UCT

UNIDAD CONTROLADA TELEFÓNICAMENTE

Boris Estudiez

0.1: Versiones del Proyecto:

Nombre de proyecto:	**UCT** **U**nidad **C**ontrolada **T**elefónicamente.
Fecha de inicio	**12/Sep/2003**
Ultima Actualización	**23/Jan/2005**
UCT DOCUMENT **V**ERSION	**DV-2.0**
UCT HARDWARE **V**ERSION	**HV-0.6**
UCT SOFTWARE **V**ERSION	**SV-0.8.1**
UCT MESSAGES **V**ERSION	**MV-1.0**

0.2: Autor:

Autor: Boris Estudiez

E-mails:

- stk@freeshell.org
- slicetex@hotpop.com
- boris_estudiez@yahoo.com.ar

Web-site:

- http://stk.freeshell.org (pagina oficial del proyecto)
- http://geocities.com/slicesoft (mirror)

Licencia:

Denegación de Responsabilidad:

El proyecto fue probado en la practica y funciona, pero el autor no se hace responsable por las fallas del mismo y los daños que pueda ocasionar su uso/construccion/etc., solo es de carácter experimental y/o educacional.

0.3: Historia:

El proyecto UCT fue presentado como practico final para regularizar la materia Técnicas Digitales II del nivel IV de la carrera Ingeniería Electrónica de la Universidad Tecnológica Nacional, Facultad Regional Córdoba.

0.4: Notas Sobre el Documento:

El presente informe fue escrito suponiendo conocimientos avanzados de electrónica por parte del lector, por lo tanto por simplificación se omiten muchas demostraciones y

explicaciones que deben ser buscadas en libros u hojas técnicas de los componentes electrónicos.
La parte de software no se explica en este documento ya que es demasiada extensa y esta abierta a modificaciones continuas. Usted debe bajar de la pagina del proyecto el software necesario para que la parte de hardware del UCT funcione.
Es probable que por cuestiones de tiempo el documento no sea actualizado frente a cambios de los circuitos electrónicos (hardware), software y mensajes de audio del proyecto.
Al momento de escribir el documento las versiones de hardware, software y mensaje de audio, eran las listadas en la sección **0.1** del documento.
El proyecto fue probado exitosamente en las líneas telefónicas de la compañía telefónica **Telecom de Córdoba-Argentina**, si usted vive en otras partes del mundo quizás deba hacer pequeñas modificaciones en los circuitos del proyecto debido a que las **normas telefónicas pueden ser distintas**. Sin embargo el proyecto fue pensado para funcionar en casi cualquier condición.

0.5: Índice:

El documento se divide en las siguientes secciones:

- **0.x: Notas del autor.**
- **1.x: Introducción.**
- **2.x: Funcionamiento general del UCT.**
- **3.x: Desarrollo y funcionamiento interno del UCT.**
- **4.x: Notas de Construcción.**
- **5.x: Apéndice / Anexo**

1.0: Introducción:

El proyecto UCT (Unidad Controlada Telefónicamente) fue desarrollado con el objetivo de poder comandar equipos, maquinas, computadoras, electro-domésticos, luces y cualquier otro tipo de dispositivo de forma remota, mediante un teléfono conectado a la línea telefónica, del cual parten las ordenes.

El UCT no solo se limita a recibir ordenes remotas de un usuario a través de la línea telefónica, sino que también puede responder e informar con mensajes de **audio (voz)**, lo que permite que el usuario pueda seguir fácilmente las acciones que se están llevando a cabo en el UCT e interactuar sencillamente.

Finalmente y quizás unas de las características mas importantes del UCT, es su **puerto de comunicaciones** denominado **BUS DCE ©**, el cual permite una comunicación bidireccional entre el UCT y otros dispositivos externos a través de señales digitales. Esta ultima utilidad permite un sin fin de aplicaciones (limitada solo por la imaginación del usuario), y posibilita que el UCT sea expandido a través de módulos (circuitos electrónicos externos al UCT), añadiendo funcionalidades extras al aparato.

La posibilidad de expandir mediante módulos al UCT puede lograr aplicaciones complejas, como por ejemplo operar computadoras remotamente a través del teléfono o informar de la activación de una alarma a un numero telefónico predeterminado por el usuario.

2.0: Desarrollo del proyecto:

2.1: Vista superior de los componentes electrónicos del circuito

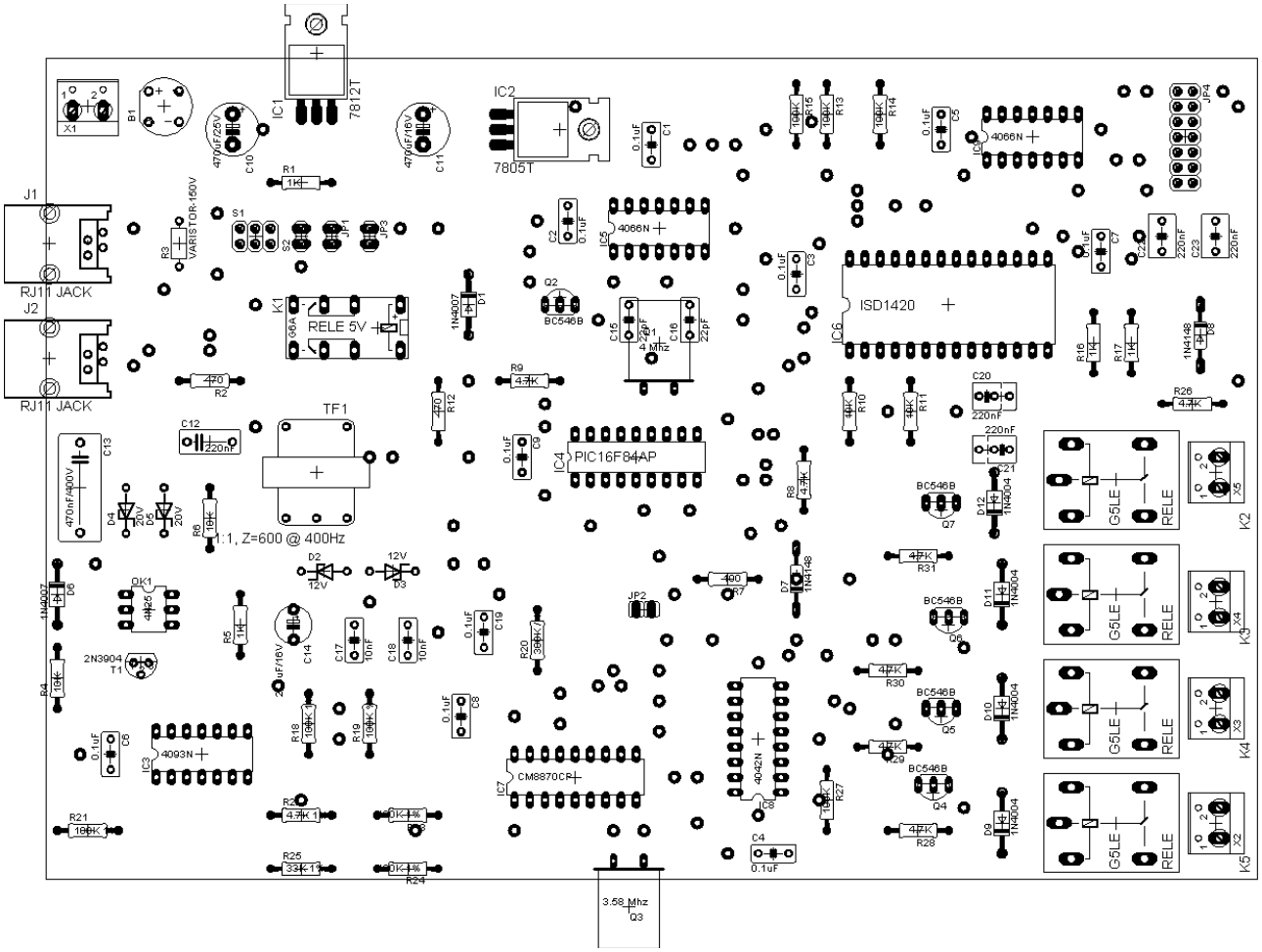


Fig. 2.1-a

La Fig. 2.1-a muestra la disposición de componentes de la placa electrónica del UCT, en la esquina superior izquierda están los conectores RJ11 que se conectan a la línea de teléfono. En el centro se ubica el microcontrolador PIC16F84A que es el “cerebro” del sistema. El integrado mas Grande es la ISD1420 que almacena los mensajes de audio. En el lado derecho de la placa se manejan los dispositivos externos de potencia y el puerto de comunicaciones BUS DCE.
Mas adelante se explicara detalladamente cada parte del circuito.

2.2: Guía de Uso del UCT

Antes de comenzar con las descripciones técnicas del proyecto, explicaremos su funcionamiento general y modo de uso, así será mas fácil de entender y explicar luego, el funcionamiento interno y desarrollo circuital del UCT.

2.2.0: Características generales del UCT

OBJETIVO GENERAL: Comandar dispositivos remotamente a través de la línea telefónica utilizando tonos DTMF generados por un aparato telefónico.

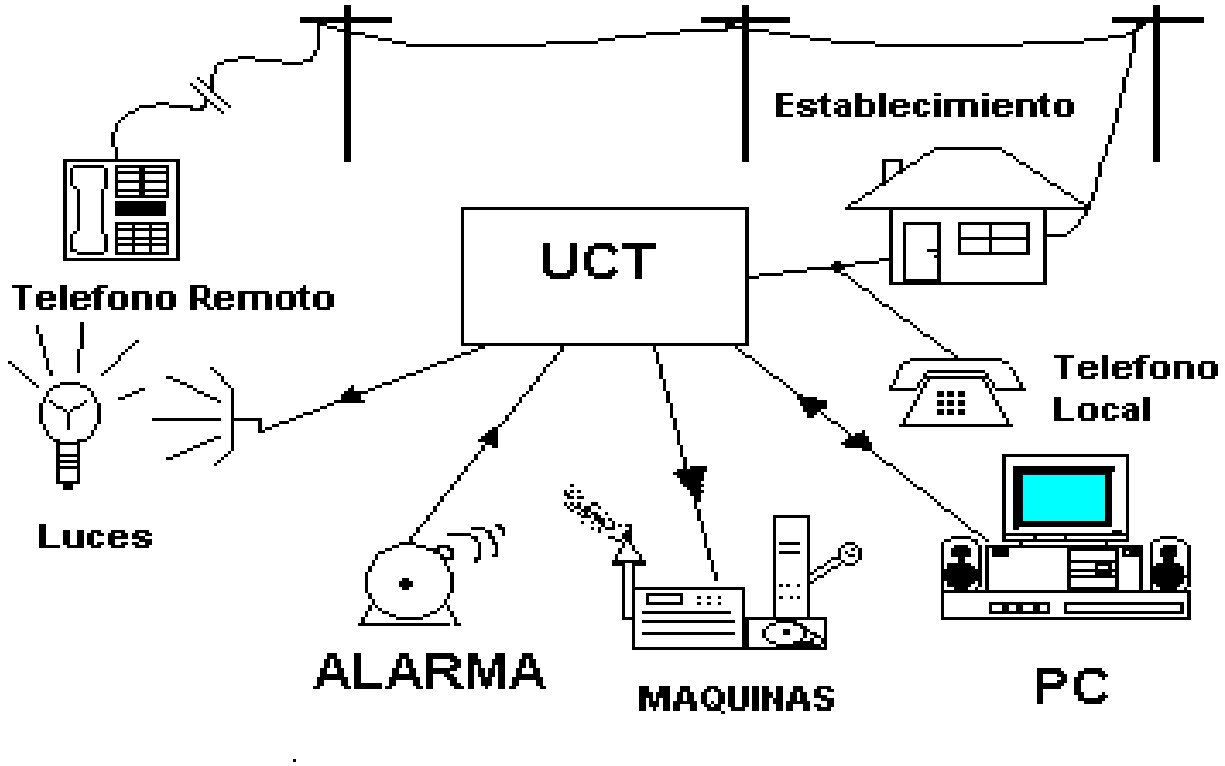


Fig. 2.2.0-a

EL UCT PUEDE:

El UCT puede:

- Activar / Desactivar 4 dispositivos **ON/OFF** de una potencia máxima de **1800 W** cada uno.
- Enviar y recibir datos de dispositivos digitales conectados al Bus DCE (puerto de comunicaciones).
- Enviar y recibir audio de dispositivos conectados al Bus DCE.
- Incrementar sus capacidades electrónicas de forma dinámica mediante módulos electrónicos externos conectados al Bus DCE.

- Interactuar e informar, a través de menús y mensajes, de audio (voz) con el usuario remoto.
- Restringir el acceso remoto al aparato a través de una clave.
- Esperar un numero X de Rings antes de contestar un llamado.
- Permanecer en modo stand-by (bajo consumo) cuando no es utilizado.
- Permite la simulación completa de una llamada real, utilizando simplemente el teléfono local y sin ocupar la línea telefónica (control local). Útil para configurar al UCT y simular efectos de forma local.

2.2.1: Requerimientos y configuración del UCT:

2.2.1.2: El UCT requiere para su funcionamiento optimo:

- Una línea telefónica fija.
- Un teléfono con discado por tonos (DTMF).
- Alimentación eléctrica de 12 Vca.
- La línea telefónica y el teléfono deben ser conectados al UCT utilizando fichas RJ11.
- Los dispositivos de potencia son conectados utilizando borneras, las cuales tienen un tornillo para poder amarrar un cable y así usted puede conectar los dispositivos de potencia al UCT, que luego pueden ser activados o desactivados. El funcionamiento, es exacto al de cualquier interruptor.
- Los requerimientos para el BUS DCE serán explicados mas adelante.

2.2.1.2: Para configurar el UCT los pasos son los siguientes:

1) Conectarlo al suministro eléctrico.
2) Conectar la línea telefónica en el conector J1.
3) Conectar el teléfono en el conector J2.
4) Encender el UCT.
5) Poner el interruptor S2 en **modo control local**.
6) Descolgar el teléfono.
7) Presionar varias veces el interruptor S1 hasta que usted escuche en el auricular del teléfono: "**Ingresar Clave**"
8) Digite la secuencia: "**1234**" en su teléfono para poder entrar al UCT. Luego podrá cambiar esta clave. Solo tiene 3 oportunidades para ingresar correctamente la clave. De lo contrario el UCT se pone en stand-by, y debe repetir el paso 7).
9) El UCT dará la bienvenida y reproducirá con audio el menú principal. Luego de que reproduzca el menú, puede elegir cualquier sub-menú u opción y ver los efectos de igual forma a como sucederían si usted estaría llamando remotamente al UCT. Por ahora simplemente presione "#" (numeral) en su teléfono para salir del UCT, y lea la sección **2.2.2** para conocer mas sobre los sub-menús del UCT.

Nota: Mientras el UCT este en **modo control local**, no podrá recibir llamadas telefónicas sobre el teléfono conectado al UCT, ni en el UCT.

2.2.2: Menús del UCT

El UCT informa los menús disponibles al usuario a través de mensajes de audio almacenados en su memoria. Los menús por lo general informan de las opciones disponibles a ejecutar en el UCT.

2.2.2.0: Reglas de los menús:

A menos que se especifique lo contrario:

- La tecla * (ASTERISCO) sirve para repetir los mensajes del menú actual.
- La tecla # (NUMERAL) sirve para salir del menú actual y volver al menú anterior. En caso de estar en el menú **principal**, desconecta al UCT de la llamada actual.
- Si no se presiona ninguna tecla del teléfono luego de que un mensaje fue reproducido o cuando el UCT espera un dato, durante el lapso de **1 minuto** el UCT se desconecta de la llamada actual.

2.2.2.1: Menú - Principal

Este menú es el primero en reproducirse al entrar con una clave correcta al UCT, el mismo informa sobre todas las funcionalidades disponibles en el UCT y permite acceder a otros sub-menús mas específicos.

Los siguientes mensajes serán reproducidos en el Menú principal:

“UNO, Control de Dispositivo”
“DOS, Ver estado de Dispositivo”
“TRES, Bus DCE”
“CUATRO, Configurar RINGS”
“CINCO, Cambiar Clave”
“NUMERAL, Salir”
“ASTERISCO, Repetir”

Donde las primero 5 opciones, son sub-menues.
La opción NUMERAL , desconecta al UCT y lo pone en stand-by a la espera de una nueva llamada.
La opción ASTERISCO, repite el menú principal.
Las demás opciones se explican por si solas, por Ej. si usted presiona la tecla DOS de su teléfono, podrá ver el estado de los dispositivos.

Hay una opción que no se reproduce en este menú, pero que es útil y puede ser usada, ella es la opción **0** (CERO)**.** Al digitar este numero habilitara o deshabilitara la reproducción de audio en el UCT, según sea su estado actual. Ello es útil para usuarios experimentados que no deseen perder el tiempo en escuchar mensajes que ya conocen.

Los siguientes párrafos explican cada submenú.

2.2.2.2: Menú – Control de Dispositivos:

El siguiente mensaje será reproducido al ingresar en el Menú - Control de Dispositivos:

“INGRESAR DISPOSITIVO”

Ahora usted debe elegir un dispositivo, para ello ingrese un numero entre el **1 y el 4**, el cual representa el numero de dispositivo a activar o desactivar.

A continuación ingrese la acción a realizar:

0 -> **para desactivar dispositivo seleccionado.**
1 -> **para activar dispositivo seleccionado.**

Luego, si todo salió bien el UCT responderá:

“DISPOSITIVO ACTIVADO” (Si usted eligió activar el dispositivo)

O

“DISPOSITIVO DESACTIVADO” (Si usted eligió desactivar el dispositivo)

Posteriormente el UCT volverá a repetir este menú para que usted pueda desactivar / activar otro dispositivo nuevamente, o simplemente use la tecla # (NUMERAL) para volver al menú principal.

Este tipo de dispositivos que solo pueden ser activado o desactivados, se llaman DISPOSITIVOS **ON/OFF** en terminología del UCT.

2.2.2.3: Menú – Ver Estado de Dispositivos:

El Menú – Ver Estado de Dispositivos, solo informa el estado actual de los dispositivos ON/OFF conectados al UCT. Una vez que ha informado el estado de los dispositivos, vuelve al menú principal sin necesidad de presionar ninguna tecla telefónica.
Por ejemplo si Usted tiene los dispositivos 1 y 3 activados, el UCT reproducirá los siguientes mensajes:

“DISPOSITIVO:”
“UNO ACTIVADO”
“DOS DESACTIVADO”
“TRES ACTIVADO”
“CUATRO DESACTIVADO”

Luego volverá al menú – principal.

2.2.2.4: Menu – Bus DCE:

EL Menú – Bus DCE es un puerto de comunicación bidireccional entre el UCT y dispositivos externos que requieren comandos mas complejos que un simple ON/OFF.
Estos tipos de dispositivos conectados al UCT se denominan **DCE** (Dispositivo de Control Externo) en la terminología del UCT.
Gracias a este puerto el UCT puede incrementar su funcionalidades o potencial, dependiendo del tipo de DCE que este conectado al UCT.

Al entrar en este menú el UCT se intentara conectar con el DCE por medio se su Bus DCE. En el caso de que la conexión se exitosa, el DCE tomara control **TOTAL** del funcionamiento del UCT y guiara al usuario por nuevos menús que serán definidos por el fabricante del DCE.

Si no hay un DCE conectado al Bus DCE, el UCT espera por 30 segundos a que el DCE intente conectarse, para luego si no hubo conexión, el UCT informara con el siguiente mensaje:

“ERROR, CONECTAR”

y volverá al menú principal. También este mensaje de reproducirá en caso de algún error en la conexión con el DCE.

Notemos, que como el DCE toma control total del UCT, el mismo puede producir fallos indeseados en UCT, en caso de que tome acciones perjudiciales para el funcionamiento del UCT. Si usted desea construir un DCE, lea la sección 3.6.1 en adelante.

2.2.2.5: Menu – Configurar RINGS:

El Menú – Configurar RINGS, permite establecer el numero de RINGS que debe esperar el UCT antes de contestar un llamado.

Al seleccionar este Menú el UCT reproducirá:

“INGRESAR RINGS”

Ahora debe ingresar un numero comprendido entre el **1 y el 9**, lo cual especificara la cantidad de RINGS que el UCT debe esperar antes de contestar un llamado.
Si usted quiere establecer un numero mayor a 9 RINGS en el UCT, puede ingresar el numero **0**, que establece 18 RINGS a esperar. Esto es útil por si se quiere tener activo el UCT, pero que conteste solo cuando haya demasiados RINGS.
Note que si usted contesta primero el teléfono, el UCT permanecerá inactivo y no interferirá con la llamada actual.

Una vez ingresado el numero de RINGS el UCT responderá:

“LISTO”

Y retornara al menú principal.

Nota: Por fabrica el UCT esta configurado para responder a los 4 RINGS.

2.2.2.6: Menú – Cambiar Clave:

El Menú – Cambiar Clave, permite establecer una clave de **4** dígitos en el UCT, que será pedida cada vez que el UCT atienda el teléfono.
Si es ingresada incorrectamente 3 veces, el UCT se desconectara del actual llamado y deberá ser llamado de vuelta para poder ingresar nuevamente la clave.

Cuando se selecciona este menú, el sig. Mensaje es reproducido:

“UNO, CAMBIAR CLAVE”
“NUMERAL, SALIR”
“ASTERISCO, REPETIR”

Usted debe presionar la tecla **1** para confirmar que desea cambiar la clave o # (NUMERAL) para volver al menú principal.

Una vez presionado **1** usted debe ingresar la nueva clave, para ello digite una secuencia de 4 dígitos, la cual será su nueva clave.
Los dígitos permitidos para la nueva clave son:

0,1,2,3,4,5,6,7,8,9,#,*

Notar que en este punto no puede presionar # (NUMERAL) para salir, ya que será interpretado como parte de la nueva clave.

Luego de ingresar la nueva clave, el UCT reproducirá:

“LISTO”

Y retornara al menú principal.

Nota: La clave configurada por fabrica es “**1234”**, usted debe cambiarla por alguna otra de inmediato!.

Nota 2: Un intruso deberá realizar 6912 llamados telefónicos para poder realizar todas las combinaciones posibles de la clave y entrar al UCT.

2.2.3: Configuración Inicial del UCT:

Ahora estableceremos una clave nueva y el numero de RINGS en el UCT.
Primero lea la sección **2.2.1.2**, y siga los pasos descriptos hasta llegar al numero **9)**, para mayor seguridad lea también la sección **2.2.2**.

Luego de que el menú principal sea reproducido digite:

5 (espere que se reproduzcan los mensajes) y luego digite **1.**

Ahora ingrese su nueva clave de **4** dígitos.

Espere a que se reproduzca el menú principal nuevamente, y digite:

4 (espere a que se reproduzcan los mensajes) y luego ingrese el numero de RINGS que el UCT deberá esperar antes de contestar el llamado.

Espere a que se reproduzca el menú principal nuevamente, y digite:

(numeral) para salir del UCT.

Ahora cambie de posición el interruptor **S2** a modo **control remoto,** para que el UCT pueda responder a un llamado telefónico.
Cuando un llamado telefónico se reciba el UCT esperara los RING establecidos por usted y luego le pedirá su clave personal para poder ingresar a comandar dispositivos.

2.2.4: Operación remota del UCT:

Para habilitar el UCT para que pueda ser operado a través de un llamado telefónico, debe poner el interruptor **S2** en modo **control remoto.**
Luego llame al UCT, ingrese su clave y luego navegue por los menú, para activar dispositivos, cambiar claves, etc... lea la sección **2.2.2** para mas información sobre los menús.

2.2.5: Errores:

Cuando ingrese números o datos incorrectos el UCT le informara por medio de mensajes de audio del error, y luego volverá a pedirle que ingrese nuevamente el dato.

2.2.6: Preguntas Frecuentes (FAQ):

P: Puedo interrumpir al UCT mientras esta reproduciendo mensajes de audio?
R: SI, esto es posible desde la versión 0.6.1 de software del UCT.

P: Como es la interfaz del BUS DCE?
R: Lea la sección técnica. (3.6.1 en adelante).

P: Un DCE conectado al Bus DCE , puede producir errores en el UCT?
R: SI, ya que el mismo toma control TOTAL del UCT. Lea sección 3.6.1 en adelante.

P: Puedo simular una llamada localmente y controlar al UCT?
R: Si lea sección 2.2.1.2.

P: Conozco perfectamente los menús del UCT y sus mensajes, hay alguna forma de evitar que se reproduzcan sus mensajes de audio?
R: Si, hay dos formas:

1) Cuando ingresa la clave correctamente, tiene **2 segundos** para digitar el numero **0 (cero)**, el cual evitara la reproducción de cualquier menú.
2) Cuando se esta en el Menú principal, digite **0 (cero)**, ello desactivara o activara los mensajes de audio, según sea su estado actual.

