Universidad Nacional de Lanús

Departamento de desarrollo productivo y tecnológico

Materia: Arquitectura de computadoras
Docente a cargo: Lic. Roberto Garcia
JTP: Lic. Miguel Lanzeni

Comisión: Viernes
Año cursada 2023 – 2º Cuatrimestre

Alumno: Santiago Leon Dal Degan
DNI: 45.421.137

UNIVERSIDAD NACIONAL DE LANUS
LICENCIATURA EN SISTEMAS
Arquitectura de Computadoras
Prof. Tit.: Lic. Roberto García
Inst. Ay.: Lic. Miguel Lanzeni

PROGRAMACION EN ASSEMBLER

GUIA DE TRABAJOS PRACTICOS

Ejercicio 1

Se trata de, utilizando un microcontrolador PIC 16F628A, controlar la temperatura de un termotanque. La idea se base en la utilización de registros de funciones generales para la temperatura máxima TM, para la temperatura mínima Tm, para la temperatura del agua Ta y los que sean necesarios para lograr un modelo que pueda ser codificado en assembler y correrlo en mplab.Las características del termotanque y lo que se debe controlar son las siguientes:

El termotanque tiene una capacidad de 110litros.

La temperatura mínima de trabajo es de 20°C y la temperatura máxima donde debe dejar de calentar es de 45°C.

Piénsese de la siguiente forma:

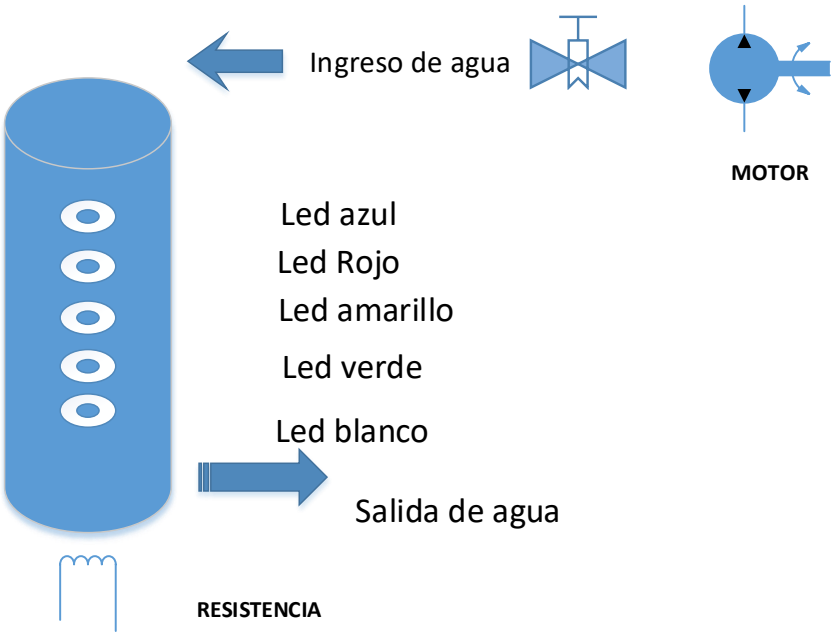
El termo esta desenchufado, cuando se da energía debe testearse si tiene agua o si la cantidad es menor de 110lts. Si es menor de 110 lts se acciona la bomba (un led debe indicar que la bomba esta prendida. Cuando llega a la cantidad de agua se verifica la temperatura del agua (un led debe indicar ese estado) y si esta debajo de la temp máxima se enciende la resistencia.(un led encendiendo y apagando en forma intermitente debe indicar que el agua se está calentando).

Llegado a la temperatura máxima se apaga la resistencia (s enciende un led que indique ese estado) se espera 1 o 2 segundos y se abre la canilla. Cuando la cantidad de agua llega a los 50 litros se cierra la cantilla se espera 1 o 2 segundos y se activa la bomba y se conecta la resistencia, el ciclo se repite.

El termotante tiene una resistencia que calienta el agua, un flujo de agua entrante y un flujo de agua saliente y cinco leds que me permiten ver el estado de funcionamiento en todo momento. El alumno debe especificar que indica cada led y que está haciendo el microcontrolador en cada parte del programa.

Esta guía con todo lo conversado en clase es orientativo. La idea es que en todo momento mediante una visión del estado de los leds se tenga conocimiento de lo que está sucediendo en el termotanque.

