Notas Técnicas de Uso y Aplicación

NT1010

3.10 INTERFASE A "PAD" DE "PLAYSTATION ONE" – PSONE
RUTINAS UTILITARIAS DE ADQUISICIÓN DE SEÑALES DEL "PAD"

Preparado por: Rangel Alvarado
Estudiante Graduando de Lic. en Ing. Electromecánica
Universidad Tecnológica de Panamá
Panamá, Panamá
"e-mail": issaiass@cwpanama.net
"web site": http://www.geocities.com/issaiass/

ÍNDICE

3.10.1 Introducción

Un mundo extenso y con amplia demanda es el mundo de los video juegos. Nosotros en particular estamos en espera del siguiente título para evaluarlo y pasar horas tras el televisor; más sin embargo, a pesar de estar descansando del estrés diario, estamos solo siendo usuarios finales de la tecnología sin usarla a nuestro favor. Esta nota trata de enfocar la adquisición de señales de un Pad o Control de PSX (Playstation), y visualizarla en una LCD 16 X 4.

Un control de PSX, consta de más de dieciséis (16) botones y el consumo por pruebas caseras realizadas, es de 7mA en modo de espera, mientras que con la mayor cantidad de teclas presionadas que el autor pudo acceder genera un consumo total de 21 mA o mayor.

La pregunta es: ¿Cómo envía la información de todos estos botones con solo siete cables?. Y su respuesta es, serialmente. A pesar de tener siete (7) cables, recuerde que se necesitan dos (2) cables de alimentación, lo que limita solo a cinco (5) cables.

Aplicaciones en las que se puede utilizar este "pad", es para generar un videojuego primitivo como "Pong", "Pac Man"; controlar un robot utilizando módulos inalámbricos como "ZigBee" de ultra bajo consumo; hacer su pad "cordless" utilizando igualmente módulos inalámbricos o para el microcontrolador JK3, que solo posee un pin KBI, PTA6, comprometido por el oscilador, (ver NT0108), puede reemplazarse por este "pad" que gracias a Sony Corp. es una de las consolas más utilizadas y se pueden adquirir localmente.

3.10.2 Materiales

1. Microcontrolador: JL3 / JK3 / GP32
2. Potenciómetro: 10k @ 20k, Jameco Part No: 43001
3. Display LCD: LCD 16 X 4, Jameco Part No: 210761
4. Tarjeta de Desarrollo TD68HC908JL3 o similar y su microcontrolador
5. Plantilla de proyectos: "Breadboard" JE27, Jameco Part No: 20811
6. Fuente de Poder
7. Alambres AWG 20
8. Pelador de Alambres
9. "Pad" de Sony Playstation
10. Terminales: Terminales de Desconección X 7, Jameco Part No: 103667
11. Socket: Machine Tool Sip Socket – 9 pines, Jameco Part No: 102200
12. Kit de Aficionado: Pinzas del Kit, Jameco Part No: 99629

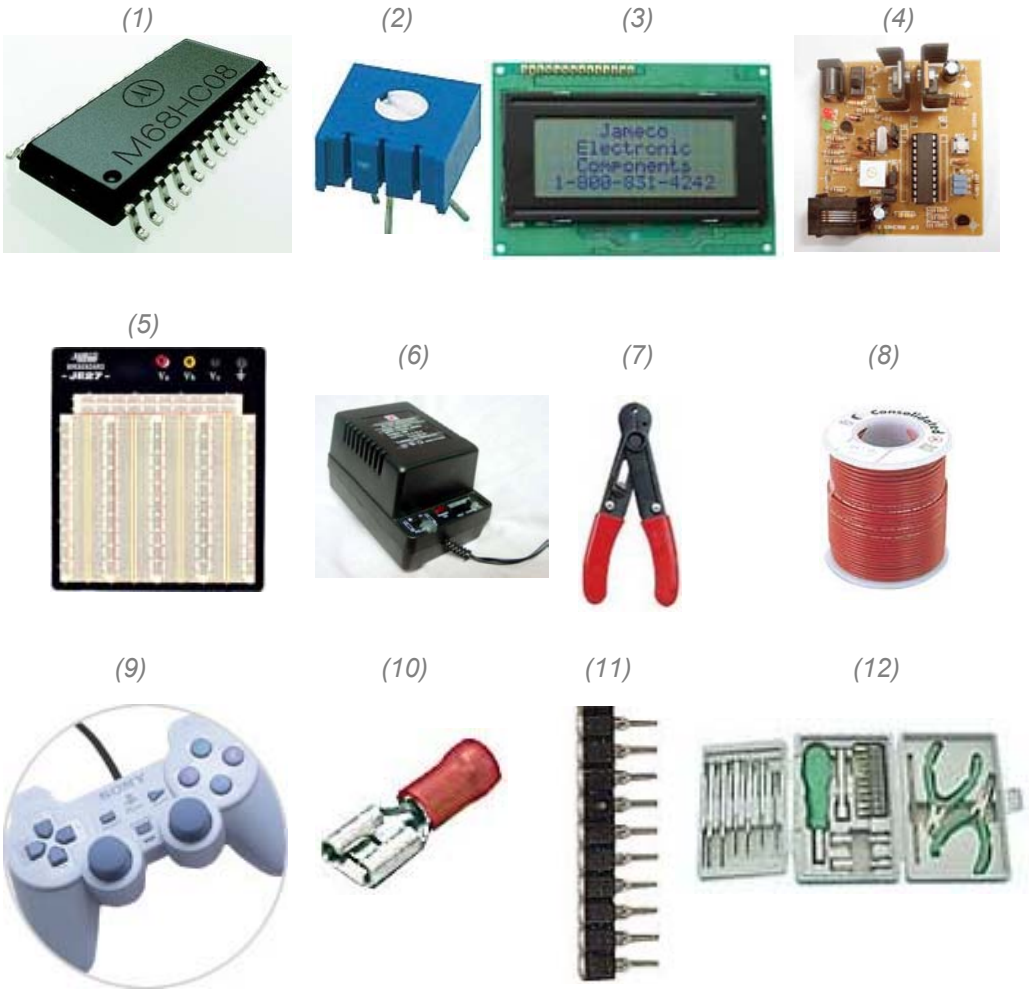


Figura 228. Listado de Materiales a Utilizar para "Pad" de "Playstation"

3.10.3 Descripción Física del "Hardware"

Para el siguiente párrafo, ayudarse por medio de la figuras de esta sección.

Cada "pad" o control de "Playstation" esta disponible en múltiples versiones, pero las más proliferadas son: la versión "dual-shock" y la versión digital, la versión digital (ver figura 229(b)) que consta de los botones clásicos, mientras un control "dual-shock" permite tener las ventajas de un control digital y su modo analógico que consta de sus "joystick", y un "rumble pack" adicional interno que causa la vibración de motores, esto por medio de ejes excéntricos que producen la vibración mecánica. Independientemente del tipo de control, el conector es universal, y consta de nueve (9) terminales, de las cuales cinco (5) de las nueve (9) son utilizadas para control, las cuales son detalladas en la tabla.

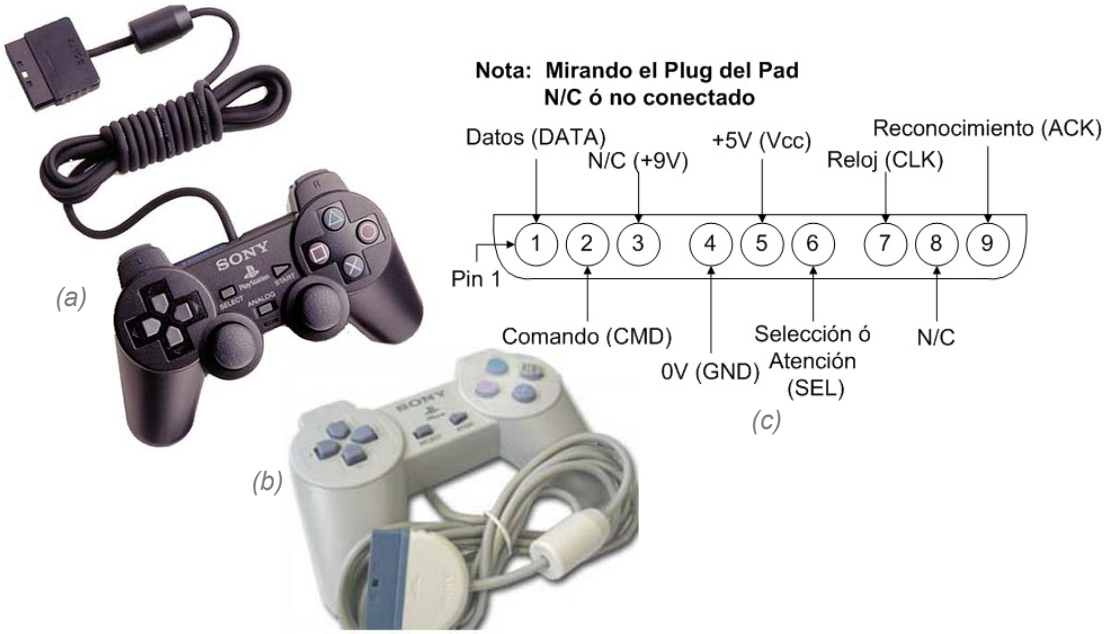


Tabla 79. Descripción de Pines del "Pad" de "PSOne"

Pin no.	Pin	Descripción
1	DATA	Información proveniente del "Pad"
2	CMD	Línea de comandos a enviar al "Pad"
3	+9V	Voltaje de operación de los motores
4	GND	Masa o Tierra
5	VCC	Alimentación del "Pad"
6	SEL	Ignora (0) o Selecciona (1) "Pad" a hablar
7	CLK	Reloj del sistema
8	N/C	Sin conexión
9	ACK	Señal de reconocimiento

N/C = No conectado.

Figura 229. Descripción Física de un "Pad" de "PSOne". (a) "Pad" de PSX Analógico. Un "pad" analógico se caracteriza por sus "joystick" analógicos, y "dual shock" interno que le da mayor realismo a los juegos. (b) Pad de PSX Digital. Un "pad" digital solo consta de los botones o contactos necesarios para la mayoría de los juegos. (c) Pines del Conector Sony hacia el Playstation. Descripción de pines, mirando desde el conector, la interfaz entre el pad y el playstation se da de manera serial, es por esto que puede tener más interruptores que pines de salida del conector.

3.10.4 Descripción de Señales

Guiarse en del grupo de figuras para el entendimiento del texto en cada sub-sección.

3.10.4.1 Transmisión de Datos del "Pad" (CMD)

Si el pad posee más de 16 interruptores y dos "joystick" analógicos, cinco (5) pines no son suficientes para que línea por línea se consigan 16 estados diferentes. Sony utilizó una transmisión serie, sincronizada por reloj, para transmitir el estado de los pines bit por bit. Analizando la figura 230(a), se aprecia la transmisión (línea CMD) del comando uno (01_{16}), empezando desde el bit menos significativo y en un flanco de bajada (CLK), hasta llegar al bit más significativo.

3.10.4.2 Recepción de Datos del "Pad" (DATA)

Al mismo tiempo que se está transmitiendo, luego de cierto tiempo que el reloj está en estado bajo, se envía el. bit de información (línea DATA), de esta forma se aprovecha el tiempo de envío para recibir los interruptores presionados.

3.10.4.3 Señal de Reconocimiento (ACK)

Cada vez que se envíe una trama o grupo de ocho bits, al final de la trama, el pad, dará una señal en bajo, por un corto período de tiempo, que simbolizará que reconoció el comando. Finalizada el grupo de tramas, dependiendo del tipo de pad habrá una señal *NACK, es decir, no habrá un estado bajo*.

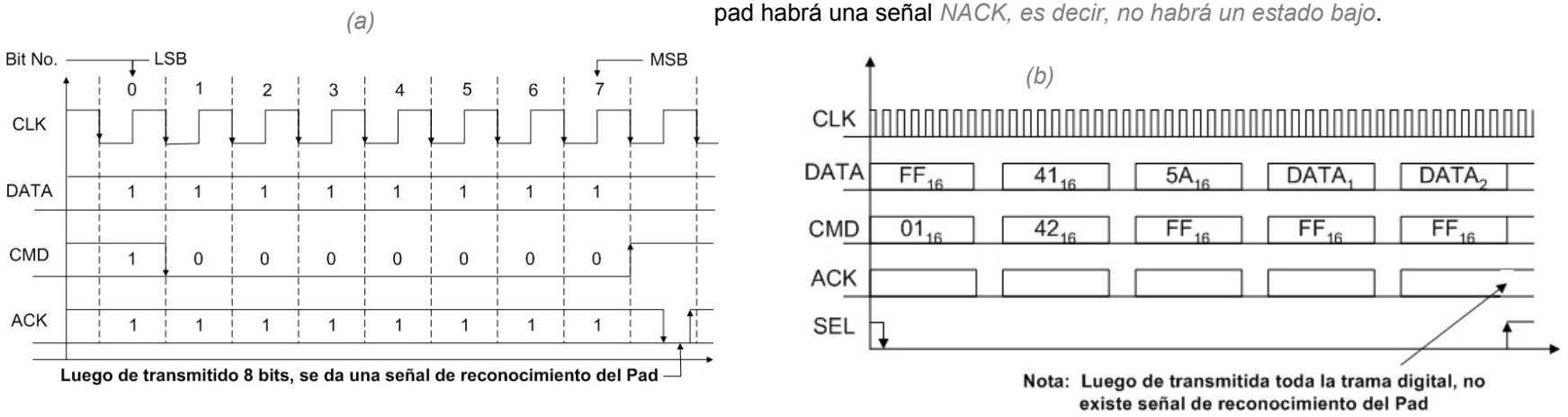


Figura 230. Señales del "Pad" de "Playstation". (a) Transmisión/Recepción del Comando/Dato. Se ejemplifica el estado de la transmisión serie sincronizada por reloj, bit por bit, y su respectiva señal de reconocimiento. Dependiendo del "pad", se enviarán: 5 tramas para un pad digital, y 9 tramas para un "pad" analógico. (b) Transmisión/Recepción del Grupo de Comandos/Datos. Para la condición de un "Pad" digital, esta es la respuesta que se esperará si responde correctamente, $DATA_1$ y $DATA_2$, son el grupo de estados de los botones, presionados o no.

