900, 900/L, 900/H, 900/L1, 900/H2 CPUコアの違いについて

TLCS-900ファミリーには、CPUコアとして① 900, ② 900/L, ③ 900/H, ④ 900/L1, ⑤ 900/H2 の 5 種のCPUコアがあり、それぞれ表1に示す点が異なります。

CPU / 相違内容	900	900/L	900/H, 900/L1	900/H2
最大アドレスバス幅	24ビット	←	←	←
最大データバス幅	16ビット	←	←	32ビット
命令キューバッファ	4バイト	←	←	12バイト
命令セット	TLCS-900	TLCS-900から 以下の命令を削除 NORMAL MAX 以下の命令を追加 MIN	TLCS-900から 以下の命令を削除 NORMAL MAX	TLCS-900から 以下の命令を削除 NORMAL MAX LDX
分岐命令実行時のコードフェッチ	分岐条件が真の時のみ、分岐先コードをフェッチする	←	←	分岐条件に関係なく、常に分岐先コードを先行フェッチする
マイクロDMA	4チャネル	←	←	8チャネル
CPU特権モード	システムモード、 ノーマルモード	システムモードのみ	←	←
CPUレジスタモード	MINモード、 MAXモード (リセット後MINモード)	← (リセット後MAXモード)	MAXモードのみ	←
割り込み方式	リスタート方式	ベクタ方式	←	←
ノーマルスタックポインタ XNSP	あり	なし	←	←
割り込みネスティングカウンタ INTNEST (主にOS用)	なし	あり	←	←

表1 CPUコア 相違点

1. 概　要

TLCS-900シリーズは、東芝オリジナルの高性能16ビットCPUを内蔵しています。このCPUと多様なI/O機能ブロック(タイマ、シリアルI/O、ADなど)をワンチップ化したTLCS-900シリーズは、多彩な応用分野に適用可能です。

TLCS-900のCPUは、16ビットCPUながら、内蔵汎用レジスタは32ビット構成のレジスタバンク方式を採用しているため、特に組み込み制御などに最適です。

TLCS-900のCPUの特長は以下のとおりです。

(1) TLCS-90(東芝オリジナル8ビットCPU)の拡張アーキテクチャ

- 命令ニモニック&レジスタセットのレベルで上位互換

(2) 汎用レジスタマシン

- 8本のレジスタがすべてアキュームレータとして使用可能

(3) レジスタバンク方式(4本のレジスタをバンク化)

- 32ビット幅4バンク

(4) 広大なリニアアドレス空間(16Mバイト)と豊富なアドレッシングモード(9種)

(5) ダイナミックバスサイジングシステム

- 8ビットまたは16ビットの外部データバスが混在可能

(6) 動作モード

- システム/ノーマル両モードのサポート(900)
- システムモードのみ(900/L, 900/H)

(7) 直交性のある豊富な命令セット

- 8/16/32ビットのデータ転送/演算命令
- 16ビット乗除算命令
  16×16→32ビット　(符号なし/付き)
  32÷16→16ビット…16ビット　(符号なし/付き)
- ビット演算を含む多様なビット処理命令
- Cコンパイラ対応命令
- フィルタ演算命令:積和演算/モジュロ増加命令

(8) 高速処理

- 最小命令実行時間:160 ns@25 MHz
- 4バイトの命令キューバッファを内蔵したパイプラインシステム
- 32ビットALU

2. CPUの動作モード

900/Hは、システムモード固定です。システムモードでは使用できる命令およびレジスタの種類に制限はありません。

システムモードでCPUが取り扱い可能なリソースには、以下のものがあります。

1) 汎用レジスタ :

- 4本の32ビット汎用レジスタ×4バンク
- 4本の32ビット汎用レジスタ (スタックポインタ : XSPを含む)

2) ステータスレジスタ (SR)

3) プログラムカウンタ (PC) : 32ビット

4) コントロールレジスタ : マイクロDMAのパラメータ設定レジスタなど

5) CPUの全命令

6) 内蔵I/Oレジスタのすべて

7) 実装されているメモリのすべて

3. レジスタ

3.1 レジスタ構成……16 MBのプログラム空間/16MBのデータ空間

レジスタ構成を図3.1.1 に示します。

汎用レジスタ4本 (32ビット幅)×4バンク
＋
汎用レジスタ4本 (32ビット幅)
＋
プログラムカウンタ (32ビット幅)
＋
ステータスレジスタ

モード切り替えについて

リセットにより、ステータスレジスタSRの<MAX>ビットは、“1”に初期化され、マキシマムモードになります。900/Hは、マキシマムモードで使用してください。

スタックポインタについて

システムモード用レジスタ (XSP) のみ持っており、リセットにより100Hにセットされます。

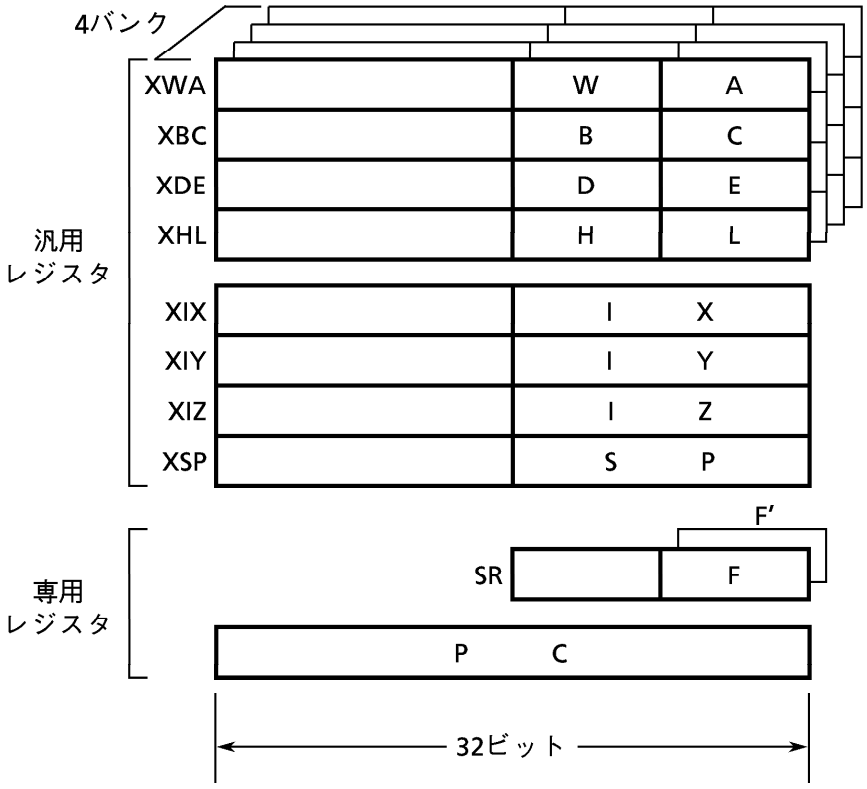


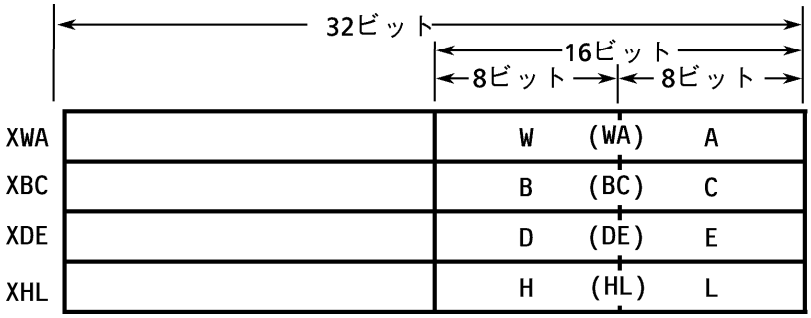
図3.1.1 レジスタ構成 (16MBプログラム)

3.2 レジスタの詳細

3.2.1 汎用バンクレジスタ

マキシマムモードでは、4バンクよりなる下図に示す4本の32ビット汎用レジスタとして使用できます。それぞれのバンクにおけるレジスタ構成は下図のとおりです。

4本の32ビットレジスタ (XWA, XBC, XDE, XHL) は汎用レジスタであり、アキュムレータ、インデックスレジスタとして使用できます。それぞれの32ビットレジスタは16ビットレジスタ (WA, BC, DE, HL) としても使用でき、32ビットレジスタの下位16ビットに割り付けられています。



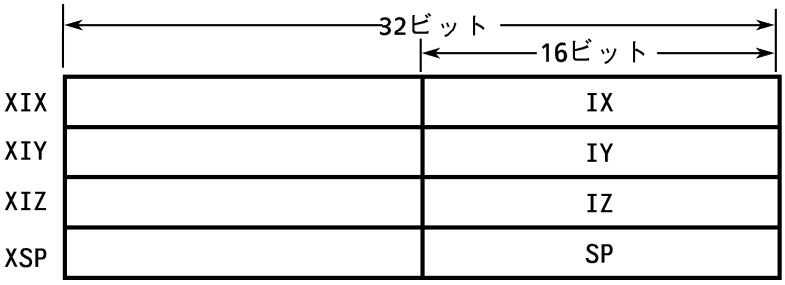
注) ( ) : カッコ内は16ビットレジスタの名称です。

16ビットレジスタはアキュムレータおよびインデックスアドレッシングモードのインデックスレジスタ、ディスプレースメントレジスタとして使用できます。さらに16ビットレジスタは2つの8ビット汎用レジスタ (W, A, B, C, D, E, H, L) としても使用することができ、アキュムレータなどになります。

3.2.2 32ビット汎用レジスタ

4本の32ビット汎用レジスタ (XIX, XIY, XIZ, XSP) があり、レジスタ構成は下記のとおりです。

このレジスタもアキュムレータ、インデックスレジスタ、ディスプレースメントレジスタとして使用でき、16ビットレジスタ/8ビットレジスタ (8ビットレジスタとして使用する場合のレジスタ名称は後述) として取り扱うことができます。



スタックポインタ

レジスタ (XSP) はスタックポインタと呼ばれ、割り込み、CALL命令実行時のスタックポインタとして動作します。

リセットにより、XSPは “100H” に初期化されます。

3.2.3 ステータスレジスタ (SR)

ステータスレジスタには、CPUの状態 (CPU動作モード、レジスタ構成など) と、演算結果の状態を表すフラグが、格納されます。このレジスタは上位バイトと下位バイトの2つの部分で構成されています。 ステータスレジスタの上位バイト (ビット8~15) はCPUの状態を表します。下位バイト (ビット0~7) は通常フラグレジスタ (F) と呼ばれ、演算結果の状態を表しています。TLCS-900は2つのフラグレジスタ (F/F') をもっており、EX命令で交換を行うことができます。

(1) ステータスレジスタ"SR"の上位バイト

15	14	13	12	11	10	9	8
SYSM	IFF2	IFF1	IFF0	MAX	RFP2	RFP1	RFP0

① SYSM (SYStem Mode):

システムモードかノーマルモードかを示します。900/Hでは、システムモードに固定されています。

リセットにより“1” (システムモード) に初期化されます。

0	ノーマルモード
1	システムモード (900/Hはこのモードに固定される)

② IFF2~IFF0 (Interrupt mask F/F 2~0) :

割り込みレベルのマスクレジスタです。割り込み要求レベルは、1~7まであり、7が、最も優先順位の高い割り込み要求レベルです。EI命令を用いて任意の値を設定してください。