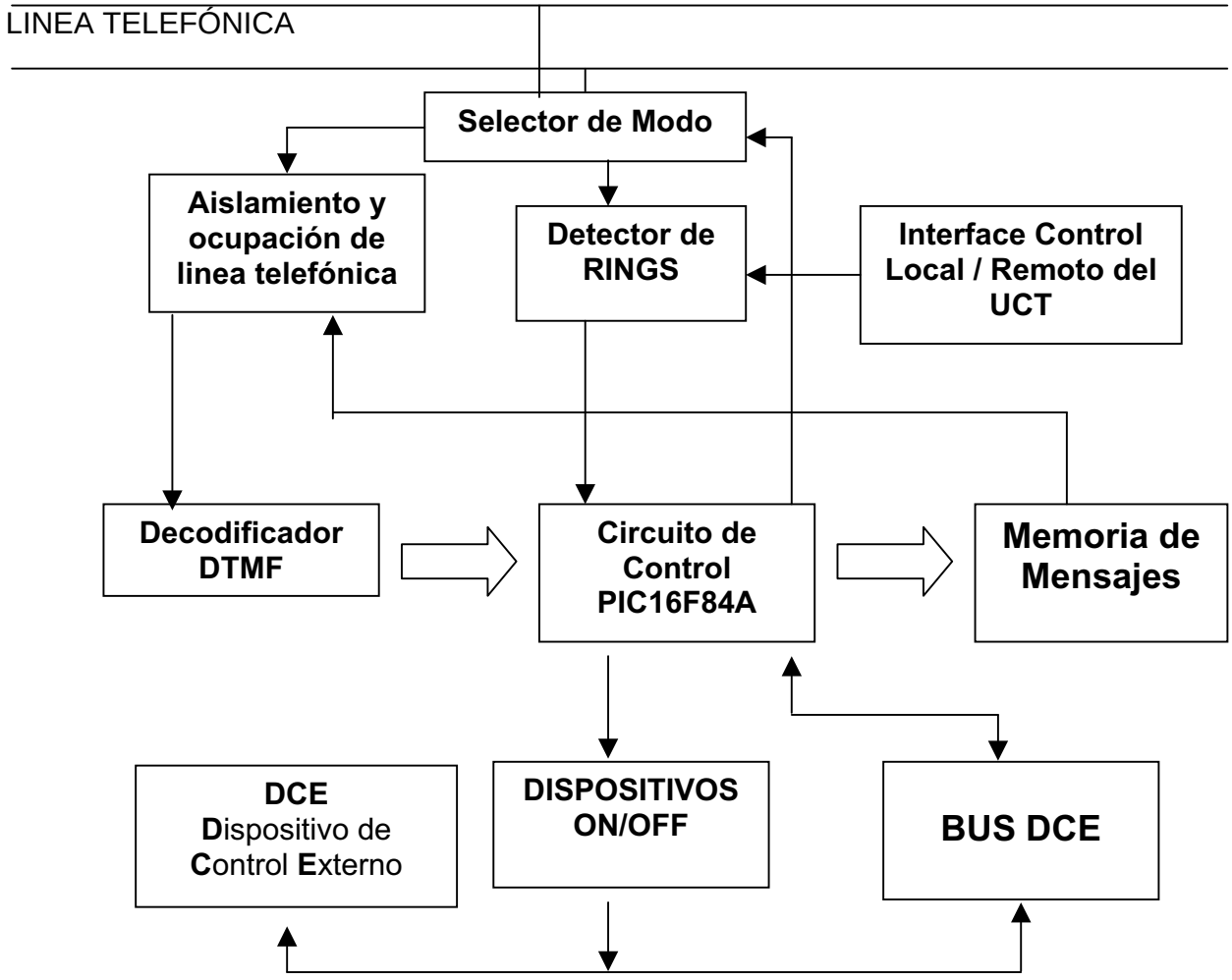
Los mensajes de confirmación, continuaran reproduciéndose. Por ejemplo:
"DISPOSITIVO ACTIVADO", se reproducirá al activar un dispositivo, pero el menú principal y el menú de control de dispositivos, no serán reproducidos.

P: Estoy usando la versión de software 0.6.2 o inferior del UCT, que debo tener en cuenta al actualizarla a la versión 0.8.1 o superior ?.
R: El funcionamiento general se mantuvo, pero el funcionamiento del Bus DCE se cambio y actualizo. Ahora se utiliza un protocolo para conectarse al DCE. Si usted no utiliza esta función, quizás no necesite actualizar a una nueva versión de software.

P: Al activarse por primera vez el UCT los reles (K2...K5) de los dispositivos ON/OFF conmutan de forma errónea. Que puedo hacer?.
R: Actualizar la versión de software a la 0.6.2.

3.0: Desarrollo Electrónico del UCT

3.1: Funcionamiento por Bloques:



Básicamente el circuito espera en stand-by un llamado telefónico a través de la línea telefónica cuando es puesto en control modo remoto a través de la **interface de control Local / Remoto del UCT**, cuando el llamado ocurre, el **detector de RINGS** le informa al **circuito de control (pic16f84a)** la presencia de un RING. Luego **circuito de control (pic16f84a)**, decide si la cantidad de RINGS es suficiente y contesta el llamado ocupando la línea telefónica a través del **selector de modo** que activa el bloque **aislamiento y ocupación de línea telefónica**.
Posteriormente el **circuito de control** reproduce los mensajes correspondientes de la **memoria de mensajes**, y el usuario ingresa, claves, datos, comandos etc a través de tonos DTMF que son decodificados por el **decodificador DTMF** y enviados al **circuito de control** para ser procesados. Luego el usuario activa o desactiva **dispositivos ON/OFF** o se comunica con **dispositivos de control externos** conectados al **bus dce.**

Lo mencionado anteriormente puede haber sido simulado de forma local poniendo al UCT en modo control local a través del bloque **interface de control Local / Remoto del UCT.**

Otra posibilidad, es que un dispositivo de control externo (DCE) se conecte e interrumpa al UCT (despertándolo) tomando control del mismo, esto es: le ordene al UCT que ocupe la línea telefónica, reproduzca un mensaje de audio, espere un código DTFM o realice un discado telefónico a través de pulsos (el UCT no puede llamar por medio de tono, aunque el DCE puede implementarlo).

3.1: Descripción Breve del funcionamiento de cada bloque:

Selector de Modo:
Cuando el UCT ha recibido una llamada, el **selector de modo** conecta la línea telefónica al **circuito de aislamiento eléctrico** (modo respuesta)**,** por donde fluirá la información al **decodificador DTMF.**
El **Selector de Modo** es controlado por el **circuito de control.**

Detector de RINGS
Detecta un llamado telefónico al UCT y activa el **circuito de control.** Permite también que el **circuito de control** cuente la cantidad de RINGs para contestar la llamada en un numero predeterminado de Rings.

Aislamiento y ocupación de línea telefónica
Circuito encargado de aislar la línea telefónica eléctricamente del UCT, también cumple la función de ocupar la línea telefónica en caso de que el UCT decida contestar una llamada telefónica. Por este circuito fluirá la información de los códigos DTMF provenientes del exterior y los mensajes de respuesta (en forma de audio) de la **memoria de mensajes**.

Decodificador DTMF
Permite decodificar los códigos DTMF a números binarios, que serán pasados en forma digital al **circuito de control** para ser procesados y consecuentemente llevar a cabo una acción.

Memoria de Mensajes
Contiene mensajes en forma de Voz (señales de audio) que permitirán indicar al usuario que llama remotamente las acciones que se están llevando a cabo en el UCT.
Los mensajes son del tipo: “Dispositivo Activado”, “Ingresar Clave “, “Dispositivo desactivado”, “Configurar RINGS”, “Bienvenido” , etc

Interface de control local / remoto del UCT
Permite elegir el modo de operación del UCT, los modos disponibles son:
Modo control Local y Modo control Remoto.
El primero permite el control local sin ocupar la línea telefónica, y el segundo permite dejar al UCT para recibir llamadas y ser controlado remotamente.

El **Modo control Local** permite configurar localmente al UCT, es decir establecer clave de acceso, numero de RINGs, activar dispositivos localmente, etc... sin la necesidad de ocupar la línea telefónica o llamar desde un teléfono remoto.
Este circuito simulara una llamada externa, de esta forma el usuario podrá cambiar la configuración usando solo el teléfono local y escuchando los cambios por el auricular, lo cual evita la necesidad de implementar controles externos en el UCT y abaratar costos.

Circuito De Control (PIC16F84A)
Este es básicamente el cerebro del UCT, permite procesar los códigos provenientes del exterior y llevar al cavo las acciones, tales como reproducir mensajes, activar dispositivos, pedir contraseña, ver el estado de los dispositivos, controlar al resto de los circuitos del UCT, etc.
Como esta construido con un microcontrolador, necesita un software para funcionar, el cual comanda el resto de los circuitos y procesa los datos proveniente de ellos.

DISPOSITIVOS ON/OFF
Etapa de manejo de potencia, permite al **circuito de control** activar / desactivar dispositivos de gran consumo eléctrico o simplemente de pequeño consumo. Brindando un aislamiento adecuado con el resto del UCT.

BUS DCE
Básicamente permite una expansión bidireccional inteligente del UCT, aquí se podrá conectar un dispositivo complejo que necesite una serie de comandos para ser activado, aquellos dispositivos que necesiten mas información que un simple ON/OFF. Por ejemplo podremos conectar una Alarma que requiera una clave para ser activada, o algún dispositivo que permita el uso de comandos para llevar acciones que el UCT no puede realizar, o quizás un sensor que reporte algún evento, etc.

DCE (Dispositivo de Control Externo)
Este bloque no forma parte del UCT, es solo un modulo electrónico que puede ser conectado al BUS DCE opcionalmente para expandir las funcionalidades del UCT o simplemente para recibir ordenes del UCT.

3.2: Términos:

- Modo de espera: El UCT se encuentra en STAND-BY consumiendo poca energía. Por lo general cuando no hay una llamada en progreso.
- Modo respuesta: El UCT recibió una llamada y se encuentra interactuando con el usuario remoto o fue activado por el bus DCE.
- Modo Control Local: El UCT funciona normalmente, pero no por acción de un usuario remoto, sino que fue activado por un usuario local que desea simular una llamada con su teléfono local. En este modo el UCT no puede recibir llamadas externas.

3.3: Esquema circuital de los bloques

Debido a que muchos bloques están estrechamente relacionados, pero no por ello dejan de ser independientes, dividiremos al UCT en 3 Circuitos principales.

1: Circuito de interfaz de línea y Alimentación.

El cual contiene los módulos:

- **Selector de Modo**
- **Detector de RINGS**
- **Aislamiento y Ocupación de línea telefónica**
- **Interface de configuración local.**

2: Circuito Principal

El cual contiene los módulos:

- **Circuito de Control (PIC16F84A)**
- **Decodificador DTMF**
- **Memoria de mensajes**

3: Circuito de Potencia y BUS DCE

El cual contiene los módulos:

- **Dispositivos ON/OFF**
- **BUS DCE**

3.4: Circuito de interfaz de línea y Alimentación

La sección de alimentación de este circuito contiene una entrada de **12 Vca**.
La cual es rectificada, para luego proveer +5V y +12V de CC a través de los reguladores de tensión LM7805 y LM7812, que serán necesarios para alimentar el resto de los circuitos.
El circuito completo del UCT consume aproximadamente como corriente mínima 20 mA (en modo stand-by) y una máxima de 250 mA (cuando se controla localmente y todos los reles de los dispositivos ON/OFF están activados).
El circuito de alimentación fue diseñado para poder suministrar 1 A como corriente máxima, y así poder alimentar con suficiente corriente a los circuitos externos conectados al **BUS DCE**.

La línea telefónica se conecta al conector J1, donde se ubica un varistor de 150 V, (R3) que preteje al resto de los circuitos de tensiones superiores a 150 V que pudieran presentarse en la línea telefónica.

Luego la llave S1 permite **seleccionar el modo** de operación del UCT, los modos pueden ser: **Control remoto** o **Control local.**
Cuando se elige **modo Control Remoto** la línea telefónica es derivada al **circuito detector de RINGS** a través del rele K5 y al conector J2 donde se coloca el aparato telefónico (el cual podrá ser operado normalmente en este modo, se podrán hacer llamadas, recibir llamadas, etc.).
El **circuito detector de RINGS** esta compuesto por:
C13: Capacitor donde se elimina CC. También ofrece relativa baja impedancia a la señal de RING de 25 Hz y soporta tensiones elevadas (470nF/400V).
D4,D5: Diodos Zener en back-to-back que filtran picos de ruidos menores a 20 V presentes en la línea TE.
D6: Protege diodo del opto acoplador de tensiones inversas superiores a 2 V.
R4: Requerido para presentar una impedancia de 10Kohms a señales de audio y limitar la corriente de RING de 10mA A 20 mA.
OK1: Opto acoplador 4N25 que detecta ciclo positivo de la señal de RING. Cuando un RING es detectado se presenta un 1 lógico en el Emisor de T1 (el cual esta en configuración Darligton con el transistor del opto acoplador, necesario ya que la corriente generada por un RING es muy pequeña para saturar completamente el transistor del Opto acoplador) .
C14: mantiene el 1 lógico del RING de forma continua, ya que la señal de RING tiene una frecuencia de 25 Hz y eso produciría un estado lógico “1” pulsante de 25 Hz a la salida del emisor de T1.
IC3A: Inversor con histéresis (trigger Schmith) que produce la señal lógica /RING (0-> RING PRESENTE, 1-> RING AUSENTE).
Por ultimo el detector de RING fue calculado para funcionar con una señal de RING de 135 Vpico-pico de 25 Hz montada sobre una tensión de 35 a 50 Vcc.

Al elegir el **modo Control local** con la llave S1, la entrada de línea es desconectada del circuito y se alimenta al conector J2 con 12 Vcc, esto permite alimentar el aparato telefónico sin necesidad de usar la línea telefónica. Permitiendo de esta manera operar al teléfono normalmente, pero sin la posibilidad de ocupar la línea telefónica.

El **circuito detector de ring** en su entrada no se ve afectado ya que los 12Vcc no generara circulación de CC gracias a C13.
Para permitir un circuito consistente y transparente desde el punto de vista de su funcionamiento, se emplea el interruptor S2 que permite polarizar la base del transistor del opto acoplador OK1 de forma manual, de esta manera generamos o simulamos RINGS de forma local a través de S1.
Debido a que S1 no presenta un circuito antirebote, se empleo un capacitor C14 de 220 uF, que junto a R5 de 1Kohms, establecen una constante de tiempo suficientemente grande para eliminar rebotes de la llave S1 y así mismo generar un 1 lógico estable al producirse un RING en el modo **control remoto.**
Cabe destacar que C14 no es un perfecto circuito antirebote, pero es barato y sastiface las necesidades del aparato.

Cuando el UCT detecta los RINGS necesarios para ocupar la línea telefónica activa al RELE K1 a través del PIC16F84, de modo que se conecta la línea telefónica con la entrada de audio telefónico del UCT.
De otra forma, T_LINEA-1 y T_LINEA-2 son conectados al transformador TF1.

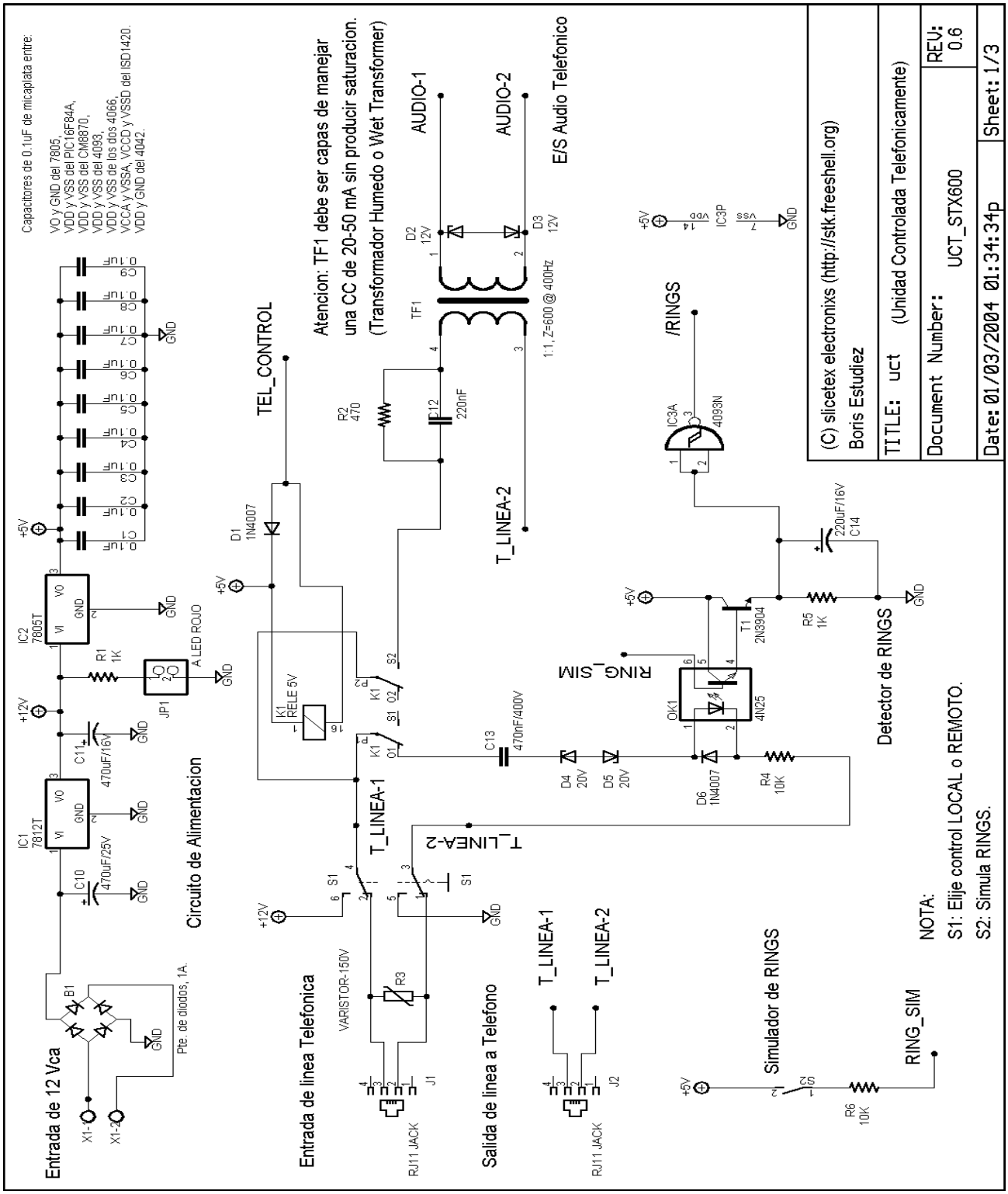
En este punto se ocupa la línea telefónica de la siguiente manera:
R2: Toma suficiente CC de la línea telefónica como para ocupar la línea telefónica (toma de unos 15 a 25 mA dependiendo de la tensión disponible en la línea telefónica). Este valor fue elegido tomando como promedio la impedancia de los aparatos telefónicos, que es del orden de los 300 a 500 Ohms.
C12: Evita la perdida innecesaria de potencia para las señales de audio superiores a 300 Hz.
TF1: es un transformador que presenta una impedancia de 600 Ohms a 400 Hz y relación 1:1, ello sastiface ciertas normas telefónicas que especifican aislamiento eléctrico del aparato conectado a la línea telefónica y adaptación de impedancias para señal de 600 Ohms a 400 Hz.
TF1 debe ser un transformador Húmedo, ello significa que permita el paso de CC a través de su bobinado, y no produzca saturación o distorsión cuando la señal alterna también pase por su bobinado. Este tipo de transformadores son ligeramente mas caros, pero simplifican el resto del circuito asociado que ocupa la línea telefónica. Finalmente a la salida de TF1 se encuentran dos diodos Zener (D2 y D3) de 12V en configuración back-to-back, que permite que el audio que sale de TF1 este limpio de picos de tensiones superiores a 12V (circuito limitador), necesario para proteger al resto de circuitos conectados a las líneas AUDIO-1 y AUDIO-2, que permiten la Entrada / salida de audio telefónico.

Algunas Notas:

- Cuando se elige **modo control local** se escucha un zumbido en el parlante del aparato telefónico. Ello se debe a que circula demasiada CC en TF1 y se produce saturación. La solución simple consiste en poner una resistencia de 220 Ohms en serie con el aparato telefónico. Ello puede ser logrado ubicándola en serie con fuente de 12Vcc y la pata 6 del interruptor S2. Si 220 ohms no es la solución busque otro rango de resistencias entre los 100 y 570 Ohms. Recuerde que a mayor resistencia, menor será la tensión disponible para alimentar su teléfono en **modo control local.**

- La forma de ocupar la línea telefónica con R2 en serie con TF1 no es una solución elegante, pero si económica y simple. En futuras versiones el UCT mejorara esta parte del circuito.
- En el esquemático del circuito, el puente de diodos B1, figura conectado al revés. Debido a un error de programa con el cual los cree no puedo editar mas el esquemático, por lo tanto si usted piensa copiar el circuito, tenga en cuenta aquel detalle. Si solamente quiere implementar el PCB (circuito impreso) no debe realizar ninguna corrección, solamente cuando suelde el puente de diodos, asegúrese que este en la posición correcta.

El esquema del **Circuito de interfaz de línea y Alimentación** se muestra a continuación:



3.5: Circuito Principal

Este circuito es el principal del UCT y su funcionamiento depende altamente de la parte de software que controla al microcontrolador PIC16F84A (IC4).
Igualmente trataremos de describir el funcionamiento por hardware:

El **IC7 es un CM8870**, el cual es un decodificador DMTF, que traduce las señales DTMF a códigos binarios.
Su funcionamiento se explica mejor en su hoja de datos, aquí solo diremos que:
AUDIO-1 y AUDIO-2 entran en forma diferencial al IC7, ello mejora el comportamiento frente a ruidos. Los capacitores y resistencias asociadas son del 1 % como se especifica en la hoja de datos.
Sus salidas digitales son:
STD: En nivel alto, informa que hay un DTMF presente en la entrada de audio y que ya esta decodificado en binario en Q1, Q2, Q3 y Q4. Esta salida esta conectado a RA4 del IC4.
Q1, Q2, Q3 y Q4: Salidas que representan el código binario del DTMF decodificado. Están conectadas al bus de datos RB[4..7] del IC4.
Su entrada digital es:
TOE: en nivel bajo, mantiene en alta impedancia a Q1, Q2, Q3 y Q4. Se conecta a RA0 del IC4.

El integrado **IC5 es un 4066**, el cual funciona como 4 llaves analógicas, permiten separar con alta impedancia el bus de datos RB[4..7] de la memoria **ISD1420 (IC5)**.
Para habilitar las llaves de este integrado se debe poner en nivel alto la línea RA1 del IC4.

La memoria **ISD1420 (IC6)** es una memoria de voz, que almacena todos los mensajes de audio del UCT.
Dichos mensajes son seleccionados por el IC4 y reproducidos por la memoria ISD1420 (IC6) a través de sus salidas analógicas SP+ y SP- conectadas a AUDIO-1 y AUDIO-2. Los capacitores C20, C21 y resistores R16 y R17 son necesarios para reducir la amplitud de la señal de salida de audio de la memoria y para filtrar frecuencias bajas.
Notar que la salida de la memoria es en forma diferencial, esto mejora los ruidos producidos por la misma y permite mejor aprovechamiento de la potencia de salida (4 veces superior que en modo común).
El modo de operación de la memoria puede ser leído de su hoja de datos o interpretado de la función **play_isd()** del software interno del PIC16F84A.

PIC16F84A:

Finalmente explicaremos el conexionado y funcionamiento general del **IC4, PIC16F84A** en el **UCT.**
El PIC16F84A es un microcontrolador de propósito general y de reducido costo.
Posee 1 K Words de memoria de código de programa, 64 Bytes de memoria EEPROM y 68 bytes de memoria RAM y un CPU RISC.
Se utilizo un oscilador de cristal de cuarzo a 4 MHz, (Q1), para generar una velocidad de clock de 1 MHz, velocidad suficiente para el proyecto.

Debido a su escasa memoria de código de programa, el UCT debió ser programado de forma tal que se aprovechara al máximo la memoria disponible del PIC16F84A, ello ocasionó, que muchas características del originales del UCT, todavía no estén disponibles debido que deben pensarse para que puedan ocupar la menor cantidad de memoria.

DISTRIBUCIÓN DE PUERTOS:

PORTB:

- **RB[4...7]**: Entrada / salida, **Bus de datos** entre dispositivos y el PIC16F84A.
- **RB3**: Salida, envía señal de ocupación de línea telefónica.
- **RB2**: Salida, envía interrupción a un DCE conectado al BUS DCE. (/DCEINT)
- **RB1**: Entrada, Identifica interrupción generada por un DCE conectado al BUS DCE. (/UCTINT_PIN).
- **RB0:** Entrada, Recibe interrupciones externas. El programa interno del PIC, se encarga de analizar el origen de tal interrupción.

PORTA:

- **RA0**: Salida, Elige al 8870 (IC7) y conecta el bus de datos a sus salidas.
- **RA1**: Salida, Elige a al 4066 (IC5) que conecta a la ISD1420 (IC6) al bus de datos para controlarla.
- **RA2**: Salida, Elige el 4042 (IC8) y conecta bus de datos a su entrada.
- **RA3**: Salida, Elige al 4066 (IC9) y conecta al bus de datos al bus dce.
- **RA4**: Entrada, recibe pulso que indica que el IC7 recibió un DTMF.

Adicionalmente se dejo un jumper JP2 que permite resetear al IC4, para fines de depuración.

RB3 al estar en nivel alto satura el transistor Q2 y el rele K5 se activa ocupando la línea telefónica como se menciono en el apartado **3.4.** La misma señal activa un LED verde conectado a JP3 que indica que hay una conexión o que se esta ocupando la línea telefónica.

