
M.H.M
ezine
3

DISCLAIMER:

Esta revista electrónica fue creada con el único propósito de educar y entretener a la gente. Nosotros en la MHM no nos hacemos responsables del uso que se le dé a la información contenida en ella.

Los textos, gráficas y diagramas publicados aquí, se exponen con el fin de proporcionar datos y material técnico y de investigación mismos que deberán ser empleados siempre con fines educativos.

Bienvenidos una vez más a su eZine preferida. Esta edición es importante para nosotros porque la empezamos a desarrollar cerca de la fecha de nuestro primer aniversario como grupo de phreaking y además está repleta de información altamente valiosa.

Como habrán notado, se sigue con los temas que ya se habían iniciado, y se sigue incluyendo temas que ayudan a los principiantes a iniciarse en este magnifico mundillo phreak.

Una gran aportación a esta edición es el código fuente del famoso emulador de Cartman para tarjetas de primera generación de Telmex, comentado y analizado de manera increíble por Illan Ivanovich. Espero que este texto ayude a todos los electrónicos neófitos a realmente comprender el funcionamiento de las Telecards y el lenguaje ensamblador para PICs.

Bueno, no les quito más el tiempo. ¡¡¡Animo, y a seguir aprendiendo!!!

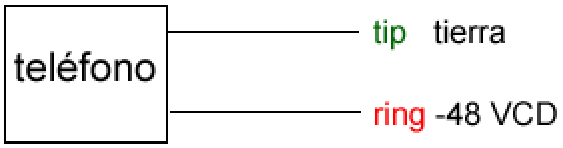
-=oSUKARu=-

EL TELÉFONO

por -=oSUKARu=-

Saludos amigos phreaks de todo el mundo. Hoy he decidido hablarles de algo realmente emocionante, intrigante y sumamente útil y divertido, así es, hoy toca escribir un texto sobre el Teléfono.

Una línea telefónica normal consiste, generalmente, de un par de cables en los que se transmite audio en forma *full duplex.* Estas líneas están alimentadas con una tensión de 48 Volts de corriente directa:



Como se puede ver en este dibujo, la tierra en realidad es el lado positivo, del otro extremo tenemos una tensión de –48 Volts, esta forma de alimentar las líneas telefónicas es muy común.

Esta forma de alimentar las líneas fue inventada en 1940 y aun un sin fin de compañías telefónicas alrededor del mundo lo emplean.

Cuando uno descuelga, el voltaje baja hasta un voltaje entre 3 y 9 VDC, y a través de él fluye una corriente del rango de 20 a 70 mA.

El ancho de banda de un teléfono es de 200 – 3200 Hz, que para cuando fue estandarizado por las compañías telefónicas era más que suficiente para el intercambio de conversaciones, e incluso era suficiente para multiplexar varias llamas a través de una misma línea (como sucede al utilizar calling cards, por ejemplo) sin perdida notable de calidad.

Examinando el por que se eligió este rango para las frecuencias manejadas por un teléfono podemos ver que el límite inferior es lo suficientemente grande como para alejarse de la frecuencia de la Corriente Alterna (50 – 60 Hz), de no haber sido así experimentaríamos una terrible interferencia al realizar llamadas.

Como mencione anteriormente, la voz transmitida por una línea telefónica está balanceada a *full duplex* lo que significa que se puede hablar y escuchar a la vez (imagina el *half duplex* como una conversación por walkie-talkies o por radio civil, en el que se necesita esperar después de hablar para poder escuchar lo que la otra persona tiene que decir).

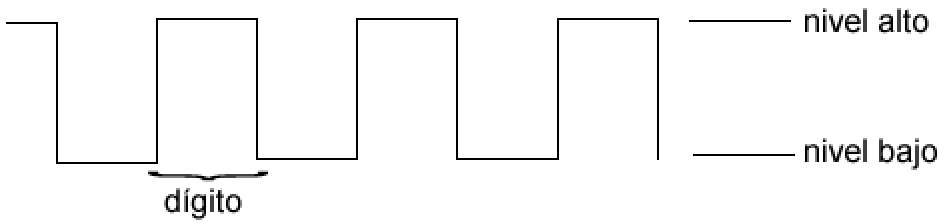
Tipos de Marcado:

Básicamente existen dos tipos de marcado:

- Por TONOS
- Por PULSOS

Los teléfonos más nuevos pueden marcar de ambas formas, pero en la antigüedad, con los teléfonos *rotatorios* o *de Dial*, la forma de marcar era por pulsos.

La forma en que se marcaba era cerrando y abriendo interruptores dentro del teléfono en tiempos predeterminados, dejando un espacio en nivel alto entre digito y digito.



Ahora como este tipo de marcación no es más que una sucesión de estar colgando y descolgando la línea, también puede ser hecho por medio del botón que tienen los teléfonos para colgar.

// Experimento casero.
// begin
Ve a donde tengas tu teléfono y descuélgalo... -espera, termina de leer esto primero ^_^ - Lo que tienes que hacer es lo siguiente, con tus dedos cuelga y descuelga de manera constante y rápida. Lo que estas haciendo es marcando por pulsos, por lo general marcaras un numero inexistente, pero si logras controlar la forma en la que mandas los pulsos al colgar y descolgar podrás realizar una llamada marcando de esta forma.
// end

La marcación por pulsos es más nueva, y es la que comúnmente utilizamos en todos los teléfonos.

La forma de marcar es por Multi Frecuencia (DTMF)

	Col 1: 1209 Hz	Col 2: 1336 Hz	Col 3: 1477 Hz	Col 4: 1633 Hz
1: 697 Hz	1	ABC 2	DEF 3	(FO) [A]
2: 770 Hz	GHI 4	JKL 5	MNO 6	(F) [B]
3: 852 Hz	PRS 7	TUV 8	WXY 9	(I) [C]
4: 941 Hz	*	OPER 0	#	(P) [D]

Como se puede ver en la tabla, cada dígito es generado por la combinación de dos frecuencias, de ahí su nombre.

Esta técnica fue inventada para evitar los problemas que suceden en la marcación por pulsos, como el que un dígito fuera mal recibido y en lugar de marcar a un numero se marcaría a otro.

Bien, creo que aquí daré por terminado este pequeño curso, ojalá les haya gustado.

¡Saludos!

-=oSUKARu=-
http://mhmphreak.da.ru

Análisis del Emulador de Cartman
Por Illan Ivanovich

INTRODUCCIÓN.

Lo siguiente es una explicación a grandes rasgos de un emu realizado por CARTMAN. El emu y los diagramas los pueden bajar de su página, así como un diagrama de flujo del mismo. Esto lo hago con el fin de atraer la atención de los que apenas comienzan con el estudio de programación de PIC. En realidad espero aprender tanto como Ustedes, les aclaro que yo apenas comienzo con esto y que si encuentran algunos errores háganmelo saber. Además de todo esto me pongo a sus órdenes para aclarar cualquier duda que esté a mi alcance. Esto se comenzó a publicar en el foro de oSUKARu pero lo concluimos aquí.

Espero les sirva de algo.
Agradecimientos a CARTMAN por las dudas que me aclaró, a oSUKARu, y o0ShellGhost0o por permitirme participar con ellos.
Estamos en lo mismo...
BK

Cartman espero puedas ayudarnos a todos.
Hace poco estuve checando un estudio hecho por el Narco en el cual me inspire para hacer lo siguiente.
voy a tratar de exponer lo que entiendo del asm de cartman y espero tener opiniones acerca del mismo.
Para evitar mezclar tanta info lo voy a hacer poco a poco.
Cabe aclarar que estoy iniciandome en esto de ensamblador por lo que es importante para todos los que empiezan.

Si de alguna manera esto puede ocacionar problemas con el foro, ruego me lo hagan saber.
Gracias.

```
        TITLE   "Emulador de tarjetas telefonicas de 256 bits"
        LIST P=PIC16F84, R=HEX
        INCLUDE "P16F84.INC"

MAXCARDS    equ     .21     ; Change it to any desired value (<22)

; PIC16F84 I/O Pin Assignment List

CRD_CLK     equ   0     ; RB0 + RA4 = Card Clock
CRD_DTA     equ   0     ; RA0 = Card Data Output
CRD_RST     equ   1     ; RB1 = Card Reset, Low-Active
CRD_WE      equ   7     ; RB7 = Card Write-Enable, Hi-Active
RESTOREFUSES        equ     3       ; RB3 = Restore Fuse counter
```

Bueno. Los tres primeros renglones nos indican
1. titulo del programa. El cual debe ir siempre entre comillas
2.list p=pic16f84, r=hex este nos indica el tipo de pic que se va a usar y que dara como resultado alser compilado un archivo hexadecimal.
3.Include "p16f84.inc" es un archivo adjunto donde vienen configuradas los registros internos del pic (Por ejemplo los puertos, el registro w, etc.)

MAXCARDS EQU .21 . Esta aclaración ya me la habia hecho cartman y se lo agradezco. Esta constante nos indica el número de tarjetas que podran ser emuladas con este programa. Se pueden emular hasta 21 tarjetas.

Pero se podria por ejemplo emular solo dos tarjetas y se tendria que cambiar el .21 por 2 y eliminar los 19 headers restantes. Para los que no saben que son los headers Cartman posteo información basica en este mismo foro.

Terminando esto nos pasamos a la asignación de los pines en el pic. Las etiquetas que estan a la izquierda (CRD_CLK, CRD_DTA, CRD_RST, CRD_WE, RESTOREFUSES) Son para indicar simplemente el número de bit que va a ser usado ya sea como salida o entrada en los puertos del pic, en este caso PORTA y PORTB.
Por ejemplo
RESTOREFUSES EQU 3 Nos indica que la constante llamada RESTOREFUSES es igual a 3 por lo tanto en cuanto se ejecuta la instruccion
btfss PORTB, RESTOREFUSES Significa que el pic va achecar en el puerto B el bit No. 3 (RB3) y saltarse una instruccion en caso de que sea 1.

Por el momento es todo, espero sus comentarios.}
Insisto: Si esto puede causar algun problema haganlo saber.
bk.

Esto es la segunda parte de estudio del emu de cartman.
Saludos Cartman

Estuve estudiando acerca de como configurar el temporizador del píc. Esto se puede hacer de dos maneras. Una de ellas es utilizando el pin RA4 que es común a la entrada de reloj externo. Esto debe configurarse atraves del registro OPTION, el cual consta de ocho bits que deben ponerse ya sea en cero o en uno según sea la conveniencia, por ejemplo tenemos el bit RTE (Bit 4 de OPTION) el cual cuando es puesto a 0 el registro TMR0 (Se explicara después) se incrementa en un flanco de subida y viceversa. Así como cuando ponemos RTS (Bit 5 de OPTION) en cero, tenemos que se usa el reloj interno y se se pone a 1 podemos usar un reloj externo que esta entrando por RA4.
Esto lo veremos mas claramente cuando lleguemos al punto donde configuraremos el registro OPTION.

Pasamos a la segunda parte del Emu de Cartman

```
; Constants

EE_FLAG             equ     0
DEFAULTFUSCNT       equ     .5

; PIC16F84 RAM Register Assignments

                    org     0xc

CRD_ID          res         12
FUSCNT          res         1
CRDCNT                      res     1
BITCNT          res         1
POSCNT                      res     1
MSKAUX                      res     1
LOOPCNT         res         1
FLAGS       res 1
TEMP_W          res         1
TEMP_S          res         1
TMP1            res         1
```

__CONFIG _CP_OFF & _WDT_OFF & _PWRTE_OFF & _XT_OSC

En las primeras lineas se estan definiendo dos constantes que van a ser utilizadas en el programa, y a las cuales se les esta asignando los valores de 0 y 5 (El punto a la izquierda del 5 significa que esta en forma decimal y no en hexadecimal)???? Estoy Bien?

La linea org 0xc nos indica en que punto de la RAM se iniciará a crear los registros que nos van a ser útiles.?????
Porque usamos 0xc (12 decimal)? Por que los primeros 12 son fijos y cumplen un proposito determinado, en tanto que del 13 al 48 son registros de propósito general.

En los siguientes renglones se esta definiendo tambien valores constantes pero que seran usados dentro de la memoria ram como registros de propósito general en los cuales se almacenará algun valor. Por ejemplo en la linea:

CRD_ID RES 12 Tenemos que en este registro será almacenado el CRD_ID o sea los primeros 12 bytes de una tarjeta telefonica (Header).
En realidad quisiera saber por que se esta usando res y no equ. Quiero creer que si se usa la opción equ tendriamos que poner las direcciones en hexadecimal.??????? O es porque significa que se van a usar por ejemplo CRD_ID usará doce direcciones en ram y segido de estas doce direcciones el registro FUSCNT usará la siguiente..... y así?????

Agradezco su atención y empeño por todo esto. De verdad es bueno e importante conocer todo esto, y no me refiero principalmente a la emulación de una tarjeta, me refiero a que si aprendemos mas acerca de programación de microcontroladores, nada nos detendrá para realizar proyectos mejores y mas complicados. Espero esto sea la base para el conocimiento y no para fines malévolos..