3.10.4.4 Señal de Reloj del Sistema (CLK)

Esta señal es provista por el dispositivo que inicie la comunicación, en este caso, nuestro microcontrolador, y la ventaja de este tipo de transmisión síncrona, es que mientras el reloj no cambie de estado, el bit de datos no será alterado hasta el siguiente flanco.

3.10.4.5 Señal de Selección del "Pad" (SEL)

Antes de iniciar una transmisión/recepción, se debe de bajar la línea SEL, que simboliza que el "pad" de momento está siendo atendido, finalizada todas las tramas, se debe de levantar la línea. Mientras esta línea esté en estado bajo, para un control tipo "dual shock", no se podrá cambiar de modo digital a analógico o viceversa.

3.10.4.6 Tramas Enviadas/Recibidas para/por el "Pad"

Nota: Los botones del "Pad" de "PSOne"" son bajos activos, es decir, al presionarlos, cambia el estado del botón, de alto a bajo.
B[0:7] Representa la data o información recibida del "pad" de "Playstation" en cada trama

Tabla 80. Tramas de un "Pad" Digital

Trama No.	CMD	DATA	B0	B1	B2	B3	B4	B5	B6	B7
1	01_{16}	FF_{16}	1	1	1	1	1	1	1	1
2	42_{16}	41_{16}	1	0	0	0	0	0	1	0
3	FF_{16}	$5A_{16}$	0	1	0	1	1	0	1	0
4	FF_{16}	$DATA_1$	SLT	1	1	STA	↑	→	↓	←
5	FF_{16}	$DATA_2$	L2	R2	L1	R1	△	○	X	□

Tabla 81. Tramas de un "Pad" Analógico.

Trama No.	CMD	DATA	B0	B1	B2	B3	B4	B5	B6	B7
1	01_{16}	FF_{16}	1	1	1	1	1	1	1	1
2	42_{16}	73_{16}	1	1	0	0	1	1	1	0
3	FF_{16}	$5A_{16}$	0	1	0	1	1	0	1	0
4	FF_{16}	$DATA_1$	SLT	JRT	JLT	STA	↑	→	↓	←
5	FF_{16}	$DATA_2$	L2	R2	L1	R1	△	○	X	□
6	FF_{16}	$DATA_3$	JRT.POT1 = 00_{16} = ←				JRT.POT1 = FF_{16} = →			
7	FF_{16}	$DATA_4$	JRT.POT2 = 00_{16} = ↑				JRT.POT2 = FF_{16} = ↓			
8	FF_{16}	$DATA_5$	JLT.POT1 = 00_{16} = ←				JLT.POT1 = FF_{16} = →			
9	FF_{16}	$DATA_6$	JLT.POT2 = 00_{16} = ↑				JLT.POT2 = FF_{16} = ↓			

Joystick en Posición Central = $7F_{16}$

↑ = Arriba
↓ = Abajo
← = Izquierda
→ = Derecha

SLT = Select
STA = Start
L1 = Izq. Superior
○ = Círculo
JRT = Interruptor Derecho del "Joystick"
L2 = Izq. Inferior
R1 = Der. Superior
X = Cruz
JLT = Interruptor Izquierdo del "Joystick"
R2 = Der. Inferior
△ = Triángulo
□ = Cuadrado

JRT.POT1 = Potenciómetro 1 del "Joystick" Derecho (←/→)
JRT.POT2 = Potenciómetro 2 del "Joystick" Derecho (↑/↓)
JLT.POT1 = Potenciómetro 1 del "Joystick" Izquierdo (←/→)
JLT.POT2 = Potenciómetro 2 del "Joystick" Izquierdo (↑/↓)

3.10.5 Esquemático de Aplicación

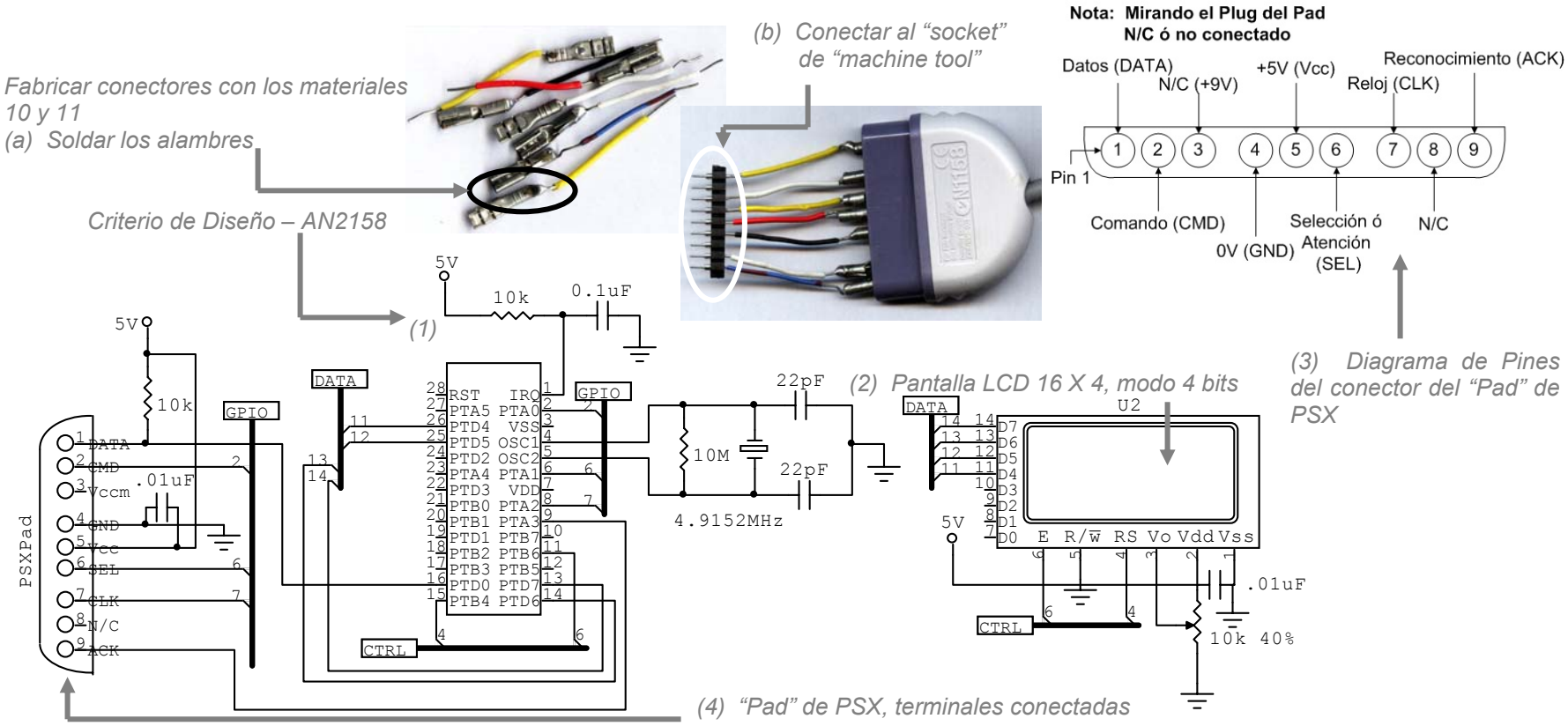
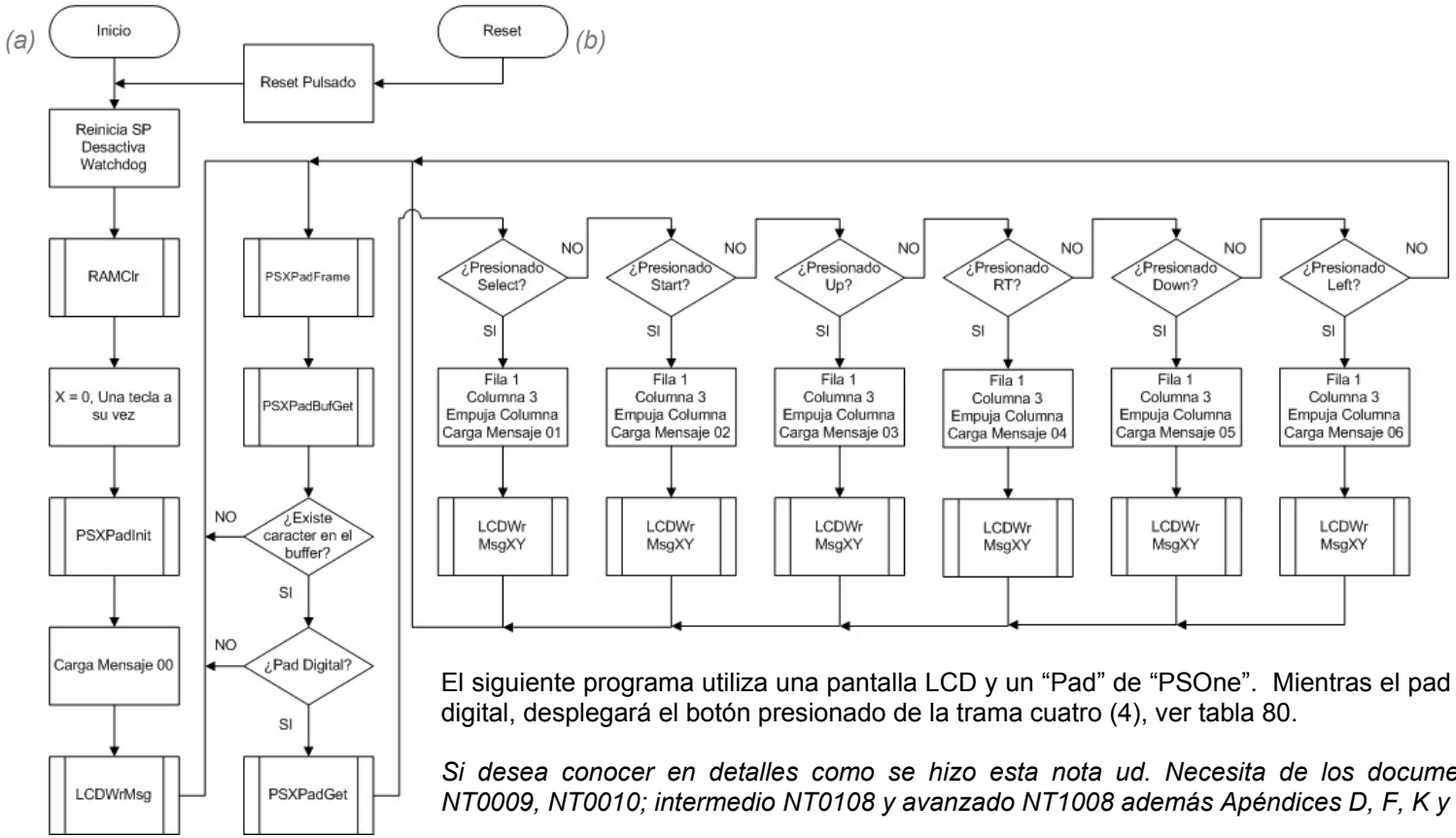


Figura 231. Esquema de Aplicación del "Pad" de PSX. Para conectar el "Pad" al "protoboard" ud. necesitará los terminales para soldar los conectores y fabricar un "socket" fijo con terminales "machine tools" y así llevarlos a la plantilla de desarrollo. El diagrama muestra también una pantalla de 16 X 4 en modo de 4 bits.

3.10.6 Diagrama de Flujo



El siguiente programa utiliza una pantalla LCD y un “Pad” de “PSOne”. Mientras el pad esté en modo digital, desplegará el botón presionado de la trama cuatro (4), ver tabla 80.

Si desea conocer en detalles como se hizo esta nota ud. Necesita de los documentos básicos, NT0009, NT0010; intermedio NT0108 y avanzado NT1008 además Apéndices D, F, K y O.

Figura 232. NT1010 – PSX. (a) Programa Principal. El programa despliega en la pantalla LCD el botón presionado del “pad” de “PSOne”, dependiendo del botón presionado de la trama 4, esto sí y solo si, el “pad” está en modo digital. (b) Señal de Reinicio del sistema. Al presionar, reinicia el sistema sin importar lo que se esté haciendo.

