PROYECTO SMART

TARJETA INTERFAZ DE TRONCAL DIGITAL – TITD

DIEGO ALFONSO AGUILAR CARDONA

PABLO JAVIER GUTIERREZ DELIOT

DIRECTOR: ING. IVAN HERNANDEZ DELGADO

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES

POPAYAN

2002

PROYECTO SMART

TARJETA INTERFAZ DE TRONCAL DIGITAL – TITD

DIEGO ALFONSO AGUILAR CARDONA

PABLO JAVIER GUTIERREZ DELIOT

Monografía presentada como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

DIRECTOR: ING. IVAN HERNANDEZ DELGADO

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES

POPAYAN

2002

A mi mamá quien me enseñó todo lo realmente importante.
Gracias por llenar de luz mi vida y por permitirme disfrutar y compartir los mejores momentos. Nunca olvidaré el brillo de tus ojos. Con tu ejemplo y tus palabras me dejas un camino a seguir; trataré de hacerlo lo mejor posible.

Diego

A mis padres, Marco y Mercedes, mis hermanos Ricardo, Oriana, Juan Carlos y Ricardito, gracias por comprenderme, apoyarme y valorar mi esfuerzo. A mis amigos por sus palabras de aliento. A mi amigo y compañero por su talento y su compañía. A todos los ingenieros involucrados este proyecto por poner de su parte para culminarlo exitosamente. Gracias.

Pablo

PRESENTACION

El Departamento de Conmutación desde el año de 1977 ha estado trabajando en desarrollos en el área de telefonía, con el objetivo de brindar soluciones y alternativas tecnológicas al país en esta materia. El eje fundamental del departamento gira en torno a la investigación y esta se ha centrado en el desarrollo de hardware y software para permitir el control del encaminamiento de la información tradicionalmente denominado conmutación y que ahora podríamos definir como la inteligencia de la red.

Alrededor del año 1994 en el proyecto Concentrador Telefónico Digital CTD, se desarrolló una tarjeta de troncal de Interfaz Digital, de esta tarjeta queda el prototipo que se montó en esos días y alguna documentación acerca de los diagramas eléctricos y los diferentes componentes que se emplearon en su fabricación, desafortunadamente la parte de los códigos del hardware y el software de la misma no fueron localizados; el presente trabajo partió de la tarjeta montada y su objetivo fue colocarla nuevamente en funcionamiento, por lo tanto para esta tarjeta fundamentalmente se tuvo que especificar, documentar, diseñar, implementar y probar:

El código fuente en VHDL del circuito de lógica programable PLD.

El código fuente correspondiente al Microcontrolador 8X51.

El código fuente del driver para el manejo de la tarjeta a través de su conector ISA desde el PC donde está alojada.

Debido a esto, y bajo una adaptación al hardware de la metodología de trabajo UML para el Proceso Unificado se ha pensado en la siguiente tentativa para la elaboración de los documentos relacionados:

Captura de Requerimientos (Capítulo 1).
Análisis del Sistema (Capítulo 2).
Diseño del Sistema (Capítulo 3).
Implementación del Sistema (Capítulo 4).
Pruebas del Sistema (Capítulo 5).
Estudio de la Arquitectura Base (Anexo).
Códigos Fuente del Sistema (Anexo).
Esquemáticos (Anexo).
Manual de Usuario (Anexo).

El presente trabajo de grado se enmarca dentro del proyecto SMART, el cual cuenta con apoyo económico de la VRI y que propende por el desarrollo de una plataforma para desarrollo de servicios de red inteligente tanto para telefonía convencional como también móvil y telefonía IP.

CAPITULO 1
ANALISIS DE REQUERIMIENTOS DEL SISTEMA

1 ESPECIFICACION DE REQUERIMIENTOS

1.1 DESCRIPCION

La TITD demanda un mínimo de dispositivos y/o módulos tanto hardware como software para poder ofrecer y soportar, con alto grado de confiabilidad, las funciones básicas de conmutación y comunicación que de ella se derivan.

Se aclara que para el proyecto TITD ya se posee una placa impresa que se puede insertar en un Slot ISA de un PC, la cual consta de:

- Un dispositivo (CI MH89790 – Interfaz de troncal digital) que permite generar, interpretar y controlar tanto códigos de línea (HDB3 o AMI) como señales de sincronismo; también permite insertar y detectar bits de señalización, además de determinar errores. Tales funciones con el fin de poder establecer un enlace CEPT PCM E1.
- Un dispositivo (CI MT8980D – Conmutador digital) que permite realizar operaciones de conmutación con características de switcheo en el ámbito del espacio y del tiempo, y generar entramados con mensajes configurados desde operador, dirigidos a un conjunto de buses seriales sincrónicos de datos multiplexados en el tiempo a 2048 Kbps y distribuidos como 32 canales de 64 Kbps.
- Un dispositivo (CI MT8941 - PLL) que proporciona los relojes y señales de temporización que hacen parte del sincronismo del enlace CEPT PCM E1.

- Un dispositivo (CI MT8952 - HDLCPC) que utiliza el protocolo HDLC para entramar y dar formato a paquetes de datos de acuerdo con las recomendaciones de la ITU-T X.25 (Nivel 2).

Se ha hecho mención hasta ahora, de aquellos dispositivos que se encuentran dentro de la tarjeta como actores externos al sistema que se pretende implementar, pero de cuyo estudio, manipulación y prueba depende la realización del mismo.

Los dispositivos descritos a continuación también se encuentran dentro de la placa impresa pero constituyen parte del sistema a desarrollar debido a que se identifican como los elementos programables del mismo; ellos difieren de los dispositivos anteriormente mencionados entendidos como los elementos configurables al exterior del sistema:

- Un dispositivo (μC familia 8752) que ejecute a su cargo las funciones de control, inicialización o configuración inicial y diagnóstico, por medio de la interoperación con los actores del sistema.
- Un dispositivo (PLD Altera EPM5130JC) que permita interfazar adecuadamente:
  - Bus ISA PC $\Leftrightarrow$ Microcontrolador
  - Bus ISA PC $\Leftrightarrow$ Conmutador
  - Microcontrolador $\Leftrightarrow$ Conmutador

Fuera de la tarjeta y dentro del PC, donde esta se encuentra alojada, se halla el elemento manejador de la misma:

- Este controlador o driver establece un puente de comunicación interfazando un actor operador (software o persona) con la tarjeta. De esta forma se permite al operador manejarla transparentemente a través de una serie de funciones, es decir, sin necesidad de que conozca su arquitectura física y configuración

interna, evitando el contacto directo con ella para vislumbrarla en últimas como una "caja negra".

1.2 DEFINICION DE LOS PROPOSITOS DEL SISTEMA

Alrededor del año 1994 en el proyecto Concentrador Telefónico Digital CTD, se desarrolló una tarjeta de troncal digital, de esta queda el prototipo, jamás probado, que se montó en esos días, el esquemático de la misma y un documento con sus especificaciones técnicas en donde se nombran los diferentes componentes que se emplearon en su fabricación. Los códigos del hardware y el software de la misma no fueron localizados. El propósito general del proyecto entonces, es modelar, desarrollar y documentar un sistema que se encargue de controlar las funciones básicas de enrutamiento, inserción y detección de señalización, lectura de datos, configuración y diagnóstico de la tarjeta siguiendo una metodología de desarrollo específica.

Ya dentro del entorno de SMART, el proyecto pretende proveer el soporte físico mínimo requerido para un SSP (Punto de Conmutación de Servicio) que realice una conmutación digital básica, mediante las funciones que ofrece la Tarjeta de Interfaz de Troncal Digital TITD, para lograr una eficiente y confiable comunicación entre la plataforma SMART y los usuarios telefónicos normales de una administración local quienes harán parte de una Red Inteligente.

Se busca también ejercitar el Proceso Unificado para el Desarrollo de Programas RUP como metodología de desarrollo, utilizando UML como herramienta de modelado, buscando tanto la robustez del sistema como la uniformidad, en cuanto a metodología de trabajo, con los demás proyectos de SMART.

Se pretende además, llevar a cabo la revisión del hardware de la TITD y proponer un esquema con componentes más actualizados con el objetivo de amoldar la

tarjeta a las nuevas tecnologías de sistemas empotrados con circuitos integrados especializados.

1.3 IDENTIFICACION DE LAS FUNCIONES MEDIANTE LA CONSTRUCCION DEL ARBOL DE FUNCIONES

1. ATENCIÓN AL SOFWARE OPERADOR
   - 1.1 Enrutar Canal
   - 1.2 Introducir Señalización
   - 1.3 Efectuar Lectura
   - 1.4 Configurar Tarjeta o Sistema
   - 1.5 Diagnosticar (Test)
   - 1.6 Acceder a HDLC
2. ATENCIÓN AL HARDWARE
   - 2.1 Funciones del Conmutador
     - 2.1.1 Conmutar
     - 2.1.2 Enviar Mensaje
     - 2.1.3 Entregar Dato
   - 2.2 Funciones de la Interfaz E1
     - 2.2.1 Adoptar Configuración
     - 2.2.2 Insertar Señalización
     - 2.2.3 Entregar Dato
   - 2.3 Funciones del PLL
     - 2.3.1 Adoptar Configuración PLL
   - 2.4 Funciones del HDLC
     - 2.4.1 Recibir Dato
     - 2.4.2 Entregar Dato

1.4 IDENTIFICACION DE ATRIBUTOS Y RESTRICCIONES

ATRIBUTOS

- El sistema trabaja u opera con E1 y no T1.
- Consta de 4 PCMs internos de entrada y 4 internos de salida.
- Consta de 1 PCM E1 de entrada (Rx) y 1 PCM E1 de salida (Tx) para comunicación distante.
- El sistema permite el acceso al HDLC para su manejo, mas no contiene funciones específicas que lo manipulen y permitan su configuración.
- El sistema interfaza los diferentes componentes o dispositivos para lograr la comunicación entre los mismos ya que cada uno se comunica con su entorno de manera diferente.
- La tarjeta en conjunto permite la libre selección de la dirección base que la identifica como otro dispositivo I/O del PC por medio de selectores análogos.

RESTRICCICIONES

- El Microcontrolador trabaja bajo el modo de acceso a memoria externa.
- El elemento PLD está restringido en cuanto al número de macroceldas (<128) y a la disposición de pines (familia genérica).
- La arquitectura de la tarjeta (ver figura 1) ya se encuentra definida, es decir, se parte de un prototipo físico construido sobre una placa impresa. Entre los aspectos que determinan esta arquitectura y que ya se han definido se tienen: El sistema de interrupciones, el sistema de buses (ST-BUS), la temporización, el tipo de enlace (CEPT-E1) y la interfaz de I/O con respecto al PC (ISA).

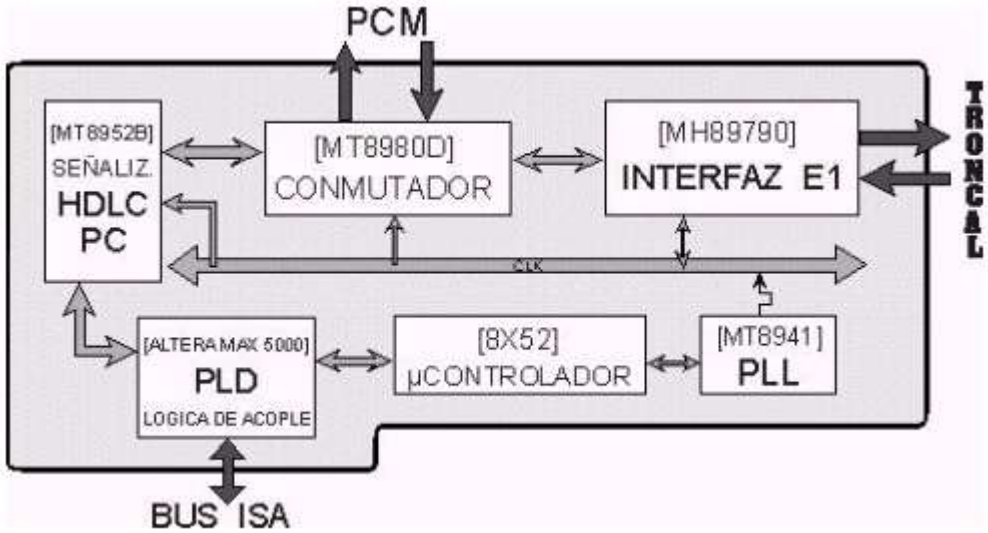


Figura 1. Diagrama Modular de la tarjeta TITD.

1.5 MAQUETAS DE LOS FORMATOS DE ENTRADA Y SALIDA

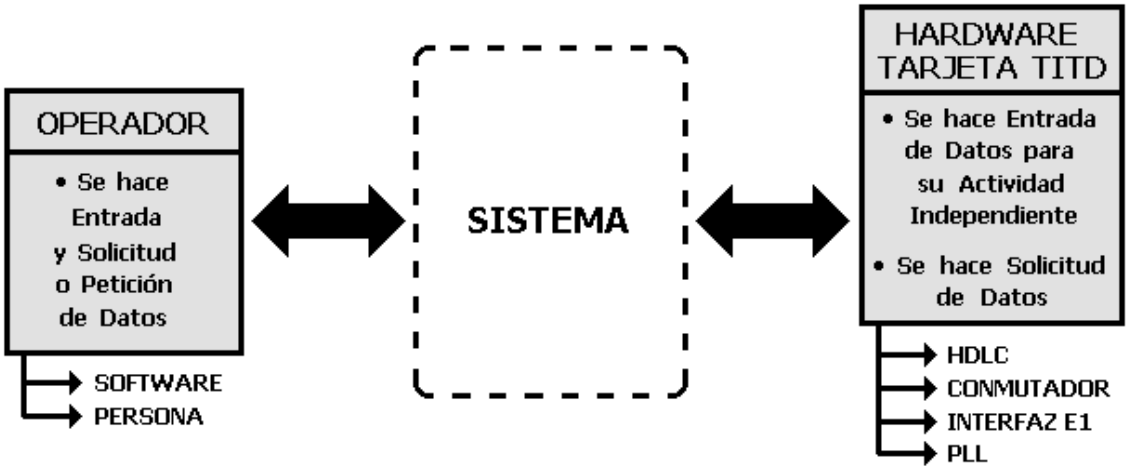


Figura 2. Entradas y salidas del sistema para la TITD.

2 MODELO DEL DOMINIO

La TITD hace parte del proyecto SMART (Sistema Modular para Aplicaciones de Redes Inteligentes y Telemática) que busca investigar y desarrollar un sistema de red inteligente basado en CORBA para ofrecer servicios de valor agregado universales a través de Internet y la PSTN. La RI está constituida por una arquitectura funcional, que a su vez está compuesta por un conjunto de entidades funcionales las cuales intercambian entre sí flujos de información, implementada sobre una arquitectura física que comprende las siguientes entidades:

- Sistema de Gestión o Administración de Servicios SMS.
- Punto de Control de Servicio SCP.
- Punto de Transferencia de Señalización STP.
- Punto de Conmutación de Servicio SSP.
- Punto de Datos del Servicio SDP.

El SCP es un nodo de la RI con capacidades de procesamiento en tiempo real con una disponibilidad comparable a la de una central telefónica. Debido a su demanda en la red requiere la capacidad de manejar grandes volúmenes de tráfico. En otras palabras es la inteligencia de la RI.

El SDP es la base de datos de la RI interfazada con el SCP.

El SMS es un nodo de la RI que realiza el control del diseño, implementación y administración de un servicio. Como ejemplo de sus funciones se pueden mencionar la administración de la base de datos de la lógica del servicio, la administración del tráfico y la recolección de datos de la red.

El SSP (Switching Service Point) es una central digital que se conecta con la PSTN la cual es el acceso a los nodos asociados de la RI. Tiene la funcionalidad de

reconocer una llamada que necesita un manejo lógico adicional y cooperar con la lógica externa en el tratamiento de las mismas. Con la característica de la conmutación del servicio, tiene la capacidad de conectarse a centrales de telefonía pública y debe permitir además interactuar con los demás componentes o nodos de la RI (ver figuras 3 y 4).

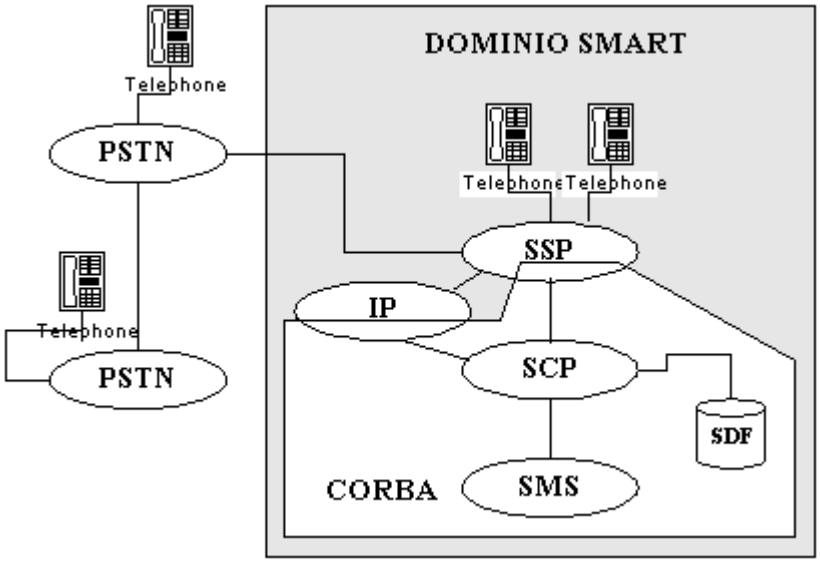


Figura 3. Estructura de Dominios SMART.

Es entonces, dentro del SSP, donde se encuentra ubicada la TITD como parte principal constituyente de la Central SMART de Conmutación la cual se conecta a una central local utilizando una troncal digital de 32 canales empleando señalización telefónica. También tiene enlace(s) PCM32 con el Periférico Inteligente IP, quien realiza la comunicación entre SSP $\Leftrightarrow$ SCP de la RI y además provee la funcionalidad de recursos especiales (ver figura 5).

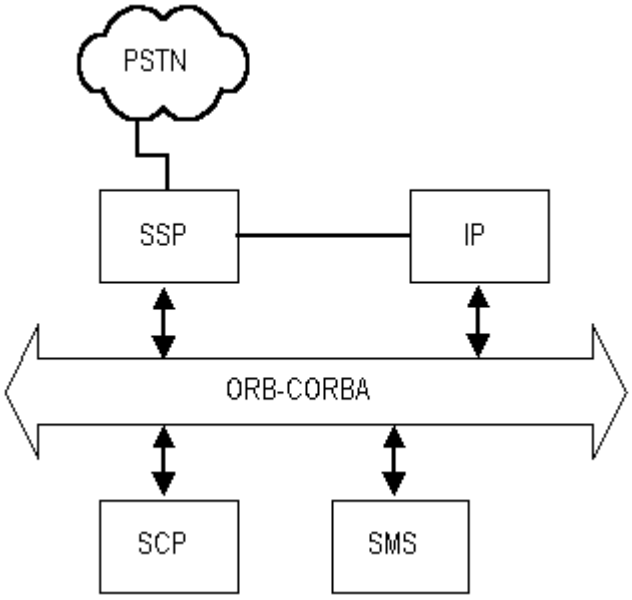


Figura 4. Arquitectura Distribuida de la RI SMART.

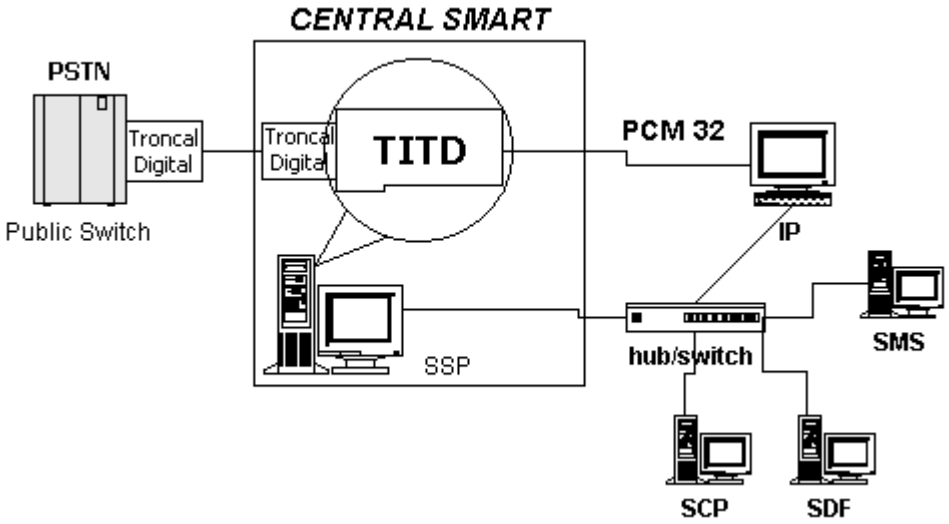


Figura 5. Distribución Física de SMART.

El proyecto de la TITD consiste en un sistema que actúa sobre un hardware especializado en telecomunicaciones y es manipulado desde un PC por un operador (ver figura 6).

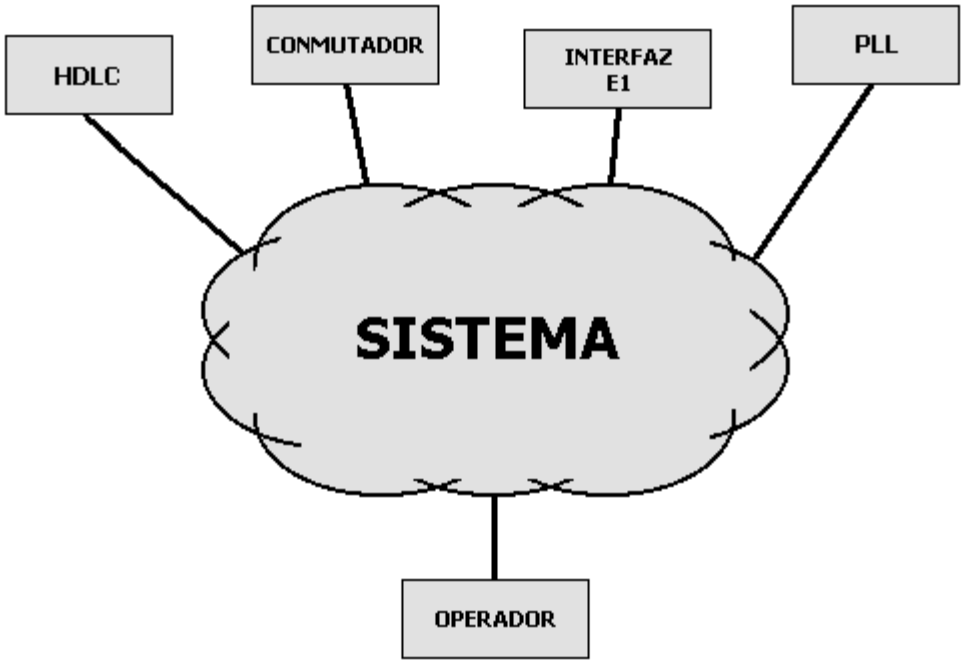


Figura 6. Modelo del dominio de la TITD general.

Dentro del proyecto de la TITD se pueden identificar 3 grandes módulos:

- **Módulo de Hardware** (Tarjeta): Conformado por la placa impresa con los circuitos integrados de telecomunicaciones (PLL, Conmutador e Interfaz E1), los dispositivos programables (PLD y Microcontrolador) y la circuitería lógica.
- **Módulo del sistema a desarrollar**: Conformado por el firmware o software empotrado en los dispositivos programables (PLD y Microcontrolador) y el driver manejador de la tarjeta.
- **Módulo contenido en el PC**: Conformado por el bus ISA, el driver manejador de la tarjeta y el software operador de la misma (ver figura 7).

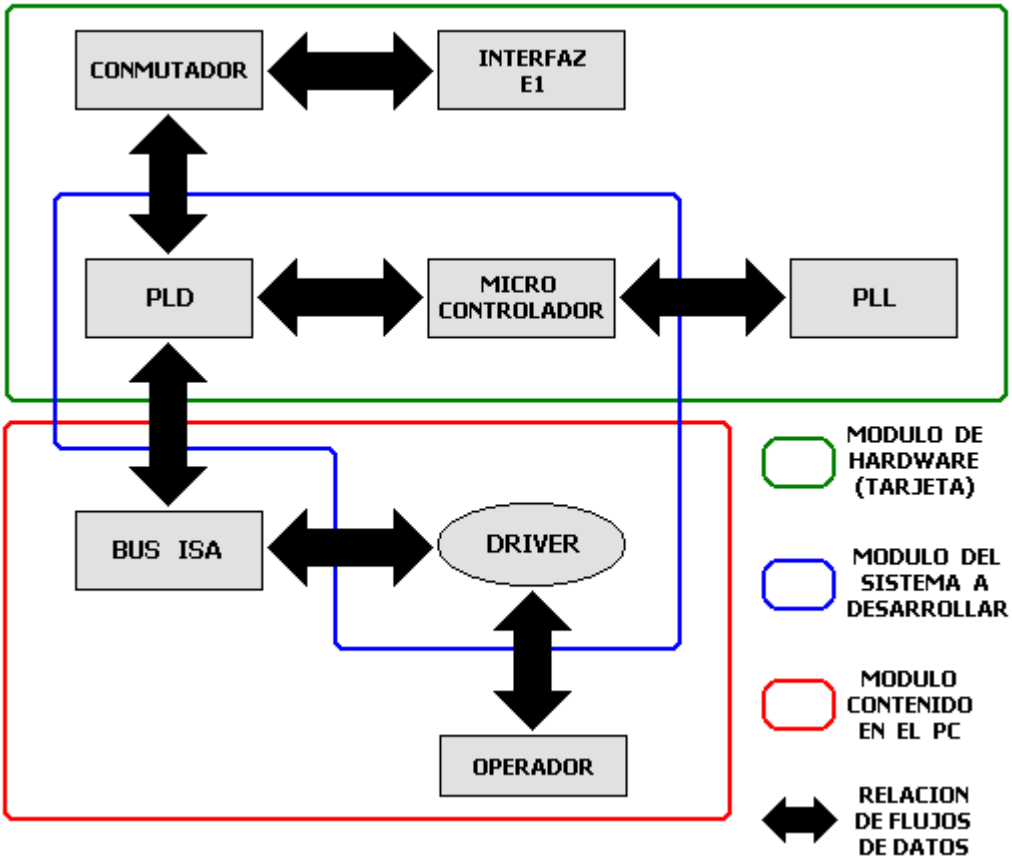


Figura 7. Modelo del dominio de la TITD específico.

3 DICCIONARIO DE DATOS

Acceso a Memoria Externa	Configuración particular del Microcontrolador en la cual este asume que todos los dispositivos con los que se comunica son memoria externa que debe ser mapeada para poder ser accesada.
AHDL	Lenguaje de descripción de Hardware propietario de Altera.
BUS	Grupo de cables o hilos dentro del sistema ordenador a través de los cuales se transmiten las señales eléctricas desde una sección funcional a otra.
Bus de Control	Bus que permite acceder a las órdenes o información de estados que coordinan la operación del sistema.
Bus de Datos	Bus bidireccional que permite acceder a la información (instrucciones y datos) entre subsistemas.
Bus de	Bus que permite acceder a la dirección del dispositivo seleccionado

Direcciones	para efectuar una transferencia.
Código de Línea	Forma apropiada de representación de la información binaria (señal digital codificada) para la transmisión a grandes distancias.
CORBA	Common Object Request Broker Architecture
CRC	El Chequeo de Redundancia Cíclica es un test para la detección de errores utilizado tanto en el almacenamiento como en la transmisión de datos.
Dirección de E/S	Hace referencia a un dispositivo de I/O en particular y es la dirección cuyo valor se coloca en el bus de direcciones del PC.
Dirección de memoria	Hace referencia a una zona de memoria en particular. En un PC el direccionamiento de memoria se realiza mediante la combinación de un segmento y un desplazamiento.
Driver	Controlador o Manejador. Programa que controla un dispositivo.
E1	Los enlaces tipo E1 o CEPT, de origen europeo, se establecen basándose en sistemas PCM multiplexados en el tiempo de 32 Its, con una velocidad de bit de 2048 Kbps, según la ley A especificada en la recomendación ITU-T G.732.
ECP	Puerto Paralelo de Capacidades Extendidas.
HDLC	Protocolo para transmisión de datos orientado a detección y corrección de errores. El dispositivo HDLCPC utiliza este protocolo para entramar y dar formato a paquetes de datos de acuerdo con las recomendaciones de la ITU-T X.25 (Nivel 2).
IP	Dentro de la arquitectura física de una RI es el Periférico Inteligente (comunica SSP con SCP).
ISA	Industry Standard Achitecture. Estándar para la transferencia de información entre tarjetas de expansión y la tarjeta madre de un PC.
IT	Intervalo de tiempo correspondiente a un canal de voz o de datos de 8 bits en sistemas PCM multiplexados en el tiempo cuya duración es de 3,9 μs en enlaces tipo E1.
Macrocelda	Mínima unidad lógica programable de un PLD.
PCM	Técnica de Modulación por Pulsos Codificados que permite la conversión de señales análogas a digitales a través de procesos de muestreo, cuantificación y codificación.
PLD	Dispositivo Lógico Programable constituido por celdas lógicas en el cual se puede implantar, por medio de programación, circuitos lógicos desarrollados mediante un lenguaje de descripción de hardware.
PLL	Dispositivo digital de ciclo cerrado de fase (Phase-Locked Loop) responsable de proveer las señales de temporización y sincronización en sistemas de telecomunicaciones.

PSTN Red Telefónica básica Públic7a Conmutada.

Puerto En un PC es el lugar específico de conexión con otro dispositivo, generalmente mediante un conector.

RI Una Red Inteligente es una arquitectura para redes de telecomunicaciones que permite la introducción de nuevos servicios y la modificación de los existentes de manera eficiente y flexible, facilitando su control y administración.

SCP Dentro de la arquitectura física de una RI es el Punto de Control de Servicio.

SDP Dentro de la arquitectura física de una RI es el Punto de Datos de Servicio (base de datos).

Señalización Proceso de generación y manejo de información necesaria para el establecimiento en los sistemas telefónicos.

SMS Dentro de la arquitectura física de una RI es el Sistema de gestión o Administración de Servicios.

SSP Dentro de la arquitectura física de una RI es el Punto de Conmutación de Servicio (central digital).

ST-BUS Bus serial síncrono multiplexado en el tiempo cuyos flujos de datos operan a una rata de 2048 Kbps y están distribuidos en 32 canales de 64 Kbps. Este bus es propietario de Mitel (Zarlink).

STP Dentro de la arquitectura física de una RI es el Punto de Transferencia de Señalización.

T1 Los enlaces tipo T1, de origen norteamericano, se establecen basándose en sistemas PCM multiplexados en el tiempo de 24 ITs, con una velocidad de bit de 1544 Kbps, según la ley u especificada en la recomendación ITU-T G.733.

TITD Tarjeta Interfaz de Troncal Digital

Troncal Los enlaces troncales se caracterizan por establecerse a grandes distancias, manejar altas velocidades y poseer gran capacidad de transporte de información. Normalmente enlazan centrales.

TST Tipo de conmutación en la cual la información se somete a un enrutamiento que implica un cambio de medio tanto temporal como espacial.

VHDL Lenguaje de descripción de Hardware para circuitos integrados de muy alta velocidad que permite la especificación de sistemas digitales a nivel de comportamiento y estructura.

4 MODELO DE CASOS DE USO

4.1 DIAGRAMA DE CASOS DE USO

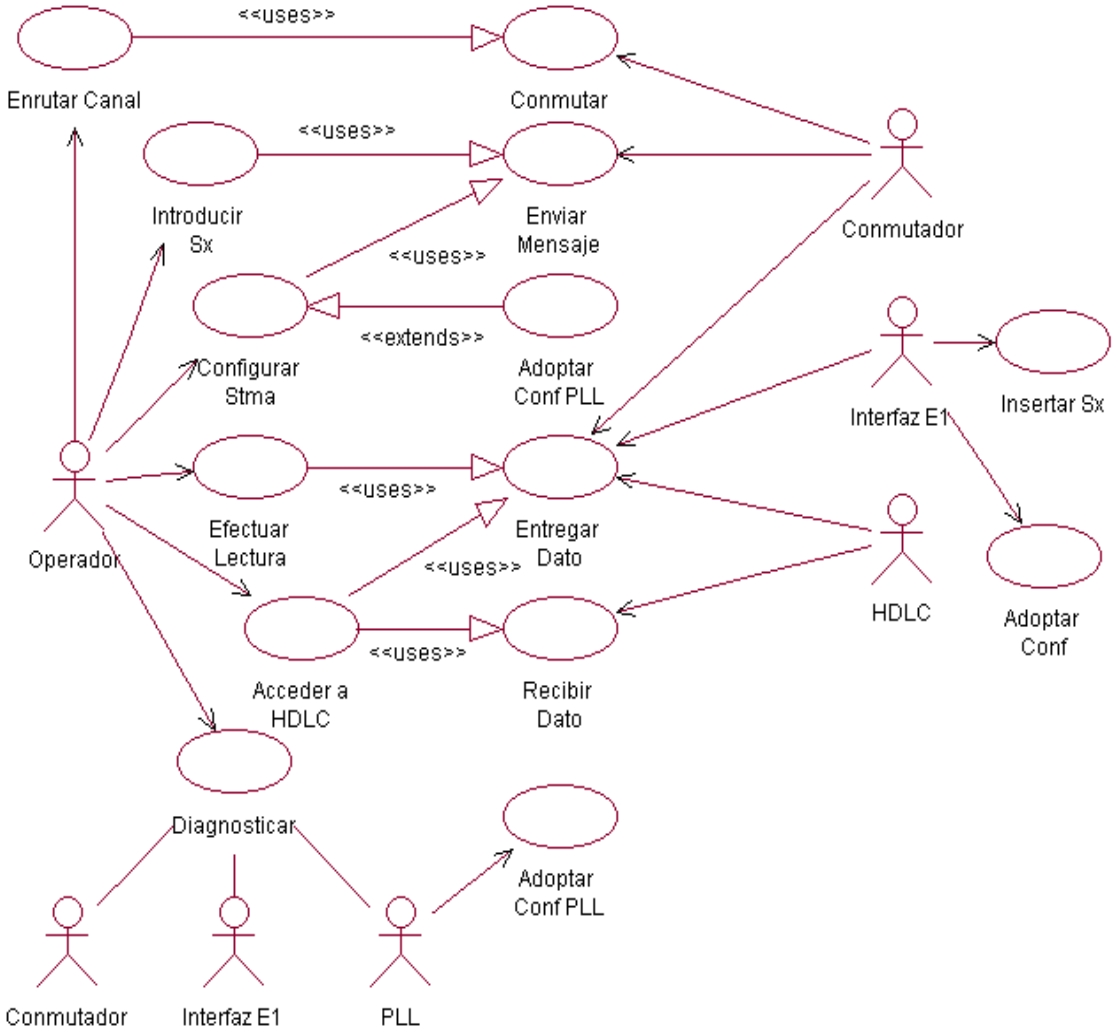


Figura 8. Diagrama de casos de uso.

4.2 DESCRIPCION DE LOS CASOS DE USO

ACTORES

- **Operador**: Es el encargado de manipular y utilizar el sistema y como actor puede ser una persona o un software. Se caracteriza por enviar hacia y recibir datos del sistema. El operador es quien debe interactuar directamente con la interfaz de usuario del sistema.
- **Conmutador**: Este actor hardware se encuentra ubicado dentro de la tarjeta y es quien se encarga de realizar el encaminamiento bidireccional de los datos en lo referente a datos correspondientes a los canales telefónicos y datos de configuración de otros actores. Se caracteriza por enviar hacia y recibir datos del sistema.
- **Interfaz E1**: Este actor hardware se encuentra ubicado dentro de la tarjeta y es el encargado tanto de insertar y detectar la señalización, como de adaptar los datos de los canales telefónicos a la troncal digital en transmisión y adecuar dichos datos al sistema en recepción. Se caracteriza por enviar hacia y recibir datos del actor conmutador y recibir señales de temporización del actor PLL.
- **PLL**: Este actor hardware se encuentra ubicado dentro de la tarjeta y se encarga de proporcionar la temporización al sistema. Se caracteriza por enviar señales de temporización a los actores hardware y recibir datos del sistema.
- **HDLC**: Este actor hardware se encuentra ubicado dentro de la tarjeta y se encarga de generar tramas ST-BUS siguiendo el protocolo HDLC. Se caracteriza por enviar hacia y recibir datos del sistema.

CASOS DE USO

- **Enrutar Canal**: Función que permite definir un camino adecuado de la información PCM bajo las características y capacidades de la tarjeta.
- **Introducir Sx**: Función que entrega la señalización al sistema.

- **Configurar Stma**: Función que entrega al sistema su configuración según las características de este y del enlace.
- **Efectuar Lectura**: Función que permite obtener datos del sistema.
- **Acceder a HDLC**: Función que abre una puerta de acceso entre el actor operador y el actor HDLC, aunque el sistema no alcanza a manipular la funcionalidad de este último.
- **Diagnosticar**: Función que permite hacer un test al sistema y sus actores hardware para obtener un diagnóstico de su estado.
- **Conmutar**: Función que define el camino para los ST-BUS de entrada y salida bajo las características y capacidades del actor conmutador.
- **Enviar Mensaje**: Función que permite el envío un dato específico por parte del sistema hacia los ST-BUS de salida por intermedio del actor conmutador.
- **Entregar Dato**: Función que permite al sistema obtener datos de los actores hardware a excepción del actor PLL.
- **Insertar Sx**: Función que permite al actor Interfaz E1 ejecutar la inserción de la señalización en las tramas que van hacia la troncal a nivel de bits.
- **Adoptar Conf**: Función que permite al actor Interfaz E1 asumir la configuración especificada por el sistema.
- **Recibir Dato**: Función que le permite al actor HDLC recibir datos suministrados por el sistema.
- **Adoptar Conf PLL**: Función que permite al actor PLL asumir la configuración especificada por el sistema.

5 ANALISIS DE RIESGOS

Los riesgos que a continuación se listan se han organizado en orden de prioridad de acuerdo al nivel de gravedad del riesgo y el grado de dificultad que acarrea su posible solución:

- **Imperfección o defectos del circuito impreso** que constituye el prototipo de la tarjeta TITD sobre la cual se va a trabajar. Su probabilidad de ocurrencia es muy baja ya que dicho prototipo se manufacturó bajo minuciosa supervisión técnica. Se hace necesario que el equipo desarrollador atienda este riesgo en la primera etapa del proyecto para determinar la elaboración de un nuevo prototipo.
- **Defectos en los circuitos integrados especializados en telecomunicaciones**. Su probabilidad de ocurrencia es media ya que el prototipo se construyó hace muchos años. Se hace necesario que las directivas del proyecto atiendan este riesgo gestionando la adquisición de otros circuitos integrados de soporte.
- **Imposibilidad para programar el PLD** de Altera debido a defectos del CI o por falla del equipo programador. Su probabilidad de ocurrencia es media ya que tanto el CI como el equipo programador han sido adquiridos hace ya bastantes años. Es posible la adquisición de un CI nuevo o migrar a otro chip equivalente teniendo en cuenta las consecuencias que de ello se derivan.
- **Imposibilidad para programar el Microcontrolador** por falla de este o del equipo programador. Su probabilidad de ocurrencia es media ya que el equipo programador ha sido adquirido hace ya bastantes años. Es muy posible la adquisición de un Microcontrolador equivalente actualizado con características de disposición de pines similares.
- **Adaptabilidad de la metodología para software intensivo orientado a objetos al proyecto**. Su probabilidad de ocurrencia es alta ya que SMART se ha propuesto buscar la uniformidad en sus diferentes proyectos utilizando la metodología RMICS, sin embargo, el proyecto de la TITD tiene alto contenido hardware razón por la cual dicha metodología no ha sido optimizada (por falta de experiencia) para el desarrollo de este tipo de proyectos. Se ha hecho necesario que el equipo desarrollador estudie y analice esta metodología con el fin de seleccionar las herramientas que realmente contribuyan y se adecuen a los criterios metodológicos que conlleven a un óptimo desarrollo del proyecto.

- **Imposibilidad de llevar a cabo**, si fuese necesario hacerlo, **la Migración Vertical**, que se define como la evolución tecnológica de un chip, en este caso el PLD de Altera MAX 5000, a uno nuevo con capacidades superiores de tal manera que conserve sus características físicas externas (encapsulado y disposición de pines). Su probabilidad de ocurrencia es alta debido a que Altera no trabajaba con migración vertical cuando diseñó la familia MAX 5000. Es posible no realizar la migración o migrar y que el equipo desarrollador recurra a la técnica de puenteo sobre un socket que aloje el nuevo chip.
- **Carencia de las licencias de las herramientas software a utilizar** tales como: software de Altera y de desarrollo del driver. Su probabilidad de ocurrencia es muy baja ya que la FIET procede adecuada y rigurosamente para evitar ese tipo de problemas. La gestión de la FIET, a través de los distintos convenios y seminarios-taller, ha logrado establecer vínculos con los distribuidores legalizando los paquetes que estos promocionan; además en Internet se encuentra a disposición una cantidad considerable de herramientas de desarrollo de buena calidad y libre distribución.
- **Tiempos de estimación de desarrollo del software excedidos**. Su probabilidad de ocurrencia es media ya que todos los demás riesgos afectan en cierta medida estos tiempos de desarrollo. El análisis de los riesgos del proyecto y los criterios con que estos se atacan hacen parte del desarrollo del proyecto y son evaluados dentro de la etapa de planeación del mismo.
- **Insuficiencia en la capacidad lógica** (número de macroceldas a usar) **del PLD** en la aplicación a implementar. Su probabilidad de ocurrencia es media ya que esta aplicación involucra gran parte del sistema a implementar. Se hace necesario que el equipo diseñador desarrolle una aplicación robusta y eficiente.
- **Resultados óptimos no presentados en la etapa de pruebas** debido a la manipulación de altas frecuencias con elementos no adecuados para su manejo (protoboards o Qts, cables, interferencia de equipos en su entorno). Su probabilidad de ocurrencia es alta. El equipo desarrollador está en la capacidad de aplicar las técnicas pertinentes adquiridas para la ejecución de esta etapa,

entre ellas se destacan: manejo de tierras, eliminación de ruidos, campos y frecuencias espúreas en altas frecuencias, técnicas de cableado.

6 PLAN DEL SISTEMA

6.1 ESTIMACION DE LA METRICA

DOCUMENTACION

- **Anteproyecto**, elaboración y aceptación.
- **Estudio** y repaso de los fundamentos **teóricos** relativos al tema central que cobija la parte de prueba de los circuitos integrados especializados, las técnicas de modulación por codificación de pulsos (PCM), velocidades y estructura de entramado, conmutación digital TST, codificación de línea, chequeo de redundancia cíclica (CRC), señalización por canal, interfaces con el puerto paralelo de capacidades extendidas (ECP).
- Referente a la etapa de **estudio** de los **circuitos integrados** especializados, su arquitectura, configuración y funcionamiento (transcripción).
- **Estudio** para manipulación del equipo **analizador lógico HP**.

IMPLEMENTACION 1:

- **Software** de **prueba y configuración** del **puerto paralelo** Modo ECP (OLPT2.C).
- **Software** de **configuración y prueba** del **conmutador** a través del puerto paralelo y para posterior prueba de la interfaz.
- **Interfaz** física entre **LPTs** y **Circuitos integrados**.
- **Montaje** de circuitos de **prueba** de los integrados especializados, **cableado** PLL, Conmutador e interfaz E1 y pruebas.

MODELAMIENTO:

- **Estudio** del Lenguaje Unificado de Modelado (**UML**), el Proceso Unificado (**RUP**) y la herramienta **Rational Rose**.
- **Análisis** de **Requerimientos**.
- **Análisis** del **Sistema**.
- **Diseño** del **Sistema**.

IMPLEMENTACION 2:

- **Implementación** y **Simulación** del Sistema TITD para el **PLD** con la herramienta Max Plus II de Altera.

METRICA:

ETAPA	PUNTO	TIEMPO	# LINEAS DE CODIGO
Documentación	Anteproyecto	4 meses Oct-Nov-Feb-Marz	
	Estudio teoría	2 meses Feb-Marz	
	Estudio CIs y Analizador lógico	3 meses Marz-Abr-May	
Implementación 1	Software prueba y Configuración LPT (ECP)	2 semanas Jun	260
	Software prueba y Configuración Conmutador	1 mes Jun-Jul	510
	Interfaz LPT $\Rightarrow$ Cis	1 semana Jul	
	Montaje, prueba y Cableado	2 semanas Jul-Ago	

Modelamiento	Estudio UML, RUP y Rational Rose.	2 semanas Ago	
	Análisis Requerimientos	2 semanas Ago-Sep	
	Análisis Sistema	3 semanas Sep	
	Diseño Sistema	5 semanas Oct-Nov	
Implementación 2	Implementación y Simulación PLD	3 semanas Nov	400

Tabla 1. Tiempos de estimación para la métrica.

6.2 PRIORIZACION DE LOS CASOS DE USO

Los casos de uso listados enseguida se han priorizado según el orden en que deben ser desarrollados, teniendo en cuenta el nivel de complejidad que su diseño e implementación implica y su grado de importancia dentro del sistema:

- **Configurar Stma**: Este caso de uso de alto nivel es el más complejo y debe ser implementado en su totalidad. Está constituido por casos de uso con niveles de abstracción más bajos.
- **Diagnosticar**: Este caso de uso de alto nivel tiene alto grado de complejidad y también debe ser implementado en su totalidad. Está constituido por casos de uso con niveles de abstracción más bajos.
- **Efectuar Lectura**: Este caso de uso de alto nivel debe ser implementado por completo. Debe ser extendido en casos de uso con niveles de abstracción más bajos.
- **Enrutar Canal**: Este caso de uso de alto nivel debe ser implementado en su totalidad y está constituido por casos de uso con niveles de abstracción más bajos.