Gracias a todos.

bk

Esta es la tercera parte del estudio del asm de Cartman. A pesar de que nos cerraron la página de MHM, no lo dejaremos para después. Espero que cuando ya esté abierta la página nos den chance de subir el Doc.

Ahora veremos la rutina de Interrupción. Con la cual vamos a hacer todas las interrupciones el el micro dependiendo de las señales de entrada: CRD_RST, CRD_WE, CRD_CLK y RB3, aunque en realidad la opción de recarga de tarjeta se hace con RB3 y no está incluida en la rutina de RST sino hasta el programa principal. Vamos por el principio.

org 0 Nos indica el inicio del programa principal en la dirección 0
goto INIT Nos vamos directamente al programa principal. En realidad el programa inicia en la etiqueta INIT

org 4 Este es el vector de interrupciones. En la direccion 4 tenemos nuestro inicio de la rutina de interrupciones que es la siguiente.

movwf TEMP_W Mueve lo que tenemos en el registro W hacia nuestro registro TEMP_W que ya fué definido. Esto es con el fin de que cuando regresemos de la interrupción y continuemos con el programa principal podamos recuperar el valor del registro W.TEMP_W puede significar registro w temporal.

swapf STATUS,W Con esta instrucción estamos intercambiando los nibbles. (Los 4 bits mas significativos son un nibble y los menos significativos son otro nibble, por lo que en cada byte tenemos dos nibbles.)Se intercambian nibbles del registro STATUS y se guarda en el registro w. Esto también con el fin de tener el valor actual del registro STATUS para cuando termine la interrupción. Por que se guarda con la instrucción SWAPF y no simplemente con movwf???
Eso lo veremos mas adelante...?

movwf TEMP_S Guardamos el registro STATUS con los nibbles intercambiados y lo guardamos en el registro especial TEMP_S (Registro temporal para STATUS) . Esto tambien con el fin de tener los valores originales despues de regresar de la interrupción. Cabe aclarar que el registro STATUS contiene información de importancia para el micro tal como c, dc y z. No estaría por demás estudiar un poco sobre este registro.

bcf STATUS,RP0 ; Access register bank 0. Como lo dice: Accesar al banco de registros 0 en el cual se encuentran nuestro registro que se va a manipular. Este tipo de instrucción se verá constantemente a lo largo del programa ya que es necesario el cambio de bancos de registros o páginas de registros, seleccionando con RP0 o RP1. La instruccion indica que se ponga a cero el
bit RP0 de la palabra de estado STATUS, y con esto entramos al banco 0.

bsf PORTA,CRD_DTA ; Set Data Output High Simple: Ponemos a la salida o nuestra linea OUT (De donde saldran todos los bits simulando una tarjeta original) en alto o en uno. Por que? Porque si checamos los diagramas de tiempos de nuestra tarjeta a emular, encontramos que durante un reset o cuando se inserta la tarjeta a la cabina esta linea esta alta. De esto se han valido las empresas que diseñan la forma de lectura y validación de las tarjetas para poner trampas o parches en sus cabinas, y es por eso que en ocaciones no funcionan las tarjetas. Es recomendable conseguirse un logger para tener los diagramas de tiempos exactos. Pero de cualquier forma el emu sigue funcionando.

btfsc PORTB,CRD_RST ; Check if reset is low Aquí es donde se ve claramente la naturaleza de la interrupción. Esta instrucción está checando el bit CRD_RST que ya fué definido (Bit 1) en el puerto B y salta si es cero, es decir: salta una posición. si se esta aplicando un RESET a la tarjeta en caso de que sea uno significa que el origen de la interrupción no fué un RESET y pasa a la siguiente instrucción.
goto NO_RST ; If not, skip reset sequence Se va directamente a la rutina de NO_RST.

clrf BITCNT ; Clear BITCNT Aquí inicia la rutina de RST (Cuando se aplica un cero a RB1) Borra el registro de contador de bits que también ya fué definido.
clrf POSCNT Borra el registro POSTCNT Nota: Este registro sirve para contar la posición de los bits, es decir, este registro solo cuenta del bit cero al siete (Ocho bits)

movlw CRD_ID Mueve hacia el registro W lo que está almacenado en CRD_ID
movwf FSR Y lo almacena en FSR (Lo cual significa que en este momento el registro FSR esta apuntando a la localidad de la RAM especificada por el registro CRD_ID)
movfw INDF En realidad este registro sirve únicamente para especificar la utilización del direccionamiento indirecto.(FSR)

movwf MSKAUX Se almacena el contenido de w en el registro MSKAUX que ya fue definido con anterioridad. En este momento el contenido del registro w sigue siendo CRD_ID. MSKAUX es un registro especial que nos sirve para almacenar un BYTE del HEADER.
goto READBIT Saltamos a la rutina READBIT (Lectura de bit).

Seguimos con nuestro estudio. Pasaremos ahora a ver lo conserniente a interrupciones y subrutinas de interrupción. Esto complementa la parte anterior donde estudiamos la parte de rutina de RESET.

NO_RST Esta es la subrutina de cuando no se ha realizado un reset en la TARJETA.

movlw 0x60 Almacenamos en el registro w el valor 60 Hexadecimal '01100000' binario o 96 decimal que es el valor de los bits protegidos que se van a leer desde el emu.

subwf BITCNT,0 Se le resta al registro w lo que tiene almacenado al contador de bits BITCNT y se almacena en el registro w.

btfsc STATUS,C Checamos el estado del carry en el registro STATUS. Es decir se checa que el valor del registro BITCNT no sea mayor a 96. En caso de ser mayor al restar BITCNT del valor almacenado en w (96) se activará el bit carry del registro STATUS. En caso de que c=0 la siguiente instrucción será READBIT. En caso de que sea 1 se va directamente a CREDIT. Porque?? Por lo que hemos visto tenemos que contar de 0 a 96 y leer el bit a la salida del emu para que reconozca la cabina todo el header. En el momento de que terminó de contar los 96 bits es impresindible que siga al area de descuento de crédito ya que en este momento ya no nos interesa la lectura del header.

goto CREDIT Se va a la rutina de descuento de crédito.

NOTA: Con esta parte se comienza lo mas complicado del asunto. Es recomendable que antes de iniciar el estudio de lo siguiente tengamos conocimientos previos del asm. De todas formas se explica paso a paso pero al menos para mí es un poco complicado.

READBIT Rutina de Lectura de datos del HEADER.

btfss MSKAUX,7 Aqui vamos a checar el contenido del bit 7 del registro MSKAUX. Recuerde que en este registro se almacenó el valor de CRD_ID (El último bit del primer BYTE del HEADER ******** En caso de ser 1 brinca una instrucción, es decir, el valor a la salida seguirá siendo1. En caso de ser cero sigue con:

bcf PORTA,CRD_DTA ; Clear Data Output En este punto como MSKAUX (Bit 7) Fué 0, entonces iniciamos con la secuencia de salida de bits del port a, que en este caso seria 0.

incf BITCNT,1 Incrementamos el contador de bits BITCNT y lo almacenamos en el mismo.

incf POSCNT,1 Incrementamos el valor de POSCNT y lo almacenamos en el mismo.

btfsc POSCNT,3 Checamos el estado del bit 3 del registro POSCNT y saltamos si es cero. Si analizamos esta instrucción encontramos que se trata de un contador de posicionamiento de bits dentro de un byte. Es decir, esta instrucción nos indica que cuando POSCNT es igual a 8 en binario se va a la rutina INCFSR que es donde se va almacenar un nuevo BYTE del HEADER, esto se explicará mas detalladamente adelante.

goto incfsr Salta a incfsr

rlf MSKAUX,1 Rotamos a la izquierda los bits contenidos en MSKAUX y se almacena en el mismo con el fin de tener el siguiente BIT del BYTE que se está leyendo.

bcf INTCON,INTF ; Clear INT Interupt Flag Limpiamos la bandera de interrupciones y vamos al final de la interrupcion. INTCON es un registro especial de control de Interrupciones el cual contiene bits con diferente significado, entre ellos se encuentra INTF. Cuando este bit es puesto a 1 entonces nos indica que hubo una interrupción en la linea RB0/INT y por lo tanto debemos regresarla a cero nuevamente , cuando se ejecuta una interrupción este bit se pone a 1 automáticamente para evitar que se ejecute otra interrupción y, valga la redundancia, interrumpa la interrupción. Es por esto que cuando se termina la interrupción este bit debe ponerse nuevamente a 0 a travez del programa.

goto ENDINT Ir al final de interrupcion.

incfsr Incrementación de fsr (Subrutina)

clrf POSCNT Limpiamos lo contenido en POSCNT. Qu en este caso debería tener almacenado 00001111.

incf FSR,1 Incrementamos el registro FSR y lo almacenamos en el mismo.

movfw INDF En realidad este registro sirve únicamente para especificar la utilización del direccionamiento indirecto.(FSR)

movwf MSKAUX Se almacena el valor de FSR en MSKAUX. En este momento tendremos el valor del segundo BYTE del HEADER almacenado en MSKAUX.

bcf INTCON,INTF Limpiamos la bandera de interrupciones y vamos al final de la interrupcion.

goto ENDINT Ir al final de interrupción.

CREDIT Rutina de credito

btfss PORTB,CRD_WE Checar el bit CRD_WE del puerto b para saber si se requiere de quitar credito de la tarjeta.

goto NO_WRT En caso de que el bit sea 1 salta. Si no se va a NO_WRT

btfss PORTB,CRD_RST Checa el estado del rst si esta a 0 se va a NO_WRT

goto NO_WRT Se va a NO_WRT

incf FUSCNT,1 Incrementamos FUSCNT y lo almacenamos en el mismo. FUSCNT nos sirve para saber cuantos bits han sido puestos a uno. Este contador comienza afuncionar despues de hacer el conteo de (96 bits)

incf BITCNT,1 Incrementamos el contador de bits

bsf FLAGS,EE_FLAG Habilitamos el flag de escritura de la EEPROM

bcf INTCON,INTF Limpiamos la bandera de interrupciones y vamos al final de la interrupcion.

goto ENDINT

NO_WRT Rutina de NO escritura.

subwf FUSCNT,0 Restamos a w lo que hay en FSCNT y lo almacenamos en w. Recordemos que en w tenemos el valor de bits que hay en el header (96)

btfss STATUS,C Checamos el estado del carry en STATUS. En caso de que el valor de FUSCNT sea mayor de 96 se activara el bit carry del registro STATUS y por lo tanto se saltará una instrucción. En caso de que sea menor de 96 sigue con la siguiente instrucción.

bcf PORTA,CRD_DTA En este momento nos encontramos con que la cabina no quiere quitarnos crédito pero si quiere leer lo que se encuentra despues de los 96 bits que en este caso solo son ceros.

incf BITCNT,1 ncrementamos el contador de bits BITCNT una posición.
Limpiamos la bandera de interrupciones y vamos al final de la interrupcion.

goto ENDINT Fin de la interrupción.

Esta parte es muy importante, y comienza desde las primeras lineas de la rutina de interrupciones. Recuerdan que en un principio se almacenaron los registros w y STATUS en registros temporales?? Pues bueno en este momento es hora de restaurarlos para volver al programa principal con los valores en los que se encontraban.

ENDINT

swapf TEMP_S,W ; Restore registers Rotamos los nibbles que se encuentran en el registro temporal para STATUS y lo almacenamos en w.

movwf STATUS Almacenamos el valor de w en STATUS. Por lo tanto TEMP_S=STATUS

swapf TEMP_W,1 Rotamos los nibbles contenidos en el registro temporal TEMP_W y lo almacenamos en el mismo.

swapf TEMP_W,0 Rotamos nuevamente el registro TEMP_W y lo almacenamos en w.

Por que se usa tanto la instrucción SWAPF y no se cargan directamente los valores???

retfie ; return from interupt & clear flag Regresamos al programa principal. Cuando se realiza una interrupción, el pic pone el bit GIE(Bit 7) del registro INTCON a 0 para evitar que se realice otra interrupción, cuando ponemos retfie, regresamos de la interrupción hacia el programa principal con este bit puesto nuevamente a 1.

Este es el inico del programa principal del emulador. Aclaremos que para entenderlo deberemos haber leido las notas anteriores. No se trata de dar una cátedra del ensamblador ni de electrónica o algo así, pero si tratamos de ampliar más nuestros alcances. Sobre todo para los que apenas comenzamos con esto. En realidad este estudio es dedicado a todos aquellos que como yo comenzamos a entender acerca de programación ya que esto es la base del emu. La cuestión electrónica puede esperar por el momento.

PROGRAMA PRINCIPAL.

