Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica-Eléctrica

DISEÑO DE UNA RED DE PAGO DE REMESAS POR MEDIO DE PUNTOS DE VENTA

Juan Manuel Flores Rodríguez

Asesorado por el Ing. Enrique Edmundo Ruiz Carballo

Guatemala, marzo de 2006

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UNA RED DE PAGO DE REMESAS POR MEDIO DE PUNTOS DE VENTA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA

FACULTAD DE INGENIERÍA

POR

JUAN MANUEL FLORES RODRIGUEZ

ASESORADO POR EL ING. ENRIQUE EDMUNDO RUIZ CARBALLO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, MARZO DE 2006

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	
VOCAL II	Lic. Amahán Sánchez Álvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Enrique Edmundo Ruiz Carballo
EXAMINADOR	Ing. Guillermo Puente Romero
EXAMINADOR	Ing. Francisco Gonzáles López
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UNA RED DE PAGO DE REMESAS POR MEDIO DE PUNTOS DE VENTA,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha noviembre 18 de 2005.

_____

Juan Manuel Flores Rodríguez

DEDICATORIA

Dedico este trabajo a Dios quien me dió la sabiduría para completar mi carrera, a mi familia que me apoyo y me animó durante estos años, a mis amigos con los que compartimos tanto en estos 5 años de estudio.

AGRADECIMIENTOS

A la Universidad de San Carlos de Guatemala.

A Ing. Enrique Edmundo Ruiz Carballo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

RESUMEN

Los Punto de venta son, ampliamente, utilizados para la realización de transacciones electrónicas, Se diseñará una red para pagar remesas enviadas desde el extranjero, las cuales podrán ser cambiados en puntos donde estén instalados estos dispositivos; las ordenes de pago de remesas se envían a una base de datos central, una ves cargadas en la base de datos estas pueden ser pagadas en cualquier punto que sea de conveniencia del beneficiario.

Cuando el beneficiario de la orden llegue a un centro de cobro dará su código original que será el código de la remesa a cambiar, por medio del punto de venta que se comunica con el contestador central se accesa a la base de datos y se verifica la orden; una ves verificada el beneficiario decide si la quiere cobrar en cheque, efectivo o depósito. Se solicitan datos de identificación, número de cheque o cuenta donde depositar, los cuales son ingresados al punto de venta el cual envía los datos al contestador central y se realiza toda la transacción para completar el pago. En la base de datos de registra el pago y se llenan los campos requeridos por cada orden y, por último, se actualiza la orden y se coloca como pagada la orden.

Los puntos de venta estarán localizados en varios puntos de la capital e interior de la república de manera que se facilite el pago de remesas a los destinatarios y, de ese modo, hacer este procedimiento rápido y eficiente.

OBJETIVOS

GENERAL

Diseñar e implementar una red de pago de remesas en Guatemala, por medio de los terminales llamados Punto De Venta.

ESPECÍFICOS

1. Diseñar y crear una base de datos de manera que funcione como un servidor de órdenes de remesas, donde se pueda llevar un control total de las órdenes.

2. Diseñar y programar un contestador capaz de interactuar con la base de datos y los terminales de punto de venta.

3. Diseñar y programar una aplicación en los puntos de venta que cumpla con los requerimientos necesarios para pagar una orden.

INTRODUCCIÓN

En la actualidad, la economía de Guatemala se ve beneficiada año con año por el envió de remesas desde el exterior, principalmente desde Estados Unidos. El envió de remesas hacia Guatemala va destinado, principalmente, hacia el interior del país, ya es de aquí de donde es originaria la mayoría de emigrantes.

Existen pocos métodos de envío de remesas, entre estos podemos mencionar el envió de efectivo o cheques por correo, envío de remesa a bancos del sistema al utilizar empresas que sirven como mediadoras , en Guatemala, gran mayoría de los beneficiarios que viven en el interior de la república se desplazan hacia la capital donde tienen que recibir sus pagos al carecer de un sistema de pago en sus respectivas comunidades o pueblos, solamente Banrural presta servicio de pago de remesas en el interior.

El proyecto que se desea desarrollar consistirá en la programación de aplicaciones destinadas a realizar los pagos en los pos e interconexión de los PDV en varios puntos y programación de un contestador central o servidor que llevara el control y monitoreo de las transacciones realizadas en estos de modo que se creará una red que hará que el pago de las remesas sea mas eficiente y ,así, ser mas competitivos respecto de otras empresas dedicadas a la misma operación.

1. ESTRUCTURA GENERAL DE LA RED

1.1 La remesa en la actualidad

La remesa se considera como una transacción privada de familia a familia, Las remesas reducen, por lo general, el nivel y la severidad de la pobreza, al igual que modifican la distribución de ingresos.

Las remesas de los trabajadores se han duplicado en la última década, alcanzando a $216 mil millones en 2004, de los cuales $151 mil millones fueron remitidos a países en desarrollo. Se cree que los flujos reales de remesas, si se cuentan aquellos a través de canales informales, son aún más grandes. Los flujos de remesas superan hoy día la ayuda total para el desarrollo y representan la fuente más grande de divisas para algunos países.

Según una próxima publicación del Banco Mundial “Perspectivas de la economía mundial 2006” cerca de 200 millones de personas viven en países distintos a su país natal y las remesas que envían a su país de origen llegarían a los US$225.000 millones en 2005. La cifra convierte a las remesas en la principal fuente de divisas de muchos países y tiene consecuencias de suma importancia para las estrategias de lucha contra la pobreza de las naciones en desarrollo.

La migración priva a los países pobres de muchos trabajadores y profesionales calificados, pero también es fuente de las remesas de dinero que desde los países ricos ayudan a aliviar la pobreza. En el ámbito global, la migración aumenta la producción económica mundial, puesto que permite que los trabajadores se trasladen a lugares donde son más productivos y reciban

sueldos más altos. Pero la salida masiva de ciudadanos altamente calificados representa un complejo dilema para muchos países pequeños y de bajo ingreso

Las remesas difieren de los flujos de inversiones extranjeras directas, de deuda o de ayuda. Las remesas conducen con frecuencia a una mayor acumulación de capital humano, gastos en educación, inversión y emprendimiento.

1.1.1 La remesa en Guatemala

Las remesas familiares provenientes de EE.UU. Desde enero hasta septiembre de este año se estima que ha llegado a una cantidad de US$2 mil 177 millones, 17 por ciento más que en el mismo período del año pasado, cuando el banco central registró un flujo de US$1 mil 864 millones. En todo el año se estima que el país recibirá US$3 mil millones.

Aparte del ingreso registrado a septiembre, en los primeros seis días de octubre el país había recibido otros US$47.3 millones, para acumular a esa fecha US$2 mil 225 millones.

En Guatemala, las remesas (de los emigrantes) reducen el nivel, la profundidad y la gravedad de la pobreza, ya que estos recursos representan más de la mitad de los ingresos del 10 por ciento más pobre de las familias. Gran parte de las ganancias de las remesas se acumula a favor de los emigrantes y de sus familias con las remesas que envían a su hogar, tras semanas de arduo trabajo en fábricas y tiendas.

En Guatemala, los hogares (rurales y urbanos) que reciben remesas gastan relativamente más en educación y, en proporción, menos en consumo diario. No obstante, en México rural, los niños de familias con emigrantes obtienen

menos educación que aquellos sin emigrantes, probablemente debido a que pretenden seguir el ejemplo de sus padres y emigrar a Estados Unidos en busca de empleos que no requieren mayores calificaciones, para lo cual la educación no es necesaria ni recibe mayor compensación.

En Estados Unidos radican alrededor de 1.2 millones de guatemaltecos, en su gran mayoría (más de 60 por ciento) en condición de indocumentados.

Se espera que este año, las remesas superen los tres mil millones de dólares, contra dos mil 550.6 millones de dólares (marca histórica) alcanzada en 2004, y dos mil 106 millones de dólares obtenida en 2003.

Entre 1960 y 2002, un total de 1,24 millones de guatemaltecos dejaron el país para vivir en el exterior: 95 por ciento se dirigieron a Estados Unidos, 2,1 por ciento a México y 1,2 por ciento a Canadá. Dentro de Estados Unidos, casi un tercio de los guatemaltecos viven en Los Angeles; otro tercio está dividido entre Nueva York, Miami, Washington D.C., Houston, Chicago y Norfolk, Virginia.

Del total de guatemaltecos que viven en el exterior, 88 por ciento, o 1,1 millón, están económicamente activos. Ellos dejaron atrás a 4,2 millones de parientes directos, que representan 36 por ciento del número total de familias de Guatemala.

Seis de cada 10 familias guatemaltecas con miembros residentes en el exterior viven en el campo. Sólo 30 por ciento de ellas viven de la agricultura o la pesca, y sólo 16 por ciento se consideran indígenas. Entre los emigrantes, sin embargo, 41 por ciento trabajó en tareas agrícolas o pesqueras en Guatemala. Esto da la impresión de que los agricultores tienden a producir más emigrantes por familia que otros grupos.

La mayoría de los emigrantes son hombres (72,7 %). Este hecho se atribuye al hecho de que la mayoría viajan por medios irregulares, y los riesgos para las mujeres son mayores. Más de 90 por ciento de los inmigrantes guatemaltecos tenían entre 15 y 44 años cuando dejaron el país, lo cual demuestra el enorme potencial de la población de esa edad para participar en actividades económicas.

Casi 80 por ciento de los emigrantes envían remesas a sus familiares. Cuarenta y tres por ciento lo hacen mensualmente, 14 por ciento dos veces al mes, 14 por ciento dos veces al año y 14 por ciento una vez al año. Cerca de 57 por ciento de los que envían remesas lo hacen a través de giros postales, y 30 por ciento mediante transferencias bancarias.

Más de 600.000 familias en Guatemala reciben remesas. En promedio, cada hogar recibe entre 1.500 y 2.000 dólares por año. En algunos casos asciende a 26 por ciento del monto enviado (por ejemplo, Western Union cobra 13 dólares por enviar 50).

De los 4,2 millones de guatemaltecos con familiares en el exterior, más de un millón procede de los departamentos occidentales de Solola, Totonicapán, Quetzaltenango, Suchitepequez, Retalhuleu y San Marcos; 755.000 viven en áreas metropolitanas y 644.000 de los departamentos sudorientales de Santa Rosa, Jalapa y Jutiapa. En las regiones del norte y noreste, que tienen mayoría de población indígena, menos de 16 por ciento de la población emigrante está constituida por mujeres. Sin embargo, en las áreas metropolitanas, más de 40 por ciento de los emigrantes son mujeres.

1.1.2 Nuevo método para pagos de remesas

Debido al alto flujo de remesas en el país (US$3 mil millones) se ha buscado una manera de captar la gran mayoría de estas que son enviadas por medio de sistemas bancarios que equivale a un 30 % del total (alrededor US$ 900 millones). Ya que la gran mayoría de emigrantes ha salido del interior de la republica es aquí a donde se concentran los envíos de las remesas.

En la actualidad son pocos los bancos que pagan estos envíos en el interior como por ejemplo BANRURAL y las casas remesadoras tienen sus oficinas en la capital, lugar hasta donde se tienen que desplazar los beneficiarios para recibir el pago de sus remesas, es debido a esto que si se quiere captar la mayor cantidad de remesas es necesario hacer el pago de las remesas mas accesible a los beneficiarios. Esto tiene como consecuencia la apertura de centros de pagos en varios puntos de la capital y del interior del país.

Para poder instalar un punto de pago, comúnmente es necesaria el alquiler de un local, la instalación de equipo de computación, reguladores de voltaje, impresoras, tinta y tener una conexión directa ya sea por medio de una red WAN o por web con la casa remesadora para obtener las remesas a pagar y confirmar los pagos ya realizados.

Debido al alto costo que tendría la instalación de un equipo de computación cuyo precio actualmente es de alrededor de Q6000 con impresora mas la tinta que se utilizaría para imprimir los recibos, papel, conexiones, etc. El costo sería muy elevado por cada centro de pago. Se busco una solución a este problema y se encontró la siguiente solución: el uso de Puntos De Venta, estos dispositivos son utilizados para realizar la gran mayoría de transacciones electrónicas monetarias en la actualidad.

Un Punto De Venta o PDV , es un dispositivo que compacta en una pequeña presentación todo lo necesario para realizar transacciones comerciales, incluye pantalla, teclado para navegar en los menús, impresora termina (la cual no requiere tinta) y MODEM para interconectarse con otros dispositivos así como su propio regulador de voltaje.

El precio de cada punto de venta oscila los US$ 250, que en moneda local equivale a unos Q 2000, lo cual comparado con una computadora es un precio mucho más accesible.

Lo más importante el punto de venta esta diseñado para estar conectado las 24 horas del día y no requiere de mantenimiento ya que por estar todo integrado en un solo dispositivo cerrado no necesita mantenimiento alguno como las computadoras que por estar compuesta de varios dispositivos es mas probable que falle alguna de sus partes por la acumulación de polvo, variaciones de voltaje o mal uso del equipo por parte del usuario.

A continuación se presenta una tabla comparativa de los costos de instalar los centros de pago de remesas con equipo de computación y con PDV.

Tabla I. Comparación de la instalación de centros de pago de remesas

DESCRIPCIÓN	PRECIO		DESCRIPCIÓN	PRECIO
Computador	Q5000		Punto de venta	Q2000
Impresora	Q1500		impresora	------
Tinta mensual	Q150		Tinta mensual	------
Papel mensual	Q50		Papel mensual	Q10
Mantenimiento mensual	Q200		Mantenimiento mensual	------
Local tamaño mediano mensual	Q1000		Local Pequeño mensual	Q500
TOTAL	Q7900		**TOTAL**	Q2510

Según la tabla anterior es más rentable la instalación de un dispositivo de punto de venta que el de una computadora personal. Además ocupa menos espacio y no necesita de demasiado entrenamiento para que su operador la pueda manejar.

1.1.3 Proceso de pago de remesas

En los locales o los kioscos donde se instalen los puntos de venta el beneficiario podrá llegar y con su número de código único y plenamente identificado podrá obtener el pago de su remesa. El beneficiario puede escoger entre 3 tipos de pagos: efectivo, cheque o depósito bancario.

Las ordenes son enviadas por casas remesadoras en EEUU. Por medio de Internet y estas cuando entran al sistema general se les asigna un estado de pendientes de pago (1) y pagada (2). Y también su forma de pago, si va a ser cambiada por medio de efectivo o cheque llevan una forma de pago 1 y si van a ser cambiadas por medio de depósito llevan forma de pago 2.

Todas los centros de pago pueden cambiar las remesas en efectivo y hacer depósitos bancarios, pero cuando se trata de cheque, se tiene que cargar dentro de la base de datos en la tabla bancos, una cuenta bancaria y a que banco pertenece esta cuenta, estos datos están asociados a cada punto de venta ya que cada punto de venta puede pagar con sus propios cheques del banco mas cercano para la conveniencia del beneficiario. Cada punto de venta va a tener una cuenta propia asociada para pagar con cheque.El punto de venta cuando se encuentra sin actividad se mantiene desplegando el logotipo de la compañía en la pantalla, esto para identificar a que compañía pertenece el punto de venta.

Para poder utilizar el punto de venta el operador de este debe presionar el botón de menú y le aparecerá una pantalla en la cual se le pedirá su número de clave para poder utilizar el punto de venta. Cada operador de punto de venta se le será asignado una clave única para poder utilizar todas las funciones. Al ingresar su clave de ingreso el operador presiona *enter* y automáticamente el punto de venta por medio del MODEM se comunicara con el contestador y le enviara el número de clave y el número de la terminal, ya recibido esto el contestador verificara en la base de datos si la clave ingresada concuerda con el número de terminal de punto de venta al que se le asigno esa clave. Si la clave es incorrecta se envía un mensaje en el cual se indica que la clave es incorrecta, si la clave es la indicada se prosigue con el proceso de pago de la

remesa. Al operador le aparecerá el menú principal en el cual hay 4 opciones: pago, consulta, depósito y reversa.

En el submenú pago se ingresara si se quiere pagar la remesa en efectivo o en cheque, si la remesa se quiere pagar en deposito se ingresa al submenú depósito.En el submenú de consulta, se ingresa el numero original asignado a la remesa y el punto de venta lo enviara al contestador y este a su vez verificara en la base de datos entre las ordenes pendientes por pagar si esta orden no se ha pagado, si la orden no se ha pagado se enviará hacia el punto de venta el monto de la orden por pagar. Si la orden es inválida o ya se ha pagado; se enviará un mensaje que indique que la orden no fue encontrada.

En el submenú reversa se ingresa el número de una orden a la cual se reversara el pago, solo se ingresa el código original y se espera a que se imprima el recibo, cuando se reversa una orden se envía el código de la orden al contestador y este cambia el estado de la orden a pendiente de pago.

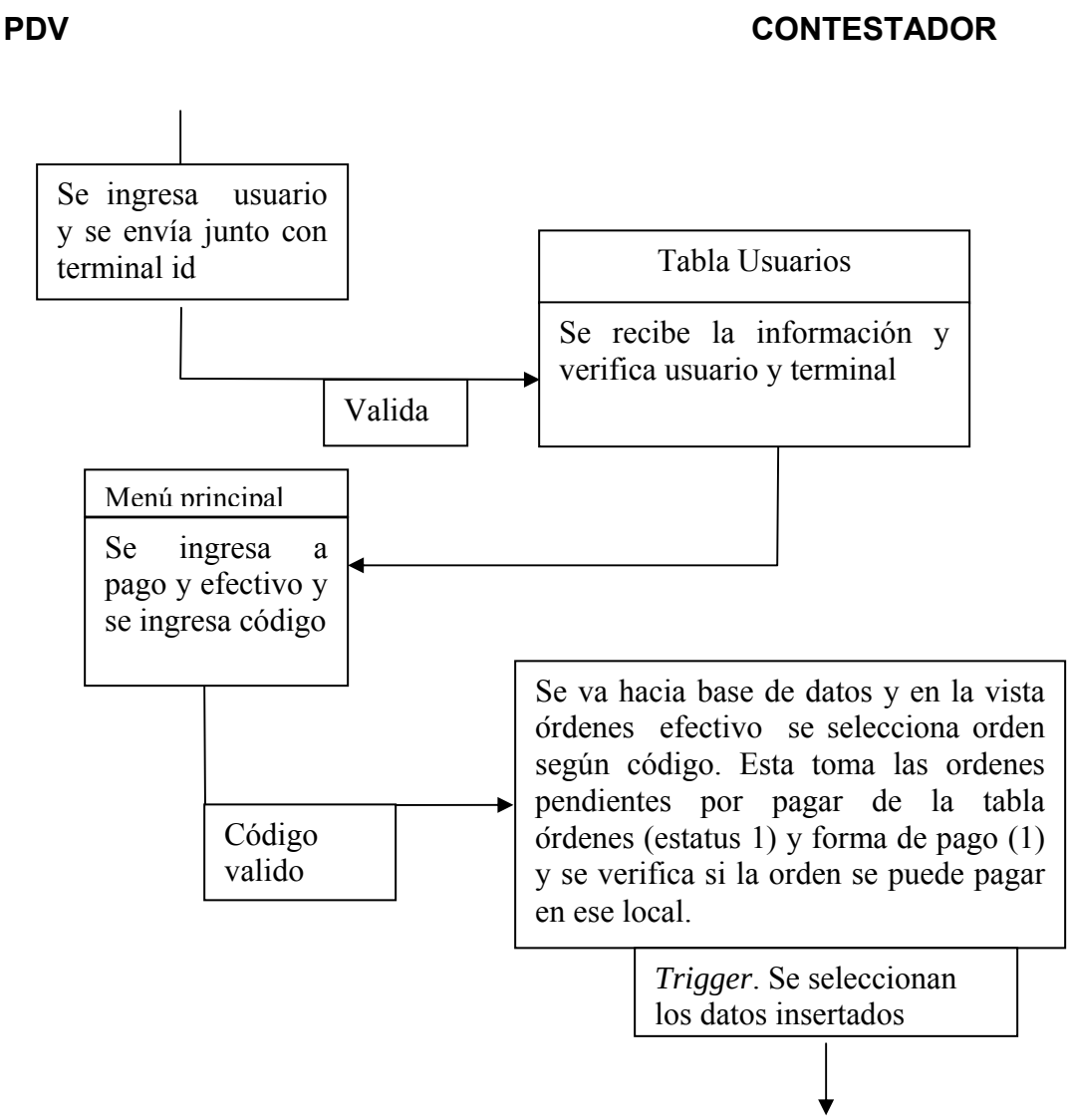
Una vez realizado el pago de la remesa, el punto de venta imprime el recibo para el beneficiario y una copia para la compañía.

