TOSHIBA

TLCS-870/C1 シリーズ　CPU

株式会社 東芝 セミコンダクター社

CMOS 8 ビット マイクロコンピュータ

TLCS-870/C1 シリーズ

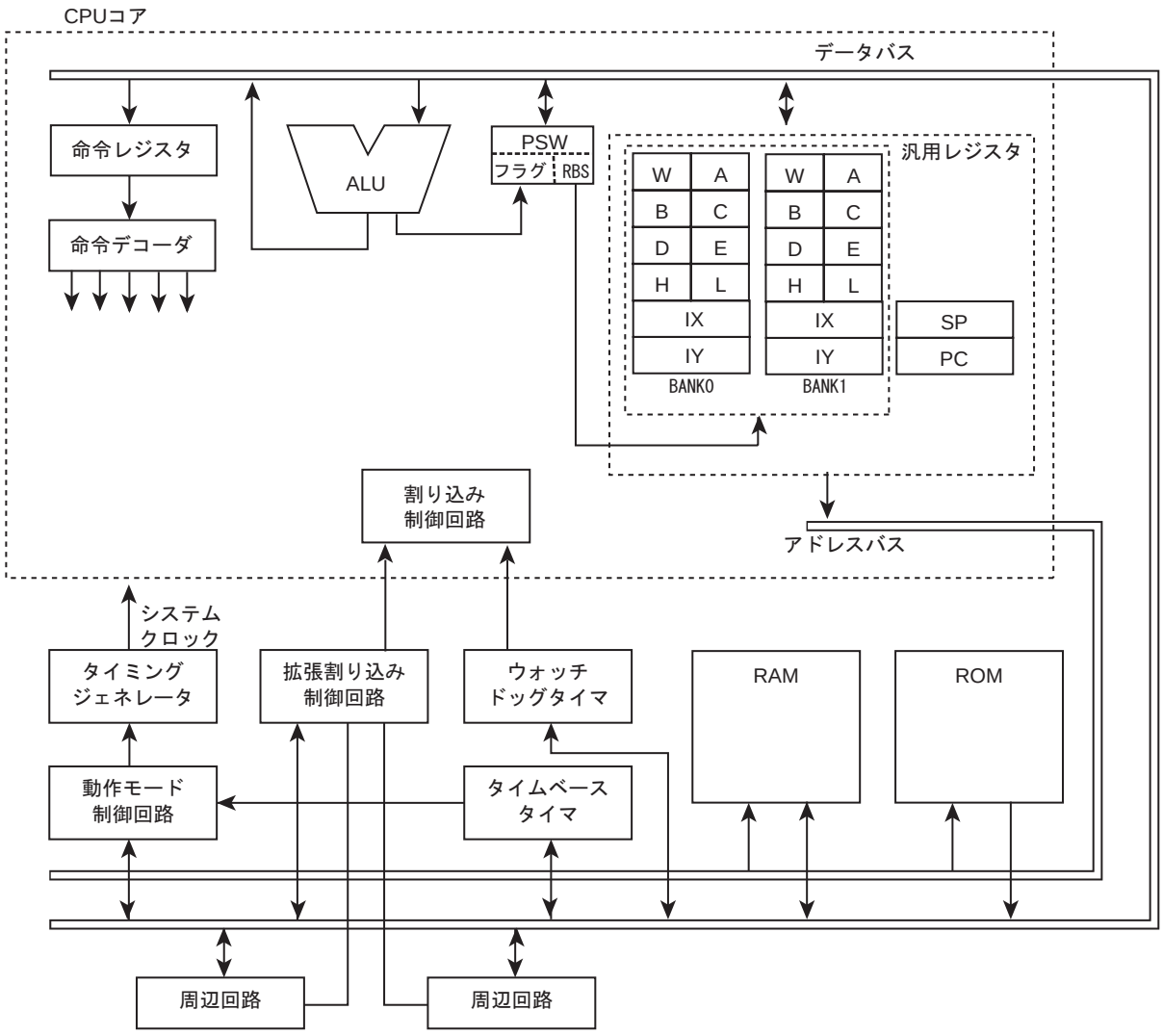
TLCS-870/C1 シリーズは、東芝オリジナルのコンパクトで高速 / 高機能 , 低消費電力の 8 ビットシングルチップマイクロコンピュータです。

