[第1章]

マイコンのソフトウェア開発

アセンブリ言語とコンパイラとラダー図

1-1　ソフトウェア開発の環境

マイコンが開発された当初はビット列である機械語(マシン語)しかありませんから，だれでもその数字をROMに直接書くか，機械語と1：1に対応するアセンブリ言語でプログラムを書き，それをアセンブルすることによって得た機械語をROMに書き込むことによってマイコンを動作させていました．

● アセンブリ言語とその開発環境

図1-1はCPUとしてZ80を使用し，8255という8ビット単位の入力/出力のラッチ(保持)素子を使って，押しボタン・スイッチによってLEDを点灯する回路の構成を示しています．SW_1がONでLED_1を点灯させ，SW_2がONでLED_2を点灯させるとします．

仮に上部の8255のアドレスを30 hとし，下部の8255のアドレスを32 hとします．押しボタン・スイッチSW_1，SW_2，およびLED_1，LED_2は，おのおのそのアドレスのビット0とビット1に接続しています．

Z80の命令をアセンブリ言語で表すと**リスト1-1(a)**のようになります．

CPUは各種レジスタをもっていて，A(アキュムレータ)，B，Cなどの名前が付けられています．**図1-2**

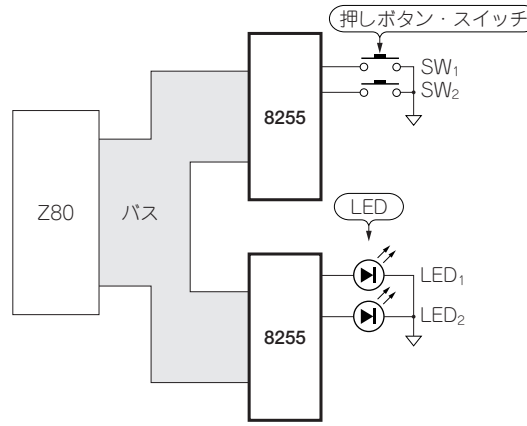


図1-1　Z80に2個の8255でスイッチとLEDを接続する回路のブロック

リスト1-1　8255を使ってスイッチでLEDを点灯させるZ80のアセンブリ言語プログラム

```
AA0：
    LD   B, 00      ; レジスタBを00にする
    IN   A, (30h)   ; レジスタAに30hのアドレスの内容を入れる
    LD   C, A       ; レジスタCにその値をキープする
    AND  01h        ; ビット0とANDを取る
    JR   NZ, AA1    ; 0でなければラベルAA1のところにジャンプ
    LD   A, 01h     ; 0（スイッチON）だったらレジスタAを01にして
    OR   B          ; Bのビット0を1にする
AA1：
    LD   A, C       ; キープした値をレジスタAに戻す
    AND  02h        ; ビット1とANDを取る
    JR   NZ, AA2    ; 0でなければラベルAA2のところにジャンプ
    LD   A, 02h     ; 0（スイッチON）だったらレジスタAを02にして
    OR   B          ; Bのビット1を1にする
AA2：
    LD   A, B       ; レジスタBの値をレジスタAに入れる
    OUT  (32h), A   ; レジスタAの値を32hに出力する
    JR   AA0        ; ラベルAA0のところに戻って同じ動作を繰り返す
```

（**a**）アセンブリ言語のソース

```
                ; AA0：
06  00          ;      LD      B, 00
DB  30          ;      IN      A, (30h)
4F              ;      LD      C, A
E6  01          ;      AND     01h
20  05          ;      JR      NZ, AA1
3E  01          ;      LD      A, 01h
B0              ;      OR      B
                ; AA1：
79              ;      LD      A, C
E6  02          ;      AND     02h
20  05          ;      JR      NZ, AA2
3E  02          ;      LD      A, 02h
B0              ;      OR      B
                ; AA2：
78              ;      LD      A, B
D3  32          ;      OUT     (32h), A
18  E9          ;      JR       AA0
```

