PROTOCOLO AS-INTERFACE: IMPLEMENTACIÓN DE NODO ESCLAVO EN UN MICROCONTROLADOR.

ARNOL RAFAEL MARRIAGA CABRALES

OSCAR AUGUSTO ABREU SILVA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE FISICOMECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

Bucaramanga – 2006

UNIVERSIDAD INDUSTRIAL DE SANTANDER
Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Perfecta combinación entre Energía e Intelecto

PROTOCOLO AS-INTERFACE: IMPLEMENTACIÓN DE NODO ESCLAVO EN UN MICROCONTROLADOR.

ARNOL RAFAEL MARRIAGA CABRALES

OSCAR AUGUSTO ABREU SILVA

Trabajo de grado para optar por el título de Ingeniero Electrónico

Director

MPe. JAIME GUILLERMO BARRERO PEREZ

Co-Director

Ing. OMAR LEONARDO NÚÑEZ GUALDRÓN

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE FISICOMECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

Bucaramanga–Mayo de 2006

Este trabajo está dedicado a nuestros padres los cuales han consagrado toda su vida para garantizar el bienestar y superación de nuestras familias, a nuestros hermanos quienes siempre han sido una gran fuente de motivación para seguir adelante a pesar de las dificultades.

AGRADECIMENTOS

Agradezco a Dios por todo lo que me ha permitido vivir hasta el momento, a mi familia por ser pilar fundamental en la consecución de cada uno de mis logros y por enseñarme a labrar los caminos por los cuales he transitado para alcanzar esta meta.

A mis amigos, por esa voz de apoyo en aquellos tiempos difíciles, por estar ahí en el momento indicado dándome una voz de aliento, en especial a quienes siento como mis hermanos, Arnol R. Marriaga Cabrales, Angélica Duarte Moya, Armando Ayala Pabón y Jerson D. Lindao Argüello; consideren este también como su logro, porque cada uno de ustedes tiene un gran aporte en él.

Gracias a mis maestros y profesores de la Universidad Industrial de Santander, por compartir cada una de sus enseñanzas conmigo, en especial a nuestro director Jaime G. Barrero P. y nuestro co-director Omar L. Nuñez G. por el respaldo y la confianza brindada en el desarrollo de este proyecto. Al profesor José A. Amaya Palacios, por el incondicional apoyo desde el primer día, por creer en nosotros y por todo ese conocimiento que compartió de manera noble y abnegada, entregando siempre lo mejor.

Finalmente a todas aquellas personas que de una u otra forma ayudaron a la conquista de esta meta, compañeros y amigos, que en el anonimato dieron en algún momento lo mejor de sí a esta causa, muchas gracias, de seguro la vida les concederá siempre lo mejor.

Oscar Augusto Abreu Silva

AGRADECIMENTOS

Inicialmente agradezco a mi familia por haber puesto en mi camino todas las herramientas necesarias para alcanzar esta meta. A mi hermano Nilson quien me apoyo y orientó durante mi carrera y que se ha convertido en un modelo a seguir, a mi hermano Tarquino quien me animó y enseñó lo bonito de la vida, y a mi padre quien es mi fuente de motivación.

Agradezco a mis amigos por su apoyo incondicional; en especial a Angélica Duarte Moya, Armando Ayala Pabón, Oscar A. Abreu Silva, Jerson D. Lindao Argüello y sus familias, ya que sin su ayuda este logro no habría sido posible.

A la Universidad Industrial de Santander y a todos los docentes que han contribuido en mi formación profesional y personal. A mi director Jaime Guillermo Barrero Pérez y mi co-director Omar Leonardo Núñez Gualdrón por haber dirigido mi trabajo de pregrado, por su incondicional motivación y respaldo. Al profesor Jose A. Amaya Palacios por su incondicional apoyo, y por todas sus enseñanzas.

Finalmente a todas aquellas personas que de una u otra forma ayudaron a la conquista de esta meta, compañeros y amigos, que en algún momento dieron lo mejor de sí a esta causa, muchas gracias.

Arnol Rafael Marriaga Cabrales

RESUMEN

TÍTULO: PROTOCOLO AS-INTERFACE: IMPLEMENTACIÓN DE NODO ESCLAVO EN UN MICROCONTROLADOR*

AUTORES: Oscar Augusto Abreu Silva, Arnol Rafael Marriaga Cabrales**

PALABRAS CLAVES: Esclavo, *AS-interface*, protocolo, hardware, software.

DESCRIPCIÓN

Este proyecto plantea la implementación de un nodo esclavo AS-interface utilizando un microcontrolador, que cumpla con los requerimientos mínimos del protocolo expuestos en la norma EN50295, con la capacidad de ser integrado en una red comercial, a través de los acoples que permitan las funciones de transmisión y recepción de datos, así como la extracción de la energía necesaria para su funcionamiento.

En este trabajo la implementación general del esclavo es abordada mediante el desarrollo de dos áreas. La primera de ellas es la implementación de la interfaz física (Hardware), en la cual se incluye la selección del microcontrolador y el diseño de las etapas de recepción, transmisión, conmutación y fuente de alimentación. La segunda, se encarga de la parte lógica del esclavo (software), donde se realizan los diagramas de flujo y la programación del microcontrolador, necesarios para el cumplimiento de la maquina de estados del esclavo, y el procesamiento la información durante el proceso de comunicación con el maestro.

El esclavo implementado presenta el perfil S-3.0, que establece su configuración como dos puertos de entrada y dos puertos salida, su consumo de corriente es de 100mA y a nivel físico se presenta como interfaz genérica que soporta cualquier tipo de perfil con solo variar el software. Su diseño de hardware y software es completamente modular, permitiendo su utilización en otro tipo de aplicaciones dentro de la red. El dispositivo fue validado en una red comercial, realizando un proceso de comunicación exitoso y cumpliendo con los rangos establecidos en la norma EN50295.

---

*Modalidad: Trabajo de grado.

**Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: MPe. Jaime Guillermo Barrero Pérez. Co-Director: Ing. Omar Leonardo Núñez Gualdrón.

SUMMARIZE

TITLE: AS-INTERFACE PROTOCOL: SLAVE NODE IMPLEMENTATION ON A MICROCONTROLLER*

AUTORS: Oscar Augusto Abreu Silva, Arnol Rafael Marriaga Cabrales**

KEY WORDS: Slave, As-Interface, Protocol, hardware, software.

DESCRIPTION

This project implements an As-Interface slave node using a microcontroller. The main goal is to accomplish the specifications of the EN50295's norm. The final outcome can be integrated into a commercial network, and, with the right connectors, start the data transmition-reception process.

In this work, the slave implementation was made using two layers. The first one is the hardware implementation. At this level, the microcontroller's selection, the transmitter, the receiver, the power source and the commutation stage were designed. The second one deals with the software layer. At this level, the flow diagram and the microcontroller's program were developed. All the achievements on this level accomplish with the slave's state machine and the information processing during the master-slave communication process.

The implemented slave had an S-03 profile, this meant that it had two input and two output ports. Its current consumption is 100 mA and its generic physical structure interface supports all kinds of profiles. Its hardware and software layers are completely modular, allowing the device to be used on many other As-Interface applications. The final system was tested on a commercial network where it showed to fulfill the response time of the EN50295's norm.

*Degree Project

**Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: MPe. Jaime Guillermo Barrero Pérez. Co-Director: Ing. Omar Leonardo Núñez Gualdrón.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

El desarrollo del control distribuido en la industria, va ligado a la generación de nuevas técnicas que permitan el control y la supervisión de procesos de forma remota. Cada vez más, es necesario disponer de dispositivos inteligentes capaces de comunicarse y a su vez de generar funciones de autodiagnóstico dentro de una red. En este sentido, se hace indispensable la apropiación de las tecnologías comerciales existentes y el estudio de protocolos de comunicación industrial.

Es así como protocolos orientados a buses de campo[1] se muestran como los de mayor proyección en el nivel sensor/actuador. La transmisión serial de datos y el proceso de autodiagnóstico, los presenta como la solución más eficiente y flexible en la implementación del control distribuido. En consecuencia, la investigación de protocolos estándar y la implementación de los diferentes dispositivos asociados a una red comercial, se presenta como un tema de actualidad mundial, orientado a la generación de soluciones menos costosas y de mayor rendimiento para uso de la industria.

En este trabajo se presenta la implementación de un nodo esclavo en un dispositivo programable, basados en el protocolo de buses de campo *AS-interface*[2], se describe la implementación de la interfaz física y la interfaz lógica necesaria para su funcionemiento dentro de una red *AS-i* comercial, especificados en la norma EN50295 [11], en la cual se establece cada uno de los parámetros asociados con el protocolo.

La elección del protocolo *AS-i*[3] se justifica por su gran versatilidad, bajos costos de man-

[1]Un bus de campo es, de manera general, un bus digital serie, bidireccional que transmite información entre los sensores/actuadores y los dispositivos de control de una red industrial.

[2]Los términos en inglés usados se presentarán en letra cursiva.

[3]En adelante se utilizará el término *AS-i* para referirse al protocolo *AS-interface*

tenimiento, alta fiabilidad, fácil integración de nuevos miembros a la red y la transmisión de datos y alimentación a través de una misma línea de dos hilos. Asi mismo, a nivel internacional cuenta con el respaldo de la AS-INTERNATIONAL ASSOCIATION, y de un número considerable de fabricantes (Siemens, Schneider, Allan Bradlley, etc) que garantizan su permanencia en el mercado, colocándola en un lugar privilegiado en el nivel sensor/actuador. Sin embargo, *AS-i* es un concepto independiente del fabricante, y define su funcionamiento a través de la norma EN50295 [11].

El módulo fue implementado con el perfil S-3.0 [11],que establece la configuración del dispositivo esclavo para el intercambio de señales digitales con el maestro. Su correcta operación fue evaluada con la puesta en funcionamiento dentro de una red comercial y la verificación del éxito de la comunicación.

El dispositivo se destaca por ser una interfaz física genérica, que permite la implementación de cualquier tipo de perfil de esclavo. Además, hece posible la modificación a través de Software de parámetros, que para esclavos convencionales son de carácter estático. Tiene la capacidad de extraer la potencia para su funcionamiento a través de la línea, con requerimientos de consumo dentro de los rangos establecidos por la norma [11].

Pirámide de automatización

La automatización de procesos en la industria moderna, ha traído consigo el manejo de una gran cantidad de información, que obliga a la división de tareas entre grupos de procesadores organizados de manera jerárquica; formando lo que se conoce en el entorno industrial como pirámide de automatización. Así, dependiendo de la función y el tipo de conexiones, se suelen distinguir cuatro[4] niveles en un proceso industrial, la figura 1.1 ilustra cada uno de estos niveles y a continuación se presnta una breve descripción de los mismos.

- **Nivel de sensor/actuador:** Este es el nivel más próximo al proceso, en donde se logra la comunicación entre sensores/actuadores y dispositivos de control.

- **Nivel de campo:** Integra pequeños automatismos(PLCs compactos, PIDs, multiplexores de e/s, etc) en subredes o islas. En este nivel se puede encontrar uno o varios autómatas modulares actuando como maestros de la red.

[4]Cabe aclarar que esta estructura no es universal, varía con el tamaño del proceso y sus características particulares.

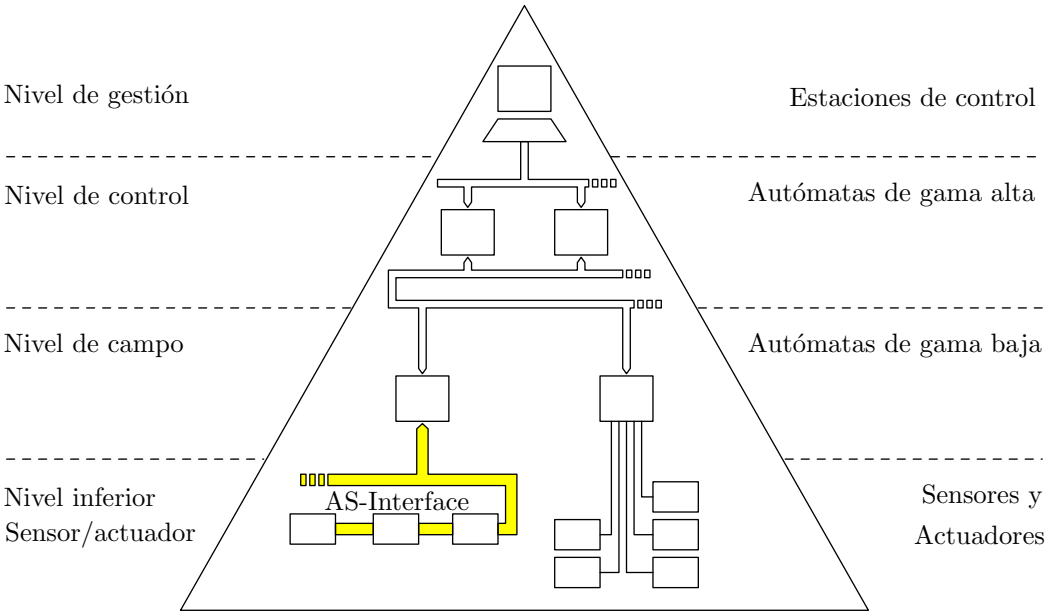


Figura 1.1: Pirámide de automatización

- **Nivel de control:** Enlaza las células de fabricación o zonas de trabajo. En este nivel se sitúan los autómatas de gama alta y los ordenadores dedicados al diseño, control de calidad, programación, etc.

- **Nivel de gestión:** Se utilizan para redes de oficinas, en las cuales se intercambia un alto volumen de información y los tiempos de respuesta no son críticos.

Asimismo, El manejo de grandes volumenes de información ha permitido un importante desarrollo en las comunicaciones industriales. La necesidad de comunicaciones fiables y en tiempo real en el nivel de sensor/actuador, ha impulsado el desarrollo de diferentes técnicas que permitan un control descentralizado y que faciliten la integración sencilla de nuevos elementos a una red industrial.

Antecedentes

A nivel local, y específicamente en la Universidad Industrial de Santander, se ha iniciado el desarrollo de trabajos orientados a fortalecer el área de comunicaciones industriales En este sentido ya se cuenta con trabajos tales como:

Nino Varon, Becerra Alvaro. Redes industriales. MODBUS y ETHERNET implementados en automatas programables TRILOGI, KOYO y MODICON-TELEMECANIQUE. Tesis de

pregrado, 2002.

Gelvez F. Julio. Redes de Comunicaciones Industriales. Monografía Especialización en Telecomunicaciones,2002.

López C. Duván A. Redes de Comunicación AS-i. Funcionamiento, Ventajas y Estado del Arte. Monografía Especialización en Telecomunicaciones, 2004.

Carreño Sinle, Ardila Pedro. MODBUS, Monitoreo de la red empleando LABVIEW. Tesis de pregrado, 2005.

Duqué P. Jorge E. Procesador de comunicación MODBUS. Implementación con microcontrolador. Tesis de Maestría, 2005.

Los resultados de este proyecto, sirven como soporte a futuros trabajos enfocados hacia las comunicaciones industriales, que involucren la implementación de protocolos en dispositivos programables, con la capacidad de poder ser integrados dentro de una red comercial. De igual forma, se genero un aporte académico, en áreas de desarrollo profesional como informática industrial, instrumentación electrónica y automatización de procesos dentro de la Universidad Industrial de Santander.

1.1. Organización del libro

El capitulo 2 realiza una presentación del protocolo AS-i, junto con los componentes que conforman la red. También se muestran las características de la señal, las estructuras de los mensajes, los tipos de errores y las especificaciones más importantes referentes al esclavo.

El capitulo 3 se presenta una descripción del diseño de la interfaz física del esclavo. En este se muestran las etapas propuestas para su puesta en funcionamiento, al igual que las características de los dispositivos utilizados en su implementación.

El capitulo 4 se desarrollan los algoritmos necesarios para el funcionamiento lógico del esclavo, recepción y transmisión de datos. De igual forma, se explica el manejo de los periféricos del microcontrolador utilizado para el procesamiento y se presenta los diagramas de flujo de cada uno de los programas realizados.

En el capitulo 5 se muestran los resultados de las pruebas realizadas al esclavo, en el se registran los valores experimentales obtenidos funcionado dentro de la red AS-i, y se comparan

estas características con los valores estipulados por la norma.

Finalmente en el capitulo 6, se condensan las experiencias, resultados obtenidos, conocimientos adquiridos y recopilados, en forma de observaciones, conclusiones y recomendaciones para trabajos futuros.