- **Introducir Sx**: Este caso de uso de alto nivel debe ser implementado por completo y es necesario extenderlo en casos de uso con niveles de abstracción más bajos.
- **Acceder a HDLC**: Este caso de uso debe ser implementado y se debe explicar su comportamiento en cuanto al flujo de datos.
- **Entregar Dato**: Este caso de uso de alto nivel ya está implementado, pero debe ser extendido para indicar cómo se lleva a cabo el flujo de datos.
- **Enviar Mensaje**: Este caso de uso ya está implementado, pero se debe explicar su comportamiento en cuanto al flujo de datos, es decir, cómo se han definido sus entradas y salidas.
- **Conmutar**: Este caso de uso ya está implementado, pero se debe indicar su comportamiento en cuanto al flujo de datos.
- **Insertar Sx**: Este caso de uso ya se encuentra implementado, pero se debe indicar cómo se lleva a cabo el flujo de datos.
- **Adoptar Conf I-E1**: Este caso de uso ya está implementado, pero se debe explicar cómo se ha definido esta configuración.
- **Adoptar Conf PLL**: Este caso de uso ya está implementado y se debe explicar cómo se ha definido dicha configuración.
- **Recibir Dato HDLC**: Este caso de uso ya está implementado y se debe indicar cómo se lleva a cabo el flujo de datos.

7 REFERENCIAS BIBLIOGRAFICAS

- Microelectronics Digital Communications Handbook.
  Mitel Semiconductor (ZARLINK).
- FOWLER Martin y SCOTT Kendal. UML gota a gota.
  Prentice Hall. México (MX). 1999.
- C-156-M. Diagnóstico, Integración y Puesta a Punto del sistema Físico de la Central Digit 1000. Monografía por Luis Fabián Troyano y Carlos Alberto Martínez. FIET UCAUCA. 1992.
- Especificaciones Técnicas de la Tarjeta Troncal Digital (TD).
  Proyecto Concentrador Telefónico Digital. FIET UCAUCA.
- HP 1650A/HP1651A Logic Analizers.
  Getting Started Guide. Setting Up the Logic Analizer.
  Hewlett Packard Handbook.
- RENDON Alvaro y OSPINA Hernando. Sistemas de Conmutación Digital.
  Conferencias. FIET UCAUCA. 1992.
- Memorias: Seminario Taller Desarrollo de Sistemas lógicos basados en FPGAs. VHDL. 2000.
  Seminario Taller Nuevas Técnicas de Diseño lógico usando dispositivos de lógica programable y lenguajes de descripción Hardware AHDL. Altera. 2001.
- MCS 51 Microcontroller Family Users manual. Versión Pdf. INTEL. 1994.
- RENDON Alvaro. Apuntes sobre el proceso unificado RUP para el desarrollo de programas. Versión Pdf. FIET UCAUCA. 2000.
- RENDON Alvaro. El Lenguaje Unificado de Modelado UML. Versión Pdf.
  FIET UCAUCA. 2000.
- ALTERA. MAX 5000 Programmable Logic Device Family. Data sheet. 1999.
- ÖGREN Joakim. The Hardware Book. Versión Pdf. ECP Parallel Technical and ISA Bus Technical.
- Interfacing the Standard Parallel Port and the Extended Capabilities Parallel Port. Versión Pdf.

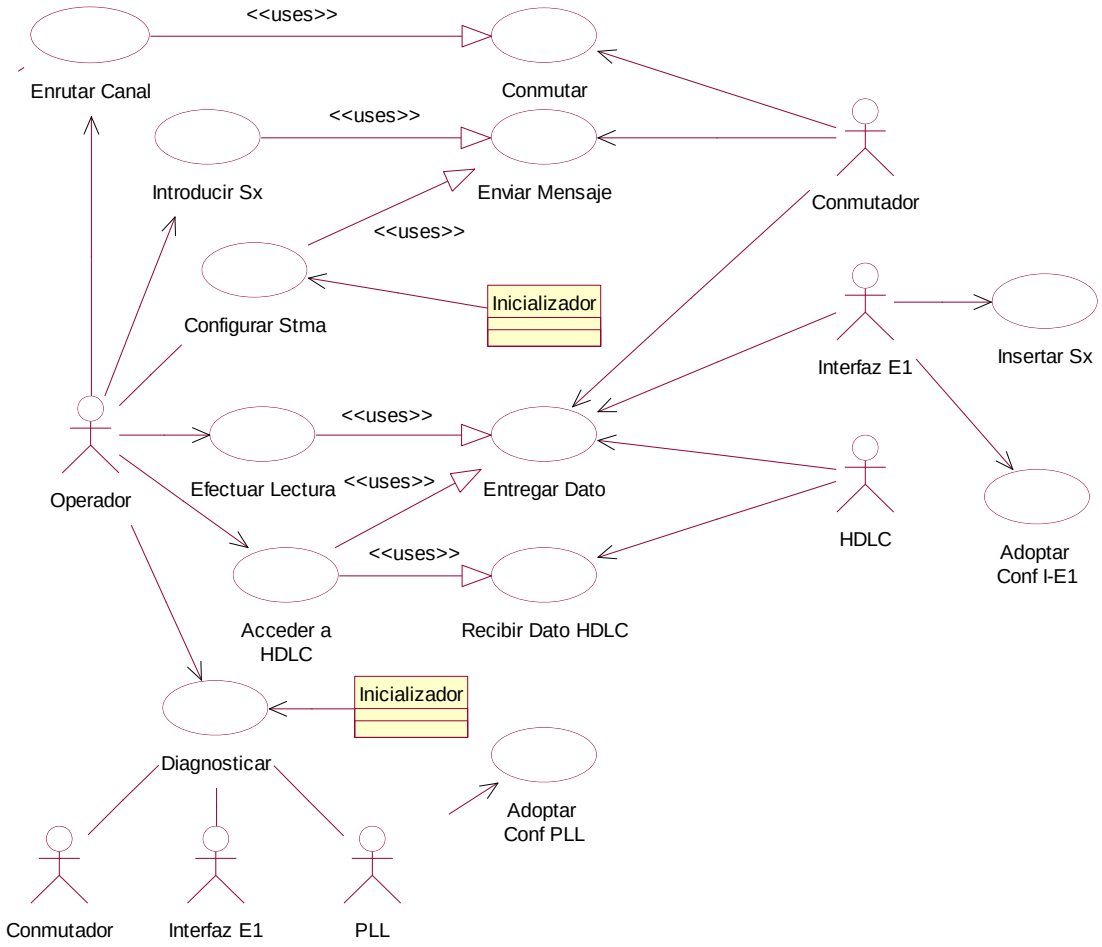
TABLA DE CONTENIDO

CAPITULO 1

CAPITULO 2
ANALISIS DEL SISTEMA

1 ANALISIS DE LOS CASOS DE USO

1.1 DIAGRAMA DE CASOS DE USO DE ANALISIS



2 ESCENARIOS DE LOS CASOS DE USO

2.1 CASO DE USO CONFIGURAR STMA

2.1.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador

Precondición: Ninguna.

Flujo de eventos:

1. El Inicializador, conociendo la arquitectura de la tarjeta, entrega los parámetros de la configuración inicial de la misma y va al flujo 4. Cuando el operador desea utilizar este caso de uso, introduce el nombre y el valor de los parámetros correspondientes a los datos de configuración de la tarjeta.
2. El sistema recoge los datos y los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la tarjeta y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para entregarle los datos de configuración (suministrados por el inicializador ó por el operador) por medio de una serie de secuencias.
5. Implícitamente se encuentra, dentro de los datos entregados, el modo de trabajo del conmutador, el cual ocasiona el llamado de la función Enviar Mensaje de este actor.

Postcondiciones:

- La tarjeta asume la configuración.
- Operador con posibilidad de actualizar la configuración o hacer uso de las demás funciones que ofrece el sistema.

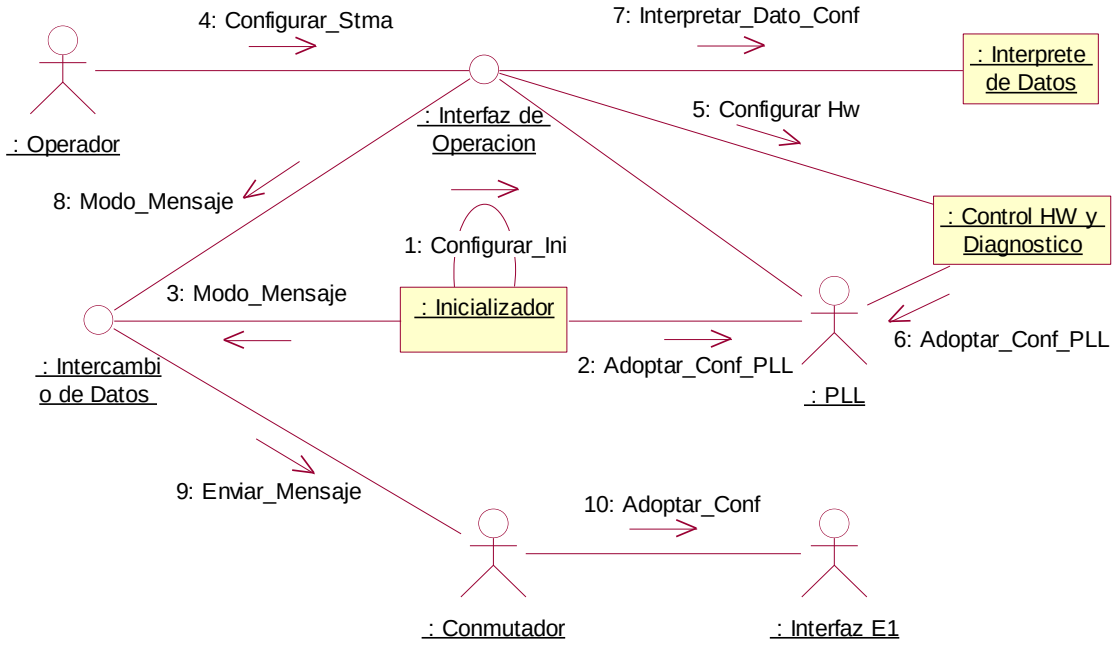
Flujos alternativos: Ninguno.

Excepciones:

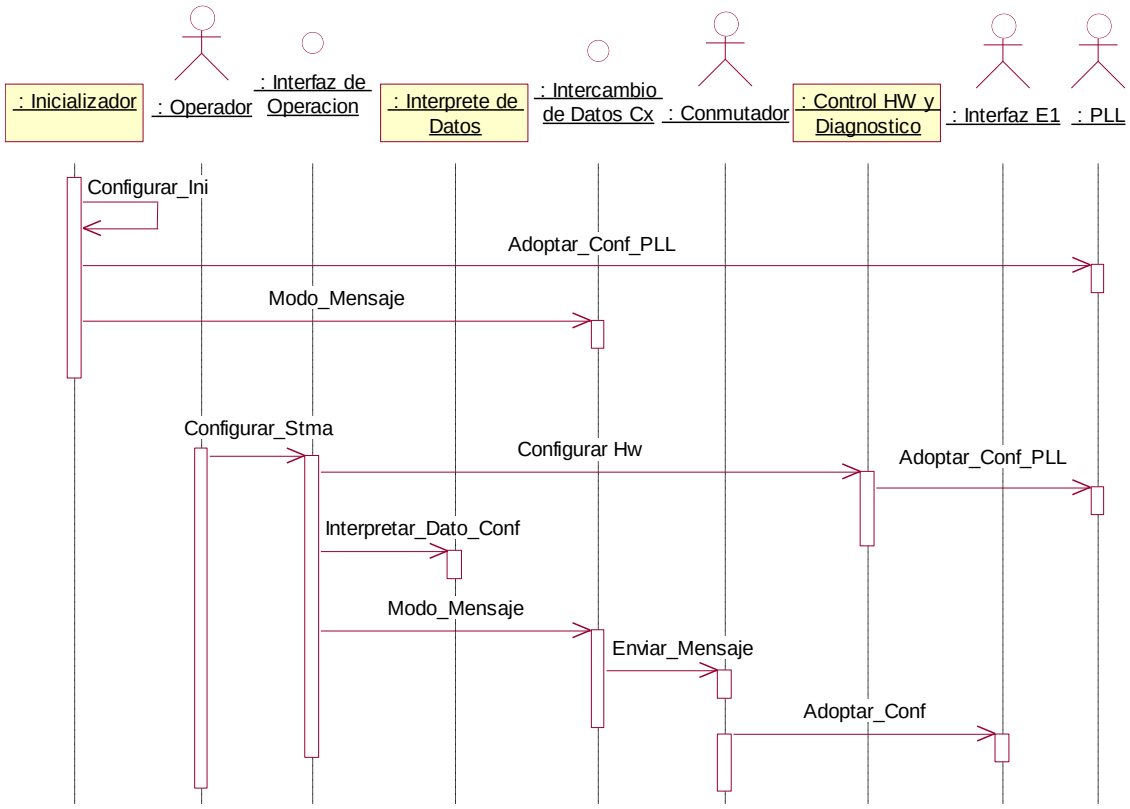
- Información fuera de rango por parte del operador.
- Fallas de comunicación.

Recursos especiales: Ninguno.

2.1.2 DIAGRAMA DE COLABORACION (INTERACCION)



2.1.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



2.2 CASO DE USO DIAGNOSTICAR

2.2.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador o Inicializador.

Precondición: Ejecución de la configuración inicial de la tarjeta.

Flujo de eventos:

1. El operador o el inicializador da la orden de proceder con el diagnóstico.
2. El sistema lleva a cabo la prueba de cada uno de los módulos de la tarjeta.
3. El sistema entrega el resultado de las pruebas de diagnóstico.

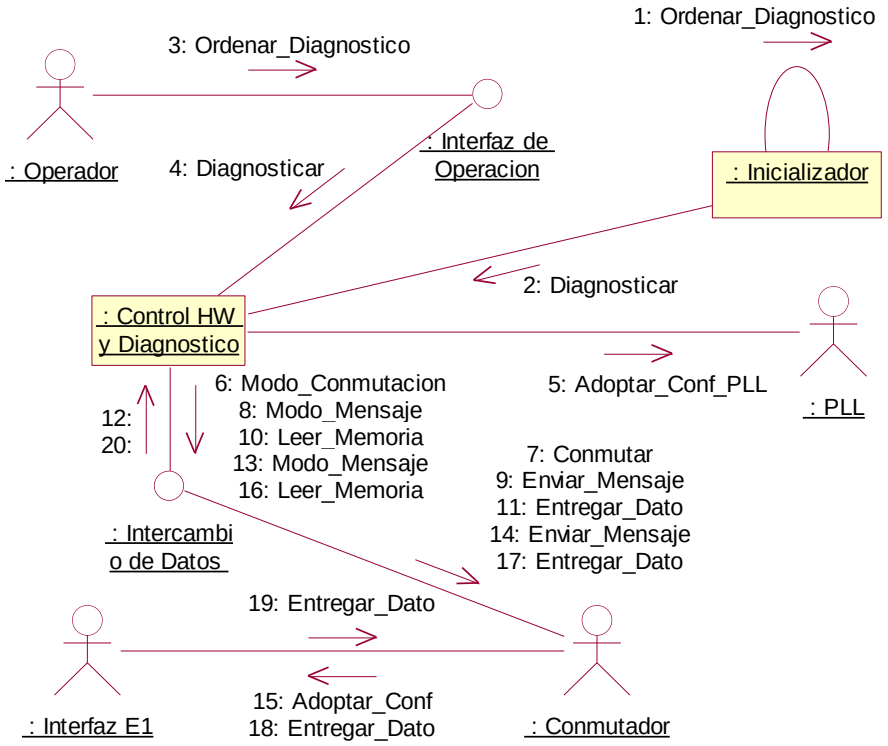
Postcondiciones: Operador con posibilidad de ordenar la ejecución de otro diagnóstico o hacer uso de las demás funciones que ofrece el sistema.

Flujos alternativos: Ninguno.

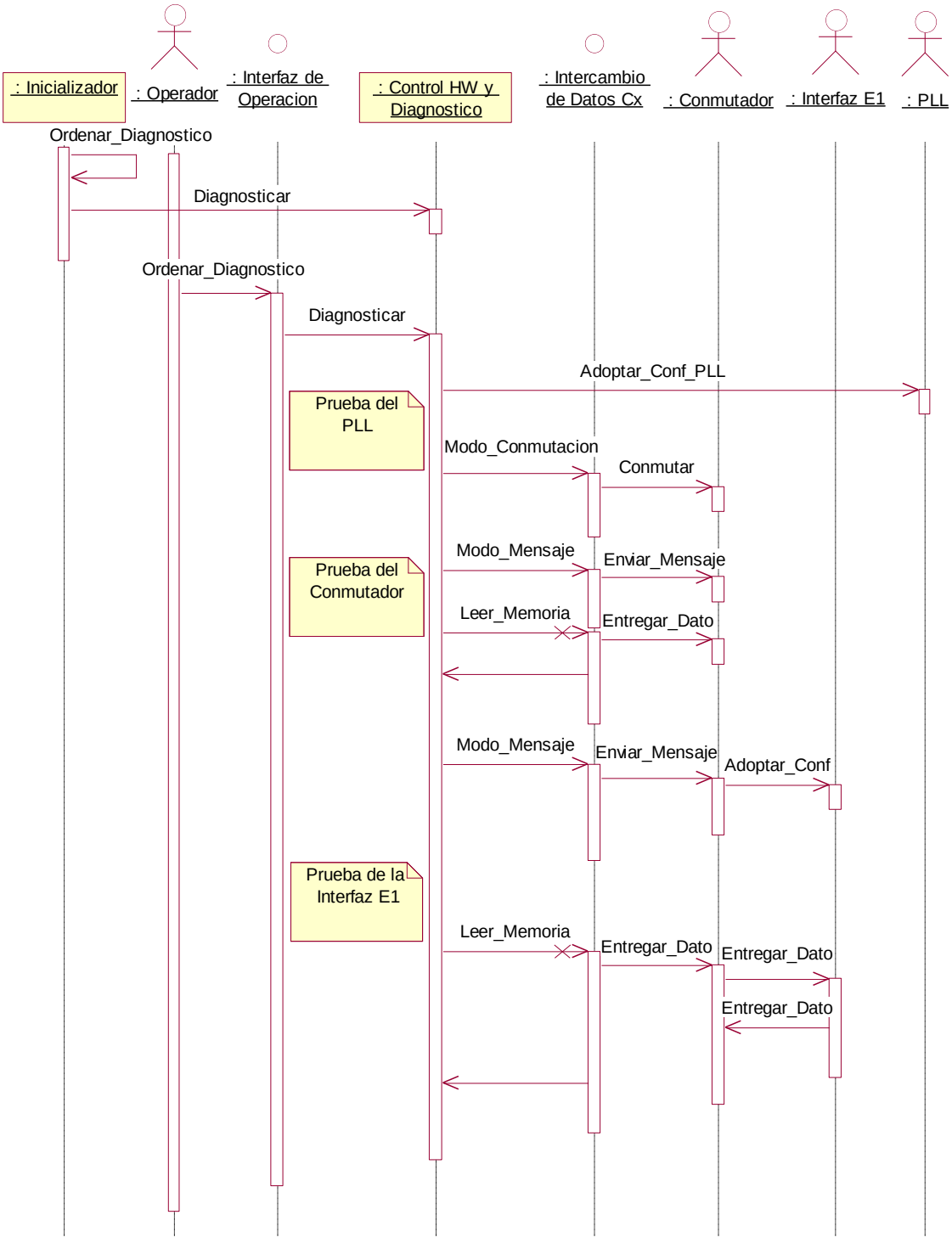
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

2.2.2 DIAGRAMA DE COLABORACION (INTERACCION)



2.2.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



2.3 CASO DE USO EFECTUAR LECTURA

2.3.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los datos que identifican el parámetro a leer.
2. El sistema recoge los datos y los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la tarjeta y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para acceder a los datos solicitados por el operador por medio de una serie de secuencias.
5. Conmutador ó Interfaz E1 efectúa el llamado de la función Entregar Dato.
6. El sistema proporciona los datos solicitados para lectura al operador.

Postcondiciones:

- Operador obtiene los datos de su petición de lectura.
- Operador con posibilidad de efectuar una nueva lectura o hacer uso de las demás funciones que ofrece el sistema.

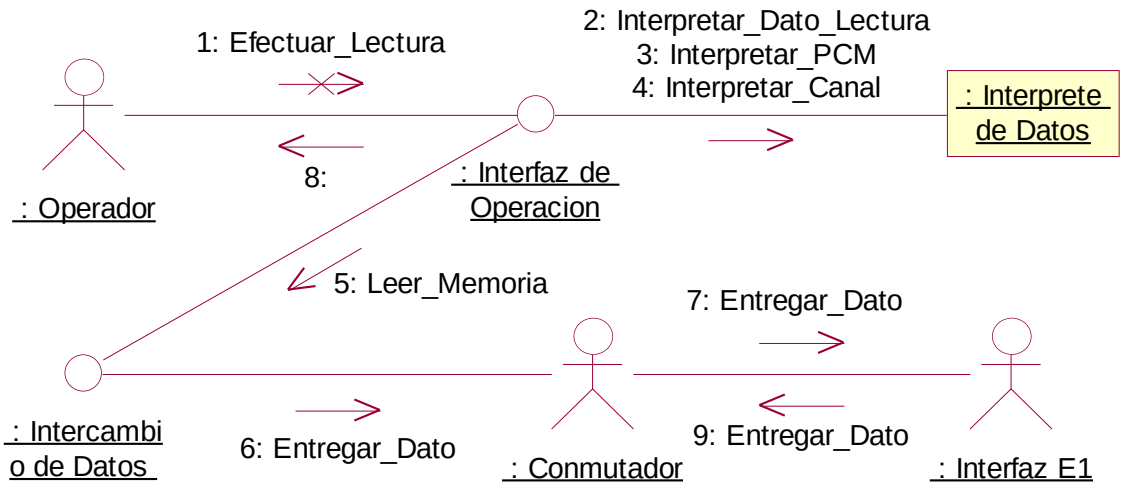
Flujos alternativos: Ninguno.

Excepciones:

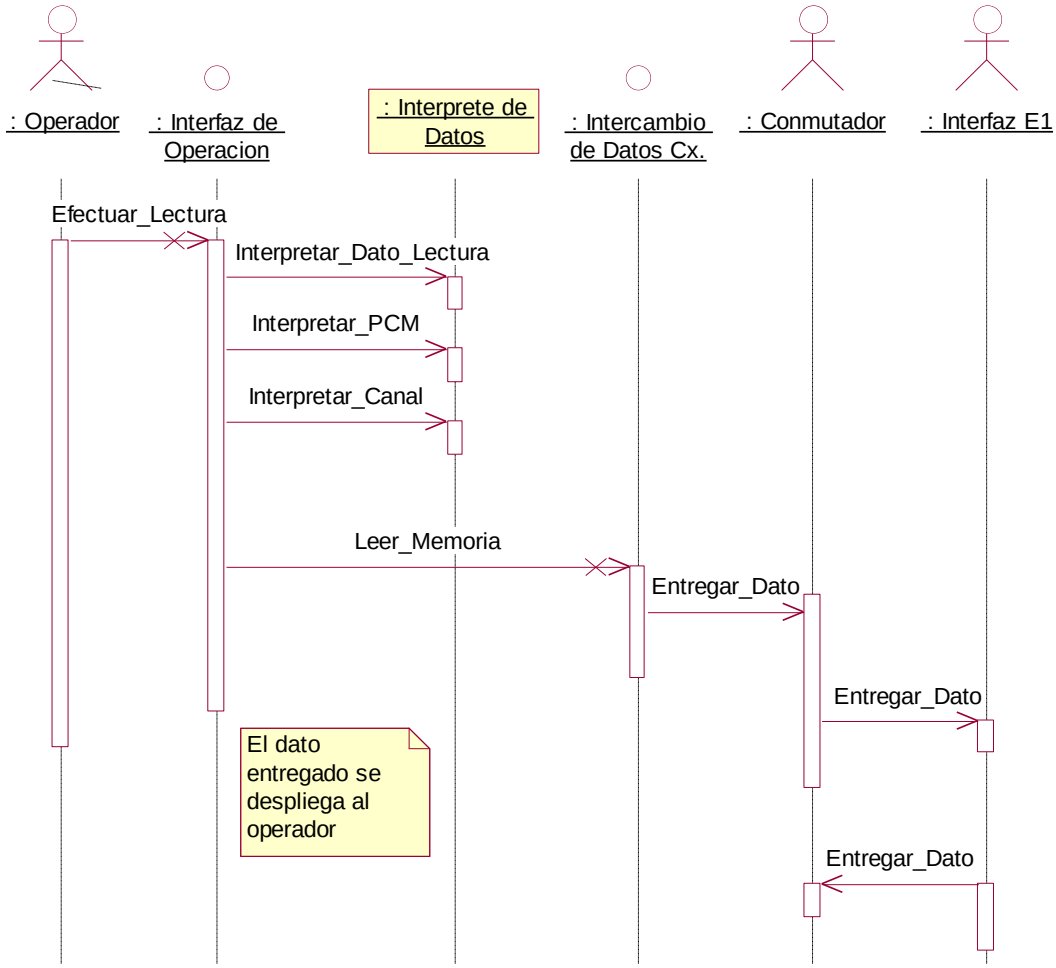
- Petición de información inexistente o inaccesible.
- Fallas de comunicación.

Recursos especiales: Ninguno.

2.3.2 DIAGRAMA DE COLABORACION (INTERACCION)



2.3.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



2.4 CASO DE USO ENRUTAR CANAL

2.4.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los PCMs e ITs de entrada y salida con los que se desea llevar a cabo el enrutamiento.
2. El sistema recoge los datos y los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la tarjeta.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para entregarle los datos de programación (suministrados por el operador) por medio de una serie de secuencias.
5. Implícitamente se encuentra, dentro de los datos entregados, el modo de trabajo del conmutador, el cual ocasiona el llamado de la función conmutar de este actor.

Postcondiciones:

- La tarjeta enruta el PCM e IT de entrada hacia el PCM e IT de salida suministrados por el operador y permanece en este estado de conmutación tipo TST de manera permanente.
- Operador con posibilidad de actualizar la información para llevar a cabo un nuevo enrutamiento o hacer uso de las demás funciones que ofrece el sistema.

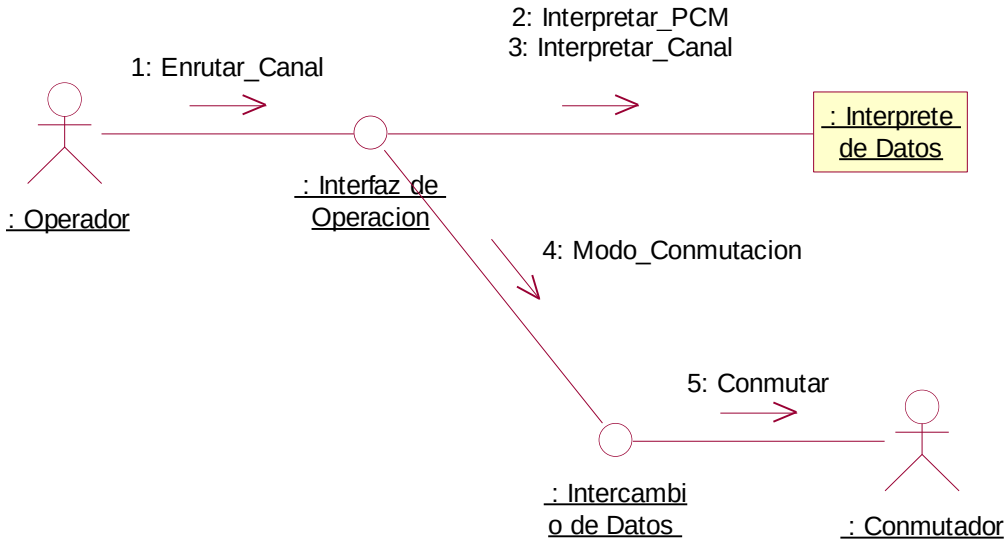
Flujos alternativos: Ninguno.

Excepciones:

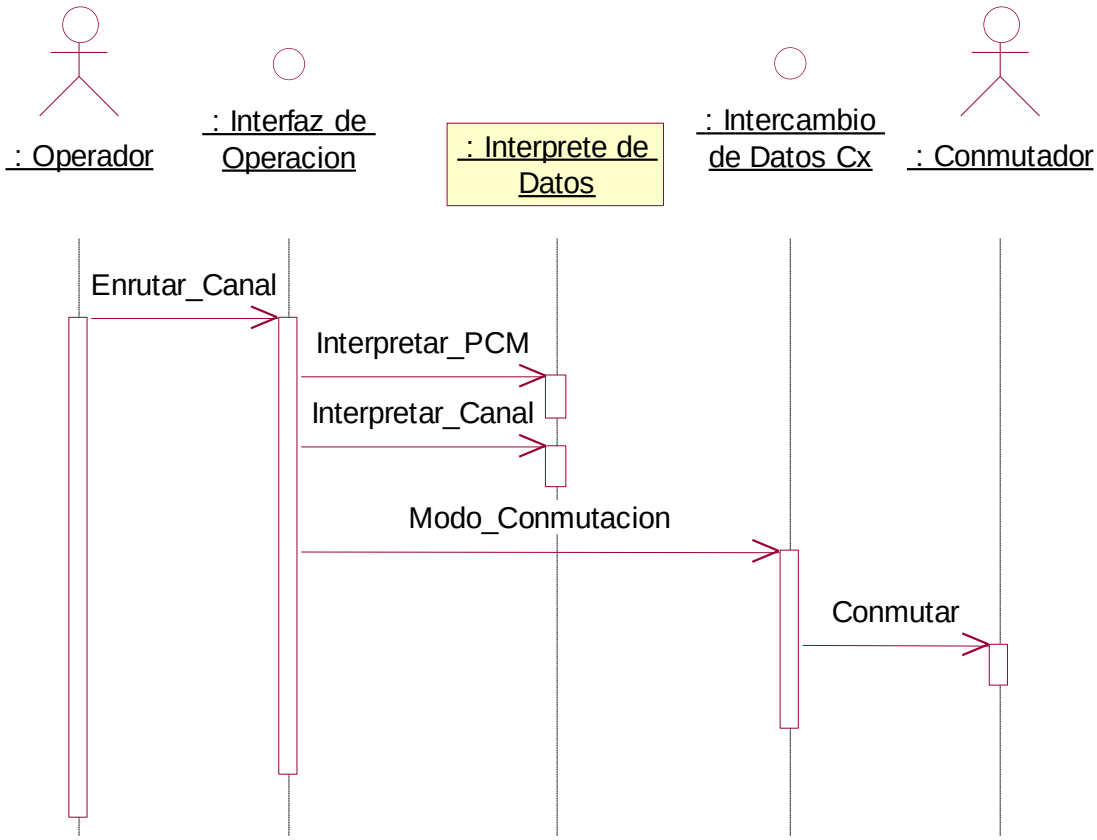
- Información fuera de rango.
- Fallas de comunicación.

Recursos especiales: Ninguno.

2.4.2 DIAGRAMA DE COLABORACION (INTERACCION)



2.4.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



2.5 CASO DE USO INTRODUCIR SX

2.5.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce el canal a señalizar y el valor de los bits de señalización.
2. El sistema recoge los datos y los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la tarjeta y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para entregarle los datos de configuración (suministrados por el operador) por medio de una serie de secuencias.
5. Implícitamente se encuentra, dentro de los datos entregados, el modo de trabajo del conmutador, el cual ocasiona el llamado de la función Enviar Mensaje de este actor.

Postcondiciones:

- La tarjeta ejecuta la inserción de los bits de señalización.
- Operador con posibilidad de actualizar la información de señalización o hacer uso de las demás funciones que ofrece el sistema.

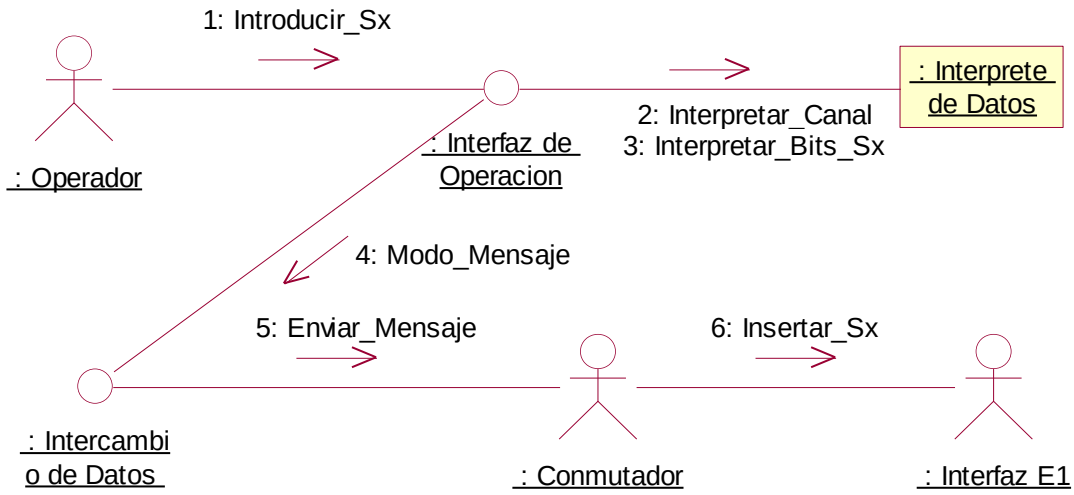
Flujos alternativos: Ninguno.

Excepciones:

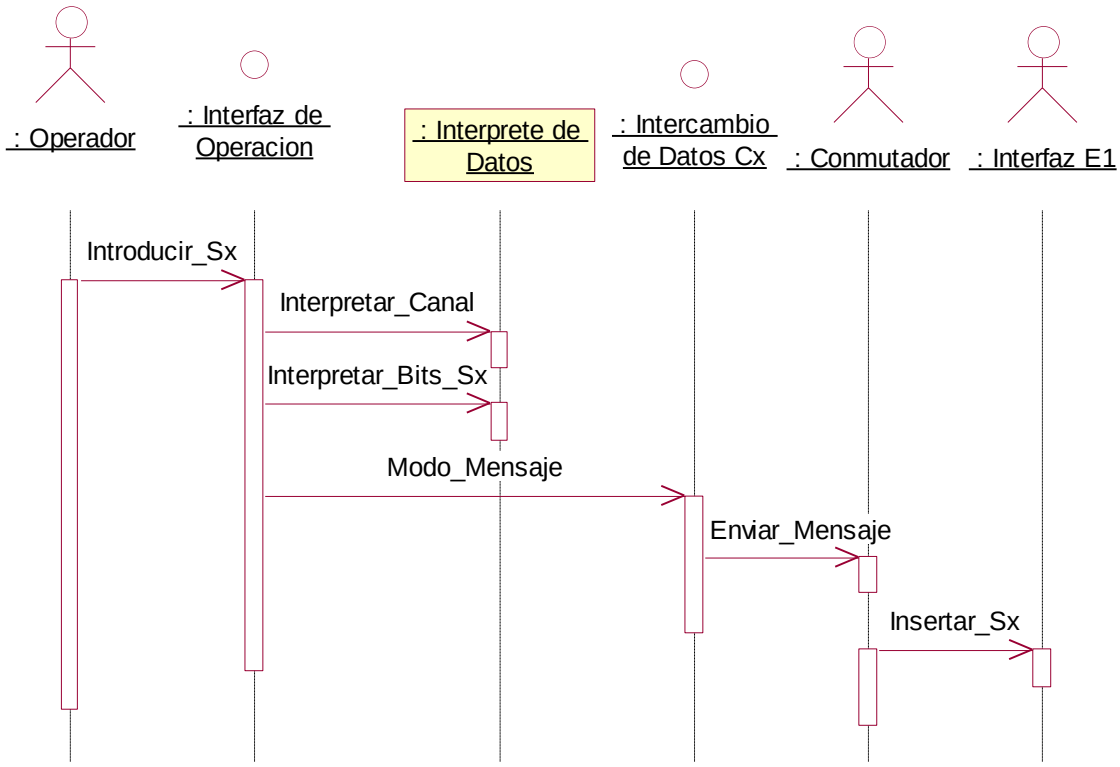
- Información fuera de rango.
- Fallas de comunicación.

Recursos especiales: Ninguno.

2.5.2 DIAGRAMA DE COLABORACION (INTERACCION)



2.5.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



2.6 CASO DE USO ACCEDER A HDLC

2.6.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los parámetros de acceso al dispositivo HDLC (Lectura/Escritura, Dirección, Dato).
2. El sistema coloca el dato en la dirección específica en caso de escritura o accede al dato en caso de lectura.
3. Se efectúa el llamado de la función Entregar Dato para facilitar el dato al operador en caso de lectura.

Postcondiciones: Operador con posibilidad de acceder de nuevo al HDLC o hacer uso de las demás funciones que ofrece el sistema.

Flujos alternativos: Ninguno.

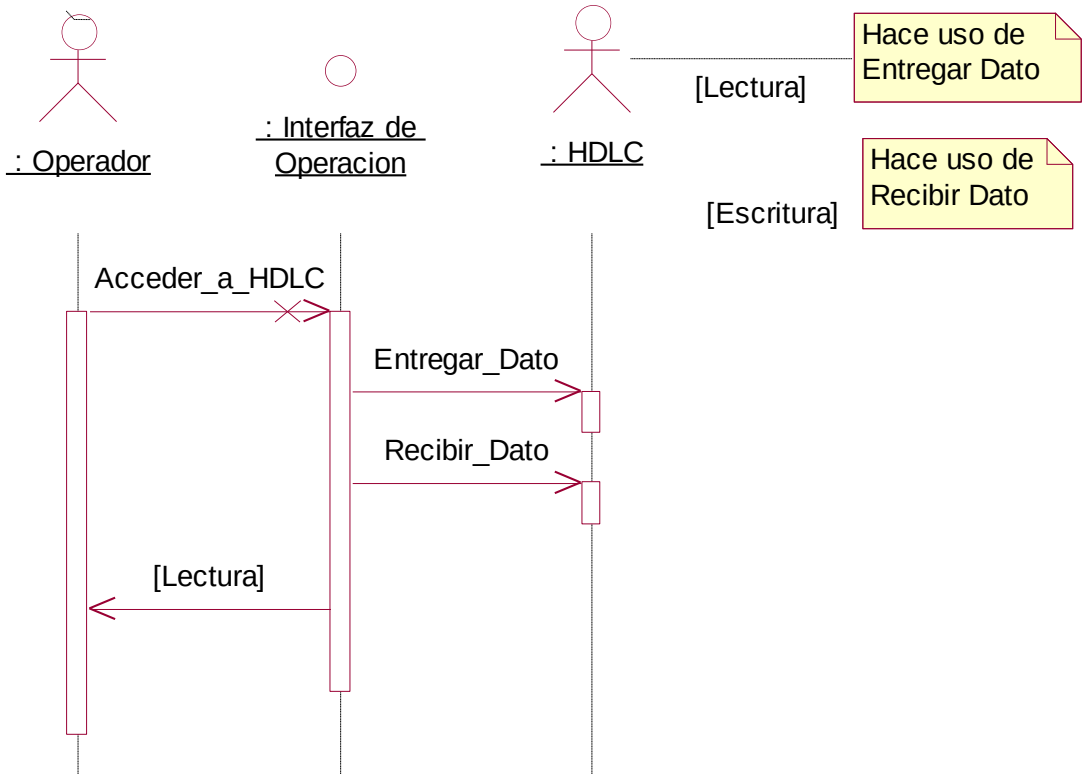
Excepciones:

- Información fuera de rango.
- Fallas de comunicación.

Recursos especiales: Ninguno.

2.6.2 DIAGRAMA DE COLABORACION (INTERACCION)

2.6.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



2.7 CASO DE USO ENTREGAR DATO

2.7.1 DESCRIPCION DE ESCENARIO

Iniciador: Conmutador, Interfaz E1 ó HDLC.

Precondición: Operador ha hecho uso de Efectuar Lectura o Acceder a HDLC.

Flujo de eventos:

1. Cualquiera de los iniciadores recibe la información referente al dato al cual se desea acceder.
2. Partiendo de esta información, Conmutador, Interfaz E1 ó HDLC entrega el dato solicitado.

Postcondiciones: Ninguna.

Flujos alternativos: Ninguno.

Excepciones: Ninguna.

Recursos especiales: Ninguno.

2.8 CASO DE USO ENVIAR MENSAJE

2.8.1 DESCRIPCION DE ESCENARIO

Iniciador: Conmutador.

Precondición: Operador ha hecho uso de Introducir Sx o Configurar Stma ó Inicializador ha hecho uso de Configurar Stma.

Flujo de eventos:

1. Conmutador recibe el PCM e IT de salida y el mensaje a enviar (datos de programación suministrados por el operador) por medio de una serie de secuencias.
2. Partiendo de estos datos, el conmutador coloca el mensaje en el PCM e IT de salida especificado por el operador.

Postcondiciones: El conmutador envía el mensaje deseado continuamente.

Flujos alternativos: Ninguno.

Excepciones: Ninguna.

Recursos especiales: Ninguno.

2.9 CASO DE USO CONMUTAR

2.9.1 DESCRIPCION DE ESCENARIO

Iniciador: Conmutador.

Precondición: Operador ha hecho uso de Enrutar canal.

Flujo de eventos:

1. Conmutador recibe los PCMs e ITs de entrada y salida (datos de programación suministrados por el operador) por medio de una serie de secuencias.

2. Partiendo de estos datos, el conmutador establece el camino de conexión entre el PCM e IT de entrada y el PCM e IT de salida llevando a cabo el enrutamiento solicitado.

Postcondiciones: El conmutador enruta el PCM e IT de entrada hacia el PCM e IT de salida suministrados por el operador y permanece en este estado de conmutación tipo TST de manera permanente.

Flujos alternativos: Ninguno.

Excepciones: Ninguna.

Recursos especiales: Ninguno.

2.10 CASO DE USO INSERTAR SX

2.10.1 DESCRIPCION DE ESCENARIO

Iniciador: Interfaz E1.

Precondición: Operador ha hecho uso de Introducir Sx.

Flujo de eventos:

1. Interfaz E1 recibe los bits de señalización de un canal específico a través de la función Enviar Mensaje del conmutador.
2. La interfaz E1 efectúa la inserción de estos bits teniendo en cuenta la disposición del entramado PCM.

Postcondiciones: La interfaz E1 continúa insertando la información de señalización actual de manera permanente.

Flujos alternativos: Ninguno.

Excepciones: Ninguna.

Recursos especiales: Ninguno.

2.11 CASO DE USO ADOPTAR CONF I-E1

2.11.1 DESCRIPCION DE ESCENARIO

Iniciador: Interfaz E1.

Precondición: Operador ó Inicializador ha hecho uso de Configurar Stma.

Flujo de eventos:

1. Interfaz E1 recibe la información de configuración especificada por el operador ó el inicializador.
2. La interfaz E1 asume dicha configuración.

Postcondiciones: La interfaz E1 queda configurada según las especificaciones deseadas.

Flujos alternativos: Ninguno.

Excepciones: Ninguna.

Recursos especiales: Ninguno.

2.12 CASO DE USO ADOPTAR CONF PLL

2.12.1 DESCRIPCION DE ESCENARIO

Iniciador: PLL.

Precondición: Operador ó Inicializador ha hecho uso de Configurar Stma.

Flujo de eventos:

1. PLL recibe la información de configuración especificada por el operador ó el inicializador.
2. PLL asume dicha configuración.

Postcondiciones: El PLL queda configurado según lo especificado.

Flujos alternativos: Ninguno.

Excepciones: Ninguna.

Recursos especiales: Ninguno.

2.13 CASO DE USO RECIBIR DATO HDLC

2.13.1 DESCRIPCION DE ESCENARIO

Iniciador: HDLC.

Precondición: Operador ha hecho uso de Acceder a HDLC para escritura.

Flujo de eventos:

1. El HDLC recibe el dato enviado por el operador a través de la puerta de acceso.
2. Esta información es adoptada e interpretada por el HDLC dependiendo de su contenido.

Postcondiciones: Ninguna.

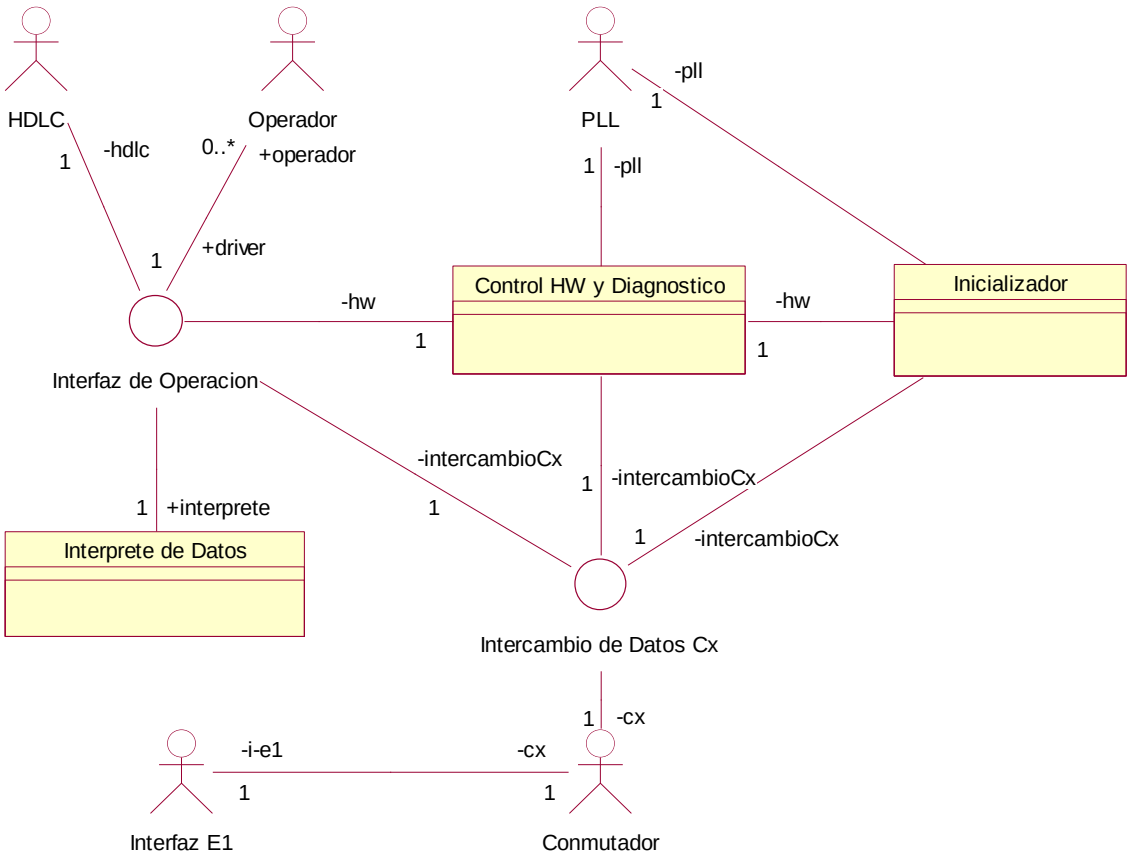
Flujos alternativos: Ninguno.

