ANALISIS, DISEÑO E IMPLEMENTACION DE UN SISTEMA AUTOMATICO DE SERVICIO DE PREPAGO PARA COLECTIVOS (FASE 1)

MONOGRAFIA DE TRABAJO DE GRADO

FELIX PARDO MOTTA

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Control
Popayán, marzo de 2005

TABLA DE CONTENIDO

INTRODUCCION 6

1 INTRODUCCION A LOS SISTEMAS DE TIEMPO REAL 8

1.1 CLASIFICACION DE LOS SISTEMAS DE TIEMPO REAL 10
1.2 UML PARA TIEMPO REAL 11

1.2.1 UML EN EL DESARROLLO DE SISTEMA EMBEBIDOS 12

1.3 MICROCONTROLADORES (PIC'S) 14
1.4 DESPLIEGUES LCD'S 18

2 EL PROBLEMA 19

2.1 DESCRIPCION DEL SISTEMA 19
2.2 ANALISIS DE REQUERIMIENTOS 23
2.2.1 EVENTOS EXTERNOS 24
2.2.2 VISTA PRINCIPAL DEL SISTEMA 26
2.2.3 EVENTOS EXTERNOS DEL SISTEMA 27
2.2.4 DIAGRAMA DE CASOS DE USO 29
2.2.4.1 DEFINICION DE ACTORES 31
2.2.4.2 CASOS DE USO DEL SISTEMA 32
2.2.5 DIAGRAMAS DE SECUENCIA 40

3 ANALISIS: DEFINIR LA ESTRUCTURA DE LOS OBJETOS 42

3.1 IDENTIFICACION DE OBJETOS 42
3.1.1 OBJETOS ACTIVOS 42
3.1.2 OBJETOS DE INFORMACION PERSISTENTE 43
3.1.3 OBJETOS DE CONTROL 43
3.1.4 ASOCIACIONES ENTRE OBJETOS 44
3.2 DIAGRAMA DE CLASES 44
3.2.1 DIAGRAMA DE CLASES CAPA SISTEMA PREPAGOP_C 44

4 ANALISIS: DEFINIR EL COMPORTAMIENTO DE LOS OBJETOS 47

4.1 DIAGRAMAS DE ESTADO 47
4.1.1 CAPA SISTEMA PREPAGOP_C 47
4.1.1.1 TARJETA 47
4.1.1.2 UNIDADCONTROL 48
4.1.1.3 DESPLIEGUE 50
4.1.2 CAPA SISTEMA PREPAGO_PC 51
4.1.2.1 UNIDADCONTROL 51
4.1.2.2 BASEDATOS 52
4.1.2.3 DESPLIEGUE 54

5 DISEÑO 56

5.1 DISEÑO ARQUITECTONICO 56
5.2 GESTIONTARJETA 57
5.2.1 VISTA DE COMPONENTES 60

5.3 TRANSFERIRINFORMACION 60
5.3.1 VISTA DE COMPONENTES 61

6 IMPLEMENTACION 63

6.1 HARDWARE 63
6.1.1 MICROCONTROLADOR PIC16F877 63
6.1.2 MEMORIA EEPROM 24LC08B 64
6.1.3 DESPLIEGUE LCD Y TECLADO 65

6.2 SOFTWARE 67

7 PRUEBAS 68

7.1 PRUEBAS DE CAJA NEGRA 68
7.2 PRUEBAS DE CAJA BLANCA 69
7.3 PRUEBAS DE INTEGRACION 70

8 CONCLUSIONES 72

9 GLOSARIO DE TERMINOS 74

10 BIBLIOGRAFIA 77

11 ANEXO 1: GENERALIDADES DEL BUS I2C 1

11.1 ESPECIFICACION DEL BUS I2C 3
11.2 TRANSFERENCIA DE DATOS 5
11.2.1 RECONOCIMIENTO 6
11.3 SINCRONISMO Y ARBITRACION 8
11.3.1 SINCRONISMO 8
11.3.2 ARBITRACION 10
11.4 FORMATOS CON 7 BITS DE DIRECCIONES 11

12 ANEXO 2: MEMORIAS FLASH 19

12.1 TECNOLOGIAS 19
12.2 ARQUITECTURA DE MEMORIA 20
12.3 MODO DE LECTURA OPERACIONAL 21
12.4 PROGRAMACION Y BORRADO 24

13 ANEXO 3: CREACION DEL COMPONENTE SPC 25

14 ANEXO 4: COMPILACION COMPONENTE SPC 31

15 ANEXO 5: CODIGO CAPA SISTEMA PREPAGOP_C 35

16 ANEXO 6: MANUAL DE USUARIO 43

LISTADO DE FIGURAS

Figura 1 Partes internas del Microcontrolador 15
Figura 2 Microcontrolador PIC16F877 16
Figura 3 Diagrama En Bloques De Un Sistema De Prepago Para Colectivos 20
Figura 4 Modelo Contextual Onion Skin 24
Figura 5 Diagrama De Contexto Para El Sistema SPC 25
Figura 6 Diagrama De Contexto SPC Con Paquetes De Sistema 26
Figura 7 Vista Principal del Sistema 27
Figura 8 Diagrama De Capas Para El Sistema Prepago Para Colectivos 29
Figura 9 Diagrama De Casos De Uso Capa Sistema PrepagoP_C 30
Figura 10 Diagrama De Casos De Uso Capa Sistema Prepago_ PC 31
Figura 11 Diagrama De Secuencia Para La Capa Sistema PrepagoP_C 41
Figura 12 Diagrama De Secuencia Para La Capa Sistema Prepago_PC 41
Figura 13 Diagrama De Clases Para La Capa Sistema PrepagoP_C 45
Figura 14 Diagrama De Clases Para La Capa Sistema Prepago_PC 46
Figura 15 Diagrama De Estados Para La Capsula Tarjeta 48
Figura 16 Diagrama De Estados Para La Capsula UnidadControl 49
Figura 17 Diagrama De Estados Para La Capsula Despliegue 51
Figura 18 Diagrama De Estados Para La Capsula UnidadControl 52
Figura 19 Diagrama De Estados Para La Capsula BaseDatos 53
Figura 20 Diagrama De Estados Para La Capsula Despliegue 54
Figura 21 Diagrama De Paquetes Para El Sistema Prepago Para Colectivos 56
Figura 22 Diagrama De Comunicación Entre Las Capsulas UnidadControl Tarjeta y Despliegue 57
Figura 23 Diagrama De Puertos Para La Capsula UnidadControl 58
Figura 24 Diagrama De Puertos Para La Capsula Tarjeta 59
Figura 25 Diagrama De Puertos Para La Capsula Despliegue 59
Figura 26 Diagrama De Comunicación Entre Las Capsulas UnidadControl Basedatos y Despliegue 61
Figura 27 Diagrama De Puertos Para La Capsula Tarjeta 61
Figura 28 Diagrama Circuital Para El Sistema SPC 66
Figura 29 ComponenteSPCInstance 70

LISTADO DE TABLAS

Tabla 1 Distribución De Pines Para Un Despliegue LCD 18
Tabla 2: Eventos Externos Del Sistema 28
Tabla 3: Definición De Actores Del Sistema 31
Tabla 4: Caso De Uso Gestionar_Tarjeta 33
Tabla 5: Caso De Uso Detectar_Tarjeta 33
Tabla 6: Caso De Uso Verificar_Tarjeta 34
Tabla 7: Caso De Uso Debitar_Tarjeta 35
Tabla 8: Caso De Uso Acumular_Pasajes 36
Tabla 9: Caso De Uso Visualizar_Transacción 37
Tabla 10: Caso De Uso Sincronizar_PC 38
Tabla 11: Caso De Uso Transferir_Información 39
Tabla 12: Caso De Uso Resetear 39
Tabla 13: Caso De Uso Autoconfigurar 40
Tabla 14: Objetos Activos 43
Tabla 15: Asociaciones Entre Objetos 44
Tabla 16 Descripción de Pines Para La Memoria EEPROM 24LC08B 64

INTRODUCCION

A mediados del siglo pasado con la aparición de los transistores se generó la revolución de la electrónica y con ello los más grandes avances conocidos hasta ahora, pero el afán del hombre por desarrollar y mejorar tecnologías no tiene límites. Es así como se observa un crecimiento en todas las áreas de la industria que hasta hace pocos años parecían imposibles de alcanzar, buscando con ello ofrecer arquitecturas abiertas y de dominio público a bajos costos y de muy alto rendimiento.

Una de las áreas que más ha evolucionado y que actualmente representa una solución real para sistemas que involucran *Hardware* y *Software –FIRMWARE–* son los sistemas empotrados, debido a que estos traen involucrado dentro del mismo dispositivo la parte inteligente. Dicha evolución ha sido posible gracias a que actualmente se cuenta con metodologías para el desarrollo de software de calidad y herramientas de simulación que permiten probar un sistema en desarrollo, bajo ciertas condiciones reales de funcionamiento.

El proyecto *"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA AUTOMÁTICO DE SERVICIO DE PREPAGO PARA COLECTIVOS(FASE 1)"* considerado como un sistema empotrado suple deficiencias de servicio a los usuarios en el momento de abordar un colectivo y proceder a pagar su respectivo pasaje, así como la dificultad que tiene la empresa para controlar el manejo de dineros por parte de sus conductores.

En este proyecto se ha utilizado en la parte de captura de requerimientos, análisis y diseño la suite de herramientas *CASE* de la *RATIONAL ROSE*, específicamente *Rational Rose RealTime* y la metodología *RUP* con notación *UML* para tiempo real.

En la parte de implementación de software se utilizaron las aplicaciones *MPLAB* de la *MICROCHIP TECHNOLOGY* y *PICBASIC*, herramientas que permitieron programar el microcontrolador *PIC16F877A*. Este microcontrolador presenta un alto rendimiento y bajos costos, lo cual permitió desarrollar el módulo *MVAT*[1], encargado de realizar todas las funciones inteligentes del sistema tales como la detección, verificación y debitación de la Tarjeta Prepago, así como el despliegue en pantalla de mensajes alusivos a cada transacción. Como herramienta de laboratorio se utilizó *ISIS PROFESSIONAL (PROTEUS VSM)*, la cual permitió realizar el diagrama de pines del sistema y la simulación del mismo antes de ser implantado en la protoboard y la tarjeta impresa.

Se considera de gran importancia revisar los Anexos. En ellos se suministra información y conceptos del bus I^2C (utilizado en la comunicación del microcontrolador PIC16F877A con las memorias EEPROM y despliegue LCD) , Memorias de tipo FLASH (parte integral del microcontrolador PIC16F877A) y la creación del C*omponenteSPC* para la capa *Sistema PrepagoP_C*, utilizando la herramienta *Rational Rose RealTime*.

[1] Módulo de Verificación Almacenamiento y Transmisión de la información

1 INTRODUCCION A LOS SISTEMAS DE TIEMPO REAL

A medida que la tecnología ha superado límites de funcionamiento, velocidad, almacenamiento, etc., las aplicaciones para sistemas de tiempo real están presentes en todos los sectores – Industria, Computadores, Comunicaciones, Consumo, Automoción – los cuales varían en tamaño y alcance. Es así como se tienen máquinas lavadoras, hornos microondas, teléfonos, televisión, automóviles, trenes, sistemas de conmutación, aviones, control de procesos químicos y plantas de control nuclear.

El software que gobierna estos sistemas embebidos es más difícil de construir que el software que normalmente se utiliza para sistemas de escritorio, debido a que los sistemas de tiempo real tienen todos los problemas que se pueden encontrar en las aplicaciones de escritorio y además deben involucrar características críticas de respuesta en tiempo, robustez, seguridad, etc. Muchos sistemas de tiempo real a diferencia de los sistemas de computo convencional no presentan despliegues de información o teclados, pero son el corazón de muchos dispositivos aparentemente no computarizados. El usuario de estos dispositivos nunca estará conciente de la CPU empotrada la cual tomará decisiones acerca de cuando y como el sistema podría actuar. Tales sistemas a menudo deben operar por días o aún años sin parar, en la mayoría de ambientes hostiles. Los servicios y controles suministrados deben ser autónomos y temporizados, estos dispositivos frecuentemente tienen el potencial para hacer gran daño si ellos fallan.

La mayoría de sistemas de tiempo real interactúan directamente con dispositivos eléctricos e indirectamente con otros mecánicos, mediante software personalizado escrito específicamente para la aplicación el cual debe controlar dispositivos, involucrando un profundo conocimiento de las propiedades eléctricas y características de tiempo de los dispositivos involucrados. De esta manera podemos encontrar aplicaciones muy típicas que hacen uso de sensores los cuales proveen información hacia el sistema acerca del estado de su ambiente externo, tales como los dispositivos de monitoreo médico – máquinas electrocardiogramas – que permiten monitorear el estado de un paciente, sensores que permiten medir la velocidad del aire, attitude y altitud de un avión para la planeación de control de vuelo adecuado, sensores de posición angular y lineal censan la posición de los brazos de un robot y ajustan estos vía motores de paso.

Muchos sistemas de tiempo real a parte de los sensores utilizan como complemento los actuadores, los cuales están constituidos básicamente por una parte mecánica que permiten controlar o guiar procesos externos. Las computadoras de mando de vuelo comandan el motor y la orientación de las alas y cola para encontrar los parámetros de vuelo. Sistemas de control de procesos químicos controlan cuando, que clase y la cantidad de reactivos adicionados para mezclar en las tinas. Los marcapasos hacen los latidos del corazón a intervalos apropiados con cargas eléctricas conectadas dentro de las paredes del corazón.

1.1 CLASIFICACION DE LOS SISTEMAS DE TIEMPO REAL

Uno de los aspectos críticos en los sistemas de tiempo real es su velocidad de respuesta en el tiempo, pues estos deben asegurar un desempeño correcto en cualquier instante. La clasificación de los sistemas de tiempo real está dada por las restricciones de tiempo, los cuales pueden ser *hard*, *soft* y *firm*.

Los sistemas *hard* Se caracteriza por que el desempeño del sistema está gobernado por la exactitud de su respuesta en un intervalo de tiempo, cualquier acción tardía se constituye en una falla del sistema.

En un sistema soft el desempeño del sistema está restringido solamente por tiempos de ejecución promedio, es decir que la llegada tarde de datos usualmente no es significante, aunque cómputos tardíos consistentemente pueden traducirse en fallas del sistema.

Algunos sistemas tienen restricciones de desempeño que combinan requerimientos tanto de tipo *hard* como *soft.* Este tipo de sistemas pueden ser mezclados en la medida que se presenten las siguientes condiciones: Inicialmente el sistema debe mantener un suficiente desempeño promedio todo el tiempo y por último las tareas deben ser finalizadas en un instante de tiempo específico.

La gran mayoría de los sistemas de tiempo real están condicionados por requerimientos de desempeño en el tiempo. En el caso de los sistemas reactivos deben actuar en forma

oportuna a eventos externos donde la reacción puede ser una simple actuación digital, tal como encender una luz, o un lazo de control complicado controlando docenas de actuadores simultáneamente. Típicamente, muchas subrutinas o tareas deben actuar entre el evento causante y la acción resultante del sistema, siendo los eventos externos una limitante en el desempeño de un sistema. En dichos sistemas cada una de las actividades en procesamiento tienen asignadas una porción de tiempo y la suma de todo el conjunto de presupuestos de tiempo debe ser menor o igual al conjunto de restricciones del sistema.

1.2 UML PARA TIEMPO REAL

UML es un lenguaje para la especificación de sistemas complejos, desarrollado originalmente en respuesta al llamado de la OMG para un propósito de estandarizar el lenguaje de modelamiento de objetos. Las metas de UML son:

- Proveer un lenguaje de modelamiento visual expresivo y común para capturar la semántica de modelos de objetos.
- Proveer un modelo con una base semántica riguroza y formal que no requiera mucho formalismo por el usuario.
- Proveer un metamodelo el cual pueda ser finalizado y extendido por el usuario.
- Ser independiente del lenguaje de programación y de los procesos de desarrollo.
- Asegurar el crecimiento y madurez de las herramientas del Mercado orientadas a objetos.

- Soportar lo ultimo en metodologías de desarrollo, incluyendo colaboraciones, patrones de diseño, desarrollo de sistemas de gran escala, arquitecturas y componentes.
- Integrar las mejores características de desarrollo activo de OO e investigación dadas en procesos anteriores.

1.3 UML EN EL DESARROLLO DE SISTEMA EMBEBIDOS

En un proceso de desarrollo modelar es importante, puesto que esto ayuda a visualizar, especificar, construir y documentar la estructura y el comportamiento de la arquitectura de un sistema.

Inicialmente se parte del análisis que busca identificar claramente que es el sistema, cuales son sus funciones y sus principales características. Esta fase a su vez se divide en el *análisis de requerimientos, análisis y definición del comportamiento de los objetos*:

- El *análisis de requerimientos*[2] que comprende la definición del problema, el conjunto de objetos externos y su interacción con el sistema, para ello se apoya en conceptos tales como actores, eventos, escenarios, Diagramas de contexto, Diagramas de casos de uso, así como la descripción de cada uno de ellos.
- El *análisis de objetos*[3], etapa que se inicia una vez que los requerimientos han sido definidos, en el cual se identifican los objetos (Activos, Persistentes, Control), clases, cápsulas y protocolos capaces de ejecutar el comportamiento descrito en los casos de uso, así como su interrelación dentro del sistema, para esta parte se asocian

[2] Remitirse al capítulo 2: Análisis de Requerimientos
[3] Remitirse al capítulo 3: Análisis de Objetos

conceptos tales como las estrategias para la identificación de objetos, asociaciones entre objetos, diagramas de clases.

- La fase de Análisis culmina con la *Definición del Comportamiento de los Objetos*[4], siendo ésta un complemento a la fase anterior en la cual se describen los diagramas de estado para cada uno de los objetos identificados y la asignación de responsabilidades.

La fase del *diseño*[5], que busca una solución particular basándose en el modelo de análisis define las decisiones estratégicas de gran escala acerca de la implementación del sistema, involucrando colaboraciones de paquetes, tareas, creación de puertos y vista de componentes del modelo.

Finalmente la metodología involucra la parte de *implementación* y *pruebas*. No obstante es importante aclarar que la metodología es iterativa e incremental lo cual indica que no existe una división completamente entre cada una de las fases, sinó que por el contrario siempre se estará refinando el modelo hasta poner a punto el sistema.

NOTA: UML para tiempo real, permite modelar completamente el sistema a nivel lógico, pero en el evento que se tiene hardware involucrado este debe ser modelado suministrando las rutinas de bajo nivel que permitan simular el comportamiento del Hardware, aunque el resultado de dicho modelamiento no puede ser llevado

[4] Remitirse al capítulo 4: Definir el Comportamiento de los Objetos

directamente al Hardware como tal debido a que este no soporte los lenguajes orientados a objetos, razón que obliga a utilizar las herramientas típicas para la programación de microcontroladores.

1.4 MICROCONTROLADORES (PIC'S)

Circuito integrado programable que contiene todos los elementos necesarios para controlar un sistema. El fabricante los define como una familia de microcontroladores de bajo costo, bajo consumo de potencia y alta velocidad de operación. Los PIC16F87X pueden operar hasta una velocidad de 20MHz y el consumo de potencia es de 50 microvatios a una velocidad de 32Khz y de 6 microvatios en modo de reposo disminuyendo el consumo de potencia. Son circuitos integrados programables que contienen todos los componentes de un PC. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que le confiere la denominación de "Controlador incrustado".

El microcontrolador es un computador dedicado debido a que en su memoria sólo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada y salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar y

[5] Remitirse al Capítulo 5: Diseño

todos los recursos complementarios disponibles, tienen como única finalidad atender sus requerimientos.

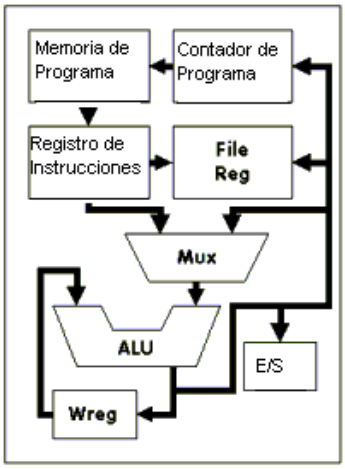

Figura 1 Partes internas del Microcontrolador

De la gráfica anterior podemos observar que un microcontrolador contiene prácticamente todos los componentes de un microprocesador así:

1. Una CPU (**C**entral **P**rocessor **U**nit o Unidad de Procesamiento Central) quien interpreta las instrucciones de programa.
2. Una memoria PROM (**P**rogrammable **R**ead **O**nly **M**emory o Memoria Programable Solamente para Lectura) la cual memoriza permanentemente las instrucciones de programa. Otros modelos de microcontroladores tienen memoria de programa de tipo EEPROM y otros de tipo FLASH, siendo esta última la más utilizada hoy en día por sus prestaciones.
3. Una memoria RAM (**R**andom **A**ccess **M**emory o Memoria de Acceso Aleatorio) utilizada para memorizar las variables utilizadas para el programa.
4. Una serie de LÍNEAS de E/S para controlar dispositivos externos o recibir pulsos de sensores, switches, etc.
5. Una serie de dispositivos auxiliares para su funcionamiento, como puede ser generador de reloj, bus, contador, etc.

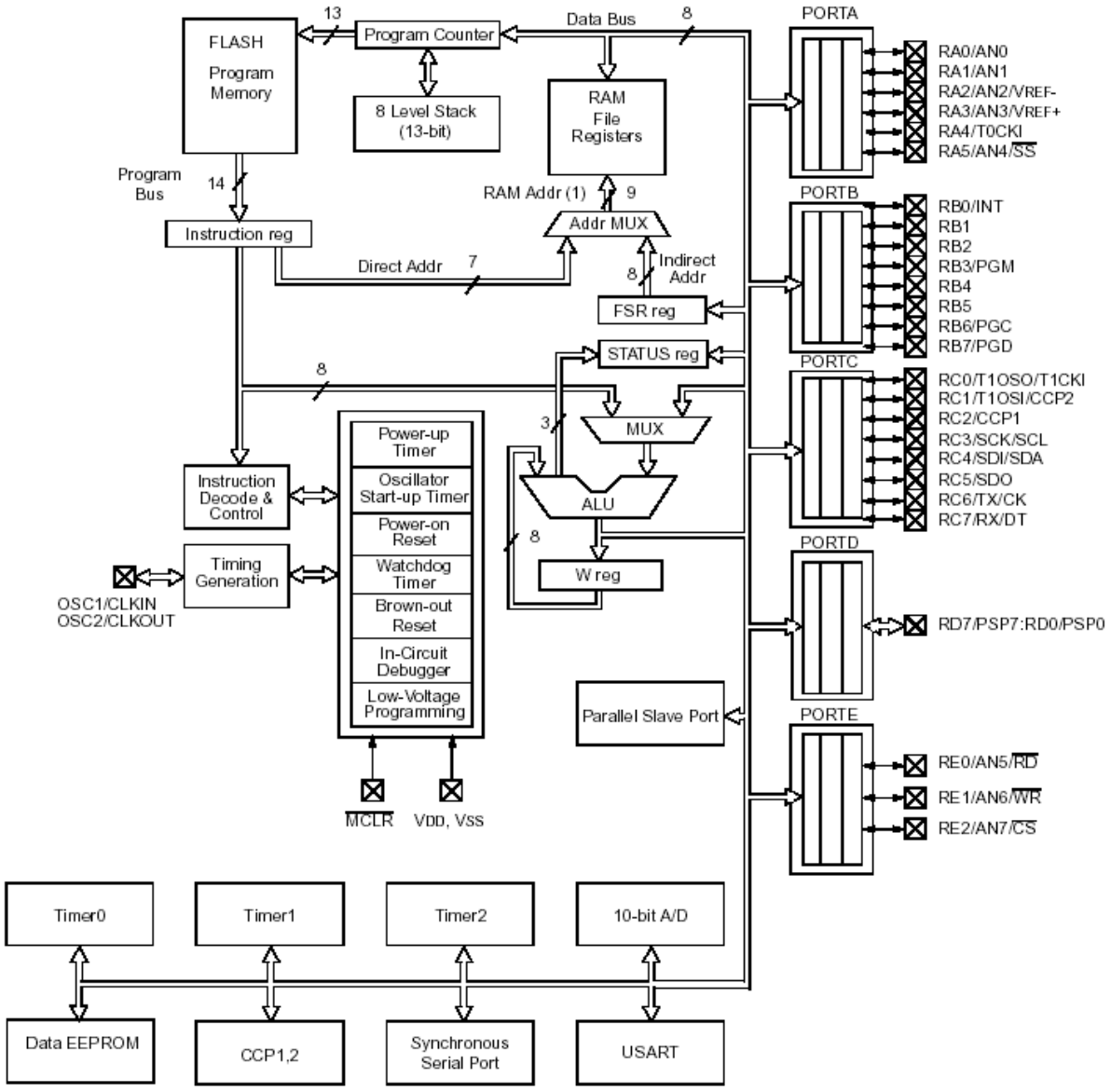

Figura 2 Microcontrolador PIC16F877

En la gráfica anterior se puede observar el diagrama en bloques del microcontrolador PIC16F877 tomado directamente de su Data Sheet. Las principales características de este microcontrolador son:

- Procesador de arquitectura RISC avanzada
- Juego de 35 instrucciones con 14 bits de longitud

- Velocidad de operación: 20 MHz de entrada de reloj – 200 ns por ciclo de instrucción.
- 8K x 14 palabras de memoria de programa FLASH.
- 368 x 8 bytes de Memoria de datos (RAM)
- 256 x 8 bytes de Memoria de datos (EEPROM)
- Pin de salida compatible a los microcontroladores PIC16C73B/74B/76/77
- Pila con 8 niveles
- Catorce puertos para interrupciones
- Modos de direccionamiento directo, indirecto y relativo
- Power-on Reset(POR)
- Power-up Timer(PWRT) y Oscilador Star-up Timer(OST)
- Watchdog Timer (WDT) con su propio on-chip RC
- Oscilador para operación fiable.
- Código de protección programable.
- Opciones de oscilador seleccionable
- Baja potencia y alta velocidad para tecnologías CMOS FLASH/EEPROM
- Amplio rango de voltajes de operación: 2.0 V hasta 5.5V
- Alto manejo de corriente 25 mA
- Rangos de temperatura industrial y comercial.
- Bajo consumo de potencia :
- 2 mA a 5V, 4MHz
- 20μA a 3V, 32 KHz
- 1μA en estado de Standbye

Características de periféricos

Timer0: Timer de 8-bits
Timer1: Timer de 16-bits
Timer2: Timer de 8-bits
Dos módulos de Captura-Comparación-PWM
Conversor Análogo-Digital de 10 bits
Puerto serial Sincrónico (SSP) con SPI (Modo maestro) e I^2C (Maestro/Esclavo)
Receptor Universal Sincrónico-Asincrónico
Puerto esclavo paralelo (PSP) de 8-bits con controles externos RD, WR y CS

1.5 DESPLIEGUES LCD'S

Los módulos de Cristal Líquido o LCD, solucionan inconvenientes y presentan algunas ventajas, como un menor consumo de corriente; no hay que preocuparse por hacer multiplexaje, no hay que hacer tablas especiales con los caracteres que se desea mostrar, se pueden conectar fácilmente con microcontroladores.

Existe en el mercado diferentes tipos de LCD's, aunque funcionalmente todos operan de la misma forma, sus diferencias básicas están en el número de líneas, caracteres y tamaño de caracteres que estos son capaces de desplegar. Así encontramos que se encuentran LCD's de 2x16, 2x20, 4x20, 4x40, etc. líneas por caracteres respectivamente. En la tabla 1 se presentan el diagrama de pines de un LCD de 2x16

TERMINAL	SÍMBOLO	NOMBRE Y FUNCIÓN
1	Vss	Tierra, 0V
2	Vdd	Alimentación +5V
3	Vo	Ajuste de voltaje de contraste
4	RS	Selección Dato / Control
5	R/W	Lectura / escritura en LCD
6	E	Habilitación
7	D0	D0 Bit menos significativo
8	D1	D1
9	D2	D2
10	D3	D3
11	D4	D4
12	D5	D5
13	D6	D6
14	D7	D7 Bit más significativo

Tabla 1 Distribución De Pines Para Un Despliegue LCD

2 EL PROBLEMA

Desplazarse de un sitio a otro siempre será importante para el hombre debido a que de esta forma desarrolla innumerables actividades tales como el estudio, el trabajo, la recreación, el deporte, etc. Debido a esto actualmente existen gran cantidad de empresas prestadoras de servicios públicos de transporte y la tendencia del mercado es mejorar su atención al cliente (pasajero), lo cual representa un reto para dichas empresas traducidas finalmente en necesidades.

Una de las actividades más importantes de suplir por parte de las empresas prestadoras de servicio público de transporte urbano en colectivos son el servicio al cliente y las dificultades presentadas para el mismo en el momento de abordar un colectivo y proceder a pagar su respectivo pasaje, así como la dificultad que tiene la empresa para controlar el manejo de dineros por parte de sus conductores.