特長

- 直交性のある豊富な命令セット : 133 種 732 命令
  東芝オリジナルマイクロコントローラ　TLCS-870/C とバイナリレベルでのコンパチビリティを持たせており、小規模システム向けに最適化した命令セットで、C コンパイラに向いています。
  - 乗除算 (8 ビット × 8 ビット , 16 ビット ÷ 8 ビット)
  - ビット操作 (Set/Clear/Complement/Load/Store/Test/Exclusive OR)
  - 16 ビット演算 / 転送
  - 1 バイト長のジャンプ / コール (Short relative jump/Vector call)
- レジスタ構成 (メモリ空間から独立)
  8 ビット /16 ビット汎用レジスタを 2 バンク化
  - 汎用レジスタ : 8 ビット幅 8 本 (16 ビット幅 4 本としても使用可) × 2
  - 16 ビット汎用レジスタ : 16 ビット幅 2 本× 2
  - プログラムカウンタ : 16 ビット幅 1 本
  - スタックポインタ : 16 ビット幅 1 本
  - プログラムステータスワード : 7 ビット幅 1 本
- メモリ空間
  - コード領域 : 最大 64K バイト
  - データ領域 : 最大 64 K バイト
- メモリマップド I/O 方式
- 割り込み
  - 最大 62 要因まで対応可 (リセットを除く)
  - エッジ選択、ノイズ除去機能付き外部割り込み入力
- 低消費電力動作モード (最大 9 種のモード)
  - STOP モード : 動作停止 (バッテリー / コンデンサバックアップ)
  - SLOW1 モード : 低周波クロックによる低周波動作 (高周波停止)
  - SLOW2 モード : 低周波クロックによる低周波動作 (高周波発振)
  - IDLE0 モード : CPU 停止、周辺ハードウエアのうち、タイムベースタイマのみ動作 (高周波クロック) 継続し、タイムベースタイマ設定の基準時間経過により解除。
  - IDLE1 モード : CPU 停止、周辺ハードウエアのみ動作 (高周波クロック)、割り込みにより解除 (CPU 再起動)。
  - IDLE2 モード : CPU 停止、周辺ハードウエアのみ動作 (高周波 / 低周波クロック)、割り込みにより解除 (CPU 再起動)。
  - SLEEP0 モード : CPU 停止、周辺ハードウエアのうち、タイムベースタイマ (TBT) のみ動作 (低周波クロック) 継続し、タイムベースタイマ設定の基準時間経過により解除。
  - SLEEP1 モード : CPU 停止、周辺ハードウエアのみ動作 (低周波クロック)、割り込みにより解除。
- クロックギア (逓倍・分周)
- 高速処理

- 低電圧 / 高速動作
- フェイルセーフ機能
  - ウォッチドッグタイマ (リセット or 割り込み)
  - システムクロックリセット
  - 未定義命令割り込み
  - ソフトウエア割り込み

ブロック図



注) PSW: プログラムステータスワード

第 1 章　動作説明

1.1　CPU コア機能

1.1.1　メモリマップ

TLCS-870/C1 シリーズのメモリマップは、コード領域とデータ領域の 2 つに分けられます。

1.1.1.1　コード領域

コード領域は最大 64K バイトの領域で、命令のオペコード，オペランドの他に、ベクタコール命令用ベクタテーブル、割り込みベクタテーブルが格納されます。

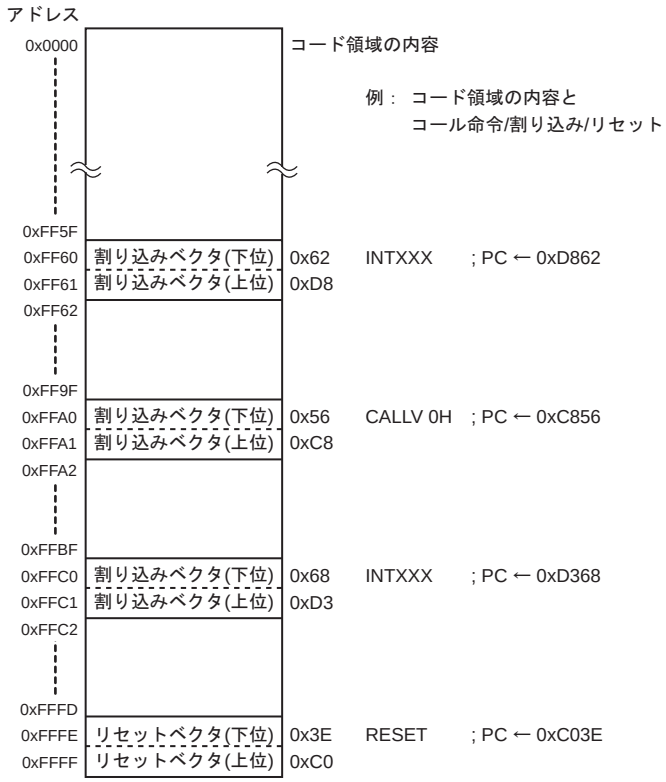


図 1-1 コード領域 (メモリスタイル：64K バイト時)

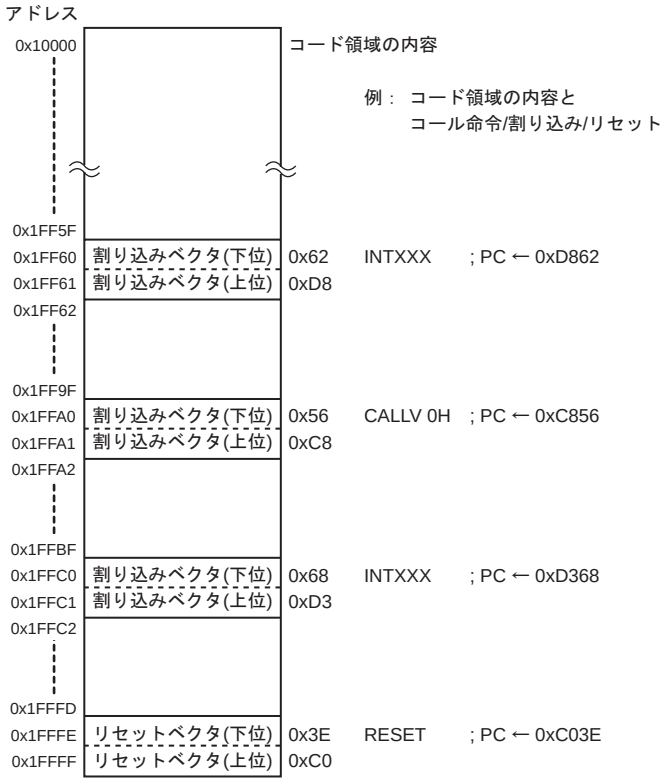


図 1-2 コード領域 (メモリスタイル：96K, 128K バイト時)

