ＬＤＣｖ！によるＲＯＭ化開発手順－Ｚ８０編

第１版

この文書では、ラダー図開発ツールＬＤＣｖ！によるＲＯＭ化開発手順について説明しています。
例題として、タイマでＬＥＤを順次点灯するだけの簡単なラダー図をＲＯＭ化して、Ｚ８０互換マイコンボード上で実行させてみることにします。

１．使用するハードウェアの説明

市販のＺ８０互換マイコンボードを使用することにします。
マイコンボードの仕様は次のとおりです。

（１）ＣＰＵ

機械語レベルでＺ８０と互換性のあるＫＬ５Ｃ８０Ａ１２を使用し、これを１０ＭＨｚで動作させます。このＣＰＵには、周辺ＬＳＩの機能（Ｉ／Ｏポート、タイマ等）が内蔵されています。

（２）メモリ

ＳＲＡＭとフラッシュＲＯＭを使用します。メモリマップは、表１のとおりです。

表１．メモリマップ

アドレス（メモリ空間）	内容	容量
００００Ｈ～７ＦＦＦＨ	ＲＯＭ	３２Ｋバイト
８０００Ｈ～ＦＦＦＦＨ	ＲＡＭ	３２Ｋバイト

（３）Ｉ／Ｏポート

ＣＰＵに内蔵されているＩ／Ｏポートを使用します。Ｉ／Ｏマップは、表２のとおりです。
ポート０を入力ポート、ポート１を出力ポートとして使用することにします。

表２　Ｉ／Ｏマップ

アドレス（Ｉ／Ｏ空間）	内容	備考
２ＣＨ	ポート０	入力ポートＰ００～Ｐ０７
２ＤＨ	ポート０方向制御	初期設定で００Ｈを書込（入力に設定）
２ＥＨ	ポート１	出力ポートＰ１０～Ｐ１７
２ＦＨ	ポート１方向制御	初期設定でＦＦＨを書込（出力に設定）

このＩ／Ｏポートに、表３のようにトグルスイッチとＬＥＤランプを接続します。

表３　ポート番号と接続部品

Ｉ／Ｏポート番号	Ｉ／Ｏポートへの接続部品
Ｐ００	トグルスイッチ（ＳＷ０）を接続
Ｐ１０～Ｐ１３	ＬＥＤ（ＬＥＤ０～ＬＥＤ３）を接続

実際の接続回路は、図１のとおりです。

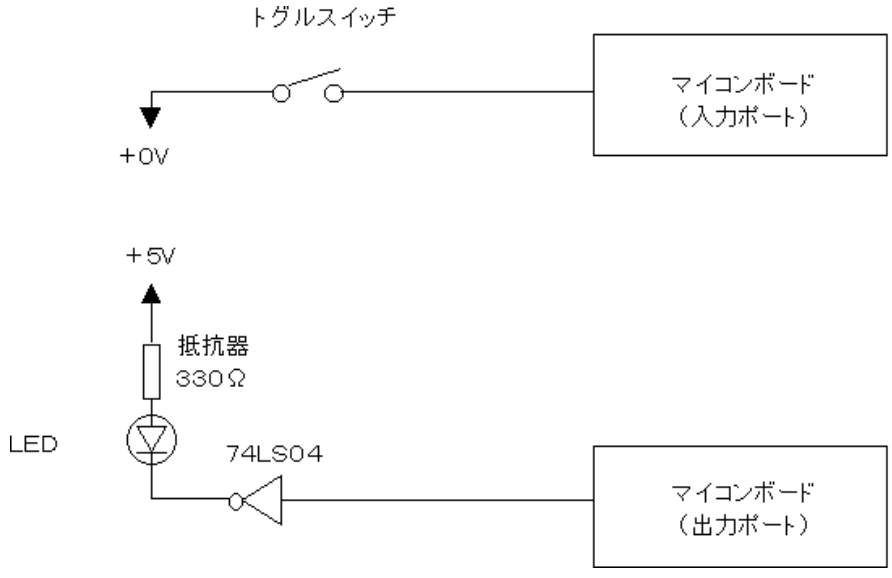

図１　入力回路（上）と出力回路（下）

（各ポートは、マイコンボード内部でプルアップされている）

２．例題の説明

次に、例題の説明します。
最初に説明したように、ＬＥＤを順次点灯するだけの簡単なオン・オフ制御です。

（１）例題の動作

トグルスイッチＳＷ０をＯＮにすると、ＬＥＤは、時間の経過と共にＬＥＤ０から順に点灯してゆきます。
ＬＥＤ３まで全部のＬＥＤが点灯すると、その後、すべてのＬＥＤが同時に消灯します。
消灯後、再びＬＥＤは、時間の経過と共にＬＥＤ０から順に点灯する動作を繰り返します。
トグルスイッチＳＷ０がＯＦＦのときは、すべてのＬＥＤは消灯したままです。