1.1.3.1 Pago de remesas en efectivo

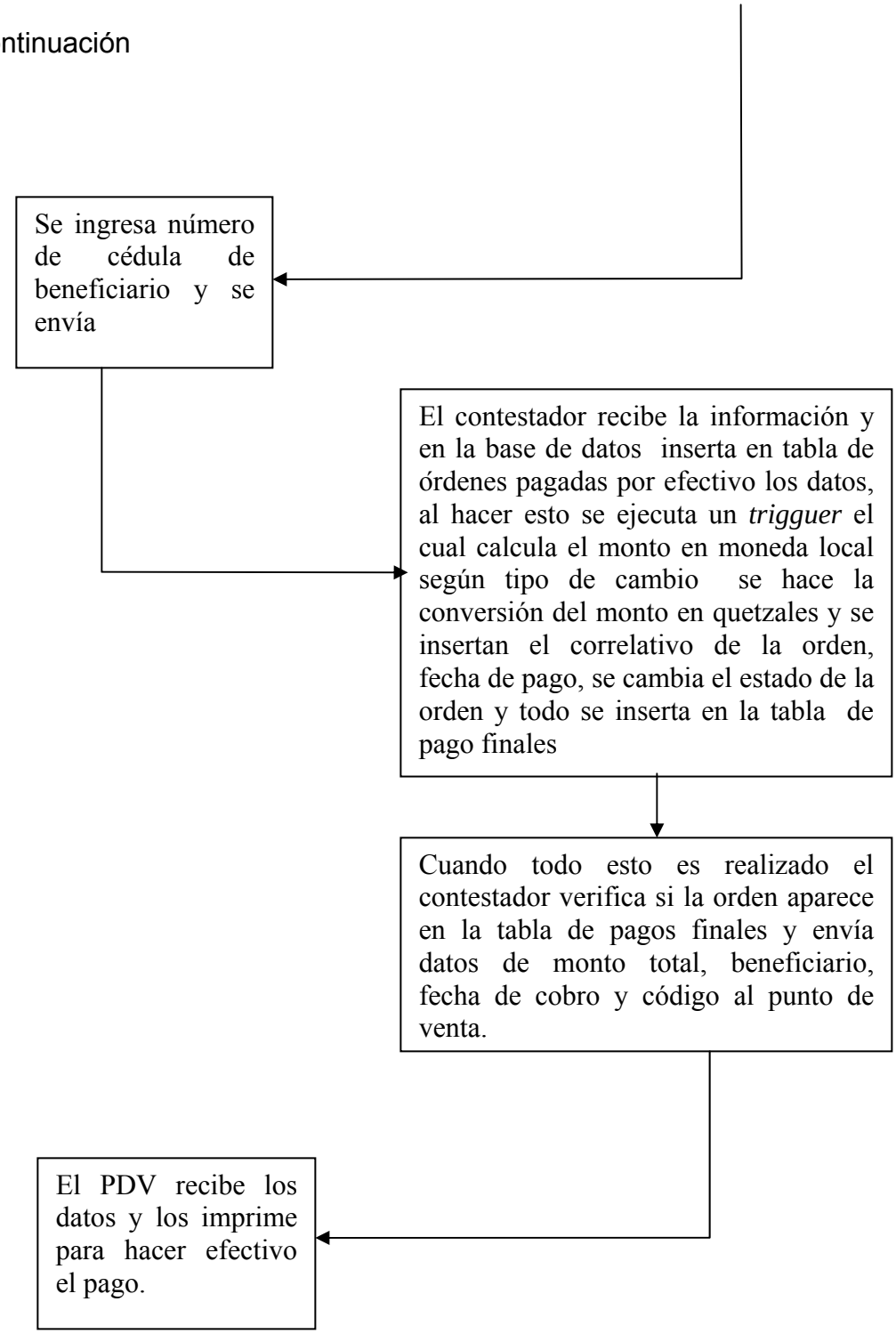
Cuando el beneficiario indica que desea cambiar su remesa en efectivo, el operador del punto de venta ingresa su código y accesa al menú principal, aquí se ingresa en el submenú pago y se despliegan 2 opciones: pago en cheque y pago en efectivo. Se selecciona pago en efectivo, para hacer efectivo este pago se le pide al beneficiario el código original asignado a su orden y su identificación, seguidamente se ingresan estos datos en el punto de venta y se envían al contestador, el contestador recibe los datos y en la base de datos los inserta en una tabla de ordenes pagadas en efectivo y también los inserta en

una tabla de pagos finales, actualiza el estado de la orden de pendiente de pago a pagada y envía hacia el punto de pago de regreso hora y fecha en que se realizo el pago, el código original, nombre del remitente de la orden y el monto total de esta. El punto de venta recibe la información e imprime 2 recibos con esta.

Figura 1. Lógica de pago en efectivo



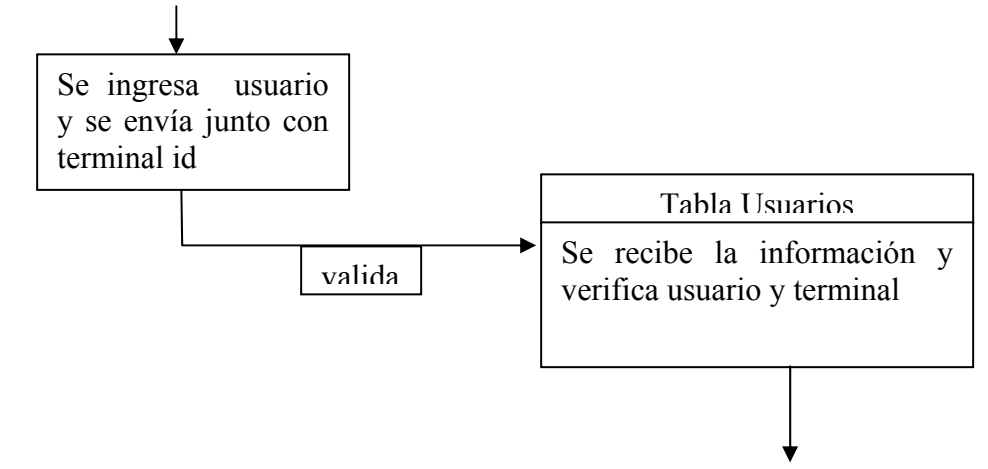
Continuación



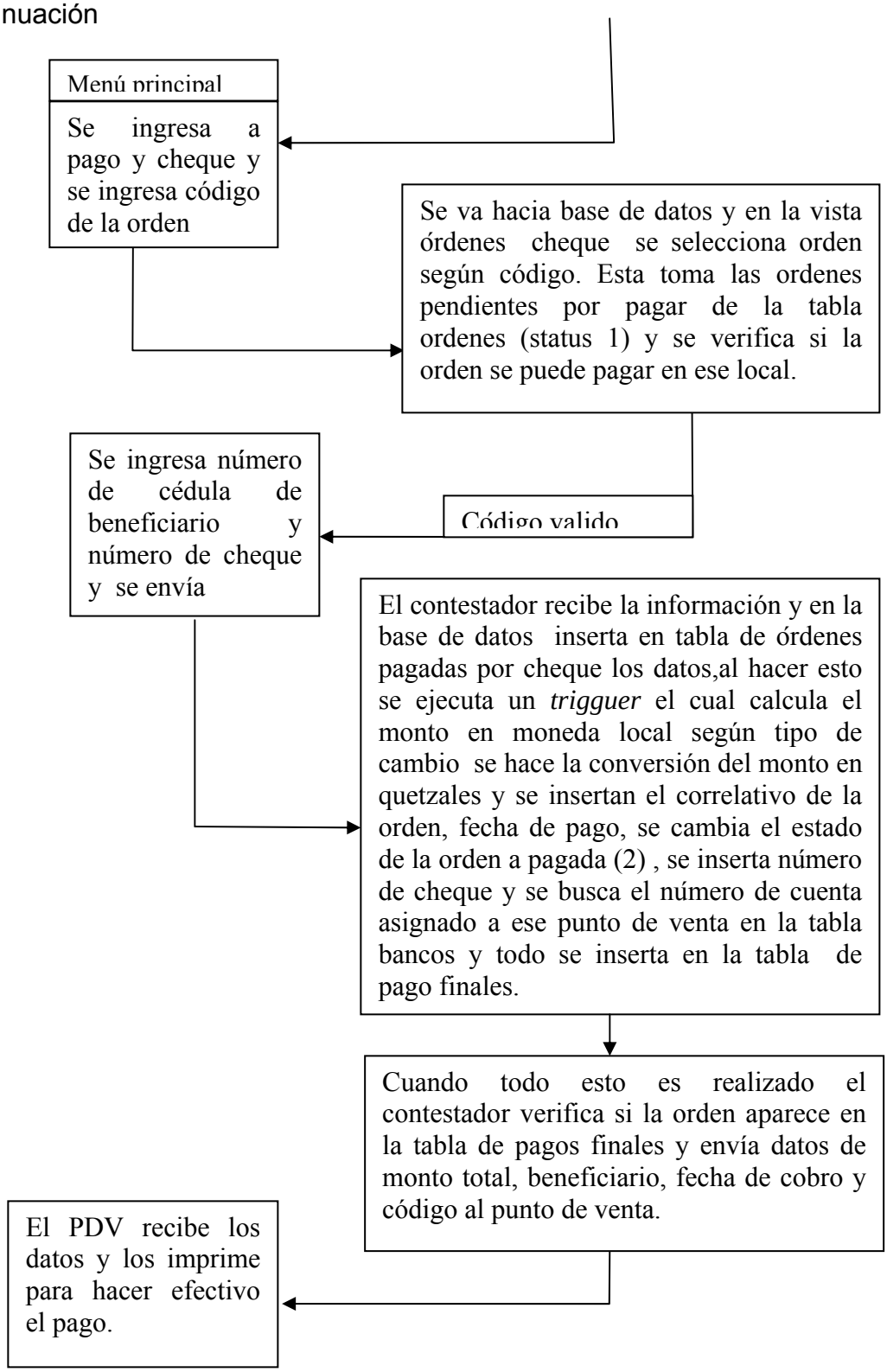
1.1.3.2 Pago de remesas en cheque

Cuando el beneficiario indica que va a cambiar su remesa en cheque, el operador del punto de venta ingresa su código y accesa al menú principal, aquí se ingresa en el submenú pago y se despliegan 2 opciones: pago en cheque y pago en efectivo. Se selecciona pago en con cheque, para hacer efectivo este pago se le pide al beneficiario el código original asignado a su orden y su identificación, y seguidamente se ingresan estos datos en el punto de venta junto con el numero correlativo del cheque con el que se va a pagar la orden y se envían al contestador, el contestador recibe los datos y en la base de datos los inserta en una tabla de ordenes pagadas con cheque y también los inserta en una tabla de pagos finales, actualiza el estado de la orden de pendiente de pago a pagada y envía hacia el punto de pago de regreso hora y fecha en que se realizo el pago, el código original, nombre del remitente de la orden, nombre del beneficiario, número de cheque, banco de donde se paga la orden y el monto total de esta. El punto de venta recibe la información e imprime 2 recibos con esta.

Figura 2. Lógica de pago de remesa con cheque



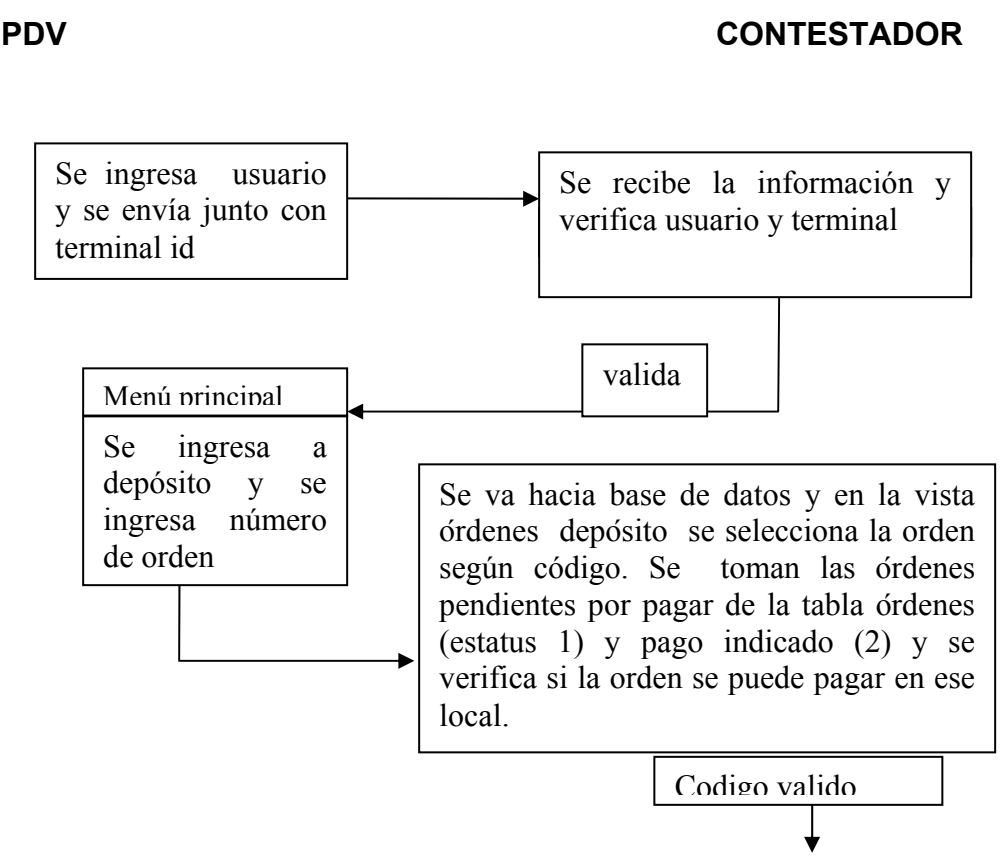
Continuación



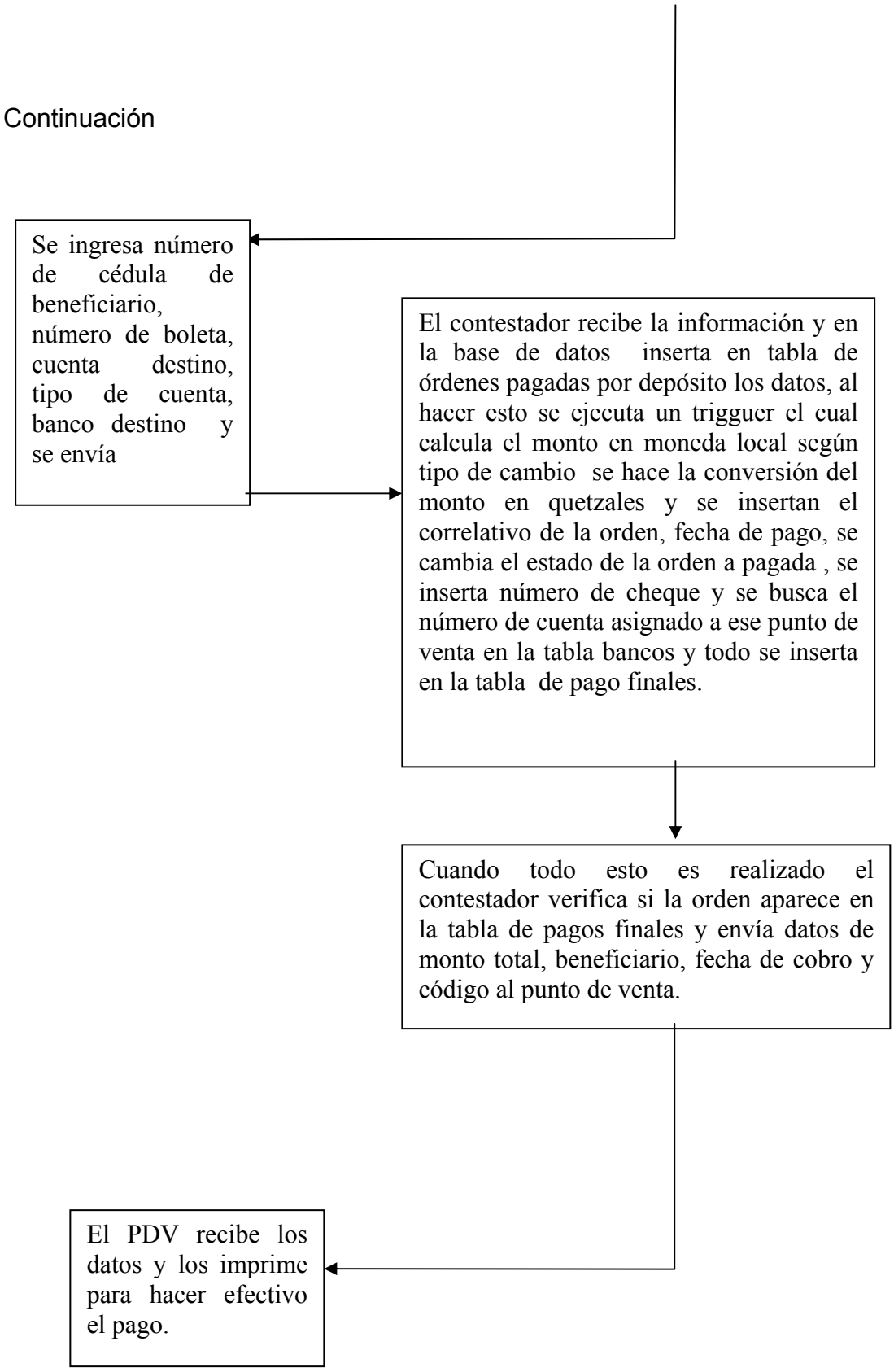
1.1.3.3 Pago de remesas en depósito

Cuando el beneficiario indica que va a cambiar su remesa en depósito bancario, el operador del punto de venta ingresa su código y accesa al menú principal, aquí se ingresa en el submenú deposito e inmediatamente se le pide el ingreso del número de código original, numero de cuenta donde se realizará el depósito, nombre del banco donde se realiza el depósito, tipo de cuenta, numero de boleta y fecha. Toda la información ingresada es enviada al contestador que a su vez la inserta en la base de datos en la tabla órdenes pagadas cheque y pago final, después el contestador regresa al punto de venta que la transacción ha sido realizada y el punto de venta imprime los recibos.

Figura 3. Lógica de pago de remesa con depósito



Continuación

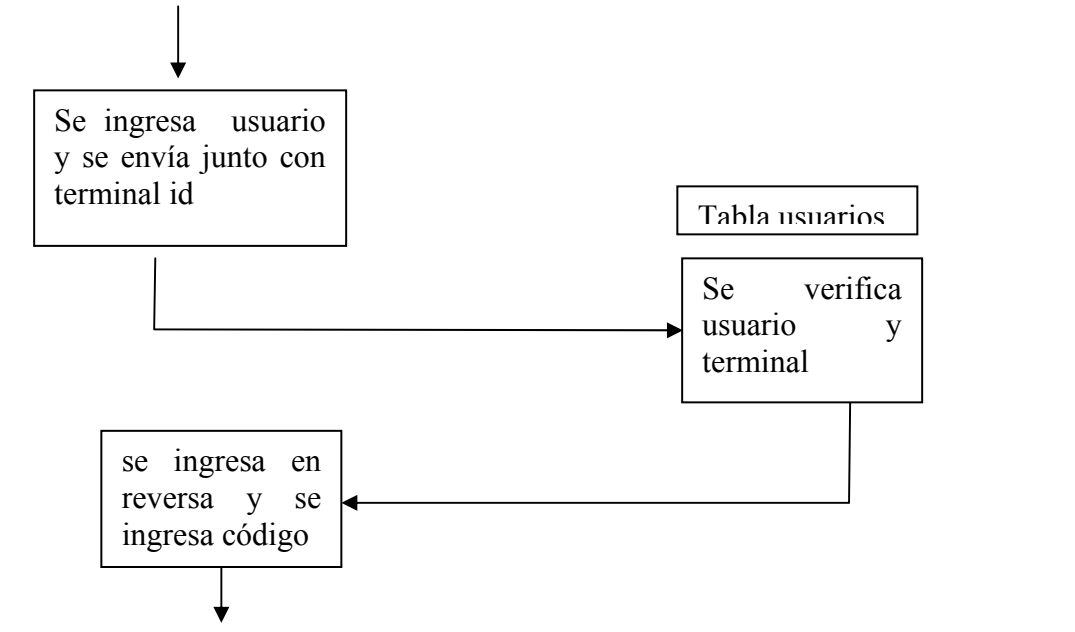


1.1.3.4 Reversa de pago

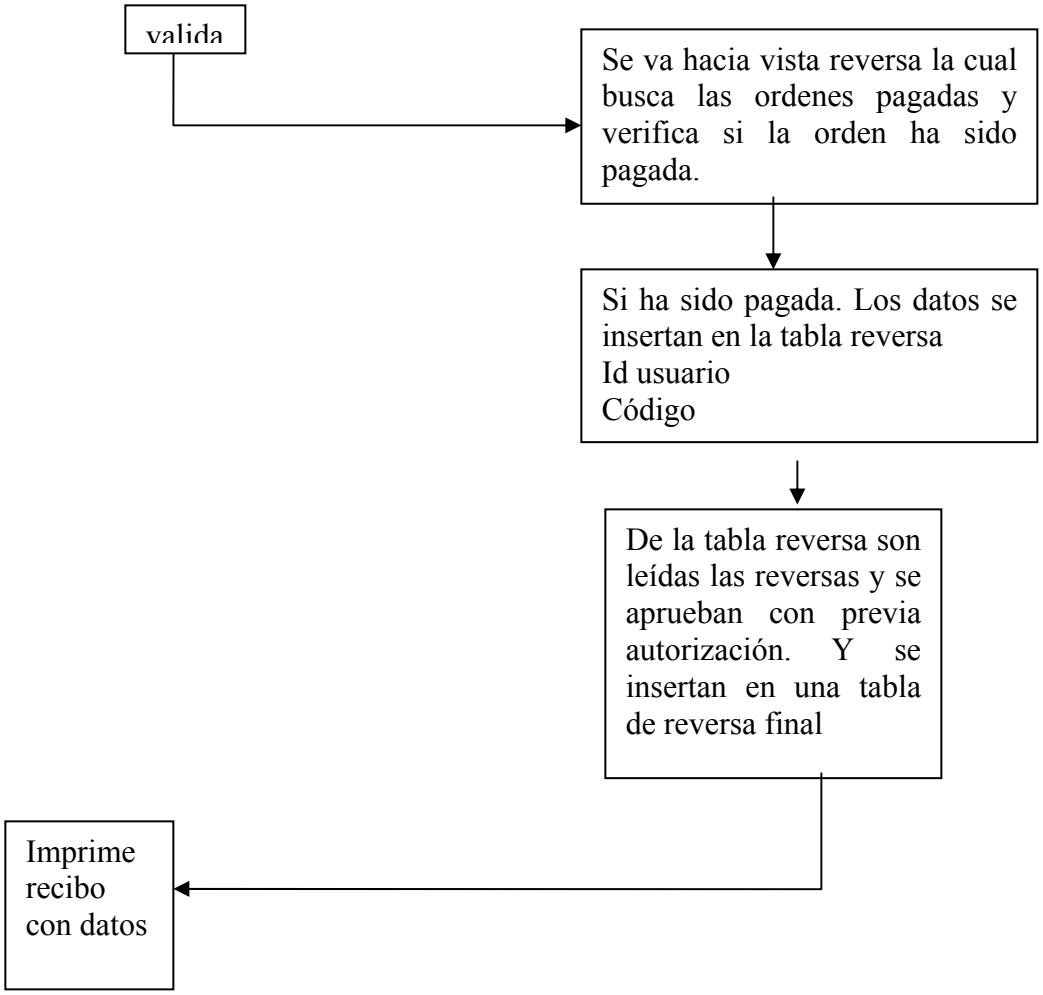
La reversa de pago se realiza cuando una orden no termina su procedimiento en su totalidad por ejemplo cuando se cae la conexión entre el punto de venta y el contestador, en este caso que se llegara a poner la orden como pagada pero no llegue a imprimir el recibo final, se necesita hacer una reversa del pago. Esta reversa pondrá disponible la orden para ser pagada de nuevo y borrará los campos insertados, cuando se realice una reversa se debe aprobar desde las oficinas centrales la reversa ya que las causas de estas deben ser verificadas.

También es posible realizar la reversa de un pago cuando se comete el error en el ingreso de los datos que se requieren para cualquier pago ya antes mencionado.

Figura 4. Lógica de reversa de pago



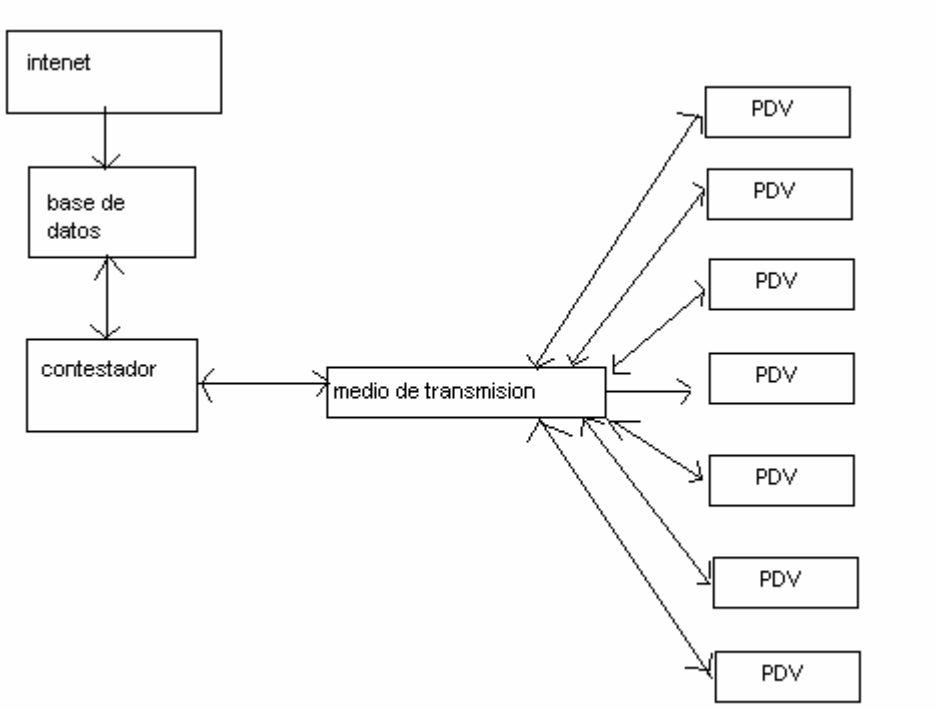
Continuación



1.2 Diagramacion

Se hará la diagramación general de la red, esta consiste en la representación de todos los bloques que se involucran en el proceso de pago de la remesa y se explicará por separado su función y cual es su función especifica dentro de la red.

Figura 5. Diagrama general de la red



1.2.2 Explicación de diagramas en bloque

INTERNET: este bloque simboliza el medio por el cual ingresan las ordenes al sistema provenientes de EEUU u otros países.

BASE DE DATOS: una vez ingresadas la ordenes estas son guardadas en una base de datos, esta base de datos esta creada en SQL, la base de datos es instalada en un servidor dedicado a esta función. La base de datos para el pago de remesas en puntos de venta consiste de 7 tablas y 4 vistas.

Las tablas que se utilizan son

ÓRDENES.

ÓRDENES PAGADAS EFECTIVO.

ÓRDENES PAGADAS CHEQUE.

ÓRDENES PAGADAS DEPÓSITO.

ÓRDENES PAGADAS FINALES.

REVERSA.

REVERSA FINAL.

Las vistas que se utilizan son

ÓRDENES DEPÓSITO.

ÓRDENES CHEQUE.

ÓRDENES EFECTIVO.

ÓRDENES PAGADAS.

CONTESTADOR: El contestador es un programa realizado en *Borland C Builder*, el cual es un lenguaje de programación orientado a objetos. El contestador es el encargado de atender todas las peticiones generadas por los diferentes puntos de venta instalados, se encarga de realizar los pedidos de los puntos de venta en la base de datos y de devolver los datos requeridos a estos así como de manejar la conexión hacia los MODEM que controlan cada línea disponible para la conexión hacia los puntos de venta.

MEDIO DE TRANSMISIÓN: El medio de transmisión de los datos es la línea telefónica. Ya que los puntos de venta tienen varias maneras de conectarse con los servidores, si se encuentran dentro de una red de computadoras el punto de venta tiene puertos de *ethernet,* y también tienen MODEM integrado el cual se utilizo para realizar la comunicación con el contestador, los MODEM utilizan la línea telefónica para su intercomunicación.

PDV: puntos de venta, son utilizados para la realización de transacciones electrónicas. En nuestra red son utilizados para poder acceder a las órdenes que están en la base de datos remota. El punto de venta a utilizar será el NURIT 8320 de la empresa LIPMAN

2. DESARROLLO DE APLICACIÓN DEL PDV

2.1 Descripción del equipo y software

2.1.1 Punto de venta

El punto de venta que se utilizara es de marca LIPMAN modelo Nurit 8320 (Fig. 6), el cual fue diseñado para negocios medianos y el cual integra características importantes como seguridad, confiabilidad y manejabilidad en un diseño compacto y ergonómico. Este punto de venta incorpora un teclado alfanumérico, una impresora térmica y lectoras de tarjetas inteligente y magnética. Esta diseñado para cumplir con los requerimientos de seguridad tan altos como los de visa o Mastercard, permitiendo así transferencias seguras y descarga de aplicaciones confiables. Este punto de venta puede correr varias aplicaciones y gracias a su sistema operativo evita el traslape de aplicaciones.

Debido a su flexibilidad se le puede conectar un teclado externo, dispositivo de captura de firma, lector de barras y cajas registradoras.

Figura 6. Punto de venta

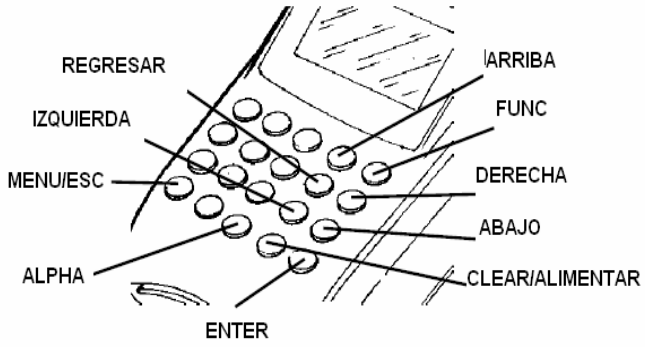
Utiliza el sistema operativo NOS (*Nurit operative Sistem*) versión 7.0 el cual tiene compatibilidad con versiones anteriores y futuras lo cual garantiza flexibilidad en el desarrollo de aplicaciones, viene en varios idiomas para mejor manejo de los menús.

Este punto de venta trae una pantalla de 128x64 píxeles, un teclado de 20 teclas que incluyen números, letras y flechas para navegar en los menús los cuales se pueden observar en la grafica 2.2, un MODEM que se puede comunicar desde una velocidad de 300 bps hasta 14,400 bps el cual puede funcionar de modo síncrono o asíncrono (Fig. 7).

El procesador que utiliza es un ARM 7 de 32 bits, tiene 4 Mega bites de memoria. Se puede interconectar con dispositivos externos por medio de un puerto RS-232, puerto *ethernet* el cual utiliza protocolo TCP/IP y MODEM.

Trae un adaptador de voltaje de DC de 12 voltios con el cual se alimenta el punto de venta y se regula el voltaje. Tiene un compartimiento para baterías con lo cual se garantiza funcionabilidad de hasta 5 horas sin alimentación.

Figura 7. Teclas de la unidad Nurit 8320



El sistema operativo NOS 7 contiene varias funciones las cuales se describen en la siguiente tabla.

Tabla II. Funciones del sistema operativo NOS 7

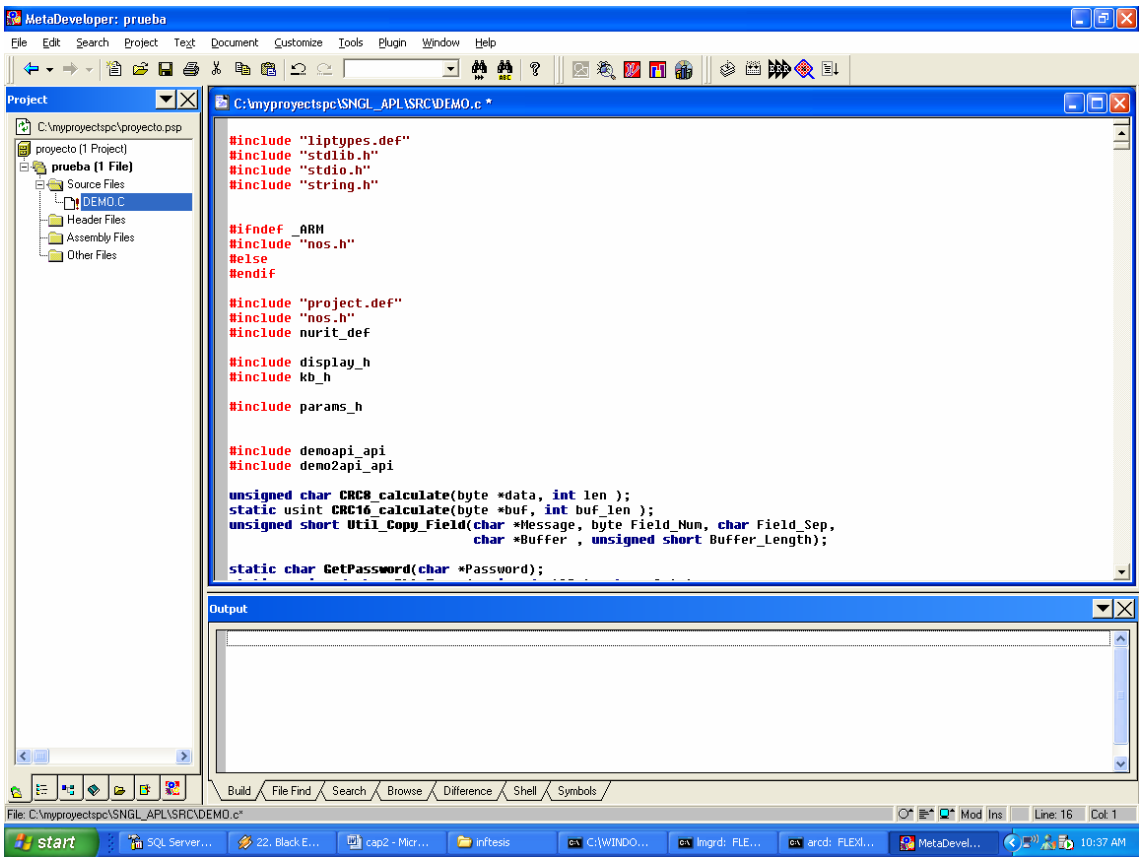
Menú	Submenú	Función
REGRESAR A APLICACIÓN		Al presionar *ENTER* en esta opción se inicia la aplicación cargada.
DESCARGA	auto descarga	Descarga automáticamente la aplicación desde la computadora.
	Parámetros de comunicación	Si se descarga por medio de serial o por MODEM y el número de teléfono así como velocidad de conexión.
	Copia IN	Prepara el dispositivo para copiar aplicación desde otra Terminal.
	Copia OUT	Prepara el dispositivo para descargar hacia otra Terminal.
	Carga manual	Se carga paso por paso la aplicación desde una computadora.
	Modo de operación	Se escoge entre simple o Mult.
INFORMACIÓN DEL PROGRAMA	Versión de aplicación	Despliega la versión de la aplicación cargada.
	NOS requerido	El tipo de sistema requerido para correr la aplicación.
	Nombre de archivo	Nombre de la aplicación.
	Fecha de aplicación	Fecha de la aplicación.
REINICIAR APLICACIÓN		Reinicia la aplicación.
HORA Y FECHA		Hora y fecha de la Terminal.
MENÚ DE SERVICIO	Prueba de hardware	Realiza una prueba a todo el hardware.
	Bitácora de errores	1. imprime. 2. limpia.
	Tipo de *ROM*	Indica el tipo y tamaño de memoria *ROM* que utiliza.
	estadísticas	Imprime y reinicia las estadísticas del Terminal.
TERMINAL ID		Muestra el número serial del Terminal.

2.1.2 Compilador Metaware

El compilador utilizado para la programación de las aplicaciones del punto de venta es Metaware C/C++ de la compañía Arc internacional (Fig. 8). Este software es especial para compilar aplicaciones para microprocesadores de tipo ARM el cual es el cerebro del punto de venta.

El compilador Metaware puede cargar archivos con extensión C, y compilarlos generando así un archivo tipo hex. El cual nos sirve para cargarlo al integrador de aplicaciones Nurit.

Figura 8. Ambiente de programación compilador Metaware



Para poder compilar una aplicación, primero se debe crear un proyecto con los siguientes pasos:

1. En el menú de proyecto se selecciona espacio de proyecto y se selecciona ‘nuevo’.

2. Se le asigna un nombre y se presiona el botón ‘*OK*’.

3. Una vez creado el proyecto se presiona la barra *member* y se presiona el botón de agregar proyecto a espacio de trabajo.

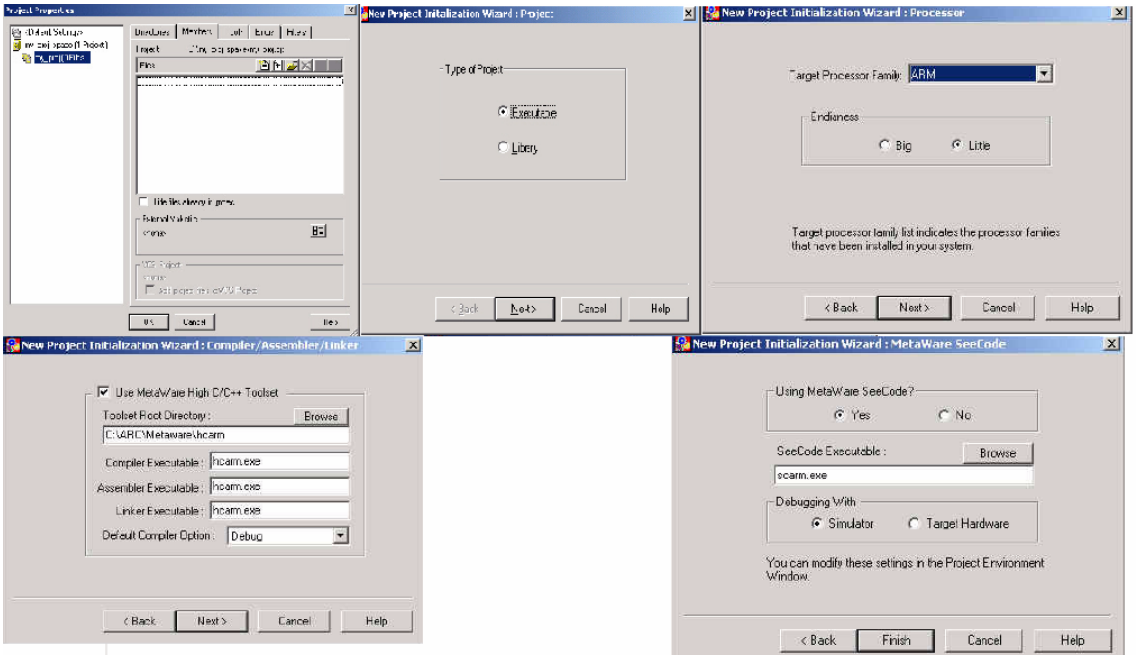
4. Seguido de esto aparecerá una ventana que indica si el proyecto será una librería o un ejecutable. Para nuestro fin se selecciona ejecutable.

5. En la siguiente pantalla se indica el procesador destino que se utilizara para correr la aplicación aquí se selecciona el procesador tipo ARM.

6. En la siguiente pantalla se selecciona que compilador se utilizara y se selecciona el Metaware c/c++.

Una vez terminado este proceso se prosigue a agregar archivos tipo C al proyecto y compilarlos para generar el código tipo hex.

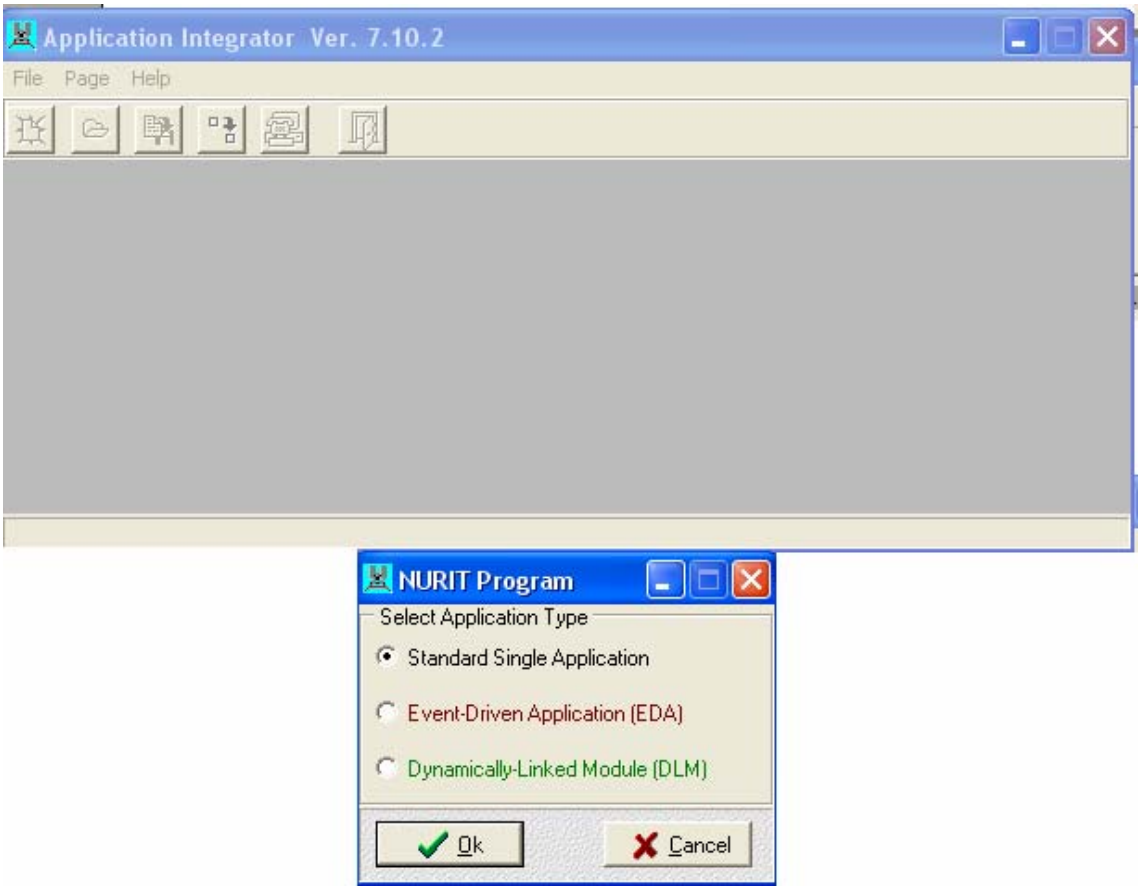
Figura 9. Pasos Para Creación De Proyecto



2.1.3 Integrador de aplicaciones NURIT

El integrador de aplicaciones Nurit (Fig. 10), es un software utilizado para descargar las aplicaciones del punto de venta desde la computadora. Este software crea la interfaz de conexión entre los terminales, se le carga un archivo tipo hex. El cual es convertido a tipo app y prepara el archivo para ser cargado en un punto de venta.

Figura 10. Integrador de aplicaciones

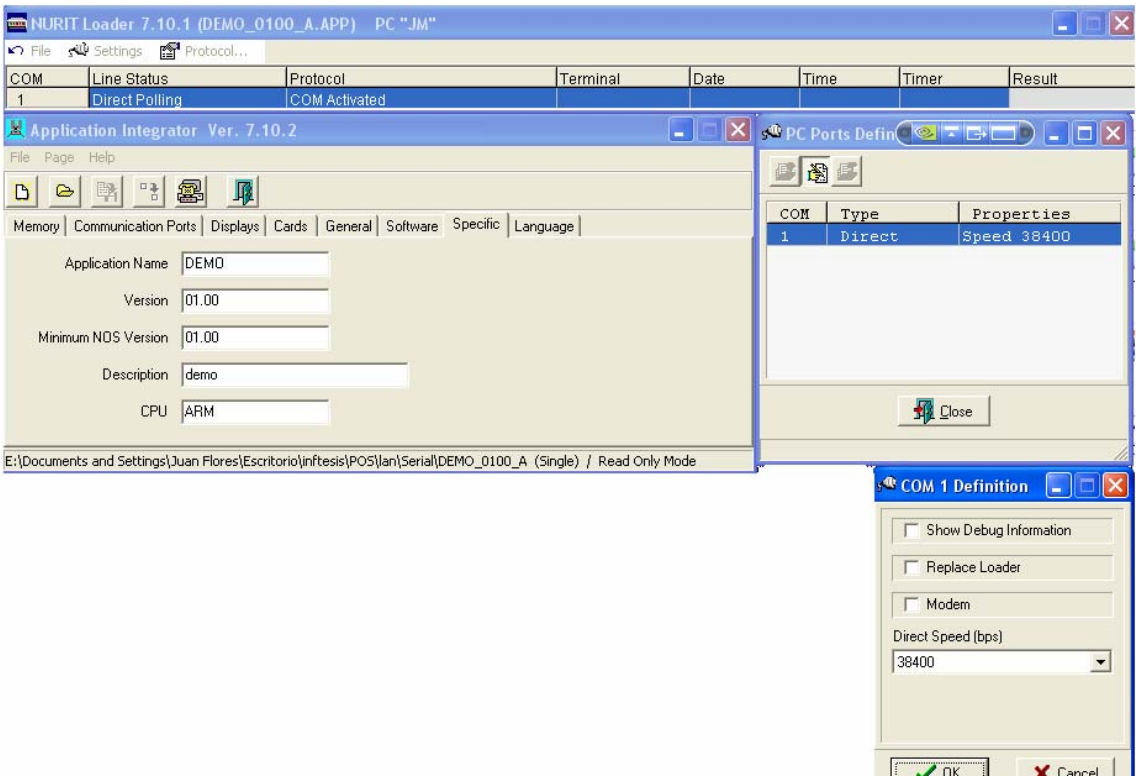


2.2 Carga del programa

Al momento de terminar la aplicación se procede a descargarla en el punto de venta, se utiliza el integrador de aplicaciones Nurit, se carga un archivo tipo hex y se genera un app. Después se configura el puerto serial para descargar el programa de la siguiente manera: se presiona el botón de iniciar comunicación, y aparecerá una pantalla el cual inicializa el puerto de comunicaciones 1 para transferir datos, para configurar la velocidad de transmisión a la que se descargaran los datos se presiona el botón de opciones y aquí se selecciona la

velocidad. Una vez configurado se prosigue a configurar el punto de venta para descargar la aplicación.

Figura 11. Configuración del integrador de aplicaciones



El punto de venta esta conectado con la computadora por medio del puerto serial RS-232, para poder configurar el punto de venta se debe ingresar al sistema operativo de este. Una vez ingresado se selecciona la opción descargar, dentro de este menú se configura el puerto por donde se conectara con la PC, se selecciona puerto serial y velocidad de transferencia de 38400, la cual es la velocidad máxima de conexión del punto de venta. Se vuelve a salir al menú de descarga y se prosigue a escoger descarga manual, aquí se selecciona cargar y empieza a descargar la información de la computadora a la

cual esta interconectado. Una vez descargada la aplicación se presiona el botón *ESC* hasta que se llegue al menú principal y ya aquí se selecciona la opción de correr la aplicación. Con esto se inicia el programa que se acaba de cargar.

El integrador de aplicaciones se puede utilizar para cargar el programa por medio del puerto serial, puerto *Ethernet* y MODEM.

2.3 Instalación de equipo

Para instalar el equipo de punto de venta se utilizarán las siguientes herramientas:

1. Bobina de cable telefónico para cablear desde toma telefónica hasta el lugar de instalación del punto de venta.

2. Ponchadora de cable telefónico para ensamblar los cables con los conectores RJ11.

3. Regletas para la conexión de la alimentación del pos a la línea eléctrica.

4. Cuando se instala el punto de venta se le programa el teléfono donde tiene que llamar para comunicarse con el contestador.

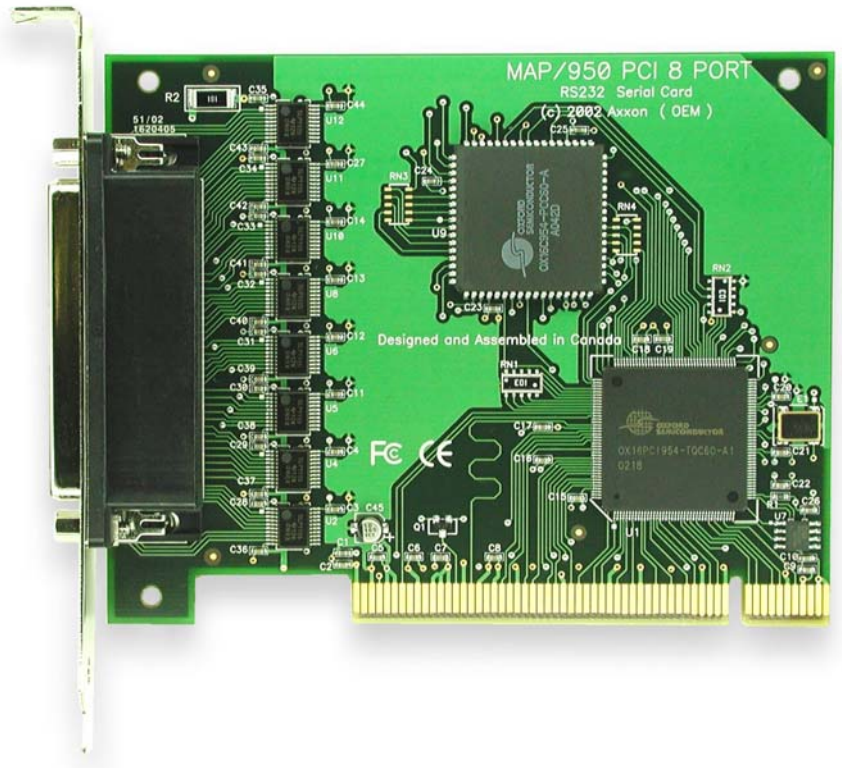
3. DESARROLLO CONTESTADOR

3.1 Descripción del equipo y software

3.1.1 Servidor Contestador

El servidor contestador es una computadora en la cual se instala el programa contestador y es utilizado únicamente para atender las peticiones de los puntos de ventas. Esta computadora debe ser lo suficientemente capaz de manejar las peticiones de varios MODEM a la vez, para esto se utilizó una tarjeta PCI para ampliación de 8 puertos seriales, en los cuales se conecto un MODEM a cada puerto. Esta tarjeta se puede observar en la figura 12.

Figura 12. Tarjeta PCI con 8 puertos seriales



El servidor también debe tener gran capacidad de procesamiento, es por eso que se uso uno con las siguientes especificaciones técnicas.

- CPU Intel Pentium 3.2 Ghz.
- 40 GB disco Duro.
- 2 GB memoria RAM.
- Tarjeta de red Lan 1 Gbps.

Una ves conectados los MODEM y instalado el programa contestador se debe crear un *ODBC* hacia el servidor de las ordenes.

El *ODBC* crea la comunicación con las bases de datos, para obtener datos o ingresar datos en las tablas. Para crear el *ODBC* se debe ingresar a panel del control. En la opción herramientas administrativas se encuentra la opción de creación de *ODBC*.

Cuando se ingresa a esta opción se indica el nombre de la base de datos a conectarse, y la dirección IP de la maquina servidora de las órdenes, se presiona *OK*, al hacer esto se debe ingresar el usuario y la clave que se tiene para hacer uso de la base de datos, después de ingresados se procede a realizar una prueba de comunicación entre las computadora y si resulta aprobada se crea el *ODBC*.

3.1.2 *C Builder*

Es un lenguaje de programación orientado a objetos con el cual se desarrollo la aplicación contestador. Orientado a objetos significa que no se programa un código lineal si no que cada objeto que forma parte del programa realiza una acción o bloque de programa especifico cuando ocurre un evento. Durante la creación de la aplicación se usaron pocos objetos ya que el principal es el de la comunicación serial el cual era el encargado de enviar y recibir los datos, la aplicación aparte de esto constaba de 3 objetos Memo para desplegar los datos recibidos desde el pos, los datos enviados hacia el pos y otro que indicaba si ocurría algún error en la comunicación. Se utilizo un botón el cual también es un objeto y es utilizado para limpiar la pantalla de los 3 memo.

Los demás se utilizarón para la interconexión con el *ODBC* creado para leer y escribir los datos en la base de datos.

3.1.3 MODEM

Los MODEM son dispositivos electrónicos utilizados para la interconexión de terminales o dispositivos. Usan la transmisión serial, existen MODEM internos y externos de voz y datos. Para la realización del proyecto se utilizaron MODEM externos, cada MODEM controla una línea de un PBX estos se conectan al puerto serial de la computadora específicamente al puerto RS-232, su capacidad de transmisión es de 34 Kbps y están configurados de manera que cuando entre una llamada la contesten de manera automática, el PBX al momento de entrar una segunda llamada se la dirige al siguiente MODEM que se encuentre disponible y así sucesivamente, teniendo la capacidad de contestar hasta 8 llamadas simultaneas.

El MODEM que se utilizó es de marca *USROBOTICS* (Fig. 13, 14) el cual puede comunicarse a 33.6/28.8 Kbps. Se puede observar en la figura

Figura 13. MODEM externo

Figura 14. MODEM externo

3.1.4 Hiperterminal

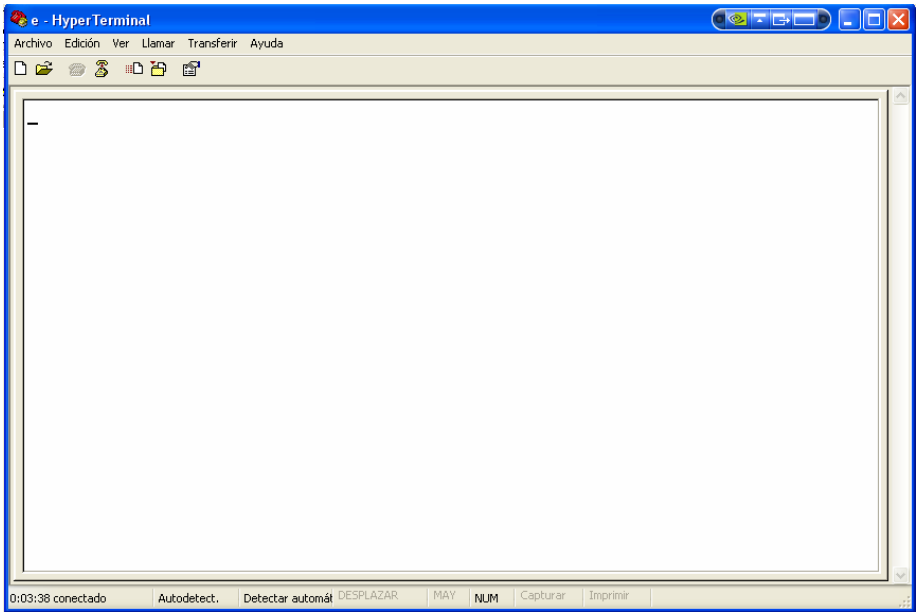
Es un programa de comunicaciones integrado en Windows el cual utiliza el estándar AT para comunicarse con dispositivos conectados a varios puertos de la computadora ya sea seriales, paralelos o MODEM (Fig. 15).

Cuando se conecta un equipo a algún puerto, primero se configura el Hiperterminal para que se pueda comunicar con este. Se escoge el puerto donde se encuentra conectado el equipo el cual puede ser el COM1, COM2, COM3, etc. Después se configura la velocidad de conexión que varía desde 110 bits por segundo a 115200 bits por segundo, aquí se puede configurar paridad en los datos y los bits de parada.

Después de hacer esto se entra en la consola donde se ingresan los comandos para programar el dispositivo. Si se ingresa el comando AT se debe ver un *OK* que es lo que responde el dispositivo y esto es indicador que existe comunicación entre ambos. Si se desean ver los comandos que se presionan se debe escribir el comando ATE. Hiperterminal se utiliza para configurar los registros internos de los dispositivos los que a su vez controlan el funcionamiento de estos.

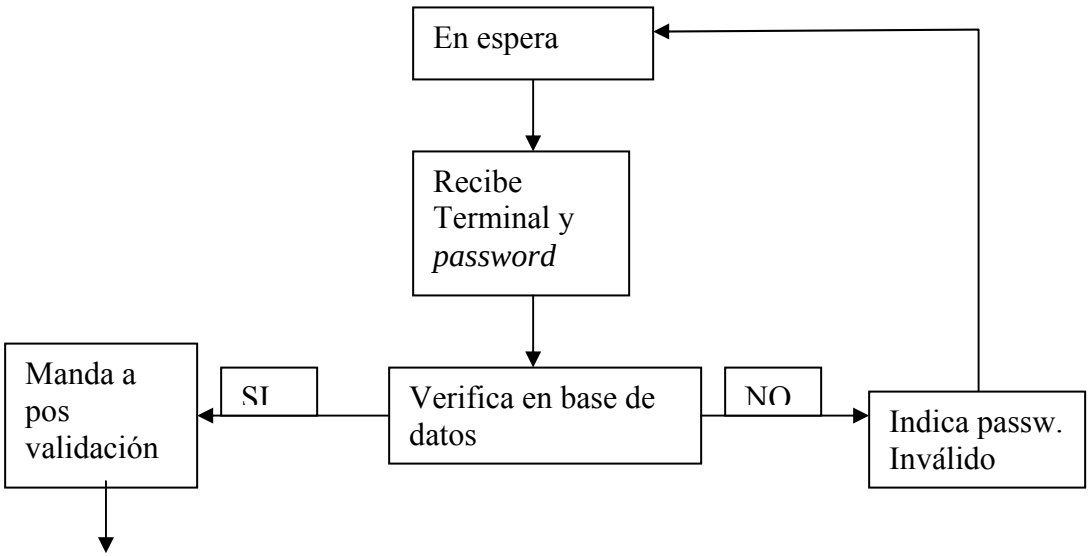
Al finalizar la programación de algún dispositivo se deben grabar los cambios en la memoria y esto se consigue al momento de programar al ingresar al final de los comandos W con esto se graba en la memoria flash.

Figura 15. Programa Hiperterminal

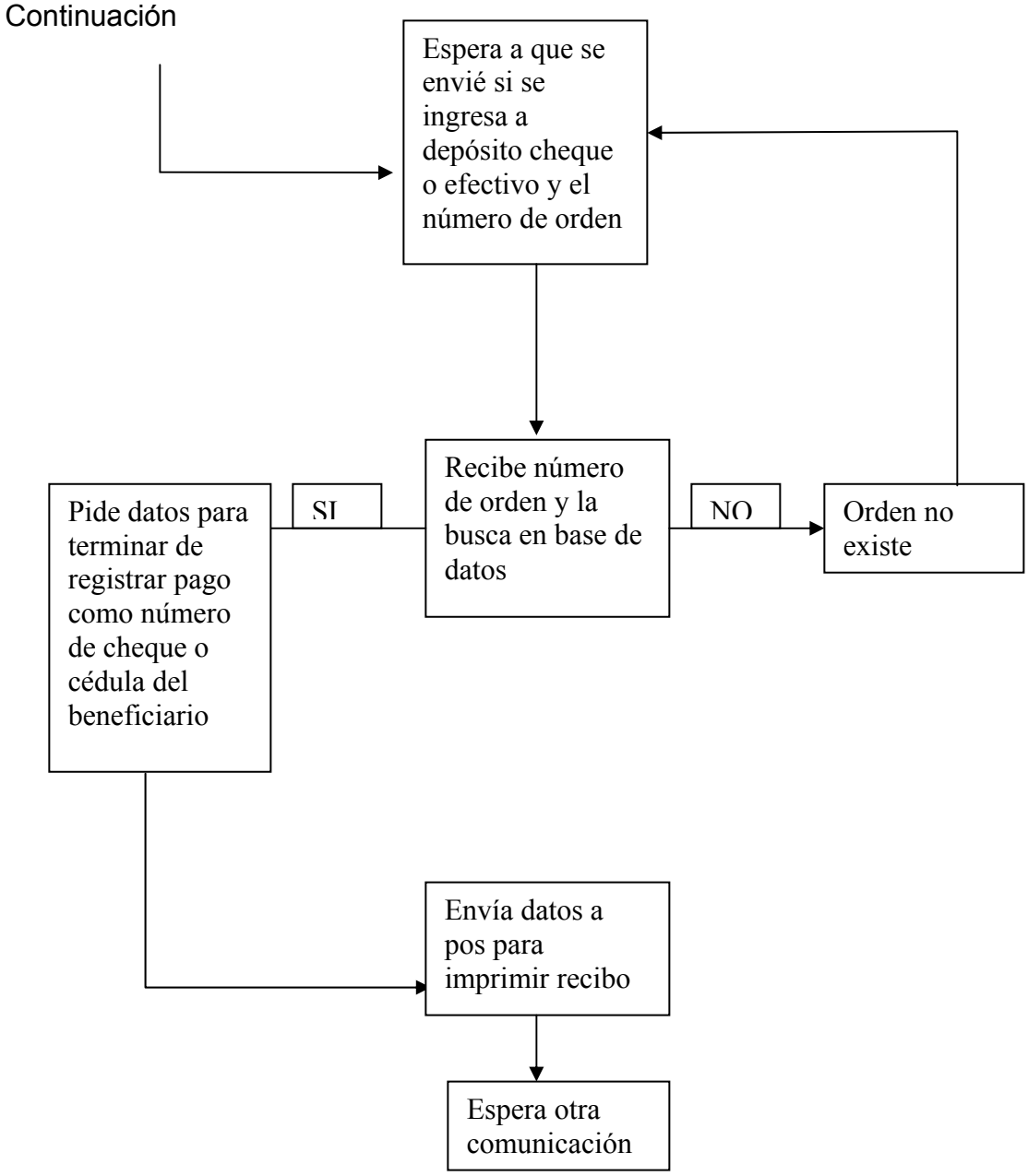


3.2 Diagrama de flujo del contestador

Figura 16. Diagrama de flujo del contestador



Continuación



3.3 Programación y configuración de los MÓDEMS

La programación de los MODEM se realiza en el programa de comunicación Hiperterminal y deben configurarse valores correctos de manera que se pueda comunicar con el punto de venta satisfactoriamente y evitar así perdida de datos en la recepción y/o transmisión de la información.

Para poder ingresar comandos AT hacia el MODEM, se debe poner en modo terminal la configuración del Hiperterminal, así los comandos ingresados son enviados directamente al MODEM.

Antes de cada comando se debe ingresar AT y presionar ENTER, los comandos básicos son A/ con el cual se re-ejecutan el último comando dado, no se necesita presionar AT o *ENTER*. A> se repite el último comando dado hasta que se cancela presionando cualquier tecla.

Los registros S son direcciones de memoria donde varios parámetros de tiempo, definiciones de caracteres ASCII y otras configuraciones son guardadas.

Inicialmente la configuración del registro S para cada *témplate* es el mismo. Pero se puede sobre escribir con la información de la *NVRAM*.

Para desplegar la información en los registros S y su función, se debe ingresar el siguiente comando mientras se esta en el modo terminal.

ATS$ (*ENTER*)

Para desplegar la configuración de los registros s en la NVRAM se hace con el comando

ATI5 (*ENTER*)

Para desplegar la configuración en RAM se hace con el comando

ATI4 (*ENTER*)

En ambos casos, los registros S aparecen como una tabla de 7 columnas de ancho, las entradas aparecen como número de registro = valor decimal.Para desplegar la configuración de solo un registro se usa

ATSr? Donde r es el registro a mostrar.

Para cambiar la configuración de un registro se utiliza

ATSr=n (*ENTER*)

Donde r es el número del registro y n es el valor decimal. Si se desea guardar el valor en memoria y que este no desaparezca durante la próxima reiniciada se debe seguir los comandos con un W.

Por ejemplo **ATS13=8&W (*ENTER*)**

Un comando alternativo para ingresar es Sr.=n.

El MODEM tiene 3 modos de operación, modo comando, modo en línea y modo en línea-comando. El modo comando es cuando el MODEM esta en modo comando y se puede controlar usando comando AT. El modo en línea es cuando esta conectado a otro dispositivo por medio de la línea telefónica. Y el modo en línea-comando es cuando se le ingresan comando por medio de la línea telefónica.

Para entrar en modo en línea-comando. Se entra +++. Dependiendo del BIT en que se encuentre el registro S 14 el MODEM estará ya sea en modo en línea o con la línea colgada.

El MODEM se configuró con los siguientes parámetros:

&A0 para que no despliegue los resultados ARQ.

&B1 el MODEM siempre se comunica con la computadora a la tasa a la cual se configuró sin importar de la tasa de conexión.

&C1 operación normal. El MODEM envía una señal de CD cuando se conecta a otro MODEM y la desactiva cuando se desconecta.

&D0 controla como el MODEM responde a señales de DTR desde la computadora.

&F0 no carga configuraciones de control de flujo.

&G0 no existe tono de guarda. Solo se utiliza en US y Canadá.

&H0 deshabilita el control de flujo de datos.

&I0 deshabilita Xon y Xoff. De los datos recibidos.

&K0 deshabilita la compresión de datos.

&M0 no existe control de errores. Esto nunca se recomienda para llamadas mayores de 2400 bps.

&N0 conexión de tasa cambiante. El MODEM negocia con el dispositivo remoto la máxima conexión posible.

&P0 llamada por medio de pulsos.

&R1 se ignora la señal RTS.

&S0 DSR esta siempre encendido.

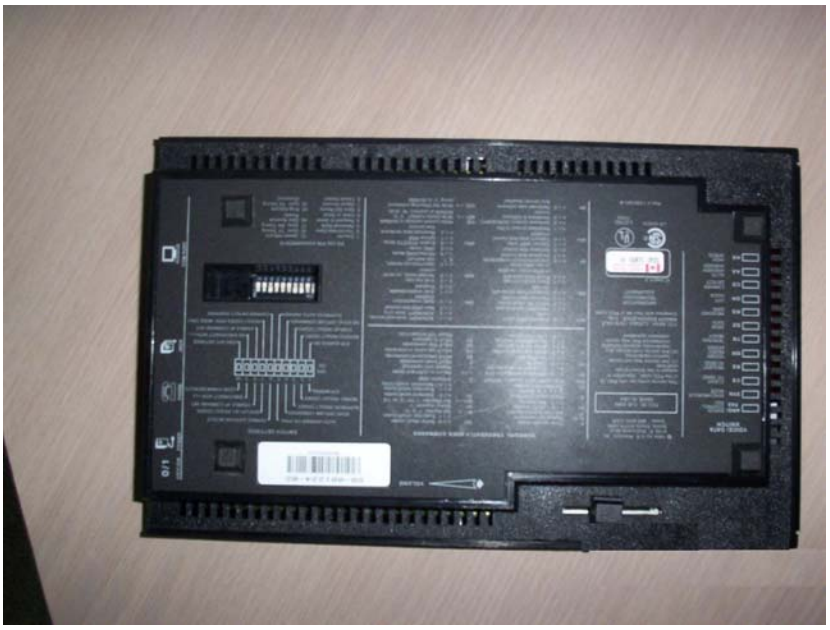
&T0 realiza prueba en el MODEM.

&X0 el MODEM envía tiempos de señalización al DTE por medio de la interfase serial.

&Y0 este comando permite enviar un alto a los datos sin necesidad de desconectar.

La posición de los interruptores DIP tiene mas prioridad que los valores de los registros internos del MODEM así que si la configuración de los DIP indican algo contrario a la configuración de los registros se debe guiar por lo que indican los interruptores.

Figura 17. Vista de los interruptores DIP



El MODEM puede cargar 3 configuraciones predeterminadas para el manejo de los errores y el flujo de los datos. Estas configuraciones se cargan con el comando AT&Fn&w. Donde n indica una de estas configuraciones.

La primera configuración provee un control del flujo de los datos por medio de hardware, si se utiliza esta configuración también se debe configurar el contestador para que utilice el control de flujo por hardware. La segunda es control de flujo por medio de software y la última utiliza el protocolo Xon/Xoff.

4. CREACIÓN DE BASE DE DATOS

4.1 Base de datos

El *SQL* (*Standar Query lenguage*) es un lenguaje estandarizado de base de datos, el cual nos permite realizar tablas y obtener datos de ella de manera muy sencilla. *SQL* es un lenguaje que consta de varias partes

- Lenguaje de definición de datos (DDL): Proporciona órdenes para definir esquemas de relación, eliminar relaciones, crear índices y modificar esquemas de relación.

- Lenguaje de manipulación de datos interactivos (DML): incluye un lenguaje de consultas que permite rescatar datos de las relaciones. También incluye órdenes para insertar, suprimir y modificar tuplas.

- Lenguaje de manipulación de datos inmerso (DML): La forma inmersa de *SQL* esta diseñada para usar dentro de los lenguajes de programación de lenguaje general.

- Definición de vistas (DDL): incluye órdenes para definir vistas.

Estructura Básica

La estructura básica de una expresión para consulta *SQL* consta de tres cláusulas:

- *SELECT*
- *FROM*
- *WHERE*

La cláusula *SELECT* se usa para listar los atributos que se desean en el resultado de una consulta. La cláusula *FROM* lista las relaciones que se van a examinar en la evaluación de la expresión. La cláusula *WHERE* costa de un predicado que implica atributos de las relaciones que aparecen en la cláusula *FROM.*

Una consulta básica en *SQL* tiene la forma:

SELECT A1, A2,..., An
FROM r1, r2,..., rn
WHERE P

Donde Ai = atributo (Campo de la tabla).
r1 = relación (Tabla).
P = predicado (condición).

El resultado de una consulta es por supuesto otra relación. Si se omite la cláusula *WHERE*, el predicado P es verdadero. La lista A1, A2,..., An puede sustituirse por un asterisco (*) para seleccionar todos los atributos de todas las relaciones que aparecen en la cláusula *FROM*.

Alias: Es posible renombrar los atributos y las relaciones, a veces por conveniencia y otras veces por ser necesario, para esto usamos la cláusula *AS*.

Cuando asociamos un alias con una relación decimos que creamos una variable de tupla. Estas variables de tuplas se definen en la cláusula *FROM* después del nombre de la relación.

En las consultas que contienen subconsultas, se aplica una regla de ámbito a las variables de tupla. En una subconsulta esta permitido usar solo variables de tupla definidas en la misma subconsulta o en cualquier consulta que tenga la subconsulta.

Predicados y conectores

Los conectores lógicos en *SQL* son:

- *AND*
- *OR*
- *NOT*

La lógica de estos conectores es igual que en cualquier lenguaje de programación y sirven para unir predicados.
Las operaciones aritméticas en *SQL* son:

- + (Suma)
- - (Resta)
- * (Multiplicación)
- / (División)

También incluye el operador de comparación *BETWEEN*, que se utiliza para valores comprendidos en un rango.

Operaciones de conjunto.

SQL incluye las operaciones de conjuntos *UNION, INTERSECT, MINUS,* que operan sobre relaciones y corresponden a las operaciones del álgebra unión, intersección y resta de conjuntos respectivamente. Para realizar esta operación de conjuntos debemos tener sumo cuidado que las relaciones tengan las mismas estructuras.

4.2 Tablas y vistas

Los datos son guardados en tablas donde estas contienen campos y los campos contienen registros.Los campos que componen una tabla pueden ser de diferentes tipos. De los que podemos mencionar los más utilizados.

Bit: Acepta valores 1 o 0 y es utilizada como un indicador de presencia o no.

Char: Acepta datos como cadena de caracteres como por ejemplo nombres.

Datetime: acepta datos como números en formato de fechas, por ejemplo 2/12/2005.

Int: acepta datos enteros como lo son los números.

Money: acepta datos en formato de dinero por ejemplo 22.33

Varchar: acepta datos en formato de cadenas de caracteres y números.

Tinyint: acepta datos en formato de números pero más limitado que *int* ya que este solo acepta números pequeños.

Smallmoney: acepta datos como *Money* solo que cantidades pequeñas.

La base de datos del PDV contiene 7 tablas y estas son:

- ÓRDENES_POS_GUA: En esta tabla se ingresan las órdenes a pagar por medio de pos.

- ÓRDENES_POS_EFECTIVO: En esta tabla se registran los pagos que se realizan por medio de efectivo.

- ÓRDENES_POS_CHEQUE: Esta tabla registrara los pagos que se realicen por medio de cheque.

- ÓRDENES_POS_DEPÓSITO: Aquí queda registrado los pagos realizados por depósitos.

- PAGO_POS_FINAL: todas las órdenes que ya han sido pagadas se ingresan a esta tabla.

- REVERSA_POS: En esta tabla quedan las reversas de los pagos hechas por el sistema del PDV.

- USUARIOS: aquí se registran los usuarios y las claves de acceso de estos. Esta tabla es solamente de lectura.

En estas tablas se almacenan los registros o pagos hechos por el PDV en sus distintas modalidades. Cada tabla de estas contiene diferentes campos en los cuales se almacenan los registros.

Vistas

Una vista es una representación lógica de subconjuntos de datos de una o más tablas. Pueden presentarse conjuntos lógicos de combinaciones de datos creando vistas de tablas.

La vista es una tabla lógica (no física) que se basa en una tabla o en otra vista. Una vista no contiene datos en si misma, es como una ventana a través de la cual pueden verse y cambiarse datos de tablas.

Actualmente el sistema de base de datos de PDV consiste en 4 vistas las cuales se encargan de mostrar las órdenes que son para el sistema de pago por medio de PDV y que serán pagadas por distintos formas.

Estas son las 4 vistas:

Órdenes_POS_Depósitos: Son las ordenes que serán pagadas por el PDV por medio de depósitos bancarios. Para poder ser pagadas por este medio tienen que tener estatus=5 y pago indicado=7.

Órdenes_POS_Efectivo: Son las órdenes que serán pagadas en efectivo por medio del POS. Para poder ser pagadas en efectivo la orden tiene que llevar un estatus igual a 5 y pago indicado = 6.

Órdenes_POS_cheque: Estas son las órdenes que serán pagadas en forma de cheque. Es el mismo caso de Órdenes_POS_Efectivo que serán pagadas si tienen el siguiente estatus= 5 y el pago indicado sea igual a 6.

Rev_pos: Estas son todas las órdenes pagadas por medio del sistema de PDV. En este caso las ordenes que ya han sido pagadas se les asignara un nuevo status que comprende a ser 2. Indicando que la rebaja de la orden ya se completo entonces nos indica que la orden ya ha sido pagada.

TRIGGERS

Los *triggers* son códigos de programa que se ejecutan cuando ocurre un evento. Por lo general se ejecutan cuando se insertan nuevos datos en una tabla o cuando se hace una actualización de un campo.

4.3 Llaves primarias

Las llaves primarias se ponen en el campo de número de orden y son utilizadas para 2 motivos, uno es que se usa para evitar que se ingresen 2 campos con el mismo valor, esto nos asegura que no existan ordenes repetidas en las tablas o que no se ingrese o se registre un pago de una orden la cual ya fue pagada.

La otra función de las llaves primarias es que al existir en un campo se hace más rápida la búsqueda de un registro al hacerlo por medio de llaves primarias.

Existen cuatro tipos de índices que podemos utilizar en *SQL*; de clave primaria, únicos, de texto completo, y ordinarios. Una clave primaria es un índice sobre uno o más campos donde cada valor es único y ninguno de los valores son nulos.

Para crear un índice de clave primaria tenemos básicamente dos opciones:

1. Crear el índice de clave primaria al momento de crear la tabla. En este caso se usa la opción *PRIMARY KEY* al final de la definición de los campos, con una lista de los campos que serán parte del índice.

CREATE TABLE nombre tabla (campo1 tipo dato, [campo2...,] *PRIMARY KEY* (campo1 [,campo2...]));

La palabra clave *NOT NULL* es obligatoria para un campo cuando éste vaya a formar parte de una clave primaria; como mencionamos anteriormente, las claves primarias no pueden contener valores nulos. Si intentamos crear una clave primaria sobre un campo nulo, *SQL* nos marcará un error.

2. Crear una clave primaria en una tabla existente con el uso del comando *ALTER TABLE*:

ALTER TABLE nombre_Tabla *ADD PRIMARY KEY*(campo1 [,campo2...]);

Por ejemplo, suponiendo que ya tenemos en nuestro sistema una tabla que fue creada de la siguiente manera (sin clave primaria, y con el campo id aceptando valores nulos):

CREATE TABLE usuarios(id int, nombre varchar(50), apellidos varchar(70));

Podemos crear una clave primaria sobre el campo id con esta sentencia:

ALTER TABLE usuarios MODIFY id INT NOT NULL, ADD PRIMARY KEY(id);

Las claves primarias pueden constar de más de un campo. Hay algunas veces en las que un solo campo no puede identificar de manera

4.4 Diseño y creación de la base de datos

Cuando se crean las tablas se harán con los siguientes campos. Y el contestador, al momento de insertar en una tabla los datos correspondientes se activara un *trigger* el cual realizará un bloque de código. Este código dependerá de si es pago en efectivo, en cheque o depósito. La estructura de este código es el siguiente.

Primero con el comando *DECLARE* se declaran las variables a usar, y se les debe poner el símbolo @ a las variables como en el siguiente ejemplo junto con el tamaño de la variable.

```
DECLARE

@Variable varchar (50)

@Variable2 datetime

@Variable3 varchar (25)

@Variable4 int

@Variable5 varchar(25)
```

Después se seleccionan los datos insertados en la tabla de la siguiente manera

```
SELECT      @variable = Codigo_Original,

            @variable2 = Fecha _ pago,

            @variable3 = Cedula,

FROM INSERTED
```

Y se seleccionan otros datos que se necesiten de otras tablas o vistas como por ejemplo datos que no se inserto el contestador en la tabla como por ejemplo datos de beneficiario y datos de remitente.

SELECT *@Variable4 = código beneficiario,*

@Variable5 = código remitente,

FROM Vista efectivo

Where codigooriginal=@variable1

Después se insertan los datos en la tabla de pagos finales de la siguiente manera.

INSERT INTO Pago_Pos_Final (código origina, fecha pago, cédula, código beneficiario, código remitente, monto, etc…..)

VALUES (@variable1, @variable2, @variable3, @variable4, @variable5, @variable6)

Y por último se realiza una actualización a la tabla principal de ordenes para regístrala como pagada.

UPDATE Sarf..Orden *SET* status =0 *WHERE* Número_Orden =@variable1

Y se repite para cheque y depósito solo que para estos 2 se trabajan con diferentes campos ya que se usan otros datos como por ejemplo número de cheque o número de cuenta.

La creación de los *triggers* las tablas y las vistas se pueden realizar en *sql enterprice manager* como se puede ven en la siguientes figuras donde se observa donde se localizan las vistas.

Figura 18. Vista de ambiente SQL

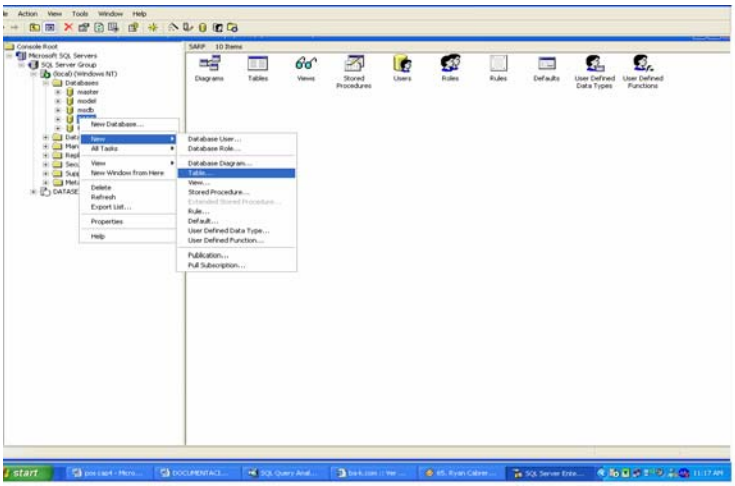
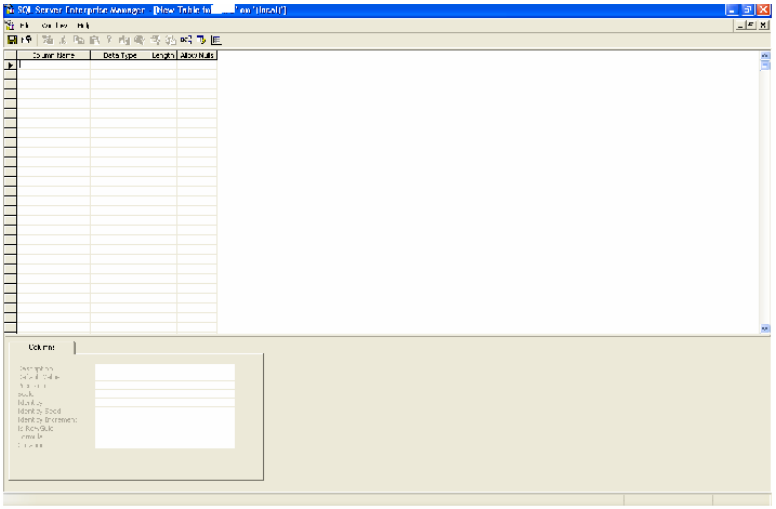


Figura 19. Creación de una tabla



4.4.1 Pagos en efectivo

Tabla III. Campos de tabla efectivo

Código_Original
Fecha_Pago
Cédula
ID_Usuario
TerminalID
Lugar
Sucursal

Código _ original: Como se había dicho antes este es el código de orden de la remesa.

Fecha _ pago: Se refiere a la fecha y hora del pago hecho en el PDV.

Cédula: Es el numero de identificación del beneficiario de la remesa.

Id _ usuario: Es la persona que se encargo de realizar la transacción de depósito bancario.

TerminalID: Es el numero de aparato o PDV en que se rebajo el pago. Esto normalmente esta unido al ID_Usuario pues si una persona que tenga un ID_Usuario desea ingresar y rebajar una orden en un POS que no tiene asignado simplemente no podrá realizar la operación. Aún cuando tenga un ID_Usuario.

Lugar: Es el lugar donde se esta pagando la orden del sistema por medio del PDV.

4.4.2 Pagos en cheque

Tabla IV. Campos de tabla cheque

Código _ original
Fecha _ pago
Número_Cheque
Cédula
ID_Usuario
TerminalID
Lugar
Agencia

Código_Original: Como se había dicho antes este es el código de orden de la remesa.

Fecha_Pago: Se refiere a la fecha y hora del pago hecho en el PDV.

Número _ cheque: Es el número de documento en que se realizó la transacción.

Cédula: Es el numero de identificación del beneficiario de la remesa.

ID_Usuario: Es la persona que se encargo de realizar la transacción de depósito bancario.

TerminalID: Es el numero de aparato o POS en que se rebajo el pago. Esto normalmente esta unido al ID_Usuario pues si una persona que tenga un ID_Usuario desea ingresar y rebajar una orden en un POS que no tiene asignado simplemente no podrá realizar la operación. Aún cuando tenga un ID_Usuario.

Lugar: Es el lugar donde se esta pagando la orden del sistema por medio del PDV.

4.4.3 Pago Depósitos

Tabla V. Campos de tabla depósitos

Código_Original
Fecha_Pago
Código_Banco
Número_Cuenta_Destino
Número _ cheque
Boleta
Número_Cuenta_Origen
TipoCuenta
Lugar
Certificación_Banco
Fecha_certificación
Agencia

Código_Original: Este es el código de orden de la remesa.

Fecha_Pago: Se refiere a la fecha y hora del pago hecho en el PDV.

Código_Banco: Este el código del banco en donde se hizo el depósito.

Número_Cuenta_Destino: El número de la cuenta en donde se depositara el dinero de la remesa en donde haya indicado el remitente.

Numero_Cheque: El número del documento con que se hizó el depósito.

ID_Usuario: Es la persona que se encargo de realizar la transacción de deposito bancario.

TerminalID: Es el número de aparato o PDV en que se rebajo el pago. Esto normalmente esta unido al ID_Usuario pues si una persona que tenga un ID_Usuario desea ingresar y rebajar una orden en un PDV que no tiene asignado simplemente no podrá realizar la operación. Aún cuando tenga un ID_Usuario.

Boleta: Es el número de la boleta de operación en que se realizó el depósito.

Número_Cuenta_Origen: Este campo no tiene utilidad relevante por el momento debido a que no todas las transacciones se realizan a partir de transferir dinero de una cuenta bancario a otra cuenta bancaria.

Tipo cuenta: Es el tipo de cuenta en donde se hizo el depósito de la remesa esta pude ser de Ahorro = A o Monetario = M.

Lugar: Es el lugar donde se esta rebajando la orden del sistema por medio del PDV.

Certificación _ banco: Es el número de certificación bancaria en que se realizo el depósito.

Fecha _ certificación: Es la fecha en que se realizó el depósito de la remesa.

Agencia: Es la sucursal bancaria en que se hizó el depósito.

4.4.4 Reversa

Tabla VI. Campos de tabla reversa

Número_Reversa
Código_Original
ID_Usuario

Número _ reversa: Es el número secuencial de la reversa hecha.

Código _ original: Es el código del que se desea reversar el pago.

ID_Usuario: Es el nombre de la persona autorizada para la reversa del pago hecho en el PDV.

5. ANÁLISIS ECONÓMICO

En el presente capítulo se realizará un análisis económico con el cual se obtendrá la rentabilidad del proyecto. Al realizar este proyecto no solo se busca una ganancia económica sino también se busca un beneficio social, de modo que la comunidad o comunidades donde se realiza tengan una mejor calidad de vida.

Se hará un calculo del costo y del beneficio por separado, para así poder concluir si el proyecto es rentable o no.

5.1 Costo

En la realización del proyecto se debe tomar en cuenta varios puntos para determinar el costo de este. Entre los costos se debe tomar el precio del equipo, mano de obra, costo por transacción, costo de instalación entre otros.

Tabla VII. Costos del proyecto

DESCRIPCIÓN	PRECIO UNIDAD	CANTIDAD	TOTAL
Punto de venta	Q 2400.00	30	Q 72,000.00
SDK Metaware	Q30000.00	1	Q30,000.00
Mano de obra	Q20000.00	1	Q20,000.00
Instalación	Q15.00	30	Q450.00
Teléfono	Q200.00	30	Q6,000.00
Gastos varios	Q1000.00	1	Q1,000.00
Mantenimiento anual	Q 10,000	1	Q10,000
TOTAL			Q139,450.00

5.2 Beneficio

El alto flujo de remesas en el país (US$3 mil millones) la gran mayoría de estas que son enviadas por medio de sistemas bancarios que equivale a un 30 % del total (alrededor US$ 900 millones). Ya que la gran mayoría de emigrantes ha salido del interior de la republica es aquí a donde se concentran los envíos de las remesas.

Con el proyecto se pretende captar parte de las remesas destinadas al sistema bancario, en las casas remesadoras americanas el método de envió

varía, en algunas cuando tienen promociones se cobra una cuota fija (US $8) por el envió de cualquier cantidad de dinero. Mientras que normalmente se cobra por cantidad de dinero enviada.

Algunas cuotas que se cobran por los envíos son

10-100 US $5

101-500 US $ 12

500-1000 US $ 15

De estas cuotas se obtiene una ganancia para la empresa del 40 % de lo cobrado por el envió. Es decir si se envía una cantidad de US $10 -$100 se obtiene una ganancia de US $2 en caso de ser US $1000 se obtiene una ganancia de US $6.

Si del dinero enviado se logra captar como mínimo un 10 % del mercado (US $90 millones) y este dinero se enviara por el sistema a pagarse por PDV. Si tomáramos como media de envió de dinero US $2000 Se obtendría una ganancia anual de:

90, 000,000/2,000 = 45000 órdenes

De las cuales se obtendrá una ganancia de US$ 6 por cada orden.

45000 * 6 = $270 000

Que sería la ganancia anual y considerando el cambio actual

US $270,000 * 7.7 = **Q 2, 079,000 anual.**

Considerando el costo del proyecto (Q139, 450) se obtiene una gran ganancia.

Otro beneficio que se obtiene es el beneficio social ya que este dinero que ingresa a las comunidades o pueblos sirve para que mejore la calidad de vida de los habitantes, pudiendo recibir una mejor educación y mejorando la vivienda ya que aquí es donde se invierte el dinero recibido

CONCLUSIONES

1. El uso de punto de ventas reduce grandemente los costos de implementación de un sistema para pago de remesas, ya que, integra varios dispositivos esenciales en uno solo.

2. Incentiva al ingreso de más remesas al país, al expandir una red que facilita el pago a los beneficiarios, el ingreso de más remesas permite que se desarrollen las comunidades creando un beneficio social.

3. El ingreso de remesas al país ayudan a mantener un equilibrio en la balanza de pagos del país, ya que, en la balanza económica ayuda a compensar el déficit y que este no sea mayor.

4. El uso de Puntos de venta disminuye el tiempo en que se realiza una transacción y es de fácil uso. Pudiéndose programar variedad de aplicaciones, como venta de pines telefónicos.

5. Se logra mantener un control de las órdenes al crear una base de datos en *SQL*.

6. Se logro crear una intercomunicación entre los dispositivos y sistemas, entre los MODEM y los servidores por medio del contestador.

RECOMENDACIONES

1. Estudiar la cantidad de remesas que se paguen por mes para calcular la rentabilidad del proyecto y su posible expansión.

2. En lugares donde no exista buena conexión telefónica o el servicio sea deficiente se puede proponer el uso de Puntos de venta que utilicen tecnología inalámbrica conocida como GPRS para el envió de las ordenes y realizar el pago.

