T-49-19-05

MN1890 Series

8ビット・1チップ・デュアルマイクロコンピュータ
8-Bit Single-Chip Dual Microcomputers

■ 概 要

MN1890シリーズは，各種制御システムに最適な8ビット・デュアル1チップ・マイクロコンピュータです。仮想的に2個のプロセッサが1チップ上に集積された構造で，おのおのが異なった仕事を時分割で交互に処理できるため，ますます高速・複雑・多様化するシステム制御に対応できます。

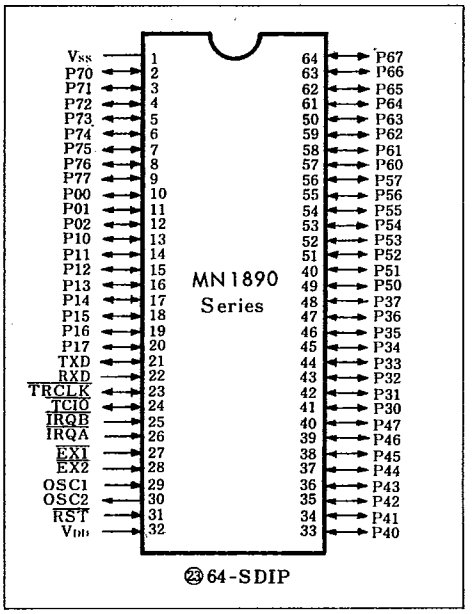
■ Description

The MN1890 series is a family of 8-bit single-chip dual microcomputers. Each microcomputer performs its program in a time-sharing manner, making the MN1890 series the best suited for complex processing of two real-time jobs.

■ 特 徴

- 時分割動作する2つのマイクロコンピュータを1チップに集積
- ROM 4Kバイト，RAM 256バイト内蔵(MN18942の場合)
- 高効率命令
  - 乗除算機能（乗算：乗数，被乗数8ビット　積16ビット／除算：被除数15ビット，除数7ビット，商8ビット，余り8ビット）
  - メモリマップドI/O
  - 8/4（10進加減算，ニブル交換）/1ビット操作
  - 命令リピート機能（ストリング操作，多方向分岐）
  - 多重ループ（スタック利用）
  - 相対分岐命令（+127～−128）
  - ROM領域内テーブルルックアップ機能
- 強力な割込み機能（外部×2，タイマ×2，シリアル）
- 16ビット・タイマ/カウンタ×2（下位8ビットはプリスケーラ）
- 8ビット・シリアルインタフェース（MN1500，MN1550，MN1890モード）

■ 端子配置図／Pin Assignment



㉓64-SDIP

- 高速な命令実行(最小500ns，平均1.5μs，8MHz発振時)
- 誤動作防止機能（暴走時トラップ割込み，低電圧検出）
- 外部ROM，RAM拡張可能（最大，各64Kバイト）
- 豊富な入出力端子(51本)(内D/A出力4本)
- クロック発振回路内蔵（セラミック発振子，Xtal用）
- 64ピン・プラスチックDILパッケージ（シュリンク）
- NMOS 5V単一電源

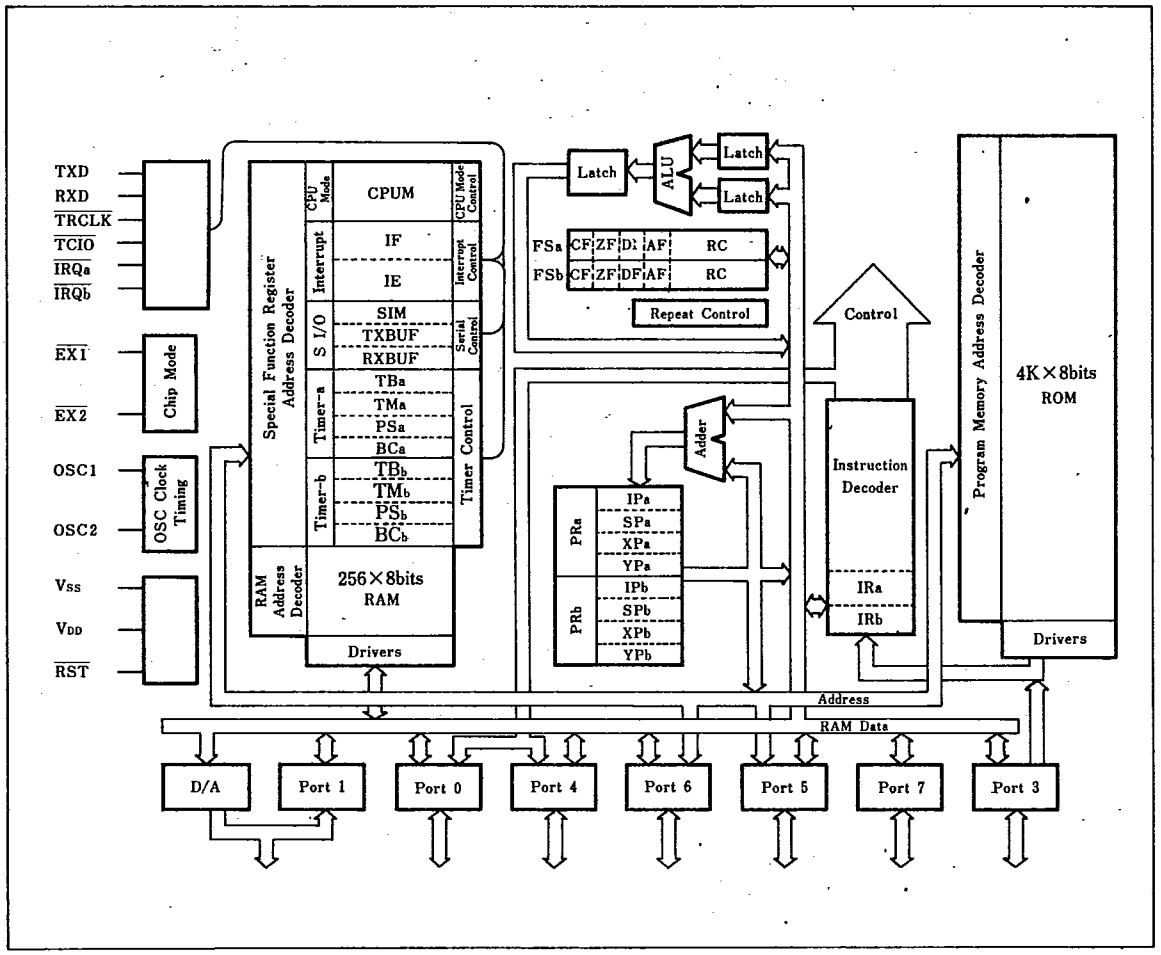
■ MN1890シリーズ製品系列／Types in MN1890 Series

形　名 Type No.	プロセス Process	ROM容量 (Byte)	RAM容量 (Byte)	パッケージ Package
MN18942	NMOS	4K	256	64-SDIP
MN18962		6K	256	64-SDIP
MN18982		8K	592	64-SDIP
MN18922		12K	656	64-SDIP

SDIP=Shrunk type Dual-In-Line Plastic Package

T-49-19-05

■ MN1890Series (MN18942) ブロック図/Block Diagram



■ デュアルマイクロコンピュータ

MN1890シリーズの最大の特徴であるデュアルマイクロコンピュータ方式について説明します。

1. 構　成

MN1890シリーズでは，図1 に示すように，2個の CPU (CPUa, CPUb)がバス(BUS)を介してプログラムメモリ(ROM)，データメモリ(RAM)および入出力回路(I/O) に接続されています。

MN1890シリーズでは ROM プログラミングの際に必要に応じて ROM，RAM，I/O を CPUa 側，CPUb 側のそれぞれに割り付け(一部は両 CPU で共用)，図2 に示すように，CPUa, ROMa, RAMa, I/Oa および CPUb, ROMb, RAMb, I/Obにより2 個のマイクロコンピュータ（マイクロコンピュータ a，マイクロコンピュータ b）を構成します。

CPUa, CPUbのそれぞれのプログラムの実行は，図3に示すように交互に実行されるため，MN1890シリーズマイクロコンピュータは見かけ上，2個のプログラムを並行に実行することができます。