INTERRUPCIONES:

La compuerta IC3C cuando esta en CERO lógico interrumpe al IC4 a través de **RB0.** Para determinar la fuente de la interrupción, nos valemos de la siguiente lógica:
Si un dispositivo conectado al bus DCE genero una interrupción, el **/UCTINT_PIN** estará en un nivel bajo, por lo tanto **RB1** estará junto a **RB0** en nivel bajo.
Por lo cual deducimos que la interrupción provino del BUS DCE.
Si solo hay un RING presente el la línea telefónica, **RB1** estará a nivel alto y **RB0** en nivel bajo**.** Por lo cual deducimos que la interrupción es por un intento de llamada o RING.
Si ambos, el bus DCE y un RING están presente, la prioridad de interrupción la tiene el BUS DCE.
Ello se explica mejor en la siguiente tabla:

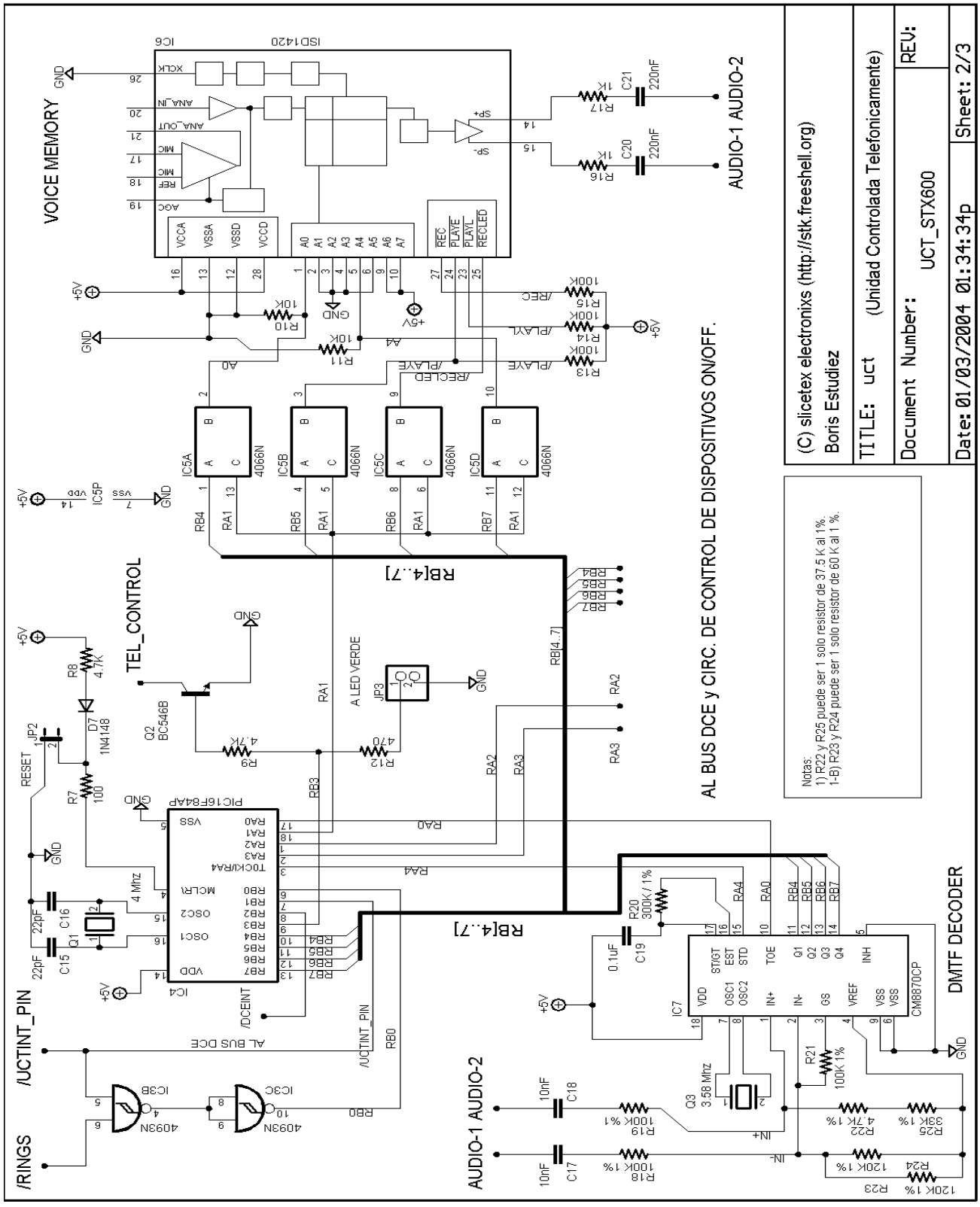
RB0	**RB1**	**Fuente de la Interrupción / Efecto**
1	1	No hay interrupción presente
0	1	Interrupción por **detector de RING**.
0	0	Interrupción por **BUS DCE**.
1	0	Nunca puede suceder!.

Finalmente, el comportamiento interno del PIC16F84A, es producido por el software que lo controla, para ello ver los archivos:

uct.asm y **uct.inc**

Que están disponible en la pagina del proyecto.

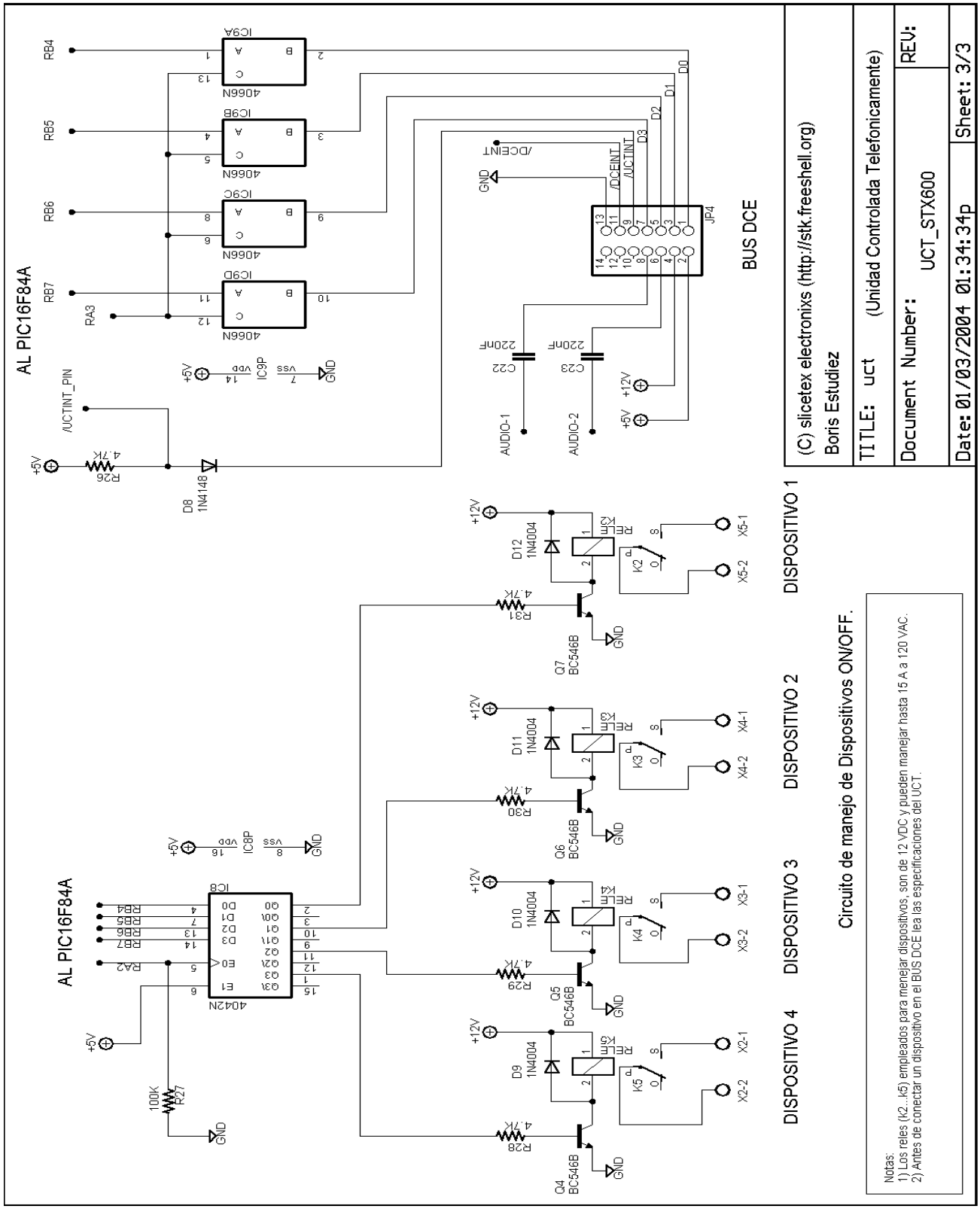
El esquema del **Circuito Principal** se muestra a continuación:



3.6: Circuito de Potencia y BUS DCE

El **circuito de potencia** consta de 4 Reles que son manejados por el **latch 4042 (IC8)**.
Cuando el Latch es habilitado a través de un pulso en E0, la salidas Q0...Q3 del latch reflejan los valores del bus de datos conectado a la entradas D0...D3.
Cuando el latch se inhabilita, con E0=0, el ultimo valor ingresado al latch es mantenido por sus salidas Q0...Q3.
De esta manera se polariza los transistores conectados a las salidas Q0...Q3, activando los reles K2...K5, según sea el ultimo valor ingresado al latch. De esta forma se mantiene el ultimo estado de los **dispositivos ON/OFF .**
Como el latch 4042 memoriza el estado de los dispositivos ON/OFF hasta que el suministro eléctrico deje de estar presente, el software de control del UCT debe actualizar el ultimo estado de los dispositivos ON/OFF al energizarse el UCT. Es por ello que se debe mantener una copia en la EEPROM del PIC16F84A del ultimo estado de los dispositivos ON/OFF.
Los reles se comportan como interruptores y pueden manejar corrientes de hasta 8A a 120Vac o 15ª a 120Vac según que tipo de rele de compre.

El **Circuito de Potencia y Bus DCE** muestra a continuación:



3.6.1 BUS DCE (PUERTO DE COMUNICACIONES DEL UCT):

El Bus DCE fue pensado para expandir dinámicamente a través de módulos electrónicos el hardware del UCT.
Es por ello que se lo diseño de tal forma que se pudiera establecer una comunicación bidireccional entre el UCT (Unidad Controlada Telefónicamente) y un DCE (Dispositivo de Control Externo).
Esta ingeniosa funcionalidad del UCT, permite la transferencia de datos digitales en ambos sentidos UCT < -- > DCE y viceversa.

HARDWARE DEL BUS DCE:

El bus DCE provee que las siguientes líneas sean accesible por un DCE:

- **D0/TX:** Salida de datos en forma serie. Transmite.
- **D1/RX:** Entrada de datos en forma serie. Recibe.
- **D2:** No utilizado actualmente. No debe usarse.
- **D3:** No utilizado actualmente. No debe usarse.
- **/DCEINT:** Permite interrumpir a un DCE conectado al BUS DCE.
- **/UCTINT:** Permite interrumpir al UCT desde un DCE conectado al Bus DCE.
- **AUDIO-1 y AUDIO-2:** Entrada / salida de audio telefónico en forma diferencial disponible para ser accedida por el DCE.
- **+5V:** Tensión de 5 Vcc disponible para alimentar un DCE.
- **+12V:** Tensión de 12 Vcc disponible para alimentar un DCE.
- **GND:** Tierra o masa de la fuente de alimentación del UCT.

Internamente, el integrado **4066 (IC9)** separa el bus de datos RB[4...7] con las líneas de datos del Bus DCE, D0...D3. El PIC16F84A, decide cuando habilitar el 4066 con RA3 y hacer accesible el bus de datos RB[4...7] en el BUS DCE.

Notemos que /DCEINT y /UCTINT, tienen lógica negada. Por lo tanto cuando se genera una interrupción, la misma se hace con nivel bajo (0V), de lo contrario están en nivel alto (5V).

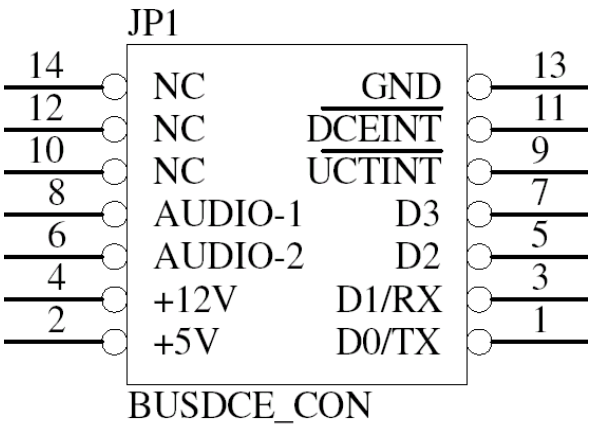


Fig. 3.6.1-A: Conector / Header 2X7 tipico del Bus DCE.

3.7: TEORIA DE COMUNICACION ENTRE EL UCT Y UN DCE

La comunicación entre el UCT y un DCE por medio del Bus DCE, se realiza con un pequeño protocolo llamado **CTP** (Protocolo de Transmisión de Comandos) el cual funciona sobre una comunicación del tipo serie, lo cual lo hace muy simple de implementar.
Cuando un DCE se conecta al UCT, el DCE toma control **TOTAL** del UCT. Esto es, el UCT pasara a ser controlado a partir de los comandos que envié el DCE.

Supongamos por ejemplo que el DCE le envía los siguientes comandos al UCT:
"Ocupar línea telefónica", "Llamar al numero telefónico 03543-42324", "Reproducir el mensaje ALARMA", "Esperar un Código DTMF".
Evidentemente, el UCT ya no se controla por si mismo, por lo tanto es el DCE quien debe enviar una secuencia de comandos "útiles" para que un usuario las interprete y pueda de esta forma controlar al UCT o al DCE, por medio de los tonos DTMF. Los comandos validos se explicaran mas adelante.

3.7.1: Transmisión de datos entre el UCT y DCE

La transmisión de datos (bytes) entre el UCT y el DCE, se realiza de forma serie. Este tipo de transmisión es muy simple y se usa el estándar NRZ (Non-Return-to-Zero), ello es un bit de start, 8 bits de datos y un bit de stop. Las computadoras (con sus puertos series) o muchos microcontroladores, traen implementado este tipo de comunicación. Aquí no la vamos a explicar, ya que puede ser encontrado su explicación con mas detalle en otros textos.

Los parámetros de la transmisión y recepción serie, se implemento con los siguientes parámetros:

Velocidad: 1200 bps
Bits de Datos: 8
Paridad: Ninguna
Bits de Stop: 1

En resumen: **1200 bps, 8N1.**

***Recordar**: Los pines utilizados del Bus DCE, son el D0/TX y el D1/RX. También, cada vez que el UCT quiere transferir o recibir un dato por el Bus DCE, pone el pin /DCEINT a nivel bajo (0 Volts).*

Nota: El PIC16F84A, no trae USART, por lo tanto el soporte del puerto serie fue implementado por software. Por ello la velocidad es baja, pero no tiene error.

3.7.2: Protocolo CTP (Command Transmission Protocol)

Este protocolo especialmente desarrollado para el UCT, es el encargado de "administrar" la conexión o los bytes que se transmiten entre el UCT y el DCE.
CTP, significa **Protocolo de Transmisión de Comandos**, por ello se diseño con la finalidad de transmitir comandos, sus resultados y por sobre todas las cosas sincronizar la transmisión entre el UCT y el DCE.

El protocolo tiene una trama de 3 bytes con el siguiente formato:

B0	B1	B2

Siempre deben enviarse, aunque no se usen todos los bytes. B0 = Primer Byte, B1 = Segundo byte y B2 = Tercer Byte.

CONEXION:
Cada vez que se quiera enviar un comando, las siguientes secuencias deben ser seguidas:

1) Para iniciar una conexión, el transmisor envía primero la trama **HELO**:

B0 = 0xAA	B1 = 0x55	B2

Donde B2 puede tener cualquier valor. Los valores de B0 y B1 están especificados en base 16 (hexadecimal) en este documento.

1.B) La trama HELO debe responderse con otra trama HELO por parte del receptor. Esto le informara al transmisor que esta listo para recibir otra trama, la cual se llama COMMAND.

2) Una vez que el transmisor verifica que su trama HELO fue contestada con otra trama HELO, se envía la trama **COMMAND** y representa un comando o función a ejecutar en el receptor.

B0 = CMD	B1 = ARG1	B2 = ARG2

Donde:
CMD: Es un numero que representa el comando o función a ejecutar en el receptor. CMD no puede ser el numero 0xAA.

ARG1 y ARG2: Son los argumentos del comando o función a ejecutar. Si no hay argumentos o no se usan todos, en el receptor se ignoran tales bytes.
3) Luego de ejecutado el comando, el receptor envía el resultado del comando ejecutado en la trama **FINISHED:**

B0 = CMD	B1 = R1	B2 = R2

CMD: Numero que representa el comando o función que se ejecuto en el receptor. Debe coincidir con el CMD de la trama COMMAND.

R1 y R2: Resultado del comando ejecutado, R1 o R2 puede contener algún valor de interés para el aparato que inicio la conexión. De lo contrario R1 y R2 pueden ser ignorados.

4) El receptor debe volver al punto 1) donde esperara otra conexión, al igual que el transmisor para enviar otro comando.

ERRORES:

Para evitar errores en la conexión o ejecución de parámetros, se aconseja:

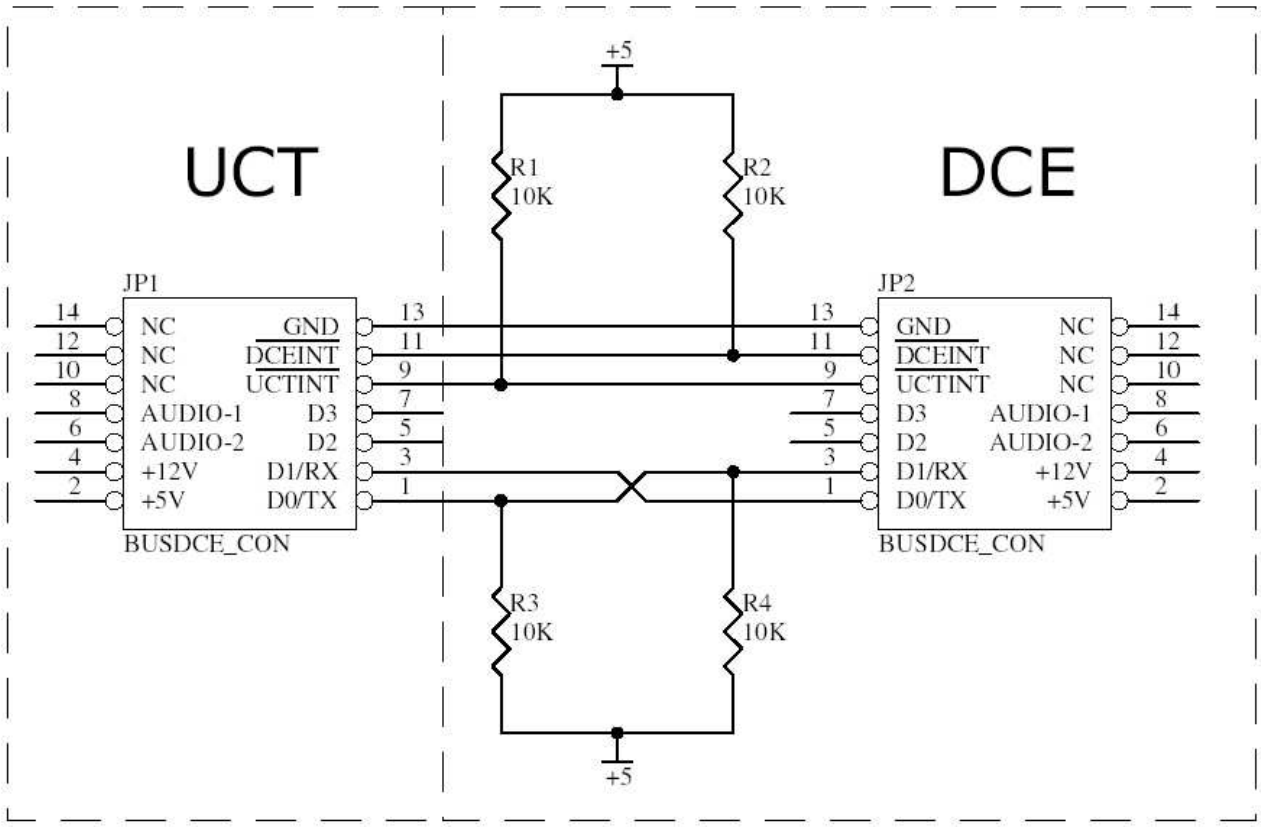
- Comprobar la respuesta a una trama HELO por parte del receptor.
- Comprobar que el numero de comando o función CMD, en la trama FINISHED coincida con el CMD de la trama COMMAND.

RESUMEN:

1) Transmisor envía HELO y espera a que el receptor conteste con otro HELO.
2) Transmisor envía comando a ejecutar en la trama COMMAND y espera.
3) Ejecutado el comando, el receptor envía la trama FINISHED.
4) El receptor y transmisor vuelven al punto 1).

Los comandos soportados por el UCT y otras cuestiones se detallan mas adelante.

3.7.3: Circuito propuesto para conectar un DCE al UCT, usando el Bus DCE



En este circuito básico, se detalla como debe conectarse el DCE al UCT. El conector JP1, se encuentra en el UCT. El conector JP2 y las resistencias se encuentran en el DCE. Los pull-ups R1...R4, son para evitar posibles ruidos cuando manipulamos los conectores de ambos aparatos o para cuando las líneas son inestables, generalmente en el RESET de algún microcontrolador. También evitan ruidos producidos por UCT al acceder al Bus DCE (debido a su hardware, ver esquemáticos).

Notemos que muchos pines del Bus DCE no fueron utilizados en este ejemplo, pero pueden utilizarse si usted los necesita. También, observe que las interrupciones son generadas por nivel BAJO, ello es 0V. Y cuando no hay interrupciones hay 5 V , en los pines /DCEINT y /UCTINT.

Atención!: No conectar este bus directamente al puerto de una computadora, los niveles de tensión no son iguales, aquí se utilizan 0-5 Volts, en una PC de utilizan –12/+12 V. Utilizar un MAX232 como interface.

3.8: Comandos Soportados por el UCT

La siguiente es una lista y resumen de los comandos que pueden ser ejecutados en el UCT a través del Bus DCE. Los mismos deben especificarse en el byte CMD de la trama **COMMAND** del protocolo CTP. Una explicación completa de los comandos se detalla en la sección 3.8.1.

COMANDO	VALOR	DESCRIPCION
END_CTP	0xF5	El UCT termina la comunicación con DCE y toma el control.
UCT_RESET	0x99	El UCT se resetea y vuelve a modo de espera.
FREE_PHONE_LINE	0x98	El UCT desocupa línea telefónica.
GET_PHONE_	0x88	El UCT ocupa la línea telefónica.
DIALP_NUMBER	0xCC	El UCT marca el numero en ARG1 por pulsos.
PING	0x87	No hace nada, solo devuelve PING.
UCT_PAUSE	0x55	El UCT se queda pausado por ARG1 segundos.
PLAYISD	0x66	Reproduce el mensaje ARG2 de la ISD1420.
WR_EEPROM	0x77	Escribir ARG1 en la dirección ARG2 de la EEPROM.
RD_EEPROM	0x44	Leer dirección ARG1 de la EEPROM.
WR_RAM	0x3A	Escribir ARG1 en dirección ARG2 de la RAM.
RD_RAM	0x13	Leer dirección ARG1 de la RAM.
GETDTMF	0xEE	Esperar por ARG1 segundos un código DTMF.
GETDTMFSTR	0xDD	Esperar 1 minuto por ARG1 códigos / dígitos DTMF.
CTRLDISP	0xBB	Activa o desactiva dispositivos ON/OFF.
GOTO_MAIN_MENU	0x93	El UCT toma el control y reproduce el menú principal.

Los valores de comandos estan definidos en uct.inc.

3.8.1: Detalles Sobre los Comandos Soportados por el UCT

Antes de continuar se sugiere repasar la sección 3.7.2, las tramas COMMAND y FINISHED del protocolo CTP.

Definiciones:
ARG=ARGUMENTO (Trama Command) , RET=RETORNO (Trama Finished)

Comandos Soportados Por el UCT:

UCT_RESET: El UCT se resetea y vuelve a modo espera.
ARG: Ninguno.
RET: Ninguno.

END_CTP: El UCT termina la comunicación con DCE.
ARG: Ninguno.
RET: Ninguno.

GET_PHONE_LINE: El UCT ocupa línea telefónica.
ARG: Ninguno.
RET: Ninguno.

FREE_PONE_LINE: Desocupa / Libera la línea telefónica.
ARG: Ninguno.
RET: Ninguno.

UCT_PAUSE: Establece una pausa en el UCT de ARG1 segundos.
ARG: ARG1 = segundos.
RET: Ninguno.

PLAYISD: Reproduce el mensaje numero ARG2 de la ISD1420. Si un código DTMF es recibido, lo toma.
ARG:
ARG1 = Segundos o timeout, ver getdtmf() en uct.asm. Por lo general poner 30.
ARG2 = Numero de mensaje reproducir.
RET:
R1 = Si llego DTMF, retorno es según función getdtmf().
R2 = Numero de mensaje reproducido o 0x00 si llego un dtmf.

GETDTMF: Espera ARG1 segundos por un codigo DTMF.
ARG: ARG1 = segundos de espera o timeout antes de retornar.
RET: R1 = según función getdtmf(), ver uct.asm.

GETDTMFSTR: Espera 1 Minuto, por ARG1 códigos DTMF.
ARG: ARG1 = números de códigos o dígitos DTMF a esperar (max. 4).
RET:
R1 = DTMF recibido. (Nible superior e inferior)
R2 = DTMF recibido. (Nible superior e inferior)
NOTA: Si ARG1 < 4, los DTMF faltantes en R2:R1, se completan con cero.
En el nible inferior de R0 se deposita el ultimo DTMF recibido.

CTRLDISP: Activa o Desactiva dispositivos ON/OFF.
ARG:
ARG1 = S000XXXX.
Donde si S: 1-> Activa dispositivo. 0 -> Desactiva dispositivo.
Y XXXX representa el numero de dispositivo a activar.
RET: Ninguno.
NOTA: Mayor información se puede encontrar en función ctrl_disp(), de uct.asm.

WR_EEPROM: Escribe en la memoria de datos EEPROM del UCT. (PIC16F84A)
ARG:
ARG1 = Dato a escribir.
ARG2 = Dirección a escribir.
RET: Ninguno.
NOTA: Tenga cuidado al escribir alguna dirección de la EEPROM, ya que puede ser utilizada por el software UCT para su configuración. Vea uct.inc .

