エンジン集中電子制御システムのROM解析

吉田　立・新海　日出夫*・溝口　貴彦#

1．序　　論

最近の自動車は多機能化が進み，例えばABS(アンチロックブレーキシステム)とか，エアバッグ装置とか，いろいろ新しい機能が付加されるようになってきた。これらの機能を具合よく運用するためには,各機能ごとにマイクロコンピュータ(マイコン)を用いた制御回路が使われている。

一般にマイコン回路は，センサからの信号を取り込む(入力)部分，信号を変換・計算処理などして判断する部分，結果を出力する部分(アクチュエータ)に分けられるが,２つ目の部分は，働きとしてさらに，演算機能，制御機能，記憶機能に分けて説明されることが多い。

回路素子として見ると，演算と制御機能は中央処理装置(CPU)と呼ばれるICに組み込まれ，記憶装置(いわゆるメモリIC)にはこれらのための命令や必要なデータが格納されている。さらに，入出力信号を外部とやり取りするための，いわゆるインターフェースが必要となる。これには，弱い信号(電圧・電流)を増幅する増幅器やアナログ信号をディジタル信号に変換するA/D変換器なども含まれる。最近では，半導体の高密度化が進んで，CPUとメモリが一体となったICが増えてきている。

われわれは，こういったマイコン回路の働きをより細かく理解するために，メモリに書き込まれているデータを解析することを試みた。対象とした回路としては，エンジン集中電子制御システム(以下，ECCSと呼ぶ)を取り上げた。これは日産自動車によって1979年に開発され，その後の日産車に搭載されており，燃料系・点火系を一括してきめ細かく制御することが可能となり，排気ガスの排出量の規制をクリアできるようになった。ECCSでは，エンジン各所のセンサ類からの入力信号をもとにコントロール・ユニット(以下，C/Uと呼ぶ)と呼ばれるマイコン回路が最適な出力信号を出力し，エンジンが制御されている。この出力信号を決定しているのがC/U内のCPUであり，決定に必要なデータは，CPUへの命令と併せて，ROMと呼ばれるメモリの中に記憶されている。

自動車は一定条件で使用されるわけではなく,様々な道路状況や色々な走行状態で使用される。これに対応するようにエンジン制御を行うためには高性能なCPUと多大なデータが必要だと思われる。例えば，燃料噴射時間を算出するためにはエアフローメータからの吸入空気量とエンジ

*,# 専攻科３期修了生，＊現：アイシン精機，＃現：岐阜ダイハツ

ン回転数が必要になるが，回転数が一定であることなどは一部の場合を除き，まずない。吸入空気量も負荷に応じて変化する。このような各状況に対応して噴射時間をどのように決定するのか，またこれに合わせて点火時期をどう制御しているのかなど，電子制御・メカトロニクスを研究しているものにとっては非常に興味があるところである。

しかしこの内容についてはメーカにとってノウハウの固まりであり，もちろん公表されていない。そこで，ECCSの制御の仕組みを以下の点について調べた。

・C/Uの構成

・プログラムの構成

・ルックアップテーブルの内容

またルックアップテーブル(プログラム上参照すべきデータの配列の部分,エンジン制御の言葉で言い換えるとマップ)のデータを変更することで,エンジン特性がどのように変わるのかを調べることで，制御に関わったデータであることの確認作業を行った。

2．概　　説

今回，解析の対象としたのは，日産セフィーロに搭載されていたエンジン RB20E のECCSである。昭和63年式の旧型であるが，これを対象に選んだ理由は以下の４点による。

・ROMが，C/Uの中で独立した素子として存在し，比較的容易に取り外すことができる。

・展示模型として，専攻科１期生の井藤賀氏(現，本学職員，ジーゼルエンジンＩ所属)により，同一車種について，ECCSに関係するすべての素子，配線類を取り外したものがある。必要な素子については特性が測定してあり,走行中の状態に相当する疑似信号を発生できるようにしてある。[1)]

・このために，実際に自動車を走行させる必要がなく，事故の危険がない。

・旧型のために，解析データを改造目的などに悪用される危険性が低い。

１）ECCS(Electronic Concentrated engine Control System)

図１にECCSのシステム図を示す。[2)] ECCSの制御内容は，基本的には，次の５種類である。

・燃料噴射制御（空燃比）

・点火時期制御

・アイドル回転数制御

・フューエルポンプ駆動制御

・自己診断

これらの制御をするために，図１のように，いろいろなセンサからの信号が入力されている。アナログ信号は後述のUPP内蔵のA/Dコンバータでディジタル信号に変換された後,CPUに入る。

