間接アドレス指定
入力・出力・メモリビット・DBのアドレス

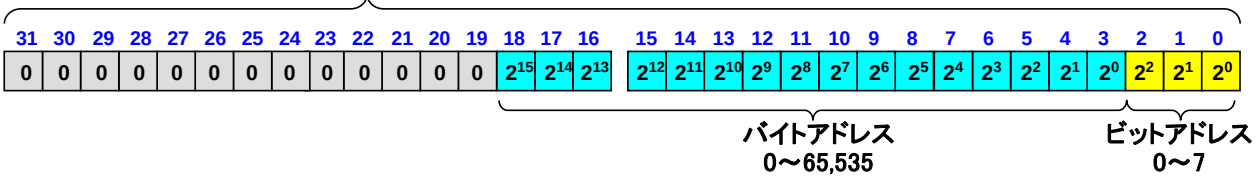
STL

A　　I［MD 40］

MDもしくはDBDの
4バイト長エリアを指定

使用可能なアドレスタイプ

I/IB/IW/ID	ディジタル入力
Q/QB/QW/QD	ディジタル出力
M/MB/MW/MD	メモリ
DBX/DBB/DBW/DBD	データブロック
PIB/PIW/PID	ダイレクト入力
PQB/PQW/PQD	ダイレクトし出力



バイトアドレス
0～65,535

ビットアドレス
0～7

```
L     P# 8.7
T     MD2
A     I[MD2]
=     Q 4.0
```

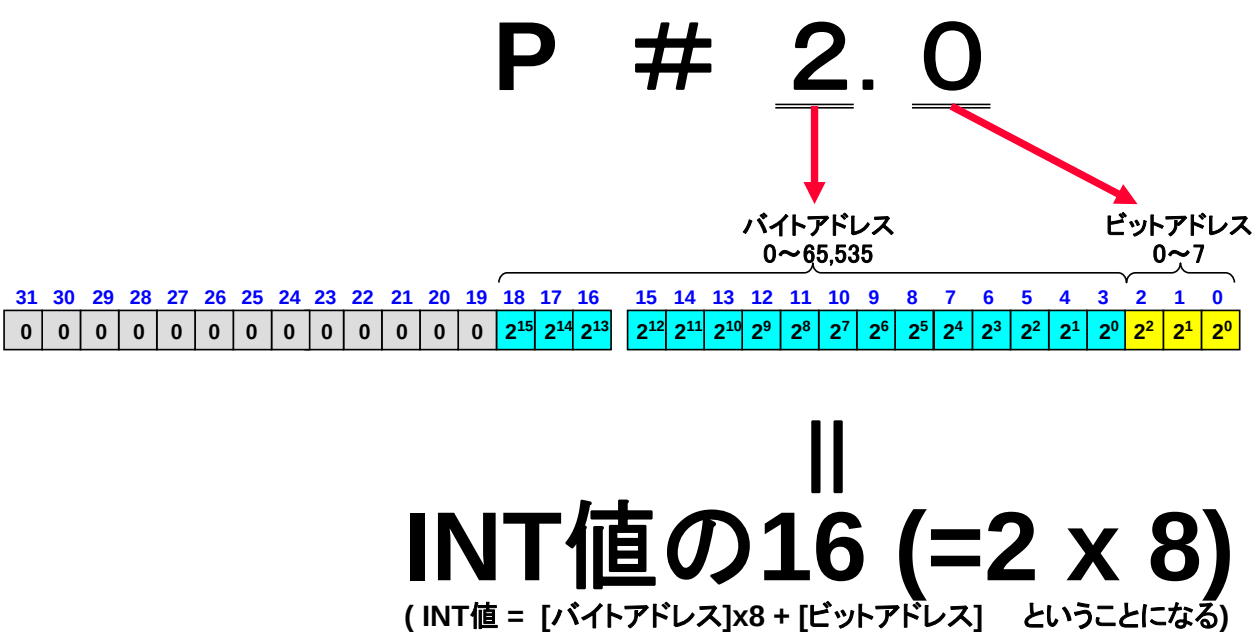
=

```
A     I 8.7
=     Q 4.0
```

- 上記アドレスに対しての間接アドレスはダブルワード指定で行います。アドレスの指定はポインタで行います。ポインタフォーマットは0～3ビットがビットアドレス、3～18ビットがバイトアドレスになります。このフォーマットを定数で指定するにはポインタ（P#）を使用します。

