UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA
TRABAJO ESPECIAL DE GRADO

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA EL CONTROL DE INVENTARIO IMPLEMENTADO CON UN DISPOSITIVO DE HARDWARE REPROGRAMABLE.

Tutor:
Wilmer Sanz
Autores:
Pineda S., Joedithm J.
C.I.: 19.655.137
Torrealba De C., María V.
C.I.: 19.197.722

Naguanagua, Diciembre de 2012.

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA
TRABAJO ESPECIAL DE GRADO

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA EL CONTROL DE INVENTARIO IMPLEMENTADO CON UN DISPOSITIVO DE HARDWARE REPROGRAMABLE.

(Trabajo Especial de Grado presentado ante la Ilustre Universidad de Carabobo para optar al título de Ingeniero Electricista).

Tutor:
Wilmer Sanz
Autores:
Pineda S., Joedithm J.
C.I.: 19.655.137
Torrealba De C., María V.
C.I.: 19.197.722

Naguanagua, Diciembre de 2012.

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA EL CONTROL DE INVENTARIO IMPLEMENTADO CON UN DISPOSITIVO DE HARDWARE REPROGRAMABLE.

RESUMEN

El inventario de mercancía constituye un instrumento fundamental que permite conocer y controlar, cualitativa y cuantitativamente, los bienes de una empresa destinados a la venta; por lo que de su buena gestión depende el éxito de una empresa. El inventario de productos que se encuentran en el almacén de una empresa normalmente se realiza en forma manual, por lo que pueden existir errores en cuanto a la cuantificación del mismo, además de que el tiempo invertido en la realización de éste puede ocasionar atraso en las operaciones normales de la empresa. En consecuencia se hace necesario agilizar y mejorar la realización de inventarios de una empresa mediante un sistema de control implementado con un dispositivo de hardware reconfigurable (FPGA) donde la identificación de cada producto se realice mediante la simulación de la tecnología RFID. Se busca construir un prototipo que permita la implementación del sistema deseado en el cual, además, exista una interfaz gráfica que permita al usuario tener acceso a la información contenida en el registro del inventario. La metodología a utilizar se basa en la concepción del funcionamiento a nivel general del sistema, definiendo un módulo de información (captura de datos), un módulo de control (procesamiento de datos) y un módulo administrador (registro de datos-interfaz gráfica), para luego definir con más detalle cada una de sus partes. Se espera que el prototipo sea capaz de establecer comunicación con el módulo administrador para que en éste se genere la base de datos del registro de inventario, y que pruebe la factibilidad de la implementación del sistema de control.

Palabras Clave: hardware reconfigurable, RFID, VHDL, protocolo de comunicación, interfaz gráfica.

DEDICATORIA

En primer lugar a Dios, por haberme permitido llegar hasta este punto y haberme dado salud y sabiduría para lograr mis objetivos, además de su infinita bondad y amor.

A mis padres, Alberto Torrealba y Reina De Caires de Torrealba, por ser ellos el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo, por sus consejos, sus valores, pero más que nada, por su amor.

Al Prof. Wilmer Sanzs, mi tutor, por su gran apoyo y motivación para la culminación de mis estudios profesionales y para la elaboración de este trabajo. Todo este trabajo ha sido posible gracias a él.

A las diferentes personas, compañeros y amigos que de una u otra manera aportaron su ayuda para que se hiciera posible este Trabajo de Grado.

Maria Verónika Torrealba D.

DEDICATORIA

A Dios, por haberme dado el privilegio de la vida y por estar siempre a mi lado en el andar de mi vida.

A mis padres, Guillermo Pineda y Judith Sánchez de Pineda , por su apoyo incondicional, por creer en mi, por sus sabios consejos, por el valor mostrado para salir adelante y por su amor.

Al Prof. Wilmer Sanzs, mi tutor, por su apoyo ofrecido en este trabajo y por impulsar el desarrollo de mi formación profesional. Todo este trabajo ha sido posible gracias a él.

A mis compañeros y amigos por su ayuda y apoyo para la realización de este Trabajo de Grado

Joedithm J. Pineda S.

AGRADECIMIENTO

En primer lugar damos infinitamente gracias a Dios, por habernos dado fuerza, capacidad, sabiduría y valor para culminar esta etapa de nuestras vidas.

A nuestros padres, por su apoyo incondicional, quienes siempre se esforzaron para ayudarnos a poder cumplir con nuestra meta académica. Estaremos siempre agradecidos.

Al Prof. Wilmer Sanz, nuestro tutor, por estar siempre dispuesto a darnos sus sabios consejos y por toda la colaboración brindada para la elaboración de este Trabajo de Grado; además de brindarnos calma y paciencia en los momentos mas difíciles.

A todos los compañeros, amigos y profesores que ayudaron directa e indirectamente en la realización de este proyecto.

Y por último un agradecimiento mutuo entre los autores de este trabajo de grado, por brindarnos mutuamente paciencia, apoyo y amistad, además del apoyo incondicional en el transcurso de la carrera universitaria.

Maria V. Torrealba y Joedithm J. Pineda.

ÍNDICE GENERAL.

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

INTRODUCCIÓN.

Como es de saber, la base de toda empresa comercial es la compra y venta de bienes y servicios, de aquí viene la importancia del manejo de inventario por parte de la misma. El hecho de que una empresa tenga un buen control y gestión de su inventario, permite tener controlado algo tan importante como todo el flujo comercial de la empresa, lo cual contribuye en gran medida al logro del éxito de la misma.

El objetivo de un inventario es tener controlado en todo momento los medios que la empresa dispone, por ello el proceso de actualización del mismo tiene que ser rápido, fácil y preciso, ya que tener un inventario desactualizado es lo mismo que no tenerlo, puesto que no será de utilidad. En vista de todo esto, el propósito de esta investigación fue el de diseñar un prototipo para el sistema de control de inventario, basado en la tecnología RFID (tecnología simulada mediante software),que permitiera gestionar el inventario de una empresa mediante una herramienta computacional que proporcione una interfaz gráfica, además de tenerlo actualizado en tiempo real.

Para hacer un poco mejor el entendimiento del presente Trabajo de Grado, éste se dividió en varios capítulos, cuyo contenido se explica brevemente a continuación:

Capítulo I: En este capítulo se presenta la problemática existente que conllevó a realizar este trabajo de investigación, para darle solución se plantea un objetivo general y siete objetivos específicos, así como también la justificación y alcance que tiene esta investigación.

Capítulo II: Aquí se presentan los antecedentes de este Trabajo de Grado como una serie de investigaciones que sirvieron como apoyo y guía para el desarrollo y ejecución del proyecto. Además se presentan las bases teóricas que fundamentaron esta investigación,

haciéndose mayor énfasis en la tecnología implementada (dispositivo de hardware reprogramable), así como también en su lenguaje de programación.

Capítulo III: Aquí se refleja el tipo, diseño y etapas en las que se basó la investigación. En las etapas se describen cada una de las fases ejecutadas en base a los objetivos específicos planteados en este Trabajo de Grado. Además se hace una breve descripción de los módulos que conforman al sistema diseñado.

Capítulo IV: En este capítulo se presenta el marco operacional para el desarrollo del sistema diseñado en relación con las etapas de la investigación planteadas en el capítulo anterior; presentando diagramas de flujo de los procesos del sistema, simulaciones necesarias para el entendimiento del funcionamiento del sistema diseñado.

Capítulo V: En este capítulo se presenta el manejo completo del Sistema de Control de Inventario diseñado en este Trabajo de Grado, especificando los requerimientos necesarios de hardware y software, así como también el funcionamiento y características de cada una de las ventanas que posee del software desarrollado, con la finalidad de presentar una especie de manual de usuario. También se presenta un manual de instalación para el software diseñado para el usuario que sirva de ayuda a la hora de instalar el software un computador. Por último se muestran los resultados obtenidos de la prueba realizada al sistema completo, en cuanto a recepción de datos del inventario.

CAPÍTULO I.

EL PROBLEMA.

1.1 PLANTEAMIENTO DEL PROBLEMA.

El objetivo principal en toda empresa es el de mejorar sus ingresos y aumentar sus ganancias y para ello es necesario que mantengan un buen control de sus inventarios, ya que de su buena gestión depende el éxito de dichas empresas; además de que los inventarios representan frecuentemente una considerable inversión, por lo que las decisiones con respecto a las cantidades de inventarios son importantes. Los modelos de inventario y la descripción matemática de los sistemas de inventario constituyen una base para estas decisiones.

El inventario de mercancía es la parte del activo que posee cualquier empresa destinado para la trasformación y venta posterior o directamente para la venta, es por ello que constituyen un elemento fundamental para la determinación del costo de venta y, por ende, el resultado económico de una empresa.

Una de las principales consideraciones al evaluar un proceso de inventario es decidir con qué frecuencia debe realizarse y hay consideraciones importantes en la decisión de este periodo de tiempo, existiendo varias opciones posibles, es decir que se puede realizar una vez a la semana, una vez al mes o cada trimestre, dependiendo de los requerimientos de cada empresa.

Como los inventarios se realizan manualmente, pueden existir errores en el conteo de unidades de productos o materia prima. El problema, en lo que se refiere a la cuantificación de los inventarios, puede ser tan grande o tan pequeño como la gama de productos que exista en el establecimiento. Los inventarios constituyen uno de los

componentes del activo más susceptible a manipulaciones, lo que puede traer como consecuencia pérdidas no justificadas en los mismos o atrasos en las operaciones normales de la empresa. Otros de los inconvenientes que se presentan en la realización de los inventarios es el relacionado con el tiempo invertido en su realización, lo cual puede generar costos ocasionados por el deterioro en el almacenaje.

Uno de los métodos más rudimentarios que han implementado las empresas y comercios para agilizar y controlar la realización de inventarios ha sido la hoja de cálculo, la cual constituye una herramienta sencilla. Actualmente también se implementa un sistema basado en la lectura de códigos de barra de los productos (con un dispositivo de mano), el cual almacena cada lectura en un servidor que genera un archivo que incluye el código del producto y la cantidad contada; pero estos métodos acarrean problemas en cuanto a la inversión de tiempo e inversión en costos de personal encargado para dicha actividad, además de todas las posibles desventajas, ya mencionadas, que pueda presentar el control de inventarios en forma manual.

En consecuencia se hace necesario mejorar, mediante la implementación de un sistema de control utilizando RFID (por sus siglas en inglés), el control de inventarios en una empresa; este control se basa en la captura automática de datos identificando objetos mediante el uso de ondas de radio frecuencia, detectando la presencia de un producto mediante la lectura de una etiqueta codificada (constituida por una antena y un chip) adherida a cada producto; la lectura de esta etiqueta debe ser almacenada en una base de datos que tendrá la identificación del producto, la cantidad, y los horarios de ingreso y egreso de los mismos, todo esto mediante la programación en un dispositivo de hardware reprogramable.

1.2 JUSTIFICACIÓN DE LA INVESTIGACIÓN.

Debido a que los inventarios requieren una atención especial, ya que de su buena y sana administración depende en gran medida el éxito de cualquier empresa, la implementación de este sistema permitirá la obtención de un registro detallado de cada uno de los productos que ingresen al área de cobertura del lector, mejorando el control de inventarios porque, como ya se dijo, el inventario lo hará el propio sistema, por lo que no será necesaria la intervención de un personal que invierta tiempo en la realización de la inspección de cada producto, aumentando la eficiencia en las operaciones de una empresa.

Además, como el sistema se encontrará detectando cada producto en tiempo real, al dejar de detectarlo el sistema registrará la hora de egreso del mismo, lo cual impide que exista una diferencia entre el contenido de los registros de inventario y la cantidad real de productos o mercancía dentro de un establecimiento, negocio o empresa.

La implementación de este sistema no afectará el volumen ni la estética de los productos a detectar ya que las etiquetas codificadas, a pesar de que son dispositivos complejos constituidos por una antena, un chip y otros elementos, pueden llegar a tener tamaños pequeños comparados con el de una etiqueta de cuaderno, debido a que se trabaja en la banda espectral de radiofrecuencia.

Este Trabajo Especial de Grado constituye un aporte a la investigación en la línea de investigación de Sistemas Digitales ya que se basa en un conjunto de dispositivos destinados a la generación, procesamiento, transmisión y almacenamiento de señales digitales constituidas por la información de registro de cada producto del inventario.

1.3 OBJETIVOS.

1.3.1 OBJETIVO GENERAL.

Construir un prototipo para un sistema de control de inventarios de la mercancía existente en una empresa, mediante la identificación de los productos que constituyen a la misma e implementado con un dispositivo de hardware reconfigurable.

1.3.2 OBJETIVOS ESPECÍFICOS.

- Estudiar la documentación referente a los dispositivos de hardware reconfigurable, específicamente los FPGAs (de sus siglas en inglés, Field Programmable Gate Array), para usarlos como dispositivos de control.
- Revisar la documentación relacionada con los lectores de las etiquetas codificadas disponibles en el mercado, necesaria para el conocimiento del protocolo de comunicación adecuado que utilizan los componentes electrónicos que conformarán el sistema RFID.
- Definir los protocolos que permitan la comunicación entre el computador que simulará la salida del sistema RFID y el dispositivo de control, y entre el dispositivo de control y el servidor, sobre la base del estándar RS232.
- Seleccionar la tarjeta de desarrollo (FPGA) y el correspondiente software que permita la implementación de una de las posibles soluciones para el sistema a desarrollar.
- Programar la FPGA mediante el lenguaje descriptor de hardware VHDL (de sus siglas en inglés VHSIC Hardware Description Language).
- Diseñar una interfaz gráfica para que el usuario tenga acceso a la información contenida en el registro del inventario.
- Ensamblar un prototipo para la demostración del correcto funcionamiento del sistema diseñado.

1.4 ALCANCE.

Este proyecto se enfocará en el desarrollo de un prototipo electrónico para el control y realización del inventario de una empresa basado en la simulación de la tecnología RFID, es decir, la salida del lector del sistema RFID será simulada con información enviada desde un computador; que sea capaz de establecer comunicación con otro computador para que en éste se genere la base de datos del registro de inventarios y que pruebe la factibilidad de la implementación del sistema de control.

CAPÍTULO II.

MARCO TEÓRICO.

2.1 ANTECEDENTES DEL PROBLEMA.

Como cualquier trabajo de investigación es necesario contar con una serie de estudios realizados por otros autores sobre el tema en cuestión, con el fin de tener una visión general de cómo abordar el problema. Algunos estudios relacionados acerca del tema son los siguientes:

- *"Diseño y Construcción de un Prototipo para el Control de Acceso a las Aulas y Laboratorios de la Escuela de Ingeniería Eléctrica de la Universidad de Carabobo utilizando Tecnología RFID"*. Elaborado por William Aponte y Rogers López. Universidad de Carabobo - Edo. Carabobo - Venezuela, Mayo del 2012.

El presente Trabajo de Grado se baso en la implementación de un control de acceso a los espacios de la Universidad de Carabobo, específicamente en la Facultad de Ingeniería, utilizando identificación por radiofrecuencia (RFID), el cual cuenta con una base de datos programada en una computadora que contiene una lista con las personas que pueden ingresar a cada aula o laboratorio además de un registro de las permanencia de las personas en cada lugar donde hay un control de acceso [1].

Este Trabajo de Grado de grado es de utilidad, con el respecto al software de control diseñado, para el desarrollo en cuanto a factores y criterios de similitud; lo cual permitirá precisar aspectos para la creación y operación del software del control de inventario.

- *"Implementación De La Etapa De Transmisión De Un Sistema De Comunicación Digital Utilizando La Tecnología FPGA"*. Elaborado por Johanna Lizeth Fernández Yépez. Escuela Politécnica Del Ejército Sangolquí - Ecuador, 2010.

En el desarrollo de este proyecto de grado se implementa la etapa de transmisión de un sistema de comunicación digital utilizando tecnología FPGA y se analiza el desempeño de los bloques de codificación de canal y modulación digital en hardware. Inicialmente se indaga en la arquitectura, las formas de implementación y las herramientas para la programación de FPGAs, así como también la configuración y características de cada etapa de un sistema de transmisión digital [2].

El logro específico obtenido en este trabajo, respecto a la implementación de comunicación mediante FPGA, sirvió como base para explorar las posibilidades del sistema aquí usado, adaptar aspectos donde se verifique similitud y formular soluciones al sistema planteado en este Trabajo de Grado.

- *"Sistema de Soporte para Control de Inventarios mediante RFID".* Elaborado por Carlos Alfredo Collao Vilches. Santiago de Chile, Enero 2008.

Este proyecto se basó en el desarrollo de un sistema de control de circulación de volúmenes dentro de la biblioteca de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. Al implementar el modelo de manejo del sistema de circulación de volúmenes, se genera una plataforma de control para el portal de circulación de volúmenes de biblioteca, pero que a su vez puede ser extensible a otras aplicaciones bibliotecarias que requieran comunicación usando módulos RFID. El modelo el almacenamiento diseñado en este trabajo permite capturar los datos en caso que circulen muchos volúmenes en un pequeño periodo de tiempo [3].

Este trabajo, respecto al sistema de soporte para el control de inventario, permitió concretar un modelo para la base de datos, su revisión y almacenamiento, contribuyendo a la programación del software para el manejo del flujo de datos que contiene la información de volúmenes de productos en el inventario.

- *"Estudio, Diseño Y Simulación De Un Sistema De RFID Basado En Epc".* Elaborado por José María Ciudad Herrera y Eduard Samá Casanovas, 2005.

El objeto de realización de este proyecto es el conocimiento de la tecnología RFID (Radio Frequency IDentification), así como el diseño de un prototipo de sistema de identificación y la simulación del canal inalámbrico en el que se realiza la comunicación entre los elementos del sistema. Este proyecto se realizó bajo el estándar EPC (Electronic Product Code) el cual es un código global divide las etiquetas de un sistema RFID en seis tipos diferentes, dependiendo de su funcionalidad [4].

Este trabajo sirvió para aprovechar sus resultados sobre el uso de la tecnología RFID en comunicación inalámbrica y diseñar el módulo o subsistema a usar en este proyecto.

2.2 MARCO CONCEPTUAL.

2.2.1 SISTEMAS RFID.

Un sistema de RFID (Identificación por Radio Frecuencia, por sus siglas en inglés) es un sistema que sirve para la comunicaciones sin cables y sin contacto físico, entre dos o más objetos, donde uno emite señales de radio (ondas de radiofrecuencia) y el otro responde en función de la señal recibida; es decir, es una tecnología inalámbrica que permite, básicamente, la comunicación entre un lector y una etiqueta capturando la información contenida en dicha etiqueta electrónica (tag). Estos sistemas permiten que el lector envíe una señal para que las etiquetas que estén en su área de cobertura le transmitan la información almacenada en su memoria (código de identificación), dicha información puede ir desde un Bit hasta KBytes, dependiendo principalmente del sistema de almacenamiento que posea el tag. La tecnología RFID tiene la ventaja de tener alta capacidad de información y que se pueden detectar simultáneamente varios tags.

2.2.2 FUNCIONAMIENTO BÁSICO DE UN SISTEMA RFID.

El funcionamiento básico de este sistema consiste en un lector que conectado a una antena envía ondas de radiofrecuencia las cuales son captadas por las microantenas que

contienen los tags que se encuentran en el área de cobertura de la antena del lector; la onda de radiofrecuencia captada por el tag activa su microchip, el cual mediante la microantena envía una señal de radiofrecuencia que contiene la información en su memoria. Esta información, que es recibida por el lector, se envía a una base de datos en la que previamente se han registrado las características del producto. [4]

2.2.3 COMPONENTES DE UN SISTEMA RFID.

Un sistema RFID se compone de los elementos que se muestran en la figura 2.1 [4], como se puede ver estos elementos son: los tags (etiquetas codificadas), el lector, la antena, y el sistema encargado de manejar la información (gestión de la data). Cada uno de los elementos del sistema lleva a cabo una función en específico de forma secuencial, permitiendo que se realice la captura de información que contienen las etiquetas o tags.

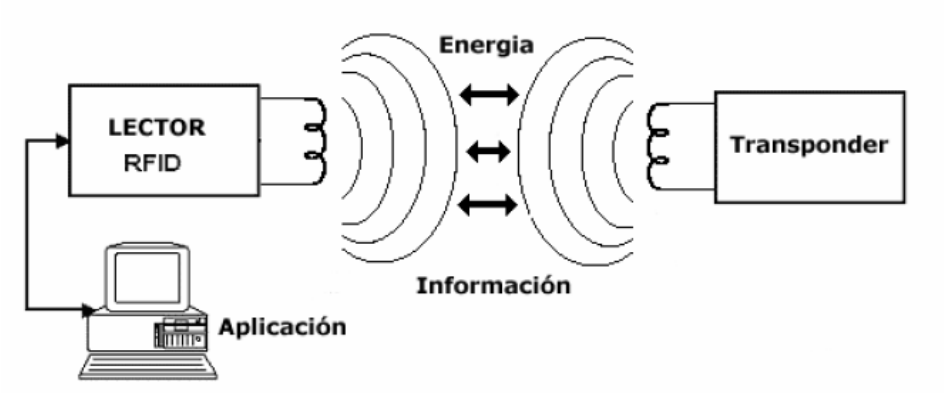


Figura 2. 1. Componentes principales de un sistema RFID.

2.2.3.1 TAG O ETIQUETA RFID (TRANSPONDER).

Su nombre Transponder se debe a su modo de operación TRANSmitter/resPONDER ya que puede transmitir (en modo de respuesta a un dispositivo lector) y recibir señales. Es un componente por lo general pasivo (sin batería), compuesto por la integración, en un substrato, de un circuito integrado (chip) y una antena de radiofrecuencia; en la figura 2.2 [5] se aprecia la estructura general de una etiqueta RFID. [4]

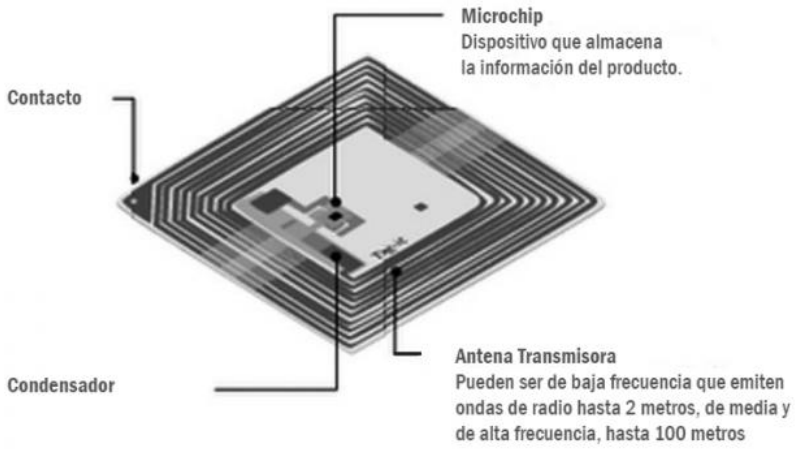


Figura 2. 2. Etiqueta RFID.

2.2.3.2 LECTOR O TRANSCEPTOR.

Es el elemento interrogador de los sistemas RFID, se compone de un circuito que emite energía electromagnética (específicamente ondas de radio) a través de una antena, y una electrónica que recibe y decodifica la información enviada por el transponder o tag y la envía al sistema de captura de datos. La información obtenida de las etiquetas es transmitida por el lector a través de interfaces de comunicación (que pueden ser con cables o inalámbricas) a la base de datos existente. [4]

2.2.3.3 BASE DE DATOS.

La base de datos es un componente de tipo software que permite almacenar la información de identificación proveniente de los tags, recibida por el lector; esta plataforma software es indispensable para gestionar una aplicación que requiera el manejo de dicha información por parte de un usuario. Es necesario almacenar la información de identificación en un formato común de manera que cualquier usuario, de nivel superior, pueda trabajar y acceder a ella. Por todo esto debe existir una interfaz entre la base de datos y el dispositivo lector, que se encargue del tratamiento previo de los datos generados por el lector.

2.2.4 MIDDLEWARE.

El Middelware es un software o es un conjunto de componentes desarrollados que sirven para integrar aplicaciones, es decir, en un ambiente donde interactúan distintas tecnologías éste se encarga de comunicar e integrar los datos de diversa índole, haciéndolo de forma conectada o desconectada, facilitando la integración de aplicaciones y plataformas. En el sistema a desarrollar el middelware es el encargado de procesar los datos extraídos de una etapa para luego transmitirlos a la etapa de administración de éstos. Es posible desarrollar un sistema Middleware implementando la tecnología FPGA.

2.2.5 TECNOLOGIA FPGA.

Los dispositivos FPGA (siglas en inglés de Field Programmable Gate Array, traducido como Matriz de Puertas Lógicas Programables) son circuitos integrados que pueden configurarse de manera que ejecuten cualquier función lógica y hacer posible la ejecución de la tarea que el diseñador o programador desee. Uno de los aspectos interesantes de los FPGA es que mientras más recursos internos posean alcanza a implementar diseños más complejos y al crear diseños digitales se tiene configurado un hardware que no requiere componentes externos para su funcionamiento.

Las FPGA se basan en una idea revolucionaria que combina la alta densidad y versatilidad de los Gate Arrays (Matrices de Puertas) con la popularidad y ventajas de los dispositivos lógicos programables PLDs (Programmable Logic Devices). Las FPGA se caracterizan por poseer una arquitectura en forma de arreglo de compuertas, con una matriz de celdas lógicas CLBs (Configurable Logic Blocks) rodeadas por celdas IOBs (Input-Output Buffers), segmentos metálicos interconectados pueden ser unidos de manera arbitraria por switches programables PSM (Programmable Switch Matrix) para formar señales deseadas entre celdas. El esquema de la arquitectura básica se muestra en la figura 2.3 [2].

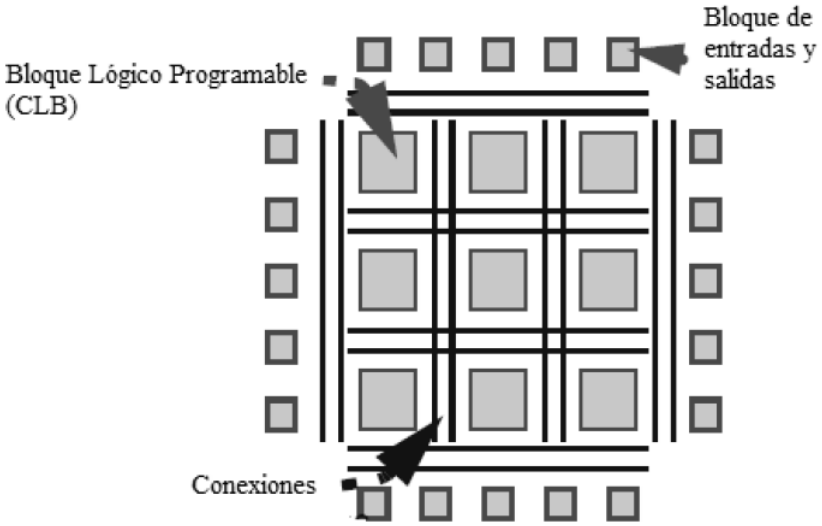


Figura 2. 3. Arquitectura Básica de una FPGA.

2.2.5.1 BLOQUES LÓGICOS CONFIGURABLES.

Los CLBs (Configurable Logic Blocks) proveen los elementos funcionales para construir la lógica demandada por el diseñador. Por medio de estos se pueden implementar tanto funcionales secuenciales como booleanas. Un CLB se basa principalmente de LUTs (Look-Up Tables, Tablas de Referencia), que se usan como generadores de funciones extraídas del diseño, y que minimizan el tiempo de propagación y compuertas en el chip. Además, contienen registros y arreglos de multiplexores para facilitar el almacenamiento de señales y la conmutación de salida desde cada una de las LUTs hacia los registros.

2.2.5.2 BLOQUES DE ENTRADA-SALIDA.

Los IOBs (Input/Output Blocks) son la interfaz entre el empaquetamiento del chip, más específicamente los pines y las líneas de señales internas. Cada IOB controla un paquete de pines, y puede ser configurado para manejar señales de entrada, salida o bidireccionales. Las señales de entrada son configuradas para ir directamente a los canales de ruteo, o a los registros de entrada sin ningún cambio; o por lo contrario, estas señales pueden ser negadas o retrasadas según el diseño. Las señales de salida pueden ser almacenadas y llevadas a los pines mediante buffers de salida.

2.2.5.3 CANALES DE RUTEO.

Las conexiones internas de una FPGA están compuestas de segmentos metálicos con puntos conmutables-programables y matrices de conmutación para implementar la ruta deseada. El ruteo se establece mediante su jerarquía para obtener una distribución de caminos automática y eficiente en la FPGA. En las FPGAs se diferencian varios tipos de interconexión entre sus componentes como lo son: las rutas de CLBs, rutas de IOBs en forma de anillo alrededor de los CLBs, rutas generales usadas para distribuir señales internas como de reloj o reset, y rutas de acuerdo con la longitud del segmento (líneas de longitud sencillas, dobles, cuádruples, largas).

2.2.6 LENGUAJES DE DESCRIPCIÓN HARDWARE.

Los lenguajes de descripción de hardware (HDLs, Hardware Description Languages) vienen utilizándose desde los años 70 en los ciclos de diseño de sistemas digitales asistidos por herramientas de CAD electrónico. En los años 80 aparece el lenguaje VHDL (de sus siglas en inglés VHSIC Hardware Description Language, y a su vez VHSIC significa Very High Speed Integrated Circuit) que, aprovechando la disponibilidad de herramientas hardware y software cada vez más potentes y asequibles y los adelantos en las tecnologías de fabricación de circuitos integrados, logra imponerse como herramienta imprescindible en el desarrollo de nuevos sistemas.

2.2.6.1 VHDL.

VHDL, se trata de un lenguaje de descripción y modelado de hardware, es decir que mediante él se puede describir la forma de comportarse de un circuito electrónico; el comportamiento puede ser llevado a algún dispositivo que dispondrá de sus propios componentes con los que lograr ese comportamiento deseado, la FPGA es un ejemplo de dicho dispositivo. La forma de comportarse del circuito diseñado es independiente del hardware donde se implementará.

VHDL es un lenguaje con una sintaxis amplia y flexible (similar a ADA y se diferencia de éste en que su semántica está orientada al modelado del hardware), que permite el modelado estructural del comportamiento de un sistema digital conocido y el desarrollo de modelos de simulación, en flujo de datos y de comportamiento hardware. Fue concebido inicialmente para modelado y simulación, no para síntesis. Por tanto, no todas las descripciones VHDL son sintetizables, esto es, no todas las descripciones tienen una equivalencia en el nivel de puertas. Por otro lado, una misma funcionalidad puede describirse de muchas maneras.

2.2.6.1.1 CARACTERÍSTICAS DEL LENGUAJE VHDL.

El lenguaje VHDL especifica los circuitos electrónicos en un formato adecuado para ser interpretado tanto por máquinas como por personas. Se trata además de un lenguaje formal, es decir, no resulta ambiguo a la hora de expresar el comportamiento o representar la estructura de un circuito. Está normalizado (norma IEEE-1076), por lo que existe un único modelo para el lenguaje, permitiendo minimizar errores de comunicación y problemas de compatibilidad. Es un lenguaje ejecutable, lo que permite que la descripción textual del hardware se materialice en una representación del mismo utilizable por herramientas auxiliares tales como simuladores y sintetizadores lógicos, compiladores de silicio, simuladores de tiempo, de cobertura de fallos, herramientas de diseño físico, etc.

Este lenguaje posee un amplio rango de capacidad descriptiva ya que posibilita la descripción del hardware con distintos niveles de abstracción, pudiendo adaptarse a distintos propósitos y utilizarse en las sucesivas fases que se dan en el desarrollo de los diseños. Además, permite dividir o descomponer un diseño hardware y su descripción VHDL en unidades más pequeñas.

2.2.6.1.2 DESCRIPCIÓN DE UN DISPOSITIVO EN VHDL.

Existen dos formas de describir un circuito en VHDL. Por un lado se puede describir un circuito indicando los diferentes componentes que lo forman y su

interconexión, de esta manera se tiene especificado un circuito y sabemos cómo funciona; esta es la forma habitual en que se han venido describiendo circuitos y las herramientas utilizadas para ello han sido las de captura de esquemas y las descripciones netlist (lista de interconexiones de un diseño electrónico).

VHDL también se puede utilizar para la descripción comportamental o funcional de un circuito, es decir, sin necesidad de conocer la estructura interna de un circuito es posible describirlo explicando su funcionalidad. Esto es especialmente útil en simulación ya que permite simular un sistema sin conocer su estructura interna.

La realización del modelo hardware de un dispositivo en VHDL consiste en la elaboración de dos unidades de código VHDL (ejemplificado en la Figura 2.4):

- La Declaración de Entidad: es la unidad de diseño VHDL que sirve para especificar el interfaz de los dispositivos. Cumple, por tanto, funciones equivalentes a las de los símbolos en las representaciones gráficas.
- El Cuerpo de Arquitectura: es la unidad de diseño VHDL que sirve para especificar el funcionamiento de un dispositivo identificado por una determinada Declaración de Entidad, por lo que se puede considerar el equivalente a las tablas de verdad o a los cronogramas.

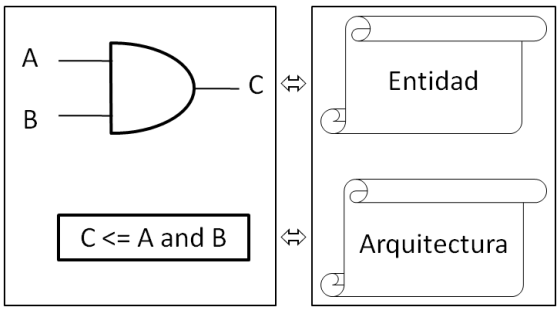


Figura 2. 4. Descripción de un dispositivo en VHDL.

2.2.6.2 MODOS DE DESCRIPCIÓN DE CIRCUITOS LÓGICOS.

El lenguaje de descripción de hardware VHDL cuenta con diferentes modos de llevar a cabo la descripción. Normalmente su aprendizaje se comienza desde la perspectiva del diseñador tradicional de hardware, utilizando las construcciones del lenguaje que permiten una descripción estructural desde las puertas lógicas hacia arriba. Este enfoque resalta la correspondencia existente entre la realidad y el lenguaje pero oculta la verdadera potencia del modelo temporal soportado por el lenguaje. La descripción comportamental y la ejecución concurrente de procesos manifiestan la verdadera potencia del VHDL. [6]

VHDL permite escoger entre dos tipos de retardo: el transporte (transport) y el inercial (inertial). El modelo de transporte propaga cualquier cambio que se produzca, mientras que el inercial filtra aquellos cambios de duración inferior a un mínimo. El modelo de transporte es el que refleja una línea de transmisión ideal, mientras que el modelo inercial es el que rige el comportamiento de una puerta lógica. Por defecto, VHDL supone que las asignaciones a señal siguen el modelo inercial.

La descripción puede ser realizada mediante sentencias secuenciales, sentencias concurrentes y/o sentencias estructurales. Las sentencias secuenciales permiten modelar la funcionalidad de un componente, se pueden clasificar en sentencias de asignación (a variable o a señal), sentencias condicionales (if, case), sentencias iterativas (loop, exit, next), otras sentencias (wait, assert, null) y llamadas a subprogramas. Las sentencias concurrentes son aquellas que se ejecutan en paralelo, por lo que no están incluidas en ningún proceso, sino que aparecen en la arquitectura del modelo. Las sentencias estructurales son también sentencias concurrentes pues se ejecutan en paralelo con cualquier otra sentencia concurrente de la descripción y aparecen en la arquitectura de un modelo fuera de cualquier proceso, declaradas como una serie de sentencias dedicadas a la descripción estructural de hardware.

El lenguaje de descripción hardware VHDL permite estos tres modos de descripción [6]:

- Modelado de flujo de datos (RTL, Register Transfer Logic), se implementa declarando una sucesión temporal de la evolución de las diferentes señales del modelo descrito.
- Modelado comportamental, permite escribir funciones complejas sin recurrir a su implantación física con lo que se revela como una herramienta de gran rendimiento ya que se proporciona la función que relaciona la salida con la entrada.
- Modelado estructural, especificando los bloques que componen un circuito y sus interconexiones. Cada bloque integrante debe contar con su descripción previa de manera que se construye una jerarquía de descripciones donde las inferiores dan lugar a superiores más complejas y así sucesivamente.

2.2.6.3 ETAPAS DE DISEÑO EN VHDL.

En un diseño con VHDL existen una serie de etapas que guían ese proceso, esas etapas se denominan flujo de diseño (Wakerly, 2001). La escritura del código en VHDL es uno de los pasos, y está interconectada con las demás fases del diseño de un circuito.

Existen siete pasos, los cuatro primeros (diagrama de bloques jerárquico, codificación, compilación y simulación-verificación) se llaman pasos del componente frontal y los tres últimos son los pasos (síntesis, verificación con tiempo y ajuste/ubicación y enrutado) del componente posterior. Los pasos de componente frontal se refieren a los que el diseñador puede detallar más de cerca con las herramientas de diseño. Los pasos de componente posterior también requiere de herramientas adecuadas, la diferencia se encuentra en que éstos se relacionan con la adaptación del diseño a un circuito que pueda ser implementado y a la verificación de funcionamiento tomando en cuenta retardos de tiempo similares a los que pueden presentarse en el circuito en físico y las limitaciones de rutado de los dispositivos utilizados. El flujo de diseño se inicia desde los pasos de

componente frontal y sigue con los pasos de componente posterior. Durante el desarrollo del diseño se puede retornar a pasos previos cuando el diseño no satisface el funcionamiento deseado en algunas de las fases. [7]

2.2.7 COMUNICACIONES SERIALES.

La comunicación serial se basa en la transmisión secuencial a través de una misma línea de los bits que componen un dato. Existen dos tipos de comunicaciones seriales: la síncrona y asíncrona. En la comunicación serial síncrona además de una línea sobre la cual se transmitirán los datos se necesita de una línea la cual contendrá los pulsos de reloj que indicaran cuando un dato es válido. Como ejemplo se tiene de este tipo de comunicación esta: I^2C, ONE WIRE y SPI.

En la comunicación serial asíncrona, no son necesarios los pulsos de reloj. La duración de cada bit está determinada por la velocidad con la cual se realiza la transferencia de datos. Si el receptor no está sincronizado con el transmisor, este desconoce cuándo se van a recibir los datos. Por lo tanto el transmisor y el receptor deberán tener los mismos parámetros de velocidad, paridad, número de bits del dato transmitido y de BIT de parada.

En los circuitos digitales, cuyas distancias son relativamente cortas, se pueden manejar transmisiones en niveles lógicos TTL (0-5V), pero cuando las distancias aumentan, estas señales tienden a distorsionarse debido al efecto capacitivo de los conductores y su resistencia eléctrica. El efecto se incrementa a medida que se incrementa la velocidad de la transmisión. Todo esto origina que los datos recibidos nos sean igual a los datos transmitidos, por lo que no se puede permitir la transferencia de datos.

Una de las soluciones más lógica es aumentar los márgenes de voltaje con que se transmiten los datos, de tal manera que las perturbaciones a causa de la línea se puedan corregir. [8]

2.2.7.1 LA NORMA RS-232.

Ante la gran variedad de equipos, sistemas y protocolos que existen surgió la necesidad de un acuerdo que permitiera a los equipos de varios fabricantes comunicarse entre sí. La EIA (Electronics Industry Association) elaboró la norma RS-232, la cual define la interfaz mecánica, los pines, las señales y los protocolos que debe cumplir la comunicación serial. Las normas RS-232 cumplen con los siguientes niveles de voltaje:

- Un "1" lógico es un voltaje comprendido entre –5v y –15v en el transmisor y entre -3v y –25v en el receptor.
- Un "0" lógico es un voltaje comprendido entre +5v y +15v en el trasmisor y entre +3v y +25 v en el receptor.

El envío de niveles lógicos (bits) a través de cables o líneas de transmisión necesita la conversión a voltajes apropiados. En general cuando se trabaja con familias TTL y CMOS se asume que un "0" lógico es igual a cero Volts y un "1" lógico es igual a cinco Volts. La importancia de conocer esta norma radica en que los niveles de voltaje son diferentes a los que utilizan los microcontroladores y los demás circuitos integrados. Por lo tanto se necesita de una interfaz que haga posible la conversión de los niveles de voltaje a los estándares manejados por los CI TTL. [8]

Existen diferentes soluciones para la interfaz de TTL a RS232, una de ellas es la implementación de un circuito integrado que cumpla con la norma, como por ejemplo: MAX232, MAX220, DS14C232, MAX233, LT1180A. Otra de las soluciones para la interfaz son circuitos basados en transistores o compuertas lógicas.

2.2.8 MICROSOFT .NET.

Microsoft .NET es una plataforma de desarrollo y ejecución de aplicaciones. Esto quiere decir que no sólo brinda todas las herramientas y servicios que se necesitan para desarrollar modernas aplicaciones, sino que también provee los mecanismos robustos,

seguros y eficientes para asegurar que la ejecución de las mismas sea óptima. El objetivo principal para el desarrollo de esta plataforma fue el de proveer un entorno específicamente diseñado para el desarrollo y ejecución del software de aplicaciones (ya sean de formularios Windows, de consola, aplicaciones Web, aplicaciones móviles, etc.) que puedan ser tanto publicadas como accedidos a través de Internet de forma independiente del lenguaje de programación, modelo de objetos, sistema operativo y hardware utilizados tanto para desarrollarlos como para publicarlos. Los componentes principales de la plataforma .NET son [9]:

- Un entorno de ejecución de aplicaciones, también llamado "Runtime".
- Un conjunto de bibliotecas de funcionalidades y controles reutilizables, con una enorme cantidad de componentes ya programados listos para ser consumidos por otras aplicaciones.
- Un conjunto de lenguajes de programación de alto nivel, junto con sus compiladores, que permitirán el desarrollo de aplicaciones sobre la plataforma .NET.
- Un conjunto de utilitarios y herramientas de desarrollo para simplificar las tareas más comunes del proceso de desarrollo de aplicaciones.

Se dice que es una plataforma de ejecución intermedia, ya que las aplicaciones .NET no son ejecutadas directamente por el sistema operativo, como ocurre en el modelo tradicional de desarrollo. En su lugar, las aplicaciones .NET están diseñadas para ser ejecutadas contra un componente de software llamado Entorno de Ejecución (Common Language Runtime, CLR). El CLR es el entorno que usan las aplicaciones escritas en diversos lenguajes en tiempo de ejecución. Este componente es el encargado de manejar el ciclo de vida de cualquier aplicación .NET, iniciándola, deteniéndola, interactuando con el Sistema Operativo y proveyéndole servicios y recursos en tiempo de ejecución.

La infraestructura .NET está comprendida por: el *Framework .NET*, *Microsoft Visual Studio .NET*, *.NET Enterprises Servers* y *Microsoft Windows .NET*.

2.2.8.1 .NET FRAMEWORK.

Para el desarrollo y ejecución de aplicaciones en este entorno tecnológico, Microsoft proporciona el conjunto de herramientas conocido como .NET Framework que incluye compiladores de lenguajes como C#, Visual Basic.NET, Managed C++ y JScript.NET, específicamente diseñados para crear aplicaciones para él.

Se le llama Framework, ("entorno de trabajo"), a las Bibliotecas de Clase Base, (también llamadas BCL) y el Common Language Runtime o CLR (ver ítem 2.2.8).

Las Bibliotecas de Clase son una estructura jerárquica de clases que envuelven diversas funcionalidades como acceso a archivos, hilos de ejecución, acceso a base de datos, etc. [10]

2.2.8.2 VISUAL STUDIO.

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, que permite escribir un programa más rápido, ofreciendo la posibilidad de compilarlo desde el mismo editor.

2.2.8.2.1 VISUAL STUDIO 2010 EXPRESS.

Es una versión del entorno de desarrollo integrado (IDE) de Microsoft, que permite crear aplicaciones utilizando el lenguaje de programación Visual Basic .NET. Es la versión utilizada para el desarrollo de la interfaz de usuario desarrollada en este Trabajo de Grado.

Dispone de un editor de código, un depurador y compilador; y permite agregar elementos gráficos fácilmente, arrastrando y soltando elementos al editor visual.

Incluye Intelli Sense Code Snippets, una colección de librerías para agregar a los proyectos, que evitan tener que escribir código para determinadas funciones que ya están hechas.

Esta versión Express es una versión reducida del **Visual Basic .NET** que forma parte del **Visual Studio**, el paquete profesional que no es gratuito como lo es esta aplicación. [11]

2.2.8.2.2 LENGUAJE DE PROGRAMACIÓN: VISUAL BASIC.

Visual Basic constituye una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan en gran parte a partir del diseño de una interfaz gráfica. En una aplicación Visual Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interfaz gráfica. Es por tanto un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos (POO). [12]

2.2.8.2.2.1 PROGRAMACIÓN ORIENTADA A OBJETOS (POO).

La programación orientada a objetos permite descomponer un problema en bloques relacionados. Cada bloque pasa a ser un objeto auto contenido que posee sus propios datos e instrucciones. De esta manera, la complejidad se reduce y se pueden realizar programas largos de manera más sencilla, mediante la combinación de varios bloques.

Los elementos básicos de la programación orientada a objetos son [12]:

- **Objetos:** es un ente o entidad que tiene atributos propios (propiedades) y unas formas de operar sobre ellos (métodos). Por tanto, un objeto tiene variables que especifican su estado y operaciones que definen su comportamiento.
- **Propiedades:** representan las características de los objetos. Hay propiedades particulares que las poseen solo algunos objetos y otras genéricas que las poseen todos los objetos.

- **Métodos:** los métodos son funciones asociadas a un objeto; pero a diferencia de los procedimientos no son programadas por el usuario, sino que vienen ya programadas en el lenguaje de programación. Cada tipo de objeto tiene sus propios métodos.
- **Eventos:** un evento es la capacidad de un objeto de reaccionar cuando ocurre una determina acción (acción y reacción). Como respuesta a un evento se envía un mensaje y se ejecuta un determinado método (procedimiento). Cada objeto responde a un conjunto de eventos. Como respuesta a un evento se ejecuta un determinado procedimiento que realiza la acción programada por el usuario para ese evento en concreto.
- **Mensajes:** Un mensaje es un llamado a un método, de tal forma que cuando un objeto recibe un mensaje, la respuesta a ese mensaje es ejecutar el procedimiento asociado. Cuando se ejecuta un programa orientado a objetos, los objetos están constantemente recibiendo, interpretando y respondiendo a mensajes de otros objetos.
- **Clases:** Una clase es una descripción para producir objetos de esa clase o tipo. Es decir, se trata de una generalización de un tipo específico de objetos. En otras palabras, un objeto es una variable del tipo definido por una clase.

La POO tiene las características de Abstracción, Encapsulamiento, Herencia y Polimorfismo. [12]

2.2.8.2.2.2 ELEMENTOS DEL ENTORNO DE DESARROLLO DE VISUAL BASIC.

- **Barra de herramientas:** Permite un acceso rápido a los comando más utilizados, ver figura 2.5 [13].

Figura 2. 5. Barra de herramientas Visual Basic.

- **Diseñador de formularios:** Es la ventana en la que se diseña la interfaz de la aplicación, en ella se pueden agregar controles gráficos e imágenes, ver figura 2.6 [13]. Cada formulario de una aplicación aparecerá en su propia ventana.

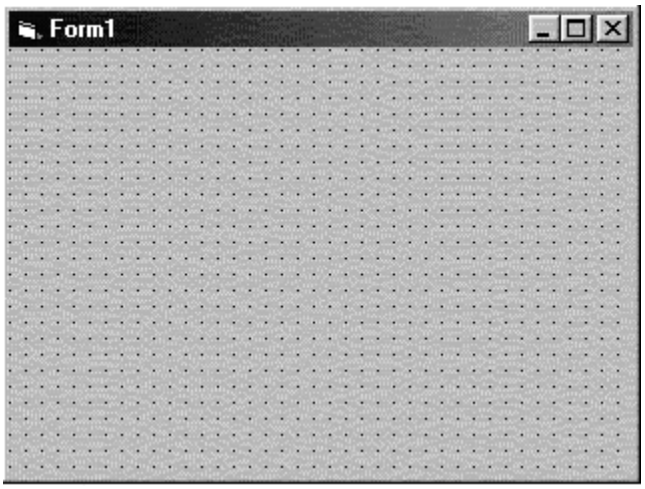


Figura 2. 6. Diseñador de formularios Visual Basic.

- **Cuadro de herramientas:** En el cuadro de herramientas se puede encontrar un conjunto de herramientas que permiten insertar los objetos o controles en el formulario durante en tiempo de diseño, ver figura 2.7 [13].

Figura 2. 7. Cuadro de herramientas Visual Basic.

- **Ventana de propiedades:** Los objetos tiene asociados unas propiedades que describen sus atributos, valores, comportamiento y apariencia del objeto; ver figura 2.8 [13].

Figura 2. 8. Ventana de propiedades Visual Basic.

Las opciones de esta ventana son [13]:

- **Lista desplegable de objetos**: Donde se puede visualizar el nombre de los objetos de la aplicación.
- **Lista de propiedades del objeto seleccionado**: Al seleccionar un objeto con la lista desplegable anteriormente mencionada aparecerán las propiedades del mismo (name, visible, appearence, borderstyle, etc.). En la lista de propiedades se pueden modificar las propiedades del objeto. Se puede visualizar de dos formas, por categoría o alfabéticamente.

- **Ventana de proyectos:** Contiene la lista de los archivos que forman parte de la aplicación, ver figura 2.9 [13].

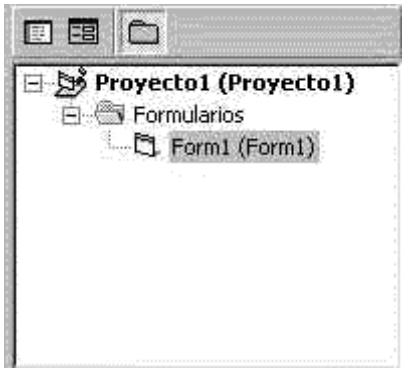


Figura 2. 9. Ventana de proyectos Visual Basic.

Los tipos de archivos que se pueden incluir en un proyecto son [13]:

- Archivo de Proyecto: Es el que realiza el seguimiento de todos los ficheros que forman parte de la aplicación. Se guarda en un fichero con la extensión .VBP
- Archivo de Recursos: Aquí se guardan cadenas de texto, mapas de bits, y demás datos que puedan modificarse sin tener que volver a modificar el código. Se guardan con una extensión .RES
- Módulo de Formulario: Contiene controles y código, sólo hay uno por formulario. Se guardan con extensión FRM

- Módulo de Clase: Son similares a los módulos de formulario. Se guardan con la extensión .CLS.
- Módulo Estándar: Sólo pueden contener código. Tienen una extensión .BAS
- Controles ActiveX: Controles que se pueden añadir al cuadro de herramientas e incluirlos en un formulario.

- **Ventana editor de código:** En esta ventana es donde se incluye el código de la aplicación, ver figura 2.10 [13]. Se creará una ventana de código para cada formulario o módulo de la aplicación. Para tener acceso a la ventana de edición, la forma más sencilla es hacer doble clic sobre el formulario o sobre el objeto al cual quiera incluir código.

Figura 2. 10. Ventana editor de código Visual Basic.

2.2.8.2.2.3 CONTROLES MÁS HABITUALES.

- **Label:** Suelen acompañar a otros controles para informar de su utilidad. También se utilizan para mostrar información al usuario. Con este tipo de control no se suele interactuar, por tanto no suelen codificarse manejadores de eventos para ellos. [14]
- **Button:** Son controles que se utilizan para realizar una determinada acción (realizar cálculos, visualizar datos, guardar información, etc). [14]

- **TextBox:** Se utilizan para recoger información del usuario necesaria para almacenar o calcular resultados. [14]
- **ListBox, ComboBox y CheckedListBox:** Son controles que permiten al usuario seleccionar uno, varios o ninguno de los elementos que se muestran en ellos. Dependerá del tipo de control y características que se le definan en tiempo de diseño. **El control Combo** permite tres formas diferentes de mostrar los elementos. [14]
- **Panel y GroupBox:** Los controles GroupBoxde formularios Windows Forms se utilizan para proporcionar un agrupamiento identificable para otros controles. Normalmente, los cuadros de grupo se utilizan para subdividir un formulario por funciones. La agrupación de todas las opciones en un cuadro de grupo ofrece al usuario una pista visual lógica. Además, en tiempo de diseño es fácil mover todos los controles, ya que, al mover el control GroupBox, también se mueve todo su contenido. Los controles GroupBoxy Panel son similares; sin embargo, el control GroupBoxes el único de los dos que muestra un título y el control Panel es el único de los dos que puede tener barras de desplazamiento (utilizando la propiedad A*utoScroll*). [14]
- **CheckBox y RadioButton:** Son controles que permiten seleccionar o no seleccionar una determinada opción. El RadioButtones un control que no se utiliza de forma separada, sino que forma parte de un grupo. Es parecido al Checkbox (puede estar activado o desactivado) pero en este caso sólo un elemento del grupo puede estar activado. [14]
- **Timer:** El componente Timer produce un evento a intervalos regulares. [14]
- **Datagridview:** Permite visualizar datos en una cuadricula personalizable. Manipulación (operaciones) y modificación (cambiar, guardar) de datos procedentes de una base de datos (tablas) tales como consultas vistas etc. También datos que se ingresen en tiempo de ejecución, o por el código directamente. Las propiedades de las columnas son diferentes a las propiedades del control [15], a ellas se tiene acceso por medio de la flecha desplegable que aparece en la parte derecha de la cuadricula, de la cual se despliega un cuadro con las opciones de “Agregar Columnas” Y “Editar Columnas”.

2.2.8.2.2.4 PASOS PARA LA CREACIÓN DE UN PROGRAMA BAJO VISUAL BASIC.

- **Creación de un interfaz de usuario**. Esta interfaz será la principal vía de comunicación hombre máquina, tanto para salida de datos como para entrada. Será necesario partir de una ventana - Formulario - a la que se le irá añadiendo los controles necesarios.
- **Definición de las propiedades de los controles u objetos que se hayan colocado en ese formulario**. Estas propiedades determinarán la forma estática de los controles, es decir, como son los controles y para qué sirven.
- **Generación del código asociado a los eventos que ocurran a estos objetos**. A la respuesta a estos eventos (clic, doble clic, una tecla pulsada, etc.) se le llama procedimiento, y deberá generarse de acuerdo a las necesidades del programa.
- **Generación del código del programa**. Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto. Sin embargo, Visual Basic ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados Módulos, en otros bloques llamados Funciones, y otros llamados Procedimientos. Estos procedimientos no responden a un evento acaecido a un objeto, sino que responden a un evento producido durante la ejecución del programa.

CAPÍTULO III.

MARCO METODOLÓGICO.

3.1 TIPO DE INVESTIGACIÓN.

Es preciso tener en cuenta el tipo de investigación a realizar ya que existen muchas estrategias para su procedimiento metodológico. Esto se refiere al tipo de estudio que se lleva a cabo con la finalidad de recoger los fundamentos necesarios de la investigación. Por tal razón, la actual investigación, se enfocó dentro de la modalidad de proyecto factible, el cual según el Manual de Trabajo de Grado de Especialización, Maestría y Tesis Doctorales de la Universidad Experimental Libertador [6], dispone que:

"La modalidad de proyecto Factible, consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viables para solucionar problemas, requerimientos o necesidades de la organización o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos".

En consecuencia este Trabajo de Grado, de acuerdo al problema planteado, está apoyado en una investigación de campo tipo descriptivo y documental. Que según el manual de la UPEL [6] destaca que:

"La investigación de campo es el análisis sistemático de problemas en la realidad con el propósito, bien sea de describirlos, interpretarlos, entender su naturaleza y factores constituyentes, explicar sus causas y efectos o producir su ocurrencia, haciendo uso de métodos característicos de cualquier paradigma o enfoques de investigaciones conocidas o en desarrollo".

3.2 METODOLOGÍA DE LA INVESTIGACIÓN.

3.2.1 DISEÑO CONCEPTUAL DEL SISTEMA.

El primer paso que se dió para la creación de este prototipo fue la concepción de su funcionamiento a nivel general para luego definir con más detalle cada una de sus partes, por lo que se utiliza la metodología Top-Down. El sistema para el control de inventarios desarrollado, se planteó como un sistema formado por tres módulos, un primer módulo de información, un segundo módulo encargado del control de dichos datos y por último un módulo administrador. En la figura 3.1se representa una visión general del sistema planteado.

Figura 3. 1. Esquema General del Sistema de Control de Inventario.

3.2.1.1 DISEÑO CONCEPTUAL DEL MÓDULO DE INFORMACIÓN.

El módulo de información es el encargado de suministrar los datos de todos los productos existentes en el establecimiento donde se implementa el sistema, es decir la cantidad de cada uno de los diferentes productos. Este módulo tiene como función emular el lector de un sistema RFID, el cual se encarga de adquirir el código del tag que se encuentra adherido al producto. Para la implementación de este módulo se crearon una serie de documentos de texto (.txt), los cuales contienen los códigos de los productos existentes y sus respectivos ejemplares; estos se envían a través del puerto serial, al módulo de control, mediante el uso del programa HyperTeminal.

3.2.1.2 DISEÑO CONCEPTUAL DEL MÓDULO DE CONTROL.

Este módulo se encarga de administrar el flujo de datos del sistema, mediante la programación de una tarjeta de desarrollo (FPGA) utilizando el lenguaje de descripción de hardware VHDL. Recibe los datos de los productos provenientes del módulo de información, los procesa y se encarga de enviar al módulo administrador las actualizaciones en cuanto a un cambio en la cantidad existente en un tipo de producto. Para el diseño de este módulo se utilizó el enfoque Bottom-Up ya que se describieron con detalle e individualmente diferentes procesos como el de recepción, transmisión, procesamiento de datos, entre otros, los cuales se enlazaron para formar el proceso completo de este módulo.

3.2.1.3 DISEÑO CONCEPTUAL DEL MÓDULO ADMINISTRADOR.

Este módulo, que consta de un ordenador, recibe la información de los productos provenientes del módulo de control y tiene como propósito registrar los eventos de entrada y salida de los productos que se producen en el módulo de información, con la hora de este ordenador, además de almacenar la cantidad existente de cada tipo de producto. Para el funcionamiento de este módulo se diseñó una interfaz gráfica que permite al usuario visualizar y descargar la información del inventario, así como también editar información de los productos; para ello se utilizó una programación orientada a objetos ya que se definen diferentes unidades lógicas que tienen las funciones de transmisión, recepción, edición y almacenamiento de datos.

3.3 TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS.

De acuerdo los objetivos planteados la información y datos necesarios para la elaboración de este Trabajo de Grado fueron recolectados en 3 etapas:

1. Problemática: Contempló la identificación de la problemática planteada con respecto al control de inventarios en las empresas.

2. Recursos y Medios: Tuvo como finalidad la búsqueda y revisión de la documentación disponible entre libros, artículos, normas, especificaciones técnicas, entre otros; relacionado con los sistemas RFID y las tarjetas de desarrollo (FPGA), su regulación y estandarización, y la tecnología destinada para su desarrollo. Todo esto para la elaboración de un marco bibliográfico que sirvió de soporte para la realización de este Trabajo de Grado.

3. Pruebas y Resultados: Contempló la recolección de datos experimentales siguiendo un protocolo de pruebas para la verificación del correcto funcionamiento de todas las funciones asociadas al prototipo del sistema construido.

3.4 ETAPAS DE LA INVESTIGACIÓN.

De acuerdo a los objetivos planteados en este Trabajo de Grado, el desarrollo de esta investigación se basó en diferentes etapas que involucraron una continuidad de actividades y procedimientos para la ejecución y cumplimiento de dichos objetivos.

3.4.1 ETAPA I - INVESTIGACIÓN DE LA DOCUMENTACIÓN REFERENTE A CADA MÓDULO QUE CONFORMA AL SISTEMA DE CONTROL PLANTEADO.

Esta etapa tuvo como finalidad la revisión de toda la documentación relacionada con los componentes y equipos necesarios para el diseño del sistema de control de inventario:

- Búsqueda y revisión de la documentación de las tarjetas de desarrollo (FPGA) requerida para el conocimiento de los detalles técnicos, los estándares y toda información relevante que fue necesaria para la configuración de la misma.
- Búsqueda y revisión del lenguaje de programación de hardware reconfigurable VHDL, para obtener un tutorial del manejo e implementación de este lenguaje, que permitió desarrollar una programación para el control de inventario.

- Búsqueda y revisión de las especificaciones técnicas del módulo lector del sistema RFID, centrando la atención en los diferentes estándares de comunicación que utiliza.

3.4.2 ETAPA II - DEFINICIÓN DEL PROTOCOLO DE COMUNICACIÓN.

Esta etapa tuvo como objetivo definir todos los aspectos correspondientes al protocolo de comunicación, de acuerdo al estándar RS232, que permite el intercambio de información entre los diferentes módulos del sistema diseñado:

- Definición de los datos en forma codificada que se transmiten del módulo de información al módulo de control, es decir, el número de bits que identifica a cada producto y la información que aporta cada uno.
- Definición de la trama de datos que se envía del módulo de control al módulo administrador, es decir, el orden y codificación en que se envían los bytes que se transmiten al módulo administrador.

3.4.3 ETAPA III - DISEÑO Y DESARROLLO DEL MÓDULO DE CONTROL.

Esta etapa tuvo como finalidad el diseño y desarrollo de la interfaz entre el módulo de información y el módulo administrador, constituida por la tarjeta de desarrollo.

- Esquematización de los procesos que conforman el módulo de control.
- Investigación y selección, en cuanto a costos y disponibilidad de una herramienta de programación y simulación, de un fabricante específico de FPGA para la adquisición de la tarjeta de desarrollo más adecuada y económica, que permitió cubrir el sistema de control de inventario.
- Prueba de la tarjeta de desarrollo para la comprobación su correcto funcionamiento.
- Adición de los puertos o componentes requeridos para compensar su falta en la tarjeta de desarrollo.
- Definición de los módulos internos mediante el lenguaje de descripción de hardware VHDL.

- Simulación, por separado, de cada módulo descrito para la comprobación su correcto funcionamiento.
- Instanciación que permitió conectar los distintos módulos o procesos mediante la asociación de sus puertos, pudiendo sintetizar los procesos en una programación final.
- Simulación del código sintetizado que permitió la verificación de su adecuado funcionamiento.
- Generación del archivo que permitió la programación de la tarjeta de desarrollo.

3.4.4 ETAPA IV – DESARROLLO Y PROGRAMACIÓN DE LA INTERFAZ GRÁFICA.

En esta etapa se basó en el desarrollo y la creación de la interfaz gráfica, para el módulo administrador, para que el usuario pueda observar la información contenida en los registros del inventario en tiempo real, donde se indica la cantidad de cada producto junto con la hora de ingreso y egreso de los mismos, además de permitir funciones de edición y generación de reportes del inventario. Todo esto mediante las siguientes actividades:

- Selección de los colores de la ventana de la aplicación que sirve de interfaz gráfica.
- Desarrollo de diagramas de flujo que permitieron identificar las diferentes actividades que se llevarán a cabo en la interfaz.
- Estructuración de la ventana de la aplicación, definiendo la ubicación de los cuadros de texto y los objetos que se encontraran dentro de ella.
- Programación referente al uso de cada objeto en la aplicación.
- Evaluación de la interfaz gráfica para comprobar su correcto funcionamiento.

3.4.5 ETAPA V – ENSAMBLAJE Y EVALUACIÓN DEL PROTOTIPO DISEÑADO.

- Ensamblaje del prototipo conectando todos los equipos y componentes que conforman el sistema.

- Evaluación del funcionamiento del prototipo ensamblado mediante su puesta en marcha y realización de una prueba del sistema completo para verificar la correcta recepción y transmisión de datos entre los diferentes módulos.
- Ejecución de las correcciones técnicas necesarias, para la obtención un sistema que cumpla los objetivos y llene las expectativas del usuario que lo implemente.

CAPÍTULO IV.

MARCO OPERACIONAL.

4.1 DOCUMENTACIÓN REFERENTE A CADA MÓDULO QUE CONFORMA AL SISTEMA DE CONTROL PLANTEADO.

Luego de una búsqueda y revisión de información y documentación en Trabajos Especiales de Grado, Trabajos de Ascenso y artículos de internet, con contenidos referentes a lo requerido para el desarrollo de este Trabajo de Grado como lo son las tarjetas de desarrollo (FPGA), el lenguaje de programación de hardware reconfigurable VHDL, además de las características más importantes del estándar definido que sirvió para definir el protocolo de comunicación utilizado; fue posible dar inicio a la creación del prototipo planteado. La documentación más destacada se encuentra en el capítulo II.

4.2 DEFINICIÓN DEL PROTOCOLO DE COMUNICACIÓN.

Para el desarrollo del prototipo se implementó el estándar de comunicación RS232 con una codificación por cada producto de la siguiente forma:

- **Protocolo de comunicación módulo de información – módulo de control.**

En vista de que el módulo de información, el cual simula el proceso de detección del lector RFID, debe transmitir al módulo de control todos los códigos de los productos detectados para que éste los procese (cuente por tipo de producto), la velocidad de comunicación se estableció en 19200 baudios. Cada producto tiene como identificación dos caracteres (basados en el código ASCII), el primero de ellos representa el código del producto a identificar y el segundo representa un ejemplar de dicho producto. La trama de datos enviada por el módulo de información tendrá las siguientes especificaciones: un bit

de start, 8 bits de datos, sin bit de paridad y dos bits de stop. Un ejemplo de una trama de datos que se envía a la tarjeta de desarrollo se muestra a continuación en la figura 4.1.

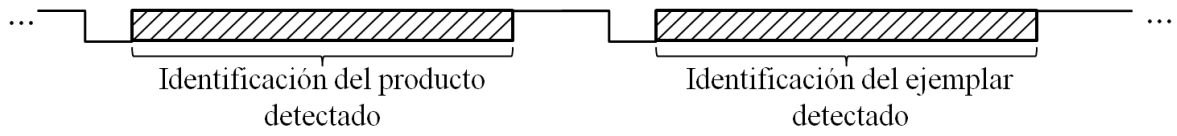


Figura 4. 1. Trama de datos del módulo de información al de control.

- **Protocolo de comunicación módulo de control – módulo administrador.**

La información de los productos ya procesada por la tarjeta de desarrollo, es enviada al módulo administrador en una trama de datos que contiene los códigos de los productos y sus respectivas cantidades, estableciéndose una velocidad de comunicación de 9600 baudios; la información por cada producto está constituida por un par de bytes codificados en el sistema Hexadecimal, donde el primer byte corresponde al código del producto y el segundo a la cantidad existente. Esta trama de datos tendrá las siguientes especificaciones: un bit de start, 8 bits de datos, sin bit de paridad y dos bits de stop. Un ejemplo de ésta se muestra en la figura 4.2.

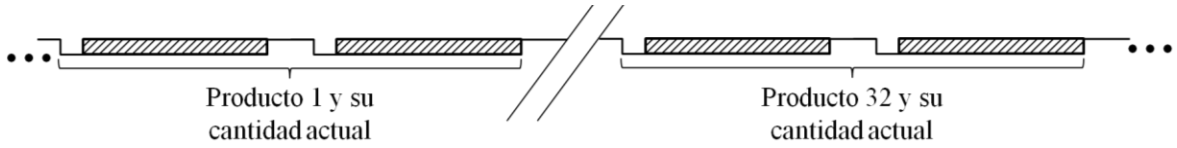


Figura 4. 2. Trama de datos del módulo de control al administrador.

4.3 DISEÑO Y DESARROLLO DEL MÓDULO DE CONTROL.

4.3.1 SELECCIÓN DEL FABRICANTE DE FPGA.

Para seleccionar un fabricante específico de FPGA, se tomaron en cuenta varios aspectos:

- Disponibilidad de una herramienta de programación y simulación preferiblemente gratuita para un fabricante en específico.

- Costo de una tarjeta de desarrollo (FPGA) que cumpla con las especificaciones del prototipo.

Una vez estudiado cada uno de estos ítems, se seleccionó un fabricante y a partir del mismo se escogerá la tarjeta de desarrollo más adecuada y económica para el desarrollo del prototipo.

4.3.1.1 HERRAMIENTA DE PROGRAMACIÓN.

La selección de una herramienta de programación se fundamentó en la utilización del lenguaje VHDL para la configuración de una tarjeta de desarrollo que cumpla con los requerimientos del sistema propuesto a diseñar. Para esto fue necesaria la búsqueda de fabricantes o distribuidores que ofrecieran aplicaciones que facilitaran la programación de control y la interfaz entre los diferentes módulos. Las compañías preseleccionadas para dicho estudio fueron Altera y Xilinx por su trayectoria en el mercado internacional y la confiablidad de los productos.

La compañía Altera dispone actualmente de la herramienta de desarrollo, simulación y programación llamada Quartus II, la cual dispone de una versión gratuita denominada Quartus II Web Edition.

La compañía Xilinx dispone de herramienta de desarrollo, simulación y programación llamada Xilinx ISE 10.1, la cual dispone de una versión gratuita llamada Xilinx ISE 10.1 WEBPACK.

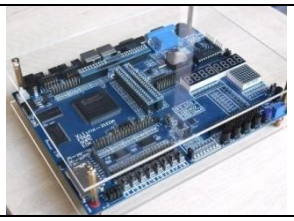
Dichas herramientas funcionan en los sistemas operativos Windows y Linux, adicionalmente ofrecen vía online toda la documentación para su correcta manipulación.

4.3.1.2 COSTOS DE UNA TARJETA DE DESARROLLO.

Para comparar los precios de una FPGA del fabricante Xilinx y una FPGA de Altera es necesario que posean características similares. Las características de FPGAs de Altera y

Xilinx en cuanto a descripción, empaque y precio entre otras, aparecen indicadas en la Tabla 4.1.

Tabla 4. 1. Tabla de comparación entre Tarjetas de Desarrollo de los fabricantes Altera y Xilinx.

		
Fabricante	Altera	Xilinx
Modelo	Cyclone II EP2C20F484C7N	Spartan-3E XC3S500E-4PQ208C
Fuentes de Reloj	24 MHz, 27 MHz, 50 MHz	50 MHz, 12 MHz
Memoria	8Mb SDRAM 512Kb SRAM 4Mb flash	64Mb SDRAM 256Kb SRAM 4 Mb
Interruptores, Indicadores y Puertos	10 interruptores 4 pulsadores 4 displays 7-segmentos 10 Leds rojos 8 Leds verdes 1 Conector de reloj externo 1 Puerto VGA 1 Puerto RS-232 2 Puertos PS/2 2 Puertos de expansión de 40-pin	8 Leds rojos 8 displays 7-segmentos 1 Alarma sonora 1 Pulsador (no programable) 5 Pulsadores (programables) 1 Circuito I2C serial EEPROM 24C04 1 Puerto VGA 1 Circuito de comunicación serial MAX232 1 Interfaz LCD (1602LCD) 1 Interfaz LCD (12864LCD) 2 Puertos PS / 2 1 Interfaz para tarjeta SD 1 Sensor de Temperatura DS18B20 8 Interruptores 1 Circuito SN74LVC16245 1 Matriz 8X8 LED 1 Circuito DAC0832
Costos	199 $ - 211,25 $	145 $ - 188 $
Enlaces	http://www.altera.com/products/devkits/altera/kit-cyc2-2C20N.html http://www.digikey.com/product-search/en?mpart=DK-CYCII-2C20N&vendor=544	http://www.aliexpress.com/item/free-shipping-Xinlinx-FPGA-Spartan-3E-XC3S500E-Xilinx-Development-Board/566728334.html http://www.ebay.com/itm/Xilinx-Spartan-3E-XC3S500E-4PQ208C-FPGA-Development-board-/230835176837

4.3.1.3 SELECCIÓN DE LA TARJETA DE DESARROLLO.

El fabricante de FPGA seleccionado para el desarrollo del prototipo de control de inventarios es Xilinx porque los costos son más bajos en comparación con otras tarjetas de desarrollo de las mismas características de otros fabricantes, la herramienta de programación y simulación es gratuita (tan solo con registrarse en la página del distribuidor) y funciona en las plataformas Windows y Linux.

4.4 ESQUEMATIZACIÓN DE LOS PROCESOS QUE CONFORMAN EL MÓDULO DE CONTROL.

4.4.1 PUERTO DE COMUNICACIÓN SERIAL (ESTÁNDAR RS-232).

Se diseñó, aplicando los aspectos conocidos del estándar de comunicación RS-232, un bloque de recepción y un bloque de trasmisión, dividiendo el diseño de cada uno de los bloques en procesos. Los procesos que componen dichos bloques son divisor(es) de frecuencia y circuito(s) de control.

1 Divisor de frecuencia.

Se implementó debido a que la frecuencia en estos bloques es diferente con respecto a la usada en la tarjeta de desarrollo seleccionada. El diseño del divisor se basó en un contador que genera una señal cuadrada con la frecuencia deseada, como se muestra en la figura 4.3. La cuenta máxima del contador para una velocidad de comunicación de 9600 baudios, se calcula como:

$$\frac{Frec.\ Tarjeta}{Frec.Salida} = \frac{50\ MHz}{9600\ kHz} = 5208$$

En la implementación de la otra velocidad de comunicación 19200 baudios, la cuenta máxima del contador es de 2604.

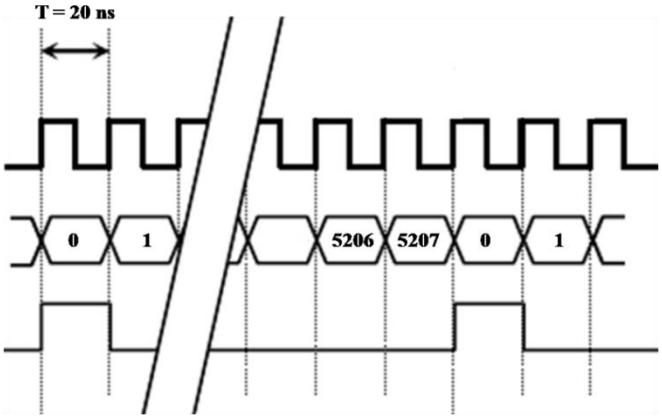


Figura 4. 3. Cronograma del divisor de frecuencia del reloj de 9600 kHz.

a. Esquema.

En la figura 4.4, se aprecia el bloque divisor de frecuencias donde la entrada es representada por ClkIn de 50 MHz y sus respectivas salidas ClkOut1 y ClkOut2 de aproximadamente 19200 kHz y 9600 kHz respectivamente.

Figura 4. 4. Bloque divisor de frecuencias.

b. Simulación.

En la figura 4.5, se muestra el resultado de la simulación del divisor de frecuencia con la entrada de la frecuencia del reloj interno de la tarjeta de desarrollo y en la salida las frecuencias resultantes.

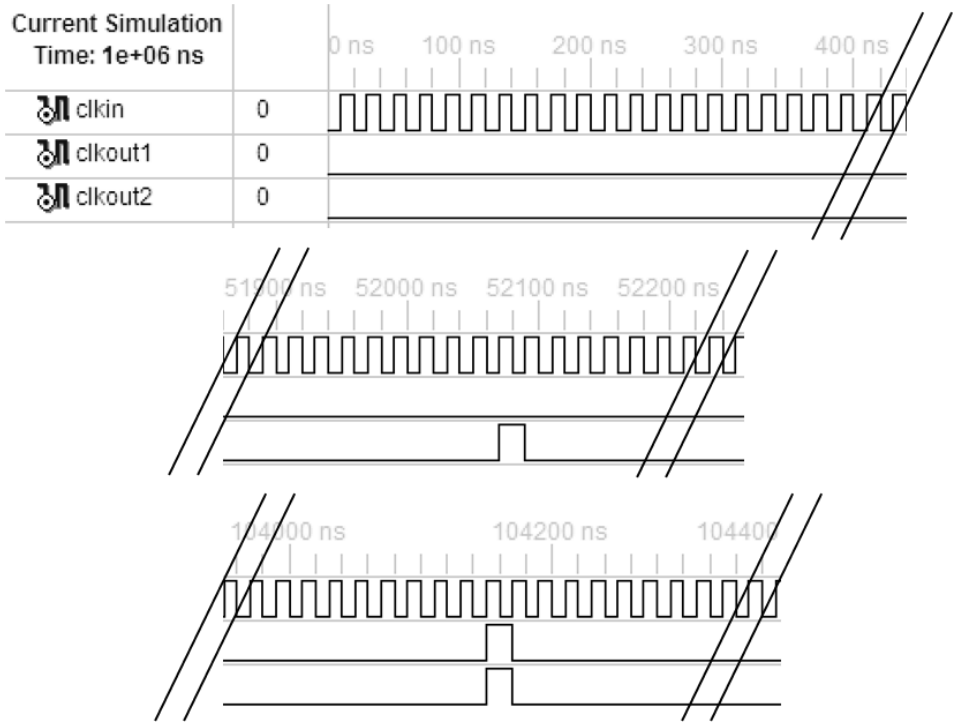


Figura 4. 5. Entradas y salidas resultantes del divisor de frecuencias simulado.

2 Circuito de control.

Éste es el encargado de controlar el momento de activación de la señal de reloj y de la captura de datos (en el caso del bloque receptor) o la secuencia de transmisión de datos. Para este diseño se utilizan señales del tipo *std_logic_vector* de 8 bits (datos recibidos o los datos a enviar), señales del tipo *std_logic* (banderas para el control de la etapa dentro del proceso o para habilitación o inhabilitación de los relojes correspondientes al proceso), y señales del tipo *natural* (cuenta de los bits recibidos o transmitidos, cuenta de la cantidad de bytes recibidos y de los bytes transmitidos).

A continuación se presenta el funcionamiento del circuito de control de acuerdo a los diferentes procesos que se ejecutan.

- **Proceso de recepción.**

Este proceso transcurre de la siguiente manera:

- Cuando el puerto observa un cambio de estado (de '1' lógico a '0' lógico) habilita el reloj de la velocidad del puerto serial.
- En cada cambio de estado (de '0' lógico a '1' lógico) del reloj se toma el valor presente en la entrada del puerto serial, durante ocho (8) ciclos de reloj; almacenando dicho valor en una variable.
- En el octavo (8) ciclo de reloj se inhabilita el reloj para esperar que se presente un nuevo dato en el puerto.

Dicho proceso se muestra en la figura 4.6.

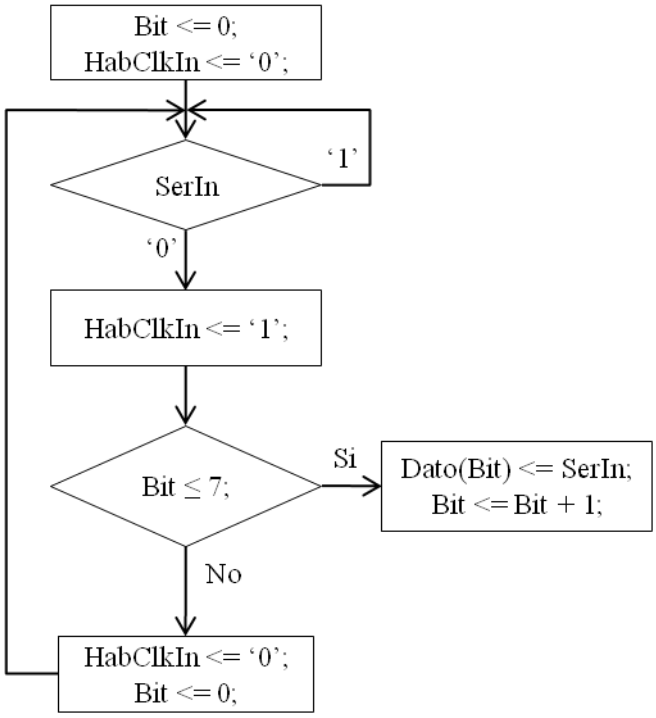


Figura 4. 6. Diagrama de flujo del proceso de recepción general.

En el presente diseño fue necesario utilizar dos procesos diferentes uno para la comunicación con el módulo de información y uno para la comunicación con el módulo

administrador; siendo en este último adicionadas condiciones para tomar el primer byte recibido como el código del producto detectado y el segundo byte recibido como el código del ejemplar censado. Dicho proceso se observa en la figura 4.7.

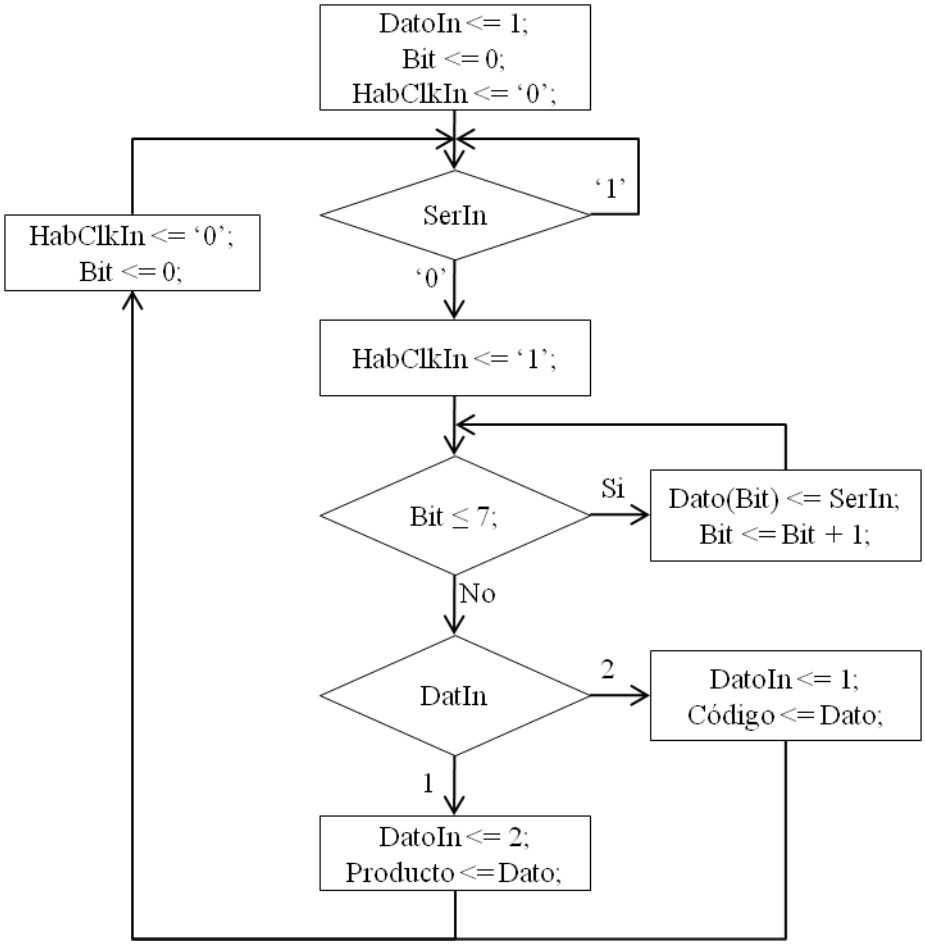


Figura 4. 7. Diagrama de flujo del proceso de recepción con selección de datos.

a. Esquema.

En la figura 4.8, se observa la representación del bloque del proceso general de recepción que posee como entradas el dato serial (DInAd) y el reloj (ClkIn) y como salida el byte recibido (Dato). Es necesario mencionar que el reloj de entrada es de 50 MHz mas internamente el bloque posee la señal del reloj de la frecuencia de comunicación.

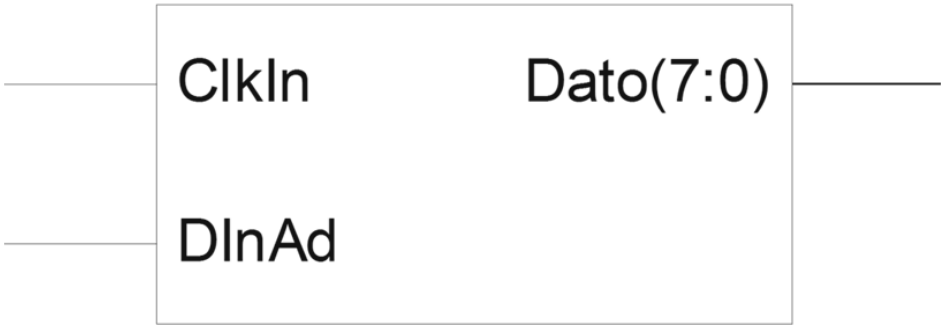


Figura 4. 8. Bloque del proceso de recepción general.

b. Simulación.

En la figura 4.9, se observa como entrada un dato serial que es capturado a la frecuencia del reloj CLkOut1 (que por razones de limitaciones de consumo de recursos del computador es de una frecuencia mayor de la requerida para la comunicación) y como resultado final se le asigna a Dato. Adicionalmente se puede observar que ClkOut1 se habilita en el momento en que comienza la recepción del dato, y una vez finalizada la recepción del mismo se inhabilita dicho reloj.

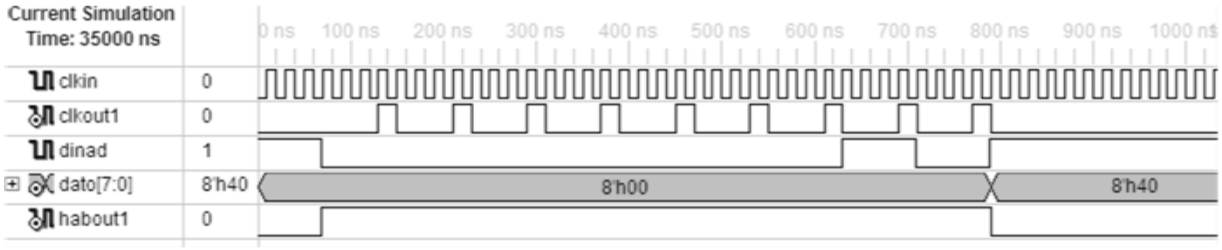


Figura 4. 9. Resultado de la simulación de las señales y datos del proceso de recepción general.

- **Proceso de transmisión.**

Este proceso, mostrado en la figura 4.10, transcurre de la siguiente manera:

- Se mantiene en '1' lógico la salida hasta que se activa la bandera de transmisión.
- Cuando se activa la bandera de transmisión se habilita el reloj y se coloca la salida en '0' lógico durante un ciclo de reloj.

- Cada vez que ocurra el cambio a estado alto del reloj se coloca en la salida del puerto el bit que corresponde enviar, teniendo en cuenta que se envía del bit menos significativo (bit 0) al bit más significativo (bit 7) del dato a enviar.
- Al finalizar el envío del byte correspondiente se coloca en el puerto de salida un '1' lógico durante dos (2) ciclos de reloj.
- Este proceso se repite hasta que se envíen toda la trama de datos.

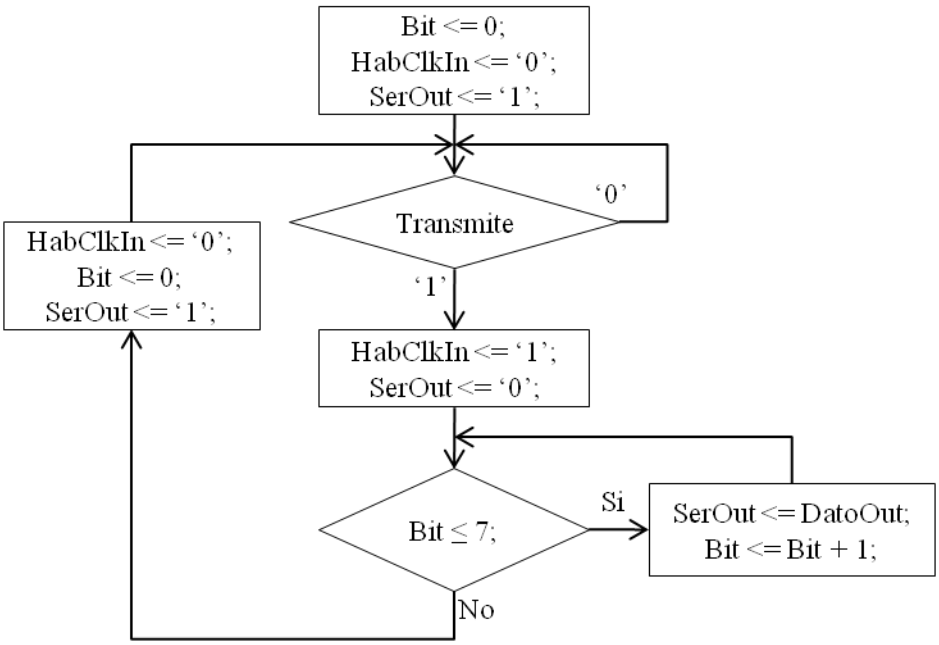


Figura 4. 10. Diagrama de flujo del proceso de transmisión de datos.

a. Esquema.

En la figura 4.11, se observa la representación del bloque del proceso de transmisión que posee como entradas la señal de transmisión (Transmite) y el reloj (ClkIn) y como salida el byte a enviar (Dato). Es necesario mencionar que el reloj de entrada es de 50 MHz mas internamente el bloque posee la señal del reloj de la frecuencia de comunicación, además el dato a enviar es una señal interna del bloque.

Figura 4. 11. Bloque del proceso de transmisión.

b. Simulación.

En la figura 4.12, se observa como entrada la señal de transmisión (Transmite) habilita el reloj ClkOut1 (que por razones de limitaciones de consumo de recursos del computador es de una frecuencia mayor de la requerida para la comunicación) para el envío de los datos almacenados internamente en el bloque (Salida), cuando finaliza el envío de información se inhabilita ClkOut1 hasta una próxima activación de Transmite.

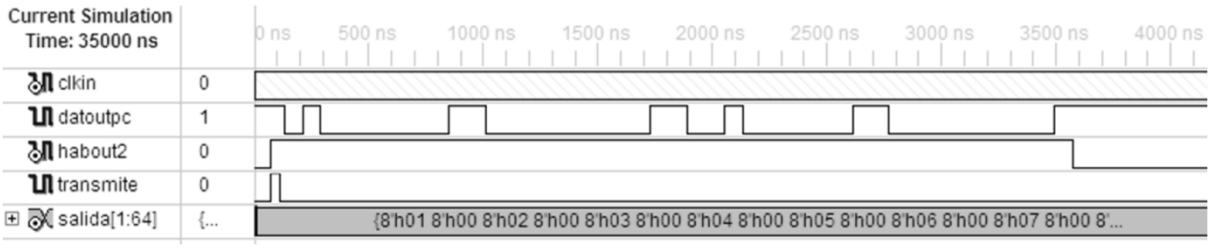


Figura 4. 12. Resultado de la simulacion del bloque del proceso de trasnmisión.

4.4.2 REINICIO DEL SISTEMA.

Todo sistema debe poseer una manera de reiniciar el dispositivo de control (tarjeta de desarrollo) con el propósito de bloquear todas las comunicaciones del sistema y permitir la limpieza de todos los registros cambiantes del sistema, todo lo mencionado anteriormente se implementó de dos maneras, una manera manual mediante un botón de reinicio (activo bajo) y otra manera a través de software (byte de reinicio enviado por el módulo administrador).

Para el botón de reinicio se tuvo en cuenta que en todo sistema mecánico los cambios de estado se toman un tiempo debido a la inercia que éste posee, y por ello se creó un proceso que evita el efecto rebote de dicho botón. Éste se puede ver en la figura 4.13.

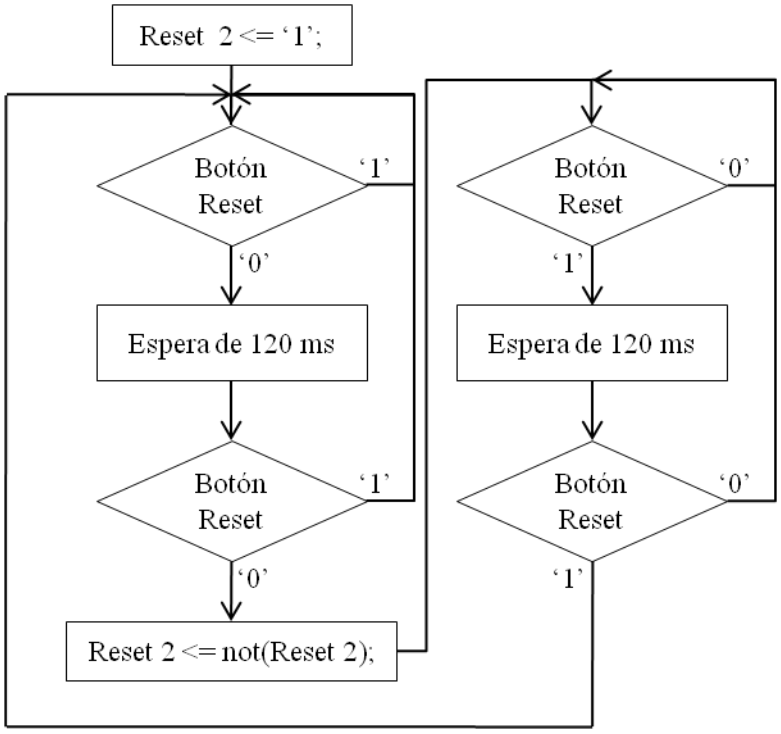


Figura 4. 13. Diagrama de flujo del reinicio manual.

El método de reinicio a través de software funciona cuando se recibe del módulo administrador un byte, específicamente el caracter "?" (en hexadecimal x"3F"), el sistema se reinicia y coloca en cero (0) todos los registros del inventario existentes en la tarjeta de desarrollo y su funcionamiento normal se comienza nuevamente cuando se recibe otro byte del módulo administrador, específicamente el caracter "@" (en hexadecimal x"40"). Este método se observa en la figura 4.14.

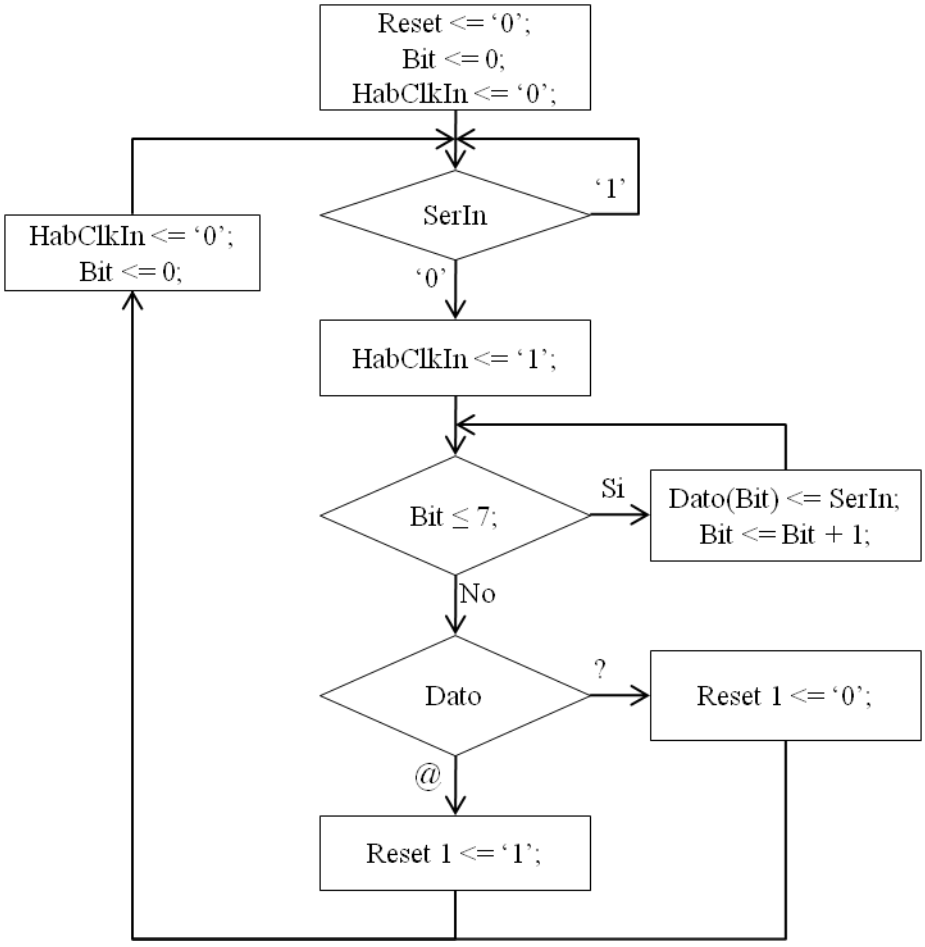


Figura 4. 14. Diagrama de flujo del reinicio mediante software.

Ambos métodos de reinicio limpian los registros del inventario existentes en la tarjeta de desarrollo, como se aprecia en la figura 4.15, pero cabe destacar que solo el método manual es el que bloquea las comunicaciones hasta que sea nuevamente presionado el botón.

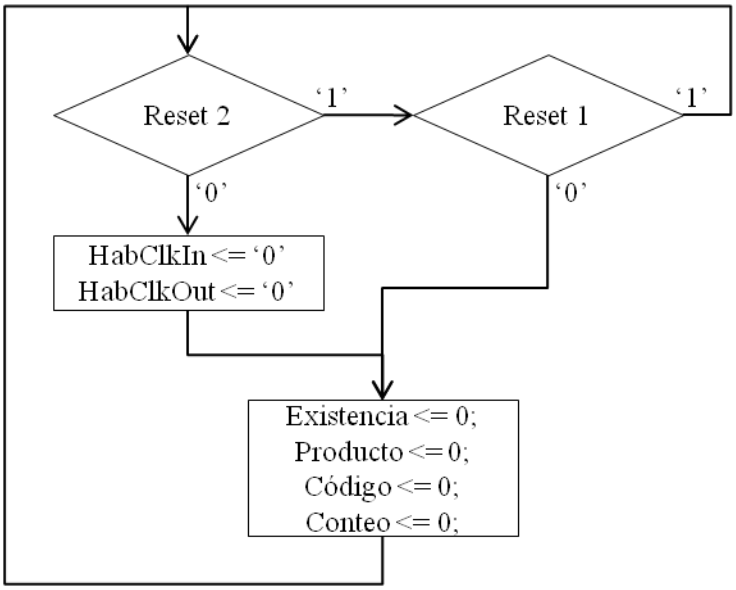


Figura 4. 15. Diagrama de flujo de los métodos de reinicio.

a. Esquema.

En la figura 4.16, se observa la representación del bloque de reinicio del sistema que posee como entradas la señal proveniente del botón (Reset) y el código recibido a través del puerto de serie (DatInPc) y como salidas las decisiones de los distintos métodos de reinicio. Es necesario mencionar que internamente éste posee el proceso de recepción general de datos serial.

Figura 4. 16. Bloque de reinicio del sistema.

b. Simulación.

En la figura 4.17, se observa como entradas el dato serial (DatInPC) que determina el estado de Reset1 y el estado del botón de Reset (que determina el estado de Reset2) que habilita/inhabilita la recepción (y transmisión) de datos por el puerto serial. La espera para evitar el efecto rebote del botón Reset por razones de limitaciones de consumo de recursos del computador es reducido durante la simulación.

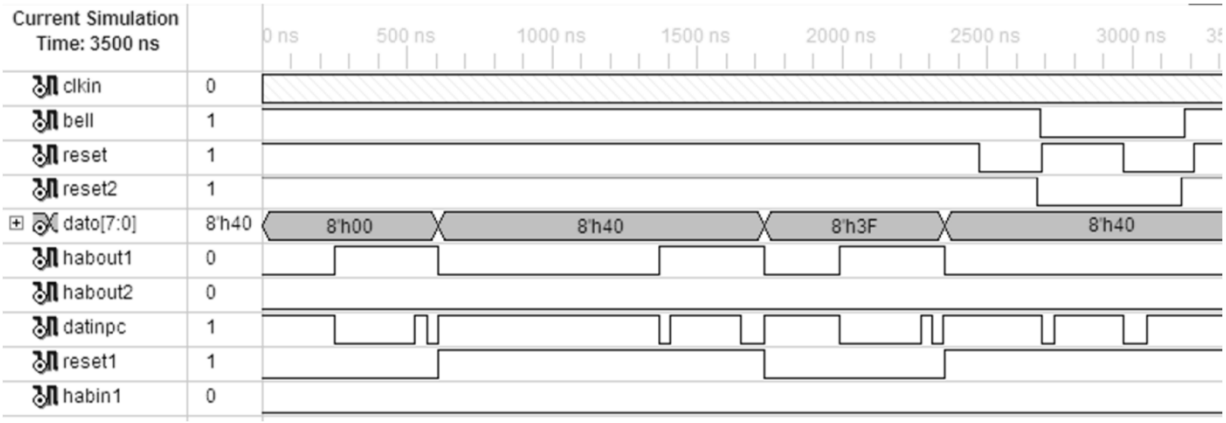


Figura 4. 17. Resultado de la simulación de los diferentes métodos de reinicio del sistema.

4.4.3 CONTROL DE LOS REGISTROS DE INVENTARIO.

Una vez terminado el diseño de la comunicación serial bajo el estándar RS-232 y el reinicio del sistema, se desarrolló el diseño de la etapa de control de inventario que ha de funcionar luego de haber recibido dos bytes (código de identificación del producto y código del ejemplar censado) del módulo de información.

El proceso, observado en la figura 4.18, se ejecuta en el siguiente orden:

- Recibidos los dos bytes correspondientes se traduce cada uno para utilizarlos como indicadores de búsqueda en un arreglo de vectores.
- Una vez ubicado el elemento de interés en el arreglo se verifica su valor (0 para indicar ausencia y 1 para indicar que se detectó previamente), si es 0 se cambia a 1 y

se espera al siguiente par de bytes. Esto evita realizar el conteo de un mismo ejemplar de producto varias veces. El presente y anterior evento se realizan cada vez que se indica la llegada de nuevos datos por el puerto serial.

- Transcurrido un intervalo de tiempo de 10 segundos se realiza el conteo de todas las unidades presentes en el inventario para ser transferidos dichos datos a la trama que contiene cada uno de los productos y la respectiva cantidad de ejemplares detectadas en el momento.
- Una vez creada la trama de datos se activa la bandera de transmisión.
- Cuando se apaga la bandera de transmisión se realiza una limpieza del registro de existencia de los productos; esto con el fin de refrescar la información contenida en los registros, ya que siempre se estará recibiendo información del módulo de información y en el momento que se deje de recibir los códigos de algún ejemplar debe ser borrada su existencia.

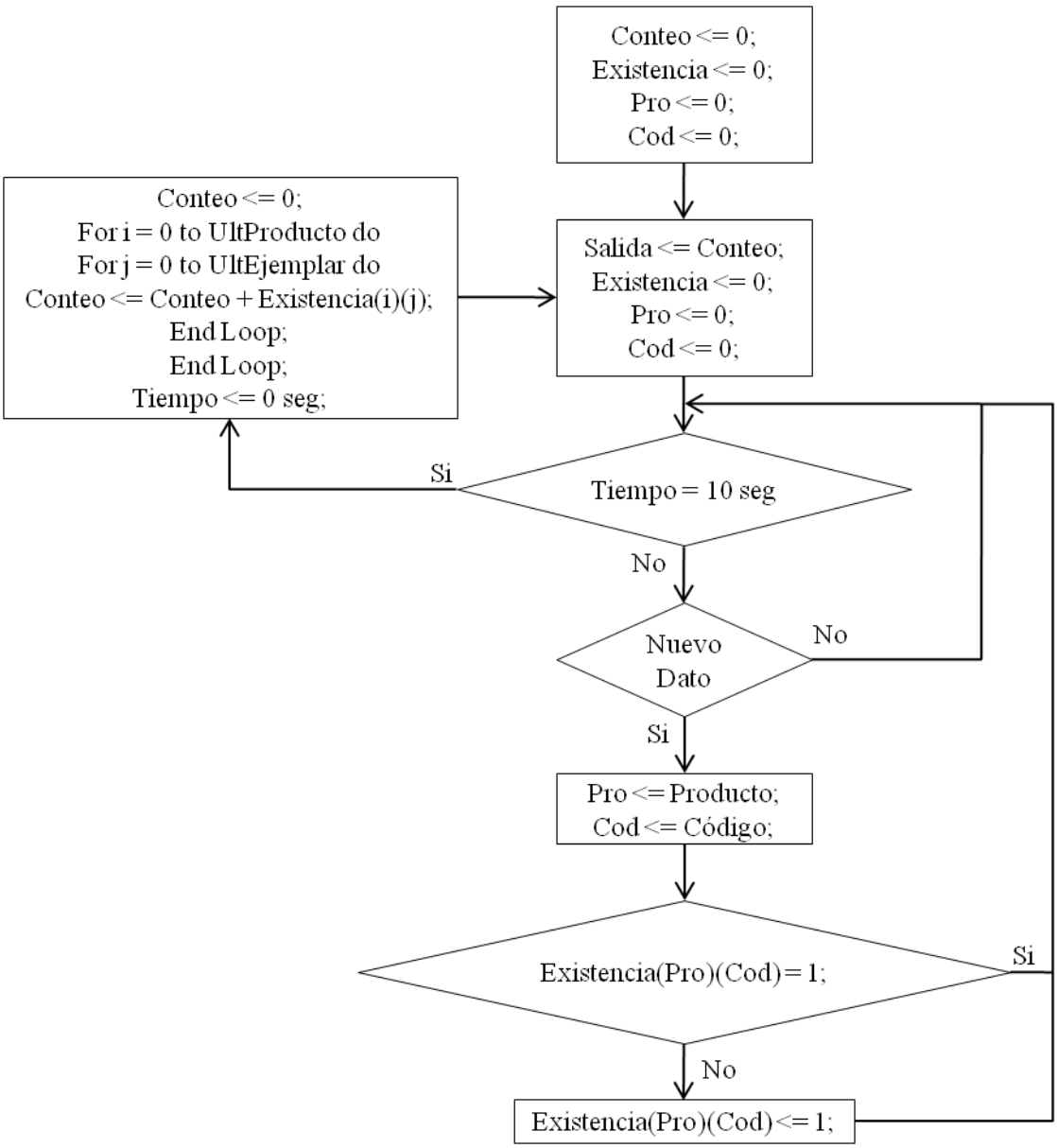


Figura 4. 18. Proceso de control de los registros de inventario.

a. Esquema.

En la figura 4.19, se observa la representación del bloque de control de registros del inventario que posee como entradas las señales de reloj (ClkIn) y dato serial (DInInf, proveniente del módulo de información) y como salida el dato serial enviado al módulo administrador.

Figura 4. 19. Bloque de control de registros del inventario.

b. Simulación.

En la figura 4.20, se observa como entrada el dato serial (DatInInf, que contiene el código del producto y ejemplar provenientes del módulo de información) y salida el dato serial a enviar al módulo administrador (DatOutPC, proveniente de la trama de datos Salida). La señal Salida será modificada un instante de tiempo antes del momento de la transmisión, en ella se reflejan las cantidades en existencia de productos (almacenados en Existencia); la señal existencia al momento de transmisión es llevada a cero para realizar un nuevo conteo de los datos provenientes del módulo de información. El proceso fue simulado con valores diferentes de velocidades de comunicación e instantes de transmisión diferentes a los reales debido a las limitaciones en los requerimientos del computador.

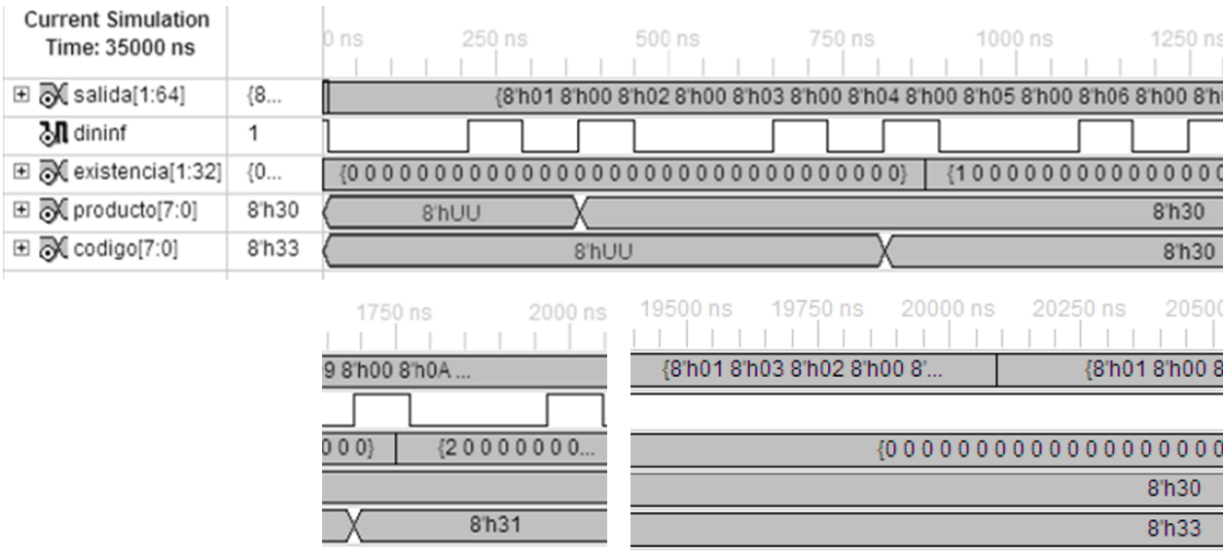


Figura 4. 20. Resultado de la simulación del control de registros de inventario.

4.5 ESQUEMA Y SIMULACIÓN DEL MÓDULO DE CONTROL.

El diseño del prototipo del módulo de control se logró unificando y relacionando los bloques antes diseñados y simulados. El funcionamiento general de éste es el siguiente: las entradas del sistema diseñado (constituidas por DInInf, DInAd y Reset) van a ser simultáneamente tomadas en cada ciclo del reloj interno de la tarjeta (ClkIn, entrada del sistema). Desde que se energiza de la tarjeta de desarrollo, ésta a través de uno de sus puertos serial va a tomar los datos enviados por el módulo de información, de manera que sin haber comenzado la comunicación entre los módulos de control y el módulo administrador, pueda ir realizándose el registro y conteo de la información.

Para la puesta en marcha de la comunicación entre el módulo de control y el módulo administrador, comienza con el envío de un caracter "@" (en forma hexadecimal x"40") por parte del módulo administrador. Cuando se recibe dicho carácter inmediatamente es enviada la trama de datos (Salida) al módulo administrador, con la información previamente registrada. Si se requiere cancelar dicha comunicación el módulo administrador envía un caracter "?" (en forma hexadecimal x"3F"), al recibir dicho caracter se detiene el envío de información.

Cada 10 segundos aproximadamente los registros del control de inventario (Existencia) serán llevados a cero (0), y se realiza un conteo de la información que sea recibida a partir de dicho momento. En caso de no recibir información los registros permanecerán en cero (0), asignándose los mismos a su respectivo lugar en la trama de datos a enviar.

Si por alguna razón es necesario cancelar las comunicaciones y limpiar los registros de control de inventario se debe presionar el botón de Reset, que activa la señal de bloqueo (Reset2) para la escritura y el envío de la información registrada en la tarjeta de desarrollo; para su puesta de funcionamiento se requiere que se presione nuevamente el botón de Reset (desactivando de esta manera la señal de bloqueo), no se requiere que el administrador envíe el caracter de comienzo de la comunicación ya que se trató un evento no relacionado con el módulo administrador.

El esquema del bloque que cumple con las funciones del módulo de control, previamente explicado, se observa en la figura 4.21, y su respectiva simulación se representa en la figura 4.22 a la figura 4.25.

Figura 4. 21. Bloque del módulo de control.

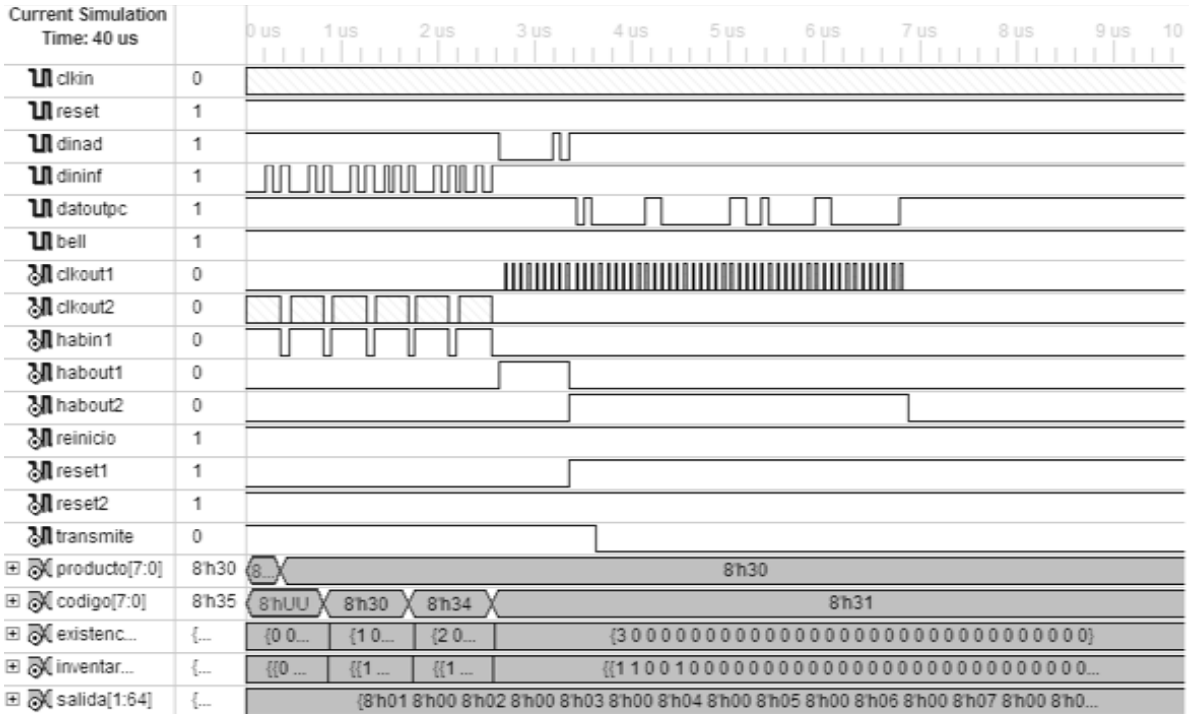


Figura 4. 22. Resultado de la simulación, entre 0 us y 10 us, del bloque del módulo de control.

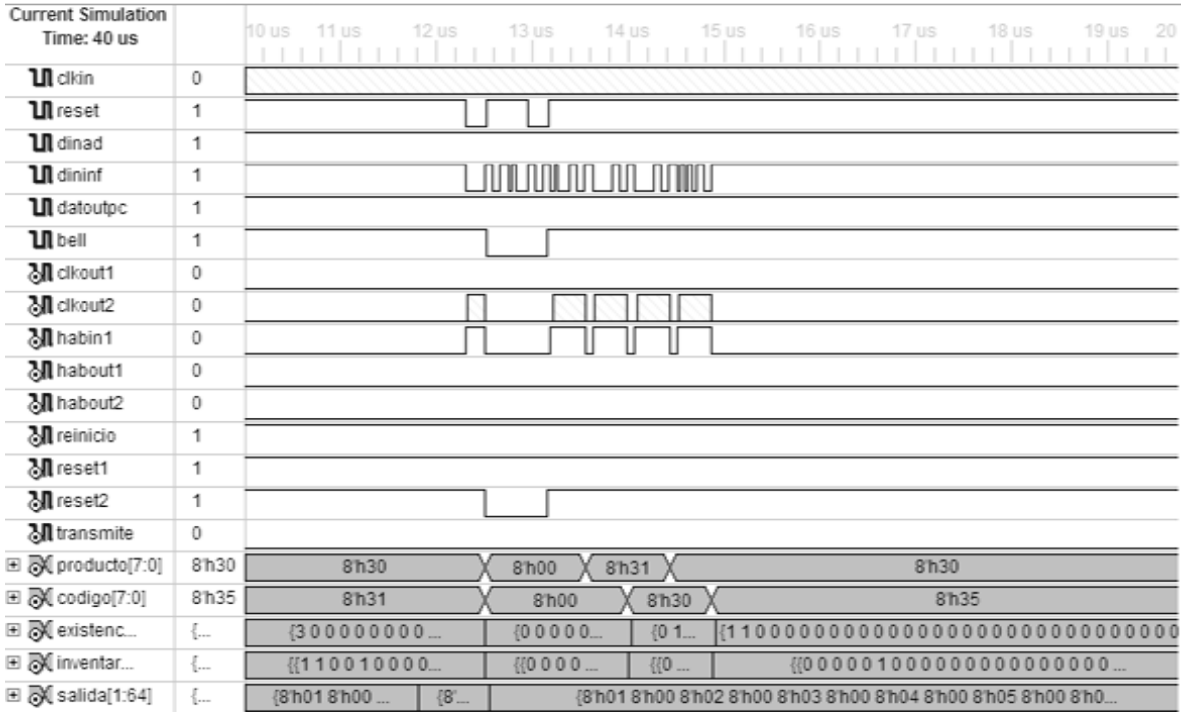


Figura 4. 23. Resultado de la simulación, entre 10 us y 20 us, del bloque del módulo de control.

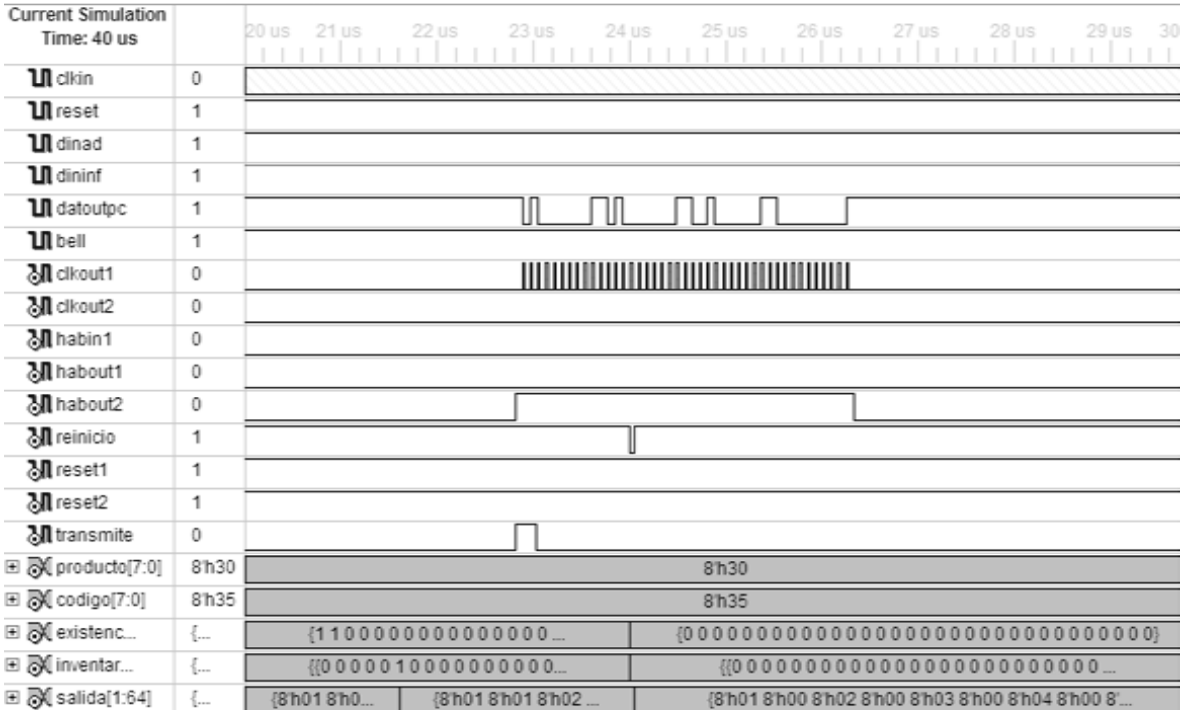


Figura 4. 24. Resultado de la simulación, entre 20 us y 30 us, del bloque del módulo de control.

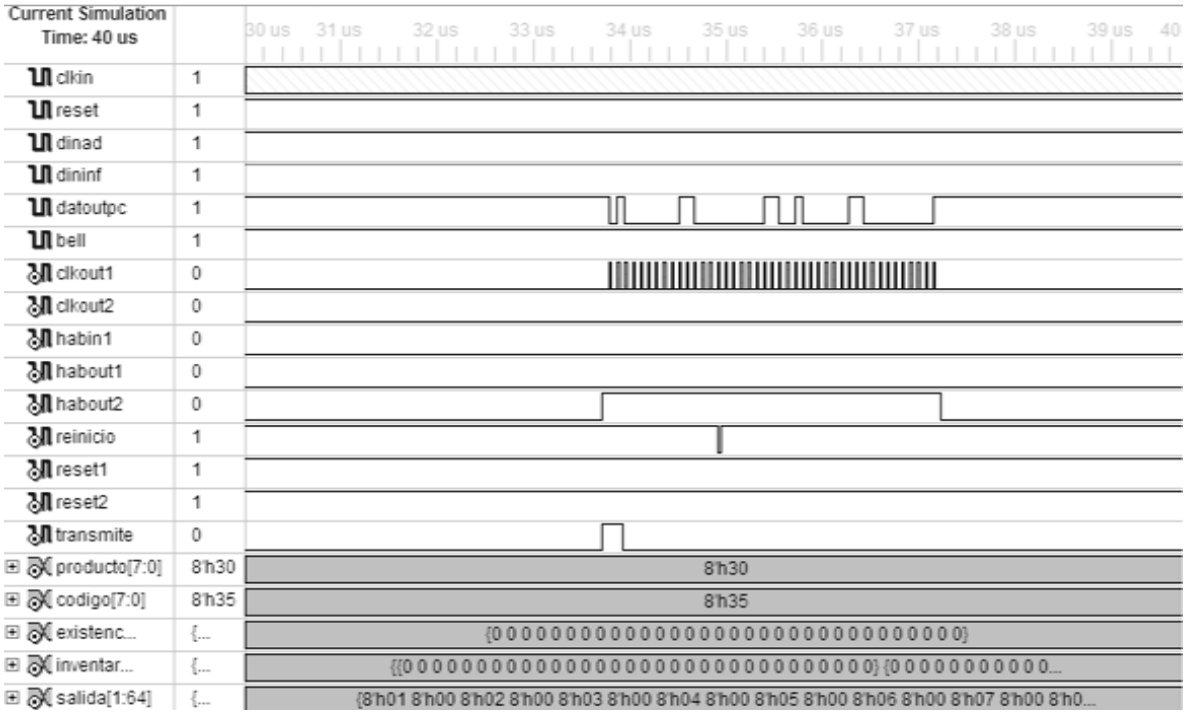


Figura 4. 25. Resultado de la simulación, entre 30 us y 40 us, del bloque del módulo de control.

4.5.1 IMPLEMENTACIÓN DE LA PROGRAMACIÓN DESARROLLADA EN LA TARJETA DE DESARROLLO.

En el momento de compilación y generación del archivo .bit que contiene la programación del módulo de control se colocó como cantidad máxima un número de 40 posibles productos y 40 ejemplares por cada uno de estos, debido a las limitaciones de la memoria de programa de la FPGA de la tarjeta de desarrollo seleccionada.

4.6 DESARROLLO DEL SOFTWARE PARA EL SISTEMA DE CONTROL DE INVENTARIO.

4.6.1 INTERFAZ GRÁFICA.

Se trata de un sistema computarizado para el control de inventarios, el cual se compone de una aplicación gráfica. Es una herramienta diseñada para el almacenamiento de los datos del inventario, la visualización y edición de los mismos por parte del usuario, así como también para la generación de reportes de inventario; todo esto para cumplir con los requerimientos del módulo administrador planteados en el capítulo III.

4.6.2 LENGUAJE DE PROGRAMACIÓN UTILIZADO.

Para la creación de esta interfaz gráfica se desarrolló la programación de un software, llamado SCI V1.0 (sistema de control de inventarios versión 1.0), utilizando el lenguaje de programación orientado a objetos Visual Basic 2010 Express en el entorno de desarrollo Visual Studio 2010. Siendo éste un lenguaje visual, permite desarrollar una interfaz gráfica más amigable al usuario, la cual por medio de iconos y otras herramientas visuales facilitan las tareas rutinarias del usuario; además y algo muy importante es que de la aplicación desarrollada con este lenguaje se obtuvo como resultado una interfaz gráfica muy similar a la del sistema operativo Windows, lo cual facilita el uso y la comprensión del modo de funcionar y manejar a la misma.

4.6.3 ESTRUCTURA DEL SOFTWARE SCI V1.0.

La administración y el control de un inventario deben realizarse de la manera más eficaz y sencilla posible. Debido a ello, se requiere que la interfaz del programa encargado del manejo de datos del inventario, sea muy sencilla y amigable para el usuario. En virtud de todo lo mencionado, la utilización de ventanas que despliegan la información solicitada, el uso de entradas de texto para búsqueda, la ejecución de comandos con botones y la aplicación de reportes para el usuario, es la mejor manera de presentar y manejar la aplicación que administra el sistema, todo esto anclado a un menú principal que facilitará la dirección y los pasos a ser tomados por la persona que esté manejando el sistema.

La aplicación está constituida principalmente por una ventana que contiene tres fichas o pestañas: pestaña Inicio, pestaña Inventarios y pestaña Editar Registros, mediante las cuales se pueden llevar a cabo acciones como:

- Conexión/desconexión del puerto COM utilizado en el sistema (puerto serial asignado a la FPGA).
- Envío de señal de control por puerto serial a la FPGA, para indicarle a ésta las acciones de conexión o desconexión del puerto COM.
- Consulta y visualización del inventario.
- Búsqueda de registros (productos) existentes en el inventario.
- Edición de información de los registros (productos) existentes en el inventario.
- Generación de reportes de inventario.
- Salir de la aplicación.

El diagrama de flujo de la figura 4.26 muestra el funcionamiento general del software desarrollado, mostrando todos los módulos, que en este caso se trata de las pestañas de la ventana principal, que componen al sistema o aplicación, así como también la manera en que se distribuyen y su jerarquía. Como puede observarse en dicho diagrama, el sistema inicia con la pestaña Inicio, la cual es un menú principal que contiene cuatro

opciones de las cuales se derivan las demás pestañas y funciones del sistema. Dichas opciones son las siguientes:

Inicio: permite abrir o cerrar el puerto de comunicación al que está conectada la tarjeta de desarrollo (FPGA), además de habilitar el acceso que permite la visualización de la pestaña Inventarios y de la pestaña Editar Registros.

Inventarios: Permite acceder a la pestaña Inventarios para hacer simples consultas a los distintos tipos de inventario que contiene el sistema.

Editar Registros: Permite acceder a la pestaña Editar Registros, la cual contiene las opciones que permiten editar los registros de producto del inventario.

Salir: Como su nombre lo indica, esta opción permite salir del programa es decir cerrar la aplicación, siempre y cuando el usuario esté seguro de dicha acción (ventana de confirmación).

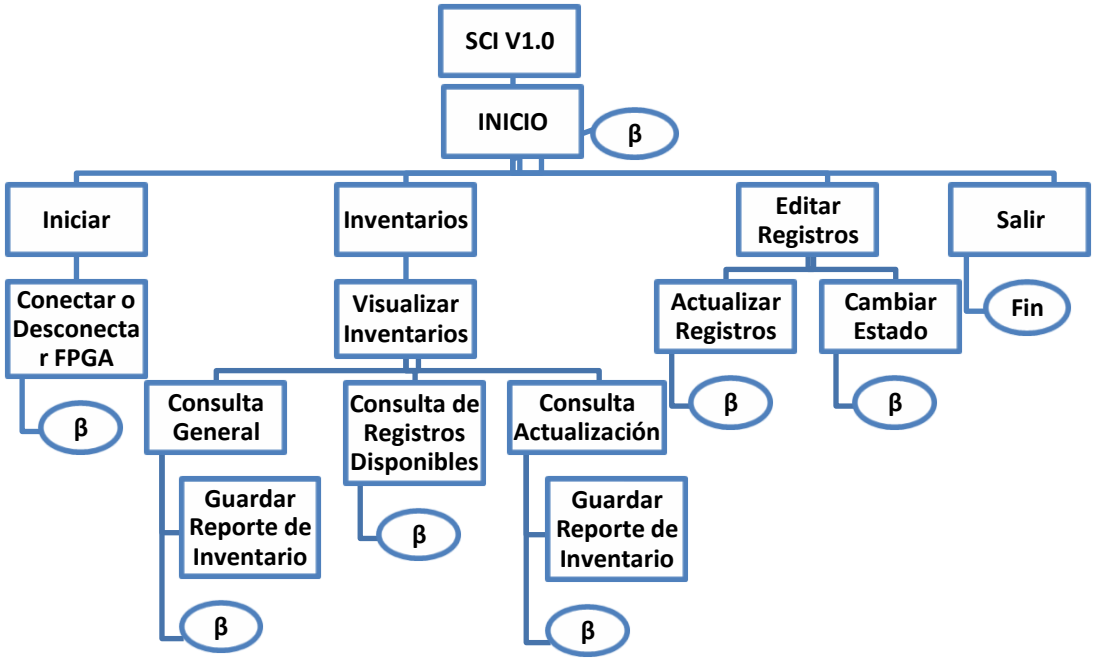


Figura 4. 26. Diagrama de flujo general de la aplicación.

La pestaña Inventario cuenta con tres opciones o sub-pestañas, donde cada una cuenta con una tabla de datos del inventario con ligeras diferencias entre ellas. Estas sub-pestañas son las siguientes:

Consulta General: Contiene una grilla o tabla (TABLA GENERAL) donde se almacenan los datos del inventario de productos, y permite al usuario visualizar y consultar el inventario.

Registros Disponibles: En ella se encuentra una grilla o tabla (TABLA DISPONIBLES) que almacena los datos de solo aquellos productos del inventario que se encuentren disponibles (Estado =Activo), y permite al usuario la visualización y consulta de este inventario.

Última Actualización: Presenta una grilla o tabla (TABLA ACTUALIZACIÓN) que permite almacenar y visualizar los datos acerca del último movimiento de todos los productos del inventario.

La pestaña Editar Registros, por su parte, consta de dos sub-pestañas, las cuales son:

Actualizar Registro: Sub-pestaña que permite al usuario modificar la información, en cuanto a la Descripción y Precio por unidad, de los registros de producto del inventario.

Cambiar Estado: Sub-pestaña que permite cambiar el estado de los productos, es decir establecer si un registro de producto del inventario está Activo o No activo.

4.6.3.1 ESTRUCTURA DE LAS TABLAS DE INVENTARIO.

Tabla General: En la tabla 4.2 se muestra la estructura de la tabla de inventario contenida en la sub-pestaña Consulta General.

Tabla 4. 2. Tabla General (sub-pestaña Consulta General).

Columna	Contenido	Posición	Formato
Código	Código del producto	1	Ninguno
Existencia	Cantidad actual del producto	2	Ninguno
Descripción	Nombre o breve descripción del producto	3	Ninguno
Precio por Unidad	Precio por unidad del producto	4	C2: moneda
Hora	Hora de última actualización de los datos de cantidad	5	Ninguno
Estado	Estado actual del producto	6	Ninguno

Tabla Disponibles: En la tabla 4.3 se observa la estructura de la tabla de inventario contenida en la sub-pestaña Consulta de Registros Disponibles.

Tabla 4. 3. Tabla Disponibles (sub-pestaña Consulta de Registros Disponibles).

Columna	Contenido	Posición	Formato
Código	Código del producto	1	Ninguno
Existencia	Cantidad actual del producto	2	Ninguno
Descripción	Nombre o breve descripción del producto	3	Ninguno
Precio por Unidad	Precio por unidad del producto	4	C2: moneda
Hora	Hora de última actualización de los datos de cantidad	5	Ninguno

Tabla Actualización: La tabla 4.2 muestra la estructura de la tabla de inventario contenida en la sub-pestaña Consulta de Último Movimiento.

Tabla 4. 4. Tabla Actualización (sub-pestaña Consulta de Último Movimiento).

Columna	Contenido	Posición	Formato
Código	Código del producto	1	Ninguno
Descripción	Nombre o breve descripción del producto	2	Ninguno
Cantidad Anterior	Cantidad del producto antes del último cambio en el producto	3	Ninguno
Cantidad Actualizada	Cantidad del producto luego de ocurrir el último cambio	4	Ninguno
Último Cambio	Fecha y Hora en la que se produjo un cambio en la cantidad del producto	5	Ninguno

4.6.4 DIAGRAMA DE FLUJO DE LOS PROCESOS CLAVES DEL SOFTWARE SCI V1.0.

Para el diseño del software (SCI V1.0) fue de gran ayuda la elaboración de diagramas de flujo, que representaran los procedimientos y estados del sistema.

4.6.4.1 CARGA DE DATOS EN TABLA DISPONIBLES (SUB-PESTAÑA CONSULTA DE REGISTROS DISPONIBLES).

Este proceso, a pesar de ser muy básico y sencillo, es clave para el desarrollo del software, debido a que éste se ejecuta cada vez que la Tabla General actualiza su información de los productos, ya que este proceso realiza un procedimiento para cargar en la Tabla Disponibles la información actualizada. Cada vez que se ejecuta el proceso éste limpia la tabla Disponibles (cero filas) y recorre las filas de la Tabla General para ir adicionando en la Tabla Disponibles solo la información de los registros de producto del inventario que estén activos. El diagrama de flujo mostrado en la figura 4.27 describe este proceso.

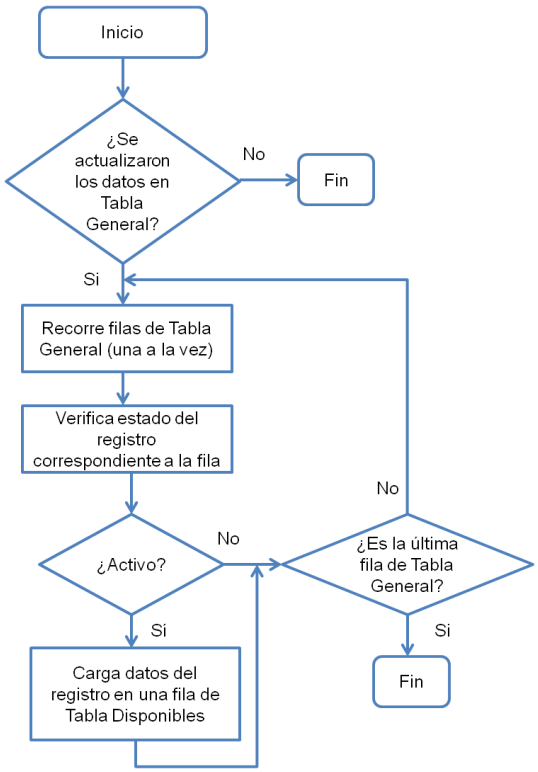


Figura 4. 27. Diagrama de flujo del proceso de carga de datos en Tabla Disponibles.

4.6.4.2 CARGA INICIAL DE DATOS EN LAS TABLAS DE INVENTARIO.

Este proceso se lleva a cabo cada vez que el usuario inicie la aplicación, en el momento en que la misma se carga para iniciar. El proceso, como puede apreciarse en el diagrama de la figura 4.28, carga en las tablas o grillas que tiene la aplicación (pestaña Inventarios), los datos e información de los registros de producto existentes en el inventario, siendo éstos los mismos que existían antes de cerrar la aplicación la última vez. Dichos datos se exportan de un documento de texto (*.txt) a las grillas. Si el usuario inicia la aplicación por primera vez, luego de instalada, el proceso carga en las grillas solo registros vacíos (filas vacías) que luego serán llenados con los respectivos datos recibidos por el puerto serial y los introducidos por el usuario; también crea un documento de texto vacío con el nombre "BaseDatos" en un directorio (oculto para que el usuario no tenga acceso a él y lo modifique por accidente) que también crea, llamado "BaseInventario", ubicado en el disco local "C" del ordenador.

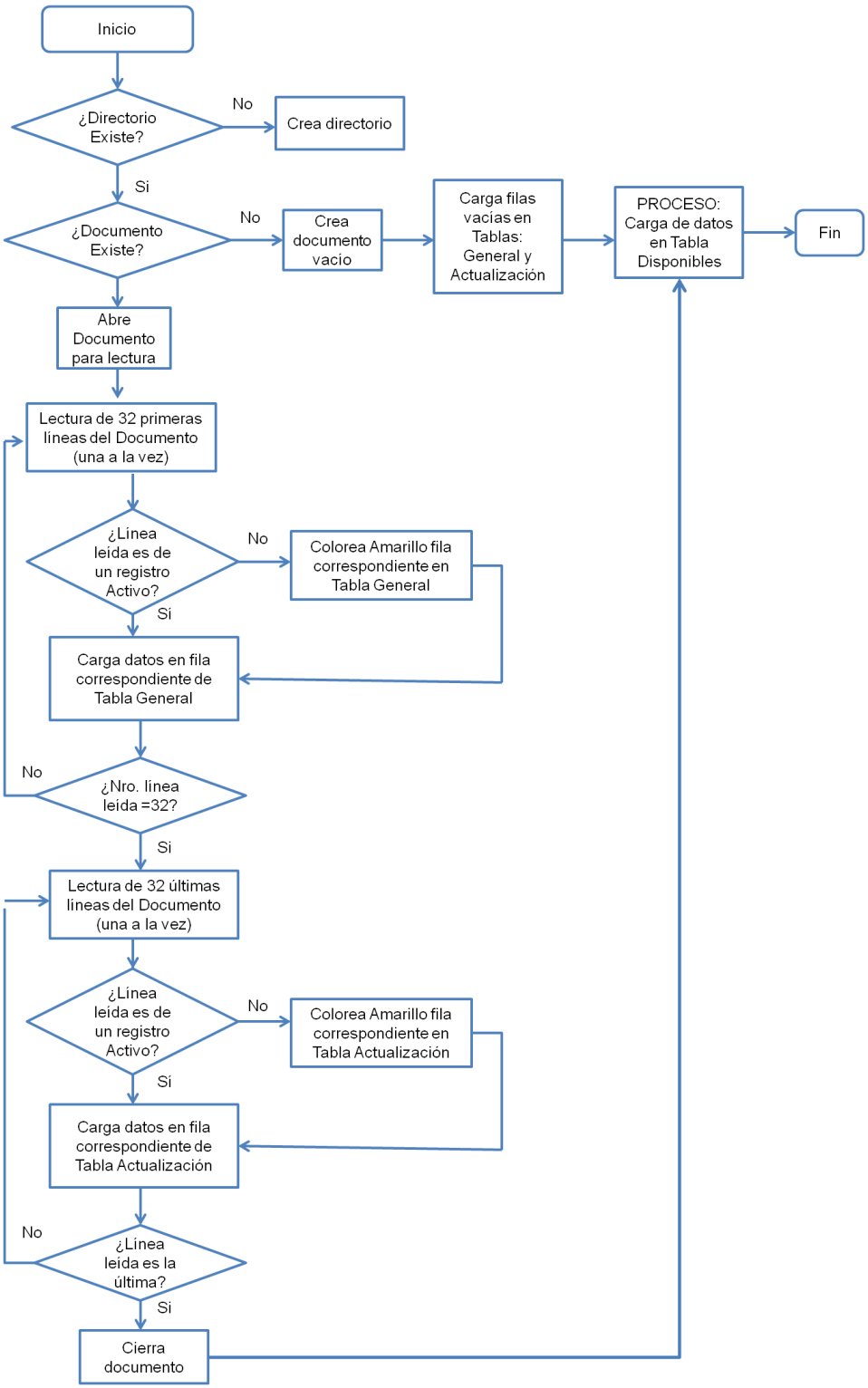


Figura 4. 28. Diagrama de flujo del proceso de carga inicial de datos en las tablas de inventario.

4.6.4.3 RESPALDO DE DATOS DE LAS TABLAS DE INVENTARIO.

Este proceso tiene como finalidad guardar en un documento de texto (*.txt) los datos contenidos en las Tablas General y Actualización al momento de cerrar o salir de la aplicación; para que luego, al iniciarse de nuevo la aplicación, se pueda llevar a cabo el proceso descrito anteriormente. Por lo tanto, el documento de texto utilizado en este proceso para guardar datos (escribir en documento) es el mismo utilizado en el proceso para cargar datos inícials (leer del documento). En la figura 4.29 se observa un diagrama de flujos que describe este proceso.

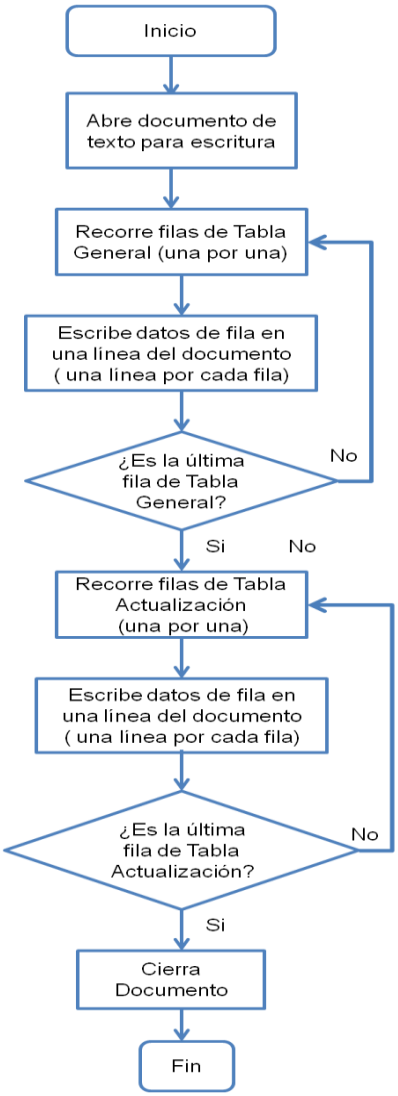


Figura 4. 29. Diagrama de flujo del proceso de respaldo de datos de las tablas inventario.

4.6.4.4 ALMACENAMIENTO DE DATOS RECIBIDOS POR EL PUERTO SERIAL.

Al ocurrir un evento de recepción de datos en el buffer de entrada del puerto configurado por la aplicación, se actualizarán, con los datos recibidos, las tablas de inventario existentes en la aplicación, es decir la Tabla General, la Tabla Disponibles y la Tabla Actualización. El proceso realiza una lectura de dos bytes (byte de código y byte de cantidad por tipo de producto) del buffer del puerto de comunicación serial, para luego cargar esos datos, junto con la hora del evento (emitida por el computador), en el registro o fila (de las distintas tablas de inventario existentes) correspondiente al número de código leído, tomándose en cuenta ciertas condiciones, como se puede apreciar en el diagrama de flujo de la figura 4.30. Este procedimiento se repite hasta terminar la lectura de todos los bytes recibidos.

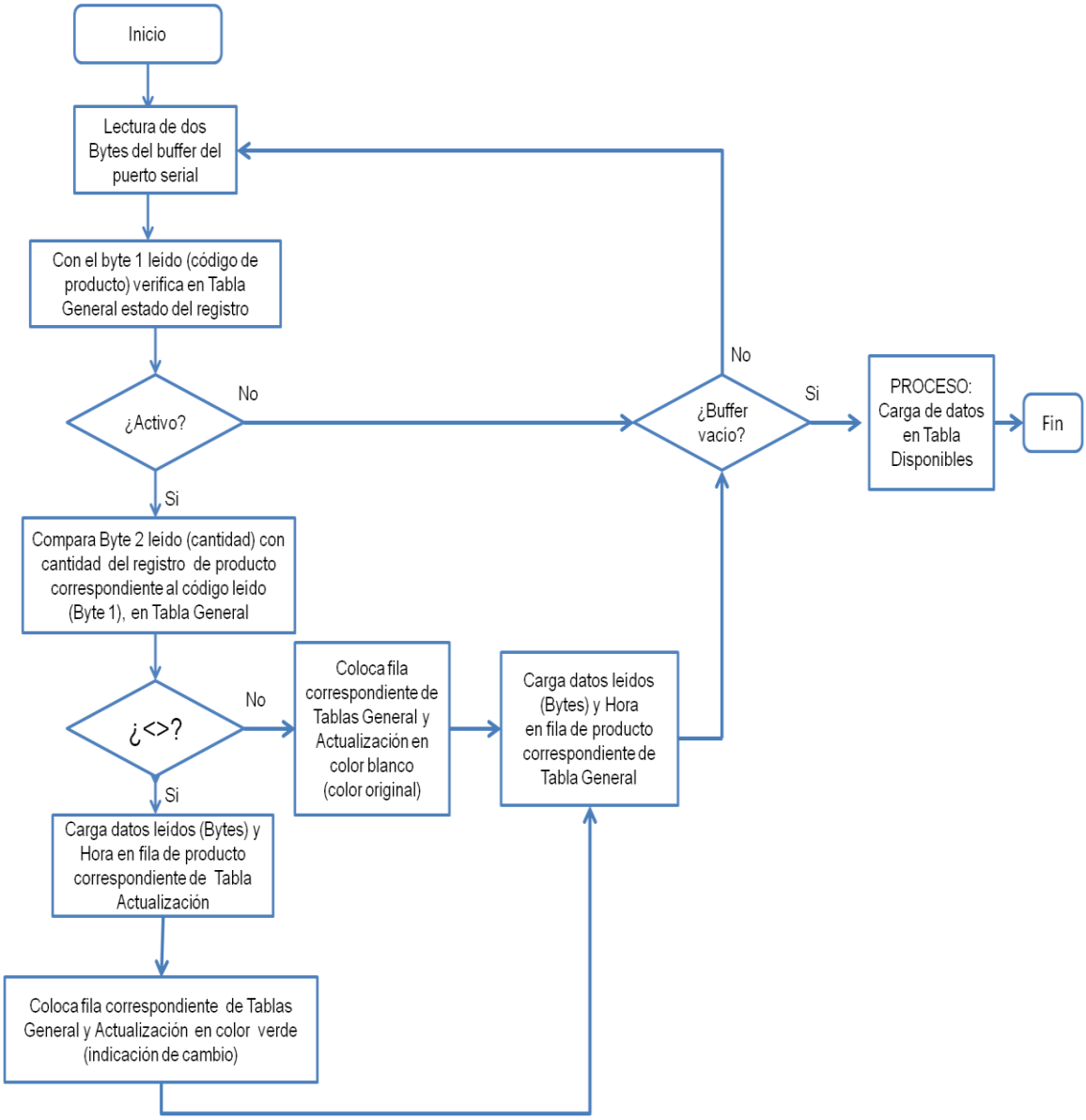


Figura 4. 30. Diagrama de flujo del proceso de almacenamiento de datos recibidos por el puerto serial.

4.6.4.5 BÚSQUEDA DE REGISTROS DE PRODUCTO EN TABLAS DE INVENTARIO.

Cada producto del inventario tiene su código registrado en el sistema, y este proceso consulta si existe, en la tabla de inventarios activa al momento de la búsqueda, el producto con el código buscado por el usuario. En la Tabla General se encuentra la opción de buscar

registros tanto por código como por descripción del producto, en cambio en la Tabla Disponibles y Tabla Actualización solo está la opción de búsqueda por código, pero el procedimiento es básicamente el mismo y se describe con más detalle en el diagrama de flujo de la figura 4.31.

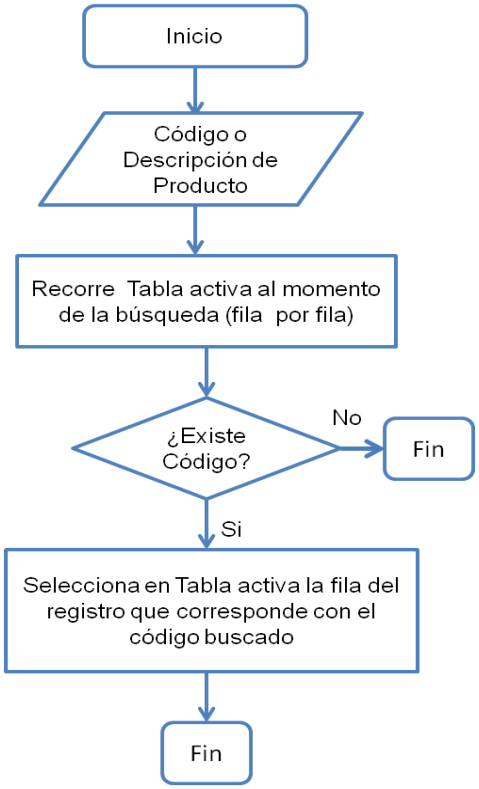


Figura 4. 31. Diagrama de flujo del proceso de búsqueda de registros.

4.6.4.6 GUARDAR REPORTE DEL INVENTARIO.

Este proceso se encarga de generar y guardar, en la ruta especificada por el usuario, un documento (Excel) que contiene los datos e información de todos los registros de producto del inventario. Esta opción de guardar está disponible para la Tabla General (sub-pestaña Consulta General) y para la Tabla Actualización (sub-pestaña Consulta de Último Movimiento). El diagrama de flujo que se muestra en la figura 4.32 describe este proceso.

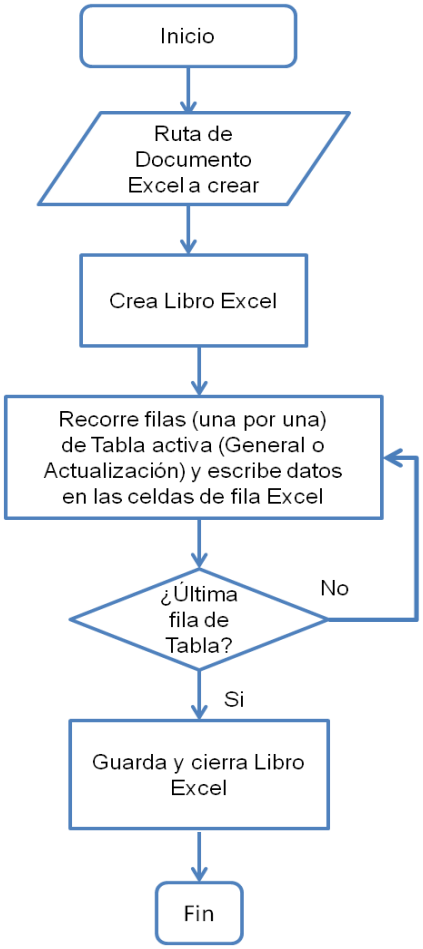


Figura 4. 32. Diagrama de flujo del proceso para guardar reporte de inventario.

4.6.4.7 ACTUALIZACIÓN DE REGISTROS.

Una vez que el usuario, en la sub-pestaña Actualizar Registros, para un código de producto ingresa una descripción y precio por unidad, dicha información es cargada en las tablas de inventario que posee la pestaña Inventarios, en los campos correspondientes (Descripción y Precio Unidad) del registro con el código de producto seleccionado. El diagrama de flujo de la figura 4.33 describe este proceso.

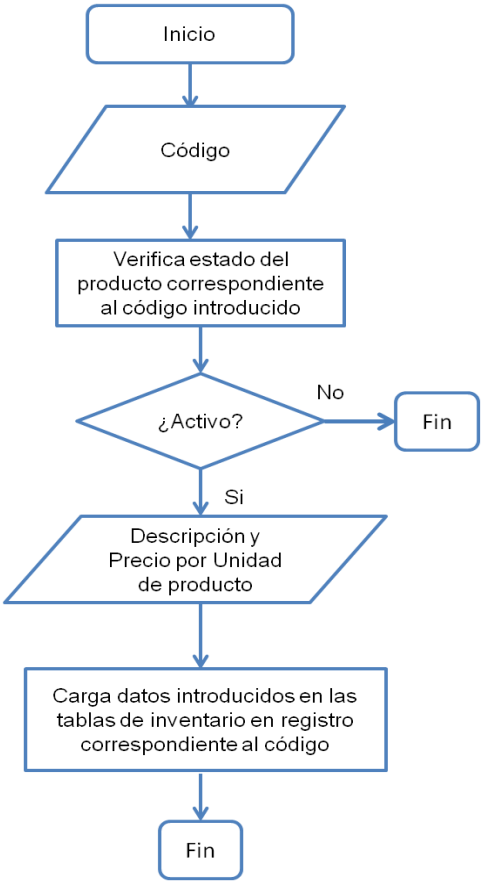


Figura 4. 33. Diagrama de flujo del proceso de actualización de registros.

4.6.4.8 CAMBIO DE ESTADO DE REGISTROS.

Una vez que el usuario, en la sub-pestaña Cambiar Estado, introduce un código de producto y le cambia su estado (Activo o No activo), dicha información es cargada en las tablas de inventario de la pestaña Inventarios, en el campo correspondiente (Estado) del registro con el código de producto introducido; además se colorea de amarillo (No activo) o blanco (Activo) la fila de dicho registro, y se elimina o agrega el registro a la Tabla Disponibles según sea el nuevo estado. El diagrama de flujo de la figura 4.34 describe con mayor detalle este proceso.

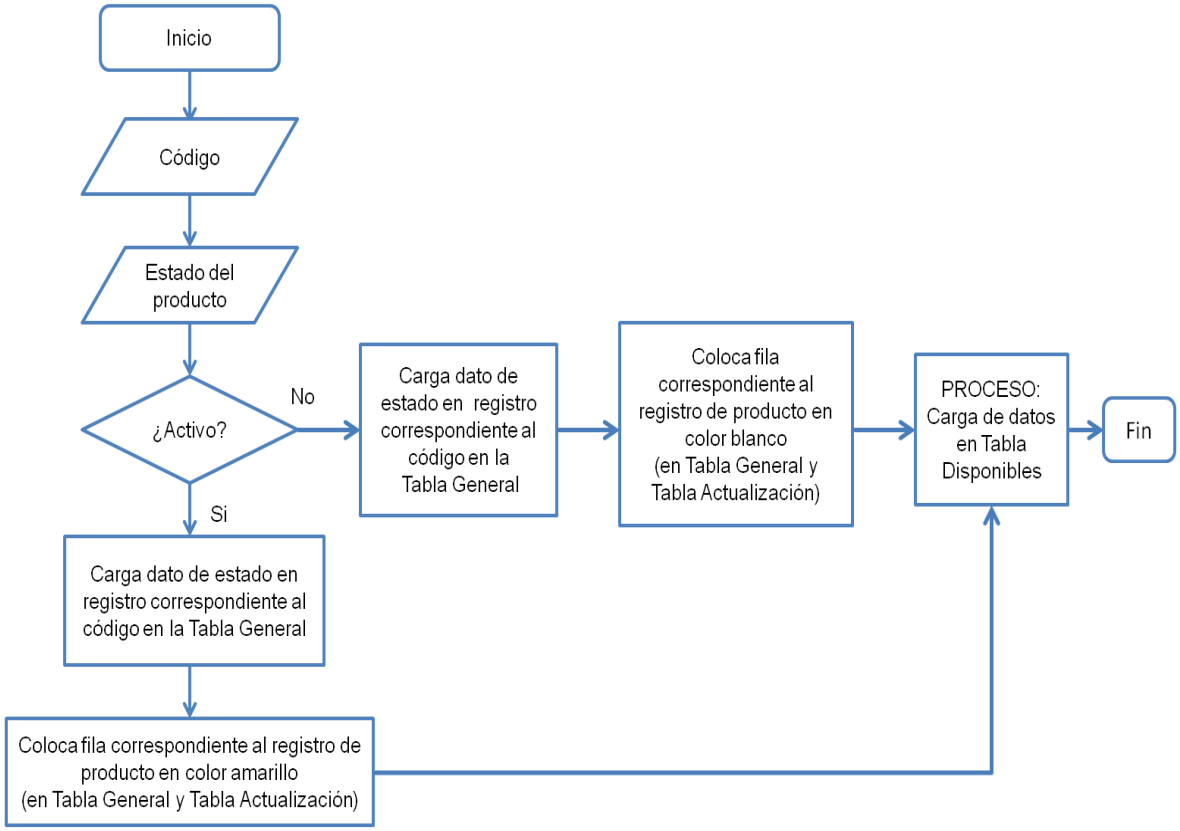


Figura 4. 34. Diagrama de flujo del proceso de cambio de estado de registros.

CAPÍTULO V.

MANEJO DEL SISTEMA DE CONTROL DE INVENTARIO.

5.1 OPERACIÓN DEL HARDWARE.

5.1.1 REQUISITOS DE HARDWARE.

Para la implementación del Sistema de Control de Inventario diseñado en este Trabajo de Grado es necesario tomar en cuenta los siguientes requisitos:

- Disponer de dos computadores, uno para implementar el módulo de información (simulación de lector RFID) y otro que funcione como interfaz de usuario (software SCI V1.0).
- Disponer de dos cables de datos serial DB9, ver figura 5.1.
- Adicionar un puerto de comunicación serial DB9 para compensar la falta del mismo en la tarjeta de desarrollo utilizada.

Figura 5. 1. Cable de datos serial DB9.

5.1.2 ENSAMBLAJE.

El ensamblaje del hardware del prototipo del sistema diseñado se basó en el esquema mostrado en la figura 5.2.

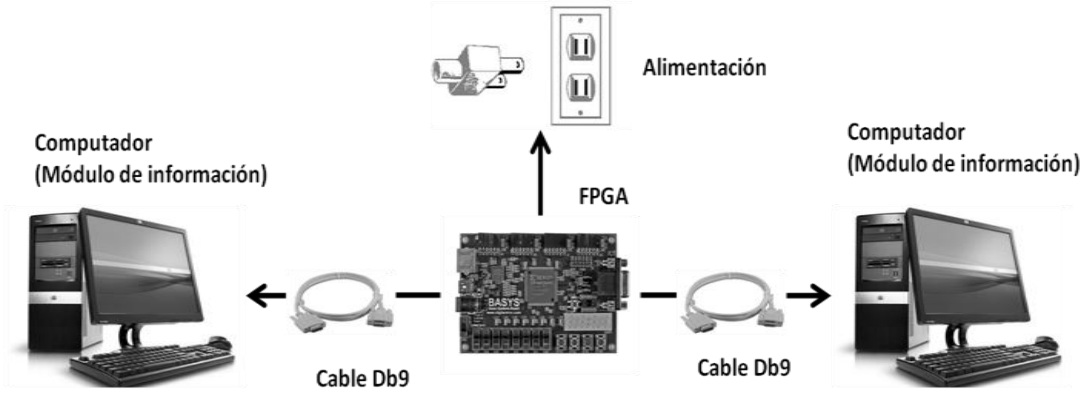


Figura 5. 2. Esquema de ensamblaje del prototipo del sistema diseñado.

En primer lugar se conectó la alimentación a la tarjeta de desarrollo (FPGA), ver figura 5.3. Luego se conectó uno de los extremos de un cable DB9 en uno de los puertos de comunicación serial que tiene la tarjeta de desarrollo, ver figura 5.3, y el otro extremo de este cable un puerto serial del computador del módulo de información; así mismo, uno de los extremos del otro cable DB9 fue conectado en el otro puerto serial que tiene la tarjeta de desarrollo, ver figura 5.3, y el otro extremo a el puerto serial del computador dispuesto para el módulo administrador.

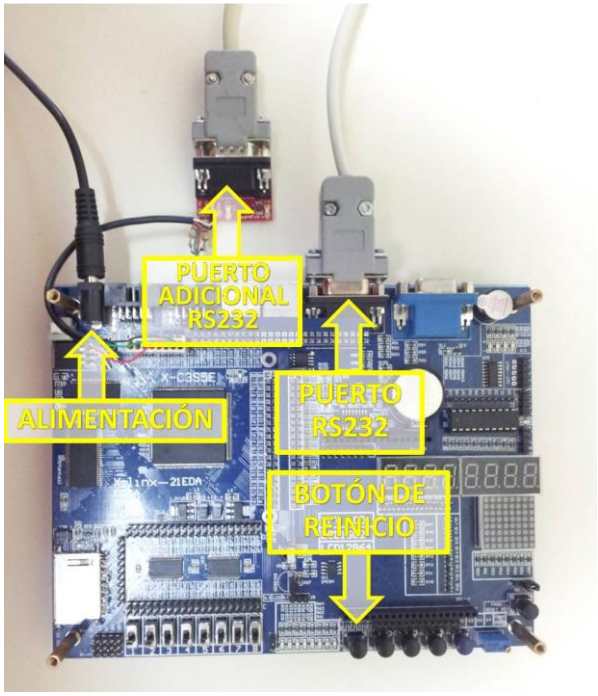


Figura 5. 3. Alimentación y conexiones de la tarjeta de desarrollo FPGA.

5.1.3 PUESTA EN MARCHA DEL SISTEMA.

Para la puesta en funcionamiento del Sistema de Control de Inventario fue necesario realizar las siguientes acciones:

- En primer lugar, iniciar aplicación diseñada (Software SCI V1.0) en el computador del módulo administrador.
- Encender la tarjeta de desarrollo FPGA con el botón que tiene destinado a para esta acción, el cual se muestra en la figura 5.3.
- Por último, iniciar la aplicación HyperTerminal, en el computador del módulo de información, para enviar los datos de cantidad de todos los productos del inventario a la FPGA (simulación del Lector RFID).

5.2 OPERACIÓN DEL SOFTWARE SCI V1.0.

En este apartado se procede a describir la estructura del software diseñado y las diferentes funciones que posee, proporcionando una especie de manual de usuario; adicionalmente contiene los requerimientos mínimos a nivel de sistema para que el software funcione correctamente en el computador y por último se desarrolla un manual de instalación.

5.2.1 REQUISITOS DE SOFTWARE.

Para el funcionamiento de la aplicación desarrollada es necesario, como se indicó anteriormente, disponer de un computador donde se instala el software que servirá de interfaz para la visualización y gestión de los datos del inventario. Los requisitos que debe cumplir este computador son:

- Tener instalado Windows 7 de 32 bits o en su defecto Windows XP.

- Contar con un puerto de comunicación serial exclusivamente asignado para la conexión con la tarjeta de desarrollo FPGA, la cual estará conectada durante toda la jornada diaria de trabajo ininterrumpidamente.
- El calendario y la hora de la computadora deberán estar configurados en los valores correctos de tiempo actual, ya que el inventario se actualizará en función de los datos de tiempo que le emita el computador de trabajo.

5.2.2 FUNCIONAMIENTO DEL SOFTWARE.

El código usado para la programación del software SCI V1.0 se puede consultar en el Apéndice B.

5.2.2.1 PANTALLA DE PRESENTACIÓN.

En la figura 5.4 se muestra la ventana que será visualizada en la pantalla del computador cuando arranca el programa, esta se muestra durante el tiempo que dura en cargar la barra de progreso; y durante este tiempo el software se encontrará cargando el resto del programa hasta dar inicio a la ventana principal del sistema.

Figura 5. 4. Pantalla de presentación del software.

5.2.2.2 PANTALLA PRINCIPAL.

El Sistema de Control de Inventario (SCI V1.0) presenta una pantalla o ventana principal, ver figura 5.5, en la cual se encuentran tres fichas o pestañas que permiten al usuario acceder a las diferentes funciones del programa. Al iniciarse esta ventana la pestaña Inicio será la que esté visible y en ella se aprecian dos paneles bien diferenciados; en el panel ubicado de el lado derecho se observa el nombre del software y una barra de estado ubicada en la parte inferior donde se visualiza la fecha y la hora emitida por el computador. En el panel ubicado a la izquierda de la ventana se visualiza un menú (menú principal), en el cual se encuentran cuatro botones, "Iniciar", "Inventarios", "Editar Registros" y "Salir", mediante los cuales el usuario puede acceder a las otras dos pestañas del sistema y llevar a cabo otras acciones del sistema que se explicarán a continuación.

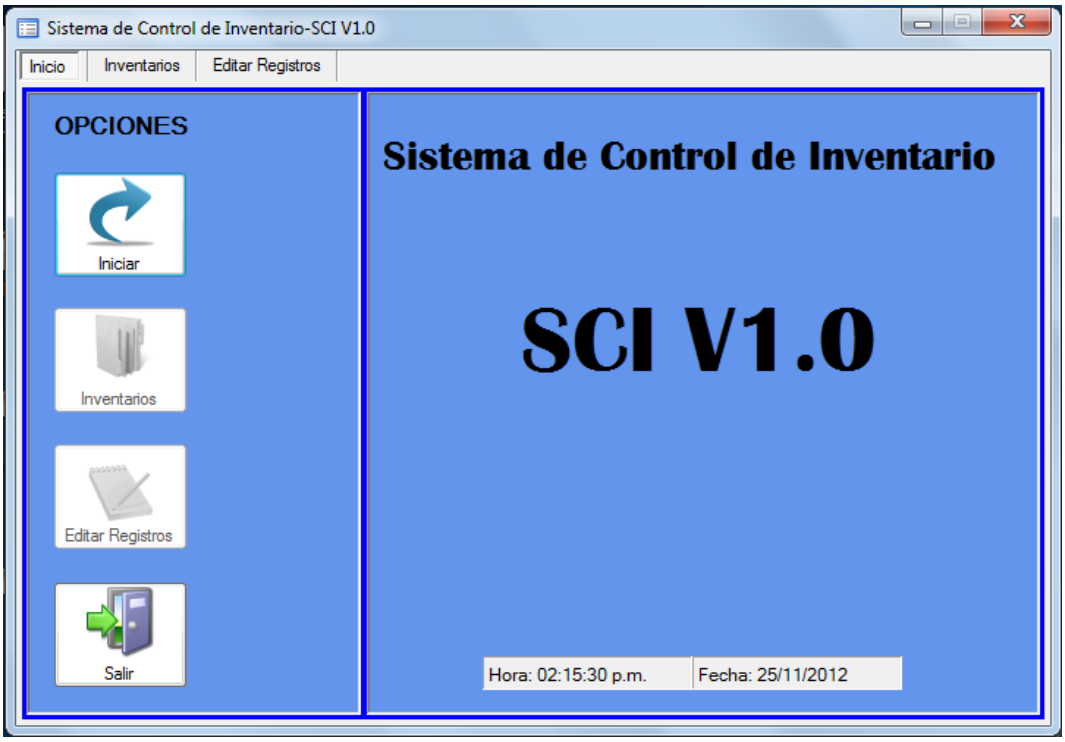


Figura 5. 5. Pantalla Principal del software.

Es importante destacar que los botones "Inventarios" y "Editar Registros" no se encuentran habilitados para su uso al cargarse la aplicación, al igual que las pestañas con el

mismo nombre. En el momento en que el usuario da un clic en el botón "Iniciar" éste automáticamente cambia su apariencia, pues su texto cambia a "Finalizar" y su icono también, además de que los botones "Inventarios" y "Editar Registros" quedan habilitados, ver figura 5.6. La función que tiene este botón es el de Abrir ("Iniciar") o Cerrar ("Finalizar") el puerto serial (COM) asignado en la aplicación para recibir los datos de la tarjeta de desarrollo FPGA; además con el evento clic de este botón se envían señales de control por el puerto serial a la FPGA, para que la misma se entere del momento en que se inicia o se finaliza la conexión con ella desde la aplicación.

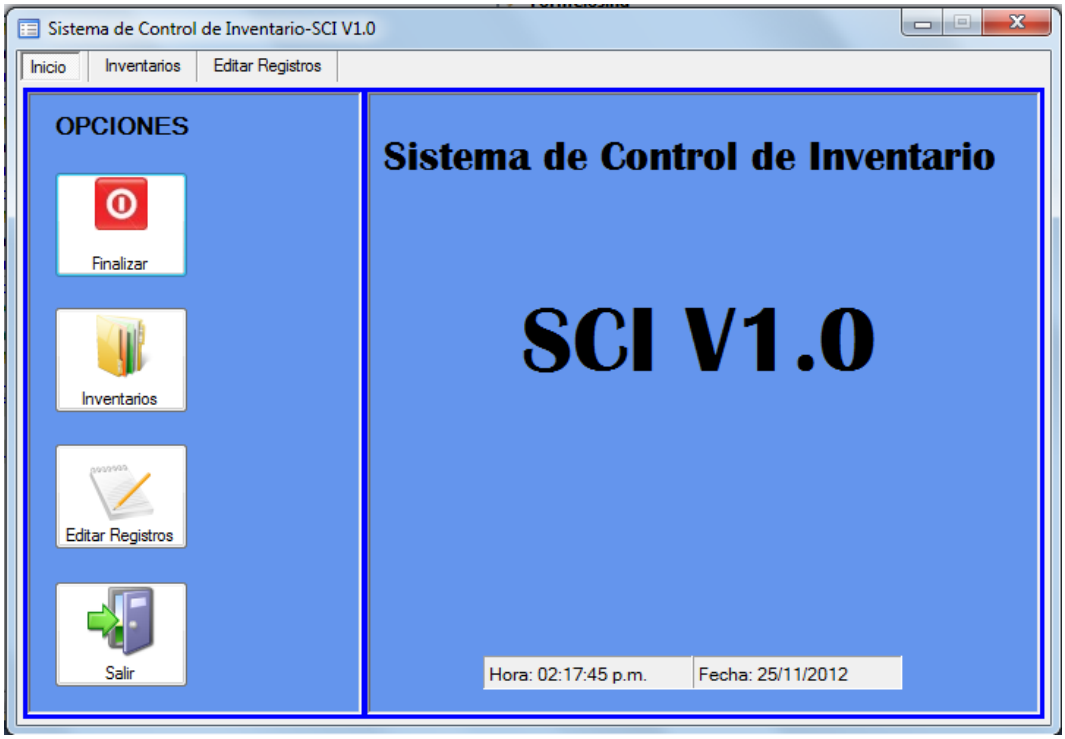


Figura 5. 6. Pantalla Principal luego de dar clic en botón "Iniciar".

Por otra parte, el botón "Salir" permite al usuario cerrar la aplicación de igual forma que si el usuario da un clic en la opción cerrar ubicada en el borde superior de la ventana principal. Al ocurrir el evento clic de este botón, automáticamente se muestra en pantalla la ventana de la figura 5.7, que no es más que una confirmación de salida para el usuario. Si el usuario elige la opción "Si" de esta ventana entonces la aplicación se cerrará y el proceso

"Respaldo de datos de las tablas de inventario" (ver ítem 4.5.4.3) se ejecutará, de lo contrario si el usuario elige la opción "No" la aplicación seguirá su curso normal.

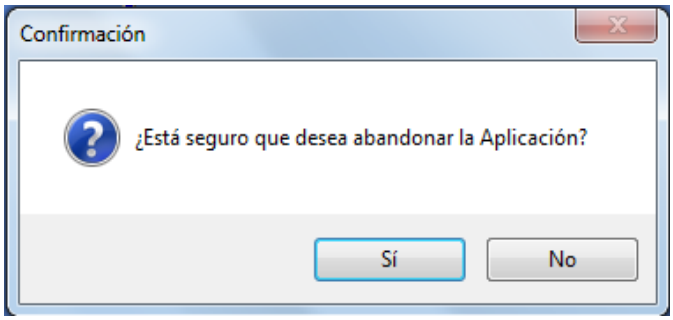
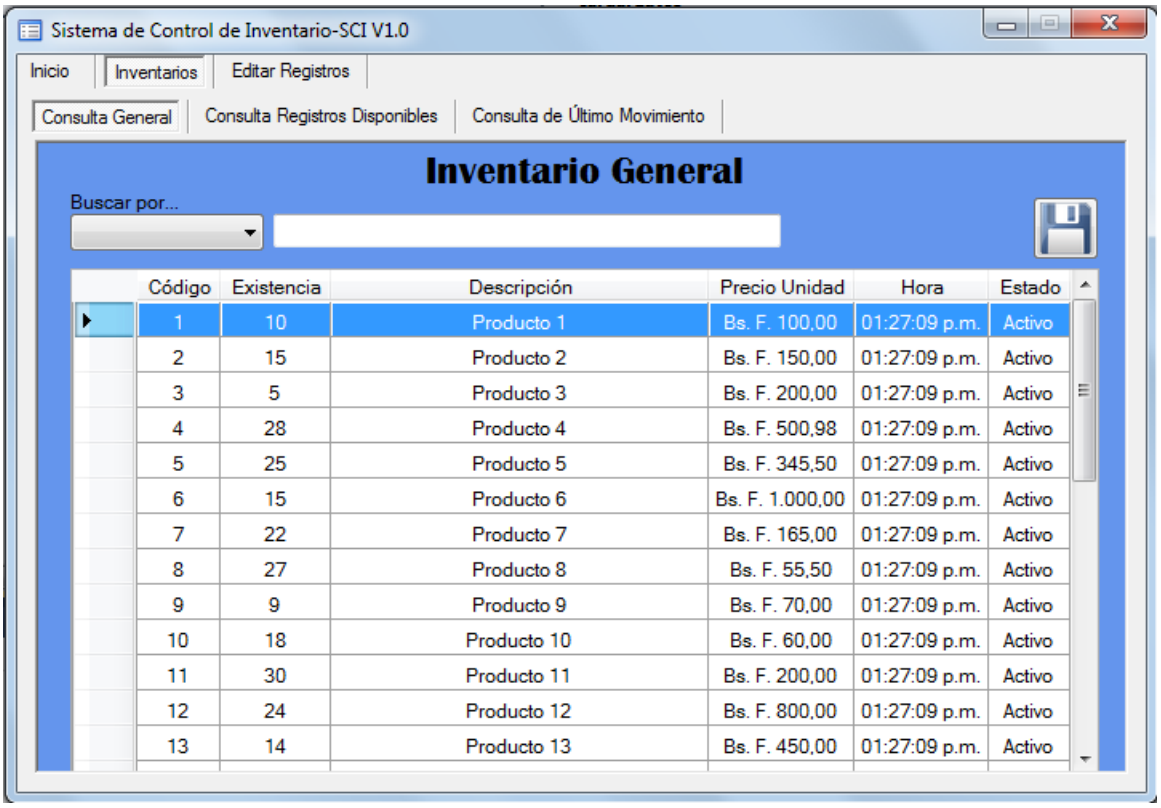


Figura 5. 7. Ventana de diálogo, "Conformación".

Por último, la ventana principal posee en el borde superior tres botones, el de maximizar, el de minimizar y el de cerrar, a diferencia de las otras ventanas del software las cuales únicamente tienen la opción de cerrarse. De las tres opciones mencionadas, la ventana principal solo tendrá habilitadas las opciones de minimizar pantalla y cerrar.

5.2.2.3 PESTAÑA INVENTARIOS.

Esta pestaña puede ser visualizada por el usuario pulsando el botón "Inventarios" de la pestaña Inicio o pulsando el encabezado de la pestaña en cuestión. En la figura 5.8 se observa que esta pestaña cuenta a su vez con tres opciones o sub-pestañas, como se indicó en el capítulo IV (Ver ítem 4.5.3), a las cuales el usuario puede acceder seleccionando con un clic el encabezado de cada una.



Código	Existencia	Descripción	Precio Unidad	Hora	Estado
1	10	Producto 1	Bs. F. 100,00	01:27:09 p.m.	Activo
2	15	Producto 2	Bs. F. 150,00	01:27:09 p.m.	Activo
3	5	Producto 3	Bs. F. 200,00	01:27:09 p.m.	Activo
4	28	Producto 4	Bs. F. 500,98	01:27:09 p.m.	Activo
5	25	Producto 5	Bs. F. 345,50	01:27:09 p.m.	Activo
6	15	Producto 6	Bs. F. 1.000,00	01:27:09 p.m.	Activo
7	22	Producto 7	Bs. F. 165,00	01:27:09 p.m.	Activo
8	27	Producto 8	Bs. F. 55,50	01:27:09 p.m.	Activo
9	9	Producto 9	Bs. F. 70,00	01:27:09 p.m.	Activo
10	18	Producto 10	Bs. F. 60,00	01:27:09 p.m.	Activo
11	30	Producto 11	Bs. F. 200,00	01:27:09 p.m.	Activo
12	24	Producto 12	Bs. F. 800,00	01:27:09 p.m.	Activo
13	14	Producto 13	Bs. F. 450,00	01:27:09 p.m.	Activo

Figura 5. 8. Ventana Principal, pestaña Inventarios.

5.2.2.3.1 SUB-PESTAÑA CONSULTA GENERAL.

Esta opción es la primera en mostrase cada vez que el usuario accede a la pestaña Inventarios. Esta sub-pestaña presenta una tabla o grilla que permite al usuario visualizar el estado actual de todos los productos que conforman el inventario de mercancía. La información o datos de cada producto se encuentran en cada fila de la tabla (Tabla General), conformando entonces cada fila un registro de producto. Como puede observarse en la figura 5.8, en la tabla de izquierda a derecha se visualiza Código (número del 1 al 32), Descripción, Existencia, Precio por Unidad, Hora y Estado de cada producto. Los datos de Código y Existencia de cada registro provienen de los datos recibidos por el puerto serial de la FPGA, y no pueden ser modificados por el usuario, en cambio los datos de los otros campos, que inicialmente se encuentran vacíos, son introducidos y modificados por el usuario.

Esta página cuenta con una opción que le permite al usuario buscar registros en el inventario, mediante el uso de la lista desplegable "Buscar por"; esta lista permite al usuario seleccionar la forma en que se desee realizar la búsqueda de registros, ya sea por el código o por la descripción del producto. Para que se pueda hacer uso de esta opción primero el usuario debe escribir en el cuadro de texto ubicado en al lado derecho de la lista desplegable, ver figura 5.8, el código o nombre según sea la forma de búsqueda deseada y luego elegir en la lista desplegable, con un clic, dicha forma de búsqueda. Una vez que se da el clic pueden suceder varios acontecimientos:

Si el usuario elige la opción de buscar por "Código" y el dato introducido en el cuadro de texto no es un valor numérico, el programa despliega en pantalla una ventana de error como la que se muestra en la figura 5.9.

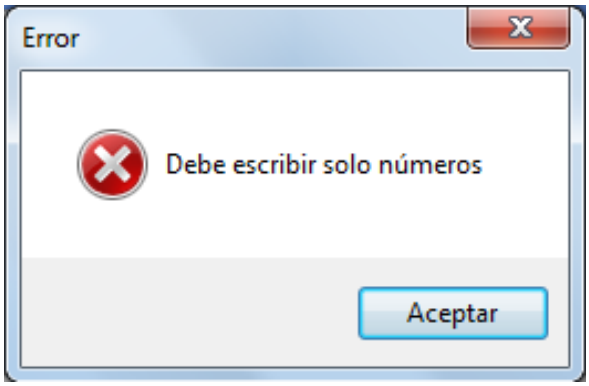


Figura 5. 9.Ventana de diálogo, "Error".

En cambio si los datos introducidos por el usuario son concordantes con la forma de búsqueda elegida, el programa ejecutará el proceso "Búsqueda de registros de producto en tablas de inventario" (ver ítem 4.5.4.5) y queda seleccionada la fila del registro buscado; pero si el registro no existe en el inventario el programa mostrará en pantalla una ventana informativa que lo indica, como se muestra en la figura 5.10 y en la figura 5.11.

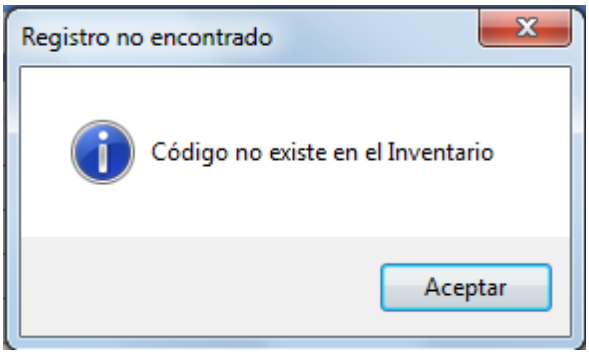


Figura 5. 10.Ventana de diálogo, "Registro no encontrado"-Buscar por: Código.

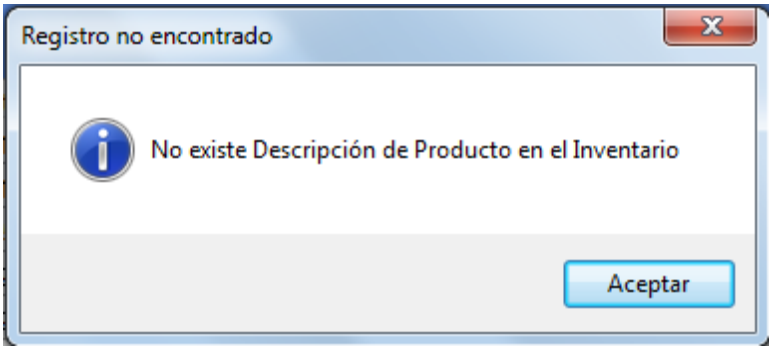


Figura 5. 11. Ventana de diálogo, "Registro no encontrado"-Buscar por: Descripción.

Si se da el caso en el que el usuario elige alguna opción de la lista desplegable "Buscar por" sin haber antes introducido algún dato en el cuadro de texto, el programa desplegará en pantalla una ventana de advertencia como la que se muestra en la figura 5.12 y en la figura 5.13, y por tanto el programa no realizará ningún proceso de búsqueda.

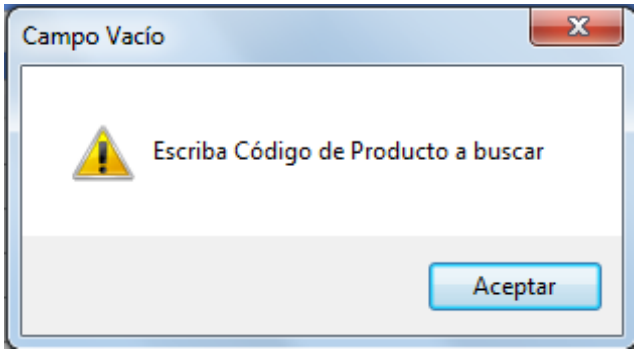


Figura 5. 12. Ventana de diálogo, "Operación No Válida"-Buscar por: Código.

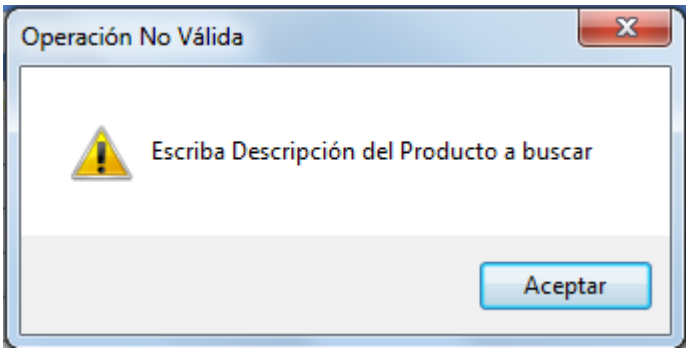


Figura 5. 13. Ventana de diálogo, "Operación No Válida"-Buscar por: Descripción.

Esta página o sub-pestaña también cuenta con la opción de Guardar, representada por el botón con el icono de diskette que se encuentra en el borde superior derecho de la tabla, ver figura 5.8; este botón permite guardar en el computador un reporte del inventario de la Tabla General en un documento tipo Excel. Al hacer clic en este botón el programa inmediatamente despliega la ventana de diálogo "Guardar como" que se muestra en la figura 5.14, en la cual el usuario elige e introduce la ruta tipo Excel (*.xlsx) donde desee guardar el reporte.

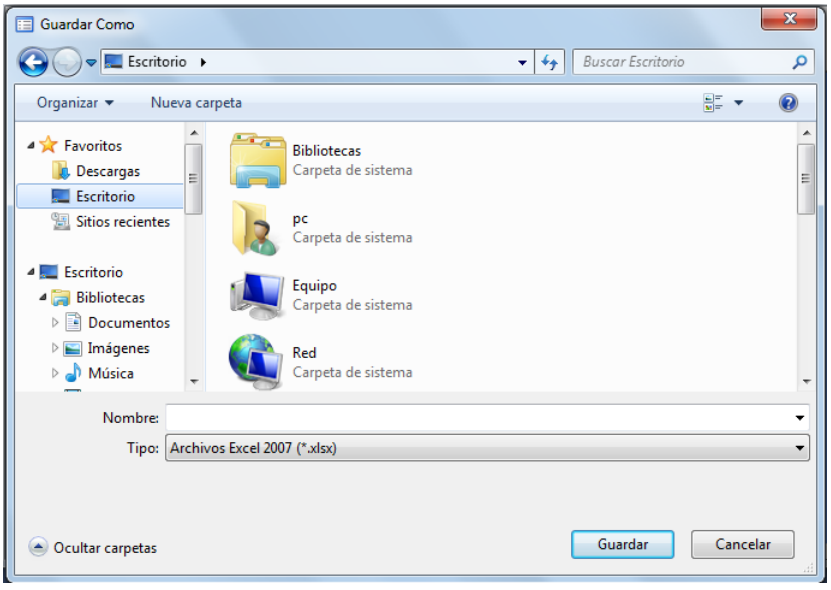


Figura 5. 14. Ventana de diálogo, "Guardar Como".

Cuando el usuario da un clic en el botón guardar de esta ventana, la misma se cierra y el programa ejecuta el proceso "Guardar reporte del inventario" (ver ítem 4.5.4.6).Pero si elige la opción "Cancelar" o la opción cerrar (borde superior derecho de la ventana), el programa no guardará reporte del inventario y emitirá un mensaje de alerta al usuario, mediante el despliegue en pantalla de la ventana Aviso que se muestra en la figura 5.15.

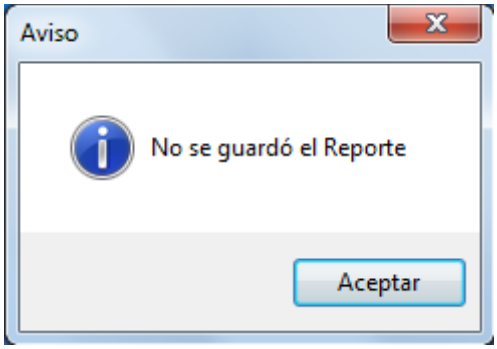


Figura 5. 15. Ventana de diálogo, "Aviso".

5.2.2.3.2 SUB-PESTAÑA CONSULTA DE REGISTROS DISPONIBLES.

En esta página el usuario puede visualizar y consultar una tabla (Tabla Disponibles) en donde se muestran los datos de todos los productos del inventario que se encuentren en estado Activo, ver figura 5.16 (Registro 3 no aparece en esta tabla, su estado es No activo). La información de cada producto activo del inventario estará en cada fila de la tabla, dispuesta según la estructura de la Tabla Disponibles (ver ítem 4.5.3.1).

Figura 5. 16. Ventana Principal, Sub-pestaña Consulta de Registros Disponibles.

Esta sub-pestaña también ofrece al usuario la opción de buscar registros mediante el botón "Buscar" que se encuentra sobre la tabla, como puede observarse en la figura 5.16; una vez que el usuario da clic en este botón el programa despliega en la pantalla una ventana para introducir el código del registro a buscar, ver figura 5.17.

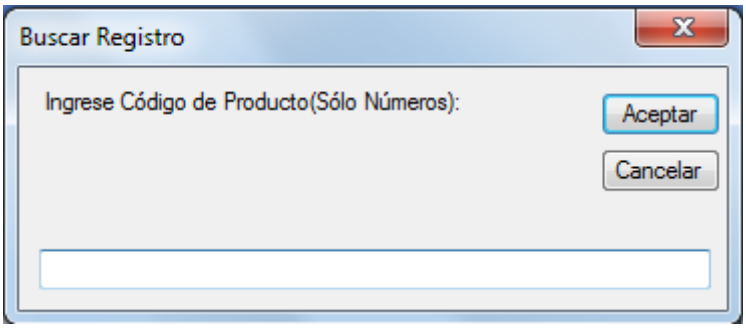


Figura 5. 17. Ventana para introducir datos, "Buscar Registro".

En el cuadro de texto de esta ventana solo pueden introducirse valores numéricos, si es así, y el usuario da clic en el botón "Aceptar", el programa ejecuta la búsqueda en la tabla (ver ítem 4.5.4.5) y si el registro existe selecciona la fila correspondiente al mismo, de lo contrario en pantalla aparece una ventana informativa como la mostrada en la figura 5.18. Pero si el usuario da clic en "Aceptar" y el valor introducido en la ventana no es un número, la ventana vuelve a aparecer. Esta ventana también tiene la opción de cancelar para cerrarla.

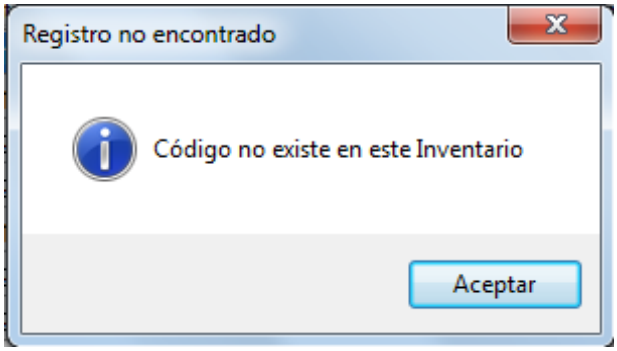


Figura 5. 18. Ventana de diálogo, "Registros No Encontrado".

5.2.2.3.3 SUB-PESTAÑA CONSULTA DE ÚLTIMO MOVIMIENTO.

Esta página le ofrece al usuario la opción de visualizar y consultar los datos acerca del último cambio en la cantidad de todos los productos del inventario. Como puede apreciarse en la figura 5.19, esta página contiene una tabla (Tabla Actualización) en la que se muestran los datos de la cantidad anterior y actualizada de cada producto y la fecha y hora del momento en que ocurrió el evento de cambio, dispuestos según la estructura de la Tabla Actualización (ver ítem 4.5.3.1).Por tanto, esta página permite al usuario monitorear los eventos de entrada y salida de productos del almacén donde se implemente el sistema desarrollado.

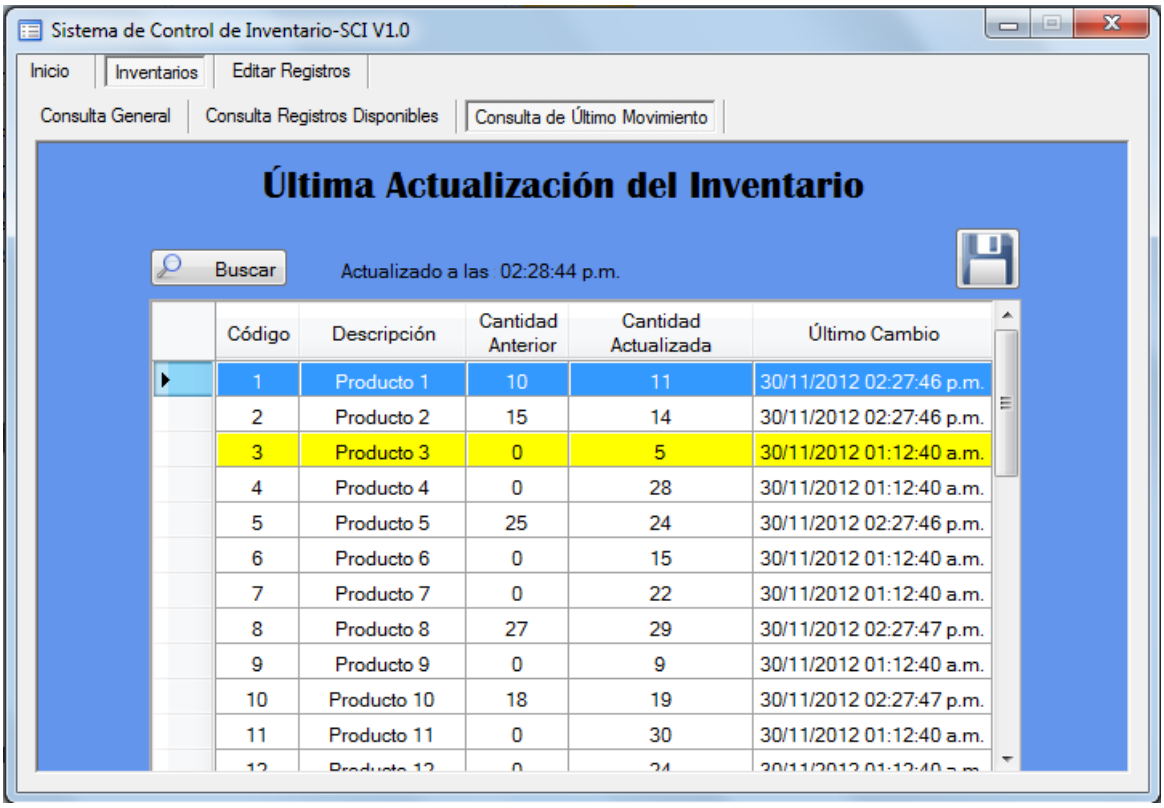


Figura 5. 19. Ventana Principal, Sub-pestaña Consulta de Último Movimiento.

Al igual que las dos sub-pestañas anteriores, esta cuenta con botones como "Buscar" y "Guardar" (botón con el icono del diskette), como puede observarse en la figura 5.19. Las funciones de estos botones y los procedimientos que el programa desarrolla con el uso de ellos por parte del usuario, son exactamente iguales a los explicados anteriormente en el apartado 5.2.2.3.2, pero usando la Tabla Actualización.

5.2.2.4 PESTAÑA EDITAR REGISTROS.

Esta pestaña puede ser visualizada por el usuario haciendo clic en el botón Editar Registros del menú principal (pestaña Inicial) o bien haciendo clic en el encabezado de esta pestaña; está compuesta a su vez por dos sub-pestañas: Actualizar Registro y Cambiar Estado, como puede observarse en la figura 5.20. Esta es la pestaña que permite al usuario actualizar los ítems de Descripción y Precio por Unidad de cada uno de los registros de producto del inventario, además de poder definir y cambiar el estado de cada uno.

Figura 5. 20. Ventana Principal, Pestaña Editar Registros.

5.2.2.4.1 SUB-PESTAÑA ACTUALIZAR REGISTRO.

Esta página es la primera en mostrase cada vez que el usuario accede a la pestaña Editar Registros. Esta sub-pestaña permite introducir y modificar los datos de Descripción y Precio por Unidad de cada uno de los registros de producto del inventario. Como se observa en la figura 5.20, la sub-pestaña cuenta con una lista desplegable "Seleccione Código de Producto" que se encuentra siempre habilitada, y un panel que se encuentra inicialmente inhabilitado, el cual contiene dos cajas de texto (para Descripción y Precio por Unidad), una etiqueta (para Cantidad actual) y un botón "Actualizar".

Mediante la lista desplegable el usuario selecciona el código del registro de producto que desea actualizar haciendo clic en ella y luego clic en el número de código, en ese momento los controles del panel se habilitan y en la etiqueta y los cuadros de texto respectivos aparece la información de cantidad, descripción y precio por unidad

correspondiente al producto del código seleccionado, ver figura 5.21; el usuario modifica los ítems de Descripción y Precio por Unidad, o los introduce por primera vez, mediante escritura por teclado y a continuación debe dar clic en el botón "Actualizar". Con este clic el programa realiza el proceso de "Actualización de registros" (ver ítem 4.5.4.7), cargando los datos en los registros correspondientes de las tablas de inventario y quedando de nuevo inhabilitados los controles del panel.

Figura 5. 21. Ventana Principal, Sub-pestaña Actualizar Registro.

Si se da el caso en el cual alguno de los cuadros de texto se encuentra vacío (no introdujo datos) y el usuario da clic en el botón "Actualizar", el programa muestra en pantalla una ventana de advertencia como la que se observa en la figura 5.22, indicando al usuario que debe llenar los campos vacíos para poder realizar la acción de actualizar el registro.

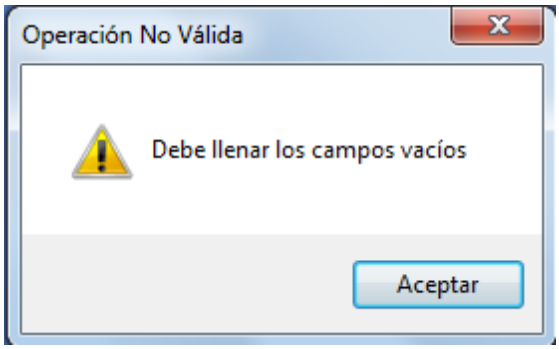


Figura 5. 22. Ventana de diálogo, "Operación No Válida".

Es importante destacar que si el usuario selecciona en la lista desplegable el código de un producto que se encuentre en estado No activo, el programa mostrará en pantalla una ventana de advertencia como la que se muestra en la figura 5.23, indicando que el registro no puede actualizarse debido a su estado, y los controles del panel quedan igualmente inhabilitados, hasta que el usuario seleccione otro código.

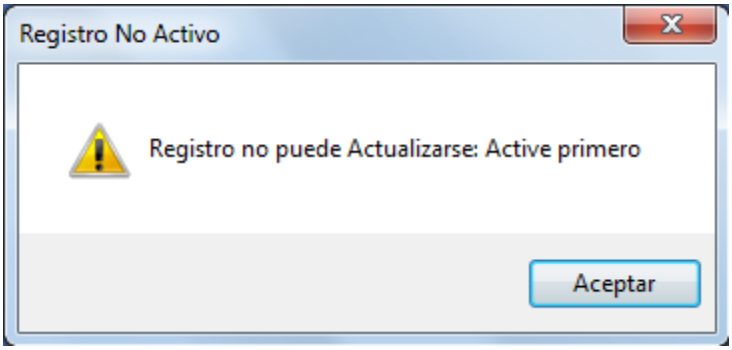


Figura 5. 23. Ventana de diálogo, "Registro No Activo".

5.2.2.4.2 SUB-PESTAÑA CAMBIAR ESTADO.

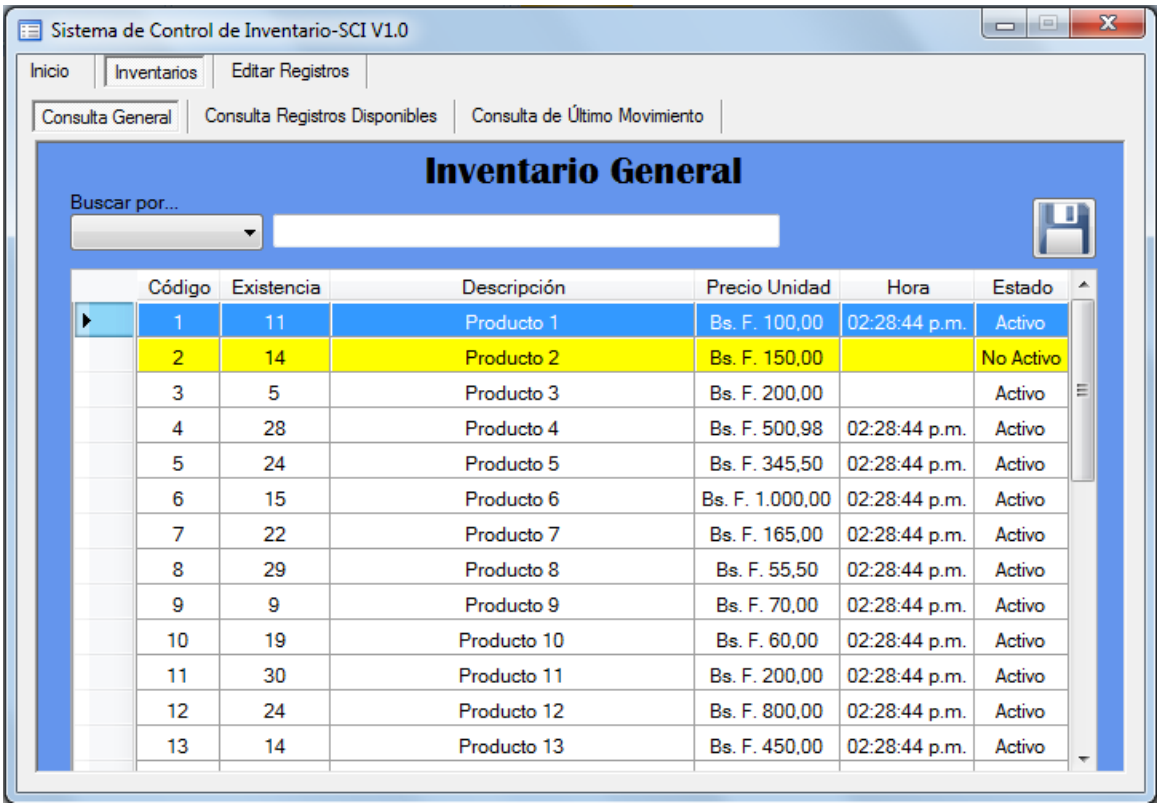
El usuario puede acceder a esta página o sub-pestaña seleccionando con un clic el encabezado de la misma; en ella el usuario puede modificar el estado de un producto, es decir estado Activo o No activo. Esta página cuenta con un panel que tiene en su interior una lista desplegable "Código del Producto" y otro panel, el cual contiene cuadros donde se muestran datos de Descripción y Estado, dos opciones mutuamente excluyentes "Activo" y "No activo", un botón "Aplicar" y un botón "Cancelar", ver figura 5.24.

Figura 5. 24. Ventana Principal, Sub-pestaña Cambiar Estado.

Para cambiar el estado de un producto el usuario debe selecciona un código de producto haciendo clic en la lista desplegable y luego clic en el número de código, en ese momento los controles del panel más interno se habilitan, mostrándose los datos de Descripción y Estado del registro seleccionado en los cuadros correspondientes, pero solo se habilita la opción de estado contraria a la que tenga el registro de producto seleccionado, como se muestra en la figura 5.25.

Figura 5. 25. Ventana Principal, Sub-pestaña Cambiar Estado.

El usuario puede cambiar el estado haciendo clic en la opción habilitada de estado y luego haciendo clic en botón "Aplicar"; el programa entonces realiza el proceso "Cambio de estado de registros" (ver ítem 4.5.4.8), quedando la modificación realizada reflejada en las tablas de inventario, ver ejemplo de figura 5.26, y los controles del panel más interno quedan inhabilitados nuevamente.



Código	Existencia	Descripción	Precio Unidad	Hora	Estado
1	11	Producto 1	Bs. F. 100,00	02:28:44 p.m.	Activo
2	14	Producto 2	Bs. F. 150,00		No Activo
3	5	Producto 3	Bs. F. 200,00		Activo
4	28	Producto 4	Bs. F. 500,98	02:28:44 p.m.	Activo
5	24	Producto 5	Bs. F. 345,50	02:28:44 p.m.	Activo
6	15	Producto 6	Bs. F. 1.000,00	02:28:44 p.m.	Activo
7	22	Producto 7	Bs. F. 165,00	02:28:44 p.m.	Activo
8	29	Producto 8	Bs. F. 55,50	02:28:44 p.m.	Activo
9	9	Producto 9	Bs. F. 70,00	02:28:44 p.m.	Activo
10	19	Producto 10	Bs. F. 60,00	02:28:44 p.m.	Activo
11	30	Producto 11	Bs. F. 200,00	02:28:44 p.m.	Activo
12	24	Producto 12	Bs. F. 800,00	02:28:44 p.m.	Activo
13	14	Producto 13	Bs. F. 450,00	02:28:44 p.m.	Activo

Figura 5. 26. Ventana Principal, Cambios que sufre Tabla General al cambiar estado del registro de código igual a 2.

Si el usuario selecciona un código y no desea modificar su estado, basta con que de un clic en el botón “Cancelar” y los controles vuelven a su estado inicial o simplemente el usuario puede abandonar la página (sub-pestaña Cambiar Estado).

5.2.2.5 VENTANA INFORMACIÓN DE PRODUCTO.

Se trata de una ventana que despliega el programa cuando el usuario hace doble clic en alguna de las filas de la Tabla General que se encuentra en la sub-pestaña Consulta General. Esta ventana, que se muestra en la figura 5.27, contiene varios cuadros en donde se muestran los datos correspondientes al registro de producto seleccionado. Proporciona al usuario información del producto seleccionado en cuanto a su Código, Descripción, Existencia, Fecha y Hora de Último Movimiento, Precio por Unidad y Estado.

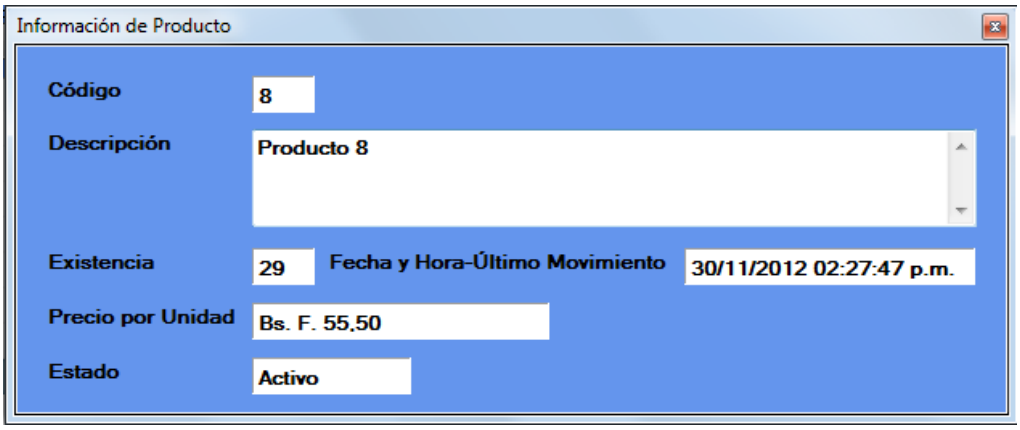


Figura 5. 27. Ventana de Información de Producto.

Para salir de esta ventana el usuario solo debe dar clic en la opción cerrar que se encuentra en el borde superior derecho de la ventana.

5.2.2.6 ICONO DE NOTIFICACIÓN.

Se trata de un icono como el que se muestra en la figura 5.28, el cual aparece en la barra de tareas del ordenador cada vez algún producto cambia su cantidad cuando se actualizan las tablas de inventario con los datos recibidos en el puerto serial asignado en la aplicación. Cuando esto ocurre el icono muestra un globo que contiene información que notifica al usuario la ocurrencia del evento que se acaba de describir, ver figura 5.29.

Figura 5. 28. Icono de notificación.

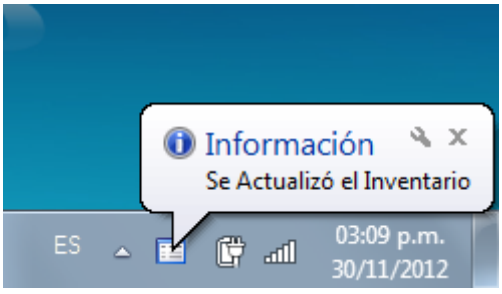


Figura 5. 29. Globo de información del icono de notificación.

5.2.3 MANUAL DE INSTALACIÓN DEL SOFTWARE.

Para llevar a cabo la instalación del software SCI V1.0 es necesario descomprimir el archivo .7Z llamado "SCI V1.0 INSTALADOR" y a continuación se descomprime una carpeta con el mismo nombre, en ella se mostrarán tres archivos y una carpeta, ver figura 5.30.

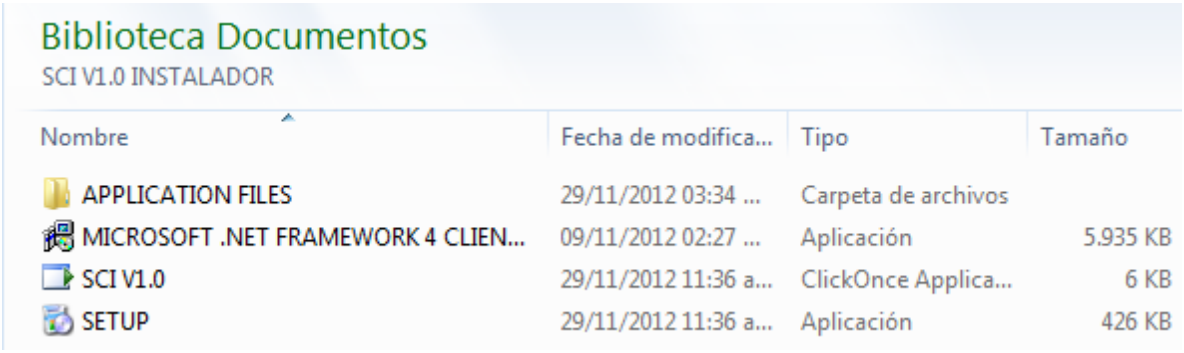


Figura 5. 30. Carpeta descomprimida "SCI V1.0 INSTALADOR".

En la carpeta con el nombre "APPLICATION FILES" se encuentran una serie de archivos que no deben ser movidos o borrados de esa carpeta, además es necesario que todos esos archivos estén contenidos en una misma carpeta para que el software pueda ejecutarse de manera adecuada sin presentar errores.

Como primer paso debe ejecutarse el instalador de la actualización "Microsoft.NET Framework 4 ClientProfile ESN Language Pack Setup", el cual es un modelo de programación de Microsoft que sirve de soporte a otros programas creados bajo esta misma plataforma. Al hacer doble clic sobre el archivo mencionado, se mostrará en pantalla la siguiente ventana, ver figura 5.31.

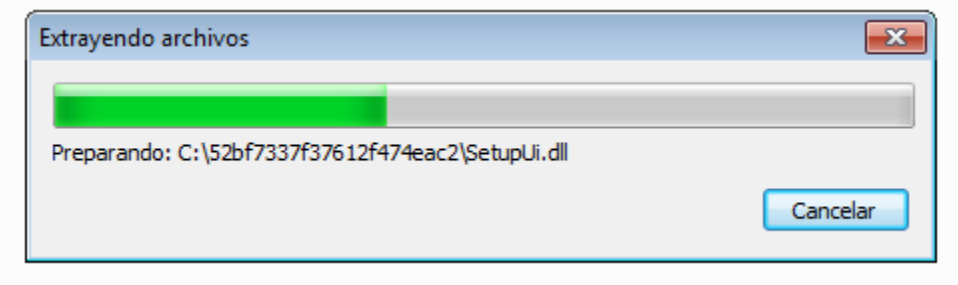


Figura 5. 31. "Microsoft.NET Framework 4 Client Profile ESN Language Pack Setup", Primera ventana.

Una vez se cargue la barra de progreso se muestra en pantalla la ventana que se muestra en la figura 5.32. Se debe seleccionar la opción como se muestra en la figura 5.32 y luego dar clic en el botón "Instalar" para continuar.

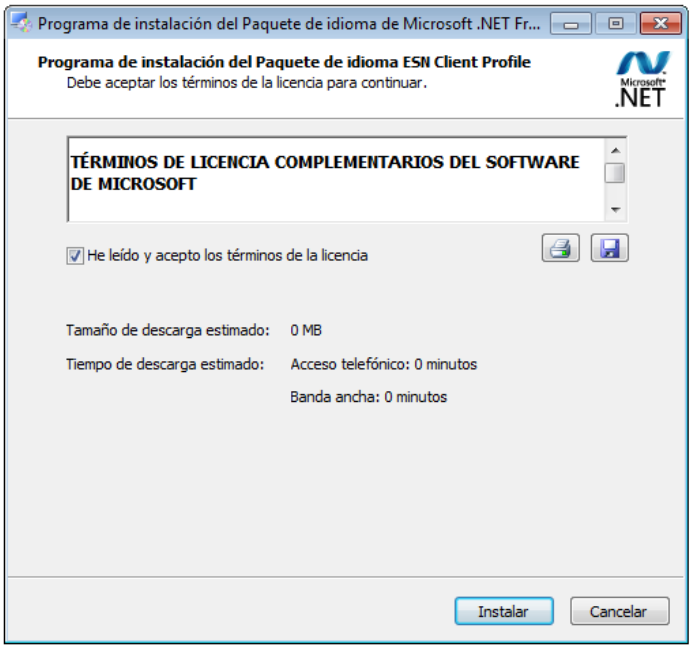


Figura 5. 32. "Microsoft.NET Framework 4 Client Profile ESN Language Pack Setup", Segunda ventana.

Luego se mostrará la ventana de la figura 5.33, donde se muestra el estado de la instalación.

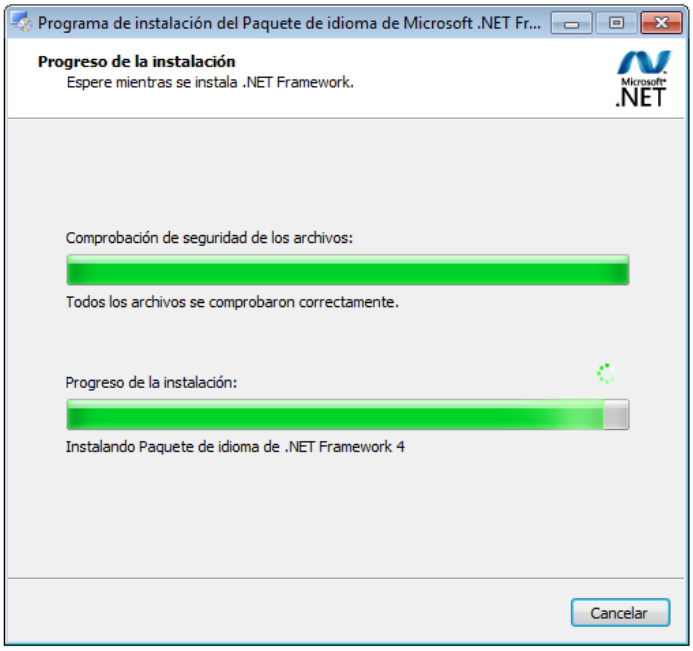


Figura 5. 33. "Microsoft.NET Framework 4 Client Profile ESN Language Pack Setup", Tercera ventana.

Una vez completada la instalación aparece la ventana de la figura 5.34, la cual indica que se completó la instalación y solo se debe hacer clic en el botón "Finalizar" para concluir la instalación de "Microsoft.NET Framework 4".

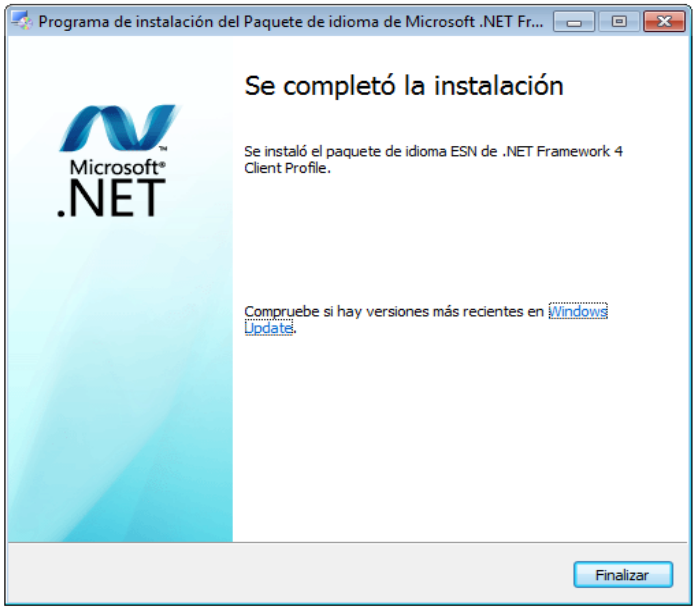


Figura 5. 34. "Microsoft.NET Framework 4 Client Profile ESN Language Pack Setup", Cuarta ventana.

Por último, se debe ejecutar el instalador del software SCI V1.0, el cual aparece con el nombre “SETUP”. Una vez hecho todo esto la aplicación está lista para iniciarse y solo se requiere dar doble clic en el archivo que aparece con el nombre “SCI V1.0”, el cual no es más que un acceso directo a la aplicación.

5.3 PRUEBA DEL SISTEMA DE CONTROL DE INVENTARIO.

Esta prueba se realizó con la finalidad de verificar que los datos que se reciben y se muestran en la interfaz de usuario, provenientes de la tarjeta de desarrollo, corresponden con los datos enviados por el módulo de información (escritos en la ventana del ComDebug) a la tarjeta de desarrollo.

Para realizar esta prueba se procedió a realizar el ensamblaje de todas las partes del sistema. Es importante destacar que se utilizaron solo 4 tipos de producto y un máximo de 8 ejemplares de cada tipo de producto, con el fin de agilizar el proceso de prueba.

En la figura 5.35 se muestra la ventana del ComDebug con los códigos de cada producto y los ejemplares existentes, que simularon los tags detectados por el lector RFID.

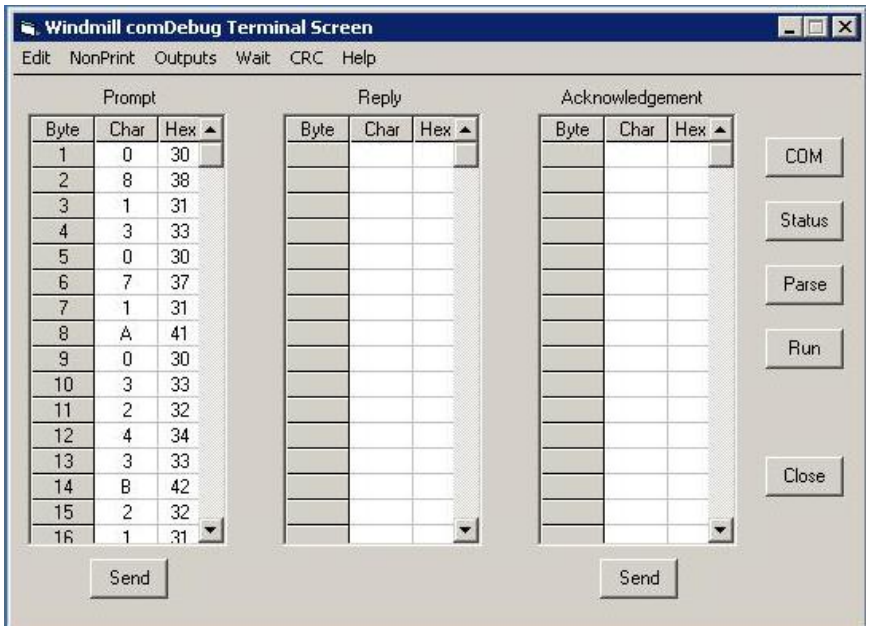


Figura 5. 35. Ventana ComDebug.

Los datos mostrados anteriormente significan lo siguiente:

- Para el producto de Código 1 existen tres ejemplares (Cantidad=3)
- Para el producto de Código 2 existen tres ejemplares (Cantidad=2)
- Para el producto de Código 3 existen tres ejemplares (Cantidad=2)
- Para el producto de Código 4 existen tres ejemplares (Cantidad=1)

Una vez procesados estos datos por la FPGA, ésta envió la trama correspondiente de dichos datos al computador del módulo administrador.

Los datos recibidos en la interfaz gráfica se almacenaron en las tablas de inventario que ella contiene y resultaron ser los esperados, como puede observarse en las figuras 5.36 y 5.37.

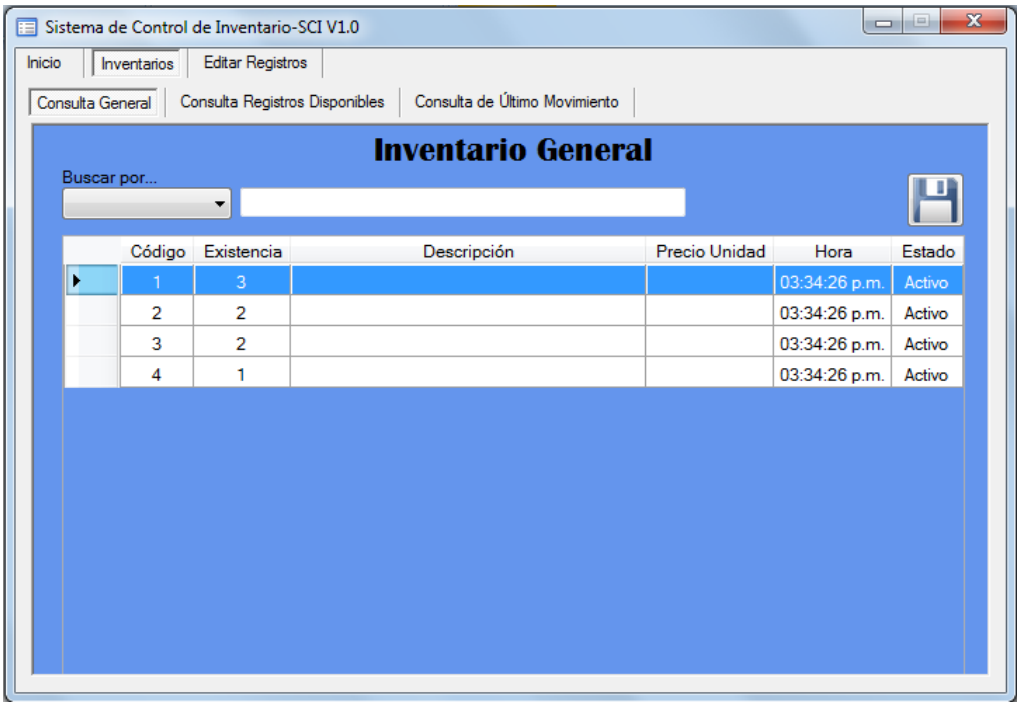


Figura 5. 36. Pestaña Inventarios, Sub-pestaña Consulta General.

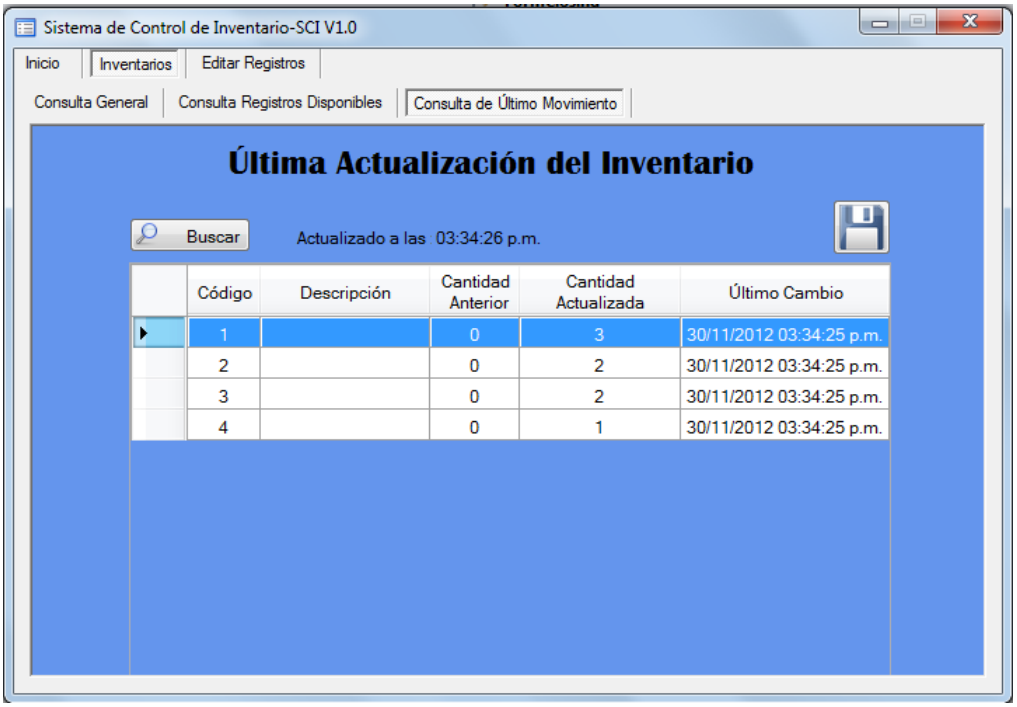


Figura 5. 37. Pestaña Inventarios, Sub-pestaña Consulta de Último Movimiento.

Un segundo envío de datos se realizó variando solo la cantidad del producto de Código 3, como se muestra en la figura 5.38.

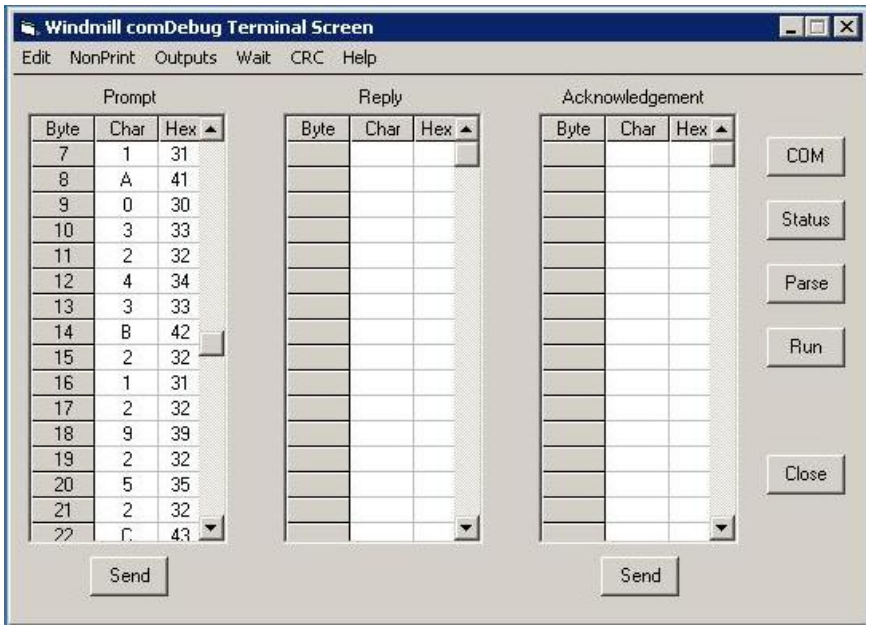


Figura 5. 38. Ventana ComDebug.

El resultado obtenido en la interfaz gráfica, se muestra en las figuras 5.39 y 5.40. Se puede apreciar en la tabla de Consulta General que evidentemente cambió la cantidad del producto de Código 3 y, además, la fila de este producto cambió a color verde (indicación de cambio). De la misma manera se puede apreciar en la tabla de Consulta de Último Movimiento un cambio en la fecha y hora solo del producto que sufrió cambio (Código 3).

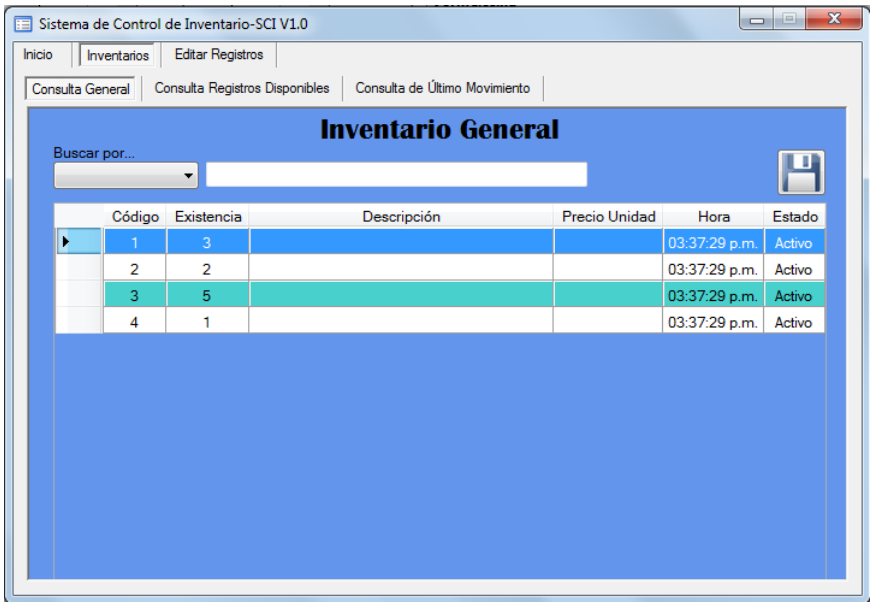


Figura 5. 39. Pestaña Inventarios, Sub-pestaña Consulta General.

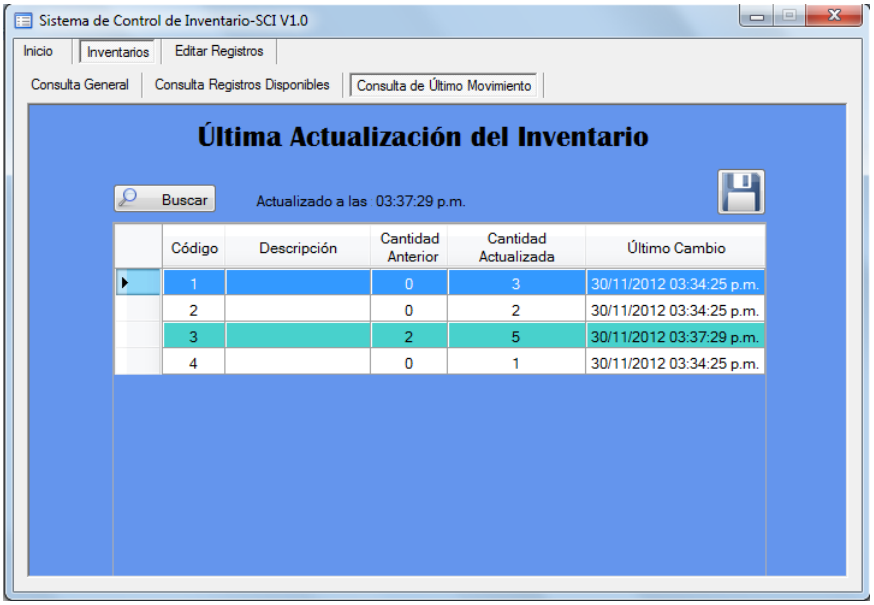


Figura 5. 40. Pestaña Inventarios, Sub-pestaña Consulta de Último Movimiento.

Es importante destacar que durante la operación del sistema completo si el usuario presiona el botón de "Reset" de la tarjeta de desarrollo FPGA, dicho botón debe volver a presionarse inmediatamente para evitar que se produzcan interrupciones indeseadas en la recepción de los datos de la interfaz, que afecten la actualización de los datos del inventario.

CONCLUSIONES.

- El prototipo construido cumple con el objetivo general propuesto en este Trabajo de Grado, ya que se logró el diseño de un sistema capaz de controlar y gestionar el inventario de una empresa, utilizando un dispositivo de hardware reconfigurable para procesar los datos referentes a la cantidad de cada producto existente, y una interfaz gráfica que permite consultar y hacerle modificaciones a cada uno de los registros de dicho inventario; constituyendo así, el sistema completo, una herramienta útil para obtener la mayor precisión en el recuento y balance de la mercancía de productos, así como también una vía factible para eliminar costos de personal y tiempo invertido en su realización.

- El software SCI V1.0 se diseñó de manera que permita que su manipulación sea lo más práctica posible para el usuario, y fue desarrollado en un entorno de diseño que permitió obtener una interfaz gráfica con características muy similares al sistema operativo Windows, lo cual resulta más amigable al usuario. Esta interfaz de usuario presenta todas las funciones necesarias para gestionar y visualizar el inventario en tiempo real, permitiendo al usuario tener conocimiento acerca de todos los movimientos (entrada y salida) de productos que ocurran en el almacén, lo cual también puede ser muy útil en caso de extravío de alguno de ellos.

- El software diseñado permite al usuario guardar reportes del inventario en un documento tipo Excel, lo cual afianza aún más el control de los mismos. Esta acción permite también corroborar la correspondencia de la información de dicho reporte con un recuento de mercancía hecho manualmente en un determinado momento.

- El ensamblaje del prototipo para el sistema de control de inventario resultó sencillo, puesto que los componentes son de fácil instalación. El sistema construido funcionó

como se esperaba, ya que los datos de cada producto que envía el modulo de información se muestran instantáneamente la interfaz de usuario, por lo que la transferencia de datos es rápida.

- El sistema está hecho para que pueda ser utilizado fácilmente por cualquier usuario con conocimientos básicos de informática sin requerir conocimientos previos sobre sistemas electrónico, ya que sólo debe remitirse al manejo del Sistema de Control de Inventario explicado en el capitulo V y seguir las instrucciones; pues allí se especificaron con detalles cada uno de los pasos necesarios a seguir para que el sistema completo quede instalado sin errores y el funcionamiento de cada una de las ventanas que presenta la interfaz de usuario.

RECOMENDACIONES.

- Se recomienda que al presionar el botón de “Reset” de la tarjeta de desarrollo FPGA inmediatamente vuelva a presionarse, para que el sistema continúe trabajando normalmente y así evitar interrupciones indeseadas que afecten la labor diaria de actualización del inventario. En vista de ello, también es recomendable configurar una alarma en el módulo administrador (computador) de manera que se proporcione un aviso al operador que le recuerde volver a pulsar el botón “Reset”.

- Para instalaciones futuras de este sistema de control en almacenes con mayor variedad y cantidad de productos, es decir, que esperen contener más 1600 registros (resultado de la productoria de 40 tipos de productos en el almacen con una cantidad máxima de 40 ejemplares); se recomienda la adquisición de una tarjeta de desarrollo FPGA que posea una mayor capacidad de memoria para cubrir los requerimientos en cuanto a los registros del inventario.

- Implementar este sistema utilizando comunicación USB, ya que en el mercado también existen lectores RFID con puertos de comunicación USB.

- Implementar las tablas de inventario existentes en la interfaz de usuario con el sistema de administración y análisis de bases de datos Microsoft® SQL Server™, junto con SQL Server Reporting Services para publicar, en internet, la información general del inventario al público global o a usuarios autorizados y autenticados.

REFERENCIAS BIBLIOGRÁFICAS.

[1] Aponte, William y López, Rogers. (2012). *"Diseño y Construcción de un Prototipo para el Control de Acceso a las Aulas y Laboratorios de la Escuela de Ingeniería Eléctrica de la Universidad de Carabobo utilizando Tecnología RFID"*. Trabajo Especial de Grado presentado para optar al título de Ingeniero Electricista, Universidad de Carabobo, Facultad de Ingeniería, Naguanagua, Edo. Carabobo, Venezuela.

[2] Fernández Yépez, Johanna Lizeth. (2010). *"Implementación De La Etapa De Transmisión De Un Sistema De Comunicación Digital Utilizando La Tecnología FPGA"*. [Online]. Disponible en: http://repositorio.espe.edu.ec/handle/21000/2637.

[3] Collao Vilches, Carlos Alfredo. (2008). "Sistema de Soporte para Control de Inventarios mediante RFID". [Online]. Disponible en: http://www.cybertesis.cl/tesis/uchile/2008/collao_c/sources/collao_c.pdf.

[4] Ciudad Herrera, José María y Samá Casanovas, Eduard. (2005). *"Estudio, Diseño Y Simulación De Un Sistema De RFID Basado En Epc"*. [Online]. Disponible en: http://upcommons.upc.edu/pfc/bitstream/2099.1/3552/2/40883-2.pdf.

[5] Reyes Meléndez, Keishla M. (2012). *"R.F.I.D.: Dispositivos de Identificación por Radio Frecuencias"*. [Online]. Disponible en: http://infusc2012.wordpress.com/2012/04/19/r-f-i-d-dispositivos-de-identificacion-por-radiofrecuencias/

[6] Rico López, Rafael. *"Apuntes de VHDL"*. [Online]. Disponible en: http://atc2.aut.uah.es/~rico/docencia/asignaturas/informatica/lab_org_comp/archivos/Documentacion/VHDL/Apuntes%20VHDL%2000.pdf

[7] Pérez R., Aída y Simone P., Andrés. (2006). *"Incorporación del lenguaje VHDL para la síntesis, descripción y simulación de circuitos lógicos digitales bajo el estándar*

IEEE 1076-2002". Trabajo presentado para ascender a la categoría de Profesor Asistente, Universidad de Carabobo, Facultad de Ingeniería, Naguanagua, Edo. Carabobo, Venezuela.

[8] López Pérez, Eric. *"Tutorial del Protocolo RS-232"*. [Online]. Disponible en: http://cselectrobomba.googlecode.com/files/Serial_RS232.pdf

[9] Ronquillo, Ulysses. (2007). "*Microsoft .NET".* [Online]. Disponible en: http://tuyub.wordpress.com/2007/07/18/microsoft-net/

[10] Panello, Demian. (2011). *"Introducción a Microsoft.NET".* [Online].Disponible en: http://www.dcp.com.ar/intro_net.htm

[11] Leal H., Javier J. (2010). *"Visual Studio 2010 Express"*. [Online]. Disponible en: http://javierleal.wordpress.com/2010/09/06/visual-studio-2010-express-espaol/

[12] Zambrano M., Ángel A. (2011). "Anexo: Introducción al Visual Basic". [Online]. Disponible en: http://webdelprofesor.ula.ve/economia/angelz/archivos/guia_vb.pdf

[13] Rondinelli, Mauro. (2008). *"Tutorial de Visual Basic"*. [Online]. Disponible en: http://www.elguruprogramador.com.ar/tutoriales/visual-basic/entorno-de-trabajo-de-visual-basic.htm

[14] Sáez A., Justo. (2010). *"Visual Basic .NET – Controles más habituales".* [Online]. Disponible en: http://mimosa.pntic.mec.es/~jsaez9/Clases/vb/Temas/04_Controles_habituales.pdf

[15] Microsoft. (2010). *"DataGridView (Propiedades)"*. [Online].Disponible en: http://msdn.microsoft.com/es-es/library/microsoft.office.tools.word.controls.datagridview_properties%28v=vs.100%29.aspx

[16] Universidad Pedagógica Experimental Libertador. (2006). *"Manuales de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales".* (3era Reimpresión). [Online]. Disponible en: http://neutron.ing.ucv.ve/NormasUPEL2006.pdf

APÉNDICES

APÉNDICE A:

Programación desarrollada en VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ModuloDeControl is
Port(ClkIn : in STD_LOGIC; Reset : in STD_LOGIC;DInAd : in STD_LOGIC;
        DInInf : in STD_LOGIC; DOutAd : out STD_LOGIC; DOutInf : out STD_LOGIC;
        Alarma : out STD_LOGIC);
end ModuloDeControl;

architecture Funcionamiento of ModuloDeControl is

-- CONTROL DEL DIVISOR DE FRECUENCIA
-- Frecuencia Interna de la Tarjeta de Desarrollo de 50 MHz
-- Frecuencia de Comunicación con:
-- Módulo de Informacion de 19200 Baudios
-- Módulo Administrador de 9600 Baudios
-- Cuenta del Módulo de Información y Administrador respectivamente
signal Cuenta1, Cuenta2 : natural range 0 to 5207 := 0;
-- Relojes Deseados
signal Clk1, Clk2 : std_logic;
-- Habilita y DesHabilita Contadores
signal HInInf, HInAd, HOutAd : std_logic := '0';

-- RELOJ REAL
-- Cuenta del Reloj
signal Reloj : natural range 0 to 500000000 := 0;

-- RESET DEL SISTEMA
-- Cuenta de Antirebote
signal Cuenta : natural range 0 to 6000000 := 0;
-- Reset Manual del Sistema
signal Reset1 : std_logic := '1';
-- Reset Automático del Sistema
signal Reinicio, R : std_logic := '1';
-- Comunicación Activa/Inactiva
signal Reset2 : std_logic := '0';

-- SEÑALES Y CONSTANTES DE CONTROL DEL MODULO DE TRANSMISIÓN
-- Control de Etapas
signal TX2 : std_logic_vector (1 downto 0) := "00";
-- Bandera de Transmisión
signal Transmite : std_logic := '1';
-- Bits Enviados o Recibidos del Módulo Servidor
signal BitEnv2, BitRec2 : natural range 0 to 7 := 7;
-- Bits Recibidos del Módulo de Información
signal BitRec1 : natural range 0 to 7;
```

```
-- SEÑALES Y CONSTANTES DE CONTROL DEL MÓDULO DE RECEPCIÓN
-- Control de Etapas
signal RX1, RX2 : std_logic := '0';
-- Almacenamiento
signal Producto, Codigo, DatPar : std_logic_vector (7 downto 0);
-- Selección de Codigo Recibido
signal ProCod : natural range 0 to 2 := 2;
-- Bandera de Información Nueva
signal Nuevo : std_logic := '0';

-- SEÑALES Y CONSTANTES PARA EL CONTROL DE INVENTARIO
-- Ejemplar del Producto Detectado
type Posibilidad is array (1 to 40) of natural range 0 to 1;
-- Producto Detectado
type Control is array (1 to 40) of Posibilidad;
signal Inventario, II : Control :=(
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --1
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --2
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --3
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --4
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --5
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --6
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --7
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --8
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --9
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --10
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --11
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --12
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --13
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --14
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --15
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --16
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --17
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --18
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --19
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --20
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --21
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --22
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --23
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --24
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --25
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --26
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --27
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --28
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --29
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --30
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --31
```

```
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --32
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --33
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --34
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --35
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --36
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --37
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --38
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), --39
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)  --40
);
-- Registro de Existencia
type Cantidad is array (1 to 40) of natural range 0 to 40;
signal Existencia, EI : Cantidad :=
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
signal Aux : std_logic := '1';
signal P, C : natural range 0 to 40 := 0;

-- REGISTRO ORGANIZADO DE LOS PRODUCTOS
type Envio is array (1 to 80) of std_logic_vector(7 downto 0);
signal Salida : Envio;

begin

DOutInf <= '1';
Alarma <= Reset1;
-------------------------------------------------------------------------------------------------------------------
ControldeReset : Process(ClkIn,Reset,Reset2,Reinicio)
variable U, V: std_logic := '0';
begin
if ClkIn'event and ClkIn = '1' then
-- Antirebote del Boton de Reset
 if V = '0' then
  if Reset = '0' then
   if Cuenta = 5999999 then Cuenta <= 0;
    if Reset = '0' then
     if U = '0' then Reset1 <= '0'; U := '1'; V := '1';
     elsif U = '1' then Reset1 <= '1'; U := '0'; V := '1';
     end if;
    end if;
   else Cuenta <= Cuenta + 1;
   end if;
  else Cuenta <= 0;
  end if;
 else
  if Reset = '1' then
   if Cuenta = 5999999 then Cuenta <= 0; if Reset = '1' then V := '0'; end if;
   else Cuenta <= Cuenta + 1;
   end if;
```

```
  else Cuenta <= 0;
  end if;
 end if;
end if;
end process;
-------------------------------------------------------------------------------------------------------------------
ControldeInventario : Process(ClkIn,Aux,Transmite,Reset1,Reinicio)
variable Conteo : Cantidad;
begin
if ClkIn'event and ClkIn = '1' then Existencia <= Conteo;
 if Reset1 = '1' and Reinicio = '1' then
  if Aux = '1' then Aux <= '0';
   if P > 0 and C > 0 then if Inventario(P)(C) = 0 then Inventario(P)(C) <= 1; end if;
   end if;
  elsif Aux = '0' then Aux <= '1';
   for i in 1 to 40 loop Conteo(i) := Inventario(i)(1);
    for j in 1 to 39 loop Conteo(i) := Inventario(i)(j+1) + Conteo(i);
    end loop;
   end loop;
  end if;
 else Inventario <= II; Conteo := EI;
 end if;
end if;
end process;
-------------------------------------------------------------------------------------------------------------------
RelojReal : Process(ClkIn,Reloj)
variable Enviar : Envio := (
x"01",x"00",x"02",x"00",x"03",x"00",x"04",x"00",x"05",x"00",x"06",x"00",x"07",x"00",
x"08",x"00",x"09",x"00",x"0A",x"00",x"0B",x"00",x"0C",x"00",x"0D",x"00",x"0E",x"00",
x"0F",x"00",x"10",x"00",x"11",x"00",x"12",x"00",x"13",x"00",x"14",x"00",x"15",x"00",
x"16",x"00",x"17",x"00",x"18",x"00",x"19",x"00",x"1A",x"00",x"1B",x"00",x"1C",x"00",
x"1D",x"00",x"1E",x"00",x"1F",x"00",x"20",x"00",x"21",x"00",x"22",x"00",x"23",x"00",
x"24",x"00",x"25",x"00",x"26",x"00",x"27",x"00",x"28",x"00");
begin
if ClkIn'event and ClkIn = '1' then
 if Reset1 = '1' then
  case Reloj is
   when 0 => Reinicio <= '0'; Reloj <= 1;
   when 10 => Reloj <= 11; Reinicio <= '1'; Transmite <= '0';
   when 490000000 => Reloj <= 490000001;
    for i in 1 to 40 loop
     case Existencia(i) is
      when 1 => Enviar(2*i):=x"01"; when 2 => Enviar(2*i):=x"02";
      when 3 => Enviar(2*i):=x"03"; when 4 => Enviar(2*i):=x"04";
      when 5 => Enviar(2*i):=x"05"; when 6 => Enviar(2*i):=x"06";
      when 7 => Enviar(2*i):=x"07"; when 8 => Enviar(2*i):=x"08";
      when 9 => Enviar(2*i):=x"09"; when 10 => Enviar(2*i):=x"0A";
      when 11 => Enviar(2*i):=x"0B"; when 12 => Enviar(2*i):=x"0C";
```

```
      when 13 => Enviar(2*i):=x"0D"; when 14 => Enviar(2*i):=x"0E";
      when 15 => Enviar(2*i):=x"0F"; when 16 => Enviar(2*i):=x"10";
      when 17 => Enviar(2*i):=x"11"; when 18 => Enviar(2*i):=x"12";
      when 19 => Enviar(2*i):=x"13"; when 20 => Enviar(2*i):=x"14";
      when 21 => Enviar(2*i):=x"15"; when 22 => Enviar(2*i):=x"16";
      when 23 => Enviar(2*i):=x"17"; when 24 => Enviar(2*i):=x"18";
      when 25 => Enviar(2*i):=x"19"; when 26 => Enviar(2*i):=x"1A";
      when 27 => Enviar(2*i):=x"1B"; when 28 => Enviar(2*i):=x"1C";
      when 29 => Enviar(2*i):=x"1D"; when 30 => Enviar(2*i):=x"1E";
      when 31 => Enviar(2*i):=x"1F"; when 32 => Enviar(2*i):=x"20";
      when 33 => Enviar(2*i):=x"21"; when 34 => Enviar(2*i):=x"22";
      when 35 => Enviar(2*i):=x"23"; when 36 => Enviar(2*i):=x"24";
      when 37 => Enviar(2*i):=x"25"; when 38 => Enviar(2*i):=x"26";
      when 39 => Enviar(2*i):=x"27"; when 40 => Enviar(2*i):=x"28";
      when others => Enviar(2*i):=x"00";
     end case;
    end loop;
   when 495000000 => Salida <= Enviar; Reloj <= 495000001;
   when 499999999 => Transmite <= '1'; Reloj <= 0;
   when others => Reloj <= (Reloj + 1);
  end case;
 else Reloj <= 0; Transmite <= '0';
 end if;
end if;
end process;
-------------------------------------------------------------------------------------------------------------------
FrecuenciaInformacion : Process(Reset1,ClkIn,HInInf)
-- Valor del Reloj de Comunicacion con el Modulo de Informacion
variable C1 : std_logic := '0';
begin
if Reset1 = '0' or HInInf = '0' then Cuenta1 <= 0; C1 := '0';
elsif ClkIn'event and ClkIn = '1' then
 if Cuenta1 = 2603 then Cuenta1 <= 0; C1 := '1'; else Cuenta1 <= (Cuenta1 + 1); C1 := '0';
 end if;
end if;
Clk2 <= C1;
end process;
-----------------------------------------------------------------------------------
FrecuenciaAdministrador : Process(Reset1,ClkIn,HInAd,HOutAd)
-- Valor del Reloj de Comunicacion con el Modulo Administrador
variable C2 : std_logic := '0';
begin
if Reset1 = '0' or (HInAd = '0' and HOutAd = '0') then Cuenta2 <= 0; C2 := '0';
elsif ClkIn'event and ClkIn = '1' then
 if Cuenta2 = 5207 then Cuenta2 <= 0; C2 := '1'; else Cuenta2 <= (Cuenta2 + 1); C2 := '0';
 end if;
end if;
Clk1 <= C2;
```

```
end process;
-------------------------------------------------------------------------------------------------------------------
RecepcionInformacion : Process(DInInf,Reset1,Clk2,HInInf,RX1,ClkIn)
-- Valor Actual de la Entrada
variable D1 : std_logic;
begin
if Reset1 = '0' or Reinicio = '0' then
 HInInf <= '0'; ProCod <= 0; Producto <= x"00"; Codigo <= x"00";
else
 if HInInf = '0' then BitRec1 <= 0;
  if DInInf = '0' then HInInf <= '1'; RX1 <= '0'; if ProCod = 2 then ProCod <= 0; end if;
  end if;
 elsif Clk2'event and CLk2 = '1' then
  if RX1 = '0' then
   if BitRec1 = 7 then D1 := DInInf; DatPar(BitRec1) <= D1; RX1 <= '1';
   else D1 := DInInf; DatPar(BitRec1) <= D1; BitRec1 <= BitRec1 + 1;
   end if;
  elsif RX1 = '1' then HInInf <= '0';
   if ProCod = 0 then Producto <= DatPar; ProCod <= 1; Nuevo <= '0';
   elsif ProCod = 1 then Codigo <= DatPar; ProCod <= 2; Nuevo <= '1';
   end if;
  end if;
 end if;
end if;
-- Manejo de Registro del Inventario
if ClkIn'event and ClkIn = '1' then
 if Reset1 = '1' then
  if ProCod = 2 then
   case Producto is
    when x"30"=> P<=1; when x"31"=> P<=2; when x"32"=> P<=3; when x"33"=> P<=4;
    when x"34"=> P<=5; when x"35"=> P<=6; when x"36"=> P<=7; when x"37"=> P<=8;
    when x"38"=> P<=9; when x"39"=> P<=10; when x"41"=> P<=11; when x"42"=> P<=12;
    when x"43"=> P<=13; when x"44"=> P<=14; when x"45"=> P<=15; when x"46"=> P<=16;
    when x"47"=> P<=17; when x"48"=> P<=18; when x"49"=> P<=19; when x"4A"=> P<=20;
    when x"4B"=> P<=21; when x"4C"=> P<=22; when x"4D"=> P<=23; when x"4E"=> P<=24;
    when x"4F"=> P<=25; when x"50"=> P<=26; when x"51"=> P<=27; when x"52"=> P<=28;
    when x"53"=> P<=29; when x"54"=> P<=30; when x"55"=> P<=31; when x"56"=> P<=32;
    when x"57"=> P<=33; when x"58"=> P<=34; when x"59"=> P<=35; when x"5A"=> P<=36;
    when x"61"=> P<=37; when x"62"=> P<=38; when x"63"=> P<=39; when x"64"=> P<=40;
    when others=> P<=0;
   end case;
   case Codigo is
    when x"30"=> C<=1; when x"31"=> C<=2; when x"32"=> C<=3; when x"33"=> C<=4;
    when x"34"=> C<=5; when x"35"=> C<=6; when x"36"=> C<=7; when x"37"=> C<=8;
    when x"38"=> C<=9; when x"39"=> C<=10; when x"41"=> C<=11; when x"42"=> C<=12;
    when x"43"=> C<=13; when x"44"=> C<=14; when x"45"=> C<=15; when x"46"=> C<=16;
    when x"47"=> C<=17; when x"48"=> C<=18; when x"49"=> C<=19; when x"4A"=> C<=20;
    when x"4B"=> C<=21; when x"4C"=> C<=22; when x"4D"=> C<=23; when x"4E"=> C<=24;
```

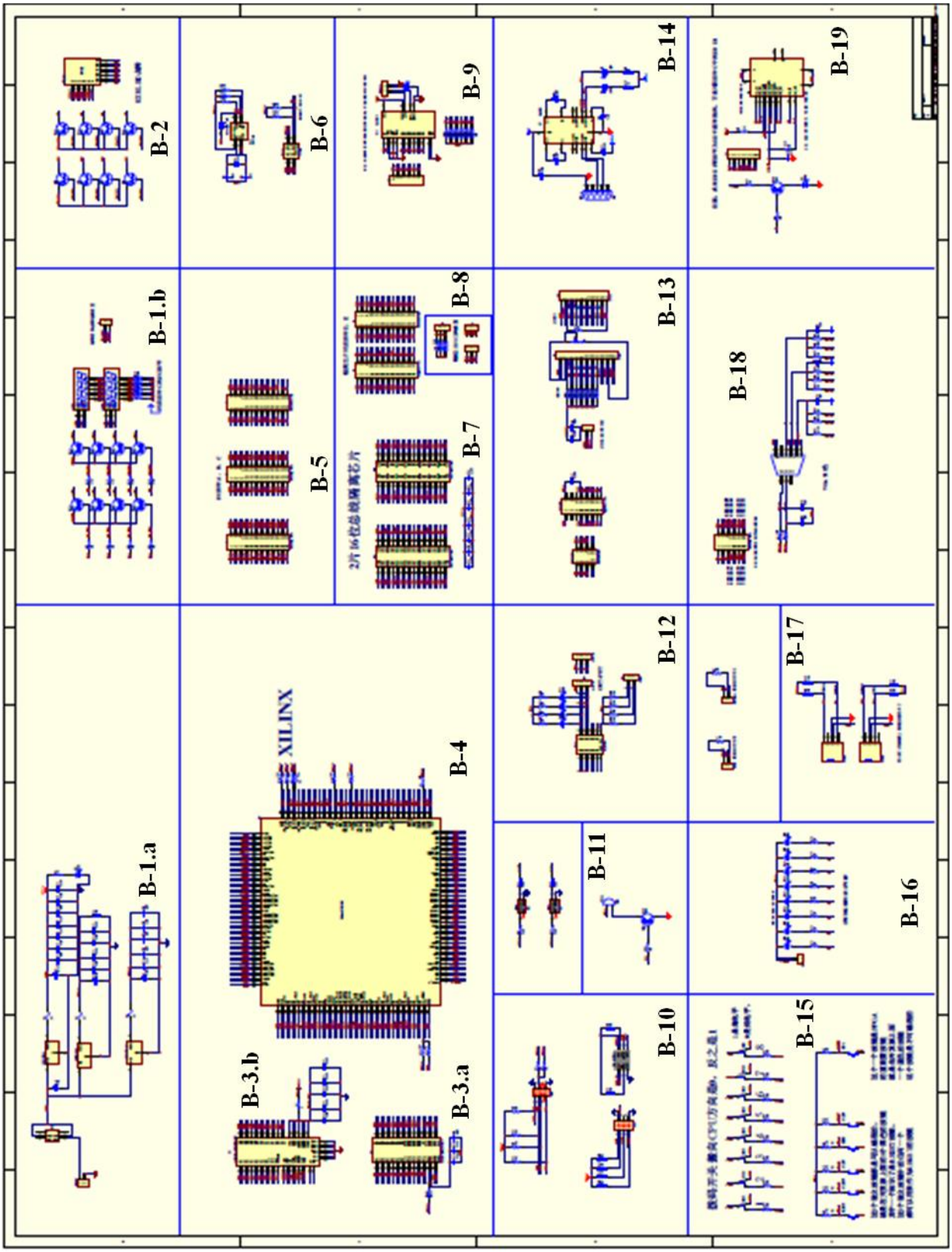
```
    when x"4F"=> C<=25; when x"50"=> C<=26; when x"51"=> C<=27; when x"52"=> C<=28;
    when x"53"=> C<=29; when x"54"=> C<=30; when x"55"=> C<=31; when x"56"=> C<=32;
    when x"57"=> C<=33; when x"58"=> C<=34; when x"59"=> C<=35; when x"5A"=> C<=36;
    when x"61"=> C<=37; when x"62"=> C<=38; when x"63"=> C<=39; when x"64"=> C<=40;
    when others=> C<=0;
   end case;
  else P <= 0; C <= 0;
  end if;
 else P <= 0; C <= 0;
 end if;
end if;
end process;
-------------------------------------------------------------------------------------------------------------------
RecepcionAdministrador : Process(DInAd,Reset1,Clk1,HInAd,RX2,ClkIn)
variable OI : std_logic_vector(7 downto 0):= x"FF";
begin
if Reset1 = '0' then HInAd <= '0';
else
 if HInAd = '0' then BitRec2 <= 0; if DInAd = '0' then HInAd <= '1'; RX2 <= '0'; end if;
 elsif Clk1'event and Clk1 = '1' then
  if RX2 = '0' then
   if BitRec2 = 7 then OI(BitRec2) := DInAd; RX2 <= '1';
   else OI(BitRec2) := DInAd; BitRec2 <= BitRec2 + 1;
   end if;
  elsif RX2 = '1' then HInAd <= '0';
-- Arranque y Parada de la Comunicacion
   if OI = x"40" then Reset2 <= '1'; elsif OI = x"3F" then Reset2 <= '0';
   end if;
  end if;
 end if;
end if;
end process;
-------------------------------------------------------------------------------------------------------------------
TransmicionAdministrador : Process (Clk1,Reset1,HOutAd,ClkIn,Reloj,Transmite)
-- Cantidad de Registros a Enviar
variable N1 : natural range 1 to 80 := 1;
begin
if Reset1 = '0' or Reset2 = '0' then DOutAd <= '1'; HOutAd <= '0'; TX2 <= "00"; N1 := 1;
else
 if HOutAd = '0' then DOutAd <= '1'; if Transmite = '1' then HOutAd <= '1'; end if;
 elsif Clk1'event and Clk1 = '1' then
  if TX2 = "00" then DOutAd <= '0'; TX2 <= "01"; BitEnv2 <= 0;
  elsif TX2 = "01" then
   if BitEnv2 = 7 then TX2 <= "10"; DOutAd <= Salida(N1)(BitEnv2);
   else BitEnv2 <= BitEnv2 + 1; DOutAd <= Salida(N1)(BitEnv2);
   end if;
  elsif TX2 = "10" then DOutAd <= '1'; TX2 <= "11";
  elsif TX2 = "11" then TX2 <= "00";
```

```
   if N1 = 80 then N1 := 1; HOutAd <= '0'; else N1 := N1 + 1;
    end if;
  end if;
 end if;
end if;
end process;
-------------------------------------------------------------------------------------------
end Funcionamiento;
```

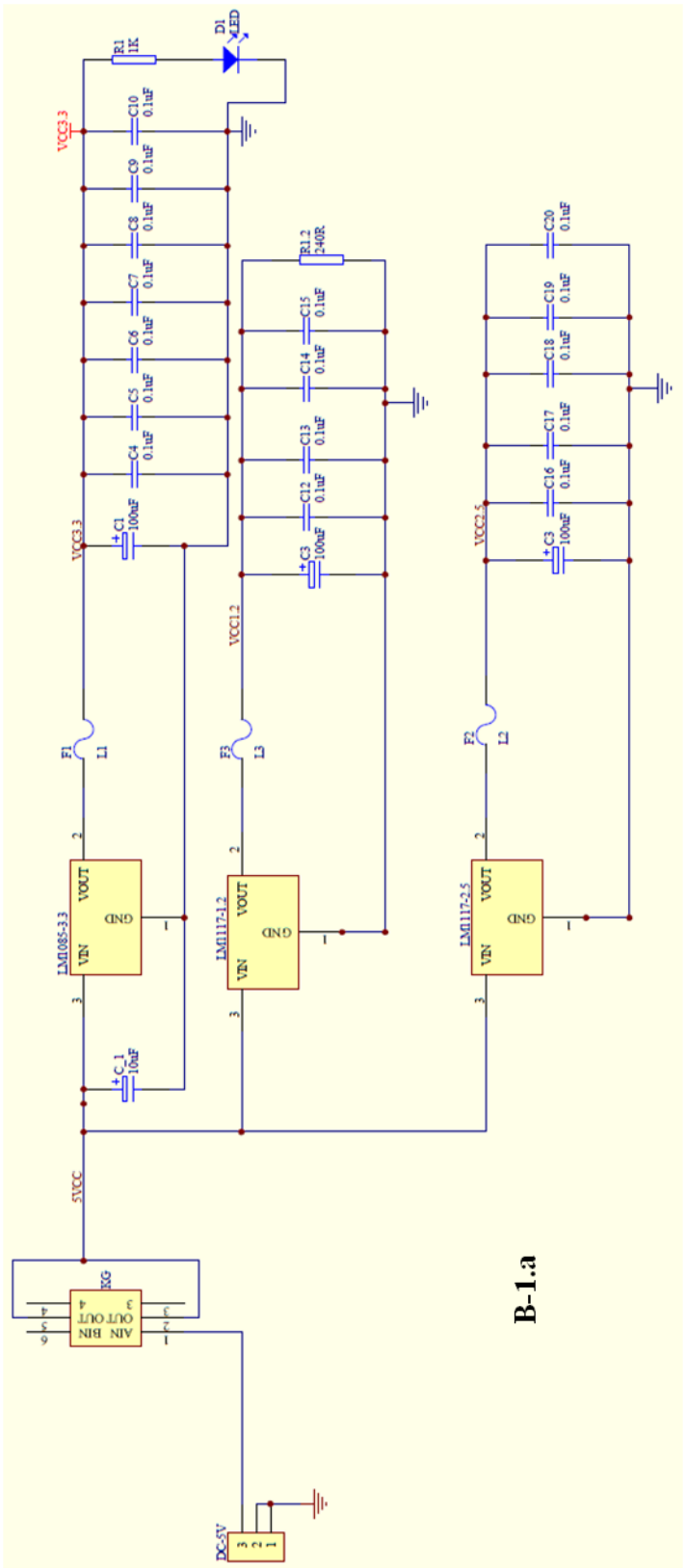
APÉNDICE B:

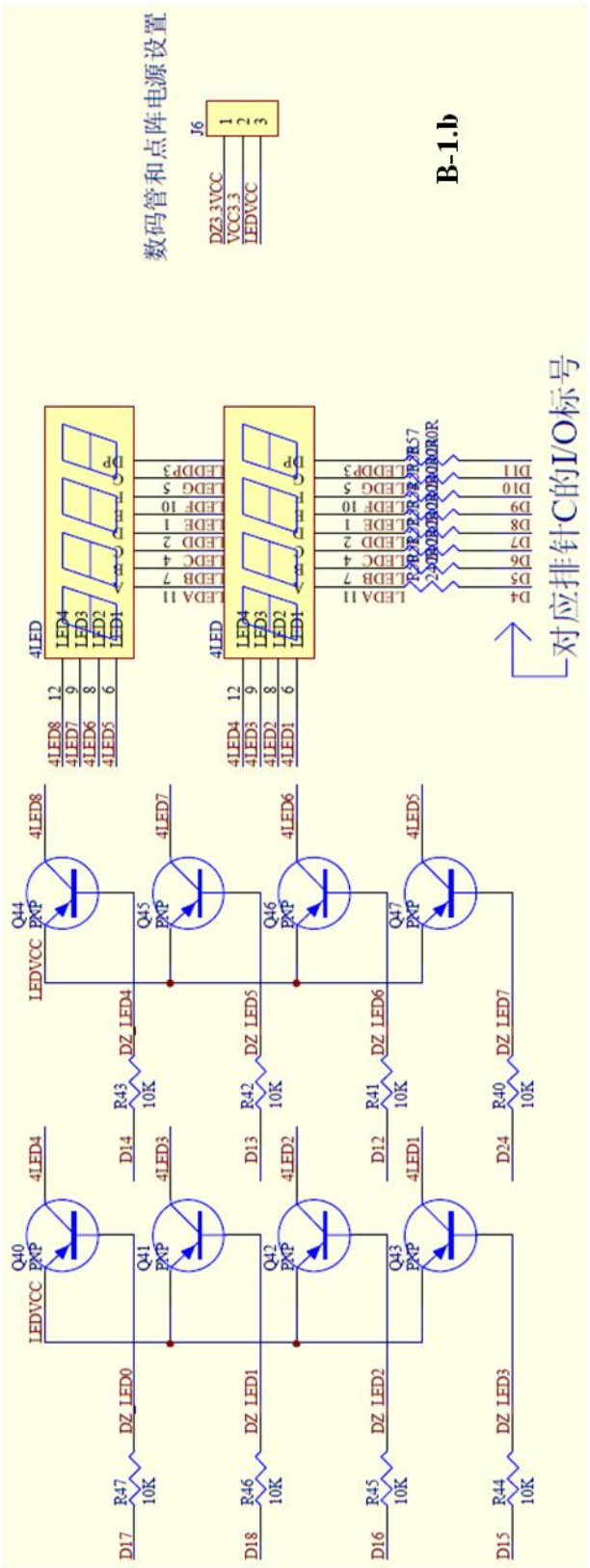
Esquemático de la

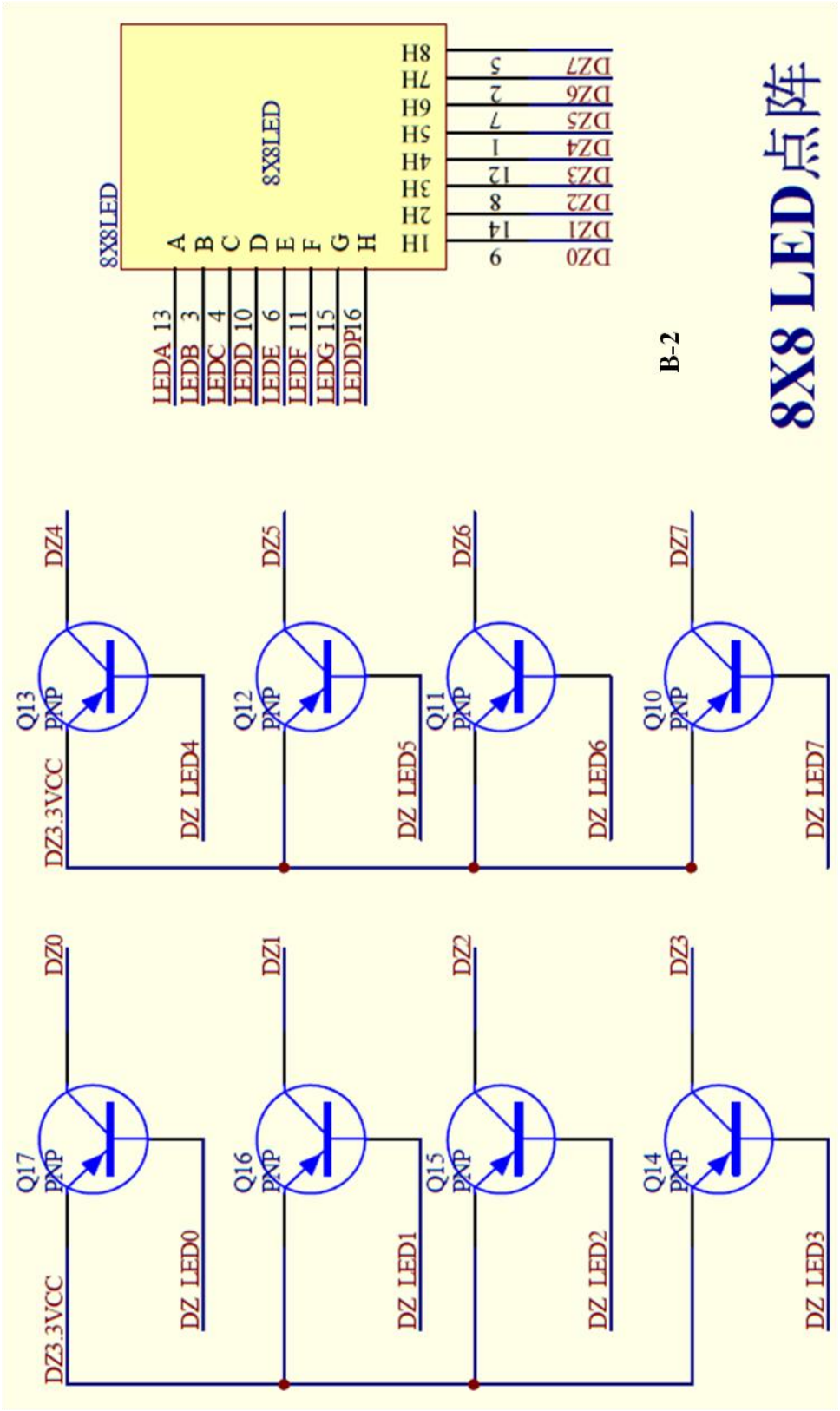
Tarjeta de Desarrollo

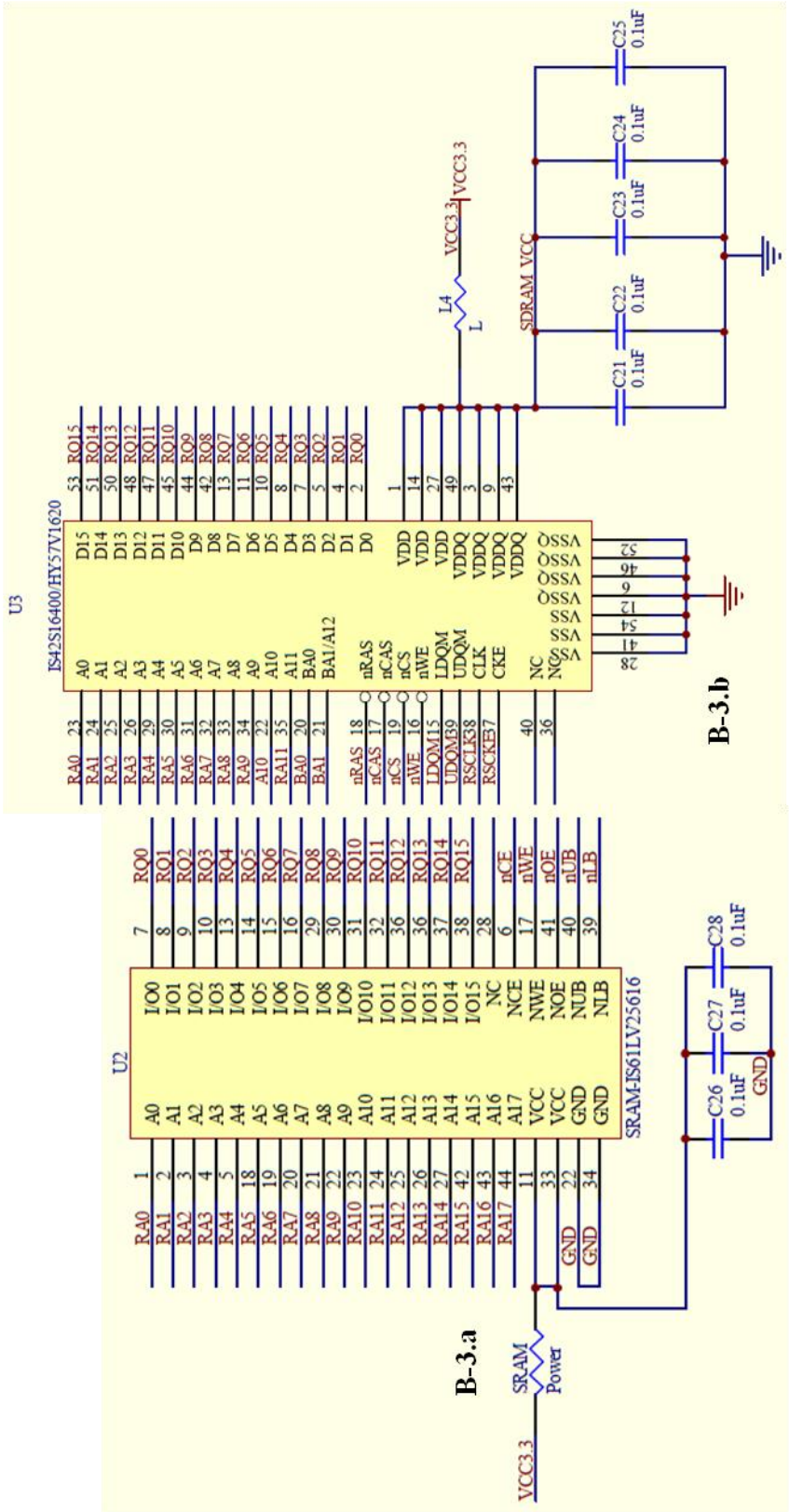

B-1.a
B-1.b
B-2
B-3.a
B-3.b
B-4
XILINX
B-5
B-6
B-7
B-8
B-9
B-10
B-11
B-12
B-13
B-14
B-15
B-16
B-17
B-18
B-19

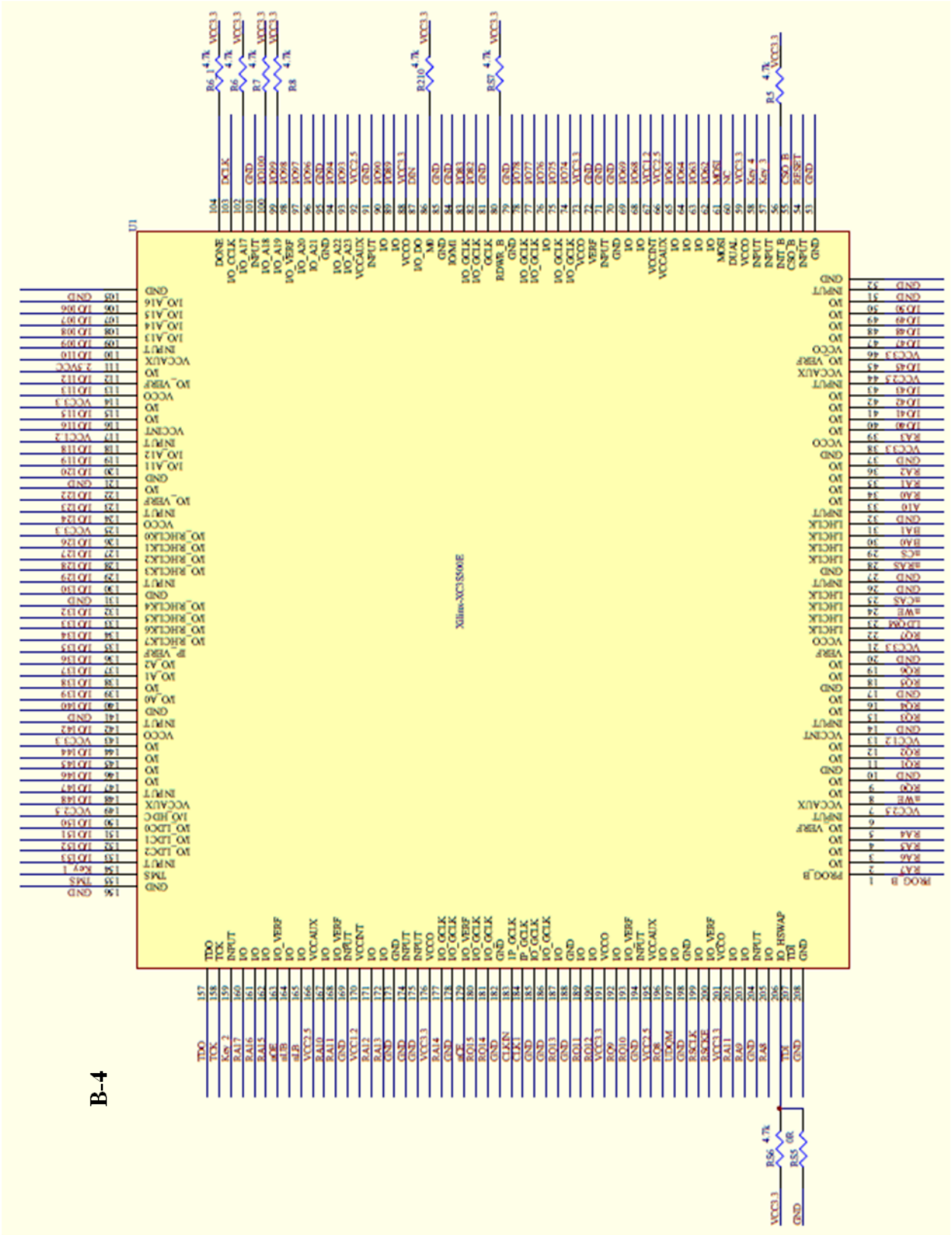
Detalles del Esquemático de la Tarjeta de Desarrollo

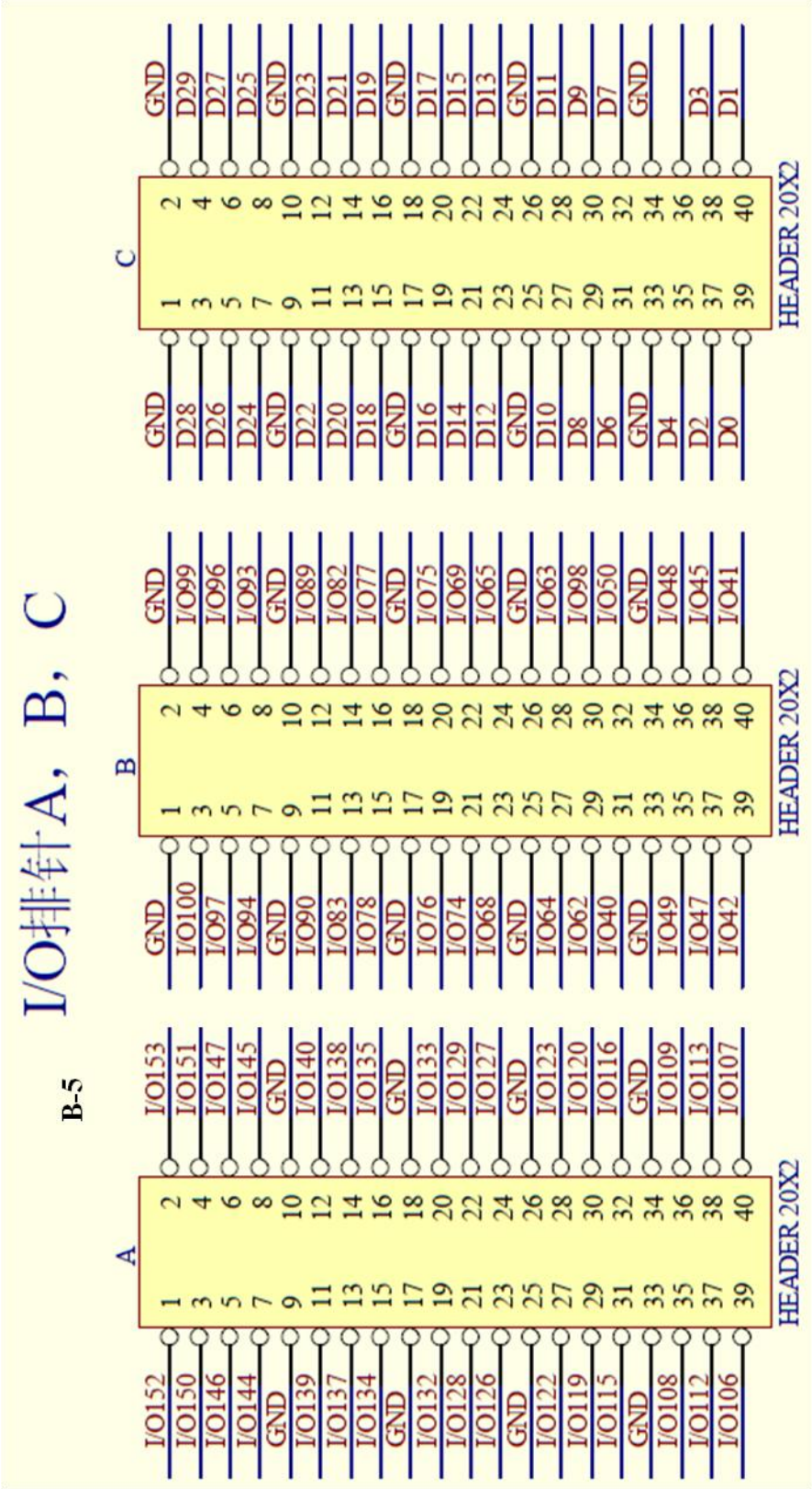

B-1.a

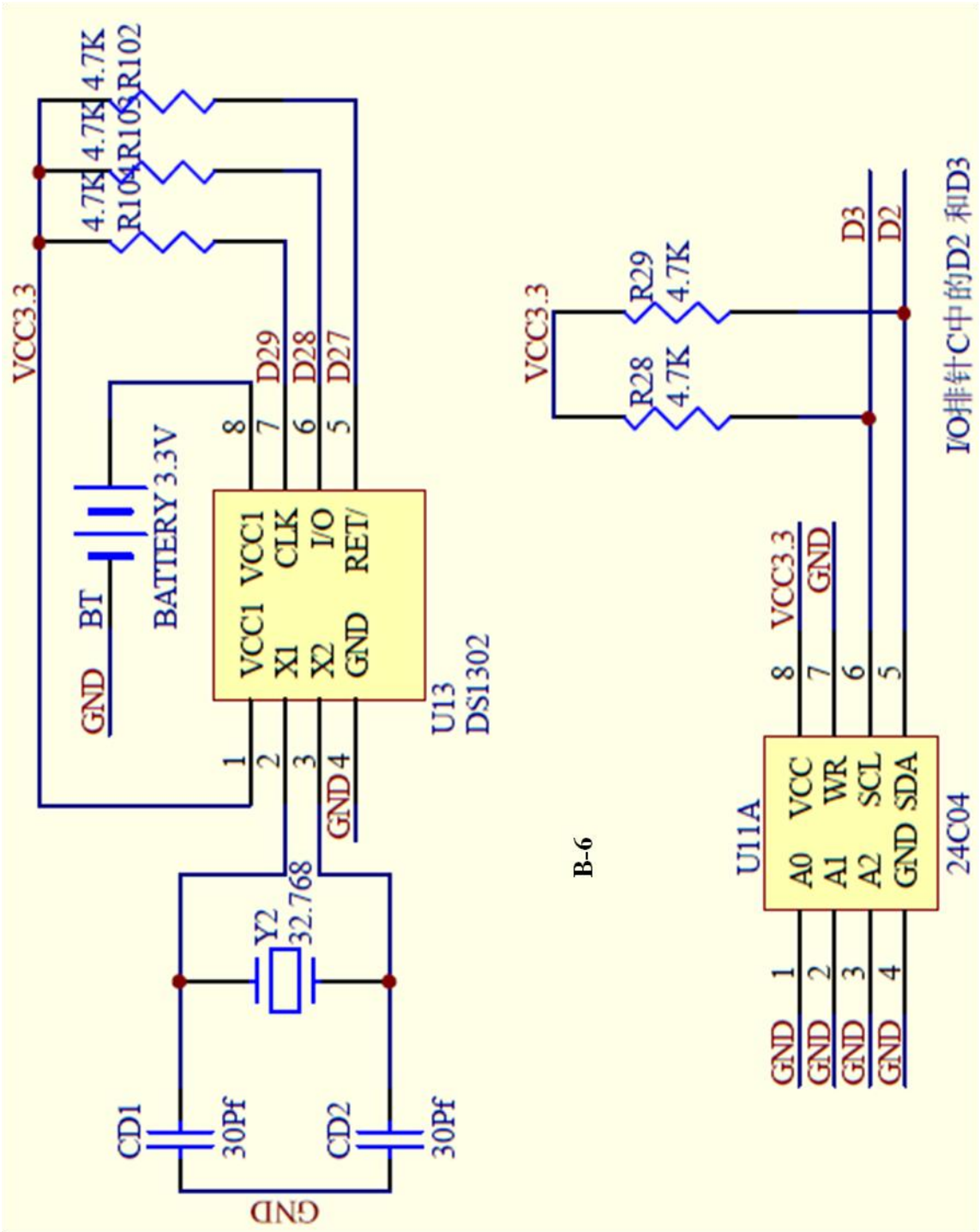

B-1.b
数码管和点阵电源设置
对应排针C的I/O标号

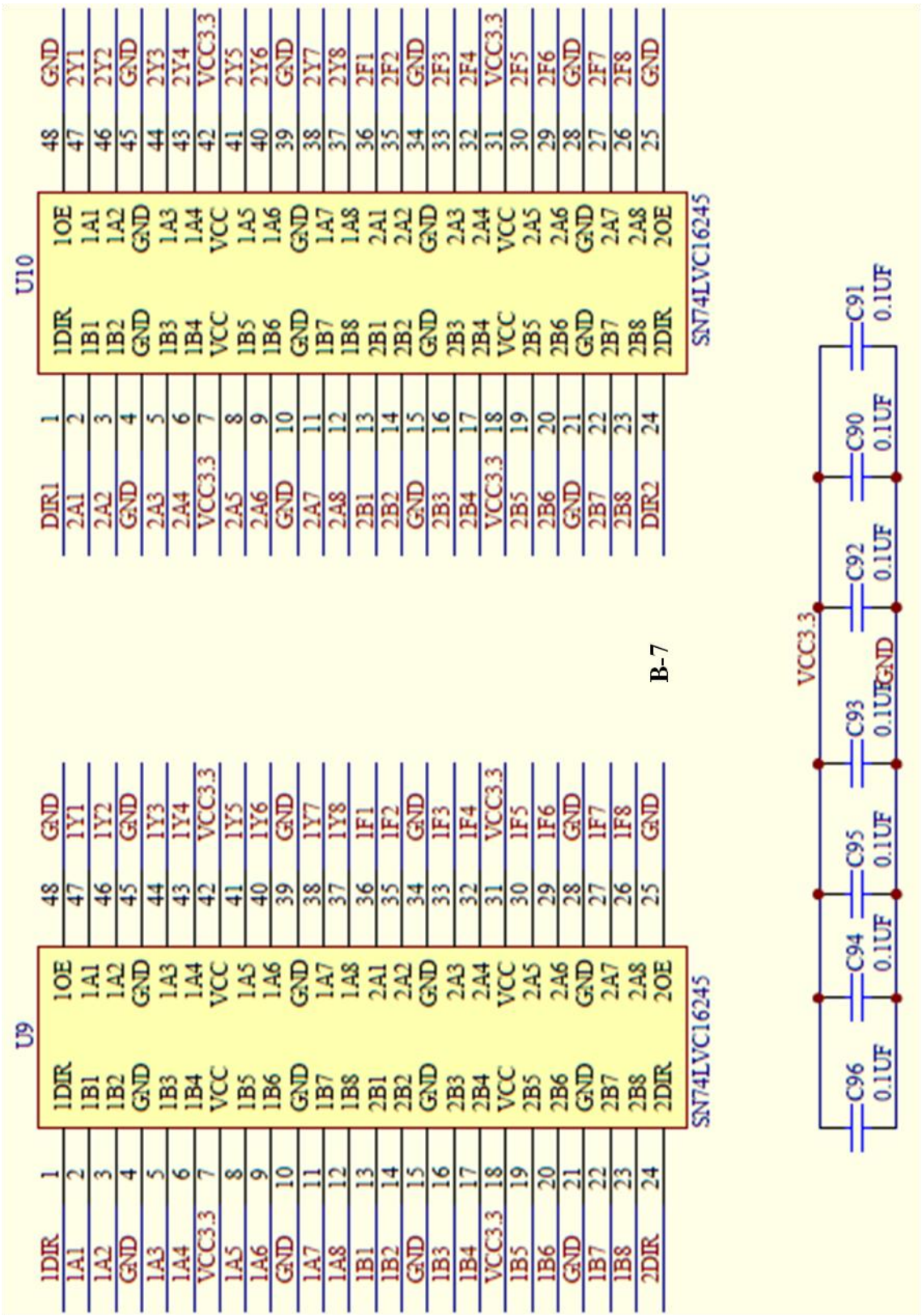

8X8 LED点阵
B-2
8X8LED
8X8LED
A
B
C
D
E
F
G
H
LEDA 13
LEDB 3
LEDC 4
LEDD 10
LEDE 6
LEDF 11
LEDG 15
LEDDP16
1H
2H
3H
4H
5H
6H
7H
8H
DZ0 9
DZ1 14
DZ2 8
DZ3 12
DZ4 1
DZ5 7
DZ6 2
DZ7 5
DZ4
DZ5
DZ6
DZ7
Q13 PNP
Q12 PNP
Q11 PNP
Q10 PNP
DZ3.3VCC
DZ_LED4
DZ_LED5
DZ_LED6
DZ_LED7
DZ0
DZ1
DZ2
DZ3
Q17 PNP
Q16 PNP
Q15 PNP
Q14 PNP
DZ3.3VCC
DZ_LED0
DZ_LED1
DZ_LED2
DZ_LED3

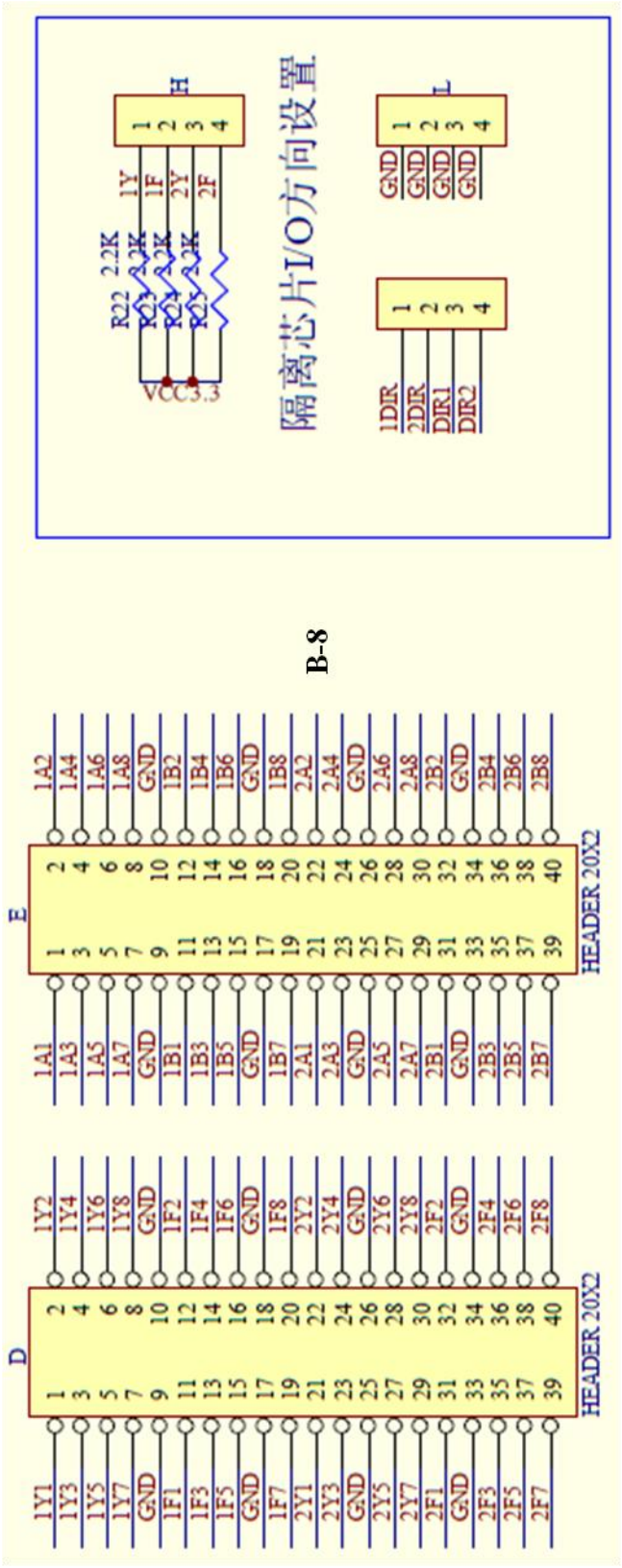

U3
IS42S16400/HY57V1620
B-3.b
U2
SRAM-IS61LV25616
B-3.a
SDRAM VCC
VCC3.3
SRAM
Power
GND

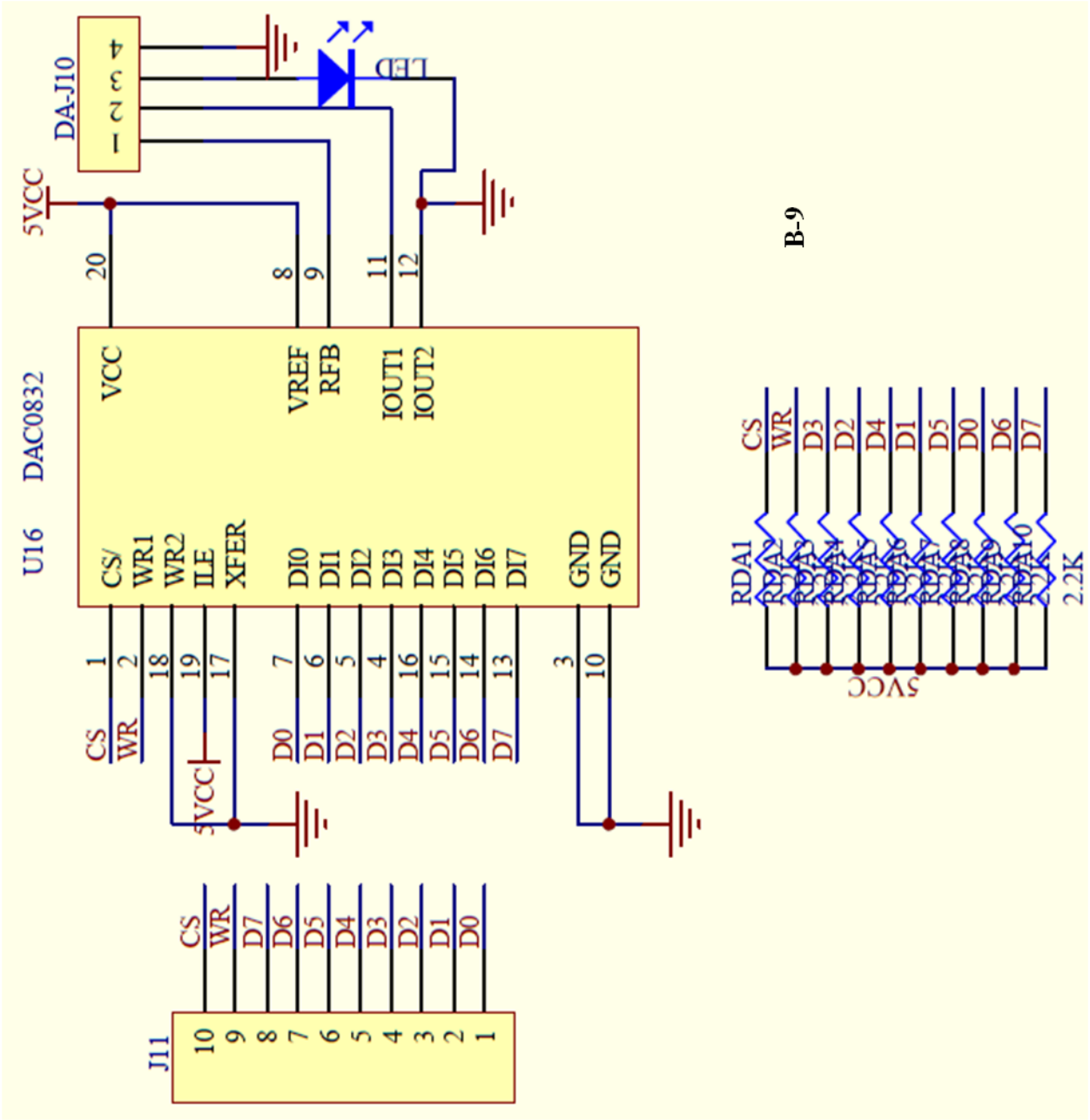

B-4

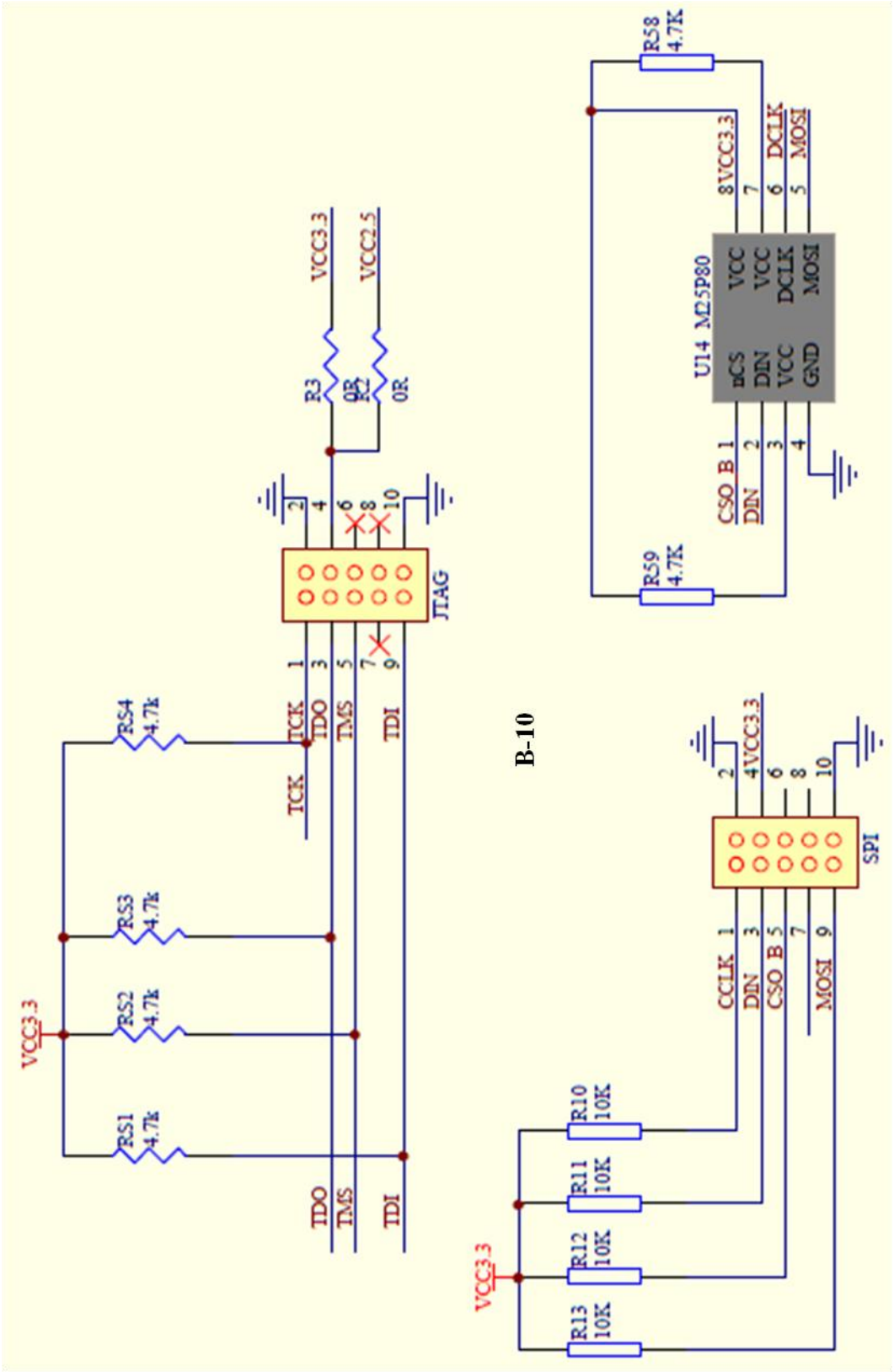

B-5
I/O排针A, B, C
A
B
C
HEADER 20X2
HEADER 20X2
HEADER 20X2
I/O152 I/O150 I/O146 I/O144 GND I/O139 I/O137 I/O134 GND I/O132 I/O128 I/O126 GND I/O122 I/O119 I/O115 GND I/O108 I/O112 I/O106
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
I/O153 I/O151 I/O147 I/O145 GND I/O140 I/O138 I/O135 GND I/O133 I/O129 I/O127 GND I/O123 I/O120 I/O116 GND I/O109 I/O113 I/O107
GND I/O100 I/O97 I/O94 GND I/O90 I/O83 I/O78 GND I/O76 I/O74 I/O68 GND I/O64 I/O62 I/O40 GND I/O49 I/O47 I/O42
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
GND I/O99 I/O96 I/O93 GND I/O89 I/O82 I/O77 GND I/O75 I/O69 I/O65 GND I/O63 I/O98 I/O50 GND I/O48 I/O45 I/O41
GND D28 D26 D24 GND D22 D20 D18 GND D16 D14 D12 GND D10 D8 D6 GND D4 D2 D0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
GND D29 D27 D25 GND D23 D21 D19 GND D17 D15 D13 GND D11 D9 D7 GND D3 D1

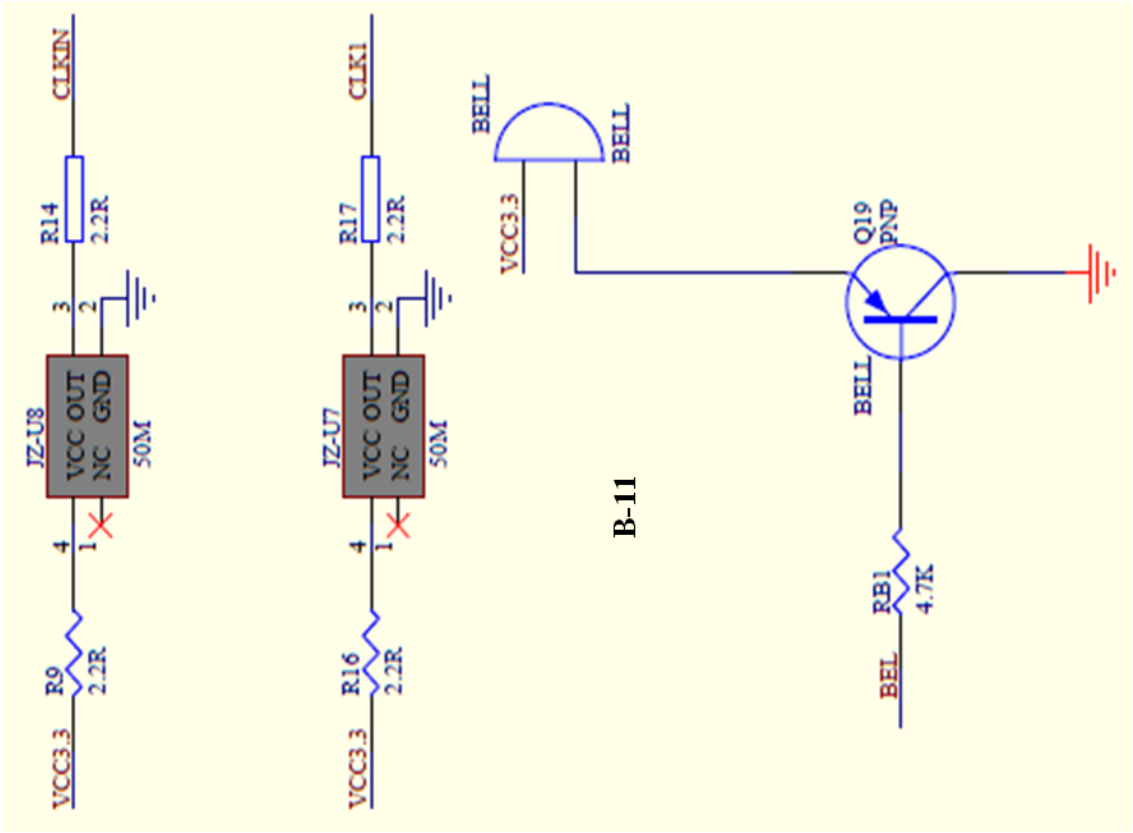

VCC3.3
4.7K
R104
4.7K
R103
4.7K
R102
BT
BATTERY 3.3V
GND
VCC1
X1
X2
GND
VCC1
CLK
I/O
RET/
D29
D28
D27
U13
DS1302
Y2
32.768
CD1
30Pf
CD2
30Pf
GND
B-6
VCC3.3
R29
4.7K
R28
4.7K
D3
D2
I/O排针C中的D2 和D3
U11A
A0
A1
A2
GND
VCC
WR
SCL
SDA
VCC3.3
GND
24C04

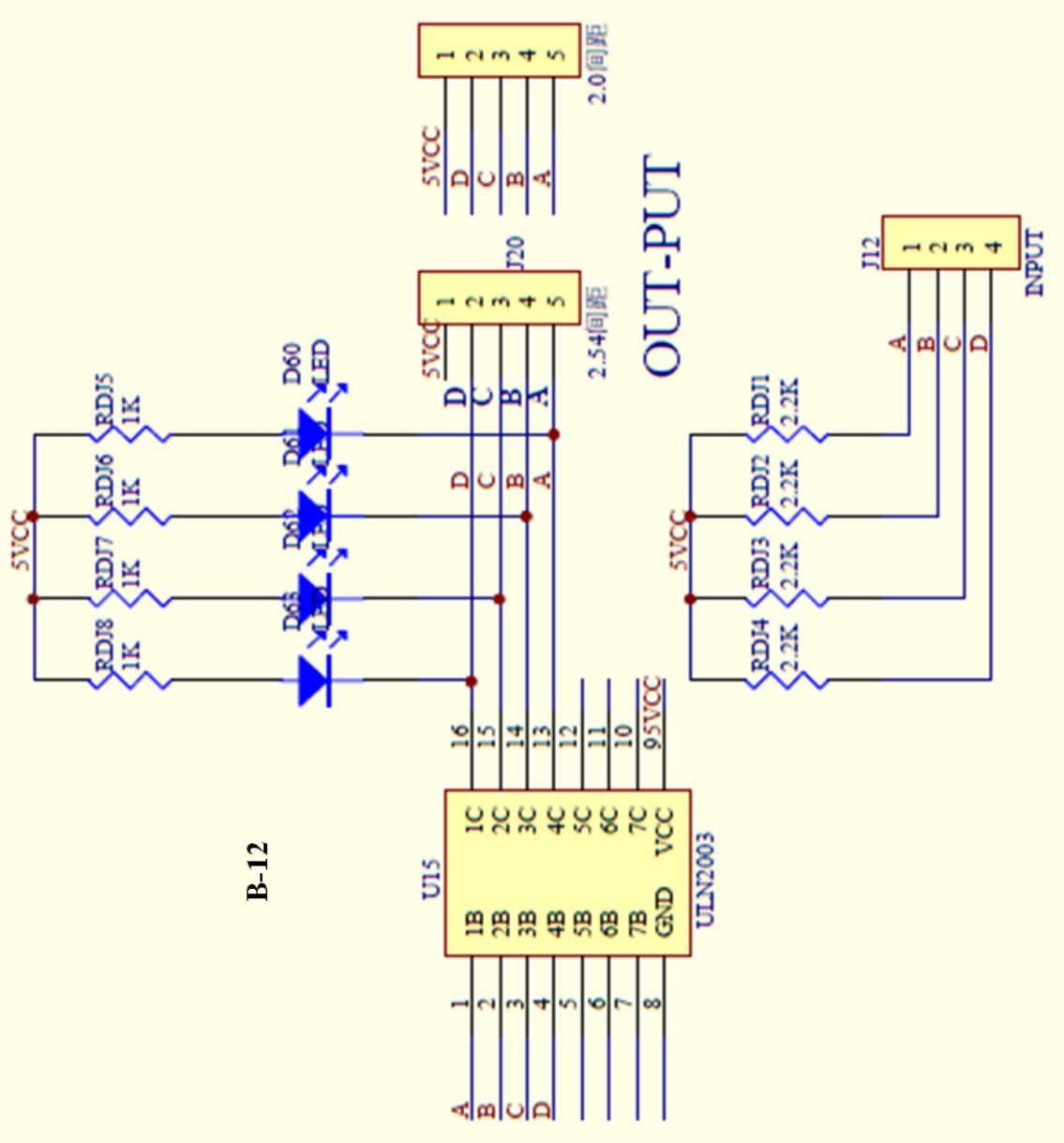

U9
SN74LVC16245
1DIR 1
1A1 2
1A2 3
GND 4
1A3 5
1A4 6
VCC3.3 7
1A5 8
1A6 9
GND 10
1A7 11
1A8 12
1B1 13
1B2 14
GND 15
1B3 16
1B4 17
VCC3.3 18
1B5 19
1B6 20
GND 21
1B7 22
1B8 23
2DIR 24
1DIR 1OE 48 GND
1B1 1A1 47 1Y1
1B2 1A2 46 1Y2
GND GND 45 GND
1B3 1A3 44 1Y3
1B4 1A4 43 1Y4
VCC VCC 42 VCC3.3
1B5 1A5 41 1Y5
1B6 1A6 40 1Y6
GND GND 39 GND
1B7 1A7 38 1Y7
1B8 1A8 37 1Y8
2B1 2A1 36 1F1
2B2 2A2 35 1F2
GND GND 34 GND
2B3 2A3 33 1F3
2B4 2A4 32 1F4
VCC3.3 VCC 31 VCC3.3
2B5 2A5 30 1F5
2B6 2A6 29 1F6
GND GND 28 GND
2B7 2A7 27 1F7
2B8 2A8 26 1F8
2DIR 2OE 25 GND
U10
SN74LVC16245
DIR1 1
2A1 2
2A2 3
GND 4
2A3 5
2A4 6
VCC3.3 7
2A5 8
2A6 9
GND 10
2A7 11
2A8 12
2B1 13
2B2 14
GND 15
2B3 16
2B4 17
VCC3.3 18
2B5 19
2B6 20
GND 21
2B7 22
2B8 23
DIR2 24
1DIR 1OE 48 GND
1B1 1A1 47 2Y1
1B2 1A2 46 2Y2
GND GND 45 GND
1B3 1A3 44 2Y3
1B4 1A4 43 2Y4
VCC VCC 42 VCC3.3
1B5 1A5 41 2Y5
1B6 1A6 40 2Y6
GND GND 39 GND
1B7 1A7 38 2Y7
1B8 1A8 37 2Y8
2B1 2A1 36 2F1
2B2 2A2 35 2F2
GND GND 34 GND
2B3 2A3 33 2F3
2B4 2A4 32 2F4
VCC3.3 VCC 31 VCC3.3
2B5 2A5 30 2F5
2B6 2A6 29 2F6
GND GND 28 GND
2B7 2A7 27 2F7
2B8 2A8 26 2F8
2DIR 2OE 25 GND
B-7
VCC3.3
C96 0.1UF
C94 0.1UF
C95 0.1UF
C93 0.1UF
GND
C92 0.1UF
C90 0.1UF
C91 0.1UF

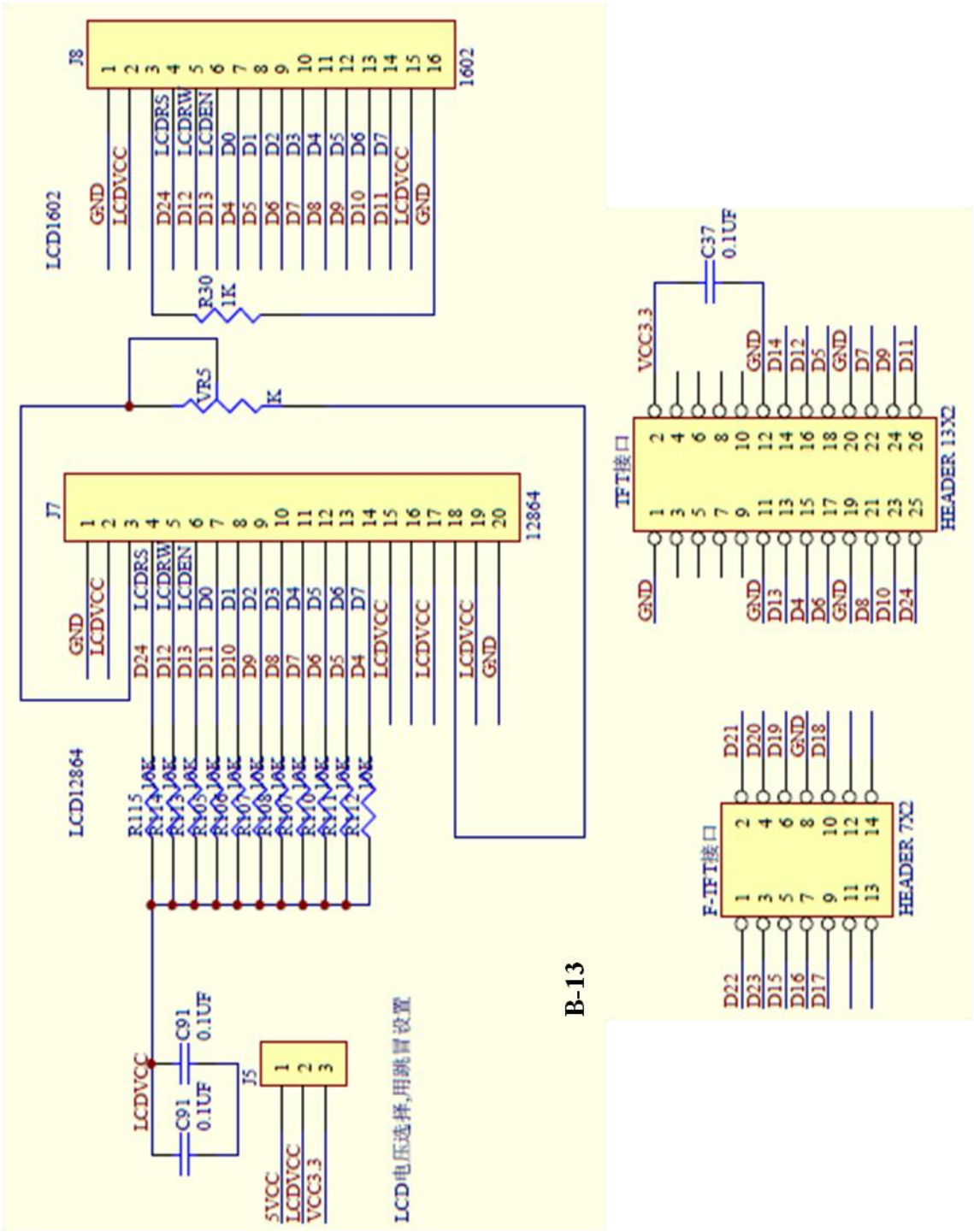

隔离芯片I/O方向设置
H
L
R22 2.2K
R23 2.2K
R24 2.2K
R25 2.2K
1Y
1F
2Y
2F
VCC3.3
GND
1DIR
2DIR
DIR1
DIR2
B-8
D
E
HEADER 20X2

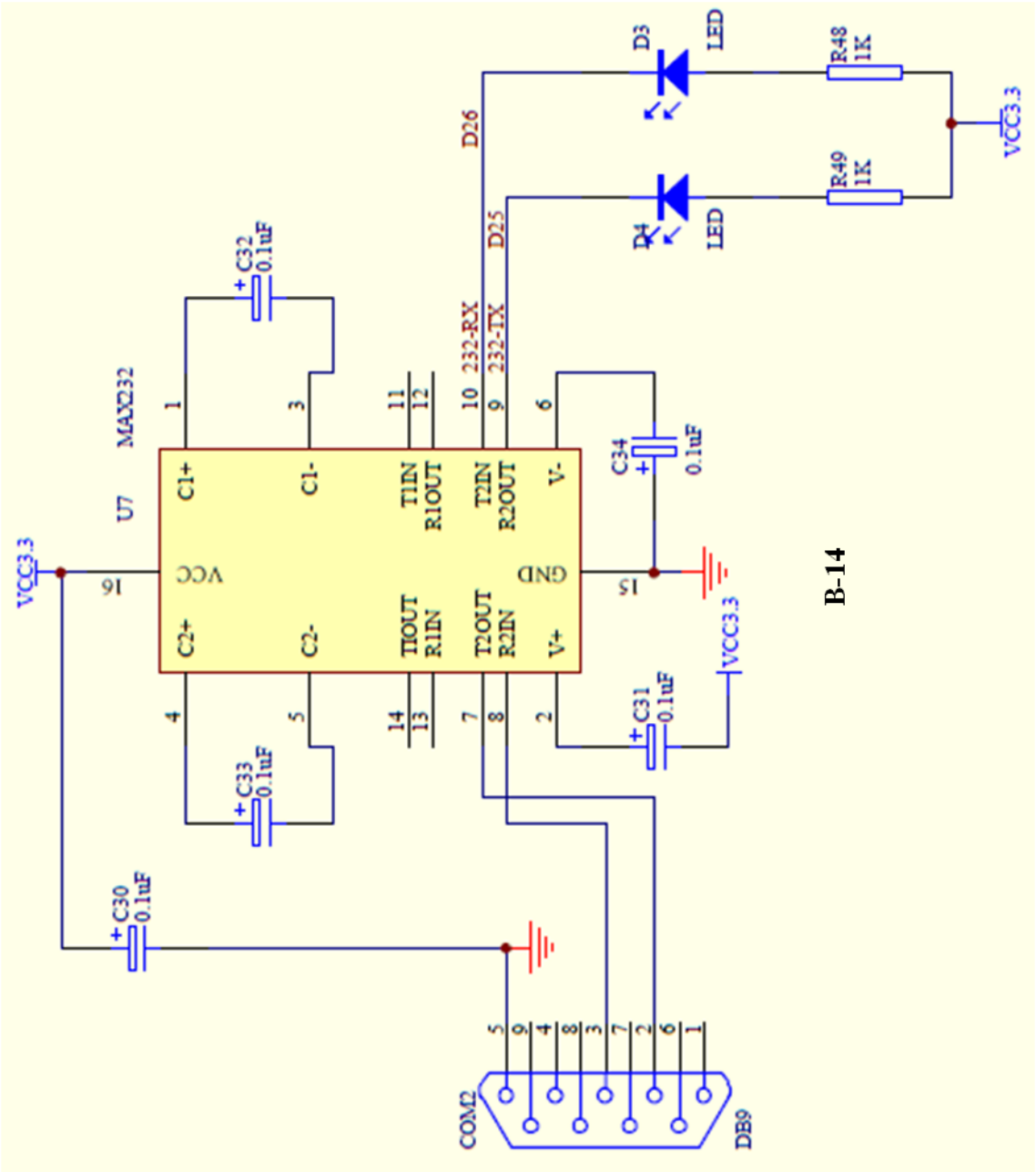

DA-J10
LED
5VCC
U16
DAC0832
VCC
VREF
RFB
IOUT1
IOUT2
CS/
WR1
WR2
ILE
XFER
DI0
DI1
DI2
DI3
DI4
DI5
DI6
DI7
GND
GND
CS
WR
D0
D1
D2
D3
D4
D5
D6
D7
J11
RDA1
2.2K
B-9

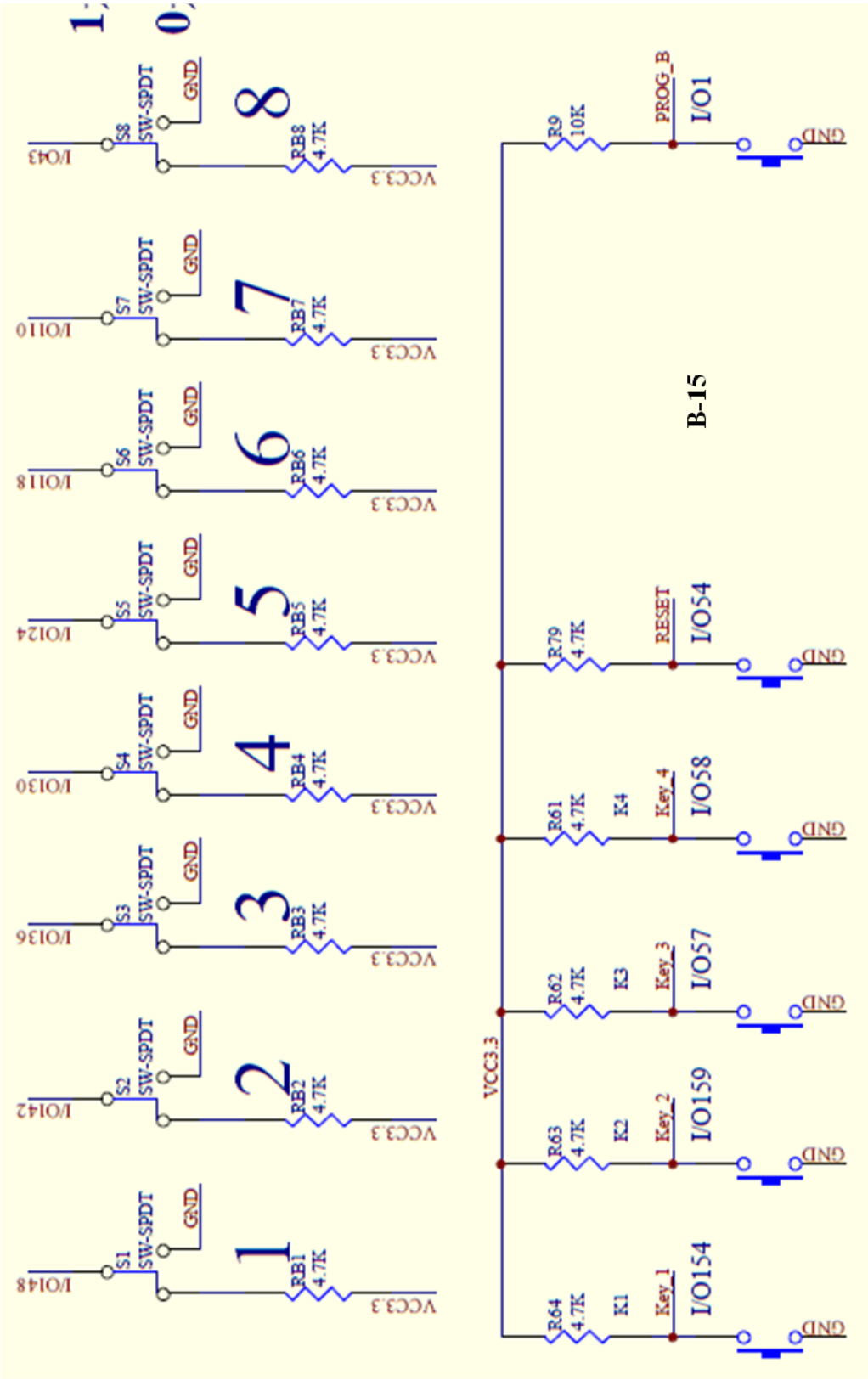

B-10
R58 4.7K
R59 4.7K
U14 M25P80
VCC
VCC
DCLK
MOSI
nCS
DIN
VCC
GND
8VCC3.3
DCLK
MOSI
CSO_B
DIN
VCC3.3
VCC2.5
R3 0R
R2 0R
JTAG
TCK
TDO
TMS
TDI
R54 4.7k
R53 4.7k
R52 4.7k
R51 4.7k
VCC3.3
4VCC3.3
SPI
CCLK
DIN
CSO_B
MOSI
R10 10K
R11 10K
R12 10K
R13 10K
VCC3.3

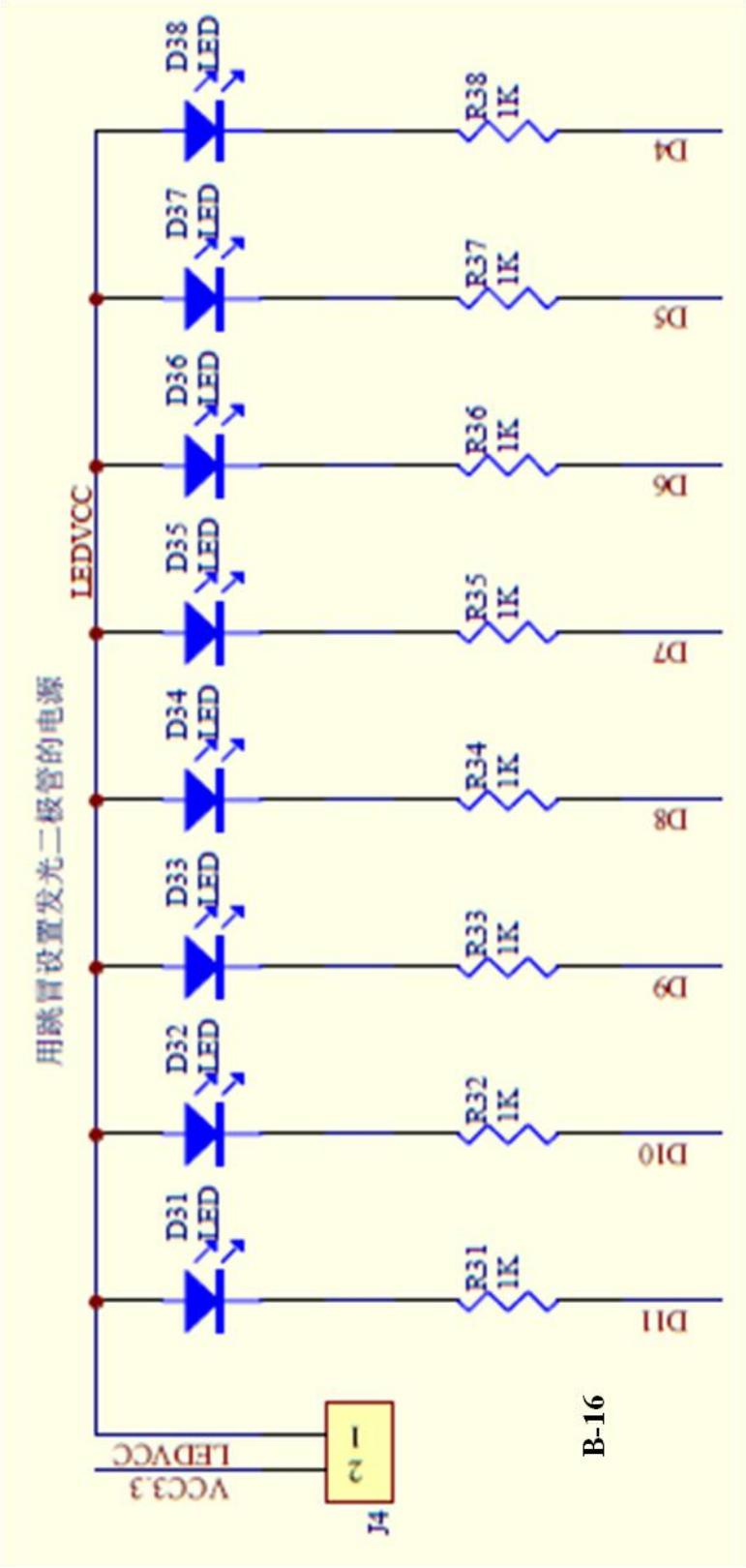

CLKIN
R14
2.2R
3
2
JZ-U8
VCC OUT
NC GND
50M
4
1
R9
2.2R
VCC3.3
CLK1
R17
2.2R
3
2
JZ-U7
VCC OUT
NC GND
50M
4
1
R16
2.2R
VCC3.3
BELL
BELL
VCC3.3
Q19
PNP
BELL
RB1
4.7K
BEL
B-11

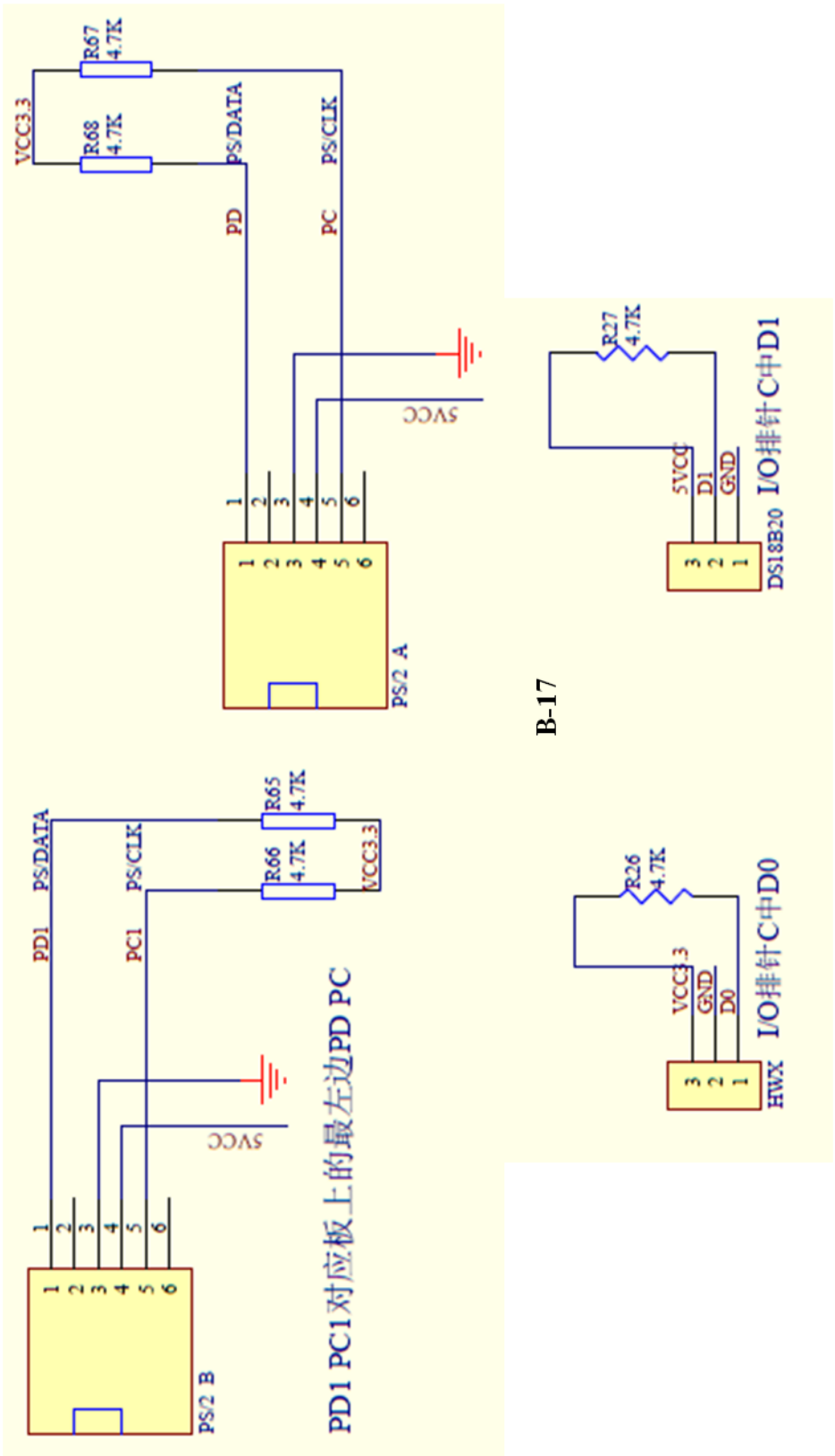

OUT-PUT
INPUT
B-12
5VCC
RDJ5 1K
RDJ6 1K
RDJ7 1K
RDJ8 1K
D60 LED
D61 LED
D62 LED
D63 LED
J20
2.54
J12
RDJ1 2.2K
RDJ2 2.2K
RDJ3 2.2K
RDJ4 2.2K
U15
ULN2003
1C 2C 3C 4C 5C 6C 7C VCC
1B 2B 3B 4B 5B 6B 7B GND

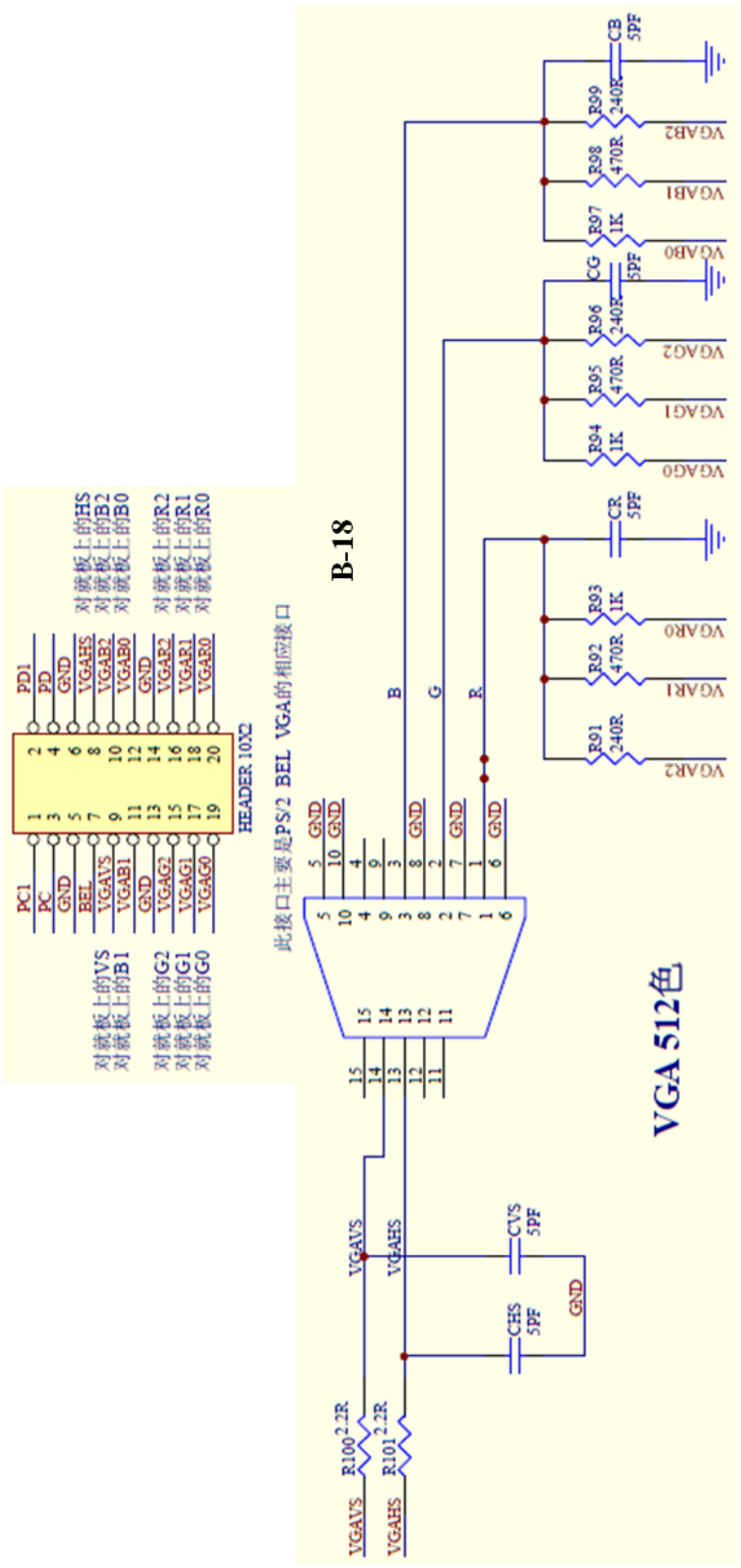

J8
1602
LCD1602
GND
LCDVCC
LCDRS
LCDRW
LCDEN
D0
D1
D2
D3
D4
D5
D6
D7
D24
D12
D13
D11
D10
D9
D8
R30
1K
VR5
K
J7
12864
LCD12864
R115
C91
0.1UF
J5
5VCC
VCC3.3
LCD电压选择,用跳冒设置
C37
IFT接口
HEADER 13X2
F-IFT接口
HEADER 7X2
B-13

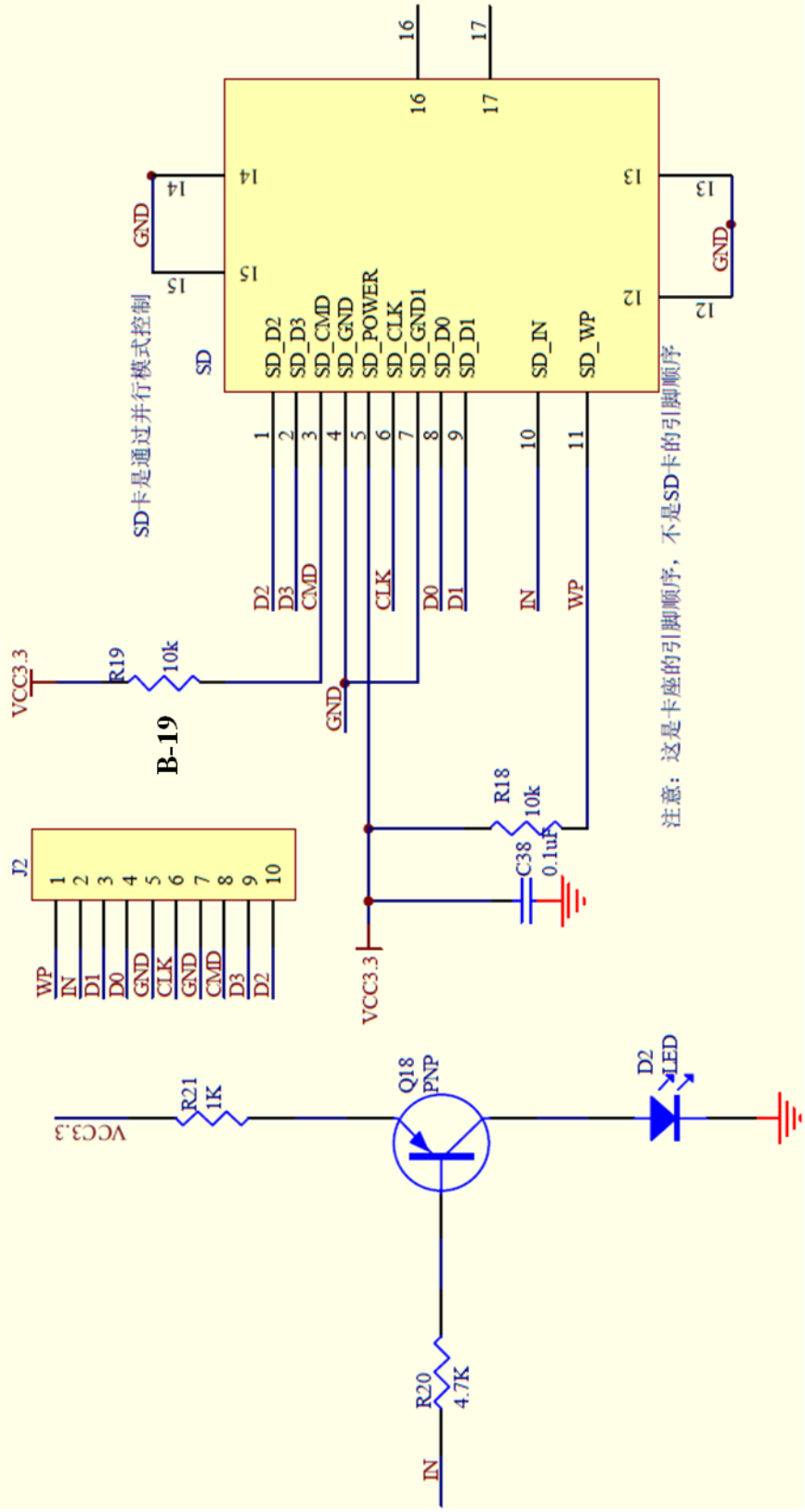

D3
LED
R48
1K
VCC3.3
D26
R49
1K
D4
LED
D25
C32
0.1uF
232-RX
232-TX
U7
MAX232
C1+
C1-
T1IN
R1OUT
T2IN
R2OUT
V-
C34
0.1uF
VCC3.3
VCC
GND
C2+
C2-
T1OUT
R1IN
T2OUT
R2IN
V+
C31
0.1uF
C33
0.1uF
C30
0.1uF
COM2
DB9
B-14


B-15
VCC3.3
GND
SW-SPDT
S1
S2
S3
S4
S5
S6
S7
S8
RB1
RB2
RB3
RB4
RB5
RB6
RB7
RB8
4.7K
I/O148
I/O142
I/O136
I/O130
I/O124
I/O118
I/O110
I/O43
R64
R63
R62
R61
R79
R9
10K
K1
K2
K3
K4
Key_1
Key_2
Key_3
Key_4
RESET
PROG_B
I/O154
I/O159
I/O57
I/O58
I/O54
I/O1


D38 LED
R38 1K
D4
D37 LED
R37 1K
D5
D36 LED
R36 1K
D6
D35 LED
R35 1K
D7
D34 LED
R34 1K
D8
D33 LED
R33 1K
D9
D32 LED
R32 1K
D10
D31 LED
R31 1K
D11
LEDVCC
用跳冒设置发光二极管的电源
VCC3.3
LEDVCC
J4
B-16


R67
4.7K
R68
4.7K
VCC3.3
PS/DATA
PS/CLK
PD
PC
5VCC
PS/2 A
R27
4.7K
5VCC
D1
GND
DS18B20
I/O排针C中D1
B-17
R65
4.7K
R66
4.7K
VCC3.3
PS/DATA
PS/CLK
PD1
PC1
5VCC
PS/2 B
PD1 PC1对应板上的最左边PD PC
R26
4.7K
VCC3.3
GND
D0
HWX
I/O排针C中D0


B-18
VGA 512色
此接口主要是PS/2 BEL VGA的相应接口
HEADER 10X2
VGAVS
VGAHS
R100 2.2R
R101 2.2R
CVS 5PF
CHS 5PF
GND
R99 240R
R98 470R
R97 1K
R96 240R
R95 470R
R94 1K
R93 1K
R92 470R
R91 240R
CB 5PF
CG 5PF
CR 5PF
VGAB2
VGAB1
VGAB0
VGAG2
VGAG1
VGAG0
VGAR0
VGAR1
VGAR2


SD
SD_D2
SD_D3
SD_CMD
SD_GND
SD_POWER
SD_CLK
SD_GND1
SD_D0
SD_D1
SD_IN
SD_WP
SD卡是通过并行模式控制
注意：这是卡座的引脚顺序，不是SD卡的引脚顺序
GND
VCC3.3
R19
10k
R18
10k
C38
0.1uF
R21
1K
Q18
PNP
D2
LED
R20
4.7K
J2
WP
IN
D1
D0
GND
CLK
GND
CMD
D3
D2
B-19

APÉNDICE C:

Programación desarrollada en Visual Basic

APÉNDICE C: PROGRAMACIÓN DESARROLLADA EN VISUAL BASIC.