Excepciones: Ninguna.

Recursos especiales: Ninguno.

3 ANALISIS DE CLASES

3.1 DIAGRAMA DE CLASES DE ANALISIS



3.2 RESPONSABILIDADES

3.2.1 CLASE INTERFAZ DE OPERACION

Clase tipo **frontera**.

Servir de interfaz entre el operador y el sistema, proporcionando al primero las funciones básicas que ofrece este último, tales como configuración, enrutamiento

de canales, señalización, lecturas y diagnósticos; sin llegar a conocer la arquitectura real de la tarjeta.

3.2.2 CLASE INTERPRETE DE DATOS

Clase tipo **entidad**.

Convertir o interpretar la información suministrada por el operador en datos equivalentes comprensibles por el sistema, basado en la arquitectura de la tarjeta y el entramado PCM.

3.2.3 CLASE INTERCAMBIO DE DATOS CX

Clase tipo **frontera**.

Servir de interfaz entre el sistema y el actor conmutador llevando a cabo el handshake para establecer y mantener la comunicación que permita intercambiar datos entre ellos.

3.2.4 CLASE CONTROL HW Y DIAGNOSTICO

Clase tipo **control**.

Partiendo de que tipo de diagnóstico se va a llevar a cabo, realiza y manipula las pruebas de diagnóstico de los dispositivos de la tarjeta.

3.2.5 CLASE INICIALIZADOR

Clase tipo **entidad**.

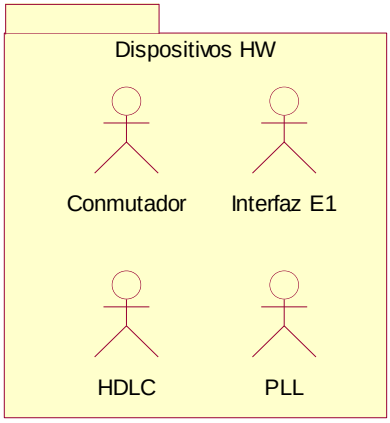
- Entregar al sistema la configuración inicial del mismo y de la tarjeta.
- Ordenar la ejecución de un diagnóstico periódico.

4 ANALISIS DE PAQUETES

4.1 DIAGRAMAS DE PAQUETES

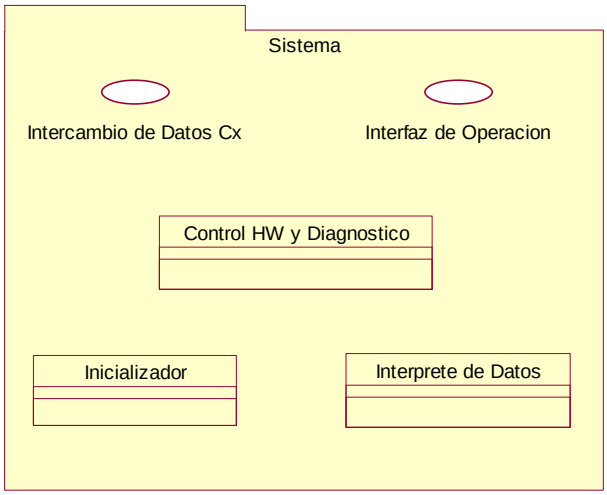
4.1.1 PAQUETE DISPOSITIVOS HW

Se refiere a los CIs de telecomunicaciones que no hacen parte del sistema de la TITD a implementar, pero están dentro de la tarjeta como dispositivos HW y son considerados por el sistema como actores ya que existe comunicación entre ellos.



4.1.2 PAQUETE SISTEMA

Este paquete hace alusión a las clases creadas que conforman el sistema de la TITD a desarrollar, distribuidas entre el PLD, el Microcontrolador y el Driver.



5 MANUAL DE USUARIO PRIMERA VERSION

En este manual se explica el manejo de la Tarjeta de Interfaz de Troncal Digital TITD en lo que se refiere a cómo es y de qué consta la arquitectura externa del sistema implementado sobre ella (entradas, salidas, direccionamiento E/S, dispositivos E/S); qué funciones básicas para comunicación ofrece y la manipulación de las mismas; cómo se configura; cómo y a qué nivel se programa; qué tipo de información se puede obtener de ella; en qué condiciones queda después del arranque; además de qué tipo y cómo se llevan a cabo sus diagnósticos, para que un desarrollador esté en la capacidad de elaborar una aplicación software que haga uso de la TITD.

El sistema implementado sobre la TITD está orientado para proveer el soporte físico mínimo que requiere un SSP (Punto de Conmutación de Servicio) de una red inteligente, ya que con él se puede realizar una conmutación digital básica, además de tener la capacidad de conectarse a centrales de telefonía pública de la PSTN y permitir el acceso a otros nodos de la RI.

5.1 ARQUITECTURA

5.1.1 ENTRADAS Y SALIDAS

El sistema TITD cuenta con:

- Una Troncal PCM E1 constituida por 2 líneas coaxiales, una para transmisión y otra para recepción, con el fin de permitir establecer un enlace distante con otra central. Para ello cuenta con su respectiva codificación de línea y multientramamiento PCM que involucra sincronismo y señalización. La información se encuentra ubicada sobre los 30 canales telefónicos, a 64 Kbps cada uno, con que dispone la troncal cuya rata de bit es 2048 Kbps.
- 5 líneas de entrada y 4 líneas de salida ST-BUS con características de PCM-32 a 2048Kbps, una trama cada 125 useg sin señalización, sincronismo ni

codificación de línea. Estas líneas tienen por objeto proveer la conexión con otros nodos de la red inteligente quienes realizan un tratamiento adicional a los datos enrutados hacia ellos.

5.1.2 DIRECCIONAMIENTO E/S

Una serie de selectores para escoger tanto la dirección E/S base (12 líneas MSB sin corrimiento) como la IRQ (de entre 11 posibles) de la tarjeta, con el fin de dar flexibilidad en su instalación para diferentes PCs cualquiera que sea su constitución.

5.1.3 DISPOSITIVOS E/S:

Una conexión al dispositivo HDLCPC (MT8952) conformada por varias líneas: Sistema-HDLC (Bus de direcciones, Bus de datos, línea ST-BUS), HDLC-Sistema (Bus de datos, línea ST-BUS, petición IRQ). Con el fin de permitir establecer una puerta de acceso con este dispositivo, el cual está en la capacidad de enviar y recibir paquetes de datos a través del protocolo HDLC.

Las figuras 1 y 2 (pág. 6) del capítulo 1 sirven de referencia para la arquitectura.

5.2 FUNCIONES BASICAS

Dentro de las funciones básicas que ofrece la TITD se encuentran:

- Enrutamiento de canales: Esta función hace referencia al encaminamiento de la información existente sobre los canales PCM correspondientes a la Troncal E1, las líneas de entrada y salida ST-BUS y las líneas ST-BUS del HDLCPC, lo cual implica que se permita establecer la interconexión entre dichos canales empleando para ello la conmutación TST característica de PCM.
- Manejo de señalización: Esta función hace referencia a la inserción y detección de los parámetros de señalización con respecto a la Troncal PCM E1, en la cual se involucra la posibilidad de Señalización por Canal Asociado SCA y Señalización por Canal Común SCC, a través del canal 16.

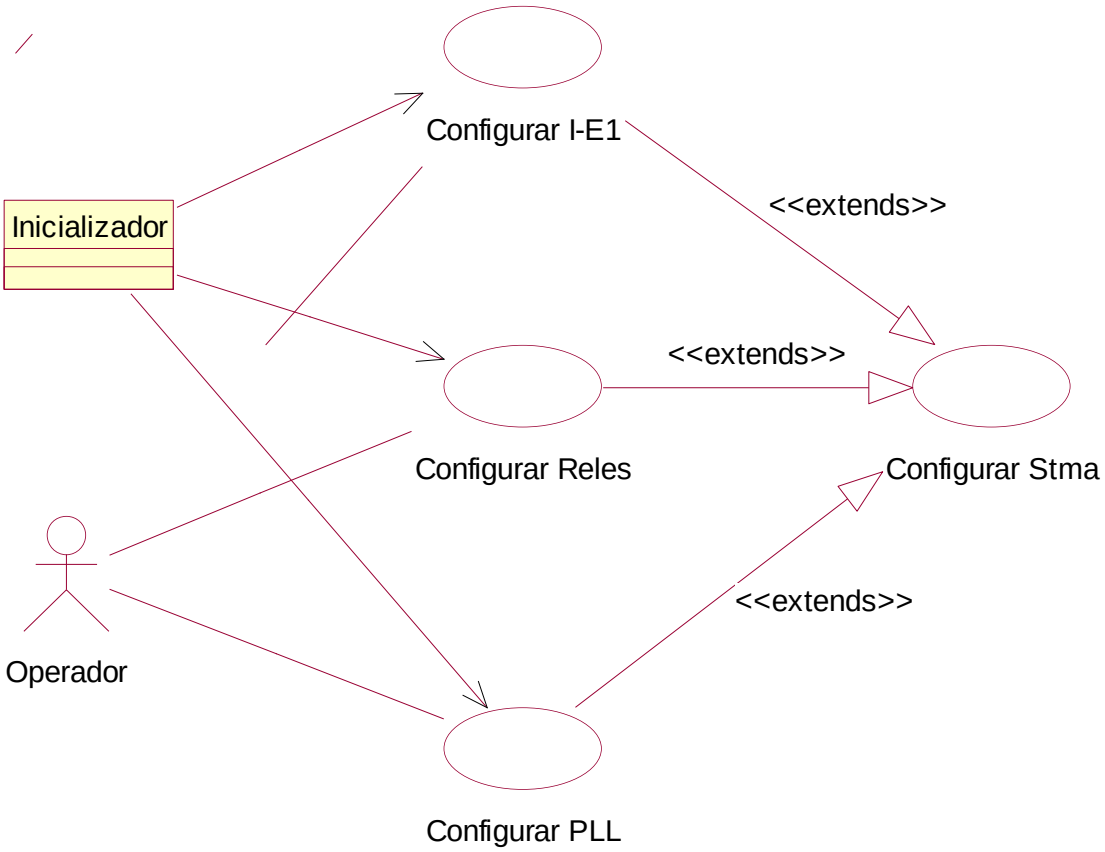
CAPITULO 2

CAPITULO 3
DISEÑO DEL SISTEMA

1 DISEÑO DE LOS CASOS DE USO

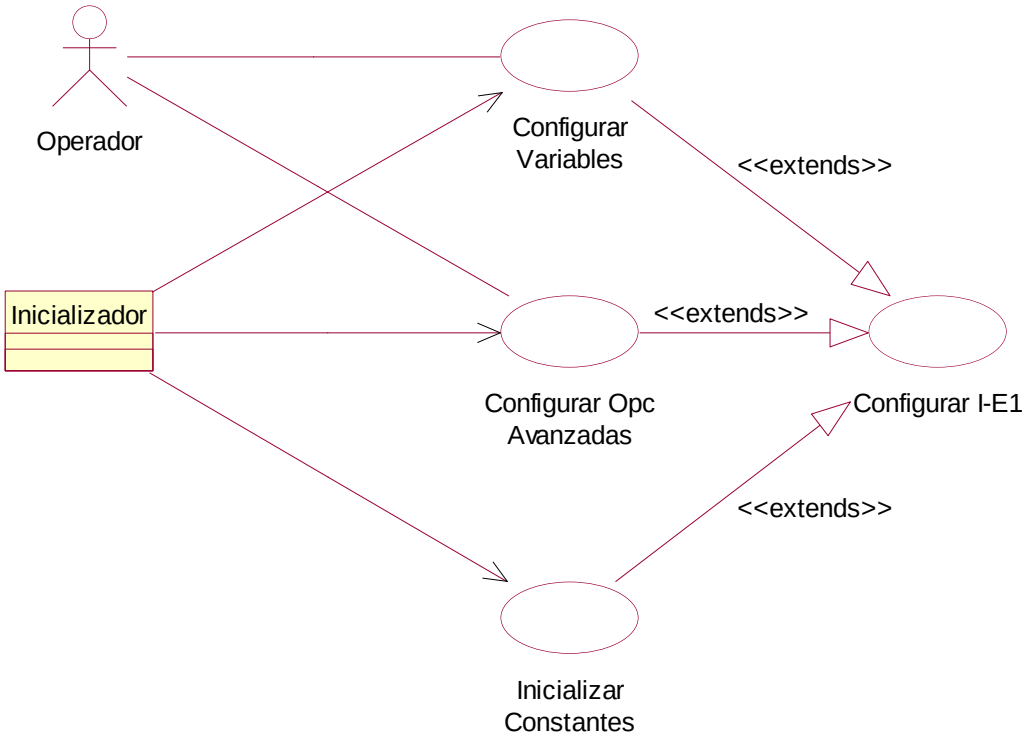
1.1 CASO DE USO CONFIGURAR STMA (NIVEL 0)

1.1.1 EXTENSION



1.2 CASO DE USO EXTENDIDO CONFIGURAR I-E1 (NIVEL 1)

1.2.1 EXTENSION



1.2.2 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador

Precondición: PLL configurado.

Flujo de eventos:

1. El Inicializador, conociendo la arquitectura y las características de entramado de la Interfaz E1, entrega los parámetros de la configuración inicial de la misma y va al flujo 4. Cuando el operador desea utilizar este caso de uso, introduce el nombre y el valor de los parámetros correspondientes a los datos de configuración de la Interfaz E1.
2. El sistema recoge los datos y los traduce a un formato entendible para él.

3. El sistema transforma los datos basándose en la arquitectura de la Interfaz E1 y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para entregarle los datos de configuración (suministrados por el inicializador ó por el operador) por medio de una serie de secuencias.

Postcondiciones:

- La Interfaz E1 asume la configuración.
- Operador con posibilidad de actualizar la configuración o hacer uso de las demás funciones que ofrece el sistema.

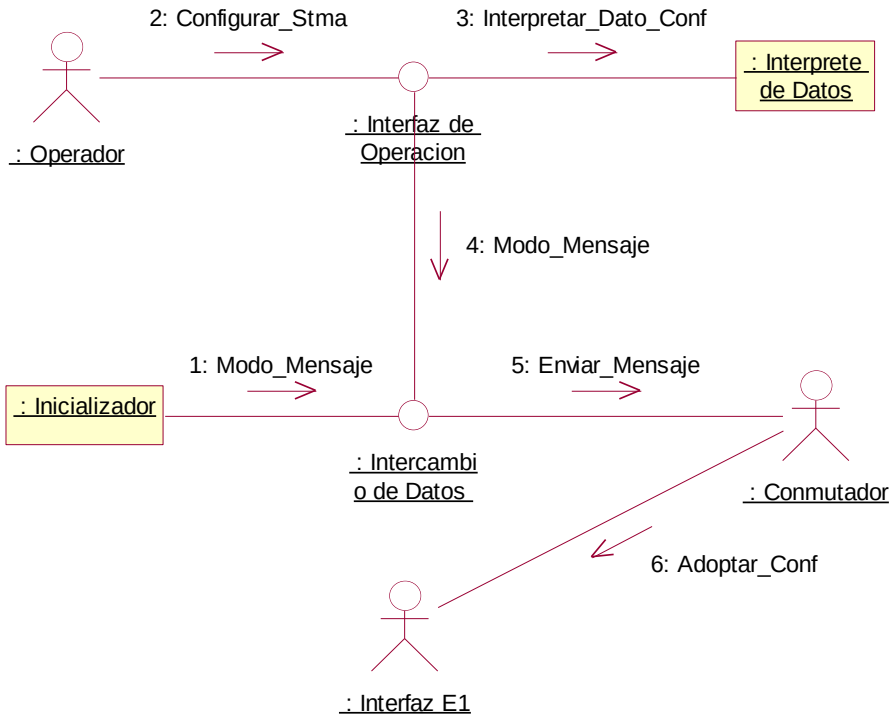
Flujos alternativos: Ninguno.

Excepciones:

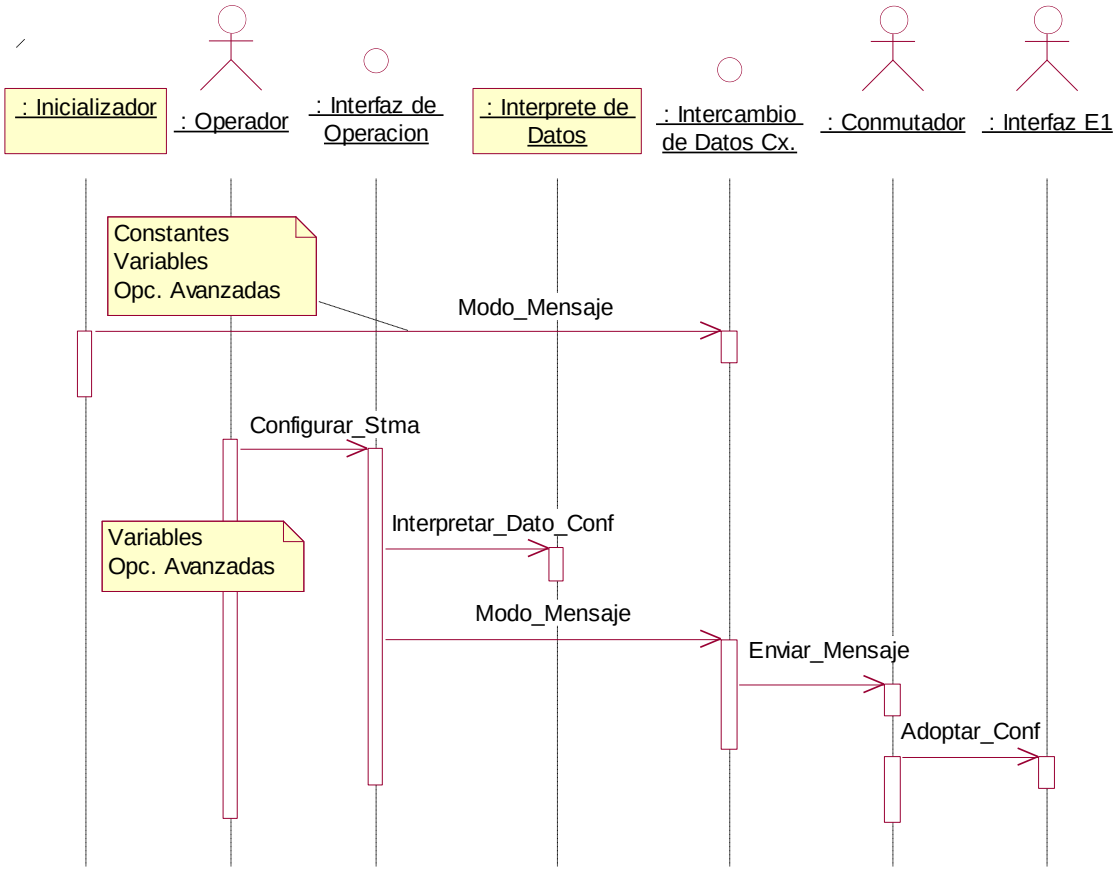
- Información fuera de rango por parte del operador.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.2.3 DIAGRAMA DE COLABORACION (INTERACCION)



1.2.4 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.3 CASO DE USO EXTENDIDO CONFIGURAR VARIABLES (NIVEL 2)

1.3.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador.

Precondición: PLL configurado.

Flujo de eventos:

1. El Inicializador, conociendo la arquitectura y las características de entramado de la Interfaz E1, entrega los parámetros referentes a las variables o información modificable de la configuración inicial de la misma y va al flujo 4. Cuando el operador desea utilizar este caso de uso, introduce el nombre y el valor de los

parámetros referentes a las variables o información modificable correspondientes a los datos de configuración de la Interfaz E1.

2. El sistema recoge los datos y los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la Interfaz E1 y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para entregarle los datos de configuración (suministrados por el inicializador ó por el operador) por medio de una serie de secuencias.

Postcondiciones:

- La Interfaz E1 asume la configuración.
- Operador con posibilidad de actualizar la configuración o hacer uso de las demás funciones que ofrece el sistema.

Flujos alternativos: Ninguno.

Excepciones:

- Información fuera de rango por parte del operador.
- Fallas de comunicación.

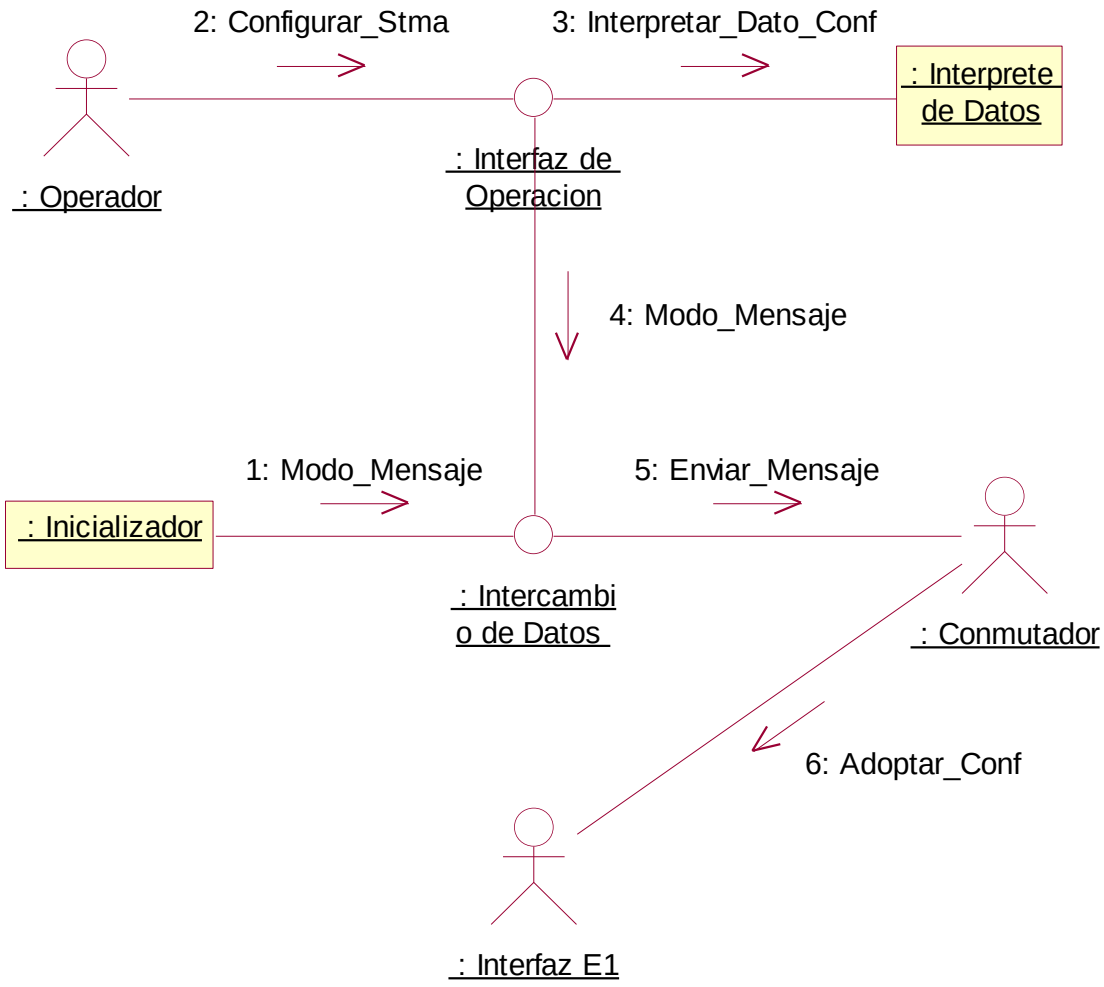
Recursos especiales: Ninguno.

Estos datos corresponden a:

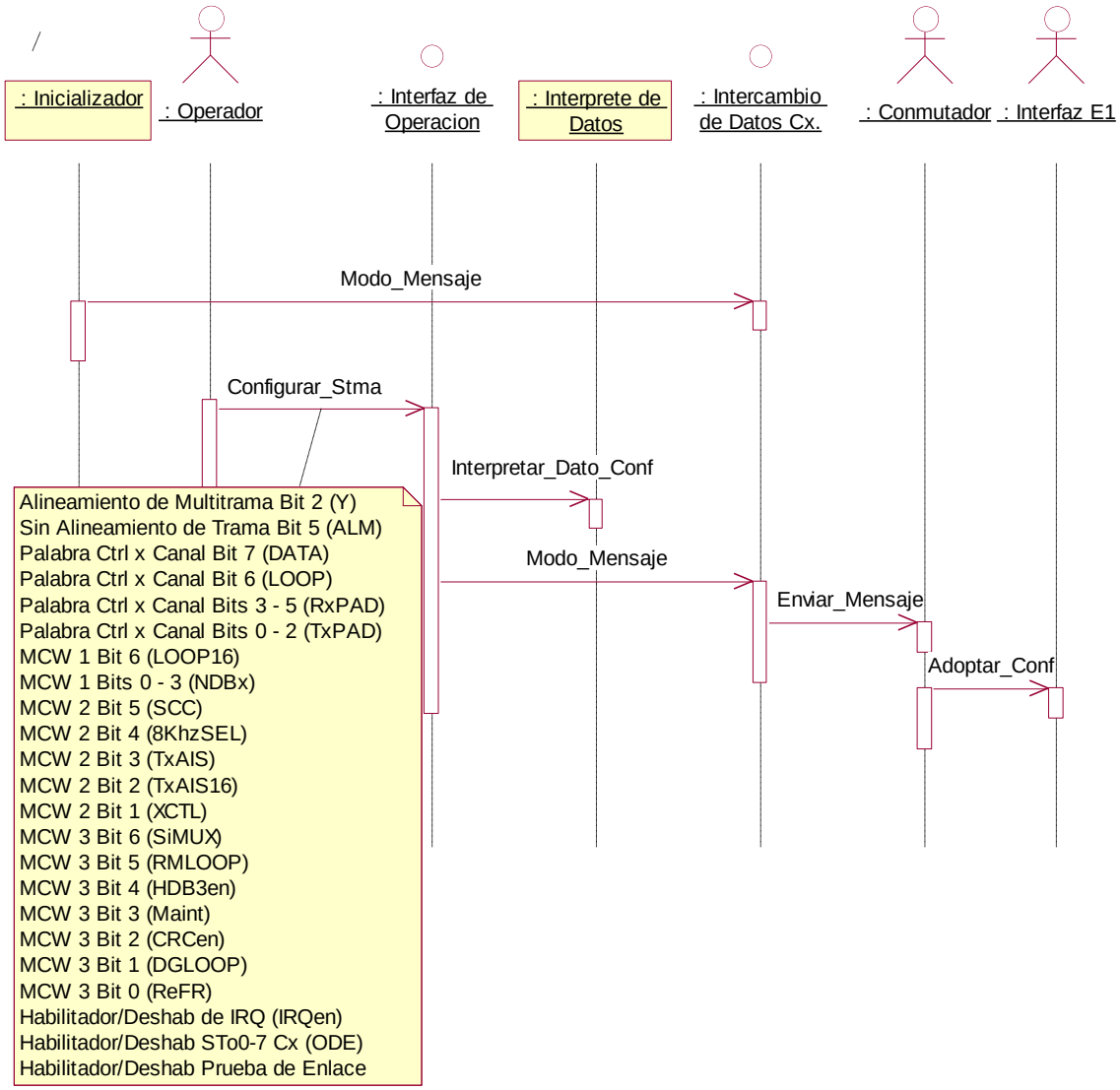
- Habilitador/Deshabilitador de Interrupción hacia el PC (IRQen).
- Habilitador/Deshabilitador PCMs de salida STo0-7 del Conmutador (ODE).
- Habilitador/Deshabilitador de la Prueba de Enlace en el diagnóstico periódico.
- El bit 2 (Y) indicador de pérdida de alineamiento de multitrama de la señal de alineamiento de multitrama.
- El bit 5 (A) indicador de alarma de la señal sin alineamiento de trama.
- El bit 7 (D) indicador de datos o voz, el bit 6 (LOOP) lazo por canal, los bits 3, 4 y 5 (RxPAD) atenuación de recepción y los bits 0, 1 y 2 (TxPAD) atenuación de transmisión, de la palabra de control por canal.
- El bit 6 (LOOP 16) lazo por IT 16 y los 4 bits menos significativos (NDBX) eliminación de rebote de los bits de señalización, de la palabra de control maestro 1 (MCW1).

- El bit 5 (SCC) señalización por canal común, el bit 4 (8KhzSEL) selector de 8 Khz, el bit 3 (ATx) alarma de transmisión por todos los ITs, el bit 2 (A16Tx) alarma de transmisión por el IT 16, y el bit 1 (XCTL) control externo, de la palabra de control maestro 2 (MCW2).
- El bit 6 (SiMUX) resultado CRC, el bit 5 (RMLOOP) lazo hacia la línea, el bit 4 (HDB3en) habilitador de codificación HDB3, el bit 3 (Maint) mantenimiento automático, el bit 2 (CRCen) habilitador de CRC, el bit 1 (DGLOOP) lazo digital y el bit 0 (ReFR) nueva trama, de la palabra de control maestro 3 (MCW3).

1.3.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.3.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.4 CASO DE USO EXTENDIDO CONFIGURAR OPC AVANZADAS (NIVEL 2)

1.4.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador.

Precondición: PLL configurado.

Flujo de eventos:

1. El Inicializador, conociendo la arquitectura y las características de entramado de la Interfaz E1, entrega los parámetros referentes a los bits de reserva de uso nacional e internacional de la configuración inicial de la misma y va al flujo 4. Cuando el operador desea utilizar este caso de uso, introduce el nombre y el valor de los parámetros referentes a los bits de reserva de uso nacional e internacional correspondientes a los datos de configuración de la Interfaz E1.
2. El sistema recoge los datos y los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la Interfaz E1 y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para entregarle los datos de configuración (suministrados por el inicializador ó por el operador) por medio de una serie de secuencias.

Postcondiciones:

- La Interfaz E1 asume la configuración.
- Operador con posibilidad de actualizar la configuración o hacer uso de las demás funciones que ofrece el sistema.

Flujos alternativos: Ninguno.

Excepciones:

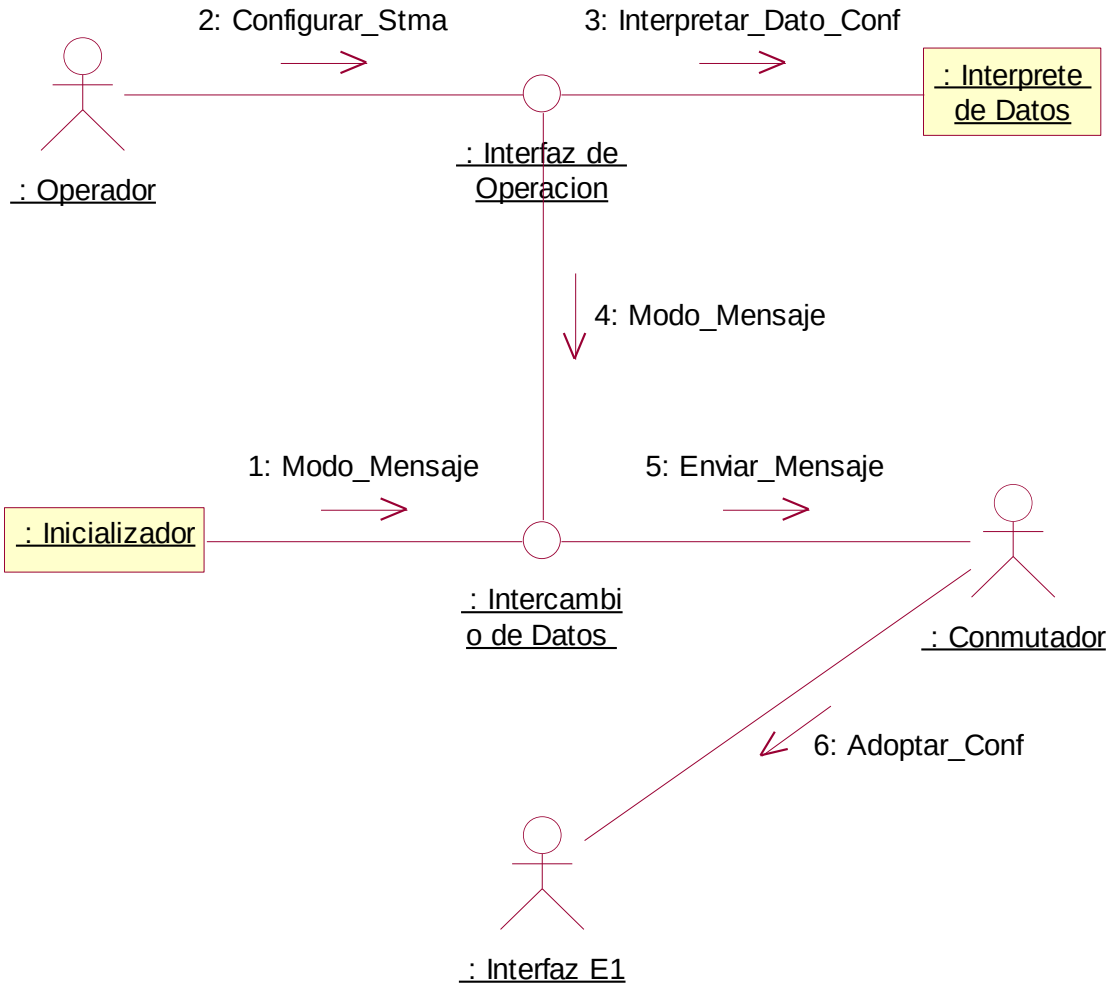
- Información fuera de rango por parte del operador.
- Fallas de comunicación.

Recursos especiales: Ninguno.

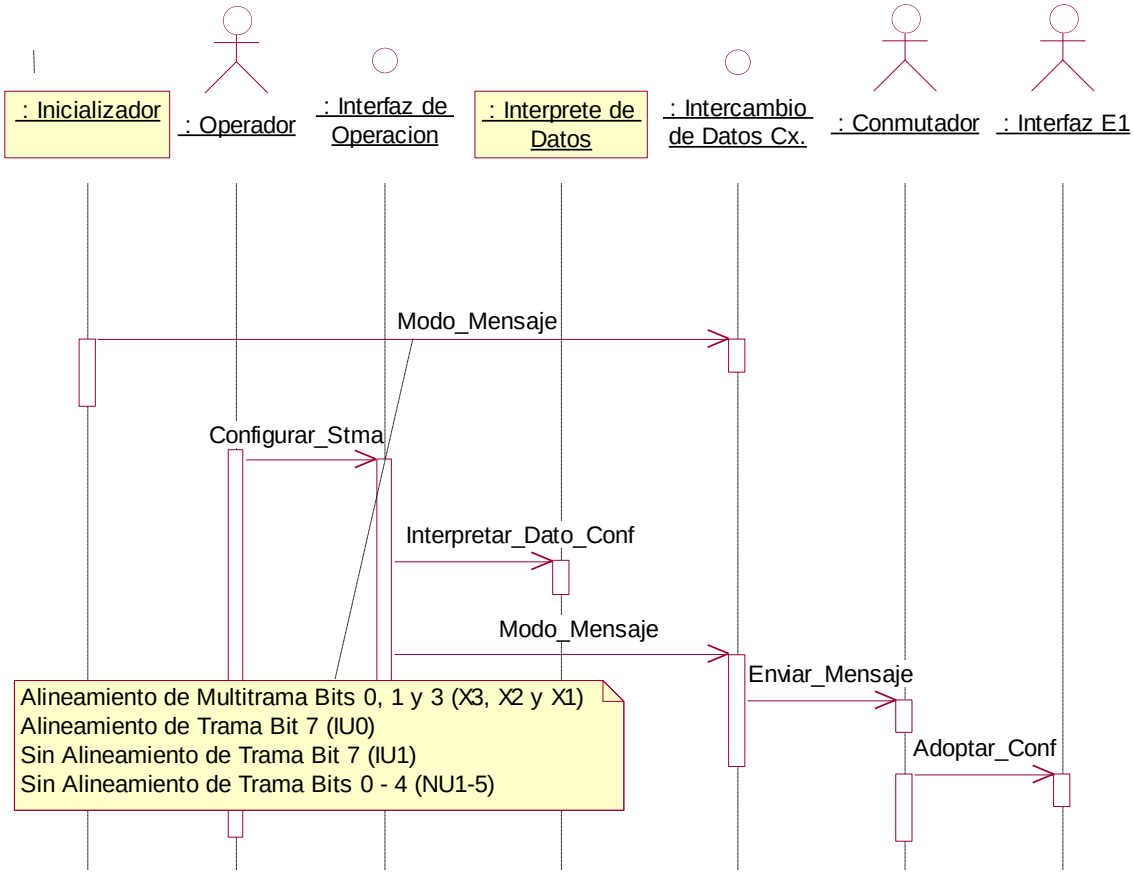
Estos datos corresponden a:

- Los bits 0, 1 y 3 (X3, X2 y X1) de la señal de alineamiento de multitrama.
- El bit 7 (IU0) de la señal de alineamiento de trama.
- El bit 7 (IU1) y los 5 bits menos significativos (NU 1-5) de la señal sin alineamiento de trama.

1.4.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.4.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.5 CASO DE USO EXTENDIDO INICIALIZAR CONSTANTES (NIVEL 2)

1.5.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador.

Precondición: PLL configurado.

Flujo de eventos:

1. El Inicializador, conociendo la arquitectura y las características de entramado de la Interfaz E1, entrega los parámetros referentes a las constantes o información no modificable de la configuración inicial de la misma.

2. El sistema se comunica con el conmutador, usando un protocolo específico, para entregarle los datos de configuración (suministrados por el inicializador) por medio de una serie de secuencias.

Postcondiciones: La Interfaz E1 asume la configuración.

Flujos alternativos: Ninguno.

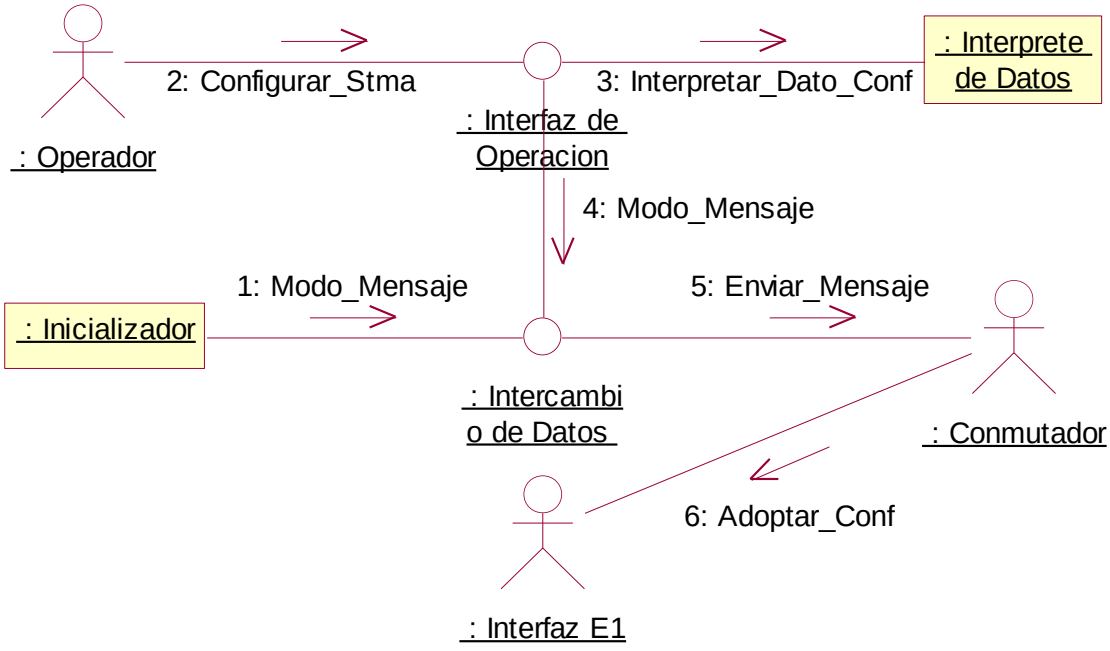
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

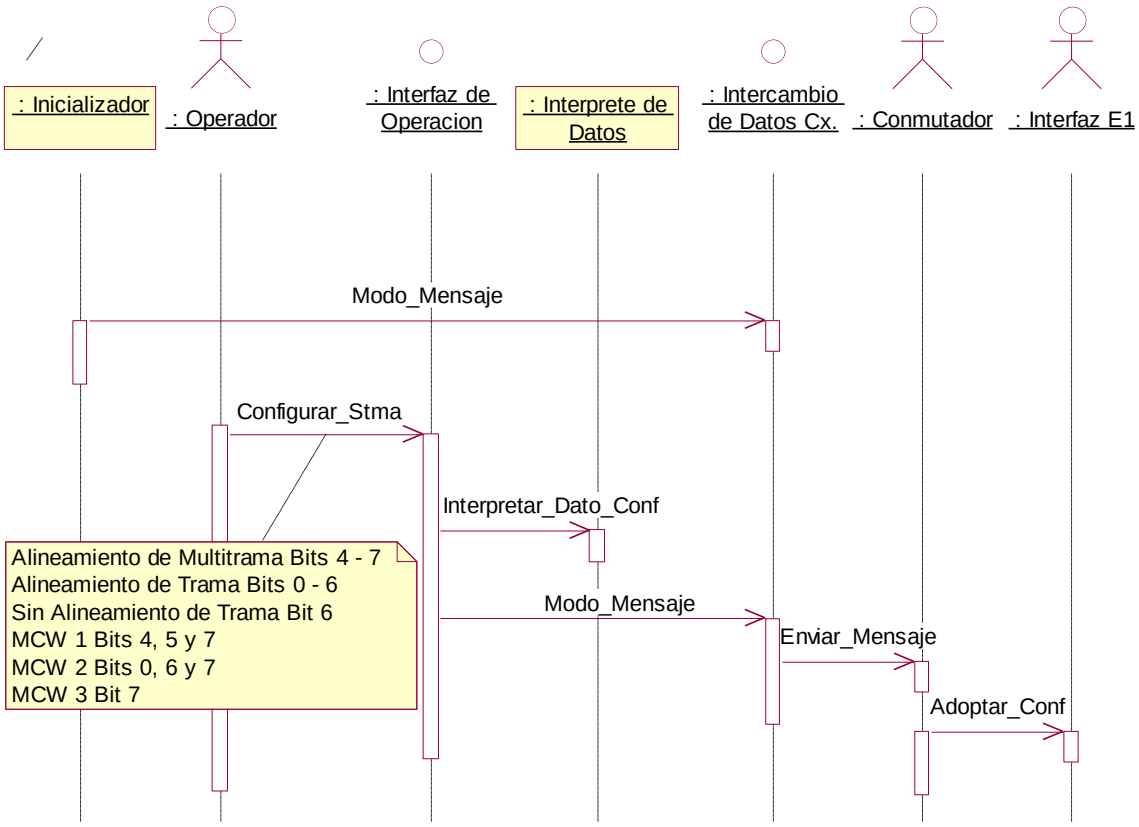
Estas constantes corresponden a:

- Los 4 bits más significativos (MA) de la señal de aliniamiento de multitrama.
- Los 7 bits menos significativos (FAF) de la señal de aliniamiento de trama.
- El bit 6 (NFAF) de la señal sin alineamiento de trama.
- Los bits 4, 5 y 7 de la palabra de control maestro 1 (MCW1).
- Los bits 0, 6 y 7 de la palabra de control maestro 2 (MCW2).
- El bit 7 (MSB) de la palabra de control maestro 3 (MCW3).

1.5.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.5.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.6 CASO DE USO EXTENDIDO CONFIGURAR RELES (NIVEL 1)

1.6.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador.

Precondición: Ninguna.

Flujo de eventos:

1. El Inicializador ordena la configuración inicial de los relés suministrando la información pertinente y va al flujo 3. Cuando el operador desea utilizar este caso de uso, introduce el valor del parámetro correspondiente al dato de configuración de los relés.
2. El sistema recoge el dato y lo traduce a un formato entendible para él.

3. El sistema configura los relés de acuerdo con la información suministrada por el inicializador ó por el operador.

Postcondiciones:

- Los relés asumen la configuración.
- Operador con posibilidad de actualizar la configuración o hacer uso de las demás funciones que ofrece el sistema.

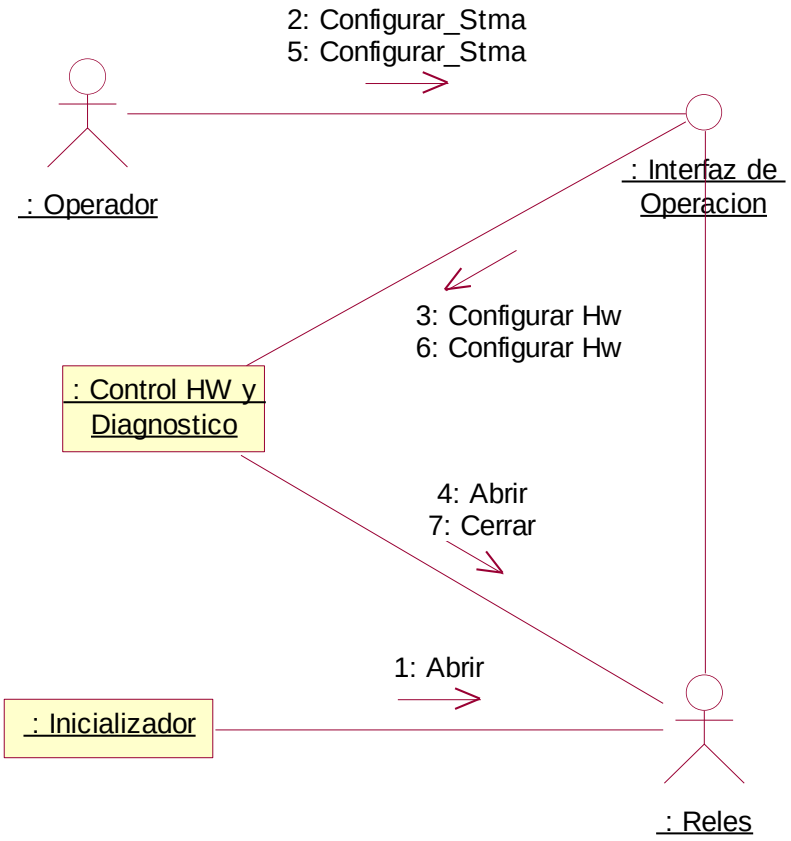
Flujos alternativos: Ninguno.

