

- Mínimo para aprobar: un problema
- Nombre y CI en cada hoja
- Numere las hojas, indique el total en la primera
- Utilice sólo un lado de las hojas
- Incluya un solo problema por hoja
- Sea prolijo

Si usa las directivas `.text` y `.data` indicar la dirección de comienzo de cada sección.

PROBLEMA 1

Se desea diseñar el controlador de un lavarropas con un sistema basado en microprocesador. El sistema contará con un puerto de salida **POUT[8]** para comandar los diferentes actuadores del lavarropas (motor, electroválvula, bomba de desagote, etc.).

El programa de lavado consiste de una sucesión de fases desde 0 hasta N que se realizan en secuencia. Para cada fase se especifica la DURACIÓN (medida en intervalos de 10 segundos) y el VALOR a escribir en el puerto POUT. Al comienzo de cada fase se escribe el valor en POUT y una vez transcurrida la DURACION se pasa a la siguiente fase.

El programa se almacena en una tabla en memoria, con los dos bytes de DURACION y VALOR para cada fase del programa. DURACION en la posición más baja.

Un valor de DURACION = 0xFF en la posición correspondiente a la fase N+1 de la tabla indica que se completaron todas las fases y se debe escribir 0 en POUT y quedar a la espera de alguna acción del operador.

Dirección	Contenido	Fase
+2*N+2	0xFF	
	VALOR_N	
+2*N	DURACION_N	N
...		
	VALOR_2	
+4	DURACION_2	2
	VALOR_1	
+2	DURACION_1	1
	VALOR_0	
TABLA_FASES	DURACION_0	0

El operador puede ordenar diferentes acciones a través de los siguientes botones que generan un pulso a 1 de corta duración:

- **START**, ordena comenzar el programa si éste no estaba activo, o recomenzar desde la primera fase de la tabla si ya se estaba ejecutando el programa.
- **NEXT**, pasa a la siguiente fase antes de que expire el tiempo DURACION de la fase actual. Si no se está ejecutando el programa, no deberá ejecutarse ninguna acción.
- **END**, termina la ejecución y queda a la espera de un nuevo start.

Además del puerto POUT, el sistema cuenta con las siguientes entradas y salidas:

- **TIC_10SEG**: señal con un pulso periódico cada 10seg. que debe alimentar un contador para llevar la cuenta de DURACION de cada fase.
- **FASE[8]**: nro de fase actual (entre 0 y N) mientras el proceso de lavado está activo, 0xFF el resto del tiempo

Hay un único programa cuya tabla está almacenada en ROM a partir de la dirección TABLA_FASES. Se puede suponer que el programa tiene al menos una fase y menos de 127, es decir que la tabla siempre cabe en una página de memoria

El sistema hardware debe contar con las siguientes características: un bloque contador con su correspondiente controlador de interrupciones para medir la DURACION de la fase y avanzar de fase, y otro controlador de interrupciones con la mayor prioridad para recibir la señal NEXT. Cuenta además con 32K de ROM, 32K de RAM que se supondrá implementada.

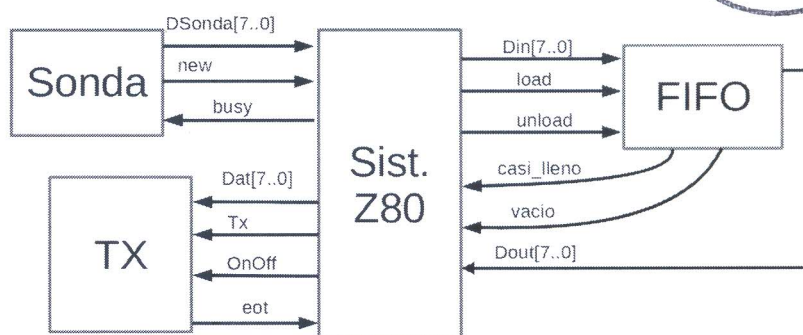
Se pide:

- Hardware T80 completo del sistema incluyendo puertos, periféricos y decodificación.
- Programas de atención de interrupciones generadas por NEXT y contador.
- Programa principal que inicializa todos los periféricos, puertos y variables y luego queda en bucle monitoreando START y END para proceder como sea necesario.

PROBLEMA 2

Se desea enviar datos generados por una sonda a través de un transmisor. Para ello se deberá diseñar el siguiente sistema.