INIT bsf STATUS,RP0 Nos vamos al banco de registros 1. Cabe aclarar que se van a manejar dos bancos de registros. En cada banco tenemos diferentes registros de trabajo con los cuales vamos a trabajar, en este caso bsf pondrá en 1 el bit RP0 del registro STATUS con lo cuyal nos vamos al banco 1, en caso de que se pusiera bcf STATUS,RP0 nos vamos al banco 0. Nos vamos al banco 1 porque ahí es donde se encuentra nuestro registro OPTION_REG. Tambien quiero aclara que el registro STATUS se encuentra en los dos bancos y además es importante saber que despues de un reset en el PIC nos encontramos en el banco 0.

movlw B'01101000' Cargamos en el registro w el número binario 01101000 con el cual vamos a configurar el registro OPTION_REG que es donde vamos a seleccionar algunas opciones importantes para el funcionamiento del emu.

movwf OPTION_REG OPTION_REG es un registro especial que nos sirve para configurar algunos parámetros en el funcionamiento del pic y esto va a depender del valor de cada bit:

· BITs 0, 1, 2. Bits PS0, PS1, PS2. Estos bits sirven para definir la tasa de división del predivisor , esta tasa difiere dependiendo de que el predivisor este asignado al temporizador 0 o al watchdog.

· BIT 3 o bit PSA (Prescaler Assignment). Si este bit está a cero, el predivisor está asignado al temporizador 0. Si está a uno estará asignado al watchdog.

· BIT 4 o bit RTE (Timer 0 signal Edge). Si este bit está a cero, el contenido del registro TMR0 se incrementará en un flanco de subida de la señal aplicada al pin RA4/ToCK1; si está a uno, se incrementará con el flanco de bajada.

· BIT 5 o bit RTS (Timer 0 signal source). Si este bit está a cero, el temporizador utilizará el reloj interno. Si está a uno, utilizará la señal aplicada al pin RA4/T0CK1.

· BIT 6 o bit INTEDG (Interrupt edge). Este bit define el sentido del flanco que provocará la interrupción a través del pin externo INT. Un uno activa el flanco de subida y un cero un flanco de bajada.

· BIT 7 o bit RBPU (RB Pull Up enable). Si está a cero, este bit activa las resistencias de pull up a nivel alto, previstas en la entrada del puerto B.

movlw B'11111110' Ahora almacenamos el numero binario 11111110 en w para pasarlo al registro TRISA, es decir vamos a configurar nuestro puerto A, en este caso tendremos solamente el primer bit (RA0) como salida. En este caso es la salida de datos de nuestro Emu.

movwf TRISA Se almacena en TRISA

movlw B'11111111' Configuramos nuestro puerto B como de entrada totalmente.

movwf TRISB

bcf STATUS,RP0 Cambiamos de banco (Banco 0) que es odnde se encuentran nuestros registros PORTA PORTB.

clrf PORTA Limpiamos el contenido de nuestro puerto A

clrf PORTB Limpiamos el contenido de nuestro puerto B

bsf PORTA,CRD_DTA Ponemos en 1 la salida de datos del emu.

clrf FLAGS Limpiamos la banderas. El registro FLAGS fué definido como un registro de propósito general, este va a ser usado en algún punto del programa principal.

btfss PORTB,RESTOREFUSES Checamos el estado del microswitch que se encuentra en el puerto B para realizar la recarga de la tarjeta.

goto RESFUSCNT En caso de estar oprimido el boton nos vamos a la rutina de restauración de bits.

En este punto se va a iniciar el proceso de lectura de la EEPROM

movlw 0x02 Cargamos en w el numero 02 hex es decir 0000 0010.

movwf EEADR Se almacena el número 2 en EEADR que es la dirección de la EEPROM

bsf STATUS,RP0 Cambiamos de banco de trabajo para poder manipular el registro EECON1

bsf EECON1,RD Ponemos el bit RD del registro EECON1 en 1 para poder habilitar la lectura de la EEPROM

bcf STATUS,RP0 Nos cambiamos de banco

movfw EEDATA Y leemos directamente el valor almacenado en la dirección EEADR (2) de la EEPROM a travez de EEDATA y lo almacenamos en el registro W. Aquí terminamos el proceso de lectura de EEPROM

btfss STATUS,Z Checamos el estado del bit z del registro STATUS. Esto significa que debemos checar si la operación lógica anterior generó un cero. En caso de que no se haya generado saltara una posicion. Es decir, debemos checar si el valor almacenado en la dirección 02 de la EEPROM es cero, en caso contrario saltamos una posición. Si no nos vamos a CONTPROGRAM

goto CONTPROGRAM Saltamos a CONTPROGRAM

Aqui se inicia un proceso de escritura en la EEPROM

movlw 0xFF Cargamos en w el número hexadecimal FF, 1111 1111 binario

movwf EEDATA Y lo almacenamos en EEDATA

bsf STATUS,RP0 Cambiamos de banco

bsf EECON1,WREN Habilitamos el proceso d escritura de la EEPROM por medio de wren del registro EECON1

movlw 0x55 Este es un proceso que se debe seguir para poder escribir dentro de la EEPROM

movwf EECON2

movlw 0xAA

movwf EECON2

bsf EECON1,WR En este punto ya se escribió el valor ff en la direccion que apunta el EEADR en la EEPROM

Lo siguiente describe la manera en que se realiza **el proceso de escritura**. (Copiado de un Doc. que anda por ahí)

La escritura en esta memoria, que consiste en realidad en una programación, ya que estamos trabajando con una EEPROM, es algo mas complicada por evidentes razones de seguridad. Este proceso se debe desarrollar de la forma siguiente:

Escritura de la dirección en la que desea escribir en el registro EEADR.

Escritura del dato en el registro EEDATA.

Ejecución de un programa tipo:

MOVLW 55

MOVWF EECON2

MOVLW AA

MOVWF EECON2

BSF EECON1,WR

Esta última instrucción inicia el proceso de escritura. Cuando termina, el bit EEIF está a uno y si la última instrucción EEPROM ha sido activada por medio del bit EEIE del registro INTCON, se genera dicha interrupción. El bit EEIF debe ponerse a cero por software.

El programa consiste en escribir 55 seguido de AA en el registro EECON2 como medida de seguridad , con el fin de evitar que un programa que tenga un despiste pueda programar la EEPROM por accidente, manipulando simplemente los bits del EECON1.

El registro de control EECON1 contiene un determinado número de bits de control, cuyo significado es el que sigue:

BIT 0 o bit RD (Read Data). Este bit debe ponerse a uno para leer un dato. El circuito lo pone automáticamente a cero.

BIT 1 o bit WR (Write data). Este bit debe ponerse a uno para escribir un dato. El circuito lo pone automáticamente a cero.

BIT 2 o bit WREN (Write Enable). Si este bit está a cero, impide cualquier escritura en la EEPROM. Debe ponerse a uno para autorizar la escritura.

BIT 3 o bit WRERR (Write Error). Este bit se pone a uno si se produce un error de escritura después de una parada prematura (reset o watchdog). En estas condiciones, los contenidos de EEDATA y EEADR no varían, para permitir una repetición correcta de la operación.

BIT 4 o bit EEIF (EEPROM Interupt flag). Este bit se pone a uno al terminar una escritura y debe ponerse a cero por software

El registro EECON2 no tiene existencia física, por lo que no se puede leer, tan solo sirve para el proceso de protección en escritura detallado anteriormente.

En el estudio anterior nos quedamos en el final del proceso de escritura en la EEPROM Por lo que seguiremos con la culminación del mismo.

waitresflg Esta es una rutina de espera para saber si el bit WR del registro EECON1 ya cambió automaticamente a cero, debemos recordar que este bit debe ser puesto a 1 para habilitar

el modo de escritura en la EEPROM. El camibio de este bit lo hace automáticamente el PIC pero debemos saber cuando lo haga.

btfsc EECON1,WR En caso de que ya haya cambiado nos saltamos una instrucción. Y en caso contrario seguimos esperando.

goto waitresflg Seguimos esperando.

bcf STATUS,RP0 Nos vamos al banco 0

Se supone que al llegar hasta aquí tenemos guardado en la EEPROM el valor ff en la direccion 02
CONTPROGRAM

clrf EEADR Limpiamos lo que contenía el EEADR (Apuntador de dirección de la EEPROM.) En este momento la dirección que apunta el header es cero.

bsf STATUS,RP0 Nos vamos al banco 1

bsf EECON1,RD ; Set EECON1 Read Data Flag Habilitamos el bit de lectura de EEPROM

bcf STATUS,RP0 ; Access register bank 0 Nos vamos al banco 0

movfw EEDATA ; Read EEPROM fused units counter Leemos lo contenido en la EEPROM en la direccion cero y lo almacenamos en nuestro registro w.

movwf FUSCNT Almacenamos el valor de lo que hay en EEPROM en la direccion cero (0) en FUSCNT.*****

incf EEADR,1 Incrementamos nuestra dirección de EEPROM
bsf STATUS,RP0 Cambiamos a banco 0
bsf EECON1,RD Habilitamos lectura de EEPROM
bcf STATUS,RP0 Cambiamos a banco 1
movfw EEDATA Leemos lo que hay en EEPROM y almacenamos en w
movwf CRDCNT Y lo descargamos a CRDCNT, por lo tanto CRDCNT contendrá lo que hay en la EEPROM en la direccion uno (1)

Aquí iniciamos un proceso de operación binaria. Comom sabemos, la rotación de bits a la izquierda nos genera una multiplicación por dos del byte que se esté rotando. Cabe aclarar que el bit que sale afectado con esto es el bit CARRY del registro STATUS, y también se le incluye en la operación, es por eso que antes de rotar los bits se debe limpiar.

bcf STATUS,C Limpiamos el bit carry del registro STATUS
rlf CRDCNT,1 Rotamos a la izquierda el contenido de CRDCNT (Multiplicación por dos)
rlf CRDCNT,1 Una vez mas lo rotamos a la izquierda (Multiplicación por dos)
movfw CRDCNT Y lo almacenamos en el registro w Aquí se almacena la operación binaria (CRDCNT*2)+(CRDCNT*2)= 4(CRDCNT)
rlf CRDCNT,1 Rotamos una vez mas CRDCNT y tenemos la operación anterior por dos, es decir: 4(CRDCNT)*2 y esto nos da el resultado de 8(CRDCNT)
addwf CRDCNT,1 Y sumamos lo que teníamos en el registro w y lo almacenamos en CRDCNT, es decir: en el registro w teníamos 4(CRDCNT) y sumándole 8(CRDCNT) Tenemos el valor de 12(CRDCNT) En el registro CRDCNT, esto para que nos sirve???, Pues bueno, en principio teníamos almacenado en el registro CRDCNT el valor de la EEPROM en la direccion 1, y cual es ese valor????
****

movlw CRD_ID Cargamos el registro w con el valor de CRD_ID

movwf FSR Y lo almacenamos en FSR. FSR va a apuntar en la dirección donde se encuentra CRD_ID, debemos recordar que este registro es donde se va almacenar nuestro HEADER y consta de 12 BYTES

movlw .12 Cargamos 12 en w
movwf LOOPCNT ; Store in LOOPCNT Ylo guardamos en el registro LOOPCNT. LOOPCNT como su nombre lo indica, es un contador de lazo cerrado, con este registro vamos a contar los 12 bytes del header.

Este punto es importante ya que es aquí donde almacenaremos nuestro HEADER en memoria para que pueda funcionar nuestro emu. Primero que nada tenemos que cambiar nuestro HEADER por uno de nosotros.

COPYHEADER

movlw HIGH CARDID ; Because CARDID table is on Page 1 Aquí vamos a implementar una tabla de datos que será nuestro HEADER

movwf PCLATH Almacenamos el valor del registro w en PCLATH
movfw CRDCNT Cargamos en w el valor de CRDCNT
call CARDID Llamamos a la subrutina de CARDID, que es donde se encuentra nuestro HEADER, en esta rutina que se verá mas adelante nos daremos cuenta de como regresa nuestro dato cargado con un byte de nuestro HEADER por medio de la instrucción RETLW.

movwf INDF Instrucción de direccionamiento indirecto la cual nos indica que el valor que traemos de retorno se almacenará en la RAM con el valor del apuntador FSR
incf FSR,1 Incrementamos el FSR para tener el siguiente valor del HEADER en la siguiente dirección de RAM.
incf CRDCNT,1 Incrementamos el valor del CRDCNT en una posición.
decfsz LOOPCNT,1 Decrementamos el LOOPCNT en una posición y saltamos una instrucción en caso de que sea cero, si no es así regresamos a la rutina de COPYHEADER para traer más bytes y almacenarlos en la RAM.
goto COPYHEADER Regresamos.

bsf STATUS,RP0 Cambiamos de banco.
bcf EECON1,EEIF Limpiamos el bit EEIF del registro EECON1, este bit se pone a uno al terminar una escritura y se pone a cero por softwre.
bcf EECON1,WREN Ponemos a cero el bit WREN el cual inhabilita el modo de escritura en la EEPROM.
bcf STATUS,RP0 Cambiamos de banco.
movlw B'10010000' Cargamos en w un número binario que nos va a servir para configurar interrupciones.

movwf INTCON Registro INTCON (Configuración de interrupciones) donde cada bit significa algo. La siguiente info la saqué de un libro que encontré por ahí.

