめて快調にスイスイと回りますから，ちょっとしたバグがあってもほとんど気にしませ

チェック・サム　データ化けなどのチェックのために，一定の範囲のデータを加算して桁溢れを切り捨てたデータを添えること．送り側と受け側とで調べることで「正しく届いたか」をチェックできる．

ん．これがROMライタでいちいちプログラムするとなると，締め切りの決まっている仕事の場合には，「失敗するとまた時間がかかる」というプレッシャーで硬くなるのですが，ROMエミュレータの場合には，「ミスったらやり直せばいい」という軽いノリが命なのです．

筆者もサラリーマン時代のシステム開発では，さんざんICEによるデバッグのお世話になったのですが，ここ数年は，数十種類のマイコン・システムを開発しているにもかかわらず，ICEやデバッガを使ったことはまったくありません．システム開発とデバッグは，すべてこのROMエミュレータのみでOKなのです．ROMエミュレータの最大の欠点は，**ブレーク・ポイント設定**によるデバッグやトレースができない，などと言われます．これは確かに事実なのですが，ある程度慣れたエンジニアにとって，「デバッガなどなくてもデバッグできる」という，より上級のテクニックを獲得するチャンスでもあるのです．

ROMエミュレータによる開発とは，「たぶんこれで動くのではないか」と思われるプログラムをエミュレーションRAM(ターゲット・システムにとってのROM)に転送し，あとは運を天に任せてリセット一発，というかなり乱暴なものです．巨大なCPUプログラムがこれで最初から動くわけもなく，また動かない原因の追求もほぼ不可能です．そこで，開発にあたっては「ボトムアップ手法」「トップダウン手法」「ソフトウェアの部品化」「デバッグ・ルーチン」などのテクニックを駆使することになります．

「**ボトムアップ手法**」とは，動作の確実な，単純な小さなルーチンを確認しながら積み上げて，全体として大きな処理に構築していく手法です．上記の例でいえば，「ある1バイトをセットする」という基本的な関数をまずボトムとして作り，さらにこれを用いて「アドレスをセット」「データをセット」という関数を作り，さらにこれらを組み合わせた「RAMのあるアドレスにデータをセットする」という関数を作り上げたような手法です．

最初からこのような美しい関数を作ろうと色気を見せると，絶対に失敗することを筆者は断言できます．

「**トップダウン手法**」とは，これとは逆に，まず全体を構成するメイン・ルーチンや基幹サブルーチンを記述するというものです．

たとえばAKI-80システムであれば，リセット後にスタートするゼロ・アドレスからの処理は，

```
リセット
↓
割り込み禁止
↓
初期化ルーチンをコール
↓
メイン・ルーチン先頭
↓
基幹サブルーチン(1)をコール
↓
基幹サブルーチン(2)をコール
↓
～
↓
メイン・ルーチンに戻る(無限ループ)
```

と決まっています．このような構造については，筆者は最初からテンプレートのように用意しておいて，毎回新しく打ち込む(ミスタイプによるバグのもと)ことはしません．基幹サブルーチンについては，名前だけ定義しておいて，最初のうちは中身を「リターンだけ」としておきます．こうすることで，全体について見通しよく開発することができます．

「**ソフトウェアの部品化**」とは，ボトムアップによって開発した基本処理については，なるべくターゲット・システムに特化しないものを心がけて，また新しいシステム開発のときには，なるべく「以前に稼動実績のあるルーチンをそのままコピーしてくる」ように心がけることです．同じ処理をするルーチンを新しく書き下ろすと，たいていは前回よりもスマートで速いものが作れるのですが，バグの可能性も宿すことになります．よほど性能向上を狙わないのであれば，基本的には「この処理は，以前にどこかで作っていないかな」と探すようなスタイルが望ましいのです．

「**デバッグ・ルーチン**」とは，文字通りデバッグをサポートするためのサービス・ルーチンです．もちろん，デバッグを支援するためのハードと組み合わせることも少なくありません．

たとえば，AKI-80に空いている出力ポートを1ビットでも確保して，ここにLEDをつけておけば，デバッグにおいてはかなり強力なツールとして使えます．

特定のポートの信号をメイン・ルーチン中に上げ下げさせることで，オシロスコープで計測してメイン・ルーチンの処理時間を調べるのも定石のテクニックです．また，動作状況を特定の変数に確保しておいて表示したり通信出力するようなデバッグ・ルーチンも重宝します．このような視点は，ビギナーにはなかなか理解してもらえないのですが，システムを開発したことのある人であれば，納得してもらえるでしょう．

ブレーク・ポイント設定　プログラム中に特定の場所まで来たら「ソフトウェア割り込み」などを発生させるようにしておいて，その場所に到達したことを検出できるようにすること．

Appendix

Windowsパソコンでの開発環境

WIndows95とMS-DOS

本書の読者の中には，最初からWindowsマシンを持っていて，今さらテキスト・ベースの中古98なんて使いたくないという人もいると思います．そこでここでは，本書で行っているAKI-80の開発環境をWindowsマシンで実現する手法についてまとめて紹介します．

実は筆者は，本書の原稿執筆が半分以上も進んだところで，ようやくWindowsマシンを購入したところです．これが現役として12台目というところまで自宅にWindowsマシンを導入しなかったのは，もちろんMacとUnixが好きだからなのですが，仕事の関係でどうしても必要になったために，例によって最新でないDynabookのサブノートを仕入れました．そして初めてWindows95にマトモに触れて4日後には，以下に紹介するようなAKI-80開発環境を半日がかりで実際に製作・構築しました．ここで紹介する環境は仮想的なものでなく，実際に稼動している現役のものです．

筆者はテキスト・エディタとしては定番の「秀丸エディタ」(シェアウェア)を使用しています．つまり，ターゲットのソース・ファイルについては，エコロジー代わりのエクスプローラからアイコンをダブルクリックして秀丸エディタを起動して編集することにしました．

またバッチについては，これもシェアウェアのWinbatchというツールを入手して登録を済ませました．これは，Windows環境から一連のバッチ処理を逐次実行して結果を表示してくれるもので，スムーズにDOS窓とWindowsを行き来できます．

結果として，デスクトップには常にエクスプローラを開いておいて(DOSでは常にエコロジー環境から仕事をしているのと似た感覚です)，ソースはアイコンを叩いて秀丸エディタでエディットし，コンパイルやエミュレーションはWinbatchのアイコンとして登録したバッチ処理を，これもアイコンを叩くことで自動起動してサクサクと進むということとなりました．もちろん，Winbatchから呼び出すバッチの最初に「秀丸エディタでソースを編集」という処理まで含めてしまって，毎回スタートアップ・メニューに登録したショートカットを呼ぶという方法でもかまいません．

98とDOS/Vマシン

アセンブラやCコンパイラ，そしてエディタとバッチ環境が揃ってくると，あと必要となるのは自作したROMエミュレータとのインターフェース部分です．ここには，

・ROMエミュレータとの物理的な接続
・DOS/Vマシンでのプリンタ・ポート処理
・ファイル転送プログラムの開発

という具体的な課題があることになります．このそれぞれで，98とDOS/Vマシン(DynaBookは完全なPC互換機)との違いを吸収していかなければなりません．

まずは，プリンタ・ポートとの接続です．筆者の場合，ROMエミュレータは98Noteの特別な20ピン・ミニ・コネクタの専用ケーブルで製作していますが，98環境でも開発したいので，このハードにはいっさいの改造を行いたくありません．いっぽう，DOS/Vマシンのプリンタ・ポートは「パラレル・ポート」と言い，必要に応じて8ビット・パラレルで出力だけでなく入力まで行えるスグレモノで，コネクタは25ピンのDSUBです．

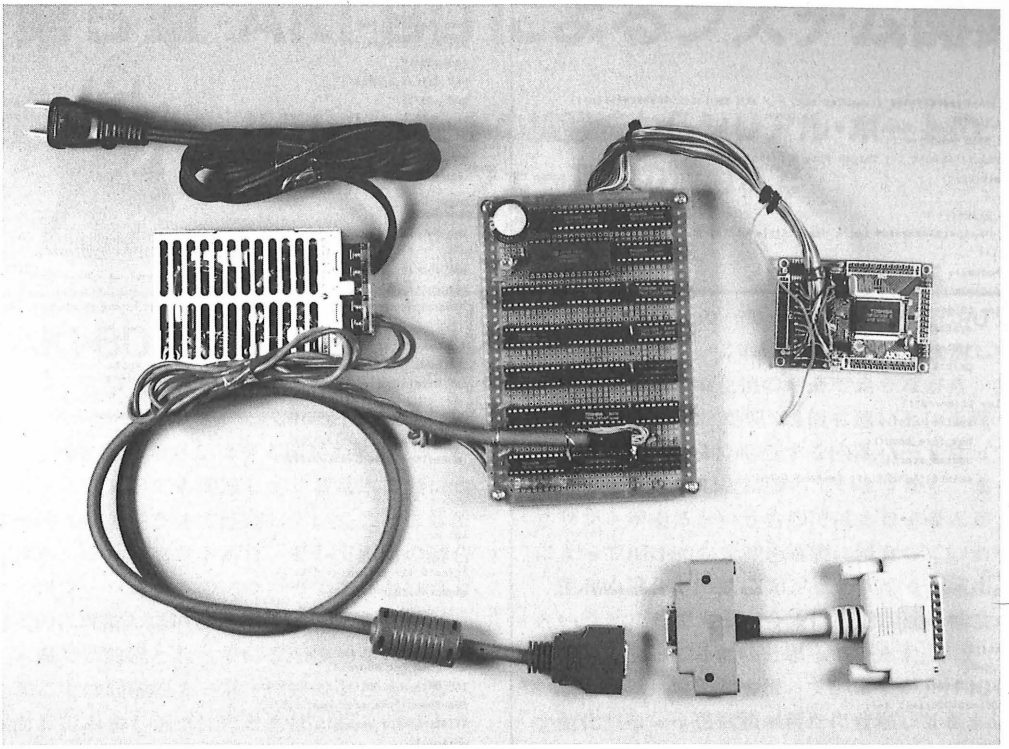
そこで筆者はパソコン・ショップで，「20ピン・ミニ・コネクタのメス側」のついたケーブルを探した(反対側を切って25ピンDSUBを付ければよい)のですが，ありませんでした．そしてようやく発見した，「20ピン・ミニのメスと36ピン・ミニのオス」とを変換するアダプタを分解して，この36ピン側を取り去り，ここに25ピンDSUBケーブルを接続して，ROMエミュレータとDOS/Vマシンを接続する変換ケーブルを製作(**写真4-1**)しました．

テスタで慎重に対応するピンを調べてはんだ付けする必要がありますが，8本の信号と1本のGNDの，基本的には単純な作業です．

そして，「DOS/VマシンのパラレルポートのI/Oアドレス」が378hであるという情報だけを頼りに，以下のCプログラムでこのポートを直接叩いてみると，なんとWindowsマシンとはいえ，ちゃんとポートの信号が上下してくれることをオシロスコープで確認できました．これで，あとは転送プログラムで開発環境が整備されるというわけです．

最後に残ったのは，インテルHEXファイルを自作のROMエミュレータに転送するためのプログラムの移

〈写真4-1〉DOS/Vマシンと接続するROMエミュレータと接続ケーブル



植です．ところが，筆者が持っていたCコンパイラ(MS-C，Turbo-C，LSI-C)は98用だったために，ここでハタと困ってしまいました．そして倉庫の奥から発掘してきたのが，ほとんど知られていない「DeSmet-C」というCコンパイラです．これはかなり大昔のものらしく，MS-DOS2.11でも動くというもちろんDOS汎用のコンパイラです．筆者は以前にも，オムロンMassifという電子手帳をハックしてパソコン化してしまったのですが，この時にもこのDeSmet-Cが活躍したのです．

筆者はWindowsでC言語によるソフト開発をするつもりはありません(Unixで開発するのが中心)ので，新しくCコンパイラを導入することなく，ここでもDeSmet-Cを利用してみました．

リスト4-12が，本文中の**リスト4-11**(p.37)の転送プログラムに相当する，DynaBookのためのプログラム[TRANS2.EXE]のソース[TRANS2.C]です．基本的にはまったく同じことをしていますが，いくつかの関数の名前が違っています．なお，この改造にあたって，筆者は98ではプリンタ・ポートのストローブ信号を利用していたMSBに，パラレル・ポートのMSBをそのまま使用するように変更しています．つまり，下位7ビット・データを添えてMSBを立てたデータをまず出力し，次いでMSBを“L”にした同じ7ビット・データを出力し，ふたたびMSBを立てて出力するというようになっています．

「ハードの変更はすべてソフトで吸収する」というシステム開発の典型的な実例というわけです．

以上で環境は完備しました．筆者は自宅や研究所では98Noteによって開発していますが，たとえば長野高専の電子情報工学科に出かけての集中講義や豊橋市での技術セミナーなどに持参する場合には，Windowsサブノートを持参しています．このメリットとしては，パソコンの画面をプロジェクタで表示して一緒にソースを見る時に，自由にフォントを拡大してよく見えるようにできることで，この点ではさすがにWindowsマシンは便利というわけです(ただし，実践的なシステムを開発しようとすると，やはりエコロジーとMIFESのほうが，はるかに開発効率は高いと実感する毎日です)．

〈リスト4-12〉 TRANS2.EXEのソース・リスト[TRANS2.C]

```
#include <stdio.h>

main(){
    int d;
    printf("\n\t########## [transfer.hex] --> ROM Emulator ##########\n");
    system_init();
    if(hex_file_send()==1){
        printf("\n\n\t !!! Target File Missing !!!\n");
        exit(0);
    }
    system_start();
    printf("\n\n\t########## Let's Start ([r]eset/[s]end) ##########\n");
    while(1){
        d=csts();
        if(d==0x1b) break;
        else if(d=='r'){
            system_init();
            system_reset();
            printf("\n\n\t !!! Reset !!!\n");
        }
        else if(d=='s'){
            system_init();
            hex_file_send();
            system_start();
            printf("\n\n\t !!! Re-Send !!!\n");
        }
    }
}

system_init(){
    set_4bit(6,0x02);                   /* 010 */     /* Bus Disable */
}

system_start(){
    long i,j=0;
    set_4bit(6,0x06);                   /* 110 */     /* Bus Enable */
    for(i=0;i<100000;i++) j++;
    set_4bit(6,0x07);                   /* 111 */     /* Go ! */
}

system_reset(){
    long i,j=0;
    set_4bit(6,0x06);                   /* 110 */     /* Reset ! */
    for(i=0;i<300000;i++) j++;
    set_4bit(6,0x07);                   /* 111 */     /* Go ! */
}

setting_1byte(address,data)
    unsigned int address;
    int data;
{
    set_address(address);
    set_data(data);
    set_4bit(6,0x00);                   /* 000 */     /* Write Pulse */
    set_4bit(6,0x02);                   /* 010 */     /* Bus Disable */
    putchar('.');
    return(0);
}

set_address(address)
    unsigned int address;
{
    unsigned int add;
    add=address;
    set_4bit(2,add%16);
    add=add/16;
    set_4bit(3,add%16);
    add=add/16;
    set_4bit(4,add%16);
    add=add/16;
    set_4bit(5,add%16);
}

set_data(data)
    int data;
{
    set_4bit(0,data%16);
    set_4bit(1,data/16);
}

set_4bit(add,data)
    int add;
    int data;
{
    _outb(16*add+data+128,0x378);
    _outb(16*add+data+0,0x378);
    _outb(16*add+data+128,0x378);
}

conv(a)
    int a;
{
    if(a<'9'+1) return(a-'0');
    else return(a-'A'+10);
}

hex_file_send(){
    unsigned int address;
    int data,d,a,dd[10],i,count,fds;
    fds=fopen("transfer.hex","rb");
    if(fds==NULL) return(1);
    while(1){
        d=fgetc(fds);
        if(d!=':') break;
        dd[0]=fgetc(fds); dd[1]=fgetc(fds);
        dd[2]=fgetc(fds); dd[3]=fgetc(fds);
        dd[4]=fgetc(fds); dd[5]=fgetc(fds);
        fgetc(fds); d=fgetc(fds); if(d=='1') break;
        a=dd[0]; count=16*conv(a);
        a=dd[1]; count=count+conv(a);
        a=dd[2]; address=4096*conv(a);
        a=dd[3]; address=address+256*conv(a);
        a=dd[4]; address=address+16*conv(a);
        a=dd[5]; address=address+conv(a);
        for(i=0;i<count;i++){
            dd[0]=fgetc(fds); dd[1]=fgetc(fds);
            a=dd[0]; data=16*conv(a);
            a=dd[1]; data=data+conv(a);
            setting_1byte(address,data);
            address++;
        }
        fgetc(fds); fgetc(fds); fgetc(fds); fgetc(fds);
    }
    fclose(fds);
    return(0);
}
```

第5章 AKI-80によるシステム開発

パラレル・ポートの拡張法からシリアル・ポートのMIDIインターフェースへの活用まで

AKI-80のメリットとデメリット

本章では，AKI-80によるシステム開発の概論として，どんなシステムにおいても関連してくる基本事項について，技術的な検討を交えて確認していくことにします．話題はハードありソフトあり，そして両者の融合こそがマイコン・システムのノウハウであるというところを感覚的に理解していただければ十分です．

まず，本書では前提となっているAKI-80そのものについて，ここでは同種のカード・マイコンともくらべながら検討してみましょう．たとえば，試みに「トランジスタ技術」誌の広告をざっと眺めてみると，同じ東芝のCPUを使用したボード・マイコンやカード・マイコンはAKI-80だけでなく，全部で数社，種類にして20機種程度の異なった製品として発表されていることがわかります．サイズも価格もオンボードの周辺機能もいろいろで，初めての人にとっては困惑してしまうかもしれません．

これらすべてを取り上げて詳細に比較するのは本章の目的ではありませんから，ここでは1社，梅澤無線のカード・マイコン(**図5-1**)についてコメントしておきましょう．これはAKI-80とほぼ同等，あるいは一回り強化された機能を搭載しているもので，筆者も何度となく試作実験に使用してきました．

AKI-80と違って，オリジナルのゲートアレイが搭載され，そして相当に価格が高いのです．しかし，これはアマチュア向けの製品ではありません．全数の出荷検査を行い，完成品として品質保証された「プロのための部品」なのです．つまり，プロは梅澤ボードを組み込んだ自動工作機械などの本格的な製品を製造できる，そのような機能と品質を支える価格という意味では，業界から見るとこれまた驚くほど低価格な製品なのです．アマチュアのキットであるAKI-80とは，本質的に比較できないというところにピンとくれば，見方として本物です．

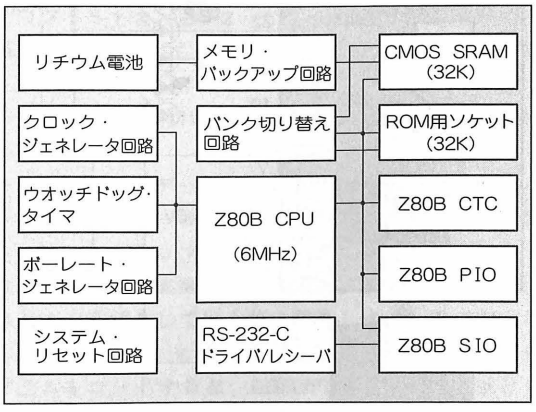
これまでカード・マイコンの世界で独走していたAKI-80ですが，最近になって大きなライバルが登場してきました．その一つが，「**川鉄4倍速CPU**カード・マイコン」です．これは，東芝CPUと決別して，CPUのコア部分はZ80互換の命令で走りながら，実際には内部パイプライン処理で4倍も速いという驚くべきものです．周辺LSIに相当する内蔵ハードウェアのポート構成も東芝CPUとは違っていますから，まったく同じソフトが走るというものではありませんが，低価格によってAKI-80の牙城を確実に浸食しています．

筆者の場合，川鉄CPUはまだ実験を行ってみただけというところですが，今後は高速用途の機会があれば本格的に使ってみたいと思っています．

また，秋月電子でも扱っていますが，「**PICによる超小型化**」という選択肢が新たに登場してきました．

これは8ビットRISCタイプの新しいCPUで，チップ内のEPROMにプログラムを書き込んでしまえるので，AKI-80どころか，それこそAKI-80の4分の1程度のサイズにまでシステムを小型化できるという画期的なシステムです．まだ現在のところでは「Z80システムを完全に置き換える」というほどの機能的，価格的なメリットが出てきていませんが，急速に機能拡張と多品種戦略を進めていますから，AKI-80にとっては最大のライバルとなっていると思います．筆者も手元にPIC開発ツールを取り寄せて実験してみているところですが，本書では残念ながら，これ以上この話題に触

〈図5-1〉梅澤無線のカード・マイコン



川鉄4倍速CPU　川崎製鉄オリジナルのZ80互換CPUを内蔵してクロックが4倍の高速になったCPU．東芝のCPUとは直接の互換性はない．

れることはありません．

これらの新しいライバルを含めて考えると，「AKI-80のメリット」と「AKI-80のデメリット」がはっきりしてきます．メリットは従来からいわれているように，

- Z80のソフトウェア資産
- 開発環境が普及している
- 超小型
- シンプルなシステム構成
- 低価格

などというところでしょう．

筆者の印象では，川鉄の4倍速ボードを別にすれば，東芝CPUを使用したカード・マイコンとしては，AKI-80がいちばん便利で簡単という印象です．もっとも，キットを製作するのが苦手な人の場合には，他社から出ている完成品を活用するのもいいと思います．本書の例は，別にキットのAKI-80を使用しなくても，どこのメーカのボードであっても同じように活用できるメリットがあるのです．

そしてAKI-80のデメリットとして最大のものは「キットであることの品質的な不安」がいちばんでしょう．筆者の経験では，マニュアルにも書いてあるように，ROMソケットの下に置かれたRAMのミニ・フラット・パッケージ周辺のはんだ付け不良(どこか隣に付いてしまう)という危険性がもっとも高いと思います．ちなみに筆者がこれまでに数十枚のAKI-80を製作した中で，原因不明の動作不良で困ったというのはたった1枚でした．あとのすべてはノーチェックで完璧動作となっていました．この事実で安心するか心配になるかは，人によるでしょう．

あとは，デメリットとして「今さらZ80なんてダサい」「性能が低い」などという，Z80全体に対する批判があります．しかし，別にAKI-80を使っているときに「これはマイコンだ」なんて思わず，完全に「部品」感覚で使っている筆者にとって，まだまだAKI-80は現役として使えるモノだと思います．まだPICでAKI-80と張り合う性能を出すのは至難の業ですし，処理能力が不足していればAKI-80を複数使って分散処理をすればよいので，それほど性能不足で困ったこともありません．制御システムの大半は，人間の動作の遅さのためにCPU時間の90%ぐらい「足踏みして待っている」というのも，かなりの状況での実態なのです．

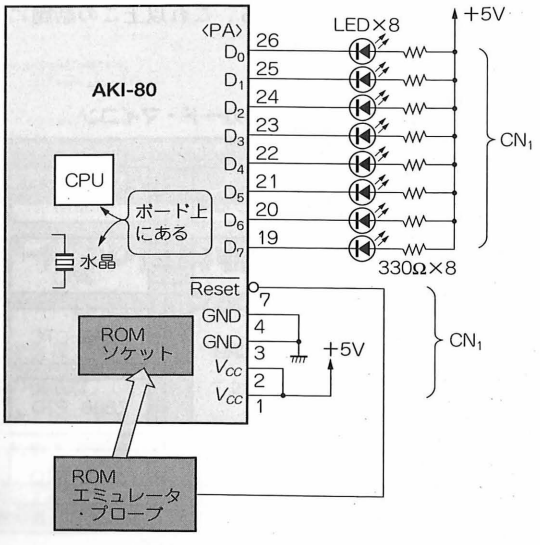
パラレル・ポート入出力とその拡張

さて，それでは具体的にAKI-80を利用する基本的な方法を，「最初に製作したAKI-80ボードの動作確認」を兼ねて進めていくことにします．

筆者ははんだ付けしたAKI-80の動作確認のために，まずパラレル・ポートを定石的に利用しています．シリアルというのは時間的に変化していて「見えにくい」のですが，パラレル・ポートの「定常的な状態」というのは人間の時間感覚まで待っていてくれますから，これはもっとも基本といえるものです．

図5-2は，とりあえずはんだ付けしたAKI-80が「正しく製作された」ことを確認するための最低限の回路図です．普通のマイコン回路をディスクリートで製作した場合には，どこか1箇所だけ配線やはんだ付けのミスがあっても動きませんから，経験的には「とりあ

〈図5-2〉AKI-80チェック回路

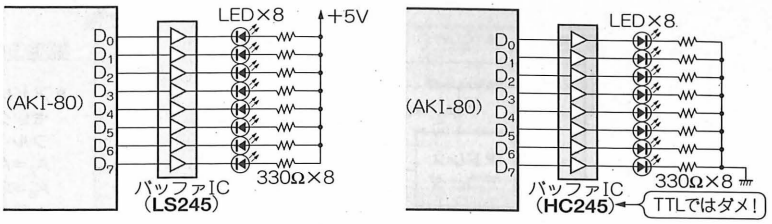


〈リスト5-1〉LED点滅のプログラム[LED.ASM]

```
;##### RAM Map #####
        org     8000h           ;← RAM領域に変数を定義
timer   ds      2

;##### I/O Map #####
pio_a   equ     001ch           ;← 内蔵PIOのアドレス定義

;##### RESET #####
        org     0000h           ;← リセットするとここから始まる
        ld      sp,09fffh       ;← スタック・ポインタをセット
        di                      ;← 割り込みを禁止
        jp      main            ;← メインに飛ぶ

;##### NMI #####
        org     0066h           ;← (NMIは来ないが一応入れておく)
        retn

;##### Main #####
main:
        ld      a,0cfh          ;Mode 3
        out      (pio_a+1),a
        ld      a,00000000b     ;0:Out / 1:In
        out     (pio_a+1),a
loop:
        call    timer_check     ;← ソフトウェア・タイマ
        jr      loop

timer_check:
        ld      a,(timer+0)
        inc     a
        ld      (timer+0),a
        cp      0
        ret     nz              ;← 256回に1回だけ下に行く
        ld      a,(timer+1)
        inc     a
        ld      (timer+1),a
        cp      100
        ret     nz              ;← 100回に1回だけ下に行く
        xor     a
        ld      (timer+1),a
        ld      a,(timer+2)
        inc     a
        ld      (timer+2),a
        xor      11111111b      ;← 負論理なので全ビット反転する
        out      (pio_a),a      ;← PAポートから出力する
        ret

        end
```

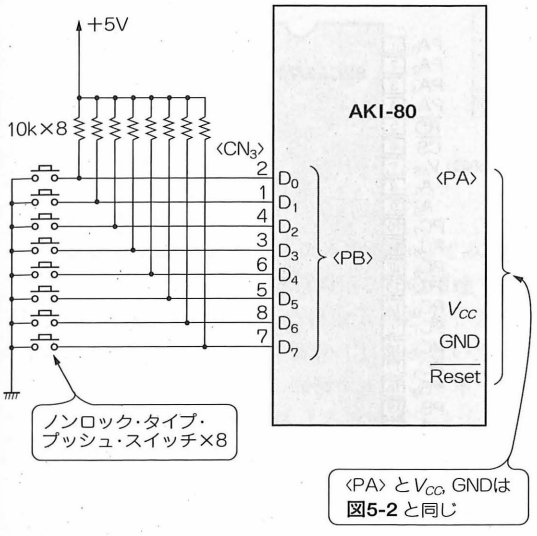

PICによる超小型化　マイクロチップ社の小型RISCタイプのCPU「PIC」を使うことで，さらにシステムは小型化できる可能性がある．

〈図5-3〉
LEDをスタティックに表示する回路



(a) 負論理ドライブ（"0"を書くとLEDが点灯）　(b) 正論理ドライブ（"1"を書くとLEDが点灯）

〈図5-4〉パラレル・ポートによるスイッチ入力



えず動く」という段階のところまで行くのがたいへんです．ところがAKI-80の場合にはキットなので，もっともシンプルなテスト用のソフトが走ってくれれば，その時点で，本来であればマイコン回路の製作で必要なチェック・ポイントである，

- バス周りの配線チェック
- アドレス周りの配線チェック
- デコーダ周辺とチップ・セレクト
- $\overline{RD}$，$\overline{WR}$，$\overline{IOREQ}$ など制御信号ライン
- 周辺LSIへの配線
- CPUのクロック周り
- リセット周り(AKI-80は専用リセットICを搭載)

などのテストは，すべて同時に済んでしまうことになります．このあたりは，従来型のマイコン・システムをいちいちすべて手配線した人でないと「ありがたみ」がピンときませんが，なんとも筆者には嬉しいことです．

図5-1 では，AKI-80の二つの8ビット・パラレル・ポート：PA，PBのうち，PA0～PA7にLEDを付けたところです．実際には，動作チェックのために1ポートを占有させるのはあまりにもったいない話です．しかし，パネルに並んだLEDをスタティックに表示させることが主目的(あとはシリアル通信のMIDI処理など)というような場合には，実際に開発するシステムがこのような回路になることもあります．

〈リスト5-2〉ポートPBのスイッチの状態をそのままポートPAから出力するプログラム[SW.ASM]

```
;##### I/O Map #####
pio_a       equ    001ch    ;← 内蔵PIOのアドレス定義
pio_b       equ    001eh

;##### RESET #####
        org    0000h         ;← リセットするとここから始まる
        ld     sp,09fffh     ;← スタックポインタをセット
        di                   ;← 割り込みを禁止
        jp     main          ;← メインに飛ぶ

;##### NMI #####
        org    0066h         ;← (NMIは来ないが一応入れておく)
        retn

;##### Main #####
main:
        ld     a,0cfh        ;Mode 3
        out    (pio_a+1),a
        ld     a,00000000b   ;0:Out / 1:In
        out    (pio_a+1),a
        ld     a,0cfh        ;Mode 3
        out    (pio_b+1),a
        ld     a,11111111b   ;0:Out / 1:In
        out    (pio_b+1),a
loop:
        call   sw_check      ;← スイッチ入力をLED表示
        jr     loop

sw_check:
        in     a,(pio_b)     ;← PBポートの状態を入力
        out    (pio_a),a     ;← PAポートから出力する
        ret                  ; (ともに負論理なので反転が不要)

        end
```

〈リスト5-3〉入力として使用しないビットはマスクする

```
main:
        ld     a,0cfh        ;Mode 3
        out    (pio_b+1),a
        ld     a,11110000b   ;0:Out / 1:In
        out    (pio_b+1),a   ; (ビット7-4が入力、3-0が出力)
loop:
        call   sw_check      ;← スイッチ入力をLED表示
        jr     loop

sw_check:
        in     a,(pio_b)     ;← PBポートの状態を入力
        and    11110000b     ;← 出力ビット情報はマスクする
        ld     (data),a      ; (入力データだけを使用する)
        .........
```

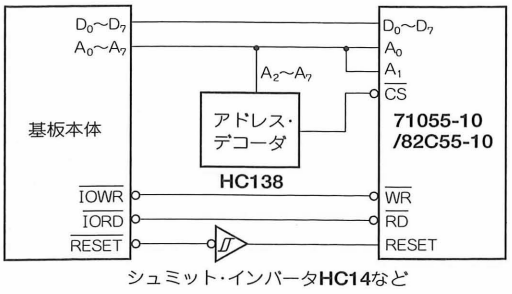
リスト5-1は，**図5-2**のAKI-80回路でLEDを順に点滅させるためのもっともシンプルなプログラム例(LED.ASM)です．ここでは，

- ポートPAの初期設定(8ビットとも出力ポートに指定)
- ポートPAにデータを出力
- メイン・ルーチン内で表示データを刻々と変化させる処理(ソフトウェア・タイマ)

などの基本的な実例が含まれていますから，AKI-80システムの確認としては十分に機能します．前章で製作したROMエミュレータによってアセンブル後のプログラムをロードすると，LEDが8個の2進数表示とし

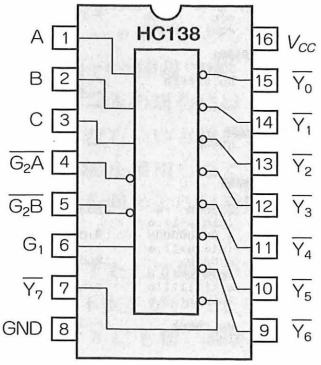
CMOSゲート　ディジタルICのうち，「TTLタイプ」と並んで多数のファミリを構成する「CMOSタイプ」のICのこと．出力電圧が+5VとGNDでフルスイングされる．

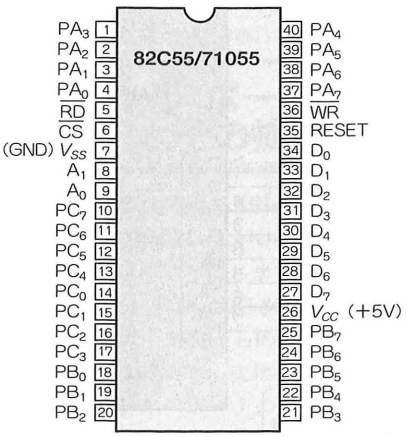
〈図5-5〉ポートの拡張



● アドレス・デコーダはHC138が便利.
セレクト，イネーブル端子すべて使えば
フル・デコードできる.
A_2=A, A_3=B, A_4=C, A_5=G_1,
A_6=G_2A, A_7=G_2B
上記のような接続で，20H番地より
4番地おきにデコードする.

ピン配置





て次々に点滅していきます.

なお，LEDをスタティックに点灯させる回路というのは，従来のTTLの時代には**図5-3(a)**のように，基本的に「負論理」でドライブするというのが常識でした. ICのゲートというのは電池ではないのですから，外部から供給される電源ラインからLEDを経由して「電流を引き込む」ことで点灯させたわけです.

ところが最近では，**図5-3(b)**のように，直接ゲートからLEDの電源を供給するような使い方ができるようになりました. これは“H”レベルをほぼ電源電圧までフルスイングして，さらに電流を10-20 mA も流せるという「CMOSゲート」の最大の特徴を活用したものです. 筆者の経験では，カタログ・データとしてはやや無理をしているような気もするのですが，AKI-80のポートから220 Ω程度の抵抗で全ビットのLEDをドライブするのは可能です.

図5-4はAKI-80のもう一つのパラレル・ポートであるPB0-PB7を用いて，8個のスイッチの状態を入力するための回路部分です. この部分に相当するのは，「ポートPBのスイッチの状態をそのままポートAから表示する」というサンプル・プログラム(SW.ASM，**リスト5-2**)の中でほんの2-3行だけであるのがわかります. なお，ここではスイッチ入力が**負論理**で，さらにLED出力も負論理のために，全ビットの反転が2回必要となり，結果として反転なしで入力をそのまま出力しています. もし，表示だけでなくスイッチ入力情報をほかのところで使う場合には，当然ビット反転が必要となることに注意しましょう. また実際の局面では，AKI-80の二つのパラレル・ポートは，それぞれビット単位で入力と出力を混在して設定できますから，かなり複雑な構成のシステムも実現できます.

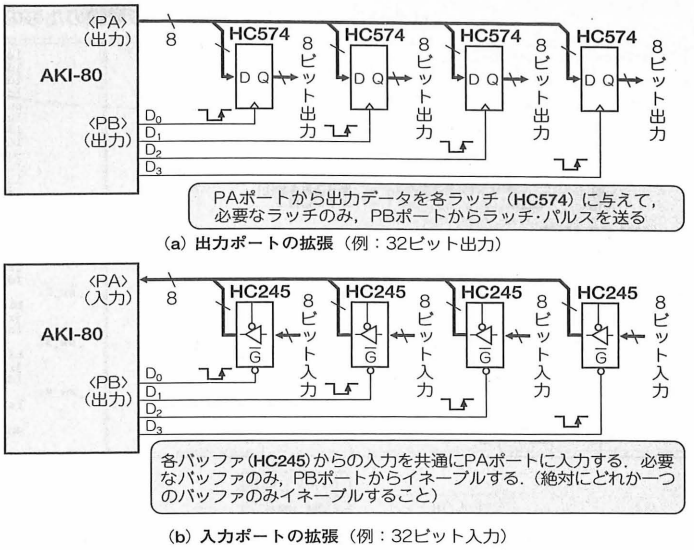
ここで，実際にAKI-80のパラレル・ポートを使う上での「**マニュアルから読みにいくノウハウ**」を二つほど紹介しておきましょう. まず，ポート入力は8ビット単位でまとめて行うので，その中に出力として指定されたビットがある場合，入力データの該当ビットは「不定」となります. そこで，入力として使用しないビットについては，**リスト5-3**のようにマスクするという習慣をつけておきましょう.

また，出力でなく入力として指定されたポートは，オシロスコープのプローブを当てるとオープン状態では「“H”レベル状態」のように見えますが，ここにLEDを接続しても点灯しません. 当然のことではあるのですが，筆者も点灯するものと勘違いしたことがありますので指摘しておきます.

さて，ここからはマニュアルに載っていない「ポート拡張」の話となります. 筆者の実感としては，AKI-80の最大の欠点は「パラレル・ポートの不足」にあり

負論理 信号の論理として，通常は「H」レベルで，必要な信号がアクティブな時だけ「L」レベルであるようにすること. 耐ノイズ特性が良好となる.

〈図5-6〉
パラレル・ポート入出力ビットの拡張



ます．ちょっとした制御システムを製作してみると，スイッチやセンサの入力，LED表示などの用途で，ポートがあと8ビット1ポートだけ欲しいとか，さらには「あと4ビットだけ余分にあれば」などということが少なくありません．他社の同種マイコン・ボードのセールス・ポイントの多くは「多数のポートを提供」ということですから，一般的な要請もこのあたりにあるようです．

ここで教科書的にポートを拡張する場合，**図5-5**のようにPIOの8255(CMOSでいえばNECのμ**PD71055**など)を増設することになります．この方法は秋月電子のマニュアルにも記載されています．しかし，筆者はこの手法はめったに採用しません．これは，「アドレス・デコーダのICが増えて美しくない」という理由もありますが，「CPUバスに直結した拡張だから」というのが最大の理由です．せっかくブラック・ボックスとして完結したAKI-80システムなのに，内部CPUのデータ・バスやアドレス・バス，そして制御信号やリセット・ラインまで外に出して増設の配線を行うといったことは，「なかなか動かない」「ノイズで誤動作」などのマイナス要因を増大させてしまうのです．

もちろん，8ビット3ポートの8255を10個使って「240ビットのポート増設」とか，バス対応の高速な処理を必要としている場合には迷わず採用しますが，日常的な制御システムでは，このような要請はそれほど多くありません．

筆者の製作するシステムは「音楽」関係が多く，MIDI処理という高速シリアル通信は非常にシビアな動作が要請されるのですが，人間との接点である「人間からの働きかけを取り込む」「人間に状態を表示して知らせる」という部分については，かなり低速でも十分に間に合うことがほとんどです．そこで，パラレル・ポートを増設する場合，筆者はまず，**図5-6**のようなアプローチを検討することにしています．これは原理的には，すでに製作例として紹介したROMエミュレータで活用しているテクニックと同等のものです．

つまり**図5-6(a)**の出力拡張の場合，AKI-80の出力ポートからいったん外部の複数のラッチに共通のデータを供給して，残りのビットでラッチごとの**ラッチ・パルス**(ストローブ)を「ソフト的に作ってやる」という方法です．

これは当然ながら，バスに8255などを拡張した場合にくらべて，いちいちソフトでポートの信号を上げ下げするために，処理の時間は10倍程度は遅くなります．しかし，AKI-80はたいていの場合，これでも十分に使えるほどのスピードで(人間に対して)動作してくれているのです．

図5-6(b)の入力拡張の場合にも，AKI-80の入力ポートに対して複数の**3ステート・ゲート**(245など)をいくつでも増設して，その中から排他的にイネーブルされるようなチップ・セレクト信号をソフト的に作ってやればよいことになります．実は，この例ではまだむだがあって，チップ・セレクトを与えるところにデコーダ(138や139)を使えば，さらにポートのビットが節約できるのですが，デメリットとしてICが増えるので，ケース・バイ・ケースで採用を検討します．なお，**図5-6(b)**の回路では，もし二つ以上の245がイネーブルされると，ポート入力の部分の信号ラインが「バス・ファイト」(衝突)状態となり，情報として不定となるだけでなく，ゲートを破壊することもありますから，ソフトウェアとして絶対にこのような状態にならないように留意します．

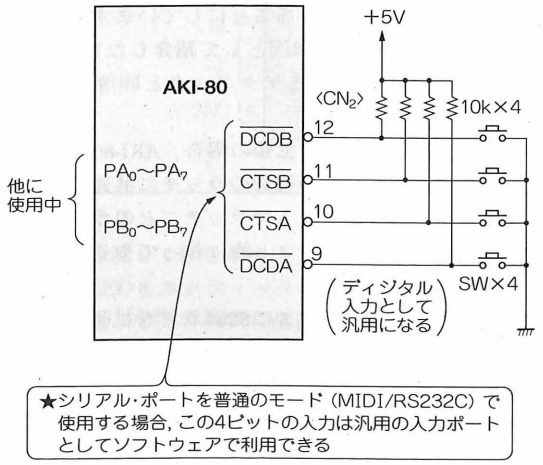
図5-6の拡張例をよく見てみると，これらを同時に

マニュアルから読みにくいノウハウ　LSIのメーカが一部ロットにあったバグを隠しているために，ソフト的に対応する「おまじない」などの，いろいろな表に出てこないノウハウ．

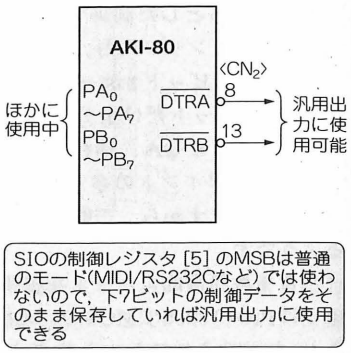
〈リスト5-4〉 汎用入力ポートの増設のためのプログラム[ADDIN.ASM]

```
;##### RAM Map #####
        org     8000h                   ; ← RAM領域に変数を定義
data    ds      4

;##### I/O Map #####
sio_a           equ             0018h   ; ← 内蔵SIOのアドレス定義
sio_b           equ             001ah

;##### RESET #####
        org     0000h                   ; ← リセットするとここから始まる
        ld      sp,09fffh               ; ← スタックポインタをセット
        di                              ; ← 割り込みを禁止
        jp      main                    ; ← メインに飛ぶ

;##### NMI #####
        org     0066h                   ; ← (NMIは来ないが一応入れておく)
        retn

;##### Main #####
main:
        ld      a,00011000b             ; ← SIO[A]チャンネルリセット
        out     (sio_a+1),a
        ld      a,00011000b             ; ← SIO[B]チャンネルリセット
        out     (sio_b+1),a
                                        ;   (本来はここで各種の初期化)
loop:
        call    sw_check                ; ← スイッチ入力を[data]に格納
        jr      loop

sw_check:
        xor     a                       ; ← データ領域をゼロクリア
        ld      (data+0),a
        ld      (data+1),a
        ld      (data+2),a
        ld      (data+3),a
        ld      a,00000000b             ; ← SIO[A]制御レジスタ[0]を選択
        out     (sio_a+1),a
        in      a,(sio_a+1)             ; ← 制御レジスタの状態を入力
        ld      b,a
        ld      a,00000000b             ; ← SIO[B]制御レジスタ[0]を選択
        out     (sio_b+1),a
        in      a,(sio_b+1)             ; ← 制御レジスタの状態を入力
        ld      c,a
        ld      a,1
        bit     3,b                      ; DCDA check
        jr      z,_sw_1
        ld      (data+0),a              ; <CN2> pin 9 の状態が bit 0 に入る
_sw_1:
        bit     5,b                      ; CTSA check
        jr      z,_sw_2
        ld      (data+1),a              ; <CN2> pin 10 の状態が bit 0 に入る
_sw_2:
        bit     3,c                      ; DCDB check
        jr      z,_sw_3
        ld      (data+2),a              ; <CN2> pin 12 の状態が bit 0 に入る
_sw_3:
        bit     5,c                      ; CTSB check
        jr      z,_sw_4
        ld      (data+3),a              ; <CN2> pin 11 の状態が bit 0 に入る
_sw_4:
        ret

        end
```

〈図5-7〉 パラレル・ポートを使わずに4ビット入力ポートを実現する方法



〈図5-9〉 パラレル・ポートを使わずに2ビット出力ポートを実現する方法



採用するのはポートが不足して無理であることがわかります．また，実際のシステム開発では，PA/PBのポートを8ビット幅まるまる使用して，拡張のための制御ラインをあと1本か2本だけ欲しいという状況によく直面します．そこで筆者が愛好している特別テクニックの出番となります．これは「シリアル・ポートの制御端子を利用すると4ビットの入力と2ビットの出力を拡張できる」というもので，知る人ぞ知るAKI-80の裏技です．

図5-7は，この必殺技で「AKI-80の入力を4ビット拡張」した例です．これに対応したのが**リスト5-4**(ADDIN.ASM)で，シリアル・ポートの謎めいた設定処理には当惑されるでしょうが，とにかく「オマジナイ」としてこの処理を埋め込んでおくと，パラレル・ポートを使用せずに「汎用入力ポート」を外付け部品なしに増設できるのです．この正体は「シリアル通信ポートのステータス・ビット」を利用したものなのですが，**図5-8**のように外部にデータ・セレクタの74HC151あたりを付けると，十分に実用になります．

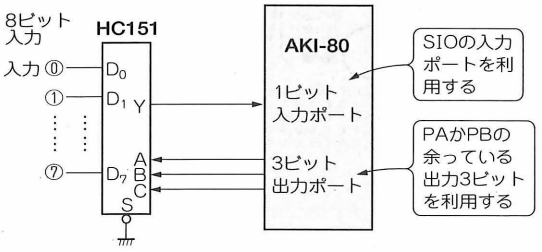
図5-9は，同様にこの必殺技で「AKI-80の出力を2ビット拡張」した例です．**リスト5-5**(ADDOUT.ASM)はこの部分に関連したソフト例で，ここでもシリアル通信の制御信号ビットを汎用出力として利用しています．なお，この場合には，出力されるほかのビットがシリアル・ポートの動作に関係していますから，「汎用として出力したいビット」と「もともと設定しているビット」とで論理和をとったデータをかならず書き込むように注意します．

μPD71055 NECのCMOS版のPIO．オリジナルはインテルの8255である．

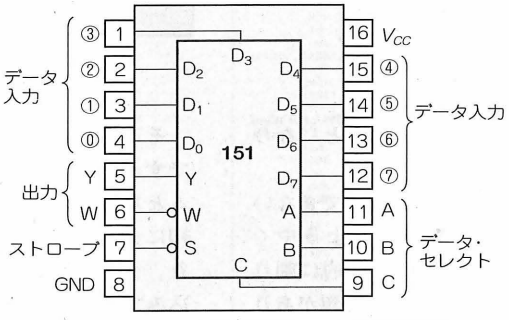
〈リスト5-5〉 出力を2ビット拡張するプログラム[ADDOUT.ASM]

```
;##### RAM Map #####
        org         8000h           ;← RAM領域に変数を定義
buffer  ds          2
data    ds          2

;##### I/O Map #####
sio_a   equ         0018h           ;← 内蔵SIOのアドレス定義
sio_b   equ         001ah

;##### RESET #####
        org         0000h           ;← リセットするとここから始まる
        ld          sp,09fffh       ;← スタックポインタをセット
        di                          ;← 割り込みを禁止
        jp           main           ;← メインに飛ぶ

;##### NMI #####
        org         0066h           ;← (NMIは来ないが一応入れておく)
        retn

;##### Main #####
main:
        ld          a,00011000b     ;← SIO[A]チャネル・リセット
        out         (sio_a+1),a
        ld          a,00011000b     ;← SIO[B]チャネル・リセット
        out         (sio_b+1),a
                                    ; (本来はここで各種の初期化)
                                    ;制御レジスタの設定は[buffer]へ
loop:
        call        port_out_a      ;← SIO[A]からのビット単位拡張出力
        call        port_out_b      ;← SIO[B]からのビット単位拡張出力
         jr         loop

port_out_a:
        ld          a,(data+0)      ;bit0に出力すべきデータがある
        rrca                        ;bit7 に移動
        ld          b,a
        ld          a,(buffer+0)    ;制御レジスタの下7ビットの内容
        and         01111111b       ;に
        or          b               ; MSBの出力データを添える
        ld          b,a
        ld          a,00000101b     ;← SIO[A]制御レジスタ[5]を選択
        out         (sio_a+1),a
        ld          a,b
        out         (sio_a+1),a     ;← これで出力(CN2 pin8)
        ret

port_out_b:
        ld          a,(data+1)      ;bit0に出力すべきデータがある
        rrca                        ;bit7に移動
        ld          b,a
        ld          a,(buffer+1)    ;制御レジスタの下7ビットの内容
        and         01111111b       ;に
        or          b               ;MSBの出力データを添える
        ld          b,a
        ld          a,00000101b     ;← SIO[B]制御レジスタ[5]を選択
        out         (sio_b+1),a
        ld          a,b
        out         (sio_b+1),a     ;← これで出力(CN2 pin13)
        ret

        end
```

〈図5-8〉 外部データ・セレクタによる入力の拡張



HC151のピン配置と真理値表



入力				出力	
セレクト			ストローブ		
C	B	A	S	Y	W
X	X	X	H	L	H
L	L	L	L	D_0	$\overline{D_0}$
L	L	H	L	D_1	$\overline{D_1}$
L	H	L	L	D_2	$\overline{D_2}$
L	H	H	L	D_3	$\overline{D_3}$
H	L	L	L	D_4	$\overline{D_4}$
H	L	H	L	D_5	$\overline{D_5}$
H	H	L	L	D_6	$\overline{D_6}$
H	H	H	L	D_7	$\overline{D_7}$

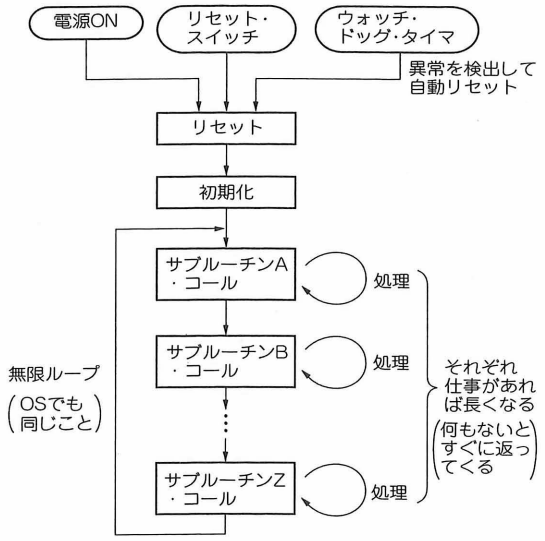
タイマと割り込みの活用

パラレル・ポートの次に検討するのは，東芝CPUに内蔵されたタイマの利用です．AKI-80に限らず，いわゆる組み込みマイコンのシステムというのは，一般に**図5-10**のような「**無限ループ**」の動作を永遠に続けています．ここで，無限ループの1周に必要な時間というのは，それぞれのサブルーチンの処理がどのように必要となるかという状況によって伸び縮みすることになります．つまり，このループの時間というのは可変というか変動してしまうので，「一定時間」の基準として使用することができません．そこで，タイマの登場となります．

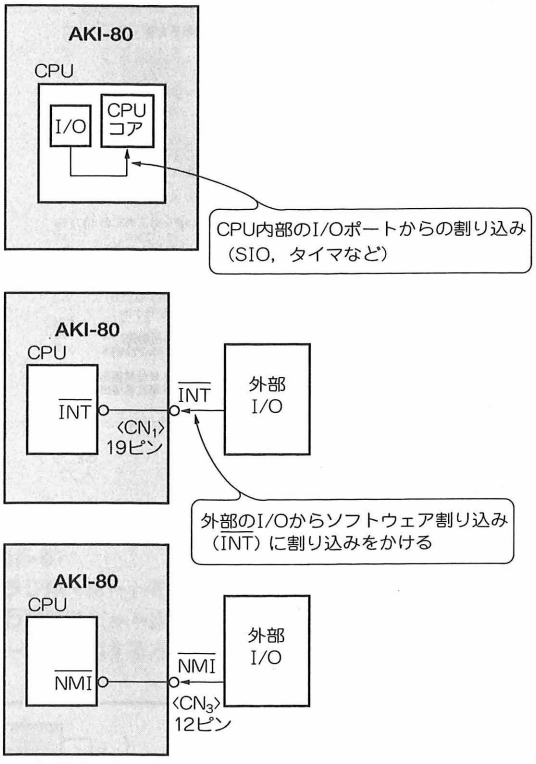
すでに紹介した例ですが，たとえば「LEDを等間隔で点滅させる」などという場合には，「ソフトウェア・タイマ」という簡易的な手法をよく用います．これは**リスト5-1**のように，メモリ中の変数を無限ループのメイン・ルーチン中でインクリメントし続けるというもので，厳密には完全な一定間隔ではないのですが，人間の時間感覚である「秒オーダ」の規模までカウントを積み上げると，メイン・ルーチン中のサブルーチンの処理時間の増減を平均して吸収してしまい，ほぼ等間隔のタイマとして使用することができます．たとえば**リスト5-1**の例では，(timer)という変数が256回のインクリメントでオーバフローすると(timer+1)がインクリメントする(100になると強制的にゼロに戻す)という仕組みですから，さらに上の桁である(timer+2)の1ステップの増加のたびに，メイ

ラッチ・パルス　ラッチにデータを取り込んで確定させるためのパルス信号．Dラッチであれば，この立ち上がりエッジで，Gラッチであればアクティブ・レベルである．

〈図5-10〉すべてのマイコン・システムは無限ループ



〈図5-11〉いろいろな割り込み



ンの無限ループを25600回ずつ回っていることになります．

このようなソフトウェア・タイマでは実現できないような高速/高精度の時間処理となると，ようやく「タイマ割り込み」が必要になります．一般的に割り込み処理には，**図5-11**のようないろいろな種類があります．「割り込みを活用して，初めてマイコン・システムとしては一人前」などといわれるように，この活用テクニックもそれぞれです．その一つ一つを詳細に検討するのは本書の目的ではありませんので，ここでは筆者が愛用するテクニックを紹介しておきましょう．その他の割り込み手法については，シリアル通信処理のところでも検討することにします．

リスト5-6(TIMER1.ASM)は，「もっとも無駄な1秒タイマ」の例，**リスト5-7**(TIMER2.ASM)はタイマ割り込みによる1秒タイマの例です．ここではいずれも，ほぼ1秒ごとにLEDが点滅するようになっていて，システムとして外部から見た場合には同等ですが，処理能力としては雲泥の違いがあります．この違いを理解することが，割り込みの正しい理解の最初の一歩です．つまり，**リスト5-6**の例では基準クロックからCPUの命令サイクル時間を割り出し，NOP命令をひたすら並べることで「1秒」という時間を作っています．

そこで，1秒ごとにLEDを点滅できるのは確かなのですが，「この他にはまったく何もできない」システムとなっています．これに対して，**リスト5-7**では最初にタイマ・ポートに割り込み時間を初期設定しておき，タイマというハードウェアからの1秒ごとの割り込みでLED処理を実行していますから，メイン・ループではすべての時間，ひたすらむだに足踏みを続けています．つまり，ここに他の処理をいろいろ詰め込んでも，1秒タイマの動作はまったく遅れずに「同時進行」してくれるというわけです．

ところで，**リスト5-7**の割り込みプログラムというのは，実はプロの視点からはちょっと問題がある簡易的な方法です．割り込みがかかったところで参照される「割り込みルーチン」で割り込み処理(ここではLEDの点灯処理)を行うという考え方のどこがいけないのかと思われる方は，まだ「割り込みの時間感覚」としてはちょっと甘いのです．

つまり，この例ではLEDの点滅というごく単純な処理ですが，一般に割り込みルーチン中の処理は，

- パラメータの退避と回復
- 必要な割り込み処理
- 関連した入出力処理
- 割り込みステータスの更新と表示

3ステート・ゲート　出力が「H」「L」「ハイ・インピーダンス」という3種類の状態を取りうるタイプの出力ゲート．バス回路に使う．

〈リスト5-6〉もっともムダな1秒タイマの例[TIMER1.ASM]

```
;##### RAM Map #####
        org     8000h           ;← RAM領域に変数を定義
led     ds      1

;##### I/O Map #####
pio_a   equ     001ch           ;← 内蔵PIOのアドレス定義

;##### RESET #####
        org     0000h           ;← リセットするとここから始まる
        ld      sp,09fffh       ;← スタックポインタをセット
        di                      ;← 割り込みを禁止
        jp      main            ;← メインに飛ぶ

;##### NMI #####
        org     0066h           ;← (NMIは来ないが一応入れておく)
        retn

;##### Main #####
main:
        ld      a,0cfh          ;Mode 3
        out     (pio_a+1),a
        ld      a,00000000b     ;0:Out / 1:In
        out     (pio_a+1),a
        xor     a
        ld      (led),a
loop:
        call    led_display     ;← ほぼ1秒たたないと帰ってこない
        jr      loop

led_display:
        ld      a,(led)
        xor     11111111b       ;← LEDデータを反転して
        ld      (led),a
        out     (pio_a),a       ;← ポートから出力
        call    nop_1sec        ;← ここでほぼ1秒,ひたすら待つ
        ret

nop_1sec:
        call    nop_100msec     ;[nop_100msec]を10回コールするので,
        call    nop_100msec     ;およそ1secかかる
        call    nop_100msec
        call    nop_100msec     ;(正確にはリターン等が余分にかかる)
        call    nop_100msec
        call    nop_100msec
        call    nop_100msec
        call    nop_100msec
        call    nop_100msec
        call    nop_100msec
        ret
```

```
nop_100msec:
        call    nop_10msec      ;[nop_10msec]を10回コールするので,
        call    nop_10msec      ;およそ100msかかる
        call    nop_10msec
        call    nop_10msec      ; (正確にはリターン等が余分にかかる)
        call    nop_10msec
        call    nop_10msec
        call    nop_10msec
        call    nop_10msec
        call    nop_10msec
        call    nop_10msec
        ret

nop_10msec:
        call    nop_1msec       ;[nop_1msec]を10回コールするので,
        call    nop_1msec       ;およそ10msかかる
        call    nop_1msec
        call    nop_1msec       ; (正確にはリターン等が余分にかかる)
        call    nop_1msec
        call    nop_1msec
        call    nop_1msec
        call    nop_1msec
        call    nop_1msec
        call    nop_1msec
        ret

nop_1msec:
        call    nop_100         ;[nop_100]を10回コールするので,
        call    nop_100         ;およそ1msかかる
        call    nop_100
        call    nop_100         ; (正確にはリターン等が余分にかかる)
        call    nop_100
        call    nop_100
        call    nop_100
        call    nop_100
        call    nop_100
        call    nop_100
        ret

nop_100:
        db   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0          ;NOP
        db   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0          ;= 1μs
        db   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0          ; 100発で
        db   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0          ;およそ
        db   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0          ;0.1ms
        ret

        end
```

〈リスト5-7〉タイマ割り込みによる1秒タイマの例[TIMER2.ASM]

```
;##### RAM Map #####
        org     8000h           ;← RAM領域に変数を定義
led     ds      1
timer   ds      1

;##### I/O Map #####
ctc_0           equ     0010h   ;← タイマのアドレス定義
pio_a           equ     001ch   ;← 内蔵PIOのアドレス定義

;##### RESET #####
        org     0000h           ;← リセットするとここから始まる
        ld      sp,09fffh       ;← スタックポインタをセット
        di                      ;← 割り込みを禁止
        jp       main           ;← メインに飛ぶ

;##### NMI #####
        org     0066h           ;← (NMIは来ないが一応入れておく)
        retn

;##### TIMER #####
        org     0070h
        dw       _timer_        ;← タイマ割り込み処理
_timer_:
        ex      af,af'          ;← レジスタを待避
        ld      a,(timer)
        inc     a
        ld      (timer),a
        cp      100             ;10msec * 100 = 1sec
        jr      c,_timer_pass
        xor     a
        ld      (timer),a
        ld       a,(led)
        xor     11111111b       ;← LEDデータを反転して
        ld      (led),a
        out     (pio_a),a       ;← ポートから出力
_timer_pass:
        ex      af,af'          ;← レジスタを復旧
        ei                      ;← 次の割り込みを許可
        reti

;##### Main #####
main:
        ld      a,70h           ;← 割り込みアドレスの設定
        out     (ctc_0),a
        ld      a,10100101b     ;Timer Mode
        out     (ctc_0),a
        ld      a,157           ;about 10msec
        out     (ctc_0),a
        ld      a,0cfh          ;Mode 3
        out     (pio_a+1),a
        ld      a,00000000b     ; 0:Out / 1:In
        out     (pio_a+1),a
        xor     a
        ld      (led),a
         ld     (timer),a
         im     2               ;内部I/O割り込みモード
        ei                      ;割り込み許可
loop:
        jr      loop            ;← メイン・ルーチンは何もない!!

        end
```

などがあります.

多くの場合，これはかなりの処理時間を必要として，これらを単純にズラズラと記述していると，「割り込み処理ルーチンの時間がかかって，終了していないのに別の割り込みや次の割り込みが発生する」という状況になりかねないのです.

そこで実際的な割り込み処理としては，筆者は**リスト5-8**(TIMER3.ASM)のような手法を愛用しています.これは簡単にいえば「割り込みルーチン内の処理は最低限にする」という，プロフェッショナルの間では常識となっていることの実践です.つまり，タイマからの割り込みルーチンでは「割り込みがあったよ」というメモリのフラグを立てるだけという最小限の仕事ですぐに抜け出してしまうのです.そして，メインの無限ループの中で定期的に呼び出すサブルーチンの冒頭で，このフラグの状態をチェックします.たいていの場合には「変化なし」で抜け出してしまいますから，全体としての処理はほとんど変わりませんが，タイマの変化があればメイン・ループの周期以内のレスポンスで対応できます.このような手法を用いるためには，メイン・ループがある程度の高速で回っていることが前提となりますが，もともとマイコン・システムのソフトはそのように設計していくものですから，何の問題もないのです.

無限ループ　プログラムが無限に同じループを回るようにジャンプ命令で元に戻るタイプのもの.

〈リスト5-8〉プロの使う割り込みルーチン[TIMER3.ASM]

```
;##### RAM Map #####
        org     8000h           ;← RAM領域に変数を定義
led     ds      1
timer   ds      1
flag    ds      1

;##### I/O Map #####
ctc_0   equ     0010h           ;← タイマのアドレス定義
pio_a   equ     001ch           ;← 内蔵PIOのアドレス定義

;##### RESET #####
        org     0000h           ;← リセットするとここから始まる
        ld      sp,09fffh       ;← スタック・ポインタをセット
        di                      ;← 割り込みを禁止
        jp      main            ;← メインに飛ぶ

;##### NMI #####
        org     0066h           ;← (NMIは来ないが一応入れておく)
        retn

;##### TIMER #####
        org     0070h
        dw      _timer_         ;← タイマ割り込み処理
_timer_:
        ex      af,af'
        ld      a,1
        ld      (flag),a        ;← フラグを立てるだけ!!!
        ex      af,af'
        ei                      ;← 次の割り込みを許可
        reti

;##### Main #####
main:
        ld      a,70h           ;← 割り込みアドレスの設定
        out     (ctc_0),a
        ld      a,10100101b     ; Timer Mode
        out     (ctc_0),a
        ld      a,157           ;about 10msec
        out     (ctc_0),a
        ld      a,0cfh          ;Mode 3
        out     (pio_a+1),a
        ld      a,00000000b     ;0:Out / 1:In
        out     (pio_a+1),a
        xor     a
        ld      (led),a
        ld      (timer),a
        ld      (flag),a
        im      2               ;内部I/O割り込みモード
        ei                      ;割り込み許可
loop:
        call    timer_check
        jr      loop

timer_check:
        ld      a,(flag)
        cp      0
        ret     z               ;← 割り込みフラグOFFならすぐ帰る
        xor     a
        ld      (flag),a
        ld      a,(timer)
        inc     a
        ld      (timer),a
        cp      100             ;10msec * 100 = 1sec
        ret     c               ;← 1sec未満ならすぐ帰る
        xor     a
        ld      (timer),a
        ld      a,(led)
        xor     11111111b       ;← LEDデータを反転して
        ld      (led),a
        out     (pio_a),a       ;← ポートから出力
        ret

        end
```

ここまでは内蔵タイマからの割り込みでしたが，AKI-80はさらに外部に増設した割り込みにまで拡張することもできます．この場合，$\overline{\text{INT}}$端子と$\overline{\text{NMI}}$端子を使用して，ここに外部からの割り込み信号を与えます．ただし，各種の内部ポートからの割り込みとの共存は面倒なところがあり，あまりビギナーにはお勧めできません．なんでもかんでもポートとのやりとりには割り込みを使うという設計をする人もいますが，筆者は「必要なところにだけ割り込みを使い，スピードや精度のいらない部分はソフトウェア的に監視(**ポーリング**)」という作戦を採用するようにしています．

なお，AKI-80の内蔵タイマは全部で4ポートあり，組み合わせて16ビットのプログラマブル・カウンタとして使うというような応用もあります．ステッピング・モータの制御や，信号発生器としての利用には役立つ機能ですが，あまりに類書での記述が多いので，本書ではこの部分については省略することにします．

シリアル・ポートとMIDI

パラレル・ポートとタイマについて検討してきましたが，いよいよ残ったのがシリアル・ポートということになります．通常はシリアルといえばRS-232-CかRS-422-Cというのが定番なのですが，本書では筆者の趣味から，テーマをMIDIに限定しています．MIDIもRSも，具体的な信号インターフェース部分に使うICが違うだけで，AKI-80システムとしてはハードもソフトも変わらないので，実用のための一般性としては問題ありません．

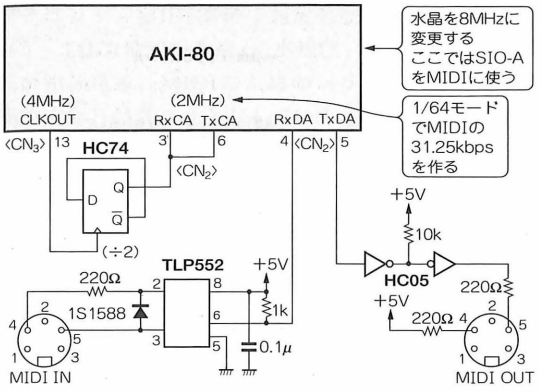
第3章のAppendixでMIDI規格と関連したSMF規格を詳細に紹介しましたが，MIDI(Musical Instrument Digital Interface)とは，もともとは鍵盤方式の電子楽器の演奏情報を伝送するための国際共通規格としてスタートしたものです．現在では電子的な弦楽器，管楽器，打楽器などの演奏情報，さらにはコンピュータと接続したりセンサやライティングの情報転送にも使用されています．**図5-12**は，AKI-80でMIDI処理を行うための標準的なハードウェア部分の回路図で，これはMIDI規格そのものの仕様の説明ともなっています．MIDIは電気的には，オープン・コレクタのドライバでバッファリングされた＋5 V電源のカレント・ループ信号であり，受信側はフォトカプラで電気的に分離して誤動作対策をとっています．

初期のMIDIはPC900(シャープ)などの低速フォトカプラを用いていたのですが，現在では**図5-12**のように，高速フォトカプラTL552(TI)などを用いるのが一般的です．

ところで，**図5-12**のようにわざわざAKI-80の外部にクロック分周回路を置くのはあまり美しくありません．東芝CPUに内蔵されている**プリスケーラ**を使うことはできないのかという疑問が出てくるようであれば，なかなかの実力です．しかし，たまに「内蔵プリスケーラを使って，SIOを[1/1モード]で使う」という方法が書いてある本もありますが，これは残念ながらごく単純なMIDI送信システム(たとえば自動演奏装置)ぐらいにしか使えません．1/1モードはデータ量が多くなってくると動作が不安定になる場合があり，とくに1ビットでもデータが化けると情報の意味がなくなるMIDI受信においては，経験上，恐ろしくて使えないのです．1/16モードとか1/64モードであればまったく問題ありませんから，「MIDIではSIOのクロックには2 MHzか500 kHzを入れる」と覚えておくといいでしょう．

割り込みルーチン　割り込みが発生した際に行う処理を記述したサブルーチン．ここからのリターンは通常のリターンでなく「return from interrupt」を用いる．

〈図5-12〉AKI-80でMIDI処理を行う標準回路



MIDIをソフトウェア的に見ると，まずシリアル通信の下層の規格として，

- 非同期シリアル(データ信号線のみで，クロック信号線はない)
- データ8ビット，スタート/ストップ各1ビット形式
- データ転送速度(ボーレート)：31.25 kbps

ということになっていて，RS-232-Cの19.2 kbpsよりも高速です．そしてデータそのもののプロトコルとして，**リスト5-9**のように英語で規定された複雑な内容が厳密に定義されています．

たとえばMIDIでは「ある高さの鍵盤がある強さで弾かれた」という一つの演奏情報(イベント)を，3バイト(ステータス＋鍵盤番号＋打鍵強度)のセットとして表現します．31.25 kbpsの1ビット幅は32 μs，スタートとストップを含めた10ビットで320 μsのデータ幅ですから，この3バイトの情報がほぼぴっしりと並んで転送されると(間隔は自在に空いてもよい)，一つの演奏情報でおよそ1 msの時間がかかります．そこで，ピアノの鍵盤を「コード(和音)」として同時にまとめて10鍵弾いた場合，これらの情報は同時といっても順に転送されて，全体として10 ms以上かかることになります．

人間の時間感覚としては，50 msまで遅延するとかなり「聴こえて」くるために(専門家はもっと敏感ですが)，このような遅延を短縮するための「**ランニング・ステータス**」という規約があります．つまり，連続して同じステータスの情報を送るときにはステータス・バイトを省略してもよいというものです．そこでMIDI処理ソフトとしては，常にこの省略があるかもしれないという両刀の構えで対処する必要があります．これはシステム的にいえば，時間的なメリットを稼ぐために，ソフトウェアに負担させて性能を獲得したという例となります．

リスト5-10(THRU.ASM)は，**図5-12**のAKI-80でのMIDI標準回路を用いて，「MIDI入力をMIDI出力としてスルーさせる」というプログラム例です．入力をそのまま出力するだけでは意味がないではないかといわれそうですが，実はこのサンプルには，AKI-80でのMIDI処理のエッセンスがすべて凝縮されて入っています．あまり派手ではありませんが，筆者のノウハウの詰まった，ある意味で本書の中で最大の秘伝の伝授

〈リスト5-9〉MIDIデータ・フォーマット

```
MIDI DATA FORMAT

For standard MIDI messages, there is a clear concept that one device
is a "transmitter" or "master", and the other a "receiver" or "slave".
Messages take the form of "status" bytes, followed by data bytes.

Status bytes are marked by bit 7 being 1.  All data bytes must
contain a 0 in bit 7, and thus lie in the range 0 - 127.

MIDI has a logical channel concept.  There are 16 logical channels,
encoded into bits 0 - 3 of the status bytes of messages for
which a channel number is significant.

status byte   meaning         data bytes

0x80-0x8f     note off        2 - 1 byte pitch, followed by 1 byte velocity
0x90-0x9f     note on         2 - 1 byte pitch, followed by 1 byte velocity
0xa0-0xaf     key pressure    2 - 1 byte pitch, 1 byte pressure (after-touch)
0xb0-0xbf     parameter       2 - 1 byte parameter number, 1 byte setting
0xc0-0xcf     program         1 byte program selected
0xd0-0xdf     chan. pressure  1 byte channel pressure (after-touch)
0xe0-0xef     pitch wheel     2 bytes gives a 14 bit value, least significant
                                7 bits first

For all of these messages, a convention called the "running status
byte" may be used.  If the transmitter wishes to send another message
of the same type on the same channel, thus the same status byte, the
status byte need not be resent.

Also, a "note on" message with a velocity of zero is to be synonymous
with a "note off".  Combined with the previous feature, this is intended
to allow long strings of notes to be sent without repeating status bytes.

( from: lee@uhccux , eharnden@auvm        4/07/89 midi-intro
        lee@uhccux , eharnden@auvm        4/07/89 messages.table
        lee@uhccux , eharnden@auvm        4/07/89 status.table
        lee@uhccux , eharnden@auvm        4/07/89 notes.table   )
```

ポーリング　相手の状態をいちいち確認して，OKであれば処理を行い，busyであれば処理をパスして行わないという方式のこと．

〈リスト5-10〉MIDI入力をMIDI出力としてスルーさせるプログラム[THRU.ASM]

```
;##### RAM Map #####
        org     8000h                   ;← RAM領域に変数を定義
rx_fifo         ds      2048
tx_fifo         ds      2048
rx_top          ds      2
rx_end          ds      2
tx_top          ds      2
tx_end          ds      2
rsb             ds      1
dcb             ds      1
channel         ds      1
keyno           ds      1

;##### I/O Map #####
sio_a           equ     0018h
sio_b           equ     001ah           ;← 内蔵SIOのアドレス定義

;##### RESET #####
        org     0000h                   ;← リセットするとここから始まる
        ld      sp,09fffh               ;← スタック・ポインタをセット
        di                              ;← 割り込みを禁止
        jp      main                    ;← メインに飛ぶ

;##### INT / NMI #####
        org     0020h                   ;← MIDI受信割り込みルーチン
        dw      _midi_
_midi_:
        ex      af,af'
        exx
        ld      de,(rx_top)
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        in      a,(sio_a+0)             ;← シリアル・ポートからデータ入力
        ld      (hl),a                  ;← これを受信FIFOに積む
        inc     de
        res     3,d
        ld      (rx_top),de
        exx
        ex      af,af'
        ei
        reti
        org     0066h
        retn

;##### Main #####
main:
        ld      hl,08000h
        ld      a,0a0h
_ram_clear_loop:
        ld      (hl),0                  ;← RAM領域をゼロクリア
        inc     hl
        cp      h
        jr      nc,_ram_clear_loop
        ld      a,00011000b             ;Channel Reset B
        out     (sio_b+1),a
        ld      a,00000100b             ;Resister Point = 4
        out     (sio_b+1),a
        ld      a,11000100b             ;Mode
        out     (sio_b+1),a
        ld      a,00000001b             ;Resister Point = 1
        out     (sio_b+1),a
        ld      a,00000000b             ;Interrupt Mode
        out     (sio_b+1),a
        ld      a,00000010b             ; Resister Point = 2
        out     (sio_b+1),a
        ld      a,20h                   ;Vector Address
        out     (sio_b+1),a
        ld      a,00011000b             ;Channel Reset A
        out     (sio_a+1),a
        ld      a,00000100b             ; Resister Point = 4
        out     (sio_a+1),a
        ld       a,11000100b            ;Mode
        out     (sio_a+1),a
        ld      a,00000001b             ;Resister Point = 1
        out     (sio_a+1),a
        ld      a,00010000b             ;Interrupt Mode
        out     (sio_a+1),a
        ld      a,00000101b             ;Resister Point = 5
        out     (sio_a+1),a
        ld      a,01101000b             ;Transmit Start
        out     (sio_a+1),a
        ld      a,00000011b             ;Resister Point = 3
        out     (sio_a+1),a
        ld      a,11000001b             ; Receive Start
        out     (sio_a+1),a
        im      2
        ei
        in      a,(sio_a+0)             ; dummy read
        im      2                       ;内部I/O割り込みモード
        ei                              ;割り込み許可
loop:
        call    tx_data_check
        call    rx_data_check
        jr      loop

tx_data_check:
        ld      de,(tx_end)
        ld      hl,(tx_top)
        and     a                       ;CY <-- 0
        sbc     hl,de
        ret     z                       ;← 新しい送信データがなければパス
        ld      a,00000000b             ; Resister Point = 0
        out     (sio_a+1),a
        in      a,(sio_a+1)
        bit     2,a
        ret     z                       ;← 送信OKでなければパス
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      a,(hl)                  ;送信FIFOからデータを取り出して
        out     (sio_a),a               ; MIDIより送信する
        inc     de
        res     3,d
        ld      (tx_end),de
        ret

tx_fifo_set:
        ld      de,(tx_top)
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      (hl),b                  ;[B]のデータを送信FIFOに積む
        inc     de
        res     3,d
        ld      (tx_top),de
        ret
rx_data_check:
        ld      de,(rx_end)
        ld      hl,(rx_top)
        and     a                       ; CY <-- 0
        sbc     hl,de
        ret     z                       ;← 受信FIFOが空ならパス
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        ld      b,(hl)
        inc     de
        res     3,d
        ld      (rx_end),de
        bit     7,b                     ; MSB=0 ならデータ・バイト
        jr      z,running               ; running
        ld      a,b
        cp      0f8h
        ret     nc
        cp      0f0h
        jr      c,statusbyte
        xor     a
        ld      (rsb),a                 ;RSB = running status buffer
        ret
statusbyte:
        ld      a,b                     ;ステータス・バイトは
        and     00001111b
        ld      (channel),a             ;チャネルと
        ld      a,b
        and     11110000b
        ld      (rsb),a                 ;RSBに分離しておく
        xor     a
        ld      (dcb),a                 ;DCB = data count buffer
        ret
running:
        ld      a,(rsb)
        cp      0
        ret     z                       ;無効ステータス時のデータは無視
        cp      0c0h
        jr      z,_2byte
        cp      0d0h
        jr      z,_2byte                ;2byte形式は[C*]と[D*]だけ
        ld      a,(dcb)
        cp      0
        jr      nz,_3byte               ;3byte目なら処理に飛ぶ
        inc     a
        ld      (dcb),a                 ;3byte形式の2byte目の場合はここ
        ld      a,b
        ld      (keyno),a
        ret
_2byte:
        ld      c,b
        ld      a,(rsb)
        ld      b,a
        ld      a,(channel)
        or      b
        cal     tx_fifo_set             ;Status Byteを復元して送信FIFOに
        ld      b,c
        call    tx_fifo_set             ;Data Byteを送信FIFOに積む
        ret
_3byte:
        xor     a
        ld      (dcb),a
        ld      c,b
        ld      a,(rsb)
        ld      b,a
        ld      a,(channel)
        or      b
        call    tx_fifo_set             ; Status Byteを復元して送信FIFOに
        ld      a,(keyno)
        ld      b,a
        call    tx_fifo_set             ;2byte目のData Byteを送信FIFOに積む
        ld      b,c
        call    tx_fifo_set             ;3byte目のData Byteを送信FIFOに積む
        ret

        end
```

なのです．

なお，このサンプルでは，「MIDIエクスクルーシブ」や「リアルタイム情報」については処理を簡略化しています．これは，そのような種類のMIDI情報をまったく使わない筆者にとって，他の処理を重くしてまで判定ルーチンを組み込む必要がないからです．当然ですが，「**MIDI準拠**」という電子楽器の製品を開発して販売するプロの場合には，このような簡便型では話になりません．

さて，**リスト5-10**は，初めて眺める人にとってはちょっと理解しにくいテクニカルなものとなっています．ここには前述の割り込み処理のノウハウ，受信お

RS232Cプリスケーラ　シリアル通信の一つの規格であるRS232C信号を発生するために，システム・クロックからRS232Cに必要なクロックを発生するための分周回路．

よび送信の両方で行っている**FIFOバッファリング**のテクニック，MIDI情報の判定処理のためのテンプレート，MIDI情報の生成/送出処理，メイン・ルーチンへの組み込み，MIDIのためのシリアル・ポートの初期設定などがズラリと並んでいます．なお，東芝CPUに固有の注意点として，シリアル・ポートのAとBとは完全に対等ではなくて，この例のように実際にはシリアル・ポートAのみを使用する場合にも，初期設定ルーチンではシリアル・ポートBの制御レジスタに対する設定が必要となります．

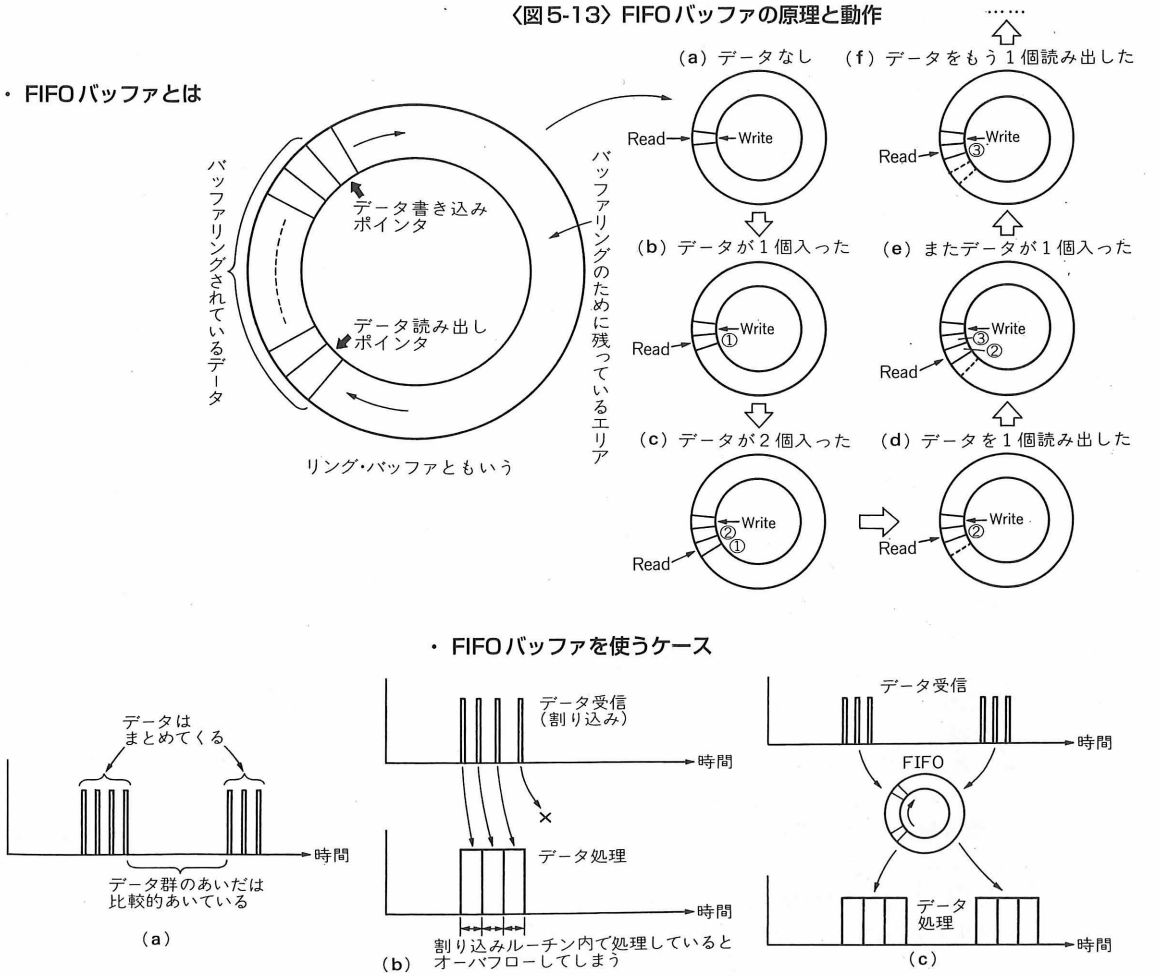
これは，両者の「割り込みベクトルの設定」はシリアル・ポートBの制御レジスタに対して行うと規定されているためです．

FIFO処理とは，FIFO(First In, First Out) バッファと呼ばれるテクニックによって，「時間的に粗密の分布がばらつくイベントの処理」を確実に行うものです．つまりMIDIでは，ある瞬間には和音演奏としてほぼ同時に多量のデータが発生し，また音符と音符の間ではかなりの期間(といっても人間には一瞬ですが)にわたってデータがないという状態を繰り返します．ところで，MIDIでは「鍵盤ON」と「鍵盤OFF」の情報はまったく別のものですから，たとえ1バイトでも情報を受け損えば，「音が鳴りっぱなしで止まらない」という状況になります．

パソコン通信などでは，このようなデータの欠落は「文字化け」として，その文字だけつぶれたりしますが，これはMIDIでは「絶対に許されない」ことなのです．そこで，受信割り込み処理ルーチン内でMIDI情報の解釈を行うなどということはできませんから，FIFOの登場となるのです．

