

Microprocessor based Digital Systems

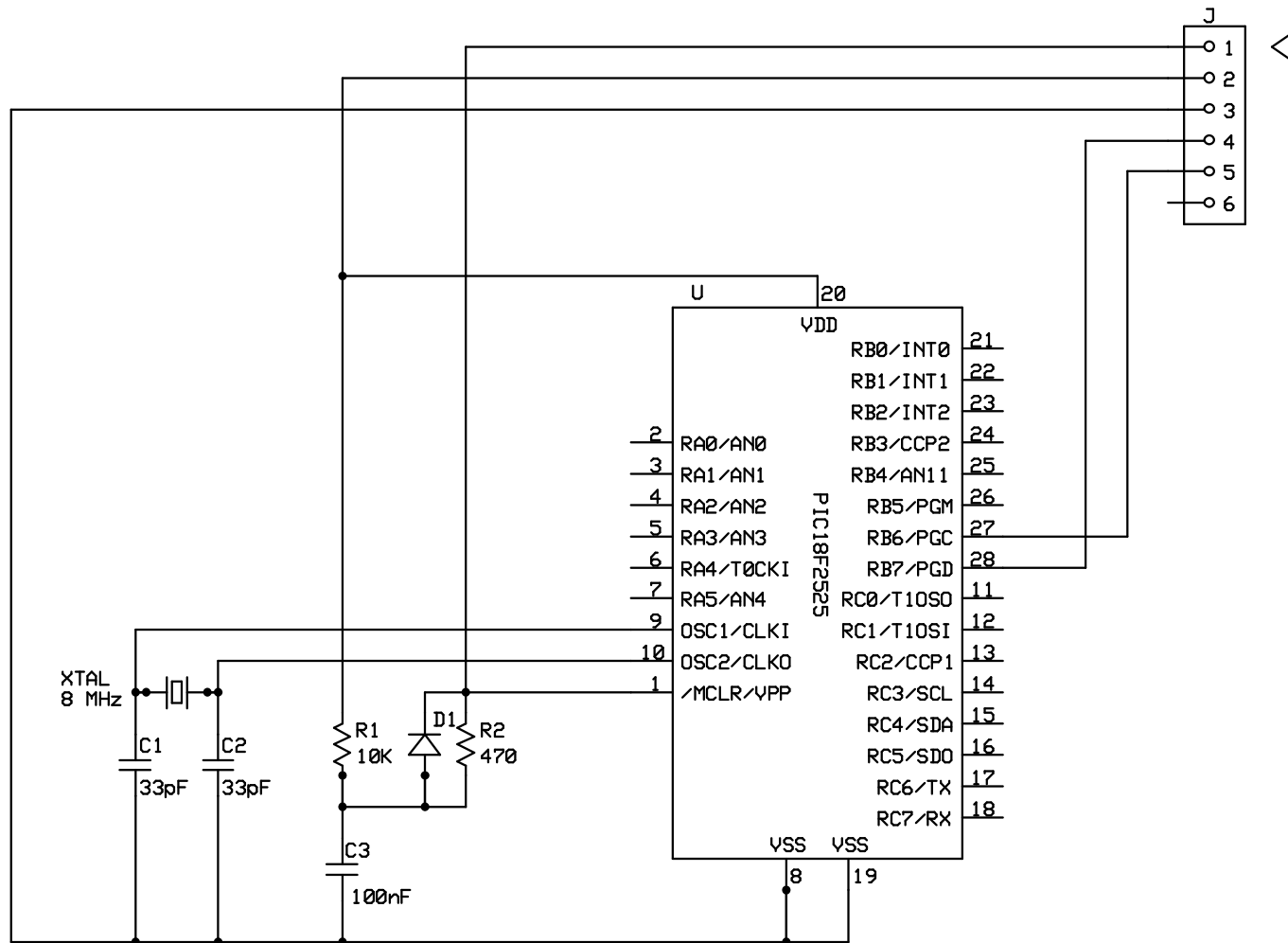
Starters Kit

Guillermo Carpintero

Universidad **Carlos III** de Madrid

Basic Hardware

1

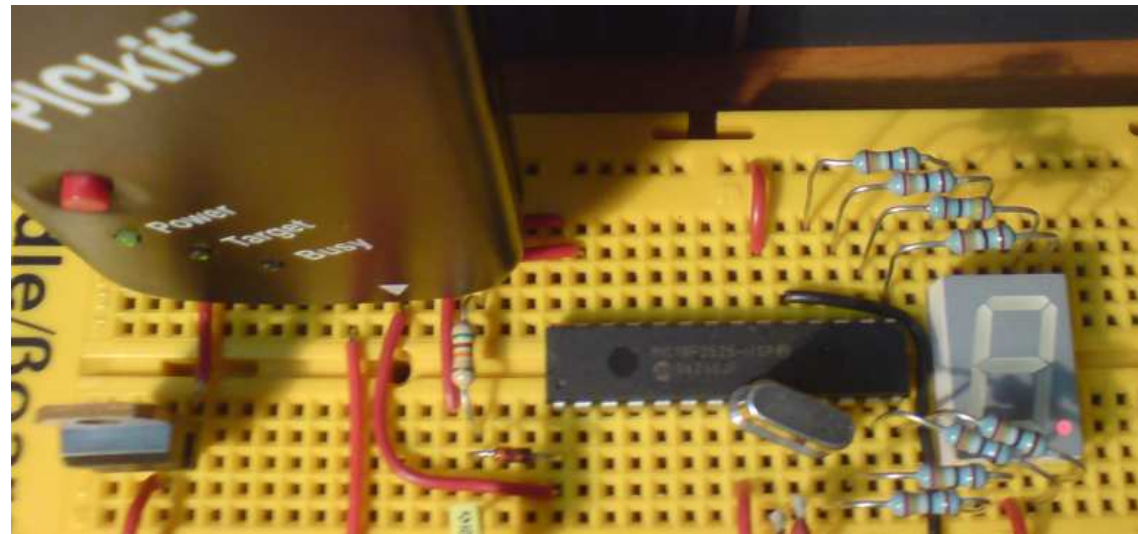


"Vital Elements":

| | |
|------------|-------------|
| Supply | VDD |
| Ground | VSS |
| Oscillator | OSC1 y OSC2 |
| Reset | /MCLR |

Basic Hardware

PICKit 2 Microcontroller Programmer



Integrated Development Environment

2

MPLAB

Write

Assemble

Simulate

Program

Debug

The screenshot displays the MPLAB IDE interface with several windows open:

- Source Editor:** Shows C code for a timer. The code includes comments and function definitions like `void main()` and `for` loops.
- Disassembly Listing:** Shows the assembly code corresponding to the source code, with addresses and instructions like `MOVWF 0x55`.
- Special Function Registers:** A table showing register names, hex values, and binary representations.
- Stopwatch:** A dialog box showing simulation statistics: Instruction Cycles (21511894), Total Simulated (21511894), Time (4.302379), and Processor Frequency (20,000,000 MHz).
- Watch:** A table monitoring variables: `index` (0x007B), `ADCON0` (0x00), `PORTB` (0x00), and `TOS` (0x000126).
- Hardware Stack:** A table showing the stack frame: `TOS`, `Stack Level`, `Return Address`, and `Location`.

| SFR Name | Hex | Binary |
|----------|-----|----------|
| INDF1 | -- | ----- |
| INDF2 | -- | ----- |
| INTCON | 80 | 10100000 |
| INTCON2 | 84 | 10000100 |
| INTCON3 | C0 | 11000000 |
| IPR1 | FF | 11111111 |
| IPR2 | 1F | 00011111 |
| LATA | 00 | 00000000 |
| LATB | 00 | 00000000 |

| Address | Symbol Name | Value |