CPUa，CPUb のプログラムの実行には，図4に示すように，各 CPU を命令ごとに切り換えるモード(Auto-swap)と，命令により一方の CPU のプログラムのみを連続的に実行し，他方を停止させるモード(Program swap) の2種の方法があり，必要に応じてプログラムにより切換えが可能です。

T-49-19-05

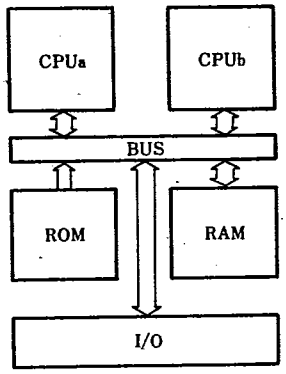


図 1 MN1890シリーズの基本構成

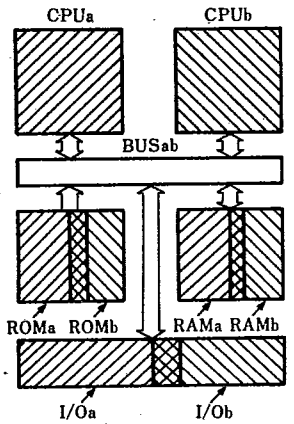


図 2 デュアルマイクロコンピュータの割付け

CPUaのプログラム

CPUa インストラクション 1
CPUa インストラクション 2
CPUa インストラクション 3
CPUa インストラクション 4
CPUa インストラクション 5
CPUa インストラクション 6
CPUa インストラクション 7
CPUa インストラクション 8
CPUa インストラクション 9
CPUa インストラクション 10
CPUa インストラクション 11
CPUa インストラクション 12
CPUa インストラクション 13
CPUa インストラクション 14
CPUa インストラクション 15
CPUa インストラクション 16

CPUbのプログラム

CPUb インストラクション 1
CPUb インストラクション 2
CPUb インストラクション 3
CPUb インストラクション 4
CPUb インストラクション 5
CPUb インストラクション 6
CPUb インストラクション 7
CPUb インストラクション 8
CPUb インストラクション 9
CPUb インストラクション 10
CPUb インストラクション 11
CPUb インストラクション 12
CPUb インストラクション 13
CPUb インストラクション 14
CPUb インストラクション 15
CPUb インストラクション 16

MN1890 シリーズのプログラムの実行

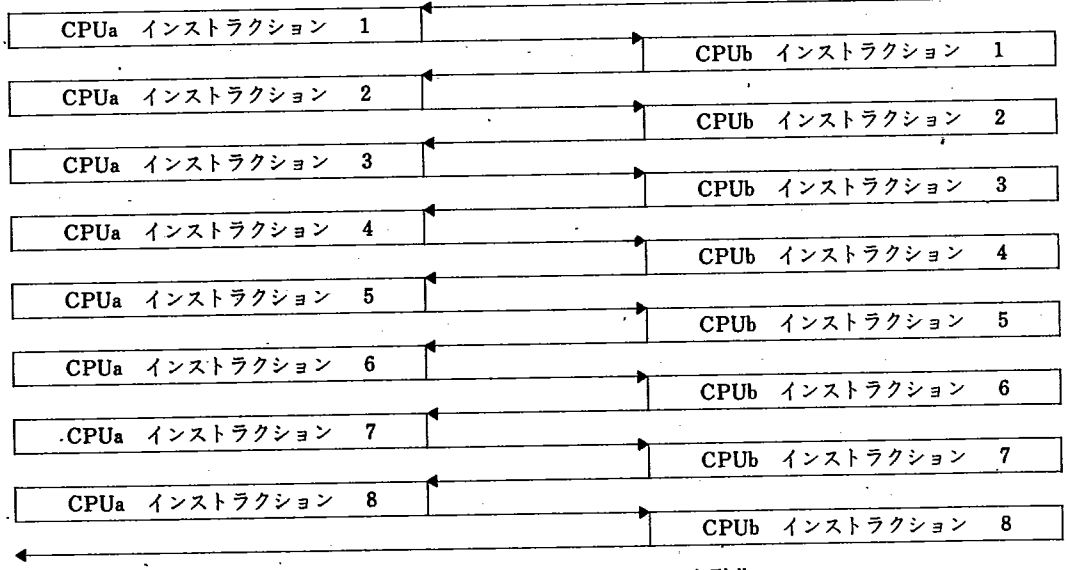
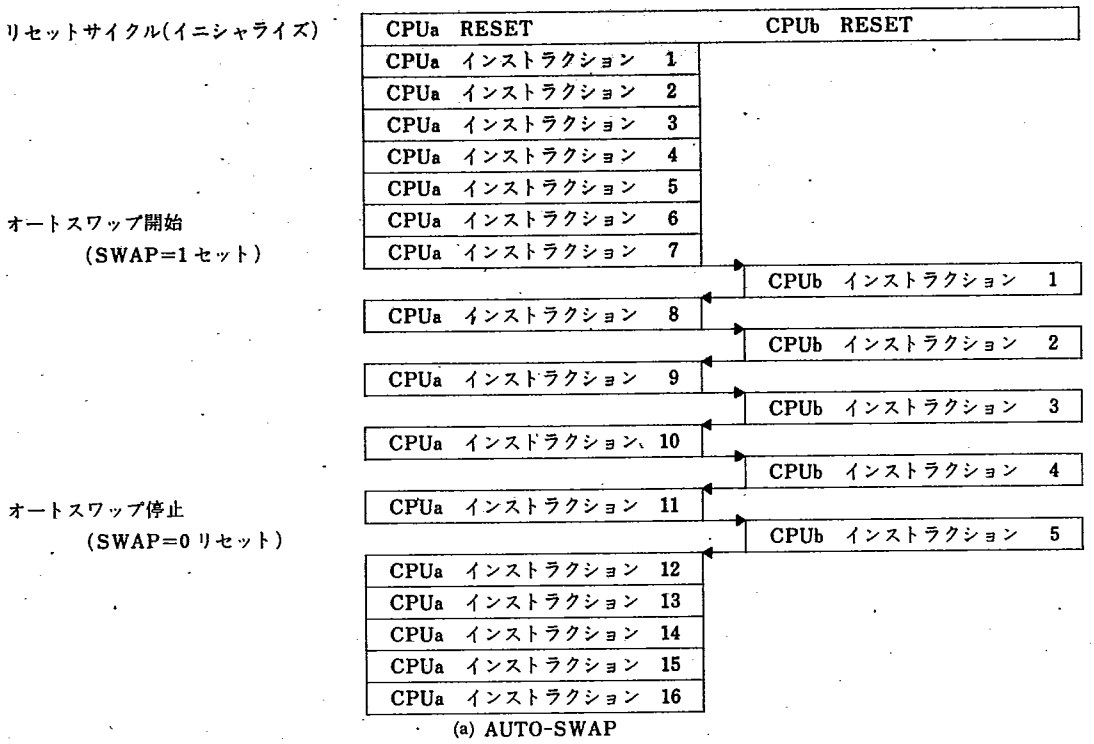
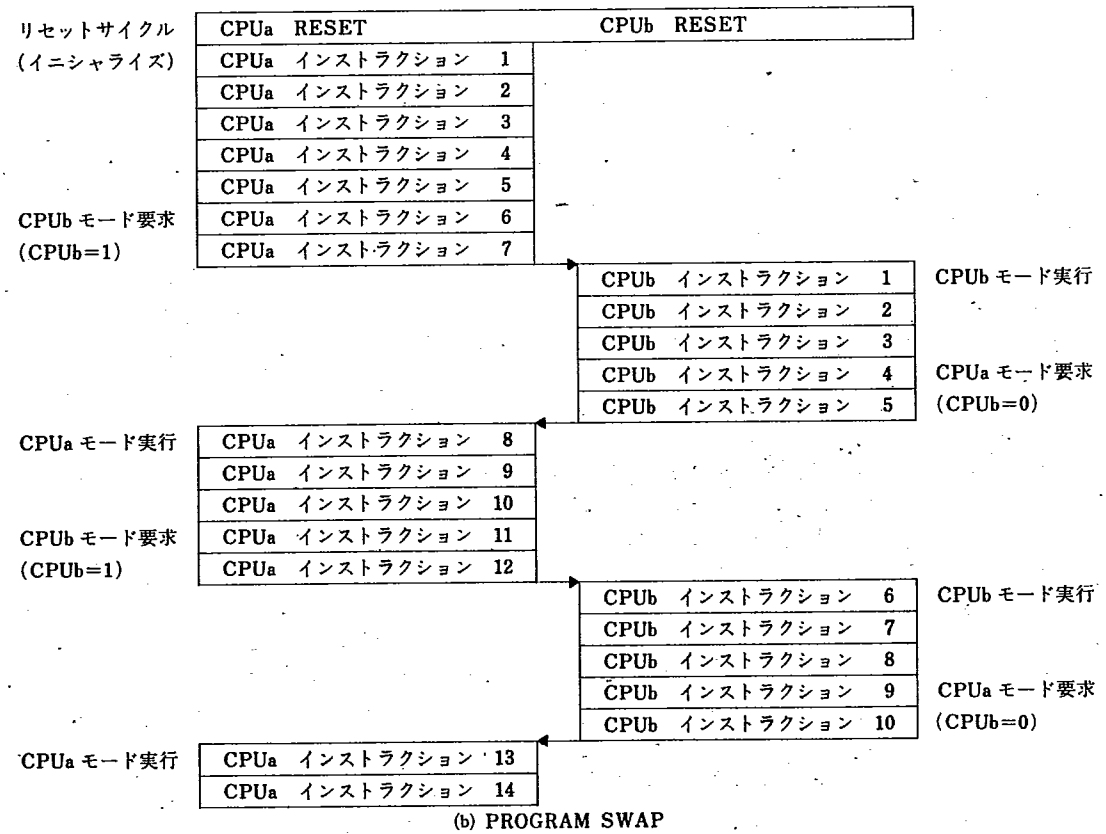


