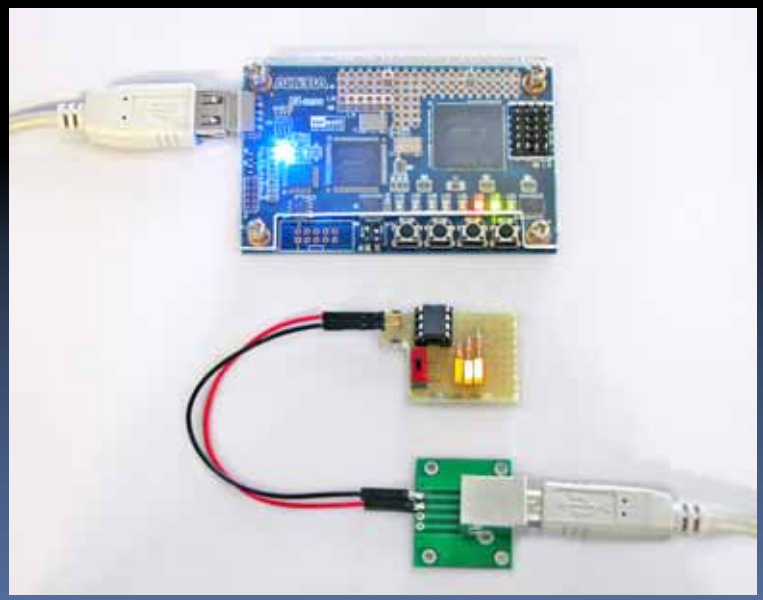
株式会社ソリトンウェーブ
芹井 滋喜

CPLDで200MHZ動作のPICを作る

PICのエミュレーションについて

- HDLの学習用として、PICマイコンを、PLDを使ってPICをエミュレーションします。
- PICは、最も基本的な構造のPIC12F508をエミュレーションします。
- CPLDには、MAX IIを搭載した、MAX IIマイクロキット（写真上部）を使用し、200MHz相当の動作をさせます。



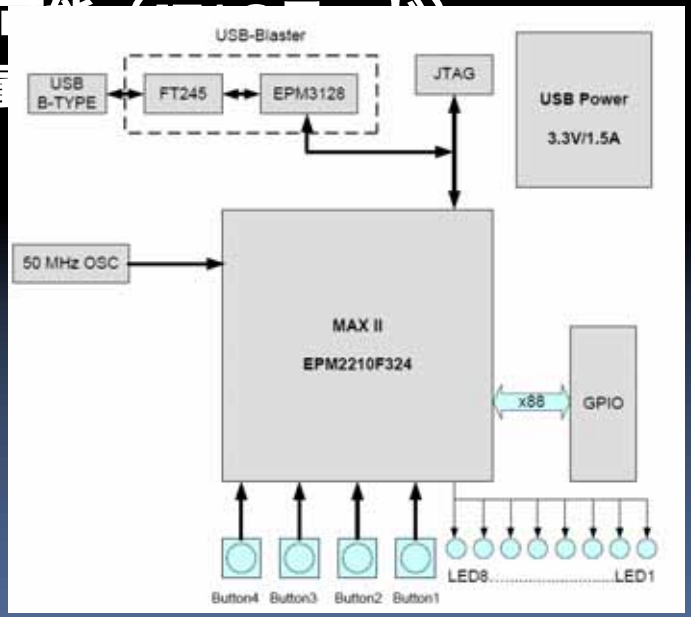
CPLDエミュレーションのねらい

- 具体的な目標を持って、実用的なHDLの学習を行う
- MPUの内部には、HDLの基礎的な要素が満載(カウンタ、デコーダ、レジスタ、ROM、etc)
- 目標が明確なため、退屈せずに学習が進められる
- HDLの学習と同時に、PICの内部回路の理解も深まる
- PICの外部回路もCPLD内部に収めることができるため、容易に周辺デバイスのテストが行える
- シミュレータを使って、PC上でデバッグが行えるため、検証が容易で、改版なども即座に行える
- 内部周辺デバイスを拡張したり、新たな周辺デバイスを追加するといったことも簡単に行える

MAX IIマイクロキットについて

MAX IIマイクロキットは、ALTERA社のCPLD MAX II を搭載した開発基板で、以下のような特徴があります。

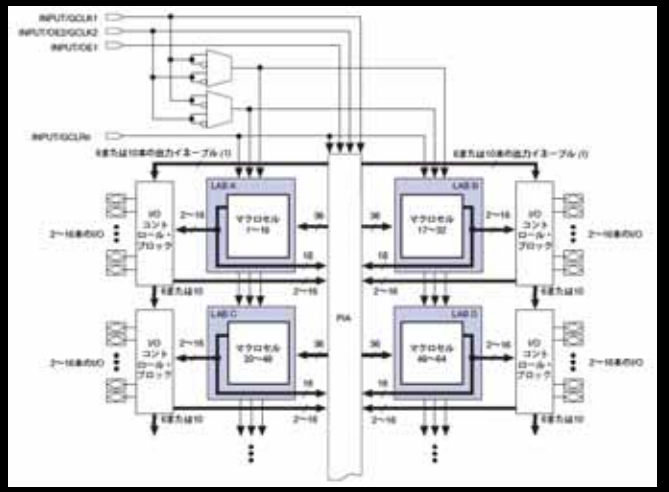
- MAX II　EPM2210F324C3（2,210LE,等価マクロセル1,700)を使用
- USBブラスタ回路内蔵のため、ダウンロードケーブル不要
- USBブラスタケーブルとしても使用可能（JTAGモード）
- Altera Quartus II DVDバージョン付属
- 8個のユーザLED,4個のスイッチ
- DE1/DE2互換の４０ピン拡張ポート
- プロトタイプエリアA－68 GPIO
- プロトタイプエリアB－20 GPIO
- USBケーブル付属



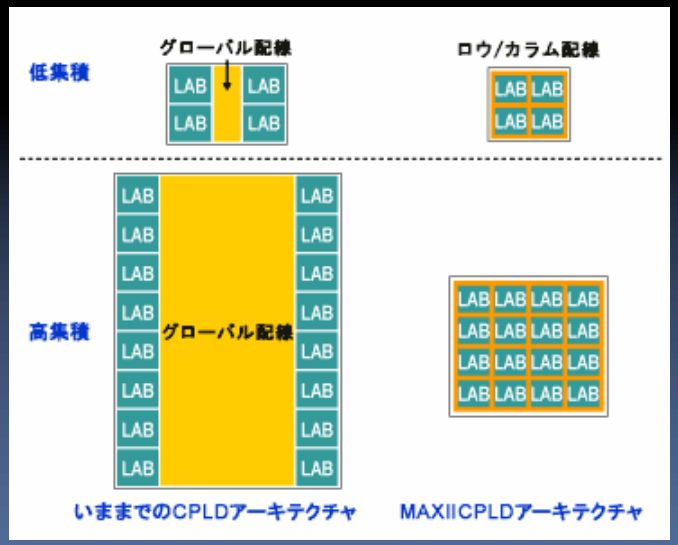
MAX IIマイクロキットブロック図
（MAX IIマイクロキット ユーザーマニュアルより）

MAX IIの概要(1)

- MAX IIはCPLDの使い勝手とFPGAを兼ね備えたデバイス
- CPLDは、高集積になると、グローバル配線エリアが増大し、高集積には限界がある
- MAX IIは、LABを図のように格子状に配置し、この問題を解決し、高集積化を可能にしている
- 再ロード用のフラッシュ・メモリ・デバイスを内蔵する事により、CLPDの使い勝手を維持している



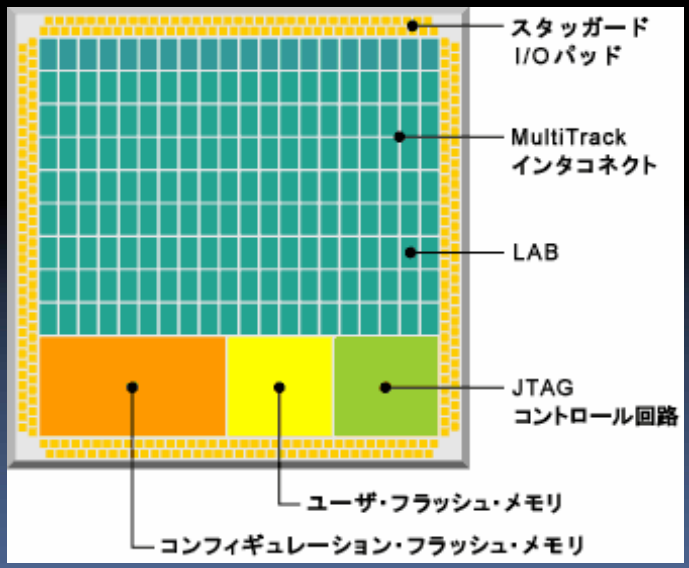
典型的なCPLDのブロック図



CPLDアーキテクチャとMAX IIアーキテクチャの比較

MAX IIの概要(2)

- コンフィグレーション・フラッシュ・メモリ内蔵
- ユーザ・フラッシュ・メモリ内蔵(512ワード)
- オンボード・オシレータ内蔵
- 低消費電力
- 1.8 V コア電圧
- 低スタンバイ消費電力
- 自動スタート／ストップ機能
- コスト最適化アーキテクチャ
- 最大300MHz の内部クロック周波数レートをサポート
- MultiVolt コア
- 3.3 V、2.5 V、または 1.8 V 電源に対応可能なオンチップ電圧レギュレータ
- シュミット・トリガ、プログラマブル・スルー・レートおよびプログラマブル・ドライブ強度