（**b**）機械語

図1-2　8080Aのレジスタ群
（8080AはZ80の前身とされる）

はZ80の前に普及していた8080Aのレジスタ構成です．

リストでセミコロン（；）以降はコメント（説明文）で，実際の機械語変換のときは無視する約束事になっています．これを機械語（16進数表記）に直すと**リスト1-1**（**b**）のようになります．

● 初期設定などのプログラムが必要

CPUの中のALUはこの数値（命令ビット列）を解釈して実行するわけですが，もちろん0番地から立ち上がってこのプログラムまでたどりつくことが必要です．ジャンプ命令が途中になければ，例えば300番地にこの命令があるとすれば，0からインクリメントして順調に300番地のこのプログラムまでたどりつくことになります．

アセンブリ・プログラムは単純な命令の集合なので，**リスト1-1**のプログラムは初心者でも，右側のコメントを読めば，ある程度の命令の意味が理解できるはずですし，もし手元にZ80のアセンブラに関する説明書があればこれを完全に理解することが可能です．

そして少し慣れれば，誰でも自分でアセンブリ・プログラムを書いてみることができます．

しかし，実際には**リスト1-1**のプログラムはこれだけでは動作せず，これ以前にCPUに対する各種モード設定や初期化などを書き込む必要があります．例えば，**図1-1**の外部8255に対しても，各ポートを入力に使うか出力に使うかとかといったCPUからのモード設定が必要です．

1-2　シーケンス回路によるプログラム開発

ここで，シーケンス回路についての記述が出てくることは，唐突な感じがする人もいるもしれませんが，シーケンス回路はマイコン以前のいわゆるオートメーション時代と喧伝された時の，機械コントロールの主役でした．

そして，

```
LD   M10
AND  M11
OR   M20
OUT  Y20
```

などと，ラダー・ニーモニックで記述できることを考えれば，C言語と同じように機械語に変換すれば，少なくとも機械コントロールについては充分に有用な高級言語たりうるわけです．そして実際に長い間，マイコン内蔵のシーケンサ処理用の高級言語として，実際に各メーカで使用され実績を積んできました．本書は，このシーケンス回路を一般のユーザにも，アセンブリ言語やC言語と同等のプログラミング言語として使用できるものとして紹介しています．

● シーケンス回路

シーケンス回路とはリレー回路とも言い，**写真1-1**のようなリレーのコイルに既定の電圧をかけて，**図1-3**のように接点を動作させ，それを利用して電気の流れを変えて，入力/出力の一連の動作を制御しようというものです．

このシーケンス回路はマイコンが誕生する以前から自動制御を処理し，電気的な機械制御のほとんどを構成していたものです．この場合の自動制御と言う用語はアナログの自動制御理論とはまったく関係ありません．自動制御理論はフィードバック理論が中心になっていますが，シーケンス動作とは「順々な動作」とか「逐一動作」とかいう意味で，アナログによるフイードバック的要素はまったくありません．自動制御盤という言葉も今では少し懐かしさを感じるほどになってしまいましたが，マイコン誕生以前の1970年代の機械の横腹には必ずリレーの集合である自動制御盤が付いていました．

今ではそれはシーケンサ回路やマイコン回路にとって変わられましたが，とにかく戦後の1980年以前の高度成長時代の電気制御を一手に引き受けていたのがこのシーケンス回路技術だったのです．

