COMPUTER MANIA & LADIES ONLY

Lime

Limited Expression of KMTU COMPUTER club

No. 2

TABLE of Contents

LIme
巻2. 第1刊
発行.
京都工芸繊維大学
コンピューター
クラブ
1982年
11月
20日

Editting Staff
Tabe
Kazu-chan
池田 くん

Thank & Love

from the Editorial office

巻頭言に替えて (Take●)

□ 去年のLimeは、読みものばっかりだった。今年は、お勉強も入れるべえと考えたが、どれを勉強だと思ってもらえるだろう。

□ 去年のLimeに、ネットワークの話を書いておいて、今年DATA・SHOWに行ったら、そこらじゅうネットワークだらけだった。Net・Workの次に来るのは、データ・ベースだろう。Net・Workによって、あちこちからOn・Lineでデータを手に入れたり、よそからもらって来る時に、ハード的な手段はあるが、大量のデータを有効と管理し、また別なマシンへデータを送る時には2つ以上のものからAccessされても平気なデータバンクが必要になる。また、人工知能が流行っているが、人工知能もNet・Workから、いっぱい情報を仕入れて知識データベースを巨大化させる可能性がある。またまた、事ム処理でも、データを保存したり、加工したりのくり返しをもっと、データ間の関係を強力に示せれば、簡単に行なえるはずなのだ。Prologは、人工知能用の言語と思っている奴もいるだろうが、データベース用にも、石膏されているし、現にPrologのプログラムは、特定の(仕事に最低必要かつ十分な)データを打ち込んでおくと、入力された質問に答えるつもりで、(さっき打たれた)データ(一種の知識データベース)を検索しながら実行されるのである。

今、記憶媒体の価格が飛躍的に下っているので、特に大量にデータを扱う可能性が増大している。そして大量のデータを手に入れることができるし、かも

大量のデータを格納することもできる。こんな時、無秩序にデータをHandlingしようとしても無理だ。やはり、ここは、O.S.の中にデータ・ベース・マネージメントの機能をある程度組み込んだ様なマシンに出て来てもらって、データ・ベース・ハンドリングの強力な例えば、構造体等のデータの構造化を、もっと自然的な他のデータとの関連が書けるような プログラミング言語でもセットで、あれば助かる。

しかし、究極的なことを言えば、プログラムなどしなくても、仕事のやり方はデータの中にもっている。そして、仕事に必要な常識的な知識もデータの中にもっている。そして、やって欲しい仕事を言うと仕事の方法を自分で見つけ出して、自動的に起動し、その仕事にいる知識は、それにふさわしい形でデータベースから持ってくる。と、いう様な生物が良い。「それでは、まるで人工知能そのものだ」と、思う人もいるだろうが、もし、手順がキカイの中にあるなら、それを捜して実行するぐらいしてもらっても、あたり前だ。

ただし、人間でも、仕事の解き方を間違う(積分の時など)から、キカイにも、非常に難しいだろう。また、データを最適な形でもってくるのも、人間でも、発想の転換が話題になるぐらいだから、キカイが誤った意味で知識をアクセスして解答不能になっても不思議は無い。

ここらで一歩ゆずって、(Prologより一歩すすんで)オブジェクト指向言語[僕の曲解かも知れないが、大規模なデータベースから自動的に必要な形でデータを自動的にアクセスしてくれる機能(不用な属性があっても七かり落す機能)のついたプログラミング言語。]というのを、*アラン・ケイは次に来るものとして、予言している。 [1982. Nov. 16]

*アラン・ケイは、「DYNA-BOOK」の発案者、XEROXの"ALTO"以来のSuper Personalに影響を与える。最近ATARI社に移ったもよう。

人工知能課御案内

「人間の思考活動というものも、物理的に見れば大脳における電気・化学作用の連鎖に他ならない。とすれば、エレクトロニクスのがように発展した今日、この力を借りて人間の頭脳と同じように、思考する人工物をつくれないか。コンピューターこそまさにこの目的にかなうものである。」人工知能とはこのような唯物論的一大見識のもとに考えられた、単に与えられた手順どおり忠実に計算をすすめるだけではなく、人間に成りかわって真に創造的なことがらを考えだせるコンピューターのことである。コンピューターが発明されて間もない頃は、このようなことがもう10年もすれば大規模に実現されると予想した人もいたようだが、現在、人工知能を研究している人で、このような安易な見通しをたてている人はいないだろう。普通人工知能で扱われる問題は、人間の思考一般のような広範多岐にわたるものではなく、ある限られたモデルで、しかも検索方法がある程度限定された場合である。この意味でコンピュ

ーターが人間の頭脳に伍しえるまでにはまだまだ前途遼遠である。

さて当人工知能課ではどのようなことをするのかといいますと、要はコンピューターにパズルを解かせたり、ゲーム（インベーダーやギャラクシアンの類ではない）をやらせたりするのだと了解して下さい。ですから、新人部員の人でコンピューターに囲碁や将棋をやらせたいとか、あるいはエイトクイーンやペントミノをやらせたいと思う人があったら早めに人工知能課の方へ受講登録して下さい。（部長に名のり出ればよい。）このあと、部長による「コンピューターが人間とオセロゲームをするプログラム」の解説がありますが、そこを読めば、将棋などで先読みしてさすということは、どういうことなのかがわかるでしょう。（課長）

Othello by Tom (とむ) [A.I. SECTION]

このクラブでは Othello は以前から目標の1つとしてとり入れられていたが，今年からは，人工知能（Artificial Intelligence；A.I.）課の Othello 係というところがうけもつことになっている．人工知能課には他にスコア フォー係，数式処理係などを設ける予定である．

現在 Othello のプログラムはクラブ員が作成したものが数種あるが，画面への表示と，ルール違反かどうかの判断を含めた部分は，Tom が作成したものがすぐれていると思っている．従って，今年はオセロの手読みという点に集中して活動していく予定である．使用言語はできるだけいろいろなものを使えるようにしたいところであるが，現在の状況では，機械語，Game，TL/1 ぐらいになると思う．表示と，ルール違反の判断の部分は，必要に応じて Tom が変更を試みるので，使用したい言語と，手読みの方法などを Tom に教えてください．

今までにこのクラブで作成された Othello プログラムをいくつか紹介します．

- 軟弱 Othello

  1手先の自分の駒数が最大となる所へ行くアホな手読みをする．実力は初心者以下．

駒の上にさらに駒を置ける という不備があった。

- Tom Othello

  Tomが独自に考えた手読み法を実現した。しかし，性能はmin-max法とα-β法の中間ぐらいで，min-max法の変形であったので，min-max法から盗んだと言われてもしかたがない。

  最終6へ8手読み，他は4手読み。

  角を異常に高く評価し，駒数が多くなるように行く。実力は初心者と同程度。

- Koma Othello (陰険 読み)

  相手が駒を置ける位置の個数を最小にするように行く。その駒の置き方がいやらしいので陰険読みと呼ばれる。実力は初心者より少し上である。

  このOthelloを少し改良して，だんだん強くなる学習機能を持つものも作成されたが；はじめのうちはだんだん強くなって，ある程度強くなると，あとは あまり変化」しないので，失敗作であったといえる。なお，技刈りはα-β法を採用しているそうである。

System section がお贈りする

ローカルネットワークシステム

CONS

(Compact Network System)

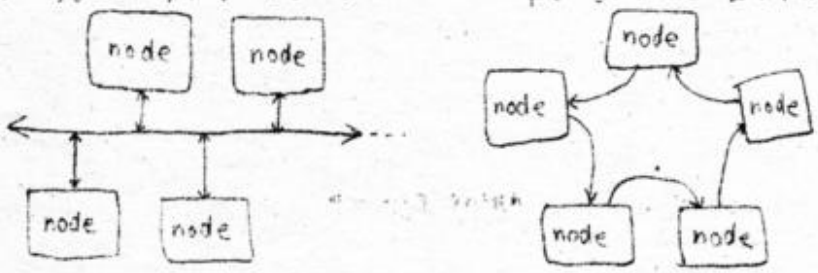
Written by MASA

なんでも、最近はネットワークが大はやりだそうで、ショーを見に行ってもネットワークが数多く展示されているそうです。

そこで我がSystem sectionは82年度のprojectとして、ローカルエリアネットワークシステムを製作することになりました。

ネットワークについては、最近多くの文献で取り上げられているため、ここで詳しく説明することは避けますが、ネットワークに関する基本的な知識は文献1等を参考にしてもらっても結構です。

ローカルネットワークシステムでは有名なものにXeroxのEthernetやZilogのZnet, OMNINET等がありますが、アマチュアレベルのものとして京大KMCのPLANET-1があります。これらはすべてバス型ネットワークという方式をとっています。これに対し、ループ型ネットワークというものがあるのですが、CONSではこの中でもループコントローラーのない



○バス型ネットワーク　　○リング・ネットワーク

リング・ネットワーク という方式を採用しています。

この2者はどちらも一長一短で、簡単に優劣を決定することはできませんが、今回採用したリング・ネットワークの特徴はほぼ次の通りです。

- データの衝突が起こらないため、簡単なプロトコルで確実なデータの通信ができる。
- トークンパケット(後述)の数により、ネットワークの転送能力を制限することができる。
- ネットワークに参加する可能性のあるステーションはすべてネットワークの起動時より作動していなくてはならない。つまりネットワークの起動後に新たに参加、脱退ということはできない。

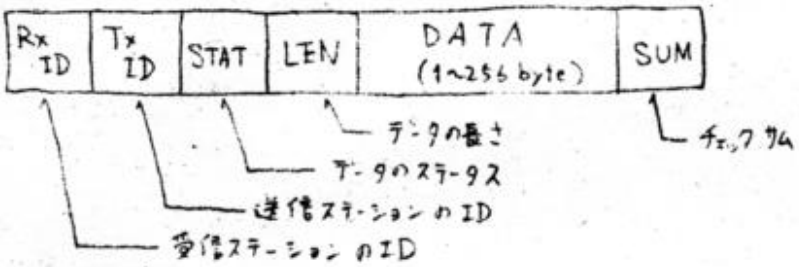
ということになります。

CONSではトークンパッシングという手法を用いてデータの通信を行なっています。これはパケット通信の一種です。パケット(Packet)とは小包という意味ですがここではひとまとまりのデータのかたまりを言います。トークンパッシングを行なうシステムではパケットはひとつの荷車のようなものと考えれば良いでしょう。普段はデータの全くのっていないパケット、つまり空パケットがネットワークのループ上を高速で巡回し、データを送信するときは、この空パケットをつかまえ、データを乗せてループ上へ流すわけです。このデータパケットには宛先や発信人等の情報も全て含まれています。受信側では時おり(実際には高速で)やってくる自分宛のデータパケットを取り込みその代りに空のパケットを返せば良いのです。

この空パケットは言いかえれば送信権すなわちトークン(token)ということになります。そしてこのトークンの数を多くすれば、回線上に多くのデータを乗せることができ、ネットワークの転送能力が上がることになります。しかし現在のパーソナルコンピュータにすべての処理を行なわせ、しかもネットワークのスピード(ボーレート)の遅いCONSではこのトークンは1個のみと