リセットにより “111” に初期化されます。割り込みが受け付けられたとき、その割り込みレベルより1だけ高い値がセットされます。

000	レベル1以上の割り込みを許可
001	レベル1以上の割り込みを許可
010	レベル2以上の割り込みを許可
011	レベル3以上の割り込みを許可
100	レベル4以上の割り込みを許可
101	レベル5以上の割り込みを許可
110	レベル6以上の割り込みを許可
111	レベル7の割り込み(ノンマスカブル割り込み)だけを許可

同じです。

ただし、レベル7の割り込み受け付け時は、“111” がセットされます。なお、EI命令はTLCS-90と異なり、実行後直ちに有効となります。

③ MAX (MINimum/MAXimum) :

レジスタバンクのレジスタ幅と、プログラムカウンタPCの幅を指定するビットです。

0	ミニマムモード
1	マキシマムモード (900/Hは、このモードで使用してください)

リセットにより、900/Hでは“1”、(マキシマムモード) に初期化されます。
900/Hはミニマムモードをサポートしていません。“0” をライトしないでください。

④ RFP2～RFP0 (Register File Pointer2～0) :

現在使用しているレジスタファイル (レジスタバンク) のナンバーを示しています。リセットにより “000” に初期化されます。
この値はレジスタバンクの切り替え命令 (3種)で操作することができます。なお、マキシマムモードのとき、“RFP2” は0に固定され、下記命令で “RFP2” が1になるような操作をしても、0のままです。

- LDF imm ; RFP←imm (0～3)(160 ns @25 MHz)
- INCF ; RFP←RFP＋1 (160 ns @25 MHz)
- DECF ; RFP←RFP－1 (160 ns @25 MHz)

(2) フラグレジスタ”F”

7	6	5	4	3	2	1	0	
S	Z	“0”	H	“0”	V	N	C	: R/W

① S (Sign flag)

サインフラグ。
演算結果が負のとき “1” がセットされ、正のとき “0” がセットされます。
(演算結果の最上位ビット (MSB) の値がコピーされます。)

② Z (Zero flag)

ゼロフラグ。
演算結果がゼロのとき“1”にセットされ、それ以外のときは “0” にセットされます。

③ H (Half carry flag)

ハーフキャリーフラグ。
演算の結果、ビット3からビット4へキャリーまたはボローが発生したとき “1” にセットされ、それ以外のときは “0” にセットされます。ただし、32ビット演算命令の場合は、不定値がセットされます。

④ V (parity/over-flow flag)

パリティ/オーバフローフラグ

演算の種類によってパリティを示す場合とオーバフローを示す場合があります。

パリティ (P): 演算の結果、1にセットされているビットの数が奇数のとき“0”にセットされ、偶数のとき“1”にセットされます。ただし、32ビット演算命令の場合は、不定値がセットされます。

オーバフロー (V): 演算の結果、オーバフローなしのとき“0”にセットされ、オーバフローのとき“1”にセットされます。

⑤ N (Negative)

ADD/SUBフラグ。

加算 (ADDなど) 命令実行後“0”にセットされ、減算 (SUBなど) 命令実行後“1”にセットされます。

このフラグは、DAA (10進補正) 命令実行時に使用されます。

⑥ C (Carry)

キャリーフラグ。

演算の結果、キャリーまたはボローが発生したとき“1”にセットされ、それ以外のときは“0”にセットされます。

ステータスレジスタのリード/ライト方法

ビット0~15のリード	① PUSH SR POP dst
ビット0~15のライト	① POP SR
ビット15のみ <SYSM>	システムモード固定のため常に“1”にセットされています。
ビット14~12のみ <IFF2 : 0>	① EI num numの値がライトされます。
ビット11のみ <MAX>	マキシマムモード (“1”にセット) で使用してください。 “0”をライトしないでください。
ビット10~8のみ <RFP2 : 0>	① LDF imm ② INCF ③ DECF
ビット7~0のみ	① PUSH F/POP F ② EX F, F’ ③ 演算命令などの実行により間接的にフラグがセットされます。

3.2.4 プログラムカウンタ (PC)

プログラムカウンタは、次に実行するメモリアドレスを示すポインタです。マキシマムモードでは、32ビットのPCとなりますが、プログラム空間は各製品のアドレスピンの数によって決まります。アドレスピンが24本 (A0~A23)の場合には、最大16M バイトのプログラム空間がリニア空間としてアクセス可能となります。この場合には、PCの上位8ビット (bit 24~31)は無視されます。

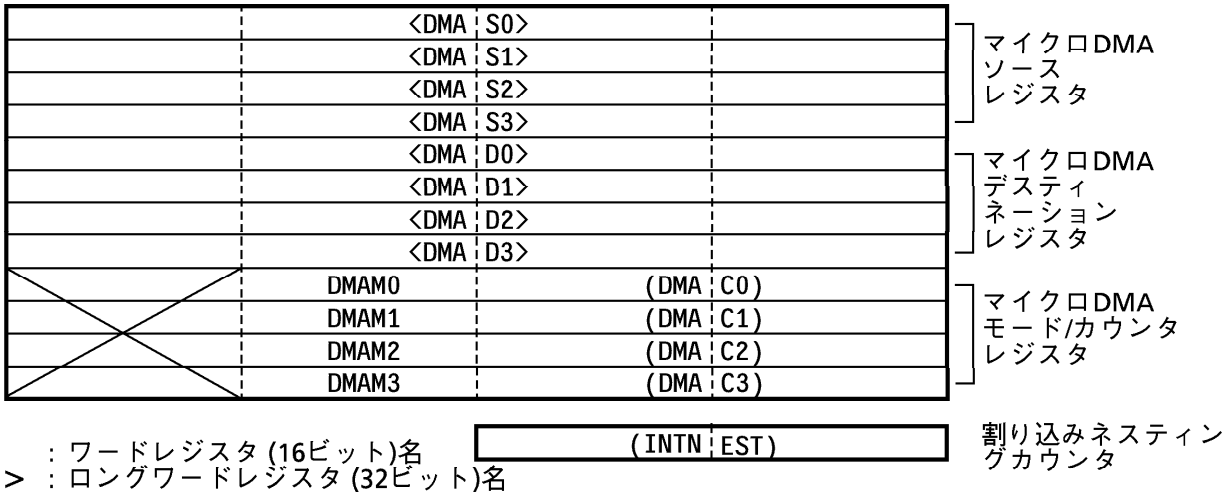
リセット後のPC

リセットによりベクタベース番地 (FFFF00H) に格納されているリセットベクタをリードした後、その値をPCにセットし、そのベクタ以降のプログラムをリードし、実行します。

3.2.5 コントロールレジスタ (CR)

コントロールレジスタは、マイクロDMAの動作を制御するレジスタと、割り込みのネスト値をカウントする割り込みネスティングカウンタより成っています。コントロールレジスタは、LDC命令で、アクセスできます。

コントロールレジスタは以下のとおりです。



( ) ：ワードレジスタ (16ビット)名
< > ：ロングワードレジスタ (32ビット)名

マイクロDMAについてはTLCS-900/H個別製品の説明の章に詳細が記載されています。

3.3 レジスタバンクの切り替え

レジスタバンクは以下の3つに大別されます。

カレントバンクレジスタ
プレビアスバンクレジスタ
アブソリュートバンクレジスタ

カレントバンクは、レジスタファイルポインタ＜RFP＞(ステータスレジスタ：SRのビット8~10)が指し示しているレジスタバンクです。カレントバンクにあるレジスタは、前節で説明した汎用レジスタとして、使用されます。この＜RFP＞の内容を変更することにより、他のレジスタバンクがカレントレジスタバンクとなります。

プレビアスバンクは、レジスタファイルポインタ＜RFP＞から“1”を減じた値により指し示される、レジスタバンクです。例えば、カレントバンクが“バンク3”のときには、“バンク2”がプレビアスバンクとなります。プレビアスバンクのレジスタは、“ダッシュ”の付いたレジスタ名(WA', BC', DE', HL'など)で取り扱われ、カレントバンクとの入れ替えをEX命令(EX A, A'など)で行うことができます。

すべてのバンクレジスタ(カレントバンク、プレビアスバンクを含む)には、バンクを表す数値(アブソリュートバンクナンバー)が付けられており、この数値付きのレジスタ名を使って、全バンクのレジスタを使用することができます。この方法でアクセスできるレジスタ(全レジスタ)を、アブソリュートバンクレジスタと呼びます。

TLCS-900のCPUは、カレントバンクのレジスタをワーキングレジスタとして動作させたときに、最大のパフォーマンスを出すように、設計されています。上述のとおり、CPUは他バンクのレジスタを使用することも可能ですが、他バンクのレジスタを使用すると、パフォーマンスが若干低下します。CPUの最大効率を引き出すために、TLCS-900では、レジスタバンクを簡単に切り替える機能が付いています。

このバンク切り替え機能により

- CPUは最大効率で動作可能となる。
- プログラムのコード量低減が可能となる。
- 特に、割り込みサービスルーチンのコンテキストスイッチとして使用した場合に、応答スピード/コード量で有利。

などの優位性が得られます。

バンクの切り替えは、下記の命令を用いて行います。

LDF imm : イミディエートの内容を＜RFP＞にセット imm：0~3
INCF : ＜RFP＞を“1”増加させる。
DECF : ＜RFP＞を“1”減少させる。

LDF命令のイミディエート値は、0~3になります。また、INCF/DECF命令を用いて、キャリー/ボローが発生したときには、そのキャリー/ボローは無視され、<RFP>の値は巡回されます。例えば、“バンク3”でINCFを実行すると“バンク0”となり、“バンク0”でDECFを実行すると“バンク3”になります。従って、不用意なINCF/DECF命令は、レジスタバンクの内容を破壊する恐れがあります。

- レジスタバンクの使用例

TLCS-900のレジスタはバンク構成をとっており、各バンクは処理用途や割り込みレベルによって使い分けることができます。以下にその例を示します。

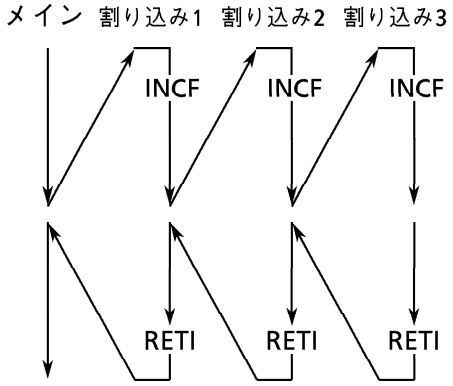
<例1>各レジスタバンクを、それぞれの割り込み処理ルーチン専用に割り当てる使用例。

レジスタバンク0=メインプログラムと下記以外の割り込み処理用
レジスタバンク1=INT0　処理専用
レジスタバンク2=タイマ0　処理専用
レジスタバンク3=タイマ1　処理専用

例えば、メインプログラム実行中、タイマ1の割り込みが発生し、その処理ルーチンへ分岐した場合、その処理ルーチンは、下記のようになり、レジスタのPUSH/POPが、不要になります。

```
LDF 3  ;レジスタバンクを“3”にする。0.16 μs (@25 MHz)
 :
 :
 :
RETI   ;<RFP>(レジスタバンク)も含めて割り込み以前の状態に戻す。
         0.96 μs (@25 MHz)
```

＜例2＞各レジスタバンクを、それぞれの割り込みレベルのネスティング専用に割り当てる例。



(注1) この使用例では、割り込みのネスティングが、レジスタバンクの数(4)を超えた場合、＜RFP＞は、“000”になり、レジスタバンク0の内容を破壊します。

(注2) INCF命令は、＜RFP＞←＜RFP＞＋1を行う命令です。0.16 μs (@25 MHz)