La sonda de la figura genera datos de 8 bits a una tasa variable. Cuando tiene un dato nuevo espera a que su entrada **busy** esté en nivel bajo, coloca el dato en su salida **DSonda** y genera un flanco de subida en su



salida **new** indicando que el dato en su salida es válido. La señal **busy** debe subir a 1 cuando se detecta un flanco de subida en la señal **new** y debe bajar a 0 cuando el sistema lee el nuevo dato.

La entrada **OnOff** enciende al transmisor cuando está en 1 y lo apaga cuando está en 0. Si está encendido, cuando hay un flanco de subida en su entrada **Tx** el transmisor envía el dato que se encuentra en su entrada **Dat**. Luego, al finalizar la transmisión genera un pulso a 1 de corta duración en su salida **eot** para indicar el fin de la transmisión.

Los datos generados por la sonda deben ser almacenados temporalmente en un dispositivo FIFO que implementa una cola y funciona de la siguiente forma. El valor de **Din[7..0]** se escribe en el FIFO con un flanco de subida en **load**. La salida **Dout[7..0]** presenta el valor del dato más antiguo almacenado en el FIFO. Un flanco de subida en **unload** retira ese dato del FIFO. Además el dispositivo tiene las salidas **vacio** (=1 si no hay datos almacenados) y **casi_lleno** (=1 si quedan pocos lugares libres en la memoria FIFO).

La transmisión de datos se debe realizar cuando el dispositivo FIFO indica que está casi lleno. Cuando sucede eso se debe prender el transmisor y comenzar a enviar datos hasta que el dispositivo FIFO indique que está vacío. Cada dato se transmitirá manejando las entradas **Dat** y **Tx** del transmisor ya descriptas. Cuando llega el pulso de **eot** se debe transmitir el siguiente dato del FIFO si hubiera. En caso contrario (FIFO vacío) se debe apagar el transmisor y esperar a que el FIFO vuelva a estar casi lleno para comenzar con una nueva transmisión.

Se debe diseñar un sistema con microprocesador Z80 que implemente el funcionamiento descripto. Se debe generar interrupciones en modo 1 con un flanco de subida de la señal **eot** para implementar la transmisión de datos hasta vaciar el FIFO.

El programa principal debe monitorear la interfaz con la sonda para recibir los datos que lleguen y escribirlos en el FIFO. Además, si no hay una transmisión iniciada deberá detectar si el FIFO está casi lleno para encender el transmisor e iniciar una transmisión enviando el primer dato (los restantes datos serán enviados desde sucesivas rutinas de atención a la interrupción). Una vez iniciada una transmisión, se deberá ignorar **casi_lleno** hasta que el FIFO está vacío, en que se vuelve a la condición inicial. Se debe evitar quedar en loop sin monitorear la sonda para evitar perder datos.