（２）点灯タイミングの実現方法

設定時間が少しずつ異なるタイマＴ０からＴ４で、ＬＥＤの点灯タイミングを作っています。
このタイマの基準となるクロックは、このラダー図自身のスキャンタイムを利用しています。
具体的には、ラダー図の先頭で、Ｓ１３のパルスをＳ０に与えることで実現しています。
ここで、Ｓ１３は「１０００スキャン反転」、Ｓ０は「タイマ基準クロック」を意味します。

（３）ラダー図上でのＩ／Ｏポート番号

ラダー図上では、Ｉ／ＯポートはＩ／Ｏメモリの１つとして取り扱い、それぞれＩ／Ｏ番号を対応させます。
ここでは、表４のように対応させることとし、ラダー図はこの表の約束に従って作成することにします。

表４　Ｉ／ＯポートとＩ／Ｏ番号の対応

Ｉ／Ｏポート	ラダー図上でのＩ／Ｏ番号
Ｐ００～Ｐ０３	Ｍ０～Ｍ３
Ｐ１０	Ｍ１００

３．実際の開発手順

では、実際に開発をおこなってみます。
以下に示す手順１～手順６の順番に開発をおこないます。

（１）ラダー図プログラムを作成する（手順１）

ラダー図開発ツールＬＤＣｖ！で、目的とする動作仕様を実現するラダー図（図２～図３）を作成します。

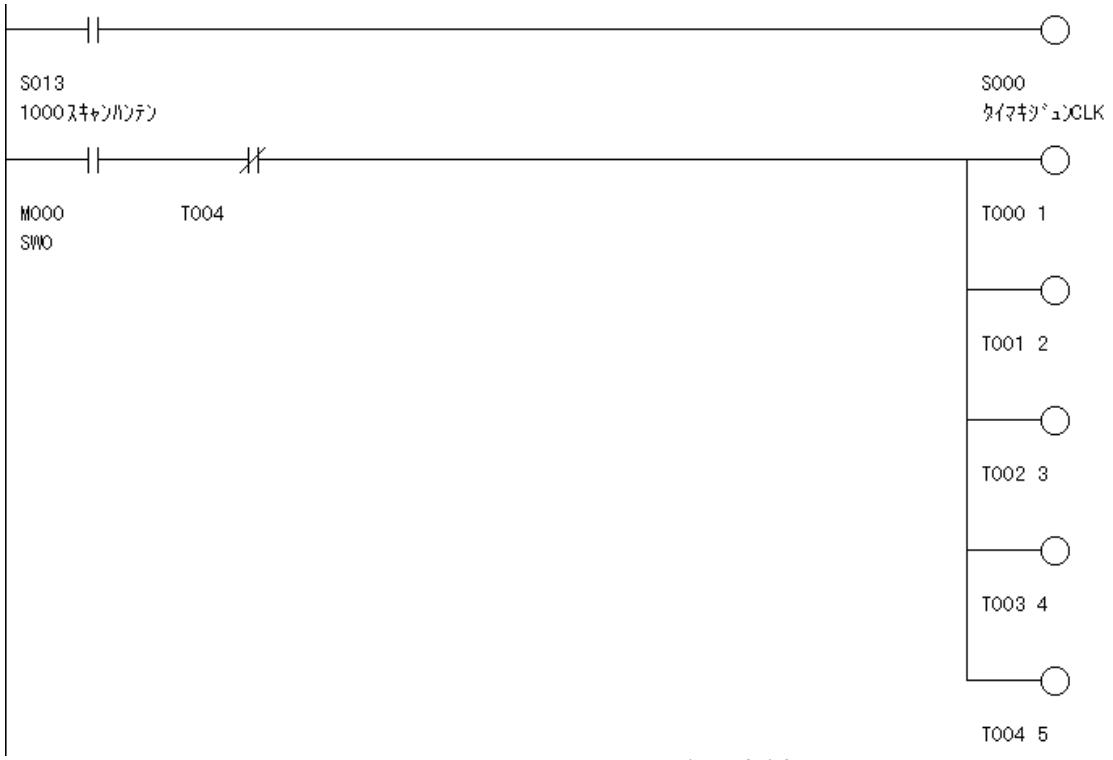

図２　動作仕様を実現するラダー図(1/2)（prog.lad）

図３ 動作仕様を実現するラダー図(2/2)（prog.lad）

（２）変換条件を設定し、変換実行する（手順２）

ラダー図を作成したら、ＬＤ Ｃｖ！の変換条件ウィンドウ上で、変換条件を表５のように設定します。

表５ 変換条件（prog.cdt）

項目	設定
変換言語	Ｚ８０アセンブリ言語
ワークＲＡＭ先頭アドレス	８０００Ｈ
予約ラベル先頭文字列	Ｌ＿
ラベルフィールド文字数	１０
元のラダー図形情報をコメント出力	（チェックする）
ＥＱＵ疑似命令	ＥＱＵ
ラベル末尾識別記号	：
コメント先頭識別記号	；
コメント末尾識別記号	なし
この変換条件のメモ	（空欄のまま）

変換条件の設定が終了したら、ＬＤ Ｃｖ！の変換実行メニューを実行し、このラダー図をＺ８０アセンブリ言語に変換させるとリスト１のようになります。

ここでは prog.asm というファイル名で保存しておきます。

リスト１ 変換結果の一部（prog.asm）

```
;(LD Cv! Version 1.3 出力結果 - Z80ｱｾﾝﾌﾞﾘ言語)
;*****************************************************************
; ﾜｰｸRAMｱﾄﾞﾚｽ定義