BIT 0 o bit RBIF (RB Interrupt Flag). Si se pone a 1, este bit indica un cambio de estado en una de las líneas de RB4 a RB7 del puerto B.

BIT 1 o bit INTF (Interrupt Flag). Si se pone a 1, este bit indica una interrupción provocada por la línea RB0/INT del puerto B.

BIT 2 o bit T0IF (Timer 0 Interrupt Flag). Si se pone a 1, este bit indica un desbordamiento del temporizador 0.

BIT 3 o bit RBIE (RB Interrupt Enable). Si se pone a uno este bit, autoriza las interrupciones provocadas por un cambio de estado de las líneas RB4 a RB7 del puerto B.

BIT 4 o bit INTE (Interrupt Enable). Si se pone a 1, este bit autoriza las interrupciones provocadas por la línea RB0/INT del puerto B.

BIT 5 o bit T0IE (Timer 0 Interrupt Enable). Si está a uno, este bit autoriza las interrupciones debidas al desbordamiento del temporizador 0.

BIT 6 o bit EEIE (EEPROM Interrupt Enable). Si está a uno, este bit autoriza las interrupciones que proceden de la memoria EEPROM de datos.

BIT 7 o bit GIE (Global Interrupt Enable). Si se pone a uno, este bit autoriza todas las interrupciones que estén enmascaradas mediante sus bits individuales de activación, a cero las inhibe.

Una vez terminado el almacenamiento del HEADER vamos a la rutina MAIN. Esta rutina va a depender del cambio de estado en el ciclo de reloj externo. En este caso esta configurado con OPTION_REG y tiene una interrupción en el ciclo de subida de Clock. Si sube el Clk nos vamos a la rutina de interrupciones. En caso de que el clk no suba seguimos con el proceso. Aclaremos que se está habalndo del Clock de la cabina y no del PIC.

MAIN **bsf STATUS,RP0** Vamos al banco 1

btfsc EECON1,WR Esta rutina es similar a una que vimos anteriormente. Es una rutina de espera con la cual no pasaremos a la siguiente instrucción si no cambia autométicamente el bit WR de EECON1

goto MAIN Regresamos aMAIN

bcf EECON1,WREN Limpiamos el bit WREN de INTCON. Anteriormente ya se dió el significado de este Bit.

bcf STATUS,RP0 Cambiamos de banco.

btfss FLAGS,EE_FLAG Checamos el bit EE_FLAG del registro FLAGS.**

goto MAIN Si sigue en cero regresamos a MAIN.

bcf FLAGS,EE_FLAG Limpiamos el contenido de este bit. FLAGS???

Proceso de escritura

clrf EEADR Limpiamos el contenido del EEADR
movfw FUSCNT Cargamos en w lo que contiene el registro FUSCNT
movwf EEDATA Y lo almacenamos en EEDATA
bsf STATUS,RP0 Cambiamos al banco 1
bcf INTCON,GIE Limpiamos GIE
bsf EECON1,WREN Ponemos a 1 WREN e iniciamos el proceso de escritura
movlw 055 Proceso de escritura.
movwf EECON2 Proceso de escritura.
movlw 0aa Proceso de escritura.
movwf EECON2 Proceso de escritura.
bsf EECON1,WR Con esto Habilitamos otra vez el proceso de escritura
bsf INTCON,GIE Habilita todas las interrupciones enmascarables.
bcf STATUS,RP0 Vamos al banco 0
goto MAIN Regresamos a MAIN

Esta es la última parte del programa. Esta parte en sí está diseñada para la recarga del emu, es decir, para generar nuevos HEADERS (No quiere decir que los va a generar de la nada, pero no estaría mal, de hecho en

España encontraron la fórmula del checksum de las tarjetas y con esto podían generar números de serie y HEADERS de tarjetas de manera aleatoria). Aclaremos que los HEADERS aquí puestos deberan ser cambiados por otros que sean válidos y que nosostros debemos leer de tarjetas agotadas.

Recordemos que en la rutina INIT teníamos una instrucción con la cual estariamos checando el estado del microswitch que se encuentra en RB3, esta instrucción nos decía btfssPORTB,RESTOREFUSES. En caso de estar oprimido nos iriamos a RESFUSCNT. Pues aquí estamos.

RESFUSCNT

movlw 0x02 Cargamos el valor 2 en w
movwf EEADR Lo almacenamos en la direccion que apunta el EEADR
bsf STATUS,RP0 Cambiamos de banco
bsf EECON1,RD Habilitamos la lectura de EEPROM
bcf STATUS,RP0 Cambiamos de banco
movfw EEDATA Leemos el dato y lo almacenamos en w. Estas seis instrucciones anteriores nos sirvieron para leer lo que contiene la direccion 2 de la EEPROM y almacenarlo en el registro w.

btfsc STATUS,Z Checamos que lo contenido en w no sea cero.
goto INFLOOP En caso de serlo nos vamos al final.

clrf EEADR Limpiamos la dirección EEADR. Recuerde que se encontraba apuntando a la dirección 2.
movlw DEFAULTFUSCNT Cargamos el valor de w en el registro DEFAULTFUSCNT, recuerde que w tiene el valor que estaba almacenado en la dirección 2 de la EEPROM.

movwf EEDATA Iniciamos proceso de escritura en la EEPROM
bsf STATUS,RP0 Cambiamos de banco
bsf EECON1,WREN Habilitamos escritura
movlw 0x55 Proceso de escritura
movwf EECON2 Proceso de escritura
movlw 0xAA Proceso de escritura
movwf EECON2 Proceso de escritura
bsf EECON1,WR Proceso de escritura

waitwrfuscnt Esperamos a que se ponga a cero el bit WR de EECON1
btfsc EECON1,WR Espera
goto waitwrfuscnt Regresamos y esperamos más.

bcf STATUS,RP0 Cambiamos de banco
incf EEADR,1 Incrementamos el EEADR. (Dirección que apunta la EEPROM)Direccion 1
bsf STATUS,RP0 Cambiamos de banco
bsf EECON1,RD Habilitamos lectura
bcf STATUS,RP0 Cambiamos de banco
movfw EEDATA Leemos lo que hay en la EEPROM
incf EEDATA,1 Y lo incrementamos
movlw MAXCARDS Cargamos el valor de MAXCARDS. El registro MAXCARDS fué creado para definir el número de tarjetas que pueden ser emuladas con este programa. El autor propone 21 pero puede ser cambiado a 2, 3, etc. dependiendo de cuantos HEADERS quieres almacenar.

subwf EEDATA,0 Restamos el valor que hay en EEDATA a MAXCARDS, esto con el fin de saber si ya se acabaron nuestros HEADERS a emular.
btfsc STATUS,C Checamos el estado de carry. Recordemos que en caso de que EEDATA fuera mayor a MAXCARDS entonces el bit c de STATUS se pondrá a uno. En caso de que sea 1 seguimos con la siguiente instrucción.

clrf EEDATA Con esta instrucción reseteamos nuevamente el valor de la posicion de la EEPROM que nos indica cuantas tarjetas se han agotado. E iniciamos nuevamente un proceso de escritura en la EEPROM donde pondremos el valor 0 para que inicie desde el HEADER 1. Cabe aclarar que el valor de las tarjetas agotadas se va almacenando en la dirección 1 de la EEPROM.

```
        bsf    STATUS,RP0      Proceso de escritura.
        bsf    EECON1,WREN     Proceso de escritura.
        movlw  0x55            Proceso de escritura.
        movwf  EECON2          Proceso de escritura.
        movlw  0xAA            Proceso de escritura.
        movwf  EECON2          Proceso de escritura.
        bsf    EECON1,WR       Proceso de escritura.
waitwrcrdcnt                   Esperamos que se complete el proceso de escritura.
        btfsc  EECON1,WR       Esperamos a que el bit WR se ponga otra vez en cero.
        goto   waitwrcrdcnt    Regresamos y esperamos mas.


        bcf    STATUS,RP0      Cambiamos de banco.
        incf   EEADR,1         Incrementamos el EEADR
        clrf   EEDATA          Borramos lo que se encuentre el la dirección  2 de la EEPROM.
        bsf    STATUS,RP0      Iniciamos proceso de escritura.
        bsf    EECON1,WREN
        movlw  0x55
        movwf  EECON2
        movlw  0xAA
        movwf  EECON2
        bsf    EECON1,WR
INFLOOP goto   INFLOOP         LOOP Infinito.
```

Y finalmente tenemos los valores de los HEADERS, en este caso solo pondré uno para ahorrar espacio.

org 0x100 Esto es importante. Nos indica el origen de la tabla donde se almacenará el HEADER?

CARDID addwf PCL,1 Subrutina CARDID. Nos indica que debemos sumar el valor de w y PCL y almacenarlo en PCL

```
        retlw  0xB6            ;1
        retlw  0x80
        retlw  0xF2            Con retlw tenemos la opción de regresar de la subrutina con el valor
        retlw  0x00            actual cargado en w.
        retlw  0x69
        retlw  0x25
        retlw  0xC5
        retlw  0x5A
        retlw  0x46
        retlw  0x4C
        retlw  0x32
        retlw  0x59
```

CONCLUSIÓN.

Cabe aclarar que se está dando una explicación a grandes rasgos del funcionamiento del emu de Cartman. **En ninguno de los casos se desea influir sobre la construcción del emu con fines de lucro**, solamente con el fín de investigación. Como se habrán dado cuenta, yo no soy un experto en programación de PIC pero con este escrito se me han aclarado muchísimas dudas (En realidad todavía tengo algunas). Pero espero que esto les sirva para a los que apenas, como yo, se inician en esto y también espero que estos conocimientos sean la base para emular las de 128. De ante mano agradecería sus correcciones y comentarios.

Es importante conocer acerca de lógica digital, electrónica digital y analógica para poder desarrollar este proyecto. No es sencillo, de hecho yo mismo lo pensé pero me dí cuenta que hay un sin fin de cosas nuevas por aprender. Y aunque parece fácil requiere de tiempo, pero al ver los resultados.....

Gracias
BK
Para correcciones y dudas que esten a mi alcance por favor:
Illan_ivanovich@yahoo.com

TEORIA DEL "TURBO"

A quien no le gustaría tener en sus manos un MODEM potente como el que nos ofrece Telmex, no te gustaría navegar a 512 bps ?. Pues ese servicio a 512 Bps no lo tendremos por el momento, dado a que es carísimo contar con un servicio "ADSL" en estos momentos tan escasos que vivimos (o no?)

Pero si podemos navegar a 128 bps !!!!! Que como ??? Pues sencillo .

Según nuestro MODEM viaja a 56 bps , pero no es cierto verdad , pero quien nos puede ofrecer ese servicio "turbo" con esa velocidad a 128bps ?? Pues Telmex.

Ahora bien estas dispuesto a pagar 950 pesos de inicio, y eso que según por que hay que cambiar tu línea a una línea digital, pagar tu renta mensual de 200 pesos o 300 según sea el caso y por si fuera poco 399 pesos(Ja Ja Ja Hasta eso no nos redondean a 400 pesos, que chido no ?) Al mes por la línea Turbo. Que robo tan descarado de estos señores !!!!

Pues el equipo de investigación de la "timmysquad" (aja como no) investigó y salió a preguntar a los "técnicos" de Telmex con respecto a este servicio.

Y descubrió este .....como decirlo sin herir esos sentimientos tan lindos que tienes .....mas bien seria .....este maldito fraude.

Para empezar el servicio turbo consta de un MODEM "ISDN",con un disquete de instalación que contiene el controlador del MODEM (que obvio no, perdón),el cambio de cableado, por que eso si, te cambian el cable que llega desde la calle hacia tu casa por un cable reforzado, que según es para que levante la línea a mayor capacitancia (ja ja y más ja ja,no es cierto), y también un puente que hacen desde la central para la conversión a línea digital.

Pues que buena onda no .....??

Que ilusos llegamos hacer a veces, el tal puente que hacen en la central nunca se hace (pero por que si acabamos de leer que si?), Pues no lo hacen dado que todas las línea están convertidas a digitales desde que están dadas de alta cuando la contratas, a verdad .....entonces por que crees que marcamos a tonos y no a pulsos como antes ?

Pues que sigue.....que nos han visto la cara !!!! El MODEM es el que hace todo !!!!!! Así es,como leiste,el MODEM realiza todo este relajo de la velocidad a 128 Kbps, por el cableado ese quesque reforzado ni te preocupes,la línea aguanta eso

y mas(bueno en caso de que tengas tu cableado ya de a tiro bien jodido pues si hay que cambiarlo,ni modo).

Ahora viene lo bueno....pide a un "tecnico" que te venda un MODEM de esos Turbo,y dejame decirte que si te lo vende no te lo va a vender mas caro a comparacion de los precios de otros MODEM externos que puedes conseguir en centros de computo,si es asi,pide que te venda todo,los cables que se conectan a tu pc y el disco con los controladores.