２）コントロール・ユニット

図２にC/Uの全体図を示す。図中の主なIC(①～④)とその機能は次のようである。

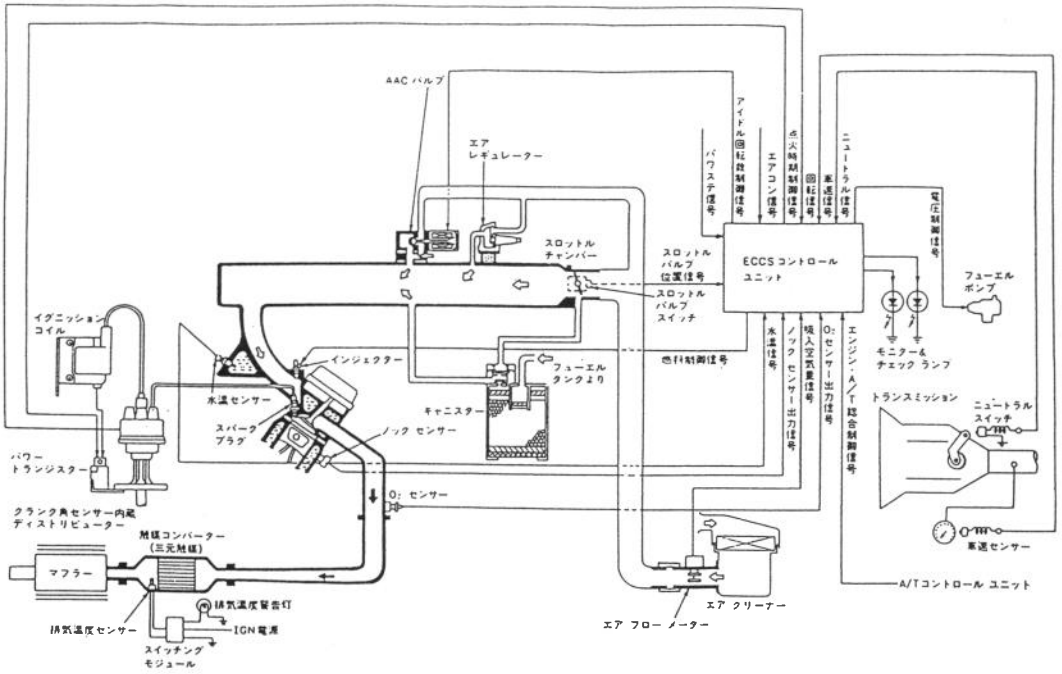

図1 日産ECCSのシステム概略図

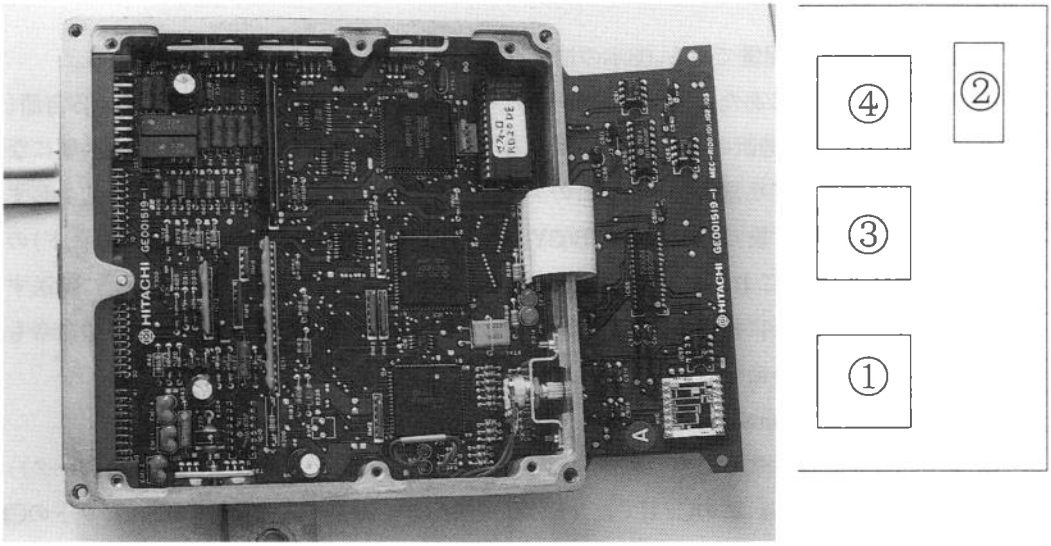

図2　コントロールユニット(C/U)基板の全体図とメイン基板上のICの配置(右)

①CPU(Central Processing Unit)

CPUは，C/Uの頭脳と言うべき制御の中心となるICである。このC/UのCPUは「HD63B03YCO(日立製)」というMPU(Micro Processing Unit)が使われていた。

このMPUは，高性能CMOS (Complementary Metal Oxide Semiconductor) 8ビットマイコンで，CPUのほか，192バイトのRAM(Randam Access Memory)，3チャンネルのパラレルI/O(入出力)端子などを内蔵している。[3)]

