APRENDER FACIL LENGUAJE VISUAL DE MICROCONTROLADORES PIC

Crea, diseña y simula tus proyectos con Niple y Proteus

ING. RICARDO ALCIDES CANAVERI

Primera edición: diciembre de 2021

Copyright © 2021 Ing. Ricardo Alcides Canaveri
Publicado por Amazon
Todos los derechos reservados. Bajo las sanciones establecidas en el ordenamiento jurídico, queda rigurosamente prohibida, sin autorización escrita de los titulares del *copyright*, la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático.

ISBN 9798458226493

safecreative 2108238847041

safecreative 2112059968298

DEDICATORIA

Dedico este libro a mi familia.

A mi esposa, quien ha estado a mi lado durante todo este tiempo que he trabajado en esta obra.

A mis amigos, quienes me han apoyado y a todos los que me han prestado ayuda, correctores de estilo, a todos ellos dedico este libro con cariño y un muy grande agradecimiento.

Ricardo Alcides Canaveri

Nació el 28 de septiembre de 1975 en Paraná (Argentina), cursó sus estudios secundarios en el Centro Provincial de Enseñanza Media (CPEM) Nº 14 (Aluminé, Provincia del Neuquén). En la actualidad se desempeña como Oficial jefe de Gendarmería Nacional Argentina, desde hace 28 años dentro de la Institución Nacional. Participado de grandes proyectos de implementación tecnológica e invención en la Seguridad Publica y Seguridad Ciudadana, tales como la FAT (Pruebas de Aceptación en Fabrica) en Motorola Chicago Illinois (EEUU) y Teltronic S.A.U Zaragoza (España).

Ejerce la docencia en la Fuerza desde año 2008, formando Oficiales y Suboficiales en la parte técnica-profesional conjuntamente con la Universidad Católica de Salta (2007).

Participó en una misión al exterior (2014-2015). Desempeñando el rol de Jefe del Elemento Seguridad de la Embajada Argentina en la Republica de Haití.

Es Ingeniero Electrónico (2007), Especialista en Sistema de Control (2013) de la Facultad de Ingeniería del Ejército Argentino (FIE) y Diplomado en Estado Mayor de Gendarmería Nacional (2017).

Enlaces de contacto con Ricardo Alcides Canaveri

YouTube	
	amazon

APRENDER FÁCIL

LENGUAJE VISUAL DE MICROCONTROLADORES PIC

Crea, diseña y simula tus proyectos con Niple y Proteus

INGENIERO RICARDO ALCIDES CANAVERI

Agradecimientos:

A mi madre ***Nilda Dominga Grigolatto*** y mi padre ***Ricardo Alcides Canaveri***, por todo su amor, cariño y enseñanzas de vida, mostrándome siempre el camino a seguir., para poder ser hoy una persona profesional.

A mi compañera de toda la vida, mi esposa ***Lidia Elizabeth Marín***, por estar siempre presente en todos mis emprendimientos, y enseñarme las cosas que a simple vista no se observan.

A ***Carlos Manzano*** por acompáñame en mis primeros años de ingeniería e invitarme a conocer la P*rogramación de Microcontroladores PIC* con Niple.

A ***Jorge Cano*** por su apoyo incondicional, predisposición y colaboración para concretar este libro.

...Muchas gracias por todo

Prefacio

Este libro está dirigido a los/las estudiantes que se inician en el área de la electrónica, la informática y las telecomunicaciones; además, a los técnicos y a los profesionales que trabajan en tareas afines, y a los/las aficionados a la electrónica.

Los ejemplos descritos están enfocados en utilizar el lenguaje visual de los microcontroladores PIC "Niple" para que esos algoritmos programados sean trasladados a la herramienta Suite de Proteus, donde podrá examinar, modificar y optimizar los códigos compilados. Esta solución simplifica la tarea de los programadores, evitando que retiren e introduzcan el microcontrolador para regrabarlo.

En el ámbito educativo, es una herramienta muy práctica, al alcance de los estudiantes de Ingeniería.

El Ingeniero Ricardo Alcides Canaveri domina el software desde hace 13 años. Cuenta con la experticia necesaria sobre las distintas versiones que adquirió, prestando especial atención a lo novedoso. Colaboró ampliamente con el equipo del Ingeniero Jorge Cano (Desarrollador de NipleSoft) en la Provincia de Entre Ríos, Argentina.

Como Ingeniero Electrónico, desplegó sus conocimientos en el área de Investigación y Desarrollo. En Gendarmería Nacional Argentina, produjo los dispositivos de iluminación y marquesinas de Led de ultra brillo. Mejoró un sector importante vinculado a la luminaria.

Sus saberes académicos, enmarcados en los adelantos tecnológicos necesarios para el avance y progreso de su Institución de pertenencia, fueron reconocidos.

Ha formado y capacitado a alumnos en el área técnico-profesional con incumbencias en las telecomunicaciones. Incentivó el desarrollo y la inventiva a través de Niple. Junto con los estudiantes, llevó a cabo varios

proyectos en la Escuela de Suboficiales de Gendarmería Nacional "Cabo Raúl Remberto Cuello", sita en la Ciudad de Jesús María, Provincia de Córdoba, Argentina.

En esta edición pretende capacitar a los lectores, para que den los primeros pasos en la programación. Mediante el conjunto de herramientas ofrecidas a los futuros aprendientes, que estos puedan asimilar y desarrollar nuevos aprendizajes con total normalidad y utilizarlos en los sectores que se necesiten o que desee abordar como un desafío personal y/o profesional.

El Ingeniero Canaveri es un sobresaliente profesional, y mejor persona. Ha demostrado sus capacidades con creces. Sus deseos de superación y de ayudar a los demás son evidentes. Es por su propósito y diseño didáctico, que recomiendo este libro.

Carlos Guillermo Blanco
Máster en Ciencias en Liderazgo y Gestión Policial

Introducción

Bienvenidos a la primera edición de *Aprender Fácil Lenguaje Visual de Microcontroladores PIC*. En la actualidad, el avance de la tecnología ha modernizado e incorporado una mayor cantidad de dispositivos electrónicos que dan respuesta o soluciones al quehacer diario de las personas y mejoran sus vidas. La fabricación de nuevos microcontroladores potenciados con una mayor cantidad de pines, memoria de trabajo y demás cuestiones, y la creación de nuevos esquemas para la programación de estos chips, resulta en que la mayoría de los casos sean complejos o requieran mayor tiempo y una capacitación académica adecuada.

El Ingeniero Ricardo Alcides Canaveri le enseñará cómo resolver esta problemática. Tiene en cuenta la importancia de la programación de microcontroladores con vistas a incorporar los conocimientos básicos de estos chips, y programar sus propios algoritmos en "Niple" con facilidad y rapidez. De modo que pueda simularlos, corregirlos y optimizarlos en el laboratorio virtual de diseño electrónico "Proteus", para que pueda alcanzar sencillamente el completo dominio de los programas Niple y Proteus. Pretende, de esa manera, que desarrolle sus proyectos en cualquier ámbito, ya sea laboral, académico o profesional.

Para lograr un aprendizaje significativo, ha puesto especial atención en que cada tema esté explicado de forma clara, didáctica y asequible. Todos los capítulos del libro cuentan con un diseño visual muy cuidado. Gracias a ello y al carácter práctico de los distintos temas, podrá incorporar todos los conceptos con asombrosa facilidad.

Para conocer y comprender este mundo fascinante de la programación visual de Microcontroladores PIC, describe un método para que pueda familiarizarse instantáneamente con el lenguaje visual de los microcontroladores, acompañado de los conocimientos prácticos que aprenderá sobre el uso del laboratorio virtual que ha de utilizar.

Este libro está divido en siete capítulos, para obtener una mejor organización de los contenidos, a fin de obtener los conocimientos y aprender el manejo de los microcontroladores PIC, su lenguaje visual y los esquemas en el laboratorio virtual para alcanzar resultados más eficientemente.

Lo que tienes que saber acerca de este libro

Como mencione en la introducción, se encuentra compuesto por siete capítulos. Iré mostrando de manera progresiva los ítems importantes, que se debe conocer para aprender fácil el Lenguaje Visual de Microcontroladores PIC. De este modo, se tendrá que realizar un camino, y seguir el paso a paso, para que nadie quede con dudas, para avanzar a la siguiente etapa.

De esta forma, pensando en cada lector/aprendiz, les comparto un ayudante que se llama "*Micrito*", este aparecerá para destacar lo relevante a saber.

Es importante enfatizar, que se extrajeron partes o fragmentos más destacados del Manual de Niple versión 5.6.3, el cual posee una antigüedad de 8 años y fue escrito por Jorge Cano (Creador y desarrollador de Niple), los cuales fueron actualizados en esta edición para plasmar nuevas gráficas y diseños visuales más didácticos para el lector.

Esta edición se encuentra compuesta en varios segmentos, diseñados para el lector:

Segmento 1: Los contenidos teóricos necesarios volcados en la presente edición, son recopilación de información necesaria para abordar los contenidos que deseo transmitir.

Segmento 2: Los ejemplos prácticos empleados, harán de guía para la resolución práctica de los contenidos teóricos arribados. Con ellos, se abordarán la practicidad en torno a un texto o contenido a conocer.

Segmento 3: El software que incluye esta edición, es el motor principal que se tendrá durante el desarrollo de cada uno de los ejemplos que plantea en cada capítulo. Y que a través de Niple 6.5.0 (versión para el libro), en conjunto, con la Suite de Proteus 8.11 se podrá experimentar las simulaciones realizadas, de este modo, se puede desarrollar proyectos ilimitados.

Segmento 4: Incluye la descarga de todos los ejemplos explicados en el libro, realizados por Niple y Proteus.

Segmento 5: El total del contenido o material adicional, incluido en esta edición, ayudará al lector/aprendiz que no posee los conocimientos básicos para poder avanzar con los capítulos. De esta forma, pensando en el lector autodidacta, incluyéndolo en los contenidos que aspiro enseñar.

INDICE

CAPÍTULO 1 1
El inicio de la programación 1
Algoritmo 4
Identificadores 5
Constante 5
Variables 5
Tipos de variables 6
Diagrama de Flujo 7
CAPÍTULO 2 15
El Microcontrolador PIC 15
¿Qué es un microcontrolador? 15
Los microcontroladores "PIC" 16
Estructura de un microcontrolador 17
El procesador o CPU 17
Memoria ROM 18
Memoria RAM 20
Registros y Bits 21
Líneas de Entrada/Salida (E/S), (Puertos) 27
Módulos Temporizadores Internos (TMRs) 28
Utilizando un microcontrolador 30
La Programación Visual de microcontroladores PIC con Niple 33
Capítulo 3 45
Esquemáticos, Simulación y Placas Electrónicas con PROTEUS 45
Proteus VSM 46
Captura de esquemático ISIS 47
Barra de menú 50

Barras de herramientas 51
Panel de Proteus 52
Layout de ARES PCB 53
PROSPICE 53
VSM 53
Programa con microprocesadores 53
Instalación de Proteus 8.13 54
Electrónica Analógica 59
Electrónica Digital 60
Compuertas Lógicas 62
Teorema de algebra booleana 62
Axiomas de algebra Boole. 63
Ares Professional Layout Design. 71
Capítulo 4 87
Metodología del lenguaje visual con Niple 87
Instalación de Niple 88
Instalación Licencia de Niple 90
Instalando la Licencia de Niple 91
Configuración del Sistema 93
Metodología de Programación Visual con Niple 95
Instrucciones: 96
Interrupciones: 98
Subrutinas de usuario: 98
Secciones de trabajo 99
Entornos 103
Crear un nuevo Entorno 104
Seleccionar el Entorno actual de trabajo 105

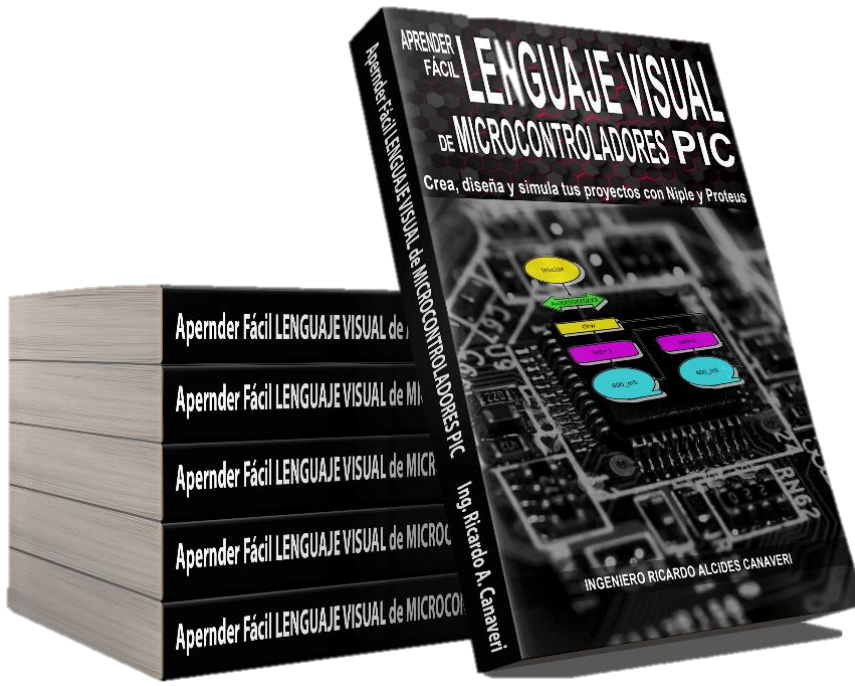
Borrar un Entorno 105
Bloques 105
Definición de cada Bloque 105
Selección de Bloques o Vínculos 108
Mover un Bloque 109
Estirar un Bloque 110
Borrar Bloques 111
Editar Bloques 111
Duplicar Bloques 113
Habilitar e Inhabilitar Bloques 114
Vínculos (Flechas) 116
Crear un Vínculo 118
Cambiar el estilo de un Vínculo 118
Borrar Vínculos 119
Ingreso de Datos 119
Control de ingreso de nombres de registros 120
Control de ingreso de valores Literales 121
Modos de Trabajo "Nivel de Usuario" 122
Rutinas de Usuario 123
Exportar Rutina de Usuario 123
Importar una Rutina de Usuario 124
Exportar Imagen 125
Declaración de Registros 127
Borrar un Registro 131
Puertos 132
Leer Puerto 132
Escribir Puerto 133

Conversión Analógica/Digital (A/D) 134
Configuración de los canales A/D 134
Conversión A/D 135
EEPROM 142
Declaración de Registros EEPROM 142
Leer EEPROM 143
Grabar EEPROM 143
Condición 144
Comparar Bit 144
Comparar Registro (8 Bits) 146
Comparar Registro (16 Bits) 147
Comparar Registros de 24 y 32 Bits 149
Matemáticas 149
Suma 149
Resta 151
División 153
Multiplicación 154
Incrementar 1 155
Decrementar 1 157
Decimal a BCD 158
Escalar 158
Capítulo 5 163
Programando con Niple 163
Mi primer programa 163
Conversión de NPL a ASM 176
Conversión de ASM a HEX 178
Armado del proyecto en ISIS 180

Capítulo 6 189
Declaración de dispositivos al Microcontrolador PIC 189
Comunicación RS232 189
Comunicación RS485 193
Modo Maestro 194
Interrogar 197
Tablas 198
Crear una nueva tabla 199
Consultar Tabla 201
Display de 7 segmentos sin Codificador 202
Declaración del Dispositivo 202
Display de 7 segmentos con Codificador 203
Declaración del Dispositivo 203
Implementación en el Diagrama de Flujo 204
Pantalla LCD 208
Configurar LCD 211
Utilizando más de una pantalla LCD 214
Generar un nuevo mensaje 216
Visualizando valores variables en un mensaje 221
Mostrar Mensaje 224
Optimización del código Assembler 225
Comandos 226
Enviar comandos 226
Enviar caracteres 228
Teclado Matricial 230
Memorias EEPROM I2C 231
Declaración del Dispositivo 231