El esquema del mismo es el siguiente



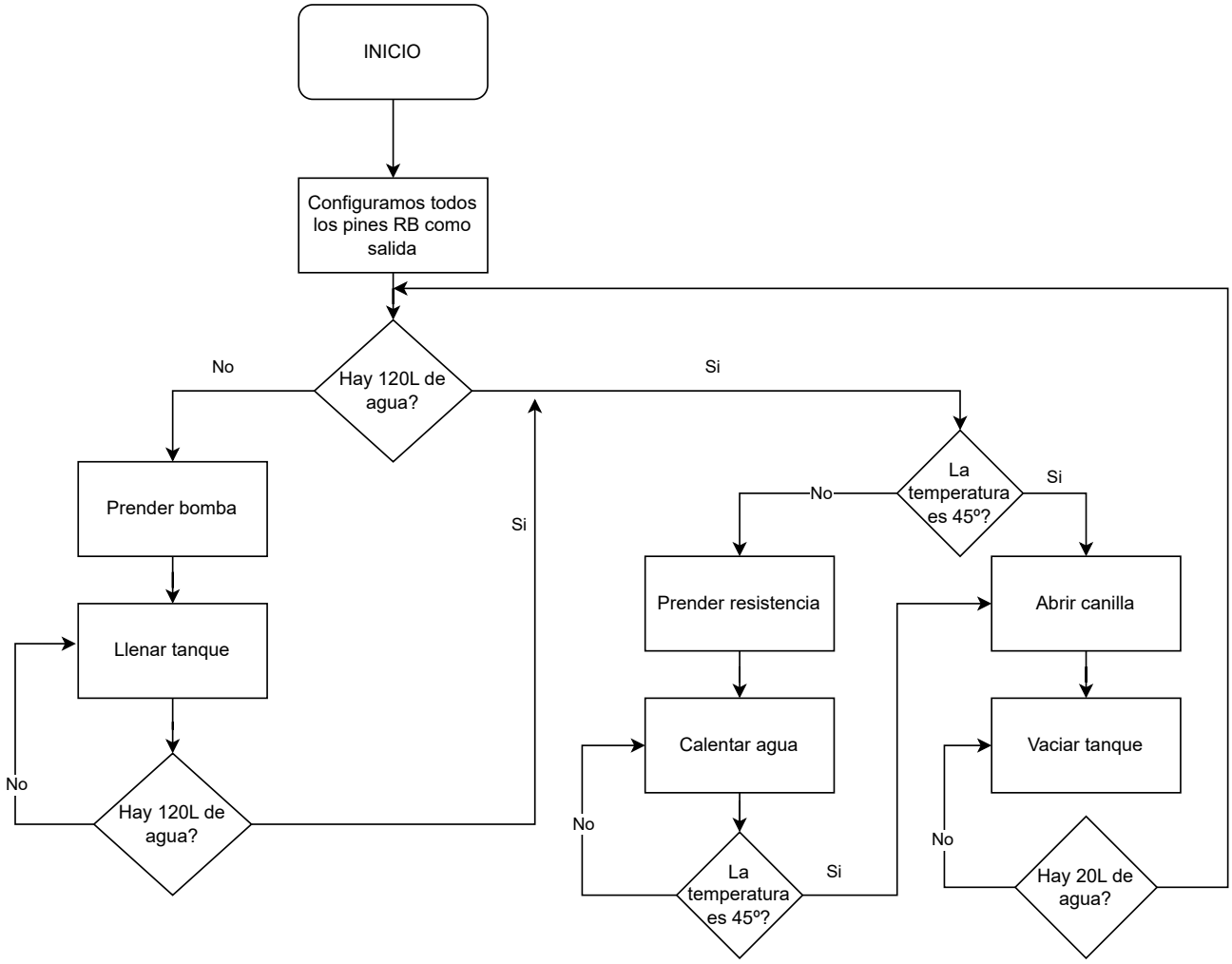
Ejercicio 2

En base a un circuito compuesto por un PIC16F628A con leds en los terminales RB0, RB1, RB2 y RB3 se desea desarrollar una serie de programas que permitan:

1. **Encender todos los leds.**
2. **Encender y apagar todos los leds cada un segundo.**
3. **Ídem [2] pero que esten un segundo prendidos y 500 ms apagados**
4. **Encender los leds desde el de RB0 hasta el RB3 con una demora de 500ms entre ellos.**
5. **Idem punto [4] pero una vez que se encienden todos, deben comenzar a apagarse desde el RB3 al RB0 con la misma demora y realizar todo el ciclo indefinidamente.**

Presentación

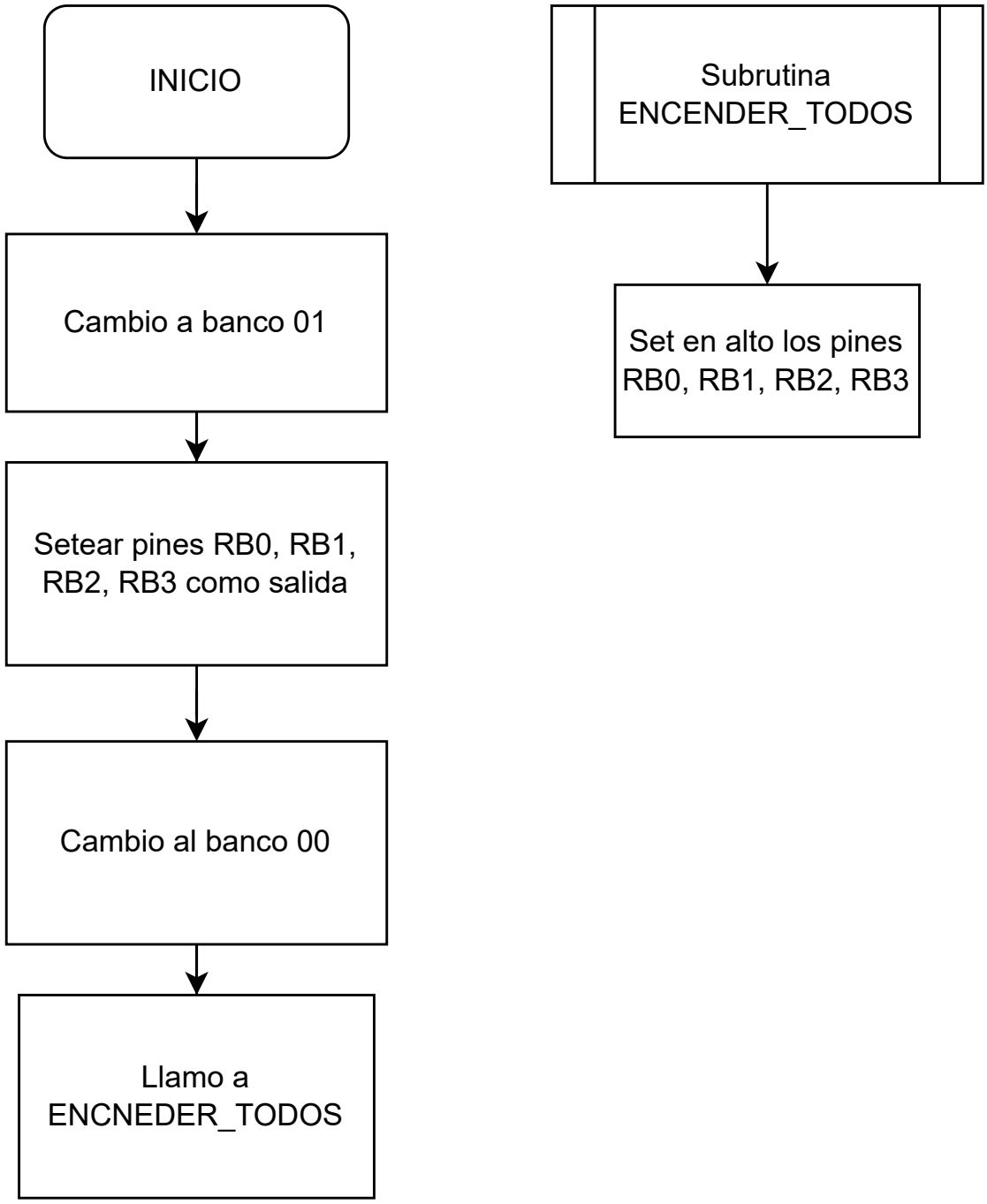
Folio con caratula donde se indique nombre y apellido del alumno, DNI, materia, año de cursada
Diagrama de flujo. Impreso del código y hacerlo funcionar en mplab sin errores ni warning.