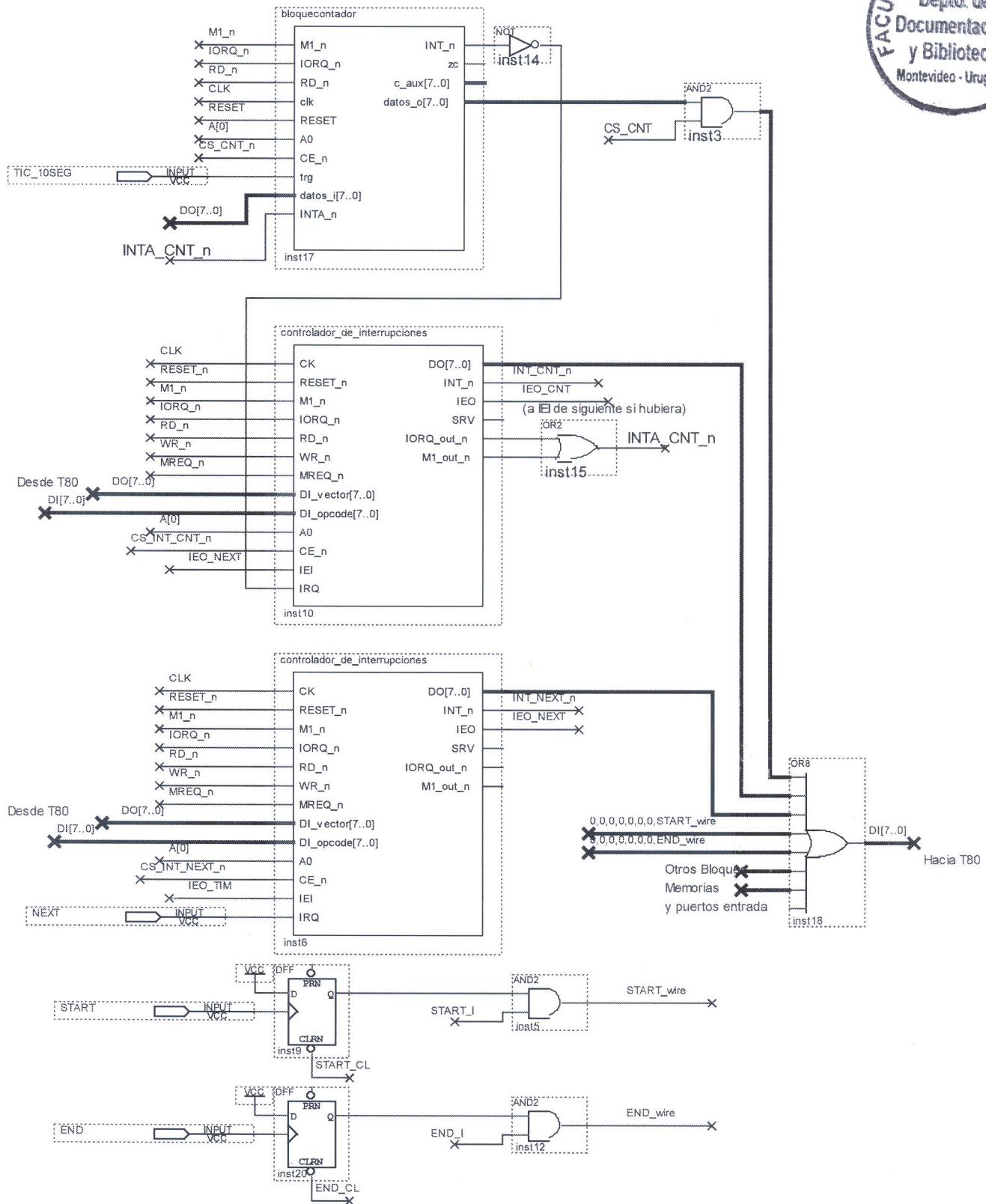
Se pide:

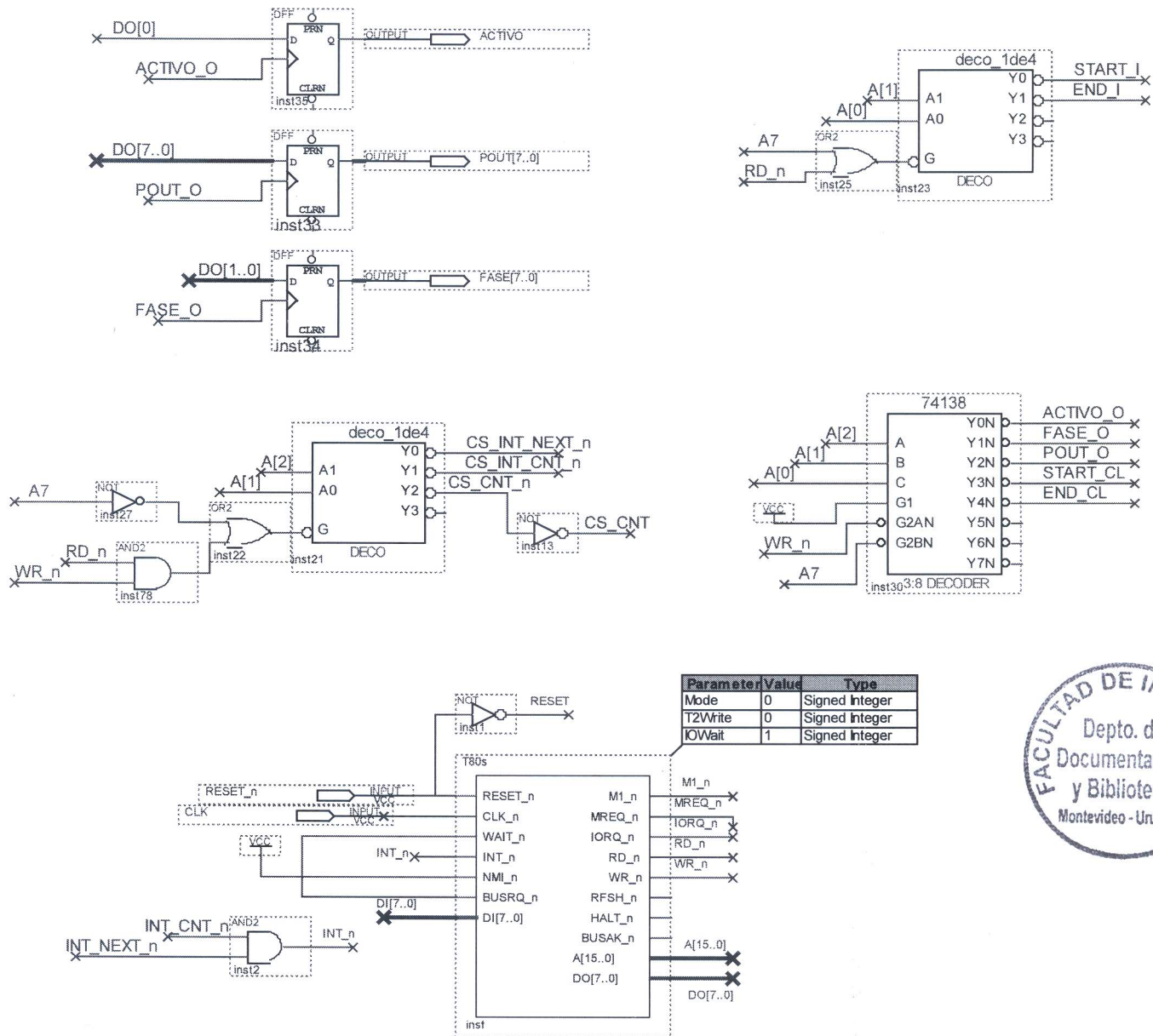
- Todo el HW necesario para el sistema, utilizando 32K de ROM y 32 K de RAM y la adecuada conexión de la sonda, el transmisor y el dispositivo FIFO en el espacio de E/S.
- Todo el software (inicialización del sistema, programa principal, directivas para reserva de memoria y rutinas de atención a la interrupción).

Nota: se puede suponer que las tasas de generación y transmisión de datos son tales que el dispositivo FIFO nunca llega a llenarse y por lo tanto no se pierden datos.

PROBLEMA 1

a) Hardware





FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA

;Puerto de entrada

START_I equ 0x00

END_I equ 0x01

;Puerto salida

ACTIVO_O equ 0x00

FASE_O equ 0x01

POUT_O equ 0x02

START_CL equ 0x04

END_CL equ 0x05

;Perifericos

CI_NEXT_VI_AD equ 0x80

CI_NEXT_CL_AD equ 0x81

CI_CONT_VI_AD equ 0x82

CI_CONT_CL_AD equ 0x83

CONT_CTE_AD equ 0x84

CONT_CW_AD equ 0x85

b)

org 0x1000

rutint_next:

ei

push AF

ld A, (ESTADO)

cp ESTADO_DETENIDO

jp Z, fin_rutint_next

call rut_siguiente_fase

fin_rutint_next:

pop AF

reti

rutint_cont:

jp rutint_next

;---Subrutina auxiliar

rut_siguiente_fase:

push AF

push HL

ld HL, TABLA_FASES

INTRODUCCIÓN A LOS MICROPROCESADORES

Febrero 2023

ld A, (NEXT_FASE)
sla A ; offset en tabla de fase a
ejecutar

ld L, A

ld A, (HL)

cp 0xFF

jp Z, detenerse

siguiente:

out (CONT_CTE_AD), A ; cargo duracion
en contador

inc HL

ld A, (HL) ; A= VALOR

out (POUT_O), A ; escribo valor

ld A, CONT_EI_CW

out (CONT_CW_AD), A ; reinicio
contador

ld A, (NEXT_FASE)

out (FASE), A ; escribo fase actual

inc A

ld (NEXT_FASE), A

jp fin_siguiente_fase

detenerse:

ld A, 0xFF

out (FASE_O) ; FASE =0xFF

ld A, ESTADO_DETENIDO

ld (ESTADO), A ; ESTADO=detenido

ld A, 0

out (POUT), A ; escribo valor

fin_siguiente_fase:

pop HL

pop AF

ret

c)