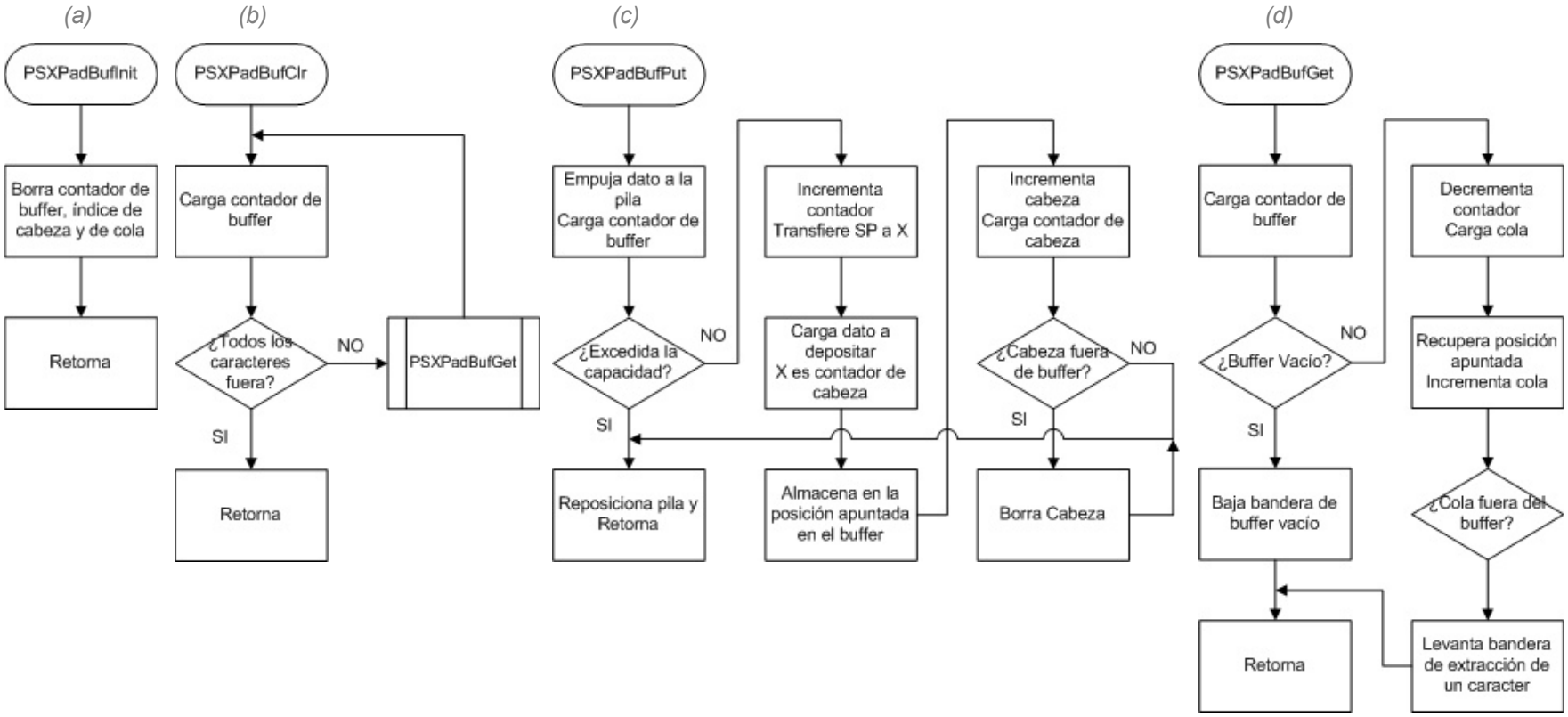


Figura 233. NT1010 – LCD – Subrutinas. (a) PSXPadBufInit. Inicializa el "buffer", índice de cabeza, cola y contador iniciados en cero. (b) PSXPadBufClrl. Saca todos los caracteres del "buffer" si existe alguno insertado. (c) PSXPadBufPut. Deposita en el "buffer" un caracter si hay espacio. (d) PSXPadBufGet. Recoge un caracter del "buffer" si y solo si existe un carácter insertado en el "buffer".

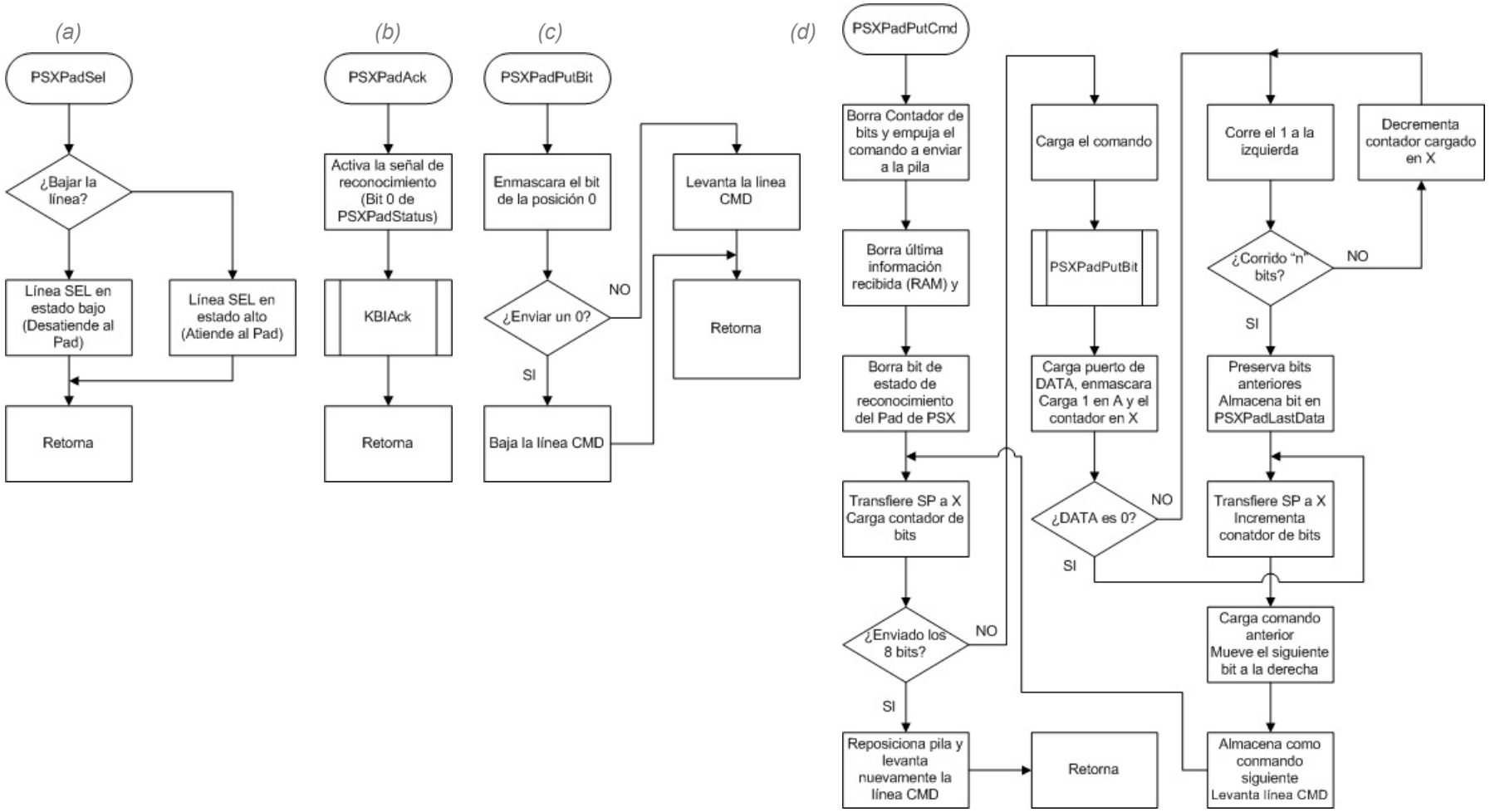


Figura 234. NT1010 – PSX – Subrutinas – Continuación. (a) PSXPadSel. Selecciona si se desea hablar o ignorar el "pad". (b) PSXPadAck. Activa la bandera de reconocimiento de trama adquirida. (c) PSXPadPutBit. Envía un bit de información al "Pad" de PSX. (d) PSXPadCmd. Envía una trama de comandos y deposita el comando recibido en la variable PSXPadLastData.

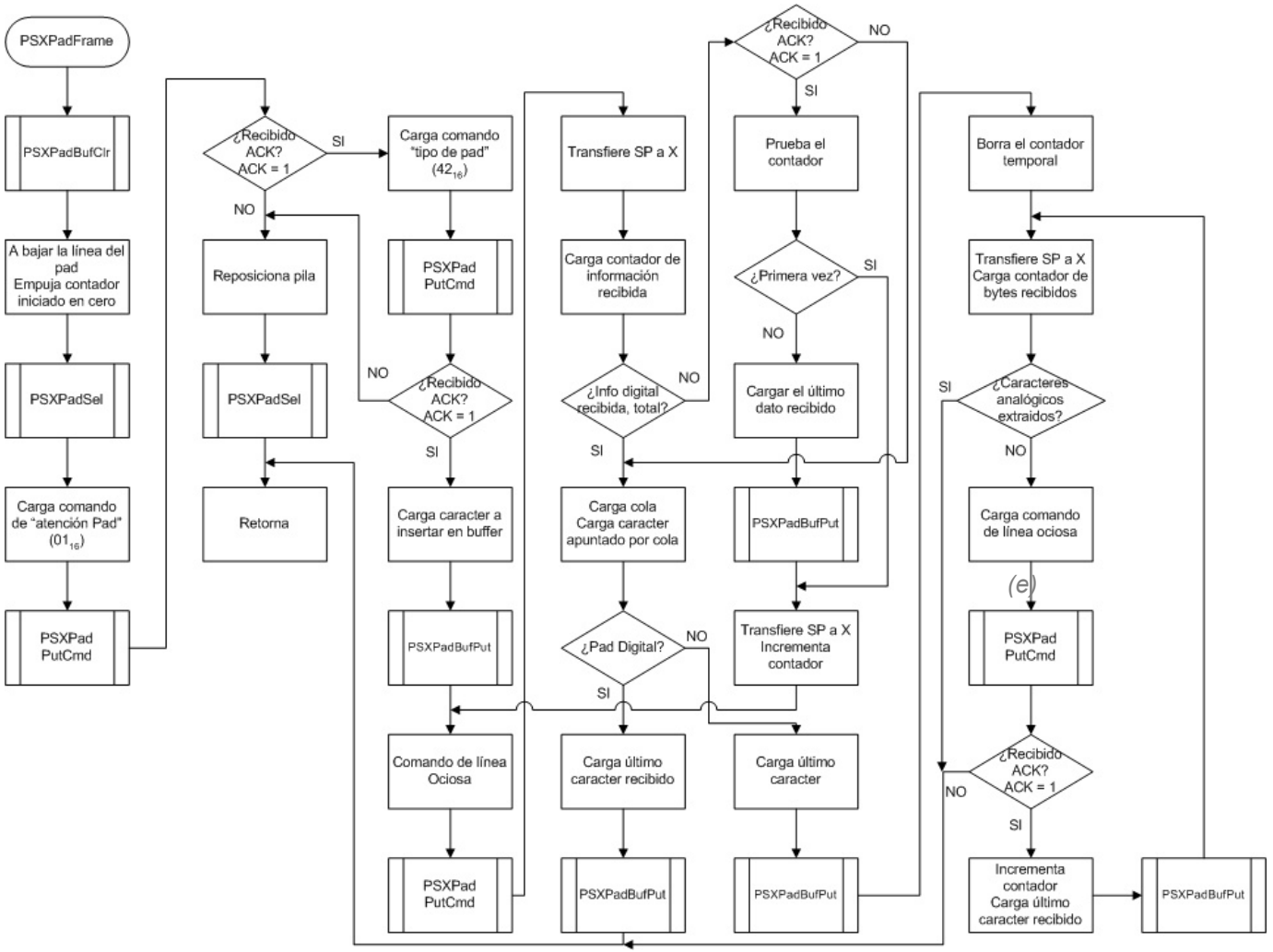


Figura 235. NT1010 – PSX – Subrutinas - Continuación. PSXFrame. Envía las tramas digitales o analógicas, dependiendo del tipo de "pad" seleccionado y deposita el estado de las tramas en un lugar de la memoria de siete (7) posiciones, PSXPadFifo.

