

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。

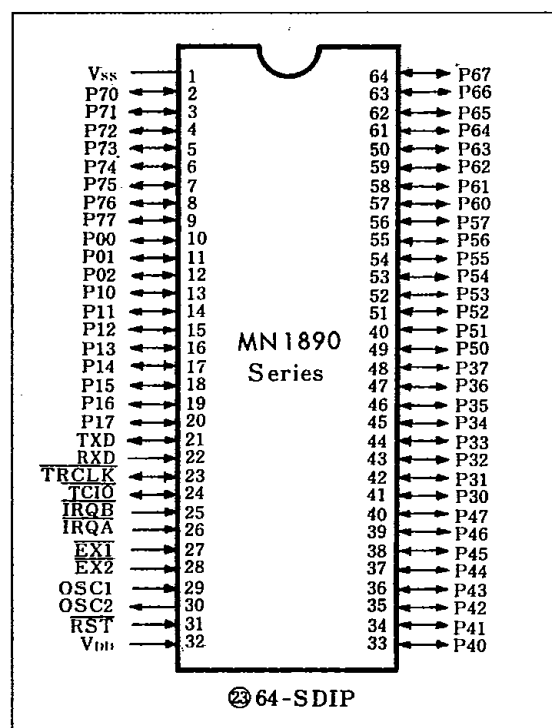
■ Description

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 ビットはプリスケアラ)
- 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



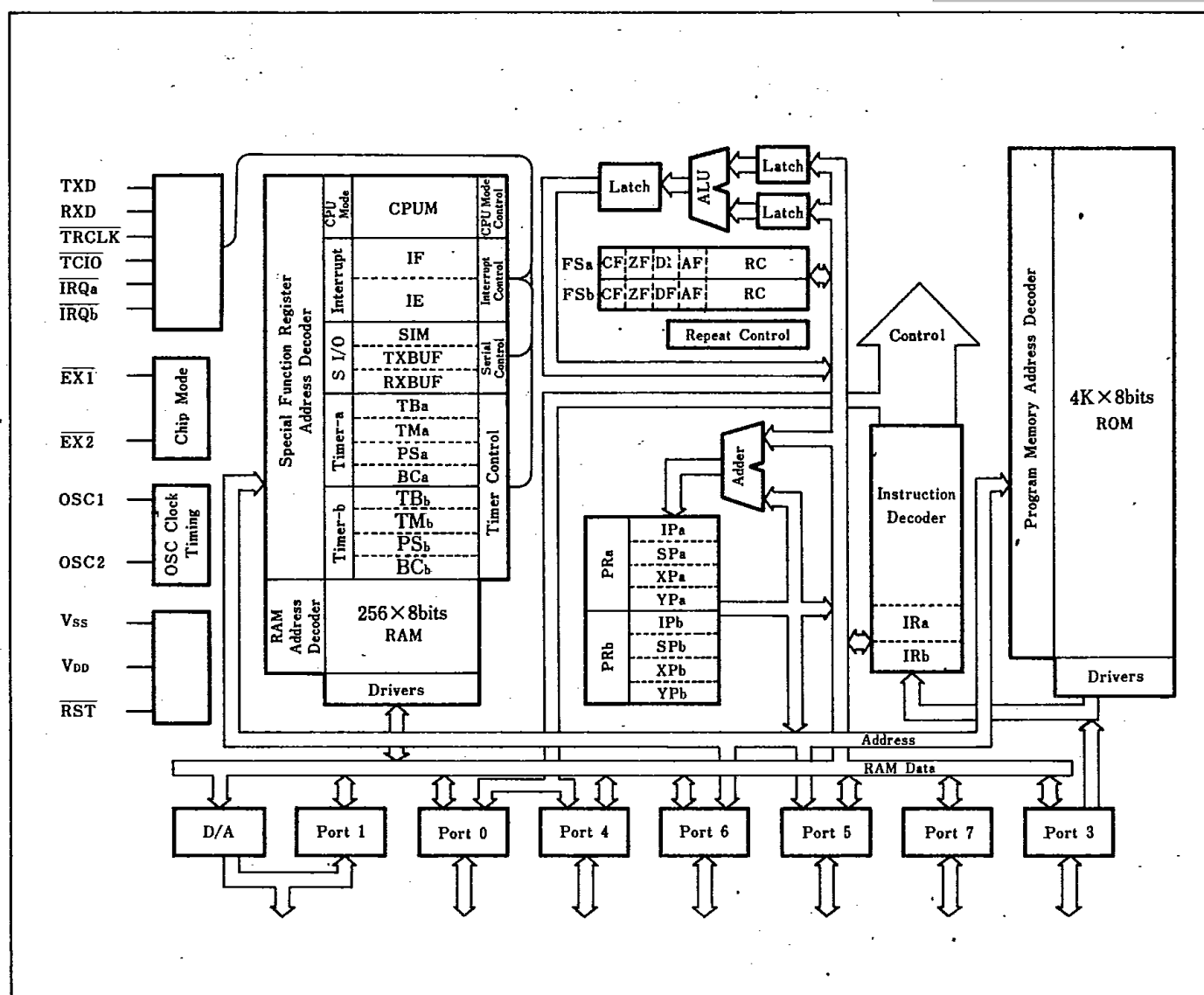
- 高速な命令実行 (最小 500ns, 平均 1.5μs, 8MHz 発振時)
- 誤動作防止機能 (暴走時トラップ割込み, 低電圧検出)
- 外部 ROM, RAM 拡張可能 (最大, 各 64K バイト)
- 豊富な入出力端子 (51 本) (内 D/A 出力 4 本)
- クロック発振回路内蔵 (セラミック発振子, Xtal 用)
- 64 ピン・プラスチック DIL パッケージ (シュリンク)
- NMOS 5V 単一電源