INICIO
Configuramos todos los pines RB como salida
Hay 120L de agua?
No
Si
Prender bomba
Llenar tanque
Hay 120L de agua?
No
Si
La temperatura es 45º?
No
Si
Prender resistencia
Calentar agua
La temperatura es 45º?
No
Si
Abrir canilla
Vaciar tanque
Hay 20L de agua?
No

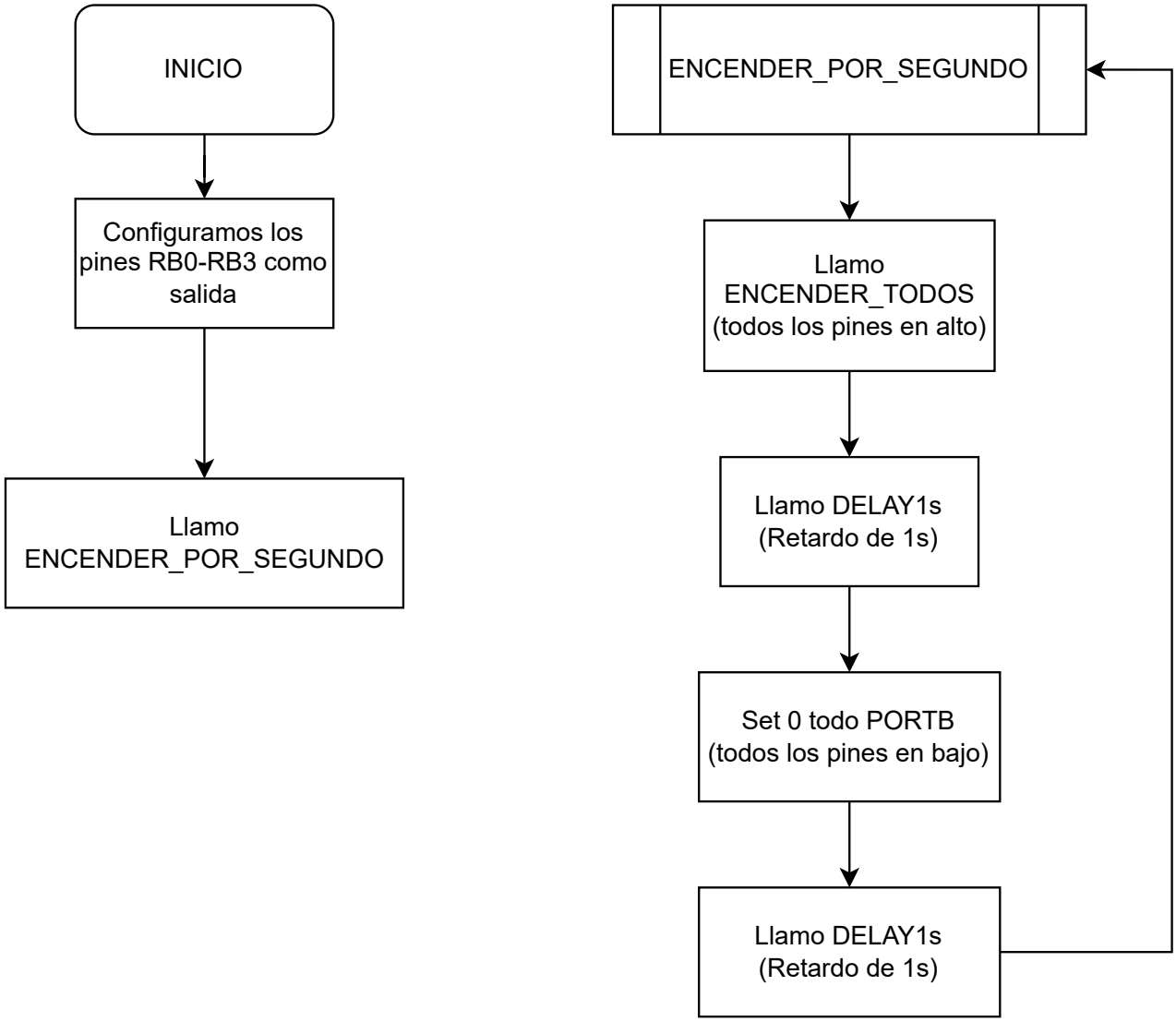
```
1   ;-------------------------------------------------------------------------------;
2   ; Dal Degan Santiago - 45421137
3   ; Ejercicio 1 - Termotanque
4   ; En este programa controla un termotanque utilizando el PIC16F628A
5   ;-------------------------------------------------------------------------------;
6
7   ; se configura el pic
8   #include <p16f628a.inc>
9       LIST    P=16f628a
10
11      org 0
12
13  ; declaracion de variables
14
15  tmax    equ d'45'
16  maxagua equ d'110'
17  minagua equ d'50'
18  tagua   equ 0x20
19  cagua   equ 0x21
20  bomba   equ 0
21  bled    equ 1
22  tled    equ 2
23  resis   equ 3
24  resled  equ 4
25  maxled  equ 5
26  canilla equ 6
27  ; 0x20-0x21 utilizado
28
29  ;-----------SALIDAS-----------
30      bsf STATUS, 5
31      clrf TRISB ; Configuramos los pines B como salida
32      bcf STATUS, 5
33  ;-----------SALIDAS-----------
34
35  ;-----------VARIABLES PRUEBA-----------
36      movlw d'30'
37      movwf tagua
38      movlw d'100'
39      movwf cagua
40      movlw 0x00
41      movwf PORTB ; Limpiamos las salias
42  ;-----------VARIABLES PRUEBA-----------
43
44  ;-----------INICIO CICLO-----------
45  INICIO
46
47      clrwdt ; Limpiamos el watchdog
48      movlw maxagua ; Movemos el nivel maximo de agua a w
49      subwf cagua, 0 ; Restamos 110 - cantidad de agua
50
51      ; if
52      btfss STATUS, 2 ; Checkeamos si el bit de STATUS en la pos 2 es 0
53
54      ; false
55      goto PRENDER_BOMBA ; si el bit es 0 (es decir no hay suficiente agua), prendemos la bomba
56
57      ; true
58      goto CHECK_TEMP
59  ;-----------INICIO CICLO-----------
60
61  ;-----------PRENDER BOMBA-----------
62  PRENDER_BOMBA
63      bsf PORTB, bomba ; Prendemos la bomba
64      bsf PORTB, bled ; Prendemos el led marcador de la bomba
65
66  PRENDER_BOMBA_LOOP
67      incf cagua, 1 ; Simulamos el agua subiendo