3. Realizar pruebas de pago de remesas en varios puntos para medir los tiempos de respuestas del servidor y calcular el tiempo que se lleva en realizar, el pago, ya que, este debe realizarse en el menor tiempo posible ,máximo 1 minuto por orden.

4. Dejar habilitada la opción de actualización de software del punto de venta, en caso de que se modifique el aplicativo del punto de venta se pueda descargar a todos los puntos de venta desde la central remota.

5. Hacer mantenimiento preventivo del contestador y de los MODEM, para evitar su mal funcionamiento.

6. En caso de que la conexión, durante un pago, se quede colgada, reiniciar el servidor del contestador, a veces, el mismo sistema operativo detiene todas las conexiones y es necesario reiniciar el sistema.

BIBLIOGRAFÍA

1. Taub, H., Schilling, D.J. **Principles Of Communications Systems**. 2ª Edición. McGraw-Hill, Singapore, 1986.

2. Fuentes de León, Frisly Alejandro. **Implementacion y Utilizacion De Redes De Cajeros Automaticos En Institucion De Servicio Financiero De Guatemala**. 1984. tesis.

3. Cevallos, Javier. **Curso De Programacion C++ Orientada a Objetos**. 1993 Editorial Iberoamericana. 774 Pág.

4. Groff, james. **SQL Complete Reference Second Edition**. 1992. Mcgraw-Hill. 1992. 1025 Pág.

5. Us. Robotics. Courier V.everything Command Reference.1996. 100 Pág.

6. **Migraciones Internacionales Y remesas.** Banco Mundial. http://web.worldbank.org/WBSITE/EXTERNAL/NEWS/0,,contentMDK:208762~pagePK:64257043~piPK:437376~theSitePK:4607,00.html.

7. **Base de Datos.** http://es.wikipedia.org/wiki/Base_de_datos#Tipos_de_bases_de_datos.

8. **Comandos AT**. http://www.sixnetio.com/htmlhelps/vtmodem/5a5a318.htm

9. **C Builder Basics**. http://www.temporaldoorway.com/programming/cbuilder/basics/index.htm

10. **Hardware Puerto serial.** http://www.softio.com/contactus.htm

APÉNDICE

CÓDIGO UTILIZADO

CÓDIGO DEL PUNTO DE VENTA

```
// LIBRERIAS A USAR
#include "liptypes.def"
#include "stdlib.h"
#include "stdio.h"
#include "string.h"
#ifndef _ARM
# proyect.def
#include "nos.h"
//*****************************************************************************
// DECLARACIÓN DE VARIABLES Y FUNCIONES
unsigned char CRC8_calculate(byte *data, int len );
static usint CRC16_calculate(byte *buf, int buf_len );
unsigned short Util_Copy_Field(char *Message, byte Field_Num, char
Field_Sep, char *Buffer , unsigned short Buffer_Length);
static char GetPassword(char *Password);
static unsigned char Pkt_Trans(unsigned ctlByte,char *Data);
static void Go_To_Menu(void* GoToMenu);
static void SynchPassword(char *PasswordToSend);
static void Consulta(void);
static void PagoFinal(void);
static void PagoInicial(void);
static void PagoEfectivoFinal(void);
static void PagoEfectivoInicial(void);
static void PagoDepositoFinal(void);
static void PagoDepositoInicial(void);
static void ReversaFinal(void);
static void ReversaInicial(void);
static void DisplayLogo(sint FrameNumber);
static void PrintLogo(void);
```

```
//*****************************************************************************

//CREACIÓN DEL MENÚ PRINCIPAL Y SUBMENÚS
static const entry PagoEntries[] = {{"Efectivo" ,
0,(void*)PagoEfectivoInicial,(void*)0},
                                    {"Cheque", 0,(void*)PagoInicial,(void*)0},};

static const menu PagoSubMenu =
{"PAGO",2,(entry*)PagoEntries,NUMBERS_SELECT|MENU_TIMEOUT_1_MIN}
;


static const entry MainEntries[] = {{"Pago",
0,(void*)Go_To_Menu,(void*)&PagoSubMenu},
                                {"Consulta" , 0,(void*)Consulta,(void*)0},
                                {"Reversa" , 0,(void*)ReversaInicial,(void*)0},
                                {"Deposito" , 0,(void*)PagoDepositoInicial,(void*)0}};

static const menu MainMenu =
{4,(entry*)MainEntries,NUMBERS_SELECT|MENU_TIMEOUT_1_MIN};

//*****************************************************************************
//DEFINICIÓN DE VARIABLES
static char
Cedula[20],Cheque[20],Boleta[20],CuentaD[20],BancoD[20],TipoCuenta[20],Rev
ersa[20],Certificacion[20],FechaCert[20],ConfirmaCertifica[20];

#ifdef _ARM

//VOID PRINCIPAL
void main (void)
void AplMain_Entry (void)
{
 static char LocalPassword[17];
 static sint FrameNo;
 static ulint TimeStart;
 memset(LocalPassword,0,sizeof(LocalPassword));
 FrameNo = 1;

//DESPLIEGA EL LOGO MIENTRAS NO SE PRESIONE ENTER
 TimeStart = Scan_MillSec();
 DisplayLogo(FrameNo);
 for(;;)
 {
```

```
  if(Kb_Read() == MENU)
  {
)
// SE LLAMA FUNCION PARA CAPTAR TECLAS DE PASSWORD

  memset(LocalPassword,0,sizeof(LocalPassword));
   if(GetPassword(LocalPassword) != MENU   {
    switch(atoi(LocalPassword))
    {
     case 1 :
     break;

     default :
     Formater_GoMenu((menu *)&MainMenu,MENU_MODE);
     SynchPassword(LocalPassword);
     break;
    }
   }
  }

  /****Velocidad logo****/
  if((Scan_MillSec() - TimeStart) >= 35)
  {
   FrameNo++;
   if(FrameNo > 28)
    FrameNo = 1;

   DisplayLogo(FrameNo);
   TimeStart = Scan_MillSec();
  }
  /****Velocidad logo end****/

 }/* Main for end*/
}/*Main end*/

/*******************************************************************************************/


static void Go_To_Menu(void* GoToMenu)
{
  Formater_GoMenu((menu*)GoToMenu,MENU_MODE);
}
```

```
//*****************************************************************************
// FUNCIÓN QUE CAPTA TECLAS DE CLAVE

static char GetPassword(char *Password)
{
  char  KeyStroke = 0x00;
  char  PassMask[17];
  sint  CharPosition = 0;

  Display_ClrDsp();

  Display_FormatWrite(DSPL1,1,0x20,"CLAVE DE ACCESO");
  memset(Password,0,sizeof(Password));
  memset(PassMask,0,sizeof(PassMask));

  while(KeyStroke != ENTER && KeyStroke != MENU)
  {
      KeyStroke = Kb_WaitForKey();

      if(   KeyStroke != ALPHA && KeyStroke != ENTER
         && KeyStroke != MENU && KeyStroke != 'A'
         && KeyStroke != 'B'  && KeyStroke != 'C'
         && KeyStroke != 'D'  && KeyStroke != 'F'
         && KeyStroke != 'X')
      {
        if(CharPosition < 17)
        {
         PassMask[CharPosition] = '*';
         Password[CharPosition++] = KeyStroke;
        }
      }

      Display_FormatWrite(DSPL1,2,0,"                ");
      Display_FormatWrite(DSPL1,2,0x20,PassMask);

  }
  return(KeyStroke);
}
//*****************************************************************************


//FUNCIÓN QUE LLENA LOS BUFFERS CON LOS  DATOS SINCRONIZADOS
Y LOS ENVIA
```

```
static unsigned char Pkt_Trans(unsigned ctlByte,char *Data)
{
 static unsigned char Buffer[261],an,sn,BCD;
 static char FieldBuffer[255],PrintBuffer[255];
 static ulint TimerStart;
 static usint ByteCount,CrC16;
 static real_time_clock localtime;
 Buffer[0] = 0x01;
 switch(ctlByte & 0xF0)
 {
  CrC16 = CRC16_calculate((byte*)&Buffer[4],Buffer[2]);
  Buffer[Buffer[2]+4] = CrC16 & 0xFF;
  Buffer[Buffer[2]+5] = CrC16 >> 8;
  Uart_SendBuf(UART_A,(byte*)Buffer,Buffer[2]+6);
  break;

}
{
   TimerStart = Scan_MillSec();
   while((ByteCount += Uart_GetBuf(UART_A,(byte*)&Buffer[ByteCount],
         4-ByteCount)) < 4 && Kb_Read() != MENU
         && Scan_MillSec() - TimerStart <=60000);
   if(CRC8_calculate(&Buffer[1],3) == 0)
   {
    switch(Buffer[1] & 0xF0)
    {
     case 0xC0 :
     if((0x01 & (((Buffer[1] & 0x04)>>2) - (1%2))) == 0)
     {
      sn = 0x01 & (((Buffer[1] & 0x04)>>2));
      an = 0x01 & (((Buffer[1] & 0x08)>>3) - (1%2));
      /*send terminal id*/
      Pkt_Trans(0xC0,Data);
     }
     else
     {
      Pkt_Trans(0x80,"");
     }
     break;

     case 0x40 :
     if((0x01 & (((Buffer[1] & 0x04)>>2) - (1%2))) == sn)
     {
```

```
sn = 0x01 & (((Buffer[1] & 0x04)>>2));
an = 0x01 & (((Buffer[1] & 0x08)>>3) - (1%2));
if(Buffer[2] > 0)
{
 while((ByteCount += Uart_GetBuf(UART_A,(byte*)&Buffer[ByteCount],
      (Buffer[2]+6)-ByteCount)) < (Buffer[2]+6) &&
       Kb_Read() != MENU && Scan_MillSec() - TimerStart <= 60000);
 if(CRC16_calculate((byte*)&Buffer[4],Buffer[2]+2) == 0)
 {
  memset(FieldBuffer,0,sizeof(FieldBuffer));
  Util_Copy_Field((char*)&Buffer[4],0,'|',FieldBuffer,255);
```

//RECIBE RESPUESTA DEL CONTESTADOR Y SI LA CLAVE ES VALIDO SE VA AL CASE 22, CASE 23 ES USADO PARA REGRESAR MENSAJES DE INVALIDO Y DESPLEGARLOS EN PANTALLA.

```
  switch(atoi(FieldBuffer))
  {
   case 21 :
   Util_Copy_Field((char*)&Buffer[4],1,'|',FieldBuffer,255);
   Util_AsciiToBcd(FieldBuffer,&BCD,strlen(FieldBuffer));
   localtime.hours   = BCD;
   Util_Copy_Field((char*)&Buffer[4],2,'|',FieldBuffer,255);
   Util_AsciiToBcd(FieldBuffer,&BCD,strlen(FieldBuffer));
   localtime.minutes = BCD;
   Util_Copy_Field((char*)&Buffer[4],3,'|',FieldBuffer,255);
   Util_AsciiToBcd(FieldBuffer,&BCD,strlen(FieldBuffer));
   localtime.seconds = BCD;
   Util_Copy_Field((char*)&Buffer[4],4,'|',FieldBuffer,255);
   Util_AsciiToBcd(FieldBuffer,&BCD,strlen(FieldBuffer));
   localtime.day = BCD;
   Util_Copy_Field((char*)&Buffer[4],5,'|',FieldBuffer,255);
   Util_AsciiToBcd(FieldBuffer,&BCD,strlen(FieldBuffer));
   localtime.year = BCD;
   Rtc_SetTime(&localtime);
   Rtc_SetDate(&localtime);
   break;
```

// SI LA CLAVE ES CORRECTA ENTRA AQUI Y DESPLIEGA EL MEŃÚ PRINCIPAL

```
   case 22 :
   Formater_GoMenu((menu *)&MainMenu,MENU_MODE);
   break;
```

```
//        SI LA CLAVE ES INCORRECTA EL CONTESTADOR DEVUELVE UN VALOR PARA ENTRAR AQUI Y SE DESPLIEGAN MENSAJES DE ERROR ENVIADOS POR EL CONTESTADOR.
        case 23 :
        Display_ClrDsp();
        Util_Copy_Field((char*)&Buffer[4],1,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,1,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],2,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,2,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],3,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,3,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],4,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,4,0x20,FieldBuffer);
        Kb_WaitForKey();
        break;

//SI SE ENVIO EL NUMERO DE ORDEN DE PAGO CON CHEQUE EL CONTESTADOR ENVIA LA VALIDACION CON 24 Y AL RECIBIRLO EL POS INGRESA A ESTE CASE EL CUAL MANDA A LLAMAR LA FUNCION QUE PIDE CÉDULA Y  NUMERO DE CHEQUE


        case 24 :
        PagoFinal();
        break;

//DESPUES QUE EL POS ENVIA LOS DATOS DE CHEQUE, EL CONTESTADOR REGISTRA EL PAGO Y DEVUELVE LOS DATOS REQUERIDOS PARA IMPRIMIR RECIBO DE CHEQUE  E IMPRIME EL RECIBO.
        case 25 :
        Printer_WaitForTextReady();
        memset(PrintBuffer,0,sizeof(PrintBuffer));
        Display_ClrDsp();
        Util_Copy_Field((char*)&Buffer[4],1,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,1,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],2,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,2,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],3,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,3,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],4,'|',FieldBuffer,255);
```

```
Display_FormatWrite(DSPL1,4,0x20,FieldBuffer);
PrintLogo ();
Printer_WriteStr("    EMPRESA REMESADORA\r");
Printer_WriteStr("    Tel: \r\r\r");
Printer_WriteStr("Clave :\r");
Util_Copy_Field((char*)&Buffer[4],8,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Beneficiario:\r");
Util_Copy_Field((char*)&Buffer[4],5,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Remitente:\r");
Util_Copy_Field((char*)&Buffer[4],6,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
/*Printer_WriteStr("Fecha Ingreso: \r");
Util_Copy_Field((char*)&Buffer[4],8,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);*/
Printer_WriteStr("Monto US$: ");
Util_Copy_Field((char*)&Buffer[4],10,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Cambio US$: ");
Util_Copy_Field((char*)&Buffer[4],7,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Monto GTQ: ");
Util_Copy_Field((char*)&Buffer[4],9,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Banco: \r");
sprintf(PrintBuffer,"Cheque No.: %s\r\r",Cheque);
Printer_WriteStr(PrintBuffer);
Util_Copy_Field((char*)&Buffer[4],11,'|',FieldBuffer,255);
sprintf(PrintBuffer,"Fecha Pago: %s \r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Util_Copy_Field((char*)&Buffer[4],12,'|',FieldBuffer,255);
sprintf(PrintBuffer,"Hora Pago: %s \r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
sprintf(PrintBuffer,"Identificacion:\r%s\r\r\r\r\r\r\r",Cedula);
```

```
//SI SE ENVIO EL NUMERO DE ORDEN DE PAGO CON DEPOSITO EL
CONTESTADOR ENVIA LA VALIDACION CON 26 Y AL RECIBIRLO EL POS
INGRESA A ESTE CASE EL CUAL MANDA A LLAMAR LA FUNCION QUE
PIDE LOS DATOS PARA HACER EFECTIVO EL DEPÓSITO.

        case 26 :
        PagoDepositoFinal();
        break;


//  CUANDO EL CONTESTADOR RECIBE LOS DATOS DE DEPOSITO Y LO
REGISTRA, ENVIA DE REGRESO AL CONTESTADOR LOS DATOS
NECESARIOS PARA IMPRIMIR EL RECIBO Y ESTOS ENTRAN AQUI .
    case 27:

        Printer_WaitForTextReady();
        memset(PrintBuffer,0,sizeof(PrintBuffer));
        Display_ClrDsp();
        Util_Copy_Field((char*)&Buffer[4],1,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,1,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],2,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,2,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],3,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,3,0x20,FieldBuffer);
        Util_Copy_Field((char*)&Buffer[4],4,'|',FieldBuffer,255);
        Display_FormatWrite(DSPL1,4,0x20,FieldBuffer);
        PrintLogo ();
        Printer_WriteStr("    EMPRESA REMESADORA\r");
        Printer_WriteStr("    Tel: \r\r\r");
        Printer_WriteStr("Clave :\r");
        Util_Copy_Field((char*)&Buffer[4],8,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Printer_WriteStr("Beneficiario:\r");
        Util_Copy_Field((char*)&Buffer[4],5,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"%s\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Printer_WriteStr("Remitente:\r");
        Util_Copy_Field((char*)&Buffer[4],6,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Printer_WriteStr("Monto GTQ: ");
        Util_Copy_Field((char*)&Buffer[4],9,'|',FieldBuffer,255);
```

```
        sprintf(PrintBuffer,"%s\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Printer_WriteStr("Cambio US$: ");
        Util_Copy_Field((char*)&Buffer[4],7,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"%s\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Printer_WriteStr("Monto US$: ");
        Util_Copy_Field((char*)&Buffer[4],10,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Printer_WriteStr("Convenio: DEPOSITO\r");
        Printer_WriteStr("Lugar Pago: \r\r");
        Printer_WriteStr("Banco Origen: \r");
        sprintf(PrintBuffer,"Cheque No.: %s\r\r",Cheque);
        Printer_WriteStr(PrintBuffer);
        Printer_WriteStr("Banco Destino: \r");
        Util_Copy_Field((char*)&Buffer[4],11,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"%s\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        sprintf(PrintBuffer,"Cuenta Destino: \r%s\r",CuentaD);
        Printer_WriteStr(PrintBuffer);
        sprintf(PrintBuffer,"Boleta: %s\r",Boleta);
        Printer_WriteStr(PrintBuffer);
        Util_Copy_Field((char*)&Buffer[4],13,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"Certificacion:\r%s\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Util_Copy_Field((char*)&Buffer[4],14,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"Fecha Certificacion:\r%s\r\r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        Util_Copy_Field((char*)&Buffer[4],15,'|',FieldBuffer,255);
        sprintf(PrintBuffer,"Fecha Pago: %s \r",FieldBuffer);
        Printer_WriteStr(PrintBuffer);
        AplMain_Entry();
        break;
//EL CONTESTADOR AL VERIFICAR SI UN CÓDIGO PUEDE SER
REVERSADO ENVIA LA VALIDACIÓN AL CASE 28, AQUI SE LE PIDE QUE
INGRESE DE NUEVO EL CÓDIGO.

        case 28 :
        ReversaFinal();
        break;

//RECIBE LOS DATOS DEL CONTESTADOR PARA IMPRIMIR EL RECIBO DE
REVERSA.
```

```
case 29:

 Printer_WaitForTextReady();
 memset(PrintBuffer,0,sizeof(PrintBuffer));
 Display_ClrDsp();
 Util_Copy_Field((char*)&Buffer[4],1,'|',FieldBuffer,255);
 Display_FormatWrite(DSPL1,1,0x20,FieldBuffer);
 Util_Copy_Field((char*)&Buffer[4],2,'|',FieldBuffer,255);
 Display_FormatWrite(DSPL1,2,0x20,FieldBuffer);
 Util_Copy_Field((char*)&Buffer[4],3,'|',FieldBuffer,255);
 Display_FormatWrite(DSPL1,3,0x20,FieldBuffer);
 Util_Copy_Field((char*)&Buffer[4],4,'|',FieldBuffer,255);
 Display_FormatWrite(DSPL1,4,0x20,FieldBuffer);
 PrintLogo ();
 Printer_WriteStr("    EMPRESA REMESADORA\r");
 Printer_WriteStr("    Tel: \r\r\r");
 Printer_WriteStr("    SOLICITUD DE\r");
 Printer_WriteStr("   REVERSA DE PAGO\r\r\r");
 Util_Copy_Field((char*)&Buffer[4],7,'|',FieldBuffer,255);
 sprintf(PrintBuffer,"Fecha Reversa:\r%s\r",FieldBuffer);
 Printer_WriteStr(PrintBuffer);
 Util_Copy_Field((char*)&Buffer[4],8,'|',FieldBuffer,255);
 sprintf(PrintBuffer,"Hora Reversa:\r%s\r\r\r\r\r\r\r\r",FieldBuffer);
 Printer_WriteStr(PrintBuffer);
 sprintf(PrintBuffer,"Usuario:\r%s \r\r\r\r\r\r\r\r",FieldBuffer);
 Printer_WriteStr(PrintBuffer);  */
 Printer_WriteStr("F:_____\r\r");
 Printer_WriteStr("1.Firma del solicitante \r");
 AplMain_Entry();
 break;
```

// CUANDO SE ENVIA EL CODIGO DE PAGO EN EFECTIVO Y ES VALIDADO POR EL CONTESTADOR, SE ENTRA AL CASE 30. DONDE SE PIDE CEDULA DE BENEFICIARIO

```
 case 30 :
 PagoEfectivoFinal();
 break;
```