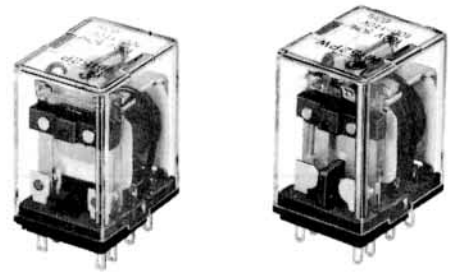
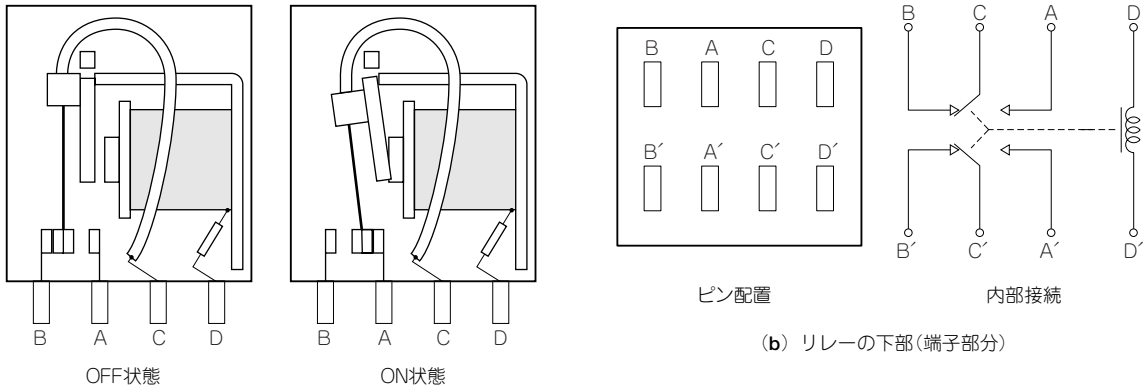


写真1-1　代表的な制御用リレーの外観



（**a**）リレーを横から見たところ

（**b**）リレーの下部（端子部分）

図1-3　リレーの構造と電気的な接続

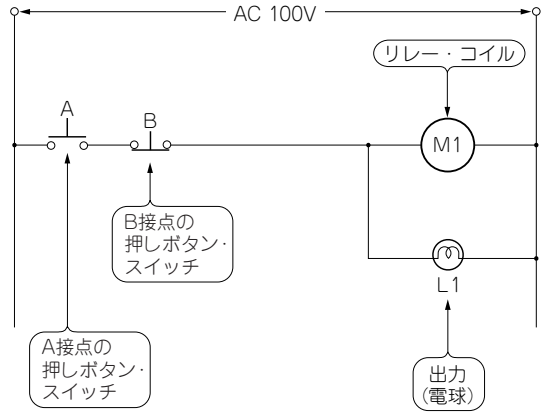


図1-4　電球L1をスイッチAとBでON/OFFしたい

● 自己保持回路

図1-4の回路を見てください．両端の縦軸には電源（例えばAC 100 V）が接続されています．

押しボタン・スイッチA（押している間だけONになる；A接点）を押すと，リレーのM1がONして同時に電球は点灯します．しかし，押しボタン・スイッチAから手を離すと，AはOFFになり，M1も電球もOFFになってしまいます．

押しボタン・スイッチB（押している間だけOFFになる；B接点）を押したままでは，いくら押しボタン・スイッチAを押しても電球はつきません．

次に**図1-5**の回路を見てください．押しボタン・スイッチAの下にあるのは右側のリレーM1の接点です．もしAを押せばM1がONし，その左下のM1の接点がONして，電気の流れに**図1-6**の太線で示す経路ができあがり，Aを離したあともM1がONし続け，電球も点灯し続けます．

M1と電球をOFFしようと思えば，一時的にBを押してOFFすれば**図1-6**の太線の経路が切断されてOFFし，**図1-5**の状態に戻ります．

これがシーケンス回路の基本である自己保持回路です．

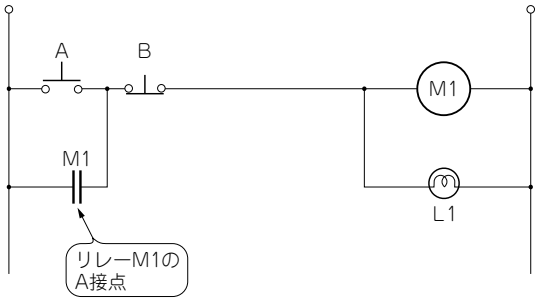


図1-5　自己保持回路(スイッチAでON，BでOFFとなる)