■ 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

■ 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なに応じて 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り換えが可能です。

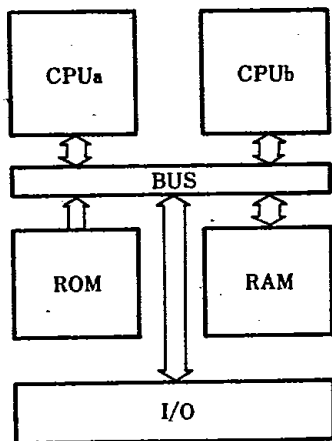


図 1 MN1890シリーズの基本構成

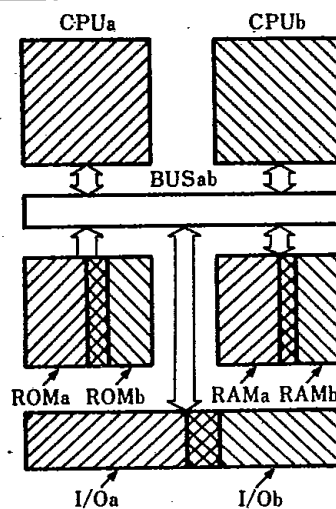


図 2 デュアルマイクロコンピュータの割付け

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

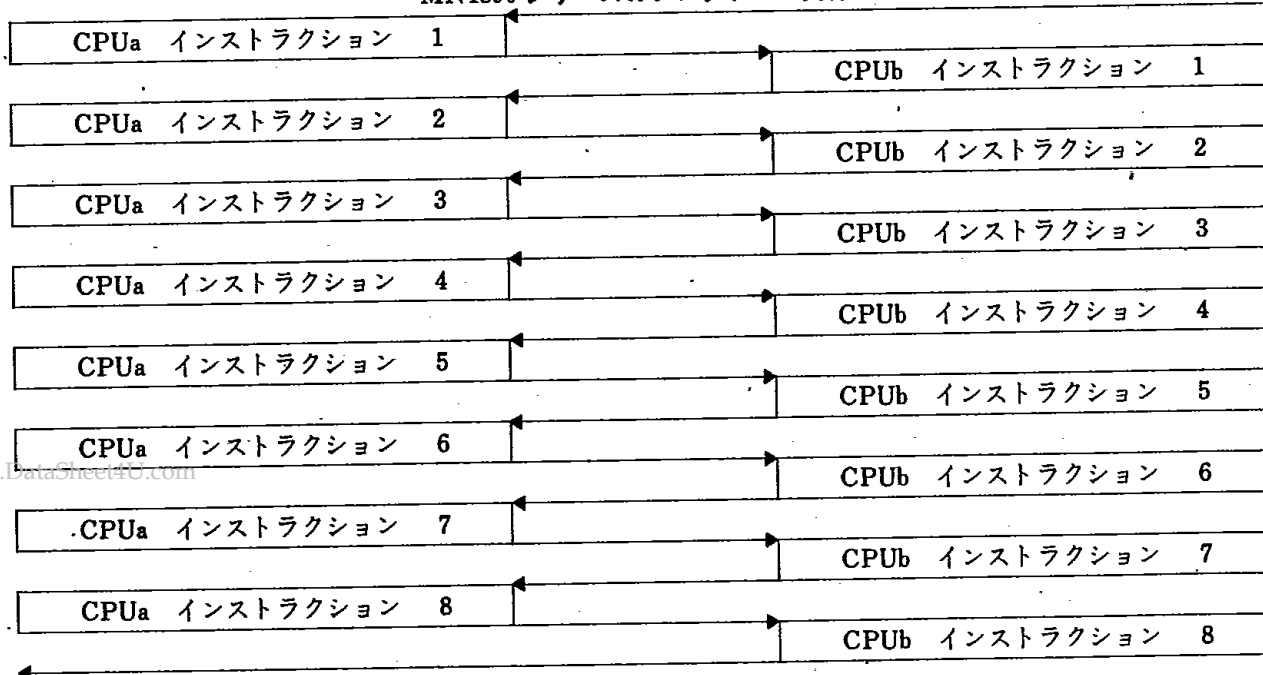


図 3 デュアルマイクロコンピュータ動作

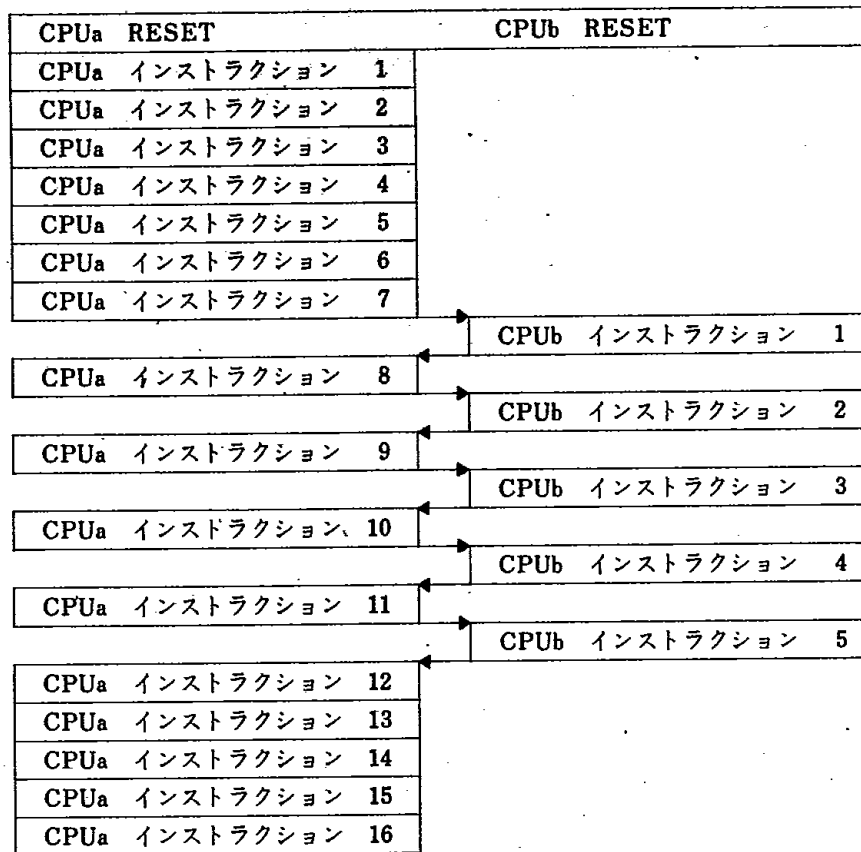
リセットサイクル(イニシャライズ)

オートスワップ開始

(SWAP=1 セット)

オートスワップ停止

(SWAP=0 リセット)



(a) AUTO-SWAP

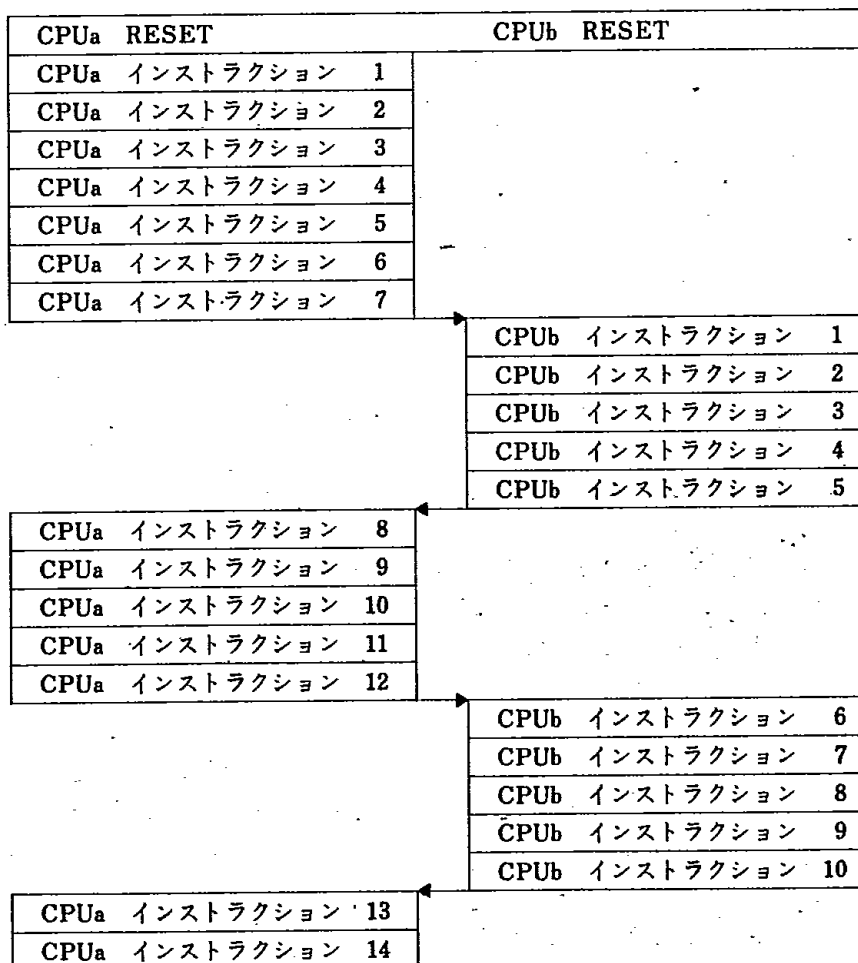
リセットサイクル
(イニシャライズ)