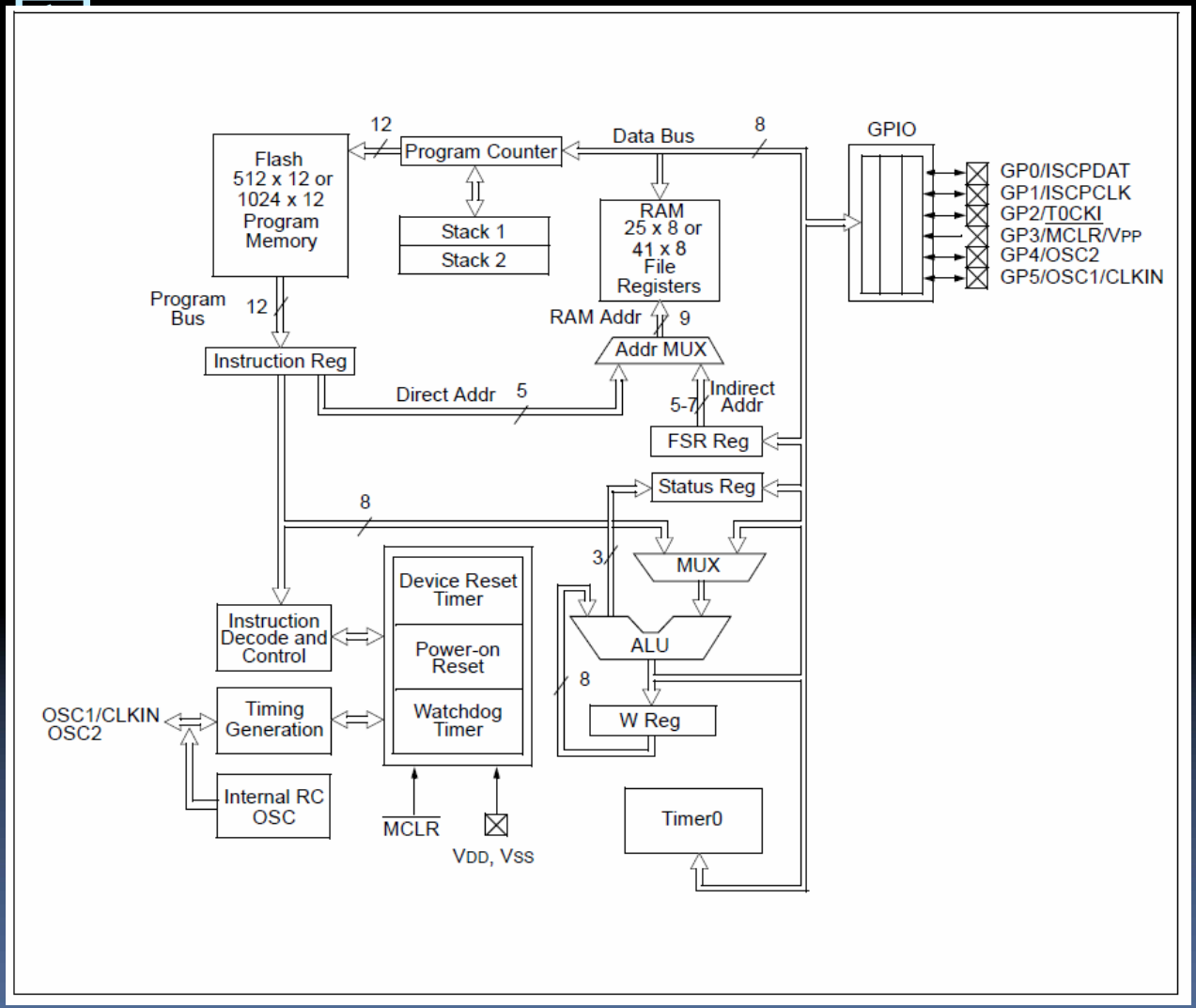


MAX IIのフロアプラン

PIC12F508の概要(1)

- ハイパフォーマンスRISC CPU
- 33個のシングル・サイクル・インストラクション
- 12ビット幅のインストラクション
- 2レベルのハードウェア・スタック
- 直接／間接、および相対アドレスモード
- 8ビット幅のデータバス
- 8個の特殊機能ハードウェア・レジスタ
- 512ワードのフラッシュプログラムメモリ
- 25バイトのデータ用SRAM
- 4MHz±１%の校正済み内部発振回路
- パワーオン・リセット
- オンチップRCオシレータ内蔵のウォッチドッグ・タイマ
- パワー・セービング・スリープモード
- 6個のI/Oピン（5個のI/Oピン＋1個の入力専用ピン）
- 8ビットのリアルタイム・クロック／カウンタ(TMR0)
- 8ビットのプログラマブル・プリスケーラ

PIC12F508の概要(2) –ブロック図



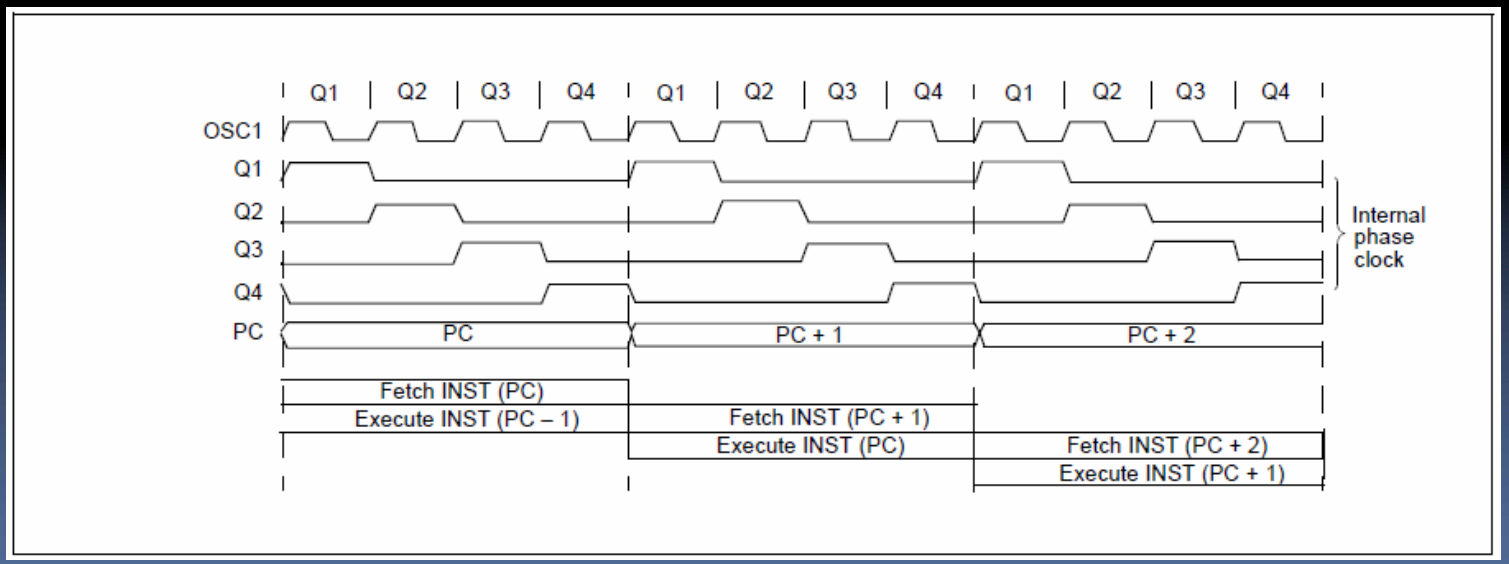
※Watchdog Timerは、エミュレータでは省略

ハーバードアーキテクチャ

- ノイマン型のアーキテクチャと異なり、命令用とデータ用のメモリを分離している
- ノイマン型では、命令とデータのメモリが、同一バス上にあるため、命令実行中にデータにアクセスする場合、速度が低下する
- ハーバードアーキテクチャでは、命令とデータでバスが分離されているため、データアクセス時も命令用のメモリは独立しているため、高速動作が可能
- PICでは、ハーバードアーキテクチャを採用することで、１命令１マシンサイクルを実現している

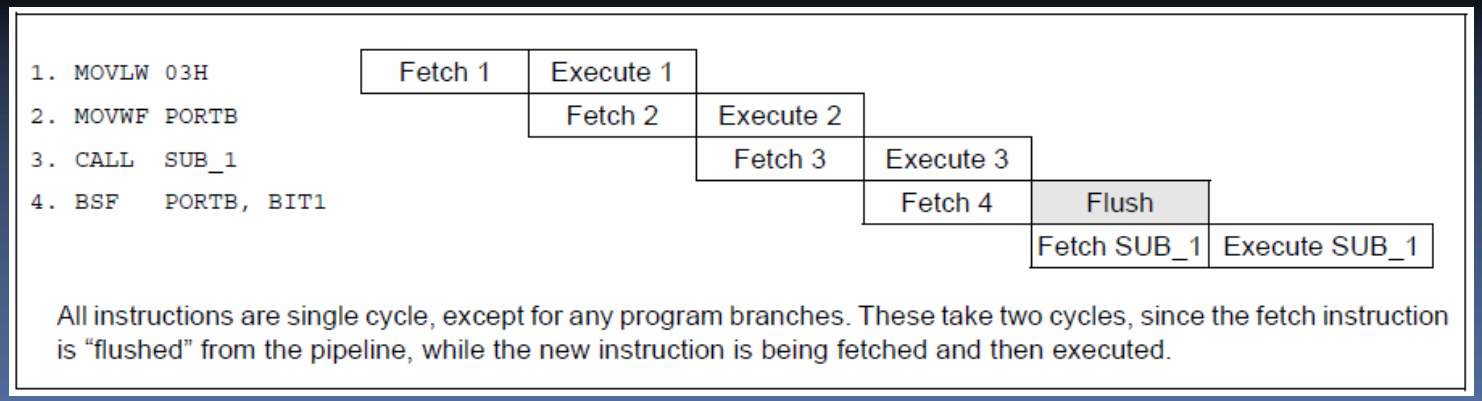
インストラクションサイクル

- PIC12F508は、4クロックで1サイクル
- 1サイクルで1命令を処理する(分岐命令を除く)