// SE RECIBEN LOS DATOS DESDE EL CONTESTADOR PARA IMPRIMIR EL RECIBO DE EFECTIVO.

```
 case 31 :
 Printer_WaitForTextReady();
 memset(PrintBuffer,0,sizeof(PrintBuffer));
```

```
Display_ClrDsp();
Util_Copy_Field((char*)&Buffer[4],1,'|',FieldBuffer,255);
Display_FormatWrite(DSPL1,1,0x20,FieldBuffer);
Util_Copy_Field((char*)&Buffer[4],2,'|',FieldBuffer,255);
Display_FormatWrite(DSPL1,2,0x20,FieldBuffer);
Util_Copy_Field((char*)&Buffer[4],3,'|',FieldBuffer,255);
Display_FormatWrite(DSPL1,3,0x20,FieldBuffer);
Util_Copy_Field((char*)&Buffer[4],4,'|',FieldBuffer,255);
Display_FormatWrite(DSPL1,4,0x20,FieldBuffer);
PrintLogo ();
Printer_WriteStr( EMPRESA REMESADORA\r");
Printer_WriteStr("    Tel: 2328-0800\r\r\r");
Printer_WriteStr("Clave Inmediata:\r");
Util_Copy_Field((char*)&Buffer[4],8,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Beneficiario:\r");
Util_Copy_Field((char*)&Buffer[4],5,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Remitente:\r");
Util_Copy_Field((char*)&Buffer[4],6,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Monto GTQ: ");
Util_Copy_Field((char*)&Buffer[4],9,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Cambio US$: ");
Util_Copy_Field((char*)&Buffer[4],7,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Printer_WriteStr("Monto US$: ");
Util_Copy_Field((char*)&Buffer[4],10,'|',FieldBuffer,255);
sprintf(PrintBuffer,"%s\r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Util_Copy_Field((char*)&Buffer[4],11,'|',FieldBuffer,255);
sprintf(PrintBuffer,"Fecha Pago: %s\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
Util_Copy_Field((char*)&Buffer[4],12,'|',FieldBuffer,255);
sprintf(PrintBuffer,"Hora Pago: %s\r\r",FieldBuffer);
Printer_WriteStr(PrintBuffer);
sprintf(PrintBuffer,"Cedula: %s\r\r\r\r\r\r\r",Cedula);
Printer_WriteStr(PrintBuffer);
```

```
          Printer_WriteStr("F:_____\r\r");
          AplMain_Entry();
          break;
         default :
          memset(Buffer,0,sizeof(Buffer));
          break;
        }
       }
      }
      return(Buffer[1]);
     }
     else
     {
      sn = 0x01 & (((Buffer[1] & 0x04)>>2));
      an = 0x01 & (((Buffer[1] & 0x08)>>3) - (1%2));
      Pkt_Trans(0x40,Data);
     }
     break;

     default : break;
}
 return(Buffer[1]);

}
//*****************************************************************************
// FUNCIÓN QUE  ENVIA EL PASSWORD INGRESADO HACIA EL
CONTESTADOR

static void SynchPassword(char *PasswordToSend)
{
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255],Password[16];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
// INICIALIZA PUERTO DE TRANSMISIÓN
 Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
 memset(DataSend,0,sizeof(DataSend));
 memset(Terminal,0,sizeof(Terminal));
 memset(Password,0,sizeof(Password));

//OBTIENE EL NUMERO DE TERMINAL DEL PUNTO DE VENTA
 strcpy(Terminal,"01|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
```

```
 Rtc_Read(&TrnTime);
// ENVIA NUMERO DE TERMINAL SEGUIDO DE LA CLAVE
 sprintf(DataSend,"%s%02x/%02x|",Terminal,TrnTime.hours,
        TrnTime.minutes, TrnTime.seconds, TrnTime.day, TrnTime.month, and
TrnTime.year);
//LLAMA A FUNCIÓN QUE TRANSMITE
 if(Pkt_Trans(0x80,DataSend) == 0)
  {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
   Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  else
  {
   sprintf(DataSend,"2002|%s|",PasswordToSend);
   Pkt_Trans(0x40,DataSend);
  }
}


*****************************************************************************************
****************
//SI SE QUIERE HACER UNA CONSULTA, DESDE EL MENÚ PRINCIPAL SE
ENTRA AQUI

Static void Consulta(void)
{
 static dialoge CodigoDialog;
 static char StrCodigoDialog[25];
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 CodigoDialog.header = "Codigo Original";
 CodigoDialog.format = (void*)StrCodigoDialog;
 CodigoDialog.format_flags = __STRING;
 CodigoDialog.length = 25;
 CodigoDialog.return_value = 0;
 Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
 memset(DataSend,0,sizeof(DataSend));
 memset(Terminal,0,sizeof(Terminal));
```

```
 memset(StrCodigoDialog,0,sizeof(StrCodigoDialog));
 strcpy(Terminal,"01|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
 Rtc_Read(&TrnTime);
intf(DataSend,"%s|%02x:%02x:%02x|%02x/%02x/%02x|",Terminal,TrnTime.hou
rs,rnTime.minutes,TrnTime.seconds,TrnTime.day,TrnTime.month,TrnTime.year)
Display_ClrDsp();
 Display_FormatWrite(DSPL1,1,0x20,"");
 strcpy(StrCodigoDialog,"- - -");
 if(Formater_DialogeBox((void*)&CodigoDialog) == ENTER)
 {
  if(Pkt_Trans(0x80,DataSend) == 0)
  {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
   Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  else
  {
   sprintf(DataSend,"2003|CIPI-%s|",StrCodigoDialog);
   Pkt_Trans(0x40,DataSend);
  }/*0x80 end*/
 }
 else
 {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CODIGO NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}

*****************************************************************************************
*****************
// AL SELECCIONAR PAGO CHEQUE EN EL MENU PRINCIPAL SE LLAMA A
ESTA FUNCION QUE ENVIA AL CONTESTADOR  EL CODIGO DE REMESA
EN OPCIÓN  CHEQUE A PAGAR.
static void PagoInicial(void)
{
 static dialoge CodigoDialog;
 static char StrCodigoDialog[25];
```

```
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 CodigoDialog.header = "Codigo Original";
 CodigoDialog.format = (void*)StrCodigoDialog;
 CodigoDialog.format_flags = __STRING;
 CodigoDialog.length = 25;
 CodigoDialog.return_value = 0;
Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
 memset(DataSend,0,sizeof(DataSend));
 memset(Terminal,0,sizeof(Terminal));
 memset(StrCodigoDialog,0,sizeof(StrCodigoDialog));
 strcpy(Terminal,"01|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
 Rtc_Read(&TrnTime);
rintf(DataSend,"%s|%02x:%02x:%02x|%02x/%02x/%02x|",Terminal,TrnTime.ho
urs,rnTime.minutes,TrnTime.seconds,TrnTime.day,TrnTime.month,TrnTime.yea
r);
 Display_ClrDsp();
 Display_FormatWrite(DSPL1,1,0x20,"");
 strcpy(StrCodigoDialog,"- - -");
 if(Formater_DialogeBox((void*)&CodigoDialog) == ENTER)
 {
  if(Pkt_Trans(0x80,DataSend) == 0)
  {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
   Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  else
  {
   sprintf(DataSend,"04|CIPI-%s|",StrCodigoDialog);
   Pkt_Trans(0x40,DataSend);
  }/*0x80 end*/
 }
 else
 {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CODIGO NO");
```

```
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}
*****************************************************************************************
***************
//CUANDO ES APROBADO EL CÓDIGO DE LA ORDEN , SE PIDE NÚMERO
DE IDENTIFICACIÓN Y DE CHEQUE CON QUE SE PAGA Y SE ENVIA AL
CONTESTADOR

static void PagoFinal(void)
{
 static dialoge CedulaDialog,ChequeDialog;
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 CedulaDialog.header = "Identificacion";
 CedulaDialog.format = (void*)Cedula;
 CedulaDialog.format_flags = __STRING;
 CedulaDialog.length = 25;
 CedulaDialog.return_value = 0;
 ChequeDialog.header = "Cheque";
 ChequeDialog.format = (void*)Cheque;
 ChequeDialog.format_flags = __STRING;
 ChequeDialog.length = 25;
 ChequeDialog.return_value = 0;
 Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
sprintf(DataSend,"%s|%02x:%02x:%02x|%02x/%02x/%02x|",Terminal,TrnTime.
hours,
 TrnTime.minutes,TrnTime.seconds,TrnTime.day,TrnTime.month,TrnTime.year);
 Display_ClrDsp();
 if(Formater_DialogeBox((void*)&CedulaDialog) == ENTER)
 {
  Display_ClrDsp();
  if(Formater_DialogeBox((void*)&ChequeDialog) == ENTER)
  {
    if(Pkt_Trans(0x80,DataSend) == 0)
    {
     Display_ClrDsp();
     Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
     Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
```

```
     Kb_WaitForKey();
     AplMain_Entry();
    }
    else
    {
     sprintf(DataSend,"05|%s|%s|",Cedula,Cheque);
     Pkt_Trans(0x40,DataSend);
    }/*0x80 end*/
  else
  {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CHEQUE NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();
  }
 }
 else
 {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"ID NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}

*****************************************************************************************
**************
// AL SELECCIONAR EN EL MENÚ PRINCIPAL LA OPCIÓN DE PAGO EFECTIVO SE LLAMA A ESTA FUNCIÓN .AQUÍ SE INICIALIZA EL BUFFER DE ENVIO DE DATOS Y SE ENVÍA EL CÓDIGO DE REMESA A PAGAR
static void PagoEfectivoInicial(void)
{
 static dialoge CodigoDialog;
 static char StrCodigoDialog[25];
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 CodigoDialog.header = "Codigo Original";
 CodigoDialog.format = (void*)StrCodigoDialog;
 CodigoDialog.format_flags = __STRING;
 CodigoDialog.length = 25;
```

```
 CodigoDialog.return_value = 0;
 Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
 memset(DataSend,0,sizeof(DataSend));
 memset(Terminal,0,sizeof(Terminal));
 memset(StrCodigoDialog,0,sizeof(StrCodigoDialog));
 strcpy(Terminal,"01|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
 Rtc_Read(&TrnTime);
printf(DataSend,"%s|%02x:%02x:%02x|%02x/%02x/%02x|",Terminal,TrnTime.h
ours,
 rnTime.minutes,TrnTime.seconds,TrnTime.day,TrnTime.month,TrnTime.year);
 Display_ClrDsp();
 Display_FormatWrite(DSPL1,1,0x20,"");
 strcpy(StrCodigoDialog,"- - -");
 if(Formater_DialogeBox((void*)&CodigoDialog) == ENTER)
 {
  if(Pkt_Trans(0x80,DataSend) == 0)
  {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
   Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  else
  {
   sprintf(DataSend,"10|%s|",StrCodigoDialog);
   Pkt_Trans(0x40,DataSend);
  }/*0x80 end*/
 }
 else
 {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CODIGO NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();


//*****************************************************************************************
*****************
```

```
//CUANDO SE VERIFICA EL NÚMERO DE ORDEN EN EFECTIVO, EL
CONTESTADOR VALIDA, Y DESPUES EL POS ENVIA EL NÚMERO DE
IDENTIFICACION HACIA EL CONTESTADOR PARA REGISTRAR EL PAGO.
static void PagoEfectivoFinal(void)
{
 static dialoge CedulaDialog;
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 CedulaDialog.header = "Identificacion";
 CedulaDialog.format = (void*)Cedula;
 CedulaDialog.format_flags = __STRING;
 CedulaDialog.length = 25;
 CedulaDialog.return_value = 0;
Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
 memset(DataSend,0,sizeof(DataSend));
 memset(Terminal,0,sizeof(Terminal));
 memset(Cedula,0,sizeof(Cedula));
strcpy(Terminal,"01|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
 Rtc_Read(&TrnTime);
printf(DataSend,"%s|%02x:%02x:%02x|%02x/%02x/%02x|",Terminal,TrnTime.h
oursrnTime.minutes,TrnTime.seconds,TrnTime.day,TrnTime.month,TrnTime.ye
ar);
 Display_ClrDsp();
 if(Formater_DialogeBox((void*)&CedulaDialog) == ENTER)
 {
   if(Pkt_Trans(0x80,DataSend) == 0)
    {
     Display_ClrDsp();
     Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
     Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
     Kb_WaitForKey();
     AplMain_Entry();
    }
    else
    {
     sprintf(DataSend,"11|%s|",Cedula);
     Pkt_Trans(0x40,DataSend);
    }/*0x80 end*/
else
 {
```

```
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"ID NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}
*****************************************************************************************
******************
//CUANDO EN EL MENÚ PRINCIPAL SE SELECCIONA PAGAR CON
DEPÓSITO SE INGRESA EL CÓDIGO Y SE ENVIA CON ESTA FUNCION.

static void PagoDepositoInicial(void)
{
 static dialoge CodigoDialog;
 static char StrCodigoDialog[25];
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 CodigoDialog.header = "Codigo Original";
 CodigoDialog.format = (void*)StrCodigoDialog;
 CodigoDialog.format_flags = __STRING;
 CodigoDialog.length = 25;
 CodigoDialog.return_value = 0;
 Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
 memset(DataSend,0,sizeof(DataSend));
 memset(Terminal,0,sizeof(Terminal));
 memset(StrCodigoDialog,0,sizeof(StrCodigoDialog));
y(Terminal,"01|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
 Rtc_Read(&TrnTime);
printf(DataSend,"%s|%02x:%02x:%02x|%02x/%02x/%02x|",Terminal,TrnTime.h
ours,TrnTime.minutes,TrnTime.seconds,TrnTime.day,TrnTime.month,TrnTime.y
ear);
 Display_ClrDsp();
 Display_FormatWrite(DSPL1,1,0x20,"");
 strcpy(StrCodigoDialog,"- - -");
 if(Formater_DialogeBox((void*)&CodigoDialog) == ENTER)
 {
  if(Pkt_Trans(0x80,DataSend) == 0)
  {
   Display_ClrDsp();
```

```
   Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
   Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  else
  {
   sprintf(DataSend,"2006|CIPI-%s|",StrCodigoDialog);
   Pkt_Trans(0x40,DataSend);
  }/*0x80 end*/
 }
 else
 {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CODIGO NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}
//*****************************************************************************************
*****************
//CUANDO SE VERIFICA EL NÚMERO DE ORDEN DE PAGO DEPÓSITO,
ENTRA A ESTA FUNCIÓN Y SE ENVIAN LOS DATOS RESTANTES PARA
HACER EFECTIVO EL PAGO DEL DEPÓSITO
static void PagoDepositoFinal(void)
{ static dialoge
BoletaDialog,ChequeDialog,CuentaDDialog,BancoDDialog,TipoCuentaDialog,
            CertificacionDialog,FechaCertDialog,ConfirmaCertificaDialog;
 static char StrFechaCertDialog[25];
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 BoletaDialog.header = "Boleta";
 BoletaDialog.format = (void*)Boleta;
 BoletaDialog.format_flags = __STRING;
 BoletaDialog.length = 25;
 BoletaDialog.return_value = 0;
 ChequeDialog.header = "Cheque";
 ChequeDialog.format = (void*)Cheque;
 ChequeDialog.format_flags = __STRING;
 ChequeDialog.length = 25;
 ChequeDialog.return_value = 0;
```

```
 CuentaDDialog.header = "Cuenta";
 CuentaDDialog.format = (void*)CuentaD;
 CuentaDDialog.format_flags = __STRING;
 CuentaDDialog.length = 25;
 CuentaDDialog.return_value = 0;
 BancoDDialog.header = "Banco";
 BancoDDialog.format = (void*)BancoD;
 BancoDDialog.format_flags = __STRING;
 BancoDDialog.length = 25;
 BancoDDialog.return_value = 0;
 TipoCuentaDialog.header = "Tipo Cuenta";
 TipoCuentaDialog.format = (void*)TipoCuenta;
 TipoCuentaDialog.format_flags = __STRING;
 TipoCuentaDialog.length = 25;
 TipoCuentaDialog.return_value = 0;
 ConfirmaCertificaDialog.header = "Certificacion";
 ConfirmaCertificaDialog.format = (void*)ConfirmaCertifica;
 ConfirmaCertificaDialog.format_flags = __STRING;
 ConfirmaCertificaDialog.length = 25;
 ConfirmaCertificaDialog.return_value = 0;
 CertificacionDialog.header = "Confirme";
 CertificacionDialog.format = (void*)Certificacion;
 CertificacionDialog.format_flags = __STRING;
 CertificacionDialog.length = 25;
 CertificacionDialog.return_value = 0;
 FechaCertDialog.header = "Fecha Certifica";
 FechaCertDialog.format = (void*)StrFechaCertDialog;
 FechaCertDialog.format_flags = __STRING;
 FechaCertDialog.length = 25;
 FechaCertDialog.return_value = 0;
Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
Display_ClrDsp();
 if(Formater_DialogeBox((void*)&BoletaDialog) == ENTER)
 {  Display_ClrDsp();
  if(Formater_DialogeBox((void*)&ChequeDialog) == ENTER)
  {   Display_ClrDsp();
   if(Formater_DialogeBox((void*)&CuentaDDialog) == ENTER)
   {    Display_ClrDsp();
    if(Formater_DialogeBox((void*)&BancoDDialog) == ENTER)
    {     Display_ClrDsp();
     if(Formater_DialogeBox((void*)&TipoCuentaDialog) == ENTER)
     {     Display_ClrDsp();
     if(Formater_DialogeBox((void*)&ConfirmaCertificaDialog) == ENTER)
```

```
    {     Display_ClrDsp();
    if(Formater_DialogeBox((void*)&CertificacionDialog) == ENTER)
    {     Display_ClrDsp();
     if(Formater_DialogeBox((void*)&FechaCertDialog) == ENTER)
     {     if(Pkt_Trans(0x80,DataSend) == 0)
     {     Display_ClrDsp();
     Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
     Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
     Kb_WaitForKey();
     AplMain_Entry();
    }
    else
    {

sprintf(DataSend,"07|s|%s|%s|%s|%s|",Boleta,Cheque,CuentaD,BancoD,TipoC
uenta,ConfirmaCertifica,Certificacion,StrFechaCertDialog);
     Pkt_Trans(0x40,DataSend);
    }/*0x80 end*/
  }
  else
  {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"FECHA NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  else
  {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CONFIRMACION NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  }
  else
  {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CERTIFICACION NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  }
  else
  {   Display_ClrDsp();
```

```
   Display_FormatWrite(DSPL1,2,0x20,"TIPO CUENTA NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
  } }
  else
  {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"BANCO NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();
  } }
 else {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CUENTA NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
  } } else {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CHEQUE");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();
 }}
else {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"BOLETA NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}
//*****************************************************************************************
********** //CUANDO SE QUIERE HACER UNA REVERSA EN EL MENÚ PRINCIPAL SE ESCOGE REVERSA SE LLAMA A ESTA FUNCIÓN, EN DONDE SE INGRESA EL CÓDIGO Y  LO ENVIA AL CONTESTADOR.

static void ReversaInicial(void)
{ static dialoge CodigoDialog;
 static char StrCodigoDialog[25];
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 CodigoDialog.header = "Codigo Original";
 CodigoDialog.format = (void*)StrCodigoDialog;
```

```
 CodigoDialog.format_flags = __STRING;
 CodigoDialog.length = 25;
 CodigoDialog.return_value = 0;
 Uart_Init(UART_A,&CommParams);
 Uart_Flush(UART_A);
 memset(DataSend,0,sizeof(DataSend));
 memset(Terminal,0,sizeof(Terminal));
 memset(StrCodigoDialog,0,sizeof(StrCodigoDialog));
 strcpy(Terminal,"01|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
Display_ClrDsp();
 Display_FormatWrite(DSPL1,1,0x20,"");
 strcpy(StrCodigoDialog,"- - -");
 if(Formater_DialogeBox((void*)&CodigoDialog) == ENTER)
 {
  if(Pkt_Trans(0x80,DataSend) == 0)
  {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
   Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
   Kb_WaitForKey();
   AplMain_Entry();
  }
  else
  {
   sprintf(DataSend,"2008|%s|",StrCodigoDialog);
   Pkt_Trans(0x40,DataSend);
  }/*0x80 end*/
 }
 else
 {
   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"CODIGO NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADO");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}
//*****************************************************************************************
*****************
//SE ENVIA EL CÓDIGO CONFIRMADO PARA REVERSA. HACIA EL
CONTESTADOR.

static void ReversaFinal(void)
```

```
{ static ialoge ReversaDialog;
 static char StrReversaDialog[25];
 comm_param  CommParams = {38400,8,'n',FALSE,TRUE};
 static char Terminal[25],DataSend[255];
 static signed char *BoardAddr;
 static real_time_clock TrnTime;
 ReversaDialog.header = "Confirma Codigo";
 ReversaDialog.format = (void*)StrReversaDialog;
  memset(StrReversaDialog,0,sizeof(StrReversaDialog));
 strcpy(Terminal,"2001|");
 memcpy(&Terminal[5],ApplMngr_GetTerminalID(&BoardAddr),16);
 Rtc_Read(&TrnTime);
printf(DataSend,"%s|%02x:%02x:%02x|%02x/%02x/%02x|",Terminal,TrnTime.h
ours,
 TrnTime.minutes,TrnTime.seconds,TrnTime.day,TrnTime.month,TrnTime.year);
 Display_ClrDsp();
 Display_FormatWrite(DSPL1,1,0x20,"CIPI-");
 strcpy(StrReversaDialog,"- - -");
 Display_ClrDsp();
 if(Formater_DialogeBox((void*)&ReversaDialog) == ENTER)
 {   if(Pkt_Trans(0x80,DataSend) == 0)
    {     Display_ClrDsp();
     Display_FormatWrite(DSPL1,2,0x20,"ERROR AL");
     Display_FormatWrite(DSPL1,3,0x20,"CONECTAR");
     Kb_WaitForKey();
     AplMain_Entry();
    }    else    {
     sprintf(DataSend,"09|%s|",Reversa);
     Pkt_Trans(0x40,DataSend);
    }/*0x80 end*/
  }
 else
 {   Display_ClrDsp();
   Display_FormatWrite(DSPL1,2,0x20,"REVERSA NO");
   Display_FormatWrite(DSPL1,3,0x20,"INGRESADA");
   Kb_WaitForKey();
   AplMain_Entry();
 }
}
```