;*****************************************************************
L_RAM     EQU 8000H ;ﾜｰｸRAMｴﾘｱ(4Kﾊﾞｲﾄ)の先頭ｱﾄﾞﾚｽ
L_M0      EQU 8000H ;Mｴﾘｱ(M0-M7FF)
L_S0      EQU 8000H+0800H ;Sｴﾘｱ(S0-S07F)
L_T0      EQU 8000H+0900H ;Tｴﾘｱ(T0-T07F)
L_C0      EQU 8000H+0B00H ;Cｴﾘｱ(C0-C07F)
L_WK      EQU 8000H+0D00H ;作業ｴﾘｱおよび未使用ｴﾘｱ
;*****************************************************************
; ﾒｲﾝﾌﾟﾛｸﾞﾗﾑ
;*****************************************************************
;*** 電源投入直後のｲﾆｼｬﾗｲｽﾞ後、このﾌﾟﾛｸﾞﾗﾑに飛び込むと運転を開始する ***
L_ENTRY:
          LD A, 0             ;*** ﾜｰｸRAMｴﾘｱをｾﾞﾛｸﾘｱ ***
          LD (L_RAM),A
```

（３）Ｉ／Ｏリフレッシュプログラムを作成する（手順３）

ＬＤ Ｃｖ！が生成するプログラムは、Ｉ／ＯリフレッシュサブルーチンをＣＡＬＬ命令で呼び出していますが、そのＩ／Ｏリフレッシュサブルーチンはどこにもありません。

このＩ／Ｏリフレッシュサブルーチンは、開発者が自分で作成する必要があります。
このサブルーチンの仕様（転送元と転送先）は、ラダー図を作成するときに決めた対応と一致していなければなりません。
ラダー図を作成するときに決めた対応にしたがって、Ｉ／Ｏリフレッシュサブルーチン仕様を表６のように決めます。
（ここでは８点単位で転送させているため、一部使用していないＩ／Ｏもリフレッシュしています。）

表６　Ｉ／Ｏリフレッシュサブルーチンの仕様

サブルーチン名	動作
L_INREF	次の入力リフレッシュ動作をおこなう Ｐ００→Ｍ０００ Ｐ０１→Ｍ００１ Ｐ０２→Ｍ００２ Ｐ０３→Ｍ００３ Ｐ０４→Ｍ００４ Ｐ０５→Ｍ００５ Ｐ０６→Ｍ００６ Ｐ０７→Ｍ００７
L_OUTREF	次の出力リフレッシュ動作をおこなう Ｍ１００→Ｐ１０ Ｍ１０１→Ｐ１１ Ｍ１０２→Ｐ１２ Ｍ１０３→Ｐ１３ Ｍ１０４→Ｐ１４ Ｍ１０５→Ｐ１５ Ｍ１０６→Ｐ１６ Ｍ１０７→Ｐ１７