図 3 デュアルマイクロコンピュータ動作



(a) AUTO-SWAP



(b) PROGRAM SWAP

図4 プログラム実行モード

T-49-19-05

2. デュアルマイクロコンピュータの特徴

MN1890シリーズでは，デュアルマイクロコンピュータ方式を採用しており，下記の特徴があります。

(1) プログラムの並行処理に最適

一般のマイクロコンピュータで2個のプログラム処理を並行して処理するには，第一に割込みによる方法，第二に2個のプログラム処理を一定の時間ごとに切換えながら時分割実行し，実効的に並行処理を行なうのが一般的です。
ところが，このような方法では割込み時のレジスタ，フラグ退避などのプログラム処理の中断，再開の制御，あるいはプログラム切換えのための時間の管理などの余分な処理が必要となり，その結果高速なプログラム処理ができないという問題があります。
MN1890シリーズでは，このような問題点を解決するため，デュアルマイクロコンピュータ方式を採用しており，2個のプログラムの並行処理が可能です。

(2) ROM，RAM，I/Oの配分の自由度が大きい

一般に2個のマイクロコンピュータで応用システムの制御を行なう場合，ROM，RAM，I/Oの容量や本数は既製のマイクロコンピュータを使用するため自由に選択することはできませんので，一方のマイクロコンピュータには余裕があっても，他方のマイクロコンピュータは無理が生じる場合があります。
ところが，MN1890シリーズでは，図2に示したように，1チップで2個のマイクロコンピュータを構成できるわけですが，そのとき，マイクロコンピュータa,マイクロコンピュータbに配分されるROM，RAMの容量，I/Oの本数は，応用プログラム(ROMプログラム)開発時に応用システムに最適なように任意に決定できます。注1)

(3) 両マイクロコンピュータ間の通信が簡単

一般に2個のマイクロコンピュータを用いて応用システムの制御を行なう場合，その2つのマイクロコンピュータ間のデータの転送が不可欠であり，その通信のために送信側のマイクロコンピュータで送信しやすい形にデータをまとめて送信する必要があり，また，受信側のマイクロコンピュータではそのデータを解読する必要があります。そのため，双方のマイクロコンピュータで相当のプログラムステップが必要となり，プログラムの実行時間が伸び結果的に高速処理が不可能になるほか，上記通信のためのプログラム開発もプログラム開発者にとって相当の負担になります。
一方，MN1890シリーズでは，通信のためにRAM領域を共有でき，両マイクロコンピュータから自由にアクセスできるため，両マイクロコンピュータ間のデータ転送はきわめて簡単に行なうことができ，ほとんど特別な配慮は必要としません。

(4) その他

◎ サブルーチン，数字テーブルなどの共用化，両マイクロコンピュータにより共通のサブルーチン，数字テーブルなどが共用できます。

◎ 並行処理プログラムの開発が簡単
基本的には図3に示したように，a,b 2本のプログラムを独立に開発し，MN1890シリーズに搭載することにより並列処理が可能です。

以上，MN1890シリーズのデュアルマイクロコンピュータ方式を採用したことによる特徴を説明してきましたが，MN1890シリーズにはこれらの基本的な特徴のほかに，割込み，タイマなどの強力な付属機能，高機能命令，高速性などの数々の特徴があり，複雑な応用システムの高速処理が可能になっています。

注1) もちろん，両マイクロコンピュータに配分されるROM，RAM，I/Oの総数は内蔵の容量の範囲でなければなりません。たとえば，MN18942の場合，ROMの合計4Kバイト，RAMの合計256バイト，I/Oの合計51本(I/Oの本数は全シリーズ一定)。

■ 機能の概略

MN1890シリーズは，LSI上のハードウェア資源の多く(ROM, RAM, I/O, ALU, 命令デコーダなど)を時分割的に共用することにより，二つのマイクロコンピュータを単一LSI上に実現した1チップ時分割デュアルマイクロコンピュータです。
リセット時に初期起動される側のマイクロコンピュータをCPUaと呼び，他方をCPUbと呼ぶことにします。CPUaとCPUbはどちらか一方が実行状態となり，他方は待機状態となります。実行状態と待機状態の切換えには，1つの命令実行ごとにCPUa, CPUbが交互に実行されるモードと，プログラムにより実行CPUを指定し切換えるモードとがあります。
CPUaとCPUbの実行切換えに必要な時間はゼロであり，ロス時間は発生しません。そこで，CPUa, CPUbをいずれかのモードで切換えて実行させることにより，実効的にCPUa, CPUbの2つのマイクロコンピュータの並列実行を実現しています。
MN1890シリーズのブロック図の中で添字のa, bの付与されたブロックは，2個のマイクロコンピュータa, bにそれぞれ別々に用意されているハードウェアで，基本的にはそれぞれのマイクロコンピュータの実行状態を示すプログラムステータスに相当します。
MN1890シリーズの各ブロックをレジスタ構成に着目して示したのが図5です。図5から明らかなようにRAM, ROM, I/Oは2個のマイクロコンピュータに共有されており，こ

れらを2個のマイクロコンピュータに固有なプログラムステータスとインストラクションレジスタ(IR)および共有のALUなどからなる2個のCPU(CPUa, CPUb)で制御します。なお，2個のCPUの切換えはCPUMによって制御します。

割込み機能に関しては共有の制御回路により，それぞれのマイクロコンピュータに外部割込みが可能です。

タイマ/カウンタ機能は，2個のマイクロコンピュータにそれぞれ1個専用のハードウェアが用意されています。また，シリアルインタフェースはマイクロコンピュータa側にのみ準備されています。