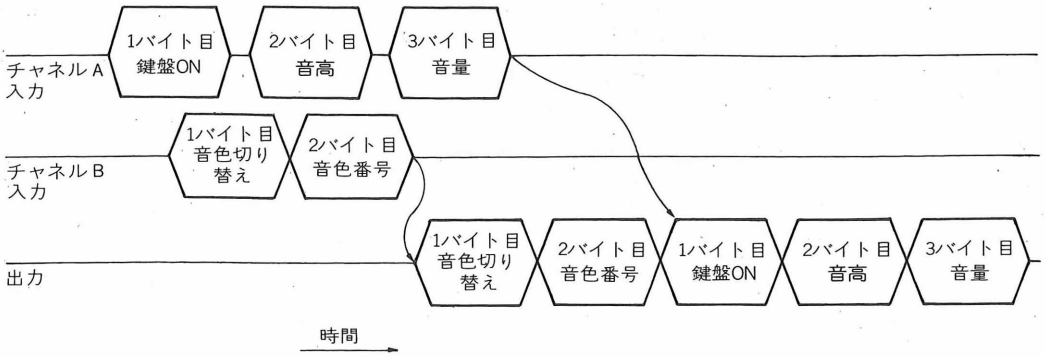
図5-13はFIFO処理の動作について説明したものです．これと**リスト5-10**の該当部分とを照らし合わせて，なんとか雰囲気だけでも理解してみましょう．なお，パソコン上のC言語プログラムなどでFIFO処理を実現する場合，FIFOメモリ内のアドレスは単純に比較と加算を行うのですが，Z80は16ビット加算に弱いた

〈図5-13〉FIFOバッファの原理と動作



ランニング・ステータス　MIDI規格の中で，同じステータス・メッセージが続くときに限って，先頭のステータス・バイトを省略してもよいと規定されていること．多量の演奏情報を高速に転送するために利用される．

〈図5-14〉MIDIマージの難しさ



めに，リストのようにメモリ空間内でのFIFOメモリの位置を限定(先頭アドレスの下位がかならずゼロになる)して，少しでも高速に処理できるようなワザを駆使しています．また，MIDI受信には割り込みを使いますが，MIDI送信については「暇な時に送信する」という立場を取っています．ここは意見の分かれるところなのですが，筆者は自信をもって，このスタイルで数多くのMIDI実用システムを製作してきました．

まず最初に，**リスト5-10**のプログラム冒頭部分のMIDI受信割り込みルーチンでは，シリアル・ポートからの内部割り込みによって，受信データをそのまま(内容の解釈を行わずに)FIFOバッファに積んでいます．

「FIFOに積む」という操作は，簡単に整理してみると，

- TOP(書き込み)ポインタを利用してFIFOバッファのアドレスを生成
- データを取り込む
- FIFOに書き込む
- TOPポインタをインクリメント(オーバフローしたらゼロに戻す)

ということに尽きます．これを文字通りにZ80で行うとステップ数がかなり必要となるため，多少テクニカルに実現しているだけです．

また，メイン・ループ内にあるMIDIチェック(処理)ルーチンでは，まず最初にFIFOバッファに「まだ処理していないデータが残っているか」というチェックがあるだけで，あとの処理手順はほぼ対応しています．

「FIFOに積む」と対比して「FIFOチェック」の操作を整理してみると，

- TOPポインタとEND(読み出し)ポインタを比較する
- 一致していたら何もしないですぐにリターンする
- 不一致なら新しいイベントがあるので，ENDポインタをインクリメント(オーバフローしたらゼロに戻す)
- ENDポインタを利用してFIFOからデータを取り出す
- このデータに対する解釈・処理に入る
- 解釈・処理の適当なところでリターンする

ということになります．

リスト5-10には，同様にMIDI送信についてもいきなりシリアル・ポートから出力せずに，送信用のFIFOバッファに積んで，また別のルーチンでこのFIFOを参照してからMIDI出力するようになっています．これは一見するとむだなことのように思えるのですが，実はそうではないのです．

新しいMIDIイベントは1バイト単位でなく，1メッセージを構成する2バイトないし3バイトで形成されています．すると，新しいイベントの送信のところでいきなりシリアル送信しようとすると，シリアル・ポートの「送信ビジー」を監視して，そこで完全に足踏みする必要があるのです．

これは「メイン・ルーチンには速やかに戻る」という原則に違反していますから，システムによってはバグの元となります．そこで，レスポンスとしては平均して遅くなるのを了解した上で，すべてのMIDI送信イベントをいったんFIFOに積み，これをメイン・ループから呼ばれる「MIDI送信チェック」ルーチンで，わざわざ1バイトずつバラして送信するのです．

ここでは，冒頭で「FIFOに送るべきデータがあるか」というチェックとともに，「シリアル送信ポートはビジーではないか」というチェックも行います(ビジーなら待たずに，次回のチェックまで持ち越してとりあえずリターンする)．ここのところがポイントで，むだな足踏みがかなり減るのです．

リスト5-11は，**図5-12**のMIDIポートをシリアル・ポートA／Bの2ポートの両方に拡張して，MIDIポートを独立した2系統に増設した場合の中核部分で，ソフトウェアとしては「二つのMIDI入力をマージ(合流)させる」という処理を行っています．このMIDIマージという操作は一見すると簡単そうですが，実は

MIDI準拠　MIDI規格に準拠しているということ．

〈リスト5-11〉二つのMIDI入力をマージするソフトウェア(一部分)

```
;##### RAM Map #####
                org     8000h           ;← RAM領域に変数を定義
rx_fifo_1       ds      2048
tx_fifo         ds      2048
rx_fifo_2       ds      2048
rx_top_1        ds      2
rx_end_1        ds      2
rx_top_2        ds      2
rx_end_2        ds      2
tx_top          ds      2
tx_end          ds      2
rsb1            ds      1
dcb1            ds      1
channel1        ds      1
keyno1          ds      1
rsb2            ds      1
dcb2            ds      1
channel2        ds      1
keyno2          ds      1

;##### INT / NMI #####
                org     0020h           ;← MIDI受信割り込みルーチン(1)
                dw      _midi1_
_midi1_:
                ex      af,af'
                exx
                ld      de,(rx_top_1)
                ld      a,10000000b
                or      d
                ld      h,a
                ld      l,e
                in      a,(sio_a+0)     ;← シリアル・ポートからデータ入力
                ld      (hl),a          ;← これを受信FIFOに積む
                inc     de
                res     3,d
                ld      (rx_top_1),de
                exx
                ex      af,af'
                ei
                reti

                org     0080h           ;← MIDI受信割り込みルーチン(2)
                dw      _midi2_
_midi2_:
                ex      af,af'
                exx
                ld      de,(rx_top_2)
                ld      a,11000000b
                or      d
                ld      h,a
                ld      l,e
                in      a,(sio_a+0)     ;← シリアル・ポートからデータ入力
                ld      (hl),a          ;← これを受信FIFOに積む
                inc     de
                res     3,d
                ld      (rx_top_2),de
                exx
                ex      af,af'
                ei
                reti

;##### Main #####

loop:                                   ;← メインからの呼び出し部分
                call    rx_data_check_1 ;← [SIO-A]データを送信FIFOに積む
                call    tx_data_check
                call    rx_data_check_2 ;← [SIO-B]データを送信FIFOに積む
                call    tx_data_check
                jr      loop
```

MIDIメッセージの規約を考えるとなかなかの強敵であることがわかります．つまり，**図5-14**のように，二つのシリアル・ポートから到着したデータをそのまま「混ぜて」MIDI出力したのでは，MIDI情報としてまったく意味のないものとなってしまい，接続された電子楽器はまったくデタラメに動作してしまうのです．そこで，

- それぞれのMIDI入力をバツファリングして貯めておく
- それぞれのMIDIデータがメッセージとして完結してから解釈する
- 解釈結果が正常であれば，メッセージ単位で送信FIFOに積む

という操作を行うことになります．

ここで，前述の「送信にもわざわざFIFOを使う」という処理が大きく役立ち，**リスト5-11**のような単純なプログラムで完全なMIDIマージ動作を実現できます．

図5-15は，さらに外部にシリアル通信用LSIのμPD71051(8251のCMOS版)を追加して，MIDIを3ポートに増設した回路例です．こうなると，内部シリアル・ポートだけでなく，割り込み端子としてIRQかNMIを使用する必要がでてきます．また8251には独特の初期設定テクニックがありますが，ここではAKI-80がメインなので，とくに詳しい紹介は省略しますから，興味のある方は後述のMIDIマージャの事例紹介のリストを参考にしてください．

BASM80とKUROKOによる開発環境との比較

AKI-80システムの設計について，いくつかのポイントを紹介しましたが，ここで本章の最後に，筆者が「XA80と自作ROMエミュレータ」という開発環境(本書の執筆のために新規にトライしてみたもの)を使う以前の開発環境と比較しておくことにしましょう．これは，次章で紹介する各種のAKI-80応用機器の大部分の開発で多くの実績のあるもので，ROMエミュレータの「KUROKO2」という製品と付属してきたBASM80という**マクロアセンブラ**です．

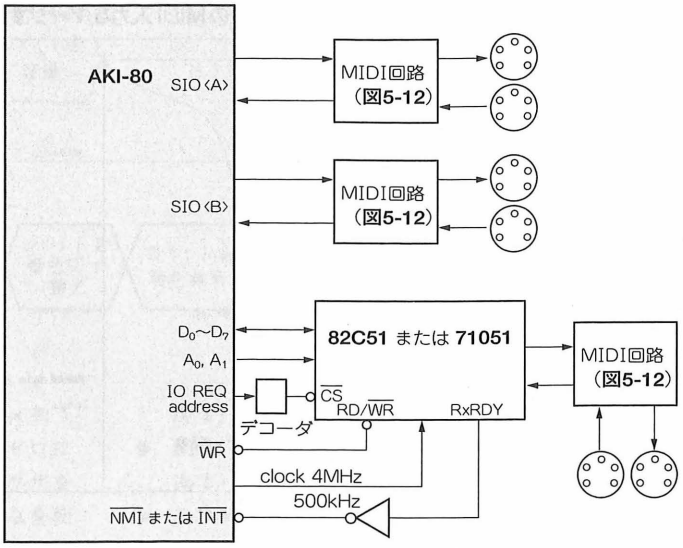
まず，リロケータブル・アセンブラ(BASM80)とアブソリュート・アセンブラ(XA80)との違いについては，筆者は使用上，この差を実感したことは一度もありません．というのも，対象として開発しているシステムがすべて「最終的にROMに焼く組み込みシステム」であり，プログラム・モジュールも複数に分割することのない「1本プログラム」ばかりだからです．これが簡単なOSの元でRAMにロードされるものであったり，複数のオブジェクト・モジュールを最終的にリンクしてメモリ空間に配置する大規模ソフトウェアであれば別なのですが，AKI-80を使って身軽に実現するようなシステムでは，この点でXA80が不便で困るということは少ないように思います．

次に「デバッグ性」ですが，これはもう，アマチュア指向とプロ仕様とでは土俵が違うということです．KUROKOとBASM80の組み合わせでは，その気になれば，ソース・レベルでデバッグを行えるほどの強力な機能を提供しています．もっとも筆者の場合，KUROKOをもっぱらROMエミュレータのモードで使っていましたから，このデバッガ機能はまったく知りません．結局は「プログラムをロードして，せーの!でリセット」ということの繰り返しですべてのシステムをちゃんと開発してしまったのです．

それでは，両者のアセンブル時間の比較をしてみましょう．当然ですが，このような「実験」はまず道具を作るところから始まります．ここでは簡単のために一連の処理をMS-DOSのバッチ・プログラムとして扱

FIFOバッファリング　先入れ先出し(first-i first-out)方式によってバッファリングすること．集中的に届いた情報をとりあえずここに溜めておいて，あとから暇な時に順に処理するための緩衝方式．

〈図5-15〉
MIDIを3ポート以上使用する場合



3ポート目はAKI-80のデータ・バスを外部に引き出して，82C51などのUARTを使用することになる．
（かなりのハード増設が必要となる）

〈リスト5-12〉 現在のシステム時計情報を秒単位で表示するプログラム[TIMER.C1]

```
#include <time.h>

time_t  ltime;

main(){
        time(&ltime);
        printf("\n\tTime = %ld (sec)\n",ltime);
}
```

〈リスト5-13〉 バッチ・プログラムT1.BAT

```
echo off
echo ◆[T1.BAT]:XA80での実験を行います◆ >result
timer >>result
xa80 test,,,; >>result
timer >>result
```

〈リスト5-14〉 BASM80の実験のためのプログラム T2.BAT

```
echo off
echo ◆[T2.BAT]:BASM80での実験を行います◆ >result
timer >>result
basm80 test -z -l -sI >>result
blink test -p:0 -d:8000 -htest >>result
timer >>result
```

〈リスト5-16〉 自作ROMエミュレータのためのプログラム T3.BAT

```
echo off
echo ◆[T3.BAT]:自作ROMエミュレータに転送します◆ >result
timer >>result
copy test.hex transfer.hex >>result
trans2 >>result
timer >>result
```

〈リスト5-17〉 KUROKOのためのプログラム T4.BAT

```
echo off
echo ◆[T4.BAT]:KUROKU2に転送します◆ >result
timer >>result
rome Rtest.hex E0000,0fff >>result
timer >>result
```

〈リスト5-15〉 実験結果

```
◆[T1.BAT]:XA80での実験を行います◆

    Time = 831853209 (sec)

Pass1....
Pass2....

No Errors

        Time = 831853216 (sec)  →  処理時間 = 7 sec

=====================================================

◆[T2.BAT]:BASM80での実験を行います◆

        Time = 831853886 (sec)

PASS-1  Make symbol table
PASS-2  Make object code

No Fatal error(s)

<ｾｸﾞﾒﾝﾄﾛｹｰｼｮﾝ>
 Segment    Start    End      Length
 7(CSEG)      0H     C32H     C33H

        Time = 831853890 (sec)  →  処理時間 = 4 sec
```

〈リスト5-18〉 実験結果

```
◆[T3.BAT]:自作ROMエミュレータに転送します◆

    Time = 831855533 (sec)

        1個のファイルをコピーしました.

    ##########[transfer.hex]-->ROM Emulator##########

.............................................................
....... transfer finished.

    Time = 831855534 (sec)      →処理時間=1 sec

=============================================================

◆[T4.BAT]:KUROKU2に転送します◆

    Time = 831857278 (sec)

Load (HEX) 0000-0C32
ROMｴﾐｭﾚｰﾀと接続中です.(<ESC>で中止します)

Ratio:10    12480[ﾊﾞｲﾄ]

ﾃﾞｰﾀ転送を開始しました.(0000-0FFF)
OO
ﾃﾞｰﾀ転送が完了しました.

    Time = 831857284 (sec)      →処理時間=6 sec
```


マクロアセンブラ　アセンブラの中で，高級言語に近い多種のマクロ言語などの定義を認めたより高度なアセンブラのこと．

いたいので，まず「時間を表示する」というストップウォッチが必要になります．**リスト5-12**は，このために3分で作った単純な「現在のシステム時計情報を秒単位で表示する」というプログラム[TIMER.EXE]のCソース・プログラム[TIMER.C]です．そして**リスト5-13**は，XA80の実験のためのバッチ・プログラム[T1.BAT]，**リスト5-14**はBASM80の実験のための[T2.BAT]です．ここでは，サンプルとしてそこそこの長さの，まったく同じAKI-80プログラム(ソースが約12 Kバイト，インテルHEXのサイズが約7KB)をかりに作成し，BASM80とXA80とでアセンブルを(BASMではリンクまで)行い，インテルHEXファイルを作成しています．

条件を揃えるために，それぞれ必要なツールはすべて同じディレクトリ内に置いて，PATH経由で探しに行かないようにしました．そして，この結果が**リスト5-15**というわけです．7秒と4秒とXA80のほうが2倍弱ほど遅いものの，筆者の体感としては，パソコンの速さのために実質的な差はないという印象です．

そして，**リスト5-16**のように同様の実験用バッチを作って，XA80のアセンブル結果を自作ROMエミュレータに転送した時間([T3.BAT])と，**リスト5-17**のBASM80のアセンブル結果をRS-232-C(12480bps)でKUROKOにダウンロードした時間[T4.BAT]とを比較したのが，**リスト5-18**にまとめた結果です．

ここでは，**リスト4-11**(p.37)で製作したROMエミュレータへのダウンロード・プログラムはそのまま返ってこないために，ロードしてターゲットをリセットしたらすぐに終了という改訂版で実験しています．

こうなると，いかに4ビット単位でわざわざ転送しているとはいえ，シリアルとの差が「6倍も速い」という体感として見えてきます．

この実験では小さなサイズの転送ですが，たとえば10倍の規模のプログラムであれば，10秒と1分という差となります．あくまで対象が「ごく簡単なAKI-80応用システム」と限定されているとはいえ，結果として，筆者はこれまで愛用してきたKUROKOを倉庫に片づけてしまうということになったのです．

第6章　AKI-80による音楽情報処理機器の開発

ファミコン用データ入力機器の改造やMIDI関連機器を作る

本章では，これまでに筆者が製作したAKI-80システムの実例の中から，参考になりそうなものをいくつか紹介します．

対象はすべて，筆者が作曲/研究のために製作したComputer Music関連のマシンです．これらの実例のテーマはたまたま音楽ですが，すでに述べたように，AKI-80を活用するというポイントとしてはほかのあらゆる分野にも活用できる事例集であろうと，数ある中から厳選したものを並べてみましたので，どうぞ活用してください．

ファミコン用PowerGlove改造マシン2題

● MIDIコントロール・グローブ

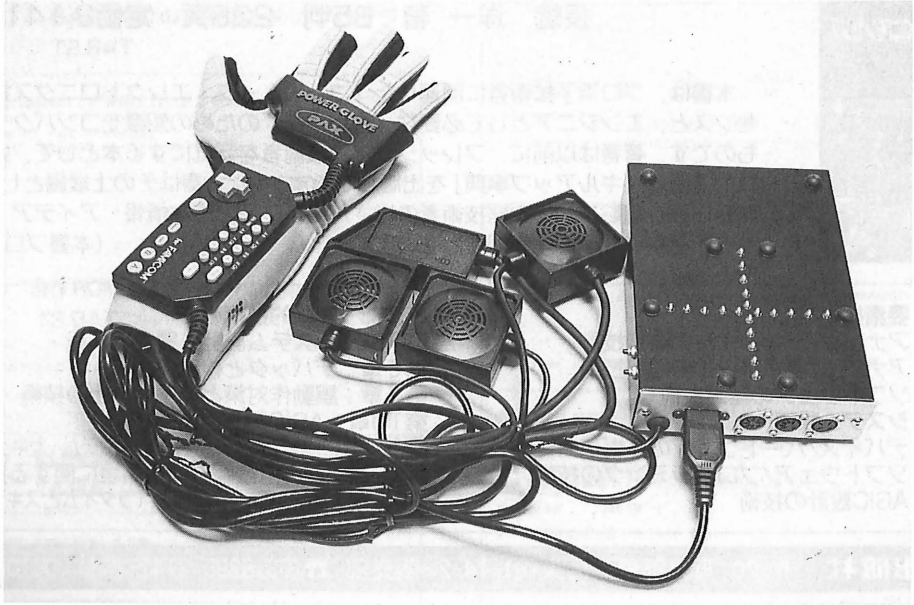
写真6-1は，ファミコン用のコントローラとしてアメリカで開発された「PowerGlove」という製品をAKI-80によって活用した例の一つ目，「MIDIコントロール・グローブ」です．

PowerGloveは，**図6-1**のように「超音波センサによる3次元移動検出」，「曲げセンサによる指の曲げ状態検出」を行いつつ，接続されたファミコンを騙して通常のコントローラの操作と思わせるような信号を生成するというもので，この優秀な処理を行うためにワンチップ・マイコンを内蔵しています(**コラムB**,p.70)．そこで，このマシンでは，PowerGloveに**ファミコン**の代わりに接続して，「ファミコンがデータを要求している」と思わせて，その情報をPowerGloveから刻々と獲得しつつ，これをMIDI信号として送信してしまおうという構想で製作しました(**図6-2**)．まさに，狐と狸の化かし合いというところです．

リスト6-1(誌面の都合により付録フロッピ・ディスクに収録)は，Nifty-ServeのMIDIフォーラムのSubSysOpでもある筆者が，MIDIBGNフォーラム(FMIDIBGN)の19番会議室「音楽情報科学の会議室」で紹介した，PowerGloveの解析情報(NIFTY.TXT)です．

図6-3(p.63)は，キタナイですがこの元となった筆

〈写真6-1〉MIDIコントロール・グローブ（PowerGlove）



ファミコン　ここでのファミコンは，初代のファミリーコンピュータのこと．アメリカではNES(Nintendo Entertainment System)と呼ばれた．

者の手描きの解析図です．ただし，これは筆者がテスタやオシロスコープ片手に独自に解析した結果なので，読者の皆さんがバッタ屋でPowerGloveを入手したとしても，この情報と同じロットであるかどうかは保証できませんので，改造する場合には自分であらためて調査してみてください．

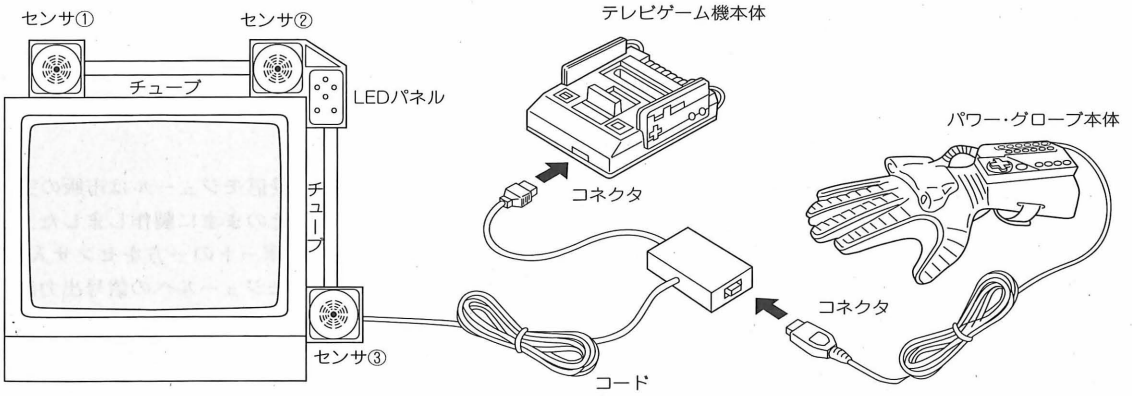
図6-4(p.63)がこのマシンの(復元された)回路図，**リスト6-2**(p.76)がそのソフト(GLOVE.SRC)です．アセンブラはBASM80なので，XA80とはわずかに形式的な宣言部分が違っています(わずかに2種類だけ，マクロを利用しています)が，まったく簡単にXA80版にもリプレースできるものです．ここでは，MIDIは送信だけなので割り込みは使わず，逆にタイミングは「正しいファミコンである」と騙すために正確なクロック(時間基準)が必要なので**タイマ割り込み**を使っています．ここでのポイントは，PowerGlove内のCPUはホストのファミコンからの信号が「ある幅の周期であるか」をチェックしているらしいことで，あまり速かったり遅かったりすると，まるで沈黙してデータを返さない状態となりました．リストにあるパラメータは，これを調整するために実験的に決定したものです．

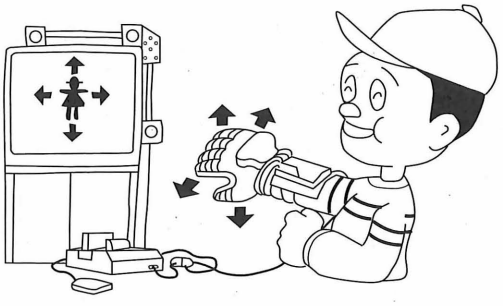
具体的な動作としては，あとあとMIDI情報は簡単に変更できるということもあって，もっともわかりやすく「上下方向で音量」，「左右方向でステレオ・パンポット定位」というMIDI情報としています．MIDIステータスでは，ボリュームは[Bn 07 **](nはチャネル，**はバリュー)，パンポットは[Bn 0A **]として定義されています．パネルには，MIDIで出力している情報をPowerGloveを操作しながらでも見えるように，中央から上下方向と左右方向にそれぞれ5個ずつのLEDを並べて，簡単のためにこの出力は4個のラッチ74HC574をI/Oとして増設し，**スタティック表示**しています．PowerGloveは，超音波センサの検出データの変動を積分(平均)して吸収しているため，ときどき超音波センサに対して，自分の真正面に手を伸ばしながら「センタリング」というボタンを押して，「ここが上下左右の中心だよ」と教えてやらないといけませんが，この状態もパネルに並んだLEDによって簡単に確認できるようにしました．

● AKI-80をPower Groveのコントローラにする

写真6-2は，この解析と改造に味をしめて行った第2弾で，今度はグローブの上面にあったコントローラに似たケースを無理やり開けてPowerGloveの心臓部であるCPU基板やコントロール・ボタンをすべて取り去ってしまい，ここにAKI-80を格納したというものです．

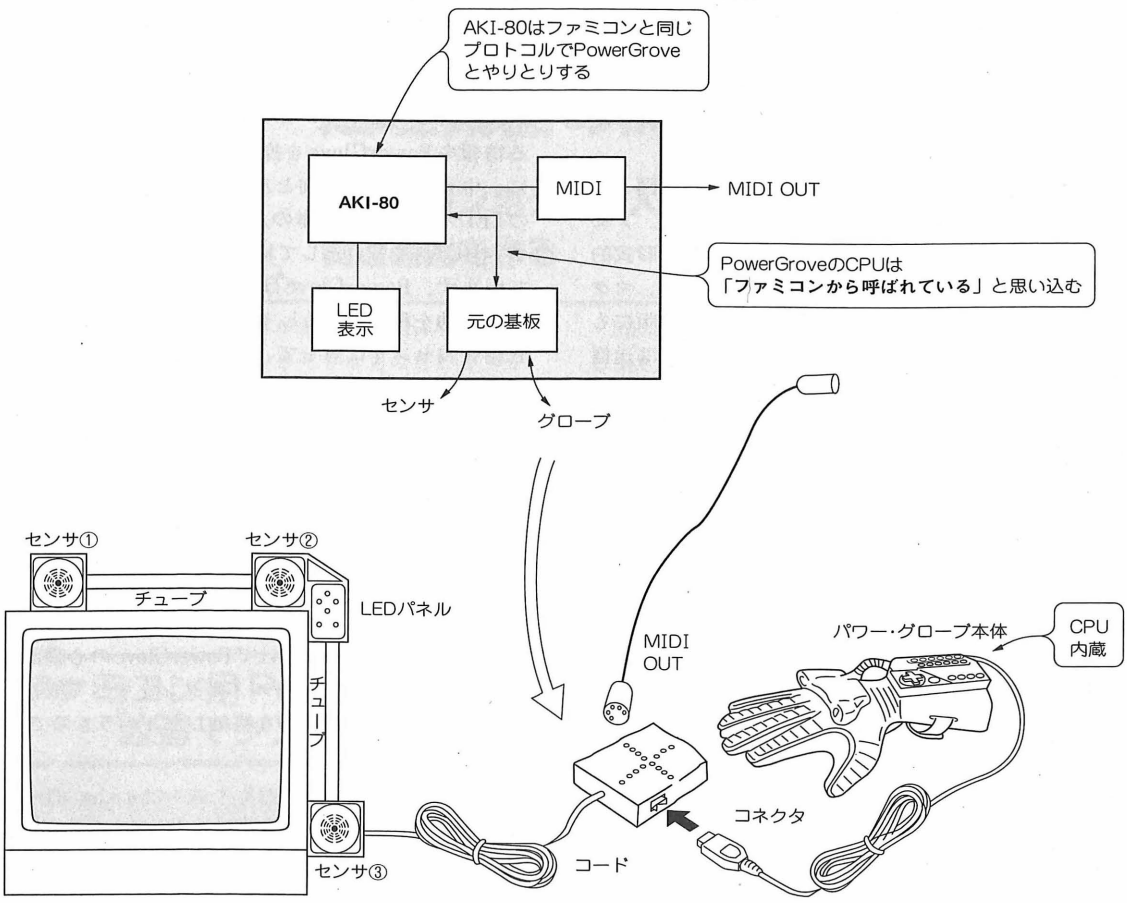
〈図6-1〉(3) PowerGloveの動作





タイマ割り込み　CPUに内蔵されている周辺LSI機能である「インターバル・タイマ」によって割り込みをかけ，定期的な処理を実行すること．

〈図6-2〉AKI-80とPowerGloveの接続



MIDIケーブルが出るのも不格好だったので，ワイヤレス送信機まで搭載してしまい，超音波センサを使用せず，指の曲げセンサ情報のみを250 MHzのワイヤレスで送信するようにしています．このセンサは筆者のComputer Music作品として，実際にコンサートのステージ上で，東京や神戸で活躍しました．このグローブ部分ではたんに2進数にデコードした「指情報」を送信しているだけですが，隣にあるのがこれをMIDI出力するためのワイヤレス受信/MIDI送信機です．

この改造はかなり本格的なものとなりました．PowerGloveとしては「指の曲げセンサ」を使うのみですから，製作のポイントは小型化の追求ということになります．ここではAKI-80の基板をさらに削り，PowerGloveのコントローラ内部もあちこち削って，ようやく格納に成功しました．曲げセンサからのアナログ抵抗値を電圧に変換してシュミット・トリガICで2値出力化して検出する部分は，超音波センサの入っていたケースの部分に空中配線で納めています．

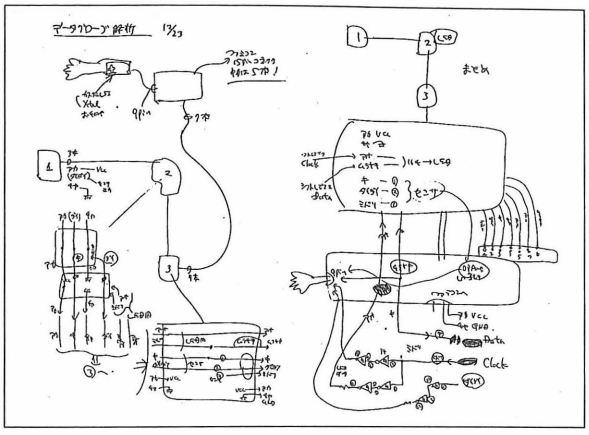
ワイヤレスの送信・受信モジュールは市販の完成品を利用して，推奨回路そのままに製作しました．AKI-80は二つのパラレル・ポートの一方をセンサ入力に，他方をLED表示と送信モジュールへの信号出力に使用しているだけで，MIDI信号処理もないので，ほとんどの時間は足踏みしているばかりとなりました(マシンのシステム概要は**図6-5**，プログラムはほとんど同等なので省略しました)．

なお，このワイヤレスPowerGloveから送られた電波を受信してMIDI情報出力するマシンから受けたMIDI情報をどのように処理するかについて少しだけ紹介しておきます．

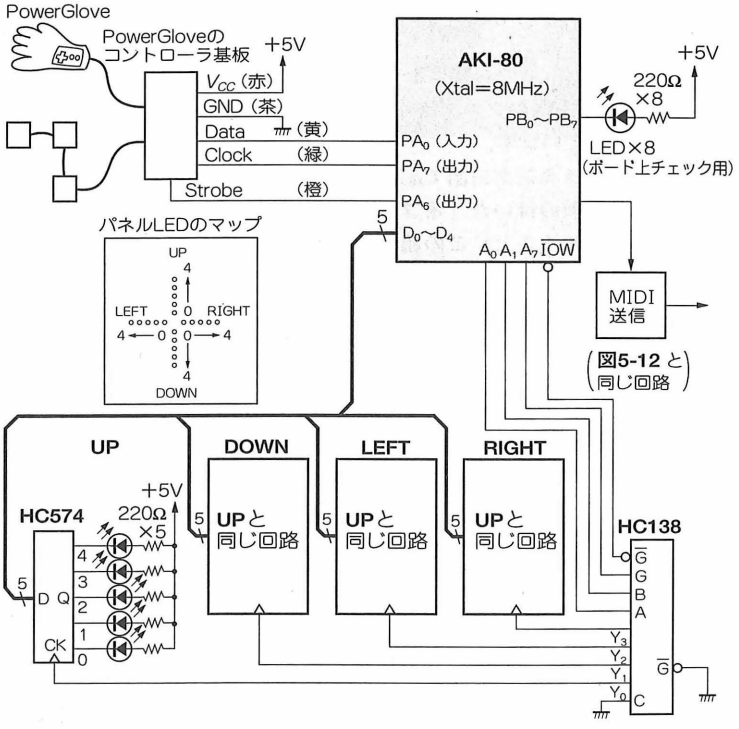
本書の対象からはずれるので詳しくは述べませんが，筆者はComputer Music環境としてMacintoshの“MAX”というソフトを活用しています．これは**図6-6**のように，見たところ各種の「箱」と「線」とからできている**GUI環境**で，必要な処理の「箱」を線でつなぐと，それだけでMIDI処理ソフトウェアが実現できてしまうというスグレモノです．筆者は一つの

バリュー　MIDIメッセージのうち，具体的な変数として表現されるデータの部分を「バリュー」と呼ぶ．

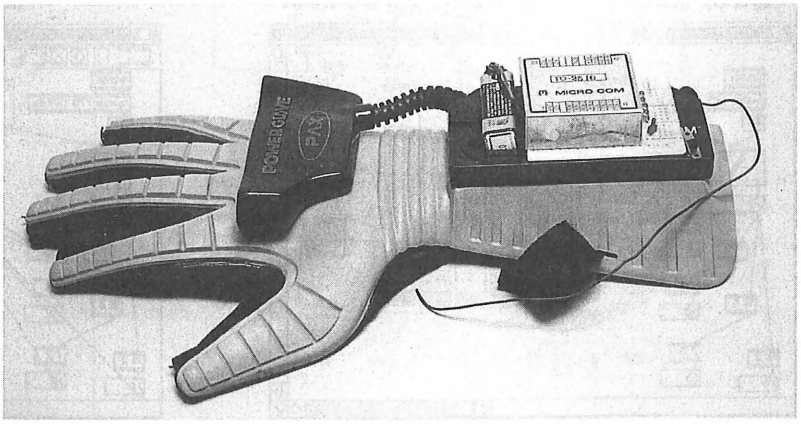
〈図6-3〉 手書き解析図



〈図6-4〉
PowerGloveのインターフェース回路

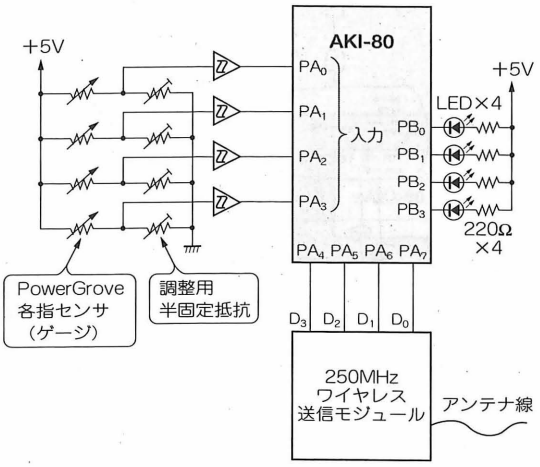


〈写真6-2〉
第2弾のPowerGloveのコントローラにAKI-80を載せたもの



スタティック表示　LEDの表示方法には，各表示セグメントのLEDをずっと点灯しているスタティック表示と，次々に大電流で点滅させ，人間にはその残像として点灯し続けているように見せるダイナミック表示とがある．

〈図6-5〉ワイヤレスPowerGloveシステム概念図



Computer Music作品で同時に3～4台のMacを使い，それぞれにこのMAXのプログラム(「パッチ」と呼ぶ)を作って走らせています．

従来はプログラミング言語で記述していたアルゴリズムが，箱に名前の付いた「**オブジェクト**」を多数並べて機能を分割すること，この箱と箱を線で結ぶことで機能を結合することという単純な形式で処理できるようになっているわけです．

図6-6の中央の5本のスライダは「グローブの指の状態」を表示するためのディスプレイ・オブジェクトですが，このスライダをマウスで上下させれば逆にその値を出力することもできます．そして**図6-7**は，このパッチを「実行モード」から「編集モード」に切り替えたところです．画面上端に各種のオブジェクト候補が並んでおり，ここから必要な部品としていくつでも呼び出して自由に配置することができます．

画面の左上端にある「midiin」というオブジェクトが，Macに入力されたすべてのMIDI情報を供給し，ここではパワー・グローブの情報として定義された特定の情報にヒットしたときにだけ，それ以降の処理がトリガされる，もっとも簡単なパターン認識となっています．情報は4本の指それぞれにON/OFFを対応させた4ビット2進数データとして与えられるので，図のような数学的処理によって「どの指が曲がっている」という情報を抽出し，このパッチではそれぞれの指に割り当てられたノートのMIDIノート・オン情報として，最終的には「noteout」オブジェクトからMIDI出力されています．

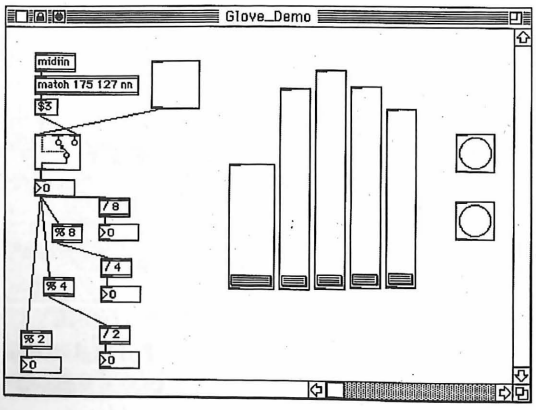
画面右側の二つのボタンは，それぞれの指に割り当てるノートのセットを切り替える**トリガ**となっています．つまり，上のボタンを押してグローブを握るとCmaj7の和音の**アルペジオ演奏**が，下のボタンにするとC#m7の和音のアルペジオになります．このような仕組みは何十種類でも簡単に編集できるため，瞬時にしてチューニングやスケールの変更ができる新しい楽器というものがMAXでは簡単に実現できてしまいます．

フランスに渡った「アナログ-MIDIコンバータ」

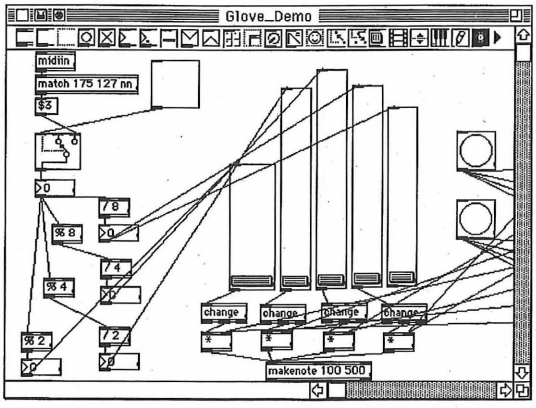
つぎの製作例は，本書に紹介 している実例としてはほぼ最新のもので，なんと地球の裏側，遠くフランスに渡って活躍を始めています．

フランスの国立音楽音響研究所IRCAM(Institute for Research and Coordination of Acoustics and Music)は，アメリカのCCRMA，CNMAT，MIT，CMUなどとともに世界のComputer Music研究をリードしているところですが，最近ではここでも「人間とComputerとが音楽を共演する」というアプローチが盛んです．ところでIRCAMにはソフトウェアの専門家は多いものの，AKI-80のようなプリミティブなハードウェア部分はあまり強くないらしく，まったく見ず知らずの

〈図6-6〉コンピュータ・ミュージック・ソフトMAX

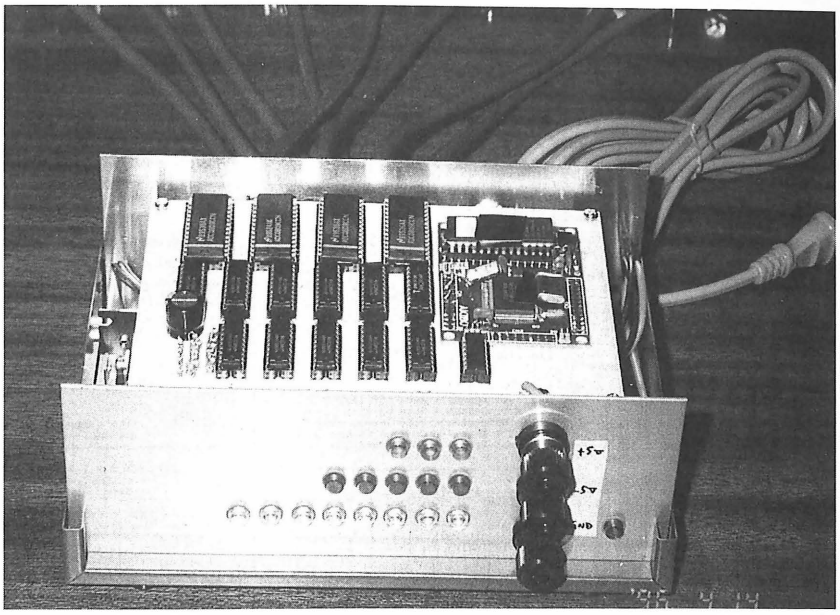


〈図6-7〉編集モードの画面

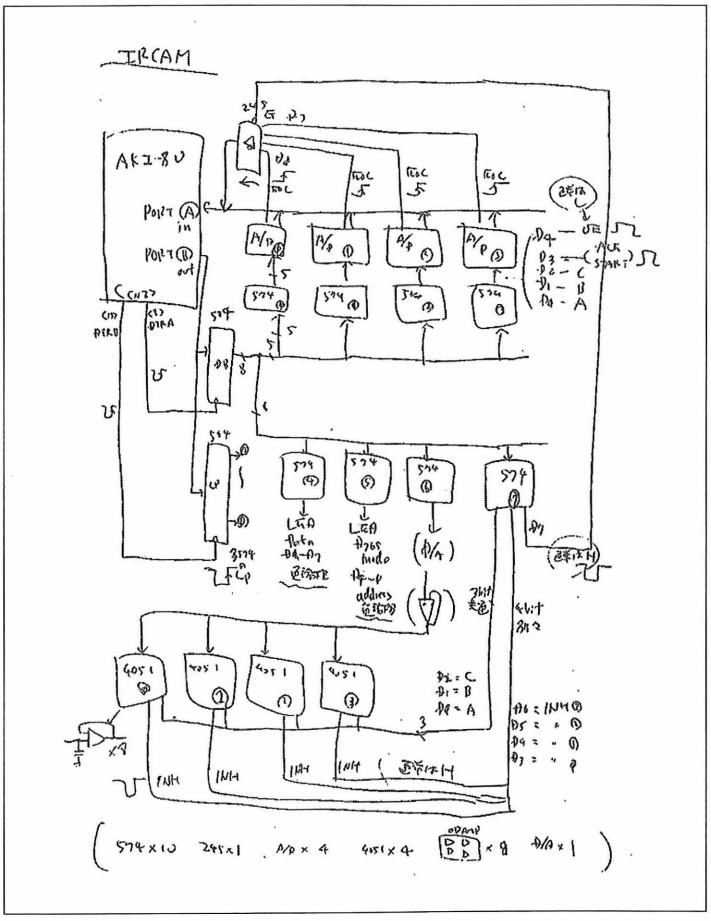


GUI環境　テキストだけが並ぶいわゆるプログラミング言語による開発でなく，GUI(Graphical User Interface)によって視覚的に分かりやすい開発環境のこと．

〈写真6-3〉
アナログ←→MIDI変換器
(在フランス)



〈図6-8〉
アナログ←→MIDI変換器の回路図
(手書きのもの)

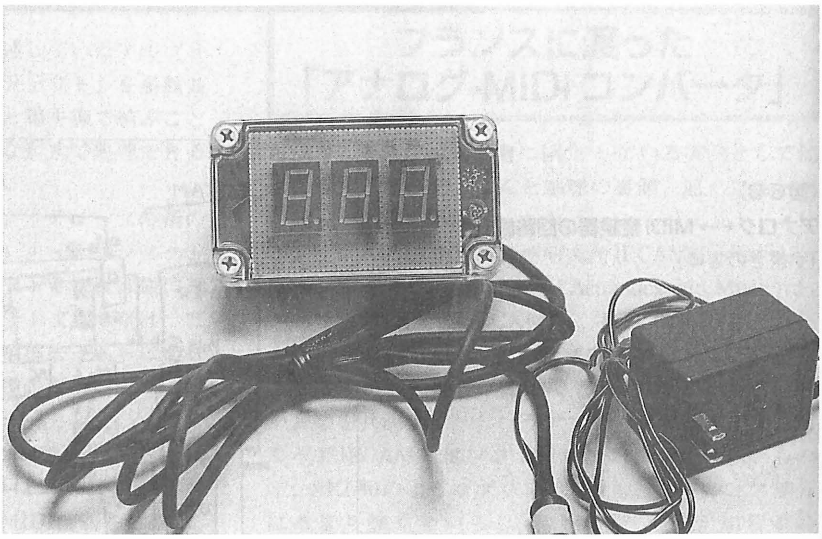


オブジェクト　ある処理を行う「もの」をオブジェクトと呼ぶ．オブジェクト同士を線で結ぶと，それらの間の情報交換が簡単に実現される．

〈リスト6-3〉アナログ←→MIDI変換器の仕様

```
  Analog <--> MIDI Converter for IRCAM : April 1996
  produced by Y.Nagashima (Art & Science Laboratory)

Analog input = 32 channel (Unipolar 0.0V-5.0V)
A/D conversion speed = about 12 msec (for each channel)
A/D output resolution = 7 bits ( data : 0[0.0V] - 127[5.0V] )

Each A/D input ON/OFF can be selected to transmit by MIDI control.

"Control Message" for input selection is fixed.
        ---> [Polyphonic Pressure, Channel=16] is used.

               [ AF 00 00 ] --- A/D (1) Off
               [ AF 00 01 ] --- A/D (1) ON, MIDI transmit
               [ AF 01 00 ] --- A/D (2) Off
               [ AF 01 01 ] --- A/D (2) ON, MIDI transmit

               [ AF 1F 00 ] --- A/D (32) Off
               [ AF 1F 01 ] --- A/D (32) ON, MIDI transmit
     Power ON default = all A/D Off.
Output A/D message is all merged to 1 MIDI output.

Output MIDI Channel is changable by MIDI.
               [ AF 20 00 ] --- Output MIDI Channel = 1
               [ AF 20 01 ] --- Output MIDI Channel = 2

               [ AF 20 0F ] --- Output MIDI Channel = 16
     Power ON default = Channel 1.

Output MIDI Status is changable by MIDI.
               [ AF 21 08 ] --- Output MIDI Status = Note Off [8n]
               [ AF 21 09 ] --- Output MIDI Status = Note On [9n]
               [ AF 21 0A ] --- Output MIDI Status = Poly Press. [An]
               [ AF 21 0B ] --- Output MIDI Status = Control Change [Bn]
               [ AF 21 0E ] --- Output MIDI Status = Pitch Bend [En]
     Power ON default = Poly Pressure [An].  n = channel

Output Data Format:  [St+Ch][No.][Data]  3 bytes
                     [St+Ch] is Status Byte : default [A0]
                     [No.]   is Input Number : [00]-[1F] for 1-32
                     [Data]  is A/D Data : [00]-[7F] for 0-127
Output Data Format:  [St+Ch][No.][Data]  3 bytes
                     [St+Ch] is Status Byte : default [A0]
                     [No.]   is Input Number : [00]-[1F] for 1-32
                     [Data]  is A/D Data : [00]-[7F] for 0-127

Selected A/D data is transmitted if the data changes bigger
than [Threshold] value.
[Threshold] value is changable by MIDI.
               [ AF 22 nn ] --- A/D data is transmitted if the difference
                                is bigger than [nn].
      Power ON default : [nn] = 4. ( data change by 5 steps )

Analog output = 32 channel
Analog output voltage range : 0.0V - about 4.0V
D/A conversion speed = about 1 msec (for each channel)
D/A output resolution = 7 bits ( data : 0[about 0.1V] - 100[about 4.0V] )

D/A analog Output Data Format:
               [ AF 30 nn ] --- Analog Output 1
               [ AF 31 nn ] --- Analog Output 2

               [ AF 4F nn ] --- Analog Output 32
      Power ON default : all D/A output OFF (data is not fixed)

Panel LED Display Message          7 6 5 4 3 2 1 0
                             mode        * * *
   (on=1, off=0)          channel          * * * * *
                          data       * * * * * * * *

      (1) mode 1 < 0 0 1 > A/D convert ON/OFF settings
          channel = 1 <00000> - 32 <11111>
          data    = OFF <00000000> / ON <00000001>

      (2) mode 2 < 0 1 0 > system setting
          channel = <00001> : output MIDI Status and Channel
          data    = <sssscccc>
                    ssss = 8/9/A/B/E
                    cccc = 0(1ch) - F(16ch)
          (this message is displayed through <00001>--<01000>)
          channel = <10000> : A/D threshold data
          data    = <0nnnnnnn> (7bit)

      (3) mode 3 < 1 0 0 > D/A convert output(setting) data
          channel = 1 <00000> - 32 <11111>
          data    = <0nnnnnnn> (7bit)

( These messages are displayed with MIDI control or A/D events,
  and auto-displayed (1)->(2)->(3)->(1)... when nothing happens. )

      (4) mode 4 < 1 1 1 > A/D converter event MIDI transmit
          channel = 1 <00000> - 32 <11111>
          data    = <0nnnnnnn> (7bit)
```

〈写真6-4〉
3桁の大型LED表示器



IRCAMの作曲家/研究者から，筆者に特製マシンについての問い合わせの電子メールが舞い込みました．

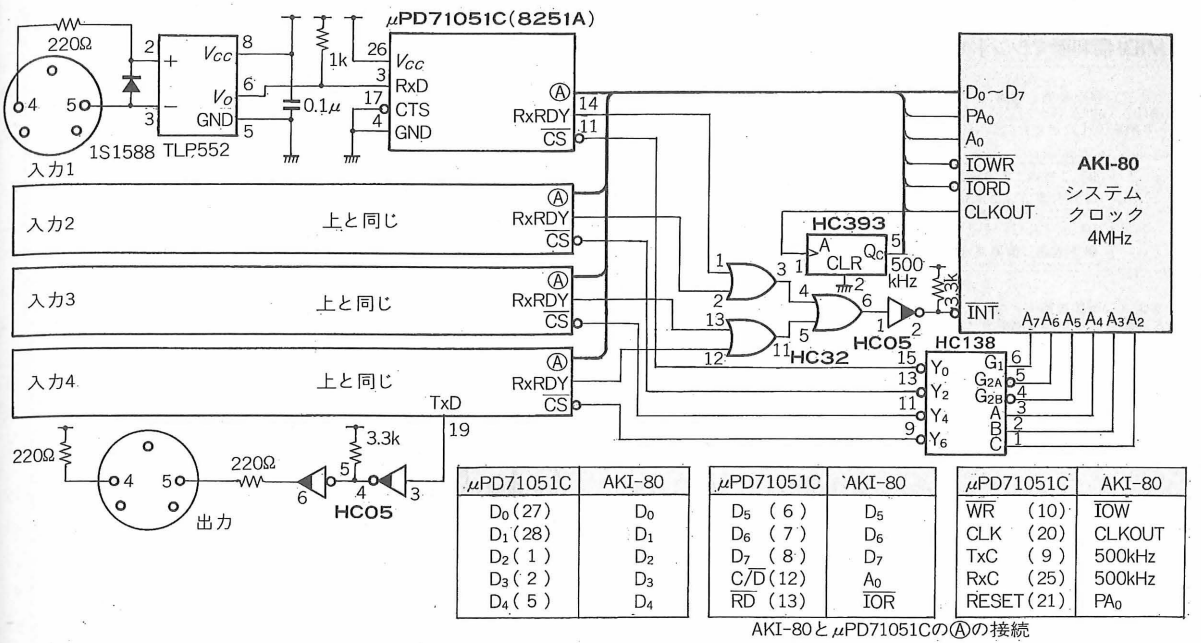
そしてほぼ毎日，インターネットの電子メールでのやりとりの末に，人間の動作を検出するセンサ部分は専用のものを利用するとして，汎用の**「アナログ←→MIDI」**の変換を担当する特製マシンの開発を依頼されるという正式な話になったのです．仕様については，最初は「8系列のアナログ電圧入力をMIDIとして送る」というだけだったのですが，筆者がうっかり「32チャネルぐらいまでは簡単ですよ」，「MIDIからアナログ制御電圧を出力するのも問題ありませんよ」などと書いてしまったために，最終的な仕様は**リスト6-3**のように肥大化しました．フロント・パネルには外部OPアンプ回路用の両極電源端子まで用意しましたから，たいていのアナログ・インターフェースとして機能できます．

写真6-3がその外観，**図6-8**がそのブロック図です(これは実際に筆者が製作の際に書いたもの，そのままです．あとは個別のICのデータシートを見るだけで，この回路図以外に何もありません)．また，**リスト6-4**(p.78）がAKI-80のプログラム(IRCAM.SRC，8ビット単位で4ブロックというワンパターン処理が多いので，こちらはマクロをいろいろと活用してみた好例)です．

リストはほとんどコメントが入っていない無愛想なものですが，ここでは筆者の製作した実例そのものを紹介する意味で補足していません．製作はほぼ3日，アナログ入出力が合計で64本となり，ケーブル回りの配線がひたすらたいへんでしたが，なかなかコンパク

トリガ　「ある処理を行うためのきっかけ」の司令のこと．具体的な数値などを渡さず，「行け！」という情報のみが送られる．

〈図6-9〉MIDIマージャの回路



μPD71051C	AKI-80
D_0(27)	D_0
D_1(28)	D_1
D_2(1)	D_2
D_3(2)	D_3
D_4(5)	D_4

μPD71051C	AKI-80
D_5(6)	D_5
D_6(7)	D_6
D_7(8)	D_7
C/$\overline{D}$(12)	A_0
$\overline{RD}$(13)	$\overline{IOR}$

μPD71051C	AKI-80
$\overline{WR}$(10)	$\overline{IOW}$
CLK(20)	CLKOUT
TxC(9)	500kHz
RxC(25)	500kHz
RESET(21)	PA_0

AKI-80とμPD71051CのⒶの接続

トにまとまりました．いざ国際宅配便で送ろうとすると，なんとフランスは個人宛の電気製品の送付(輸出)は一切禁止ということで困ったのですが，「国立機関からの依頼で送った研究用途の試作品である」というINVOICEが効いたのか，無事に届いてくれました．現在でも，ここにつなぐセンサ回りとかOPアンプ回路などの質問の電子メールに，つたない英語で回答する日々が続いています．

MIDIメッセージ・ディスプレイとMIDIマージャ

次の例は，研究のための道具，あるいは筆者自身が作曲家として自分の作品を発表するコンサートなどの場で活躍している「MIDI小物」の中から二つ紹介します．

最初に背景となる状況を簡単に説明します．Computer Musicシステムと人間の演奏家が共演する場合には，「いまシステムがどのような状態にある」という情報を演奏家に伝えることが非常に重要です．現在の小節番号とか，システムの待機状態などを知らずに演奏するのは，演奏家にとって限りないストレスなのです．

そこで，「3桁の大型LEDにMIDIから自由な数字を与えて表示する」という，**写真6-4**のような単純な装置を作りました．ここでは回路図もリストも省略しますが，中身はすでにここまでに述べたものとまったく同等です．

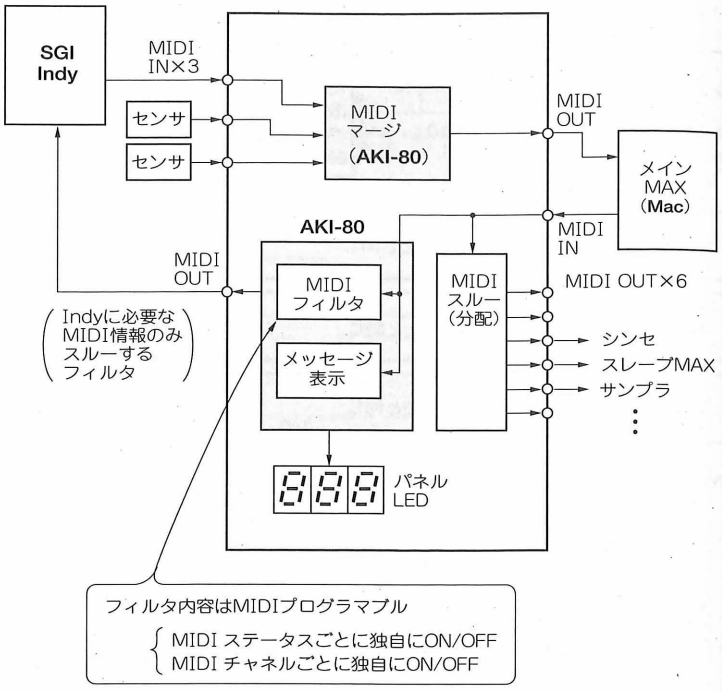
MIDIデータの中には，本来のMIDI規格とは別に**勝手な情報プロトコル**を定義して，送信側のMAXでこれを送ります．作曲するのもシステムを製作するのも筆者自身ですから，このような特殊な使い方も可能です(筆者にとって，システムの製作や専用ソフトの制作というのは，すでに作曲の一部として重要です)．

ここでは，MAXからのMIDI情報はこのメッセージ・ディスプレイだけでなく，同時にいろいろなシンセサイザなどにも分配されていますから，メッセージ表示の情報でヘンな音が出ては困りますが，MIDIには「まずほぼ絶対に使われていない」という情報も少なくありません．たとえば筆者の場合，「**ポリフォニック・プレッシャ**」というほとんどすべての電子楽器がサポートしていない情報の，さらに第15，16チャネルという後ろのあたりを利用して，各種のオリジナルMIDI機器の情報を送っています．対応していないMIDI情報については無視するルールとなっていますから，これらのMIDI情報が音源などに同時に供給されても問題ないのです．

メッセージは3桁ですが，MIDIのデータ・バイトは7ビットで0～127までしか表現できませんから，ここは冗長であっても，「桁番号＋数字」という情報を三つの桁ごとに定義しています．たとえば「256」と表示する場合には，「100の桁に[2]」「10の桁に[5]」「1の桁に[6]」という3メッセージを送らなければなりませんが，実際の音楽演奏ではこの情報は少なくとも1秒以上の間隔でしか変化しませんから，まったく問題ありません．このマシンはとても便利なため，コンサ

アルペジオ演奏　和音を構成している複数の音階が，バラバラと分散的に演奏されること．ギターで言えばポロロン，と順に弾いたようなもの．

〈図6-10〉
「MIDI便利屋マシン」のシステム図



ートのたびに新しく作り(たとえば演奏者用だけでなく指揮者用にもという具合)，作曲家の**志村哲**氏のComputer Music作品をデンマークや香港のコンサートで演奏する際にも活躍してくれました．

もう一つのMIDI小物として，これも要請があるたびに製作しているのが，「MIDIマージャ」です．MIDIマージ機能についてはすでに前章で述べたとおりですが，本章で紹介しているような多数のMIDIセンサからの情報を処理するには，多量のMIDI情報がすべて1本にまとまる必要があり，**MIDIマージャ**の出番となるわけです．

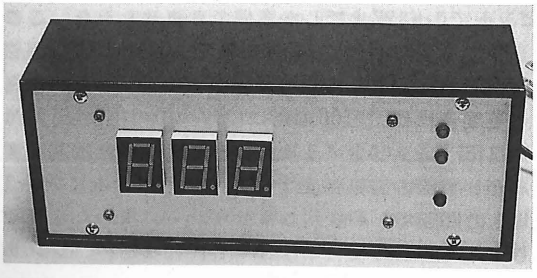
図6-9はこのようなMIDIマージャとして製作したものの回路図，**リスト6-5**(p.81)はそのAKI-80プログラム(MERGER.SRC)です．ここではMIDI入力が4系列もあるので，AKI-80の内蔵SIOでは不足しますから，μ**PD71051**を外部に4個並べて，完全に同じルーチンを使うようにしています．ポイントとしては，4個のμPD71051の割り込みをすべて論理和してAKI-80の割り込みに入力し，割り込みルーチンの先頭で「1番目のμPD71051の割り込みか」「2番目か」などと割り込み元をポーリングして調べているというところでしょう．

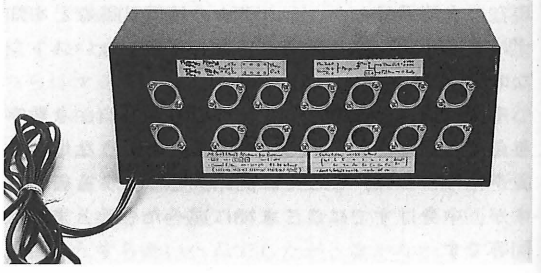
本当はこれはあまりスピードを稼げない，美しくない方法なのですが，割り込みラインをそのままジャンプのためのテーブルに利用するという必殺テクニックまで使わなくても，実用的にはまったく問題なく動作しています．

〈リスト6-6〉MIDIフィルタとしての機能

```
;  [AE] [nn] [para] : Polyphonic Key Pressure
;
;     nn =  5 : 100 * n       LED Display
;           6 :  10 * n       LED Display
;           7 :   1 * n       LED Display
;           8 :  (off)        LED OFF
;          21 : Channel Filter bitmap : CH1-CH4 [1=on]
;          22 : Channel Filter bitmap : CH5-CH8 [1=on]
;          23 : Channel Filter bitmap : CH9-CH12 [1=on]
;          24 : Channel Filter bitmap : CH13-CH16 [1=on]
;                                       (Default : CH16 only ON)
;          25 : Status Filter bitmap : [1=on]
;                    bit6   5   4   3   2   1   0
;                       8n  9n  An  Bn  Cn  Dn  En
;                                       (Default : <Bn> ON)
;          26 : Setting Default Paratemers (only <BF> ON)
```

〈写真6-5〉MIDIマージ付き周辺インターフェース





アナログ←→MIDI変換　アナログ電圧を受けてMIDI情報に変換して送信するという処理と，MIDI情報を受けてこれを変換したアナログ電圧を出力する処理．

〈図6-11〉[1] ヤマハの新楽器「MIBURI」

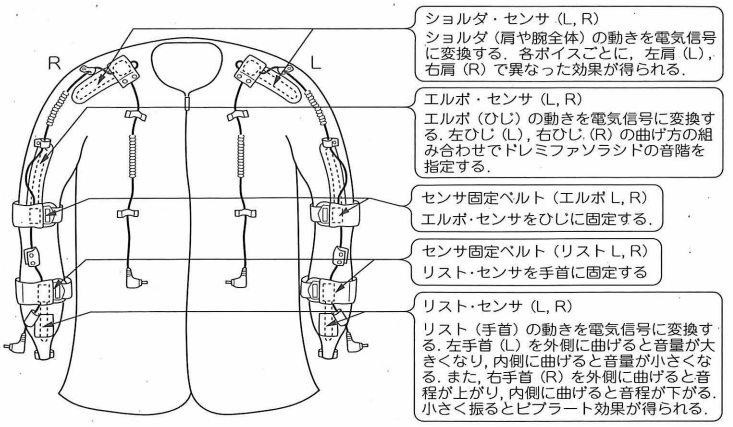


写真6-5は筆者のComputer MusicシステムになくてはならないMIDI小物ですが，ここでも同様のMIDIマージ機能を利用しています．このシステムは**図6-10**のような構成の，いわば「MIDI便利屋」マシンで，自室の**SGI Indy**ワークステーションと3台のMacintoshを有効に結びつけています(Indyというマシンは標準でMIDIをサポートしていて，MIDIインターフェースの部分はMac互換となっています．筆者の場合，このインターフェースも市販のものを買うのが面白くないので，「4台のMac用MIDIインターフェース」として自作しています)．

そして，MIDIサポートといいながら，実はIndyのMIDI機能はそれほど強力ではないため，多量のMIDI情報を受信しているとハングアップしてしまいます．そこで「Indyのプログラム(これも当然，自作)で必要とする情報だけを通す」という，いわば「MIDIフィルタ」が必須となります．

このマシンの重要な機能の一つはこのMIDIフィルタで，あとはすでに述べた，

- 3入力MIDIマージャ(IndyからのMIDIと二つのセンサ情報をマージ)
- LEDによる3桁のメッセージ・ディスプレイ
- 6出力のMIDIスルー・ボックス(多数のシンセやMacに情報を分配)

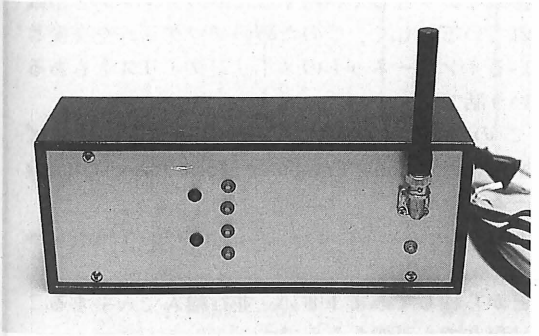
という機能も同時に同じケースに搭載しています．

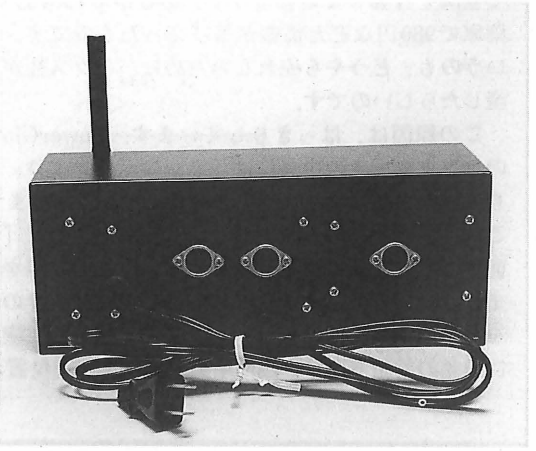
MIDIマージ部分に1枚のAKI-80，そしてさらに別のAKI-80によって，「特別のMIDI制御プロトコルによって，MIDIフィルタとしての機能そのものを外部からプログラマブルにする」という仕様になっています．フィルタとしての機能をROMに固定していないのです．**リスト6-6**は，この機能仕様の部分をプログラム冒頭で定義したもので，筆者の新しいシステム開発は，このような「MIDI独自プロトコルの定義」の記述からスタートします．

新楽器「MIBURI-Sensor」と「SNAKEMAN」

Computer Musicの世界で最近の新しい動きとしては，1995年後半からようやく市販の電子楽器(各社)で

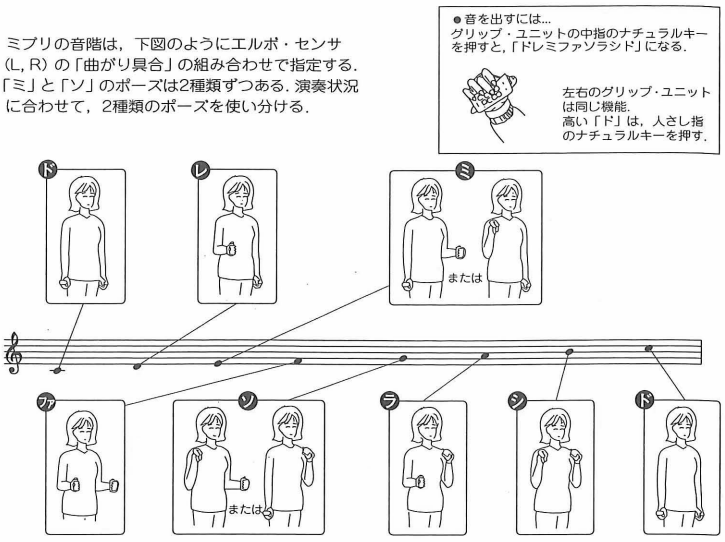
〈写真6-6〉ワイヤレスPowerGrove受信→MIDI送信マシン





INVOICE　海外に機材等を送る際に，税関を通過するために作成する「送り状」．

〈図6-12〉(1) MIBURIの音階のポーズ



登場してきた「**物理モデル音源**」と，これも1995年にヤマハから発表された新楽器「MIBURI」があります．

MIBURIとは**図6-11**のように，6個の曲げセンサを取り付けたウェアを着てボディ・パフォーマンスで「演奏」するという電子楽器で，標準の内蔵音源には物理モデル・シンセを使っています．普通の音階を演奏する場合には，**図6-12**のように両腕の曲げの組み合わせで音階を割り当てていくのですが，筆者の印象としてはどうやっても不自然な「アテ振り」(音楽に合わせて振り付けを強制されている)ようなことになり

●コラムB● パワー・グローブについて

PowerGloveとは，初代ファミコンのコントローラとして使えるという非常に安価なセンサです．アメリカのマテル社による開発で，日本ではパックス・コーポレーションという会社が販売代理店を行っていたようです．

ところがこの製品，実際にはまったく売れず，筆者は定価15,000円というものを，街のバッタ屋で598円で買いました．

このPowerGloveはバッタ屋さんの広告商品として全国を行脚しているようで，ある県で798円，秋葉原で980円などと情報が飛び交ったものです．というのも，どうやら売れないためにパックス社が倒産したらしいのです．

この原因は，はっきりしています．PowerGloveのコントローラの中央には，ほかのボタンより一回り大きな「センタリング」というボタンがあります．これは，上下左右の超音波センサからの情報で「画面中央」を新たに指定しなおすというためのボタンです．つまり，ノイズなどに対応して，センサのデータは刻々と平均（積分）されていますから，次第に誤差の蓄積で自分が画面中央と思っていた位置がずれてきます．

そこで，たびたび画面中央にグローブをはめた腕を突き出しては，このボタンを押す必要があるのです．ゲームをしている最中に，たとえば30秒に一度，この「センタリング」をしていては興ざめします．

このPowerGloveの入手方法ですが，そういうわけで確実に在庫している会社は日本にはありませんが，まだまだアメリカでは商品として存続しているようです．欧米では，PowerGloveをMacに接続して拡張センサとして利用するためのアダプタも市販されているらしく，このためのプログラムを交換しているインターネットのメイリング・リストもあるという話です．

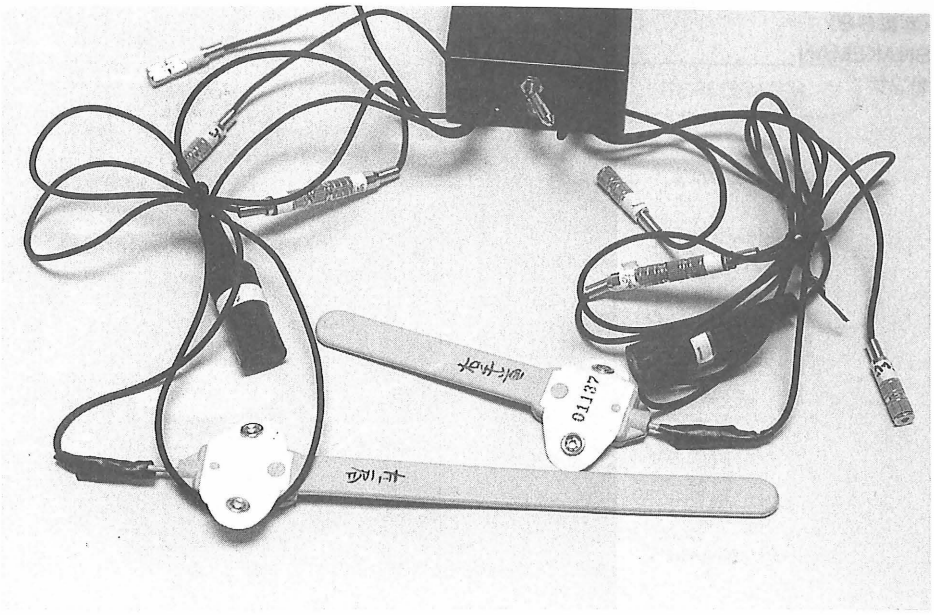
このあたりの詳しい情報は，たとえばICMA(International Computer Music Association)のホームページ：

`http://www.coos.dartmouth.edu/~rsn/icma/icma.html`

などから探してみて下さい．並行輸入で入手することは現在でも可能だそうです．

勝手な情報プロトコル　MIDI規格と同じ信号形態とデータ形式を利用しながら，情報としてのMIDIメッセージの規格はまったく勝手に規定して使用すること．送り手も受け手もオリジナルの機器であれば，これでも問題はない．

〈写真6-7〉
MIBURI-Sensor

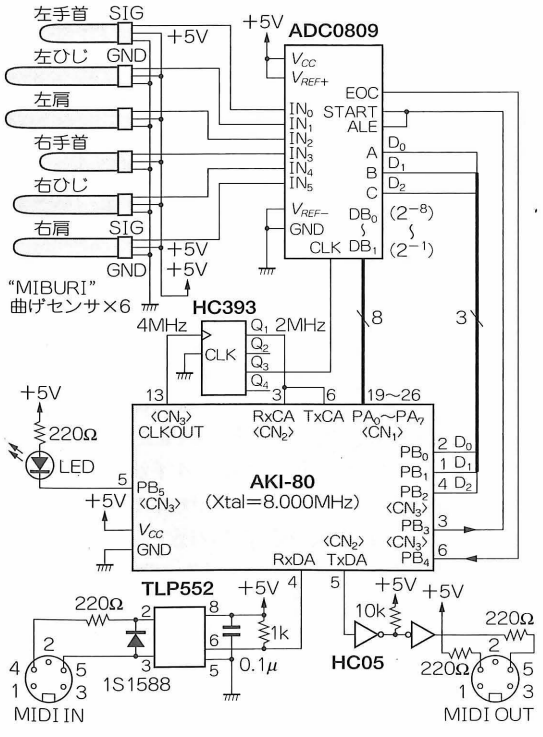


ます．

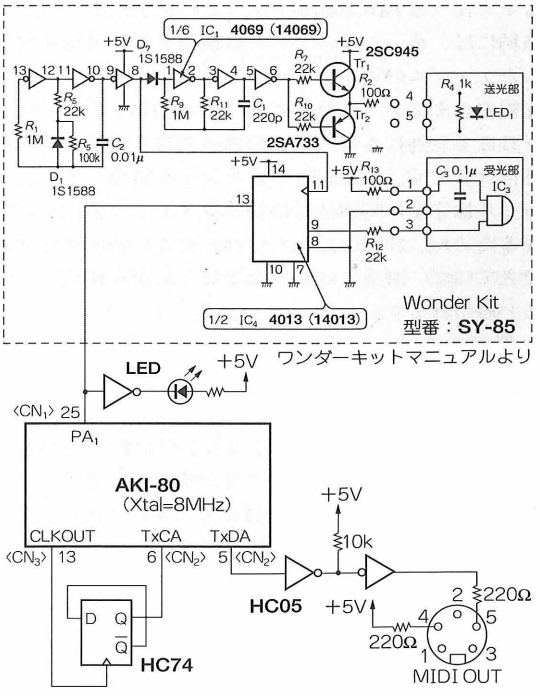
そこで筆者は，**写真6-7**のような「MIBURI-Sensor」と名付けた小型のマシンを製作しました．ポケット・サイズのケースには電池まで内蔵して，さらにMIDI入力で外部のMIDIキーボードなどから入力された情報と6個のMIBURI曲げセンサからの情報とをマージしてMIDI出力しています．

MIBURIのセンサは，中がたんに**ストレイン・ゲージ**による受動的な曲げセンサとなっているのではなく，ちゃんと電源を供給し，センサの部位ごとに値を補正するOPアンプ回路が内蔵されているようです．このセンサ部分，あるいはセンサ出力を内蔵音源にだ

〈図6-13〉MIBURI-Sensorの回路



〈図6-14〉SNAKEMANセンサとAKI-80



ポリフォニック・プレッシャ　個々の鍵盤が演奏される「ベロシティ」（弾く強さ）でなく，それぞれの鍵盤がONとなったあとでさらに「押し込まれる」という圧力を規定した情報．この部分に対応した電子楽器はほとんど存在しないので，送ってもたいていのMIDI機器で無視されて問題が起きにくい．

〈写真6-8〉
SNAKEMAN
センサ

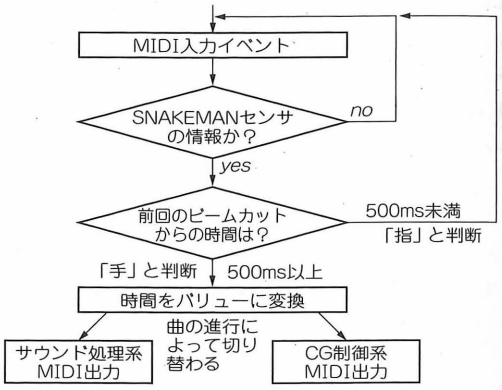
け供給するのでなく，そのままMIDIとして外部に出力するような次のステップのMIBURIが発表されれば，多くの音楽家が飛びついてヒット商品となるような期待があります．

図6-13はMIBURI-Sensorの回路図，**リスト6-7**(p.82)はそのAKI-80プログラム(MIBURI.SRC)です．アナログ入力には，ちょっと大きいのですが，もっともポピュラな**ADC0809**を使っています．ケースへの格納には，まったくネジなどは使用せず，基板を配線したケーブルのジャングルをクッションとして，空中配線のままでケースのフタをパチンと閉めています．

ここで意外にも苦労したのは，実は「信号線のはんだ付け」です．MIBURIの各センサを結合しているケーブルは完全に特別仕様の特殊ワイヤで，強度を確保するためか，非常にはんだ付けしにくい特殊な線材でできていて，製造には特殊な器具と溶剤を使用していると思われます．

他のセンサ・システムでも同様ですが，このMIBURI-Sensorでも，最初のバージョンで「センサ・データに変化があったらMIDI出力」とすると，非常に多量のMIDI情報が送信されて，受信側ではパニックに近い状態となりました．センサ・データに乗ったノイズや変動成分や誤差に対して，A-Dコンバータが律儀に対応するために起きる現象で，このため筆者は「前回値と現在値とを比較して，ある閾値を越えたらイベント発生としてMIDI送信する」というパラメータを実験的に設定しました．この手法はあらゆるセンシング・システムで必須のものですが，実質的に分解能を低下させることになりますので，十分に検討する

〈図6-15〉一定時間のマスキングをかけるアルゴリズム



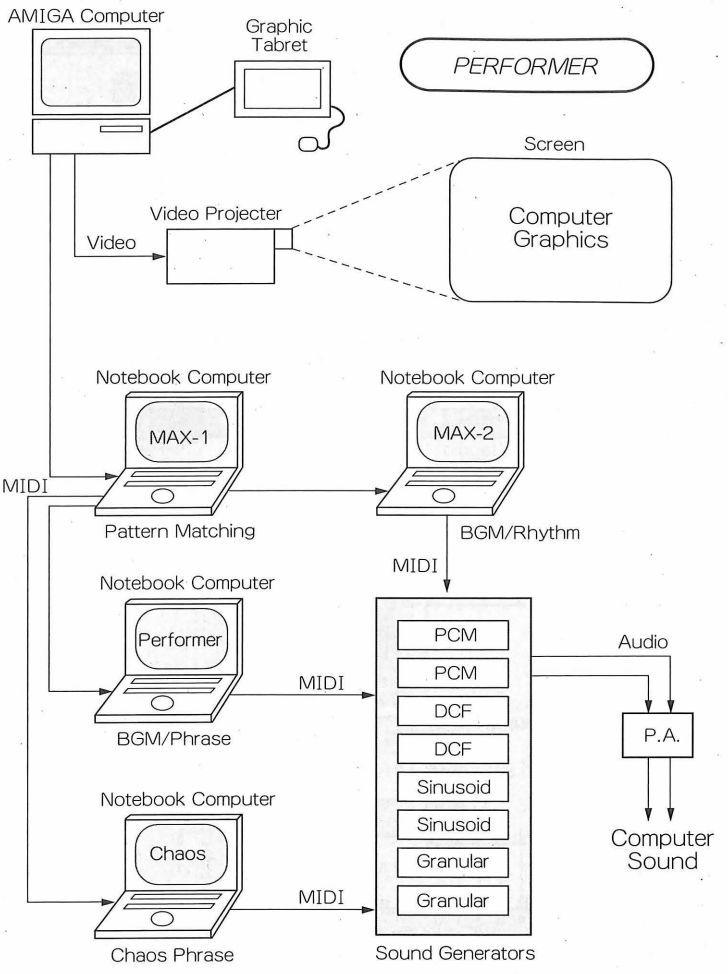
ことが必要です．

もう一つの「新楽器」は，もっとアヤシゲなものです．筆者は1996年7月13日に神戸で発表した新作“Asian Edge”に使用するセンサ(MIBURIを着たPerformerが手に持つ楽器)の材料を探すために，難波の街を歩きました．そしてインド民芸品のお店で，おもちゃの1本弦の楽器をまず仕入れました．この弦の振動をピックアップで拾ってという手法は以前にも作ったことがありましたが，何か足りません．

そして次に，日本橋のジャンク屋で，放送室の調整卓から伸びているような，フレキシブル・パイプのマイクを発見して，これを2本仕入れました．そしてあれこれいじっているうちに，**写真6-8**のような不思議なスタイルを思いつきました．弦楽器のネック部分の棒を太鼓のようなボディから抜き去ってしまい，この

志村　哲　大阪芸術大学でComputer Musicを教える一方で，電子音楽や現代音楽の作曲，そして虚無僧として尺八演奏活動を行う．情報処理学会音楽情報科学研究会の幹事でもある．

〈図6-16〉
Muromachiのシステム図



穴の部分にマイクのフレキシブル・パイプを両方に取り付け，ぐいっと曲げるとパイプの先端がボディの上で向き合います．「設計」のない，まったく場当たり的な「製作」の作業です．

パイプの先端のマイクを取り去ると，当然ですが根本から先端まで電線が貫通しています．そこで電子工作キット(ワンダーキット)の「**通過センサ**」を仕入れて，両方のパイプの先端の間にこの光ビームが走るようにと考えました．「目に見えない赤外線の弦を弾く」という怪しげな楽器のアイデアが，これで決定しました．

赤外線ビームを遮るという動作をセンシングしてMIDI送信するというマシンはこれまでにもいくつか製作(筆者の現代音楽の師匠である，作曲家・中村滋延氏のための専用マシン)していたので，あとはほぼ一本道となりました．構想10日，製作3日というところです．

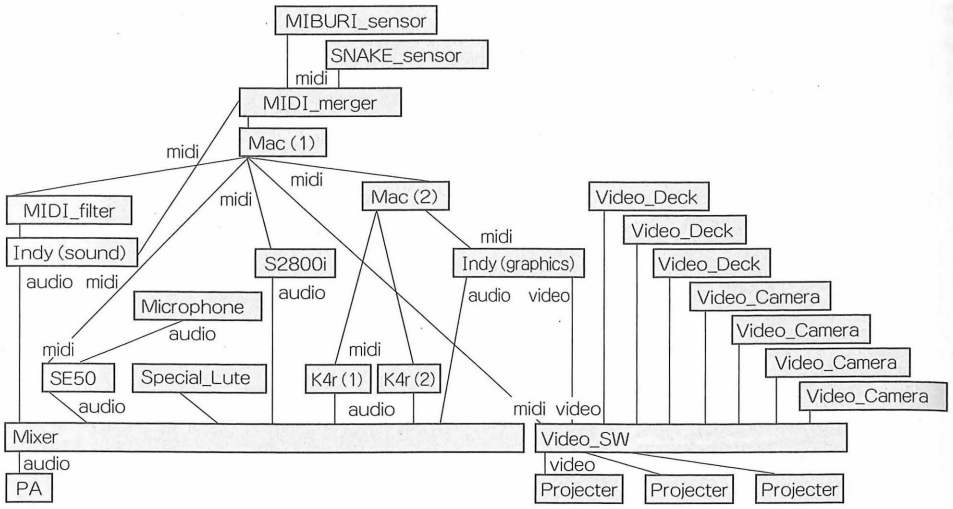
図6-14はこの「SNAKEMAN」センサの回路図，**リスト6-8**(p.83)はそのAKI-80プログラム(SNAKE.SRC)です．

ここで重要なのは，たんに「赤外線を遮った」「赤外線が通過した」というON/OFF情報ではないところです．AKI-80のセンシングは人間の身体が光線をよぎる時間よりも十分に高速ですから，「ビームがOFFとなる時間」を計測することが可能です．そこで，ビームがOFFされて次にONされるというイベント発生の際に，この時間を「遮るスピード」の情報としてMIDI出力しています．これは，ほとんどの電子鍵盤楽器が「個々の鍵盤の打鍵強度」の検出のために，鍵盤の下にある二つのスイッチがON/OFFされる時間差からMIDIのベロシティ情報を抽出しているのとまったく同じ発想によるものです．

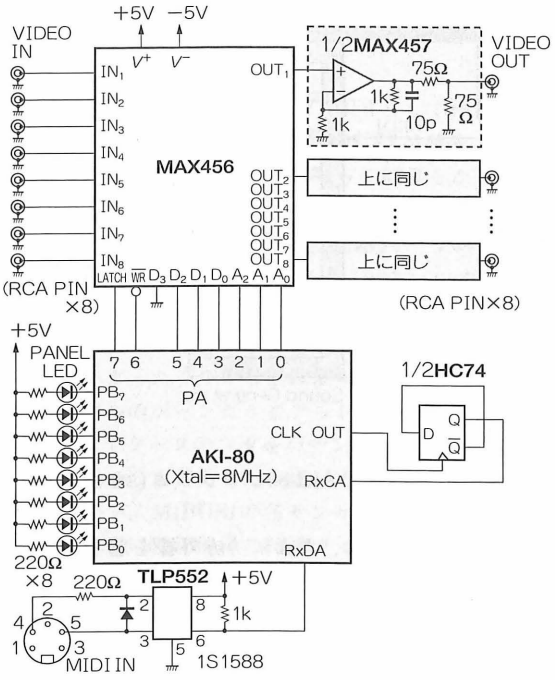
実際にSNAKEMANセンサを使ってみると，確かにビームを遮断する速度に応じて値の変わるデータが送信されるのですが，ここで新しい問題点も出てきました．それは，「人間の手は，複数の細い指からできて