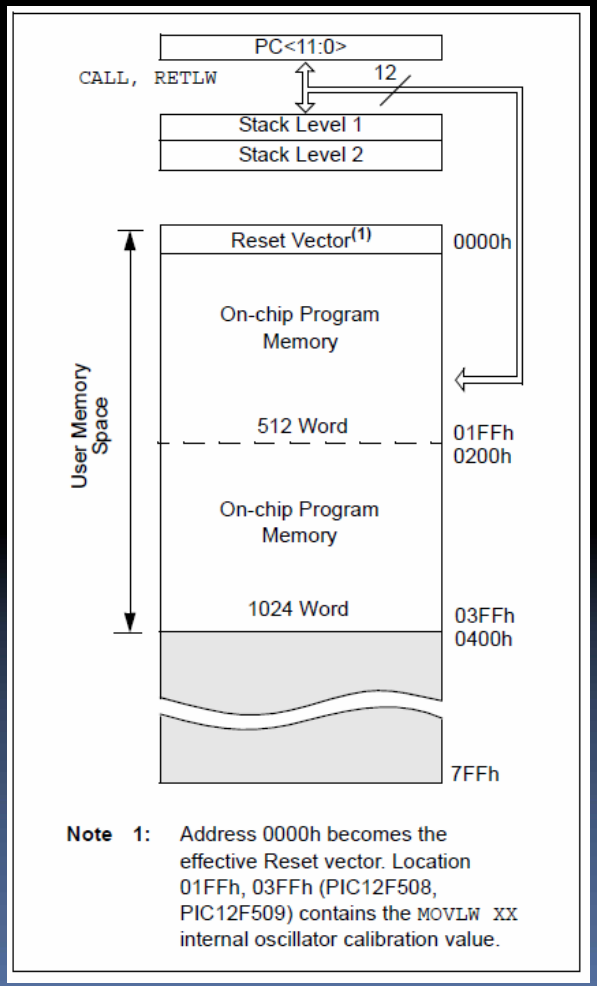
パイプライン動作

- PIC12F508は、1命令のキャッシュを行い、パイプライン動作をしている
- パイプライン動作により、実行する命令は、常に1つ前の命令となる
- 分岐命令の場合は、キャッシュをフラッシュし、新しい命令を読み直すため、2サイクルの動作となる



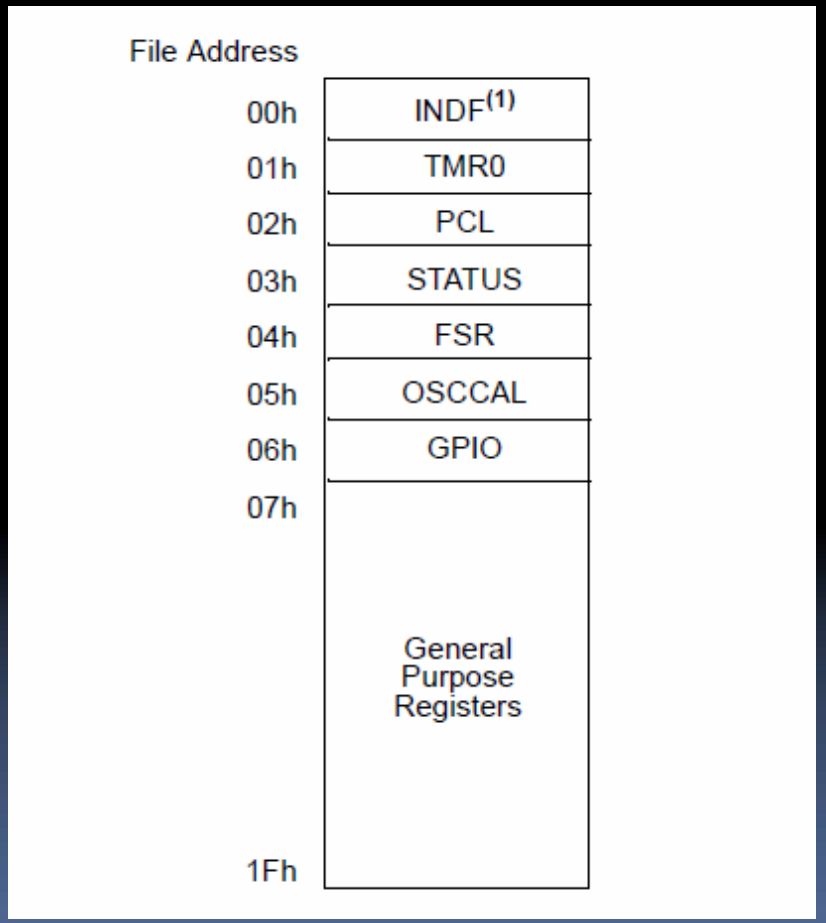
プログラム・メモリ

- 512ワードのプログラム・メモリを内蔵
- リセットベクタは000h番地
- リセット時はメモリの最終番地でスタートし、OSCのキャリブレーションデータを読み込んでから、リセットベクタに移動する



ファイルレジスタ

- 00h～06hは、特殊機能レジスタ(SFR)で、CPUや周辺デバイスの制御に利用される
- PICのデータメモリは、ファイルレジスタの一部をデータ用に利用する
- PIC12F508では、07h～1Fhまでの25バイトがデータ用として利用可能



ステータス・レジスタ（STATUS）

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
GPWUF	—	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

- GPWUF
  通常０で、スリープからピンチェンジで復帰すると、１となります
- PA0(*)
  プログラムページの切り替えビットですが、PIC12F508では未使用です。PIC12F509との互換性のために、存在しています。
- TO#（Time-Out）(*)
  通常１で、WDT（ウォッチドッグ・タイマ）のタイムアウトが発生すると、０になります。
- PD#(Power-Down) (*)
  通常１で、SLEEP命令を実行すると、０となります。
- Z(Zero)
  演算結果のフラグビットで、演算結果がゼロの時、１となります。
- DC(Digit Carry/Borrow)
  演算結果のフラグビットで、下位4ビットからの桁上げがあった場合、１となります。
- C(Carry/Borrow)
  演算結果のフラグビットで、桁上げがあった場合、１となります。

***エミュレータでは実装しない**

オプション・レジスタ（OPTION）

W-1	W-1	W-1	W-1	W-1	W-1	W-1	W-1
$\overline{\text{GPWU}}$	$\overline{\text{GPPU}}$	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

- GPWU#(*)
  ピンチェンジ（GP0,GP1,GP3）のウェイクアップを制御します。０でピンチェンジが有効になり、１で無効になります。
- GPPU#(*)
  GPIO（GP0,GP1,GP3）**の内部プルアップを制御します。０でプルアップが有効になり、１で無効になります。**
- T0CS(*)
  タイマ０（TMR0）**のクロック・ソースを設定します。**
  １：T0CKI**ピンを使用する**
  ０：内部インストラクション・サイクルの、FOSC/4**を使用する。**
- T0SE(*)
  タイマ０（TMR0）**のクロック・エッジを選択します。**
  １：T0CKI**ピンの立下りでインクリメントします。**
  ０：T0CKI**ピンの立上りでインクリメントします。**
- PSA(*)
  プリスケーラのターゲットを選択します。
  １：プリスケーラを、WDT**に使用します。**
  ０：プリスケーラを、タイマに使用します。
- PS<2:0>
  プリスケーラの値です。

*エミュレータでは実装しない

プリスケーラの設定値

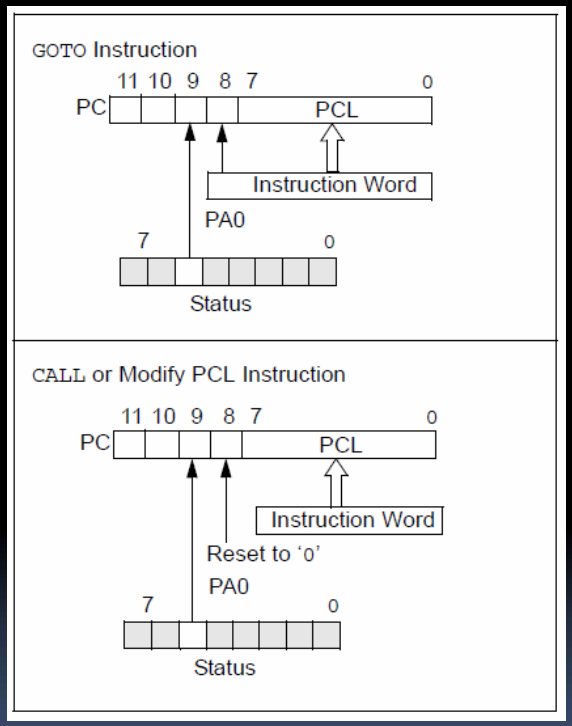
Bit Value	Timer0 Rate	WDT Rate
3'b000	1:2	1:1
3'b001	1:4	1:2
3'b010	1:8	1:4
3'b011	1:16	1:8
3'b100	1:32	1:16
3'b101	1:64	1:32
3'b110	1:128	1:64
3'b111	1:256	1:128

OSCCALレジスタ（OSCCAL）

- OSCCALレジスタは、ファイル・レジスタのアドレス05hにマップされ、工場出荷時の、OSCの校正値を設定する
- エミュレータでは、この機能は実装しない

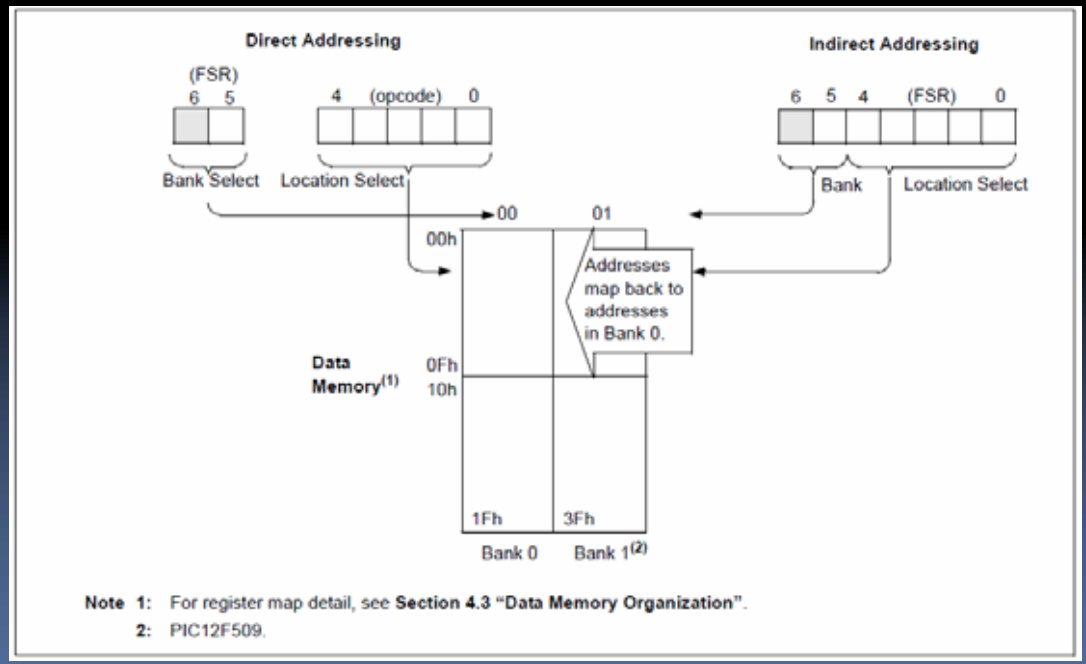
プログラム・カウンタ（PC）

- プログラム・カウンタは、12ビットのうち、実際に使われるのは下位の10ビット
- プログラム・カウンタの下位8ビットは、PCLレジスタとして、ファイル・レジスタのアドレス02hにマップされ、プログラム中から、参照や変更が可能
- 通常は、プログラム・カウンタは、1命令サイクル毎にインクリメントされる
- GOTO命令やCALL命令の場合は、命令のオペランドで指定された値により書き換えられ、プログラムのジャンプが実現される