Dejame decirte este chisme,esos modem te los pueden conseguir ya que los tienen arrumbados en la bodega,dado que empresas que tenian ese servicio de "turbo" ,cambiaron a servicio "infinitum" y pues los modems "turbo" ya son obsoletos,pues los tienen ya abandonados en la bodega (que ojetes no?).

Esta teoria la neta no la eh probado,ya que no eh podido conseguir ese dichoso modem "turbo",pero no dudo que funcione,por que la neta para empezar,el
Modem es el que nos hace todo esto de la velocidad y no la línea,asi que de todas maneras no cuesta nada conseguirlo y probarlo,para eso estamos.

A lo mejor y no creen esto,has de decir "como nos dice esto si ni si quiera

Lo a probado",pues no nos hagamos tontos ,pues todos sabemos de que pie cojea telmex.

Esto solo es una teoria dejenme volver a aclararlo.

Gracias

La hora de los heroes a llegado.
Http://mx.geocities.com/timmysquad

Adios

Timmy
“Timmysquad”

TIPOS DE CENTRALITAS

Mucha gente cree que solo existen dos tipos de las centralitas llamadas comunmente "analogicas" y "digitales" pero lo cierto es que telefonica a usado y usa los siguientes modelos (ordenados de mayor a menor antiguedad):

Sistemas rotativos:
- 7A1, 7A2 y 7B

Sistemas de barras cruzadas convencionales:
- Pentaconta-1000
- Pentaconta-32
- ARF
- ARM (version del ARF para centrales de transito)

Semielectronicos:
- Pentaconta-2000
- ARE
- Metaconta

Electronicos digitales:
- AXE (Fabricante Ericsson)
- 1240 (Fabricante Alcatel)
- 5ESS (Fabricante AT&T)

Actualmente casi todas las centrales instaladas son las de tipo electronico digital, aunque todavia quedaran un pequeño numero desemielectronicas del tipo pentaconta, las rotary desaparecieron hace cosa de 3 años o asi.

Como el paso de la tipica telefonia "analogica" a la actual RDSI y linea multiservicio (Llamada a tres, en espera, CAR, etc) se a seguido una estrategia de evolucion segun el esquema siguiente:

- Centralitas analogicas unidas por enlaces analogicos
- Aplicacion de tecnicas PCM para enlazarcentrales mateniendo señalizacion analogica a MF.
- Señalizacion por mensajes CCITT N§ 7. La comunicacion con el abonado sigue siendo analogica.
- Paso a RDSI siendo todo el sistema telefonico totalmente digital.

TIPOS

Existen muchos tipos de boxes, aunque las masconocidas sean la blue, la red, la black y la beige. Aqui pongo una relacion de todas las que yo he podido encontrar -hay algunas muy interesantes-

;):

NOTA: he omitido algunas boxes que he encontrado, ya que aportaban muy poca informacion sobre si mismas, o eran similares a otras.

ACRYLIC BOX: Otorga servicios como llamada en espera o llamada a 3 a la linea que la usa, quitandoselos a la linea situada inmediatamente al lado en el cajetin. Para sistemas antiguos de 4 cables.

AQUA BOX: Su finalidad es evitar el pinchado telefonico que realiza el FBI sobre algunas lineas.

BEIGE BOX: Consiste en conectar un telefono que lleves encima a uno de los muchos cajetines de telefonos que hay repartidos por todos lados para llamar gratis (eso si utilizas las lineas de prueba, sino se la cobran a alguien -a ti no, desde luego-).

BLACK BOX: Es un circuito electronico que se conecta a la linea y permite que todas las llamadas entrantes (los que te llaman, vamos) sean gratuitas.

BLAST BOX: Funciona como amplificador del microfono del telefono.

BLUE BOX: La box por excelencia. Reproduce los tonos exactos que utilizan los sistemas de la Bell, entre ellos el famoso de 2600 Hz.

BROWN BOX: Crea un sistema de partyline a partir de 2 lineas existentes. Tambien llamada Party o Pink (si son + de 2 lineas).

BUD BOX: Pincha el telefono de otra linea.

BUSY BOX: Elimina el tono de marcacion de una linea telefonica, evitando asi que pueda llamar o recibir llamadas.

CHARTREUSE BOX: Da acceso al sistema electrico de la linea para tus propios fines.

CHEESE BOX: Provoca que el telefono se comporte como una cabina.

COLOR BOX: Permite grabar conversaciones telefonicas.

COPPER BOX: Provoca interferencias de lineas cruzadas.

DARK BOX: Redirige llamadas entrantes o salientes a otro telefono.

GOLD BOX: Permite rastrear una llamada, comprobar si estan rastrando o cambiar un rastreo. Existe una nueva versión llamada New Gold.

GREEN BOX: Genera codigos como el de devolución de monedas o el de llamada a cobro revertido.

MAGENTA BOX: Conecta una linea telefonica remota a otra.

MAHOU BOX: Circuito que anula la se¤al de 12 kHz que activa los contadores de pasos evitando asi que pagemos la llamada. Uso exclusivo en España.

NEON BOX: Conecta a la linea un micrófono externo.

PAISLEY BOX: Permite apoderarse de lineas reservadas a operadores de las compañias telefonicas.

PEARL BOX: Genera tonos en la linea telefonica.

PURPLE BOX: Actua como un boton de bloqueo en el telefono.

RAINBOW BOX: Evita un rastreo de llamada enviando 120 voltios por la línea.

RED BOX: Emula el sonido de introducir monedas en las cabinas, asi la (porque suele ser LA) operadora cree que se han introducido monedas.

ROCK BOX: Añade musica a la linea.

SCARLET BOX: Provoca interferencias en la linea para que la recepcion sea muy pobre.

STATIC BOX: Eleva el voltaje de la linea, manteniendolo alto.

TRON BOX: Provoca que el contador electrico de una casa funcione mas despacio.

WHITE BOX: Juego de botones DTMF transportable.

YELLOW BOX: Añade una linea de extension.

REALIDAD

Llegados a este punto existen ciertos "puntososcuros" que habria que aclarar:

La box mas "aconsejable" es la beige, por varias razones:
No requiere conocimientos de electronica, no es necesario ni salir de casa para encontrar los componentes necesarios y ademas telefonica (telmex) nos ha hecho un favor sembrando telefonos publicos en los rincones mas reconditos del mapa, para poder hacer todas las pruebas que se nos plazcan.

-SI es facil que telmex(fuck) te agarre si tienes linea digital. Te ReCoMiEnDo QuE sIeMpReQuE kE uSeS cUaLqUiEr BoX tOmEs ToDaS lAs PrEcAuCiOnEs pOsIbLeS.¡

-Muchas (por no decir la mayoria) boxes de las arriba comentadas estan diseñadas para funcionar en el sistema de EEUU. Existen boxes diseñadas particularmente para funcionar en MEXICO¡

-Si alguien quiere los esquemas de montaje de las diferentes boxes, que busque por inet: los hay a MADRES en paginas del tema...

-Los nombres de las boxes pueden ser muy subjetivos: lo que uno puede llamar red box, para otro puede ser black box, y viceversa. Siempre han habido confusiones en torno a esto; el dia que se regulen las boxes... ;)

-Un punto que no es estrictamente del articulo, pero siempre viene bien tocarlo: en casi todos los articulos de montaje de algo relacionado con el phreak (en su mayoria boxes) siempre se habla de "cable rojo, cable verde, etc"; luego al abrir el telefono no hay cables de esos colores, o no estan donde deberian estar, y ya la hemos jodido!

Lo que realmente hay que hacer es experimentar: experimentar con cualquier cosa que tengamos al alcance de la mano; que pone "verde" y es blanco, pues a probar !, que ademas probando siempre se aprenden cosas; si luego te sale, pues escribes algo explicandolo todo, que siempre le ira bien a alguien con menos conocimientos (aplicable a TODO) rEcUeRdA kE lO qUe aSe AuN vErDaDeRo pHreaK.. eS Su lmPeTuD dE aPrEnDeR y De ExPrImEnTaR aSi QuE sOlO aSlo Y nOlo PiEnSeS MáS sOlo AcTuA. ASTA LA PROXIMA.¡

K@b®On smok123mx@yahoo.com.mx

- CAPITULO III -

Muy buenas a todos, seguimos en la lucha tratando de comprender un poco mas de lo que estamos desarrollando, en este capitulo veremos las secuencias de RESET, CLOCK y como se lleva a cabo a través de estas señales la lectura de datos para que la cabina acepte la tarjeta.

La noticia:

Como noticia reciente, me ha llegado un rumor y que pude comprobar exitosamente con respecto a los emuladores de 256 bits para Argentina, como sabemos todos, por lo menos en la parte donde yo vivo, las cabinas de esta compañía (**TELEFÓNICA**) eran compatibles con tarjetas de 256 bits pero de su empresa competidora (**TELECOM**) que en la actualidad esta ultima compañía funcionan con estas tarjetas de primera generación, como yo poseía un emulador, cuando se produjo el parche, en la fecha mencionada del primer capitulo, me dejo de funcionar, pero me ha llegado un rumor a mis oídos de que poniéndole el número de serie de una tarjeta de **TELECOM** mi tarjeta volvería a ser valida. Realice los cambios y que paso???, mi tarjeta marcaba en la pantalla de LCD el valor de $2,30...

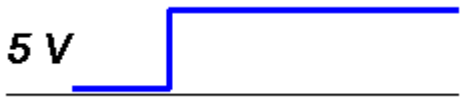
Pero bueno ya que este documento trata de tarjetas de segunda generación, no volveremos a tocar el tema de las viejas cards, amenos que se presente una noticia como la que hablado anteriormente...

Aceptando la tarjeta:

La cabina realiza ciertas secuencias donde la tarjeta debe responder a lo emitido por la cabina. Esto vendría siendo como un juego de preguntas y respuestas...

Voy a aclarar que hablaremos a partir de ahora un poco más técnico, es decir cuando yo ponga nivel alto o nivel activo estaremos hablando de 5 V, y cuando yo mencione nivel bajo o nivel inactivo estaremos hablando de 0 V

Una señal de nivel alto seria:



Una señal de nivel bajo seria:

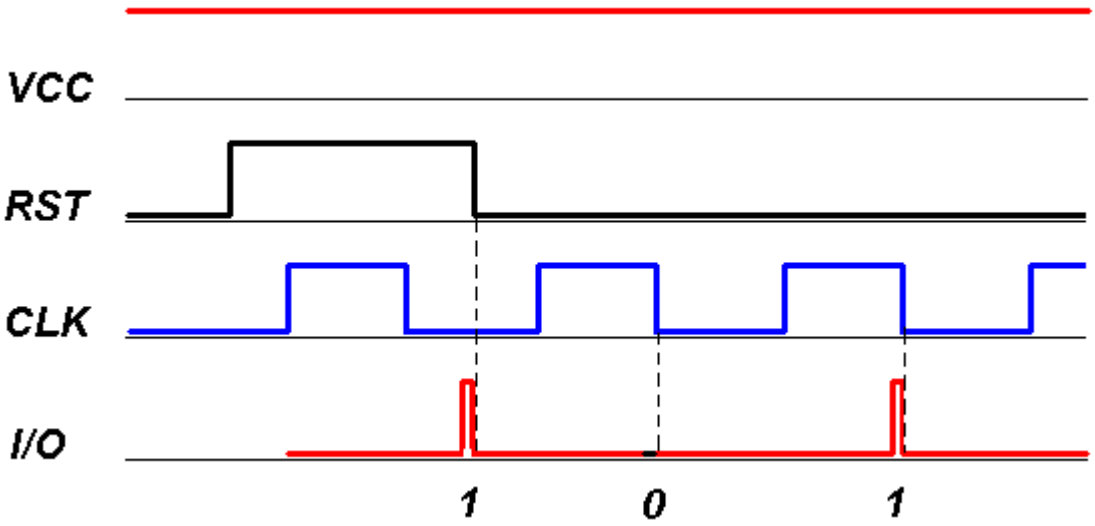
Un pulso se representaría:

La parte roja vendría siendo un flanco ascendente y la parte negra un flanco descendente

La cabina alimentará la tarjeta hasta que ella responda todo lo que tenga que responder, una vez echo esto la cabina dejara de alimentarla hasta que decida que es lo que va volver hacer.

La cabina entonces va aplicar un nivel alto en la patita RST (de la chapita) y va a permanecer activo hasta que se aplique otro nivel alto pero esta vez en la patita CLK, cuando suba CLK el contador de direcciones sé resetea, es decir va a comenzar de la primera dirección (Recuerden que tenemos 16 direcciones, es decir 16 bytes para leer) una vez echo esto CLK debe bajar antes que RST, y posteriormente RST baja sacando el primer dato por I/O. el primer dato es el primer BIT del primer byte, explicándolo un poco mejor, nuestro primera dirección o primer byte es A1 (10100001) y nuestro primer BIT es el que esta por el lado izquierdo de dicho byte es decir el 1. Luego RST va a permanecer bajo, y clock va a realizar 127 pulsos más, por cada bajada del pulso va aparecer por I/O su BIT correspondiente.