MIDIマージャ　複数のMIDI情報を受信して，それら相互の情報に矛盾がないように管理しながら合流（マージ）したMIDI情報を出力する装置．

〈図6-17〉
Asian Edge
のシステム図



〈図6-18〉MIDIビデオ・スイッチャの回路



いる」という当然の事実です．つまり，ゆっくりと手を下ろしても，センサとしては，個々の指ごとに高速に移動したというデータを出してしまうのです．しっかり手を握っていれば問題ありませんが，これは演奏姿勢としてやや不自然です．そこで筆者は，MIDI情報を受けるMAXのパッチに**図6-15**のようなアルゴリズムを仕込むことで対応しました．

つまり，一定時間のマスキングをかけて，あまり不自然に(個々の指として解釈しないと実現できないほど高速)移動する場合には，一定時間後まで無視するというものです．AKI-80のソフトに処理を入れてROMとして固定すると，将来的に「指単位の情報」が欲しい時に困りますから，ここではソフト側で対応してみたというものです．

MIDIビデオ・スイッチャと新作のComputer Musicシステム

さて，本章の最後に紹介する例は，最近の筆者の作品でアプローチしている「マルチメディア・アート」に関連するツールです．

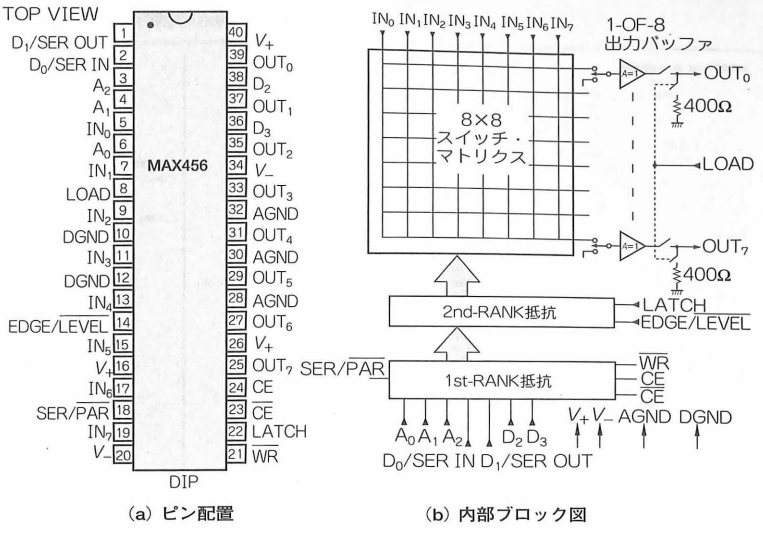
筆者はCGアーティスト(**由良泰人**氏)と組んで，いくつかのインタラクティブ・マルチメディア・アート作品を発表してきましたが，ここではMIDIを音楽だけでなく，映像系とのリンクにも活用しています．たとえば**コラボレーション作品**“Muromachi”では，ステージ上のPerformerが由良さん制作のCGソフトで「お絵描き」すると，この描画情報はAMIGAコンピュータからMIDIで音楽系を駆動して，聴衆はステージ後方のスクリーンに投射されるCG映像とともに変化する音響を楽しみました．

図6-16はその“Muromachi”のシステム図で，このときには音楽系では4台のノート・パソコンと8台のシンセサイザ(うち4台は自作)をMIDIで結びつけた分散処理を行っています．この翌年には，コンサート形式でなく「来場者が自分で実際に試して体験する」インスタレーション版に改訂した“Muromachi3”を「芸術祭典・京」で発表しましたが，このときはポータビリティを考慮して，ノート・パソコンと音源それぞれ1台で音楽系を実現するようにしています．

そして，**図6-17**は新作“Asian Edge”のシステム図です．この作品はコラボレーションというよりも，Computer Music作品に由良さんの映像サポートをお

μPD71051　NECのCMOSのUSART．オリジナルはインテルの8251．

〈図6-19〉(2)
ビデオIC MAX456のピン配置と内部ブロック図



(a) ピン配置　　(b) 内部ブロック図

願いしたというものなのですが，映像系はかなり充実しています．CGは音楽系とは別に用意したIndyで行い，センサからのMIDI情報によって変化する3D映像を描画するオリジナル・ソフトです．さらに**BGV**としてイメージ映像を複数系列，並列に再生しておいて，これをPerformerの動作によって瞬時に切り替えるという「MIDIビデオ・スイッチャ」というマシンも新規開発しました．

会場にはビデオ・プロジェクタが3系列も用意され，このマシンはIndyのCG，3台のビデオ，4台のビデオ・カメラからの映像を自在にスイッチングして3台のビデオ・プロジェクタに分配します．**図6-17**のシステム図には，さらに本書でここまでに述べた各種の専用マシンも駆使されていることがわかるでしょう．

図6-18は，このMIDIビデオ・スイッチャの回路図です．昔はビデオ回路といえばトランジスタなどのディスクリート素子による面倒なものだったのですが，最近ではまったくディジタル感覚で製作できるようになりました．

筆者が利用したのは，マキシム社のビデオ・スイッチICのMAX456(**図6-19**)とビデオ・バッファICのMAX457で，ほぼデータブックそのままの回路となっています．MAX456は，8入力8出力のビデオ信号をマトリクス・スイッチで任意に切り替えられるすぐれものので，このマシンを簡単なMAXパッチによってMIDI経由で高速に切り替えてみると，**サブリミナル効果**のような瞬間的な映像の挟み込みや，複数のビデオ・カメラを切り替えてのアニメーション効果などがリアルタイムに簡単に実現できてしまいます．

このマシンのAKI-80も，これまでのシステムと同様にMIDI入力を解釈して，新たに規定された情報にヒットしたときだけ，ビデオ・スイッチICの選択入力端子にパラレル出力しているだけというものです．パネルのコネクタまでの内部配線にシールド線を使ったりという程度の違いだけで，ここまでのシステムがこんなに簡単になったというのは感慨深いものがあります(同等なので，リストは省略)．

筆者がこれまでにAKI-80を活用して製作してきた音楽関連，メディアアート関連のオリジナル・マシンはすでに数十種類になっていますから，ここで紹介したのはごく一部です．AKI-80によって「アイデアを実現するまでの手間」が相当にスリム化されていることはおわかりいただけたと思います．たいていのマシンは，構想と穴開けに1日，はんだ付けに1日，ソフト開発とデバッグに1日という3日仕事で完成してしまうのです．

たかが8ビット・マイコン，されど8ビット・マイコン，このようなシステム開発を馬鹿にして実際に作りもしない(できない)ソフト屋さんにとっては理解できないかもしれませんが，世界に一つしかない自作マシンの完成の喜びというのは，こんな簡単なシステムであっても十分な手応えです．そして，この程度でできてしまうマシンを作れなければ，市販の「仕様がメーカに限定されている」機器を，嫌々ながらでも使わなければなりません．とくにMIDI関連について言えば，これは音楽の可能性をかなり狭めることになっているのが，メーカに提供されている現在のMIDI製品の最大の課題だと思っています．

◆引用文献◆

(1) ヤマハ，MIBURIマニュアル
(2) マキシム，データブック
(3) パックス・コーポレーション，PowerGlove取り扱い説明書

SGI Indy　シリコン・グラフィクス社のグラフィック・ワークステーションで世界で初めて，パーソナルでも使えるマシンとして登場したマシン．

〈リスト6-2〉PowerGloveの制御ソフト[GLOVE.SRC] ①

```
;##### RAM Map #####
            dseg
            org         0000h
tx_fifo     ds          256
tx_top      ds          1
tx_end      ds          1
timer_flag  ds          4
glove_status ds         1
x_value     ds          2
y_value     ds          2
counts      ds          4
channel     ds          1

;##### I/O Map #####
            cseg
ctc_0       equ         0010h
s_gen       equ         0017h
sio_a       equ         0018h
sio_b       equ         001ah
pio_a       equ         001ch
pio_b       equ         001eh
led_up      equ         0080h
led_down    equ         0081h
led_left    equ         0082h
led_right   equ         0083h

;##### MACRO #####
io_set      macro       @1,@2
            ld          a,@2
            out         (@1+1),a
            endm
io_put      macro       @1,@2
            ld          a,@2
            out         (@1+0),a
            endm

;##### RESET #####
            org         0000h
            ld          sp,0ffffh
            di
            jp          main

;##### INT / NMI #####
            org         0020h
            dw          _timer_
_timer_:
            ex          af,af'
            ld          a,1
            ld          (timer_flag+0),a
            ex          af,af'
            ei
            reti
            org         0066h
            retn

;##### Data Table #####
            org         0100h
display_table_1:
            db          10h,10h,10h,10h,10h,10h,10h,10h ;  1 -  8
            db          18h,18h,18h,18h,18h,18h,18h,18h ;  9 - 16
            db          08h,08h,08h,08h,0ch,0ch,0ch,0ch ; 17 - 24
            db          0ch,0ch,0ch,0ch,04h,04h,04h,04h ; 25 - 32
            db          06h,06h,06h,06h,06h,06h,06h,06h ; 33 - 40
            db          02h,02h,02h,02h,03h,03h,03h,03h ; 41 - 48
            db          03h,03h,03h,03h,01h,01h,01h,01h ; 49 - 56
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 57 - 64
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 65 - 72
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 73 - 80
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 81 - 88
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 89 - 96
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 97 - 104
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 105 - 112
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 113 - 120
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 121 - 128
            org         0200h
display_table_2:
            db          10h,10h,10h,10h,10h,10h,10h,10h ;  1 -  8
            db          10h,10h,10h,10h,08h,08h,08h,08h ;  9 - 16
            db          08h,08h,08h,08h,08h,08h,08h,08h ; 17 - 24
            db          04h,04h,04h,04h,04h,04h,04h,04h ; 25 - 32
            db          04h,04h,04h,04h,02h,02h,02h,02h ; 33 - 40
            db          02h,02h,02h,02h,02h,02h,02h,02h ; 41 - 48
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 49 - 56
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 57 - 64
            db          01h,01h,01h,01h,00h,00h,00h,00h ; 65 - 72
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 73 - 80
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 81 - 88
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 89 - 96
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 97 - 104
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 105 - 112
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 113 - 120
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 121 - 128
            org         0300h
display_table_3:
            db          00h,00h,00h,00h,00h,00h,00h,00h ;  1 -  8
            db          00h,00h,00h,00h,00h,00h,00h,00h ;  9 - 16
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 17 - 24
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 25 - 32
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 33 - 40
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 41 - 48
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 49 - 56
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 57 - 64
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 65 - 72
            db          03h,03h,03h,03h,02h,02h,02h,02h ; 73 - 80
            db          06h,06h,06h,06h,06h,06h,06h,06h ; 89 - 96
            db          04h,04h,04h,04h,0ch,0ch,0ch,0ch ; 97 - 104
            db          0ch,0ch,0ch,0ch,08h,08h,08h,08h ; 105 - 112
            db          18h,18h,18h,18h,18h,18h,18h,18h ; 113 - 120
            db          10h,10h,10h,10h,10h,10h,10h,10h ; 121 - 128
            org         0400h
display_table_4:
            db          00h,00h,00h,00h,00h,00h,00h,00h ;  1 -  8
            db          00h,00h,00h,00h,00h,00h,00h,00h ;  9 - 16
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 17 - 24
            db           00h,00h,00h,00h,00h,00h,00h,00h ; 25 - 32
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 33 - 40
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 41 - 48
            db          00h,00h,00h,00h,00h,00h,00h,00h ; 49 - 56
            db          00h,00h,00h,00h,01h,01h,01h,01h ; 57 - 64
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 65 - 72
            db          01h,01h,01h,01h,01h,01h,01h,01h ; 73 - 80
            db          02h,02h,02h,02h,02h,02h,02h,02h ; 81 - 88
            db          02h,02h,02h,02h,04h,04h,04h,04h ; 89 - 96
            db          04h,04h,04h,04h,04h,04h,04h,04h ; 97 - 104
            db          08h,08h,08h,08h,08h,08h,08h,08h ; 105 - 112
            db          08h,08h,08h,08h,10h,10h,10h,10h ; 113 - 120
            db          10h,10h,10h,10h,10h,10h,10h,10h ; 121 - 128

;##### Main #####
main:
            ld          hl,08000h
            ld          a,090h
_ram_clear_loop:
            ld          (hl),0
            inc         hl
            cp          h
            jr          nc,_ram_clear_loop
            io_put      led_up,0ffh
            io_put      led_down,0ffh
            io_put      led_right,0ffh
            io_put      led_left,0ffh
            io_set      pio_a,0cfh              ; Mode 3
            io_set      pio_a,00001111b ; 0:Out / 1:In
            io_set      pio_a,007h              ; Interrupt Disable
            io_set      pio_b,0cfh              ; Mode 3
            io_set      pio_b,00000000b ; 0:Out / 1:In
            io_set      pio_b,007h              ; Interrupt Disable
            io_put      pio_a,10000000b
            io_put      pio_b,0ffh
            io_put      s_gen,00000000b ; Clock Generator
            io_set      sio_a,00011000b ; Channel Reset A
            io_set      sio_a,00000100b ; Resister Point = 4
            io_set      sio_a,10000100b ; Mode
            io_set      sio_a,00000001b ; Resister Point = 1
            io_set      sio_a,00000000b ; Interrupt Mode
            io_set      sio_a,00000101b ; Resister Point = 5
            io_set      sio_a,01101000b ; Transmit Start
            io_put      ctc_0,20h                ; Int. Address
            io_put      ctc_0,10100101b
            io_put      ctc_0,80                 ; about 5msec
            ld          a,0eh
            ld          (channel),a
            call        centering
            im          2
            ei
loop:
            call        glove_scan
            call        display_timer
            call        tx_data_check
            jr          loop

;##### Subroutines #####
x_value_set:
            ld          a,(x_value+1)
            ld          b,a
            ld          a,(x_value)
            cp          b
            ret         z
            ld          (x_value+1),a
            ld          a,(channel)
            or          0b0h
            ld          b,a
            call        tx_data_set
            ld          b,10
            call        tx_data_set
            ld          a,(x_value)
            ld          b,a
            call        tx_data_set
            call        x_value_disp
            ret
y_value_set:
            ld          a,(y_value+1)
            ld          b,a
            ld          a,(y_value)
            cp          b
            ret         z
            ld          (y_value+1),a
            ld          a,(channel)
            or          0b0h
            ld          b,a
            call        tx_data_set
            ld          b,7
            call        tx_data_set
            ld          a,(y_value)
            ld          b,a
            call        tx_data_set
            call        y_value_disp
            ret
tx_data_check:
            ld          a,(tx_end)
            ld          b,a
            ld          a,(tx_top)
            cp          b
            ret         z
            io_set      sio_a,00000000b ; Resister Point = 0
            in          a,(sio_a+1)
            bit         2,a
            ret         z
            ld          hl,tx_fifo
            ld          l,b
            ld          a,(hl)
            out         (sio_a),a
            ld          a,b
            inc         a
            ld          (tx_end),a
            ret
tx_data_set:
            ld          hl,tx_fifo
            ld          a,(tx_top)
            ld          l,a
            inc         a
            ld          (tx_top),a
            ld          (hl),b
            ret
centering:
            ld          a,040h
            ld          (x_value),a
            ld          (y_value),a
            call        x_value_set
            call        y_value_set
            ret
```


物理モデル音源　楽器から出てくる音響をサンプリングするような従来の方法でなく，弦や膜の振動をDSPでシミュレーションして楽器の音源とする新しい方式の音源．

〈リスト6-2〉PowerGloveの制御ソフト(GLOVE.SRC)②

```
;##### Panel LED Display #####
display_timer:
          ld        a,(timer_flag+1)
          cp        50                    ; about 250msec
          ret       c
          xor       a
          ld        (timer_flag+1),a
          ld        a,(timer_flag+2)
          inc       a
          ld        (timer_flag+2),a
          call      x_value_disp
          call      y_value_disp
          xor       a
          ld        (counts+0),a
          ld        (counts+1),a
          ld        (counts+2),a
          ld        (counts+3),a
          ret
x_value_disp:
          ld        a,(timer_flag+2)
          bit       0,a
          jr        nz,_x_disp_off
          ld        hl,display_table_3
          ld        de,display_table_1
          jr        _x_disp_mix
_x_disp_off:
          ld        hl,display_table_4
          ld        de,display_table_2
_x_disp_mix:
          ld        a,(x_value)
          ld        l,a
          ld        a,(hl)
          xor       0ffh
          out       (led_right),a
          ld        h,d
          ld        a,(hl)
          xor       0ffh
          out       (led_left),a
          ret
y_value_disp:
          ld        a,(timer_flag+2)
          bit       0,a
          jr        z,_y_disp_off
          ld        hl,display_table_3
          ld        de,display_table_1
          jr        _y_disp_mix
_y_disp_off:
          ld        hl,display_table_4
          ld        de,display_table_2
_y_disp_mix:
          ld        a,(y_value)
          ld        l,a
          ld        a,(hl)
          xor       0ffh
          out       (led_up),a
          ld        h,d
          ld        a,(hl)
          xor       0ffh
          out       (led_down),a
          ret

;##### Power Glove Event Check #####
;-------------------------------------------------------------------------*
; Bit  :  7     6     5     4     3     2     1     0
;       right  left down   up   start select first(B) thumb(A)
;-------------------------------------------------------------------------*
glove_check:
          ld        a,(glove_status)
          and       00001100b
          jr        z,_glove_cont
          call      centering
          ret
_glove_cont:
          ld        a,(glove_status)
          bit       1,a
          jp        z,_glove_2
          ld        c,3                   ; Fast Skip
          ld        a,(glove_status)
          bit       4,a                   ; Up ?
          jr      z,_glove_1
          ld        a,(counts+0)
          add       a,c
          ld        (counts+0),a
          bit       3,a
          jr        z,_glove_1
          ld        a,(y_value)
          cp        7fh
          jr        z,_glove_1
          inc       a
          ld        (y_value),a
          call      y_value_set
          xor       a
          ld        (counts+0),a
          ld        (counts+1),a
_glove_1:
          ld        a,(glove_status)
          bit       5,a                   ; Down ?
          jr        z,_glove_2
          ld        a,(counts+1)
          add       a,c
          ld        (counts+1),a
          bit       3,a
          jr        z,_glove_2
          ld        a,(y_value)
          cp        0
          jr        z,_glove_2
          dec       a
          ld        (y_value),a
          call      y_value_set
          xor       a
          ld        (counts+0),a
          ld        (counts+1),a
_glove_2:
          ld        a,(glove_status)
          bit       0,a
          ret       z
          ld        c,2                   ; Slow Skip
          ld        a,(glove_status)
          bit       6,a                   ; Left ?
          jr        z,_glove_3
          ld        a,(counts+2)
          add       a,c
          ld        (counts+2),a
          bit       3,a
          jr        z,_glove_3
          ld        a,(x_value)
          cp        0
          jr        z,_glove_3
          dec       a
          ld        (x_value),a
          call      x_value_set
          xor       a
          ld        (counts+2),a
          ld        (counts+3),a
_glove_3:
          ld        a,(glove_status)
          bit       7,a                   ; Right ?
          ret       z
          ld        a,(counts+3)
          add       a,c
          ld        (counts+3),a
          bit       3,a
          ret       z
          ld        a,(x_value)
          cp        7fh
          ret       z
          inc       a
          ld        (x_value),a
          call      x_value_set
          xor       a
          ld        (counts+2),a
          ld        (counts+3),a
          ret

;##### Power Glove Status Scan #####
glove_scan:
          ld        a,(timer_flag+0)
          cp        0
          ret       z
          xor       a
          ld        (timer_flag+0),a
          ld        a,(timer_flag+1)
          inc       a
          ld        (timer_flag+1),a
          ld        b,0
          call      p_s_pulse
          in        a,(pio_a)
          bit       0,a
          jr        z,1$
          set       0,b
1$:
          call      clock_shift
          in        a,(pio_a)
          bit       0,a
          jr        z,2$
          set       1,b
2$:
          call      clock_shift
          in        a,(pio_a)
          bit       0,a
          jr        z,3$
          set       2,b
3$:
          call      clock_shift
          in        a,(pio_a)
          bit       0,a
          jr        z,4$
          set       3,b
4$:
          call      clock_shift
          in        a,(pio_a)
          bit       0,a
          jr        z,5$
          set       4,b
5$:
          call      clock_shift
          in        a,(pio_a)
          bit       0,a
          jr        z,6$
          set       5,b
6$:
          call      clock_shift
          in        a,(pio_a)
          bit       0,a
          jr        z,7$
          set       6,b
7$:
          call      clock_shift
          in        a,(pio_a)
          bit       0,a
          jr        z,8$
          set       7,b
8$:
          call      clock_shift
          ld        a,b
          out       (pio_b),a
          xor       0ffh
          ld        (glove_status),a      ; New Status
          call      glove_check
          ret
p_s_pulse:
          io_put    pio_a,11000000b
          nop
          nop
          io_put    pio_a,10000000b
          ret
clock_shift:
          io_put    pio_a,00000000b
          nop
          nop
          io_put    pio_a,10000000b
          ret

          end
```

〈リスト6-4〉AKI-80のアナログ←→MIDI変換器プログラム[IRCAM.SRC]①

```
;##### RAM Map #####
        dseg
        org     0000h
rx_fifo         ds              2048
tx_fifo         ds              2048
rx_top          ds              2
rx_end          ds              2
tx_top          ds              2
tx_end          ds              2
rsb             ds              1
dcb             ds              1
channel         ds              1
keyno           ds              1
timer_flag      ds              1
timer           ds              1
dac_no          ds              1
dac_status      ds              32 ; on=[0-254] , off=[255]
sel_4051        ds              1
ad_no           ds              1
ad_comm         ds              1
ad_status       ds              32 ; off=[0] , on=[1] ,
converting=[2]
ad_old          ds              32
eoc_mask        ds              1
threshold       ds              1
outstatus       ds              1
led_mode        ds              1
led_ch          ds              1
led_data        ds              1
led_phase       ds              2

;##### I/O Map #####
        cseg
ctc_0           equ             0010h
sio_a           equ             0018h
sio_b           equ             001ah
pio_a           equ             001ch
pio_b           equ             001eh

;##### MACRO #####
io_set  macro   @1,@2
        ld      a,@2
        out     (@1+1),a
        endm
io_put  macro   @1,@2
        ld      a,@2
        out     (@1+0),a
        endm
dtra_lp macro
        io_set  sio_a,00000101b ; Resister Point = 5
        io_set  sio_a,11101000b ; Transmit Start, DTR = Low
        io_set  sio_a,00000101b ; Resister Point = 5
        io_set  sio_a,01101000b ; Transmit Start, DTR = High
        endm
dtrb_lp macro
        io_set  sio_b,00000101b ; Resister Point = 5
        io_set  sio_b,11100000b ; Transmit Disable, DTR = Low
        io_set  sio_b,00000101b ; Resister Point = 5
        io_set  sio_b,01100000b ; Transmit Disable, DTR = High
        endm
pb_set  macro   @1
        out     (pio_b),a
        dtra_lp
        ld      a,11111111b
        res     @1,a
        out     (pio_b),a
        dtrb_lp
        ld      a,11111111b
        out     (pio_b),a
        dtrb_lp
        endm
pb_init macro
        xor     a
        pb_set  0
        xor     a
        pb_set  1
        xor     a
        pb_set  2
        xor     a
        pb_set  3
        endm
led_out macro
        ld      a,(led_data)
        xor     0ffh
        pb_set  4
        endm
led_set macro
        ld      a,(led_mode)
        rrca
        rrca
        rrca
        and     11100000b
        ld      b,a
        ld      a,(led_ch)
        and     00011111b
        or      b
        xor     0ffh
        pb_set  5
        endm
dac_out macro
        pb_set  6               ; input = [A]
        endm
latch_7 macro
        ld      a,(sel_4051)
        pb_set  7
        endm

;##### RESET #####
        org     0000h
        ld      sp,09fffh
        di
        jp      main

;##### INT / NMI #####
        org     0020h
        dw      _midi_
_midi_:
        ex      af,af'
        exx
        ld      de,(rx_top)
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        in      a,(sio_a+0)
        ld      (hl),a
        inc     de
        res     3,d
        ld      (rx_top),de
        exx
        ex      af,af'
        ei
        reti
        org     0066h
        retn
        org     0070h
        dw      _timer_
_timer_:
        ex      af,af'
        ld      a,1
        ld      (timer_flag),a
        ex      af,af'
        ei
        reti

;##### Main #####
        org     0100h
main:
        ld      hl,08000h
        ld      a,09fh
_ram_clear_loop:
        ld      (hl),0
        inc     hl
        cp      h
        jr      nc,_ram_clear_loop
        io_set  pio_a,0cfh      ; Mode 3
        io_set  pio_a,11111111b ; 0:Out / 1:In
        io_set  pio_a,007h      ; Interrupt Disable
        io_set  pio_b,0cfh      ; Mode 3
        io_set  pio_b,00000000b ; 0:Out / 1:In
        io_set  pio_b,007h      ; Interrupt Disable
        io_put  ctc_0,70h       ; Int. Address
        io_put  ctc_0,10100101b ; Timer Mode
        io_put  ctc_0,157       ; about 10msec
        io_set  sio_b,00011000b ; Channel Reset B
        io_set  sio_b,00000100b ; Resister Point = 4
        io_set  sio_b,11000100b ; Mode
        io_set  sio_b,00000001b ; Resister Point = 1
        io_set  sio_b,00000000b ; Interrupt Mode
        io_set  sio_b,00000101b ; Resister Point = 5
        io_set  sio_b,01100000b ; Transmit Disable, DTR = High
        io_set  sio_b,00000010b ; Resister Point = 2
        io_set  sio_b,20h       ; Vector Address
        io_set  sio_a,00011000b ; Channel Reset A
        io_set  sio_a,00000100b ; Resister Point = 4
        io_set  sio_a,11000100b ; Mode
        io_set  sio_a,00000001b ; Resister Point = 1
        io_set  sio_a,00010000b ; Interrupt Mode
        io_set  sio_a,00000101b ; Resister Point = 5
        io_set  sio_a,01101000b ; Transmit Start, DTR = High
        io_set  sio_a,00000011b ; Resister Point = 3
        io_set  sio_a,11000001b ; Receive Start
        nop
        latch_7                 ; Latch 7 normal waiting
        pb_init
        ld      b,32
lp_001:
        push    bc
        call    dac_off         ; all DAC off
        pop     bc
        djnz    lp_001
        led_out                 ; Data LED 8bit
        led_se                  ; Mode/Channel LED 3+5bit
        ld      a,4
        ld      (threshold),a
        ld      a,0a0h
        ld      (outstatus),a
        ld      a,00000001b
        ld      (led_phase+0),a
        xor     a
        ld      (led_phase+1),a
        im      2
        ei
        in      a,(sio_a+0)     ; dummy read
loop:
        call    rx_data_check
        call    tx_data_check
        call    dac_trans       ; DAC data re-new
        call    ad_convert
        jr      loop

;##### Timer Subroutines #####
timer_check:
        ld      a,(timer_flag)
        cp      0
        ret     z
        xor     a
        ld      (timer_flag),a
        ld      a,(timer)
        inc     a
        ld      (timer),a
        cp      100             ; about 1sec demo scan
        ret     c
        xor     a
        ld      (timer),a
        ld      a,(led_phase+0)
        cp      00000001b
        jp      z,_timer_001
        cp      00000010b
        jp      z,_timer_010
        cp      00000100b
        jp      z,_timer_100
        ld      a,00000001b
        ld      (led_phase+0),a
        xor     a
        ld      (led_phase+1),a
        ret
_timer_001:
        ld      (led_mode),a
        ld      a,(led_phase+1)
        ld      (led_ch),a
        ld      c,a
```


ADC0809　8ビット8チャネルのA-Dコンバータ．やや低速だがパラレル出力のため，簡単に実験を行うのに適している．オリジナルはアナログ・デバイセズ社．

〈リスト6-4〉AKI-80のアナログ←→MIDI変換器プログラム[IRCAM.SRC]②

```
        ld      b,0
        ld      hl,ad_status
        add     hl,bc
        ld      a,(hl)
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
        ld      a,(led_phase+1)
        inc     a
        ld      (led_phase+1),a
        bit     5,a
        ret     z
        ld      a,00000010b
        ld      (led_phase+0),a
        ld      a,00000001b
        ld      (led_phase+1),a
        ret
_timer_010:
        ld      (led_mode),a
        ld      a,(led_phase+1)
        ld      (led_ch),a
        cp      00010000b
        jr      nz,_timer_010_status
        ld      a,(threshold)
        jr      _timer_010_threshold
_timer_010_status:
        ld      a,(outstatus)
_timer_010_threshold:
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
        ld      a,(led_phase+1)
        sla     a
        ld      (led_phase+1),a
        bit     5,a
        ret     z
        ld      a,00000100b
        ld      (led_phase+0),a
        xor     a
        ld      (led_phase+1),a
        ret
_timer_100:
        ld      (led_mode),a
        ld      a,(led_phase+1)
        ld      (led_ch),a
        ld      c,a
        ld      b,0
        ld      hl,dac_status
        add     hl,bc
        ld      a,(hl)
        cp      255
        jr      nz,_timer_dac_off
        xor     a
        jr      _timer_dac_mix
_timer_dac_off:
        srl     a
_timer_dac_mix:
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
        ld      a,(led_phase+1)
        inc     a
        ld      (led_phase+1),a
        bit     5,a
        ret     z
        ld      a,00000001b
        ld      (led_phase+0),a
        xor     a
        ld      (led_phase+1),a
        ret

;##### A/D Subroutunes #####
ad_convert:
        call    timer_check
        ld      hl,ad_status
        ld      a,(ad_no)
        ld      c,a
        ld      b,0
        add     hl,bc
        ld      a,(hl)
        cp      0
        jr      nz,_ad_conv
        ld      a,(ad_no)
        inc     a
        and     00011111b
        ld      (ad_no),a
        ret
_ad_conv:
        cp      2
        jp      z,_ad_check
_ad_new:
        cp      1
        jr      z,_ad_new_go
        xor     a
        ld      (hl),a
        ret
_ad_new_go:
        ld      a,2
        ld      (hl),a
        ld      a,(ad_no)
        and     00000111b
        ld      (ad_comm),a
        ld      a,(ad_no)
        and     00011000b
        cp      00000000b
        jp      z,_ad_0
        cp      00001000b
        jp      z,_ad_1
        cp      00010000b
        jp      z,_ad_2
        jp      _ad_3
_ad_0:
        ld      a,(ad_comm)
        pb_set  0
        ld      a,(ad_comm)
        o       00001000b
        pb_set  0
        ld      a,(ad_comm)
        and     11110111b
        pb_set  0
        ld      a,00000001b
        ld      (eoc_mask),a
        ret
_ad_1:
        ld      a,(ad_comm)
        pb_set  1
        ld      a,(ad_comm)
        or      00001000b
        pb_set  1
        ld      a,(ad_comm)
        and     11110111b
        pb_set  1
        ld      a,00000010b
        ld      (eoc_mask),a
        ret
_ad_2:
        ld      a,(ad_comm)
        pb_set  2
        ld      a,(ad_comm)
        or      00001000b
        pb_set  2
        ld      a,(ad_comm)
        and     11110111b
        pb_set  2
        ld      a,00000100b
        ld      (eoc_mask),a
        ret
_ad_3:
        ld      a,(ad_comm)
        pb_set  3
        ld      a,(ad_comm)
        or      00001000b
        pb_set  3
        ld      a,(ad_comm)
        and     11110111b
        pb_set  3
        ld      a,00001000b
        ld      (eoc_mask),a
        ret
_ad_check:
        ld      a,(sel_4051)
        and     01111111b
        pb_set  7
        in      a,(pio_a)
        and     00001111b
        ld      b,a
        ld      a,(sel_4051)
        pb_set  7
        ld      a,(eoc_mask)
        and     b
        ret     z
        ld      a,(ad_no)
        and     00011000b
        cp      00000000b
        jp      z,_ad_d_0
        cp      00001000b
        jp      z,_ad_d_1
        cp      00010000b
        jp      z,_ad_d_2
        jp      _ad_d_3
_ad_d_0:
        ld      a,00010000b
        pb_set  0
        in      a,(pio_a)
        ld      d,a
        xor     a
        pb_set  0
        jp      _ad_trsnsmit
_ad_d_1:
        ld      a,00010000b
        pb_set  1
        in      a,(pio_a)
        ld      d,a
        xor     a
        pb_set  1
        jp      _ad_trsnsmit
_ad_d_2:
        ld      a,00010000b
        pb_set  2
        in      a,(pio_a)
        ld      d,a
        xor     a
        pb_set  2
        jp      _ad_trsnsmit
_ad_d_3:
        ld      a,00010000b
        pb_set  3
        in      a,(pio_a)
        ld      d,a
        xor     a
        pb_set  3
_ad_trsnsmit:
        ld      hl,ad_status
        ld      a,(ad_no)
        ld      c,a
        ld      b,0
        add     hl,bc
        ld      a,1
        ld      (hl),a
        ld      hl,ad_old
        ld      a,(ad_no)
        ld      c,a
        ld      b,0
        add     hl,bc
        srl     d               ; [d] = new 7bit data
        ld      a,d
        ld      b,(hl)          ; [b] = old
        cp      b               ; a - b ?
        jr      c,_ad_inv       ; a < b
        sub     b
_ad_cmp:
        ld      b,a
        ld      a,(threshold)
        cp      b               ; (a-b)<thrshold ?
        jp      nc,_ad_out      ; then NOP
        ld      (hl),d          ; new [old] data set
```


通過センサ　赤外線LEDとフォト・トランジスタを利用して，この赤外線ビームを遮断したことを検出するセンサ．

〈リスト6-4〉AKI-80のアナログ←→MIDI変換器プログラム[IRCAM.SRC]③

```
        jr      _ad_event
_ad_inv:
        ld      c,a
        ld      a,b
        sub     c
        jr      _ad_cmp
_ad_event:
        ld      c,d
        ld      a,(outstatus)
        ld      b,a
        call    tx_fifo_set
        ld      a,(ad_no)
        ld      b,a
        call    tx_fifo_set
        ld      b,c
        call    tx_fifo_set
        ld      a,00000111b
        ld      (led_mode),a
        ld      a,(ad_no)
        ld      (led_ch),a
        ld      a,c
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
_ad_out:
        ld      a,(ad_no)
        inc     a
        and     00011111b
        ld      (ad_no),a
        ret

;##### D/A Subroutines #####
dac_trans:
        latch_7
        ld      a,(dac_no)
        inc     a
        and     00011111b
        ld      (dac_no),a
        ld      c,a
        ld      hl,dac_status
        ld      b,0
        add     hl,bc
        ld      a,(hl)
        cp      255             ; D/A on ?
        jr      nz,_dac_set
        ret
dac_off:
        latch_7
        ld      hl,dac_status
        ld      a,b             ; off channel = [B]
        ld      (dac_no),a
        ld      c,a
        ld      b,0
        add     hl,bc
        ld      a,255           ; off = 255
        ld      (hl),a
        xor     a
_dac_set:
        dac_out                 ; input = [A]
        ld      a,(dac_no)
        or      11111000b
        ld      (sel_4051),a
        ld      a,(dac_no)
        and     00011000b
        cp      00000000b
        jr      z,_dac_off_00
        cp      00001000b
        jr      z,_dac_off_01
        cp      00010000b
        jr      z,_dac_off_10
        jr      _dac_off_11
_dac_off_00:
        ld      a,(sel_4051)
        and     11110111b
        jr      _dac_off_exit
_dac_off_01:
        ld      a,(sel_4051)
        and     11101111b
        jr      _dac_off_exit
_dac_off_10:
        ld      a,(sel_4051)
        and     11011111b
        jr      _dac_off_exit
_dac_off_11:
        ld      a,(sel_4051)
        and     10111111b
_dac_off_exit:
        pb_set  7
        ret

;##### MIDI Subroutines #####
tx_data_check:
        ld      de,(tx_end)
        ld      hl,(tx_top)
        and     a               ; CY <-- 0
        sbc     hl,de
        ret     z
        io_set  sio_a,00000000b ; Resister Point = 0
        in      a,(sio_a+1)
        bit     2,a
        ret     z
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      a,(hl)
        out     (sio_a),a
        inc     de
        res     3,d
        ld      (tx_end),de
        ret
tx_fifo_set:
        ld      de,(tx_top)
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      (hl),b
        inc     de
        res     3,d
        ld      (tx_top),de
        ret
rx_data_check:
        ld      de,(rx_end)
        ld      hl,(rx_top)
        and     a               ; CY <-- 0
        sbc     hl,de
        ret     z
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        ld      b,(hl)
        inc     de
        res     3,d
        ld      (rx_end),de
        bit     7,b
        jr      z,50$           ; running
        ld      a,b
        cp      0f8h
        ret     nc
        cp      0f0h
        jr      c,10$
        xor     a
        ld      (rsb),a
        ret
10$:
        ld      a,b
        and     00001111b
        ld      (channel),a
        ld      a,b
        and     11110000b
        ld      (rsb),a
        xor     a
        ld      (dcb),a
        ret
50$:
        ld      a,(rsb)
        cp      0
        ret     z
        cp      0c0h
        ret     z
        cp      0d0h
        ret     z
        ld      a,(dcb)
        cp      0
        jr      nz,90$
        inc     a
        ld      (dcb),a
        ld      a,b
        ld      (keyno),a
        ret
90$:
        xor     a
        ld      (dcb),a
        ld      c,b             ; 3rd Data
        ld      a,(rsb)
        ld      b,a
        ld      a,(channel)
        or      b
        cp      0afh
        ret     nz
        ld      a,(keyno)
        and     11110000b
        cp      00110000b
        jp      z,_af_30_4f
        cp      01000000b
        jp      z,_af_30_4f
        ld      a,(keyno)
        and     11100000b
        jp      z,_af_00_1f
        ld      a,(keyno)
        cp      20h
        jp      z,_af_20
        cp      21h
        jp      z,_af_21
        cp      22h
        jp      z,_af_22
        ret
_af_00_1f:
        ld      a,c
        cp      0
        jr      z,_af_00_ok
        cp      1
        jr      z,_af_00_ok
        ret
_af_00_ok:
        ld      hl,ad_status
        ld      a,(keyno)
        ld      e,a
        ld      d,0
        add     hl,de
        ld      (hl),c
        ld      a,00000001b
        ld      (led_mode),a
        ld      (led_phase+0),a
        ld      a,(keyno)
        ld      (led_ch),a
        ld      (led_phase+1),a
        ld      a,c
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
        ret
_af_20:
        ld      a,c
        and     11110000b
        ret     nz
        ld      a,(outstatus)
        and     11110000b
        or      c
        ld      (outstatus),a
_led_disp:
        ld      a,00000010b
        ld      (led_mode),a
        ld      (led_phase+0),a
        ld      a,00000001b
        ld      (led_ch),a
```


由良　泰人　京都出身の映像作家．ビデオ作品，インスタレーション作品などで国内・海外の多くの賞を受賞した新進気鋭の作家．1997年3月にも東京で個展を開いた．

〈リスト6-4〉AKI-80のアナログ←→MIDI変換器プログラム[IRCAM.SRC]④

```
        ld      (led_phase+1),a
        ld      a,(outstatus)
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
        ret
_af_21:
        ld      a,c
        cp      08h
        jr      nz,_21_not_8
        ld      b,080h
        jp      _21_mix
_21_not_8:
        cp      09h
        jr      nz,_21_not_9
        ld      b,090h
        jp      _21_mix
_21_not_9:
        cp      0ah
        jr      nz,_21_not_a
        ld      b,0a0h
        jp      _21_mix
_21_not_a:
        cp      0bh
        jr      nz,_21_not_b
        ld      b,0b0h
        jp      _21_mix
_21_not_b:
        cp      0eh
        jr      nz,_21_not_e
        ld      b,0e0h
        jp      _21_mix
_21_not_e:
        ret
_21_mix:
        ld      a,(outstatus)
        and     00001111b
        or      b
        ld      (outstatus),a
        jp      _led_disp

_af_22:
        ld      a,c
        ld      (threshold),a
        ld      a,00000010b
        ld      (led_mode),a
        ld      (led_phase+0),a
        ld      a,00010000b
        ld      (led_ch),a
        ld      (led_phase+1),a
        ld      a,c
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
        ret
_af_30_4f:
        ld      a,(keyno)
        sub     30h
        ld      e,a
        ld      hl,dac_status
        ld      d,0
        add     hl,de
        ld      b,c
        sla     c
        ld      (hl),c
        ld      a,00000100b
        ld      (led_mode),a
        ld      (led_phase+0),a
        ld      a,e
        ld      (led_ch),a
        ld      (led_phase+1),a
        ld      a,b
        ld      (led_data),a
        xor     a
        ld      (timer),a
        led_set
        led_out
        ret

        end
```

〈リスト6-5〉MIDIマージャのAKI-80プログラム[MERGER.SRC]①

```
;##### RAM Map #####
        dseg
        org     0000h
rx_fifo_0 ds    4096
rx_fifo_1 ds    4096
rx_fifo_2 ds    4096
rx_fifo_3 ds    4096
tx_fifo         ds          8192
rx_top_0        ds          2
rx_end_0        ds          2
rx_top_1        ds          2
rx_end_1        ds          2
rx_top_2        ds          2
rx_end_2        ds          2
rx_top_3        ds          2
rx_end_3        ds          2
tx_top          ds          2
tx_end          ds          2
rx_data         ds          4
rsb             ds          4
dcb             ds          4
channel         ds          4
keyno           ds          4

;##### I/O Map #####
        cseg
uart_0          equ         0080h
uart_1          equ         0082h
uart_2          equ         0084h
uart_3          equ         0086h

;##### MACRO #####
all_out macro   @1
        ld      a,@1
        out     (uart_0+1),a
        out     (uart_1+1),a
        out     (uart_2+1),a
        out     (uart_3+1),a
        endm
rx_set  macro   @1,@2,@3
        ld      de,(@3)
        ld      a,@1
        or      d
        ld      h,a
        ld      l,e
        in      a,(@2+0)
        ld      (hl),a
        inc     de
        res     4,d
        ld      (@3),de
        ret
        endm
rx_chk  macro   @1,@2,@3,@4
        ld      de,(@2)
        ld      hl,(@1)
        and     a           ; CY <-- 0
        sbc     hl,de
        ret     z
        ld      a,@3
        or      d
        ld      h,a
        ld      l,e
        ld      b,(hl)      ; [B] = Rx
Data
        inc     de
        res     4,d
        ld      (@2),de
        bit     7,b
        jr      z,50$       ; running
        ld      a,b
        cp      0f8h
        ret     nc
        cp      0f0h
        jr      c,10$
        xor     a
        ld      (rsb+@4),a
        ret
10$:
        ld      a,b
        and     00001111b
        ld      (channel+@4),a
        ld      a,b
        and     11110000b
        ld      (rsb+@4),a
        xor     a
        ld      (dcb+@4),a
        ret
50$:
        ld      a,(rsb+@4)
        cp      0
        ret     z
        cp      0c0h
        jr      z,70$
        cp      0d0h
        jr      z,70$
        ld      a,(dcb+@4)
        cp      0
        jr      nz,90$
        inc     a
        ld      (dcb+@4),a
        ld      a,b
        ld      (keyno+@4),a
        ret
70$:
        ld      c,b
        ld      a,(rsb+@4)
        ld      d,a
        ld      a,(channel+@4)
        or      d
        ld      b,a
        call    tx_fifo_set
        ld      b,c
        call    tx_fifo_set
        ret
90$:
        xor     a
        ld      (dcb+@4),a
        ld      c,b
        ld      a,(rsb+@4)
        ld      d,a
        ld      a,(channel+@4)
        or      d
        ld      b,a
        call    tx_fifo_set
        ld      a,(keyno+@4)
        ld      b,a
        call    tx_fifo_set
        ld      b,c
        call    tx_fifo_set
        ret
        endm

;##### RESET #####
        org     0000h
        ld      sp,0ffffh
        di
        im      1
        call    initialize
        ei
        jp      loop

;##### INT / NMI #####
        org     0038h
        ex      af,af'
        exx
        call    int_sequence
        exx
        ex      af,af'
        ei
        reti
        org     0066h
        ei
        retn
int_sequence:
        in      a,(uart_0+1)
        bit     1,a
        jr      z,_next_1
        rx_set  10000000b,uart_0,rx_top_0
_next_1:
        in      a,(uart_1+1)
        bit     1,a
        jr      z,_next_2
        rx_set  10010000b,uart_1,rx_top_1
_next_2:
        in      a,(uart_2+1)
        bit     1,a
        jr      z,_next_3
        rx_set  10100000b,uart_2,rx_top_2
_next_3:
        in      a,(uart_3+1)
        bit     1,a
        ret     z
        rx_set  10110000b,uart_3,rx_top_3

;##### MIDI Transmit Data Set : Input=[B] #####
tx_fifo_set:
        ld      de,(tx_top)
        ld      a,11000000b
        or      d
        ld      h,a
        ld      l,e
        ld      (hl),b
        inc     de
        res     5,d
        ld      (tx_top),de
        ret

;##### MIDI Transmit Check #####
tx_fifo_check:
        in      a,(uart_0+1)
        bit     0,a
        ret     z
        ld      de,(tx_end)
        ld      hl,(tx_top)
        and     a           ; CY <-- 0
        sbc     hl,de
        ret     z
        ld      a,11000000b
        or      d
        ld      h,a
        ld      l,e
        ld      a,(hl)
        out     (uart_0+0),a
        inc     de
        res     5,d
        ld      (tx_end),de
        ret

;##### MIDI Receive Check #####
rx_fifo_0_check:
        rx_chk  rx_top_0,rx_end_0,10000000b,0
rx_fifo_1_check:
        rx_chk  rx_top_1,rx_end_1,10010000b,1
```


コラボレーション作品　映像，造形，作曲，舞踏など，複数の異なる領域の芸術家が一緒になって統合的な芸術作品を作り上げていく，というコラボレーションは現代アートの一つの潮流である．

〈リスト6-5〉 MIDIマージャのAKI-80プログラム[MERGER.SRC]②

```
rx_fifo_2_check:
        rx_chk  rx_top_2,rx_end_2,10100000b,2
rx_fifo_3_check:
        rx_chk  rx_top_3,rx_end_3,10110000b,3

;##### Program #####
loop:
        call    tx_fifo_check
        call    rx_fifo_0_check
        call    tx_fifo_check
        call    rx_fifo_1_check
        call    tx_fifo_check
        call    rx_fifo_2_check
        call    tx_fifo_check
        call    rx_fifo_3_check
        jr      loop

;##### INITIALIZE #####
initialize:
        all_out 0
        all_out 0
        all_out 0
        ld      hl,0e000h
        ld      a,0e8h
_ram_clear_loop:
        ld      (hl),0
        inc     hl
        cp      h
        jr      nc,_ram_clear_loop
        all_out 01000000b
        all_out 01001110b
        all_out 00000101b
        ld      a,0fch
        out     (uart_0+0),a
        ret
```

〈リスト6-7〉 MIBURI-Sensor のAKI-80プログラム[MIBURI.SRC]

```
;##### RAM Map #####
        dseg
        org     0000h
rx_fifo         ds              2048
tx_fifo         ds              2048
rx_top          ds              2
rx_end          ds              2
tx_top          ds              2
tx_end          ds              2
rsb             ds              1
dcb             ds              1
channel         ds              1
keyno           ds              1
buff            ds              1
timer           ds              2
led             ds              1
ad              ds              1
ad_old          ds              6
ad_buff         ds              1
ad_timer  ds    1

;##### I/O Map #####
        cseg
sio_a           equ             0018h
sio_b           equ             001ah
pio_a           equ             001ch
pio_b           equ             001eh

;##### MACRO #####
io_set  macro   @1,@2
        ld      a,@2
        out     (@1+1),a
        endm
io_put  macro   @1,@2
        ld      a,@2
        out     (@1+0),a
        endm

;##### RESET #####
        org     0000h
        ld      sp,09fffh
        di
        jp      main

;##### INT / NMI #####
        org     0020h
        dw      _midi_
_midi_:
        ex      af,af'
        exx
        ld      de,(rx_top)
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        in      a,(sio_a+0)
        ld      (hl),a
        inc     de
        res     3,d
        ld      (rx_top),de
        exx
        ex      af,af'
        ei
        reti
        org     0066h
        retn

;##### Main #####
main:
        ld      hl,08000h
        ld      a,0a0h
_ram_clear_loop:
        ld      (hl),0
        inc     hl
        cp      h
        jr      nc,_ram_clear_loop
        io_set  pio_a,0cfh      ; Mode 3
        io_set  pio_a,11111111b ; 0:Out / 1:In
        io_set  pio_a,007h      ; Interrupt Disable
        io_set  pio_b,0cfh      ; Mode 3
        io_set  pio_b,00010000b ; 0:Out / 1:In
        io_set  pio_b,007h      ; Interrupt Disable
        io_set  sio_b,00011000b ; Channel Reset B
        io_set  sio_b,00000100b ; Resister Point = 4
        io_set  sio_b,11000100b ; Mode
        io_set  sio_b,00000001b ; Resister Point = 1
        io_set  sio_b,00000000b ; Interrupt Mode
        io_set  sio_b,00000010b ; Resister Point = 2
        io_set  sio_b,20h       ; Vector Address
        io_set  sio_a,00011000b ;Channel Reset A
        io_set  sio_a,00000100b ; Resister Point = 4
        io_set  sio_a,11000100b ; Mode
        io_set  sio_a,00000001b ; Resister Point = 1
        io_set  sio_a,00010000b ; Interrupt Mode
        io_set  sio_a,00000101b ; Resister Point = 5
        io_set  sio_a,01101000b ; Transmit Start

        io_set  sio_a,11000001b ; Receive Start
        im      2
        ei
        in      a,(sio_a+0)     ; dummy read
        ld      a,00000000b
        ld      (led),a
        out     (pio_b),a
        xor     a
        ld      (ad),a
        call    ad_start
loop:
        call    living_led
        cal     tx_data_check
        call    rx_data_check
        call    ad_check
        jr      loop

tx_data_check:
        ld      de,(tx_end)
        ld      hl,(tx_top)
        and     a               ; CY <-- 0
        sbc     hl,de
        ret     z
        io_set  sio_a,00000000b ; Resister Point = 0
        in      a,(sio_a+1)
        bit     2,a
        ret     z
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      a,(hl)
        out     (sio_a),a
        inc     de
        res     3,d
        ld      (tx_end),de
        ret
tx_fifo_set:
        ld      de,(tx_top)
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      (hl),b
        inc     de
        res     3,d
        ld      (tx_top),de
        ret
rx_data_check:
        ld      de,(rx_end)
        ld      hl,(rx_top)
        and     a; CY <-- 0
        sbc     hl,de
        ret     z
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        ld      b,(hl)
        inc     de
        res     3,d
        ld      (rx_end),de
        bit     7,b
        jr      z,50$; running
        ld      a,b
        cp      0f8h
        ret     nc
        cp      0f0h
        jr      c,10$
        xor     a
        ld      (rsb),a
        ret
10$:
        lda,b
        and     00001111b
        ld      (channel),a
        ld      a,b
        and     11110000b
        ld      (rsb),a
        xor     a
        ld      (dcb),a
        ret
50$:
        ld      a,(rsb)
        cp      0
        ret     z
        cp      0c0h
        ret     z
        cp      0d0h
        ret     z
        ld      a,(dcb)
        cp      0
        jr      nz,90$
        inc     a
        ld      (dcb),a
        ld      a,b
        ld      (keyno),a
        ret
90$:
        xor     a
        ld      (dcb),a
        ld      a,(rsb)
        cp      090h
        ret     nz
        ld      a,b
        ld      (buff),a
        ld      b,090h ; MIDI Channel = [1] fixed
!!!
        call    tx_fifo_set
        ld      a,(keyno)
        ld      b,a
        call    tx_fifo_set
        ld      a,(buff)
        ld      b,a
        call    tx_fifo_set
        ret
living_led:
        ld      a,(timer+0)
        inc     a
        ld      (timer+0),a
        cp      0
        ret     nz
        ld      a,(timer+1)
        inc     a
        and     01111111b
        ld      (timer+1),a
        cp      30
        jr      z,_led_on
        cp      45
        jr      z,_led_off
        ret
_led_on:
        ld      a,(led)
        and     00000001b
        or      00100000b
        ld      (led),a
        out     (pio_b),a
        ret
_led_off:
        ld      a,(led)
        and     00000001b
        and     11011111b
        ld      (led),a
        out     (pio_b),a
        ret
ad_check:
        ld      a,(ad_timer)
        inc     a
        ld      (ad_timer),a
        cp      50
        ret     c
        xor     a
        ld      (ad_timer),a
        in      a,(pio_b)
        bit     4,a
        ret     z
        in      a,(pio_a)
        srl     a
        ld      c,a
        ld      a,(ad)
        ld      l,a
        ld      h,0
        ld      de,ad_old
        add     hl,de
        ld      a,(hl)
        cp      c
        jr      z,_ad_next
        ld      a,c
        ld      (hl),a
        ld      (ad_buff),a
        ld      a,(ad)
        or      0d0h
        ld      b,a
        call    tx_fifo_set
        ld      a,(ad_buff)
        ld      b,a
        call    tx_fifo_set
_ad_next:
        ld      a,(ad)
        inc     a
        ld      (ad),a
        cp      6
        jr      nz,ad_start
        xor     a
        ld      (ad),a
ad_start:
        ld      a,(led)
        and     00100000b
        ld      c,a
        ld      a,(ad)
        or      c
        out     (pio_b),a
        or      00001000b
        out     (pio_b),a
        nop
        and     11110111b
        out     (pio_b),a
        ld      (led),a
        ret
```


BGV バック・グラウンド映像．BGMに対して，最近では環境映像をこう呼ぶ．

〈リスト6-8〉SNAKEMANセンサのAKI-80プログラム[SNAKE.SRC]

```
;##### RAM Map #####
            dseg
            org     0000h
tx_fifo     ds      256
tx_top      ds      1
tx_end      ds      1
status      ds      2
counter     ds      3

;##### I/O Map #####
            cseg
sio_a       equ     0018h
pio_a       equ     001ch
pio_b       equ     001eh

;##### MACRO #####
io_set      macro   @1,@2
            ld      a,@2
            out     (@1+1),a
            endm
io_put      macro   @1,@2
            ld      a,@2
            out     (@1+0),a
            endm

;##### RESET #####
            org     0000h
            ld      sp,09fffh
            di
            jp      main

;##### INT / NMI #####
            org     0066h
            retn

;##### Main #####
main:
            ld      hl,08000h
            ld      a,090h
_ram_clear_loop:
            ld      (hl),0
            inc     hl
            cp      h
            jr      nc,_ram_clear_loop
            io_set  pio_a,0cfh      ; Mode 3
            io_set  pio_a,00001010b ; 0:Out / 1:In
            io_set  pio_a,007h      ; Interrupt Disable
            io_set  pio_b,0cfh      ; Mode 3
            io_set  pio_b,11111110b ; 0:Out / 1:In
            io_set  pio_b,007h      ; Interrupt Disable
            io_set  sio_a,00011000b ; Channel Reset A
            io_set  sio_a,00000100b ; Resister Point = 4
            io_set  sio_a,11000100b ; Mode
            io_set  sio_a,00000001b ; Resister Point = 1
            io_set  sio_a,00000000b ; Interrupt Mode
            io_set  sio_a,00000101b ; Resister Point = 5
            io_set  sio_a,01101000b ; Transmit Start
            io_put  pio_b,1
            call    check
            ld      (status+1),a
loop:
            call    sw_check
            call    tx_data_check
            jr      loop

;##### Subroutines #####
tx_data_check:
            ld      a,(tx_end)
            ld      b,a
            ld      a,(tx_top)
            cp      b
            ret     z
            io_set  sio_a,00000000b ; Resister Point = 0
            in      a,(sio_a+1)
            bit     2,a
            ret     z
            ld      hl,tx_fifo
            ld      l,b
            ld      a,(hl)
            out     (sio_a),a
            ld      a,b
            inc     a
            ld      (tx_end),a
            ret
tx_data_set:
            ld      hl,tx_fifo
            ld      a,(tx_top)
            ld      l,a
            inc     a
            ld      (tx_top),a
            ld      (hl),b
            ret

;##### Sw Check #####
check:
            in      a,(pio_a)
            and     00000010b
            cp      0
            jr      z,_check_zero
            ld      a,1
            ret
_check_zero:
            xor     a
            ret
sw_check:
            call    check
            ld      (status+0),a
            ld      b,a
            ld      a,(status+1)
            cp      b
            jp      z,not_event
new_event:
            ld      a,b
            ld      (status+1),a
            cp      1
            jp      z,new_on
new_off:
            ld      a,(counter+2)
            cp      0
            jr      z,_count_1
            bit     6,a
            jr      z,_2_5
            ld      c,1
            jp      midi_tx
_2_5:
            bit     5,a
            jr      z,_2_4
            ld      c,2
            jp      midi_tx
_2_4:
            bit     4,a
            jr      z,_2_3
            ld      c,3
            jp      midi_tx
_2_3:
            bit     3,a
            jr      z,_2_2
            ld      c,4
            jp      midi_tx
_2_2:
            bit     2,a
            jr      z,_2_1
            ld      c,5
            jp      midi_tx
_2_1:
            bit     1,a
            jr      z,_2_0
            ld      c,6
            jp      midi_tx
_2_0:
            ld      c,7
            jp      midi_tx
_count_1:
            ld      a,(counter+1)
            cp      0
            jp      z,_count_0
            cp      120
            jr      c,_linear
            ld      c,8
            jp      midi_tx
_linear:
            xor     07fh
            ld      c,a
            jp      midi_tx
_count_0:
            ld      c,07fh
midi_tx:
            ld      b,0d6h
            call    tx_data_set
            ld      a,c
            ld      b,a
            call    tx_data_set
            ret
new_on:
            xor     a
            ld      (counter+0),a
            ld      (counter+1),a
            ld      (counter+2),a
            ret
not_event:
            ld      a,(status+1)
            cp      0
            ret     z
            ld      a,(counter+0)
            inc     a
            ld      (counter+0),a
            bit     7,a
            ret     z
            xor     a
            ld      (counter+0),a
            ld      a,(counter+1)
            inc     a
            ld      (counter+1),a
            bit     7,a
            ret     z
            xor     a
            ld      (counter+1),a
            ld      a,(counter+2)
            inc     a
            ld      (counter+2),a
            bit     7,a
            ret     z
            ld      a,01000000b
            ld      (counter+2),a
            ret
```


サブリミナル効果　1秒間に30コマのビデオ映像のうち，ある1コマにだけ特定の映像を混入しておくと，見ている人間はそれと気付かないのに，その映像の影響を心理的に受けてしまうというもの．

第7章　C言語によるAKI-80システムの開発

Mini-Cでソフトウェア開発する課程を詳述する

Mini-Cの動作を解析する

ここまでAKI-80を利用したシステムの実験と筆者の実例を紹介してきたことで，だいぶAKI-80とアセンブラが身近なものになったことでしょう．本書と同時進行で製作や実験を進めてきた方にとっては，アセンブラでのAKI-80システム開発はすでに手持ちの技術になっているかもしれません．

そこで，いよいよ本章では，このAKI-80システムをC言語で開発していくという「Javaへの道」の次なるステップへと進んでいきます．

本書の姿勢として，与えられたCコンパイラをブラックボックスとして使うだけでなく，その内部処理を解析し，コンパイラの環境自体を独自に拡張，改造，改良していくことになるので，Cの本質的動作の勉強としても収穫があるものと期待していきましょう．

さて，まずはともかく，XA80といっしょに2,500円で入手してきたMini-Cコンパイラを実際に使ってみましょう．

リスト7-1は，アセンブラのときに**リスト4-1**(p.35)として作ったものと同等の，一連の処理を確認するためのバッチ・プログラム(C.BAT)です．たった1行だけ増えている，2行目のところの「mc　-s」というのが，ソース・ファイル(拡張子は[.c])をコンパイルするためのMini-Cの**起動コマンド**です．

この結果はXA80のソースになるので，あとは以下同文ということです．

そして，Cでは誰でも最初に制作するという**「Hello World」プログラム**よりも簡単な実験ソース・プログラムが**リスト7-2**の[test0.c]です．

Mini-Cは**インライン・アセンブル機能**をもっているので，このように「#asm」と「#endasm」で囲まれた部分はそのままアセンブラ・ソースとして出力してくれます．つまり，このプログラムは「NOPが1発」という最高にシンプルなもので，ROMに焼いても暴走するだけのものです(NOPを実行したあとの処理が不定)．

さて，**リスト7-2**のプログラムでは暴走するはずなのですが，実際には暴走しません．これは，コンパイラがMini-Cでは**リスト7-3**のファイル名が[STARTUP.H]と予約された特殊なファイルを，自動的に読み込むためなのです．試しにこのファイル名を変更するとエラーとなります．これは**ROM化のためのスタートアップ・モジュール**と呼ばれるもので，Cプログラムのmain()として記述される「本体」を，リセットされたAKI-80がゼロ番地から正常にスタートしたあとで呼び出すという一連の流れを規定しているものです．

この中身を見てみると，なんのことはない，ゼロ番地からとりあえずスタートして，最初に

- ▶ 割り込みを禁止
- ▶ スタック・ポインタを設定
- ▶ 割り込みモードを設定

〈リスト7-1〉　解析実験用バッチ・プログラム C.BATの内容

```
mifes test.c                    ← 実験用ソース・ファイルの編集
mc -s test.c                    ← コンパイラ［MC］で処理
xa80 test,,,;                   ← アセンブラ［XA80］で処理
copy test.hex transfer.hex      ← 予約ファイル名にコピー
trans                           ← ターゲット・システムにロード
```

〈リスト7-2〉　サンプル・ソース test0.cの内容

```
main(){
#asm
        nop
#endasm
}
```

〈リスト7-3〉　自動的に組み込まれる STARTUP.H(ファイル名は予約)の内容

```
RAM_TOP EQU     08000h

        ORG     0000h
        jp      startup_start

        ORG     0066h
        nop
        retn

startup_start:
        di
        ld      sp,9fffh
        im      2
        call    __main
__st_loop__:
        DI
        nop
        jr      __st_loop__
```


起動コマンド　あるプログラムを起動するためのコマンド．そのプログラム自体の名前とともに，起動時のオプションまで添えて入力することが多い

〈リスト7-5〉 コンパイラMCの出力したアセンブラ・ソース test0.asmの内容

```
;System Load Compiler [mini-'C' V1.0] for Z80/KC80 1995.3
;By System Load CO.,LTD. & K.Kino Copyright (C) All rights
reserved

include         "startup.h" ← アセンブラ・ソースに自動組み込み

MC_GCHAR:                   ← このライブラリも自動組み込み
        LD      A,(HL)
MC_SXT:
        LD      L,A
        RLCA
        SBC     A,A
        LD      H,A
        RET

(中略，この部分は［IOLIBZ.LIB］と同じ)

DEBUGSP2:
        ADD     HL,SP
        PUSH    HL
        POP     IY
        RET

;main(){                    ← ここからCソースのコンパイル結果
__main:
;#asm                       ← 「#asm」の部分はそのまま出力

        nop

;#endasm                    ← 「#endasm」まで
;}
        RET

        END

;; --- End of Compilation ---
```

を行って，その後に「main()」という関数をコールして，ここから返ってくると「割り込み禁止の無限ループ」に入ったままという，全体の枠組みを提供しているだけなのです．すでにこのあたりの機構は本書をここまで読んできた皆さんなら楽勝でしょう．

そしてここからが面白いところなのですが，コンパイラMini-Cは，［STARTUP.H］というスタートアップ・モジュールとともに，**リスト7-4**(p.94)のような，これまた[IOLIBZ.LIB]という予約名のファイルを自動的に読み込みます．これはなんだか意味不明の，しかし完全にアセンブラばりばりのプログラムです．このリストを眺めて，その特徴として「レジスタ処理ばかりで，特定のI/Oアドレスの I/O処理どころか，特定のアドレスのメモリ・アクセスすら存在していない」という点に気付いた人は，なかなか鋭いところを突いています．

内部的にコールしたりすることはあっても，これらのすべてのサブルーチンは完全に閉じた世界を形成していて，コールする外部との接点はすべてレジスタ経由，それも正確に言えばスタック経由でだけ接しているのです．

これはC言語の内部構造を理解する上でもっとも重要なポイントです．C言語とは，レジスタ構成やメモリ構成など，CPUシステムのアーキテクチャに左右されない汎用のソフトウェアを実現するのですが，その秘訣は「パラメータはすべて**スタック**で受け渡しする」というところにあります．

たとえば，もっとも単純な制御文の
「for(...;...;...){}」
という場合，ループを回すカウンタをどのように内部処理するかと考えてみましょう．

これを特定のメモリに変数として，あるいは特定のレジスタに割り当ててというのでは，システムに依存していることになります．そこで，ループ・カウンタもスタックで渡すというのがCコンパイラの動作原理ということになります．**リスト7-4**の膨大な「MC_」で始まるサブルーチン群は，このような視点から用意された，「Cプログラムの基本構造を実現する道具」というべきものなのです．

さて，このような背景を受けて，実際にコンパイルされた結果を調べてみましょう．**リスト7-5**は，［test0.c］をMini-Cでコンパイルした結果の[test0.asm]というファイルです，**リスト7-4**に相当する部分はまったく同じなので省略していますが，スタートアップ・ファイルを自動的にインクルードしているのがわかります．そして最後の部分でCのソースを展開していますが，Cのソースそのものはコメントをつけて，main()という関数は__mainというラベルにして，インライン・アセンブラの部分をそのまま記述して，最後に自動的にリターンを添えています．

なんだか騙されたようですが，ここではこのように簡単なものです．

そして，これをアセンブラXA80が処理した結果を，**リスト7-6**(p.95)の［test0.lst］で調べてみましょう．今度はインクルードされたスタートアップ・ファイルもちゃんと展開され，絶対アドレスが添えられています．Cではmain()となっているメイン関数は，コンパイラで変換された__mainというラベルと符合しています．あとは，[iolibz.lib]に記述された標準サービス・ルーチン群が，すべてアセンブラ・プログラムとして内部に取り込まれているということがわかります．つまり，Cの中から使われているものも使われていないものも，このサイズの分量だけはかならずROMの中を占有するということになります．

最近の電化製品に内蔵されているワンチップ・マイコンは，たいていがCで開発されていますが，プロの使うCコンパイラは，Mini-Cよりもはるかに多量のサービス関数をこのように内包しているために，簡単な処理プログラムでも多量のROMを必要とすることになります．業界で「大容量ROM」タイプのワンチップCPUが必要とされているのはこのためです．

さて，Mini-Cの動作が少しわかってきたところで，それではもう少し現実的なCプログラムがどのようになっているかを調べてみることにしましょう．**リスト7-7**(p.86)は，Cプログラムのいろいろな「関数呼び出し」をテストしてみるためのサンプル[test1.c]です．外部とのやりとりが何もないプログラムですが，肝心なところをいくつか実験しています．

まず先にmain()を見てみると，変数としてaとbを定義し，とりあえずaに1を代入しています．アセンブラの感覚ならメモリ中に「a」という変数領域を定

Hello Worldプログラム　画面に「Hello World」と出力するという，もっとも簡単なプログラムのこと．C言語の教科書がここからスタートすることが有名．

〈リスト7-7〉 関数呼び出しテスト用のサンプル・ソース test1.cの内容

```
int sub_1(int a){
        int c;
        c = a * 2;              ←呼び出し元から引き数をもらう
      return(c);                ←リターンで値を返す
}

void sub_2(int b){
        return;                 ←ただ戻るだけのリターン
}

main(){
        int a,b;
        a = 1;
        b = sub_1(a);           ←引き数を与え、値を返させる
        sub_2(b);               ←引き数を与えてただ呼び出す
}
```

義して，ここに代入しているだけなのですが，**ローカル変数**には固定メモリをスタティックに割り当てないCコンパイラではどうなるでしょうか．そして次に，このaをsub_1という関数に与えて，その戻り値をbに代入しています．さらに，このbをsub_2という関数に与えて，そこで終了です．さて，このような動作をどのようにしてアセンブラで記述できるか，ちょっと考えてみましょう．

そして**リスト7-7**の前半にある(Mini-Cは後方参照できないので，あとで呼ばれる関数は先に記述しておかないといけない)，sub_1の関数では，ローカル変数cを定義して，呼び出し元から与えられた値を2倍してcに代入し，この値を返り値としてリターンしています．またsub_2 の関数では，何もせずそのままリターンしています．

さて，これらの関数はどのように記述されるのでしょうか．

リスト7-8(p.95)は，この単純なプログラム[test1.c]をMini-Cでコンパイルした結果のアセンブラ・ソース[test1.asm]の中から，これまでの説明と変わらない部分を除いた「該当部分」のリストです．これはある意味で本書のもっとも重要なポイントに関係していますから，なんとか頑張って追いかけてみる価値のあるリストといえます．

まず最初に，関数sub_1の冒頭の「int　c;」というローカル変数定義がコメント・アウトされていますから，これに続いた「push　BC」がcに該当していることになります．つまり，16ビット変数のcは，スタック領域に確保されたことになります．絶対的なアドレスでアクセスするのでなく，あくまでこの関数が呼ばれたときのスタック・ポインタを基準にしてアクセスするわけです．

そして，「c=c*2;」というCソースは，これに続く9行に変換されています．実際には9行の中にさらにサブルーチン・コールが3回もありますから，アセンブラならなんでもない「2倍する」という処理も，Cではとんでもなく長い道のりに旅立つのがよくわかります．

これ以降をスタック図を書きながら1ステップずつ追いかけてみるという作業は膨大な仕事になります．**リスト7-4**にすべての材料は提供されていますから，時間と興味のある方はぜひ調べてみてください．ただし，筆者のように根気のない人は，肝心な部分だけ押さえておけば，とりあえず先に進むことができます．それは「関数が呼ばれる際に与えられる値はスタック経由で扱う」「関数が呼び出し元に返す値はレジスタ・ペアに入れておく」「関数の最後にreturnは入れても入れなくてもよい(なければコンパイラが入れてくれる)」などという点です．

リスト7-8だけでも，本気でじっくり眺めると相当な時間を必要としますが，あくまで本書はコンパイラの本ではなくてAKI-80システムのための活用を目指していますから，さらに次に進みます．**リスト7-9**の[test2.c]は，広域変数と，char型の8ビット変数と，配列型の変数の扱いを調べるためのソースです．同じようにMini-Cでコンパイルした結果の**リスト7-10**[test2.asm](p.95)を見てみると，

- ▶ int型にくらべてchar型変数はかえって内部処理に手間がかかる
- ▶ 配列処理もかなり面倒なことをしている

ということがひしひしと伝わってきます．

ただし，ここではこのような内部的な展開を調べてみましたが，実際にはコンパイラ内部でどのように展開されているかをブラックボックスとして使うのですから，あまり気にしなくてもよいという部分もあります．アセンブラでは十分に速かったAKI-80ですが，このように展開された処理を行うとなると，ちょっとCプログラムの処理能力については心配になるというあたりの感想が健全な反応でしょう．

アセンブラ処理をCに置き換える

とりあえずMini-Cの動作がなんとなく見えてきたところで，実際にAKI-80のハードウェアをCで動かしてみたくなるのが人情でしょう．ここでは，最初からすべてをCにするのでなく，稼動実績のある部分はアセンブラで記述(インライン・アセンブラ機能を活用)して，制御系の高級言語で表現する部分をCでというハ

〈リスト7-9〉 変数定義テスト用のサンプル・ソース test2.cの内容

```
char a[10];                     ←char型配列を広域変数として定義
int b[10];                      ←int型配列も定義

main(){
        char c1, c2[5];         ← char型変数定義
    int d1, d2[5];              ←int型変数定義

        c1 = 1;
        d1 = 1;
        c2[0] = 1;
        d2[0] = 1;
}
```


インライン・アセンブル機能　C言語コンパイラの機能として，特定の予約語とともにある部分をアセンブリ言語で記述できるように対応していること．

イブリッドな作戦が有効です．そして，やがてはアセンブラ記述のC関数の中で標準的なものをまとめてオリジナルの「AKI-80用ライブラリ」としていくという作戦をとることにしましょう．

リスト7-11(p.96)は，第3章の**リスト3-2**(p.18)で紹介した「LEDを点灯表示」というプログラムを，そのままCに置き換えたソース[ctest.c]です．ここでは，なるべく**リスト3-2**と対比できるようにしてあるので，両者を比較しながら追いかけてみましょう．まず最初にI/Oポートのアドレスをdefine文で定義し，メモリ中に定義した変数は広域変数としてスタティック定義しています．また，LEDのセグメント情報を変換するテーブルは，**配列変数**として用意しました．

そして最初の部分のport_outとport_inという二つの関数は，今後何度となく登場する，とても重要なサンプルとなっています．まず，port_out()に与えられるportとdataという変数は，HLレジスタのオフセットを4と2にして，リストのようにスタックから取り出します．また，port_in()のように，int型の値を呼び出して元に返すには，HLレジスタに格納してリターンすればよいことがわかります．ある意味では，この部分のような「外部との接点の約束ごと」さえ使えれば，あとの拡張は同じパターンで拡げていけばいいのです．

リスト7-11のwait_shortでは，for文を回してnopを5回としていますが，これがどうアセンブラ展開されるかは注目に値します．あとのsw_scan()以降の記述は，ようやく見慣れたC言語らしいスタイルとなっていますから，ここまでくると安心する読者も少なくないでしょう．このように，道具となるアセンブラがらみの部分を実験しながらライブラリとして外部に出してしまって，開発するシステムに固有の機能を実現する部分だけは高級言語のCで記述していくというのが目標なのです．

リスト7-11のCソースをMini-Cでコンパイルした結果が**リスト7-12**(p.97)の[ctest.asm]です．ポート入出力の関数はそのままインライン・アセンブラ展開されているだけですから何ということもありません．それより驚かされるのが，wait_short()の展開結果です．軽い気持ちで書いたfor文がこのように長大に展開されています．

考えてみれば，ここではtimerとしているローカル変数をスタックに確保し，条件判定のたびにスタック経由で呼び出して比較し，さらに毎回，スタック経由での**インクリメント処理**を行っているのですから，これも当然ということなのです．

また，sw_scan()関数では，ポート入力の上3ビットを5ビット右シフトして，下3ビットだけマスクするという処理なのですが，アセンブラでの記述では想像できない展開となっています．そしてもっとも筆者が驚いたのが，led_dispの一連の関数です．dispという変数でテーブルを参照した結果をポート出力するだけと思っていたものが，ここまで難解に展開されることに感動すら覚えました．main()の中でも同じように感心したのですが，これを知ったからといってどうなるものではないにしても，なかなか得難い実験でした．

AKI-80用スタートアップ・モジュールとライブラリの開発

さて，このような準備を経て，いよいよAKI-80システムとMini-Cのための「オリジナルのソフトウェア開発環境」を整備していくことにしましょう．実はこの部分は，筆者もまだ現在進行形でいろいろと実験しながら拡張している最中なので，おそらく本書が読者の皆さんの手元にある頃には，筆者の環境はさらに進展しているかもしれません．

しかし，ソフトウェア開発のためのライブラリなどというものは，固定せずに日々進化していくところに価値があるのですから，皆さんもどんどんオリジナリティを発揮して欲しいと思います(いずれNifty-ServeのMIDIフォーラムか筆者のホームページで，最新のライブラリ集をフリーウェアとして公開しているかもしれません)．

筆者のライブラリ開発は，製品として「AKI-80汎用ライブラリ集」を発売するというものではありませんから，必要な処理が出てくるとその中でシステムに固有でない汎用部分を次第にライブラリに加えるという開発手法になります．そこで，ここで紹介するまでに膨れてきた経緯や，今後さらに膨れていくであろう汎用の各種スペックについての議論はありますが，「現在の時点での状況」を紹介するというスタイルにならざるをえません．具体的には，後述するLCDパネルへの出力とMIDI処理モジュールが筆者にとって当面の汎用機能なので，この部分をライブラリ化した実例ということとなります．

リスト7-13(p.98)は，スタートアップ・ファイル[STARTUP.H]の最新バージョンです．**システム予約名ファイル**といっても，中身は自由に変更してかまわないのです．ここではまず，冒頭にAKI-80の周辺I/Oポートのアドレスをequ文で定義しています．これらのアドレスは，アセンブラ・ソースとしてインクルードされるライブラリ・モジュール中ばかりでなく，Cプログラム中でインライン・アセンブラ展開される際に使用されるラベルに対しても必要となりますから，当面使っていないI/Oポートでも定義しておいたほうがよいでしょう．

これに続いて，8000h番地以降のメモリ空間に，汎

ROM化のためのスタートアップ・モジュール　C言語のプログラムをROM化して固定するためには，普通はRAM領域にある予約部分までをROMに置くことが必要であり，そのための処理を記述しているソフトウェア・モジュールのこと．

用として予約する各種の変数エリアのラベルを定義しています．なお，1バイト・サイズの各種変数はどこに置かれてもかまわないのですが，MIDIの送受信FIFOバッファとして定義している先頭の部分(LCD表示にもFIFOを使っています)については指定席です．これは，割り込み処理などでのアクセスを最短時間で処理するために，アドレスの上位部分を固定したかなりテクニカルな処理を行っているためで，ほかの場所にこのまま移すことはできません．

I/Oとメモリのアドレス定義部分に続いて，**リスト7-13**は0000h番地からいよいよプログラム本体に入ります．このサンプルでは，もっとも廉価版の「AKI-80シルバーキット」を対象としているので，RAMは8Kバイトしかありませんから，スタック・ポインタは9fffhまでしか使えません．スタック・ポインタ設定，割り込み不許可というお決まりの処理に続いて，RAMエリアをすべてゼロ・クリアしています．筆者の経験では，何かのバグで起動時に問題がある場合，この処理によってとりあえずRAMの初期値が常にゼロであるか，毎回が不定状態であるかというのは「**原因究明の際の条件統一**」という意味でとても重要なのです．

メモリのクリアの後にあるのは，割り込みモードを2にする(内部I/Oポートを活用する)という設定(まだ周辺ポートを初期化していないので，割り込みは許可されていないことに注意)があるだけで，すぐにmain()に飛び込みます．このような初期設定は，大きくシステムが変わらなければそのまま使っていけるので，速攻で新しいシステムを構築する必要のあるときにはとても重宝します．

そして，[STARTUP.H]はまだこれでは終わりません．0020hの固定アドレス以下の部分には，本書ですでに述べたようなMIDI割り込み処理ルーチンが置かれています．当然，ここのアドレスを割り込みベクトルとするような設定がなければ，この部分は利用されません．

また，0070hにはAKI-80の内部タイマ割り込みを利用した基準時間処理のためのタイマ割り込み処理ルーチンがあります．これらの定型処理は，デフォルトで組み込まれる[IOLIBZ.LIB]とともに，ROMエリアの先頭にいつも置かれることになります．

さて，**リスト7-13**のスタートアップ・モジュールとペアになっているのが，**リスト7-14**(p.99)のAKI-80用オリジナル・ライブラリ[library.c]です．これはCのプログラム本体からインクルードされて，あくまでCプログラムとして解釈されるので，[STARTUP.H]でのコメント記号の「;」でなく，コメントは「/*.....*/」でなければなりません．なお，Mini-CではJavaと同じように，行の先頭に「//」を置いた場合にも，それ以降の行末までをコメントとして解釈してくれます．

リスト7-14の冒頭では，今度はCプログラムの#define文として，**リスト7-13**とほぼ同等のメモリ中の変数の宣言を行っています．これは，新しい定義を増やすと両方に加えなければならないという，ちょっと美しくない点で気になっているのですが，一方はアセンブラでもう一方はCコンパイラを相手にしていますから，面倒ですがきちんと対応させて記述する必要があるようです．

Cプログラム部分の先頭には，筆者のシステムではほぼ欠かさず使用するMIDI関連などの変数がスタティックに定義されています．実はどうも，Mini-Cの挙動からすると「すべての変数をスタティックに定義して，スタック処理の重そうなローカル変数はなるべく使わない」という方針がよさそうなのですが，ここは悩むところです．Cのプログラミング・マナーからすれば，「不必要に変数をスタティック宣言するのは美しくない」という美学もあるからです．まぁ，ここは読者の皆さんもそれぞれ実験的にすり合わせてみてください．

[library.c]の続く部分からは，いよいよ核心の「**周辺I/Oの処理**」がズラリと並んでいます．まずはポート・アドレスとデータを与えるポート書き込み，次に指定ポートからのポート読み出し，メモリ中の指定アドレスへの書き込みと読み出し，割り込みの許可と禁止，さらに(処理は遅そうですが)16進データから16進文字への変換などの汎用ルーチンが続きます．

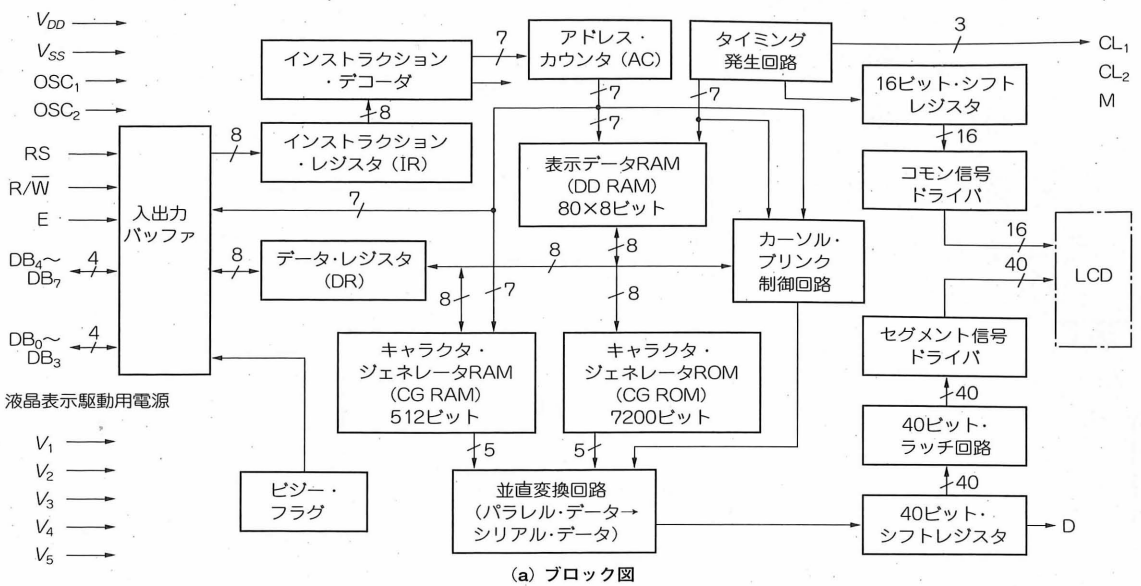
これに続く「I/O初期化ルーチン群」では，AKI-80のポートを簡単に初期設定するためのサービスが並びます．これらは何度も呼ばれることはなく，基本的に1回だけ使用されるものです．また，使わないポートに関する部分は完全にメモリ中にむだに存在するだけなのですが，このようなサービスをいったん定義しておくと，あとあと新しいシステムを開発するたびに「東芝CPUマニュアル」を開かなくてもよくなります．

PIOのセッティング関数では，変数「bitmap」はそれぞれのビットごとの入出力を設定します．0なら出力，1なら入力です．ポートを双方向に使う場合には，ここを頻繁に切り替える必要があります．PIOに続いてタイマ1の設定(割り込みベクトルの設定を含む)，そしてオマジナイのようなMIDIポート(SIO-A)の設定が続きます．とりあえずAKI-80で水晶を8MHzとした場合，これで何も考えずに最初からキツいMIDI割り込み受信でもびしばし行けるというのは，MIDIマシンを製作した人でないと，その有難味が理解できないかもしれません．MIDIというのは，実はなかなか最初から簡単には動いてくれないものなのです．

ポート設定に続いて，[library.c]では一連の

スタック　先入れ後出し(FILO)タイプのバッファ・メモリで，ソフトウェアによって実現する．ある変数をいったん格納しておいて，あとでふたたび利用するために使われる．サブルーチンのリターン・アドレスはスタックの典型である．

〈図7-1〉 LCDコントローラHD44780



（a）ブロック図

インストラクション	コード										機　能	実行時間（max）
	RS	R/$\overline{W}$	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0		
(1) 表示クリア	0	0	0	0	0	0	0	0	0	1	全表示クリア後，カーソルをホーム位置（0番地）へ戻す.	1.52ms
(2) カーソル・ホーム	0	0	0	0	0	0	0	0	1	*	カーソルをホーム位置へ戻す．シフトしていた表示ももとへ戻る．DO RAMの内容は変化しない.	1.52ms
(3) エントリー・モード・セット	0	0	0	0	0	0	0	1	I/D	S	データの書き込みおよび読み出し時に，カーソルの進む方向，表示をシフトさせるどうかの設定を行う.	37μs
(4) 表示ON/OFFコントロール	0	0	0	0	0	0	1	D	C	B	全表示のON/OFF (D)．カーソルのON/OFF (C)．カーソル位置にある桁のブリンク (B) をセットする.	37μs
(5) カーソル・表示シフト	0	0	0	0	0	1	S/C	R/L	*	*	DD RAMの内容を変えずに，カーソルの移動と，表示シフトを行う.	37μs
(6) ファンクション・セット	0	0	0	0	1	DL	N	F	*	*	インターフェース・データ長 (DL)．デューティ (N) および文字フォントを設定する.	37μs
(7) CG RAMアドレス・セット	0	0	0	1	A_{CC}						CG RAMのアドレスをセットする．以後受信するデータはCG RAMのデータ.	37μs
(8) DO RAMアドレス・セット	0	0	1	A_{DD}							DD RAMのアドレスをセットする．以後受信するデータはDD RAMのデータ.	37μs
(9) BF/アドレス読み出し	0	1	BF	AC							モジュールが内部動作中であることを示すBFおよびACの内容を読み出す．CG RAM, DD RAMの両方に使う.	6μs
(10) CG RAM, DO RAMへのデータ書き込み	1	0	書き込みデータ								DD RAMまたはCG RAMにデータを書き込む.	37μs t_{ADD}=5.6μs
(11) CG RAM, DO RAMからのデータ読み出し	1	1	読み出しデータ								DD RAMまたはCG RAMからデータを読み出す.	37μs t_{ADD}=5.6μs

＊：無効のビット
A_{CG}：CG ROMのアドレス
A_{DD}：DD RAMのアドレス
AC：アドレス・カウンタ

I/D=1：インクリメント
I/D=0：デクリメント

S=1：表示をシフトさせる
S=0：表示をシフトさせない

D=1：表示ON
D=1：表示OFF

C=1：カーソルON
C=0：カーソルOFF

B=1：ブリンクON
B=0：ブリンクOFF

S/C=1：表示シフト
S/C=0：カーソル移動

R/L=1：右シフト
R/L=0：左シフト

DL=1：8ビット
DL=0：4ビット

N=1：1/16デューティ
N=0：1/8または1/1デューティ

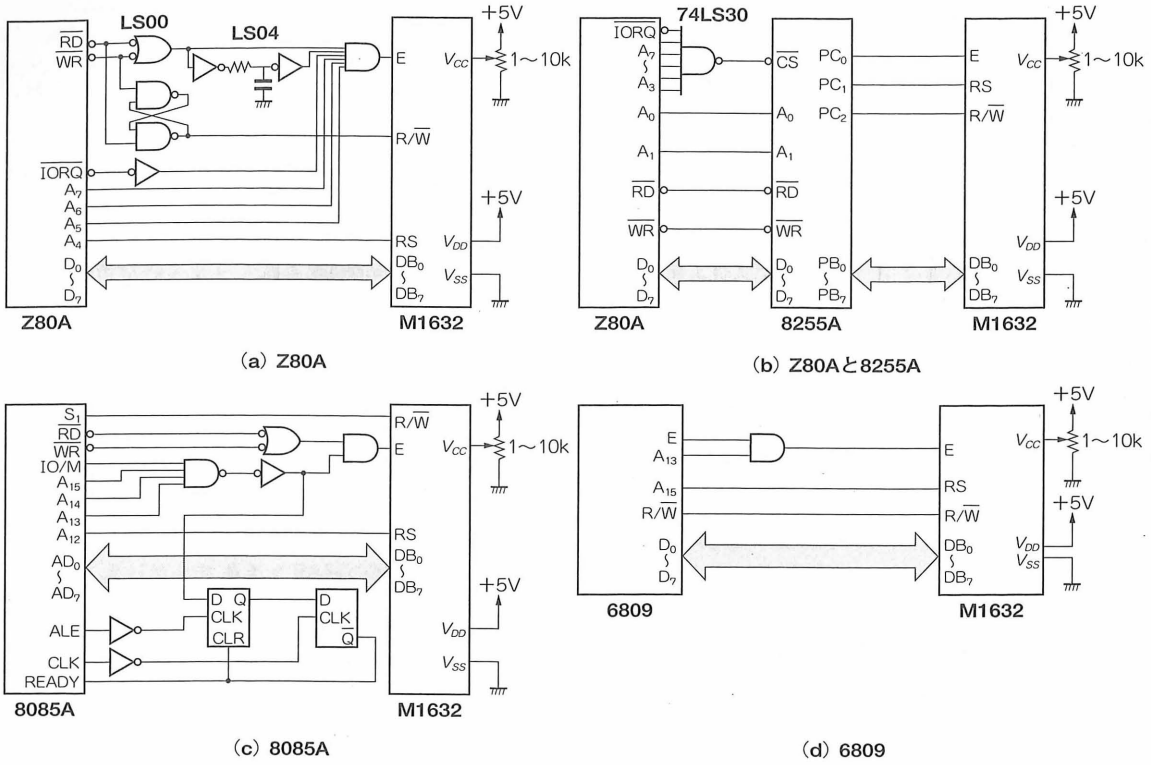
F=1：5×10ドット・マトリクス
F=0：5×7ドット・マトリクス

BF=1：内部動作中
BF=2：インストラクション受付可

（b）インストラクション一覧

ローカル変数　局所的に変数として定義して利用し，そのルーチンから戻る時点でその値を保存することなく破棄してしまう変数のこと.