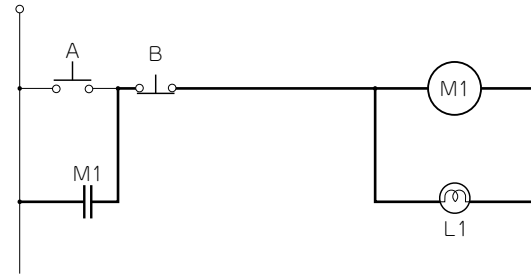


図1-6　図1-5の回路が自己保持しているときの状態

また，この自己保持回路こそが唯一の基本で，ほかに基本として学ぶべき理論はなにもありません．あとは自分で工夫して，より複雑な動作をする回路を組んでいくだけなのです．マイコンの学習に比較して驚くべき簡単さです(マイコンはいくら学んでも，月日の経過とともにその難解さがどんどん増加することはあっても，減少することはありません)．

こんな簡単な原理だけで，機械コントロールに関してだけを考えれば，今もてはやされているマイコンとそう遜色のない制御が可能になるのです．科学的な問題に対する解答は「単純であるほど貴い」とされています．それを考えれば，シーケンス回路には充分に捨て難い価値があると思われます．

リレーによるシーケンス回路は，現在はマイコン内蔵のシーケンサ(PLC；Programmable Logic Controller)にとって変わられましたが，シーケンサはリレーによるシーケンス回路をマイコンのメモリ操作でシミュレーションしたもので，あくまで考え方の基本はシーケンス回路です．もし違いが生じれば原則として，実際のリレー回路のほうを正しいとしてシーケンサ側のソフトウェアを修正することになっています．

実際のリレー回路は電気のON/OFFによって機械的に接点がバタバタと動作していき，そのバラつきを許容しながらある回路動作を形成していくのに対し，シーケンサはあくまで内蔵のマイコンが順々に指定した回路動作を実行していくので，その二つの動作は必然的に違ってきます．なかには，例外としてシーケンサの動作のほうがわかりやすいので，リレー回路と違う動作を原則にしている場合もあります(コラム1-1参照)．

1-3　シーケンス回路の開発環境

シーケンサ(PLC)を使用する場合は，おもなツールは自分のパソコンとメーカの提供するソフトウェアとRS-232Cなどの通信ケーブルがあれば，開発環境が成立します．パソコンのかわりにメーカの提供する回路の入力機器だけで開発する場合もあります(初期の頃はすべてその方法によっていました)．

ソフト/ハードともにツールは必ずメーカから，無料か有料かは別として提供されるということで，ユーザはあれにしようかこれにしようか,組み合わせをどうしようかなどと考える必要があまりありません．ROMライタについても書き込み用ソフトウェアは提供されており，ハードウェア部分はシーケンサ自体に内蔵されています．デバッガ機能も内蔵されています．

マイコン回路の場合は，メーカの提供するものをそのまま使用することはむしろまれで，いろいろなメーカの製品のなかから自分で選んで，シーケンサの何倍もの種類のソフト/ハードを組み合わせて開発環

Column…1-1　リレー接点回路の話

シーケンサは，リレー回路をマイコンでシミュレートしたものです．1970年頃の日本の高度成長期の時代には，日本の工場という工場にこのリレー回路による制御装置(自動制御盤)が入り，機械に取り付けられて活躍していたものです．

その頃から日本の機械技術は世界でも群を抜く存在になったのですから，シーケンス回路に関しては日本は間違いなく世界一だと思われます．欧米のシーケンス技術を確認したわけではありませんが，その頃の日本全体の状況と周囲の技術者達のレベルを考えると，多分そうだろうと確信めいたものが湧いてきます．