なっています。

●CONSのデータパケット

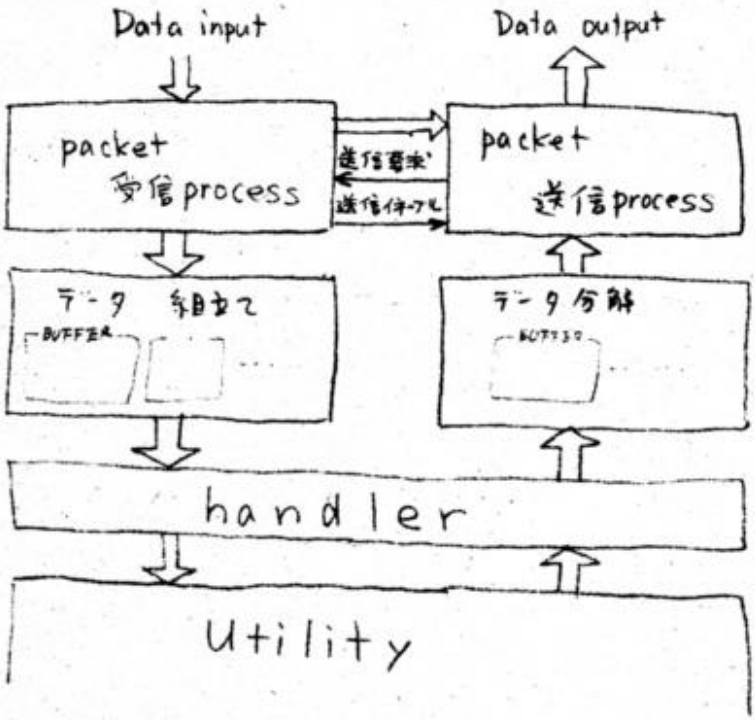


また このように1つのパケットの長さが短かいのは、回線の独占を避けるために有効となって来ます。例えば、Aさんが10Kバイトの大量のデータをBさんに送っている間 Cさんが Dさんに数十バイトのデータを急いで送らなければならないとき、パケット単位で分割されていれば、その間に 割り込むことが可能です。

—— CONSのアーキテクチュア ——

CONSのアーキテクチュアを図で書けば 大体下のようになります。

ネットワーク上のデータはいつやってくるか分かりませんし、常にデータの



監視をしていないとデータがそこでストップしてしまうことになりますから、当然マルチプロセスのシステムが必要となります。ここでネットワーク専用のプロセッサをステーションに備えてやれば良いのです。このプロセッサをNFP (Network Front Processor)と呼びますが、今回はコンパクトさ、簡単さを追求した実験ということで、NFPはホストコンピュータの簡易マルチタスクモニタを用いることによりsoft的に実現されています。

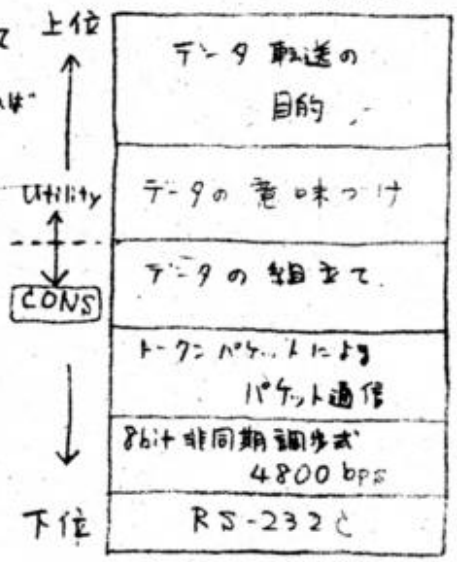
本来ならば送信processもNFPによって行なわれるのですが、Utilityとの同時性は必要ないのでCONSではUtilityと同一タスクになっています。

ここでプロトコルというものにすこし触れてみましょう。

プロトコルとはネットワークにおいて混乱がおきないように定められた一定の規則にもとづいた手続きのことです。プロトコルには、さまざまなレベルのものがあり、下位のものでは、ディジタル回路における論理の'0'と'1'をどのように表すか、といったことから、パケット通信のためのアルゴリズムや送られてきたデータの意味づけなど、これらの全てが送信側と受信側とで同期していなければデータのやりとりはできないのです。

通常プロトコルというとネットワークのシステムの部分、つまり図の点線より下がとくに注目されますが、実際には全てがプロトコルであるということをついつい忘れがちなので注意を要します。

@プロトコルの階層構造



また、これよりもさらに重要なことは、このプロトコルの最上位のものはデータ転送の目的ということです。

どんな通信でも、それには 通信のための目的 があるはずです。
そしてそれは 送信側も受信側も当然ながら一致していなくてはなりません。
つまり、Aからこへこのプログラムを送ろう、ということさえもプロトコルのひとつであるということです。ひとつというより最も重要なプロトコルといっても良いかもしれません。
このこのが 矛盾していると、正しいデータのやりとりが行なわれないだけでなく、ネットワーク自身を混乱させる原因ともなりかねないのです。

さて、CONSの概容をお話ししましたが、今ではCONSは実験用として開発されましたが、その理由は、信頼性とシステムの柔軟性が実用レベルにあるかどうかまだ論議の余地があります。
ただ、アマチュアレベルにおける、ループ型ネットワークの実験はそれなりの価値があり、また NFPの採用で固定的なシステムではかなり強力になるのではないかと思っています。

CONS 開発team (Star fleet simulationを含む)

中島KAZU氏, @氏, 大神氏, 吉沢氏、並河氏、池田氏?
MASA.
そして
監修=もんくたれ

Take♡大先生 でした。

Mのたわごとコーナー

○この間TVで粘菌アメーバというのを見ました。彼らは普段は普通のアメーバと同じようにバクテリアなんて食っているのですがいざ食物がなくなると、なんと何千，何万という数のアメーバがひとかたまりの高さ2～3mmのかたまりとなってうごき出すのです。ひとつひとつのアメーバはただの単細胞生物であるのにその全体はまるで一つの生き物の様に動き出す様はまさに生命の神秘を感じると共に団体というものの在り方を見せられたようで本当に感動的でした。その後彼らはひとかたまりとなって卵のようになり、活動を停止して次の世代へ生命を続けさせるということです。

本当に30分でできる 風雅な

80 Micro Eighty Computer の つくり方 なのだよ。

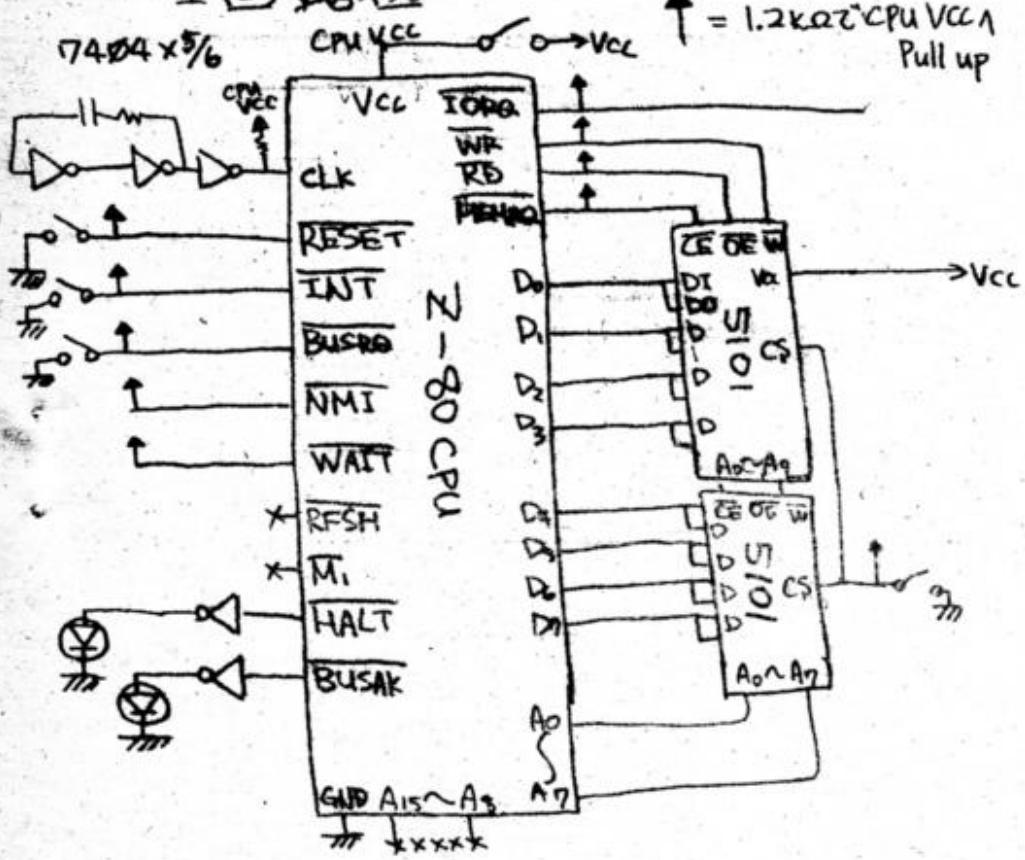
▷SPEC

はずかしいので、後で若干説明するにとどめる。

▷回路説明

不用だが、後で若干補足する

▷全回路図



D補足説明

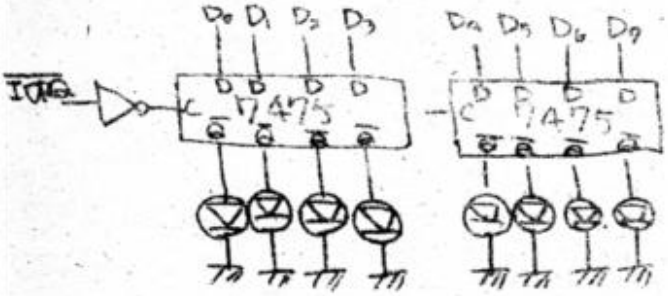
これは、実は CMOS-RAMに プログラムを入れておいて、Inputには INT. を、Outputには HALTを 使って送受するようになっているんですな。だから、RAMのVCCは 絶対に 切れない。

また、電源を切る時には、CPUの RESETを押したままで、RAMの CSの スイッチを ON、つまり RAMを Protectした状態にして、それから、CPU VCC を OFFにする。それから、CPU VCCを入れたら、90秒程 待ってやらないと、CR 発振の CLOCKが 不安定で CPUが 暴走する。CPUが あったまったら、RESETをかけながら RAMを ENABLE (つまり、RAMのCSのスイッチを OFF) する。

尚、VCCは 単3電池 4本で 動かしている。

▶ さてそこで、大問題、どうやって 5101に Programを 入れるか？だが、やっぱりここで、別な コンピュータさんに 登場 ねがわないと だめですね。(だから 3000円で できるとは、書かなかった)

まず、Host Machine には、8bit Portが 最低2つ、あとそれに 3bitあれば良い。そして、Z-80に BUSRQをかけて、DBに まず8bitつなぎ、次に ADRSに 8bitつなぐ。そして、MEMRQ, RD, WR それぞれに 1bitの出力を つなげば、あとは、読み書き自由、『君の MB-6890に RAMを256 Byte ふやそう。』という風情である。WAITに、細工を すれば、Host Machineで Z-80を SingleStep させることも 可能なので、6809を 使ったマシンでも、完全な Z-80の 開発が できる。しかも ¥3000 !!! 下は、このマシン用の おまけ出力



これだけで、何か できることが あったら 考えて下さい。

TEL. 075-721-1931
Takeo

べんとうばこに、組み込んで、『べんとばこん』と 名前でも つけてあげてね♡

マイクロ鉾

〔制御班〕

§0 概略

制御班というのは、マイコンでものをコントロールすることを目標としています。そこで今年は、去年もですが、CPUボードから設計製作をしていこうということになりました。そして、そのボードでモーターを制御し、マイクロマウスぐらいのものを製作してみようと考えてみました。このマウスは、去年も試みたものです。去年はCPUボードまでは順調に行ったようですが、センサー部分で大きくつまづいたらしく、CPUボードにひびかは入るという不幸と相まって、あと一歩のところで、時間切れとなりました。(今は、この名作も袋の中で寝っています。)