68      movlw maxagua
69      subwf cagua, 0
70
71      ; if
```

```
72      btfss STATUS, 2 ; checkeamos nuevamente el nivel del agua
73
74      ; false
75      goto PRENDER_BOMBA_LOOP ; Si sigue bajo repetimos el loop
76
77      ; true
78      bcf PORTB, bomba ; Apagamos la bomba si el agua llego al nivel correcto
79      bcf PORTB, bled ; Apagamos el led de la bomba
80      bsf PORTB, maxled ; Prendemos el led de termotanque lleno
81      goto CHECK_TEMP
82  ;-----------PRENDER BOMBA-----------
83
84  ;-----------CHECK TEMPERATURA-----------
85  CHECK_TEMP
86      movlw tmax
87      subwf tagua, 0
88
89      ; if
90      btfss STATUS, 2 ; Checkeamos si la temperatura del agua es la maxima
91
92      ; false
93      goto PRENDER_RES ; Si no lo es prendemos la resistencia
94
95      ; true
96      bsf PORTB, tled ; Prendemos el led de temperatuar alcanzada
97      goto ABRIR_CANILLA ; Si lo es abrimos la canilla
98  ;-----------CHECK TEMPERATURA-----------
99
100 ;-----------PRENDER RESISTENCIA-----------
101 PRENDER_RES
102     bsf PORTB, resis
103
104 PRENDER_RES_LOOP
105     incf tagua, 1 ; Simulamos el aumento de temperatura del agua
106     movlw tmax
107     subwf tagua, 0
108
109     call DELAY250 ; Hacemos titilar el led
110     bsf PORTB, resled
111     call DELAY250
112     bcf PORTB, resled
113
114     ; if
115     btfss STATUS, 2 ; Checkeamos la temperatura del agua
116
117     ; false
118     goto PRENDER_RES_LOOP ; Si no se alcanzo la temperatura repetimos
119
120     ; true
121     ; Si la temperatura fue alcanzada apagamos la resistencia y el led
122     bcf PORTB, resis
123     bcf PORTB, resled
124     bsf PORTB, tled ; Prendemos el led de temperatuar alcanzada
125     goto ABRIR_CANILLA
126 ;-----------PRENDER RESISTENCIA-----------
127
128 ;-----------ABRIR CANILLA-----------
129 ABRIR_CANILLA
130     call DELAY1 ; Delay de 1 segundo antes de abrir la canilla
131
132     bsf PORTB, canilla ; Abrimos la canilla
133
134 ABRIR_CANILLA_LOOP
135     decf cagua, 1
136     movlw minagua
137     subwf cagua, 0
138
139     btfss STATUS, 2 ; Checkeamos si el agua llego a los 50 litros
140     ;false
141     goto ABRIR_CANILLA_LOOP
142
```