残念ながらシーケンサは日本の発明ではなく，1968年のGM社の提案と同時期のマイコンの開発によるものです．私も30年近くまえのある見本市で，シーケンサについての小さいブースをテキサス・インスツルメンツ社が出していて米人社員が1人立っているのを見ましたが，パンフレットを受け取っただけでその説明を詳しく聞こうともしませんでした．「まあ，そんなことも考えてみればできるんだ」とほんやり考えたくらいで，これが自動制御盤に革命をもたらすなどとは思ってもみなかったものです．その後，日本でも各メーカがシーケンサを発売し，機械コントロールについてはマイコン制御とともにシェアを二分することになりました．ちなみに，「シーケンサ」は三菱電機の登録商標で，PLC(プログラマブル・ロジック・コントローラ)が一般的な名称です．

シーケンサと実際のリレー回路の本質的な違いは，前者はマイコンの論理に従って確実に順々にリレー接点が動作しロジックが進んでいくのに対し，後者のリレー回路の場合は接点のON/OFFが各デバイスの電気的特性によりms単位の差でバラバラに動作していくことです．通常，一つの接点がONして次の接点がONするという具合に論理が進んでいくので，そんなバラツキが問題になることはありません．二つの論理が同時に進んで，さあどっちが先にONするかなどというようには回路を組まないからです．

シーケンサはリレー回路の動作をエミュレーションしたもので，二つの動作に違いが出た場合は原則的にリレー回路の動作に従うことになっていますが，本文中にもあるようにかなりの例外もあります．シーケンサのみを扱い，リレー回路を組まない人には必要のない知識かもしれませんが，シーケンサの原

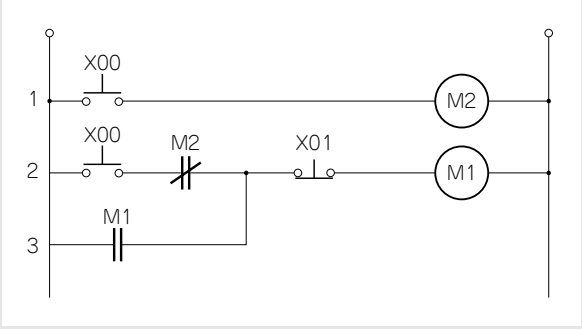


図1-A　リレー回路では動作するがシーケンサではうまく動かない回路

点であるリレー回路の実際の動作を心に刻んでもらうためにもう一度，**図1-A**の自己保持回路について詳しく見ておきたいと思います．

X00を押したときに下段のM1の自己保持が成立するかどうかなのですが，まず前提として，電源を入れた直後にX01を押してM1を確実にOFFにします．また，最上段はX00がOFFになっているので，M2はOFFのままです．したがって，2段目のM2のB接点はONした状態になっています．

次にX00を押します．ここからが問題です．

X00を押したことによって，2段目のX00のA接点のONによってM1はONしてそのまま自己保持をするかどうか？ シーケンサなら答えは簡単で，M1はまったくONしません．なぜなら，最上段のM2がONするので，2段目でX00がONしたとしても直列のM2のB接点がOFF(ブレーク)しているのでM1はONしないのです．そしてこの状態がずっと継続します．

しかし，実際のリレー回路の場合は実はM1は自己保持します．問題はX00を指で押してX00のA接点がONした瞬間です．このとき，最上段のM2がONしようとします(マイコンの論理ではないので物理的に数十ms単位の遅れがある)．M2のB接点がONを短い期間継続することにより，次に2段めのM1がONしようとします．そのとき，2段目のM1のコイルがONして3段目のM1のA接点がくっつくのが早いか，2段目のB接点が離れるのが早いかがまず問題になります．