Leer Memoria EEPROM I2C 232
Escribir Memoria EEPROM I2C 234
Memorias RAM 235
Declaración del Dispositivo 235
Leer Memoria RAM I2C 237
Escribir Memoria RAM I2C 239
Reloj de Tiempo Real RTC PCF8583 241
Declaración del Dispositivo 241
Implementando el PCF8583 en el Diagrama de Flujo. 242
Inicialización 242
Escritura 243
Configuración de Alarmas 248
Lectura 251
Utilizando reloj de tiempo real (DS1307) 251
Configuración de Pines 252
Configuración del Módulo 253
Declaración de Dispositivos 253
Inicialización 253
Escritura 254
Lectura 255
Registro Desplazamiento Ingresar valor y rotar 256
Capítulo 7 267
Proyecto. ¡Manos a la obra! 267
Proyecto: Baliza electrónica y luz de emergencia 267
Elementos constitutivos 268
Led (diodo emisor de luz) 268
Tansistores (NPN y PNP) 269

Cálculos de resistencias para los Leds rojo y blanco: 271
Programando en Niple 272
Mensaje para el Lector 292
Enlaces de contacto con Ricardo Alcides Canaveri 295


APRENDER FÁCIL
LENGUAJE VISUAL
DE MICROCONTROLADORES PIC
Crea, diseña y simula tus proyectos con Niple y Proteus
INGENIERO RICARDO ALCIDES CANAVERI
Aprender Fácil LENGUAJE VISUAL de MICROCONTROLADORES PIC
Ing. Ricardo A. Canaveri
Apernder Fácil LENGUAJE VISUAL de
Apernder Fácil LENGUAJE VISUAL de MI
Apernder Fácil LENGUAJE VISUAL de MICR
Apernder Fácil LENGUAJE VISUAL de MICROC
Apernder Fácil LENGUAJE VISUAL de MICROCO

CAPÍTULO 1
El inicio de la programación

Desde muy chico me llamó mucho la atención la electrónica y las tareas afines, entre ellas: De qué manera se reparan las cosas, mediante la observación y con pocos conocimientos comprender cómo se solucionaban los problemas en los sistemas electrónicos, tan sólo con ver a los compañeros de mi padre en el trabajo hacer esta actividad, me entusiasmaba muchísimo.

Pero las dudas siempre se fueron incrementando, dado que no estaba estudiando en una escuela secundaria técnica, y que, donde vivíamos no existían escuelas dedicadas a este campo. Acceder en aquella época a cursos de esta índole en el pueblito de Aluminé (Neuquén), el lugar donde residíamos era una misión imposible.

Con el pasar del tiempo, completé mi formación técnica en un terciario con orientación en telecomunicaciones e informática en la provincia de Buenos Aries. Luego de unos años, obtuve una beca para estudiar ingeniería electrónica, la cual me gané con el esfuerzo al estudio y dedicación de los trabajos técnicos realizados en mis primeros años de trabajo.

Esos 5 años de estudio en la universidad, fueron los mejores de mi vida, ya que pude completar mi formación de grado y recibirme de ingeniero electrónico, gracias al empeño, y dedicación de Lidia (mi esposa) y mis 3 pequeños hijos (Melani, Yamil y Maximiliano), un gran orgullo personal y familiar.

He aprendido que las cosas se construyen en la vida de la misma forma como se diseñan y fabrican las cosas electrónicas, con esfuerzo y dedicación, palabras que siempre mi padre me inculcó desde muy pequeño, entre ellas: -Primero ser buena persona ante todas las cosas.
-Y segundo, que nadie en la vida te regala nada, que sólo uno forja su destino y su trabajo.

A partir de esas premisas, puedo decirles, cual fue el sentido y comienzo de este gran camino que me llevó a comprender más. Luego de haber pasado un tiempo y haberme recibido de ingeniero, un compañero de trabajo Carlos Manzano (técnico del alma) me invitó a realizar una capacitación referida al mundo de los microcontroladores, que me interesó sobremanera y lo realizamos juntos. Allí, pude ampliar mis conocimientos, y al mismo tiempo, concretar un invento, que tenía en la mente, el que deseaba con muchas ansias poder materializarlo, un dispositivo electrónico portátil con capacidad de efectos lumínicos, que sería empleado en seguridad vial, que se podía visualizarse a gran distancia en condiciones climáticas adversas. Esto abrió en mi mente en la creatividad, además de poder ampliar mi ámbito de actuación en la creación y diseño electrónico.

En aquel curso, logré completar los conocimientos en la programación visual de microcontroladores, también fui alumno destacado, para explicarles a mis compañeros del curso, como realizar un circuito PCB en forma muy sencilla y fácil con Proteus (Laboratorio virtual de electrónica).

Con estas pequeñas líneas, les invito que me conozcan, de modo que pueda contarles cuáles fueron mis primeros pasos, aquellos que me dirigieron a la programación de microcontroladores, un mundo fascinante del cual no tenía incorporado su magnitud. No quiero descartar el momento y aprovechar mencionarles lo importante que fue esto en mi vida profesional, en mi trabajo, ya que fui el primer ingeniero más joven que volcó sus esfuerzos en introducir la programación visual de microcontroladores dentro de mis tareas (Gendarmería Nacional

Argentina), habiendo formado parte del área de desarrollo e investigación (I+D) de proyectos de interés para la fuerza de seguridad en el año 2008.

Tuve el privilegio de ser docente y capacitar alumnos introduciéndolos en este fascinante universo de la programación embebida. Con el uso del ingenio y la pedagogía pude plasmar el conocimiento de forma dinámica y fácil generando en mis alumnos un conocimiento crítico y un aprendizaje proactivo en la programación.

Luego de haber hecho una pequeña introducción, que es una parte de este capítulo. Ahora puedo preguntarte:
—¿Sí sabes programar?—seguramente me responderás que no, pero en este caso preciso, te mostraré que —¡ya sabes programar! —, pero que hasta el momento, —¡No lo sabías! —.

En esta situación, comenzaré por explicar con un sencillo ejemplo, el que permitirá comprender como es el funcionamiento de un microcontrolador, comparándolo o haciendo una analogía con el hombre.
No voy definir todavía su funcionamiento interno, sino que, me basaré más en explicarte el principio de la programación, para que puedas interpretar y comprender lo fácil que te resultará la programación visual.

El primer ejemplo, detalla las actividades que realiza una persona en un día de su vida, desde que inicia sus actividades, hasta que se finaliza el día.
Comencemos: Se despierta a las 6:30 a.m, luego realiza su aseo personal, se pregunta: —¿Desayuno en casa? —en el caso que sea "SÍ", prepara el desayuno y desayuna en su casa, en el caso contrario "NO", desayunaría en la cafetería que siempre frecuenta. Luego prepara las cosas para ir a la universidad en su vehículo. Llega y asiste a su primera clase del día. En la siguiente hora, se pregunta: —¿Hay hora libre?, en caso que sea "SÍ"; espera en el aula a la siguiente hora, en caso contrario "NO", completaría sus apuntes hasta la siguiente hora. Finalizando la última hora de clases, recibe una llamada telefónica de un amigo, quien lo invita a salir con sus amigos, aquí se pregunta: —¿Salgo con mis amigos? —en el caso que sea "SÍ", se traslada a la casa de su amigo, en el caso contrario "NO", se dirige a su domicilio para compartir con su familia. Luego de estar en su casa descansando, se prepara para cenar y comparte la cena con sus padres y hermanos. Finalizada la cena se pregunta: —¿Reviso el correo electrónico y portal de la universidad? —en caso que sea "SÍ", chequea

el correo y completa archivos para el siguiente día, en el caso contrario "NO", prepara sus cosas para irse a dormir las 10 p.m.

El relato comentado en el párrafo anterior, resume las actividades que realizamos en un día determinado de nuestra vida. Ahora bien, a este relato el que forma parte de esquemas mentales o ideas que formulamos en nuestro cerebro, lo vamos a transformar en una serie de pasos que permitan resolver el problema, a este conjunto de pasos se le denomina "*algoritmo"*, el cual debe tener como característica final la posibilidad de transcribirlo fácilmente a un lenguaje de programación, para esto se utilizan herramientas de programación, las cuales son métodos que permiten la elaboración de algoritmos escritos en un lenguaje entendible, como el que lograremos con *diagramas de flujos*.

Ustedes se preguntarán: ¿Qué es un Algoritmo? y ¿qué es un Diagrama de Flujo? permítanme que lo defina, para poder comprender el procedimiento que emplearemos en el primer ejemplo que desarrollaremos.

Algoritmo

Un algoritmo es una serie de operaciones detalladas y no ambiguas para ejecutar paso a paso que conducen a la resolución de un problema, y se representan mediante una herramienta o técnica. O bien, es una forma de describir la solución de un problema planteado en forma adecuada y de manera genérica.

Además de esto, consideraremos que el algoritmo, que posteriormente se transformará en un programa de computadora, debe tener las siguientes partes:

- Una descripción de los datos que serán manipulados (empleados).
- Una descripción de acciones que deben ser ejecutadas para manipular los datos.
- Los resultados que se obtendrán por la manipulación de los datos.

Las herramientas o técnicas de programación que más se utilizan y que se emplearán para la representación de algoritmos son por lo general, Pseudocódigo y Diagramas de flujo.

Identificadores

Antes de comenzar con el análisis cada una las herramientas que usaremos en la representación de algoritmos para la solución de problemas, se establecerán qué son los *identificadores* que se emplean dentro de un algoritmo.

Los *identificadores* son los nombres que se les asignan a los objetos, los cuales se pueden considerar como variables o constantes, éstos intervienen en los procesos que se realizan para la solución de un problema, por consiguiente, es necesario establecer qué características tienen.

Para establecer los nombres de los *identificadores* se deben respetar ciertas reglas que determinan cada uno de los lenguajes de programación, para el caso, se establecen de forma indistinta según el problema que se esté abordando, sin seguir regla alguna, generalmente se utilizará la letra, o las letras, con la que inicia el nombre de la *variable* que representa el objeto que se va a identificar.

Constante

Un identificador se clasifica como *constante* cuando el valor que se le asigna a este identificador no cambia durante la ejecución o proceso de solución del problema. Por ejemplo, en problemas donde se utiliza el valor de PI, si el lenguaje que se utiliza para codificar el programa y ejecutarlo en la computadora no lo tiene definido, entonces se puede establecer de forma constante estableciendo un identificador llamado PI "π" y asignarle el valor correspondiente de la siguiente manera:

$$PI = 3.141692.$$

De igual manera, se puede asignar valores constantes para otros identificadores según las necesidades del algoritmo que se esté diseñando.

Variables

Los identificadores de tipo *variable* son todos aquellos objetos cuyo valor

cambia durante la ejecución o proceso de solución del problema. Por ejemplo, la temperatura, el tiempo, la tensión, etcétera, que se deben calcular con un algoritmo determinado, o en su caso, contar con el largo (H) y ancho (A) de un rectángulo que servirán para calcular y obtener su área. Como se puede ver, tanto H como A son variables que se proporcionan para que el algoritmo pueda funcionar, y no necesariamente se calculen dentro del proceso de solución.

Tipos de variables

Los elementos que cambian durante la solución de un problema se denominan variables. Se clasifican dependiendo de lo que deben representar en el algoritmo, por lo cual pueden ser: de tipo *entero*, real y *string* o de cadena, sin embargo, existen otros tipos de variables que son permitidos con base en el lenguaje de programación que se utilice para crear los programas, por consiguiente, al momento de estudiar algún lenguaje de programación en especial se deben dar a conocer esas clasificaciones.

Para este caso, se denominará variables de tipo entero a todas aquellas cuyo valor no tenga valores decimales; contrario a las de tipo real, la cual podrá tomar valores con decimales. Como ejemplo de variables enteras se puede considerar el número de personas, días trabajados, edad de una persona, etcétera. Y para el caso de reales, se puede considerar la temperatura del agua, la tensión continua que se mide en una batería, etcétera.

En caso de que las variables tomen valores de caracteres, se designarán *string* o de cadena; como ejemplo de éstas se pueden mencionar el sexo de una persona, falso o verdadero, el nombre de una persona, el tipo de sangre, etcétera.

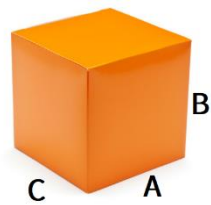
Pseudocódigo

El pseudocódigo es una de las herramientas más conocidas para el diseño de solución de problemas por computadora. Esta herramienta permite

pasar casi de manera directa la solución del problema a un lenguaje de programación específico. El pseudocódigo es una *serie de pasos* bien detallados y claros que conducen a la resolución de un problema.

La forma fácil de pasar casi de forma directa el pseudocódigo a la computadora ha dado como resultado que muchos programadores implementen de forma directa los programas en la computadora, cosa que no es muy recomendable, sobre todo cuando no se tiene la suficiente experiencia para tal aventura, pues se podrían tener errores propios de la poca experiencia acumulada con la solución de diferentes problemas.

Por ejemplo: el pseudocódigo para determinar el volumen de una caja de dimensiones A, B y C se puede establecer de la siguiente forma:



1. Inicio.
2. Leer las medidas A, B y C.
3. Realizar el producto de A * B * C y guardarlo en V (V = A * B * C).
4. Escribir el resultado V.

Fig. 1.1 pseudocódigo volumen de una caja

Como podemos ver, se estableció de forma precisa la secuencia de los pasos por realizar; además, si se les asignan siempre los mismos valores a las *variables* A, B y C, el resultado del volumen será el mismo y, en consecuencia, se obtendrá un valor final.

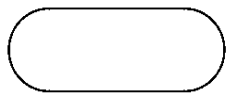
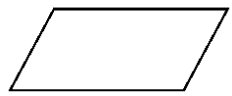
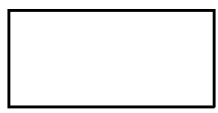
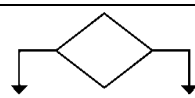
Diagrama de Flujo

El diagrama de flujo es una herramienta que permite representar en modo visual, qué operaciones se necesitan y en qué secuencia se deben establecer para solucionar un problema determinado. Por lo tanto, un diagrama de flujo es la representación gráfica mediante símbolos especiales, de los pasos o procedimientos de manera secuencial y lógica que se deben efectuar para solucionar un problema dado.

Los *diagramas de flujo* desempeñan un papel importante en la programación de un problema, ya que facilitan la comprensión de problemas complicados y sobre todo aquellos en que sus procesos son muy largos.

Los diagramas de flujo facilitan la comunicación entre los programadores y los usuarios, además de que permiten de una manera más rápida detectar los posibles errores de lógica que se presenten al implementar el algoritmo. En la tabla 1.1 se muestran algunos de los principales símbolos utilizados para construir un diagrama de flujo.

Dentro de los diagramas de flujo se pueden utilizar los símbolos que se visualizan en la tabla 1.2, con los cuales se puede identificar las operaciones que se efectuarán a los datos con el fin de producir un resultado.