INDFとFSRレジスタ

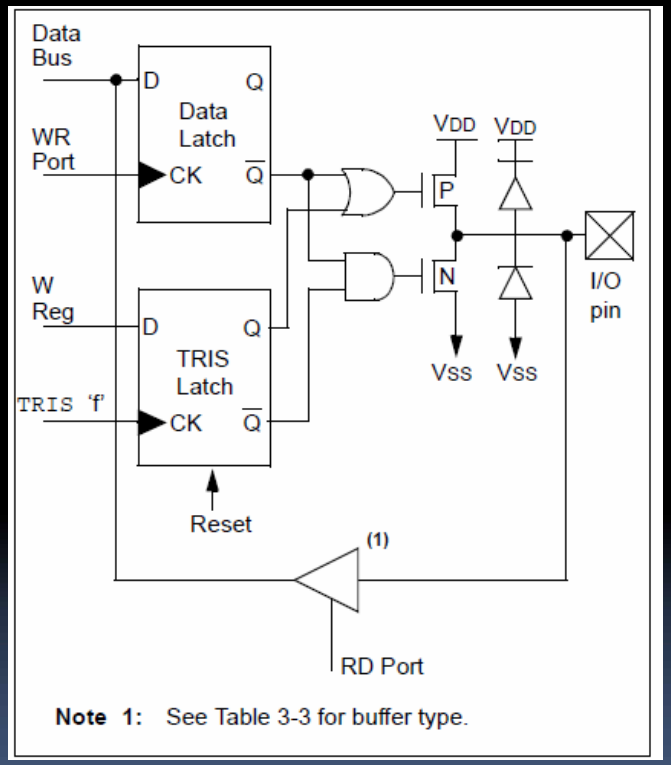
- 間接アドレスレジスタのINDFレジスタは、物理的なレジスタではなく、仮想的なレジスタとなっている
- INDFレジスタへのアクセスは、FSRレジスタで指定されたレジスタへのアクセスに置き換えらる



Note 1: For register map detail, see **Section 4.3 "Data Memory Organization"**.
2: PIC12F509.

I/Oポート

- PIC12F508は、GPIOレジスタを使って、外部のピンに出力を行ったり、外部のピンの状態を読み出すことができる
- GPIOは、GP0～GP5の6ビットが有効で、それぞれのビットは、対応するピンに接続されている
- GP3は入力専用で、それ以外のピンは、入出力が可能
- 各ピンの入出力方向の設定は、TRISレジスタで行う
- TRISレジスタの対応するビットが１ならば、GPIOの対応するビットは入力になり、０ならば出力となる

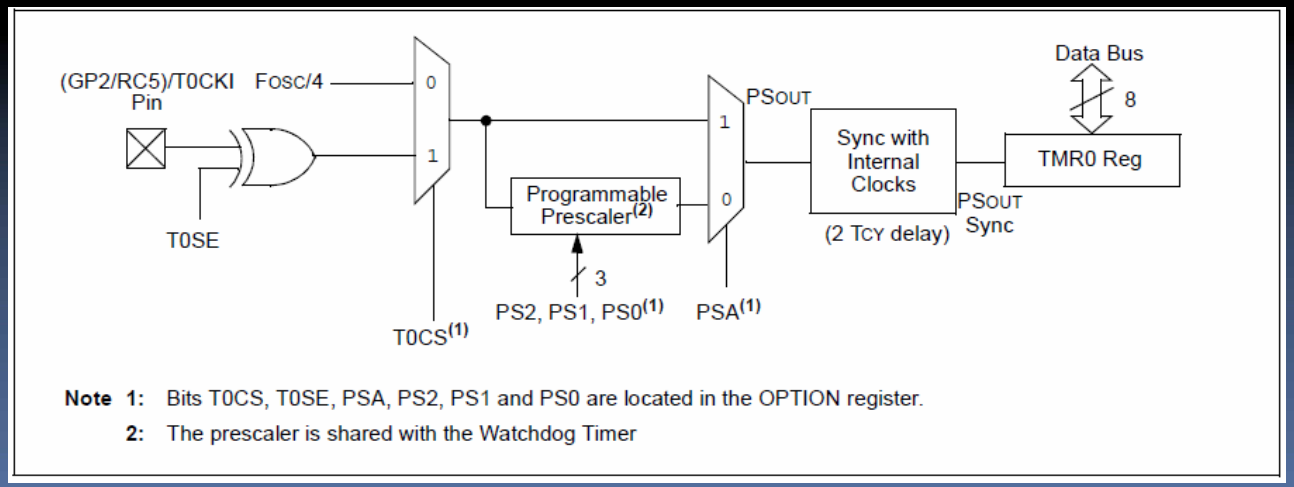


TRISレジスタ

- GPIOレジスタへの書き込みは、Data Latchに書き込まる
- ラッチの出力は、TRIS Latchのビットの設定により、出力をハイ・インピーダンスにするか、Data Latchの出力をそのままピンに出力するかどうかが決定される
- TRISレジスタは、ファイル・レジスタではないが、PIC12F508内部のレジスタとなっている
- TRISレジスタへの書き込みは、専用の命令を使って行う

タイマ０（TMR0）

- 8ビットのタイマ
- クロックソースは、システム・クロック（FOSC/4）または、T0CKI入力の何れか
- 8ビットのプリスケーラを使用可能
- タイマ０の値を読み出して、時間を計測するような使い方をする



Note 1: Bits T0CS, T0SE, PSA, PS2, PS1 and PS0 are located in the OPTION register.

2: The prescaler is shared with the Watchdog Timer

ALUとWレジスタ

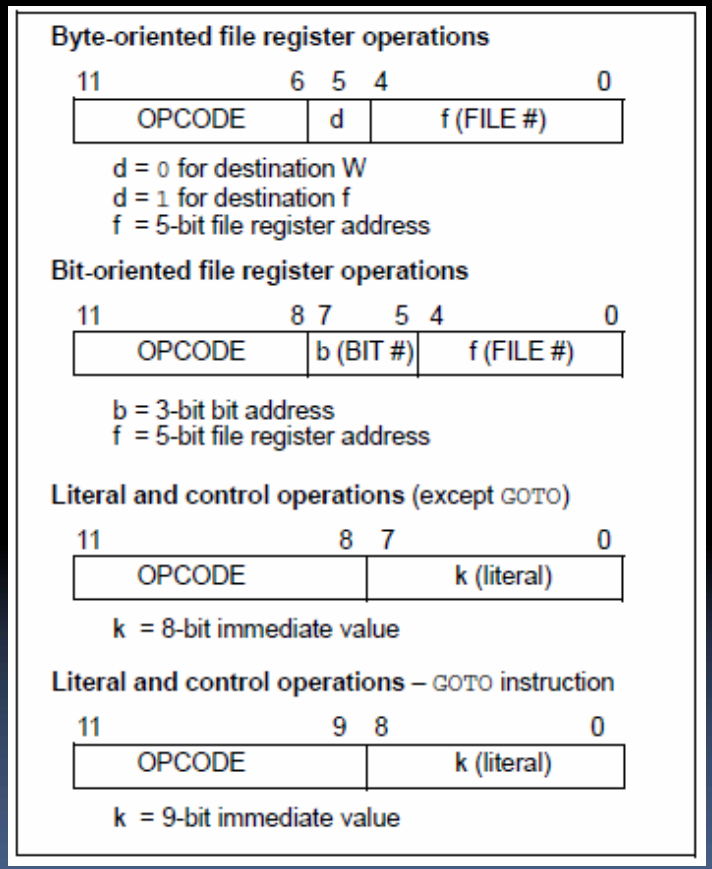
- PICの演算は、他のCPUと同様、ALUで行われる
- 演算には、Wレジスタが使用される
- Wレジスタは、ファイル・レジスタにはマッピングされていない
- 数値／論理演算は、ファイル・レジスタとWレジスタの間で行わる
- 演算命令は、結果をWレジスタまたはファイル・レジスタに書き戻すかを指定できる
- ２つのファイル・レジスタ間の演算を行う場合は、事前にWレジスタに、一方の値を読み込んでおく

コンフィグレーション・レジスタ

- コンフィグレーション・レジスタは、CPUの基本的な動作にかかわる設定を行う
- コンフィグレーション・レジスタは、CPUからアクセスできない
- 通常フラッシュ・メモリへ、プログラムを書き込む際、プログラマから、同時に設定する
- MCLREの機能、コードプロテクト、発振回路の選択などが行える
- エミュレータでは、このレジスタは実装しない

PIC12F508の命令フォーマット

- すべての命令コードは、オペランドを含め、12ビットに収まる
- 大きく分けて、4つの命令フォーマットがある
  1）ファイル・レジスタのアクセスを行う命令
  2）ビット演算命令
  3）即値データを扱う命令
  4）ジャンプ命令