3.4 汎用レジスタのアクセス

TLCS-900の命令は、バイト単位で可変長になっています。カレントバンクのレジスタは、最短コード長でアクセスできます。また、命令コードが1バイト長くなりますが、すべての汎用レジスタをアクセスすることも可能です。“すべての汎用レジスタ”とは、下記のとおりです。

① カレントバンクの汎用レジスタ

QW	(Q	WA)	QA	<X	WA >	W	(W	A) A
QB	(Q	BC)	QC	<X	BC >	B	(B	C) C
QD	(Q	DE)	QE	<X	DE >	D	(D	E) E
QH	(Q	HL)	QL	<X	HL >	H	(H	L) L

( ) ：ワードレジスタ (16ビット)名
< > ：ロングワードレジスタ (32ビット)名

② プレビアスバンクの汎用レジスタ

QW’	(Q	WA’)	QA’	<X	WA’>	W’	(W	A’) A’
QB’	(Q	BC’)	QC’	<X	BC’>	B’	(B	C’) C’
QD’	(Q	DE’)	QE’	<X	DE’>	D’	(D	E’) E’
QH’	(Q	HL’)	QL’	<X	HL’>	H’	(H	L’) L’

③ 32ビット汎用レジスタ

QIXH	(Q	IX)	QIXL	<X	IX>	IXH	(I	X) IXL
QIYH	(Q	IY)	QIYL	<X	IY>	IYH	(I	Y) IYL
QIZH	(Q	IZ)	QIZL	<X	IZ>	IZH	(I	Z) IZL
QSPH	(Q	SP)	QSPL	<X	SP>	SPH	(S	P) SPL

④ アブソリュートバンクレジスタ

QW0	(QWA	0)	QA0	<XWA	0>	RW0	(RWA	0)	RA0
QB0	(QBC	0)	QC0	<XBC	0>	RB0	(RBC	0)	RC0
QD0	(QDE	0)	QE0	<XDE	0>	RD0	(RDE	0)	RE0
QH0	(QHL	0)	QL0	<XHL	0>	RH0	(RHL	0)	RL0
QW1	(QWA	1)	QA1	<XWA	1>	RW1	(RWA	1)	RA1
QB1	(QBC	1)	QC1	<XBC	1>	RB1	(RBC	1)	RC1
QD1	(QDE	1)	QE1	<XDE	1>	RD1	(RDE	1)	RE1
QH1	(QHL	1)	QL1	<XHL	1>	RH1	(RHL	1)	RL1
QW2	(QWA	2)	QA2	<XWA	2>	RW2	(RWA	2)	RA2
QB2	(QBC	2)	QC2	<XBC	2>	RB2	(RBC	2)	RC2
QD2	(QDE	2)	QE2	<XDE	2>	RD2	(RDE	2)	RE2
QH2	(QHL	2)	QL2	<XHL	2>	RH2	(RHL	2)	RL2
QW3	(QWA	3)	QA3	<XWA	3>	RW3	(RWA	3)	RA3
QB3	(QBC	3)	QC3	<XBC	3>	RB3	(RBC	3)	RC3
QD3	(QDE	3)	QE3	<XDE	3>	RD3	(RDE	3)	RE3
QH3	(QHL	3)	QL3	<XHL	3>	RH3	(RHL	3)	RL3

バンク0 / バンク1 / バンク2 / バンク3

( ) ：ワードレジスタ (16ビット)名
< > ：ロングワードレジスタ (32ビット)名

4. アドレッシングモード

TLCS-900には、9種類のアドレッシングモードがあります。これらは、ほとんどの命令と組み合わされて、CPUの処理能力を向上させています。

アドレッシングモードは、下記のようになっています。これは、TLCS-90のアドレッシングモードのすべてを包含しています。

No.	アドレッシングモード	記　述
1.	レジスタ	reg8 reg16 reg32
2.	イミディエート	n8 n16 n32
3.	レジスタ間接	(reg)
4.	レジスタ間接 プリデクリメント	(– reg)
5.	レジスタ間接 ポストインクリメント	(reg +)
6.	インデックス	(reg + d8) (reg + d16)
7.	レジスタインデックス	(reg + reg8) (reg + reg16)
8.	絶対 (ダイレクトアドレッシング)	(n8) (n16) (n24)
9.	相対	(PC + d8) (PC + d16)

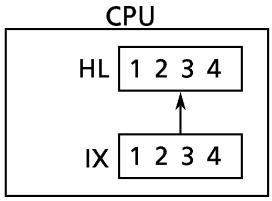
reg 8　：W, A, B, C, D, E, H, L, などの全8ビットレジスタ
reg 16　：WA, BC, DE, HL, IX, IY, IZ, SP, などの全16ビットレジスタ
reg 32　：XWA, XBC, XDE, XHL, XIX, XIY, XIZ, XSP, などの全32ビットレジスタ
reg　：XWA, XBC, XDE, XHL, XIX, XIY, XIZ, XSP, などの全32ビットレジスタ
d8　：8ビットディスプレースメント　(−80H〜+7FH)
d16　：16ビットディスプレースメント (−8000H〜+7FFFH)
n8　：8ビット定数　(00H〜FFH)
n16　：16ビット定数　(0000H〜FFFFH)
n32　：32ビット定数　(00000000H〜FFFFFFFFH)

(注1) 相対アドレッシングモードは、下記の命令でのみ使用可能です。
LDAR, JR, JRL, DJNZ, CALR

(1) レジスタ

オペランドは、指定されたレジスタになります。

例：LD HL, IX

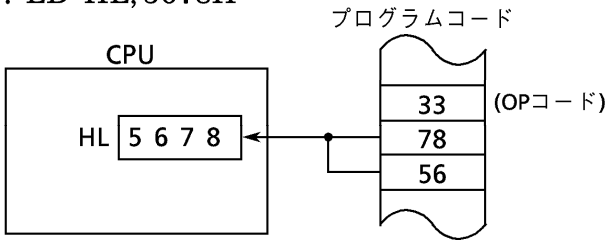


IXレジスタの内容1234Hが、HLレジスタへロードされます。

(2) イミディエート

オペランドは、その命令コード中になります。

例：LD HL, 5678H

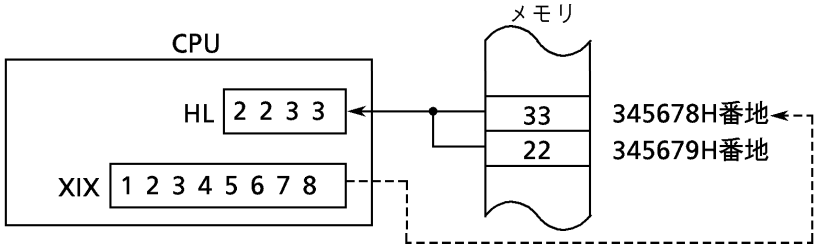


イミディエートデータ5678Hが、HLレジスタへロードされます。

(3) レジスタ間接

オペランドは、レジスタの内容で指定されるメモリ番地になります。

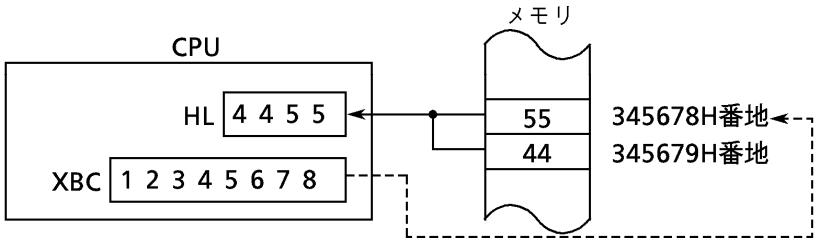
例1：LD HL, (XIX)



345678H番地のメモリデータ2233Hが、HLレジスタへロードされます。

バンクレジスタ(XWA, XBC, XDE, XHLレジスタ)をアドレッシングに使うと、ビット0～23の値が、そのままアドレスバスに出力されます。

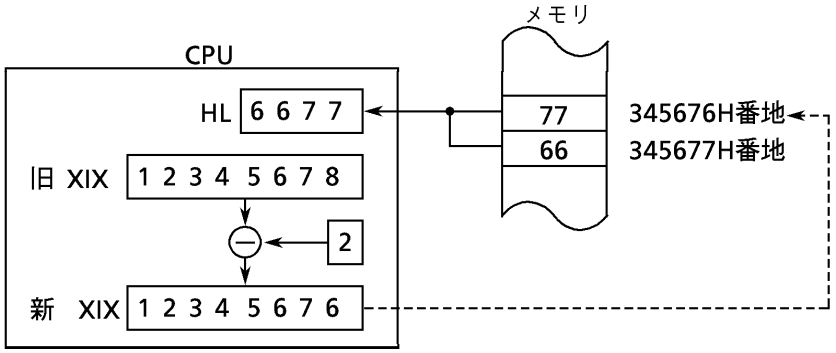
例2 : LD HL, (XBC)



(4) レジスタ間接プリデクリメント

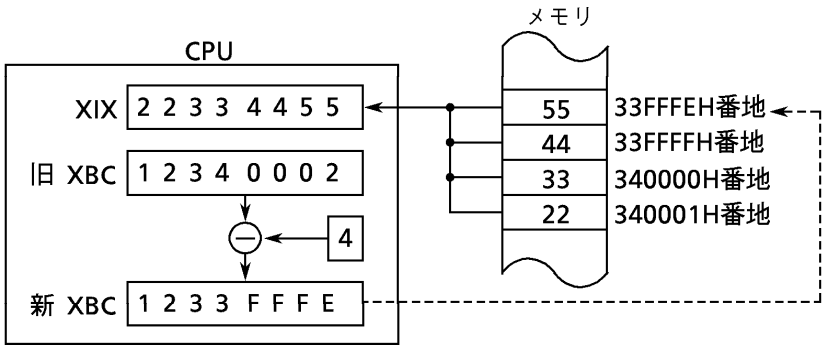
まず、レジスタ内容がオペランドサイズ分、減算されます。そして、オペランドは、減算されたレジスタの内容で指定されるメモリ番地になります。

例1 : LD HL, (−XIX)



プリデクリメント数は、下記のようになります。
オペランドサイズが バイト (8ビット) のとき＝−1
オペランドサイズが ワード (16ビット) のとき＝−2
オペランドサイズが ロング (32ビット) のとき＝−4

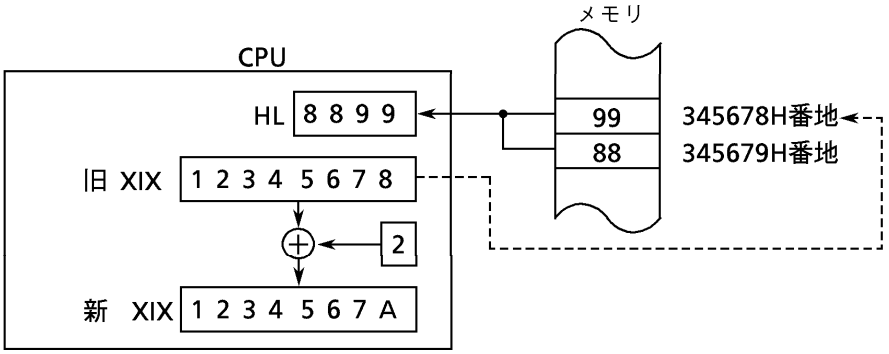
例2 : LD XIX, (−XBC)