今年は、去年の教訓を生かして、みんなでいろいろ相談して進めようということになりました。

§1 CPUボード

まずCPUの選択ですが、制御の人達は全員レベル3を持っているので、「家でプログラムが製作できる」「6809のアセンブラを理解している」「GAMEと、コンパイラがあるので、それを使えば、ソ

フトが楽になり、拡張も容易である」などの理由から、CPUは"6809"となりました。原稿を書いている現段階では、「6809は、消費電力が大きくマウス向きではない」「予定していた程、各自が家でプログラムをつくるということは無理ではない」ということがわかりました。

ボードの設計は、いろいろな雑誌に掲載されているのを参考に、部品が少なくなるようにつくりました。ここで、プログラムをどうやって入れるのか、という問題が出てきます。去年は、パネルスイッチで入力していたようですが、今年は、マイコンでプログラムをつくり、それをCPUボードに転送しようということになりました。そこで、ボードにはIPL用のROMと、RAMを4K積むことにしました。また、マイコンとのプログラムの転送は、シリアルかパラレルかどちらにするのか、いろいろ考えたのですが、センサーのインターフェイスや、ステッピングモーターのコントロールに、PIAを使うので、それを流用すれば、すぐにマイコンとつなげるということで、パラレルということになりました。コネクタの抜き差しは、ピン数が多いので、やはりシリアルの方が楽のような気もします。以上で、ボードの仕様が決まり、あとはハンダ付けと、ROM焼きです。CPUボードのハンダ付けは、1人の人がやったので、すんなりといったのですがいざ電源を入れてみると、…動かない!? オシロを使って、必死のバグ取り、大きなバグが3匹

程ボードに住んでいました。まず、IPLのプログラム、次にノイズ、最後はテンプラハンダでした。幸い、どの部品も壊すことなく動くようになりました。（日立の6809は強い？アドレスバスをビンビンとけんかさせたのに，元気に動いてくれました。）

§2. 駆動部

駆動部分は、ステッピングモーターを使い、正確に動けると同時に、ソフトも楽になるようにしました。ところがステッピングモーターは高価なので、中古を使うしかない！しかたなく、日本橋へ行き、1個2000円で仕入れてきました。始めは、電池で動かそうとしていたので、電圧は低い方が良いと思い、規格が4Vのものを買いました。ところがこいつは電流を食い過ぎて電池ではパワー不足。電源は、外部から供給を受けることとなりました。外部供給にしてみたところ、今度は線が10mぐらいの長さなので、その線での損失が問題となり、3通りの供給方式をやってみたのですがどうもうまくいかず、持ち運びにも不便なので、結局、"トランス搭載型"ということになりました。ステッピングモーターのコントロールは専用のICを使い、プログラムが楽になるようにしました。予定以上に車体が重くなり過ぎてしまいました。

§3. コントロール・プログラム、

迷路ぬけが最終目標であるが、迷路の中を徹底的に調べて帰ってくるというソフトを先に考えることにした。それができれば、調べた結果をもとに、最短コースを計算し、ゴールに向って突っ走ればよいからだ。迷路探索プログラムのアルゴリズムは、ASCII誌に掲載されていたものを参考にしました。しかし、センサー部分が本に載っていたものと異なっており、探索法や、Mapへの結果の書き込みは、独自のものとなっている。Mapは、メモリ容量の関係で、一区画に1バイトを割りあてたので、bit操作が必要となり、GAMEでは少々記述が複雑になった。また、コンパイルしたオブジェクトも長くなってしまった。現段階では、まだ完成に至っていないが、動くだけなら何とか動けるので、センサー部分がうまく動かないなら、「あらかじめ、プログラムした道をきちんと動く」というようなことで、終ってしまうかもしれない。

以上で制御班の活動内容の報告を終りたいと思います。

(古泉)

COMPUTER LANGUAGE

Project: Lime (Limited Expression)

Report

1981年度 報告書 (しまっしょ) by team System'81

Limeを奥田先生から引きついでやることに(勝手に)決ったので、昨年度のしめくくりをしておこうと思う。MASA氏のLime秘話と合わせて読むと爆笑できると思う。

そもそも、言語とは、表現の手段である(また説法か…)
しかし、コンピュータ言語とは、手続きの表現であるから命令語の列となってしまう。めっちゃに新しい言語でも(Ada等)それは同じだ。
[ただし、Prologだけは全く違うが。(プロローグというぐらいだからアマチュアには、手におえない???)何が、どう違うかは、自分で勉強せえ]
あと、Ada等は、例外処理、並列動作制御、ぐらいが強化されている。LispやLogoは、全く違う考えの言語で、データは、ほとんど構造化できない。が、人工知能には、便利!

さて、Limeは、ALGOL-60 typeの古い構造化言語で、DATAの構造化は、いっさい考えられていない。(えらそうに言うな!)
ひょっとするとLime-2は、構造体を含むかもしれないが、
「Small is beautiful」というのが、Limeの根本思想の一つなのでまだ解からない。

Limeの基本思想

☆ いかなるHard-ware上でもOBJECTが実行できる。
☆ 手続きの表現手段としていかなる時も使える。
☆ 核(Core)にProcessor-Aなるマシンを想定している。
☆ Processor-Aはインプリメンテーションが容易である。
☆ データ・ファイルも単なるI/Oも同一のファイルとし、統一的にI/Oチャンネルで扱える(ただしサブセットでは、はぶかれる予定)
☆ System記述が可能なこと。

上記のことから、各Hard-ware上にProcessor-Aを移植すれば Limeで書かれた、プログラム(のオブジェクト)を実行できることがわかる。それによって、もしNet Workを組む時、非常に有効なリソースとなることが予想できる。

そこで、データ通信が技術的にも、金銭的にも十分手近になっている現在、Limeは非常に有利な特性をもっている。

しかし、コンパイラを使るのは、それなりに しんどいなあ。
しかも、多人数でやる時は、しっかりとコーディングしないと、うまくいかないし…。
はっきり言って、Game程度の言語でLime-Compilerを複数人で書くのは、自殺的だ。こんな危険なProjectのリーダーは、やはり言いだしべえのMASAが、やるしかないのでは…?!

昨年度はstaffの分担を

MASA … PROCEDURE 頭部, VARIABLE ACCESS, PROCESSOR-Aなど
@ …… VARIABLE STORE
澤谷 …… EXPRESSION
奥田 …… PRINT & FORMATTERS
Take … STATEMENTS

としたが この2人は、全然 働から分なかった。

Lime in '81

written by MASA

○Prologue (*Prolog)

去る1981年の春、我がコンピュータ部システム班における今後の主な活動として構造化言語をつくることになりました。これがLime計画のはじまりなのであります。その後一班では他の言語の勉強をしたり、薬を飲んだりしながら、新言語の体系は少しずつ実体化してきました。

ところが、現在のところこの言語は完成されたSyntax(文法)が未だ公表されていません。なぜか?(ほーやたからさ)答えは簡単です。Limeという構造化言語は実はまだ実在していないからなのです。一部を除いて、Lime計画というのは現実的な完成を見たものはほとんどないのです。私はOne of the System班として声を大にしていいたい。「いったいワシはなにをやってきとったんじゃ!」

そこでここではLime計画の去年一年間における つまづき具合とLimeのcoreである仮想プロセッサの存在意義について書かしてもらいたいと思います。

○Chapter I —— Lime計画 in '81 ——

まず、この構造化言語の主な使用目的ですが、これは一応システム記述ということに限定されています。というわけで名前も一応Lime…Limited expressionということになったのですが、これはこの名前が誕生した場所…LIPTONに"ライム・ティー"があったためであるかどうかは今も謎につつまれています。

システム記述とはLimeを使ってO.Sや他の言語自身を記述すること

ですが、実際には汎用性もかなりあるのではないかと思います。(要するに、特殊な機能はついていない) そしてこれに適応するために、byte単位のDATAのアクセスとFile管理をつける、ということになったのですが、これを実現するため、前者はとりあえずLimeに2つの変数のタイプ・2byteのinteger typeと1byteのcharacter typeを設定しました。後者については、File操作専用の仮想プロセッサをprocessor-Aの外部に置き、この2つをLinkすることも考えました。しかしどちらも細かな問題や計画の変更点が多くなりかなり実現がむずかしくなります。そこでsystem班では力強くこう次のことがなされました。「これからつくるのはtiny Limeにしよう♥」

最終計画で上記の機能が省略されたのはいうまでもありません。

○Chapter Ⅱ —— processor-A ——

ここで、私の担当であったprocessor-Aについて、ちびっと書いてみたいと思います。

processor-A (以下Aと略す)は、Lime計画の中心となる、高級言語指向の仮想プロセッサで、このプロセッサの解するマシン・コードをA-codeと呼びます。

Aはひとつのcpuなのですが当然ながらその様なLSIがあるわけでもなく実際には、現存するマシンの上でシミュレートされます。つまりA-codeを一種のインタプリタによって現存するCPU (z80, i009 etc..) のもとで実行されるのです。しかしA-codeは高級言語とは違い、一種の原始的なマシン語ですからインタプリタといってもかなりの高速で実行され、インタプリタ自体も非常にコンパクトになっています。(1KB強)

さて、このAがLime計画の中でどのような役割をしていたかといいますと、Lime systemでは、ソフトウェアはすべてこのAのもとで駆動されます。具体的にいえば、Limeという高級言語はAcodeで記述されたコンパイラによってAcodeに変換され、コンパイルされたプログラムは基本的にはAの上で実行されます。

つまりLime systemではソフトあるところ必ずAがあるわけで、このことが"Aが Lime systemのcoreである"ゆえんです。

しかしこの体系はUCSDのPASCALを中心としたP-systemなどですでに実現されているもので特筆すべきものではないかもしれません。またP-systemはOSも含めた非常に大規模で多機能なものでLimeとは比べものにはなりません。しかしその分Limeは手軽さ、ポータビリティの良さを備えているといえるでしょう。

この仮想プロセッサを中心とした体系には一体どの様な利点があるのでしょうか。高級言語を一度Acodeにコンパイルし、それをインタプリタで実行すれば二度手間になるような気がします。また一部のBASICでは中間言語でメモリ上に格納してそれを実行するものがありますが、それらとはどう違うのでしょうか?