SIMBOLO	SIGNIFICADO
	Inicio/Finalización
	Entrada de datos
	Proceso
	Decisión

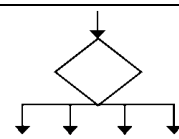
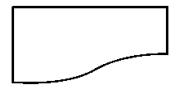
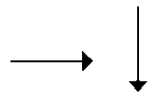
	Decisión Múltiple
	Imprimir Resultados
	Flujo de Datos
	Conectores

Tabla 1.1 Principales símbolos utilizados para construir los diagramas de flujo

SIMBOLO	OPERACION
+	Suma
-	Resta
*	Multiplicación
/	División
^	Exponenciación
>	Mayor que
<	Menor que
>=	Mayor e igual que
<=	Menor e igual que

=	Igual que
< >	Diferente que

Tabla 1.2 Principales símbolos utilizados en los diagramas de flujo para indicar las operaciones que se realizan para producir un resultado.

Podríamos establecer la solución del diagrama de flujo para determinar el volumen de una caja de dimensiones A, B y C como se muestra en la figura 1.1.

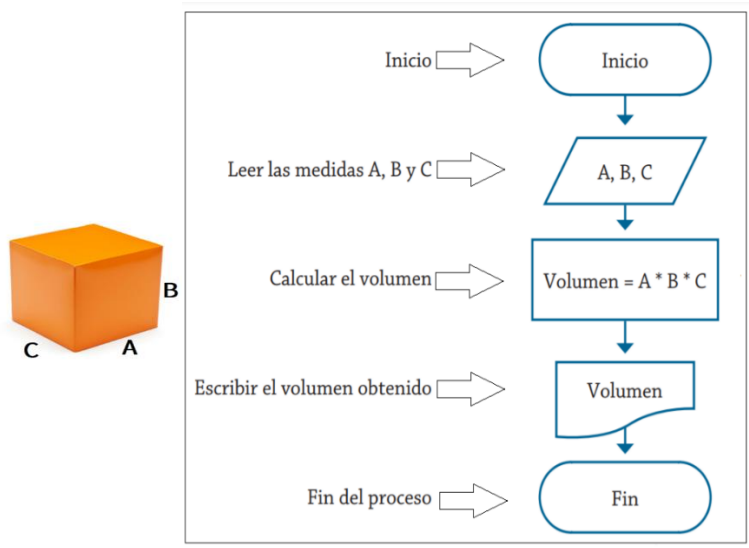


Fig. 1.2 diagrama de flujo volumen de caja

De esta forma se puede ver de manera gráfica, estableciéndose en forma precisa la secuencia de los pasos a efectuar para obtener el resultado del volumen. Por lo tanto, verificamos que, son los mismos pasos que se establecieron en el algoritmo presentado previamente mediante el pseudocódigo.

Ahora bien, pudiendo comprender estos conceptos expresados con

antelación, retomamos nuestro caso, pasando por el pseudocódigo y luego por el diagrama de flujo. Entonces ¡Manos a la obra!
Pseudocódigo propuesto:

Inicio:
1. Me despierto a las 6.30 horas.
2. Realizo mi aseo personal.
3. Pregunto ¿Desayuno en casa?
4. "SÍ". Preparo el desayuno.
5. "NO". Desayuno en la cafetería.
6. Me traslado a la facultad en mi vehículo.
7. Asisto a clases.
8. Tomo apuntes.
9.Pregunto: ¿Hora Libre?
10. "SÍ". Espero a la siguiente hora.
11. "NO". Tomar el resto de mis apuntes hasta finalizar la clase.
12. Llamada telefónica de un amigo. Invitación para juntarme en su casa.
13. Pregunto: ¿Juntarme con mis amigos?
14. "SÍ". Me traslado con mi vehículo a la reunión acordada con ellos.
15. "NO". Me traslado con mi vehículo a mi domicilio.
16. Recreación con mis hermanos en mi casa. Cenar con mi familia.
17. Pregunto: ¿Revisar correo y portal de la universidad?
18. "SÍ". Chequear correo y subir archivos para el siguiente día.
19. "NO". Prepararme para dormir.
20. Colocar alarma para las 6.30 horas
21. Dormir.
Fin

Diagrama de flujo propuesto:

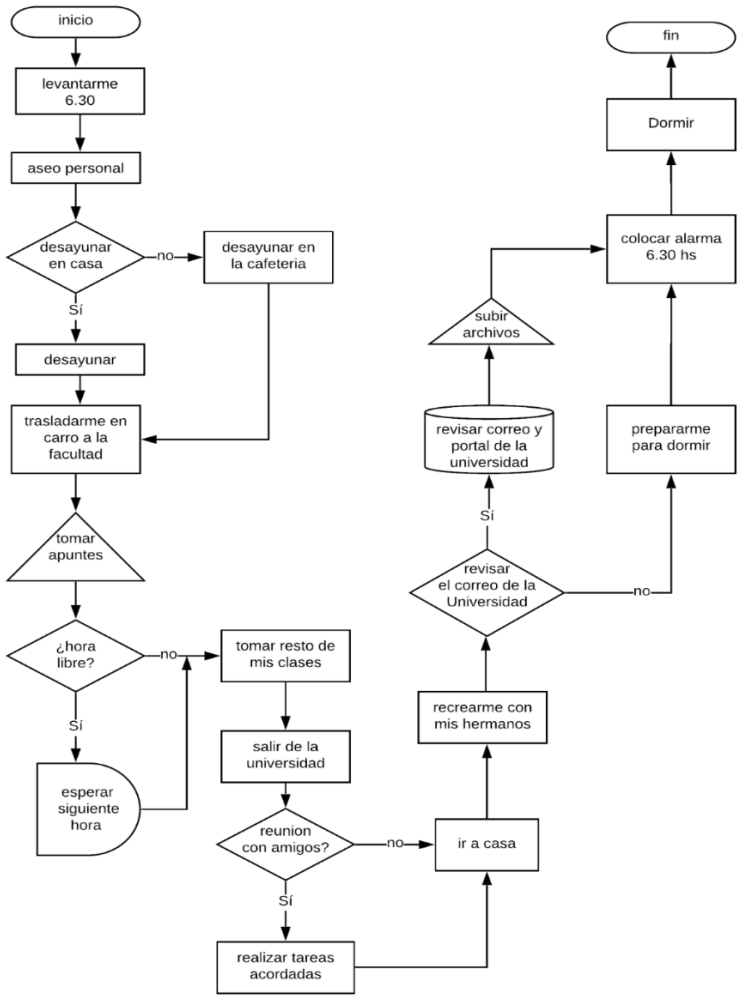


Fig. 1.3 Diagrama de flujo de un día x de una persona

Podemos citar como siguiente ejemplo un nuevo problema: ¿Cómo cocinar un huevo? y con la práctica y experiencia, prescindiendo del algoritmo escrito, siempre tendremos que tenerlo en mente para hacer el diagrama de flujo sin equivocarnos.

CAPÍTULO 2
El Microcontrolador PIC

¿Qué es un microcontrolador?

Los microcontroladores hicieron su aparición a principio de los 80' y se trata de un circuito integrado programable que contiene toda la estructura (arquitectura) de una microcomputadora. Es decir que, dentro de un microcontrolador podemos encontrar:

- Una CPU (Unidad Central de Proceso).
- Memoria RAM.
- Memoria ROM.
- Memoria EEPROM (Memoria de lectura y escritura no volátil, es decir, que no se pierden los datos cuando el circuito es desconectado).
- Puertos de Entrada/Salida (Pines de E/S) e incluso muchos modelos de microcontroladores incorporan distintos módulos "periféricos", como pueden ser: conversores analógicos/digital

(A/D), módulos PWM (control por ancho de pulso), módulos de comunicaciones seriales o en paralelo, y más.

Todo esto lo podemos encontrar dentro del mismo circuito integrado.

Cada vez existen más productos que incorporan microcontroladores con el fin de aumentar en forma sustancial sus prestaciones, reducir su tamaño y costo, mejorar su confiabilidad y disminuir el consumo de energía.

Los microcontroladores "PIC"

Los microcontroladores denominados "PIC" corresponden a la marca exclusiva de "Microchip" PIC significa "Peripheral Interface Controller" y fue desarrollado por Microchip.

Existe una gran cantidad de modelos de microcontroladores cuyas características y prestaciones varían de un modelo a otro. De esta manera los desarrolladores pueden seleccionar el modelo que mejor se ajuste a sus necesidades.

Los distintos modelos de microcontroladores se agrupan por "familia". Una familia puede estar formada por un conjunto de modelos cuyas características y prestaciones son bastante similares.

Cuando compramos un microcontrolador, la memoria del mismo se encuentra "vacía" y para que funcione es necesario que sea "programado", es decir que, el desarrollador debe escribir un programa que contenga todos los procesos que el microcontrolador debe ejecutar.

Este programa se escribe en un lenguaje llamado "Assembler" (ensamblador) cuya principal característica es su alta complejidad ya que se trata de un lenguaje "de bajo nivel", es decir, que se encuentra "más cercano" al lenguaje de la máquina que del lenguaje humano.

Por esto, los técnicos con alta capacitación están en condiciones de realizar desarrollos electrónicos que incluyan microcontroladores.

Incluso a estos especialistas les implica un gran esfuerzo intelectual y mucho tiempo de desarrollo.

Estructura de un microcontrolador

Un microcontrolador está compuesto por los siguientes componentes:

- Procesador o CPU (del inglés Central Prossesing Unit o Unidad Central de Proceso).
- Memoria para el programa tipo ROM.
- Memoria RAM para contener los datos.
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, Puertas Serie y Paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.).

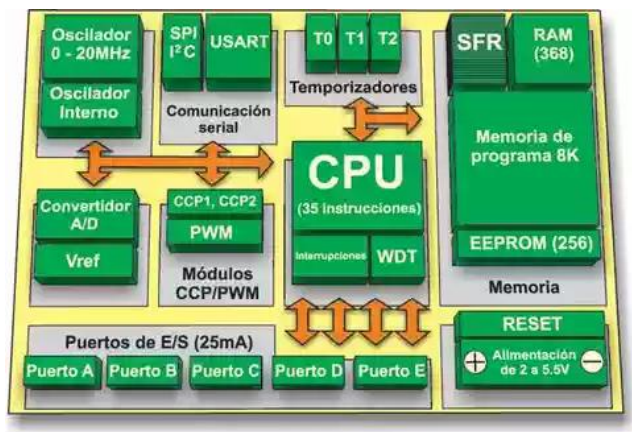


Fig. 2.1 Arquitectura del microcontrolador

El procesador o CPU

Este es el elemento más importante de un microcontrolador y determina sus principales características, tanto a nivel hardware como software. La

CPU (Central Processing Unit o Unidad Central de Proceso) se encarga de la decodificación y ejecución del programa.

Existen en la actualidad, 3 tipos de arquitectura de procesadores:

- *CISC* (Computadores de Juego de Instrucciones Complejo): Disponen más de 80 instrucciones en su portafolio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución. Una ventaja de los procesadores CISC es que ofrecen instrucciones complejas que actúan como macros.

- *RISC* (Computadores de Juego de Instrucciones Reducido): En estos procesadores el portafolio de instrucciones es muy reducido y las instrucciones son simples y en general, se ejecutan en un ciclo. La ventaja de éstos son la sencillez y rapidez de las instrucciones las que permiten optimizar el hardware y el software del procesador.

- *SISC* (Computadores de Juego de Instrucciones Específico): En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es "específico", de modo que las instrucciones se adaptan a las necesidades de la aplicación prevista.

Memoria ROM

La memoria ROM es una memoria no volátil, significa que no se pierden los datos al desconectar el equipo y se destina a contener el programa de instrucciones que gobierna la aplicación. Los microcontroladores disponen de capacidades de ROM comprendidas entre 512 bytes y 8 kbytes.

Existen distintos tipos de memorias ROM, que determinarán la aplicación del microcontrolador.

- *ROM* con máscara: Es una memoria no volátil sólo de lectura, cuyo contenido se graba durante la fabricación del chip. El elevado costo del diseño de la máscara hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a variantes de miles de unidades.

- *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 una 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.

- *EPROM*: Los microcontroladores que disponen de memoria EPROM (Erasable Programmable Read OnIy Memory) estas 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 luego 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.

- *EEPROM*: Se trata de memorias de sólo lectura, las cuales se puede escribir y borrar sólo en forma eléctrica. EEPROM (Electrical Erasable Programmable Read OnIy Memory). Tanto la programación como el borrado eléctrico de las memorias, se realizan 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 efectuar modificaciones en el programa de trabajo. El número de veces que puede grabarse y borrarse una memoria EE- PROM es finito, por lo que no es recomendable una reprogramación continua. 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 una serie de parámetros que adecuan el dispositivo a las condiciones del entorno. Este tipo de memoria es lenta.

- *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 y borrado.

Memoria RAM

La memoria de datos (RAM) de los microcontroladores PIC tienen poca capacidad, dado que sólo contiene las variables y los cambios de información que se produzcan en el transcurso de la ejecución del programa. Otra razón por la que los microcontroladores tienen poca memoria RAM es que como sólo existe un programa activo, no se requiere guardar una copia del mismo en la RAM solo se ejecuta en forma directa desde la memoria de programa (ROM).

En analogía con los usuarios de PC's y desarrolladores de programas para PC, es común que estén acostumbrados a trabajar con valores de Megabytes de memoria, pero en el caso de los diseñadores con microcontroladores trabajan con capacidades de memoria de programa de 512 bytes, 1K, 2K (hasta 64K) y de RAM de 20 bytes, 68 bytes, 512 bytes (hasta 4K)..

PIC	Memoria de Programa (en palabras de 14 bits) FLASH	Memoria de Datos (en bytes)	
		RAM	EEPROM
PIC18F83	512	36	64
PIC16F84/84A	1024	68	64
PIC16F873/874	4096	192	128
PIC16F876/877	8192	368	256

Fig. 2.2 Memoria de Programa y Memoria de Datos

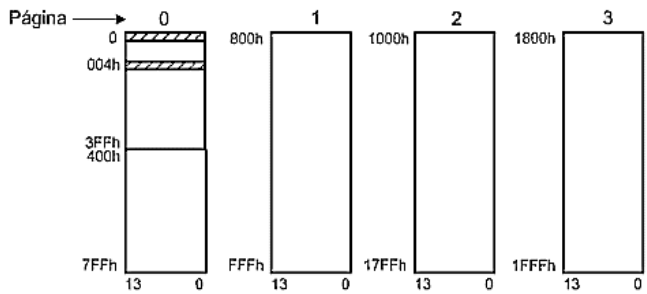


Fig. 2.3 Pagina y dirección de memoria

Registros y Bits

Un registro es una posición de memoria en la cual se puede almacenar un dato. Es decir que la memoria está dividida en pequeñas "partes" llamadas "Registros".

Dentro de la memoria, cada registro se identifica mediante un número, el cual se denomina "Dirección de memoria" y en general está expresado

en formato Hexadecimal. El primer registro de una memoria corresponde a la dirección 00H.

Dado que identificar cada registro mediante un número hexadecimal puede resultar muy complejo a la hora de diseñar el programa, existe la posibilidad de asignar un "nombre" a una dirección de registro. En general, este nombre está relacionado con la función que cada registro cumple dentro del sistema.

Los registros son utilizados por el programador (usuario) para almacenar los datos que la aplicación debe procesar, sino que, además, sirven para controlar todo el funcionamiento del microcontrolador en su conjunto. Esta función, la cumple un conjunto de registros que ya vienen "Predefinidos" desde la fábrica. Es decir, que el fabricante asigna las funciones de configuración y control del microcontrolador en un grupo de registros y el usuario no puede modificar la función que cumple cada uno de éstos.

Cada Registro está dividido en 8 "partes" a los cuales se los denomina "Bits". Entonces podemos decir que un Registro está formado por un conjunto de 8 bits.

El Bit es la menor unidad de información que un sistema digital puede procesar y sólo puede contener los valores lógicos 0 y 1.

La palabra BIT, es la contracción de las palabras "Binary Digit"

La representación de la información en los sistemas digitales en la forma de bits, dado que sus circuitos sólo pueden tener 2 estados: "encendido" o "apagado".