(1) ベクタコール命令用ベクタテーブル

ベクタコール命令用ベクタテーブルは、ベクタコール命令 [CALLV n] 用のベクタ (サブルーチン エントリーアドレス , 2 バイト / ベクタ) を格納するテーブルで、16 ベクタあります。ベクタコール命令用ベクタテーブルは、メモリスタイルに従って以下の領域に割り当てられます。

メモリスタイル	ベクタコール命令用ベクタテーブル領域
64K バイト	0xFFA0 ~ 0xFFBF
96K バイト	0x1FFA0 ~ 0x1FFBF
128K バイト	

通常コール命令 [CALL mn] は、一つの命令に 3 バイト長の ROM 容量が必要ですが、ベクタコール命令 [CALLV n] は 1 バイト長の命令のため、使用頻度の高いサブルーチンコールに使うことによりメモリ効率を上げることができます。

	CALLV 命令	CALL 命令	
1 箇所から CALL	1 + 2 = 3 バイト	3 バイト	→ CALLV と CALL の効率同じ
2 箇所から CALL	2 + 2 = 4 バイト	6 バイト	→ CALLV の方が効率良い
3 箇所から CALL	3 + 2 = 5 バイト	9 バイト	
4 箇所から CALL	4 + 2 = 6 バイト	12 バイト	

*CALLV の +2 はベクタコール領域に配置されたジャンプ先アドレスの 2 バイト

(プログラム例)　ベクタコール (メモリスタイルが 64K バイトの場合)

CALLV で呼ばれる定数を 2 倍して 0xFFA0 を加えた値をアドレスとするメモリから呼び出した 16 ビットアドレスにジャンプします。

```
                CALLV       0x02
                 :  :        :  :
                 :  :        :  :
PSUB _ 2:       LD          (HL),A
                DEC         A
                 :  :        :  :
                 :  :        :  :
VECTOR _ CALL section code abs = 0xFFA0
                DW          PSUB _ 0            ; 0x00
                DW          PSUB _ 1            ; 0x01
                DW          PSUB _ 2            ; 0x02   最大 16 個
                 :  :        :  :
                DW          PSUB _ F            ; 0x0F
```

(プログラム例)　ベクタコール (メモリスタイルが 96K, 128K バイトの場合)

CALLV で呼ばれる定数を 2 倍して 0x1FFA0 を加えた値をアドレスとするメモリから呼び出した 16 ビットアドレスにジャンプします。

```
                CALLV       0x02
                 :  :        :  :
                 :  :        :  :
PSUB _ 2:       LD          (HL),A
                DEC         A
                 :  :        :  :
                 :  :        :  :
VECTOR _ CALL section code abs = 0x1FFA0
                DW          code _ addr(PSUB _ 0)     ; 0x00
                DW          code _ addr(PSUB _ 1)     ; 0x01
                DW          code _ addr(PSUB _ 2)     ; 0x02   最大 16 個
                 :  :        :  :
                DW          code _ addr(PSUB _ F)     ; 0x0F
```

（2） 割り込みベクタテーブル

割り込みベクタテーブルは、リセットおよび割り込みのベクタ (2 バイト / ベクタ) を格納するテーブルで、最大 63 ベクタあります。割り込みベクタテーブルは、メモリスタイルに従って以下の領域に割り当てられます。ベクタには、リセット解除からのスタートアドレス , 割り込みサービスルーチンのエントリアドレスを格納します。

メモリスタイル	割り込みベクタテーブル領域
64K バイト	0xFF60 ~ 0xFF9F, 0xFFC0 ~ 0xFFFF
96K バイト	0x1FF60 ~ 0x1FF9F, 0x1FFC0 ~ 0x1FFFF
128K バイト	

1.1.1.2　データ領域

データ領域には転送命令、演算命令などのソースまたはディスティネーションとしてアクセスされるデータが格納されます。

なお､データ領域のアドレス 0x00000 ~ 0x000FF (メモリスタイルが 64K バイトのときは 0x0000 ~ 0x00FF) はダイレクト領域となっています。この領域に対しては、実行時間を短縮した命令による処理が可能です。

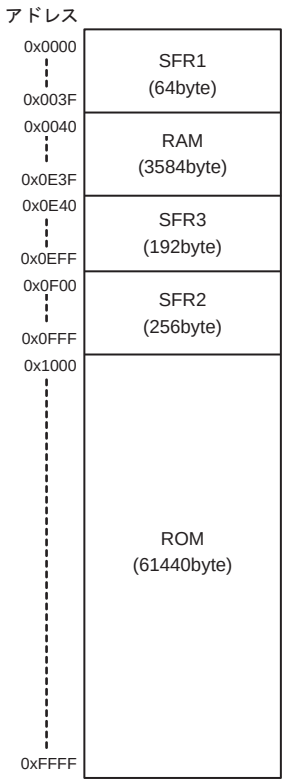


図 1-3　データ領域 (メモリスタイル：64K バイト時)