RD_EEPROM: Lee la memoria de datos EEPROM del UCT. (PIC16F84A)
ARG: ARG1 = Dirección a leer.
RET: R1 = Valor leído.

RD_RAM: Lee la memoria de datos RAM del UCT. (BANK0, PIC16F84A).
ARG: ARG1 = Dirección a leer.
RET: R1 = Valor leído.

WR_RAM: Escribe en la memoria de datos RAM del UCT. (BANK0, PIC16F84A).
ARG:
ARG1 = Dato a escribir.
ARG2 = Dirección a escribir.
RET: Ninguno
NOTA: Usar con cuidado! No intentar cambiar de banco!.

DIALP_NUMBER: El UCT marca el numero ARG1 por Pulsos.
ARG: ARG1 = Numero a marcar
RET: Ninguno. Ver función dialp() para mas info., en uct.asm. No se comprueban errores. Se debe marcar de a un numero por vez y debe estar entre 0 y 9.
NOTA: Según las normas telefónicas del marcado por pulso, se debe esperar al menos 800 mS +20%/-10%, entre cada numero discado. Y se deben esperar al menos 600 mS antes de discar el primer numero, al ocupar la línea telefónica (esto es al menos en Argentina).

GOTO_MAIN_MENU: El UCT toma el control y reproduce el menú principal.
ARG: Ninguno.
RET: Ninguno.

3.8.2: Formas de Conectarse UCT por el Bus DCE

La conexión al UCT desde un DCE puede realizarse de dos formas:

1) El usuario, luego de llamar al UCT, entra al menú “Bus DCE” (según se describe en las secciones 2.2.2 y 2.2.2.4).
2) El UCT se encuentra en modo de espera y recibe una interrupción por el DCE.

En el primer caso, es el usuario quien desea establecer una conexión al DCE. Aquí, al ingresar el menú “Bus DCE”, el UCT se pondrá a esperar una conexión por parte del DCE.
Como sabe el DCE que el UCT esta esperando una conexión ???.
En la sección 3.7.1, se comento que el UCT cada vez que accede al Bus DCE, ya sea para transferir o recibir datos, pone el pin **/DCEINT** a nivel bajo (0V).
Esto puede ser utilizado por el DCE para saber que el UCT esta esperando una conexión. Entonces aquí el DCE debe iniciar con una Trama HELO y seguir el protocolo CTP.
Notar que aquí el DCE, debe crear un menú coherente con los comandos soportados por el UCT, así el usuario puede interpretarlos. Si el usuario, decide salir del DCE, el DCE debe interpretar tal intención y enviar el comando END_CTP al UCT para que el UCT vuelva a tomar el control y reproduzca sus menús.

En el segundo caso, la situación es la siguiente: El UCT se encuentra en modo de espera, ello es esperando una conexión del DCE o un llamado por parte del usuario. Si el DCE tiene intenciones de conectarse al UCT, debe poner la el pin **/UCTINT** a nivel bajo (0V) para generar una interrupción, luego esperar al menos 150 mS para que el UCT despierte y se encuentre listo para recibir una conexión por el Bus DCE (también se puede mirar el estado del pin **/DCEINT**).
Posteriormente iniciar la conexión como se especifica en el protocolo CTP. Aquí, el DCE podrá como siempre hacer la combinación de comandos que desee para lograr algo útil en el UCT.
Luego puede finalizar enviando el comando END_CTP o UCT_RESET si deseamos reinicializar el UCT. Notemos que si el UCT accede al puerto para esperar una trama y esta no llega en 30 segundos, el UCT finalizara su conexión y volverá a su estado anterior.

Otro caso es el de conectarse mientras el UCT esta atendiendo al usuario en algún menú cualquiera distinto el de “Bus DCE”. Actualmente esta forma **NO** esta soportada, el DCE no puede interrumpir al UCT en este caso. Debe ser obligación del usuario, que al llamar verifique por medio del menú “Bus DCE”, si un DCE quiere comunicarse. También es poco probable esta ultima situación.

3.8.3: Cuestiones a tener en Cuenta al conectarse el UCT

Aquí resumimos algunas reglas o cuestiones:

- Cada vez que el UCT espera para recibir una trama de bytes o desea enviar una trama de bytes, pone el pin **/DCEINT** a nivel bajo (0V).
- El UCT esperara una trama de bytes como máximo 30 segundos. De lo contrario vuelve a su estado anterior (modo de espera, reproducción de menús, etc).
- El cuando el UCT ejecuta un comando, el retorno de dicho comando a través de una trama FINISHED, será en un tiempo definido de acuerdo a la duración del comando. Ejemplo: Los segundos que tarde el UCT en reproducir el mensaje “Bienvenido”, si el DCE requirió reproducir tal mensaje. El DCE puede saber si el UCT retorno, mirando el pin **/DCEINT.**
- Debe considerarse al UCT como un “servidor de comandos”, y el DCE debe saber cuando conectarse (vea sección 3.8.2).
- El UCT hará una pausa de 3 ms como mínimo antes de transmitir una trama, para que DCE lentos puedan seguir el ritmo de la conexión.
- Recordar despertar al UCT poniendo un nivel bajo (0V) en **/UCTINT.**
- Verificar que el UCT ejecuto el comando correcto, ello se puede verificar comparando el byte CMD de la trama FINISHED con el byte CMD de la trama COMMAND. Ambos deben coincidir.

3.8.4: Notas para Desarrolladores de un DCE

Si usted decide construir un DCE, lo dicho anteriormente es suficiente, pero como siempre, si desea profundizar mas lea el archivo uct.inc, la definiciones del CTP. También, encontrara muy útil la función uct_ctpd(), del archivo uct.asm, quien es la encargada de poner al UCT como servidor de comandos.
Cuando diseñe un DCE, haga que el DCE verifique la versión de software grabada en el UCT, ello se puede hacer leyendo las direcciones FC_2, FV_1 y FV_0 de la memoria EEPROM del UCT. Mire archivo uct.inc. Esto puede evitar que el DCE intente ejecutar comandos no soportados en otras versiones.
También, puede preguntarme algunas cuestiones por e-mail, siempre y cuando no se hallan explicado aquí o no se detallen claramente.

3.8.5: Ejemplos de Conexión al Bus DCE

Cada comando enviado al UCT sigue la siguiente secuencia de tramas:

DCE -> HELO -> UCT
UCT -> HELO -> DCE
DCE -> COMMAND -> UCT (ejecuta comando)
UCT -> FINISHED -> DCE (recibe resultado)

Un ejemplo clásico, seria:

Enviar los siguiente comandos:

1) CMD=GET_PHONE_LINE
2) CMD=DIALP_NUMBER , ARG1=1
3) CMD=DIALP_NUMBER , ARG1=0
4) CMD=DIALP_MUMBER , ARG1=1
5) CMD=PLAY ISD , ARG1=60 , ARG2=30
6) CMD=FREE_PHONE_LINE

En este ejemplo, el UCT ocupo la linea telefónica, Marco el numero “101” (policía en Argentina) y le reprodujo el mensaje “ALARMA” (numero 30) del UCT. Luego
Libero la línea telefónica.
En el ejemplo anterior no se mostró, pero el DCE debería esperar 600 ms antes de marcar el primer numero telefónico y 800 ms entre cada numero. Según se especifico en sección 3.8.1, comando DIALP_NUMBER. Un timeout no se producirá, se pueden esperar esos tiempos entre comando y comando, ya que si recordamos, el UCT esperara 30 segundos como máximo entre cada trama.

Otro ejemplo podría ser, grabar algunos mensajes en la computadora. Conectar la salida de la placa de sonido a las líneas AUDIO-1 y AUDIO-2 del Bus DCE, y reproducirlos luego por la línea telefónica, previa ocupación de la misma.
Si el mensaje dura mucho, podemos simultáneamente enviar un UCT_PAUSE, para que no se produzca un timeout. También, podemos al mismo tiempo esperar algún código dtmf, con GETDTMF.

Otro ejemplo?, bueno, como vemos las posibilidades son infinitas. Quizás en la pagina publique algunos DCE o también puede mandarme sus DCE construidos para que los publique.

3.8.6: Mensajes almacenados en la memoria del UCT

Ellos deben ser reproducidos con el comando PLAYISD, especificando su numero en ARG2 . También puede leerse del archivo uct.inc, donde se enumeran:

Mensaje	Numero
SILENCIO (*)	1
SILENCIO (*)	2
“NUMERO”	3
“INCORRECTO”	4
“INGRESAR”	5
“DISPOSITIVO”	6
“LISTO”	7
“REPETIR”	8
“DESACTIVADO”	9
“ACTIVADO”	10
“CLAVE”	11
“CAMBIAR”	12
“SALIR”	13
“CONFIGURAR”	14
“ERROR”	15
“CONTROL DE”	16
“VER ESTADO DE”	17
“BUS DCE”	18
“UNO”	19
“DOS”	20
“TRES”	21
“CUATRO”	22
“CINCO”	23
“ASTERISCO”	24
“NUMERAL”	25
“EXTERNO”	26
“CONECTAR”	27
“RINGS”	28
“TELEFONICO”	29
“ALARMA”	30
“BIENVENIDO”	31

(*) Contienen solamente silencio.

3.8.7: Limitaciones del Bus DCE

Debido a que el PIC16F84A utilizado en el UCT contiene muy poca memoria, debió exprimirse hasta el ultimo byte para que las rutinas del Bus DCE entraran en el microcontrolador. La versión de software 0.8.1 ocupa el 99.8 % del total de memoria, quedando disponibles algunos bytes. Por ello no se pudo crear un protocolo demasiado elaborado. También notara la baja velocidad de 1200 bps, ello es debido a que el PIC16F84A no tiene USART y la transmisión serie debió ser realizada por software. A mayor velocidad (en la teoría y en la practica) debido al cristal utilizado, se producían errores.

Un análisis simple, muestra que si cada trama ocupa 3 bytes, y se necesitan las seguientes tramas para ejecutar un comando:

TX -> HELO -> RX
RX -> HELO -> TX
TX -> COMMAND -> RX
RX -> FINISHED -> TX

Es decir 4 tramas, en total 12 bytes transferidos y si suponemos que las mismas se envían una a continuación de la otra de forma inmediata a una velocidad de 1200 bits/segundo. Podríamos ejecutar un comando cada 100 mS, lo que es lo mismo a 10 comandos por segundo.
Lo cual debido a la naturaleza de cada comando del UCT, la velocidad es suficiente.

Porque se llama Bus DCE, si no es un BUS propiamente dicho?.
Hace 2 años cuando desarrolle el UCT, plante todo lo que se escribe en este documento de forma teórica, muchas ideas originales no se pudieron concretarse por cuestiones de tiempo y hardware. El Bus DCE, estaba originalmente diseñado para soportar múltiples DCE, pero debido a que solo estaba aprendiendo el uso por primera vez del PIC16F84A, no podía aspirar demasiado. Por ello quedo “Bus DCE”, como la forma de conectarse al UCT con un DCE, aunque solo pueda hacerlo con un DCE a la vez. Sin embargo sigue siendo útil ;).

Se piensa modificar / mejorar el hardware del UCT?
Quizás. Pero poco probable, actualmente estoy muy ocupado con otros proyectos.

Se piensa modificar / mejorar el software del UCT?
Si, ya que demandan menos tiempo ☺. De hecho, ya han salido numerosas versiones del software del UCT.

4.0: Notas de Construcción:

Esta sección pretende ser una guía general para construir al UCT. Lógicamente existen muchos caminos para llegar al mismo resultado. Usted debe tener experiencia en este tipo de circuitos ya que explicare los pasos generales:

1) Obtener los componentes electrónicos.

Luego de obtener los componentes electrónicos del proyecto, aconsejo comprar zócalos para la ISD1420, PIC16F84A, CD4066 (IC9) y el opto acoplador 4N25, que puede ser útil cambiarlos una vez que estén en la plaqueta.
También puede comprar un transformador de 220 Vca a 12Vca de 500 mA para poder alimentar al UCT.
Gabinetes, borneras, cables, etc a Gusto.
Con respecto al Bus DCE yo utilice cable IDC o plano (como el de los disco duros) de 14 hilos con su respectivos conectores hembras en sus puntas y en la plaqueta utilice 2x7 pin-headers o jumpers, que en total hacen 14 conectores machos. Luego en el gabinete puse un DB15 para dejarlo mas presentable.... pero eso es relativo.
Los interruptores S1, S2 y leds son a gusto, siempre y cuando sean los correctos para el circuito.
El transformador TF1, es de relación 1:1 y una impedancia de 600 Ohms a 400 Hz. El mismo lo puede encontrar como transformadores para telefonía y son baratos. Los módems de PC traen un transformador del mismo tipo

2) Grabar Mensajes de Audio y Software .

En la pagina del proyecto (ver sección **0.2**) usted debe bajar las ultimas versiones disponible del:

Software:
Este contiene las instrucciones en assembler del PIC16F84A, que usted debe ensamblar (puede usar el MPLAB de Microchip) y grabárselas al PIC16F84A mediante un programador (puede usar el programador JDM). Lea el README.txt que viene con tales archivos.

Mensajes de Audio:
Los mensajes de audio están en formato **.wav** y deben ser grabados en el orden correcto (según se especifica en uct.inc o en la sección 3.8.7) a la memoria ISD1420.
Luego estos mensajes de audio serán reproducidos a través de la línea telefónica por el UCT.
Debido a que no encontré ningún grabador gratuito para este tipo de memorias decidí construirme uno.
El mismo se llama **RAP-ISD** y funciona para GNU/LINUX. Puede bajarse en la pagina del proyecto (ver sección **0.2**). Leer su README.TXT para saber como funciona.

3) PCB
El PCB es el circuito impreso del UCT y sirve para hacer la plaqueta.

En la pagina del proyecto esta disponible el PCB a escala y en formato PDF. También esta disponible en formato imagen PNG.

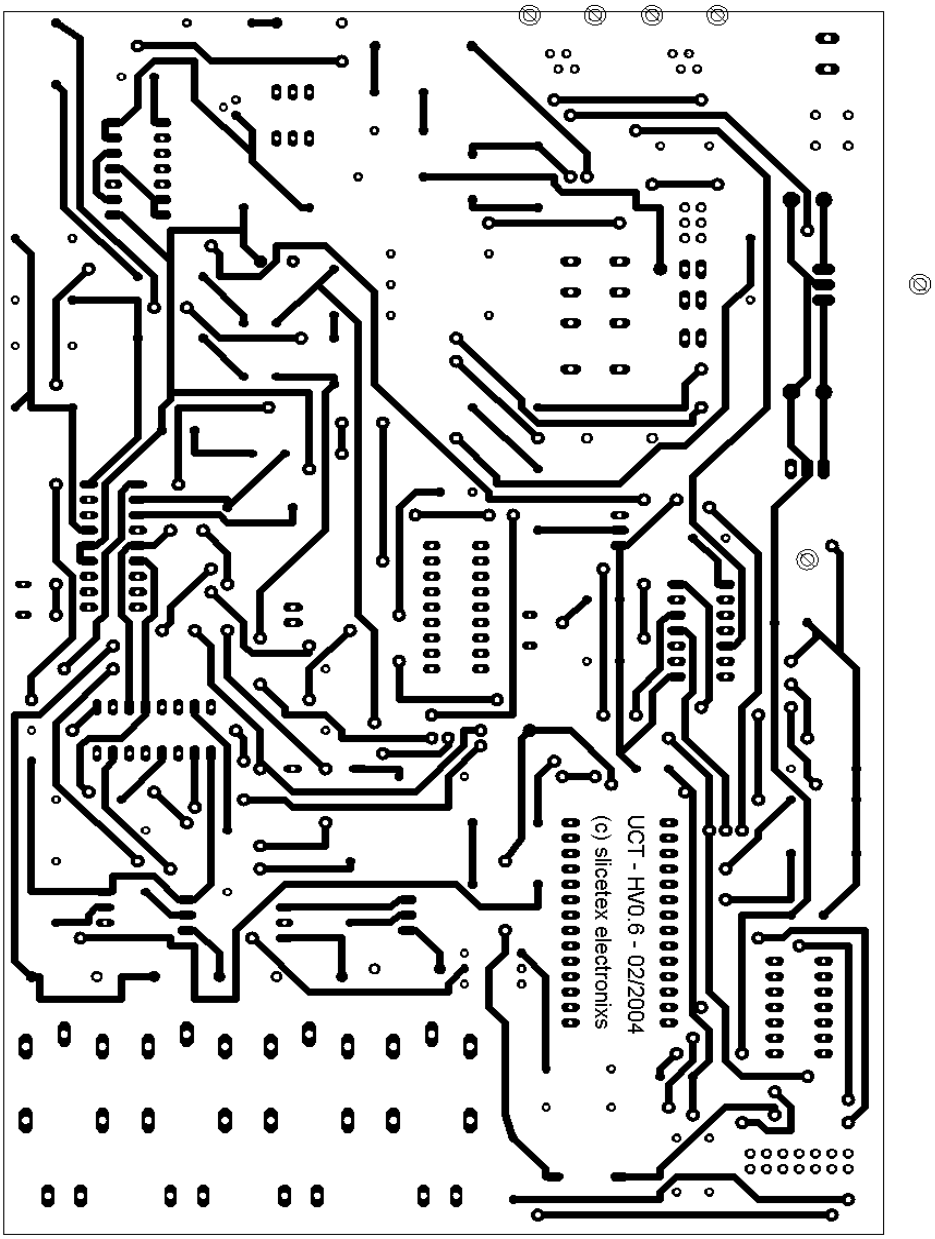
La plaqueta electrónica del UCT es en doble Faz (ambas caras de la plaqueta tienen pistas electrónicas) y su dimensiones son de: 135.89 mm x 201.93 mm.
Recomiendo realizarla en una plaqueta de 15 cm x 21 cm para tener suficiente espacio y de material fibra de vidrio, para mantener buenas prestaciones del circuito con el paso del tiempo.

Con respecto a la construcción usted puede mandarme las dudas especificas por e-mail, pero no aceptare preguntas del tipo: Como Grabo el PIC? Como grabo la ISD1420? Como realizo el PCB?, etc... igualmente este tipo de preguntas generales se pueden aplicar al resto del documento (RTFM!) y creo que fueron bien documentadas.

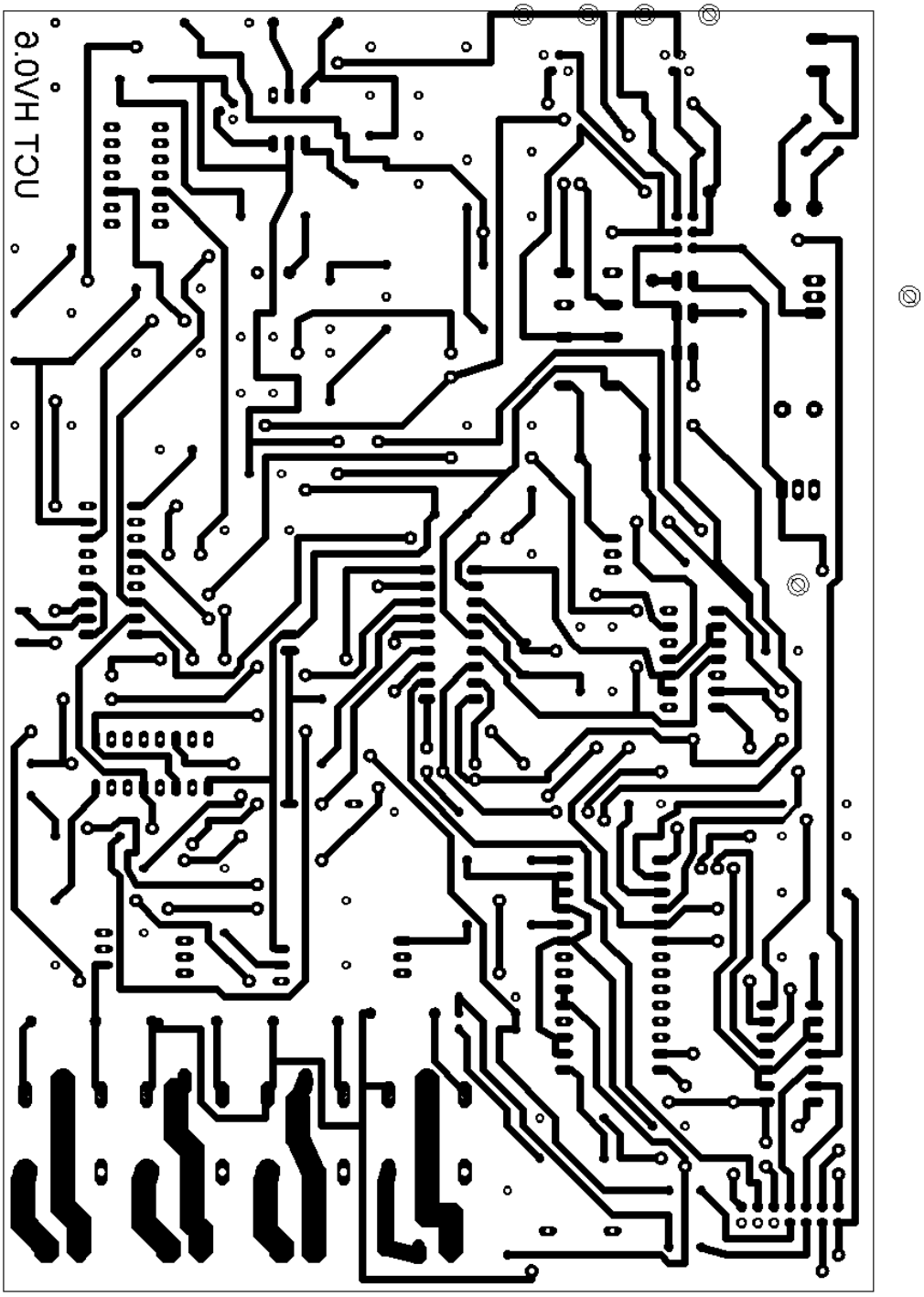
5.0: Apéndice / Anexo

5.1: PCB (PRINTED BOARD CIRCUIT) del UCT

5.1.1 TOP VIEW



5.1.2 BOTTOM VIEW



NOTA: El PCB no esta a escala. Bajarlo desde la pagina del proyecto. En formato PDF.

5.2: Software del PIC16F84A usado en el UCT

Debe bajarlo de la pagina del proyecto (ver sección 0.2). Aunque igualmente los adjuntamos en el documento.

ATENCIÓN: Quizás no es la ultima versión disponible y puede contener menos funcionalidades y mas errores que la disponible en la pagina web.
Ello es debido a que no actualizare el documento con la misma frecuencia del software. Por otro lado para imprimirlos, se visualiza mejor en los archivos de la pagina web, ya que al transferirlo a este tipo de documento los espacios en blanco entre las líneas del programa se desordenan. Pero solo es una cuestión visual, el programa debería funcionar de forma correcta si se lo copia.

Archivo: UCT.ASM (codigo principal)
Archivo: UCT.INC (constantes, variables, macros, etc de UCT.ASM).