それにはAのオブジェクト効率や実行速度が問題となります。Acodeは高級言語の中間コードよりはずっと原始的ですからインタプリタの実行速度はBASIC等のそれよりずっと高速で、直接マシン語を生成するコンパイル言語より少々劣るだけであると思われます。(言語のレベルに左右される。)しかし、マシン語に比較するとより高級言語に対応しやすいAcodeでは、コンパイルの際生成されるオブジェクトの大きさはかなり小さく、数式処理では実に半分以下になると予想されます。

また、別の大きな特徴として、異なるCPU間でオブジェクトが共有できるという点が挙げられます。例えばPC-8001でLimeでプログラムを開発すればその作成されたオブジェクトをL3の上でそのまま使用できるということです。このとき他機種(L3)で必要なものはAのみです。つまりLime Systemのすべてのソフトは Acodeで記述されていますから、自分でハードウェアを作っても、比較的簡単なAのみをつくれば Lime systemがすぐに起動でき、また他機種ともソフトの互換性が生まれるというわけです。現在GAME言語か

多くの機種で使用され、この役割を少なからずとも果たしているわけですが、実行速度が少し遅い(GAMEはコンパイルすれば互換性は失くなり、またコンパイルするのにはかなり多くのメモリを必要とする)のと、Run time routine としてのGAMEインタプリタが少々大きいことなどが問題として残ります。しかしGAMEは現にアマチュアレベルで実用化されており、(くそ!!)その存在価値は大きいといえるでしょう。

Aの場合ではそのインタプリタのコンパクトさとLimeに限らず、他の言語への拡張性のあることなどGAMEにはない特長があるわけですが、そこには欠点がないわけでもありません。互換性に重点を置いたため、入出力の管理が貧弱であることや、グラフィックなど機種独特の機能が生かせないことがあります。システム記述という本来の目的からすれば、前者は非常に大きな問題ですが、後者は不必要ということになります。が、Lime systemという統合的なsystemを考えた場合やはりひとつの障害となってくるでしょう。

残された問題は、この2つの機能を実現するための、他種の仮想プロセッサ、および Native coad processor つまり実在するプロセッサをAから呼び出すという機能を備えるということになったのですが、この点についてはまだまだ難しい点があるだけでなく、次章で述べる大きな問題にぶつかり、Lime計画は、その進行を停止したのでありました。

○Chapter Ⅲ —— 提言 ——

実は、この記事と併載されているLimeに関する紹介記事に「言いだしっぺの……」とありますが、実はこの言葉が重要なカギを握っていたのです。

実はこの記事は、今年の春にコケかけたLime計画の復活を目ざし、書かれたものなのですが、その中に例の言葉を見つけた私は仰天してしまったのです。なぜなら、私はこの計画の言いだしっぺは他でもないSystem班の影の親分Take氏であると信じて疑わなかったからなのですが…

ということは今までTake氏と私は言いだしっぺをお互いに思いあずけして、知らぬ間に責任転化をしてきたことになります。これでは効率よく計画が進行するわけはありません。

Take氏と私はこのことを知ったショックで戦意を喪失し、その結果、Lime計画はお蔵入になったままその活動は停止して現在に至り、そうしてこの記事もLime計画の復活の詔を高らかに告げる内容より一変して計画の挫折のいいわけをずるずると書くはめになったのです。

ちょっと、そこのprojectの親分をやっている貴方！気いつけてくださいよ!!　貴方のとこの計画の言いだしっぺ知ってはりますか？　また!?知らん!?　そーか、もうそら破滅の時が近こおまっせ!!

○Epilogue

以上のように、Lime '81計画の概要とその結末を書いてきましたが、そもそもひとつの計画なんていうものは苦労と困難の繰り返しのはずです。このLime計画でも、いくつかの困難はのり越えたはずですが、ひとつつまずいただけで全てが終ってしまうというのは、根本的なところの体系が弱かったせいかもしれません。

このような記事をこれから誰も書かなくてよいような活動が行えることを願って止みません。

—— E N D ——

DATA SHOW '82 日記(その1)

▷ うちら System では、PERQ, CONCEPT, DOMAIN や XEROX の J-STAR に、あこがれていましたのでした。

▷ 10月20日に突然「はるみに行くべ」という Takeo の言葉に、@サン は「行きましょか」と、答えて、実験も かなぐり捨てて PM. 6:Rのどんこーで、一路 Tokyo へ、向かったのでした。

PERQ
by Three Rivers Computer

* Multi Process O.S. が、できたてで Multi Window で、ガンガン デモってた。
@:「はっ、はやい!?」
* Editor は、ASCII誌にのってたとおりだった。しかし、それは、Single Task O.S. の上で、動かしてたのみ。Multi O.S. は Bug っているような気がする。

▷ XEROX 以外は、みんな商社が輸入しているので ちっこいブース ばっかりだった。でも、PERQ の"理緒"は、割りと大きいブースだったのだね。

▷ しっ、しかし、あんまりちっこいので、なっ、なんと Lisp Machine の、「LAMBDA」、「CADR」を 一応、見かけていながら よく見つめなかったばかりに 見損なってしもた。あへ〜

MultiProcessのおべんきょう

(その2)

Takeb

▶「MultiProcessのおべんきょう(その2)」と、なっているが、どこを捜しても、(その1)は ありまへん。

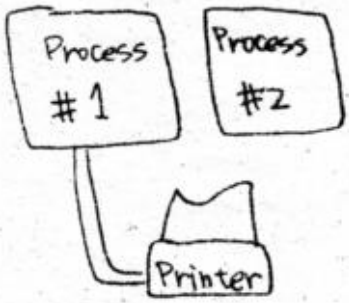
(その1)は、実は 夏の合宿中に、私めが、黒板に書きなぐってしゃべったことです。

▶ そこで、知らない人のために 復習しておきます。

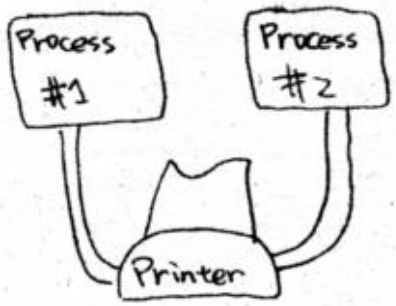
(その1)では、感覚的に 解かりやすい様に、プリンターなどを、2つ以上のプロセスで共有するような場合などを例にとって セマフォや、セマフォの書き替えについて、おべんきょうしました。

下の絵を順番に、見てネ。

1 プロセス#1さんが、プリンタを使っていました。



2 プロセス#2さんもプリンタを使いました



3 プロセス#1さんが打ちかけなのにプロセス#2さんが打ちだしたのでプリンタの上には、#1と#2の出力がまざって、出てしまいました

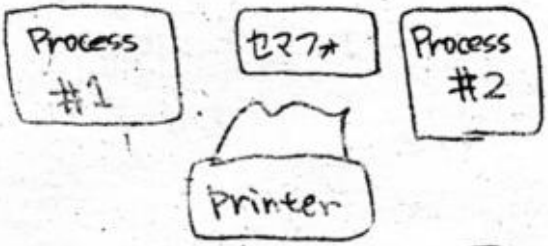
おワれはタぷシろハプロセせすいちスニ毎デのじスゃ。!

「おれはぷろせすいちなのじゃ!」

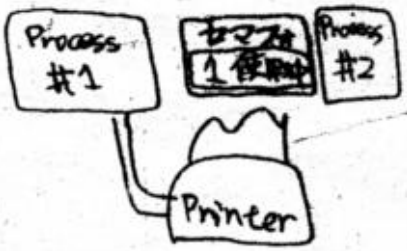
「ワタシハプロセスニデス。」

のつもり。

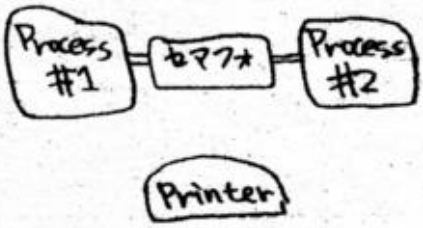
4 そこで、プロセス#1さんも プロセス#2さんも セマフォを見てから、だれも使っていなければ プリンタを使うことにしました。



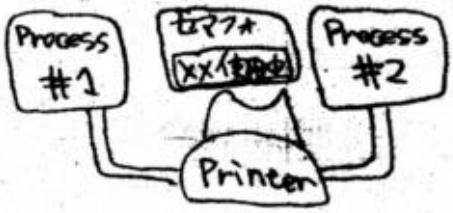
5 しばらくは平和な日々がつづきました



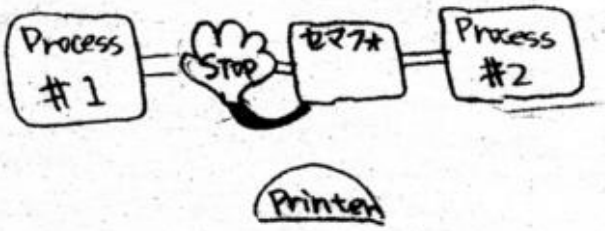
6 ところがある日、プロセス#1さんと プロセス#2さんが同時に セマフォをのぞくと・・・



7 セマフォには 使用中 と書いてなかったので、2人とも 使用中 の札を出して プリンタを使ってしまいました



8 そこで、セマフォをのぞいてから 使用中 と書くまでは、もう一人は、待っていることにしました



9 それからは、ぜったいに2人同時にPrinterを つかうことはなくなりました めでたしめでたし。

という様なことで、「セマフォのAccess(Test&Set)は非可分で排他処理(リージョン)を行なわねばならない。」というのが、夏のMulti Processのおべんきょう(その1)の内容でした。

▶さて、そこで おべんきょう(その2)だが、今回は、プロセス間通信という様なことをやりたい。そこで、少し話しはちがうが、下を読む！

▶昨夏の合宿の時も、少し言ったが、特に I/O 装置の Driver は メインの Process と、七切り離して 別をプロセスとしておくのが良い。そして、メインの Process とのやりとりは、FIFO (First In First Out Buff.) を、通して やるのが良い。そして I/O Driver は、いつ起動されるかというと、それは、I/O 装置から I/O 要求が来た時で良い。

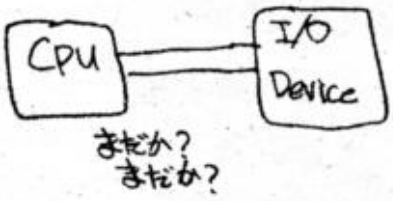
逆に いうと、I/O 装置は、Processer に 比べると、遅すぎるので、I/O 装置から、1つ入力(又は出力)した時でも、次の入力(出力)までは、ものすごく時間があるので、そこで ポーリングなどしてれては、むちゃくちゃ むだなのだ。(ただし、普通の パーコンは、キー入力以外には、することが ないことが多い。)実際 Z-80 の System は 周辺 Device が Interrupt Oriented に 設計されていて、インタラプトの応答は、非常な高速で可能に なっている。しかし、ポーリングしようとすると、絶対に 不可能で、仕方なしに、PIO を 1 port もって、その内の 1 bit に ポーリング入力を入れて、レベルセンス するしかない。という おそろしく すばらしいものだ。実際 Zilog の LSI 群を使いこなすには、かなり コンピュータを 知らなければ 無理である。(Z-8K + MMU を、TV-Game 屋が 使ったら、グラブに PERQ 買うだろう。)

変な 事を 言ったが、要するに、I/O DRIVER は 別な Process にすれば 得ということである。しかし もし、HARD WARE が、許すなら I/O DRIVE ROUTINE と I/O 装置の間に HARD の FIFO をつけた方が もっと、Processer (CPU のこと) の 負担は、軽くなる。

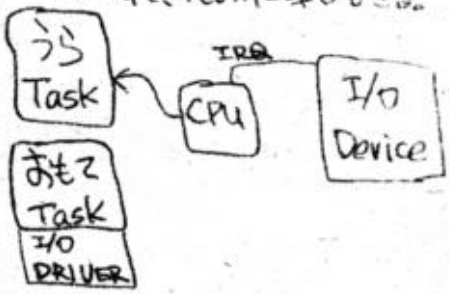
次のページの 絵を みてネ！

30 page

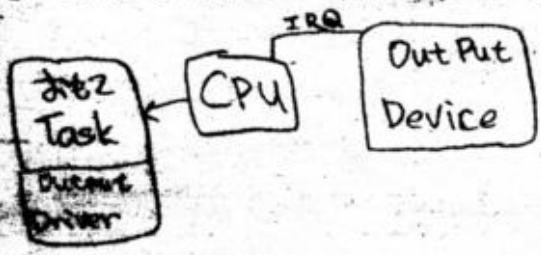
1 CPUが ポーリングすると、他の仕事できない



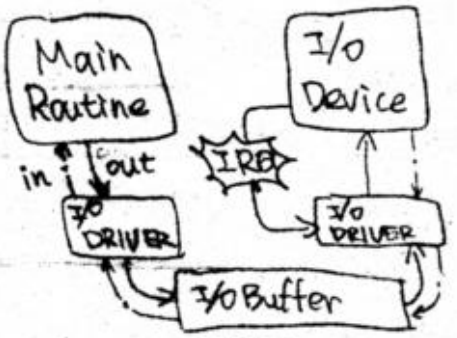
2 I/O Driverを割り込みで うごかすと、他の仕事もできる。



3 I/O Driverを割り込みで うごかす時、出力を待っているなら 同じプログラムの実行を続けても良い。 (ただし、マン・マンインターフェース部ではまずいことがある)

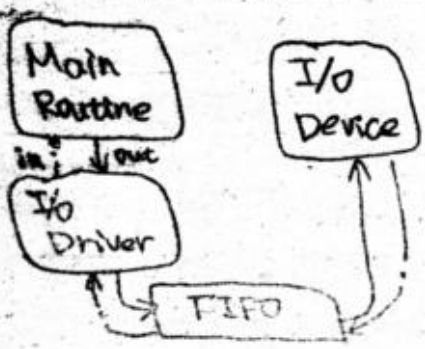


4 2も3も実際には Softwareで FIFOを形成しているだけなのだ。



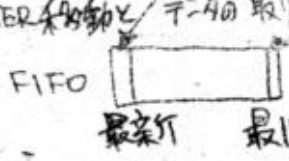
(※I/O DRIVERは2つの部分に分かれる)