Es decir, por ejemplo si quisiéramos sacar los tres primeros bits del primer byte, la secuencia vendría siendo de la siguiente manera:



Y así sucesivamente dando pulsos de clock hasta sacar todos los bits, luego cuando lleguemos al BIT 128 el siguiente pulso de clock, quedara apuntando al primer BIT del primer byte.

Programando en ensamblador

Muchos de ustedes tal vez no sepan ensamblador para programar el microcontrolador PIC16F84, pueden estudiar de un libro, o agarrar el datashett de www.microchip.com o leer el informe de o0ShellGhost0o del zine de Mexican Hackers Mafia.

Estas rutinas que desarrollare son para que se den una idea de cómo se puede realizar un emulador programando un microcontrolador

Por ejemplo:

Para realizar el reset de la tarjeta podemos hacer lo siguiente:
Para apuntar a los bytes usamos el registro EEADR que vendría siendo el contador de direcciones, y para apuntar a los bits usamos un registro CONBIT que lo definiremos en la dirección 0x0C. Hagamos de cuenta que a nuestro PIC le configuramos las patinas RST en la RB0, a CLK en la patita RB7 y a I/O en la patita RA4.

```
;********************************** RUTINA DE RESET ***************************************

RESET       btfss     PORTB , 0     ; RST se encuentra en nivel alto???
            goto      RESET         ; No, seguimos esperando
ESP_CLK     btfss     PORTB , 7     ; Si, y subió CLK???
            goto      ESP_CLK       ; No, seguimos esperando a que suba
            clrf      EEADR         ; CLK = 1, reseteamos el contador de bytes
            clrf      CONBIT        ; Reseteamos el contador de bits
            movlw     0xA1          ; El registro EEDATA se carga con el valor A1
            movwf     EEDATA

ESP_RST     btfsc     PORTB , 0     ; Bajo RST ???
            goto      ESP_RST       ; No, seguimos esperando
            bsf       PORTA , 4     ; Si, sacamos el primer bit del byte A1
            goto      CLOCK         ; Saltamos a la rutina clock para seguir
                                    ; sacando bits

;********************************** FIN DE LA RUTINA RESET ****************************
```

Esto fue un claro ejemplo de cómo se debe programar un microcontrolador para poder realizar un emulador para tarjetas de 128 bits. Es obvio que esta no es la única manera para poder llevar a cabo dicho reset, cada programador tiene su forma y estilo de programar.....

Bueno eso fue todo hasta el momento, espero haberles aclarado sus dudas e ideas, nos veremos en los próximos capítulos donde seguiremos explicando un poco mas de este mundillo.

Nota

Cualquier error por favor comunicarse a: tecnicoinforma@yahoo.com.ar
Este documento puede ser editado en cualquier página web siempre y cuando respeten el contenido y a su autor.

Autor : PitufoEnrike
País : MENDOZA – ARGENTINA
Creado el : 31 / 10 / 2001
Sitio Oficial : http://www.tecnicos.da.ru

- CAPITULO IV -

Aloha!!! Ya falta poco para llegar a la totalidad de este documento que estoy desarrollando, espero que lo explicado anteriormente lo ayas entendido o me sentiré como un profesor frustrado.

Bueno, adelantando lo que se viene vamos a ver como se llevan a cabo las rutinas de grabación **(Write)** y de transporte de acarreo **(WriteCarry).** Es decir, una vez que la cabina halla terminado de aceptar la tarjeta y luego invitarnos a marcar él numero al que hay que llamar, se queda esperando hasta que la llamada se realice, cuando nos atienden del otro lado la cabina realizara un procedimiento de descuento de bits que puede ser de tipo **write** o **writecarry**...

Operación WRITE

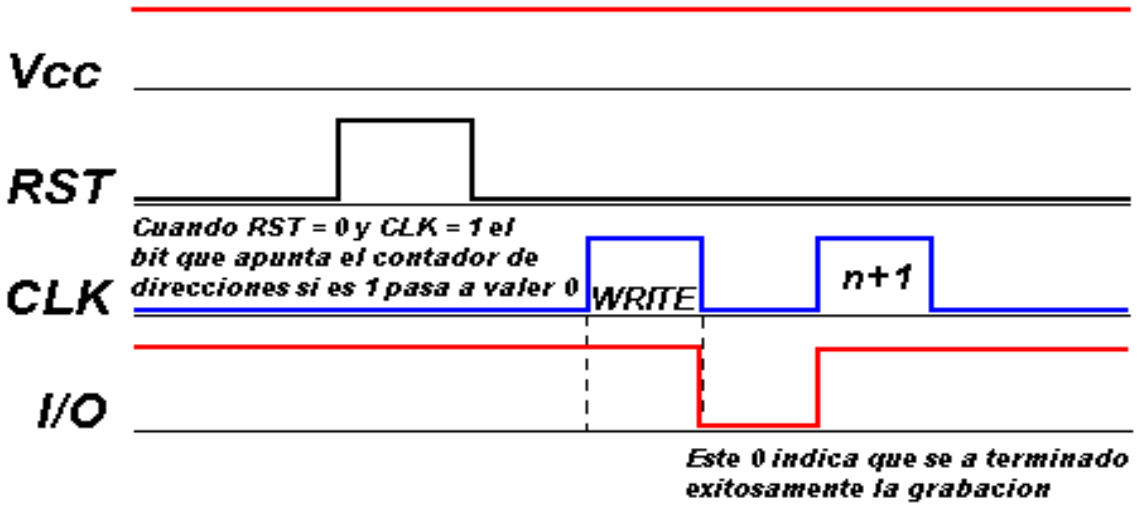
Debo aclarar que los bytes del 0 al 7 jamás deben escribirse, cuando la cabina intenta escribir en estos bytes la tarjeta debe bloquearse, es decir no debe pasar absolutamente nada hasta que se produzca un corte en la alimentación.
La cabina solamente graba en el contador octal.

El descuento de bits es básicamente muy sencillo, se trata de pasar los bits 1 a 0 y para que esto se lleve a cabo, la cabina realizara una secuencia con las señales mencionadas en el capitulo anterior (RESET y RELOJ).

1 - **RST** se levanta mientras que **CLK** permanece inactivo, para invalidar en un pulso el incremento del contador de direcciones. Luego de esto **RST** debe bajar.

2 - CLK se pone a 1 un mínimo de 10 ms y en ese momento nosotros debemos pasar el bit que esta a 1 y ponerlo a 0 y luego esperar que CLK baje y sacar por I/O un 0 y así la máquina interpreta que la grabación se llevo a cabo correctamente, luego de esto vuelve a leer la tarjeta y comprueba que ese bit fue grabado.

3 - En el siguiente pulso de **CLK**, se incrementará el contador de direcciones en uno, y es posible repetir la secuencia de escritura para escribir en la siguiente posición de memoria.



Programando un poco..

Pasemos un poco a la práctica

Hagamos de cuenta que la cabina leyó en su totalidad la tarjeta y que quiere grabar en el bit que a ella mas le guste, por ejemplo:

Tenemos el siguiente contador octal

DIRECCIÓN	BYTE en HEXADECIMAL	BYTE en BINARIO
08	00	0000 0000
09	00	0000 0000
0A	7F	0111 1111
0B	3F	0011 1111
0C	0F	0000 1111

En la dirección 0A quiere grabar el bit 2. Entonces la cabina una vez realizada la lectura y decidida grabar el bit 81...

Bueno pero como sabemos que quiere grabar ???, es decir como lo implementaríamos en nuestro programa ???

Bueno en mi forma y no quiere decir que es la única manera de lograrlo lo he implementado en la rutina de RESET de la siguiente manera

```
;********************************************** RUTINA DE RESET ***************************************
RESET               btfss    PORTB , 0      ; Subio RST ???
                    goto     RESET          ; No, seguimos esperando
                    bcf      PORTA , 4      ; Si, limpiamos la salida
ESPERAR_CLOCK2      btfsc    PORTB , 7      ; Subio clock ???
                    goto     RESETEAR       ; Si, reseteamos principalmente
                    btfsc    PORTB , 0      ; Bajo RESET ???
```

```
                goto    ESPERAR_CLOCK2      ; No, seguimos esperando
                btfsc   EEDATA , 7      ; Leemos el bit 7 de EEDATA
                bsf     PORTA , 4       ; El bit 7 es un 1 y lo sacamos por I/O
                goto    WRITE           ; Saltamos a grabar

;************************************* FIN DE LA RUTINA RESET ***************************************
```

Como vemos, preguntamos si subió RST, si no lo ha hecho esperamos hasta que se ejecute, una vez que se ejecuta debemos poner un 0 en la salida para hacerle creer a la maquina que no se hace nada, luego verificamos si subió CLK , si se ejecuto saltamos a la rutina RESETEAR en donde ponemos a 0 todos los registros, ya sea contadores de bits, de programas, flags, etc. Si CLK sigue bajo preguntamos si RST bajó, si no se llevo a cabo esto seguimos esperando a que baje, pero si bajo verificamos si lo que quiere grabar es un 0 o un 1, si es un 1 le respondemos a la cabina haciendo sacar un 1 por I/O (recuerden que nuestra salida la habíamos configurado en RA4) y luego saltamos a la rutina de grabación, y si es un 0 también saltamos a la rutina grabación.

```
;************************************* RUTINA DE WRITE ************************************************
WRITE           btfss   PORTB , 7       ; Subio CLOCK ??? ( quiere grabar )
                goto    WRITE           ; No, seguimos esperando hasta que suba
                btfss   EEDATA , 7      ; El bit que quiere grabar es un 0 o un 1 ???
                goto    NOGRABA         ; Es un cero entonces no grabamos nada
                movf    EEDATA , W      ; W se carga con el valor de EEDATA
                movwf   RETENER         ; Retenemos este valor
                bsf     STATUS , RP0    ; Accedemos al banco 1
                bsf     EECON1 , RD     ; Leemos
                bcf     STATUS , RP0    ; Volvemos al banco 0
                movf    CONBIT , W      ;  W se carga con el valor de CONBIT
                addwf   PCL , F         ; CONBIT se lo sumamos al contador de
                                        ; programas y saltamos a grabar el bit
                                        ; correspondiente
                goto    BIT0
                goto    BIT1
                goto    BIT2
                goto    BIT3
                goto    BIT4
                goto    BIT5
                goto    BIT6
                goto    BIT7

BIT0            bcf     EEDATA , 7      ; Ponemos a 0 el bit 7
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
BIT1            bcf     EEDATA , 6      ; Ponemos a 0 el bit 6
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
BIT2            bcf     EEDATA , 5      ; Ponemos a 0 el bit 5
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
BIT3            bcf     EEDATA , 4      ; Ponemos a 0 el bit 4
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
BIT4            bcf     EEDATA , 3      ; Ponemos a 0 el bit 3
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
BIT5            bcf     EEDATA , 2      ; Ponemos a 0 el bit 2
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
BIT6            bcf     EEDATA , 1      ; Ponemos a 0 el bit 1
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
BIT7            bcf     EEDATA , 0      ; Ponemos a 0 el bit 0
                goto    GRABAR          ; Saltamos a grabarlo en la eeprom
```

Antes de seguir vamos a explicar lo que desarrollé:

Primero lo que hacemos es verificar si subió CLK, si no subió seguimos esperando hasta que suba, cuando sube significa que quiere grabar, entonces preguntamos si lo que quiere grabar es un 0 o un 1, si es un cero saltamos a la rutina NOGRABAR que ya la hemos explicado, si lo que quiere grabar es un 1 entonces lo que hacemos es poner el valor de EEDATA en un registro RETENER que lo tendremos que definir por ejemplo en la dirección 0x0E, esto lo hacemos porque cuando ya hemos grabado tenemos que volver a la dirección antes de grabar, luego leemos la dirección que la cabina asignó, es decir leemos la posición que indica el contador de direcciones EEADR. El siguiente paso es saber en que bit la cabina se posicionó, entonces lo que hacemos es pasar a nuestro registro de trabajo (W) el valor que tiene el contador de bits CONBIT y luego sumárselo al contador de programas para así luego saltar dentro de la tabla y pasar a 0 el bit correspondiente de EEDATA, acuérdense que a nuestro contador de bits lo incrementamos, por ejemplo, si el contador de bits es 0 estaríamos apuntando al bit 7 de EEDATA, si el contador de bits es un 1, estaríamos apuntando al bit 6 de EEDATA y así sucesivamente. Como ven en nuestra tabla una vez pasado a 0 con la instrucción bcf el bit correspondiente de EEDATA luego saltamos a la rutina GRABAR, donde en ella grabamos el nuevo valor de EEDATA en la dirección que apunta el contador de direcciones EEADR, no la explicaré por que se puede encontrar en cualquier tutorial de ensamblador para pic.