2

Protocolo *AS-Interface*

AS-Interface es un bus de campo que ha sido utilizado desde 1994, para la conexión de sensores/actuadores digitales y analógicos a una red industrial. Este bus fue especialmente diseñado para el nivel más bajo de la pirámide de automatización, en donde permite adquirir información del proceso y enviar ordenes de control a los elementos conectados a la red; su principal característica y ventaja, es la transmisión de datos y alimentación a través de la misma línea, además, las especificaciones del protocolo *AS-Interface* están abiertas a dominio público, en las normas IEC[1] 62026-2 y EN50295 [9, 11].

2.1. Componentes de la red *AS-i*

Una red AS-i tiene cuatro componentes básicos que permiten el éxito de su funcionamiento; asociados a estos componentes se encuentran tres interfaces físicas que facilitan la operación entre ellos y los niveles superiores, como se muestra en la figura 2.1. A continuación se describe brevemente cada uno de estos elementos y las interfaces respectivas.

- **Interfaz 1:** Esta interfaz permite la conexión del esclavo con los actuadores, sensores u otros elementos que hacen parte del proceso. Posee puertos que pueden ser configurados como entrada, salida o de forma bidireccional; además define los tiempos de la señal y provee la alimentación de los actuadores/sensores que se encuentren conectados a ella.

- **Esclavo:** Permite el monitoreo e intercambio de datos entre el maestro y los sensores/actuadores de la red, además asegura que el mal funcionamiento de un sensor, actuador u otro elemento no altere el funcionamiento normal de la misma.

[1] *International Electrotechnical Commission(IEC)*, norma americana.

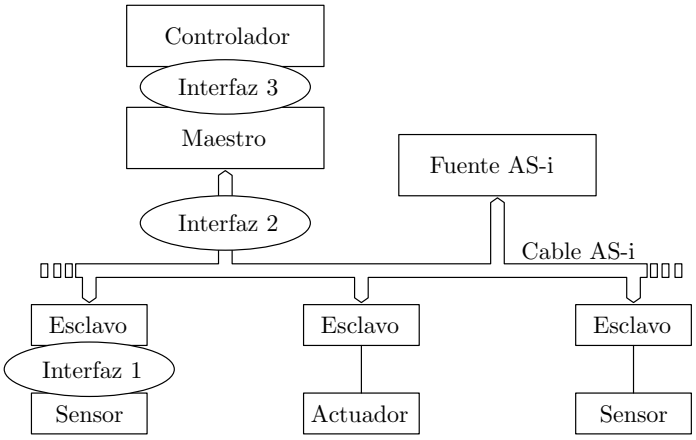


Figura 2.1: Componentes básicos de una red *AS-Interface*

- **Interfaz 2:** Se encarga de proporcionar todos los requerimientos lógicos, físicos y mecánicos necesarios para el intercambio de datos y la distribución de potencia en la red.

- **Fuente *AS-i*:** Proporciona potencia DC para todos los miembros de la red, a través de la línea *AS-i*. Este elemento de manera física forma parte de la interfaz 2.

- **Maestro:** Es el encargado de organizar y monitoriar el funcionamiento de la red, controlando el proceso de comunicación a través de requerimientos a cada uno de los esclavos, y la recepción respectiva de sus respuestas con el fin de ser procesadas en un nivel superior.

- **Interfaz 3:** Se presenta entre el host (controlador) y el maestro *AS-interface*; esta interfaz se encarga de proveer todas las funciones usadas por el host para acceder al maestro, y de esta manera poder enviar y recibir datos cíclicamente hacia y desde el esclavo. Además, permite al host controlar el comportamiento del maestro y por ende el comportamiento del sistema *AS-interface*.

- **Cable *AS-i*:** Es el camino a través del cual viajan los datos y la alimentación por toda la red.

2.2. Características

2.2.1. Sistema de transmisión *AS-i*

El sistema de transmisión *AS-i* permite la comunicación entre los esclavos y el maestro de la red, a través de un cable a 2 hilos sin apantallamiento. A través de este cable se transmiten simultáneamente los datos de proceso y la alimentación de los sensores/actuadores conectados a la red.

Algunas de las características principales del protocolo *AS-interface* se consignan en la Tabla 2.1 [8, 9, 11].

Tabla 2.1: Características principales de las redes *AS-i*

Característica	**Descripción**
Velocidad de transmisión	166 $\frac{2}{3}$ *kbps*
Tiempo de bit(TB)	6 μs
Número de esclavos	31 en modo normal
Tiempo máximo de ciclo en modo normal	5 ms
Detección de error	Bit de paridad
Correción de error	Reenvio de datos por parte del maestro
Acceso al medio	Sondeo cíclico por parte del maestro
Tipo de bus	Maestro-esclavo, monomaestro[2]

Características de la señal

En el proceso de comunicación a través del cable, la información a transmitir es codificada con el formato Manchester II mediante la modulación APM[3](*Alternate Pulse Modulation*) [9, 11]; los pulsos resultantes pueden ser modelados idealmente, tal como se menciona en [11],

[2]Por cada red *AS-i* existe solo un maestro que controla el intercambio de datos; este maestro llama consecutivamente todos los esclavos *AS-i* y espera su respuesta.

[3]La modulación APM transforma una señal digital en pulsos de amplitud positiva y negativa, que representan los flancos de bajada y subida de la señal, respectivamente.

por la ecuación,

$$u(t) \approx \pm U_{send} sen^2\left(\frac{2\pi}{6\mu s}t\right) \quad con \quad U_{send} \approx 2\ [V]$$

La señal resultante se muestra en la figura 2.2. Esta señal es transmitida a una velocidad de 166 $\frac{2}{3}$ $kbps$, con un tiempo de bit(T_{bit}) de 6 μs.

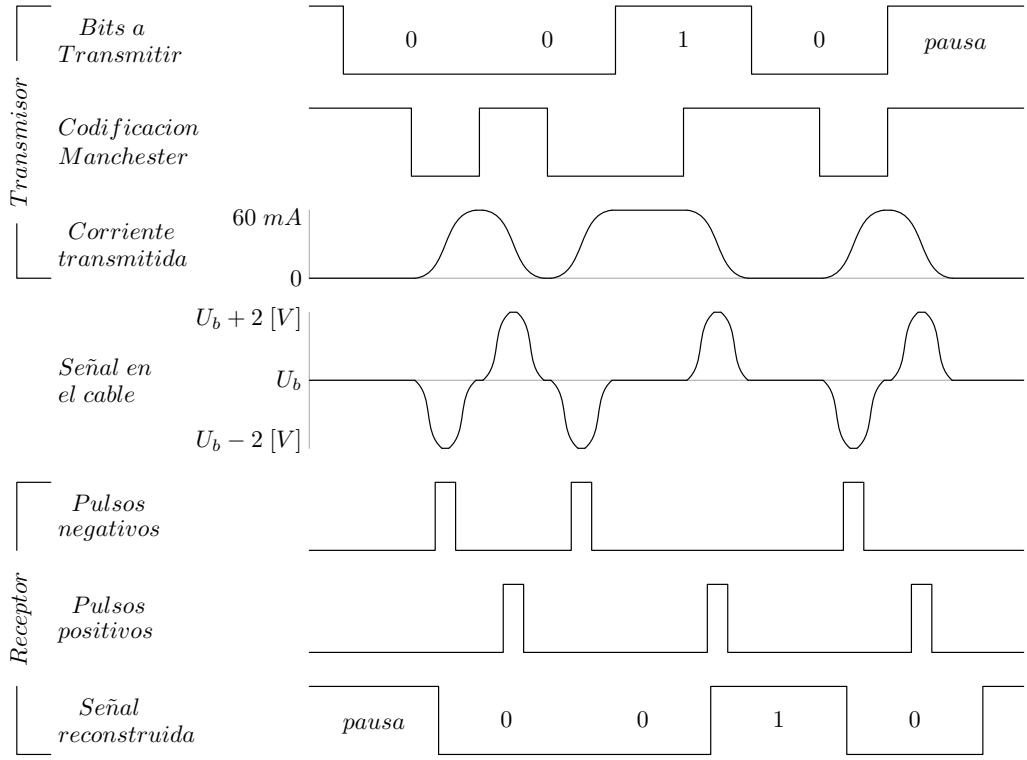


Figura 2.2: Formas de onda de la señal *AS-i*

2.2.2. Estructura de los mensajes

La estructura de los mensajes para la petición del maestro y la respuesta del esclavo, se muestran en las figura 2.3a y 2.3b respectivamente, y sus especificaciones se muestran en la tabla 2.2.

Como se puede observar en estas figuras, tanto las peticiones de los maestros como las respuestas de los esclavos poseen un bit de inicio en cero, y un bit de fin en 1. Adicionalmente,

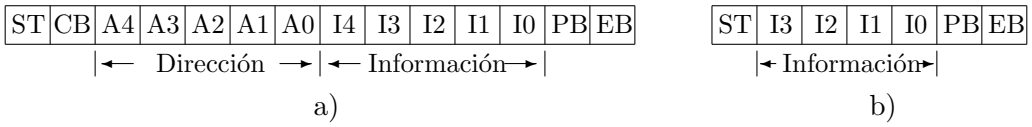


Figura 2.3: Estructura de los mensajes: a) petición del maestro b) respuesta del esclavo

Tabla 2.2: Significado de cada bit

Bit	**Significado**	**Comentario**
ST	Bit de inicio	Identifica el inicio del mensaje. Siempre ST=0
CB	Bit de control	Identifica el tipo de pregunta del maestro
A4..A0	Dirección (5 bits)	Dirección de los esclavos
I4..I0	Información	Bits que contienen la petición del maestro
I3..I0	Información	Bits que contienen la respuesta del esclavo
PB	Bit de paridad	Bit de detección de errores
EB	Bit de fin	Identifica el final del mensaje. Siempre EB=1

ambos dispositivos utilizan un bit de paridad para la detección de errores, lo cual permite mayor seguridad en el intercambio de datos entre el maestro y el esclavo. Es importante anotar, además, que el protocolo maneja paridad par, esto es, el mensaje siempre tiene un número par de unos sin incluir los bit de inicio y fin.

2.2.3. Tipos de transacciones

La red AS-i posee cuatro tipos de transacciones, estas son: *Data_Exchange*, *Write_Parameter*, *Address_assignment* y comandos. A continuación se describe cada una de ellas y en la tabla 2.3 se muestran los respectivos patrones de bits que las identifican[4].

Data_Exchange

Permite al maestro manipular el estado de los actuadores y consultar el estado de los sensores conectados al esclavo. Inicialmente, el maestro transfiere 4 bits al *Data_Output register* del esclavo, luego, el esclavo transfiere este patrón de bits a los puertos de salida correspon-

[4]Es importante anotar que, en caso de falla en la comunicación con el esclavo, el maestro solo realiza una retransmisión por ciclo para cada uno de los esclavos.

dientes y muestra, opcionalmente, la validez de estos datos, mediante un pulso de 1 T_B[5] de duración en la salida *data strobe*. Por último, el esclavo debe cargar el patrón de bits de los puertos de entrada y salida dentro del registro de transmisión, y enviarlos al maestro.

Write_Parameter

Esta transacción permite al maestro manipular el registro de los parámetros del esclavo. Inicialmente, el maestro transfiere 4 bits de parámetros al *Parameter_Output register* del esclavo, luego, este transfiere el patrón de bits a los puertos de salida correspondientes y muestra, opcionalmente, la validez de estos datos, mediante un pulso de $1T_{bit}$ de duración en la salida *parameter strobe port*. Después de recibir los parámetros, estos deberán ser transferidos a los puertos de parámetros del esclavo; y por último, el esclavo debe cargar el patrón de bits de estos puertos, colocarlos dentro del registro de transmisión y enviarlos al maestro. Es importante anotar que para algunos perfiles de los esclavos, no se hace uso de estos parámetros, en la norma EN50295.

Address_Assignment

Esta transacción permite que el maestro asigne una dirección al esclavo que tenga dirección cero, y que este la almacene en la memoria no volátil. Inicialmente, cuando el esclavo recibe la dirección enviada por el maestro, responde con el patrón de bits "0110", lo cual confirma que la dirección esta siendo procesada y que se recibió sin errores. Posteriormente, la dirección asignada por el maestro debe ser almacenada en la memoria no volátil del esclavo en un periodo inferior a 15 ms, tiempo durante el cual el bit $S0$ del registro de estado debe permanecer en 1 y, adicionalmente, no se debe procesar el comando *Delete_Address* si es enviado por el maestro.

Comandos

A continuación se describen cada uno de los comandos posibles en una red AS-i:

- *Reset_ASI_Slave:* Reinicia la máquina de estados del esclavo(enviándolo al estado *INIT*) en un tiempo inferior a 2 ms, tiempo durante el cual el esclavo no debe responder

[5] 1 T_B equivale a un tiempo de $6\mu s$.

ninguna pregunta hecha por el maestro. Además, este comando permite recuperar la dirección del esclavo, si esta ha sido borrada con el comando *Delete_Address*.

- *Delete_Address:* Borra la dirección actual del esclavo, para permitir la asignación de una nueva dirección con la transacción *Address_Assignment*.
- *Read_I/O_Configuration:* Permite al maestro leer el registro *I/O_Configuration* del esclavo.
- *Read_Identification_Code:* Permite al maestro leer el registro *ID_Code* del esclavo.
- *Read_Status:* Permite al maestro leer el registro *Status* del esclavo.

El patrón de bits de cada una de las transacciones entre el maestro y el esclavo[6] se resume en la tabla 2.3, aqui se muestra el patrón de bits que conforma cada una de las transacciones, junto con el patrón de bits que conformaría la respuesta del esclavo.

Tabla 2.3: Patrón de bits que identifica cada una de las transacciones

	Maestro			**Esclavo**
Transacción	**CB**	$\mathbf{A_4...A_0}$	$\mathbf{I_4...I_0}$	$\mathbf{I_3I_2I_1I_0}$
Data _ Exchange	0		$0D_3...D_0$	$D_3...D_0$
Write _ Parameter	0		$1P_3...P_0$	$P_3...P_0$
Address_Assignment	0	00000	$A_4...A_0$	01110
Comandos				
Reset_ASI_Slave	1		11100	0110
Delete_Address	1		00000	0000
Read_I/O_Configuration	1		10000	I/O_Configuration
Read_Identification_Code	1		10001	ID_Code
Read_Status	1		11110	$S_3...S_0$

[6]No se incluye en esta tabla la transacción *Read_Reset_Status*, ya que ha dejado de ser utilizada en normas mas recientes.

2.2.4. Pausas en las transacciones

En el protocolo *AS-i* se manejan algunos tiempos de espera entre preguntas de maestro y respuestas de esclavo, estas esperas o pausas son descritas en la norma [11], asi:

- Pausa de maestro(*Master pause*): Durante este tiempo el esclavo procesa la pregunta del maestro y produce su respuesta.
- Pausa de esclavo(*Slave pause*): Después de recibir una respuesta de esclavo, debe existir un tiempo mínimo durante el cual no debe realizarse ninguna transmisión. Esta pausa debe ser de 1 a 2 TB.
- Enviar pausa(*Send pause*): Después de recibir una respuesta de esclavo, debe existir un tiempo mínimo durante el cual no debe realizarse ninguna transmisión. En operación normal con más de 30 transacciones por ciclo, esta pausa debe ser igual a una pausa de esclavo. Si existen 30 o menos transacciones por ciclos, esta pausa puede ser prolongada hasta 500 μs.
- *Slave response time-out*: Si el maestro no recibe ninguna respuesta del esclavo que consulta, dentro de un tiempo definido(*Slave response time-out*), el maestro debe finalizar la transacción o repetir la transmisión. Esta pausa debe ser máximo de 10 TB.