リレーの接点の動きでは，電磁コイルに引かれたバネ接点(C接点)はB側から離れたあとにA側にくっつくのですが，C接点につながったM1の接点という物体がX00がONになった瞬間からの残留磁気により(また運動の慣性の法則もあって)，2段目のM2のB接点がOFFになりM1のコイルが電気的にOFFになったあともA側に向かって動作を続け，ついにA側に到達して3段目のM1のA接点からの電気の流入によりM1を自己保持させてしまうのです．このことはリレーの種類に関わらず確実に起こります．このことを実際にリレーを使って回路を組んでいた時代のシーケンス屋さんは知っていて，回路的に利用したわけです．

さほどかように，マイコンが遂行するロジック回路とリレー接点の実際の動作は違うわけです．これらのことは科学史的な資料としても，しっかりと記録しておく必要はあると思います．ごく最近では，現場で回路中にシーケンサが入っていると，メンテナンスにやはり専門技術者が必要になるとか，バージョンアップによって簡単な交換ができなくなる可能性があるなどの理由で，シーケンサを使用せずにリレーそのものを電線でつないで，昔のように回路を組もうという動きもあるようです．リレーとタイマとカウンタと電線によるシーケンス回路にも，捨てがたいメリットがあるようです．

少なくとも，リレー回路とシーケンサの併用時代は今後ともしばらく続きそうなので，あまり安易にPLSやPLFなどの微分命令を使用したり，またリレーをONするのに自己保持回路を使用せずSET，RST命令を使用したりして，実際の部品としては存在しない命令を多用してリレー回路からまったく離れてしまうことについては，その問題点を心に留めておいてほしいものです．これから未来にわたって，シーケンサの戻るべき原点は実際のリレー回路であることは，ずっと変わることはないと思うからです．

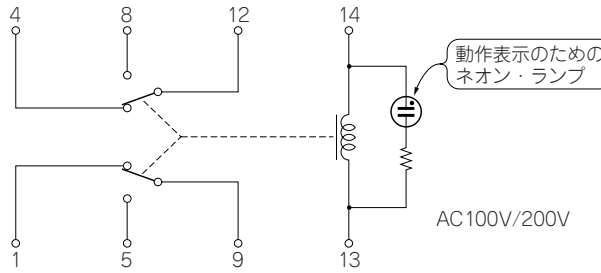


図1-7　動作表示ランプ付きリレーの内部接続

境を構築していかなければならないのが普通でしょう．

リレーには**図1-7**のようにON/OFFを示すランプがあるものもあり，そうでなくてもその動きは現場の専門外の人でもある程度把握できます．修理ができないまでも，どこのリミットが働いていないとか，どこの押しボタンが効かないとかの報告ができ，自分でもある程度の対処が可能になるので，ユーザとメーカの両方のサイドにとって好都合ということになります．なにより，リレーという部品の組み立てによる方法には，特別な開発環境なるものは存在しません(リレー回路華やかなりし1970年代には全国津々浦々どんな小さな会社でも社員をはじめ部課長自らテスタ片手に機械コントロールのためのリレー回路を修理していたものです)．

その点，シーケンサを動かしたりメンテナンスしたりするにはやはり，その開発環境が必要です．そしてそれを使いこなせるのはやはりシーケンサのプロであり，使用しているメーカの同一シリーズのシーケンサになれたものでないと簡単にいじることはできません．

しかし，マイコンとおなじように一応の開発環境が必要だとは言ってもシーケンサの開発環境を獲得することは，マイコンのそれに比較すれば，格段に(比較にならないくらい)簡単です．

● シーケンス回路で使用するラダー・ニーモニック

実際の電気的なリレーで組む場合は必ずシーケンス回路図に従って組んでいきますが，シーケンサは回路図以外にニーモニックでも動作します．

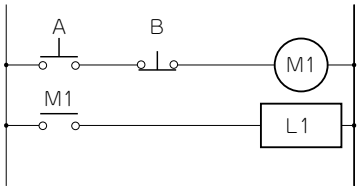
リレー回路図(ラダー図)はニーモニックという言語と1:1に対応しています．ニーモニックは，AND，ORの論理演算と1:1に対応します．