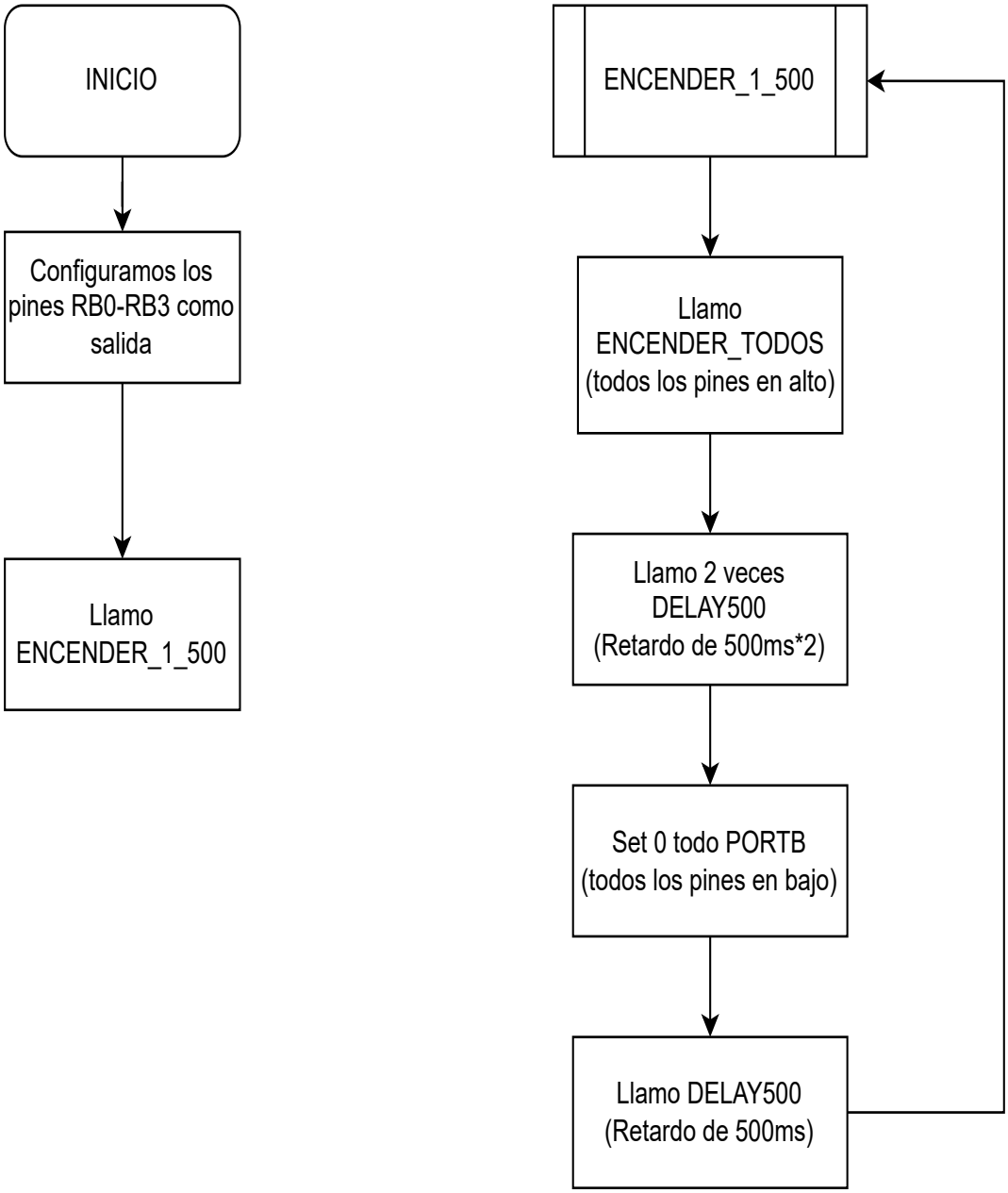
```
143      ;true
144      call DELAY1
145      call DELAY1
146      bcf PORTB, canilla
147      bcf PORTB, tled
148      bcf PORTB, maxled
149      goto INICIO
150  ;-----------ABRIR CANILLA-----------
151
152  ;-----------DELAY 1s-----------
153  DELAY1
154      ; estamos andando a 4Mhz
155      ; un ciclo de instruccion son 4 ciclos de relojs es decir 4/4 = 1Mhz
156      ; para calcular el tiempo hacemos 1/1Mhz = 1us
157      ; si queremos lograr un delay de 1s necesitamos
158      ; 1M ciclos de maquina
159      ; sin embargo como toma 3 ciclos de maquina hacer el proceso
160      ; dividimos 1M/3 = 333.333,33...
161      ; ya que no entra eso en un registro lo separaremos en 3
162      ; por cada valor del un registro el otro registro contara
163      ; regresivamente su valor
164      ; es decir reg1=10 reg=20, por cada 10 ciclos restando reg1
165      ; se restara uno de reg2
166      ; para saber los valores necesitamos reg1*reg2*reg3 = 333.333
167      ; raiz cubica 333.333 = 69.3
168
169      movlw d'69'
170      movwf 0x24
171  REG2
172      movlw d'69'
173      movwf 0x25
174  REG3
175      movlw d'70' ; ya que da un valor con coma a la 3ra le sumo uno
176      movwf 0x26 ; no es un delay exacto asi que no deberia importar
177
178  START
179      decfsz 0x26, 1
180      goto START
181      decfsz 0x25, 1
182      goto REG3
183      decfsz 0x24, 1
184      goto REG2
185      clrwdt
186      return
187  ;-----------DELAY 1s-----------
188
189  ;-----------DELAY 250ms-----------
190  DELAY250
191      ; La logica es la misma pero para 250ms
192
193      movlw d'43'
194      movwf 0x24
195  REG5
196      movlw d'43'
197      movwf 0x25
198  REG4
199      movlw d'44'
200      movwf 0x26
201
202  START1
203      decfsz 0x26, 1
204      goto START
205      decfsz 0x25, 1
206      goto REG5
207      decfsz 0x24, 1
208      goto REG4
209      clrwdt
210      return
211  ;-----------DELAY 500ms-----------
212      end
```