2.1 DESCRIPCION DEL SISTEMA

Este sistema inicia cuando un usuario (pasajero) aborda un colectivo y ha adquirido previamente una tarjeta prepago para colectivos, el usuario introduce la tarjeta en el *Módulo Lector de Tarjetas,* el cual realiza la lectura que es enviada al *MVAT* para su respectivo análisis. Dependiendo del análisis el sistema procederá a debitar automáticamente de la tarjeta el valor correspondiente a un pasaje y almacenarlo en memoria o por el contrario desplegará en una pantalla un mensaje de invalidez de la

tarjeta. El sistema repetirá el mismo procedimiento cada vez que un pasajero aborde un colectivo e irá incrementando la memoria de almacenamiento del sistema con cada pasaje debitado. Al final de la jornada, el sistema permitirá bajar la información almacenada en el *MVAT* a través de la interfase RS232C a la base de datos contenida en un PC, de la empresa a la cual pertenece el automotor. Una vez el sistema ha permitido efectuar la descarga de la información, se encontrará listo para iniciar el ciclo en un nuevo día. La Figura 3 muestra la descripción general del sistema con los siguientes bloques:

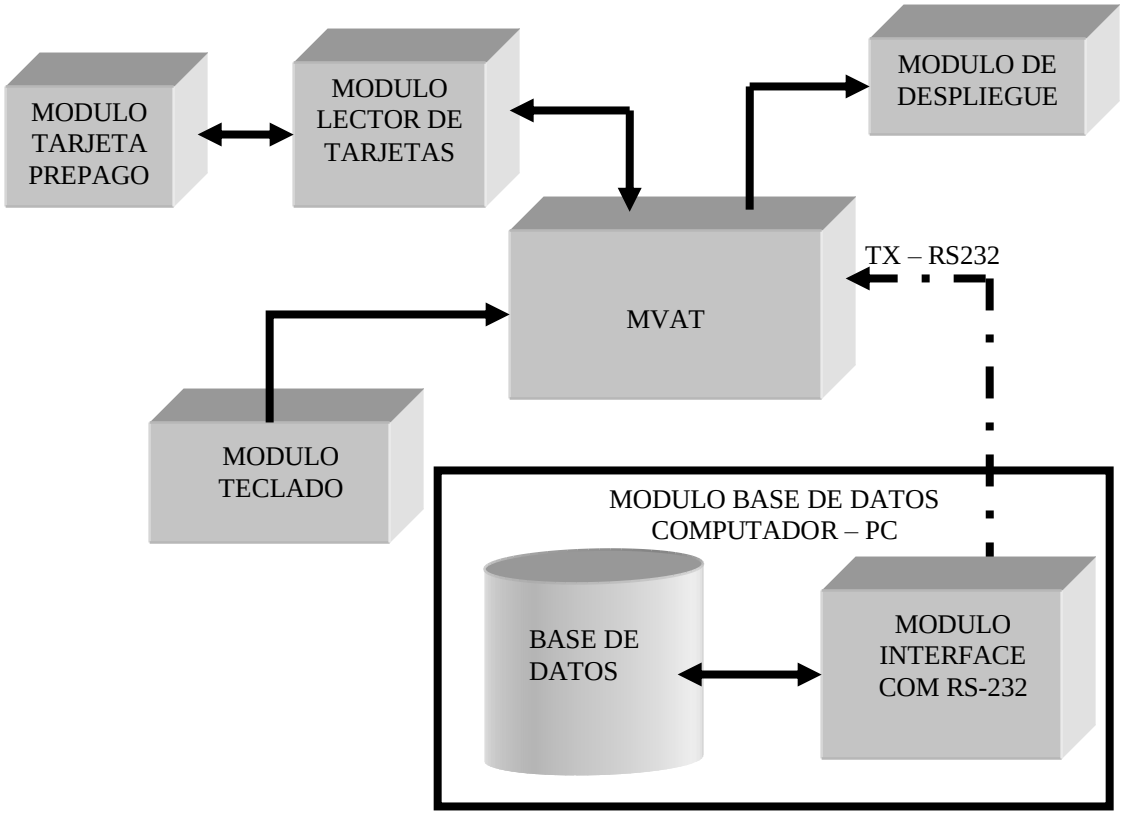

Figura 3 Diagrama En Bloques De Un Sistema De Prepago Para Colectivos

MODULO TARJETA PREPAGO: Representa la tarjeta prepago que es previamente adquirida por los usuarios. Como características más relevantes de este módulo se enuncian:

- Poseerá una memoria EEPROM, que contendrá el valor almacenado en proporción directa al número de pasajes y el código de identificación único de la tarjeta.
- La tarjeta estará en capacidad de ser programada para un número ilimitado de pasajes según las necesidades del cliente (Por ejemplo, para el caso de costo por transporte mensual se proyecta un promedio de 4 pasajes diarios en un lapso de 30 días para un total de 120 pasajes).
- Una tarjeta nueva podrá ser adquirida en supermercados y tiendas.
- La pérdida ocasional por parte del dueño de una tarjeta, será asumida directamente por este.

MODULO LECTOR DE TARJETAS: Encargado básicamente de generar la orden de interrupción al microcontrolador para indicarle a este que una tarjeta ha sido introducida, activar el mecanismo de polarización, lectura y envío de información para su respectivo análisis por parte del microcontrolador. Si el análisis es positivo activará el mecanismo de escritura y debitación del pasaje correspondiente. Es importante anotar que la información procesada a este nivel es básicamente a nivel de bits (0 y 1).

MVAT: Compuesto básicamente por el microcontrolador PIC16F877 y una memoria EEPROM, las principales funciones de este módulo serán:

- Verificar la información enviada desde el módulo lector de tarjetas, bien sea para proceder a autorizar la debitación de un pasaje de la tarjeta prepago o para desplegar en pantalla (módulo de despliegue) un mensaje de tarjeta dañada o tarjeta sin pasajes.
- Almacenar e incrementar la memoria por cada pasaje que haya sido debitado correctamente de la tarjeta prepago de un usuario.
- Contener en la base de datos del microcontrolador PIC16F887 la información más relevante para la empresa en cuanto al movimiento relacionado con un automotor como: Día, hora del servicio, ruta, número del automotor, valor del pasaje, pasajes debitados y el valor total de pasajes debitados.
- Permitir la transmisión de la información contenida en la base de datos del microcontrolador a través de la interfase RS232C al PC (contiene la base de datos de la empresa para cada automotor en cuestión) al final de cada jornada.

MODULO BASE DE DATOS: Módulo que le permitirá a la empresa gestionar y controlar a diario el movimiento de todos los automotores afiliados a una empresa. Este recibe a través de una interfase RS232C la información de cada automotor que se encuentra almacenada en el *MVAT*. Este contendrá toda la información que la empresa necesite tener sobre cada uno de los automotores afiliados tales como: Número interno del vehículo, placas, propietario(s), conductor, número del motor, modelo del carro, Marca de vehículo, rutas asignadas, capacidad, movimiento diario, Infracciones, Administración(papeles, seguro), Mantenimiento(cambio de aceite, llantas, estado

vehículo,etc.), generar informes y estadísticas sobre el movimiento diario de un vehículo en el momento que la empresa lo requiera o el dueño(s) lo deseen.

MODULO TECLADO: Será la interfase que permitirá interactuar con el sistema para la debitación de pasajes, la transferencia de la información almacenada en el *MVAT* al módulo *Base de Datos Computador-PC*.

MODULO DE DESPLIEGUE: Este módulo tiene como funciones desplegar información visual al usuario, conductor, así como al administrador sobre la operación realizada.

2.2 ANALISIS DE REQUERIMIENTOS

El éxito en el desarrollo de cualquier sistema es asegurar la comprensión de los requerimientos solicitados, tales como la funcionalidad básica del sistema. Para ello se hizo uso de los conceptos de contexto de eventos externos, en el cual el sistema se muestra como una caja negra compuesta de objetos enviando y recibiendo mensajes hacia objetos actores externos. Adicionalmente las herramientas básicas del análisis de requerimientos son los casos de uso y escenarios, los cuales permiten descomponer la funcionalidad primaria del sistema y establecer protocolos necesarios para encontrar los requerimientos funcionales.

2.2.1 EVENTOS EXTERNOS

El contexto del sistema es un mapa del mundo de interés para el sistema. El diagrama de contexto muestra el sistema como una entidad única rodeada por otros objetos del mundo real. En la figura 4 se muestra el modelo ONION de Ellis, el cual está compuesto de capas sucesivas con el sistema como corazón.

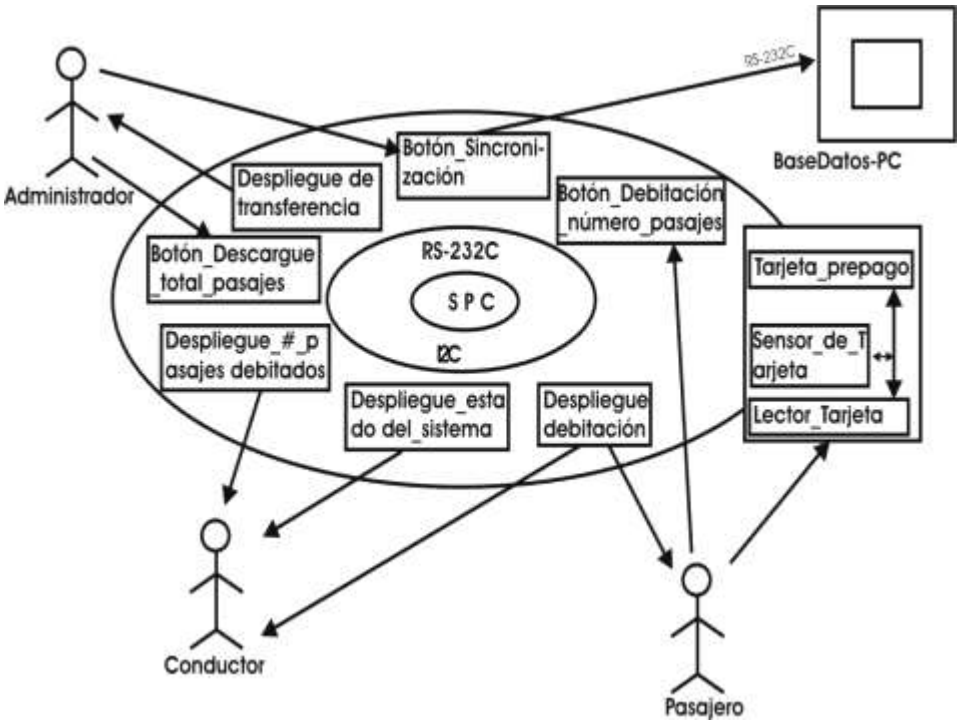

Figura 4 Modelo Contextual Onion Skin

En la figura 5 se muestra el sistema SPC[6] como un diagrama de contexto en el cual el sistema se muestra como una caja negra interaccionando con los actores del sistema.

[6] Sistema Prepago para Colectivos

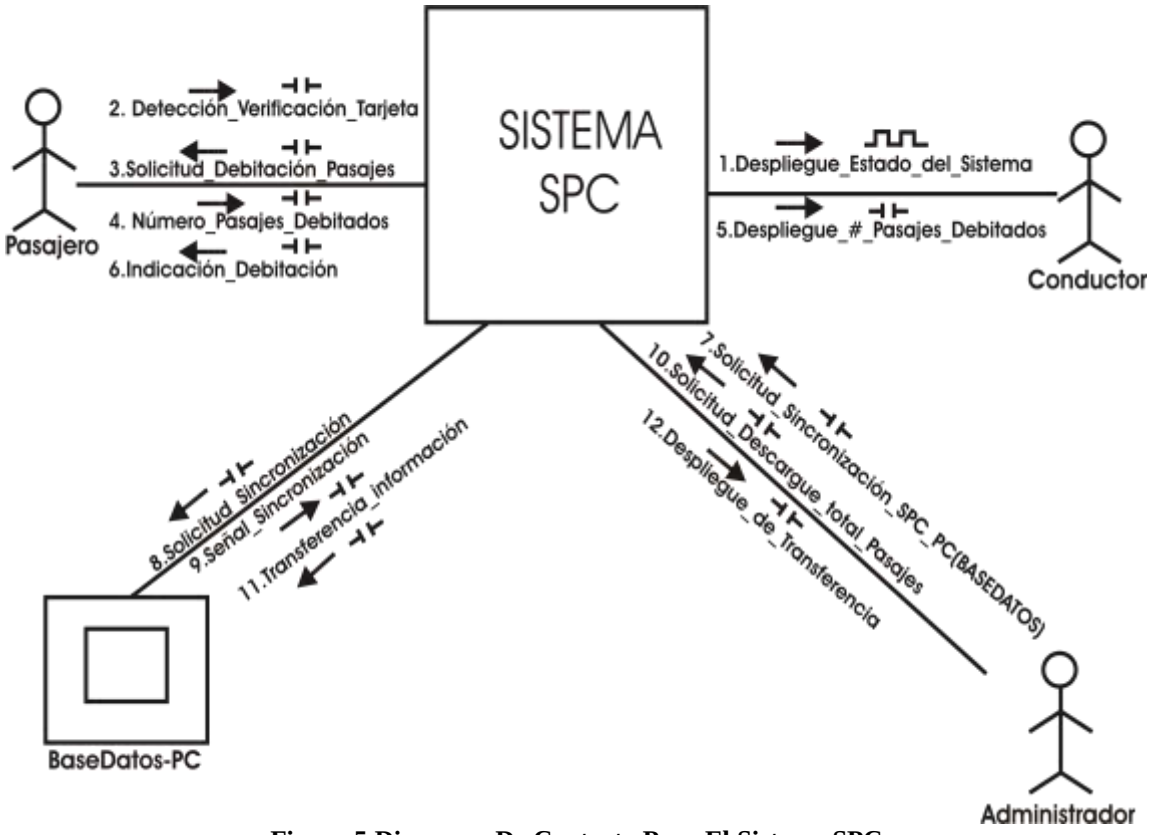

Figura 5 Diagrama De Contexto Para El Sistema SPC

Adicionalmente el diagrama de contexto puede mostrar las capas usadas en el modelo ONION como un conjunto de paquetes UML para agrupar las interfaces de acuerdo a su funcionalidad. La Figura 6 muestra el SPC, en el cual los paquetes se han agrupado por área de funcionalidad dependiendo de si estos hacen parte del corazón del sistema, de las interfaces o de los sensores/actuadores tal como se describe el modelo ONION de Ellis, incluyendo la relación que existe entre los diferentes paquetes, así como los actores que interactúan a través de mensajes con el sistema.

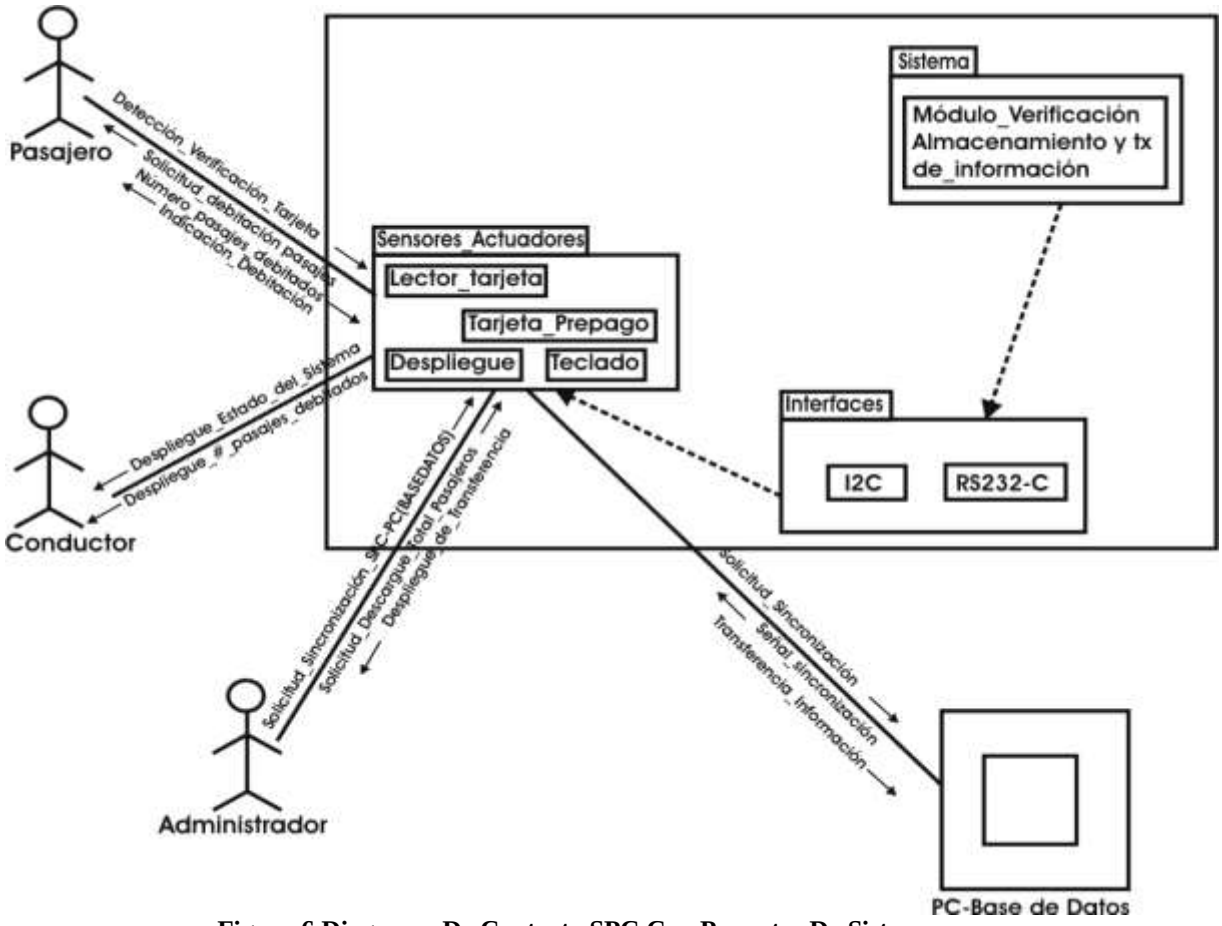

Figura 6 Diagrama De Contexto SPC Con Paquetes De Sistema

2.2.2 VISTA PRINCIPAL DEL SISTEMA

A continuación se presenta la vista del sistema para el usuario, el conductor y el administrador. Este presenta un botón de encendido (***POWER***), un teclado con las teclas de números decimales (**0,1...9**) las cuales permiten realizar la debitación de pasajes, la tecla número (**#**) que permite aceptar la transacción de debitación de pasajes, la tecla asterisco (*****) que permite anular la transacción de debitación, el resto de las teclas se prevén para una aplicación posterior. Lo anterior se observa en la figura 7.

Figura 7 Vista Principal del Sistema

2.2.3 EVENTOS EXTERNOS DEL SISTEMA

Una de las características más importantes en el análisis de requerimientos es el descubrir los eventos externos al sistema, ya que estos definen en gran medida cuales serán las restricciones y el comportamiento del sistema; para el caso del SPC se presenta a continuación la tabla 2 con la lista de eventos externos que son de interés para el sistema, así como las respuestas por parte de este a dichos mensajes, cual es el tiempo que tarda el sistema para responder, los patrones de arribo y la fuente que originó el evento.

EVENTO	RESPUESTA DEL SISTEMA	PATRON DIRECCION	PATRON ARRIBO	SINCRONIZACION	TIEMPO RPUESTA
Pasajero_introduce_Tarjeta	1. Detectar_Tarjeta 2. Verificar_Tarjeta 3. Despliegar_Solicitud_Para debitar_Número_Pasajes.	IN	Episodic	Asíncrono	1/8 S 1/4 S 1/8 S
Pasajero_Pulsa_Botón_Número Pasajes	1. Debitar_el_Valor_correspondiente_al_Número de_Pasajes 2. Acumular _Pasajes Debitados 3. Despliegar_Número_Pasajes_Debitados_Para_Conductor_y_Pasajero 4. Mostrar_Transacción_Exitosa. 5. Listo_Para_Recepción_Nueva_Tarjeta	IN	Episodic	Asíncrono	1/8 S 1/8 S 1/8 S 1/4 S 1/8 S
Administrador_Conecta_Puerto RS232C Con_PC	1. Sincronizar_buffers_Para_Transmisión_de información. 2. Envíar_Información 3. Indicar_Fin_Transmisión 4. Listo_Para_Recepción_Nuevas_Tarjetas	IN	Episodic	Síncrono	5 S

Tabla 2: Eventos Externos Del Sistema

2.2.4 DIAGRAMA DE CASOS DE USO

Una de las herramientas más útiles en el análisis de requerimientos son los diagramas de casos de uso y los diagramas de secuencia, los cuales permiten abstraer el comportamiento real del sistema en el entorno. Es importante resaltar que el sistema prepago para colectivos se ha divido en las capas *SistemaPrepagoP_C* y *SistemaPrepago_PC* (figura 8) con el objetivo de entender mejor el funcionamiento del sistema, adicionalmente las gráficas relacionadas aparecen con un esquema de navegación y documentación propio para la herramienta *Rational Rose RealTime* en el evento que se requiera realizar una comprensión más detallada del sistema prepago para colectivos.

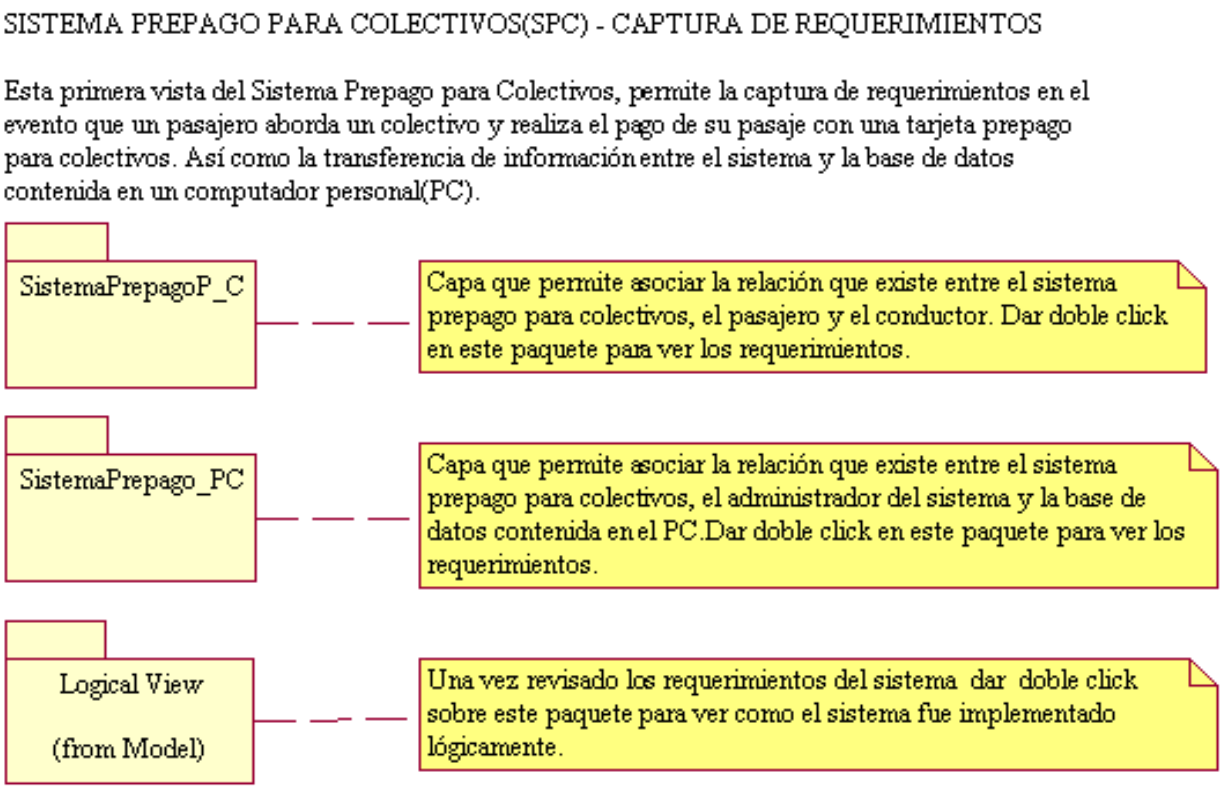

Figura 8 Diagrama De Capas Para El Sistema Prepago Para Colectivos

En la figura 9 se presenta el diagrama de casos de uso para el SPC correspondiente a la capa *SistemaPrepagoP_C*.

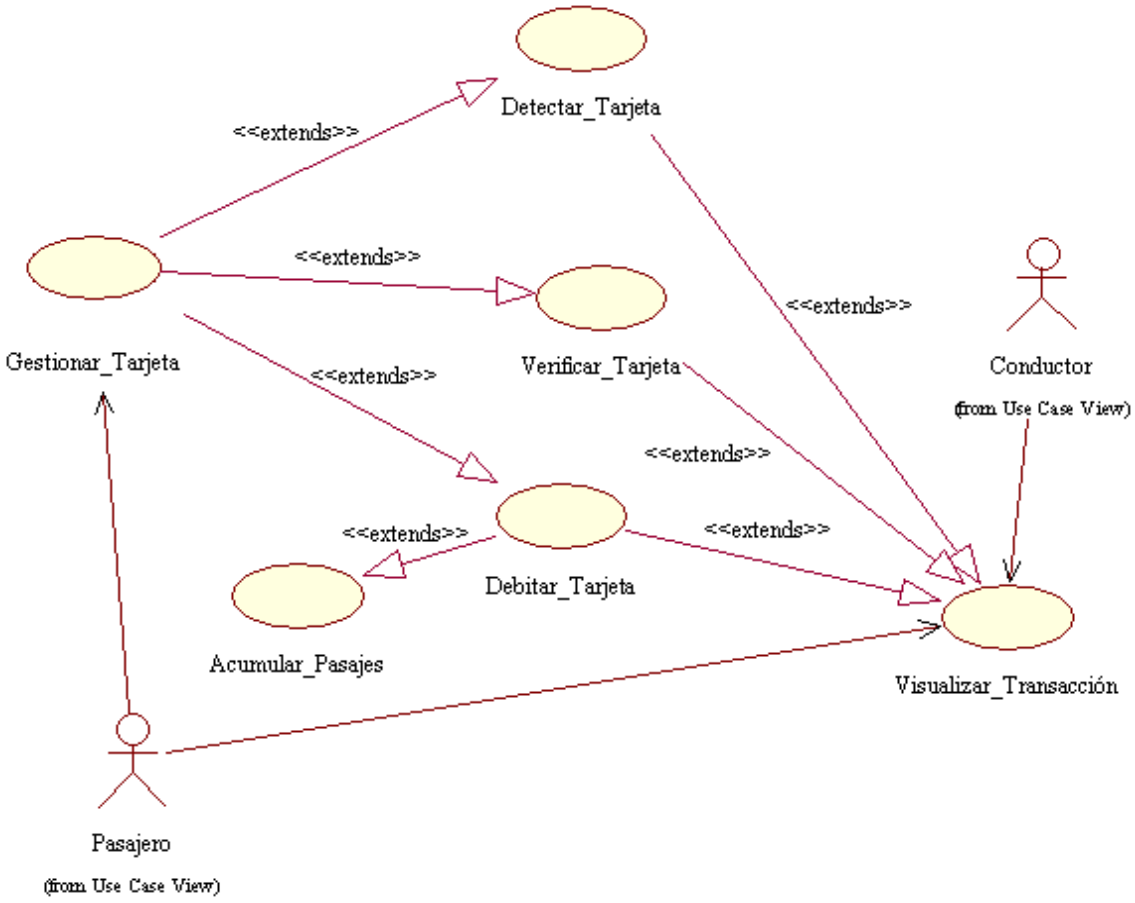

Figura 9 Diagrama De Casos De Uso Capa Sistema PrepagoP_C

En la gráfica 10 se muestra el diagrama de casos de uso para la capa *SistemaPrepago_PC*, donde se puede observar como es la relación entre el administrador, el SPC, la base de datos y el conductor.

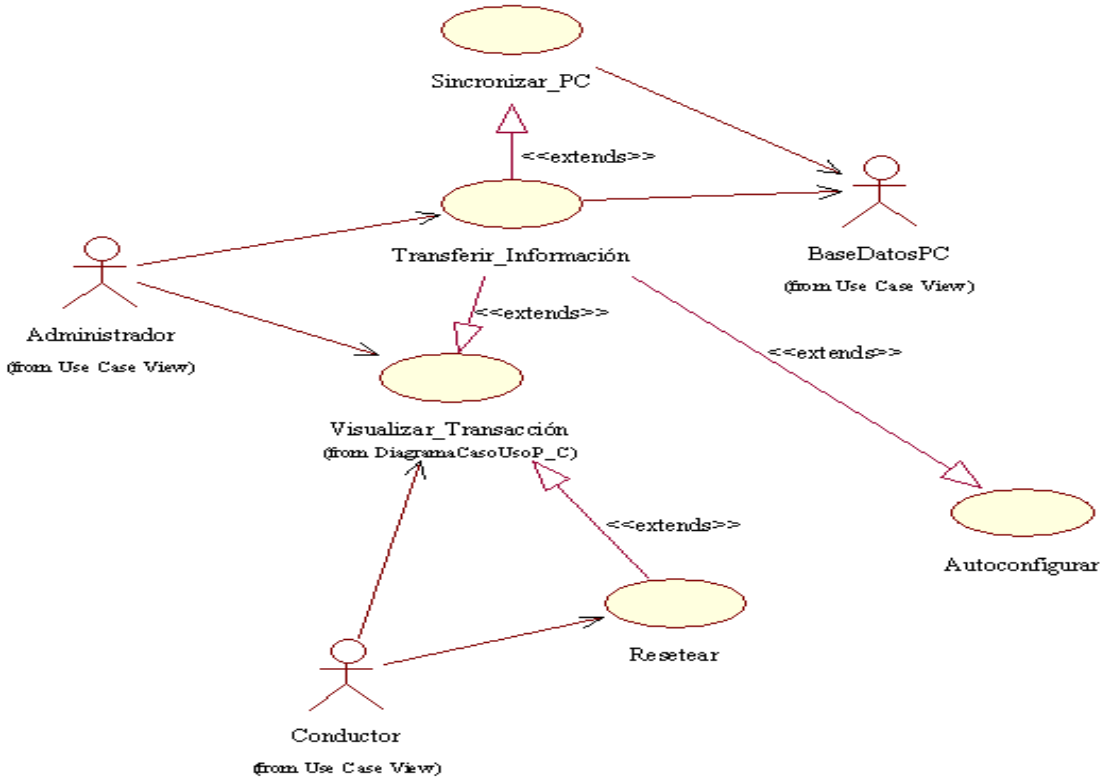

Figura 10 Diagrama De Casos De Uso Capa Sistema Prepago_ PC

2.2.4.1 DEFINICION DE ACTORES

DEFINICION DE ACTORES	
Actor	**Descripción**
Pasajero	Este actor representa a la persona que previamente ha comprado una tarjeta prepago para colectivos y quien la utilizará directamente al transportarse en un microbús de servicio urbano.
Conductor	Este actor representa a la persona que maneja el microbús y quien velará porque se haga un buen uso del servicio, así como que cada pasajero pague su respectivo pasaje.
Administrador	Este actor representa a la persona encargada de interfazar el sistema SPC con un Computador Personal (PC), quien descargará toda la información almacenada en el sistema SPC en un día de trabajo a una base de datos contenida en el PC.
BaseDatosPC	Este actor representa a un sistema encargado de almacenar la información diaria contenida en el sistema SPC, para realizar posteriormente el gestionamiento de la misma.