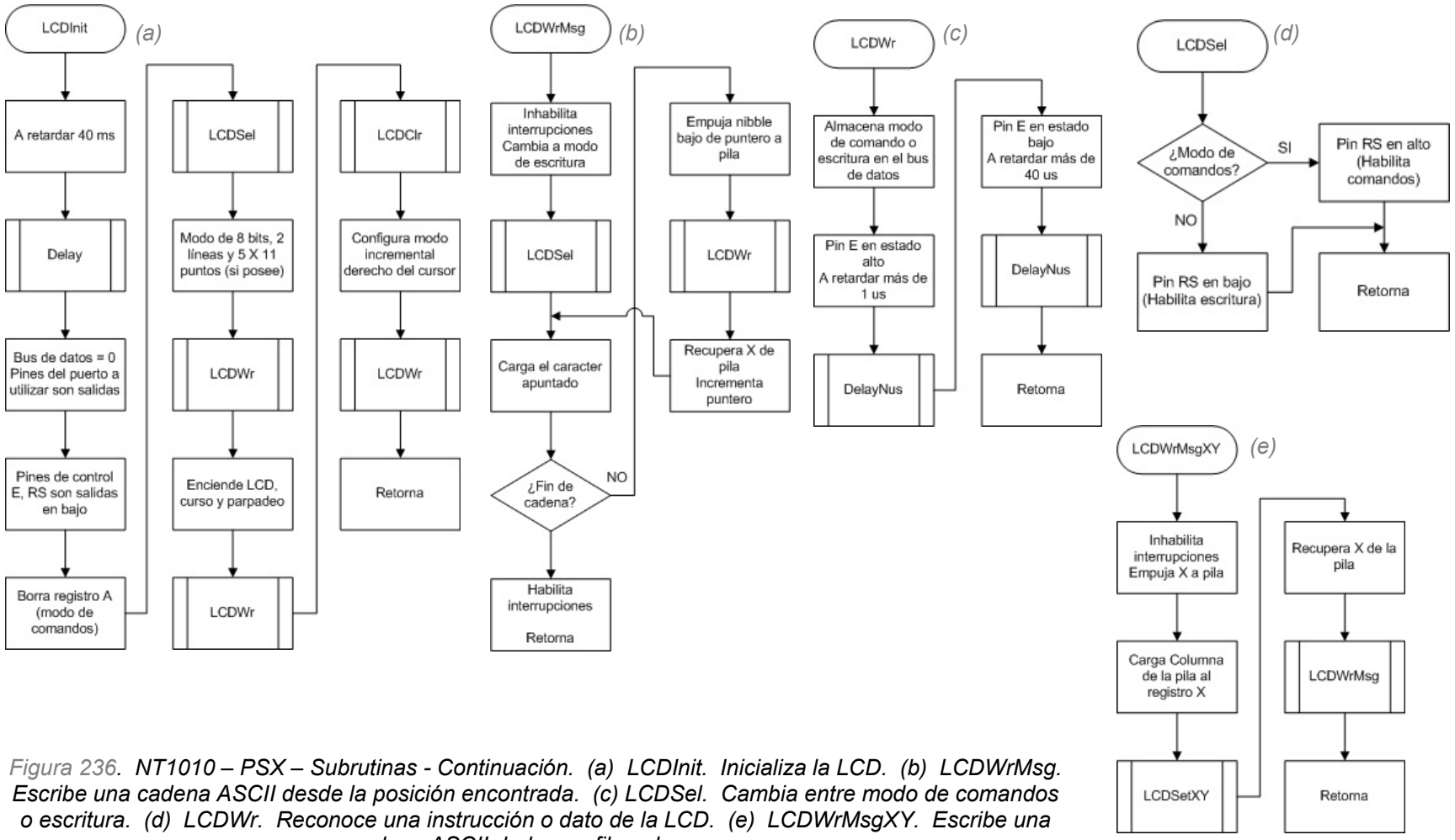


Figura 236. NT1010 – PSX – Subrutinas - Continuación. (a) LCDInit. Inicializa la LCD. (b) LCDWrMsg. Escribe una cadena ASCII desde la posición encontrada. (c) LCDSel. Cambia entre modo de comandos o escritura. (d) LCDWr. Reconoce una instrucción o dato de la LCD. (e) LCDWrMsgXY. Escribe una cadena ASCII dada una fila-columna.

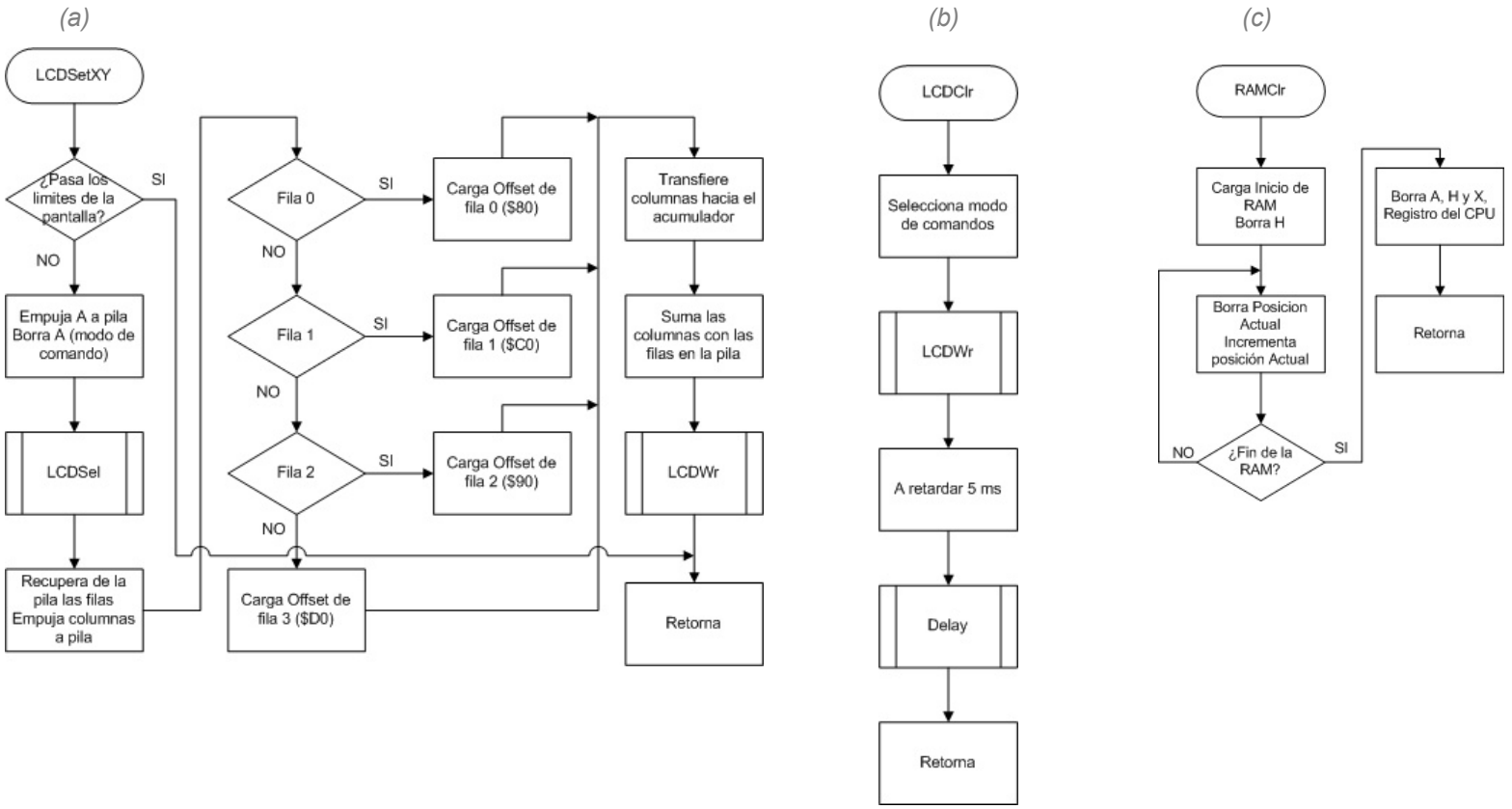


Figura 237. NT1010 – PSX – Subrutinas - Continuación. (a) LCDSetXY. Posiciona la pantalla LCD en la fila-columna. (b) LCDClr. Borra la pantalla LCD. (c) RAMClr. Borra la memoria RAM y registros del CPU.

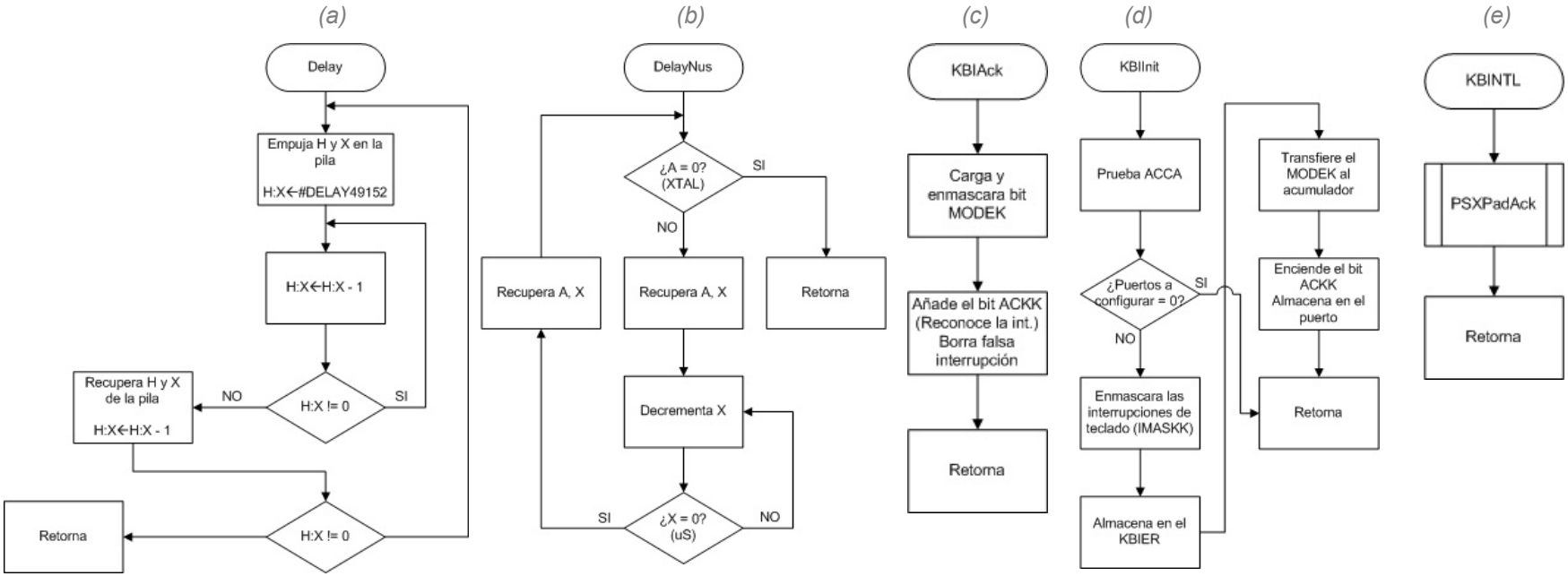


Figura 238. NT1010 – PSX – Subrutinas - Continuación. (a) Delay. Rutina de retardo de "software" programable de 0 a 65.535 segundos. (b) DelayNus. Rutina de retardo de tiempo programable en microsegundos. (c) KBIAck. Reconoce la interrupción del módulo KBI del microcontrolador. (d) KBIInit. Inicializa el módulo KBI dado los puertos de interrupción de teclado y el modo de detección. (e) KBINTL. Rutina de interrupción del teclado del microcontrolador. Reconoce la interrupción del "pad" de playstation y activa una bandera.