2.2.5. Detección de errores

Cualquier pregunta del maestro o respuesta de esclavo debe ser revisada en busca de errores de transmision, entre los cuales se encuentran[7]:

[7]Los errores mostrados en esta sección, son explicados desde el punto de vista del esclavo, por lo tanto se asume que el mensaje que se esta analizando en busca de errores, es una pregunta de maestro

Start_bit_error	Se presenta cuando el primer pulso que se recibe después de una pausa, no es de polaridad negativa.
Alternation_error	Se presenta cuando dos pulsos consecutivos tienen la misma polaridad.
No_information_error	Este error es detectado por el esclavo, cuando la pregunta del maestro no es recibida en una ventana de tiempo de $(78\ \mu s)^{+1\ \mu s}_{-0,5\ \mu s}$ a partir del bit de inicio.
Parity_error	Se presenta cuando la suma de todos los bits en 1 de la pregunta del maestro, omitiendo el bit de inicio y el de fin, es impar.
End_bit_error	Se presenta cuando el último bit de la pregunta el maestro no es de polaridad positiva.
Length_error	Se presenta cuando el esclavo recibe algún otro pulso después de recibir el bit de fin

2.3. Módulo esclavo

Los esclavos son módulos de E/S descentralizados que permiten la comunicación entre el maestro de la red y los sensores/actuadores del proceso, reconociendo la petición hecha por parte del maestro y devolviendo una respuesta adecuada [12]. Para esta comunicación, cada esclavo posee una dirección asociada para ser reconocido, la cual es asignada y administrada por el maestro de la red. Además, es posible integrar a la red hasta 4 sensores y 4 actuadores binarios por cada módulo conectado, lo que permite tener hasta 124 señales de entrada y 124 señales de salida por cada maestro, cuando se ha conectado un máximo de 31 esclavos.

2.3.1. Registros

Para el funcionamiento normal del esclavo, es necesario el manejo de banderas y registros internos que son almacenados en la memoria volátil del esclavo y que pueden ser perdidos en caso de una falla en la alimentación; estos registros son:

- *Address register (5 bits):* Contiene la dirección del esclavo[8].
- *I/O_Configuration register (4 bits)*: Almacena la configuración de los puertos de entrada/salida del esclavo, esto es, define cuales puertos son de entrada y cuales son de salida[9].
- *ID_Code register (4 bits)*: Contiene el *ID Code*[10] del esclavo.
- *Data_Output register (4 bits)*: Contiene la última información transmitida por medio de la transacción *Data_exchange*. Después de resetear o encender el esclavo, el registro debe tomar el valor por defecto 0xF[11].
- *Data_Input register (4 bits)*: Contiene la última información disponible en la interfaz 1[12].
- *Parameter_output register (4 bits):* Contiene la última información transmitida por medio de la transacción *Write_Parameter*. Después de resetear o encender el esclavo, el registro debe tomar el valor por defecto 0xF.
- *Receive register*: Contiene la última pregunta hecha por el maestro, con los bits de inicio y final omitidos.
- *Transmit register*: Contiene la respuesta del esclavo, con los bits de inicio y final omitidos.
- *Status register*: Este registro contiene 4 bits independientes ($S_3...S_0$) que muestran información sobre el estado del esclavo, como muestra la tabla 2.4.

[8]Esta dirección es cargada desde la memoria no volátil del esclavo durante su encendido y puede ser modificada por el maestro mediante la transacción *Address_assignment.*

[9]El *I/O_Configuration* y el *ID_Code* son fijados en la memoria no volátil del dispositivo en el proceso de fabricación y no pueden ser alterados; A su vez, los registros *I/O_Configuration* e *ID_Code* son cargados desde la memoria no volátil del esclavo durante su encendido y, de igual forma, no pueden ser alterados.

[10]El *ID code* permite distinguir entre esclavos que tengan la misma configuración de entrada/salida (*I/O_configuration*).

[11]Comúnmente, la salida lógica del esclavo es la inversa de la salida física del dispositivo.

[12]Esta interfaz es la que permite la conexión entre los sensores/actuadores y el esclavo.

Tabla 2.4: Significado de los bits del registro *Status*.

Bit	no	si	Característica
$S_0 =$	0	1	Dirección volátil
$S_1 =$	0	1	Error en bit de paridad
$S_2 =$	0	1	Error en bit de fin de trama
$S_3 =$	0	1	Error al leer memoria no volátil

- *Sync flag*: Esta bandera indica la sincronización del esclavo, con las preguntas del maestro.
- *Data_exchange_disabled flag*: Esta bandera es "1" al resetear el esclavo, y en este estado, no permite que el maestro realice la transacción *Data_Exchange*, y es "0" al recibir la primera transacción *Write_Parameter*.

2.3.2. Máquina de estados

Cada módulo esclavo debe cumplir unos estados específicos que se encuentran enfocados hacia un correcto funcionamiento de la red, estableciendo mecanismos para la administración del proceso de comunicación por parte del maestro [11]. En la figura 2.4 se presentan los siete estados que un esclavo debe cumplir cuando hace parte de una red *AS-i*.

A continuación se describen las acciones que el esclavo debe realizar en cada una de los estados, y que se encuentran definidas en la norma EN50295.

En el estado *INIT*,

- Inicializar los registros *Data_output* y *Parameter_output* con el valor 0xF.
- Reiniciar el registro *Status* a cero.
- Cargar la dirección, la configuración de E/S y el código ID del esclavo, desde la memoria no volátil hastas los registros apropiados.
- Asignar un uno la bandera *Data_exchange_disabled*.
- Ir al estado *ASYNC*

En el estado *ASYNC*,

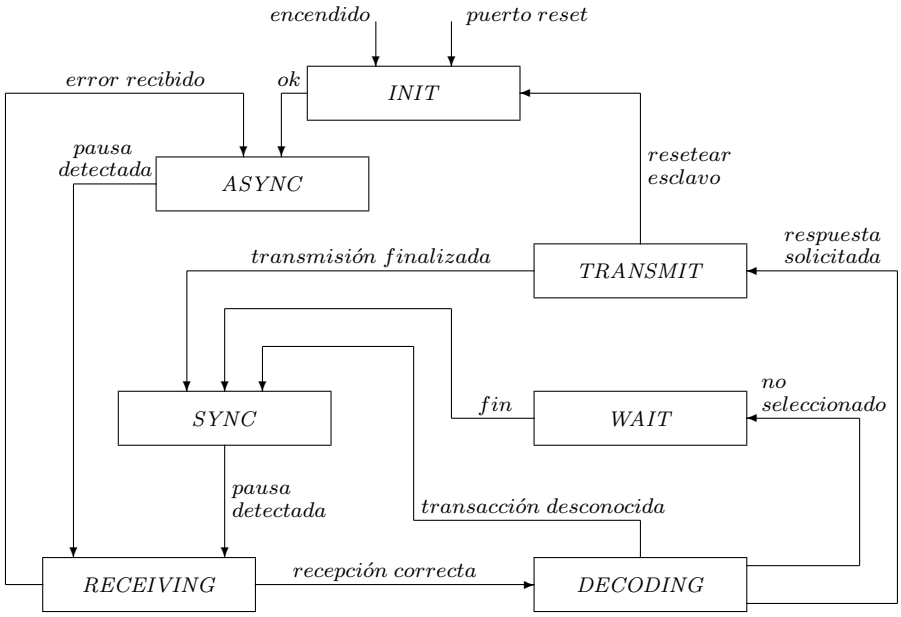


Figura 2.4: Máquina de estados de un esclavo *AS-i*.

- Asignar cero a la bandera *sync*.
- Revisar los datos de entrada hasta que se detecte una pausa.
- Ir al estado *RECEIVING*.

En el estado *RECEIVING*

- Esperar por el bit de inicio y cargar la pregunta del maestro dentro del registro *Receive*.
- Ejecutar las rutinas de verificación de errores.
- Chequear la línea en busca de una pausa de maestro.
- Si algún error es detectado, ir al estado *ASYNC*, sino ir al estado de *DECODING*.

En el estado *DECODING*

- Comparar la dirección recibida con la dirección interna, si no es igual, pasar al estado de espera.
- Analizar la información recibida y, si es una transacción desconocida, ir al estado síncrono.

- Si esta en 1 la bandera *data_exchange_disable* y el maestro envía una transacción *data_exchange*, ir al estado *SYNC*.
- RealizaR las tareas necesarias para la transmisión de la respuesta apropiada al maestro.
- Ir al estado de transmisión.

En el estado *WAIT*,

- Esperar la detección de una señal proveniente de otro esclavo.
- Si no se detecta ninguna señal, esperar entre 7 o 9 tiempos de bit.
- Si detecta una señal, esperar a que el esclavo deje de transmitir.
- Pasa al estado *SYNC*.

En el estado *TRANSMIT*,

- Esperar el final de la pausa del maestro.
- Enviar la respuesta del esclavo.
- Si es recibida una petición de reset, ir al estado *INIT*, sino ir al estado *SYNC*.

En el estado *SYNC*,

- Asignar un uno a la bandera *sync*.
- Revisar los datos de entrada hasta que se detecte una pausa.
- Ir al estado *RECEIVING*.

3

Diseño del Hardware

La implementación de un nodo esclavo *AS-i* implica la existencia de una capa física, que proporcione la capacidad de transmitir y recibir información a través de la red[1], cumpliendo con los requerimientos mínimos para poder establecer una comunicación [11]. Además, el esclavo debe permitir extraer los niveles de potencia de la línea, necesarios para su funcionamiento.

En este capítulo se presenta una descripción del diseño e implementación de cada una de las etapas que conforman la interfaz física realizada. Inicialmente se dará una explicación a nivel de diagrama de bloques de la estructura general abordada para el diseño del esclavo; posteriormente se presentará en detalle la implementación de cada una de las etapas, junto con los dispositivos utilizados para su correcto funcionamiento.

El diseño de la interfaz física fue realizado de acuerdo a las funciones básicas que debe presentar el dispositivo. Estas funciones fueron definidas por etapas, como se describe a continuación.

- Una etapa de control y procesamiento, que se encarga de la toma de decisiones del esclavo y la sincronización con el maestro de la red. Se encarga del control del conmutador, que permite alternar entre las etapas de recepción y transmisión respectivamente.
- Una etapa de recepción de datos provenientes de la línea *AS-i*, encargada de recibir las señales analógicas y convertirla en pulsos para su posterior procesamiento.
- Una etapa de transmisión encargada de transformar la señal de tensión proveniente de

[1]El protocolo *AS-i* se caracteriza por la transmisión de datos y alimentación a través de una línea de dos hilos.

la etapa de control en señales de corriente[2], y posteriormente inyectar esta corriente en la línea *AS-i*, para responder las peticiones realizadas por el maestro.

- Una etapa de conmutación, regulada por la etapa de control, que permita alternar entre la etapa de transmisión y recepción, dado que son dos funciones independientes.
- Una etapa de alimentación, capaz de extraer la potencia de la línea *AS-i* y entregarla a las etapas mencionadas anteriormente para su adecuado funcionamiento.

La figura 3.1 muestra el diagrama de bloques del esclavo *AS-i*, con cada una de las etapas definidas anteriormente, donde se observa como el bus *AS-i* es conectado al esclavo y es enlazado internamente hacia dos bloques diferentes:

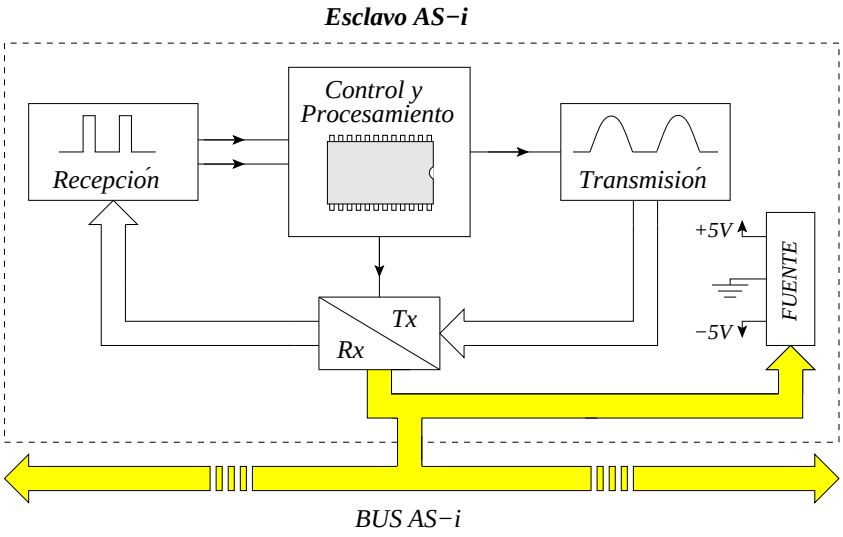


Figura 3.1: Diagrama de bloques del Esclavo *AS-i*.

El primero de ellos es el bloque de conmutación, que inicialmente se encontrará habilitado para que la etapa de recepción lleve la información proveniente de la red a la etapa de control. Una vez la información es procesada, la etapa de control se encarga de generar una transmisión de datos si hay algún tipo de requerimiento por parte del maestro, para lo cual generará la señal voltaje con la forma de onda de corriente a la etapa de transmisión y posteriormente habilitará el conmutador para inyectar la corriente a través de la línea.

[2]La transmisión de datos debe ser realizada a través de una señal de corriente de acuerdo a lo establecido por el protocolo *AS-i*.

El segundo bloque al cual llega la línea *AS-i*, corresponde a la etapa de alimentación, allí las componentes de señal que contengan información serán eliminadas a través de un filtro de tipo pasa bajo, obteniendo una tensión DC que se utilizará para proporcionar los voltajes de alimentación para las demás etapas internas.

A continuación se dará una explicación más detallada del diseño de cada bloque y una breve descripción de los dispositivos que fueron utilizados para su implementación.

3.1. Etapa de control y procesamiento

Esta etapa proporciona el soporte lógico necesario para el funcionamiento del esclavo. Permite procesar la información proveniente de la etapa de recepción, discriminando en que momento debe o no realizarse una transmisión. Cuando sea necesario transmitir información, se encarga de generar la señal modulada de tensión que será convertida en corriente para su posterior inyección a línea.

También se encarga de la sincronización con el maestro y del intercambio de datos y parámetros con el mismo. Se encuentra conectada con las etapas de transmisión, recepción y conmutación, controlando a esta última para habilitar cualquiera de las dos primeras.

En síntesis, la etapa de control y procesamiento se encarga de la toma de decisiones en el esclavo, utilizando las etapas siguientes como herramientas para hacer efectivas cada una de sus disposiciones. En esta etapa se habilitaron dos entradas digitales para la recepción de datos, una salida analógica para la transmisión y una salida digital para el control del conmutador. También se dispuso su alimentación a 5 V con respecto al nivel de tierra interno manejado por el esclavo. Su programación lógica se presenta en el capitulo 4, con cada una de los algoritmos utilizados para su desarrollo.

El microcontrolador utilizado para la implementación del funcionamiento lógico del esclavo, es el MC9S12E128 de 16 bits. Este microcontrolador posee 128 kB de memoria *flash*, 8kB de memoria RAM, 3 módulos SCI, un módulo SPI, dos módulos DAC, un módulo ADC de 16 canales y 10 bits de resolución, tres temporizadores de 4 canales cada uno, y otros periféricos.

Adicionalmente, el microcontrolador permite frecuencias de bus de hasta 25 MHz que permiten que el esclavo implementado cumpla con la pausa de maestro especificada en la

seccion 2.2.4.

3.2. Etapa de recepción de datos

Esta etapa se encarga de recibir los patrones analógicos de tensión provenientes de la línea *AS-i* y transformarlos en niveles lógicos para ser entregados a la etapa de control. Los pulsos analógicos[3] recibidos son convertidos en pulsos [4] positivos o negativos, los cuales son ingresados a los puertos del microcontrolador para posteriormente ser procesados. El mecanismo general de recepción es realizado siguiendo las recomendaciones de la norma EN50295, la figura2.2 ilustra las señales recibidas a partir de la tensión de la línea *AS-i*.

La etapa de recepción fue diseñada en cuatro bloques funcionales como se presenta en la figura 3.2. El primero de ellos es un bloque de desacople, el segundo una bloque de amplificación y adecuación de señal, el tercero un bloque de comparación y por ultimo una adecuación final de los pulsos generados por el comparador, a través de un aislador digital.

El primer bloque realiza un proceso de desacople de la señal diferencial a través de un transformador, la señal proveniente de la línea es referenciada al nivel de tierra interno del dispositivo esclavo. El bloque fue implementado con el transformador $WB2010$ [6] con una relación de $1:1$ y una corriente de saturación del núcleo de 250 mA.