Podemos generalizar con estas igualdades:

1 = Encendido = Verdadero = "SÍ" = +5 Volts

0 = Apagado = Falso = "NO" = 0 Volts

Cada Bit se identifica dentro de un registro por su posición, siendo el primer Bit el número 0, encontrándose en el extremo derecho del registro (bit menos significativo o de menor peso).

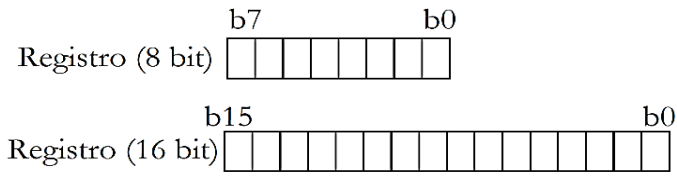


Fig. 2.4 Registro 8 y 16 bit

Al igual que los registros, se puede asignar un nombre a cada Bit para facilitar su identificación.

En un registro se puede almacenar una combinación de 8 ceros y unos. Esto nos da una cantidad de 2^8combinaciones, es decir, 256 posibles combinaciones de ceros y unos.

De este modo, un registro puede procesar valores entre 0 y 255.

La *Electrónica Digital* tiene la característica de procesar números (*dígitos*), logrando combinaciones de ceros y unos para formar la información. De esta forma, cualquier señal de naturaleza *analógica* debe pasar por un proceso de "*digitalización*", es decir, convertida a números discretos, para ser interpretada o procesada en un sistema digital.

Podemos afirmar que, el conjunto de ceros y unos almacenados en un registro se trata de un número entre 0 y 255, expresado en *sistema Binario*, esto es, en base 2. Para interpretar dicho número debemos realizar el cálculo de conversión de base para poder expresarlo en *sistema decimal*.

El siguiente ejemplo muestra el desarrollo de un cálculo de conversión de base de *sistema binario* (base 2) a *sistema decimal* (base 10):

$$10101000_2 = 0x2^0 + 0x2^1 + 0x2^2 + 1x2^3 + 0x2^4 + 1x2^5 + 0x2^6 + 1x2^7 =$$

$$10101000_2 = 0 + 0 + 0 + 8 + 0 + 32 + 0 + 128$$

$$10101000_2 = 168_{10}$$

Se llama "Peso Binario" al valor que representa un Bit según la posición que ocupa dentro del registro. El Bit que está ubicado más hacia la derecha del registro, es el Bit menos significativo (*LSB*, Least Significant Bit) y tiene un peso de $2^0 = 1$. El Bit del extremo izquierdo del registro es el Bit más significativo (*MSB*, Most Significant Bit) y tiene un peso de $2^7 = 128$. Los pesos binarios crecen de derecha a izquierda en potencias de 2.

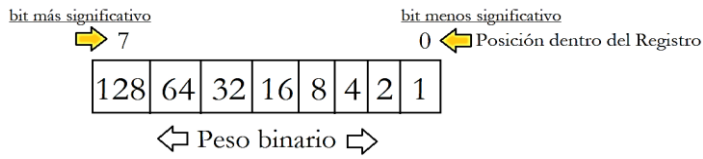


Fig. 2.5 Posición de bits de registro más y menos significativos

Una manera de simplificar el cálculo de conversión de binario a decimal es directa, sumar los valores de los pesos binarios de los bits cuyo valor sea 1.

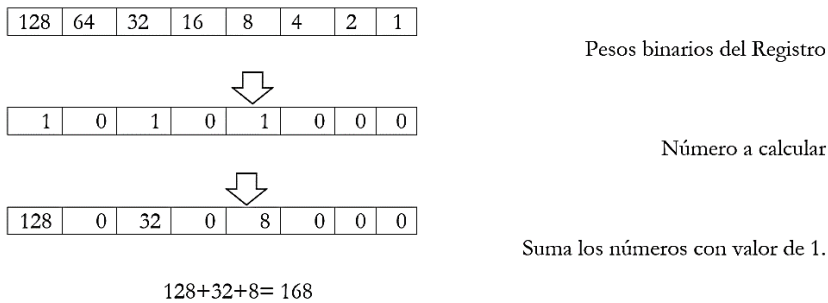


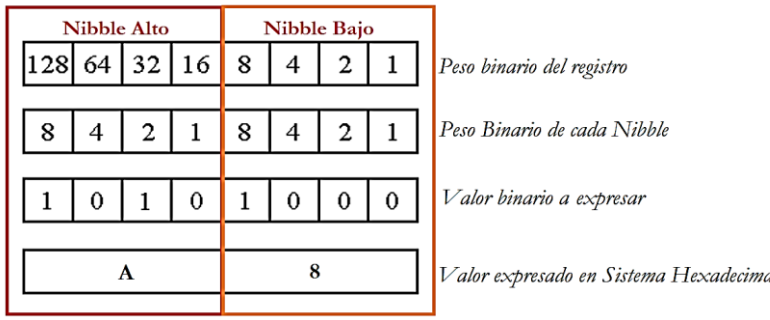
Fig. 2.6 Calculo conversión binaria a decimal

El sistema hexadecimal es un sistema en base 16 y consta de 16 dígitos diferentes que son: del 0 al 9 y luego de la letra "A" a la "F", es decir, 10 dígitos numéricos y seis caracteres alfabéticos.

El sistema hexadecimal se usa como forma simplificada de representación de números binarios y debido a que 16 es una potencia de 2 (2^4=16), resulta muy sencilla la conversión de los números del sistema binario al hexadecimal y viceversa.

Mediante el sistema hexadecimal podemos representar un valor numérico de 8 bits utilizando sólo 2 dígitos. De ésta manera estamos dividiendo el registro de 8 bits en 2 partes de 4 bits cada una llamada Nibble. Al Nibble correspondiente a los 4 bits menos significativos, se le

denomina "Nibble Bajo" y al Nibble correspondiente a los 4 bits más significativos se le denomina "Nibble Alto".



$$10101000_2 = 168 = A8_{16}$$

Fig. 2.7 Representación de 2 dígitos en Hexadecimal

El sistema hexadecimal es utilizado para identificar las direcciones de registros de las memorias en sistemas digitales porque permite representar el valor de un Nibble con 1 digito, ya que:

$$1111_2 \rightarrow (8 + 4 + 2 + 1) = 15 = F_{16}$$

Esto permite representar números grandes utilizando unos pocos dígitos. Por ejemplo:

$$FF\ FF_{16} = 11111111\ 11111111_2 = 65535$$

En este tipo de programación es muy habitual utilizar tres sistemas de numeración (*Binario, Decimal y Hexadecimal*), por supuesto, dependiendo que necesitamos realizar o hacer. Por esto, es fundamental manejar muy bien estos conceptos.

Líneas de Entrada/Salida (E/S), (Puertos)

Los microcontroladores poseen una serie de pines destinados a la entrada y salida de datos o señales digitales. Estos pines son denominados "Puertos".

Como mencionamos antes, todo el funcionamiento del microcontrolador está controlado a través de los registros. Los puertos no son la excepción, también están controlados por los registros. Por esto, un puerto no puede estar formado por más de 8 pines; 1 Pin por cada Bit de un registro. Un puerto si puede estar formado por menos de 8 pines.

Un microcontrolador puede contener varios puertos dependiendo del modelo.

A cada puerto se lo identifica con una letra. Por ejemplo: el puerto A "PORTA", el puerto B "PORTB", etc.

Para poder utilizar un puerto, primero el mismo debe ser configurado. Cada pin de un puerto puede ser configurado como entrada o salida independiente del resto de los pines del mismo puerto.

Los puertos están relacionados al tamaño del procesador, es decir que un puerto de 8 bits es porque el procesador es de 8 bits. Para el caso de un procesador de 64 bits, posee la capacidad de tener un puerto de 64 bits. Para una mayor ilustración, podemos ver en la siguiente imagen el puerto A y B de un microcontrolador PIC16F628A.

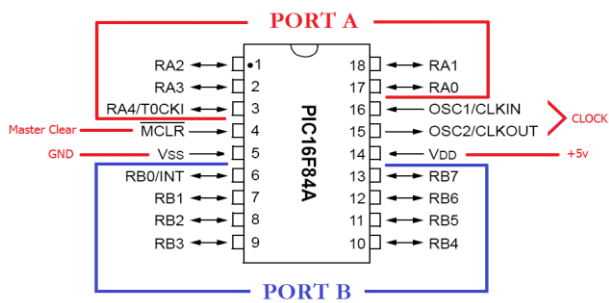


Fig. 2.8 Puertos de microcontrolador PIC16F628A

Módulos Temporizadores Internos (TMRs)

Un temporizador interno (TMR) es un módulo de hardware incluido en el mismo microcontrolador el cual está diseñado para incrementar en forma automática el valor de un registro asociado al TMR cada vez que el módulo TMR recibe un pulso. A este pulso se le llama "señal de reloj".

El *módulo* TMR siempre incrementa el valor del registro asociado, nunca decrementa dicho valor.

Algunas de las familias de microcontroladores pueden incluir más de un módulo TMR y la señal de reloj de cada uno de éstos puede ser de origen interno o externo.

Si el origen de la señal de reloj está configurado como externo, el módulo temporizador puede ser utilizado como un contador de eventos externos, incrementando el TMR con cada pulso recibido mediante el pin correspondiente.

Si el origen de la señal de reloj es interno, el TMR incrementa con cada ciclo del oscilador. Esto permite utilizar el temporizador como "contador de ciclos de programa", donde, un ciclo corresponde al tiempo de ejecución de una instrucción ($t_{e.i.}$), lo cual se puede calcular con la siguiente fórmula:

$$t_{e.i.} = \frac{1}{\frac{F.Osc}{4}}$$

Donde "F.Osc." es la frecuencia del oscilador utilizado.

Dado que la velocidad de ejecución del microcontrolador corresponde a ¼ de la velocidad del cristal utilizado, cada ciclo de programa se ejecuta

en un tiempo determinado según el cristal que estemos utilizando.

Por ejemplo; con un cristal de 4 Mhz la velocidad real de procesamiento del microcontrolador es de 1 Mhz. Aplicando la siguiente fórmula:

$$\frac{1}{\frac{4.000.000}{4}} = \frac{1}{1.000.000} = 0.000001 = 1uS\ (microsegundo)$$

Esto significa que cada ciclo de programa se ejecuta a 1/1.000.000 (1 uS) y dado que cada incremento del TMR corresponde a un ciclo de programa, si contamos los incrementos de un TMR, podremos en forma indirecta calcular el tiempo transcurrido.

El incremento del TMR se realiza de manera automática y de forma paralela a la ejecución del resto del programa. Esto significa que el programa no debe "controlar" el incremento del TMR, lo cual sí debe realizarse en las temporizaciones por bucles, en las cuales el microcontrolador se queda "esperando" a que transcurra un tiempo, y no pueden ejecutarse otras tareas mientras no finalice la temporización.

Entonces el TMR puede ser utilizado como temporizador mientras el microcontrolador ejecuta otros procesos.

Por ejemplo: si diseñamos un programa que encienda y apague un led cada 1 segundo, el microcontrolador enciende el led y luego puede ocuparse de esperar a que transcurran los ciclos correspondientes a 1 segundo, luego apaga el led y continúa esperando.

Pero si el microcontrolador debe realizar alguna otra tarea además de temporizar, por ejemplo, mostrar valores en dígitos de 7 segmentos, el micro no puede quedarse temporizando por bucles ya que mientras se encuentre temporizando no podrá realizar el proceso de muestreo de los datos en los displays, y como resultado tendremos que los displays se apagan mientras el microcontrolador se encuentra temporizando.

En este caso, se utiliza un TMR para realizar la temporización mientras el microcontrolador se ocupa de mostrar los datos en los displays.

En general los microcontroladores cuentan con una interrupción por desbordamiento del TRM, es decir que, el microcontrolador se interrumpe cuando el TMR0 pasa de 255 (*B'11111111'*) a 0 (*B'00000000'*) en los temporizadores de 8 bits y cuando pasa de 65535 (*B'11111111_11111111'*) a 0 (*B'00000000_00000000'*) en los temporizadores de 16 bits.

Dado que 256 μs es un tiempo muy corto (para un TMR de 8 bits a 4 MHz), existe la posibilidad de preescalar el TMR0 para que el registro asociado al TMR se incremente cada 2, 4, 8, o más ciclos. De esta manera, estamos multiplicando la cantidad de ciclos que deben ejecutarse para realizar un incremento del TMR. Al preescalador también se le denomina predivisor.

Por ejemplo, con un predivisor de 8, cada incremento del TMR se realiza cada 8 ciclos de programa y el desbordamiento del TMR se realizará cada 2048 ciclos (255 incrementos * 8 ciclos).

Así podemos efectuar interrupciones con periodos de tiempo mayores.

Utilizando un microcontrolador

Como mencionamos antes, el microcontrolador tiene una memoria de programa, donde grabamos las instrucciones necesarias para que el micro realice el trabajo que necesitamos. Cuando compramos un microcontrolador, la memoria de programa viene vacía. Para que un microcontrolador funcione es necesario "programarlo".

Los microcontroladores se programan en un lenguaje de programación llamado Ensamblador (en inglés Assembler) cuya principal característica es su altísima complejidad.

Los lenguajes de programación se clasifican según el "Nivel" de programación en:

- Lenguaje de "***Alto Nivel***": permite que los algoritmos se expresen en un nivel y estilo de escritura legible y comprensible

por el hombre. En la actualidad se trata de lenguajes de tipo visual.

- Lenguaje de "***Bajo Nivel***": el usuario se acerca un poco más al lenguaje de máquina. Permiten un acceso más amplio al control físico de la máquina (hardware).

- Lenguaje ***Ensamblador***. Podríamos considerarlo el lenguaje de más bajo nivel. El usuario escribe código en el mismo "idioma" del procesador. Se tiene control total del sistema. Es necesario un conocimiento de la arquitectura mecánica del procesador para realizar una programación efectiva. El lenguaje de programación es muy específico para cada modelo de procesador, incluso puede variar de un modelo a otro de procesador dentro de un mismo fabricante.

Podemos decir que los lenguajes de alto Nivel se asemejan más al lenguaje humano y que los lenguajes de bajo Nivel se asemejan más al lenguaje de máquina y en el lenguaje ensamblador el usuario debe programar en el propio "idioma del procesador".

Fig. 2.9 Niveles de lenguajes de programación

El microcontrolador unicamente entiende números, significa que, el código Assembler (*texto*) no puede ser procesado directo por el microcontrolador. Para poder grabar el programa en el micro, existe un paso previo, en el que, primero se debe convertir el texto del código Assembler a números, en general, en formato hexadecimal. Este proceso se llama "Compilación".

Una vez desarrollado el código Assembler, el mismo debe ser "compilado". Esto se realiza con un "software compilador" el cual se puede obtener de manera gratuita desde la página Web del fabricante del microcontrolador.

El archivo una vez compilado tiene una extensión (nombre).hex. Por ejemplo: micro.*hex*

Por último, después de compilado, el programa está listo para ser grabado al microcontrolador. Esto se efectuar mediante una "placa programadora". Esta placa programadora se denomina "programador". Existen distintos tipos de programadores los cuales pueden para conectar a la PC mediante el puerto Serie (COM) o Paralelo (LPT). En la actualidad, los más usados son los programadores por puerto USB.

Cada programador trabaja en conjunto con un software, mediante el cual se realiza el proceso de lectura y escritura de los microcontroladores.