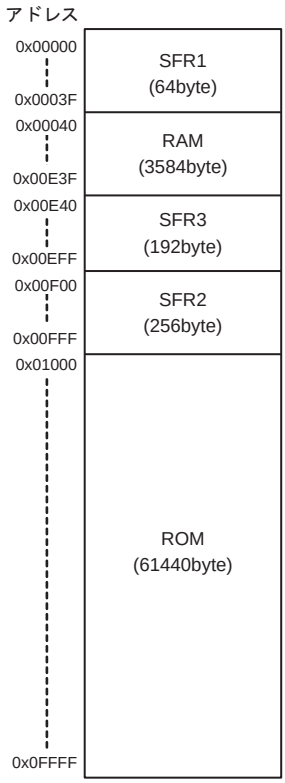


図 1-4　データ領域 (メモリスタイル：96K, 128K バイト時)

1.1.2 汎用レジスタ

TLCS-870/C1は、8つの 8 ビット汎用レジスタ W, A, B, C, D, E, H, L を各BANKに1組づつ、計2BANK内蔵しています。各レジスタは、WA, BC, DE, HL のペアで、16 ビットレジスタとしても使用できます。

汎用レジスタはアドレス空間にはマッピングされていません。また、汎用レジスタは電源投入時及びリセット時の値は不定となります。

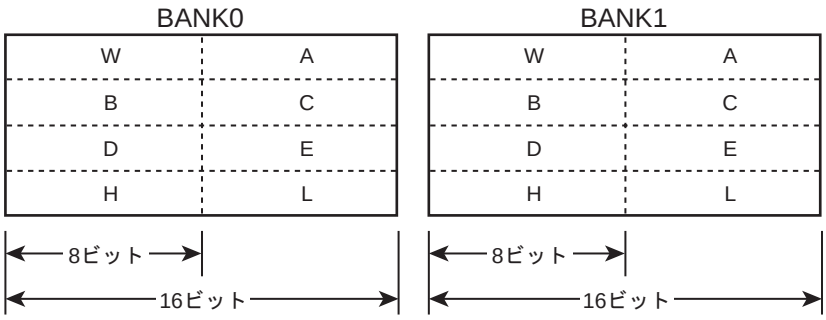


図 1-5 汎用レジスタ

W, A, B, C, D, E, H, L は 8 ビットの転送・演算に、WA, BC, DE, HL は 16 ビットの転送・演算に用いられます。また、汎用レジスタとしての機能のほかに、次の機能を有しています。

1.1.2.1 A レジスタ

A レジスタは、レジスタ間接ビット指定のビット操作命令における、ビット指定レジスタとして使用されます。

（プログラム例）

```
①    SET    (0x56).A      ; 0x00056 番地 ( メモリスタイルが 64K バイトのとき
                            は 0x0056 番地 ) の、A の内容で指定されるビットを、
                            “1” にセットします。

②    CPL    (IX + 0x03).A ; IX に即値 0x03 を符号拡張加算した値で指定される
                            アドレスのメモリの、A の内容で指定されるビットを
                            反転します。
```

また、レジスタオフセット相対アドレッシング (PC + A) のオフセットレジスタとしても使用されます。

（プログラム例）

```
     LD     A, (PC + A)   ; PC に A の内容を符号拡張加算した値で指定される
                            アドレスのメモリの内容を A にロードします。
```

1.1.2.2 C レジスタ

C レジスタは、除算命令における除数レジスタとして使用されます。除算命令実行後に被除数の上位バイトに余りが、被除数の下位バイトに商がそれぞれ格納されます。

（プログラム例）

```
     DIV    WA, C         ; A ← WA ÷ C,  W ← 余り
```

また、レジスタオフセット相対アドレッシング (HL + C) オフセットレジスタとしても使用されます。

（プログラム例）

```
LD      A, (HL + C)     ; HL に C の内容を符号拡張加算した値で指定される
                          アドレスのメモリの内容を A にロードします。
```

1.1.2.3　DE レジスタ

DE レジスタは、レジスタ間接アドレッシングのアドレス指定レジスタとして使用されます。

（プログラム例）

```
LD      A, (DE)         ; DE で指定されるアドレスのメモリの内容を A に
                          ロードします。
```

1.1.2.4　HL レジスタ

HL レジスタは、レジスタ間接アドレッシングのアドレス指定レジスタ, インデックスアドレッシングのインデックスレジスタとして使用されます。

（プログラム例）

```
①   LD      A, (HL)         ; HL で指定されるアドレスのメモリの内容を
                              A にロードします。
②   LD      A, (HL + 0x52)  ; HL に即値 0x52 を符号拡張加算した値で指定される
                              アドレスのメモリの内容を A にロードします。
③   LD      A, (HL + C)     ; HL に C の内容を符号拡張加算した値で指定される
                              アドレスのメモリの内容を A にロードします。
```

1.1.3　16 ビット汎用レジスタ (IX, IY)