〈図7-2〉 HD44780と各種CPUとの接続(M1632は，HD44780内蔵のLCDユニット)



MIDI処理サービス関数がずらりと続きます．これはいわば筆者の経験とノウハウの塊です．tx_midi_check()，tx_midi_set()，rx_midi_check()の三つの関数とMIDI割り込みルーチンの四つは，「内部はともかく確実にMIDI処理をしてくれるブラックボックス」として利用できるようになっています．

そしてこの後には，筆者がこれも秋月電子で1,200円で仕入れてきた「高コントラスト16文字2行LCDユニット」を「標準出力コンソール」として活用するための，各種のサービス関数が置かれています．この部分については，次項でハードをからめて詳しく検討します．

標準出力：LCDユニットの利用

さて，ここまでいろいろなAKI-80システムを紹介し，アセンブラからCコンパイラへと開発環境を進めてきましたが，まだ残念ながら，世界中のCプログラマが最初に書くという「Hello Worldプログラム」にまで到達していません．というのも，Cでは標準とされているテキスト・ベースの「標準入出力」がないからです．これではとてもJavaにも進めないので，ここでAKI-80らしい標準出力コンソールを増設することにしましょう(なお，同時に筆者のこだわりとして，Cによる標準MIDI入出力も押さえていきます)．

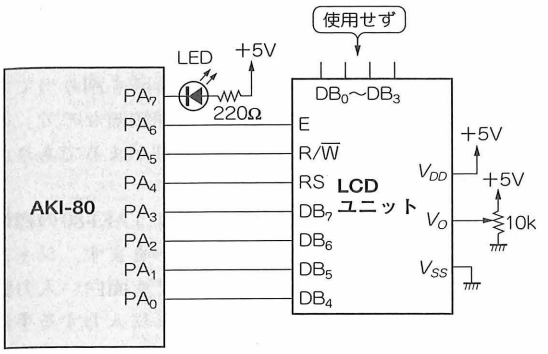
筆者が目をつけたのは，秋月電子のLCDユニットですが，実はこれはLCDコントローラが業界標準の日立のLSI(**HD44780**)なので，同等のモジュールであっても完全に互換です．秋月のものはセイコーの濃紺のハイ・コントラスト・タイプであるというだけのことです．たった32文字のせつないほど単純なディスプレイですが，それでも初めて文字が表示され，スクロール動作によって目を引くようにアピールしているAKI-80の姿はなかなか感動的なものでした．

図7-1はLCDコントローラLSIのブロック図とインストラクション表です．初めて見る人にとっては「異様でわかりにくい」という印象があるかもしれませんが，これは製品にLCDモジュールを組み込んで開発したことのあるプロには「毎度お馴染みの無愛想な」定番(筆者の私見では，最初に出てきた日立のマニュアルからずっと各社がこのスタイルを継承している)ので，ある程度は慣れてしまわないといけません．初期設定とか，アドレスを指定して文字データ(ASCIIコードそのもの)を書き込むという一連の手順が確立してしまえば，簡単にいろいろな応用ができます．

図7-2のように，このLCDユニットは基本的にどんなCPUにも接続でき，そのCPUへのインターフェー

配列変数　単独の変数でなく，その変数名に「添字」という変数を添えて，同じ変数名で複数の要素をまとめて扱えるように宣言した変数のこと．

〈図7-3〉 AKI-80とLCDユニットの接続



ス方法も，バス接続でもポート接続でもかまいません．また，CPUによるステータス管理も，割り込みでもポーリングでも監視省略でもOK，さらに，バス幅も8ビットでも4ビット2回でもOKという，なかなか寛容な体制になっています．

リセット直後のモード設定以降は，LCD全域描画の処理にはかなりの時間がかかるものの，個別の文字書き込みであればほぼ瞬時に行ってくれるという単純さが売り物の表示機器というわけです．

ここで筆者が悩んだのが，AKI-80との接続と処理スピードについてです．もっとも表示スピードを上げる方法は，AKI-80のCPUバスに直結して，ソフトの側ではLCDユニットの**ステータス・ビット**を常に監視して，「OKになったらすかさず表示文字を書き込む」ということに決まっています．しかし，これまでの経緯からわかるとおり，わざわざAKI-80からCPUバスを引き出したくありません．

また，8ビット・バスにすると，PIO-Aをバスにしたとしてもから何ビットか必要になり，他への拡張ができません．

そこで今回は，**図7-3**のように「PIO-Aだけを使って，4ビット・モードでCPUとインターフェースし，ポーリングすら行わない」という，およそ考えつく最低スピードのLCD方式を採用しました．これは，システムのプライオリティはあくまでMIDIにあること，もともとLCDの表示はかなり遅いこと，人間が文字を読み取るスピードならこれでも十分に速いことなどの経験則によります．

AKI-80のポートBは各種の応用のために空けておき，LCDユニットのデータ・バスの上4ビットだけを用いてAKI-80のPIO-Aの下4ビットと接続します．そして，8ビット形式のコマンドやデータを2回に分けて転送し，さらにレジスタ・セレクト(RS)信号と，基準パルスのE信号やライト信号までPIO-Aの上3ビットでソフト的にいちいち上げ下げしてやるのですから，ほとんど究極の遅さとなります．

このような背景で，もう一度**リスト7-14**のなかほどからのLCD処理ライブラリ関数を眺めてみましょう．コメントとして記述しているように，LCDモジュールは「RS」「E」「R/$\overline{W}$」の信号でタイミングを作っています．本書の読者でももう知らない人の方が多いと思いますが，これはもともとはモトローラ系の8ビットCPUの6800とか6809のバス・タイミングからきています．

このLCDコントローラをオリジナル開発した日立はその昔，モトローラ系CPUのセカンド・ソースを作っていたのです．そのつもりで**図7-2**をもう一度眺めてみると，確かに6809がもっとも単純にインターフェースできます．Z80でバス接続するためには，図のようになんとアナログの時定数によって遅延させたパルスを作ってやるか，別に74シリーズなどのICを使って，システム・クロックを利用してE信号を生成しなければならないのです．

リスト7-14にはいきなり「37μsのNOP」とか「500μsのウェイト」などという強烈なものがありますが，これは最終的には使っていません．LCDユニットの状態を監視しない場合，このようにダミーの時間を置いたのですが，あまりに全体の処理を遅らせることが判明したために，結局は取りやめてしまったものです．

このあとの`lcd_command_set()`という関数(実体はすべてインライン・アセンブラ)が，LCDユニットとのインターフェースの根幹部分です．中身としては，LCDユニットのマニュアルにしたがって，8ビット幅のコマンドを上下4ビットずつに分けて，さらに「E」信号に相当するPIO-Aのビット6をせわしく上げ下げしているだけです．あとで出てくる「文字データの書き込み」もまったく同様です．

なお，このあとの`lcd_module_setting()`は初期設定部分で，ここはとりあえずオマジナイとしておけば十分でしょう．8ビット幅で高速にインターフェースしたい場合には，ここを慎重に変更する必要があります．

`lcd_display()`という関数まで上位概念になれば，もうCプログラムからの制御は楽勝となります．ここではポート書き込みのスタイルと完全に合わせて，LCDユニット内のアドレスと文字データのセットを添えた関数呼び出しに対応しています．なお，このLCDユニットのコントローラLSIは，もともと40文字2列のタイプまで制御できるもので，1列目については文字のアドレスそのものでよいのですが，2列目の最初の文字のアドレスは**オフセット40h**を加える必要があります．

このような環境で，当初は「LCDが次の命令を受け入れられるまでむだに待つ」というシステムにしたのですが，「多量のMIDI情報を受信しながらLCD表示し

インクリメント処理　increment，すなわち「プラス1」という処理．加算で「1を足す」というよりも，コンピュータではincrementの方が処理が速い．

つつMIDI送信する」というシステムで実験してみたところ，かなりの遅れが問題となってしまいました．

そこでリストの例では，LCD表示の部分にもFIFOを設定して，表示すべきアドレスと文字データのペアをバッファリングしています．詳しい説明にはかなりの誌面を必要とするのでここでは省略しますが，興味のある人はリストの解読に挑戦してみてください．

実際にLCDユニットから情報を表示するためのもっともシンプルなC言語サンプルは，**リスト7-15** [HELLO.C] の`line_disp()`という関数にあります．

これは「Hello World」という文字列を，LCDユニットの下段でスクロールさせ続けるという単純なもので，中身も馬鹿正直に1文字ずつずらした文字列を次々に1行マルマル転送しています．

もっと内部アドレスを操作したりするテクニカルな方法はいくつもあるのですが，筆者はそのようなプログラミングに凝っている暇がないので，ディスプレイに関しては「表示したい情報をそのまま出す」ということに徹しています．

また，具体的な表示サービス・ルーチンは，指定位置から文字列を表示する`disp_string()`という関数，指定位置から16進数値を表示する`disp_hex()`という関数を用意しているわけですが，これも**リスト7-14**を追いかけると容易に理解できるでしょう．すぐにLCDに書かずにFIFOに積んで，`main()`の中で無限ループから周期的に呼び出される`lcd_disp_check()`という関数の中で，LCD表示情報のFIFOをチェックしているのですが，これが結果として，LCDユニットが次のデータを受け取れるためのウェイトの確保になっているのです．

標準入力の検討

さて，LCDユニットによる標準出力コンソールに続いて「標準入力」をという流れになるのですが，筆者は今のところ作っていません．これは，筆者の用途はおもに音楽なので，楽器の鍵盤というキーボードは使っても，コンピュータのキーボードのような入力は必要ないというのが最大の理由です．そして，秋葉原や日本橋を捜しても，中古パソコンのキーボードは入手できるものの，なかなか汎用性がないため，「必要に応じてどんどん作れる安価な標準入力装置」が見あたらないという理由もあります．

AKI-80はシリアル・ポートが2ポートありますから，筆者は手元のノート・パソコンからRS-232-CやMIDIで文字データを送って標準入力にすることも可能です．そして何より，どうしてもパソコンのキーボードから情報が欲しい場合には，前述のMacの環境“MAX”が，「キーボード入力」というオブジェクトをサポートしているので，そのままMIDI環境で何の苦もなく文字キーボード情報を使えてしまうのです．つまり，パソコンのキーボードごとに音階を割り当てて電子楽器で演奏するなどというのは朝飯前なので，ここにAKI-80のハードをもち込む必要性がまるでありません．

このような状況ですから，本書では「AKI-80の標準入力」については触れないで進めていきます．ジャンク屋のテンキー(古い電卓の部品)なども面白い入力装置ですが，AKI-80のパラレル・ポートに入力する手法はすでに述べていますから，手に入った入力装置を活用して，オリジナルの標準入力環境の構築にチャレンジしてみてください．

CによるMIDI処理の実現

C言語によるAKI-80システム開発の最後のテーマは，筆者の日常的なインターフェースであるMIDIです．すでにアセンブラの実例はさんざん検討してきましたから，ここでは簡単にその「差分」をまとめてみましょう．まず確認として，ハードウェアについては，**図5-12**(p.53)の「AKI-80標準MIDI回路」で，まったくアセンブラの場合と変わりはありません．このハードウェアはLCD表示とは大違いで，もっともスピードを重視したソフトウェアに対応できるものです．

アセンブラからCに進展するためにもっともポイントとなるのは，本章で検討した「Cのまわりくどさでしょう．MIDIの受信処理は，基本的に割り込みでFIFOに積んでいればデータ落としはありませんが，メイン処理でこのFIFOからデータを読み出す部分があまりに遅ければ，次第にFIFOがあふれてきて，最後にはデータが上書きされて無意味になってしまいます．

MIDI情報というのは，音楽の性質から粗密の差が大きいというのがFIFO利用の理由だったのですから，時間とともにFIFOにデータが溜まっていくようなソフトでは基本的に失格なのです．

リスト7-15では，

- MIDI受信データを割り込みルーチン内でFIFOに積む
- メイン・ルーチンでこのFIFOを読み出す`rx_midi_-check()`を呼ぶ
- MIDIイベントがあれば，`tx_midi_set()`でMIDI送信のFIFOに積む
- これに続いて，MIDIデータをLCDユニットに16進表示する
- メイン・ルーチンでは送信FIFOを読み出す

MIDI処理モジュール　MIDI処理に関するサブルーチンやサービス・ルーチンを集めた一群のソフトウエア部品集．

〈リスト7-15〉 Hello Worldプログラム[HELLO.C]

```
#include        "library.c"             /* Common Tools for AKI-80 */

initial(){
        lcd_module_setting();   /* using Port[A] */
        timer_0_setting();      /* about 10msec */
        midi_port_setting();    /* sio_a */
        enable_interrupt();     /* EI */
        sio_dummy_read();       /* omajinai */
}

line_disp( int tt ){
        int t;
        t = 14-(tt%15);
        switch(t){
                case  0: disp_string( 0x40, "Hello World    H"); break;
                case  1: disp_string( 0x40, " Hello World    "); break;
                case  2: disp_string( 0x40, "  Hello World   "); break;
                case  3: disp_string( 0x40, "   Hello World  "); break;
                case  4: disp_string( 0x40, "    Hello World "); break;
                case  5: disp_string( 0x40, "d    Hello World"); break;
                case  6: disp_string( 0x40, "ld    Hello Worl"); break;
                case  7: disp_string( 0x40, "rld    Hello Wor"); break;
                case  8: disp_string( 0x40, "orld    Hello Wo"); break;
                case  9: disp_string( 0x40, "World    Hello W"); break;
                case 10: disp_string( 0x40, " World    Hello "); break;
                case 11: disp_string( 0x40, "o World    Hello"); break;
                case 12: disp_string( 0x40, "lo World    Hell"); break;
                case 13: disp_string( 0x40, "llo World    Hel"); break;
                case 14: disp_string( 0x40, "ello World    He"); break;
        }
}

midi_display(){
        if( midi > 127 ){
                if( midi > 0xef ) rsb = 0;
                else{
                        rsb = midi & 0xf0;
                        channel = midi & 0x0f;
                        dcb = 0;
                }
        }
        else{
                if( rsb == 0 ){}
                else if( (rsb == 0xc0) || (rsb == 0xc0) ){
                        disp_hex( 7, rsb+channel );
                        disp_hex( 10, midi );
                        disp_string( 13, "  ");
                }
                else{
                        if( dcb == 0 ){
                                keyno = midi;
                                dcb = 1;
                        }
                        else{
                                dcb = 0;
                                if( (rsb==0x90) && (midi!=0) ){
                                        disp_hex( 7, rsb+channel );
                                        disp_hex( 10, keyno );
                                        disp_hex( 13, midi );
                                }
                        }
                }
        }
}

main(){
        int timer,t;
        t = 0;
        initial();
        disp_string( 0, "MIDI =          ");
        while(1){
                tx_midi_check();
                lcd_disp_check();
                midi = rx_midi_check();
                if( midi < 256 ){
                        tx_midi_set( midi );
                        midi_display( midi );
                }
                if( ram_get(timer_flag) != 0){
                        ram_put( timer_flag, 0 );
                        timer++;
                        if( timer > 50 ){
                                timer = 0;
                                line_disp(t++);
                        }
                }
        }
}
```

tx_midi_check()も呼ぶ

▶ 送信データがあれば，MIDI送信ポートのOKを確認してMIDI送信

という処理を行っています．つまり，ソフトウェア的にMIDI情報をスルーさせつつ，これをLCDで16進表示しているというものです．パソコンでこのような処理を行うプログラムを書くのも意外と難しいものなのですが，興味のある方は挑戦してみてください．下手に書くと，ちょっとMIDIトラフィックが増えただけでハングアップしたりします．

リスト7-14のオリジナル・ライブラリ内では，基本的にこのようなMIDI処理はほとんどすべてをインライン・アセンブラで記述しています．Cの本体とのインターフェース部分にだけ，最低限のスタック渡しの処理を入れています．ここは，これ以上に高速化するのは無理ですから，ほぼ汎用モジュールとして利用していけるものです．

実際に**リスト7-15**のプログラムはきちんと動作してくれます．LCD表示ルーチンをFIFO構成にしたために，ほとんどMIDI処理の遅れを感じさせないレベルにあります．最初のバージョンで「LCDに文字を書き込むたびに，そこでNOPの嵐でウェイトする」という方法をとったところ，これはもうMIDI情報がベタベタに遅れて，音楽の形が完全に崩壊しました．

しかし，まだ**リスト7-14**と**リスト7-15**の組み合わせは発展途上段階にあります．それは，Cでmidi_display()という関数を記述しているところが問題なのです．ここは，FIFOからイベントがあったときに取り出した生データを，MIDIプロトコルにしたがって判定，解釈する部分です．中身としては，多くのif文による比較判定が行われるのですが，これはCソースのシンプルな外見とまったく違って，コンパイルされたアセンブラ・ソースは驚くほど長大な処理になってしまいます．普通に電子ピアノを弾いたMIDI情報を処理するという程度であればまったく問題ないのですが，多量のセンサ情報をMIDI経由で処理したいというようなシステムには，ちょっと恐くて使えないように思います．

筆者の構想としては，完全に一つのサービス・ルーチンで汎用とするのでなく，代表的なアプリケーションに応じた複数のMIDI判定処理モジュールを標準ライブラリに置いて，Cの本体ではこの判定処理を行わずにインライン・アセンブラ処理に任せるというのが現実的な手法であると思っています．ただ，机上のシステムとして架空のライブラリを作るのはあまりにも不毛ですから，次の機会に新しいシステムを開発するときに，今度はCを使ってアプローチしてみて，その中でライブラリをさらに充実させていくということになりそうです．

システム予約名ファイル　コンパイラやアセンブラのシステムとして，特定の名前のファイルについてはその役割が予約されている，そのファイル．

〈リスト7-4〉 自動的に組み込まれる IOLIBZ.LIB(ファイル名は予約)の内容①

```
MC_GCHAR:
        LD      A,(HL)
MC_SXT:
        LD      L,A
        RLCA
        SBC     A,A
        LD      H,A
        RET
MC_CDR:
        INC     HL
        INC     HL
MC_GINT:
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        RET
MC_PINT:
        POP     BC
        POP     DE
        PUSH    BC
        LD      A,L
        LD      (DE),A
        INC     DE
        LD      A,H
        LD      (DE),A
        RET
MC_ORAND:
        POP     BC
        POP     DE
        PUSH    BC
        LD      A,L
        OR      H
        JP      NZ,__OR2R
        LD      A,E
        OR      D
        JP      NZ,__OR2R
        LD      HL,0
        RET
__OR2R:
        LD      HL,1
        RET
MC_OR:
        POP     BC
        POP     DE
        PUSH    BC
        LD      A,L
        OR      E
        LD      L,A
        LD      A,H
        OR      D
        LD      H,A
        RET
MC_XOR:
        POP     BC
        POP     DE
        PUSH    BC
        LD      A,L
        XOR     E
        LD      L,A
        LD      A,H
        XOR     D
        LD      H,A
        RET
MC_ANDAND:
        POP     BC
        POP     DE
        PUSH    BC
        LD      A,L
        OR      H
        JP      Z,__AND2R
        LD      A,E
        OR      D
        JP      Z,__AND2R
        LD      HL,1
        RET
__AND2R:
        LD      HL,0
        RET
MC_AND:
        POP     BC
        POP     DE
        PUSH    BC
        LD      A,L
        AND     E
        LD      L,A
        LD      A,H
        AND     D
        LD      H,A
        RET
MC_EQ:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_CMP
        RET     Z
        DEC     HL
        RET
MC_NE:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_CMP
        RET     NZ
        DEC     HL
        RET
MC_GT:
        POP     BC
        POP     DE
        PUSH    BC
        EX      DE,HL
        CALL    MC_CMP
        RET     C
        DEC     HL
        RET
MC_LE:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_CMP
        RET     Z
        RET     C
        DEC     HL
        RET
MC_GE:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_CMP
        RET     NC
        DEC     HL
        RET
MC_LT:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_CMP
        RET     C
        DEC     HL
        RET
MC_CMP:
        LD      A,E
        SUB     L
        LD      E,A
        LD      A,D
        SBC     A,H
        LD      HL,00001H
        JP      M,MC_CMP1
        OR      E
        RET
MC_CMP1:
        OR      E
        SCF
        RET
MC_LNEG:
        LD      A,H
        OR      L
        LD      HL,0
        JR      Z,MC_LNEG1
        RET
MC_LNEG1:
        INC     L
        RET
MC_UGE:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_UCMP
        RET     NC
        DEC     HL
        RET
MC_ULT:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_UCMP
        RET     C
        DEC     HL
        RET
MC_UGT:
        POP     BC
        POP     DE
        PUSH    BC
        EX      DE,HL
        CALL    MC_UCMP
        RET     C
        DEC     HL
        RET
MC_ULE:
        POP     BC
        POP     DE
        PUSH    BC
        CALL    MC_UCMP
        RET     Z
        RET     C
        DEC     HL
        RET
MC_UCMP:
        LD      A,D
        CP      H
        JR      NZ,CUCMP1
        LD      A,E
        CP      L
CUCMP1:
        LD      HL,00001H
        RET
MC_ASR:
        EX      DE,HL
        DEC     E
        RET     M
        LD      A,H
        RLA
        LD      A,H
        RRA
        LD      H,A
        LD      A,L
        RRA
        LD      L,A
        JR      MC_ASR+1
MC_ASL:
        POP     BC
        POP     DE
        PUSH    BC
        EX      DE,HL
MC_ASL4:
        DEC     E
        RET     M
        ADD     HL,HL
        JR      MC_ASL4
MC_SUB:
        POP     BC
        POP     DE
        PUSH    BC
        LD      A,E
        SUB     L
        LD      L,A
        LD      A,D
        SBC     A,H
        LD      H,A
        RET
MC_NEG:
        CALL    MC_COM
        INC     HL
        RET
MC_COM:
        LD      A,H
        CPL
        LD      H,A
        LD      A,L
        CPL
        LD      L,A
        RET
MC_MULT:
        POP     BC
        POP     DE
        PUSH    BC
        LD      B,H
        LD      C,L
        LD      HL,00000H
MC_MLT1:
        LD      A,C
        RRCA
        JR      NC,CMLT2
        ADD     HL,DE
CMLT2:
        XOR     A
        LD      A,B
        RRA
        LD      B,A
        LD      A,C
        RRA
        LD      C,A
        OR      B
        RET     Z
        XOR     A
        LD      A,E
        RLA
        LD      E,A
        LD      A,D
        RLA
        LD      D,A
        OR      E
        RET     Z
        JR      MC_MLT1
MC_DIV:
        LD      B,H
        LD      C,L
        LD      A,D
        XOR     B
        PUSH    AF
        LD      A,D
        OR      A
        CALL    M,MC_DENEG
        LD      A,B
        OR      A
        CALL    M,MC_BCNEG
        LD      A,010H
        PUSH    AF
        EX      DE,HL
        LD      DE,00000H
MC_DIV1:
        ADD     HL,HL
        CALL    MC_RDEL
        JR      Z,MC_DIV2
        CALL    MC_PBCDE
        JP      M,MC_DIV2
        LD      A,L
        OR      001H
        LD      L,A
        LD      A,E
        SUB     C
        LD      E,A
        LD      A,D
        SBC     A,B
        LD      D,A
MC_DIV2:
        POP     AF
        DEC     A
        JR      Z,MC_DIV3
        PUSH    AF
        JR      MC_DIV1
MC_DIV3:
        POP     AF
        RET     P
        CALL    MC_DENEG
        EX      DE,HL
        CALL    MC_DENEG
```


原因究明の際の条件統一　原因を追いかける際に，RAMの初期状態など不定要素を統一しておくことで状況が整理されて，違いが明確になること．

〈リスト7-4〉 自動的に組み込まれるIOLIBZ.LIB(ファイル名は予約)の内容②

```
        EX      DE,HL
        RET
MC_DENEG:
        LD      A,D
        CPL
        LD      D,A
        LD      A,E
        CPL
        LD      E,A
        INC     DE
        RET
MC_BCNEG:
        LD      A,B
        CPL
        LD      B,A
        LD      A,C
        CPL
        LD      C,A
        INC     BC
        RET
MC_RDEL:
        LD      A,E
        RLA
        LD      E,A
        LD      A,D
        RLA
        LD      D,A
        OR      E
        RET
MC_PBCDE:
        LD      A,E
        SUB     C
        LD      A,D
        SBC     A,B
        RET
MC_SWITCH:
        EX      DE,HL
        POP     HL
SWLOOP:
        LD      C,(HL)
        INC     HL
        LD      B,(HL)
        INC     HL
        LD      A,B
        OR      C
        JR      Z,SWEND
        LD      A,(HL)
        INC     HL
        CP      E
        LD      A,(HL)
        INC     HL
        JR      NZ,SWLOOP
        CP      D
        JR      NZ,SWLOOP
        LD      H,B
        LD      L,C
SWEND:
        JP      (HL)
DEBUGSP:
        LD      HL,2
DEBUGSP2:
        ADD     HL,SP
        PUSH    HL
        POP     IY
        RET
```

〈リスト7-6〉 アセンブラXAの出力したアセンブル・リストtest0.lstの内容

```
;System Load Compiler [mini-'C' V1.0] for Z80/KC80 1995.3
;By System Load CO.,LTD. & K.Kino Copyright (C) All rights reserved

                        include         "startup.h"

8000                    RAM_TOP EQU     08000h

0000                            ORG     0000h
0000    C3 69 00                jp      startup_start

0066                            ORG     0066h
0066    00                      nop
0067    ED 45                   retn

0069                    startup_start:
0069    F3                      di
006A    31 FF 9F                ld      sp,9fffh
006D    ED 5E                   im      2
006F    CD 1D 02                call    __main
0072                    __st_loop__:
0072    F3                      DI
0073    00                      nop
0074    18 FC                   jr      __st_loop__

0076                    MC_GCHAR:
0076    7E                      LD      A,(HL)
0077                    MC_SXT:
0077    6F                      LD      L,A
0078    07                      RLCA
0079    9F                      SBC     A,A
007A    67                      LD      H,A
007B    C9                      RET

(中略，この部分は［IOLIBZ.LIB］と同じ)

0218                    DEBUGSP2:
0218    39                      ADD     HL,SP
0219    E5                      PUSH    HL
021A    FD E1                   POP     IY
021C    C9                      RET

                        ;main(){
021D                    __main:
                        ;#asm

021D    00                      nop

                        ;#endasm
                        ;}
021E    C9                      RET

021F                            END
```

〈リスト7-8〉 コンパイラMCの出力したアセンブラ・ソースのtest1.asmの該当部分

```
;int sub_1(int a){
__sub_1:
;       int c;
        PUSH BC         ← 変数の確保はスタックに
;       c = a * 2;
        LD HL,00H
        ADD HL,SP
        PUSH HL
        LD HL,06H
        ADD HL,SP
        CALL mc_gint    ← (HL)の内容をHLに移す
        PUSH HL
        LD HL,02H
        CALL mc_mult    ← 乗算ライブラリを使用
        CALL mc_pint    ← HLの内容を(HL)に移す
;       return(c);
        LD HL,00H
        ADD HL,SP
        CALL mc_gint
        POP BC          ← 演算結果を［BC］の変数に格納
        RET
;}
;void sub_2(int b){
__sub_2:
;       return;
        RET             ← 何もしない時はそのままリターン
;}
;main(){
__main:
;       int a,b;
        PUSH BC         ← 二つのint型変数の領域を確保
        PUSH BC
;       a = 1;
        LD HL,02H
        ADD HL,SP
        PUSH HL
        LD HL,01H
        CALL mc_pint
;       b = sub_1(a);
        LD HL,00H
        ADD HL,SP
        PUSH HL
        LD HL,04H
        ADD HL,SP
        CALL mc_gint
        PUSH HL
        CALL __sub_1    ← 関数呼び出しはサブルーチン・コール
        POP BC
        CALL mc_pint
;       sub_2(b);
        LD HL,00H
        ADD HL,SP
        CALL mc_gint
        PUSH HL
        CALL __sub_2    ← 関数呼び出しはサブルーチン・コール
        POP BC
;}
        POP BC          ← 確保した変数領域を解放して終了
        POP BC
        RET
```

〈リスト7-10〉 コンパイラMCの出力したアセンブラ・ソース［test2.asm］①

```
;char a[10];             ← 広域変数はあとで定義される
;int b[10];                (アドレスが上位であるため?)
;main(){
__main:
;       char c1, c2[5]; ← char型変数のスタックへの確保
        DEC SP
        DEC SP
        PUSH BC
        PUSH BC
;       int d1, d2[5];  ← int型変数のスタックへの確保
        PUSH BC
        EX DE,HL
        LD HL,0FFF6H
        ADD HL,SP
        LD SP,HL
        EX DE,HL
;       c1 = 1;         ← char型変数への定数のセット
        LD HL,11H
        ADD HL,SP
        PUSH HL
        LD HL,01H
        POP DE
        LD A,L
        LD (DE),A
```

〈リスト7-10〉 コンパイラMCの出力したアセンブラ・ソース[test2.asm]②

```
;       d1 = 1;             ← int型変数への定数のセット
        LD HL,0AH
        ADD HL,SP
        PUSH HL
        LD HL,01H
        CALL mc_pint
;       c2[0] = 1;          ← char型配列への定数のセット
        LD HL,0CH
        ADD HL,SP
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,01H
        POP DE
        LD A,L
        LD (DE),A
;       d2[0] = 1;          ← int型配列への定数のセット
        LD HL,00H
        ADD HL,SP
        PUSH HL
        LD HL,00H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,01H
        CALL mc_pint
;}
        EX DE,HL            ← 確保された変数領域の開放
        LD HL,12H
        ADD HL,SP
        LD SP,HL
        EX DE,HL
        RET
        ORG     RAM_TOP     ← RAM領域への広域変数の確保
__a:    DEFS 0AH            ← char型は配列要素数だけ
__b:    DEFS 14H            ← int型は配列要素数の2倍
```

〈リスト7-11〉 アセンブラ版を置き換えたサンプル・ソースctest.cの内容

```
#define         pio_a   0x1c
#define         pio_b   0x1e

int     disp, sw, counter, table_0[11];     ← テーブルは配列で用意

void port_out( int port, int data ){        ← 汎用ポート出力
#asm
        ld      hl,4
        add     hl,sp
        ld      c,(hl)
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        out     (c),a
        nop
        ret
#endasm
}

int port_in( int port ){                    ← 汎用ポート入力
#asm
        ld      hl,2
        add     hl,sp
        ld      c,(hl)
        in      a,(c)
        ld      l,a
        ld      h,0
        ret
#endasm
}

void wait_short(){
        int timer1;
        for(timer1=0;timer1<10;timer1++){
#asm
        nop                                 ← NOPを5回、それを10回ループ
        nop
        nop
        nop
        nop
#endasm
        }
}

void sw_scan(){
        sw = 0x07 & (port_in( pio_b ) / 32); ← スイッチ入力は簡潔
}

void led_disp_0(){
        port_out( pio_a+0, table_0[disp] ); ← テーブルから1の位を表示
        port_out( pio_b+0, 0xfe );
        port_out( pio_b+0, 0xff );
}

void led_disp_1(){
        port_out( pio_a+0, table_0[disp] ); ← テーブルから10の位を表示
        port_out( pio_b+0, 0xfd );
        port_out( pio_b+0, 0xff );
}

void led_disp_2(){
        port_out( pio_a+0, table_0[disp] ); ← テーブルから100の位を表示
        port_out( pio_b+0, 0xfb );
        port_out( pio_b+0, 0xff );
}

void led_display(){
        disp = counter % 10;
        led_disp_0();
        if( counter > 9 ) disp = (counter/10) % 10;
        else disp = 10;
        led_disp_1();
        if( counter > 99 ) disp = counter / 100;
        else disp = 10;
        led_disp_2();
}

void wait_long(){
        int timer0;
        for(timer0=0;timer0<50;timer0++){
                sw_scan();                  ← ウエイト中にスイッチを見る
                if( sw != 0x07 ) counter = 0;
                wait_short();               ← 足踏みルーチンを呼ぶ
        }
}

main(){
        table_0[0] = 0xf5;                  ← テーブルの初期値をセット
        table_0[1] = 0x05;
        table_0[2] = 0xd3;
        table_0[3] = 0x57;
        table_0[4] = 0x27;
        table_0[5] = 0x76;
        table_0[6] = 0xf6;
        table_0[7] = 0x65;
        table_0[8] = 0xf7;
        table_0[9] = 0x77;
        table_0[10] = 0x00;
        port_out( pio_a+1, 0xcf );          /* Mode 3 */
        port_out( pio_a+1, 0x00 );          /* 0:out / 1:in */
        port_out( pio_a+0, 0x00 );
        port_out( pio_b+1, 0xcf );          /* Mode 3 */
        port_out( pio_b+1, 0xe0 );          /* 0:out / 1:in */
        port_out( pio_b+0, 0xff );
        port_out( pio_b+0, 0xf8 );
        port_out( pio_b+0, 0xff );
        counter = 0;
        while(1){
                wait_long();                ← ソフトウェア・タイマ
                counter = (counter+1)%256;  ← インクリメントしたデータを
                led_display();              ← 3桁のLEDで表示
        }
}
```


MIDI処理サービス関数 MIDIに関して，個々の処理ごとにシンプルにした多数のサブルーチンを用意して，必要に応じて組み合わせて使用するための，C言語の関数の形になったソフトウェア部品集．

〈リスト7-12〉　アセンブラ・ソースctest.asmの該当部分①

```
;#define                pio_a   0x1c
;#define                pio_b   0x1e
;int    disp, sw, counter, table_0[11];

;void port_out( int port, int data ){
__port_out:
;#asm
        ld      hl,4
        add     hl,sp
        ld      c,(hl)
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        out     (c),a
        nop
        ret
;#endasm
;}
        RET

;int port_in( int port ){
__port_in:
;#asm
        ld      hl,2
        add     hl,sp
        ld      c,(hl)
        in      a,(c)
        ld      l,a
        ld      h,0
        ret

;#endasm
;}
        RET

;void wait_short(){
__wait_short:
;       int timer1;
        PUSH BC
;       for(timer1=0;timer1<10;timer1++){
        LD HL,00H
        ADD HL,SP
        PUSH HL
        LD HL,00H
        CALL mc_pint
mc_6:
        LD HL,00H
        ADD HL,SP
        CALL mc_gint
        PUSH HL
        LD HL,0AH   ←for文を回すカウンタ値はここ
        CALL mc_lt
        LD A,H
        OR L
        JP Z,mc_5
        JP mc_7
mc_4:
        LD HL,00H
        ADD HL,SP
        PUSH HL
        CALL mc_gint
        INC HL
        CALL mc_pint
        DEC HL
        JP mc_6
mc_7:
;#asm
        nop
        nop
        nop
        nop
        nop
;#endasm
;       }
        JP mc_4
mc_5:
;}
        POP BC
        RET

;void sw_scan(){
__sw_scan:
;       sw = 0x07 & (port_in( pio_b ) / 32);
        LD HL,07H
        PUSH HL
        LD HL,1EH
        PUSH HL
        CALL __port_in
        POP BC
        PUSH HL
        LD HL,20H
        POP DE
        CALL mc_div
        CALL mc_and
        LD (__sw),HL
;}
        RET

;void led_disp_0(){
__led_disp_0:
;       port_out( pio_a+0, table_0[disp] );
        LD HL,1CH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,__table_0
        PUSH HL
        LD HL,(__disp)
        ADD HL,HL
        POP DE
        ADD HL,DE
        CALL mc_gint
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xfe );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0FEH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xff );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0FFH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;}
        RET

;void led_disp_1(){
__led_disp_1:
;       port_out( pio_a+0, table_0[disp] );
        LD HL,1CH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,__table_0
        PUSH HL
        LD HL,(__disp)
        ADD HL,HL
        POP DE
        ADD HL,DE
        CALL mc_gint
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xfd );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0FDH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xff );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0FFH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;}
        RET

;void led_disp_2(){
__led_disp_2:
;       port_out( pio_a+0, table_0[disp] );
        LD HL,1CH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,__table_0
        PUSH HL
        LD HL,(__disp)
        ADD HL,HL
        POP DE
        ADD HL,DE
        CALL mc_gint
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xfb );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0FBH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xff );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0FFH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;}
        RET

;void led_display(){
__led_display:
;       disp = counter % 10;
        LD HL,(__counter)
        PUSH HL
        LD HL,0AH
        POP DE
        CALL mc_div
        EX DE,HL
        LD (__disp),HL
;       led_disp_0();
        CALL __led_disp_0
;       if( counter > 9 ) disp = (counter/10) % 10;
        LD HL,(__counter)
        PUSH HL
        LD HL,09H
        CALL mc_gt
        LD A,H
        OR L
        JP Z,mc_13     ← if文の判定はここ
        LD HL,(__counter)
        PUSH HL
        LD HL,0AH
        POP DE
        CALL mc_div
        PUSH HL
        LD HL,0AH
        POP DE
        CALL mc_div
        EX DE,HL
        LD (__disp),HL
;       else disp = 10;
        JP mc_14
mc_13:
        LD HL,0AH
        LD (__disp),HL
mc_14:
;       led_disp_1();
        CALL __led_disp_1
;       if( counter > 99 ) disp = counter / 100;
        LD HL,(__counter)
        PUSH HL
        LD HL,63H
        CALL mc_gt
        LD A,H
        OR L
        JP Z,mc_15
        LD HL,(__counter)
        PUSH HL
        LD HL,64H
        POP DE
        CALL mc_div
        LD (__disp),HL
;       else disp = 10;
        JP mc_16
mc_15:
        LD HL,0AH
        LD (__disp),HL
mc_16:
;       led_disp_2();
        CALL __led_disp_2
;}
        RET

;void wait_long(){
__wait_long:
;       int timer0;
        PUSH BC
;       for(timer0=0;timer0<50;timer0++){
        LD HL,00H
        ADD HL,SP
        PUSH HL
        LD HL,00H
        CALL mc_pint
mc_20:
        LD HL,00H
        ADD HL,SP
        CALL mc_gint
        PUSH HL
        LD HL,32H
        CALL mc_lt
        LD A,H
        OR L
        JP Z,mc_19
        JP mc_21
mc_18:
        LD HL,00H
        ADD HL,SP
        PUSH HL
        CALL mc_gint
        INC HL
        CALL mc_pint
        DEC HL
        JP mc_20
mc_21:
;               sw_scan();
        CALL __sw_scan
;               if( sw != 0x07 ) counter = 0;
        LD HL,(__sw)
        PUSH HL
        LD HL,07H
        CALL mc_ne
        LD A,H
        OR L
        JP Z,mc_22
        LD HL,00H
        LD (__counter),HL
;               wait_short();
mc_22:
        CALL __wait_short
;       }
        JP mc_18
mc_19:
;}
        POP BC
        RET

;main(){
__main:
;       table_0[0] = 0xf5;     ←　配列への定数代入だけ
        LD HL,__table_0
        PUSH HL
        LD HL,00H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0F5H
        CALL mc_pint
;       table_0[1] = 0x05;
        LD HL,__table_0
        PUSH HL
        LD HL,01H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,05H
        CALL mc_pint
;       table_0[2] = 0xd3;
        LD HL,__table_0
        PUSH HL
        LD HL,02H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0D3H
        CALL mc_pint
;       table_0[3] = 0x57;
        LD HL,__table_0
        PUSH HL
        LD HL,03H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,57H
        CALL mc_pint
;       table_0[4] = 0x27;
        LD HL,__table_0
        PUSH HL
        LD HL,04H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,27H
        CALL mc_pint
;       table_0[5] = 0x76;
        LD HL,__table_0
        PUSH HL
        LD HL,05H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,76H
        CALL mc_pint
;       table_0[6] = 0xf6;
        LD HL,__table_0
        PUSH HL
        LD HL,06H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0F6H
        CALL mc_pint
;       table_0[7] = 0x65;
        LD HL,__table_0
        PUSH HL
        LD HL,07H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,65H
        CALL mc_pint
```

〈リスト7-12〉 アセンブラ・ソースctest.asmの該当部分②

```
;       table_0[8] = 0xf7;
        LD HL,__table_0
        PUSH HL
        LD HL,08H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0F7H
        CALL mc_pint
;       table_0[9] = 0x77;
        LD HL,__table_0
        PUSH HL
        LD HL,09H
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,77H
        CALL mc_pint
;       table_0[10] = 0x00;
        LD HL,__table_0
        PUSH HL
        LD HL,0AH
        ADD HL,HL
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,00H
        CALL mc_pint
;       port_out( pio_a+1, 0xcf );    /* Mode 3 */
        LD HL,1CH
        PUSH HL
        LD HL,01H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0CFH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_a+1, 0x00 );    /* 0:out / 1:in */
        LD HL,1CH
        PUSH HL
        LD HL,01H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,00H
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_a+0, 0x00 );
        LD HL,1CH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,00H
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+1, 0xcf );    /* Mode 3 */
        LD HL,1EH
        PUSH HL
        LD HL,01H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0CFH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+1, 0xe0 );    /* 0:out / 1:in */
        LD HL,1EH
        PUSH HL
        LD HL,01H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0E0H
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xff );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0FFH
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xf8 );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0F8H
        PUSH HL
        CALL __port_out
        POP BC
        POP BC
;       port_out( pio_b+0, 0xff );
        LD HL,1EH
        PUSH HL
        LD HL,00H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,01H
        POP DE
        ADD HL,DE
        PUSH HL
        LD HL,0100H
        POP DE
        CALL mc_div
        EX DE,HL
        LD (__counter),HL
;               led_display();
        CALL __led_display
;       }
        JP mc_24
mc_25:
;}
        RET

        ORG     RAM_TOP
__disp:         DEFS 02H
__counter:      DEFS 02H
__table_0:      DEFS 16H
__sw:           DEFS 02H
```

〈リスト7-13〉 スタートアップ・ファイル[STARTUP.H]

```
ctc_0           equ     010h
ctc_1           equ     011h
ctc_2           equ     012h
ctc_3           equ     013h
sio_a           equ     018h
sio_b           equ     01ah
pio_a           equ     01ch
pio_b           equ     01eh

rx_fifo         equ     08000h  ; MIDI Receive
tx_fifo         equ     08800h  ; MIDI Transmit
lcd_fifo        equ     09000h
rx_top          equ     09100h + 2 * 0
rx_end          equ     09100h + 2 * 1
tx_top          equ     09100h + 2 * 2
tx_end          equ     09100h + 2 * 3
timer_flag      equ     09100h + 2 * 4
lcd_top         equ     09100h + 2 * 5
lcd_end         equ     09100h + 2 * 6
lcd             equ     09100h + 2 * 7
work            equ     09200h  ; universal use
ram_top         equ     09300h  ; [C] constant

        org     0000h

        ld      sp,9fffh
        di
        ld      hl,08000h
        ld      a,09fh
__ram_clear__:
        ld      (hl),0
        inc     hl
        cp      h
        jr      nc,__ram_clear__
        im      2
        call    __main
__st_loop__:
        di
        nop
        jr      __st_loop__

        org     0020h
        dw      __midi__
__midi__:
        ex      af,af'
        exx
        ld      de,(rx_top)
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        in      a,(sio_a+0)
        ld      (hl),a
        inc     de
        res     3,d
        ld      (rx_top),de
        exx
        ex      af,af'
        ei
        reti

        org     0066h
        retn

        org     0070h
        dw      _timer_
_timer_:
        ex      af,af'
        ld      a,1
        ld      (timer_flag),a
        ex      af,af'
        ei
        reti
```


ステータス・ビット　制御側のCPUとやりとりするために，LSIの状態を示すビット．これが「busy」でなければ，CPUは制御コマンドやデータをLSIに送る．

〈リスト7-14〉 AKI-80用オリジナル・ライブラリ [library.c]

```
#define         ctc_0           0x10
#define         ctc_1           0x11
#define         ctc_2           0x12
#define         ctc_3           0x13
#define         sio_a           0x18
#define         sio_b           0x1a
#define         pio_a           0x1c
#define         pio_b           0x1e

#define         rx_fifo         0x8000  /* MIDI Receive */
#define         tx_fifo         0x8800  /* MIDI Transmit */
#define         lcd_fifo        0x9000  /* LCD Display */
#define         rx_top          0x9100 + 2 * 0
#define         rx_end          0x9100 + 2 * 1
#define         tx_top          0x9100 + 2 * 2
#define         tx_end          0x9100 + 2 * 3
#define         timer_flag      0x9100 + 2 * 4
#define         lcd_top         0x9100 + 2 * 5
#define         lcd_end         0x9100 + 2 * 6
#define         lcd             0x9100 + 2 * 7
#define         work            0x9200  /* unversal use */

int  i,j,k,l,m,n;               /* universal for local use */
int  midi,rsb,dcb,channel,keyno;/* universal MIDI parameter
*/

/*****  Common Functions  ******************************/

void port_out( int port, int data ){
#asm
        ld      hl,4
        add     hl,sp
        ld      c,(hl)
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        out     (c),a
        nop
        ret
#endasm
}

int port_in( int port ){
#asm
        ld      hl,2
        add     hl,sp
        ld      c,(hl)
        in      a,(c)
        ld      l,a
        ld      h,0
        ret
#endasm
}

void ram_put( int address, int data ){
#asm
        ld      hl,4
        add     hl,sp
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        ld      (bc),a
#endasm
}

int ram_get( int address ){
#asm
        ld      hl,2
        add     hl,sp
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
        ld      a,(bc)
        ld      l,a
        ld      h,0
        ret
#endasm
}

void enable_interrupt(){
#asm
        ei
#endasm
}

void disable_interrupt(){
#asm
        di
#endasm
}

char hex_conv( int data ){
        switch(data){
                case 0:  return('0');
                case 1:  return('1');
                case 2:  return('2');
                case 3:  return('3');
                case 4:  return('4');
                case 5:  return('5');
                case 6:  return('6');
                case 7:  return('7');
                case 8:  return('8');
                case 9:  return('9');
                case 10: return('A');
                case 11: return('B');
                case 12: return('C');
                case 13: return('D');
                case 14: return('E');
                case 15: return('F');
        }
}

/*****  I/O Setting Functions  *************************/

void pio_a_setting( int bitmap ){
        port_out( pio_a+1, 0xcf );  /* Mode 3 */
        port_out( pio_a+1, bitmap );
}

void pio_b_setting( int bitmap ){
        port_out( pio_b+1, 0xcf );  /* Mode 3 */
        port_out( pio_b+1, bitmap );
}

void timer_0_setting(){
        port_out( ctc_0, 0x70 );     /* Interrupt Address */
        port_out( ctc_0, 0xa5 );     /* Timer Mode */
        port_out( ctc_0, 158);       /* about 10msec */
}

void sio_dummy_read(){
#asm
        in      a,(sio_a+0)
#endasm
}

void midi_port_setting(){
        port_out( sio_b+1, 0x18 );
        port_out( sio_b+1, 0x04 );
        port_out( sio_b+1, 0xc4 );
        port_out( sio_b+1, 0x01 );
        port_out( sio_b+1, 0x00 );
        port_out( sio_b+1, 0x05 );
        port_out( sio_b+1, 0x60 );
        port_out( sio_b+1, 0x02 );
```

```
        port_out( sio_b+1, 0x20 );
        port_out( sio_a+1, 0x18 );
        port_out( sio_a+1, 0x04 );
        port_out( sio_a+1, 0xc4 );
        port_out( sio_a+1, 0x01 );
        port_out( sio_a+1, 0x10 );
        port_out( sio_a+1, 0x05 );
        port_out( sio_a+1, 0x68 );
        port_out( sio_a+1, 0x03 );
        port_out( sio_a+1, 0xc1 );
}

/*****  MIDI Functions  ****************************/

void tx_midi_check(){
#asm
        ld      de,(tx_end)
        ld      hl,(tx_top)
        and     a
        sbc     hl,de
        ret     z
        xor     a
        out     (sio_a+1),a
        in      a,(sio_a+1)
        bit     2,a
        ret     z
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      a,(hl)
        out     (sio_a),a
        inc     de
        res     3,d
        ld      (tx_end),de
        ret
#endasm
}

void tx_midi_set(int data){
#asm
        ld      hl,2
        add     hl,sp
        ld      b,(hl)
        ld      de,(tx_top)
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      (hl),b
        inc     de
        res     3,d
        ld      (tx_top),de
        ret
#endasm
}

int rx_midi_check(){
#asm
        ld      de,(rx_end)
        ld      hl,(rx_top)
        and     a
        sbc     hl,de
        jr      nz,___non_event
        ld      h,10
        ret
___non_event:
        ld      a,10000000b
        or      d
        ld      h,a
        ld      l,e
        ld      b,(hl)
        inc     de
        res     3,d
        ld      (rx_end),de
        ld      l,b
        ld      h,0
        ret
#endasm
}

/*****  LCD Functions  *******************************/

//      This Library uses [PA0-PA7] port for LCD !
//
//      Connection Mapping :
//         ===== AKI-80 ===== LCD Module =====
//                   PA 7  -----------------(LED on AKI-80)
//                      6  --  No. 9  E
//                      5  --     10  R/W
//                      4  --     11  RS
//                      3  --      1  DB7
//                      2  --      2  DB6
//                      1  --      3  DB5
//                   PA 0  --      4  DB4
//                    +5V  --     14  Vdd
//                    GND  --     13  Vss
//            contrast VR  --     12  Vo
//                   (nc)  --      5  DB3
//                   (nc)  --      6  DB2
//                   (nc)  --      7  DB1
//                   (nc)  --      8  DB0
//
//      Work Memory Assignment :
//              (lcd + 0) -- 8 bit data buffer to be
transfered
//              (lcd + 1) -- bit 7 (MSB) only for LED

void wait_37usec(){
#asm
    db  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
#endasm
}

void wait_500usec(){
        for(j=0;j<15;j++) wait_37usec();
}

void lcd_command_set( int rs, int data ){
#asm
        ld      a,(lcd+1)
        ld      b,a
        ld      hl,4
        add     hl,sp
        ld      a,(hl)
        and     00000001b
        sla     a
        sla     a
        sla     a
        sla     a
        or      b
        ld      b,a
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        ld      (lcd+0),a
        srl     a
        srl     a
        srl     a
        srl     a
        or      b
        out     (pio_a),a
        set     6,a
        out     (pio_a),a
        nop
        res     6,a
        out     (pio_a),a
        ld      a,(lcd+0)
```

```
        and     00001111b
        or      b
        out     (pio_a),a
        set     6,a
        out     (pio_a),a
        nop
        res     6,a
        out     (pio_a),a
#endasm
        wait_37usec();
}

void lcd_module_setting(){
        pio_a_setting( 0x00 );                  /* bit0-7 output */
        for(i=0;i<10;i++) wait_500usec();
        port_out( pio_a, 0x83 );                /* 10000011 */
        port_out( pio_a, 0xc3 );                /* 11000011 */
        port_out( pio_a, 0x83 );                /* 10000011 */
        for(i=0;i<10;i++) wait_500usec();
        port_out( pio_a, 0x83 );                /* 10000011 */
        port_out( pio_a, 0xc3 );                /* 11000011 */
        port_out( pio_a, 0x83 );                /* 10000011 */
        wait_500usec();
        port_out( pio_a, 0x83 );                /* 10000011 */
        port_out( pio_a, 0xc3 );                /* 11000011 */
        port_out( pio_a, 0x83 );                /* 10000011 */
        wait_37usec();
        port_out( pio_a, 0x82 );                /* 10000010 */
        port_out( pio_a, 0xc2 );                /* 11000010 */
        port_out( pio_a, 0x82 );                /* 10000010 */
        wait_37usec();
        lcd_command_set( 0, 0x28 );             /* function reset */
        lcd_command_set( 0, 0x08 );             /* display off */
        lcd_command_set( 0, 0x01 );             /* all clear */
        for(i=0;i<5;i++) wait_500usec();
        lcd_command_set( 0, 0x06 );             /* entry mode */
        lcd_command_set( 0, 0x0c );             /* display ON */
}

void lcd_fifo_set( int data ){
#asm
        ld      hl,2
        add     hl,sp
        ld      c,(hl)          ; [C] = raw data
        ld      a,(lcd_top)
        ld      l,a
        ld      h,090h
        ld      (hl),c
        inc     l
        ld      a,l
        ld      (lcd_top),a
        ret
#endasm
}

void lcd_display( int address, int data ){
#asm
        ld      hl,4
        add     hl,sp
        ld      a,(hl)
        or      10000000b
        ld      b,a             ; [B] = address
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        and     07fh
        ld      c,a             ; [C] = data
        ld      a,(lcd_top)
        ld      l,a
        ld      h,090h
        ld      (hl),b
        inc     l
        ld      (hl),c
        inc     l
        ld      a,l
        ld      (lcd_top),a
        ret
#endasm
}

void disp_string( int address, char *s ){
        char a;
        lcd_fifo_set( 0x80+address );
        i = 0;
        while(1){
                a = s[i++];
                if( a==0 ) break;
                lcd_fifo_set( a & 0x7f );
        }
}

void disp_hex( int address, int data ){
        lcd_fifo_set( 0x80+address );
        lcd_fifo_set( hex_conv(data/16) );
        lcd_fifo_set( hex_conv(data%16) );
}

void lcd_disp_check(){
#asm
        ld      a,(lcd_end)
        ld      l,a
        ld      a,(lcd_top)
        cp      l
        ret     z
        ld      h,090h
        ld      a,(lcd+1)
        ld      b,a
        ld      a,(hl)
        xor     10000000b
        and     10000000b
        srl     a
        srl     a
        srl     a
        or      b
        ld      b,a
        ld      a,(hl)
        ld      (lcd+0),a
        srl     a
        srl     a
        srl     a
        srl     a
        or      b
        out     (pio_a),a
        set     6,a
        out     (pio_a),a
        nop
        res     6,a
        out     (pio_a),a
        ld      a,(lcd+0)
        and     00001111b
        or      b
        out     (pio_a),a
        set     6,a
        out     (pio_a),a
        nop
        res     6,a
        out     (pio_a),a
        inc     l
        ld      a,l
        ld      (lcd_end),a
        ret
#endasm
}
```


オフセット40h　アドレスの基準位置が16進で「40h」，つまり10進で64だけずれたところから平行に扱っていけるということ．

第8章 JavaからAKI-80へ(前編)

ネットワーク指向の言語JavaでAKI-80プログラムを作る

Javaの思想とAKI-80までの道のり

前章までで，1980年代の主流であったマイコン技術の一つの集大成とも言えるAKI-80をフルに活用してきましたが，いよいよ本章から「20世紀最後の新技術」Javaを取り上げていきましょう

もっとも，Javaはまったく斬新なアイデアによる言語というものではありません．基本的には**C++**をベースとして，さらにこれまで登場している数多くのプログラミング言語をすべて網羅的に検討して，オイシイところを取り込むとともに，問題のある部分(過去の経緯から従来は捨てられなかった仕様）をバッサリと切り捨てたところがユニークであると言えると思います

当初は，**sun**のサイト［http://java.sun.com］において，Javaに関する情報はすべて世界中にフリーで公開されていました．本章の記事は，この頃に筆者が入手したsunのサイトのJavaマニュアルに準拠して書いています．

ところがその後，この情報は正式に出版社から出版するとともに，sunのサイトでの無償公開がなくなってしまいました．そこで，読者の皆さんは，オリジナルに当たるためには，参考文献として，Addison Wesley社の「In The Java Virtual Machine Specification」という本を並行して読むことをお勧めします．本章で「文献」と言う場合には，この本のことを示します．ただし，ここで書かれていることの中身の概略は本書でも紹介しながら検討を進めますので，オリジナルの表現そのものは必要でない，という読者の皆さんはこのまま読み進めてください．

なお，本書では最初の公式バージョンのJDK1.0.2というJava環境について述べていますが，世の中のバージョンは刻々と進化しています．そして，最新のものはバグやセキュリティ・ホールが報告されるとすぐに改訂される，と安定していません．ただし，本書で触れている部分のJavaは，新しい世代にも継承されている基本ですので，安心して読み進めてください．

Javaを開発したSunのホームページにあった，Javaの解説ドキュメントの最初の部分に紹介されている「Javaとは」の部分には，特徴として，

Java: A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language.

と書かれています．インターネット時代の切り札のプログラミング言語のように思われているJavaですが，その最初はなんと，AKI-80のような機器制御のための組み込みマイコンなどのソフトウェアを開発するための問題点からスタートしているというのが面白いと思います．

事実，1997年からは，Sunやセカンド・ソースのメーカからダイレクトにJavaプログラムを実行できる「**ネイティブJavaプロセッサ**」が各種登場してくることになっていますから，1998年頃になれば，秋葉原でも「Javaカード・マイコン」が出てくることでしょう．この意味で，Javaに早くから慣れておくことはよいことかもしれません．

とりあえずこの項目を順に追いながら，AKI-80に代表される従来のマイコン・システムのソフト開発の問題点を確認してみましょう．このマイコン・システムは，「小さく」，「信頼性重視で」，「ポータブルで」，「分散システムで」，「リアルタイムで」，「機器に組み込まれる」というような特性をもつという共通点があります．

前章までに検討したように，AKI-80はまさにこのリクエストに応えたハードウェアとなっていることがわかります．本書では取り上げませんでしたが，AKI-80の中核にある東芝CPUは暴走検出のウォッチドッグ・タイマを内蔵していますから，CPUが暴走してもそれを検出してリセットをかける機構が完備していて，reliableという条件もクリアしています．ハードウェアとしては対応が進んでいるのですが，問題はソフトウェアの開発です．

Sunのグループは，最初はC++でプロジェクトをスタートしましたが，数多くの問題に突き当たったといいます．最初はコンパイラの問題点かと思っていろいろトライしていたものの，時間がたつにつれて，問題

C++　C言語を拡張した「オブジェクト指向」と言われるプログラミング言語．

は言語そのものにあるという方向に進みました．このあたり，日本であれば「なんとか不満な環境をやりくりして開発していく」という姿勢が一般的であるのに対して，言語そのものを作ってしまおうというのは，さすがアメリカのフロンティア精神だと感心します．日本ではTRONとかシグマ・プロジェクトとか第5世代コンピュータとか，お役所が旗振りして打ち上げた新技術はすべていつのまにか消えてしまうという状況ですから，本質的にこのあたりは無理なのかもしれません．

多くの「Java本」や雑誌に紹介されているので，Javaの前身の言語とかHotJavaとかの話については省略します．とりあえずJavaはSunのWWWサイトに置かれた仕様をベースに，1995年後半から世界中はこの「まだ姿を見ないけれどもこれから登場してくる新言語」に注目し，テスト版が出てくるとともに爆発的に普及して，現在ではスタンダードになってしまいました．これまでのプログラミング言語で，これほど短期間に世界標準となったものはありませんでしたし，実体のない時期から世界中で仕様が前向きに議論された言語もJavaが最初のものとなったのです．

さて，それでは上記の「Javaとは」の項目を検討してみましょう．この内容はすでに翻訳された本でもいろいろと紹介されていますので，あえて筆者の独断のフィルタをかけて，AKI-80と関係づけて解説してみることにします．

▶ simple

「simple」というのは，かなり判断に主観が入りますので，あまり詳しく検討する必要はありません．ただし，従来のC++言語の複雑に肥大化した仕様をかなりスッキリと切り捨てたというのは確かにシンプルと言えるでしょう．

JavaとC言語との違いでプログラマにもっとも影響の多いのは「ポインタがなくなった」ということで，**ポインタ操作のテクニック**を駆使してしてきた従来のCプログラミング・テクニックは不要の時代となりました．ポインタは初学者にはなかなか理解しにくい壁だったので，この意味ではJavaは学習に適していると思います．

▶ object-oriented

「object-oriented」が，BASICやCのプログラマにとって最大の脅威点かもしれません．実際，筆者も正直言って，本当にオブジェクト指向の概念を理解しているかどうか，各種のオブジェクト指向環境を使っている今でも不安があります．

しかし，気付かないうちに**OOPS**の考え方はなんとなく身に付いているものであり，肩肘張って「さあ，Javaをやるにはオブジェクト指向をマスターしなくては」と頑張る必要はありません．極端に言えば，OOPSを知らなくてもJavaは使えるのです．Simpleという特性とも関係していますが，オブジェクト指向によって小さな「ソフトウェア部品」(オブジェクト)を組み合わせて巨大なシステムでも構築できるというメリットは，使っていくうちに理解できるでしょう．

▶ distributed

「distributed」という意味は，インターネット環境においては「必要なものがほかにあれば，それを利用する」というソフトウェア実行上の特性と，「自律的に情報交換をして複数のプログラムが協調動作を行う」という特性との両方があるように思います．

たとえば，AKI-80のプログラムはすべての処理がメモリに記述されていなければならないのは当然ですが，もしJavaでAKI-80の開発を行った場合には，システムに固有の部分だけを記述してオシマイという可能性があります．詳しいことは次節で検討しますが，Javaが本質的にネットワーク指向であるという点は，多くのソフトウェア・ハウスがもっとも注目しているのです．

▶ interpreted

「interpreted」とは，Javaプログラムの「**バイト・コード**」という独特のプログラム形式と対応しています．Java言語で記述されたソース・プログラムは「Javaコンパイラ」でコンパイルされますが，その結果は「classファイル」という一種のバイナリ・コードに変換されます．

AKI-80の場合であれば，アセンブラ・ソースやCソースが，バイナリ形式のオブジェクト・ファイルや，これをテキスト表現したインテルHEXファイルになるのと同じです．

▶ architecture neutral

ところで，従来のCPU言語であれば，これが実際のハードウェア上で動くためには，OSごとに別々の環境にさらに変換されなければなりませんでした．たとえば80486のプログラムといっても，OSのないシステムか，MS-DOSか，ROM-DOSかという環境ごとにさらに変換を必要としました．ところがJavaでは，すべてのプラットフォームはインタプリタでバイト・コードを解釈して実行するので，コンパイルしてバイト・コードができれば，もうなにもする必要がありません．

たとえば筆者が，あるAKI-80用Javaプログラムを開発してコンパイルできたとすると，このバイト・コードはWindowsパソコンでもMacでもUnixワークステーションでもAKI-80でも同じようにそのまま走るということなのです．これが「architecture neutral」ということ，そのものです．

▶ robust

「robust」という用語は，**ニューラル・ネットワーク**やファジイ技術の世界でポピュラになった概念で

す．ガチガチに堅いという意味でなく，曖昧な条件や不確定な要素に対してシステムがヘンなところに陥らないという意味でのシステムのタフさ，と筆者は理解しています．従来のプログラミング言語では，コロン一つ，ピリオド一つのミスや違いでロケットが落ちたりロボットが暴走したのですが，このようなクリチカルな敏感さを吸収して，ある程度の幅をもって異常な動作を避けてくれるというような意味だと嬉しいのですが，この点については筆者はあまり実感していません．

つぎのセキュリティの特徴のために，むしろ厳格に文法チェックをしてくれる面倒な言語という印象があるのは，まだJavaプログラマとして慣れていないからなのでしょうか．

言語仕様の上では，**例外処理**についてのサポートがこの点に相当しているようで，従来の言語ではなかなか網羅できなかった異常時処理，例外処理を明示的に記述できるようになっています．

▶ secure

「secure」とは，インターネット時代には必須のポイントです．Unixネットワークで全米のコンピュータ・システムをダウンさせた事件で有名になった「ワーム」というプログラムは，パソコン・ウイルスとはくらべものにならない高度な自律プログラムです．通信ネットワークで人間になりすまして相手のサイトの情報を入手して，自分自身のCソース・コードを相手のマシンに転送し，そこでコンパイルして自分自身を再生させます．さらにそこからあちこちのサイトに自分の分身を送り出すと，なんと自分の痕跡を抹消してしまいます．

このようなウイルスの動作をJavaプログラムに仕込まれてはたいへんなことになります．ワームがシステムに進入するためには，当時のUnixのバグを利用したのですが，バグのないバージョンのC言語であっても，ポインタを利用してメモリ内のデータを破壊したり改変する可能性は残っています．そこでJavaでは，**WWWネットワーク**上でのアプリケーションである「アプレット」では，このような問題のあるコードが含まれていると，コンパイル・エラーとなってバイト・コードができません．

また，バイト・コードは実行に際してインタプリタでふたたびこのようなチェックを受けます．このようなセキュリティ機構によって，「現在のところでは」(Sunが言っている範囲というだけで，本当に絶対に大丈夫であると筆者は断言できません)，Javaはセキュリティの問題のない言語であるとされています．

▶ portable，high-performance

「portable」という特徴もやや曖昧なものですが，これまでに述べた特徴を「複数のシステムに簡単に共用できる」という視点でまとめれば確かにポータブルでもあると思います．「high-performance」という言葉も「言った者勝ち」です．パフォーマンスはプラットフォームの進展とともに評価が変わるものですから，これからの時代の指標としては相対的に意味が低下していると思います．

▶ multithreaded

「multithreaded」は，あまり目立たないながらJavaの大きなポイントです．Javaでは，1本のプログラム内に複数の同時進行するプロセスを混在させた，一種の並列処理システムを実現できます．Javaでは「スレッド」と呼ぶこの概念は，別にシステムがマルチタスクをサポートしていなくても，Javaプログラム自体がマルチタスク的な動作を実現してくれることを保証しています．まだ現在のところはあまりパフォーマンスが出た実例は多くありませんが，**ジャスト・イン・タイム・コンパイラ**やプロセッサの登場とともに，Javaのマルチスレッド機能は大きく注目されていくものと思います．

▶ dynamic

「dynamic」というのは，メモリ管理のサポートという，プログラマにとっては嬉しい機能です．従来では，メモリ・アロケートとかメモリの開放などの処理がたいへんで，致命的なバグの基にもなっていたのですが，セキュリティの意味からJavaではこの機能をプログラマから取り上げて，Javaのシステムがすべてやってくれることとなりました．

このため，標準的なJavaシステムでは「必要メモリ」がかなり大きくなってしまったのですが，メモリは時間とともに安くなっている時代ですから，今後はインターネットに接続される機器では「最小メモリ16 Mバイト」程度は常識となっていくのでしょう．ちなみに筆者のパソコンでJavaを開発したり走らせているものは，Macが20 Mバイト(doublerで40 Mバイト化)，Windowsノートが40 Mバイト，Unixマシンが256 Mバイトほど搭載しています．

さて，ここまでJavaという新言語そのものについて紹介してきましたが，本書ではなんとAKI-80とJavaとを結びつけるという無謀なところに進んでいかなければなりません．もちろん，SunのJavaチップとか，完全にAKI-80をプラットフォームとしたJavaコンパイラやJavaインタプリタが出てくれば(出てくるとも思えませんが，考えているシステム・ハウスはあるかもしれません)，それを使って普通のJava環境として開発するというだけのことです．

しかし本書では，「完全なAKI-80対応のJava開発キット」(SunではこれをJDKと言います)を作るという方向は考えていません．前章までの流れを受けて，AKI-80で機動性のあるシステムを開発していくための

ネイティブJavaプロセッサ　Javaの実行を最初から目的として設計されたCPU．他のCPUはソフトウェアでエミュレーションしてJavaを実行するのに対して高速で強力である．

〈リスト8-1〉 Hello, World!(Java)

```
public class HelloWorld {
        public static void main() {
                System.out.println("Hello, World!");
        }
}
```

〈リスト8-2〉 Hello, World!(C言語)

```
static void main() {
        printf("Hello, World!");
}
```

選択肢の一つとして，アセンブラとCに続いてJavaでも開発できるような手を考えてみようという作戦です．

ここからの流れをざっと展望しておくと，まずはJavaのソフト開発環境とJavaのバイト・コードについて検討して，Javaの思想そのものである「**Java仮想マシン**」について整理します．そして，既存の環境でJavaプログラムを開発してできたバイト・コードを解釈して分析するソフトを自作します．その上で，AKI-80のためのJavaバイト・コードをパソコン上でエミュレーションする環境と，JavaバイトコードをAKI-80のCソース・プログラムに変換する環境とを自作して，最終的に「JavaでAKI-80プログラムを作る」という目標にチャレンジします．

まだJavaそのものが動いている最中でのアプローチですから，完全にどんなAKI-80ソフトもJavaで書けるなどというレベルにはならない実験なのですが，「ハードからソフトまで広く視野に入れる」という本書の目標に向かって，さらに頑張っていきましょう．

Javaソフトの開発とwww用アプレットの開発

ここまで概念的に述べてきたJavaですが，それでは実際にはどんなものなのか，まずはサンプルを眺めてみましょう．世界中のあらゆるCプログラマが最初にトライするサンプルと同じく，**リスト8-1**の例では「Hello, World!」と表示してくれるJavaプログラムを示しています．

これと同じようなCプログラムが**リスト8-2**ですが，両方を比較してみると，ほとんど変わらないことがわかります．違いとしては，Javaでは`main()`の外側に，さらにclassという枠組みがあることです．また，**リスト8-2**のCプログラムの名前は自由に決められたのに対して，**リスト8-1**のJavaプログラムでは，「`class HelloWorld {}`」と記述されているために，このソース・プログラムの名前は[HelloWorld.java]で固定的になります．もし別の名前にリネームすると，エラーとなってコンパイルに失敗します．

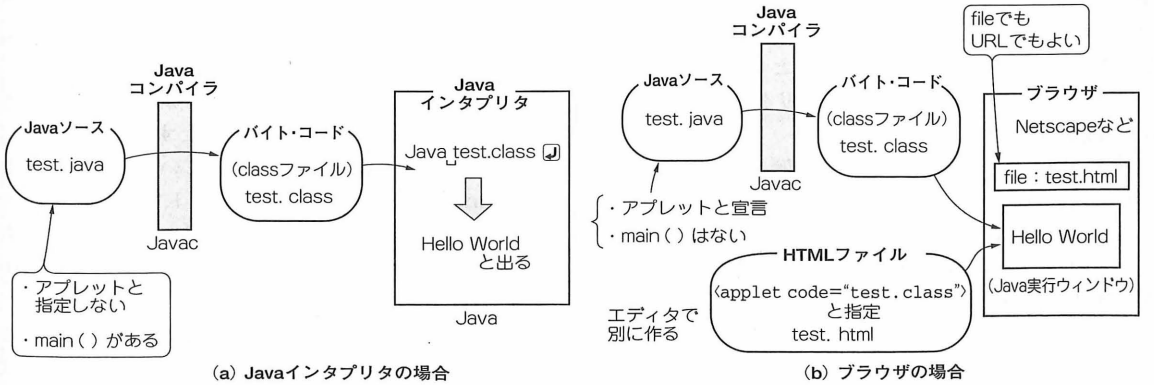
また，この名前でコンパイルに成功したあとでバイト・コードのファイル名を[HelloWorld.class]から別の名前にリネームした場合には，実行の際にJavaインタプリタからエラーが出て実行されません．Javaのセキュリティがここにも効いています．

ところで，ここまではJavaプログラムとひとくくりにしてきましたが，実は「Javaプログラム」には大きく2種類の形態があります．一方は従来のBASICインタプリタのようなプログラミング言語として任意のアプリケーションを作るものであり，もう一方は「アプレット」と呼ばれる**WWWホームページ**上で動作するプログラムです．

リスト8-1の例は普通のアプリケーションの例なので，コンパイルされた結果の[HelloWorld.class]というファイルを指定してJavaコンパイラjavacを起動すると，コンソール(通常のコマンド・ラインを表示している標準入出力端末)に「`Hello World!`」と表示されます．

これに対して，Javaアプレットというのは，**Netscape**などのWWWブラウザで**HTMLファイル**を表示(ブラウジング)しているときに，Javaアプレット

〈図8-1〉 Javaスタンドアロン・アプリケーションの開発と実行



ポインタ操作のテクニック　C言語では，配列変数のアクセスに，その配列自体の置かれたアドレスである「ポインタ」を利用するというマニアックな手法が流行した．

〈リスト8-3〉Javaアプレットの記述

```
import java.awt.*;
import java.applet.*;
public class test001 extends Applet {
        public void paint(Graphics context) {
                context.drawString("Hello World !", 100, 50);
        }
}
```

〈リスト8-4〉WWWのためのHTMLファイル

```
<title>Java Applet Test</title>
<BODY>
<H1>「Hello World」と表示する</H1>
<applet code="test001.class" width=300 height=100 >
</applet>
</BODY>
```

を示す「タグ」と呼ばれるコマンドがあると，その部分に相当するプログラムとしてダウンロードされて実行されるというものです．ここではmain()は不要であるとか，セキュリティ上の理由からいくつかの動作を禁止されているとかの条件があり，Javaアプリケーションとは呼び出し方法も異なります．

図8-1は，このような2種類のJavaプログラムについて，開発から実行までの流れをまとめたものです．まず**図8-1(a)**上の「Javaアプリケーション」では，Javaソース・プログラム(拡張子は[.java])は，通常のテキスト・エディタで編集して，これをJavaコンパイラjavacでコンパイルします．エラーがなければ，結果は[.class]という拡張子の**クラス・ファイル**となります．これを実行するためには，Javaインタプリタjavaにクラス・ファイルを指定して渡せばいいのです．

図8-1(b)の「Javaアプレット」の場合にも，外見上はほぼ同じ流れとなります．つまり，ソース・プログラム(拡張子[.java])をテキスト・エディタで編集して，これをJavaコンパイラjavacでコンパイルして，エラーがなければ，結果は[.class]という拡張子のクラス・ファイルとなります．ただし，アプレットの場合には**リスト8-3**の例のように，「extends Applet」という表現が加わって，これがJavaアプレットとなることを宣言しています．このようなアプレット・プログラムはJavaインタプリタに食わせようとしても拒絶されて，スタンドアロンのアプリケーションとしては実行できません．そこで**リスト8-4**のような，wwwのためのHTMLファイルの中で，「<applet>」と「</applet>」とで囲まれた部分に，「code="test001.class"」として指定します．このようなHTMLをNetscapeのようなブラウザに与えると，その画面の中でJavaプログラムとして実行されるというのがアプレット(小さなアプリケーションという意味)なのです．

もう一つだけ，Javaアプレットの例を紹介しておきましょう．**リスト8-5**は，筆者のホームページの最初の[index.html]に置いてある二つのJavaプログラムのうちの一つ[bound.java]で，筆者の子供の顔と3次元球体に張り付けたモナリザの顔という2種類の画像ファイルが，それぞれ別々のスピードで移動して，衝突すると向きを変えるというものです．実例は，

http://www.kobe-yamate.ac.jp/~nagasm/

にあるので見ていただくとして(一種のアニメーションで，誌面で説明するのは困難)，この概略を説明しておきましょう．

1行目と2行目は，C言語プログラムであれば「#include」によって取り込んでいる一種の**拡張ライブラリ**だと思えば十分です．Javaでは，このimport文によって，より上位の概念のサービスを「継承」して取り込むことができます．「*」印はワイルド・カードです．

4行目の「class moving_gif extends Applet implements Runnable」という部分が，このJavaアプレットの全体の意味付けを示しています．

つまり，このJavaプログラムのソース名は[moving_gif.java]，コンパイルされたバイト・コードは[moving_gif.class]であること，スタンドアロンのアプリケーションでなくWWWで見るアプレットであること，さらに連続アニメーションのように刻々と実行され続ける**「Runnable」プログラム**であることが宣言されています．5～10行目ではいろいろな変数などが宣言されていますが，Java固有の名称などには，ここではあまり深入りしないでおきましょう．

さて，**リスト8-5**をちょっと全体として眺めておくと，これは大きく「class moving_gif{}」というクラスとしてくくられたプログラムで，その中にC言語でいえば関数のような，「init()」，「start()」，「paint()」などの「**メソッド**」が並んでいることがわかります．また，スタンドアロンのアプリケーションではないので，main()はありません．Javaアプレットは，WWWブラウザにダウンロードされてくると，もし明示的に記述されていればまずinit()を行い，次にstart()　を行い，ブラウザが別のページにジャンプするなどアプレットを停止すると自動的にstop()を行う，…などと決まっているのです．そしてRunnable指定されたアプレットでは，run()やpaint()を実行し続けるように決まっています．これが，アニメーションの枠組みを簡単に提供しているのです．

init()で行っている初期設定は，image1からimage8までの変数に，指定された8種類の[gif]形式

〈リスト8-5〉筆者のホームページの最初のJavaプログラム
[bound.java]

```
import java.awt.*;
import java.applet.*;

public class moving_gif extends Applet implements Runnable {
  int x, y, xx, yy;
  int d_x=1, d_y=1, d_xx=-1, d_yy=-1;
  int speed=4, speedd=2;
  Image image1,image2,image3,image4,image5,image6,image7,image8;
  private Thread flow;
  boolean done, pause;

  public void init() {
    image1 = getImage(getDocumentBase(), "image/s1.gif");
    image2 = getImage(getDocumentBase(), "image/s2.gif");
    image3 = getImage(getDocumentBase(), "image/s3.gif");
    image4 = getImage(getDocumentBase(), "image/s4.gif");
    image5 = getImage(getDocumentBase(), "image/s5.gif");
    image6 = getImage(getDocumentBase(), "image/s6.gif");
    image7 = getImage(getDocumentBase(), "image/s7.gif");
    image8 = getImage(getDocumentBase(), "image/s8.gif");
    setBackground(Color.black);
  }

  public void start() {
    done = false;
    pause = false;
    x = 0; y = 0; xx = 500; yy = 200;
    flow = new Thread(this);
    flow.start();
  }

  public void run() {
    try {
      while (!done) {
        Thread.sleep(200);
        if (!pause) repaint();
      }
    }
    catch (Exception e){};
  }

  public void paint(Graphics g) {
    Dimension d = size();
    if(d_x==1 && d_y==1) g.drawImage(image1, x, y, 50, 50, this);
    else if(d_x==-1 && d_y==1) g.drawImage(image2, x, y, 50, 50, this);
    else if(d_x==1 && d_y==-1) g.drawImage(image3, x, y, 50, 50, this);
    else if(d_x==-1 && d_y==-1) g.drawImage(image4, x, y, 50, 50, this);
    if(d_xx==1 && d_yy==1) g.drawImage(image5, xx, yy, 40, 40, this);
    else if(d_xx==-1 && d_yy==1) g.drawImage(image6, xx, yy, 40, 40, this);
    else if(d_xx==1 && d_yy==-1) g.drawImage(image7, xx, yy, 40, 40, this);
    else if(d_xx==-1 && d_yy==-1) g.drawImage(image8, xx, yy, 40, 40, this);
    x += d_x * (3+speed); y += d_y * (3+speed);
    xx += d_xx * (3+speedd); yy += d_yy * (3+speedd);
    if( (xx>x-40) && (xx<x+50) && (yy>y-40) && (yy<y+50) ){
      if(yy<y-25){
        if(d_yy==1) d_yy=-1;
        if(d_y==-1) d_y=1;
      }
      else if(xx<x-25){
        if(d_xx==1) d_xx=-1;
        if(d_x==-1) d_x=1;
      }
      else if(xx>x+35){
        if(d_xx==-1) d_xx=1;
        if(d_x==1) d_x=-1;
      }
      else if(yy>y+35){
        if(d_yy==-1) d_yy=1;
        if(d_y==1) d_y=-1;
      }
    }
    if(x<0) d_x = 1;
    if(y<0) d_y = 1;
    if(x>d.width-50) d_x = -1;
    if(y>d.height-50) d_y = -1;
    if(xx<0) d_xx = 1;
    if(yy<0) d_yy = 1;
    if(xx>d.width-40) d_xx = -1;
    if(yy>d.height-40) d_yy = -1;
  }

    public boolean mouseDown(java.awt.Event evt, int x, int y) {
      pause = !pause;
      if(pause == true){
        speed = (speed+2)%7;
        speedd = (speedd+1)%6;
      }
      return(true);
    }

    public void stop() {
      done = true;
      flow.stop();
    }

  }
```