A continuación, se presenta un resumen del proceso de desarrollo del código y grabación de un microcontrolador:

1. Escribir el código Assembler. Se crea un archivo con extensión *ASM*.
2. Compilar el código Assembler. Se crea un archivo con extensión *HEX*.
3. Grabar (transferir) el programa desde la PC al microcontrolador mediante un programador.

De los pasos antes mencionados, podemos destacar que, el primero gasta el 98 % de tiempo y esfuerzo y es aquí donde Niple le ofrece sus ventajas.

La Programación Visual de microcontroladores PIC con Niple.

Como mencioné en los párrafos anteriores, la programación de los microcontroladores resulta una tarea muy compleja dado que el usuario debe conocer la arquitectura del procesador y además debe dominar el lenguaje de programación "Ensamblador". Si bien existen distintas versiones del lenguaje C o Basic adaptadas a la programación de microcontroladores, todavía la programación resulta muy compleja por el alto grado de abstracción que implica.

Fig. 2.10 Representación lenguaje programación Niple

Niple es un entorno de desarrollo visual para programación de microcontroladores PIC. Con Niple, el usuario únicamente diseña un diagrama de flujo de manera visual y en alto nivel de programación, en lenguaje humano, y el software genera el código Assembler de manera

automática.

Además, el software controla y supervisa todo el proceso de diseño, advirtiendo al usuario sobre posibles errores en la lógica del programa que se está diseñando. Esta ventaja no la tiene ningún otro lenguaje de programación. Niple nos ofrece la posibilidad de realizar desarrollos con microcontroladores con un mínimo de esfuerzo.

El principal objetivo de Niple es facilitar al máximo la programación de los microcontroladores. Para esto, el software ofrece un entorno visual, donde todo el sistema se adapta a las características del modelo de microcontrolador seleccionado; es decir, que carga el mapa de memoria con sus respectivos nombres de registros y bits, asigna las funciones correspondientes a cada Registro y Bit dentro del sistema, ajusta el tamaño de la memoria EEPROM, activa los módulos correspondientes, etc. Es decir que, Ud. tendrá disponible solamente las características que corresponden al modelo de micro en uso.

El concepto de la programación es que el usuario se concentre en "*¿qué es lo que quiere hacer?*", y Niple "*¿sabe cómo hacerlo?*". Es decir que el usuario sólo se dedica a diseñar la estructura lógica del proceso que desea realizar y el software se encarga de resolver todo el desarrollo del código Assembler, controlando hasta el más mínimo detalle.

Los PIC más antiguos, con memoria PROM o EPROM se están renovando gradualmente por chips con memoria Flash. Así mismo, el juego de instrucciones original de 12 bits ha sido suplantado por juegos de instrucciones de 14 y 16 bits. Microchip todavía vende versiones PROM y EPROM de la mayoría de los PIC para soporte de aplicaciones antiguas o grandes pedidos.

Podemos pensar tres grandes gamas de MCU PIC en la actualidad: Los básicos (Line base), los de medio rango (Mid Range) y los de alto desempeño (high performance).

Los PIC18 son considerados de alto desempeño y tienen entre sus miembros a PIC con módulos de comunicación y protocolos avanzados

(USB, Ethernet y Zigbee por ejemplo).

El microcontrolador r f PIC integra todas las prestaciones del PICmicro de Microchip con la capacidad de comunicación inalámbrica UHF para aplicaciones RF de baja potencia. Estos dispositivos ofrecen un diseño muy comprimido para ajustarse a los cada vez más demandados requerimientos de miniaturización en aparatos electrónicos. Aun así, no parecen tener mucha salida en el mercado.

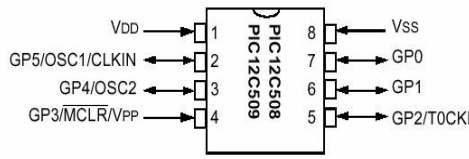
Los dsPIC son el penúltimo lanzamiento de Microchip, comenzando a producirlos a gran escala a finales de 2004. Son los primeros PIC con bus de datos inherente de 16 bits. Incorporan todas las posibilidades de los anteriores PIC y añaden varias operaciones de DSP implementadas en hardware, como multiplicación con suma de acumulador (multiply-accumulate, o MAC), barrel shifting, bit reversion o multiplicación 16x16 bits.

Microchip en el año 2007 lanza los nuevos microcontroladores de 32 bits con una velocidad de procesamiento de hasta 1. 6 DMIPS/ MHz con capacidad HOST USB. Sus frecuencias de reloj pueden alcanzar los 80 MHz a partir de cristales de cuarzo estándares de 4 a 5 MHz gracias a un PLL interno.

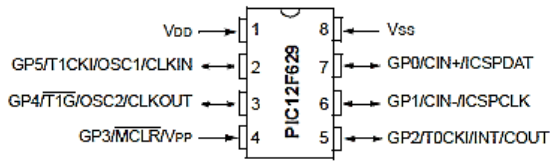
Funcionan a 3.3V en sus puertos de entrada y salida, aunque el fabricante indica que salvo en los pines con función analógica, en la mayoría se toleran tensiones de hasta 5 V. Disponen de una arquitectura optimizada con alto grado de paralelismo y núcleo de tipo M 4K y una elevada capacidad de memoria RAM y FLASH ROM. Todo ello hace que estos MCUs permitan una elevada velocidad de procesamiento de información.

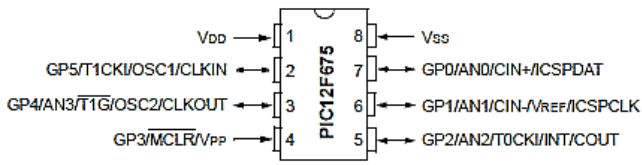
Los PICS más comúnmente utilizados son:

- **PIC12C508/509** (Encapsulado reducido de 8 pines, oscilador interno, popular en pequeños diseños como el iPod remote).
-



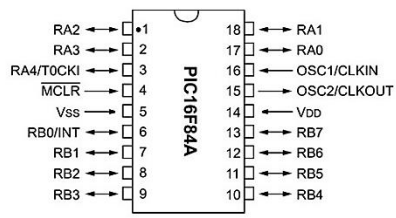
- **PIC12F629/675** (Encapsulado reducido de 8 pines, oscilador interno, popular en pequeños diseños proyectos, protector de tensión, transmisor RF, etc.)
-





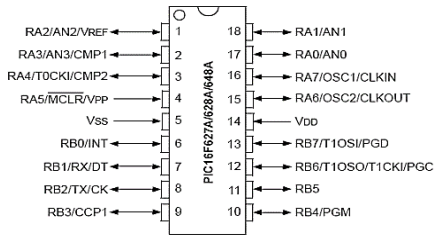
- **PIC16F84** (Considerado obsoleto, pero imposible de descartar y muy popular por la gran cantidad de proyectos que lo utilizan).

- **PIC16F84A** (Actualización del anterior, la A indica que posee con módulos de comparación A/D, algunas versiones funcionan a 20 MHz, compatible 1:1).

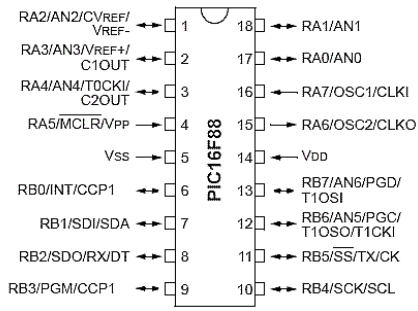


- **PIC16F628A/648A** (Opciones típicas para migrar o actualizar desde diseños antiguos hechos con el PIC16F84A. Poseen

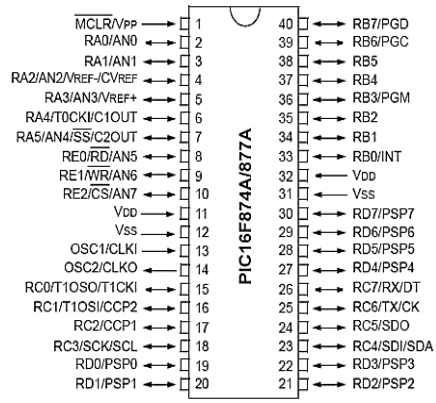
módulos de comparación Analógico/Digital, Capture & Compare CCP (PWM), comunicación en serie USART y oscilador interno, entre otras mejoras y además es más económico).



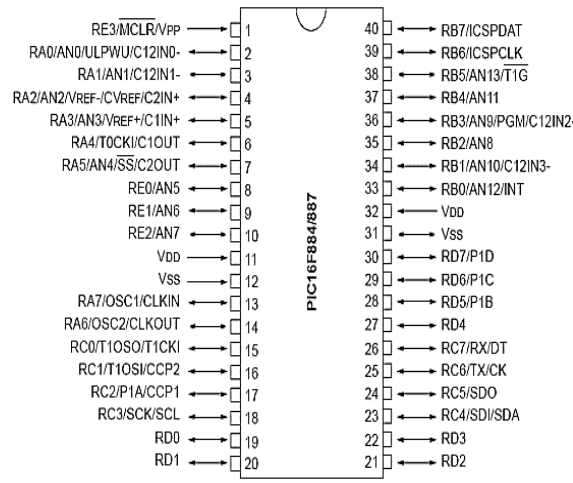
- **PIC16F88** (Nuevo sustituto del PIC16F84 A y superior al PIC16F648A con más memoria, oscilador interno, USART, SSP, comparador y convertidor A/D).



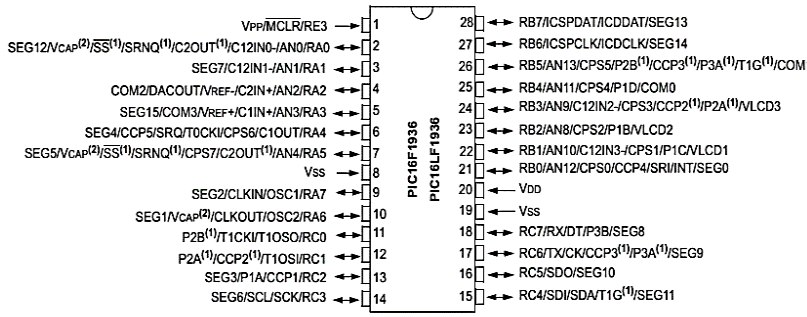
- **PIC16F87X/87XA** (Hermanos mayores del popular PIC16F84/84A, con mejoras importantes en su hardware. Muy populares en proyectos de *«aficionados»*).



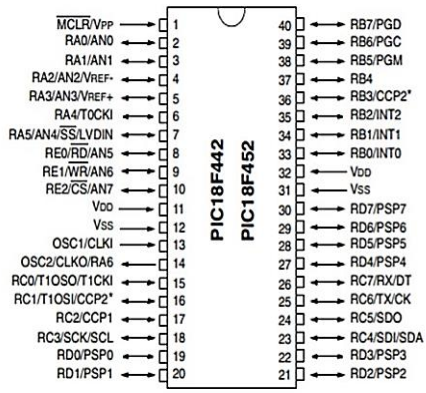
- **PIC16F886/887** (Nuevo sustituto del 16F876A y 16F877A con la diferencia que ahora ya incluye oscilador interno, considerados los más cercanos a la familia 18 F).



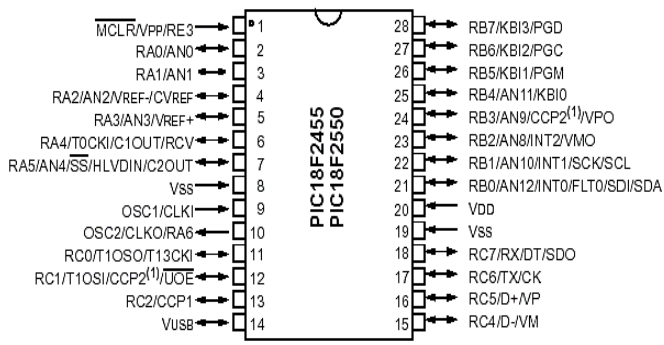
- **PIC16F193X** (Nueva gama media de PIC optimizado y con mayor RAM, ampliando su set de instrucciones a 49 instrucciones frente al conjunto de 35 más utilizadas).



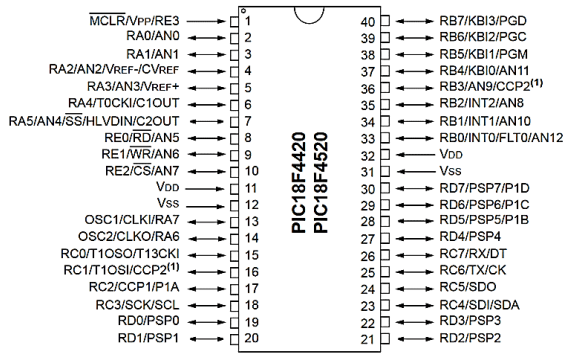
- **PIC18F452**. Muy utilizado para uso de tarjetas entrenadoras.



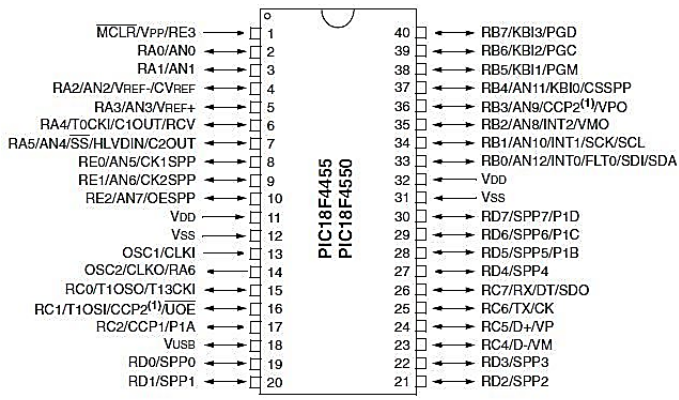
- **PIC18F2455.** (Para desarrollo de uso de puerto USB 2. 0.)



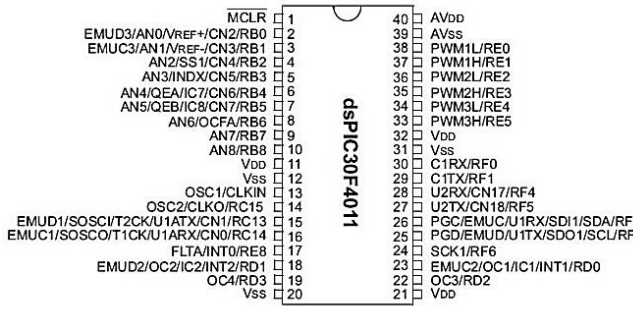
- **PIC18F2520/4520**. (Múltiples comparadores y convertidores A/ D).



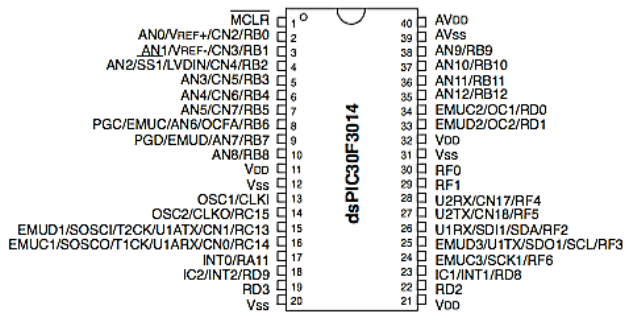
- **PIC18F2550/4550.** (Manejo de puertos USB 2. 0 y muy versátil. Por ello, es muy útil para el desarrollo de proyectos del ambiente académico).
-



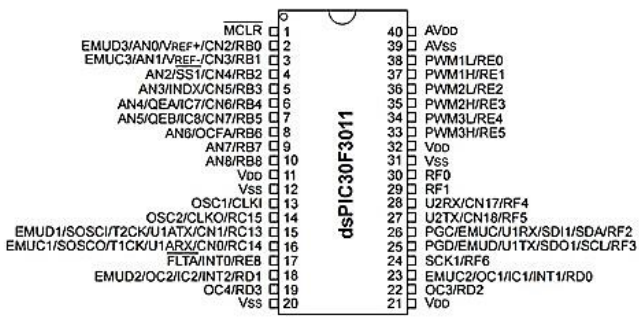
- **dsPIC30F2010**. Muy empleado para control de servo motores y control PID.