5 Hard wareでFIFOを つけても 4と同じ。



Z-80K Systemには、周辺LSIに FIFOが用意されている。

注意: FIFOをSoftで形成する場合 送信のPOINTERと 受信のPOINTERとくるわない様にするため、FIFOのPOINTERの動作と データの取り出し セットは 非可分である。



▶さて プロセス間通信だが、

▶今、見て来た I/O Driver と Main Routine の間では、2つの独立したプロセスの間で通信が、行なわれているのが解かる。(あたりまえだのクラッカー! ふる～。)

そして重要なことは、(隠しておいたのだが)実は、プロセス間通信とは、同期の作用があることだ。つまり、Main Routine が入力を欲しがって Input Driver へ データを取りに行く、ところが FIFO は空(Empty)で データが無いと Main Routine は 待ち(Wait)になる。もっと 正石確に書くと、Main Routine から 呼ばれた Input Driver が Wait になる。ところが、Main Routine も、後ろの プログラムを実行できないので、Wait に なるしかないのだ。もし、ここで、うら Task があれば そっちに制御が 移り、再び 表にもどれるのは、さっきの Input Device から入力が あった時だけなのだ。

▶だから、単純に、サブルーチンにすれば 良いものも 実は マルチプロセスで 動いていると考える。ただ、パラメータを与えないと、同期作用によって、Wait になっていると考えれば、マルチプロセッサや、データフローマシンを考える時の足しになる。サブルーチンを 呼んでいる間、Main ルーチンは どうなっているかというと、サブルーチンからの戻りパラメータを要求して Wait になっていると考えるのだ。

すると、ルーチンという概念は、突然 プロセスという概念に 内包され(部分集合になる)てしまい、また 美しい System へ一歩 近づくのであった。

▶かけ足で、マルチプロセスをやったが、(その3)では、夏の時、わりとくわしくしらべった、マルチプロセッサと、コンカレント・パスカル、それと ベーシックで 割り込み 記述と実習を、やって みたい。(いつのことやら)
コルーチンを 忘れとったから、それも やらんと あかん。

32 page

▷突然ですが、前ページのおまけ。

今、流行のUNIXやOS-9 LEVEL 2 は、マルチプロセスはもちろんサポートしているが、その中に"Pipe"という奴があってこれがプロセス間通信用のFIFOなのだね。おわり。(1982.Nov.16)

☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆

DATA SHOW '82 日記(その2)

▷Tokyoに着いたら朝の5時ごろで、まだ暗かったね。しかたないから、三軒茶屋のつれの所へ行ってねこみをおそった。野郎のふとんに2人もいたからびっくりして、「おっ、すまん。ええか?」と、尋ねたら、もう1匹も男だったチクショー、ビビらすな!

DOMAIN

by apollo Computer Inc.

昔、ASCII誌に68000のDOMAINと載っていたので、どんなもんやと思って見たら、もう無茶苦茶速い。そう256 userサポートする能力があるらしいから、シングルユーザーで10か20のプロセスやったら、へみたいな、もんやろ。カタログ見たら、オリジナルのワイヤードロジックのプロセッサらしい。もちろんMicro Programableとにらんだ。1ユーザー15プロセスできるらしい。そう、マルチウィンドに、それぞれちがう絵を描きながら、TEXT-Editorを動かしたり、Windowを作ったり、消したり、拡げたりしとった。Key Boardの右に15cm×20cmぐらいのゴムを張ったAreaが有り、Digitizerの代わりをしていた。

CONCEPT

by CORVUS Systems ← OMNINET & CRASTER? で有名!!

DOMAINのとなりのブースでやっていた。こいつは、うわさどおり正真正銘の68000やった。どつぼの遅さやった。やっぱりMicro Processerでは、Bit Map Displayのハンドリングは無理らしい。UCSD-PASCALのネイティブコードコンパイラを持ってるらしいが、それでも、おそかった。単に見かけの印象では、PC-8801のちょっと速いだけという、グラフィクと、エディタだった。ガックリ。

ちなみに¥2490,000

FM 中間色ペイント

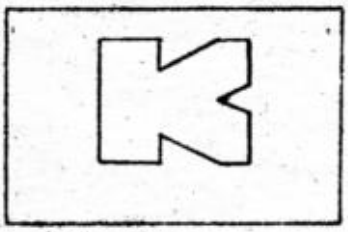
by Fukushima

お絵かきマシンとして定評のあるFM-8のグラフィックに使用されている中間色塗り法について少々書いてみたいと思い筆をとりました。

FM-8は 640x200, ドット単位8色の色分解能をもっています。ここで誰でも思いつくのが1ドットごと交互に違った色を並べれば人の目には混って色が合成されたように見えるだろう，ということです。事実その通りでPC-8801等にはタイルペイントという機能でそれが簡単に実現できます。しかし残念ながらFBASICやサブシステムにその機能がありません。そこで当初は色のラインを交互に並べる方法で色を作り不要部分を消まして必要部分に色をつけていました。

しかし、これでは塗る範囲が自由でなくまた、色をつけるのに頭をしぼらねばなりませんでした。そこでタイルペイントルーチンを作り中間色を塗ることを考えましたが、この場合、メインとサブとの関係上メインルーチンをどこに置くかということ，また実際に境界線の判断に大変時間がかかることがわかったこと、そして何よりも、

バッファのメモリの問題と、色ドットをセットするのに時間がかかるという点があり、結局中断してしまいました。しかし中断の最大の原因はすでに中間色を簡単に作成し自在に色つけが出来る別の方法があったからです。つまり中間色直接塗り法は、この別の方法が出来た後に作り始めたわけです。これはFM-8のGET@ルーチンおよびPUT@ルーチンの特性を利用するものです。



CRTの図

左の様な図を考えます。Kの内部を緑と黄の中間の色で塗る場合を考えます。

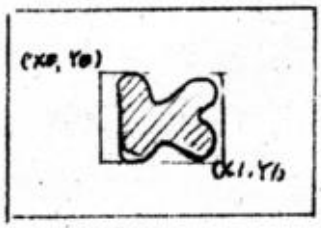
この方法ではまずKの内部をバックカラーと異る別の色、たとえば青で塗ります。これで青の色となったところが、目的とする領域であることが簡単にわかります。そして、この青の色を他の色に置き換えていくのです。これにはGET@のルーチンとPUT@のルーチンを使用します。簡単にはX軸に垂直な方向に1ドットラインずつ青の色のみをGETし同じ位置に色指定を変えてPUTすればよく、そしてその色をX座標の偶奇によって変えれば交互に色が並び見た目に中間色となります。そしてこの色変換を必要とする図形を囲むBox形領域に適用すればよいのです。これにより、1ライン分のメモリをバッファとして使用するだけで、簡単に中間色が実現できます。この方法をクロマキー縦変換法と呼んでいます。この方法により合成可能な色は2ドット混色で28色、3ドット混色で228色

の合計242色で原色の8色と合わせて260色表示可能です。しかし、3色混合においては濃い色と淡い色との混色のとき縦縞が目立ち、使えない場合もあり使用上注意事項です。

このクロマキー法は先に述べた縦変換法の他に当然横変換法もあります。この方法は見たほか縦変換よりずっと良いので現在はすべてこの方法を用いていますが若干プログラムが大きくなります。また、色が縦縞になりにくいように工夫することが出来ます。

以上のクロマキー法と呼んでいる方法はペイントしてからクロマキーを行うため2度手間を行っているようですが、結局直接塗りよりメリットが大きいのでした。唯一欠点があるのはGET@でとりこむ色を工夫しないと、先に塗った別の色まで色変換してしまうということですが、使用者が注意すれば十分防げ、またこれを利用して模様を作ったりすることも出来ます。(3色塗りと2色塗りを利用すればモアレ縞が出来る。ハードコピーをとればきれいに見える。)

参考までに縦変換と横変換のプログラムを次頁に載せておきます。オールBASICですが、BIOS利用等のマシン語ルーチン使用のものはほぼPAINTと同等の速度で色変換(2ドット混色)できます。



CRT

使用 変数

X0, Y0, X1, Y1 … 領域
C0 元の色
C1, C2 置き換る色
X, Y, I, J, A%(50) ワーキング用

縦クロマキー（2色）：

```
FOR X=X0 TO X1
  GET@(X,Y0)-(X,Y1),A%,G,C0
  PUT@(X,Y0)-(X,Y1),A%,PSET,C1
  SWAP C1,C2
NEXT X
```

横クロマキー（2色クロスかも）

```
I=(X1-X0)¥16+1
FOR Y=Y0 TO Y1
  GET@(X0,Y)-(X1,Y),A%,G,C0
  PUT@(X0,Y)-(X1,Y),A%,PSET,C1
  FOR J=0 TO I
    A%(J)=A%(J) AND 21845!   ← &H5555
  NEXT J
  PUT@(X0,Y)-(X1,Y),A%,PSET,C2
  SWAP C1,C2
NEXT Y
```

3色混合は 応用で 出来ます。

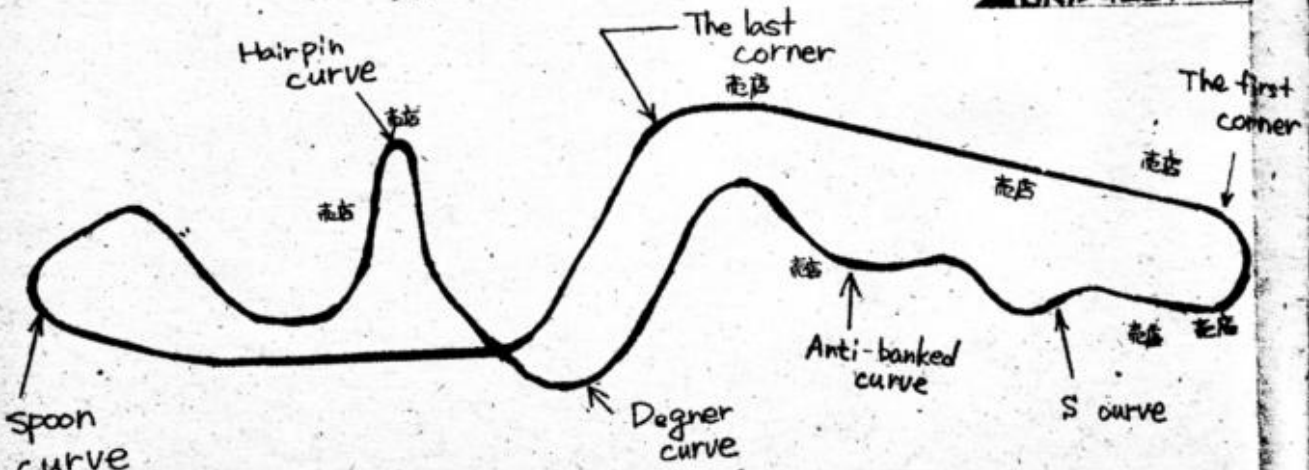
おしまい。

'82 JAF 鈴鹿グランプリ自動車レース
初体験レポート　　KAZU

11月6日(土)　昨日飲んだシーバスリーガルは"うまかったな～"と思いつつ Takeo と、アルコールの残った体にむちうって朝潮橋へ Xカトロショーの最終日を見るために阪急に乗り込む。しかし梅田についた時には PM4:26 であった。ちなみに Xカトロショーは PM4:30 までであった。しかたがないので、YISを見になんばに行き、うろついて PM10:30 頃帰宅し明日、朝は早く出発しようと思うので Mを乗りなれない MARKⅡ (但し53年式)でむかえに行き、帰っていっしょに寝る。雨がつよく降りだしていた。