Luego seguimos de la siguiente manera:

```
ESPERAR_CLKBAJO   btfsc    PORTB , 7          ; Bajo CLK ???
                  goto     ESPERAR_CLKBAJO    ; No, seguimos esperando
                  bcf      PORTA , 4          ; Si, sacamos un 0 por I/O
                  goto     CLK                ; Saltamos a CLOCK

NOGRABA           btfsc    PORTB , 7          ; Bajo CLOCK ???
                  goto     NOGRABA            ; No, seguimos esperando
                  goto     CLK                ; Saltamos a CLOCK

;******************************** FIN DE LA RUTINA DE WRITE  ****************************************
```

Y bueno básicamente es como dice la teoría, una vez terminado de grabar verificamos si bajo CLOCK, si bajo sacamos el 0 por I/O y saltamos a esperar mas pulsos de CLOCK.

Se me olvido!!!!, antes de saltar a esperar pulsos debemos pasar el valor del registro RETENER a EEDATA. (acuérdense que la cabina se va a posicionar en la dirección donde estaba antes).

Como ya me canse de escribir voy a dar por concluido esta parte del informe, en el próximo capitulo explicaré como se lleva a cabo el writecarry..., entonces aprovechen a estudiar todo lo que e explicado así en nuestro próximo capitulo están mas cancheros en entender las cosas ;oPPP

Nota

Cualquier error por favor comunicarse a: tecnicoinforma@yahoo.com.ar

Este documento puede ser editado en cualquier página web siempre y cuando respeten el contenido y a su autor.

Autor : PitufoEnrike
País : MENDOZA – ARGENTINA
Creado el : 16 / 11 / 2001
Sitio Oficial : http://www.tecnicos.da.ru

Seminario de Programación
(tercera parte)
por XROLEX

Teléfonos incluidos en esta edición del eZine:

NOVATEL PTR800/PTR825
OKI 700
OKI 710/892/910/1145
OKI 750/892/900/1150
PANASONIC HP600
PHILIPS FIZZ (ANALOGICO)
PHILIPS ISIS (ANALOGICO)
QUALCOMM Q PHONE (DUAL)
QUALCOMM QCP 820 (CDMA)/QPHONE
QUALCOMM GLOBAL STAR (SATALITAL, CDMA 800, ANALOGICO)
RADIO SHACK 17/3001

NOVATEL PTR800/PTR825 (ANALOGICOS)

- **OBTENCION DE LA SERIE:**

DIRECTAMENTE EN ETIQUETA DE PARTE POSTERIOR DEL TELEFONO, ESTA EN DECIMAL Y EMPIEZA CON 142

- **PARA ENTRAR A PROGRAMACION:**

PARA EL MODELO PTR 800 ENCENDER Y TECLEAR **FCN FCN *626776***
PARA EL MODELO PTR 825 ENCENDER Y TECLEAR **FCN FCN *697201***

APARECE EL NUMERO DE VERSION DE SOFWARE COMO AVISO DE QUE SE ENTRO A PROGRAMACION.

- **PARA GRABAR LOS DATOS:**

CON VOLUMEN HACIA ARRIBA HASTA NAM NUMBER	SELECCIONAR **1-4 # VOL UP**
SYSTEM ID	**01525 # VOL UP**
SCM	NO CAMBIA **VOL UP**
TEL	**10 DIGITOS TEL VOL UP**
INITIAL PAGING CH A	**0333 # VOL UP**
INITIAL PAGING CH B	**0334 # VOL UP**
INITIAL PAGING CH	**0333 # VOL UP**
ACCESS OVERLOAD	**02 # VOL UP**
GROUP ID	**00 # VOL UP**
FULL LOCK A	**123 # VOL UP**
RESTRICTED LOCK B	**000 # VOL UP**
LOCAL USE	**1 # VOL UP**
MIN OPT	**1 # VOL UP**

AUNQUE HAY MAS PASOS NO ES CONVENIENTE MODIFICAR MAS, DE AQUÍ YA PUEDE SALIRSE DE PROGRAMACION.

- **PARA SALIR DE PROGRAMACION:**

FCN END

- **PARA PONER EN SISTEMA:**

FCN 53

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

FCN 7 *

OKI 700 (ANALOGICO)

- **OBTENCION DE LA SERIE:**

EN LA PARTE POSTERIOR DEL TELEFONO HAY UNA ETIQUETA CON CODIGO DE BARRAS, BAJO ESTE CODIGO HAY UN GRUPO DE CARACTERES ENCERRADOS ENTRE DOS ASTERISCOS, TOMAR LOS ULTIMOS 7 U 8 DIGITOS (HASTA DONDE HAY UNA LETRA) Y SE LE ANTEPONE 1290 ó 129.

- **PARA ENTRAR A PROGRAMACION:**

ENCENDER Y EN MENOS DE 10 SEGUNDOS PRESIONAR RCL MENU SIMULTANEAMENTE, EL TELEFONO NO DEBE MOSTRAR NADA AL HACER ESTO, SOLTAR Y TECLEAR **0008693427**, APARECE LA SERIE EN HEXADECIMAL, ESTO ES MUY RAPIDO A VECES NO SE ALCANZA A VER EL NUMERO COMPLETO, LUEGO APARECE LA VERSION, AHI SE PRESIONA **CLR** Y APARECE PRESS 1 TO 9, PRESIONAR **1**

- **PARA GRABAR LOS DATOS:**

APARECE EL TEL	**10 DIGITOS DE TEL STO VOL UP**
SIDH	**01525 STO VOL UP**
IPCH	**333 STO VOL UP**
ACCOLC	**02 STO VOL UP**
LOCK	**1234 STO VOL UP**
GROUP ID	**00 STO VOL UP**
SCM	**1010 STO VOL UP**
OPTION	NO MODIFICAR **VOL UP**
SEC	**ULTIMOS 6 DIGITOS DEL ESN STO VOL UP**
SIDH	

- **PARA SALIR DE PROGRAMACION:**

CLR CLR APAGAR Y ENCENDER

- **PARA PONER EN SISTEMA:**

MENU VARIAS VECES HASTA SYSTEM ALGO, CON **RCL** BUSCAR SYSTEM A ONLY **STO**

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

RCL #

OKI 710/890/910/1145 (ANALOGICOS)

- **OBTENCION DE LA SERIE:**

EN LA PARTE POSTERIOR DEL TELEFONO HAY UNA ETIQUETA CON CODIGO DE BARRAS, BAJO ESTE CODIGO HAY UN GRUPO DE CARACTERES ENCERRADOS ENTRE DOS ASTERISCOS, TOMAR LOS ULTIMOS 7 U 8 DIGITOS (HASTA DONDE HAY UNA LETRA) Y SE LE ANTEPONE 1290 ó 129.

- **PARA ENTRAR A PROGRAMACION:**

ENCENDER, PRESIONAR SIMULTANEMENTE LAS TECLAS **RCL** Y **FUNC**, EL TELEFONO NO DEBE MOSTRAR NADA (COMO SI NO SE HUBIERAN PRESIONADO LAS TECLAS), TECLEAR LUEGO CUALQUIERA DE LAS SIGUIENTES CLAVES PARA VER CUAL RESULTA, PARA CADA INTENTO SE DEBE APAGAR Y VOLVER A ENCENDER EL TELEFONO.

***12345678#**, **1234567890**, **0123456789**, EL NUMERO DE SERIE SIN EL CUARTO DIGITO, **POR EJEMPLO** SI LA SERIE ES 129**0**1234567 QUITANDO EL CUARTO DIGITO QUE ES UN **CERO** LA CLAVE SERIA 1291234567. SI NINGUNA DE ESTA CLAVES FUNCIONA ACUDIR A SERVICIO TECNICO.

SI AL TERMINAR DE TECLEAR **RCL FUNC** Y LA CLAVE APARECE UN NUMERO DE CUATRO DIGITOS QUE LUEGO CAMBIA A OTRO DE 8 DIGITOS CON LA TECLA DE VOL BUSCAR DONDE APAREZCA PHON Y EL NUMERO QUE TIENE ACTUALMENTE.

- **PARA GRABAR LOS DATOS:**

PHON	**10 DIGITOS DEL TEL STO VOL UP**
SC	ULTIMOS 6 DIGITOS DE LA SERIE **VOL UP**
OPTN	NO MODIFICAR **VOL UP**
SCM	NO MODIFICAR **VOL UP**
ULOC	**1234 STO VOL UP**
GID	**00 STO VOL UP**
ACCOLC	**02 STO VOL UP**
IPCH	**0333 STO VOL UP**
SID	**01525 STO VOL UP**
PHON	

- **PARA SALIR DE PROGRAMACION:**

CLR CLR APAGAR Y ENCENDER

- **PARA PONER EN SISTEMA:**

FUNC 2, CODIGO DE SEGURIDAD DE 6 DIGITOS, CON **2** HASTA ONLY A **STO CLR**

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

RCL #

OKI 750/892/900/1150 (ANALOGICOS)

- **OBTENCION DE LA SERIE:**

EN LA PARTE POSTERIOR DEL TELEFONO HAY UNA ETIQUETA CON CODIGO DE BARRAS, BAJO ESTE CODIGO HAY UN GRUPO DE CARACTERES ENCERRADOS ENTRE DOS ASTERISCOS, TOMAR LOS ULTIMOS 7 U 8 DIGITOS (HASTA DONDE HAY UNA LETRA) Y SE LE ANTEPONE 1290 ó 129.

- **PARA ENTRAR A PROGRAMACION:**

ENCENDER, PRESIONAR SIMULTANEMENTE LAS TECLAS **RCL** Y **MENU**, EL TELEFONO NO DEBE MOSTRAR NADA (COMO SI NO SE HUBIERAN PRESIONADO LAS TECLAS), TECLEAR LUAGO CUALQUIERA DE LAS SIGUIENTES CLAVES PARA VER CUAL RESULTA, PARA CADA INTENTO SE DEBE APAGAR Y VOLVER A ENCENDER EL TELEFONO.

***12345678#**, **1234567890**, **0123456789**, EL NUMERO DE SERIE SIN EL CUARTO DIGITO, **POR EJEMPLO** SI LA SERIE ES 129**0**1234567 QUITANDO EL CUARTO DIGITO QUE ES UN **CERO** LA CLAVE SERIA 1291234567. SI NINGUNA DE ESTA CLAVES FUNCIONA ACUDIR A SERVICIO TECNICO.

SI AL TERMINAR DE TECLEAR **RCL MENU** Y LA CLAVE APARECE UN NUMERO DE CUATRO DIGITOS QUE LUEGO CAMBIA A OTRO DE 8 DIGITOS CON LA TECLA DE VOL BUSCAR DONDE APAREZCA NAM 1 PROGRAM ESPERAR 2 SEGUNDOS PARA QUE APAREZCA EL NUMERO QUE TIENE ACTUALMENTE, SI ENTRA A OTRO NAM DAR **CLR** UNA VEZ Y CON **VOL** BUSCAR EL NAM 1 PROGRAM.

- **PARA GRABAR LOS DATOS:**

OWN#	10 DIGITOS TEL STO VOL UP
SC	ULTIMOS 6 DIGITOS DE LA SERIE **VOL UP**
OPTN	NO MODIFICAR **VOL UP**
SCM	NO MODIFICAR **VOL UP**
UNLOCK	**1234 STO VOL UP**
GIM	**00 STO VOL UP**
ACCOLC#	**02 STO VOL UP**
IPCH NO.	**0333 STO VOL UP**
SYSTEM OWN#	**01525 STO VOL UP**

- **PARA SALIR DE PROGRAMACION:**

CLR CLR APAGAR Y ENCENDER

- **PARA PONER EN SISTEMA:**

MENU VARIAS VECES HASTA ADM MENU **RCL 6 DIGITOS DE SEGURIDAD** (ESN) **VOL** HASTA QUE APARECE ALGO DE SELECCIÓN DE SISTEMA, CON RCL BUSCAR A ONLY **STO CLR**

PARA EL 1150 MENU VARIAS VECES HASTA QUE APARECE SEC MENU, RCL 4 DIGITOS DE CANDADO CON VOL HASTA ALGO DE SELECCIÓN DE SISTEMA, CON RCL HASTA A ONLY Y STO CLR.

- **PARA VERIFICAR EL NUMERO DE TELEFONO:RCL #**

PANASONIC HP600 (ANALOGICO)

- **OBTENCION DE LA SERIE:**

EL TELEFONO TIENE UNA ETIQUETA EN EL ESPACIO DONDE VA LA BATERIA, AHI HAY UN NUMERO DEL TIPO N-OPQRST, A PARTIR DE ESTE NUMERO SE OBTIENE LA SERIE EN DECIMAL DE LA SIGUIENTE FORMA:

EL NUMERO N SE MULTIPLICA POR 262144 Y AL RESULTAD SE LE SUMA EL OPQRST; POR EJEMPLO: SI EL NUMERO ES 4-042540, LA OPERACIÓN SERIA:

4X262144 = 1048576 + 042540 = 1091116 Y LA SERIE ES 13601091116

- **PARA ENTRAR A PROGRAMACION:**

ENCENDER, TECLEAR RAPIDAMENTE *0#0*0#0*1 APARECE NAM 1 MODE
EN LOS PRIMEROS TELEFONOS DE ESTE MODELO, SE REQUIERE UN ADAPTADOR PARA PROGRAMACION, SI DE LA MANERA ANTERIOR NO ENTRA, RECURRIR A SERVICIO TEC.