Tabla 3: Definición De Actores Del Sistema

2.2.4.2 CASOS DE USO DEL SISTEMA

2.2.4.2.1 CAPA SISTEMA PREPAGOP_C

CASO DE USO: GESTIONAR _ TARJETA	
Actores Participantes	Pasajero
Tipo	Primario
Casos de Uso Relacionados	• Detectar_Tarjeta • Verificar_Tarjeta • Debitar _ Tarjeta
Pre-Condiciónes:	1. Sistema Prepago para Colectivos(encendido) 2. Tarjeta Prepago para Colectivos insertada en el lector de tarjetas
Descripción:	Este caso de uso inicia cuando <Pasajero> aborda el colectivo e introduce una tarjeta prepago para colectivos en el lector de tarjetas, el sistema devuelve un mensaje invitando al usuario a debitar el número de pasajes en el evento que la tarjeta sea válida, luego <Pasajero> indica al sistema prepago para colectivos el número de pasajes a debitar y espera el mensaje de confirmación del mismo. Este caso de uso finaliza cuando <Pasajero> retira la tarjeta del lector de tarjetas.

CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El Actor <Pasajero> introduce la Tarjeta Prepago para Colectivos en el Lector de Tarjetas.	2. El SPC detecta la Tarjeta Prepago. 3. El SPC Verifica la Tarjeta Prepago. 4. El SPC despliega un mensaje para la debitación de pasajes.
5. El <Pasajero> indica al sistema el número de pasajes a cancelar	6. En consecuencia el SPC procede a debitar la cantidad de pasajes de la Tarjeta Prepago 7. El SPC almacena en Memoria la cantidad de pasajes debitados. 8. El SPC suministra un mensaje de confirmación de la transacción.
9. Finalmente el <Pasajero> retira la Tarjeta Prepago del Lector de Tarjetas.	10. El SPC queda listo para recepcionar una nueva Tarjeta Prepago.
Pos-Condiciones	1. El SPC deberá contener en memoria el valor actualizado de la suma total de pasajes.

EXCEPCIONES-CURSO ALTERNO 1:TARJETA DEFECTUOSA	
Acción del Actor	**Respuesta del Sistema**
1. El Actor <Pasajero> introduce una Tarjeta Prepago para Colectivos defectuosa en el Lector de Tarjetas.	2. El SPC detecta la Tarjeta Prepago. 3. El SPC Verifica la Tarjeta Prepago. 4. El SPC despliega un mensaje de Tarjeta Prepago Errónea. 5. El SPC invita a introducir una nueva Tarjeta Prepago.

CASO DE USO: GESTIONAR _ TARJETA	
EXCEPCIONES-CURSO ALTERNO 2: TARJETA INVALIDA	
Acción del Actor	**Respuesta del Sistema**
1. El Actor <Pasajero> introduce una Tarjeta Prepago para Colectivos inválida.	2. El SPC detecta la Tarjeta Prepago. 3. El SPC Verifica la Tarjeta Prepago. 4. El SPC despliega un mensaje de invalidez de Tarjeta Prepago. 5. El SPC invita a introducir una nueva Tarjeta Prepago.
EXCEPCIONES-CURSO ALTERNO 3: RETIRAR TARJETA	
Acción del Actor	**Respuesta del Sistema**
1. El Actor <Pasajero> no retira la Tarjeta Prepago para colectivos.	2. El SPC despliega un mensaje indicando que el Pasajero debe retirar la tarjeta.

Tabla 4: Caso De Uso Gestionar_Tarjeta

CASO DE USO: DETECTAR _ TARJETA	
Actores Participantes	
Tipo	Primario y abstracto
Casos de Uso Relacionados	• Gestionar_Tarjeta • Visualizar_Transacción_Tarjeta.
Pre-Condiciónes:	1. La Tarjeta Prepago para Colectivos debe estar insertada en el Lector de Tarjetas. 2. El caso de uso Gestionar _ Tarjeta debe haber iniciado.
Descripción:	Este caso de uso inicia cuando se ha insertado en el Lector de Tarjetas una Tarjeta Prepago para Colectivos, el SPC detecta la presencia de la Tarjeta Prepago mediante un sensor ubicado en el Lector de Tarjetas, le suministra una polarización adecuada y finaliza cuando detecta la ausencia de tarjeta.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC detecta la presencia de la Tarjeta Prepago. 2. El SPC continúa sensando hasta detectar la ausencia de Tarjeta. 3. El SPC queda a la espera de una nueva Tarjeta Prepago para realizar de nuevo el proceso.
Pos-Condiciones	Ninguna

Tabla 5: Caso De Uso Detectar_Tarjeta

CASO DE USO: VERIFICAR _ TARJETA	
Actores Participantes	
Tipo	Primario y abstracto
Casos de Uso Relacionados	• Gestionar_Tarjeta • Visualizar_Transacción_Tarjeta.

CASO DE USO: VERIFICAR _ TARJETA	
Pre-Condiciónes:	1. Tarjeta Prepago para Colectivos insertada en el Lector de Tarjetas. 2. Tarjeta Prepago para Colectivos en buen estado. 3. Tarjeta Prepago para Colectivos polarizada adecuadamente. 4. El caso de uso Gestionar_Tarjeta debe estar en ejecución.
Descripción:	Este caso de uso inicia cuando la Tarjeta Prepago para Colectivos ha sido polarizada correctamente, a continuación el SPC verifica la validez de la Tarjeta Prepago mediante la lectura de un identificador único de tarjeta. El caso de uso finaliza cuando se ha terminado el proceso de comprobación de identificador de Tarjeta Prepago.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC realiza la lectura del código de identificación único de Tarjeta Prepago. 2. El SPC verifica que el código de la Tarjeta Prepago sea válido. 3. El SPC despliega un mensaje ilustrando el resultado de la transacción. 4. El SPC queda a la espera de una nueva Tarjeta Prepago para iniciar un nuevo proceso.
Pos-Condiciones	Ninguna
EXCEPCIONES-CURSO ALTERNO 1:CODIGO INVALIDO	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC realiza la lectura del código de identificación único de Tarjeta Prepago. 2. El SPC encuentra que que el código de la Tarjeta Prepago no es válido. 3. El SPC despliega un mensaje de código inválido de Tarjeta Prepago. 4. El SPC despliega un mensaje para recepcionar una nueva Tarjeta Prepago.
EXCEPCIONES-CURSO ALTERNO 2: TARJETA DEFECTUOSA	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC encuentra que la Tarjeta Prepago para Colectivos se encuentra defectuosa. 2. El SPC despliega un mensaje de Tarjeta Defectuosa. 3. El SPC despliega un mensaje para recepcionar una nueva Tarjeta Prepago.

Tabla 6: Caso De Uso Verificar_Tarjeta

CASO DE USO: DEBITAR _ TARJETA	
Actores Participantes	
Tipo	Primario y abstracto
Casos de Uso Relacionados	• Gestionar_Tarjeta • Visualizar_Transacción_Tarjeta. • Acumular_Pasajes

CASO DE USO: DEBITAR _ TARJETA	
Pre-Condiciónes:	1. Tarjeta Prepago para Colectivos insertada en el lector de tarjetas 2. La Tarjeta Prepago para Colectivos debe tener almacenado por lo menos un (1) pasaje. 3. El caso de uso Gestionar_Tarjeta debe estar en ejecución.
Descripción:	Este caso de uso inicia cuando <Pasajero> le indica al SPC el número de pasajes a debitar, el SPC realiza la lectura de la cantidad de pasajes almacenados en la Tarjeta Prepago, actualiza mediante el proceso de escritura el nuevo valor de pasajes en la Tarjeta Prepago y despliega en pantalla el resultado de la transacción. El caso de uso finaliza cuando se termina el proceso de escritura en la Tarjeta Prepago.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC realiza el proceso de lectura de la cantidad de pasajes almacenados en la Tarjeta Prepago. 2. En consecuencia el SPC resta el número de pasajes especificados por el <Pasajero> al valor leído de la Tarjeta Prepago. 3. A continuación el SPC realiza el proceso de escritura en la Tarjeta Prepago para actualizar el nuevo valor. 4. El SPC almacena en su memoria el número de pasajes especificados por el <Pasajero>. 5. El SPC despliega un mensaje alusivo a la transacción. 6. El SPC queda a la espera de una nueva Tarjeta Prepago.
Pos-Condiciones	Ninguna
EXCEPCIONES-CURSO ALTERNO 1:TARJETA SIN PASAJES SUFICIENTES	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC encuentra que la cantidad de pasajes especificados por el <Pasajero> es superior a la cantidad existente en memoria de la Tarjeta Prepago. 2. El SPC despliega un mensaje de Tarjeta Prepago sin pasajes suficientes. 3. El SPC invita a introducir una nueva Tarjeta Prepago.
EXCEPCIONES-CURSO ALTERNO 2: TARJETA SIN PASAJES	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC encuentra que la Tarjeta Prepago no tiene pasajes. 2. El SPC despliega un mensaje de Tarjeta Prepago sin pasajes. 3. El SPC invita a introducir una nueva Tarjeta Prepago.

Tabla 7: Caso De Uso Debitar_Tarjeta

CASO DE USO: ACUMULAR _ PASAJES	
Actores Participantes	
Tipo	Primario y abstracto
Casos de Uso Relacionados	• Debitar_Tarjeta
Pre-Condiciónes:	1. Caso de uso Debitar_Tarjeta en ejecución.
Descripción:	Este caso de uso inicia cuando el SPC ha terminado el proceso de escritura en la Tarjeta_Prepago, el sistema realiza el almacenamiento en memoria de la cantidad de pasajes especificados por un <Pasajero>. El caso de uso finaliza una vez ha terminado el proceso de escritura en la memoria del SPC.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
	1. El SPC realiza el almacenamiento en memoria de la cantidad de pasajes especificados por un <Pasajero>.
Pos-Condiciones	1. El SPC deberá contener en memoria el valor actualizado de la suma total de pasajes debitados en una jornada de trabajo.

Tabla 8: Caso De Uso Acumular_Pasajes

CASO DE USO: VISUALIZAR _TRANSACCION	
Actores Participantes	Pasajero, Conductor, Administrador
Tipo	Primario y abstracto
Casos de Uso Relacionados	• Detectar _ tarjeta • Debitar_Tarjeta • Verificar _ tarjeta
Pre-Condiciónes:	1. SPC(encendido)
Descripción:	Este caso de uso inicia cuando el actor <Conductor> enciende el SPC, el SPC despliega un mensaje invitando al <Pasajero> a introducir una Tarjeta Prepago en el Lector de Tarjetas, una vez el usuario introduce la Tarjeta Prepago el Sistema despliega un mensaje ilustrando el estado de la Tarjeta Prepago en el evento que ésta se encuentre defectuosa ó la invitación para debitar el número de pasajes a pagar por parte del <Pasajero>, si la transacción de pago de pasaje se ha realizado correctamente el SPC despliega un mensaje alusivo al mismo de lo contrario muestra un mensaje de transacción errónea y se inicia de nuevo el ciclo. Una vez finalizada la jornada de trabajo el Actor <Administrador> indica al SPC realizar la transferencia de la información almacenada en su memoria, el SPC despliega en pantalla la secuencia de la transacción. El caso de uso finaliza cuando el SPC se lleva al estado apagado.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**

CASO DE USO: VISUALIZAR _TRANSACCION	
1. El Actor <Conductor> enciende el SPC.	2. El SPC realiza el proceso de calibración del sistema y despliega un mensaje de invitación para insertar una Tarjeta Prepago.
3. El <Pasajero> inserta la Tarjeta Prepago en el Lector de Tarjetas	4. El SPC despliega un mensaje invitando al <Pasajero> a introducir la cantidad de pasajes a cancelar ó en su defecto un mensaje de Tarjeta Prepago incorrecta.
5. El <Pasajero> indica la cantidad de pasajes a cancelar.	6. El SPC despliega el resultado de la transacción en el evento que se haya podido cancelar el número de pasajes especificados por el usuario ó en su defecto despliega un mensaje de Tarjeta Prepago con pasajes insuficientes ó sin pasajes.
7. El <Pasajero> retira la Tarjeta Prepago	8. El SPC inicia el ciclo con una nueva Tarjeta Prepago.
9. El actor <Administrador> indica al SPC realizar la transferencia de información hacia la base de datos contenida en el PC, mediante la conexión del SPC al PC(Computador Personal).	10. El SPC sincroniza su buffer del puerto de transmisión/Recepción con el puerto de Transmisión/Recepción contenido en el PC que contiene la base de datos. 11. El SPC despliega un mensaje de inicio de Transmisión y otro de fin de Transmisión de la información. 12. El SPC regresa a su estado inicial desplegando un mensaje de invitación a introducir una nueva Tarjeta Prepago.
13. El actor <Conductor> corta el suministro de energía para el SPC.	14. El SPC detiene su funcionamiento normal.
Pos-Condiciones	Ninguna.

Tabla 9: Caso De Uso Visualizar_Transacción

2.2.4.2.2 CAPA SISTEMA PREPAGO_PC

CASO DE USO: SINCRONIZAR _ PC	
Actores Participantes	
Tipo	Primario y abstracto
Casos de Uso Relacionados	• Transferir_Información
Pre-Condiciónes:	1. Caso de uso Transferir_Información debe haber iniciado

CASO DE USO: SINCRONIZAR _ PC	
Descripción:	Este caso de uso inicia cuando el Actor <Administrador> conecta el SPC al PC(computador personal) para llevar a cabo la transferencia de información hacia la Base de Datos contenida en este, a continuación el SPC realiza la sincronización de buffers con la Base de Datos contenida en el PC, establecen las condiciones para la transmisión y recepción de la información. Una vez establecido dicho proceso el SPC se mantiene sincronizado con la Base de Datos hasta que el proceso de transferencia de la información ha terminado con lo cual finaliza el caso de uso.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El Actor <Administrador> indica al SPC que debe iniciar el proceso de transferencia de información hacia la Base de Datos contenida en el PC.	2. El SPC envía una petición de sincronización a la Base de Datos contenida en el PC. 3. El SPC y la Base de Datos establecen las condiciones para el envío y la recepción de la información. 4. El SPC y la Base de Datos mantienen el proceso de comunicación hasta que se dé por terminado el proceso de transferencia de información.
Pos-Condiciones	Ninguna

Tabla 10: Caso De Uso Sincronizar_PC

CASO DE USO: TRANSFERIR _ INFORMACION	
Actores Participantes	Administrador, BaseDatosPC
Tipo	Primario
Casos de Uso Relacionados	• Sincronizar_PC • Visualizar_Transacción • Autoconfigurar
Pre-Condiciónes:	1. SPC conectado con el PC. 2. La Base de Datos del PC estar en ejecución.
Descripción:	Este caso de uso inicia cuando el SPC se ha sincronizado con la Base de Datos contenida en el PC, a continuación el sistema despliega un mensaje para proceder a realizar la transferencia de la información en el evento que los dos sistemas hayan podido establecer el proceso de comunicación. A Continuación se realiza el intercambio de información entre los dos sistemas y el caso de uso finaliza cuando se ha recepcionado el total de la información transmitida.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**

CASO DE USO: TRANSFERIR _ INFORMACION	
	1. El SPC establece los mecanismos de comunicación con la Base de Datos contenida en el PC. 2. A continuación el SPC inicia la transmisión de la información. 3. Finalmente el SPC despliega un mensaje indicando el fin de transmisión de la información.
Pos-Condiciones	1. La información contenida en la memoria del SPC se debe haber actualizado en la Base de Datos contenida en el PC. 2. El SPC deberá restaurar sus registros de memoria a ceros.
EXCEPCIONES-CURSO ALTERNO 1:ERROR_COMUNICACIÓN	
Acción del Actor	**Respuesta del Sistema**
1. El Actor <Administrador> indica al SPC arrancar el proceso de transferencia de información.	2. El SPC no puede establecer la comunicación con la Base de Datos Contenida en el PC. 3. El SPC despliega un mensaje de transferencia errónea.

Tabla 11: Caso De Uso Transferir_Información

CASO DE USO: RESETEAR	
Actores Participantes	Conductor
Tipo	Primario y abstracto
Casos de Uso Relacionados	• Visualizar_Transacción
Pre-Condiciónes:	1. El SPC se encuentra en estado bloqueado
Descripción:	Este caso de uso inicia cuando el Actor <Conductor> indica al SPC se encuentra en estado no operativo y por consiguiente obliga al Sistema a salir de dicho estado enviando un pulso de reseteo de Sistema. A continuación el SPC sale de su estado de bucle infinito no operativo y queda listo para recepcionar una nueva tarjeta. El caso de uso finaliza cuando despliega un mensaje de recepción de nueva tarjeta.
CURSO NORMAL DE LOS EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El Actor <Conductor> obliga al sistema a salir de su estado de bucle infinito no operativo.	2. El SPC sale de su estado no operativo. 3. El SPC borra los buffers de despliegue de información. 4. El SPC queda listo para recepcionar una nueva tarjeta. 5. El SPC despliega un mensaje de aceptación de nueva tarjeta.
Pos-Condiciones	1. El SPC conserva su información contenida en memoria.

Tabla 12: Caso De Uso Resetear

CASO DE USO: AUTOCONFIGURAR		
Actores Participantes		
Tipo	Primario y abstracto	
Casos de Uso Relacionados	• Transferir_Información	
Pre-Condiciónes:	Ninguna	
Descripción:	Este caso de uso inicia cuando el Actor <Conductor> enciende el sistema ó cuando el Actor <Administrador> ha realizado un proceso de descargue de información. A continuación el sistema limpia el contenido de su memoria y clarea los buffers de despliegue. Este caso de uso finaliza cuando el SPC muestra un mensaje de espera por una nueva tarjeta.	
CURSO NORMAL DE LOS EVENTOS		
Acción del Actor		**Respuesta del Sistema**
1. El Actor <Conductor> enciende el sistema		2. El SPC limpia el contenido de su memoria. 3. El SPC clarea los buffers de despliegue. 4. El SPC muestra un mensaje de espera por una nueva tarjeta.
5. El Actor <Administrador> ha confirmado el proceso exitoso de descargue de información.		6. El SPC repite los pasos 2,3 y 4.
Pos-Condiciones		1. Borrado total de los registros de memoria del sistema.

Tabla 13: Caso De Uso Autoconfigurar

2.2.5 DIAGRAMAS DE SECUENCIA

Una vez definido los requerimientos del sistema por medio de los casos de uso, se identifica el conjunto de clases y cápsulas que serán capaces de ejecutar el comportamiento descrito en los casos de uso. A continuación se presenta el diagrama de secuencia para la capa *Sistema PrepagoP_C* y *Sistema Prepago_PC* respectivamente.

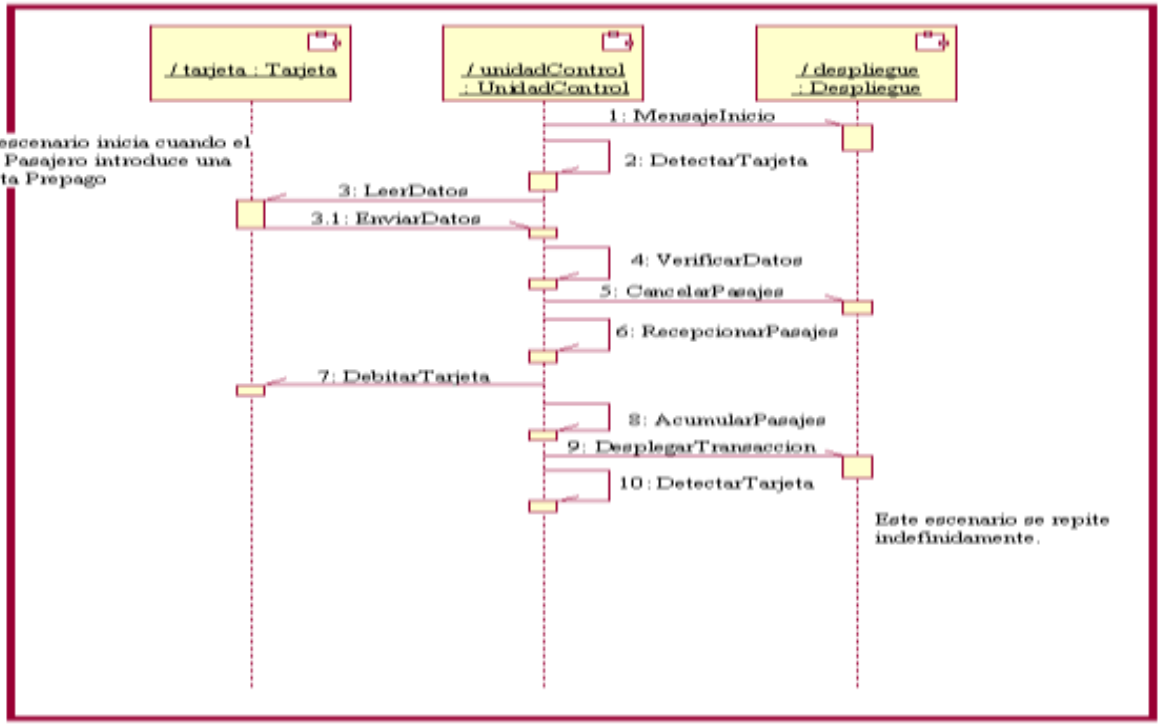

Figura 11 Diagrama De Secuencia Para La Capa Sistema PrepagoP_C

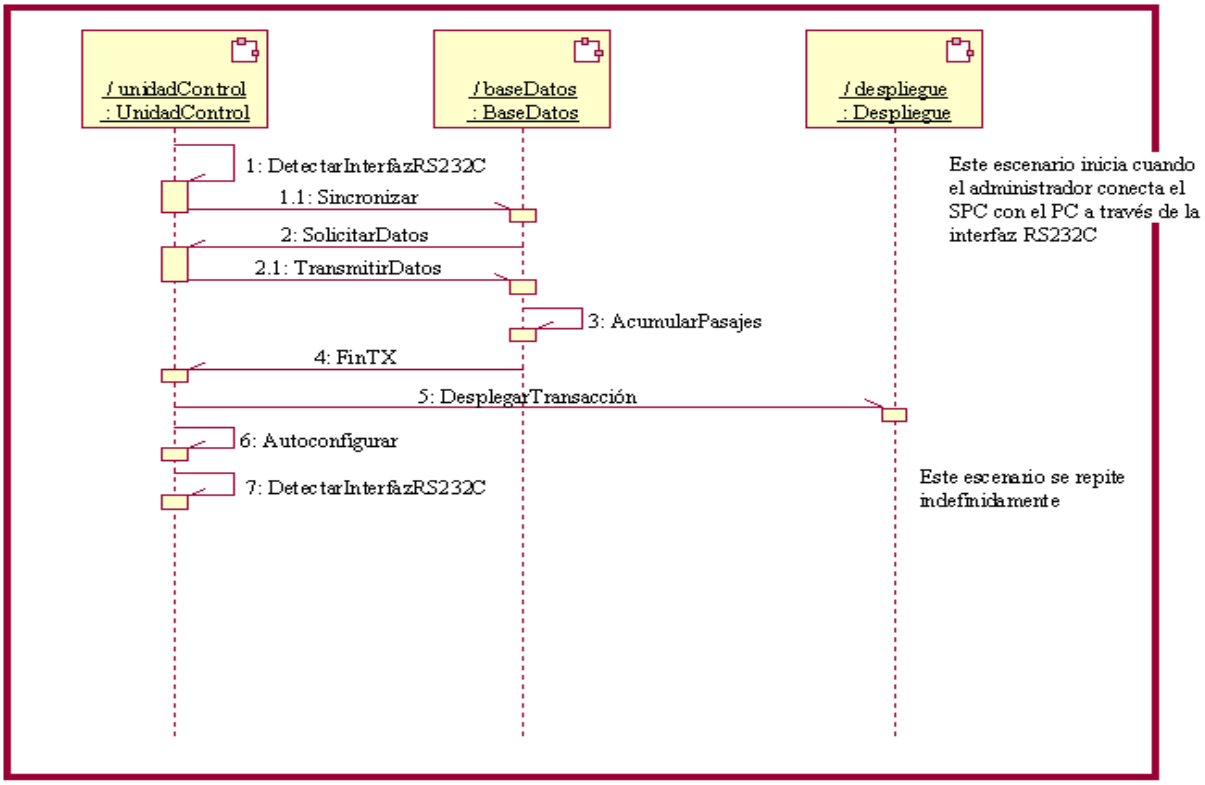

Figura 12 Diagrama De Secuencia Para La Capa Sistema Prepago_PC

3 ANALISIS: DEFINIR LA ESTRUCTURA DE LOS OBJETOS

Una vez finalizado el análisis de requerimientos se procede a la primera fase del análisis "Definir la estructura de los objetos" donde se define la estructura del sistema identificando mediante diversas estrategias los objetos claves del sistema, sus atributos y las operaciones asociadas a estos, se clasifican los objetos de acuerdo a la función que estos desempeñan (activos, persistentes, control) se identifican las asociaciones entre objetos y finalmente se crea el modelo estático del sistema a través del diagrama de clases.

3.1 IDENTIFICACION DE OBJETOS

3.1.1 OBJETOS ACTIVOS

Un objeto activo es aquel que ejecuta acciones autónomamente, coordina las actividades de objetos componentes o genera eventos.

OBJETOS ACTIVOS		
Objeto	**Atributos**	**Operaciones**
Tarjeta	1. Código_Identificación de tarjeta(IdTar). 2. Número de pasajes almacenados(Pasajes).	1. DebitarPasajes() 2. EnviarDatos() 3. LeerDatos()
Lector de Tarjetas	1. TarjetaPresente 2. TarjetaAusente	1. DebitarPasajes() 2. EnviarDatos() 3. LeerDatos()
MVAT	1. Código de Identificación del Módulo MVAT(IdMVAT) 2. Día 3. ValorPasaje 4. PasajesDebitados 5. ValorTotalPasajesDebitados	1. VerificarTarjeta() 2. VerificarDatos() 3. RecepcionarPasajes() 4. AcumularPasajes() 5. DetectarInterfazRS232C 6. DesplegarTransacción() 7. Autoconfigurar 8. TransmitirDatos() 9. Sincronizar()

OBJETOS ACTIVOS		
Interface RS232C	1. EstadoComunicación 2. VelocidadTransmisión	1. Transmitir() 2. Recibir() 3. Sincronizar()
BaseDatosPC	1. NúmeroVehículo 2. Placas 3. Propietario 4. Conductor 5. NúmeroMotor 6. ModeloCarro 7. MarcaVehículo 8. RutasAsignadas 9. Capacidad 10. MovimientoDiario 11. Infracciones 12. Administración 13. Mantenimiento	1. Almacenar() 2. Gestionar() 3. Consultar() 4. GenerarInformes() 5. GenerarEstadísticas() 6. Eliminar()
Despliegue	1. Encendido 2. Apagado 3. Transacción	1. MensajeInicio() 2. CancelarPasajes() 3. DesplegarTransacción()
Teclado	1. ValorTecla	1. GenerarInterrupciónde Tecla()

Tabla 14: Objetos Activos

3.1.2 OBJETOS DE INFORMACION PERSISTENTE

Un objeto de información persistente almacena información no volátil. Dentro de esta categoría se encuentran la *Tarjeta*, el *MVAT* y la *BaseDatosPC*.

3.1.3 OBJETOS DE CONTROL

Un objeto de control es aquella entidad que controla otros objetos, se clasifican como tipos específicos de objetos activos. Dentro de esta categoría para el sistema SPC se encuentran el *Teclado*, el *MVAT* y la *Tarjeta.*

3.1.4 ASOCIACIONES ENTRE OBJETOS

ASOCIACIONES ENTRE OBJETOS		
Fuente del Mensaje	**Destino del Mensaje**	**Mensaje**
Lector_de_Tarjetas	MVAT	DetectarTarjeta
Lector_de_Tarjetas	MVAT	VerificarTarjeta
MVAT	Módulo Despliegue	SolicitarDebitaciónPasajes
Teclado	MVAT	Debitarpasajes
MVAT	Lector_de_Tarjeta	DebitarPasajesTarjeta
MVAT	Despliegue	DesplegarPasajesdebitados
MVAT	MVAT	AcumularPasajesDebitados
MVAT	Despliegue	DesplegarMensajeNueva Transacción
MVAT	BaseDatosPC	TxInformación
BaseDatosPC	MVAT	FinTx
MVAT	Despliegue	FinTx
MVAT	MVAT	Autoconfiguración
MVAT	Modulo_Despliegue	Tarjeta inválida

Tabla 15: Asociaciones Entre Objetos

3.2 DIAGRAMA DE CLASES

Debido a que el SPC se ha divido en capas, la *Figura 13* y la *Figura 14* muestran el diagrama de clases de las capas correspondientes, donde se observan las clases encargadas del almacenamiento y manejo de la información, las cápsulas encargadas de suministrar la coordinación del comportamiento del sistema y los protocolos encargados de permitir la comunicación entre las cápsulas a través de los puertos de estas.