11月7日(日)　ついに待った GP·F2 決勝の日だ。下宿人の B氏と Y氏がブルーバード(2000 SSS L)でむかえに来た。AM 6:30 山科を出発した。雨はまだ降っている。道はさすがにすいていて1時間半ぐらいでついたと思ったら、HONDA工場の裏の駐車場に止めさされた。ついに鈴鹿だ!! テレビで見るのと同じような音がしている。ブォーン。いたたまれなくなって、第1コーナー前の直線のフェンスにかじりついた。ブォーンという目にもとまらない物体が前を横切った。これがF2だと思った。雨はまだ降っている。

*** 鈴鹿 GP コース ***



第1コーナーを越えてS字カーブ手前の売店前に4人は陣取った雨は少し上がってきている。AM10:00 F3チャンピオン決勝の幕は上がった ポールポジションの中本恵喜のラルトRT3が優勝 期待の女性ドライバー、マーチ783の吉川とみ子はおしくもスピン、リタイヤとなってしまった。
その後FFスーパーシビックがあった。30台もほとんど同じ性能の車が走るので大変に笑った。だれが勝ったかは知らん
またその後PM12すぎ(4覚えてない)女性ドライバーによるCity TURBOレースがあったが、とみちゃんが段突でとっぷ。優勝
PM1:30 ついについにF2 GP決勝である。ポールポジションのスピリット201ホンダのステファンヨハンソンが出おくれた。2位につけていたJOHN PLAYER SPECIAL-HONDAの中島悟はいつものことながらスタート出おくれ4位ぐらいであった グォーンと ド迫力で第1コーナへ。
とDHL マーチの松本恵二がクラッシュした2周目トップのヨハンソンにTail to Noseで中島が第1コーナに突っこんできたが抜けぬままにS字コーナへと消えて行った グォーン 4周目場内アナウンス"トップ・2位を大きく離しました!!"と耳に入ってきた。グォーンと直線を中島が独走、きれいにコーナをクリアして行った。このまま彼は優勝を手にした。2位につけていたヨハンソンはあせって最終コーナでスピン。その間にマイペースのPENTAX MARCHの星野一義が2位に浮上したよって日本人が1,2位をとった。
このあとFJ-1600があったが見ずに渋滞の1号線へと向いPM8:00頃大津のスカイラークでデラックスハンバーグを食って帰った

本日の教訓
・雨ふりは長ぐつをはこう!
・いらぬカメラより目に焼きつけろ!
・鈴鹿へは前日の夜から行こう!
(駐車でけへん)

この後4人が毎日
グォーンと言いつづけていることは言うまでもないであろう

目的＋計算機＝プログラム

Identity (PLUS) COMPUTER (EQUAL) Artificial Interigence

一見哲学風乱筆乱文、あくびが出る!!!

Take

そもそも、Computer なるキカイは、アイデンティティが ないのである。「そーじき」でも、「せんたくき」でも、おおそこの世のキカイは、生まれるべくして、存在の目的を与えられて 生み出された。

しかし、Computerは、最初は、ある程度 計算キとして、生を受けたが、そのうち 欲張りな人間によって「なんでも させてやる」と言われて 不幸な 運命を 歩みはじめた。

そして 今、大量の パーコンが、「なんでも できる」と言われて 売りさばかれている。(又は、いた) しかし、その逆に その使用は、確たる目的意識なしには、ほぼ不可能である。BASIC、マシン語、アッセンブリ、フォートラン、何でも知っていて 自由に あやつれるが なんにも プログラムを書かない。そういう人間には、何が たりないか?

それは、目的意識に 他ならない。自分の是非ともしたいこと、希望、理想、色々な 言葉が あるが、それは 行動の源動力なのだ。そして、その欲望を 達成すべく (正しい方向へ) 力をつくすから、前向きに、進み、プログラムも生産できるのだ。そして それこそ 自分自身を 形成する根本に ちがいない。(と決めつける) その人間

40 page

ただ、物を、ぼんやり、ながめても何も感じない。それは、ただ視野に入っているに、すぎない。「うまそうだ」とか「きれいだ」とか「いかがわしい」とか感じるのは、すでに「それなりの意識」で視ているのだ。そして、そこに、働いている「それなりの意識」とは、視てやろうとする欲望であり、その根底には、その欲望(欲求)を起こした、自分自身つまりは、アイデンティティが存在しているのである。

☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆

話しは、全然変わるが、このクラブは、全体としての個性が薄い。これも、全体としての統一テーマが、よくしぼり込めていないからだ。

だから、部員の中には士気のあがらない者もいる(俺や)だからよけいにクラブ全体が散漫として、まとまりにくいのだ。今年の場合は、ついに、3つの団体が独立して存在し、その連合体としてクラブがあるという体裁を持つに至った。これを、喜ぶべきか、悲しみべきかは、部員間の議論の待たれるところだが、多分この傾向はもっと激しくなり、ちょうど、科学部→{物理(部)化学(部)生物(部)}となったように、分化するだろう。その時、お互いに赤の他人のような顔はしないように、今から気をつけたい。

しかし、今、一つのクラブ(団体)としては、不安な程のアイデンティティーの欠如だ。

☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆

知覚、思考というのは、非常に人間的、また高度に知的であると、信じられている。

知覚とは、何か、たいていは「認識したものを、理解する」ことだと思う。理解とは何か、よく解からないが「思考する時に、有効に使えるように記憶しておくこと。」というのは、当たらずといえども遠からずだろう。認識は、また後で。

思考とは何だろう。それは「問題を解決するための戦略を計画すること」かな？そう、思考するには、まず問題がないといけない。問題とは目標が達成できないから問題なのだ。そう、だから目標がないといけない。目標をもってそれを達成すべく動き回っている自分、その自分こそ、アイデンティティーのかたまりなのである。

おっと脱線。

それでは、Computerに思考はできないか？

そのとおり！そうでないと、パーコンの電源を入っぱなしておいて、ブーブーいわれたら、たまらん。これは、あたりまえ。

しかし、パーコンは、何でもできるのだから、人格を、パーコンの上にインプリメントすれば、パーコンでも人間に、なれる。そして、それをプログラムなら、パーコンのBASICでも、チョチョイのチョイで、できあがって…しまうか!!!

それに、あんまり、自分勝手に、希望をもたれて、うろうろされたら邪魔なので、目標を限定して「オセロ」なり「碁」なり「三目並べ」なり（そう、「三目並べ」が重要なのだよ）を解かせたりするのだよ。

その時の「手の先読み」と言いうるのは、真しく「問題解決のために戦略を計画」しているのだ。しかし、不本意にも、今のアマチュアには、良いToolがないために、そのProgram中では、データを数値で表わさねばならない。もしLispでも使えば、全く同じProgramでも、データが生きた表現をとれる。

なにしろ、ただ数値演算しかしていないようなProgramでも適当な手を探索するようなものは、思考しているといっても まちがいではないと思う。逆に、どんな知能的プログラムでも、普通の計算機の上で動いていれば、最下位はビットパターンの集まりをあつかっているにすぎないのだ。

ただし、人間の思考が、可能な手を探索するだけに終止しているとも思えない節もある。しかし、発明、発見とは、可能だった手を今まで見落していただけなので、全部の可能な手を探索するプログラムなら、同じような発明、発見を行なう可能性もあるのだ。ちょっと、現実的な時間中には、無理だとは思うが。

そうは言っても、俺の作った「三目並べ」は、深さ3Levelの Mini-Max法で、俺よりよっぽど、うまくGameを行なった。もともと俺は世にあるGame全般へたで、「三目並べ」も一手ほどしか読めなかった。すると、キカイは、俺の思いもしなかった、場所へ打ってくるのだ。

つまり、キカイは、俺よりもよっぽど良い方法を発見するのだ。その頃、大先輩、藤井サンは、4手先を読んで、「おい、こいつおかしなとこに、打ちよったで」と、のたもうた。

上のように、可能な手を全部(または、もうちょっと かもり という説もある さぼって)探索するような方法(アルゴリズム?)を「Heuristic(発見的) adj」だと言う。そんなことをするプログラムを「Heuristic に Programming してある」と言う。そして、Heuristic Programming してある Program は、たいてい「Mini-Max法」を使っていて、Mini-Max Back upを起こすために、局面を評価するのにたいてい「静的な評価関数(EVAL. function)」をもっている。昔、人工知能の草創期の頃は、評価したい局面の、先の手や前の手をFactorとして与えたもの(動的な評価関数)も多かったが、あまりうまく動かないので流行ってない。とは、いえ そういうのも特定のGameでは、強いかもしれない。あとは、Mini-Maxを速くするために一般的には α-β枝かりをしたりするが、そんなことは、十年以上前から知られていて、ちっともおもろくない。

(木探索のやり方を知りたかったらスレイグルの人工知能(産報)」を読め。図書館にもある)

その他にも、やり方は色々あるのだろうが、よく知らん。
上の、Mini-Max法も使うには、評価関数が重要なのは、あたり前である。なにしろ、その局面が「どの程度有利か？」ということを「認識」するのが、評価関数なのだから。人間で言えば「将軍」か「社長みたいなもので、先を見こす力の知見が必要なのだ。
ここでいう認識も、仕事をうまく達成するという立場があるからこそ、その仕事にとって、今の情況が「どんな風か？」という価値判断ができるのだ。（その仕事にとって）
そこには、一個のアイデンティティーが存在して…うんぬん。（きれか）

認識といえば、認識自体、十分知的な作業で音声認識、図形認識、言語理解などは、十分困難な作業でそれぞれで一冊の本ができてしまう。
それは、ちょっと置き、例えばチェスの盤面を正しく認識できれば、記憶してある手筋をもってきて使うという、極あたりまえの動作ができるのだが、その時の認識がええかげんだと、きちがった手筋を使ってしまう等ということを行なえる?!
また、ある局面のすぐ起ったことを正しく認識（評価）できるなら、その局面は（後の局面で）「非常に良い手だった」とか、「あかんかった」とかの評価と供に記憶して、後で使えるのだ。（俗にいう「学習」じゃ）

ちなみに、評価とか認識とは、実はえーかげんなものである。
道を歩いていて、「おっ、あのおねえさまきれいやなぁ」と言っても、「そうか？」という会話がよくあるのは、その証拠じゃ?!?!
（主観（を形成している自分自身）が、他人とはちがうからだ。
そこには、自分自身のアイデンティティーが働いている）