Una vez la señal de la línea es desacoplada, el nivel de tensión es amplificado para obtener señales con un rango de excursión más amplio, facilitando el proceso de comparación. La señal es amplificada a través de dos circuitos diferentes: Uno utiliza una configuración no inversora, que se encarga de la señal que será utilizada para obtener los pulsos positivos, y el otro una configuración inversora de la cual serán obtenidos los pulsos negativos. Estas etapas de amplificación hacen posible que se utilice un valor único de referencia para el proceso de com-

[3]Los pulsos analógicos son aquellos niveles de tensión que aparecen en la línea *AS-i* como consecuencia de la corriente que circula por la red y se pueden representar como una función seno al cuadrado para cada pulso como se presenta en la sección 2.2

[4]De aquí en adelante la palabra pulso hará referencia a niveles lógicos de 5 V, un pulso positivo será aquel que se produce a partir de un pulso analógico positivo, de igual manera un pulso negativo es un nivel de 5 V producido a partir de un pulso analógico negativo, la figura 2.2 ilustra el significado de cada pulso.

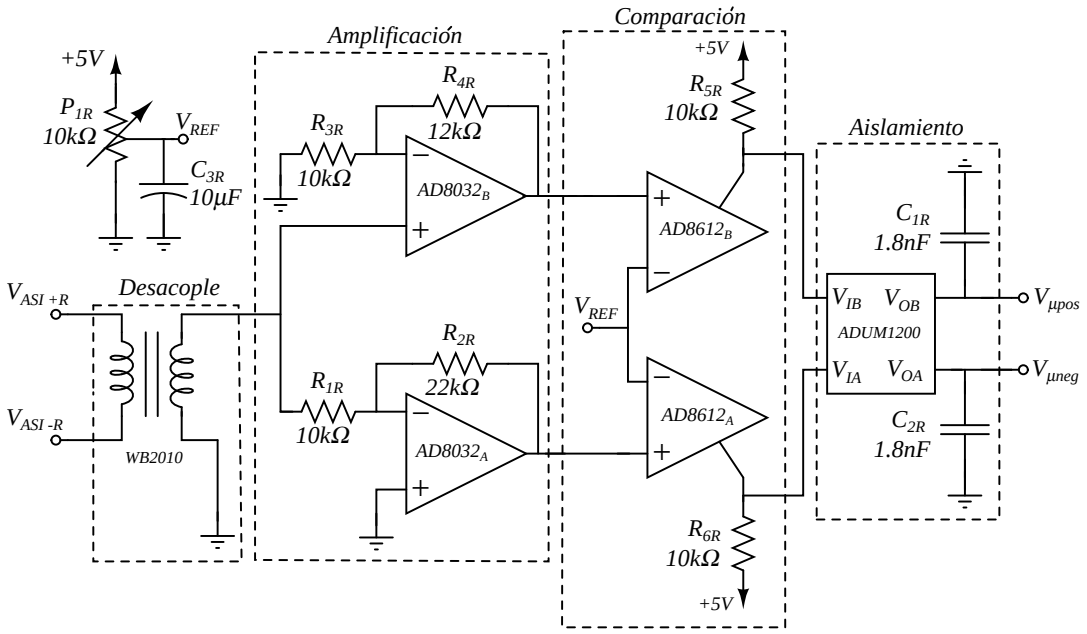


Figura 3.2: Etapa de recepción de datos.

paración, y un mismo mecanismo para obtener los pulsos tanto positivos como negativos. La implementación de este bloque fue realizado utilizando el circuito integrado $AD8032$ [1], que se caracteriza por tener dos amplificadores operacionales de propósito general y una respuesta en frecuencia acorde con las características de la señal, mostradas en el capitulo 2. Los valores de resistencia fueron escogidos para una ganancia de 2.2 V/V en las dos etapas de amplificación, de esta forma las señales de salida podrán obtener un rango máximo de excursión menor 10 V_{p-p}, que sería el valor en el cual se saturarían los amplificadores operacionales, y entregarán una señal de mayor amplitud que facilitará el proceso de comparación.

El bloque de comparación es el más importante de esta etapa, permite discriminar pulsos analógicos positivos y negativos convirtiéndolos en niveles lógicos para ser enviados al microcontrolador. Utiliza dos canales diferentes, que toman la señal de las etapas de amplificación nombradas anteriormente. Estas señales son comparadas con un nivel de referencia, que corresponde a la menor amplitud pico que deben tener los pulsos analógicos para ser válidos[5], la

[5]La norma EN50295 no establece un valor mínimo para los pulsos analógicos válidos, sin embargo con el fin de ajustarnos a un valor adecuado, se tomo como referencia los valores especificados en la norma IEC62026-2

cual es de 1 V_p de acuerdo a la recomendación de [9]. Debido al proceso de amplificación que se realiza previamente, el nivel de referencia de comparación es un valor escalado del mínimo establecido, con lo cual el valor equivalente es de 2.2 V. Su implementación fue realizada con un divisor resistivo, utilizando un potenciómetro, que proporciona mayor flexibilidad para obtener el nivel tensión requerido.

El proceso de comparación fue realizado utilizando el circuito integrado $AD8612$ [2], que consta de dos comparadores de alta velocidad, con un retardo de 4 ns a ± 5 V, y con un ancho de banda de 100 MHz.

Finalmente el bloque de adecuación de pulsos es realizado por un aislador digital, que al ser saturado mejora la forma de los pulsos obtenidos en el bloque anterior y aisla en señal los puertos de entrada del microcontrolador de las salidas del comparador.

3.3. Etapa de transmisión de datos

La transmisión tanto en el maestro como en el esclavo debe ser implementada como una fuente de corriente de acuerdo a [11], en razón a que la aplicación de una señal de corriente a la impedancia vista por esclavo desde los terminales de conexión, garantiza la generación de los pulsos de tensión a través de la línea como se muestra en la figura 2.2, conservando de esta manera la integridad de la información trasmitida.

La forma de la señal de corriente es generada de manera indirecta a través del microcontrolador, utilizando la salida del conversor digital-analógico del mismo, como se muestra en el capitulo 4. Esta señal se presenta como un nivel de tensión que toma valores de 0 V a 2 V, y presenta la forma de onda de la corriente requerida.

Sin embargo, dado que la máxima corriente que puede circular a través de la salida del puerto DAC del microcontrolador es 40 mA, de acuerdo a su hoja de datos, fue necesario implementar un circuito de acople entre el microcontrolador y la fuente de corriente. Lo anterior fue realizado utilizando un amplificador operacional configurado como seguidor, cuya propiedad de alta impedancia de entrada, exige una corriente mínima al microcontrolador, protegiendo su puerto de salida contra una sobre-corriente que produzca su deterioro. De esta manera lo que se obtiene es una ganancia de potencia, que asegura la transferencia del nivel

necesario de voltaje a la etapa siguiente.

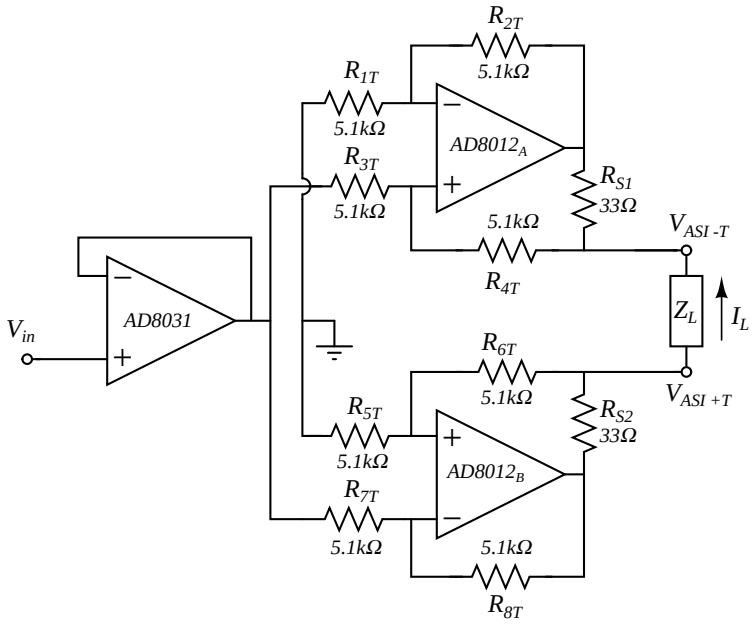


Figura 3.3: Etapa de transmisión de datos.

Una vez acoplada la señal del microcontrolador se procedió a implementar una fuente de corriente controlada por tensión, con capacidad para alimentar una carga diferencial, representada por la impedancia vista por el esclavo a través de los puertos $ASi+$ y $ASi-$. La fuente de corriente debe ser independiente de la carga y debe presentar una resistencia alta de salida. Con estas características la topología escogida fue una fuente de corriente *Howland* diferencial o complementaria, la cual es mostrada en figura 3.3, junto con el seguidor de tensión mencionado anteriormente.

La fuente de corriente *Howland* balanceada[6] se caracteriza por la independencia de la corriente de salida respecto a la carga, de esta manera se garantiza un flujo de corriente constante, aun si las condiciones de la impedancia de salida varían. La función de transferencia para la configuración diferencial depende del cumplimiento simultaneo de las ecuaciones [3.1] y [3.2]

$$I_l = \frac{V_{in}}{R_{s1}} \tag{3.1}$$

[6]La condición de balance para la *Howland* diferencial implica que $\frac{R_{2t}}{R_{1t}} = \frac{R_{4t}}{R_{3t}} = \frac{R_{6t}}{R_{5t}} = \frac{R_{8t}}{R_{7t}}$

$$I_l = \frac{-V_{in}}{R_{s2}} \tag{3.2}$$

La ecuación [3.1] hace referencia a la configuración no inversora, mientras que la ecuación [3.2] define la función de transferencia para la fuente inversora. Esta condición implica que R_{s1} debe ser igual a R_{s2} para asegurar que la corriente que entra a la carga sea la misma corriente que sale de ella. La fuente superior que se muestra en la figura 3.3 presenta una configuración no inversora, mientras que la fuente inferior se comporta como una fuente inversora de corriente, de esta manera se garantiza el carácter complementario de la fuente implementada.

Cabe destacar que con un nivel fijo de tensión a la entrada de la fuente *Howland*, la corriente de salida dependerá únicamente de R_{s1} y R_{s2}. Para la implementación del esclavo, el valor de tensión suministrado por el microcontrolador es de 2 V_{p-p}, de tal manera que R_{s1} y R_{s2} se ajustaron a 33 Ω, generando una corriente de 60 mA, siendo un punto medio entre los niveles establecidos por el protocolo en [11]. Adicionalmente la resistencia de salida del circuito en condiciones de balance garantizan una impedancia idealmente infinita[7], y por lo tanto la máxima transferencia de la corriente generada.

Las resistencias R_{nt} son resistencias de precisión de una tolerancia del 1 % y un valor de 5,1 $k\Omega$, que garantizan un balance adecuado para el circuito. Este valor fue escogido en base a un análisis de resultados experimentales que se muestra en [7], que muestran un comportamiento adecuado de la *Howland* para resistencias de este valor. Con valores más elevados disminuye la resistencia de salida al aumentar el margen de precisión de las resistencias y con resistencias de menor valor disminuye la resistencia de entrada del circuito. Los amplificadores operacionales fueron seleccionados tomando como referencia las siguientes características :

- La primera de ellas es la capacidad de corriente de salida que debe tener el dispositivo, el cual deben ser superiores a 70 mA.
- El *slew rate* del amplificador debe ser superior a 1,05 $V/\mu s$, que es valor obtenido al ser calculado con una señal en la salida con las características descritas en el capítulo 2. Con un *slew rate* superior a valor se garantiza un funcionamiento adecuado del circuito.

[7]El cumplimiento de esta característica depende de la precisión de las resistencias de realimentación de los amplificadores operacionales R_{nt} para n de 1 a 8.

- El ancho de banda del amplificador debe ser superior a 167 kHz cuando este se encuentre realimentado, con las condiciones de fuente de corriente especificada.
- Alimentación dual de $\pm 5\ V$.

Con estas condiciones, una vez seleccionado y estudiado un conjunto de amplificadores disponibles en el mercado, se escogió el $AD8012$ [3], dado que se ajusta a los requerimientos necesarios para la implementación de la fuente de corriente.

3.4. Etapa de conmutación

La función principal de la etapa de conmutación es alternar entre la recepción y transmisión de datos, de acuerdo a la señal de control del microcontrolador. Un nivel lógico alto permanente en la entrada de control del conmutador habilita la etapa de recepción, mientras que un nivel lógico bajo habilita la etapa transmisión.

Asimismo, esta etapa incluye dos condensadores C1 y C2 como se ilustra en la figura 3.4, que actúan como filtros de desacople de los niveles de tensión DC provenientes de la línea *AS-i*, de esta manera se asegura un valor de tensión DC nulo en los terminales del conmutador, y aumentando el grado de protección del dispositivo, ya que el carácter de baja impedancia entre los terminales de entrada y salida cuando el interruptor se cierra, lo hacen vulnerable a altos niveles de voltaje DC entre sus puertos, que ocasionen corrientes elevadas y por tanto el daño del integrado.

El dispositivo que se utilizó para la implementación fue el interruptor analógico $ADG787$ de dos canales, con una capacidad de corriente de 300 mA, una resistencia de encendido de 2,5 Ω y un tiempo de retardo menor a 20 ns [4]. La figura 3.4 presenta implementación completa de esta etapa, junto con la definición de sus entradas y salidas, el bloque funcional del $ADG787$ es mostrado para un nivel alto en los puertos de control (IN1 e IN2).

3.5. Etapa de Alimentación

La extracción de potencia se realizó mediante un filtrado inicial que permitió aislar los niveles de señal de información de los circuitos de regulación, esto se logró a través de un

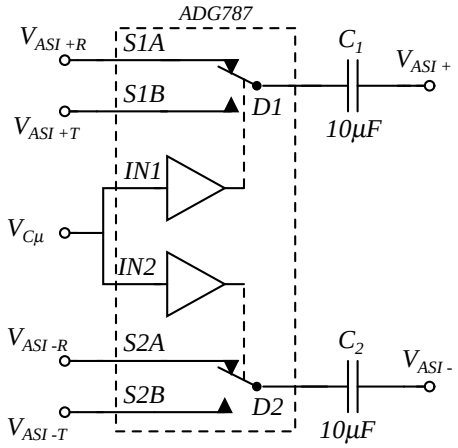


Figura 3.4: Conmutador Analógico controlado digitalmente.

par de inductancias de 1,8 mH, que actúan como un camino de alta impedancia en señal y como un corto circuito para los niveles de DC. Una vez se obtiene el nivel diferencial de aproximadamente 30 V, se establece la referencia de tierra de todos los dispositivos como se muestra en la figura 3.5. Para efectos de regulación y reducción de tensión se utilizó el $LM317$ para llevar la tensión a un valor de 12 V, luego en cascada se conectó un $LM7805$ que permitió obtener el nivel de referencia positiva de 5 V.

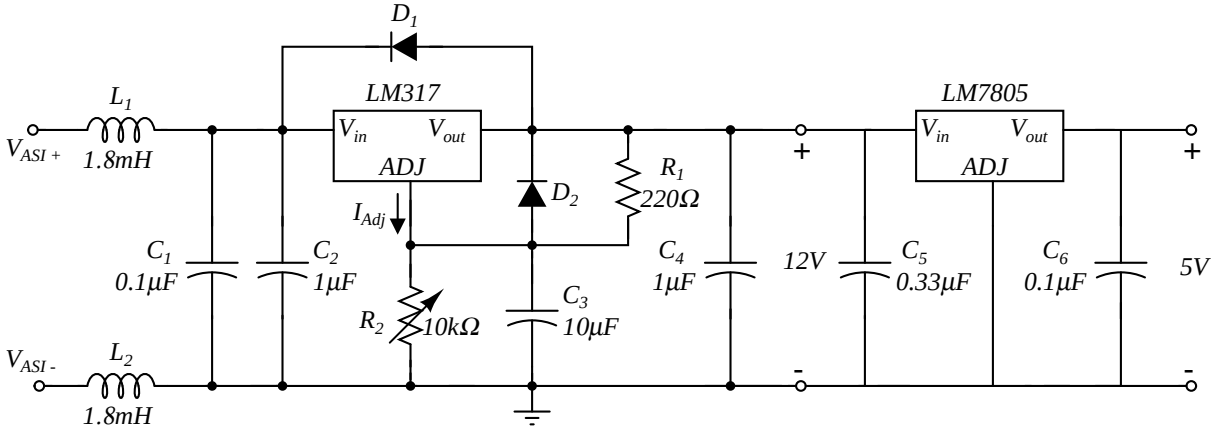


Figura 3.5: Etapa de extracción de potencia DC.

En cada una de las etapas de regulación, fueron utilizados condensadores electrolíticos y de tántalio para disminuir los niveles de ruido e interferencia ocasionados por fluctuaciones en la fuente primaria de tensión.