このようにMN1890シリーズでは，最小限のハードウェアの増加で高機能の2個のマイクロコンピュータを1チップ上に実現できるよう設計されています。

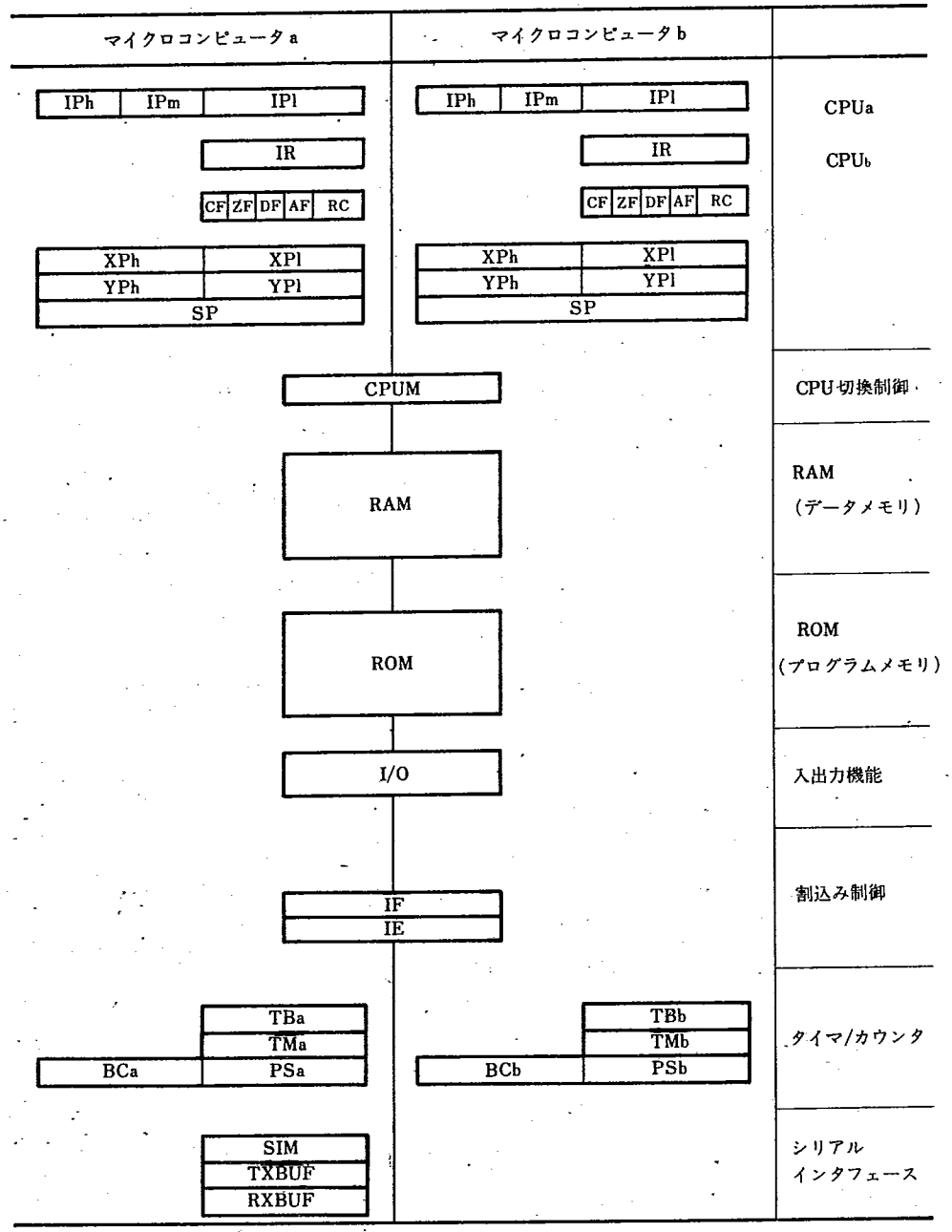


図5 MN1890シリーズのレジスタ構成

■ ブロックの機能

ブロック		機能
命令実行制御部	IRa IRb	インストラクションレジスタ(IRa, IRb)には，CPUがこれから実行しようとする命令がROMより読み出され，ラッチされます。 IRaはCPUa, IRbはCPUbに用いられます。
	Instruction Decoder	命令解読回路(Instruction Decoder)は，命令実行状態にあるCPUのインストラクションレジスタ(IRaまたはIRb)の内容を解読し，その命令の実行に必要な制御信号を順次発生し，チップ内の各ブロックを制御することにより命令を実行させます。
	ROM	読出し専用メモリ（ROM）はプログラムメモリであり，実行させようとするプログラムが格納されます。MN1890シリーズでは，内蔵ROM容量は4Kバイト，6Kバイト，8Kバイト，12Kバイトの4種類用意されています。
ポインタレジスタ部	IPa IPb	インストラクションポインタ（IPa, IPb）は，プログラムメモリ内の命令群の実行順序を制御する16ビットレジスタです。加算回路（Adder）と組み合わせることにより，いわゆるプログラムカウンタに相当する機能を行ないます。 IPaはCPUa, IPbはCPUbで使用します。
	SPa SPb	スタックポインタ（SP）は，データRAMの一部を使用するスタック領域のアドレスを示す16ビットのレジスタです。スタック領域はサブルーチンコール時，割込み時のIPなどの待避のほか，多重ループを構成する場合にも使用できます。SPaはCPUa, SPbはCPUbで使用します。
	XPa XPb	XPは，RAM空間をレジスタ間接アドレスするための16ビットのレジスタです。シングルオペランド命令もしくはダブルオペランド命令で使用されます。XPaはCPUa, XPbはCPUbで使用します。
	YPa YPb	YPは，RAM空間をレジスタ間接アドレスするための16ビットのレジスタです。ダブルオペランド命令のソースオペランド用として使用されます。YPaはCPUa, YPbはCPUbで使用します。
	Adder	加算器（Adder）はポインタレジスタのインクリメント，デクリメントおよび相対アドレスブランチ時のアドレス演算（IPの値の計算）に用います。
演算部	ALU	算術論理演算ユニット（ALU）はデータバスから2個の8ビットラッチ（Latch）に入力されたデータに対して，算術演算（加減算，乗除算，10進加減算，インクリメント，デクリメント，比較）および論理演算（AND, OR, XOR, 補数，ローテート，ニブルスワップ）を行ない，出力ラッチ（Latch）を介してデータバスに出力します。
フラグ部	FSa FSb (CF ZF DF AF RC)	フラグステータス（FS）は，CPUの実行状態を示す5種類のフラグより成ります。FSaはCPUa, FSbはCPUbで使用します。 ●キャリフラグ（CF）はALU演算結果がオーバフロー，またはアンダフローした場合にセットされ，それ以外のときはリセットされます。 ●ゼロフラグ（ZF）は，ALUの演算結果がゼロのときセットされ，それ以外のときリセットされます。 ●ダイレクトフラグ（DF）は，ダイレクトアドレスを下記のように制御します。 DF=0：アドレス上位8ビットを0として処理します。 DF=1：XP，YPのアドレス上位8ビットをRAMアドレスの上位8ビットとして処理します。 ●オートリピートフラグ（AF）は，命令のリピート（繰返し）処理用のフラグで，プログラムにより操作することは禁止されています。 ●リピート制御レジスタ（RC）は，命令のリピート実行制御を行うための4ビットレジスタです。RCには最大15までの値がセットでき，命令1回実行ごとにデクリメントされるので最大16回の命令の繰返しが可能です。
	Repeat Control	AF, RCの内容を判定し，命令の繰返し実行制御を行ないます。