[ARCHIVO: UCT.ASM]

```
; asmsyntax=pic16f84
;
; File     : uct.asm
; Compiler : mplab v6.61
;
; MCU      : PIC-16F84A @ 4 MHz
;
; Created  : 18/Sep/2003
; Revision : 05/Jan/2005
; Version  : 0.8.1
;
; Author   : Boris Estudiez <stk@freeshell.org>
; Licence  : GPL
;
; DESCRIPTION:
; Software de control del UCT (Unidad Controlada por Telefono).
;
; MODEL/ID : STX600
;
; HARDWARE VERSION: UCT HV-0.5 o superior.
;
; CONFIGURATION BITS: WDT=ON, OSC=XT, PWRTE=OFF, CP=OFF.
; EEPROM DATA VALUES: Mire el archivo README.TXT que se adjunta. .
;
; (c) Copyright slicetex electronixs (2001-2005)
;     http://stk.freeshell.org
;

   LIST P=PIC16F84A
   #include <p16f84a.inc>
   #include "uct.inc"

   __CONFIG _PWRTE_OFF & _WDT_ON & _XT_OSC & _CP_OFF

;*****************************************************************
;*                        INICIO DEL CODIGO DE PROGRAMA          *
;*****************************************************************
RESET: ORG 0x00
   clrf INTCON          ;Interrupciones Desactivadas.
   goto SETUP

;*****************************************************************
;*                        RUTINAS DE ATENCION A INTERRUPCIONES   *
;*****************************************************************

;*****************************************************************
;* RSI: Rutina de Servicio a Interrupciones.                     *
;*****************************************************************
RSI: ORG 0x04

   bcf INTCON, GIE       ;Inhibir toda interrupcion
   call push_STATUS      ;Salvamos registro STATUS y W.
   call push_w
```

```
   ;RESPETAR ORDEN DE COMPROBACION DE INTERRUPCION.
   btfsc INTCON, T0IF    ;Int. TMR0 Overflow?
      goto RSITMR0       ;Si, atender int. por TMR0.
   btfsc INTCON, INTF    ;Int. externa?
      goto int_ext       ;Si, determinar tipo de interrupcion.
   goto end_RSI          ;No, salir.

int_ext:
   movlw d'25'           ;Esperamos que tensiones se estabilicen
   call mpause           ;en los puertos.
   btfss UCTINT_PIN      ;Interrupcion por BUS DCE ?
      goto RSI_DCE       ;Si, int. por BUS DCE. -> UCTINT_PIN=0.
   goto RSIR             ;No, int. por Detector de RINGS.

end_RSI:                 ;Devolver control a programa.
   call pop_w            ;Reponer W y STATUS.
   call pop_STATUS
   retfie                ;Devolver control.

;*********************************************************************
;* RSIR: Rutina de Servicio a Interrupcion por RING.                 *
;* Modifica: RINGS_LEFT, TMR0 y UCT_STATUS.                          *
;*********************************************************************
RSIR:
   bcf INTCON, INTF              ;INTF=0, reponer flag.

   bsf UCT_STATUS, RINGING       ;Intento de llamada. RINGING=1
   movlw S10
   call set_timeout              ;Seteo TIMEOUT en 10 segundos.
   decf RINGS_LEFT, F            ;RINGS_LEFT--

   movf RINGS_LEFT, F
   btfsc STATUS, Z               ;RINGS_LEFT==0?
      bsf UCT_STATUS, ANSWERCALL ;Si, ANSWERCALL=1 -> responder llamada.

end_RSILL:
   goto end_RSI                  ;Devolver control a programa.

;*********************************************************************
;* RSI_DCE: Rutina de Servicio a Interrupcion por bus DCE.           *
;* Modifica: UCT_STATUS.                                             *
;*********************************************************************
RSI_DCE:
   bcf INTCON, INTF              ;INTF=0, reponer flag.
   bsf UCT_STATUS, DCEINT        ;DCEINT=1.

   btfsc UCT_STATUS, RINGING     ;RINGING==1???
      call unset_timeout         ;Si, Sacar TIMEOUT debido a RSIR.

end_RSI_DCE:
   goto end_RSI                  ;Devolver control a programa.
```

```
;*********************************************************************
;* RSITMR0: Rutina de Servicio a Interrupcion por TMR0 Overflow.     *
;* Dependiendo de los valores de los bits del UCT_OPTION toma        *
;* una accion. Este servicio solo puede ser usado con                *
;* la funcion set_timeout().                                         *
;* SI:                                                               *
;* QTIMEOUT-> 1: vuelve a stand-by ; 0:vuelve al programa.           *
;*                                                                   *
;* Setea con 1 el flag TIMEOUT de UCT_STATUS al ocurrir un timeout.  *
;* Nota: esta rutina deja seteado el Banco 0.                        *
;*********************************************************************
RSITMR0:
   bcf INTCON, T0IF           ;T0IF=0, Reponer flag.

   BANK_0
   movlw 0x06                 ;Recordar que TMR0 esta en el banco 0.
   movwf TMR0                 ;Proxima interrupcion en 250 cuentas.

   decfsz TMR0_AUX, F         ;TMR0_AUX == 0?, paso 1 segundo?
      goto end_RSITMR0        ;No, seguir esperando.
   movlw d'125'               ;Si cargar TMR0_AUX con 125 y decrementar
   movwf TMR0_AUX             ;SECONDS.
   decfsz SECONDS, F          ;SECONDS==0? TIMEOUT?
      goto end_RSITMR0        ;No, seguir esperando.
   bsf UCT_STATUS, TIMEOUT    ;Si, setear TIMEOUT.
   btfsc UCT_OPTION, QTIMEOUT ;QTIMEOUT==1?
      call close_call         ;Si, inicializar UCT (RESET).

end_RSITMR0:
   goto end_RSI               ;devolver control.


;*********************************************************************
;*                   RUTINA PRINCIPALES DEL UCT                      *
;*********************************************************************
;* Notas de programacion:                                            *
;* 1) Banco 0 se supone siempre por defecto, si se cambia a          *
;*    banco 1 en una rutina, luego al salir, volver a poner a 1.     *
;* 2) QTIMEOUT se supone siempre 1, si se pone a 0 en una rutina     *
;*    luego al salir de rutina volver a poner a 1.                   *
;*********************************************************************

;*********************************************************************
;* SETUP: Seteamos Condiciones Iniciales de Funcionamiento.          *
;*********************************************************************
SETUP:

   BANK_1            ;accedemos al banco 1 de la mem. ram
   ;Config. Puerto A
   movlw b'00010000' ;A7-5=X,A4=E,A3=S,A2=S,A1=S,A0=S
   movwf TRISA
   ;Config. Puerto B
   movlw b'00000011' ;B7-4=S,B3=S,B2=S,B1=E,B0=E
   movwf TRISB

   ;Conf. OPTION_REG Reg.
   movlw b'1000111'
```

```
   movwf OPTION_REG  ;RBPU=OFF PSA=ON->TMR0 DIV=111, INTEDG=0 FALL EDGE.

   BANK_0            ;accedemos al banco 0 de la mem. ram

   ;Valor inicial Puerto A
   ;A0 | A1 | A2 | A3 | Poner Dispositivo en el bus RB4-7:
   ; 1 |  0 |  0 |  0 |8870 (DTMF DECODER)
   ; 0 |  1 |  0 |  0 |ISD1420/4016 (Memoria De Voz)
   ; 0 |  0 |  1 |  0 |4042 (LATCH/DISP. ON OFF)
   ; 0 |  0 |  0 |  1 |BUS DCE/4016
   ; 0 |  0 |  0 |  0 |Ningun Dispositivo.

   clrf PORTA        ;Limpiamos Latch's de puerto A, Salidas=0.

   ;Valor inicial Puerto B
   ;Telefono/Linea Desocupado.
   clrf PORTB        ;Limpiamos Latch's de puerto B, Salidas=0
   bsf DCEINT_PIN    ;DCEINT_PIN=1 -> RB2=1.

   ;Actualizamos estado de Dispositivos ON/OFF.
   movlw 0x05        ;W > 4.
   call ctrl_disp    ;Actualizar Latch 4042. Ver ctrl_disp().

   ;Borramos registro de estado y opciones del UCT
   clrf UCT_STATUS

   ;SKIPMENU=0, QMSG=0, QF_DTMF=0, ISD_SURE_RESET=0.
   clrf UCT_OPTION
   bsf  UCT_OPTION, QTIMEOUT ;QTIMEOUT=1 -> salir si hay timeout.

   ;Seteamos variables
   movlw EE_RINGS
   call eeprom_read     ;W=*EE_RINGS
   movwf RINGS_LEFT     ;Numero de RINGS que faltan para contestar llamado.

   ;Activar Interrupcion solo externa por RB0/INT
   movlw b'10010000'
   movwf INTCON         ;GIE=1, INTE=1

;*****************************************************************
;* wait_event: Mantiene al UCT a la espera de un LLAMADO o conexion  *
;* por BUS DCE.                                                  *
;*                                                               *
;* Debido a un error inexplicable, cuando usaba 'sleep' aqui, el     *
;* DCE no podia interrupir al UCT. Por ello no lo uso :( .           *
;*****************************************************************
wait_event:

   clrwdt                       ;WDT=0.
   btfsc UCT_STATUS, ANSWERCALL ;RESPONDER LLAMADO? ANSWERCALL==1?.
      goto reply_call           ;Si!.
   btfss UCTINT_PIN             ;Interrupcion por DCE? UCTINT_PIN==0?
      goto dce_connection       ;Si!, establecer coneccion con DCE.
   goto  wait_event             ;No, Seguir esperando algun evento.
```

```
;*********************************************************************
;*                  RUTINAS DE ATENCION DE LLAMADO DEL UCT           *
;*********************************************************************

;*********************************************************************
;* reply_call: establece la comunicacion entre el UCT y el usuario   *
;* que llama al UCT.                                                 *
;*********************************************************************
reply_call:

   BANK_0
   call unset_timeout          ;Sacar TIMEOUT introducido por RSIR.
   bcf INTCON, INTE            ;NO permitir Interrupcion Externa.
   bcf UCT_STATUS, RINGING     ;RINGING=0
   bsf UCT_STATUS, ANSWERCALL  ;ANSWERCALL=1 - UCT respondiendo llamada.
   bsf UCT_OPTION, QMSG        ;QMSG=1, Desconectar con mensage de aviso.

   bsf PORTB, 3      ;RB3=1, Ocupar linea, al circ. selector de modo.
   movlw S2
   call spause       ;Esperar aprox. 2 segundos antes de contestar

   ;Pedimos clave de acceso de 4 digitos. y damos 3 oportunidades.
   movlw 3
   movwf cx          ;cx = 3

for_passwd:
      movlw ISD_INGRESAR
      call play_isd        ;reproducir "INGRESAR"
      movlw ISD_CLAVE
      call play_isd        ;reproducir "CLAVE"

      movlw 4
      call get_dtmfstr     ;Tomar 4 digitos DTMF y almacenar en AX:BX.
      movlw EE_PASSW_0     ;Comprobamos password.
      call eeprom_read
      xorwf ax, W
      btfss STATUS, Z      ;ax==EE_PASSW_0? (digito 3 y 2)
         goto wrong_pass   ;No, passwd incorrecto.
      movlw EE_PASSW_1
      call eeprom_read
      xorwf bx, W
      btfss STATUS, Z      ;bx==EE_PASSW_1?  (digito 1 y 0)
         goto wrong_pass   ;No, passwd incorrecto.
      goto end_for_passwd  ;Passwd Correcto.

wrong_pass:
         movlw ISD_CLAVE
         call play_isd        ;"CLAVE"
         movlw ISD_INCORRECTO
         call play_isd        ;"INCORRECTO"

         decfsz cx, F         ;Pedir de nuevo clave?
            goto for_passwd   ;Si. Uhh :)
end_for_passwd:

   movf cx, F
   btfsc STATUS, Z              ;Desconectar? cx==0?
```

```
      call close_call            ;Si, claves incorrectas. Desconectar :(

   bcf UCT_OPTION, QTIMEOUT      ;QTIMEOUT=0
   movlw S2
   call getdtmf                  ;Esperar 2 segundos para DTMF_0.
   bsf UCT_OPTION, QTIMEOUT      ;QTIMEOUT=1

   xorlw 0x00
   btfsc STATUS, Z               ;DTMF_0 recibido?
      bsf UCT_OPTION, SKIPMENU   ;Si, UCT_OPTION/SKIPMENU=1

;*********************************************************************
;* uct_main_menu_start: Rutina de Menu principal.                    *
;*********************************************************************
uct_main_menu_start:
   movlw ISD_BIENVENIDO         ;Reproducir solo por primera vez:
   call play_isd                ;"BIENVENIDO"

uct_main_menu:

   btfsc UCT_OPTION, SKIPMENU   ;SKIPMENU==1?
      goto get_menu_option      ;Si, omitir mensages del main_menu.
   movlw ISD_UNO
   call play_isd                ;"UNO"
   movlw ISD_CONTROL_DE
   call play_isd                ;"CONTROL DE"
   movlw ISD_DISPOSITIVO
   call play_isd                ;"DISPOSITIVO"
   movlw ISD_DOS
   call play_isd                ;"DOS"
   movlw ISD_VER_ESTADO_DE
   call play_isd                ;"VER ESTADO DE"
   movlw ISD_DISPOSITIVO
   call play_isd                ;"DISPOSITIVO"
   movlw ISD_TRES
   call play_isd                ;"TRES"
   movlw ISD_BUS_DCE
   call play_isd                ;"BUS_DCE"
   movlw ISD_CUATRO
   call play_isd                ;"CUATRO"
   movlw ISD_CONFIGURAR
   call play_isd
   movlw ISD_RINGS
   call play_isd                ;"CONFIGURAR RINGS"
   movlw ISD_CINCO
   call play_isd                ;"CINCO"
   movlw ISD_CAMBIAR
   call play_isd                ;"CAMBIAR"
   movlw ISD_CLAVE
   call play_isd                ;"CLAVE"
   call play_sal_rep            ;"NUMERAL SALIR, ASTERISCO REPETIR".

   ;Tomamos DTMF y ejecutamos lo que corresponda.
get_menu_option:
   movlw S60                  ;Tiempo de espera maximo para getdtmf.
   call getdtmf
   movwf ax                   ;ax=DTMF recibido.
```

```
   movf ax, F
   btfsc STATUS, Z            ;ax==DTMF_0 ?
      goto skip_menu_msg      ;Si, des/activar UCT_OPTION/SKIPMENU
   movlw 0x01
   xorwf ax, W
   btfsc STATUS, Z            ;ax==DTMF_1 ?
      goto ctrl_disp_menu     ;Si, ir a control de dispositivos.
   movlw 0x02
   xorwf ax, W
   btfsc STATUS, Z            ;ax==DTMF_2 ?
      goto disp_stat_menu     ;Si, ver estado de dispositivos.
   movlw 0x03
   xorwf ax, W
   btfsc STATUS, Z            ;ax==DTMF_3 ?
      goto bus_dce_menu       ;Si, ir a BUS DCE.
   movlw 0x04
   xorwf ax, W
   btfsc STATUS, Z            ;ax==DTMF_4 ?
      goto conf_ring_menu     ;Si, ir menu de configurar RINGS.
   movlw 0x05
   xorwf ax, W
   btfsc STATUS, Z            ;ax==DTMF_5 ?
      goto chpassw_menu       ;Si, ir a menu de cambio de clave.
   movlw DTMF_AST
   xorwf ax, W
   btfsc STATUS, Z            ;ax==DTMF_* ?
      goto uct_main_menu      ;Si, repetir main menu.
   movlw DTMF_NUM
   xorwf ax, W
   btfsc STATUS, Z            ;ax==DTMF_# ?
      call close_call         ;Si, desconectar UCT.

   call play_incorrecto       ;"NUMERO INCORRECTO"
   goto uct_main_menu         ;Volver a reproducir el menu principal.

;*********************************************************************
;* skip_menu_msg: des/activar UCT_OPTION/SKIPMENU.                   *
;*********************************************************************
skip_menu_msg:
      btfsc UCT_OPTION, SKIPMENU    ;SKIPMENU==0? Desactivado?
         goto skipmenu_off          ;No, desactivar.
      bsf UCT_OPTION, SKIPMENU      ;Si, activar.
      goto uct_main_menu
skipmenu_off:
      bcf UCT_OPTION, SKIPMENU      ;Desactivar.
      goto uct_main_menu

;*********************************************************************
;* ctrl_disp_menu: menu de control de dispositivos ON/OFF.           *
;*********************************************************************
ctrl_disp_menu:
      btfsc UCT_OPTION, SKIPMENU    ;SKIPMENU==1?
         goto get_disp              ;Si, omitir descripcion del menu.
      movlw ISD_INGRESAR
      call play_isd                 ;"INGRESAR"
      movlw ISD_DISPOSITIVO
```

```
      call play_isd                ;"DISPOSITIVO"

get_disp:                          ;Esperamos numero de dispositivo.
      movlw S60
      call getdtmf
      movwf ax                     ;ax=Numero de Dispositivo.

      movf ax, F
      btfsc STATUS, Z              ;ax==0 ?
         goto wrong_disp           ;Si, dispositivo incorrecto.
      movlw 5
      subwf ax, W                  ;w=ax-5
      btfss STATUS, C              ;ax < 5 ?
         goto on_off_disp          ;Si, des/activar dispositivo.
      movlw DTMF_AST
      xorwf ax, W
      btfsc STATUS, Z              ;ax==DTMF_* ?
         goto ctrl_disp_menu       ;Si, repetir menu.
      movlw DTMF_NUM
      xorwf ax, W
      btfsc STATUS, Z              ;ax==DTMF_# ?
         goto uct_main_menu        ;Si, volver a menu principal.

wrong_disp:
      call play_incorrecto         ;"NUMERO INCORRECTO"
      goto ctrl_disp_menu          ;Repetir menu.

on_off_disp:
      movlw S60
      call getdtmf                 ;Esperamos accion a tomar,
      movwf bx                     ;activar o desactivar dispositivo?

      movlw 0x01
      xorwf bx, W
      btfsc STATUS, Z              ;bx==DTMF_1 ?
         goto on_disp              ;Si, activar dispositivo.
      movf bx, F
      btfsc STATUS, Z              ;bx==DTMF_0 ?
         goto off_disp             ;Si, desactivar dispositivo.
      movlw DTMF_NUM
      xorwf bx, W
      btfsc STATUS, Z              ;bx==DTMF_# ?
         goto ctrl_disp_menu       ;Si, volver a pedir dispositivo.
      movlw DTMF_AST
      xorwf bx, W
      btfsc STATUS, Z              ;bx==DTMF_* ? (No tiene sentido repetir)
         goto ctrl_disp_menu       ;Si, volver a pedir dispositivo.

      call play_incorrecto         ;"NUMERO INCORRECTO"
      goto on_off_disp             ;Pedir de vuelta accion.

on_disp:
         bsf ax, 7
         movf ax, W                ;w=1000xxxx
         call ctrl_disp            ;activar dispositivo.
         movlw ISD_DISPOSITIVO
         call play_isd             ;"DISPOSITIVO"
```

```
         movlw ISD_ACTIVADO
         call play_isd           ;"ACTIVADO"
         goto ctrl_disp_menu     ;Repetir Menu.

off_disp:
         movf ax, W              ;w=0000xxxx
         call ctrl_disp          ;desactivar dispositivo.
         movlw ISD_DISPOSITIVO
         call play_isd           ;"DISPOSITIVO"
         movlw ISD_DESACTIVADO
         call play_isd           ;"DESACTIVADO"
         goto ctrl_disp_menu     ;Repetir Menu.

;*********************************************************************
;* disp_stat_menu: muestra estado de los dispositivos ON/OFF         *
;* conectados.                                                       *
;*********************************************************************
disp_stat_menu:
      movlw EE_DISPSTAT
      call eeprom_read
      movwf ax                   ;ax=estado de dispositivos.

      movlw ISD_DISPOSITIVO
      call play_isd              ;"DISPOSITIVO"

      movlw ISD_UNO
      call play_isd              ;"UNO"
      btfss ax, 0                ;Disp. 1 activado ?
      goto d1_off                ;No.
         movlw ISD_ACTIVADO      ;Si.
         call play_isd           ;"ACTIVADO"
         goto d1_end
d1_off:
         movlw ISD_DESACTIVADO
         call play_isd           ;"DESACTIVADO"
d1_end:

      movlw ISD_DOS
      call play_isd              ;"DOS"
      btfss ax, 1                ;Disp. 2 activado?
      goto d2_off                ;No.
         movlw ISD_ACTIVADO      ;Si.
         call play_isd           ;"ACTIVADO"
         goto d2_end
d2_off:
         movlw ISD_DESACTIVADO
         call play_isd           ;"DESACTIVADO"
d2_end:

      movlw ISD_TRES
      call play_isd              ;"TRES"
      btfss ax, 2                ;Disp. 3 activado?
      goto d3_off                ;No.
         movlw ISD_ACTIVADO      ;Si.
         call play_isd           ;"ACTIVADO"
         goto d3_end
d3_off:
```

```
         movlw ISD_DESACTIVADO
         call play_isd           ;"DESACTIVADO"
d3_end:

      movlw ISD_CUATRO
      call play_isd              ;"CUATRO"
      btfss ax, 3                ;Disp. 4 activado?
      goto d4_off                ;No.
         movlw ISD_ACTIVADO      ;Si.
         call play_isd           ;"ACTIVADO"
         goto d4_end
d4_off:
         movlw ISD_DESACTIVADO
         call play_isd           ;"DESACTIVADO"
d4_end:
      goto uct_main_menu         ;Volver a menu Princial.

;*********************************************************************
;* bus_dce_menu: Menu para interactuar con un dispositivo conectado  *
;* al bus dce.                                                       *
;* Se intentara Conectar con un dispositivo (DCE) usando el          *
;* BUS DCE mediante el protocolo CTP .                               *
;*                                                                   *
;* NOTA:                                                             *
;* 1) EL UCT SE PONE COMO SERVIDOR A LA ESPERA DE COMANDOS           *
;*    Y PASARA EL CONTROL TOTAL AL DCE. POR LO TANTO EL DCE DECIDIRA *
;*    SI RETORNA EN ESTE MENU, RESETEA, CORTA LLAMADA, ETC, EN EL    *
;*    UCT.                                                           *
;*    USARSE CON CUIDADO!. VERIFICAR FUNCIONAMIENTO DEL DCE.         *
;*                                                                   *
;* 2) En caso de conexion fallida, se retorna al menu principal      *
;*    con previo aviso del error.                                    *
;*********************************************************************
bus_dce_menu:

      call uct_ctpd          ;Interrumpir al DCE y esperar conexion.
      xorlw OK               ;El DCE retorno luego de una conexion
      btfsc STATUS, Z        ;exitosa?.
         goto uct_main_menu  ;Si, Volver al menu Princial!.
      movlw ISD_ERROR        ;No, ERROR en conexion.
      call play_isd          ;Reproducir mensaje de error y volver.
      movlw ISD_CONECTAR     ;DEMONIOS!
      call play_isd
         goto uct_main_menu  ;Volver al menu Princial. :(.

;*********************************************************************
;* conf_ring_menu: Menu para configurar el numero de rings que       *
;* deben pasar antes del que UCT conteste un llamado.                *
;* Si el usuario ingresa DTMF_0, EE_RINGS se establece a _LOT_RINGS. *
;*********************************************************************
conf_ring_menu:
      btfsc UCT_OPTION, SKIPMENU    ;SKIPMENU==1?
         goto get_rings             ;Si, omitir descripcion del menu.
      movlw ISD_INGRESAR
      call play_isd                 ;"INGRESAR"
      movlw ISD_RINGS
      call play_isd                 ;"RINGS"
```

```
get_rings:                    ;Esperamos cantidad de RINGS.
      movlw S60
      call getdtmf
      movwf ax                ;ax=Numero de RINGS.

      movf ax, F
      btfsc STATUS, Z         ;ax == 0?
         goto set_LOT_RINGS   ;Si, setar EE_RINGS = _LOT_RINGS
      movlw DTMF_AST
      xorwf ax, W
      btfsc STATUS, Z         ;ax = DTMF_*?
         goto  conf_ring_menu ;Si, Repetir Menu.
      movlw DTMF_NUM
      xorwf ax, W
      btfsc STATUS, Z         ;ax = DTMF_#?
         goto uct_main_menu   ;Si, Volver a menu Princial.

      goto set_EE_RINGS       ;No, Setear EE_RINGS = ax.

set_LOT_RINGS:
      movlw _LOT_RINGS
      movwf ax                ;ax = _LOT_RINGS

set_EE_RINGS:
      movlw EE_RINGS
      movwf ex
      movf ax, W
      call eeprom_write       ;EE_RINGS = ax

      movlw ISD_LISTO
      call play_isd           ;"LISTO"

      goto uct_main_menu      ;Volver a menu Princial.

;*********************************************************************
;* chpassw_menu: Menu para cambiar clave de acceso en el UCT.        *
;* Por seguridad, primero debe ingresar DTMF UNO, y luego la         *
;* nueva clave.                                                      *
;*********************************************************************

chpassw_menu:
      btfsc UCT_OPTION, SKIPMENU      ;SKIPMENU==1?
         goto get_chpassw_menu_option ;Si, omitir descripcion.
      movlw ISD_UNO
      call play_isd                ;"UNO"
      movlw ISD_CAMBIAR
      call play_isd                ;"CAMBIAR"
      movlw ISD_CLAVE
      call play_isd                ;"CLAVE"
      call play_sal_rep            ;"# SALIR, * REPETIR"


get_chpassw_menu_option:      ;Esperamos opcion.
      movlw S60
      call getdtmf
      movwf ax                ;ax=opcion.
```

```
      movlw 0x01
      xorwf ax, W
      btfsc STATUS, Z         ;ax == 1?
         goto set_passwd      ;Si, setear nuevo password.
      movlw DTMF_AST
      xorwf ax, W
      btfsc STATUS, Z         ;ax = DTMF_*?
         goto chpassw_menu    ;Si, Repetir Menu.
      movlw DTMF_NUM
      xorwf ax, W
      btfsc STATUS, Z         ;ax = DTMF_#?
         goto uct_main_menu   ;Si, Volver a menu Princial.

      call play_incorrecto
      goto chpassw_menu       ;Volver a repetir Menu.


set_passwd:
      movlw ISD_INGRESAR
      call play_isd           ;"INGRESAR"
      movlw ISD_CLAVE
      call play_isd           ;"CLAVE"

      movlw 0x04              ;Tomar 4 digitos DTMF y almacenar en
      call get_dtmfstr        ;AX:BX.

      movlw EE_PASSW_0        ;EE_PASSWD_0 = Digito 3 y 4
      movwf ex
      movf ax, W
      call eeprom_write       ;EE_PASSWD_0 = ax

      movlw EE_PASSW_1        ;EE_PASSWD_1 = Digito 1 y 2
      movwf ex
      movf bx, W
      call eeprom_write       ;EE_PASSWD_1 = bx

      movlw ISD_LISTO
      call play_isd           ;"LISTO"

      goto uct_main_menu      ;Volver a menu Princial.

;*********************************************************************
;*                        RUTINAS DEL BUS DCE DEL UCT                *
;*********************************************************************

;*********************************************************************
;* RUTINAS DEL UCT PARA COMUNICARSE POR EL BUS DCE                   *
;* MODO: CONECTADO.                                                  *
;* PROTOCOLO: CTP (Command Transmission Protocol)                    *
;*********************************************************************

;*********************************************************************
;* dce_connection(): establece conexion con DCE al producirce una    *
;* interrupcion por el BUS DCE.                                      *
;*********************************************************************
dce_connection:
```

```
   clrwdt
   bcf INTCON, INTE        ;No permitir Interrupciones externas.

   ;NOTA: EL UCT PASARA EL CONTROL TOTAL AL DCE, Luego se Resetea.
   call uct_ctpd           ;Poner al UCT como servidor de comandos.
   goto RESET              ;Resetear al finalizar la conexion.


;*********************************************************************
;* uct_ctpd(): pone al UCT como servidor de comandos, para la        *
;* recepcion y transmision de datos, utilizamos el protocolo CTP.    *
;* Los comandos recibidos de un DCE, son ejecutados en el UCT.       *
;* Los tiempos de timeout son definidos en el CTP, y cuando el UCT   *
;* o el DCE cierra una conexion o la conexion se pierde, la funcion  *
;* vuelve a su llamante (return).                                    *
;* O:                                                                *
;*    W = ERR -> Si se produjo un TIMEOUT o hubo en error en la      *
;*        conexion.                                                  *
;*    W = OK -> El DCE cerro la conexion correctamente.              *
;*                                                                   *
;*    CTPSTAT, es modificado en sus bits tambien.                    *
;* Notas:                                                            *
;* 1) Al esperar para recibir, o enviar una trama de bytes, se pone  *
;*    el DCEINT_PIN=0, lo cual es util para saber desde el UCT, si   *
;*    el DCE quiere recibir comandos, ya que debe responder con HELO.*
;* 2) Esta funcion de acuerdo a los comandos que el DCE envie al UCT *
;*    podra destruir o modificar cualquier tipo de variable.         *
;*********************************************************************
uct_ctpd:

   clrf CTPSTAT                 ;Limpiamos Registro de Estado.

chk_HELO:                       ;Comprobar Trama HELO.
   call ctp_rcv                 ;Esperamos recibir una TRAMA.
   xorlw ERR                    ;
   btfsc STATUS, Z              ;W==ERR?? Hubo un timeout?
      goto uct_ctpd_err         ;Si, salir con ERR.
   movlw HELOLSB                ;No, comprobar Trama HELO.
   xorwf B0, W
   btfss STATUS, Z              ;B0==HELOLSB?
      goto uct_ctpd_err         ;No, salir con ERR.
   movlw HELOMSB
   xorwf B1, W
   btfss STATUS, Z              ;B1==HELOMSB?
      goto uct_ctpd_err         ;No, salir con ERR.

snd_HELO:                       ;Enviar HELO al DCE.
   movlw TRDELAY                ;Esperamos que receptor Este Listo,
   call mpause                  ;para recibir.
   movlw HELOLSB
   movwf B0
   movlw HELOMSB
   movwf B1
   call ctp_snd                 ;Enviar B0:B1:B2=HELOLSB:HELOMSB:0xXX

chk_COMMAND                     ;Comprobar Trama COMMAND.
```

```
   call ctp_rcv                  ;Esperamos recibir una TRAMA.
   xorlw ERR                     ;
   btfsc STATUS, Z               ;W==ERR?? Hubo un timeout?
      goto uct_ctpd_err          ;Si, salir con ERR.

determine_COMMAND:               ;Determinamos y ejecutamos comando.

   movlw UCT_RESET               ;B0=CMD
   xorwf B0, W
   btfsc STATUS, Z               ;B0==UCT_RESET?
      goto _uct_reset            ;Si, Resetear UCT.

   movlw END_CTP
   xorwf B0, W
   btfsc STATUS, Z               ;B0==END_CTP?
      goto _end_ctp              ;Si, salir de funcion.

   movlw GET_PHONE_LINE
   xorwf B0, W
   btfsc STATUS, Z               ;B0==GET_PHONE_LINE?
      goto _get_phone_line       ;Si, ejecutar comando.

   movlw FREE_PHONE_LINE
   xorwf B0, W
   btfsc STATUS, Z               ;B0==FREE_PHONE_LINE?
      goto _free_phone_line      ;Si, ejecutar comando.

   movlw UCT_PAUSE
   xorwf B0, W
   btfsc STATUS, Z               ;B0==UCT_PAUSE?
      goto _uct_pause            ;Si, ejecutar comando.

   movlw PLAYISD
   xorwf B0, W
   btfsc STATUS, Z               ;B0==PLAYISD?
      goto _playisd              ;Si, ejecutar comando.

   movlw GETDTMF
   xorwf B0, W
   btfsc STATUS, Z               ;B0==GETDTMF?
      goto _getdtmf              ;Si, ejecutar comando.

   movlw GETDTMFSTR
   xorwf B0, W
   btfsc STATUS, Z               ;B0==GETDTMFSTR?
      goto _getdtmfstr           ;Si, ejecutar comando.

   movlw CTRLDISP
   xorwf B0, W
   btfsc STATUS, Z               ;B0==CTRLDISP?
      goto _ctrldisp             ;Si, ejecutar comando.

   movlw WR_EEPROM
   xorwf B0, W
   btfsc STATUS, Z               ;B0==WR_EEPROM?
      goto _wr_eeprom            ;Si, ejecutar comando.
```

```
   movlw RD_EEPROM
   xorwf B0, W
   btfsc STATUS, Z               ;B0==RD_EEPROM?
      goto _rd_eeprom            ;Si, ejecutar comando.

   movlw RD_RAM
   xorwf B0, W
   btfsc STATUS, Z               ;B0==RD_RAM?
      goto _rd_ram               ;Si, ejecutar comando.

   movlw WR_RAM
   xorwf B0, W
   btfsc STATUS, Z               ;B0==WR_RAM?
      goto _wr_ram               ;Si, ejecutar comando.

   movlw DIALP_NUMBER
   xorwf B0, W
   btfsc STATUS, Z               ;B0==DIALP_NUMBER?
      goto _dialp_number         ;Si, ejecutar comando.

   movlw GOTO_MAIN_MENU
   xorwf B0, W
   btfsc STATUS, Z               ;B0==GOTO_MAIN_MENU?
      goto _goto_main_menu       ;Si, ejecutar comando.

   movlw PING
   xorwf B0, W
   btfsc STATUS, Z               ;B0==PING?
      goto snd_FINISHED          ;Si, solo devolver PING en B0.

   goto _cmd_not_found           ;NO hubo coincidencias, salir con ERR.


snd_FINISHED:                    ;Enviar trama FINISHED al DCE.
   movlw TRDELAY                 ;Esperamos que receptor Este Listo,
   call mpause                   ;para recibir.
   call ctp_snd                  ;Enviar resultado de comando.

   btfsc CTPSTAT, CMD_NOT_FOUND  ;Comando Recibido inexistente?
      goto uct_ctpd_err          ;Si, salir con error.
   btfsc CTPSTAT, RESET_RCV      ;Comando UCT_RESET Recibido?
      goto RESET                 ;Si, resetear UCT.
   btfsc CTPSTAT, END_RCV        ;Comando END_CTP Recibido?
      goto uct_ctpd_ok           ;Si, salir de funcion.
   btfsc CTPSTAT, GM_MENU_RCV    ;Comando GOTO_MAIN_MENU Recibido?
      goto uct_main_menu_start   ;Si, pasar control a UCT.
   goto chk_HELO                 ;No, Esperar proximo comando.

uct_ctpd_err:
   bsf CTPSTAT, CTP_ERR          ;CTP_ERR=1 -> Implica ERROR.
   retlw ERR                     ;Retorno con W=ERR, :(.
uct_ctpd_ok:
   retlw OK                      ;Retorno con W=OK, yeah!.
```

```
;-----------------------------------------------------------------
;COMANDOS EJECUTADOS POR: uct_ctpd()
;Argumento en: B1, B2.
;Lugar de retorno: snd_FINISHED .
;Resultado depositado en: B1, B2.
;-----------------------------------------------------------------

;_uct_reset: El UCT se resetea y vuelve a modo espera.
;ARG: Ninguno.
;RET: Ninguno.
_uct_reset:
   bsf CTPSTAT, RESET_RCV        ;RESET_RCV=1.
   goto snd_FINISHED

;_end_ctp: El UCT termina la comunicacion con DCE.
;ARG: Ninguno.
;RET: Ninguno.
_end_ctp:
   bsf CTPSTAT, END_RCV          ;END_RCV=1
   goto snd_FINISHED

;_get_phone_line: Ocupa linea linea telefonica.
;ARG: Ninguno.
;RET: Ninguno.
_get_phone_line:
   bsf PORTB, 3      ;RB3=1.
   goto snd_FINISHED

;_free_phone_line: Desocupa/Libera la linea telefonica.
;ARG: Ninguno.
;RET: Ninguno.
_free_phone_line:
   bcf PORTB, 3      ;RB3=0.
   goto snd_FINISHED

;_uct_pause: Establece una pausa en el UCT de B1 segundos.
;ARG: B1 = segundos.
;RET: Ninguno.
_uct_pause:
   movf B1, W
   call spause       ;PAUSA! STOP IT!
   goto snd_FINISHED

;_playisd: Reproduce el mensaje numero B2 de la ISD1420.
;          Si un codigo DTMF es recibido, lo toma.
;ARG: B1 = Segundos para getdtmf()
;     B2 = Num. de msj. a reproducir.
;RET: B1 = Si llego dmtf, retorno es segun funcion getdtmf().
;     B2 = Num. de msj. reproducido o 0x00 si llego dtmf.
_playisd:
   movf B2, W           ;Reproducir mensaje numero B2.
   call play_isd        ;Reproducir!.
   btfss PORTA, 4       ;Llego un codigo DMTF???
      goto snd_FINISHED ;No, retornar y B2 != 0x00.
   clrf B2
   goto _getdtmf        ;Tomar DTMF segun _getdmtf.
```

```
;_getdtmf: Espera B1 segundos por un codigo DTMF.
;ARG: B1 = segundos de espera antes de retornar.
;RET: B1 = segun funcion getdtmf().
_getdtmf:
   bcf UCT_OPTION, QTIMEOUT
   movf B1, W
   call getdtmf
   movwf B1
   bsf UCT_OPTION, QTIMEOUT
   goto snd_FINISHED

;_getdtmfstr: Espera 1 Minuto, por B1 codigos DTMF.
;ARG: B1 = numeros de codigos o digitos DTMF a esperar (max. 4).
;RET: B1 = DTMF recibido. (Nible superior e inferior)
;     B2 = DTMF recibido. (Nible superior e inferior)
;NOTA: Si B1 < 4, los DTMF faltantes en B2:B1, se completan con cero.
;      En el nible inferior de B1 se deposita el ultimo DTMF recibido.
_getdtmfstr:
   bcf UCT_OPTION, QTIMEOUT
   movf B1, W                ;Numero de DTMF a esperar.
   call get_dtmfstr          ;Retorno en ax:bx. Ver funcion para mas info.
   movf bx, W
   movwf B1                  ;B1 = BX
   movf ax, W
   movwf B2                  ;B2 = AX
   bsf UCT_OPTION, QTIMEOUT
   goto snd_FINISHED

;_ctrldisp: Controla dispositivos ON/OFF.
;ARG: B1 = segun funcion ctrl_disp().
;RET: Ninguno.
_ctrldisp:
   movf B1, W
   call ctrl_disp
   goto snd_FINISHED

;_wr_eeprom: Escribe en la memoria de datos EEPROM del UCT.
;ARG: B1 = Dato a escribir.
;     B2 = Direccion a escribir.
;RET: Ninguno.
_wr_eeprom:
   movf B2, W
   movwf ex
   movf B1, W
   call eeprom_write    ;Write EEPROM!.
   goto snd_FINISHED

;_rd_eeprom: Lee la memoria de datos EEPROM del UCT.
;ARG: B1 = Direccion a leer.
;RET: B1 = Valor leido.
_rd_eeprom:
   movf B1, W
   call eeprom_read     ;Read EEPROM!.
   movwf B1
   goto snd_FINISHED
```

```
;_rd_ram: Lee la memoria de datos RAM del UCT. (BANK0).
;ARG: B1 = Direccion a leer.
;RET: B1 = Valor leido.
_rd_ram:
   movf B1, W           ;Initialize pointer
   movwf FSR            ;to RAM.
   movf INDF, W         ;W = INDF = *FSR = *B1 (READ RAM)
   movwf B1             ;B1 = *B1
   goto snd_FINISHED

;_wr_ram: Escribe en la memoria de datos RAM del UCT. (BANK0).
;ARG: B1 = Dato a escribir.
;     B2 = Direccion a escribir.
;RET: Ninguno
;
;NOTA: Usar con cuidado! No intentar cambiar de banco!.
_wr_ram:
   movf B2, W           ;Initialize pointer
   movwf FSR            ;to RAM.

   movf B1, W           ;W = B1 = Dato..
   movwf INDF           ;INDF = *FSR = *B2 = B1 = W (WRITE RAM)
   goto snd_FINISHED


;_dialp_number: El UCT marca el numero B1 por Pulsos.
;ARG: B1 = Numero a marcar
;RET: Ninguno. Ver funcion dialp() para mas info. No se comprueban errores.
_dialp_number:
   movf B1, W
   call dialp          ;Dial Number!.
   goto snd_FINISHED

;_goto_main_menu: El UCT toma el control y repoduce el menu principal.
;ARG: Ninguno.
;RET: Ninguno.
_goto_main_menu:
   bsf UCT_OPTION, QMSG       ;QMSG=1, Desconectar con mensage de aviso.
   bsf CTPSTAT, GM_MENU_RCV   ;GM_MENU_RCV=1
   goto snd_FINISHED

;_cmd_not_found: El Comando recibido no existe!.
;Nota: El UCT terminara la conexion.
;ARG: Ninguno.
;RET: Ninguno.
_cmd_not_found
   bsf CTPSTAT, CMD_NOT_FOUND    ;CMD_NOT_FOUND=1
   goto snd_FINISHED
```

```
;*********************************************************************
;* ctp_rcv(): recibe una trama de 3 bytes y lo deposita en  B0,B1,B2.*
;* O:                                                                *
;*    W = ERR -> TIMEOUT, los 3 bytes no se recibieron en RXWAIT seg.*
;*    W = OK -> los 3 bytes se recibieron antes de RXWAIT segundos.  *
;*    B0 = BYTE 0 De trama.                                          *
;*    B1 = BYTE 1 De trama.                                          *
;*    B2 = BYTE 2 De trama.                                          *
;*********************************************************************
ctp_rcv:

   call open_bdce             ;Habilitamos Tx,Rx, serie.

   bcf UCT_OPTION, QTIMEOUT   ;No RESET al producirse TIMEOUT.
   movlw RXWAIT               ;Tiempo de espera para recibir trama.
   call set_timeout           ;TIMEOUT en RXWAIT segundos.

   call srcv                  ;Esperar BYTES.
   movwf B0
   call srcv
   movwf B1
   call srcv
   movwf B2

   ;Comprobamos si hubo TIMEOUT.
   btfsc UCT_STATUS, TIMEOUT  ;TIMEOUT==1?
      movlw ERR               ;Si, W=ERR. ERROR en recepcion.
   btfss UCT_STATUS, TIMEOUT  ;TIMEOUT==0?
      movlw OK                ;Si, W=OK. Trama recibida!

   call unset_timeout         ;Sacamos el TIMEOUT.
   bsf UCT_OPTION, QTIMEOUT   ;QTIMEOUT=1 por convension.

   call close_bdce            ;Inhabilitamos Tx,Rx serie.

   return

;*********************************************************************
;* ctp_snd(): envia una trama de 3 bytes, con los valores de         *
;* B0,B1,B2.                                                         *
;*                                                                   *
;* Notas:                                                            *
;* 1) Orden de envio:                                                *
;*    B0 = BYTE 0 De trama.                                          *
;*    B1 = BYTE 1 De trama.                                          *
;*    B2 = BYTE 2 De trama.                                          *
;*********************************************************************
ctp_snd:

   call open_bdce             ;Habilitamos Tx,Rx, serie.

   movf B0, W
   call ssend
   movf B1, W
   call ssend
   movf B2, W
   call ssend
```

```
   call close_bdce            ;Inhabilitamos Tx,Rx serie.
   return                     ;Back to Caller!

;*********************************************************************
;* Rutinas de transmision/recepcion serie, asincrona del UCT         *
;*********************************************************************

;*********************************************************************
;* open_bdce(): configura y  habilita al BUS DCE para la transmision *
;  o recepcion serie.                                                *
;*********************************************************************
open_bdce:
   clrf PORTA          ;Ningun dispositivo seleccionado en RB<7:4>.

   BANK_1              ;TRISB=1110xxxx -> RB5-7=E, RB4=S.
   movlw 0xF0          ;RB4=TXPIN=S, RB5=RXPIN=E
   iorwf TRISB, F      ;TRISB=1111xxxx=EEEExxxx
   bcf TRISB, 4        ;TRISB=1110xxxx=EEESxxxx
   BANK_0

   bsf TXPIN           ;Mantener TXPIN=1.

   bsf PORTA, 3        ;Habilitar 4016 (sacar de alta Z).
   bcf DCEINT_PIN      ;Ocupamos BUS DCE. -> DCEINT_PIN =0.

   return

;*********************************************************************
;* close_bdce(): inhabilita al BUS DCE para la transmision           *
;  o recepcion serie.                                                *
;*********************************************************************
close_bdce:

   bsf DCEINT_PIN       ;Desocupamos BUS DCE. -> DCEINT_PIN=1.
   bcf PORTA, 3         ;Deshabilitar 4016 (poner en alta Z).

   return
```

```
;*********************************************************************
;* srcv(): Espera hasta recibir un BYTE en forma serie               *
;* a traves de RXPIN, luego deposita el BYTE en W y en RXTX_BUF.     *
;* Si, UCT_STATUS/TIMEOUT==1, retorna imediatamente y el valor       *
;* en W y en RXTX_BUF sera indefinido.                               *
;* O:                                                                *
;*    W=RXTX_BUF = BYTE recibido.                                    *
;* Notas:                                                            *
;* 1) La recepcion es de forma asincrona y los parametros son        *
;*    Velocidad: 1200, 8N1.                                          *
;* 2) Esta funcion depende de la frecuencia de clock del PIC.        *
;*    Fue pensada para funcionar con un cristal de 4 MHZ.            *
;* 3) Para usar esta funcion, antes debe usarse open_bdce().         *
;* 4) Si antes de esta funcion se usa set_timeout(), la funcion      *
;*    esperara un tiempo determinado por el BYTE. Ello se puede usar *
;*    para comprobar si byte recibido es valido, ya que TIMEOUT debe *
;*    ser 0.                                                         *
;*********************************************************************
srcv:

   clrf RXTX_BUF        ;Limpiamos el buffer de recepcion.
   movlw BITS           ;Numero de bits de datos a recibir.
   movwf BCOUNTER       ;BCOUNTER=BITS.

chk_start_bit:
   clrwdt
   btfsc UCT_STATUS, TIMEOUT  ;Se produjo un TIMEOUT==1?
      goto end_srcv           ;Si, Retornar.
   btfsc RXPIN                ;Se recibio el 'start bit'?, RXPIN==0?
      goto chk_start_bit      ;No, seguir esperando.

start_reception:
   delay_HalfBit        ;Esperar hasta la mitad del 'start bit'.
   delay_FullBit        ;Ignorar 'start bit' y samplear el primer bit
                        ;de datos en su mitad.

;Recibir los bits de datos.
receive_bit:
   clrwdt
   btfsc RXPIN          ;RXPIN==1?
   bsf STATUS,C         ;Si -> set carry bit to 1.
   btfss RXPIN          ;RXPIN==0?
   bcf STATUS,C         ;Si -> set carry bit to 0.

   rrf RXTX_BUF, F      ;Poner bit en RXTX_BUF (Rotar buffer de recepcion).
   delay_FullBit        ;Esperar proximo bit de datos.
   decfsz BCOUNTER, F   ;(--BCOUNTER)==0?
   goto receive_bit     ;No, BCOUNTER!=0 -> recibir proximo bit.

   movf RXTX_BUF, W     ;W=TXRXB_BUF

end_srcv:
   return               ;Back to caller!.
```

```
;*********************************************************************
;* ssend(): transmite un BYTE en forma serie a traves                *
;* de TXPIN.                                                         *
;* O:                                                                *
;*    W= Byte a transmitir.                                          *
;* Notas:                                                            *
;* 1) La transmision es de forma asincrona y los parametros son      *
;*    Velocidad: 1200, 8N1.                                          *
;* 2) Esta funcion depende de la frecuencia de clock del PIC.        *
;*    Fue pensada para funcionar con un cristal de 4 MHZ.            *
;* 3) Para usar esta funcion, antes debe usarse open_bdce().         *
;*********************************************************************
ssend:

   movwf RXTX_BUF       ; RXTX_BUF=W. BYTE a trabsmitir.

   movlw BITS           ; Numero de bits a transmitir.
   movwf BCOUNTER       ; BCOUNTER=BITS.

   bcf TXPIN            ; Generar 'start bit'
   delay_FullBit        ; Generar un retardo de un bit.

transmit_bit:
   clrwdt
   rrf RXTX_BUF, F      ; STATUS/C = Bit a transmitir (rotar RXTX_BUF)
   btfsc STATUS, C      ; (STATUS/C)==1?
   bsf TXPIN            ; Si, -> TXPIN=1.
   btfss STATUS, C      ; (STATUS/C)==0?
   bcf TXPIN            ; Si, -> TXPIN=0.
   delay_FullBit        ; Generar un retardo de un bit.
   decfsz BCOUNTER, F   ; (--BCOUNTER)==0?
      goto transmit_bit ; No, transmitir proximo bit.
   bsf TXPIN            ; Si, generar 'stop bit'
   delay_FullBit        ; Generar un retardo de un bit.

   return               ; Back to caller!

;*********************************************************************
;* HalfBit_Pause(): establece una pausa en el programa de 416 uS o   *
;* medio Bit cuando la transmision serie tiene 1200 Bits por segundo.*
;*********************************************************************
HalfBit_Pause:

   movlw d'137'         ;(1 uS)
   movwf SDCNT          ;(1 uS)

dec_SDCNT:              ;Tiempo del loop = SDCNT*3uS -1 = 410 uS
   decfsz SDCNT, F
   goto dec_SDCNT

   ;Tiempo Total= 414 uS, y 416 uS cuando es llamada con un call.
   return               ;(2 uS)
```

```
;*********************************************************************
;*                     RUTINAS GENERALES DEL UCT                     *
;*********************************************************************

;*********************************************************************
;*dialp(): Marca el numero Telefonico W por medio de pulsos.         *
;*I: W = Numero a Marcar.                                            *
;*Destruye: ax.                                                      *
;*                                                                   *
;*Nota/Normas:                                                       *
;*  1) El marcado se efectua segun la los sig. Parametros:           *
;*     Frecuencia de los Pulsos: 10 Hz +/- 1 Hz                      *
;*     Relación de impulsos (apertura - cierre):                     *
;*     El tiempo de apertura estara comprendido entre un 57.5 % a un *
;*     70.6 % del total del ciclo.                                   *
;*                                                                   *
;*  2) Recomendaciones para el uso de esta funcion:                  *
;*     Pausa interdigital: 800 ms +20% / -10%.                       *
;*     Luego de ocupar la linea esperar 600 ms antes de iniciar      *
;*     el Marcado de algun numero, para evitar asi que el primer     *
;*     digito puede ser mal entendido por la central telefonica.     *
;*********************************************************************
dialp:

   movwf ax                     ;ax = W = Numero a Marcar.
   movlw d'10'                  ;W = 10.
   movf ax, F
   btfsc STATUS, Z              ;ax == 0?
      movwf ax                  ;Si, hacer ax = 10.

dialp_number:
   bcf PORTB, 3                 ;APERTURA DE LINEA
   movlw d'64'
   call mpause                  ;Mantener por 64 ms.
   bsf PORTB, 3                 ;CIERRE DE LINEA
   movlw d'36'
   call mpause                  ;Mantener por 36 ms.
   decfsz ax, F                 ;(--ax) == 0?
   goto dialp_number            ;No, seguir Marcando.

   return                       ;Back to caller!.

;*********************************************************************
;* close_call(): Desconecta al UCT de la llamada actual.             *
;*********************************************************************
close_call:

   btfss UCT_OPTION, QMSG  ;Reproducir mensages antes de salir?
      goto end_close_call  ;No, salir sigilosamente...shhh!
   movlw ISD_SALIR         ;Si, "SALIR".
   call play_isd           ;antes de desconectar.
   movlw S2
   call spause             ;Esperar 2 seg. antes de RESETEAR.

end_close_call:
   goto RESET              ;Good Nigth!
```

```
;*********************************************************************
;* get_dtmfstr(): Toma una cadena de 4 codigos DTMF como maximo,     *
;* usando la funcion getdtmf().                                      *
;* I: W = Numero de codigos DTMF a tomar (max. 4).                   *
;* O: AX y BX almacenan los 4 codigo DTMF recibidos. Donde el nibble *
;* menos significativo de BX contiene el ultimo codigo DTMF recibido.*
;* Si W < 4 los codigos DTMF faltantes en AX y BX se completan con   *
;* cero. TIMEOUT en 60 segundos si no se recive ningun codigo DTMF.  *
;* Destruye: W                                                       *
;* Registros stack-eados: ax, cx, dx, ex                             *
;*********************************************************************
get_dtmfstr:

   ;ax y ex se stakea en getdtmf.
   call push_cx
   call push_dx

   clrf ax
   clrf bx                  ;BX y AX Almacenan DMTF's recibido.
   movwf cx                 ;Numero de digitos DTMF a tomar.
for_get_dtmfstr:
   movlw S60                ;timeout en 60 segundos para getdtmf
   call getdtmf             ;W = codigo DTMF recibido. W=0000XXXX
   addwf bx, F              ;bx=DTMF
   decfsz cx, F             ;Tomar otro codigo DTMF?
   goto get_next_dtmfstr    ;Si!
   goto end_for_get_dtmfstr ;No! cx==0.

get_next_dtmfstr:        ;Preparo [ax:bx] para recibir proximo DTMF.
   movlw 0x04
   movwf dx              ;dx=4 numero de rotaciones.
rotate_4left:            ;Desplazo [ax:bx] 4 lugares a la izquierda.
      bcf STATUS, C      ;Limpio CARRY, en proxima rotacion meto C=0.
      rlf bx, F          ;Roto izq. bx y 7 bit se pone en CARRY.
      rlf ax, F          ;Roto izq. ax e introdusco CARRY anterior.
      decfsz dx, F
      goto rotate_4left
   goto for_get_dtmfstr  ;Busco proximo DMTF!

end_for_get_dtmfstr:     ;No hay mas DTMF que obtener.

   call pop_cx
   call pop_dx

   return               ;back to caller!

;*********************************************************************
;* play_incorrecto(): reproduce "NUMERO INCORRECTO".                 *
;* Destruye: W                                                       *
;*********************************************************************
play_incorrecto:
   movlw ISD_NUMERO
   call play_isd             ;"NUMERO"
   movlw ISD_INCORRECTO
   call play_isd             ;"INCORRECTO"
   return
```

```
;*********************************************************************
;* play_rep-sal(): reproduce "NUMERAL SALIR, ASTERISCO REPETIR".     *
;* Destruye: W                                                       *
;*********************************************************************
play_sal_rep:
   movlw ISD_NUMERAL
   call play_isd              ;"NUMERAL"
   movlw ISD_SALIR
   call play_isd              ;"SALIR"
   movlw ISD_ASTERISCO
   call play_isd              ;"ASTERISCO"
   movlw ISD_REPETIR
   call play_isd              ;"REPETIR"
   return

;*********************************************************************
;* mpause(): establece una pausa de W mili-segundos en el programa.  *
;* La pausa es de: [((1Mhz/8)/125)]^(-1) * W = 1 ms * W .            *
;* I: W = Multiplicador o factor .                                   *
;* Destruye: W, ex.                                                  *
;* Nota:                                                             *
;* 1) Desactiva interrupcion Por TMR0 y modifica OPTION_REG.         *
;* 2) No produce interrupciones.                                     *
;* 3) Si UCT_OPTION\QF_DTFM==1 y si hay un codigo DTMF presente la   *
;*    funcion retorna imediatamente.                                 *
;*********************************************************************
mpause:

   movwf ex                ;ex=W -> Multiplicador.

   BANK_1
   movlw b'11010000'       ;TOCS=0,PSA=0,PS2=0,PS1=1,PS0=0.
   andwf OPTION_REG, F     ;OPTION_REG=xx0x0000
   bsf OPTION_REG, PS1     ;OPTION_REG=xx0x0010

   BANK_0
   bcf INTCON, T0IE        ;T0IE=0 -> no interrupcion por TMR0.
   bcf INTCON, T0IF        ;TOIF=0 -> Ponemos Flag a cero.

wait_mpause:
   clrwdt
   movlw 0x83              ;W=neg(0x7D)=neg(d'125')=0x83
   movwf TMR0              ;Cargamos TMR0=W

tmr0_overflow:

   btfss UCT_OPTION, QF_DTMF ;Salir si hay un Codigo DTMF presente?
   goto no_dtmf_chk          ;No. Ignorar comprobacion de DTMF.
   btfsc PORTA, 4            ;Hay un codigo DTMF presente en el 8870 ?
   return                    ;Si. Salir de funcion.

no_dtmf_chk:
   btfss INTCON, T0IF      ;Desbordo el TMR0?
      goto tmr0_overflow   ;No, Seguir esperando.
   bcf INTCON, T0IF        ;Si, Reponemos flag.

   decfsz ex, F            ;ex == 0?
```

```
      goto wait_mpause     ;No, seguir esperando.
   return                  ;Si, volver.

;*****************************************************************
;* spause(): genera una pausa de W segundos en el programa.      *
;* I: W = Pausa de N segundos.                                   *
;*****************************************************************
spause:

   bcf UCT_OPTION, QTIMEOUT ;No resetear al producirse un TIMEOUT
   call set_timeout         ;Producir un TIMEOUT en W segundos.

wait_TIMEOUT:
   clrwdt
   btfss UCT_STATUS, TIMEOUT  ;Ocurrio un TIMEOUT==1?
   goto  wait_TIMEOUT         ;No, seguir esperando. (TIMEOUT==0).

end_spause:                   ;Si, volver. TIMEOUT==1.
   call unset_timeout         ;Desactivar interrupcion por TMR0.
   bsf UCT_OPTION, QTIMEOUT   ;QTIMEOUT siempre debe estar activada.
   return                     ;back to caller.

;*****************************************************************
;* set_timeout(): setea el temporizador para que use clock interno   *
;* del pic16f84a y produce un TIMEOUT en W segundos.                 *
;*                                                                   *
;* I: W = tiempo en segundos antes de producir el TIMEOUT.           *
;* Modifica: SECONDS, TMR0_AUX, TMR0 y OPTION_REG.                   *
;*                                                                   *
;* Notas:                                                            *
;* 1) Se debe usar unset_timeout() para desactivar esta funcion.     *
;* 2) GIE debe estar habilitado y para desabilitar esta funcion      *
;* se debe inhibir el flag de interrupcion por TMR0. Para ello usar: *
;* call unset_timeout -> desactiva un TIMEOUT seteado por            *
;* set_timeout().                                                    *
;*                                                                   *
;* Funcionamiento:                                                   *
;* La frecuencia del clock interno sera fi=4Mhz/4 = 1 Mhz .          *
;* Luego el prescaler divide por 32, fi/32= 31250 Hz. E incrementara *
;* al TMR0 cada 32 pulsos de clock.                                  *
;* El registro TMR0 se carga con TMR0=6, por lo tanto                *
;* se producira una interrupcion cada 250 incrementos de TMR0.       *
;* Esto es: (fi/32)/250 = 125 Hz.                                    *
;* Posteriormente se carga la variable TMR0_AUX con 125, y se la     *
;* decrementa cada 125 interrupciones del timer.                     *
;* Quedando: ((fi/32)/250)/125 = 1 HZ. Lo que equivale a 1 segundo.  *
;* Finalmente se decrementa la variable SECONDS y si resulta cero    *
;* activa el flag TIMEOUT de la variable UCT_STATUS.                 *
;* Luego si flag QTIMEOUT esta activada en UCT_OPTION el UCT se      *
;* reiniciara.                                                       *
;* El encargado de manegar cada interrupcion es la funcion RSITMR0 y *
;* decide que hacer con dicha interrupcion.                          *
;*****************************************************************
set_timeout:

   movwf SECONDS        ;Numero de segundos para que ocurra el TIMEOUT.
```

```
   BANK_1
   movlw b'11010000'    ;TOCS=0,PSA=0,PS2=0,PS1=1,PS0=0.
   andwf OPTION_REG, F  ;OPTION_REG=xx0x0000
   bsf OPTION_REG, PS2  ;OPTION_REG=xx0x0100
   BANK_0

   movlw d'125'         ;Decrementar SECONDS cada 125 interrupciones
   movwf TMR0_AUX       ;del timer 0.
   movlw 0x06           ;Producir una interrupcion cada 250 cuentas
   movwf TMR0           ;o incrementos del TMR0
   bsf INTCON, T0IE     ;habilitar interrupcion por TMR0
   return               ;back to caller!

;*********************************************************************
;* unset_timeout(): desactiva un TIMEOUT y asigna prescaler al       *
;* watchdog.                                                         *
;* Repone el flag TIMEOUT a cero.                                    *
;*********************************************************************
unset_timeout:
   bcf UCT_STATUS, TIMEOUT ;Nos aseguramos que TIMEOUT=0.
   bcf INTCON, T0IE        ;desactivar interrupcion por TMR0.
   BANK_1
   bsf OPTION_REG, PSA
   BANK_0
   return

;*********************************************************************
;                RUTINAS DE MANEJO DE CIRCUITOS DEL UCT              *
;*********************************************************************

;*********************************************************************
;* ctrl_disp(): activa o desactiva dispositivo externo.              *
;* I: W=S000xxxx                                                     *
;* Donde si S: 1 -> activa dispositivo 0 -> desactiva dispositivo.   *
;* y xxxx representa el numero de dispositivo a activar.             *
;* Destruye: cx, dx, ex,W                                            *
;* NOTA:                                                             *
;* Si el numero de dispositivo es distinto de [1,2,3,4], solo        *
;* actualiza el latch 4042, con el estado de los disp. almacenado en *
;* EEPROM. Util para inicializar dispositivos al comienzo del UCT.   *
;*********************************************************************
ctrl_disp:

   movwf ex       ;Dispositivo a activar.

   BANK_1
   ;Configuramos BUS de datos RB4-7 entre dispositivos.
   ;RB4-7=S -> conectados a latch 4042.
   movlw 0x0F
   andwf TRISB, F ;TRISB=0000XXXX

   BANK_0
   clrf cx        ;cx bit-0 = Tipo de operacion. 1=Activar/0=Desac.
   btfsc ex, 7    ;Activar disp. -> S=1?
      bsf cx, 0   ;Si activar dispositivo.

   bcf ex, 7      ;Borramos bit-7 de ex.
```

```
   movlw EE_DISPSTAT    ;EE_DISPSTAT=0000XXXX
   call eeprom_read
   movwf dx             ;Estado de dispositivos ON/OFF actual.

   ;Determinamos que dispositivo activar o desactivar.
   ;El nuevo estado de los disp. ON/OFF queda en dx.

;if_d4
   movlw 4
   xorwf ex, W
   btfss STATUS, Z            ; ex==4, Dispositivo 4 ?
   goto else_if_d3            ; No.
      btfss cx, 0             ; Activar?
      goto d4_turn_off        ; No.
         bsf dx, 3            ; D4 Activado.
         goto end_if_switch
d4_turn_off:
         bcf dx, 3            ; D4 Desactivado.
         goto end_if_switch

else_if_d3:
   movlw 3
   xorwf ex, W
   btfss STATUS, Z            ; ex==3, Dispositivo 3 ?
   goto else_if_d2            ; No.
      btfss cx, 0             ; Activar?
      goto d3_turn_off        ; No.
         bsf dx, 2            ; D3 Activado.
         goto end_if_switch
d3_turn_off:
         bcf dx, 2            ; D3 Desactivado.
         goto end_if_switch

else_if_d2:
   movlw 2
   xorwf ex, W
   btfss STATUS, Z            ; ex==2, Dispositivo 2 ?
   goto else_if_d1            ; No.
      btfss cx, 0             ; Activar?
      goto d2_turn_off        ; No.
         bsf dx, 1            ; D2 Activado.
         goto end_if_switch
d2_turn_off:
         bcf dx, 1            ; D2 Desactivado.
         goto end_if_switch

else_if_d1:
   movlw 1
   xorwf ex, W
   btfss STATUS, Z            ; ex==1, Dispositivo 1 ?
   goto end_if_switch         ; No, solo actualizar latch 4042.
      btfss cx, 0             ; Activar?
      goto d1_turn_off        ; No.
         bsf dx, 0            ; D1 Activado.
         goto end_if_switch
d1_turn_off:
```

```
         bcf dx, 0            ; D1 Desactivado.

end_if_switch:

   ;Grabamos estado de los dispositivos en el memoria EEPROM.
   ;Si el nuevo estado de disp. es igual al anterior, no
   ;grabamos en la mem. eeprom, pero si actualizamos latch 4042.
   movlw EE_DISPSTAT
   call eeprom_read
   xorwf dx, W
   btfsc STATUS, Z           ;dx == EE_DISPSTAT ?
   goto latch_update         ;Si. Solo actualizar latch.
      movlw EE_DISPSTAT      ;No. Grabar en eeprom y actualizar latch.
      movwf ex               ;Direccion eeprom y
      movf dx, W             ;Dato a grabar.
      call eeprom_write

latch_update:
   ;Ponemos estado de disp. en el BUS de datos RB4-7
   movlw 0x0F
   andwf PORTB, F             ; PORTB=0000xxxx
   swapf dx, F                ; dx=xxxx0000
   movf dx, W                 ; W=xxxx0000
   iorwf PORTB, F             ; PORTB[4-7]=dx

   ;Seleccionamos Latch 4042 para usar con BUS de datos RB4-7 y
   ;el estado de los dispositivos se transfieren al LATCH.
   bsf PORTA, 2      ;RA2=1 -> activa entradas de 4042.

   nop               ;Esperamos tiempo de propagacion del 4042
   nop               ;del orden de nano-segundos.

   ;Desactivamos Latch 4042 del Bus de datos RB4-7
   bcf PORTA, 2      ;RA2=0 -> desactiva entradas de 4042.

   movlw 0x0F
   andwf PORTB, F    ;PORTB=0000XXXX, Bus de datos a CERO.

end_ctrl_disp:
   return            ;back to caller!

;*********************************************************************
;* getdtmf(): Espera un codigo DTMF por un tiempo determinado        *
;* almacenado en W. Si el codigo dtmf no se recibe en el             *
;* tiempo especificado en W, produce una intertupcion                *
;* que es manejada por RSITMR0. De lo contrario devuelve             *
;* en W el codigo DTMF recibido.                                     *
;* Si QTIMEOUT=0, y el tiempo de espera expira la funcion regresa    *
;* W=0xFF, como indicacion que ningun dtmf fue recibido.             *
;* I: W = time-out en segundos.                                      *
;* O: W = codigo DTMF recibido. (DMTF 0 se codifica como W=0x00)     *
;* Registros stack-eados: ax, ex                                     *
;*********************************************************************
getdtmf:

   call push_ax      ; salvamos ax.
   call push_ex      ; salvamos ex.
```

```
   call set_timeout  ; seteamos un timeout en W segundos.

wait_dtmf:
      clrwdt
      btfsc PORTA, 4         ;Hay un codigo DTMF presente en el 8870 ?
         goto dtmf_recived       ;Si, tomar codigo.
      btfss UCT_STATUS, TIMEOUT  ;Se produjo un TIMEOUT==1?
         goto wait_dtmf          ;No, seguir esperando.
      call unset_timeout         ;Si, sacar timeout y retornar
      movlw 0xFF                 ;con W=0xFF.
      goto end_getdtmf           ;Salir de funcion.

dtmf_recived:
   ;Codigo DTMF recibido.
   call unset_timeout            ;No permitir timeout en W segundos.

   BANK_1
   ;Configuramos BUS de datos RB4-7 entre dispositivos como entradas.
   movlw 0xF0
   iorwf TRISB, F      ;RB4-7=Entradas -> conectado a Q1-4 de 8870.

   BANK_0
   ;Seleccionamos al decodificador 8870 para usar con BUS de datos.
   bsf PORTA, 0      ;RA0=1 -> saca de alta Z, Q1-Q4 de 8870.
   nop               ;2 uS para que Q1-Q4 salgan de alta impedancia.
   nop               ;(del orden de nano-segundos)

   ;Leemos codigo DTMF
   movf PORTB, W     ;En 8870 Leo Codigo DTMF de Q1-Q4
   andlw 0xF0        ;W=xxxx0000
   movwf ax          ;ax=W=xxxx0000
   swapf ax,F        ;ax=0000xxxx

   ;Quitamos 8870 del Bus de datos RB4-7
   bcf PORTA, 0      ;RA0=0 -> pone en alta impedancia Q1-Q4 de 8870.

   ;Dejamos Bus de datos RB4-7 conf. como salidas.
   BANK_1
   movlw 0x0F
   andwf TRISB, F    ;RB4-7=Salidas
   BANK_0            ;Dejamos banco cero (por convencion)
   movlw 0x0F
   andwf PORTB, F    ;PORTB=0000XXXX, Bus de datos a CERO.

   ;El 8870 codifica el DTMF 0 como 0x0A, si se recibe DTMF 0
   ;la funcion devolvera W=0. Para preservar naturalidad en numeros.
   movlw 0x0A
   xorwf ax, W
   btfsc STATUS, Z   ;ax == 0x0A ?
      clrf  ax       ;Si, hacer ax=0.

   movf ax, W        ;W = DTMF recibido.

   ;Esperamos que el usuario deje de enviar el mismo DTMF.
wait_dtmf_end:
   clrwdt
   btfsc PORTA, 4      ;El codigo DTMF todavia esta presente?
```

```
   goto wait_dtmf_end  ;Si, esperar a que finalize.

end_getdtmf:
   call pop_ax       ;reponemos viejo valor de ax.
   call pop_ex       ;reponemos viejo valor de ex.
   return            ;back to caller!

;*********************************************************************
;* play_isd(): Reproduce el mensaje numero W almacenado en la        *
;* memoria ISD1420, retorna el control al programa cuando termina de *
;* reproducirse el mensaje.                                          *
;* I: W = Numero de mensage                                          *
;* Destruye: W.                                                      *
;* Registros stack-eados: ax, bx y ex.                               *
;* Nota:                                                             *
;* 1) Si hay un codigo DTMF presente la funcion retorna imediatamente*
;*    sin reproducir el mensage de audio.                            *
;* 2) Si el mensaje se esta reproduciendo, retorna imediatamente,    *
;*    pero el mensaje no sera cortado y seguira reproduciendoce      *
;*    hasta que la isd1420 encuente un EOM.                          *
;* 3) El retardo desde que detecta un DTMF y la funcion retorna,     *
;*    puede ser desde 4 uS a 180 uS.                                 *
;*********************************************************************
play_isd:

   btfsc PORTA, 4    ;Hay un codigo DTMF presente en el 8870 ?
   return            ;Si. Salir de funcion.

   call push_ax
   call push_bx
   call push_ex
   movwf ax          ;ax=W -> Numero de mensage.

   ;Permitimos que mpause() retorne imediatamente si hay un DTMF.
   bsf UCT_OPTION, QF_DTMF

   ;Si play_isd() retorno de un llamado anterior sin haber esperado
   ;el EOM de un mensaje (por lo tanto no se reseteo el address
   ;pointer) esperamos un tiempo ISD_SURE_RTIME para que la isd1420
   ;pueda resetear su puntero interno, al dejar de reproducir el mensaje.
   ;Esto es util cuando se llama a play_isd() y el mensaje anterior
   ;todavia se esta reproduciendo. ISD_SURE_RTIME depende del
   ;tama#o en segundos del mayor mensaje almacenado en la ISD1420.
   ;El reset se produce por hardware debido a que en la isd1420,
   ;el pin 5 (A4) esta con un pull-down. Ver esquematicos.

   btfss UCT_OPTION, ISD_SURE_RESET ;ISD_SURE_RESET==1?
      goto configure_isd            ;No, entonces omitir espera.

   ;Esperar por (ISD_SURE_RTIME * 0xFF) milisegundos.
   movlw ISD_SURE_RTIME
   movwf bx
wait_ISD_SURE_RTIME:
      movlw 0xFF
      call mpause
      decfsz bx, F
      goto wait_ISD_SURE_RTIME
```

```
   bcf UCT_OPTION, ISD_SURE_RESET   ;ISD_SURE_RESET=0.

configure_isd:
   BANK_1
   ;Configuramos BUS de datos RB4-7 entre dispositivos.
   ;RB4=S Conectado a A0 de ISD1420, RB5=S Conectado a /PLAYE de ISD1420
   ;RB6=E Conectado a /RECLED de ISD1420, RB7=S Conectado a A4 de ISD1420
   movlw 0x0F
   andwf TRISB, F    ;TRISB=0000xxxx=SSSSxxxx
   bsf TRISB, 6      ;TRISB=0100xxxx=SESSxxxx

   BANK_0
   ;Valor inicial del BUS de datos RB4-7
   ;ISD1420: en modo MESSAGE CUEING + CONSECUTIVE ADDRESS
   movlw b'10110000'       ;RB7=RB5=RB4=1 => (A4=1,A0=1, /PLAYE=1)
   iorwf PORTB, F

   ;Seleccionamos ISD1420 para usar con BUS de datos.
   bsf PORTA, 1      ;RA1=1 -> Saca de alta Z a 4016.
   nop               ;Esperamos Tiempos de propagacion.
   nop

   ;Elegimos mensage a reproducir en ISD1420.
   decf ax, F           ;AX=W-1 (Ver principio de funcion)
loop_msg_select:
      bcf PORTB, 5      ;RB5=0 -> /PLAYE=0 pone puntero interno
                        ;de ISD en siguiente mensage.
      movlw d'20'
      call mpause       ;Esperar 20 mS.

      btfsc PORTA, 4    ;Hay un codigo DTMF presente en el 8870 ?
      goto end_play_isd ;Si. Salir de funcion.

      bsf PORTB, 5      ;RB5=1 -> /PLAYE=1
      movlw d'20'
      call mpause       ;Esperar 20 mS. Esperar Avance de Ptro. Interno.

      btfsc PORTA, 4    ;Hay un codigo DTMF presente en el 8870 ?
      goto end_play_isd ;Si. Salir de funcion.

      decfsz ax, F
      goto loop_msg_select

   ;Quitamos modo MESSAGE CUEING, pero dejamos CONSECUTIVE ADDRESS
   ;en ISD1420 para que no se resetee puntero interno.
   bcf PORTB, 4      ;RB4=0 -> A0=0 en ISD1420
   movlw d'5'        ;Esperar 5 mS
   call mpause

   btfsc PORTA, 4    ;Hay un codigo DTMF presente en el 8870 ?
   goto end_play_isd ;Si. Salir de funcion.

   ;Reproducimos mensage
   bcf PORTB, 5      ;RB5=0 -> /PLAYE=0 reproduce mensage seleccionado.
   movlw d'5'        ;Esperar 5 mS
   call mpause
```

```
   bsf PORTB, 5      ;RB5=1 -> /PLAYE=1

   ;Esperamos a que el mensage se termine de reproducir.
wait_play_msg:
      clrwdt
      btfsc PORTA, 4       ;Hay un codigo DTMF presente en el 8870 ?
      goto set_sure_reset  ;Si, Salir de funcion pero con SURE RESET.
      btfsc PORTB, 6       ;(RB6=/EOM) == 0?
      goto wait_play_msg   ;No, /EOM == 1. Seguir esperando.
      goto end_play_isd    ;Si, /EOM == 0. Salir de funcion.

set_sure_reset:
   ;Si hay un codigo DTMF presente en el 8870 y la ISD esta repro-
   ;duciendo un mensaje, ponemos ISD_SURE_RESET=1,
   ;asi en la proxima llamada a play_isd()
   ;esperamos ISD_SURE_RTIME, y nos aseguramos que la memoria
   ;ISD1420 sea reseteada, antes de reproducir un mensaje.

   bsf UCT_OPTION, ISD_SURE_RESET

end_play_isd:

   ;Reseteamos puntero interno de memoria de la ISD1420.
   ;SOLO valido, si la memoria no esta reproduciendo mensaje.
   bcf PORTB, 7       ;RB7=0 -> A4=0 en ISD1420.

   ;Quitamos ISD1420 del Bus de datos RB4-7
   bcf PORTA, 1       ;RA1 -> pone en alta Z a 4016.

   ;Dejamos Bus de datos RB4-7 conf. como salidas.
   BANK_1
   bcf TRISB, 6       ;RB6=S
   BANK_0             ;Dejamos banco cero (por convencion)
   movlw 0x0F
   andwf PORTB, F     ;PORTB=0000XXXX, Bus de datos a CERO.

   movlw d'20'
   call mpause        ;Esperar 20 mS. Por si se usa play_isd() en loop.

   ;QF_DTFM=0.
   bcf UCT_OPTION, QF_DTMF

   call pop_ax
   call pop_bx
   call pop_ex
   return             ;back to caller!
```

```
;*********************************************************************
;              RUTINAS DE APOYO INDEPENDIENTES DEL UCT                *
;*********************************************************************

;*********************************************************************
;* eeprom_read(): W=*(W)                                             *
;* Regresa en W el contenido de una direccion de la memoria EEPROM   *
;* almacenado en W.                                                  *
;* I: W = direcccion de memoria EEPROM                               *
;* O: W = valor de la variable en la direccion de memoria EEPROM.    *
;*********************************************************************
eeprom_read:
   BANK_0
   movwf EEADR       ;Direccion a leer
   BANK_1
   bsf EECON1, RD    ;Leer EEPROM
   BANK_0
   movf EEDATA, W    ;W = EEDATA

   return            ;Back to caller!

;*********************************************************************
;* eeprom_write(): Escribe el valor de W en la memoria EEPROM, en la *
;* direccion contenida en ex.                                        *
;* I: W  = valor a escribir en la mem EEPROM.                        *
;*    ex = direccion de la mem EEPROM a escribir.                    *
;* Destruye: W                                                       *
;* Nota: bit GIE de INTCON es activado.                              *
;*********************************************************************
eeprom_write:

   BANK_0
   movwf EEDATA
   movf ex, W
   movwf EEADR

   BANK_1
   bcf INTCON, GIE      ;impedir interrupciones.
   bsf EECON1, WREN     ;activar permiso de escritura de eeprom.
   movlw 0x55           ;start: secuencia de inicializacion.
   movwf EECON2
   movlw 0xAA
   movwf EECON2         ;end: secuencia de inicializacion.

   bsf EECON1, WR       ;empezar a escribir en eeprom.

wait_eeprom_write_time:
      clrwdt
      btfss EECON1, EEIF            ;escritura finalizada?
      goto wait_eeprom_write_time   ;No, seguir esperando.

   bcf EECON1, EEIF     ;reponer flag.
   bcf EECON1, WREN     ;desactivar permiso de escritura de eeprom.

   bsf INTCON, GIE      ;activar interrupciones.
   BANK_0
```

```
   return               ;back to caller!


;*****************************************************************
;* MINI-STACK_SYSTEM:                                            *
;* Funciones para salvar y reponer registros de forma individual.*
;* Funciones pop_*() reponen un registro salvado previamente con *
;* push_*().                                                     *
;* Donde * = [ax|bx|cx|dx|ex|w|STATUS]                           *
;* ATENCION: Usar dos veces concecutivas push_*() sobreescribira el *
;* valor almacenado previamente con el primer push_*().          *
;* Por ello en cada funcion que use el stack debemos, especificar *
;* que registro pone en el stack para no sobrescribir y perder   *
;* datos.                                                        *
;* NOTA:                                                         *
;* 1) Ninguna funcion modifica el registro W, excepto pop_w().   *
;* 2) Excepto push_w(), push_STATUS(), pop_w() y pop_STATUS(), todas *
;*    las funciones PUEDEN modificar el bit Z del registro STATUS.*
;* AUTOR: Boris Estudiez.                                        *
;*****************************************************************

;*****************************************************************
;* pop_*(): Reponen un registro que fue almacenado con push_*(). *
;*****************************************************************

pop_ax:
   movwf TEMPX       ;TEMPX=W
   movf TEMP_ax, W   ;W=TEMP_ax
   movwf ax          ;ax=TEMP_ax=W
   movf TEMPX, W     ;W=TEMPX
   return
pop_bx:
   movwf TEMPX
   movf TEMP_bx, W
   movwf bx
   movf TEMPX, W
   return
pop_cx:
   movwf TEMPX
   movf TEMP_cx, W
   movwf cx
   movf TEMPX, W
   return
pop_dx:
   movwf TEMPX
   movf TEMP_dx, W
   movwf dx
   movf TEMPX, W
   return
pop_ex:
   movwf TEMPX
   movf TEMP_ex, W
   movwf ex
   movf TEMPX, W
   return
pop_w:                   ;pop_w no altera al registro STATUS.
   swapf TEMP_W, F       ;TEMP_W = swap(TEMP_W)
```

```
   swapf TEMP_W, W       ;W=swap(TEMP_W)
   return
pop_STATUS:              ;pop_STATUS, Restaura STATUS y no afecta el
   movwf TEMPX           ;registro STATUS ni a W.
   swapf TEMP_STATUS, W  ;Ver push_STATUS para entender esta linea.
   movwf STATUS
   swapf TEMPX, F
   swapf TEMPX, W
   return

;*********************************************************************
;* push_*(): Ponen un registro en el stack.                          *
;*********************************************************************
push_ax:
   movwf TEMPX       ;TEMPX=W
   movf ax, W        ;W=ax
   movwf TEMP_ax     ;TEMP_ax=W=ax
   movf TEMPX, W     ;W=TEMPX
   return
push_bx:
   movwf TEMPX
   movf bx, W
   movwf TEMP_bx
   movf TEMPX, W
   return
push_cx:
   movwf TEMPX
   movf cx, W
   movwf TEMP_cx
   movf TEMPX, W
   return
push_dx:
   movwf TEMPX
   movf dx, W
   movwf TEMP_dx
   movf TEMPX, W
   return
push_ex:
   movwf TEMPX
   movf ex, W
   movwf TEMP_ex
   movf TEMPX, W
   return
push_w:                  ;push_w y push_STATUS no modifican el registro
   movwf TEMP_W          ;STATUS.
   return
push_STATUS:
   movwf TEMPX           ;TEMPX=W
   swapf STATUS, W       ;W=swap(STATUS)
   movwf TEMP_STATUS     ;TEMP_STATUS=W
   swapf TEMPX, F        ;TEMPX=swap(TEMPX)
   swapf TEMPX, W        ;W=swap(swap(TEMPX))=TEMPX
   return

;*********************************************************************
   END   ;Fin de codigo/Archivo. (EOF?)
;*********************************************************************
```