3.2.1 DIAGRAMA DE CLASES CAPA SISTEMA PREPAGOP_C

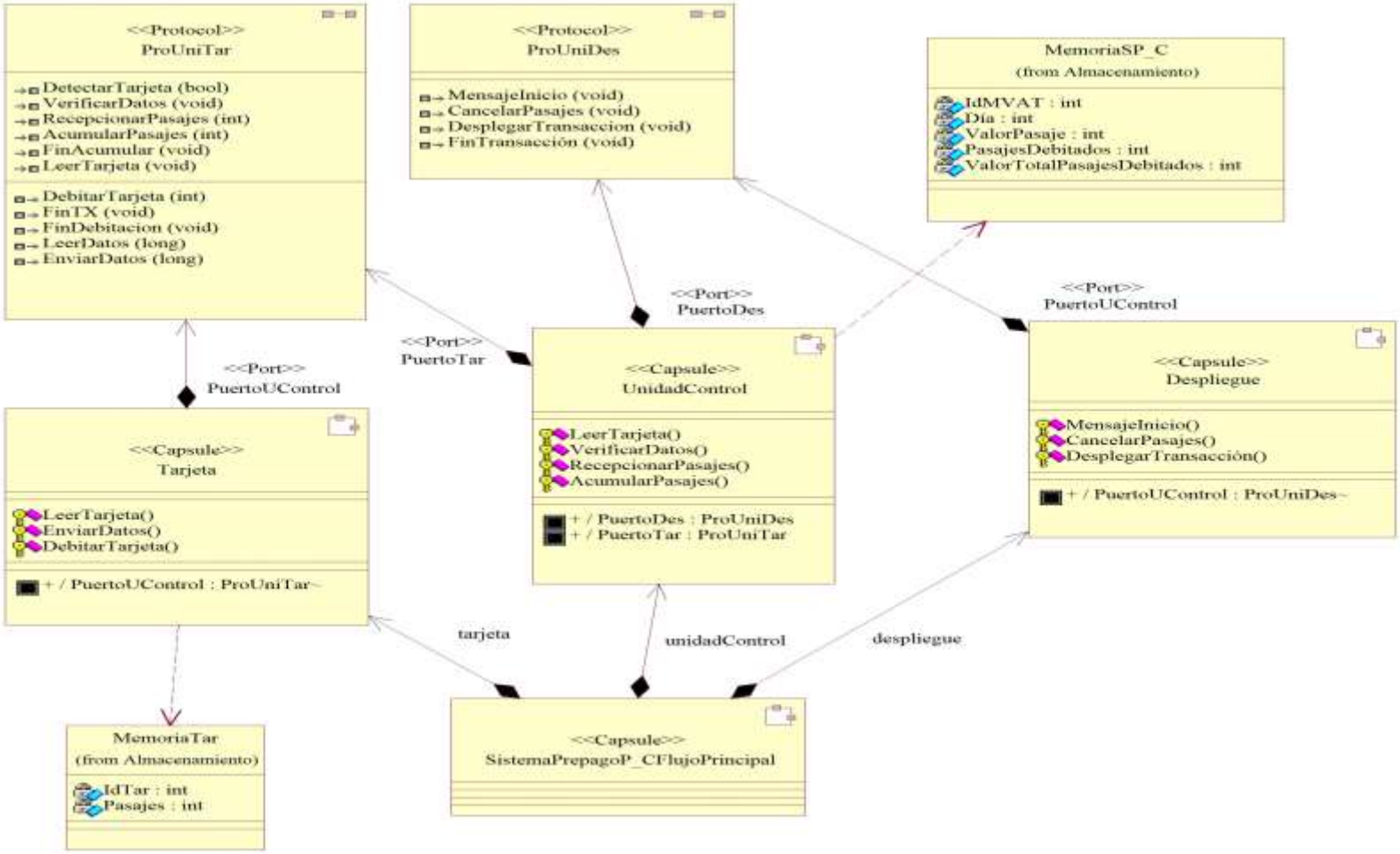

Figura 13 Diagrama De Clases Para La Capa Sistema PrepagoP_C

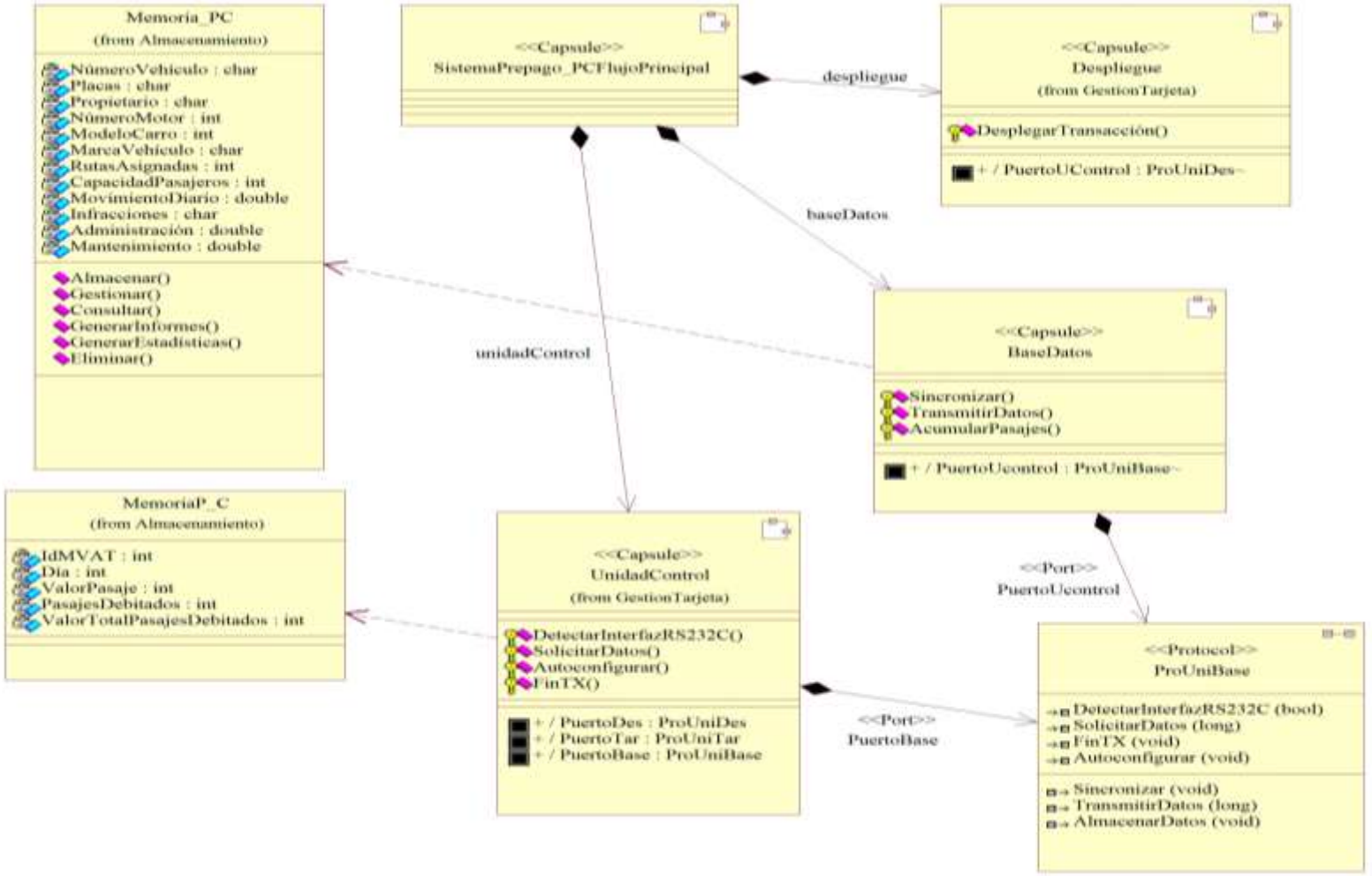

Figura 14 Diagrama De Clases Para La Capa Sistema Prepago_PC

4 ANALISIS: DEFINIR EL COMPORTAMIENTO DE LOS OBJETOS

En esta fase del análisis se refinan las operaciones y los comportamientos, se modelan cada uno de los objetos identificados en la fase anterior con su correspondiente diagrama de estado identificando en este los posibles subestados en los cuales se encontrará un objeto que se encuentra en ejecución. Es importante anotar que como se observa en los diagramas de clases la Capa Sistema PrepagoP_C involucra las cápsulas *Tarjeta*, *UnidadControl* y *Despliegue*, la Capa Sistema Prepago_PC involucra las cápsulas *UnidadControl*, *BaseDatos* y *Despliegue* para las cuales se definirán a continuación sus respectivos diagramas de estado.

4.1 DIAGRAMAS DE ESTADO

4.1.1 CAPA SISTEMA PREPAGOP_C

4.1.1.1 TARJETA

La Cápsula *Tarjeta* (Figura 15) se encuentra en su estado inicial *EsperandoArrancar*, cuando esta recibe un llamado de direccionamiento por parte del módulo MVAT esta pasa al estado *LeyendoDatos*, estado desde el cual lee el identificador de tarjeta (*IdTar*) y el número de pasajes almacenados (*Pasajes*). Una vez realizado este proceso cambia al estado *TransmitiendoDatos* envía los Datos leídos al módulo MVAT para su respectiva verificación, regresando en estos momentos al estado *EsperandoArrancar*.

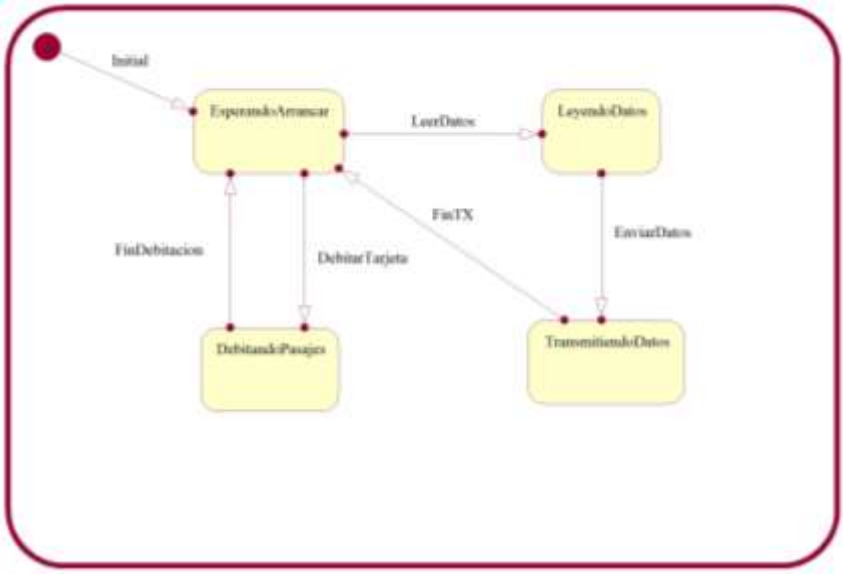

Figura 15 Diagrama De Estados Para La Capsula Tarjeta

Una vez el módulo MVAT ha verificado los datos almacenados en la tarjeta y tras haber recepcionado el número de pasajes a debitar por parte del pasajero, direcciona nuevamente a la Cápsula Tarjeta para proceder a realizar la debitación de pasajes, la cápsula Tarjeta cambia de su estado *EsperandoArrancar* al estado *DebitandoPasajes* permitiendo de esta manera actualizar el nuevo saldo del atributo *pasajes* y regresar al estado inicial *EsperandoArrancar*.

4.1.1.2 UNIDADCONTROL

La Capsula UnidadControl (Figura 16) se encuentra en su estado inicial *DetectandoTarjeta,* una vez se ha detectado una tarjeta la cápsula cambia al estado *LeyendoTarjeta* estado en el cual se envía la palabra de control para la lectura de la tarjeta. Una vez leídos los datos de la tarjeta la cápsula entra al punto de selección

ValidandoTarjeta en la cual se verifican los datos *(IdTar* y *Pasajes*), si el valor de verdad de la condición es

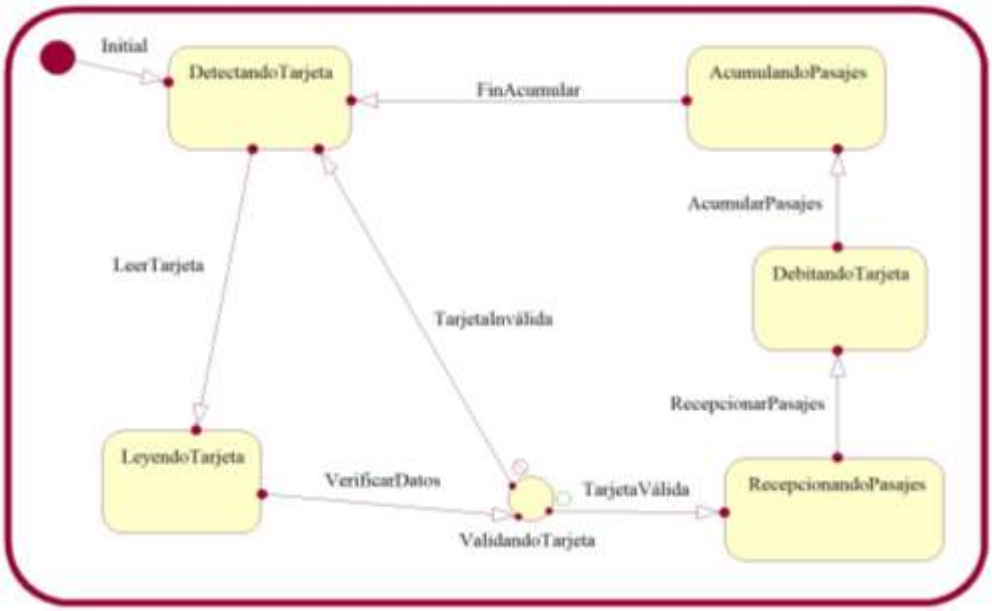

Figura 16 Diagrama De Estados Para La Capsula UnidadControl

falso se regresa al estado inicial *DetectandoTarjeta* por medio del evento de *TarjetaInválida*, si por el contrario el valor de verdad de la condición es verdadero se pasa al estado *RecepcionandoPasajes* mediante el evento *TarjetaVálida*, en este estado la cápsula recepciona un valor entero digitado por el pasajero que corresponde al número de pasajes a cancelar, esto lleva a la cápsula a pasar al estado *DebitandoTarjeta* mediante el evento *RecepcionarPasajes*, estado desde el cual se vuelve a direccionar la memoria utilizando una palabra de control la cual indica que se realice un proceso de escritura y por tanto la actualización del parámetro número de pasajes almacenados en la tarjeta denominado *pasajes*. Seguidamente la cápsula pasa al estado *AcumulandoPasajes*, estado al que cambia mediante la transición *AcumularPasajes* y en

el cual se desarrolla el direccionamiento de la memoria asociada a este estado y que se encuentra contenida en la Clase MemoriaSP_C actualizando los parámetros *PasajesDebitados, ValorTotalPasajesDebitados.* Finalmente, cuando se ha realizado el proceso de escritura en memoria se cambia al estado inicial *DetectandoTarjeta* mediante la transición *FinAcumular*, estado en el que se encuentra listo para recepcionar una nueva tarjeta.

4.1.1.3 DESPLIEGUE

La cápsula despliegue se encuentra en su estado inicial *MensajeIniciando* estado al cual ha entrado debido a que la cápsula *UnidadControl* ha direccionado esta cápsula para indicar la condición de estado del sistema mediante el despliegue de un mensaje de invitación a introducir una tarjeta en el sistema. Una vez el pasajero ha introducido una tarjeta prepago en el sistema y la cápsula *UnidadControl* ha realizado los procesos de lectura y verificación respectivamente, ésta direcciona la cápsula *despliegue* a través de la transición *CancelarPasajes,* evento que obligará a la cápsula a pasar al estado *InvitandoDebitar* desplegando un mensaje que indicará al pasajero que introduzca el número de pasajes a cancelar. Cuando el pasajero ha introducido el número de pasajes ha cancelar se cambia al estado *DesplegandoTransacción* a través del evento *DesplegarTransacción,* el cual muestra el proceso de la transacción y regresa al estado inicial *MensajeIniciando* por medio de la transición *FinTransacción*

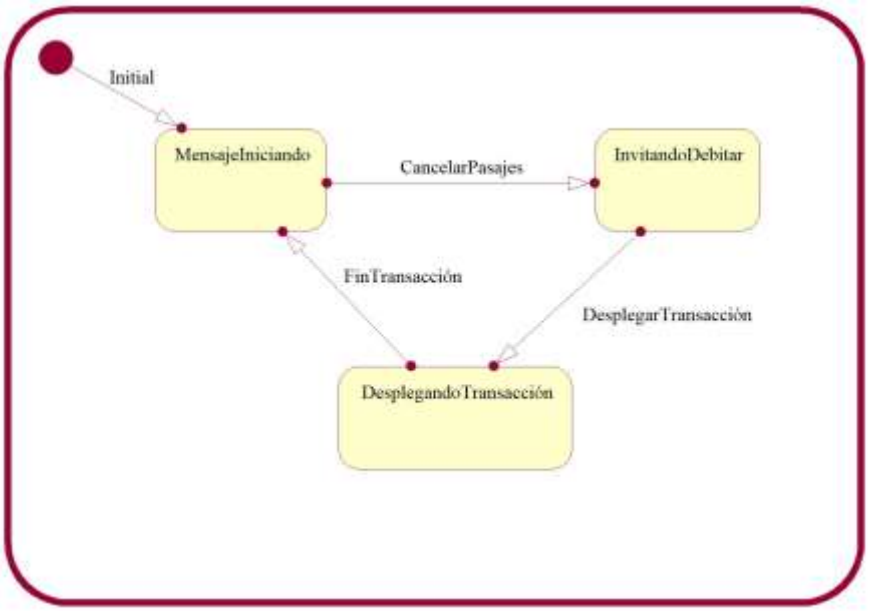

Figura 17 Diagrama De Estados Para La Capsula Despliegue

4.1.2 CAPA SISTEMA PREPAGO_PC

4.1.2.1 UNIDADCONTROL

Esta cápsula se encuentra en su estado inicial *DetectandoInterfazRS232C* hasta que detecta una conexión en el puerto de comunicaciones RS232C lo cual obliga a cambiar al estado *sincronizando*, estableciendo los mecanismos de conexión entre el SPC y la Base de Datos contenida en el Computador Personal, como producto de este estado se pasa al estado condicional *EstableciendoConexión* a través de la transición *conectar*. Si el valor de verdad de la condición es falso se regresa al estado *DetectandoInterfazRS232C* por medio de la transición *ConexiónFallida*, si el valor de verdad de la condición es verdadero se pasa al estado *TransmitiendoDatos*, permitiendo actualizar los campos de información de la Base de Datos contenida en el Computador

Personal. Cuando se ha completado el proceso de transmisión de datos la cápsula cambia al estado *Autoconfigurar* indicado por la transición *FinTX* , en este estado el módulo *MVAT* borra los parámetros *PasajesDebitados y ValorTotalPasajesDebitados* que tiene almacenados en su memoria y regresa al estado inicial *DetectandoInterfazRS232C* a través de la transición *FinAutoconfigurar*.

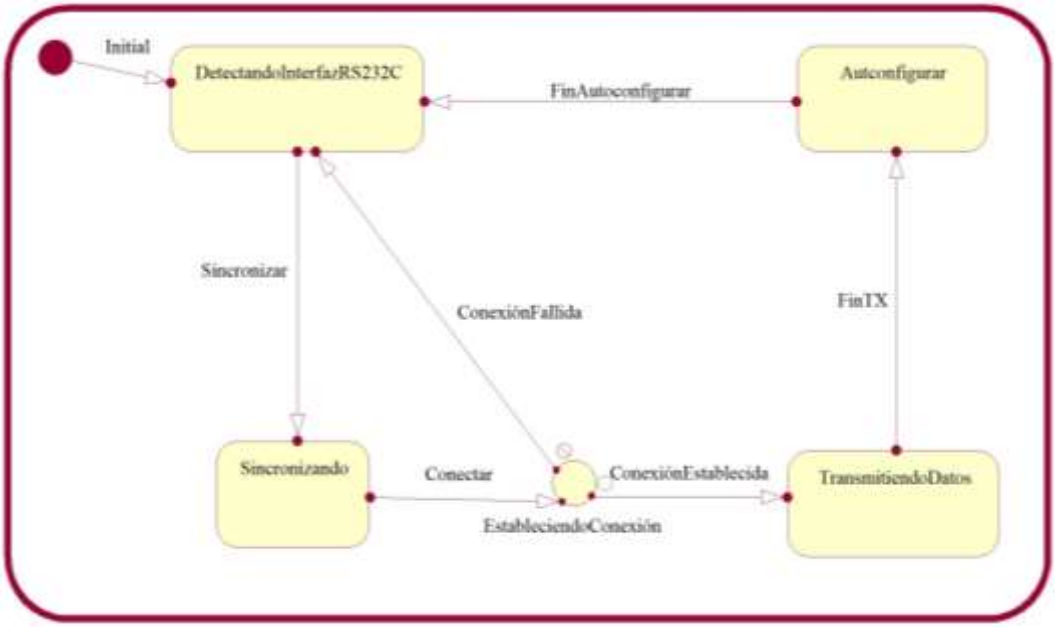

Figura 18 Diagrama De Estados Para La Capsula UnidadControl

4.1.2.2 BASEDATOS

La cápsula BaseDatos se encuentra en su estado inicial *EsperandoSincronismo* y permanece en este estado hasta que recibe la señal *sincronizar*, una vez se ha establecido

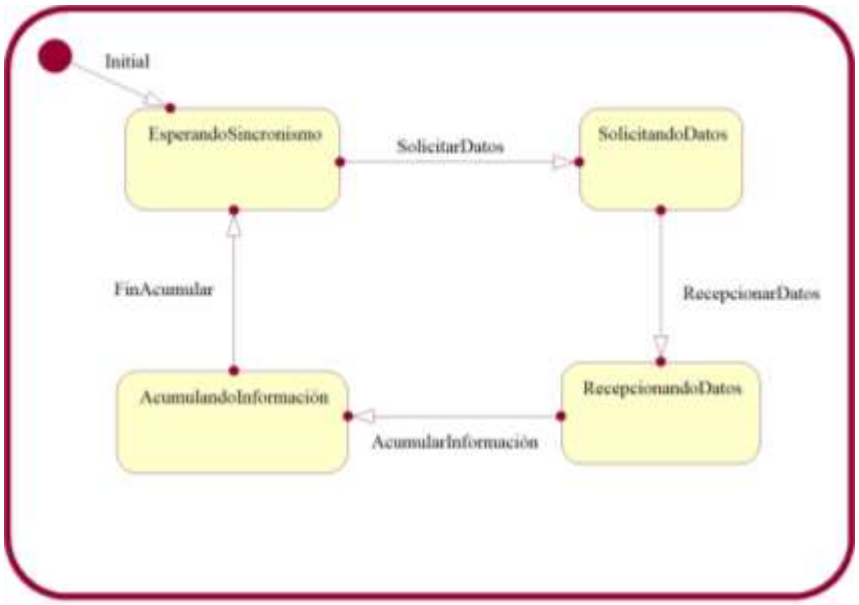

Figura 19 Diagrama De Estados Para La Capsula BaseDatos

la sincronización se pasa al estado *SolicitandoDatos*, estado desde el cual se solicita la información almacenada en la *MemoriaP_C* mediante una palabra de control que direcciona e indica el proceso de lectura. Mediante la transición *RecepcionarDatos* la cápsula cambia al estado *RecepcionandoDatos* estado en el cual permanece hasta que los datos contenidos en la memoria del módulo MVAT son transferidos en su totalidad a la memoria del Computador Personal pasando al estado *AcumulandoInformación* por medio de la transición *AcumularInformación.* En este estado la información es almacenada en los registros de la Base de Datos correspondientes a los parámetros *IdMVAT, Día, ValorPasaje, PasajesDebitados y ValorTotalPasajesDebitados*. Con la actualización de los registros correspondientes a la anterior información se completa el ciclo de almacenamiento de información para este estado y regresa nuevamente al estado inicial *EsperandoSincronismo* a través de la transición *FinAcumular*.

4.1.2.3 DESPLIEGUE

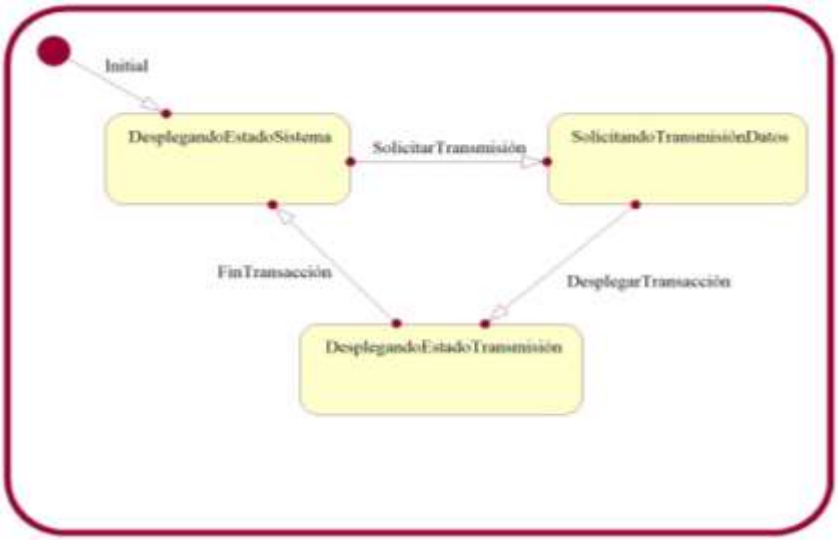

Figura 20 Diagrama De Estados Para La Capsula Despliegue

Inicialmente la cápsula *Despliegue* se encuentra en el estado *DesplegandoEstadoSistema*, es decir que se está mostrando un mensaje en el cual se está invitando a un usuario a introducir una tarjeta. Una vez el módulo *MVAT* ha detectado una interfaz RS232C se cambia al estado *SolicitandoTransmisiónDatos* mediante la transición *SolicitarTransmisión* donde se despliega un mensajes alusivo a la transmisión de los datos. Cuando se ha terminado de realizar la transferencia entre el SPC y la Base de Datos contenida en el Computador Personal se pasa al estado *DesplegandoTransferencia* como respuesta a la transición *DesplegarTransacción*, estado en el que se despliega el mensaje que ilustra el estado de la transacción y finalmente se pasa al estado inicial *DesplegandoEstadoSistema* mediante el evento *FinTransacción* con lo cual se cierra el ciclo de ejecución de esta cápsula y se queda a la

espera de una nueva conexión de la interfaz RS232C entre el SPC y la Base de Datos contenida en el Computador Personal.

5 DISEÑO

5.1 DISEÑO ARQUITECTONICO

En el diseño arquitectónico se define el mapeo de paquetes software a procesadores, selección de protocolos y modelo de concurrencia así como la interrelación existente entre las capas y paquetes identificados.

SISTEMA PREPAGO PARA COLECTIVOS - VISTA LOGICA DEL SOFTWARE

Para efectos de mejorar la comprensión del sistema, este se ha dividido en capas donde cada una de las cuales tiene una función específica y se relaciona con otras prestando básicamente sus servicios.

Las ventajas de tener un sistema dividido en capas, es que cada una de estas tiene una función muy definida y su objetivo es prestar servicios a otras capas en el evento que estas lo requieran. Al tener un sistema dividido por capas a una capa no le interesa como esta implementada otra ya que solo se limita a solicitar o dar servicios.

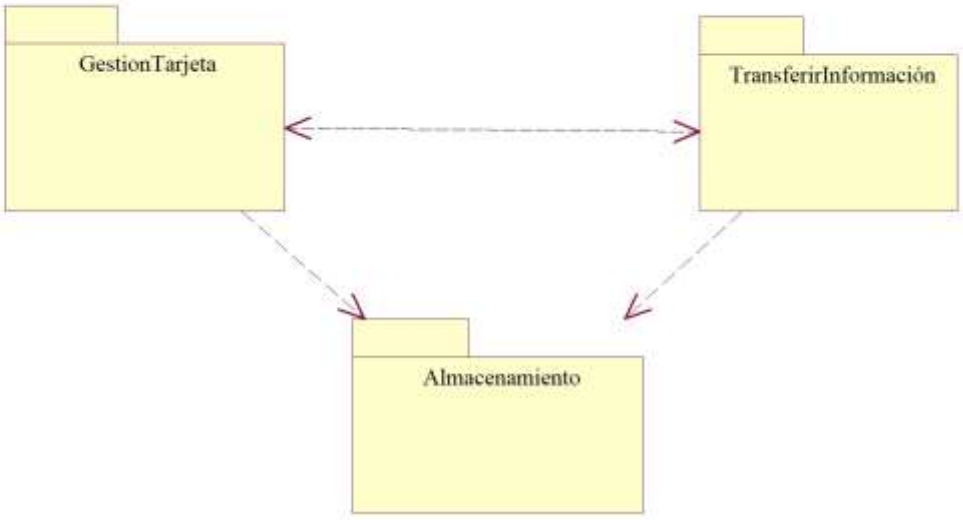

Figura 21 Diagrama De Paquetes Para El Sistema Prepago Para Colectivos

El sistema se ha reducido básicamente a tres paquetes: *GestiónTarjeta*, *TransferirInformación y Almacenamiento*, los cuales se relacionan tal como se muestra en la gráfica anterior.