3.10.7 Código

Si desea conocer en detalles como se hizo esta nota ud. Necesita de los documentos básicos, NT0009, NT0010; intermedio NT0108; avanzado NT1008 y Apéndices D, F, K y O.

```
;=====================================================================
; ARCHIVO      : NT1010 - PSX - 19 12 04.asm
; PROPÓSITO    : Adquisición de señales del Pad de PSX.
;                El programa inicializa el Temporizador, LCD, y Pad de
;                PSX para decidir que tecla "standard" del pad fue pre-
;                sionada y desplegarla por la pantalla.
;                El programa solo reconocerá solo si es un Pad Digital
;                si se enciende el modo analógico, no visualizará los
;                datos.
;
;
; NOTAS        : La librería LCD utiliza rutinas de la librería DELAY
;                y la pantalla tiene retardos con cristal de 4.9152 MHz
;                La librería PSX utiliza rutinas de la librería KBI.
;
;
;
; REFERENCIA:  NT1010 - PSX - 11 12 04.doc
;
; LENGUAJE     : IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   26   05   03   Creado.
;   24   12   04   Modificado.
;=====================================================================

;=============================================
;            Cabecera de Macros, Const. y Memoria
;=============================================
$include '\MAP\includes.equ'              ; Definiciones de usuario y mapa de memoria

;=============================================
; OBJETIVO     : Inicio de Codif. del Ensam-
;                blador en Memoria FLASH.
;=============================================
         org FLASH_START                  ; Inicio Mem. FLASH
```

```
;===========================================
; OBJETIVO     : Despliega en una Pantalla LCD
;                la tecla presionada estándar
;                si y solo si el pad está
;                configurado como digital.
;                Si está en modo analógico, no
;                muestra la tecla estándar
;                presionada.
;===========================================
START
        rsp                             ; Inic.Stack = $00ff
        bset BIT0,CONFIG1               ; Desactiva Watchdog
        jsr RAMClr                      ; Borra RAM y registros
        jsr LCDInit                     ; Inicializa la pantalla LCD
        clrx                            ; Una tecla a su vez
        jsr PSXPadInit                  ; Inicia Pad de PSX
        ldhx #LCDMESSAGE00              ; Mensaje 0
        jsr LCDWrMsg                    ; Escribe mensaje
PSXPADWAIT1010.AA
        jsr PSXPadFrame                 ; Recibe la trama
        jsr PSXPadBufGet                ; Recoge ID del buffer si existe
        tstx                            ; Compara si existe caracter
        beq PSXPADWAIT1010.AA           ; Seguir recogiendo si no hay caracteres
        cmp #PSX_DIG                    ; ¿Digital?
        bne PSXPADWAIT1010.AA           ; ¿No es digital?, NO, ir a extraer un caracter
        jsr PSXPadBufGet                ; SI, Recoge las teclas estándar
        cmp #PAD_SELECT                 ; SI, ¿Select?
        beq PSXSELECT1010.AB            ; SI, Ir a escribir el mensaje
        cmp #PAD_START                  ; NO, ¿Start?
        beq PSXSTART1010.AC             ; SI, Ir a escribir el mensaje
        cmp #PAD_UP                     ; NO, ¿Arriba?
        beq PSXUP1010.AD                ; SI, Ir a escribir el mensaje
        cmp #PAD_RT                     ; NO, ¿Derecha?
        beq PSXRT1010.AE                ; SI, Ir a escribir el mensaje
        cmp #PAD_DN                     ; NO, ¿Abajo?
        beq PSXDN1010.AF                ; SI, Ir a escribir el mensaje
        cmp #PAD_LT                     ; NO, ¿Izquierda?
        beq PSXLT1010.AG                ; SI, Ir a escribir el mensaje
        bra PSXPADWAIT1010.AA           ; NO, Cualquier otro valor, ir a extraer del
                                        ; buffer
PSXSELECT1010.AB
        lda #1T                         ; Fila 1
        ldx #3T                         ; Columna 3
        pshx                            ; Empuja Columna
        ldhx #LCDMESSAGE01              ; Carga Puntero de Mensaje
        jsr LCDWrMsgXY                  ; Escribe mensaje
        pulx                            ; Reposiciona pila
        bra PSXPADWAIT1010.AA           ; Ir a preguntar nuevamente el tipo de pad
PSXSTART1010.AC
        lda #1T                         ; Fila 1
        ldx #3T                         ; Columna 3
        pshx                            ; Empuja Columna
```

```
        ldhx #LCDMESSAGE02      ; Carga Puntero de Mensaje
        jsr LCDWrMsgXY          ; Escribe mensaje
        pulx                    ; Reposiciona pila
        bra PSXPADWAIT1010.AA   ; Ir a preguntar nuevamente el tipo de pad
PSXUP1010.AD   lda #1T          ; Fila 1
        ldx #3T                 ; Columna 3
        pshx                    ; Empuja columna
        ldhx #LCDMESSAGE03      ; Carga Puntero de Mensaje
        jsr LCDWrMsgXY          ; Escribe mensaje
        pulx                    ; Reposiciona pila
        bra PSXPADWAIT1010.AA   ; Ir a preguntar nuevamente el tipo de pad
PSXRT1010.AE   lda #1T          ; Fila 1
        ldx #3T                 ; Columna 3
        pshx                    ; Empuja columna
        ldhx #LCDMESSAGE04      ; Carga Puntero de Mensaje
        jsr LCDWrMsgXY          ; Escribe Mensaje
        pulx                    ; Reposiciona pila
        bra PSXPADWAIT1010.AA   ; Ir a preguntar nuevamente el tipo de pad
PSXDN1010.AF   lda #1T          ; Fila 1
        ldx #3T                 ; Columna 3
        pshx                    ; Empuja columna
        ldhx #LCDMESSAGE05      ; Carga puntero de Mensaje
        jsr LCDWrMsgXY          ; Escribe Mensaje
        pulx                    ; Reposiciona pila
        bra PSXPADWAIT1010.AA   ; Ir a preguntar nuevamente el tipo de pad
PSXLT1010.AG   lda #1T          ; Fila 1
        ldx #3T                 ; Columna 3
        pshx                    ; Empuja columna
        ldhx #LCDMESSAGE06      ; Carga puntero de Mensaje
        jsr LCDWrMsgXY          ; Escribe mensaje
        pulx                    ; Reposiciona pila
        jmp PSXPADWAIT1010.AA   ; Ir a preguntar nuevamente el tipo de pad

;=============================================
;   Declaración y definición de funciones
;=============================================
$include '\FUNCTIONS\INCLUDES.inc'          ; Incluye Funciones

;=============================================
;  Declaración y Definición de interrupciones
;=============================================
$include '\INTERRUPTS\interrupt.inc'        ; Incluye interrupciones del
microcontrolador
```

Listado 72. NT1010 – PSX. Inicializa el control y "pad" de PSX para desplegar en pantalla el texto del botón que fue presionado. Todo esto se cumple si y solo si el "pad" es digital, de lo contrario no desplegará la información.

```
;=================================================================
; ARCHIVO      : PSX.inc
; PROPÓSITO    : Librería para utilidad de Pad de Playstation
; LENGUAJE     : IN-LINE ASSEMBLER
; NOTAS        : LA LIBRERÍA PSX.INC UTILIZA LA LIBRERÍA KBI.INC
; ---------------------------------------------------------------------
; HISTORIAL
;  DD MM AA
;  18  12  04   Creado.
;  18  12  04   Modificado.
;=================================================================
; INICIALIZACIÓN
;                : Configurar...
;                1 - Puerto General E/S (CMD, CLK, SEL):  PSX_PAD_GPIO_PORT
;                2 - Registro General de E/S: PSX_PAD_GPIO_DDR
;                        2.1 - Bit de Comandos del Pad de PSX:  PSX_PAD_CMD_BIT
;                        2.2 - Bit de Selección del Pad de PSX:  PSX_PAD_SEL_BIT
;                        2.3 - Bit de Reloj del Sistema:  PSX_PAD_CLK_BIT
;                3 - Bit del Puerto de señal de reconoc: PSX_PAD_ACK_BIT
;                4 - Puerto de Lectura de Datos del Pad:  PSX_PAD_DATA_PORT
;                5 - Registro de Dir. de Lectura del Pad:  PSX_PAD_DATA_DDR
;                        5.1 - Bit de Datos del Pad:  PSX_PAD_DATA_BIT
;
;                6 - Dir. Registro de Estado y Control del Pad:  PSXPadStatus
;                        6.1 - Bandera Bit de Reconocimiento:  PSX_PAD_STAT_ACK
;                Nota:  La variable de última Tecla, PSXPadLastData
;                estará 1 posición después de la variable
;                PSXPadStatus.
;                7 - Contador de Caracteres del Buffer:  PSXPadFifoCtr
;                Nota:  PSXPadFifoHd (Cabeza del Buffer)
;                PSXPadFifoTl (Cola del Buffer)
;                PSXPadFifo (Posición inicial del buffer)
;                Son variables consecutivas (estructura) defini
;                das por la dirección de PSXPadFifoCtr
;
;
; NOTA           :
;                1 - El Bit PSX_PAD_ACK_BIT, NECESARIAMENTE debe pertenecer
;                al puerto de interrupción por teclado.
;                2 - Por el momento, solo se tiene la rutina para interrup-
;                ciones para microcontroladores con puerto de teclado.
;                Modificaciones futuras incluirán configuración para
;                para microcontroladores sin este módulo.
;
;                        MIRANDO AL CONECTOR
;                        ___________________
;           Pin 1------> | o  o  o   o  o  o   o  o  o | <--------Pin 9
;                        \___________________________/
;                        N/C = No conectado
;                        DATA - Pin 1
;                        CMD  - Pin 2
;                        N/C  - Pin 3
```

```
;                       GND  - Pin 4
;                       VCC  - Pin 5
;                       ATT  - Pin 6
;                       CLK  - Pin 7
;                       N/C  - Pin 8
;                       ACK  - Pin 9
;
;=================================================================

;=========================================
;                 Constantes & Macros
;=========================================
;=========================================
;      DEFINA PUERTO DE CONTROL DEL PAD PSX
;=========================================
PSX_PAD_GPIO_PORT       equ PORTA     ; Puerto General de E/S para el Pad
PSX_PAD_GPIO_DDR        equ DDRA      ; Registro General de E/S para el Pad
PSX_PAD_CMD_BIT         equ BIT0      ; GPIO.Bit0, CMD
PSX_PAD_SEL_BIT         equ BIT1      ; GPIO.Bit1, SEL
PSX_PAD_CLK_BIT         equ BIT2      ; GPIO.Bit2, CLK

;=========================================
; DEFINA EL BIT DE LA SEÑAL DE RECONOCIMIENTO
;=========================================
PSX_PAD_ACK_BIT         equ 3         ; Bit del Puerto de señal de reconoc.

;=========================================
;   DEFINA EL PUERTO DE DATOS DEL PAD DE PSX
;=========================================
PSX_PAD_DATA_PORT       equ PORTD     ; Puerto de Lectura de datos del Pad
PSX_PAD_DATA_DDR        equ DDRD      ; Registro de Selección del Pad
PSX_PAD_DATA_BIT        equ 0T        ; Bit del Puerto de Selección del Pad

;=========================================
; DEFINA EL BYTE DE CONTROL DE ESTADO DEL PAD
;=========================================
PSXPadStatus            equ $80       ; Estado del Pad, ACK.B0
PSX_PAD_STAT_ACK        equ 0T        ; Bit de Reconocimiento del Pad

PSXPadLastData          equ PSXPadStatus+1
                                      ; Última información del Pad
```

```
;=============================================
; DEFINA TAMAÑO Y UBICACIÓN DEL BUFFER DE PAD
;=============================================
PSX_PAD_FIFO_SIZE       equ 7           ; Caracteres máximos dig. + analógic.
PSXPadFifoCtr           equ $A1         ; Contador de bytes del buffer
PSXPadFifoHd            equ PSXPadFifoCtr+1
                                        ; Cabeza del Buffer
PSXPadFifoTl            equ PSXPadFifoCtr+2
                                        ; Cola del Buffer
PSXPadFifo              equ PSXPadFifoCtr+3
                                        ; Inicio del Buffer

PSX_DIG                 equ $41         ; Identificador de PAD Digital
PSX_ALG                 equ $73         ; Identificador de PAD Analógico

PSX_PAD_CMD_PAD         equ 1T          ; Comando = Pad
PSX_PAD_CMD_ID          equ $42         ; Comando = ID o tipo de Pad
PSX_PAD_CMD_WAIT        equ $FF         ; Comando = Línea ociosa

;=============================================
; PSXPADBUFINIT
;               : Inicializa el buffer anillo
; OBJETIVO      : Cabeza y cola en posición i-
;                 nicial; contador en cero.
; ENTRADA       : Ninguna
; SALIDA        : Ninguna
; REGISTROS
; AFECTADOS     : PSXPadFifo (Estructura RAM)
;=============================================
PSXPadBufInit
        clr PSXPadFifoCtr       ; contador buffer, ...
        clr PSXPadFifoHd        ; ... cabeza y ...
        clr PSXPadFifoTl        ; cola iniciados.
        rts                     ; Retorna
```

```
;=========================================
; PSXPADBUFPUT
;                 : Impone 1 byte en el buffer
; OBJETIVO        : Arroja 1 byte en el buffer
; ENTRADA         : A    es el dato a enviar al
;                 buffer
; SALIDA          : Ninguna
; REGISTROS
; AFECTADOS       : ACCA, H:X, CCR
;                   PSXPadFifo (Estructura - RAM)
;=========================================
PSXPadBufPut
        psha                        ; Empuja dato a la pila
        lda PSXPadFifoCtr           ; Carga contador
        cmp #PSX_PAD_FIFO_SIZE      ; Capacidad máxima a comparar
        bge PSXPADOUT1010.A         ; ¿Hay espacio en el buffer?, NO, Salir
        inc PSXPadFifoCtr           ; SI, incrementa contador
        tsx                         ; Transfiere SP a X
        lda 0,x                     ; Carga dato a depositar
        ldx PSXPadFifoHd            ; Carga contador de cabeza
        pshh                        ; Empuja H a pila
        clrh                        ; Borra H
        sta PSXPadFifo,x            ; Almacena en la posición apuntada
        pulh                        ; Recupera H
        inc PSXPadFifoHd            ; Actualiza cabeza
        lda PSXPadFifoHd            ; Carga contador de cabeza
        cmp #PSX_PAD_FIFO_SIZE      ; Compara con la capacidad máxima
        bne PSXPADOUT1010.A         ; NO, no son iguales, salir
        clr PSXPadFifoHd            ; SI, Cabeza en posición inicial
PSXPADOUT1010.A
        pula                        ; Reposiciona pila
        rts                         ; retorna
```

```
;===========================================
; PSXPADBUFGET
;                : Recoge 1 byte en el buffer
; OBJETIVO       : Recoge 1 caracter existente
;                  en el buffer
; ENTRADA        : Ninguna
; SALIDA         : X = 0 si no se recupera
;                  del buffer... ó
;                  X = 1 si se extrajo un carac
;                  ter del buffer
;                  A     contiene el caracter
;                  recuperado del buffer
; REGISTROS
; AFECTADOS      :  ACCA, H:X, PSXPadFifo (stru)
;===========================================
PSXPadBufGet
        lda PSXPadFifoCtr          ; Verifica si hay un nuevo dato en el buffer
        ble PSXPADOUT1010.B        ; Si está vacío, salir
        dec PSXPadFifoCtr          ; decrementa contador
        ldx PSXPadFifoTl           ; Carga contador de cola
        pshh                       ; Empuja H a pila
        clrh                       ; Borra H
        lda PSXPadFifo,x           ; Recupera de la posición apuntada
        pulh                       ; Recupera H
        inc PSXPadFifoTl           ; Actualiza cola
        ldx PSXPadFifoTl           ; Carga cola
        cpx #PSX_PAD_FIFO_SIZE     ; Compara con la capacidad máxima
        bne PSXPADOUT1010.C        ; NO, no son iguales, salir
        clr PSXPadFifoTl           ; SI, Apunta cola al inicio nuevamente
PSXPADOUT1010.C
        ldx #1T                    ; Bandera de extracción del caracter
        bra PSXPADOUT1010.D        ; Salir
PSXPADOUT1010.B
        clrx                       ; Levanta bandera
PSXPADOUT1010.D
        rts                        ; retorna
```

```
;===========================================
; PSXPADBUFCLR
;                 : Remueve todo el buffer
; OBJETIVO        : Remueve caracteres insertados
;                   del buffer.
; ENTRADA         : Ninguna
; SALIDA          : Ninguna
; REGISTROS
; AFECTADOS       : ACCA, PSXPadFifo (Estructura)
;===========================================
PSXPadBufClr
        lda PSXPadFifoCtr           ; Carga el contador
        beq PSXPADOUT1010.E         ; ¿Todos los caracteres fuera?, SI, Salir
        jsr PSXPadBufGet            ; NO, Recoger del buffer
        bra PSXPadBufClr            ; Siguiente extracción
PSXPADOUT1010.E
        rts                         ; Retornar

;===========================================
; PSXPADINIT      : Inicializa el Pad de PSX
; OBJETIVO        : Inicializa el Pad
; ENTRADA         : A  contiene el modo del Pad
; SALIDA          : Ninguna
; REGISTROS
; AFECTADOS       : Ninguno
;===========================================
PSXPadInit
        lda #{(1 < PSX_PAD_CMD_BIT)|(1 < PSX_PAD_SEL_BIT)|(1
            <PSX_PAD_CLK_BIT)}
                                    ; Carga bits a configurar
        ora PSX_PAD_GPIO_PORT       ; Preserva bits
        sta PSX_PAD_GPIO_PORT       ; Almacena en el puerto
        ora PSX_PAD_GPIO_DDR        ; Preserva bits
        sta PSX_PAD_GPIO_DDR        ; Salidas
        lda #{1 < PSX_PAD_DATA_BIT}
                                    ; Carga  puerto de entrada
        coma                        ; Conmuta
        and PSX_PAD_DATA_DDR        ; Preserva bits
        sta PSX_PAD_DATA_DDR        ; Almacena en el puerto
        lda #{1 < PSX_PAD_ACK_BIT}  ; Carga bit ACK
        jsr KBIInit                 ; Inicializa módulo KBI
        jsr PSXPadBufInit           ; Inicializa buffer anillo
        cli                         ; Habilita interrupciones
        rts                         ; Retorna
```

```
;=============================================
; PSXPADSEL  :  Atiende al Pad de momento
; OBJETIVO   :  Baja o sube la línea de
;               atención del Pad.
; ENTRADA    :  A   es el estado lógico para
;               atención del Pad
;               A = 1, línea en estado alto
;               A = 0, línea en estado bajo
; SALIDA     :  Ninguna
; REGISTROS
; AFECTADOS  :  Ninguno
;=============================================
PSXPadSel
        bne PSXPADSEL1010.F         ; Ir a desatender el pad
        bclr PSX_PAD_SEL_BIT,PSX_PAD_GPIO_PORT
                                    ; Atiende al Pad
        bra PSXPADOUT1010.G         ; Salir
PSXPADSEL1010.F
        bset PSX_PAD_SEL_BIT,PSX_PAD_GPIO_PORT
                                    ; Desatiende el Pad
PSXPADOUT1010.G
        rts                         ; Retorna

;=============================================
; PSXPADACK  :  Levanta la bandera de recono-
;               cimiento.
; OBJETIVO   :  Avisa que ha reconocido un co
;               mando.
; ENTRADA    :  Ninguna
; SALIDA     :  Ninguna
; REGISTROS
; AFECTADOS  :  PSXPadStatus.B0 = ACK (RAM)
;=============================================
PSXPadAck
        bset PSX_PAD_STAT_ACK,PSXPadStatus
                                    ; Activa la señal de reconocimiento
        jsr KBIAck                  ; Reconoce la interrupción
        rts                         ; Retorna
```

```
;===========================================
; PSXPADPUTBIT
;                 : Envía un bit de comando
; OBJETIVO        : Emascara y envía bit
; ENTRADA         : A     contiene el comando a
;                   enviar.
; SALIDA          : Ninguna
; REGISTROS
; AFECTADOS       : ACCA, PSX_PAD_GPIO_PORT.CMD
;===========================================
PSXPadPutBit
        and #1T                         ; Enmascara el bit
        beq PSXPADSET1010.N             ; ¿0?, SI, Enviar un 0
        bset PSX_PAD_CMD_BIT,PSX_PAD_GPIO_PORT
                                        ; NO, Levanta la línea CMD
        bra PSXPADOUT1010.O             ; Salir
PSXPADSET1010.N
        bclr PSX_PAD_CMD_BIT,PSX_PAD_GPIO_PORT
                                        ; Baja la línea CMD
PSXPADOUT1010.O
        rts                             ; Retorna
```

```
;===========================================
; PSXPADPUTCMD
;                : Envía un comando al Pad
; OBJETIVO       : Envía el comando
; ENTRADA        : A     es el comando a enviar
; SALIDA         : Ninguna
; REGISTROS
; AFECTADOS      : ACCA, H:X, SP, GPIO_PORT,
;                  PSXPadLastData (RAM)
;===========================================
PSXPadPutCmd
        clrx                            ; Borra contador de bits
        psha                            ; Empuja comando a la pila
        pshx                            ; Empuja contador de bits a pila
        clr PSXPadLastData              ; Borra la última información
        bclr PSX_PAD_STAT_ACK,PSXPadStatus
                                        ; Borra bit de reconocimiento
PSXPADCLKLO1010.H
        bclr PSX_PAD_CLK_BIT,PSX_PAD_GPIO_PORT
                                        ; Baja solo CLK
        tsx                             ; Transfiere SP a X
        lda 0,x                         ; Carga contador
        cbeqa #8T,PSXPADOUT1010.K
                                        ; ¿Enviado 8 bits?, SI, Salir
        lda 1,x                         ; NO, Carga comando
        jsr PSXPadPutBit                ; Envía bit
        lda PSX_PAD_DATA_PORT           ; Carga puerto de Datos
        and #{1 < PSX_PAD_DATA_BIT}
                                        ; Enmascara bit
        beq PSXPADZERO1010.J            ; ¿0?, SI, subir el reloj
        lda #1T                         ; Carga 1
        ldx 0,x                         ; Carga contador
        beq PSXPADSET1010.L             ; ¿Primera vez?, Primer bit en 1
PSXPADSET1010.M
        lsla                            ; Corre el 1 a la izquierda
        dbnzx PSXPADSET1010.M           ; Salta hasta posicionar el 1 correctamente
PSXPADSET1010.L
        ora PSXPadLastData              ; Preserva bits anteriores
        sta PSXPadLastData              ; Almacena siguiente bit
PSXPADZERO1010.J
        tsx                             ; Transfiere SP a X
        inc 0,x                         ; Incrementa contador
        lda 1,x                         ; Carga el comando
        asra                            ; mueve el siguiente bit a la derecha
        sta 1,x                         ; Almacena como comando
PSXPADCLKHI1010.I
        bset PSX_PAD_CLK_BIT,PSX_PAD_GPIO_PORT
                                        ; Levanta CLK
        bra PSXPADCLKLO1010.H           ; Siguiente bit a enviar
PSXPADOUT1010.K ais #2T                 ; Reposiciona la pila
        bset PSX_PAD_CMD_BIT,PSX_PAD_GPIO_PORT
                                        ; Levanta nuevamente la línea CMD
        rts                             ; Retorna
```

```
;=========================================
; PSXPADFRAME
;                : Envía un comando al Pad
; OBJETIVO       : Envía el comando
; ENTRADA        : Ninguna
; SALIDA         : Ninguna
; REGISTROS
; AFECTADOS      : ACCA, PSXPadLastData (RAM),
;                  PSXPadStatus (RAM)
;=========================================
PSXPadFrame
        jsr PSXPadBufClr            ; Vacía el buffer
        clra                        ; A seleccionar el pad
        psha                        ; Aprovecha y utiliza a para empujar el
                                    ; contador iniciado en 0.
        jsr PSXPadSel               ; Selecciona el pad
        lda #PSX_PAD_CMD_PAD        ; Comando de preguntar Pad
        jsr PSXPadPutCmd            ; Pregunta si es un pad
        brclr PSX_PAD_STAT_ACK,PSXPadStatus,PSXPADOUT1010.P
                                    ; ¿No hubo respuesta del pad?, SI, Salir
        lda #PSX_PAD_CMD_ID         ; NO, Pregunta el tipo de Pad
        jsr PSXPadPutCmd            ; Envía comando
        brclr PSX_PAD_STAT_ACK,PSXPadStatus,PSXPADOUT1010.P
                                    ; ¿No hubo respuesta del pad?, SI, Salir
        lda PSXPadLastData          ; NO, Carga caracter a insertar en el buffer
        jsr PSXPadBufPut            ; Envía al buffer
PSXPADDATA1010.T
        lda #PSX_PAD_CMD_WAIT       ; Comando de línea ociosa
        jsr PSXPadPutCmd            ; Envía comando
        tsx                         ; Transfiere SP a X
        lda 0,x                     ; Carga contador
        cbeqa #2T,PSXPADNEXT1010.Q
                                    ; ¿Transmitidos todos los bytes de Información
                                    ; digital?
        brclr PSX_PAD_STAT_ACK,PSXPadStatus,PSXPADNEXT1010.Q
                                    ; SI, ¿No hubo respuesta del pad?, Salir
        tsta                        ; NO, ¿Primera vez?
        beq PSXPADNEXT1010.S        ; SI, no almacenar
        lda PSXPadLastData          ; NO, cargar dato recibido
        jsr PSXPadBufPut            ; Almacenar en el buffer
PSXPADNEXT1010.S
        tsx                         ; Tranfiere SP a X
        inc 0,x                     ; Incrementa contador
        bra PSXPADDATA1010.T        ; Siguiente información
PSXPADNEXT1010.Q
        ldx PSXPadFifoTl            ; Carga puntero cola
        lda PSXPadFifo,x            ; Carga caracter
        cbeqa #PSX_DIG,PSXPADOUT1010.R
                                    ; ¿Pad es Digital?, SI, Salir
        lda PSXPadLastData          ; Carga último caracter
        jsr PSXPadBufPut            ; Envía al buffer
        tsx                         ; Transfiere SP a X
```

```
        clr 0,x                         ; Borra el contador temporal
PSXPADNEXT1010.U
        tsx                             ; Transfiere SP a X
        lda 0,x                         ; Carga contador
        cbeqa #4T,PSXPADOUT1010.R
                                        ; ¿Todos los datos analógicos extraídos?, SI,
                                        ; Salir
        lda #PSX_PAD_CMD_WAIT           ; Comando de línea ociosa
        jsr PSXPadPutCmd                ; Envía comando
        brclr PSX_PAD_STAT_ACK,PSXPadStatus,PSXPADOUT1010.R
        tsx                             ; Transfiere SP a X
        inc 0,x                         ; Incrementa contador
        lda PSXPadLastData              ; Carga caracter recibido
        jsr PSXPadBufPut                ; Deposita en el buffer
        bra PSXPADNEXT1010.U            ; Siguiente dato analógico
PSXPADOUT1010.R
        lda PSXPadLastData
                                        ; Carga último caracter recibido
        jsr PSXPadBufPut                ; Envía al buffer
PSXPADOUT1010.P
        lda #1                          ; A Deseleccionar Pad
        jsr PSXPadSel                   ; Deselecciona
        pula                            ; Reposiciona pila
        rts                             ; Retorna
```

Listado 73. NT1010 – PSX – PSX.inc. Archivo de funciones del módulo de "Pad" de "Playstation".

```
;=================================================================
; ARCHIVO      : RAM.inc
; PROPÓSITO    : Funciones de uso de la RAM
; NOTAS        : ADVERTENCIA - Se debe especificar el inicio y el origen de
;                RAM de su microcontrolador por medio de los
;                macros RAM_ORG y RAM_END
;
; LENGUAJE     : IN-LINE ASSEMBLER
; -----------------------------------------------------------------------
; HISTORIAL
;  DD MM AA
;  05  10  04   Creado.
;  06  10  04   Modificado.
;=================================================================

;=========================================
;                CONSTANTES & MACROS
;=========================================
RAM_ORG      equ $0080                  ; Inicio de la memoria RAM
RAM_END      equ $00FF-1                ; Fin de limpieza de la RAM

;=========================================
; RAMCLR       : Borra la Ram utilizada y re-
;                gistros inherentes
; OBJETIVO     : Borra registros
; ENTRADA      : Ninguna
; SALIDA       : A, H:X y RAM en 0
; REGISTROS
; AFECTADOS    : RAM, H:X, A
;=========================================
RAMClr                                  ; Borra la RAM y registros
        clrh                            ; Borra H
        ldx #RAM_ORG                    ; Carga con el origen
RAM_EMPTY0000.A
        clr ,x                          ; rellena con "0" la posición actual
        aix #1                          ; incrementa puntero de RAM
        cphx #RAM_END                   ; Compara hasta el final deseado
        bne RAM_EMPTY0000.A             ; Si no concuerda entonces sigue
                                        ; limpiando
        clra                            ; Borra A
        clrh                            ; Borra H
        clrx                            ; Borra X
        rts                             ; retorna
```

Listado 74. NT1010 – PSX – RAM.inc. RAMClr es una rutina que borra la RAM y los registros del CPU.

```
;=====================================================================
; ARCHIVO      :  LCD.inc
; PROPÓSITO    :  Librería para pantalla LCD
; LENGUAJE     :  IN-LINE ASSEMBLER
; NOTAS        :  LA LIBRERÍA LCD.INC UTILIZA LAS FUNCIONES DE LA
;                 LIBRERÍA DE FUNCIONES DE RETARDO DELAY.INC.
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   05   11   04   Creado.
;   11   11   04   Modificado.
;=====================================================================
; INICIALIZACIÓN
;           :  Esta rutina puede utilizarse con cualquier LCD con
;                    driver Hitachi HD44780.
;                    Para utilizar esta rutina usted deberá inicializar
;                    el módulo de la siguiente manera.  Configurar:
;                    1 - Puerto de datos:  LCD_DATA_PORT
;                    2 - Reg. de direccionamiento para datos:  LCD_DATA_DDR
;                    3 - Pines de salida de datos a utilizar:  LCD_DATA_OUTPUT
;                    4 - Puerto de la señal RS:  LCD_RS_PORT
;                    5 - Reg. de direccionamiento de la señal RS:  LCD_RS_DDR
;                    6 - Pin del puerto para señal RS:  LCD_RS_PIN
;                    7 - Puero de la señal E:  LCD_E_PORT
;                    8 - Reg. de direccionamiento de E:  LCD_E_DDR
;                    9 - Pin del puerto para E:  LCD_E_PIN
;
;   0,0
;    +-> (Columna)
;    |          Pantalla LCD de 16 X 4 Caracteres
;    U               (Caracteres Visibles)
; (Fila)
;
;              00  01  02  03  04  05  06  07  08  09  10  11  12  13  14  15
;            +*****************************************************************
;          00* A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P
;          01* Q   R   S   T   U   V   W   X   Y   Z   0   1   2   3   4   5
;          02* 6   7   8   9   a   b   c   d   e   f   g   h   i   j   k   l
;          03* m   n   o   p   q   r   s   t   u   v   w   x   y   z   #   !
;            +*****************************************************************
;
;
;=====================================================================
;$SET            LCD_8BIT_MODE_EN          ; Configura para 8 Bits
$SETNOT         LCD_8BIT_MODE_EN          ; Configura para 4 Bits
```

```
;=========================================
;                Constantes & Macros
;=========================================
;=========================================
;            DEFINA EL PUERTO DE DATOS
;=========================================
LCD_DATA_PORT     equ PORTD          ; Puerto de Salida de la Información
LCD_DATA_DDR      equ DDRD           ; Registro de dir. del Puerto de Salida
                                     ; de la Información
$IF LCD_8BIT_MODE_EN
LCD_DATA_OUTPUT   equ $FF            ; Puertos a Utilizar en el LCD ($FF =
                                     ; Todos son utilizados, modo 8bits)
$ELSEIF
LCD_DATA_OUTPUT   equ $F0            ; Puertos a Utilizar en el LCD ($F0 =
                                     ; Solo nibble alto)
$ENDIF

;=========================================
;  DEFINA PUERTO DE REGISTRO/INSTRUCCIÓN LCD
;=========================================
LCD_RS_PORT       equ PORTB          ; Puerto de Registro/Instrucción del
                                     ; LCD
LCD_RS_DDR        equ DDRB           ; Registro de dir. del Puerto del RS del
                                     ; LCD
LCD_RS_PIN        equ BIT4           ; Pin del Puerto

;=========================================
;  DEFINA PUERTO HABILITADOR DE INSTRUCCIÓN
;=========================================
LCD_E_PORT        equ PORTB          ; Puerto de Habilitador de Instrucción
                                     ; del LCD
LCD_E_DDR         equ DDRB           ; Registro de dir. del Puerto de RS
LCD_E_PIN         equ BIT6           ; Pin del Puerto

;=========================================
; DEFINA MÁXIMO DE FILAS Y COLUMNAS DE SU LCD
;=========================================
LCD_MAX_ROW       equ 4T             ; Máximo de filas
LCD_MAX_COL       equ 16T            ; Máximo de columnas

;=========================================
;              INSTRUCCIONES DEL LCD
;=========================================
LCD_CLR           equ %00000001      ; Borra pantalla y ubica en posición
inicial izq.
LCD_HOME          equ %00000010      ; Regresa a posición inicial
LCD_INC           equ %00000010      ; Incremento de Cursor
LCD_SHIFT         equ %00000001      ; Cambio extremo activado
LCD_ON            equ %00000100      ; Enciende el LCD
LCD_CSR           equ %00000010      ; Enciende el Cursor
LCD_BLINK         equ %00000001      ; Parpadea LCD
LCD_RT            equ %00000100      ; Display Shift Derecho
```

```
LCD_SET         equ %00001000       ; Cursor Encendido
LCD_8BIT        equ %00010000       ; LCD de 8 bit
LCD_2LINE       equ %00001000       ; LCD 2 líneas
LCD_5X11        equ %00000100       ; LCD 5X11 Puntos

LCD_CSR_RT      equ %00010100       ; Mueve el cursor a la derecha
LCD_CSR_LT      equ %00010000       ; Mueve el cursor a la izquierda
LCD_MOVE_RT     equ %00011100       ; Mueve la pantalla a la derecha
LCD_MOVE_LT     equ %00011000       ; Mueve la pantalla a la izquierda

;=============================================
;     CONSTANTES DE RETARDO E INHERENCIAS
;=============================================
LCD_RS_HIGH     equ 1T              ; Comando de escritura
LCD_SPACE_CHAR  equ $20             ; Caracter de espacio
LCD_CLR_DLY     equ 5T              ; Retardo de estabilidad de comando
CLR

LCD_ROW1        equ 1T              ; Comprobante de fila 1
LCD_ROW2        equ 2T              ; Comprobante de fila 2
LCD_ROW0_OFFSET equ $80             ; Desfase hacia fila 0 ($80 + $00)
LCD_ROW1_OFFSET equ $C0             ; Desfase hacia fila 1 ($80 + $40)
LCD_ROW2_OFFSET equ $90             ; Desfase hacia fila 2 ($80 + $10)
LCD_ROW3_OFFSET equ $D0             ; Desfase hacia fila 3 ($80 + $50)
```

```
;=========================================
; LCDINIT       :  Inicializa el módulo LCD
; OBJETIVO      :  Configura el LCD en modo de
;                  8bits, 2 líneas, matriz de
;                  5 X 11 puntos si existe en el
;                  LCD, Incremento
; ENTRADA       :  Ninguna
; SALIDA        :  Ninguna
; REGISTROS
; AFECTADOS     :  PORTX, DDRX, A
;=========================================
LCDInit
        clrh                                    ; Borra H
        ldx #40T                                ; A retardar 40 ms
        jsr Delay                               ; Retarda
        clr LCD_DATA_PORT                       ; DATA     = LOW
        mov #LCD_DATA_OUTPUT,LCD_DATA_DDR
                                                ; DDR_DATA  = OUTPUT
        bset LCD_E_PIN,LCD_E_DDR                ; DDR_E    = OUTPUT
        bset LCD_RS_PIN,LCD_RS_DDR              ; DDR_RS   = OUTPUT
        bclr LCD_RS_PIN,LCD_RS_PORT             ; RS       = LOW
        bclr LCD_E_PIN,LCD_E_PORT               ; E        = LOW
        clra                                    ; A        = 0
        jsr LCDSel                              ; RS       = LOW
$IF LCD_8BIT_MODE_EN
        lda #{$20|LCD_8BIT|LCD_2LINE|LCD_5X11}
                                                ; LCD 8bits, 2 líneas y 5 X 11
$ELSEIF
        lda #LCD_HOME                           ; Regresa a Casa
        jsr LCDWr                               ; Ejecuta comando
        lda #{$20|LCD_2LINE|LCD_5X11}           ; Dos líneas, 5X11, si posee y 4 bits
$ENDIF
        jsr LCDWr                               ; Ejecuta comando
        lda #{$08|LCD_ON|LCD_CSR|LCD_BLINK}
                                                ; Enciende LCD, Cursor, Parpadeo.
        jsr LCDWr                               ; Ejecuta comando
        jsr LCDClr                              ; Borra pantalla
        lda #{$04|LCD_INC}                      ; Modo incremental
        jsr LCDWr                               ; Ejecuta comando
        rts                                     ; Retorna
```

```
;===========================================
; LCDCLR     :  Borra la pantalla
; OBJETIVO   :  Borra completamente el LCD
;               y ubica cursor en la parte
;               superior izquierda
; ENTRADA    :  Ninguna
; SALIDA     :  Ninguna
; REGISTROS
; AFECTADOS  :  A, H:X
;===========================================
LCDClr
        clra                                ; RS = LOW
        jsr LCDSel                          ; Cambia estado de RS a bajo
        lda #LCD_CLR                        ; Comando borrar
        jsr LCDWr                           ; Ejecuta comando
        ldhx #LCD_CLR_DLY                   ; A retardar 5 ms
        jsr Delay                           ; Ejecuta retardo
        rts                                 ; Retorna

;===========================================
; LCDWRMSG  :  Escribe una cadena de carac-
;              teres
; OBJETIVO  :  Escribe una cadena dada por
;              la dirección de una tabla
; ENTRADA   :  H:X   contiene la dirección
;                    de la cadena a enviar
; SALIDA    :  Ninguna
; REGISTROS
; AFECTADOS :  CCR, A, H:X
;===========================================
LCDWrMsg
        sei                                 ; Inhabilita interrupciones
        lda #LCD_RS_HIGH                    ; Comando RS = 1
        jsr LCDSel                          ; RS = 1
LCDWRITE1008.I
        lda ,x                              ; Carga caracter
        beq LCDOUT1008.J                    ; ¿Cero?, salir
        pshx                                ; Empuja a pila
        jsr LCDWr                           ; Envía al LCD
        pulx                                ; Recupera de pila
        aix #1                              ; Incrementa puntero
        bra LCDWRITE1008.I                  ; Siguiente caracter
LCDOUT1008.J
        cli                                 ; Habilita interrupciones
        rts                                 ; Retorna
```

```
;=========================================
; LCDWRMSGXY
;                : Escribe un mensaje en la po-
;                  sición X-Y
; OBJETIVO       : Escribe un conjunto de carac-
;                  teres en la posición X-Y
; ENTRADA        : A      es la fila del mensaje
;                  X (PUSH)
;                         es la columna del men-
;                         saje
;                  H:X    es la dirección del
;                         mensaje a escribir
; SALIDA         : Ninguna
; REGISTROS
; AFECTADOS      : X, SP, CCR
;=========================================
LCDWrMsgXY
        sei                                     ; Inhabilita Interrupciones
        pshx                                    ; Empuja parte baja de puntero a pila
        ldx 4,SP                                ; Carga Columna
        jsr LCDSetXY                            ; Posiciona en X-Y
        pulx                                    ; Recupera parte baja del puntero
        jsr LCDWrMsg                            ; Escribe cadena y habilita
                                                ; interrupciones
        rts                                     ; Retorna

;=========================================
; LCDSEL         : Selecciona modo de instruc-
;                  ciones o datos
; OBJETIVO       : RS = HIGH ó RS = LOW
; ENTRADA        : A      es el estado booleano a
;                         imponer para control
;                         de RS
;                  A = 1  RS = HIGH
;                  A = 0  RS = LOW
; SALIDA         : Ninguna
; REGISTROS
; AFECTADOS      : DDRX, PORTX
;=========================================
LCDSel
        beq LCDOUT1008.A                        ; A = 0, Saltar a imponer RS bajo
        bset LCD_RS_PIN,LCD_RS_PORT             ; A = 1, Impone RS alto
        bra LCDOUT1008.B                        ; Salir
LCDOUT1008.A
        bclr LCD_RS_PIN,LCD_RS_PORT             ; Impone RS bajo
LCDOUT1008.B
        rts                                     ; Retorna
```

```
;=========================================
; LCDWR       : Escribe un caracter
; OBJETIVO    : Muestra un caracter en la
;               pantalla
; ENTRADA     : A      contiene el caracter a
;                      desplegar
; SALIDA      : Ninguna
; REGISTROS
; AFECTADOS   : DDRDX, PORTX, A, X
;=========================================
LCDWr
$IF LCD_8BIT_MODE_EN
        sta LCD_DATA_PORT               ; Almacena dato en puerto de LCD
$ELSEIF
        psha                            ; Empuja el caracter a pila
        and #LCD_DATA_OUTPUT            ; Preserva byte alto LCD
        psha                            ; Empuja Dato LCD a pila
        lda LCD_DATA_PORT               ; Carga Puerto LCD
        and #$0F                        ; Preserva byte bajo
        add 1,SP                        ; Añade el byte preservado
        sta LCD_DATA_PORT               ; Almacena en puerto
        pula                            ; Reposiciona pila
$ENDIF
        bset LCD_E_PIN,LCD_E_PORT       ; Levanta pin Enable
        ldx #5T                         ; XTAL
        lda #10T                        ; A retardar +1 us
        jsr DelayNus                    ; Retardar +1
        bclr LCD_E_PIN,LCD_E_PORT       ; Baja línea enable
        ldx #5T                         ; XTAL
        lda #60T                        ; A retardar +40 us
        jsr DelayNus                    ; Retarda +40 us y más
$IF LCD_8BIT_MODE_EN

$ELSEIF
        pula                            ; Recupera byte caracter de pila
        and #$0F                        ; Preserva byte bajo LCD
        nsa                             ; Posiciona el byte alto
        psha                            ; Empuja Dato LCD a pila
        lda LCD_DATA_PORT               ; Carga Puerto LCD
        and #$0F                        ; Preserva byte bajo
        add 1,SP                        ; Añade el byte preservado
        sta LCD_DATA_PORT               ; Almacena en puerto
        pula                            ; Reposiciona pila
        bset LCD_E_PIN,LCD_E_PORT       ; Levanta pin Enable
        ldx #5T                         ; XTAL
        lda #10T                        ; A retardar +1 us
        jsr DelayNus                    ; Retardar +1
        bclr LCD_E_PIN,LCD_E_PORT       ; Baja línea enable
        ldx #5T                         ; XTAL
        lda #60T                        ; A retardar +40 us
        jsr DelayNus                    ; Retarda +40 us y más
$ENDIF
        rts                             ; Retorna
```

```
;=========================================
; LCDSETXY    : Cursor de LCD ubicado en
;               Posición de pantalla
; OBJETIVO    : Ubica cursor en posición X-Y
; ENTRADA     : A     es la fila en número de-
;                     cimal
;               X     es la columna en número
;                     decimal
; SALIDA      : Ninguna
; REGISTROS
; AFECTADOS   : CCR
;=========================================
LCDSetXY
        cmp #LCD_MAX_ROW            ; Compara A con máximo de filas
        bge LCDOUT1008.G            ; Mayor o igual?, salir
        cpx #LCD_MAX_COL            ; Compara X con máximo de
                                    ; columnas
        bge LCDOUT1008.G            ; Mayor o igual?, salir
        psha                        ; Empuja fila a la pila
        clra                        ; A = 0
        jsr LCDSel                  ; RS = LOW
        pula                        ; Recupera A
        pshx                        ; Empuja columna a pila
        tsta                        ; Comprueba si es fila 0
        beq LCDOUT1008.C            ; Salir si es 0
        cmp #LCD_ROW1               ; Comprueba si es fila 1
        beq LCDOUT1008.D            ; Ir a offset de fila 1
        cmp #LCD_ROW2               ; Comprueba si es fila 2
        beq LCDOUT1008.E            ; Ir a offset de fila 2
        ldx #LCD_ROW3_OFFSET        ; Carga offset de fila 3
        bra LCDOUT1008.F            ; Salta a posicionar en fila
LCDOUT1008.C
        ldx #LCD_ROW0_OFFSET        ; Carga offset de fila 0
        bra LCDOUT1008.F            ; Salta a posicionar en fila
LCDOUT1008.D
        ldx #LCD_ROW1_OFFSET        ; Carga offset de fila 1
        bra LCDOUT1008.F            ; Salta a posicionar en fila
LCDOUT1008.E
        ldx #LCD_ROW2_OFFSET        ; Carga offset de fila 2
LCDOUT1008.F
        txa                         ; Carga columna
        add 1,sp                    ; Suma fila + columna
        jsr LCDWr                   ; Posiciona en Fila, repite X veces
        pulx                        ; Posiciona stack
LCDOUT1008.G
        rts                         ; Retorna
```