バイト・コード　Java言語で書かれたJavaソース・プログラムをJavaコンパイラで変換してできたオブジェクト・コードのこと．マシンやOSに依存しない．

〈リスト8-6〉 アプレットを呼び出すHTMLファイル

```
<TITLE>Yoichi Nagashima Home Page</TITLE>
<BODY BGCOLOR="#000000" TEXT="#ffffff" LINK="#00ff00" VLINK="#ff00df" ALINK="#0000ff">
<CENTER>

<applet code="moving_gif.class" width=589 height=221 >
</applet>

</CENTER>
</BODY>
```

の画像ファイルを呼び出してくることだけです．次のstart()では，マウス状態のブーリアン変数を初期設定するとともに，新しいスレッド(並列実行されるタスクのようなもの)を規定しています．これまでC言語で「無限ループの中でマウスの状態を見る」というタイプのプログラムしか書いたことのない人にはちょっと飛躍がありますが，JavaではUnixのXウィンドウ・システムのような「イベント・ドリブン」方式のタスクを簡単に表現できるのです．

つまりここの場合，明示的に無限ループを作る必要はなく，「このプログラムが有効である限り連続的にチェックされている処理(スレッド)がある」とだけ宣言しておけば，もしマウス・ボタンが押されるというイベントが発生すると，自動的にあとに記述されたmouseDown()という処理を実行してくれるのです．

run()というメソッドでは，このバックグラウンド処理を200 msごとに行う(その間はsleepしている)こと，そして目覚めたらrepaint()すなわち再度paint()を呼ぶと記述しています．paint()の中の処理は，ちょっと馬鹿丁寧に書いていますが，要するに，

- それぞれの画像は斜め45度の角度で進む
- それぞれの画像はエリアの縁に衝突したら反射する
- それぞれの画像同士が衝突しても反射する

という判定を行って，新しい移動ベクトルを保存しているだけです．

mouseDown()というイベントについては，アニメーションを停止するか再開するかをトグル動作させるとともに，二つの画像の新しい移動速度を設定しています．この画像サイズと全体の枠組みのサイズとは，かなり大きな公倍数になるように半端に設定していますから(このアプレットを呼び出すリスト8-6のHTMLファイルを参照)，かなり長い時間にわたって，ほぼ毎回少しずつ状態が変わって，眺めていて飽きないようにしています．

本書はJavaプログラミングそのもの，あるいはWWWのためのホームページ制作について述べた本というものではありませんので，ここでこれ以上，Javaの言語仕様やプログラミング，HTML言語やブラウザの動作などについて詳しく解説する誌面もありません．本書の流れから必要となるのは，

- NetscapeなどのブラウザはHTMLファイルを読み込んで表示する
- HTMLファイル中にappletタグがあると，該当するJavaアプレットがダウンロードされて実行される
- JavaアプレットやプログラムはJava言語で作成してコンパイルする
- コンパイルされたバイト・コードはアプレットでもスタンドアロン・アプリケーションでもまったく共通である

というような点です．興味のある方は，巷に多く溢れている「Java本」などで調べてみてください．

さて，ここで本書の「JavaでAKI-80」という目標から，具体的に何をしていくのかがおぼろげに見えてきます．対象システムはAKI-80で，このプログラムをJavaで開発するという目的の外枠はシンプルです．このJavaプログラムがスタンドアロン・アプリケーションであるのかアプレットであるのかというのは，コンパイルしてバイト・コードであるappletプログラムができるところまでは変わらないのです．筆者の構想は，この共通のバイト・コード・ファイルをCソースに変換するコンバータを自作することなのですが，同時にWWWブラウザで擬似的にデバッグ(エミュレーション)できたら面白いかなと考えています．つまり，システムをうまく騙したJavaアプレットとしてJavaでAKI-80プログラムを開発しようという魂胆なのです．どこまでうまくいくかは別として，この路線で進めていくことにします．

Java Virtual Machine SPECの検討

ここまでのJavaの紹介は，一般的な「Java本」でも数多く扱われているのですが，ここからはいよいよ本書ならではの道のりをたどることになります．ここで大きな手がかりとなるのが，すでに紹介したSunの「Java仮想マシン仕様」という膨大な資料の本です．

これは，普通のプログラマにはあまり必要のないものです．

すでに紹介したJavaの特徴として，「システムのハードウェアを選ばない」というものがありました．しかし，実際にUnixやMacやWindows，さらには今後登場してくるであろう新しいアーキテクチャのシステムでJavaが動く(過去にコンパイルされたバイト・コ

ニューラル・ネットワーク　神経回路網をモデルにした，コンピュータ・システムの一つの形態，学習機能や，特定の神経細胞が異常になっても全体として壊れないなどの特徴をもつ．

ードが，新しい環境で再コンパイルすることなくかならず動く)という凄い状況を保証するためには，「仮想的なマシン」を明確に規定して，その規定のもとでは絶対に間違いのない動作を実現するという，壮大で厳格な体系が必要となります．それが，このVM(仮想マシン)SPECなのです．筆者の感想としては，よく読めば読むほど味の出る，Sunの底力を実感できる素晴らしいドキュメントだと思います．

本書はこのVMSPECをすべて詳細に解説して，「新しいJavaチップを作る」，「新しいJavaコンパイラを作る」というメーカのための参考書にするという性格でもありませんので，つまみ食いでいくつかのポイントを紹介しつつ検討していきましょう．

なお、参考文献から該当する部分を引用するわけにはいきませんので，以下の解説だけを読み流すということでも結構です．

まずあらゆるプログラミング言語に必要となる，「サポートするデータのタイプ」については，従来のC言語の混乱しきった状況(マシンによって定義がいろいろに異なる)を避けて，明確に規定しています．Javaではパソコンでもワークステーションでも，「byte」は8ビット幅，「short」は16ビット幅，「int」は32ビット幅の2の補数表現の整数，ということです．

また，Javaでは**Unicode**という文字表現を用いているために，char変数は16ビット幅になります．まだ日本語環境との相性はあまりよくないのですが，今後は日本もUnicodeになっていく必要があるのかもしれません．

レジスタとしては32ビット幅として規定し，ローカル変数もレジスタによるオフセットとして表現されます．これは，AKI-80のCコンパイラのところで見てきたスタックと同じような考え方によるものです．また，ジャンプ先や文字列の指定や変数などのすべてのオペランドも，Java仮想マシンでは**オペランド・スタック**としてスタックに積んで表現します．個々に具体的なメモリ空間のアドレスやレジスタを割り当てるわけにはいかない(なんせ仮想CPUですから)ために，これも当然のこととなります．AKI-80のときにMini-Cで体験した発想法(すべてをスタック経由で処理)が，ここで俄然，役に立ってきます．

そして「命令セット」の部分についてです．ここは簡単に言えば，当然ですがJavaはノイマン方式だということです．オペコードをフェッチして，さらにオペコードごとに必要なデータを読み出して処理を行い，これを「終了」まで繰り返すという当たり前のことです．

本書のアプローチでは，この当たり前のことを地道にフォローしていくことになります．なお，ここで注意すべきことは，Javaでは8ビット単位の数を組み合わせてより長いデータを表現するときに，記述されているように「first_byte * 256 + second_byte」という順序，つまり上位バイトから順に表現するというところです．インテル・タイプのCPUに慣れた人は，発想の転換を要求されます．

具体的には，32ビット幅のintデータであれば，上位の8ビットから順にそのまま並んでいることになります．従来のCPUでは，「16ビットは下位・上位の順として，これが32ビット幅の高位なのか低位なのか」と悩むことも少なくなかったのですが，この種の混乱が払拭されるのは嬉しいですね．

最後に「制限」として，アーギュメントが255とか，エントリが65535とかの規定がありますが，基本的にJavaは「小さな単位を組み合わせる」という思想がベースにありますから，こんな制限が問題になるような巨大な1本プログラムを作るという現実性はほとんどありません．基本的には，これまでC言語とC++言語の環境でかなり混乱していた部分をスッキリさせたのがJavaであるということなのです．

Javaバイナリ・ファイル解析ツールの製作

ここまでの「概要」に続いて，VMSPECではJavaバイト・コードの個々のデータ・フォーマットの詳細の解説が延々と続きます．

まず最初に，Javaバイト・コード(classファイル)の具体的な姿を見ておくと，**リスト8-7**(p.108),後述するあるJavaプログラムの冒頭の一部)のダンプ・リストのようになっています．これは完全なバイナリ・ファイルであり，このような形態のJavaバイト・コードが，どのようなプラットフォームでも同じ動作をするプログラムとなっているのです．

そして，バイト・コードのフォーマットを一般的な形式として記述した部分を見ると，「すべてのJavaバイト・コードはこの形式になっている」と明確に規定しています．したがって，コンパイルを通過したあとのclassファイルを，たった1バイトでもヘタに改変すると，この形式に対して違反となって受け入れられないという厳格な意味をもちます．文献の中の「u4」とは4バイトにわたる32ビット・データ，「u2」とは16ビット・データということです．たとえば，**リスト8-7**のいちばん最初の「u4 magic」という部分をバイト・コードの最初の4バイトとして見てみると，16進データとして解釈すると「0xCAFEBABE」となっています．参考文献の記述では，「最初の4バイトはマジック・ナンバーで，16進でCAFEBABEでなければならない」というものに対応しています．さすがにJavaティーはコーヒーを符丁にしているというわけです．

文献の続く記述によると，magicに続く二つの16ビ

例外処理　本来の処理とは別に「時間オーバー」「異常信号」「規定外に出た」などの例外的な条件に対して，あらかじめその処理をプログラミングしておくこと．

〈リスト8-7〉 Javaバイト・コードの一部

```
00000000 : CA FE BA BE 00 03 00 2D - 00 B9 08 00 AC 08 00 B4   .......-........
00000010 : 07 00 96 07 00 8C 07 00 - 88 07 00 9C 07 00 7A 07   ..............z.
00000020 : 00 67 07 00 91 07 00 9E - 07 00 A5 07 00 9D 07 00   .g..............
00000030 : AD 07 00 84 07 00 A0 07 - 00 92 07 00 89 0A 00 0F   ................
00000040 : 00 53 09 00 10 00 5E 0A - 00 07 00 44 09 00 0F 00   .S....^....D....
00000050 : 57 0A 00 09 00 55 0A 00 - 09 00 4A 0A 00 07 00 5A   W....U....J....Z
00000060 : 0A 00 0D 00 54 0A 00 05 - 00 50 0A 00 11 00 39 09   ....T....P....9.
00000070 : 00 0F 00 3F 0A 00 0F 00 - 41 0A 00 07 00 59 0A 00   ...?....A....Y..
00000080 : 09 00 3B 09 00 0F 00 4D - 0A 00 11 00 56 0A 00 11   ..;....M....V...
00000090 : 00 3D 09 00 0F 00 5B 0A - 00 06 00 3C 09 00 0F 00   .=....[....<....
000000A0 : 5F 0A 00 0F 00 42 0A 00 - 0F 00 4C 0A 00 05 00 52   _....B....L....R
000000B0 : 0A 00 05 00 3E 0A 00 0A - 00 40 09 00 0F 00 43 0A   ....>....@....C.
000000C0 : 00 0D 00 3A 0A 00 0D 00 - 4F 09 00 0F 00 38 0A 00   ...:....O....8..
000000D0 : 0A 00 51 09 00 0F 00 5D - 09 00 08 00 5C 0A 00 09   ..Q....]....\...
000000E0 : 00 4B 09 00 0F 00 58 0A - 00 11 00 54 09 00 0F 00   .K....X....T....
000000F0 : 49 0A 00 07 00 54 09 00 - 10 00 4E 0C 00 9F 00 6C   I....T....N....l
00000100 : 0C 00 AB 00 9B 0C 00 8A - 00 7D 0C 00 7B 00 8B 0C   .........}..{...
00000110 : 00 93 00 8F 0C 00 AB 00 - 7E 0C 00 86 00 82 0C 00   ........~.......
00000120 : 79 00 AA 0C 00 80 00 7C - 0C 00 99 00 90 0C 00 87   y......|........
00000130 : 00 6F 0C 00 9A 00 AA 0C - 00 98 00 8D 05 00 00 00   .o..............
00000140 : 00 00 00 03 E8 06 40 8F - 38 00 00 00 00 00 0C 00   ......@.8.......
00000150 : A1 00 AA 0C 00 69 00 6D - 0C 00 94 00 B7 0C 00 6E   .....i.m.......n
00000160 : 00 6F 0C 00 B8 00 B6 0C - 00 B0 00 AA 0C 00 85 00   .o..............
00000170 : 78 0C 00 75 00 A7 0C 00 - 8E 00 68 0C 00 76 00 8B   x..u......h..v..
00000180 : 0C 00 97 00 90 0C 00 69 - 00 8B 0C 00 72 00 8B 0C   .......i....r...
00000190 : 00 70 00 7F 0C 00 A4 00 - AA 0C 00 81 00 AA 0C 00   .p..............
000001A0 : 6A 00 8D 0C 00 B3 00 8D - 0C 00 77 00 AA 0C 00 74   j.........w....t
000001B0 : 00 B5 0C 00 A6 00 AA 0C - 00 A3 00 AA 0C 00 A2 00   ................
000001C0 : B2 06 40 12 00 00 00 00 - 00 00 06 3F F8 00 00 00   ..@........?....
000001D0 : 00 00 00 06 3F ED C2 8F - 5C 28 F5 C3 01 00 16 28   ....?...\(.....(
000001E0 : 4C 6A 61 76 61 2F 61 77 - 74 2F 47 72 61 70 68 69   Ljava/awt/Graphi
000001F0 : 63 73 3B 29 56 01 00 0E - 6A 61 76 61 2F 61 77 74   cs;)V...java/awt
```

ット・データがバージョンを示すとあります．確かに**リスト8-7**を調べてみると，minor_versionが3，major_versionが45ということで，マニュアルと合っています．将来的なJava環境が変わっても，この部分でチェックしていくことになるわけです．そして，次にconstant_pool_countという定数がある，…という調子で順に読み込んでいけば簡単と思っていると，ここからがややこしくなってきます．

文献で，cp_info constant_pool [constant_pool_count - 1]と簡単に記述されている部分が最初のくせ者で，これは一種の構造体のような階層的な仕組みで，**可変長のデータ・エリア**を示しているものです．つまり，この定義だけ見ると単なる1行ですが，実際にはここから先に，膨大なデータ領域が展開されていて，その実際のサイズはフォーマットにしたがって解読していかないと判明しないという仕組みになっています．

後にも同様の形式が出てくるので，constant_poolの部分だけは我慢してちょっと詳しく検討してみると，VMSPECでの規定を**リスト8-7**の実例にあてはめてみると，アドレス・オフセット0x0008からの2バイトの00B9というのが，「constant_pool_count」になりますから，この例では10進で185個のconstant_poolがここに続いて定義されている，ということになります．constant_poolとは，Javaプログラムに登場するすべての名前の定義，とでもいうもので，変数名やそれぞれのメソッド名だけでなく，Javaライブラリで参照されるクラス・ライブラリなどの名前もすべてここで定義しています．

さて，では最初のデータを追いかけてみましょう．185個があるという情報に続く**リスト8-7**のアドレス・オフセット0x000Aからの3バイトは，「08　00　AC」となっています．これは文献の「CONSTANT_Class_info」という構造体の最初の要素で，定義によって最初の1バイトである「08」が，「u1　tag」に相当します．

これは文献にある定義から，「CONSTANT_String」であることになり，後で具体的なストリングが定義されている部分をアクセスするためのインデックス「u2 name_index」が0xAC，つまり10進で172であるということを示しています．ここから以下は同様に3バイト単位でしばらく続くのですが，アドレス・オフセット0x003Dからは「CONSTANT_Methodref」という5バイト形式の構造体が登場し，アドレス・オフセット0x0145では「CONSTANT_Double」という9バイト形式の構造体も出てきます．

手作業ではかなり厳しいものがありますが，このようにVMSPECの定義に厳格にさらに**リスト8-7**のバイト・コードを追っていくと，アドレス・オフセット0x01DC，constant_poolの102番目からは，タイプが「CONSTANT_Utf8」という新しいグループが始まり，ここではそれぞれのサイズも可変長となって，さらに追いかけるのがたいへんになってきます．つまり，tagに続く2バイトのlengthで規定される文字列としてデータが続くわけです．ここで登場するのが，すでに名前だけ登場したUnicodeという形式で，文献の中でも解説されているように，1バイト・データはそのまま，2バイトないし3バイト・データは上位ビットの状態で判定していくという可変長の文字データ表現形式です．

WWWネットワーク　ここでは，いわゆるインターネットのこと．

〈リスト8-8〉Javaバイト・コードの一部

```
000006F0 : 00 AE 00 00 00 6A 00 06 - 00 03 00 00 00 3E 2A B6   .....j.......>*.
00000700 : 00 2C 4C 04 3D A7 00 29 - 2A B4 00 25 1C 2A 2B BB   .,L.=..)*..%.*+.
00000710 : 00 11 59 B7 00 34 12 02 - B6 00 22 1C B6 00 1B 12   ..Y..4....".....
00000720 : 01 B6 00 22 B6 00 21 B6 - 00 2D 53 84 02 01 1C 10   ...".."..!..-S.....
00000730 : 0E A4 FF D7 2A B2 00 31 - B6 00 29 B1 00 00 00 01   ....*..1..).....
00000740 : 00 73 00 00 00 1A 00 06 - 00 00 00 19 00 05 00 1A   .s..............
00000750 : 00 0A 00 1B 00 2D 00 1A - 00 36 00 1D 00 3D 00 18   .....-...6...=..
00000760 : 00 01 00 7B 00 8B 00 01 - 00 AE 00 00 00 34 00 04   ...{.........4..
00000770 : 00 01 00 00 00 14 2A BB - 00 09 59 2A B7 00 17 B5   ......*...Y*....
00000780 : 00 20 2A B4 00 20 B6 00 - 1F B1 00 00 00 01 00 73   . *.. .........s
00000790 : 00 00 00 0E 00 03 00 00 - 00 21 00 0C 00 22 00 13   .........!...".. 
000007A0 : 00 20 00 01 00 95 00 8B - 00 01 00 AE 00 00 00 4D   . .............M
000007B0 : 00 03 00 01 00 00 00 19 - 14 00 45 B8 00 32 2A B6   ..........E..2*.
000007C0 : 00 28 A7 FF F6 57 2A 59 - B4 00 35 04 60 B5 00 35   .(...W*Y..5.`..5
000007D0 : B1 00 01 00 00 00 0D 00 - 0D 00 03 00 01 00 73 00   ..............s.
```

基本的に1バイトで文字を表現できる欧米では簡単でコンパクトなのですが，日本など2バイト文字文化圏では，ちょっと処理が面倒になってしまいます．

このように長大なcp_info constant_pool[]のエリアが並ぶと，続いて「access_flags」，「this_class」，「super_class」などという情報のフラグが並びますが，本書ではこの詳細については省略します．そしてさらに，可変長のinterfaces[]，あるいは上記の例と同様の階層化された構造体の集まりとして，

- field_info fields[fields_count]
- method_info methods[methods_count]
- attribute_info attributes[attribute_count]

という一連のグループがずらりと並んでいます．

実際のJavaプログラムとして，C言語あるいはアセンブラのように見える「プログラム・コード」部分はどこかというと，Cの関数に相当する「method_info」の部分，さらに具体的にはこの構造体の中の一つの可変長要素の「type = Code」というブロック(後述)です．VMSPECの中に定義されている，これ以外の個々の規定については，本書で詳細に紹介する誌面がありませんので，興味のある方は原著にあたってください．

この先，文献では複雑な内部階層構造をもつ「attribute_info」の中の，「Code」に相当する部分を記述しています．これが実際のJavaバイト・コードでどうなっているか，を**リスト8-8**の実例(部分)から調べてみましょう．VMSPECの定義にしたがって先頭から順にたどっていくと，このプログラムで最初に登場するCode_attribute構造体のアドレス・オフセットは0x06F0となります．そして，ここからの情報を解析してみると，「u2 attribute_name_index」が0x00AEということで，先に定義されていたconstant_poolの174番目を参照することで，クラスの属性が「ACC_PUBLIC」とパブリックであること，名前が「init 」であることなどが判明します．続いてattribute_lengthの0x6Aから，このメソッドに関する情報が全部で106バイトであること，さらに「max_stackが6個」，「max_localsが3個」，そしてこの部分の「code_length」が0x3Eすなわち62バイトであることなどが記述されていて，この次の「u1 code[code_length]」の部分に実際のバイト・コード本体が並んでいるということになります．

では，この最初の部分「2A B6 00 2C 4C 04 3D A7 00 29」という，たった10バイトを追いかけてみましょう．ここからは，もう昔のこととなった「**手作業による逆アセンブラ**」という作業の世界となります．最初の「2A」は10進で42ですから，Sunの提供しているVMSPECの定義から，この数値の部分を捜します．すると，この1バイトは「ローカル変数<0>をオペランド・スタックにプッシュする」という動作であることがわかります．ここで思い出したいのが，AKI-80のCコンパイラ・プログラムで，関数の先頭にローカル変数を定義したときにも，これをスタック領域に確保したという事実です．これは本質的に同じことを行っているのです．

次の3バイトは，実は「B6 00 2C」とセットになっています．これは最初のB6，すなわち182を調べてみると，VMSPECにあるように他のメソッドを呼び出していて，インデックスとして続く2バイトを伴っていることがわかります．同様にして，さらに続く6バイトも，

- 4C … astore_1(Store object reference into local variable)
- 04…iconst_1(Push integer constant)
- 3D…istore_2(Store integer into local variable)
- A7 00 29…goto 0029(Branch)

などとなっています．これはまさに，逆アセンブラの解読ゲームで，最新のJavaといっても基本的にはプッシュ・ポップやロード・セーブ，さらにGotoでできていると知ってホッとします．

さて，こうしてJavaバイト・コードの内部を詳細に分析してみると，バラしてしまえば基本的なプログラムであるということから，「Javaプログラムの解析ツール」を作りたくなってきます．材料はすべてSunのサイトから全世界に提供されていたのですから，当然

ジャスト・イン・タイム・コンパイラ　Javaの実行速度を向上させるために，Javaバイト・コードをロードしてきた時点で，機種に依存させて再コンパイルするようになっているシステムのこと．

ながら手作業でなくオートマチックにやってみたいものでしょう．

そこで，約2週間(実作業としては1週間程度)で筆者が開発したのが，**リスト8-9**(p.112)にある「**Java逆コンパイラ**」(rejava.c)です．ちょっと長いのですが，これは原理的に256種類あるはずのバイト・コードのすべてをswitch文で判定して，VMSPECに載っていた用語を張り付けてみたからであって，全体の処理としてはそれほど複雑ではありません．厳密に言えば，まだこのプログラムは発展途上であって，完成したものではありません．しかし，Javaプログラムの全体構造を分析するところまではできていると思います．

このツール「rejava」での解析例を一つだけ紹介しましょう．

リスト8-10は，これも筆者のホームページの最初のページに置いているJavaプログラムで(sample.java)，「あなたのマシンの標準時」というディジタル時計となっています．具体的には，マシンから時間情報を取り込んで，カラフルな3次元球体のグラフィックをドット・マトリクス状に並べるというだけのものですが，たまにランダムにドキッとするような仕掛けを入れています．これをコンパイルした結果[sample.class]を「rejava」に与えて解析させた結果をファイルに書き出したのが，**リスト8-11**(p.119)の結果ファイル[result.txt]です．ここでは，まず最初にバイナリ・データをダンプしつつ全データ数をカウントして，次にVMSPECに従って順に内容を解析しています．

「Constant Pool」の部分では，参照番号だけでなく参照先の内容も付加しているので，ほかの情報と突き合わせて解析するのに便利になっています．実際には，それぞれメソッドの内部のアセンブラ的挙動を全部追いかけているわけではないのですが，このくらいの内容が自動解析できれば，AKI-80のためにJavaの側から攻めていくための材料は仕入れることができます．

なお，**リスト8-9**の解析プログラム[rejava.c]は，筆者はUnixワークステーション上で開発したのですが，おそらくちょっとした変更でパソコンでもそのままコンパイルできると思いますので，より高度なツールに完成度を上げるという実験は読者の皆さんにお任せしたいと思います．もし面白いツールができたら，Nifty-Serveの人工知能フォーラムFAIの「Javaの会議室」ででも報告していただきたいと思います．

〈リスト8-10〉
あなたのマシンの標準時[sample.java]①

```
import java.awt.*;
import java.applet.*;
import java.util.*;
import java.net.*;

public class sample sample Applet implements Runnable {

        int j=0, color_sel=1, xx=70, yy=25, sz=10, st=12, mode=1;
        Image i[] = new Image[15];
        private Thread flow;
        int m[][][] = {
{  {0,1,1,1,0}, {1,0,0,0,1}, {1,0,0,0,1}, {1,0,0,0,1}, {1,0,0,0,1}, {1,0,0,0,1}, {0,1,1,1,0}  }, // 0
{  {0,0,1,0,0}, {0,1,1,0,0}, {0,0,1,0,0}, {0,0,1,0,0}, {0,0,1,0,0}, {0,0,1,0,0}, {0,1,1,1,0}  }, // 1
                {  {0,1,1,1,0}, {1,0,0,0,1}, {1,0,0,0,1}, {0,0,0,1,0}, {0,0,1,0,0},
{0,1,0,0,0}, {1,1,1,1,1}  }, // 2
{  {0,1,1,1,0}, {1,0,0,0,1}, {0,0,0,0,1}, {0,0,1,1,0}, {0,0,0,0,1}, {1,0,0,0,1}, {0,1,1,1,0}  }, // 3
                {  {0,0,0,1,0}, {0,0,1,1,0}, {0,1,0,1,0}, {1,0,0,1,0}, {1,1,1,1,1},
{0,0,0,1,0}, {0,0,0,1,0}  }, // 4
                {  {1,1,1,1,1}, {1,0,0,0,0}, {1,0,0,0,0}, {1,1,1,1,0}, {0,0,0,0,1},
{1,0,0,0,1}, {0,1,1,1,0}  }, // 5
                {  {0,1,1,1,0}, {1,0,0,0,1}, {1,0,0,0,0}, {1,1,1,1,0}, {1,0,0,0,1},
{1,0,0,0,1}, {0,1,1,1,0}  }, // 6
                {  {1,1,1,1,1}, {1,0,0,0,1}, {0,0,0,0,1}, {0,0,0,1,0}, {0,0,1,0,0},
{0,1,0,0,0}, {1,0,0,0,0}  }, // 7
                {  {0,1,1,1,0}, {1,0,0,0,1}, {1,0,0,0,1}, {0,1,1,1,0}, {1,0,0,0,1},
{1,0,0,0,1}, {0,1,1,1,0}  }, // 8
                {  {0,1,1,1,0}, {1,0,0,0,1}, {1,0,0,0,1}, {0,1,1,1,1}, {0,0,0,0,1},
{1,0,0,0,1}, {0,1,1,1,0}  }, // 9
        };
        public void init(){
                URL docBase = getDocumentBase();
                for(int n=1;n<=14;n++){
                        i[n] = getImage( docBase, "java/gif/b"+n+".gif");
                }
                setBackground(Color.black);
        }

        public void start(){
                flow = new Thread(this);
                flow.start();
        }

        public void run() {
                try {
                        while (true) {
                                flow.sleep(1000);
                                repaint();
                         }
                }
                catch (Exception e){ j++; };
        }
```


Java仮想マシン　Javaの言語仕様が想定している，「Javaが走るであろうコンピュータ」の仕様の想定のこと．実際のコンピュータは，ソフトウェアによってこのJava仮想マシンをエミュレーションすることになる

〈リスト8-10〉
あなたのマシンの標準時[sample.java]②

```
        public void stop() {
                flow.stop();
        }

        public void paint(Graphics g) {
                Dimension dd = size();
                g.clearRect(0, 0, dd.width, dd.height);
                Date d = new Date();
                int u1 = d.getHours();
                int u2 = d.getMinutes();
                int u3 = d.getSeconds();
                clock_disp(g,u1,u2,u3);
        }

        void clock_disp(Graphics g, int u1, int u2, int u3){
                int addx, addy, dat, datt, color_buf=1;
                addy = yy;
                if(mode != 0){
                        color_buf = color_sel;
                        sz += 3; st += 2; xx -= 40; yy -= 5;
                }
                addx = xx + st * 28; dat = u3 / 10; char_disp(g,addx,addy,dat);
                addx = xx + st * 34; datt = u3 % 10; char_disp(g,addx,addy,datt);
                addx = xx + st * 14; dat = u2 / 10; char_disp(g,addx,addy,dat);
                addx = xx + st * 20; dat = u2 % 10; char_disp(g,addx,addy,dat);
                addx = xx; dat = u1 / 10; char_disp(g,addx,addy,dat);
                addx = xx + st * 6; dat = u1 % 10; char_disp(g,addx,addy,dat);
                addx = xx + st * 12; char_dot(g,addx,addy);
                addx = xx + st * 26; char_dot(g,addx,addy);
                if(mode != 0){
                        color_sel = color_buf;
                        sz -= 3; st -= 2; xx += 40; yy += 5;
                }
                        if(datt ==9){
                        color_sel++;
                        if(color_sel>14) color_sel=1;
                }
                if(Math.random() > 0.93) mode=1;
                else mode=0;
        }

        void char_dot(Graphics g, int addx, int addy){
                dot_draw(g, addx, addy + (int)((double)st * 1.5));
                dot_draw(g, addx, addy + (int)((double)st * 4.5));
        }

        void dot_draw(Graphics g, int adx, int ady){
                int color_res;
                if(mode == 0){
                        color_res = color_sel;
                }
                else{
                        color_res = 1 + ((int)(Math.random() * 999.0)) % 14;
                }
                g.drawImage(i[color_res], adx, ady, sz, sz, this);
        }

        void char_disp(Graphics g, int addx, int addy, int dat){
                for( int i=0; i<5; i++){
                        for (int j=0; j<7; j++){
                                if( m[dat][j][i] ==1 ){
                                        dot_draw(g, addx + st * i, addy + st * j);
                                }
                        }
                }
        }

}
```


WWWホームページ　ここでは，いわゆるインターネットの「ホームページ」のこと．

〈リスト8-9〉 Java逆コンパイラ[rejava.c]①

```
#include <stdio.h>
#include <stdlib.h>

FILE *fp;
int ni[3000];
unsigned char m[1000000], n[3000][256];

void byte_code_printf( long address, long length ){
        int k, l, p1, p2, p3, mm, m1, m2, m3, m4;
        long pp, p4, p5;
        pp = address;
        p1 = m[pp+1]+256*m[pp]; pp += 2;
        p2 = m[pp+1]+256*m[pp]; pp += 2;
        printf(" , max_stack = %d , max_locals = %d , ", p1, p2 );
        p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
        printf("code_length = %d\n", p4);
        k = 0;
        while( k < p4 ){
                printf("¥n¥t¥t¥t¥t%06X¥t", k );
                mm = m[pp++];
                switch(mm){
                        default :       printf("?? %02X ??¥t¥t(..... not defined .....)", mm);
                                break;
                        case 16:        m1 = m[pp++];
                                printf("bipush¥t%02X¥t(Push one-byte signed integer)", m1);
                                k += 1; break;
                        case 17:        m1 = m[pp++]; m2 = m[pp++];
                                printf("sipush¥t%04X¥t(Push two-byte signed integer)", 256*m1+m2);
                                k += 2; break;
                        case 18:        m1 = m[pp++];
                                printf("ldc1¥t%02X¥t(Push item from constant pool)", m1);
                                k += 1; break;
                        case 19:        m1 = m[pp++]; m2 = m[pp++];
                                printf("ldc2¥t%04X¥t(Push item from constant pool)", 256*m1+m2);
                                k += 2; break;
                        case 20:        m1 = m[pp++]; m2 = m[pp++];
                                    printf("ldc2w¥t%04X¥t(Push long or double from constant pool)", 256*m1+m2);
                                k += 2; break;
                        case 1: printf("aconst_null¥t¥t(Push null object reference)");
                                break;
                        case 2: printf("iconst_m1¥t¥t(Push integer constant -1)");
                                break;
                        case 3: case 4: case 5: case 6: case 7: case 8:
                                printf("iconst_%d¥t¥t(Push integer constant)", mm-3);
                                break;
                        case 9: case 10:
                                printf("lconst_%d¥t¥t(Push long integer constant)", mm-9);
                                break;
                        case 11: case 12: case 13:
                                printf("fconst_%d¥t¥t(Push single float)", mm-11);
                                break;
                        case 14: case 15:
                                printf("dconst_%d¥t¥t(Push double float)", mm-14);
                                break;
                        case 21:        m1 = m[pp++];
                                printf("iload¥t%02X¥t(Load integer from local variable)", m1);
                                k += 1; break;
                        case 26: case 27: case 28: case 29:
                                printf("iload_%d¥t¥t(Load integer from local variable)", mm-26);
                                break;
                        case 22:        m1 = m[pp++];
                                printf("lload¥t%02X¥t(Load long integer from local variable)", m1);
                                k += 1; break;
                        case 30: case 31: case 32: case 33:
                                printf("lload_%d¥t¥t(Load long integer from local variable)", mm-30);
                                break;
                        case 23:        m1 = m[pp++];
                                printf("fload¥t%02X¥t(Load single float from local variable)", m1);
                                k += 1; break;
                        case 34: case 35: case 36: case 37:
                                printf("fload_%d¥t¥t(Load single float from local variable)", mm-34);
                                break;
                        case 24:        m1 = m[pp++];
                                printf("dload¥t%02X¥t(Load double float from local variable)", m1);
                                k += 1; break;
                        case 38: case 39: case 40: case 41:
                                printf("dload_%d¥t¥t(Load double float from local variable)", mm-38);
                                break;
                        case 25:        m1 = m[pp++];
                                printf("aload¥t%02X¥t(Load object reference from local variable)", m1);
                                k += 1; break;
                        case 42: case 43: case 44: case 45:
                                printf("aload_%d¥t¥t(Load object reference from local variable)", mm-42);
                                break;
                        case 54:        m1 = m[pp++];
                                printf("istore¥t%02X¥t(Store integer into local variable)", m1);
                                k += 1; break;
                        case 59: case 60: case 61: case 62:
                                printf("istore_%d¥t¥t(Store integer into local variable)", mm-59);
                                break;
                        case 55:        m1 = m[pp++];
                                printf("lstore¥t%02X¥t(Store long integer into local variable)", m1);
                                k += 1; break;
                        case 63: case 64: case 65: case 66:
                                printf("lstore_%d¥t¥t(Store long integer into local variable)", mm-63);
                                break;
```


Netscape　ここでは，Netscape Communication社のブラウザ．

〈リスト8-9〉Java逆コンパイラ[rejava.c]②

```
case 56:        m1 = m[pp++];
        printf("fstore¥t%02X¥t(Store float into local variable)", m1);
        k += 1; break;
case 67: case 68: case 69: case 70:
        printf("fstore_%d¥t¥t(Store float into local variable)", mm-67);
        break;
case 57:        m1 = m[pp++];
        printf("dstore¥t%02X¥t(Store double float into local variable)", m1);
        k += 1; break;
case 71: case 72: case 73: case 74:
        printf("dstore_%d¥t¥t(Store double float into local variable)", mm-71);
        break;
case 58:        m1 = m[pp++];
        printf("astore¥t%02X¥t(Store object reference into local variable)", m1);
        k += 1; break;
case 75: case 76: case 77: case 78:
        printf("astore_%d¥t¥t(Store object reference into local variable)", mm-75);
        break;
case 132:       m1 = m[pp++]; m2 = m[pp++];
        printf("iinc¥t%02X %02X¥t(Increment local variable by constant)", m1, m2);
        k += 2; break;
case 196:       m1 = m[pp++];
        printf("wide¥t%02X¥t(Wider index for accessing local variables in load, store and increment)", m1);
        k += 1; break;
case 188:       m1 = m[pp++];
        printf("newarray¥t%02X¥t(Allocate new array) : ", m1);
        k += 1;
        switch(m1){
                case 4: printf("array type = T_BOOLEAN");       break;
                case 5: printf("array type = T_CHAR");          break;
                case 6: printf("array type = T_FLOAT");         break;
                case 7: printf("array type = T_DOUBLE");        break;
                case 8: printf("array type = T_BYTE");          break;
                case 9: printf("array type = T_SHORT");         break;
                case 10:        printf("array type = T_INT");           break;
                case 11:        printf("array type = T_LONG");          break;
        }
        break;
case 189:       m1 = m[pp++]; m2 = m[pp++];
        printf("anewarray¥t%04X¥t(Allocate new array of references to objects)", 256*m1+m2);
        k += 2; break;
case 197:       m1 = m[pp++]; m2 = m[pp++]; m3 = m[pp++];
        printf("multianewarray¥t%04X %02X\t(Allocate new multi-dimensional array)", 256*m1+m2, m3);
        k += 3; break;
case 190:       printf("arraylength¥t¥t(Get length of array)");
        break;
case 46:        printf("iaload¥t¥t(Load integer from array)");
        break;
case 47:        printf("laload¥t¥t(Load long integer from array)");
        break;
case 48:        printf("faload¥t¥t(Load single float from array)");
        break;
case 49:        printf("daload¥t¥t(Load double float from array)");
        break;
case 50:        printf("aaload¥t¥t(Load object reference from array)");
        break;
case 51:        printf("baload¥t¥t(Load signed byte from array)");
        break;
case 52:        printf("caload¥t¥t(Load character from array)");
        break;
case 53:        printf("saload¥t¥t(Load short from array)");
        break;
case 79:        printf("iastore¥t¥t(Store into integer array)");
        break;
case 80:        printf("lastore¥t¥t(Store into long integer array)");
        break;
case 81:        printf("fastore¥t¥t(Store into single float array)");
        break;
case 82:        printf("dastore¥t¥t(Store into double float array)");
        break;
case 83:        printf("aastore¥t¥t(Store into object reference array)");
        break;
case 84:        printf("bastore¥t¥t(Store into signed byte array)");
        break;
case 85:        printf("castore¥t¥t(Store into character array)");
        break;
case 86:        printf("sastore¥t¥t(Store into short array)");
        break;
case 0: printf("nop¥t¥t(Do nothing)");
        break;
case 87:        printf("pop¥t¥t(Pop top stack word)");
        break;
case 88:        printf("pop2¥t¥t(Pop top two stack words)");
        break;
case 89:        printf("dup¥t¥t(Duplicate top stack word)");
        break;
case 92:        printf("dup2¥t¥t(Duplicate top two stack words)");
        break;
case 90:        printf("dup_x1¥t¥t(Duplicate top stack word and put two down)");
        break;
case 93:        printf("dup2_x1¥t¥t(Duplicate top two stack words and put two down)");
        break;
case 91:        printf("dup_x2¥t¥t(Duplicate top stack word and put three down)");
        break;
case 94:        printf("dup2_x2¥t¥t(Duplicate top two stack words and put three down)");
        break;
```


HTMLファイル　HTML(Hyper Markup Language)によって書かれた，ホームページを規定するファイルのこと．

〈リスト8-9〉Java逆コンパイラ[rejava.c]③

```
case 95:        printf("swap¥t¥t(Swap top two stack words)");
        break;
case 96:        printf("iadd¥t¥t(Integer add)");
        break;
case 97:        printf("ladd¥t¥t(Long integer add)");
        break;
case 98:        printf("fadd¥t¥t(Single floats add)");
        break;
case 99:        printf("dadd¥t¥t(Double floats add)");
        break;
case 100:       printf("isub¥t¥t(Integer subtract)");
        break;
case 101:       printf("lsub¥t¥t(Long integer subtract)");
        break;
case 102:       printf("fsub¥t¥t(Single float subtract)");
        break;
case 103:       printf("dsub¥t¥t(Double float subtract)");
        break;
case 104:       printf("imul¥t¥t(Integer multiply)");
        break;
case 105:       printf("lmul¥t¥t(Long integer multiply)");
        break;
case 106:       printf("fmul¥t¥t(Single float multiply)");
        break;
case 107:       printf("dmul¥t¥t(Double float multiply)");
        break;
case 108:       printf("idiv¥t¥t(Integer divide)");
        break;
case 109:       printf("ldiv¥t¥t(Long integer divide)");
        break;
case 110:       printf("fdiv¥t¥t(Single float divide)");
        break;
case 111:       printf("ddiv¥t¥t(Double float divide)");
        break;
case 112:       printf("irem¥t¥t(Integer remainder)");
        break;
case 113:       printf("lrem¥t¥t(Long integer remainder)");
        break;
case 114:       printf("frem¥t¥t(Single float remainder)");
        break;
case 115:       printf("drem¥t¥t(Double float remainder)");
        break;
case 116:       printf("ineg¥t¥t(Integer negate)");
        break;
case 117:       printf("lneg¥t¥t(Long integer negate)");
        break;
case 118:       printf("fneg¥t¥t(Single float negate)");
        break;
case 119:       printf("dneg¥t¥t(Double float negate)");
        break;
case 120:       printf("ishl¥t¥t(Integer shift left)");
        break;
case 122:       printf("ishr¥t¥t(Integer arithmetic shift right)");
        break;
case 124:       printf("iushr¥t¥t(Integer logical shift right)");
        break;
case 121:       printf("lshl¥t¥t(Long integer shift left)");
        break;
case 123:       printf("lshr¥t¥t(Long integer arithmetic shift right)");
        break;
case 125:       printf("lushr¥t¥t(Long integer logical shift right)");
        break;
case 126:       printf("iand¥t¥t(Integer boolean AND)");
        break;
case 127:       printf("land¥t¥t(Long integer boolean AND)");
        break;
case 128:       printf("ior¥t¥t(Integer boolean OR)");
        break;
case 129:       printf("lor¥t¥t(Long integer boolean OR)");
        break;
case 130:       printf("ixor¥t¥t(Integer boolean XOR)");
        break;
case 131:       printf("lxor¥t¥t(Long integer boolean XOR)");
        break;
case 133:       printf("i2l¥t¥t(Integer to long integer conversion)");
        break;
case 134:       printf("i2f¥t¥t(Integer to single float)");
        break;
case 135:       printf("i2d¥t¥t(Integer to double float)");
        break;
case 136:       printf("l2i¥t¥t(Long integer to integer)");
        break;
case 137:       printf("l2f¥t¥t(Long integer to single float)");
        break;
case 138:       printf("l2d¥t¥t(Long integer to double float)");
        break;
case 139:       printf("f2i¥t¥t(Single float to integer)");
        break;
case 140:       printf("f2l¥t¥t(Single float to long integer)");
        break;
case 141:       printf("f2d¥t¥t(Single float to double float)");
        break;
case 142:       printf("d2i¥t¥t(Double float to integer)");
        break;
case 143:       printf("d2l¥t¥t(Double float to long integer)");
        break;
```


クラス・ファイル　Java言語で書かれたJavaソース・プログラムをJavaコンパイラで変換してできたオブジェクト・コードを，「バイト・コード」とともに「クラス・ファイル」とも言う．拡張子がclassであるため．

〈リスト8-9〉Java逆コンパイラ[rejava.c]④

```
case 144:       printf("d2f¥t¥t(Double float to single float)");
        break;
case 145:       printf("int2byte¥t¥t(Integer to signed byte)");
        break;
case 146:       printf("int2char¥t¥t(Integer to char)");
        break;
case 147:       printf("int2short¥t¥t(Integer to short)");
        break;
case 153:       m1 = m[pp++]; m2 = m[pp++];
        printf("ifeq¥t%04X¥t(Branch if equal to 0)", 256*m1+m2);
        k += 2; break;
case 198:       m1 = m[pp++]; m2 = m[pp++];
        printf("ifnull¥t%04X¥t(Branch if null)", 256*m1+m2);
        k += 2; break;
case 155:       m1 = m[pp++]; m2 = m[pp++];
        printf("iflt¥t%04X¥t(Branch if less than 0)", 256*m1+m2);
        k += 2; break;
case 158:       m1 = m[pp++]; m2 = m[pp++];
        printf("ifle¥t%04X¥t(Branch if less than or equal to 0)", 256*m1+m2);
        k += 2; break;
case 154:       m1 = m[pp++]; m2 = m[pp++];
        printf("ifne¥t%04X¥t(Branch if not equal to 0)", 256*m1+m2);
        k += 2; break;
case 199:       m1 = m[pp++]; m2 = m[pp++];
        printf("ifnonnull¥t%04X¥t(Branch if not null)", 256*m1+m2);
        k += 2; break;
case 157:       m1 = m[pp++]; m2 = m[pp++];
        printf("ifgt¥t%04X¥t(Branch if greater than 0)", 256*m1+m2);
        k += 2; break;
case 156:       m1 = m[pp++]; m2 = m[pp++];
        printf("ifge¥t%04X¥t(Branch if greater than or equal to 0)", 256*m1+m2);
        k += 2; break;
case 159:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_icmpeq¥t%04X¥t(Branch if integers equal)", 256*m1+m2);
        k += 2; break;
case 160:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_icmpne¥t%04X¥t(Branch if integers not equal)", 256*m1+m2);
        k += 2; break;
case 161:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_icmplt¥t%04X¥t(Branch if integer less than)", 256*m1+m2);
        k += 2; break;
case 163:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_icmpgt¥t%04X¥t(Branch if integer greater than)", 256*m1+m2);
        k += 2; break;
case 164:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_icmple¥t%04X¥t(Branch if integer less than or equal to)", 256*m1+m2);
        k += 2; break;
case 162:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_icmpge¥t%04X¥t(Branch if integer greater than or equal to)", 256*m1+m2);
        k += 2; break;
case 148:       printf("lcmp¥t¥t(Long integer compare)");
        break;
case 149:       printf("fcmpl¥t¥t(Single float compare (-1 on NaN))");
        break;
case 150:       printf("fcmpg¥t¥t(Single float compare (1 on NaN))");
        break;
case 151:       printf("dcmpl¥t¥t(Double float compare (-1 on NaN))");
        break;
case 152:       printf("dcmpg¥t¥t(Double float compare (1 on NaN))");
        break;
case 165:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_acmpeq¥t%04X¥t(Branch if object references are equal)", 256*m1+m2);
        k += 2; break;
case 166:       m1 = m[pp++]; m2 = m[pp++];
        printf("if_acmpne¥t%04X¥t()Branch if object references not equal", 256*m1+m2);
        k += 2;
case 167:       m1 = m[pp++]; m2 = m[pp++];
        printf("goto¥t%04X¥t(Branch)", 256*m1+m2);
        k += 2; break;
case 200:       m1 = m[pp++]; m2 = m[pp++];
        m3 = m[pp++]; m4 = m[pp++];
        printf("goto_w¥t%04X %04X¥t(Branch always (wide index))", 256*m1+m2, 256*m3+m4);
        k += 2; break;
case 168:       m1 = m[pp++]; m2 = m[pp++];
        printf("jsr¥t%04X¥t(Jump subroutine)", 256*m1+m2);
        k += 2; break;
case 201:       m1 = m[pp++]; m2 = m[pp++];
        m3 = m[pp++]; m4 = m[pp++];
        printf("jsr_w¥t%04X %04X¥t(Jump subroutine (wide index))", 256*m1+m2, 256*m3+m4);
        k += 2; break;
case 169:       m1 = m[pp++];
        printf("ret¥t%02X¥t(Return from subroutine)", m1);
        k += 1; break;
case 209:       m1 = m[pp++]; m2 = m[pp++];
        printf("ret_w¥t%04X¥t(Return from subroutine (wide index))", 256*m1+m2);
        k += 2; break;
case 172:       printf("ireturn¥t¥t(Return integer from function)");
        break;
case 173:       printf("lreturn¥t¥t(Return long integer from function)");
        break;
case 174:       printf("freturn¥t¥t(Return single float from function)");
        break;
case 175:       printf("dreturn¥t¥t(Return double float from function)");
        break;
case 176:       printf("areturn¥t¥t(Return object reference from function)");
        break;
```


拡張ライブラリ　機能を拡張するために「含める」と宣言しておくサービス・ライブラリのこと．あらかじめ言語体系として提供されている．

〈リスト8-9〉Java逆コンパイラ[rejava.c]⑤

```
                        case 177:       printf("return¥t¥t(Return (void) from procedure)");
                                break;
                        case 202:       printf("breakpoint¥t¥t(Stop and pass control to breakpoint handler)");
                                break;
                        case 170:       printf("tableswitch¥t¥t(Access jump table by index and jump)");
                                printf("¥n¥t¥t¥t¥t¥t¥t**********************");
                                break;
                        case 171:       printf("lookupswitch¥t¥t(Access jump table by key match and jump)");
                                printf("¥n¥t¥t¥t¥t¥t¥t**********************");
                                break;
                        case 181:       m1 = m[pp++]; m2 = m[pp++];
                                printf("putfield¥t%04X¥t(Set field in object)", 256*m1+m2);
                                k += 2; break;
                        case 180:       m1 = m[pp++]; m2 = m[pp++];
                                printf("getfield¥t%04X¥t(Fetch field from object)", 256*m1+m2);
                                k += 2; break;
                        case 179:       m1 = m[pp++]; m2 = m[pp++];
                                printf("putstatic¥t%04X¥t(Set static field in class)", 256*m1+m2);
                                k += 2; break;
                        case 178:       m1 = m[pp++]; m2 = m[pp++];
                                printf("getstatic¥t%04X¥t(Get static field from class)", 256*m1+m2);
                                k += 2; break;
                        case 182:       m1 = m[pp++]; m2 = m[pp++];
                                printf("invokevirtual¥t%04X¥t(Invoke instance method)", 256*m1+m2);
                                k += 2; break;
                        case 183:       m1 = m[pp++]; m2 = m[pp++];
                                printf("invokenonvirt¥t%04X¥t(Invoke instance method, dispatching based on compile-time type)",256*m1+m2);
                                k += 2; break;
                        case 184:       m1 = m[pp++]; m2 = m[pp++];
                                printf("invokestatic¥t%04X¥t(Invoke a class (static) method)", 256*m1+m2);
                                k += 2; break;
                        case 185:       m1 = m[pp++]; m2 = m[pp++];
                                m3 = m[pp++]; m4 = m[pp++];
                                printf("invokeinterf¥t%04X %02X %02X¥t(Invoke interface method)", 256*m1+m2, m3, m4);
                                k += 2; break;
                        case 191:       printf("athrow¥t¥t(Throw exception or error)");
                                break;
                        case 187:       m1 = m[pp++]; m2 = m[pp++];
                                printf("new¥t%04X¥t(Create new object)", 256*m1+m2);
                                k += 2; break;
                        case 192:       m1 = m[pp++]; m2 = m[pp++];
                                printf("checkcast¥t%04X¥t(Make sure object is of given type)", 256*m1+m2);
                                k += 2; break;
                        case 193:       m1 = m[pp++]; m2 = m[pp++];
                                printf("instanceof¥t%04X¥t(Determine if an object is of given type)", 256*m1+m2);
                                k += 2; break;
                        case 194:       printf("monitorenter¥t¥t(Enter monitored region of code)");
                                break;
                        case 195:       printf("monitorexit¥t¥t(Exit monitored region of code)");
                                break;
                }
                k++;
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2;
        printf("¥n¥n¥t¥t¥texception_table_length = %d", p1 );
        if( p1 != 0 ){
                printf("¥n");
                for(k=0;k<p1;k++){
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("¥n¥t¥t¥t¥t%08X  start_pc = %d , ", pp, p2 );
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("end_pc = %d , ", p2 );
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("handler_pc = %d , ", p2 );
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        if( p2 != 0 ) printf("catch_type = %s", n[ni[p2]] );
                        else printf("catch_type = all exceptions");
                }
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2;
        printf("¥n¥n¥t¥t¥tattributes_count = %d", p1 );
        if( p1 != 0 ){
                printf("¥n");
                for(k=0;k<p1;k++){
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("¥n¥t¥t¥t¥t%08X  type = %s , ", pp, n[p2] );
                        p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                        printf("attribute_length = %d¥n", p4);
                        for(l=0;l<p4;l++){
                                if((l%20)==0) printf("¥n¥t¥t¥t¥t¥t");
                                printf("%02X ", m[pp++] );
                        }
                }
        }
}
```


Runnableプログラム　Javaでは「実行し続ける」という処理を「run」と呼ぶ，これを組み込むことをあらかじめ「Runnable」と宣言しておく．

〈リスト8-9〉Java逆コンパイラ[rejava.c]⑥

```
void access_flag_printf(int para){
        switch(para){
                case 0x0001:    printf("ACC_PUBLIC");   break;
                case 0x0002:    printf("ACC_PRIVATE");  break;
                case 0x0004:    printf("ACC_PROTECTED");        break;
                case 0x0008:    printf("ACC_STATIC");   break;
                case 0x0010:    printf("ACC_FINAL");    break;
                case 0x0020:    printf("ACC_SYNCHRONIZED");     break;
                case 0x0040:    printf("ACC_VOLATILE"); break;
                case 0x0080:    printf("ACC_TRANSIENT");        break;
                case 0x0100:    printf("ACC_NATIVE");   break;
                case 0x0200:    printf("ACC_INTERFACE");        break;
                case 0x0400:    printf("ACC_ABSTRACT"); break;
                default:        printf("( no flag )");  break;
        }
}

void main(int argc,char **argv){
        int d, i, j, k, ct, p0, p1, p2, p3, address=0;
        long pp=0, count=0, p4, p5;
        unsigned char ss, st[20], d0, d1, d2, d3;
        st[16]=0;
        if( argc != 2 ){ printf("¥n... target file name missing (;_;) ...¥n¥n"); exit(1); }
        else if( (fp=fopen( argv[1], "rb" ))==NULL ){ printf("¥n... target file [ %s ] is not found (;_;) ...¥n¥n", argv[1]); exit(1); }
        printf("¥nTarget Java File Name = %s¥n", argv[1]);
        while( (d = fgetc(fp)) >= 0 ){
                if( ( address % 16 ) == 0 ){ printf("¥n¥t%08X  :  ", address); ct = 0; }
                ss = d & 0xff; printf("%02X  ",ss); m[count++] = ss;
                if( ( address % 16 ) == 7 ) printf("-  ");
                if( ( ss > 0x1f ) && ( ss < 0x7f ) ) st[ct++] = ss;
                else st[ct++] = '.';
                if( ( address++ % 16 ) == 15 ) printf("  %s", st);
        }
        fclose(fp);
        printf("¥n¥nTotal File Length = %d bytes.", count);
        p0 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp]));
        printf("¥n¥n¥t%08X  Magic Code [CAFEBABE] = %08X", pp, p0 );
        pp += 4;
        if( p0 != 0xcafebabe ){ printf("¥n¥n... target file is not Java (;_;) ...¥n¥n"); exit(1); }
        else printf("  ---  OK (^_^)");
        printf("¥n¥n¥t%08X  Version : major version %d , minor version %d", pp, m[pp+3]+256*m[pp+2], m[pp+1]+256*m[pp] );
        pp += 4;
        p1 = m[pp+1]+256*m[pp]; printf("¥n¥n¥t%08X  Constant Pool : total number = %d¥n", pp, p1 ); pp += 2;
        for(i=1;i<p1;i++){
                printf("¥n¥t¥t%08X   constant_pool[%d] = ", pp, i);
                d0 = m[pp++]; printf("<%X> ",d0);
                switch(d0){
                        case 7: p2 = m[pp+1]+256*m[pp]; pp += 2; ni[i] = p2;
                                printf("CONSTANT_Class , name_index = %d", p2);
                                break;
                        case 9: p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_Fieldref , class_index = %d", p2);
                                printf(" , name_and_type_index = %d", p3);
                                break;
                        case 10:        p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_Methodref , class_index = %d", p2);
                                printf(" , name_and_type_index = %d", p3);
                                break;
                        case 11:        p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_InterfaceMethodref , class_index = %d", p2);
                                printf(" , name_and_type_index = %d", p3);
                                break;
                        case 8: p2 = m[pp+1]+256*m[pp]; pp += 2; ni[i] = p2;
                                printf("CONSTANT_String , name_index = %d", p2);
                                break;
                        case 3: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Integer , value = %d", p4);
                                break;
                        case 4: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Float , value = %d", p4);
                                break;
                        case 5: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                p5 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Float , high = %d , low = %d", p4, p5); i++;
                                break;
                        case 6: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                p5 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Double , high = %d , low = %d", p4, p5); i++;
                                break;
                        case 12:        p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_NameAndType , name_index = %d", p2); ni[i] = p2;
                                printf(" , signature_index = %d", p3);
                                break;
                        case 1: case 2:
                                p2 = m[pp+1]+256*m[pp]; pp += 2;
                                if(d0==1) printf("CONSTANT_Utf8 , ");
                                else printf("CONSTANT_Unicode , ");
                                k = 0;
                                for(j=0;j<p2;j++){
                                        d1 = m[pp++];
                                        if((d1&0x80)==0) n[i][k++] = d1;
                                        else{
```


メソッド．C言語の「関数」に相当する，ソフトウェアの処理の小さな単位．Javaではメソッドと呼ぶ．

〈リスト8-9〉 Java逆コンパイラ[rejava.c]⑦

```
                                        d2 = m[pp++];
                                        if((d1&0xe0)==0xc0){
                                                n[i][k++] = (64*(d1&0x1f)+(d2&0x3f))/256;
                                                n[i][k++] = (64*(d1&0x1f)+(d2&0x3f))%256;
                                        }
                                        else{
                                                d3 = m[pp++];
                                                n[i][k++] = (256*16*(d1&0x0f)+64*(d2&0x3f)+(d3&0x3f))/256;
                                                n[i][k++] = (256*16*(d1&0x0f)+64*(d2&0x3f)+(d3&0x3f))%256;
                                        }
                                }
                        }
                        n[i][k] = 0; printf("Data = %s",n[i]);
                        break;
                default:        printf("¥nFormat Error !!!¥n");
                        break;
        }
    }
    p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%08X  Access Flag = %04X --- ", pp-2, p1 );
    access_flag_printf( p1 );
    p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%08X  This Class = %s", pp-2, n[ni[p1]] );
    p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%08X  Super Class = %s", pp-2, n[ni[p1]] );
    p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%08X  Interface Count = %d", pp-2, p1 );
    if( p1 != 0 ){
        printf("¥n");
        for(i=0;i<p1;i++){
            p2 = m[pp+1]+256*m[pp]; pp += 2;
            printf("¥n¥t¥t%08X  interface[%d] : Name = %s", pp-2, i, n[ni[p2]] );
        }
    }
    p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%08X  Fields Count = %d¥n", pp-2, p1 );
    for(i=0;i<p1;i++){
        printf("¥n¥t¥t%08X   field[%d] : ", pp, i);
        p2 = m[pp+1]+256*m[pp]; pp += 2; access_flag_printf( p2 );
        p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Name = %s", n[p2] );
        p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Signature = %s", n[p2] );
        p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Attribute = %d", p2 );
        if( p2 != 0 ){
            printf("¥n");
            for(j=0;j<p2;j++){
                p3 = m[pp+1]+256*m[pp]; pp += 2;
                printf("¥n¥t¥t¥ttype = %s , ", n[p3] );
                p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                printf("length = %d¥n", p4);
                for(k=0;k<p4;k++){
                    if((k%25)==0) printf("¥n¥t¥t¥t¥t");
                    printf("%02X ", m[pp++] );
                }
            }
        }
    }
    p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%08X  Methods Count = %d", pp-2, p1 );
    for(i=0;i<p1;i++){
        printf("¥n¥n¥t¥t%08X   method[%d] : ", pp, i);
        p2 = m[pp+1]+256*m[pp]; pp += 2; access_flag_printf( p2 );
        p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Name = %s", n[p2] );
        p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Signature = %s", n[p2] );
        p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Attribute = %d", p2 );
        if( p2 != 0 ){
            printf("¥n");
            for(j=0;j<p2;j++){
                p3 = m[pp+1]+256*m[pp]; pp += 2;
                printf("¥n¥t¥t¥ttype = %s , ", n[p3] );
                p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                printf("length = %d", p4);
                p5 = pp; pp += p4;
                if( strcmp( n[p3], "Code" ) == 0 ) byte_code_printf( p5, p4 );
                else if( strcmp( n[p3], "Exceptions" ) == 0 ) byte_code_printf( p5, p4 );
            }
        }
    }
    p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%08X  Attributes Count = %d", pp-2, p1 );
    if( p1 != 0 ){
        for(i=0;i<p1;i++){
            p2 = m[pp+1]+256*m[pp]; pp += 2;
            printf("¥n¥n¥t¥t%08X  attribute[%d] : type = %s , ", pp-2, i, n[p2] );
            p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
            p5 = pp;
            printf("length = %d¥n", p4);
            for(k=0;k<p4;k++){
                if((k%25)==0) printf("¥n¥t¥t¥t¥t");
                printf("%02X ", m[pp++] );
            }
            if( ( p4 == 2 ) && ( strcmp( n[p2], "SourceFile" ) == 0 ) ){
                p3 = m[p5+1]+256*m[p5];
                printf("¥n¥n¥t¥t¥tSource File Name = %s", n[p3] );
            }
        }
    }
    printf("¥n¥n..... (^_^) Java analyze completely finished. (^_^)¥n¥n");
    exit(0);
}
```


Xウィンドウ・システム　Unix環境でテキスト・ベースでなくグラフィック環境を実現するためのソフトウェア体系．MacやWindowsもこれに似ている．

〈リスト8-11〉sample.classのrejava出力①

```
Target Java File Name = sample.class

        00000000  :  CA  FE  BA  BE  00  03  00  2D  -  00  B9  08  00  AC  08  00  B4    .......-........
        00000010  :  07  00  96  07  00  8C  07  00  -  88  07  00  9C  07  00  7A  07    ..............z.
        00000020  :  00  67  07  00  91  07  00  9E  -  07  00  A5  07  00  9D  07  00    .g..............
        00000030  :  AD  07  00  84  07  00  A0  07  -  00  92  07  00  89  0A  00  0F    ................
        00000040  :  00  53  09  00  10  00  5E  0A  -  00  07  00  44  09  00  0F  00    .S....^....D....
        00000050  :  57  0A  00  09  00  55  0A  00  -  09  00  4A  0A  00  07  00  5A    W....U....J....Z
        00000060  :  0A  00  0D  00  54  0A  00  05  -  00  50  0A  00  11  00  39  09    ....T....P....9.
        00000070  :  00  0F  00  3F  0A  00  0F  00  -  41  0A  00  07  00  59  0A  00    ...?....A....Y..
        00000080  :  09  00  3B  09  00  0F  00  4D  -  0A  00  11  00  56  0A  00  11    ..;....M....V...
        00000090  :  00  3D  09  00  0F  00  5B  0A  -  00  06  00  3C  09  00  0F  00    .=....[....<....
        000000A0  :  5F  0A  00  0F  00  42  0A  00  -  0F  00  4C  0A  00  05  00  52    _....B....L....R
        000000B0  :  0A  00  05  00  3E  0A  00  0A  -  00  40  09  00  0F  00  43  0A    ....>....@....C.
        000000C0  :  00  0D  00  3A  0A  00  0D  00  -  4F  09  00  0F  00  38  0A  00    ...:....O....8..
        000000D0  :  0A  00  51  09  00  0F  00  5D  -  09  00  08  00  5C  0A  00  09    ..Q....]....\...
        000000E0  :  00  4B  09  00  0F  00  58  0A  -  00  11  00  54  09  00  0F  00    .K....X....T....
        000000F0  :  49  0A  00  07  00  54  09  00  -  10  00  4E  0C  00  9F  00  6C    I....T....N....l
        00000100  :  0C  00  AB  00  9B  0C  00  8A  -  00  7D  0C  00  7B  00  8B  0C    ........}..{....
        00000110  :  00  93  00  8F  0C  00  AB  00  -  7E  0C  00  86  00  82  0C  00    ........~.......
        00000120  :  79  00  AA  0C  00  80  00  7C  -  0C  00  99  00  90  0C  00  87    y......|........
        00000130  :  00  6F  0C  00  9A  00  AA  0C  -  00  98  00  8D  05  00  00  00    .o..............
        00000140  :  00  00  00  03  E8  06  40  8F  -  38  00  00  00  00  00  0C  00    ......@.8.......
        00000150  :  A1  00  AA  0C  00  69  00  6D  -  0C  00  94  00  B7  0C  00  6E    .....i.m.......n
        00000160  :  00  6F  0C  00  B8  00  B6  0C  -  00  B0  00  AA  0C  00  85  00    .o..............
        00000170  :  78  0C  00  75  00  A7  0C  00  -  8E  00  68  0C  00  76  00  8B    x..u.....h..v...
        00000180  :  0C  00  97  00  90  0C  00  69  -  00  8B  0C  00  72  00  8B  0C    .......i....r...
        00000190  :  00  70  00  7F  0C  00  A4  00  -  AA  0C  00  81  00  AA  0C  00    .p..............
        000001A0  :  6A  00  8D  0C  00  B3  00  8D  -  0C  00  77  00  AA  0C  00  74    j.........w....t
        000001B0  :  00  B5  0C  00  A6  00  AA  0C  -  00  A3  00  AA  0C  00  A2  00    ................
        000001C0  :  B2  06  40  12  00  00  00  00  -  00  00  06  3F  F8  00  00  00    ..@........?....
        000001D0  :  00  00  00  06  3F  ED  C2  8F  -  5C  28  F5  C3  01  00  16  28    ....?...\(.....(
        000001E0  :  4C  6A  61  76  61  2F  61  77  -  74  2F  47  72  61  70  68  69    Ljava/awt/Graphi
        000001F0  :  63  73  3B  29  56  01  00  0E  -  6A  61  76  61  2F  61  77  74    cs;)V...java/awt
        00000200  :  2F  43  6F  6C  6F  72  01  00  -  07  28  49  49  49  49  29  56    /Color...(IIII)V
        00000210  :  01  00  06  3C  69  6E  69  74  -  3E  01  00  08  67  65  74  48    ...<init>...getH
        00000220  :  6F  75  72  73  01  00  0A  53  -  6F  75  72  63  65  46  69  6C    ours...SourceFil
        00000230  :  65  01  00  04  5B  5B  5B  49  -  01  00  17  28  4C  6A  61  76    e...[[[I...(Ljav
        00000240  :  61  2F  6C  61  6E  67  2F  52  -  75  6E  6E  61  62  6C  65  3B    a/lang/Runnable;
        00000250  :  29  56  01  00  08  64  6F  74  -  5F  64  72  61  77  01  00  18    )V...dot_draw...
        00000260  :  28  4C  6A  61  76  61  2F  61  -  77  74  2F  47  72  61  70  68    (Ljava/awt/Graph
        00000270  :  69  63  73  3B  49  49  29  56  -  01  00  08  74  6F  53  74  72    ics;II)V...toStr
        00000280  :  69  6E  67  01  00  0A  45  78  -  63  65  70  74  69  6F  6E  73    ing...Exceptions
        00000290  :  01  00  04  73  74  6F  70  01  -  00  0F  4C  69  6E  65  4E  75    ...stop...LineNu
        000002A0  :  6D  62  65  72  54  61  62  6C  -  65  01  00  05  62  6C  61  63    mberTable...blac
        000002B0  :  6B  01  00  04  73  69  7A  65  -  01  00  07  72  65  70  61  69    k...size...repai
        000002C0  :  6E  74  01  00  02  73  7A  01  -  00  32  28  4C  6A  61  76  61    nt...sz..2(Ljava
        000002D0  :  2F  6E  65  74  2F  55  52  4C  -  3B  4C  6A  61  76  61  2F  6C    /net/URL;Ljava/l
        000002E0  :  61  6E  67  2F  53  74  72  69  -  6E  67  3B  29  4C  6A  61  76    ang/String;)Ljav
        000002F0  :  61  2F  61  77  74  2F  49  6D  -  61  67  65  3B  01  00  02  73    a/awt/Image;...s
        00000300  :  74  01  00  0E  6A  61  76  61  -  2F  75  74  69  6C  2F  44  61    t...java/util/Da
        00000310  :  74  65  01  00  05  73  74  61  -  72  74  01  00  35  28  4C  6A    te...start..5(Lj
        00000320  :  61  76  61  2F  61  77  74  2F  -  49  6D  61  67  65  3B  49  49    ava/awt/Image;II
        00000330  :  49  49  4C  6A  61  76  61  2F  -  61  77  74  2F  69  6D  61  67    IILjava/awt/imag
        00000340  :  65  2F  49  6D  61  67  65  4F  -  62  73  65  72  76  65  72  3B    e/ImageObserver;
        00000350  :  29  5A  01  00  10  28  29  4C  -  6A  61  76  61  2F  6E  65  74    )Z...()Ljava/net
        00000360  :  2F  55  52  4C  3B  01  00  2C  -  28  4C  6A  61  76  61  2F  6C    /URL;..,(Ljava/l
        00000370  :  61  6E  67  2F  53  74  72  69  -  6E  67  3B  29  4C  6A  61  76    ang/String;)Ljav
        00000380  :  61  2F  6C  61  6E  67  2F  53  -  74  72  69  6E  67  42  75  66    a/lang/StringBuf
        00000390  :  66  65  72  3B  01  00  14  28  -  29  4C  6A  61  76  61  2F  6C    fer;...()Ljava/l
        000003A0  :  61  6E  67  2F  53  74  72  69  -  6E  67  3B  01  00  09  64  72    ang/String;...dr
        000003B0  :  61  77  49  6D  61  67  65  01  -  00  09  63  6F  6C  6F  72  5F    awImage...color_
        000003C0  :  73  65  6C  01  00  13  28  4C  -  6A  61  76  61  2F  61  77  74    sel...(Ljava/awt
        000003D0  :  2F  43  6F  6C  6F  72  3B  29  -  56  01  00  04  69  6E  69  74    /Color;)V...init
        000003E0  :  01  00  03  5B  5B  49  01  00  -  08  67  65  74  49  6D  61  67    ...[[I...getImag
        000003F0  :  65  01  00  0D  73  65  74  42  -  61  63  6B  67  72  6F  75  6E    e...setBackgroun
        00000400  :  64  01  00  08  63  68  61  72  -  5F  64  6F  74  01  00  12  6A    d...char_dot...j
        00000410  :  61  76  61  2F  61  77  74  2F  -  43  6F  6D  70  6F  6E  65  6E    ava/awt/Componen
        00000420  :  74  01  00  16  6A  61  76  61  -  2F  6C  61  6E  67  2F  53  74    t...java/lang/St
        00000430  :  72  69  6E  67  42  75  66  66  -  65  72  01  00  0F  67  65  74    ringBuffer...get
        00000440  :  44  6F  63  75  6D  65  6E  74  -  42  61  73  65  01  00  03  28    DocumentBase...(
        00000450  :  29  56  01  00  02  5B  49  01  -  00  03  28  29  49  01  00  09    )V...[I...()I...
        00000460  :  63  6C  65  61  72  52  65  63  -  74  01  00  03  28  29  44  01    clearRect...()D.
        00000470  :  00  19  28  4C  6A  61  76  61  -  2F  61  77  74  2F  47  72  61    ..(Ljava/awt/Gra
        00000480  :  70  68  69  63  73  3B  49  49  -  49  29  56  01  00  10  6A  61    phics;III)V...ja
        00000490  :  76  61  2F  6C  61  6E  67  2F  -  54  68  72  65  61  64  01  00    va/lang/Thread..
        000004A0  :  12  6A  61  76  61  2F  61  77  -  74  2F  44  69  6D  65  6E  73    .java/awt/Dimens
        000004B0  :  69  6F  6E  01  00  06  72  61  -  6E  64  6F  6D  01  00  05  73    ion...random...s
        000004C0  :  6C  65  65  70  01  00  03  72  -  75  6E  01  00  13  6A  61  76    leep...run...jav
        000004D0  :  61  2F  6C  61  6E  67  2F  45  -  78  63  65  70  74  69  6F  6E    a/lang/Exception
        000004E0  :  01  00  0A  63  6C  6F  63  6B  -  5F  64  69  73  70  01  00  0A    ...clock_disp...
        000004F0  :  67  65  74  53  65  63  6F  6E  -  64  73  01  00  09  63  68  61    getSeconds...cha
        00000500  :  72  5F  64  69  73  70  01  00  -  02  79  79  01  00  1B  28  49    r_disp...yy...(I
        00000510  :  29  4C  6A  61  76  61  2F  6C  -  61  6E  67  2F  53  74  72  69    )Ljava/lang/Stri
```

Unicode　1バイトだけでなく2バイト以上のデータを統一的に実現するための規約．2バイト・データの日本語の場合，Unicodeでは3バイトかかってようやく表現されるために，Javaの世界では日本語は非常に不利である．

〈リスト8-11〉sample.classのrejava出力②

```
00000520 : 6E 67 42 75 66 66 65 72 - 3B 01 00 0E 6A 61 76 61   ngBuffer;...java
00000530 : 2F 6C 61 6E 67 2F 4D 61 - 74 68 01 00 12 6A 61 76   /lang/Math...jav
00000540 : 61 2F 6C 61 6E 67 2F 52 - 75 6E 6E 61 62 6C 65 01   a/lang/Runnable.
00000550 : 00 11 6A 61 76 61 2F 61 - 77 74 2F 47 72 61 70 68   ..java/awt/Graph
00000560 : 69 63 73 01 00 01 6D 01 - 00 06 73 61 6D 70 6C 65   ics...m...sample
00000570 : 01 00 01 6A 01 00 01 69 - 01 00 05 77 69 64 74 68   ...j...i...width
00000580 : 01 00 04 6D 6F 64 65 01 - 00 0E 6A 61 76 61 2F 61   ...mode...java/a
00000590 : 77 74 2F 49 6D 61 67 65 - 01 00 02 78 78 01 00 16   wt/Image...xx...
000005A0 : 28 29 4C 6A 61 76 61 2F - 61 77 74 2F 44 69 6D 65   ()Ljava/awt/Dime
000005B0 : 6E 73 69 6F 6E 3B 01 00 - 05 70 61 69 6E 74 01 00   nsion;...paint..
000005C0 : 0D 43 6F 6E 73 74 61 6E - 74 56 61 6C 75 65 01 00   .ConstantValue..
000005D0 : 01 49 01 00 06 61 70 70 - 65 6E 64 01 00 04 2E 67   .I...append....g
000005E0 : 69 66 01 00 12 6A 61 76 - 61 2F 61 70 70 6C 65 74   if...java/applet
000005F0 : 2F 41 70 70 6C 65 74 01 - 00 04 43 6F 64 65 01 00   /Applet...Code..
00000600 : 0B 73 61 6D 70 6C 65 2E - 6A 61 76 61 01 00 06 68   .sample.java...h
00000610 : 65 69 67 68 74 01 00 0E - 4C 6F 63 61 6C 56 61 72   eight...LocalVar
00000620 : 69 61 62 6C 65 73 01 00 - 11 5B 4C 6A 61 76 61 2F   iables...[Ljava/
00000630 : 61 77 74 2F 49 6D 61 67 - 65 3B 01 00 0A 67 65 74   awt/Image;...get
00000640 : 4D 69 6E 75 74 65 73 01 - 00 0A 6A 61 76 61 2F 67   Minutes...java/g
00000650 : 69 66 2F 62 01 00 10 4C - 6A 61 76 61 2F 61 77 74   if/b...Ljava/awt
00000660 : 2F 43 6F 6C 6F 72 3B 01 - 00 12 4C 6A 61 76 61 2F   /Color;...Ljava/
00000670 : 6C 61 6E 67 2F 54 68 72 - 65 61 64 3B 01 00 04 28   lang/Thread;...(
00000680 : 4A 29 56 01 00 04 66 6C - 6F 77 00 01 00 0F 00 0D   J)V...flow......
00000690 : 00 01 00 0C 00 0A 00 00 - 00 A1 00 AA 00 00 00 00   ................
000006A0 : 00 81 00 AA 00 00 00 00 - 00 A6 00 AA 00 00 00 00   ................
000006B0 : 00 9A 00 AA 00 00 00 00 - 00 77 00 AA 00 00 00 00   .........w......
000006C0 : 00 79 00 AA 00 00 00 00 - 00 A4 00 AA 00 00 00 00   .y..............
000006D0 : 00 A2 00 B2 00 00 00 02 - 00 B8 00 B6 00 00 00 00   ................
000006E0 : 00 9F 00 6C 00 00 00 0A - 00 01 00 83 00 8B 00 01   ...l............
000006F0 : 00 AE 00 00 00 6A 00 06 - 00 03 00 00 00 3E 2A B6   .....j.......>*.
00000700 : 00 2C 4C 04 3D A7 00 29 - 2A B4 00 25 1C 2A 2B BB   .,L.=..)*..%.*+.
00000710 : 00 11 59 B7 00 34 12 02 - B6 00 22 1C B6 00 1B 12   ..Y..4....".....
00000720 : 01 B6 00 22 B6 00 21 B6 - 00 2D 53 84 02 01 1C 10   ..."..!..-S.....
00000730 : 0E A4 FF D7 2A B2 00 31 - B6 00 29 B1 00 00 00 01   ....*..1..).....
00000740 : 00 73 00 00 00 1A 00 06 - 00 00 00 19 00 05 00 1A   .s..............
00000750 : 00 0A 00 1B 00 2D 00 1A - 00 36 00 1D 00 3D 00 18   .....-...6...=..
00000760 : 00 01 00 7B 00 8B 00 01 - 00 AE 00 00 00 34 00 04   ...{.........4..
00000770 : 00 01 00 00 00 14 2A BB - 00 09 59 2A B7 00 17 B5   ......*...Y*....
00000780 : 00 20 2A B4 00 20 B6 00 - 1F B1 00 00 00 01 00 73   . *.. .........s
00000790 : 00 00 00 0E 00 03 00 00 - 00 21 00 0C 00 22 00 13   .........!..."..
000007A0 : 00 20 00 01 00 95 00 8B - 00 01 00 AE 00 00 00 4D   . .............M
000007B0 : 00 03 00 01 00 00 00 19 - 14 00 45 B8 00 32 2A B6   ..........E..2*.
000007C0 : 00 28 A7 FF F6 57 2A 59 - B4 00 35 04 60 B5 00 35   .(...W*Y..5.`..5
000007D0 : B1 00 01 00 00 00 0D 00 - 0D 00 03 00 01 00 73 00   ..............s.
000007E0 : 00 00 1A 00 06 00 00 00 - 26 00 00 00 28 00 06 00   ........&...(...
000007F0 : 29 00 0A 00 27 00 0D 00 - 2C 00 18 00 25 00 01 00   )...'...,...%...
00000800 : 72 00 8B 00 01 00 AE 00 - 00 00 24 00 01 00 01 00   r.........$.....
00000810 : 00 00 08 2A B4 00 20 B6 - 00 16 B1 00 00 00 01 00   ...*.. .........
00000820 : 73 00 00 00 0A 00 02 00 - 00 00 30 00 07 00 2F 00   s.........0.../.
00000830 : 01 00 A8 00 66 00 01 00 - AE 00 00 00 6D 00 05 00   ....f.......m...
00000840 : 07 00 00 00 39 2A B6 00 - 1A 4D 2B 03 03 2C B4 00   ....9*...M+..,..
00000850 : 13 2C B4 00 37 B6 00 2F - BB 00 07 59 B7 00 36 4E   .,..7../...Y..6N
00000860 : 2D B6 00 1E 36 04 2D B6 - 00 18 36 05 2D B6 00 14   -...6.-...6.-...
00000870 : 36 06 2A 2B 15 04 15 05 - 15 06 B6 00 12 B1 00 00   6.*+............
00000880 : 00 01 00 73 00 00 00 22 - 00 08 00 00 00 34 00 05   ...s..."......4..
00000890 : 00 35 00 13 00 36 00 1B - 00 37 00 21 00 38 00 27   .5...6...7.!.8.'
000008A0 : 00 39 00 2D 00 3A 00 38 - 00 33 00 00 00 97 00 90   .9.-.:.8.3......
000008B0 : 00 01 00 AE 00 00 01 F8 - 00 05 00 0A 00 00 01 8C   ................
000008C0 : 04 36 09 2A B4 00 2B 36 - 06 2A B4 00 15 99 00 32   .6.*..+6.*.....2
000008D0 : 2A B4 00 33 36 09 2A 59 - B4 00 23 06 60 B5 00 23   *..36.*Y..#.`..#
000008E0 : 2A 59 B4 00 1C 05 60 B5 - 00 1C 2A 59 B4 00 30 10   *Y....`...*Y..0.
000008F0 : 28 64 B5 00 30 2A 59 B4 - 00 2B 08 64 B5 00 2B 2A   (d..0*Y..+.d..+*
00000900 : B4 00 30 2A B4 00 1C 10 - 1C 68 60 36 05 15 04 10   ..0*.....h`6....
00000910 : 0A 6C 36 07 2A 2B 15 05 - 15 06 15 07 B6 00 1D 2A   .l6.*+.........*
00000920 : B4 00 30 2A B4 00 1C 10 - 22 68 60 36 05 15 04 10   ..0*....."h`6....
00000930 : 0A 70 36 08 2A 2B 15 05 - 15 06 15 08 B6 00 1D 2A   .p6.*+.........*
00000940 : B4 00 30 2A B4 00 1C 10 - 0E 68 60 36 05 1D 10 0A   ..0*.....h`6....
00000950 : 6C 36 07 2A 2B 15 05 15 - 06 15 07 B6 00 1D 2A B4   l6.*+.........*.
00000960 : 00 30 2A B4 00 1C 10 14 - 68 60 36 05 1D 10 0A 70   .0*.....h`6....p
00000970 : 36 07 2A 2B 15 05 15 06 - 15 07 B6 00 1D 2A B4 00   6.*+.........*..
00000980 : 30 36 05 1C 10 0A 6C 36 - 07 2A 2B 15 05 15 06 15   06....l6.*+.....
00000990 : 07 B6 00 1D 2A B4 00 30 - 2A B4 00 1C 10 06 68 60   ....*..0*.....h`
000009A0 : 36 05 1C 10 0A 70 36 07 - 2A 2B 15 05 15 06 15 07   6....p6.*+......
000009B0 : B6 00 1D 2A B4 00 30 2A - B4 00 1C 10 0C 68 60 36   ...*..0*.....h`6
000009C0 : 05 2A 2B 15 05 15 06 B6 - 00 26 2A B4 00 30 2A B4   .*+......&*..0*.
000009D0 : 00 1C 10 1A 68 60 36 05 - 2A 2B 15 05 15 06 B6 00   ....h`6.*+......
000009E0 : 26 2A B4 00 15 99 00 32 - 2A 15 09 B5 00 33 2A 59   &*.....2*....3*Y
000009F0 : B4 00 23 06 64 B5 00 23 - 2A 59 B4 00 1C 05 64 B5   ..#.d..#*Y....d.
00000A00 : 00 1C 2A 59 B4 00 30 10 - 28 60 B5 00 30 2A 59 B4   ..*Y..0.(`..0*Y.
00000A10 : 00 2B 08 60 B5 00 2B 15 - 08 10 09 A0 00 1B 2A 59   .+.`..+.......*Y
00000A20 : B4 00 33 04 60 B5 00 33 - 2A B4 00 33 10 0E A4 00   ..3.`..3*..3....
00000A30 : 08 2A 04 B5 00 33 B8 00 - 24 14 00 64 97 9E 00 09   .*...3..$..d....
00000A40 : 2A 04 B5 00 15 B1 2A 03 - B5 00 15 B1 00 00 00 01   *.....*.........
00000A50 : 00 73 00 00 00 5A 00 16 - 00 00 00 3E 00 03 00 3F   .s...Z.....>...?
```