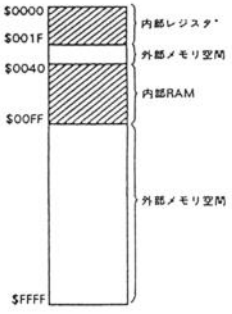

図3　MPUのメモリマップ

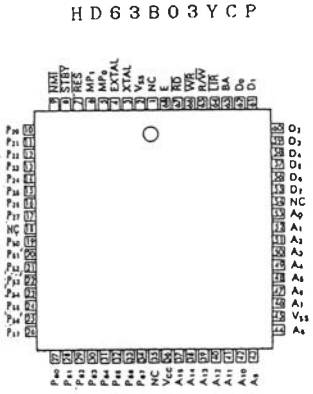

図4　MPUのピン配置

P20～P27, P50～P57, P60～P67は入出力ポート。

図3にMPUのメモリマップを示す。斜線の部分がMPU内部のメモリを使用している。空白の部分はROMや外部RAMに対応する。左に書かれた0000～FFFFの数(16進数)がアドレスで，$マークは16進数で記述されていることを意味する。以下の本文では，数のあとにhを付けて表す。アドレスの最下部に割り込みベクトル領域がある。割り込みベクトルに関しては後に詳しく説明する。

図4はMPUのピン配置である。68本の端子を持つフラットパッケージ型をしている。

プログラムを書くにあたって，CPUが理解できる言語(機械語)と我々が理解できる言語(日本語，英語など)の間に通訳の役割をする(簡単な文法・限られた言葉を使った)言語が必要になってくる。これをアセンブリ言語と呼ぶ。このアセンブリ言語は対象となるCPUによって少しずつ違うが，今回の解析の対象である「HD63B03YCO」用のアセンブリ言語の命令(オペコード)の一覧表を図7(b)に示す。これらの言葉は，ある場所に格納してあるデータを別の場所に転送するといった具合に，目的語に相当する部分(オペランドと呼ぶ)を含んだ3～4バイト長の命令もあれば，停止といった短い命令もある。

②ROM(Read Only Memory)

ROMはCPUを動作させるためのデータ(プログラムやルックアップテーブル(数値データ))を記憶させておくICである。このC/Uでは「HN27C256A-20-XGT(日立製)」の256kビットのCMOS UV-EPROMが使用されている。15本のアドレス信号の入力ピン，8本のデータ出力線，その他の制御線端子を持つ28ピンのROM ICである。高温度対応型となっているのは，車載を意識しているためかと思われる。256kビットは32kバイトであり，32kバイトのアドレスを16進数で表すと0000h～7FFFhになる。

③UPP(Universal Pulse Processor)

UPPはCPUの補助をする周辺ICである。このUPPはHD63140(日立製)でユニバーサルパルスプロセッサ(パルス信号を発生する),10ビットのA/Dコンバータ,1024バイトRAM，ウォッチドッグタイマの4つの独立したモジュールをワンチップ化したCMOSのICである。図5にピン接続図を掲げる。[3)]

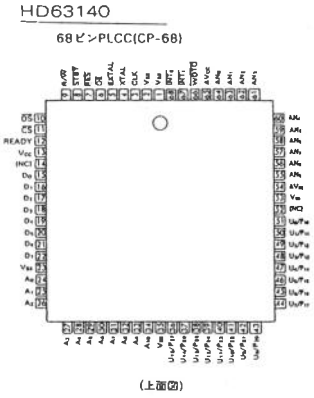

図5　UPPのピン接続図

④ゲートアレーロジック

ゲートアレーロジックとは論理回路の集まりで，このICは，CPUのアドレス信号を実在のICのチップセレクト信号に変換する，読み書きのタイミング信号をつくるなど，入出力制御信号の変換を司っている，いわばCPUの秘書のような役割をする。(最近のパソコンでは，チップセットと呼ばれ，この性能が総合性能に大きな影響をするとも言われている。)このC/UではD65022L(NEC製)が使われている。NECに尋ねたところ，カスタムメイドであり，日産との関係で教えることが出来ないということで，どのような回路になっているのかは不明である。このために，アドレス解析の障害となった。

3．逆アセンブル

1）逆アセンブルとは

通常のプログラム開発ではプログラムをまずアセンブリ語あるいは高級言語（C，BASICなど）で作成し，それを機械語に変換する。このことをアセンブルという。

しかし今回行うのは，すでにメーカの技術者によって作成され機械語に変換されているプログラムとデータを区別して，前者は元のアセンブリ語に戻して内容を解析することである。この作業のことを逆アセンブルという。

逆アセンブルを行うには16進数で書かれているデータをオペコードマップに対応させて変換し，その他の文法にもあわせて一つの命令に組み立てる。この作業をプログラムの開始点から一つ一つ行っていくが，データの中には命令だけでなく，計算に必要なデータも含まれているので，区別しなければならない。また，1バイトずれて解読を始めても，それらしいアセンブラ語ができあがるので，内容を見ながら，解析をしなければならない。具体的な作業内容を次章に示す。