CPUb モード要求
(CPUb=1)

CPUa モード実行

CPUb モード要求
(CPUb=1)

CPUa モード実行



CPUb モード実行

CPUa モード要求
(CPUb=0)

CPUb モード実行

CPUa モード要求
(CPUb=0)

(b) PROGRAM SWAP

図4 プログラム実行モード

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の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。

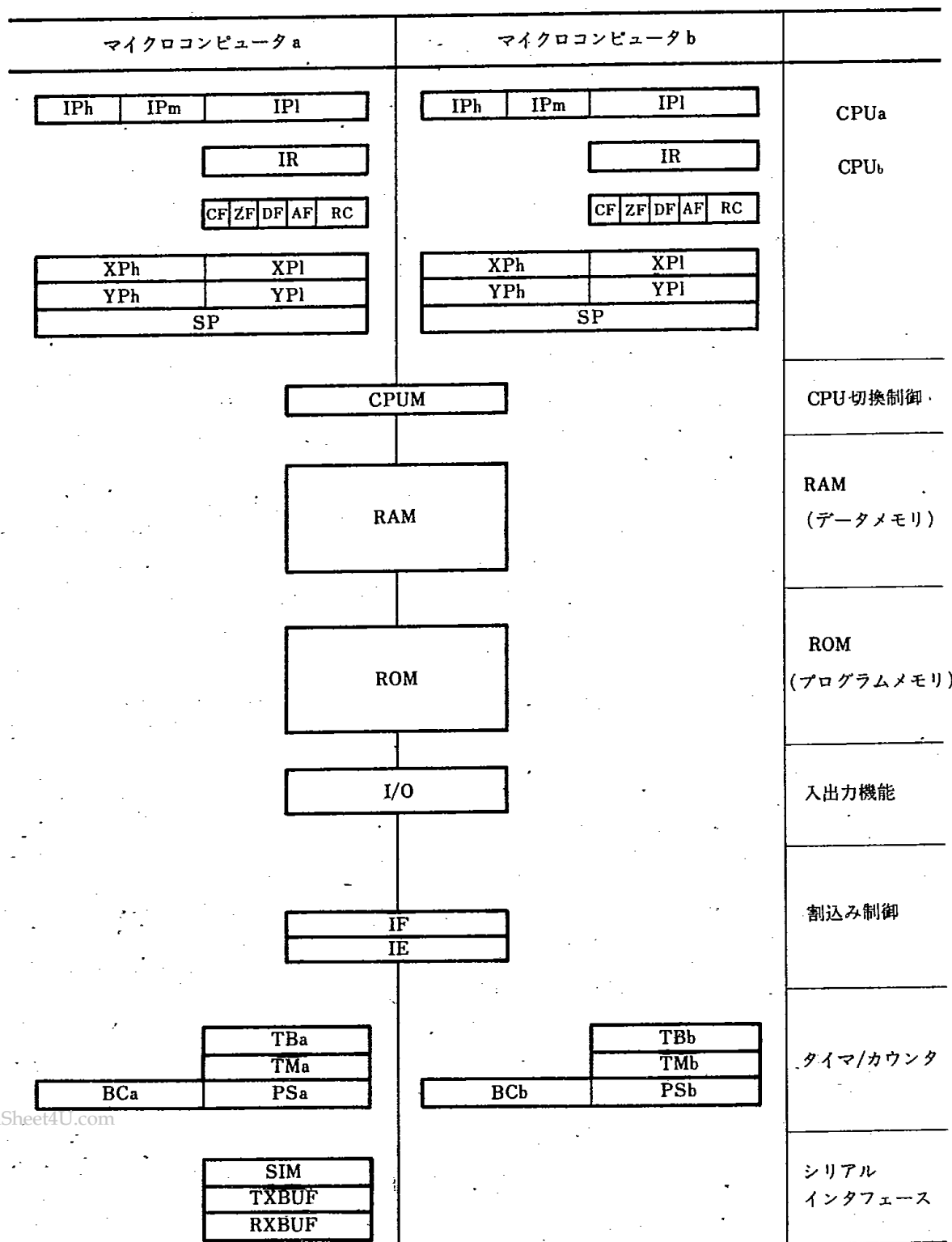


図5 MN1890シリーズのレジスタ構成

■ 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、8 K 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。

www.DataSheet4U.com

■ ブロックの機能 (つづき)