T-49-19-05

■ ブロックの機能（つづき）

ブロック		機能
データメモリ部	RAM	ランダムアクセスメモリ（RAM）は，スタック領域およびプログラムの実行時に必要なデータを蓄積するデータ領域として使用します。MN1890シリーズでは，内蔵RAM容量は256バイト，592バイト，656バイトの3種類用意されています。
特殊レジスタ部	CPUM 〜 BCb	特殊レジスタ（CPUM〜BCb）は，RAMアドレス空間に割付けられており，RAMと同様にRead/Writeできます。 詳細は以下に説明します。
スワップ制御部	CPUM	CPUモードレジスタ（CPUM）は，CPUの命令実行モードを制御します。 命令実行モードには，①オートスワップモード ②プログラムスワップモードがあります。
割込み制御部	IF IE	割込要求フラグレジスタ（IF）と割込み許可フラグレジスタ（IE）のデータにより割込みを制御します。
シリアルインタフェース部	S I/O (SIM TXBUF RXBUF)	シリアルインタフェース（S I/O）はシリアルデータの送受を行ないます。 シリアルインタフェースレジスタ（SIM）はシリアル転送モードを制御します。 送信用バッファ（TXBUF）は送信データを出力します。 受信用バッファ（RXBUF）へは受信データを入力します。
タイマカウンタ部	Timer a (TBa TMa PSa BCa) Timer b (TBb TMb PSb BCb)	タイマインタフェース（Timer）は，タイマまたはイベント・カウンタとして使用します。 ●タイマモードレジスタ（TM）は，タイマインタフェースの動作モードを制御します。 ●プリスケーラ（PS）は，タイマインタフェースへの入力クロックを÷1, ÷16, ÷256のいずれかの割合で分周します。 ●タイマバッファ（TB）は，タイマの分周比を設定するレジスタです。 ●バイナリカウンタ（BC）は，プリスケーラの出力をカウントします。 タイマモードでは，BCがオーバフローするとTBの値がBCにプリセットされるため一定周期の時間が得られます。イベントカウンタモードでは，BCの値がイベントの数を示します。
入出力部	Port 0 Port 3〜7	パラレル入出力ポートです。ただし，チップ拡張時には，ROM／RAMアドレス，各種制御信号などの入出力端子となります。Port 0は3ビット構成，その他は8ビット構成です。
	D/A	4チャンネルの8ビットPWM方式DAコンバータ
	Port 1	4端子（P10〜P13）には，4チャンネルのD/A変換結果が出力され，他の4端子（P14〜P17）は4ビットの入出力ポートになります。
	Port 2 (TXD RXD $\overline{\text{TRCLK}}$ $\overline{\text{TCIO}}$ $\overline{\text{IRQa}}$ $\overline{\text{IRQb}}$)	下記の特殊な信号が入出力されます。 シリアルインタフェース送信データ（TXD） シリアルインタフェース受信データ（RXD） シリアルインタフェース送受信クロック（$\overline{\text{TRCLK}}$） タイマカウンタ入出力（$\overline{\text{TCIO}}$） CPUa外部割込み入力（$\overline{\text{IRQa}}$），CPUb外部割込み入力（$\overline{\text{IRQb}}$）
	Chip Mode ($\overline{\text{EX1}}$ $\overline{\text{EX2}}$)	チップモード（Chip Mode）は，$\overline{\text{EX1}}$, $\overline{\text{EX2}}$の状態により，以下のチップの動作モードを制御します。

動作モード	$\overline{\text{EX1}}$	$\overline{\text{EX2}}$
シングルチップモード	開放	開放
ROM拡張モード	開放	L
RAM拡張モード	L	開放
ROM, RAM拡張モード	L	L

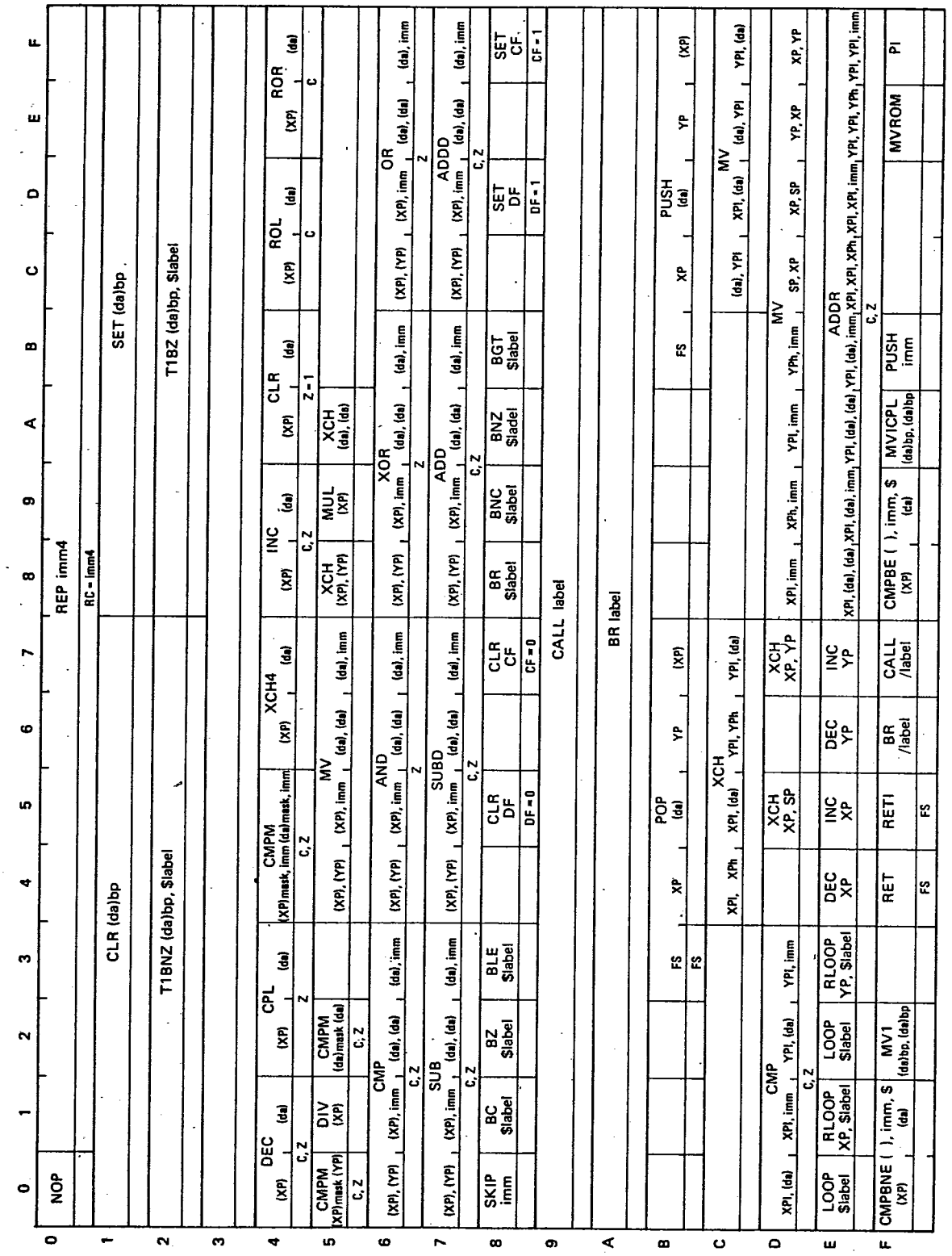
OSC CLOCK Timimg	水晶またはセラミック発振子(標準8MHz)を接続することにより，必要な内部タイミングを発生します。
Vss VDD $\overline{\text{RST}}$	Vss, VDDは電源端子でVDDには+5Vを印加します。(Vssは接地電位) $\overline{\text{RST}}$はリセット端子であり，$\overline{\text{RST}}$端子が'H'レベルでMN1890シリーズマイクロコンピュータは動作状態になります。

■ 端子説明

端子名	機能
V_{SS} (GND) V_{DD} (+5V)	電源供給端子であり，この間の交流インピーダンスを十分に低くするため，バイパスコンデンサを必要とし，配線も十分太くする必要があります。
OSC1 OSC2	セラミック発振子，または水晶発振子を接続するクロック発振端子です。 コンデンサを両端子と V_{SS} 間に入れるとき，配線が短くなるような設計上の配慮が必要です。 クロックを外部から入力する場合は OSC1 に入力してください。
$\overline{EX1}$ $\overline{EX2}$	MN1890 シリーズの動作モードを決めるための入力端子であり，シングルチップ使用時は解放しておきます。$\overline{EX1}$ 端子には，内部クロック S2 に同期したパルスを出力し，$\overline{EX2}$ 端子には，S0 に同期したパルスを出力します。クロック発振部が正常に動作しているか否かをユーザー応用システムに実装後検査するためにも利用できます。
$\overline{RST}$	電源投入起動時のリセットを行なうための端子です。クロック発振が安定発振状態に到達した後，少なくとも 2 マシンサイクル間はローレベルに保持しておく必要があります。$\overline{RST}$ 端子は内部にプルアップ抵抗を内蔵しています。また，供給電源 V_{DD} が低下した場合に，自動的にリセットを行なう回路を動作させることができます。$\overline{RST}$ と V_{SS} 間にコンデンサを入れる場合には，放電用ダイオードを $\overline{RST}$ と V_{DD} 間に入れることを推奨します。
TXD	シリアルインタフェース送信データ用端子であり，内部にプルアップ抵抗を内蔵しています。 複数の MN1890 シリーズを並列に接続した場合，同時に複数の MN1890 から送信が行なわれた場合の衝突検出を行なうために TXD 端子値と，送信しようとする値の比較が行なわれます。
RXD	シリアルインタフェース受信データ用端子であり，入力専用端子です。これら RXD と TXD は接続して使用すると，MN1500 シリーズの SBD 端子と接続できます。RXD はプルアップ抵抗を内蔵しています。
$\overline{TRCLK}$	シリアルインタフェース送受信クロック用入出力端子です。 内部クロックモード時は出力端子となり，外部クロックモード時は入力端子となります。
$\overline{TCIO}$	タイマインタフェース入出力用端子です。タイマモード時は出力端子となり，イベントカウント時は入力端子となります。タイマモード時は 2 本のタイマのうち，どちらか一方だけがこの端子を利用できます。
$\overline{IRQA}$ $\overline{IRQB}$	CPUa 用と，CPUb 用の 2 本の外部割込み用端子です。 この端子はプルアップ抵抗を内蔵しています。この端子を 2 マシンサイクル以上，ローレベルにすることにより割込みが発生します。2 マシンサイクルよりも短いパルスは正常な割込み信号として処理されません。
P00～02 P14～17 P30～77	双方向の入出力ポートであり，プルアップ抵抗を内蔵しています。 このポートを入力ポートとして使用する場合，ハイレベルをポートに出力しておきます。これらのポートは，8 ビット，4 ビット，1 ビットのデータを入出力することができます。また，8 ビットを単位とする同一アドレスをもつポート内で入力ビットと出力ビットを混在させることができます。ポートは全体で 47 本あります。
P10～13	PWM を用いた D/A 変換出力端子です。