Listado 75. NT1010 – PSX – LCD.inc. Archivo de funciones de manejo de una LCD. Para esta nota, se hace énfasis en las funciones utilizadas y la configuración en modo de 4 bits.

```
;=============================================
; DELAY       : Genera un retardo de tiempo
; OBJETIVO    : Retardo de tiempo, base 1ms
; ENTRADA     : H:X = Retardo en ms
; SALIDA      : H:X = 0
; REGISTROS
; AFECTADOS   : H:X
; USO         :
;               MIN = H:X = 1T
;               MÁX = H:X = 65535T
;               ldhx #500T
;               jsr Delay ; retarda 0.5 seg
;=============================================
Delay
        cphx #0                         ; [3] Compara con 0
        beq DELAY0009.A                 ; [3] Salir si es 0
        pshx                            ; [2] Salva X en la pila
        pshh                            ; [2] Salva H en la pila
        ldhx #DELAY49152                ; [3] Carga constante de bucle fino
Delay0009.B
        aix #-1                         ; [2] Decrementa H:X en 1
        cphx #0                         ; [3] LLegó a cero (0)
        bne Delay0009.B                 ; [3] Si no es igual, salta a Delay0
        pulh                            ; [2] Si es igual, recupera H de la pila
        pulx                            ; [2] Recupera X de la pila
        aix #-1                         ; [2] Decrementa H:X en 1
        cphx #0                         ; [3] LLegó a cero (0)
        bne Delay                       ; [3] Si no es igual, salta a Delay
DELAY0009.A
        rts                             ; [4] retorna
```

```
;=========================================
; DELAYNUS    :  Genera un retardo de tiempo
; OBJETIVO    :  Retardo de tiempo, base 1us
; ENTRADA     :  X  =  Valor entero del cris-
;                      tal externo
;                A  =  Tiempo a retardar en us
; SALIDA      :  Ninguna
; REGISTROS
; AFECTADOS   :  A, X
;=========================================
DelayNus
        tsta                            ; [1] Verifica estado de A
        beq DELAY0009.C                 ; [3] Salir si es cero
        deca                            ; [1] Decrementa A
        psha                            ; [2] Empuja A a pila
        pshx                            ; [2] Empuja variable X
        dbnzx *                         ; [3] Decrementa hasta 0
        pulx                            ; [2] Recupera variable X
        pula                            ; [2] Recupera A
        bra DelayNus                    ; [3] Salta a comprobar estado de A
DELAY0009.C   rts                       ; [4] Retorna
```

Listado 76. NT1010 – PSX – DELAY.inc. Archivo que contiene las funciones de retardo programables.

```
;=====================================================================
; ARCHIVO      : KBI.inc
; PROPÓSITO    : Inclusión de la función de inicialización del módulo de
;                teclado
; LENGUAJE     : IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   10  09  04   Creado.
;   10  09  04   Modificado.
;=====================================================================

;=============================================
; KBIINIT      : Inicializa registro de inte-
;                rrupción de teclado.
; OBJETIVO     : Teclas PTA como interrup-
;                tores de teclado.
; ENTRADA      : A     teclas a inicializar
;                      como teclado
;                X     valor booleano del
;                      bit MODEK
; SALIDA       : Ninguna
; REGISTROS
; AFECTADOS    : KBSCR, KBIER
;=============================================
KBIInit
        tsta                    ; prueba ACCA
        beq KBIOUT0108.A        ; Si es cero, salir
        bset BIT1,KBSCR         ; IMASK = 1, Enmascaro interrupción de
                                ; teclado
        sta KBIER               ; Escribo en KBIER y habilito las teclas
                                ; Se generó una falsa interrupción.
        txa                     ; Transfiere MODEK a ACCA
        ora #ACKK               ; Reconoce la interrupción
        sta KBSCR               ; Almacena en el registro
KBIOUT0108.A
        rts                     ; Retorno de la subrutina
```

```
;===========================================
; KBIACK        :  Borra Interrupción de Tecla
; OBJETIVO      :  Reconoce IRQF del KBSCR
; ENTRADA       :  Ninguna
; SALIDA        :  Ninguna
; REGISTROS
; AFECTADOS     :  KBSCR, A
;===========================================
KBIAck
        lda #MODEK              ; A = MODEK
        and KBSCR               ; Preservo bit MODEK
        ora #ACKK               ; Cargo ACCK + A
        sta KBSCR               ; Borro falsa interrupción
        rts                     ; Retorna
```

Listado 77. NT1010 – PSX – KBI.inc. Archivo de funciones del módulo de teclado del microcontrolador.

```
;=====================================================================
; ARCHIVO      :  USER.inc
; PROPÓSITO    :  Funciones de usuario
; LENGUAJE     :  IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   06   09   04   Creado.
;   21   12   04   Modificado.
;=====================================================================

;=============================================
;                    Macros & Constantes
;=============================================
PAD_SELECT          equ %11111110          ; Botón Select presionado
PAD_START           equ %11110111          ; Botón Start  presionado
PAD_UP              equ %11101111          ; Botón Up     presionado
PAD_RT              equ %11011111          ; Botón Right  presionado
PAD_DN              equ %10111111          ; Botón Down   presionado
PAD_LT              equ %01111111          ; Botón Left   presionado

;=============================================
;                    Tablas del Usuario
;=============================================
$include '\USER\TABLES.inc'                ; Tablas del Usuario
```

Listado 78. NT1010 – PSX – USER.inc. Incluye las funciones de usuario y macros definidos por el usuario.

```
;=================================================================
; ARCHIVO      :  TABLES.inc
; PROPÓSITO    :  Tablas de búsqueda predefinidas por el usuario
; LENGUAJE     :  IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   06   09   04   Creado.
;   11   11   04   Modificado.
;=================================================================

;=============================================
;           Tablas de Búsquedas
;=============================================
;=============================================
;            TABLA DE MENSAJES
;=============================================
LCDMESSAGE00   db 'Presionado:',0
LCDMESSAGE01   db 'Select',0
LCDMESSAGE02   db 'Start ',0
LCDMESSAGE03   db 'Up    ',0
LCDMESSAGE04   db 'Right ',0
LCDMESSAGE05   db 'Down  ',0
LCDMESSAGE06   db 'Left  ',0
```

Listado 79. NT1010 – PSX – TABLES.inc. Muestra la tabla de mensajes utilizada.

```
;=====================================================================
; ARCHIVO      :  INTERRUPTSJL3.inc
; PROPÓSITO    :  Plantilla de Funciones de Interrupciones para
;                 JL3/JK1/JK3/Serie QT/QY
; LENGUAJE     :  IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   22   08   04 Creado.
;   06   12   04 Modificado.
;=====================================================================

;=============================================
;               Interrupción del Módulo de Teclado
;=============================================
KBINTL                                  ; Teclado (Bajo)
      jsr PSXPadAck                     ; Decodifica teclado, almacena
                                        ; en buffer si hay espacio,
                                        ; almacena la última tecla
                                        ; presionada y reconoce la
                                        ; Interrupción
      rti                               ; Retorno de la interrupción.
```

Listado 80. NT1010 – PSX – INTERRUPTS.inc. Archivo que contiene las definiciones de funciones utilizadas en la nota.

```
;=====================================================================
; ARCHIVO      :  VECTORSJL3.inc
; PROPÓSITO  :  Definir el vector de búsqueda de cada interrupción
; LENGUAJE    :  IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   22   08   04   Creado.
;   06   12   04   Modificado.
;=====================================================================

;=============================================
;                  Vector del Módulo de Teclado
;=============================================
        org KBINTH                  ; Teclado (Alto)
        dw KBINTL                   ; Teclado (Bajo)

;=============================================
;                  Vector de Reinicio del Sistema
;=============================================
        org RESET_VEC               ; Puntero Vec - RESET
        dw START                    ; al darse reset salta a Stara
```

Listado 81. NT1010 – PSX – VECTORSJL3.inc. Activa los vectores de interrupción a utilizar.

3.10.8 Conclusión

Se puede diversificar y conseguir variantes de teclado utilizando un “Pad” de “Playstation”, ya sea para aplicaciones de robótica B.E.A.M. o simple deseos de realizar videojuegos utilizando un microcontrolador como esclavo de la PC. Como los microcontroladores como el JK3 no se les puede conectar un puerto de teclado directamente (sin “hardware” KBI completo), esta puede ser una alternativa auxiliar, bastante vistosa para aplicaciones de aficionados.

La nota cubrió el uso del para adquisición de señales de un solo “pad”, demostrando que se puede tener un firme y confiable control para mando sin necesidad de realizar un teclado. Inclusive se puede conectar al mismo bus otro “pad”, solo configurando el pin SEL dependiendo de que “pad“ se requiera atender.

3.10.9 Referencias

3.10.9.1 “Embedded Systems Building Blocks, Second Edition” - ”Complete and Ready-to-Use Modules in C”
Autor: Jean J. Labrosse
Recurso: Capítulo 05 – Character LCD Modules

3.10.9.2 Información Avanzada sobre el Microcontrolador
(a) http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC68HC08JL3.pdf

3.10.9.3 Manual de Referencia del CPU
(a) http://www.freescale.com/files/microcontrollers/doc/ref_manual/CPU08RM.pdf

3.10.9.4 Página “web” sobre esta Nota Técnica
(a) http://www.geocities.com/issaiass/

NT0026 – Módulos – Rutinas reusables para programación. Apéndices de Tesis

3.10.9.5 Páginas sobre la Consola de “Playstation”
(a) http://www.iespana.es/wk3/psxtopc.htm
(b) http://es.geocities.com/gamemasterquilpue/MemoryControlProject.htm
(c) http://www.gamesx.com/controldata/psxcont/psxcont.htm#SIGNAL
(d) http://www.raphnet.net/electronique/psx_cardmgr/Playstation.txt
(e) http://es.tldp.org/NuLies/web/2.2/Documentation/joystick-parport.txt
(f) http://www.emulatronia.com/secciones/frames/frames-doctec.htm

3.10.10 Problemas Propuestos

3.10.10.1 Escriba un código reusable que permita hablarle a dos “pads” utilizando el mismo bus. Nota: Puede conectar la línea SEL de cada “pad” a dos pines diferentes.

3.10.10.2 Cree un robot móvil sencillo que realice movimientos básicos, ya sea por medio de módulos inalámbricos o con cables hacia el mismo.