Debido a la inexistencia de una referencia negativa de voltaje, fue necesario recurrir a un inversor de tensión para obtener la tensión negativa a partir de un valor de tensión positiva. Para esto se utilizó el $LT1054$, que es un conversor de voltaje a partir de la conmutación de condensadores, configurado como se muestra en la figura 3.6. La tensión obtenida fue una referencia de $-4{,}9\ V$, utilizando una tensión de entrada de $5\ V$, los valores medidos son mostrados en el capítulo 5.

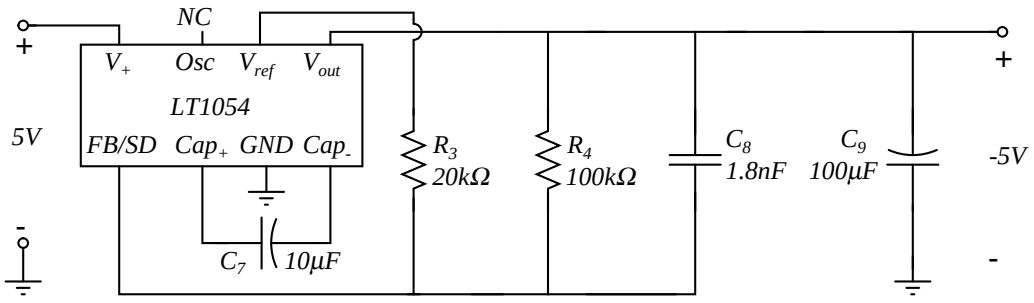


Figura 3.6: Inversor de tensión LT1054.

Dado que nuestra aplicación requiere una corriente proveniente de la fuente de tensión negativa inferior a 20 mA y con el fin de utilizar un circuito sencillo para su generación, se estableció un rango de tolerancia de $0{,}2\ V$ en el peor de los casos para la disminución de la tensión de salida, de tal manera que la referencia de tensión obtenida a plena carga de los requerimientos del esclavo es de $-4{,}7\ V$, sin que esto afecte el funcionamiento adecuado de los circuitos que la utilizan.

El resultado final una vez culminó el proceso de implementación, fue una interfaz física genérica con la capacidad de soportar cualquiera de los perfiles de esclavo descritos en [11], que utiliza una entrada para el bus $AS\text{-}i$, y a la cual se le pueden habilitar cuatro puertos para comunicación externa con dispositivos de entrada-salida, para el intercambio de información digital. Una vez la interfaz física ha sido definida, se mostrará de forma más detallada el funcionamiento lógico del esclavo en el capítulo 4. Allí serán presentados los algoritmos necesarios para recepción, transmisión y control del conmutador, que permitan finalmente la implementación integral del esclavo $AS\text{-}i$.

4 Diseño del Software

En el capítulo 3 se describe la forma de convertir la señal de voltaje disponible en un bus *AS-i*, en pulsos digitales. En este capítulo por su parte, se explica como procesar estos pulsos digitales con el fin de emular el funcionamiento lógico de un esclavo *AS-i* con perfil S-3.0, fundamentado en las especificaciones de manejo de registros, manejo de errores, tiempos de ejecución y funcionamiento lógico expuestos en el capítulo 2.

El programa implementado fue dividido en varios archivos fuente en los que se agruparon las funciones afines. También se utilizaron diversos archivos de cabecera para la definición de variables globales, en cada uno de estos archivos fuente. De esta manera, se obtuvieron archivos de menor extensión que facilitaron la depuración de los programas y simplifican en gran manera, el estudio y comprensión posterior de este proyecto.

Por otro lado, se usó compilación condicional mediante ordenes al preprocesador[1], con el fin de permitir la utilización de los archivos fuente y los respectivos archivos de cabecera, en otros programas.

En este capítulo se inicia con la descripción de la máquina de estados implementada en el proyecto, junto con el resumen de los archivos creados y las funciones definidas en cada uno de ellos. De igual forma se describen cada una de los estados que conforma esta máquina de estados y se da una breve explicación de los diagramas de flujos implementados.

[1]En [5] se explica que "El preprocesador de C es un procesador de texto que manipula el texto de un fichero fuente como parte de la primera fase de traducción del código fuente de un programa C y antes de que la compilación propiamente dicha comience".

4.1. Máquina de estados

En el archivo Esclavo_ASi.c se encuentra el programa principal de este proyecto, donde se implementaron las 7 etapas de la máquina de estados del esclavo especificadas en la sección 2.3.2, y se hace el llamado de las funciones de inicialización de los periféricos.

Cada una de las etapas que conforman esta máquina de estados, fue realizada mediante una función que tiene el mismo nombre de la etapa, seguido de la letra f. En la tabla 4.1 se especifican cada una de estas funciones, las etapas a las que representan y los archivos fuente en los que fueron definidas.

Tabla 4.1: Archivos fuente creados en el proyecto

Archivo fuente	**Funciones**	**Estado**
Esclavo_ASi.c	Máquina de estados	
Funciones.c	asyncf(), syncf(), waitf() y receivingf()	*ASYNC*, *SYNC*, *WAIT* y *RECEIVING*
Transmision.c	transmitf()	*TRANSMIT*
Decoding.c	decodingf()	*DECODING*

De otro lado, los datos que son compartidos por varias funciones fueron definidos como variables globales, lo que permite hacer el llamado de estas subrutinas sin utilizar argumentos; evitando así, la pérdida de tiempo adicional que representaría el almacenamiento de estos argumentos por cada una de las funciones. Estas variables globales son definidas en el archivo de cabecera Datos_ASi.h y se resumen en la tabla 4.2.

Además de definir las variables globales que contienen los registros propios del protocolo, fue necesaria la utilización de un conjunto de indicadores o banderas, que permitiesen a las etapas informar al programa principal, el resultado de su ejecución. Este conjunto de indicadores fue agrupado en un campo de *bit* llamado "bandera", cuyo contenido se muestra en la tabla 4.3, y son utilizadas por el programa principal para decidir cual es la siguiente etapa a ejecutar en la máquina de estados, como se ilustra en el diagrama de flujo de la figura 4.1.

Al llegar a este punto, es importante aclarar que la información disponible en el bus es

Tabla 4.2: Variables globales utilizadas en el proyecto

Tipo	**Nombre**	**Descripción**
byte	Transmit	Registro *Transmit*
byte	address	Registro *address*
byte	IO_configuration	Registro *IO_configuration*
byte	ID_code	Registro *ID_code*
byte	Data_output	Registro *Data_output*
byte	Data_input	Registro *Data_input*
byte	Parameter_output	Registro *Parameter_output*
byte	addressv	Dirección del esclavo en memoría volátil
byte	IO_configurationv	*IO_configuration* del esclavo, en memoría volátil
byte	ID_codev	*ID_code* del esclavo en memoría volátil
unsigned int	Receive	Registro *Receive*
campo de bit	Status	Registro *Status*

Tabla 4.3: Contenido del campo de bit `bandera`

Nombre del bit	**Descripción**
sync	1: Esclavo sincronizado
	0: Esclavo no sincronizado
data_exchange_disabled	1: Intercambio de datos deshabilitada
	0: Intercambio de datos habilitada
error	1: Error en la recepción
	0: Recepción sin errores
reset	1: Reiniciar esclavo
	0: No reiniciar esclavo
no_seleccionado	1: Esclavo no seleccionado
	0: Esclavo seleccionado
tran_desc	1: El mensaje recibido es desconocido
	0: El mensaje recibido es conocido
pa	1: Próximo pulso: pulso negativo
	0: Próximo pulso: pulso positivo

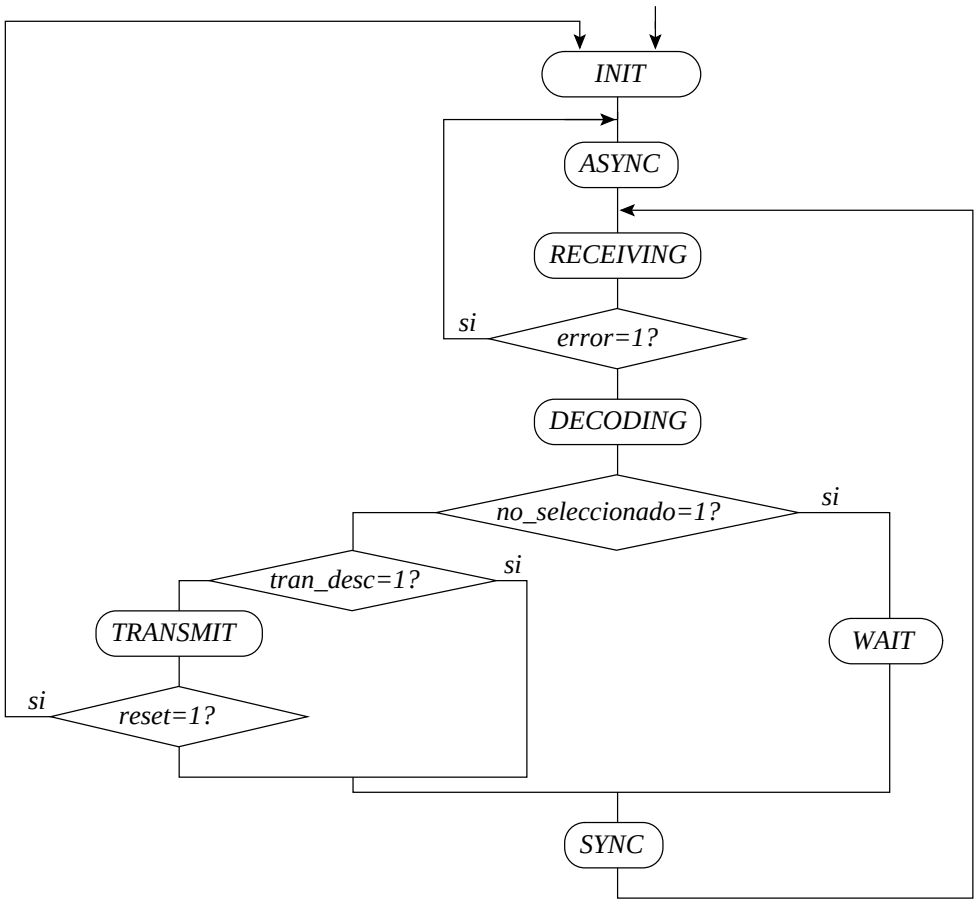


Figura 4.1: Diagrama de flujo de la máquina de estados del esclavo.

transmitida al microcontrolador en forma de pulsos negativos y positivos, como se describe en la sección 3.2; para los cuales, se han habilitado los puertos de entrada IOC06 e IOC07 respectivamente. Estos puertos corresponden a los canales 6 y 7 del temporizador TIM0 configurado en modo *input capture*, y fueron seleccionados por permitir la detección de flancos de subida en sus entradas, eliminando la dependencia del programa del ancho del pulso recibido [2], pero obligando al *hardware* de recepción, a generar pulsos digitales libres de *glitches* indeseados que puedan causar errores. En adelante, las variables `pos` y `neg` se utilizarán en los diagramas de flujo para representar la recepción de pulsos positivos y negativos respectivamente.

Finalmente, para utilizar la máquina de estados es necesario incluir en el archivo principal,

[2]El ancho del pulso de salida del hardware de recepción, depende de la amplitud de la señal sobre el bus ASi y del nivel de comparación.

el archivo de cabecera Datos_ASi.h, en donde se definen las variables globales y se declaran la funciones utilizadas en la máquina de estados.

4.2. *Init*

En esta etapa se inicializan los registros *Data_output* y *Parameter_output* al valor por defecto $0x0F$; se asigna el valor de 0 al registro *Status*; se activa la bandera *Data_exchange_disabled* y se transfieren los registros *address*, *I/O_configuration* e *ID_code* de la memoria no volátil a la memoria volátil.

4.3. *Async*

En esta etapa se asigna el valor de cero a la bandera *sync* y se detecta la presencia de una pausa entre los pulsos recibidos. Con este fin es utilizado el temporizador T_1, el cual se encarga de informar a la función *asyncf()*, la ausencia de pulsos en la entrada por un tiempo superior a 7,5 μs. Luego de detectar la pausa, la máquina de estados ejecuta la etapa *Receiving*, a la cual se ingresa justo antes de recibir el bit de inicio de la siguiente trama en el bus, aunque no se garantiza que esta trama sea una pregunta de maestro.

En concreto, esta etapa permite la sincronización del esclavo[3] mediante la detección de una pausa en la recepción de pulsos y la detección de errores implementada en la etapa de recepción, su diagrama de flujo se ilustra en la figura 4.2.

4.4. *Receiving*

La etapa de Recepción implementada, permite que el esclavo reciba correctamente la pregunta del maestro, y detecte cualquiera de los 6 tipos de errores especificados en la sección 2.2.5.

[3]Un esclavo *AS-interface* esta sincronizado, cuando ejecuta la etapa de recepción si hay una pregunta del maestro en el bus, y ejecuta la etapa de espera (*wait*) si hay una respuesta de otro esclavo en el mismo.

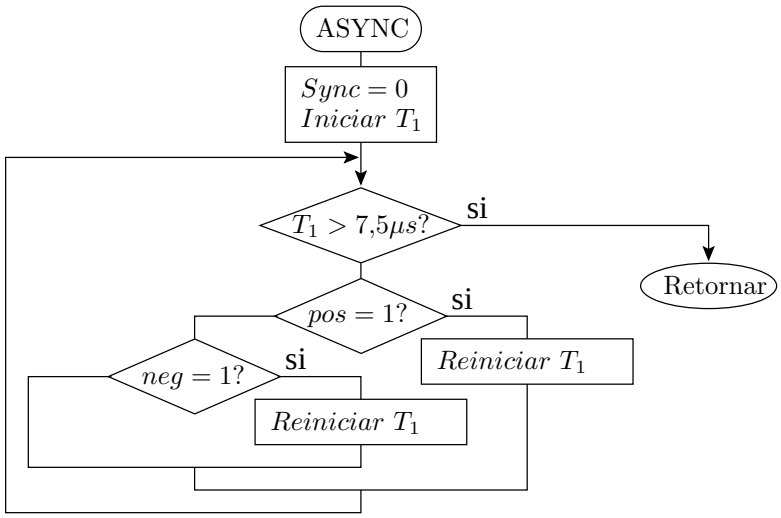


Figura 4.2: Diagrama de flujo de la etapa Async.

Cada una de estas rutinas de demodulación/decodificación[4] y detección de errores, es realizada de manera simultánea a la recepción de los pulsos, esto es, se realizan en los 3 μs que existe entre la recepción de dos pulsos seguidos. Por esta razón, fue necesaria la implementación de la etapa de recepción en lenguaje ensamblador, ya que permite la reducción de tiempos de ejecución en el programa, mediante la utilización al máximo de los registros de la CPU y de una mejor utilización del conjunto de instrucciones del microcontrolador.

Adicionalmente, como se explicará más adelante en esta misma sección, fue necesaria la utilización de un temporizador para identificar si la trama recibida es una pregunta del maestro o es una respuesta de otro esclavo, para facilitar la sincronización del dispositivo.

Para comprender la metodología utilizada en la recepción de los datos, se ilustra en la figura 4.3 la codificación *Manchester II* y la modulación APM aplicadas al mensaje "0011001". En esta figura se observa que cada bit de información es convertido en dos *bits* lógicos por la codificación *Manchester II*, de los cuales, el segundo posee el mismo valor que el *bit* de información original, como se muestra resaltado en gris en la figura 4.3. Este segundo *bit* es convertido por la modulación APM, en un pulso analógico positivo si es un uno, o en un pulso

[4]Se hace referencia aquí a rutinas de demodulación/decodificación de la señal de entrada, debido a que la señal de voltaje es convertida de manera directa a bits de información, sin realizar etapas intermedias de demodulación y posterior decodificación de la información demodulada.

analógico negativo si es un cero. Por otro lado, el primer bit es convertido en un descanso[5], un pulso positivo o un pulso negativo para mantener la alternancia de la señal de voltaje modulada.