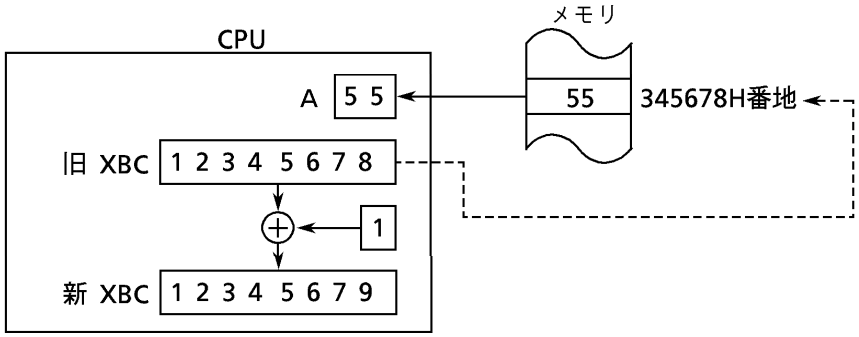
(5) レジスタ間接ポストインクリメント

オペランドは、レジスタの内容で指定されるメモリ番地になります。その後、レジスタの内容がオペランドサイズ分、加算されます。

例1 : LD HL, (XIX+)



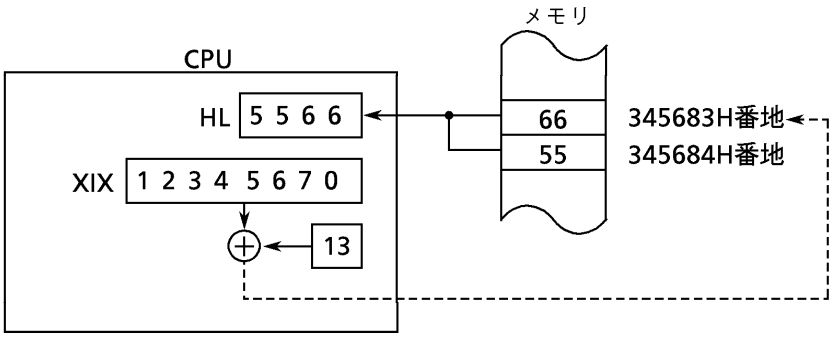
例2 : LD A, (XBC+)



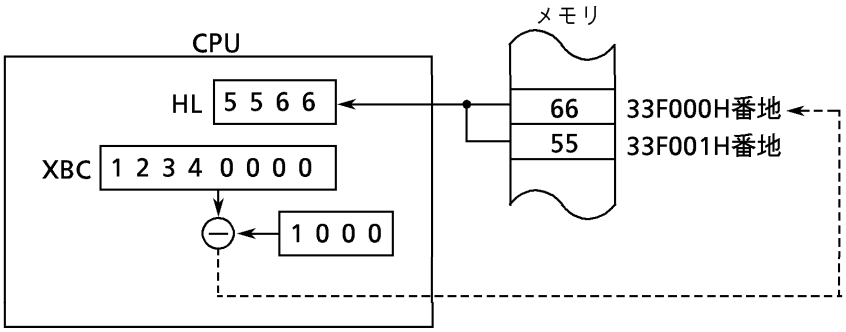
(6) インデックス

オペランドは、命令コード中の8ビットまたは16ビットのディスプレースメント値と、指定されたレジスタの内容を加算して得られるメモリ番地になります。

例1 : LD HL, (XIX+13H)



例2 : LD HL, (XBC−1000H)

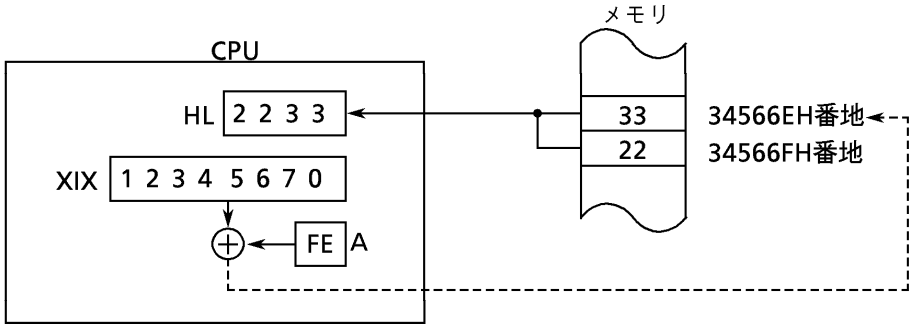


なお、ディスプレースメント値の範囲は、−8000H～+7FFFHに限られます。

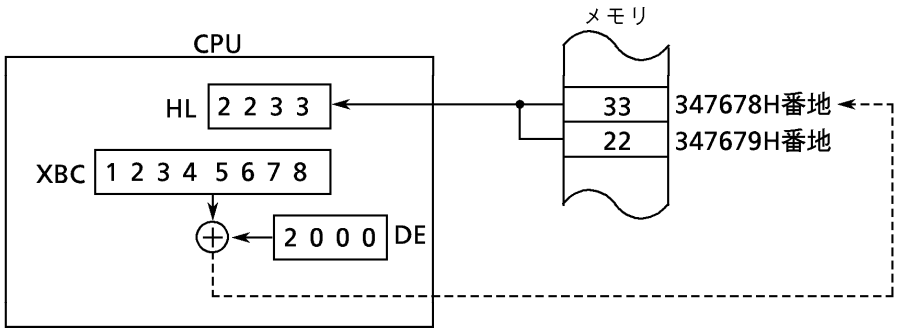
(7) レジスタインデックス

オペランドは、ディスプレースメント(符号付の8ビットまたは16ビット整数)に指定されたレジスタと、ベースに指定されたレジスタの内容を加算して得られるメモリ番地になります。

例1 : LD HL,(XIX+A)



例2 : LD HL,(XBC+DE)



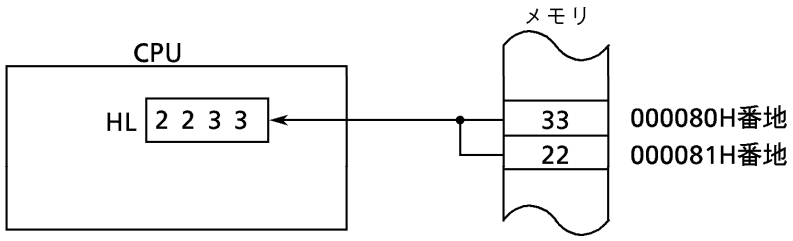
なお、ディスプレースメント値の範囲は、8ビットの場合−80H～+7FH、16ビットの場合−8000H～+7FFFHになります。

(8) 絶対

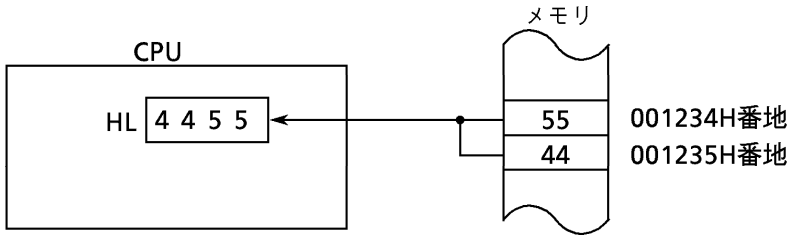
オペランドは、命令コード中の1~3バイトで指定されるメモリ番地になります。000000H~0000FFH 番地は、1バイトで指定できます。000000H~00FFFFH番地は、2バイトで指定でき、000000H~FFFFFFH番地は、3バイトで指定できます。

この中で、1バイトで指定できるメモリ領域 (0H~FFH) の256バイト空間へのアドレッシングを、ダイレクトアドレッシングモードと呼んでいます。ダイレクトアドレッシングモードは、プログラムメモリが節約でき、実行時間も短縮できます。

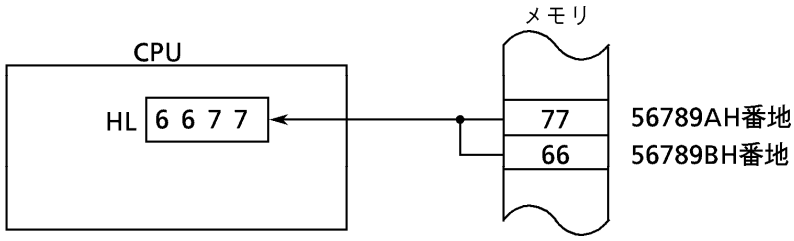
例1 : LD HL, (80H)



例2 : LD HL, (1234H)



例3 : LD HL, (56789AH)



(9) 相対

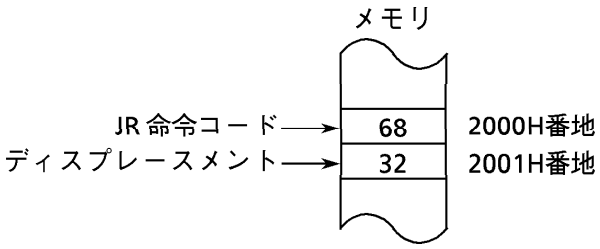
オペランドは、現在実行中の命令コードのある番地と、8ビットまたは16ビットのディスプレースメント値を加算して得られるメモリ番地になります。

このアドレッシングモードを使用できる命令は、限定されており、下記の5種類です。

```
LDAR  R,$+4+d16
JR    cc,$+2+d8
JRL   cc,$+3+d16
CALR  $+3+d16
DJNZ  r,$+3+d8                    ($:命令コードの先頭番地)
```

なお、ディスプレースメントのオブジェクトコード値の計算は、命令の種類によって、補正値が異なり (+2～+4) ます。

例1 : JR 2034H



上記の例での、ディスプレースメントのオブジェクトコード値は、

2034H－(2000H＋2)

であり、32Hとなります。

5. 命　令

TLCS-900には、豊富なアドレッシングモードとともに、強力な命令セットを持っています。

基本命令は、大別して以下の9グループに分類されます。

- 転送命令 (8/16/32ビット)
- 交換命令 (8/16ビット)
- ブロック転送/ブロックサーチ命令 (8/16ビット)
- 算術演算命令 (8/16/32ビット)
- 論理演算命令 (8/16/32ビット)
- ビット操作命令 (1ビット)
- 特別演算、CPU制御命令
- ローテート、シフト命令 (8/16/32ビット)
- ジャンプ、コール、リターン命令

表5.1 に、TLCS-900の基本命令を示します。
付録Aに「命令の詳細」を、付録Bに「命令一覧表」を、付録Cに「命令コードマップ」を示します。
また、付録Dに「TLCS-90との相違点」を示します。