T-49-19-05

■ 命令マップ／Instruction Map



T-49-19-05

■ MN1890 Series 命令セット／Instruction Set

Group	Mnemonic	
Data Transfer Group	MV	(DOUBLE OPERAND) XP, YP YP, XP XP, SP SP, XP XPI, imm XPh imm YPI, imm YPh, imm XPI, (da) YPI, (da) (da), XPI (da), YPI
	MVROM	(XP), (YP)
	MV1 MV1CPL	(dad)bpd, (das)bps (dad)bpd, (das)bps
	XCH	XP, YP XP, SP XPI, XPh YPI, YPh XPI, (da) YPI, (da) (XP), (YP) (da), (da)
	XCH4	(SINGLE OPERAND)
	CLR	(SINGLE OPERAND) (da)bp DF CF
	SET	(da)bp DF CF
Logical Group	CPL ROL ROR AND OR XOR	(SINGLE OPERAND) (SINGLE OPERAND) (SINGLE OPERAND) (DOUBLE OPERAND) (DOUBLE OPERAND) (DOUBLE OPERAND)

Group	Mnemonic	
Arithmetic Group	ADD ADDD ADDR	(DOUBLE OPERAND) (DOUBLE OPERAND) XPI, XPI, XPh YPI, YPI, YPh XPI, YPI, imm YPI, YPI, imm XPI, (da1), imm YPI, (ad1), imm XPI, (da1), (da2) YPI, (da1), (da2)
	INC	(SINGLE OPERAND) XP YP
	DEC	(SINGLE OPERAND) XP YP
	SUB SUBD	(DOUBLE OPRAND) (DOUBLE OPERAND)
	MUL DIV	(XP) (XP)
	CMP	(DOUBLE OPERAND) XPI, imm YPI, imm XPI, (da) YPI, (da)
	CMPM	(XP)mask, imm (da)mask, imm (XP)mask, (YP)mask (da1)mask, (da2)mask
Stack, Hardware Group	PUSH	(SINGLE OPERAND) XP YP imm FS
	POP	(SINGLE OPERAND) XP YP FS
	REP PI NOP	imm4

Group	Mnemonic	
Branch Group	BR	$label label /label
	BC BZ BLE BNC BNZ BGT SKIP T1BNZ T1BZ	$label $label $label $label $label $label imm (da)bp, $label (da)bp, $label
	LOOP	$label
	RLOOP	XP, $label YP, $label
	CMPBNE	(XP), imm, $label (da), imm, $label
	CMPBE	(XP), imm, $label (da), imm, $label
	CALL	label /label
	RET RETI	

**** (SINGLE OPERAND) ****
(XP)
(da)

**** (DOUBLE OPERAND) ****
(XP), (YP)
(YP), imm
(dad), (das)
(da), imm

$ label = relative address (+127 −128)
label = absolute address 4K-byte/page
/ label = absolute address 64K-byte

■ MN1890 Series 詳細説明／Detail

Group	Mnemonic		Operation	CZ	In	machine code (hexa.)	IUO
Data Transfer Group	MV	XP, YP	XP ← YP move			DF	*1H
		YP, XP	YP ← XP			DE	*1H
		XP, SP	XP ← SP			DD	*1H
		SP, XP	SP ← XP			DC	*1H
		XPI, imm	XPI ← imm			D8 : imm	110
		XPh, imm	XPh ← imm			D9 : imm	110
		YPI, imm	YPI ← imm			DA : imm	110
		YPh, imm	YPh ← imm			DB : imm	110
		XPI, (da)	XPI ← RAM(XPh ∧ DF, da)		T	CD : da	120
		YPI, (da)	YPI ← RAM(YPh ∧ DF, da)		T	CF : da	120
		(da), XPI	RAM(XPh ∧ DF, da) ← XPI			CC : da	110 11H
		(da), YPI	RAM(YPh ∧ DF, da) ← YPI			CE : da	110 11H
		(XP), (YP)	RAM(XP) ← RAM(YP)		T	54	*21
		(XP), imm	RAM(XP) ← imm			55 : imm	12H
		(dad), (das)	RAM(XPh ∧ DF, dad) ← RAM(YPh ∧ DF, das)		T	56 : das : dad	120 12H
		(da), imm	RAM(XPh ∧ DF, da) ← imm			57 : imm : da	120 12H
	MVROM	(XP), (YP)	RAM(XP) ← ROM(YP) ROM table look-up			FE	*40 *41
	MV1	(dad)bpd, (das)bps	RAM(XPh ∧ DF, dad)bpd ← RAM(YPh ∧ das)bps move bit		LT	F2 : das : bpd4, bps4 : dad	14H
	MV1CPL	(dad)bpd, (das)bps	RAM(XPh ∧ DF, dad)bpd ← .cpl. RAM(Yph ∧ DF, das)bps move bit complement		LT	FA : das : bpd4, bps4 : dad	14H

T-49-19-05

Group	Mnemonic	Operation	CZ	in	machine code (hexa.)	IUO
	XCH XP, YP XP, SP XPl, XPh YPl, YPh XPl, (da) YPl, (da) (XP), (YP) (da1), (da2)	XP ↔ YP exchange XP ↔ SP XPl ↔ XPh YPl ↔ YPh XPl ↔ RAM(XPh ∧ DF, da) YPl ↔ RAM(YPh ∧ DF, da) RAM(XP) ↔ RAM(YP) RAM(XPh ∧ DF, da1) ↔ RAM(YPh ∧ DF, da2)		 L L LL LL	D7 D5 C4 C6 C5 : da C7 : da 58 5A : da2 : da1	120 120 *21 *21 120 120 *51 15H
	XCH4 (XP) (da)	RAM(XP)un ↔ RAM(XP)ln exchange nibble RAM(XPh ∧ DF, da)un ↔ RAM(XPh ∧ DF, da)ln		L L	46 47 : da	*21 12H
	CLR (XP) (da) (da)bp DF CF	RAM(XP) ← 0 clear RAM(XPh ∧ DF, da) ← 0 RAM(XPh ∧ DF, da)bp ← 0 clear bit DF ← 0 clear DF CF ← 0 clear CF	1 1 0	L L L 	4A 4B : da 10+bp : da(bp = 0 ~ 7) 85 87	*21 12H 12H *21 *21
	SET (da)bp DF CF	RAM(XPh ∧ DF, da)bp ← 1 set bit DF ← 1 set DF CF ← 1 set CF	 1	L 	18+bp : da(bp = 0 ~ 7) 8D 8F	12H *21 *21
Logical Group	CPL (XP) (da)	RAM(XP) ← RAM(XP) . cpl. complement RAM(XPh ∧ DF, da) ← RAM(XPh ∧ DF, da) . cpl.	Z Z	L L	42 43 : da	*21 12H
	ROL (XP) (da)	CF ← RAM( )7 : RAM( )bp ← RAM( )bp−1 : RAM( )0 ← CF [RAM( ) = RAM(XP) or RAM(XPh ∧ DF, da)] rotate left with CF	C C	L L	4C 4D : da	*21 12H
	ROR (XP) (da)	CF ← RAM( )0 : RAM( )bp−1 ← RAM( )bp : RAM( )7 ← CF [RAM( ) = RAM(XP) or RAM(XPh ∧ DF, da)] rotate right with CF	C C	L L	4E 4F : da	*21 12H
Logical Group	AND (XP), (YP) (XP), imm (dad), (das) (da), imm	RAM(XP) ← RAM(XP) . ∧ . RAM(YP) RAM(XP) ← RAM(XP) . ∧ . imm RAM(XPh ∧ DF, dad) ← RAM(XPh ∧ DF, dad) . ∧ . RAM(YPh ∧ DF, das) RAM(XPh ∧ DF, dad) ← RAM(XPh ∧ DF, dad) . ∧ . imm	Z Z Z Z	LT L LT L	64 65 : imm 66 : das : dad 67 : imm : da	*31 13H 13H 13H
	OR (XP), (YP) (XP), imm (dad), (das) (da), imm	RAM(XP) ← RAM(XP) . ∨ . RAM(YP) RAM(XP) ← RAM(XP) . ∨ . imm RAM(XPh ∧ DF, dad) ← RAM(XPh ∧ DF, dad) . ∨ . RAM(YPh ∧ DF, das) RAM(XPh ∧ DF, da) ← RAM(XPh ∧ DF, da) . ∨ . imm	Z Z Z Z	LT L LT L	6C 6D : imm 6E : das :dad 6F : imm : da	*31 13H 13H 13H
	XOR (XP), (YP) (XP), imm (dad), (das) (da), imm	RAM(XP) ← RAM(XP) . ∀ . RAM(YP) exclusive OR RAM(XP) ← RAM(XP) . ∀ . imm RAM(XPh ∧ DF, dad) ← RAM(XPh ∧ DF, dad) . ∀ . RAM(YPh ∧ DF, das) RAM(XPh ∧ DF, da) ← RAM(XPh ∧ DF, da) . ∀ . imm	Z Z Z Z	LT L LT L	68 69 : imm 6A : das : dad 6B : imm : da	*31 13H 13H 13H
Arithmetic Group	ADD (XP), (YP) (XP), imm (dad), (das) (da), imm	RAM(XP) ← RAM(XP) + RAM(YP) + CF add with CF RAM(XP) ← RAM(XP) + imm + CF RAM(XPh ∧ DF, dad) ← RAM(XPh ∧ DF, dad) + RAM(YPh ∧ DF, das) + CF RAM(XPh ∧ DF, da) ← RAM(XPh ∧ DF, da) + imm + CF	CZ CZ CZ CZ	LT L LT L	78 79 : imm 7A : das : dad 7B : imm : da	*31 13H 13H 13H
	ADDD (XP), (YP) (XP), imm4 (dad), (das) (da), imm4	RAM(XP)ln ← RAM(XP)ln + RAM(YP)ln + CF add decimal RAM(XP)ln ← RAM(XP)ln + imm4 + CF RAM(XPh ∧ DF, dad)ln ← RAM(XPh ∧ DF, dad)ln + RAM(YPh ∧ das)ln + CF RAM(XPh ∧ DF, da)ln ← RAM(XPh ∧ DF, da)ln + imm4 + CF	CZ CZ CZ CZ	LT L LT L	7C 7D : imm4 7E : das : dad 7F : imm4 : da	*41 14H 14H 14H
	ADDR XPl, XPl, XPh YPl, YPl, YPh XPl, XPl, imm YPl, YPl, imm XPl, (da), imm YPl, (da), imm XPl, (da1), (da2) YPl, (da1), (da2)	XPl ← XPl + XPh add and store register YPl ← YPl + YPh XPl ← XPl + imm YPl ← YPl + imm XPl ← RAM(XPh ∧ DF, da) + imm YPl ← RAM(YPh ∧ DF, da) + imm XPl ← RAM(XPh ∧ DF, da1) + RAM(YPh ∧ DF, da2) YPl ← RAM(XPh ∧ DF, da1) + RAM(YPh ∧ DF, da2)	CZ CZ CZ CZ CZ CZ CZ CZ	 T T TT TT	EC EE ED : imm EF : imm E9 : imm : da EB : imm : da E8 : da2 : da1 EA : da2 : da1	*31 *31 13H 13H 13H 13H 13H 13H
	INC XP YP (XP) (da)	XP ← XP + 1 increment YP ← YP + 1 RAM(XP) ← RAM(XP) + 1 RAM(XPh ∧ DF, da) ← RAM(XPh ∧ DF, da) + 1	 CZ CZ	 L L	E5 E7 48 49 : da	*1H *1H *21 12H
	DEC XP YP (XP) (da)	XP ← XP − 1 decrement YP ← YP − 1 RAM(XP) ← RAM(XP) −1 RAM(XPh ∧ DF, da) ← RAM(XPh ∧ DF, da) − 1	 CZ CZ	 L L	E4 E6 40 41 : da	*1H *1H *21 12H
	SUB (XP), (YP) (XP), imm (dad), (das) (da), imm	RAM(XP) ← RAM(XP) − RAM(YP) − CF subtract RAM(XP) ← RAM(XP) − imm − CF RAM(XPh ∧ DF, dad) ← RAM(XPh ∧ DF, dad) − RAM(YPh ∧ DF, das) − CF RAM(XPh ∧ DF, da) ← RAM(XPh ∧ DF, da) − imm − CF	CZ CZ CZ CZ	LT L LT L	70 71 : imm 72 : das : dad 73 : imm : da	*31 13H 13H 13H
	SUBD (XP), (YP) (XP), imm4 (dad), (das) (da), imm4	RAM(XP)ln ← RAM(XP)ln − RAM(YP)ln − CF sub decimal RAM(XP)ln ← RAM(XP)ln − imm4 − CF RAM(XPh ∧ DF, dad)ln ← RAM(XPh ∧ DF, dad)ln − RAM(YPh ∧ DF, das)ln − CF RAM(XPh ∧ DF, da)ln ← RAM(XPh ∧ DF, da)ln − imm4 − CF	CZ CZ CZ CZ	LT L LT L	74 75 : imm4 76 : das : dad 77 : imm4 : da	*41 14H 14H 14H
	MUL (XP) DIV (XP)	RAM(XP+1), RAM(XP) ← RAM(XP) × RAM(XP+1) RAM(XP+1), RAM(XP) ← RAM(XP+1), RAM(XP)/YPl		TT TT	59 51	*81 *81
Arithmetic Group	CMP XPl, imm YPl, imm XPl, (da) YPl, (da) (XP), (YP) (XP), imm (da1), (da2) (da), imm	XPl − imm compare YPl − imm XPl − RAM(XPh ∧ DF, da) YPl − RAM(YPh ∧ DF, da) RAM(XP) − RAM(YP) RAM(XP) − imm RAM(XPh ∧ DF, da1) − RAM(YPh ∧ DF, da2) RAM(XPh ∧ DF, da) − imm	CZ CZ CZ CZ CZ CZ CZ CZ	 T T TT T TT T	D1 : imm D3 : imm D0 : da D2 : da 60 61 : imm 62 : da2 : da1 63 : imm : da	*31 *31 13H 13H *31 13H 13H 13H
	CMPM (XP)mask, imm (da)mask, imm (XP)mask, (YP)mask (da1)mask, (da2)mask	[RAM(XP) . ∧ . mask] − imm compare with mask [RAM(XPh ∧ DF, da) . ∧ . mask] − imm [RAM(XP) . ∧ . mask] − [RAM(YP) . ∧ . mask] [RAM(XPh ∧ DF, da1) . ∧ . mask] − [RAM(YPh ∧ DF, da2) . ∧ . mask]	CZ CZ CZ CZ	T T TT TT	44 : mask : imm 45 : da : mask : imm 50 : mask 52 : da2 : mask : da1	*41 14H *61 16H

Group	Mnemonic		Operation		CZ	in	machine code (hexa.)	IUO
Branch Group	BR	$label label /label	IP ← IP+2 + $label IP ← IPh, label IP ← /label	branch relative (+127 –128) branch absolute (4K-byte/page) branch absolute long (64K-byte)			88 : $label A0+label-m : label-l F6 : label-h, m : label-l	130 120 120
	SKIP	imm	IP ← IP+2	skip next one byte			80 : imm	130
	BC	$label	if CF=1	then IP ← IP+2 + $label / else IP ← IP+2			81 : $label	130
	BZ	$label	if ZF=1	then IP ← IP+2 + $label / else IP ← IP+2			82 : $label	130
	BLE	$label	if CF=1. or. ZF=1	then IP ← IP+2 + $label / else IP ← IP+2			83 : $label	130
	BNC	$label	if CF=0	then IP ← IP+2 + $label / else IP ← IP+2			89 : $label	130
	BNZ	$label	if ZF=0	then IP ← IP+2 + $label / else IP ← IP+2			8A : $label	130
	BGT	$label	if CF=0. and. ZF=0	then IP ← IP+2 + $label / else IP ← IP+2			8B : $label	130
	T1BNZ	(da)bp, $label	if RAM(XPh ∧ DF, da)bp=1	then IP ← IP+3 + $label / else IP ← IP+3		T	20+bp : da : $label	130 13H
	T1BZ	(da)bp, $label	if RAM(XPh ∧ DF, da)bp=0	then IP ← IP+3 + $label / else IP ← IP+3		T	28+bp : da : $label	130 13H
	LOOP	$label	RAM(SP) ← RAM(SP)–1 if RAM(SP)>0 then IP ← IP+2 + $label / else IP ← IP+2 : SP ← SP+1			L	E0 : $label	130
	RLOOP	XP, $label YP, $label	RAM(SP) ← RAM(SP)–1 : XP ← XP+1 if RAM(SP)>0 then IP ← IP+2 + $label / else IP ← IP+2 : SP ← SP+1 RAM(SP) ← RAM(SP)–1 : YP ← YP+1 if RAM(SP)>0 then IP ← IP+2 + $label / else IP ← IP+2 : SP ← SP+1			L L	E1 : $label E3 : $label	130 130
	CMPBNE	(XP), imm, $label (da), imm, $label	if RAM(XP) ≠ imm if RAM(XPh ∧ DF, da) ≠ imm	then IP ← IP+3 + $label / else IP ← IP+3 then IP ← IP+4 + $label / else IP ← IP+4		T T	F0 : imm : $label F1 : imm : da : $label	140 14H 140 14H
	CMPBE	(XP), imm, $label (da), imm, $label	if RAM(XP)=imm if RAM(XPh ∧ DF, da)=imm	then IP ← IP+3 + $label / else IP ← IP+3 then IP ← IP+4 + $label / else IP ← IP+4		T T	F8 : imm : $label F9 : imm : da : $label	140 14H 140 14H
	CALL	label /label	call absolute (4K-byet) call absolute (64K-byte) hardware interrupt	IP ← IPh, label IP ← /label	RAM(SP–1) ← IPh, IPm RAM(SP–2) ← IPl SP ← SP–2		90+label-m : label-l F7 : label-h, m : label-l don't care :	*30 *30 *50 *60
	RET		IPl ← RAM(SP) : IPh, IPm ← RAM(SP+1) : SP ← SP+2			L	F4	*40
	RETI		repeat FS ← RAM(SP) IPl ← RAM(SP+1) IPh, IPm ← RAM(SP+2) IR ← RAM(SP+3) SP ← SP+4	normal FS ← RAM(SP) IPl ← RAM(SP+1) IPh, IPm ← RAM(SP+2) SP ← SP+3	CZ	L	F5 normal	*50 *51
Stack, Hardware Group	PUSH	XP YP (XP) (da) imm FS	RAM(SP–1) ← XPh : RAM(SP–2) ← XPl : SP ← SP–2 RAM(SP–1) ← YPh : RAM(SP–2) ← YPl : SP ← SP–2 RAM(SP–1) ← RAM(XP) : SP ← SP–1 RAM(SP–1) ← RAM(XPh ∧ DF, da) : SP ← SP–1 RAM(SP–1) ← imm : SP ← SP–1 RAM(SP–1) ← FS : SP ← SP–1	push stack		L L	BC BE BF BD : da FB : imm BB	*31 *31 *31 *31 12H *21
	POP	XP YP (XP) (da) FS	XPl ← RAM(SP) : XPh ← RAM(SP+1) : SP ← SP+2 YPl ← RAM(SP) : XPh ← RAM(SP+1) : SP ← SP+2 RAM(XP) ← RAM(SP) : SP ← SP+1 RAM(XPh ∧ DF, da) ← RAM(SP) : SP ← SP+1 FS ← RAM(SP) : SP ← SP+1	pop stack	CZ	L L L L L	B4 B6 B7 B5 : da B3	*31 *31 *21 *20 *21 *21
	PI		RAM(SP-1) ← IPh, IPm RAM(SP-2) ← IPl RAM(SP-3) ← FS	: IPh, IPm ← ROM(VECTOR) : IPl ← ROM(VECTOR+1) : SP ← SP-3	interrupt		FF	*21
	REP	imm4	RC ← repeat times		repeat control register RC 4-bit set		00 + 1 ~ F	*21
	NOP		IP ← IP+1		No operation		00	*21
			reset CPU IPa ← ROM(1), ROM(0) SPa ← X'130 IF, IE, CPUM, SIM, TMa, TMb ← 0	IPb ← ROM(11), ROM(10) SPb ← X'110 I/O-port ← X'FF				*10 0

input select
example
ADD (dad), (das) LT

L = port latch
T = terminal

input que
(* = don't care)

execution state

output que
(H = hold)

repeat mode que

XP, YP, SP, XPh, XPl, YPh, YPl, — Pointer Register
(XP), (YP), (SP), (da), (das), (dad), (da1), (da2) — RAM data ( )
imm, imm4, mask — immediate data imm4 = 4-bit data
bp, bps, bpd — bit position
da, das, dad, da1, da2 — direct address
DF — direct flag (reset = 0 . . . . . RAM address upper 8-bit = 0 at : da addressing mode)
CF — carry / borrow flag
ZF — zero flag
un, in — upper / lower nibble (4-bit)
IP, IPh, IPm, IPl — instruction pointer h(4-bit, m(4-bit), l(8-bit)
FS — flag status (CF, ZF, AF, DF, RC)
IR — instruction register

T-49-19-05

■ 絶対最大定格／Absolute Maximum Ratings (Ta=25°C)

Item	Symbol	Rating	Unit
電源電圧	V_{DD}	−0.3～+7.0	V
入力電圧	V_I	−0.3～+7.0	V
出力端子電圧	V_O	−0.3～+7.0	V
動作周囲温度	T_{opr}	−20～+70	°C
保存温度	T_{stg}	−55～+125	°C

■ DC特性／DC Characteristics (V_{DD}=4.5～5.5V, Ta=25°C)

Item	Symbol	Condition	min.	typ.	max.	Unit
電源電流	I_{DD}	V_{DD}=5V		180*		mA
消費電力	P_{tot}	外付負荷なし		900		mW
入力端子1 P21(RXD), P24($\overline{IRQB}$), P25($\overline{IRQA}$), $\overline{RST}$						
電圧ハイレベル	V_{IH1}		3.2		V_{DD}	V
電圧ローレベル	V_{IL1}		V_{SS}		0.8	V
入力電流	I_{I1}	V_{DD}=5V V_I=0.8V			−250	μA
入出力端子 P0～P1, T20(TXD), P22($\overline{TRCLK}$), P23($\overline{TCIO}$), P3～P7**						
出力電圧ハイレベル	V_{OH3}	I_{OH}=−100μA	3.0			V
出力電圧ローレベル	V_{OL3}	I_{OL}=1.2mA			0.5	V
入力電圧ハイレベル	V_{IH3}		2.4		V_{DD}	V
入力電圧ローレベル	V_{IL3}		V_{SS}		0.8	V
入力電流	I_{I3}	V_I=0.8V			−600	μA
端子容量						
入力端子	C_I	V_I=0.8V		5		pF
入出力端子	C_B	V_O=2V		15		pF
OSC端子	C_{OSC}	V_{OSC}=2V		10		pF

* MN18942の標準値です。品種によって標準値は異なります。

** P10～P13はオープンドレイン構造で出力電圧ローレベルの特性しかありません。