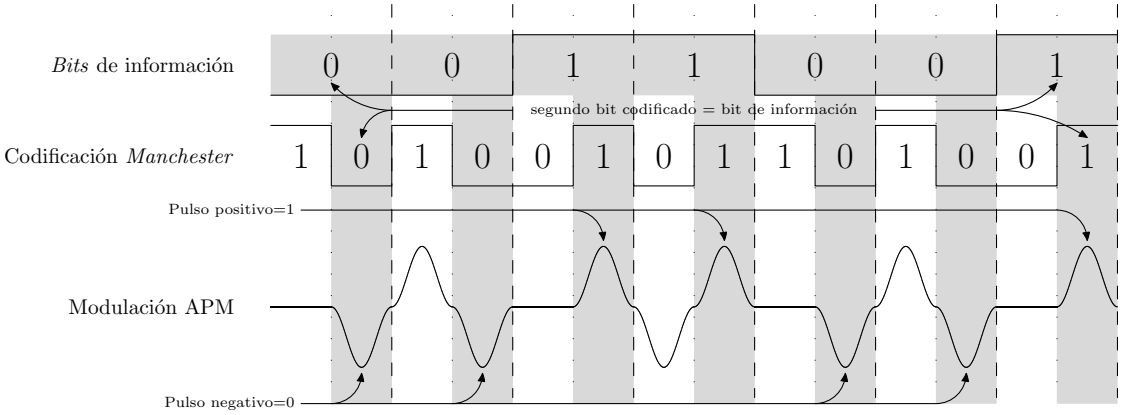


Figura 4.3: Codificación *Manchester* y modulación APM aplicados al mensaje "0011001".

En resumen, para demodular/decodificar la señal de voltaje en el bus, se deben revisar los pulsos de entrada que resultan de la modulación APM del segundo bit codificado, que en adelante serán llamados pulsos de información, en busca de la información transmitida; y se debe revisar la trama completa en busca de un error de alternancia. Para realizar estas revisiones de los pulsos de entrada, se utiliza el método de encuestamiento en lugar de interrupciones[6], ya que estas últimas, tienen un retardo adicional provocado por el almacenamiento del estado actual del microcontrolador, esto es, el almacenamiento de los registros de la CPU en la pila.

Para la detección de un descanso en la señal modulada se utiliza el temporizador T_2, que se encarga de informar la ausencia de pulsos en la entrada por un tiempo superior a 4 μs. Este valor de espera fue seleccionado, teniendo en cuenta que la señal de voltaje en el bus

[5]Un descanso es la ausencia de un pulso en el mensaje modulado y aparece cada vez que hay un cambio de valor en los *bits* de información que lo conforman.

[6]El método de encuestamiento aplicado, revisa constantemente el estado del indicador de recepción de flancos, del temporizador TIM0. Este indicador tiene el valor de "1çuando recibe un flanco de subida en la entrada, y mantiene su valor sin importar el estado de la misma. Por esta razón se puede afirmar que no existe pérdida de información al utilizar el método de encuestamiento en esta etapa, ya que el indicador del temporizador solo es reiniciado después de revisar su estado.

puede tomar un valor máximo de 4 V_p como se muestra en [9]; entonces, si se presenta un pulso de voltaje con este valor máximo, seguido de un pulso cuyo valor sea igual al nivel de comparación, los pulsos digitales generados por el *hardware* de recepción tendrán una separación aproximada de 3,9 μs. De esta manera, al seleccionar el valor de espera en 4 μs, se evita la posible recepción de un descanso, cuando en realidad hay un pulso en la señal de entrada.

La etapa de recepción se ha dividido en tres fases: Fase I, recepción del bit de inicio; fase II, recepción y decodificación de los pulsos de entrada; y fase III, recepción del bit de fin.

La fase I se ilustra en la figura 4.4, en esta se inicializan las variables, se recibe el bit de inicio y se revisa si ha ocurrido un error de bit de inicio o *Start_bit_error*(STerror). Luego, se inician los temporizadores T_2 y T_3, este último se utiliza para la detección de un error *no_información_error*(NIerror), el cual se encarga de garantizar que la pregunta del maestro es recibida dentro de una ventana de tiempo determinada. Finalmente, se revisa si hay un descanso o un pulso positivo; si se presenta este último, se asigna el valor de 1 al indicador `pa` para informar a la función de recepción, que el próximo pulso que debe recibir es un pulso negativo(ver tabla 4.3), para cumplir con la alternancia inherente a la modulación APM.

Es importante anotar, que el inicio y reinicio de los temporizadores no se puede realizar mediante la asignación directa del valor cero a su contador[7]. Por esta razón, fue necesaria la utilización de la propiedad de autoreinicio del contador del temporizador, que posee el canal 7 de cada uno de los temporizadores. Para un mejor entendimiento de esta técnica, consulte la función *Receivingf()* en el proyecto Esclavo_ASi.mcp.

En la fase II se demodula/decodifica la señal de voltaje en el bus, se almacena el dato recibido en el registro *Receive*, se calcula el bit de paridad que debe tener el mensaje recibido y se detectan los errores *Alternating_error* y *no_master_error*. Este último error no se encuentra descrito en la norma, pero fue necesaria su utilización para detectar cuando se esta recibiendo una pregunta del maestro o una respuesta de otro esclavo; con este fin, se utilizó el temporizador T_1 el cual informa la ausencia de pulsos en la entrada por más de 7,5 μs. Al ocurrir este error, se retorna a la fase I de la etapa de recepción para recibir la siguiente trama, que correspondera a la pregunta del maestro.

[7]El registro del contador de los temporizadores, en el MC9S12E128, no permite ser manipulado directamente mediante ordenes de asignación como se explica en [10]

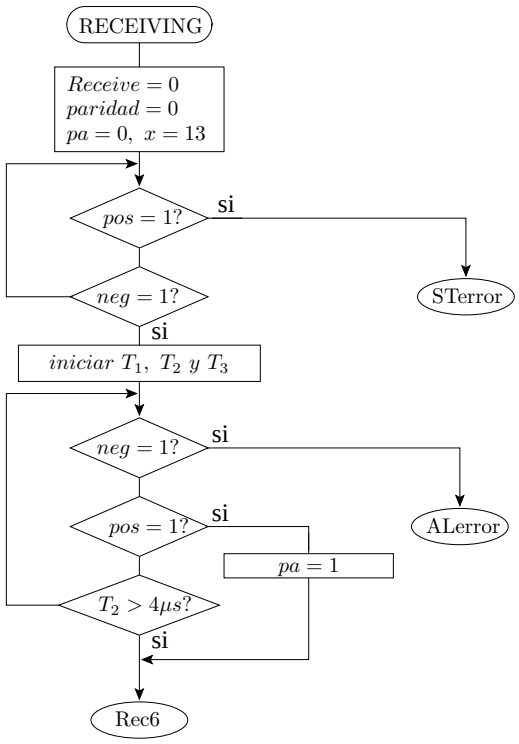


Figura 4.4: Recepción del bit de inicio.

El diagrama de flujo correspondiente a la segunda fase se muestra en la figura 4.5, en donde se observa que esta se subdivide en dos bloques: bloque A, en donde se reciben los pulsos de información,se almacena el dato recibido y se reinician los temporizadores T_1 y T_2; y el bloque B, en el que se revisa si ha ocurrido un error de alternancia.

Al recibir un pulso de información en el bloque A, el dato es almacenado en el registro *Receive*. Con este fin, se asigna el valor recibido al primer bit del registro, y luego se hace un desplazamiento lógico hacia la izquierda[8] para dejar espacio al siguiente bit a almacenar. Es decir, cada uno de estos desplazamientos se realiza con el fin de salvaguardar el bit recién almacenado y de habilitar el primer bit del registro para la asignación de otro valor. Al final, el registro *Receive* esta ordenado de tal manera, que el *bit* siguiente al de inicio es el bit mas significativo del registro y el menos significativo es el de paridad.

En el programa implementado, el dato recibido es almacenado temporalmente en el acumulador de la CPU y es transferido al registro *Receive* al final de la recepción, si no se ha recibido

[8]Este desplazamiento lógico es representado en el diagrama de flujo por el símbolo "<<".

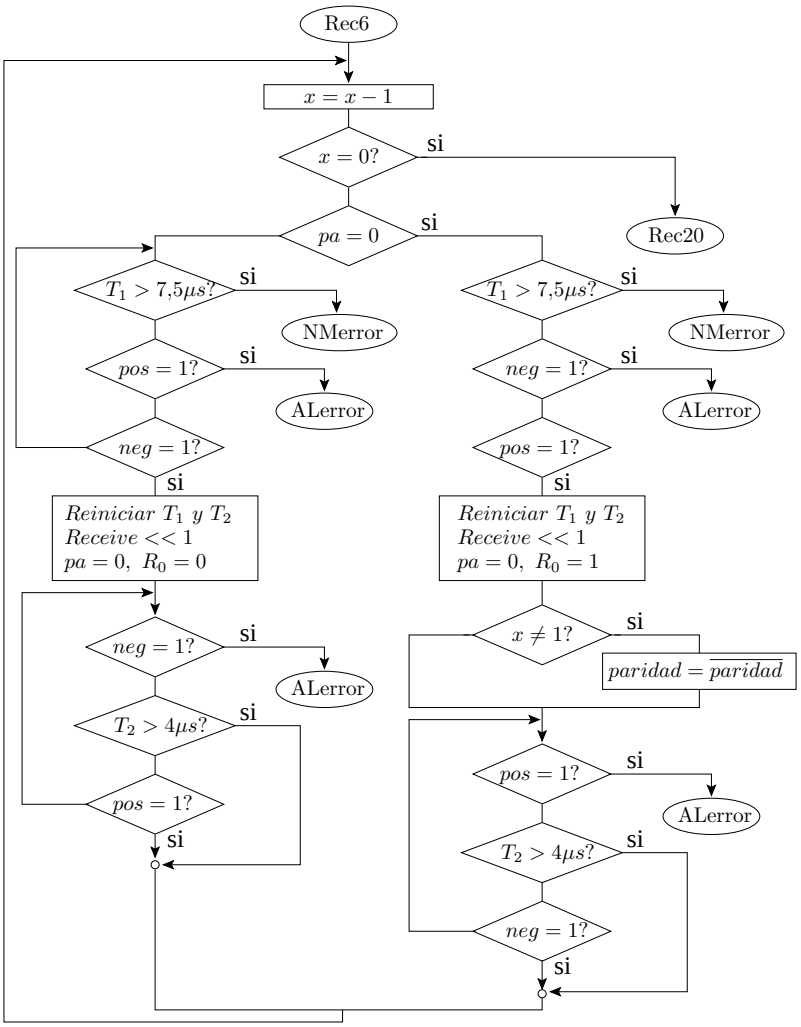


Figura 4.5: Fase de recepción y decodificación de los pulsos de entrada.

algún error. Esto permite, que el desplazamiento lógico hacia la izquierda y la asignación del valor recibido, consuman un total de 2 ciclos de reloj, en lugar de los 12 ciclos de reloj que consumiría si se hubiese utilizado directamente el registro *Receive*.

Igualmente, en el bloque A se creó la variable paridad cuyo valor inicial en esta etapa, es cero. Esta variable contiene el valor calculado por el esclavo, del *bit* de paridad que debe tener el mensaje recibido, para cumplir con la paridad par que debe tener toda trama *AS-i*. En consecuencia, la variable paridad permite la detección de un error de paridad o *parity error* mediante la comparación de esta variable con el *bit* de paridad almacenado en el registro *Receive*.

Para calcular el valor de la variable paridad, esta es invertida cada vez que se recibe un *bit* de información igual a uno: si se recibe un número par de unos, la variable paridad se invierte un número par de veces y conserva el valor de cero; mientras que, si se recibe un número impar de unos, la variable paridad se invertirá un número impar de veces y su valor final será uno. De esta manera se evita el uso de una variable que cuente el número de bits en uno, presentes en el dato recibido.

Después de recibir y decodificar el mensaje del maestro, solo resta la recepción del bit de fin, y la detección de los errores *end_bit_error* (EBerror), *no_information_error* (NIerror) y *length_error*(LNerror); con este objetivo se realizó la fase III que se muestra en la figura 4.6. El temporizador T_4 es iniciado luego de recibir el bit de fin, para evitar el envío de la respuesta del esclavo en un tiempo inferior a 18 μs (3 TB). Mientras que los temporizadores T_3 y T_2 son utilizados para la detección de los errores *no_information_error* y *length_error* respectivamente. En esta fase también se revisa si ha ocurrido un error *parity_error* al comparar el *bit* de paridad calculado con el bit de paridad almacenado en el registro *Receive*.

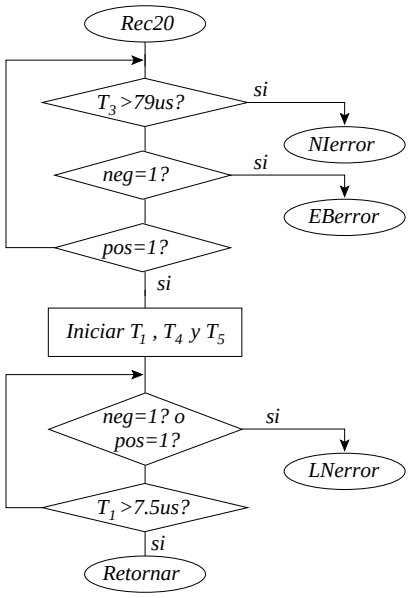


Figura 4.6: Recepción del bit de fin y detección de los errores: *Length_error*, No_information_error y *Parity_error*.

Finalmente, en la etapa de recepción, se asigna el valor del acumulador al registro *Receive* y se asigna el valor cero a la variable paridad.

4.5. *Decoding*

Esta etapa compara la dirección recibida en la etapa de recepción(`bits_direccion`), con la dirección almacenada en el registro *addressv*; si es diferente, activa la bandera `no_seleccionado`. Si por el contrario la dirección es igual y la transacción es conocida, el registro *Receive* es revisado con el fin de almacenar el valor correcto de la respuesta del esclavo en el registro *Transmit*, este valor almacenado no contiene aún, el valor del bit de paridad del mensaje a enviar.

Finalmente, si la información almacenada en el registro *Receive* es desconocida, se asignará el valor de uno a la bandera `tran_desc` como se muestra en el diagrama de flujo de la figura 4.7. Al final, el registro *Transmit* esta ordenado de tal manera, que el *bit* siguiente al de inicio es el bit mas significativo del registro. Es importante aclarar que cada una de las respuestas almacenadas en el registro *Transmit* y las acciones realizadas en cada una de las transacciones, están acordes con las especificaciones vistas en la sección 2.2.3.

Para simplificar el diagrama de flujo mostrado en la figura 4.7, se utilizan los nombres de las transacciones como condiciones de salto y sus patrones de bits se resumen en la tabla 4.4. En el programa implementado, las condiciones de salto están definidas en el archivo de cabecera decodificacion_digital.h.

Tabla 4.4: Patrón de bits de las transacciones *AS-i*

Transacción	**CB**	$A_4A_3A_2A_1A_0$	I_4	I_3	I_2	I_1	I_0
Data_exchange	0	$A_4A_3A_2A_1A_0$	0	I_3	I_2	I_1	I_0
Write_parameter	0	$A_4A_3A_2A_1A_0$	1	I_3	I_2	I_1	I_0
Reset_AS-i_slave	1	$A_4A_3A_2A_1A_0$	1	1	1	0	0
Read_IO_configuration	1	$A_4A_3A_2A_1A_0$	1	0	0	0	0
Read_identification_code	1	$A_4A_3A_2A_1A_0$	1	0	0	0	1
Read_Status	1	$A_4A_3A_2A_1A_0$	1	1	1	1	0
Read_reset_Status	1	$A_4A_3A_2A_1A_0$	1	1	1	1	1

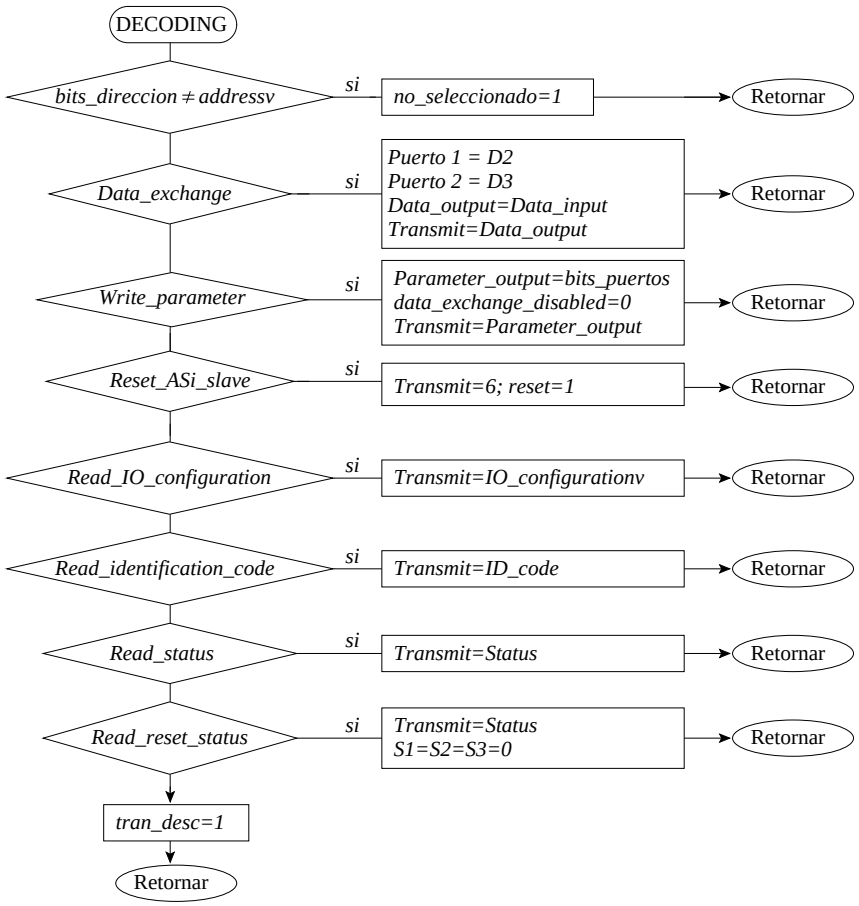


Figura 4.7: Diagrama de flujo de la etapa de decodificación.

