UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD TÉCNICA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

NIVEL LICENCIATURA

PROYECTO DE GRADO

"CONTROL EN EL PROCESO DE PASTEURIZACIÓN DE LA LECHE"

POSTULANTE: ERWIN NELSON ZABALETA AGUILAR

TUTOR: ING. LUIS RAMIRO VELARDE CHÁVEZ

LA PAZ – BOLIVIA

SEPTIEMBRE, 2012

UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD TÉCNICA

CARRERA ELECTRÓNICA Y TELECOMUNICACIONES

Proyecto de grado:

SISTEMA DE CONTROL EN EL PROCESO DE PASTEURIZACIÓN DE LA LECHE

Presentado por: Univ. Erwin Nelson Zabaleta Aguilar

Para optar el grado académico de *Licenciado en Electrónica y Telecomunicaciones*

Nota numeral: .....

Nota literal: .....

Ha sido: .....

Director de la carrera de Electrónica y Telecomunicaciones: Lic. Nixon Vargas Mamani

Tutor: Ing. Luis Ramiro Velarde Chávez

Tribunal: Ing. Pedro Franklin Rada Telleria

Tribunal: Lic. Edwin Jesús Alave Alavi

Tribunal: Ing Roger Reynaldo Guachalla Narváez

A Dios y a mis queridos padres Carmen y Víctor, que me dieron una nueva oportunidad.

AGRADECIMIENTO

Primeramente agradezco a Dios, por la vida, bendiciones y oportunidades que me da.

Agradezco y valoro infinitamente a nuestra Prestigiosa Casa de Estudios Superiores, Universidad Mayor de San Andrés, por acogerme durante este tiempo de formación, proporcionándome una calidad de enseñanza académica base fundamental para mi vida profesional, y a las oportunidades que me dieron durante el tiempo en que fui auxiliar de docencia, demostrando de esta manera el infinito cariño que tengo a mi Universidad.

Agradezco de manera sincera y especial al Ing. Ramiro Velarde, docente de la UMSA, por sus consejos, orientaciones y sugerencias, que tan acertadamente me brindó y cuyo manuscrito final fue posible gracias a su colaboración.

Por último, agradezco a mis amigos de la Facultad que durante todo este tiempo me colaboraron brindándome todo el apoyo y comprensión que incondicionalmente me brindaron y me brindan.

TABLA DE CONTENIDOS **Página**

ANEXO A Especificaciones PIC18F4550 (MICROCHIP)

ANEXO B Compilador C para Microchip PICmicro (Cánovas López Andrés)

FIGURAS **Página**

CUADROS **Página**

RESUMEN

Con el avance de las tecnologías en complemento a procesos industriales, hace posible que se busquen alternativas en especifico al Proceso de Pasteurización de la Leche en cuestión a las formas que posibiliten dicho proceso siempre desde un enfoque automático.

De esta forma se ve necesario crear Sistemas de Control que posibiliten en los procesos específicos que conlleva la Pasteurización de la Leche, tomando en cuenta que éste conlleva el uso de parámetros, métodos, técnicas, entre otros, el cual hace posible el propósito de la Pasteurización. Motivo por el cual se decide realizar un estudio y análisis de los problemas, los recursos y medios que a menudo se presenta en dicho proceso, para lo cual es necesario hacer uso de Metodologías enfocadas a la creación de Prototipos, que busquen las soluciones y satisfagan las necesidades siempre tomando en cuenta las características y/o restricciones del problema que posibiliten de esta forma, obtener resultados óptimos y satisfactorios.

Desarrollar Prototipos, significa hacer un desarrollo evolutivo del Sistema, observando detalladamente si los objetivos están siendo alcanzados a través de resultados factibles que de alguna u otra forma definirán el éxito o fracaso del Proyecto.

En el Primer Capitulo del Proyecto, se plantean los problemas, los objetivos y el Planteamiento de la Solución, que sin lugar a dudas, son la base fundamental para la creación del Sistema; El Segundo Capítulo contiene y engloba todos los conceptos y definiciones necesarios de los medios que posibilitarán llegar al objetivo principal de éste Proyecto; El Tercer Capítulo muestra la Construcción del Prototipo enfocado a resolver los problemas de una forma tangible, sujeto a pruebas y medidas que definan su calidad; finalmente el Cuarto Capítulo contiene las conclusiones y recomendaciones concorde a todo el trabajo que se desarrollo.

SUMMARY

With the advancement of technologies in addition to industrial processes, makes it possible to specifically look for alternatives to Process Pasteurization of milk in question forms that facilitate the process always from an auto focus.

This is necessary to create control systems that enable specific processes involved in Pasteurized Milk, considering that it involves the use of parameters, methods, techniques, among others, which makes possible the purpose of pasteurization. Why it was decided to conduct a study and analysis of problems, resources and means that often occurs in this process, which is necessary to make use of methodologies aimed at the creation of prototypes, to seek solutions and meet needs while taking into account the characteristics and / or restrictions that allow the problem this way, optimal and satisfactory results.

Develop Prototypes, signifies an evolutionary development of the observing system in detail whether the objectives are being achieved through feasible results in some form or another will define success or failure of the Project.

The First Chapter of the project, discuss the challenges, objectives and approach of the solution, which undoubtedly are the foundation for the creation of the system; The second chapter contains and encompasses all the necessary concepts and definitions the means that will enable to reach the main objective of this project; the third chapter shows the construction of the prototype focused on solving problems in a tangible way, subject to tests and measures to define quality, and finally the fourth chapter contains the conclusions and recommendations concorde to all work developed.

PALABRAS CLAVES

Planta: Conjunto de componentes y piezas que van a tener un determinado objetivo.

Proceso: Conjunto de operaciones que se van a suceder y que van a tener un fin determinado.

Sistema: Combinación de componentes que actúan juntos para realizar el control.

Perturbaciones: Todas las señales indeseadas que intervienen de forma ad-versa en el funcionamiento de un sistema. Pueden ser *internas* si se generan dentro del sistema, o *externas* si se generan fuera del sistema y constituyen una entrada.

Entrada de mando: Señal excitadora del sistema que es independiente de la salida del mismo.

Selector de referencia: Elemento que se coloca para tener una referencia. Unidad que establece el valor de la entrada de referencia. Se calibra en función del valor deseado en la salida del sistema.

Entrada de referencia: Señal producida por el selector de referencia.

Unidad de control: Unidad que reacciona con una señal activa para producir la salida deseada. Realiza el trabajo de gobernar la salida.

Salida: Cantidad que debe mantenerse en un valor fijado de antemano. Se considera la variable gobernada.

Sistema de control en bucle abierto: Sistema en el que la salida no tiene influencia sobre la entrada.

Elemento de realimentación: Unidad que facilita medios para aumentar o disminuir la señal de salida.

Señal activa: Señal que es la diferencia entre la señal de entrada y la salida realimentada.

Sistema de control de bucle cerrado: Sistema en el que la salida afecta a la entrada, de tal manera que mantenga el valor de salida deseado.

CAPÍTULO I

INTRODUCCIÓN

CAPÍTULO I
INTRODUCCIÓN

1.1. INTRODUCCIÓN.

Para destruir los microorganismos de la leche es necesario someterlos a tratamientos térmicos, los cuales también pueden ocasionar transformaciones no deseables en la leche, que provocan alteraciones de sabor, rendimiento, y calidad principalmente. El proceso de pasteurización disminuye toda la flora de microorganismos como parásitos no patógenos y la totalidad de los agentes microbianos patógenos, pero alterando en lo mínimo posible la estructura física y química de la leche y las sustancias con actividad biológica tales como enzimas y vitaminas.

La temperatura y tiempo aplicados en la pasteurización aseguran la destrucción de los agentes patógenos, pero no destruye algunos microorganismos como por ejemplo los responsables de la acidez. Es por tal razón que el presente proyecto está enfocado a coadyuvar al proceso de pasteurización de la leche, utilizando medios tecnológicos que posibiliten el alcance del objetivo de la pasteurización.

1.2. ANTECEDENTES.

Se han estudiado distintas combinaciones de temperatura y tiempo para pasteurizar pero fundamentalmente se han reducido a dos:

- Pasteurización lenta o discontinua.
- Pasteurización rápida o continua.

1.2.1. PASTEURIZACIÓN LENTA O DISCONTINUA.

Este método consiste en calentar la leche a temperaturas entre 62 y 64ºC y mantenerla a esta temperatura durante 30 minutos. La leche es calentada en recipientes o tanques de capacidad variable (generalmente de 200 a 1500 litros); esos tanques son de acero inoxidable preferentemente y están encamisados (doble pared); la leche se calienta por medio de vapor o agua caliente que vincula entre las paredes del tanque, provisto este de un agitador para hacer más homogéneo el tratamiento.

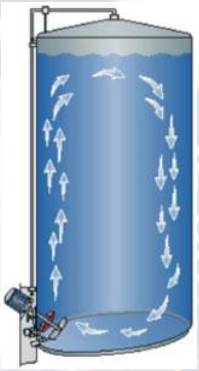


Figura 1.1.: Recipiente o tanque
Fuente: Nasanovsky, Garijo, Kimmich.

Luego de los 30 minutos, la leche es enfriada a temperaturas entre 4 y 10ºC según la conveniencia. Para efectuar este enfriamiento se puede usar el mismo recipiente haciendo circular por la camisa de doble fondo agua helada hasta que la leche tenga la temperatura deseada. Otra manera, es enfriar utilizando el enfriador de superficie (o cortina de enfriamiento).

Ambos métodos de enfriamiento tienen sus inconvenientes: en el primer caso (utilizando el mismo tanque), la temperatura desciende cada vez más lentamente a medida que se acerca a la temperatura del agua helada, lo cual hace que la leche, durante un cierto tiempo, este a las temperaturas en que crecen los microorganismos que quedarán luego del tratamiento térmico, lo cual hace que aumente la cuenta de agentes microbianos. Por otra parte, usando la cortina de enfriamiento la leche forma una película sobre la superficie de la cortina y el enfriamiento es más rápido, pero, por quedar la leche en contacto con el ambiente, es presa de la contaminación.

El uso de la pasteurización lenta es adecuada para procesar pequeñas cantidades de leche hasta aproximadamente 2000 litros diarios, de lo contrario no es aconsejable.

1.2.2. PASTEURIZACIÓN RÁPIDA O CONTINUA.

Este tratamiento consiste en aplicar a la leche una temperatura de 72 a 73ºC en un tiempo de 15 a 20 segundos.

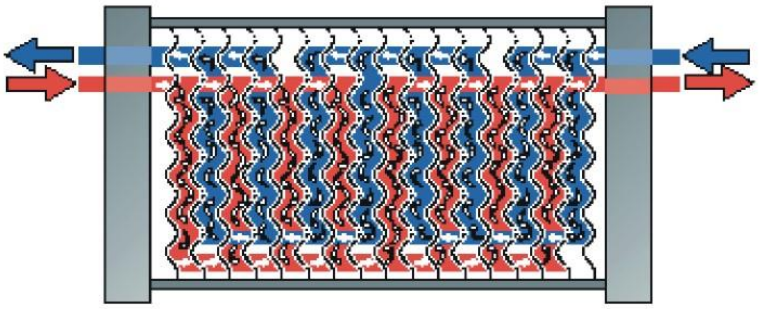


Figura 1.2.: Intercambiadores de placas
Fuente: Nasanovsky, Garijo, Kimmich.

Esta pasteurización se realiza en intercambiadores de calor de placas, y el recorrido que hace la leche en el mismo es el siguiente:

La leche llega al equipo intercambiador a 4ºC aproximadamente, proveniente de un tanque regulador; en el primer tramo se calienta por regeneración. En esta sección de regeneración o precalentamiento, la leche cruda se calienta a 58ºC aproximadamente por medio de la leche ya pasteurizada cuya temperatura se aprovecha en esta zona de regeneración. Al salir de la sección de regeneración, la leche pasa a través de un filtro que elimina impurezas que pueda contener, luego la leche pasa a los cambiadores de calor de la zona o área de calentamiento donde se la calienta hasta la temperatura de pasteurización, esta es 72 a 73ºC por medio de agua caliente.

Alcanzada esta temperatura la leche pasa a la sección de retención de temperatura; esta sección puede estar constituida por un tubo externo o bien un retardador incluido en el propio intercambiador; el más común es el tubo de retención, en donde el tiempo que la leche es retenida es de 15 a 20 segundos. A la salida de esta zona de retención, la leche pasa por una válvula de desviación; en esta válvula, si la leche no alcanza la temperatura de 72 a 73ºC, automáticamente la hace regresar al tanque regulador o de alimentación para ser luego reprocesada; pero si la leche alcanza la temperatura de 72 a 73ºC, pasa entonces a la zona de regeneración o precalentamiento, donde es enfriada por la leche cruda hasta los 18ºC. De aquí la leche pasa a la sección de enfriamiento en donde se distinguen dos zonas: una por donde se hace circular agua fría y la otra en donde circula agua helada, para terminar de esta manera el recorrido de la leche, saliendo del intercambiador a la temperatura de 4ºC generalmente. En el esquema siguiente se muestra el recorrido de la leche por el intercambiador:

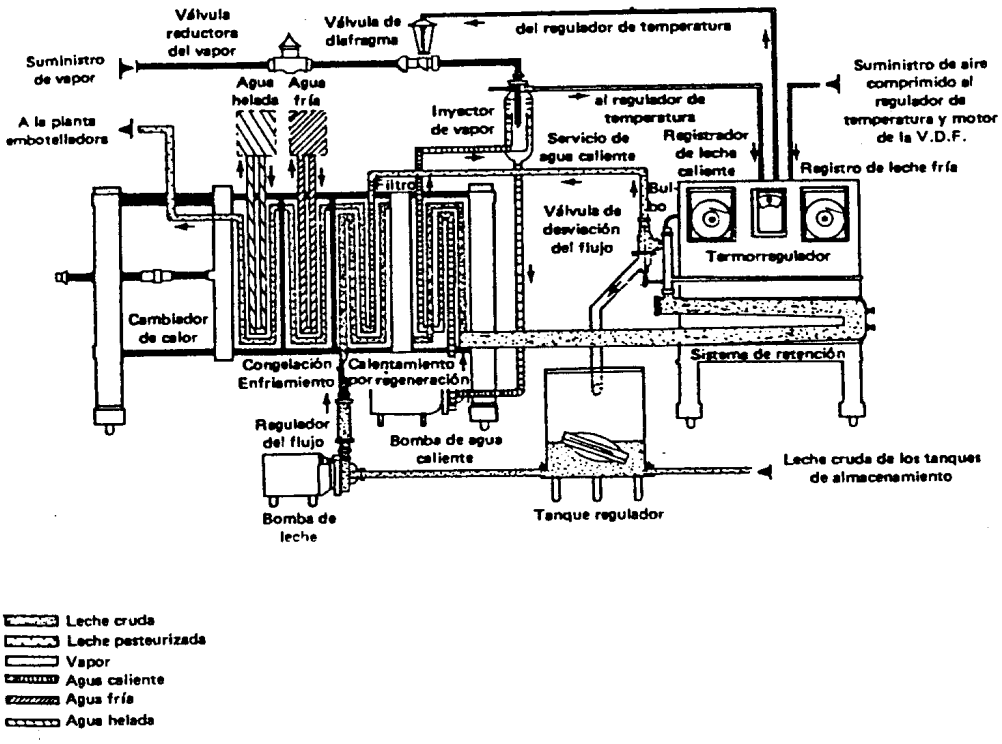


Figura 1.3.: Esquema del intercambiador
Fuente: Nasanovsky, Garijo, Kimmich.

El intercambiador de calor, es utilizado por su alta velocidad de transferencia y su facilidad de limpieza. Son construidos en acero inoxidable; las placas tienen generalmente un espesor aproximado de 0.05 a 0.125 pulgadas; están aisladas mediante juntas de goma que forman una camisa de entre 0.05 y 0.3 pulgadas entre cada par de placas; estas últimas se ordenan en secciones: precalentamiento, calentamiento y enfriamiento.

Cada sección aislada se ordena de tal forma que los líquidos fluyen por una o más placas en paralelo. En la figura siguiente se muestra la disposición de las placas y circulación de los fluidos. Las placas tienen nervaduras o estrías que provocan turbulencia y aumentan la superficie de intercambio.

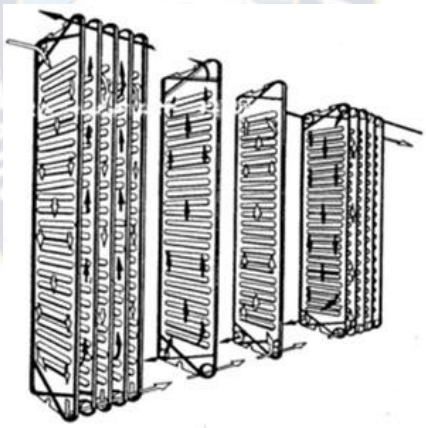


Figura 1.4.: Placas de intercambio
Fuente: Nasanovsky, Garijo, Kimmich.

1.2.3. LAS VENTAJAS DE LA PASTEURIZACIÓN RAPIDA RESPECTO A LA LENTA.

a) Pueden procesarse en forma continua grandes volúmenes de leche.
b) Es de fácil limpieza y requiere poco espacio.
c) Por ser de sistema cerrado se evitan contaminaciones.
d) Rapidez del proceso.

En cuanto a las desventajas se pueden nombrar:

a) No puede adaptarse al procesamiento de pequeñas cantidades de leche.

b) Las gomas que acoplan las placas son demasiado frágiles.

c) Es difícil un drenaje o desagote completo.

Muchas plantas industriales hacen una clasificación de la leche previa y posterior a la pasteurización. Es así que se pueden tener leches que antes del tratamiento no contengan más de 50000 microorganismos por mililitro y luego de la pasteurización no contienen más de 15000 microorganismos por milímetro. Otra clasificación es de aquellos que tienen no más de 300000 microorganismos/ml antes y no más de 30000 ml luego de la pasteurización y finalmente lo que antes del tratamiento térmico no tengan más de 2000000 de microorganismos/ml y que luego del mismo no contengan más de 30000 ml.

1.2.4. ESTADO DE LA LECHE LUEGO DE SER PASTEURIZADA.

Respecto a los componentes de la leche, luego de la pasteurización, no está afectada la línea de crema, la lactosa prácticamente no sufre ningún cambio, tampoco sufren cambios las proteínas del suero de la leche, por lo cual no se forman olor y sabor a cocido. Por último, las pasteurizaciones no afectan o afectan poco a las vitaminas.

1.3. PLANTEAMIENTO DE PROBLEMAS.

1.3.1. PROBLEMA PRINCIPAL.

El proceso de pasteurización, determina el establecimiento de procesos tanto de calentamiento como de enfriamiento por el cual se necesita un control riguroso para lograr alcanzar los niveles de temperatura según el tipo de pasteurización en determinados tiempos.

1.3.2. PROBLEMAS SECUNDARIOS.

- Programar el punto de ajuste de temperatura - tiempo.
- Adaptar la potencia en los procesos de calentamiento y enfriamiento según las necesidades reales.

- Elevar la temperatura al nivel deseado.
- Disminuir la temperatura al nivel deseado.
- Establecer estadísticas del proceso de pasteurización, para cualquier contingencia que pueda ocurrir.

1.4. PLANTEAMIENTO DE OBJETIVOS.

1.4.1. OBJETIVO GENERAL.

Desarrollar un sistema de control de procesos el cual verifique y controle el calentamiento de la leche durante un tiempo, determinado por el tipo de pasteurización, y luego ingrese al proceso de enfriamiento de una forma automática previa programación de los parámetros requeridos.

1.4.2. OBJETIVOS ESPECÍFICOS.

- Automatizar los procesos de pasteurización de la leche pudiendo aplicarse o implementarse en empresas cuyo proceso es de forma rudimentaria.
- Mostrar los niveles de energía, a través de cuadros estadísticos, los cuales mostrarán los consumos reales para realizar correcciones de errores en los diferentes procesos.
- Confirmar que las decisiones para controlar el consumo de energía dan los resultados adecuados.
- Asegurar que los procedimientos de operación sean los adecuados.
- Generar los reportes de uso y costo de energía adecuados.
- Establecer interfaces de comunicación entre el sistema y el operador, utilizando medios tecnológicos.

1.5. JUSTIFICACIONES.

1.5.1. SOCIAL.

Considerando que éste es un sistema que se orienta a empresas cuyo trabajo de pasteurización es rudimentario, se justifica socialmente ya que coadyuva al trabajo de la Empresa, para obtener los resultados requeridos, cumpliendo con estándares, nacionales e internacionales.

1.5.2. TÉCNICA.

La conjunción de tecnologías electrónicas con mecanismos y/o procesos industriales de pasteurización, hace que sea una alternativa para mejorar el proceso de pasteurización, a través de un control automático.

1.5.3. OPERATIVA.

La incorporación de ésta tecnología a mecanismos o procesos de pasteurización, hace aplicable la utilización de éste Sistema de Control.

1.5.4. CIENTÍFICA.

La conjunción de la Electrónica y el uso de Metodologías que se utilizarán, para la construcción del sistema, hace preponderante el trabajo que se realice, ya que en la actualidad son pocos los que recurren a controles automáticos de procesos industriales.

1.6. DELIMITACIONES.

1.6.1. TEMÁTICOS.

Se toma en cuenta el uso de circuitos y/o sistemas de control, delimitados a un área de trabajo, que en este caso son las Empresas Lecheras.

1.6.2. TEMPORALES.

Se establecen tiempos específicos tanto para el Análisis, Diseño, Implementación y Evaluación del Sistema.

1.7. VIABILIDAD.

1.7.1. RECURSOS FINANCIEROS.

El proyecto está sustentado por financiamiento propio.

1.7.2. RECURSOS MATERIALES.

En el periodo de prueba, se requerirá de un prototipo el cual emule los procesos de pasteurización para evaluar la efectividad del Sistema, considerando recursos tanto Operativos y de Análisis para la culminación del Proyecto.

CAPÍTULO II

MARCO TEÓRICO

CAPÍTULO II

MARCO TEÓRICO

2.1. PROCESOS DE CONTROL.

2.1.1. SISTEMAS DE CONTROL EN LAZO ABIERTO.

Los sistemas de control en lazo abierto son aquellos en la cual salida no tiene efecto sobre la acción de control, o dicho de otra forma, son aquellos en los que la señal de salida no tiene influencia sobre la señal de entrada.

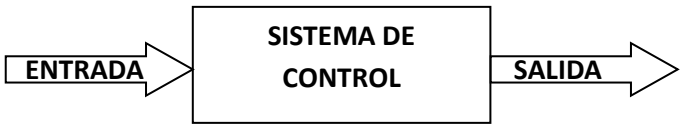


Figura 2.1.: Sistema de Control en Lazo Abierto
Fuente: Porras,Montanero.

La variable que se desea controlar puede divergir considerablemente del valor deseado debido a las perturbaciones externas, en tal sentido, en estos sistemas se debe calibrar los componentes que forman las diversas etapas, así como la no existencia de dichas perturbaciones. Por ejemplo, Un sistema en la que de desea mantener constante la temperatura de una habitación. En los procesos en lazo abierto, el tiempo es una variable de mucha importancia, ya que determinará el alcance de los objetivos mientras no cambien las condiciones, los cuales produzcan perturbaciones en el sistema.

Como otros ejemplos de sistemas de control en lazo abierto, se puede citar desde un simple tostador de pan, pasando por una máquina de lavar, hasta incluso el control de la velocidad de un motor.

2.1.2. SISTEMAS DE CONTROL EN LAZO CERRADO.

Si en un proceso se presentan alteraciones exteriores o perturbaciones, no se puede utilizar sistemas de control en lazo abierto. Motivo por el cual, resulta más conveniente cuantificar la señal o variable controladora e intervenir en la cadena de mando para que la variable controlada se parezca lo más posible a la señal de referencia dada por la señal de mando. Por ello, es necesario realizar una realimentación de la variable de salida a la entrada. Este proceso se denomina, Sistema de Control en Lazo Cerrado.

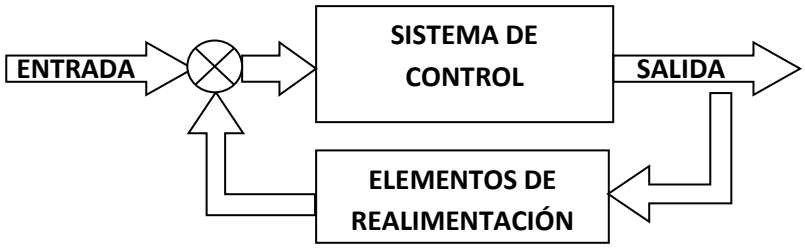


Figura 2.2.: Sistema de Control en Lazo Cerrado
Fuente: Porras,Montanero.

Dentro los Sistemas de Control en Lazo Cerrado, se puede mencionar a: los mecanismos de llenado de una cisterna de agua, la acción de un ser humano al desplazarse, el sistema de evaluación de un alumno en el colegio, relación salarios-precio-inflación, sistemas de control de nivel de líquidos y sólidos, entre otros.

2.1.3. COMPONENTES DE UN SISTEMA DE CONTROL.

2.1.3.1. TRANSDUCTORES Y CAPTADORES.

Los transductores son los elementos que transmiten información de su entrada a su salida; adaptan un tipo de energía a otro tipo de energía más conveniente para ser usada por el controlador. El transductor transforma la señal de mando en otra señal, denominada señal de referencia, que pueda adaptarse para ser comparada en el detector de error con la señal de realimentación.

Esta señal procede de una adaptación de la señal de salida mediante un elemento que se llama captador.

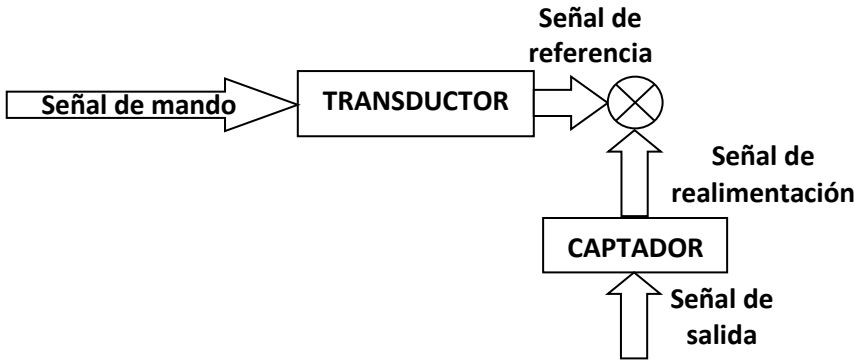


Figura 2.3.: Transductor y Captador
Fuente: Porras,Montanero.

El captador es un transductor colocado en un lugar distinto del sistema, lo que indica que la función dada a cada uno de estos dos componentes puede ser desempeñada por los mismos componentes físicos.

Se llama captador porque la misión que tiene este transductor es la de captar una determinada información, en lugar de reaccionar a una determinada señal de mando.

Ejemplo:

Un interruptor normal puede ser un transductor de entrada a un sistema, pero no podrá ser un captador, dado que está diseñado para funcionar manualmente. Un interruptor final de carrera es un captador, pero no es un transductor; ya que informa de que una pieza ha llegado a una determinada posición. Sin embargo, tanto el interruptor normal como el final de carrera son interruptores (transductores y captadores).

Éstos se clasifican en función de la naturaleza de la señal de mando, las cuales pueden ser de posición, de velocidad, de presión, de temperatura, entre otros.

2.1.3.2. DETECTORES DE ERROR O COMPARADORES.

Son los elementos encargados de proporcionar una señal en función de la diferencia existente entre la señal de entrada y la señal realimentada.

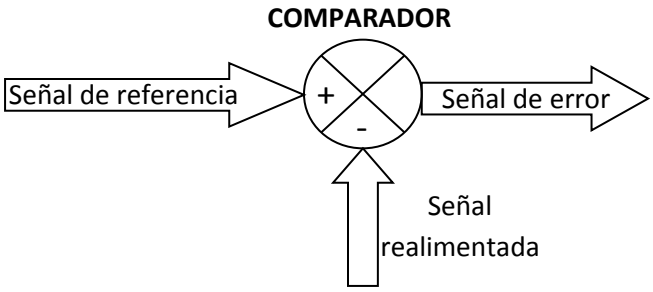


Figura 2.4.: Comparador
Fuente: Porras,Montanero.

2.1.3.3. ACTUADORES.

Son los elementos que proporcionan la variable de salida, generalmente es un dispositivo inherentemente mecánico cuya función es proporcionar fuerza para mover o "actuar", por lo cual existen actuadores lineales los cuales generan una fuerza en línea recta, tal como un pistón, y los actuadores rotatorios, los cuales generan una fuerza rotatoria, como lo haría un motor eléctrico.

2.2. METODOLOGÍA PARA LA CONSTRUCCIÓN DEL PROTOTIPO.

2.2.1. PROTOTIPOS.

La construcción del prototipo es muy eficaz bajo condiciones correctas, este método si no se usa de manera apropiada, sería poco útil en el desarrollo del sistema. El prototipo es un método que funciona no solamente en papel, un prototipo se desarrolla para probar las ideas y suposiciones relacionadas con el nuevo diseño en la cual se va mejorando en cada fase.

Un usuario a menudo define un conjunto de objetivos generales para el desarrollo en un sistema pero no identifica los requisitos detallados de entrada, procesamiento o

salida en otros casos. La construcción de prototipos comienza con la recolección de requisitos; para encontrar y definir los objetivos globales del sistema, identificando los requisitos conocidos, las áreas del esquema en donde es obligatorio aclarar. A partir de esto se realiza un diseño rápido que centra una representación de los aspectos considerados. El diseño rápido lleva a la construcción de un prototipo el cual se evalúa y lo utiliza para refinar los requisitos. La interacción ocurre cuando el prototipo satisface las necesidades requeridas, a la vez permite al desarrollador comprender mejor sobre los objetivos del sistema.

El prototipo puede servir como primer sistema para enfocar todos los objetivos del proyecto ya que el sistema se construirá a medida y con todos los requerimientos, para de esta manera cumplir y satisfacer todas las necesidades. Sin embargo la construcción de prototipo puede ser un problema en las siguientes condiciones: El desarrollador a menudo hace compromisos de implementación para que el prototipo funcione rápidamente. Se puede utilizar un circuito en especial o un dispositivo ya establecido inadecuado, simplemente porque está disponible y porque es conocido, un circuito eficiente se puede implementar para demostrar la capacidad del desarrollador. Después de algún tiempo el desarrollador debe familiarizarse con estas selecciones y olvidarse de las razones por las que son inadecuadas.

Construir un prototipo implica tomar en cuenta los siguientes pasos:

1. Determinar si el sistema a desarrollar es un buen candidato para la construcción del prototipo.
2. Si el prototipo es aceptado se debe desarrollar una representación abreviada de los requisitos.
3. Crear un conjunto abreviado de especificaciones de diseño para el prototipo.
4. Creación del prototipo, prueba y refinamiento.

5. Una vez probado el prototipo, según la estrategia de evaluación seleccionada, se debe presentar el proyecto para que "conduzca una prueba" de aplicación y sugiera modificaciones.
6. Repetición de los pasos cuatro y cinco.

2.2.2. CORRECCIÓN DEL PROTOTIPO.

Durante la construcción del prototipo se deja abierta la posibilidad de realizar ajustes en pro de ir logrando por aproximaciones sucesivas, mejoras hasta obtener lo deseado.

2.2.3. EVALUACIÓN.

La evaluación se debe hacer constantemente y se debe llevar a cabo para identificar puntos débiles y fuertes del Prototipo; hay una evaluación continua e independientemente, esta evaluación se hace en función de los resultados que se van obteniendo durante todo el proceso.

2.2.4. EVALUACIÓN OPERACIONAL.

Es el momento en que se evalúa la manera en que funciona el sistema de control, esto incluye su facilidad de uso, tiempo de respuesta ante una necesidad o proceso, como se adecuan los formatos en que se presenta y su nivel de utilidad.

Las pruebas directas son adecuadas en este tipo de evaluación, consiste en realizar pruebas que se centran específicamente en los requisitos funcionales del sistema de control como ser:

- Funciones incorrectas o ausentes, que de alguna manera deben efectuarse para que se establezcan funciones o procesos que son elementales para el sistema.

- Errores en la interacción entre el operador y el sistema el cual determinará e influirá de una u otra manera, los resultados que se obtengan con el uso del sistema de control.
- Errores de estructura de señales, desde un punto de vista de flujo, para verificar si las señales son correctas a la hora de procesarlas.
- Errores de rendimiento, verificar si se ha logrado superar los problemas en un enfoque funcional con el uso del sistema de control.
- Errores de inicialización y terminación

En el proceso de **EVALUACIÓN**, se irá corrigiendo los errores descritos anteriormente y permitir de esta manera que el avance del prototipo siga un curso regular en cuestión a su desarrollo. Se logrará saber si los objetivos se han alcanzado en un nivel de plenitud, a través de aspectos de desempeño, es decir, tomar en cuenta un control de actuación en los procesos de pasteurización, el tiempo que tarda en un proceso en particular, el número y el tipo de errores cometidos, entre otros. De esta manera el prototipo estará en posibilidad de ser ajustado y revisado constantemente hasta obtener lo deseado.

2.2.5. PROTOTIPO TERMINADO.

Etapa en la que quedan formalizados todos los requisitos o hasta que el prototipo llegue al punto de conclusión de la evolución.

2.3. DISEÑO METODOLÓGICO.

2.3.1. ALCANCE DE LA INVESTIGACIÓN.

El estudio descriptivo permite definir qué se va medir o sobre qué se habrán de recolectar los datos; es necesario especificar quiénes deben estar incluidos en la medición, o recolección o qué contexto, hecho, ambiente, comunidad o equivalente habrá de describirse.

Para tal efecto se toma en cuenta en la planificación, actividades totalmente experimentales, debido a que se está realizando un Sistema de Control, para dar una

solución al planteamiento del problema. Se consideran las siguientes actividades experimentales:

a) Para la determinación de los requerimientos y sugerencias.

Se lo realizará mediante entrevistas orientadas al proceso de pasteurización de la leche, llevadas en forma de dialogo y no así en forma de interrogación, para recabar información mediante el uso de tablas estructuradas en tres pilares fundamentales los cuales son:

- **Proceso de Pasteurización**: enmarcando las acciones o mecanismos que utilizan las empresas para pasteurizar la leche.
- **Deficiencias**: que problemas se tiene a menudo en cuestión del control de los procesos de pasteurización.
- **Resultado esperado utilizando el prototipo**: resaltar las acciones en la que el prototipo puede colaborar o facilitar.

b) Para el análisis del prototipo.

Se toma en cuenta mediante diagramas, lo siguiente:

- **Estructura Funcional**: especificando las funciones que de alguna manera expresan las tareas necesarias que permitan la obtención del prototipo.
- **Estructura de Procesos**: sistematizar los procesos que conllevará el uso del prototipo.

c) Para diseño del prototipo.

Una vez realizado el análisis, se procederá a realizar el diseño o acoplamiento de circuitos como también en elección de componentes electrónicos.

d) Para el desarrollo del prototipo.

Una vez diseñado se procederá al montado del prototipo.

e) Para la validación del prototipo.

Se lo realizará tomando en cuenta las siguientes pruebas, siempre descritas en tablas estructuradas:

- **Pruebas de módulos con procesos de prueba**: tomando en cuenta y/o verificando si satisface los requerimientos obtenidos en la actividad, **determinación de los requerimientos y sugerencias.**
- **Pruebas de integración**: para evaluar la integración del prototipo acoplado a los procesos de pasteurización.

f) Para la evaluación del prototipo.

Se toman en cuenta evaluaciones del prototipo descritas en tablas, tanto internas como externas en los siguientes puntos

- Funcionalidad.
- Fiabilidad en densidad de defectos.
- Usabilidad.
- Operatividad.
- Efectividad en tareas.
- Efectividad en frecuencia de errores.
- Satisfacción.

2.4. MICROCONTROLADORES.

Un microcontrolador es un dispositivo electrónico capaz de llevar a cabo procesos lógicos. Estos procesos o acciones son programados en lenguaje ensamblador por el usuario, y son introducidos en este a través de un programador.

2.4.1. ARQUITECTURA BÁSICA DE LOS MICROCONTROLADORES.

2.4.1.1. ARQUITECTURA VON NEUMANN.

La arquitectura tradicional de computadoras y microprocesadores se basa en el esquema propuesto por John Von Neumann, en el cual la unidad central de proceso, o CPU, está conectada a una memoria única que contiene las instrucciones del programa y los datos.

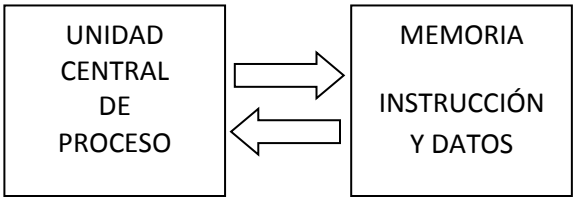


Figura 2.5.: Arquitectura Von Neumann
Fuente: Elaboración propia.

Las principales limitaciones de esta arquitectura son:

- La longitud de las instrucciones es limitada por la unidad de longitud de datos por lo que se tiene que hacer varios accesos a memoria para buscar instrucciones complejas.
- La velocidad de operación está limitada por el cuello de botella que se forma al tener un único bus de datos e instrucciones.

2.4.1.2. LA ARQUITECTURA HARVARD.

La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que el CPU está conectado a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos. Ambos buses son totalmente independientes y pueden ser de distintos anchos.

Figura 2.6.: Arquitectura Harvard
Fuente: Elaboración propia.

Las ventajas de esta arquitectura son:

a. Que el tamaño de las instrucciones no esta relacionado con el de los datos, por lo que permite que cada instrucción solo ocupe una sola posición de memoria de programa.

b. Que la velocidad de acceso de aumenta al poseer acceso de instrucciones separado del acceso a los datos.

   En contraste con una arquitectura de Von Neumann, la arquitectura Harvard es definitivamente la más óptima en velocidad, pero no en conveniencia a la hora de ejecutar grupos de instrucciones distintas. Un ejemplo de computadoras programables que usan la arquitectura Harvard, son los microcontroladores PIC.

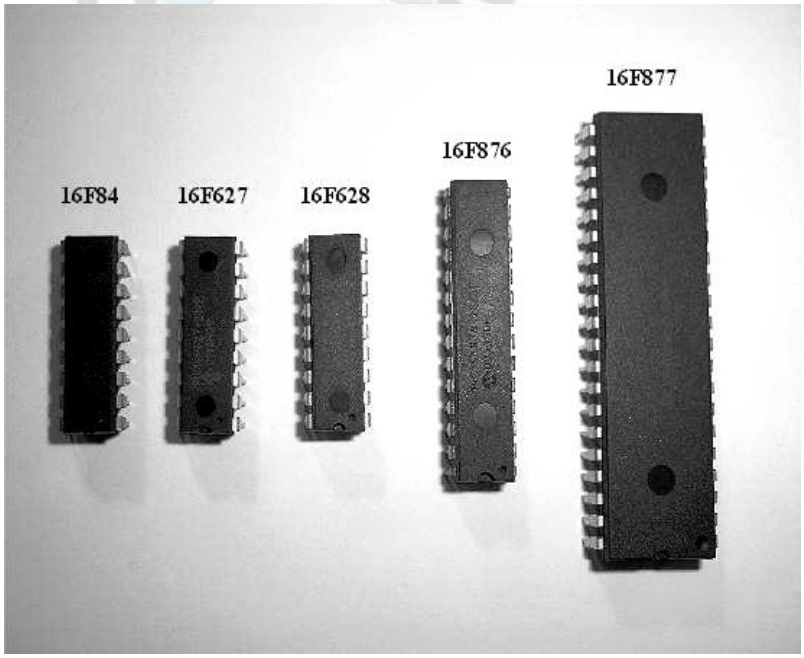


Figura 2.7.: Microcontroladores PIC.
Fuente: MICROCHIP.

2.4.2. COMPONENTES DE UN MICROCONTROLADOR.

2.4.2.1. PROCESADOR.

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. Se encarga de direccionar la memoria de instrucciones, recibir el código OP de la instrucción en curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado.

2.4.2.2. CISC.

Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores de Juego de Instrucciones Complejo). Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución. Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros, es decir, que si las tuviésemos que implementar con instrucciones básicas, acabaríamos con dolor de cabeza.

2.4.2.3. RISC.

Tanto la industria de los computadoras comerciales como la de los microcontroladores están decantándose hacia la filosofía RISC (Computadoras de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo. La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

2.4.2.4. SISC.

En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es específico, o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadoras de Juego de Instrucciones Específico).

2.4.2.5. MEMORIA.

2.4.2.5.1. ROM CON MÁSCARA.

Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip. Si tenemos idea de cómo se fabrican los circuitos integrados, sabremos de donde viene el nombre. Estos se fabrican en obleas que contienen varias decenas de chips. Estas obleas se fabrican a partir de procesos foto - químicos, donde se impregnan capas de silicio y oxido de silicio, y según convenga, se erosionan al exponerlos a la luz.

Como no todos los puntos han de ser erosionados, se sitúa entre la luz y la oblea una máscara con agujeros, de manera que donde deba incidir la luz, esta pasará. Con varios procesos similares pero más complicados se consigue fabricar los transistores y diodos micrométricos que componen un chip. El elevado coste del diseño de la máscara sólo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

2.4.2.5.2. OTP.

El microcontrolador contiene una memoria no volátil de sólo lectura programable una sola vez por el usuario. OTP (One Time Programmable). Es el usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde un PC. La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas. Tanto en este tipo de memoria como en la EPROM, se suele usar la encriptación mediante fusibles para proteger el código contenido.

2.4.2.5.3. EPROM.

Los microcontroladores que disponen de memoria EPROM (Erasable Programmable Read OnIy Memory) pueden borrarse y grabarse muchas veces. La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde un PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos.

Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico.

2.4.2.5.4. EEPROM, E2PROM o E² PROM.

Se trata de memorias de sólo lectura, programables y borrables eléctricamente EEPROM (Electrical Erasable Programmable Read OnIy Memory). Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. Es muy cómoda y rápida la operación de grabado y la de borrado. No disponen de ventana de cristal en la superficie. Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo. El número de veces que puede grabarse y borrarse una memoria EEPROM es infinito, por lo que no es recomendable una reprogramación continúa. Son muy idóneos para la enseñanza y la Ingeniería de diseño. Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria EEPROM en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que adecuan el dispositivo a las condiciones del entorno. Este tipo de memoria es relativamente lenta.

2.4.2.5.5. FLASH.

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM. La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado. Las memorias EEPROM y FLASH son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados en circuito, es decir, sin tener que sacar el circuito integrado de la tarjeta.

2.4.2.6. PUERTAS DE E/S.

Las puertas de Entrada y Salida (E/S) permiten comunicar al procesador con el mundo exterior, a través de interfaces, o con otros dispositivos. Estas puertas, también llamadas puertos, son la principal utilidad de las patas o pines de un microprocesador. Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control.

2.4.2.7. RELOJ PRINCIPAL.

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema. Esta señal del reloj es el motor del sistema y la que hace que el programa y los contadores avancen. Generalmente, el circuito de reloj está incorporado en el Microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C. Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía y de calor generado.

2.4.2.8. RECURSOS ESPECIALES.

Los principales recursos específicos que incorporan los microcontroladores son:

- Temporizadores o **Timers**.
- Perro guardián o **Watchdog**.
- Protección ante fallo de alimentación o **Brownout**.
- Estado de reposo o de bajo consumo **(Sleep mode).**
- Conversor A/D (Analógico ->Digital).
- Conversor D/A (Digital ->Analógico).
- Comparador analógico.
- Modulador de anchura de impulsos o PWM **(Pulse Wide Modulation).**

- Puertas de E/S digitales.
- Puertas de comunicación.

A continuación se denota los recursos especiales:

2.4.2.8.1. TEMPORIZADORES O TIMERS.

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso. Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patitas del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

2.4.2.8.2. PERRO GUARDIÁN O WATCHDOG.

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicia el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El Perro Guardián consiste en un contador que, cuando llega al máximo, provoca un reset automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que resetee al Perro Guardián de vez en cuando antes de que provoque el reset. Si falla el programa o se bloquea (si cae en bucle infinito), no se refrescará al Perro guardián y, al completar su temporización, provocará el reset del sistema.

2.4.2.8.3. PROTECCIÓN ANTE FALLO DE ALIMENTACIÓN O BROWNOUT.

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo (brownout). Mientras el voltaje de alimentación sea inferior al de brownout el dispositivo se mantiene reseteado, comenzando a

funcionar normalmente cuando sobrepasa dicho valor. Esto es muy útil para evitar datos erróneos por transiciones y ruidos en la línea de alimentación.

2.4.2.8.4. ESTADO DE REPOSO Ó DE BAJO CONSUMO.

Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los microcontroladores disponen de una instrucción especial (SLEEP en los PIC), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se congelan sus circuitos asociados, quedando sumido en un profundo sueño. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

2.4.2.8.5. CONVERSOR A/D (CAD).

Los microcontroladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patillas del circuito integrado.

2.4.2.8.6. CONVERSOR D/A (CDA).

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de las patillas del chip. Existen muchos circuitos que trabajan con señales analógicas.

2.4.2.8.7. COMPARADOR ANALÓGICO.

Algunos modelos de microcontroladores disponen internamente de un Amplificador Operacional que actúa como Comparador entre una señal fija de referencia y otra variable que se aplica por una de las patillas de la cápsula. La salida del Comparador

proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra. También hay modelos de microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia que se pueden aplicar en los comparadores.

2.4.2.8.8. MODULADOR DE ANCHURA DE IMPULSOS O PWM.

Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patillas del encapsulado.

2.4.2.8.9. PUERTOS DIGITALES DE E/S.

Todos los microcontroladores destinan parte de su patillaje a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos. Las líneas digitales de las Puertos pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

2.4.2.8.10. PUERTAS DE COMUNICACIÓN.

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos.

Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

- **UART.** Adaptador de comunicación serie asíncrona. Por ejemplo el Puerto Serie.
- **USART.** Adaptador de comunicación serie síncrona y asíncrona
- **Puerta paralela esclava.** Para poder conectarse con los buses de otros microprocesadores.
- **USB (Universal Serial Bus).** Que es un moderno bus serie para los PC.

- **Bus I2C.** Que es un interfaz serie de dos hilos desarrollado por Philips.
- **CAN (Controller Área Network).** Para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE.UU. se usa el J185O.

2.4.3. PROGRAMACIÓN DEL MICROCONTROLADOR.

Programar el Microcontrolador, quiere decir que se puede planificar la manera de cómo va a funcionar, que se puede adaptar a nuestras necesidades, en otras palabras que el integrado es capaz de modificar su comportamiento en función de una serie de instrucciones que es posible comunicarle.

2.4.3.1. EDICIÓN DEL CÓDIGO.

Editar es escribir el programa, es hacer una lista de instrucciones en un lenguaje que nos permita indicarle al Microcontrolador lo que deseamos que haga. Existen varios lenguajes como: Ensamblador, Basic, C, entre otros.

Todos ellos pretenden acercarse a nuestra manera de pensar y de hablar. Sin embargo los Microcontroladores no conocen más que unos y ceros.

2.4.3.2. COMPILACIÓN DEL CÓDIGO.

Compilar es traducir el programa al lenguaje de máquina que "entiende" el Microcontrolador. Para realizar esta traducción se hace uso de un software que transforma el "Programa Fuente", aquel que se edita, en otro que si se puede comunicar al Microcontrolador.

2.4.3.3. "QUEMAR" EL MICROCONTROLADOR.

Mediante una tarjeta electrónica y un software especifico, se pasa el programa compilado de la PC al Microcontrolador. Son solamente unos cuantos pasos el que se

necesita, frecuentemente se llama Programador del Microcontrolador a la tarjeta electrónica que transfiere el programa compilado de la PC al PIC.

2.4.3.4. PROBAR EL PROGRAMA.

En éste proceso se trata de verificar el funcionamiento del programa, comprobando que el Microcontrolador se comporta como se lo ha programado. Para realizar esta actividad se puede hacer uso de un Protoboard, instalar la fuente de alimentación, entre otros.

2.4.4. MICROCONTROLADOR PIC18F4550.

2.4.4.1. FAMILIA PIC18.

La familia del microcontrolador PIC18 tienen las siguientes características fundamentales:

- Arquitectura RISC avanzada Harvard 16 bits con 8 bits de datos.
- 77 instrucciones
- Desde 18 a 80 pines
- Hasta 64Kbytes de programa.
- Multiplicador Hardware 8x8
- Hasta 3968 bytes de RAM y 1KBytes de EEPROM
- Frecuencia máxima de reloj 40MHz, hasta 10 MIPS
- Pila de 32 niveles
- Múltiples fuentes de interrupción
- Periféricos de comunicación avanzados (CAN y uSB)

2.4.4.2. PATILLAJE DEL MICROCONTROLADOR PIC 18F4550.

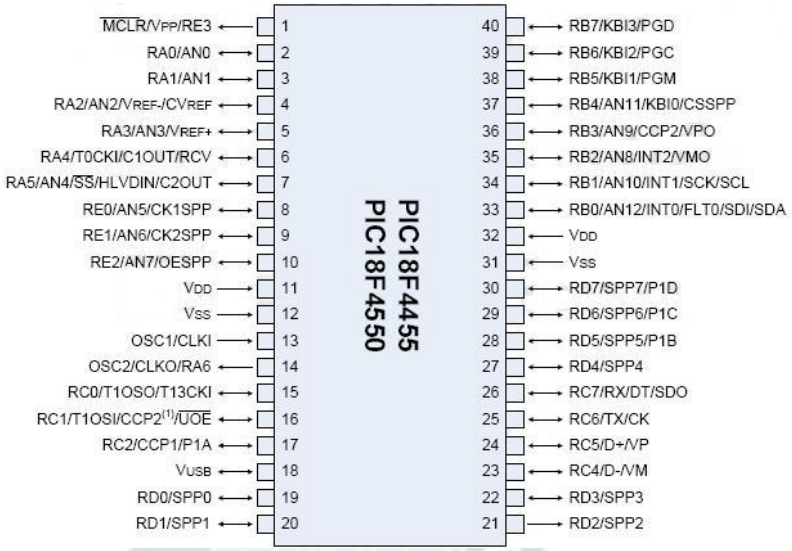


Figura 2.8.: Patillaje del PIC18F4550
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

2.4.4.3. CARACTERÍSTICAS DEL PIC18F4550.

CARACTERÍSTICA	VALOR
Frecuencia de Operación	Hasta 48MHz
Memoria de Programa (bytes)	32.768
Memoria RAM de Datos (bytes)	2.048
Memoria EEPROM Datos (bytes)	256
Interrupciones	20
Líneas de E/S	35
Temporizadores	4
Módulos de Comparación/Captura/PWM (CCP)	1
Módulos de Comparación/Captura/PWM mejorado (ECCP)	1
Canales de Comunicación Serie	MSSP,EUSART
Canal USB	1
Puerto Paralelo de Transmisión de Datos (SPP)	1
Canales de Conversión A/D de 10 bits	13 Canales
Comparadores analógicos	2
Juego de instrucciones	75 (83 ext.)
Encapsulados	PDIP 40 pines QFN 40 pines TQFP 40 pines

Cuadro 2.1. Características del PIC18F4550.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

2.4.4.4. ORGANIZACIÓN DE MEMORIA.

El Microcontrolador PIC18F4550 dispone de las siguientes memorias:

a) Memoria de programa: memoria flash interna de 32.768 bytes:
   - Almacena instrucciones y constantes/datos.
   - Puede ser escrita/leída mediante un programador externo o durante la ejecución programa mediante unos punteros.

b) Memoria RAM de datos: memoria SRAM interna de 2048 bytes en la que están incluidos los registros de función especial.
   - Almacena datos de forma temporal durante la ejecución del programa.
   - Puede ser escrita/leída en tiempo de ejecución mediante diversas instrucciones.

c) Memoria EEPROM de datos: memoria no volátil de 256 bytes.
   - Almacena datos que se deben conservar aun en ausencia de tensión de alimentación.
   - Puede ser escrita/leída en tiempo de ejecución a través de registros.

d) Pila: bloque de 31 palabras de 21 bits
   - Almacena la dirección de la instrucción que debe ser ejecutada después de una interrupción o subrutina

e) Memoria de configuración: memoria en la que se incluyen los bits de configuración (12 bytes de memoria flash) y los registros de identificación (2 bytes de memoria de solo lectura).

2.4.4.5. MEMORIA DE CONFIGURACIÓN.

Se trata de un bloque de memoria situado a partir de la posición 30000H de memoria de programa (más allá de la zona de memoria de programa de usuario). En esta memoria de configuración se incluyen:

a) Bits de configuración: contenidos en 12 bytes de memoria flash permiten la configuración de algunas opciones del uC como:
    - Opciones del oscilador.
    - Opciones de reset.
    - Opciones del watchdog.
    - Opciones de la circuiteria de depuración y programación.
    - Opciones de protección contra escritura de memoria de programa y memoria EEPROM de datos.

Estos bits se configuran generalmente durante la programación del microcontrolador, aunque también pueden ser leídos y modificados durante la ejecución del programa.

b) Registros de identificación: se trata de dos registros situados en las direcciones 3FFFFEH y 3FFFFFH que contienen información del modelo y revisión del dispositivo. Son registros de solo lectura y no pueden ser modificados por el usuario.

2.4.4.6. ARQUITECTURA HARDVARD PIC18F4550.

El Microcontrolador PIC18F4550 dispone de diferentes buses para el acceso a memoria de programa y memoria de datos (arquitectura Harvard):

a) Bus de la memoria de programa:
    - 21 líneas de dirección.
    - 16/8 líneas de datos (16 líneas para instrucciones/8 líneas para datos)

b) Bus de la memoria de datos:
    - 12 líneas de dirección
    - 8 líneas de datos

c) Esto permite acceder simultáneamente a la memoria de programa y a la memoria de datos. Es decir se puede ejecutar una instrucción (lo que por lo general requiere acceso a memoria de datos) mientras se lee de la memoria de programa la siguiente instrucción (proceso pipeline).

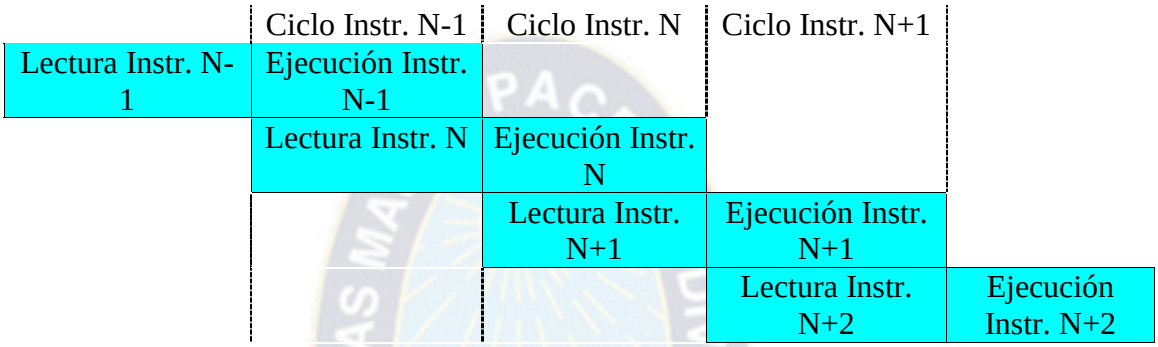
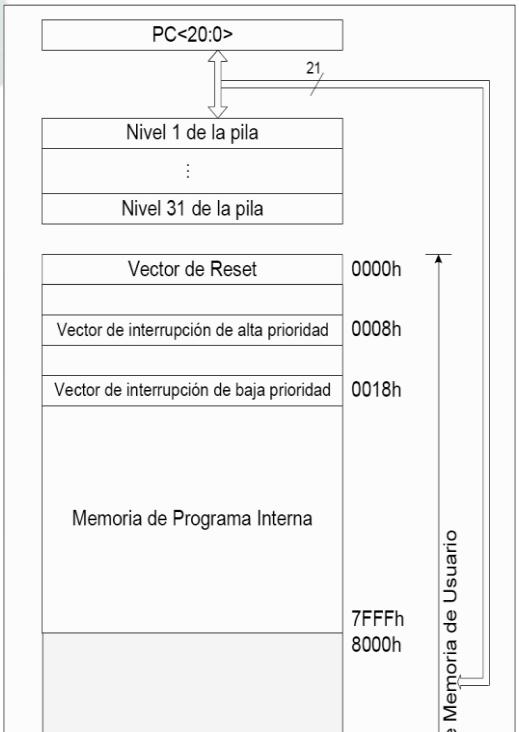


Figura 2.9. Lectura y Ejecución de la Instrucción.
Fuente: Usategui, Angulo.

d) Por tanto la ejecución completa de 1 instrucción (lectura instrucción más ejecución) se hace en un 1 ciclo de instrucción (4 TOSC). EXCEPCION: las instrucciones que modifican el contenido del PC requieren 2 ciclos de instrucción.

2.4.4.7. MEMORIA DE PROGRAMA.

El Microcontrolador PIC18F4550 dispone una memoria de programa de 32.768 bytes de memoria de programa (0000H - 7FFFH). Las instrucciones ocupan 2 bytes (excepto CALL, MOVFF, GOTO y LSFR que ocupan 4, por ejemplo). Por lo tanto la memoria de programa puede almacenar hasta 16.384 instrucciones.



La operación de lectura en posición de memoria por encima de 7FFFH da ´0´ como

resultado (equivalente a la instrucción NOP).Las direcciones especiales de la memoria de programa son:

- Vectorización del Reset es 0000H
- Vectorización de las interrupciones de alta prioridad es la 0008H.
- Vectorización de las interrupciones de baja prioridad es la 0018H.

Figura 2.10.: Memoria de Programa
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

2.4.4.8. ALMACENAMIENTO DE INST PROGRAMA.

	Memoria de Programa	
MOVLW 55H	**55H**	0020H
	0FH	0021H
CPFSEQ 20H	**20H**	0022H
	63H	0023H
	88H	0024H
GOTO 0110H	**EFH**	0025H
	00H	0026H
	F0H	0027H
INCF 20H	**20H**	0028H
	2BH	0029H

Figura 2.11.: Almacenamiento de Instrucciones.
Fuente: Usategui, Angulo.

Primero se almacena la parte baja de la instrucción y luego la parte alta (para las instrucciones de 4 bytes primero los bytes menos significativos y luego los más significativos).Las instrucciones siempre empiezan en direcciones pares.

2.4.4.9. CONTADOR DE PROGRAMA (PC).

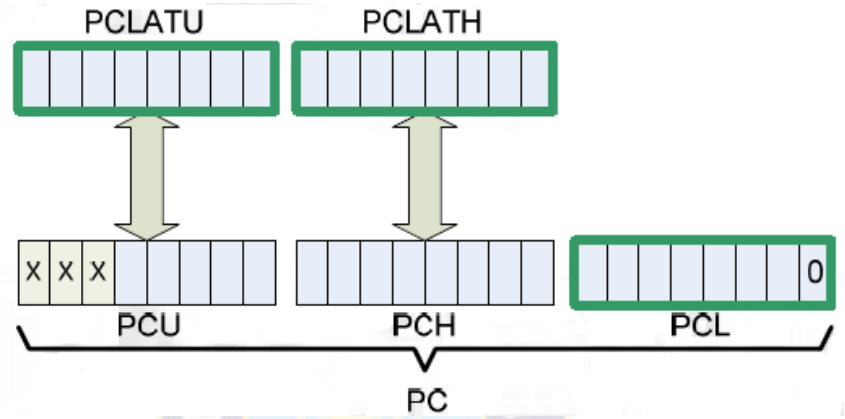


Figura 2.12.: Contador de Programa.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

Puntero de 21 bits que indica la dirección en memoria de programa de la instrucción que se debe ejecutar. Está compuesto por 3 bytes:

- PCU: parte superior del PC, registro no directamente accesible; las operaciones de lectura/escritura sobre este registro se hacen a través del registro PCLATU.
- PCH: parte alta del PC, registro no directamente accesible; las operaciones de lectura/escritura sobre este registro se hacen a través del registro PCLATH.
- PCL: parte baja del PC, registro directamente accesible. Una operación de lectura sobre PCL provoca que los valores de PCU y PCH pasen a PCLATU y PCLATH respectivamente. Y una operación de escritura sobre PCL provoca que los valores de PCLATU y PCLATH pasen a PCU y PCH respectivamente. El PCL siempre tiene el bit menos significativo a '0', debido a que las instrucciones siempre empiezan en direcciones pares.

2.4.4.10. LA PILA DE DIRECCIÓN.

La Pila es un bloque de memoria RAM independiente de 31 palabras de 21 bits que sirve para almacenar temporalmente el valor del PC cuando se produce una llamada a subrutina o una interrupción.

El puntero de pila (contenido en el registro STKPTR) es un contador de 5 bits que indica la posición actual del final de pila. El contenido del final de pila es accesible mediante los registros TOSU, TOSH, TOSL. Cuando se procesa una interrupción o se ejecutan las instrucciones las instrucciones CALL o RCALL (el PC está apuntando a la siguiente instrucción) se incrementa el STKPR y se almacena en el final de pila el valor del PC.

Cuando se ejecutan las instrucciones RETURN, RETLW o RETFIE se copia el valor almacenado en la cima de pila en el PC y se decrementa el STKPTR.

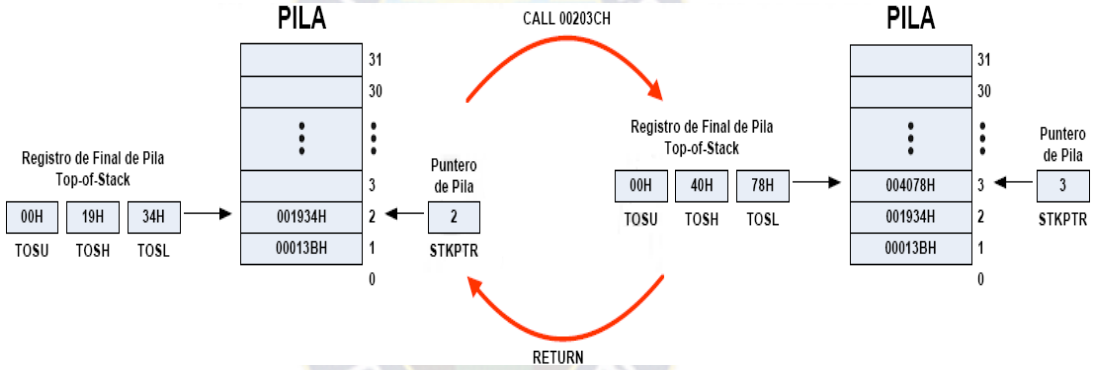


Figura 2.13.: Pila de Dirección.
Fuente: Manual del Referencia PIC18F4550, MICROCHIP.

2.4.4.10.1. REGISTRO STKPTR.

	L/R-0	L/R-0	-0	L/E-0	*L/E-0*	L/E-0	L/E-0	L/E-0
STKPTR	**STKFUL**	**STKUNF**	**-**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**

Figura 2.14.: Registro STKPTR.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

a) **STKFUL:** Flag de llenado de la pila (en modo escritura únicamente puede ser puesto a '0')
   - STKFUL='0': No se ha producido el llenado o desbordamiento de la pila.
   - STKFUL='1': Se ha producido el llenado o desbordamiento de la pila.

b) **STKUNF:** Flag de vaciado de la pila (en modo escritura únicamente puede ser puesto a ‘0’).
   - STKUNF=‘0’: No se ha producido el desbordamiento por vaciado de la pila.
   - STKUNF=‘1’: Se ha producido el desbordamiento por vaciado de la pila.

c) **SP4..SP0:** Puntero de pila. Estos 5 bits indican la posición del final de la pila (valor de 0 a 31).

2.4.4.11. LLENADO Y VACIADO DE LA PILA DE DIRECCIONES.

2.4.4.11.1. LLENADO DE LA PILA.

En el llenado de la Pila, si la pila llega al máximo de su capacidad (31 elementos apilados):

- Si el bit de configuración STRVEN está a ‘0’: el bit STKFUL del registro STKPTR se pone a ‘1’ y si se producen nuevos apilamientos no afectarán a la pila.
- Si el bit de configuración STRVEN está a ‘1’: el bit STKFUL del registro STKPTR se pone a ‘1’ y se producirá un reset del Microcontrolador.

2.4.4.11.2. VACIADO DE LA PILA.

En el vaciado de la Pila, si la pila está vacía y se intenta desapilar de nuevo:

- Si el bit de configuración STRVEN está a ‘0’: el bit STKUNF del registro STKPTR se pone a ‘1’, el PC se pondrá a 0000H y Puntero de pila permanecerá a 0.
- Si el bit de configuración STRVEN está a ‘1’: el bit STKUNF del registro STKPTR se pone a ‘1’ y se producirá un reset del Microcontrolador.

2.4.4.12. PILA RAPIDA DE REGISTRO.

Se trata de una pila de un solo nivel en la que se apilan los valores del registro de estado, del W y del registro BSR cada vez que se produce una interrupción. Estos valores pueden ser recuperados si al salir de la interrupción se utiliza la instrucción "RETFIE, FAST". Si están habilitadas las interrupciones de baja y alta prioridad, esta pila no debe ser utilizada en interrupciones de baja prioridad. Si no hay interrupciones habilitadas esta pila puede ser utilizada en llamadas a subrutinas ("CALL, FAST, RETURN, FAST).

2.4.4.13. MEMORIA RAM DE DATOS.

El uC PIC18F4550 dispone una memoria RAM de datos 2.048 bytes (8 bancos de 256 bytes). Además dispone de 160 bytes dedicados a los registros de función especial (SFR's) situados en la parte alta del banco 15.

Para acceder a un byte de la memoria RAM de datos primero debe seleccionarse el banco al que pertenece el byte mediante el registro de selección de banco (BSR) y a continuación direccionar el byte dentro del banco. Además existe una modalidad de acceso rápido a las 96 posiciones de la parte baja del banco 0 y a los 160 bytes de SFR's (banco de acceso rápido). Los bancos 4, 5, 6 y 7 se utilizan también para el USB.

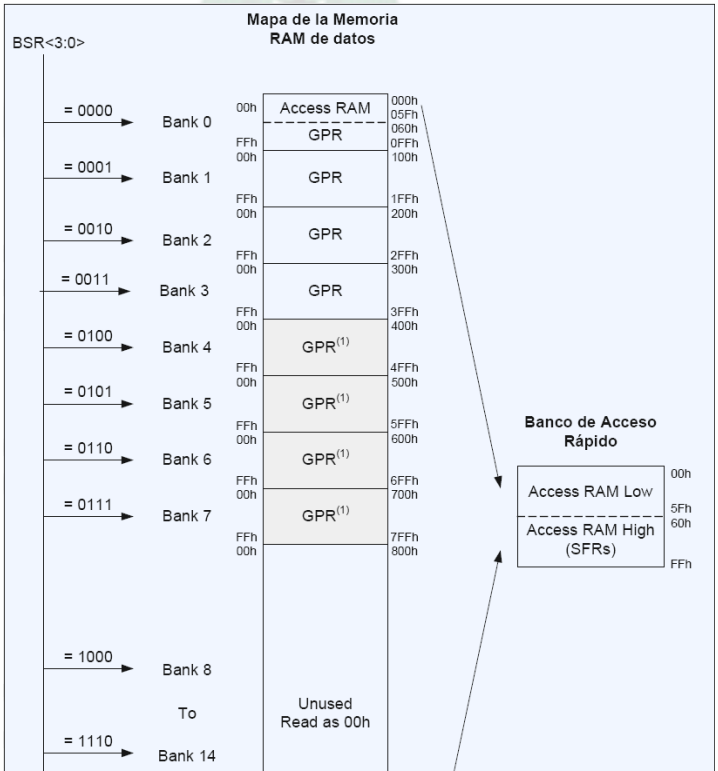


Figura 2.15.: Memoria RAM de Datos.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP

.

2.4.4.13.1. REGISTROS DE FUNCION ESPECIAL.

La memoria RAM de datos se compone de registros de propósito general (GPR's) y de registros de función especial (SFR's). Los SFR's son los registros mediante los cuales se pueden monitorizar/controlar el funcionamiento de la CPU y de las unidades funcionales del Microcontrolador.

Se distinguen dos conjuntos de SFR's:

a) SFR's asociados con el núcleo del Microcontrolador.
   - CPU: WREG, STATUS, BSR, entre otros.
   - Interrupciones: INTCON, PIE1, PIR1, IPR1, entre otros.
   - Reset: RCON.

b) SFR's asociados con las unidades funcionales.
   - Timers: T0CON, TMR1H, TMR1L, T1CON, entre otros.
   - Convertidor A/D: ADRESH, ADRESL, ADCON0, ADCON1, entre otros.
   - EUSART: TXREG, TXSTA, RCSTA, entre otros.
   - CCP: CCPR1H, CCPR1L, CCP1CON, entre otros.
   - MSSP: SSPSTAT, SSPDATA, SSPCFG, entre otros.
   - Puertos de E/S: TRISA, PORTA, TRISB, PORTB, entre otros.

2.4.4.14. SISTEMA DE INTERRUPCIONES.

El Microcontrolador PIC18F4550 dispone de 21 fuentes de interrupciones. Se distinguen dos grupos de interrupciones:

a) Grupo general de interrupciones:

Interrupción del Temporizador 0
Interrupción por cambio en PORTB
Interrupción externa 0
Interrupción externa 1
Interrupción externa 2

Cuadro 2.2. Interrupciones de grupo general.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

b) Grupo de Interrupciones de Periféricos:

Interrupción del SPP	Interrupción de fallo del oscilador
Interrupción del A/D	Interrupción del comparador
Interrupción de recepción de la EUSART	Interrupción del USB
Interrupción de transmisión de la EUSART	Interrupción de escritura en Flash/EEPROM
Interrupción del MSSP	Interrupción de colisión del bus (MSSP)
Interrupción del CCP1	Interrupción de detección de anomalías en V_{DD}
Interrupción del Temporizador 1	Interrupción del Temporizador 3
Interrupción del Temporizador 2	Interrupción del CCP2

Cuadro 2.3. Interrupciones de Periféricos.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

Se dispone de dos niveles de prioridad:

- Nivel alto, vectorizado en la dirección 0008H
- Nivel bajo, vectorizado en la dirección 0018H

Todas las interrupciones pueden ser programadas con cualquiera de las dos prioridades, salvo la interrupción externa 0 (que siempre tiene alta prioridad).

Todas las interrupciones disponen de 3 bits de configuración (excepto la interrupción externa 0 que tiene dos):

- Bit de habilitación de interrupción: permite habilitar a nivel individual la interrupción.
- Flag de interrupción: se pone a '1' cuando se produce la condición de interrupción independientemente de si la interrupción está habilitada o no. Este flag debe ponerse '0' por software cuando se procesa la interrupción.
- Bit de prioridad de interrupción: establece si la interrupción es de alta o de baja prioridad (este bit no está disponible para la interrupción externa 0).

2.4.4.15. PUERTOS DE ENTRADA Y SALIDA.

El Microcontrolador PIC18F4550 dispone 5 puertos de E/S que incluyen un total de 35 líneas digitales de Entrada / Salida:

PUERTO	LINEAS DE ENTRADA/SALIDA
PORTA	7 LINEAS DE ENTRADA/SALIDA
PORTB	8 LINEAS DE ENTRADA/SALIDA
PORTC	6 LINEAS DE ENTRADA/SALIDA+ 2 LINEAS DE ENTRADA
PORTD	8 LINEAS DE ENTRADA/SALIDA
PORTE	3 LINEAS DE ENTRADA/SALIDA + 1 LINEAS DE ENTRADA

Cuadro 2.4. Puertos de Entrada y Salida.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

Todas las líneas digitales de Entrada / Salida disponen de al menos una función alternativa asociada a alguna circuiteria especifica del Microcontrolador. Cuando una línea trabaja en el modo alternativo no puede ser utilizada como línea digital de Entrada / Salida estándar.

2.4.4.15.1. PUERTO A.

Dispone de 7 líneas de E/S. Las funciones alternativas son:

- RA0: entrada analógica (AN0)/ entrada de comparación (C1IN-)
- RA1: entrada analógica (AN1)/ entrada de comparación (C2IN-)
- RA2: entrada analógica (AN2)/ entrada de comparación (C2IN+)
- RA3: entrada analógica (AN3)/ entrada de comparación (C1IN+)
- RA4: entrada de reloj del Temporizador 0 (T0CKI)/salida de comparación (C1OUT)
- RA5: entrada analógica (AN4)/ salida de comparación (C2OUT)/HLVDIN entrada de detección de tensión alta/baja.
- RA6: entrada del oscilador principal (OSC2)/salida de señal de reloj (CLK0)

En el reset las líneas RA0, RA1, RA2, RA3 y RA5 se configuran como líneas de entrada analógicas. Para poder utilizarlas como líneas digitales de E/S hay que desactivar la función analógica.

2.4.4.15.2. PUERTO B.

Dispone de 8 líneas de E/S. Las funciones alternativas son:

- RB0: entrada analógica (AN12)/ interrupción externa 0 (INT0)/entrada de fallo del ECCP (FLT0)/entrada de datos del SPI (SDI)/línea de datos del I2C (SDA).
- RB1: entrada analógica (AN10)/ interrupción externa 1 (INT1)/línea de reloj del SPI (SDI)/línea de reloj del I2C (SDA).
- RB2: entrada analógica (AN8)/ interrupción externa 2 (INT2)/salida de datos del USB (VCMO).
- RB3: entrada analógica (AN9)/ línea de E/S del CCP2 (CCP2)/salida de datos del USB (VPO).
- RB4: entrada analógica (AN11)/ interrupción por cambio en pin (KBI0)/ salida de CS del SSP (CSSP).
- RB5: interrupción por cambio en pin (KBI1)/ línea de programación (PGM).
- RB6: interrupción por cambio en pin (KBI2)/ línea de programación (PGC).
- RB7: interrupción por cambio en pin (KBI3)/ línea de programación (PGD).

Resistencias de pull-up: Todas las líneas del puerto B disponen de resistencias de pull-up internas que pueden ser activadas poniendo el bit RBPU del registro INTCON2 a ‘0’ (RPBU=‘1’ después de un reset). Si una línea del puerto B se configura como salida la resistencia de pull-up correspondiente se desactiva automáticamente.

Por defecto, en el reset las líneas RB4..RB0 están programadas como entradas analógicas. Existen dos formas de configurar RB4..RB0 como líneas de E/S digitales:

- Poniendo a ‘0’ el bit PBADEN del registro de configuración CONFIG3H=> en el reset RB4..RB0 se configuran como líneas de E/S digitales.
- Si PBADEN=‘1’ (valor por defecto) se pueden configurar RB4..RB0 como líneas del E/S digitales desactivando la función analógica.

2.4.4.15.3. PUERTO C.

Dispone de 5 líneas de E/S (RC0, RC1, RC2, RC6 y RC7) y 2 líneas de solo entrada (RC4 y RC5). Las funciones alternativas son:

- RC0: salida del oscilador del Temp. 1 (T1OSO)/ entrada de contador de los Temp. 1 y 3 (T13CKI).
- RC1: entrada del oscilador del Temp. 1 (T1OSI)/ línea de E/S del CCP2 (CCP2)/ salida OE del transceiver del USB (UOE).
- RC2: línea de E/S del CCP1 (CCP1)/ salida PWM del ECCP1 (P1A).
- RC4: línea menos del bus USB (D-) / línea de entrada del USB (VM).
- RC5: línea más del bus USB (D-) / línea de entrada del USB (VP).
- RC6: salida de transmisión del EUSART (TX)/ línea de reloj del EUSART (CK)
- RC7: entrada de recepción del EUSART (RX)/ línea de datos síncrona del EUSART (DT)/ salida de datos del SPI (SDO)

En el reset todas las líneas del puerto C quedan configuradas como entradas digitales.

2.4.4.15.4. PUERTO D.

Dispone de 8 líneas de E/S. Las funciones alternativas son:

- RD0: línea de datos del SPP (SPP0)
- RD1: línea de datos del SPP (SPP1)

- RD2: línea de datos del SPP (SPP2)
- RD3: línea de datos del SPP (SPP3)
- RD4: línea de datos del SPP (SPP4)
- RD5: línea de datos del SPP (SPP5) / salida PWM del ECCP1 (P1B)
- RD6: línea de datos del SPP (SPP6) / salida PWM del ECCP1 (P1C)
- RD7: línea de datos del SPP (SPP7) / salida PWM del ECCP1 (P1D)

Resistencias de pull-up: Todas las líneas del puerto D disponen de resistencias de pull-up internas que pueden ser activadas poniendo el bit RDPU del registro PORTE a '1' (RPDU='0' después de un reset). Si una línea del puerto D se configura como salida la resistencia de pull-up correspondiente se desactiva automáticamente.

2.4.4.15.5. PUERTO E.

Dispone de 3 líneas de E/S (RE0, RE1 y RE2) y 1 línea de solo entrada (RE3). Las funciones alternativas son:

- RE0: entrada analógica (AN5)/ salida de reloj 1 del SPP (CK1SPP)
- RE1: entrada analógica (AN6)/ salida de reloj 2 del SPP (CK2SPP)
- RE2: entrada analógica (AN7)/ salida de habilitación del SPP (OESPP)
- RE3: Línea de reset externo (MCLR) / línea de programación (VPP)

En el reset todas las líneas RE2..RE0 se configuran como entradas analógicas. Para poder utilizarlas como líneas digitales de E/S hay que desactivar la función analógica. La línea RE3 por defecto tiene la función de Reset del Microcontrolador. Si se desea desactivar la función de Reset y utilizar RE3 como línea de entrada digital hay que poner a '0' el bit MCLRE del registro de configuración CONFIG3H.

2.4.4.16. CONVERSOR ANALÓGICO – DIGITAL.

Las características fundamentales son:

- 10 bits de resolución.
- 13 canales multiplexados.

- Señal de reloj de conversión configurable.
- Tiempo de adquisición programable (0 a 20TAD).
- Posibilidad de establecer el rango de tensiones de conversión mediante tensiones de referencia externas.

2.4.4.16.1. REGISTRO ADCON0.

	-0	-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0
ADCON0	-	-	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON

Figura 2.16.: Registro ADCON0.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

a) **CHS3..CHS0:** Bits selección del canal de conversión A/D (13 canales)

b) **GO/DONE:** Bit de inicio y de monitorización del estado de la conversión A/D:
   - GO/DONE='0': Proceso de conversión parado.
   - GO/DONE='1': Proceso de conversión en marcha.

c) **ADON:** Bit de habilitación del convertidor A/D.
   - ADON='0': Convertidor A/D desactivado.
   - ADON='1': Convertidor A/D activado.

2.4.4.16.2. REGISTRO ADCON1.

	-0	-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0
ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0

Figura 2.17.: Registro ADCON1.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

a) **VCFG1:** Bit de configuración de la tensión de referencia V_{REF-}:
   - VCFG1='0': V_{REF-} se conecta a V_{SS}.
   - VCFG1='1': V_{REF-} se conecta a la línea física RA2

b) **VCFG0:** Bit de configuración de la tensión de referencia V_{REF+}:
   - VCFG1='0': V_{REF+} se conecta a V_{DD}.
   - VCFG1='1': V_{REF+} se conecta a la línea física RA3.

c) **PCFG3..PCFG0:** Bits configuración de los puertos de conversión A/D. Mediante estos bits se establecen las líneas físicas (RA5..RA0, RB4..RB0, RE1 y RE0) que van a trabajar como entradas del convertidor A/D.

2.4.4.16.3. REGISTRO ADCON2.

	L/E-0	-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0
ADCON2	**ADFM**	**-**	**ACQT2**	**ACQT1**	**ACQT0**	**ADCS2**	**ADCS1**	**ADCS0**

Figura 2.18.: Registro ADCON2.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

a) **ADFM:** Bit de configuración del tipo de almacenamiento del resultado de la conversión en los registros ADRESH y ADRESL:
   - ADFM='0': El resultado de la conversión se almacena con justificación a izquierdas.
   - ADFM='1': El resultado de la conversión se almacena con justificación a derechas.

b) **ACQT2..ACQT0:** Bits de configuración del tiempo de adquisición

c) **ADCS2..ADCS0:** Bits selección de la señal de reloj del convertidor A/D

2.4.4.16.4. SELECCIÓN DEL CANAL DE CONVERSIÓN

Para que uno de los 13 canales pueda ser seleccionado, previamente debe haber sido configurado como entrada analógica mediante los bits PCFG3..PCFG0 del registro ADCON1 (A: analógico / D: digital).

PCFG3... PCFG0	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

Cuadro 2.5. Cuadro de Configuración del Canal de Conversión,
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

Una vez configurado como línea de entrada analógica, un canal puede ser seleccionado mediante los bits CHS3..CHS0 del registro ADCON0.

CHS3	CHS2	CHS1	CHS0	CANAL SELECCIONADO
0	0	0	0	CANAL AN0 (RA0)
0	0	0	1	CANAL AN1 (RA1)
0	0	1	0	CANAL AN2 (RA2)
0	0	1	1	CANAL AN3 (RA3)
0	1	0	0	CANAL AN4 (RA5)
0	1	0	1	CANAL AN5 (RE0)
0	1	1	0	CANAL AN6 (RE1)
0	1	1	1	CANAL AN7 (RE2)

1	0	0	0	CANAL AN8 (RB2)
1	0	0	1	CANAL AN9 (RB3)
1	0	1	0	CANAL AN10 (RB1)
1	0	1	1	CANAL AN11 (RB4)
1	1	0	0	CANAL AN12 (RB0)
1	1	0	1	No implementado
1	1	1	0	No implementado
1	1	1	1	No implementado

Cuadro 2.6. Cuadro de Selección de Canal de Conversión.
Fuente: Manual de Referencia PIC18F4550, MICROCHIP.

2.4.4.17. ESTÁNDAR USB.

2.4.4.17.1. INTERFAZ FÍSICA.

La interfaz física está formada por cuatro hilos dos para la alimentación 5v (Rojo) GND (Negro) y dos para datos D+ (verde) y D- (Blanco), lo del signo positivo y negativo, es porque es una señal diferencial cuyo valor depende de la velocidad del bus (3,3v para low-speed y 400mV para high-speed).

A los conectores USB, se los clasifica de dos tipos: La primera de tipo A y la segunda de tipo B.

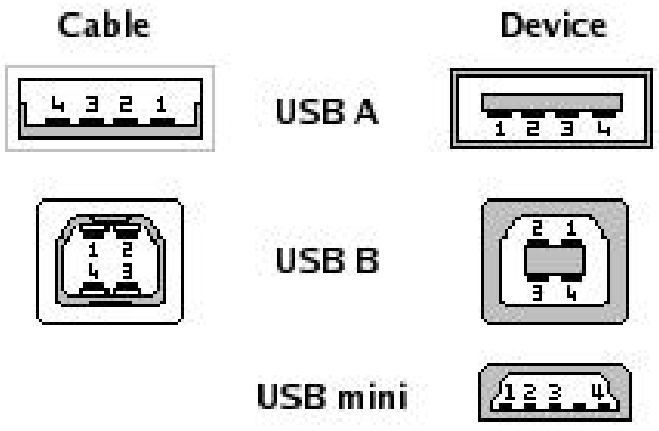


Figura 2.19.: Tipos de Interfaz Física USB.
Fuente: Jhon Gamey

PIN	SEÑAL	COLOR	DESCRIPCIÓN
1	V_{CC}	ROJO	+5 V
2	D –	BLANCO	DATO –
3	D +	VERDE	DATO +
4	GND	NEGRO	GROUND

Cuadro 2.7. Cuadro Descriptivo de la Interfaz USB
Fuente: Jhon Gamey

2.4.4.17.2. VELOCIDADES DEL BUS.

Es necesario indicar que el Host es el que controla la velocidad en la que circulan los datos en el bus USB y que al ser un bus compartido, la velocidad real dependerá de la cantidad de dispositivos que tengamos conectados a el en un momento determinado (máximo 127 incluyendo al Host). Por tanto los datos siguientes son solo teóricos y de referencia.

- **LOW SPEED**: 1,5 Mbps. Soportado por las especificaciones 1.1, 2.0 y 3.0. Es la velocidad utilizada por dispositivos como teclados, ratones, joystick, etc.
- **FULL SPEED**: 12 Mbps. Soportado por USB 1.1, USB 2.0 y USB 3.0. Un ejemplo donde se utilizan estas velocidades es en transmisiones de audio.
- **HIGH SPEED**: 480 Mbps. Solo USB 2.0 y USB 3.0. Ejemplo transmisiones de video.
- **SUPER SPEED**: 5Gbps solo soportado en dispositivos USB 3.0.

2.4.4.17.3. TRANSFERENCIAS.

Una transferencia se puede definir como el conjunto global de los datos que forman una comunicación USB, una transferencia está formada a su vez por una o varias transacciones que a su vez están formadas por diferentes paquetes de datos que contienen las tramas de una comunicación USB.

No existe un formato único de transferencia, la especificación USB permite cuatro tipos de transferencias:

a) **CONTROL**: se utilizan para configurar y enviar comandos, por ejemplo en la enumeración del dispositivo.

b) **BULK (masivas)**: se utilizan cuando se precisa una transferencia de datos grandes, es el tipo más rápido de transferencia, sin embargo no hay garantía de que los datos se transmitan en un tiempo determinado (no garantizada la latencia). Si se verifica que los datos se han transmitido con éxito ya que dispone del Sistema de Corrección de Errores (CRC), esta transferencia solo la pueden utilizar dispositivos que soporten velocidades Full y High Speed. Este tipo de transferencia es utilizada por dispositivos como por ejemplo: discos duros, pen drivers, escaners, impresoras, entre otros.

c) **ISÓCRONAS**: Es usada en dispositivos que transmiten señales de audio y de vídeo en tiempo real. Se garantiza una tasa de velocidad de transmisión determinada (latencia asegurada). Si no fuera así, por ejemplo en una transmisión de voz el audio se oiría entrecortado. No contempla la corrección de errores, si en un archivo de sonido se pierde un BIT, no es importante su recuperación. Para usar este tipo de transferencia es necesario que los dispositivos soporten velocidades Full Speed.

d) **INTERRUPCIÓN**: latencia asegurada y verificación de que los datos se han transmitido con éxito, Se utiliza en dispositivos como: Teclados, Mouse, Sensores, Pantallas táctiles y dispositivos que no requieran mucho ancho de banda.

Los dispositivos usan uno o más tipos de transferencia, la de control es utilizada siempre por todos los dispositivos en el proceso de enumeración. Si se utiliza otra transferencia habrá que hacerlo en función del tipo y cantidad de datos a transmitir.

2.4.4.17.4. ENUMERACIÓN.

El Host es el encargado de detectar cualquier dispositivo que se conecta al bus. Cuando un dispositivo es detectado el Host necesita obtener información sobre él, a este proceso es al que se le llama **enumeración**. Esta información que necesita el Host se encuentra definida en el dispositivo en los llamados descriptores. Los descriptores son datos que se guardan en la memoria no volátil del PIC y contienen la siguiente información: El ID del vendedor (VID) y del producto (PID), consumo de corriente del dispositivo, tipo de transferencia que se va ha utilizar, endpoint utilizados, versión USB soportada, clase utilizada, entre otros.

El **VID** (Vendor ID) y el **PIC** (Product ID) son dos números de 16 bits representados en Hexadecimal, si utilizamos la clase CDC (Communications Device Class) de CCS para la comunicación USB estos valores los podemos modificar en el archivo "*usb_desc_cdc.h*" y según el sistema operativo instalado en el Host deberemos modificarlos también en el archivo de extensión "*inf*".

2.4.4.17.5. ENDPOINT.

Los endpoint son simplemente buffer de memoria RAM que son utilizados para el envío y recepción de datos o comandos de control durante una comunicación USB. Cada endpoint puede ser de entrada o salida de datos o bidireccional, el endpoint 0 está reservado para comandos de control, el proceso de enumeración se realiza a través del endpoint número 0. Este concepto solo se aplica al dispositivo, en el host existen también buffer para el envío y recepción de datos pero no se les denomina con este nombre.

2.4.4.17.6. PIPE O TUBERÍA.

Es una conexión lógica entre un endpoint y el software del controlador del host que se produce tras el proceso de enumeración. Los Pipes se usan mucho en Sistemas Operativos como UNIX/LINUX para enlazar la salida de un proceso con la entrada de otro, en este caso el concepto es el mismo.

2.4.4.17.7. CLASES.

Una clase es un modelo o plantilla que describe el estado y el comportamiento de los objetos que la comparten. La clase provee de propiedades y métodos (funciones) reutilizables por los objetos o miembros que comparten la clase. Esta definición general de clase utilizada en la programación orientada a objetos para definir este concepto utilizado en USB. La especificación USB provee de propiedades y funciones que pueden ser utilizadas por los dispositivos que tengan características similares. Por ejemplo, un teclado y un ratón por sus características pertenecerán a la misma clase la llamada

Human Interface Device (HID), pues bien si se diseña el firmware de un dispositivo con las especificaciones que exige esta clase, se podrá beneficiar de esas propiedades y funciones comunes a la clase, una ventaja de utilizar esta clase por ejemplo es que no se necesita instalar ningún driver para el dispositivo, ya que el sistema operativo utilizará uno genérico para todos.

Las clases más utilizadas con Microcontroladores son:

a) **HID (Human Interface Device)**: ejemplos de dispositivos que utilizan esta clase como son: teclados, ratones, pantallas táctiles, joystick, entre otros. Velocidad low-speed (64 KB/s de velocidad máxima), tipos de transferencias soportadas: de control y de Interrupción. Una característica interesante al utilizar esta clase es que no se necesita instalar un driver específico en el Sistema Operativo, se utiliza uno estándar que ya está incluido en el sistema. En el Sistema Operativo Windows la aplicación de escritorio accede al dispositivo con ayuda de las APIS win32.

b) **MSD (Mass Storage Device Class)**: Como su propio nombre indica, es utilizado en dispositivos de almacenamiento masivo como discos duros, memorias flash, cámaras digitales, dispositivos ópticos externos como lectores y grabadoras de CD y DVD, entre otros. Esta clase se puede utilizar solo en dispositivos que soporten velocidades Full y High Speed. El tipo de transferencias utilizadas es Bulk o una combinación formada por transferencias del tipo Control, Bulk e Interrupt. Microchip tiene notas de aplicación sobre esta clase como la AN1003, CCS también implementa ejemplos sobre esta clase. No se necesita la instalación de un driver específico, se utilizan drivers genéricos instalados ya en los Sistemas Operativos, en Windows se utiliza el driver llamado usbstor.sys ubicado en C:\Windows\Sistem32\drivers.

c) **CDC (Communications Device Class)**: Un ejemplo de dispositivo que utiliza esta clase son los Modems. La velocidad máxima al utilizar esta clase será de 80 kBytes/s y el tipo de transferencias soportadas son del tipo interrupción y Bulk.

Utiliza también driver estándar incluidos ya en el Sistema Operativo, según el sistema operativo utilizado precisará o no de la instalación del archivo .INF, cuando se utiliza esta clase en la PC, se creará un puerto serie virtual y la comunicación entre el dispositivo y la aplicación de escritorio se hará a través de él al igual que se haría con un puerto serie físico, esto supone una gran ventaja a la hora de diseñar la aplicación de escritorio, ya que cualquier IDE de programación sea del lenguaje que sea, dispone de un componente o librería que permite el acceso al puerto serie fácilmente.

d) Existe una clase genérica llamada "**Custom Class**" que se utiliza cuando el dispositivo no se asemeja a las características de ninguno de los miembros pertenecientes a otras clases. Un ejemplo de dispositivo que utiliza esta clase es el ICD2 o ICD3 de Microchip.

Si el firmware de un dispositivo no cumple con las especificaciones de alguna de las clases que se beneficia del uso de drivers genéricos instalados ya en el sistema operativo no queda otra que diseñar un driver para el dispositivo, sino el dispositivo no será reconocido por la PC, diseñar un driver desde cero es muy complicado ya que se requiere profundos conocimientos tanto de la arquitectura del PC como del Sistema Operativo que tenga instalado. Afortunadamente ya hay drivers personalizados que se pueden utilizar en proyectos y naturalmente el que proporciona Microchip para la utilización de este driver Microchip proporciona los siguientes archivos:

- **mchpusb.sys** es el driver en si y solo está disponible en formato binario.
- **mchpusb.ini** este archivo contiene información del driver y es requerido por el sistema operativo para que cuando se conecta por primera vez el dispositivo al PC este sepa que driver tiene que asignarle. El Sistema Operativo también

obtiene información de este archivo para configurar correctamente el dispositivo. Es un archivo de texto que se puede editar y modificar algunas cosas en él, como los valores del VID y PID que deben de coincidir con los definidos en los descriptores del dispositivo, también se pueden modificar algunas cadenas de texto que se mostraran como información del dispositivo una vez instalado el driver en el Sistema Operativo.

- **La DLL mpusbapi**: una dll es una capa software que en este caso facilita por medio de funciones la comunicación entre la aplicación de escritorio y el driver del dispositivo. Esta dll está compilada con el compilador de Borland C++ Builder 6. Microchip facilita su código fuente, por lo que hay dos formas de utilizarla. Si utilizamos el mismo IDE de Borland para crear aplicaciones de escritorio, simplemente se tendrá que añadir la librería (el archivo mpusbapi.lib) al proyecto como una librería más, si se utiliza otro compilador diferente se deberá de compilar de nuevo los códigos fuentes de la librería para obtener una nueva versión del archivo mpusbapi.lib que sea compatible con el compilador utilizado. La otra opción de utilización de la librería es añadirla al proyecto de forma dinámica, para ello solo se necesita el archivo con extensión DLL (mpusbapi.dll) proporcionado también por Microchip, de esta forma si se utiliza un compilador diferente como los que nos proporciona Microsoft en su plataforma .NET u otros como LabVIEW, Matlab, RealBAsic, entre otros. No se tendrá que modificar el código fuente de la DLL, simplemente se utilizara importando las funciones públicas de la DLL desde la aplicación de escritorio. Microchip proporciona ejemplos de ambas formas. Utilizar USB da pie a realizar las cosas de diferentes formas. Utilizando un driver personalizado en vez de los genéricos que proporciona Windows se obtiene una comunicación USB más versátil y con mayores prestaciones. Por ejemplo, en cuanto a velocidad se puede alcanzar velocidades de hasta 1MB/s frente a los 64 KB/s de la clase HID y los 80 kBytes/s de la clase CDC, además el driver de Microchip soporta todos los tipos de transferencia (Control, Isócronas, Interrup y Bulk). En su última versión

este driver está soportado por los siguientes sistemas operativos: Windows 2000, Windows Server 2003, Windows XP, Windows XP 64, Windows Vista, Windows Vista 64 y Windows 7.

Cuando se utiliza el driver personalizado de Microchip los datos se transmiten en raw (crudo) y la aplicación de escritorio intercambia información con el PIC directamente a través de los endpoints utilizados por éste, en este caso se usa un protocolo de comunicación que a priori no está definido. Es importante copiar del archivo de extensión INF el valor del campo definido como GUI (Global Unique Identifier), que es un número que sirve para que la aplicación pueda acceder al dispositivo, luego se puede acceder a la dll desde el IDE que se haya elegido para desarrollar la aplicación de escritorio, se editara las propiedades y funciones publicas de la dll buscando un campo que se llame GUI, si no se lo hace no se podrá crear la tubería de conexión (pipe) entre la aplicación y el driver. Cuando se utiliza la clase CDC esto no es necesario ya que la comunicación está definida a través del puerto serie virtual creado y la información se transmite por defecto en forma de caracteres ASCII al igual que un puerto serie físico.

Otros drivers que se puede utilizar para el PIC 18f4550 son:

- **WinUSB**: es el driver genérico para USB que propone utilizar Microsof cuando el dispositivo no puede ser manejado directamente por los drivers que ya incluye por defecto en su sistema operativo. Para ello al igual que Microchip proporciona los siguientes archivos: Winusb.sys, Winusb.ini y la librería dinámica Winusb.dll que al igual que la dll de Microchip proporciona una serie de funciones públicas para acceder fácilmente al driver y a las APIs win32 de Windows desde la aplicación de escritorio.

- **libUSB**: Es un driver de código abierto con licencia GNU, se puede instalar en múltiples sistemas operativos como Linux, MAC y otros, incluyendo Windows a través de su versión libusb-win32

2.5. SENSORES DE TEMPERATURA.

Una resistencia que es sensible a la temperatura es llamada Termistor. A la resistencia con coeficiente de temperatura negativo es conocido como Termistor **NTC** y con coeficiente de temperatura positivo **PTC**, a este tipo de resistencias la temperatura ambiente les afecta de modo que modifican su valor dentro de unos parámetros. Hay varios tipos de encapsulado.

La resistencia de la mayoría de los tipos comunes de termistores disminuye mientras que se eleva la temperatura, a estos se los llaman de coeficiente negativo de temperatura o termistores NTC. Los semiconductores tienen la característica de ofrecer la mitad de la resistencia entre los conductores y los aislantes, mientras más se eleva la temperatura, más portadores de carga están disponibles y esto causa la caída del valor de la resistencia.

Aunque es menos utilizado, es posible fabricar termistores de temperatura de coeficiente positivo o PTC, éstos se hacen de diversos materiales y muestran un aumento de resistencia que varía con temperatura.

Un circuito apropiado que entregue una tensión alta cuando se detecten las condiciones de temperatura caliente, es el divisor de tensión con un termistor NTC, denotado en el siguiente circuito.

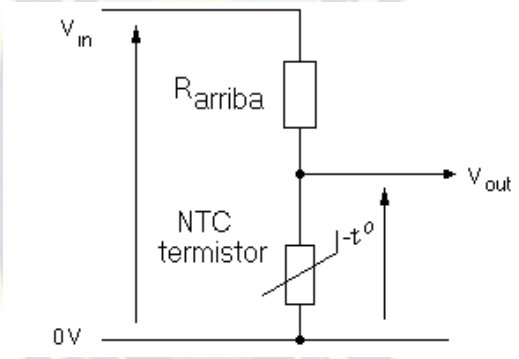


Figura 2.20.: Divisor de Tensión, con NTC.
Fuente: Ramón Pallas Areny

2.5.1. CURVA CARACTERÍSTICA DEL TERMISTOR

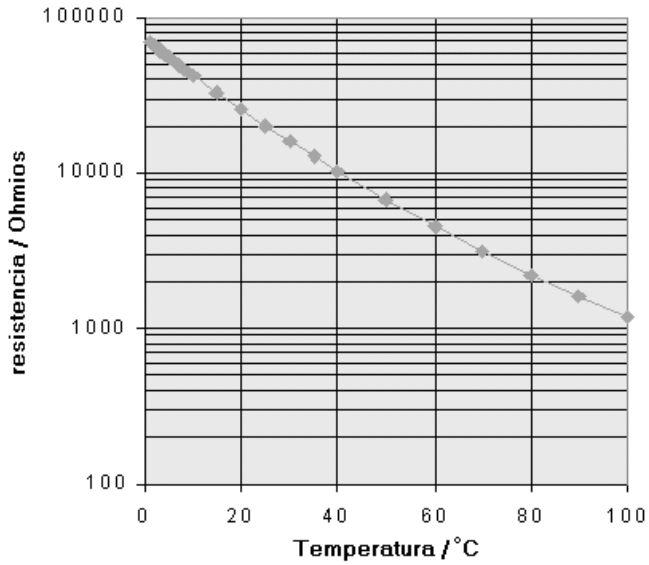


Figura 2.21.: Curva característica de un Termistor.
Fuente: Ramón Pallas Areny

En el eje Y, se representa la resistencia con una escala logarítmica. Ésta es una manera de comprimir el gráfico de modo que sea más fácil ver cómo cambia la resistencia. Entre 100 Ω y 1000 Ω, cada división horizontal corresponde a 100 Ω. Por otra parte, entre 1000 Ω y 10000 Ω, cada división corresponde a 1000 Ω. Y sobre 10000 Ω, representa 10000 Ω cada división.

Como se puede apreciar, este Termistor tiene una resistencia que varía de alrededor 70 kΩ en 0°C a cerca de 1 kΩ a 100°C. Los catálogos de los suministradores, dan generalmente la resistencia a 25°C, que en este caso será 20 kΩ, generalmente, los catálogos también especifican un valor “beta”, cuando se especifican estos dos números, es posible calcular un valor aproximado para la resistencia del Termistor en cualquier temperatura, mediante la ecuación particular:

$$R_T = R_{T0} * e^{B*\left(\frac{1}{T}-\frac{1}{T0}\right)}$$

Figura 2.22.: Ecuación particular del Termistor.
Fuente: Ramón Pallas Areny

Donde:

- $\boldsymbol{R_T}$ es la resistencia a temperatura T en grados **Kelvin** (ºk = ºC + 273)
- $\boldsymbol{R_{T0}}$ es la resistencia de referencia a una temperatura T_o en Kelvin. Cuando la temperatura de la referencia es 25 ºC, T_0 = 25+273.
- $\boldsymbol{e}$ es la base del logaritmo natural.
- $\boldsymbol{B}$ es el 'B-valor' especificado para este termistor.

Es posible generar las curvas características para cualquier termistor, calculando los valores de la resistencia para una gama de temperaturas dadas. Por ejemplo:

Con R_{To} = 20 kΩ y B = 4200, saltos de resistencia a partir de 0 a 10°C se obtiene la siguiente curva:

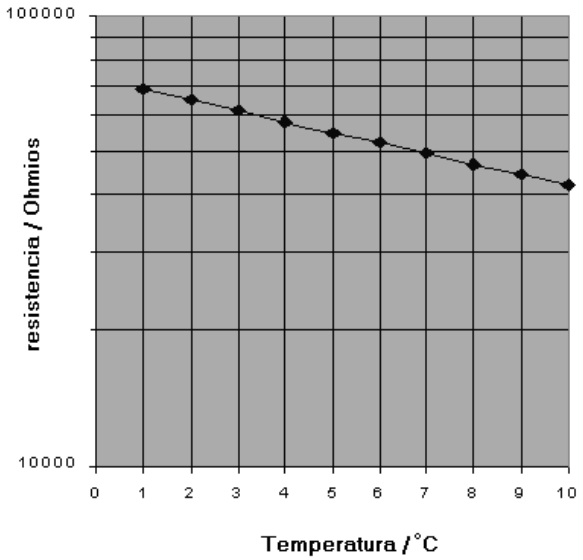


Figura 2.23.: Curva Característica del termistor ejemplo.
Fuente: Ramón Pallas Areny

Según el gráfico, la resistencia para 4°C, se puede estimar poco menos de 60 kΩ. Mediante la ecuación se obtiene 58.2 kΩ. Con estos datos se elige el que da un valor para la resistencia de arriba, cerca de 58.2 kΩ.

Los dispositivos sensores varían considerablemente su resistencia, se puede aplicar esta regla para cerciorarse de que los divisores de tensión que se construya serán siempre tan sensibles como sea posible en el punto crítico.

Los termistores se utilizan extensivamente en coches, por ejemplo en:

- Inyección electrónica de combustible, en la cual la entrada de aire, la mezcla aire/combustible y las temperaturas del agua que le enfría, se supervisan

para ayudar a determinar la concentración del combustible para la inyección óptima.

- Controles de temperatura del aire acondicionado y de asientos en vehículos.
- Los indicadores de alertas, tales como temperaturas de aceite y de líquido, nivel de aceite y turbo - cargador.
- Control del motor de ventilador, basado en la temperatura del agua que se enfría.
- Sensores de escarcha, para la medida de la temperatura exterior.
- Sistemas acústicos.
- Control en la medición de flujo de aire, por ejemplo en la supervisión de la respiración en bebés prematuros, entre otras aplicaciones.

2.5.2. SENSOR DE TEMPERATURA LM35

La medición de magnitudes mecánicas, térmicas, eléctricas y químicas se realiza empleando dispositivos denominados sensores y transductores. El sensor es sensible a los cambios de la magnitud a medir, como una temperatura, una presión, o una concentración química. El transductor convierte estas mediciones en señales eléctricas, que pueden alimentar a instrumentos de lectura, registro o control de las magnitudes medidas. Los sensores y transductores pueden funcionar en ubicaciones alejadas del observador, así como en entornos inadecuados o impracticables para los seres humanos.

Algunos dispositivos actúan de forma simultánea como sensor y transductor. Por ejemplo un termopar consta de dos uniones de diferentes metales que generan una pequeña tensión que depende del diferencial térmico entre las uniones. El termistor es un reóstato especial, cuya resistencia varía según la temperatura. Para medir distancias se emplean condensadores de diseño especial, y para detectar la luz se utilizan fotocélulas. Para medir velocidades, aceleraciones o flujos de líquidos se recurre a otro tipo de dispositivos. En la mayoría de los casos, la señal eléctrica es débil y debe ser amplificada por un circuito electrónico.

En general, los transductores transforman un tipo de variable física (fuerza, presión, temperatura, velocidad, caudal, entre otros) en otro (voltaje, corriente, resistencia, frecuencia, pulsos). El transductor necesita estar calibrado para ser útil como dispositivo de medida. La calibración es el procedimiento mediante el cual se establece la relación o función de transferencia entre la variable medida y la señal de salida convertida. Los transductores pueden clasificarse en dos tipos básicos dependiendo de la forma de la señal convertida.

El LM35 es un sensor de temperatura de precisión. Esta calibrado para proveer una salida de 10mVolts por grado de temperatura. Puede medir temperaturas entre los -50 y los 128ºC.

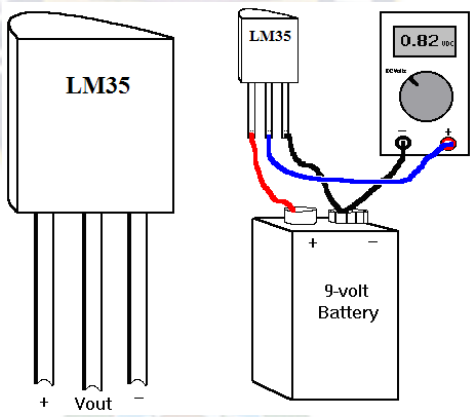


Figura 2.24.: Sensor de temperatura LM35.
Fuente: Ramón Pallas Areny

Es necesario recalcar que los valores entregados por el sensor están en grados Fahrenheit, por lo cual es necesario utilizar la siguiente fórmula de conversión:

$$F = 9/5 * ºC + 32$$

Figura 2.25.: Conversión de grados Centígrados a grados Fahrenheit.
Fuente: Ramón Pallas Areny

2.6. MÓDULO LCD

Esta representado por 2 líneas de 16 caracteres cada una, a través de 8 líneas de datos se le envía el carácter ASCII que se desea visualizar así como ciertos códigos de control que permiten realizar diferentes efectos de visualización. Igualmente mediante estas líneas de datos el módulo devuelve información de su estado interno. Con otras tres señales adicionales se controla el flujo de información entre el módulo LCD y el Microcontrolador que lo gestiona. A continuación se presenta la descripción de señales empleadas por el módulo LCD así como el número de patilla a la que corresponden.

PIN	SIMBOLO	CONEXIÓN	DESCRIPCIÓN
1	V_{SS}	V_{SS}	Patilla de tierra de alimentación
2	V_{dd}	V_{dd}	Patilla de alimentación de +5Vcc
3	V_o	V_o	Patilla de contraste de cristal líquido. Normalmente se conecta aun potenciometro a través del cual se aplica una tensión variable entre 0 + 5Vcc que permite regular el contraste del cristal líquido
4	RS	RA0	Selección del registro de control / registro de datos: **RS – 0** Selección del registro de control **RS – 1** Selección del registro de datos
5	R/W	RA1	Señal de lectura / escritura **R/W – 0** El Módulo LCD es escrito **R/W – 0** El Módulo LCD es leído
6	E	RA2	Señal de activación del módulo LCD **E – 0** Módulo desconectado **E – 1** Módulo conectado
7 al 14	D0 – D7	RB0 – RB7	Bus de datos ni – direccional a través de estas líneas se realiza la transferencia de información entre el módulo LCD y el sistema informático que lo gestiona.

Cuadro 2.8. Descripción del Módulo LCD.
Fuente: www.teleline.terra.es/personal/fremiro

PATILLA	NOMBRE	NIVEL	DESCRIPCIÓN

1	V_{SS}	0 V	Tierra
2	V_{CC}	5 V	Alimentación
3	V_{ee}	Nota	Polarización del Cristal
4	RS	Lógico	Selección: 1=Datos; 0=Instrucción
5	R/W	Lógico	Read/Write: 1=Lectura; 0=Escritura
6	E	Lógico	Habilitación
7 – 14	DB0 – DB7	Lógico	Bus de Datos (DBO, LSB)

Cuadro 2.9.: Patillas del Módulo LCD.
Fuente: www.teleline.terra.es/personal/fremiro

2.6.1. JUEGO DE INSTRUCCIONES:

Los comandos o instrucciones, permiten configurar diferentes opciones de trabajo del módulo LCD y conseguir con ello distintos efectos de visualización. El juego de instrucciones consiste en diferentes códigos que se introducen a través del Bus de datos del módulo LCD conectado al Microcontrolador.

a) **CLEAR DISPLAY**

Borra el módulo LCD y coloca el cursor en la primera posición (dirección 0). Pone el bit **I/D** a "1" por defecto.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Tiempo de ejecución:1.64 mS

Figura 2.26.: Configuración Borrar LCD.
Fuente: www.teleline.terra.es/personal/fremiro

b) **HOME**

Coloca el cursor en la posición de inicio (dirección 0) y hace que el display comience a desplazarse desde la posición original. El contenido de la memoria RAM

de datos de visualización (DD RAM) permanece invariable. La dirección de la memoria RAM de datos para la visualización (DD RAM) es puesta a 0.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	X

Tiempo de ejecución:1.64 mS

Figura 2.27.: Configuración Posición de inicio.
Fuente: www.teleline.terra.es/personal/fremiro

c) **ENTRY MODE SET**

Establece la dirección de movimiento del cursor y especifica si la visualización se va desplazando a la siguiente posición de la pantalla o no. Estas operaciones se ejecutan durante la lectura o escritura de la DD RAM o CG RAM. Para visualizar normalmente poner el bit **S** a "0".

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

Tiempo de ejecución:40 µS

Figura 2.28.: Configuración establece dirección del cursor del LCD.
Fuente: www.teleline.terra.es/personal/fremiro

d) **DISPLAY ON/OFF CONTROL**

Activa o desactiva poniendo en ON/OFF tanto al display (D) como al cursor (C) y se establece si este último debe o no parpadear (B).

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

Tiempo de ejecución:40 µS

Figura 2.29.: Configuración Activa/Desactiva LCD.
Fuente: www.teleline.terra.es/personal/fremiro

e) **CURSOR OR DISPLAY SHIFT**

Mueve el cursor y desplaza el display sin cambiar el contenido de la memoria de datos de visualización DD RAM.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	X	X

Tiempo de ejecución:40 µS

Figura 2.30.: Configuración desplaza cursor.
Fuente: www.teleline.terra.es/personal/fremiro

f) **FUNCTION SET**

Establece el tamaño de interfase con el bus de datos (DL), número de líneas del display (N) y tipo de carácter (F).

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	X	X

Tiempo de ejecución:40 µS

Figura 2.31.: Configuración tamaño de interfase con el bus de datos.
Fuente: www.teleline.terra.es/personal/fremiro

g) **SET THE CGRAM ADDRESS**

El módulo LCD además de tener definidos todo el conjunto de caracteres ASCII, permite al usuario definir 4 u 8 caracteres gráficos. La composición de estos caracteres se va guardando en una memoria llamada CG RAM con capacidad para 64 bytes. Cada carácter gráfico definido por el usuario se compone de 16 u 6 bytes que se almacenan en sucesivas posiciones de la CG RAM. Mediante esta instrucción se establece la dirección de la memoria CG RAM a partir de la cual se irán almacenando los bytes que definen un carácter gráfico. Ejecutado este comando todos los datos que se escriban o se lean posteriormente, lo hacen desde esta memoria CG RAM.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	Dirección de la CG RAM					

Tiempo de ejecución:40 µs

Figura 2.32.: Configuración establece dirección de memoria CG RAM.
Fuente: www.teleline.terra.es/personal/fremiro

h) **SET THE DD RAM ADDRESS**

Los caracteres o datos que se van visualizando, se van almacenando previamente en una memoria llamada DD RAM para de aquí pasar a la pantalla. Mediante esta instrucción se establece la dirección de memoria DD RAM a partir de la cual se irán almacenando los datos a visualizar. Ejecutado este comando, todos los datos que se escriban o lean posteriormente los hacen desde esta memoria DD RAM. Las direcciones de la 80h a la 8Fh corresponden con los 16 caracteres del primer renglón y de la C0h a la CFh con los 16 caracteres del segundo renglón, para este modelo.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	Dirección de la RAM						

Tiempo de ejecución:40 μs

Figura 2.33.: Configuración establece dirección de memoria DD RAM.
Fuente: www.teleline.terra.es/personal/fremiro

i) **READ BUSY FLAG & ADDRESS**

Cuando el módulo LCD está ejecutando cualquiera de estas instrucciones, tarda un cierto tiempo de ejecución en el que no se le debe mandar ninguna otra instrucción. Para ello dispone de un flag llamado BUSY (BF) que indica que se está ejecutando una instrucción previa. Esta instrucción de lectura informa del estado de dicho flag además de proporcionar el valor del contador de direcciones de la CG RAM o de la DD RAM según la última que se haya empleado.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	Dirección de la CG RAM o de la DDRAM						

Tiempo de ejecución:1 μs

Figura 2.34.: Configuración informa estado de BUSY.
Fuente: www.teleline.terra.es/personal/fremiro

j) **WRITE DATA TO CG OR DDRAM**

Mediante este comando se escribe en la memoria DD RAM los datos que se quieren presentar en pantalla y que serán los diferentes códigos ASCII de los caracteres a visualizar. Igualmente se escribe en la memoria CG RAM los diferentes bytes que permiten confeccionar caracteres gráficos a gusto del usuario. El escribir en uno u

otro tipo de memoria depende de si se ha empleado previamente la instrucción de direccionamiento DD RAM o la de direccionamiento CG RAM.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	Código ASCII o byte del carácter gráfico							

Tiempo de ejecución:40 μs

Figura 2.35.: Configuración escribe en DD RAM.
Fuente: www.teleline.terra.es/personal/fremiro

k) **READ DATA FRW CG OR DD RAM**

Mediante este comando se lee de la memoria DD RAM los datos que haya almacenado y que serán los códigos ASCII de los caracteres visualizados. Igualmente se lee de la memoria CG RAM los diferentes bytes con los que se ha confeccionado un determinado carácter gráfico. El leer de uno u otro tipo de memoria depende de si se ha empleado previamente la instrucción de direccionamiento de la DD RAM o la de direccionamiento CG RAM.

Código:

RA0	RA1	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	Código ASCII o byte del carácter gráfico							

Tiempo de ejecución:40 μs

Figura 2.36.: Configuración lee en DD RAM.
Fuente: www.teleline.terra.es/personal/fremiro

2.6.2. JUEGO DE CARACTERES

Las posiciones marcadas como CG RAM (n) corresponden a uno de los 8 posibles caracteres gráficos definidos por el usuario.

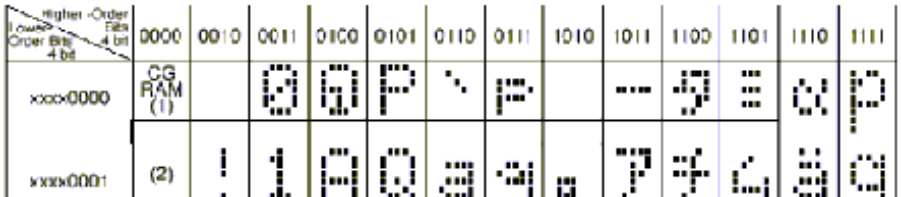


Figura 2.37.: Juego de Caracteres del módulo LCD.
Fuente: www.teleline.terra.es/personal/fremiro

2.7. PROGRAMACIÓN DE LA INTERFASE

2.7.1. LENGUAJE DE PROGRAMACIÓN VISUAL BASIC 6.0

Visual Basic 6.0 es uno de los lenguajes de programación que más entusiasmo despiertan entre los programadores de Computadoras Personales, tanto expertos como novatos. En el caso de los programadores expertos por la facilidad con la que desarrollan aplicaciones complejas en poco tiempo (comparado con lo que cuesta programar en Visual C++, por ejemplo). En el caso de los programadores novatos por el hecho de ver de lo que son capaces a los pocos minutos de empezar su aprendizaje. El precio que hay que pagar por utilizar Visual Basic 6.0 es una menor velocidad o eficiencia en las aplicaciones.

Visual Basic 6.0 es un lenguaje de programación visual, también llamado lenguaje de Cuarta Generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla. Visual Basic 6.0 es también un programa basado en objetos, aunque no orientado a objetos como C++ o Java. La diferencia está en que Visual Basic 6.0 utiliza objetos con propiedades y métodos, pero carece de los mecanismos de herencia y polimorfismo propios de los verdaderos lenguajes orientados a objetos como Java y C++.

Visual Basic 6.0 está orientado a la realización de programas para Windows, pudiendo incorporar todos los elementos de este entorno informático: ventanas, botones, cajas de diálogo y de texto, botones de opción y de selección, barras de desplazamiento, gráficos, menús, etc. Prácticamente todos los elementos de interacción con el usuario de los que dispone Windows 95/98/NT pueden ser programados en Visual Basic 6.0 de un modo muy sencillo.

En ocasiones bastan unas pocas operaciones con el ratón y la introducción a través del teclado de algunas sentencias para disponer de aplicaciones con todas las características de Windows 95/98/NT. En los siguientes apartados se introducirán algunos conceptos de este tipo de programación.

2.7.1.1. FORMULARIOS Y CONTROLES

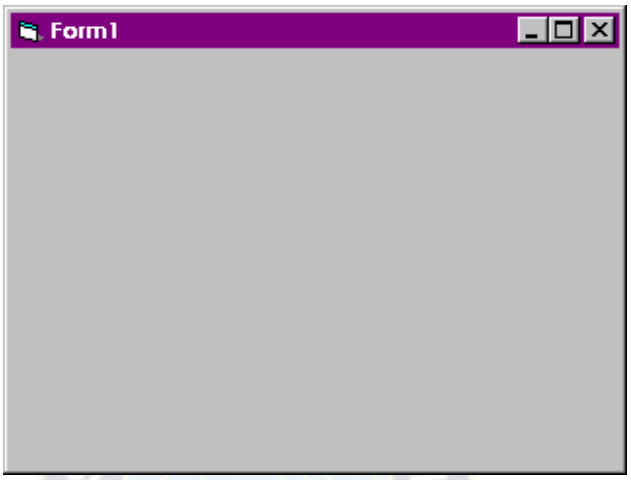


Figura 2.38.: Formulario de Visual Basic.

Fuente: Manual de Referencia Visual Basic 6.0

Cada uno de los elementos gráficos que pueden formar parte de una aplicación típica de Windows 95/98/NT es un tipo de control: los botones, las cajas de diálogo y de texto, las cajas de selección desplegables, los botones de opción y de selección, las barras de desplazamiento horizontales y verticales, los gráficos, los menús, y muchos otros tipos de elementos son controles para Visual Basic 6.0. Cada control debe tener un nombre a través del cual se puede hacer referencia a él en el programa. Visual Basic 6.0 proporciona nombres por defecto que el usuario puede modificar.

En la terminología de Visual Basic 6.0 se llama formulario (form) a una ventana. Un formulario puede ser considerado como una especie de contenedor para los controles. Una aplicación puede tener varios formularios, pero un único formulario puede ser suficiente para las aplicaciones más sencillas. Los formularios deben también tener un nombre, que puede crearse siguiendo las mismas reglas que para los controles.

2.7.1.2. OBJETOS Y PROPIEDADES

Los formularios y los distintos tipos de controles son entidades genéricas de las que puede haber varios ejemplares concretos en cada programa. En programación orientada a objetos (más bien basada en objetos, habría que decir) se llama clase a estas entidades genéricas, mientras que se llama objeto a cada ejemplar de una clase determinada. Por ejemplo, en un programa puede haber varios botones, cada uno de los cuales es un objeto del tipo de control command button, que sería la clase.

Cada formulario y cada tipo de control tienen un conjunto de propiedades que definen su aspecto gráfico (tamaño, color, posición en la ventana, tipo y tamaño de letra, etc.) y su forma de responder a las acciones del usuario (si está activo o no, por ejemplo). Cada propiedad tiene un nombre que viene ya definido por el lenguaje.

Por lo general, las propiedades de un objeto son datos que tienen valores lógicos (True, False) o numéricos concretos, propios de ese objeto y distintos de las de otros objetos de su clase. Así pues, cada clase, tipo de objeto o control tiene su conjunto de propiedades, y cada objeto o control concreto tiene unos valores determinados para las propiedades de su clase.

Casi todas las propiedades de los objetos pueden establecerse en tiempo de diseño y también - casi siempre - en tiempo de ejecución. En este segundo caso se accede a sus valores por medio de las sentencias del programa, en forma análoga a como se accede a cualquier variable en un lenguaje de programación. Para ciertas propiedades ésta es la única forma de acceder a ellas. Por supuesto Visual Basic 6.0 permite crear distintos tipos de variables.

2.7.1.3. EL ENTORNO DE PROGRAMACIÓN VISUAL BASIC 6.0

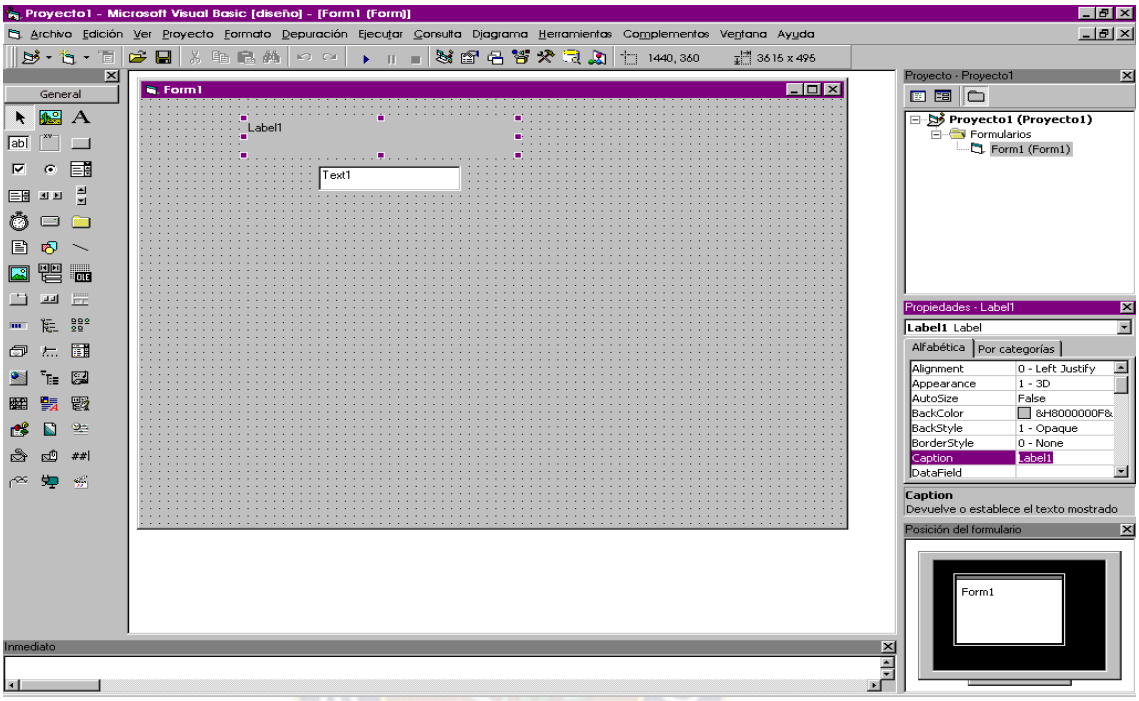


Figura 2.39.: Entorno de Programación.
Fuente: Manual de Referencia Visual Basic 6.0

Cuando se inicia Visual Basic 6.0 aparece en la pantalla una configuración similar a la mostrada en la Figura. En ella se pueden distinguir los siguientes elementos:

1. La barra de títulos, la barra de menús y la barra de herramientas de Visual Basic 6.0 en modo Diseño (parte superior de la pantalla).
2. Caja de herramientas (toolbox) con los controles disponibles (a la izquierda de la ventana).
3. Formulario (form) en gris, en que se pueden ir situando los controles (en el centro). Está dotado de una rejilla (grid) para facilitar la alineación de los controles.
4. Ventana de proyecto, que muestra los formularios y otros módulos de programas que forman parte de la aplicación (arriba a la derecha).
5. Ventana de Propiedades, en la que se pueden ver las propiedades del objeto seleccionado o del propio formulario (en el centro a la derecha). Si esta ventana no aparece, se puede hacer visible con la tecla <F4>.

6. Ventana FormLayout, que permite determinar la forma en que se abrirá la aplicación cuando comience a ejecutarse (abajo a la derecha).

Existen otras ventanas para edición de código (Code Editor) y para ver variables en tiempo de ejecución con el depurador o Debugger (ventanas Immediate, Locals y Watch). Todo este conjunto de herramientas y de ventanas es lo que se llama un entorno integrado de desarrollo o IDE (Integrated Development Environment).

Construir aplicaciones con Visual Basic 6.0 es muy sencillo: basta crear los controles en el formulario con ayuda de la toolbox y del ratón, establecer sus propiedades con ayuda de la ventana de propiedades y programar el código que realice las acciones adecuadas en respuesta a los eventos o acciones que realice el usuario.

CAPÍTULO III

CONSTRUCCIÓN DEL PROTOTIPO

CAPÍTULO III

CONSTRUCCIÓN DEL PROTOTIPO

3.1. ANÁLISIS Y DISEÑO.

3.1.1. INTRODUCCIÓN.

Realizar el análisis es una de las tareas más significativas por ser éste el puntal de construcción del prototipo, se tendrá en cuenta la ordenación y tabulación de la información que se obtendrá mediante técnicas, instrumentos y/o métodos descritos en el anterior capítulo, permitiendo de esta manera realizar criterios sistemáticos y sacar conclusiones dentro de un enfoque orientado a la construcción del Sistema de Control.

El análisis debe estar basado de manera funcional con respecto a los procesos de la pasteurización en el cual se podrá extraer los elementos que actuarán o influirán dentro de la herramienta tomando en cuenta que este conjunto de disposiciones de procedimientos o programas relacionados forman una sola unidad. El conjunto de hechos, principios y reglas clasificadas, dispuestas de manera ordenada mostrará un plan lógico en la unión de las partes enfocadas. Todo esto se lleva a cabo teniendo en cuenta ciertos principios:

- Presentar y entender el dominio de la información de cada problema descritos en el Capítulo I.
- Definir las funciones que debe realizar el Sistema.
- Representar el comportamiento del Sistema a consecuencias de acontecimientos externos.
- Dividir en forma jerárquica los modelos que representan la información, funciones y comportamiento.

El proceso debe partir desde la información esencial hasta el detalle de la Implementación del Sistema. Dentro la función del análisis, el punto preponderante es el de dar un soporte de control al proceso de pasteurización de la lecha; para conseguir este objetivo, el Sistema hace uso de seis elementos fundamentales:

1. **SOFTWARE**: que son programas, con estructuras de datos, conjunto de instrucciones y su documentación que hacen efectiva la logística metodológica o controles de requerimientos del programa.

2. **HARDWARE**: dispositivos electrónicos y electromecánicos, que proporcionan capacidad de cálculos y funciones rápidas, exactas y efectivas, que proporcionan una función externa dentro del Sistema.

3. **PERSONAL:** son los operadores o usuarios directos del Sistema.

4. **ARCHIVOS DE DATOS:** una gran colección de informaciones organizadas y enlazadas al Sistema a las que se accede por medio del Software, el cual contará con un formato (extensión) propia del Sistema.

5. **DOCUMENTACIÓN**; manuales, formularios, y otra información descriptiva que detalla o da instrucciones sobre el empleo y operación del Sistema.

6. **PROCEDIMIENTOS**: que definen el uso especifico de cada uno de los elementos o componentes del Sistema y las reglas de su manejo y mantenimiento.

El Análisis se lleva a cabo teniendo en cuenta los siguientes objetivos en mente:

a) Identificar las necesidades en el proceso de Pasteurización de la lecha.
b) Evaluar los conocimientos previos para establecer la viabilidad del proyecto.
c) Realizar un Análisis Técnico exhaustivo.
d) Asignar funciones al Hardware, Software, y otros elementos del sistema.
e) Crear una definición de la herramienta que forme el fundamento de todo el trabajo de Ingeniería.

3.2. ANÁLISIS DE NECESIDADES Y ESTUDIO DE VIABILIDAD.

3.2.1. INICIO DEL PROYECTO

Las necesidades hicieron que se enfoque en un problema general o principal, como también en problemas específicos cuyos contenidos están descritos en el Capitulo I. Por tal razón, la funcionalidad del sistema está expresamente dirigida al control del procesos de pasteurización de la leche tomando desde luego procedimientos o métodos ya

mencionados, por eso es necesario llevar a cabo estudios y análisis adecuados a actividades que están sujetos a dichos procesos.

3.2.2. ESTUDIO DE VIABILIDAD.

3.2.2.1. VIABILIDAD TÉCNICA.

La aplicación de estándares permite la viabilidad técnica del proyecto porque hace uso de especificaciones de Hardware y Software dispuestos al proceso de control. Por lo tanto, la base que constituye el proceso de control en el proyecto, determina la continuidad de éste, sin correr el riesgo de que no funcione o no tenga el rendimiento deseado.

3.2.2.2. VIABILIDAD OPERATIVA.

El uso de normas y reglas dentro del proceso de control, hace viable la interacción del sistema con el usuario, tomando en cuenta que solamente se tendrá la posibilidad de estar sujeto al control del proceso de pasteurización descritos en los anteriores capítulos. Por tal motivo los resultados que se obtienen son visibles gracias a la interacción de tecnologías a través de sus métodos y herramientas.

3.3. TÉCNICAS DE RECOGIDA DE INFORMACIÓN.

La mejor manera para recolectar información debida y apropiada en los procedimientos estructurales, es sin duda la utilización de la técnica de la entrevista, el cuestionario y la observación directa. El enfoque principal que se está adoptando en la construcción del Sistema es el uso del método del prototipo (Capítulo II – Prototipos).

3.4. ESPECIFICACIONES DE LOS REQUISITOS.

3.4.1. REQUISITOS DEL USUARIO.

El sistema permite el control directo en el proceso de pasteurización, motivo por el cual, es necesario contar con un nivel de conocimiento en dicho proceso. Según estos aspectos, el usuario puede escoger entre parámetros ya establecidos en el sistema y, en consecuencia, obtendrá procesamientos automáticos en el proceso de pasteurización. A su vez, es necesario que la interfase con el usuario sea sencilla y amigable.

3.4.2. ESPECIFICACIÓN DEL REQUERIMIENTO DEL SISTEMA

Es necesario hacer notar que el Sistema será diseñado según el proceso de pasteurización lenta o discontinua por el hecho de establecer procesos automáticos y adecuarlos a empresas cuyo proceso de pasteurización lo realizan en pequeños volúmenes de leche, además que hace posible el funcionamiento del prototipo para su demostración en especial en la parte de control. Establecido el ámbito funcional del Sistema cabe hacer notar que los parámetros definidos en éste tipo de pasteurización se dan a través de la relación Temperatura/Tiempo, motivo por el cual se define como la unidad de medida de la Temperatura al Grado Centígrado, y como unidad de Tiempo al Segundo.

3.4.3. DIAGRAMA DE BLOQUES DEL SISTEMA.

Descrita la especificación del sistema, en la figura siguiente se muestra las funciones que expresan las tareas y/o actividades necesarias que permitan el control del proceso de pasteurización de la leche.

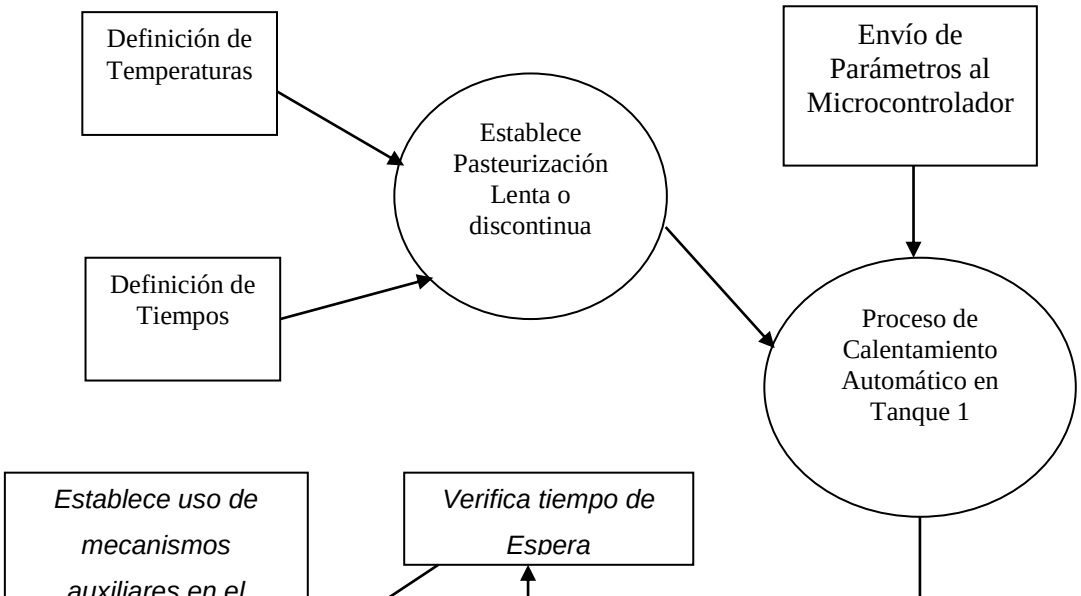
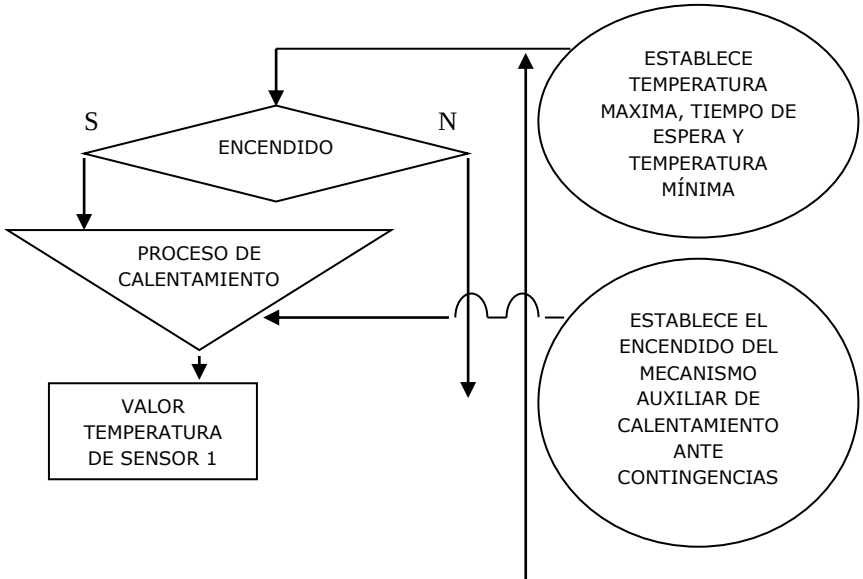
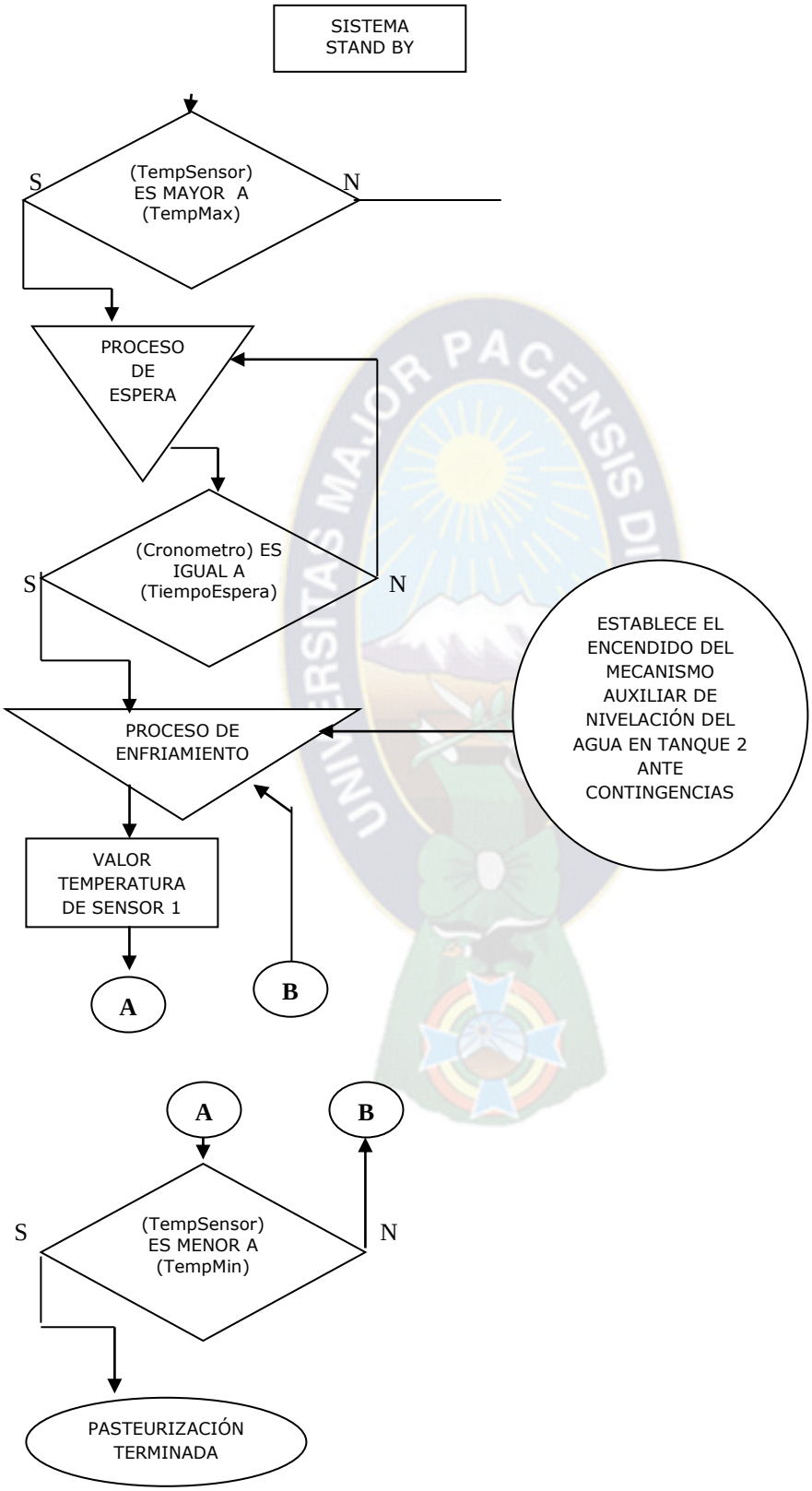


Figura 3.1.: Diagrama de Bloques.
Fuente: Elaboración Propia

3.4.4. DIAGRAMA DE PROCESOS

Es necesario realizar un análisis de los procesos que están inmersos en el proceso de pasteurización de la leche, tomando en cuenta que el proceso es de tipo lento o Discontinuo.

PROCESO DE PASTEURIZACIÓN	USUARIO	DESCRIPCIÓN
		El Usuario establece los parámetros: Temperatura Máxima (TempMax), Tiempo de Espera (TiempoEspera) y la Temperatura Mínima (TempMin). Al ser iniciado el sistema, éste ingresara al proceso de Calentamiento hasta alcanzar la temperatura Máxima establecida por el Usuario, en dicho proceso se puede recurrir al uso del



Una vez alcanzado la temperatura mínima el sistema desactiva todo mecanismo e ingresa al estado de "Apagado". Con todos estos procesos se logra pasteurizar la leche en forma discontinua.

Figura 3.2. Diagrama de procesos
Fuente: Elaboración Propia

3.5. ESPECIFICACIÓN DE CONOCIMIENTOS PREVIOS.

3.5.1. MODELO MATEMÁTICO DEL TERMISTOR.

Realizando la conversión análogo digital de los datos del sensor se tiene el siguiente modelo:

$$Temp = \frac{5 * lectura}{1023}$$

Figura 3.3. Cuantificación del Dato abstraído del Sensor.
Fuente: Elaboración Propia

Una vez realizado la cuantificación, se procede a hallar el valor de la resistencia del Sensor, a través del siguiente modelo:

$$R_T = \frac{10000 * Temp}{5 - Temp}$$

Figura 3.4. Ecuación para hallar el valor resistivo del Sensor.
Fuente: Elaboración Propia

En la figura 2.20, se muestran el modelo matemático del valor de la resistencia del termistor, por lo cual se hace el siguiente cálculo para el sensor que en el proyecto se está utilizando:

$$T = \frac{T_0 * B}{T_0 * \ln\left(\frac{R_T}{R_{T0}}\right) + B} - 273$$

Figura 3.5. Ecuación de la Temperatura en Grados Centígrados.
Fuente: Elaboración Propia

Donde:

- B = 4425 (Constante. Fuente: Catálogo)
- Rto = 480(Resistencia de Referencia. Fuente: Catálogo)
- To = 341 (Temperatura de Referencia. Fuente: Catálogo)

3.5.2. MODELO MATEMÁTICO DEL LM35.

Este sensor según catálogo determina através de un proceso y cuantificación del valor de lectura que la temperatura hallada en Grados Centígrados se basa en el siguiente modelo matemático:

$$T_{LM\,35} = \frac{4.888 * lectura}{10}$$

Figura 3.6. Ecuación de la Temperatura del LM35.
Fuente: Elaboración Propia

3.6. SELECCIÓN DE ESTRATEGIAS INSTRUCCIONALES.

Los diferentes contenidos son parte de los diferentes módulos con los que cuenta el Sistema y el cual está sujeto a verificación y prueba.

3.6.1. ORGANIZACIÓN DE CONTENIDOS.

Se tomó en cuenta los siguientes contenidos para que el usuario pueda interactuar con el sistema:

3.6.1.1. DEFINICIÓN DE PARÁMETROS.

Para lo cual es necesario ir definiendo todos los parámetros necesarios que ésta necesita como ser:

- **Número de Proceso:** Indica el número de proceso correspondiente a la Pasteurización, ya que éste identificara y diferenciará los procesos en el cual sistema irá controlando cada proceso de pasteurización.

- **Temperatura Máxima:** Indica el valor de la temperatura máxima al que debe llegarse durante el proceso de calentamiento. Éste valor está definido por el tipo de Pasteurización que se está utilizando en el Proyecto.

- **Tiempo de Espera:** Indica el tiempo del proceso de espera, en el cual la flora de microorganismos como parásitos no patógenos y la totalidad de los agentes microbianos patógenos disminuyen.

- **Temperatura Mínima:** Indica el valor de la temperatura minima al que debe llegarse durante el proceso de enfriamiento. Éste valor está definido por el tipo de Pasteurización que se está utilizando en el Proyecto.

3.7. SELECCIÓN DE ESTRATEGIAS DE EVALUACIÓN.

La evaluación permite asegurar la adecuación, factibilidad técnica y completitud de los requerimientos definidos en la Especificación de Requerimientos y verificar que se cumplan los atributos del estándar IEEE 830-1993 para los Requerimientos funcionales y no funcionales.

Los atributos del estándar IEEE 830-1993 para los Requerimientos Funcionales y No Funcionales son:

1) Verificación interna (de acuerdo al alcance) y externa (todos los elementos están definidos).
2) Consistentes (Ausencia absoluta de elementos contradictorios).
3) No Ambiguos (Debe existir un glosario para los términos utilizados).
4) Verificables (Existencia de métodos para verificar si se cumplen o no).
5) Modificables (No deben existir grandes dificultades para cambiarlos, como abuso de referencias cruzadas).
6) Priorizados (Los requerimientos están ordenados por su importancia en el alcance).

Para realizar la evaluación del Sistema se debe tomar en cuenta el tiempo de creación de todo el proyecto, evaluándose lo planificado contra lo realizado, el apego al proceso en todo el transcurso del proyecto y el cumplimiento o no de las propiedades de calidad identificadas al inicio.

3.8. DETERMINACIÓN DE VARIABLES TÉCNICAS.

La aplicación se realiza mediante el uso de:

- Botones de Comandos.
- Cajas de Texto.
- Cuadros de Imágenes.
- Colores.
- Scrooll Bar.
- Control Gráfico de la Temperatura.
- Tablas de Datos
- Cuadros de Diálogos.

3.9. PROTOTIPO DEL SISTEMA.

Es evidente que la elaboración de prototipos es una alternativa con muchas características a comparación con otras metodologías, para esto es necesario optar por una concepción de prototipos de características seleccionadas ya que ésta contendrá algunas de las características del sistema final (Prototipo Terminado), debido a que la evaluación será continua independientemente de la fase en la que se encuentre.

3.9.1. ANÁLISIS DE COSTOS Y BENEFICIOS.

Considerando que al empezar a construir el Sistema es necesario realizar ciertos análisis que permitan el desarrollo de dicha actividad, para eso es imperioso realizar los análisis de costos y beneficios.

3.9.1.1. COSTOS DEL SISTEMA.

Entre los costos se denotan los siguientes puntos:

1. Costos de avituallamiento: Es necesario hacer mención, que son costos que involucran el aprovisionamiento de los medios elementales para el inicio de la construcción del Sistema
   - Coste de Consultoría (No Aplicable)
   - Costo de compra o alquiler del equipo (No Aplicable)
   - Costo de la instalación del equipo (No Aplicable)
   - Costo del acondicionamiento del lugar destinado al equipo (No Aplicable)
   - Coste de capital (No Aplicable)
   - Coste de los gestores y el personal encargados del avituallamiento (No Aplicable)
2. Costos relativos al proyecto:
   - Costo de la compra del Sistema Operativo (Suministrado por ser Proyecto de Grado)
   - Costo de la compra de software de aplicación (Suministrado por ser Proyecto de Grado)
   - Costo de modificaciones del software (Suministrado por ser Proyecto de Grado)
   - Costo del personal encargado del desarrollo del Proyecto (Suministrado por ser Proyecto de Grado)
   - Costo de la formación del personal en el uso del Sistema (Suministrado por ser Proyecto de Grado)
   - Costo de los procedimientos de recolección de datos (Suministrado por ser Proyecto de Grado)

- Costo de la preparación de documentación (Suministrado por ser Proyecto de Grado)
- Costo de la gestión de desarrollo (Suministrado por ser Proyecto de Grado)

3. Costos de puesta a punto:
   - Costo del software del sistema operativo (Suministrado por ser Proyecto de Grado)
   - Costo de hardware (Suministrado por ser Proyecto de Grado)
4. Costos continuos:
   - Costo del mantenimiento del Sistema (Suministrado por ser Proyecto de Grado)
   - Costo de la depreciación del hardware (No aplicable por ser Proyecto de Grado)

3.9.1.2. BENEFICIOS

Es menester hacer mención que el cien por cien de los beneficios que conlleva el proyecto es de tipo intangible ya que beneficia a empresas pequeñas y/o medianas que pasteurizan la leche:

BENEFICIOS	DESCRIPCIÓN
Control de Procesos Efectivos	Gracias a la concentración en un solo módulo de toda la información de los procesos que incurren en la Pasteurización, hace efectivo el seguimiento de dicha tarea, por lo cual todo proceso ira siendo controlada de una forma automática.
Posibilidad de realizar modificaciones al proceso de Pasteurización.	Es factible realizar modificaciones al proceso de pasteurización dependiendo de factores que influyan en dicho proceso. Por lo cual se ofrece al Usuario establecer y definir parámetros en base y fundamentos metodológicos al proceso de Pasteurización.
Posibilidad de Complemento de Mecanismos Auxiliares.	A través de los dispositivos auxiliares, permiten verificar el proceso de pasteurización.

Obtención de la Estadística al proceso de pasteurización	Se lo realiza mediante un seguimiento secuencial entre los módulos del sistema, los cuales van entregando información en un determinado tiempo, para luego ser guardadas en archivos digitales para su posterior estudio.
Posibilidad de analizar y/o revisar los procesos de Pasteurización.	Los medios dispuestos en el Sistema, permitirá la verificación de los procesos anteriores llevados o controlados por el Sistema.
Reducción de la necesidad de trabajo forzado en el control de los procesos.	Debido a que los procesos se tornan automáticos, estos reducen el trabajo del Usuario.
Mejores posibilidades de análisis y revisión de los procesos	Por medio gráfico es factible realizar el análisis y revisión de la pasteurización de la leche.

Cuadro 3.1. Beneficios del desarrollo del Sistema
Fuente: Elaboración Propia

3.9.2. ALCANCES

3.9.2.1. ALCANCES DEL SISTEMA

- Recepción de datos (parámetros).
- Registro de datos automáticos generados por cada sensor.
- Activación de cada proceso de forma automática.
- Control visual emulado por el Sistema.

3.9.2.2. LIMITES DEL SISTEMA

- Se determina el Proceso de Pasteurización según el tipo elegido para el Presente Proyecto (Pasteurización Lenta o Discontinua)
- Acoplamiento de módulos de Control con la de módulos de Fuerza, en base o disposiciones con las que cuenta una empresa ya establecida.
- Acoplamiento de Sensores, dependiendo de la cantidad de Estaciones con las que cuenta la empresa ya establecida.

- Interacción con dispositivos de entrada utilizando únicamente el Mouse y el Teclado.

Es necesario recalcar que el presente proyecto basa su trabajo en el control de la Pasteurización, dejando de lado todo tipo de elementos mecánicos los cuales están presentes en una empresa de pasteurización ya establecida.

3.9.3. CONSTRUCCIÓN DEL SISTEMA.

Una vez definido la funcionalidad del Sistema (ver figura 3.1.), es necesario tomar en cuenta los siguientes aspectos:

3.9.3.1. INTERFASES CON OTROS SOFTWARE.

- Sistema Operativo Windows XP SP3 de Microsoft.
- Lenguaje de Programación Microsoft Visual Basic 6.0 parte del paquete de Microsoft Visual Studio Versión 6.0.
- Simulador PROTEUS 7.6 Profesional.
- Lenguaje de Programación PIC C Compiler, para Microcontroladores PIC.

3.9.3.2. CIRCUITO GENERAL DEL SISTEMA

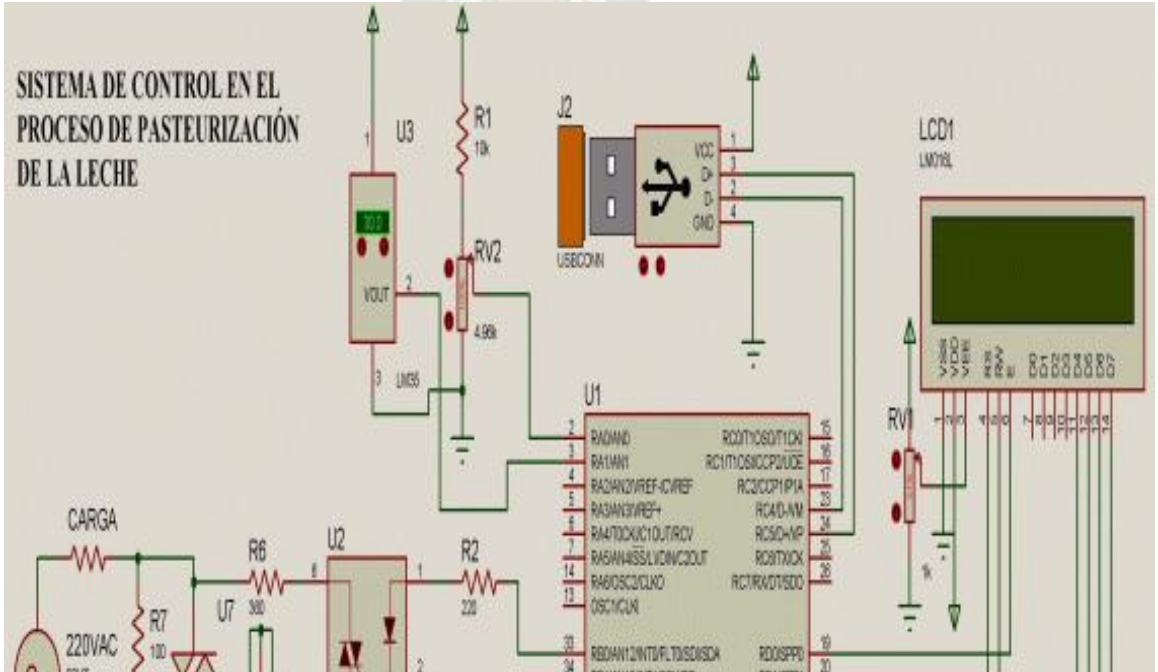


Figura 3.7. Circuito General del Sistema.
Fuente: Elaboración Propia

3.9.3.3. PROGRAMACIÓN DEL PIC18F4550

- **CODIGO FUENTE**

```
#include <18f4550.h>
#device adc =10
#fuses
HSPLL,NOMCLR,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPU
DIV4,VREGEN,PBADEN
#use delay(clock=48000000)
#include <lcd.c>
#include <math.h>
```

```
#DEFINE USB_HID_DEVICE  TRUE
//Habilitar la Transferencia
#define USB_EP1_TX_ENABLE   USB_ENABLE_INTERRUPT    //Activa el punto final 1 para transferencias por bloque
#define USB_EP1_TX_SIZE    8                    //8 bytes  para envio
//Habilitar la Recepcion
#define USB_EP1_RX_ENABLE   USB_ENABLE_INTERRUPT    //Activa el punto final 1 para recepcion por bloque
#define USB_EP1_RX_SIZE    8           //8 bytes para recepcion
#include <pic18_usb.h>          //Funciones de bajo nivel(hardware) para la serie PIC 18Fxx5x que serviran en usb.c
#include "usb_desc_hid.h"     //Aqui es donde van las descripciones de este dispositivo (como lo reconocera windows)
#include <usb.c>            //libreria para el manejo del usb
// definimos variables globales
int8 Tmin; // variable de la Temperatura mínima
int8 Tmax; // variable de la Temperatura máxima
int Band; // variable bandera
int B2; // varialbe bandera
int R1; // variable de la carga 1
int R2; // variable de la carga 2
int R3; // variable de la carga 3
int R4; // variable de la carga 4
int8 rx_msg[8];
char digito[8];
int16 data;
int contador;
int i;
int j;
```

```
int esperando; // variable de espera
int salto;
void usb_rx_task(void)
{
  if (usb_kbhit(true))
  {
    usb_get_packet(1, rx_msg, sizeof(rx_msg));
    delay_ms(50);
    Band=rx_msg[0];
    Tmin=rx_msg[1];
    Tmax=rx_msg[2];
    R1=rx_msg[3];
    R2=rx_msg[4];
    R3=rx_msg[5];
    R4=rx_msg[6];
    esperando=rx_msg[7];
  }
}
void main()
{
  int16 lectura;
  float temp,temp2;
  float Rt,T,T1,B=4425,Rto=480,To=341;
  setup_adc_ports(0x04);
  setup_adc(adc_clock_internal);
  lcd_init();
  usb_init(); //inicializa y espera a ser encendido
  usb_wait_for_enumeration(); //ahora espera hasta ser enumerado (reconocido por la PC)
```

```
lcd_gotoxy(1,1);
printf(lcd_putc, "Sist de Control");
printf(lcd_putc, "\nErwin Zabaleta A");
Band=0;
salto=0;
B2 = 0;
while (true)
{
  if(Band==0)
  {
    usb_rx_task(); //mensaje del panel de control
    output_LOW(pin_b0); //nivel bajo B0
    output_LOW(pin_b1); //nivel bajo B1
    output_LOW(pin_b2); //nivel bajo B2
    output_LOW(pin_b3); //nivel bajo B3
    salto=0;
  }
  else
  {
    usb_rx_task();
    set_adc_channel(0);
    lectura=read_adc();

    temp=(5*lectura)/1023.0;
    Rt=(10000*temp)/(5-temp);
    T=(1/(((1/B)*log(Rt/Rto))+(1/To)))-273;
    printf(lcd_putc,"\fTemp1=%03.1f C",T);
    i = 0;
    while(lectura != 0)
```

```
{
      digito[i] = lectura % 10;
      lectura = lectura / 10;
      i++;
}
digito[i] = 0;
i++;
digito[i] = 0;
i++;
digito[i] = 0;
i++;
digito[i] = 0;
digito[7] ='#';
usb_put_packet(1,digito,8,USB_DTS_TOGGLE);
delay_ms(50);

set_adc_channel(1);
lectura=read_adc();

T1 = (4.888*lectura)/10;
printf(lcd_putc,"\nTemp2=%03.1f C",T1);

i = 0;
while(lectura != 0)
{
      digito[i] = lectura % 10;
      lectura = lectura / 10;
      i++;
}
```

```
digito[i] = 0;
i++;
digito[i] = 0;
i++;
digito[i] = 0;
i++;
digito[i] = 0;
digito[7] ='*';
usb_put_packet(1,digito,8,USB_DTS_TOGGLE);
delay_ms(50);

digito[0] = 1;
digito[7] ='$';
usb_put_packet(1,digito,8,USB_DTS_TOGGLE);

if (B2==0)
{
  if (T<Tmax)
  {
    if(R1==1)
    {
      output_HIGH(pin_b0);
    }
    else
    {
      output_LOW(pin_b0);
    }
    if(R2==1)
    {
```

```
      output_HIGH(pin_b1);
    }
    else
    {
      output_LOW(pin_b1);
    }
  }
  else
  {
    B2 = 1;
    output_LOW(pin_b0);
    output_LOW(pin_b1);
    output_LOW(pin_b2);
    output_LOW(pin_b3);
  }
}
else
{
  if (T>Tmin)
  {
    if (salto==0)
    {
      digito[0] = 3;
      digito[7] ='$';
      usb_put_packet(1,digito,8,USB_DTS_TOGGLE);
            // empezamos el proceso de espera
            for(contador=1;contador<=esperando;contador++)
            {
                  delay_ms(250);
```

```
                delay_ms(250);
                delay_ms(250);
                delay_ms(250);
                printf(lcd_putc,"\fTiempo=%d segundos",contador);
            }
      salto=1;
    }
    digito[0] = 2;
    digito[7] ='$';
    usb_put_packet(1,digito,8,USB_DTS_TOGGLE);
    delay_ms(50);
    if(R3==1)
    {
      output_HIGH(pin_b2);
    }
    else
    {
      output_LOW(pin_b2);
    }
    if(R4==1)
    {
      output_HIGH(pin_b3);
    }
    else
    {
      output_LOW(pin_b3);
    }
  }
  else
```

```
        {
          B2=0;
          Band=0;
          output_LOW(pin_b2);
          output_LOW(pin_b3);
          lcd_gotoxy(1,1);
          printf(lcd_putc, "Sist de Control");
          printf(lcd_putc, "\nErwin Zabaleta A");
          // le ordenamos al panel de control que se apague
          digito[0] = 0;
          digito[7] ='$';
          usb_put_packet(1,digito,8,USB_DTS_TOGGLE);
          delay_ms(50);
        }
      }
      delay_ms(900);
    }
  }
}
```

3.9.3.4. INTERFAZ DE USUARIO.

- Acceder al Sistema de forma fácil y congruente con sus necesidades.
- Aumentar la velocidad de trabajo y reducir los errores.
- Proporcionar la retroalimentación adecuada a procesos establecidos.
- Maximizar el trabajo.

El Sistema tiene el entorno de aplicaciones del sistema operativo Windows, cuya interfase serán los componentes ya descritos en el proceso de variables técnicas.

3.9.3.4.1. DIÁLOGOS

Es necesario presentar al usuario diálogos que permitan:

- La buena comunicación; de tal forma que los datos que introduzca el usuario en el panel de control, sean correctamente editados para su validación.
- La acción del usuario sea mínima; a través de tecleos para la selección de opciones con interacción del Mouse, para que se acelere la captura de datos y minimizar los errores.
- La operación estándar y consistencia; para que el usuario aprenda la forma de uso de las partes del sistema y así familiarizarse con cada uno de sus componentes, a través del uso estándares como por ejemplo:
  - Salir del programa mediante la misma tecla.
  - Cancelar procesos mediante un botón.
  - Estandarizar el color de cada módulo o proceso.
  - Estandarizar el uso de iconos, entre otros.

3.9.3.4.2. ENTRADA EFECTIVA

Es vital la calidad de la entrada porque de ésta dependerá la calidad de la salida, para eso es necesario diseñar pantallas que tengan facilidad de uso, efectividad, consistencia, simplicidad, precisión y atractivo.

- **Ingreso de Datos**: Los dispositivos de entrada que permiten interactuar a la herramienta con el usuario son el Teclado y el Mouse.
- **Salida de Datos**: La salida de datos se lo realiza mediante el monitor y en la parte de Control con un LCD, donde se presenta los datos.

3.9.3.5. INTERFACES EXTERNAS

3.9.3.5.1. INTERFACES DE USUARIO

Se considera las siguientes pantallas en la interfaz de usuario:

- Pantalla Principal del Panel de Control.

- Sub Pantalla de Parámetros
- Sub Pantalla de Control
- Sub Pantalla de Graficador de función
- Sub Pantalla de Control Grafico de Procesos
- Sub Pantalla de Cuadros Estadísticos

3.9.3.5.2. RESTRICCIONES DE DISEÑO

- Estándar de Interfaz de usuario tipo Productos de Microsoft
- Normas Base de La Pasteurización Lenta.
- Estándar de comunicación USB con el Microcontrolador

3.9.3.5.3. ATRIBUTOS DEL PROTOTIPO

- Interfaz amigable
- Facilidad de mantenimiento
- Fácil uso
- Resguardo de información (Trabajo con archivos), ante eventualidades

3.9.3.5.4. DISEÑO DEL SOFTWARE EN VISUAL BASIC 6.0 PARA EL CONTROL EN EL COMPUTADOR

- Pantalla principal

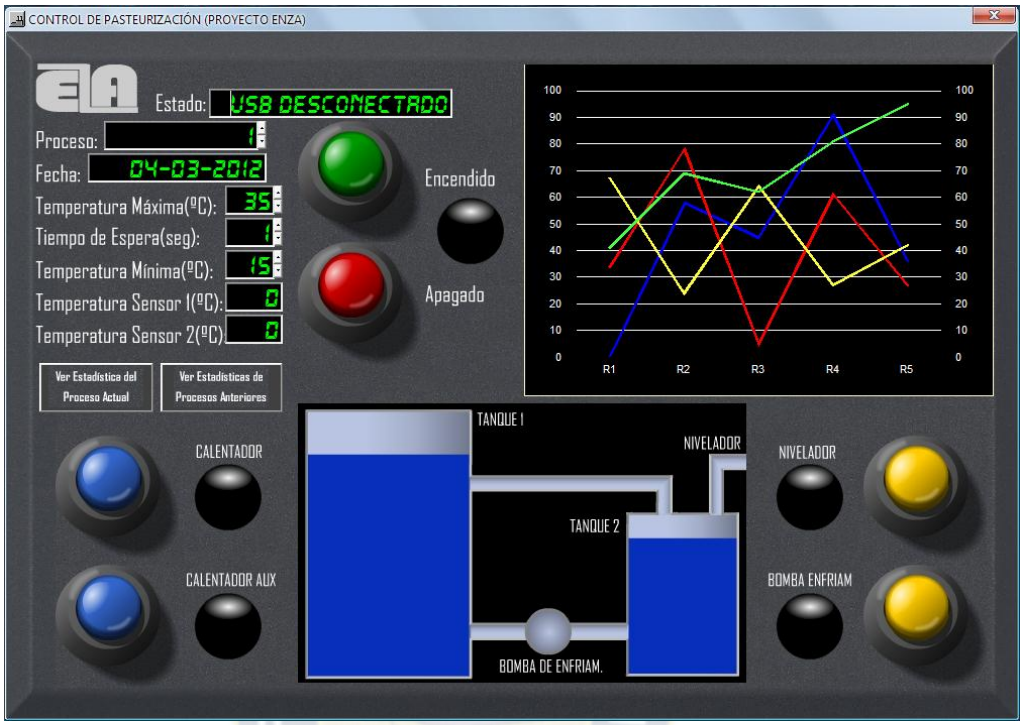


Figura 3.8. Pantalla Principal.
Fuente: Elaboración Propia

- Pantalla de Cuadros Estadísticos.

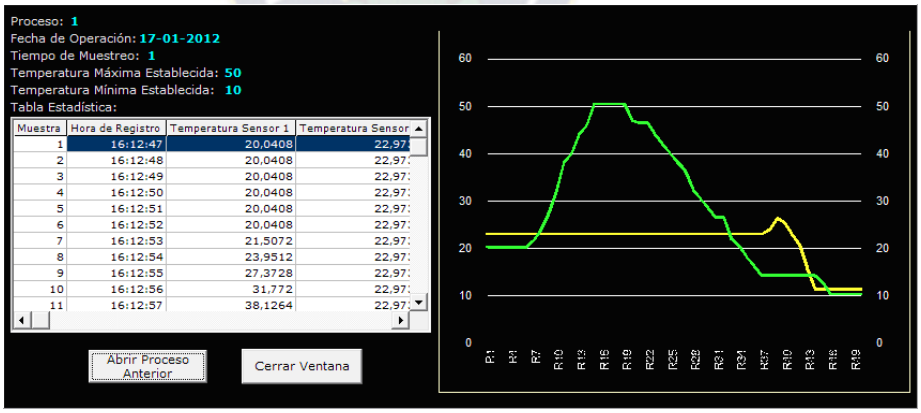


Figura 3.9. Cuadro Estadístico.
Fuente: Elaboración Propia

- **CODIGO FUENTE**

'SISTEMA DE CONTROL EN LA PASTEURIZACIÓN DE LA LECHE

‘DECLARACIÓN DE VARIABLES GLOBALES

```
Private Const VendorID = 1121
Private Const ProductID = 32
Dim BufferIn(0 To 8) As Byte
Dim BufferOut(0 To 8) As Byte
Dim lectura As String
Dim band As String
Dim R1 As Integer
Dim R2 As Integer
Dim R3 As Integer
Dim R4 As Integer
Dim tanque As Integer
Dim sel_1, sel_2, sel_3, sel_4 As Boolean
Dim llave1, llave2 As Boolean
Dim aux_stado As Integer
' variables de la curva
Private Values() As Single
Private Valores() As Single
Dim Mx(1 To 9000, 1 To 2) As Long
Dim Contador, j As Long
Dim temp As Double
Dim RT As Double
Dim RTO As Double
Dim B As Double
Dim TO0 As Double
'FUNCIONES APLICADAS A CADA OBJETO DEL FORMULARIO PRINCIPAL
Private Sub btn_1_Click()
If sel_1 = True Then
   ' ENCENDIDO DEL RESISTOR 1
```

```
    btn_1.Picture = i_5.Picture
    sel_1 = False
    led_r1.Picture = led_on.Picture
    BufferOut(0) = 0
    BufferOut(4) = "1"
    hidWriteEx VendorID, ProductID, BufferOut(0)
    R1 = 1
Else
    ' APAGADO DEL RESISTOR 1
    btn_1.Picture = i_6.Picture
    sel_1 = True
    led_r1.Picture = led_off.Picture
    BufferOut(0) = 0
    BufferOut(4) = "0"
    hidWriteEx VendorID, ProductID, BufferOut(0)
    R1 = 0
End If
End Sub

Private Sub btn_2_Click()
If sel_2 = True Then
    ' ENCENDIDO DEL RESISTOR 2
    btn_2.Picture = i_5.Picture
    sel_2 = False
    led_r2.Picture = led_on.Picture
    BufferOut(0) = 0
    BufferOut(5) = "1"
    hidWriteEx VendorID, ProductID, BufferOut(0)
    R2 = 1
```

```
Else
   ' APAGADO DEL RESISTOR 2
   btn_2.Picture = i_6.Picture
   sel_2 = True
   led_r2.Picture = led_off.Picture
   BufferOut(0) = 0
   BufferOut(5) = "0"
   hidWriteEx VendorID, ProductID, BufferOut(0)
   R2 = 0
End If
End Sub

Private Sub btn_3_Click()
If sel_3 = True Then
   ' ENCENDIDO DEL MOTOR 1
   btn_3.Picture = i_7.Picture
   sel_3 = False
   led_r3.Picture = led_on.Picture
   BufferOut(0) = 0
   BufferOut(6) = "1"
   hidWriteEx VendorID, ProductID, BufferOut(0)
   R3 = 1
Else
   ' APAGADO DEL MOTOR 1
   btn_3.Picture = i_8.Picture
   sel_3 = True
   led_r3.Picture = led_off.Picture
   BufferOut(0) = 0
   BufferOut(6) = "0"
```

```
    hidWriteEx VendorID, ProductID, BufferOut(0)
    R3 = 0
End If
End Sub

Private Sub btn_4_Click()
If sel_4 = True Then
    ' ENCENDIDO DEL MOTOR 2
    btn_4.Picture = i_7.Picture
    sel_4 = False
    led_r4.Picture = led_on.Picture
    BufferOut(0) = 0
    BufferOut(7) = "1"
    hidWriteEx VendorID, ProductID, BufferOut(0)
    R4 = 1
Else
    ' APAGADO DEL MOTOR 2
    btn_4.Picture = i_8.Picture
    sel_4 = True
    led_r4.Picture = led_off.Picture
    BufferOut(0) = 0
    BufferOut(7) = "0"
    hidWriteEx VendorID, ProductID, BufferOut(0)
    R4 = 0
End If
End Sub

Private Sub btn_off_Click()
Apaga_todo
```

```
llave1 = False
llave2 = False
tanque = 0
tmrMuestra.Enabled = False
imgCentral.Picture = TANQUE1.Picture
BufferOut(0) = 0
BufferOut(1) = "0"
hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub

Private Sub Apaga_todo()
led_state.Picture = led_off.Picture
led_r1.Picture = led_off.Picture
led_r2.Picture = led_off.Picture
led_r3.Picture = led_off.Picture
led_r4.Picture = led_off.Picture
btn_1.Picture = i_6.Picture
btn_2.Picture = i_6.Picture
btn_3.Picture = i_8.Picture
btn_4.Picture = i_8.Picture
sel_1 = True: sel_2 = True: sel_3 = True: sel_4 = True
End Sub

Private Sub btn_off_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
btn_off.Picture = i_3.Picture
End Sub
Private Sub btn_off_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
btn_off.Picture = i_4.Picture
End Sub
Private Sub btn_on_Click()
' VERIFICAMOS QUE POR LO MENOS UNA RESISTENCIA Y UN MOTOR DE ENFRIAMIENTO ESTÉN ENCENDIDOS
If sel_1 = False Or sel_2 = False Then
    Contador = 0
    tmrMuestra.Enabled = True
    led_state.Picture = led_on.Picture
    BufferOut(0) = 0
    BufferOut(1) = 1
    BufferOut(2) = Val(txtTempMin.Text)  ' temperatura minima
    BufferOut(3) = Val(txtTempMax.Text)  ' temperatura maxima
    BufferOut(4) = R1
    BufferOut(5) = R2
    BufferOut(6) = R3
    BufferOut(7) = R4
    BufferOut(8) = Val(txtTiempo.Text)
    hidWriteEx VendorID, ProductID, BufferOut(0)
Else
    btn_1_Click
    Contador = 0
    tmrMuestra.Enabled = True
    led_state.Picture = led_on.Picture
    BufferOut(0) = 0
    BufferOut(1) = 1
    BufferOut(2) = Val(txtTempMin.Text)  ' temperatura minima
    BufferOut(3) = Val(txtTempMax.Text)  ' temperatura maxima
    BufferOut(4) = R1
```

```
        BufferOut(5) = R2
        BufferOut(6) = R3
        BufferOut(7) = R4
        BufferOut(8) = Val(txtTiempo.Text)
        hidWriteEx VendorID, ProductID, BufferOut(0)
End If
FlexResultado.Clear
FlexResultado.Rows = 2
FlexResultado.FormatString = "Muestra |Hora de Registro |Temperatura Sensor 1 |Temperatura Sensor 2 "
dato_1.Caption = txtProceso.Text
dato_2.Caption = txtFecha.Text
dato_3.Caption = txtTiempo.Text
dato_4.Caption = txtTempMax.Text
dato_5.Caption = txtTempMin.Text
End Sub

Private Sub btn_on_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
btn_on.Picture = i_1.Picture
End Sub

Private Sub btn_on_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
btn_on.Picture = i_2.Picture
End Sub
Private Sub cmdAbrir_Click() ‘ FUNCION PARA ABRIR EL ARCHIVO DE DATOS
Dim n As Integer
```

```
Dim s As String
Dim cont As Integer
Dim sFile As String
dlgCommonDialog.DialogTitle = "Abrir archivo EZA"
dlgCommonDialog.CancelError = True
On Error GoTo errHandler
dlgCommonDialog.Filter = "Archivo Sistema de Control EZA (*.eza)|*.eza"
dlgCommonDialog.Flags = &H4& + &H1000&
dlgCommonDialog.InitDir = App.Path & "\Examples"
dlgCommonDialog.ShowOpen
sFile = dlgCommonDialog.FileName
Open dlgCommonDialog.FileName For Input As #2
   Input #2, s: ant_1.Caption = s
   Input #2, s: ant_2.Caption = s
   Input #2, s: ant_3.Caption = s
   Input #2, s: ant_4.Caption = s
   Input #2, s: ant_5.Caption = s
   Input #2, s: n = (Int(s)) - 2
   FlexResultado2.Clear
   FlexResultado2.FormatString = "Muestra |Hora de Registro |Temperatura
Sensor 1 |Temperatura Sensor 2 "
   FlexResultado2.Rows = 2
   ReDim Valores(1 To n, 1 To 2)
   For cont = 1 To n
      FlexResultado2.TextMatrix(cont, 0) = cont
      Input #2, s: FlexResultado2.TextMatrix(cont, 1) = s
      Input #2, s: FlexResultado2.TextMatrix(cont, 2) = s
      Valores(cont, 1) = s
      Input #2, s: FlexResultado2.TextMatrix(cont, 3) = s
```

```
        Valores(cont, 2) = s
        FlexResultado2.AddItem ""
    Next cont
    Close #2
    ' dibujamos la curva
    curva_aux_2.ChartData = Valores
    curva_aux_2.chartType = 3
    curva_aux_2.RowCount = n
    curva_aux_2.ColumnCount = 2
    curva_aux_2.Refresh
    Exit Sub
errHandler:
Exit Sub
End Sub

Private Sub cmdAnteriores_Click()
ant_1.Caption = "":ant_2.Caption = "":ant_3.Caption = "":ant_4.Caption = "":ant_5.Caption = ""
FlexResultado2.Clear
cmdAnteriores.Enabled = False
cmdEstadistica.Enabled = False
frameAnteriores.Visible = True
End Sub
Private Sub cmdCerrar_Click()
frameEstadistico.Visible = False
cmdEstadistica.Enabled = True
cmdAnteriores.Enabled = True
End Sub
Private Sub cmdCerrar2_Click()
```

```
cmdAnteriores.Enabled = True
cmdEstadistica.Enabled = True
frameAnteriores.Visible = False
End Sub
Private Sub cmdEstadistica_Click()
If Contador <> 0 Then
   curva_aux.ChartData = Values
   curva_aux.chartType = 3
   curva_aux.RowCount = Contador
   curva_aux.ColumnCount = 2
   curva_aux.Refresh
End If
frameEstadistico.Visible = True
frameAnteriores.Visible = False
cmdEstadistica.Enabled = False
cmdAnteriores.Enabled = False
End Sub

Private Sub cmdGuardar_Click()'FUNCION PARA GUARDAR ARCHIVO DE
DATO
   Dim sFile As String
   Dim cont As Integer
With dlgCommonDialog
      .DialogTitle = "Guardar archivo EZA como..."
      .CancelError = True
      On Error GoTo errHandler
      .Filter = "Archivo Sistema de Control EZA (*.eza)|*.eza"
      .Flags = &H2& + &H4&
      .InitDir = App.Path & "\Examples"
```

```
        .ShowSave
        sFile = .FileName
End With
Open dlgCommonDialog.FileName For Output As #1
Write #1, dato_1.Caption, dato_2.Caption, dato_3.Caption, dato_4.Caption, dato_5.Caption, FlexResultado.Rows
For cont = 1 To Int(FlexResultado.Rows) - 2
   Write #1, FlexResultado.TextMatrix(cont, 1), FlexResultado.TextMatrix(cont, 2), FlexResultado.TextMatrix(cont, 3)
Next cont
Close #1
MsgBox "Archivo guardado satisfactoriamente...", vbOKOnly + vbExclamation, "Control EZA"
Exit Sub
errHandler:
Exit Sub
End Sub
Private Sub curva_PointActivated(Series As Integer, DataPoint As Integer, MouseFlags As Integer, Cancel As Integer)
   Dim vtPoint
   With curva
      .Column = Series
      .Row = DataPoint
      vtPoint = InputBox("Change the data point:", , .Data)
      If vtPoint <> "" Then
         If IsNumeric(vtPoint) Then
            .Data = vtPoint
         Else
            MsgBox "Dato no valido", vbExclamation + vbOKOnly, "ERROR"
```

```
            End If
        End If
    End With
End Sub
Private Sub curva_PointSelected(Series As Integer, DataPoint As Integer, MouseFlags As Integer, Cancel As Integer)
    curva.Column = Series
    curva.Row = DataPoint
End Sub
Private Sub Form_Load()
   ConnectToHID (Me.hwnd)
   Move (Screen.Width - Width) \ 2, (Screen.Height - Height) \ 2
   txtState.Text = "USB DESCONECTADO"
   sel_1 = True: sel_2 = True: sel_3 = True: sel_4 = True
   txtFecha.Text = Format(Date, "dd""-""mm""-""yyyy")
   tanque = 0
   llave1 = False: llave2 = False
End Sub
Private Sub Form_Unload(Cancel As Integer)
   DisconnectFromHID
End Sub
'evento de conexion del dispositivo HID
Public Sub OnPlugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
   txtState.Text = "USB CONECTADO"
   End If
End Sub
'evento de desconexion del dispositivo HID
```

```
Public Sub OnUnplugged(ByVal pHandle As Long)
  If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
  txtState.Text = "USB DESCONECTADO"
  End If
End Sub
'Compara los ID con los ID del controlador HID
Public Sub OnChanged()
  Dim DeviceHandle As Long
  DeviceHandle = hidGetHandle(VendorID, ProductID)
  hidSetReadNotify DeviceHandle, True
End Sub
'Evento de llegada de datos HID por USB, sirve para leer informacion
Public Sub OnRead(ByVal pHandle As Long)
If hidRead(pHandle, BufferIn(0)) Then
   ' Codigo para leer
   ' para el sensor 1
   band = Chr(BufferIn(8))
   If (band = "#") Then
      lectura = Val(BufferIn(4) & BufferIn(3) & BufferIn(2) & BufferIn(1))
      B = 4425
      TO0 = 341
      RTO = 480
      temp = (5 * (Val(lectura))) / 1023#
      RT = (10000 * temp) / (5 - temp)
      txtS1.Text = (1 / (((1 / B) * Log(RT / RTO)) + (1 / TO0))) - 273
   End If
   ' para el sensor 2
   band = Chr(BufferIn(8))
```

```
If (band = "*") Then
   lectura = Val(BufferIn(4) & BufferIn(3) & BufferIn(2) & BufferIn(1))
   txtS2.Text = (4.888 * Val(lectura)) / 10
End If
' para las banderas
band = Chr(BufferIn(8))
If (band = "$") Then
   lectura = Val(BufferIn(1))
   aux_stado = lectura
   If aux_stado = 3 Then
      led_r1.Picture = led_off.Picture
      led_r2.Picture = led_off.Picture
      btn_1.Picture = i_6.Picture
      btn_2.Picture = i_6.Picture
      sel_1 = True: sel_2 = True
   End If
   If aux_stado = 2 Then
      If llave2 = False Then
         tanque = 2
         llave2 = True
      End If
      led_r1.Picture = led_off.Picture
      led_r2.Picture = led_off.Picture
      btn_1.Picture = i_6.Picture
      btn_2.Picture = i_6.Picture
      sel_1 = True: sel_2 = True
      If sel_3 = True Then
         btn_3_Click
      End If
```

```
        Else
            If llave1 = False Then
                tanque = 1
                llave1 = True
            End If
            If Val(txtS1.Text) < Val(txtTempMin.Text) Then
                btn_off_Click
            End If
        End If
    End If
End If
End Sub

Private Sub tmrMuestra_Timer()
Contador = Contador + 1
Mx(Contador, 1) = txtS1.Text
Mx(Contador, 2) = txtS2.Text
ReDim Values(1 To Contador, 1 To 2)
For j = 1 To Contador
Values(j, 1) = Mx(j, 1)
Values(j, 2) = Mx(j, 2)
Next j
' dibujamos la curva
curva.ChartData = Values
curva.chartType = 3
curva.RowCount = Contador
curva.ColumnCount = 2
curva.Refresh 'escribimos en la tabla
    FlexResultado.TextMatrix(FlexResultado.Rows - 1, 0) = Contador
```

```
    FlexResultado.TextMatrix(FlexResultado.Rows - 1, 1) = Time
    FlexResultado.TextMatrix(FlexResultado.Rows - 1, 2) = txtS1.Text
    FlexResultado.TextMatrix(FlexResultado.Rows - 1, 3) = txtS2.Text
    FlexResultado.AddItem ""
' Ahora dibujamos el tanque verificamos que esta en CALENTAMIENTO
If tanque = 1 Then
    imgCentral.Picture = TANQUE1.Picture
End If
' verificamos que esta en ENFRIAMIENTO
If tanque = 2 Then
    If sel_4 = False Then
        imgCentral.Picture = TANQUE3.Picture
    Else
        imgCentral.Picture = TANQUE2.Picture
    End If
End If
End Sub

Private Sub vs_Espera_Change()
txtTiempo.Text = Format(vs_Espera.Value)
End Sub

Private Sub vs_Tmax_Change()
txtTempMax.Text = Format(vs_Tmax.Value)
End Sub

Private Sub vs_Tmin_Change()
txtTempMin.Text = Format(vs_Tmin.Value)
End Sub
```

```
Private Sub vsProceso_Change()
txtProceso.Text = Format(vsProceso.Value)
End Sub
```

- **MODULO HIDDLLInterface.bas**

```
' this is the interface to the HID controller DLL - you should not
' normally need to change anything in this file.
' WinProc() calls your main form 'event' procedures - these are currently set to..
'
' MainForm.OnPlugged(ByVal pHandle as long)
' MainForm.OnUnplugged(ByVal pHandle as long)
' MainForm.OnChanged()
' MainForm.OnRead(ByVal pHandle as long)
Option Explicit
' HID interface API declarations...
Declare Function hidConnect Lib "mcHID.dll" Alias "Connect" (ByVal pHostWin As Long) As Boolean
Declare Function hidDisconnect Lib "mcHID.dll" Alias "Disconnect" () As Boolean
Declare Function hidGetItem Lib "mcHID.dll" Alias "GetItem" (ByVal pIndex As Long) As Long
Declare Function hidGetItemCount Lib "mcHID.dll" Alias "GetItemCount" () As Long
Declare Function hidRead Lib "mcHID.dll" Alias "Read" (ByVal pHandle As Long, ByRef pData As Byte) As Boolean
Declare Function hidWrite Lib "mcHID.dll" Alias "Write" (ByVal pHandle As Long, ByRef pData As Byte) As Boolean
```

Declare Function hidReadEx Lib "mcHID.dll" Alias "ReadEx" (ByVal pVendorID As Long, ByVal pProductID As Long, ByRef pData As Byte) As Boolean

Declare Function hidWriteEx Lib "mcHID.dll" Alias "WriteEx" (ByVal pVendorID As Long, ByVal pProductID As Long, ByRef pData As Byte) As Boolean

Declare Function hidGetHandle Lib "mcHID.dll" Alias "GetHandle" (ByVal pVendoID As Long, ByVal pProductID As Long) As Long

Declare Function hidGetVendorID Lib "mcHID.dll" Alias "GetVendorID" (ByVal pHandle As Long) As Long

Declare Function hidGetProductID Lib "mcHID.dll" Alias "GetProductID" (ByVal pHandle As Long) As Long

Declare Function hidGetVersion Lib "mcHID.dll" Alias "GetVersion" (ByVal pHandle As Long) As Long

Declare Function hidGetVendorName Lib "mcHID.dll" Alias "GetVendorName" (ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long

Declare Function hidGetProductName Lib "mcHID.dll" Alias "GetProductName" (ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long

Declare Function hidGetSerialNumber Lib "mcHID.dll" Alias "GetSerialNumber" (ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long

Declare Function hidGetInputReportLength Lib "mcHID.dll" Alias "GetInputReportLength" (ByVal pHandle As Long) As Long

Declare Function hidGetOutputReportLength Lib "mcHID.dll" Alias "GetOutputReportLength" (ByVal pHandle As Long) As Long

Declare Sub hidSetReadNotify Lib "mcHID.dll" Alias "SetReadNotify" (ByVal pHandle As Long, ByVal pValue As Boolean)

```
Declare Function hidIsReadNotifyEnabled Lib "mcHID.dll" Alias "IsReadNotifyEnabled" (ByVal pHandle As Long) As Boolean
Declare Function hidIsAvailable Lib "mcHID.dll" Alias "IsAvailable" (ByVal pVendorID As Long, ByVal pProductID As Long) As Boolean

' windows API declarations - used to set up messaging...
Private Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" (ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

' windows API Constants
Private Const WM_APP = 32768
Private Const GWL_WNDPROC = -4
' HID message constants
Private Const WM_HID_EVENT = WM_APP + 200
Private Const NOTIFY_PLUGGED = 1
Private Const NOTIFY_UNPLUGGED = 2
Private Const NOTIFY_CHANGED = 3
Private Const NOTIFY_READ = 4

' local variables
Private FPrevWinProc As Long    ' Handle to previous window procedure
Private FWinHandle As Long      ' Handle to message window

' Set up a windows hook to receive notification
```

```
' messages from the HID controller DLL - then connect
' to the controller
Public Function ConnectToHID(ByVal pHostWin As Long) As Boolean
  FWinHandle = pHostWin
  ConnectToHID = hidConnect(FWinHandle)
  FPrevWinProc = SetWindowLong(FWinHandle, GWL_WNDPROC,
AddressOf WinProc)
End Function
' Unhook from the HID controller and disconnect...
Public Function DisconnectFromHID() As Boolean
  DisconnectFromHID = hidDisconnect
  SetWindowLong FWinHandle, GWL_WNDPROC, FPrevWinProc
End Function
' This is the procedure that intercepts the HID controller messages...
Private Function WinProc(ByVal pHWnd As Long, ByVal pMsg As Long,
ByVal wParam As Long, ByVal lParam As Long) As Long
  If pMsg = WM_HID_EVENT Then
    Select Case wParam
' HID device has been plugged message...
      Case Is = NOTIFY_PLUGGED
        MainForm.OnPlugged (lParam)
' HID device has been unplugged
      Case Is = NOTIFY_UNPLUGGED
        MainForm.OnUnplugged (lParam)
' controller has changed...
      Case Is = NOTIFY_CHANGED
        MainForm.OnChanged
' read event...
      Case Is = NOTIFY_READ
```

```
            MainForm.OnRead (lParam)
        End Select
    End If
    ' next...
      WinProc = CallWindowProc(FPrevWinProc, pHWnd, pMsg, wParam, lParam)
    End Function
```

3.9.4. PRUEBA DEL PROTOTIPO

Durante las dos fases tanto de Análisis y Diseño como de la Construcción del Prototipo, se realizaron pruebas y/o evaluaciones de verificación tanto interna como externa cuya consistencia se verificó a través del acoplamiento y cohesión de los módulos.

1. Pruebas de módulo con datos de prueba

PROCESO	RESULTADO	DEFICIENCIAS
Proceso de Calentamiento.	- El Sistema identifica al proceso de Calentamiento, ya que automáticamente activa los mecanismos de necesarios para el Calentamiento del Tanque. - Se ejecuta el proceso automáticamente con la activación del pulsador general.	Ninguna.
Proceso de Espera	- El Sistema entra al proceso automático de espera, una vez que se haya alcanzado el nivel de temperatura máxima alcanzada.	Deficiencia en el tiempo de espera, Microcontrolador encargado de realizar el proceso de Espera.
Proceso de Enfriamiento	- El Sistema entra al proceso automático de enfriamiento una vez que haya terminado el proceso de espera. Este proceso se ejecuta hasta alcanzar el nivel de	Ninguna

	temperatura mínima definida en los parámetros de Pasteurización	

Cuadro 3.2.: Evaluación de Módulos
Fuente: Elaboración Propia

Se dedujo que es necesario controlar rigurosamente los datos que son enviados del microcontrolador a la interfase de la PC, no solamente en la monitor sino también en el Módulo LCD.

2. Pruebas de integración.

Esta prueba se realizó para evaluar la integración entre módulos a través del acoplamiento, verificando que la integración del Microcontrolador con la Aplicación no tenga problemas en su comunicación, no tiene ningún problema.

3. Pruebas de desempeño.

En esta prueba se tomo en cuenta las entradas y salidas del Sistema, el cual denotó eficiencia a la hora de aplicarlas; se dedujo que la entrada de pocos datos contribuye al procesos de pasteurización, en cuestión a las salidas se obtuvo una respuesta inmediata del sistema, ya sea en la activación de los mecanismos, y del almacenamiento de toda la información.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO IV
CONCLUSIONES Y RECOMENDACIONES

4.1. INTRODUCCIÓN

A la conclusión del sistema (Prototipo Terminado), se toma en cuenta que gracias a la metodología presentada en este proyecto, la cual puede servir como base para futuros trabajos, ha contribuido de gran manera para la construcción de la misma, tomando en cuenta que simplemente son tres fases a la que se está sujeto, es por eso que el desarrollo del prototipo ha alcanzado a un nivel en la cual ha cumplido con todos los requerimientos expuestos anteriormente.

4.2. CONCLUSIONES

- La aplicación del sistema al proceso de Pasteurización de la Leche facilita el trabajo ya que los procesos son automáticos, tomando en cuenta que todo está centralizado a un panel de control.
- La interfaz del usuario determina el grado y/o nivel de aplicación del Sistema, ya que utilizando una interfaz amigable, permite que el usuario interactúe de una forma dinámica.
- Deja abierta la posibilidad de incluir más mecanismos Actuadores en el proceso de Pasteurización.
- Con la conjunción de metodologías, se pudo afianzar el trabajo que se realizó.

4.3. RECOMENDACIONES

- Tomar en cuenta que los procesos están sujetos a Métodos ya establecidos dentro del proceso de Pasteurización, caso del proyecto a la Pasteurización Lenta.
- Tomar en cuenta que la abstracción del conocimiento del usuario será la base fundamental para la construcción del sistema, para tal efecto es necesario trabajar con un solo usuario, porque de está manera la información que se consiga será absolutamente confiable y coherente sobre todas las cosas.

- Consolidar los requerimientos tanto del usuario como del sistema, con un análisis riguroso.
- Tomar en cuenta que ésta metodología permite realizar ajustes permanentes durante todo el proceso de desarrollo del sistema, facilitando de esta manera el trabajo que se realiza.

REFERENCIAS BIBLIOGRÁFICAS

RAMÓN PALLAS ARENY: **SENSORES Y ACONDICIONADORES DE SEÑAL PRÁCTICAS**: PUBLICADO POR MARCOMBO. 2007.

M. ANGULO USATEGUI e IGNACIO ANGULO MARTÍNEZ: **MICROCONTROLADORES PIC**. PUBLICADO POR Mc Graw Hill. 2003.

PRACTICAL CONCEPTS INCORPORATES (PCI): **SISTEMA DE MANEJO DE PROYECTOS**: PUBLICADO POR SMP s.E. s.L. 1989.

J. ELDON WHITESITT: **ÁLGEBRA BOOLEANA Y SUS APLICACIONES.** PUBLICADO POR C.E.C. S.A.

JOSEP BALCELLS Y JOSÉ LUIS ROMERAL: **AUTÓMATAS PROGRAMABLES.** PUBLICADO POR Marcombo

L.J. GIACOLETTO: **ELECTRONIC DESIGNER'S HANDBOOK - 2DA EDITION.** PUBLICADO POR MCGRAW-HILL BOOKS

LUIS GIL SÁNCHEZ**: INTRODUCCIÓN A LA ELECTRÓNICA DIGITAL.** PUBLICADO POR UNIVERSIDAD POLITÉCNICA DE VALENCIA.

ABRAHAM IOFFE**: SEMICONDUCTOR THERMOELEMENTS.** PUBLICADO POR. AKADEMIA NAUK.

ALEJANDRO PORRAS CRIADO Y ANTONIO PLÁCIDO MONTANERO MOLINA: **AUTÓMATAS PROGRAMABLES. FUNDAMENTO, MANEJO, INSTALACIÓN Y PRÁCTICAS.** PUBLICADO POR. ED. MCGRAW-HILL

JHON GAME: **USB – HARDWARE AND SOFTWARE**. PUBLICADO POR ED. SOLARI, KOSAR JAFF

- **PAGINAS WEB**

http://alcabot.org/seminario2006/SEM06_Sensores_V3_2.pdf

http://www.superrobotica.com/Sensores.htm

http://www.superrobotica.com/S320110.htm

http://www.teleline.terra.es/personal/fremiro

http://www.neoteo.com/tag/pic16f84a.neo

http://www.directindustry.es/fabricante-industrial/75154.html

http://www.x-robotics.com/sensores.htm

http://www.superrobotica.com/Sensores.htm

http://www.junun.org/MarkIII/Store.jsp#item1

ANEXO A

Especificaciones PIC18F4550 (MICROCHIP)

PIC18F2455/2550/4455/4550
Data Sheet

28/40/44-Pin, High Performance,
Enhanced Flash, USB Microcontrollers
with nanoWatt Technology

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

MICROCHIP

PIC18F2455/2550/4455/4550

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 Oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

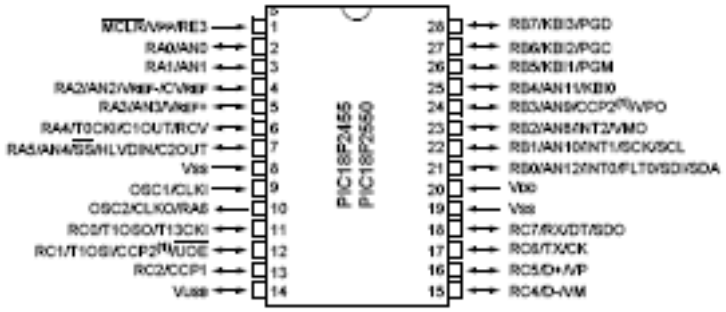
Special Microcontroller Features:

- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide Operating Voltage Range (2.0V to 5.5V)

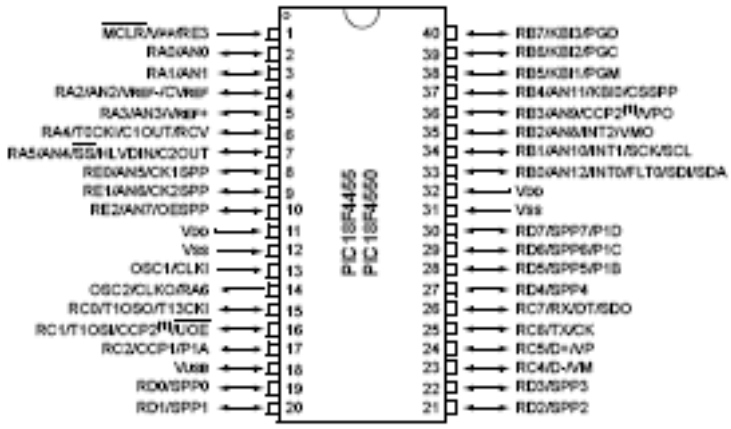
Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin PDIP, SOIC



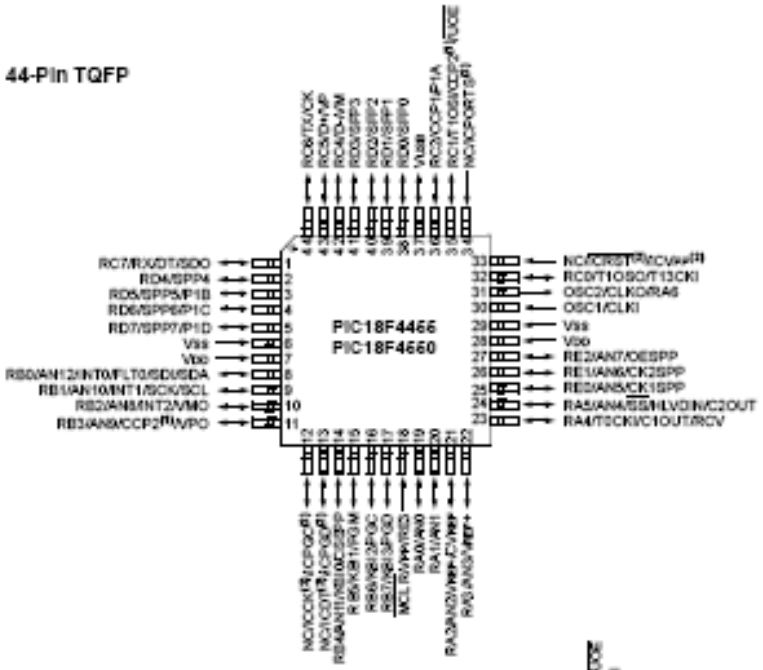
40-Pin PDIP



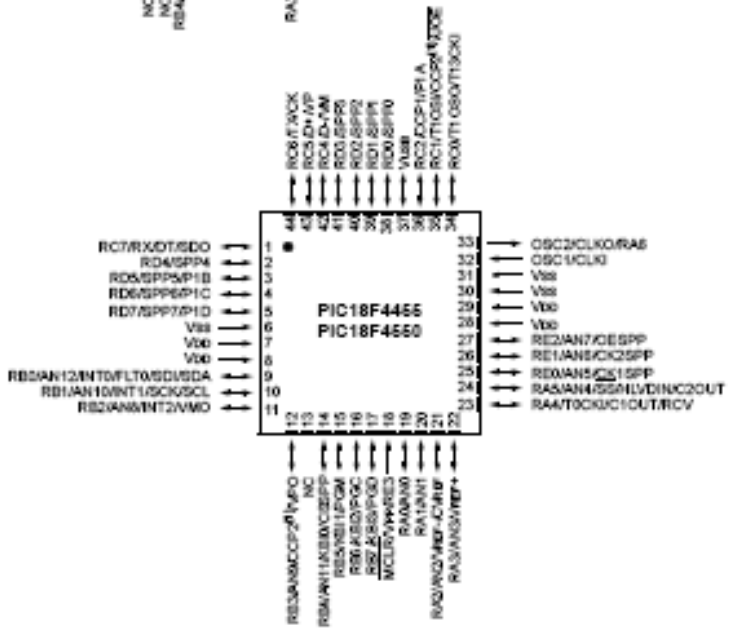
Note 1: RB3 is the alternate pin for CCP2 multiplexing.

Pin Diagrams (Continued)

44-Pin TQFP



44-Pin QFN



Note 1: RB3 is the alternate pin for CCP2 multiplexing.
2: Special ICPORTS features available in select circumstances. See Section 25.0 "Special ICPORT Features (Designated Packages Only)" for more information.

1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

- PIC18F2455
- PIC18F2550
- PIC18F4455
- PIC18F4550
- PIC18LF2455
- PIC18LF2550
- PIC18LF4455
- PIC18LF4550

This family of devices offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high endurance, Enhanced Flash program memory. In addition to these features, the PIC18F2455/2550/4455/4550 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2455/2550/4455/4550 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 28.0 "Electrical Characteristics"** for values.

1.1.2 UNIVERSAL SERIAL BUS (USB)

Devices in the PIC18F2455/2550/4455/4550 family incorporate a fully featured Universal Serial Bus communications module that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full-speed communication for all supported data transfer types. It also incorporates its own on-chip transceiver and 3.3V regulator and supports the use of external transceivers and voltage regulators.

1.1.3 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2455/2550/4455/4550 family offer twelve different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes using crystals or ceramic resonators.
- Four External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O).
- An internal oscillator block which provides an 8 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and VDD), as well as a range of 6 user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the High-Speed Crystal and External Oscillator modes, which allows a wide range of clock speeds from 4 MHz to 48 MHz.
- Asynchronous dual clock operation, allowing the USB module to run from a high-frequency oscillator while the rest of the microcontroller is clocked from an internal low-power oscillator.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-Programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine, located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F2455/2550/4455/4550 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Literal Offset Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages such as C.
- **Enhanced CCP Module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include auto-shutdown for disabling PWM outputs on interrupt or other select conditions and auto-restart to reactivate outputs once the condition has cleared.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include Automatic Baud Rate Detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated, without waiting for a sampling period and thus, reducing code overhead.
- **Dedicated ICD/ICSP Port:** These devices introduce the use of debugger and programming pins that are not multiplexed with other microcontroller features. Offered as an option in select packages, this feature allows users to develop I/O intensive applications while retaining the ability to program and debug in the circuit.

1.3 Details on Individual Family Members

Devices in the PIC18F2455/2550/4455/4550 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in six ways:

1. Flash program memory (24 Kbytes for PIC18FX455 devices, 32 Kbytes for PIC18FX550).
2. A/D channels (10 for 28-pin devices, 13 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports and 1 input only port on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. CCP and Enhanced CCP implementation (28-pin devices have two standard CCP modules, 40/44-pin devices have one standard CCP module and one ECCP module).
5. Streaming Parallel Port (present only on 40/44-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1.

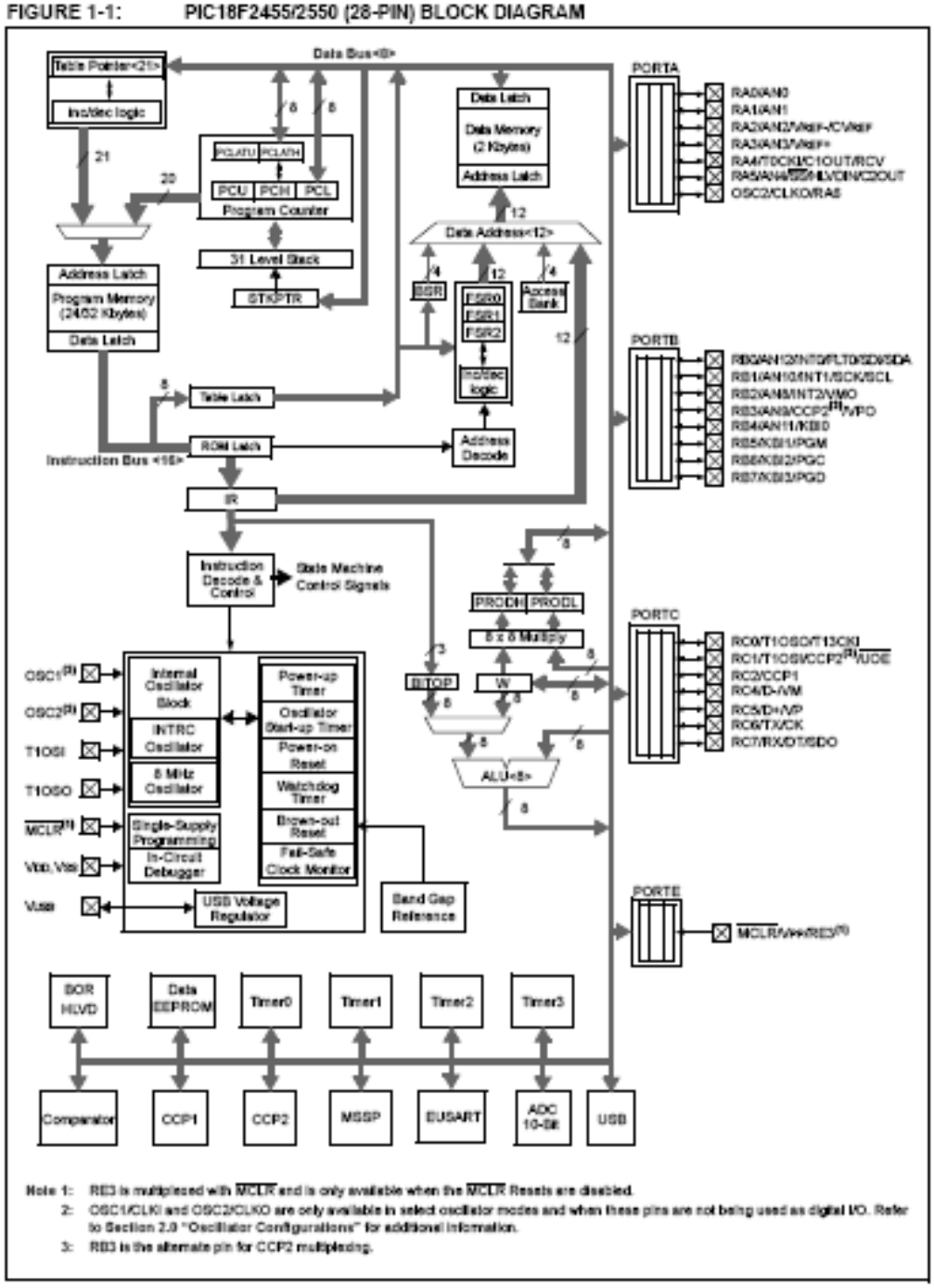
The pinouts for all devices are listed in Table 1-2 and Table 1-3.

Like all Microchip PIC18 devices, members of the PIC18F2455/2550/4455/4550 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an "F" in the part number (such as PIC18F2550), accommodate an operating VDD range of 4.2V to 5.5V. Low-voltage parts, designated by "LF" (such as PIC18LF2550), function over an extended VDD range of 2.0V to 5.5V.

TABLE 1-1: DEVICE FEATURES

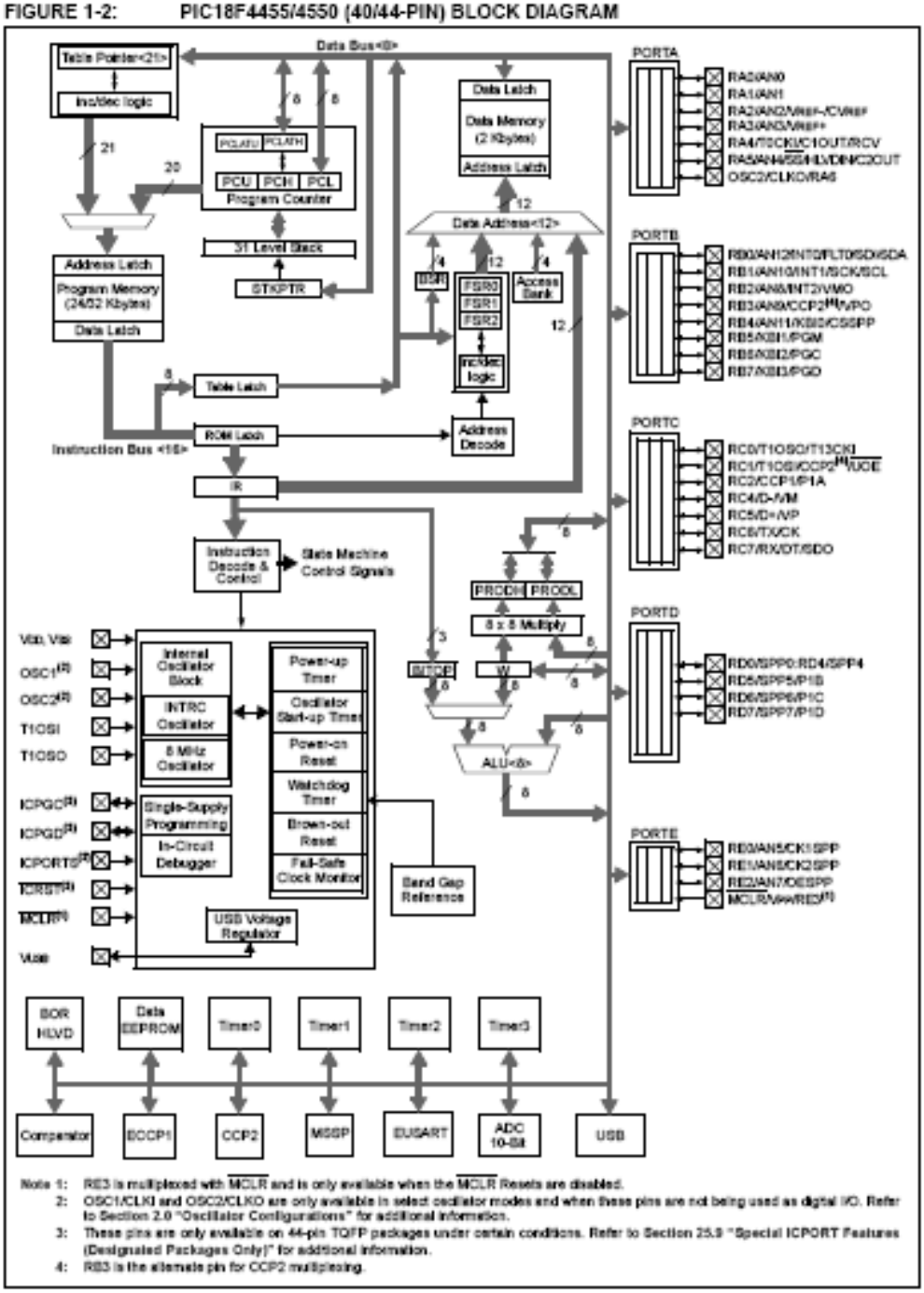
Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC	28-pin PDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

FIGURE 1-1: PIC18F2455/2550 (28-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with MCLR and is only available when the MCLR Resets are disabled.
2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to Section 2.0 "Oscillator Configurations" for additional information.
3: RB3 is the alternate pin for CCP2 multiplexing.

FIGURE 1-2: PIC18F4455/4550 (40/44-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with MCLR and is only available when the MCLR Resets are disabled.
2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to Section 2.0 "Oscillator Configurations" for additional information.
3: These pins are only available on 44-pin TQFP packages under certain conditions. Refer to Section 25.9 "Special ICPORT Features (Designated Packages Only)" for additional information.
4: RB3 is the alternate pin for CCP2 multiplexing.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
MCLR/VPP/RE3	1			Master Clear (input) or programming voltage (input).
MCLR		I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP		P		Programming voltage input.
RE3		I	ST	Digital input.
OSC1/CLKI	9			Oscillator crystal or external clock input.
OSC1		I	Analog	Oscillator crystal input or external clock source input.
CLKI		I	Analog	External clock source input. Always associated with pin function OSC1. (See OSC2/CLKO pin.)
OSC2/CLKO/RA6	10			Oscillator crystal or clock output.
OSC2		O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO		O	—	In select modes, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
				PORTA is a bidirectional I/O port.
RA0/AN0	2			
RA0		I/O	TTL	Digital I/O.
AN0		I	Analog	Analog Input 0.
RA1/AN1	3			
RA1		I/O	TTL	Digital I/O.
AN1		I	Analog	Analog Input 1.
RA2/AN2/VREF-/CVREF	4			
RA2		I/O	TTL	Digital I/O.
AN2		I	Analog	Analog Input 2.
VREF-		I	Analog	A/D reference voltage (low) input.
CVREF		O	Analog	Analog comparator reference output.
RA3/AN3/VREF+	5			
RA3		I/O	TTL	Digital I/O.
AN3		I	Analog	Analog Input 3.
VREF+		I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT/RCV	6			
RA4		I/O	ST	Digital I/O.
T0CKI		I	ST	Timer0 external clock input.
C1OUT		O	—	Comparator 1 output.
RCV		I	TTL	External USB transceiver RCV input.
RA5/AN4/$\overline{SS}$/ HLVDIN/C2OUT	7			
RA5		I/O	TTL	Digital I/O.
AN4		I	Analog	Analog Input 4.
$\overline{SS}$		I	TTL	SPI slave select input.
HLVDIN		I	Analog	High/Low-Voltage Detect input.
C2OUT		O	—	Comparator 2 output.
RA6	—	—	—	See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PDIP, SOIC	Pin Type	Buffer Type	Description
				PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/FLT0/SDI/SDA	21			
RB0		I/O	TTL	Digital I/O.
AN12		I	Analog	Analog input 12.
INT0		I	ST	External interrupt 0.
FLT0		I	ST	PWM Fault input (CCP1 module).
SDI		I	ST	SPI data in.
SDA		I/O	ST	I^2C™ data I/O.
RB1/AN10/INT1/SCK/SCL	22			
RB1		I/O	TTL	Digital I/O.
AN10		I	Analog	Analog input 10.
INT1		I	ST	External interrupt 1.
SCK		I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL		I/O	ST	Synchronous serial clock input/output for I^2C mode.
RB2/AN8/INT2/VMO	23			
RB2		I/O	TTL	Digital I/O.
AN8		I	Analog	Analog input 8.
INT2		I	ST	External interrupt 2.
VMO		O	—	External USB transceiver VMO output.
RB3/AN9/CCP2/VPO	24			
RB3		I/O	TTL	Digital I/O.
AN9		I	Analog	Analog input 9.
CCP2(1)		I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
VPO		O	—	External USB transceiver VPO output.
RB4/AN11/KBI0	25			
RB4		I/O	TTL	Digital I/O.
AN11		I	Analog	Analog input 11.
KBI0		I	TTL	Interrupt-on-change pin.
RB5/KBI1/PGM	26			
RB5		I/O	TTL	Digital I/O.
KBI1		I	TTL	Interrupt-on-change pin.
PGM		I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	27			
RB6		I/O	TTL	Digital I/O.
KBI2		I	TTL	Interrupt-on-change pin.
PGC		I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	28			
RB7		I/O	TTL	Digital I/O.
KBI3		I	TTL	Interrupt-on-change pin.
PGD		I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
				PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	11			
RC0		I/O	ST	Digital I/O.
T1OSO		O	—	Timer1 oscillator output.
T13CKI		I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2/$\overline{UOE}$	12			
RC1		I/O	ST	Digital I/O.
T1OSI		I	CMOS	Timer1 oscillator input.
CCP2(2)		I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
$\overline{UOE}$		—	—	External USB transceiver $\overline{OE}$ output.
RC2/CCP1	13			
RC2		I/O	ST	Digital I/O.
CCP1		I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
RC4/D-/VM	15			
RC4		I	TTL	Digital input.
D-		I/O	—	USB differential minus line (input/output).
VM		I	TTL	External USB transceiver VM input.
RC5/D+/VP	16			
RC5		I	TTL	Digital input.
D+		I/O	—	USB differential plus line (input/output).
VP		O	TTL	External USB transceiver VP input.
RC6/TX/CK	17			
RC6		I/O	ST	Digital I/O.
TX		O	—	EUSART asynchronous transmit.
CK		I/O	ST	EUSART synchronous clock (see RX/DT).
RC7/RX/DT/SDO	18			
RC7		I/O	ST	Digital I/O.
RX		I	ST	EUSART asynchronous receive.
DT		I/O	ST	EUSART synchronous data (see TX/CK).
SDO		O	—	SPI data out.
RE3	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin.
VUSB	14	O	—	Internal USB 3.3V voltage regulator.
VSS	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	P	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
MCLR/VPP/RE3	1	18	18			Master Clear (input) or programming voltage (input).
MCLR				I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP				P		Programming voltage input.
RE3				I	ST	Digital input.
OSC1/CLKI	13	32	30			Oscillator crystal or external clock input.
OSC1				I	Analog	Oscillator crystal input or external clock source input.
CLKI				I	Analog	External clock source input. Always associated with pin function OSC1. (See OSC2/CLKO pin.)
OSC2/CLKO/RA6	14	33	31			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTA is a bidirectional I/O port.
RA0/AN0	2	19	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	20	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4	21	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D reference voltage (low) input.
CVREF				O	Analog	Analog comparator reference output.
RA3/AN3/VREF+	5	22	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT/RCV	6	23	23			
RA4				I/O	ST	Digital I/O.
T0CKI				I	ST	Timer0 external clock input.
C1OUT				O	—	Comparator 1 output.
RCV				I	TTL	External USB transceiver RCV input.
RA5/AN4/$\overline{SS}$/HLVDIN/C2OUT	7	24	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
$\overline{SS}$				I	TTL	SPI slave select input.
HLVDIN				I	Analog	High/Low-Voltage Detect input.
C2OUT				O	—	Comparator 2 output.
RA6	—	—	—	—	—	See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the $\overline{DEBUG}$ Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/FLT0/SDI/SDA	33	9	8			
RB0				I/O	TTL	Digital I/O.
AN12				I	Analog	Analog input 12.
INT0				I	ST	External interrupt 0.
FLT0				I	ST	Enhanced PWM Fault input (ECCP1 module).
SDI				I	ST	SPI data in.
SDA				I/O	ST	I^2C™ data I/O.
RB1/AN10/INT1/SCK/SCL	34	10	9			
RB1				I/O	TTL	Digital I/O.
AN10				I	Analog	Analog input 10.
INT1				I	ST	External interrupt 1.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RB2/AN8/INT2/VMO	35	11	10			
RB2				I/O	TTL	Digital I/O.
AN8				I	Analog	Analog input 8.
INT2				I	ST	External interrupt 2.
VMO				O	—	External USB transceiver VMO output.
RB3/AN9/CCP2/VPO	36	12	11			
RB3				I/O	TTL	Digital I/O.
AN9				I	Analog	Analog input 9.
CCP2(1)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
VPO				O	—	External USB transceiver VPO output.
RB4/AN11/KBI0/CSSPP	37	14	14			
RB4				I/O	TTL	Digital I/O.
AN11				I	Analog	Analog input 11.
KBI0				I	TTL	Interrupt-on-change pin.
CSSPP				O	—	SPP chip select control output.
RB5/KBI1/PGM	38	15	15			
RB5				I/O	TTL	Digital I/O.
KBI1				I	TTL	Interrupt-on-change pin.
PGM				I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	39	16	16			
RB6				I/O	TTL	Digital I/O.
KBI2				I	TTL	Interrupt-on-change pin.
PGC				I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	40	17	17			
RB7				I/O	TTL	Digital I/O.
KBI3				I	TTL	Interrupt-on-change pin.
PGD				I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the $\overline{\text{DEBUG}}$ Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	15	34	32			
RC0				I/O	ST	Digital I/O.
T1OSO				O	—	Timer1 oscillator output.
T13CKI				I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2/ $\overline{UOE}$	16	35	35			
RC1				I/O	ST	Digital I/O.
T1OSI				I	CMOS	Timer1 oscillator input.
CCP2(2)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
$\overline{UOE}$				O	—	External USB transceiver $\overline{OE}$ output.
RC2/CCP1/P1A	17	36	36			
RC2				I/O	ST	Digital I/O.
CCP1				I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
P1A				O	TTL	Enhanced CCP1 PWM output, channel A.
RC4/D-/VM	23	42	42			
RC4				I	TTL	Digital input.
D-				I/O	—	USB differential minus line (input/output).
VM				I	TTL	External USB transceiver VM input.
RC5/D+/VP	24	43	43			
RC5				I	TTL	Digital input.
D+				I/O	—	USB differential plus line (input/output).
VP				I	TTL	External USB transceiver VP input.
RC6/TX/CK	25	44	44			
RC6				I/O	ST	Digital I/O.
TX				O	—	EUSART asynchronous transmit.
CK				I/O	ST	EUSART synchronous clock (see RX/DT).
RC7/RX/DT/SDO	26	1	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	EUSART asynchronous receive.
DT				I/O	ST	EUSART synchronous data (see TX/CK).
SDO				O	—	SPI data out.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the $\overline{DEBUG}$ Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTD is a bidirectional I/O port or a Streaming Parallel Port (SPP). These pins have TTL input buffers when the SPP module is enabled.
RD0/SPP0	19	38	38			
RD0				I/O	ST	Digital I/O.
SPP0				I/O	TTL	Streaming Parallel Port data.
RD1/SPP1	20	39	39			
RD1				I/O	ST	Digital I/O.
SPP1				I/O	TTL	Streaming Parallel Port data.
RD2/SPP2	21	40	40			
RD2				I/O	ST	Digital I/O.
SPP2				I/O	TTL	Streaming Parallel Port data.
RD3/SPP3	22	41	41			
RD3				I/O	ST	Digital I/O.
SPP3				I/O	TTL	Streaming Parallel Port data.
RD4/SPP4	27	2	2			
RD4				I/O	ST	Digital I/O.
SPP4				I/O	TTL	Streaming Parallel Port data.
RD5/SPP5/P1B	28	3	3			
RD5				I/O	ST	Digital I/O.
SPP5				I/O	TTL	Streaming Parallel Port data.
P1B				O	—	Enhanced CCP1 PWM output, channel B.
RD6/SPP6/P1C	29	4	4			
RD6				I/O	ST	Digital I/O.
SPP6				I/O	TTL	Streaming Parallel Port data.
P1C				O	—	Enhanced CCP1 PWM output, channel C.
RD7/SPP7/P1D	30	5	5			
RD7				I/O	ST	Digital I/O.
SPP7				I/O	TTL	Streaming Parallel Port data.
P1D				O	—	Enhanced CCP1 PWM output, channel D.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

TABLE 1-3: PIC18F4455/4550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTE is a bidirectional I/O port.
RE0/AN5/CK1SPP	8	25	25			
RE0				I/O	ST	Digital I/O.
AN5				I	Analog	Analog input 5.
CK1SPP				O	—	SPP clock 1 output.
RE1/AN6/CK2SPP	9	26	26			
RE1				I/O	ST	Digital I/O.
AN6				I	Analog	Analog input 6.
CK2SPP				O	—	SPP clock 2 output.
RE2/AN7/OESPP	10	27	27			
RE2				I/O	ST	Digital I/O.
AN7				I	Analog	Analog input 7.
OESPP				O	—	SPP output enable output.
RE3	—	—	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin.
VSS	12, 31	6, 30, 31	6, 29	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	7, 8, 28, 29	7, 28	P	—	Positive supply for logic and I/O pins.
VUSB	18	37	37	O	—	Internal USB 3.3V voltage regulator output.
NC/ICCK/ICPGC(3)	—	—	12			No Connect or dedicated ICD/ICSP™ port clock.
ICCK				I/O	ST	In-Circuit Debugger clock.
ICPGC				I/O	ST	ICSP programming clock.
NC/ICDT/ICPGD(3)	—	—	13			No Connect or dedicated ICD/ICSP port clock.
ICDT				I/O	ST	In-Circuit Debugger data.
ICPGD				I/O	ST	ICSP programming data.
NC/$\overline{ICRST}$/ICVPP(3)	—	—	33			No Connect or dedicated ICD/ICSP port Reset.
ICRST				I	—	Master Clear (Reset) input.
ICVPP				P	—	Programming voltage input.
NC/ICPORTS(3)	—	—	34	P	—	No Connect or 28-pin device emulation.
ICPORTS						Enable 28-pin device emulation when connected to VSS.
NC	—	13	—	—	—	No Connect.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the $\overline{DEBUG}$ Configuration bit is cleared.

ANEXO B

Compilador C para Microchip PICmicro

(Cánovas López Andrés)

1. INTRODUCCIÓN

Si queremos realizar la programación de los microcontroladores PIC en un lenguaje como el C, es preciso utilizar un *compilador de C*.

Dicho compilador nos genera ficheros en formato Intel-hexadedimal, que es el necesario para programar (utilizando un programador de PIC) un microcontrolador de 6, 8, 18 ó 40 patillas.

El compilador de C que vamos a utilizar es el PCW de la casa CCS Inc. A su vez, el compilador lo integraremos en un *entorno de desarrollo integrado* (IDE) que nos va a permitir desarrollar todas y cada una de las fases que se compone un proyecto, desde la edición hasta la compilación pasando por la depuración de errores. La última fase, a excepción de la depuración y retoques hardware finales, será programar el PIC.

Al igual que el compilador de Turbo C, éste *"traduce"* el código C del archivo fuente (.C) a lenguaje máquina para los microcontroladores PIC, generando así un archivo en formato hexadecimal (.HEX). Además de éste, también genera otros seis ficheros, tal como se observa en la figura de la siguiente página.

Finalmente, deciros que esta vez os presento los apuntes en soporte

electrónico, a diferencia de ocasiones anteriores que estaban en formato impreso. Es una experiencia nueva y que como toda prueba tiene sus riesgos, aunque espero y deseo que, de una u otra forma logre prender la llama de la ilusión, o por lo menos despertar el interés por el estudio de la electrónica y en particular de este mundo inacabado de la programación en C y los microcontroladores.

2. PROGRAMAS DE UTILIDAD

- **SIO**

  SIO (Serial Input Output) es un simple programa "terminal no inteligente" que puede ejecutarse desde el DOS para realizar entradas y salidas sobre un puerto serie. SIO es útil ya que muestra todos los caracteres entrantes, excepto los no imprimibles que mostrará su código hexadecimal en rojo.

- **PICCHIPS**

  PICCHIPS es un programa de utilidad que lee la base de datos de un dispositivo. El compilador utiliza esta base de datos para determinar las características específicas del dispositivo durante la compilación. Al ejecutar el programa sin ningún parámetro, listará todos los dispositivos (PIC) disponibles. Si especificamos un dispositivo como parámetro p.ej. pic16c84, es decir, escribimos picchips pic16c84, obtenemos información detallada sobre este dispositivo. A modo de ejemplo y para el citado PIC se obtiene la siguiente información:

  PIC16C84---------- Opcode:

  14 bits, ROM: 1024, RAM: 36, I/O: 13 HW: EEPROM(64) POR TIM0 TRIS
  RAM: 0C-2F
  Ports: [A:01234---] [B:01234567] [C:--------] [D:--------] [E:--------]
  Fuses: LP: 0003/0000 XT: 0003/0001
  HS: 0003/0002 RC: 0003/0003
  NOWDT: 0004/0000 WDT: 0004/0004
  NOPUT: 0008/0000 PUT: 0008/0008
  PROTECT: 3FF0/0000 NOPROTECT: 3FF0/3FF0
  ID is at 2000
  Par Device value: 0084
  C Device value: 84, C-Scratch at: 0C

- **CHIPEDIT**

  ChipEdit es una utilidad de Windows (sólo para PCW) que permite editar la base de datos de un dispositivo. Con esta utilidad podemos agregar dispositivos, modificarlos o eliminarlos de la base de datos. Para agregar un dispositivo, seleccionar de la lista otro equivalente, de caracteriticas similares, y pulsar el botón ADD. Para editar o borrar un dispositivo, seleccionarlo y pulsar el botón EDIT o DELETE.

- **CONVERT**

  PConvert es una utilidad de Windows (PCW sólo) que permite realizar conversiones de un tipo de datos a otros tipos. Por ejemplo, de decimal en Punto Flotante a Hexadecimal de 4 byte. La utilidad abre una ventana pequeña para realizar las conversiones y puede permanecer activa durante una sesión con PCW o con MPLAB. Esto puede ser útil durante el proceso de depuración de un programa.

3. OPERADORES Y EXPRESIONES

- **Operadores de asignación**

  Una expresión de asignación tradicional es de la forma

  expr1 = *expr1 operador expr2* , es decir, I = I + 5. Esta expresión se puede representar por otra forma más corta: *expr1 operador*= *expr2* siguiendo con el mismo ejemplo I += 5.

  Es en las expresiones complejas, y no en una tan simple como la del ejemplo, donde se puede apreciar la conveniencia de usar esta notación. La siguiente tabla resúme los operadores de asignación compuesta y su significado.

Operador	Descripción
+=	Asignación de suma
-=	Asignación de resta
*=	Asignación de multiplicación
/=	Asignación de división
%=	Asignación de resto de división
<<=	Asignación de desplazamiento a la izquierda
>>=	Asignación de desplazamiento a la derecha
&=	Asignación de AND de bits
|=	Asignación de OR de bits
^=	Asignación de OR exclusivo de bits
~=	Asignación de negación de bits

- **Operadores aritméticos**

  Los operadores aritméticos se usan para realizar operaciones matemáticas. Se listan en la siguiente tabla:

Operador	Descripción	Ejemplo
+	Suma (enteros o reales)	resul = var1 + var2
-	Resta (enteros o reales)	resul = var1 - var2
*	Multiplicación (enteros o reales)	resul = var1 * var2
/	División (enteros o reales)	resul = var1 / var2
-	Cambio de signo en enteros o reales	-var1
%	Módulo; resto de una división entera	rango = n [A1]% 256

- **Operadores relacionales**

  Su misión es comparar dos operandos y dar un resultado entero:
  1 (verdadero); 0 (falso).
  La siguiente tabla ilustra estos operadores:

Operador	Descripción
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual a
!=	Distinto de

- **Operadores lógicos**

  Al igual que los operadores relacionales, éstos devuelven 1 (verdadero), 0 (falso) tras la evaluación de sus operandos. La tabla siguiente ilustra estos operadores.

Operador	Descripción
!	NO lógico
&&	Y lógico
||	O lógico

- **Operadores de manejo de bits**

  Estos operadores permiten actuar sobre los operandos a nivel de bits y sólo pueden ser de tipo entero (incluyendo el tipo char). Son los que siguen:

Operador	Descripción
~	Negación de bits (complemento a 1)
&	Y de bits (AND)
^	O exclusivo de bits (EXOR)
|	O de bits (OR)

- **Operadores de incremento y decremento**

Aunque estos operadores forman parte del grupo de operadores de asignación, he preferido separarlos en aras a una mayor claridad. Su comportamiento se asemeja a las instrucciones de incremento Incf f,d del ensamblador del µcontrolador PIC 16x84 o Inc variable del Intel 8051.

Operador	Descripción
++	Incremento
--	Decremento

- **Operadores de desplazamiento de bits**

Los operadores de desplazamiento otorgan al C capacidad de control a bajo nivel similar al lenguaje ensamblador. Estos operadores utilizan dos operandos enteros (tipo Int): el primero es el elemento a desplazar y el segundo, el número de posiciones de bits que se desplaza. Se resúmen en la siguiente tabla:

Operador	Descripción
>>	Desplazamiento a la derecha
<<	Desplazamiento a la izquierda

- **Operadores de dirección (&) e indirección (*)**

Los operadores & y * se utilizan para trabajar con punteros (véase tema 11). El lenguaje C está muy influenciado por el uso de punteros. Un puntero es una variable que contiene la dirección de una variable o de una función, es decir, es una variable que apunta a otra variable. Los punteros permiten la manipulación indirecta de datos y códigos. Disponemos de dos operadores; véase la siguiente tabla:

Operador	Descripción
&	De dirección
*	De indirección

El operador de dirección &, nos da la dirección de memoria de su operando. El resultado es un puntero al objeto, esto es, a un tipo de datos. Por ejemplo, si queremos guardar en el puntero p la dirección de memoria de la variable entera contador, debemos hacer lo siguiente:

```
p = &contador;          /* p apunta a la dirección de contador */
```

El operador de indirección *, nos da el valor o contenido de la variable cuya dirección está apuntada por el puntero.

```
p = &contador;          /* p apunta a la dirección de contador */
a = *p;                 /* guarda en a el contenido de la var. apuntada por p */
```

- **Expresiones**

Constantes	
123	Decimal
0123	Octal
0x123	Hex
0b010010	Binario
'x'	Carácter
'\010'	Carácter octal
'\x'	Carácter especial; x puede ser: ¿n,t,b,r,f, ", \d,v?
"abcdef"	Cadena (el carácter nulo se agrega al final)

Identificadores	
ABCDE	Hasta 32 caracteres (no puede empezar con números)
ID[X]	Un subíndice
ID[X][X]	Múltiples subíndices
ID.ID	Referencia a una estructura o una unión
ID-->ID	Referencia a una estructura o una unión

Expresiones en orden descendente de precedencia					
(expr)					
!expr	~expr	++expr	expr++	-expr	expr-
(type)expr	*expr	&value	sizeof(type)		
expr*expr	expr/expr	expr%expr			
expr+expr	expr-expr				
expr<<expr	expr>>expr				
expr<expr	expr<=expr	expr>expr	expr>=expr		
expr==expr	expr!=expr				
expr&expr					
expr^expr					
expr | expr					
expr&& expr					
expr || expr					
!value ? expr: expr					
value=expr	value+=expr	value-=expr			
value*=expr	value/=expr	value%=expr			
value>>=expr	value<<=expr	value&=expr			
value^=expr	value|=expr	expr,expr			

4. DIRECTIVAS DEL PREPROCESADOR

Todas las directivas del pre-procesador comienzan con el caracter # seguido por un comando específico. Algunas de estas directivas son extensiones del C estandar. El C proporciona una directiva del preprocesador, que los compiladores aceptan, y que permite ignorar o actuar sobre los datos que siguen. Nuestro compilador admite cualquier directiva del pre-procesador que comienze con *PRAGMA*, lo que nos asegura la compatibilidad con otros compiladores.

Ejemplo:

```
#INLINE                                  /* Estas dos lineas son válidas */
#PRAGMA INLINE
```

A continuación se describen todas y cada una de las directivas del compilador que utilizaremos para programar los microcontroladores PIC.

4.1 CONTROL DE MEMORIA

- **#ASM**
  #ENDASM

Las lineas entre *#ASM* y *#ENDASM* se tratan como código ensamblador. La variable predefinida _RETURN_ puede utilizarse para asignar un valor de retorno a la función desde el código en ensamblador. Tener presente que cualquier código C después de ^/^B #ENDASM ^I^b y antes del final de la función puede falsear el valor de retorno. La sintaxis de los comandos en ensamblador se describen en la tabla 1.

Ejemplo:

```
int paridad (int dato)  {
int contador;

#asm

        moviw  8
        movwf  contador
        moviw  0
        lazo:
        xorwf  dato,w
        rrf    dato,f
        decfsz contador,f
        goto   lazo
        movwf  _return_
 #endasm
}
```

Resumen de las instrucciones en ensamblador	
ADDWF f,d	SUBWF f,d
ANDWF f,d	SWAPF f,d
CLRF f	XORWF f,d
CLRW	BCF f,b
COMF f,d	BSF f,b
DECF f,d	BTFSC f,b
DECFSZ f,d	BTFSS f,b
INCF f,d	ANDLW k
INCFSZ f,d	CALL k
IORWF f,d	CLRWDT
MOVF f,d	GOTO k
MOVPHW	IORLW k
MOVPLW	MOVLW k
MOVWF f	RETLW k
NOP	SLEEP
RLF f,d	XORLW
RRF f,d	OPTION
TRIS k	
Solo PCM	
ADDLW k	RETFIE
SUBLW k	RETURN

- **Nota:**
  - **f** ⇒ Operando fuente;puede ser una constante, un registro o una variable
  - **d** ⇒ Operando destino;puede ser una constante (0 o 1) y también W o F
  - **b** ⇒ Campo que referencia la posición de un bit dentro de un registro
  - **k** ⇒ Operando inmediato o literal; puede ser una expresión constante

- *#BIT identificador = x.y*

Esta directiva creará un identificador "id" que puede utilizarse como cualquier SHORT INT (entero corto; un bit). El identificador referenciará un objeto en la posición de memoria x más el bit de desplazamiento y.

Ejemplos:

```
#bit tiempo = 3.4
int resultado;
#bit resultado_impar = resultado.0
...
if (resultado_ impar)
...
```

- ***#BYTE Identificador = X***

Esta directiva creará un identificador "Id" que puede utilizarse como cualquier NT (un byte). El identificador referenciará un objeto en la posición de memoria x, donde x puede ser una constante u otro identificador. Si x es otro identificador, entonces éste estará localizado en la misma dirección que el identificador "Id".

Ejemplos:

```
#byte  status = 3
#byte  port_b = 6
struct  {
short int r_w;
short int c_d;
int no_usado : 2;
int dato     : 4; } port_a;
#byte port_a = 6
...

port_a.c_d = 1;
```

- ***#RESERVE***

Permite reservar posiciones de la RAM para uso del compilador.

#RESERVE debe aparecer después de la directiva #DEVICE, de lo contrario no tendrá efecto.

Ejemplo:

```
#RESERVE  0x7d, 0x7e, 0x7f
```

- ***#ROM***

Esta directiva permite insertar datos en el archivo .HEX. En particular, se puede usar para programar la EEPROM de datos de la serie 84 de PIC.

Ejemplo:

```
#rom  0x2100={1,2,3,4,5,6,7,8}
```

- ***#ZERO_RAM***

Directiva que pone a cero todos los registros internos que pueden usarse para mantener variables, antes de que comience la ejecución del programa.

Y si en lugar de salidas son entradas ¿cómo se pone: #USE FIXED_IO (puerto_INPUTS=pin_x#, pin_x#...) ¿Puede ser PSP???

4.2 CONTROL DEL COMPILADOR

- *#CASE*

Hace que el compilador diferencie entre mayúsculas y minúsculas. Por defecto el compilador hace esta distinción.

- *#OPT n*

Esta directiva sólo se usa con el paquete PCW y, establece el nivel de optimización. Se aplica al programa entero y puede aparecer en cualquier parte del archivo fuente. El nivel de optimización 5 es el nivel para los compiladores DOS. El valor por defecto para el compilador PCW es 9 que proporciona una optimización total.

- *#PRIORITY*

Esta directiva se usa para establecer la prioridad de las interrupciones.
Los elementos de mayor prioridad van primero.

Ejemplo:

```
#priority rtcc,rb          /* la interrupción rtcc ahora tiene mayor prioridad */
```

5.9 FUNCIONES PARA EL LCD

- ***LCD_LOAD(buffer_pointer, offset, length);***

Carga *length* bytes del buffer_pointer en el segmento del área de datos del LCD, comenzando en el offset cuyo valor puede ser (0, ... 15).

- ***LCD_SYMBOL(symbol, b7_addr, b6_addr, b5_addr, b4_addr, b3_addr,***

b2_addr, b1_addr, b0_addr);

Carga 8 bits en el segmento del área de datos del LCD, con cada dirección del bit especificado. Si el bit 7 de symbol está a '1' el segmento en B7_addr se pone a '1', en otro caso se pone a '0'.

Esto es igualmente cierto para todos los otros bits de symbol. Nótese que B7_addr es un bit de dirección de la RAM del LCD.

Ejemplo:

```
byte CONST DIGIT_MAP[10]={0X90,0XB7,0X19,0X36,0X54,0X50,0XB6,0X24};

#define DIGIT_1_CONFIG
COM0+2, COM0+4, COM0+6, COM2+4, COM2+1, COM1+4, COM1+5

for(i = 1;i <= 9; ++i){
        LCD_SYMBOL(DIGIT_MAP[i],
        DIGIT_1_CONFIG);
        delay_ms(1000);
}
```

- ***SETUP_LCD(modo,prescale,segments);***

Esta función se usa para inicializar al controlador 923/924 del LCD, donde *mode* puede ser:

- LCD_DISABLED
- LCD_STATIC
- LCD_MUX12
- LCD_MUX13
- LCD_MUX14

y puede ser calificado por:

- STOP_ON_SLEEP
- USE_TIMER_1

Además, *prescale* puede valer entre 0 y 15; *segments* pueden ser cualquiera de los siguiente:

- SEG0_4
- SEG5_8
- SEG9_11
- SEG12_15
- SEG16_19
- SEG0_28
- SEG29_31
- ALL_LCD_PINS

5.10 FUNCIONES DEL C ESTÁNDAR

- *f=ABS(x)*

Calcula el valor absoluto de un entero. Si el resultado no se puede representar, el comportamiento es impreciso. El prototipo de esta función está en el fichero de cabecera stdlib.h

- *f=ACOS(x)*

Calcula el valor del arco coseno del número real x. El valor de retorno está en el rango [0,pi]radianes. Si el argumento no está dentro del rango [-1,1] el comportamiento es impreciso. El prototipo de esta función está en el fichero de cabecera math.h

- *f=ASIN(x)*

Obtiene el valor del arco seno del número real x. El valor de retorno está en el rango [-pi/2, pi/2]radianes. Si el argumento no está dentro del rango [-1,1] el comportamiento es impreciso. El prototipo de esta función está en el fichero de cabecera math.h

- *f=ATAN(x)*

Calcula el valor del arco tangente del número real x. El valor de retorno está en el rango [-pi/2, pi/2]radianes. El prototipo de esta función está en el fichero de cabecera math.h

- *i=ATOI(char *ptr)*

Esta función convierte la cadena de caracteres apuntada por ptr en un valor de tipo entero. Acepta argumentos en decimal y en hexadecimal. Si el resultado no se puede representar, el comportamiento es indeterminado. El prototipo de esta función está en el fichero de cabecera stdlib.h

- *l=ATOL(char *ptr)*

Esta función convierte la cadena de caracteres apuntada por ptr en un número entero largo (long). Acepta argumentos en decimal y en hexadecimal. Si el resultado no se puede representar, el comportamiento es indeterminado. El prototipo de esta función está en el fichero de cabecera stdlib.h

- *f=CEIL(x)*

Obtiene el valor entero más pequeño, mayor que el número real x, es decir, hace un redondeo por exceso del número real x. El prototipo de esta función está en el fichero de cabecera math.h

Ejemplo:

```
float x = 2.671;
num = ceil(x)              // num = 3
```

- *f=EXP(x) ^b^i*

Calcula la función exponencial del número real x. Si la magnitud de x es demasiado grande, el comportamiento es impreciso. El prototipo de esta función está en el fichero de cabecera math.h

Ejemplo:

```
float v1, pi = 3.1416;
v1 = exp(pi);
```

- *f=FLOOR(x)*

Calcula el valor entero más grande, menor que el número real x, es decir, hace un redondeo por defecto del número real x. El prototipo de esta función está en el fichero de cabecera math.h.

Ejemplo:

```
float x = 3.671;
num = floor(x)          // num = 3
```

- *c = ISALNUM(char)*
  c = ISALPHA(char)
  c = ISDIGIT(char)
  c = ISLOWER(char)
  c = ISSPACE(char)
  c = ISUPPER(char)
  c = ISXDIGIT(char)

Todas estas funciones manejan cadenas de caracteres y sus prototipos están en el fichero de cabecera ctype.h. Este fichero contiene las siguientes macros:

Cada función devuelve un valor distinto de cero si:

ISALNUM(X)	X es 0..9, 'A'..'Z', o 'a'..'z'
ISALPHA(X)	X es 'A'..'Z' o 'a'..'z'
ISDIGIT(X)	X es '0'..'9'
ISLOWER(X)	X es 'a'..'z'
ISUPPER(X)	X es 'A'..'Z'
ISSPACE(X)	X es un espacio
ISXDIGIT(X)	X es '0'..'9', 'A'..'F', o 'a'..'f'

- *LABS(x)*

Obtiene el valor absoluto del entero largo x. Si el resultado no puede representarse, el comportamiento es indefinido. El prototipo de esta función está en el fichero de cabecera stdlib.h

- *LOG(x)*

Calcula el logaritmo natural del número real x. Si el argumento es menor o igual que cero o demasiado grande, el comportamiento es impreciso. El prototipo de esta función está en el fichero de cabecera math.h

- *LOG10(x)*

Calcula el logaritmo decimal o base-diez del número real x. Si el argumento es menor o igual que cero o demasiado grande, el comportamiento es impreciso. El prototipo de esta función está en el fichero de cabecera math.h

- *MEMCPY(dest, source, n)*

Esta función copia ^B^I n ^b^I bytes desde source a dest en la memoria RAM.
Tanto *dest* como *source* deben ser punteros.

Ejemplo:

memcpy(&structA, &structB, sizeof (structA));

memcpy(arrayA, arrayB, sizeof (arrayA));

memcpy(&structA, &databyte, 1);

- *MEMSET(dest, value, n) ^b^I*

Esta función pone *n* bytes de memoria a partir de *dest* con el valor value.
dest debe ser un puntero.

Ejemplo:

memset(arrayA, 0, sizeof(arrayA));

memset(%structA, 0xff,sizeof(structA));

- *SQRT(x)*

Obtiene la raiz cuadrada del número real x. Si el argumento es negativo, el comportamiento es indeterminado.

6. DEFINICIÓN DE DATOS

Si TYPEDEF se pone delante de la definición de un dato, entonces no se asigna espacio de memoria al identificador a menos que sea utilizado como un especificador de tipo en otras definiciones de datos.

Si delante del identificador ponemos *CONST* entonces, el identificador es tratado como constante. Las constantes deben ser inicializadas y no pueden cambiar en tiempo de ejecución.

No están permitidos punteros a constantes. *SHORT* es un tipo especial utilizado para generar código muy eficiente para las *operaciones de I/O*.

No se permiten las arrays de *SHORT* ni los punteros a *SHORT*. La siguiente tabla muestra la sintaxis para las definiciones de datos.

Ejemplos:

```
int a,b,c,d;
typedef int byte;
typedef short bit;

bit e,f;
byte g[3][2];
char *h;
enum boolean  {false, true};
boolean j;
byte k = 5;
byte const SEMANAS = 52;
byte const FACTORES [4] = {8, 16, 64, 128};

struct registro_datos {
        byte   a [2];
        byte  b : 2;  /*2 bits */
        byte  c : 3;  /*3  bits*/
        int d;
}
```

DEFINICIONES DE DATOS

typedef	[calificador_tipo] [especificador_tipo] [identificador]
static	Variable global e inicializada a 0
auto	La variable existe mientras el procedimiento está activo Es el valor por defecto, por eso no es necesario poner auto

Especificadores de tipo:

unsigned	define un número de 8 bits sin signo
unsigned int	define un número de 8 bits sin signo
int	define un número de 8 bits sin signo
char	define un número de 8 bits sin signo
long	define un número de 16 bits sin signo
long int	define un número de 16 bits sin signo
signed	define un número de 8 bits con signo
signed int	define un número de 8 bits con signo
signed long	define un número de 16 bits con signo
float	define un número de 32 bits en punto flotante
short	define un bit
short int	define un bit

identificador	identificador de una definición TYPE de tipo
Enum	tipo enumerado, véase sintaxis a continuación
Struct	estructura, véase sintaxis a continuación
Unión	unión, véase sintaxis a continuación

declarador:

[const] [*]identificador [expr._constante][= valor_inicial]

enumerador:

```
enum [identificador]{
[lista_identificadores[= expresión_constante]]

}
```

estructura y unión:

```
struct [identificador] {
    [calificador_tipo [[*]]identificador
    :expresión_constante [expresión_constante]]]
  }

union  ídem
```

Ejemplo:

```
struct lcd_pin_map {
        boolean enable;
        boolean rs;
        boolean rw;
        boolean unused;
        int     data : 4;
      } lcd;
```

Si ponemos *expresión_constante* después de un *identificador*, determina el número de bits que utilizará dicho identificador. Este número puede ser 1,2, ...8. Se pueden utilizar arrays de dimensión múltiple poniendo los [] que se precisen. También podemos anidar estructuras y uniones.

Un identificador después de **struct** puede usarse en otra **struct** y las {} no se pueden usar para reutilizar la estructura de la misma forma otra vez.

Nota: Recordar que todo lo expresado entre [corchetes]es opcional, es decir, se puede especificar o no, según convenga. Si la expresión va entre {llaves} entonces indica que debe aparecer *una o más veces* y en ese caso separados por comas; véase más abajo el siguiente ejemplo.

El identificador después de **enum** es un tipo de datos tan grande como la mayor de las constantes de la lista. Cada uno de los identificadores de la lista son creados como una constante. Por defecto, el primer identificador se pone a cero y los siguientes se incrementan en uno. Si incluimos "=*expresión_constante*" después de un identificador éste tendrá el valor de la *expresión_constante* y los siguientes se incrementarán en uno.

Ejemplo:

```
enum ADC_SOURCE {an0, an1, an2, an3, bandgap, srefh, srefl, itemp, refa, refb};
```

7. DEFINICIÓN DE FUNCIÓN

El formato de la definición de una función es como sigue:

[calificador_tipo] identificador ([[especificador_tipo identificador]]) {

[cuerpo de la función]

}

El *calificador_tipo* para una función pueden ser:

void o un
especificador de tipo (véase la lista de la página anterior)

La definición de una función puede ir precedida por una de las siguientes directivas del preprocesador (*calificadores de función*) para identificar una característica especial de la función: *#separate #inline #int_...*

Cuando utilizamos una de las directivas mencionadas y la función tiene un prototipo (declaración anterior a la definición de la función, y colocada al principio del fichero fuente) hay que incluir la misma *#directiva* en el *prototipo* y en la *definición de la función*.

Una característica no muy corriente, se ha añadido al compilador para ayudar a evitar los problemas creados por el hecho de que no pueden crearse punteros a constantes de cadenas. Una función que tiene un parámetro de tipo char aceptará una constante de cadena. El compilador generará un bucle que llama a la función una vez para cada carácter de la cadena.

Ejemplo:

```
void lcd_putc(char c ) {
// definición de la función
...
}
lcd_putc ("Máquina parada".);
```

UNIVERSITAS MAJOR PACENSIS DIVI ANDREÆ