TLCS-870/C1 は、2 つの 16 ビット汎用レジスタ IX, IY を各 BANK に 1 組づつ、計 2BANK 内蔵しています。これらは 16 ビット長のレジスタで、レジスタ間接アドレッシングのアドレス指定レジスタ , インデックスアドレッシングのインデックスレジスタとして使用されます。

16 ビット汎用レジスタは、電源投入時及びリセット時の値は不定となります。

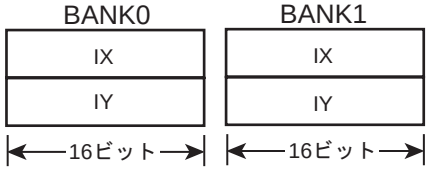


図 1-6　16 ビット汎用レジスタ

(プログラム例)

```
①   LD    A, (IX)          ; IX で指定されるアドレスのメモリの内容を A に
                               ロードします。
②   LD    A, (IY + 0x52)   ; IY に即値 0x52 を符号拡張加算した値で指定される
                               アドレスのメモリの内容を A にロードします。
```

また、16 ビット長の汎用レジスタとして、転送・演算に使用することもできます。

(プログラム例)

```
①   LD    IX, (0x3A)       ; IX に 0x0003A, 0x0003B 番地の内容をロードします。
                               ( メモリスタイルが 64K バイトのときは 0x003A, 0x003B)
②   ADD   IX, 0x5678       ; IX に即値 0x5678 を加算し、結果を IX に格納します。
```

1.1.4　プログラムステータスワード (PSW)

プログラムステータスワードは、

- ジャンプステータスフラグ (JF)
- ゼロフラグ (ZF)
- キャリーフラグ (CF)
- ハーフキャリーフラグ (HF)
- サインフラグ (SF)
- オーバフローフラグ (VF)
- レジスタバンクセレクタ (RBS)

の 7 つのフラグから構成され、SFR 内の 0x0003F 番地 (メモリスタイルが 64K バイトのときは 0x003F) に割り付けられています。

PSW のアクセスは、専用命令で行います。また、メモリアクセス命令で読み出すこともできます。

PSW の構成

PSW (0x003F)	7	6	5	4	3	2	1	0
	JF	ZF	CF	HF	SF	VF	RBS	-

フラグは、命令で指定される条件に従いセット / クリアされます。また、レジスタバンクセレクタと、ハーフキャリーフラグ以外のフラグは条件付きジャンプ命令 [JR cc, a], [JRS cc, a] のジャンプ条件 cc となり得ます。

cc	意味	条件
T	True	JF = 1
F	False	JF = 0
Z	Zero	ZF = 1
NZ	Not zero	ZF = 0
CS	Carry set	CF = 1
CC	Carry clear	CF = 0
VS	Overflow set	VF = 1
VC	Overflow clear	VF = 0
M	Minus	SF = 1
P	Plus	SF = 0
EQ	Equal	ZF = 1
NE	Not equal	ZF = 0
LT	Unsigned less than	CF = 1
GE	Unsigned greater than or equal	CF = 0
LE	Unsigned less than or equal	(CF ∨ ZF) = 1
GT	Unsigned greater than	(CF ∨ ZF) = 0
SLT	Signed less than	(SF ∨ VF) = 1
SGE	Signed greater than or equal	(SF ∨ VF) = 0
SLE	Signed less than or equal	ZF ∨ (SF ∨ VF) = 1
SGT	Signed greater than	ZF ∨ (SF ∨ VF) = 0

（プログラム例）

```
①    LD      PSW, 0x00      ; PSW に 0x00 を書き込む
②    PUSH    PSW            ; PSW の内容をスタックに退避
③    POP     PSW            ; スタックの内容を PSW にリストア
④    LD      A, (0x3F)      ; A レジスタに PSW の内容を転送
⑤    LD      RBS, 1         ; PSW の RBS に 1 を書き込む (BANK1 に切り替え )
⑥    LD      RBS, 0         ; PSW の RBS に 0 を書き込む (BANK0 に切り替え )
⑦    LD      PSW, 0x02      ; PSW の RBS に 1 を書き込む
                            (BANK1 に切り替え。全フラグを 0 にクリア。)
⑧    LD      PSW, 0x00      ; PSW の RBS に 0 を書き込む
                            (BANK0 に切り替え。全フラグを 0 にクリア )
```

LD PSW,n 命令を実行した場合、レジスタバンクの値だけでなく各フラグの値も変更されます。よってフラグの値に影響を与えずレジスタバンクを変更する場合は、LD PSW,n 命令では無く LD RBS,0 または LD RBS,1 命令を使用してください。

(0x3F) に対し、メモリアクセス命令での書き込みを行った場合、データは書き込まれず、その命令で定まった変化をします。

（プログラム例）

```
     LD      (0x3F), 0x00   ; PSW = 0x00 にはならず、[LD (x), n] 命令の指定に従い
                            JF が “1” になり、ほかのフラグは変化しません。
```

割り込み受け付け時、PSW はプログラムカウンタとともにスタックに退避されます。また、割り込みリターン命令 [RETI], [RETN] の実行によりスタック上のデータが PSW にリストアされます。