INICIO
Cambio a banco 01
Setear pines RB0, RB1, RB2, RB3 como salida
Cambio al banco 00
Llamo a ENCNEDER_TODOS
Subrutina ENCENDER_TODOS
Set en alto los pines RB0, RB1, RB2, RB3

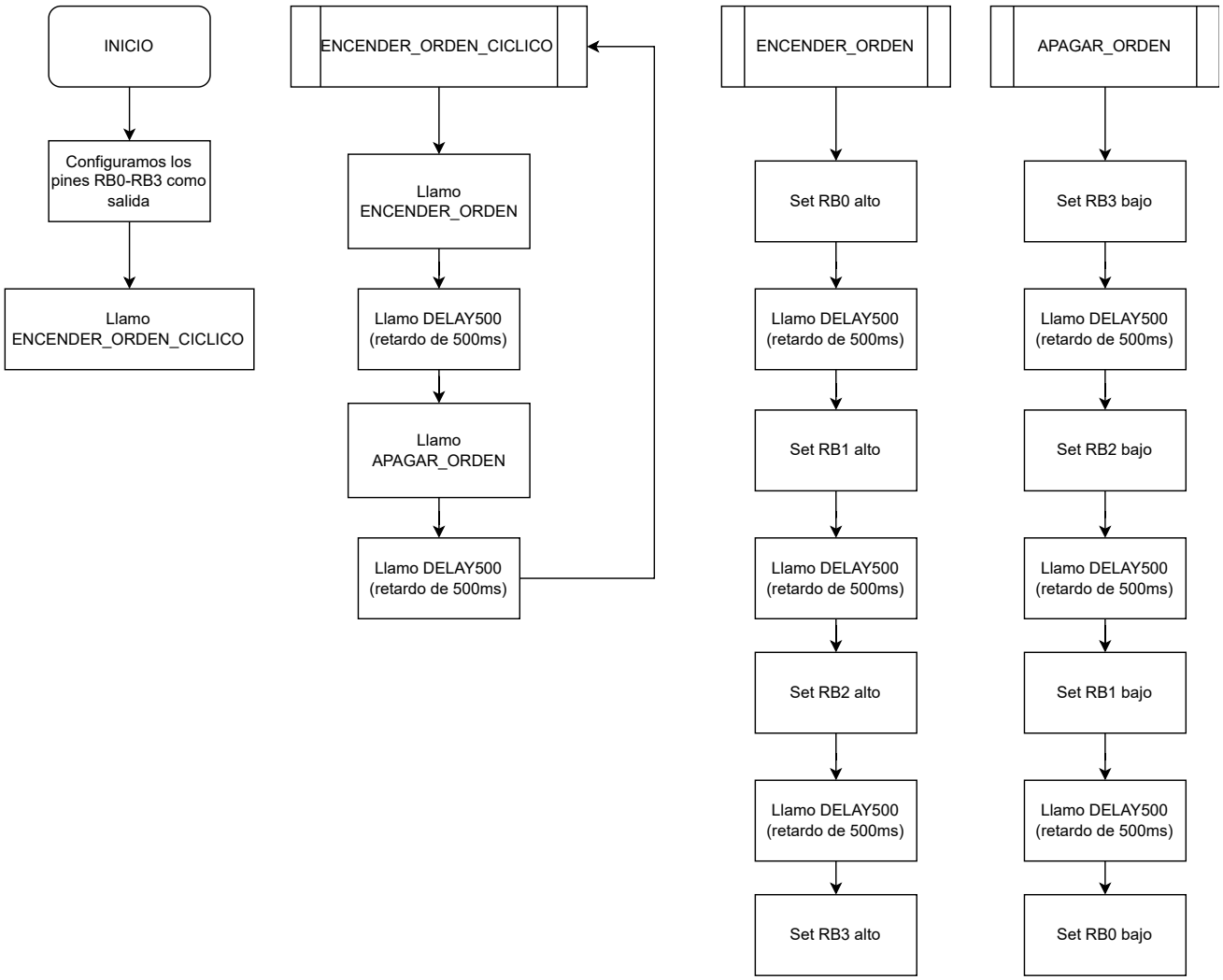
```
1   ;-------------------------------------------------------------------------------------;
2   ; Dal Degan Santiago - 45421137
3   ; Ejercicio 2 - Punto 1
4   ; En este programa se encienden los leds conectados a RB0, RB1, RB2, RB3
5   ;-------------------------------------------------------------------------------------;
6
7   ; se configura el pic
8   #include <p16f628a.inc>
9       LIST    P=16f628a
10
11      org 0
12
13      ; configuramos los puertos
14      bsf STATUS, RP0 ; cambiamos al segundo banco de memoria
15      movlw b'11110000'
16      movwf TRISB ; ponemos desde RB0 hasta RB3
17      bcf STATUS, RP0 ; volvemos al primer banco
18
19  INICIO
20      call ENCENDER_TODOS ; encendemos todos los leds
21      clrwdt ; limpiamos el watchdog
22      goto INICIO ; repetimos
23
24
25  ENCENDER_TODOS
26      movlw b'00001111' ; ponemos el 1 en los ultimos 4 bits de PORTB
27      movwf PORTB
28      return
29
30      end
```