2）割り込みベクトル

プログラム(命令)の開始点のアドレスは割り込みベクトルを調べることで見つけることができる。割り込みとは，卑近な例で例えると，我々が通常の仕事をしているときでも，例えば電話が鳴ったり，来客があったりすると，そちらを優先して処理せざるを得ないことがある。ちょっとだけ応対して，相手を待たせることもある。このように，CPUの動作においても，入力信号に変化があったときなど，その変化に対応して別の処理プログラムを実行する必要があり，このこと

優先順位	ベクタ MSB	ベクタ LSB	割り込みの種類	ROMデータに記載のアドレス
最上位	FFFE	FFFF	$\overline{RES}$	8E00
	FFEE	FFEF	TRAP	A14A
	FFFC	FFFD	$\overline{NMI}$	A3E3
	FFFA	FFFB	SWI (Software Interrupt)	A14A
	FFF8	FFF9	$\overline{IRQ_1}$	A14A
	FFF6	FFF7	ICI (Timer 1 Input Capture)	A18B
	FFF4	FFF5	OCI (Timer 1 Output Compare 1,2)	A374
	FFF2	FFF3	TOI (Timer 1 Overflow)	A3C6
	FFEC	FFED	CMI (Timer2 Counter Match)	A3E3
	FFEA	FFEB	$\overline{IRQ_2}$	A468
最下位	FFF0	FFF1	SIO (RDRF + ORFE + TDRE)	A5FF

図6　割り込みの種類とベクトル
右に示したのは解析したROMに書かれていたアドレス

を「割り込み」があったという。

割り込みベクトルにはそれぞれの割り込みの時に実行すべきプログラムのアドレスが指定されている。図6に今回のCPU割り込みベクトルのメモリマップを示す。割り込みの種類は11種類あり，それぞれに優先度が付いている。最優先で割り込むのは，リセットによる割り込みで，エンジン始動時の場合に相当する。走行状態の変化によるセンサーからの信号の変化はNMI割り込みを通して取り込まれ(Non Maskable Interrupt)，CPUはどのセンサー出力からの割り込みかを調べて，その変化に対応する。

ベクトルのアドレスは512 kビットのROMに対応しているが，今回のROMは256 kビットなので最上位の1ビットを無視して，アドレスの変換を行った。(例えば，リセットRESのベクトルアドレス：8E00hは0E00hと読む)

4．解　析　作　業

ROMの解析にいたるまでの作業を，順に説明する。

1)ROMの取り外しとデータの読み取り

C/Uの基板に半田付けされているROMを取り外した。(両面基板にしっかりと付けてあるので，半田吸い取り器などを使って，丹念に取り外す必要がある。)

これの内容を，ROMライタで，読み取った。ライタ(writer)は，本来作成したデータをROMに書き込む装置であるが，ROMのデータの読み出しもできる。本実験では，(株)アーバルデータのPECKER 11を使用した。RS-232Cを通して，パソコン側からすべての操作が可能であり，データの転送も行える。この操作については以前に報告がある。[4)]

2) ヘキサファイルのバイナリー変換

データの保存形式の違いがあり，これを変換しなければならなかった。ここでは，(株)システムロードの変換ソフトHBを用いた。

有効なデータは，バイナリー形式で，32768バイト(32 kバイト)の大きさがあった。ただし，データでない空白部分は3 Fhで埋められていた。(この部分は，図8では，斜線の部分に相当する。)

3)逆アセンブル

逆アセンブルの作業の概略を，今回のプログラム部分の先頭部を使い，図7(次ページ)で説明する。上部(a)の機械語のデータを，中間部(b)のオペコードマップに基づいて，下部(c)のアセンブリ語のプログラムに変換する。例えばアドレス0E00hに書かれた8EhというデータはLDS(メモリーのデータをスタックポインターというレジスターにロードする)という命令と理解でき，この命令は文法から次に2バイトのオペランドが伴ってくることが分かり，アドレスの0E01h以降の07，FFはメモリー内のアドレスを示す07FFhとなる。

この作業は膨大であるので，パソコン上で「8ビット対応逆アセンブラ ZZ ver .3.0」(以下，ZZ)というソフトを使って行った。開発元はマイクロオーグ，発売元はアクエリアスである。[4)]

このソフトは，8ビットで表される機械語(2^8=196種類)とアセンブリ語との対応表に基づいて，逆変換を行っていく。これには「6303」の変換テーブルが無かったので類似のCPUである「6809」の変換テーブルをもとにして「6303」の変換テーブルを作った。このためか解析がうまく行かないところもあった。