論理演算の考え方は，マイコン以前のロジックICや無接点リレーが使用するMIL記号の使用ですでに確立されていました(現在でも欧米製品を中心としてMIL記号で動作させるシーケンサがあります)．

マイコンは言語による論理の表現ですから，この論理演算を言語化したものがそのままニーモニックになったと思われます．

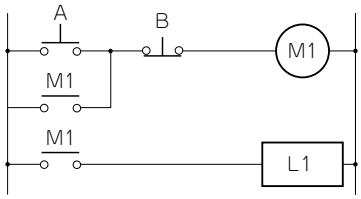
このシーケンサのニーモニックという用語は，マイコンの機械語を意味するニーモニックとは意味を異にしています．この場合，シーケンス・ロジックと言うような意味でしょうか(今後これをラダー・ニーモニックと呼ぶことにします)．

これからそのラダー・ニーモニックについて説明していきますが，詳しくは専門書を参照してください．



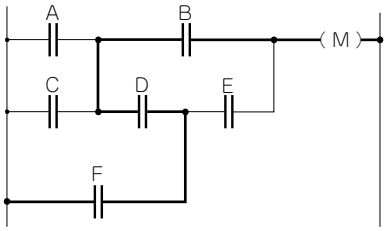

```
LD    A
AND   ~B
OUT   M1
LD    M1
OUT   L1
```

図1-8　簡単なリレー回路とラダー・ニーモニック

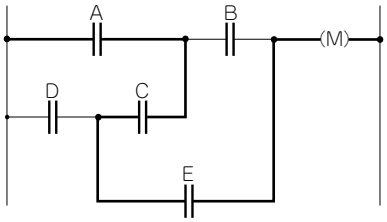



```
LD    A
OR    M1
AND   ~B
OUT   M1
LD    M1
OUT   L1
```

図1-9　自己保持回路とラダー・ニーモニック



（**a**）ニーモニックで作れない回路



（**b**）逆流を利用した回路

図1-10　シーケンサでは作れない回路

例えば**図1-8**の回路は，

```
LD     A
AND    ~B          ; ANI   B，または，ANDNOT   Bと書く
OUT    M1
LD     M1
OUT    L1
```

となり，**図1-9**の回路は，

```
LD     A
OR     M1
AND    ~B          ; ANI   B，またはANDNOT   Bと書く
OUT    M1
LD     M1
OUT    L1
```

となります．

すべてAND，ORのビットの論理演算になり，これらに加えて回路が複雑になったときの回路ブロックどうしのANDを取るANB(AND BLOCK)と，同じく回路ブロックどうしのORを取るORB(OR BLOCK)

があります．

ほとんどの場合，このように簡単なラダー・ニーモニックに訳せますが，そうでない場合もあります．例えば**図1-10**がそうです．この回路は，ラダー・ニーモニックでは書けません．対応するマイコンのア

Column…1-2　シーケンサ小史

リレーによる当時の自動制御盤には以下のような欠点がありました(MELFANS web▶http://www.nagoya.melco.co.jp/index.jhtm参照)．

(1) 接点の磨耗
(2) 多数リレーの取り付けと配線作業が大変
(3) 制御内容への変更が面倒

このような背景から多くのFA関連のユーザの意見を代弁して，1968年にアメリカのGM(General Motors)社から出された下記10項目を満足する製品の要求が出発点になりました．これによる製品が，当時のリレーによる自動制御盤にとって変わり得ることが要求の原点でした．

(1) プログラミングおよびプログラムの変更が容易であり，シーケンスの変更は現場で可能であること
(2) 保守が容易であること．完全なプラグインが望ましい
(3) リレー制御盤より現場での信頼性が高いこと
(4) リレー制御盤より小型のこと
(5) コントローラ・ユニットは中央データ収集システムにデータを送出できること
(6) リレー制御盤と経済的に競合できること
(7) 入力はAC 115 Vが可能なこと
(8) 出力はAC 115 V/2 A以上で，ソレノイド・バルブ，モータ・スタータなどを駆動できること
(9) システム変更を最小限にして，基本システムを拡張できること
(10) 最低4Kワードまで拡張できるプログラマブルなメモリを有すること