Excepciones:

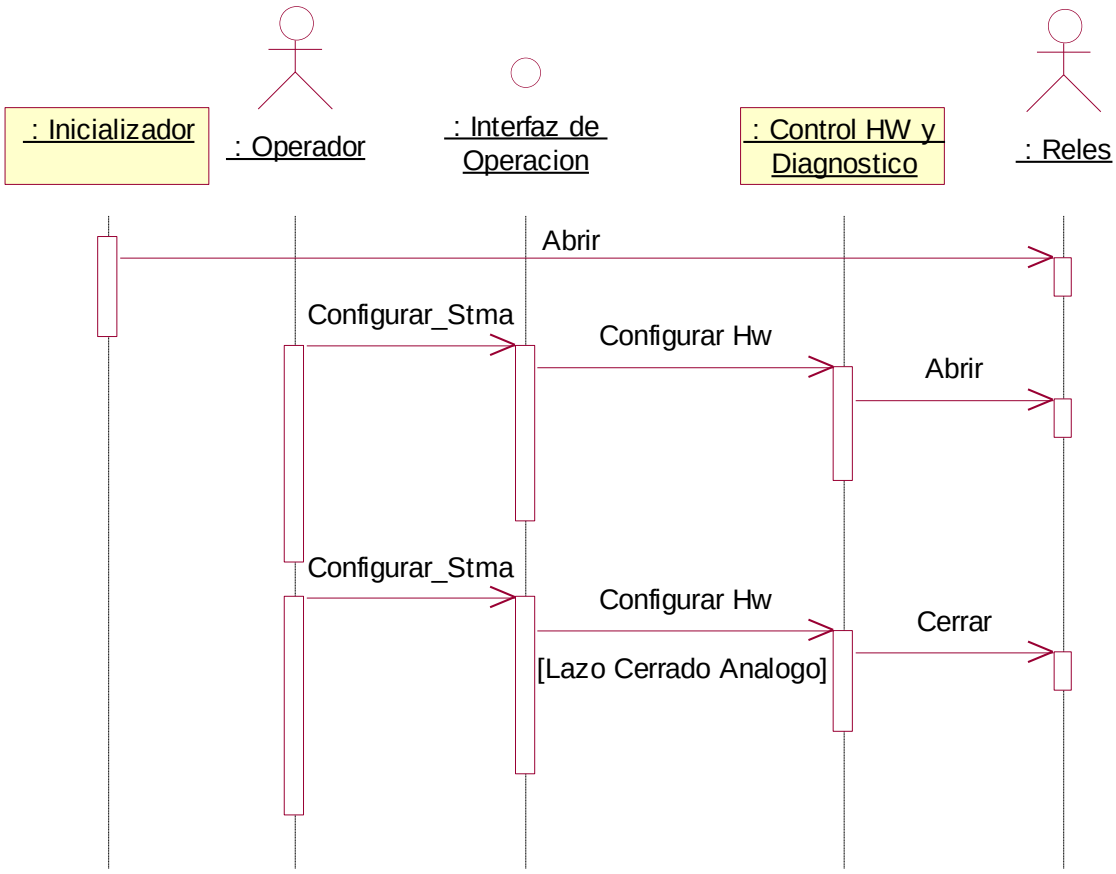
- Información fuera de rango por parte del operador.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.6.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.6.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.7 CASO DE USO EXTENDIDO CONFIGURAR PLL (NIVEL 1)

1.7.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador.

Precondición: Ninguna.

Flujo de eventos:

1. El Inicializador ordena la configuración inicial del PLL suministrando la información pertinente y va al flujo 3. Cuando el operador desea utilizar este caso de uso, introduce el valor del parámetro correspondiente al dato de configuración del PLL.

2. El sistema recoge el dato y lo traduce a un formato entendible para él.
3. El sistema configura el PLL de acuerdo con la información suministrada por el inicializador ó por el operador.

Postcondiciones:

- El PLL asume la configuración.
- Operador con posibilidad de actualizar la configuración o hacer uso de las demás funciones que ofrece el sistema.

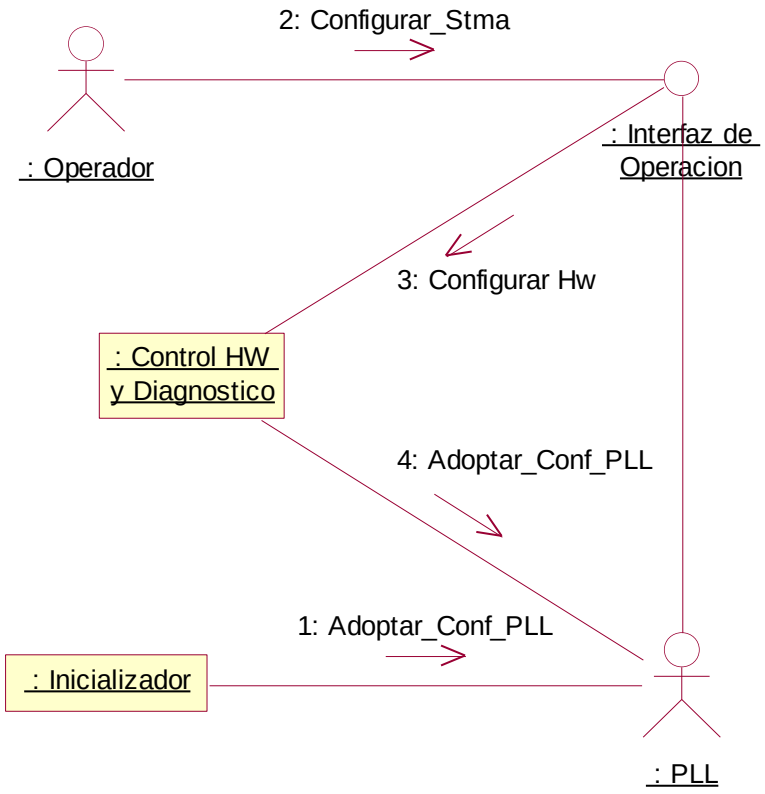
Flujos alternativos: Ninguno.

Excepciones:

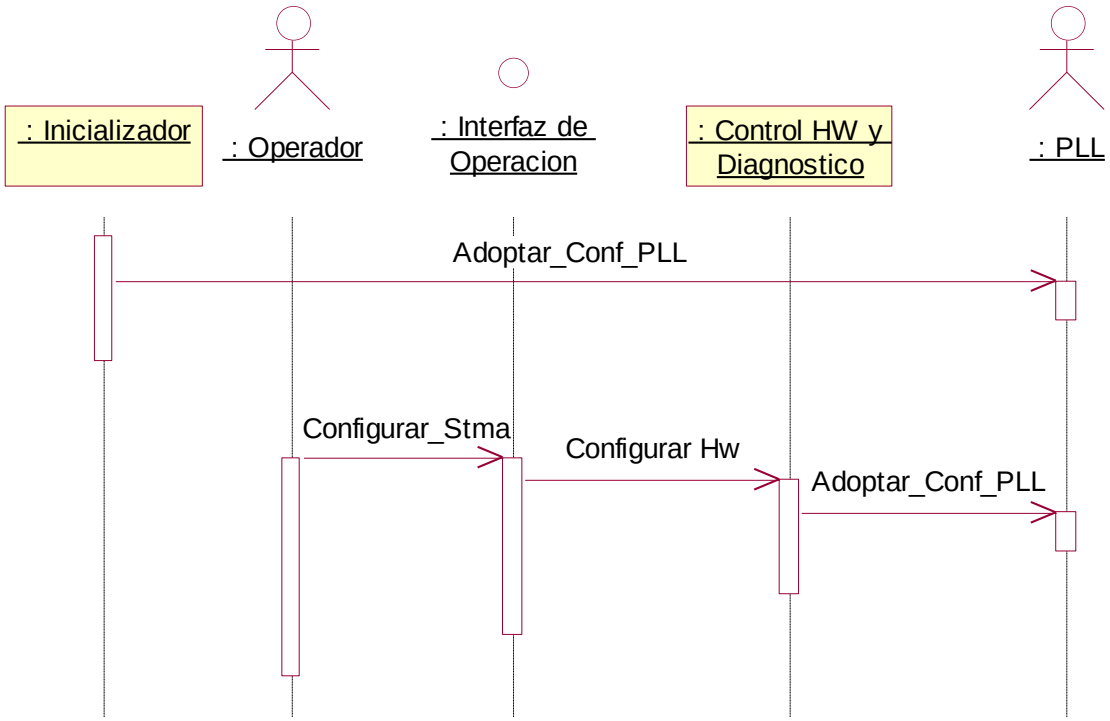
- Información fuera de rango por parte del operador.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.7.2 DIAGRAMA DE COLABORACION (INTERACCION)

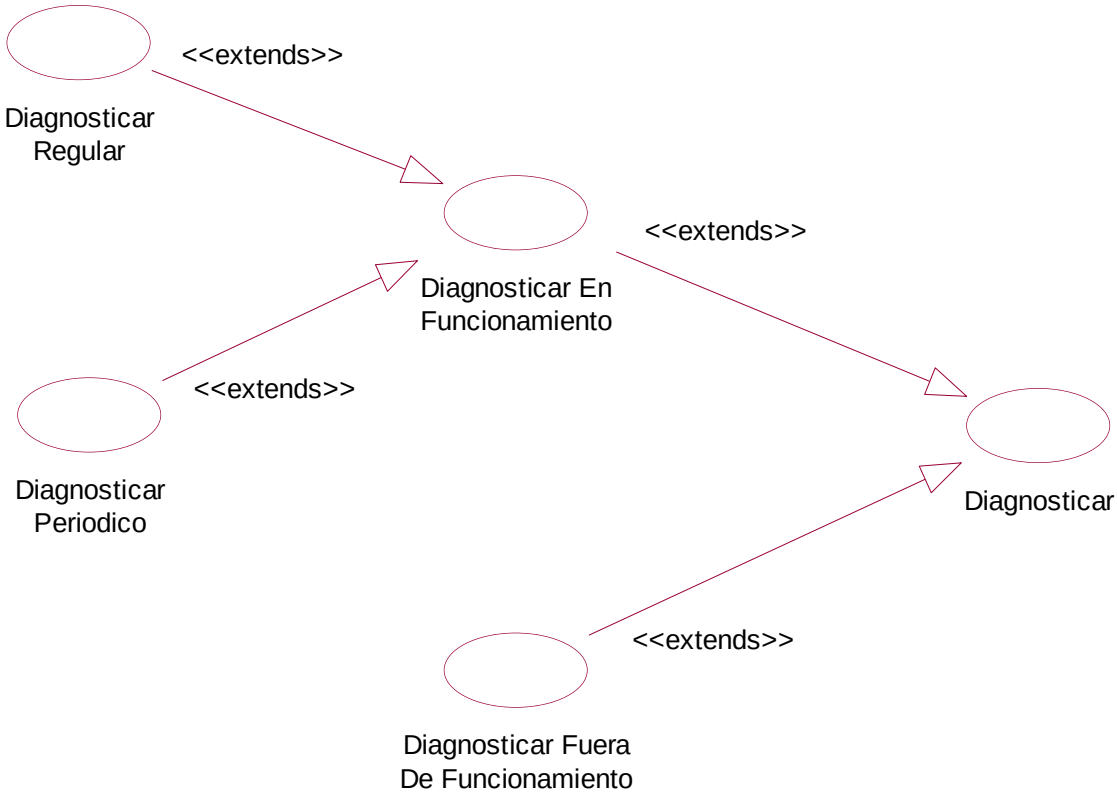


1.7.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.8 CASO DE USO DIAGNOSTICAR (NIVEL 0)

1.8.1 EXTENSION



1.9 CASO DE USO EXTENDIDO DIAGNOSTICAR EN FUNCIONAMIENTO (NIVEL 1)

1.9.1 EXTENSION

Se extiende en los casos de uso Diagnosticar Regular y Diagnosticar Periódico. Refiérase a estos para ver más detalles de este caso de uso.

1.10 CASO DE USO EXTENDIDO DIAGNOSTICAR REGULAR (NIVEL 2)

1.10.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ejecución de la configuración inicial de la tarjeta.

Flujo de eventos:

1. El operador da la orden de proceder con el diagnóstico e indica el canal en el se va a llevar a cabo la prueba.
2. El sistema ejecuta un lazo cerrado sólo en el IT correspondiente al canal deseado.
3. El sistema envía un mensaje en el IT mencionado.
4. El sistema efectúa la lectura de las memorias baja y de datos comparando sus contenidos con el mensaje mencionado en el paso 3 a fin de verificar su equivalencia.
5. El sistema efectúa la apertura del lazo referido al canal deseado.
6. De acuerdo al resultado de la comparación realizada en el paso 4, el sistema entrega el resultado que arroja la prueba, al operador.

Postcondiciones: Operador con posibilidad de ordenar la ejecución de otro diagnóstico o hacer uso de las demás funciones que ofrece el sistema.

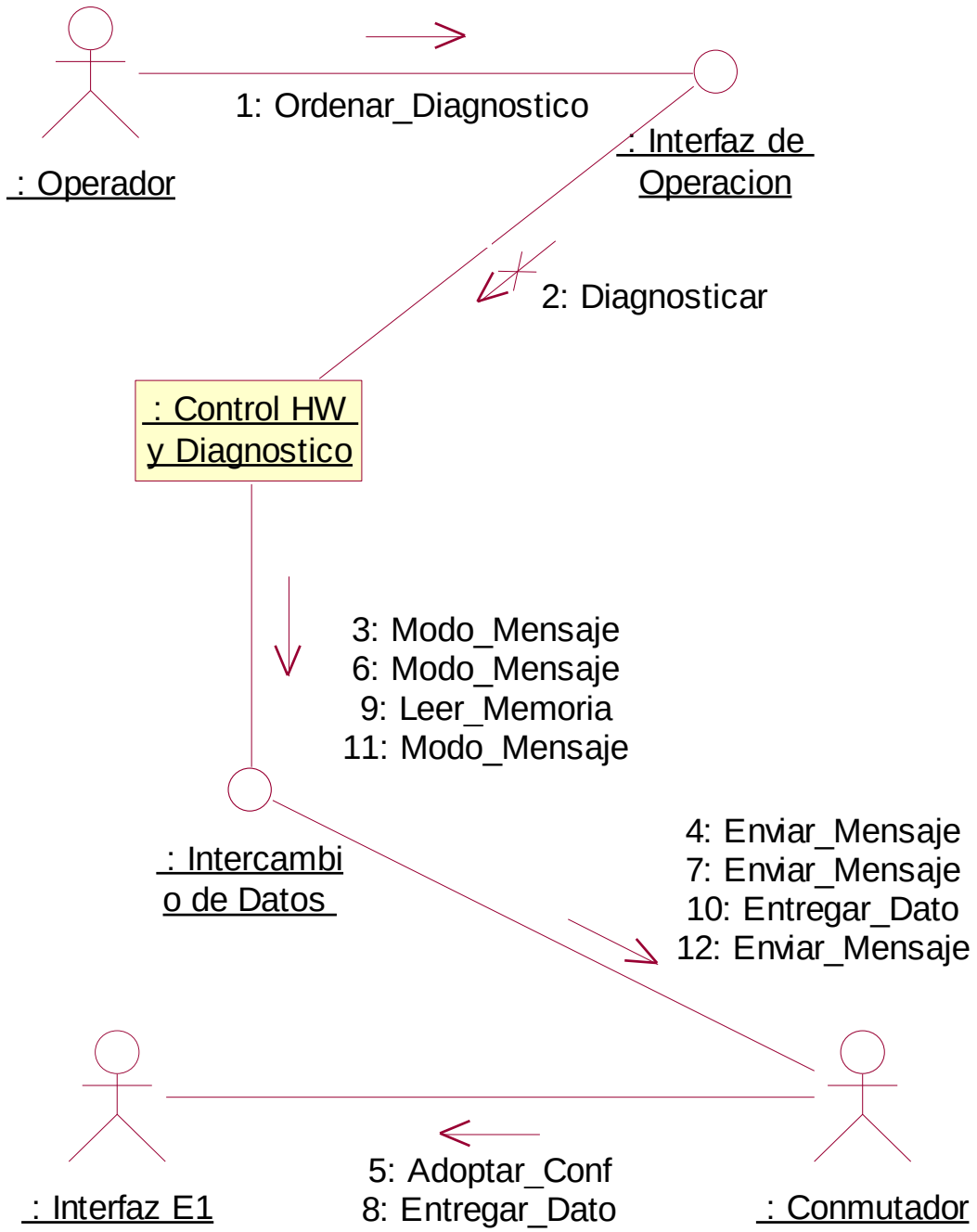
Flujos alternativos: Ninguno.

Excepciones:

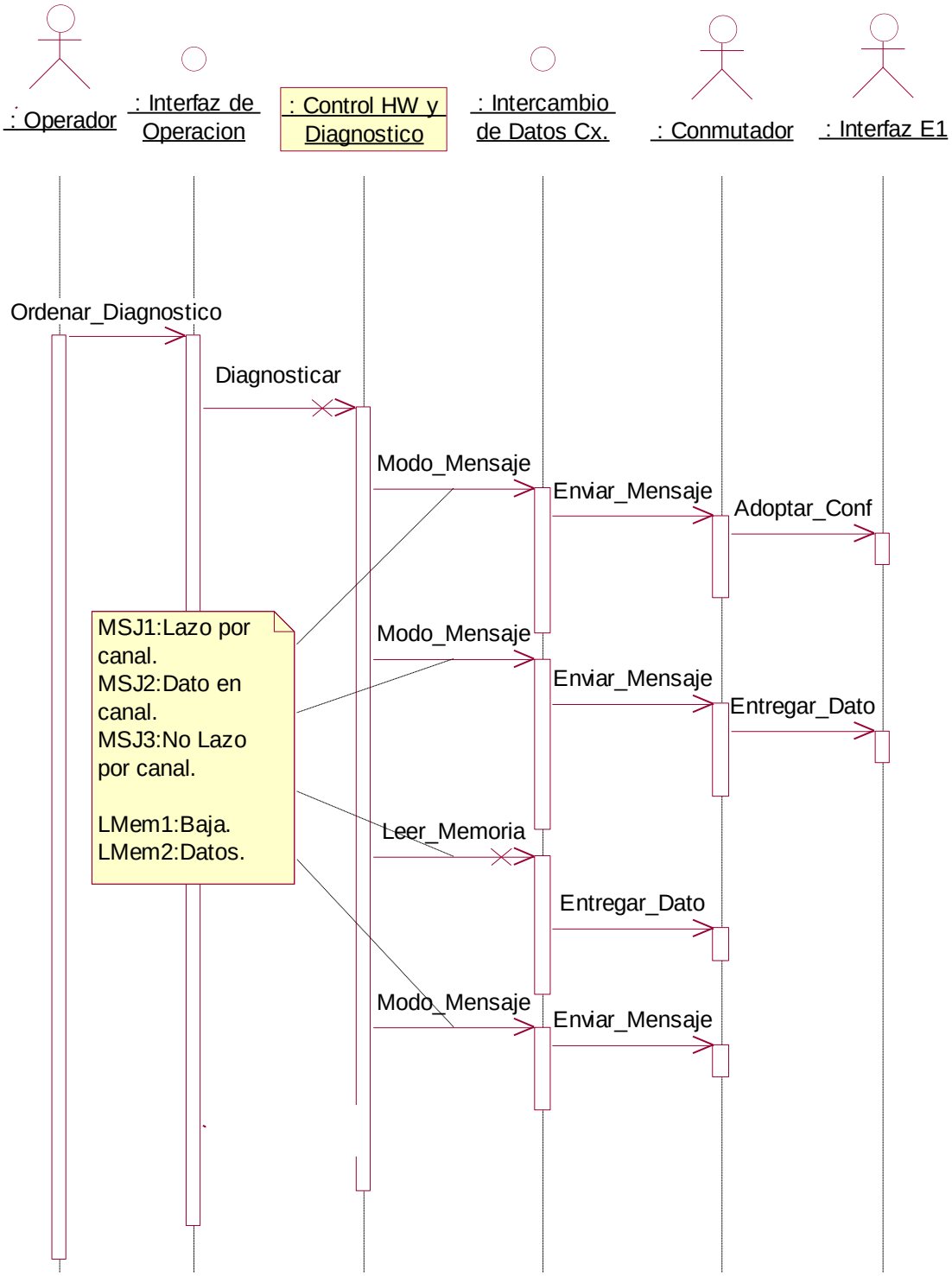
- Información suministrada por el operador fuera de rango.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.10.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.10.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.11 CASO DE USO EXTENDIDO DIAGNOSTICAR PERIODICO (NIVEL 2)

1.11.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador.

Precondiciones:

- Ejecución de la configuración inicial de la tarjeta.
- Finalización del conteo del temporizador/contador del diagnóstico.

Flujo de eventos:

1. Inicializador da la orden de proceder con el diagnóstico.
2. El sistema efectúa la lectura de la memoria baja de conexión con el fin de comparar y verificar que las constantes de programación de la interfaz E1 son correctas y se están enviando adecuadamente.
3. Si dentro de la configuración del sistema se encuentra habilitada la prueba del enlace, el sistema efectúa la lectura de la memoria de datos con el fin de revisar los parámetros de estado relacionados con el enlace comprobando su funcionamiento apropiado.
4. De acuerdo al resultado de la comparación realizada en los pasos anteriores, el sistema genera una interrupción hacia el PC en el caso en que alguna de las pruebas haya sido fallida.
5. El sistema atiende la interrupción e informa al operador del suceso ocurrido.

Postcondiciones:

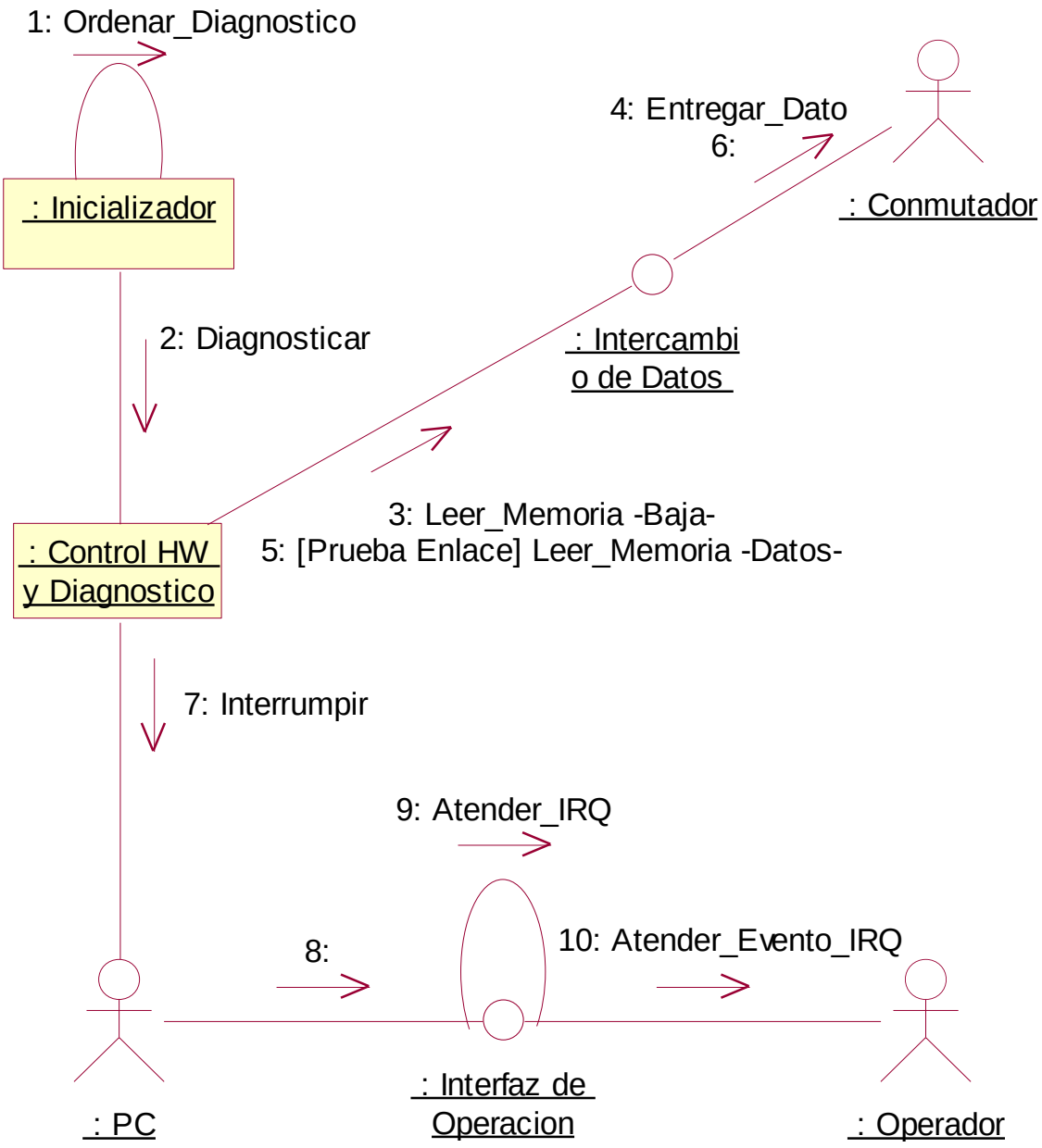
- El operador tiene la posibilidad de atender el evento ocasionado por la interrupción.
- Se reinicia el conteo del temporizador/contador del diagnóstico.

Flujos alternativos: Ninguno.

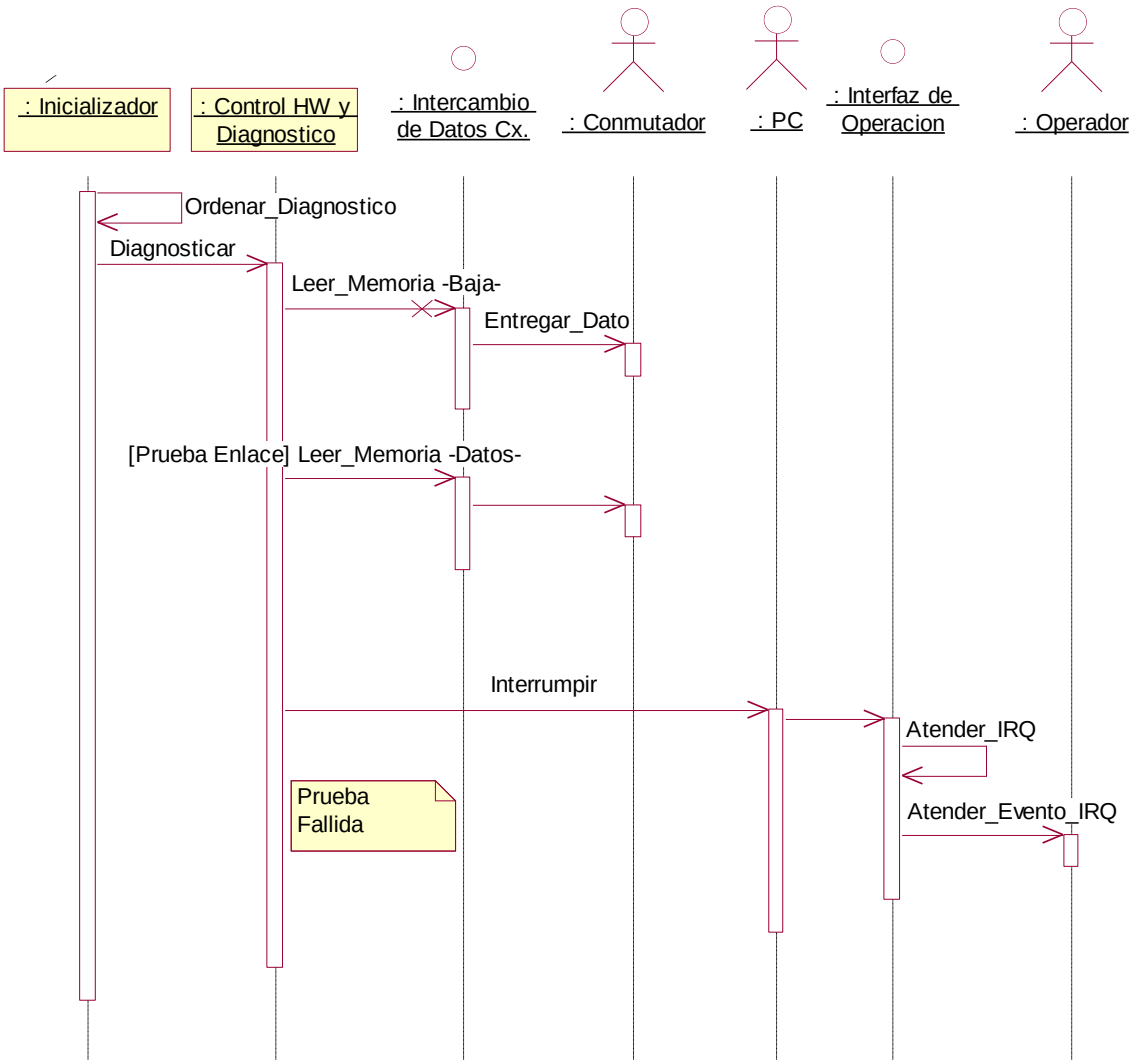
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

1.11.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.11.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.12 CASO DE USO EXTENDIDO DIAGNOSTICAR FUERA DE FUNCIONAMIENTO (NIVEL 1)

1.12.1 DESCRIPCION DE ESCENARIO

Iniciador: Inicializador.

Precondición: Ejecución de la configuración inicial de la tarjeta.

Flujo de eventos:

1. Una vez terminado el proceso de arranque de la tarjeta el Inicializador da la orden de proceder con el diagnóstico. El Operador puede ordenar que se lleve a cabo la prueba, cuando lo desee, teniendo en cuenta que la tarjeta queda fuera de funcionamiento.
2. El sistema corre la rutina del POST que se encarga de la prueba del dispositivo microcontrolador de la tarjeta.
3. El sistema configura el PLL en modo maestro.
4. El sistema ejecuta un lazo cerrado sólo en el IT 16 (canal de Sx), envía un mensaje en este IT y verifica que se reciba el mismo que ha sido enviado.
5. El sistema ejecuta un lazo cerrado, envía un mensaje y verifica que se reciba el mismo que ha sido enviado, para cada uno de los ITs.
6. El sistema ejecuta un lazo cerrado digital total, envía un mensaje en el primer IT, este se conmuta con el IT siguiente, y así sucesivamente; finalmente se verifica que se reciba el mensaje (en el último IT) que ha sido enviado (en el primer IT).
7. El sistema ejecuta el lazo cerrado análogo, envía un mensaje en un IT con su respectiva señalización y se verifica que se reciba el mensaje que ha sido enviado en dicho IT con su respectiva señalización en el canal y trama correspondientes.

Postcondiciones:

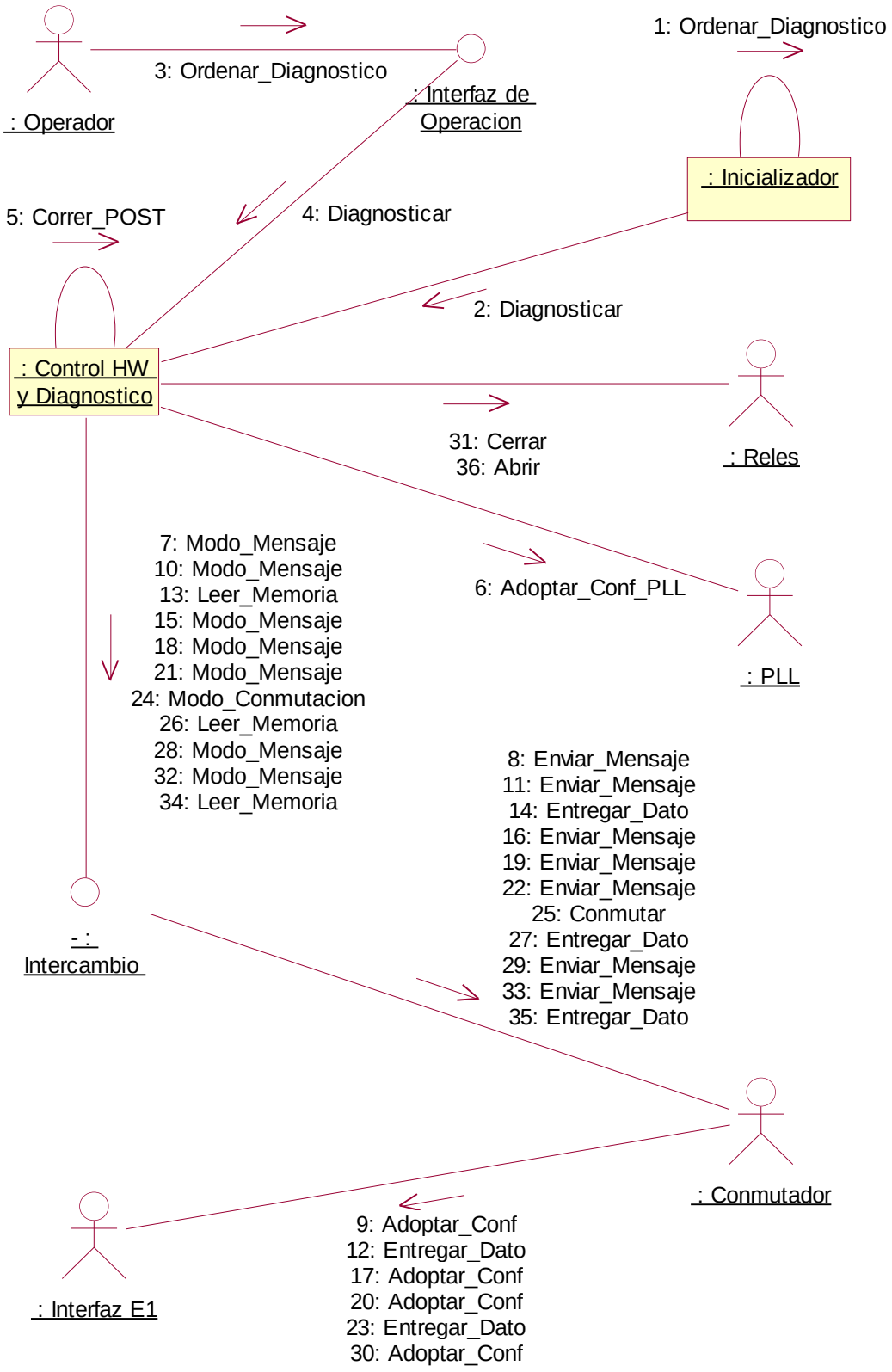
- La tarjeta se resetea y queda con su configuración inicial.
- Operador con posibilidad de ordenar la ejecución de otro diagnóstico o hacer uso de las demás funciones que ofrece el sistema.

Flujos alternativos: Ninguno.

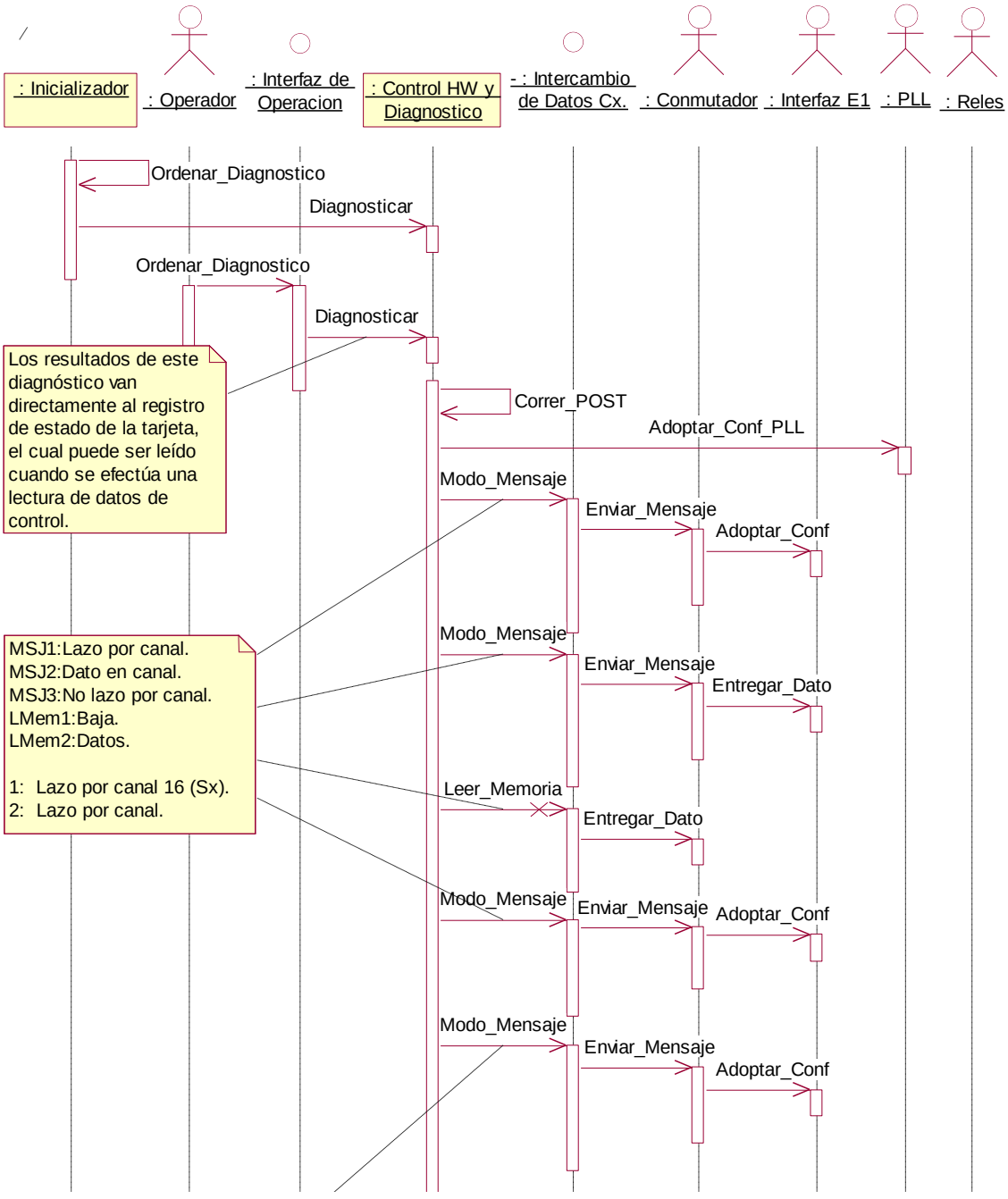
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

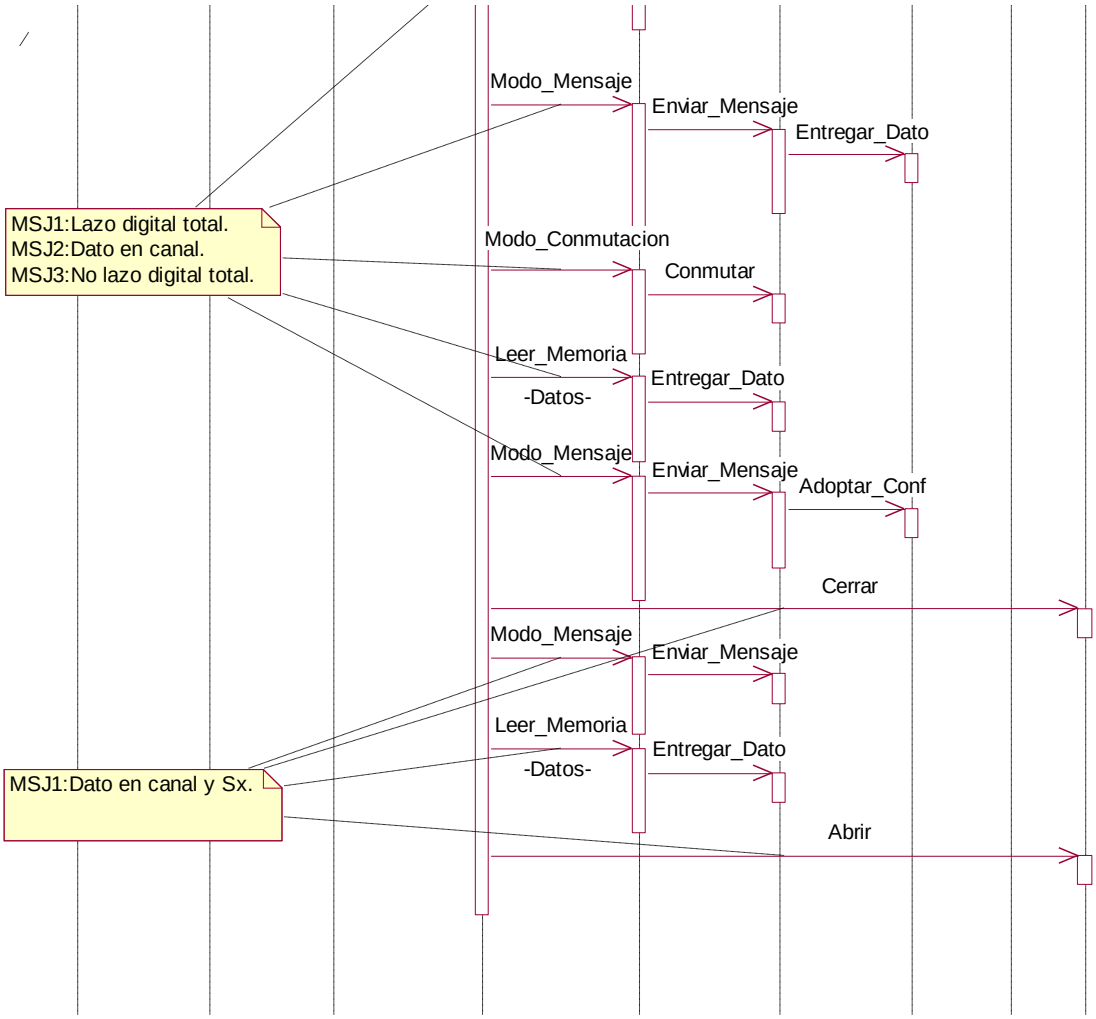
1.12.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.12.3 DIAGRAMA DE SECUENCIAS (INTERACCION)

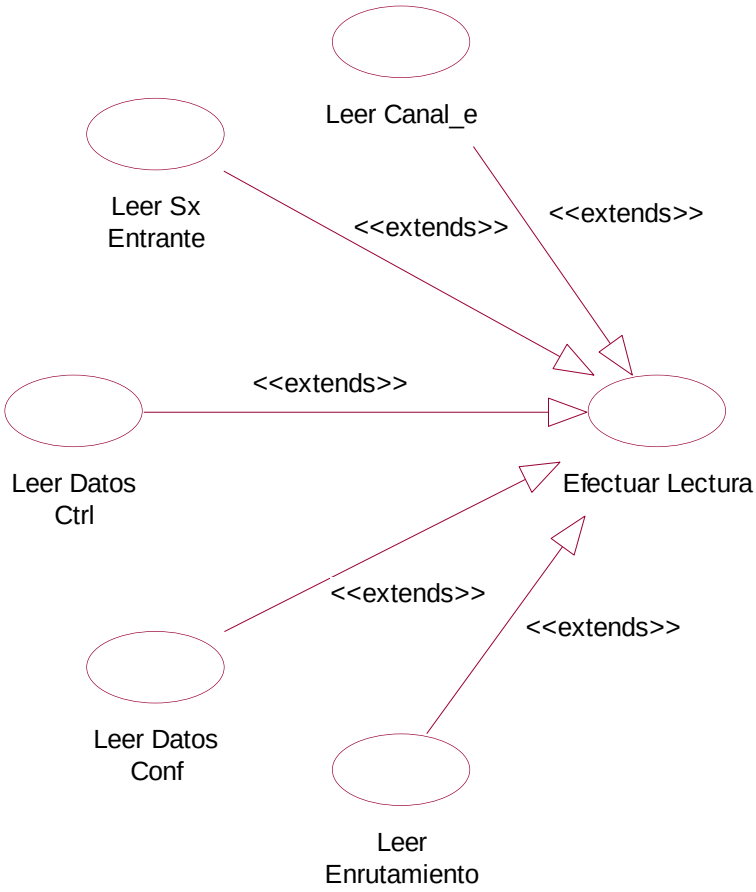


Continuación...



1.13 CASO DE USO EFECTUAR LECTURA (NIVEL 0)

1.13.1 EXTENSION



1.14 CASO DE USO EXTENDIDO LEER CANAL_E (NIVEL 1)

1.14.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los datos que identifican el parámetro a leer.

2. El sistema recoge los datos correspondientes a: Nombre del dato a leer (Canal_e), PCM de entrada y número del canal telefónico. Luego los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la tarjeta y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para acceder al dato solicitado por el operador por medio de una serie de secuencias.
5. Conmutador efectúa el llamado de la función Entregar Dato.
6. El sistema proporciona el dato solicitado para lectura al operador.

Postcondiciones:

- Operador obtiene el dato de su petición de lectura de un canal telefónico de entrada.
- Operador con posibilidad de efectuar una nueva lectura o hacer uso de las demás funciones que ofrece el sistema.

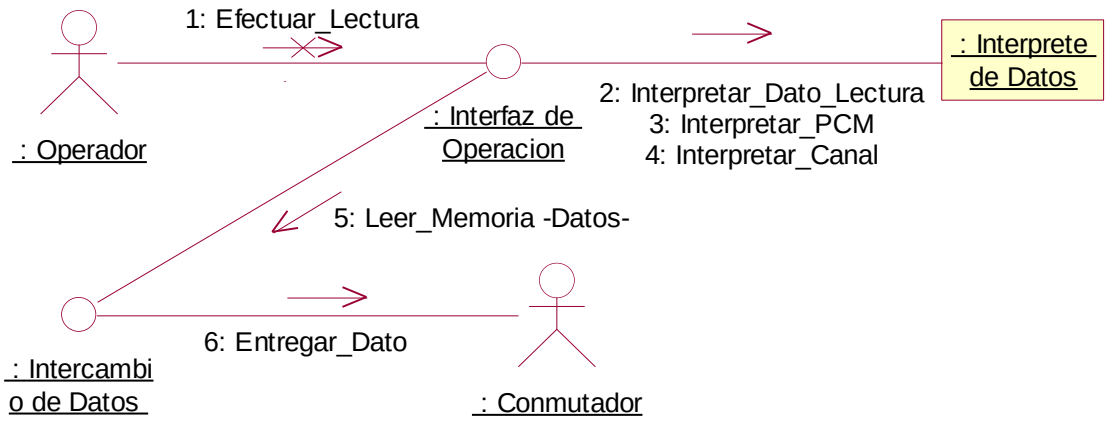
Flujos alternativos: Ninguno.