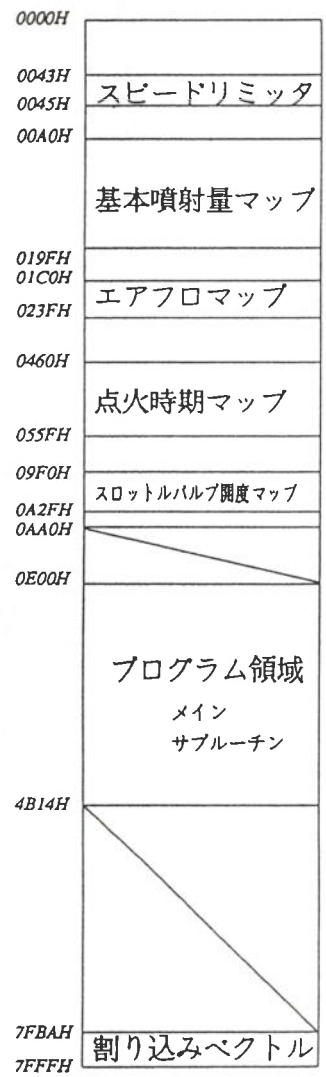

図8　プログラムの構成(ROMマップ)

ZZを起動させ初期設定を行う。まず自動解析モードをZZに行わせる。しかしすべてのデータがDB(Define Byte)(数値データ)として扱われてしまった。そこでこれをプログラムに変換する作業を行う。前項で説明した割り込みベクトルからプログラムの開始点を探しそのアドレスから命令に変換する解析を始めた。一つ一つ解析していくと，ジャンプサブルーチン(以下，JSR)という命令が出てくる。これは，別の作業を行って戻って来い，と言う命令なので，そのサブルーチンの開始アドレスに行き解析を行い，リターン(RST)まで行ったら元のアドレスに戻り，メインルーチンでの解析を続ける。サブルーチンの中にもJSRがあるのでその場合も同様のことをする。この作業をプログラムらしきものが出来上がるまで何度も繰り返す。

プログラム“らしきもの”と書いたのは，データを1バイトずらして解析してもそれらしいプログラムが逆変換でできることがあるからである。しかし実際には，内容がどこかおかしなプログラムになっていたり，オペランドがデタラメになってきたりするので、しばらく解析しているとそれが分かる。そのため，今までの作業を最寄りの確実な開始点からやり直し，プログラムの

(a)

```
00000E00  8E 07 FF 86 13 97 1B 86-41 97 03 86 72 97 01 86
00000E10  00 97 21 86 6E 97 15 86-00 97 20 86 AA 97 14 86
00000E20  C0 97 17 86 F6 97 16 86-C2 B7 20 00 36 86 1C 97
00000E30  08 86 03 97 0F DC 09 C3-00 09 DD 0B DC 09 C3 00
00000E40  09 DD 19 7F 05 12 BD 92-4D 32 97 41 B6 01 56 B1
00000E50  86 0B 25 0C 7B 02 C4 26-07 B6 01 0B 81 C8 25 03
00000E60  7F 04 8F 4F CE 01 FF A7-00 09 8C 00 41 26 F8 86
00000E70  C2 97 41 FE 05 10 8C AA-55 26 23 B6 86 A0 85 10
00000E80  26 1C CE 03 00 B6 86 AB-5F A3 00 25 11 B6 86 AC
00000E90  5F A3 00 22 09 08 08 8C-03 80 25 E9 20 03 BD C5
00000EA0  3C FC 04 80 DD 89 FC 04-82 DD 8B B6 04 84 84 EF
00000EB0  97 8D B6 04 8F 85 80 27-03 7C 04 90 B6 04 90 B1
00000EC0  86 0E 25 06 72 40 8D 7F-04 90 86 40 B7 10 00 4F
00000ED0  CE 00 6A 16 58 3A B7 10-01 F6 10 02 C5 40 26 F9
00000EE0  C4 03 E7 00 F6 10 03 E7-01 4C 81 08 25 E2 DC 6E
00000EF0  B3 05 13 24 02 4F 5F DD-CB 86 58 BD CA 80 4D 27
```

(b)

オペコードマップ

OP CODE						ACC A	ACC B	IND	EXT / DIR	ACCA or SP				ACCB or X				
HI		0000	0001	0010	0011	0100	0101	0110	0111	IMM 1000	DIR 1001	IND 1010	EXT 1011	IMM 1100	DIR 1101	IND 1110	EXT 1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0		SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS			AIM		CMP								1
0010	2			BHI	PULA			OIM		SBC								2
0011	3			BLS	PULB	COM				SUBD			ADDD					3
0100	4	LSRD		BCC	DES	LSR				AND								4
0101	5	ASLD		BCS	TXS			EIM		BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR					STA		STA					7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI			TIM		ADD								B
1100	C	CLC		BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC		BLT	MUL	TST				BSR	JSR				STD			D
1110	E	CLI		BGT	WAI			JMP		LDS				LDX				E
1111	F	SEI		BLE	SWI	CLR					STS				STX			F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