エミュレータの制作

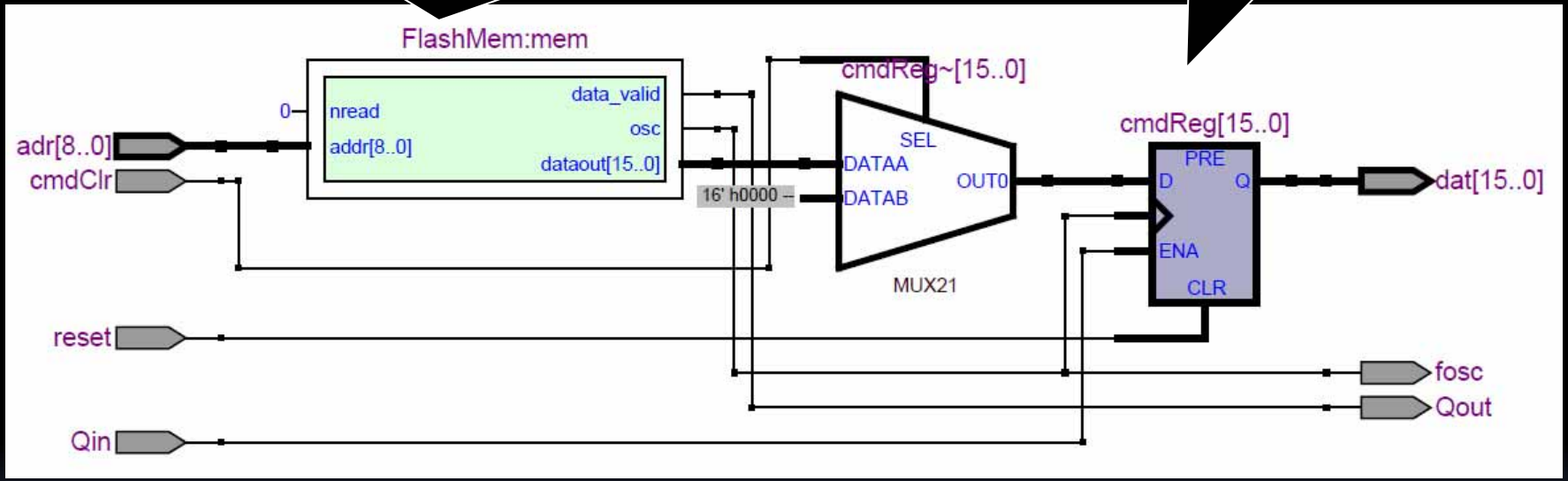
- HDLにはVerilogを使用する
- コンフィグレーション・レジスタは、存在しない
- プログラム・メモリの外部プログラム機能はない
- ウォッチドッグ・タイマは存在しない
- 発振回路は、内部クロックもしくは外部クロック（ソースコードで変更）
- タイマ0のクロックは、システムクロックのみ
- リセット回路は、内部リセットを利用
- スリープ機能はサポートしない
- 内蔵プルアップの機能はない
- OSCCALレジスタは存在しない
- リセット・ベクタは、最終アドレスではなく、000h（OSCCALが無いため）

フラッシュ・メモリ・ブロック

■ブロック図(1)

MEGA Functionで自動生成されるパラレル16bitのFlash メモリ (Read Only)

パイプライン処理用のキャッシュ



■ 入出力ポート

Signal	Direction	Note
reset	in	リセット入力
adr[8:0]	in	プログラム・メモリのアドレス入力
dat[15:0]	out	プログラム・メモリのデータ出力
fosc	out	内蔵の4MHzのオシレータ出力
Qout	out	命令サイクルの最後である事を示すQ信号出力
Qin	in	Q信号の入力
cmdClr	in	キャッシュを0でフラッシュする為の信号。ジャンプ命令等で使用する

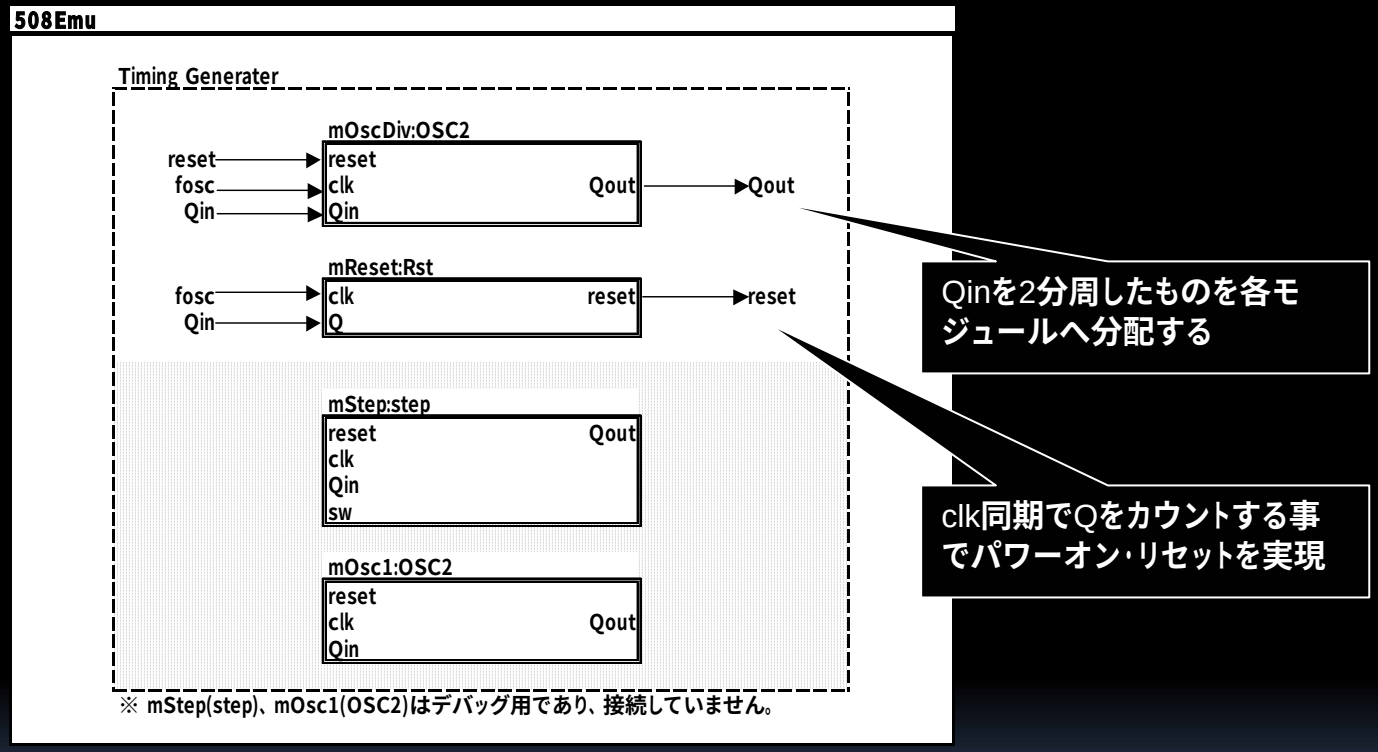
フラッシュ・メモリ・ブロック(2)

- PICのプログラムデータを、VerilogのROMとして構成する

```
module mFlashMem(adr,dat);
    input [8:0] adr;
    output [11:0] dat;
    reg [11:0] romdat;
    assign dat=romdat;
    always @(adr) begin
      case (adr)
        12'h00 : romdat=12'h0A12;
        12'h01 : romdat=12'h0C08;
        12'h02 : romdat=12'h0006;
        12'h03 : romdat=12'h0CC7;
        12'h04 : romdat=12'h0002;
        12'h05 : romdat=12'h0066;
        12'h06 : romdat=12'h0069;
        12'h07 : romdat=12'h0800;
        12'h08 : romdat=12'h0C0D;
        12'h09 : romdat=12'h0028;
        12'h0A : romdat=12'h0C01;
        12'h0B : romdat=12'h0021;
        12'h0C : romdat=12'h0201;
        12'h0D : romdat=12'h0743;
        12'h0E : romdat=12'h0A0C;
        12'h0F : romdat=12'h02E8;
        12'h10 : romdat=12'h0A0A;
        default : romdat=12'h0fff;
      endcase
    end
endmodule
```

タイミング・ジェネレータ・ブロック

■ ブロック図

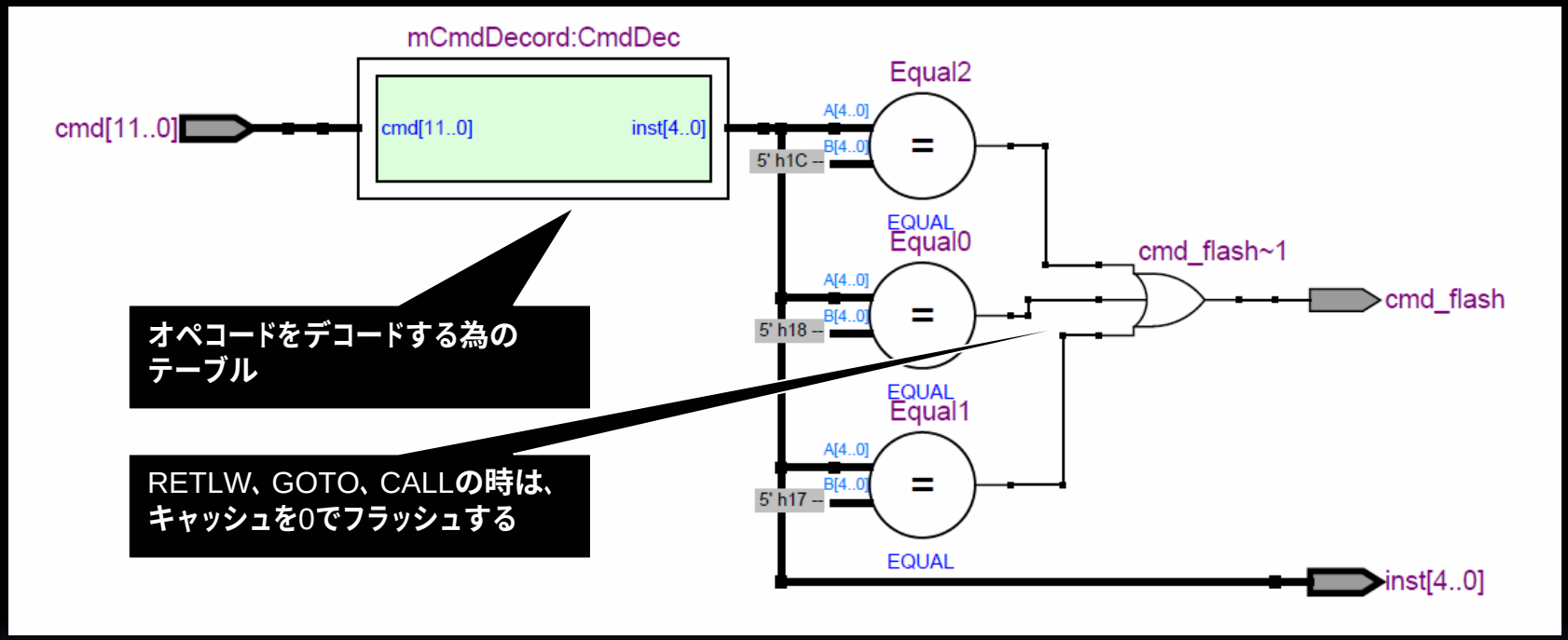


■ 入出力ポート

Signal	Direction	Note
reset	out	パワーオン･リセット
fosc	in	フラッシュ･メモリ･ブロックで生成されたシステムクロック
Qin	in	フラッシュ･メモリ･ブロックで生成されたQ信号
Qout	out	Qinを基に作られたQ信号。各モジュールで動作タイミングとして参照される