Excepciones:

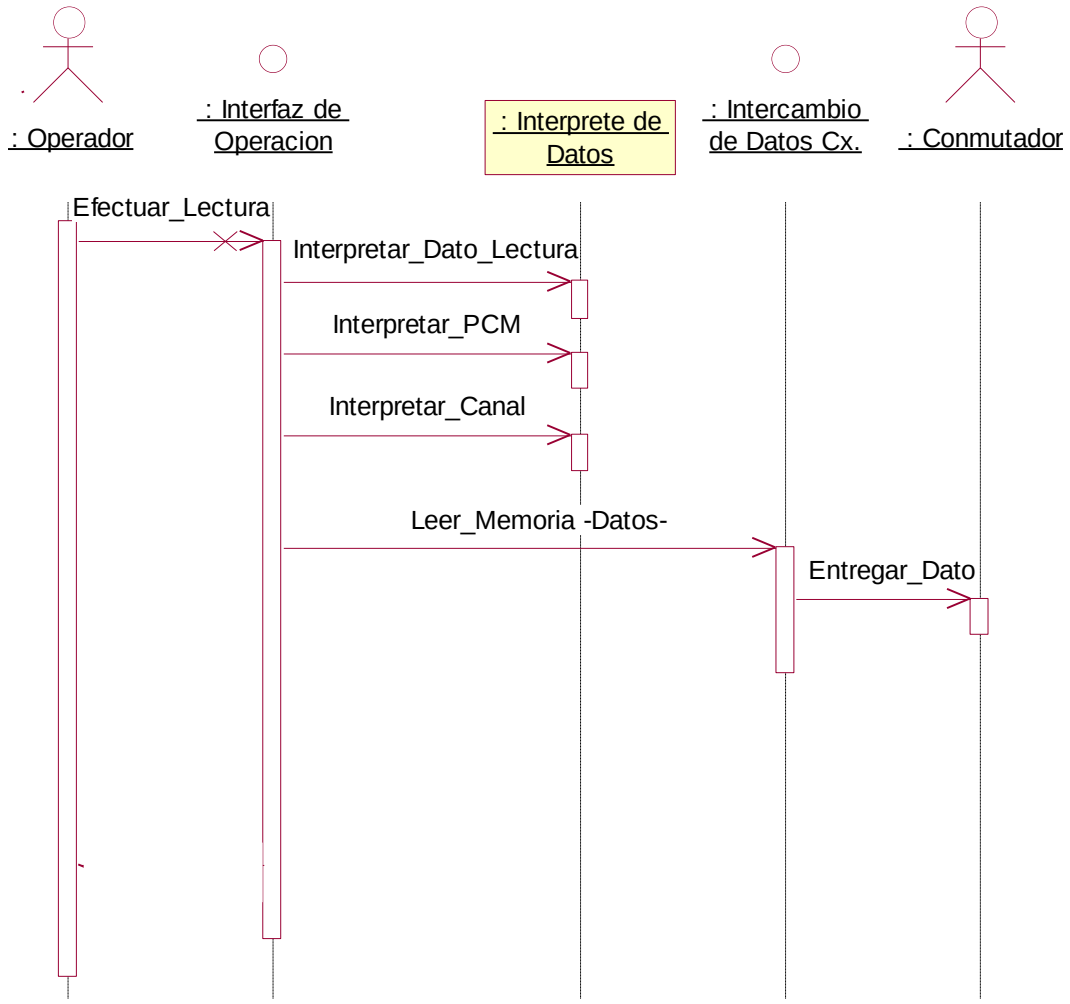
- Petición de información inexistente o inaccesible.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.14.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.14.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.15 CASO DE USO EXTENDIDO LEER SX ENTRANTE (NIVEL 1)

1.15.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los datos que identifican el parámetro a leer.

2. El sistema recoge los datos correspondientes a: Nombre del dato a leer (Sx Entrante) y número del canal telefónico señalizado. Luego los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la tarjeta y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para acceder al dato solicitado por el operador por medio de una serie de secuencias.
5. Interfaz E1 y Conmutador efectúan el llamado de la función Entregar Dato.
6. El sistema proporciona el dato solicitado para lectura al operador.

Postcondiciones:

- Operador obtiene el dato de su petición de lectura de la señalización de un canal telefónico.
- Operador con posibilidad de efectuar una nueva lectura o hacer uso de las demás funciones que ofrece el sistema.

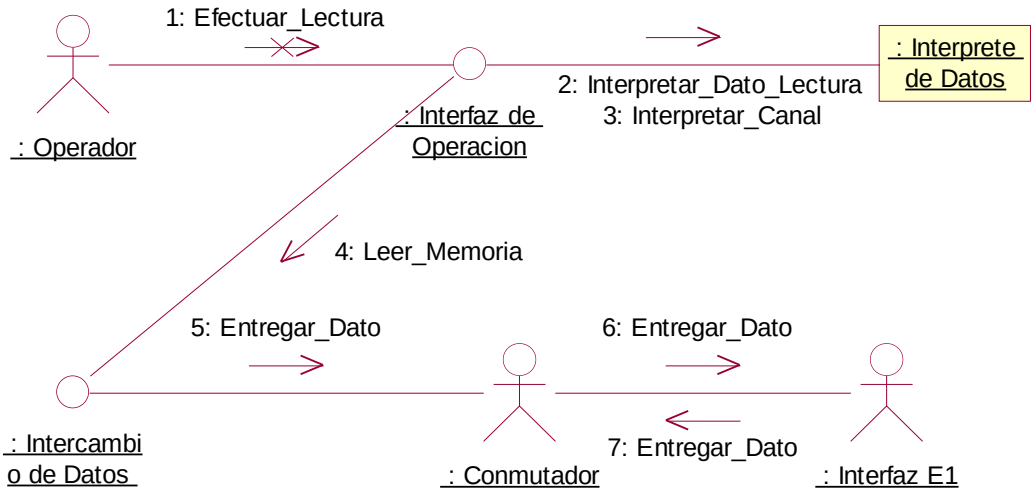
Flujos alternativos: Ninguno.

Excepciones:

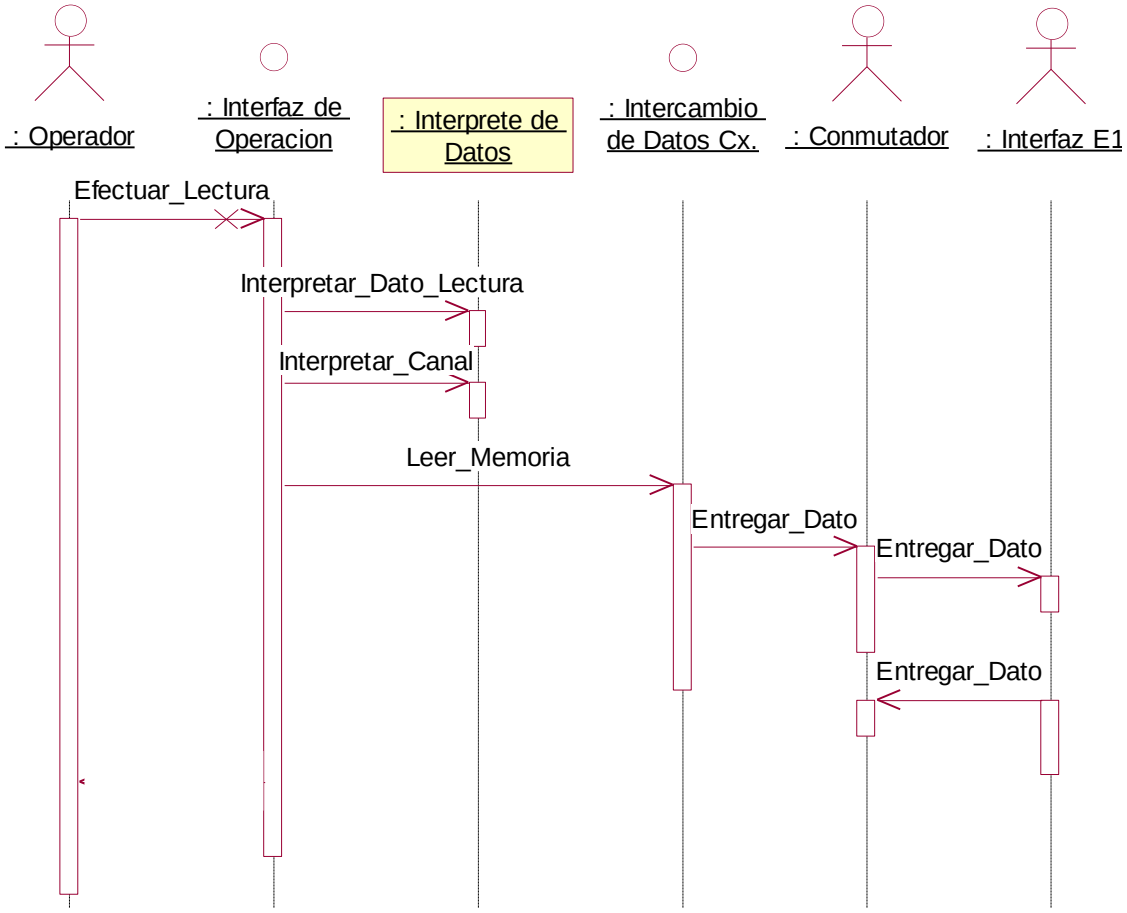
- Petición de información inexistente o inaccesible.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.15.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.15.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.16 CASO DE USO EXTENDIDO LEER DATOS CTRL (NIVEL 1)

1.16.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los datos que identifican el parámetro a leer.
2. El sistema recoge el dato correspondientes a: Nombre del dato a leer (Dato Control). Luego lo traduce a un formato entendible para él.

3. El sistema transforma el dato basándose en la arquitectura de la tarjeta y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para acceder al dato solicitado por el operador por medio de una serie de secuencias.
5. Interfaz E1 y Conmutador efectúan el llamado de la función Entregar Dato.
6. El sistema proporciona el dato solicitado para lectura al operador.

Postcondiciones:

- Operador obtiene el dato de su petición de lectura del dato de control.
- Operador con posibilidad de efectuar una nueva lectura o hacer uso de las demás funciones que ofrece el sistema.

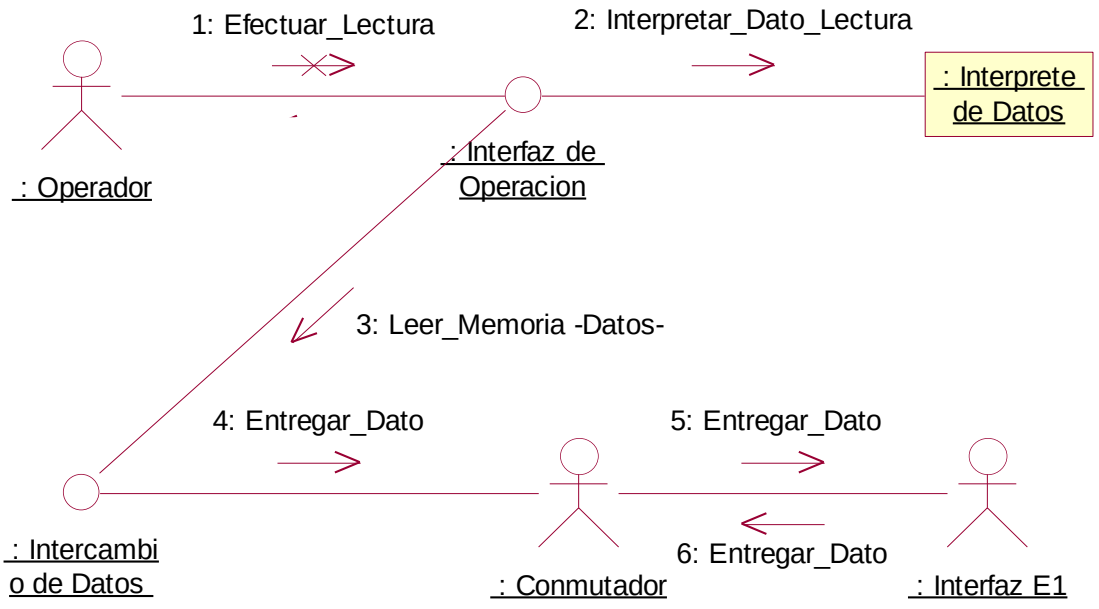
Flujos alternativos: Ninguno.

Excepciones:

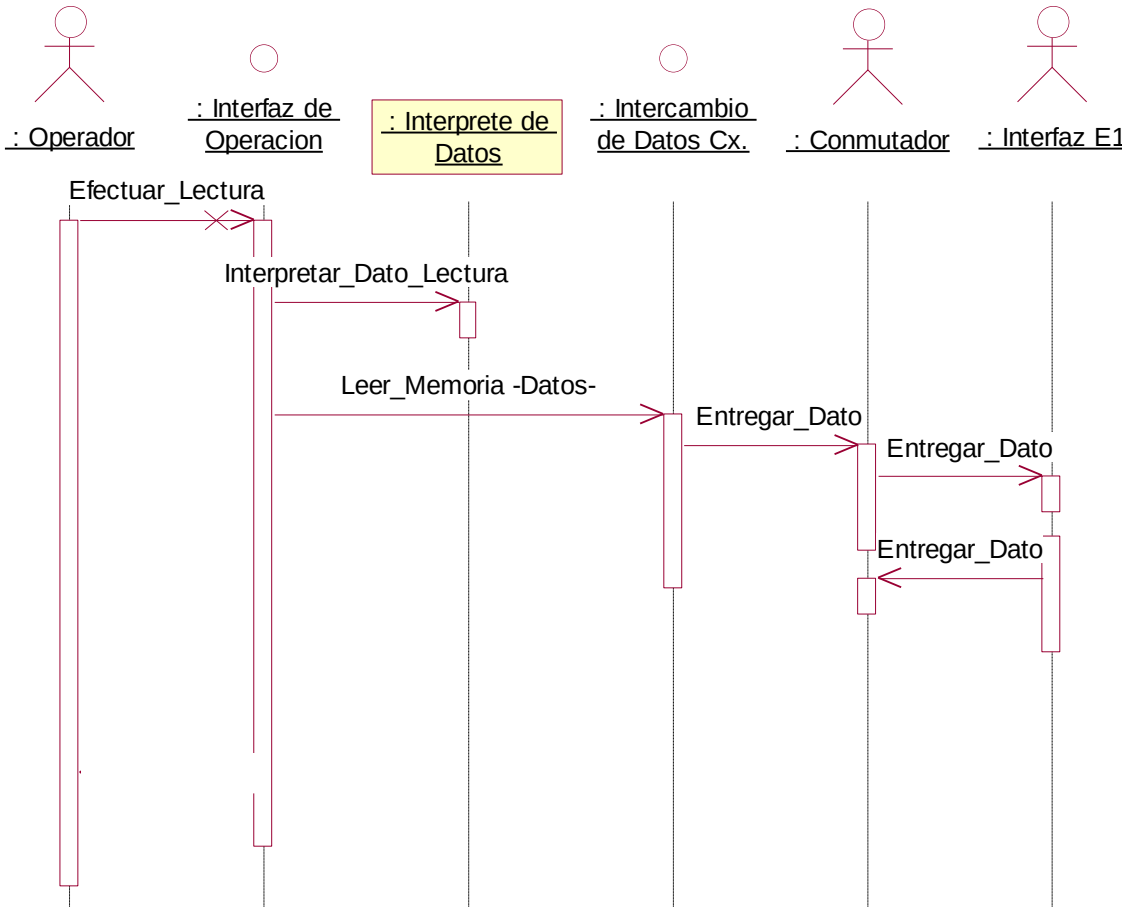
- Petición de información inexistente o inaccesible.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.16.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.16.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.17 CASO DE USO EXTENDIDO LEER DATOS CONF (NIVEL 1)

1.17.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los datos que identifican el parámetro a leer.
2. El sistema recoge el dato correspondientes a: Nombre del dato a leer (Dato Configuración). Luego lo traduce a un formato entendible para él.

3. El sistema transforma el dato basándose en la arquitectura de la tarjeta y el formato de entramado PCM.
4. El sistema se comunica con el conmutador, usando un protocolo específico, para acceder al dato solicitado por el operador por medio de una serie de secuencias.
5. Conmutador efectúa el llamado de la función Entregar Dato.
6. El sistema proporciona el dato solicitado para lectura al operador.

Postcondiciones:

- Operador obtiene el dato de su petición de lectura del dato de configuración.
- Operador con posibilidad de efectuar una nueva lectura o hacer uso de las demás funciones que ofrece el sistema.

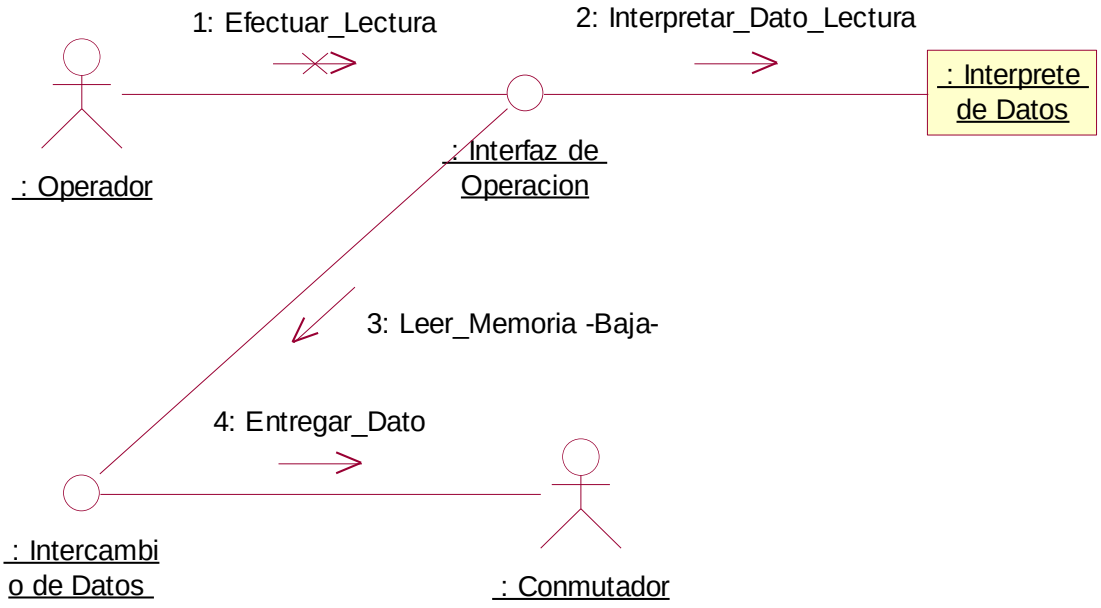
Flujos alternativos: Ninguno.

Excepciones:

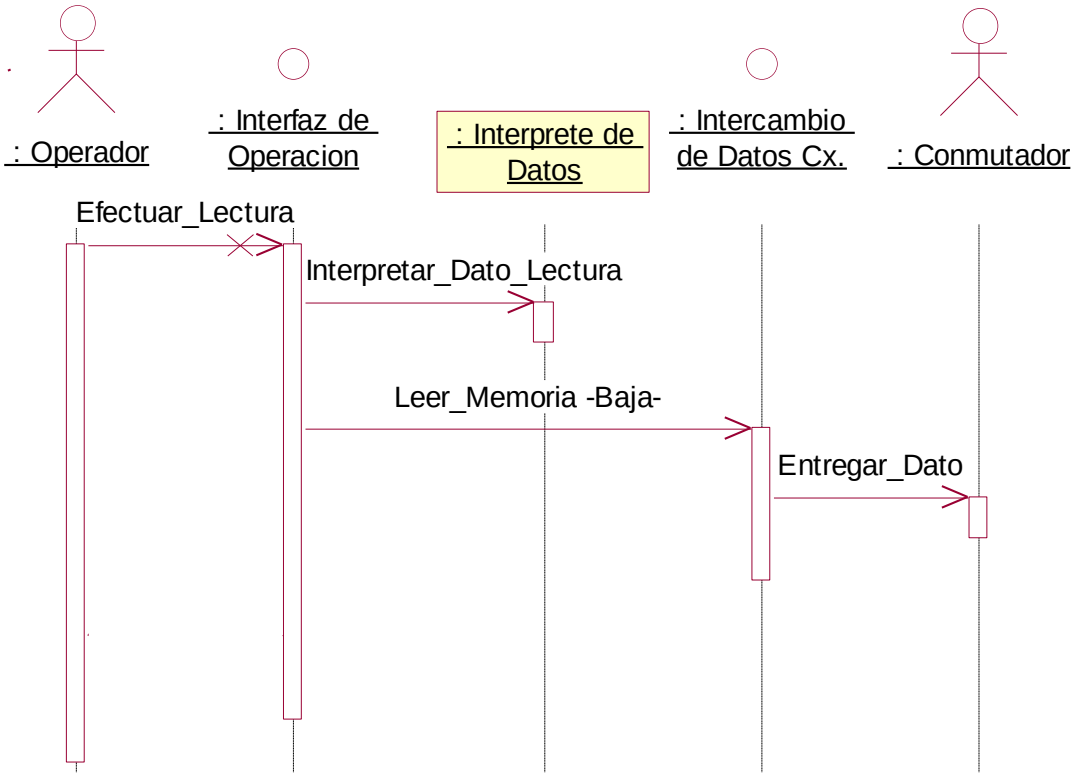
- Petición de información inexistente o inaccesible.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.17.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.17.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.18 CASO DE USO EXTENDIDO LEER ENRUTAMIENTO (NIVEL 1)

1.18.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los datos que identifican el parámetro a leer.
2. El sistema recoge los datos correspondientes a: Nombre del dato a leer (Enrutamiento de canal), PCM de salida y número del canal telefónico de salida. Luego los traduce a un formato entendible para él.
3. El sistema transforma los datos basándose en la arquitectura de la tarjeta y el formato de entramado PCM.

4. El sistema se comunica con el conmutador usando un protocolo específico para verificar, leyendo el modo de operación del canal en la memoria alta de conexión, si el dato introducido por el operador se encuentra en estado enrutado; de ser así va al paso 5, de lo contrario se informa que el enrutamiento para dicho canal es inexistente y se aborta la operación de lectura.
5. El sistema se comunica con el conmutador para leer la memoria baja de conexión cuyo contenido es el PCM e IT de entrada referenciados y por consiguiente enrutados al canal de salida.
6. El sistema recibe estos datos y los traduce a un formato comprensible para el operador.
7. El sistema proporciona el PCM y el canal solicitado para lectura al operador.

Postcondiciones:

- Operador obtiene el dato del enrutamiento de un canal telefónico de salida o la información referente a la inexistencia del enrutamiento para dicho canal.
- Operador con posibilidad de efectuar una nueva lectura o hacer uso de las demás funciones que ofrece el sistema.

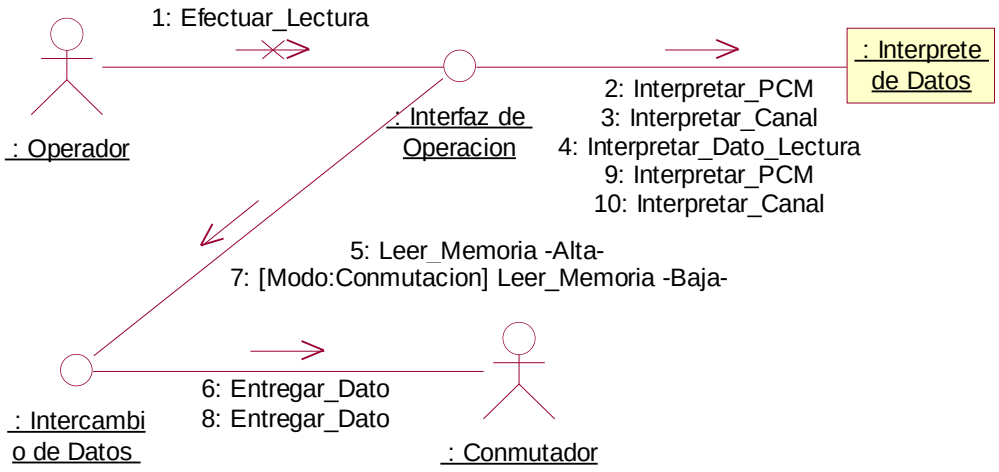
Flujos alternativos: Ninguno.

Excepciones:

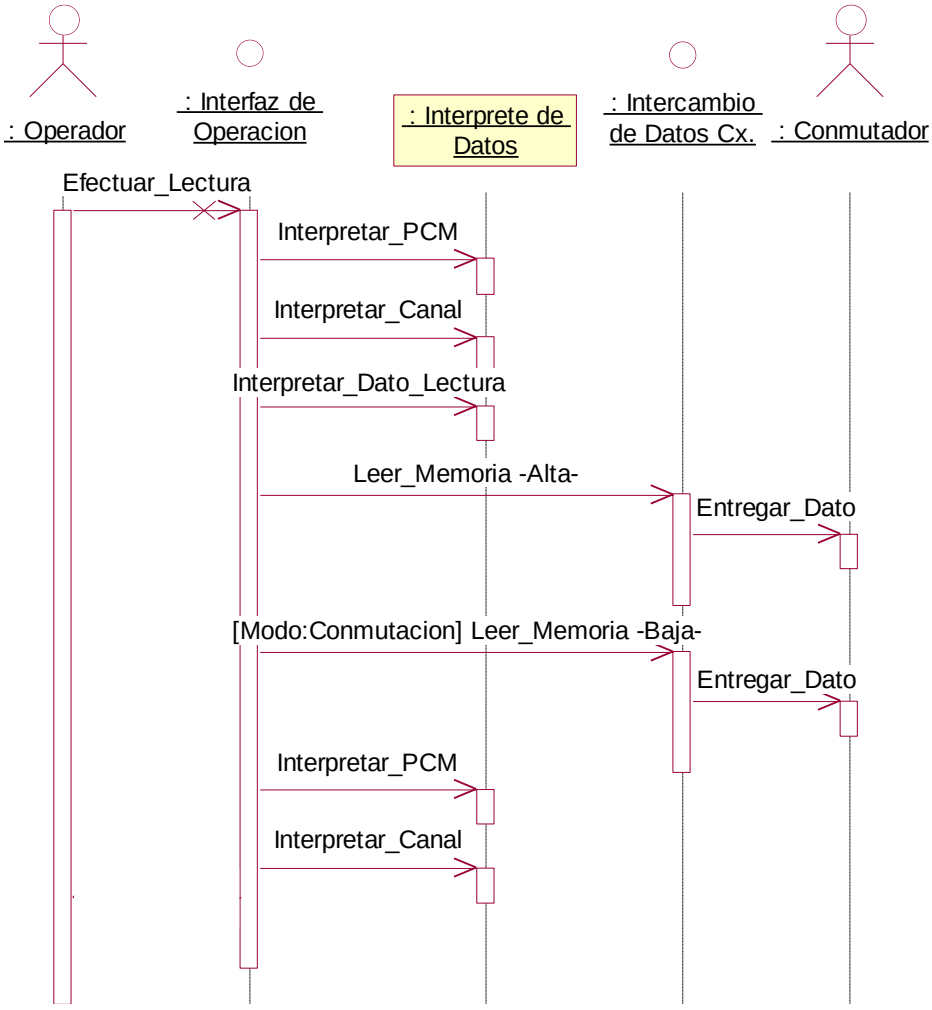
- Petición de información inexistente o inaccesible.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.18.2 DIAGRAMA DE COLABORACION (INTERACCION)

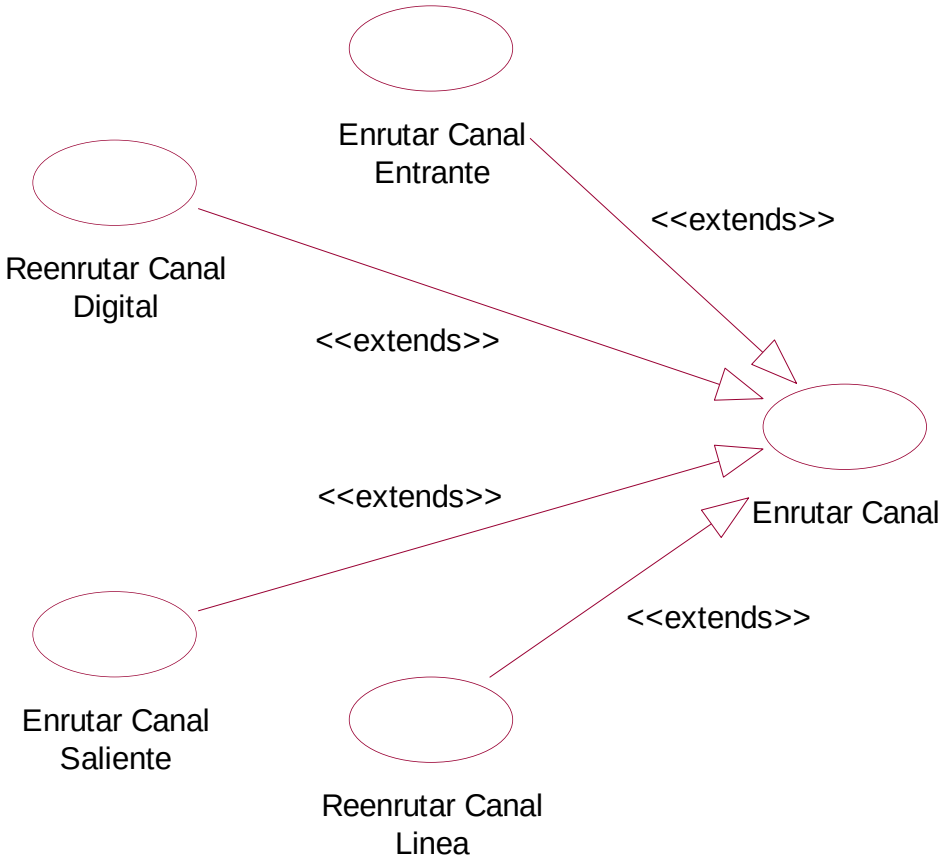


1.18.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.19 CASO DE USO ENRUTAR CANAL (NIVEL 0)

1.19.1 EXTENSION



1.20 CASO DE USO EXTENDIDO ENRUTAR CANAL ENTRANTE (NIVEL 1)

1.20.1 DESCRIPCION DE ESCENARIO

Similar (escenario y diagramas de interacción) al caso de uso Enrutar canal, cuyo nivel de abstracción es 0.

1.21 CASO DE USO EXTENDIDO REENRUTAR CANAL DIGITAL (NIVEL 1)

1.21.1 DESCRIPCION DE ESCENARIO

Similar (escenario y diagramas de interacción) al caso de uso Enrutar canal, cuyo nivel de abstracción es 0.

1.22 CASO DE USO EXTENDIDO ENRUTAR CANAL SALIENTE (NIVEL 1)

1.22.1 DESCRIPCION DE ESCENARIO

Similar (escenario y diagramas de interacción) al caso de uso Enrutar canal, cuyo nivel de abstracción es 0.

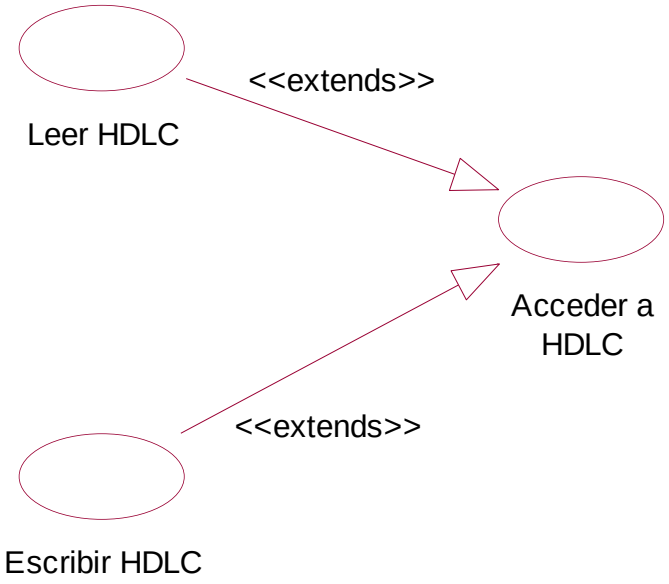
1.23 CASO DE USO EXTENDIDO REENRUTAR CANAL LINEA (NIVEL 1)

1.23.1 DESCRIPCION DE ESCENARIO

Similar (escenario y diagramas de interacción) al caso de uso Enrutar canal, cuyo nivel de abstracción es 0.

1.24 CASO DE USO ACCEDER A HDLC (NIVEL 0)

1.24.1 EXTENSION



1.25 CASO DE USO EXTENDIDO LEER HDLC (NIVEL 1)

1.25.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los parámetros de acceso al dispositivo HDLC (Lectura, Dirección).
2. El sistema accede al dato.
3. Se efectúa el llamado de la función Entregar Dato para facilitar el dato al operador.

Postcondiciones: Operador con posibilidad de acceder de nuevo al HDLC o hacer uso de las demás funciones que ofrece el sistema.

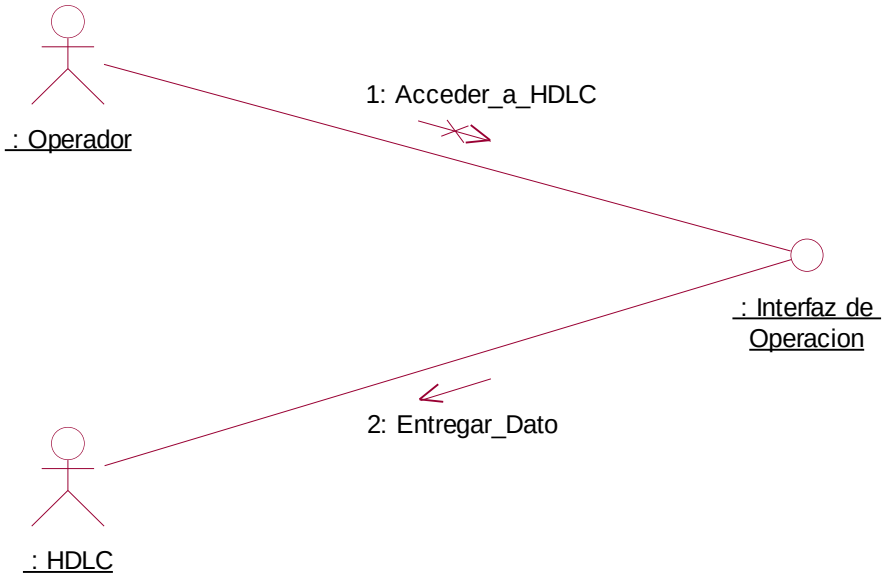
Flujos alternativos: Ninguno.

Excepciones:

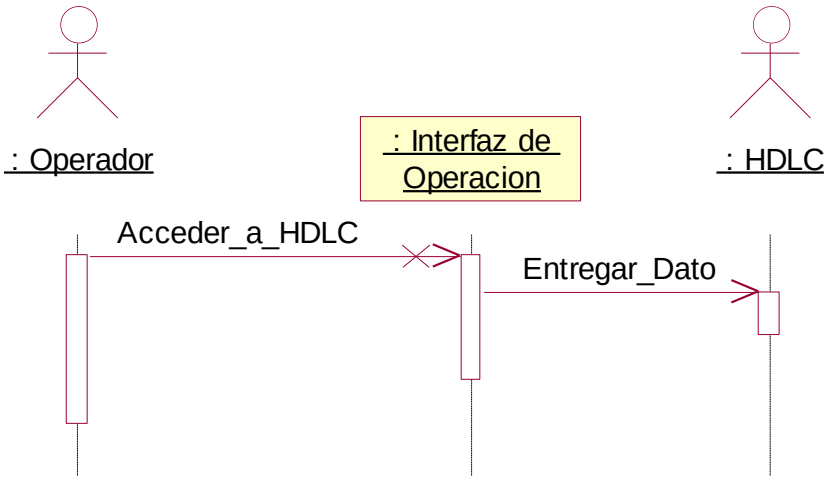
- Información fuera de rango.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.25.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.25.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.26 CASO DE USO EXTENDIDO ESCRIBIR HDLC (NIVEL 1)

1.26.1 DESCRIPCION DE ESCENARIO

Iniciador: Operador.

Precondición: Ninguna.

Flujo de eventos:

1. El operador introduce los parámetros de acceso al dispositivo HDLC (Escritura, Dirección, Dato).
2. El sistema coloca el dato en la dirección específica.
3. Se efectúa el llamado de la función Recibir Dato.

Postcondiciones: Operador con posibilidad de acceder de nuevo al HDLC o hacer uso de las demás funciones que ofrece el sistema.

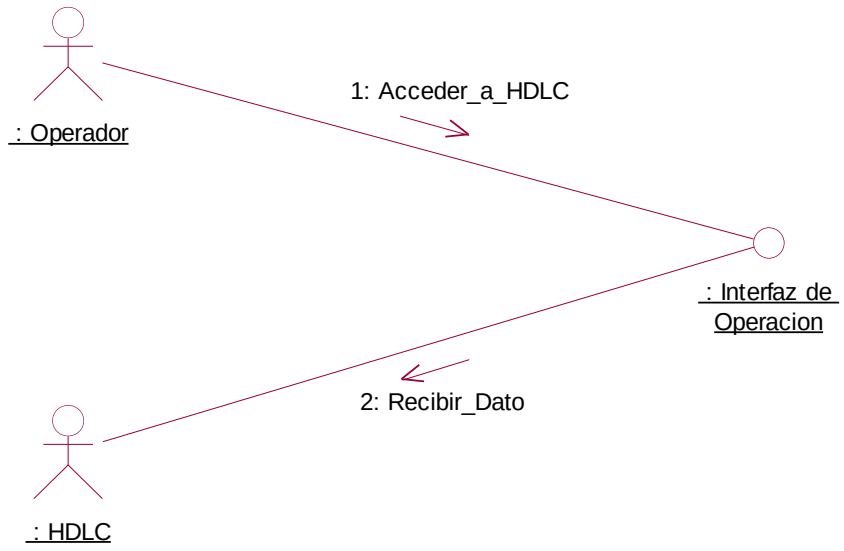
Flujos alternativos: Ninguno.

Excepciones:

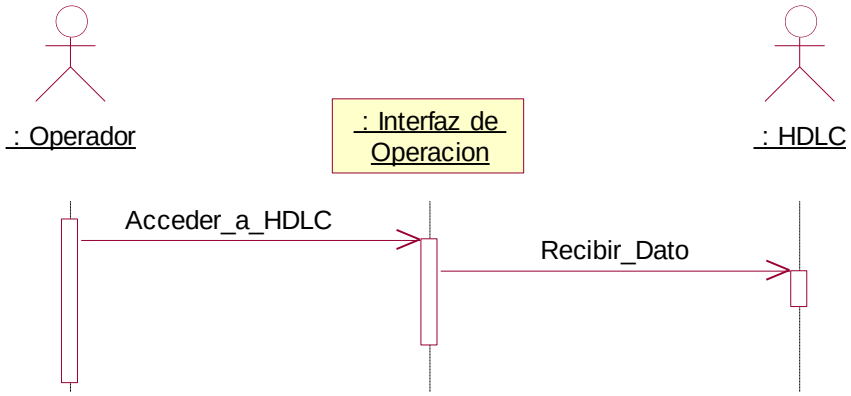
- Información fuera de rango.
- Fallas de comunicación.

Recursos especiales: Ninguno.

1.26.2 DIAGRAMA DE COLABORACION (INTERACCION)

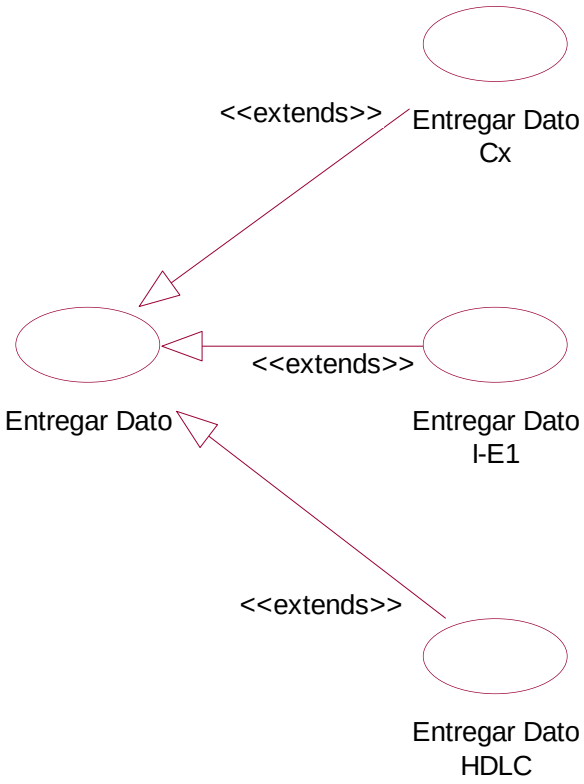


1.26.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



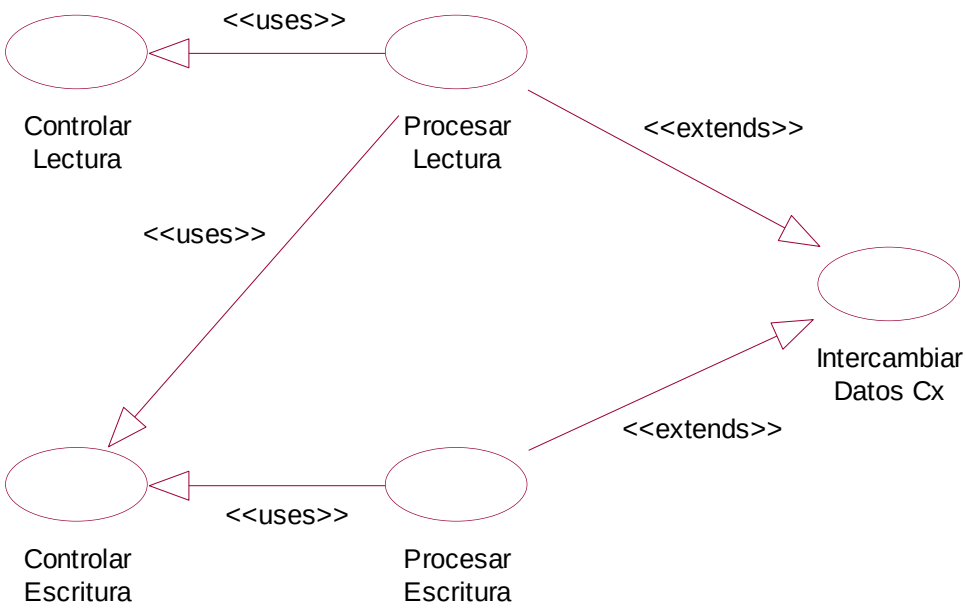
1.27 CASO DE USO ENTREGAR DATO (NIVEL 0)

1.27.1 EXTENSION



1.28 CASO DE USO INTERCAMBIAR DATOS CX (NIVEL 0)

1.28.1 EXTENSION



1.28.2 DESCRIPCION DE ESCENARIO

Iniciador: Intercambio de Datos Cx.

Precondición: Operador ha hecho uso de Enrutar Canal, Introducir Sx, Configurar Stma, Efectuar Lectura o Diagnosticar.

Flujo de eventos:

1. Intercambio de Datos Cx recibe la información pertinente para la programación del actor conmutador.
2. Se procede a formatear adecuadamente esos datos, según el protocolo de comunicación y la manera de programar el conmutador, con el fin de realizar el "handshake" entre el sistema y este actor.
3. Se entabla la comunicación con el conmutador ejecutando el handshake.

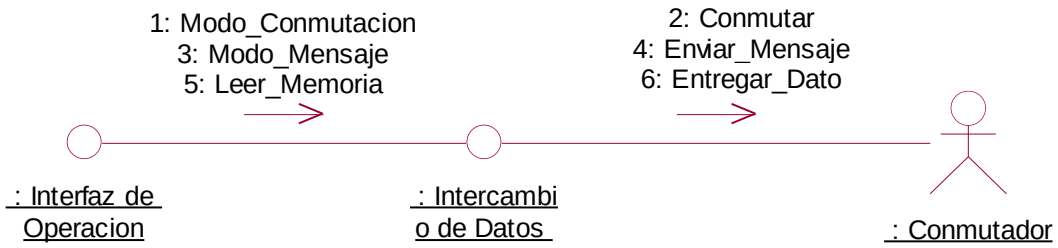
Postcondiciones: El conmutador se programa y permanece ejecutando la función solicitada.

Flujos alternativos: Ninguno.

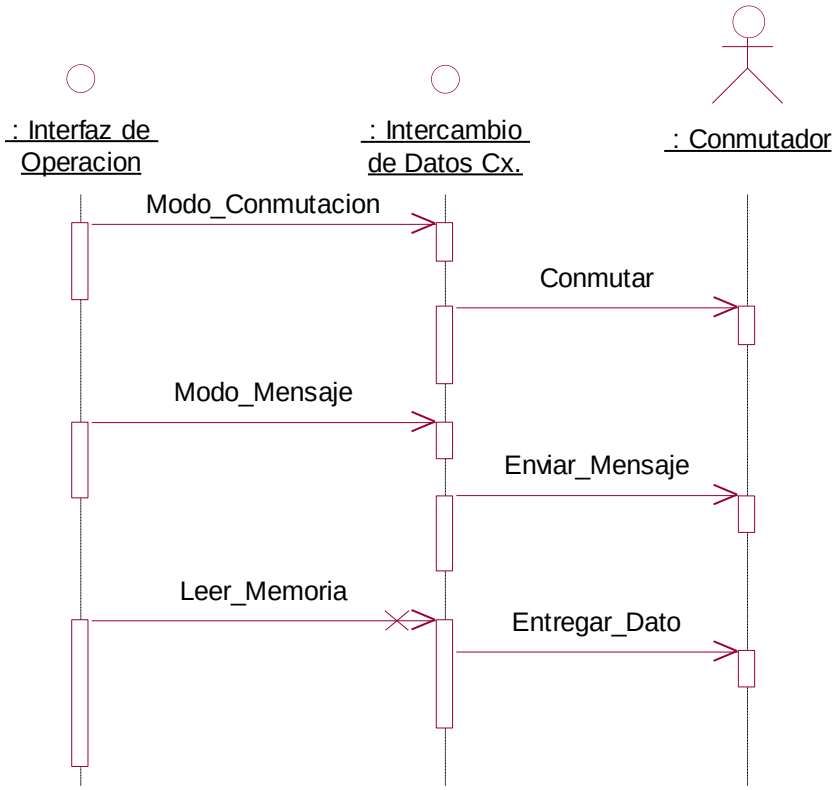
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

1.28.3 DIAGRAMA DE COLABORACION (INTERACCION)



1.28.4 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.29 CASO DE USO EXTENDIDO PROCESAR LECTURA (NIVEL 1)

1.29.1 DESCRIPCION DE ESCENARIO

Este caso de uso es el encargado de armar las palabras de programación del actor conmutador para lectura, ya que luego serán dirigidas de manera secuencial hacia el registro de control a través de los buses de datos y direcciones del mismo. Posteriormente recibirá la información solicitada en dicho proceso de lectura.