INICIO
Configuramos los pines RB0-RB3 como salida
Llamo ENCENDER_POR_SEGUNDO
ENCENDER_POR_SEGUNDO
Llamo ENCENDER_TODOS (todos los pines en alto)
Llamo DELAY1s (Retardo de 1s)
Set 0 todo PORTB (todos los pines en bajo)
Llamo DELAY1s (Retardo de 1s)

```
1   ;-------------------------------------------------------------------------------;
2   ; Dal Degan Santiago - 45421137
3   ; Ejercicio 2 - Punto 2
4   ; En este programa se encienden los leds conectados a RB0, RB1, RB2, RB3 espera un segundo y los
    apaga
5   ;-------------------------------------------------------------------------------;
6
7   ; se configura el pic
8   #include <p16f628a.inc>
9       LIST    P=16f628a
10
11      org 0
12
13      ; configuramos los puertos
14      bsf STATUS, RP0 ; cambiamos al segundo banco de memoria
15      movlw b'11110000'
16      movwf TRISB ; ponemos desde RB0 hasta RB3
17      bcf STATUS, RP0 ; volvemos al primer banco
18
19  INICIO
20      call ENCENDER_POR_SEGUNDO ; Encender los leds cada un segundo
21      goto INICIO ; repetimos
22
23  ENCENDER_TODOS
24      movlw b'00001111' ; ponemos el 1 en los ultimos 4 bits de PORTB
25      movwf PORTB
26      return
27
28  ENCENDER_POR_SEGUNDO
29      call ENCENDER_TODOS ; Encendemos todos los leds
30      call DELAY1s ; delay de un segundo
31      clrf PORTB ; apalagamos los leds
32      call DELAY1s ; delay de un segundo
33      clrwdt ; limpiamos el watchdog
34      return
35
36  DELAY1s
37      ; estamos andando a 4Mhz
38      ; un ciclo de instruccion son 4 ciclos de relojs es decir 4/4 = 1Mhz
39      ; para calcular el tiempo hacemos 1/1Mhz = 1us
40      ; si queremos lograr un delay de 1s necesitamos
41      ; 1.000.000 ciclos de maquina
42      ; sin embargo como toma 3 ciclos de maquina hacer el proceso
43      ; dividimos 1.000.000/3 = 333.333,333...
44      ; ya que no entra eso en un registro lo separaremos en 3
45      ; por cada valor del un registro el otro registro contara
46      ; regresivamente su valor
47      ; es decir reg1=10 reg=20, por cada 10 ciclos restando reg1
48      ; se restara uno de reg2
49      ; para saber los valores necesitamos reg1*reg2*reg3 = 333.333
50      ; raiz cubica 333.333 = 69.3
51
52      movlw d'69'
53      movwf 0x20
54  REG2
55      movlw d'69'
56      movwf 0x21
57  REG3
58      movlw d'70'
59      movwf 0x22
60
61  START
62      decfsz 0x22, 1
63      goto START
64      decfsz 0x21, 1
65      goto REG3
66      decfsz 0x20, 1
67      goto REG2
68      return
69
70      end
```


INICIO
Configuramos los pines RB0-RB3 como salida
Llamo ENCENDER_1_500
ENCENDER_1_500
Llamo ENCENDER_TODOS (todos los pines en alto)
Llamo 2 veces DELAY500 (Retardo de 500ms*2)
Set 0 todo PORTB (todos los pines en bajo)
Llamo DELAY500 (Retardo de 500ms)

```
1   ;-------------------------------------------------------------------------------;
2   ; Dal Degan Santiago - 45421137
3   ; Ejercicio 2 - Punto 3
4   ; En este programa se encienden los leds conectados a RB0, RB1, RB2, RB3 espera un segundo y los
    apaga por 500ms
5   ;-------------------------------------------------------------------------------;
6
7   ; se configura el pic
8   #include <p16f628a.inc>
9       LIST    P=16f628a
10      org 0
11
12      ; configuramos los puertos
13      bsf STATUS, RP0 ; cambiamos al segundo banco de memoria
14      movlw b'11110000'
15      movwf TRISB ; Ponemos desde RB0 hasta RB3
16      bcf STATUS, RP0 ; Volvemos al primer banco
17
18  INICIO
19      call ENCENDER_1_500
20      goto INICIO
21
22  ENCENDER_1_500
23      call ENCENDER_TODOS ; Encendemos todos los leds
24      call DELAY500 ; Esperamos 500ms 2 veces
25      call DELAY500
26      clrf PORTB ; Apagamos los leds
27      call DELAY500 ; Delay 500
28      call ENCENDER_TODOS ; Encendemos todos los leds
29      clrwdt ; Limpiamos el watchdog
30      return
31
32  ENCENDER_TODOS
33      movlw b'00001111'
34      movwf PORTB
35      return
36
37  DELAY500
38      ; estamos andando a 4Mhz
39      ; un ciclo de instruccion son 4 ciclos de relojs es decir 4/4 = 1Mhz
40      ; para calcular el tiempo hacemos 1/1Mhz = 1us
41      ; si queremos lograr un delay de 1s necesitamos
42      ; 500.000 ciclos de maquina
43      ; sin embargo como toma 3 ciclos de maquina hacer el proceso
44      ; dividimos 500.000/3 = 166.666,6666...
45      ; ya que no entra eso en un registro lo separaremos en 3
46      ; por cada valor del un registro el otro registro contara
47      ; regresivamente su valor
48      ; es decir reg1=10 reg=20, por cada 10 ciclos restando reg1
49      ; se restara uno de reg2
50      ; para saber los valores necesitamos reg1*reg2*reg3 = 166.666
51      ; raiz cubica 166.666 = 55.03
52
53      movlw d'55'
54      movwf 0x20
55  REG2
56      movlw d'55'
57      movwf 0x21
58  REG3
59      movlw d'55'
60      movwf 0x22
61
62  START
63      decfsz 0x22, 1
64      goto START
65      decfsz 0x21, 1
66      goto REG3
67      decfsz 0x20, 1
68      goto REG2
69      return
70      end
```