5.2 GESTIONTARJETA

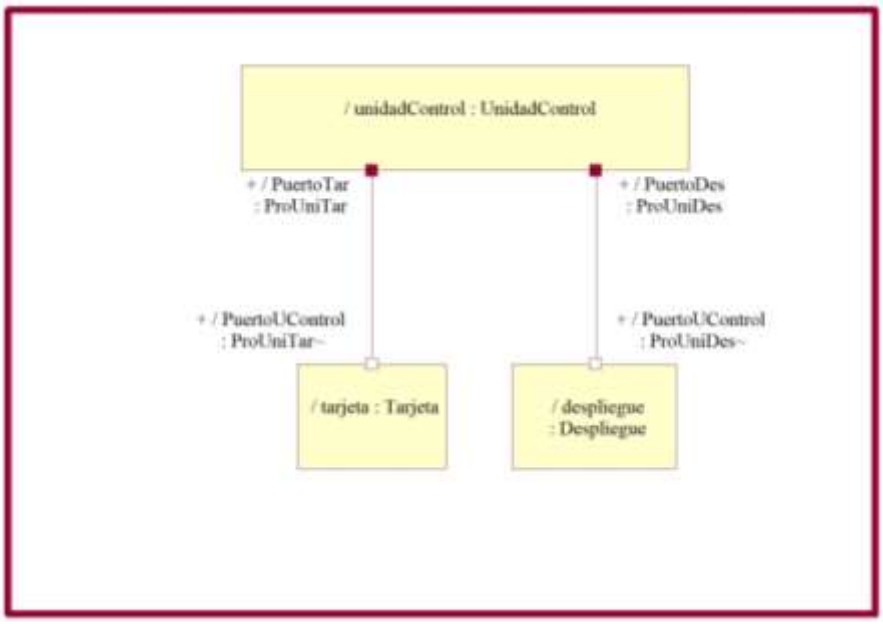

Figura 22 Diagrama De Comunicación Entre Las Capsulas UnidadControl Tarjeta y Despliegue

Este paquete está conformado básicamente por las cápsulas *UnidadControl*, *Despliegue* y *Tarjeta*, así como los protocolos *ProUniDes* y *ProUniTar*. Es importante anotar que las cápsulas se definen como aquellas entidades que permiten observar el comportamiento dinámico del sistema y estas se comunican con otras cápsulas a través de los puertos definidos en cada una de ellas.

La Figura 22 ilustra como están comunicadas las cápsulas mencionadas, a través de sus respectivos puertos y sus respectivos protocolos: La cápsula *UnidadControl* se comunica

con la cápsula *Tarjeta* a través de los puertos *PuertoTar* y *PuertoUControl* utilizando el protocolo *ProUniTar.*

La cápsula *UnidadControl* se comunica con la cápsula *Despliegue* a través de sus puertos *PuertoDes* y *PuertoUControl* respectivamente utilizando el protocolo *ProUniDes.*

Figura 23 Diagrama De Puertos Para La Capsula UnidadControl

En la Figura 23 se observa que la cápsula *UnidadControl* contiene dos puertos finales(End Port[7]) y de tipo públicos a través de los cuales se comunica con las cápsulas *Tarjeta* y *Despliegue,*

[7]Puerto capaz de proveer una conexión entre el comportamiento de la cápsula que contiene el puerto y el resto de objetos del sistema, a través de la recepción y envío de mensajes.

Figura 24 Diagrama De Puertos Para La Capsula Tarjeta

En la Figura 24 se muestra el puerto final para la cápsula *Tarjeta*, pero a diferencia de los puertos para la cápsula UnidadControl se observa que este puerto está conjugado (Conjugated) y alambrado (Wired). La primera propiedad de los puertos permite establecer una comunicación bidireccional entre las cápsulas y la segunda propiedad establece que para poder transferir mensajes entre los puertos estos deben ser conectados a través de un conector tal como se observa en la Figura 22.

Al igual que para la cápsula *Tarjeta* la Figura 25 presenta una connotación muy similar para la cápsula *Despliegue*.

Figura 25 Diagrama De Puertos Para La Capsula Despliegue

5.2.1 VISTA DE COMPONENTES

La vista de componentes especifica como compilar varias partes del modelo, especificando las cápsulas y clases a compilar, como compilar estos elementos, los protocolos asociados, la cápsula de alto nivel sobre la que se va a construir y ejecutar el modelo. De esta forma se creó el componente denominado *ComponenteSPC*, el cual permite validar el diseño asociado a la capa *Sistema PrepagoP_C*, tal como se muestra en el *Anexo 3*.

5.3 TRANSFERIRINFORMACION

Paquete conformado por las cápsulas *UnidadControl*, *Despliegue* y *BaseDatos*, así como los protocolos *ProUniBase* y *ProUniDes*. La Figura 26 ilustra como están comunicadas las cápsulas mencionadas, a través de sus respectivos puertos y sus respectivos protocolos: La cápsula *UnidadControl* se comunica con la cápsula *BaseDatos* a través de los puertos *PuertoBas* y *PuertoUControl* utilizando el protocolo *ProUniBas.*

La cápsula *UnidadControl* se comunica con la cápsula *Despliegue* a través de sus puertos *PuertoDes* y *PuertoUControl* respectivamente utilizando el protocolo *ProUniDes*.

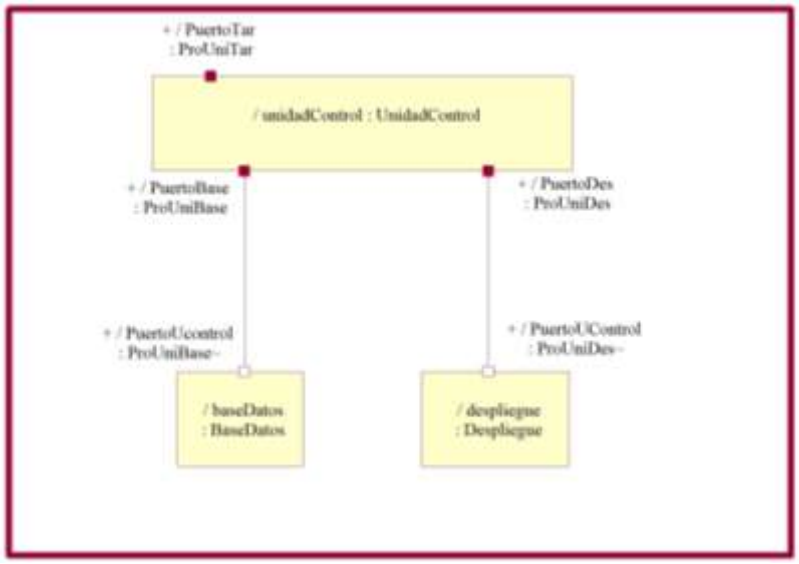

Figura 26 Diagrama De Comunicación Entre Las Capsulas UnidadControl Basedatos y Despliegue

La cápsula BaseDatos se presenta una arquitectura similar a las cápsulas mencionadas, la cual utiliza el puerto *PuertoUcontrol* y como protocolo de comunicaciones *ProUniBase*.

Figura 27 Diagrama De Puertos Para La Capsula Tarjeta

5.3.1 VISTA DE COMPONENTES

La vista de componentes para este paquete y la generación de su correspondiente modelo, tienen el mismo procedimiento que el descrito para el paquete *GestiónTarjeta* por lo cual no se describirá su proceso.

6 IMPLEMENTACION

En este capítulo se describe solamente la implementación del Sistema SPC correspondiente a la capa *Sistema PrepagoP_C*, debido a que los objetivos planteados en el anteproyecto se cumplen implementando esta parte de funcionalidad del sistema.

6.1 HARDWARE

En la Figura 28 se muestra el Diagrama Circuital para el Sistema Prepago para Colectivos SPC, el cual consta básicamente de un microcontrolador *PIC16F877*, dos memorias *24LC08B EEPROM*, un teclado de *4x4*, un *LCD 2x16*, una fuente regulada de 5 voltios de DC compuesta por un *CI LM7805* y un *Puente de Diodos*. Adicionalmente el diagrama muestra otra serie de componentes circuitales de propósito general tales como resistencias, condensadores, transformador de 110 a 6-9 voltios, etc.

6.1.1 MICROCONTROLADOR PIC16F877

Es el cerebro en el funcionamiento del Sistema, pues en el recae toda la responsabilidad de realizar todas las funciones inteligentes del sistema: Atender las interrupciones generadas desde teclado, Detectar la Tarjeta, Verificarla, Debitarla, Mostrar Mensajes en el despliegue LCD alusivos al estado del sistema. En la Figura 2 se muestra el diagrama de pines, resaltando que este microcontrolador tiene incorporado una gran cantidad de

funciones, lo que permite simplificar enormemente la cantidad de dispositivos a incorporar en el sistema. Es así como se utilizó el *Puerto D* para la transmisión de datos hacia el *LCD,* el *Puerto B* para las interrupciones de *Teclado,* Los pines *SDA* y *SCL* para la *Transmisión* y *Sincronización* con las Memorias *EEPROM 24LC08B.*

6.1.2 MEMORIA EEPROM 24LC08B

Dispositivo que permite borrar y escribir eléctricamente sus valores almacenados muchas veces, lo que hace de este un elemento esencial en la construcción de sistemas que se encuentran en desarrollo. En el diagrama circuital mostrado en la Figura 28 se ha utilizado dos memorias una para el almacenamiento de la información del sistema Prepago para Colectivos referenciada como U2 y la otra para la tarjeta prepago del usuario referenciada como U4. A continuación se presenta en la tabla 16 la descripción de pines para estas.

TERMINAL	SÍMBOLO	NOMBRE Y FUNCIÓN
1,2,3	A0,A1,A2	Dirección del Dispositivo en el Bus
4	Vss	Tierra
5	SDA	Datos y Direcciones Seriales I/O
6	SCL	Reloj
7	WP	Protección de escritura. 0 habilita 1 deshabilita
8	Vcc	+5V

Tabla 16 Descripción de Pines Para La Memoria EEPROM 24LC08B

Los pines A2,A1 y A0 se han utilizado para direccionar cada una de las memorias, siendo estas las direcciones 000 para la memoria identificada como U2 y 001 para la memoria identificada como U4. El pin identificado como SDA es utilizado para la transmisión serial de datos en forma bidireccional a través del bus I^2C por lo cual va directamente conectado al pin SDA del microcontrolador. El pin SCL permite recibir los pulsos de reloj generados por el dispositivo maestro (microcontrolador). Finalmente el pin denominado WP se colocó directamente a tierra en ambas memorias para permitir su escritura.

6.1.3 DESPLIEGUE LCD Y TECLADO

El Despliegue LCD y el Teclado son dos dispositivos fundamentales en la construcción del diagrama circuital para el sistema prepago para colectivos SPC, ya que gracias a estos es posible que el usuario se comunique con el sistema y a la vez pueda estar visualizando el estado del sistema. En la Figura 28 se muestra la conexión de dichos dispositivos con el microcontrolador, donde se observa que sus conexiones son estándares y que se manejan para todo tipo de aplicaciones.

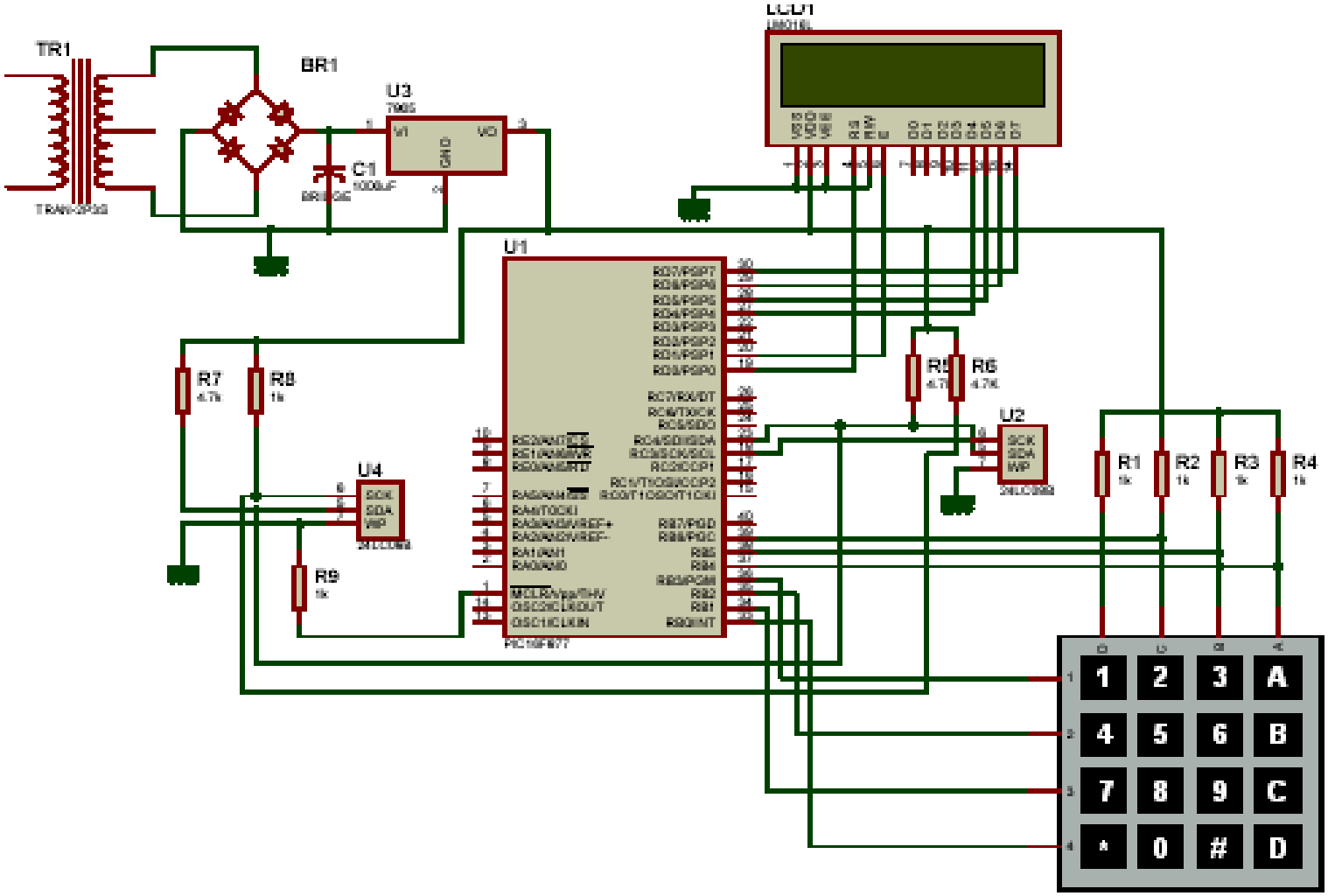

Figura 28 Diagrama Circuital Para El Sistema SPC

6.2 SOFTWARE

Constituido por la aplicación embebida en el microcontrolador PIC16F877 y que se desarrolló con la herramienta para programación de microcontroladores PICBASIC, implementando cada uno de los casos funcionales del sistema para la *Capa Sistema PrepagoP_C*. El código asociado a la aplicación se puede observar en el *Anexo 4*.

7 PRUEBAS

El plan de pruebas establecido para el Sistema SPC, se realizó a través de las fases de Análisis, Diseño e Implementación. Dichas pruebas consistieron en pruebas de: *Caja Negra, Caja Blanca e Integración*.

7.1 PRUEBAS DE CAJA NEGRA

- Se enciende el SPC: Se espera que se muestre un mensaje de Bienvenida y un mensaje de invitación a utilizar el Sistema.

Resultado: Se despliega el mensaje " SISTEMA SPC COLECTIVOS" y el mensaje "Introduzca su Tarjeta".

- Se introduce una Tarjeta: Se espera que el sistema Detecte, Verifique y despliegue un mensaje para el usuario.

Resultado: El sistema detecta la Tarjeta, la Verifica y despliega el mensaje "Introduzca el # de pasajes".

- El usuario introduce desde teclado el número de pasajes a cancelar: Se espera que el sistema despliegue un mensaje indicando el número de pasajes digitados por el usuario, su valor y su nuevo saldo.

Resultado: El sistema Despliega el mensaje *Pasajes: 1 Valor: 650 SaldoPasajes 190*

- El usuario olvida retirar la tarjeta: Se espera que el sistema despliegue un mensaje indicando al usuario que debe retirar la tarjeta.

Resultado: El sistema despliega el mensaje *Retirar **TARJETA**

- El usuario digita un número(122) de pasajes mayor al que contiene su tarjeta: Se espera que el sistema despliegue un mensaje indicando que no tiene suficiente crédito en pasajes.

Resultado: El sistema despliega el mensaje “Pasajes Insufici” Tiene: 61

7.2 PRUEBAS DE CAJA BLANCA

- En la parte de modelamiento del sistema se crea el componente *ComponenteSPC*: Se espera que la herramienta Rational Rose RealTime genere el ejecutable para la cápsula de Alto Nivel.

Resultado: la herramienta genera el mensaje *Build successful* indicando que el componente se ha creado exitosamente (Ver proceso detallado en el *Anexo 5*).

- Se ejecuta la instancia del componente *ComponenteSPCInstance*: Se espera que la herramienta *Rational Rose RealTime* genere un navegador con botones que permita cambiar de estado de la instancia.

Resultado: La herramienta genera la siguiente interfaz

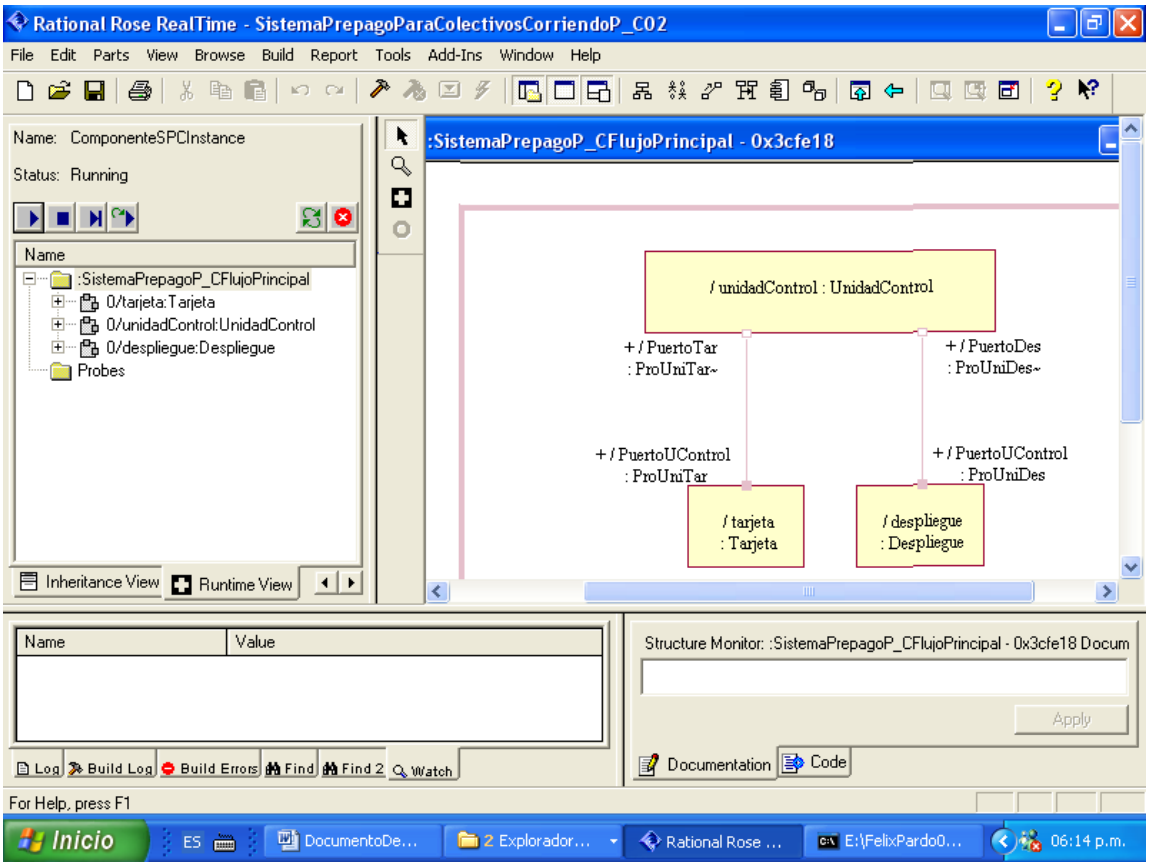

Figura 29 ComponenteSPCInstance

- Se debitan pasajes de la tarjeta de un usuario y se acumulan en memoria: Se espera que el sistema contenga el registro actualizado con el valor correspondiente a la última actualización.

Resultado: Se revisó el contenido de la variable VALOR_MEM y su valor almacenado se despliega en pantalla mostrando el mensaje " Alm 16".

7.3 PRUEBAS DE INTEGRACION

Las pruebas de integración se realizaron básicamente entre el Software embebido que gobierna el sistema y el Hardware que lo compone. Inicialmente se probó el despliegue

LCD para confirmar que este se encontrara en buen estado. Seguidamente se procedió a detectar y configurar el teclado, una vez se estableció la comunicación del teclado con el microcontrolador se confirmó que este respondiera exactamente como lo requería el sistema. Después se procedió a direccionar las memorias y realizar en estas los procesos de lectura y escritura determinando de esta forma que los dispositivos se encontraban en buen estado y funcionando adecuadamente. El resto de las pruebas fueron la integración de los diferentes módulos funcionales del sistema SPC que permitieron poner a punto el sistema.

8 CONCLUSIONES

- Uno de los impactos fundamentales en la creación de un Sistema de Tiempo Real es la utilización de una metodología adecuada en conjunto con una herramienta de modelamiento, matrimonio que permite asegurar un alto flujo de desarrollo en la creación de un sistema. Esto asegura un grado de iteratividad que ayuda a resolver de forma más adecuada los riesgos antes de hacer una gran inversión, permitir la retroalimentación con el usuario y estar realizando constantemente pruebas de integración mediante la creación de prototipos que permitan validar el diseño. Adicionalmente esta unión permite construir un modelo con suficientes detalles para testear el comportamiento aproximado del sistema en un ambiente real y descubrir tempranamente requerimientos ocultos, encontrar y corregir problemas así como adicionar detalles de comportamiento adicional.

- Se encuentra que una de las tendencias más fuertes en el mercado del futuro es la creación de sistemas totalmente autónomos que permitan suministrar un alto nivel de funcionalidad y flexibilidad sin depender de sistemas centrales, los cuales tienden cada día a exigir mejores características físicas de los elementos sobre los cuales se soportan. Esta fuerte competencia hace que las empresas de sistemas embebidos desarrollen tecnologías de punta con altas escalas de integración, lo cual se ve reflejado en dispositivos cada vez más pequeños pero más robustos en su capacidad funcional.

- El Sistema Prepago para Colectivos SPC es un sistema embebido que busca solucionar los problemas existentes en las empresas de transporte público de colectivos para la ciudad de Popayán, con la posibilidad de extenderse hacia el resto de ciudades del País. Aunque el sistema SPC es la primera de las cuatro fases en las cuales se ha dividido el proyecto, se observa que la viabilidad de implantación del mismo en un ambiente real de trabajo representa un reto no sólo en la parte técnica sino también en la parte económica, debido a que se debe tener un sistema de información distribuido en diferentes puntos de la ciudad donde los colectivos harán el descargue de su información, esto con el objetivo de evitar la congestión en los puntos de recolección de información en el momento que los conductores se acerquen con sus vehículos a realizar su respectiva transacción.

9 GLOSARIO DE TERMINOS

ACTOR. Se define como algo que intercambia datos con el sistema, donde puede ser un usuario, Hardware externo u otro sistema.

ACTUADOR. Dispositivo controlado eléctricamente que efectúa una acción mecánica.

CAPSULA. Son los elementos fundamentales de modelamiento en sistemas de tiempo real, estas determinan el comportamiento dinámico del sistema.

CASO DE USO. Descripción de un conjunto de acciones secuenciales que un sistema ejecuta produciendo un efecto significativo para un actor, describe que hace el sistema pero no como está implementado.

COMPONENT INSTANCE. Instancias de componentes, proveen la forma de ejecutar un componente específico ejecutable sobre un procesador específico.

COMPONENTE. Usado para modelar elementos físicos que pueden residir sobre un nodo, tales como clases y cápsulas.

CONJUGATED. Forma más común de establecer una comunicación bidirecional entre dos puertos.

DIAGRAMA DE ESTADO. Muestra la secuencia de estados que un objeto realiza durante su tiempo de vida, como respuesta a mensajes recibidos con sus respuestas y acciones.

DIAGRAMA DE SECUENCIA. Describe en forma gráfica el intercambio de mensajes de forma ordenada entre objetos en tiempo de ejecución.

EEPROM. Tipo de memoria no volátil que es borrable eléctricamente

ESCLAVO. Dispositivo seleccionado por el maestro.

EXTEND. Tipo de relación generalizada utilizada en UML para indicar que un caso de uso base extiende su funcionalidad a través de un caso de uso hijo.

FIRMWARE. Término dado para sistemas que están compuestos por Hardware y Software.

I2C. Inter-Integrated Circuit. Protocolo de comunicación desarrollado por Phillips para cubrir una elevada interconexión entre circuitos integrados.

MAESTRO. Dispositivo que inicia la transferencia, genera la señal de reloj y finaliza la transferencia.

MPLAB. Aplicación desarrollada por la empresa MICROCHIP TECHNOLOGY para la programación de microcontroladores.

ONION DE ELLIS. Modelo que permite categorizar los objetos internos de los externos. Se representa como un modelo de capas donde el sistema es el corazón central, este se relaciona con los sensores/actuadores a través de las interfaces.

PAQUETE. Artefacto utilizado en UML como contenedor de elementos(clases, protocolos, capsulas,etc) del modelo.

PICBASIC. Herramienta que permite la programación y depuración de microcontroladores.

PROCESADOR. Hardware que es capaz de ejecutar programas.

PROTOCOLO. Representan el conjunto de mensajes intercambiados entre dos objetos, permitiendo establecer un patrón de comunicación.

PUERTO. Son objetos que permiten enviar y recibir mensajes entre cápsulas.

PWM. Modo de modulación de anchura de pulsos.

RISC. Computadores de juego de instrucciones reducido.

RS232C. Interface que permite la conexión de dispositivos mediante una comunicación serial estándar.

SCL. Línea de reloj para el bus I2C.

SDA. Línea de datos seriales para el bus I2C.

SPC. Sistema Prepago para Colectivos.

UML. Lenguaje de Modelamiento Unificado.

VALOR_MEM. Variable que contiene un valor entero, correspondiente a la acumulación del total de pasajes debitados por el sistema SPC.

WIRED. Característica que indica que un puerto debe ser conectado a otro a través de un conector.

10 BIBLIOGRAFIA

- BRUCE Powel Douglas. RealTime UML. Developing Efficient Objects for Embedded Systems. Editorial Addison Wesley. 1998.
- BRUCE Powel Douglas. Doing Hard Time. Developing Real-Time Systems With UML, Objects, Frameworks and Patterns. Editorial Addison Wesley. 1998.
- ANGULO Usategui José María, Romero Yesa Susana, Angulo Martinez Ignacio. Microcontroladores PIC. Diseño Práctico de Aplicaciones. Mc-Graw Hill. 2000.
- ANGULO Usategui José María, Angulo Martinez Ignacio. Microcontroladores PIC. 2ª Edición. . Mc-Graw Hill. 1999.
- BRUCE Powel Douglass. Designing Real-Time Systems Whit UML-Part I http://www.embedded.com/98/9803fe2.htm
- BRUCE Powel Douglass. Designing Real-Time Systems Whit UML-Part II http://www.embedded.com/98/9804fe3.htm
- BRUCE Powel Douglass. Designing Real-Time Systems Whit UML-Part III http://www.embedded.com/98/9805fe3.htm.

11 ANEXO 1: GENERALIDADES DEL BUS I2C

Bus cableado serial bidireccional para control eficiente entre circuitos integrados, que permite maximizar la eficiencia del Hardware así como la simplificación de circuitos, además de satisfacer las condiciones de diseños impuestas por diferentes sistemas.

Actualmente Philips incluye más de un centenar de circuitos construidos con diferentes tecnologías (NMOS, CMOS, I^2L) compatibles con el bus I^2C, los cuales traen incorporada una interfaz que permite a estos comunicarse directamente con cualquier otro por medio del bus. Algunas de las características más importantes de los circuitos compatibles con el bus I^2C son las siguientes:

- No necesitan diseñar interfaces de bus debido a que estas ya están integradas dentro de los mismos chips.
- Direccionamiento integrado y protocolos de transferencia de datos permiten al sistema estar definido completamente por Software.
- Los circuitos integrados pueden ser usados indistintamente en muchas aplicaciones diferentes.
- Reducción de tiempo en el diseño, pues los proveedores rápidamente llegan a estar familiarizados con los frecuentes bloques funcionales de los circuitos integrados compatibles con el bus.
- Los circuitos integrados pueden ser adicionados o removidos de un sistema sin que con ello se afecte el normal funcionamiento de otros circuitos integrados incorporados en el sistema.

El bus I^2C, es un bus serial bidireccional desarrollado por Philips para control eficiente entre circuitos integrados, que permite maximizar la eficiencia del Hardware así como la simplificación de circuitos, además de satisfacer las condiciones de diseños impuestas por diferentes sistemas.

El bus I^2C presenta una topología en "bus", donde los datos son trasmitidos serialmente a través de un solo alambre denominado SDA y al cual van conectados los diferentes dispositivos. Adicionalmente los circuitos integrados compatibles con el bus I^2C construidos con tecnología CMOS, ofrecen características especiales de diseño atractivas particularmente para equipos portátiles y sistemas de batería recargables como son: Bajo consumo de corriente, alta inmunidad al ruido, amplio rango de fuente de voltaje y operación a amplios rangos de temperatura. En la Figura 3 se observa la configuración eléctrica básica del bus.

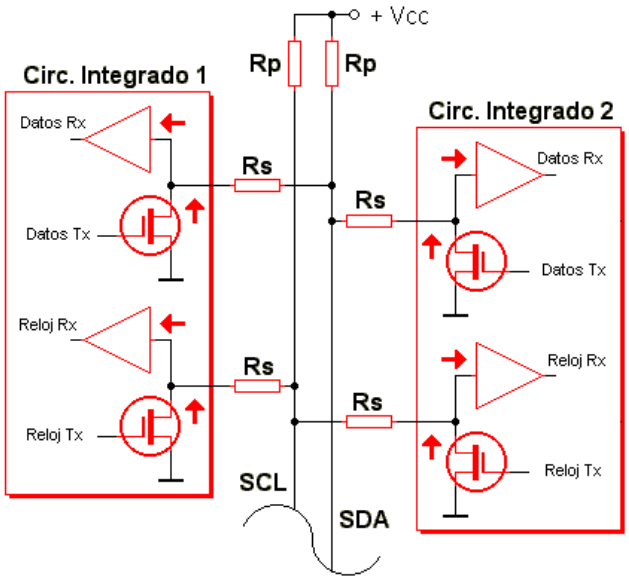

Figura 1 Configuración Eléctrica del bus I2C

11.1 ESPECIFICACION DEL BUS I2C

La especificación del bus I^2C incorpora una serie de normas y procedimientos que permiten la comunicación dentro del sistema, evitando las posibilidades de confusión del bus, pérdida de datos y bloqueo de la información, permite la comunicación de los dispositivos rápidos con los lentos, define que circuito controlará el bus y como lo hará, además de permitir conectar dispositivos de diferentes velocidades definiendo una fuente de reloj para el bus.