(c)

```
                                ;*** reset start
0E00  8E 07FF           CI_0E00:        LDS     #$07FF          ; ...
0E03  86 13                             LDA     #$13            ; ..
0E05  97 1B                             STAA    $1B             ; ..
0E07  86 41                             LDA     #$41            ; □
0E09  97 03                             STAA    $03             ; ..
0E0B  86 72                             LDA     #$72            ; □
0E0D  97 01                             STAA    $01             ; ..
0E0F  86 00                             LDA     #$00            ; ..
0E11  97 21                             STAA    $21             ; .!
0E13  86 6E                             LDA     #$6E            ; □
0E15  97 15                             STAA    $15             ; ..
0E17  86 00                             LDA     #$00            ; ..
0E19  97 20                             STAA    $20             ; .
0E1B  86 AA                             LDA     #$AA            ; □
0E1D  97 14                             STAA    $14             ; ..
0E1F  86 C0                             LDA     #$C0            ; □
0E21  97 17                             STAA    $17             ; ..
0E23  86 F6                             LDA     #$F6            ; □
0E25  97 16                             STAA    $16             ; ..
0E27  86 C2                             LDA     #$C2            ; □
0E29  B7 2000                           STAA    UD2000          ; ｷ.
0E2C  36                                PSHA                    ; 6
0E2D  86 1C                             LDA     #$1C            ; ..
```

図7　逆アセンブルの流れ(a):機械語のデータ,(b):オペコードマップ,(c):アセンブリ語のプログラム

つじつまが合うまで何度も繰り返して解析しなければならなかった。さらに，どうしてもプログラムに戻すことのできない領域があり，これらはルックアップテーブルだと予想されたが，どの機能のためのマップであるかの判断がここだけではつかなかった。

この作業を繰り返し行い出来上がったプログラムは，ルックアップテーブルも含めるとA４で300ページにもなった。このプログラムの構成を図８に示すメモリマップに示す。プログラムは0E00hから始まり4B14hまでの17.2kバイトで，この中には無限ループになっているメインプログラムのほかにサブルーチンが102個ある。およそのプログラム内容は読みとれるが，入出力アドレスと実際の入出力端子との関係がゲートアレイICを介しているので，詳細までの解析は進んでいない。

ルックアップテーブルは,0000hから0A9Fhの2.7kバイトである。この中には16行16列が１固まりとなったようないわゆるマップデータがいくつか含まれているが，同じ文字が数個連続したような意味不明のデータも多い。そして割り込みベクトルが最下部の7FBAhから7FFFhにある。

４)ROMマップ内のデータの解析

ルックアップテーブルのデータは，センサーの出力信号を実際の値に変換するための換算値が入っていたり，エンジン状態に応じて最適の燃料噴射量(時間)，点火位置の情報などが与えられていたりする。この区別をするには，実際にプログラムを実行し(エンジンを掛けた状態にし)，マップ内のデータのどこの部分をアクセスしたか，それによりどう出力が変化したか，など総合的に調べる必要があった。このために，展示模型とROMエミュレータを使用した。

ROMエミュレータ

ROMエミュレータは，名前のようにROMの真似をさせる という装置で，本体はRAMとパソコンからの制御により動くサブCPUから成っている。エミュレートすべきROMを取り外して，これから信号線を引き，ROMの代わりを本体のRAMにさせるわけである。本来ならばROMに書かれるデータをパソコンからRAMに転送しておいて，起動するとCPUはこのRAMデータにアクセスして，ROMの時と同じように作動する。このアクセス状況をサブCPUが監視していて，パソコンの画面上でそのデータの色を変えて表示してくれる。またRAMのデータ内容をリアルタイムで変更させることもできるすぐれものである。プログラムのデバッキングの最有力の方法であるが，本実験で使用した(株)プリズムのROMSCOPEは，さらに自動車での利用を意識した製品であった。

エミュレータ本体は，DC12V(シガーライタ)で作動し，電源ユニットには，大きな電解コンデンサが組み込まれ，エンジン始動時の電圧低下にも対応している。パソコンとは，RS-232Cで接続され，付属するソフトウェアにより通信を行う。ソフトの起動画面の一例を図９に示す。*

---

*マニュアルの最初に「一般公道で絶対に使用しないでください」とある。

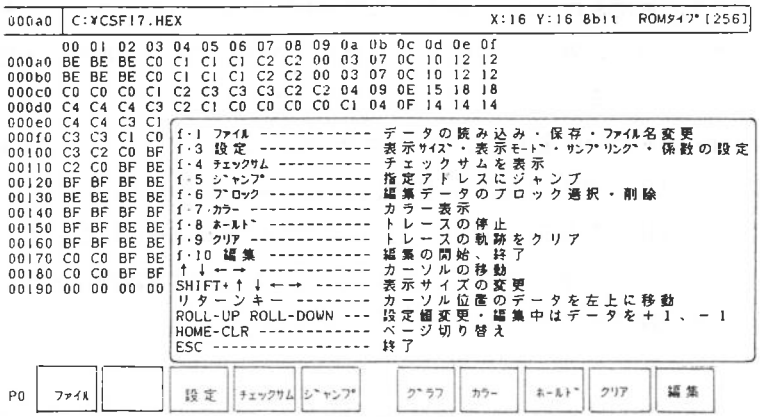

図9　ROMSCOPEの起動画面

ヘルプキーを押して，ファンクションキーの説明を表示している。背景は00A0hから始まる基本噴射量マップのデータ。

確認作業

C/U内のROMのあった場所にICソケットを半田付けし，ソケット穴を使ってROMエミュレータとの接続線を接続した(図10参照)。パソコンからROMエミュレータにプログラムを転送したのち，展示装置を起動した。

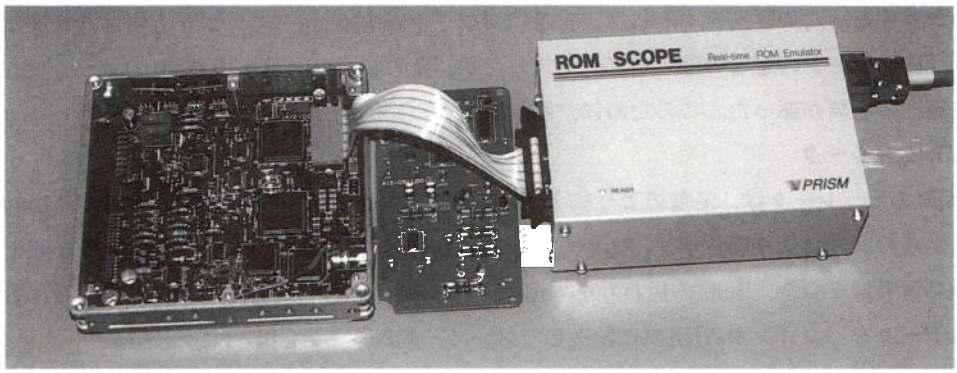

図10　ROMエミュレータ
コントロールユニット(左側)とフラットケーブルで接続したところ

展示模型でエンジン回転数やエアフローメータの電圧を変更しながら，ROMデータのアクセス状況を調べたところ，16進データ表にある，アドレス00A0h～019Fhや，0460h～055Fhが，特にアクセスが多いことがわかった。また特に00A0h～019Fhは回転数に，0460h～055Fhはエアフロ電圧にすばやく反応していた。このアドレス部分をROMエミュレータの付属機能にあった3次元グラフ表示でグラフ化したのが，図11と図12である。この2つのグラフを比較すると，図11のグラフは，高回転，高負荷になるほど数値が大きくなるのに対し，図12のグラフは高回転，低負荷で数値が大きいことがわかり，前者が基本噴射量マップのグラフ，後者が点火時期マップのグラフであるとみることができた。[5)]

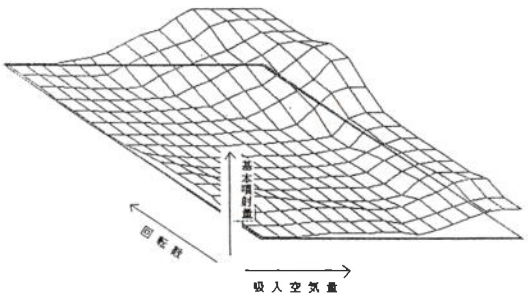

図11　基本噴射量マップのグラフ

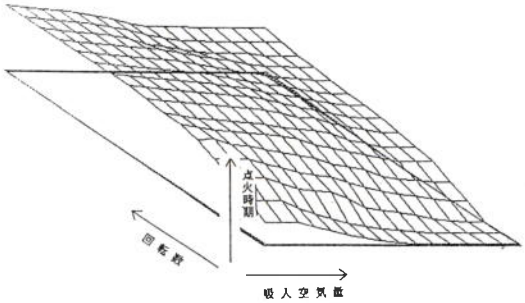

図12　点火時期マップのグラフ

これを確認するために，例えば，基本噴射量マップの数値を変更させた時の燃料の噴射時間を，オシロスコープにより測定した。実験は所定の回転数に合うようにクランク角センサーをモータで回しておき，エアフロ電圧は低電圧発生器からの疑似電圧を入れた。データの数値をノーマル，10%増，20%増，30%増，10%減，20%減にしたときの燃料インジェクタ弁の開時間(噴射時間)を測定した。得られたデータの一例を図13に示す。点火位置についても，クランク角の120°パルス信号からのズレ時間を測定することで確認できた。

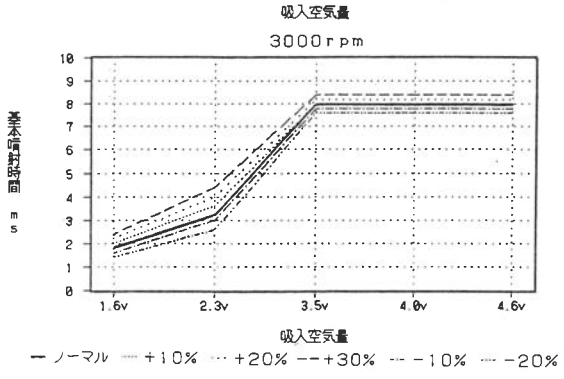

図13　燃料噴射量を加減したときの基本噴射時間の変化

また，車速センサ入力部にファンクションジェネレータから疑似波形信号を送ることにより，スピードリミッタの作動を確認した。回転数を3000回転，エアフロセンサの出力電圧などには走行中に相当する疑似電圧を加えておいて，16進数データ表のうち0045hの数値を変更した。表1に示す周波数以上のとき，燃料噴射がカットされてしまうことが分かった。また，0043h，0044hのデータは，リミッタによって完全に燃料カットされる以前に予備的にいくぶんか燃料を減らす領域の設定に関係する数値であることも確認された。