この要求に対する解答として1969年に，モトローラ社やロックウェル・オートメーション社をはじめとする7社から開発製品が提示されましたが，このときはマイコンがまだ開発されていないので，ロジックICで構成されていたと思われます(論理はMIL記号によっていたのでしょう)．契約にこぎつけたのはBedford Associates社(マサチューセッツ州ベッドフォード)で，それによって同社はPLC専業のModicon社を創立しました．

1971年のマイコンの登場と，後述のラダー・ニーモニックとの1：1対応の確立により，シーケンサ(PLC；Programmable Logic Controller)が急速に発達し，価格も劇的に下がって，順調にそのシェアを伸ばしていくことになりました(最初のマイコンであるインテル社の4004の発表の直後に，1ビットの論理計算によるシーケンサが発売されています)．

正式に上記仕様に従った国産PLCが誕生したのは1970年でした．その頃の先発メーカとしては豊田工機やオムロンがあり，コア・メモリや出たばかりのICを組み合わせて，すでに外部ツールによりプログラム作成して書き込むストアード方式の製品が発売されていました．続いて，安川，三菱，富士，日立などの各社が次々と参入して，現在の活況にいたっています．

現在，シーケンサがマイコンとシェアを二分して普及している主な理由には次の二つが上げられます．

(1) 機械コントロールに関するプログラムの簡便さ
(2) ノイズに対する経験の蓄積による強靭さ

特に，(2)の耐ノイズ性は圧倒的とさえ言えるもので，何層もの基板と効果的なアイソレートの組み合わせにより充分信頼のおけるものになっています．

マイコン基板を自作する人は，ソフトウェア作成でアセンブリ言語やC言語やラダー・ニーモニックなどのうち，言語として何を使用するかにかかわらず，このノイズの問題に自ら取り組んで解決していく必要があります．

センブリ言語の論理が急激に複雑になってしまいます．それではこんな回路は禁止しようということで，この問題を解決しています（実際のリレー回路では例外的ではありますが普通にある回路ですから，強引といえば強引な話です）．

図1-10（**a**）の場合，B，D，FがONするとMがONしてしまいますし，A，D，EのONでもMがONしてしまいます．このように論理の流れが左から右にも，また右から左にも発生するような回路は簡単にラダー・ニーモニックでは表現できないのです（論理の流れを電気の流れと言い換えることができます）．

シーケンサの場合，接点の論理の流れは，必ず左から右，上から下で，それのみです．ラダー・ニーモニックはその単純な流れに従って論理を展開していきます．

シーケンス回路図がラダー・ニーモニックに1：1に対応していることを考えれば，このラダー・ニーモニックまたはそれに対応する回路図そのものを高級言語として，これをコンパイルすればアセンブリ言語にすることができるわけです．

本書では，このシーケンス回路のコンパイラである「連枝」（れんり）のスタンダード版をCD-ROMに内蔵し，H8/3694F基板を動作させることができます．C言語でよく問題になるプログラムの移植性は，この方法では抜群にスピーディにできます．CPU種の設定さえすれば，各種モード設定にかかわる部分や，アドレスの定義にかかわるヘッダ・ファイルの部分をすべて提供されたコンパイラが作ってしまうので，回路図さえあれば，たとえPICからH8への変更であっても，CPU種の設定の変更と対応するI/O番号を変更するだけで容易に，変更した新しいCPUに対応する機械語（MOT）ファイルができあがります．

シーケンス回路の作成は大変に簡単なので，おそらくアセンブリ言語より格段に簡単な方法でマイコンを動作させることができるでしょう．ぜひこれをマスタし，気軽に各種の機械コントロールに利用することをお薦めします．