表5.1 TLSC-900 基本命令

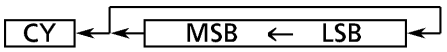
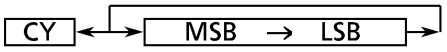
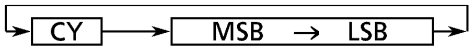
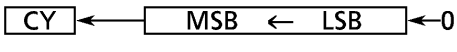
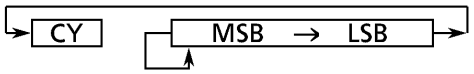
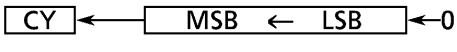
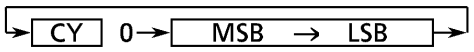
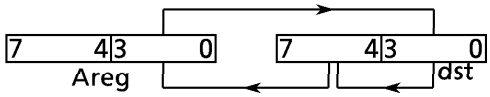
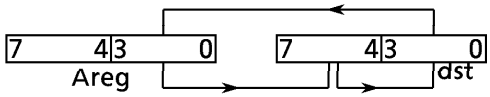
LD	dst, src	データ転送命令。dst←src
PUSH	src	srcのデータをスタックへPUSHします。 SP←SP – size: (SP) ←src
POP	dst	スタックからdstへデータをPOPします。 dst←(SP) : SP←SP + size
LDA	dst, src	srcの実効アドレスをdstにセットします。
LDAR	dst, PC + dd	PC相対アドレス値をdstにセットします。dst←PC + dd
EX	dst1, dst2	dst1とdst2のデータを交換します。
MIRR	dst	dstのビットパターンをミラー反転します。
LDI		ブロック転送命令。
LDIR		ブロック転送命令。
LDD		ブロック転送命令。
LDDR		ブロック転送命令。
CPI		ブロック比較命令。
CPIR		ブロック比較命令。
CPD		ブロック比較命令。
CPDR		ブロック比較命令。
ADD	dst, src	加算命令。 dst←dst + src
ADC	dst, src	拡張加算命令。 dst←dst + src + CY
SUB	dst, src	減算命令。 dst←dst – src
SBC	dst, src	拡張減算命令。 dst←dst – src – CY
CP	dst, src	比較命令。 dst – src
AND	dst, src	論理積命令。 dst←dst AND src
OR	dst, src	論理和命令。 dst←dst OR src
XOR	dst, src	排他的論理和命令。 dst←dst XOR src
INC	imm, dst	増加命令。 dst←dst + imm
DEC	imm, dst	減少命令。 dst←dst – imm
MUL	dst, src	符号なし乗算命令。 dst←dst (low) × src
MULS	dst, src	符号付き乗算命令。 dst←dst (low) × src
DIV	dst, src	符号なし除算命令。 dst (low) ←dst ÷ src dst (high)←余り 0による除算またはオーバフローで、Vフラグがセットされます。
DIVS	dst, src	符号付き除算命令。 dst (low) ←dst ÷ src dst (high)←余り ; 符号は常に被除数と同じです。 0による除算またはオーバフローで、Vフラグがセットされます。

MULA	dst	符号付き積和演算命令。 dst(32bit) ← dst(32bit) + (XDE)(16bit) × (XHL−)(16bit)
MINC1	num, dst	モジュロインクリメント (+1)。
MINC2	num, dst	モジュロインクリメント (+2)。
MINC4	num, dst	モジュロインクリメント (+4)。
MDEC1	num, dst	モジュロデクリメント (−1)。
MDEC2	num, dst	モジュロデクリメント (−2)。
MDEC4	num, dst	モジュロデクリメント (−4)。
NEG	dst	2の補数。dst←0 − dst
CPL	dst	1の補数。dst←not dst
EXTZ	dst	ゼロ拡張。dstの上位データを0にセットします。
EXTS	dst	符号拡張。dstの下位データのMSB値を上位にコピーします。
DAA	dst	10進補正。
PAA	dst	ポインタ補正。dstが奇数の時、+1を加えて偶数にします。 if dst (0) = 1 then dst←dst + 1.
LDCF	bit, src	src<bit>の値を、Cフラグにコピーします。
STCF	bit, dst	Cフラグの値を、dst<bit>へコピーします。
ANDCF	bit, src	src<bit>の値とCフラグのANDをとり、Cフラグへ格納します。
ORCF	bit, src	src<bit>の値とCフラグのORをとり、Cフラグへ格納します。
XORCF	bit, src	src<bit>の値とCフラグのXORをとり、Cフラグへ格納します。
RCF		Cフラグを0に、リセットします。
SCF		Cフラグを1に、セットします。
CCF		Cフラグの値を、反転します。
ZCF		Zフラグの反転値を、Cフラグへコピーします。
BIT	bit, src	ビットテスト命令。Zフラグ←not src<bit>
RES	bit, dst	ビットリセット命令。 dst<bit>←0
SET	bit, dst	ビットセット命令。 dst<bit>←1
CHG	bit, dst	ビットチェンジ命令。 dst<bit>←not dst<bit>
TSET	bit, dst	ビットテスト&セット命令。 Zフラグ ←not dst<bit> dst<bit> ←1

BS1F	A, dst	dst中にある最初の1のビットを、Forward (LSB) からサーチし、そのビット番号をAレジスタにセットします。
BS1B	A,dst	dst中にある最初の1のビットを、Backward (MSB) からサーチし、そのビット番号をAレジスタにセットします。
NOP		ノーオペレーション。
EI	imm	割り込みフラグの設定をします。 IFF←imm
DI		マスカブル割り込みを禁止します。 IFF←7
PUSH	SR	ステータスレジスタをPUSHします。
POP	SR	ステータスレジスタをPOPします。
SWI	imm	ソフトウエア割り込み。 PUSH PC&SR JP FFFF00H + 10H × imm
HALT		CPUを停止します。
LDC	CTRL – REG, reg	レジスタの内容を、CPUのコントロールレジスタへコピーします。
LDC	reg, CTRL – REG	CPUのコントロールレジスタの内容を、レジスタへコピーします。
LDX	dst, src	抜き取り転送命令。dst←src

LINK	reg, dd	スタックフレームを生成します。
		PUSH reg
		LD reg, XSP
		ADD XSP, dd

UNLK	reg	スタックフレームを削除します。 LD XSP, reg POP reg
LDF	imm	レジスタバンクを指定します。 RFP←imm
INCF		新しいレジスタバンクへ移ります。 RFP←RFP + 1
DECF		以前のレジスタバンクへ戻ります。 RFP←RFP – 1
SCC	cc, dst	コンディションコードによるdstのセットを行います。 if cc then dst ←1 else dst ←0.

RLC	num, dst	左ローテート命令(CYなし)。 
RRC	num, dst	右ローテート命令(CYなし)。 
RL	num, dst	左ローテート命令(CYあり)。 
RR	num, dst	右ローテート命令(CYあり)。 
SLA	num, dst	算術左シフト命令。 
SRA	num, dst	算術右シフト命令。 
SLL	num, dst	論理左シフト命令。 
SRL	num, dst	論理右シフト命令。 
RLD	dst	左ローテートデジット命令。 
RRD	dst	右ローテートデジット命令。 

JR	cc, PC + d	相対ジャンプ命令 (8ビットディスプレースメント)。 if cc then PC←PC + d.
JRL	cc, PC + dd	相対ロングジャンプ命令 (16ビットディスプレースメント)。 if cc then PC←PC + dd.
JP	cc, dst	ジャンプ命令。 if cc then PC←dst.
CALR	RC + dd	相対コール命令 (16ビットディスプレースメント)。 PUSH PC PC←PC + dd.
CALL	cc, dst	コール命令。 if cc then PUSH PC: PC←dst.
DJNZ	dst, PC + d	デクリメント&相対ジャンプ命令。 dst←dst – 1 if dst≠0 then PC←PC + d.
RET	cc	リターン命令。 if cc then POP PC.
RETD	dd	リターン&引き数領域の削除命令。 RET XSP←XSP + dd
RETI		割り込み用リターン命令。 POP SR&PC

表5.2 命令一覧

BWL	LD	reg, reg	BWL	INC	imm3, reg	---	NOP	
BWL	LD	reg, imm		DEC	imm3, mem.B/W			
BWL	LD	reg, mem				---	EI	[imm3]
BWL	LD	mem, reg				---	DI	
BW-	LD	mem, imm	BW-	MUL	reg, reg	-W-	*PUSH	SR
BW-	LD	(nn), mem		*MULS	reg, imm	-W-	*POP	SR
BW-	LD	mem, (nn)		DIV	reg, mem	---	SWI	[imm3]
				*DIVS		---	HALT	
BWL	PUSH	reg/F				BWL	*LDC	CTRL – R, reg
BW-	PUSH	imm	-W-	*MULA	reg	BWL	*LDC	reg, CTRL – R
BW-	PUSH	mem				B--	*LDX	(n), n
			-W-	*MINC1	imm, reg			
BWL	POP	reg/F	-W-	*MINC2	imm, reg	--L	*LINK	reg, dd
BW-	POP	mem	-W-	*MINC4	imm, reg	--L	*UNLK	reg
			-W-	*MDEC1	imm, reg	---	*LDF	imm3
			-W-	*MDEC2	imm, reg	---	*INCF	
-WL	LDA	reg, mem	-W-	*MDEC4	imm, reg	---	*DECF	
-WL	LDAR	reg, PC + dd				BW-	*SCC	cc, reg
			BW-	NEG	reg			
			BW-	CPL	reg	BWL	RLC	imm, reg
			-WL	*EXTZ	reg		RRC	A, reg
B--	EX	F, F'	-WL	*EXTS	reg		RL	mem. B/W
BW-	EX	reg, reg	B--	DAA	reg		RR	
BW-	EX	mem, reg	-WL	*PAA	reg		SLA	
							SRA	
							SLL	
-W-	*MIRR	reg	BW-	*LDCF	imm, reg		SRL	
				*STCF	A, reg			
				*ANDCF	imm, mem.B	B--	RLD	[A,] mem
				*ORCF	A, mem.B	B--	RRD	[A,] mem
BW-	LDI			*XORCF				
BW-	LDIR							
BW-	LDD		---	RCF		---	JR	[cc,] PC + d
BW-	LDDR		---	SCF		---	JRL	[cc,] PC + dd
			---	CCF		---	JP	[cc,] mem
			---	*ZCF		---	CALR	PC + dd
BW-	CPI					---	CALL	[cc,] mem
BW-	CPIR		BW-	BIT	imm, reg			
BW-	CPD			RES	imm, mem.B	BW-	DJNZ	[reg], PC + d
BW-	CPDR			SET				
				*CHG		---	RET	[cc]
				TSET		---	*RETD	dd
BWL	ADD	reg, reg				---	RETI	
	ADC	reg, imm	-W-	*BS1F	A, reg			
	SUB	reg, mem		*BS1B				
	SBC	mem, reg						
	CP	mem, imm.B/W						
	AND							
	OR							
	XOR							

B = Byte (8bit), W = Word (16bit), L = Long-Word (32bit).