Iniciador: Intercambio de Datos Cx.

Precondición: Operador ha hecho uso de Efectuar Lectura o Diagnosticar.

Flujo de eventos:

1. Intercambio de Datos Cx recibe la información pertinente para la programación del actor conmutador.
2. Con esa información se procede a armar las palabras de programación del actor conmutador para lectura, según la manera como este se programa.
3. Estas palabras se envían de manera secuencial hacia el conmutador.
4. Se reciben los datos leídos.

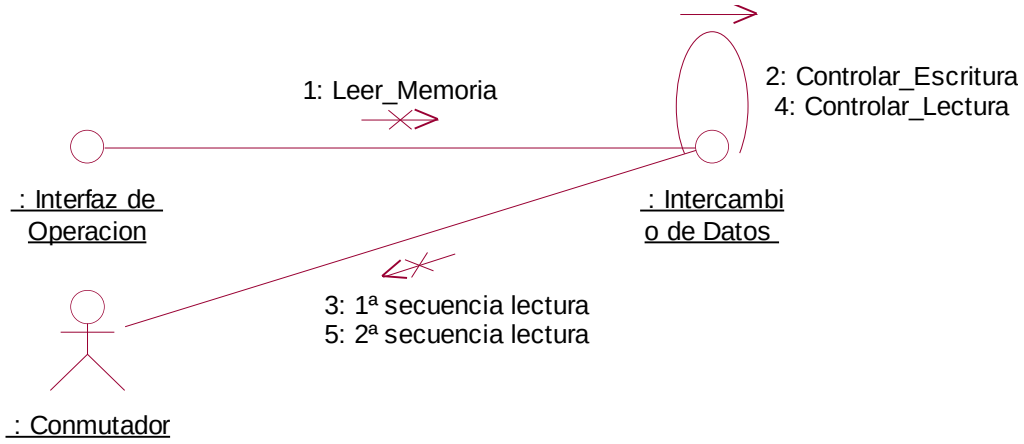
Postcondiciones: El conmutador se programa y permanece ejecutando la función solicitada.

Flujos alternativos: Ninguno.

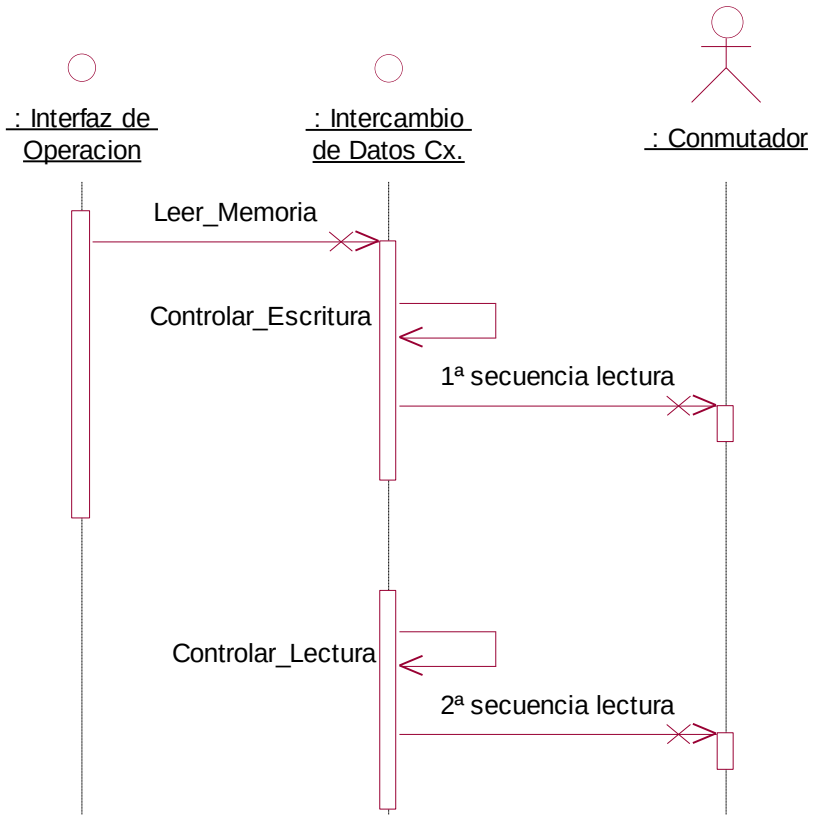
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

1.29.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.29.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.30 CASO DE USO EXTENDIDO PROCESAR ESCRITURA (NIVEL 1)

1.30.1 DESCRIPCION DE ESCENARIO

Este caso de uso es el encargado de armar las palabras de programación del actor conmutador para escritura, ya que luego serán dirigidas de manera secuencial hacia el registro de control a través de los buses de datos y direcciones del mismo.

Iniciador: Intercambio de Datos Cx.

Precondición: Operador ha hecho uso de Enrutar Canal, Introducir Sx, Configurar Stma o Diagnosticar.

Flujo de eventos:

1. Intercambio de Datos Cx recibe la información pertinente para la programación del actor conmutador.
2. Con esa información se procede a armar las palabras de programacion del actor conmutador para escritura, según la manera como este se programa.
3. Estas palabras se envían de manera secuencial hacia el conmutador.

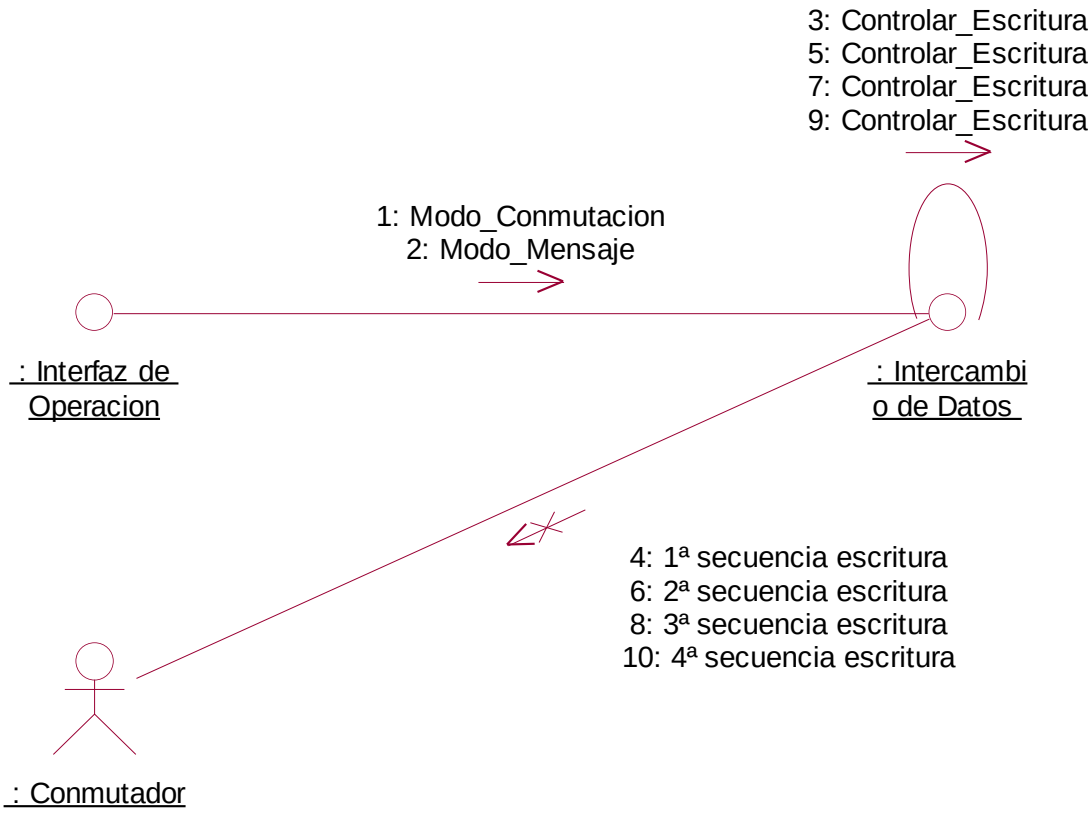
Postcondiciones: El conmutador se programa y permanece ejecutando la función solicitada.

Flujos alternativos: Ninguno.

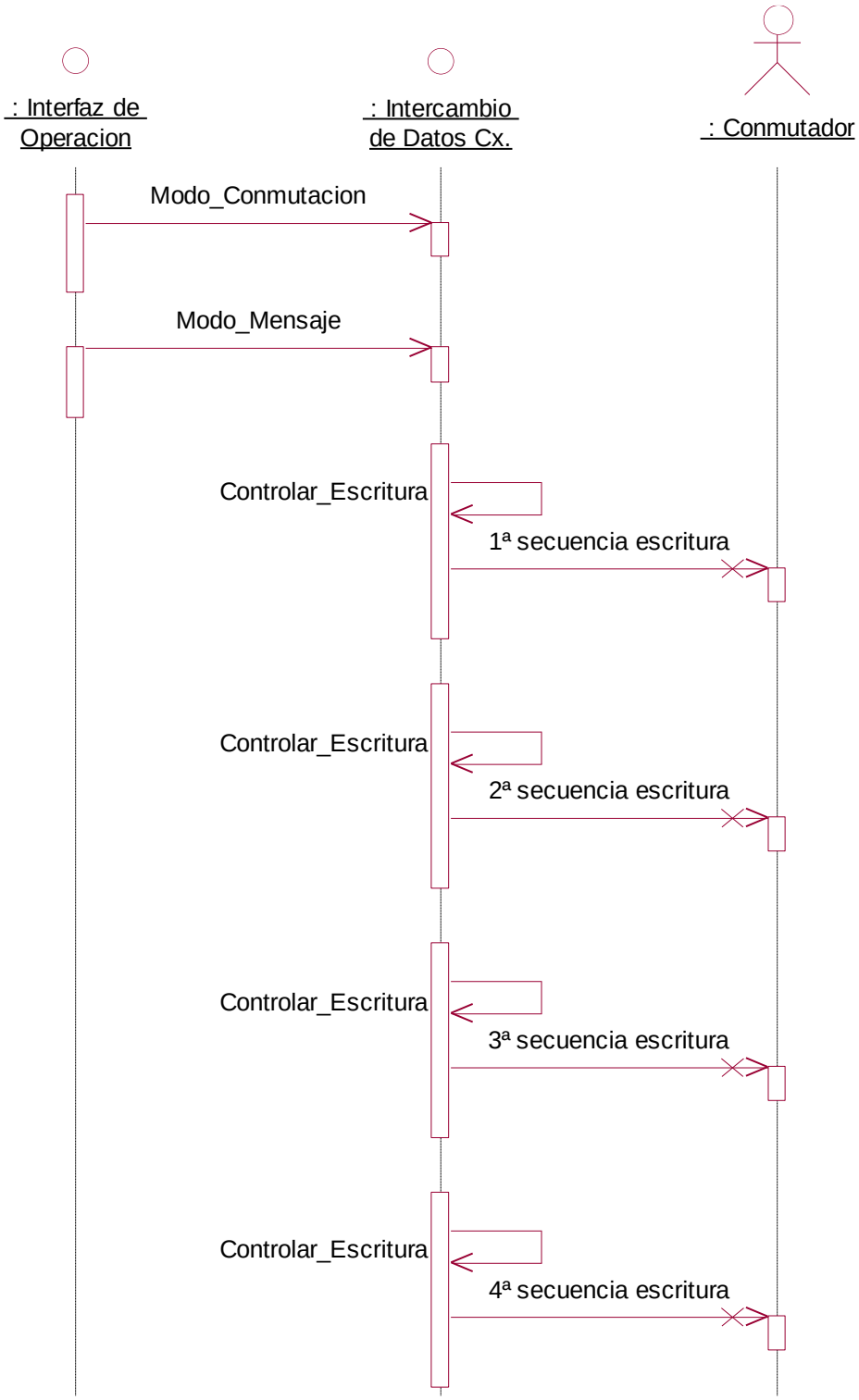
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

1.30.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.30.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.31 CASO DE USO EXTENDIDO CONTROLAR LECTURA (NIVEL 1)

1.31.1 DESCRIPCION DE ESCENARIO

Este caso de uso ejecuta el proceso de lectura sobre el registro de control o una zona de memoria del actor conmutador, teniendo en cuenta los cambios de estado de los bits de control del mismo (handshake).

Iniciador: Intercambio de Datos Cx.

Precondición: Operador ha hecho uso de Efectuar Lectura o Diagnosticar.

Flujo de eventos:

1. Las palabras de programación son dirigidas de manera secuencial hacia el registro de control a través de los buses de datos y direcciones del conmutador llevándose a cabo el handshake, en el cual se tienen en cuenta los cambios de estado de los bits de control del mismo.
2. Se obtiene el dato a leer, procedente de una zona de memoria, a través del registro de control del conmutador.

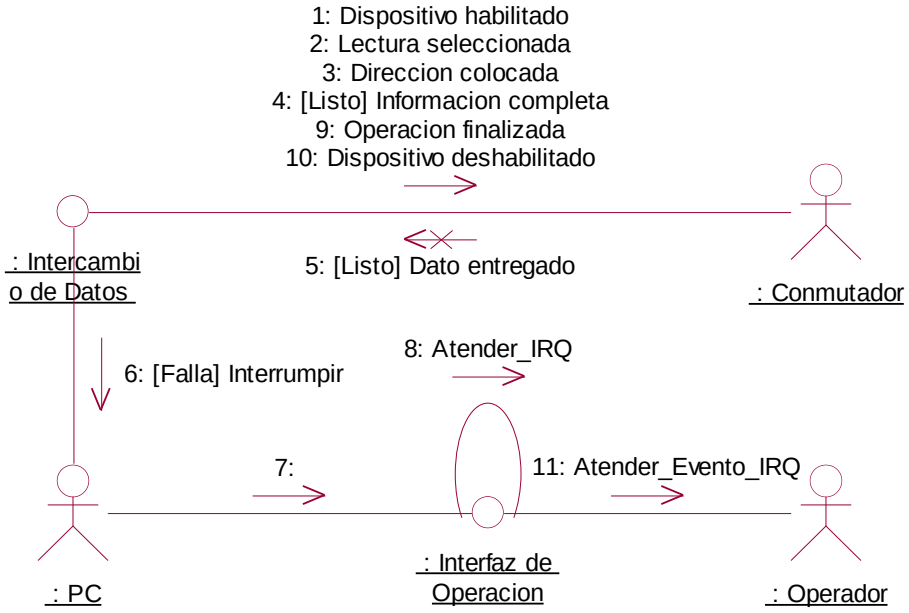
Postcondiciones: El conmutador queda en espera de otra secuencia de programación.

Flujos alternativos: Si después de cierto tiempo no se detecta el informe del dato entregado por parte del conmutador, se incurre en una falla de comunicación, se genera una interrupción hacia el PC y se finaliza la operación.

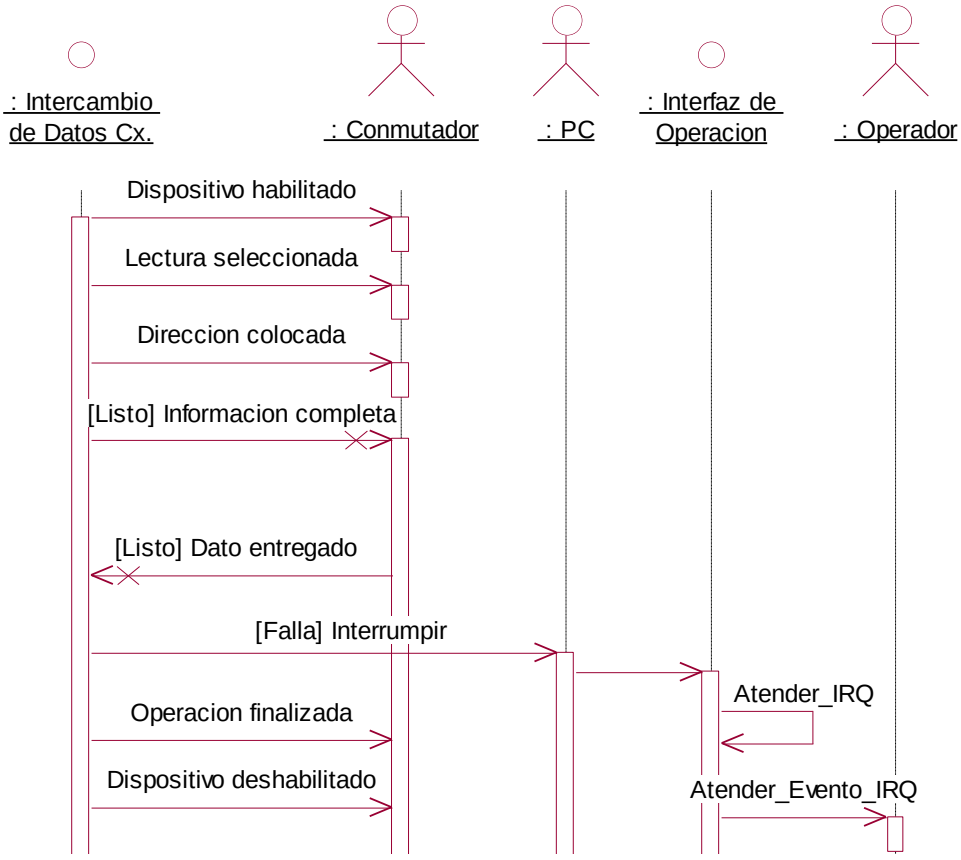
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

1.31.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.31.3 DIAGRAMA DE SECUENCIAS (INTERACCION)



1.32 CASO DE USO EXTENDIDO CONTROLAR ESCRITURA (NIVEL 1)

1.32.1 DESCRIPCION DE ESCENARIO

Este caso de uso ejecuta el proceso de escritura sobre el registro de control o una zona de memoria del actor conmutador, teniendo en cuenta los cambios de estado de los bits de control del mismo (handshake).

Iniciador: Intercambio de Datos Cx.

Precondición: Operador ha hecho uso de Enrutar Canal, Introducir Sx, Configurar Stma, Efectuar Lectura o Diagnosticar.

Flujo de eventos:

1. Las palabras de programación son dirigidas de manera secuencial hacia el registro de control a través de los buses de datos y direcciones del conmutador llevándose a cabo el handshake, en el cual se tienen en cuenta los cambios de estado de los bits de control del mismo.

Postcondiciones:

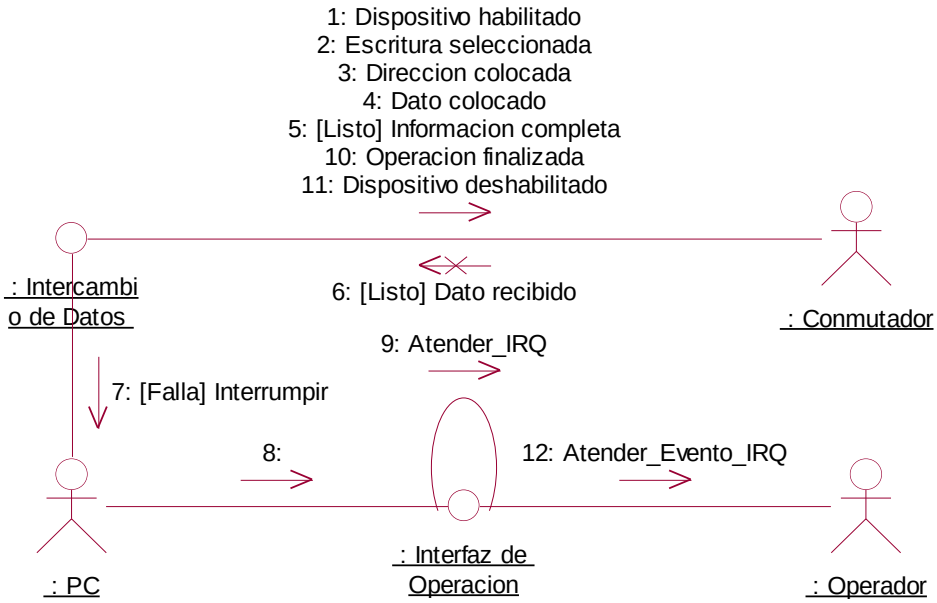
- El contenido del registro de control o de una zona de memoria queda modificado con el valor del dato escrito.
- El conmutador queda en espera de otra secuencia de programación.

Flujos alternativos: Si después de cierto tiempo no se detecta el informe del dato recibido por parte del conmutador, se incurre en una falla de comunicación, se genera una interrupción hacia el PC y se finaliza la operación.

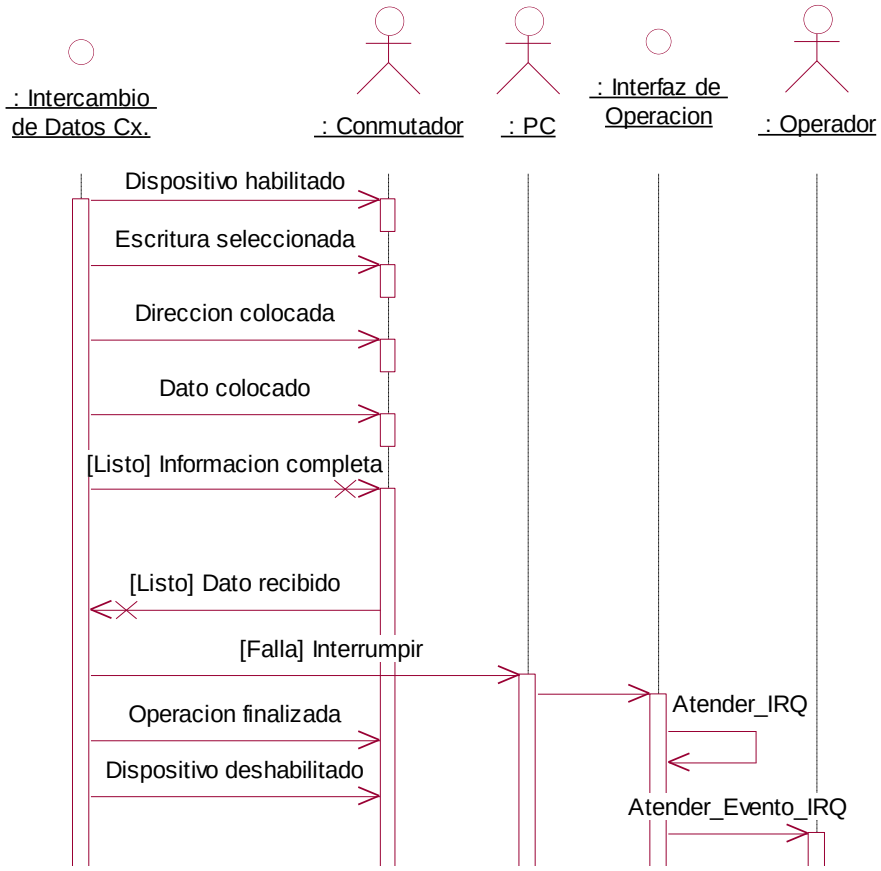
Excepciones: Fallas de comunicación.

Recursos especiales: Ninguno.

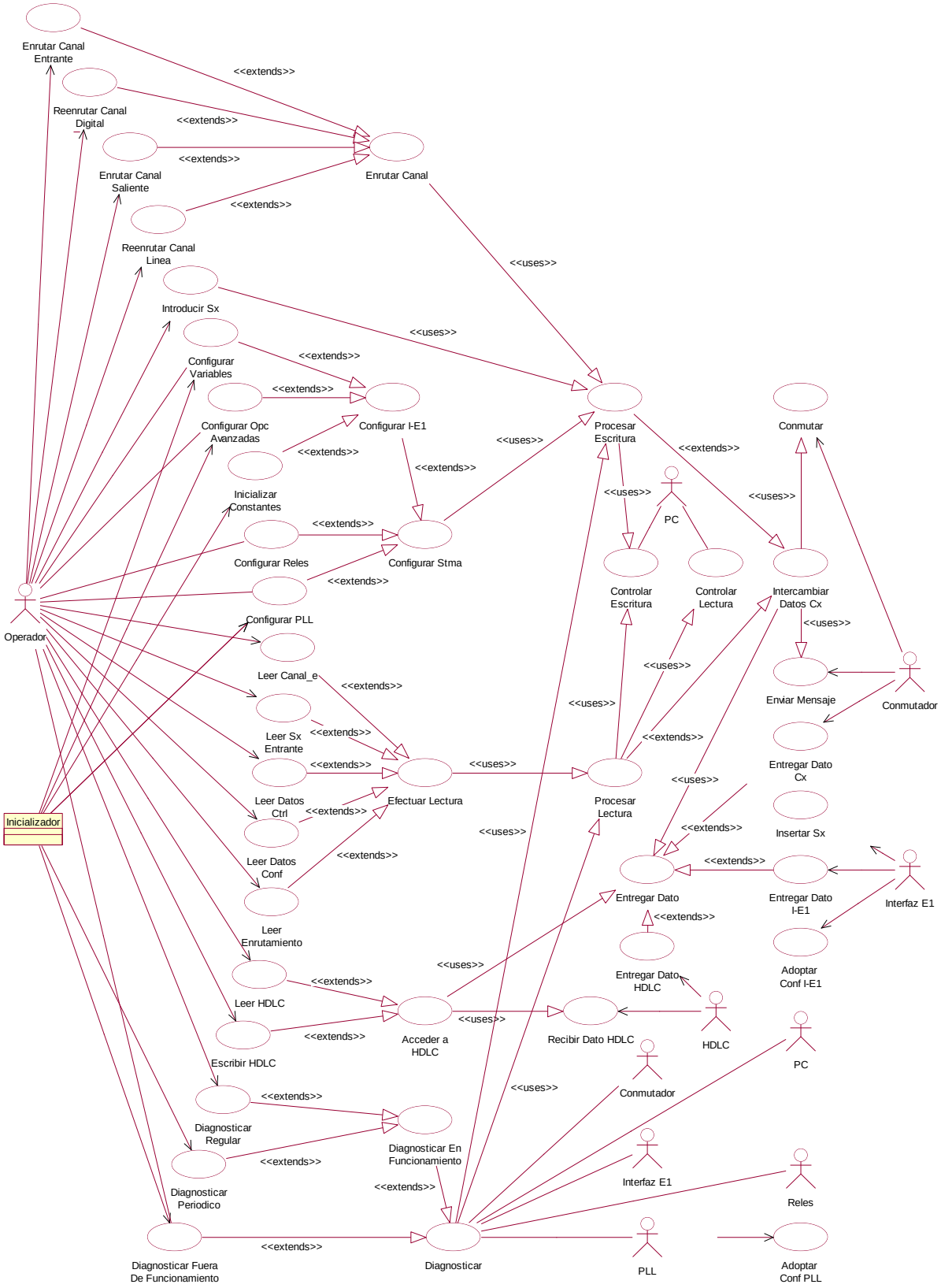
1.32.2 DIAGRAMA DE COLABORACION (INTERACCION)



1.32.3 DIAGRAMA DE SECUENCIAS (INTERACCION)

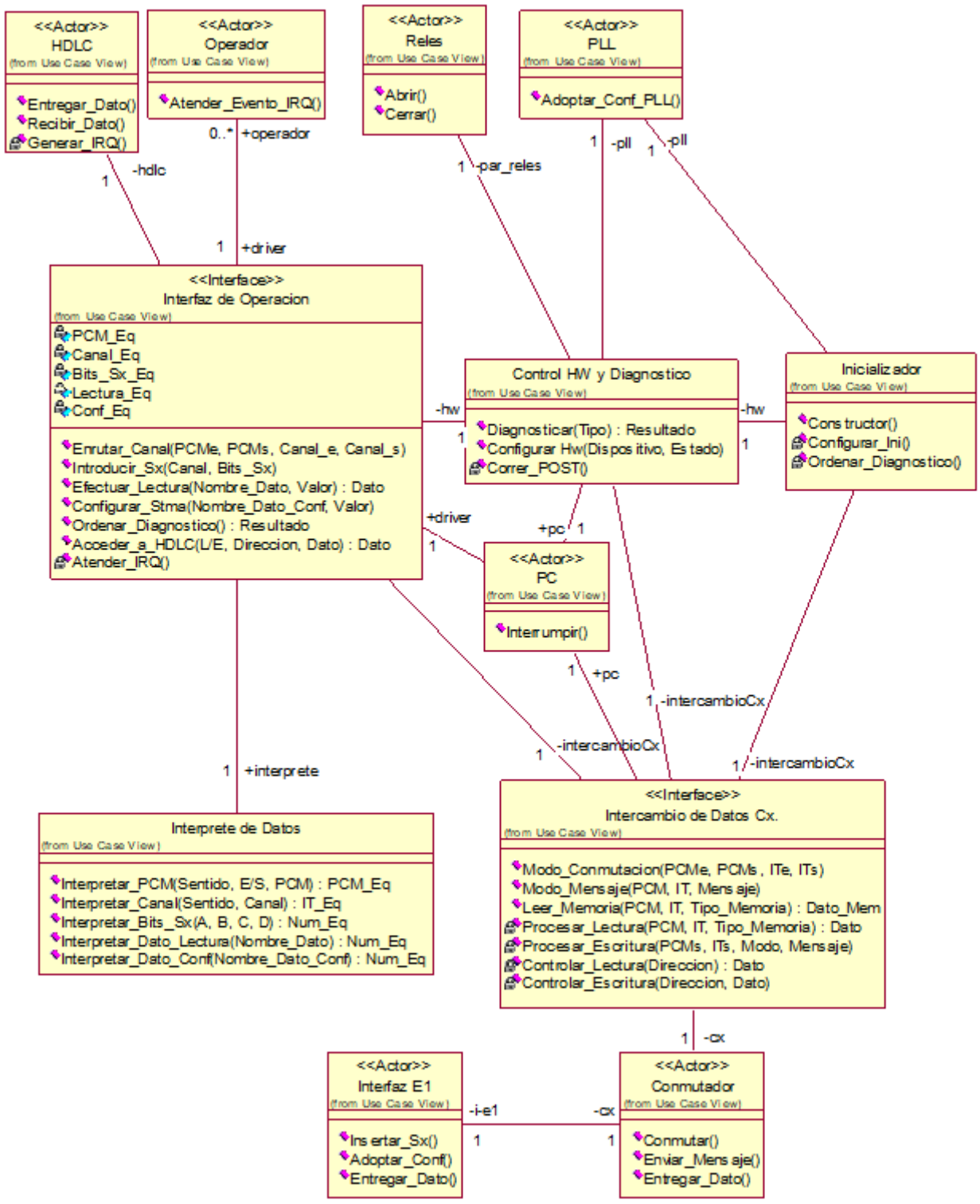


2 DIAGRAMA DE CASOS DE USO DE DISEÑO



3 DISEÑO DE CLASES

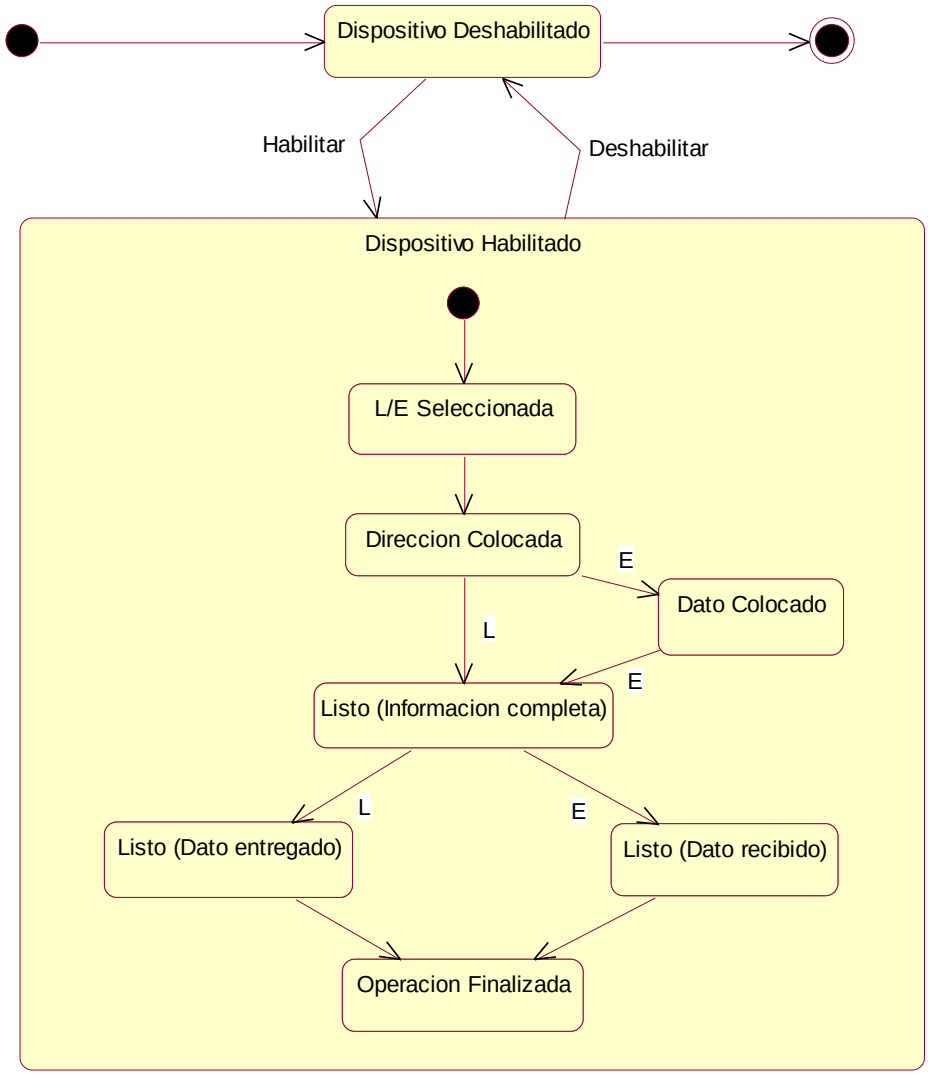
3.1 DIAGRAMA DE CLASES DE DISEÑO



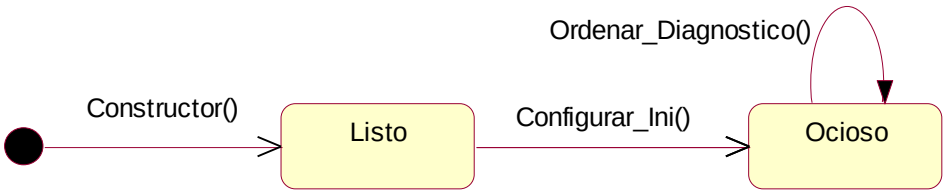
3.2 DIAGRAMAS DE ESTADOS

3.2.1 CONMUTADOR

Comportamiento del handshake para una operación de lectura ó escritura.

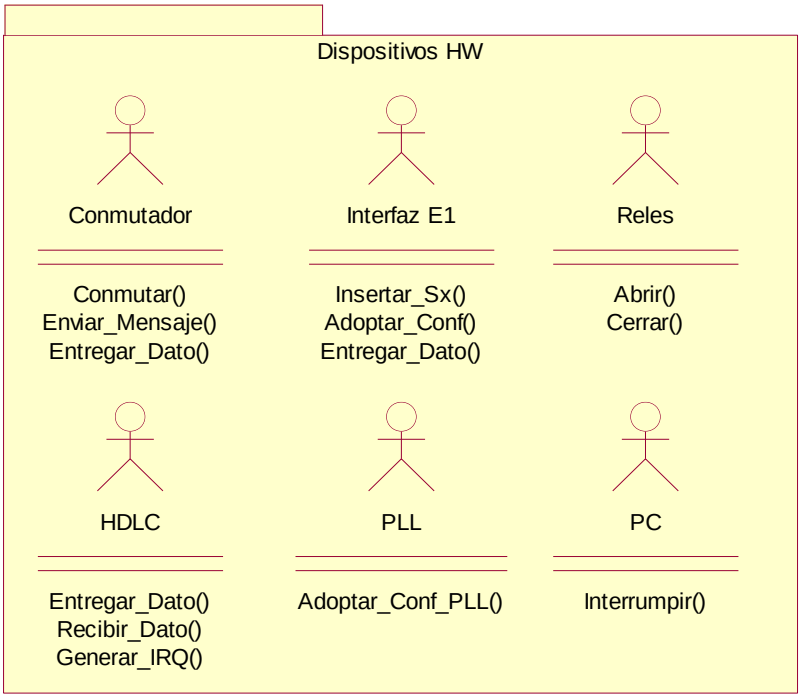


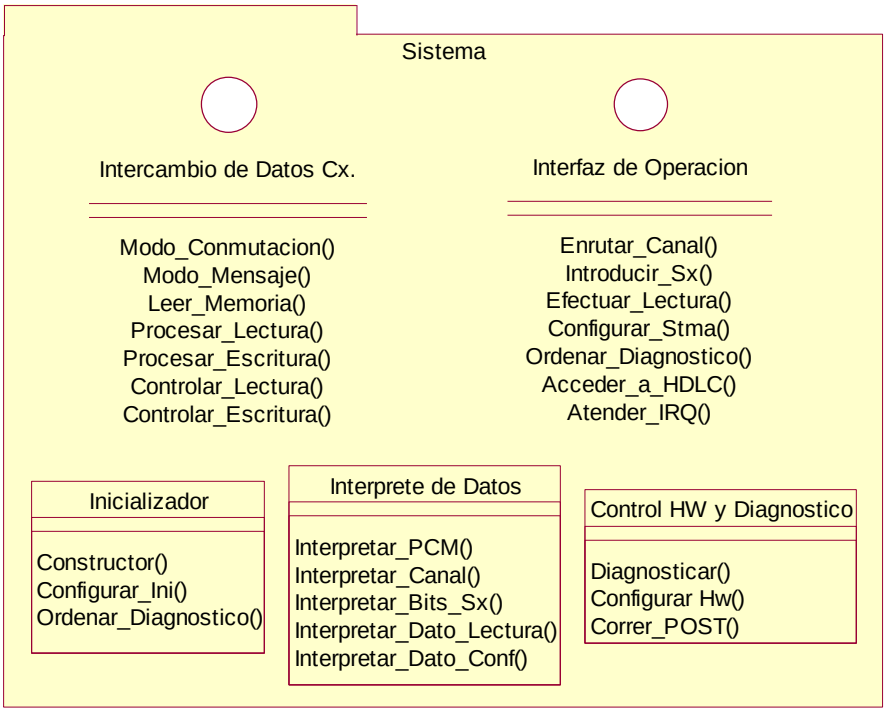
3.2.2 INICIALIZADOR



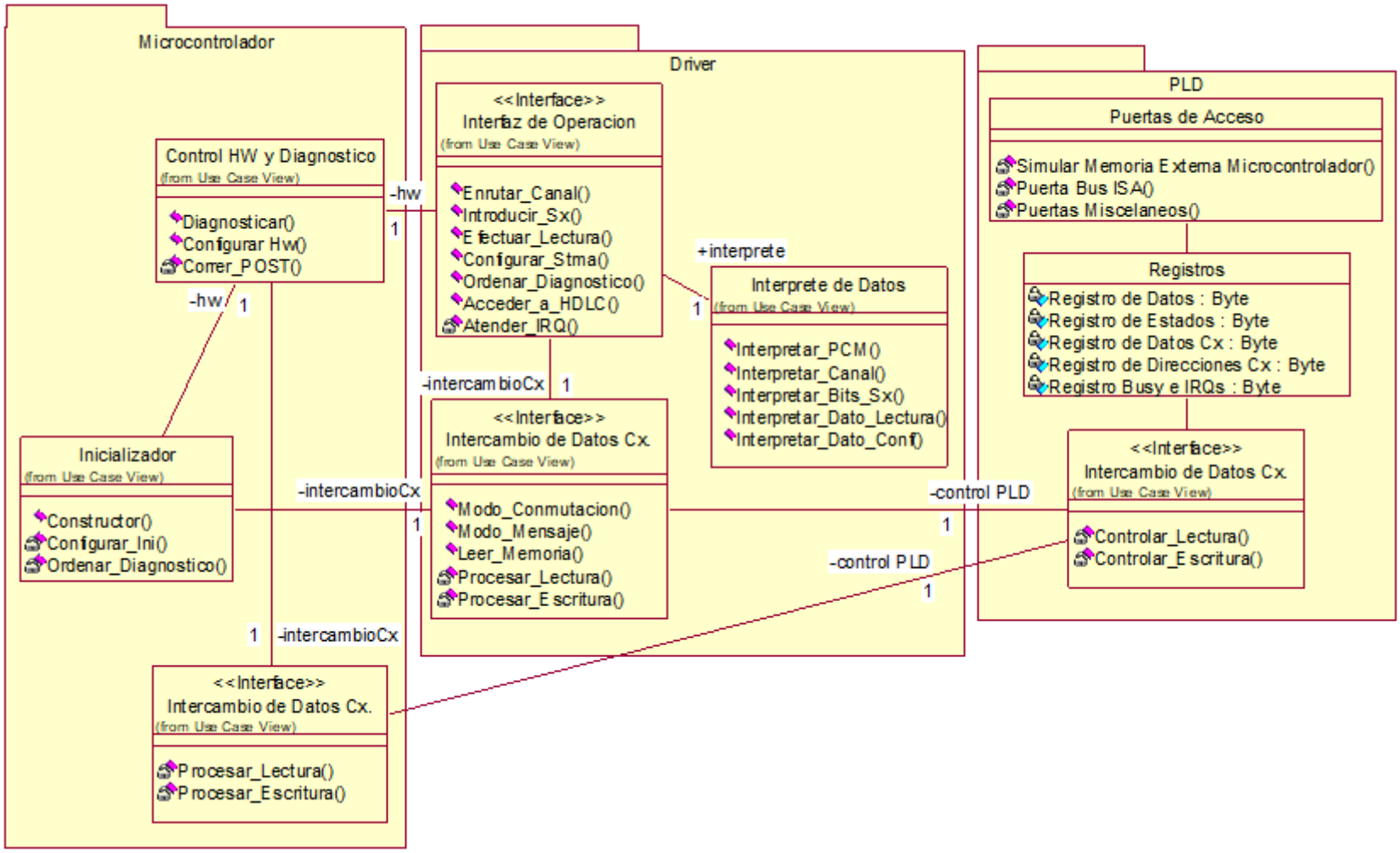
4 DISEÑO DE LA ARQUITECTURA

4.1 DIAGRAMAS DE PAQUETES DE DISEÑO





4.2 DIAGRAMA DE DISTRIBUCION DE PAQUETES Y CLASES



En el diagrama anterior se aprecia cómo están ubicadas las clases dentro de los paquetes microcontrolador, driver y PLD que hacen parte tanto de dispositivos como del sistema. Este diagrama se ha elaborado con el objetivo de profundizar el diagrama de implantación que viene en el siguiente numeral. A continuación se hace la descripción de las clases que no han sido comentadas anteriormente.

4.2.1 CLASE PUERTA DE ACCESO

Clase tipo frontera.

Responsabilidades:

Proveer al sistema de las distintas interfaces entre el PLD y los demás dispositivos, además de dar tratamientos adicionales para algunas señales provenientes estos.

4.2.2 CLASE REGISTROS

Clase tipo entidad.

Sin responsabilidades.

Registros que proporciona el PLD para el sistema.

4.2.2.1 REGISTRO DE DATOS

Se le asigna una dirección Base+Corrimiento (BASE + 0C) físicamente a un registro. Cuando el PC escribe en éste, se debe generar una interrupción hacia el microcontrolador para que lo lea. Es utilizado para que el Driver (a través del bus ISA) entregue parámetros al microcontrolador con la posterior ejecución de alguna de sus funciones, y reciba los valores de retorno de estas.

4.2.2.2 REGISTRO DE ESTADOS

Se le asigna una dirección Base+Corrimiento (BASE + 0D) físicamente a un registro. Cuando el microcontrolador escribe en éste, se debe generar una interrupción hacia el PC para que lo lea. Es utilizado para que el microcontrolador entregue al PC, información de eventos extraordinarios (fallas de diagnósticos) que deben ser atendidos de manera inmediata.

4.2.2.3 REGISTRO DE DATOS CX

Se le asigna una dirección Base+Corrimiento (BASE + 0A) físicamente a un registro. El PC o el microcontrolador pueden escribir en o leer de éste. Es utilizado para guardar los datos que se envían hacia o se reciben del bus de datos del conmutador.