[ARCHIVO: UCT.INC]

```
; asmsyntax=pic16f84
;
; File    : uct.inc
; Compiler: mplab v6.61
;
; MCU     : PIC-16F84A @ 4 MHz
; Created : 18/Sep/2003
; Revision: 05/Jan/2005
;
; Author  : Boris Estudiez <stk@freeshell.org>
; Licence : GPL
;
; DESCRIPCION:
; Definicion de Constantes/Macros/Variables para uct.asm.
;
; (c) Copyright slicetex electronixs (2001-2005)
;     http://stk.freeshell.org
;


;*****************************************************************
;* Definicion de Macros Generales                                *
;*****************************************************************
#define BANK_0 bcf STATUS, RP0
#define BANK_1 bsf STATUS, RP0

;*****************************************************************
;* Definicion de "variables estaticas" en EEPROM .               *
;*                                                               *
;* PRIMEROS 15 BYTES RESERVADOS PARA CONF. DEL UCT               *
;*****************************************************************

;[0x00 - 0x02] ID DEL DISPOSITIVO:
;ID_2:ID_1:ID_0 = 0x000258 = 600 -> STX600

EEPROM_ADDR:   ORG 0x2100
STX_ID_0       equ   0x00 ;ID_0 del dispositivo
               de    0x58 ;ID_0 = 0x58
STX_ID_1       equ   0x01 ;ID_1 del dispositivo
               de    0x02 ;ID_1 = 0x02
STX_ID_2       equ   0x02 ;ID_2 del dispositivo
               de    0x00 ;ID_2 = 0x00

;[0x03 - 0x05] VERSION DEL FIRMWARE:
;FV_2:FV_1:FV_0 = '0' '8' '1' -> 0.8.1,  en ASCII.

FV_2           equ   0x03 ;FV_0 del dispositivo
               de    '0'  ;FV_0 = '0'
FV_1           equ   0x04 ;FV_1 del dispositivo
               de    '8'  ;FV_1 = '8'
FV_0           equ   0x05 ;FV_2 del dispositivo
               de    '1'  ;FV_2 = '1'

;[0x06 - 0x0E] CONFIGURACION DEL UCT:
```

```
EE_RINGS        equ   0x06 ;Cantidad de RINGS para atender llamado.
                de    d'4' ;Valor de fabrica: 4 RINGS.
EE_PASSW_0      equ   0x07 ;Digito 3 y 2 de la clave de acceso (MSB)
                de    0x12 ;Valor de fabrica: 1 y 2
EE_PASSW_1      equ   0x08 ;Digito 1 y 0 de la clave de acceso (LSB)
                de    0x34 ;Valor de fabrica: 3 y 4
EE_DISPSTAT     equ   0x09 ;Ultimo Estado de Dispositivos ON/OFF 0000xxxx
                de    0x00 ;Valor de fabrica: Todos disp. desactivados.

;[0xF - 0x3F] LIBRES.

;*****************************************************************
;* Declaracion de Nombres de Variables en RAM                    *
;*****************************************************************
ax              equ 0x20 ;Variable de proposito general
bx              equ 0x21 ;Variable de proposito general
cx              equ 0x22 ;Variable de proposito general
dx              equ 0x23 ;Variable de proposito general
ex              equ 0x24 ;Variable de proposito general
UCT_STATUS      equ 0x30 ;Registro de estado del UCT.
                         ;Bit0=RINGING. Intento de llamada.
                         ;Bit1=ANSWERCALL. Respondiendo llamada.
                         ;Bit2=DCEINT. Interrupcion por DCE.
                         ;Bit3=TIMEOUT. Ocurrio un time-out.
                         ;(se debe borrar antes luego de ser usada).
UCT_OPTION      equ 0x31 ;Opciones de funcionamiento del UCT .
                         ;Bit3=ISD_SURE_RESET. Reset seguro de puntero
                         ;     de direcciones para ISD1420.
                         ;Bit4=SKIPMENU. Evita repro. mensages de menu.
                         ;Bit5=QF_DTMF. Salir de funcion al recibirse un
                         ;     codigo DTMF. Solo valido para mpause().
                         ;Bit6=QMSG. Reiniciar reproduciendo mensage/aviso.
                         ;Bit7=QTIMEOUT. Reiniciar al producirse un TIMEOUT.
RINGS_LEFT      equ 0x32 ;Numero de RINGS que faltan para contestar llamado.

                         ;DIRECCIONES 0x33 a 0x39 reservadas para el
                         ;Bus DCE. No USAR!. Ver mas abajo.

TMR0_AUX        equ 0x46 ;Variable auxiliar para ser usada como contador
                         ;del timer.
SECONDS         equ 0x47 ;Almacena numero de segundos que se deben
                         ;temporizar. Usada por set_timeout.
TEMPX           equ 0x48 ;Variable para almacenar valor temporal.
                         ;No usar. P/uso interno de funciones del stack.
TEMP_ax         equ 0x49 ;Ultimas 7 direcciones reservadas para usar
TEMP_bx         equ 0x4A ;con funciones pop_* y push_*
TEMP_cx         equ 0x4B ;Ambas manejan el stack de profundidad: 1
TEMP_dx         equ 0x4C
TEMP_ex         equ 0x4D
TEMP_W          equ 0x4E
TEMP_STATUS     equ 0x4F
```

```
;*****************************************************************
;* Definicion de constantes numericas del programa.              *
;*****************************************************************

DTMF_AST       equ 0x0B  ;Codigo del DTMF Asterisco.
DTMF_NUM       equ 0x0C  ;Codigo del DTMF Numeral.

_LOT_RINGS     equ d'18' ;Permite poner 18 RINGS. Ver conf_ring_menu().

ERR            equ 0xFF  ;Operacion no exitosa.
OK             equ 0x00  ;Operacion exitosa.

;UCT_OPTION BITS:
QTIMEOUT       equ 7
QMSG           equ 6
QF_DTMF        equ 5
SKIPMENU       equ 4
ISD_SURE_RESET equ 3

;UCT_STATUS BITS:
TIMEOUT        equ 3
DCEINT         equ 2
ANSWERCALL     equ 1
RINGING        equ 0

;CTES. DE TIEMPO - Pueden usarse con set_timeout().
M04            equ d'240'  ;240 segundos -> 4 Minutos.
S60            equ d'60'   ; 60 segundos -> 1 Minuto.
S30            equ d'30'   ; 30 segundos
S10            equ d'10'   ; 10 segundos
S2             equ d'2'    ;  2 segundos
S1             equ d'1'    ;  1 segundos

;CTES. DE TIEMPO - Usadas por play_isd().
ISD_SURE_RTIME equ d'3'    ; ISD_RESET_RTIME*0xFF milisegundos.

;*****************************************************************
;* DEFINICIONES PARA EL BUS DCE                                  *
;* DCE = Dispositivo de Control Externo                          *
;* (Variables, Constantes, Macros)                               *
;*****************************************************************

;PINES DEL BUS DCE.
;Interrupciones (utilizan logica negada):
#define        DCEINT_PIN    PORTB, 2  ;Interrupme a un DCE.
#define        UCTINT_PIN    PORTB, 1  ;Comprueba Interrupcion por DCE.
#define        TXPIN         PORTB, 4  ;Pto. para transmision serie (TX).
#define        RXPIN         PORTB, 5  ;Pto. para recepcion serie (RX).

;*****************************************************************
;* TRANSMISION SERIE ASINCORNA (1200 BAUD, 8N1).                 *
;*****************************************************************

;VARIABLES PARA LA TRANSMISION SERIE, ASINCRONA:
RXTX_BUF       equ 0x33               ;RX/TX BUFFER.
BCOUNTER       equ 0x34               ;BIT COUNTER.
SDCNT          equ 0x35               ;SERIAL DELAY COUNTER.
```

```
;CTES. PARA LA TRANSMISION SERIE, ASINCRONA:
BITS           equ d'8'                ;Numbers of bits to transmit.


;MACROS PARA LA TRANSMISION SERIE, ASINCRONA:

;delay_HalfBit(): produce un retardo de 416 uS o de Medio Bit.
delay_HalfBit macro
   call HalfBit_Pause
   endm
;delay_FullBit(): produce un retardo de 832 uS o de Un Bit.
delay_FullBit macro
   call HalfBit_Pause
   call HalfBit_Pause
   endm

;*****************************************************************
;* CTP (COMMAND TRANSMISSION PROTOCOL).                          *
;*****************************************************************

;VARIABLES: usadas por funciones del CTP.
B0             equ 0x36  ;BYTE 0 De Trama.
B1             equ 0x37  ;BYTE 1 De Trama.
B2             equ 0x38  ;BYTE 2 De Trama.
CTPSTAT        equ 0x39  ;Registro de estado del CTP.

;CTPSTAT BITS:
CTP_ERR        equ 0     ;1 -> Error En Conexion.
RESET_RCV      equ 1     ;1 -> Comando UCT_RESET Recibido.
END_RCV        equ 2     ;1 -> Comando END_CTP Recibido.
GM_MENU_RCV    equ 3     ;1 -> Comando GOTO_MAIN_MENU Recibido.
CMD_NOT_FOUND  equ 7     ;1 -> Comando No existente.

;CONSTANTES: usadas por funciones del CTP.
RXWAIT         equ d'30' ;Espera en segundos para la recepcion de 1 Byte.
TRDELAY        equ d'3'  ;Retardo en mS para la transmision de una trama.
WAITWAKEUP     equ d'100';100 mS de espera para que el DCE o UCT despierte.

HELOLSB        equ 0xAA  ;Byte 0 de trama HELO
HELOMSB        equ 0x55  ;Byte 1 de trama HELO

;*****************************************************************
;* COMANDOS soportados por el UCT para ser ejecutados a traves   *
;* del BUS DCE.                                                  *
;* Utilizados en el byte CMD de la trama COMMAND del protocolo   *
;* CTP.                                                          *
;* Ver funcion uct_ctpd() en uct.asm, para ver descripcion       *
;* detallada de los comandos.                                    *
;*****************************************************************

END_CTP         equ 0xF5  ;El UCT termina la comunicacion con DCE.
UCT_RESET       equ 0x99  ;El UCT se resetea y vuelve a modo espera.

FREE_PHONE_LINE equ 0x98  ;El UCT desocupa linea telefonica.
GET_PHONE_LINE  equ 0x88  ;El UCT ocupa linea telefonica.
DIALP_NUMBER    equ 0xCC  ;El UCT marca el numero B1 por Pulsos.
```

```
PING            equ 0x87  ;No hace nada, solo devuelve PING. En B0.
UCT_PAUSE       equ 0x55  ;El UCT se queda pausado por B1 segundos.
PLAYISD         equ 0x66  ;Repoduce el mensage B2 de la ISD1420.
WR_EEPROM       equ 0x77  ;Escribir B1 en direccion B2 de eeprom.
RD_EEPROM       equ 0x44  ;Leer direccion B1 de eeprom.
WR_RAM          equ 0x3A  ;Escribir B1 en direccion B2 de la RAM.
RD_RAM          equ 0x13  ;Leer direccion B1 de la RAM.
GETDTMF         equ 0xEE  ;Esperar por B1 segundos un codigo DTMF.
GETDTMFSTR      equ 0xDD  ;Esperar 1 minuto por B1 codigos/Digitos DTMF.
CTRLDISP        equ 0xBB  ;Ver ctrl_disp(). Argumento en B1.
GOTO_MAIN_MENU  equ 0x93  ;El UCT toma el control y repoduce el menu principal.


;*****************************************************************
;* Numeros de posicion Asociados a los Mensages Almacenados en la *
;* memoria ISD1420.                                              *
;*****************************************************************

;Silencio (primeros dos mensages), Duracion 25 ms cada uno.
ISD_SILENCIO_1              equ d'1'
ISD_SILENCIO_2              equ d'2'

;Generales/Genericos
ISD_NUMERO                  equ d'3'
ISD_INCORRECTO              equ d'4'
ISD_INGRESAR                equ d'5'
ISD_DISPOSITIVO             equ d'6'
ISD_LISTO                   equ d'7'
ISD_REPETIR                 equ d'8'
ISD_DESACTIVADO             equ d'9'
ISD_ACTIVADO                equ d'10'
ISD_CLAVE                   equ d'11'
ISD_CAMBIAR                 equ d'12'
ISD_SALIR                   equ d'13'
ISD_CONFIGURAR              equ d'14'
ISD_ERROR                   equ d'15'

;Menus
ISD_CONTROL_DE              equ d'16'
ISD_VER_ESTADO_DE           equ d'17'
ISD_BUS_DCE                 equ d'18'

;Teclas
ISD_UNO                     equ d'19'
ISD_DOS                     equ d'20'
ISD_TRES                    equ d'21'
ISD_CUATRO                  equ d'22'
ISD_CINCO                   equ d'23'
ISD_ASTERISCO               equ d'24'
ISD_NUMERAL                 equ d'25'

;Especiales
ISD_EXTERNO                 equ d'26'
ISD_CONECTAR                equ d'27'
ISD_RINGS                   equ d'28'
ISD_TELEFONICO              equ d'29'
ISD_ALARMA_2                equ d'30'
```

```
;Otros
ISD_BIENVENIDO              equ d'31'
```