5)基本噴射量マップを変更させたときのエンジン出力変化

実験車両をシャシダイナモメータに乗せ，基本噴射量マップの数値を変更させたときのエンジン出力の変化を調べた。4)と同じく，データの数値をノーマル，10%増，20%増，30%増，10%減，20%減にしたときを測定した。

16進数データ	10進数	リミッタが作動したときの周波数(Hz)
5 A*	90	68
6 4	100	70
3 C	60	44
1 E	30	22

表1　スピードリミッタを変更したときの作動結果　＊は既定のデータ値

結果については図14に示す。燃料を多くすると，10%増では出力が増加したが，それ以上になると，出力が減少した。逆に，燃料を少なくすると，出力が増加した。これは，燃料噴射量に合わせて点火時期などの補正をしていないことによると思われる。

データの数値を20%以下にしようと試みたが，アイドリングが不調になり危険と思われたので中止した。

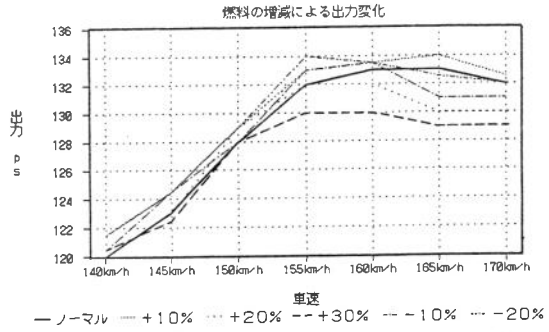

図14　燃料噴射量を加減したときのエンジン出力の変化

5．結　　論

ROMデータを静的解析では，逆アセンブラによりルックアップテーブル領域とプログラム領域に分けることができ，さらにプログラム領域を解析することができた。また，動的解析として，ROMエミュレータなどを使用することにより，いくつかのルックアップテーブルの存在が確認できた。まとめとして図8のROMマップが得られた。

6．所　　感

これまでの実験では，データ解析が主となって，確認作業は燃料の増減のみしか行っていない。併せて点火時期を変化させれば，出力変化の結果はまったく変わってくると思われる。今回は燃料を増やすと予想に反して出力が低下したが，点火時期マップのデータなど色々なデータを調整すれば燃料増にしたとき一層の出力増加が望めるだろう。さらに，排気温度の変化や，NO の量なども併せて調べ，ノッキングの具合もチェックしながらデータ変更を行えば，最高出力を向上

させることもできそうである。また，逆に燃費向上を目指してチューニングすることもできる。これらの試みを、今年度の専攻科生と行っているので、成果が出れば続報したい。

最後に，最新の日立製のエンジン制御用MPUについて。97年春に発売されたMPU(SH7051)は,32ビットRISCのCPUを持ち，乗算等の計算機能が強化され，メモリもフラッシュメモリ 256 kBを内蔵している。さらにA/Dコンバータ，点火・燃料噴射制御のためのタイマ，パルス発生器，果てはパワートランジスタまで内蔵していて，これ1個あればプログラムさえ組めば8気筒エンジンは回ってしまう。CPUパワーが大きいため，これまではルックアップテーブルの値を内挿してデータを近似計算していたのを，現代制御理論を駆使して，最適なデータを各入力データを元に瞬時に計算して出してしまおうという。それでも，CPUパワーの10%を使っていないという(残りは自己診断などに使う，でも余る)。実際に搭載車を走行させながら，計算式に必要なパラメータを決めて(そのための開発ソフトも準備済み)しまいましょうといわれると，我々は手出しができなくなる。

7．謝　　辞

太平洋工業株式会社の大久保陽一さん，日立製作所中部支社の山本栄治さんにはMPUの資料などを送っていただいた。本学実習室の吉田豊彦さん，佐藤幹夫さんには，実験車両の貸与，シャシダイナモメータの使用などで便宜を図っていただいた。この場を借りて，御礼申し上げます。

参 考 文 献

1) 井藤賀久岳：中日本自動車短大 専攻科修了研究報告書，1995
2) 日産自動車サービス週報，B-6，1988
3) 日立半導体マニュアル，HD6303X，HD63140の項
4) 福井　稔，岡田俊治、及川浩和：中日本自動車短大 論叢，vol.24，p.15，1994
5) 浅原　宏：電子制御エンジンの基礎・応用，第4章，第5章，CQ出版，1995