＊ : TLCS-90より追加された命令であることを示します。

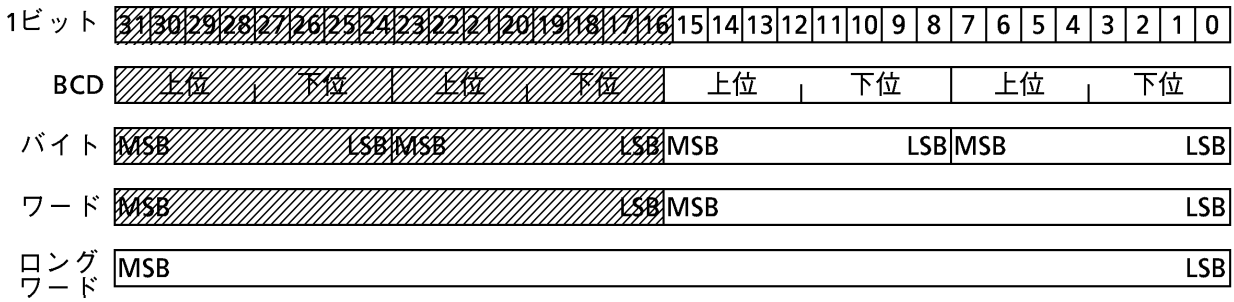
[] : 省略可能を示します。

6. データ構成

TLCS-900は、1/4/8/16/32ビットの各種サイズのデータを扱うことができます。

(1) レジスタのデータフォーマット

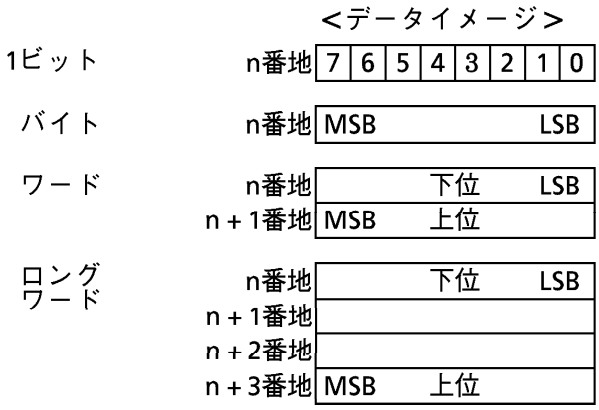
<データイメージ>



(注1) 斜線部分のアクセスは、斜線なしの部分のアクセスに比べて、命令コード長が1バイト長くなります。

(2) メモリのデータフォーマット

<データイメージ>



(注2) メモリ上でワードデータ/ロングワードデータを扱う場合、ワードアライメントの制限はありません。偶数番地または奇数番地のどちらからでも配置できます。

(注3) スタックエリアへデータをセーブする命令(PUSH)を実行した場合、まずスタックポインタをデクリメントした後、データをセーブします。

例: PUSH HL; XSP←XSP－2
(XSP) ←L
(XSP＋1)←H

これは、“プリデクリメント”のアドレッシングモードの場合も同じです。
TLCS-90では、その動作が逆で、セーブした後、スタックポインタをデクリメントします。

例: PUSH HL; (XSP－1)←H
(XSP－2)←L
XSP←XSP－2

(3) ダイナミックバスサイジング

TLCS-900では、8ビットデータバスと16ビットデータバスを、ダイナミック(各バスサイクルごと)に切り替えることができます。これを、ダイナミックバスサイジングと呼びます。

この機能により、TLCS-900では、外部に8ビットデータバス幅のメモリと16ビットデータバス幅のメモリを、混在して拡張することが可能です。

チップセレクト/ウェイトコントローラを内蔵した製品では、それにより各アドレス領域ごとに外部データバス幅の制御を行うことが可能です。

表6.1 ダイナミックバスサイジング

オペランドデータ幅	オペランドスタート番地	メモリ側データ幅	CPUアドレス	CPUデータ	
				D15 – D8	D7 – D0
8ビット	2n + 0 (偶数)	8ビット	2n + 0	xxxxx	b7 – b0
		16ビット	2n + 0	xxxxx	b7 – b0
	2n + 1 (奇数)	8ビット	2n + 1	xxxxx	b7 – b0
		16ビット	2n + 1	b7 – b0	xxxxx
16ビット	2n + 0 (偶数)	8ビット	2n + 0	xxxxx	b7 – b0
			2n + 1	xxxxx	b15 – b8
		16ビット	2n + 0	b15 – b8	b7 – b0
	2n + 1 (奇数)	8ビット	2n + 1	xxxxx	b7 – b0
			2n + 2	xxxxx	b15 – b8
		16ビット	2n + 1	b7 – b0	xxxxx
			2n + 2	xxxxx	b15 – b8
32ビット	2n + 0 (偶数)	8ビット	2n + 0	xxxxx	b7 – b0
			2n + 1	xxxxx	b15 – b8
			2n + 2	xxxxx	b23 – b16
			2n + 3	xxxxx	b31 – b24
		16ビット	2n + 0	b15 – b8	b7 – b0
			2n + 2	b31 – b24	b23 – b16
	2n + 1 (奇数)	8ビット	2n + 1	xxxxx	b7 – b0
			2n + 2	xxxxx	b15 – b8
			2n + 3	xxxxx	b23 – b16
			2n + 4	xxxxx	b31 – b24
		16ビット	2n + 1	b7 – b0	xxxxx
			2n + 2	b23 – b16	b15 – b8
			2n + 4	xxxxx	b31 – b24

xxxxx : リード時は、そのバスの入力データが無視されることを示します。ライト時は、そのバスがハイインピーダンスで、そのバスのライトストローブ信号はノンアクティブのままであることを示します。

(4) 内部データバスの構成

TLCS-900では、CPUと内蔵メモリ (内蔵ROMまたは内蔵RAM) は16ビットの内部データバスで接続されています。内蔵メモリは、0ウェイトで動作します。

一方、内蔵I/Oとは8ビットの内部データバスで接続されています。これは、内蔵I/Oのアクセススピードがほとんどシステム全体の動作スピードに影響を与えないからです。システム全体の動作スピードは、プログラムメモリのアクセススピードに大きく依存します。

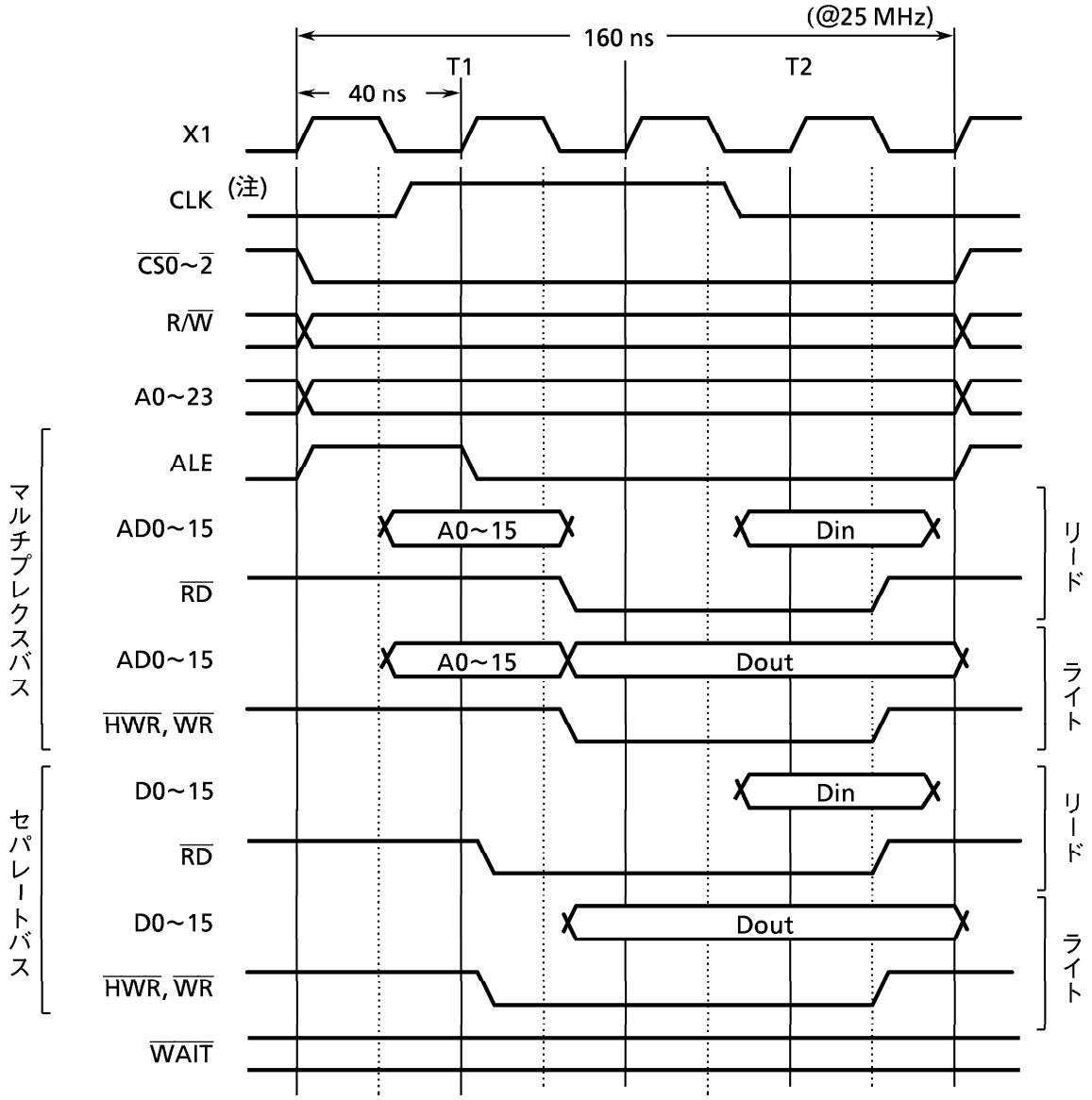
また、内蔵I/OはCLK端子の信号位相と同期して動くようになっており、常に、バスサイクルの中央でCLKが ⎍⎍ になるよう同期がとられます (次ページの図7.1で示されているような位相関係になります)。ALE信号が立ち上がったときにCLKが“1”の場合、自動的に1ウェイトが挿入され、同期がとられます。