|---------|-------------|----------|
| 007B | index | 0x007B |
| 07C0 | ADCON0 | 0x00 |
| 07B1 | PORTB | 0x00 |
| 07FD | TOS | 0x000126 |

| TOS | Stack Level | Return Address | Location |
|-----|-------------|----------------|-------------|
| | 0 | Empty | |
| | 1 | 000044 | _startup + |
| | 2 | 000126 | main + 0x4D |
| | 3 | 000198 | .file |
| | 4 | 000000 | |
| | 5 | 000000 | |
| | 6 | 000000 | |
| | 7 | 000000 | |
| | 8 | 000000 | |

Integrated Development Environment

With support (and free of charge if possible)



[Introduction to MPLAB IDE](#)

03/30/2004

[Tips and Tricks Using MPLAB v6.61](#)

09/16/2004

[Introduction to Microchip's Development Tools](#)

02/17/2004

[Choosing a Debug Tool](#)

02/24/2006

[Introduction to the MPLAB Visual Device Initializer \(VDI\)](#)

08/26/2004

Integrated Development Environment

Example (I)

The screenshot displays the MPLAB IDE v7.60 interface. The main window shows the assembly source file `C:\guiller\WPASMP1\fuente1.asm`. The code includes two interrupt routines and a main loop:

```
ORG 0x0008
retfie ;go to high priority interrupt routine

ORG 0x0018
retfie ;go to low priority interrupt routine

Main:

;vamos a utilizar el puerto A
MOVE  ADCON1,W,A ; Configurar Puerto A para E/S
IORLW 0Fh
MOVWF ADCON1

CLRF  TRISA ; TRISA para que PORTA, todo output

Loop:
btg  PORTA,BITS,A
goto Loop

END
```

A Logic Analyzer window is overlaid on the code, showing a digital signal trace for RA5. The trace displays a square wave with a period of approximately 10 units. A trigger point is set at PC 22, marked with a red vertical line and the number 3. The x-axis represents time from 0.0 to 60.0.