CÓDIGO DEL CONTESTADOR

```
//---------------------------------------------------------------------
//DECLARACIÓN DE LIBRERIAS

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "stdio.h"
#define ETX 0x03
#pragma package(smart_init)
#pragma link "nrclasses"
#pragma link "nrcomm"
#pragma link "nrlogfile"
#pragma resource "*.dfm"
TForm1 *Form1;

unsigned char CRC8_calculate(byte *data, int len );
static unsigned short CRC16_calculate(byte *buf, int buf_len );
unsigned short Util_Copy_Field(char *Message, byte Field_Num, char
Field_Sep,
char *Buffer , unsigned short Buffer_Length);

//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
//---------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{ nrComm1->Active = true;
}
//---------------------------------------------------------------------
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{ nrComm1->Active = false;
}
//---------------------------------------------------------------------
char Data[255];
void __fastcall TForm1::nrComm1AfterReceive(TObject *Com, Pointer Buffer,
     DWORD Received)
```

```
{ static String StrBuffer =
"",DisplayStr,TerminalID,Codigo,Cedula,Cheque,Passwd,Boleta,CuentaD,Banco
D,TipoCuenta,Certificacion,FechaCert;
 static unsigned char State = 0x00,an,sn,SendBuffer[261];
 static unsigned short CrC16;
 static String Usuario;
 static String Lugar;
try
{    StrBuffer.c_str()[3] = CRC8_calculate(&StrBuffer.c_str()[1],2);
     nrComm1->SendData(StrBuffer.c_str(),4);
     State = 0; StrBuffer = "";
     sn = 0; an = 0;
     break;
     case 0xC0 :
     if((0x01 & (((StrBuffer.c_str()[1] & 0x04)>>2) - (1%2))) == sn)
     {
      sn = 0x01 & (((StrBuffer.c_str()[1] & 0x04)>>2));
      an = 0x01 & (((StrBuffer.c_str()[1] & 0x08)>>3) - (1%2));
      Memo1->Lines->Add("comunicacion establecida");
      if(StrBuffer.c_str()[2] > 0)
      {       Memo1->Lines->Add("<--------DATA/ACK PKT");
       State = 2;
      }      else
      {
       Memo1->Lines->Add("<--------ACK PKT");
       State = 0;
       StrBuffer = "";
      }    }
     else    {      State = 0;
      StrBuffer = "";
     }     break;

     case 0x40 :
     if((0x01 & (((StrBuffer.c_str()[1] & 0x04)>>2) - (1%2))) == sn)
     {
      sn = 0x01 & (((StrBuffer.c_str()[1] & 0x04)>>2));
      an = 0x01 & (((StrBuffer.c_str()[1] & 0x08)>>3) - (1%2));
      if(StrBuffer.c_str()[2] > 0)
      {       State = 2;
      }
      else     {       Memo1->Lines->Add("<--------ACK PKT");
       State = 0;
       StrBuffer = "";
      }    }
```

```
   else     {
    State = 0;
    StrBuffer = "";
   }
   break;
   default  :
   State = 0;
   StrBuffer ="";
   break;
  }
 }
 else
 {
  if(StrBuffer.Length() >= 4)
  {     Memo1->Lines->Add("[error de cabecera]");
   State = 0;
   for(int j = 0; j <= 4;j++)
   {      Memo1->Lines->Add("[error de sincronia]");
      else
    {       if(j == 4)
     {
       StrBuffer = "";
       State = 0;
    }   }   }    }   }
 break;
 case 2 :
 StrBuffer += PChar(Buffer)[i];
 if(StrBuffer.Length() == StrBuffer.c_str()[2]+6  &&
 CRC16_calculate((byte*)&StrBuffer.c_str()[4],StrBuffer.c_str()[2]+2) == 0)
 {
  Memo1->Lines->Add("[ OK]");
  Memo1->Lines->Add("<--------" + StrBuffer.SubString(5,StrBuffer.c_str()[2]));
  memset(Data,0,sizeof(Data));
  Util_Copy_Field(&StrBuffer.c_str()[4],0,'|',Data,255);

// SE RECIBE TERMINAL Y CLAVE

  switch(atoi(Data))
  {
   case 01 : /*Terminal */
   Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
   Memo2->Lines->Add("TERMINAL: " + AnsiString(Data));
   Terminal = Data;
```

```
    case 02 :
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo2->Lines->Add("Password: " + AnsiString(Data));
    Autenticacion->ParamByName("TERMINA")->AsString = Terminal;

    try
    {     Autenticacion->Active = false;
     Autenticacion->ParamByName("TERMINALID")->AsString = TerminalID;
     Autenticacion->ParamByName("PASSWD")->AsString = AnsiString(Data);
     Autenticacion->Active = true;
     if(Autenticacion->Active && !Autenticacion-
>FieldValues["TERMINALID"].IsNull()
      && !Autenticacion->FieldValues["TERMINALID"].IsEmpty())
     {
      Memo3->Lines->Add("Password VALIDO");
      Memo3->Lines->Add(AnsiString(Autenticacion-
>FieldValues["TERMINALID"]));
      Memo3->Lines->Add(AnsiString(Autenticacion->FieldValues["PASSWD"]));
      IDUsuario = Autenticacion->FieldValues["ID_USUARIO"];
      Memo3->Lines->Add("USUARIO");
      Memo3->Lines->Add(IDUsuario);
      IDLugar = Autenticacion->FieldValues["LUGAR"];
      Memo3->Lines->Add("LUGAR");
      Memo3->Lines->Add(IDLugar);
           strcpy(Data,"22|Password Valido|");
     }
     else
     {      Memo3->Lines->Add("Password INVALIDO");
      strcpy(Data,"23| |PASSWORD|INVALIDO| |");
     }    }
    catch(...)
    {
     memset(Data,0,sizeof(Data));
    }
    break;

   /** CONSULTA**/

    case 03:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Codigo Original: " + AnsiString(Data));
    try    {     Consulta->Active = false;
     Consulta->ParamByName("CODIGO")->AsString = AnsiString(Data);
     Consulta->Active = true;
```

```
      Memo3->Lines->Add(AnsiString(Consulta-
>FieldValues["CODIGO_ORIGINAL"]));
      Memo3->Lines->Add(AnsiString(Consulta->FieldValues["MONTO"]));
      sprintf(Data,"23|CODIGO:|%s|MONTO:|%s|",AnsiString(Consulta-
>FieldValues["CODIGO_ORIGINAL"]),AnsiString(Consulta-
>FieldValues["MONTO"]));
     }     else     {
       Memo3->Lines->Add("CODIGO INVALIDO");
       strcpy(Data,"23| |CODIGO|INVALIDO| |");
      }
     }
     catch(Exception &E)
    {
      Memo1->Lines->Add(E.Message);
      Memo1->Lines->Add(E.ClassName());
    }
    break;

    /** PAGO CHEQUE **/
    case 04:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Codigo Original: " + AnsiString(Data));
    Codigo = AnsiString(Data);
    try    {     Conslta->Active = false;
     Consulta->ParamByName("CODIGO")->AsString = AnsiString(Data);
     Consulta->Active = true;
     if(Consulta->Active && !Consulta-
>FieldValues["CODIGO_ORIGINAL"].IsNull()
       && !Consulta->FieldValues["CODIGO_ORIGINAL"].IsEmpty())
     {
      Memo3->Lines->Add(AnsiString(Consulta-
>FieldValues["CODIGO_ORIGINAL"]));
      Memo3->Lines->Add(AnsiString(Consulta->FieldValues["MONTO"]));
      strcpy(Data,"24|");
     }
     else
     {
       Memo3->Lines->Add("CODIGO INVALIDO");
       strcpy(Data,"23| |CODIGO|INVALIDO| |");
      }     }
     catch(Exception &E)
    {
      Memo1->Lines->Add(E.Message);
      Memo1->Lines->Add(E.ClassName());
```

```
    }
    break;

    case 05:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Cedula: " + AnsiString(Data));
    Cedula = AnsiString(Data);
    Util_Copy_Field(&StrBuffer.c_str()[4],2,'|',Data,255);
    Memo3->Lines->Add("Cheque: " + AnsiString(Data));
    Cheque = AnsiString(Data);

    try
    {
     Pago->ParamByName("NOCHEQUE")->AsString = Cheque;
     Pago->ParamByName("CEDULA")->AsString = Cedula;
     Pago->ParamByName("CODIGO")->AsString = Codigo;
     Pago->ParamByName("TERMINALID")->AsString = TerminalID;
     Pago->ParamByName("IDUSUARIO")->AsString = IDUsuario;
     Pago->ParamByName("LUGAR")->AsString = IDLugar;
     Pago->ExecSQL();
     Memo3->Lines->Add("PAGO INGRESADO");
     Consulta2->Active = false;
     Consulta2->ParamByName("CODIGO")->AsString = Codigo;
     Consulta2->Active = true;
      Memo3->Lines->Add(AnsiString(Consulta2-
>FieldValues["NUMERO_ORDEN"]));
      Memo3->Lines->Add(AnsiString(Consulta2-
>FieldValues["BENEFICIARIO"]));
      Memo3->Lines->Add(AnsiString(Consulta2->FieldValues["REMITENTE"]));
      Memo3->Lines->Add(AnsiString(Consulta2-
>FieldValues["MONTO_DOLARES"]));
      Memo3->Lines->Add(AnsiString(Consulta2-
>FieldValues["TIPO_CAMBIOP"]));
      Memo3->Lines->Add(AnsiString(Consulta2->FieldValues["MONTO_ML"]));
      DisplayStr = FormatDateTime("dd/mm/yyyy|",Date())
forrmatDateTime("hh:mm:ss|",Time());
      memset(Data,0,sizeof(Data));
      sprintf(Data,"25|ORDEN NUMERO:|%s|||%s|%s|%s|%s|%s|%s|%s|",
       AnsiString(Consulta2->FieldValues["TIPO_CAMBIOP"]),
      Codigo.c_str(),
      AnsiString(Consulta2->FieldValues["MONTO_ML"]),
      AnsiString(Consulta2->FieldValues["MONTO_DOLARES"]),
      DisplayStr.c_str());
```

```
     }
     else
     {
      Memo3->Lines->Add("CODIGO DE ORDEN NO ENCONTRADO");
     }
    }
    catch(Exception &E)
    {
     Memo3->Lines->Add("ERROR AL INGRESAR PAGO");
     Memo3->Lines->Add(E.Message);
     Memo3->Lines->Add(E.ClassName());
    }
    break;

  //DEPÓSITOS

  case 06:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Codigo Original: " + AnsiString(Data));
    Codigo = AnsiString(Data);
    try
    {     Consulta_Depositos->Active = false;
     Consulta_Depositos->ParamByName("CODIGO")->AsString =
AnsiString(Data);
     Consulta_Depositos->Active = true;
     if(Consulta_Depositos->Active && !Consulta_Depositos-
>FieldValues["CODIGO_ORIGINAL"].IsNull()
       && !Consulta_Depositos->FieldValues["CODIGO_ORIGINAL"].IsEmpty())
     {
      Memo3->Lines->Add(AnsiString(Consulta_Depositos-
>FieldValues["CODIGO_ORIGINAL"]));
      Memo3->Lines->Add(AnsiString(Consulta_Depositos-
>FieldValues["MONTO"]));
      strcpy(Data,"26|");
     }
        catch(Exception &E)
    {
      Memo1->Lines->Add(E.Message);
      Memo1->Lines->Add(E.ClassName());
    }
    break;

  case 07:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
```

```
    Memo3->Lines->Add("Boleta: " + AnsiString(Data));
    Boleta = AnsiString(Data);
    Util_Copy_Field(&StrBuffer.c_str()[4],2,'|',Data,255);
    Memo3->Lines->Add("Cheque: " + AnsiString(Data));
    Cheque = AnsiString(Data);
    Util_Copy_Field(&StrBuffer.c_str()[4],3,'|',Data,255);
    Memo3->Lines->Add("Cuenta Destino: " + AnsiString(Data));
    CuentaD = AnsiString(Data);
    Util_Copy_Field(&StrBuffer.c_str()[4],4,'|',Data,255);
    Memo3->Lines->Add("Banco Destino: " + AnsiString(Data));
    TipoCuenta = AnsiString(Data);
    Util_Copy_Field(&StrBuffer.c_str()[4],6,'|',Data,255);
    Memo3->Lines->Add("Certificacion Banco: " + AnsiString(Data));
    Certificacion = AnsiString(Data);
    Util_Copy_Field(&StrBuffer.c_str()[4],7,'|',Data,255);
    Memo3->Lines->Add("Fecha Certificacion: " + AnsiString(Data));
    FechaCert = AnsiString(Data);

    try
    {
     Pago_Depositos->ParamByName("NOCHEQUE")->AsString = Cheque;
     Pago_Depositos->ParamByName("BOLETA")->AsString = Boleta;
     Pago_Depositos->ParamByName("CODIGO")->AsString = Codigo;
     Pago_Depositos->ParamByName("TERMINALID")->AsString = TerminalID;
     Pago_Depositos->ParamByName("CUENTAD")->AsString = CuentaD;
     Pago_Depositos->ParamByName("IDUSUARIO")->AsString = IDUsuario;
      Pago_Depositos->ParamByName("FECHACERT")->AsString = FechaCert;
     Pago_Depositos->ExecSQL();
     Memo3->Lines->Add("DEPOSITO INGRESADO");
     Consulta_Depositos2->Active = false;
     Consulta_Depositos2->ParamByName("CODIGO")->AsString = Codigo;
     Consulta_Depositos2->Active = true;
     if(Consulta_Depositos2->Active && !Consulta_Depositos2-
>FieldValues["CODIGO_ORIGINAL"].IsNull()
        && !Consulta_Depositos2->FieldValues["CODIGO_ORIGINAL"].IsEmpty())
     {
           Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["BENEFICIARIO"]));
      Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["REMITENTE"]));
      Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["MONTO_ML"]));
      Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["TIPO_CAMBIOP"]));
```

```
      Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["MONTO_DOLARES"]));
      Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["NOMBRE_BANCO"]));
      Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["TIPOCUENTA"]));
      Memo3->Lines->Add(AnsiString(Consulta_Depositos2-
>FieldValues["CERTIFICACION_BANCO"]));
   FieldValues["FECHA_INGRESO"]));

      sprintf(Data,"27|ORDEN NUMERO:|%s|",
      AnsiString(Consulta_Depositos->FieldValues["NUMERO_ORDEN"]),
      AnsiString(Consulta_Depositos2->FieldValues["TIPO_CAMBIOP"]),
      Codigo.c_str(),
      AnsiString(Consulta_Depositos2->FieldValues["MONTO_ML"]),
      AnsiString(Consulta_Depositos2->FieldValues["MONTO_DOLARES"]),
      AnsiString(Consulta_Depositos2->FieldValues["NOMBRE_BANCO"]),
      AnsiString(Consulta_Depositos2->FieldValues["TIPOCUENTA"]),
      AnsiString(Consulta_Depositos2-
>FieldValues["CERTIFICACION_BANCO"]),
      AnsiString(Consulta_Depositos2-
>FieldValues["FECHA_CERTIFICACION"]),
      DisplayStr.c_str());

     Memo3->Lines->Add("ERROR AL INGRESAR DEPOSITO");
     Memo3->Lines->Add(E.Message);
     Memo3->Lines->Add(E.ClassName());
    }
    break;

    /***REVERSA DE PAGOS***/


    case 08:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Codigo Original: " + AnsiString(Data));
    Codigo = AnsiString(Data);
    try
    {
     Consulta_Reversa->Active = false;
     Consulta_Reversa->ParamByName("CODIGO")->AsString =
AnsiString(Data);
     Consulta_Reversa->Active = true;
```

```
     if(Consulta_Reversa->Active && !Consulta_Reversa-
>FieldValues["CODIGO_ORIGINAL"].IsNull()
       && !Consulta_Reversa->FieldValues["CODIGO_ORIGINAL"].IsEmpty())
     {
      Memo3->Lines->Add(AnsiString(Consulta_Reversa-
>FieldValues["CODIGO_ORIGINAL"]));
      Memo3->Lines->Add(AnsiString(Consulta_Reversa-
>FieldValues["MONTO"]));

    break;

    case 09:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Codigo: " + AnsiString(Data));
    Cedula = AnsiString(Data);

    try
    {
     Reversa->ParamByName("CODIGO")->AsString = Codigo;
     Reversa->ParamByName("IDUSUARIO")->AsString = IDUsuario;
AsString = Codigo;
     Consulta_Reversa->Active = true;
     if(Consulta_Reversa->Active && !Consulta_Reversa-
>FieldValues["CODIGO_ORIGINAL"].IsNull()
        && !Consulta_Reversa->FieldValues["CODIGO_ORIGINAL"].IsEmpty())
     {
      Memo3->Lines->Add(AnsiString(Consulta_Reversa-
>FieldValues["NUMERO_ORDEN"]));
      Memo3->Lines->Add(AnsiString(Consulta_Reversa-
>FieldValues["CODIGO_ORIGINAL"]));
      DisplayStr = FormatDateTime("dd/mm/yyyy|",Date()) +
FormatDateTime("hh:mm:ss|",Time());

      sprintf(Data,"29|ORDEN NUMERO:|%s|||%s|%s|%s|",
      AnsiString(Consulta_Reversa->FieldValues["NUMERO_ORDEN"]),
      AnsiString(Consulta_Reversa->FieldValues["MONTO"]),
      Codigo.c_str(),
      DisplayStr.c_str());

     }   else
     {
      Memo3->Lines->Add("CODIGO DE ORDEN NO ENCONTRADO");
     }   }
    catch(Exception &E)
```

```
    {
     Memo3->Lines->Add("ERROR AL INGRESAR REVERSA");
     Memo3->Lines->Add(E.Message);
     Memo3->Lines->Add(E.ClassName());
    }
    break;

    /** PAGO EN EFECTIVO **/
    case 10:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Codigo Original: " + AnsiString(Data));
    Codigo = AnsiString(Data);
    try
    {
     Consulta_Efectivo->Active = false;
     Consulta_Efectivo->ParamByName("CODIGO")->AsString =
AnsiString(Data);
     Consulta_Efectivo->Active = true;
     if(Consulta_Efectivo->Active && !Consulta_Efectivo-
>FieldValues["CODIGO_ORIGINAL"].IsNull()
       && !Consulta_Efectivo->FieldValues["CODIGO_ORIGINAL"].IsEmpty())
     {
      Memo3->Lines->Add(AnsiString(Consulta_Efectivo-
>FieldValues["CODIGO_ORIGINAL"]));
      Memo3->Lines->Add(AnsiString(Consulta_Efectivo-
>FieldValues["MONTO"]));
      strcpy(Data,"2030|");
     }
     else
     {
       Memo3->Lines->Add("CODIGO INVALIDO");
       strcpy(Data,"23| |CODIGO|INVALIDO| |");
      }
     }
     catch(Exception &E)
    {
      Memo1->Lines->Add(E.Message);
      Memo1->Lines->Add(E.ClassName());
    }
    break;

    case 11:
    Util_Copy_Field(&StrBuffer.c_str()[4],1,'|',Data,255);
    Memo3->Lines->Add("Cedula: " + AnsiString(Data));
```

```
    Cedula = AnsiString(Data);

    try
    {
     Pago_Efectivo->ParamByName("CEDULA")->AsString = Cedula;
     Pago_Efectivo->ParamByName("CODIGO")->AsString = Codigo;
     Pago_Efectivo->ParamByName("TERMINALID")->AsString = TerminalID;
     Pago_Efectivo->ExecSQL();

     {
      Memo3->Lines->Add(AnsiString(Consulta_Efectivo2-
>FieldValues["NUMERO_ORDEN"]));
      Memo3->Lines->Add(AnsiString(Consulta_Efectivo2-
>FieldValues["BENEFICIARIO"]));
      Memo3->Lines->Add(AnsiString(Consulta_Efectivo2-
>FieldValues["REMITENTE"]));
      Memo3->Lines->Add(AnsiString(Consulta_Efectivo2-
>FieldValues["MONTO_ML"]));
      Memo3->Lines->Add(AnsiString(Consulta_Efectivo2

      DisplayStr = FormatDateTime("dd/mm/yyyy|",Date()) +
FormatDateTime("hh:mm:ss|",Time());
      memset(Data,0,sizeof(Data));
      sprintf(Data,"31|ORDEN NUMERO|",
      AnsiString(Consulta_Efectivo->FieldValues["NUMERO_ORDEN"]),
      AnsiString(Consulta_Efectivo2->FieldValues["BENEFICIARIO"]),
      AnsiString(Consulta_Efectivo2->FieldValues["REMITENTE"]),
      AnsiString(Consulta_Efectivo2->FieldValues["TIPO_CAMBIOP"]),
      Codigo.c_str(),
      AnsiString(Consulta_Efectivo2->FieldValues["MONTO_ML"]),
      AnsiString(Consulta_Efectivo2->FieldValues["MONTO_DOLARES"]),
      DisplayStr.c_str());

     }
     else
     {
      Memo3->Lines->Add("CODIGO DE ORDEN NO ENCONTRADO");
     }
    }
    catch(Exception &E)
    break;
    default :
    memset(Data,0,sizeof(Data));
    break;
```

```
    }
    Memo2->Lines->Add(Data);
    memset(SendBuffer,0,sizeof(SendBuffer));
    SendBuffer[0] = 0x01;
    SendBuffer[1] = (0x40 | (an << 2)) | (sn << 3);
    SendBuffer[2] = strlen(Data);
    SendBuffer[3] = CRC8_calculate(&SendBuffer[1],2);

    if(SendBuffer[2] > 0)
    {
        }
    else
    {
     nrComm1->SendData(StrBuffer.c_str(),4);
     Memo1->Lines->Add("-------->ACK");
    }
    StrBuffer = "";
    State = 0;
   }
   default :
    StrBuffer = "";
    State = 0;
   break;
  } } }
catch(Exception &E)
{
 Application->ProcessMessages();
 Memo2->Lines->Add(AnsiString(E.ClassName())+ E.Message);
} }

//---------------------------------------------------------------------------

unsigned char CRC8_calculate(byte *data, int len )
{
 unsigned char CRC8_table[] = {
         0x00, 0x07, 0x0E, 0x09, 0x1C, 0x1B, 0x12, 0x15,
                  0x48, 0x4F, 0x46, 0x41, 0x54, 0x53, 0x5A, 0x5D,
         0xE0, 0xE7,  0xE6, 0xE1, 0xE8, 0xEF, 0xFA, 0xFD,
 };

 unsigned char crc = 0;

 while(len--)
  crc = CRC8_table[crc ^ *data++];
```

```
 return crc;
}

static unsigned short CRC16_calculate(byte *buf, int buf_len )
{
static const unsigned short crc16_table[] =
{
   0x0000 , ,0x4c80 ,0x8c41 ,0x4400 ,0x84c1 ,0x8581
 ,0x4540 ,0x8701 ,0x47c0 ,0x4680 ,0x8641  ,0x8201
,0x42c0 ,0x4380 ,0x8341 ,0x4100 ,0x81c1 ,0x8081 ,0x4040
} ;

 unsigned short          crc;
 int            i;

   while ((*Message != 0) && (*Message != ETX) && Field_Num != 0)
   {
      if ((*Message == Field_Sep) && (++Field_Cnt == Field_Num))
               break;
      Message++;
   }
   if ((*Message == Field_Sep) || (Field_Num == 0))
   {
      if (*Message == Field_Sep)
         Message++;
      while ( (Message[Cnt] != 0) && (Message[Cnt] != ETX)
             &&(Message[Cnt] != Field_Sep) && (Cnt < Buffer_Length))
      {
           Buffer[Cnt] = Message[Cnt];
           Cnt++;
      }
      Buffer[Cnt] = 0;
   }
   return (Cnt);
}
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 Memo1->Clear();
 Memo2->Clear();
 Memo3->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Salir1Click(TObject *Sender)
```

```
{
 Close();
```