7. 基本タイミング

TLCS-900には、以下の基本タイミングがあります。

- リードサイクル
- ライトサイクル
- ダミーサイクル
- 割り込み受け付けタイミング
- リセット

図7.1~7.10 に、それぞれの基本タイミングを示します。



(注) CLK出力は、必ずしも上記の位相であるとは限りません。

図7.1 0 WAITのリードサイクル/ライトサイクル

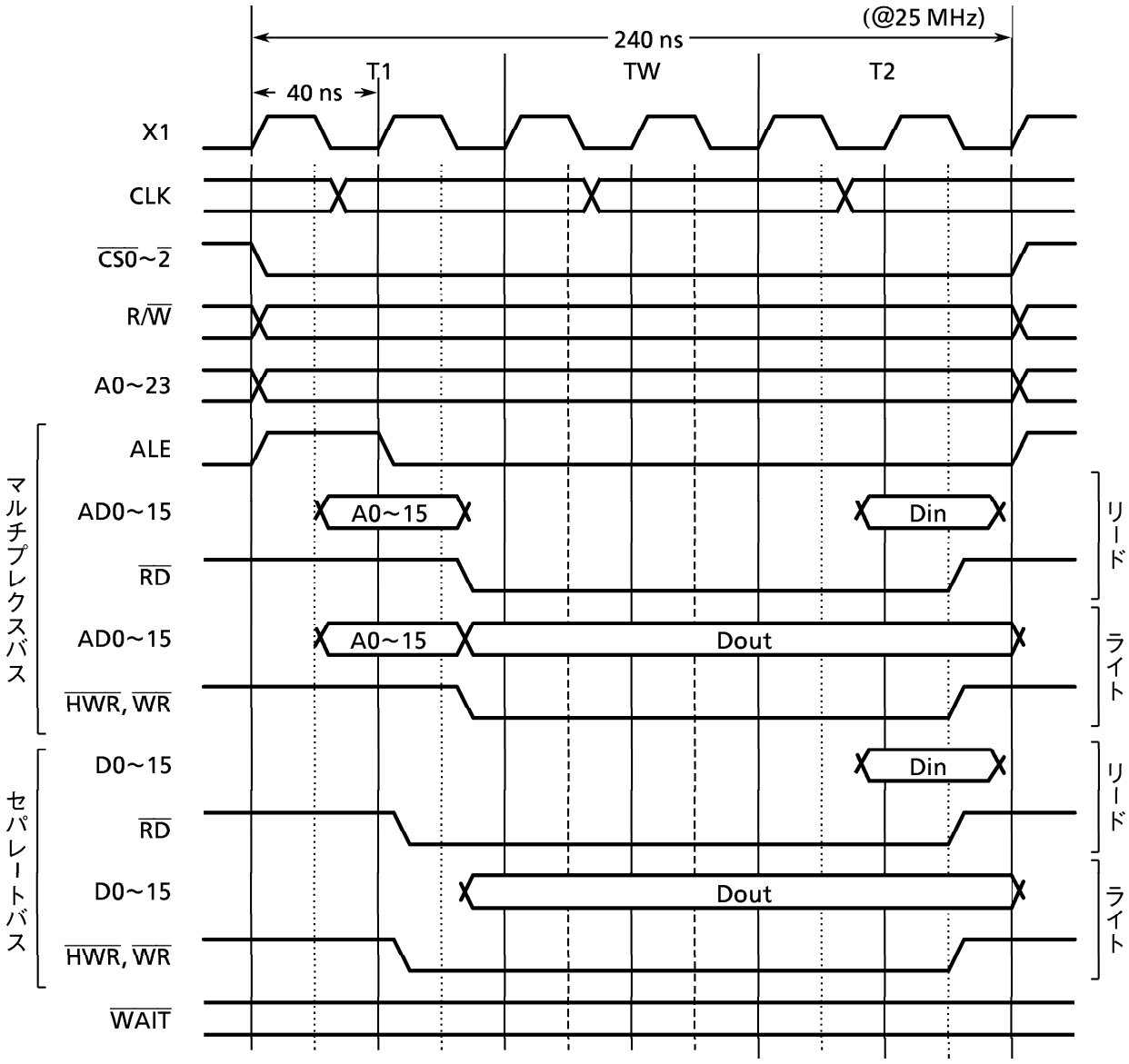


図7.2 1WAITのリード/ライトサイクル

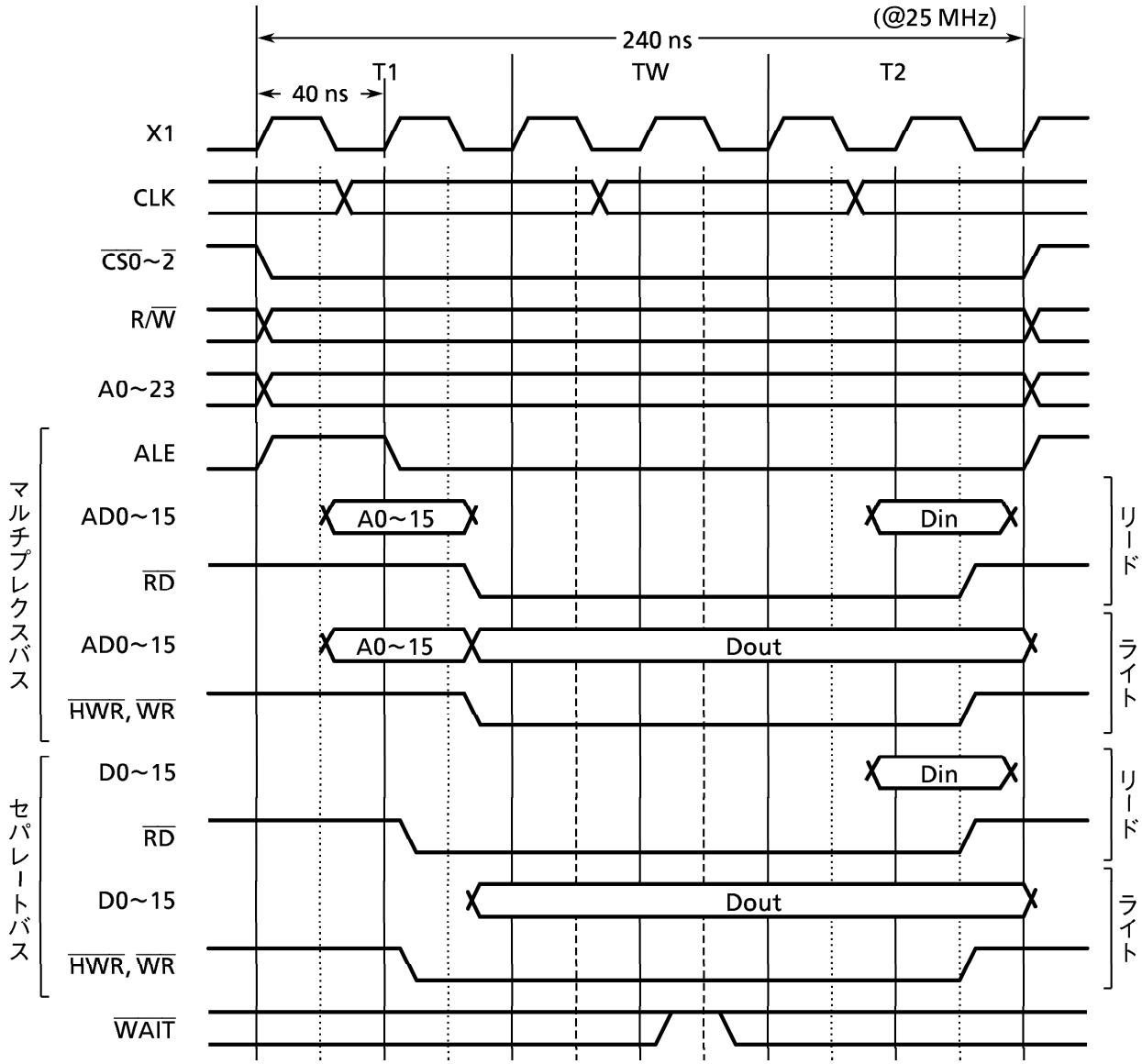


図7.3 1WAIT + n のリード/ライトサイクル (n = 0 の場合)

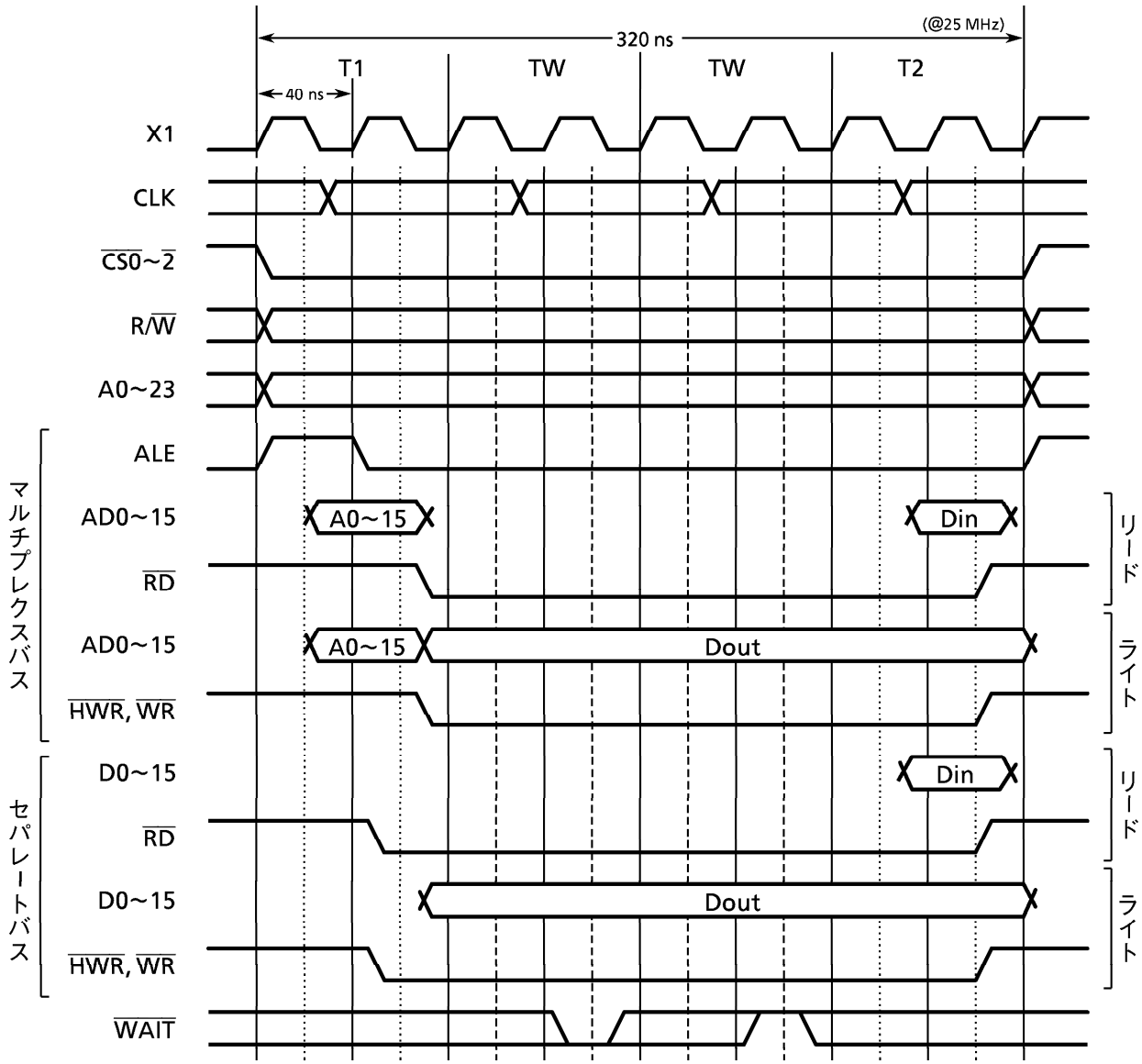


図7.4 1WAIT + n のリード/ライトサイクル (n = 1 の場合)