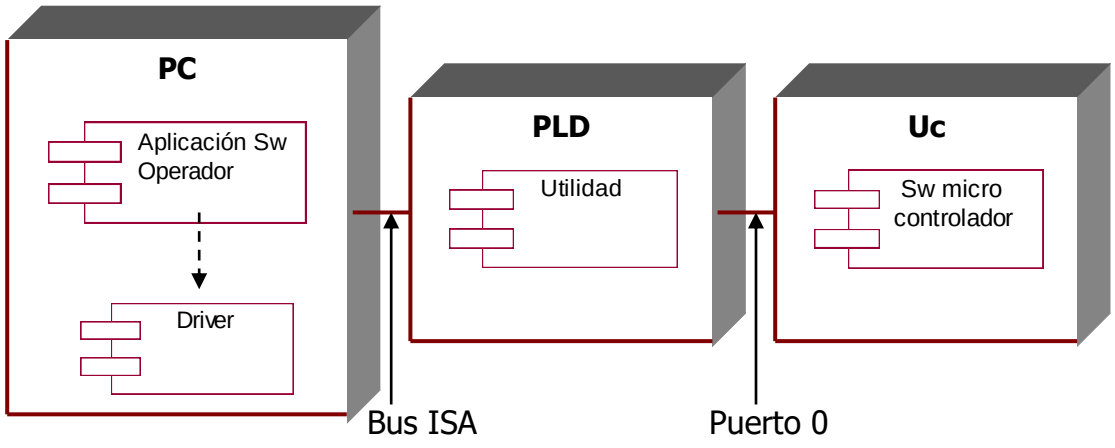
4.2.2.4 REGISTRO DE DIRECCIONES CX

Se le asigna una dirección Base+Corrimiento (BASE + 0B) físicamente a un registro. El PC o el microcontrolador pueden escribir en éste. Es utilizado para guardar las direcciones que se envían hacia el bus de direcciones del conmutado (los 6 Bits LSB); además a través del siguiente Bit (bit 6) determina si se produce un llamado a controlar lectura o escritura y por último da inicio al handshake con el conmutador.

4.2.2.5 REGISTRO BUSY E IRQs

Se le asigna una dirección Base+Corrimiento (BASE + 0E) físicamente a un registro. El PC o el microcontrolador pueden escribir en o leer de éste. Es utilizado para determinar el estado de ocupación de la tarjeta por parte del PC o del microcontrolador; y para generar las interrupciones hacia el microcontrolador o hacia el PC.

4.3 DIAGRAMA DE IMPLANTACION



CAPITULO 3

CAPITULO 4
IMPLEMENTACION DEL SISTEMA

Con base en el sistema TITD modelado bajo la metodología del Proceso Unificado RUP mediante UML, la cual está dirigida hacia el diseño de software intensivo orientado a objetos, se realizó una adaptación de la misma para distribuir adecuadamente el sistema entre las tres entidades o actores hardware que lo constituyen: el PLD, el Microcontrolador y el PC-Driver (como se muestra en el diagrama de distribución de paquetes y clases del capítulo 3). Se aprovechó entonces, la creación de las distintas clases pertenecientes al sistema para identificarlas, clasificarlas, determinar sus propiedades (funcionalidad y atributos) e instanciarlas como objetos independientes relacionados y asociados entre sí, a fin de llevar a cabo tal distribución como sigue a continuación.

1 SUBSISTEMA PLD ALTERA

1.1 DESCRIPCION DEL CODIGO PRODUCIDO, ARCHIVOS GENERADOS, HERRAMIENTAS UTILIZADAS

De la totalidad del sistema TITD modelado una pequeña parte corresponde al PLD de Altera, ya que solamente una de las clases creadas fue trabajada dentro de este dispositivo. Sin embargo, por que según lo anterior se podría llegar a pensar que este integrado ha sido subutilizado, las funciones que cumple el PLD dentro de la TITD no radican únicamente en el sistema mismo; en él se encuentra el hardware referente a la lógica que se encarga tanto de proporcionar la interfaz para establecer la comunicación de los actores inteligentes del sistema (el Microcontrolador y el PC-Driver) con los integrados de la tarjeta especializados en

telecomunicaciones, así como de mantener y organizar el flujo de datos que resulta de la interacción entre todos estos actores, causada por la ejecución de una operación en alguno de ellos.

Esta lógica implementada para el PLD; mostrada en la figura 1 con archivos referentes: utilidad.gdf (código fuente AHDL formato gráfico, ver código en Anexo B numeral 3) y utilidad.scf (simulación, ver Anexo C numeral 3); cuyos componentes se describen abajo, se caracteriza por ser asíncrona, modularizada y con la propiedad de tener un medio compartido constituido por un bus de datos de 8 bits para todos los dispositivos a los cuales interfaza.

La descripción de todos los módulos se realiza especificando las entradas y salidas, las características, la funcionalidad y los archivos asociados a cada componente.

1.1.1 MODULOS DE LOS REGISTROS

1.1.1.1 REGISTRO DE DATOS DEL CONMUTADOR

Tiene 5 líneas de entrada, 1 de ellas de 3 bits; 1 línea de salida; y 2 líneas bidireccionales tri-state cada una de 8 bits.

Este registro consta de 10 flipflops tipo D, 8 para las 2 líneas bidireccionales de 8 bits; 1, con entrada de clear, para guardar el estado de reset de la maquina de estados referente al modulo del handshake con el conmutador; y otro, con entrada de preset, tanto para restringir dicho reset cuando se efectúe una lectura sobre este módulo, como para evitar accesos indeseados al registro ya sea por parte del microcontrolador o por el bus ISA principalmente. Este componente se habilita cuando en la línea de entrada de 3 bits se recibe 0x0H (000B).

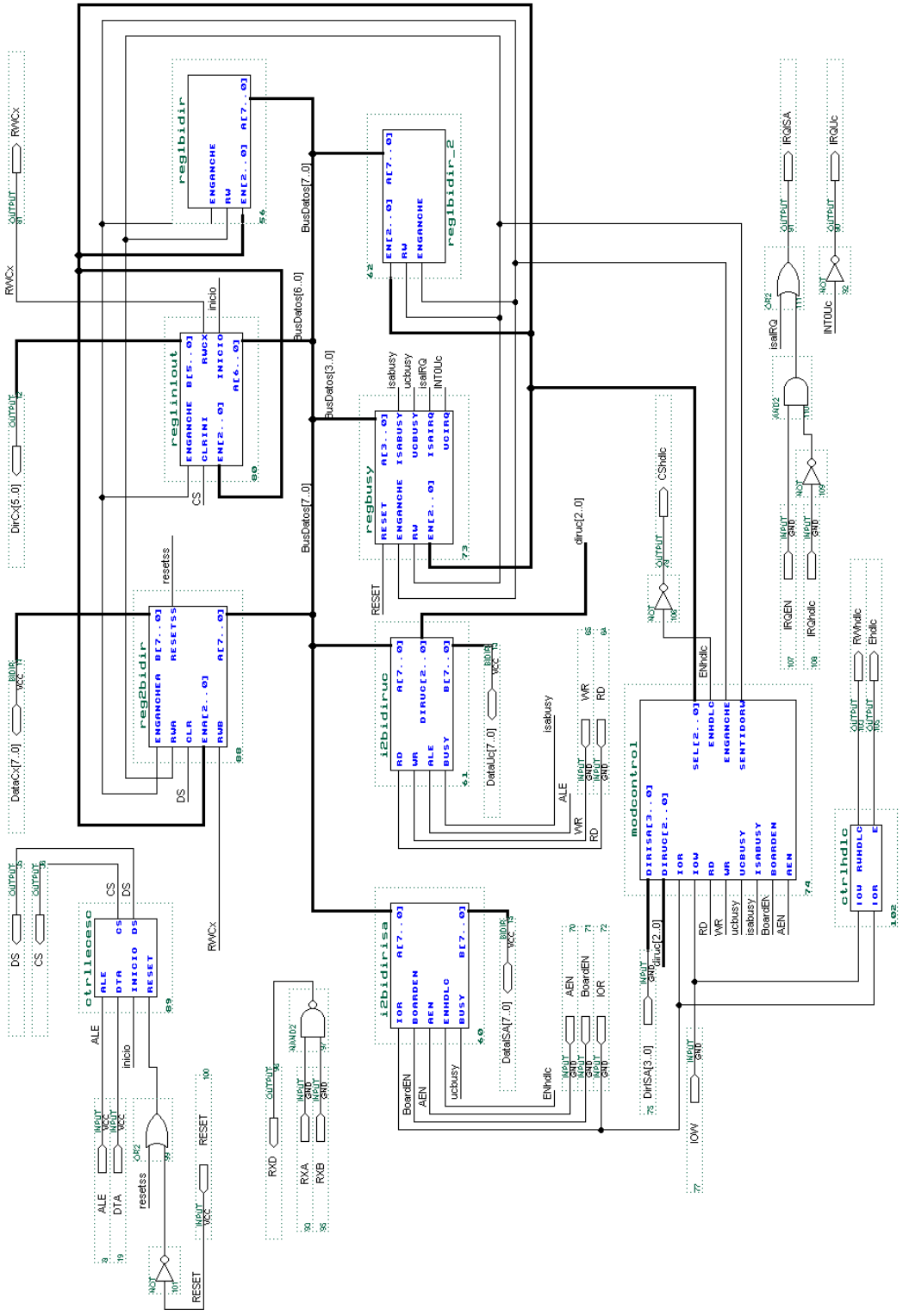


Figura 1. Aplicación de la lógica implementada para el PLD

Sus funciones son:

- Guardar el dato proveniente del microcontrolador o del bus ISA vía bus de datos (medio compartido) para entregárselo al conmutador, a través de sus líneas de datos, cuando éste lo solicite.
- Colocar el dato proveniente del conmutador en el bus de datos para que el microcontrolador o el bus ISA (quien tenga reservado el medio compartido) lo reciba. Nótese que este dato no se guarda en este componente.
- Enviar la señal de reset de la máquina de estados referente al componente de intercambio de datos con el conmutador a fin de finalizar el handshake, cuando se efectúa una lectura sobre este módulo vía bus de datos (microcontrolador o ISA leen) como es el caso de la función anterior.

Los archivos referentes a este módulo son: reg2bidir.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3) y reg2bidir.scf (simulación, ver Anexo C numeral 3).

1.1.1.2 REGISTRO DE DIRECCIONES DEL CONMUTADOR

Tiene 4 líneas de entrada, 1 de ellas de 3 bits y otra de 7 bits provenientes del medio compartido; 3 líneas de salida, 1 de ellas de 6 bits que constituyen el bus de direcciones hacia el conmutador.

Este registro unidireccional consta de 8 flipflops tipo D, 6 para la dirección del conmutador; 1 para guardar el estado del pin R/W del conmutador; y otro, con entrada de clear, para guardar el estado de inicio de la maquina de estados referente al modulo del handshake con el conmutador. Este componente se habilita cuando en la línea de entrada de 3 bits se recibe 0x1H (001B).

Sus funciones son:

- Guardar el dato de la dirección proveniente del microcontrolador o del bus ISA vía bus de datos (medio compartido) para entregárselo al conmutador, a través de sus líneas de direcciones, cuando éste lo solicite.

- Enviar la señal de inicio de la máquina de estados referente al componente de intercambio de datos con el conmutador a fin de iniciar el handshake, cuando se efectúa una escritura sobre este módulo vía bus de datos (microcontrolador o ISA escriben).

Los archivos referentes a este módulo son: reg1in1out.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3) y reg1in1out.scf (simulación, ver Anexo C numeral 3).

1.1.1.3 REGISTRO DE DATOS

Tiene 3 líneas de entrada, 1 de ellas de 3 bits; y 1 línea bidireccional tri-state de 8 bits provenientes del medio compartido.

Este registro consta de 8 flipflops tipo D para la línea bidireccional de 8 bits. Este componente se habilita cuando en la línea de entrada de 3 bits se recibe 0x2H (010B).

Sus funciones son:

- Guardar el dato proveniente del microcontrolador o del bus ISA vía bus de datos (medio compartido).
- Entregar el dato guardado al medio compartido para que el microcontrolador o el bus ISA (quien tenga reservado el medio) lo reciba, cuando alguno de ellos desea leerlo.

Los archivos referentes a este módulo son: reg1bidir.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3) y reg1bidir.scf (simulación, ver Anexo C numeral 3).

1.1.1.4 REGISTRO DE ESTADOS

Este registro posee entradas, salidas, características y comportamiento similares al registro de datos. Este componente se habilita cuando en la línea de entrada de 3 bits se recibe 0x3H (011B).

El archivo referente a este módulo es: reg1bidir_2.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3).

1.1.1.5 REGISTRO DE OCUPACION DEL MEDIO COMPARTIDO

Tiene 4 líneas de entrada, 1 de ellas de 3 bits; 4 líneas de salida; y 1 línea bidireccional tri-state de 4 bits provenientes del medio compartido.

Este registro consta de 4 flipflops tipo D, con entrada de clear, para la línea bidireccional de 4 bits. Este componente se habilita cuando en la linea de entrada de 3 bits se recibe 0x4H (100B).

Sus funciones son:

- Guardar el dato de reservación del medio compartido proveniente del microcontrolador o del bus ISA, según quien haya hecho la petición de reserva, vía bus de datos (medio compartido), para entregárselo a ambos, microcontrolador y bus ISA, con el fin de indicarle a uno que el medio ya ha sido reservado y confirmarle al otro (quien hizo la solicitud y le fue concedida) que él tiene la autorización para ejecutar una operación haciendo uso de dicho medio.
- Guardar el dato de interrupción proveniente del microcontrolador o del bus ISA vía bus de datos (medio compartido), para entregárselo al microcontrolador, si la petición de IRQ fue realizada por medio del bus ISA; o al ISA, si la petición de IRQ fue realizada por el microcontrolador.

El archivo referente a este módulo es: regbusy.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3).

1.1.2 MODULOS DE LAS INTERFACES

1.1.2.1 INTERFAZ CON EL BUS ISA DEL PC

Tiene 5 líneas de entrada; y 2 líneas bidireccionales tri-state cada una de 8 bits.

Esta interfaz está constituida por elementos digitales de lógica combinatoria tales como tres estados, nodos, compuertas OR, NOT, AND y multiplexores, los cuales, en conjunto, conectan y evalúan la habilitación de las líneas bidireccionales en uno u otro sentido, según la operación que se desee efectuar desde el bus ISA.

Su función es:

- Habilitar la conexión entre el bus de datos o medio compartido y las líneas de datos del bus ISA; según la evaluación de la línea AEN de control del ISA, el resultado de la comparación de las líneas de direcciones del ISA con la dirección base (verificando que se desea entablar comunicación con la tarjeta), la línea de habilitación del HDLC (cuyo acceso no se realiza a través del medio compartido) y el indicador de reserva del medio compartido por parte del microcontrolador. El sentido del flujo de los datos está determinado por la línea IOR de control del ISA.

Los archivos referentes a este módulo son: i2bidirisa.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3) y i2bidirisa.scf (simulación, ver Anexo C numeral 3).

1.1.2.2 INTERFAZ CON EL MICROCONTROLADOR

Tiene 4 líneas de entrada; 1 línea de salida de 3 bits; y 2 líneas bidireccionales tri-state cada una de 8 bits.

Esta interfaz está constituida por 3 flipflops tipo D para las líneas de direcciones (los 3 bits menos significativos de la línea bidireccional de 8 bits) provenientes del bus multiplexado de datos/direcciones del microcontrolador; además de elementos digitales de lógica combinatoria tales como nodos, compuertas NAND, NOT, AND y multiplexores, que en conjunto, conectan y evalúan la habilitación de las líneas bidireccionales en uno u otro sentido, según la operación que se desee efectuar desde el microcontrolador.

Sus funciones son:

- Proporcionar la configuración hardware necesaria para simular la memoria de datos durante los ciclos de Acceso a Memoria de datos Externa del microcontrolador, como se muestra en la figura 2.

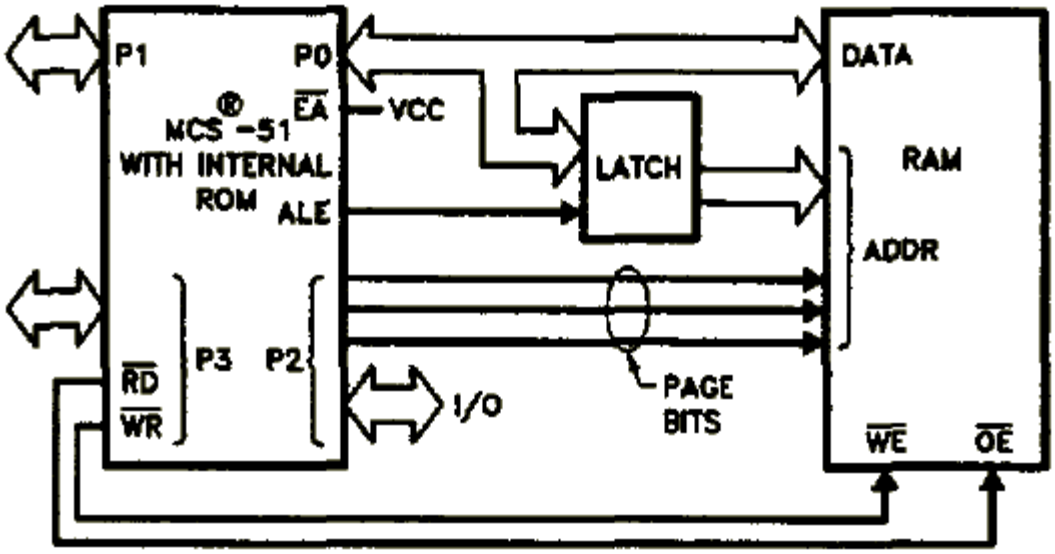


Figura 2. Acceso externo a memoria de datos externa.

En este tipo de acceso, el byte bajo de la dirección es multiplexado en el tiempo con el byte de datos en el puerto 0; la señal ALE se utiliza para capturar el byte de dirección en un cerrojo o latch externo (ubicado en esta interfaz); de tal byte sólo se han tomado 3 bits ya que con ellos es suficiente para direccionar los 5 módulos de registros existentes en el PLD y que constituyen la memoria de datos externa accesada por el microcontrolador. Los bits de dirección (3 bits LSB) son válidos en la transición negativa de la señal ALE, la cual interrumpe su periodicidad en tal transición para dar inicio al ciclo de acceso externo, comenzando con la captura de la dirección, hasta que una de las líneas de control WR o RD (activas bajas) se hayan activado y desactivado. Luego, en un ciclo de escritura, mostrado en la figura 3, el byte de datos a ser escrito aparece en el puerto 0 solamente después de que WR sea activado, y se mantiene allí hasta después de que WR se desactive. En un ciclo de lectura, como se muestra en la figura 4, el byte entrante es aceptado en el puerto 0 inmediatamente después de que la señal RD sea desactivada.

- Habilitar la conexión entre el bus de datos o medio compartido y las líneas del puerto 0 correspondientes al bus multiplexado de datos/direcciones del microcontrolador; según la evaluación de las líneas WR y RD de control del microcontrolador, y el indicador de reserva del medio compartido por parte del bus ISA. El sentido del flujo de los datos está determinado por la línea RD de control del microcontrolador.

Los archivos referentes a este módulo son: i2bidiruc.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3) y i2bidiruc.scf (simulación, ver Anexo C numeral 3).

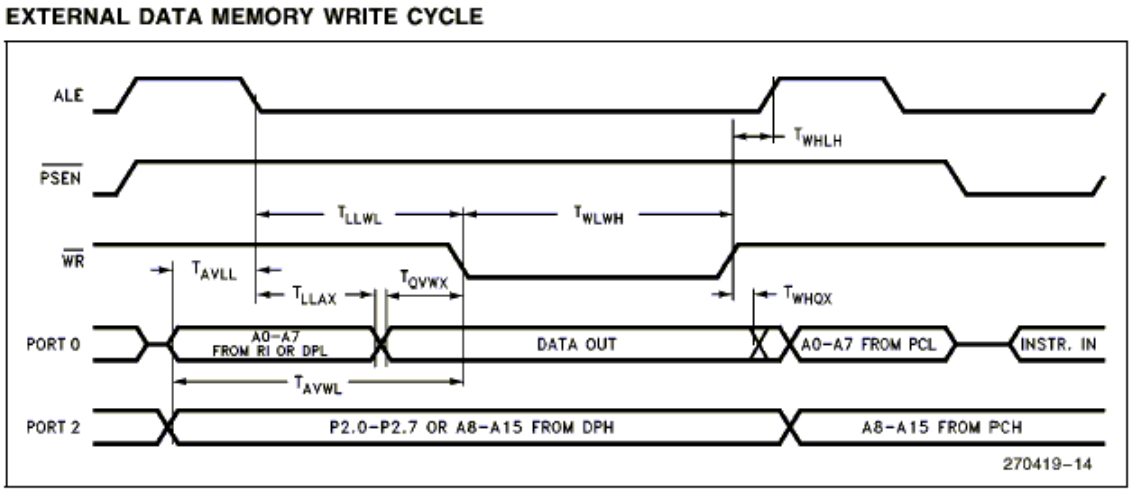


Figura 3. Ciclo de escritura en memoria externa de datos.

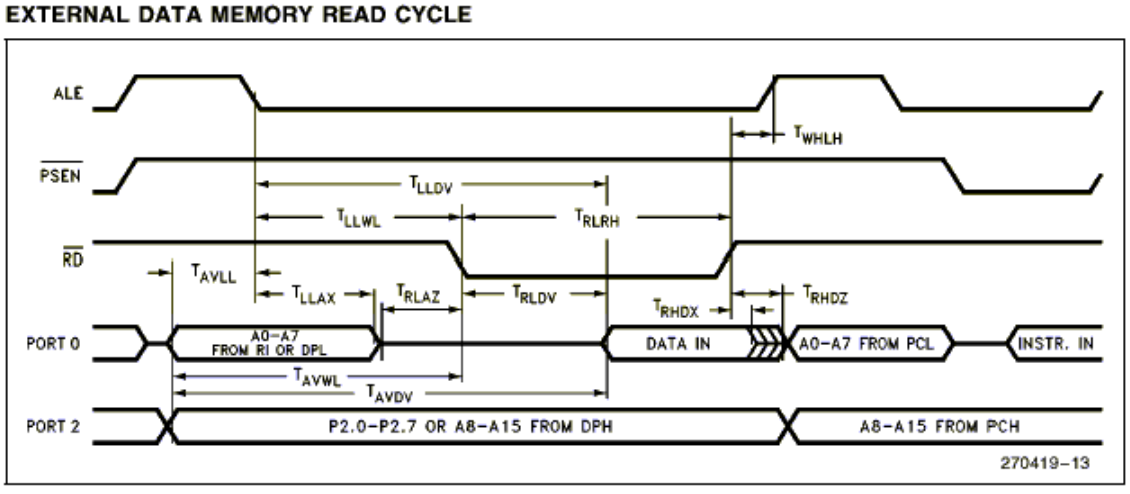


Figura 4. Ciclo de lectura en memoria externa de datos.

1.1.3 MODULOS DE LOS CONTROLES

1.1.3.1 CONTROL DE INTERCAMBIO DE DATOS CON EL CONMUTADOR

Tiene 4 líneas de entrada; y 2 líneas de salida.

Este módulo está constituido por una máquina de estados tipo Moore en donde las salidas son función sólo del estado actual y no de las entradas. Esta máquina se caracteriza por tener 4 estados posibles, determinados por el estado de las líneas de control del conmutador, entre ellos un estado de latencia o de espera, en el cual permanece la máquina de estados la mayor parte del tiempo; además de un reset, que al hacerse activo lleva a la máquina a su estado inicial que difiere del estado latente; y un reloj proporcionado por el microcontrolador a través de la señal periódica ALE cuya frecuencia es de aproximadamente 1/6 de la frecuencia del oscilador.

Su función es:

- Llevar a cabo el Handshake o protocolo de intercambio de datos entre el conmutador y los módulos de los registros de datos y de direcciones del conmutador, mediante la evaluación de las líneas de control de este último. Para dar comienzo al handshake es necesario que primero se encuentre el dato en el registro de datos del conmutador (si se trata de una escritura sobre el conmutador) y luego se escriba la dirección en el registro de direcciones del conmutador, ya que es éste quien activa la línea de inicio de la máquina de estados. Para finalizar el handshake es necesario que se efectúe una lectura sobre el registro de datos del conmutador, ya que es éste quien activa la línea de reset de la máquina de estados.

Los archivos referentes a este módulo son: ctrllecesc.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3) y ctrllecesc.scf (simulación, ver Anexo C numeral 3).

1.1.3.2 CONTROL PRINCIPAL DE LOS REGISTROS

Tiene 10 líneas de entrada, 1 de ellas de 3 bits, otra de 4 bits; y 4 líneas de salida, 1 de ellas de 3 bits.

Este componente está constituido por elementos digitales de lógica combinatoria tales como nodos, compuertas OR, XOR, NOT, AND, NAND, multiplexores, operadores sumadores (para procesos de substracción) y primitivas LCELL (búfer para crear retardos intencionales) quienes, en conjunto, conforman la lógica de control que regula el acceso a todos los módulos de los registros del PLD y al dispositivo HDLC; a partir del análisis de tablas de verdad que evalúan las distintas líneas de control provenientes del microcontrolador y del bus ISA las cuales generan las funciones lógicas que constituyen este módulo.

Sus funciones son:

- Controlar el acceso a todos los registros del PLD, lo que implica el control sobre la habilitación, el enganche de datos y el sentido del flujo de los datos en cada uno de estos registros.
- Habilitar la puerta de acceso a los diferentes registros del dispositivo HDLC.
- Transformar el corrimiento (offset) de la dirección base proveniente del bus ISA en las direcciones correspondientes a los 5 módulos de los registros del PLD, esto es de 0xAH-0xEH del offset a 0x0H-0x4H, ya que las primeras 10 direcciones 0x0H-0x9H del offset corresponden a los diferentes registros del dispositivo HDLC.

Estas funciones se efectúan según la evaluación de las líneas AEN, IOR, IOW de control del ISA, las líneas RD, WR de control del microcontrolador, el resultado de la comparación de las líneas de direcciones del ISA con la dirección base (verificando que se desea entablar comunicación con la tarjeta), los indicadores de reserva del medio compartido del microcontrolador y del ISA, y teniendo en cuenta tanto el offset (línea de entrada de 4 bits) de la dirección base proveniente del ISA, así como la dirección (línea de entrada de 3 bits) proveniente del microcontrolador.

Los archivos referentes a este módulo son: modcontrol.tdf (código fuente AHDL formato texto, ver código en Anexo B numeral 3) y modcontrol.scf (simulación, ver Anexo C numeral 3).

1.1.3.3 CONTROL DEL HDLC

Tiene 2 líneas de entrada; y 2 líneas de salida.

Este componente está constituido por 1 flipflop SR, con entradas de preset y clear ignoradas, y elementos digitales de lógica combinatoria como compuertas NOT, NAND para guardar el estado de la línea R/W del HDLC.

Sus funciones son:

- Guardar y seleccionar el sentido del acceso y del flujo de datos, ya sea para escritura o lectura, del HDLC.
- Determinar el estado de la línea E del dispositivo HDLC que activa el bus de direcciones, la línea R/W y la transferencia de datos en el bus de datos del mismo.

Estas funciones se efectúan según la evaluación de las líneas IOR, IOW de control del bus ISA.

Los archivos referentes a este módulo son: ctrlhdlc.gdf (código fuente AHDL formato gráfico, ver código en Anexo B numeral 3) y ctrlhdlc.scf (simulación, ver Anexo C numeral 3).

1.1.4 BUSES

El bus de datos del PLD es un medio compartido constituido por un bus bidireccional de 8 bits de alta velocidad que se encuentra conectado con todos los módulos de los registros, con la interfaz con el bus ISA del PC y con la interfaz con el microcontrolador. La distribución de sus conexiones es la siguiente:

- Registro de datos del conmutador: 8 bits bidireccionales.
- Registro de direcciones del conmutador: 7 bits unidireccionales (escritura sobre el registro).

- Registro de datos: 8 bits bidireccionales.
- Registro de estados: 8 bits bidireccionales.
- Registro de Ocupación: 4 bits bidireccionales.
- Interfaz con el Bus ISA del PC: 8 bits bidireccionales.
- Interfaz con el Microcontrolador: 8 bits bidireccionales.

1.1.5 MODULOS ADICIONALES O MISCELANEOS

1.1.5.1 LOGICA RESET

La lógica implementada para el RESET consiste de 2 compuertas, NOT y OR. El reset de la tarjeta, activo alto, proveniente del bus ISA pasa por un negador (CI 74LS04 en la tarjeta) antes de ingresar como entrada al PLD. Ya en el PLD, esta entrada de reset es utilizada para resetear la máquina de estados del módulo de la interfaz de intercambio de datos con el conmutador. Como el reset de una máquina de estados también es activo alto, se hace necesario que la entrada de reset del PLD pase por una compuerta NOT, recuperando de esta manera su estado por defecto.

Además, se necesita una compuerta OR ya que la salida resetss del registro de datos del conmutador también puede resetear la máquina de estados cuando se desea finalizar el handshake.

1.1.5.2 LOGICA DISPOSITIVO INTERFAZ E1

Hacia el lado de recepción de la Interfaz E1 se tienen las salidas -RXA y -RXB que transforman la señal bipolar CEPT a un formato unipolar que se sugiere, deben combinarse como entradas de una compuerta NAND para así obtener la entrada correspondiente al dato recibido RXD. Ya en el PLD se tiene entonces las entradas RXA y RXB como entradas de una NAND cuya salida se asocia a RXD.

1.1.5.3 LOGICA DE INTERRUPCIONES

Esta lógica consiste de compuertas NOT, AND y OR. La interrupción externa INT0 hacia el microcontrolador es una entrada activa baja y debido a que dentro del PLD

se manifiesta con un estado alto, es necesario pasarla por un negador antes de establecerla como pin de salida. Hacia el bus ISA se puede generar interrupciones activas altas tanto desde el microcontrolador como desde el HDLC justificándose la compuerta OR entre ellas; desde el primero, activa alta y desde el segundo, activa baja (esta última se hace pasar por un negador). Además la IRQ del HDLC consta de un habilitador activo alto que se combina con la misma IRQ mediante una compuerta AND para permitir tal petición.

La herramienta utilizada para la implementación del subsistema TITD para el PLD es MAX PLUS II 9.23 Baseline de Altera versión académica. De ella, básicamente se usaron las aplicaciones disponibles para los editores gráficos y de texto (códigos fuente), el editor de forma de onda (simulación), el editor de plano y distribución (asignación de pines) el compilador (síntesis) y el programador (programación del dispositivo), entre otros.

1.2 DESCRIPCION DE LAS CLASES

Solamente una de las clases modeladas fue trabajada dentro del dispositivo PLD y no en su totalidad. Esta clase es Intercambio de Datos Cx, de cuyas funciones se implementaron Controlar_Lectura y Controlar_Escritura, las cuales ejecutan el proceso físico de lectura o escritura respectivamente sobre el registro de control o una zona de memoria del conmutador teniendo en cuenta los cambios de estado de las líneas de control del mismo para que así se lleve a cabo el handshake.

El componente directamente involucrado en la ejecución de estas funciones es ctrlecesc o Control de Intercambio de Datos con el Conmutador soportado por los registros de datos (reg2bidir) y direcciones (reg1in1out) del conmutador.

Con el objeto de mostrar las demás funcionalidades del PLD se adicionaron las clases Registros y Puertas de Acceso, como se muestra en el diagrama de distribución de paquetes y clases del capítulo 3. Cabe anotar que estas clases no hacen parte del sistema modelado, pero se han colocado sólo para dimensionar la consecuente implementación en el PLD.

1.3 DIFICULTADES EN LA IMPLEMENTACION

En este apartado se han querido resaltar los aspectos de implementación del sistema TITD PLD de mayor complejidad, ya sea por haberse analizado y confrontado por primera vez o por las limitaciones forzadas de la arquitectura de la tarjeta.

Como primera instancia, el manejo de los pines o líneas bidireccionales (bidir) es de cuidado ya que una línea equivale a una entrada y a una salida independientes pero nombradas de igual manera, sin embargo, lo que se ve a la entrada siempre se refleja en la salida y es aquí donde se debe realizar una adecuada implementación para evitar problemas con pérdida de datos, datos malinterpretados o irreconocibles o cortos; estos casos suelen notarse claramente durante la simulación. Es por eso que a una línea bidireccional se le suele asociar un tres estados conectando la salida de este último a la línea (entonces automáticamente el bidir asociará tal conexión a su salida) o conectando la entrada del tri-state a la línea (donde automáticamente el bidir asociará tal conexión a su entrada). De esta forma cuando por ejemplo, se tenga una salida de un dato por el bidir, la entrada deberá mantenerse en estado de alta impedancia para así evitar el reflejo de otro posible dato presente en ese instante y no causar el daño del dato inicial; por el contrario, si se tiene una entrada de un dato por el bidir, la salida deberá mantenerse en estado de alta impedancia para así evitar un reflejo indeseado de tal dato en la salida ya que se podría causar un corto digital por la utilización de la línea en ambos sentidos. Esta situación se resuelve colocando un bajo en el output enable del tri-state, el cual pondría su salida en alta impedancia causando la misma condición en el bidir.

Por otra parte, implementar medios compartidos tales como buses puede representar una ventaja si se tiene que accesar a los mismos dispositivos desde entes independientes y separados físicamente entre sí, ya que se acierta en cuanto a un ahorro de área de implementación (para el caso particular de un PLD, las macroceldas). Sin embargo, se debe tener en cuenta que tal medio debe ajustarse

obligatoriamente a un procedimiento de priorización para su ocupación y posterior liberación; así, si uno de los entes se encuentra haciendo uso del medio en un instante determinado, no se debe permitir que otro pueda llegar a acceder al mismo medio mientras éste no sea liberado; obviamente el acceso de parte y parte depararía en un corto digital que terminaría por causar un daño tanto en los entes, el medio y los dispositivos accesados conectados a éste.

Otro inconveniente que hace realmente tediosa la implementación de una aplicación usando PLDs, cuando ella involucra interfaces y registros o memorias, es la falta de sincronismo. La mayoría de los PLDs son dispositivos diseñados para soportar aplicaciones síncronas en donde se tiene un reloj maestro global que recorre todas las macroceldas de todos los LABs que conforman el integrado y que fácilmente optimiza la ejecución de la aplicación. En el caso particular del sistema TITD, se optó por implementar un subsistema asíncrono debido a la discrepancia entre los relojes suministrados y la subutilización de los mismos. Es decir que, las señales de reloj tanto del bus ISA como del microcontrolador y del PLL no sólo difieren en cuanto a frecuencia entre sí, sino que además una de ellas, el CLK del bus ISA, no se tuvo en cuenta como señal de entrada hacia la tarjeta; razón por la cual se hizo prácticamente imposible la elaboración de una aplicación sincrónica.

2 SUBSISTEMA MICROCONTROLADOR

2.1 DESCRIPCION DEL CODIGO PRODUCIDO, ARCHIVOS GENERADOS, HERRAMIENTAS UTILIZADAS

En el microcontrolador se encuentra el software encargado de dejar la tarjeta a punto, en funcionamiento, bajo condiciones primarias o iniciales, disponible y lista para ser utilizada. Por otra parte también se encarga de diagnosticar la tarjeta con el fin de verificar e informar que todos los dispositivos que la constituyen funcionan adecuadamente. Para hacer esto, establece una comunicación indirecta con los dispositivos especializados en telecomunicaciones, a través del PLD, y una

comunicación directa con otros dispositivos (PLL, Cx pin ODE) y elementos hardware (Relés, Led) que hacen parte de la tarjeta.

Este software implementado para el microcontrolador, con el archivo referente: sw_uc.asm (ver código en Anexo B numeral 4), cuyas funciones se describen abajo, se caracteriza por ser modularizado y secuencial. Así, en primer lugar se ejecuta una rutina de POST o autodiagnóstico, seguido por una inicialización y un diagnóstico de los dispositivos de la tarjeta. Entonces la tarjeta queda dispuesta para ser utilizada y mientras tanto el microcontrolador permanece en espera de una orden del operador al mismo tiempo que realiza un diagnóstico de manera periódica.

2.1.1 FUNCION DE POST (POST)

Esta función ejecuta un diagnóstico sobre el mismo microcontrolador para verificar que su hardware constituyente no presenta ninguna anomalía y asegurar que su funcionamiento es correcto.

Este autodiagnóstico básicamente escribe un dato a lo largo de toda la memoria del microcontrolador encadenadamente, para que al final verifique el estado de la memoria comparando el dato entre la primera y última localidad de ésta.

2.1.2 FUNCION DE RESERVA (RESERV)

Esta rutina se encarga de reservar el medio compartido (Bus de datos) del PLD para que el microcontrolador pueda efectuar una operación que involucre una comunicación indirecta con los integrados de telecomunicaciones o con el bus ISA.

Mediante un acceso a memoria externa, escribe la palabra de reserva en el registro de ocupación del PLD (regbusy); después lee este registro para verificar si la palabra escrita es igual a la palabra leída. En caso de serlo confirma que la petición de reserva ha sido concedida, de lo contrario, el medio ya ha sido reservado, permanece intentando la solicitud hasta que se le conceda.

2.1.3 FUNCION PROCESO ESCRITURA (PRESC)

Esta función se encarga de armar las palabras de programación del conmutador para escritura cuando el microcontrolador desea establecer una comunicación indirecta con el conmutador a través del PLD, y de enviarlas mediante acceso a memoria externa a los registros de datos y direcciones del conmutador.

Recibe los datos correspondientes a PCM, IT, Modo y Mensaje necesarios para conformar las 4 secuencias de programación requeridas en este proceso.

2.1.4 FUNCION PROCESO LECTURA (PRLEC)

Esta función se encarga de armar las palabras de programación del conmutador para lectura cuando el microcontrolador desea establecer una comunicación indirecta con el conmutador a través del PLD, enviarlas mediante acceso a memoria externa a los registros de datos y direcciones del conmutador y recibir el dato a leer.

Recibe los datos correspondientes a PCM, IT y Memoria necesarios para conformar las 2 secuencias de programación requeridas en este proceso y retorna el dato leído.

2.1.5 FUNCION DE INICIALIZACION (INIC)

Conociendo la arquitectura y las características de entramado PCM de la tarjeta, esta rutina envía los parámetros referentes a la configuración inicial de la misma por medio de llamados a procesos de escritura.

2.1.6 FUNCION DIAGNOSTICO FUERA DE FUNCIONAMIENTO (DIAGFF)

Esta función efectúa un diagnóstico general de los dispositivos de la tarjeta dejándola fuera de funcionamiento y con su configuración inicial.

Este diagnóstico se lleva a cabo mediante la ejecución de 4 pruebas: Prueba digital de señalización, Prueba digital de canales, Prueba digital de conmutación de canales y Prueba análoga de canal y señalización.

Reporta mediante procesos de escritura el resultado de las pruebas, si falla o no, en el registro de estados; y el dato de la falla en el registro de datos.

2.1.7 FUNCION DIAGNOSTICO REGULAR (DIAGRG)

Esta función efectúa un diagnóstico de los integrados de telecomunicaciones de la tarjeta permitiendo que esta continúe en funcionamiento y con su configuración actual.

Este diagnóstico se lleva a cabo mediante la ejecución de 1 prueba: Prueba digital de Canal.

Recibe el canal sobre el cual se va a realizar la prueba perdiendo en este su configuración anterior.

Si falla se reporta en el registro de estados y se genera una interrupción hacia el bus ISA.

2.1.8 FUNCION DIAGNOSTICO PERIODICO (DIAGPE)

Esta función verifica que la tarjeta se encuentre configurada apropiadamente permitiendo que esta continúe en funcionamiento y con su configuración actual.

Este diagnóstico se lleva a cabo mediante la ejecución de 2 pruebas: Verificación de constantes de configuración y Chequeo de parámetros de prueba de enlace. La ejecución de esta última prueba es configurable por el operador.

Si falla, se genera una interrupción hacia el bus ISA y reporta el resultado de las pruebas en el registro de estados y el estado del enlace en el registro de datos.

2.1.9 FUNCION DE ATENCION A INTERRUPCION EXTERNA (ATENINT)

Una vez que se ha generado una interrupción externa hacia el microcontrolador, esta rutina recibe una orden a través del registro de datos del PLD proveniente del operador vía bus ISA. Según la orden, efectúa el llamado a la rutina que la cumple. Las órdenes pueden ser: Efectuar Diagnóstico Regular, Efectuar Diagnóstico Fuera de funcionamiento y Configurar Hardware.

2.1.10 FUNCION DE ATENCION A INTERRUPCION DE TEMPORIZADOR(TEMPO)

Esta función se encarga de hacer parpadear el Led con una base de tiempo que depende del estado funcional actual de la tarjeta, además es la encargada de llamar al diagnóstico periódico, según una base de tiempo predeterminada, si éste está habilitado.

El lenguaje utilizado para la implementación del subsistema TITD para el microcontrolador es Ensamblador (para familia Intel 8051) con ayuda de la herramienta Avocet para DOS, de la cual se usaron las aplicaciones avsim51 (simulación y depuración de código de máquina), avmac51 y avlink (compilación y enlace para la generación de los archivos obj y hex). Además se utilizó el software programador EXPRO para DOS (programación del integrado microcontrolador).

2.2 DESCRIPCION DE LAS CLASES

De las clases modeladas, se implementaron Inicializador, Control Hw y Diagnóstico y parte de Intercambio de Datos Cx, en el software del microcontrolador.

La correspondencia entre las operaciones de las clases y las funciones del software del microcontrolador es la siguiente:

Clase Inicializador:

Configurar_Ini()	-->	Inicialización (INIC)
Ordenar_Diagnostico()	-->	Atención a interrupción de temporizador(TEMPO)--Llamado a DIAGPE

Clase Control Hw y Diagnostico:

Diagnosticar()	-->	Diagnóstico Fuera de Funcionamiento (DIAGFF) Diagnóstico Regular (DIAGRG) Diagnóstico Periódico (DIAGPE)
Configurar_Hw()	-->	Atención a interrupción externa (ATENINT)--ORDEN2
Correr_POST()	-->	POST (POST)

Clase Intercambio de datos Cx:

Procesar_Escritura()	-->	Proceso Escritura (PRESC)
Procesar_Lectura()	-->	Proceso Lectura (PRLEC)

3 SUBSISTEMA DRIVER

3.1 DESCRIPCION DE CLASES, CODIGO PRODUCIDO, ARCHIVOS GENERADOS, HERRAMIENTAS UTILIZADAS

En el PC se encuentra el Driver como software o programa residente en memoria encargado de controlar el dispositivo hardware correspondiente a la tarjeta TITD. Por otra parte también sirve de puente o interfaz entre la tarjeta y el software operador.

El subsistema del driver está basado en DLLs. Una DLL (Dynamic Link Library) es un archivo conectado dinámicamente, una librería que almacena funciones ejecutables o datos de un dispositivo (en este caso la TITD) que pueden ser usados por una aplicación Windows.

Para lograr que el driver se comporte adecuadamente, se ha utilizado la ayuda de la herramienta DriverX de Tetradyne cuyas aplicaciones y componentes conforman un programa que es el responsable de la activación de un dispositivo determinado, la TITD, en el entorno Windows conocido como dispositivo virtual.

De esta manera se consigue una jerarquía de niveles o capas de componentes dependientes unos de otros, en donde la capa inferior será aquella que se encuentra más íntimamente relacionada con el hardware dentro del entorno Windows. Este componente perteneciente al nivel inferior es utilizado por otro(s) componente(s) que pertenece(n) al nivel inmediatamente superior, originando así la relación de dependencia entre ellos, hasta llegar al componente de la capa más superior que consiste, por ejemplo, en una aplicación Windows correspondiente al software operador del dispositivo hardware.

De las clases modeladas, en diseño, se implementaron en el software del driver: Interfaz de Operacion, Interprete de Datos e Intercambio de Datos Cx (no en su totalidad). En cuanto a la instanciación de estas clases, sólo para la clase Interfaz de Operacion se creó el objeto interfaz; sin embargo, este objeto tiene los objetos atributos interprete del tipo Interprete de Datos e intercambio del tipo Intercambio

de Datos Cx. Dentro de la clases agregadas se tiene BD que es una estructura, o sea una clase con atributos pero sin funciones.

Este software implementado para el driver, con los archivos referentes: drvtitd.cpp, drvtitd.def, drvtitd.h y libtitd.h (ver códigos en Anexo B numeral 5), cuyas clases se describen abajo, se caracteriza por ser Orientado a Objetos.

3.1.1 CLASE INTERPRETE_DE_DATOS

3.1.1.1 PROPOSITO

El propósito de esta clase del tipo entidad ha sido el de convertir o interpretar la información suministrada por el operador (mediante la aplicación del software operador) en datos equivalentes comprensibles por el driver, basándose en la arquitectura y características de entramado PCM de la tarjeta.

Su modificación con respecto al diseño fue la adición de la función constructor Interprete_de_Datos() y la adición del atributo entramado.

3.1.1.2 ATRIBUTOS

entramado	Objeto atributo privado del tipo BD, un arreglo de 20 estructuras. Es la tabla de referencia o pequeña base de datos del intérprete.

3.1.1.3 FUNCIONES

Interpretar_PCM()

Esta función pública convierte el dato del PCM suministrado por el operador en su equivalente comprensible por el subsistema (STbus de la tarjeta).

Recibe los parámetros E_S (entrada/salida) del tipo boleano y PCM del tipo byte.

Retorna un valor del tipo byte.

Su modificación con respecto al diseño fue la substracción del parámetro Sentido.

Interpretar_Canal()

Esta función pública convierte el dato del Canal suministrado por el operador en su equivalente comprensible por el subsistema (IT de un PCM).

Recibe los parámetros PCM del tipo byte y Canal del tipo byte.

Retorna un valor del tipo byte.

Su modificación con respecto al diseño fue la substracción del parámetro Sentido y la adición del parámetro PCM.

Interpretar_Bits_Sx()

Esta función pública convierte el dato de los bits de señalización suministrados por el operador en su equivalente comprensible por el subsistema (Palabra de Sx).

Recibe los parámetros A, B, C, D del tipo boleano.

Retorna un valor del tipo byte.

No tiene modificación alguna con respecto al diseño.

Interpretar_Dato_Lectura()

Esta función pública convierte el dato del parámetro a leer suministrado por el operador en su equivalente comprensible por el subsistema.

Recibe el parámetro Nombre_Dato del tipo char* (puntero a cadena de caracteres).

Retorna un valor del tipo word (16 bits).

Su modificación con respecto al diseño fue el cambio del tipo de retorno de byte a word.

Interpretar_Dato_Conf()

Esta función pública convierte el dato del parámetro a configurar suministrado por el operador en su equivalente comprensible por el subsistema.

Recibe el parámetro Nombre_Dato_Conf del tipo char* (puntero a cadena de caracteres).

Retorna un valor del tipo word (16 bits).

Su modificación con respecto al diseño fue el cambio del tipo de retorno de byte a word.

Interprete_de_Datos()

Esta función pública es el constructor de la clase y se encarga de inicializar la tabla de referencia (entramado) del intérprete con las palabras clave (kword) y su respectivo valor de retorno.

No recibe parámetros.

No tiene valor de retorno.