間接アドレス指定
入力・出力・メモリビット・DB のアドレスの定数表示

STL



=

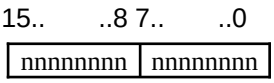
INT値の16 (=2 x 8)

(INT値 = [バイトアドレス]x8 + [ビットアドレス]　ということになる)

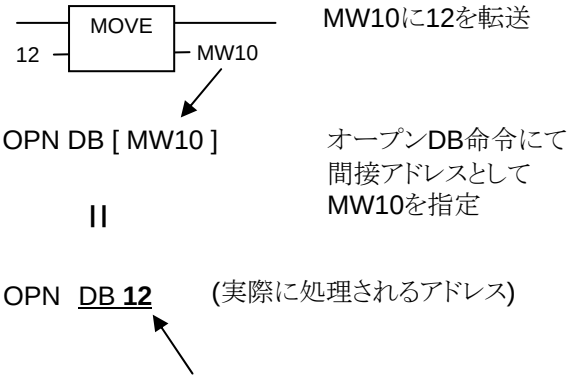
- 入力/出力/メモリビット/DBのアドレスはポインタ表記で表します。
- これは上記のごとく、“P#”の後の整数部分をビット18～3、小数点以下の少数部分をビット2～0の数値を表示しています。

- 間接アドレス指定の使用例

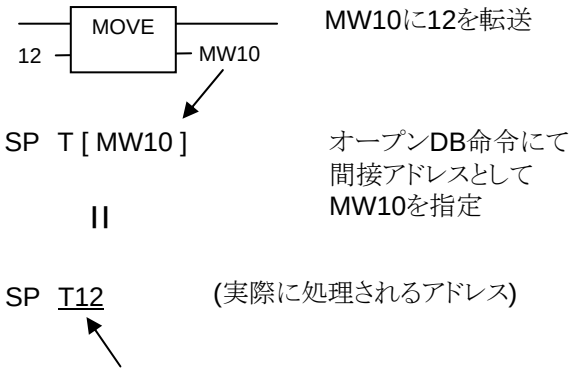
INTフォーマット



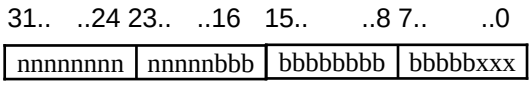
例:DB番号の間接アドレス指定



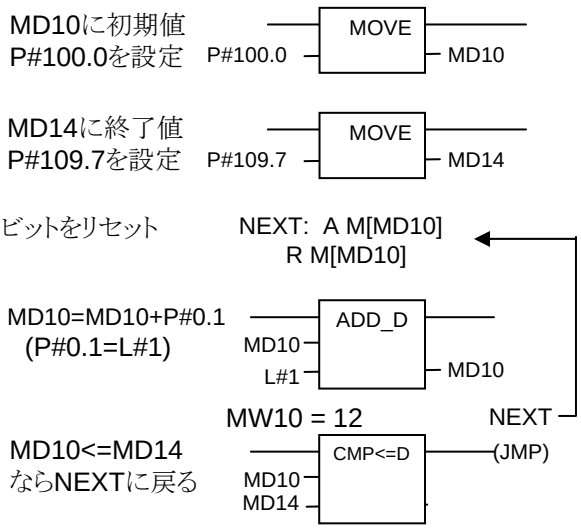
例:タイマ番号の間接アドレス指定



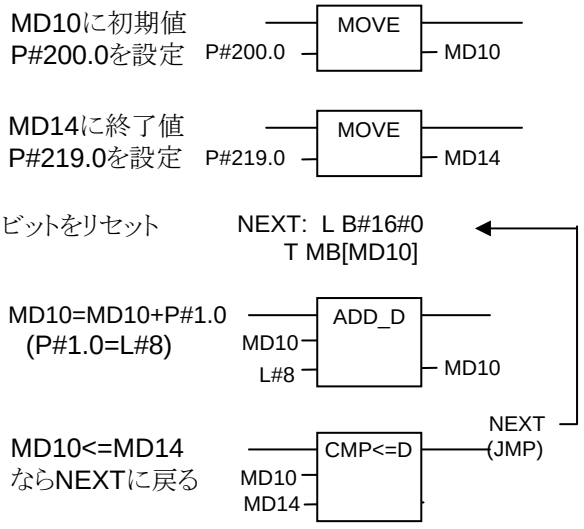
ダブルワードポインタフォーマット



例:M100.0～M109.7のビットをリセットする



例:MB200～MB219のバイトをクリアする



間接アドレス指定
入力・出力・メモリビット・DB のアドレス(アドレスタイプも指定)

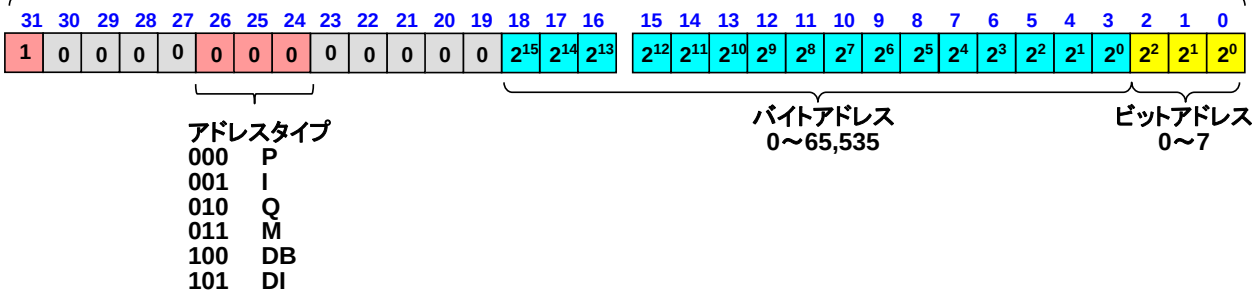
STL

```
L     W [ MD  40 ]
```

MDもしくはDBDの
4バイト長エリアを指定

使用可能アドレス

I/IB/IW/ID	ディジタル入力
Q/QB/QW/QD	ディジタル出力
M/MB/MW/MD	メモリ
DBX/DBB/DBW/DBD	データブロック
PIB/PIW/PID	ダイレクト入力
PQB/PQW/PQD	ダイレクトし出力



アドレスタイプ
000 P
001 I
010 Q
011 M
100 DB
101 DI

バイトアドレス
0～65,535

ビットアドレス
0～7

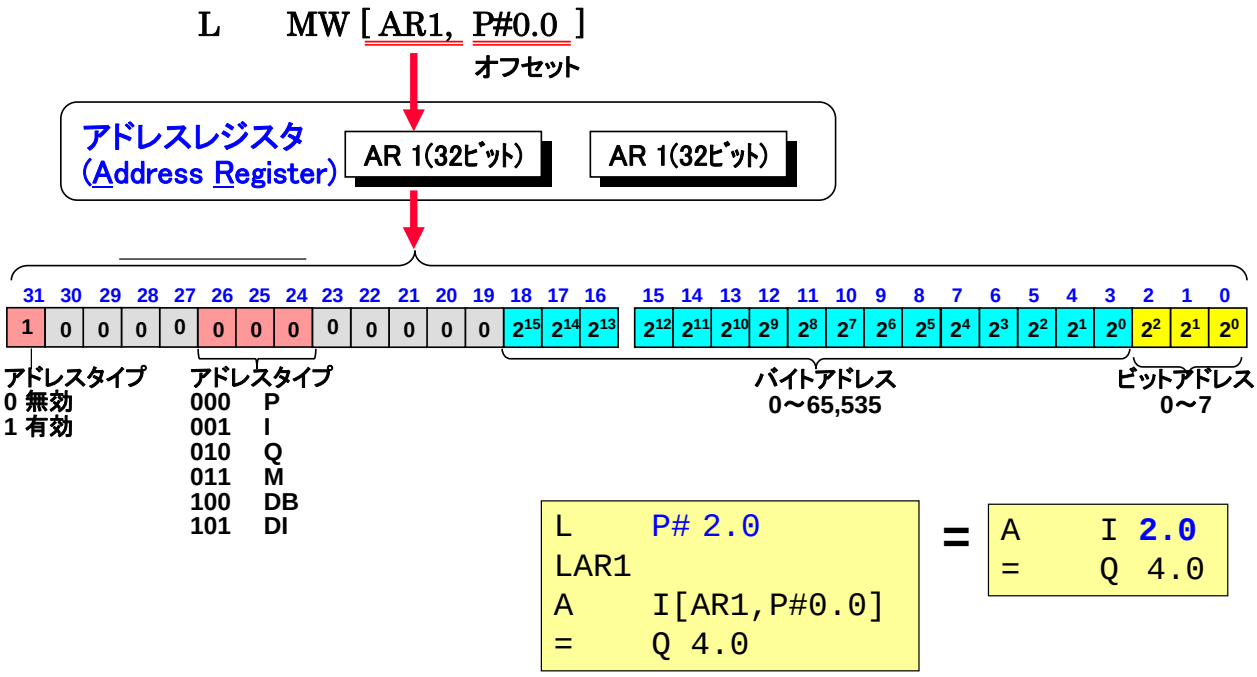
- 入力/出力/メモリビット/DBの間接アドレス指定をする場合、I/Q/M/DBのアドレスタイプをも変数にして指定することができます。

- 上記の様に、ビット31が“1”になると、アドレスタイプも変数として取り扱うことを意味します。
- さらにビット26～24の3ビットに、上記の様にアドレスタイプを指定します。

- “P#”で定数指定するときも、“P#M10.0”というようにアドレスタイプを付けて指定することもできます。

アドレスレジスタ

STL




```
L     P# 2.0
LAR1
A     I[AR1,P#0.0]
=     Q 4.0
```

=

```
A     I 2.0
=     Q 4.0
```

- S7-300/S7-400のCPUには、アドレスレジスタというレジスタがあります。
- このエリアはポインタアドレスを扱うための専用レジスタであり、MやDBなどのエリアを使わずに間接アドレスすることが可能になります。

アドレスレジスタ
アドレスレジスタを操作する命令

STL

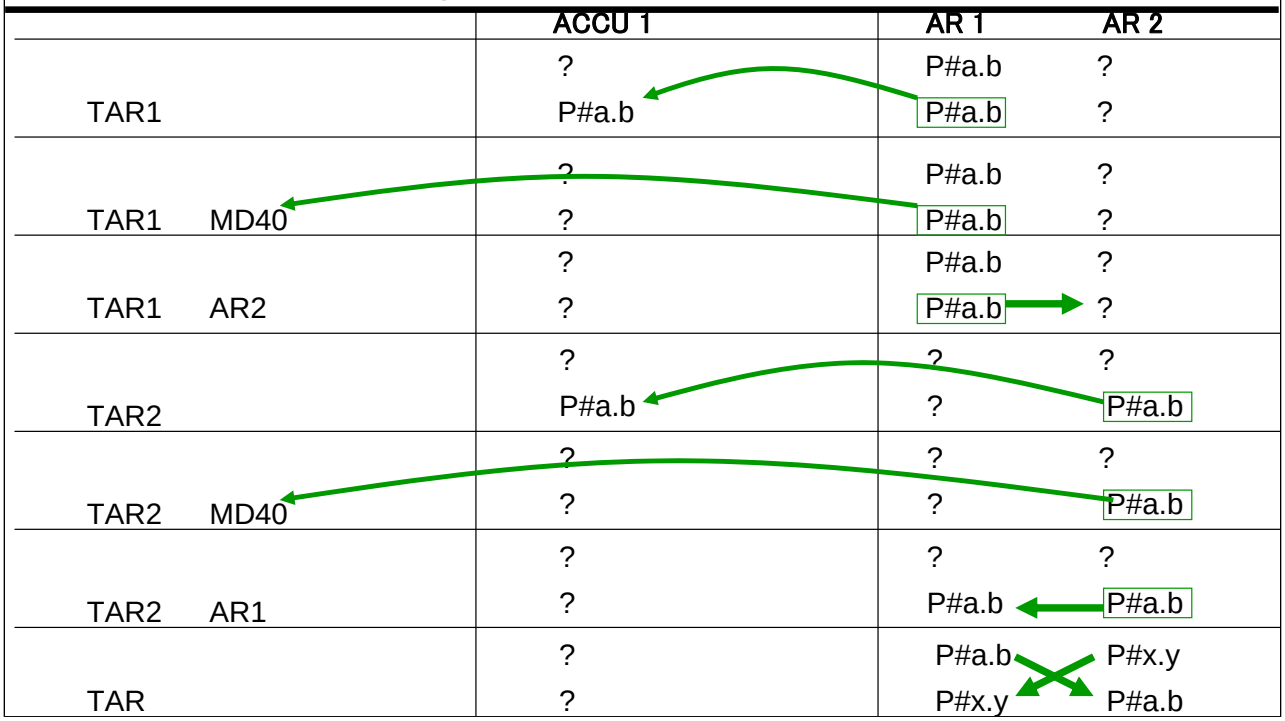
		ACCU 1	AR 1	AR 2
L	P#a.b	P#a.b	?	?
LAR1		P#a.b	P#a.b	?
LAR1	P#x.y	?	P#x.y	?
LAR1	P#Mx.y	?	P#Mx.y	?
LAR1	MD40	?	P#x.y	?
LAR1	P##tmp_name	?	P#Lx.y	?
LAR1	AR2	?	P#a.b	P#a.b
L	P#a.b	P#a.b	?	?
LAR2		P#a.b	?	P#a.b
LAR2	P#x.y	?	?	P#x.y
LAR2	P#Mx.y	?	?	P#Mx.y
LAR2	MD40	?	?	P#x.y
LAR2	P##tmp_name	?	?	P#Lx.y
LAR2	AR1	?	P#a.b	P#a.b

- LAR1　　AR1にポインタ形式のアドレスを書き込む
  “LAR1”命令の後にオペランドがないと、ACCUM1の内容をAR1に書き込む。
  “LAR1”命令の後にオペランドが入力されていると、そのオペランド値がAR1に書き込まれる。

- LAR2　　AR2にポインタ形式のアドレスを書き込む
  “LAR2”命令の後にオペランドがないと、ACCUM1の内容をAR2に書き込む。
  “LAR2”命令の後にオペランドが入力されていると、そのオペランド値がAR2に書き込まれる。

アドレスレジスタ
アドレスレジスタを操作する命令

STL



- TAR1　　AR1のポインタ形式のアドレスを指定したエリアに書き込む
  “TAR1”命令の後にオペランドがないと、AR1の内容をACCUM1に書き込む。
  “TAR1”命令の後にオペランドが入力されていると、AR1の内容をそのオペランドに書き込む。

- TAR2　　AR2のポインタ形式のアドレスを指定したエリアに書き込む
  “TAR2”命令の後にオペランドがないと、AR2の内容をACCUM1に書き込む。
  “TAR2”命令の後にオペランドが入力されていると、AR2の内容をそのオペランドに書き込む。

- TAR　　AR1とAR2の内容を入れ替える

アドレスレジスタ
アドレスレジスタを操作する命令

STL

	ACCU 1	AR 1	AR 2
L P#a.b	P#a.b	P#x.y	?
+AR1	P#a.b	P#x.y + a.b	?
	?	P#x.y	?
+AR1 P#a.b	?	P#x.y + a.b	?
L P#a.b	P#a.b		P#x.y
+AR2	P#a.b		P#x.y + a.b
	?	?	P#x.y
+AR2 P#a.b	?	?	P#x.y + a.b

- +AR1　　AR1に指定した数を加算する
  ”+AR1“の後にオペランドがないと、ACCUM1の値を加算する。
  “+AR1”の後にオペランドがあると、その数を加算する。

- +AR2　　AR2に指定した数を加算する
  ”+AR2“の後にオペランドがないと、ACCUM1の値を加算する。
  “+AR2”の後にオペランドがあると、その数を加算する。

アドレスレジスタ LAR1の使用例

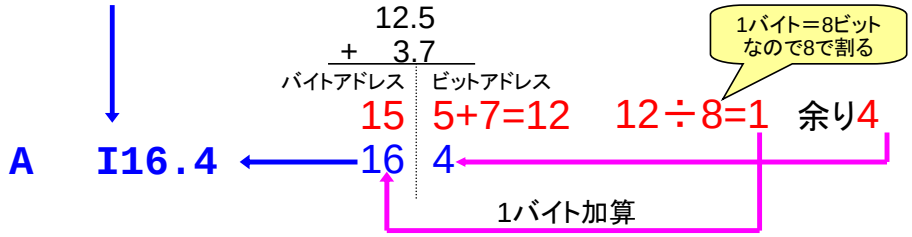
STL

ビットアドレス≦7

```
LAR1   P#8.2              AR1(アドレスレジスタ1)に32ビットポインタ P#8.2 を読み込む
A      I[AR1,P#10.2]      AR1の値にこのポインタP#10.2を加算したアドレス
                          8.2+10.2=18.4 になり A I18.4
```

ビットアドレス＞7

```
L      P#12.5
LAR1                      AR1(アドレスレジスタ1)にACCU1 のポインタを読み込む
A      I[AR1,P#3.7]       AR1の値にこのポインタP#3.2を加算したアドレス
```



LAR1

- アドレスレジスタ1(AR1)へ値を読み込む命令。
- LAR1 にパラメータがある場合は、そのパラメータを AR1 へ読み込みます。パラメータが記述されていない場合は ACUU1 の値を AR1 へ読み込みます。