認識とは、記憶と結びついている。「パターン認識」という言葉もそのためにある。過去にある記憶の中のパターンと、今認識しなければいけない情報を、うまい方法で記憶の中の型式に変換した後に、照合して、もしマッチするものがあれば、新しい情報は、それだったと認識できるのだ。

しかし、そこには、過去の記憶というものがある。そんなものキカイにあるか！それでは人間はどうだろう？そこそこの人間なら「常識」と呼ばれる記憶がある。だが、それがまた恐しいものなのだ。

この間「UNO」をしている時「にんじん色」と言った。それは「日本人なら(連想型記憶)にんじんを赤いと思う」という常識を俺は信じていたからだ。ところが「にんじんはだいだい色じゃ」と言われたのだ。ここに、もう二人の人間の間で、常識という記憶が異なっているのだ。

もとに帰って、キカイに常識は、ないので、プログラムには、それなりに常識を与えてやらねばならない。ちなみに昔、駒嵐と「非数値積分」の話をしていた時、駒嵐先生は「積分公式も人が入れとかなあかんの～、そんなんボロや」と、のたもうたが、積分公式は、積分の演算を定義した公理系の一部なのだから、人間でも、公理があれば、働いてくれると思う。

☆☆　☆　☆　☆　☆　☆　☆　☆　☆　☆

数学の公理とは、空間の定義であって、「今から僕(おれ)は空間はこんな性格ですよ」と言っているのじゃ。そして、それは数学の記号では書ききれない。(全く新しい概念を、今まで用の道具では表しきれない)そこで、自然言語の助けを借りる。

ここで おもしろいのは、さっきの「にんじん」の話に おなろうて、人間の底に ある ものは、普通名詞でさえ あれだけの誤解があるのに、もっと 抽象的な ものを 言葉で表現しようとすることである。

非常に 馬鹿げているのは、言葉を言葉だけで 解説することである。言葉というものは、幼児期の生活で ほぼ決まってしまうので、違う環境に いた人間の間では、自ずと、言葉の間に 食い違いが 出るものなのだ。それに、国語辞典を ひけば、そのことの証明と なるだろう。

まんにしろ、人間全員には 共通な常識(全く同じ記憶データベース)が 存在する はずは ないので、ほんの少しの表現で 完全な コミュニケートを 期待するのは 間違いである。欧米で、精神科が はやるのは、「自分の考えを 全て言葉で 言いつくせないことから 来る 精神的圧迫感の せいでは ないか?」と、分析している人が いたのを、最近の雑誌で 読んだ。

✲ ✲ ✲ ✲ ✲ ✲ ✲ ✲ ✲ ✲ ✲

そして、その幼児期の時に、埋められた 記憶に 照らして、根本的な 認識論や、価値判断が 行なわれるのだろうと思う。

だから、キカイにも、それなりの(公理に あたる、活動すべき空間の定義)ものを、与えないと 動けない。それは、プログラムの 目的も あるし、データの形を しているとも あるし、もっと 人工知能っぽい ものなら、学習で 与えられるであろう。

「三目並べ」の 場合は、3×3の 盤面で、どんなことを するかは、(ルールも含めて) プログラム全体に、ちらばって 書かれている。

もっと 事務用の プログラムでも、その目的(何に 何を、どう 演算するか)や、その プログラムの 活動する世界の 範囲(たいていは、数値の レンジ 等)も

1 Page

プログラム やその中のデータで示されている。
MASA(田中)氏が作っている「LIPS」「TALK」とか、俺の「APRICOT(TALKとコンパチ)」は、自然言語(英語)で、人間らしく会話するプログラムで、そいつのいる空間(まさに空間!)は、限定された名詞、動詞、前置詞、その言葉のSyntaxもごく限定されたものしかない。

しかし、けっこう人間的れ動くし、そいつの空間中で、ものを考えたり、尋ねたり、動かしたりできるので、かなり知的と言えるだろう。
(そいつの知識データ・ベースは、学習用と、分離性している)

ここで、一つ人工知能屋のなぐさめになるのは、うそか、本当かしらないが、チューリングが言ったという「人間が長いこと対話して、その返事が人間のものと見わけがつかなければ、どんな常識的な定義からも、それは思考している。」という言葉だ。(ただしこれは"2001: A SPACE ODYSSEY"の一節からの引用なので、本当にチューリングが、そう言ったかどうかは不明)

アルゴリズム + データ・ストラクチャ = プログラム というのも、なかなか哲学的真実を含んでいると思うが、この文は、このあたりで終り。

この文は、長々と何を言いたかったのかというと、要は、「手がかに、目的と、その目的にあっただけの知識を与えなさい。」ということ、あとは、人間として、物を考えるためのきっかけ作り。知識を学習するプログラムも十分おもしろいが、実用的には、せめて、幼児なみの知識(自然言葉と、しゃべるくらいの知識=「TALK」程度の常識)は与えておいた方がおもしろいだろう。その後で、その知識データ・ベースに会話で、知識を教えるというのは、興味深い。

ちなみに HAL は Heuristically-programmed ALgorithmic computer のことで、先の、Heuristic-Programming してあることが、わかる。

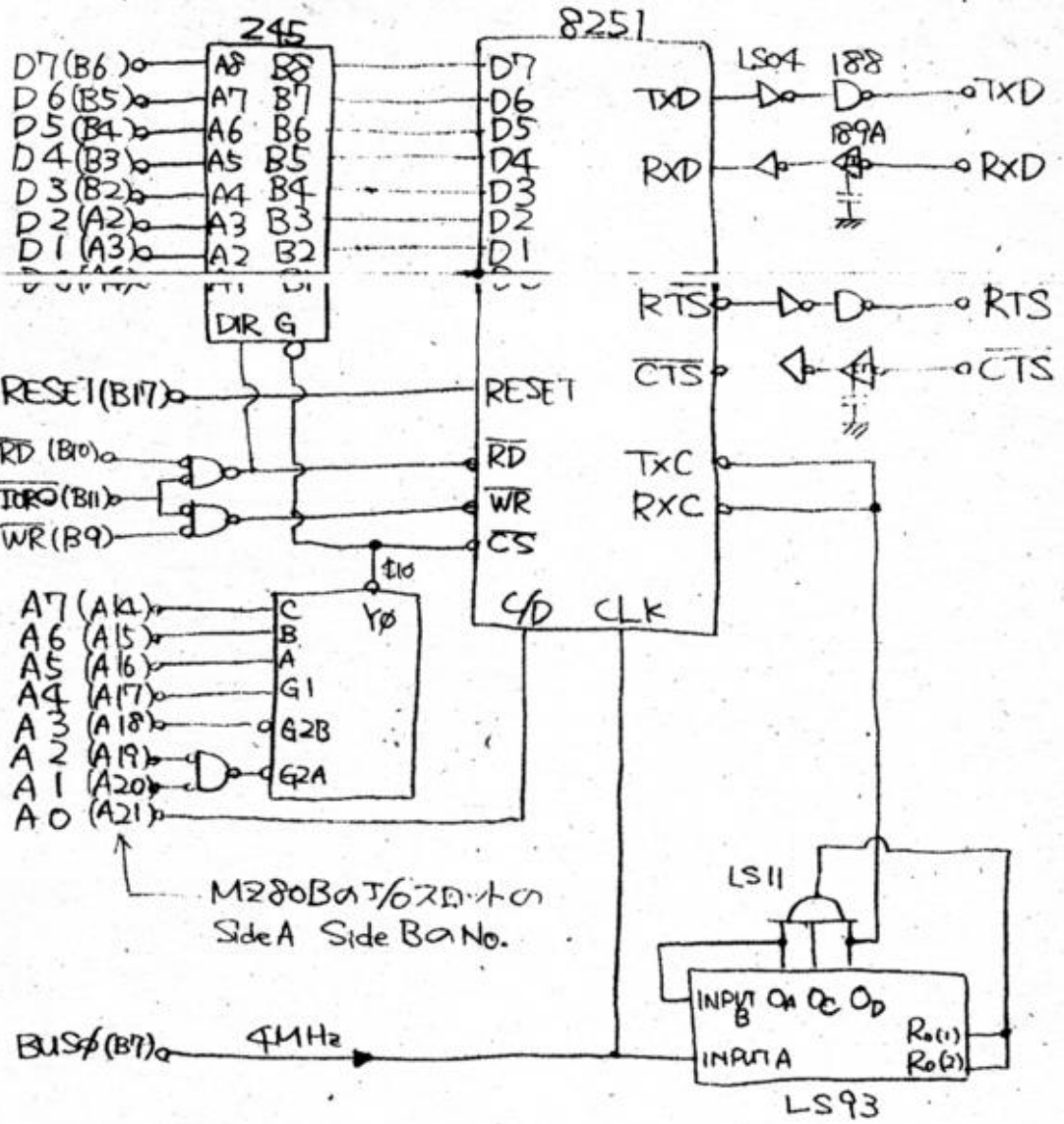
うちの MB-6890も、「3目並べ」するし「APRICOT」で会話するし、HAL並の知能がある???

おそまつ。 [1982, Spring]

RS-232C for 余裕のMZ80B

by KAZU

ついに出ました自家製 MZ80B用 シリアルI/O.
必殺の8251 4MHzを 13×64分周 etc.. カッコイイ?



こいを寺町ヒエ淵のおくの階段へ登る前のフロアーの
かたすみに置いてある 2.54mmピッチのガラエポ基板
に糸巻き込むとすぽりと スロットに入ります.
なお電源ON時に画面が出なければ やばいっすよ!!

10月の始め　まず、ボードを作り　はじめる
材料をかいあさる。水晶はもったいないので
4MHzのCLOCKを使用することに決定
*CONSのボーレートが4800ボーと決ったので　4M/13/64≒4808　とこれでOK!

10月中頃　アマチュア無線のコールサインが来たので、ラグチュウしながら　はんだ付けをする。(チック時に1ヶ所忘れていたことに気がつく)

10月　終り　家でいろいろとやるが動かない!
M氏のSSS TURBOでMZ80BをBOXへ輸送する

まだまだ動かない(原因、IORQ、RD、WRのかけかたがまずかった)これは　ハッカーのKoma氏が直してくれた。

11月の始め　13分周しているはずのCLOCKが　よく考えると現在LS93を使用している所にLS90を使っていたので　13も数定でけなんだ。(ちなみにLS90は10進のカウンターで　LS93は16進のカウンターです)
UARTのイニシャライズのかけ方の誤りから
M氏、Take氏、Koma氏の3本のプログラムができた。動くかどうかわからないので　この時にはTXDとRXDはショートしてあった
この3本のプログラムはどれも　1/2ぐらいの確立でしか動かなかった。原因を追求すると最下位bitが立ちっぱなしで、元気がよすぎた
それで偶数を送ると　その偶数+1の答が帰ってくるのであった。これはCLOCKが2MHzを入れてたためである。

ふげの声。CLOCKが2MHzのくせは、実は　じょーだんじゃなかっさ。そして CLOCKを　4MHzに直して LSBが立ちっぱなしだ　ったので ジャンパーもあったのでした。　こうして なんとか動いたのです。それを　そのことは 誰がすることもなく書いていた

* CONSはPage 7を参照

11月中頃　もう最近はだれも余裕のMZ80Bをさわってくれなくなっていた。しかたがないのでCLOCKに水晶を使おうと、4.9152MHzのクリスタルと8251並びにLS93を導電パットにさして、BOXへやって来た。

そしてまず8251をさし替えてみた。Takeo氏のTAKEというfile nameのプログラムを走らす。なんなんと、一発で動いた。Breakをして、もう一度走らす。やっぱりうまく走る。

とあとの2氏のもやってみる。どっちもうまく走る　ということで、どうやら8251をいろいろやっているうちにこわした？かもしれないのです。

現在MZ80Bは大きな余裕のある顔で、BOXのまん中におります。

'82 FALL

人工知能課より秋の原稿休載のおわび!!