Las principales características del bus I^2C se presentan a continuación:

- Son requeridas solamente dos líneas de bus; una línea de datos seriales (SDA) y una línea serial de reloj (SCL).
- Cada dispositivo conectado al bus es direccionado por software con una única dirección ya sea un microordenador, un excitador LCD, una memoria, un adaptador de teclado, etc., donde puede operar como transmisor o como receptor dependiendo de la función del circuito integrado que se considere.
- Este es un verdadero bus multimaestro, incluye detección de colisiones y chequeo para prevenir la pérdida de datos si dos o más maestros inician simultáneamente una transferencia de datos.
- Bus serial orientado a 8 bits, donde la transferencia bidireccional puede ser hecha hasta 100 Kbps en el modo estándar, 400 Kbps en el modo rápido y 3.4 Mbps en el modo de alta velocidad.

Existen supresores de picos en el chip que permiten filtrar la línea de datos del bus, para conservar la integridad de los datos.

El número de Circuitos integrados que pueden ser conectados al bus está limitado solamente por una capacitancia máxima de bus de 400 pf.

Las dos líneas de bus SDA y SCL son bidireccionales y se encuentran conectadas a una fuente positiva de voltaje por medio de unas resistencias de <<Pull-up>>. Cuando el bus está libre, ambas se encuentran en estado ALTO. El bus I^2C es multimaestro lo cual implica que diferentes dispositivos capaces de controlar el bus puedan iniciar una transferencia de datos al mismo tiempo, para lo cual se ha desarrollado un procedimiento de arbitración que permite evitar el caos, conocida como la función << Y-cableada (Wired-And)>> que realiza la conexión de todos los circuitos integrados que se conectan al bus.

Las etapas de salida de todos los dispositivos conectados al bus deben ser drenador abierto o colector abierto para poder realizar la función <<Y-cableada>>. La Figura 2 ilustra la conexión típica de dispositivos de modo estándar y rápido al bus I^2C.

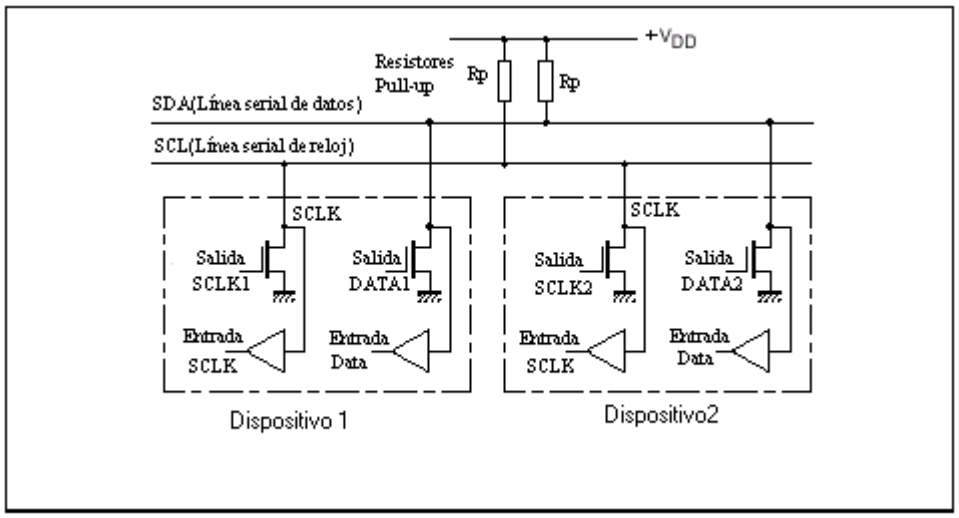

Figura 2 Conexión de dispositivos de modo estándar y rápido al bus I2C

11.2 TRANSFERENCIA DE DATOS

Debido a que los dispositivos que se pueden conectar al bus están construidos de diferentes tecnologías los niveles lógicos “0” (bajo) y “1” (alto) no son fijos y dependen del nivel de polarización asociado con cada tecnología. Por cada bit de datos transferidos se genera un pulso de reloj. Los datos sobre la línea SDA son válidos solamente cuando la línea de reloj se encuentra en estado alto y estos son estables en dicho periodo. La línea de datos SDA solo puede cambiar cuando la línea de reloj SCL se encuentra en estado bajo.

Al margen de lo mencionado anteriormente aparecen las condiciones de arranque y parada, una condición de arranque se define como una transición de ALTO a BAJO de la línea SDA cuando la línea SCL se encuentra en estado alto, mientras que una transición de BAJO a ALTO de la línea SDA cuando la línea SCL se encuentra en estado alto indica una condición de parada.

Las dos condiciones mencionadas anteriormente son generadas por los maestros, donde el bus se considera ocupado cuando se establece una condición de inicio y se considera libre un tiempo después de la condición de parada.

La transferencia de datos está caracterizada por un formato de bytes, lo cual implica que cada dato transmitido sobre la línea de datos SDA debe tener una longitud de 8 bits. La cantidad de octetos que pueden ser transmitidos es indefinido, pero después de cada uno de ellos debe seguir un bit de Reconocimiento. Dado el caso que un dispositivo no pueda recibir otro octeto por encontrarse realizando cualquier otra función, como atendiendo

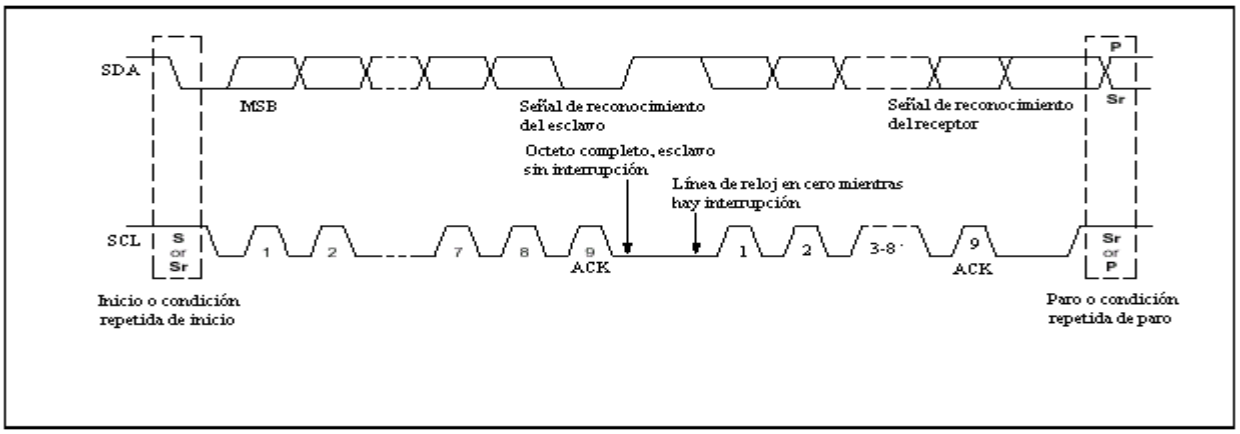

Figura 3 Transferencia de Datos Sobre el Bus I2C

una interrupción interna, este forza la línea de reloj SCL a permanecer en estado bajo para hacer que el transmisor pase a un estado de espera, hasta que el receptor esté listo para recibir otro octeto y deje libre la línea de reloj SCL. La Figura 3 representa la transferencia de datos en el bus I2C.

11.2.1 RECONOCIMIENTO

La transferencia de datos con reconocimiento es obligatoria, donde el encargado de generar el pulso de reconocimiento es el maestro. Durante la generación de este bit el transmisor deja su línea de datos libre (SDA en estado alto) y el receptor forza su línea de datos SDA a un nivel bajo estable.

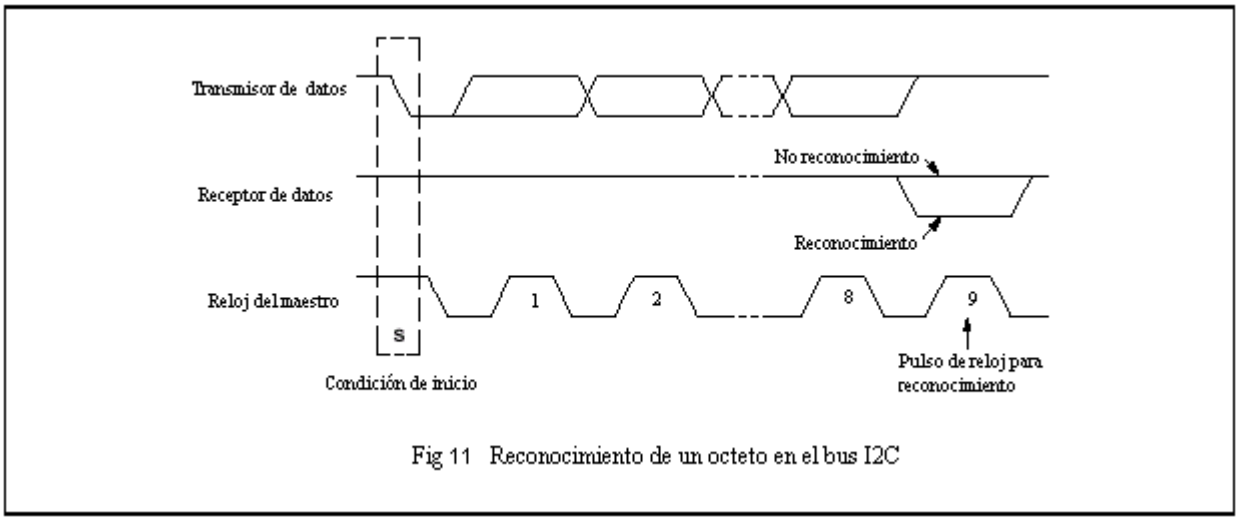

Figura 4 Reconocimiento de un Octeto en el Bus I2C

Usualmente un receptor que ha sido direccionado está obligado a generar un reconocimiento después de haber recibido un octeto, a excepción de los mensajes que arrancan con una dirección RC-5 o CBUS. Si un receptor esclavo no genera la condición de reconocimiento el maestro corta la transferencia de datos mediante una condición de paro, dicha condición se debe a lo siguientes aspectos:

- Que el receptor esclavo no reconoce su propia dirección de esclavo(puede encontrarse realizando alguna función en tiempo real), luego la línea de datos se encuentra en estado ALTO.

- Que el receptor ha reconocido como suya la dirección de esclavo, pero cierto tiempo después no puede recibir más octetos de datos, por lo cual la línea de datos queda en estado ALTO.
- Si el involucrado en una transferencia es el maestro, éste indica el fin de la transmisión de datos al esclavo transmisor mediante la omisión del bit de reconocimiento al recibir el último octeto de datos, condición que permite que el esclavo transmisor deje libre la línea de datos(SDA en alto) y el maestro pueda generar la condición de paro.

La Figura 4 ilustra los estados del receptor y del transmisor en el momento del reconocimiento.

11.3 SINCRONISMO Y ARBITRACION

Debido a la naturaleza de bus multimaestro, dos o más maestros conectados al bus I^2C pueden arrancar a transmitir datos al mismo tiempo para lo cual cada uno de ellos debe haber definido previamente su propia señal de reloj (*sincronismo*) y el mecanismo que permita ganar el acceso al bus denominado *arbitración*. Ambos procedimientos son realizados utilizando la función <<Y-cableada>>, sobre la línea SCL y SDA para sincronismo y arbitración respectivamente.

11.3.1 SINCRONISMO

Proceso mediante el cual cada maestro conectado al bus I^2C genera sus propios impulsos de reloj en la línea SCL para poder arrancar la transmisión de mensajes a través del bus.

Dicha sincronización está basada sobre la función << Y-cableada>> entre los dispositivos conectados y la línea SCL de la siguiente manera:

Una transición de ALTO a BAJO sobre la línea SCL implica a todos los dispositivos involucrados, indicándoles que se ha iniciado su periodo bajo donde cada dispositivo mantendrá la línea SCL en estado bajo el tiempo que le imponga su propia frecuencia de operación. Si un maestro cambia su señal de reloj de BAJO a ALTO, no condiciona el cambio de la línea SCL, pues al existir otro maestro con periodo de operación más bajo mantendrá la línea SCL en estado bajo. Los dispositivos con periodo bajo más largo quedan en estado ALTO de espera durante este periodo.

Una vez se ha terminado el periodo BAJO de todos los maestros que se encuentran implicados, la línea de reloj es liberada y pasa al estado ALTO razón por la cual los estados de reloj y de línea SCL son iguales, por tanto los dispositivos entienden que se ha iniciado su periodo ALTO, el dispositivo con periodo alto más corto agotará más rápidamente este estado y forzará nuevamente a la línea SCL al estado bajo, de esta forma se genera un impulso de reloj sincronizado donde el maestro con periodo de reloj BAJO más largo determina el periodo BAJO y el maestro con periodo de reloj ALTO más corto determina el periodo ALTO (Ver Figura 5).

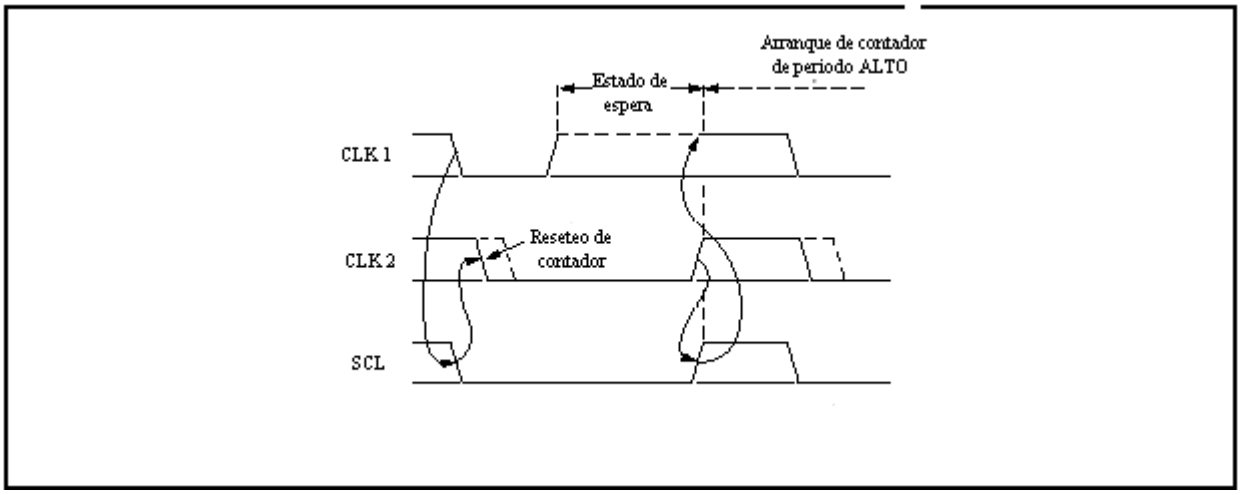

Figura 5 Sincronización de Reloj Durante el Procedimiento de Arbitración

11.3.2 ARBITRACION

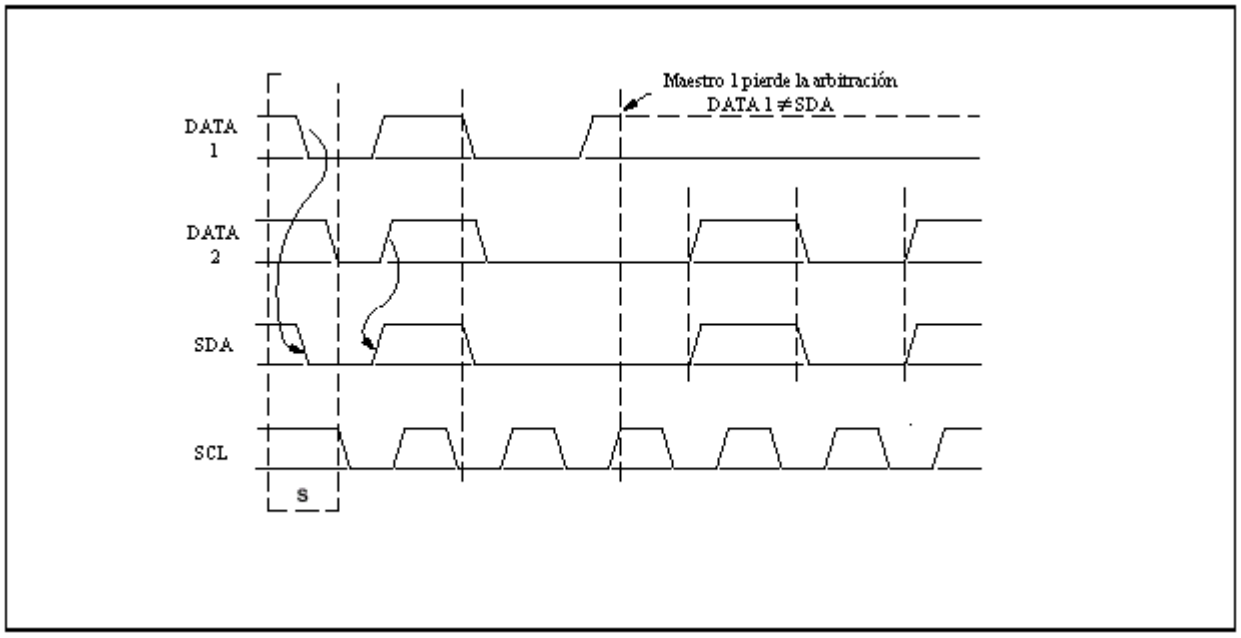

Figura 6 Procedimiento de Arbitración Entre Dos Maestros

Proceso que permite evitar el caos una vez que dos o más dispositivos maestros intenten iniciar una transferencia de datos al mismo tiempo, un maestro solo puede arrancar una transferencia de datos cuando el bus se encuentra libre.

Cuando esta situación ocurre el proceso denominado arbitración tiene lugar sobre la línea SDA, de tal forma que si dos maestros transmiten un nivel ALTO y un nivel BAJO respectivamente, el maestro que transmite el nivel ALTO desconectará su etapa de salida de datos debido a que el nivel sobre la línea SDA es diferente al de su propio nivel de datos. La primera fase de arbitración consiste en la comparación de los bits de dirección del dispositivo que se quiere direccionar, si ocurre el caso que los dos maestros estén intentando redireccionar el mismo dispositivo la siguiente fase consiste en la comparación de los datos, donde los datos que son enviados por el maestro ganador y utilizados en la arbitración no se pierden. Si se encuentran diferentes maestros conectados al bus el proceso de arbitración es el mismo que se describió anteriormente para dos maestros, donde la arbitración se determina por la comparación del nivel de datos interno del maestro que genera los datos y el nivel real en la línea SDA, los maestros que pierden la arbitración ponen su nivel de salida en ALTO (colector abierto o drenador abierto) en el bus, por lo cual no se afectan los datos transmitidos por el maestro ganador. Es posible que un maestro perdedor tenga una función esclava la cual puede estar siendo direccionada por el maestro ganador, lo cual obliga al maestro perdedor a conmutar a su estado de receptor esclavo. Lo anterior determina que sobre el bus I^2C no hay ningún orden de prioridades o un maestro central que determine el control sobre el bus.

11.4 FORMATOS CON 7 BITS DE DIRECCIONES

Una vez definidos los procesos de sincronismo y arbitración continúa la transferencia de datos que siguen el formato mostrado por la Figura 7. La secuencia de transmisión de datos arranca con una condición de inicio, después de esta se envía la dirección del esclavo que está constituida por 7 bits y el octavo bit representa la dirección de datos (R/W). Un "cero" indica una transmisión (Escritura) y un "Uno" una petición de datos (Lectura), el marco termina con la condición de parada generada por el maestro, sin embargo un maestro puede direccionar otro esclavo sin generar la condición de parada mediante otra condición de inicio.

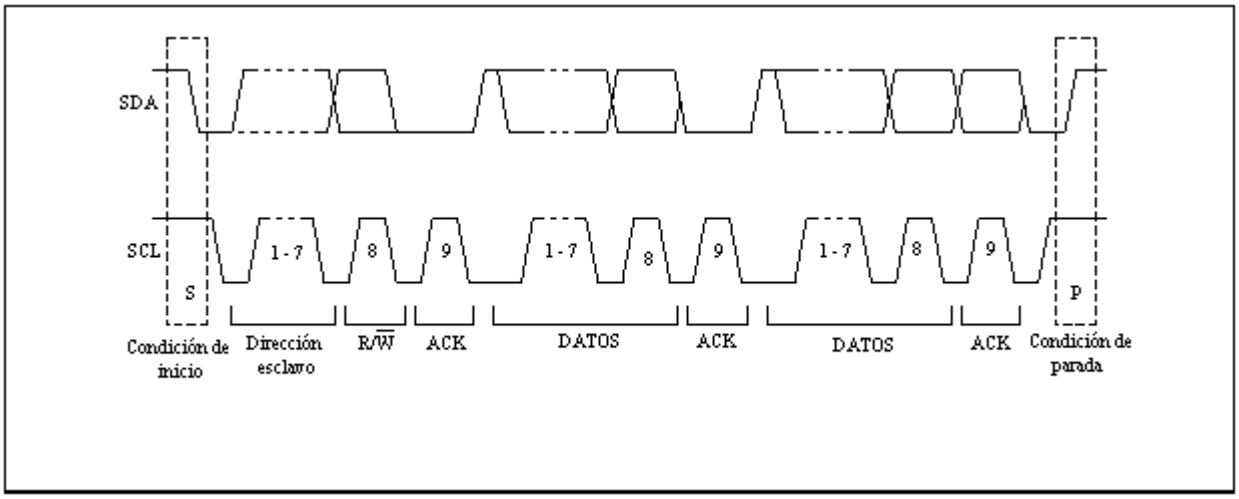

Figura 7 Transferencia Completa de Datos

Debido a lo anterior se tienen diferentes formatos de transferencia para datos como son:

- El maestro transmisor envía datos al esclavo receptor.

El sentido de la transmisión no varía y se caracteriza por que el octavo bit tiene un valor de "Cero" lo cual indica una condición de escritura hacia el dispositivo receptor.

- El maestro lee datos del esclavo transmisor inmediatamente después del primer octeto.

Una vez enviados los siete bits de dirección el octavo bit es un “Uno” valor que indica una lectura desde el dispositivo, razón por la cual los dos dispositivos involucrados en la transmisión deben conmutar hacia los estados de transmisor esclavo y receptor maestro no obstante la condición de parada siempre es generada por el maestro.

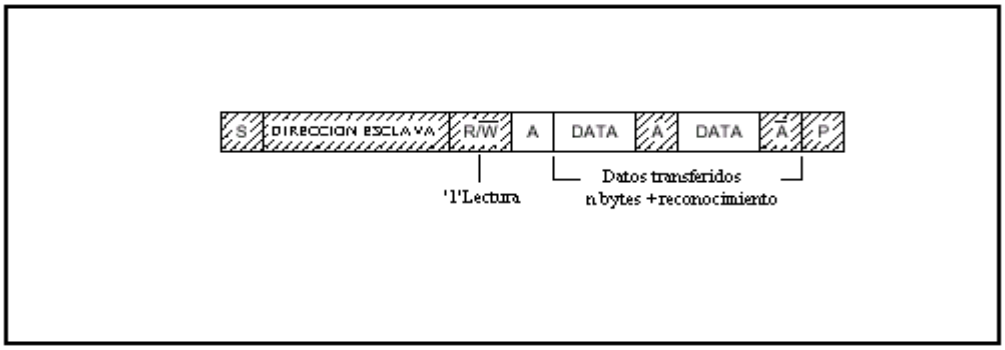

Figura 8 Maestro Lee Datos del Esclavo Transmisor Después del Primer Octeto

- Formato combinado

Combina los dos formatos anteriores, donde el maestro se encarga de iniciar una nueva condición de arranque en el caso que este quiera redireccionar un nuevo dispositivo, este nuevo redireccionamiento puede variar en el sentido de transmisión de datos dependiendo de sí el valor del octavo bit es un “Uno” o es un “Cero”.

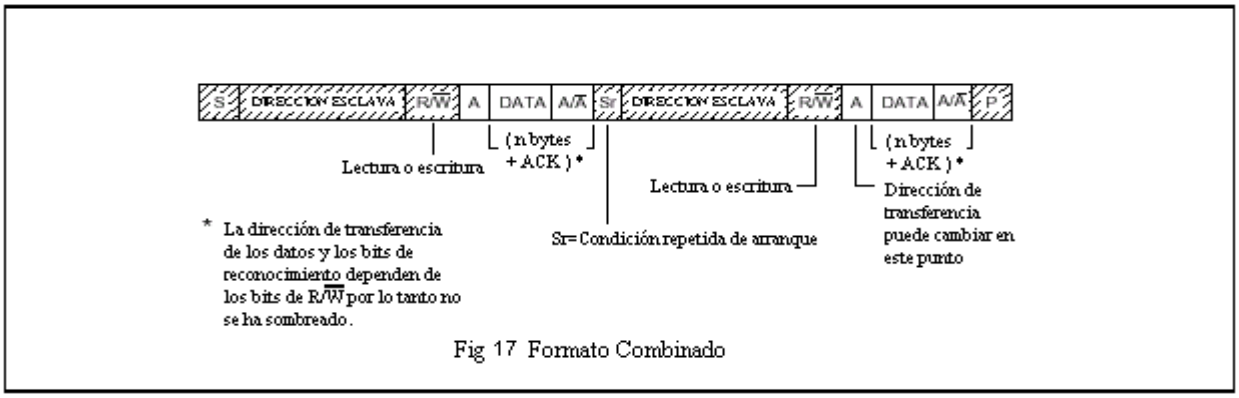

Figura 9 Formato Combinado

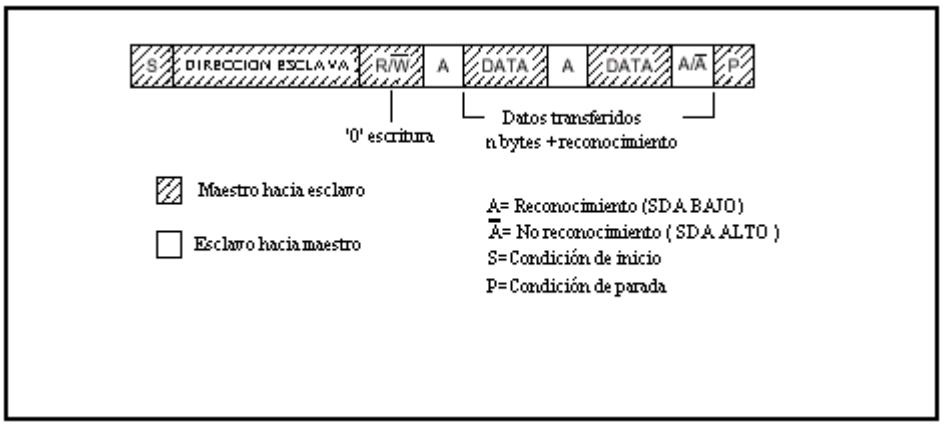

Figura 10 Maestro Direcciona al Esclavo Receptor con 7 Bits de Dirección

Es importante aclarar el significado del primer octeto enviado, pues estos corresponden a la dirección del esclavo que se quiere redireccionar, una vez un esclavo ha identificado que ha sido direccionado por un maestro mediante la comparación interna de los primeros siete bits que siguen a la condición de inicio con su dirección, dicho esclavo entenderá que ha sido direccionado y deberá conmutar a un estado de receptor esclavo o transmisor esclavo dependiendo del valor que tenga el bit LSB del octeto de dirección.

Una dirección esclava puede estar compuesta de una parte fija y una parte programable, la parte programable se encarga de determinar cuantos dispositivos de la misma clase pueden ser conectados al bus. El número de pines disponibles determina el número de bits programables de dirección, dado el caso que puedan ser conectados 8 dispositivos idénticos al bus determina que el número de pines disponibles para la parte programable deben ser tres. La siguiente tabla ilustra un grupo especial de direcciones de las cuales se tienen asignadas las siguientes:

DIRECCION ESCLAVA	BIT DE LECTURA/ESCRITURA(R/W)	DESCRIPCION
0000 000	0	Dirección de llamada general
0000 000	1	Octeto de inicio
0000 001	X	Dirección CBUS
0000 010	X	reservada a formatos diferentes de bus
0000 011	X	Reservada para propósitos futuros
0000 1XX	X	Código maestro de modo Hs
1111 1XX	X	Reservada para propósitos futuros
1111 0XX	X	Direccionamiento esclavo de 10 bits

Una situación importante se presenta cuando aparece una dirección de llamada general, la cual se encarga de direccionar a todos los dispositivos conectados al bus. Un dispositivo que necesita de los datos contenidos en la dirección de llamada general después del reconocimiento se comporta como esclavo receptor, mientras que otro dispositivo que no necesite de ella solamente ignorará dicha llamada sin generar el bit de reconocimiento. El segundo octeto de transmisión es el que determina el objetivo de la dirección de llamada general, en este caso se debe tener en cuenta el valor del octavo bit de la siguiente forma:

Cuando el octavo bit es un cero, el segundo octeto tiene las siguientes definiciones:

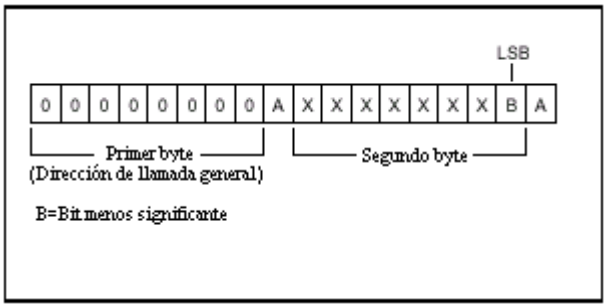

Figura 11 Formato de Llamada General

- 00000110 (06H), octeto que representa la inicialización y escritura de la parte programable de la dirección del esclavo mediante Hardware. Los dispositivos que están diseñados para responder a la dirección de llamada general (00000000) se tornarán a cero y tomarán la parte programable de su dirección. Niveles bajos de las líneas SDA o SCL bloquearían el bus por lo que se han tomado medidas preventivas en el momento de aplicar tensiones de alimentación.
- 00000100 (04H), escribe la parte programable de la dirección esclava por Hardware.
- 00000000 (00H), no está especificada esta secuencia para ser usada como segundo byte.