これをＺ８０アセンブリ言語でコーディングすると、リスト２のようになります。
ここでは ioref.asm というファイル名で保存しておきます。

リスト２　Ｉ／Ｏリフレッシュサブルーチン（ioref.asm）

```
;入力リフレッシュサブルーチン（ P00-07 → M000-M007 ）
L_INPREF: IN A,(P0)
          LD B,8
          LD HL,L_M0
L001:     SRL A
          RL (HL)
          INC HL
          DJNZ L001
          RET

;出力リフレッシュサブルーチン（ M100-M107 → P10-17 ）
L_OUTREF: LD B,8
          LD HL,L_M0+100H
L002:     LD C,(HL)
          RR C
          RR A
          INC HL
          DJNZ L002
          OUT (P1),A
          RET
```

（４）電源投入からスキャン開始までのプログラムを作成する（手順４）
ＬＤ　Ｃｖ！が生成するプログラムは、電源投入直後の初期設定からスキャン開始までの処理がありません。
この処理をおこなうプログラムも、開発者が自分で作成する必要があります。
ここでは、リスト３のように作成し、main.asm というファイル名で保存しておきます。

リスト３ 電源投入からスキャン開始まで（main.asm）

```
ROM       EQU 0000H          ;ＲＯＭ先頭アドレス
RAM       EQU 8000H          ;ＲＡＭ先頭アドレス
P0        EQU 2CH            ;ポート０
P0EN      EQU 2DH            ;ポート０方向制御
P1        EQU 2EH            ;ポート１
P1EN      EQU 2FH            ;ポート１方向制御
;
          SEGMENT CODE
          ORG 0000H
          JP START           ;電源投入後、ここから実行
;
          ORG 0100H
START:    LD SP,ROM          ;スタックポインタをＲＡＭの末尾＋１に設定
          LD A,00H           ;ポート０を入力ポートに設定
          OUT (P0EN),A
          LD A,0FFH          ;ポート１を出力ポートに設定
          OUT (P1EN),A
          JP L_ENTRY         ;スキャン動作を開始
;
          INCLUDE (prog.asm) ;ＬＤＣｖ！が生成したプログラムをここに取り込む
          INCLUDE (ioref.asm) ;Ｉ／Ｏリフレッシュサブルーチンをここに取り込む
          END
```

リスト３をみてわかるように、このプログラムの中で、他の２つのプログラムをインクルードしています。

（５）アセンブル・リンクし、ＲＯＭに書き込む（手順５）

作成したプログラム main.asm をアセンブル・リンクします。

prog.asm と ioref.asm は、アセンブル時に main.asm が自動的に取り込む（インクルードする）ため、指定する必要はありません。

ここでは、リスト４のようなバッチファイルを作り、ＷｉｎｄｏｗｓのＤＯＳ窓から実行しました。

なお、このリストは、使用するアセンブラ・リンカによって異なります。

リスト４ アセンブル・リンクバッチファイル（asmlink.bat）

```
YA80 /L MAIN.ASM
YLINK /SA:0 /O:MAIN.HEX /OF:H /M:MAIN.MAP /W MAIN.OBJ
```

このバッチファイルを実行すると、main.hex というＲＯＭ書込器が読み込み可能なインテルＨＥＸフォーマットのファイルが生成されます。

これをＲＯＭ書込器に転送し、ＲＯＭに書き込みます。

（６）マイコンボードへＲＯＭを実装、電源投入する（手順６）

書き込んだＲＯＭをマイコンボードのＲＯＭソケットへ実装し、電源を投入します。

トグルスイッチを操作し、ＬＥＤが動作仕様のとおり点灯すれば成功です。

４．使用した開発ツールの説明

参考までに、ここで使用した開発ソフト・ツールを表７に示します。

表７ 使用した開発ソフト・ツール

品名	製品名・形式	購入先
マイコンボード	ＥＡＩ－０１	有限会社イエローソフト
Ｚ８０アセンブラ	ＹＣ８０	有限会社イエローソフト
リンカ	ＹＬＩＮＫ	有限会社イエローソフト
ＲＯＭライタ	Ｓｍａｒｔ－Ｗｒｉｔｅｒ	有限会社システムロード

以上

来歴

1999/04/22 新規作成し、第１版とする。