VI_NEXT equ 0x00

VI_CONT equ 0x02

CONT_EI_CW equ 1010 0000; int habilitadas
y SW reset

ESTADO_DETENIDO equ 0xFF

ESTADO_FUNCIONANDO equ 0x00



FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA

org 0x8000 ; comienzo RAM

NEXT_FASE db 0

ESTADO db 0

org 0x4000

TABLA_FASES ; alineada a página

org 0x2000

TABLA_INT:

dw rutint_next

dw rutint_cont

org 0x0000

ld SP, 0x0000 ; tope de RAM + 1

im 2

ld A, TABLA_INT / 256

ld I, A

ld A, 0xFF

out (FASE_O), A

out (START_CL), A

out (END_CL), A

ld A, 0

out (POUT), A

ld A, ESTADO_DETENIDO

ld (ESTADO), A

ld A, VI_NEXT

out (CI_NEXT_VI_AD), A

out (CI_NEXT_CL_AD), A

ld A, VI_CONT

out (CI_CONT_VI_AD), A

out (CI_CONT_CL_AD), A

EI

loop:

in A, (START_I)

bit 0, A

INTRODUCCIÓN A LOS MICROPROCESADORES

Febrero 2023

jp NZ, start

ld A, (END_I)

bit 0, A

jp Z, loop

detener:

out (END_CL), A

ld A, ESTADO_DETENIDO

ld (ESTADO), A

ld A, 0

out (POUT_O), A

ld A, 0xFF

out (FASE_O), A

jp loop

start:

out (START_CL), A

ld A, ESTADO_FUNCIONANDO

ld (ESTADO), A

ld A, (TABLA_FASES)

out (CONT_CTE_AD), A

ld A, (TABLA_FASES + 1)

out (POUT_O), A

ld A, 0

out (FASE), A ; escribo fase actual

ld A, 1

ld (NEXT_FASE), A

ld A, CONT_EI_CW

out (CONT_CW_AD), A ; reinicio contador

jp loop



b) todo el software

```

busy EQU 0x00
DSONDA EQU 0x01
Dout EQU 0x02
s_fifo EQU 0x03
bit_vacio EQU 1
bit_casi_lleno EQU 0

DAT EQU 0x00
OnOff EQU 0x01
load EQU 0x02
unload EQU 0x03

; .text desde 0x0000
; .data desde 0x8000

.text
;; -- inic --
;; stack pointer
ld sp, 0
;; var y puerto OnOff=0
ld a, 0
ld (vOnOff), a
out (OnOff), a
;; borrar ff busy
in a, (busy)
;; interrupciones
im 1
ei
jp prog_ppal

;; -- isr --
;
; preservar estado
; si vacío {
;   OnOff = 0
; }
; else{
;   sacar dato de fifo
;   enviarlo a transmisor
; }
; restaurar estado

org 0x0038
isr:
push af
in a, (s_fifo)
bit bit_vacio, a
;; si vacío {
;;   OnOff = 0
jr z, else_vacio
ld a, 0
ld (vOnOff), a
out (OnOff), a
jr fin_isr
else_vacio:
;; else{
;;   sacar dato de fifo

```

```

in a, (Dout)
out (unload), a
;; enviarlo a transmisor
out (DAT), a
fin_isr:
pop af
ei
ret

```

;; -- prog_ppal --

```

; forever{
;   si nuevo dato sonda{
;     leo DSONDA
;     escribo en FIFO
;   }
;   si (OnOff == 0) {
;     si FIFO casi lleno{
;       OnOff=1
;       sacar dato de fifo
;       enviarlo a transmisor
;     }
;   }
; }

```

prog_ppal:

```

in a, (busy)
bit 0, a
jr z, fin_si_busy
;; si nuevo dato sonda{
;;   leo DSONDA
;;   escribo en FIFO
in a, (DSONDA)
out (load), a
ld a, (vOnOff)
or a
jr nz, fin_si_OnOff
;; si (OnOff == 0) {
in a, (s_fifo)
bit bit_casi_lleno, a
jr z, fin_si_casi_lleno
;; si FIFO casi lleno{
;;   OnOff=1
ld a, 0xFF
out (OnOff), a
ld (vOnOff), a
;; sacar dato de fifo
in a, (Dout)
out (unload), a
;; enviarlo a transmisor
out (DAT), a
fin_si_casi_lleno:
fin_si_OnOff:
fin_si_busy:
jr prog_ppal

```

.data

```

vOnOff: db 0

```