PSW の RBS を除く各種フラグは、電源投入時及びリセット時の値は不定となります。RBS はリセットで初期化され、レジスタバンクは "0" が選択されます。

1.1.4.1 ゼロフラグ (ZF)

演算結果または転送データが 0x00 (8 ビット演算 / 転送時) / 0x0000 (16 ビット演算時) のとき “1” にセットされ、その他のときは “0” にクリアされます。ビット操作命令では、指定ビットの内容が “0” のとき “1” にセットされ、指定ビットの内容が “1” のとき “0” にクリアされます (ビットテスト)。乗算命令の場合、積の上位 8 ビットが 0x00 のとき、除算命令の場合、剰余が 0x00 のとき、“1” にセットされ、その他のときは “0” にクリアされます。

1.1.4.2 キャリーフラグ (CF)

演算時のキャリーまたはボローがセットされます。除算命令の場合、除数が 0x00 のとき (Divided by zero error)、または商が 0x100 以上のとき (Quotient-overflow error)、“1” にセットされます。シフト / ローテート命令では、レジスタからシフトアウトされるデータがセットされます。

ビット操作命令では、1 ビット長のレジスタ (ブーリアンアキュムレータ) として機能します。また、キャリーフラグ操作命令によりセット / クリア / 反転ができます。

（プログラム例）0x0007 番地のビット 5 の内容と 0x009A 番地のビット 0 の内容とで排他的論理和をとり、結果を 0x0001 番地のビット 2 に書き込みます。
(メモリスタイルが 64K バイトの場合)

```
LD      CF, (0x07).5        ; (0x0001).2 ← (0x0007).5 ∨ (0x009A).0
XOR     CF, (0x9A).0
LD      (0x01).2, CF
```

（プログラム例）0x00007 番地のビット 5 の内容と 0x0009A 番地のビット 0 の内容とで排他的論理和をとり、結果を 0x00001 番地のビット 2 に書き込みます。
(メモリスタイルが 96K, 128K バイトの場合)

```
LD      CF, (0x07).5        ; (0x00001).2 ← (0x00007).5 ∨ (0x0009A).0
XOR     CF, (0x9A).0
LD      (0x01).2, CF
```

1.1.4.3　ハーフキャリーフラグ (HF)

8 ビット演算時、4 ビット目へのキャリーまたは 4 ビット目からのボローがセットされます。HF は、BCD データの加減算の際の十進補正用のフラグです ([DAA　r], [DAS　r] 命令による十進補正)。

（プログラム例）BCD 演算 (A = 0x19, B = 0x28 のとき、次の命令を実行すると、A は 0x47 になります)。

```
ADD     A, B        ; A ← 0x41, HF ← 1, CF ← 0
DAA     A           ; A ← 0x41 + 0x06 = 0x47 ( 十進補正 )
```

1.1.4.4　サインフラグ (SF)

算術演算の演算結果の MSB が “1” のとき、“1” にセットされ、その他のときは “0” にクリアされます。

1.1.4.5　オーバフローフラグ (VF)

算術演算の演算結果にオーバフローが生じたときに “1” にセットされ、その他のときは “0” にクリアされます。例えば、加算命令で 2 つの正の数を加算した結果が負の数になった場合、あるいは 2 つの負の数を加算した結果が正の数になった場合に、VF は “1” にセットされます。

1.1.4.6 ジャンプステータスフラグ (JF)

通常、“1” にセットされるフラグで、命令に従いゼロまたはキャリー情報がセットされ、条件付きジャンプ命令 [JR T/F, a], [JRS T/F, a] (T, F は条件コード) のジャンプ条件となります。

(プログラム例) ジャンプステータスフラグと条件付きジャンプ命令 (メモリスタイルが 64K バイトの場合)

```
INC     A
JRS     T,SLABEL1          ; 直前の演算命令で桁上げが発生した場合ジャンプします。
 :  :    :  :
LD      A,(HL)
JRS     T,SLABEL2          ; 直前の命令で JF は "1" にセットされますので、無条件ジャンプと見
                           なされます。
```

(プログラム例) ジャンプステータスフラグと条件付きジャンプ命令 (メモリスタイルが 96K, 128K バイトの場合)

```
INC     A
JRS     T,code _ addr(SLABEL1)   ; 直前の演算命令で桁上げが発生した場合ジャンプします。
 :  :    :  :
LD      A,(HL)
JRS     T,code _ addr(SLABEL2)   ; 直前の命令で JF は "1" にセットされますので、無条件ジャンプと見
                                 なされます。
```

例 : WA レジスタ , HL レジスタ , データメモリの 0x000C5 番地 , CF, HF, SF, VF の内容がそれぞれ “0x219A”, “0x00C5”, “0xD7”, “1”, “0”, “1”, “0” のとき、下記命令を実行すると、A, WA レジスタおよび各フラグは次のようになります。

命令	実行後のアキュムレータ	PSW					
		JF	ZF	CF	HF	SF	VF
ADDC A, (HL)	72	1	0	1	1	0	1
SUBB A, (HL)	C2	1	0	1	0	1	0
CMP A, (HL)	9A	0	0	1	0	1	0
AND A, (HL)	92	0	0	1	0	1	0
LD A, (HL)	D7	1	0	1	0	1	0
ADD A, 0x66	00	1	1	1	1	0	0
INC A	9B	0	0	1	0	1	0
ROLC A	35	1	0	1	0	1	0
RORC A	CD	0	0	0	0	1	0
ADD WA, 0xF508	16A2	1	0	1	0	0	0
MUL WA	13DA	0	0	1	0	1	0
SET A.5	BA	1	1	1	0	1	0