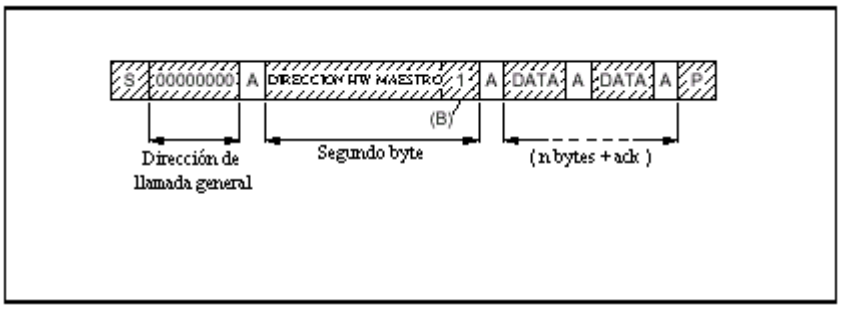

Figura 12 Transferencia de Datos Desde un Hardware Transmisor Maestro

Cuando el octavo bit es un uno, la secuencia de los dos octetos representan una llamada general de Hardware, lo cual indica que un dispositivo tal como un explorador de

teclado el cual no puede direccionar un dispositivo, envía una solicitud de llamada general por hardware en la cual se envía la dirección del esclavo deseado y su propia dirección con la cual se identifica al sistema. Esta dirección de hardware maestro es reconocida por un circuito integrado inteligente tal como un microordenador, el cual se encarga de dirigir la información procedente del hardware maestro.

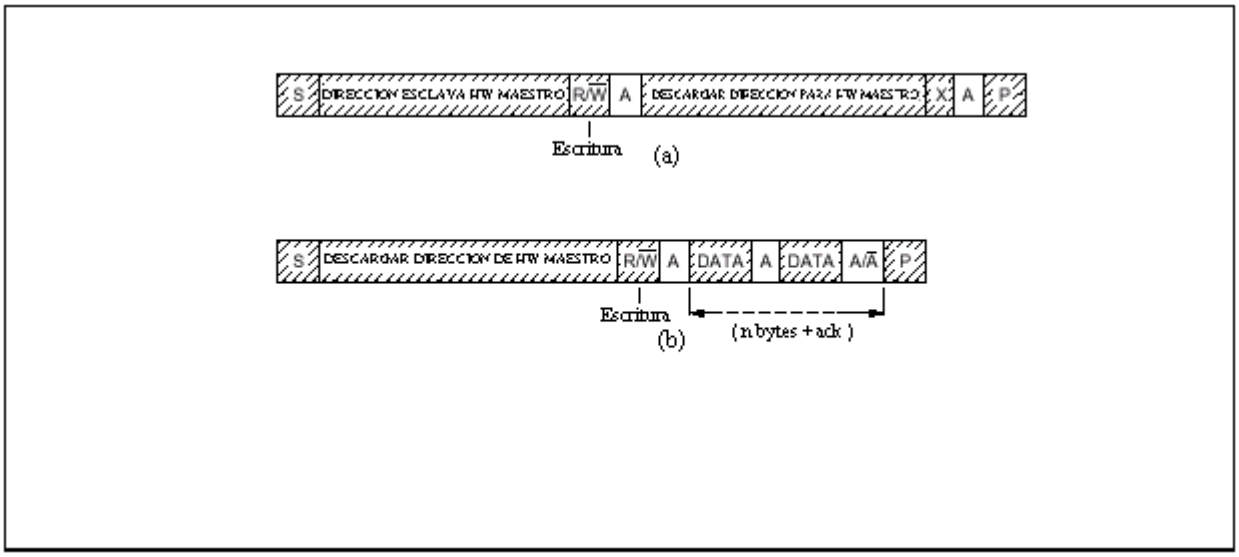

Figura 13 Transferencia de Datos por un Transmisor Hardware Capaz de Descargar Datos Directamente a Dispositivos Esclavos

Los microcontroladores que pueden ser conectados al bus pueden tener o no una interface compatible con el bus I^2C. Los que tienen la interface compatible con el bus pueden ser programados para que reciban solicitudes de interrupción por parte del bus, cuando el dispositivo no tiene la interface como tal estos deben monitorear el bus vía software, lo cual hace que los tiempos de monitoreo sean mucho mayores inclusive que el tiempo que puede gastarse llevando los datos hacia la función deseada. En este caso la transferencia de datos tiene un procedimiento que es mucho más largo que el normal y es mostrado en la Figura 14. Como se observa en la gráfica el procedimiento de arranque

consiste de una condición de arranque(S), un byte de arranque (00000001), el pulso de reloj de reconocimiento(ACK) y finalmente una condición repetida de arranque(Sr). Una vez un maestro ha emitido la condición S de arranque debido a que desea tener acceso al bus, es transmitido el byte de arranque 00000001. Entre tanto otro microcontrolador puede encuestar la línea SDA a una rata de muestreo baja hasta que uno de los siete ceros en el byte de arranque es detectado, en este momento el microcontrolador conmutará hacia una rata de muestreo alta en busca de la condición repetida de arranque Sr condición que es usada posteriormente para sincronización.

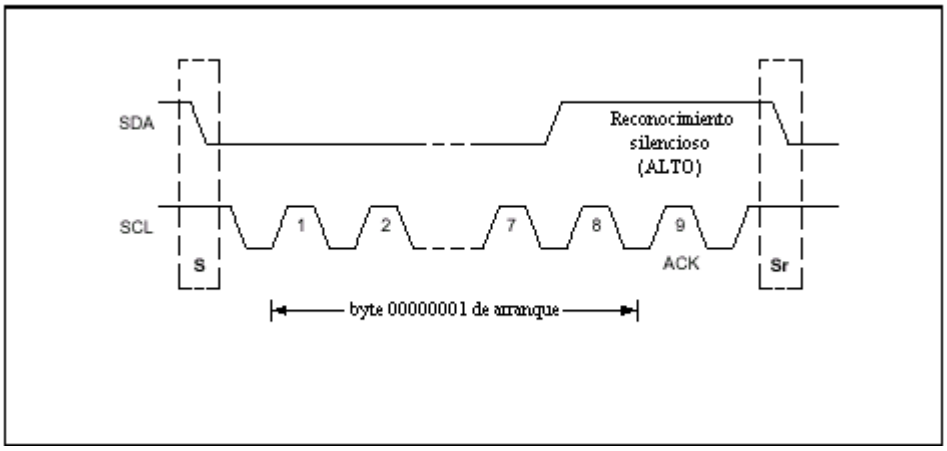

Figura 14 Procedimiento Byte de Arranque

12 ANEXO 2: MEMORIAS FLASH

Una memoria flash es capaz de retener información digital bajo ciertas condiciones. La información almacenada puede ser código operacional o archivos de datos o la combinación de los dos. La memoria ideal optimiza densidad, preserva material crítico en una condición no volátil, es fácil de programar y reprogramar, lectura rápida. Es una memoria no volátil que utiliza tecnología NOR, la cual permite al usuario programar y borrar información eléctricamente. Las memorias Flash de Intel® usan celdas de memoria similares a una EPROM, pero con una capa muy delgada de óxido entre la compuerta flotante y la fuente. La programación Flash ocurre cuando los electrones son colocados sobre la compuerta flotante. La carga es almacenada sobre la compuerta flotante, con la capa de óxido permitiendo a la celda ser borrada eléctricamente a través de la fuente.

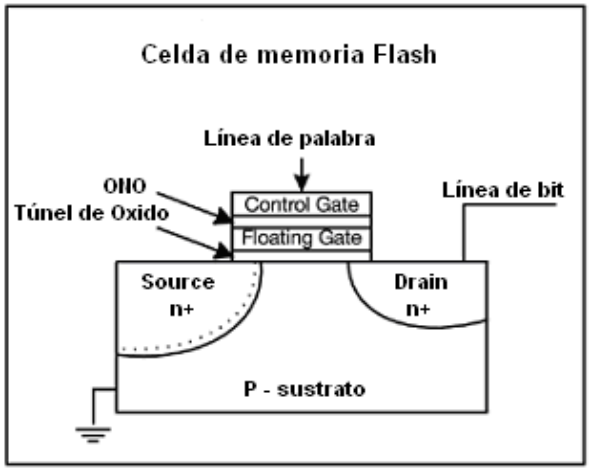

Figura 15 Celda de Memoria Flash

12.1 TECNOLOGIAS

Intel tiene dos tecnologías flash muy fiables basadas sobre el mismo núcleo de la celda de memoria (Figura 16). La primera tecnología es la original memoria flash de un solo bit por celda, que permite almacenar un solo bit de información en cada celda (1 = borrado y 0 = programado). La segunda y más reciente tecnología es llamada estructura de celda multinivel, usada en memorias Intel StrataFlash®. Esta tecnología permite almacenar dos bits de información en un solo transistor. La precisión es la clave para almacenar dos bits por celda. Programar una celda (remplazar carga) y leer debe ser controlado precisamente de acuerdo a tener cuatro estados dentro de un solo transistor. Memorias flash con celdas Multinivel es una arquitectura fiable basada sobre tecnología NOR, recomendada idealmente para aplicaciones de alta densidad.

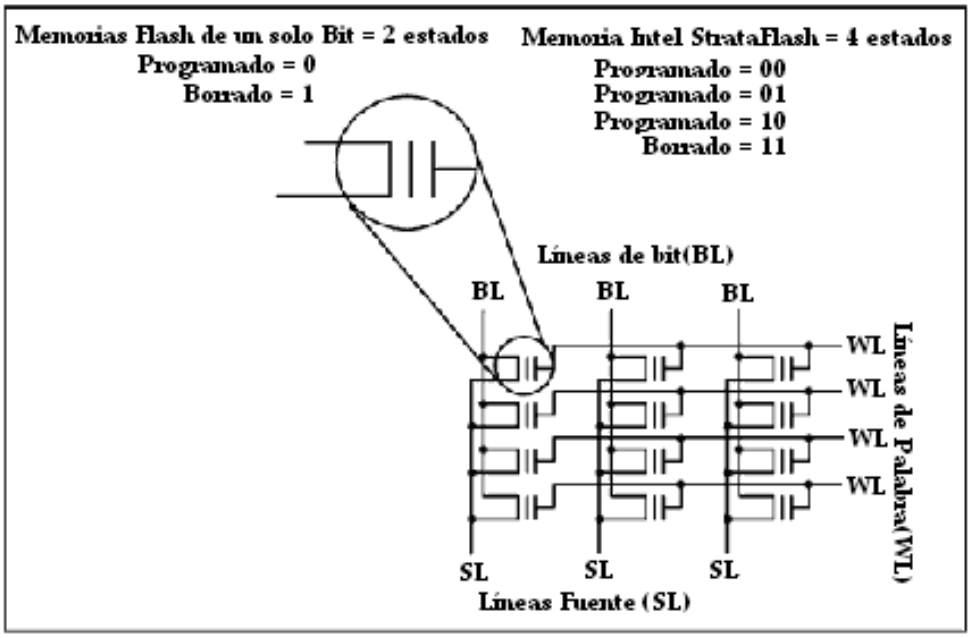

Figura 16 Memorias de un Solo Bit Contra Memorias Intel StrataFlash®

- **ARQUITECTURA DE MEMORIA**

Las memorias Flash de Intel están disponibles en arquitecturas de bloque simétrico y asimétrico (ver figura 17). La flexibilidad de arquitectura de bloques habilita el sistema de integración de código y datos dentro de un solo dispositivo flash. La familia de memorias Flash de bloques Intel®Boot tiene arreglos de capas (Bloques Asimétricos) para habilitar pequeños parámetros o almacenar código de buteo como las (EEPROM), acopladas con eficientes bloques grandes para código y almacenamiento de archivos de datos. Las familias de memorias Intel® FlashFile™ y StrataFlash son arreglos de memorias (Bloques simétricos), los cuales habilitan mejor manejo de código y archivos de datos.

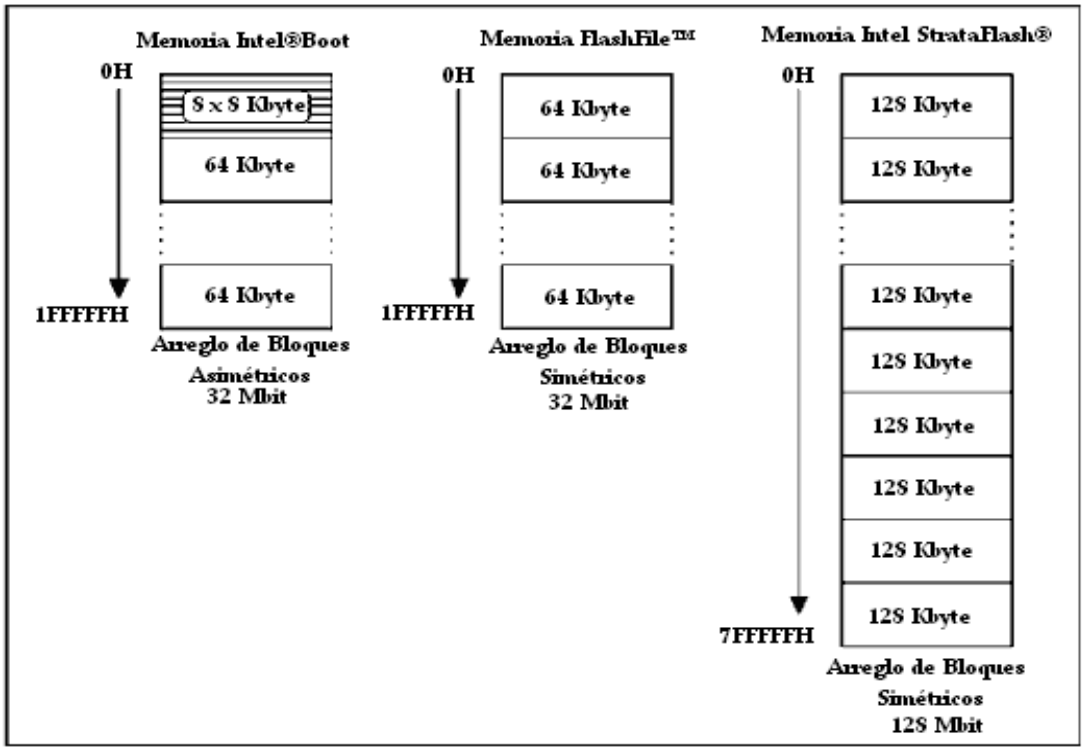

Figura 17 Arquitectura de Memoria Flash Intel

12.2 MODO DE LECTURA OPERACIONAL

Los modos de operación para componentes flash son lectura, programado y borrado. Intel provee tres tipos de operaciones de lectura: Modo palabra Asincrónico, Modo de página Asincrónico y modo burst sincrónico. El modo más común es el ***modo palabra asincrónico*** (Figura 18), donde el tiempo de acceso para los datos arranca sobre una transición de dirección, una OE# (salida habilitada) o una CE# (chip habilitado). Este tipo de acceso provee un desempeño aceptable para muchas aplicaciones y requiere solamente un diseño de sistema simple no sincronizado. Las otras dos formas de lectura tienen un alto desempeño y ellos requieren una CPU o ASIC para habilitar el modo sobre estos dispositivos.

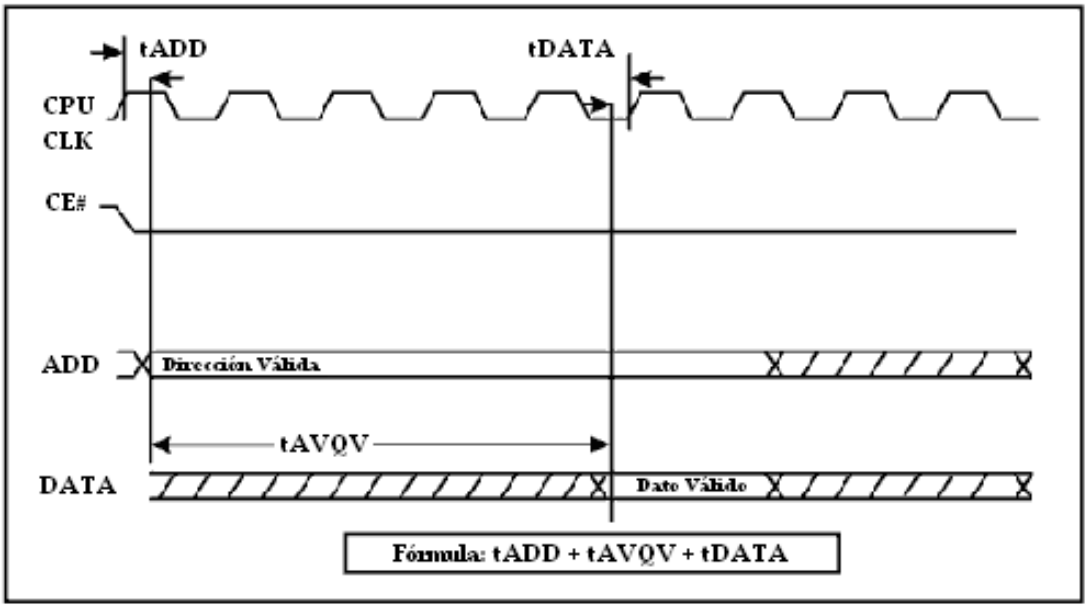

Figura 18 Modo de Lectura de Palabra Asincrónico

El ***modo de página Asincrónico*** (Figura 19) provee una alta rata de transferencia de datos para sistemas de memorias no sincronizadas. En este modo, los datos son leídos internamente y almacenados en un buffer de página de alta velocidad, requiere un

tiempo de acceso inicial similar al modo asincrónico, pero luego los datos permanecientes en la página pueden ser accesado tan rápido como 20 ns cada vez que las líneas de direcciones son pegadas dentro de la página.

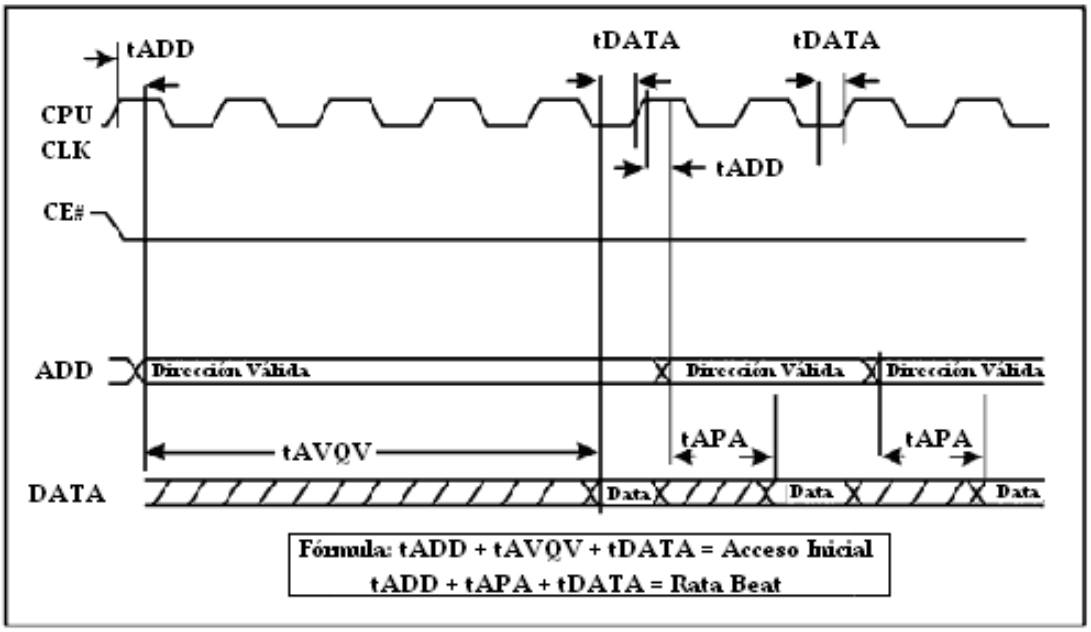

Figura 19 Modo de Lectura de Página Asincrónico

Finalmente, el modo de lectura de más alto desempeño es el ***burst sincrónico*** (Figura 20). Este modo soporta hasta 66 MHz de operación sin estado de espera. Esta interface es alcanzada a través de tres pines de control adicional: CLK# (Entrada de reloj), WAIT# (CPU espera de salida), y ADV# (dirección válida de entrada). En este modo, el dispositivo toma la dirección inicial, luego envía una secuencia de datos con respecto a la entrada de reloj y lee parámetros de configuración. Un registro de configuración de lectura programable (RCR) es suministrado para proveer conexiones de hardware.

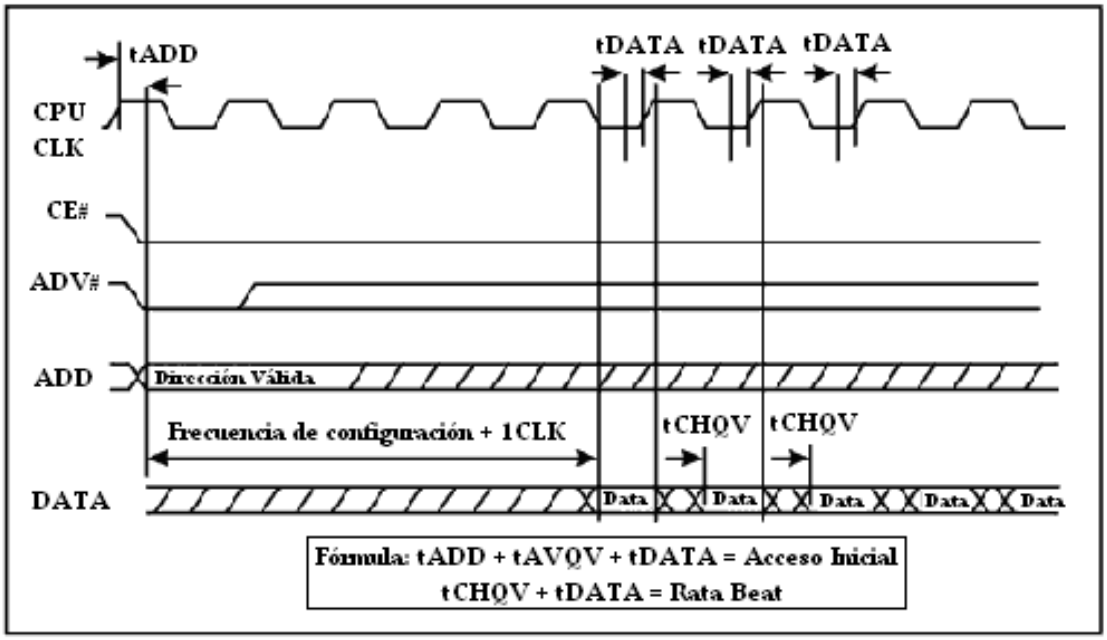

Figura 20 Modo lectura Burst Sincrónico

12.3 PROGRAMACION Y BORRADO

La programación sobre memorias Flash Intel es realizado comúnmente en un byte o modo palabra; sin embargo varios dispositivos proveen buffers de escritura de 32-bytes. El borrado de los dispositivos flash es hecho a través de comandos de borrado de bloque y el tiempo de realización es dependiente del tamaño del bloque y de la tecnología utilizada. Funciones tales como suspensión de programa, resumen de programa, suspensión borrada y resumen borrado, permiten al dispositivo pausar y leer datos y luego resumir la operación previa. La arquitectura multipartición de Intel permite al procesador del sistema leer desde una partición mientras completa una escritura o un borrado en la otra partición. Por ejemplo, este permite ejecutar código y datos de programación desde el mismo dispositivo flash al mismo tiempo.

13 ANEXO 3: CREACION DEL COMPONENTE SPC

En la Figura 21 se muestra en negrilla el componente que va a ser construido denominado *ComponenteSPC*, el cual se distingue de otros componentes debido a que se le ha especificado a la herramienta Rational Rose RealTime que establezca este componente como activo dando clic derecho en el componente y escogiendo la opción *Set As Active.*

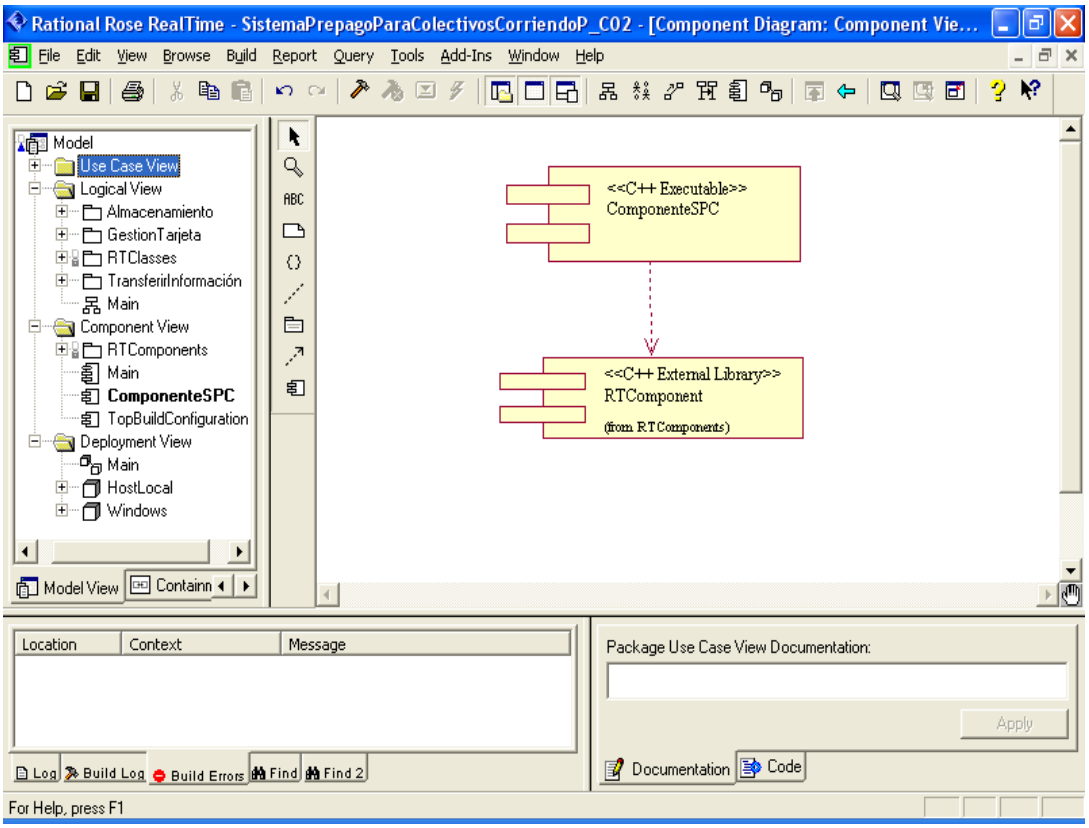

Figura 21 Diagrama De Vista De Componentes

En la Figura 22 se muestra la especificación de componentes para el *ComponenteSPC*, en la cual se establecen los parámetros bajo los cuales el componente será construido y

que se hace seleccionando una a una las pestañas que aparecen en el gráfico de acuerdo a las necesidades de la construcción del modelo.

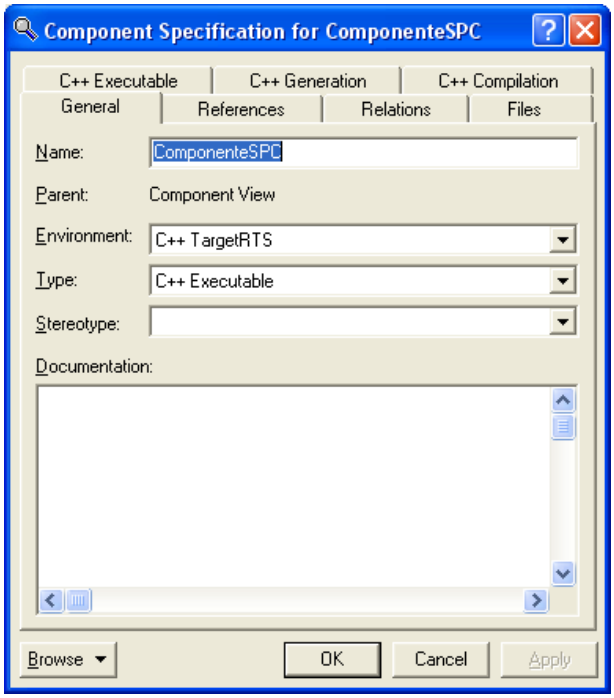

Figura 22 Especificación De Componentes Para ComponenteSPC

En la Figura 23 se muestra las cápsulas y protocolos que están asociadas en la construcción del componente ComponenteSPC.