ブロック		機 能														
メモリー部	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。														
スワッチ制御部	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。 ●プリスケアラ (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、プリスケア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 TRCLK TCIO IRQa IRQb)	下記の特種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 (TRCLK) タイマカウンタ入出力 (TCIO) CPUa 外部割込み入力 (IRQa)、CPUb 外部割込み入力 (IRQb)														
	Chip Mode (EX1 EX2)	チップモード (Chip Mode) は、EX1、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。 <table><tr><td>動作モード</td><td>EX1</td><td>EX2</td></tr><tr><td>シングルチップモード</td><td>開放</td><td>開放</td></tr><tr><td>ROM 拡張モード</td><td>開放</td><td>L</td></tr><tr><td>RAM 拡張モード</td><td>L</td><td>開放</td></tr><tr><td>ROM, RAM 拡張モード</td><td>L</td><td>L</td></tr></table>	動作モード	EX1	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
動作モード	EX1	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 Timing	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 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) RST はリセット端子であり、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。
$\overline{\text{EX1}}$ $\overline{\text{EX2}}$	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。 $\overline{\text{EX1}}$ 端子には、内部クロック S2 に同期したパルスを出し、 $\overline{\text{EX2}}$ 端子には、S0 に同期したパルスを出します。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。
$\overline{\text{RST}}$	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。 $\overline{\text{RST}}$ 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。 $\overline{\text{RST}}$ 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 $\overline{\text{RST}}$ 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。
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。
$\overline{\text{TCIO}}$	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、イベントカウンタモード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。
$\overline{\text{IRQA}}$ $\overline{\text{IRQB}}$	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。

■ 命令マップ / Instruction Map

www.DataSheet4U.com

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP															
1																
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

■ MN1890 Series 命令セット / Instruction Set

Group	Mnemonic	Group	Mnemonic	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	Arithmetic Group	ADD (DOUBLE OPERAND) ADDD (DOUBLE OPERAND) ADDR XPI, XPI, XPh YPI, YPI, YPh XPI, YPI, imm YPI, YPI, imm XPI, (da1), imm YPI, (da1), imm XPI, (da1), (da2) YPI, (da1), (da2)	Branch Group	BR \$label label /label
	MVROM (XP), (YP)		INC (SINGLE OPERAND) XP YP		BC \$label BZ \$label BLE \$label BNC \$label BNZ \$label BGT \$label SKIP imm T1BNZ (da)bp, \$label T1BZ (da)bp, \$label
	MV1 (dad)bp, (das)bps MV1CPL (dad)bp, (das)bps		DEC (SINGLE OPERAND) XP YP		LOOP \$label
	XCH XP, YP XP, SP XPI, XPh YPI, YPh XPI, (da) YPI, (da) (XP), (YP) (da), (da)		SUB (DOUBLE OPERAND) SUBD (DOUBLE OPERAND)		RLOOP XP, \$label YP, \$label
	XCH4 (SINGLE OPERAND)		MUL (XP) DIV (XP)		CMPBNE (XP), imm, \$label (da), imm, \$label
	CLR (SINGLE OPERAND) (da)bp DF CF		CMP (DOUBLE OPERAND) XPI, imm YPI, imm XPI, (da) YPI, (da)		CMPBE (XP), imm, \$label (da), imm, \$label
	SET (da)bp DF CF		CMPM (XP)mask, imm (da)mask, imm (XP)mask, (YP)mask (da1)mask, (da2)mask		CALL label /label
Logical Group	CPL (SINGLE OPERAND)	Stack, Hardware Group	PUSH (SINGLE OPERAND) XP YP imm FS	**** (SINGLE OPERAND) **** (XP) (da)	
	ROL (SINGLE OPERAND)		POP (SINGLE OPERAND) XP YP FS	**** (DOUBLE OPERAND) **** (XP), (YP) (YP), imm (dad), (das) (da), imm	
	ROR (SINGLE OPERAND)		REP imm4		
	AND (DOUBLE OPERAND)		PI		
	OR (DOUBLE OPERAND)		NOP		
	XOR (DOUBLE OPERAND)				