1.1.4.7 レジスタバンクセレクタ (RBS)

汎用レジスタバンクを選択する 1 ビットのレジスタです。例えば RBS=1 の時、バンク 1 が現在選択されていることになります。リセット時、RBS は "0" に初期化され、バンク 0 が選択されます。RBS を操作する命令としては、即値設定及びプッシュ / ポップの専用命令［LD RBS,n］、

［LD　PSW,n］、［PUSH　PSW］、［POP　PSW］が用意されていますが、［LD　PSW,n］、［PUSH　PSW］命令は RBS に加えフラグも上書きしますので、RBS だけを変更したい場合は［LD RBS,n］命令を使用してください。

割り込み受付時、RBS は PSW の他のフラグと一緒にスタックに退避されます。その際、RBS は値を変更されません。また、割り込みリターン命令［RETI］,［RETN］の実行により、退避されていた PSW の 1 ビットとして復帰され、レジスタバンクも復帰された値にしたがって切り替わります。

1.1.5　スタック、スタックポインタ

1.1.5.1　スタック

スタックは、サブルーチンコール命令実行時または割り込み受け付け時に、その処理ルーチンへジャンプするのに先立ってプログラムカウンタの内容 (戻り番地) やプログラムステータスワードの内容などをセーブするエリアです。

サブルーチンコール命令 [CALL　mn], [CALLV　n] 実行時、戻り番地が (上位バイト、下位バイトの順に) スタックに退避 (プッシュダウン) されます。ソフトウエア割り込み命令 [SWI] 実行時および割り込み受け付け時は、まずプログラムステータスワードの内容がスタックに退避され、次に戻り番地が退避されます。

処理ルーチンから復帰する場合、サブルーチンリターン命令 [RET] を実行することによりスタックからプログラムカウンタへ、割り込みリターン命令 [RETI], [RETN] を実行することによりスタックからプログラムカウンタおよびプログラムステータスワードへ、それぞれの内容がリストア (ポップアップ) されます。

スタックは、データ領域内の任意のエリアに設定できます。

1.1.5.2　スタックポインタ

スタックポインタは、スタックの先頭番地を指す 16 ビットのレジスタです。スタックポインタは、サブルーチンコール、プッシュ命令実行時、および割り込み受け付け時にポストデクリメントされ、リターン、ポップ命令実行時にプリインクリメントされます。従って、スタックはアドレスの若い方に向かって深くなります。

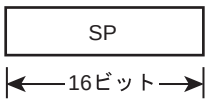


図 1-7　スタックポインタ

スタックのアクセスとスタックのポインタの変化を図 1-8 に示します。

スタックポインタは、ハードウエアリセットで 0x00FF に初期化されます。

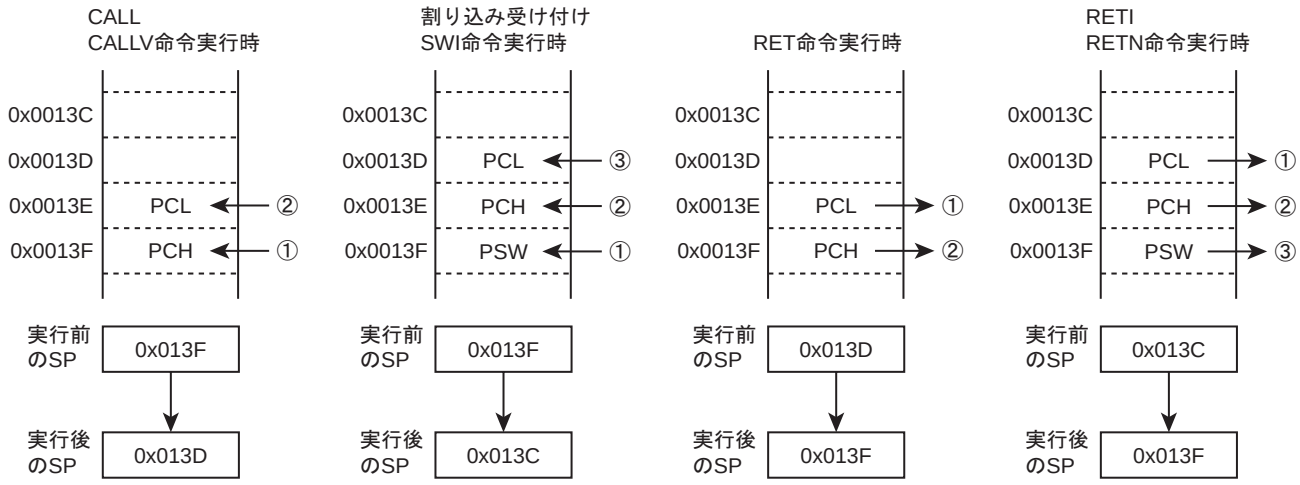
スタックポインタは、インデックスレジスタと同様に転送・演算命令で操作できるほか、インデックスアドレッシングのインデックスレジスタとして使用できます。