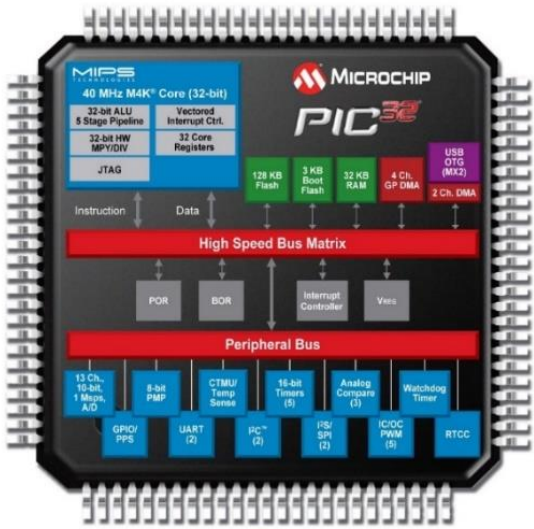
- **dsPIC30F3014**. periféricos de control de motor y una interfaz de Codec que hacen posible usar convertidores digitales de potencia eficientes, algoritmos avanzados de control de motor y aplicaciones de voz y audio.
-



- **dsPIC30F3011.** (Muy utilizado en desarrollos para control electrónico de motores eléctricos de inducción, control sobre audio, aplicaciones de sistema de control PID, etc.).

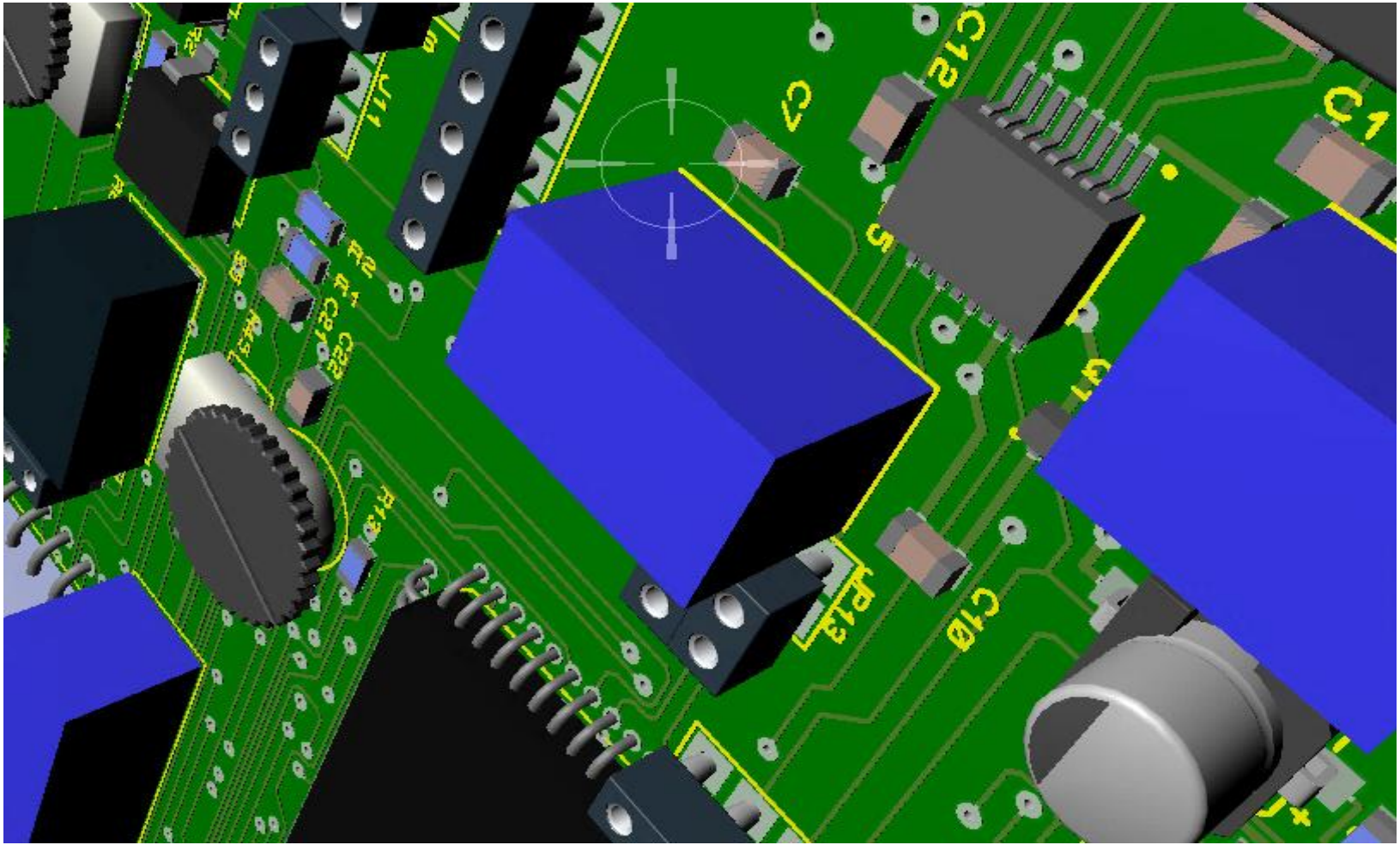


- **PIC32.** (Nueva gama de PIC de 32 bits, los más modernos son compatibles con USB 2 . 0).



Se puede encontrar extensa información y documentación sobre los micros PIC en Internet, principalmente por dos motivos: el primero, porque han sido muy usados para romper los sistemas de seguridad de varios productos de consumo masivo (televisión de pago, PlayStation, etc.), lo que atrae la atención de los crackers *«rompedor»*; y segundo, porque el PIC 16C84 fue uno de los primeros microcontroladores fácilmente reprogramables para aficionados. Hay muchos foros y listas de correo dedicados al PIC, en los que un usuario puede proponer sus dudas y recibir respuestas.

Con estos dispositivos también se pueden desarrollar sistemas SCADA (microSCADA), para adquirir y enviar datos al puerto serial de un computador utilizando, por ejemplo, transmisión UART y el protocolo RS-232, o implementando el protocolo TCP/IP directamente para construir un sistema completamente embebido.


C12
C7
C1
C10
JP13

Capítulo 3
Esquemáticos, Simulación y Placas Electrónicas con PROTEUS

Hasta hace poco tiempo atrás, no disponíamos de las herramientas físicas y las de software «programas» con las prestaciones que se presentan en estos tiempos. Esta es una verdad excelente. En lo referente a equipos físicos, el que empieza es evidente que no pueda disponerlas, sin embargo, en el apartado de software, existen una serie de herramientas que prestan gran ayuda a los diseñadores incluso a los que comienzan con este tipo de trabajo, facilitando su tarea y aprendizaje en los últimos, esto repercute en un mejor aprovechamiento del tiempo y un mayor rendimiento.

Existe una gran variedad de paquetes EDA (Herramientas de CAD electrónico), estas herramientas unas más conocidas que otras, entre ellas podemos citar: Elegance, TangoPCB, Livewire, ExpresPCB, Eagle

«Aguila», etc. Todas son marcas registradas, algunas de ellas disponen de versiones demo que pueden servir para conocer su funcionamiento y saber los detalles técnicos de funcionalidades.

Pero me ocupare en mostrarles las herramientas que posee **Proteus**, comentarles que existen mucha información y manuales en español, muy muy buenos y muy extensos, los cuales sacarán de dudas a quien las tenga. Ahora bien, voy a poner en manos del lector, estudiante o interesado, las bases mínimas para colocar en práctica las ideas o necesidades que le surjan en el ejercicio de todos los casos que trataremos en esta edición y para en un futuro sea para su trabajo o simplemente para su entretenimiento (hobby), por lo tanto, será sólo eso un asomo al uso de esta herramienta de diseño y análisis electrónico.

El objetivo de este capítulo es mostrar, como crear un simple esquemático, como realizar una simulación interactiva que usa Proteus VSM. Mientras nos concentramos en el empleo de Componentes Activos y las instalaciones de eliminación por depurado de fallos del editor ISIS también miraremos lo básico de presentar una dirección de circuito esquemática y general. La cobertura plena de estos temas puede ser encontrada en el Manual ISIS.

Proteus VSM

La Suite de Proteus es un entorno integrado diseñado para la realización completa de proyectos para construcción de equipos electrónicos en todas sus etapas. Proteus el Sistema Virtual de Modelado (VSM) combina el modo mixto la simulación de circuito de SPICE, componentes animados y modelos completos de diseños basados en microprocesador para facilitar la co-simulación del microcontrolador. Esta herramienta dispone de los módulos conocidos por: Captura de esquemáticos ISIS, Layout de ARES PCB y Simulador (ProSpice/VSM). Por primera vez, es posible desarrollar y probar diseños antes de construir un prototipo físico.

Captura de esquemático ISIS

Una vez instalado el programa, iniciar ISIS, este presenta la suite de diseño en la que se aprecian dos zonas, a la izquierda un visor del plano del proyecto, debajo, la ventana para mostrar dispositivos y a la derecha la zona de trabajo propiamente dicha, en la que el diseñador trazará los circuitos eléctricos con sus componentes, posteriormente podrá ejecutar un modelo virtual que simule su proyecto en funcionamiento.

Al iniciar el programa la primera venta que nos muestra es (inicio-home), para dar inicio a un circuito eléctrico (esquemático), debemos dar clic sobre esquema eléctrico (icono de un diodo y un condensador)

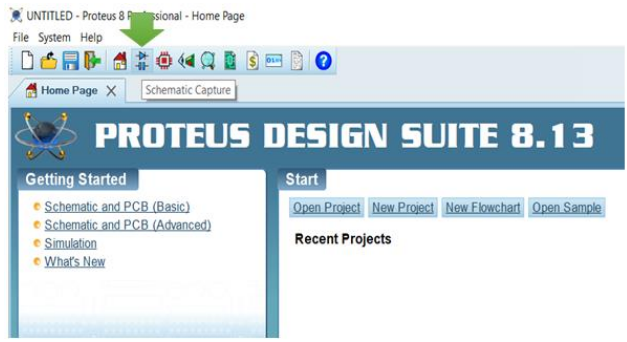


Fig. 3.1 Suite Proteus 8.13

O bien, podemos empezar asignando el nombre del proyecto, en este caso le pondremos "prende-apaga".

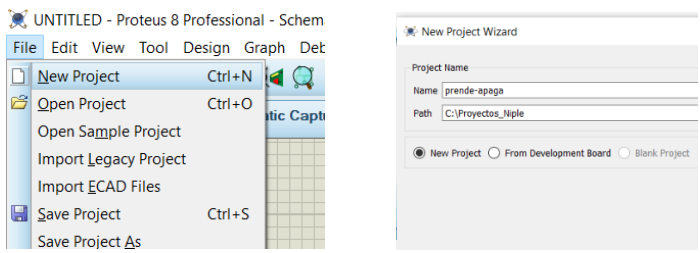


Fig. 3.2 Nuevo proyecto en Suite Proteus

Luego de hacer siguiente, para la creación del esquemático,

seleccionamos como template "Landspace A4", luego damos siguiente (Next). Allí, en la creación del circuito impreso o placa PCB, seleccionamos la opción del template "Generic Simple Layer" (capa simple genérica). Luego siguiente, allí podemos ver la selección de una placa FR4, con un dieléctrico de 4.8 y un diámetro de 1.55 milímetros.

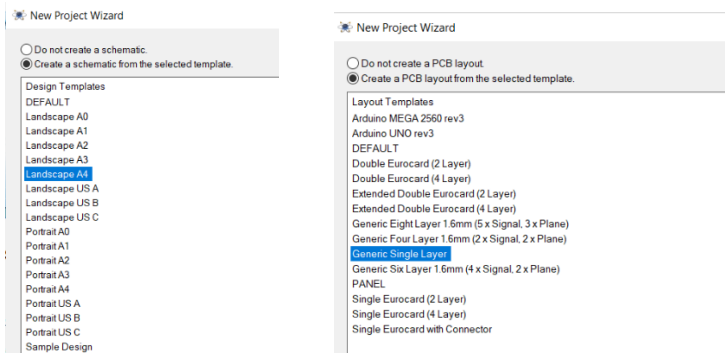


Fig. 3.3 Configuración hoja de trabajo

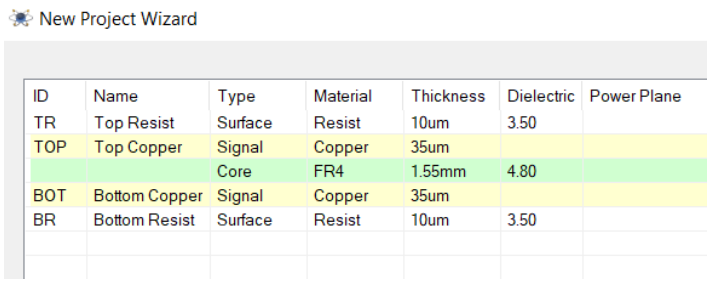


Fig. 3.4 Capas de trabajo

En la siguiente ventana, nos muestra que el inicio de la capa en "Top Cooper" y finalización en "Botton Copper" del agujero metalizado "Through Hole" en la placa, hacemos clic en siguiente, luego en la próxima ventana nos brinda una información gráfica del agujero metalizado y ambas capas de la placa.

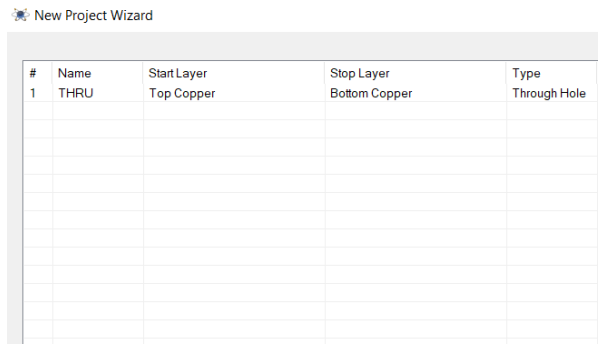


Fig. 3.5 Agujero metalizado

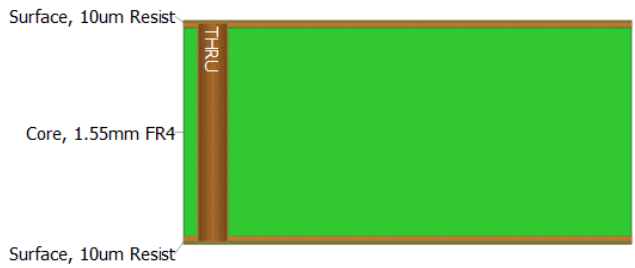


Fig. 3.6 Dimensión agujero metalizado

En la siguiente ventana nos pregunta que deseamos ingresar, si algún firmware o programa para algún microcontrolador, que por el momento no lo vamos a introducir, dejando en "No Firmware Proyect", luego presionamos el botón "Finalizar" del asistente de proyectos para finalizar.