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

Group	Mnemonic	Operation	CZ	in	machine code (hexa.)	I/O
	XCH XP, YP XP, SP XPI, XPh YPI, YPh XPI, (da) YPI, (da) (XP), (YP) (da1), (da2)	XP $\leftrightarrow$ YP XP $\leftrightarrow$ SP XPI $\leftrightarrow$ XPh YPI $\leftrightarrow$ YPh XPI $\leftrightarrow$ RAM(XPh $\wedge$ DF, da) YPI $\leftrightarrow$ RAM(YPh $\wedge$ DF, da) RAM(XP) $\leftrightarrow$ RAM(YP) RAM(XPh $\wedge$ DF, da1) $\leftrightarrow$ RAM(YPh $\wedge$ DF, da2)			D7 D5 C4 C6 L C5 : da L C7 : da LL 58 LL 5A : da2 : da1	120 120 *21 *21 120 120 *51 15H
	XCH4 (XP) (da)	RAM(XP)un $\leftrightarrow$ RAM(XP)in RAM(XPh $\wedge$ DF, da)un $\leftrightarrow$ RAM(XPh $\wedge$ DF, da)in		L	46 47 : da	*21 12H
	CLR (XP) (da) (da)bp DF CF	RAM(XP) $\leftarrow$ 0 RAM(XPh $\wedge$ DF, da) $\leftarrow$ 0 RAM(XPh $\wedge$ DF, da)bp $\leftarrow$ 0 DF $\leftarrow$ 0 CF $\leftarrow$ 0	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 $\wedge$ DF, da)bp $\leftarrow$ 1 DF $\leftarrow$ 1 CF $\leftarrow$ 1	1	L	18+bp : da(bp = 0 ~ 7) 8D 8F	12H *21 *21
Logical Group	CPL (XP) (da)	RAM(XP) $\leftarrow$ RAM(XP) . cpl. RAM(XPh $\wedge$ DF, da) $\leftarrow$ RAM(XPh $\wedge$ DF, da) . cpl.	Z Z	L L	42 43 : da	*21 12H
	ROL (XP) (da)	CF $\leftarrow$ RAM(17 : RAM()bp $\leftarrow$ RAM()bp-1 : RAM()0 $\leftarrow$ CF [RAM() = RAM(XP) or RAM(XPh $\wedge$ DF, da)]	C C	L L	4C 4D : da	*21 12H
	ROR (XP) (da)	CF $\leftarrow$ RAM(10 : RAM()bp-1 $\leftarrow$ RAM()bp : RAM()17 $\leftarrow$ CF [RAM() = RAM(XP) or RAM(XPh $\wedge$ DF, da)]	C C	L L	4E 4F : da	*21 12H
Logical Group	AND (XP), (YP) (XP), imm (dad), (das) (da), imm	RAM(XP) $\leftarrow$ RAM(XP) . $\wedge$. RAM(YP) RAM(XP) $\leftarrow$ RAM(XP) . $\wedge$. imm RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) . $\wedge$. RAM(YPh $\wedge$ DF, das) RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) . $\wedge$. imm	Z Z Z Z	LT L L 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) $\leftarrow$ RAM(XP) . $\vee$. RAM(YP) RAM(XP) $\leftarrow$ RAM(XP) . $\vee$. imm RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) . $\vee$. RAM(YPh $\wedge$ DF, das) RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) . $\vee$. imm	Z Z Z Z	LT L L 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) $\leftarrow$ RAM(XP) . $\vee$. RAM(YP) RAM(XP) $\leftarrow$ RAM(XP) . $\vee$. imm RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) . $\vee$. RAM(YPh $\wedge$ DF, das) RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) . $\vee$. imm	Z Z Z Z	LT L L 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) $\leftarrow$ RAM(XP) + RAM(YP) + CF RAM(XP) $\leftarrow$ RAM(XP) + imm + CF RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) + RAM(YPh $\wedge$ DF, das) + CF RAM(XPh $\wedge$ DF, da) $\leftarrow$ RAM(XPh $\wedge$ DF, da) + imm + CF	CZ CZ CZ CZ	LT L L L	78 79 : imm 7A : das : dad 7B : imm : da	*31 13H 13H 13H
	ADD4 (XP), (YP) (XP), imm4 (dad), (das) (da), imm4	RAM(XP)in $\leftarrow$ RAM(XP)in + RAM(YP)in + CF RAM(XP)in $\leftarrow$ RAM(XP)in + imm4 + CF RAM(XPh $\wedge$ DF, dad)in $\leftarrow$ RAM(XPh $\wedge$ DF, dad)in + RAM(YPh $\wedge$ DF, das)in + CF RAM(XPh $\wedge$ DF, da)in $\leftarrow$ RAM(XPh $\wedge$ DF, da)in + imm4 + CF	CZ CZ CZ CZ	LT L L L	7C 7D : imm4 7E : das : dad 7F : imm4 : da	*41 14H 14H 14H
	ADDR XPI, XPI, XPh YPI, YPI, YPh XPI, XPI, imm YPI, YPI, imm XPI, (da), imm YPI, (da), imm XPI, (da1), (da2) YPI, (da1), (da2)	XPI $\leftarrow$ XPI + XPh YPI $\leftarrow$ YPI + YPh XPI $\leftarrow$ XPI + imm YPI $\leftarrow$ YPI + imm XPI $\leftarrow$ RAM(XPh $\wedge$ DF, da) + imm YPI $\leftarrow$ RAM(YPh $\wedge$ DF, da) + imm XPI $\leftarrow$ RAM(XPh $\wedge$ DF, da1) + RAM(YPh $\wedge$ DF, da2) YPI $\leftarrow$ RAM(XPh $\wedge$ DF, da1) + RAM(YPh $\wedge$ DF, da2)	CZ CZ CZ CZ CZ CZ CZ CZ	LT L L L T T T 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 $\leftarrow$ XP + 1 YP $\leftarrow$ YP + 1 RAM(XP) $\leftarrow$ RAM(XP) + 1 RAM(XPh $\wedge$ DF, da) $\leftarrow$ RAM(XPh $\wedge$ DF, da) + 1	CZ CZ	L L	E5 E7 48 49 : da	*1H *1H *21 12H