INICIO
Configuramos los pines RB0-RB3 como salida
Llamo ENCENDER_ORDEN_CICLICO
ENCENDER_ORDEN_CICLICO
Llamo ENCENDER_ORDEN
Llamo DELAY500 (retardo de 500ms)
Llamo APAGAR_ORDEN
Llamo DELAY500 (retardo de 500ms)
ENCENDER_ORDEN
Set RB0 alto
Llamo DELAY500 (retardo de 500ms)
Set RB1 alto
Llamo DELAY500 (retardo de 500ms)
Set RB2 alto
Llamo DELAY500 (retardo de 500ms)
Set RB3 alto
APAGAR_ORDEN
Set RB3 bajo
Llamo DELAY500 (retardo de 500ms)
Set RB2 bajo
Llamo DELAY500 (retardo de 500ms)
Set RB1 bajo
Llamo DELAY500 (retardo de 500ms)
Set RB0 bajo

```
1   ;-------------------------------------------------------------------------------------;
2   ; Dal Degan Santiago - 45421137
3   ; Ejercicio 2 - Punto 4 y 5
4   ; En este programa se encienden los leds conectados a RB0, RB1, RB2, RB3 en orden con
5   ; un delay de 500ms entre ellos
6   ; Luego se apagan en el orden contrario
7   ;-------------------------------------------------------------------------------------;
8
9   ; se configura el pic
10  #include <p16f628a.inc>
11      LIST    P=16f628a
12
13      org 0
14
15      ; configuramos los puertos
16      bsf STATUS, RP0 ; cambiamos al segundo banco de memoria
17      movlw b'11110000'
18      movwf TRISB ; Ponemos desde RB0 hasta RB3
19      bcf STATUS, RP0 ; Volvemos al primer banco
20
21  INICIO
22      call ENCENDER_ORDEN_CICLICO ; Empezamos el programa
23      goto INICIO
24
25
26  ENCENDER_ORDEN
27      ; Prendemos los led en orden con 500ms de delay entre ellos
28      bsf PORTB, 0
29      call DELAY500
30      bsf PORTB, 1
31      call DELAY500
32      bsf PORTB, 2
33      call DELAY500
34      bsf PORTB, 3
35      return
36
37  APAGAR_ORDEN
38      ; Apagamos los led en orden inverso con 500ms de delay entre ellos
39      bcf PORTB, 3
40      call DELAY500
41      bcf PORTB, 2
42      call DELAY500
43      bcf PORTB, 1
44      call DELAY500
45      bcf PORTB, 0
46      return
47
48  ENCENDER_ORDEN_CICLICO
49      call ENCENDER_ORDEN ; Encendemos los led en orden
50      call DELAY500 ; Delay de 500ms
51      call APAGAR_ORDEN ; Apagamos los led en orden
52      call DELAY500
53      return
54
55  DELAY500
56      ; estamos andando a 4Mhz
57      ; un ciclo de instruccion son 4 ciclos de relojs es decir 4/4 = 1Mhz
58      ; para calcular el tiempo hacemos 1/1Mhz = 1us
59      ; si queremos lograr un delay de 1s necesitamos
60      ; 500.000 ciclos de maquina
61      ; sin embargo como toma 3 ciclos de maquina hacer el proceso
62      ; dividimos 500.000/3 = 166.666,6666...
63      ; ya que no entra eso en un registro lo separaremos en 3
64      ; por cada valor del un registro el otro registro contara
65      ; regresivamente su valor
66      ; es decir reg1=10 reg=20, por cada 10 ciclos restando reg1
67      ; se restara uno de reg2
68      ; para saber los valores necesitamos reg1*reg2*reg3 = 166.666
69      ; raiz cubica 166.666 = 55.03
70
71      movlw d'55'
```

```
72      movwf 0x20
73  REG2
74      movlw d'55'
75      movwf 0x21
76  REG3
77      movlw d'55'
78      movwf 0x22
79
80  START
81      decfsz 0x22, 1
82      goto START
83      decfsz 0x21, 1
84      goto REG3
85      decfsz 0x20, 1
86      goto REG2
87      clrwdt
88      return
89
90      end
```