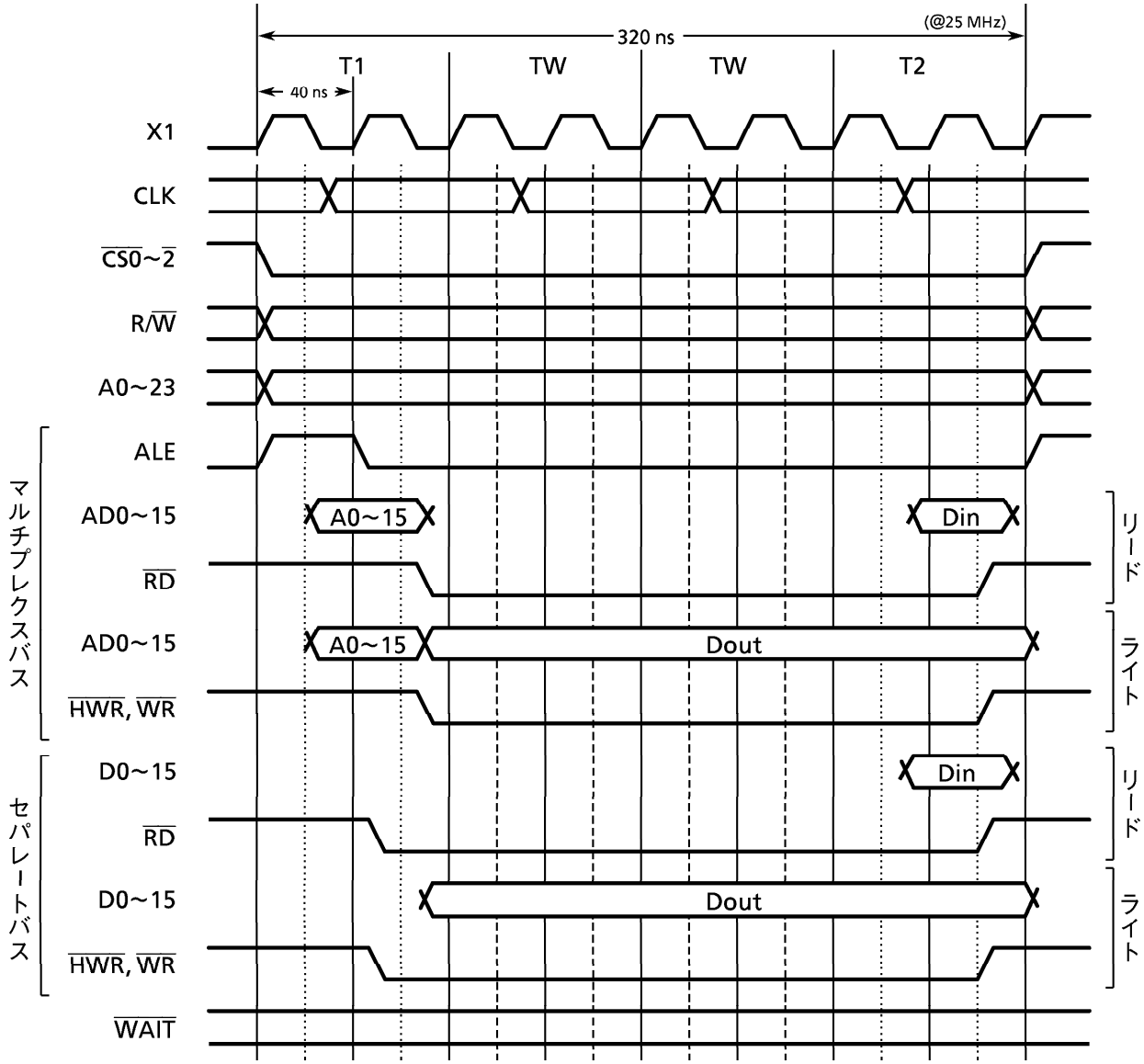


図7.5 2WAITのリード/ライトサイクル

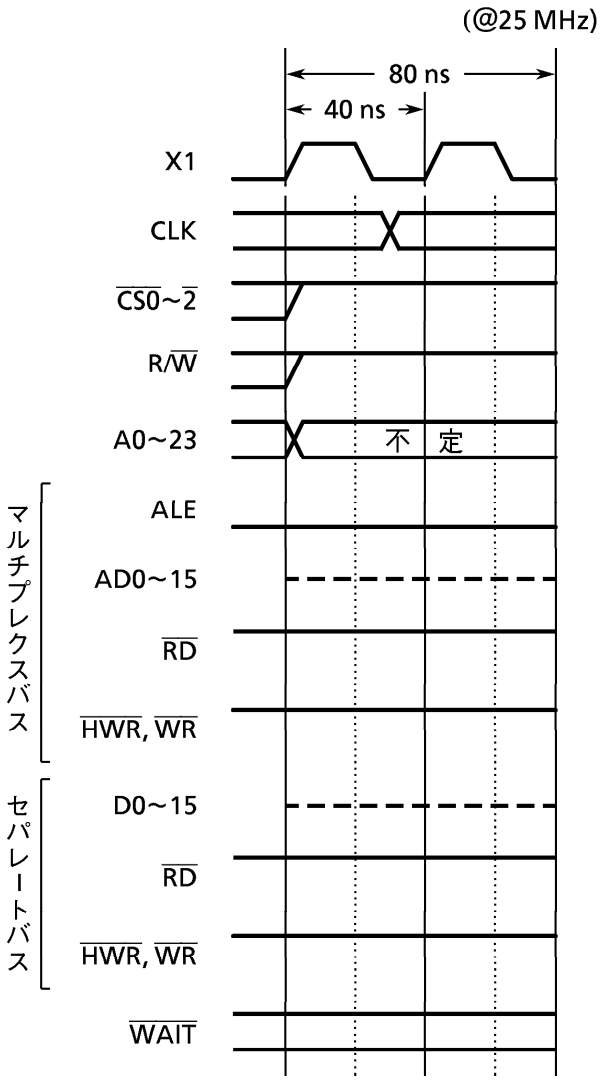
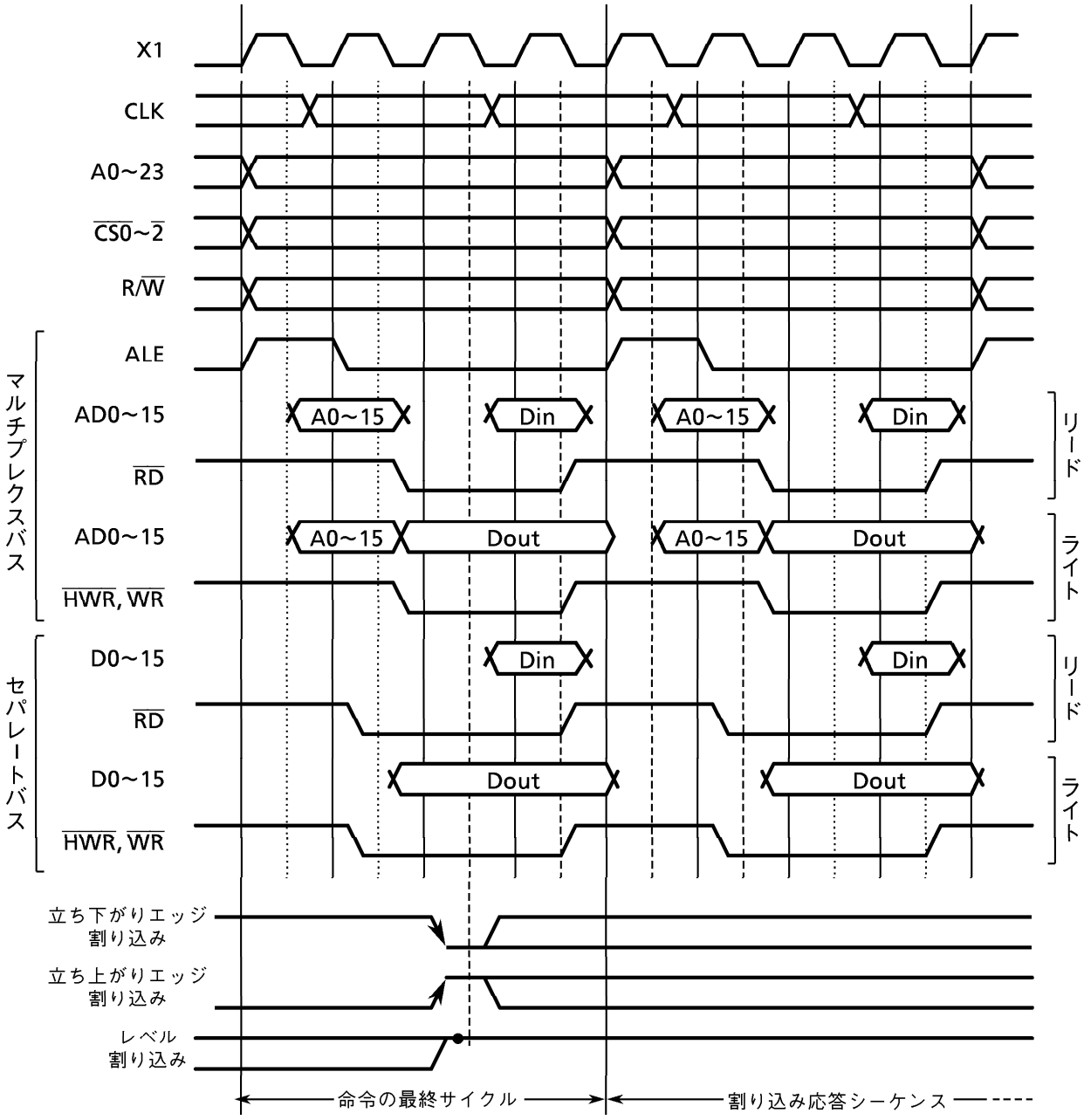


図7.6 1ステートのダミーサイクル



(注) このタイミング図は、典型例です。実際は、CPU内部のバスインタフェースユニットの働きにより、外部バスタイミングと内部割り込み受け付けタイミングは、一対一に対応しません。

図7.7 割り込み受け付けのタイミング

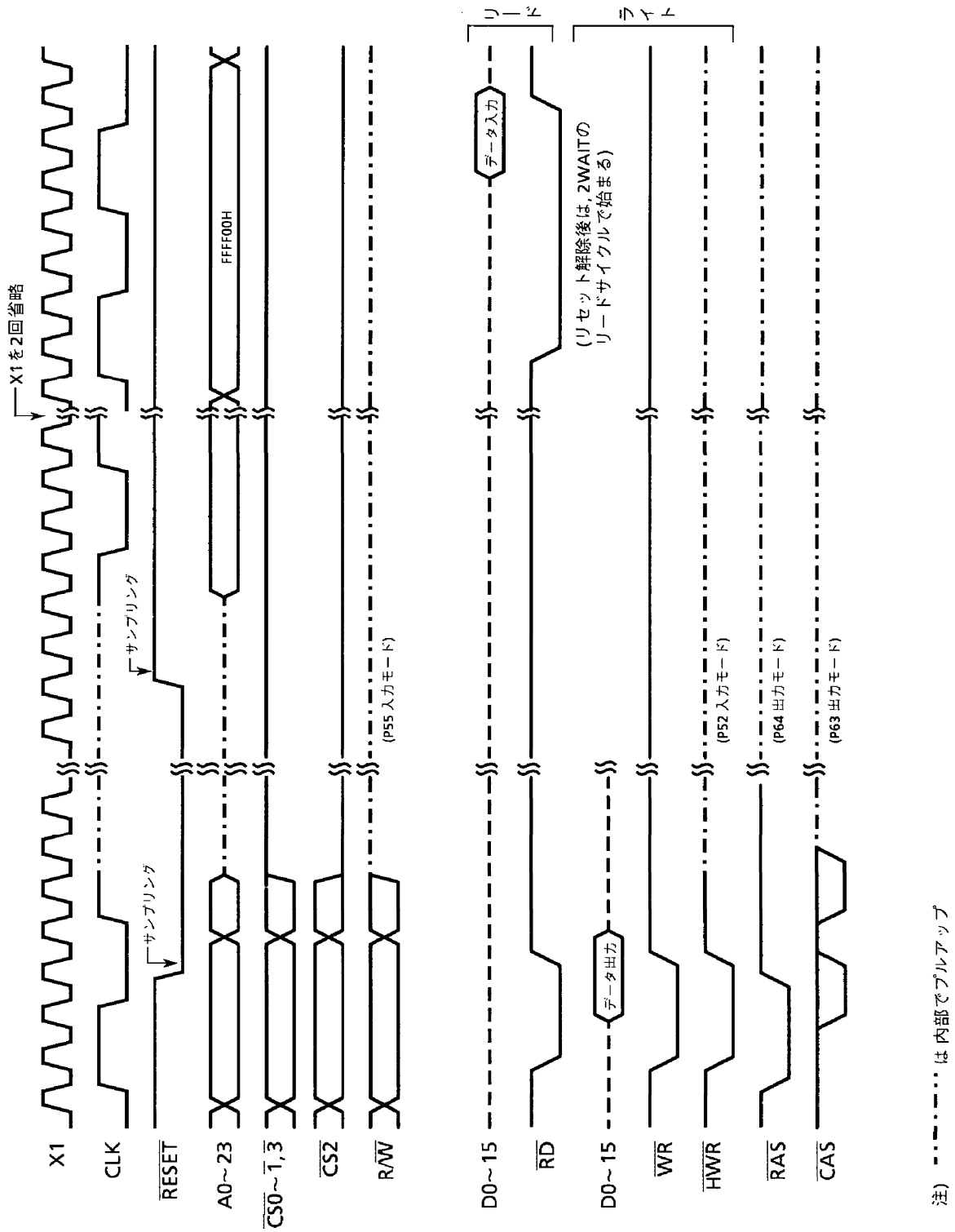


図7.8 リセットのタイミング (外部ROM動作 : TMP95C061)