4.6. *Transmit*

En esta etapa se envía al maestro de la red, la respuesta del esclavo almacenada en el registro *Transmit*, junto con el bit de paridad adecuado y el bit de fin. Con este objetivo, se utiliza el DAC del microcontrolador para generar la forma de onda de corriente necesaria para el correcto funcionamiento del hardware de transmisión, descrito en la seccion 3.3. La señal resultante tiene una amplitud de 2 V, y fue escogida por la limitación del *slew rate* del seguidor conectado al DAC en el sistema de desarrollo, el cual es de 2 $V/\mu s$ como se muestra en la figura **??**; y por la necesidad de maximizar la rata de envío de datos al DAC. En definitiva, los datos son enviados a una frecuencia de 5 MSPS, para formar la onda de corriente con un total de 30 muestras por tiempo de bit(TB). Cabe aclarar, que la frecuencia

de bus utilizada en el microcontrolador es de 25 MHz[9].

La etapa *Transmit* se ha dividido en dos fases: Fase I, calculo del bit de paridad; y fase II, generación de la forma de onda de corriente.

En la fase I mostrada en la figura 4.8, se calcula y asigna el bit de paridad al registro *Transmit*. Para hacerlo, se utiliza la misma metodología aplicada en la etapa *Receiving*; esto es, se asigna inicialmente el valor de cero al bit de paridad(PB) del registro *Transmit*, se revisa cada uno de los bits que lo conforman y cada vez que se encuentra un bit de información igual a uno, se invierte el bit de paridad del registro: si se encuentran un número par de unos, el bit de paridad se invierte un número par de veces y conserva el valor de cero; mientras que, si se encuentra un número impar de unos, el bit de paridad se invertirá un número impar de veces y su valor final será uno.

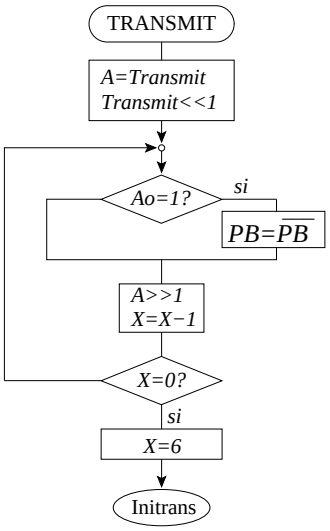


Figura 4.8: Cálculo del bit de paridad en la etapa *Transmit*

Para revisar todos los *bits* del registro *Transmit*, primero se asigna este al acumulador A del microcontrolador para disminuir los tiempos requeridos para su manipulación, luego se hace la revisión del *bit* menos significativo de este registro(A_0) y si es un uno se invierte el bit de paridad del registro *Transmit*, como se explicó en el párrafo anterior; y finalmente, para revisar el siguiente bit del acumulador A, se realiza un desplazamiento lógico hacia la derecha para que este ocupe ahora la posición A_0 y repetir nuevamente el ciclo.

[9]Es decir, se envía una muestra al DAC cada 5 ciclos de reloj.

En la fase II se genera la forma de onda de corriente mediante el envío de datos al DAC. Para realizar esta generación es necesario decidir, cuales son los datos que se deben enviar a este periférico por cada bit de información que es leído del registro *Transmit*. La estrategia utilizada para realizar estas decisiones se resume en la tabla 4.5, en donde se indica cual debe ser el conjunto de datos que se debe enviar al DAC(salto), teniendo en cuenta el bit recién enviado(presente) y el próximo bit a enviar(futuro). Las formas de onda que describen estos datos, se muestran en la figura 4.9.

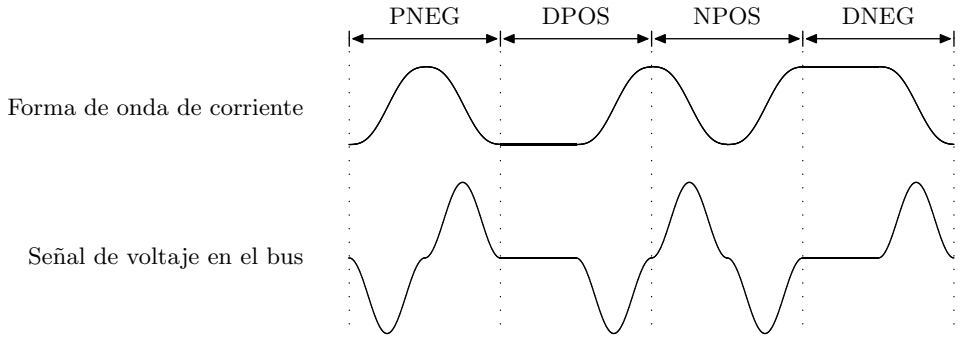


Figura 4.9: Formas de onda descritas por los datos enviados al DAC.

De otro lado, el diagrama de flujo correspondiente a la segunda fase se muestra en la figura 4.10, en donde se observa que inicialmente se revisa el valor del temporizador T_4, quien se encarga de garantizar que la respuesta del esclavo cumple con la pausa de maestro descrita en 2.2.4. Luego se envía el bit de inicio, para el cual no es necesario la revisión del registro *Transmit* ya que siempre posee el mismo valor.

Después de la revisión del temporizador y del envío del bit de inicio, comienza el bucle que se encarga de generar la forma de onda de corriente. Con este fin, se aplica el método utilizado anteriormente; esto es, se revisa el bit mas significativo del registro *Transmit* y luego

Tabla 4.5: Conjunto de datos a enviar al DAC.

presente	**futuro**	**salto**
0	0	NPOS
0	1	DNEG
1	0	DPOS
1	1	PNEG

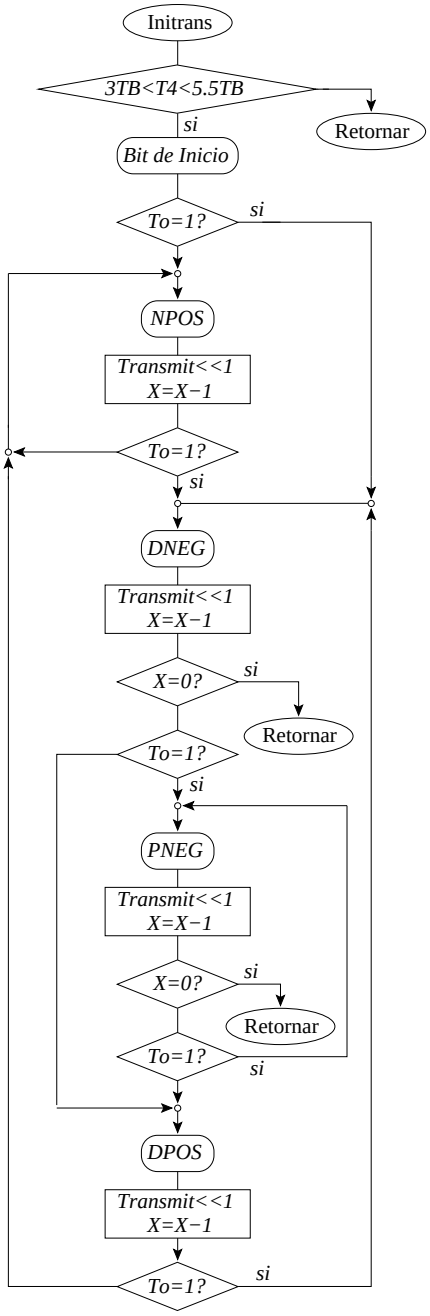


Figura 4.10: Generación de la forma de onda de corriente.

se realiza un desplazamiento lógico a la izquierda(<<) para revisar el siguiente bit. Al final del envío de cada conjunto de datos, se revisa el estado del siguiente bit de información y se toma la decisión sobre cual debe ser el siguiente salto a ejecutar. Adicionalmente, en esta fase se utiliza una variable para identificar cuando ha llegado el fin de los datos a enviar, esta

variable se indica en el diagrama de flujo como “x”.

4.7. *Sync*

En esta etapa se asigna el valor de uno a la bandera *sync* y se detecta la presencia de una pausa entre los pulsos recibidos, como se ilustra en la figura 4.11. Con el fin de detectar la pausa se utiliza el temporizador T_2, el cual se encarga de informar a la función *syncf()*, la ausencia de pulsos en la entrada por un tiempo superior a 4 μs. Se prefirió utilizar el temporizador T_2 en lugar de T_1, para disminuir el tiempo existente entre el fin de la transmisión y el comienzo de la recepción de datos. Luego de detectar la pausa, la máquina de estados ejecuta la etapa *Receiving*, a la cual se ingresa justo antes de recibir el bit de inicio de la siguiente trama en el bus, que corresponde a una pregunta de maestro.

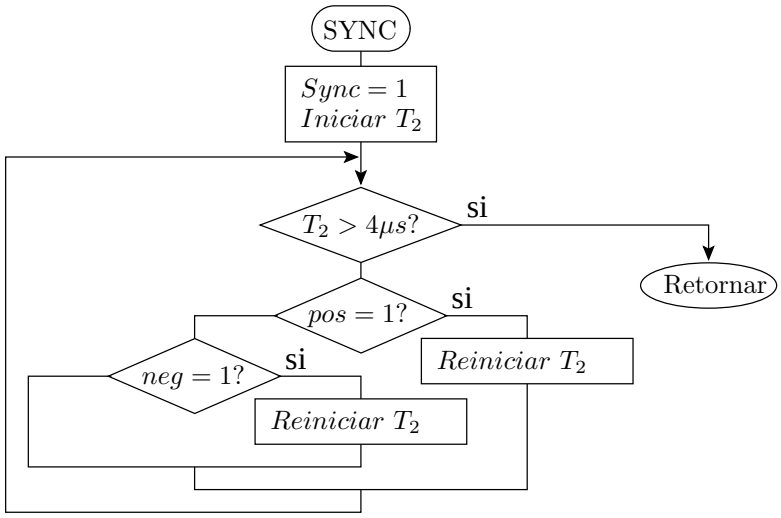


Figura 4.11: Diagrama de flujo de la etapa Sync.

4.8. *Wait*

A la etapa de *Wait* se ingresa cada vez que el maestro pregunte a otro esclavo, con el fin de esperar que este termine de responder para ingresar a la etapa *Sync* y mantener así la sincronía. Si el otro esclavo responde, el tiempo esperado esta dado por el temporizador T_6

cuyo valor de temporización es de 7 TB contados a partir de la recepción del pulso de inicio de la respuesta del otro esclavo; mientras que si el esclavo no responde, el tiempo esperado por la etapa esta dado por el temporizador T_5 cuyo valor de temporización es de 9 TB y es iniciado luego de recibir el bit de fin de la trama de maestro. El diagrama de flujo de esta etapa se muestra en la figura 4.12.

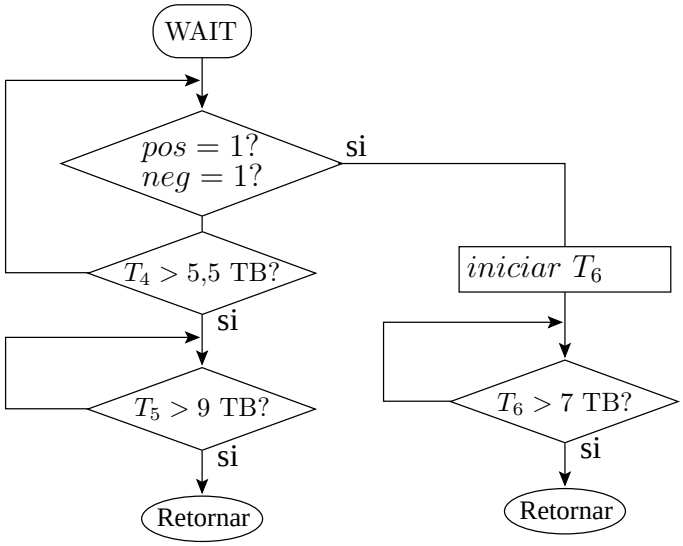


Figura 4.12: Diagrama de flujo de la etapa *wait*.

En conclusión, en este capítulo se presentaron las técnicas utilizadas para implementar la máquina de estados de un esclavo *AS-interface*, junto con los diagramas de flujo necesarios para su entendimiento. Estos diagramas de flujo fueron presentados de manera general, para que no sean dependientes del microcontrolador en el cual se realizó el software.

5 Resultados

Una vez se ha implementado el esclavo *AS-i* a nivel de Hardware y Software, es necesario validar el dispositivo, mediante pruebas de funcionamiento sobre una red comercial. En este capitulo se presentan los resultados obtenidos a través de las medidas y registros realizados con el osciloscopio a través de las diferentes etapas del procesador *AS-i*, haciendo una confrontación de dichos resultados con los requerimientos establecidos por la norma EN50295.

5.1. Señal de transmisión

Como ya fue presentado en capítulos anteriores, el patrón de transmisión es inicialmente generado como una forma de onda de tensión que posteriormente es convertida en corriente. La generación de esta señal es realizada por el DAC del microcontrolador, sin embargo su salida se ve limitada por el *Slew rate* del amplificador ubicado en la salida del módulo dentro de la tarjeta de desarrollo. La figura 5.1 muestra la medida del *Slew rate* del puerto del DAC en conjunto con el amplificador, utilizando una onda. Este valor de slew rate es 2 $V/\mu s$ y limita la amplitud de salida de la señal de transmisión, justificando de esta manera la amplitud de 2 V_{p-p} con la cual se genera la onda.

En la figura 5.2(a) se presenta un patrón analógico equivalente a un bit de información generado por el microcontrolador, con la forma de onda de la corriente que se va a transmitir. La tensión medida es de 1,96 V_{p-p}, con un tiempo de duración de 6 μs, que equivale al tiempo de bit, el cual no excede la desviación de 0,2 % determinada por la norma. La figura 5.2(b), presenta una trama completa de la forma de onda de corriente para una respuesta de esclavo, donde se resalta la ventana de tiempo de transmisión de 42 μs, teniendo en cuenta el descanso inicial requerido.