コマンド・プロセッサ

■ ブロック図

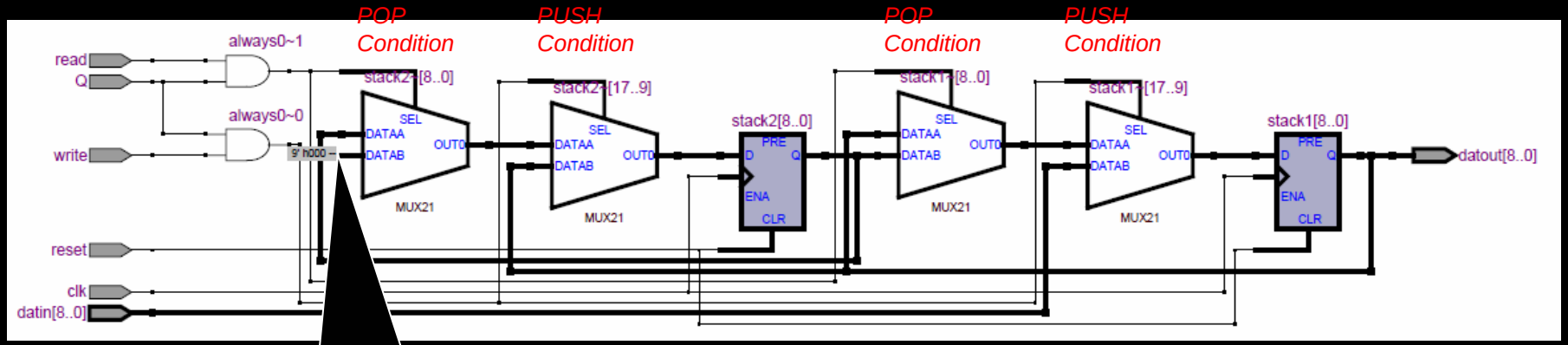


■ 入出力ポート

Signal	Direction	Note
cmd[11:0]	in	フラッシュ・メモリからロードされたプログラム
cmd_flash	out	フラッシュ・メモリ内のキャッシュを0でフラッシュすることを指示
inst[4:0]	out	インストラクション番号

スタック

■ ブロック図



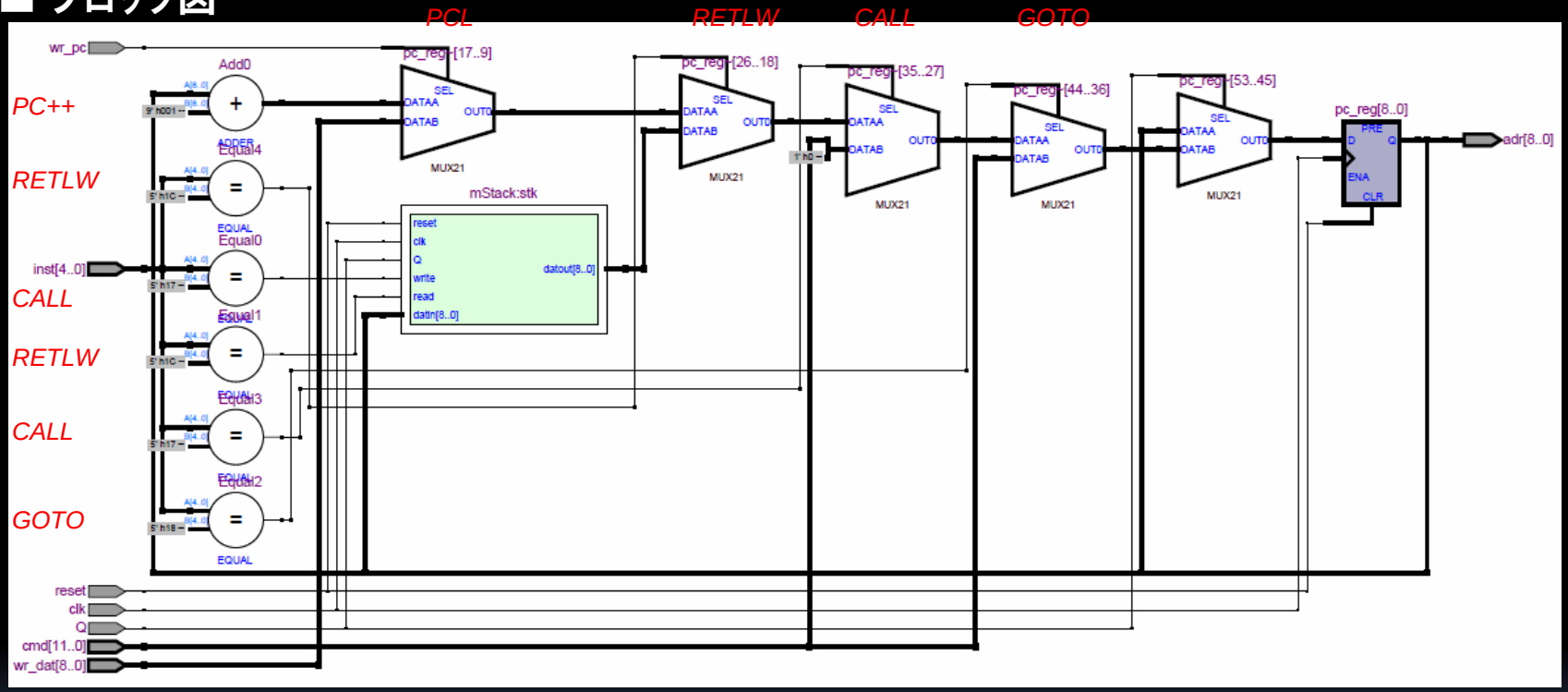
POP**動作時、**Stack2**は**9'h00**をラッチする**

■ 入出力ポート

Signal	Direction	Note
reset	in	内部パワーオン・リセット
clk	in	システム・クロック
Q	in	命令サイクルの最後である事を示すQ信号
write	in	PUSH動作を示すWrite信号
read	in	POP動作を示すRead信号
datin[8:0]	in	Stack1へPUSHされるプログラム・カウンタの値
datout[8:0]	out	Stack1からPOPされるプログラム・カウンタの値

プログラム・カウンタ

■ ブロック図

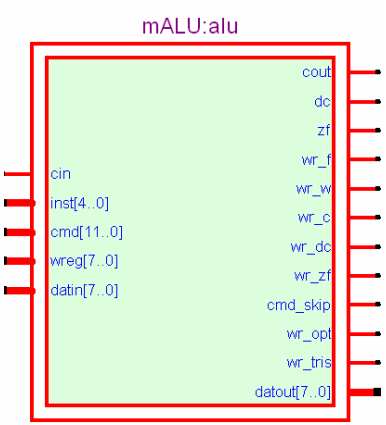


■ 入出力ポート

Signal	Direction	Note
reset	in	内部パワーオン･リセット
clk	in	システム･クロック
Q	in	命令サイクルの最後である事を示すQ信号
cmd[11:0]	in	フラッシュ･メモリからロードされたプログラム
inst[4:0]	in	プログラムからデコードされたインストラクション
wr_pc	in	PCLによってよって朝一とされるPC書き込みイネーブル信号
wr_dat[8:0]	in	PCLによってPC書き込みに使用するWREGの値
adr[8:0]	out	プログラム･カウンタの出力値

ALU（1）

■ ブロック図



■ 入出力ポート

Signal	Direction	Note
cin	in	ステータス・レジスタのCビット入力
inst[4:0]	in	デコードされたインストラクション入力
cmd[11:0]	in	フラッシュ・メモリからの命令データ入力
wreg[7:0]	in	Wレジスタの入力
datin[7:0]	in	データバスの入力
cout	out	ステータス・レジスタへCビットの出力
dc	out	ステータス・レジスタへDCビットの出力
zf	out	ステータス・レジスタへZビットの出力
wr_f	out	ファイル・レジスタの書き込み信号
wr_w	out	Wレジスタの書き込み信号
wr_c	out	ステータス・レジスタへCビットの書き込み信号
wr_dc	out	ステータス・レジスタへDCビットの書き込み信号
wr_zf	out	ステータス・レジスタへZビットの書き込み信号
cmd_skip	out	スキップ命令の出力信号
wr_opt	out	オプション・レジスタの書き込み信号
wr_tris	out	TRISレジスタの書き込み信号
datout[7:0]	out	データバスへの出力