The status bar at the bottom indicates the target device is PIC18F2525, the processor is pc:0x22, and the current instruction is `W:0xf`. The taskbar shows the system is running on Windows with the time 21:36.

Integrated Development Environment

Example (II)

The screenshot displays the MPLAB IDE v7.60 interface. The main window shows the assembly source file `C:\guiller\MPASM\P1\Fuente1.asm`. The code includes comments in Spanish and assembly instructions for configuring and polling Port A.

```
----- Necesario para ver los cambios en el Puerto A -----
Main:
    MOVF   ADCON1,W,A      ; Configurar Puerto A para E/S
    IORLW  0Fh
    MOVWF  ADCON1

    CLRF   TRISA           ; TRISA para que PORTA, todo outputs
;-----
Loop:
    BTG   PORTA,BIT5,A

;----- Para aumentar el periodo -----
    MOVLW 10
    MOVWF cntr0,A
del_loop:
    DECF  cntr0,F,A
    GOTO del_loop
;-----
    GOTO Loop
END
```

The Watch window shows the following data:

| Address | Symbol Name | Value |
|---------|-------------|-------|
| F08 | WREG | 0x10 |
| 000 | cntr0 | 0x10 |

The Logic Analyzer window shows a digital signal trace for RA5. The signal is high from 0.0 to approximately 50.0, then drops to low until about 100.0, then returns to high. A trigger point is marked at approximately 130.0 with a value of 53.

The status bar at the bottom indicates the simulation is running on a PIC18F2525 with PC:0x28, W:0x10, and bank 0. The Windows taskbar shows the Start button and several open applications including MPLAB IDE v7.60, Segundo, Microsoft PowerPoint..., Libertad Digital: Aguir..., and Fantastico_g - Bloc d... The system clock shows 22:26.

Flux Diagrams

4

Help to organize the program before writing it

Allow to organize the program flow (sequence of instructions).

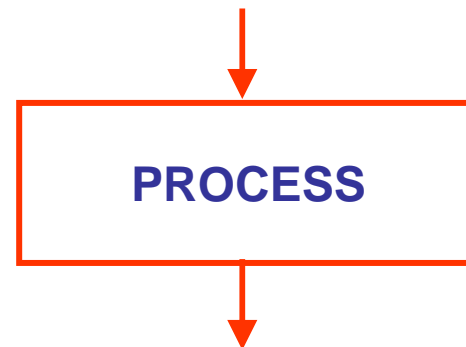
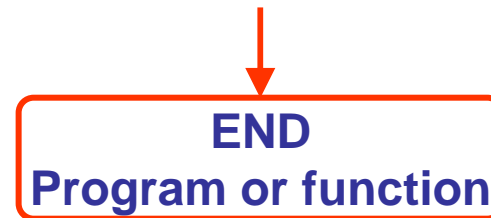
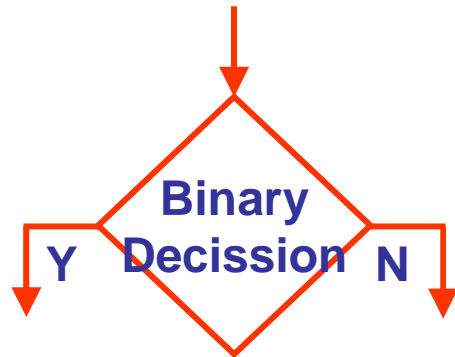
It is a tool to:

design
document a program

Use standard symbols that are connected between arrows to indicate the course of the program execution (sequential execution principle) of the program instructions.

Flux Diagrams

Symbols



Programming Techniques

5

¿Why use programming techniques?

It is very difficult to write **good** code.

¿What defines a **good** code?:

- Program memory usage
- Execution time
- Readable, and reusable
- Delivered on time

Techniques:

~~Trial and Error~~

Structured Programming

Programming Techniques

Structured Programming

Flux diagram allow an infinite ways of being connected.

Structured programming is a discipline which basically limits the possible flux diagram combinations that we can use.