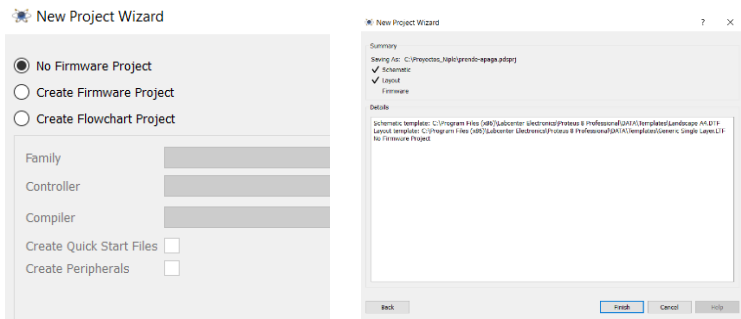


Fig. 3.7 Creación Código Fuente para proyecto

Vista de las barras de menú, herramientas "horizontal", herramientas "vertical", área de trabajo, visor de área de trabajo, rotulo de la hoja del proyecto y herramienta de simulación.

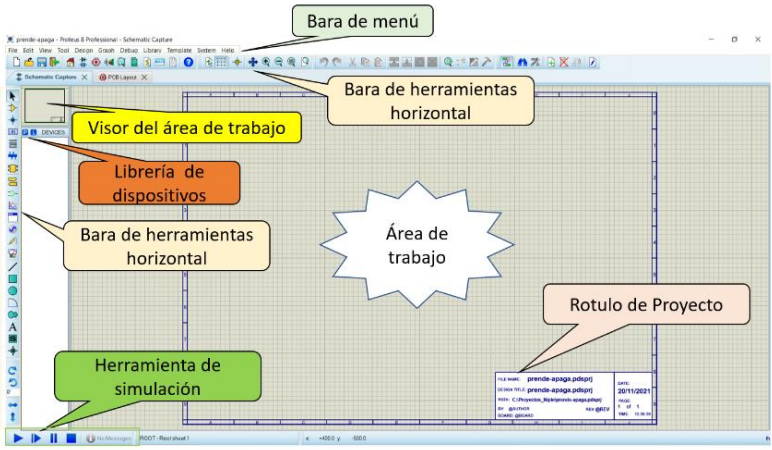


Fig. 3.8 ISIS barra de herramientas y hoja de trabajo

Barra de menú

A continuación, se mostrará un desglose de todas las solapas de la barra de menú, para una mayor ilustración para el lector.

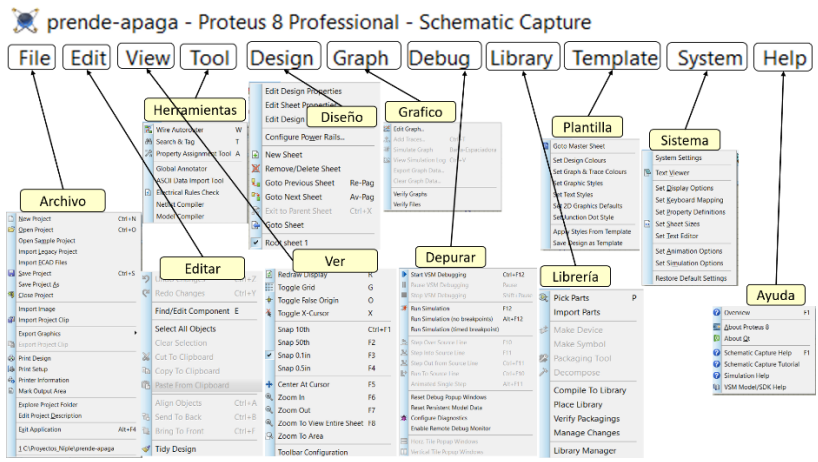


Fig. 3.9 Barra de menú ISIS

Barras de herramientas

Para dar una mayor ilustración, graficare las barras de herramientas horizontales con la descripción de cada icono.

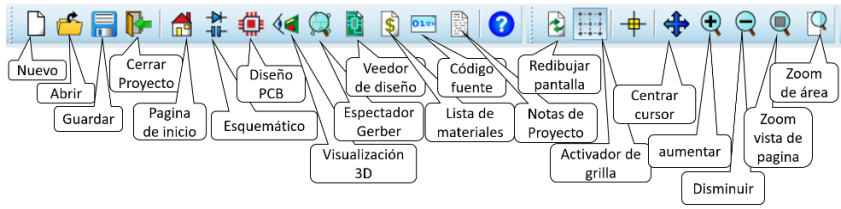


Fig. 3.10 Barra Herramientas Horizontal

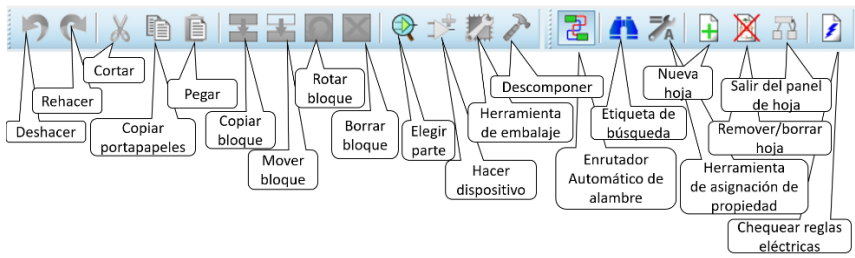


Fig. 3.11 Continuación barra de herramientas horizontal

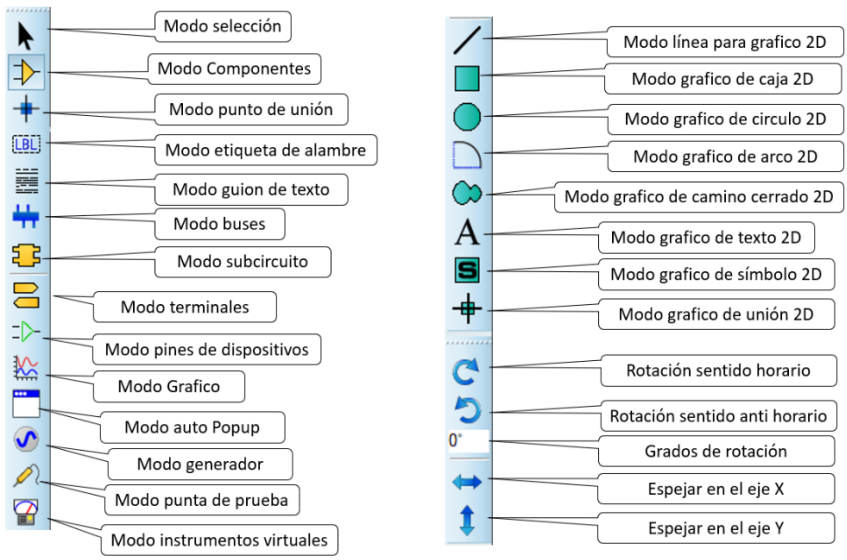


Fig. 3.12 Barra de herramienta vertical

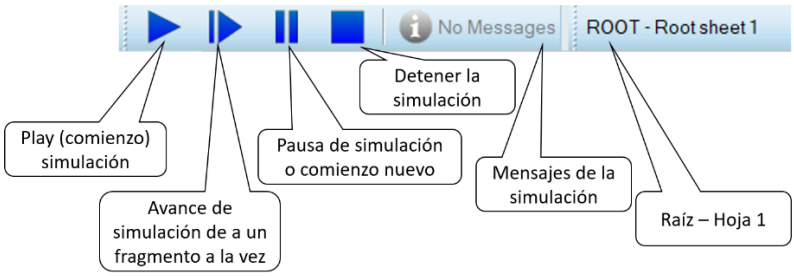


Fig. 3.13 Barra de herramienta de simulación.

Panel de Proteus

En la columna de la izquierda pulsamos sobre el botón P que nos antepone la ventana de librerías, en esta ventana elegiremos el dispositivo que necesitemos con doble-clic, eligiendo otros componentes del mismo modo, lo que enviará dichos dispositivos a la ventana principal, en la columna de la izquierda donde aparecen los componentes a usar. Incorpora una librería de más de 7.000 modelos de dispositivos digitales y analógicos.

Fig. 3.14 Herramienta de dispositivos.

Layout de ARES PCB

Esta herramienta utilizaremos para la elaboración de nuestras placas de circuito impreso, ARES coloca de un posicionador automático de elementos, con generación automática de pistas. Un diseñador experto ya no tiene que hacer el trabajo duro, ya que el ordenador es quien se encarga de esta tarea.

PROSPICE

Se trata de una herramienta de simulación de circuitos según el estándar industrial. La versión básica, suministrada con todas las versiones de Proteus, sólo soporta análisis de transitorios.

VSM

La herramienta VSM está integrada a PROTEUS, se trata de un completo simulador para esquemas electrónicos que contienen microprocesador. El corazón de VSM es ProSPICE, un producto que combina un núcleo de simulación analógica usando el estándar ISO/IEC, con modelos animados de los componentes electrónicos y los microprocesadores que comprende el circuito, tanto si el programa se ha escrito en ensamblador como si se ha utilizado un lenguaje de alto nivel, permitiendo interactuar con nuestro diseño, utilizando elementos gráficos animados realizando operaciones de indicadores de entrada y salida.

La simulación se ejecuta en tiempo casi real, los efectos se pueden considerar prácticamente como a tiempo real. Incorpora prácticos controles de depuración paso a paso y visualización del estado de las variables. La característica más sorprendente e importante de VSM es su capacidad de simular el software que se ejecuta en el microcontrolador y su interacción con cualquier componente electrónico digital o analógico conectado a él.

Programa con microprocesadores

Como podremos comprobar y según dice el fabricante, la característica más sorprendente y relevante de VSM es su capacidad de simular el

software que se ejecuta en el microcontrolador y su interacción con cualquier componente electrónico digital o analógico conectado a él.
Veamos un ejemplo sencillo: Trazamos un circuito sencillo compuesto por unos pocos componentes alrededor de un microcontrolador como el PIC16F675, el siguiente puede ser el esquema de lo descrito.

Instalación de Proteus 8.13

Nos dirigimos a descargas, para luego descomprimir el archivo ZIP, dando clic derecho.

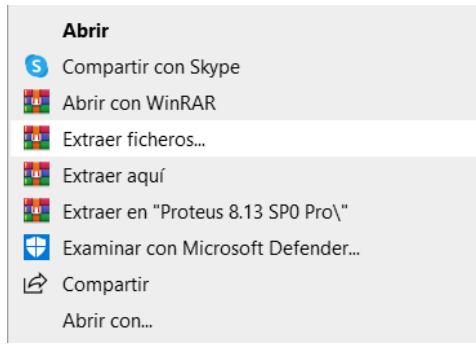


Fig. 3.15 Extracción de archivos Winrar

Esperamos a que termine de descomprimir el archivo ZIP.

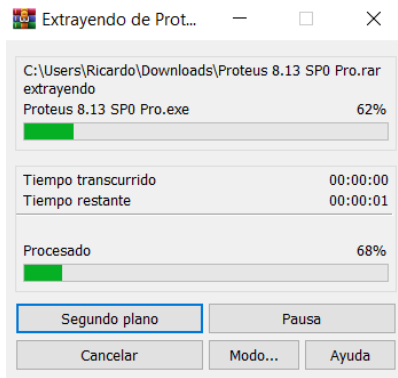


Fig. 3.16 Extracción archivos Proteus 8.13

Una vez descomprimido, procederemos a ejecutar como administrador dando clic derecho en el asistente de instalación del software Proteus Design Suite 8.13.

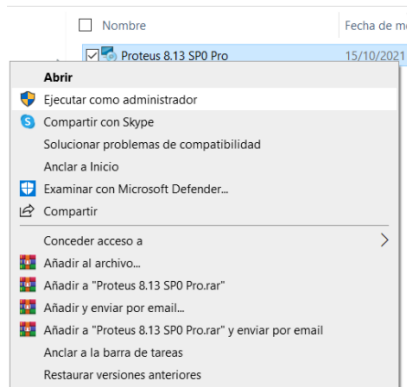


Fig. 3.17 Ejecutar instalación Proteus 8.13

Luego comienza la instalación de Microsoft Visual C++, en sus distribuciones 2008, 2010, 2013 y 2015.

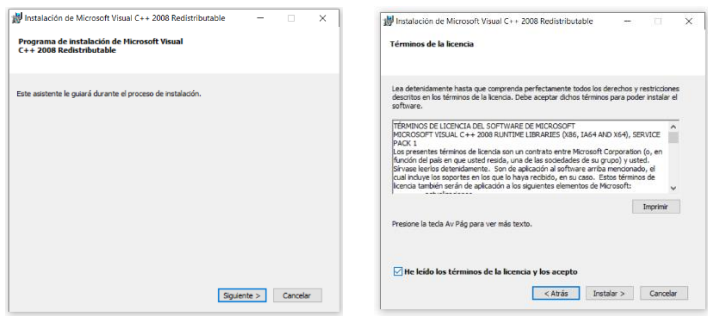


Fig. 3.18 Instalación Proteus y condiciones de licencia

Aceptamos la licencia y le damos siguiente.

Finalizamos, luego aceptamos los términos y hacemos clic en "instalar"

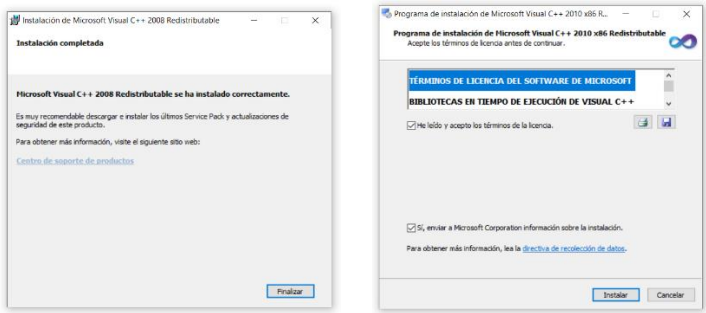


Fig. 3.19 Instalación paquetes adicionales a Proteus

Finalizada la instalación, damos a finalizar. Luego aceptamos las siguientes distribuciones, instalamos y aceptamos la finalización del programa.

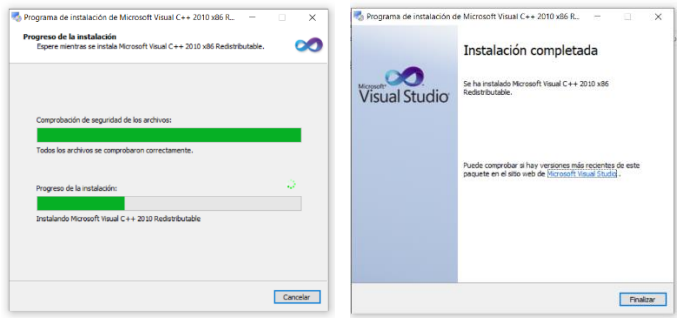


Fig. 3.20 Instalación complemento Visual Studio

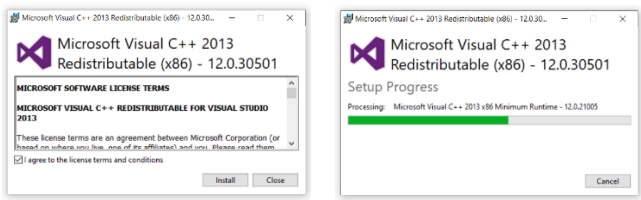


Fig. 3.21 Instalación complemento VS 2013

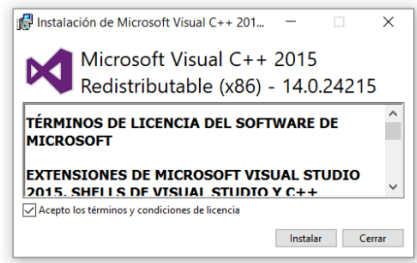


Fig. 3.22 Instalación complemento VS 2015

Seleccionamos la carpeta de destino de instalación, dando clic en siguiente.