[LISTA DE CAMBIOS DESDE VERSIÓN 0.1 DE SOFTWARE]

```
0.8.1 (05/Jan/2005):
* Se agrega comando WR_RAM para ser ejecutado por el Bus DCE.

0.8.0 (04/Jan/2005):
* Pocos cambios. Se prueba bien la version 0.7.2, obteniendose
  resultados correctos.

0.7.2 (03/Jul/2004) [EXPERIMENTAL]:
* Se corrige funcion uct_ctpd() y se le agregan nuevos
  comandos para ser ejecutados a traves del Bus DCE.

0.7.1 (02/Jul/2004) [EXPERIMENTAL]:
* Se agrega soporte para marcado de numeros
  telefonicos mediante pulsos. Funcion dialp().
* Se elimina soporte para Bus DCE en modo desconectado
  para ahorrar espacio en memoria.
* Se elimina bdce_write().
* Se modifica menu del Bus DCE.

0.7.0 (23/May/2004) [EXPERIMENTAL]:
* Bus DCE terminado por software, se permite conectar unidirecionalmente
  con un DCE (Dispositivo de Control Externo) a traves del protocolo
  CTP (Protocolo de Transmision de Comandos).
* Funciones agregadas: ssend(), srcv(), ctp_snd(), ctp_rcv(),
  bdce_open(), bdce_close(), uct_ctpd().
* Codigo general Optimisado para ahorrar memoria de programa.
* Correciones Varias.
* Bug: instruccion sleep eliminada ya que producia error
  cuando el DCE interrumpia al UCT. En el futuro
  sera reincorporada nuevamente.

0.6.2 (16/May/2004):
* PWRTE=OFF, soluciona conmutacion aleatoria de reles
  (K2,...,K5) de lo dispositivos ON/OFF, al energizarse
  el UCT por primera vez.

0.6.1 (15/May/2004):
* Correccion en funcion play_isd().
  Ahora resetea correctamente el address pointer de la
  isd1420, al ser interrumpida por un codigo DTMF.

0.6.0 (15/May/2004):
* Se permite interrumpir reproduccion de un mensaje de audio
  al recibirse un codigo DTMF.
* Funciones play_isd() y mpause() modificadas.

0.5.2 (05/Apr/2004):
* El programa soporta el Watchdog Timer (WDT) del PIC16F84A.

0.5.1 (30/Jan/2004):
* Se agrega bdce_write().
* Bus DCE en modo desconectado terminado.
* Nuevo Hardware/software actualizado.
* Mejor organizacion del codigo fuente.
```

```
* Nueva correccion de errores.

0.4.8 (19/Jan/2004):
* Agrega funciones para configurar RINGS y
  cambiar clave.
* Corrige funcion getdtmf().
* Corrige ctrl_disp_menu.
* Cambios en uct.inc.
* RB0/INT por flanco negativo.

0.4.4 (18/Jan/2004):
* Mejor uso del stack.
* Mejor uso de interrupciones.
* Corrige funcion ctrl_disp().
* Corrige errores de version anterior.
* Agrega funcion close_call().

0.4.0 (18/Jan/2004):
* Corrige errores de version anterior.
* Rutinas de dispositivos verificadas.
* Rutinas principales Verificadas.
* Version Estable.

0.3.9 (18/Jan/2004):
* Primera version Funcional, simplifica funciones y
  estructura interna.
* Utiliza temporizacion Interna.
  Funciones de manejo de tiempo set_timeout(),
  unset_timeout() y pause() y mpause().
* Soluciona problemas de reproduccion de audio.
  Corrige funcion play_isd().
* Testeo en hardware exitoso.
* Muchos errores Corregidos.


0.0.11 (19/Dec/2003):
* Funciones principales terminadas.
* Version llena de errores, inestable y no
  funciona en hardware :(...

0.0.1 (18/Sep/2003):
* Primera version beta!...Estructura de programa principal
  finalizada.
* Temporizacion externa por NE555 :(...
```