Allowed structures:

SEQUENCE

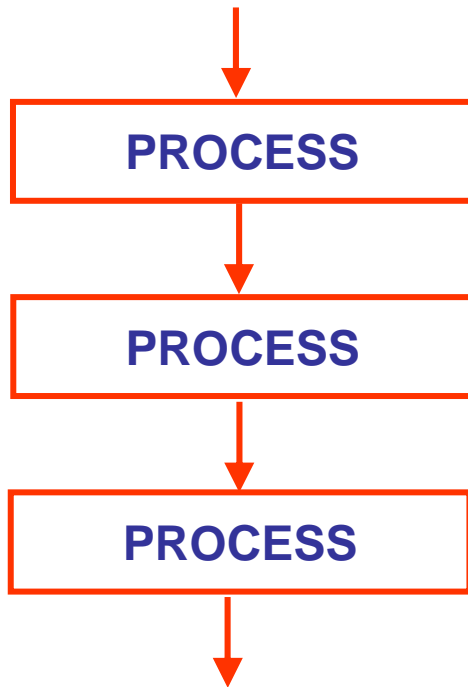
IF-THEN-ELSE

DO-WHILE

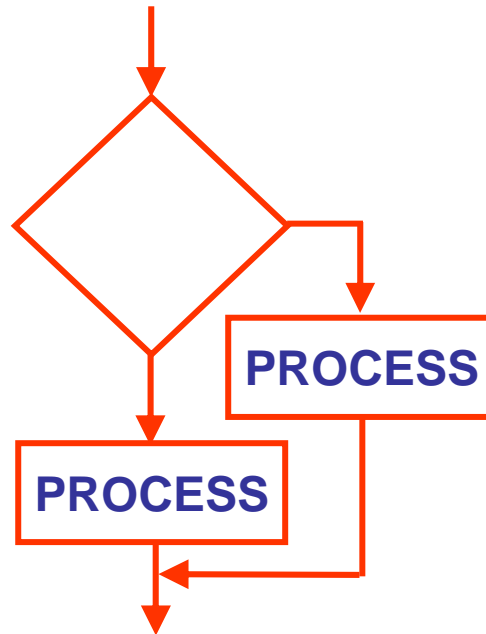


Programming Techniques

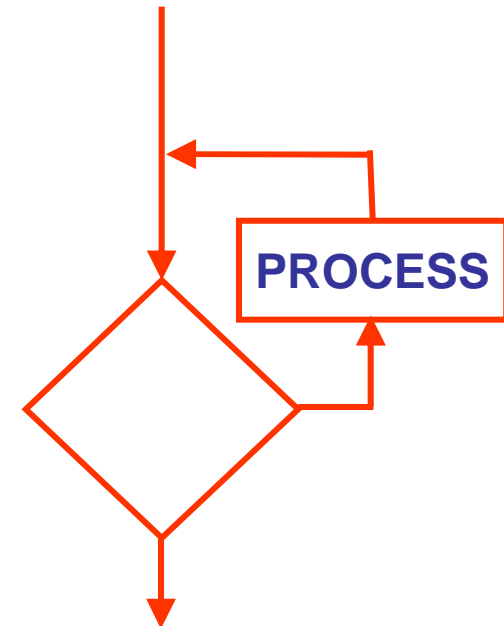
SEQUENCE



IF-THEN-ELSE

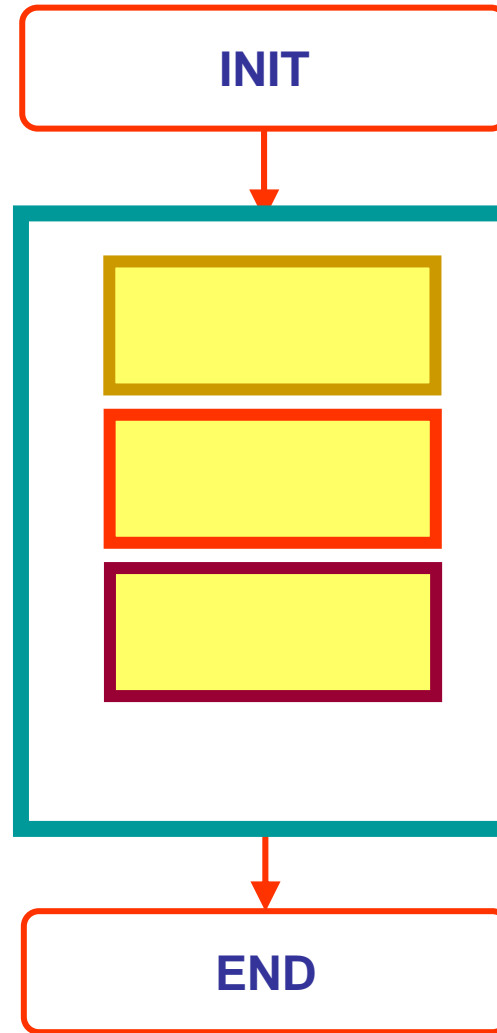


DO-WHILE



What do all these structures have in common?

Top-Down Design



Divide and Conquer

Construct your program as a sequence of **functions**