	DEC XP YP (XP) (da)	XP $\leftarrow$ XP - 1 YP $\leftarrow$ YP - 1 RAM(XP) $\leftarrow$ RAM(XP) - 1 RAM(XPh $\wedge$ DF, da) $\leftarrow$ RAM(XPh $\wedge$ 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) $\leftarrow$ RAM(XP) - RAM(YP) - CF RAM(XP) $\leftarrow$ RAM(XP) - imm - CF RAM(XPh $\wedge$ DF, dad) $\leftarrow$ RAM(XPh $\wedge$ DF, dad) - RAM(YPh $\wedge$ DF, das) - CF RAM(XPh $\wedge$ DF, da) $\leftarrow$ RAM(XPh $\wedge$ DF, da) - imm - CF	CZ CZ CZ CZ	LT L L L	70 71 : imm 72 : das : dad 73 : imm : da	*31 13H 13H 13H
	SUB4 (XP), (YP) (XP), imm4 (dad), (das) (da), imm4	RAM(XP)in $\leftarrow$ RAM(XP)in - RAM(YP)in - CF RAM(XP)in $\leftarrow$ RAM(XP)in - imm4 - CF RAM(XPh $\wedge$ DF, dad)in $\leftarrow$ RAM(XPh $\wedge$ DF, dad)in - RAM(YPh $\wedge$ DF, das)in - CF RAM(XPh $\wedge$ DF, da)in $\leftarrow$ RAM(XPh $\wedge$ DF, da)in - imm4 - CF	CZ CZ CZ CZ	LT L L L	74 75 : imm4 76 : das : dad 77 : imm4 : da	*41 14H 14H 14H
	MUL DIV (XP) (XP)	RAM(XP+1), RAM(XP) $\leftarrow$ RAM(XP) $\times$ RAM(XP+1) RAM(XP+1), RAM(XP) $\leftarrow$ RAM(XP) / YPI		TT TT	59 61	*81 *81
	CMP XPI, imm YPI, imm XPI, (da) YPI, (da) (XP), (YP) (XP), imm (da1), (da2) (da), imm	XPI - imm YPI - imm XPI - RAM(XPh $\wedge$ DF, da) YPI - RAM(YPh $\wedge$ DF, da) RAM(XP) - RAM(YP) RAM(XP) - imm RAM(XPh $\wedge$ DF, da1) - RAM(YPh $\wedge$ DF, da2) RAM(XPh $\wedge$ DF, da) - imm	CZ CZ CZ CZ CZ CZ CZ CZ	T 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) . $\wedge$. mask] - imm [RAM(XPh $\wedge$ DF, da) . $\wedge$. mask] - imm [RAM(XP) . $\wedge$. mask] - [RAM(YP) . $\wedge$. mask] [RAM(XPh $\wedge$ DF, da1) . $\wedge$. mask] - [RAM(YPh $\wedge$ DF, da2) . $\wedge$. 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.)	I/O
Branch Group	BR label /label	IP ← IP+2 + \$label branch relative (+127 ~-128) IP ← IPh, label branch absolute (4K-byte/page) IP ← /label branch absolute long (64K-byte)			B8 : \$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			B1 : \$label	130
	BZ \$label	if ZF=1 then IP ← IP+2 + \$label / else IP ← IP+2			B2 : \$label	130
	BLE \$label	- if CF=1, or, ZF=1 then IP ← IP+2 + \$label / else IP ← IP+2			B3 : \$label	130
	BNC \$label	if CF=0 then IP ← IP+2 + \$label / else IP ← IP+2			B9 : \$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			B8 : \$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 then IP ← IP+3 + \$label / else IP ← IP+3 if RAM(XPh ∧ DF, da) ≠ imm 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 then IP ← IP+3 + \$label / else IP ← IP+3 if RAM(XPh ∧ DF, da)=imm 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-byte) IP ← IPh, label RAM(SP-1) ← IPh, IPm call absolute (64K-byte) IP ← /label RAM(SP-2) ← IPI hardware interrupt SP ← SP-2			90+label-m : label-l F7 : label-h, m : label-l don't care :	*30 *30 *50 *80
	RET	IPI ← RAM(SP) : IPh, IPm ← RAM(SP+1) : SP ← SP+2		L	F4	*40
	RETI	repeat FS ← RAM(SP) normal FS ← RAM(SP) IPI ← RAM(SP+1) IPI ← RAM(SP+1) IPh, IPm ← RAM(SP+2) IPh, IPm ← RAM(SP+2) IR ← RAM(SP+3) SP ← SP+3 SP ← SP+4	CZ	L	F5 normal	*50 *51
Stack, Hardware Group	PUSH XP YP (XP) (da) imm FS	RAM(SP-1) ← XPh : RAM(SP-2) ← XPI : SP ← SP-2 push stack RAM(SP-1) ← YPh : RAM(SP-2) ← YPI : 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		L L	BC BE BF BD : da FB : imm BB	*31 *31 *31 *31 12H *21
	POP XP YP (XP) (da) FS	XPI ← RAM(SP) : XPh ← RAM(SP+1) : SP ← SP+2 pop stack YPI ← RAM(SP) : YPh ← 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		L L L CZ	B4 B6 B7 B5 : da B3	*31 *31 *21 *20 *21 *21
	PI	RAM(SP-1) ← IPh, IPm : IPh, IPm ← ROM(VECTOR) interrupt RAM(SP-2) ← IPI : IPI ← ROM(VECTOR+1) RAM(SP-3) ← FS : SP ← SP-3			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) IPb ← ROM(11), ROM(10) SPa ← X'130 SPb ← X'110 IF, IE, CPUM, SIM, TMa, TMb ← 0 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, XPI, YPh, YPI,
(XP), (YPI), (SP), (da), (das), (dad), (da1), (da2)
imm, imm4, mask
bp, bps, bpd
da, das, dad, da1, da2
DF
CF
ZF
un, in
IP, IPh, IPm, IPI
FS
IR