5.3: Lista de componentes electrónicos del UCT

Esta lista puede bajarse mas actualizada quizás de la pagina del proyecto.

PARTLIST:

Exported from uct.sch at 25/02/2004 01:37:45a

EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft

PART	VALUE	DEVICE	PACKAGE	LIBRARY	SHEET
B1	1-AMP	RB1A	RB1A	rectifier	1
C1	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C2	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C3	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C4	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C5	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C6	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C7	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C8	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C9	0.1uF	C-EU050-030X075	C050-030X075	rcl	1
C10	470uF/25V	CPOL-USE5-8.5	E5-8,5	rcl	1
C11	470uF/16V	CPOL-USE5-8.5	E5-8,5	rcl	1
C12	220nF	C-EU075-042X103	C075-042X103	rcl	1
C13	470nF/400V	C-EU150-072X183	C150-072X183	rcl	1
C14	220uF/16V	CPOL-USE5-6	E5-6	rcl	1
C15	22pF	C-EU050-030X075	C050-030X075	rcl	2
C16	22pF	C-EU050-030X075	C050-030X075	rcl	2
C17	10nF	C-EU050-030X075	C050-030X075	rcl	2
C18	10nF	C-EU050-030X075	C050-030X075	rcl	2
C19	0.1uF	C-EU050-030X075	C050-030X075	rcl	2
C20	220nF	C-EU025_050-045X075	C025_050-045X075	rcl	2
C21	220nF	C-EU025_050-045X075	C025_050-045X075	rcl	2
C22	220nF	C-EU050-045X075	C050-045X075	rcl	3
C23	220nF	C-EU050-045X075	C050-045X075	rcl	3
D1	1N4007	1N4004	DO41-10	diode	1
D2	12V	ZENER-DIODEZD-7.5	ZDIO-7.5	diode	1
D3	12V	ZENER-DIODEZD-7.5	ZDIO-7.5	diode	1
D4	20V	ZENER-DIODEZD-7.5	ZDIO-7.5	diode	1
D5	20V	ZENER-DIODEZD-7.5	ZDIO-7.5	diode	1
D6	1N4007	1N4004	DO41-10	diode	1
D7	1N4148	1N4148	DO35-10	diode	2
D8	1N4148	1N4148	DO35-10	diode	3
D9	1N4004	1N4004	DO41-10	diode	3
D10	1N4004	1N4004	DO41-10	diode	3
D11	1N4004	1N4004	DO41-10	diode	3
D12	1N4004	1N4004	DO41-10	diode	3
IC1	7812T	7812T	TO220H	linear	1
IC2	7805T	7805T	TO220H	linear	1
IC3	4093N	4093N	DIL14	40xx	1
IC4	PIC16F84AP	PIC16F84AP	DIL18	microchip	2
IC5	4066N	4066N	DIL14	40xx	2
IC6	ISD1420	ISD1400D	DIL28-6	isd	2
IC7	CM8870CP	CM8870CP	DIL18	california-micro-devices	2
IC8	4042N	4042N	DIL16	stdlib	3
IC9	4066N	4066N	DIL14	40xx	3
J1	RJ11 JACK	RJ11	RJ11	stdlib	1
J2	RJ11 JACK	RJ11	RJ11	stdlib	1
JP1		PINHD-1X2	1X02	pinhead	1

JP2		JP1E	JP1	jumper	2
JP3		PINHD-1X2	1X02	pinhead	2
JP4		PINHD-2X7	2X07	pinhead	3
K1	RELE 5V	G6A-234P	G6A-234P	relay	1
K2	RELE	G5L	G5LE	relay	3
K3	RELE	G5L	G5LE	relay	3
K4	RELE	G5L	G5LE	relay	3
K5	RELE	G5L	G5LE	relay	3
OK1	4N25	4N35	DIL06	optocoupler	1
Q1	4 Mhz	XTAL	Q	special	2
Q2	BC546B	BC546B	TO92-EBC	transistor-npn	2
Q3	3.58 Mhz	XTAL	Q	special	2
Q4	BC546B	BC546B	TO92-EBC	transistor-npn	3
Q5	BC546B	BC546B	TO92-EBC	transistor-npn	3
Q6	BC546B	BC546B	TO92-EBC	transistor-npn	3
Q7	BC546B	BC546B	TO92-EBC	transistor-npn	3
R1	1K	R-US_0207/10	0207/10	rcl	1
R2	470	R-US_0207/10	0207/10	rcl	1
R3	VARISTOR-150V	VARISTOR-7,5	R-7,5	varistor	1
R4	10K	R-US_0207/10	0207/10	rcl	1
R5	1K	R-US_0207/10	0207/10	rcl	1
R6	10K	R-US_0207/10	0207/10	rcl	1
R7	100	R-US_0207/10	0207/10	rcl	2
R8	4.7K	R-US_0207/10	0207/10	rcl	2
R9	4.7K	R-US_0207/10	0207/10	rcl	2
R10	10K	R-US_0207/10	0207/10	rcl	2
R11	10K	R-US_0207/10	0207/10	rcl	2
R12	470	R-US_0207/10	0207/10	rcl	2
R13	100K	R-US_0207/10	0207/10	rcl	2
R14	100K	R-US_0207/10	0207/10	rcl	2
R15	100K	R-US_0207/10	0207/10	rcl	2
R16	1K	R-US_0207/10	0207/10	rcl	2
R17	1K	R-US_0207/10	0207/10	rcl	2
R18	100K 1%	R-US_0207/10	0207/10	rcl	2
R19	100K %1	R-US_0207/10	0207/10	rcl	2
R20	300K / 1%	R-US_0207/10	0207/10	rcl	2
R21	100K 1%	R-US_0207/10	0207/10	rcl	2
R22	4.7K 1%	R-US_0207/10	0207/10	rcl	2
R23	120K 1%	R-US_0207/10	0207/10	rcl	2
R24	120K 1%	R-US_0207/10	0207/10	rcl	2
R25	33K 1%	R-US_0207/10	0207/10	rcl	2
R26	4.7K	R-US_0207/10	0207/10	rcl	3
R27	100K	R-US_0207/10	0207/10	rcl	3
R28	4.7K	R-US_0207/10	0207/10	rcl	3
R29	4.7K	R-US_0207/10	0207/10	rcl	3
R30	4.7K	R-US_0207/10	0207/10	rcl	3
R31	4.7K	R-US_0207/10	0207/10	rcl	3
S1		SLIDING-INVERSOR-SW	2X03	stdlib	1
S2		NON_INV_SWITCH	1X02	stdlib	1
T1	2N3904	2N3904	TO92	transistor	1
TF1	1:1, Z=600 @ 400Hz	TEL_TRAFO_1_1	TEL_TRAFO_1_1	stdlib	1
X1		W237-02P	W237-132	con-wago-508	1
X2		W237-02P	W237-132	con-wago-508	3
X3		W237-02P	W237-132	con-wago-508	3
X4		W237-02P	W237-132	con-wago-508	3
X5		W237-02P	W237-132	con-wago-508	3

[FIN DE PROYECTO UCT]

Website : http://stk.freeshell.org
E-Mail : stk@freeshell.org

Notas: