

VI. Architecture de cœur de calculateur

UT (UAL), UC, Umem, Ucomm



ENSGTI
ÉCOLE D'INGÉNIEURS

Arnaud Martin

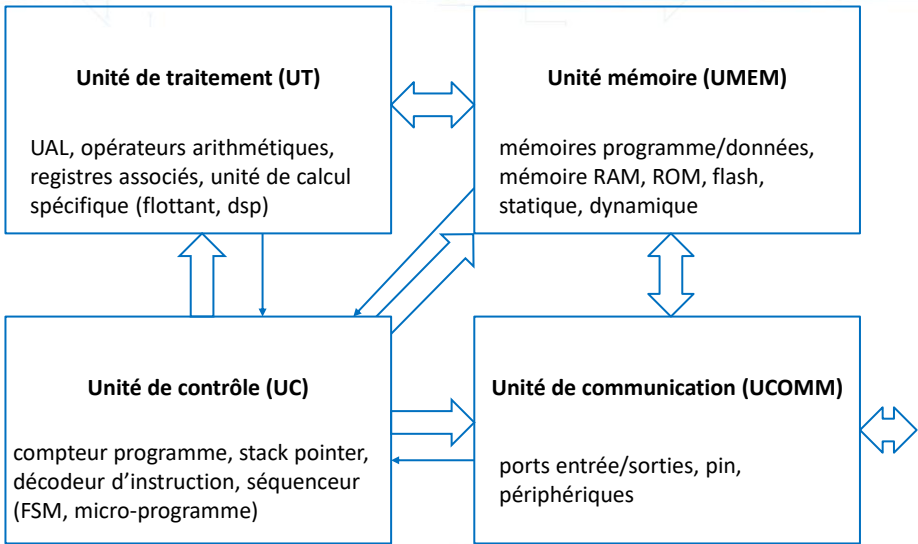
1

1



VI. Architecture
de cœur de
calculateur

Architecture de cœur de calculateur



2

2

Vue générale

Compteur programme

Multiplexeur

Séquenceur (FSM)

Multiplieur

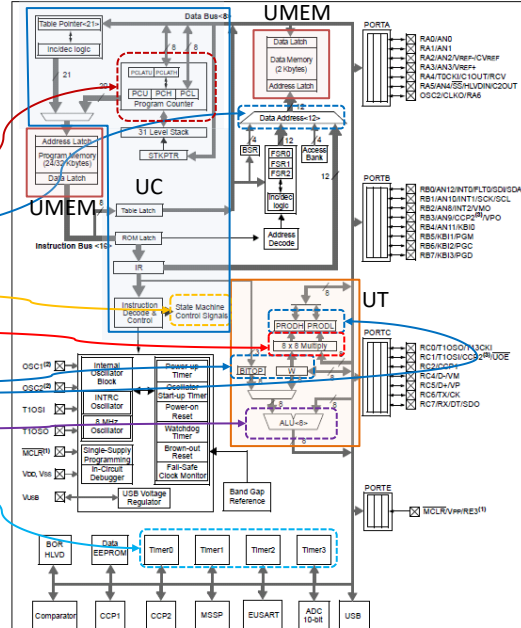
Registres

ALU

Compteurs

PIC18F2455/2550/4455/4550

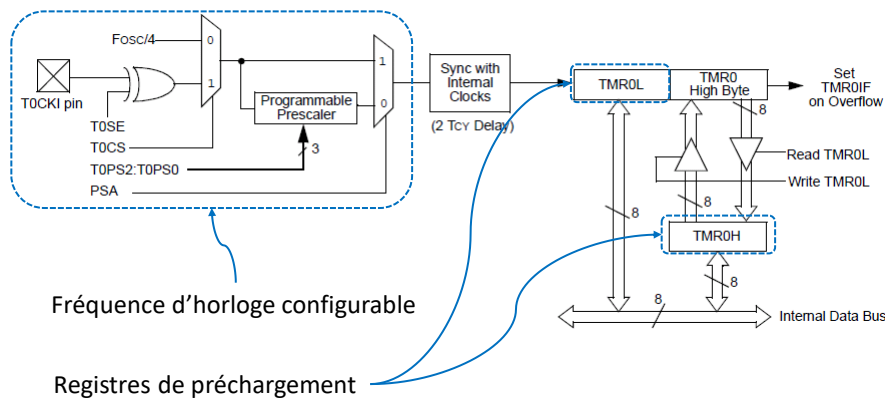
FIGURE 1-1: PIC18F2455/2550 (28-PIN) BLOCK DIAGRAM



3

3

Timer 0



Fréquence d'horloge configurable

Registres de préchargement

1

4

Multiplication

PIC18F2455/2550/4455/4550

8.0 8 x 8 HARDWARE MULTIPLIER

8.1 Introduction

All PIC18 devices include an 8 x 8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the Status register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 8-1.

8.2 Operation

Example 8-1 shows the instruction sequence for an 8 x 8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 8-2 shows the sequence to do an 8 x 8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 8-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

```
MOVW    ARG1, W      ;
MULWF   ARG2          ; ARG1 * ARG2 ->
                     ; PRODH:PRODL
```

EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE

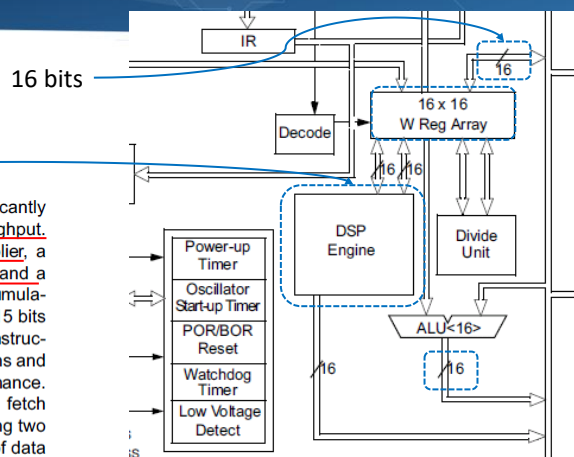
```
MOVWF ARG1, W
MULWF ARG2          ; ARG1 * ARG2 ->
                     ; PRODH:PRODL
BTFSRC ARG2, SB      ; Test Sign Bit
SUBWF PRODH, F       ; PRODH = PRODH
                     ; - ARG1
MOVWF ARG2, W
BTFSRC ARG1, SB      ; Test Sign Bit
SUBWF PRODH, F       ; PRODH = PRODH
                     ; - ARG2
```

TABLE 8-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 6.9 μs	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 μs	27.6 μs	69 μs
	Hardware multiply	1	1	100 ns	400 ns	1 μs
8 x 8 signed	Without hardware multiply	33	91	9.1 μs	36.4 μs	91 μs
	Hardware multiply	6	6	600 ns	2.4 μs	6 μs
16 x 16 unsigned	Without hardware multiply	21	242	24.2 μs	96.8 μs	242 μs
	Hardware multiply	28	28	2.8 μs	11.2 μs	28 μs
16 x 16 signed	Without hardware multiply	52	254	25.4 μs	102.6 μs	254 μs
	Hardware multiply	35	40	4.0 μs	16.0 μs	40 μs

Cœur

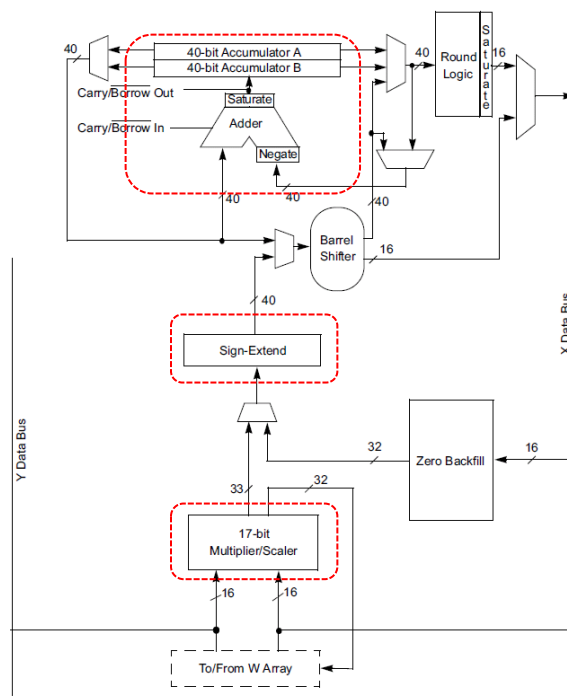
A DSP engine has been included to significantly enhance the core arithmetic capability and throughput. It features a high speed 17-bit by 17-bit multiplier, a 40-bit ALU, two 40-bit saturating accumulators and a 40-bit bidirectional barrel shifter. Data in the accumulator or any working register can be shifted up to 15 bits right, or 16 bits left in a single cycle. The DSP instructions operate seamlessly with all other instructions and have been designed for optimal real-time performance. The MAC class of instructions can concurrently fetch two data operands from memory while multiplying two W registers. To enable this concurrent fetching of data operands, the data space has been split for these instructions and linear for all others. This has been achieved in a transparent and flexible manner, by dedicating certain working registers to each address space for the MAC class of instructions.



Architecture dsPIC30

VI. Architecture de cœur de calculateur

DSP engine



7

7

Architecture dsPIC30

VI. Architecture de cœur de calculateur

Multiplication

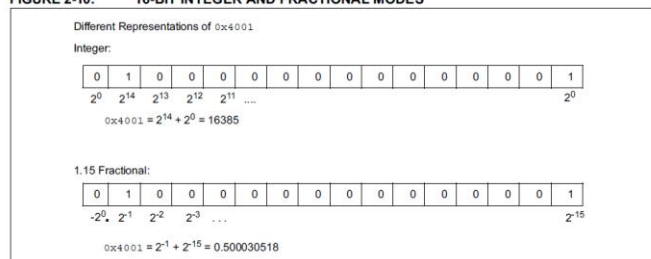
dsPIC30F

2.5.1 MULTIPLIER

The 17 x 17-bit multiplier is capable of signed or unsigned operation and can multiply its output using a scaler to support either 1.31 fractional (Q31) or 32-bit integer results. The respective number representation formats are shown in Figure 2-10. Unsigned operands are zero-extended into the 17th bit of the multiplier input value. Signed operands are sign-extended into the 17th bit of the multiplier input value. The output of the 17 x 17-bit multiplier/scaler is a 33-bit value which is sign-extended to 40 bits. Integer data is inherently represented as a signed two's complement value, where the MSB is defined as a sign bit. Generally speaking, the range of an N-bit two's complement integer is -2^{N-1} to $2^{N-1} - 1$. For a 16-bit integer, the data range is -32768 (0x8000) to 32767 (0x7FFF) including '0' (see Figure 2-10). For a 32-bit integer, the data range is -2,147,483,648 (0x8000 0000) to 2,147,483,645 (0x7FFF FFFF).

When the multiplier is configured for fractional multiplication, the data is represented as a two's complement fraction, where the MSB is defined as a sign bit and the radix point is implied to lie just after the sign bit (QX format). The range of an N-bit two's complement fraction with this implied radix point is -1.0 to $(1 - 2^{1-N})$. For a 16-bit fraction, the Q15 data range is -1.0 (0x8000) to 0.999969482 (0x7FFF) including '0' and has a precision of 3.01518×10^{-5} . In Fractional mode, the 16x16 multiply operation generates a 1.31 product which has a precision of 4.65661×10^{-10} .

FIGURE 2-10: 16-BIT INTEGER AND FRACTIONAL MODES



8

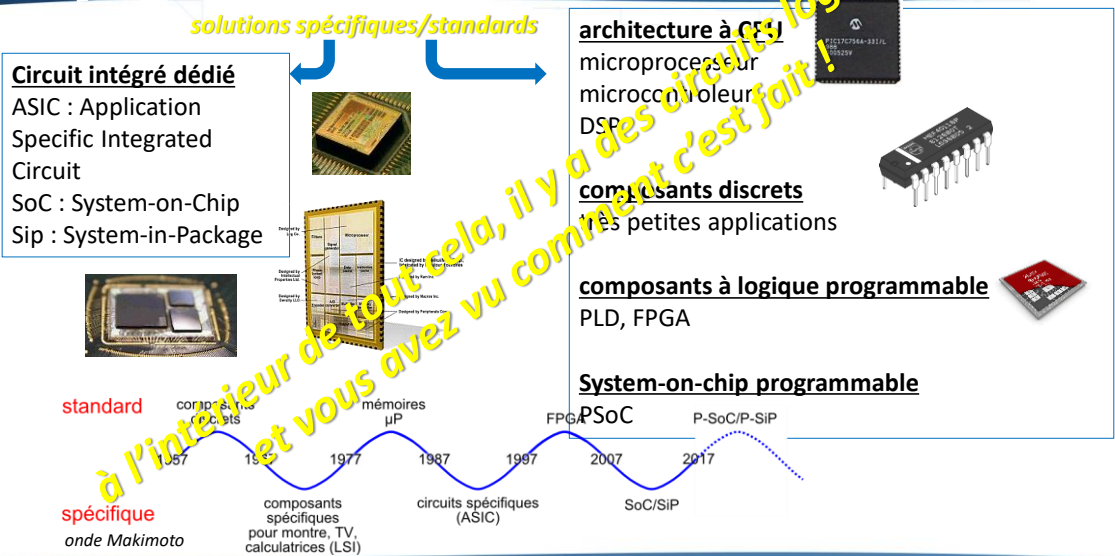
8

Multiplication

Assembly Syntax	Description	# of Words	# of Cycles	Status Flags Affected
MPY Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wm by Wn to Accumulator	1	1	OA,OB,OAB,SA,SB,SAB
MPY Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square Wm to Accumulator	1	1	OA,OB,OAB,SA,SB,SAB
MPY.N Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wm by Wn) to Accumulator	1	1	None
MSC Wm*Wm,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and Subtract from Accumulator	1	1	OA,OB,OAB,SA,SB,SAB
MUL.SS Wb,Ws,Wnd	{Wnd+1, Wnd} = signed(Wb) * signed(Ws)	1	1	None
MUL.SU Wb,Ws,Wnd	{Wnd+1, Wnd} = signed(Wb) * unsigned(Ws)	1	1	None
MUL.US Wb,Ws,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * signed(Ws)	1	1	None
MUL.UU Wb,Ws,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * unsigned(Ws)	1	1	None
MUL.SU Wb,#lit5,Wnd	{Wnd+1, Wnd} = signed(Wb) * unsigned(lit5)	1	1	None
MUL.UU Wb,#lit5,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * unsigned(lit5)	1	1	None
MUL f	W3:W2 = f * WREG	1	1	None

9

9



10

10