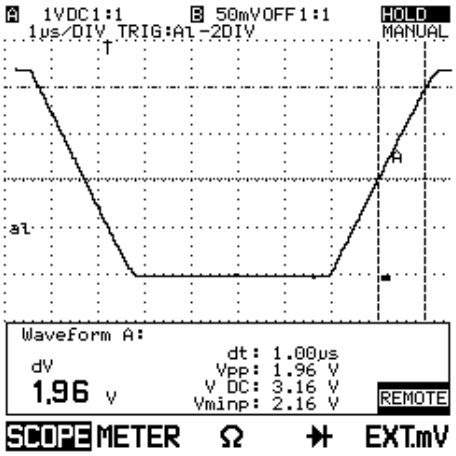
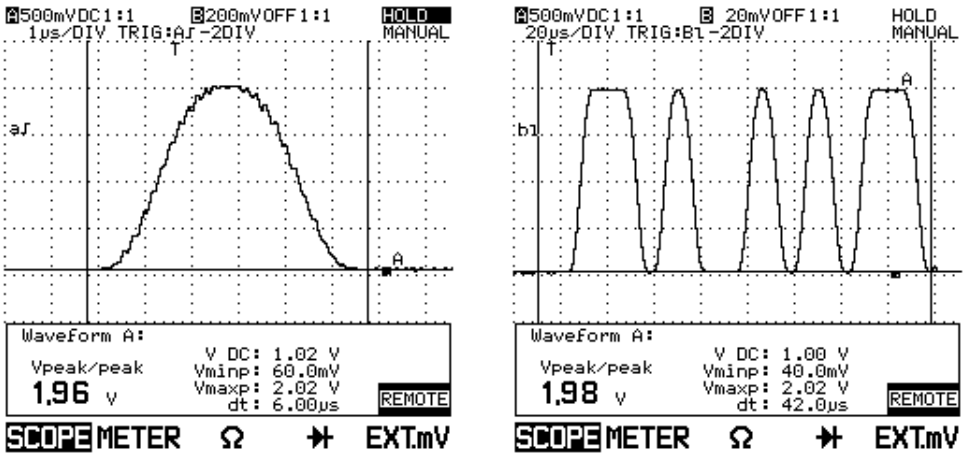


Figura 5.1: .



(a) (b)

Figura 5.2: .

Con este valor de tensión generado, la corriente transmitida de acuerdo a la ecuación [3.1] y [3.2], es de 60mA, valor que se ubica dentro del rango establecido por el protocolo [11]. Esta corriente es inyectada por la etapa de transmisión a línea *AS-i* generando la forma de onda que se ilustra en la figura 5.1, la cual toma un valor máximo de 4,64 V_{p-p} y mínimo de 4 V_{p-p}. La variación en el valor de los picos de tensión es ocasionada por el circuito de desacople de la fuente de alimentación de la red y las características físicas de la línea de transmisión [9]. Además la amplitud obtenida se encuentra dentro del nivel de 2 V_{p-p} a 8 V_{p-p}, aceptado como

pulsos válidos en el estándar IEC62026-2[1].

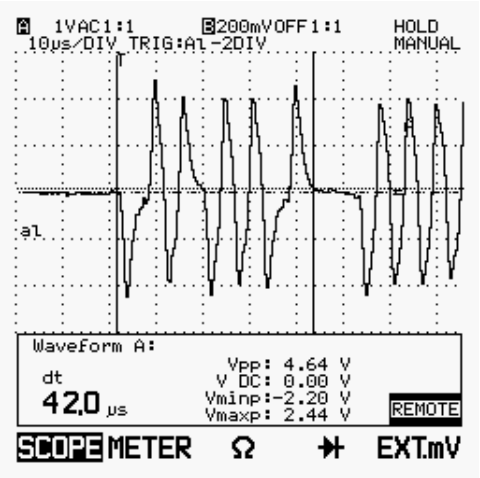


Figura 5.3: .

5.2. Señales de recepción

La norma IEC62026-2 establece como válidos aquellos pulsos analógicos que sean superiores a $1V_p$. Utilizando este valor de tensión para determinar las amplitudes válidas, la etapa de recepción fue ajustada para generar pulsos positivos o negativos, una vez sea superada este nivel de comparación [2]. La figura 5.4, muestra la señal mensaje en salida de la etapa de desacople (canal B) y los pulsos positivos que ingresan a los puertos del microcontrolador (Canal A), siendo evidente como el pulso es generado una vez la señal toma un nivel superior a 1 V. El ancho del pulso generado depende de la señal de entrada y en este caso toma un tiempo aproximado de 2 μs.

[1]se ha tomado esta norma como referencia cuando los valores no son especificados en la norma EN50295

[2]La referencia de la etapa de comparación debe ser ajustada a un valor de 2.2 V, ya que debemos tomar en cuenta la amplificación previa a la etapa de comparación como se describe en el capítulo 3

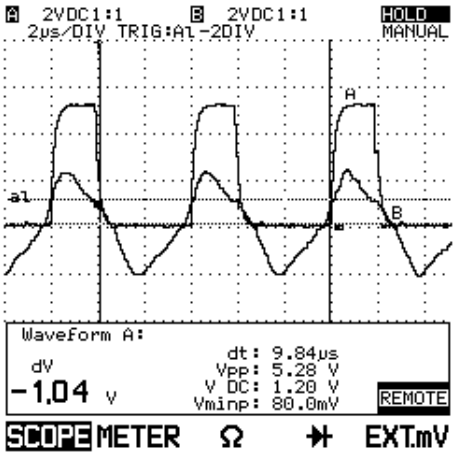
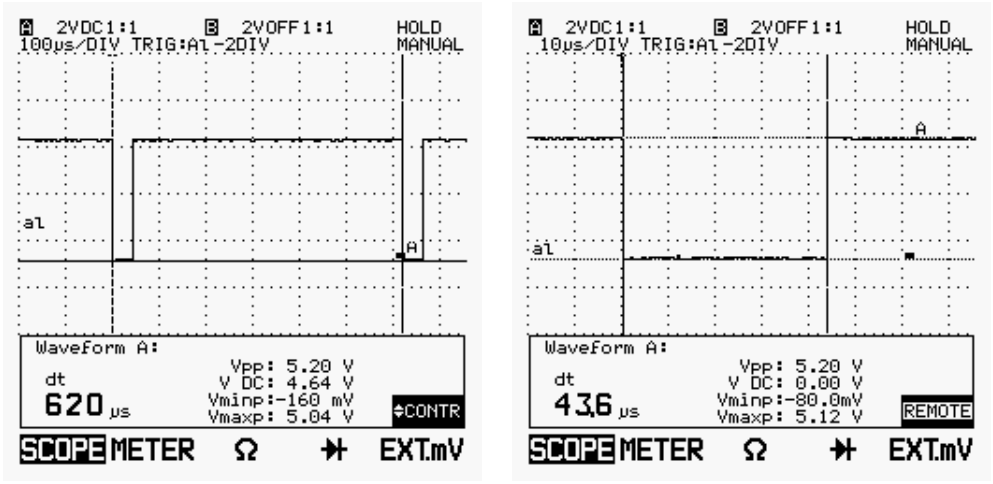


Figura 5.4: .

5.3. Señal de control del conmutador

La señal de control del conmutador refleja las ventanas de tiempo en las cuales se recibe o se transmite información. La figura 5.5(a) muestra la señal de control del conmutador que permite alternar entre transmisión y recepción en el esclavo *AS-i*. Con un valor lógico bajo se habilita la etapa de transmisión, y con un nivel alto se habilita la etapa de recepción. En la figura 5.5(a) se muestra como la etapa de recepción permanece habilitada la mayor parte del tiempo, lo cual es coherente con el estado por defecto que debe mantener el esclavo, a menos que sea necesario responder algún requerimiento del maestro. La figura 5.5(b) ilustra de forma más detallada la ventana de tiempo de transmisión, la cual se mantiene aproximadamente en 43,6 μs, tiempo suficiente para transmitir la señal de corriente que tiene una duración de 42 μs. El estado de recepción tiene un tiempo variable que depende de la regularidad con la cual sea interrogado el esclavo por parte del maestro, este tiempo depende del número de esclavos conectados a la red, y de las posibles retransmisiones que se puedan realizar por parte del maestro. Para este caso específico con tres esclavos conectados a la red, la etapa de recepción permanece activa un tiempo de aproximadamente 576 μs.



(a) (b)

Figura 5.5: .

5.4. Especificaciones de la fuente de alimentación

La fuente de alimentación diseñada genera dos referencias internas de voltaje, $+5\ V$ y $-5\ V$, que son utilizadas para alimentar los dispositivos internos del procesador, junto con la tierra del mismo. La tabla 5.1 muestra los valores medidos en esta etapa, en los cuales se incluyen las tensiones intermedias previas. Allí también se incluye el valor del rizado máximo que se puede presentar en condiciones de plena carga del dispositivo durante su operación.

Tabla 5.1: .

Tensión $[V]$	**Rizo**$_{máx}$ $[mV_{p-p}]$	**%Rizo**
30	20	0,06
11,4	20	0,18
5	20	0,4
−4,7	40	0,8

Los valores de rizado máximo medidos en la fuente de alimentación, son lo suficientemente pequeños comparados con los niveles de tensión generados, en ninguna de las cuatro tensiones el porcentaje de rizado supera el 1 %, de manera que no afectan el funcionamiento del dispositivo esclavo, garantizando una tensión regulada para todos sus elementos internos.

Otro factor importante es el consumo de corriente del esclavo. Para la implementación realizada el consumo de corriente del esclavo es de 100 mA, siendo un valor menor a los 150 mA establecidos como consumo máximo en la norma EN50295.

En síntesis el esclavo cumple con los rangos y parámetros establecidos en la norma EN50295 para un perfil de esclavo S-3.0, aquellos que no se definen en esta norma fueron ajustados en base a la norma IEC62026-2. Los procesos de desacople para la extracción de potencia, la recepción y transmisión de señal son realizados de manera exitosa por parte del esclavo implementado.

Finalmente es necesario destacar que todas las medidas realizadas fueron tomadas con la puesta en funcionamiento del esclavo $AS\text{-}i$ sobre una red comercial [3], validando su funcionamiento mediante el reconocimiento por parte del maestro como componente activo de la red. La dirección interna del esclavo fue cambiada en varias oportunidades para constatar su versatilidad y funcionamiento adecuado con diferentes direcciones.

[3] la red utilizada es la que se encuentra en la escuela de mecánica

6

Observaciones y Conclusiones

En este trabajo se realizó la implementación de un esclavo *AS-interface* en el microcontrolador MC9S12E128, cumpliendo con los requerimientos mínimos expuestos en la norma EN50295 [11]. La configuración de entradas/salidas del esclavo esta formada de dos entradas binarias y dos salidas binarias, conformando el perfil S-3.0.

Con el fin de permitir la conexión del esclavo en una red *AS-i* comercial, se implementó una interfaz física(hardware) que brinda al microcontrolador los acoples necesarios, para recibir y transmitir información a través del bus *AS-i*. Cabe aclarar que estas dos funciones no son realizadas de manera simultánea por el esclavo, sino que por el contrario, se alternan mediante el uso de un conmutador que es controlado por el microcontrolador.

Adicionalmente, con el propósito de tener un esclavo plenamente funcional mediante su conexión al bus, se implementó un circuito que permite la extracción de potencia del bus, para la alimentación de los circuitos internos del esclavo. Este circuito genera voltajes de referencia internos de 5 V, -5 V y tierra.

El microcontrolador utilizado trabaja con una frecuencia de bus de 25 MHz y permite mediante el uso de 3 módulos de temporizadores, el módulo del DAC, y el manejo de registros de 16 bits; establecer un proceso de comunicación con el maestro de la red. Este proceso de comunicación incluye las transacciones: *Data_exchange*, *Write_parameter*, *Reset_AS-i_slave*, *Read_IO_configuration*, *Read_identification_code*, *Read _Status* y *Read_reset_Status*.

De otro lado, el software desarrollado incluye la implementación de las 7 etapas de la máquina de estados del esclavo; la detección de los errores *Start_bit_error*, *Alternation_error*, *No_information_error*, *Parity_error*, *End_bit_error* y *Length_error*; y el manejo de los registros

propios del protocolo.

El correcto funcionamiento de las etapas *Sync*, *Async*, *Wait* y *Receiving*; depende en gran medida del uso adecuado de los temporizadores del microcontrolador. El uso de estos periféricos lleva consigo, la comparación del tiempo que demora su iniciación y reiniciación, y su tiempo de conteo; con el fin de revisar si son comparables[1]. El resultado obtenido de dicha comparación es de suma importancia, para tener en cuenta los tiempos de iniciación y reiniciación a la hora de asignar los tiempos de conteo, y evitar así, posibles diferencias entre el tiempo de conteo esperado y el tiempo obtenido.

Al llegar a este punto es importante anotar que en la etapa *Receiving*, los bits de información enviados por el maestro, son recibidos por el esclavo mediante la revisión directa de los pulsos de información[2], sin realizar la etapa intermedia de decodificación *Manchester* de la señal demodulada. Igualmente, en la etapa *Transmit* la señal modulada es formada mediante la revisión directa de los bits de información, sin realizar la etapa intermedia de codificación de estos bits. De este modo se disminuye el tiempo de respuesta del esclavo, al no perder el tiempo adicional que se hubiese requerido, si se utilizan rutinas de codificación y decodificación de la señal.

En el diseño de software embebido[3] es importante que el programador posea conocimientos del lenguaje ensamblador, ya que si es necesario, permite realizar códigos más compactos y más eficientes; debido a la manipulación al máximo de los registros de la CPU y una mejor utilización del conjunto de instrucciones del microcontrolador. Además, este lenguaje facilita la depuración de los programas, ya que permite comprender el resultado del proceso de compilación, realizado por el software de programación. Por estas razones, las etapas *Receiving* y *Transmit* han sido implementadas en lenguaje ensamblador. En la primera se permite la recepción y almacenamiento de los datos de manera simultánea a la recepción de los pulsos; mientras que en la segunda, se permite la generación de la señal modulada de manera

[1]Esta cualidad de comparable es totalmente dependiente de la aplicación y de la precisión en el tiempo de conteo que esta requiera.

[2]En la sección 4.4 se define el termino "pulsos de información", y se describe de manera detallada el proceso de recepción.

[3]Se entiende aquí por software embebido, todo software que ha sido implementado sobre un microcontrolador o DSP.

simultánea a la revisión de los bits de información.

En resumen, el dispositivo implementado permite la utilización de cualquier perfil de esclavo, mediante la manipulación del software realizado. Además, el diseño modular tanto del hardware como del software, hacen posible que estos puedan ser utilizados en la implementación de otros dispositivos *AS-i*, como lo sería un *sniffer* o un maestro de red.

Finalmente, el dispositivo esclavo fue validado mediante su conexión a una red *AS-i* comercial y mediante su reconocimiento por parte del maestro de la red. Las pruebas fueron realizadas con diversas direcciones del esclavo, y se verificó: la amplitud de la señal de corriente generada por el hardware de recepción; la señal de voltaje que resulta de inyectar esta señal de corriente en el bus *AS-i*; el ancho del pulso de salida del hardware de recepción; el ancho de la ventana de tiempo del bit de control del conmutador, que permite la transmisión; y la verificación de las especificaciones de la fuente de alimentación.

6.1. Recomendaciones

- Integración de todas las etapas en una tarjeta única, con el fin de brindar una solución más compacta desde el punto de vista de hardware.
- El desarrollo de Hardware y software debe ser un proceso de realimentación conjunta, que permita la depuración adecuada de la implementación final que se desee realizar. Con esto se garantiza avanzar en una misma dirección, teniendo en cuenta los factores mutuos de incidencia.
- Habilitar mediante el software, la transacción de cambio de dirección del esclavo.
- Implementar funciones auxiliares del protocolo, como el *watchdog*, que serían manipulados mediante el registro de parámetros del esclavo.

Bibliografía

[1] Analog Devices. *AD8031/AD8032 Datasheet,Rail-to-Rail I/O Amplifiers*, 1999.

[2] Analog Devices. *AD8612 Datasheet, Single Supply Comparators*, 2000.

[3] Analog Devices. *AD8012 Datasheet, Low Power Amplifier*, 2003.

[4] Analog Devices. *ADG787 Datasheet, Dual 2:1 Mux/Demux USB 1.1 Switch*, 2006.

[5] Francisco J. Ceballos. *C/C++ curso de programación.* Alfaomega, 2nd edition, 2002.

[6] Coilcraft. *WB2010-PC Datasheet, Wideband RF Transformers*, 2004.

[7] Omar L.Ñuñez Gualdrón, Jaime G. Barrero Pérez, and Jorge Meneses. Módulo de integracion de sensores y actuadores basado en el protocolo as-i de comunicaciones industriales, 2006.

[8] InterlinkBT-Bus stop. *AS-Interface Tutorial*, 2002.

[9] As international Association. *Actuator Sensor Interface-Complete Specification Version 3.0.* International electrotechnical commission(IEC), 2004.

[10] Motorola. *HC12 CPU12 Reference Manual.*

[11] Comite técnico CENELEC TC 17B. *Low-voltage switchgear and controlgear, controller-device interface systems. Actuator sensor interface (AS-i).* CENELEC, 1998.

[12] Grupo Automatización y Accionamientos(A&D). *Todo Sobre AS-Interface.* Siemens, 2000.