Takeo:「人工知能屋はんだけ、ぜーんぜんゲンコー出てへんで」

㊐:「そらそやで　なんにもしてへんからなんにも書くことあれへんで!」

Takeo:「へっへ〜」と　ひれふす

1982年11月15日　タケ、BOXにて。

ちゃん　ちゃん

DATA SHOW '82 日記

▷ 10月21日に、はるみに 行った。

▷ @:『「ミューラックス」に 行ったら、森本サン おんのとちがうやろか』

▷ そして ミューラックスに行くと、なっ、なんと しっかり森本サンが いらっしゃったのでございましたノサ。

「TRS の PLOTTER・PRINTER 買うたで、かいらしいわっちゃで」

と、かわいくない声して おっしゃいました。

J・STAR by XEROX

さ、さすが XEROX

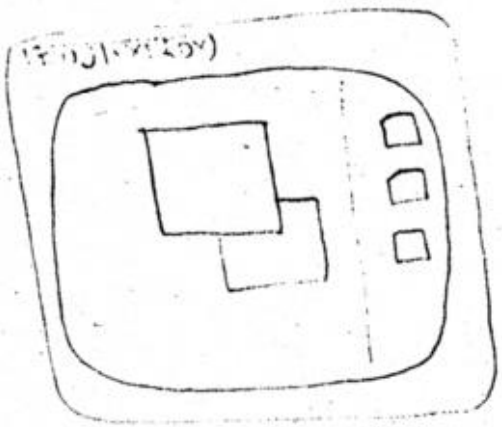
XINS とかいうとるが

"ALTO" "Dolphine" "DORADO"

の、伝統じゃ。

Display の上に Aicon が

おったのは、J-STAR だけじゃ。



○ ポインタ(カーソル)を動かす プロセスは、マウスと 直結してあり。そのポインタを参照するものは、別プロセスで うごいとったみたい。

○ WINDOW と設定するプロセスの わくの REFRESH が、おそく感じた。やっぱり、MultiProcess と Bit Map Display は、負担が大きいのだろう。それに、巨大な Soft も 負担にも かぶっているようだった。

○ Key Top は、上面に ひらがなが、書いてある。前面に ALPHABET が、書いてあった。

○ ポインタ(カーソルのこと)が うごきまわって、場所によって 形が 田に なったり ↖に なったり した。それで、今、何を指定できるのかを 表わしている。

1N9251

小売店におけるマイコンの役割

電子工回西 宏幸 西

最近いろいろな所にマイコンがはいってくるようになってきた。それにつれて、市販ソフトも多種多様になってきている。その中で、小売店用としては、販売管理、顧客管理、在庫管理などがある。これらはあくまでも、どのような店でもつかえるようにしてあるので、逆にいえば、どことなく、ものたりさを感じさせるものである。その為にやはり、自分の店にあうソフトがほしいものである。そこで、ソフトを組む前にまず考えることは使用目的である。ここで例として電機店を考える。電機店というのは、食料品店等と異なり、顧客が買う回数はたいへんに少ない。それだけに、余計に顧客の管理をしっかりとしなければならない。他に、小売店であるから当然のことながら、販売管理をしなければならない。今回は、この2つを主要な目的として考える。次に行なうことは、どれ位のデータをあつかうかである。市販のソフトであれば、ディスケットの容量が両面倍密で500名位である。しかしこれは1人のデータが256バイトとした時の話で

あるから，ここでも1枚に500名おさめられるとは限らない。もっといえば，顧客1人についてより詳しいデータをほしいのならば，より大きなデータ量となり，一枚当りの収容人数も減ってくる。そこでプログラムによって，ディスケットを管理する必要がある。その例として，顧客Noを通し番号として，ここではNとする。次に1枚に収容可能な人数をMとしたとき，Nの人のデータがはいっているディスケットは INT((N-1)/M)+1 となる。ここでNは1以上とする。こうして，ディスケットをわけることができる。しかし，使用者がまちがってちがうディスケットをいれれば，データをこわす可能性があるので，例として，上で与えられるディスケットの番号をDとした時に，ランダムファイルのファイルネームを "DATA"+CHR$(D) とすることによって，こわすこともなくなると思われる。プログラムを走らせて，読み出すのにそのファイルがなければ，エラーとなるので，プログラム中に ON ERROR GOTO というのをつかえば，もう一度ディスケットをいれるところにもどすことによって，データの保存ができる。

次に考えるのは，データ構造である。このようなプログラムにおいては，データが命であるから，よく考えて，データ構造をつくる必要がある。そこで問題となるのは，FIELD文の文型である。機

機によっては，レコード長が可変なものもあり，128，256バイト固定のものもある．例えば，128バイト固定ならばたとえ10バイトのデータであっても128バイトのメモリがつかわれる．以上のことをよく考えて，データをつくる必要がある．右の例は，顧客管理のデータ構造の例である。左の数字は必要メモリバイト数である。この例は256バイト固定用であるが，128バイト固定ならば，支店名までを№1，それ以後を№2とすれば同じことになる。ここで，取業区分はあらかじめ，数字できめたものとしている預金区分については，文字1文字できめている。できれば，銀行名，支店名も数字におきかえた方があとあと楽になることは確かである。又，最近は漢字も使用されるので，それように，128バイト位の別のデータをつくっても便利である。次に販売管理については，過去の履歴については，購入日及び，商品コ

1	読み出し可能か　不可か
2	顧客No　CVI
10	姓
3	名
1	性別
1	生年月日　元号　CVI
2	〃　年　〃
2	〃　月　〃
2	〃　日　〃
5	郵便番号
5	住所
1	職業区分　CVI
2	TEL　市外局番　CVI
2	〃　局番　〃
2	〃　番号　〃
2	請求日　CVI
2	クレジット期限　年　CVI
2	〃　月
14	銀行名　～銀行まで決める
10	支店名　店はいれない
1	預金区分
4	口座番号　前
7	〃　後
1	家族人数
8	家族1　名前
1	〃　性別
2	〃　生年月日
23	〃　職業区分
23	家族2
23	〃　3
23	〃　4
23	〃　5

ード，商品番号，購入個数，購入単価をおぼえておき，過去1年間については，前月までの未払金及び今月の振り込み金，そして各月の購入個数及び購入金額をおぼえておくのがよいと思う。しかし，ここで例にあげたのは電機店であった．．．を考えると，これではまだたりない。電機製品の購入のサイクルは普通年単位である。そのために，商品コードをつくる時に，あらかじめ，顧客の購入状況を知りたいもの前にもってくる。そして，商品コード1～32について，購入年，月をCVIを使って，1つ当り4バイト使って覚えておくと一層便利であると思う。以上のことより，データは大きくわけて，3つ．大きさにして，800バイトで1人のデータとなる。市販のソフトに比べて，大きなデータであるが，逆に，1枚当りの収容人数は減ってしまうので，先にのべたようにきちんと，ディスケット管理をする必要がある。プログラムの詳しいことについては，紙面のつごうで次回，機会があれば書いてみることとする。しかし，このようなプログラムで必要なことは必ずエラーチェックをする必要がある。それも何重にも。他のプログラムに比べて，この手のプログラムは素人の使用頻度が多いので。

TOM の自己紹介

現在、コンピュータークラブ部長の田中敏宏(たなかとしひろ)でいます。

誕生月日時　1961年 11月 23日(木) 9.20 J.S.T
(昭和36年) (1年の最後の祝日)
⇓
工機大と京大の学祭の日だ!

は Tom です。 Tom の由来は下のとおり。

高校3年生の時、高校のクラブ局 JA3YJS で、21.100 MHz 付近で Sweden の局と交信していた時、相手局の信号が fade-out しかかっていた。この時 handle name が伝わっていなかったのだが、Toshi, Toshi と何度送ってもわかってもSえなかった。それでやけくそになって Tom と送ったS.C.がにわかってもSえたので、それ以後 overseas では handle Tom と送ることにした。

数か月後、Philippine の局と交信していた時、handle Tom と送ると、相手局が不思議に思ったらしく、きき返してきた。よくきいてみると相手局の operator は日本人だったのである。

それまでは overseas のみ Tom であったが、以後国内でも [illegible] になってしまった。

ところで，現在　このクラブには　田中が　3人もいる。
そこで，区別するために，こう呼ぶことになっている。

工芸	電子	3	Tom
短大	電気	3	Katsu
工芸	電子	2	MASA

ついでに，田中　以外の　nickname も書いておく．

工芸	機械	3	竹岡	→ Take.
	電気	3	井	→ 井（字型が似てる）
			駒嵐	→ Koma!
	電子	3	梅垣	→ Ume サン
			奥田	→ Cordia TURBO，くせんせ，おくだ，おくーだ
			須藤	→ ㋜（すっぱまん）／ π（有名なπ4万ケタの作者）
			西	→ Carina（TOYOTA CARINAに乗れない）
	生産	2	高津	→ 天下御免（のハキホーダイ）
	電子	2	大中	→ MarkII（TOYOTA MarkIIに乗ってる）
			渦原	→ @（@は渦を連想させる）
繊維	S	2	川端	→ Rina（なまえじゃ．）

なお，他にもあるが　省略．

上に書いたものでも，定着していないものもある．

むき　2　中島 → かず ちゃん〔XL みたら 石投げよう〕

editor注：この記事は．1982年末 [illegible] on Date(なけ)．

巻末言

KANTOKU

LINE Vol.2. 発行おめでとう。各チームがそれぞれ頑張って、活動が地に着いて来たのが感じられる紙面である。学園祭の数日前に印刷が終わっているというのもすばらしい。編集諸氏の尽力と部員の協力の賜物である。

Chief Editor 氏の説くように、ネットワーク，データベース，数値制御（NC），パターン認識，人工知能といった10年前には，アマチュアにはちょっと手の届かなかったテーマが，LSI革命と学術研究の成果の恩恵と，そして，それを受け入れる社会背景によって，マイクロコンピュータシステムの今後の重要な方向となるのは間違いない。そのためには，しっかりしたオペレーティングシステム，使用目的に合ったプログラミング言語，およびデバッグツールなど，ソフトウェアの充実が必要である。ハードウェアの限り無い高密度化とspeed upと機能充実の革命に遅れないためには、出来るだけmachine independentなプログラムの開発が必要であり．レジスタ配置やステップ数を極限まで最適化する職人芸より，データ構造とアルゴリズムの質が重視される。煩わしい最適化はコンパイラにまかせて，プログラムのモジュール化と拡張性，そして何よりも読み易さ（readability）を大切にしよう。

さて、LIMEの記者になってみるも、いくつかの苦情と希望を述べて今後の改良を促さないわけにはいかない。

(1)散文が多くて、少々うんざりする。苦労談と成果が混在して、やったことが不明瞭でアピールしない。感想だけで、具体的な事が何も分らない記事もある。

(2)ブロック図を示して、話のあらすじと結論としての成果を先に述べよう。

(3)学園祭で配布することを考えれば、もっと図表を中心とした記事が望ましく、展示内容をこの紙面を使って詳しく説明して欲しい。図表は定規を用いよ。

(4)箇条書きを多用せよ。

(5)話題を類別して、いくつかの章に分ければ読み易い。

(6)ローカルな話題を全編に散りばめる前に、各プロジェクトチームの紹介をするのが良いだろう。

最後に、LIMEも、学園祭の時期ばかりではなく、年に何回か発行して、部の活力となることを期待して筆を置く。