オペランド・スタック　命令である「オペコード」とともに，対象となるデータである「オペランド」を保持しておくためのスタック．

〈リスト8-11〉sample.classのrejava出力③

```
00000A60 : 00 09 00 40 00 10 00 41 - 00 16 00 42 00 3F 00 44  ...@...A...B.?.D
00000A70 : 00 5F 00 45 00 7F 00 46 - 00 9E 00 47 00 BD 00 48  ._.E...F...G...H
00000A80 : 00 D4 00 49 00 F3 00 4A - 01 0A 00 4B 01 21 00 4C  ...I...J...K.!.L
00000A90 : 01 28 00 4D 01 2E 00 4E - 01 57 00 50 01 5E 00 51  .(.M...N.W.P.^.Q
00000AA0 : 01 68 00 52 01 76 00 54 - 01 86 00 55 01 8B 00 3D  .h.R.v.T...U...=
00000AB0 : 00 00 00 87 00 6F 00 01 - 00 AE 00 00 00 45 00 08  .....o.......E..
00000AC0 : 00 04 00 00 00 25 2A 2B - 1C 1D 2A B4 00 1C 87 14  .....%*+..*.....
00000AD0 : 00 62 6B 8E 60 B6 00 27 - 2A 2B 1C 1D 2A B4 00 1C  .bk.`..'*+..*...
00000AE0 : 87 14 00 60 6B 8E 60 B6 - 00 27 B1 00 00 00 01 00  ...`k.`..'......
00000AF0 : 73 00 00 00 0E 00 03 00 - 00 00 59 00 12 00 5A 00  s.........Y...Z.
00000B00 : 24 00 58 00 00 00 6E 00 - 6F 00 01 00 AE 00 00 00  $.X...n.o.......
00000B10 : 63 00 07 00 05 00 00 00 - 37 2A B4 00 15 9A 00 0C  c.......7*......
00000B20 : 2A B4 00 33 36 04 A7 00 - 12 04 B8 00 24 14 00 47  *..36.......$..G
00000B30 : 6B 8E 10 0E 70 60 36 04 - 2B 2A B4 00 25 15 04 32  k...p`6.+*..%..2
00000B40 : 1C 1D 2A B4 00 23 2A B4 - 00 23 2A B6 00 2A 57 B1  ..*..#*..#*..*W.
00000B50 : 00 00 00 01 00 73 00 00 - 00 1A 00 06 00 00 00 5F  .....s........_
00000B60 : 00 07 00 60 00 0D 00 5F - 00 10 00 63 00 1F 00 65  ...`..._...c...e
00000B70 : 00 36 00 5D 00 00 00 99 - 00 90 00 01 00 AE 00 00  .6.]............
00000B80 : 00 78 00 06 00 07 00 00 - 00 48 03 36 05 A7 00 3E  .x.......H.6...>
00000B90 : 03 36 06 A7 00 2E 2A B4 - 00 2E 15 04 32 15 06 32  .6....*.....2..2
00000BA0 : 15 05 2E 04 A0 00 1A 2A - 2B 1C 2A B4 00 1C 15 05  .......*+.*.....
00000BB0 : 68 60 1D 2A B4 00 1C 15 - 06 68 60 B6 00 27 84 06  h`.*.....h`..'..
00000BC0 : 01 15 06 10 07 A1 FF D1 - 84 05 01 15 05 08 A1 FF  ................
00000BD0 : C2 B1 00 00 00 01 00 73 - 00 00 00 1E 00 07 00 00  .......s........
00000BE0 : 00 69 00 06 00 6A 00 0C - 00 6B 00 1D 00 6C 00 34  .i...j...k...l.4
00000BF0 : 00 6A 00 3E 00 69 00 47 - 00 68 00 01 00 69 00 8B  .j.>.i.G.h...i..
00000C00 : 00 01 00 AE 00 00 08 2B - 00 0B 00 01 00 00 07 B3  .......+........
00000C10 : 2A B7 00 19 2A 04 B5 00 - 33 2A 10 46 B5 00 30 2A  *...*...3*.F..0*
00000C20 : 10 19 B5 00 2B 2A 10 0A - B5 00 23 2A 10 0C B5 00  ....+*....#*....
00000C30 : 1C 2A 04 B5 00 15 2A 10 - 0F BD 00 0B B5 00 25 2A  .*....*.......%*
00000C40 : 10 0A BD 00 0E 59 03 10 - 07 BD 00 04 59 03 08 BC  .....Y......Y...
00000C50 : 0A 59 03 03 4F 59 04 04 - 4F 59 05 04 4F 59 06 04  .Y..OY..OY..OY..
00000C60 : 4F 59 07 03 4F 53 59 04 - 08 BC 0A 59 03 04 4F 59  OY..OSY....Y..OY
00000C70 : 04 03 4F 59 05 03 4F 59 - 06 03 4F 59 07 04 4F 53  ..OY..OY..OY..OS
00000C80 : 59 05 08 BC 0A 59 03 04 - 4F 59 04 03 4F 59 05 03  Y....Y..OY..OY..
00000C90 : 4F 59 06 03 4F 59 07 04 - 4F 53 59 06 08 BC 0A 59  OY..OY..OSY....Y
00000CA0 : 03 04 4F 59 04 03 4F 59 - 05 03 4F 59 06 03 4F 59  ..OY..OY..OY..OY
00000CB0 : 07 04 4F 53 59 07 08 BC - 0A 59 03 04 4F 59 04 03  ..OSY....Y..OY..
00000CC0 : 4F 59 05 03 4F 59 06 03 - 4F 59 07 04 4F 53 59 08  OY..OY..OY..OSY.
00000CD0 : 08 BC 0A 59 03 04 4F 59 - 04 03 4F 59 05 03 4F 59  ...Y..OY..OY..OY
00000CE0 : 06 03 4F 59 07 04 4F 53 - 59 10 06 08 BC 0A 59 03  ..OY..OSY.....Y.
00000CF0 : 03 4F 59 04 04 4F 59 05 - 04 4F 59 06 04 4F 59 07  .OY..OY..OY..OY.
00000D00 : 03 4F 53 53 59 04 10 07 - BD 00 04 59 03 08 BC 0A  .OSSY......Y....
00000D10 : 59 03 03 4F 59 04 03 4F - 59 05 04 4F 59 06 03 4F  Y..OY..OY..OY..O
00000D20 : 59 07 03 4F 53 59 04 08 - BC 0A 59 03 03 4F 59 04  Y..OSY....Y..OY.
00000D30 : 04 4F 59 05 04 4F 59 06 - 03 4F 59 07 03 4F 53 59  .OY..OY..OY..OSY
00000D40 : 05 08 BC 0A 59 03 03 4F - 59 04 03 4F 59 05 04 4F  ....Y..OY..OY..O
00000D50 : 59 06 03 4F 59 07 03 4F - 53 59 06 08 BC 0A 59 03  Y..OY..OSY....Y.
00000D60 : 03 4F 59 04 03 4F 59 05 - 04 4F 59 06 03 4F 59 07  .OY..OY..OY..OY.
00000D70 : 03 4F 53 59 07 08 BC 0A - 59 03 03 4F 59 04 03 4F  .OSY....Y..OY..O
00000D80 : 59 05 04 4F 59 06 03 4F - 59 07 03 4F 53 59 08 08  Y..OY..OY..OSY..
00000D90 : BC 0A 59 03 03 4F 59 04 - 03 4F 59 05 04 4F 59 06  ..Y..OY..OY..OY.
00000DA0 : 03 4F 59 07 03 4F 53 59 - 10 06 08 BC 0A 59 03 03  .OY..OSY.....Y..
00000DB0 : 4F 59 04 04 4F 59 05 04 - 4F 59 06 04 4F 59 07 03  OY..OY..OY..OY..
00000DC0 : 4F 53 53 59 05 10 07 BD - 00 04 59 03 08 BC 0A 59  OSSY......Y....Y
00000DD0 : 03 03 4F 59 04 04 4F 59 - 05 04 4F 59 06 04 4F 59  ..OY..OY..OY..OY
00000DE0 : 07 03 4F 53 59 04 08 BC - 0A 59 03 04 4F 59 04 03  ..OSY....Y..OY..
00000DF0 : 4F 59 05 03 4F 59 06 03 - 4F 59 07 04 4F 53 59 05  OY..OY..OY..OSY.
00000E00 : 08 BC 0A 59 03 04 4F 59 - 04 03 4F 59 05 03 4F 59  ...Y..OY..OY..OY
00000E10 : 06 03 4F 59 07 04 4F 53 - 59 06 08 BC 0A 59 03 03  ..OY..OSY....Y..
00000E20 : 4F 59 04 03 4F 59 05 03 - 4F 59 06 04 4F 59 07 03  OY..OY..OY..OY..
00000E30 : 4F 53 59 07 08 BC 0A 59 - 03 03 4F 59 04 03 4F 59  OSY....Y..OY..OY
00000E40 : 05 04 4F 59 06 03 4F 59 - 07 03 4F 53 59 08 08 BC  ..OY..OY..OSY...
00000E50 : 0A 59 03 03 4F 59 04 04 - 4F 59 05 03 4F 59 06 03  .Y..OY..OY..OY..
00000E60 : 4F 59 07 03 4F 53 59 10 - 06 08 BC 0A 59 03 04 4F  OY..OSY.....Y..O
00000E70 : 59 04 04 4F 59 05 04 4F - 59 06 04 4F 59 07 04 4F  Y..OY..OY..OY..O
00000E80 : 53 53 59 06 10 07 BD 00 - 04 59 03 08 BC 0A 59 03  SSY......Y....Y.
00000E90 : 03 4F 59 04 04 4F 59 05 - 04 4F 59 06 04 4F 59 07  .OY..OY..OY..OY.
00000EA0 : 03 4F 53 59 04 08 BC 0A - 59 03 04 4F 59 04 03 4F  .OSY....Y..OY..O
00000EB0 : 59 05 03 4F 59 06 03 4F - 59 07 04 4F 53 59 05 08  Y..OY..OY..OSY..
00000EC0 : BC 0A 59 03 03 4F 59 04 - 03 4F 59 05 03 4F 59 06  ..Y..OY..OY..OY.
00000ED0 : 03 4F 59 07 04 4F 53 59 - 06 08 BC 0A 59 03 03 4F  .OY..OSY....Y..O
00000EE0 : 59 04 03 4F 59 05 04 4F - 59 06 04 4F 59 07 03 4F  Y..OY..OY..OY..O
00000EF0 : 53 59 07 08 BC 0A 59 03 - 03 4F 59 04 03 4F 59 05  SY....Y..OY..OY.
00000F00 : 03 4F 59 06 03 4F 59 07 - 04 4F 53 59 08 08 BC 0A  .OY..OY..OSY....
00000F10 : 59 03 04 4F 59 04 03 4F - 59 05 03 4F 59 06 03 4F  Y..OY..OY..OY..O
00000F20 : 59 07 04 4F 53 59 10 06 - 08 BC 0A 59 03 03 4F 59  Y..OSY.....Y..OY
00000F30 : 04 04 4F 59 05 04 4F 59 - 06 04 4F 59 07 03 4F 53  ..OY..OY..OY..OS
00000F40 : 53 59 07 10 07 BD 00 04 - 59 03 08 BC 0A 59 03 03  SY......Y....Y..
00000F50 : 4F 59 04 03 4F 59 05 03 - 4F 59 06 04 4F 59 07 03  OY..OY..OY..OY..
00000F60 : 4F 53 59 04 08 BC 0A 59 - 03 03 4F 59 04 03 4F 59  OSY....Y..OY..OY
00000F70 : 05 04 4F 59 06 04 4F 59 - 07 03 4F 53 59 05 08 BC  ..OY..OY..OSY...
00000F80 : 0A 59 03 03 4F 59 04 04 - 4F 59 05 03 4F 59 06 04  .Y..OY..OY..OY..
00000F90 : 4F 59 07 03 4F 53 59 06 - 08 BC 0A 59 03 04 4F 59  OY..OSY....Y..OY
```

可変長のデータ・エリア　データ領域はあらかじめ特定のサイズとなっているのではなく，最初に「以下に続くデータの総数」を宣言してデータがこれに続くという形式．読み出す方ではこの数を正しく管理しなければならない．

〈リスト8-11〉 sample.classのrejava出力④

```
        00000FA0  :  04  03  4F  59  05  03  4F  59  -  06  04  4F  59  07  03  4F  53     ..OY..OY..OY..OS
        00000FB0  :  59  07  08  BC  0A  59  03  04  -  4F  59  04  04  4F  59  05  04     Y....Y..OY..OY..
        00000FC0  :  4F  59  06  04  4F  59  07  04  -  4F  53  59  08  08  BC  0A  59     OY..OY..OSY....Y
        00000FD0  :  03  03  4F  59  04  03  4F  59  -  05  03  4F  59  06  04  4F  59     ..OY..OY..OY..OY
        00000FE0  :  07  03  4F  53  59  10  06  08  -  BC  0A  59  03  03  4F  59  04     ..OSY.....Y..OY.
        00000FF0  :  03  4F  59  05  03  4F  59  06  -  04  4F  59  07  03  4F  53  53     .OY..OY..OY..OSS
        00001000  :  59  08  10  07  BD  00  04  59  -  03  08  BC  0A  59  03  04  4F     Y......Y....Y..O
        00001010  :  59  04  04  4F  59  05  04  4F  -  59  06  04  4F  59  07  04  4F     Y..OY..OY..OY..O
        00001020  :  53  59  04  08  BC  0A  59  03  -  04  4F  59  04  03  4F  59  05     SY....Y..OY..OY.
        00001030  :  03  4F  59  06  03  4F  59  07  -  03  4F  53  59  05  08  BC  0A     .OY..OY..OSY....
        00001040  :  59  03  04  4F  59  04  03  4F  -  59  05  03  4F  59  06  03  4F     Y..OY..OY..OY..O
        00001050  :  59  07  03  4F  53  59  06  08  -  BC  0A  59  03  04  4F  59  04     Y..OSY....Y..OY.
        00001060  :  04  4F  59  05  04  4F  59  06  -  04  4F  59  07  03  4F  53  59     .OY..OY..OY..OSY
        00001070  :  07  08  BC  0A  59  03  03  4F  -  59  04  03  4F  59  05  03  4F     ....Y..OY..OY..O
        00001080  :  59  06  03  4F  59  07  04  4F  -  53  59  08  08  BC  0A  59  03     Y..OY..OSY....Y.
        00001090  :  04  4F  59  04  03  4F  59  05  -  03  4F  59  06  03  4F  59  07     .OY..OY..OY..OY.
        000010A0  :  04  4F  53  59  10  06  08  BC  -  0A  59  03  03  4F  59  04  04     .OSY.....Y..OY..
        000010B0  :  4F  59  05  04  4F  59  06  04  -  4F  59  07  03  4F  53  53  59     OY..OY..OY..OSSY
        000010C0  :  10  06  10  07  BD  00  04  59  -  03  08  BC  0A  59  03  03  4F     .......Y....Y..O
        000010D0  :  59  04  04  4F  59  05  04  4F  -  59  06  04  4F  59  07  03  4F     Y..OY..OY..OY..O
        000010E0  :  53  59  04  08  BC  0A  59  03  -  04  4F  59  04  03  4F  59  05     SY....Y..OY..OY.
        000010F0  :  03  4F  59  06  03  4F  59  07  -  04  4F  53  59  05  08  BC  0A     .OY..OY..OSY....
        00001100  :  59  03  04  4F  59  04  03  4F  -  59  05  03  4F  59  06  03  4F     Y..OY..OY..OY..O
        00001110  :  59  07  03  4F  53  59  06  08  -  BC  0A  59  03  04  4F  59  04     Y..OSY....Y..OY.
        00001120  :  04  4F  59  05  04  4F  59  06  -  04  4F  59  07  03  4F  53  59     .OY..OY..OY..OSY
        00001130  :  07  08  BC  0A  59  03  04  4F  -  59  04  03  4F  59  05  03  4F     ....Y..OY..OY..O
        00001140  :  59  06  03  4F  59  07  04  4F  -  53  59  08  08  BC  0A  59  03     Y..OY..OSY....Y.
        00001150  :  04  4F  59  04  03  4F  59  05  -  03  4F  59  06  03  4F  59  07     .OY..OY..OY..OY.
        00001160  :  04  4F  53  59  10  06  08  BC  -  0A  59  03  03  4F  59  04  04     .OSY.....Y..OY..
        00001170  :  4F  59  05  04  4F  59  06  04  -  4F  59  07  03  4F  53  53  59     OY..OY..OY..OSSY
        00001180  :  10  07  10  07  BD  00  04  59  -  03  08  BC  0A  59  03  04  4F     .......Y....Y..O
        00001190  :  59  04  04  4F  59  05  04  4F  -  59  06  04  4F  59  07  04  4F     Y..OY..OY..OY..O
        000011A0  :  53  59  04  08  BC  0A  59  03  -  04  4F  59  04  03  4F  59  05     SY....Y..OY..OY.
        000011B0  :  03  4F  59  06  03  4F  59  07  -  04  4F  53  59  05  08  BC  0A     .OY..OY..OSY....
        000011C0  :  59  03  03  4F  59  04  03  4F  -  59  05  03  4F  59  06  03  4F     Y..OY..OY..OY..O
        000011D0  :  59  07  04  4F  53  59  06  08  -  BC  0A  59  03  03  4F  59  04     Y..OSY....Y..OY.
        000011E0  :  03  4F  59  05  03  4F  59  06  -  04  4F  59  07  03  4F  53  59     .OY..OY..OY..OSY
        000011F0  :  07  08  BC  0A  59  03  03  4F  -  59  04  03  4F  59  05  04  4F     ....Y..OY..OY..O
        00001200  :  59  06  03  4F  59  07  03  4F  -  53  59  08  08  BC  0A  59  03     Y..OY..OSY....Y.
        00001210  :  03  4F  59  04  04  4F  59  05  -  03  4F  59  06  03  4F  59  07     .OY..OY..OY..OY.
        00001220  :  03  4F  53  59  10  06  08  BC  -  0A  59  03  04  4F  59  04  03     .OSY.....Y..OY..
        00001230  :  4F  59  05  03  4F  59  06  03  -  4F  59  07  03  4F  53  53  59     OY..OY..OY..OSSY
        00001240  :  10  08  10  07  BD  00  04  59  -  03  08  BC  0A  59  03  03  4F     .......Y....Y..O
        00001250  :  59  04  04  4F  59  05  04  4F  -  59  06  04  4F  59  07  03  4F     Y..OY..OY..OY..O
        00001260  :  53  59  04  08  BC  0A  59  03  -  04  4F  59  04  03  4F  59  05     SY....Y..OY..OY.
        00001270  :  03  4F  59  06  03  4F  59  07  -  04  4F  53  59  05  08  BC  0A     .OY..OY..OSY....
        00001280  :  59  03  04  4F  59  04  03  4F  -  59  05  03  4F  59  06  03  4F     Y..OY..OY..OY..O
        00001290  :  59  07  04  4F  53  59  06  08  -  BC  0A  59  03  03  4F  59  04     Y..OSY....Y..OY.
        000012A0  :  04  4F  59  05  04  4F  59  06  -  04  4F  59  07  03  4F  53  59     .OY..OY..OY..OSY
        000012B0  :  07  08  BC  0A  59  03  04  4F  -  59  04  03  4F  59  05  03  4F     ....Y..OY..OY..O
        000012C0  :  59  06  03  4F  59  07  04  4F  -  53  59  08  08  BC  0A  59  03     Y..OY..OSY....Y.
        000012D0  :  04  4F  59  04  03  4F  59  05  -  03  4F  59  06  03  4F  59  07     .OY..OY..OY..OY.
        000012E0  :  04  4F  53  59  10  06  08  BC  -  0A  59  03  03  4F  59  04  04     .OSY.....Y..OY..
        000012F0  :  4F  59  05  04  4F  59  06  04  -  4F  59  07  03  4F  53  53  59     OY..OY..OY..OSSY
        00001300  :  10  09  10  07  BD  00  04  59  -  03  08  BC  0A  59  03  03  4F     .......Y....Y..O
        00001310  :  59  04  04  4F  59  05  04  4F  -  59  06  04  4F  59  07  03  4F     Y..OY..OY..OY..O
        00001320  :  53  59  04  08  BC  0A  59  03  -  04  4F  59  04  03  4F  59  05     SY....Y..OY..OY.
        00001330  :  03  4F  59  06  03  4F  59  07  -  04  4F  53  59  05  08  BC  0A     .OY..OY..OSY....
        00001340  :  59  03  04  4F  59  04  03  4F  -  59  05  03  4F  59  06  03  4F     Y..OY..OY..OY..O
        00001350  :  59  07  04  4F  53  59  06  08  -  BC  0A  59  03  03  4F  59  04     Y..OSY....Y..OY.
        00001360  :  04  4F  59  05  04  4F  59  06  -  04  4F  59  07  04  4F  53  59     .OY..OY..OY..OSY
        00001370  :  07  08  BC  0A  59  03  03  4F  -  59  04  03  4F  59  05  03  4F     ....Y..OY..OY..O
        00001380  :  59  06  03  4F  59  07  04  4F  -  53  59  08  08  BC  0A  59  03     Y..OY..OSY....Y.
        00001390  :  04  4F  59  04  03  4F  59  05  -  03  4F  59  06  03  4F  59  07     .OY..OY..OY..OY.
        000013A0  :  04  4F  53  59  10  06  08  BC  -  0A  59  03  03  4F  59  04  04     .OSY.....Y..OY..
        000013B0  :  4F  59  05  04  4F  59  06  04  -  4F  59  07  03  4F  53  53  B5     OY..OY..OY..OSS.
        000013C0  :  00  2E  B1  00  00  00  01  00  -  73  00  00  00  66  00  19  00     ........s...f...
        000013D0  :  00  00  06  00  04  00  08  00  -  26  00  09  00  2F  00  0B  00     ........&.../...
        000013E0  :  37  00  0C  00  F3  00  0B  00  -  F6  00  0D  01  B2  00  0B  01     7...............
        000013F0  :  B5  00  0E  02  71  00  0B  02  -  74  00  0F  03  30  00  0B  03     ....q...t...0...
        00001400  :  33  00  10  03  EF  00  0B  03  -  F2  00  11  04  AE  00  0B  04     3...............
        00001410  :  B2  00  12  05  6E  00  0B  05  -  72  00  13  06  2E  00  0B  06     ....n...r.......
        00001420  :  32  00  14  06  EE  00  0B  06  -  F2  00  15  07  AE  00  0B  07     2...............
        00001430  :  B2  00  06  00  01  00  6B  00  -  00  00  02  00  AF

Total File Length = 5181 bytes.

        00000000  Magic Code [CAFEBABE] = CAFEBABE  ---  OK (^_^)

        00000004  Version : major version 45 , minor version 3
        00000008  Constant Pool : total number = 185

                0000000A   constant_pool[1] = <8> CONSTANT_String , name_index = 172
                0000000D   constant_pool[2] = <8> CONSTANT_String , name_index = 180
```


手作業による逆アセンブラ　本文にあるように，メモをとりながら手作業によってJavaバイト・コードを解析していく作業のこと．

〈リスト8-11〉sample.classのrejava出力⑤

```
00000010   constant_pool[3] = <7> CONSTANT_Class , name_index = 150
00000013   constant_pool[4] = <7> CONSTANT_Class , name_index = 140
00000016   constant_pool[5] = <7> CONSTANT_Class , name_index = 136
00000019   constant_pool[6] = <7> CONSTANT_Class , name_index = 156
0000001C   constant_pool[7] = <7> CONSTANT_Class , name_index = 122
0000001F   constant_pool[8] = <7> CONSTANT_Class , name_index = 103
00000022   constant_pool[9] = <7> CONSTANT_Class , name_index = 145
00000025   constant_pool[10] = <7> CONSTANT_Class , name_index = 158
00000028   constant_pool[11] = <7> CONSTANT_Class , name_index = 165
0000002B   constant_pool[12] = <7> CONSTANT_Class , name_index = 157
0000002E   constant_pool[13] = <7> CONSTANT_Class , name_index = 173
00000031   constant_pool[14] = <7> CONSTANT_Class , name_index = 132
00000034   constant_pool[15] = <7> CONSTANT_Class , name_index = 160
00000037   constant_pool[16] = <7> CONSTANT_Class , name_index = 146
0000003A   constant_pool[17] = <7> CONSTANT_Class , name_index = 137
0000003D   constant_pool[18] = <A> CONSTANT_Methodref , class_index = 15 , name_and_type_index = 83
00000042   constant_pool[19] = <9> CONSTANT_Fieldref , class_index = 16 , name_and_type_index = 94
00000047   constant_pool[20] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 68
0000004C   constant_pool[21] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 87
00000051   constant_pool[22] = <A> CONSTANT_Methodref , class_index = 9 , name_and_type_index = 85
00000056   constant_pool[23] = <A> CONSTANT_Methodref , class_index = 9 , name_and_type_index = 74
0000005B   constant_pool[24] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 90
00000060   constant_pool[25] = <A> CONSTANT_Methodref , class_index = 13 , name_and_type_index = 84
00000065   constant_pool[26] = <A> CONSTANT_Methodref , class_index = 5 , name_and_type_index = 80
0000006A   constant_pool[27] = <A> CONSTANT_Methodref , class_index = 17 , name_and_type_index = 57
0000006F   constant_pool[28] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 63
00000074   constant_pool[29] = <A> CONSTANT_Methodref , class_index = 15 , name_and_type_index = 65
00000079   constant_pool[30] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 89
0000007E   constant_pool[31] = <A> CONSTANT_Methodref , class_index = 9 , name_and_type_index = 59
00000083   constant_pool[32] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 77
00000088   constant_pool[33] = <A> CONSTANT_Methodref , class_index = 17 , name_and_type_index = 86
0000008D   constant_pool[34] = <A> CONSTANT_Methodref , class_index = 17 , name_and_type_index = 61
00000092   constant_pool[35] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 91
00000097   constant_pool[36] = <A> CONSTANT_Methodref , class_index = 6 , name_and_type_index = 60
0000009C   constant_pool[37] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 95
000000A1   constant_pool[38] = <A> CONSTANT_Methodref , class_index = 15 , name_and_type_index = 66
000000A6   constant_pool[39] = <A> CONSTANT_Methodref , class_index = 15 , name_and_type_index = 76
000000AB   constant_pool[40] = <A> CONSTANT_Methodref , class_index = 5 , name_and_type_index = 82
000000B0   constant_pool[41] = <A> CONSTANT_Methodref , class_index = 5 , name_and_type_index = 62
000000B5   constant_pool[42] = <A> CONSTANT_Methodref , class_index = 10 , name_and_type_index = 64
000000BA   constant_pool[43] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 67
000000BF   constant_pool[44] = <A> CONSTANT_Methodref , class_index = 13 , name_and_type_index = 58
000000C4   constant_pool[45] = <A> CONSTANT_Methodref , class_index = 13 , name_and_type_index = 79
000000C9   constant_pool[46] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 56
000000CE   constant_pool[47] = <A> CONSTANT_Methodref , class_index = 10 , name_and_type_index = 81
000000D3   constant_pool[48] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 93
000000D8   constant_pool[49] = <9> CONSTANT_Fieldref , class_index = 8 , name_and_type_index = 92
000000DD   constant_pool[50] = <A> CONSTANT_Methodref , class_index = 9 , name_and_type_index = 75
000000E2   constant_pool[51] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 88
000000E7   constant_pool[52] = <A> CONSTANT_Methodref , class_index = 17 , name_and_type_index = 84
000000EC   constant_pool[53] = <9> CONSTANT_Fieldref , class_index = 15 , name_and_type_index = 73
000000F1   constant_pool[54] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 84
000000F6   constant_pool[55] = <9> CONSTANT_Fieldref , class_index = 16 , name_and_type_index = 78
000000FB   constant_pool[56] = <C> CONSTANT_NameAndType , name_index = 159 , signature_index = 108
00000100   constant_pool[57] = <C> CONSTANT_NameAndType , name_index = 171 , signature_index = 155
00000105   constant_pool[58] = <C> CONSTANT_NameAndType , name_index = 138 , signature_index = 125
0000010A   constant_pool[59] = <C> CONSTANT_NameAndType , name_index = 123 , signature_index = 139
0000010F   constant_pool[60] = <C> CONSTANT_NameAndType , name_index = 147 , signature_index = 143
00000114   constant_pool[61] = <C> CONSTANT_NameAndType , name_index = 171 , signature_index = 126
00000119   constant_pool[62] = <C> CONSTANT_NameAndType , name_index = 134 , signature_index = 130
0000011E   constant_pool[63] = <C> CONSTANT_NameAndType , name_index = 121 , signature_index = 170
00000123   constant_pool[64] = <C> CONSTANT_NameAndType , name_index = 128 , signature_index = 124
00000128   constant_pool[65] = <C> CONSTANT_NameAndType , name_index = 153 , signature_index = 144
0000012D   constant_pool[66] = <C> CONSTANT_NameAndType , name_index = 135 , signature_index = 111
00000132   constant_pool[67] = <C> CONSTANT_NameAndType , name_index = 154 , signature_index = 170
00000137   constant_pool[68] = <C> CONSTANT_NameAndType , name_index = 152 , signature_index = 141
0000013C   constant_pool[69] = <5> CONSTANT_Float , high = 0 , low = 1000
00000145   constant_pool[71] = <6> CONSTANT_Double , high = 1083127808 , low = 0
0000014E   constant_pool[73] = <C> CONSTANT_NameAndType , name_index = 161 , signature_index = 170
00000153   constant_pool[74] = <C> CONSTANT_NameAndType , name_index = 105 , signature_index = 109
00000158   constant_pool[75] = <C> CONSTANT_NameAndType , name_index = 148 , signature_index = 183
0000015D   constant_pool[76] = <C> CONSTANT_NameAndType , name_index = 110 , signature_index = 111
00000162   constant_pool[77] = <C> CONSTANT_NameAndType , name_index = 184 , signature_index = 182
00000167   constant_pool[78] = <C> CONSTANT_NameAndType , name_index = 176 , signature_index = 170
0000016C   constant_pool[79] = <C> CONSTANT_NameAndType , name_index = 133 , signature_index = 120
00000171   constant_pool[80] = <C> CONSTANT_NameAndType , name_index = 117 , signature_index = 167
00000176   constant_pool[81] = <C> CONSTANT_NameAndType , name_index = 142 , signature_index = 104
0000017B   constant_pool[82] = <C> CONSTANT_NameAndType , name_index = 118 , signature_index = 139
00000180   constant_pool[83] = <C> CONSTANT_NameAndType , name_index = 151 , signature_index = 144
00000185   constant_pool[84] = <C> CONSTANT_NameAndType , name_index = 105 , signature_index = 139
0000018A   constant_pool[85] = <C> CONSTANT_NameAndType , name_index = 114 , signature_index = 139
0000018F   constant_pool[86] = <C> CONSTANT_NameAndType , name_index = 112 , signature_index = 127
00000194   constant_pool[87] = <C> CONSTANT_NameAndType , name_index = 164 , signature_index = 170
00000199   constant_pool[88] = <C> CONSTANT_NameAndType , name_index = 129 , signature_index = 170
```

〈リスト8-11〉sample.classのrejava出力⑥

```
0000019E   constant_pool[89] = <C> CONSTANT_NameAndType , name_index = 106 , signature_index = 141
000001A3   constant_pool[90] = <C> CONSTANT_NameAndType , name_index = 179 , signature_index = 141
000001A8   constant_pool[91] = <C> CONSTANT_NameAndType , name_index = 119 , signature_index = 170
000001AD   constant_pool[92] = <C> CONSTANT_NameAndType , name_index = 116 , signature_index = 181
000001B2   constant_pool[93] = <C> CONSTANT_NameAndType , name_index = 166 , signature_index = 170
000001B7   constant_pool[94] = <C> CONSTANT_NameAndType , name_index = 163 , signature_index = 170
000001BC   constant_pool[95] = <C> CONSTANT_NameAndType , name_index = 162 , signature_index = 178
000001C1   constant_pool[96] = <6> CONSTANT_Double , high = 1074921472 , low = 0
000001CA   constant_pool[98] = <6> CONSTANT_Double , high = 1073217536 , low = 0
000001D3   constant_pool[100] = <6> CONSTANT_Double , high = 1072546447 , low = 1546188227
000001DC   constant_pool[102] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Graphics;)V
000001F5   constant_pool[103] = <1> CONSTANT_Utf8 , Data = java/awt/Color
00000206   constant_pool[104] = <1> CONSTANT_Utf8 , Data = (IIII)V
00000210   constant_pool[105] = <1> CONSTANT_Utf8 , Data = <init>
00000219   constant_pool[106] = <1> CONSTANT_Utf8 , Data = getHours
00000224   constant_pool[107] = <1> CONSTANT_Utf8 , Data = SourceFile
00000231   constant_pool[108] = <1> CONSTANT_Utf8 , Data = [[[I
00000238   constant_pool[109] = <1> CONSTANT_Utf8 , Data = (Ljava/lang/Runnable;)V
00000252   constant_pool[110] = <1> CONSTANT_Utf8 , Data = dot_draw
0000025D   constant_pool[111] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Graphics;II)V
00000278   constant_pool[112] = <1> CONSTANT_Utf8 , Data = toString
00000283   constant_pool[113] = <1> CONSTANT_Utf8 , Data = Exceptions
00000290   constant_pool[114] = <1> CONSTANT_Utf8 , Data = stop
00000297   constant_pool[115] = <1> CONSTANT_Utf8 , Data = LineNumberTable
000002A9   constant_pool[116] = <1> CONSTANT_Utf8 , Data = black
000002B1   constant_pool[117] = <1> CONSTANT_Utf8 , Data = size
000002B8   constant_pool[118] = <1> CONSTANT_Utf8 , Data = repaint
000002C2   constant_pool[119] = <1> CONSTANT_Utf8 , Data = sz
000002C7   constant_pool[120] = <1> CONSTANT_Utf8 , Data = (Ljava/net/URL;Ljava/lang/String;)Ljava/awt/Image;
000002FC   constant_pool[121] = <1> CONSTANT_Utf8 , Data = st
00000301   constant_pool[122] = <1> CONSTANT_Utf8 , Data = java/util/Date
00000312   constant_pool[123] = <1> CONSTANT_Utf8 , Data = start
0000031A   constant_pool[124] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Image;IIIILjava/awt/image/ImageObserver;)Z
00000352   constant_pool[125] = <1> CONSTANT_Utf8 , Data = ()Ljava/net/URL;
00000365   constant_pool[126] = <1> CONSTANT_Utf8 , Data = (Ljava/lang/String;)Ljava/lang/StringBuffer;
00000394   constant_pool[127] = <1> CONSTANT_Utf8 , Data = ()Ljava/lang/String;
000003AB   constant_pool[128] = <1> CONSTANT_Utf8 , Data = drawImage
000003B7   constant_pool[129] = <1> CONSTANT_Utf8 , Data = color_sel
000003C3   constant_pool[130] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Color;)V
000003D9   constant_pool[131] = <1> CONSTANT_Utf8 , Data = init
000003E0   constant_pool[132] = <1> CONSTANT_Utf8 , Data = [[I
000003E6   constant_pool[133] = <1> CONSTANT_Utf8 , Data = getImage
000003F1   constant_pool[134] = <1> CONSTANT_Utf8 , Data = setBackground
00000401   constant_pool[135] = <1> CONSTANT_Utf8 , Data = char_dot
0000040C   constant_pool[136] = <1> CONSTANT_Utf8 , Data = java/awt/Component
00000421   constant_pool[137] = <1> CONSTANT_Utf8 , Data = java/lang/StringBuffer
0000043A   constant_pool[138] = <1> CONSTANT_Utf8 , Data = getDocumentBase
0000044C   constant_pool[139] = <1> CONSTANT_Utf8 , Data = ()V
00000452   constant_pool[140] = <1> CONSTANT_Utf8 , Data = [I
00000457   constant_pool[141] = <1> CONSTANT_Utf8 , Data = ()I
0000045D   constant_pool[142] = <1> CONSTANT_Utf8 , Data = clearRect
00000469   constant_pool[143] = <1> CONSTANT_Utf8 , Data = ()D
0000046F   constant_pool[144] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Graphics;III)V
0000048B   constant_pool[145] = <1> CONSTANT_Utf8 , Data = java/lang/Thread
0000049E   constant_pool[146] = <1> CONSTANT_Utf8 , Data = java/awt/Dimension
000004B3   constant_pool[147] = <1> CONSTANT_Utf8 , Data = random
000004BC   constant_pool[148] = <1> CONSTANT_Utf8 , Data = sleep
000004C4   constant_pool[149] = <1> CONSTANT_Utf8 , Data = run
000004CA   constant_pool[150] = <1> CONSTANT_Utf8 , Data = java/lang/Exception
000004E0   constant_pool[151] = <1> CONSTANT_Utf8 , Data = clock_disp
000004ED   constant_pool[152] = <1> CONSTANT_Utf8 , Data = getSeconds
000004FA   constant_pool[153] = <1> CONSTANT_Utf8 , Data = char_disp
00000506   constant_pool[154] = <1> CONSTANT_Utf8 , Data = yy
0000050B   constant_pool[155] = <1> CONSTANT_Utf8 , Data = (I)Ljava/lang/StringBuffer;
00000529   constant_pool[156] = <1> CONSTANT_Utf8 , Data = java/lang/Math
0000053A   constant_pool[157] = <1> CONSTANT_Utf8 , Data = java/lang/Runnable
0000054F   constant_pool[158] = <1> CONSTANT_Utf8 , Data = java/awt/Graphics
00000563   constant_pool[159] = <1> CONSTANT_Utf8 , Data = m
00000567   constant_pool[160] = <1> CONSTANT_Utf8 , Data = sample
00000570   constant_pool[161] = <1> CONSTANT_Utf8 , Data = j
00000574   constant_pool[162] = <1> CONSTANT_Utf8 , Data = i
00000578   constant_pool[163] = <1> CONSTANT_Utf8 , Data = width
00000580   constant_pool[164] = <1> CONSTANT_Utf8 , Data = mode
00000587   constant_pool[165] = <1> CONSTANT_Utf8 , Data = java/awt/Image
00000598   constant_pool[166] = <1> CONSTANT_Utf8 , Data = xx
0000059D   constant_pool[167] = <1> CONSTANT_Utf8 , Data = ()Ljava/awt/Dimension;
000005B6   constant_pool[168] = <1> CONSTANT_Utf8 , Data = paint
000005BE   constant_pool[169] = <1> CONSTANT_Utf8 , Data = ConstantValue
000005CE   constant_pool[170] = <1> CONSTANT_Utf8 , Data = I
000005D2   constant_pool[171] = <1> CONSTANT_Utf8 , Data = append
000005DB   constant_pool[172] = <1> CONSTANT_Utf8 , Data = .gif
000005E2   constant_pool[173] = <1> CONSTANT_Utf8 , Data = java/applet/Applet
000005F7   constant_pool[174] = <1> CONSTANT_Utf8 , Data = Code
000005FE   constant_pool[175] = <1> CONSTANT_Utf8 , Data = sample.java
```

〈リスト8-11〉sample.classのrejava出力⑦

```
        0000060C   constant_pool[176] = <1> CONSTANT_Utf8 , Data = height
        00000615   constant_pool[177] = <1> CONSTANT_Utf8 , Data = LocalVariables
        00000626   constant_pool[178] = <1> CONSTANT_Utf8 , Data = [Ljava/awt/Image;
        0000063A   constant_pool[179] = <1> CONSTANT_Utf8 , Data = getMinutes
        00000647   constant_pool[180] = <1> CONSTANT_Utf8 , Data = java/gif/b
        00000654   constant_pool[181] = <1> CONSTANT_Utf8 , Data = Ljava/awt/Color;
        00000667   constant_pool[182] = <1> CONSTANT_Utf8 , Data = Ljava/lang/Thread;
        0000067C   constant_pool[183] = <1> CONSTANT_Utf8 , Data = (J)V
        00000683   constant_pool[184] = <1> CONSTANT_Utf8 , Data = flow

0000068A  Access Flag = 0001 --- ACC_PUBLIC

0000068C  This Class = sample

0000068E  Super Class = java/applet/Applet

00000690  Interface Count = 1

        00000692  interface[0] : Name = java/lang/Runnable

00000694  Fields Count = 10

        00000696   field[0] : ( no flag ) , Name = j , Signature = I , Attribute = 0
        0000069E   field[1] : ( no flag ) , Name = color_sel , Signature = I , Attribute = 0
        000006A6   field[2] : ( no flag ) , Name = xx , Signature = I , Attribute = 0
        000006AE   field[3] : ( no flag ) , Name = yy , Signature = I , Attribute = 0
        000006B6   field[4] : ( no flag ) , Name = sz , Signature = I , Attribute = 0
        000006BE   field[5] : ( no flag ) , Name = st , Signature = I , Attribute = 0
        000006C6   field[6] : ( no flag ) , Name = mode , Signature = I , Attribute = 0
        000006CE   field[7] : ( no flag ) , Name = i , Signature = [Ljava/awt/Image; , Attribute = 0
        000006D6   field[8] : ACC_PRIVATE , Name = flow , Signature = Ljava/lang/Thread; , Attribute = 0
        000006DE   field[9] : ( no flag ) , Name = m , Signature = [[[I , Attribute = 0

000006E6  Methods Count = 10

        000006E8   method[0] : ACC_PUBLIC , Name = init , Signature = ()V , Attribute = 1

                type = Code , length = 106 , max_stack = 6 , max_locals = 3 , code_length = 62

                        000000  aload_0                 (Load object reference from local variable)
                        000001  invokevirtual   002C    (Invoke instance method)
                        000004  astore_1                (Store object reference into local variable)
                        000005  iconst_1                (Push integer constant)
                        000006  istore_2                (Store integer into local variable)
                        000007  goto    0029            (Branch)
                        00000A  aload_0                 (Load object reference from local variable)
                        00000B  getfield        0025    (Fetch field from object)
                        00000E  iload_2                 (Load integer from local variable)
                        00000F  aload_0                 (Load object reference from local variable)
                        000010  aload_1                 (Load object reference from local variable)
                        000011  new     0011            (Create new object)
                        000014  dup                     (Duplicate top stack word)
                        000015  invokenonvirt   0034    (Invoke instance method, dispatching based on compile-time type)
                        000018  ldc1    02              (Push item from constant pool)
                        00001A  invokevirtual   0022    (Invoke instance method)
                        00001D  iload_2                 (Load integer from local variable)
                        00001E  invokevirtual   001B    (Invoke instance method)
                        000021  ldc1    01              (Push item from constant pool)
                        000023  invokevirtual   0022    (Invoke instance method)
                        000026  invokevirtual   0021    (Invoke instance method)
                        000029  invokevirtual   002D    (Invoke instance method)
                        00002C  aastore                 (Store into object reference array)
                        00002D  iinc    02 01           (Increment local variable by constant)
                        000030  iload_2                 (Load integer from local variable)
                        000031  bipush  0E              (Push one-byte signed integer)
                        000033  if_icmple       FFD7    (Branch if integer less than or equal to)
                        000036  aload_0                 (Load object reference from local variable)
                        000037  getstatic       0031    (Get static field from class)
                        00003A  invokevirtual   0029    (Invoke instance method)
                        00003D  return                  (Return (void) from procedure)

                exception_table_length = 0

                attributes_count = 1

                        00000742  type = LineNumberTable , attribute_length = 26

                                00 06 00 00 00 19 00 05 00 1A 00 0A 00 1B 00 2D 00 1A 00 36
                                00 1D 00 3D 00 18

        00000760   method[1] : ACC_PUBLIC , Name = start , Signature = ()V , Attribute = 1

                type = Code , length = 52 , max_stack = 4 , max_locals = 1 , code_length = 20
```

〈リスト8-11〉 sample.classのrejava出力⑧

```
                000000  aload_0                 (Load object reference from local variable)
                000001  new     0009            (Create new object)
                000004  dup                     (Duplicate top stack word)
                000005  aload_0                 (Load object reference from local variable)
                000006  invokenonvirt   0017    (Invoke instance method, dispatching based on compile-time type)
                000009  putfield        0020    (Set field in object)
                00000C  aload_0                 (Load object reference from local variable)
                00000D  getfield        0020    (Fetch field from object)
                000010  invokevirtual   001F    (Invoke instance method)
                000013  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00000790  type = LineNumberTable , attribute_length = 14

                        00 03 00 00 00 21 00 0C 00 22 00 13 00 20

000007A2   method[2] : ACC_PUBLIC , Name = run , Signature = ()V , Attribute = 1

        type = Code , length = 77 , max_stack = 3 , max_locals = 1 , code_length = 25

                000000  ldc2w   0045            (Push long or double from constant pool)
                000003  invokestatic    0032    (Invoke a class (static) method)
                000006  aload_0                 (Load object reference from local variable)
                000007  invokevirtual   0028    (Invoke instance method)
                00000A  goto    FFF6            (Branch)
                00000D  pop                     (Pop top stack word)
                00000E  aload_0                 (Load object reference from local variable)
                00000F  dup                     (Duplicate top stack word)
                000010  getfield        0035    (Fetch field from object)
                000013  iconst_1                (Push integer constant)
                000014  iadd                    (Integer add)
                000015  putfield        0035    (Set field in object)
                000018  return                  (Return (void) from procedure)

        exception_table_length = 1

                000007D5  start_pc = 0 , end_pc = 13 , handler_pc = 13 , catch_type = java/lang/Exception

        attributes_count = 1

                000007DF  type = LineNumberTable , attribute_length = 26

                        00 06 00 00 00 26 00 00 00 28 00 06 00 29 00 0A 00 27 00 0D
                        00 2C 00 18 00 25

000007FD   method[3] : ACC_PUBLIC , Name = stop , Signature = ()V , Attribute = 1

        type = Code , length = 36 , max_stack = 1 , max_locals = 1 , code_length = 8

                000000  aload_0                 (Load object reference from local variable)
                000001  getfield        0020    (Fetch field from object)
                000004  invokevirtual   0016    (Invoke instance method)
                000007  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00000821  type = LineNumberTable , attribute_length = 10

                        00 02 00 00 00 30 00 07 00 2F

0000082F   method[4] : ACC_PUBLIC , Name = paint , Signature = (Ljava/awt/Graphics;)V , Attribute = 1

        type = Code , length = 109 , max_stack = 5 , max_locals = 7 , code_length = 57

                000000  aload_0                 (Load object reference from local variable)
                000001  invokevirtual   001A    (Invoke instance method)
                000004  astore_2                (Store object reference into local variable)
                000005  aload_1                 (Load object reference from local variable)
                000006  iconst_0                (Push integer constant)
                000007  iconst_0                (Push integer constant)
                000008  aload_2                 (Load object reference from local variable)
                000009  getfield        0013    (Fetch field from object)
                00000C  aload_2                 (Load object reference from local variable)
                00000D  getfield        0037    (Fetch field from object)
                000010  invokevirtual   002F    (Invoke instance method)
                000013  new     0007            (Create new object)
                000016  dup                     (Duplicate top stack word)
                000017  invokenonvirt   0036    (Invoke instance method, dispatching based on compile-time type)
                00001A  astore_3                (Store object reference into local variable)
```

〈リスト8-11〉sample.classのrejava出力⑨

```
                00001B  aload_3                 (Load object reference from local variable)
                00001C  invokevirtual   001E    (Invoke instance method)
                00001F  istore  04              (Store integer into local variable)
                000021  aload_3                 (Load object reference from local variable)
                000022  invokevirtual   0018    (Invoke instance method)
                000025  istore  05              (Store integer into local variable)
                000027  aload_3                 (Load object reference from local variable)
                000028  invokevirtual   0014    (Invoke instance method)
                00002B  istore  06              (Store integer into local variable)
                00002D  aload_0                 (Load object reference from local variable)
                00002E  aload_1                 (Load object reference from local variable)
                00002F  iload   04              (Load integer from local variable)
                000031  iload   05              (Load integer from local variable)
                000033  iload   06              (Load integer from local variable)
                000035  invokevirtual   0012    (Invoke instance method)
                000038  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00000884  type = LineNumberTable , attribute_length = 34

                        00 08 00 00 00 34 00 05 00 35 00 13 00 36 00 1B 00 37 00 21
                        00 38 00 27 00 39 00 2D 00 3A 00 38 00 33

000008AA    method[5] : ( no flag ) , Name = clock_disp , Signature = (Ljava/awt/Graphics;III)V , Attribute = 1

        type = Code , length = 504 , max_stack = 5 , max_locals = 10 , code_length = 396

                000000  iconst_1                (Push integer constant)
                000001  istore  09              (Store integer into local variable)
                000003  aload_0                 (Load object reference from local variable)
                000004  getfield        002B    (Fetch field from object)
                000007  istore  06              (Store integer into local variable)
                000009  aload_0                 (Load object reference from local variable)
                00000A  getfield        0015    (Fetch field from object)
                00000D  ifeq    0032            (Branch if equal to 0)
                000010  aload_0                 (Load object reference from local variable)
                000011  getfield        0033    (Fetch field from object)
                000014  istore  09              (Store integer into local variable)
                000016  aload_0                 (Load object reference from local variable)
                000017  dup                     (Duplicate top stack word)
                000018  getfield        0023    (Fetch field from object)
                00001B  iconst_3                (Push integer constant)
                00001C  iadd                    (Integer add)
                00001D  putfield        0023    (Set field in object)
                000020  aload_0                 (Load object reference from local variable)
                000021  dup                     (Duplicate top stack word)
                000022  getfield        001C    (Fetch field from object)
                000025  iconst_2                (Push integer constant)
                000026  iadd                    (Integer add)
                000027  putfield        001C    (Set field in object)
                00002A  aload_0                 (Load object reference from local variable)
                00002B  dup                     (Duplicate top stack word)
                00002C  getfield        0030    (Fetch field from object)
                00002F  bipush  28              (Push one-byte signed integer)
                000031  isub                    (Integer subtract)
                000032  putfield        0030    (Set field in object)
                000035  aload_0                 (Load object reference from local variable)
                000036  dup                     (Duplicate top stack word)
                000037  getfield        002B    (Fetch field from object)
                00003A  iconst_5                (Push integer constant)
                00003B  isub                    (Integer subtract)
                00003C  putfield        002B    (Set field in object)
                00003F  aload_0                 (Load object reference from local variable)
                000040  getfield        0030    (Fetch field from object)
                000043  aload_0                 (Load object reference from local variable)
                000044  getfield        001C    (Fetch field from object)
                000047  bipush  1C              (Push one-byte signed integer)
                000049  imul                    (Integer multiply)
                00004A  iadd                    (Integer add)
                00004B  istore  05              (Store integer into local variable)
                00004D  iload   04              (Load integer from local variable)
                00004F  bipush  0A              (Push one-byte signed integer)
                000051  idiv                    (Integer divide)
                000052  istore  07              (Store integer into local variable)
                000054  aload_0                 (Load object reference from local variable)
                000055  aload_1                 (Load object reference from local variable)
                000056  iload   05              (Load integer from local variable)
                000058  iload   06              (Load integer from local variable)
                00005A  iload   07              (Load integer from local variable)
                00005C  invokevirtual   001D    (Invoke instance method)
                00005F  aload_0                 (Load object reference from local variable)
```

〈リスト8-11〉sample.classのrejava出力⑩

```
000060  getfield        0030    (Fetch field from object)
000063  aload_0                  (Load object reference from local variable)
000064  getfield        001C    (Fetch field from object)
000067  bipush  22               (Push one-byte signed integer)
000069  imul                     (Integer multiply)
00006A  iadd                     (Integer add)
00006B  istore  05               (Store integer into local variable)
00006D  iload   04               (Load integer from local variable)
00006F  bipush  0A               (Push one-byte signed integer)
000071  irem                     (Integer remainder)
000072  istore  08               (Store integer into local variable)
000074  aload_0                  (Load object reference from local variable)
000075  aload_1                  (Load object reference from local variable)
000076  iload   05               (Load integer from local variable)
000078  iload   06               (Load integer from local variable)
00007A  iload   08               (Load integer from local variable)
00007C  invokevirtual   001D     (Invoke instance method)
00007F  aload_0                  (Load object reference from local variable)
000080  getfield        0030    (Fetch field from object)
000083  aload_0                  (Load object reference from local variable)
000084  getfield        001C    (Fetch field from object)
000087  bipush  0E               (Push one-byte signed integer)
000089  imul                     (Integer multiply)
00008A  iadd                     (Integer add)
00008B  istore  05               (Store integer into local variable)
00008D  iload_3                  (Load integer from local variable)
00008E  bipush  0A               (Push one-byte signed integer)
000090  idiv                     (Integer divide)
000091  istore  07               (Store integer into local variable)
000093  aload_0                  (Load object reference from local variable)
000094  aload_1                  (Load object reference from local variable)
000095  iload   05               (Load integer from local variable)
000097  iload   06               (Load integer from local variable)
000099  iload   07               (Load integer from local variable)
00009B  invokevirtual   001D     (Invoke instance method)
00009E  aload_0                  (Load object reference from local variable)
00009F  getfield        0030    (Fetch field from object)
0000A2  aload_0                  (Load object reference from local variable)
0000A3  getfield        001C    (Fetch field from object)
0000A6  bipush  14               (Push one-byte signed integer)
0000A8  imul                     (Integer multiply)
0000A9  iadd                     (Integer add)
0000AA  istore  05               (Store integer into local variable)
0000AC  iload_3                  (Load integer from local variable)
0000AD  bipush  0A               (Push one-byte signed integer)
0000AF  irem                     (Integer remainder)
0000B0  istore  07               (Store integer into local variable)
0000B2  aload_0                  (Load object reference from local variable)
0000B3  aload_1                  (Load object reference from local variable)
0000B4  iload   05               (Load integer from local variable)
0000B6  iload   06               (Load integer from local variable)
0000B8  iload   07               (Load integer from local variable)
0000BA  invokevirtual   001D     (Invoke instance method)
0000BD  aload_0                  (Load object reference from local variable)
0000BE  getfield        0030    (Fetch field from object)
0000C1  istore  05               (Store integer into local variable)
0000C3  iload_2                  (Load integer from local variable)
0000C4  bipush  0A               (Push one-byte signed integer)
0000C6  idiv                     (Integer divide)
0000C7  istore  07               (Store integer into local variable)
0000C9  aload_0                  (Load object reference from local variable)
0000CA  aload_1                  (Load object reference from local variable)
0000CB  iload   05               (Load integer from local variable)
0000CD  iload   06               (Load integer from local variable)
0000CF  iload   07               (Load integer from local variable)
0000D1  invokevirtual   001D     (Invoke instance method)
0000D4  aload_0                  (Load object reference from local variable)
0000D5  getfield        0030    (Fetch field from object)
0000D8  aload_0                  (Load object reference from local variable)
0000D9  getfield        001C    (Fetch field from object)
0000DC  bipush  06               (Push one-byte signed integer)
0000DE  imul                     (Integer multiply)
0000DF  iadd                     (Integer add)
0000E0  istore  05               (Store integer into local variable)
0000E2  iload_2                  (Load integer from local variable)
0000E3  bipush  0A               (Push one-byte signed integer)
0000E5  irem                     (Integer remainder)
0000E6  istore  07               (Store integer into local variable)
0000E8  aload_0                  (Load object reference from local variable)
0000E9  aload_1                  (Load object reference from local variable)
0000EA  iload   05               (Load integer from local variable)
0000EC  iload   06               (Load integer from local variable)
0000EE  iload   07               (Load integer from local variable)
0000F0  invokevirtual   001D     (Invoke instance method)
```

〈リスト8-11〉sample.classのrejava出力⑪

```
0000F3  aload_0                 (Load object reference from local variable)
0000F4  getfield        0030    (Fetch field from object)
0000F7  aload_0                 (Load object reference from local variable)
0000F8  getfield        001C    (Fetch field from object)
0000FB  bipush  0C              (Push one-byte signed integer)
0000FD  imul                    (Integer multiply)
0000FE  iadd                    (Integer add)
0000FF  istore  05              (Store integer into local variable)
000101  aload_0                 (Load object reference from local variable)
000102  aload_1                 (Load object reference from local variable)
000103  iload   05              (Load integer from local variable)
000105  iload   06              (Load integer from local variable)
000107  invokevirtual   0026    (Invoke instance method)
00010A  aload_0                 (Load object reference from local variable)
00010B  getfield        0030    (Fetch field from object)
00010E  aload_0                 (Load object reference from local variable)
00010F  getfield        001C    (Fetch field from object)
000112  bipush  1A              (Push one-byte signed integer)
000114  imul                    (Integer multiply)
000115  iadd                    (Integer add)
000116  istore  05              (Store integer into local variable)
000118  aload_0                 (Load object reference from local variable)
000119  aload_1                 (Load object reference from local variable)
00011A  iload   05              (Load integer from local variable)
00011C  iload   06              (Load integer from local variable)
00011E  invokevirtual   0026    (Invoke instance method)
000121  aload_0                 (Load object reference from local variable)
000122  getfield        0015    (Fetch field from object)
000125  ifeq    0032            (Branch if equal to 0)
000128  aload_0                 (Load object reference from local variable)
000129  iload   09              (Load integer from local variable)
00012B  putfield        0033    (Set field in object)
00012E  aload_0                 (Load object reference from local variable)
00012F  dup                     (Duplicate top stack word)
000130  getfield        0023    (Fetch field from object)
000133  iconst_3               (Push integer constant)
000134  isub                    (Integer subtract)
000135  putfield        0023    (Set field in object)
000138  aload_0                 (Load object reference from local variable)
000139  dup                     (Duplicate top stack word)
00013A  getfield        001C    (Fetch field from object)
00013D  iconst_2                (Push integer constant)
00013E  isub                    (Integer subtract)
00013F  putfield        001C    (Set field in object)
000142  aload_0                 (Load object reference from local variable)
000143  dup                     (Duplicate top stack word)
000144  getfield        0030    (Fetch field from object)
000147  bipush  28              (Push one-byte signed integer)
000149  iadd                    (Integer add)
00014A  putfield        0030    (Set field in object)
00014D  aload_0                 (Load object reference from local variable)
00014E  dup                     (Duplicate top stack word)
00014F  getfield        002B    (Fetch field from object)
000152  iconst_5                (Push integer constant)
000153  iadd                    (Integer add)
000154  putfield        002B    (Set field in object)
000157  iload   08              (Load integer from local variable)
000159  bipush  09              (Push one-byte signed integer)
00015B  if_icmpne       001B    (Branch if integers not equal)
00015E  aload_0                 (Load object reference from local variable)
00015F  dup                     (Duplicate top stack word)
000160  getfield        0033    (Fetch field from object)
000163  iconst_1                (Push integer constant)
000164  iadd                    (Integer add)
000165  putfield        0033    (Set field in object)
000168  aload_0                 (Load object reference from local variable)
000169  getfield        0033    (Fetch field from object)
00016C  bipush  0E              (Push one-byte signed integer)
00016E  if_icmple       0008    (Branch if integer less than or equal to)
000171  aload_0                 (Load object reference from local variable)
000172  iconst_1                (Push integer constant)
000173  putfield        0033    (Set field in object)
000176  invokestatic    0024    (Invoke a class (static) method)
000179  ldc2w   0064            (Push long or double from constant pool)
00017C  dcmpl                   (Double float compare (-1 on NaN))
00017D  ifle    0009            (Branch if less than or equal to 0)
000180  aload_0                 (Load object reference from local variable)
000181  iconst_1                (Push integer constant)
000182  putfield        0015    (Set field in object)
000185  return                  (Return (void) from procedure)
000186  aload_0                 (Load object reference from local variable)
000187  iconst_0                (Push integer constant)
000188  putfield        0015    (Set field in object)
00018B  return                  (Return (void) from procedure)
```

〈リスト8-11〉 sample.classのrejava出力⑫

```
        exception_table_length = 0

        attributes_count = 1

                00000A52  type = LineNumberTable , attribute_length = 90

                        00 16 00 00 00 3E 00 03 00 3F 00 09 00 40 00 10 00 41 00 16
                        00 42 00 3F 00 44 00 5F 00 45 00 7F 00 46 00 9E 00 47 00 BD
                        00 48 00 D4 00 49 00 F3 00 4A 01 0A 00 4B 01 21 00 4C 01 28
                        00 4D 01 2E 00 4E 01 57 00 50 01 5E 00 51 01 68 00 52 01 76
                        00 54 01 86 00 55 01 8B 00 3D

00000AB0    method[6] : ( no flag ) , Name = char_dot , Signature = (Ljava/awt/Graphics;II)V , Attribute = 1

        type = Code , length = 69 , max_stack = 8 , max_locals = 4 , code_length = 37

                000000  aload_0                 (Load object reference from local variable)
                000001  aload_1                 (Load object reference from local variable)
                000002  iload_2                 (Load integer from local variable)
                000003  iload_3                 (Load integer from local variable)
                000004  aload_0                 (Load object reference from local variable)
                000005  getfield        001C    (Fetch field from object)
                000008  i2d                     (Integer to double float)
                000009  ldc2w   0062            (Push long or double from constant pool)
                00000C  dmul                    (Double float multiply)
                00000D  d2i                     (Double float to integer)
                00000E  iadd                    (Integer add)
                00000F  invokevirtual   0027    (Invoke instance method)
                000012  aload_0                 (Load object reference from local variable)
                000013  aload_1                 (Load object reference from local variable)
                000014  iload_2                 (Load integer from local variable)
                000015  iload_3                 (Load integer from local variable)
                000016  aload_0                 (Load object reference from local variable)
                000017  getfield        001C    (Fetch field from object)
                00001A  i2d                     (Integer to double float)
                00001B  ldc2w   0060            (Push long or double from constant pool)
                00001E  dmul                    (Double float multiply)
                00001F  d2i                     (Double float to integer)
                000020  iadd                    (Integer add)
                000021  invokevirtual   0027    (Invoke instance method)
                000024  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00000AF1  type = LineNumberTable , attribute_length = 14

                        00 03 00 00 00 59 00 12 00 5A 00 24 00 58

00000B03    method[7] : ( no flag ) , Name = dot_draw , Signature = (Ljava/awt/Graphics;II)V , Attribute = 1

        type = Code , length = 99 , max_stack = 7 , max_locals = 5 , code_length = 55

                000000  aload_0                 (Load object reference from local variable)
                000001  getfield        0015    (Fetch field from object)
                000004  ifne    000C            (Branch if not equal to 0)
                000007  aload_0                 (Load object reference from local variable)
                000008  getfield        0033    (Fetch field from object)
                00000B  istore  04              (Store integer into local variable)
                00000D  goto    0012            (Branch)
                000010  iconst_1                (Push integer constant)
                000011  invokestatic    0024    (Invoke a class (static) method)
                000014  ldc2w   0047            (Push long or double from constant pool)
                000017  dmul                    (Double float multiply)
                000018  d2i                     (Double float to integer)
                000019  bipush  0E              (Push one-byte signed integer)
                00001B  irem                    (Integer remainder)
                00001C  iadd                    (Integer add)
                00001D  istore  04              (Store integer into local variable)
                00001F  aload_1                 (Load object reference from local variable)
                000020  aload_0                 (Load object reference from local variable)
                000021  getfield        0025    (Fetch field from object)
                000024  iload   04              (Load integer from local variable)
                000026  aaload                   (Load object reference from array)
                000027  iload_2                 (Load integer from local variable)
                000028  iload_3                 (Load integer from local variable)
                000029  aload_0                 (Load object reference from local variable)
                00002A  getfield        0023    (Fetch field from object)
                00002D  aload_0                 (Load object reference from local variable)
                00002E  getfield        0023    (Fetch field from object)
                000031  aload_0                 (Load object reference from local variable)
                000032  invokevirtual   002A    (Invoke instance method)
                000035  pop                     (Pop top stack word)
```

〈リスト8-11〉sample.classのrejava出力⑬

```
                000036  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00000B56  type = LineNumberTable , attribute_length = 26

                        00 06 00 00 00 5F 00 07 00 60 00 0D 00 5F 00 10 00 63 00 1F
                        00 65 00 36 00 5D

00000B74    method[8] : ( no flag ) , Name = char_disp , Signature = (Ljava/awt/Graphics;III)V , Attribute = 1

        type = Code , length = 120 , max_stack = 6 , max_locals = 7 , code_length = 72

                000000  iconst_0                (Push integer constant)
                000001  istore  05              (Store integer into local variable)
                000003  goto    003E            (Branch)
                000006  iconst_0                (Push integer constant)
                000007  istore  06              (Store integer into local variable)
                000009  goto    002E            (Branch)
                00000C  aload_0                 (Load object reference from local variable)
                00000D  getfield        002E    (Fetch field from object)
                000010  iload   04              (Load integer from local variable)
                000012  aaload                 (Load object reference from array)
                000013  iload   06              (Load integer from local variable)
                000015  aaload                 (Load object reference from array)
                000016  iload   05              (Load integer from local variable)
                000018  iaload                 (Load integer from array)
                000019  iconst_1                (Push integer constant)
                00001A  if_icmpne       001A    (Branch if integers not equal)
                00001D  aload_0                 (Load object reference from local variable)
                00001E  aload_1                 (Load object reference from local variable)
                00001F  iload_2                 (Load integer from local variable)
                000020  aload_0                 (Load object reference from local variable)
                000021  getfield        001C    (Fetch field from object)
                000024  iload   05              (Load integer from local variable)
                000026  imul                   (Integer multiply)
                000027  iadd                   (Integer add)
                000028  iload_3                 (Load integer from local variable)
                000029  aload_0                 (Load object reference from local variable)
                00002A  getfield        001C    (Fetch field from object)
                00002D  iload   06              (Load integer from local variable)
                00002F  imul                   (Integer multiply)
                000030  iadd                   (Integer add)
                000031  invokevirtual   0027    (Invoke instance method)
                000034  iinc    06 01           (Increment local variable by constant)
                000037  iload   06              (Load integer from local variable)
                000039  bipush  07              (Push one-byte signed integer)
                00003B  if_icmplt       FFD1    (Branch if integer less than)
                00003E  iinc    05 01           (Increment local variable by constant)
                000041  iload   05              (Load integer from local variable)
                000043  iconst_5                (Push integer constant)
                000044  if_icmplt       FFC2    (Branch if integer less than)
                000047  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00000BD8  type = LineNumberTable , attribute_length = 30

                        00 07 00 00 00 69 00 06 00 6A 00 0C 00 6B 00 1D 00 6C 00 34
                        00 6A 00 3E 00 69 00 47 00 68

00000BFA    method[9] : ACC_PUBLIC , Name = <init> , Signature = ()V , Attribute = 1

        type = Code , length = 2091 , max_stack = 11 , max_locals = 1 , code_length = 1971

                000000  aload_0                 (Load object reference from local variable)
                000001  invokenonvirt   0019    (Invoke instance method, dispatching based on compile-time type)
                000004  aload_0                 (Load object reference from local variable)
                000005  iconst_1                (Push integer constant)
                000006  putfield        0033    (Set field in object)
                000009  aload_0                 (Load object reference from local variable)
                00000A  bipush  46              (Push one-byte signed integer)
                00000C  putfield        0030    (Set field in object)
                00000F  aload_0                 (Load object reference from local variable)
                000010  bipush  19              (Push one-byte signed integer)
                000012  putfield        002B    (Set field in object)
                000015  aload_0                 (Load object reference from local variable)
                000016  bipush  0A              (Push one-byte signed integer)
                000018  putfield        0023    (Set field in object)
                00001B  aload_0                 (Load object reference from local variable)
```

〈リスト8-11〉 sample.classのrejava出力⑭

```
00001C  bipush  0C              (Push one-byte signed integer)
00001E  putfield        001C    (Set field in object)
000021  aload_0                 (Load object reference from local variable)
000022  iconst_1                (Push integer constant)
000023  putfield        0015    (Set field in object)
000026  aload_0                 (Load object reference from local variable)
000027  bipush  0F              (Push one-byte signed integer)
000029  anewarray       000B    (Allocate new array of references to objects)
00002C  putfield        0025    (Set field in object)
00002F  aload_0                 (Load object reference from local variable)
000030  bipush  0A              (Push one-byte signed integer)
000032  anewarray       000E    (Allocate new array of references to objects)
000035  dup                     (Duplicate top stack word)
000036  iconst_0                (Push integer constant)
000037  bipush  07              (Push one-byte signed integer)
000039  anewarray       0004    (Allocate new array of references to objects)
00003C  dup                     (Duplicate top stack word)
00003D  iconst_0                (Push integer constant)
00003E  iconst_5                (Push integer constant)
00003F  newarray        0A      (Allocate new array) : array type = T_INT
000041  dup                     (Duplicate top stack word)
000042  iconst_0                (Push integer constant)
000043  iconst_0                (Push integer constant)
000044  iastore                 (Store into integer array)
000045  dup                     (Duplicate top stack word)
000046  iconst_1                (Push integer constant)
000047  iconst_1                (Push integer constant)
000048  iastore                 (Store into integer array)
000049  dup                     (Duplicate top stack word)
00004A  iconst_2                (Push integer constant)
00004B  iconst_1                (Push integer constant)
00004C  iastore                 (Store into integer array)
00004D  dup                     (Duplicate top stack word)
00004E  iconst_3                (Push integer constant)
00004F  iconst_1                (Push integer constant)
000050  iastore                 (Store into integer array)
000051  dup                     (Duplicate top stack word)
000052  iconst_4                (Push integer constant)
000053  iconst_0                (Push integer constant)
000054  iastore                 (Store into integer array)
000055  aastore                 (Store into object reference array)
000056  dup                     (Duplicate top stack word)
000057  iconst_1                (Push integer constant)
000058  iconst_5                (Push integer constant)
000059  newarray        0A      (Allocate new array) : array type = T_INT
00005B  dup                     (Duplicate top stack word)
00005C  iconst_0                (Push integer constant)
00005D  iconst_1                (Push integer constant)
00005E  iastore                 (Store into integer array)
00005F  dup                     (Duplicate top stack word)
000060  iconst_1                (Push integer constant)
000061  iconst_0                (Push integer constant)
000062  iastore                 (Store into integer array)
000063  dup                     (Duplicate top stack word)
000064  iconst_2                (Push integer constant)
000065  iconst_0                (Push integer constant)
000066  iastore                 (Store into integer array)
000067  dup                     (Duplicate top stack word)
000068  iconst_3                (Push integer constant)
000069  iconst_0                (Push integer constant)
00006A  iastore                 (Store into integer array)
00006B  dup                     (Duplicate top stack word)
00006C  iconst_4                (Push integer constant)
00006D  iconst_1                (Push integer constant)
00006E  iastore                 (Store into integer array)
00006F  aastore                 (Store into object reference array)
000070  dup                     (Duplicate top stack word)
000071  iconst_2                (Push integer constant)
000072  iconst_5                (Push integer constant)
000073  newarray        0A      (Allocate new array) : array type = T_INT
000075  dup                     (Duplicate top stack word)
000076  iconst_0                (Push integer constant)
000077  iconst_1                (Push integer constant)
000078  iastore                 (Store into integer array)
000079  dup                     (Duplicate top stack word)
00007A  iconst_1                (Push integer constant)
00007B  iconst_0                (Push integer constant)
00007C  iastore                 (Store into integer array)
00007D  dup                     (Duplicate top stack word)
00007E  iconst_2                (Push integer constant)
00007F  iconst_0                (Push integer constant)
000080  iastore                 (Store into integer array)
000081  dup                     (Duplicate top stack word)
000082  iconst_3                (Push integer constant)
```

〈リスト8-11〉sample.classのrejava出力⑮

```
000083  iconst_0                (Push integer constant)
000084  iastore                 (Store into integer array)
000085  dup                     (Duplicate top stack word)
000086  iconst_4                (Push integer constant)
000087  iconst_1                (Push integer constant)
000088  iastore                 (Store into integer array)
000089  aastore                 (Store into object reference array)
00008A  dup                     (Duplicate top stack word)
00008B  iconst_3                (Push integer constant)
00008C  iconst_5                (Push integer constant)
00008D  newarray        0A      (Allocate new array) : array type = T_INT
00008F  dup                     (Duplicate top stack word)
000090  iconst_0                (Push integer constant)
000091  iconst_1                (Push integer constant)
000092  iastore                 (Store into integer array)
000093  dup                     (Duplicate top stack word)
000094  iconst_1                (Push integer constant)
000095  iconst_0                (Push integer constant)
000096  iastore                 (Store into integer array)
000097  dup                     (Duplicate top stack word)
000098  iconst_2                (Push integer constant)
000099  iconst_0                (Push integer constant)
00009A  iastore                 (Store into integer array)
00009B  dup                     (Duplicate top stack word)
00009C  iconst_3                (Push integer constant)
00009D  iconst_0                (Push integer constant)
00009E  iastore                 (Store into integer array)
00009F  dup                     (Duplicate top stack word)
0000A0  iconst_4                (Push integer constant)
0000A1  iconst_1                (Push integer constant)
0000A2  iastore                 (Store into integer array)
0000A3  aastore                 (Store into object reference array)
0000A4  dup                     (Duplicate top stack word)
0000A5  iconst_4                (Push integer constant)
0000A6  iconst_5                (Push integer constant)
0000A7  newarray        0A      (Allocate new array) : array type = T_INT
0000A9  dup                     (Duplicate top stack word)
0000AA  iconst_0                (Push integer constant)
0000AB  iconst_1                (Push integer constant)
0000AC  iastore                 (Store into integer array)
0000AD  dup                     (Duplicate top stack word)
0000AE  iconst_1                (Push integer constant)
0000AF  iconst_0                (Push integer constant)
0000B0  iastore                 (Store into integer array)
0000B1  dup                     (Duplicate top stack word)
0000B2  iconst_2                (Push integer constant)
0000B3  iconst_0                (Push integer constant)
0000B4  iastore                 (Store into integer array)
0000B5  dup                     (Duplicate top stack word)
0000B6  iconst_3                (Push integer constant)
0000B7  iconst_0                (Push integer constant)
0000B8  iastore                 (Store into integer array)
0000B9  dup                     (Duplicate top stack word)
0000BA  iconst_4                (Push integer constant)
0000BB  iconst_1                (Push integer constant)
0000BC  iastore                 (Store into integer array)
0000BD  aastore                 (Store into object reference array)
0000BE  dup                     (Duplicate top stack word)
0000BF  iconst_5                (Push integer constant)
0000C0  iconst_5                (Push integer constant)
0000C1  newarray        0A      (Allocate new array) : array type = T_INT
0000C3  dup                     (Duplicate top stack word)
0000C4  iconst_0                (Push integer constant)
0000C5  iconst_1                (Push integer constant)
0000C6  iastore                 (Store into integer array)
0000C7  dup                     (Duplicate top stack word)
0000C8  iconst_1                (Push integer constant)
0000C9  iconst_0                (Push integer constant)
0000CA  iastore                 (Store into integer array)
0000CB  dup                     (Duplicate top stack word)
0000CC  iconst_2                (Push integer constant)
0000CD  iconst_0                (Push integer constant)
0000CE  iastore                 (Store into integer array)
0000CF  dup                     (Duplicate top stack word)
0000D0  iconst_3                (Push integer constant)
0000D1  iconst_0                (Push integer constant)
0000D2  iastore                 (Store into integer array)
0000D3  dup                     (Duplicate top stack word)
0000D4  iconst_4                (Push integer constant)
0000D5  iconst_1                (Push integer constant)
0000D6  iastore                 (Store into integer array)
0000D7  aastore                 (Store into object reference array)
0000D8  dup                     (Duplicate top stack word)
0000D9  bipush   06             (Push one-byte signed integer)
```

〈リスト8-11〉sample.classのrejava出力⑯

```
0000DB  iconst_5                (Push integer constant)
0000DC  newarray        0A      (Allocate new array) : array type = T_INT
0000DE  dup                     (Duplicate top stack word)
0000DF  iconst_0                (Push integer constant)
0000E0  iconst_0                (Push integer constant)
0000E1  iastore                 (Store into integer array)
0000E2  dup                     (Duplicate top stack word)
0000E3  iconst_1                (Push integer constant)
0000E4  iconst_1                (Push integer constant)
0000E5  iastore                 (Store into integer array)
0000E6  dup                     (Duplicate top stack word)
0000E7  iconst_2                (Push integer constant)
0000E8  iconst_1                (Push integer constant)
0000E9  iastore                 (Store into integer array)
0000EA  dup                     (Duplicate top stack word)
0000EB  iconst_3                (Push integer constant)
0000EC  iconst_1                (Push integer constant)
0000ED  iastore                 (Store into integer array)
0000EE  dup                     (Duplicate top stack word)
0000EF  iconst_4                (Push integer constant)
0000F0  iconst_0                (Push integer constant)
0000F1  iastore                 (Store into integer array)
0000F2  aastore                 (Store into object reference array)
0000F3  aastore                 (Store into object reference array)
0000F4  dup                     (Duplicate top stack word)
0000F5  iconst_1                (Push integer constant)
0000F6  bipush  07              (Push one-byte signed integer)
0000F8  anewarray       0004    (Allocate new array of references to objects)
0000FB  dup                     (Duplicate top stack word)
0000FC  iconst_0                (Push integer constant)
0000FD  iconst_5                (Push integer constant)
0000FE  newarray        0A      (Allocate new array) : array type = T_INT
000100  dup                     (Duplicate top stack word)
000101  iconst_0                (Push integer constant)
000102  iconst_0                (Push integer constant)
000103  iastore                 (Store into integer array)
000104  dup                     (Duplicate top stack word)
000105  iconst_1                (Push integer constant)
000106  iconst_0                (Push integer constant)
000107  iastore                 (Store into integer array)
000108  dup                     (Duplicate top stack word)
000109  iconst_2                (Push integer constant)
00010A  iconst_1                (Push integer constant)
00010B  iastore                 (Store into integer array)
00010C  dup                     (Duplicate top stack word)
00010D  iconst_3                (Push integer constant)
00010E  iconst_0                (Push integer constant)
00010F  iastore                 (Store into integer array)
000110  dup                     (Duplicate top stack word)
000111  iconst_4                (Push integer constant)
000112  iconst_0                (Push integer constant)
000113  iastore                 (Store into integer array)
000114  aastore                 (Store into object reference array)
000115  dup                     (Duplicate top stack word)
000116  iconst_1                (Push integer constant)
000117  iconst_5                (Push integer constant)
000118  newarray        0A      (Allocate new array) : array type = T_INT
00011A  dup                     (Duplicate top stack word)
00011B  iconst_0                (Push integer constant)
00011C  iconst_0                (Push integer constant)
00011D  iastore                 (Store into integer array)
00011E  dup                     (Duplicate top stack word)
00011F  iconst_1                (Push integer constant)
000120  iconst_1                (Push integer constant)
000121  iastore                 (Store into integer array)
000122  dup                     (Duplicate top stack word)
000123  iconst_2                (Push integer constant)
000124  iconst_1                (Push integer constant)
000125  iastore                 (Store into integer array)
000126  dup                     (Duplicate top stack word)
000127  iconst_3                (Push integer constant)
000128  iconst_0                (Push integer constant)
000129  iastore                 (Store into integer array)
00012A  dup                     (Duplicate top stack word)
00012B  iconst_4                (Push integer constant)
00012C  iconst_0                (Push integer constant)
00012D  iastore                 (Store into integer array)
00012E  aastore                 (Store into object reference array)
00012F  dup                     (Duplicate top stack word)
000130  iconst_2                (Push integer constant)
000131  iconst_5                (Push integer constant)
000132  newarray        0A      (Allocate new array) : array type = T_INT
000134  dup                     (Duplicate top stack word)
000135  iconst_0                (Push integer constant)
```

〈リスト8-11〉sample.classのrejava出力⑰

```
000136  iconst_0              (Push integer constant)
000137  iastore               (Store into integer array)
000138  dup                   (Duplicate top stack word)
000139  iconst_1              (Push integer constant)
00013A  iconst_0              (Push integer constant)
00013B  iastore               (Store into integer array)
00013C  dup                   (Duplicate top stack word)
00013D  iconst_2              (Push integer constant)
00013E  iconst_1              (Push integer constant)
00013F  iastore               (Store into integer array)
000140  dup                   (Duplicate top stack word)
000141  iconst_3              (Push integer constant)
000142  iconst_0              (Push integer constant)
000143  iastore               (Store into integer array)
000144  dup                   (Duplicate top stack word)
000145  iconst_4              (Push integer constant)
000146  iconst_0              (Push integer constant)
000147  iastore               (Store into integer array)
000148  aastore               (Store into object reference array)
000149  dup                   (Duplicate top stack word)
00014A  iconst_3              (Push integer constant)
00014B  iconst_5              (Push integer constant)
00014C  newarray        0A    (Allocate new array) : array type = T_INT
00014E  dup                   (Duplicate top stack word)
00014F  iconst_0              (Push integer constant)
000150  iconst_0              (Push integer constant)
000151  iastore               (Store into integer array)
000152  dup                   (Duplicate top stack word)
000153  iconst_1              (Push integer constant)
000154  iconst_0              (Push integer constant)
000155  iastore               (Store into integer array)
000156  dup                   (Duplicate top stack word)
000157  iconst_2              (Push integer constant)
000158  iconst_1              (Push integer constant)
000159  iastore               (Store into integer array)
00015A  dup                   (Duplicate top stack word)
00015B  iconst_3              (Push integer constant)
00015C  iconst_0              (Push integer constant)
00015D  iastore               (Store into integer array)
00015E  dup                   (Duplicate top stack word)
00015F  iconst_4              (Push integer constant)
000160  iconst_0              (Push integer constant)
000161  iastore               (Store into integer array)
000162  aastore               (Store into object reference array)
000163  dup                   (Duplicate top stack word)
000164  iconst_4              (Push integer constant)
000165  iconst_5              (Push integer constant)
000166  newarray        0A    (Allocate new array) : array type = T_INT
000168  dup                   (Duplicate top stack word)
000169  iconst_0              (Push integer constant)
00016A  iconst_0              (Push integer constant)
00016B  iastore               (Store into integer array)
00016C  dup                   (Duplicate top stack word)
00016D  iconst_1              (Push integer constant)
00016E  iconst_0              (Push integer constant)
00016F  iastore               (Store into integer array)
000170  dup                   (Duplicate top stack word)
000171  iconst_2              (Push integer constant)
000172  iconst_1              (Push integer constant)
000173  iastore               (Store into integer array)
000174  dup                   (Duplicate top stack word)
000175  iconst_3              (Push integer constant)
000176  iconst_0              (Push integer constant)
000177  iastore               (Store into integer array)
000178  dup                   (Duplicate top stack word)
000179  iconst_4              (Push integer constant)
00017A  iconst_0              (Push integer constant)
00017B  iastore               (Store into integer array)
00017C  aastore               (Store into object reference array)
00017D  dup                   (Duplicate top stack word)
00017E  iconst_5              (Push integer constant)
00017F  iconst_5              (Push integer constant)
000180  newarray        0A    (Allocate new array) : array type = T_INT
000182  dup                   (Duplicate top stack word)
000183  iconst_0              (Push integer constant)
000184  iconst_0              (Push integer constant)
000185  iastore               (Store into integer array)
000186  dup                   (Duplicate top stack word)
000187  iconst_1              (Push integer constant)
000188  iconst_0              (Push integer constant)
000189  iastore               (Store into integer array)
00018A  dup                   (Duplicate top stack word)
00018B  iconst_2              (Push integer constant)
00018C  iconst_1              (Push integer constant)
```

〈リスト8-11〉 sample.classのrejava出力⑱

```
00018D  iastore                 (Store into integer array)
00018E  dup                     (Duplicate top stack word)
00018F  iconst_3                (Push integer constant)
000190  iconst_0                (Push integer constant)
000191  iastore                 (Store into integer array)
000192  dup                     (Duplicate top stack word)
000193  iconst_4                (Push integer constant)
000194  iconst_0                (Push integer constant)
000195  iastore                 (Store into integer array)
000196  aastore                 (Store into object reference array)
000197  dup                     (Duplicate top stack word)
000198  bipush  06              (Push one-byte signed integer)
00019A  iconst_5                (Push integer constant)
00019B  newarray        0A      (Allocate new array) : array type = T_INT
00019D  dup                     (Duplicate top stack word)
00019E  iconst_0                (Push integer constant)
00019F  iconst_0                (Push integer constant)
0001A0  iastore                 (Store into integer array)
0001A1  dup                     (Duplicate top stack word)
0001A2  iconst_1                (Push integer constant)
0001A3  iconst_1                (Push integer constant)
0001A4  iastore                 (Store into integer array)
0001A5  dup                     (Duplicate top stack word)
0001A6  iconst_2                (Push integer constant)
0001A7  iconst_1                (Push integer constant)
0001A8  iastore                 (Store into integer array)
0001A9  dup                     (Duplicate top stack word)
0001AA  iconst_3                (Push integer constant)
0001AB  iconst_1                (Push integer constant)
0001AC  iastore                 (Store into integer array)
0001AD  dup                     (Duplicate top stack word)
0001AE  iconst_4                (Push integer constant)
0001AF  iconst_0                (Push integer constant)
0001B0  iastore                 (Store into integer array)
0001B1  aastore                 (Store into object reference array)
0001B2  aastore                 (Store into object reference array)
0001B3  dup                     (Duplicate top stack word)
0001B4  iconst_2                (Push integer constant)
0001B5  bipush  07              (Push one-byte signed integer)
0001B7  anewarray       0004    (Allocate new array of references to objects)
0001BA  dup                     (Duplicate top stack word)
0001BB  iconst_0                (Push integer constant)
0001BC  iconst_5                (Push integer constant)
0001BD  newarray        0A      (Allocate new array) : array type = T_INT
0001BF  dup                     (Duplicate top stack word)
0001C0  iconst_0                (Push integer constant)
0001C1  iconst_0                (Push integer constant)
0001C2  iastore                 (Store into integer array)
0001C3  dup                     (Duplicate top stack word)
0001C4  iconst_1                (Push integer constant)
0001C5  iconst_1                (Push integer constant)
0001C6  iastore                 (Store into integer array)
0001C7  dup                     (Duplicate top stack word)
0001C8  iconst_2                (Push integer constant)
0001C9  iconst_1                (Push integer constant)
0001CA  iastore                 (Store into integer array)
0001CB  dup                     (Duplicate top stack word)
0001CC  iconst_3                (Push integer constant)
0001CD  iconst_1                (Push integer constant)
0001CE  iastore                 (Store into integer array)
0001CF  dup                     (Duplicate top stack word)
0001D0  iconst_4                (Push integer constant)
0001D1  iconst_0                (Push integer constant)
0001D2  iastore                 (Store into integer array)
0001D3  aastore                 (Store into object reference array)
0001D4  dup                     (Duplicate top stack word)
0001D5  iconst_1                (Push integer constant)
0001D6  iconst_5                (Push integer constant)
0001D7  newarray        0A      (Allocate new array) : array type = T_INT
0001D9  dup                     (Duplicate top stack word)
0001DA  iconst_0                (Push integer constant)
0001DB  iconst_1                (Push integer constant)
0001DC  iastore                 (Store into integer array)
0001DD  dup                     (Duplicate top stack word)
0001DE  iconst_1                (Push integer constant)
0001DF  iconst_0                (Push integer constant)
0001E0  iastore                 (Store into integer array)
0001E1  dup                     (Duplicate top stack word)
0001E2  iconst_2                (Push integer constant)
0001E3  iconst_0                (Push integer constant)
0001E4  iastore                 (Store into integer array)
0001E5  dup                     (Duplicate top stack word)
0001E6  iconst_3                (Push integer constant)
0001E7  iconst_0                (Push integer constant)
```