- **PARA GRABAR LOS DATOS:**

01525	STO 01
EL TELEFONO DE 10 DIGITOS	STO 02
1	STO 03
333	STO 04
02	STO 05
00	STO 06
32	STO 07
10	STO 08
1234 (CANDADO)	STO 10
11001000	STO 11
00010010	STO 12
11110111	STO 13

- **PARA SALIR DE PROGRAMACION:**

STO **

- **PARA PONER EN SISTEMA:**

SE PROGRAMO DE MODO QUE SIEMPRE ESTE EN SOLO A, NO HAY MANERA DE CAMBIARLO.

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

F 9

PHILPS FIZZ (ANALOGICO)

- **OBTENCION DE LA SERIE:**

TRAE UNA ETIQUETA EN LA PARTE POSTERIOR CON LA SERIE EN DECIMAL E INICIA CON 228

- **PARA ENTRAR A PROGRAMACION:**

ENCENDER, PRESIONAR DE INMEDIATO **#** Y MANTENER SOSTENIDO HASTA QUE APAREZCA **MODO PROVEED .** PRESIONAR LA TECLA DEBAJO DEL INDICADOR **OK,** LA PANTALLA MOSTRARA **PROVEEDOR.** EN SEGUIDA INTRODUCIR LA SIGUIENTE SECUENCIA DE NUMEROS **13325371** Y PRESIONAR **OK,** LA PANTALLA MOSTRARA **CODIGO CORRECTO,** LUEGO **NAM1,** POSTERIORMENTE PRESIONE **OK.** A CONTINUACION INTRODUCIR CADA UNO DE LOS SIGUIENTES PARAMETROS COMO SE INDICA PRESIONANDO **END-C** PARA BORRAR Y **OK** PARA GRABAR Y AVANZAR

- **PARA GRABAR LOS DATOS:**

SOS	911	OK	
PREFIJO INT	011		OK
NUMERO PROPIO	524 XXX XXXX	OK	
SIS PREF	A	OK	
IDS	01525	OK	
CANAL CONT	0333		OK
CANAL AVIS	0333	OK	

- **PARA SALIR DE PROGRAMACION:**

A CONTINUACION LA PANTALLA MUESTRA **NAM1** ALMACENADO, SEGUIDO DE **NAM 1, OTRO?** PARA TERMINAR PRESIONE **NO** SEGUIDO DE **SALIR**

- **PARA PONER EN SISTEMA:**

MENU 25 OK ^ HASTA ***SOLAMENTE A*** **OK SALIR, SALIR.**

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

MENU 17 OK

PHILIPS ISIS (ANALOGICO)

- **OBTENCION DE LA SERIE:**

TRAE UNA ETIQUETA EN LA PARTE POSTERIOR CON LA SERIE EN DECIMAL E INICIA CON 228

- **PARA ENTRAR A PROGRAMACION:**
- LA PROGRAMACION SE INICIA PRESIONANDO Y SOSTENIENDO LAS TECLAS DE ENCENDIO **(I)** Y **4** SIMULTANEAMENTE HASTA QUE APAREZCA **PHILIPS** EN EL DISPLAY, EN SEGUIDA SE INTRODUCE **6267** Y OPRIMIR **---V** PARA AVANZAR.

- **PARA GRABAR LOS DATOS:**

524 XXX XXXX	**V**
01525	**V**
011	**V**
A	**V**
524 XXX XXXX	**V**
333	**V**
333	**V**
334	**V**
1	**V**
911	**V**
02	**V**
ON	**V**
ON	**V**

- **PARA SALIR DE PROGRAMACION:**

OPRIMA SEND

- **PARA PONER EN SISTEMA:**

MENU SOSTENIDO POR 2 SEG. HASTA QUE APAREZCA UNA LLAVE, ENSEGUIDA PRESIONAR CODIGO DE SEGURIDAD (DE FABRICA 0000), DESPUES VVVVV <C HASTA A FINALMENTE PRESIONAR END

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

MENU VVVV.

QUALCOMM Q PHONE (DUAL)

- **OBTENCION DE LA SERIE:**

TRAE UNA ETIQUETA EN LA PARTE POSTERIOR CON EL NUMERO DE SERIE ELECTRONICO DIRECTO EN DECIMAL Y EN HEXADECIMAL E INICIA CON 179 O B3.

- **PARA ENTRAR A PROGRAMACION:**

PRESIONE **MENU 40** Y ENSEGUIDA **000000,** APARECERA EL NUMERO DE SERIE ELECTRONICO.

- **PARA GRABAR LOS DATOS:**

AL ENTRAR A PROGRAMACION APARECERA EL ESN PRESIONAR		**OK**
PHONE NUMBER	**10 DIGITOS DE NUMERO**	**OK**
SIDH	**01525**	**OK**
NAM1 NAME	**----------**	**OK**
NAM PROGRAMING	**PRESIONAR**	**EXIT**

SI DESEA UN 2do. NAM EN LUGAR DEPRESIONAR **EXIT** PRESIONE **MORE** AVANZAR CON **OK** HASTA *NAM 2 SETTINGS* Y PRESIONAR **EDIT**

INGRESAR **10 DIGITOS DE NUMERO** Y PRESIONAR **OK** INGRESAR SIDH (**01525**, PARA REGION 6) AVANZAR CON **OK** HASTA *BASIC NAM 2 PROGRAMMING IS SOMPLETE* Y PRESIONAR **EXIT**. SI DESA OTRO NAM SELECCIONAR **EDIT** Y SEGUR MISMA SECUENCIA.

- **PARA SALIR DE PROGRAMACION:**

AL TERMINAR DE INTRODUCIR LOS PARAMETROS PRESIONAR EL BOTON DE EXIT

- **PARA PONER EN SISTEMA:**

MENU **41**, CON NEXT SE SELECCIONA **A ONLY** Y CON **OK** SE GRABA.

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

PRESIONAR TECLA **i**

- **PARA CAMBIAR DE NAM**: PRESIONAR **MENU 42** Y CON **NEXT** SELECCIONAR NAM Y FIJARLO CON **OK**.

QUALCOM QCP-820 (DUAL)

OBTENCION DE LA SERIE:

TRAE UNA ETIQUETA EN LA PARTE POSTERIOR CON LA SERIE EN DECIMAL E INICIA CON **159**

- **PARA ENTRAR A PROGRAMACION:**

INTRODUZCA LA SIGUIENTE SECUENCIA, **111111** , EN SEGUIDA PRESIONAR EL DIAL 2 VECES, DIGITE **000000.**

INTRIDUCIR LOS DATOS UNO A UNO Y PRESIONAR EL DIAL PARA GRABAR Y AVANZAR

- **PARA GRABAR LOS DATOS:**

AL INICIO APARECE EL ESN			PRESIONAR DIAL OK
PHONE#		10 DIGITOS DE NUMERO	PRESIONAR DIAL OK
HOME SIDH	01525		PRESIONAR DIAL OK
NAME	---------		PRESIONAR DIAL OK
BASIC NAM 1			PRESIONAR DIAL OPTIONS
OPTIONS		SELECCIONAR EXIT	PRESIONAR DIAL EXIT

- **PARA SALIR DE PROGRAMACION:**

  EN OPTIONS SELECCIONAR EXIT Y PRESIONAR DIAL

- **PARA PONER EN SISTEMA:**

  PRESIONAR DIAL EN **FEATURES** Y EN SEGUIDA **76**, SELECIONAR HOME SIDE GIRANDO EL DIAL Y ACEPTAR PRESIONANDO EL DIAL

- **PARA VERIFICAR EL NUMERO DE TELEFONO:**

  PRESIONAR DIAL EN **FEATURES** Y EN SEGUIDA **31**

QUALCOMM GLOBALSTAR
(SATELITAL, CDMA 800, ANALOGICO)

MANUAL DE PROGRAMACION UT

I. Encender UT
II. Oprimir Menú 88
III. Seleccionar (con las teclas de volumen laterales) GStar Pref y OK
IV. Desplegar antena Satelital
V. Oprimir End.
VI. Continuar con pasos siguientes (A, B…):

A. Oprimir Menú 800
B. Ingresar código de acceso (de fábrica el valor es 000 000)
C. Selecciona el NAM a programar: para NAM 1 oprimir 2; para NAM 2 oprimir 3

Continuar con los siguientes parámetros:

N°	En Pantalla	Valor a Introducir	Avanza con
1	NAM1 Phone #		
000-000-0000	525-XXX-XXXX (Donde X es el Número Celular)		Ok
2	NAM 1 Home SID		
0	01525 (Para números activados en el D.F.)		Ok
3	NAM 1 Name	Con teclado, Nombre del Carrier Celular	Ok
4	Basic NAM 1 Programming is Complete	Nada	Options
5	NAM 1 Options 1 A-key Entry 2 More Prog		
3 Exit	2	Avanza automáticamente	
6	NAM 1 MCC		
393	000	Ok	
7	NAM 1 NMSID		
00525XXXXXX	Nada	Ok	
8	NAM 1 CDMA Directory #		
525-XXX-XXXX	Nada	Ok	
9	NAM 1 CDMA Home SID 1		
0	01525	Ok	
10	NAM 1 CDMA Home NID 1		
0	999	Ok	
11	NAM 1 CDMA Home SID 2		
0	01525	Ok	
12	NAM 1 CDMA Home NID 2		
0	65535	Ok	
13 NAM 1 CDMA Home SID 3			
0	0	Ok	
14	NAM 1 CDMA Home NID 3		
0	65535	Ok	
15	NAM 1 CDMA Home SID 4		
0	0	Ok	
16	NAM 1 CDMA Home NID 4		
0	65535	Ok	
17	NAM 1 CDMA Primary CH A		
_694	283	Ok	

N°	En Pantalla	Valor a Introducir	Avanza con
18	NAM 1 CDMA Second CH A		
_000	691	Ok	
19	NAM 1 CDMA Primary CH B		
_000	384	Ok	
20	NAM 1 CDMA Second CH B		
_000	777	Ok	
21	NAM 1 CDMA HomeSys Term		
Yes	Scroll con las teclas de volumen hasta yes (Banda A) ¿o No (para Banda B)?		Ok
22	NAM 1 CDMA FornSID Term		
Yes	Scroll con las teclas de volumen hasta yes (Banda A) ¿o No (para Banda B)?		Ok
23	NAM 1 CDMA FornNID Term		
Yes	Scroll con las teclas de volumen hasta yes (Banda A) ¿o No (para Banda B)?		Ok
24	NAM 1 AMPS Directory #		
XXX-XXX-XXXX	Nada	Ok	
25	NAM 1 AMPS Home SID		
0	01525	Ok	
26	NAM 1 AMPS 1st Page CH		
_0	333 (Banda A) ó 334 (Banda B)		Ok
27	NAM 1 AMPS 1st Control CH A		
_0	333	Ok	
28	NAM 1 AMPS 1st Control CH B		
_0	334	Ok	
29	NAM 1 AMPS Auto-reg		
Hide	Scroll con las teclas de volumen hasta enable, entonces seleccionar con ok		Ok
30	NAM 1 Lock Out SID 1	Nada	Ok
31	NAM 1 Acc. Overload Class		
Y (Donde Y es el último dígito del número celular)	Nada	Ok	
32	NAM Program		
1 ESN (Sólo vista)			
2 Cell NAM 1			
3Cell NAM 2			
4 G* Svc Pgm			
5 More Program	End	Nada	

D. El teléfono muestra en pantalla "REBOOTING" y se reinicializa.

E. Para poner en banda: Oprimir Menú 893

F. Seleccionar con las teclas de Volumen la banda Deseada :

+ Automatic (para banda A Digital)
+ A Only (para banda A exclusivo)
+ B Only (Para banda B Telcel)
+ No roaming (Para restringir Servicio a la región celular local)

H. Oprimir OK cuando haya seleccionado la opción deseada

I. Para poner en Español, oprimir Menú 727

Fin. De programación.

Despedida

Aquí termina nuestra tercera entrega del único eZine dedicado exclusivamente al phreaking y la electrónica underground en México, espero que les haya gustado.

En el transcurso de este año puede que haya una infinidad de cambios en cuanto al grupo, la pagina, el ezine, etc... Recuerden entrar regularmente a hackers.com.mx y escuchar AcidRadio los viernes en la tarde para enterarse de las noticias de la MHM.

Por cierto, para la cuarta ezine vamos a organizar un pequeño concurso para que diseñen la portada, daremos más noticias en la página.