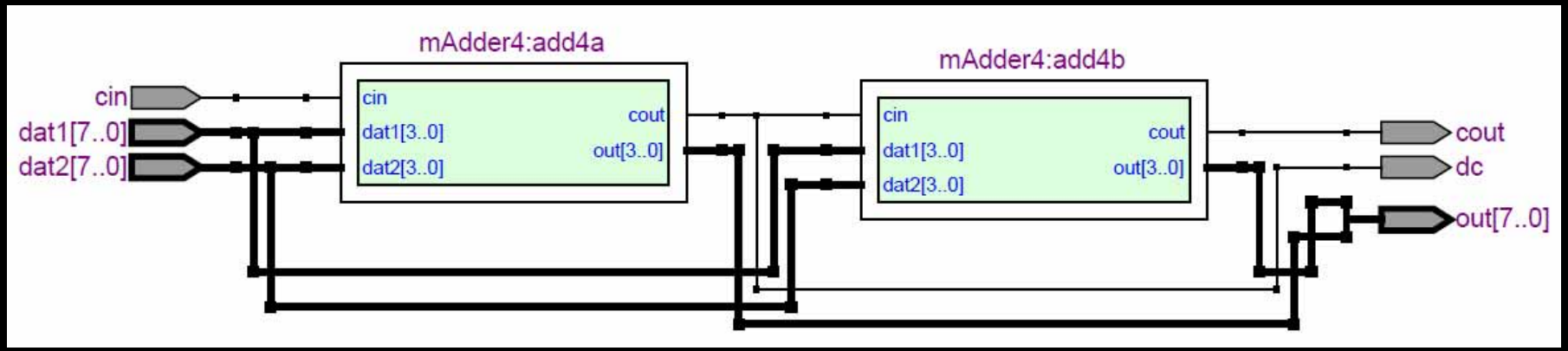
ALU（2）

■ 命令別のALUの処理内容

命令	機能	ALUの処理	フラグの処理	備考
ADDWF	WとFの加算	W+Fをデータバスに出力	C,DC,Zを書き込む	dビットにより、WまたはFに書きこむ
ANDWF	WとFの論理積	W and Fをデータバスに出力	Zを書き込む	dビットにより、WまたはFに書きこむ
CLRF	Fをクリア	0をデータバスに出力	Zを書き込む	Fに書き込む
CLRW	Wをクリア	0をデータバスに出力	Zを書き込む	Wに書き込む
COMF	Fのビット反転	~Fをデータバスに出力	Zを書き込む	dビットにより、WまたはFに書きこむ
DECF	Fをデクリメント	F-1をデータバスに出力	Zを書き込む	dビットにより、WまたはFに書きこむ
DECFSZ	Fをデクリメントしてゼロならスキップ	F-1をデータバスに出力		dビットにより、WまたはFに書きこむ
INCF	Fをインクリメント	F＋1をデータバスに出力	Zを書き込む	dビットにより、WまたはFに書きこむ
INCFSZ	Fをインクリメントしてゼロならスキップ	F＋1をデータバスに出力		dビットにより、WまたはFに書きこむ
IORWF	WとFの論理和	W or Fをデータバスに出力	Zを書き込む	dビットにより、WまたはFに書きこむ
MOVF	Fを移動	Fをデータバスに出力	Zを書き込む	dビットにより、WまたはFに書きこむ
MOVWF	WをFに移動	Wをデータバスに出力		Fに書き込む
NOP	何もしない			
RLF	左シフト	Fの左シフトをデータバスに出力	Cを書き込む	dビットにより、WまたはFに書きこむ
RRF	右シフト	Fno右をデータバスに出力	Cを書き込む	dビットにより、WまたはFに書きこむ
SUBWF	FからWを減算	F-Wをデータバスに出力	C,DC,Zを書き込む	dビットにより、WまたはFに書きこむ
SWAPF	Fの上下4ビットを入れ替える	Fの上下4ビットを入れ替えて、データバスに出力		dビットにより、WまたはFに書きこむ
XORWF	WとFの排他的論理和	WとFの排他的論理和をデータバスに出力	Zを書き込む	dビットにより、WまたはFに書きこむ
BCF	Fのビットをクリア	Fとビットマスクの反転の論理積をデータバスに出力		Fに書き込む
BSF	Fのビットをセット	Fとビットマスクの論理和をデータバスに出力		Fに書き込む
BTFSC	Fのビットをテストして、0ならスキップ	Fとビットマスクの論理積を計算		結果が0ならスキップコマンドを発行
BTFSS	Fのビットをテストして、1ならスキップ	Fとビットマスクの論理積を計算		結果が0以外ならスキップコマンドを発行
ANDLW	Wと数値の論理積	Wと命令の下位8ビットの論理積を出力	Zを書き込む	Wに書き込む
CALL	サブルーチンコール	なし		
CLRWDT	WDTをクリア	なし		
GOTO	無条件ジャンプ	なし		
IORLW	Wと数値の論理和	Wと命令の下位8ビットの論理和を出力	Zを書き込む	Wに書き込む
MOVLW	Wに数値を読み込む	命令の下位8ビットを出力		Wに書き込む
OPTION	Wをオプション・レジスタに書き込む	Wをデータバスに出力		オプション・レジスタに書き込む
RETLW	サブルーチンからの復帰	命令の下位8ビットを出力		Wに書き込む
SLEEP	スリープ	なし		
TRIS	WをTRISレジに書き込む	Wをデータバスに出力		TRISレジスタに書き込む
XORLW	Wと数値の排他的論理和	Wと命令の下位8ビットの排他的論理和を出力	Zを書き込む	Wに書き込む

ALU（3）

■ 加算器



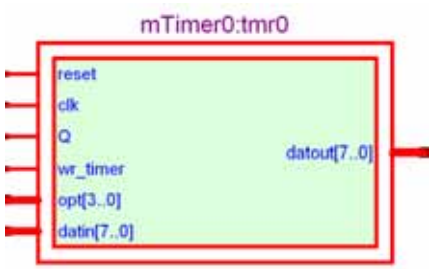
4bit加算器を2つ使用することで8bit加算器を実現します

■ 入出力ポート

Signal	Direction	Note
dat1[7:0]	in	加算対象のデータ1
dat2[7:0]	in	加算対象のデータ2
cin	in	加算対象のキャリー情報
out[7:0]	out	加算結果出力
dc	out	Digit Carry(桁上がり) 出力
cout	out	加算結果のキャリー出力

タイマ0

■ ブロック図

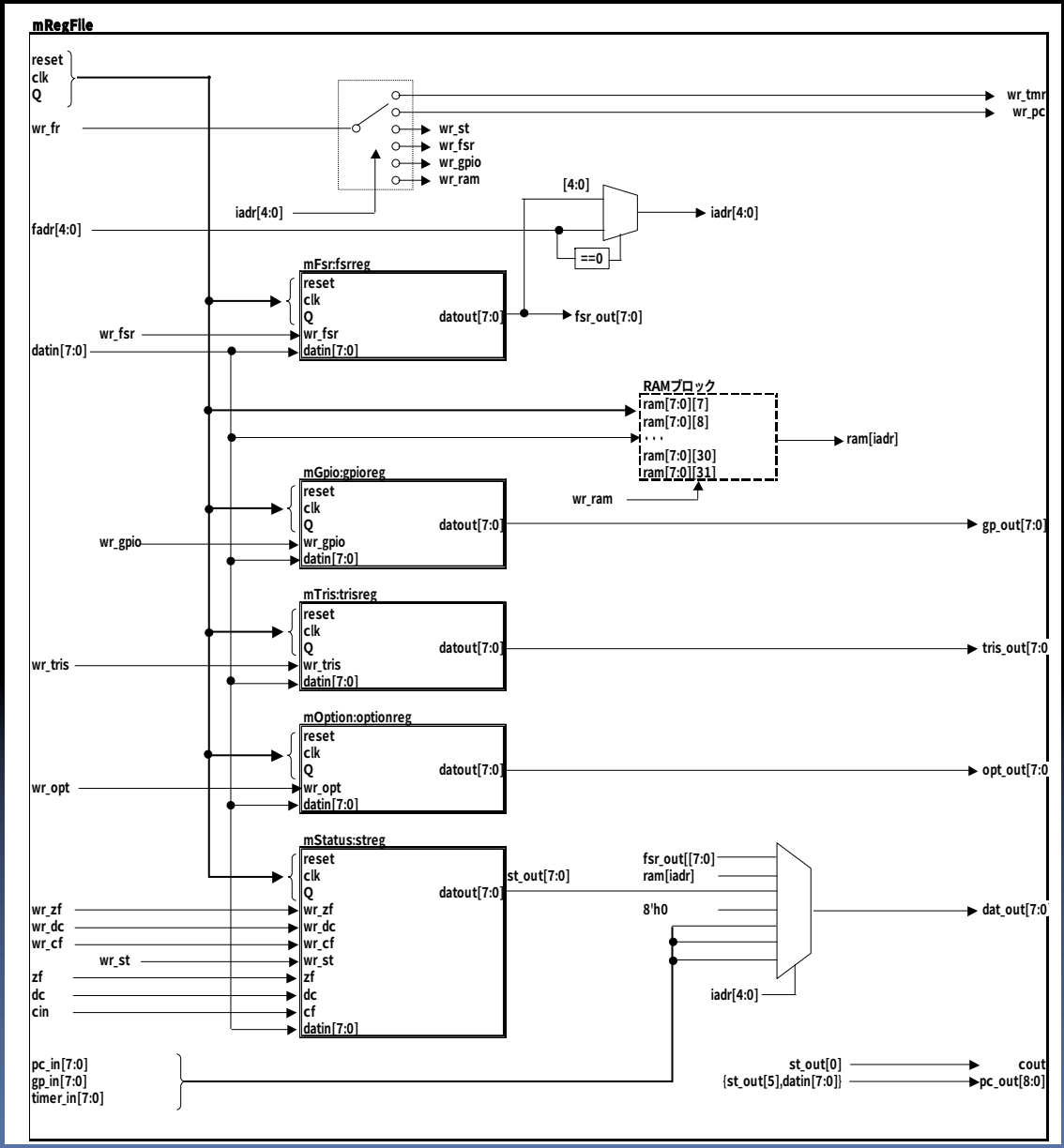


■ 入出力ポート

Signal	Direction	Note
reset	in	内部パワーオン･リセット
clk	in	システム･クロック
Q	in	命令サイクルの最後である事を示すQ信号
wr_timer	in	タイマの内部カウンタへの書き込み信号
opt[3:0]	in	OPTIONレジスタのプリスケーラ設定情報
datin[7:0]	in	タイマの内部カウンタへの設定値
datout[7:0]	out	タイマ0の現在のカウンタ値