3.1.2 CLASE INTERCAMBIO_DE_DATOS_CX

3.1.2.1 PROPOSITO

El propósito de esta clase del tipo frontera ha sido el de servir de interfaz entre el subsistema driver y el conmutador llevando a cabo el handshake, a fin de establecer y mantener la comunicación que permite el intercambio de datos entre ellos.

Su modificación con respecto al diseño fue la adición de las funciones Reservar() y Liberar().

3.1.2.2 ATRIBUTOS

No tiene atributos.

3.1.2.3 FUNCIONES

Modo_Conmutacion()

Esta función pública recibe la información pertinente para la programación del conmutador bajo el modo de conmutación.

Recibe los parámetros PCMe, PCMs, ITe e ITs del tipo byte (e:entrada, s:salida).

No tiene valor de retorno.

No tiene modificación alguna con respecto al diseño.

Modo_Mensaje()

Esta función pública recibe la información pertinente para la programación del conmutador bajo el modo de mensaje.

Recibe los parámetros PCM, IT y Mensaje del tipo byte.

No tiene valor de retorno.

No tiene modificación alguna con respecto al diseño.

Leer_Memoria()

Esta función pública recibe la información pertinente para la lectura de las distintas memorias existentes en el conmutador las cuales se encuentran dispuestas como una matriz CEPT de 256 bytes (8PCM * 32IT).

Recibe los parámetros PCM, IT y Tipo_Memoria del tipo byte.

Retorna un valor del tipo byte.

No tiene modificación alguna con respecto al diseño.

Reservar()

Esta función pública se encarga de reservar el medio compartido (Bus de datos) del PLD para que, por medio del bus ISA, el driver pueda efectuar una operación que involucre una comunicación, a través del PLD, con los integrados de telecomunicaciones o el microcontrolador.

No recibe parámetros.

No tiene valor de retorno.

Liberar()

Esta función pública se encarga de liberar el medio compartido (Bus de datos) del PLD después de haber finalizado una operación que haya involucrado una comunicación, a través del PLD, con los integrados de telecomunicaciones o el microcontrolador. Si tal operación debe acarrear una interrupción (hacia el microcontrolador), esta función la genera.

Recibe el parámetro IRQuc del tipo boleano.

No tiene valor de retorno.

Procesar_Lectura()

Esta función privada se encarga de armar las palabras de programación del conmutador para lectura cuando el driver, vía bus ISA, desea establecer una comunicación, a través del PLD, con el conmutador.

Recibe los parámetros PCM, IT y Tipo_Memoria del tipo byte, necesarios para conformar las 2 secuencias de programación requeridas en este proceso.

Retorna un valor del tipo byte.

No tiene modificación alguna con respecto al diseño.

Procesar_Escritura()

Esta función privada se encarga de armar las palabras de programación del conmutador para escritura cuando el driver, vía bus ISA, desea establecer una comunicación, a través del PLD, con el conmutador.

Recibe los parámetros PCMs, ITs del tipo byte (s:salida), Modo del tipo boleano y Mensaje del tipo byte, necesarios para conformar las 4 secuencias de programación requeridas en este proceso.

No tiene valor de retorno.

No tiene modificación alguna con respecto al diseño.

3.1.3 CLASE INTERFAZ_DE_OPERACION

3.1.3.1 PROPOSITO

El propósito de esta clase del tipo frontera ha sido el de servir de interfaz entre el operador, por intermedio de la aplicación del software operador, y el subsistema driver, proporcionando al primero las funciones que ofrece éste último. Tales funciones son las funciones básicas que ofrece la TITD.

Su modificación con respecto al diseño fue la adición de la función Introducir_Dato() y el cambio de condición de la función Atender_IRQ de privada a pública.

3.1.3.2 ATRIBUTOS

interprete	Objeto atributo privado del tipo Interprete_de_Datos.
intercambio	Objeto atributo privado del tipo Intercambio_de_Datos_Cx.
PCM_Eq	Atributo privado del tipo byte. PCM equivalente que devuelve el intérprete.
IT_Eq	Atributo privado del tipo byte. IT equivalente que devuelve el intérprete.
Bits_Sx_Eq	Atributo privado del tipo byte. Palabra de Sx equivalente que devuelve el intérprete.
Lectura_Eq	Atributo privado del tipo word. Lectura equivalente que devuelve el intérprete.
Conf_Eq	Atributo privado del tipo byte. Configuración equivalente que devuelve el intérprete.

3.1.3.3 FUNCIONES

Enrutar_Canal()

Esta función pública recibe la información correspondiente a un enrutamiento directamente de la aplicación del software operador para indicarle al subsistema driver que se va a llevar a cabo este procedimiento.

Recibe los parámetros PCMe, PCMs, Canal_e y Canal_s del tipo byte (e:entrada, s:salida).

No tiene valor de retorno.

No tiene modificación alguna con respecto al diseño.

Introducir_Sx()

Esta función pública recibe la información relacionada con la señalización directamente de la aplicación del software operador para indicarle al subsistema driver que se va a señalizar un canal determinado.

Recibe los parámetros Canal del tipo byte, A, B, C y D del tipo boleano.

No tiene valor de retorno.

Su modificación con respecto al diseño fue la substracción del parámetro Bits_Sx y la adición de los parámetros A, B, C y D.

Efectuar_Lectura()

Esta función pública recibe la información correspondiente a una lectura, la palabra clave o keyword y los valores asociados, directamente de la aplicación del software operador para indicarle al subsistema driver que se va a llevar a cabo este procedimiento; además entrega al software operador el dato leído.

Recibe los parámetros Nombre_Dato del tipo char* (puntero a cadena de caracteres), Valor1 y Valor2 del tipo byte.

Retorna un valor del tipo byte.

Su modificación con respecto al diseño fue la substracción del parámetro Valor (word) la adición de los parámetros Valor1 y Valor2.

Configurar_Stma()

Esta función pública recibe la información correspondiente a una configuración, la palabra clave o keyword y los valores asociados, directamente de la aplicación del software operador para indicarle al subsistema driver que se va a llevar a cabo este procedimiento.

Recibe los parámetros Nombre_Dato_Conf del tipo char* (puntero a cadena de caracteres), Valor1 y Valor2 del tipo byte.

No tiene valor de retorno.

Su modificación con respecto al diseño fue la substracción del parámetro Valor (word) la adición de los parámetros Valor1 y Valor2.

Ordenar_Diagnostico()

Esta función pública recibe la información pertinente a un diagnóstico directamente de la aplicación del software operador para indicarle al subsistema driver que se va a ordenar la ejecución de un diagnóstico específico; además entrega al software operador el resultado del diagnóstico efectuado.

Recibe los parámetros Tipo del tipo boleano y Canal del tipo byte.

Retorna un valor del tipo word.

Su modificación con respecto al diseño fue la adición de los parámetros Tipo y Canal.

Acceder_a_HDLC()

Esta función pública recibe la información referida al acceso sobre el dispositivo HDLC directamente de la aplicación del software operador para indicarle al subsistema driver que se va a llevar a cabo este procedimiento, además entrega al software operador el dato que resulta del acceso en el caso de lectura.

Recibe los parámetros L_E (lectura/escritura) del tipo boleano, Direccion y Dato del tipo byte.

Retorna un valor del tipo byte (el dato en lectura y 0 en escritura).

No tiene modificación alguna con respecto al diseño.

Atender_IRQ()

Esta función pública le indica al subsistema driver si se ha generado o no una interrupción hacia él; en el caso afirmativo, entrega a la aplicación del software operador el contenido de los registros de datos y de estados del PLD.

No recibe parámetros.

No tiene valor de retorno.

Su modificación con respecto al diseño fue el cambio en la condición de privada a pública.

Introducir_Dato()

Esta función pública recibe la información relacionada con la inserción de un dato en un canal determinado de cierto PCM, directamente de la aplicación del software operador para indicarle al subsistema driver que se va a lleva a cabo este procedimiento.

Recibe los parámetros PCM, Canal y Dato del tipo byte.

No tiene valor de retorno.

Aunque esta función no hace parte de los requerimientos del sistema TITD, se dedujo que era necesario implementarla para efectos de pruebas del mismo.

3.1.4 CLASE AGREGADA BD

3.1.4.1 PROPOSITO

El propósito de esta estructura-clase ha sido el de proporcionar a las clases Interprete_de Datos e Interfaz_de Operacion, y en particular al intérprete, una base de datos o tabla de referencia con las palabras claves o keywords, presentes en la aplicación del software operador como parámetros de las funciones Efectuar_Lectura() y Configurar_Stma, y sus respectivos valores equivalentes basados en la arquitectura y características de entramado PCM de la TITD. Esta clase es instanciada con entramado como objeto atributo de la clase Interprete_de Datos, que consiste en un arreglo de estructuras del tipo BD.

3.1.4.2 ATRIBUTOS

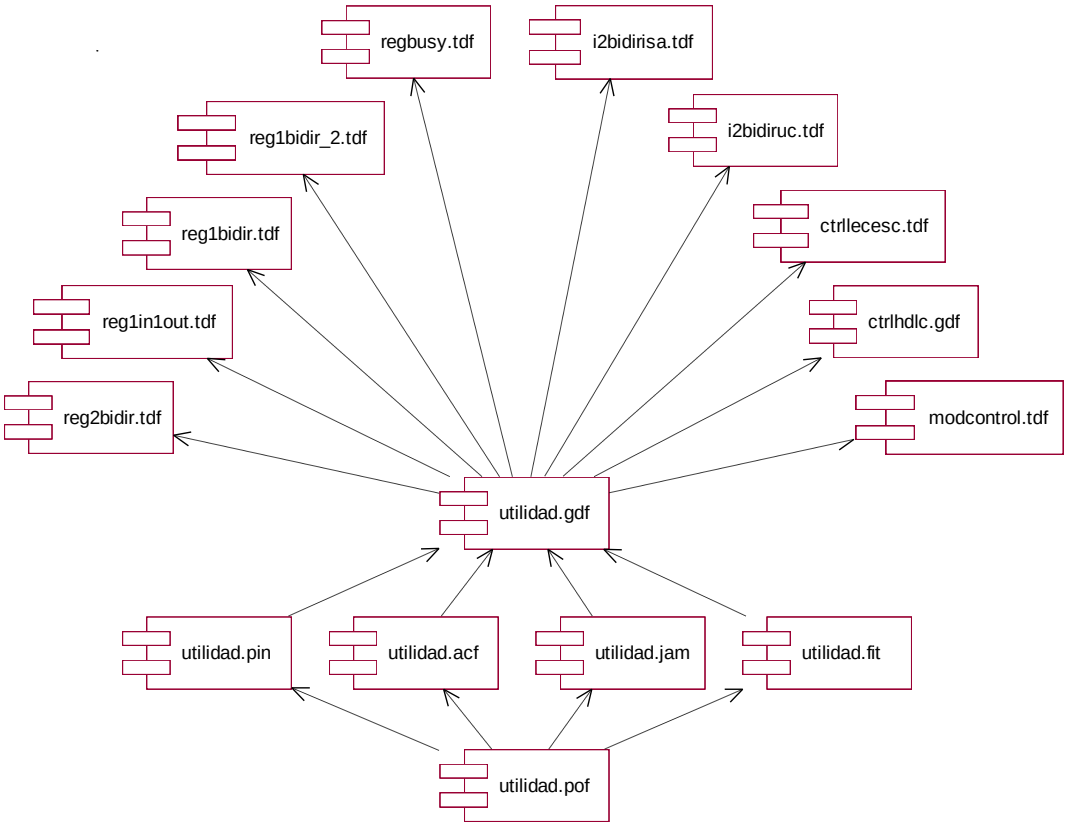
kword	Atributo del tipo arreglo de caracteres.
retorno	Atributo del tipo word.

El lenguaje utilizado para la implementación del subsistema TITD para el driver es VisualC++ (C++ para entorno Windows) del paquete Microsoft Developer Studio 97 (VC++ v5.0) con ayuda de la herramienta DriverX de Tetradyne con sus aplicaciones DriverX Device Configuration Utility v4.04 (Devcon.exe) para la configuración del dispositivo virtual; y Tetradyne Hardware Viewer v3.30 (Hardview.exe) para lectura/escritura de puertos y memorias, y enganches y conteos de IRQs.

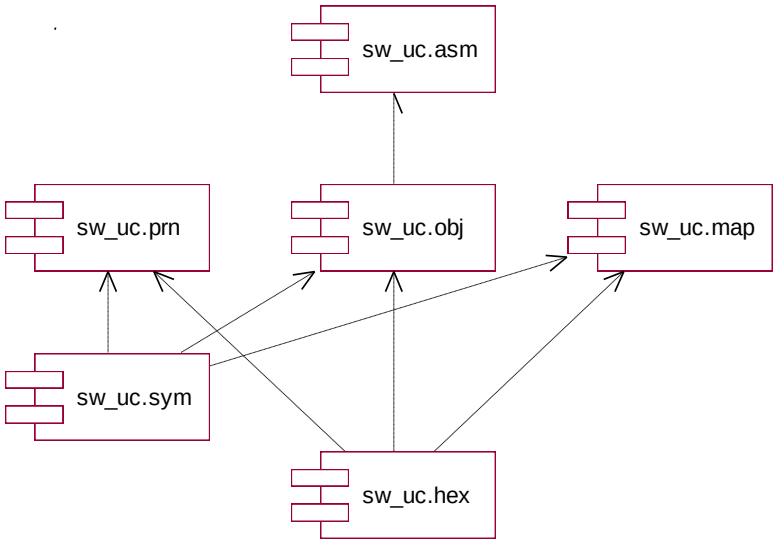
Para más detalles ver funciones específicas del driver en el anexo D (Manual de Usuario).

4 DIAGRAMAS DE COMPONENTES

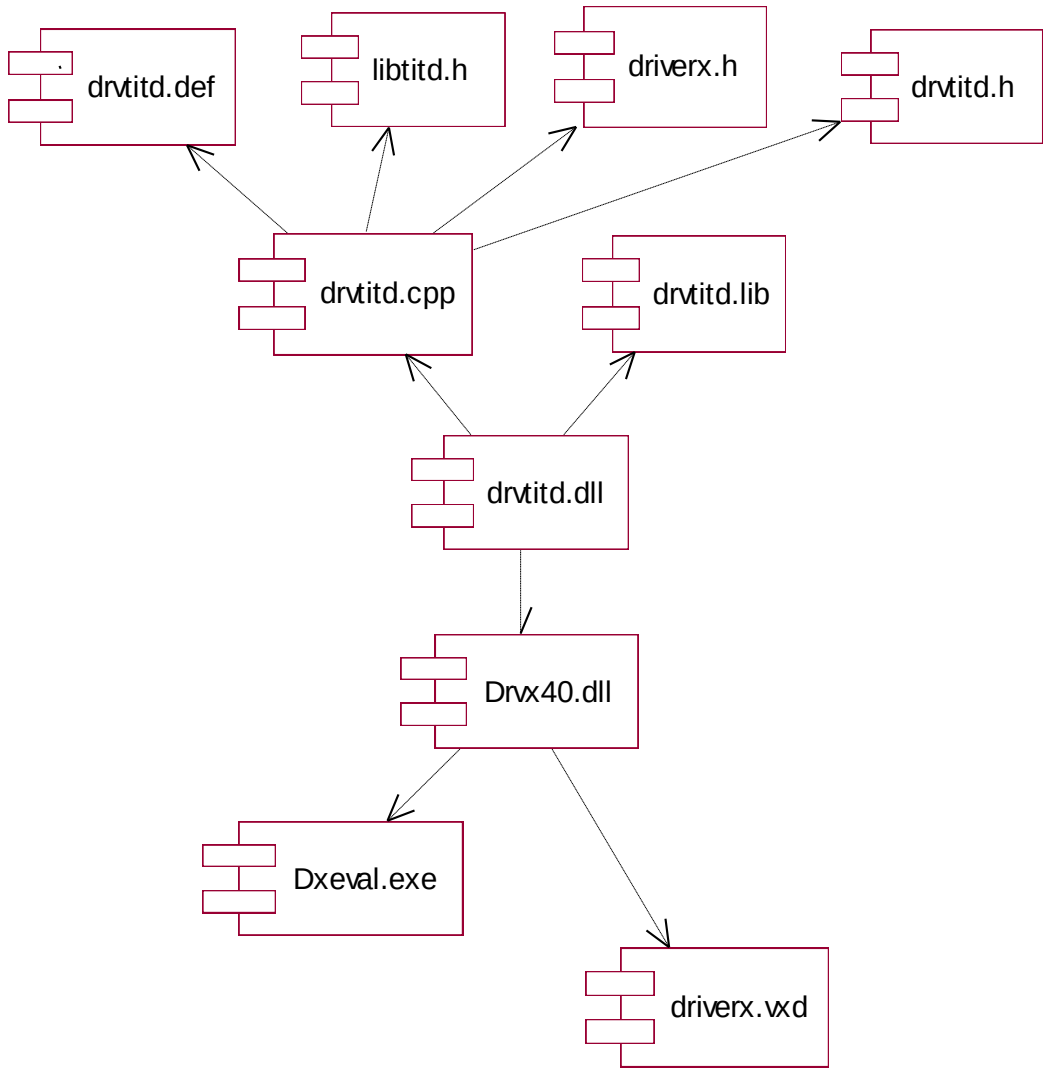
4.1 SUBSISTEMA PLD



4.2 SUBSISTEMA MICROCONTROLADOR



4.3 SUBSISTEMA DRIVER



5 ENTORNO DE PRODUCCION

A continuación se hará la descripción de las herramientas y procedimientos utilizados para generar los diferentes archivos del sistema y la puesta en funcionamiento del mismo.

5.1 SUBSISTEMA PLD

El archivo del sistema es **utilidad.pof** cuya extensión significa Programmer Object File.

Este archivo se genera con la aplicación Compilador de la herramienta Windows MAX PLUS II 9.23 Baseline de Altera.

El procedimiento de generación consiste en correr (start) el compilador quien secuencialmente compila, construye, sintetiza, ajusta y ensambla el código fuente.

La puesta en funcionamiento del subsistema se logra programando el integrado del PLD con el archivo del sistema utilizado el equipo programador para MAX5000 EPM5130JC PLD que consta de una tarjeta ISA LP6 de Altera conectada vía cable a una base PL MPU sobre la cual se ajusta una placa Adaptador MPU PLAD3-12. Este adaptador tiene los sockets para ajustar una segunda placa Adaptador PLEJ5130A en cuyo socket se inserta el PLD. Estas partes se controlan desde un PC con el software Programador de la herramienta Windows MAX PLUS II 3.10 de Altera. Una vez se corre el programador, la correcta programación del chip comienza con el botón de program y finaliza cuando la barra de progreso ha llegado al 100%.

5.2 SUBSISTEMA MICROCONTROLADOR

El archivo del sistema es **sw_uc.hex** cuya extensión alude al código de máquina en formato hexadecimal.

Este archivo se genera con las aplicaciones Avmac51 v2.04 (Avocet Macro Preprocessor) y Avlink v2.0 de la herramienta DOS Avocet (Avo51) de Avocet Systems.

El procedimiento de generación consiste en ejecutar Avmac51 que ensambla el .asm (código fuente en ensamblador) para crear el .obj (archivo objeto), el .map

(archivo macro preprocesador) y el .prn (archivo lista). Después se ejecuta Avlink que enlaza el los anteriores para crear el .sym (archivo de simulación en la aplicación Avsim51) y finalmente el .hex.

La puesta en funcionamiento del subsistema se logra programando el integrado del microprocesador con el archivo del sistema utilizado el equipo programador para microcontroladores que consta de una tarjeta ISA conectada vía cable a una base hardware programadora la cual tiene el socket en donde se inserta el chip. Estas partes se controlan desde un PC con el software Programador EXPRO para DOS. La óptima programación del chip depende de la adecuada selección software del dispositivo a programar (referencia y fabricante) y del estado del búfer en la memoria del PC asignado para la programación.

5.3 SUBSISTEMA DRIVER

Los archivos del sistema son: **drvtitd.dll**, **Drvx40.dll** (archivos o librerías de enlace dinámico), **Dxeval.exe** (aplicación ejecutable) y **driverx.vxd** (Virtual eXtended Driver o dispositivo virtual).

Estos archivos se generan gracias al Compilador (Builder) de VisualC++ 5.0 de la herramienta MS Developer Studio, con la ayuda de DriverX.

El procedimiento de generación consiste en ejecutar el comando Build del compilador quien compila y enlaza los códigos fuente, .h (archivos de cabecera), .def (archivo definiciones) .cpp (archivo en C++), junto con la librerías asociadas (.lib) y los archivos relacionados de DriverX.

La puesta en funcionamiento del subsistema se logra transportando los archivos del sistema al directorio System de Windows cuya ruta suele ser C:\WINDOWS\SYSTEM, en el cual se encuentran todas las DLLs y demás archivos destinados al manejo de dispositivos bajo el entorno Windows. Por otra parte, un

archivo .inf creado posteriormente, será el archivo que se encargue del transporte dichas DLLs y le proporcione al entorno Windows del PC, modificando el Registro de Windows, la información para la instalación de nuevo hardware. El .inf queda ubicado en el directorio Inf de Windows (C:\WINDOWS\INF\OTHER) con el nombre del fabricante o empresa y el dispositivo hardware (Universidad_del_CaucaTITDv1.inf en nuestro caso) una vez se haya instalado el dispositivo desde la aplicación Agregar Nuevo Hardware del Panel de Control.

De esta manera, se podrá observar y modificar la información relacionada con el controlador y los recursos (dirección E/S e Irq) del dispositivo (TITDV) desde el Administrador de Dispositivos, como de muestra en la figura 5, en las Propiedades del Sistema del Panel de Control.

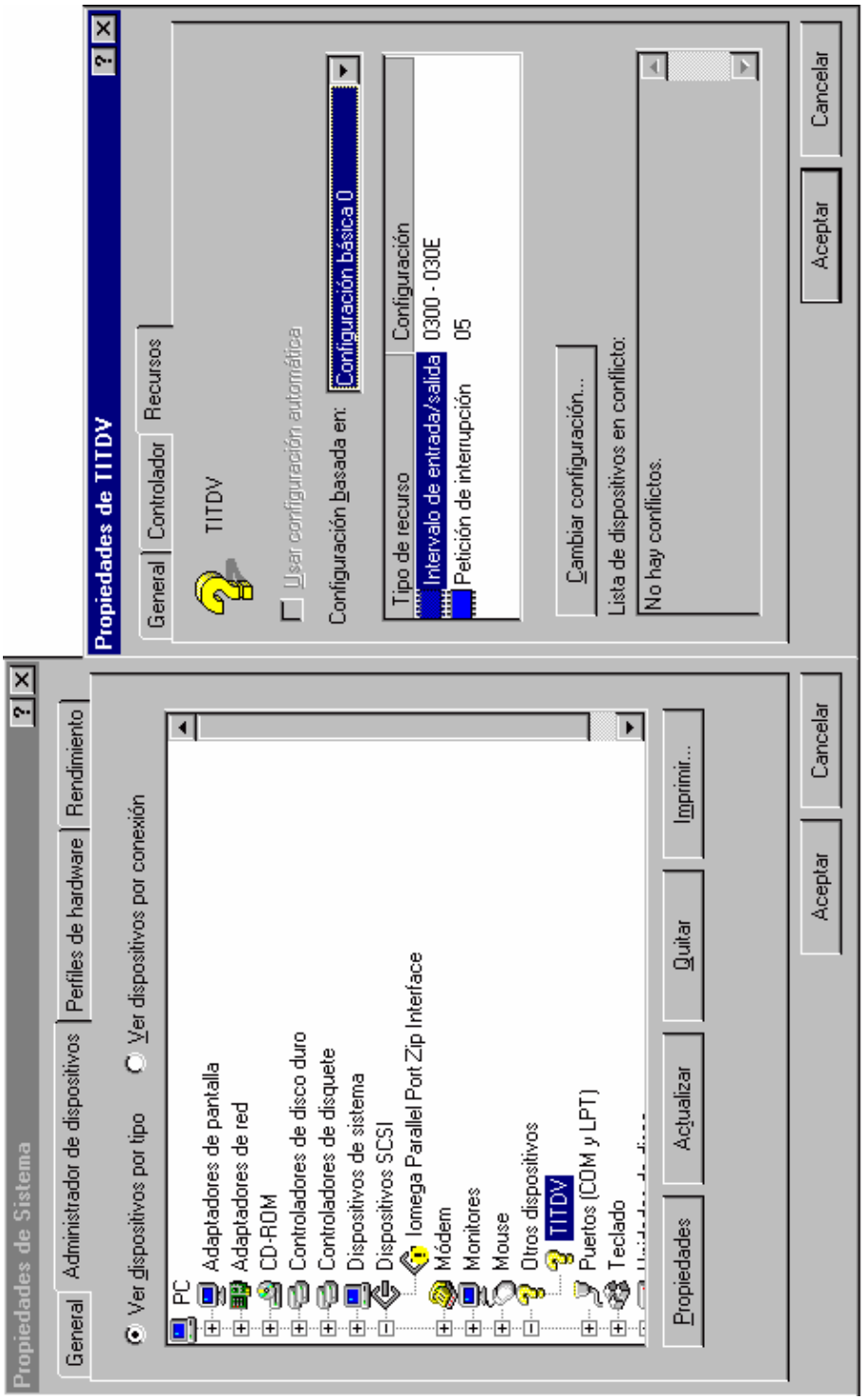


Figura 5. Información de la TITD en el Administrador de Dispositivos.

CAPITULO 4

CAPITULO 5
PRUEBAS DEL SISTEMA

Este capítulo contiene la información sobre las pruebas realizadas durante el desarrollo del sistema TITD que permitieron su puesta a punto.

Las pruebas son actividades que se han realizado en un entorno de ejecución definido y consisten en llevar el sistema implementado, ya sea en parte (componentes) o en su totalidad (prototipos operacionales), a dicho entorno, proceder con su respectivo montaje o instalación y ensayarlo. Por esto se tienen varios tipos de pruebas, clasificándose en pruebas de unidad, de integración, del sistema y de aceptación.

Es entonces, el objetivo del desarrollo de estas pruebas, establecer mecanismos para recolectar información referente a la detección de problemas que pueden ser errores de diseño o implementación, fallas de operación o defectos de los componentes, que impliquen un posible daño físico del dispositivo y que puedan incrementar los riesgos en la realización del proyecto; para hacer los respectivos correctivos, solucionar los inconvenientes y efectuar cambios o mejoras posiblemente no tenidas en cuenta.

De las pruebas se deduce la importancia de publicar un documento o manual con las instrucciones específicas de cómo instalar, probar, operar el producto final (la TITD) y qué medidas se deben tomar ante los reportes de posibles fallas o conflictos; garantizando así un soporte adecuado al cliente y la autonomía del producto con respecto al equipo desarrollador.

La información suministrada cubre todo el ciclo de vida de cada prueba: Su planeación y diseño (por qué y para qué se realizó, resultados esperados), la implementación y ejecución (cómo se llevó a cabo, entorno, descripción del hardware y software asociado, equipos utilizados), y su evaluación (problemas presentados y los resultados conseguidos).

1 PRUEBAS DE UNIDAD

Estas pruebas se realizan sobre un solo componente determinado, un dispositivo hardware o una aplicación software de interfaz o control separado e independiente.

1.1 PRUEBA DEL PUERTO PARALELO

Esta prueba surgió ante la necesidad de ensayar la implementación de un sistema de manejo sencillo y confiable de intercambio de datos entre el PC y el dispositivo Conmutador, para probar la funcionalidad del puerto como recurso interfaz bidireccional entre ellos. Se esperaba entonces, que la configuración ECP del puerto cumpliera satisfactoriamente con los requerimientos de obtener un flujo de datos bidireccional.

Para obtener información y detalles acerca del puerto y su configuración, consultar el Anexo A, numeral 5 documentos de soporte.

Para llevar a cabo esta prueba se implementó una aplicación tipo consola cuyo objetivo era, una vez ejecutada, chequear y desplegar la dirección E/S base del puerto, la configuración ECP, el modo de trabajo de tal configuración, la opción de modificar dicho modo, además de mostrar un menú para el envío y recepción de datos en las líneas de datos, estado y control del puerto.

El entorno de ejecución de esta prueba es el PC bajo DOS y se limitó al desarrollo de una aplicación software, así que el único montaje hardware realizado consistió en un puenteo, usando cables sencillos, de las distintas líneas del puerto.

El programa se elaboró con TurboC++ de DOS en lenguaje C que, una vez compilado, genera la aplicación de consola. Consta de varias funciones que realizan las siguientes tareas:

- Determinar la dirección E/S del puerto LPT1.
- Verificar la configuración ECP y el modo de trabajo de ésta.
- Desplegar el menú de llamado a las distintas funciones que escriben o leen del puerto.
- Configurar las líneas de datos como entrada o salida.
- Enviar datos ya sea a las líneas de datos o control del puerto.
- Recibir datos ya sea por las líneas de datos, control o estado del puerto.

Para obtener información y detalles acerca del programa, consultar el Anexo B, numeral 1 código fuente.

La evaluación de esta prueba, bajo el previo estudio del puerto, logró resultados satisfactorios ya que se corroboró la funcionalidad del mismo como interface bidireccional.

1.2 PRUEBA DE SIMULACION PARA EL MICROCONTROLADOR

Esta prueba surgió ante la necesidad de ensayar el funcionamiento del subsistema microcontrolador, para probar su comportamiento operacional. Se esperaba entonces, que la aplicación desarrollada operara del modo adecuado según los requerimientos del subsistema modelado.

Para llevar a cabo esta prueba se utilizó la aplicación Avsim51 de la herramienta Avocet en la cual se realizó la simulación del programa ensamblador. Avsim51 despliega en pantalla la memoria de programa, la zona de memoria, los registros especiales, banderas, puertos, entradas y salidas del microcontrolador permitiendo visualizar la ejecución del código de máquina y su comportamiento hacia el interior del integrado.

El entorno de ejecución de esta prueba es el PC bajo DOS y se limitó a la simulación de una aplicación software, así que no hubo necesidad de realizar ningún montaje hardware.

Para obtener información y detalles acerca de la aplicación software para este subsistema, consultar el Anexo B, numeral 4 código fuente.

La evaluación de esta prueba, bajo el previo estudio del microcontrolador, logró resultados satisfactorios ya que se comprobó que el programa operaba de forma adecuada dentro del microcontrolador. La simulación demostró en un principio algunas fallas de código y detalles no tenidos en cuenta, así que Avsim51 también se utilizó como herramienta de depuración.

1.3 PRUEBA DE SIMULACION PARA LOS PLDS EPM5130JC Y EPM7128SLC84

Esta prueba surgió ante la necesidad de ensayar el funcionamiento del subsistema PLD, para probar su comportamiento operacional. Se esperaba entonces, que la aplicación desarrollada operara del modo adecuado según los requerimientos del subsistema modelado.

Para llevar a cabo esta prueba se utilizó la aplicación del Editor de Forma de Onda (Waveform editor) de la herramienta MAX PLUS II de Altera en la cual se realizó la simulación del programa AHDL. El editor de forma de onda despliega en pantalla el estado lógico de las señales o conjuntos de señales digitales de entrada y salida externas e internas de un componente, módulo o aplicación en el transcurso del tiempo asignado. De esta manera permite visualizar el comportamiento del código descriptivo del hardware hacia el interior del integrado.

Para obtener información y detalles acerca de la simulación para este subsistema, consultar el Anexo C, numeral 3 diagramas de simulación del PLD.

El entorno de ejecución de esta prueba es el PC bajo Windows y se limitó a la simulación de una aplicación software descriptiva, así que no hubo necesidad de realizar ningún montaje hardware.

Para obtener información y detalles acerca del software para este subsistema, consultar el Anexo B, numeral 3 código fuente.

La evaluación de esta prueba, bajo el previo estudio de los PLDs y FPGAs, logró resultados satisfactorios ya que se comprobó que el programa operaba de forma adecuada dentro del PLD. La simulación demostró en un principio algunas fallas de código y detalles no tenidos en cuenta, así que el editor de forma de onda también sirvió como herramienta de depuración.

2 PRUEBAS DE INTEGRACION

Estas pruebas se realizan sobre un grupo de componentes, dispositivos hardware o aplicaciones software de interfaz o control separados e independientes.

2.1 PRUEBA DE LOS DISPOSITIVOS ESPECIALIZADOS EN TELECOMUNICACIONES 1

Esta prueba surgió ante la necesidad de ensayar la configuración, programación, comportamiento y modo de operación de los dispositivos digitales Conmutador y PLL, para probar la integridad y confiabilidad tanto de los encapsulados de estos dos integrados (en su aspecto físico), como de los datos introducidos hacia y recibidos del conmutador. Se esperaba entonces que los dos integrados quedaran configurados y se conectaran entre sí de forma apropiada, de tal manera que su funcionamiento en conjunto fuera lo más óptimo posible para garantizar que la lectura de datos sobre ellos fuera correcta.

Para llevar a cabo esta prueba se utilizó el sistema de intercambio de datos entre el PC y el conmutador implementado a través del puerto paralelo ECP LPT1 y un

puerto extra LPT2 proveniente de una tarjeta ISA Multi_I/O. También se implementó una aplicación tipo consola cuyo objetivo era, una vez ejecutada, desplegar la dirección E/S base de los puertos LPT1 y LPT2, configurar y desplegar la configuración ECP del LPT1 bajo el modo de trabajo 1 (líneas de datos tipo byte bidireccionales), además de mostrar los menús para la programación del conmutador bajo el modo de mensaje o conmutación y lectura de sus memorias.

Para obtener información y detalles acerca del conmutador y el PLL, consultar el Anexo A, numerales 2 y 3.

El entorno de ejecución de esta prueba es el PC bajo DOS para la parte software y un montaje sobre una QT o Protoboard conectada a los LPT1 y LPT2 del PC con cables DB25 para la parte hardware.

Para obtener información y detalles acerca del montaje, consultar el Anexo C, numeral 1 esquemático (pinout o diagrama de pines) 1.

Los equipos utilizados en esta prueba fueron: Una Fuente Switching de PC, un Multímetro digital, un Osciloscopio para visualización de señales análogas bipolares y digitales, y un Analizador Lógico o de Estados Hewlett Packard 1651A para captura detallada de las señales de reloj y las tramas PCM32 digitales.

El programa se elaboró con TurboC++ de DOS en lenguaje C que, una vez compilado, genera la aplicación de consola. Consta de las funciones para la prueba de unidad del puerto paralelo y las funciones para el intercambio de datos con el conmutador descritas en la clase Intercambio_de_Datos_Cx (a excepción de Reservar() y Liberar()) y la funcionalidad del módulo para el handshake del PLD (ver capítulo 4).

La evaluación de esta prueba, bajo el previo estudio de los integrados conmutador y PLL, logró resultados satisfactorios ya que se comprobó la óptima calidad de los

datos obtenidos en lectura corroborando la adecuada programación de los integrados.

Se tuvo problemas con la interpretación del protocolo de intercambio de datos con el conmutador o handshake y por eso se recurrió a un estudio más profundo y detallado del mismo.

2.2 PRUEBA DE LOS DISPOSITIVOS ESPECIALIZADOS EN TELECOMUNICACIONES 2

Esta prueba surgió ante la necesidad de ensayar la configuración, comportamiento y modo de operación del dispositivo digital Interfaz E1, para probar la integridad y confiabilidad tanto de los encapsulados del integrado (en su aspecto físico), como de los datos introducidos hacia y recibidos de la interfaz E1, con el Conmutador como intermediario. Se esperaba entonces que el integrado quedara configurado y se conectara con el PLL y el conmutador de forma apropiada, de tal manera que su funcionamiento en conjunto fuera lo más óptimo posible para garantizar que la lectura de datos sobre ellos fuera correcta.

Para llevar a cabo esta prueba se utilizó el mismo sistema de la prueba anterior y la misma aplicación de consola.

Para obtener información y detalles acerca de la Interfaz E1, consultar el Anexo A, numeral 1.

El entorno de ejecución de esta prueba es el mismo de la prueba anterior pero con un montaje adicional.

Para obtener información y detalles acerca del montaje, consultar el Anexo C, numeral 1 esquemático (pinout o diagrama de pines) 2.

Los equipos utilizados en esta prueba fueron los mismos para la prueba anterior.

La evaluación de esta prueba, bajo el previo estudio del integrado Interfaz E1, logró resultados satisfactorios ya que se comprobó la óptima calidad de los datos obtenidos en lectura, a través del conmutador, corroborando la adecuada configuración del dispositivo.

Se tuvo problemas con el montaje correcto adicional para el híbrido Interfaz E1 y por eso se recurrió a un estudio más profundo y detallado del mismo.

2.3 PRUEBA DE LOS DISPOSITIVOS ESPECIALIZADOS EN TELECOMUNICACIONES, PLD Y MICROCONTROLADOR

Esta prueba surgió ante la necesidad de ensayar el comportamiento de los dispositivos de telecomunicaciones, PLD y microcontrolador en conjunto, para probar el prototipo operacional del sistema TITD fuera de la tarjeta sin la intervención del PC por medio del bus ISA. Se esperaba entonces que la comunicación entre todos los dispositivos fuera adecuada, de tal manera que su funcionamiento en conjunto fuera lo más óptimo posible para garantizar que la operación de esta parte del sistema fuera correcta.

Para llevar a cabo esta prueba se utilizaron los integrados microcontrolador de Atmel 89C52, tras previa programación, y el PLD EPM7128SLC84 de Altera, tras previa programación mediante la tarjeta UPx de Intectra. El PLD reside en la tarjeta y es ésta la que recibe los datos de programación vía puerto paralelo del PC con el software programador de MAX Plus II de Altera. La tarjeta se comunica con el microcontrolador y con el conmutador a través de una Interface de 2 conectores tipo IDE.

El entorno de ejecución de esta prueba es un montaje sobre una QT o Protoboard conectada a la tarjeta UPx.

Para obtener información y detalles acerca del montaje, consultar el Anexo C, numeral 2 esquemático (pinout o diagrama de pines).

Los equipos utilizados en esta prueba fueron los mismos de las 2 pruebas anteriores.

La evaluación de esta prueba, bajo el previo estudio de la tarjeta UPx, logró resultados satisfactorios ya que se comprobó la óptima calidad de los datos obtenidos en lectura corroborando el correcto funcionamiento grupal de los dispositivos.

Se tuvo problemas con la programación del microcontrolador ya que el equipo programador se encontraba distante de la locación del proyecto. Otros inconvenientes surgieron, en cuanto a fallos en la prueba, debido a fallas de codificación relacionadas con la velocidad de procesamiento del microcontrolador y con pérdida de datos por reasignación de variables. Por esto se trabajó con cristales de menor frecuencia y se depuró el código ensamblador del microcontrolador.

3 PRUEBAS DEL SISTEMA

Estas pruebas se realizan directamente sobre el dispositivo hardware y la aplicación software que lo controla, en este caso la tarjeta TITD y los subsistemas que lo conforman.

3.1 PRUEBAS DE CONSOLA 1 Y 2

Estas pruebas surgieron ante la necesidad de ensayar el comportamiento del sistema completo, para probar el prototipo operacional del sistema TITD dentro de la tarjeta con la intervención del Driver-PC por medio del bus ISA. Se esperaba entonces que la comunicación entre todos los dispositivos incluyendo el driver fuera adecuada, de tal manera que su funcionamiento en conjunto fuera lo más óptimo posible para garantizar que la operación de todo el sistema fuera correcta.

Para llevar a cabo estas pruebas se insertó la tarjeta en un slot ISA del PC y se implementaron dos aplicaciones de consola con VisualC++ que utilizan el driver y sus respectivas funciones. La primera efectúa, en el programa principal, la última prueba del diagnóstico fuera de funcionamiento (implementado para el microcontrolador) sin tener en cuenta la IRQ hacia el ISA. La segunda presenta un menú que permite utilizar todas las funciones básicas de la TITD, en donde para cada una, se introducen por teclado y se validan los parámetros que dichas funciones reciben.

Para obtener información y detalles acerca de la aplicación software de las consolas, consultar el Anexo B, numeral 6 código fuente.

El entorno de ejecución de esta prueba es el PC bajo DOS.

La herramienta utilizada en esta prueba fue la aplicación de DriverX Hardware View descrita en el capítulo 4.

La evaluación de esta prueba, bajo el previo estudio de los Drivers y la herramienta DriverX, logró resultados satisfactorios ya que se comprobó la óptima calidad de los datos obtenidos en lectura corroborando el correcto funcionamiento del sistema TITD en la tarjeta.

Se tuvo problemas con defectos de fabricación en la placa impresa de la tarjeta pero se solucionó con la ayuda de personal especializado en el tema.
Otro inconveniente surgió, en cuanto a fallos en las pruebas, debido a fallas de codificación relacionadas con la velocidad de procesamiento del microcontrolador, ya que los relés exigían tiempos mayores de actividad. Estas fallas se reflejaban en pérdidas de datos por falta de sincronismo. Se llegó a una solución implementado retardos mayores en tiempo de ejecución para dar tiempo suficiente a las actividades de los relés.

También se presentaron problemas con la interrupción hacia el bus ISA, y por ello se optó por profundizar en el tema.

3.2 PRUEBA DE APLICACION WINDOWS

Esta prueba se realizó y evaluó bajo los mismos criterios de la prueba anterior, pero ya no desde una consola DOS sino en el entorno Windows, con el fin de permitir manipular, de una manera amigable, todas las funciones básicas que ofrece la TITD.

La aplicación despliega en pantalla una interfaz de usuario Windows que permite utilizar todas las funciones básicas de la TITD, en donde para cada una, se introducen por teclado los parámetros que dichas funciones reciben y se despliegan en pantalla los resultados correspondientes a las funciones que tienen valores de retorno.

Para obtener información y detalles acerca de la aplicación Windows, consultar el Anexo B, numeral 7 código fuente.

El entorno de ejecución de esta prueba es el PC bajo Windows.

4 PRUEBAS DE ACEPTACION

Estas pruebas se realizan en presencia del cliente ensayando el sistema desde un punto de vista que el cliente pueda entender, mostrando la operabilidad completa del sistema e indicando las instrucciones que se tratarían a nivel de un manual de usuario.

En este caso corresponde a la sustentación del proyecto frente a los directores, asesores, jurados y demás personal relacionado con el mismo.
Para obtener información y detalles acerca del manual de usuario, consultar el Anexo D.

5 ESTIMACION DEL PROYECTO

En este numeral se presenta una actualización del Plan del sistema (numeral 6), Estimación de la Métrica (numeral 6.1) mostrado en el capítulo 1.

DOCUMENTACION 1

- **Anteproyecto**, elaboración y aceptación.
- **Estudio** y repaso de los fundamentos **teóricos** relativos al tema central que cobija la parte de prueba de los circuitos integrados especializados, las técnicas de modulación por codificación de pulsos (PCM), velocidades y estructura de entramado, conmutación digital TST, codificación de línea, chequeo de redundancia cíclica (CRC), señalización por canal, interfaces con el puerto paralelo de capacidades extendidas (ECP). Referente a la etapa de **estudio** de los **circuitos integrados** especializados, su arquitectura, configuración y funcionamiento (transcripción).
- **Estudio** para manipulación del equipo **analizador lógico HP**.

IMPLEMENTACION 1

- **Software** de **prueba y configuración** del **puerto paralelo** Modo ECP (OLPT2.C).
- **Software** de **configuración y prueba** del **conmutador** a través del puerto paralelo y para posterior prueba de la interfaz.
- **Interfaz** física entre **LPTs** y **Circuitos integrados**.
- **Montaje** de circuitos de **prueba** de los integrados especializados, **cableado** PLL, Conmutador e interfaz E1 y pruebas.

MODELAMIENTO

- **Estudio** del Lenguaje Unificado de Modelado (**UML**), el Proceso Unificado (**RUP**) y la herramienta **Rational Rose**.
- **Análisis** de **Requerimientos**.

- **Análisis** del **Sistema**.
- **Diseño** del **Sistema**.

IMPLEMENTACION 2

- **Implementación** y **Simulación** del Sistema TITD para el **PLD** (utilidad.gdf) con la herramienta Max Plus II de Altera.

IMPLEMENTACION 3

- **Implementación** y **Simulación** del Sistema TITD para el **Microcontrolador** (sw_uc.asm) en Assembler con ayuda de la herramienta Av51 de Avocet Systems.

DOCUMENTACION 2

- **Estudio** de los fundamentos **teóricos** relativos a la elaboración de un **Driver** en Windows. **Estudio** para manipulación de la herramienta DriverX.

IMPLEMENTACION 4

- **Implementación** y **Simulación** del Sistema TITD para el **Driver** (drvtitd) en VisualC++ con la herramienta Developer Studio de Microsoft y DriverX de Tetradyne.

IMPLEMENTACION 5

- **Software** de **prueba** Sistema **TITD** en VisualC++ aplicación de **consola** (testc2dll).
- **Software** de **prueba** Sistema **TITD** en VisualC++ aplicación **Windows** (testdll).

DOCUMENTACION 3

- **Monografía**, elaboración de capítulos.
- **Anexos**, elaboración.

METRICA

ETAPA	PUNTO	TIEMPO	# LINEAS DE CODIGO
Documentación 1	Anteproyecto	4 meses Oct-Nov-Feb-Marz	
	Estudio teoría	2 meses Feb-Marz	
	Estudio CIs y Analizador lógico	3 meses Marz-Abr-May	
Implementación 1	Software prueba y Configuración LPT (ECP)	2 semanas Jun	260
	Software prueba y Configuración Conmutador	1 mes Jun-Jul	510
	Interfaz LPT $\Rightarrow$ Cis	1 semana Jul	
	Montaje, prueba y Cableado	2 semanas Jul-Ago	
Modelamiento	Estudio UML, RUP y Rational Rose.	2 semanas Ago	
	Análisis Requerimientos	2 semanas Ago-Sep	
	Análisis Sistema	3 semanas Sep	
	Diseño Sistema	5 semanas Oct-Nov	
Implementación 2	Implementación y Simulación PLD	3 semanas Nov	400
Implementación 3	Implementación y Simulación Microcontrolador	3 semanas Dic	735

ETAPA	PUNTO	TIEMPO	# LINEAS DE CODIGO
Documentación 2	Estudio teoría Driver	1 semana Ene	
Implementación 4	Implementación y Simulación Driver	3 semanas Ene	760
Implementación 5	Software prueba TITD consola	1 semana Feb	300
	Software prueba TITD Windows	2 semanas Feb	530
Documentación 3	Monografía	6 semanas Ene-Feb	
	Anexos	4 semanas Feb	

CAPITULO 5