〈リスト8-11〉sample.classのrejava出力⑲

```
0001E8  iastore                 (Store into integer array)
0001E9  dup                     (Duplicate top stack word)
0001EA  iconst_4                (Push integer constant)
0001EB  iconst_1                (Push integer constant)
0001EC  iastore                 (Store into integer array)
0001ED  aastore                 (Store into object reference array)
0001EE  dup                     (Duplicate top stack word)
0001EF  iconst_2                (Push integer constant)
0001F0  iconst_5                (Push integer constant)
0001F1  newarray        0A      (Allocate new array) : array type = T_INT
0001F3  dup                     (Duplicate top stack word)
0001F4  iconst_0                (Push integer constant)
0001F5  iconst_1                (Push integer constant)
0001F6  iastore                 (Store into integer array)
0001F7  dup                     (Duplicate top stack word)
0001F8  iconst_1                (Push integer constant)
0001F9  iconst_0                (Push integer constant)
0001FA  iastore                 (Store into integer array)
0001FB  dup                     (Duplicate top stack word)
0001FC  iconst_2                (Push integer constant)
0001FD  iconst_0                (Push integer constant)
0001FE  iastore                 (Store into integer array)
0001FF  dup                     (Duplicate top stack word)
000200  iconst_3                (Push integer constant)
000201  iconst_0                (Push integer constant)
000202  iastore                 (Store into integer array)
000203  dup                     (Duplicate top stack word)
000204  iconst_4                (Push integer constant)
000205  iconst_1                (Push integer constant)
000206  iastore                 (Store into integer array)
000207  aastore                 (Store into object reference array)
000208  dup                     (Duplicate top stack word)
000209  iconst_3                (Push integer constant)
00020A  iconst_5                (Push integer constant)
00020B  newarray        0A      (Allocate new array) : array type = T_INT
00020D  dup                     (Duplicate top stack word)
00020E  iconst_0                (Push integer constant)
00020F  iconst_0                (Push integer constant)
000210  iastore                 (Store into integer array)
000211  dup                     (Duplicate top stack word)
000212  iconst_1                (Push integer constant)
000213  iconst_0                (Push integer constant)
000214  iastore                 (Store into integer array)
000215  dup                     (Duplicate top stack word)
000216  iconst_2                (Push integer constant)
000217  iconst_0                (Push integer constant)
000218  iastore                 (Store into integer array)
000219  dup                     (Duplicate top stack word)
00021A  iconst_3                (Push integer constant)
00021B  iconst_1                (Push integer constant)
00021C  iastore                 (Store into integer array)
00021D  dup                     (Duplicate top stack word)
00021E  iconst_4                (Push integer constant)
00021F  iconst_0                (Push integer constant)
000220  iastore                 (Store into integer array)
000221  aastore                 (Store into object reference array)
000222  dup                     (Duplicate top stack word)
000223  iconst_4                (Push integer constant)
000224  iconst_5                (Push integer constant)
000225  newarray        0A      (Allocate new array) : array type = T_INT
000227  dup                     (Duplicate top stack word)
000228  iconst_0                (Push integer constant)
000229  iconst_0                (Push integer constant)
00022A  iastore                 (Store into integer array)
00022B  dup                     (Duplicate top stack word)
00022C  iconst_1                (Push integer constant)
00022D  iconst_0                (Push integer constant)
00022E  iastore                 (Store into integer array)
00022F  dup                     (Duplicate top stack word)
000230  iconst_2                (Push integer constant)
000231  iconst_1                (Push integer constant)
000232  iastore                 (Store into integer array)
000233  dup                     (Duplicate top stack word)
000234  iconst_3                (Push integer constant)
000235  iconst_0                (Push integer constant)
000236  iastore                 (Store into integer array)
000237  dup                     (Duplicate top stack word)
000238  iconst_4                (Push integer constant)
000239  iconst_0                (Push integer constant)
00023A  iastore                 (Store into integer array)
00023B  aastore                 (Store into object reference array)
00023C  dup                     (Duplicate top stack word)
00023D  iconst_5                (Push integer constant)
00023E  iconst_5                (Push integer constant)
```

〈リスト8-11〉 sample.classのrejava出力⑳

```
00023F  newarray        0A      (Allocate new array) : array type = T_INT
000241  dup                     (Duplicate top stack word)
000242  iconst_0                (Push integer constant)
000243  iconst_0                (Push integer constant)
000244  iastore                 (Store into integer array)
000245  dup                     (Duplicate top stack word)
000246  iconst_1                (Push integer constant)
000247  iconst_1                (Push integer constant)
000248  iastore                 (Store into integer array)
000249  dup                     (Duplicate top stack word)
00024A  iconst_2                (Push integer constant)
00024B  iconst_0                (Push integer constant)
00024C  iastore                 (Store into integer array)
00024D  dup                     (Duplicate top stack word)
00024E  iconst_3                (Push integer constant)
00024F  iconst_0                (Push integer constant)
000250  iastore                 (Store into integer array)
000251  dup                     (Duplicate top stack word)
000252  iconst_4                (Push integer constant)
000253  iconst_0                (Push integer constant)
000254  iastore                 (Store into integer array)
000255  aastore                 (Store into object reference array)
000256  dup                     (Duplicate top stack word)
000257  bipush  06              (Push one-byte signed integer)
000259  iconst_5                (Push integer constant)
00025A  newarray        0A      (Allocate new array) : array type = T_INT
00025C  dup                     (Duplicate top stack word)
00025D  iconst_0                (Push integer constant)
00025E  iconst_1                (Push integer constant)
00025F  iastore                 (Store into integer array)
000260  dup                     (Duplicate top stack word)
000261  iconst_1                (Push integer constant)
000262  iconst_1                (Push integer constant)
000263  iastore                 (Store into integer array)
000264  dup                     (Duplicate top stack word)
000265  iconst_2                (Push integer constant)
000266  iconst_1                (Push integer constant)
000267  iastore                 (Store into integer array)
000268  dup                     (Duplicate top stack word)
000269  iconst_3                (Push integer constant)
00026A  iconst_1                (Push integer constant)
00026B  iastore                 (Store into integer array)
00026C  dup                     (Duplicate top stack word)
00026D  iconst_4                (Push integer constant)
00026E  iconst_1                (Push integer constant)
00026F  iastore                 (Store into integer array)
000270  aastore                 (Store into object reference array)
000271  aastore                 (Store into object reference array)
000272  dup                     (Duplicate top stack word)
000273  iconst_3                (Push integer constant)
000274  bipush  07              (Push one-byte signed integer)
000276  anewarray       0004    (Allocate new array of references to objects)
000279  dup                     (Duplicate top stack word)
00027A  iconst_0                (Push integer constant)
00027B  iconst_5                (Push integer constant)
00027C  newarray        0A      (Allocate new array) : array type = T_INT
00027E  dup                     (Duplicate top stack word)
00027F  iconst_0                (Push integer constant)
000280  iconst_0                (Push integer constant)
000281  iastore                 (Store into integer array)
000282  dup                     (Duplicate top stack word)
000283  iconst_1                (Push integer constant)
000284  iconst_1                (Push integer constant)
000285  iastore                 (Store into integer array)
000286  dup                     (Duplicate top stack word)
000287  iconst_2                (Push integer constant)
000288  iconst_1                (Push integer constant)
000289  iastore                 (Store into integer array)
00028A  dup                     (Duplicate top stack word)
00028B  iconst_3                (Push integer constant)
00028C  iconst_1                (Push integer constant)
00028D  iastore                 (Store into integer array)
00028E  dup                     (Duplicate top stack word)
00028F  iconst_4                (Push integer constant)
000290  iconst_0                (Push integer constant)
000291  iastore                 (Store into integer array)
000292  aastore                 (Store into object reference array)
000293  dup                     (Duplicate top stack word)
000294  iconst_1                (Push integer constant)
000295  iconst_5                (Push integer constant)
000296  newarray        0A      (Allocate new array) : array type = T_INT
000298  dup                     (Duplicate top stack word)
000299  iconst_0                (Push integer constant)
00029A  iconst_1                (Push integer constant)
```

〈リスト8-11〉sample.classのrejava出力㉑

```
00029B  iastore                 (Store into integer array)
00029C  dup                     (Duplicate top stack word)
00029D  iconst_1                (Push integer constant)
00029E  iconst_0                (Push integer constant)
00029F  iastore                 (Store into integer array)
0002A0  dup                     (Duplicate top stack word)
0002A1  iconst_2                (Push integer constant)
0002A2  iconst_0                (Push integer constant)
0002A3  iastore                 (Store into integer array)
0002A4  dup                     (Duplicate top stack word)
0002A5  iconst_3                (Push integer constant)
0002A6  iconst_0                (Push integer constant)
0002A7  iastore                 (Store into integer array)
0002A8  dup                     (Duplicate top stack word)
0002A9  iconst_4                (Push integer constant)
0002AA  iconst_1                (Push integer constant)
0002AB  iastore                 (Store into integer array)
0002AC  aastore                 (Store into object reference array)
0002AD  dup                     (Duplicate top stack word)
0002AE  iconst_2                (Push integer constant)
0002AF  iconst_5                (Push integer constant)
0002B0  newarray        0A      (Allocate new array) : array type = T_INT
0002B2  dup                     (Duplicate top stack word)
0002B3  iconst_0                (Push integer constant)
0002B4  iconst_0                (Push integer constant)
0002B5  iastore                 (Store into integer array)
0002B6  dup                     (Duplicate top stack word)
0002B7  iconst_1                (Push integer constant)
0002B8  iconst_0                (Push integer constant)
0002B9  iastore                 (Store into integer array)
0002BA  dup                     (Duplicate top stack word)
0002BB  iconst_2                (Push integer constant)
0002BC  iconst_0                (Push integer constant)
0002BD  iastore                 (Store into integer array)
0002BE  dup                     (Duplicate top stack word)
0002BF  iconst_3                (Push integer constant)
0002C0  iconst_0                (Push integer constant)
0002C1  iastore                 (Store into integer array)
0002C2  dup                     (Duplicate top stack word)
0002C3  iconst_4                (Push integer constant)
0002C4  iconst_1                (Push integer constant)
0002C5  iastore                 (Store into integer array)
0002C6  aastore                 (Store into object reference array)
0002C7  dup                     (Duplicate top stack word)
0002C8  iconst_3                (Push integer constant)
0002C9  iconst_5                (Push integer constant)
0002CA  newarray        0A      (Allocate new array) : array type = T_INT
0002CC  dup                     (Duplicate top stack word)
0002CD  iconst_0                (Push integer constant)
0002CE  iconst_0                (Push integer constant)
0002CF  iastore                 (Store into integer array)
0002D0  dup                     (Duplicate top stack word)
0002D1  iconst_1                (Push integer constant)
0002D2  iconst_0                (Push integer constant)
0002D3  iastore                 (Store into integer array)
0002D4  dup                     (Duplicate top stack word)
0002D5  iconst_2                (Push integer constant)
0002D6  iconst_1                (Push integer constant)
0002D7  iastore                 (Store into integer array)
0002D8  dup                     (Duplicate top stack word)
0002D9  iconst_3                (Push integer constant)
0002DA  iconst_1                (Push integer constant)
0002DB  iastore                 (Store into integer array)
0002DC  dup                     (Duplicate top stack word)
0002DD  iconst_4                (Push integer constant)
0002DE  iconst_0                (Push integer constant)
0002DF  iastore                 (Store into integer array)
0002E0  aastore                 (Store into object reference array)
0002E1  dup                     (Duplicate top stack word)
0002E2  iconst_4                (Push integer constant)
0002E3  iconst_5                (Push integer constant)
0002E4  newarray        0A      (Allocate new array) : array type = T_INT
0002E6  dup                     (Duplicate top stack word)
0002E7  iconst_0                (Push integer constant)
0002E8  iconst_0                (Push integer constant)
0002E9  iastore                 (Store into integer array)
0002EA  dup                     (Duplicate top stack word)
0002EB  iconst_1                (Push integer constant)
0002EC  iconst_0                (Push integer constant)
0002ED  iastore                 (Store into integer array)
0002EE  dup                     (Duplicate top stack word)
0002EF  iconst_2                (Push integer constant)
0002F0  iconst_0                (Push integer constant)
0002F1  iastore                 (Store into integer array)
```

〈リスト8-11〉sample.classのrejava出力㉒

```
0002F2  dup                     (Duplicate top stack word)
0002F3  iconst_3                (Push integer constant)
0002F4  iconst_0                (Push integer constant)
0002F5  iastore                 (Store into integer array)
0002F6  dup                     (Duplicate top stack word)
0002F7  iconst_4                (Push integer constant)
0002F8  iconst_1                (Push integer constant)
0002F9  iastore                 (Store into integer array)
0002FA  aastore                 (Store into object reference array)
0002FB  dup                     (Duplicate top stack word)
0002FC  iconst_5                (Push integer constant)
0002FD  iconst_5                (Push integer constant)
0002FE  newarray        0A      (Allocate new array) : array type = T_INT
000300  dup                     (Duplicate top stack word)
000301  iconst_0                (Push integer constant)
000302  iconst_1                (Push integer constant)
000303  iastore                 (Store into integer array)
000304  dup                     (Duplicate top stack word)
000305  iconst_1                (Push integer constant)
000306  iconst_0                (Push integer constant)
000307  iastore                 (Store into integer array)
000308  dup                     (Duplicate top stack word)
000309  iconst_2                (Push integer constant)
00030A  iconst_0                (Push integer constant)
00030B  iastore                 (Store into integer array)
00030C  dup                     (Duplicate top stack word)
00030D  iconst_3                (Push integer constant)
00030E  iconst_0                (Push integer constant)
00030F  iastore                 (Store into integer array)
000310  dup                     (Duplicate top stack word)
000311  iconst_4                (Push integer constant)
000312  iconst_1                (Push integer constant)
000313  iastore                 (Store into integer array)
000314  aastore                 (Store into object reference array)
000315  dup                     (Duplicate top stack word)
000316  bipush  06              (Push one-byte signed integer)
000318  iconst_5                (Push integer constant)
000319  newarray        0A      (Allocate new array) : array type = T_INT
00031B  dup                     (Duplicate top stack word)
00031C  iconst_0                (Push integer constant)
00031D  iconst_0                (Push integer constant)
00031E  iastore                 (Store into integer array)
00031F  dup                     (Duplicate top stack word)
000320  iconst_1                (Push integer constant)
000321  iconst_1                (Push integer constant)
000322  iastore                 (Store into integer array)
000323  dup                     (Duplicate top stack word)
000324  iconst_2                (Push integer constant)
000325  iconst_1                (Push integer constant)
000326  iastore                 (Store into integer array)
000327  dup                     (Duplicate top stack word)
000328  iconst_3                (Push integer constant)
000329  iconst_1                (Push integer constant)
00032A  iastore                 (Store into integer array)
00032B  dup                     (Duplicate top stack word)
00032C  iconst_4                (Push integer constant)
00032D  iconst_0                (Push integer constant)
00032E  iastore                 (Store into integer array)
00032F  aastore                 (Store into object reference array)
000330  aastore                 (Store into object reference array)
000331  dup                     (Duplicate top stack word)
000332  iconst_4                (Push integer constant)
000333  bipush  07              (Push one-byte signed integer)
000335  anewarray       0004    (Allocate new array of references to objects)
000338  dup                     (Duplicate top stack word)
000339  iconst_0                (Push integer constant)
00033A  iconst_5                (Push integer constant)
00033B  newarray        0A      (Allocate new array) : array type = T_INT
00033D  dup                     (Duplicate top stack word)
00033E  iconst_0                (Push integer constant)
00033F  iconst_0                (Push integer constant)
000340  iastore                 (Store into integer array)
000341  dup                     (Duplicate top stack word)
000342  iconst_1                (Push integer constant)
000343  iconst_0                (Push integer constant)
000344  iastore                 (Store into integer array)
000345  dup                     (Duplicate top stack word)
000346  iconst_2                (Push integer constant)
000347  iconst_0                (Push integer constant)
000348  iastore                 (Store into integer array)
000349  dup                     (Duplicate top stack word)
00034A  iconst_3                (Push integer constant)
00034B  iconst_1                (Push integer constant)
00034C  iastore                 (Store into integer array)
```

〈リスト8-11〉sample.classのrejava出力㉓

```
000350  iastore                 (Store into integer array)
000351  aastore                 (Store into object reference array)
000352  dup                     (Duplicate top stack word)
000353  iconst_1                (Push integer constant)
000354  iconst_5                (Push integer constant)
000355  newarray        0A      (Allocate new array) : array type = T_INT
000357  dup                     (Duplicate top stack word)
000358  iconst_0                (Push integer constant)
000359  iconst_0                (Push integer constant)
00035A  iastore                 (Store into integer array)
00035B  dup                     (Duplicate top stack word)
00035C  iconst_1                (Push integer constant)
00035D  iconst_0                (Push integer constant)
00035E  iastore                 (Store into integer array)
00035F  dup                     (Duplicate top stack word)
000360  iconst_2                (Push integer constant)
000361  iconst_1                (Push integer constant)
000362  iastore                 (Store into integer array)
000363  dup                     (Duplicate top stack word)
000364  iconst_3                (Push integer constant)
000365  iconst_1                (Push integer constant)
000366  iastore                 (Store into integer array)
000367  dup                     (Duplicate top stack word)
000368  iconst_4                (Push integer constant)
000369  iconst_0                (Push integer constant)
00036A  iastore                 (Store into integer array)
00036B  aastore                 (Store into object reference array)
00036C  dup                     (Duplicate top stack word)
00036D  iconst_2                (Push integer constant)
00036E  iconst_5                (Push integer constant)
00036F  newarray        0A      (Allocate new array) : array type = T_INT
000371  dup                     (Duplicate top stack word)
000372  iconst_0                (Push integer constant)
000373  iconst_0                (Push integer constant)
000374  iastore                 (Store into integer array)
000375  dup                     (Duplicate top stack word)
000376  iconst_1                (Push integer constant)
000377  iconst_1                (Push integer constant)
000378  iastore                 (Store into integer array)
000379  dup                     (Duplicate top stack word)
00037A  iconst_2                (Push integer constant)
00037B  iconst_0                (Push integer constant)
00037C  iastore                 (Store into integer array)
00037D  dup                     (Duplicate top stack word)
00037E  iconst_3                (Push integer constant)
00037F  iconst_1                (Push integer constant)
000380  iastore                 (Store into integer array)
000381  dup                     (Duplicate top stack word)
000382  iconst_4                (Push integer constant)
000383  iconst_0                (Push integer constant)
000384  iastore                 (Store into integer array)
000385  aastore                 (Store into object reference array)
000386  dup                     (Duplicate top stack word)
000387  iconst_3                (Push integer constant)
000388  iconst_5                (Push integer constant)
000389  newarray        0A      (Allocate new array) : array type = T_INT
00038B  dup                     (Duplicate top stack word)
00038C  iconst_0                (Push integer constant)
00038D  iconst_1                (Push integer constant)
00038E  iastore                 (Store into integer array)
00038F  dup                     (Duplicate top stack word)
000390  iconst_1                (Push integer constant)
000391  iconst_0                (Push integer constant)
000392  iastore                 (Store into integer array)
000393  dup                     (Duplicate top stack word)
000394  iconst_2                (Push integer constant)
000395  iconst_0                (Push integer constant)
000396  iastore                 (Store into integer array)
000397  dup                     (Duplicate top stack word)
000398  iconst_3                (Push integer constant)
000399  iconst_1                (Push integer constant)
00039A  iastore                 (Store into integer array)
00039B  dup                     (Duplicate top stack word)
00039C  iconst_4                (Push integer constant)
00039D  iconst_0                (Push integer constant)
00039E  iastore                 (Store into integer array)
00039F  aastore                 (Store into object reference array)
0003A0  dup                     (Duplicate top stack word)
0003A1  iconst_4                (Push integer constant)
0003A2  iconst_5                (Push integer constant)
0003A3  newarray        0A      (Allocate new array) : array type = T_INT
```

〈リスト8-11〉 sample.classのrejava出力㉔

```
0003A5  dup                 (Duplicate top stack word)
0003A6  iconst_0            (Push integer constant)
0003A7  iconst_1            (Push integer constant)
0003A8  iastore             (Store into integer array)
0003A9  dup                 (Duplicate top stack word)
0003AA  iconst_1            (Push integer constant)
0003AB  iconst_1            (Push integer constant)
0003AC  iastore             (Store into integer array)
0003AD  dup                 (Duplicate top stack word)
0003AE  iconst_2            (Push integer constant)
0003AF  iconst_1            (Push integer constant)
0003B0  iastore             (Store into integer array)
0003B1  dup                 (Duplicate top stack word)
0003B2  iconst_3            (Push integer constant)
0003B3  iconst_1            (Push integer constant)
0003B4  iastore             (Store into integer array)
0003B5  dup                 (Duplicate top stack word)
0003B6  iconst_4            (Push integer constant)
0003B7  iconst_1            (Push integer constant)
0003B8  iastore             (Store into integer array)
0003B9  aastore             (Store into object reference array)
0003BA  dup                 (Duplicate top stack word)
0003BB  iconst_5            (Push integer constant)
0003BC  iconst_5            (Push integer constant)
0003BD  newarray        0A  (Allocate new array) : array type = T_INT
0003BF  dup                 (Duplicate top stack word)
0003C0  iconst_0            (Push integer constant)
0003C1  iconst_0            (Push integer constant)
0003C2  iastore             (Store into integer array)
0003C3  dup                 (Duplicate top stack word)
0003C4  iconst_1            (Push integer constant)
0003C5  iconst_0            (Push integer constant)
0003C6  iastore             (Store into integer array)
0003C7  dup                 (Duplicate top stack word)
0003C8  iconst_2            (Push integer constant)
0003C9  iconst_0            (Push integer constant)
0003CA  iastore             (Store into integer array)
0003CB  dup                 (Duplicate top stack word)
0003CC  iconst_3            (Push integer constant)
0003CD  iconst_1            (Push integer constant)
0003CE  iastore             (Store into integer array)
0003CF  dup                 (Duplicate top stack word)
0003D0  iconst_4            (Push integer constant)
0003D1  iconst_0            (Push integer constant)
0003D2  iastore             (Store into integer array)
0003D3  aastore             (Store into object reference array)
0003D4  dup                 (Duplicate top stack word)
0003D5  bipush  06          (Push one-byte signed integer)
0003D7  iconst_5            (Push integer constant)
0003D8  newarray        0A  (Allocate new array) : array type = T_INT
0003DA  dup                 (Duplicate top stack word)
0003DB  iconst_0            (Push integer constant)
0003DC  iconst_0            (Push integer constant)
0003DD  iastore             (Store into integer array)
0003DE  dup                 (Duplicate top stack word)
0003DF  iconst_1            (Push integer constant)
0003E0  iconst_0            (Push integer constant)
0003E1  iastore             (Store into integer array)
0003E2  dup                 (Duplicate top stack word)
0003E3  iconst_2            (Push integer constant)
0003E4  iconst_0            (Push integer constant)
0003E5  iastore             (Store into integer array)
0003E6  dup                 (Duplicate top stack word)
0003E7  iconst_3            (Push integer constant)
0003E8  iconst_1            (Push integer constant)
0003E9  iastore             (Store into integer array)
0003EA  dup                 (Duplicate top stack word)
0003EB  iconst_4            (Push integer constant)
0003EC  iconst_0            (Push integer constant)
0003ED  iastore             (Store into integer array)
0003EE  aastore             (Store into object reference array)
0003EF  aastore             (Store into object reference array)
0003F0  dup                 (Duplicate top stack word)
0003F1  iconst_5            (Push integer constant)
0003F2  bipush  07          (Push one-byte signed integer)
0003F4  anewarray     0004  (Allocate new array of references to objects)
0003F7  dup                 (Duplicate top stack word)
0003F8  iconst_0            (Push integer constant)
0003F9  iconst_5            (Push integer constant)
0003FA  newarray        0A  (Allocate new array) : array type = T_INT
0003FC  dup                 (Duplicate top stack word)
0003FD  iconst_0            (Push integer constant)
0003FE  iconst_1            (Push integer constant)
0003FF  iastore             (Store into integer array)
```

〈リスト8-11〉sample.classのrejava出力㉕

```
000400  dup                 (Duplicate top stack word)
000401  iconst_1            (Push integer constant)
000402  iconst_1            (Push integer constant)
000403  iastore             (Store into integer array)
000404  dup                 (Duplicate top stack word)
000405  iconst_2            (Push integer constant)
000406  iconst_1            (Push integer constant)
000407  iastore             (Store into integer array)
000408  dup                 (Duplicate top stack word)
000409  iconst_3            (Push integer constant)
00040A  iconst_1            (Push integer constant)
00040B  iastore             (Store into integer array)
00040C  dup                 (Duplicate top stack word)
00040D  iconst_4            (Push integer constant)
00040E  iconst_1            (Push integer constant)
00040F  iastore             (Store into integer array)
000410  aastore             (Store into object reference array)
000411  dup                 (Duplicate top stack word)
000412  iconst_1            (Push integer constant)
000413  iconst_5            (Push integer constant)
000414  newarray        0A  (Allocate new array) : array type = T_INT
000416  dup                 (Duplicate top stack word)
000417  iconst_0            (Push integer constant)
000418  iconst_1            (Push integer constant)
000419  iastore             (Store into integer array)
00041A  dup                 (Duplicate top stack word)
00041B  iconst_1            (Push integer constant)
00041C  iconst_0            (Push integer constant)
00041D  iastore             (Store into integer array)
00041E  dup                 (Duplicate top stack word)
00041F  iconst_2            (Push integer constant)
000420  iconst_0            (Push integer constant)
000421  iastore             (Store into integer array)
000422  dup                 (Duplicate top stack word)
000423  iconst_3            (Push integer constant)
000424  iconst_0            (Push integer constant)
000425  iastore             (Store into integer array)
000426  dup                 (Duplicate top stack word)
000427  iconst_4            (Push integer constant)
000428  iconst_0            (Push integer constant)
000429  iastore             (Store into integer array)
00042A  aastore             (Store into object reference array)
00042B  dup                 (Duplicate top stack word)
00042C  iconst_2            (Push integer constant)
00042D  iconst_5            (Push integer constant)
00042E  newarray        0A  (Allocate new array) : array type = T_INT
000430  dup                 (Duplicate top stack word)
000431  iconst_0            (Push integer constant)
000432  iconst_1            (Push integer constant)
000433  iastore             (Store into integer array)
000434  dup                 (Duplicate top stack word)
000435  iconst_1            (Push integer constant)
000436  iconst_0            (Push integer constant)
000437  iastore             (Store into integer array)
000438  dup                 (Duplicate top stack word)
000439  iconst_2            (Push integer constant)
00043A  iconst_0            (Push integer constant)
00043B  iastore             (Store into integer array)
00043C  dup                 (Duplicate top stack word)
00043D  iconst_3            (Push integer constant)
00043E  iconst_0            (Push integer constant)
00043F  iastore             (Store into integer array)
000440  dup                 (Duplicate top stack word)
000441  iconst_4            (Push integer constant)
000442  iconst_0            (Push integer constant)
000443  iastore             (Store into integer array)
000444  aastore             (Store into object reference array)
000445  dup                 (Duplicate top stack word)
000446  iconst_3            (Push integer constant)
000447  iconst_5            (Push integer constant)
000448  newarray        0A  (Allocate new array) : array type = T_INT
00044A  dup                 (Duplicate top stack word)
00044B  iconst_0            (Push integer constant)
00044C  iconst_1            (Push integer constant)
00044D  iastore             (Store into integer array)
00044E  dup                 (Duplicate top stack word)
00044F  iconst_1            (Push integer constant)
000450  iconst_1            (Push integer constant)
000451  iastore             (Store into integer array)
000452  dup                 (Duplicate top stack word)
000453  iconst_2            (Push integer constant)
000454  iconst_1            (Push integer constant)
000455  iastore             (Store into integer array)
000456  dup                 (Duplicate top stack word)
```

〈リスト8-11〉sample.classのrejava出力㉖

```
000457  iconst_3              (Push integer constant)
000458  iconst_1              (Push integer constant)
000459  iastore               (Store into integer array)
00045A  dup                   (Duplicate top stack word)
00045B  iconst_4              (Push integer constant)
00045C  iconst_0              (Push integer constant)
00045D  iastore               (Store into integer array)
00045E  aastore               (Store into object reference array)
00045F  dup                   (Duplicate top stack word)
000460  iconst_4              (Push integer constant)
000461  iconst_5              (Push integer constant)
000462  newarray        0A    (Allocate new array) : array type = T_INT
000464  dup                   (Duplicate top stack word)
000465  iconst_0              (Push integer constant)
000466  iconst_0              (Push integer constant)
000467  iastore               (Store into integer array)
000468  dup                   (Duplicate top stack word)
000469  iconst_1              (Push integer constant)
00046A  iconst_0              (Push integer constant)
00046B  iastore               (Store into integer array)
00046C  dup                   (Duplicate top stack word)
00046D  iconst_2              (Push integer constant)
00046E  iconst_0              (Push integer constant)
00046F  iastore               (Store into integer array)
000470  dup                   (Duplicate top stack word)
000471  iconst_3              (Push integer constant)
000472  iconst_0              (Push integer constant)
000473  iastore               (Store into integer array)
000474  dup                   (Duplicate top stack word)
000475  iconst_4              (Push integer constant)
000476  iconst_1              (Push integer constant)
000477  iastore               (Store into integer array)
000478  aastore               (Store into object reference array)
000479  dup                   (Duplicate top stack word)
00047A  iconst_5              (Push integer constant)
00047B  iconst_5              (Push integer constant)
00047C  newarray        0A    (Allocate new array) : array type = T_INT
00047E  dup                   (Duplicate top stack word)
00047F  iconst_0              (Push integer constant)
000480  iconst_1              (Push integer constant)
000481  iastore               (Store into integer array)
000482  dup                   (Duplicate top stack word)
000483  iconst_1              (Push integer constant)
000484  iconst_0              (Push integer constant)
000485  iastore               (Store into integer array)
000486  dup                   (Duplicate top stack word)
000487  iconst_2              (Push integer constant)
000488  iconst_0              (Push integer constant)
000489  iastore               (Store into integer array)
00048A  dup                    (Duplicate top stack word)
00048B  iconst_3              (Push integer constant)
00048C  iconst_0              (Push integer constant)
00048D  iastore               (Store into integer array)
00048E  dup                   (Duplicate top stack word)
00048F  iconst_4              (Push integer constant)
000490  iconst_1              (Push integer constant)
000491  iastore               (Store into integer array)
000492  aastore               (Store into object reference array)
000493  dup                   (Duplicate top stack word)
000494  bipush  06            (Push one-byte signed integer)
000496  iconst_5              (Push integer constant)
000497  newarray        0A    (Allocate new array) : array type = T_INT
000499  dup                   (Duplicate top stack word)
00049A  iconst_0              (Push integer constant)
00049B  iconst_0              (Push integer constant)
00049C  iastore               (Store into integer array)
00049D  dup                   (Duplicate top stack word)
00049E  iconst_1              (Push integer constant)
00049F  iconst_1              (Push integer constant)
0004A0  iastore               (Store into integer array)
0004A1  dup                   (Duplicate top stack word)
0004A2  iconst_2              (Push integer constant)
0004A3  iconst_1              (Push integer constant)
0004A4  iastore               (Store into integer array)
0004A5  dup                   (Duplicate top stack word)
0004A6  iconst_3              (Push integer constant)
0004A7  iconst_1              (Push integer constant)
0004A8  iastore               (Store into integer array)
0004A9  dup                   (Duplicate top stack word)
0004AA  iconst_4              (Push integer constant)
0004AB  iconst_0              (Push integer constant)
0004AC  iastore               (Store into integer array)
0004AD  aastore               (Store into object reference array)
0004AE  aastore               (Store into object reference array)
```

〈リスト8-11〉sample.classのrejava出力㉗

```
0004AF  dup                     (Duplicate top stack word)
0004B0  bipush  06              (Push one-byte signed integer)
0004B2  bipush  07              (Push one-byte signed integer)
0004B4  anewarray       0004    (Allocate new array of references to objects)
0004B7  dup                     (Duplicate top stack word)
0004B8  iconst_0                (Push integer constant)
0004B9  iconst_5                (Push integer constant)
0004BA  newarray        0A      (Allocate new array) : array type = T_INT
0004BC  dup                     (Duplicate top stack word)
0004BD  iconst_0                (Push integer constant)
0004BE  iconst_0                (Push integer constant)
0004BF  iastore                 (Store into integer array)
0004C0  dup                     (Duplicate top stack word)
0004C1  iconst_1                (Push integer constant)
0004C2  iconst_1                (Push integer constant)
0004C3  iastore                 (Store into integer array)
0004C4  dup                     (Duplicate top stack word)
0004C5  iconst_2                (Push integer constant)
0004C6  iconst_1                (Push integer constant)
0004C7  iastore                 (Store into integer array)
0004C8  dup                     (Duplicate top stack word)
0004C9  iconst_3                (Push integer constant)
0004CA  iconst_1                (Push integer constant)
0004CB  iastore                 (Store into integer array)
0004CC  dup                     (Duplicate top stack word)
0004CD  iconst_4                (Push integer constant)
0004CE  iconst_0                (Push integer constant)
0004CF  iastore                 (Store into integer array)
0004D0  aastore                 (Store into object reference array)
0004D1  dup                     (Duplicate top stack word)
0004D2  iconst_1                (Push integer constant)
0004D3  iconst_5                (Push integer constant)
0004D4  newarray        0A      (Allocate new array) : array type = T_INT
0004D6  dup                     (Duplicate top stack word)
0004D7  iconst_0                (Push integer constant)
0004D8  iconst_1                (Push integer constant)
0004D9  iastore                 (Store into integer array)
0004DA  dup                     (Duplicate top stack word)
0004DB  iconst_1                (Push integer constant)
0004DC  iconst_0                (Push integer constant)
0004DD  iastore                 (Store into integer array)
0004DE  dup                     (Duplicate top stack word)
0004DF  iconst_2                (Push integer constant)
0004E0  iconst_0                (Push integer constant)
0004E1  iastore                 (Store into integer array)
0004E2  dup                     (Duplicate top stack word)
0004E3  iconst_3                (Push integer constant)
0004E4  iconst_0                (Push integer constant)
0004E5  iastore                 (Store into integer array)
0004E6  dup                     (Duplicate top stack word)
0004E7  iconst_4                (Push integer constant)
0004E8  iconst_1                (Push integer constant)
0004E9  iastore                 (Store into integer array)
0004EA  aastore                 (Store into object reference array)
0004EB  dup                     (Duplicate top stack word)
0004EC  iconst_2                (Push integer constant)
0004ED  iconst_5                (Push integer constant)
0004EE  newarray        0A      (Allocate new array) : array type = T_INT
0004F0  dup                     (Duplicate top stack word)
0004F1  iconst_0                (Push integer constant)
0004F2  iconst_1                (Push integer constant)
0004F3  iastore                 (Store into integer array)
0004F4  dup                     (Duplicate top stack word)
0004F5  iconst_1                (Push integer constant)
0004F6  iconst_0                (Push integer constant)
0004F7  iastore                 (Store into integer array)
0004F8  dup                     (Duplicate top stack word)
0004F9  iconst_2                (Push integer constant)
0004FA  iconst_0                (Push integer constant)
0004FB  iastore                 (Store into integer array)
0004FC  dup                     (Duplicate top stack word)
0004FD  iconst_3                (Push integer constant)
0004FE  iconst_0                (Push integer constant)
0004FF  iastore                 (Store into integer array)
000500  dup                     (Duplicate top stack word)
000501  iconst_4                (Push integer constant)
000502  iconst_0                (Push integer constant)
000503  iastore                 (Store into integer array)
000504  aastore                 (Store into object reference array)
000505  dup                     (Duplicate top stack word)
000506  iconst_3                (Push integer constant)
000507  iconst_5                (Push integer constant)
000508  newarray        0A      (Allocate new array) : array type = T_INT
00050A  dup                     (Duplicate top stack word)
```

〈リスト8-11〉 sample.classのrejava出力㉘

```
00050B  iconst_0                (Push integer constant)
00050C  iconst_1                (Push integer constant)
00050D  iastore                 (Store into integer array)
00050E  dup                     (Duplicate top stack word)
00050F  iconst_1                (Push integer constant)
000510  iconst_1                (Push integer constant)
000511  iastore                 (Store into integer array)
000512  dup                     (Duplicate top stack word)
000513  iconst_2                (Push integer constant)
000514  iconst_1                (Push integer constant)
000515  iastore                 (Store into integer array)
000516  dup                     (Duplicate top stack word)
000517  iconst_3                (Push integer constant)
000518  iconst_1                (Push integer constant)
000519  iastore                 (Store into integer array)
00051A  dup                     (Duplicate top stack word)
00051B  iconst_4                (Push integer constant)
00051C  iconst_0                (Push integer constant)
00051D  iastore                 (Store into integer array)
00051E  aastore                 (Store into object reference array)
00051F  dup                     (Duplicate top stack word)
000520  iconst_4                (Push integer constant)
000521  iconst_5                (Push integer constant)
000522  newarray        0A      (Allocate new array) : array type = T_INT
000524  dup                     (Duplicate top stack word)
000525  iconst_0                (Push integer constant)
000526  iconst_1                (Push integer constant)
000527  iastore                 (Store into integer array)
000528  dup                     (Duplicate top stack word)
000529  iconst_1                (Push integer constant)
00052A  iconst_0                (Push integer constant)
00052B  iastore                 (Store into integer array)
00052C  dup                     (Duplicate top stack word)
00052D  iconst_2                (Push integer constant)
00052E  iconst_0                (Push integer constant)
00052F  iastore                 (Store into integer array)
000530  dup                     (Duplicate top stack word)
000531  iconst_3                (Push integer constant)
000532  iconst_0                (Push integer constant)
000533  iastore                 (Store into integer array)
000534  dup                     (Duplicate top stack word)
000535  iconst_4                (Push integer constant)
000536  iconst_1                (Push integer constant)
000537  iastore                 (Store into integer array)
000538  aastore                 (Store into object reference array)
000539  dup                     (Duplicate top stack word)
00053A  iconst_5                (Push integer constant)
00053B  iconst_5                (Push integer constant)
00053C  newarray        0A      (Allocate new array) : array type = T_INT
00053E  dup                     (Duplicate top stack word)
00053F  iconst_0                (Push integer constant)
000540  iconst_1                (Push integer constant)
000541  iastore                 (Store into integer array)
000542  dup                     (Duplicate top stack word)
000543  iconst_1                (Push integer constant)
000544  iconst_0                (Push integer constant)
000545  iastore                 (Store into integer array)
000546  dup                     (Duplicate top stack word)
000547  iconst_2                (Push integer constant)
000548  iconst_0                (Push integer constant)
000549  iastore                 (Store into integer array)
00054A  dup                     (Duplicate top stack word)
00054B  iconst_3                (Push integer constant)
00054C  iconst_0                (Push integer constant)
00054D  iastore                 (Store into integer array)
00054E  dup                     (Duplicate top stack word)
00054F  iconst_4                (Push integer constant)
000550  iconst_1                (Push integer constant)
000551  iastore                 (Store into integer array)
000552  aastore                 (Store into object reference array)
000553  dup                     (Duplicate top stack word)
000554  bipush  06              (Push one-byte signed integer)
000556  iconst_5                (Push integer constant)
000557  newarray        0A      (Allocate new array) : array type = T_INT
000559  dup                     (Duplicate top stack word)
00055A  iconst_0                (Push integer constant)
00055B  iconst_0                (Push integer constant)
00055C  iastore                 (Store into integer array)
00055D  dup                     (Duplicate top stack word)
00055E  iconst_1                (Push integer constant)
00055F  iconst_1                (Push integer constant)
000560  iastore                 (Store into integer array)
000561  dup                     (Duplicate top stack word)
000562  iconst_2                (Push integer constant)
```

〈リスト8-11〉sample.classのrejava出力㉙

```
000563  iconst_1                (Push integer constant)
000564  iastore                 (Store into integer array)
000565  dup                     (Duplicate top stack word)
000566  iconst_3                (Push integer constant)
000567  iconst_1                (Push integer constant)
000568  iastore                 (Store into integer array)
000569  dup                     (Duplicate top stack word)
00056A  iconst_4                (Push integer constant)
00056B  iconst_0                (Push integer constant)
00056C  iastore                 (Store into integer array)
00056D  aastore                 (Store into object reference array)
00056E  aastore                 (Store into object reference array)
00056F  dup                     (Duplicate top stack word)
000570  bipush  07              (Push one-byte signed integer)
000572  bipush  07              (Push one-byte signed integer)
000574  anewarray       0004    (Allocate new array of references to objects)
000577  dup                     (Duplicate top stack word)
000578  iconst_0                (Push integer constant)
000579  iconst_5                (Push integer constant)
00057A  newarray        0A      (Allocate new array) : array type = T_INT
00057C  dup                     (Duplicate top stack word)
00057D  iconst_0                (Push integer constant)
00057E  iconst_1                (Push integer constant)
00057F  iastore                 (Store into integer array)
000580  dup                     (Duplicate top stack word)
000581  iconst_1                (Push integer constant)
000582  iconst_1                (Push integer constant)
000583  iastore                 (Store into integer array)
000584  dup                     (Duplicate top stack word)
000585  iconst_2                (Push integer constant)
000586  iconst_1                (Push integer constant)
000587  iastore                 (Store into integer array)
000588  dup                     (Duplicate top stack word)
000589  iconst_3                (Push integer constant)
00058A  iconst_1                (Push integer constant)
00058B  iastore                 (Store into integer array)
00058C  dup                     (Duplicate top stack word)
00058D  iconst_4                (Push integer constant)
00058E  iconst_1                (Push integer constant)
00058F  iastore                 (Store into integer array)
000590  aastore                 (Store into object reference array)
000591  dup                     (Duplicate top stack word)
000592  iconst_1                (Push integer constant)
000593  iconst_5                (Push integer constant)
000594  newarray        0A      (Allocate new array) : array type = T_INT
000596  dup                     (Duplicate top stack word)
000597  iconst_0                (Push integer constant)
000598  iconst_1                (Push integer constant)
000599  iastore                 (Store into integer array)
00059A  dup                     (Duplicate top stack word)
00059B  iconst_1                (Push integer constant)
00059C  iconst_0                (Push integer constant)
00059D  iastore                 (Store into integer array)
00059E  dup                     (Duplicate top stack word)
00059F  iconst_2                (Push integer constant)
0005A0  iconst_0                (Push integer constant)
0005A1  iastore                 (Store into integer array)
0005A2  dup                     (Duplicate top stack word)
0005A3  iconst_3                (Push integer constant)
0005A4  iconst_0                (Push integer constant)
0005A5  iastore                 (Store into integer array)
0005A6  dup                     (Duplicate top stack word)
0005A7  iconst_4                (Push integer constant)
0005A8  iconst_1                (Push integer constant)
0005A9  iastore                 (Store into integer array)
0005AA  aastore                 (Store into object reference array)
0005AB  dup                     (Duplicate top stack word)
0005AC  iconst_2                (Push integer constant)
0005AD  iconst_5                (Push integer constant)
0005AE  newarray        0A      (Allocate new array) : array type = T_INT
0005B0  dup                     (Duplicate top stack word)
0005B1  iconst_0                (Push integer constant)
0005B2  iconst_0                (Push integer constant)
0005B3  iastore                 (Store into integer array)
0005B4  dup                     (Duplicate top stack word)
0005B5  iconst_1                (Push integer constant)
0005B6  iconst_0                (Push integer constant)
0005B7  iastore                 (Store into integer array)
0005B8  dup                     (Duplicate top stack word)
0005B9  iconst_2                (Push integer constant)
0005BA  iconst_0                (Push integer constant)
0005BB  iastore                 (Store into integer array)
0005BC  dup                     (Duplicate top stack word)
0005BD  iconst_3                (Push integer constant)
```

〈リスト8-11〉 sample.classのrejava出力㉚

```
0005BE  iconst_0                (Push integer constant)
0005BF  iastore                 (Store into integer array)
0005C0  dup                     (Duplicate top stack word)
0005C1  iconst_4                (Push integer constant)
0005C2  iconst_1                (Push integer constant)
0005C3  iastore                 (Store into integer array)
0005C4  aastore                 (Store into object reference array)
0005C5  dup                     (Duplicate top stack word)
0005C6  iconst_3                (Push integer constant)
0005C7  iconst_5                (Push integer constant)
0005C8  newarray        0A      (Allocate new array) : array type = T_INT
0005CA  dup                     (Duplicate top stack word)
0005CB  iconst_0                (Push integer constant)
0005CC  iconst_0                (Push integer constant)
0005CD  iastore                 (Store into integer array)
0005CE  dup                     (Duplicate top stack word)
0005CF  iconst_1                (Push integer constant)
0005D0  iconst_0                (Push integer constant)
0005D1  iastore                 (Store into integer array)
0005D2  dup                     (Duplicate top stack word)
0005D3  iconst_2                (Push integer constant)
0005D4  iconst_0                (Push integer constant)
0005D5  iastore                 (Store into integer array)
0005D6  dup                     (Duplicate top stack word)
0005D7  iconst_3                (Push integer constant)
0005D8  iconst_1                (Push integer constant)
0005D9  iastore                 (Store into integer array)
0005DA  dup                     (Duplicate top stack word)
0005DB  iconst_4                (Push integer constant)
0005DC  iconst_0                (Push integer constant)
0005DD  iastore                 (Store into integer array)
0005DE  aastore                 (Store into object reference array)
0005DF  dup                     (Duplicate top stack word)
0005E0  iconst_4                (Push integer constant)
0005E1  iconst_5                (Push integer constant)
0005E2  newarray        0A      (Allocate new array) : array type = T_INT
0005E4  dup                     (Duplicate top stack word)
0005E5  iconst_0                (Push integer constant)
0005E6  iconst_0                (Push integer constant)
0005E7  iastore                 (Store into integer array)
0005E8  dup                     (Duplicate top stack word)
0005E9  iconst_1                (Push integer constant)
0005EA  iconst_0                (Push integer constant)
0005EB  iastore                 (Store into integer array)
0005EC  dup                     (Duplicate top stack word)
0005ED  iconst_2                (Push integer constant)
0005EE  iconst_1                (Push integer constant)
0005EF  iastore                 (Store into integer array)
0005F0  dup                     (Duplicate top stack word)
0005F1  iconst_3                (Push integer constant)
0005F2  iconst_0                (Push integer constant)
0005F3  iastore                 (Store into integer array)
0005F4  dup                     (Duplicate top stack word)
0005F5  iconst_4                (Push integer constant)
0005F6  iconst_0                (Push integer constant)
0005F7  iastore                 (Store into integer array)
0005F8  aastore                 (Store into object reference array)
0005F9  dup                     (Duplicate top stack word)
0005FA  iconst_5                (Push integer constant)
0005FB  iconst_5                (Push integer constant)
0005FC  newarray        0A      (Allocate new array) : array type = T_INT
0005FE  dup                     (Duplicate top stack word)
0005FF  iconst_0                (Push integer constant)
000600  iconst_0                (Push integer constant)
000601  iastore                 (Store into integer array)
000602  dup                     (Duplicate top stack word)
000603  iconst_1                (Push integer constant)
000604  iconst_1                (Push integer constant)
000605  iastore                 (Store into integer array)
000606  dup                     (Duplicate top stack word)
000607  iconst_2                (Push integer constant)
000608  iconst_0                (Push integer constant)
000609  iastore                 (Store into integer array)
00060A  dup                     (Duplicate top stack word)
00060B  iconst_3                (Push integer constant)
00060C  iconst_0                (Push integer constant)
00060D  iastore                 (Store into integer array)
00060E  dup                     (Duplicate top stack word)
00060F  iconst_4                (Push integer constant)
000610  iconst_0                (Push integer constant)
000611  iastore                 (Store into integer array)
000612  aastore                 (Store into object reference array)
000613  dup                     (Duplicate top stack word)
000614  bipush  06              (Push one-byte signed integer)
```

〈リスト8-11〉sample.classのrejava出力㉛

```
000616  iconst_5                (Push integer constant)
000617  newarray        0A      (Allocate new array) : array type = T_INT
000619  dup                     (Duplicate top stack word)
00061A  iconst_0                (Push integer constant)
00061B  iconst_1                (Push integer constant)
00061C  iastore                 (Store into integer array)
00061D  dup                     (Duplicate top stack word)
00061E  iconst_1                (Push integer constant)
00061F  iconst_0                (Push integer constant)
000620  iastore                 (Store into integer array)
000621  dup                      (Duplicate top stack word)
000622  iconst_2                (Push integer constant)
000623  iconst_0                (Push integer constant)
000624  iastore                 (Store into integer array)
000625  dup                     (Duplicate top stack word)
000626  iconst_3                (Push integer constant)
000627  iconst_0                (Push integer constant)
000628  iastore                 (Store into integer array)
000629  dup                     (Duplicate top stack word)
00062A  iconst_4                (Push integer constant)
00062B  iconst_0                (Push integer constant)
00062C  iastore                 (Store into integer array)
00062D  aastore                 (Store into object reference array)
00062E  aastore                 (Store into object reference array)
00062F  dup                     (Duplicate top stack word)
000630  bipush  08              (Push one-byte signed integer)
000632  bipush  07              (Push one-byte signed integer)
000634  anewarray       0004    (Allocate new array of references to objects)
000637  dup                     (Duplicate top stack word)
000638  iconst_0                (Push integer constant)
000639  iconst_5                (Push integer constant)
00063A  newarray        0A      (Allocate new array) : array type = T_INT
00063C  dup                     (Duplicate top stack word)
00063D  iconst_0                (Push integer constant)
00063E  iconst_0                (Push integer constant)
00063F  iastore                 (Store into integer array)
000640  dup                     (Duplicate top stack word)
000641  iconst_1                (Push integer constant)
000642  iconst_1                (Push integer constant)
000643  iastore                 (Store into integer array)
000644  dup                     (Duplicate top stack word)
000645  iconst_2                (Push integer constant)
000646  iconst_1                (Push integer constant)
000647  iastore                 (Store into integer array)
000648  dup                     (Duplicate top stack word)
000649  iconst_3                (Push integer constant)
00064A  iconst_1                (Push integer constant)
00064B  iastore                 (Store into integer array)
00064C  dup                     (Duplicate top stack word)
00064D  iconst_4                (Push integer constant)
00064E  iconst_0                (Push integer constant)
00064F  iastore                 (Store into integer array)
000650  aastore                 (Store into object reference array)
000651  dup                     (Duplicate top stack word)
000652  iconst_1                (Push integer constant)
000653  iconst_5                (Push integer constant)
000654  newarray        0A      (Allocate new array) : array type = T_INT
000656  dup                     (Duplicate top stack word)
000657  iconst_0                (Push integer constant)
000658  iconst_1                (Push integer constant)
000659  iastore                 (Store into integer array)
00065A  dup                     (Duplicate top stack word)
00065B  iconst_1                (Push integer constant)
00065C  iconst_0                (Push integer constant)
00065D  iastore                 (Store into integer array)
00065E  dup                     (Duplicate top stack word)
00065F  iconst_2                (Push integer constant)
000660  iconst_0                (Push integer constant)
000661  iastore                 (Store into integer array)
000662  dup                     (Duplicate top stack word)
000663  iconst_3                (Push integer constant)
000664  iconst_0                (Push integer constant)
000665  iastore                 (Store into integer array)
000666  dup                     (Duplicate top stack word)
000667  iconst_4                (Push integer constant)
000668  iconst_1                (Push integer constant)
000669  iastore                 (Store into integer array)
00066A  aastore                 (Store into object reference array)
00066B  dup                     (Duplicate top stack word)
00066C  iconst_2                (Push integer constant)
00066D  iconst_5                (Push integer constant)
00066E  newarray        0A      (Allocate new array) : array type = T_INT
000670  dup                     (Duplicate top stack word)
000671  iconst_0                (Push integer constant)
```

〈リスト8-11〉 sample.classのrejava出力㉜

```
000672  iconst_1              (Push integer constant)
000673  iastore               (Store into integer array)
000674  dup                   (Duplicate top stack word)
000675  iconst_1              (Push integer constant)
000676  iconst_0              (Push integer constant)
000677  iastore               (Store into integer array)
000678  dup                   (Duplicate top stack word)
000679  iconst_2              (Push integer constant)
00067A  iconst_0              (Push integer constant)
00067B  iastore               (Store into integer array)
00067C  dup                   (Duplicate top stack word)
00067D  iconst_3              (Push integer constant)
00067E  iconst_0              (Push integer constant)
00067F  iastore               (Store into integer array)
000680  dup                   (Duplicate top stack word)
000681  iconst_4              (Push integer constant)
000682  iconst_1              (Push integer constant)
000683  iastore               (Store into integer array)
000684  aastore               (Store into object reference array)
000685  dup                   (Duplicate top stack word)
000686  iconst_3              (Push integer constant)
000687  iconst_5              (Push integer constant)
000688  newarray        0A    (Allocate new array) : array type = T_INT
00068A  dup                   (Duplicate top stack word)
00068B  iconst_0              (Push integer constant)
00068C  iconst_0              (Push integer constant)
00068D  iastore               (Store into integer array)
00068E  dup                   (Duplicate top stack word)
00068F  iconst_1              (Push integer constant)
000690  iconst_1              (Push integer constant)
000691  iastore               (Store into integer array)
000692  dup                   (Duplicate top stack word)
000693  iconst_2              (Push integer constant)
000694  iconst_1              (Push integer constant)
000695  iastore               (Store into integer array)
000696  dup                   (Duplicate top stack word)
000697  iconst_3              (Push integer constant)
000698  iconst_1              (Push integer constant)
000699  iastore               (Store into integer array)
00069A  dup                   (Duplicate top stack word)
00069B  iconst_4              (Push integer constant)
00069C  iconst_0              (Push integer constant)
00069D  iastore               (Store into integer array)
00069E  aastore               (Store into object reference array)
00069F  dup                   (Duplicate top stack word)
0006A0  iconst_4              (Push integer constant)
0006A1  iconst_5              (Push integer constant)
0006A2  newarray        0A    (Allocate new array) : array type = T_INT
0006A4  dup                   (Duplicate top stack word)
0006A5  iconst_0              (Push integer constant)
0006A6  iconst_1              (Push integer constant)
0006A7  iastore               (Store into integer array)
0006A8  dup                   (Duplicate top stack word)
0006A9  iconst_1              (Push integer constant)
0006AA  iconst_0              (Push integer constant)
0006AB  iastore               (Store into integer array)
0006AC  dup                   (Duplicate top stack word)
0006AD  iconst_2              (Push integer constant)
0006AE  iconst_0              (Push integer constant)
0006AF  iastore               (Store into integer array)
0006B0  dup                   (Duplicate top stack word)
0006B1  iconst_3              (Push integer constant)
0006B2  iconst_0              (Push integer constant)
0006B3  iastore               (Store into integer array)
0006B4  dup                   (Duplicate top stack word)
0006B5  iconst_4              (Push integer constant)
0006B6  iconst_1              (Push integer constant)
0006B7  iastore               (Store into integer array)
0006B8  aastore               (Store into object reference array)
0006B9  dup                   (Duplicate top stack word)
0006BA  iconst_5              (Push integer constant)
0006BB  iconst_5              (Push integer constant)
0006BC  newarray        0A    (Allocate new array) : array type = T_INT
0006BE  dup                   (Duplicate top stack word)
0006BF  iconst_0              (Push integer constant)
0006C0  iconst_1              (Push integer constant)
0006C1  iastore               (Store into integer array)
0006C2  dup                   (Duplicate top stack word)
0006C3  iconst_1              (Push integer constant)
0006C4  iconst_0              (Push integer constant)
0006C5  iastore               (Store into integer array)
0006C6  dup                   (Duplicate top stack word)
0006C7  iconst_2              (Push integer constant)
0006C8  iconst_0              (Push integer constant)
```

〈リスト8-11〉sample.classのrejava出力㉝

```
0006C9  iastore                 (Store into integer array)
0006CA  dup                     (Duplicate top stack word)
0006CB  iconst_3                (Push integer constant)
0006CC  iconst_0                (Push integer constant)
0006CD  iastore                 (Store into integer array)
0006CE  dup                     (Duplicate top stack word)
0006CF  iconst_4                (Push integer constant)
0006D0  iconst_1                (Push integer constant)
0006D1  iastore                 (Store into integer array)
0006D2  aastore                 (Store into object reference array)
0006D3  dup                     (Duplicate top stack word)
0006D4  bipush  06              (Push one-byte signed integer)
0006D6  iconst_5                (Push integer constant)
0006D7  newarray        0A      (Allocate new array) : array type = T_INT
0006D9  dup                     (Duplicate top stack word)
0006DA  iconst_0                (Push integer constant)
0006DB  iconst_0                (Push integer constant)
0006DC  iastore                 (Store into integer array)
0006DD  dup                     (Duplicate top stack word)
0006DE  iconst_1                (Push integer constant)
0006DF  iconst_1                (Push integer constant)
0006E0  iastore                 (Store into integer array)
0006E1  dup                     (Duplicate top stack word)
0006E2  iconst_2                (Push integer constant)
0006E3  iconst_1                (Push integer constant)
0006E4  iastore                 (Store into integer array)
0006E5  dup                     (Duplicate top stack word)
0006E6  iconst_3                (Push integer constant)
0006E7  iconst_1                (Push integer constant)
0006E8  iastore                 (Store into integer array)
0006E9  dup                     (Duplicate top stack word)
0006EA  iconst_4                (Push integer constant)
0006EB  iconst_0                (Push integer constant)
0006EC  iastore                 (Store into integer array)
0006ED  aastore                 (Store into object reference array)
0006EE  aastore                 (Store into object reference array)
0006EF  dup                     (Duplicate top stack word)
0006F0  bipush  09              (Push one-byte signed integer)
0006F2  bipush  07              (Push one-byte signed integer)
0006F4  anewarray       0004    (Allocate new array of references to objects)
0006F7  dup                     (Duplicate top stack word)
0006F8  iconst_0                (Push integer constant)
0006F9  iconst_5                (Push integer constant)
0006FA  newarray        0A      (Allocate new array) : array type = T_INT
0006FC  dup                     (Duplicate top stack word)
0006FD  iconst_0                (Push integer constant)
0006FE  iconst_0                (Push integer constant)
0006FF  iastore                 (Store into integer array)
000700  dup                     (Duplicate top stack word)
000701  iconst_1                (Push integer constant)
000702  iconst_1                (Push integer constant)
000703  iastore                 (Store into integer array)
000704  dup                     (Duplicate top stack word)
000705  iconst_2                (Push integer constant)
000706  iconst_1                (Push integer constant)
000707  iastore                 (Store into integer array)
000708  dup                     (Duplicate top stack word)
000709  iconst_3                (Push integer constant)
00070A  iconst_1                (Push integer constant)
00070B  iastore                 (Store into integer array)
00070C  dup                     (Duplicate top stack word)
00070D  iconst_4                (Push integer constant)
00070E  iconst_0                (Push integer constant)
00070F  iastore                 (Store into integer array)
000710  aastore                 (Store into object reference array)
000711  dup                     (Duplicate top stack word)
000712  iconst_1                (Push integer constant)
000713  iconst_5                (Push integer constant)
000714  newarray        0A      (Allocate new array) : array type = T_INT
000716  dup                     (Duplicate top stack word)
000717  iconst_0                (Push integer constant)
000718  iconst_1                (Push integer constant)
000719  iastore                 (Store into integer array)
00071A  dup                     (Duplicate top stack word)
00071B  iconst_1                (Push integer constant)
00071C  iconst_0                (Push integer constant)
00071D  iastore                 (Store into integer array)
00071E  dup                     (Duplicate top stack word)
00071F  iconst_2                (Push integer constant)
000720  iconst_0                (Push integer constant)
000721  iastore                 (Store into integer array)
000722  dup                     (Duplicate top stack word)
000723  iconst_3                (Push integer constant)
000724  iconst_0                (Push integer constant)
```

〈リスト8-11〉 sample.classのrejava出力㉞

```
000725  iastore                 (Store into integer array)
000726  dup                     (Duplicate top stack word)
000727  iconst_4                (Push integer constant)
000728  iconst_1                (Push integer constant)
000729  iastore                 (Store into integer array)
00072A  aastore                 (Store into object reference array)
00072B  dup                     (Duplicate top stack word)
00072C  iconst_2                (Push integer constant)
00072D  iconst_5                (Push integer constant)
00072E  newarray        0A      (Allocate new array) : array type = T_INT
000730  dup                     (Duplicate top stack word)
000731  iconst_0                (Push integer constant)
000732  iconst_1                (Push integer constant)
000733  iastore                 (Store into integer array)
000734  dup                     (Duplicate top stack word)
000735  iconst_1                (Push integer constant)
000736  iconst_0                (Push integer constant)
000737  iastore                 (Store into integer array)
000738  dup                     (Duplicate top stack word)
000739  iconst_2                (Push integer constant)
00073A  iconst_0                (Push integer constant)
00073B  iastore                 (Store into integer array)
00073C  dup                     (Duplicate top stack word)
00073D  iconst_3                (Push integer constant)
00073E  iconst_0                (Push integer constant)
00073F  iastore                 (Store into integer array)
000740  dup                     (Duplicate top stack word)
000741  iconst_4                (Push integer constant)
000742  iconst_1                (Push integer constant)
000743  iastore                 (Store into integer array)
000744  aastore                 (Store into object reference array)
000745  dup                     (Duplicate top stack word)
000746  iconst_3                (Push integer constant)
000747  iconst_5                (Push integer constant)
000748  newarray        0A      (Allocate new array) : array type = T_INT
00074A  dup                     (Duplicate top stack word)
00074B  iconst_0                (Push integer constant)
00074C  iconst_0                (Push integer constant)
00074D  iastore                 (Store into integer array)
00074E  dup                     (Duplicate top stack word)
00074F  iconst_1                (Push integer constant)
000750  iconst_1                (Push integer constant)
000751  iastore                 (Store into integer array)
000752  dup                     (Duplicate top stack word)
000753  iconst_2                (Push integer constant)
000754  iconst_1                (Push integer constant)
000755  iastore                 (Store into integer array)
000756  dup                     (Duplicate top stack word)
000757  iconst_3                (Push integer constant)
000758  iconst_1                (Push integer constant)
000759  iastore                 (Store into integer array)
00075A  dup                     (Duplicate top stack word)
00075B  iconst_4                (Push integer constant)
00075C  iconst_1                (Push integer constant)
00075D  iastore                 (Store into integer array)
00075E  aastore                 (Store into object reference array)
00075F  dup                     (Duplicate top stack word)
000760  iconst_4                (Push integer constant)
000761  iconst_5                (Push integer constant)
000762  newarray        0A      (Allocate new array) : array type = T_INT
000764  dup                     (Duplicate top stack word)
000765  iconst_0                (Push integer constant)
000766  iconst_0                (Push integer constant)
000767  iastore                 (Store into integer array)
000768  dup                     (Duplicate top stack word)
000769  iconst_1                (Push integer constant)
00076A  iconst_0                (Push integer constant)
00076B  iastore                 (Store into integer array)
00076C  dup                     (Duplicate top stack word)
00076D  iconst_2                (Push integer constant)
00076E  iconst_0                (Push integer constant)
00076F  iastore                 (Store into integer array)
000770  dup                     (Duplicate top stack word)
000771  iconst_3                (Push integer constant)
000772  iconst_0                (Push integer constant)
000773  iastore                 (Store into integer array)
000774  dup                     (Duplicate top stack word)
000775  iconst_4                (Push integer constant)
000776  iconst_1                (Push integer constant)
000777  iastore                 (Store into integer array)
000778  aastore                 (Store into object reference array)
000779  dup                     (Duplicate top stack word)
00077A  iconst_5                (Push integer constant)
00077B  iconst_5                (Push integer constant)
```

〈リスト8-11〉sample.classのrejava出力㉟

```
                        00077C  newarray        0A      (Allocate new array) : array type = T_INT
                        00077E  dup                     (Duplicate top stack word)
                        00077F  iconst_0                (Push integer constant)
                        000780  iconst_1                (Push integer constant)
                        000781  iastore                 (Store into integer array)
                        000782  dup                     (Duplicate top stack word)
                        000783  iconst_1                (Push integer constant)
                        000784  iconst_0                (Push integer constant)
                        000785  iastore                 (Store into integer array)
                        000786  dup                     (Duplicate top stack word)
                        000787  iconst_2                (Push integer constant)
                        000788  iconst_0                (Push integer constant)
                        000789  iastore                 (Store into integer array)
                        00078A  dup                     (Duplicate top stack word)
                        00078B  iconst_3                (Push integer constant)
                        00078C  iconst_0                (Push integer constant)
                        00078D  iastore                 (Store into integer array)
                        00078E  dup                     (Duplicate top stack word)
                        00078F  iconst_4                (Push integer constant)
                        000790  iconst_1                (Push integer constant)
                        000791  iastore                 (Store into integer array)
                        000792  aastore                 (Store into object reference array)
                        000793  dup                     (Duplicate top stack word)
                        000794  bipush  06              (Push one-byte signed integer)
                        000796  iconst_5                (Push integer constant)
                        000797  newarray        0A      (Allocate new array) : array type = T_INT
                        000799  dup                     (Duplicate top stack word)
                        00079A  iconst_0                (Push integer constant)
                        00079B  iconst_0                (Push integer constant)
                        00079C  iastore                 (Store into integer array)
                        00079D  dup                     (Duplicate top stack word)
                        00079E  iconst_1                (Push integer constant)
                        00079F  iconst_1                (Push integer constant)
                        0007A0  iastore                 (Store into integer array)
                        0007A1  dup                     (Duplicate top stack word)
                        0007A2  iconst_2                (Push integer constant)
                        0007A3  iconst_1                (Push integer constant)
                        0007A4  iastore                 (Store into integer array)
                        0007A5  dup                     (Duplicate top stack word)
                        0007A6  iconst_3                (Push integer constant)
                        0007A7  iconst_1                (Push integer constant)
                        0007A8  iastore                 (Store into integer array)
                        0007A9  dup                     (Duplicate top stack word)
                        0007AA  iconst_4                (Push integer constant)
                        0007AB  iconst_0                (Push integer constant)
                        0007AC  iastore                 (Store into integer array)
                        0007AD  aastore                 (Store into object reference array)
                        0007AE  aastore                 (Store into object reference array)
                        0007AF  putfield        002E    (Set field in object)
                        0007B2  return                  (Return (void) from procedure)

                exception_table_length = 0

                attributes_count = 1

                        000013C9  type = LineNumberTable , attribute_length = 102

                                00 19 00 00 00 06 00 04 00 08 00 26 00 09 00 2F 00 0B 00 37
                                00 0C 00 F3 00 0B 00 F6 00 0D 01 B2 00 0B 01 B5 00 0E 02 71
                                00 0B 02 74 00 0F 03 30 00 0B 03 33 00 10 03 EF 00 0B 03 F2
                                00 11 04 AE 00 0B 04 B2 00 12 05 6E 00 0B 05 72 00 13 06 2E
                                00 0B 06 32 00 14 06 EE 00 0B 06 F2 00 15 07 AE 00 0B 07 B2
                                00 06

        00001433  Attributes Count = 1

                00001435  attribute[0] : type = SourceFile , length = 2

                                00 AF

                Source File Name = sample.java

..... (^_^) Java analyze completely finished. (^_^)
```

第9章 JavaからAKI-80へ(後編)

Javaアプレットを制作する→Cプログラムに変換する→AKI-80プログラムを生成する

Javaバイナリ→Cソースのコンバータの製作

さて，いよいよ本章では「JavaでAKI-80プログラムを作る」というテーマに挑戦していきましょう．具体的なサンプルとして，本書で一貫して扱っている「MIDIもの」を取り上げてみることにします．ただし，Javaには本質的な制限があります．これはセキュリティ上の要請として必要なものなので，たとえば「**MIDI信号を扱うアプレット**」をJavaで制作することは不可能です．このあたりについて，まず簡単に整理しておきましょう．

Nifty-Serve「人工知能フォーラム」SysOpの村上さんが「Java/Javascriptの会議室」の記事として明確に整理してくれていますが，「Java の制限」とは，以下の4種類に起因する現象が乱立しています．

① アプレットのセキュリティ・ポリシからくる制限
② アプレットがベースとしている httpからくる制限
③ Javaの制限
④ ブラウザの制限

ここで，Java の「アプレット」とは，以下のようなプログラムのことです．

- プログラムの保存場所はリンクを張られたサーバ上である
- ブラウザで参照すると，プログラムが自動的に端末にロードされる
- プログラムのロードは httpプロトコルによって行われる
- 実行は端末のブラウザ環境上で行われる

そこで，このアプレットには**セキュリティ・ポリシ**という厳しい原則が用意されていて，これをクリアしないと実行できないようになっています．具体的には以下のような制限があります．

- プログラムが保管されていたサーバ上の資源にしかアクセスできない
- 端末の資源（ファイル等）にはアクセスできない
- 端末上のプログラムを開始することはできない
- 端末上の環境変数や特定のシステム・プロパティを参照できない
- ライブラリのロードや**native**メソッドのロードが行えない

また，アプレットがベースとしているhttpの仕様からくる制限として，「httpは情報提供サービス用のプロトコルであるため，ファイルの参照はできても更新することができない」というものもあります．そして，本書のテーマとはあまり関係しませんが，Java の制限として「日本語が表示できない」という点，さらにMIDIと密接に関連した制限として「外部機器の制御ができない」という致命的な仕様があります．また，ブラウザの制限としては，Netscape Navigatorでは，サーバに対するファイルの入出力を一切サポートしない」というものもあります．

このような状況ですから，たとえば「MIDI信号を出力するJavaアプレット」というのは，普通の方法では作ることができません．実際には「**プラグイン・モジュール**」としてMIDI処理を行う組み込みソフトを自作してこれとやりとりするという道がないわけではないのですが，本書の対象としては外れるのでここでは深入りしないことにします．

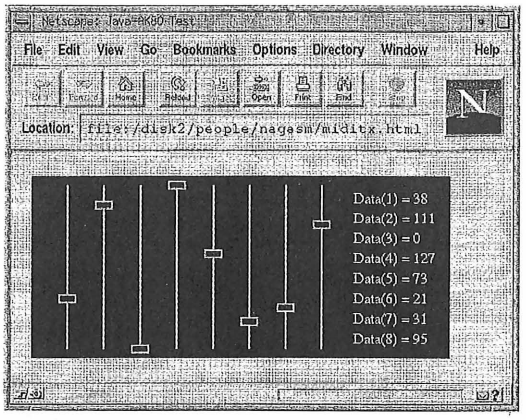
そこで，本章では，「AKI-80にさせたい仕事をJavaアプレットとして制作する」，「このJavaバイト・コードをAKI-80のためのCプログラムに変換する」，「このCプログラムから実際に仕事をするAKI-80プログラムを生成する」という3ステップによって，JavaとAKI-80を結びつけることにします．いわば，WWWブラウザの中で動くJavaアプレットが，自動変換で作成されるAKI-80システムのシミュレータとなるというイメージです．

あくまで擬似的な開発環境ではありますが，ソフトからハードまでを統合して理解するという本書の究極の目標の格好の題材といえるでしょう．

図9-1は，具体的にブラウザ上で動くサンプルのJavaアプレットの画面です．ここでは8系統のスライダーがあり，それぞれがマウスのクリック，またはドラッグによって，最小値:ゼロから，最大値:127までを

MIDI信号を扱うアプレット　Webブラウザの中で走るJavaアプレットで，そのコンピュータと外部との間でMIDI信号をやりとりするようなもの，という意味．Javaだけでは実現できない．

〈図9-1〉ブラウザ上で動作させたJavaアプレット
(8系統のスライダー)



動きます.

そして，この値が画面右側に同時に表示されています．この動作イメージは，AKI-80で実現される場合には，8系統の入力データ(8チャネルの7ビット・ディジタル入力でも，8チャネルのA-D変換入力でもかまわない)に対して，別々のMIDIチャネルを添えたバリューとしてMIDI出力するという動作に対応しています．

リスト9-1(p.161)は，このJavaアプレットのソース・プログラム[miditx.java]です．AKI-80システムにリンクするために，ちょっとだけ記述を特殊な形式にしています．すでに紹介したように，Javaアプレットには

- アクセスされた時に一度だけ表示/描画の動作を行って終了する
- Runnableタイプで刻々と処理を実行し続ける

という異なるタイプのものがありました．

ところで，AKI-80のようなマイコン・システムでは，電源が入ってリセットされれば，あとはROMのプログラムにしたがった無限ループを実行し続ける以外の選択肢はありませんから，ここではJavaアプレットとして後者のRunnableにすることになります．つまりデフォルトのメソッドとして，

- init()：最初に呼ばれた時に実行する
- run()：スレッドとしてこの部分が連続して実行される
- start()：Javaアプレットがブラウザ内で有効になると呼ばれる
- stop()：Javaアプレットがブラウザ画面から去ると呼ばれる
- paint()/repaint()：画面の(再)描画の際に呼ばれる

などが使われます．このうち，start()とstop()はブラウザとの関係なので無視することにすると，Javaプログラムの中でAKI-80システムに関係するのは，

- リセット直後に実行されるinit()
- その後の無限ループを与えるrun()

という2種類がエッセンス的な意味をもつことになります．

このような視点で**リスト9-1**を眺めてみると，AKI-80のようなマイコン・システムのプログラムを意識して記述していることがわかります．つまり，まず最初のinit()メソッドでは，「処理が8チャネルであること」と「初期設定としてsystem_initial()を呼ぶこと」という規定だけになっています．これは，system_initial()の内部として，Java側でもAKI-80側でも具体的な初期設定を行うこととして，Javaプログラムとしてはinit()が同じような記述になるようにしているというわけです．

これに続くrun()の中でも同様に，try{}の中のwhile(1)の無限ループの中には，100 msの待機であるThread.sleep(100)，入力チェックのinput_check()，さらにMIDI出力のmidi_transmit()と再描画のrepaint()が呼ばれています．Java側では実際に行わないMIDI出力などの部分は，ダミーとしてメソッド名だけのものを置いておき，AKI-80システム側ではここをデフォルトのサービス・ライブラリとして提供するという作戦なのです．

イベント検出の機構としてJavaとAKI-80とで異なるのが，Javaでは**Xウィンドウ・システム**と同じようにイベント・ハンドラをメイン・ループとは別に定義できるところです．ここはAKI-80システムでは，メイン・ループ内にチェック・ルーチンを置くという古典的な方法で実現することになります．今回の例の場合，入力としてMIDIを受信するのかディジタル・ポート入力なのかA-Dポートなのかという区別をしていませんので，ここはシステムのデフォルト環境に依存することになります．

このような具体的なサンプルが用意できたところで，次には前章で制作した[rejava]という解析ツールをさらに改良して，**簡易型のコンバータ**に成長させていくことになります．

リスト9-2(p.162)は，**リスト8-9**(p.112)のプログラムを改造したもの(rejava2.c)で，このツールによって[miditx.class]を変換した結果が**リスト9-3**(p.165)です．ここでは重要な進歩として，メソッドの呼び出し(関数やサブルーチンのコールに相当)の部分を追跡しています．**リスト9-3**の全体は長いので，**リスト9-4**(p.156)にこの実例の部分を抜き出して解説しておきましょう．

まず，**リスト9-4(1)**の部分が，Javaソースとして記述されたinit()の内容です．channelという変数は予約名として，Javaの側でもAKI-80の側でも**広域変数**

セキュリティ・ポリシ　パソコン内部のファイルを削除したり，秘密情報をインターネット経由で外部に漏洩したりするような不正な動作をするJavaプログラムは，仕様として開発が不可能であるように設計されているという方針のこと．

〈リスト9-5〉miditx.classの必要最小限の情報

```
Target Java File Name = miditx.class

Name = init
        Call [ system_initial ]

Name = run
        Call [ sleep ]
        Call [ input_check ]
        Get Field [ flag ]
        Call [ midi_transmit ]
        Call [ repaint ]

Name = start
        Define [ java/lang/Thread ]
        Call [ <init> ]
        Put Field [ flow ]
        Get Field [ flow ]
        Call [ start ]

Name = stop
        Get Field [ flow ]
        Call [ stop ]

Name = mouseDrag
        Call [ event_check ]

Name = mouseDown
        Call [ event_check ]

Name = event_check
        Get Field [ oy ]
        Get Field [ py ]
        Get Field [ py ]
        Get Field [ flag ]
        Put Field [ flag ]

Name = input_check

Name = midi_transmit

Name = paint
        Put Field [ flag ]
        Get Static [ black ]
        Call [ setColor ]
        Get Field [ oy ]
        Call [ fillRect ]
        Get Static [ yellow ]
        Call [ setColor ]
        Call [ fillRect ]
        Get Static [ green ]
        Call [ setColor ]
        Get Field [ py ]
        Call [ fillRect ]
        Get Static [ blue ]
        Call [ setColor ]
        Get Field [ py ]
        Call [ fillRect ]
        Get Field [ data ]
        Get Field [ py ]
        Get Static [ white ]
        Call [ setColor ]
        Get Field [ font ]
        Call [ setFont ]
        Define [
java/lang/StringBuffer ]
        Call [ <init> ]
        Push Item [ Data( ]
        Call [ append ]
        Call [ valueOf ]
        Call [ append ]
        Push Item [ ) =  ]
        Call [ append ]
        Get Field [ data ]
        Call [ valueOf ]
        Call [ append ]
        Call [ toString ]
        Call [ drawString ]

Name = system_initial
        Get Static [ black ]
        Call [ setBackground ]
        Define [ java/awt/Font ]
        Push Item [ TimesRoman ]
        Call [ <init> ]
        Put Field [ font ]
        Put Field [ py ]
        Put Field [ oy ]
        Put Field [ data ]
        Get Field [ py ]
        Put Field [ flag ]

Name = <init>
        Call [ <init> ]
```

〈リスト9-4〉リスト9-3の主要な部分

```
================================================================================
(1) Javaソースの該当部分

  public void init(){
        channel = 8;
        system_initial();
    }

================================================================================
(2) 解析されたバイトコード内の該当部分の情報

0005C7   method[0] : ACC_PUBLIC , Name = init , Signature = ()V , Attribute = 1
         type = Code , length = 43 , max_stack = 2 , max_locals = 1 , code_length = 11
             000000  aload_0          (Load object reference from local variable)
             000001  bipush  08       (Push one-byte signed integer)
             000003  putfield         002D    (Set field in object)
             000006  aload_0          (Load object reference from local variable)
             000007  invokevirtual    002A    (Invoke instance method)
             00000A  return           (Return (void) from procedure)

================================================================================
(3) メソッドを呼び出している部分

 000007  invokevirtual   002A    (Invoke instance method)

================================================================================
(4) [002A]で指定されたConstant_Poolの該当する情報

0000BB   constant_pool[042][2A] = <A> CONSTANT_Methodref ,
                        class_index = 10 , name_and_type_index = 67

================================================================================
(5)「name_and_type_index = 67」で指定されたConstant_Poolの該当する情報

 000138   constant_pool[067][43] = <C> CONSTANT_NameAndType ,
                         name_index = 105 , signature_index = 119

================================================================================
(6)「name_index = 105」で指定されたConstant_Poolの該当する情報

 0002CF   constant_pool[105][69] = <1> CONSTANT_Utf8 , Data = system_initial

================================================================================
(7) 得られた情報をこのように表示出力する

 000007  invokevirtual   002A    (Invoke instance method)
                                          ##### Call [ system_initial ]#####

================================================================================
```

として用意することになります．これを解析ツールに通すと，**(2)**のように変換され．この中でJavaから「system_initial()を呼ぶ」という記述に対応しているのは，**(3)** の「invokevirtual　002A」という部分になります．ここで問題となるのは，002Aという数値がどうやって具体的なメソッド名に結びつくかという部分です．

そこでふたたびSunのJava VMSPECをよく読んでみると，この002AというのはConstant_Poolに並んだリストの番号である，とあります．そこで**(4)**のように該当する部分を見ると，この002Aが，

▶ 種類はCONSTANT_Methodref（メソッドの参照）

端末の資源　ここでの「端末」とは，Javaの走るマシンそのもののことで，資源とはそのマシンのもっているハードウェアや周辺機器を言う．

〈リスト9-7〉miditx.classをリスト9-6に通した結果

```
/* Source Java File Name =
miditx.class */

main(){
        init();
        while(1){
                run();
        }
}

init(){
        system_initial();
}

run(){
        sleep();
        input_check();
        midi_transmit();
        repaint();
}

start(){
/*
        <init>();
        start();
*/
}

stop(){
/*
        stop();
*/
}

mouseDrag(){
/*
        event_check();
*/
}

mouseDown(){
/*
        event_check();
*/
}

event_check(){
/*
*/
}

input_check(){
/*
*/
}

midi_transmit(){
/*
*/
}

paint(){
/*
        setColor();
        fillRect();
        setColor();
        fillRect();
        setColor();
        fillRect();
        setColor();
        setFont();
        <init>();
        append();
        valueOf();
        append();
        append();
        valueOf();
        append();
        toString();
        drawString();
*/
}

system_initial(){
/*
        setBackground();
        <init>();
*/
}

/*

<init>(){
        <init>();
}

*/
```

〈リスト9-8〉channel変数の「8」という値を抽出したもの

```
/* Source Java File Name =
miditx.class */

main(){
        init();
        while(1){
                run();
        }
}

init(){
        channel = 8;
        system_initial();
}

run(){
        sleep();
        input_check();
        midi_transmit();
        repaint();
}

start(){
/*
        <init>();
        flow = 0;
        start();
*/
}

stop(){
/*
        stop();
*/
}

mouseDrag(){
/*
        event_check();
*/
}

mouseDown(){
/*
        event_check();
*/
}

event_check(){
/*
        flag = 9;
*/
}

input_check(){
/*
*/
}

midi_transmit(){
/*
*/
}

paint(){
/*
        flag = 0;
        setColor();
        fillRect();
        setColor();
        fillRect();
        setColor();
        fillRect();
        setColor();
        fillRect();
        setColor();
        setFont();
        <init>();
        append();
        valueOf();
        append();
        append();
        valueOf();
        append();
        toString();
        drawString();
*/
}

system_initial(){
/*
        setBackground();
        <init>();
        font = 16;
        px = 16;
        oy = 16;
        data = 16;
        flag = 16;
*/
}

/*

<init>(){
        <init>();
}

*/
```

▶ class_index = 10
▶ name_and_type_index = 67

というものであることがわかります．

ここで必要なのは名前ですから，つぎに「name_and_type_index = 67」に従って，ふたたびConstant_Poolのリストをこの番号で検索します．すると(5)のように，

▶ 種類はCONSTANT_NameAndType
▶ name_index = 105
▶ signature_index = 119

とわかり，さらに(6)のように同様に検索して，最終的なデータとして，「system_initial」という名前を獲得します．これを(7)のように別途に表示出力するようにしているというわけです．

このように，Javaのプログラム中で具体的に呼び出しているメソッド名が抜き出せるようになれば，もう峠は越えたようなものです．長くなるのでソースは省略しますが，**リスト9-5**のような結果を出力するために筆者が行った改造は，

▶ d_modeというフラグを設けた
▶ すべてのprintf()文に「if(d_mode==0)」というものを付加した
▶ 上記のメソッド名を表示する部分だけは「if(d_mode==1)」とした

nativeメソッド　そのマシンに固有の処理を行うように最適化されて開発されたメソッドのこと．ここではインターネット経由でダウンロードしたものではないもの．

〈リスト9-9〉 リスト9-8の出力からAKI-80に関係ないものを切り取ったもの[java_out.c]

```
/* Source Java File Name = miditx.class */

main(){
        init();
        while(1){
                run();
        }
}

init(){
        channel = 8;
        system_initial();
}

run(){
        sleep();
        input_check();
        midi_transmit();
        repaint();
}
```

▶ 最初に「d_mode=1;」を入れた

というだけのものです．この結果，**リスト9-5**では必要最低限の情報だけが抜き出されていることがわかります．

これをさらに進めたのが，**リスト9-6**(p.172)のようなコンバータによる変換で，AKI-80のCソースに自動変換するために，C言語の形態となるように関数名の出力フォーマットを加工しています．その結果，同じJavaプログラムが**リスト9-7**(p.157)のように出力されてきました．これはもう，予約しておくべき関数名の未定義を除けば，まったく問題のないCプログラムとなっています．筆者はさらに，**channel変数の「8」**という値を抽出するように改造して，**リスト9-8**(p.157)のように表示されるバージョンも実験していますが，本質的にはほとんど同じレベルのものとなっています．