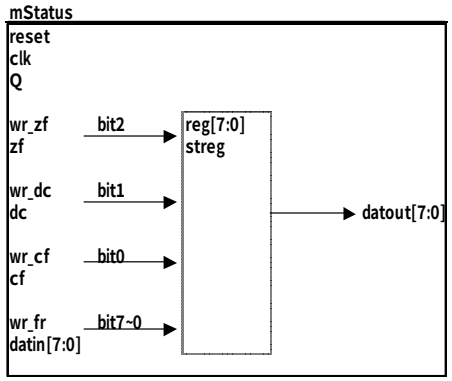
ファイル・レジスタ（1）

■ ブロック図

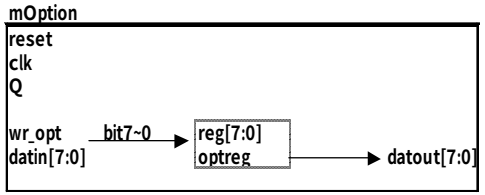


ファイル・レジスタ（2）

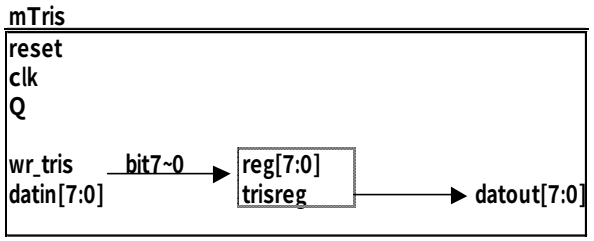
■ STATUSレジスタ



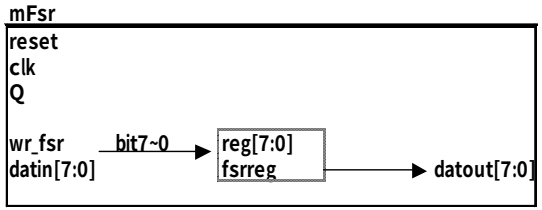
■ OPTIONレジスタ



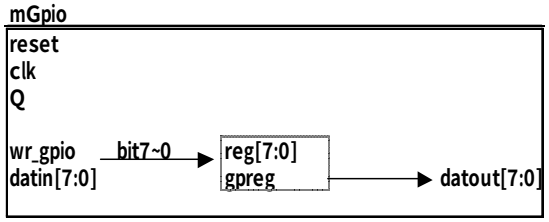
■ TRISレジスタ



■ FSRレジスタ



■ GPIOレジスタ



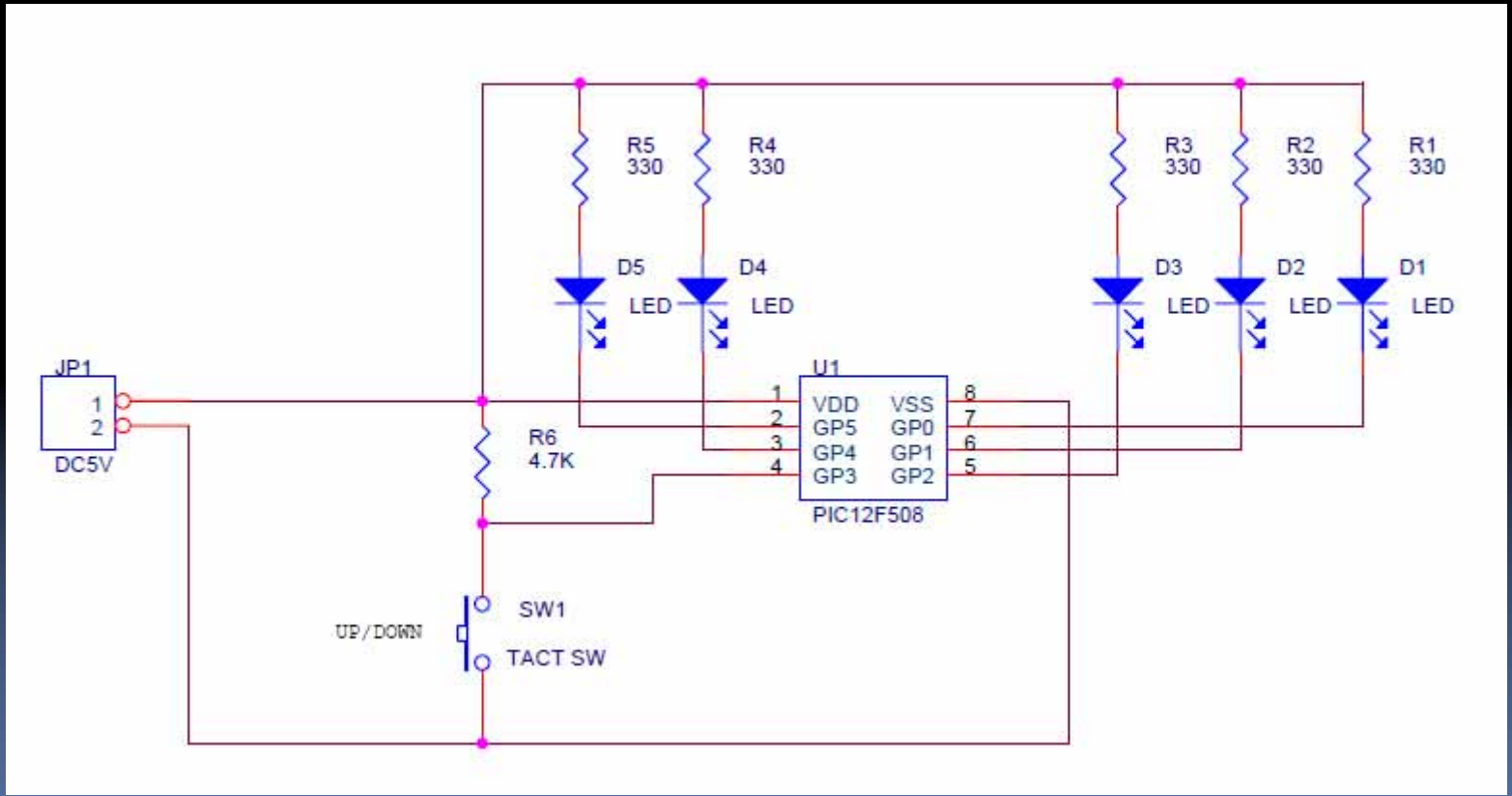
各ファイルレジスタの構造は同一です。

wr_**の書き込み信号がアサートされている時、datin[7:0]の値をラッチします。

ただし、STATUSレジスタについてはビットアクセスの仕組みが備わっています。

動作サンプル

- **図のような回路の、簡単なサンプルで動作を検証する**



テストプログラムソース

```
        list      p=12F508
        #include <p12F508.inc>
        __CONFIG   _MCLRE_OFF & _CP_OFF & _WDT_OFF & _IntRC_OSC
WORK            EQU                             0x08
LED             EQU                             0x09
        ORG       0x1FF             ; Reset Vector


;====== Program start ================================================
        ORG                         0x000             ; Start Address
        GOTO      start
;====== Initialize  ==================================================
init
        ;Initialize GPIO
        MOVLW     0x08              ; GP3は入力
        TRIS      GPIO
        MOVLW     0xC7              ;プリスケーラ=1/256
        OPTION
        MOVLW     0xFF
        MOVWF     GPIO
        RETLW     0
```

```
;====== wait_sec  ====================================================
wait_sec
                MOVLW           0x0d
                MOVWF           WORK
wait_1
                MOVLW           1
                MOVWF           TMR0
wait_2
                MOVF            TMR0,W
                BTFSS           STATUS,Z
                GOTO            wait_2
                DECFSZ          WORK,F
                GOTO            wait_1
                RETLW           0
;====== Main Program  ================================================
start
                CALL            init
main_1
                MOVLW           0x01
                MOVWF           LED           ;LEDの初期化
                BCF             STATUS,C      ;キャリーフラグをクリア
main_2
                BTFSC           GPIO,3
                ;SWのチェック
                GOTO            main_2      ;SWが押されていなければループ
                COMF            LED,W       ;LEDの値反転して表示
                MOVWF           GPIO
                CALL            wait_sec    ;1秒のウェイト
                BTFSC           LED,5       ;ビット5が１なら、再初期化
                GOTO            main_1
                RLF             LED,F       ;左に1ビットシフト
                GOTO            main_2
                END
```

フラッシュメモリモジュール

```
module mFlashMem(adr,dat);
   input [8:0] adr;
   output [11:0] dat;
   reg [11:0] romdat;
   assign dat=romdat;
   always @(adr) begin
      case (adr)
         12'h000 : romdat = 12'hA12 ;
         12'h001 : romdat = 12'hC08 ;
         12'h002 : romdat = 12'h006 ;
         12'h003 : romdat = 12'hCC7 ;
         12'h004 : romdat = 12'h002 ;
         12'h005 : romdat = 12'hCFF ;
         12'h006 : romdat = 12'h026 ;
         12'h007 : romdat = 12'h800 ;
         12'h008 : romdat = 12'hC0D ;
         12'h009 : romdat = 12'h028 ;
         12'h00A : romdat = 12'hC01 ;
         12'h00B : romdat = 12'h021 ;
         12'h00C : romdat = 12'h201 ;
         12'h00D : romdat = 12'h743 ;
         12'h00E : romdat = 12'hA0C ;
         12'h00F : romdat = 12'h2E8 ;
         12'h010 : romdat = 12'hA0A ;
         12'h011 : romdat = 12'h800 ;
         12'h012 : romdat = 12'h901 ;
         12'h013 : romdat = 12'hC01 ;
         12'h014 : romdat = 12'h029 ;
         12'h015 : romdat = 12'h403 ;
         12'h016 : romdat = 12'h666 ;
         12'h017 : romdat = 12'hA16 ;
         12'h018 : romdat = 12'h249 ;
         12'h019 : romdat = 12'h026 ;
         12'h01A : romdat = 12'h908 ;
         12'h01B : romdat = 12'h6A9 ;
         12'h01C : romdat = 12'hA13 ;
         12'h01D : romdat = 12'h369 ;
         12'h01E : romdat = 12'hA16 ;
         default : romdat = 12'hfff ;
      endcase
   end
endmodule
```

参考・引用文献ほか

PIC12F508/509/16F505 Data Sheet
Microchip
http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en020094
日本アルテラ
http://www.altera.co.jp/index.html
MAX II
http://www.altera.co.jp/products/devices/cpld/max2/mx2-index.jsp

■MAX II **マイクロキット**
株式会社ソリトンウェーブ
http://www.solitonwave.co.jp
MAX II **マイクロキット**
http://www.solitonwave.co.jp/products/max2kit.html