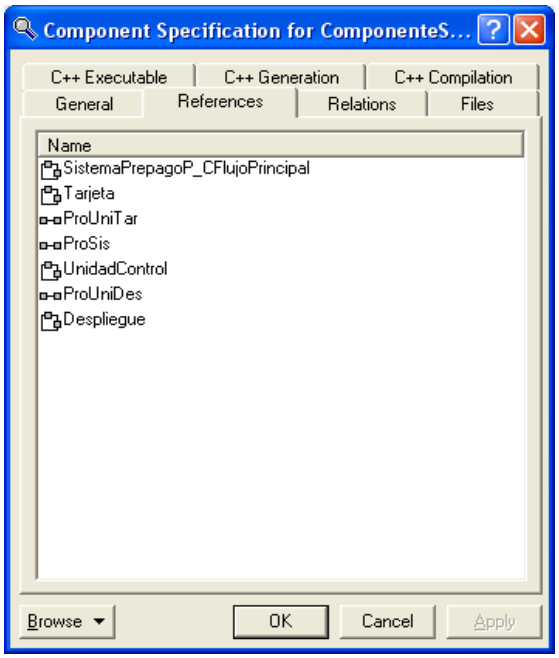

Figura 23 Especificación De Referencias Para ComponenteSPC

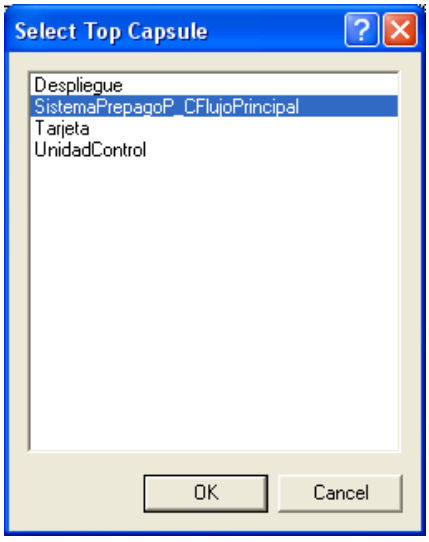

Figura 24 Selección Capsula Alto Nivel

En la Figura 24 se selecciona la cápsula de alto-nivel, la cual encapsulará todos los componentes que están involucrados en la construcción del modelo.

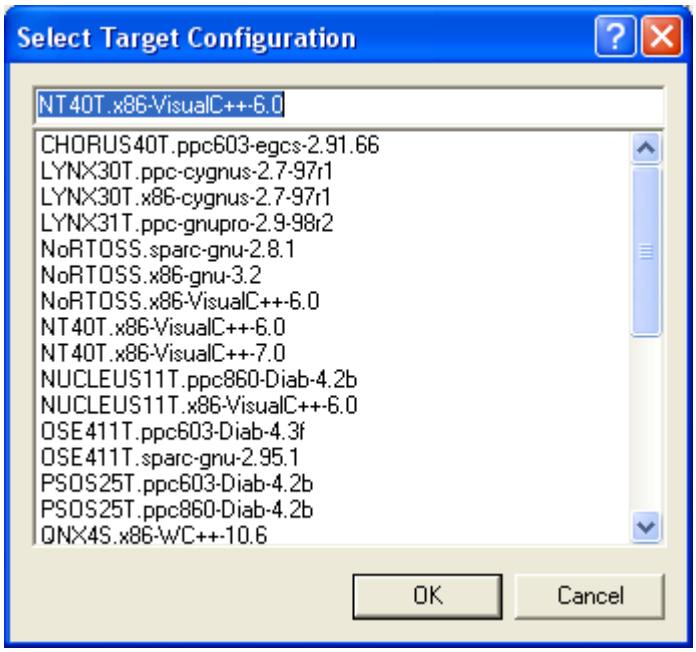

Figura 25 Selección De La Configuración Local

La Figura 25 muestra la selección de una línea, la cual especifica el *Sistema Operativo*, la *Arquitectura del Procesador* y el *Compilador* donde se va a construir el componente.

Una vez realizado los anteriores pasos se procede a la construcción del componente mediante la compilación del componente utilizando la opción construir componente suministrada por la herramienta y que se ilustra en la gráfica 26.

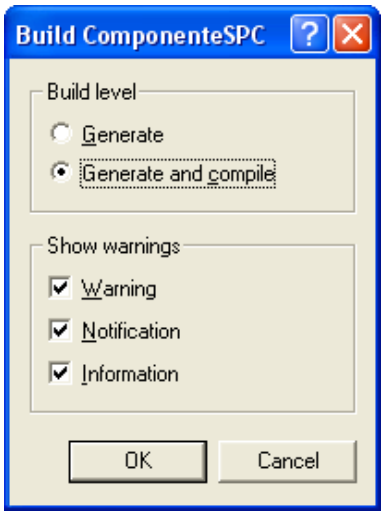

Figura 26 Compilación Del Componente ComponenteSPC

En la Figura 27 se observa en la pestaña *Build Log* el resultado de la construcción del componente *ComponenteSPC*, el cual nos muestra *Build Successful* indicando que la construcción del componente se realizó con éxito y esto da paso para que se pueda correr una instancia de dicho componente que para este caso se denomina *ComponenteSPCInstance*. En la pestaña *Model View* de la gráfica 27 se observa la instancia creada y que se encuentra lista para su ejecución.

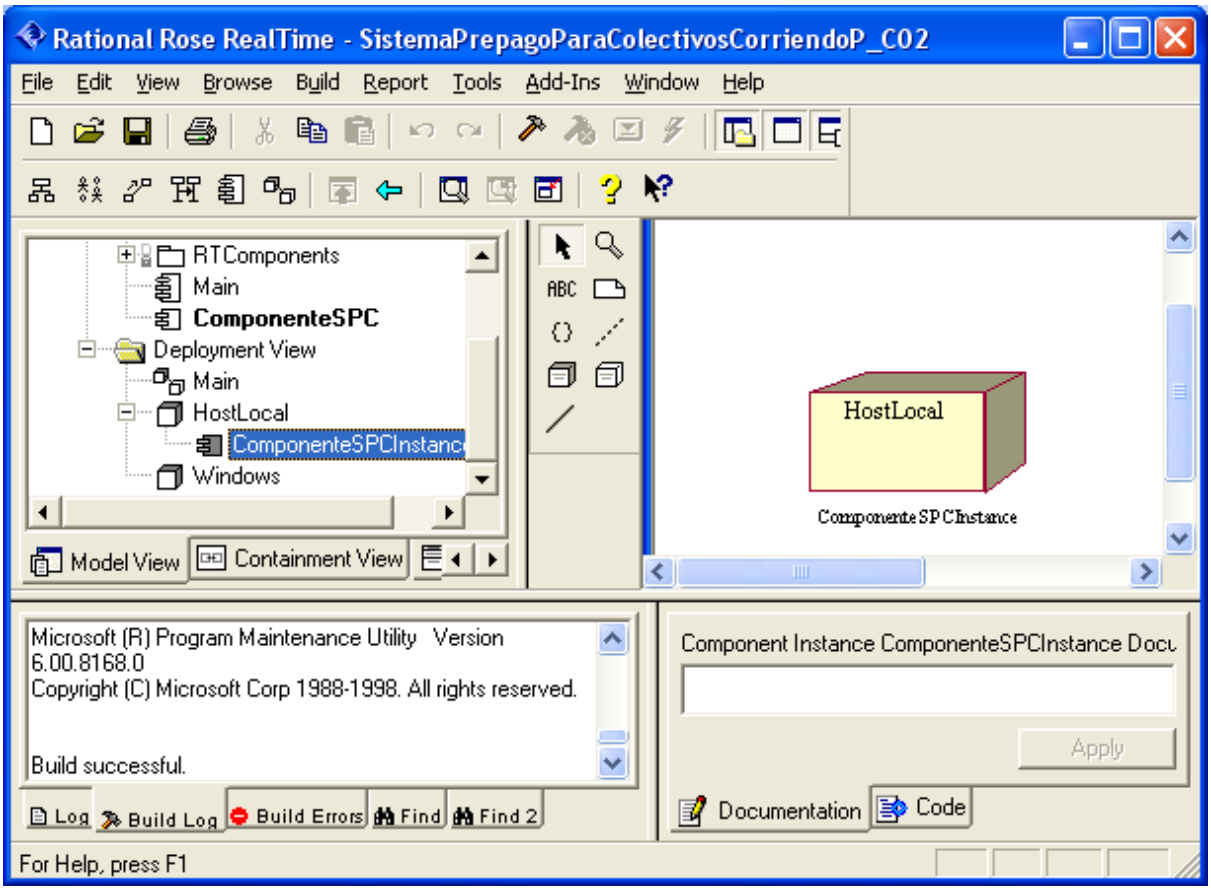

Figura 27 Resultado De La Compilación Del Componente ComponenteSPC

14 ANEXO 4: COMPILACION COMPONENTE SPC

OutputPath: E:/FelixPardo04/ComponenteSPC
Running: rtperl -S vssetup.pl nmake RTclean

Microsoft (R) Program Maintenance Utility Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

rtperl "C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/codegen/rm.pl" -f -R src build Makefile Makefile.dep rtbuild.dat
Running: rtperl -S vssetup.pl nmake -f "C:\Archivos de programa\Rational\Rose RealTime/codegen/bootstrap/MS_nmake.mk" RTS_HOME="C:\Archivos de programa\Rational\Rose RealTime/C++/TargetRTS" MODEL="E:/FelixPardo04/SistemaPrepagoParaColectivosCorriendoP_C02.rtmdl" COMPONENT="Component View::ComponenteSPC" RTmakefiles

Microsoft (R) Program Maintenance Utility Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

rtcppgen -model "E:/FelixPardo04/SistemaPrepagoParaColectivosCorriendoP_C02.rtmdl" -component "Component View::ComponenteSPC" -makegen
Warning: This component implicitly depends upon 'Component View::RTComponents::RTComponent'. The model should be updated to include an explicit dependency.
!> Generating Makefiles for ComponenteSPC
Running: rtperl -S vssetup.pl nmake RTcompile

Microsoft (R) Program Maintenance Utility Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

rtperl "C:\Archivos de programa\Rational\Rose RealTime/codegen/chdir_run.pl" src rtperl -S vssetup.pl nmake -s RTgenerate

Microsoft (R) Program Maintenance Utility Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

!> Generating Component Unit
!> Generating Despliegue
!> Generating ProSis
!> Generating ProUniDes
!> Generating ProUniTar

```
!> Generating SistemaPrepagoP_CFlujoPrincipal
!> Generating Tarjeta
!> Generating UnidadControl
        rtperl "C:\Archivos de programa\Rational\Rose RealTime/codegen/chdir_run.pl"
build rtperl -S rtsetup.pl nmake -k RTcompile

Microsoft (R) Program Maintenance Utility   Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
component "Component View::ComponenteSPC" -src Despliegue ../src/Despliegue.cpp
--  cl /c  /FoDespliegue.OBJ  /nologo /G5 /GX /GF /MD /TP    /I"C:/Archivos de
programa/Rational/Rose RealTime/C++/TargetRTS/libset/x86-VisualC++-6.0"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/target/NT40T"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/include" /Zi
/I../src  ../src/Despliegue.cpp
!> Compiling Despliegue
Despliegue.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
component "Component View::ComponenteSPC" -src ProSis ../src/ProSis.cpp --  cl /c
/FoProSis.OBJ  /nologo /G5 /GX /GF /MD /TP    /I"C:/Archivos de
programa/Rational/Rose RealTime/C++/TargetRTS/libset/x86-VisualC++-6.0"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/target/NT40T"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/include" /Zi
/I../src  ../src/ProSis.cpp
!> Compiling ProSis
ProSis.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
component "Component View::ComponenteSPC" -src ProUniDes ../src/ProUniDes.cpp
--  cl /c  /FoProUniDes.OBJ  /nologo /G5 /GX /GF /MD /TP    /I"C:/Archivos de
programa/Rational/Rose RealTime/C++/TargetRTS/libset/x86-VisualC++-6.0"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/target/NT40T"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/include" /Zi
/I../src  ../src/ProUniDes.cpp
!> Compiling ProUniDes
ProUniDes.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
component "Component View::ComponenteSPC" -src ProUniTar ../src/ProUniTar.cpp -
-  cl /c  /FoProUniTar.OBJ  /nologo /G5 /GX /GF /MD /TP    /I"C:/Archivos de
programa/Rational/Rose RealTime/C++/TargetRTS/libset/x86-VisualC++-6.0"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/target/NT40T"
```

```
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/include" /Zi
/I../src  ../src/ProUniTar.cpp
!> Compiling ProUniTar
ProUniTar.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
component "Component View::ComponenteSPC" -src
SistemaPrepagoP_CFlujoPrincipal ../src/SistemaPrepagoP_CFlujoPrincipal.cpp --  cl /c
/FoSistemaPrepagoP_CFlujoPrincipal.OBJ  /nologo /G5 /GX /GF /MD /TP
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/libset/x86-
VisualC++-6.0"  /I"C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/target/NT40T"  /I"C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/include" /Zi /I../src
../src/SistemaPrepagoP_CFlujoPrincipal.cpp
!> Compiling SistemaPrepagoP_CFlujoPrincipal
SistemaPrepagoP_CFlujoPrincipal.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
component "Component View::ComponenteSPC" -src Tarjeta ../src/Tarjeta.cpp --  cl /c
/FoTarjeta.OBJ  /nologo /G5 /GX /GF /MD /TP    /I"C:/Archivos de
programa/Rational/Rose RealTime/C++/TargetRTS/libset/x86-VisualC++-6.0"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/target/NT40T"
/I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/include" /Zi
/I../src  ../src/Tarjeta.cpp
!> Compiling Tarjeta
Tarjeta.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
component "Component View::ComponenteSPC" -src UnidadControl
../src/UnidadControl.cpp --  cl /c  /FoUnidadControl.OBJ  /nologo /G5 /GX /GF /MD
/TP    /I"C:/Archivos de programa/Rational/Rose RealTime/C++/TargetRTS/libset/x86-
VisualC++-6.0"  /I"C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/target/NT40T"  /I"C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/include" /Zi /I../src  ../src/UnidadControl.cpp
!> Compiling UnidadControl
UnidadControl.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtcomp.pl" -vendor VisualC++ -spacify dq -I ../src  -
dep ComponenteSPC_SYSDEPS RTSystem/ComponenteSPC.dep  -component
"Component View::ComponenteSPC" -src ComponenteSPC
../src/RTSystem/ComponenteSPC.cpp --  cl /c  /FoRTSystem/ComponenteSPC.OBJ
/nologo /G5 /GX /GF /MD /TP    /I"C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/libset/x86-VisualC++-6.0"  /I"C:/Archivos de
programa/Rational/Rose RealTime/C++/TargetRTS/target/NT40T"  /I"C:/Archivos de
```

```
programa/Rational/Rose RealTime/C++/TargetRTS/include" /Zi /I../src
../src/RTSystem/ComponenteSPC.cpp
!> Compiling ComponenteSPC
ComponenteSPC.cpp
        rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/codegen/rtlink.pl" -vendor VisualC++ -exe
"SistemaPrepagoP_CFlujoPrincipal.EXE" -component "Component
View::ComponenteSPC" --  rtperl "C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/libset/x86-VisualC++-6.0/ld.pl" link
/OUT:SistemaPrepagoP_CFlujoPrincipal.EXE  /nologo  "C:/Archivos de
programa/Rational/Rose RealTime/C++/TargetRTS/lib/NT40T.x86-VisualC++-
6.0/main.OBJ"   %ALL_OBJS.olist     -L"C:/Archivos de programa/Rational/Rose
RealTime/C++/TargetRTS/lib/NT40T.x86-VisualC++-6.0"  -llibObjecTime  -
llibObjecTimeTypes    -lwsock32
!> Linking SistemaPrepagoP_CFlujoPrincipal.EXE

Build successful.
```

15 ANEXO 5: CODIGO CAPA SISTEMA PREPAGOP_C

```
'IMPLEMENTACION CAPA SISTEMA PREPAGOP_C

DEVICE = 16f877A
' Variable que define la serie del Microcontrolador estandarizada por el fabricante ( PIC ).

DECLARE LCD_TYPE 0
' Tipo de LCD Texto (0) - Gráfico (1)
'Declaración de variables para el control del Despliegue LCD. Se toma un LCD de 2 X 16 Caracteres No Grafico o Texto.

DECLARE LCD_DTPIN PORTD.4
' A partir del PIN 4 o PORD.4 (Puerto D4,D5,D6,D7) Se transfieren los datos en dos grupos de 4 bits.

DECLARE LCD_RSPIN PORTD.0
'Definición del PIN para la señal de sincronismo en la transferencia de datos entre el Microcontrolador (Maestro) y el LCD (Esclavo).

DECLARE LCD_ENPIN PORTD.1
'Declaración del PIN para la señal de Habilitación (E- ENABLED) de LCD controlado por el Microcontrolador.

DECLARE LCD_INTERFACE 4
'Define el numero de datos a transferir por paquete, en esta caso 4 Bits.

DECLARE LCD_LINES 2
'Define La característica del número de líneas del LCD.

DECLARE KEYPAD_PORT PORTB
'Define el Puerto de entrada (4)y salida(4) para el control del teclado 4 X4

TRISD = 0
'Se define el Puerto D como salida

TRISB=%11110000
'Se define el puerto B como entrada/salida

TRISA=%100000
'Se define el puerto A como entrada/salida
```

```
TRISA 0.5 = 1
'Se declara del puerto A el bit  # 5 para detectar la tarjeta prepago.

TRISC 0# = 0
' Trabaja con el PITO

'En este bloque se definen variables de programación.

Dim TECLA As Byte
'Para la lectura del Teclado.

Dim TEM_TECLA As Byte
'Variable donde se almacena temporalmente el resultado de la función INKEY

Dim TEM_DEBITO As Byte
Dim KEY As Byte
Dim VALOR As WORD
Dim VALOR_PASAJE As WORD
Dim i As Byte

Dim VALOR_MEM As Byte
'Variable donde se almacena la cantidad de pasajes almacenados en la memoria del
sistema SPC

Dim DIRECCION As WORD
Dim TEM As Byte
Dim BANDERA As Byte


DECLARE SDA_PIN PORTD.2
DECLARE SCL_PIN PORTD.3
'Definición de pines para las señales SDA y SCL  del protocolo I2C utilizados en la
trasmisión de
'información entre el microcontrolador y la memoria.
Dim T As Byte

SYMBOL C $A0
'Palabra de Control para seleccionar la memoria de la tarjeta prepago

ALL_DIGITAL = True
'Establece pines análogos a digitales sobre cualquier dispositivo compatible
ADCON1 = 7
'Comando especial qu convierte todos los pines a modo digital
```

```
i = 0
TEM_DEBITO = 0
VALOR_PASAJE = 650
BANDERA = 0

cls

Print at; 1, 1, "SISTEMA SPC"
'Mensaje de Inicio del sistema

Print at; 2, 1, "COLECTIVOS"
'Mensaje de Inicio del sistema

delayms 2000

'Inicio de Ciclo Infinito
CICLO:

        Gosub detectar_targeta
        'Saltar a la etiqueta detectar_tarjeta, con retorno al sitio de salto.

        'Una vez se haya detectado la tarjeta se solicita por mensaje en el LCD los pasajes
        a pagar.

         cls
          Print at; 1, 1, "Introduzca el "
           Print at; 2, 1, "# de pasajes"
          delayms 50

          'Se prosigue a leer teclado, para almacenar los pasajes.

          GoSub leer_teclado
          'Salta a la rutina leer_teclado, con retorno de lectura de teclado.

           'Borra las Variables.
           TEM_TECLA = 0
            i = 0
            KEY = 0
            BANDERA = 0

            'Solicita que termine la operación con el retiro de la tarjeta prepago.
retirar:

            Print at; 1, 1, "* Retirar..."
```

```
        Print at; 2, 1, "** TARJETA **"
        if PORTA.5 =0 then
           GOTO retirar
        end if

       'Hasta que  no retire la tarjeta, no se inicia el nuevo procedimiento para debito
         por otro usuario.

GOTO CICLO
 'Fin del ciclo infinito, retorna al INICIO

detectar_targeta:

cls
'Borra el LCD

volver:
delayms 10

Print at; 1, 1, "Introduzca su"
Print at; 2, 1, "Tarjeta"
'Solicita mediante mensaje, introducir tarjeta prepago para iniciar operación de
debitación.

'Almacena el estado del pin que verifica si la tarjeta esta introducida
if PORTA.5 = 0 then
GOTO Salir
'Si hay una tarjeta introducida , salta de la subrutina y regresa al ciclo principal.

else
GOTO volver
'Se mantiene en el ciclo para que se le introduzca la tarjeta.
end if

Salir:
'Retorna al inicio de la llamada en el ciclo principal.
Return


leer_teclado:

seguir_lectura:

    TECLA = INKEY
     'Función que lee el teclado en el puerto B.
```

```
KEY = LOOKUP TECLA,[1,2,3,"A",4,5,6,"B",7,8,9,"C","*",0,"#","D"]
'Organizar la secuencia de Teclas del teclado. 16.
 if TECLA = 16 then
 ' No se hace nada, 16 es el valor por defecto cuando no se ha accionado ninguna
 tecla.
 else
 FREQOUT PORTC.0 , 60 , 2600
 'Genera un tono a una frecuencia de 2600Hz por 60milisengundos sobre
 'el bit 0 del puerto C
 else



    Select Case KEY
    Case 65
    'Si se presiona el caracter A, permite consultar el valor de pasajes de la memoria
     de la tarjeta prepago.
     VALOR_MEM = 0
      DIRECCION = 1
      BUSIN C, DIRECCION, [VALOR_MEM]
      'Lee el byte desde la EEPROM
      delayms 40
      cls
       Print at; 1, 1, "Saldo Pasajes"
       Print at; 2, 1, " * ", DEC; VALOR_MEM, " * "
       delayms 2000

       Case 66
       'Si se presiona la tecla B se recarga la tarjeta prepago con un valor de 100
       DIRECCION = 1
       BUSOUT C, DIRECCION, [100]
       delayms 50

       Case 67, 68

       Case 42
       'Si se presiona la tecla * se resetean las variables y el LCD
       TEM_TECLA = 0
        i = 0
        BANDERA = 1

       Case 35 'Valor de #
        'Si se presiona la tecla #  se termina
```

```
'con la Introducción de los datos o número de pasajes e inicia el debito de la
tarjeta prepago del usuario
VALOR = TEM_TECLA * VALOR_PASAJE
cls
Print at; 1, 1, "Operacion..."
Print at; 2, 1, "$", DEC; VALOR

'Una vez calculado el número de pasajes a debitar se procede a realizar la
operación sobre las Memorias .
GOTO debitar

TEM_TECLA = 0
 i = 0
KEY = 0
BANDERA = 1

case else

if  i = 0 then
TEM_TECLA = KEY
 else
TEM_TECLA = TEM_TECLA * 10 + KEY
end if
 i = i + 1
 cls
 Print at; 1, 1, "Pasajes: ", DEC; TEM_TECLA
Print at; 2, 1, "Valor: ", DEC; TEM_TECLA * VALOR_PASAJE
delayms 100

end select

end if

if  BANDERA = 0 Then
GOTO seguir_lectura
 end if

Return

'si hay Pasaje disponibles en la memoria de tarjeta del pasajero.
debitar:
DIRECCION = 1
BUSIN C, DIRECCION, [TEM_DEBITO]        ' Read the byte de la memoria I2C
delayms 20
```

```
if PP4.0 = 1 then
               for i=0 to 35 then
                 cls
                 Print at; 1, 1, "*TARGETA INVALIDA*"
                 Print at; 2, 1, "*TARGETA INVALIDA*"
                 FREQOUT PORTC.0 , 200 , 1500 , 2500
                 delayms 30
               Next i
 BANDERA = 1
cls
GOTO retirar
end if


'Si el valor de la cantidad de pasajes excede el disponible en la tarjeta prepago, se
rechaza la operación

delayms 40

if TEM_DEBITO < TEM_TECLA then
  cls
  Print at; 1, 1, "Pasajes Insufici"
  Print at; 2, 1, "Tiene: ", DEC; TEM_DEBITO
  delayms 3000
else
   'Se procede a realiza el debito y almacenamiento de un nuevo valor producto de la
   'diferencia entre el número de pasajes disponibles y el número de pasajes a pagar.
    TEM_DEBITO = TEM_DEBITO - TEM_TECLA
     BUSOUT C, DIRECCION, [TEM_DEBITO]
    delayms 40
    cls
    Print at; 1, 1, "Saldo Pasajes"
    Print at; 2, 1, " ** ", DEC; TEM_DEBITO, "**"
    delayms 2000

    Gosub almacenar
     'ALmacena el numero de pasaje enla Memoria del Colectivo
   end if

Return



almacenar:
```

```
  VALOR_MEM = 0
  DIRECCION = 10
  'Dirección de memoria donde se almacena los pasajes Acumulados

  VALOR_MEM = EREAD DIRECCION
  'Lee información desde la memoria integrada en el Microcontrolador en la posición 10
  delayms 100

  cls
  Print at; 1, 1, "Ant:", DEC; VALOR_MEM
  delayms 4000
  VALOR_MEM = VALOR_MEM + TEM_TECLA

  DIRECCION = 10
  EWRITE 10, [VALOR_MEM]
 'Escribe información sobre la memoria del microcontrolador en la dirección de
 'memoria 10

 delayms 20
 VALOR_MEM = 0
 DIRECCION = 10
 'Dirección de memoria donde se almacena los pasajes Acumulados
 VALOR_MEM = EREAD 10
 delayms 20
  Print at; 2, 1, "Alm:", DEC; VALOR_MEM
  delayms 4000

Return
```

16 ANEXO 6: MANUAL DE USUARIO SISTEMA SPC

FUNCIÓN PRINCIPAL

- Cancelar un determinado número de pasajes especificados por un pasajero que aborda un colectivo y que previamente ha comprado una tarjeta prepago para colectivos.
- Permitir a un pasajero consultar el saldo de pasajes que tiene almacenado en su tarjeta prepago para colectivos.
- Desplegar mediante mensajes en pantalla LCD cada una de las transacciones realizadas por cualquier usuario del sistema.
- Debitar de la tarjeta prepago para colectivos el valor correspondiente al número de pasajes digitados por el usuario.
- Acumular en memoria todas las transacciones (Pasajes debitados) realizadas en un día.
- Indicar mediante mensajes en el despliegue LCD cuando un usuario ha olvidado retirar su tarjeta prepago para colectivos.

INSTALACIÓN

- Este sistema requiere de una fuente de alimentación DC de 5 Voltios y una corriente máxima de 300mA.
- El dispositivo no debe ser expuesto directamente a los rayos solares o a la lluvia pues estos factores disminuirán la vida útil del sistema.

- El sistema consta de un teclado de 4 X 4 y será la única forma de interactuar con el sistema.
- Se tiene un despliegue LCD de 2 X 16 y será el único medio a través del cual se podrá visualizar las diferentes operaciones realizadas en el sistema SPC.

INSTRUCCIONES DE USO

En la Figura 28 se presenta la vista principal del sistema donde las teclas tienen las siguientes funciones:

Figura 28 Vista Principal del Sistema

- Las teclas etiquetadas con los números digitos **0,1,2,3,4,5,6,7,8,9** permiten a los usuarios digitar el número de pasajes a debitar de la tarjeta prepago para colectivos.
- La tecla etiquetada con el carácter **A**, permite a un pasajero consultar el saldo de pasajes contenido en su tarjeta prepago para colectivos.
- La tecla etiquetada con el caracter ***** permite a un pasajero borrar el despliegue LCD en el evento que haya digitado un valor equivocado.
- La tecla etiquetada con el caracter **#** sirve para confirmarle al sistema por parte de un usuario el número de pasajes a debitar.

- Los caracteres **B,C,D** están reservados para aplicaciones futuras.

Cuando se enciende el sistema se presenta el mensaje de Bienvenida “SISTEMA SPC COLECTIVOS, tal como se muestra en la Figura 29.

Figura 29 Inicio

A continuación se describe como realizar una transacción por parte de un usuario utilizando el sistema prepago para colectivos SPC:

1. El sistema estará desplegando el mensaje “ Introduzca su Tarjeta” como se muestra en la Figura 30.

Figura 30 Insertar Tarjeta

2. El pasajero introduce la tarjeta prepago para colectivos en la posición especificada para esta.
3. El sistema indica al pasajero mediante el mensaje “Introduzca el # de pasajes”

que proceda a realizar la debitación, como se indica en la Figura 31.

Figura 31 Debitar Pasajes

4. El pasajero debe introducir desde teclado el número de pasajes a cancelar (Figura 32).
5. Si el pasajero no introduce ningún valor el sistema continuará esperando hasta que se le digite algún valor.

Figura 32 Desplegar Pasajes

6. El pasajero debe digitar la tecla # para confirmar la aceptación de los pasajes a cancelar.
7. El pasajero debe digitar la tecla * para cancelar la transacción.
8. Si la transacción ha sido cancelada el sistema regresa al ítem especificado con el numeral 3 de este manual.

9. Si el pasajero digitó la tecla # debe esperar el mensaje de confirmación de la transacción (Figura 33).

Figura 33 Saldo Pasajes

10. El mensaje de confirmación indica el número de pasajes que tiene como crédito en la tarjeta prepago mediante el mensaje “Saldo Pasajes ** XX” donde XX depende del valor almacenado en cada tarjeta (Figura 33).
11. Si el pasajero no tiene suficiente crédito en su tarjeta prepago, el sistema desplegará en pantalla el mensaje “Pasajes Insufici Tiene: XX” indicando la condición de pasajes insuficientes y siendo XX el número de pasajes que tiene almacenados actualmente (Figura 34).

Figura 34 Pasajes Insuficientes

12. El sistema regresa al ítem etiquetado con el numeral 3 de este manual.

13. Si la transacción ha sido exitosa pero el usuario olvida retirar su tarjeta, el sistema le muestra el mensaje “Retirar... TARJETA”, el cual permanece desplegado por el tiempo que la tarjeta continúe insertada (Figura 35).

Figura 35 Retirar Tarjeta

14. Una vez el pasajero retira su tarjeta el sistema regresa a solicitar la introducción de una nueva tarjeta como se especifica en el númeral 3 de este manual.

NOTA: Si la tarjeta está deteriorada el sistema enviará un mensaje de tarjeta inválida.