このコンバータでは，実際にAKI-80で自由なシステムを製作して，そのソフトを何でもJavaでプログラム/シミュレーションできるというものではありません．しかし，Javaの考え方やソフトウェアの基本的な流れを理解するための教材としては，とても重要なエッセンスにあふれたものと言えると思います．少なくともここでの実例については，以降の環境を整備することで，本当に「AKI-80のMIDIシステムをJavaで開発」というところまで実現してみることにしましょう．

JavaによるAKI-80開発環境の実現

Javaは具体的な周辺ポートなどはまったく叩けないものですから，AKI-80のようなハードウェア丸出しのシステムの場合には，「Javaで仮想的に記述された処理を具体的な処理に置換する」という環境が必要になります．これは，前述のコンバータでなく，むしろAKI-80のCコンパイラの側に**サービス・ライブラリ**として提供するのが自然な方法でしょう．そこで，本節ではMini-CによるAKI-80開発環境と，前節でJavaプログラムから変換して得られているCプログラムとを結合するための環境を実際に構築してみます．これは，さらに異なるターゲットのシステムに拡張する場合のテンプレートとしても応用できるものとなるでしょう．

第7章で行った実験の最後の例，**リスト7-14**(p.99)と**リスト7-15**(p.93)の組み合わせがここでのヒントとなります．当面のターゲットでは，出力デバイスであるLCDパネルやMIDI入力の部分は省略できますから，簡単のために必要最低限の項目だけを整理して並べてみることにします．まず最初に，**リスト9-9**のような[java_out.c]というファイルを用意します．これは**リスト9-8**の出力から，さらにAKI-80システムに関係しないメソッド名をコメントアウト，あるいは切り取ったものです．実験のためには手作業で切り取ってもかまいませんし，前節で検討したコンバータをちょっと変更すれば，このような出力を得ることは簡単でしょう．ここでは，この[java_out.c]はそのまま加工せずに読み込まれる，つまりCプログラムにそのまま予約名としてincludeされるというところがポイントです．

次に，**リスト7-14**の[library.c]に相当するサービス・ライブラリとして，**リスト9-10**(p.173)のような[java_lib.c]というものを用意します．ここではスペースの関係でLCDモジュールの処理などを外していますが，**リスト7-14**と同じものを使っても問題ありません．

なお，[STARTUP.H]は，**リスト7-13**(p.98)のものからまったく手を加えないことにします．これはAKI-80システム用にチューニングされていれば，個々のアプリケーションごとには手を加えない「標準ライブラリ」であることが重要です．

そして最後に，このターゲットに特化しつつ，汎用性を最大限に持たせたC言語のメイン・モジュールとして，**リスト9-11**のような[test.c]を作ります．ここではまず，Mini-Cに特有の機能と冒頭の2行のinclude宣言によって，

▶ C言語環境のための[STARTUP.H]の自動組み込み

▶ AKI-80システム用サービス・ライブラリ[java_lib.c]の組み込み

プラグイン・モジュール ブラウザが，特定の処理を行うソフトウェア・モジュールとしてあらかじめ「プラグイン」として設定しておくと，該当する形式のデータを受信した時に自動的にこのソフトウェア・モジュールが対応する処理を実行してくれるもの．

〈リスト9-11〉C言語のメイン・モジュール test.c

```
#include "java_lib.c"
#include "java_out.c"

system_initial(){
        pio_a_setting( 0x00 );/* output */
        pio_b_setting( 0xff );/* input */
        midi_port_setting();
}

input_check(){
        int i,data;
        for(i=0;i<channel;i++){
                port_out( pio_a, i );
                data = port_in( pio_b );
                tx_midi_set( 0xd0 + i );
                tx_midi_set( data/2 );
        }
}

midi_transmit(){
        tx_midi_check();
}

/*****  Dummy Functions  ********************/

sleep(){}
repaint(){}
```

▶ Javaアプレットからコンバージョンされた[java_out.c]の組み込み

が行われます．

これに続くsystem_initial()という関数は，Javaプログラム中ではRunnableタイプの決まりパターンとして使われるinit()というメソッドから形式的に呼び出されているもので，ここでAKI-80システムを初期化することになります．この例では，Javaプログラムで**マウス・イベント**を検出している動作に対応するものとして，

▶ PIO_Aにチャネル数をセット
▶ この状態でPIO_Bを8ビット・ポートとしてデータを入力

という動作を想定したものとなっています．実際に，CPUバスと接続するタイプの多くのA-Dコンバータは，このような形式で複数チャネルを指定してデータを入力しますから，実際のシステムに即した変更は容易です．

system_initial()では，さらにmidi_port_setting()という共通の関数を呼び出します．これは第7章の実例がまったくそのまま使えます．

このあとのinput_check()とmidi_transmit()という二つの関数は，Javaプログラム中ではrun()メソッドの中の無限ループで形式的に呼ばれているものです．Javaではマウス・イベントはイベント・ハンドラで処理しますが，AKI-80ではこのように，具体的に入力のチェックを繰り返します．本当はこの例では，「前回のチェックとデータが変わっていない」という場合にもデータをMIDI出力してしまうので，実際的なシステムにする場合には「前回値と比較して，変化があれば出力」という判定処理を加えることになります．ここの例では，サービス・ライブラリに定義されているtx_midi_set()という関数をそのまま利用して，MIDIの「**チャネル・プレッシャ**」という形式によってMIDI出力しています．

Javaから無限ループの中で呼ばれているmidi_transmit()は，ここではすでに完成しているtx_midi_check()というサービス関数によって，十分に深いFIFOバッファによる確実なMIDI送信処理が簡単に実現できます．そして最後に，Javaのrun()メソッド中に置かれたsleep()とrepaint()というJavaアプレットとしての動作のみに必要な名前が，中身のないダミーとして置かれます．これによって，C言語としてエラーのないプログラムが形式的に完成します．

以上のような準備によって，このJavaから引き継がれたプログラムはちゃんとAKI-80のためのMini-C開発環境でコンパイルされ，アセンブルされ，さらにROMエミュレータに転送して実際に動作することができました．Netscape Navigatorの画面内で動作している**図9-1**のアプレットと，実際にAKI-80ボードでA-D入力のボリュームからMIDI出力されている動作とが，実は同じプログラムでつながっているというのはなかなか不思議な気がします．簡易型の実験とはいえ，世界最先端のJavaとシンプルなAKI-80とが結ばれたという事実は，筆者にとってある種の感慨を与えてくれました．

「JavaでAKI-80」の制限と課題

ここまでの長い道のりで，ようやくなんとか，JavaでAKI-80を動かすという最初の道すじをつけることができました．実際には，ここまでのところでは，ごく単純な例について実験してみたにすぎません．ここから実用レベルのシステムに挑戦してみると，途端にいろいろな壁に突き当たることは，筆者がもっとも痛感しています．

しかし，もともとJavaそのものがセキュリティに関係した多くの制限をもっていて，なかなか組み込みシステムなどで実用化されていないのも事実です．その意味では，今後，多くのメーカから「**Javaチップ**」が組み込み用途のために発表される時代を前にして，AKI-80という単純なシステムでJavaを体験する，さらにマイコン・システムの基本を理解するという本書のアプローチもそれなりの意義があると思います．

ここではまとめとして，簡単に「JavaでAKI-80の課題」を考えておくことにしましょう．まず最大の問題点は，なんといっても「AKI-80の処理能力の低さ」ということになります．JavaはPentiumクラス以上の

スレッド　Javaでは，複数のタスクを時分割的に並列処理する処理単位をスレッドと呼ぶ．

CPUでようやく動くのですが，これは考えてみると，ちょっと昔の「ミニコン」以上の強力なコンピュータ能力を必要としているということを意味しています．これをそのまま，クロック20MHz程度の8ビットCPUであるAKI-80システムで実現できるわけがありません．川崎製鉄の4倍速Z80でもちょっと無理というところでしょうか．

そして，本章でも最初から触れていることですが，Javaはもともとハードウェアを直接に叩くものではないのですが，マイコン・システムとは大げさなOSなどを用いずに周辺ハードを叩く軽快さが売りものです．この両者は，この点ではある意味で「相いれない」ものなのかもしれません．しかし，本書のようにアマチュアイズムでそこを強引に結び付けるという面白さもあると言えるかもしれません．

Javaとはインターネット時代の申し子ですから，言語仕様そのものから仮想的なシステムとして登場しました．一方のAKI-80のような古典的CPUは，内部の独特のレジスタ構造まで含めて，あらゆる仕様が現実のハードウェアに縛られています．この両者は本質的に異なるものであるのは当然なのですが，一方で，「スタック経由での処理」など，驚くほど共通の概念もあります．どちらか一方の専門馬鹿になるのでなく，このかけ離れた両者を視野に入れるというのは重要な技術だと思います．

このような基本的理解のもとで，なんとか両者を組み合わせた目標を設定して，実現に向けてチャレンジしていく，そういう柔軟な姿勢が最大の「課題」なのではないでしょうか．

AKI-80による インターネット端末の構想

さて，本章の最後に，せっかくのJavaに関連して，「AKI-80によるインターネット端末」という無謀な構想について触れておきましょう．

処理能力の高い新しいCPUがどんどん登場してくるのに，わざわざAKI-80にしがみつかなくてもかまわないのですが，ここでのポイントは「チープでシンプルなシステムでインターネットできるか」というチャレンジの姿勢にあります．

筆者がAKI-80を使ったMIDIシステムをいろいろと日曜大工的に開発した実感としては，処理能力だけでなく「システム実現の簡便さ」までを考慮すると，まだまだAKI-80のようなカード・マイコンは捨てたものではないという印象があります．すべてをパソコンというブラック・ボックスに任せてソフトウェアだけで行こうという姿勢は，どこか地に足が着いていないようで，心もとない側面があります．自分の作ったソフトがたまに暴走してしまうというときに，実はそれがパソコンのハードやOSの欠陥に起因するという可能性を考えてみればわかります．

たとえば，「Javaは安全だ」と言いますが，実際にはエラーのない（コンパイル成功した）Javaプログラムがパソコンやワークステーションをハングアップさせたという実例は山のようにあります．ここにはJavaの開発環境のバグ，ブラウザのバグ，OSのバグ，さらにハードの設計不良などが関係しています．「絶対に安心できるプラットフォーム」などというのは，いつの時代にも存在しないのです．ところがAKI-80システムで同様のトラブルがあれば，筆者はオシロスコープ片手に原因を明確に究明して，確実な対策をとることができます．パソコンでは無理なのに，カード・マイコンなら可能なのです．どちらが「信頼」できるシステムでしょうか．

そこで，AKI-80によって**ネットワーク対応のシステム**を開発するという意義もここにあります．別に開発言語にJavaを使う必要はありません．ハードが明瞭なシステムであれば，情報ネットワーキングについても確実に理解できるというのが最大のメリットともなるのです．もちろん複雑な階層のプロトコルをすべて制作するのはたいへんなことですが，利用できるソフトウェア部品はおおいに利用して，さらに必要があればオシロスコープのプローブを当てながら開発するというのはAKI-80クラスの得意な領域でしょう．

あまり高速のネットワーク処理はできないかもしれませんが，まだ**PHS**を利用したインターネットは9600 bpsがやっとという時代です．筆者の体感では，MIDI処理の能力からみて，コンパクトで軽快な処理端末を実現するプラットフォームとして，AKI-80をはじめとする8/16ビットCPUのカード・マイコンは，今後も面白いテーマを提供してくれるような気がしています．Windows，Mac，Unixマシンなどを駆使してネットワーキングしている筆者ですが，ここに簡単ツールとしてAKI-80マシンを加えてみるというテーマにチャレンジしていきたいと思っているところです．

興味のある方は，どうぞ一緒に挑戦していきましょう．

Xウィンドウ・システム　Unix上でグラフィカル・ユーザ・インターフェースを実現するためのソフトウェア環境のこと．簡単なプログラミングで複雑な対話的処理などを実現できる．

〈リスト9-1〉
Javaアプレットのソース・プログラム miditx.java(8系統の入力データに対して別々のMIDIチャネルを添えたバリューとしてMIDI出力する)

```
import java.awt.*;
import java.applet.*;
import java.lang.*;
import java.io.*;

public class miditx extends Applet implements Runnable{
    int py[],oy[],data[],flag,channel;
    Graphics g;
    private Thread flow;
    Font font;

    public void init(){
        channel = 8;
        system_initial();
    }

    public void run(){
        try {
                while(true){
                        Thread.sleep(100);
                        input_check();
                        if( flag != 0 ){
                        midi_transmit();
                        repaint();
                        }
                }
        }
        catch(Exception e) {}
    }

    public void start(){
        flow = new Thread(this);
        flow.start();
    }

    public void stop(){
        flow.stop();
    }

    public boolean mouseDrag(Event e, int x, int y){
        event_check(x,y);
        return true;
    }

    public boolean mouseDown(Event e, int x, int y){
        event_check(x,y);
        return true;
    }

    void event_check(int x, int y){
        int z;
        for(int i=0;i<channel;i++){
                z = 40 + 40*i;
                if( (x>z-10) && (z+10>x) && (9<y) && (191>y) ){
                        oy[i] = py[i]; py[i] = y;
                        flag++;
                }
        }
    }

    void input_check(){
        /* dummy */
    }

    void midi_transmit(){
        /* dummy */
    }

    public void paint(Graphics g){
        int z;
        flag = 0;
        for(int i=0;i<channel;i++){
           z = 40+40*i;
           g.setColor(Color.black);
           g.fillRect(z-10,oy[i]-5,20,10);
           g.setColor(Color.yellow);
           g.fillRect(z-1,10,2,180);
           g.setColor(Color.green);
           g.fillRect(z-10,py[i]-5,20,10);
           g.setColor(Color.blue);
           g.fillRect(z-8,py[i]-3,16,6);
           data[i] = (190-py[i])*127/180;
           g.setColor(Color.white);
           g.setFont(font);
           String word = "Data(" + String.valueOf(i+1) + ") = " + String.valueOf(data[i]);
           g.drawString(word, 355, 30+22*i);
        }
    }

    void system_initial(){
        setBackground(Color.black);
        font = new java.awt.Font("TimesRoman", Font.PLAIN, 16);
        py = new int[channel]; oy = new int[channel]; data = new int[channel];
        for(int i=0;i<channel;i++) py[i] = 190;
        flag = 0;
    }

}
```


簡易型のコンバータ　ここでは「JavaからCへのコンバータ」の意味.

〈リスト9-2〉リスト8-9(p.112)の改造[rejava2.c] ①

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

FILE *fp;
int ni[1000], m[15000], p[300][10];
char n[300][60];

static void access_flag_printf(int para);
static void byte_code_printf(long address);
static void display_information(int mode, int id);

int main(int argc,char **argv){
        int d, i, j, k, ct;
        unsigned int p0, p00, p1, p2, p3;
        int pp=0, count=0, p4, p5;
        unsigned char ss, st[20], d0, d1, d2, d3;
        st[16]=0;
        if( argc != 2 ){ printf("... target file name missing (;_;) ...¥n¥n"); exit(1); }
        else if( (fp=fopen( argv[1], "rb" ))==NULL ){ printf("... target file [ %s ] is not found (;_;) ...¥n¥n", argv[1]); exit(1); }
        printf("Target Java File Name = %s¥n", argv[1]);
        while( (d = fgetc(fp)) >= 0 ){
                m[count++] = d & 0xff;
        }
        fclose(fp);
        printf("¥nTotal File Length = %d bytes.", count);
        p0 = m[pp+3]+256*m[pp+2];
        p00 = m[pp+1]+256*m[pp];
        printf("¥n¥n¥t%06X  Magic Code [CAFEBABE] = %04X%04X",pp,p00,p0); pp += 4;
        if(( p00 != 0xcafe )||( p0 != 0xbabe )){
                printf("¥n¥n... target file is not Java (;_;) ...¥n¥n"); exit(1);
        }
        else printf("  ---  OK (^_^)");
        printf("¥n¥n¥t%06X  Version : major version %d , minor version %d", pp, m[pp+3]+256*m[pp+2], m[pp+1]+256*m[pp] ); pp += 4;
        p1 = m[pp+1]+256*m[pp]; printf("¥n¥n¥t%06X  Constant Pool : total number = %d¥n", pp, p1 ); pp += 2;
        for(i=1;i<p1;i++){
                printf("¥n¥t¥t%06X   constant_pool[%03d][%02X] = ", pp, i, i);
                d0 = m[pp++]; printf("<%X> ",d0);
                switch(d0){
                        case 7: p2 = m[pp+1]+256*m[pp]; pp += 2; ni[i] = p2;
                                printf("CONSTANT_Class , name_index = %d", p2);
                                p[i][2] = p2;
                                break;
                        case 9: p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_Fieldref , class_index = %d", p2);
                                printf(" , name_and_type_index = %d", p3);
                                p[i][3] = p3;
                                break;
                        case 10: p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_Methodref , class_index = %d", p2);
                                printf(" , name_and_type_index = %d", p3);
                                p[i][0] = p3;
                                break;
                        case 11: p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_InterfaceMethodref , class_index = %d", p2);
                                printf(" , name_and_type_index = %d", p3);
                                break;
                        case 8: p2 = m[pp+1]+256*m[pp]; pp += 2; ni[i] = p2;
                                printf("CONSTANT_String , name_index = %d", p2);
                                p[i][4] = p2;
                                break;
                        case 3: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Integer , value = %d", p4);
                                break;
                        case 4: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Float , value = %d", p4);
                                break;
                        case 5: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                p5 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Float , high = %d , low = %d", p4, p5); i++;
                                break;
                        case 6: p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                p5 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("CONSTANT_Double , high = %d , low = %d", p4, p5); i++;
                                break;
                        case 12: p2 = m[pp+1]+256*m[pp]; pp += 2;
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("CONSTANT_NameAndType , name_index = %d", p2); ni[i] = p2;
                                printf(" , signature_index = %d", p3);
                                p[i][1] = p2;
                                break;
                       case 1: case 2:
                                p2 = m[pp+1]+256*m[pp]; pp += 2;
                                if(d0==1) printf("CONSTANT_Utf8 , ");
                                else printf("CONSTANT_Unicode , ");
                                k = 0;
                                for(j=0;j<p2;j++){
                                        d1 = m[pp++];
                                        if((d1&0x80)==0) n[i][k++] = d1;
                                        else{
                                                d2 = m[pp++];
                                                if((d1&0xe0)==0xc0){
                                                        n[i][k++] = (64*(d1&0x1f)+(d2&0x3f))/256;
                                                        n[i][k++] = (64*(d1&0x1f)+(d2&0x3f))%256;
                                                }
                                                else{
                                                        d3 = m[pp++];
                   n[i][k++] = (256*16*(d1&0x0f)+64*(d2&0x3f)+(d3&0x3f))/256;
                                                        n[i][k++] = (256*16*(d1&0x0f)+64*(d2&0x3f)+(d3&0x3f))%256;
                                                }
                                        }
                                }
                }
```


広域変数　Cのスタティック変数，アセンブラのセグメント外変数というように，ソフトウェア・システムの全体で共通に参照されるように宣言された変数という意味．

〈リスト9-2〉リスト8-9(p.112)の改造[rejava2.c] ②

```
                                n[i][k] = 0; printf("Data = %s",n[i]);
                                break;
                        default:        printf("\nFormat Error !!!\n");
                                break;
                }
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%06X  Access Flag = %04X --- ", pp-2, p1 );
        access_flag_printf( p1 );
        p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%06X  This Class = %s", pp-2, n[ni[p1]] );
        p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%06X  Super Class = %s", pp-2, n[ni[p1]] );
        p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%06X  Interface Count = %d", pp-2, p1 );
        if( p1 != 0 ){
                printf("¥n");
                for(i=0;i<p1;i++){
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("¥n¥t¥t%06X  interface[%d] : Name = %s", pp-2, i, n[ni[p2]] );
                }
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%06X  Fields Count = %d¥n", pp-2, p1 );
        for(i=0;i<p1;i++){
                printf("¥n¥t¥t%06X   field[%d] : ", pp, i);
                p2 = m[pp+1]+256*m[pp]; pp += 2; access_flag_printf( p2 );
                p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Name = %s", n[p2] );
                p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Signature = %s", n[p2] );
                p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Attribute = %d", p2 );
                if( p2 != 0 ){
                        printf("¥n");
                        for(j=0;j<p2;j++){
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("¥n¥t¥t¥ttype = %s , ", n[p3] );
                                p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("length = %d¥n", p4);
                                for(k=0;k<p4;k++){
                                        if((k%25)==0) printf("¥n¥t¥t¥t¥t");
                                        printf("%02X ", m[pp++] );
                                }
                        }
                }
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%06X  Methods Count = %d", pp-2, p1 );
        for(i=0;i<p1;i++){
                printf("¥n¥n¥t¥t%06X   method[%d] : ", pp, i);
                p2 = m[pp+1]+256*m[pp]; pp += 2; access_flag_printf( p2 );
                p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Name = %s", n[p2] );
                p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Signature = %s", n[p2] );
                p2 = m[pp+1]+256*m[pp]; pp += 2; printf(" , Attribute = %d", p2 );
                if( p2 != 0 ){
                        printf("¥n");
                        for(j=0;j<p2;j++){
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                printf("¥n¥t¥t¥ttype = %s , ", n[p3] );
                                p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                printf("length = %d", p4);
                                p5 = pp; pp += p4;
                                if( strcmp( n[p3], "Code" ) == 0 ) byte_code_printf( p5 );
                                else if( strcmp( n[p3], "Exceptions" ) == 0 ) byte_code_printf( p5 );
                        }
                }
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2; printf("¥n¥n¥t%06X  Attributes Count = %d", pp-2, p1 );
        if( p1 != 0 ){
                for(i=0;i<p1;i++){
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("¥n¥n¥t¥t%06X  attribute[%d] : type = %s , ", pp-2, i, n[p2] );
                        p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                        p5 = pp;
                        printf("length = %d¥n", p4);
                        for(k=0;k<p4;k++){
                                if((k%25)==0) printf("¥n¥t¥t¥t¥t");
                                printf("%02X ", m[pp++] );
                        }
                        if( ( p4 == 2 ) && ( strcmp( n[p2], "SourceFile" ) == 0 ) ){
                                p3 = m[p5+1]+256*m[p5];
                                printf("¥n¥n¥t¥t¥tSource File Name = %s", n[p3] );
                        }
                }
        }
        printf("¥n¥n..... (^_^) Java analyze completely finished. (^_^)¥n¥n");          return;
}


void byte_code_printf( long address ){
        int k, l, p1, p2, p3, mm, m1, m2, m3, m4;
        long pp, p4, p5;
        pp = address;
        p1 = m[pp+1]+256*m[pp]; pp += 2;
        p2 = m[pp+1]+256*m[pp]; pp += 2;
        printf(" , max_stack = %d , max_locals = %d , ", p1, p2 );
        p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
        printf("code_length = %d\n", p4);
        k = 0;
        while( k < p4 ){
                printf("¥n¥t¥t¥t¥t%06X¥t", k );
                mm = m[pp++];
                switch(mm){
                        default :       printf("?? %02X ??¥t¥t(..... not defined .....)", mm);
                                break;
/***** この部分は以下にあるもの以外はリスト8-9と同様なので省略 *****/
                        case 181:       m1 = m[pp++]; m2 = m[pp++];
                                printf("putfield¥t%04X¥t(Set field in object)", 256*m1+m2);
                                display_information(5,256*m1+m2);
                                k += 2; break;
```

〈リスト9-2〉リスト8-9(p.112)の改造[rejava2.c] ③

```
                        case 180:       m1 = m[pp++]; m2 = m[pp++];
                                printf("getfield¥t%04X¥t(Fetch field from object)", 256*m1+m2);
                                display_information(6,256*m1+m2);
                                k += 2; break;
                        case 178:       m1 = m[pp++]; m2 = m[pp++];
                                printf("getstatic¥t%04X¥t(Get static field from class)", 256*m1+m2);
                                display_information(3,256*m1+m2);
                                k += 2; break;
                        case 182:       m1 = m[pp++]; m2 = m[pp++];
                                printf("invokevirtual¥t%04X¥t(Invoke instance method)", 256*m1+m2);
                                display_information(1,256*m1+m2);
                                k += 2; break;
                        case 183:       m1 = m[pp++]; m2 = m[pp++];
                                        printf("invokenonvirt¥t%04X¥t(Invoke instance method, dispatching based on compile-time type)",
256*m1+m2);
                                display_information(1,256*m1+m2);
                                k += 2; break;
                        case 184:       m1 = m[pp++]; m2 = m[pp++];
                                printf("invokestatic¥t%04X¥t(Invoke a class (static) method)", 256*m1+m2);
                                display_information(1,256*m1+m2);
                                k += 2; break;
                        case 185:       m1 = m[pp++]; m2 = m[pp++];
                                m3 = m[pp++]; m4 = m[pp++];
                                printf("invokeinterf¥t%04X %02X %02X¥t(Invoke interface method)", 256*m1+m2, m3, m4);
                                display_information(1,256*m1+m2);
                                k += 2; break;
                        case 187:       m1 = m[pp++]; m2 = m[pp++];
                                printf("new¥t%04X\t(Create new object)", 256*m1+m2);
                                display_information(2,256*m1+m2);
                                k += 2; break;
                }
                k++;
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2;
        printf("¥n¥n¥t¥t¥texception_table_length = %d", p1 );
        if( p1 != 0 ){
                printf("¥n");
                for(k=0;k<p1;k++){
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("¥n¥t¥t¥t¥t%06X  start_pc = %d , ", pp, p2 );
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("end_pc = %d , ", p2 );
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("handler_pc = %d , ", p2 );
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        if( p2 != 0 ) printf("catch_type = %s", n[ni[p2]] );
                        else printf("catch_type = all exceptions");
                }
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2;
        printf("¥n¥n¥t¥t¥tattributes_count = %d", p1 );
        if( p1 != 0 ){
                printf("¥n");
                for(k=0;k<p1;k++){
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        printf("¥n¥t¥t¥t¥t%06X  type = %s , ", pp, n[p2] );
                        p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                        printf("attribute_length = %d¥n", p4);
                        for(l=0;l<p4;l++){
                                if((l%20)==0) printf("¥n¥t¥t¥t¥t¥t");
                                printf("%02X ", m[pp++] );
                        }
                }
        }
}

void display_information(int mode, int id){
        switch(mode){
                case 1:
                        printf("¥n¥t¥t¥t¥t¥t     ##### Call [ %s ]#####", n[(p[(p[id][0])][1])]);
                        break;
                case 2:
                        printf("¥n¥t¥t¥t¥t¥t     ##### Define [ %s ]#####", n[(p[id][2])]);
                        break;
                case 3:
                        printf("¥n¥t¥t¥t¥t¥t     ##### Get Static [ %s ]#####", n[(p[(p[id][3])][1])]);
                        break;
                case 4:
                        printf("¥n¥t¥t¥t¥t¥t     ##### Push Item [ %s ]#####", n[(p[id][4])]);
                        break;
                case 5:
                        printf("¥n¥t¥t¥t¥t¥t     ##### Put Field [ %s ]#####", n[(p[(p[id][3])][1])]);
                        break;
                case 6:
                        printf("¥n¥t¥t¥t¥t¥t     ##### Get Field [ %s ]#####", n[(p[(p[id][3])][1])]);
                        break;
        }
}

void access_flag_printf(int para){
        switch(para){
                case 0x0001:    printf("ACC_PUBLIC");   break;
                case 0x0002:    printf("ACC_PRIVATE");  break;
                case 0x0004:    printf("ACC_PROTECTED");        break;
                case 0x0008:    printf("ACC_STATIC");   break;
                case 0x0010:    printf("ACC_FINAL");    break;
                case 0x0020:    printf("ACC_SYNCHRONIZED");     break;
                case 0x0040:    printf("ACC_VOLATILE"); break;
                case 0x0080:    printf("ACC_TRANSIENT");        break;
                case 0x0100:    printf("ACC_NATIVE");   break;
                case 0x0200:    printf("ACC_INTERFACE");        break;
                case 0x0400:    printf("ACC_ABSTRACT"); break;
                default:        printf("( no flag )");  break;
        }
}
```


サービス・ライブラリ　関連する処理をサブルーチン群として多数並べてソース・プログラム中に置いておくというライブラリ．実際に使うかどうかはそれぞれのターゲットによって変わる．

〈リスト9-3〉 rejava2によってmiditx.classを解析したもの ①

```
Target Java File Name = miditx.class

Total File Length = 2618 bytes.

       000000  Magic Code [CAFEBABE] = CAFEBABE  ---  OK (^_^)

       000004  Version : major version 45 , minor version 3

       000008  Constant Pool : total number = 155

               00000A   constant_pool[001][01] = <8> CONSTANT_String , name_index = 87
               00000D   constant_pool[002][02] = <8> CONSTANT_String , name_index = 93
               000010   constant_pool[003][03] = <8> CONSTANT_String , name_index = 83
               000013   constant_pool[004][04] = <7> CONSTANT_Class , name_index = 101
               000016   constant_pool[005][05] = <7> CONSTANT_Class , name_index = 117
               000019   constant_pool[006][06] = <7> CONSTANT_Class , name_index = 134
               00001C   constant_pool[007][07] = <7> CONSTANT_Class , name_index = 124
               00001F   constant_pool[008][08] = <7> CONSTANT_Class , name_index = 133
               000022   constant_pool[009][09] = <7> CONSTANT_Class , name_index = 145
               000025   constant_pool[010][0A] = <7> CONSTANT_Class , name_index = 131
               000028   constant_pool[011][0B] = <7> CONSTANT_Class , name_index = 118
               00002B   constant_pool[012][0C] = <7> CONSTANT_Class , name_index = 81
               00002E   constant_pool[013][0D] = <7> CONSTANT_Class , name_index = 116
               000031   constant_pool[014][0E] = <7> CONSTANT_Class , name_index = 130
               000034   constant_pool[015][0F] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 65
               000039   constant_pool[016][10] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 52
               00003E   constant_pool[017][11] = <9> CONSTANT_Fieldref , class_index = 12 , name_and_type_index = 79
               000043   constant_pool[018][12] = <A> CONSTANT_Methodref , class_index = 10 , name_and_type_index = 68
               000048   constant_pool[019][13] = <9> CONSTANT_Fieldref , class_index = 10 , name_and_type_index = 76
               00004D   constant_pool[020][14] = <A> CONSTANT_Methodref , class_index = 5 , name_and_type_index = 56
               000052   constant_pool[021][15] = <A> CONSTANT_Methodref , class_index = 6 , name_and_type_index = 64
               000057   constant_pool[022][16] = <A> CONSTANT_Methodref , class_index = 11 , name_and_type_index = 62
               00005C   constant_pool[023][17] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 49
               000061   constant_pool[024][18] = <A> CONSTANT_Methodref , class_index = 6 , name_and_type_index = 58
               000066   constant_pool[025][19] = <9> CONSTANT_Fieldref , class_index = 12 , name_and_type_index = 63
               00006B   constant_pool[026][1A] = <9> CONSTANT_Fieldref , class_index = 10 , name_and_type_index = 72
               000070   constant_pool[027][1B] = <A> CONSTANT_Methodref , class_index = 6 , name_and_type_index = 54
               000075   constant_pool[028][1C] = <A> CONSTANT_Methodref , class_index = 6 , name_and_type_index = 61
               00007A   constant_pool[029][1D] = <9> CONSTANT_Fieldref , class_index = 12 , name_and_type_index = 74
               00007F   constant_pool[030][1E] = <9> CONSTANT_Fieldref , class_index = 10 , name_and_type_index = 48
               000084   constant_pool[031][1F] = <9> CONSTANT_Fieldref , class_index = 12 , name_and_type_index = 78
               000089   constant_pool[032][20] = <A> CONSTANT_Methodref , class_index = 10 , name_and_type_index = 77
               00008E   constant_pool[033][21] = <A> CONSTANT_Methodref , class_index = 13 , name_and_type_index = 60
               000093   constant_pool[034][22] = <A> CONSTANT_Methodref , class_index = 13 , name_and_type_index = 66
               000098   constant_pool[035][23] = <A> CONSTANT_Methodref , class_index = 10 , name_and_type_index = 50
               00009D   constant_pool[036][24] = <A> CONSTANT_Methodref , class_index = 11 , name_and_type_index = 75
               0000A2   constant_pool[037][25] = <A> CONSTANT_Methodref , class_index = 7 , name_and_type_index = 53
               0000A7   constant_pool[038][26] = <A> CONSTANT_Methodref , class_index = 11 , name_and_type_index = 69
               0000AC   constant_pool[039][27] = <9> CONSTANT_Fieldref , class_index = 10 , name_and_type_index = 55
               0000B1   constant_pool[040][28] = <9> CONSTANT_Fieldref , class_index = 10 , name_and_type_index = 57
               0000B6   constant_pool[041][29] = <A> CONSTANT_Methodref , class_index = 9 , name_and_type_index = 62
               0000BB   constant_pool[042][2A] = <A> CONSTANT_Methodref , class_index = 10 , name_and_type_index = 67
               0000C0   constant_pool[043][2B] = <9> CONSTANT_Fieldref , class_index = 12 , name_and_type_index = 47
               0000C5   constant_pool[044][2C] = <A> CONSTANT_Methodref , class_index = 4 , name_and_type_index = 73
               0000CA   constant_pool[045][2D] = <9> CONSTANT_Fieldref , class_index = 10 , name_and_type_index = 59
               0000CF   constant_pool[046][2E] = <9> CONSTANT_Fieldref , class_index = 10 , name_and_type_index = 51
               0000D4   constant_pool[047][2F] = <C> CONSTANT_NameAndType , name_index = 96 , signature_index = 150
               0000D9   constant_pool[048][30] = <C> CONSTANT_NameAndType , name_index = 120 , signature_index = 121
               0000DE   constant_pool[049][31] = <C> CONSTANT_NameAndType , name_index = 100 , signature_index = 119
               0000E3   constant_pool[050][32] = <C> CONSTANT_NameAndType , name_index = 122 , signature_index = 126
               0000E8   constant_pool[051][33] = <C> CONSTANT_NameAndType , name_index = 102 , signature_index = 143
               0000ED   constant_pool[052][34] = <C> CONSTANT_NameAndType , name_index = 85 , signature_index = 88
               0000F2   constant_pool[053][35] = <C> CONSTANT_NameAndType , name_index = 128 , signature_index = 152
               0000F7   constant_pool[054][36] = <C> CONSTANT_NameAndType , name_index = 138 , signature_index = 84
               0000FC   constant_pool[055][37] = <C> CONSTANT_NameAndType , name_index = 153 , signature_index = 151
               000101   constant_pool[056][38] = <C> CONSTANT_NameAndType , name_index = 85 , signature_index = 125
               000106   constant_pool[057][39] = <C> CONSTANT_NameAndType , name_index = 154 , signature_index = 91
               00010B   constant_pool[058][3A] = <C> CONSTANT_NameAndType , name_index = 123 , signature_index = 148
               000110   constant_pool[059][3B] = <C> CONSTANT_NameAndType , name_index = 108 , signature_index = 143
               000115   constant_pool[060][3C] = <C> CONSTANT_NameAndType , name_index = 98 , signature_index = 119
               00011A   constant_pool[061][3D] = <C> CONSTANT_NameAndType , name_index = 132 , signature_index = 110
               00011F   constant_pool[062][3E] = <C> CONSTANT_NameAndType , name_index = 85 , signature_index = 119
               000124   constant_pool[063][3F] = <C> CONSTANT_NameAndType , name_index = 103 , signature_index = 150
               000129   constant_pool[064][40] = <C> CONSTANT_NameAndType , name_index = 97 , signature_index = 125
               00012E   constant_pool[065][41] = <C> CONSTANT_NameAndType , name_index = 94 , signature_index = 119
               000133   constant_pool[066][42] = <C> CONSTANT_NameAndType , name_index = 114 , signature_index = 110
               000138   constant_pool[067][43] = <C> CONSTANT_NameAndType , name_index = 105 , signature_index = 119
               00013D   constant_pool[068][44] = <C> CONSTANT_NameAndType , name_index = 147 , signature_index = 119
               000142   constant_pool[069][45] = <C> CONSTANT_NameAndType , name_index = 144 , signature_index = 106
               000147   constant_pool[070][46] = <5> CONSTANT_Float , high = 0 , low = 100
               000150   constant_pool[072][48] = <C> CONSTANT_NameAndType , name_index = 127 , signature_index = 121
               000155   constant_pool[073][49] = <C> CONSTANT_NameAndType , name_index = 113 , signature_index = 82
               00015A   constant_pool[074][4A] = <C> CONSTANT_NameAndType , name_index = 135 , signature_index = 150
               00015F   constant_pool[075][4B] = <C> CONSTANT_NameAndType , name_index = 89 , signature_index = 107
               000164   constant_pool[076][4C] = <C> CONSTANT_NameAndType , name_index = 115 , signature_index = 121
               000169   constant_pool[077][4D] = <C> CONSTANT_NameAndType , name_index = 104 , signature_index = 119
               00016E   constant_pool[078][4E] = <C> CONSTANT_NameAndType , name_index = 111 , signature_index = 150
               000173   constant_pool[079][4F] = <C> CONSTANT_NameAndType , name_index = 109 , signature_index = 150
               000178   constant_pool[080][50] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Graphics;)V
               000191   constant_pool[081][51] = <1> CONSTANT_Utf8 , Data = java/awt/Color
               0001A2   constant_pool[082][52] = <1> CONSTANT_Utf8 , Data = (I)Ljava/lang/String;
               0001BA   constant_pool[083][53] = <1> CONSTANT_Utf8 , Data = TimesRoman
               0001C7   constant_pool[084][54] = <1> CONSTANT_Utf8 , Data = (IIII)V
               0001D1   constant_pool[085][55] = <1> CONSTANT_Utf8 , Data = <init>
               0001DA   constant_pool[086][56] = <1> CONSTANT_Utf8 , Data = SourceFile
               0001E7   constant_pool[087][57] = <1> CONSTANT_Utf8 , Data = ) =
               0001EE   constant_pool[088][58] = <1> CONSTANT_Utf8 , Data = (Ljava/lang/Runnable;)V
               000208   constant_pool[089][59] = <1> CONSTANT_Utf8 , Data = toString
               000213   constant_pool[090][5A] = <1> CONSTANT_Utf8 , Data = miditx.java
               000221   constant_pool[091][5B] = <1> CONSTANT_Utf8 , Data = Ljava/awt/Font;
               000233   constant_pool[092][5C] = <1> CONSTANT_Utf8 , Data = Exceptions
```


マウス・イベント　Javaではマシンに依存しないので，マウス・ボタンが一つ(Mac)でも二つ(Windows)でも三つ(Unix)でも，どれかのボタンが押されればマウス・イベントが発生する．

〈リスト9-3〉 rejava2によってmiditx.classを解析したもの ②

```
        000240   constant_pool[093][5D] = <1> CONSTANT_Utf8 , Data = Data(
        000248   constant_pool[094][5E] = <1> CONSTANT_Utf8 , Data = stop
        00024F   constant_pool[095][5F] = <1> CONSTANT_Utf8 , Data = LineNumberTable
        000261   constant_pool[096][60] = <1> CONSTANT_Utf8 , Data = black
        000269   constant_pool[097][61] = <1> CONSTANT_Utf8 , Data = drawString
        000276   constant_pool[098][62] = <1> CONSTANT_Utf8 , Data = repaint
        000280   constant_pool[099][63] = <1> CONSTANT_Utf8 , Data = Ljava/awt/Graphics;
        000296   constant_pool[100][64] = <1> CONSTANT_Utf8 , Data = start
        00029E   constant_pool[101][65] = <1> CONSTANT_Utf8 , Data = java/lang/String
        0002B1   constant_pool[102][66] = <1> CONSTANT_Utf8 , Data = flag
        0002B8   constant_pool[103][67] = <1> CONSTANT_Utf8 , Data = blue
        0002BF   constant_pool[104][68] = <1> CONSTANT_Utf8 , Data = midi_transmit
        0002CF   constant_pool[105][69] = <1> CONSTANT_Utf8 , Data = system_initial
        0002E0   constant_pool[106][6A] = <1> CONSTANT_Utf8 , Data = (Ljava/lang/String;)Ljava/lang/StringBuffer;
        00030F   constant_pool[107][6B] = <1> CONSTANT_Utf8 , Data = ()Ljava/lang/String;
        000326   constant_pool[108][6C] = <1> CONSTANT_Utf8 , Data = channel
        000330   constant_pool[109][6D] = <1> CONSTANT_Utf8 , Data = white
        000338   constant_pool[110][6E] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Color;)V
        00034E   constant_pool[111][6F] = <1> CONSTANT_Utf8 , Data = yellow
        000357   constant_pool[112][70] = <1> CONSTANT_Utf8 , Data = init
        00035E   constant_pool[113][71] = <1> CONSTANT_Utf8 , Data = valueOf
        000368   constant_pool[114][72] = <1> CONSTANT_Utf8 , Data = setBackground
        000378   constant_pool[115][73] = <1> CONSTANT_Utf8 , Data = data
        00037F   constant_pool[116][74] = <1> CONSTANT_Utf8 , Data = java/awt/Component
        000394   constant_pool[117][75] = <1> CONSTANT_Utf8 , Data = java/awt/Font
        0003A4   constant_pool[118][76] = <1> CONSTANT_Utf8 , Data = java/lang/StringBuffer
        0003BD   constant_pool[119][77] = <1> CONSTANT_Utf8 , Data = ()V
        0003C3   constant_pool[120][78] = <1> CONSTANT_Utf8 , Data = py
        0003C8   constant_pool[121][79] = <1> CONSTANT_Utf8 , Data = [I
        0003CD   constant_pool[122][7A] = <1> CONSTANT_Utf8 , Data = event_check
        0003DB   constant_pool[123][7B] = <1> CONSTANT_Utf8 , Data = setFont
        0003E5   constant_pool[124][7C] = <1> CONSTANT_Utf8 , Data = java/lang/Thread
        0003F8   constant_pool[125][7D] = <1> CONSTANT_Utf8 , Data = (Ljava/lang/String;II)V
        000412   constant_pool[126][7E] = <1> CONSTANT_Utf8 , Data = (II)V
        00041A   constant_pool[127][7F] = <1> CONSTANT_Utf8 , Data = oy
        00041F   constant_pool[128][80] = <1> CONSTANT_Utf8 , Data = sleep
        000427   constant_pool[129][81] = <1> CONSTANT_Utf8 , Data = run
        00042D   constant_pool[130][82] = <1> CONSTANT_Utf8 , Data = java/lang/Exception
        000443   constant_pool[131][83] = <1> CONSTANT_Utf8 , Data = miditx
        00044C   constant_pool[132][84] = <1> CONSTANT_Utf8 , Data = setColor
        000457   constant_pool[133][85] = <1> CONSTANT_Utf8 , Data = java/lang/Runnable
        00046C   constant_pool[134][86] = <1> CONSTANT_Utf8 , Data = java/awt/Graphics
        000480   constant_pool[135][87] = <1> CONSTANT_Utf8 , Data = green
        000488   constant_pool[136][88] = <1> CONSTANT_Utf8 , Data = mouseDrag
        000494   constant_pool[137][89] = <1> CONSTANT_Utf8 , Data = g
        000498   constant_pool[138][8A] = <1> CONSTANT_Utf8 , Data = fillRect
        0004A3   constant_pool[139][8B] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Event;II)Z
        0004BB   constant_pool[140][8C] = <1> CONSTANT_Utf8 , Data = mouseDown
        0004C7   constant_pool[141][8D] = <1> CONSTANT_Utf8 , Data = paint
        0004CF   constant_pool[142][8E] = <1> CONSTANT_Utf8 , Data = ConstantValue
        0004DF   constant_pool[143][8F] = <1> CONSTANT_Utf8 , Data = I
        0004E3   constant_pool[144][90] = <1> CONSTANT_Utf8 , Data = append
        0004EC   constant_pool[145][91] = <1> CONSTANT_Utf8 , Data = java/applet/Applet
        000501   constant_pool[146][92] = <1> CONSTANT_Utf8 , Data = Code
        000508   constant_pool[147][93] = <1> CONSTANT_Utf8 , Data = input_check
        000516   constant_pool[148][94] = <1> CONSTANT_Utf8 , Data = (Ljava/awt/Font;)V
        00052B   constant_pool[149][95] = <1> CONSTANT_Utf8 , Data = LocalVariables
        00053C   constant_pool[150][96] = <1> CONSTANT_Utf8 , Data = Ljava/awt/Color;
        00054F   constant_pool[151][97] = <1> CONSTANT_Utf8 , Data = Ljava/lang/Thread;
        000564   constant_pool[152][98] = <1> CONSTANT_Utf8 , Data = (J)V
        00056B   constant_pool[153][99] = <1> CONSTANT_Utf8 , Data = flow
        000572   constant_pool[154][9A] = <1> CONSTANT_Utf8 , Data = font

000579  Access Flag = 0001 --- ACC_PUBLIC

00057B  This Class = miditx

00057D  Super Class = java/applet/Applet

00057F  Interface Count = 1

        000581  interface[0] : Name = java/lang/Runnable

000583  Fields Count = 8

        000585   field[0] : ( no flag ) , Name = py , Signature = [I , Attribute = 0
        00058D   field[1] : ( no flag ) , Name = oy , Signature = [I , Attribute = 0
        000595   field[2] : ( no flag ) , Name = data , Signature = [I , Attribute = 0
        00059D   field[3] : ( no flag ) , Name = flag , Signature = I , Attribute = 0
        0005A5   field[4] : ( no flag ) , Name = channel , Signature = I , Attribute = 0
        0005AD   field[5] : ( no flag ) , Name = g , Signature = Ljava/awt/Graphics; , Attribute = 0
        0005B5   field[6] : ACC_PRIVATE , Name = flow , Signature = Ljava/lang/Thread; , Attribute = 0
        0005BD   field[7] : ( no flag ) , Name = font , Signature = Ljava/awt/Font; , Attribute = 0

0005C5  Methods Count = 12

        0005C7   method[0] : ACC_PUBLIC , Name = init , Signature = ()V , Attribute = 1

                type = Code , length = 43 , max_stack = 2 , max_locals = 1 , code_length = 11

                        000000  aload_0                 (Load object reference from local variable)
                        000001  bipush  08              (Push one-byte signed integer)
                        000003  putfield        002D    (Set field in object)
                                ##### Put Field [ channel ]#####
                        000006  aload_0                 (Load object reference from local variable)
                        000007  invokevirtual   002A    (Invoke instance method)
                                ##### Call [ system_initial ]#####
                        00000A  return                  (Return (void) from procedure)

                exception_table_length = 0
```


チャネル・プレッシャ あるMIDIチャネルの全体に影響するパラメータで，イメージとしては，複音シンセサイザのキーボードの下に共通の圧力センサがあって，これを押す圧力の情報を送るのが一般的である．

〈リスト9-3〉 rejava2によってmiditx.classを解析したもの ③

```
        attributes_count = 1

                0005EE  type = LineNumberTable , attribute_length = 14

                        00 03 00 00 00 0D 00 06 00 0E 00 0A 00 0C

000600   method[1] : ACC_PUBLIC , Name = run , Signature = ()V , Attribute = 1

        type = Code , length = 94 , max_stack = 2 , max_locals = 1 , code_length = 30

                000000  ldc2w   0046            (Push long or double from constant pool)
                000003  invokestatic    0025    (Invoke a class (static) method)
                            ##### Call [ sleep ]#####
                000006  aload_0                 (Load object reference from local variable)
                000007  invokevirtual   0012    (Invoke instance method)
                            ##### Call [ input_check ]#####
                00000A  aload_0                 (Load object reference from local variable)
                00000B  getfield        002E    (Fetch field from object)
                            ##### Get Field [ flag ]#####
                00000E  ifeq    FFF2            (Branch if equal to 0)
                000011  aload_0                 (Load object reference from local variable)
                000012  invokevirtual   0020    (Invoke instance method)
                            ##### Call [ midi_transmit ]#####
                000015  aload_0                 (Load object reference from local variable)
                000016  invokevirtual   0021    (Invoke instance method)
                            ##### Call [ repaint ]#####
                000019  goto    FFE7           (Branch)
                00001C  pop                     (Pop top stack word)
                00001D  return                  (Return (void) from procedure)

        exception_table_length = 1

                000638  start_pc = 0 , end_pc = 28 , handler_pc = 28 , catch_type = java/lang/Exception

        attributes_count = 1

                000642  type = LineNumberTable , attribute_length = 38

                        00 09 00 00 00 12 00 00 00 14 00 06 00 15 00 0A 00 16 00 11
                        00 17 00 15 00 18 00 19 00 13 00 1C 00 1C 00 1D 00 11

00066C   method[2] : ACC_PUBLIC , Name = start , Signature = ()V , Attribute = 1

        type = Code , length = 52 , max_stack = 4 , max_locals = 1 , code_length = 20

                000000  aload_0                 (Load object reference from local variable)
                000001  new     0007            (Create new object)
                            ##### Define [ java/lang/Thread ]#####
                000004  dup                     (Duplicate top stack word)
                000005  aload_0                 (Load object reference from local variable)
                000006  invokenonvirt   0010    (Invoke instance method, dispatching based on compile-time type)
                            ##### Call [ <init> ]#####
                000009  putfield        0027    (Set field in object)
                            ##### Put Field [ flow ]#####
                00000C  aload_0                 (Load object reference from local variable)
                00000D  getfield        0027    (Fetch field from object)
                            ##### Get Field [ flow ]#####
                000010  invokevirtual   0017    (Invoke instance method)
                            ##### Call [ start ]#####
                000013  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00069C  type = LineNumberTable , attribute_length = 14

                        00 03 00 00 00 20 00 0C 00 21 00 13 00 1F

0006AE   method[3] : ACC_PUBLIC , Name = stop , Signature = ()V , Attribute = 1

        type = Code , length = 36 , max_stack = 1 , max_locals = 1 , code_length = 8

                000000  aload_0                 (Load object reference from local variable)
                000001  getfield        0027    (Fetch field from object)
                            ##### Get Field [ flow ]#####
                000004  invokevirtual   000F    (Invoke instance method)
                            ##### Call [ stop ]#####
                000007  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                0006D2  type = LineNumberTable , attribute_length = 10

                        00 02 00 00 00 25 00 07 00 24

0006E0   method[4] : ACC_PUBLIC , Name = mouseDrag , Signature = (Ljava/awt/Event;II)Z , Attribute = 1

        type = Code , length = 36 , max_stack = 3 , max_locals = 4 , code_length = 8

                000000  aload_0                 (Load object reference from local variable)
                000001  iload_2                 (Load integer from local variable)
                000002  iload_3                 (Load integer from local variable)
                000003  invokevirtual   0023    (Invoke instance method)
                            ##### Call [ event_check ]#####
                000006  iconst_1                (Push integer constant)
                000007  ireturn                 (Return integer from function)

        exception_table_length = 0
```

〈リスト9-3〉 rejava2によってmiditx.classを解析したもの ④

```
        attributes_count = 1

                000704  type = LineNumberTable , attribute_length = 10

                        00 02 00 00 00 29 00 06 00 2A

000712   method[5] : ACC_PUBLIC , Name = mouseDown , Signature = (Ljava/awt/Event;II)Z , Attribute = 1

        type = Code , length = 36 , max_stack = 3 , max_locals = 4 , code_length = 8

                000000  aload_0                 (Load object reference from local variable)
                000001  iload_2                 (Load integer from local variable)
                000002  iload_3                 (Load integer from local variable)
                000003  invokevirtual   0023    (Invoke instance method)
                            ##### Call [ event_check ]#####
                000006  iconst_1                (Push integer constant)
                000007  ireturn                 (Return integer from function)

        exception_table_length = 0

        attributes_count = 1

                000736  type = LineNumberTable , attribute_length = 10

                        00 02 00 00 00 2E 00 06 00 2F

000744   method[6] : ( no flag ) , Name = event_check , Signature = (II)V , Attribute = 1

        type = Code , length = 137 , max_stack = 4 , max_locals = 5 , code_length = 89

                000000  iconst_0                (Push integer constant)
                000001  istore  04              (Store integer into local variable)
                000003  goto    004C            (Branch)
                000006  bipush  28              (Push one-byte signed integer)
                000008  bipush  28              (Push one-byte signed integer)
                00000A  iload   04              (Load integer from local variable)
                00000C  imul                    (Integer multiply)
                00000D  iadd                    (Integer add)
                00000E  istore_3                (Store integer into local variable)
                00000F  iload_1                 (Load integer from local variable)
                000010  iload_3                 (Load integer from local variable)
                000011  bipush  0A              (Push one-byte signed integer)
                000013  isub                    (Integer subtract)
                000014  if_icmple       0038    (Branch if integer less than or equal to)
                000017  iload_3                 (Load integer from local variable)
                000018  bipush  0A              (Push one-byte signed integer)
                00001A  iadd                    (Integer add)
                00001B  iload_1                 (Load integer from local variable)
                00001C  if_icmple       0030    (Branch if integer less than or equal to)
                00001F  iload_2                 (Load integer from local variable)
                000020  bipush  09              (Push one-byte signed integer)
                000022  if_icmple       002A    (Branch if integer less than or equal to)
                000025  iload_2                 (Load integer from local variable)
                000026  sipush  00BF            (Push two-byte signed integer)
                000029  if_icmpge       0023    (Branch if integer greater than or equal to)
                00002C  aload_0                 (Load object reference from local variable)
                00002D  getfield        001A    (Fetch field from object)
                            ##### Get Field [ oy ]#####
                000030  iload   04              (Load integer from local variable)
                000032  aload_0                 (Load object reference from local variable)
                000033  getfield        001E    (Fetch field from object)
                            ##### Get Field [ py ]#####
                000036  iload   04              (Load integer from local variable)
                000038  iaload                   (Load integer from array)
                000039  iastore                 (Store into integer array)
                00003A  aload_0                 (Load object reference from local variable)
                00003B  getfield        001E    (Fetch field from object)
                            ##### Get Field [ py ]#####
                00003E  iload   04              (Load integer from local variable)
                000040  iload_2                 (Load integer from local variable)
                000041  iastore                 (Store into integer array)
                000042  aload_0                 (Load object reference from local variable)
                000043  dup                     (Duplicate top stack word)
                000044  getfield        002E    (Fetch field from object)
                            ##### Get Field [ flag ]#####
                000047  iconst_1                (Push integer constant)
                000048  iadd                    (Integer add)
                000049  putfield        002E    (Set field in object)
                            ##### Put Field [ flag ]#####
                00004C  iinc    04 01           (Increment local variable by constant)
                00004F  iload   04              (Load integer from local variable)
                000051  aload_0                 (Load object reference from local variable)
                000052  getfield        002D    (Fetch field from object)
                            ##### Get Field [ channel ]#####
                000055  if_icmplt       FFB1    (Branch if integer less than)
                000058  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                0007B9  type = LineNumberTable , attribute_length = 30

                        00 07 00 00 00 34 00 06 00 35 00 0F 00 36 00 2C 00 37 00 42
                        00 38 00 4C 00 34 00 58 00 32

0007DB   method[7] : ( no flag ) , Name = input_check , Signature = ()V , Attribute = 1

        type = Code , length = 25 , max_stack = 0 , max_locals = 1 , code_length = 1
```


ネットワーク対応のシステム　RS232Cなどの通信ネットワークを利用して複数で分散処理を行うようなシステムという意味.

〈リスト9-3〉 rejava2によってmiditx.classを解析したもの ⑤

```
                000000  return          (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                0007F8  type = LineNumberTable , attribute_length = 6

                        00 01 00 00 00 3D

000802   method[8] : ( no flag ) , Name = midi_transmit , Signature = ()V , Attribute = 1

        type = Code , length = 25 , max_stack = 0 , max_locals = 1 , code_length = 1

                000000  return          (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                00081F  type = LineNumberTable , attribute_length = 6

                        00 01 00 00 00 41

000829   method[9] : ACC_PUBLIC , Name = paint , Signature = (Ljava/awt/Graphics;)V , Attribute = 1

        type = Code , length = 320 , max_stack = 6 , max_locals = 5 , code_length = 228

                000000  aload_0                 (Load object reference from local variable)
                000001  iconst_0                (Push integer constant)
                000002  putfield        002E    (Set field in object)
                            ##### Put Field [ flag ]#####
                000005  iconst_0                (Push integer constant)
                000006  istore_3                (Store integer into local variable)
                000007  goto    00D4            (Branch)
                00000A  bipush  28              (Push one-byte signed integer)
                00000C  bipush  28              (Push one-byte signed integer)
                00000E  iload_3                 (Load integer from local variable)
                00000F  imul                    (Integer multiply)
                000010  iadd                    (Integer add)
                000011  istore_2                (Store integer into local variable)
                000012  aload_1                 (Load object reference from local variable)
                000013  getstatic       002B    (Get static field from class)
                            ##### Get Static [ black ]#####
                000016  invokevirtual   001C    (Invoke instance method)
                            ##### Call [ setColor ]#####
                000019  aload_1                 (Load object reference from local variable)
                00001A  iload_2                 (Load integer from local variable)
                00001B  bipush  0A              (Push one-byte signed integer)
                00001D  isub                    (Integer subtract)
                00001E  aload_0                 (Load object reference from local variable)
                00001F  getfield        001A    (Fetch field from object)
                            ##### Get Field [ oy ]#####
                000022  iload_3                 (Load integer from local variable)
                000023  iaload                   (Load integer from array)
                000024  iconst_5                (Push integer constant)
                000025  isub                    (Integer subtract)
                000026  bipush  14              (Push one-byte signed integer)
                000028  bipush  0A              (Push one-byte signed integer)
                00002A  invokevirtual   001B    (Invoke instance method)
                            ##### Call [ fillRect ]#####
                00002D  aload_1                 (Load object reference from local variable)
                00002E  getstatic       001F    (Get static field from class)
                            ##### Get Static [ yellow ]#####
                000031  invokevirtual   001C    (Invoke instance method)
                            ##### Call [ setColor ]#####
                000034  aload_1                 (Load object reference from local variable)
                000035  iload_2                 (Load integer from local variable)
                000036  iconst_1                (Push integer constant)
                000037  isub                    (Integer subtract)
                000038  bipush  0A              (Push one-byte signed integer)
                00003A  iconst_2                (Push integer constant)
                00003B  sipush  00B4            (Push two-byte signed integer)
                00003E  invokevirtual   001B    (Invoke instance method)
                            ##### Call [ fillRect ]#####
                000041  aload_1                 (Load object reference from local variable)
                000042  getstatic       001D    (Get static field from class)
                            ##### Get Static [ green ]#####
                000045  invokevirtual   001C    (Invoke instance method)
                            ##### Call [ setColor ]#####
                000048  aload_1                 (Load object reference from local variable)
                000049  iload_2                 (Load integer from local variable)
                00004A  bipush  0A              (Push one-byte signed integer)
                00004C  isub                    (Integer subtract)
                00004D  aload_0                 (Load object reference from local variable)
                00004E  getfield        001E    (Fetch field from object)
                            ##### Get Field [ py ]#####
                000051  iload_3                 (Load integer from local variable)
                000052  iaload                   (Load integer from array)
                000053  iconst_5                (Push integer constant)
                000054  isub                    (Integer subtract)
                000055  bipush  14              (Push one-byte signed integer)
                000057  bipush  0A              (Push one-byte signed integer)
                000059  invokevirtual   001B    (Invoke instance method)
                            ##### Call [ fillRect ]#####
                00005C  aload_1                 (Load object reference from local variable)
                00005D  getstatic       0019    (Get static field from class)
                            ##### Get Static [ blue ]#####
                000060  invokevirtual   001C    (Invoke instance method)
                            ##### Call [ setColor ]#####
                000063  aload_1                 (Load object reference from local variable)
```


PHS　PHSのデータ通信によるインターネット利用のスピードは，ようやく28.8kbpsから32kbpsあたりになった．MIDIは有線ではあるが，最初から31.25kbpsである．

〈リスト9-3〉 rejava2によってmiditx.classを解析したもの ⑥

```
                000064  iload_2                 (Load integer from local variable)
                000065  bipush  08              (Push one-byte signed integer)
                000067  isub                    (Integer subtract)
                000068  aload_0                 (Load object reference from local variable)
                000069  getfield        001E    (Fetch field from object)
                        ##### Get Field [ py ]#####
                00006C  iload_3                 (Load integer from local variable)
                00006D  iaload                  (Load integer from array)
                00006E  iconst_3                (Push integer constant)
                00006F  isub                    (Integer subtract)
                000070  bipush  10              (Push one-byte signed integer)
                000072  bipush  06              (Push one-byte signed integer)
                000074  invokevirtual   001B    (Invoke instance method)
                        ##### Call [ fillRect ]#####
                000077  aload_0                 (Load object reference from local variable)
                000078  getfield        0013    (Fetch field from object)
                        ##### Get Field [ data ]#####
                00007B  iload_3                 (Load integer from local variable)
                00007C  sipush  00BE            (Push two-byte signed integer)
                00007F  aload_0                 (Load object reference from local variable)
                000080  getfield        001E    (Fetch field from object)
                        ##### Get Field [ py ]#####
                000083  iload_3                 (Load integer from local variable)
                000084  iaload                  (Load integer from array)
                000085  isub                    (Integer subtract)
                000086  bipush  7F              (Push one-byte signed integer)
                000088  imul                    (Integer multiply)
                000089  sipush  00B4            (Push two-byte signed integer)
                00008C  idiv                    (Integer divide)
                00008D  iastore                 (Store into integer array)
                00008E  aload_1                 (Load object reference from local variable)
                00008F  getstatic       0011    (Get static field from class)
                        ##### Get Static [ white ]#####
                000092  invokevirtual   001C    (Invoke instance method)
                        ##### Call [ setColor ]#####
                000095  aload_1                 (Load object reference from local variable)
                000096  aload_0                 (Load object reference from local variable)
                000097  getfield        0028    (Fetch field from object)
                        ##### Get Field [ font ]#####
                00009A  invokevirtual   0018    (Invoke instance method)
                        ##### Call [ setFont ]#####
                00009D  new     000B            (Create new object)
                        ##### Define [ java/lang/StringBuffer ]#####
                0000A0  dup                     (Duplicate top stack word)
                0000A1  invokenonvirt   0016    (Invoke instance method, dispatching based on compile-time type)
                        ##### Call [ <init> ]#####
                0000A4  ldc1    02              (Push item from constant pool)
                        ##### Push Item [ Data( ]#####
                0000A6  invokevirtual   0026    (Invoke instance method)
                        ##### Call [ append ]#####
                0000A9  iload_3                 (Load integer from local variable)
                0000AA  iconst_1                (Push integer constant)
                0000AB  iadd                    (Integer add)
                0000AC  invokestatic    002C    (Invoke a class (static) method)
                        ##### Call [ valueOf ]#####
                0000AF  invokevirtual   0026    (Invoke instance method)
                        ##### Call [ append ]#####
                0000B2  ldc1    01              (Push item from constant pool)
                        ##### Push Item [ ) =  ]#####
                0000B4  invokevirtual   0026    (Invoke instance method)
                        ##### Call [ append ]#####
                0000B7  aload_0                 (Load object reference from local variable)
                0000B8  getfield        0013    (Fetch field from object)
                        ##### Get Field [ data ]#####
                0000BB  iload_3                 (Load integer from local variable)
                0000BC  iaload                  (Load integer from array)
                0000BD  invokestatic    002C    (Invoke a class (static) method)
                        ##### Call [ valueOf ]#####
                0000C0  invokevirtual   0026    (Invoke instance method)
                        ##### Call [ append ]#####
                0000C3  invokevirtual   0024    (Invoke instance method)
                        ##### Call [ toString ]#####
                0000C6  astore  04              (Store object reference into local variable)
                0000C8  aload_1                 (Load object reference from local variable)
                0000C9  aload   04              (Load object reference from local variable)
                0000CB  sipush  0163            (Push two-byte signed integer)
                0000CE  bipush  1E              (Push one-byte signed integer)
                0000D0  bipush  16              (Push one-byte signed integer)
                0000D2  iload_3                 (Load integer from local variable)
                0000D3  imul                    (Integer multiply)
                0000D4  iadd                    (Integer add)
                0000D5  invokevirtual   0015    (Invoke instance method)
                        ##### Call [ drawString ]#####
                0000D8  iinc    03 01           (Increment local variable by constant)
                0000DB  iload_3                 (Load integer from local variable)
                0000DC  aload_0                 (Load object reference from local variable)
                0000DD  getfield        002D    (Fetch field from object)
                        ##### Get Field [ channel ]#####
                0000E0  if_icmplt       FF2A    (Branch if integer less than)
                0000E3  return                  (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                000929  type = LineNumberTable , attribute_length = 74

                        00 12 00 00 00 47 00 05 00 48 00 0A 00 49 00 12 00 4A 00 19
                        00 4B 00 2D 00 4C 00 34 00 4D 00 41 00 4E 00 48 00 4F 00 5C
                        00 50 00 63 00 51 00 77 00 52 00 8E 00 53 00 95 00 54 00 9D
                        00 55 00 C8 00 56 00 D8 00 48 00 E3 00 45
```

〈リスト9-3〉 rejava2によってmiditx.classを解析したもの ⑦

```
000977   method[10] : ( no flag ) , Name = system_initial , Signature = ()V , Attribute = 1

        type = Code , length = 128 , max_stack = 6 , max_locals = 2 , code_length = 84

                000000  aload_0                  (Load object reference from local variable)
                000001  getstatic       002B     (Get static field from class)
                             ##### Get Static [ black ]#####
                000004  invokevirtual   0022     (Invoke instance method)
                             ##### Call [ setBackground ]#####
                000007  aload_0                  (Load object reference from local variable)
                000008  new     0005             (Create new object)
                             ##### Define [ java/awt/Font ]#####
                00000B  dup                      (Duplicate top stack word)
                00000C  ldc1    03               (Push item from constant pool)
                             ##### Push Item [ TimesRoman ]#####
                00000E  iconst_0                 (Push integer constant)
                00000F  bipush  10               (Push one-byte signed integer)
                000011  invokenonvirt   0014     (Invoke instance method, dispatching based on compile-time type)
                             ##### Call [ <init> ]#####
                000014  putfield        0028     (Set field in object)
                             ##### Put Field [ font ]#####
                000017  aload_0                  (Load object reference from local variable)
                000018  aload_0                  (Load object reference from local variable)
                000019  getfield        002D     (Fetch field from object)
                             ##### Get Field [ channel ]#####
                00001C  newarray        0A       (Allocate new array) : array type = T_INT
                00001E  putfield        001E     (Set field in object)
                             ##### Put Field [ py ]#####
                000021  aload_0                  (Load object reference from local variable)
                000022  aload_0                  (Load object reference from local variable)
                000023  getfield        002D     (Fetch field from object)
                             ##### Get Field [ channel ]#####
                000026  newarray        0A       (Allocate new array) : array type = T_INT
                000028  putfield        001A     (Set field in object)
                             ##### Put Field [ oy ]#####
                00002B  aload_0                  (Load object reference from local variable)
                00002C  aload_0                  (Load object reference from local variable)
                00002D  getfield        002D     (Fetch field from object)
                             ##### Get Field [ channel ]#####
                000030  newarray        0A       (Allocate new array) : array type = T_INT
                000032  putfield        0013     (Set field in object)
                             ##### Put Field [ data ]#####
                000035  iconst_0                 (Push integer constant)
                000036  istore_1                 (Store integer into local variable)
                000037  goto    000F             (Branch)
                00003A  aload_0                  (Load object reference from local variable)
                00003B  getfield        001E     (Fetch field from object)
                             ##### Get Field [ py ]#####
                00003E  iload_1                  (Load integer from local variable)
                00003F  sipush  00BE             (Push two-byte signed integer)
                000042  iastore                 (Store into integer array)
                000043  iinc    01 01           (Increment local variable by constant)
                000046  iload_1                  (Load integer from local variable)
                000047  aload_0                  (Load object reference from local variable)
                000048  getfield        002D     (Fetch field from object)
                             ##### Get Field [ channel ]#####
                00004B  if_icmplt       FFEF     (Branch if integer less than)
                00004E  aload_0                  (Load object reference from local variable)
                00004F  iconst_0                 (Push integer constant)
                000050  putfield        002E     (Set field in object)
                             ##### Put Field [ flag ]#####
                000053  return                   (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                0009E7  type = LineNumberTable , attribute_length = 26

                        00 06 00 00 00 5B 00 07 00 5C 00 17 00 5D 00 35 00 5E 00 4E
                        00 5F 00 53 00 5A

000A05   method[11] : ACC_PUBLIC , Name = <init> , Signature = ()V , Attribute = 1

        type = Code , length = 29 , max_stack = 1 , max_locals = 1 , code_length = 5

                000000  aload_0                  (Load object reference from local variable)
                000001  invokenonvirt   0029     (Invoke instance method, dispatching based on compile-time type)
                             ##### Call [ <init> ]#####
                000004  return                   (Return (void) from procedure)

        exception_table_length = 0

        attributes_count = 1

                000A26  type = LineNumberTable , attribute_length = 6

                        00 01 00 00 00 06

000A30  Attributes Count = 1

        000A32  attribute[0] : type = SourceFile , length = 2

                00 5A

        Source File Name = miditx.java

..... (^_^) Java analyze completely finished. (^_^)
```

〈リスト9-6〉 C言語の形態となるように関数名の出力フォーマットを加工する

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

FILE *fp;
int ni[1000], m[150000], p[300][10], d_mode;
char n[300][60], tail[3];

static void access_flag_printf(int para);
static void byte_code_printf(long address);
static void display_information(int mode, int id);

int main(int argc,char **argv){
        int d, i, j, k, ct;
        unsigned int p0, p00, p1, p2, p3;
        int pp=0, count=0, p4, p5, sb;
        unsigned char ss, st[20], d0, d1, d2, d3;
        st[16]=0; tail[0]=59; tail[1]=0;
        d_mode=1;

        if( argc != 2 ){ printf("... target file name missing (;_;) ...¥n¥n"); exit(1); }
        else if( (fp=fopen( argv[1], "rb" ))==NULL ){ printf("... target file [ %s ] is not found (;_;) ...¥n¥n", argv[1]); exit(1); }
        if(d_mode==0) printf("Target Java File Name = %s¥n", argv[1]);
        else if(d_mode==1) printf("/* Source Java File Name = %s */¥n¥nmain(){¥n¥tinit()%s¥n¥twhile(1){¥n¥t¥trun()%s¥n¥t)¥n}", argv[1], tail, tail);
        while( (d = fgetc(fp)) >= 0 ){
                m[count++] = d & 0xff;
        }
        fclose(fp);
        if(d_mode==0) printf("¥nTotal File Length = %d bytes.", count);
        p0 = m[pp+3]+256*m[pp+2];
        p00 = m[pp+1]+256*m[pp];
        if(d_mode==0) printf("¥n¥n¥t%06X  Magic Code [CAFEBABE] = %04X%04X",pp,p00,p0); pp += 4;
        if(( p00 != 0xcafe )||( p0 != 0xbabe )){
                if(d_mode==0) printf("¥n¥n... target file is not Java (;_;) ...¥;¥n"); exit(1);
        }
        else if(d_mode==0) printf("  ---  OK (^_^)");
        if(d_mode==0) printf("¥n¥n¥t%06X  Version : major version %d , minor version %d", pp, m[pp+3]+256*m[pp+2], m[pp+1]+256*m[pp] ); pp += 4;
        p1 = m[pp+1]+256*m[pp]; if(d_mode==0) printf("¥n¥n¥t%06X  Constant Pool : total number = %d\n", pp, p1 ); pp += 2;
/***** 中略 *****/
        p1 = m[pp+1]+256*m[pp]; pp += 2; if(d_mode==0) printf("¥n¥n¥t%06X  Methods Count = %d", pp-2, p1 );
        for(i=0;i<p1;i++){
                if(d_mode==0) printf("¥n¥n¥t%06X   method[%d] : ", pp, i);
                p2 = m[pp+1]+256*m[pp]; pp += 2; access_flag_printf( p2 );
                p2 = m[pp+1]+256*m[pp]; pp += 2;
        if((d_mode==1)&&( n[p2][0]=='<' )){
                printf("¥n¥n/**);
        }
        if(d_mode==1) printf("¥n¥n%s(){", n[p2] );
        else if(d_mode==0) printf(" , Name = %s", n[p2] );
        if((d_mode==1)&&( strcmp( n[p2], "init" ) != 0 )&&( strcmp( n[p2], "run" ) != 0 )&&( n[p2][0]!='<' )){
                printf("¥n/**);
        }
        sb = p2;
                p2 = m[pp+1]+256*m[pp]; pp += 2; if(d_mode==0) printf(" , Signature = %s", n[p2] );
                p2 = m[pp+1]+256*m[pp]; pp += 2; if(d_mode==0) printf(" , Attribute = %d", p2 );
                if( p2 != 0 ){
                        if(d_mode==0) printf("¥n");
                        for(j=0;j<p2;j++){
                                p3 = m[pp+1]+256*m[pp]; pp += 2;
                                if(d_mode==0) printf("¥n¥t¥t¥ttype = %s , ", n[p3] );
                                p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                                if(d_mode==0) printf("length = %d", p4);
                                p5 = pp; pp += p4;
                                if( strcmp( n[p3], "Code" ) == 0 ) byte_code_printf( p5 );
                                else if( strcmp( n[p3], "Exceptions" ) == 0 ) byte_code_printf( p5 );
                        }
                }
        if((d_mode==1)&&( strcmp( n[sb], "init" ) != 0 )&&( strcmp( n[sb], "run" ) != 0 )&&( n[sb][0]!='<' )){
                printf("¥n*/");
        }
        if(d_mode==1) printf("¥n}");
        if((d_mode==1)&&( n[sb][0]=='<' )){
                printf("¥n¥n*/");
        }
        }
        p1 = m[pp+1]+256*m[pp]; pp += 2; if(d_mode==0) printf("¥n¥n¥t%06X  Attributes Count = %d", pp-2, p1 );
        if( p1 != 0 ){
                for(i=0;i<p1;i++){
                        p2 = m[pp+1]+256*m[pp]; pp += 2;
                        if(d_mode==0) printf("¥n¥n¥t¥t%06X  attribute[%d] : type = %s , ", pp-2, i, n[p2] );
                        p4 = m[pp+3]+256*(m[pp+2]+256*(m[pp+1]+256*m[pp])); pp += 4;
                        p5 = pp;
                        if(d_mode==0) printf("length = %d¥n", p4);
                        for(k=0;k<p4;k++){
                                if((k%25)==0) if(d_mode==0) printf("¥n¥t¥t¥t¥t");
                                if(d_mode==0) printf("%02X ", m[pp++] );
                        }
                        if( ( p4 == 2 ) && ( strcmp( n[p2], "SourceFile" ) == 0 ) ){
                                p3 = m[p5+1]+256*m[p5];
                                if(d_mode==0) printf("¥n¥n¥t¥t¥tSource File Name = %s", n[p3] );
                        }
                }
        }
        if(d_mode==0) printf("¥n¥n..... (^_^) Java analyze completely finished. (^_^)¥n¥n");            return;
}
/***** 後略 *****/
```

〈リスト9-10〉　サービス・ライブラリ java_lib.c

```
#define         ctc_0           0x10
#define         ctc_1           0x11
#define         ctc_2           0x12
#define         ctc_3           0x13
#define         sio_a           0x18
#define         sio_b           0x1a
#define         pio_a           0x1c
#define         pio_b           0x1e

#define         rx_fifo         0x8000  /* MIDI Receive */
#define         tx_fifo         0x8800  /* MIDI Transmit */
#define         lcd_fifo        0x9000  /* LCD Display */
#define         rx_top          0x9100 + 2 * 0
#define         rx_end          0x9100 + 2 * 1
#define         tx_top          0x9100 + 2 * 2
#define         tx_end          0x9100 + 2 * 3
#define         timer_flag      0x9100 + 2 * 4
#define         lcd_top         0x9100 + 2 * 5
#define         lcd_end         0x9100 + 2 * 6
#define         lcd             0x9100 + 2 * 7
#define         work            0x9200  /* unversal use */

int     i,j,k,l,m,n;                    /* universal for local use */
int     midi,rsb,dcb,channel,keyno; /* universal MIDI parameter */

/*****  Common Functions  *********************************/

void port_out( int port, int data ){
#asm
        ld      hl,4
        add     hl,sp
        ld      c,(hl)
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        out     (c),a
        nop
        ret
#endasm
}

int port_in( int port ){
#asm
        ld      hl,2
        add     hl,sp
        ld      c,(hl)
        in      a,(c)
        ld      l,a
        ld      h,0
        ret
#endasm
}

void ram_put( int address, int data ){
#asm
        ld      hl,4
        add     hl,sp
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
        ld      hl,2
        add     hl,sp
        ld      a,(hl)
        ld      (bc),a
#endasm
}

int ram_get( int address ){
#asm
        ld      hl,2
        add     hl,sp
        ld      c,(hl)
        inc     hl
        ld      b,(hl)
        ld      a,(bc)
        ld      l,a
        ld      h,0
        ret
#endasm
}

void enable_interrupt(){
#asm
        ei
#endasm
}

void disable_interrupt(){
#asm
        di
#endasm
}

/*****  I/O Setting Functions  ****************************/

void pio_a_setting( int bitmap ){
        port_out( pio_a+1, 0xcf );              /* Mode 3 */
        port_out( pio_a+1, bitmap );
}

void pio_b_setting( int bitmap ){
        port_out( pio_b+1, 0xcf );              /* Mode 3 */
        port_out( pio_b+1, bitmap );
}

void timer_0_setting(){
        port_out( ctc_0, 0x70 );        /* Interrupt Address */
        port_out( ctc_0, 0xa5 );        /* Timer Mode */
        port_out( ctc_0, 158);          /* about 10msec */
}

void sio_dummy_read(){
#asm
        in      a,(sio_a+0)
#endasm
}

void midi_port_setting(){
        port_out( sio_b+1, 0x18 );
        port_out( sio_b+1, 0x04 );
        port_out( sio_b+1, 0xc4 );
        port_out( sio_b+1, 0x01 );
        port_out( sio_b+1, 0x00 );
        port_out( sio_b+1, 0x05 );
        port_out( sio_b+1, 0x60 );
        port_out( sio_b+1, 0x02 );
        port_out( sio_b+1, 0x20 );
        port_out( sio_a+1, 0x18 );
        port_out( sio_a+1, 0x04 );
        port_out( sio_a+1, 0xc4 );
        port_out( sio_a+1, 0x01 );
        port_out( sio_a+1, 0x10 );
        port_out( sio_a+1, 0x05 );
        port_out( sio_a+1, 0x68 );
        port_out( sio_a+1, 0x03 );
        port_out( sio_a+1, 0xc1 );
}

/*****  MIDI Functions  ***********************************/

void tx_midi_check(){
#asm
        ld      de,(tx_end)
        ld      hl,(tx_top)
        and     a
        sbc     hl,de
        ret     z
        xor     a
        out     (sio_a+1),a
        in      a,(sio_a+1)
        bit     2,a
        ret     z
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      a,(hl)
        out     (sio_a),a
        inc     de
        res     3,d
        ld      (tx_end),de
        ret
#endasm
}

void tx_midi_set(int data){
#asm
        ld      hl,2
        add     hl,sp
        ld      b,(hl)
        ld      de,(tx_top)
        ld      a,10001000b
        or      d
        ld      h,a
        ld      l,e
        ld      (hl),b
        inc     de
        res     3,d
        ld      (tx_top),de
        ret
#endasm
}
```

索　引

【マ　行】

【ラ　行】

【ワ　行】

【欧　文】

【記号，数字】

著者紹介

長嶋洋一（ながしま・よういち）

1958年茨城県生まれ．京都大学理学部卒業．河合楽器を経て，
1991年ASL(Art & Science Laboratory)長嶋技術士事務所を開設，
1993年独立開業．技術士(科学技術庁登録の技術コンサルタント)
として企業の経営／技術指導や人材育成業務に従事するとともに，
音楽情報科学(Computer Music)の研究者・作曲家としても活躍中．
イメージ情報科学研究所研究員，静岡大学・京都芸術短期大学・
神戸山手女子短期大学・国立長野高専講師．IEEE・ACM・情報処理学会・
人工知能学会・音楽知覚認知学会・ICMA・JACOM・日本技術士会等
各会員，Nifty-Serve SubSysOp(FMIDI)，技術士(情報工学・電気電子)．

E-mail nagasm@kobe-yamate.ac.jp
http://www.kobe-yamate.ac.jp/~nagasm
Nifty-Serve NBD03033

著書
『マイコン技術者スキルアップ事典』，ＣＱ出版(株)
『プロ電子技術者のコモンセンス』，ＣＱ出版(株)
『はじめて学ぶノイズ対策』，工業調査会
『はじめて学ぶ情報セキュリティ』，工業調査会
『明解・電子回路とノイズ対策』，工業調査会
『Java活用情報発信マニュアル』，新技術開発センター
『研究技術開発成果促進マニュアル』(共著)，アーバンプロデュース
『技術コンサルタント入門』(共著)，日本能率協会
『新製品開発がわかる事典』(共著)，時潮社

３.５インチＦＤ
１枚付き

トランジスタ技術SPECIAL 増刊

パソコンでつくるマイコン組み込みMIDI関連機器の製作

Java & AKI-80

定価は表四に表示してあります

1997年8月1日　発行

編集人　山　形　孝　雄
発行人　蒲　生　良　治
発行所　ＣＱ出版株式会社
〒170　東京都豊島区巣鴨1-14-2
☎ 販売部 03-5395-2141
☎ 編集部 03-5395-2121
振替 00100-7-10665

DTP・印刷・製本/三晃印刷㈱
乱丁，落丁本はお取り替えします

Printed in Japan

ハードウェア・デザイン・シリーズ　好評発売中

3端子/チョッパ/フライバック各種レギュレータICの使い方

電源用IC活用マニュアル

トランジスタ技術編集部 編
B5判　160頁
定価1,682円

本書は各種電源用ICのデータの要点と活用方法を網羅しています．
各メーカーのデバイスの中から汎用の電源ICを精選し，タイプ別に電源の設計法や回路例，実測データなどポイントをおさえてわかりやすく解説をしています．
いまや電源システムは，エレクトロニクスの世界では必須であり，その心臓部とも言える役割りをこのICが担っています．技術進歩に伴い，電源ICも高効率で優れたものが開発され，さらに用途別に多種多様の製品が世に出まわっています．
電源システムを構築するための必携マニュアルとして本書をお勧めします．

抵抗，コンデンサ，インダクタ，機構部品の特徴と仕様

わかる電子部品の基礎と活用法

薊　利明／竹田俊夫 著
B5判　184頁
定価1,733円

本書では抵抗，コンデンサ，インダクタ，機構部品の種類とその構造，仕様，特徴をイラストを豊富に使ってわかりやすく解説しています．それに加え，部品の故障率や故障モードなど高信頼設計のための基礎データなどもまとめてみました．ハードウェア・エンジニアには必読の書です．

計測制御の信号処理からセンサ/通信インターフェースまで

モジュール化に役立つ実用電子回路集

トランジスタ技術編集部 編
B5判　160頁
定価1,631円

本書では，あらゆる場面で役立つ，モジュール設計のための回路として，汎用部品でコンパクトに構成した粋な回路を集めました．また設計した回路をより実用的なものにするために，モジュール化設計した回路同士やパソコン，測定器との接続などに役立つ，便利なインターフェース回路も豊富に紹介しています．

DOS/Vマシンのインターフェースを拡張するハードウェア設計

IBM PCとISAバスの活用法

トランジスタ技術編集部 編
B5判　164頁
定価1,835円

本書ではIBM PC/AT互換機の標準入出力インターフェースの仕様をまとめたあと，ISAバスのハードウェアについて詳細に解説しています．さらに，16550Aを使用した拡張シリアル・ポート・アダプタ，高速FIFOを使用したファンクション・ジェネレータ・ボードなど，IBM PC/AT互換機用のISA拡張アダプタ・カードの設計・製作事例を具体的な回路図とサンプル・プログラムを示しながら解説しています．

新つくるシリーズ　●B5判●160頁●各定価1,529円●

No.1　〈好評発売中〉
つくるツール&測定器
おもな内容●ディジタル電圧計／ファンクション・ジェネレータ／カーブ・トレーサ／LCメータ／etc.

No.2　〈好評発売中〉
つくるオーディオ&ビデオ
おもな内容●オーディオ・アンプ／サウンド・プロセッサ／ビデオ・セレクタ／ビデオ・エフェクタ／etc.

No.3　〈好評発売中〉
つくるオリジナル・グッズ
おもな内容●電子ゲーム／キッチン・タイマ／電子温度計／電磁波時計／ニカド電池充電器／紫外線メータ／etc.

CQ出版社　〒170 東京都豊島区巣鴨1-14-2 販売部 ☎(03)5395-2141 振替 00100-7-10665

定価は1997年4月1日現在の消費税5％を含んだ表示です．

トランジスタ技術SPECIAL増刊

CQ出版社
〒170 東京都豊島区巣鴨1-14-2
☎(03)5395-2141(販売部)

定価1,800円 本体1,714円

雑誌 16712-8
Ⓛ-11/14
T1116712081806