(プログラム例)

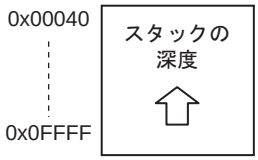
```
①    LD     SP, 0x043F      ; SP ← 0x043F
②    LD     HL, SP          ; HL ← SP
③    LD     SP, SP+0x04     ; SP ← SP + 0x04
④    ADD    SP, 0x5678      ; SP ← SP + 0x5678
```

(プログラム例)

```
⑤          LD          A, (SP+0x12)          ; A ← アドレス (SP + 0x12) のメモリの内容
```



(a) スタックのアクセス例 (プッシュ/ポップ)



(b) スタックの深度

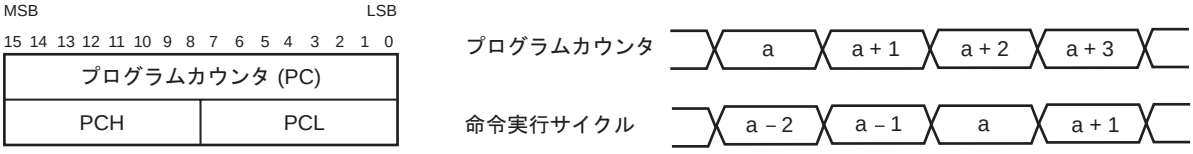
図 1-8 スタック

1.1.6 プログラムカウンタ (PC)

プログラムカウンタは、次に実行すべき命令の格納されているコード領域のアドレスを指す 16 ビットのレジスタです。リセット解除時、ベクタテーブル (0x1FFFF, 0x1FFFE (0xFFFF, 0xFFFE) 番地：MCU モード時) に格納されているリセット ベクタがプログラムカウンタにロードされますので、任意のアドレスからプログラムの実行を開始することができます。例えば、0x1FFFF, 0x1FFFE (0xFFFF, 0xFFFE) 番地にそれぞれ、0xC0, 0x3E が格納されている場合、リセット解除後 0x1C03E (0xC03E) 番地から実行開始します。

TLCS-870/C1 シリーズは、パイプライン処理 (命令先行フェッチ) を行っていますので、プログラムカウンタは、常に 2 アドレス先を指します。例えば、0x1C123 (0xC123) 番地に格納されている 1 バイト命令の実行中、プログラムカウンタの内容は 0x1C125 (0xC125) です。

注） 括弧内はメモリスタイルが 64K バイトのときのアドレスを示します。



(a) 構成

(b) プログラムカウンタと命令実行サイクル

図 1-9 プログラムカウンタ

1.1.6.1　ジャンプ命令とプログラムカウンタの関係

ジャンプ命令には相対ジャンプ命令と絶対ジャンプ命令があり、コード領域内へジャンプします。データ領域へのジャンプはできません。

以下にジャンプ命令とプログラムカウンタの関係の例を示します。

1. 5 ビット相対ジャンプ命令 [JRS cc, $ + 2 + d]
   0x1E8C4 (0xE8C4) : JRS T, $ + 2 + 0x08 の場合
   JF = 1 のとき、プログラムカウンタの内容に 0x08 を加算した 0x1E8CE (0xE8CE) にジャンプします (プログラムカウンタの内容は実行命令の置かれたアドレス +2 になっています。従って、この場合のプログラムカウンタの値は 0x1E8C4 (0xE8C4) + 2 = 0x1E8C6 (0xE8C6) となります)。
2. 8 ビット相対ジャンプ命令 [JR cc, $ + 2 + d]/[JR cc, $ + 3 + d]
   0x1E8C4 (0xE8C4) : JR Z, $ + 2 + 0x80 の場合
   ZF = 1 のとき、プログラムカウンタの内容に 0x1FF80 (0xFF80) (-128) を加算した 0x1E846 (0xE846) にジャンプします。
3. 16 ビット絶対ジャンプ命令 [JP a]
   0x1E8C4 (0xE8C4) : JP 0x1C235 (0xC235) の場合
   無条件に 0x1C235 (0xC235) 番地にジャンプします。絶対ジャンプ命令はコード領域 64 K バイト内の任意のアドレスにジャンプできます。

注) 括弧内はメモリスタイルが 64K バイトのときのアドレスを示します。

1.2 修正履歴

Rev	修正内容
RA001	16 進数の表記を H から 0x に、2 進数の表記を B から 0y に修正しました。
	「図 1-2 コード領域 (メモリスタイル：96K, 128K バイト時)」を修正しました。
	「1.1.4 プログラムステータスワード (PSW)」のプログラム例、文章を修正しました。
	「1.1.4.7 レジスタバンクセレクタ (RBS)」の文章を修正しました。
RA002	「図 1-2 コード領域 (メモリスタイル：96K, 128K バイト時)」の CALLV アドレスに誤りがあったのを修正
	「図 1-4 データ領域 (メモリスタイル：96K, 128K バイト時)」に SFR3 を追加
RA003	最小命令実行時間の記述を削除。最低電源電圧の記述を削除。(製品によって異なるため)
	メモリスタイル別の表記に変更しました。
	「1.1.4.6 ジャンプステータスフラグ (JF)」表中の「実行フラグ」を「PSW」に修正しました。