```
Imports System.IO 'Importa librería que contiene tipos que permiten leer y escribir en los archivos y secuencias de datos, así como tipos que proporcionan compatibilidad básica con los archivos y directorios.
Class Form1
    Dim datagrid As Object 'Variable para asignar un datagrid al proceso de GuardarReporte
    Delegate Sub delegado(ByVal data As Byte, ByVal data1 As Byte) 'Crea delegado con dos entradas
    Delegate Sub delegado2() 'Crea delegado sin entradas
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        'Configuración de Puerto Serial (sp1)
        sp1.BaudRate = 9600
        sp1.DataBits = 8
        sp1.StopBits = IO.Ports.StopBits.Two
        sp1.Parity = IO.Ports.Parity.None
        sp1.ReadBufferSize = 6912
        'Inicialización de controles
        Call inicializacion()
        'Carga datos iniciales en los datagrid existentes al abrir la Aplicación
        Call cargardatos() ' datagrid General y Actualización
        Call cargardisponibles() 'datagrid Disponibles
        'Carga items en Combobox Buscar por (sub-pestaña Consulta General) y Combobox Código (Pestaña Editar Registros)
        Call itemscombobuscar()
        Call itemscombocodigo()
    End Sub
    Sub inicializacion()
        CBbuscarpor.TabStop = False
        CbCodigoActualizar.TabStop = False
        CbCodigoEstado.TabStop = False
        'Estado de Botones en Pantalla Principal
        btnInventarios.Enabled = False
        btnEditar.Enabled = False
        btnSalir.Enabled = True
        btnIniciar.Enabled = True
        'Propiedad Visible de los TabControl (Inventarios, Editar Registros, Principal)
        TabControlprincipal.Visible = True
        TabControlInventarios.Visible = False
        TabControlEdicion.Visible = False
        'Datagridview solo lectura
        DataGridView1.ReadOnly = True
        DataGridView2.ReadOnly = True
        DataGridView3.ReadOnly = True
        'Textbox solo lectura de sub-pestaña Cambiar Estado
        TxtDesEstado.ReadOnly = True
        'Groupbox de pestaña Editar Registros inhabilitado
        GBactualizar.Enabled = False
        GBcambioestado.Enabled = False
        'Estado botón Aplicar sub-pestaña Cambiar Estado
        btnAplicar.Enabled = False
        'Radiobotones de pestaña Editar Registros inhabilitados y no marcados
        RbActivo.Enabled = False
        RbNoactivo.Enabled = False
        RbActivo.Checked = False
```

```
        RbNoactivo.Checked = False
    End Sub
    Sub cargardatos()
        Dim rutaDir As String = "c:\BaseInventario" 'Ruta del directorio donde está archivo a cargar en datagrid
        Dim ruta As String = "c:\BaseInventario\" & "BaseDatos.txt" 'Ruta archivo a cargar en datagrid

        If My.Computer.FileSystem.DirectoryExists(rutaDir) Then 'Verifica si existe directorio
        Else
            Directory.CreateDirectory(rutaDir) 'Crea directorio si no existe
            Dim folder As DirectoryInfo = My.Computer.FileSystem.GetDirectoryInfo(rutaDir)
            folder.Attributes = FileAttributes.Hidden 'Oculta directorio
        End If
        If My.Computer.FileSystem.FileExists(ruta) Then 'Verifica si existe archivo .txt a cargar en datagrid
            Dim objReader As New StreamReader(ruta) 'Crea objeto para lectura de archivo
            Dim count As Integer = 6 'número de columnas de datagrid General
            Dim count1 As Integer = 5 'número de columnas de datagrid Actualización
            Dim texto As String = " "
            Dim split As String() = Nothing
            Dim split1 As String() = Nothing
            For linea As Integer = 0 To 39 'Ciclo para leer cadenas de cada línea del archivo y cargar en datagrid
General
                texto = objReader.ReadLine 'Lee una línea del archivo
                split = texto.Split(New Char() {";"}, count) 'Separa línea leída en subcadenas delimitadas por ";"
                'Carga de los datos leídos (subcadenas) en datagrid
                If split(3) = "" Then
                    DataGridView1.Rows.Add(split(0), split(1), split(2), split(3), split(4), split(5))
                Else
                    DataGridView1.Rows.Add(split(0), split(1), split(2), CDec(split(3)), split(4), split(5))
                End If
            Next
            For linea As Integer = 0 To 39 'Ciclo para leer cadenas de las líneas restantes del archivo y cargar en
datagrid Actualización
                texto = objReader.ReadLine()
                split1 = texto.Split(New Char() {";"}, count1) 'Separa línea leída en subcadenas delimitadas por ";"
                DataGridView3.Rows.Add(split1(0), split1(1), split1(2), split1(3), split1(4)) 'Carga en datagrid los
datos (subcadenas) del archivo leído
            Next
            objReader.Close() 'Cierra archivo
            For fila As Integer = 0 To DataGridView1.RowCount - 1 'Ciclo para colocar en amarillo filas de
registro en estado"No activo" de datagrid
                If DataGridView1.Rows(fila).Cells(5).Value = "No Activo" Then
                    DataGridView1.Rows(fila).DefaultCellStyle.BackColor = Color.Yellow
                    DataGridView3.Rows(fila).DefaultCellStyle.BackColor = Color.Yellow
                End If
            Next
        Else
            Try
                System.IO.File.Create(ruta) ' Crea un archivo vacío .txt si no existe
                ' error
            Catch ex As Exception
                MsgBox(ex.Message, MsgBoxStyle.Critical)
            End Try
```

```
        'Carga filas iniciales en datagrid General
        For fila = 0 To 39
            DataGridView1.Rows.Add()
            DataGridView1.Rows(fila).Cells(0).Value = Nothing
            DataGridView1.Rows(fila).Cells(1).Value = 0
            DataGridView1.Rows(fila).Cells(2).Value = Nothing
            DataGridView1.Rows(fila).Cells(3).Value = Nothing
            DataGridView1.Rows(fila).Cells(4).Value = Nothing
            DataGridView1.Rows(fila).Cells(5).Value = "Activo"
        Next
        'Carga filas iniciales en datagrid Actualización
        For fila = 0 To 39
            DataGridView3.Rows.Add()
            DataGridView3.Rows(fila).Cells(0).Value = Nothing
            DataGridView3.Rows(fila).Cells(1).Value = Nothing
            DataGridView3.Rows(fila).Cells(2).Value = Nothing
            DataGridView3.Rows(fila).Cells(3).Value = Nothing
            DataGridView3.Rows(fila).Cells(4).Value = Nothing
        Next
    End If
End Sub
Sub cargardisponibles() 'Proceso para carga de datos en datagrid Disponibles
    DataGridView2.Rows.Clear() 'Limpia datagrid Disponibles
    Dim numrow As Integer = DataGridView1.RowCount 'Número filas datagrid General
    Dim fila As Integer = 0 'Variable de control de filas en datagrid Disponibles
    'Ciclo para cargar en datagrid Disponibles sólo los registros en estado "Activo" del datagrid General
    For count = 0 To numrow - 1
        If DataGridView1.Rows(count).DefaultCellStyle.BackColor <> Color.Yellow Then
            DataGridView2.Rows.Add() 'Agrega fila en datagrid Disponibles
            DataGridView2.Rows(fila).Cells(0).Value = DataGridView1.Rows(count).Cells(0).Value
            DataGridView2.Rows(fila).Cells(1).Value = DataGridView1.Rows(count).Cells(1).Value
            DataGridView2.Rows(fila).Cells(2).Value = DataGridView1.Rows(count).Cells(2).Value
            DataGridView2.Rows(fila).Cells(3).Value = DataGridView1.Rows(count).Cells(3).Value
            DataGridView2.Rows(fila).Cells(4).Value = DataGridView1.Rows(count).Cells(4).Value
            fila = fila + 1
        End If
    Next
End Sub
Sub itemscombocodigo()
    Dim item(47) As String 'Crea vector que contendrá los items del combobox
    ' Ciclo para cargar los items(códigos) en vector item
    For cod As Integer = 0 To 47
        item(cod) = (cod + 1).ToString
    Next
    'Adición del vector item a combobox
    CbCodigoEstado.Items.AddRange(item)
    CbCodigoActualizar.Items.AddRange(item)
    CbCodigoEstado.DropDownStyle = ComboBoxStyle.DropDownList
    CbCodigoActualizar.DropDownStyle = ComboBoxStyle.DropDownList
End Sub
Sub itemscombobuscar()
    Dim forma(1) As String 'Crea vector que contendrá los items del combobox
```

```
        'Carga de items en vector forma
        forma(0) = "Código"
        forma(1) = "Descripción"
        'Adición del vector forma a combobox
        CBbuscarpor.Items.AddRange(forma)
        CBbuscarpor.DropDownStyle = ComboBoxStyle.DropDownList
    End Sub
    Private Sub btnIniciar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnIniciar.Click 'Proceso al hacer click en botón "Iniciar" (btnIniciar)
        If btnIniciar.Text = "Iniciar" Then  'Si se presiona para iniciar
            btnIniciar.Text = "Finalizar" 'Cambia texto en btnIniciar
            btnIniciar.Image = SCI.My.Resources.Resources.fin38 'Cambia imagen en btnIniciar
            Try
                sp1 = My.Computer.Ports.OpenSerialPort("COM1") 'Abre puerto COM1
                sp1.DiscardInBuffer() 'Limpia buffer entrada
                sp1.Write("@") 'Coloca parámetro "@" en buffer de salida
            Catch ex As Exception 'Error
                MessageBox.Show("Error de conexión")
            End Try
            'Visible TabControl de Inventarios y de Editar Registros
            TabControlInventarios.Visible = True
            TabControlEdicion.Visible = True
            'Habilita botones "Inventarios" y "Editar Registro" (Pantalla Principal)
            btnInventarios.Enabled = True
            btnEditar.Enabled = True
        ElseIf btnIniciar.Text = "Finalizar" Then  'Si se presiona para finalizar
            btnIniciar.Text = "Iniciar" 'Cambia texto en btnIniciar
            btnIniciar.Image = SCI.My.Resources.Resources.inicio 'Cambia imagen en btnIniciar
            Try
                sp1.Write("?") 'Coloca parámetro "?" en buffer de salida
                'Limpia el buffer
                sp1.DiscardOutBuffer()
                sp1.DiscardInBuffer()
                sp1.Close() 'Cierra puerto
            Catch ex As Exception
                MessageBox.Show(ex.Message)
            End Try
            ' No Visible TabControl de Inventarios y de Editar Registros
            TabControlInventarios.Visible = False
            TabControlEdicion.Visible = False
            'Inhablita botones "Inventarios" y "Editar Registros" (Pantalla Principal)
            btnInventarios.Enabled = False
            btnEditar.Enabled = False
        End If
    End Sub
    Private Sub sp1_DataReceived_1(ByVal sender As System.Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles sp1.DataReceived 'Proceso a ejecutar cuando se
reciben datos por el puerto serial (sp1)

        Dim count As Integer
        Dim codproducto As Byte 'Variable que almacena  byte de código
        Dim cantproducto As Byte 'Variable que almacena byte de cantidad
```

```
        'Ciclo para leer los bytes en el buffer de entrada
        For count = 0 To 39
            If sp1.BytesToRead = 0 Then
                Exit For
            Else
                codproducto = sp1.ReadByte 'Lee byte de código de producto
                cantproducto = sp1.ReadByte 'Lee byte de cantidad de producto
                Dim metodo As New delegado(AddressOf Me.mostrar) 'Asocia el método "mostrar" al delegado
(Apunta el delegado al método "mostrar")
                Me.Invoke(metodo, codproducto, cantproducto) 'Invoca al método asociado (mostrar) a través del
delegado
            End If
        Next
        'Se invoca proceso "cargardisponibles" a través del delegado para actualizar datagrid Disponibles
        Dim metodo1 As New delegado2(AddressOf Me.cargardisponibles)
        Me.Invoke(metodo1)
    End Sub
    Sub mostrar(ByVal data As Byte, ByVal data1 As Byte) 'Proceso para cargar en datagrid (General y
Actualización) los datos leídos en el puerto serial
        If CbCodigoActualizar.SelectedItem = data Then
            LbCantActualizar.Text = data1
        End If
        'Actualiza datagrid General y datagrid Actualización cada dos lecturas de un byte en el buffer de entrada
        lbhoraActualización.Text = Date.Now.ToLongTimeString
        'Condición para actualizar solo registros en estado "Activo"
        If DataGridView1.Rows(data - 1).Cells(5).Value = "No Activo" Then
        Else
            If data1 <> DataGridView1.Rows(data - 1).Cells(1).Value Then 'Condición para actualizar en datagrid
Actualización solo los registros con cambio en la cantidad (data1)
                NotifyIcon1.ShowBalloonTip(3000, "Información", "Se Actualizó el Inventario", ToolTipIcon.Info)
'Un icono muestra un balón de información en barra de tareas del ordenador (indica que se actualizó el
inventario)
                DataGridView1.Rows(data - 1).DefaultCellStyle.BackColor = Color.MediumTurquoise 'Cambia
color fila datagrid General
                DataGridView3.Rows(data - 1).DefaultCellStyle.BackColor = Color.MediumTurquoise 'Cambia
color fila datagrid Actualización
                DataGridView3.Rows(data - 1).Cells(0).Value = data
                DataGridView3.Rows(data - 1).Cells(1).Value = DataGridView1.Rows(data - 1).Cells(2).Value
                DataGridView3.Rows(data - 1).Cells(2).Value = DataGridView1.Rows(data - 1).Cells(1).Value
                DataGridView3.Rows(data - 1).Cells(3).Value = data1
                DataGridView3.Rows(data - 1).Cells(4).Value = Date.Now
            Else
                'Devuelve a color original filas de registros sin cambio en cantidad (data1)
                DataGridView1.Rows(data - 1).DefaultCellStyle.BackColor = Color.White
                DataGridView3.Rows(data - 1).DefaultCellStyle.BackColor = Color.White
            End If
                ' Actualiza registros en datagrid General
                DataGridView1.Rows(data - 1).Cells(0).Value = data
                DataGridView1.Rows(data - 1).Cells(1).Value = data1
                DataGridView1.Rows(data - 1).Cells(4).Value = Date.Now.ToLongTimeString
        End If
    End Sub
```

```
    Private Sub CBbuscarpor_SelectedIndexChanged_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CBbuscarpor.SelectedIndexChanged 'Proceso que se ejecuta al cambiar item
seleccionado en combobox de Buscar por en datagrid General
        'Llama a proceso de búsqueda de acuerdo al item seleccionado
        If CBbuscarpor.SelectedItem = "Código" Then
            Call buscarporcodigo()
        ElseIf CBbuscarpor.SelectedItem = "Descripción" Then
            Call buscarpornombre()
        End If
    End Sub
    Sub buscarporcodigo() 'Proceso de búsqueda por Código en datagrid General
        Dim x As Integer
        Dim buscado As String
        buscado = Trim(TxtBuscar.Text)
        If buscado = "" Then
            MsgBox("Escriba Código de Producto a buscar", MsgBoxStyle.Exclamation, "Campo Vacío")
            CBbuscarpor.SelectedIndex = -1
        Else
            If IsNumeric(TxtBuscar.Text) Then
                Try
                    x = 0
                    While buscado <> DataGridView1.Item(0, x).Value.ToString
                        x += 1
                    End While
                    DataGridView1.Rows(x).Selected = True
                    DataGridView1.Rows(x).Cells(0).Selected = True
                    CBbuscarpor.SelectedIndex = -1
                    TxtBuscar.Clear()
                Catch
                    MsgBox("Código no existe en el Inventario", MsgBoxStyle.Information, "Registro no
encontrado")
                    DataGridView1.ClearSelection()
                    CBbuscarpor.SelectedIndex = -1
                    TxtBuscar.Clear()
                End Try
            Else
                MsgBox("Debe escribir solo números", MsgBoxStyle.Critical, "Error")
                TxtBuscar.Clear()
                DataGridView1.ClearSelection()
                CBbuscarpor.SelectedIndex = -1
            End If
        End If
    End Sub
    Sub buscarpornombre() 'Proceso de búsqueda por Descripción en datagrid General
        Dim x As Integer
        Dim buscado As String
        buscado = Trim(TxtBuscar.Text)
        If buscado = "" Then
            MsgBox("Escriba Descripción del Producto a buscar", MsgBoxStyle.Exclamation, "Operación No
Válida")
            CBbuscarpor.SelectedIndex = -1
        Else
```

```
        Try
            x = 0
            While buscado <> Trim(DataGridView1.Item(2, x).Value)
                x += 1
            End While
            DataGridView1.Rows(x).Selected = True
            DataGridView1.Rows(x).Cells(2).Selected = True
            CBbuscarpor.SelectedIndex = -1
            TxtBuscar.Clear()
        Catch
            MsgBox("No existe Descripción de Producto en el Inventario", MsgBoxStyle.Information,
"Registro no encontrado")
            DataGridView1.ClearSelection()
            CBbuscarpor.SelectedIndex = -1
            TxtBuscar.Clear()
        End Try
    End If
  End Sub
  Private Sub btnBuscarDisp_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnBuscarDisp.Click 'Llama al proceso de búsqueda para buscar en datagrid Disponibles al hacer
click en botón Buscar (btnBuscarDisp)
    datagrid = DataGridView2
    Call Buscar()
  End Sub
  Private Sub btnBuscarActu_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnBuscarActu.Click
    'Llama al proceso de búsqueda para buscaren datagrid Actualización al hacer click en botón Buscar
(btnBuscarActu)
    datagrid = DataGridView3
    Call Buscar()
  End Sub
  Sub Buscar() 'Proceso de búsqueda de registros en datagrid Disponibles o datagrid Actualización
    Dim cod As String
    Dim x As Integer
    Do
      cod = InputBox("Ingrese Código de Producto(Sólo Números):", "Buscar Registro")
      If cod = "" Then Exit Sub
    Loop Until IsNumeric(cod)
    Try
      x = 0
      While cod <> datagrid.Item(0, x).Value.ToString
        x += 1
      End While
      datagrid.Rows(x).Selected = True
      datagrid.Rows(x).Cells(0).Selected = True
    Catch
      MsgBox("Código no existe en este Inventario", MsgBoxStyle.Information, "Registro no encontrado")
      datagrid.ClearSelection()
    End Try
  End Sub
```

```
    Private Sub CbCodigoActualizar_SelectionChangeCommitted(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles CbCodigoActualizar.SelectionChangeCommitted 'Proceso al seleccionar un
código de registro en CbCodigoActualizar (sub-pestaña Actualizar Registro)
        If DataGridView1.Rows(CbCodigoActualizar.SelectedIndex).Cells(5).Value = "Activo" Then
'Condición que permite actualizar solo registros en estado "Activo"
            GBactualizar.Enabled = True
            TxtDesActualizar.Text =
Trim(DataGridView1.Rows(CbCodigoActualizar.SelectedIndex).Cells(2).Value)
            LbCantActualizar.Text =
Trim(DataGridView1.Rows(CbCodigoActualizar.SelectedIndex).Cells(1).Value)
            If Trim(DataGridView1.Rows(CbCodigoActualizar.SelectedIndex).Cells(3).Value) = "" Then
                TxtPreActualizar.Text = ""
            Else
                TxtPreActualizar.Text =
Trim(FormatCurrency(DataGridView1.Rows(CbCodigoActualizar.SelectedIndex).Cells(3).Value, 2))
            End If
        Else
            'Instrucciones en caso de que el código seleccionado sea un registro en estado "No activo"
            MsgBox("Registro no puede Actualizarse: Active primero", MsgBoxStyle.Exclamation, "Registro No
Activo")
            CbCodigoActualizar.SelectedIndex = -1
            GBactualizar.Enabled = False
        End If
    End Sub
    Private Sub btnActualizar_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnActualizar.Click 'Proceso de actualización de registro (click en botón Actualizar de sub-pestaña
Actualizar Registro)
        If Trim(TxtDesActualizar.Text) <> Nothing And Trim(TxtPreActualizar.Text) <> Nothing Then
            'Descarga de datos actualizados en datagrid General y datagrid Actualización
            DataGridView1.Rows(CbCodigoActualizar.SelectedIndex).Cells(2).Value =
Trim(TxtDesActualizar.Text)
            DataGridView1.Rows(CbCodigoActualizar.SelectedIndex).Cells(3).Value =
CDec(TxtPreActualizar.Text)
            DataGridView3.Rows(CbCodigoActualizar.SelectedIndex).Cells(1).Value =
Trim(TxtDesActualizar.Text)
            'Inicialización de controles sub-pestaña Actualizar Registro
            TxtDesActualizar.Clear()
            TxtPreActualizar.Clear()
            LbCantActualizar.ResetText()
            GBactualizar.Enabled = False
            CbCodigoActualizar.SelectedIndex = -1
            'Actualiza datagrid Disponibles
            Call cargardisponibles()
        Else
            MsgBox("Debe llenar los campos vacíos", MsgBoxStyle.Exclamation, "Operación No Válida")
        End If
    End Sub
    Private Sub TabPageActualizar_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TabPageActualizar.Leave
        'Inicializa controles internos de salir de sub-pestaña Actualizar Registro
        CbCodigoActualizar.SelectedIndex = -1
        TxtDesActualizar.Clear()
```

```
        TxtPreActualizar.Clear()
        LbCantActualizar.ResetText()
        GBactualizar.Enabled = False
    End Sub
    Private Sub CbCodigoEstado_SelectionChangeCommitted(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CbCodigoEstado.SelectionChangeCommitted 'Proceso al seleccionar un código
de registro en CbCodigoEstado (sub-pestaña Cambiar Estado)
        CbCodigoEstado.Enabled = False
        GBcambioestado.Enabled = True
        TxtDesEstado.Text = Trim(DataGridView1.Rows(CbCodigoEstado.SelectedIndex).Cells(2).Value)
        LbEstadoCambiar.Text = Trim(DataGridView1.Rows(CbCodigoEstado.SelectedIndex).Cells(5).Value)
        ' Selecciona y habilita los radiobotones de acuerdo al estado de registro seleccionado
        If Trim(DataGridView1.Rows(CbCodigoEstado.SelectedIndex).Cells(5).Value) = "Activo" Then
            RbNoactivo.Enabled = True
            RbActivo.Checked = True
        Else
            RbActivo.Enabled = True
            RbNoactivo.Checked = True
        End If
    End Sub
    Private Sub RbActivo_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles RbActivo.CheckedChanged
        'Habilitación de Botón Aplicar al seleccionar radiobotón "Activo"
        btnAplicar.Enabled = True
    End Sub
    Private Sub RbNoactivo_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RbNoactivo.CheckedChanged
        'Habilitación de botón Aplicar al seleccionar radiobotón "No activo"
        btnAplicar.Enabled = True
    End Sub
    Private Sub btnAplicar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnAplicar.Click 'Proceso de cambio de estado de registro (click en botón Aplicar- sub-pestaña Cambiar
Estado)
        If RbActivo.Checked = True Then ' Instrucciones si registro se Habilitará (estado "Activo)
            DataGridView1.Rows(CbCodigoEstado.SelectedIndex).Cells(5).Value = "Activo"
            DataGridView1.Rows(CbCodigoEstado.SelectedIndex).DefaultCellStyle.BackColor = Color.White
            DataGridView3.Rows(CbCodigoEstado.SelectedIndex).DefaultCellStyle.BackColor = Color.White
        Else 'Instrucciones si registro se inhabilitará (estado "No activo")
            DataGridView1.Rows(CbCodigoEstado.SelectedIndex).Cells(5).Value = "No Activo"
            DataGridView1.Rows(CbCodigoEstado.SelectedIndex).Cells(4).Value = Nothing
            DataGridView1.Rows(CbCodigoEstado.SelectedIndex).DefaultCellStyle.BackColor = Color.Yellow
            DataGridView3.Rows(CbCodigoEstado.SelectedIndex).DefaultCellStyle.BackColor = Color.Yellow
        End If
        'Inicialización de controles (sub-pestaña Cambiar Estado)
        TxtDesEstado.Clear()
        LbEstadoCambiar.ResetText()
        GBcambioestado.Enabled = False
        CbCodigoEstado.Enabled = True
        CbCodigoEstado.SelectedIndex = -1
        RbActivo.Checked = False
        RbNoactivo.Checked = False
        RbNoactivo.Enabled = False
```

```
        RbActivo.Enabled = False
        btnAplicar.Enabled = False
        'Actualiza datagrid Disponibles
        Call cargardisponibles()
    End Sub
    Private Sub btnCancelar_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnCancelar.Click
        'Inicializa controles internos de sub-pestaña Cambiar Estado al cancelar operación de cambio de estado
(click btnCancelar)
        TxtDesEstado.Clear()
        LbEstadoCambiar.ResetText()
        GBcambioestado.Enabled = False
        CbCodigoEstado.Enabled = True
        CbCodigoEstado.SelectedIndex = -1
        RbActivo.Checked = False
        RbNoactivo.Checked = False
        RbNoactivo.Enabled = False
        RbActivo.Enabled = False
        btnAplicar.Enabled = False
    End Sub
    Private Sub TabPageEstado_Leave(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TabPageEstado.Leave
        'Inicializa controles internos de sub-pestaña Cambiar Estado al salir de la misma
        btnCancelar.PerformClick() 'Evento click botón Cancelar
    End Sub
    Private Sub btnGuardarGeneral_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnGuardarGeneral.Click
        'Ejecuta proceso GuardarReporte para inventario General
        datagrid = DataGridView1
        Call GuardarReporte()
    End Sub
    Private Sub BtnguardarActualizacion_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnguardarActualizacion.Click
        'Ejecuta proceso GuardarReporte para invenatrio de Último Movimiento
        datagrid = DataGridView3
        Call GuardarReporte()
    End Sub
    Sub GuardarReporte() 'Proceso para guardar reporte del inventario
        'Características de la ventana de diálogo "Guardar Como"
        Dim saveFileDialog1 As New SaveFileDialog()
        saveFileDialog1.Title = "Guardar Como"
        saveFileDialog1.InitialDirectory = "C:\Usuarios\pc\Documentos"
        saveFileDialog1.Filter = "Archivos Excel 2007 (*.xlsx)|*.xlsx"
        saveFileDialog1.FilterIndex = 1
        saveFileDialog1.OverwritePrompt = True
        saveFileDialog1.DefaultExt = ".xlsx"
        saveFileDialog1.AddExtension = True
        saveFileDialog1.RestoreDirectory = True
        Try
            If saveFileDialog1.ShowDialog() = DialogResult.OK Then 'Si se presionó tecla Guardar en ventana de
diálogo
```

```
            Dim ruta As String = saveFileDialog1.FileName 'Asigna a variable "ruta" el nombre de archivo
escrito en la ventana de diálogo
            Dim oExcel As Object 'Objeto para aplicación excel
            Dim oBook As Object 'Objeto para libro excel
            Dim oSheet As Object 'Objeto para hoja excel
            'Inicia un nuevo libro en Excel
            oExcel = CreateObject("Excel.Application")
            oBook = oExcel.Workbooks.Add
            'Número de columnas y filas del datagrid
            Dim NCol As Integer = datagrid.ColumnCount
            Dim NRow As Integer = datagrid.RowCount
            'Agrega encabezados de datagrid a la hoja de cálculo en la fila 1
            oSheet = oBook.Worksheets(1)
            For i As Integer = 1 To NCol
               oSheet.Cells.Item(1, i) = datagrid.Columns(i - 1).HeaderText
               oSheet.Cells.Item(1, i).HorizontalAlignment = 3
            Next
            'Se recorren todas las filas, y por cada fila todas las columnas y se va escribiendo en excel
            For Fila As Integer = 0 To NRow - 1
               For Col As Integer = 0 To NCol - 1
                  oSheet.Cells.Item(Fila + 2, Col + 1) = datagrid.Rows(Fila).Cells(Col).Value
               Next
            Next
            'Se da formato a excel
            oSheet.Columns.HorizontalAlignment = 2
            'Titulo en negrita, Alineado al centro y que el tamaño de la columna se ajuste al texto
            oSheet.Rows.Item(1).Font.Bold = 1
            oSheet.Rows.Item(1).HorizontalAlignment = 3
            oSheet.Columns.AutoFit()
            'Se guarda el libro en la ruta especificada y se finaliza proceso Excel
            oBook.SaveAs(ruta)
            System.Runtime.InteropServices.Marshal.ReleaseComObject(oExcel)
            oSheet = Nothing
            oBook = Nothing
            oExcel = Nothing
            Dim proc As System.Diagnostics.Process
            For Each proc In System.Diagnostics.Process.GetProcessesByName("EXCEL")
               proc.Kill()
            Next
            MessageBox.Show("se guardo como" & ruta)
         Else
            MsgBox("No se guardó el Reporte", MsgBoxStyle.Information, "Aviso")
         End If
         'Error
      Catch ex As Exception
         MsgBox(ex.Message, MsgBoxStyle.Critical, "Error al exportar a Excel")
      End Try
   End Sub
   Private Sub btnInventarios_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnInventarios.Click
      'Muestra página de inventarios: Pestaña Inventarios
      TabControlprincipal.SelectedTab = TabPageInventarios
```

```
    End Sub
    Private Sub btnEditar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnEditar.Click
        'Muestra página de edición: Pestaña Editar Registros
        TabControlprincipal.SelectedTab = TabPageEdicion
    End Sub
    Private Sub TabPageInventarios_Layout(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LayoutEventArgs) Handles TabPageInventarios.Layout
        'Muestra sub-pestaña Consulta General al mostrarse pestaña Inventarios
        TabControlInventarios.SelectedTab = TabPageGeneral
    End Sub
    Private Sub TabPageEdicion_Layout(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LayoutEventArgs) Handles TabPageEdicion.Layout
        'Muestra sub-pestaña Actualizar Registro al mostrase pestaña Editar Registros
        TabControlEdicion.SelectedTab = TabPageActualizar
    End Sub
    Private Sub Form1_FormClosing(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles MyBase.FormClosing 'Proceso para guardar en un
archivo datos de datagrid General  y datagrid Actualización al cerrar Aplicación, para poder cargar mismos
datos en los datagrid al momento de abrir de nuevo la Aplicación
        If btnIniciar.Text = "Iniciar" Then
            Dim resultado As MsgBoxResult = MsgBox("¿Está seguro que desea abandonar la Aplicación?",
MsgBoxStyle.YesNo + MsgBoxStyle.Question, "Confirmación") 'Mensaje de confirmación de cierre de
Aplicación
            If resultado <> MsgBoxResult.Yes Then
                e.Cancel = True
            Else
                e.Cancel = False
                Dim numCols As Integer = DataGridView1.ColumnCount 'número columnas datagrid General
                Dim numRows As Integer = DataGridView1.RowCount 'número filas datagrid Genaral
                Dim numCols1 As Integer = DataGridView3.ColumnCount 'número columnas datagrid
Actualización
                Dim ruta As String = "C:\BaseInventario\BaseDatos.txt" 'Ruta de archivo para escritura
                Dim objWriter As New StreamWriter(ruta) 'Crea objeto para escritura de archivo
                'Ciclo que recorre datagrid General y escribe en el archivo el dato de cada celda
                For count As Integer = 0 To numRows - 1
                    For count2 As Integer = 0 To numCols - 1
                        objWriter.Write(DataGridView1.Rows(count).Cells(count2).Value)
                        If (count2 <> numCols - 1) Then
                            objWriter.Write(";") 'Separa dato de cada celda
                        End If
                    Next
                    objWriter.WriteLine() 'Termina escritura de línea en archivo
                Next
                'Ciclo que recorre datagrid Actualización y escribe en el archivo el dato de cada celda
                For count As Integer = 0 To numRows - 1
                    For count2 As Integer = 0 To numCols1 - 1
                        objWriter.Write(DataGridView3.Rows(count).Cells(count2).Value)
                        If (count2 <> numCols1 - 1) Then
                            objWriter.Write(";") 'Separa dato de cada celda
                        End If
                    Next
```

```
            objWriter.WriteLine() 'Termina escritura de línea en archivo
          Next
          objWriter.Close() 'Cierra el archivo
        End If
        NotifyIcon1.Visible = False 'Elimina el icono de la barra de tareas del ordenador
      Else
        MsgBox("Debe finalizar la conexión antes de salir", MsgBoxStyle.Critical, "Operación no válida")
        e.Cancel = True
      End If
    End Sub
    Private Sub btnSalir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnSalir.Click'Proceso para evento Click de botón Salir (btnsalir)
      Me.Close() 'Cierra aplicación
    End Sub
    Private Sub DataGridView1_CellDoubleClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles DataGridView1.CellDoubleClick ' Proceso
para evento doble click en celda de datagrid General
      FormDetalleRegistro.Show() 'Muestra "Ventana Información de Producto" del registro seleccionado en
datagrid General
    End Sub
    Private Sub TxtPreActualizar_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TxtPreActualizar.KeyPress 'Proceso cuando cuando
usuario presiona y suleta una tecla en txtPreActualizar(sub-pestaña Actualizar Registro)
      'Permite escribir en Textbox de Precio (sub-pestaña Actualizar Registro) solo números y "," para
decimales
      Dim KeyAscii As Short = CShort(Asc(e.KeyChar)) 'Habilita el parámetro «KeyAccii» del antiguo VB
      Select Case KeyAscii
        Case 8 'Tecla de retroceso
        Case 44 'Coma decimal
        Case Else
          If Not Char.IsDigit(e.KeyChar) Then
            'Se ha especificado una letra
            e.Handled = True 'Indica que esa pulsación (letra) ha sido "manejada" por el evento (keyPress) y
se ignorará
          End If
      End Select
    End Sub
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick
      LbHora.Text = "Hora: " & TimeOfDay 'Reloj en Pantalla Principal
      LbFecha.Text = "Fecha: " & Today   'Fecha en Pantalla Principal
    End Sub
End Class
-------------------------------------------------------------------------------------------------------------------
Public Class FormDetalleRegistro 'Ventana Información de Producto
    Private Sub FormDetalleRegistro_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
      'Carga en las etiquetas del formulario los detalles del producto seleccionado (fila seleccionada en
datagrid General)
      LbCodigo2.Text = Trim(Form1.DataGridView1.CurrentRow.Cells(0).Value)
      TxtDescripcion2.Text = Trim(Form1.DataGridView1.CurrentRow.Cells(2).Value)
      LbCantidad2.Text = Trim(Form1.DataGridView1.CurrentRow.Cells(1).Value)
```

```
        LbHora2.Text =
Trim(Form1.DataGridView3.Rows(Form1.DataGridView1.CurrentRow.Index).Cells(4).Value)
        If Trim(Form1.DataGridView1.CurrentRow.Cells(3).Value) = "" Then
            LbPrecio2.Text = ""
        Else
            LbPrecio2.Text = FormatCurrency(Form1.DataGridView1.CurrentRow.Cells(3).Value, 2) 'Coloca
dato de Precio en formato moneda
        End If
        LbEstado2.Text = Trim(Form1.DataGridView1.CurrentRow.Cells(5).Value)
    End Sub
End Class
-------------------------------------------------------------------------------------------------------------------------------
Public Class FormPresentación 'Pantalla de Presentación
    Delegate Sub delegado()
    Private Sub FormPresentación_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load 'Proceso para cargar barra de progreso de Pantalla de Presentación
        ProgressBar1.Value = 0 'Inicia la posición de la barra de progreso
        Timer1.Start() 'Inicia el temporizador
    End Sub
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick
        'Proceso que se ejecuta transcurrido cada intervalo del temporizador
        Dim metodo As New delegado(AddressOf Me.incremento) 'Asocia el método "incremento" con el
delegado
        Me.Invoke(metodo) 'Invoca al método a través del delegado
    End Sub
    Sub incremento() 'Proceso para incrementar progreso de barra
        ProgressBar1.Increment(10) 'Incremento de la barra de progreso
        Lbprogreso.Text = ProgressBar1.Value & "%" 'Coloca en una etiqueta el porcentaje de progreso de la
barra
    End Sub
End Class
```