— Pointer Register
— RAM data ()
— immediate data
imm4 = 4-bit data
— bit position
— direct address
— direct flag (reset = 0, ..., RAM address upper 8-bit = 0 at : da addressing mode)
— carry / borrow flag
— zero flag
— upper / lower nibble (4-bit)
— instruction pointer h(4-bit), m(4-bit), l(8-bit)
— flag status (CF, ZF, AF, DF, RC)
— instruction register

■ 絶対最大定格/Absolute Maximum Ratings ($T_a=25^{\circ}\text{C}$)

Item	Symbol	Rating	Unit
電源電圧	V_{DD}	$-0.3 \sim +7.0$	V
入力電圧	V_I	$-0.3 \sim +7.0$	V
出力端子電圧	V_O	$-0.3 \sim +7.0$	V
動作周囲温度	T_{opr}	$-20 \sim +70$	$^{\circ}\text{C}$
保存温度	T_{stg}	$-55 \sim +125$	$^{\circ}\text{C}$

■ DC 特性/DC Characteristics ($V_{DD}=4.5 \sim 5.5\text{V}$, $T_a=25^{\circ}\text{C}$)

Item	Symbol	Condition	min.	typ.	max.	Unit
電源電流	I_{DD}	$V_{DD}=5\text{V}$		180*		mA
消費電力	P_{tot}	外付負荷なし		900		mW

入力端子 1 P21(RXD), P24($\overline{\text{IRQB}}$), P25($\overline{\text{IRQA}}$), 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}	$V_{DD}=5\text{V}$ $V_I=0.8\text{V}$			-250	μA

入出力端子 P0~P1, T20(TXD), P22($\overline{\text{TRCLK}}$), P23($\overline{\text{TCIO}}$), P3~P7**

出力電圧ハイレベル	V_{OH3}	$I_{OH}=-100\mu\text{A}$	3.0			V
出力電圧ローレベル	V_{OL3}	$I_{OL}=1.2\text{mA}$			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}	$V_I=0.8\text{V}$			-600	μA

端子容量

入力端子	C_I	$V_I=0.8\text{V}$		5		pF
入出力端子	C_B	$V_O=2\text{V}$		15		pF
OSC 端子	C_{OSC}	$V_{OSC}=2\text{V}$		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。