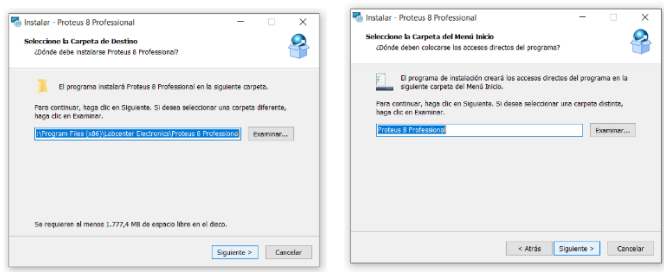


Fig. 3.23 Selección carpeta de instalación

Aparecerá la opción donde indicará la ubicación del icono de acceso directo. Y de esta manera finaliza la instalación de Proteus Design Suite 8.13 para Windows 10

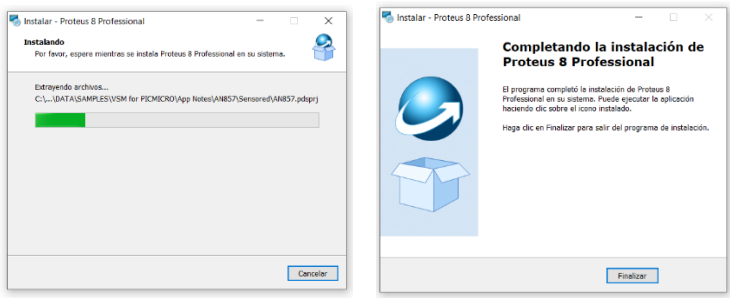


Fig. 3.24 Finalización instalación Proteus 8.13

Damos clic en siguiente y esperamos a proceder a la instalación de Proteus Design Suite.

Una vez finalizada la instalación, vamos al escritorio, seleccionamos el icono de Proteus 8.13 luego le hacemos clic con el botón derecho, en la ventana que se despliega, vamos abajo en propiedades, allí elegimos la solapa "compatibilidad", y seleccionamos el casillero "Ejecutar este programa como administrador".

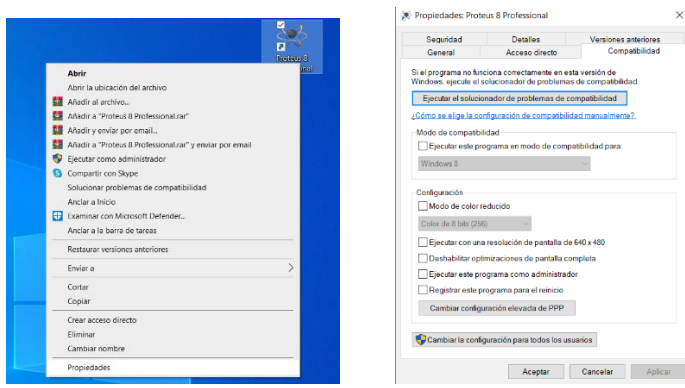


Fig. 3.25 Configuración modo administrador

Electrónica Analógica

La electrónica analógica es una rama de la electrónica que se ocupa de señales continuas qu varían en el tiempo. Se usa ampliamente en equipos de radio y audio junto con otras aplicaciones donde las señales se derivan de sensores analógicos antes de convertirse en señales digitales para su posterior almacenamiento y procesamiento.

Aunque los circuitos digitales se consideran una parte dominante del mundo tecnológico actual, algunos de los componentes más fundamentales de un sistema digital son en realidad de naturaleza analógica.

Me permito tomar este preciso momento, para definir entonces la palabra «Analógico». Analógico significa continuo y real. El mundo en el que vivimos es de naturaleza analógica (temperatura, presión atmosférica, el sonido que escuchamos, etc.), lo que implica que está lleno de posibilidades infinitas. Las personas que trabajan en el campo de la electrónica analógica se ocupan básicamente de dispositivos y circuitos analógicos.

Entonces, mis queridos lectores, he pensado mucho en facilitarles material extra que esta por fuera de la edición de este libro, pero no quiero sesgar sus conocimientos y poder de alguna manera transmitírselos, este material referido a contenido sobre electrónica analógica de otros autores o colegas que tienen en sus canales de YouTube. Para ello les facilito el acceso en los siguientes links:

Electrónica Analógica

Electrónica Analógica

Electrónica Digital

La Electrónica Digital es una parte de la electrónica que se encarga de sistemas electrónicos en los cuales la información está codificada, estudia la manipulación de dígitos binarios en función de administrar procesos automatizados o no y la implementación de circuitos digitales... La electrónica digital es binaria, es decir, cada dígito admite solamente dos posibilidades, que solemos expresar con los símbolos 0 y 1, de forma que el sistema de numeración que le es propio es el sistema de base 2 (binario).

Se diferencia de la electrónica *analógica* en que, para la electrónica *digital* un valor de voltaje codifica uno de estos dos estados, mientras que para la electrónica analógica hay una infinidad de estados de información que codificar según el valor del voltaje.

Esta particularidad permite que, usando **Álgebra Booleana** y un sistema de numeración binario, se puedan realizar complejas operaciones lógicas o aritméticas sobre las señales de entrada, muy costosas de hacer empleando métodos analógicos.

La electrónica digital ha alcanzado una gran importancia debido a que es utilizada para realizar autómatas y por ser el diamante de los sistemas *microprocesados* como son los ordenadores o computadoras, como así también, los sistemas *microcontrolados* que en esta última década a

evolucionado exponencialmente.

Los contenidos que abarcan a la electrónica digital, y van de la mano, con los contenidos que se enseñó en esta edición, pretendiendo poder dar un gran panorama sobre compuertas lógicas, algebra booleana, todos los teoremas que involucran, multiplexores, demultiplexores, codificadores, decodificadores, memorias, flip-flops, etc; voy a compartir material compilado de otros autores que explican y combinan en forma muy practica y sencilla a la electrónica digital, para si de esta forma comprender el avance de los capítulos que explicare más adelante. Por ello, les facilito los links para que disfruten este material:

Con este material complementario, podrán comprender y ampliar sus conocimientos, aquellos lectores que estén interesados y no poseen los conocimientos adecuados, recordando aquellos viejos tiempo, cuando me entusiasmaba conocer pero no podía comprender algunas cuestiones,

dado a la falta de conocimientos técnicos, que por suerte en nuestros días es muy distinto ya que gracias al avance de la tecnología y acceso a los grandes medios podemos lograrlo en muy poco tiempo, de forma práctica y visual con este material valioso.

Luego de dar un paseo por la electrónica analógica y digital, vamos a señalar algunos datos que son relevantes a la hora de tocar los contenidos que estamos arribando con esta edición, como ser la teoría sobre las funciones lógicas básicas.

Compuertas Lógicas

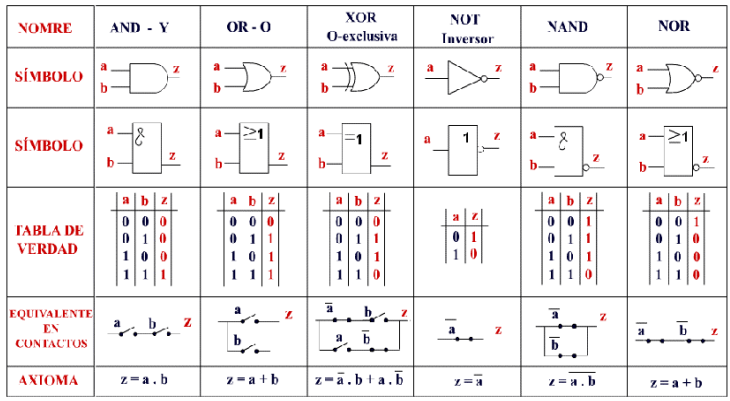


Fig. 3.26 Compuertas lógicas y tablas de verdad

Teorema de algebra booleana

El álgebra de Boole, también llamada álgebra booleana, en matemática, electrónica digital e informática es una estructura algebraica que esquematiza las operaciones lógicas. El teorema de la dualidad expresa lo siguiente: "Cada expresión algebraica deducida de los postulados del álgebra booleana permanece válida si los operadores y elementos identidad se intercambian".

Leyes/Teoremas	Respecto a la suma	Respecto al producto
Conmutativa.	$A + B = B + A$	$A \cdot B = B \cdot A$
Asociativa	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Distributiva	$A \cdot (B + C) = A \cdot B + A \cdot C$	$A + (B \cdot C) = (A \cdot B) + (A \cdot C)$
Cancelación	$(A \cdot B) + A = A$	$(A + B) \cdot A = A$
De Morgan	$\overline{A + B} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$

Fig. 3.27 Teorema de algebra booleana

Axiomas de algebra Boole.

El álgebra de Boole es un método para simplificar los circuitos lógicos (o a veces llamados circuitos de conmutación lógica) en electrónica digital. Estas implicaciones responden a axiomas o postulaciones del algebra Boole. Siendo los siguientes;

- 1a: '0' • '0' = '0'
- 1b: '1' + '1' = '1'
- 2a: '1' • '1' = '1'
- 2b: '0' + '0' = '0'
- 3a: '0' • '1' = '1' • '0' = '0'
- 3b: '1' + '0' = '0' + '1' = '1'
- 4a: si X = '0', entonces $\bar{X}$ = '1'
- 4b: si X = '1', entonces $\bar{X}$ = '0'
- 5a: X • '0' = '0'
- 5b: X + '1' = '1'
- 6a: X • '1' = X
- 6b: X + '0' = X
- 7a: X • X = X
- 7b: X + X = X
- 8a: X • $\bar{X}$ = '0'
- 8b: X + $\bar{X}$ = '1'
- 9 : $\bar{\bar{X}}$ = X
- Teorema de Morgan:
  - 15a: $\overline{(X \cdot Y)} = \bar{X} + \bar{Y}$
  - 15b: $\overline{(X + Y)} = \bar{X} \cdot \bar{Y}$
  - 16a: X + ($\bar{X}$ • Y) = X + Y
  - 16b: X • ($\bar{X}$ + Y) = X • Y
- Propiedad conmutativa:
  - 10a: X • Y = Y • X
  - 10b: X + Y = Y + X
- Propiedad asociativa:
  - 11a: X • (Y • Z) = (Y • X) • Z
  - 11b: X + (Y + Z) = (Y + X) + Z
- Propiedad distributiva:
  - 12a: X • (Y + Z) = (X • Y) + (X • Z)
  - 12b: X + (Y • Z) = (X + Y) • (X + Z)
- Propiedad de absorción:
  - 13a: X + (X • Y) = X
  - 13b: X • (X + Y) = X
- Propiedad de combinación:
  - 14a: (X • Y) + (X • $\bar{Y}$) = X
  - 14b: (X + Y) • (X + $\bar{Y}$) = X

Fig. 3.28 Axiomas de álgebra de Boole

FUNCIÓN: $Z = \bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + \bar{A}.B.\bar{C} + \bar{A}.B.C + A.B.C$

RESULTADO:

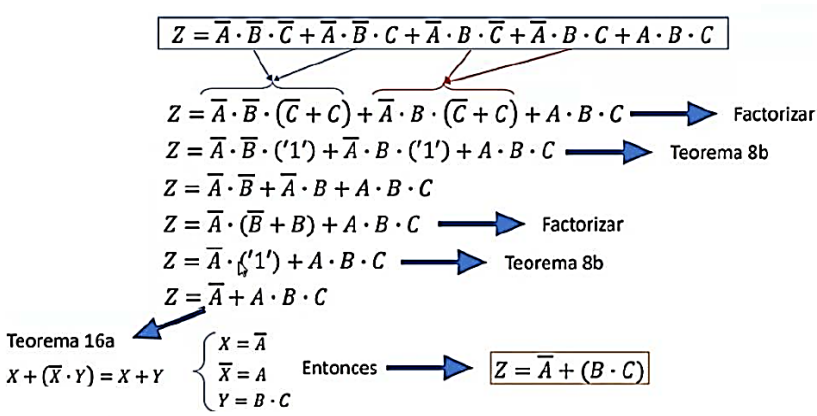


Fig. 3.29 Simplificación por teoremas

Ahora vamos a convertir la ecuación booleana anterior, en una representación esquemática de compuertas lógicas.

$$Z = \bar{A} + (B.C)$$

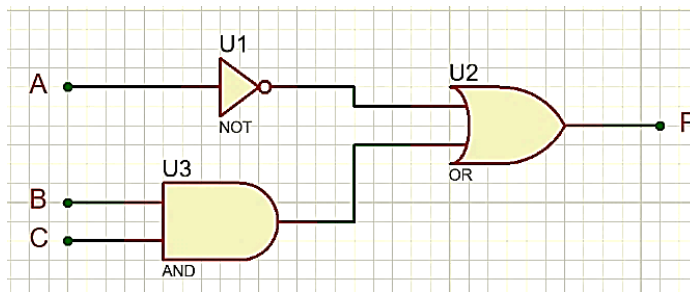


Fig. 3.30 Representación grafica de la ecuación booleana

Pasaremos a armar en Proteus un circuito digital, damos clic en el icono "Modo Componente". Ahora damos clic en el icono "P Devices". Procedemos a buscar las compuertas lógicas necesarias para la función booleana y hacer su representación esquemática en el área de búsqueda

"Keywords". Escribimos "AND", y seleccionamos AND: Simple Two Input AND Gate (Digital Primitive [AND_2]).

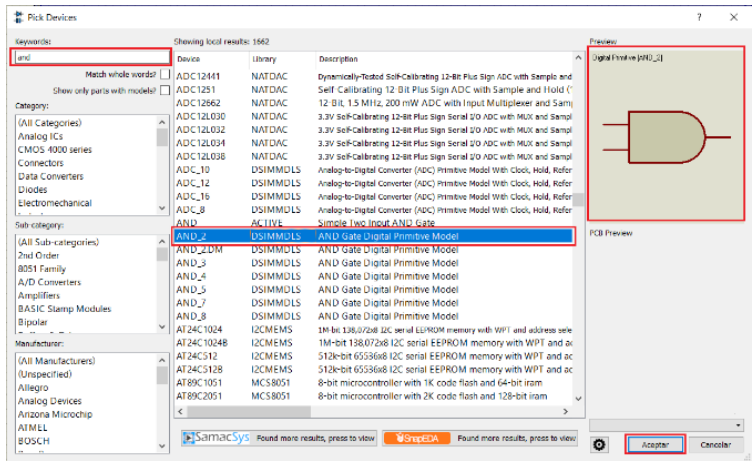


Fig. 3.31 Selección de componente AND

Luego seleccionamos otra compuerta, como figura en la lista de dispositivos OR: Simple Two Input OR Gate (Digital Primitive [OR_2]).

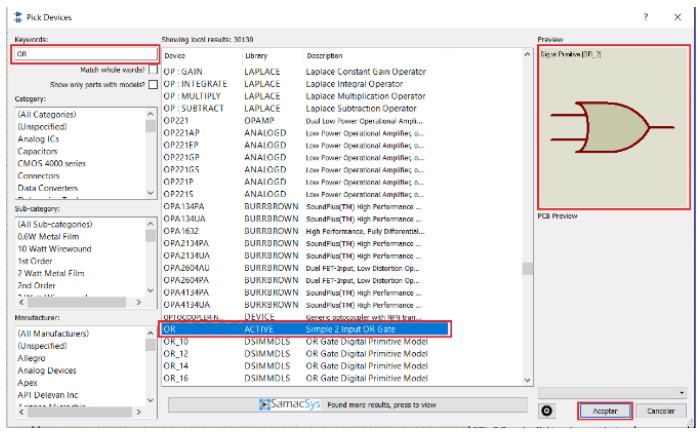


Fig. 3.32 Selección de componente OR

Ahora seleccionamos una compuerta XOR: Simple Two Input XOR Gate (Digital Primitive [XOR_2]).

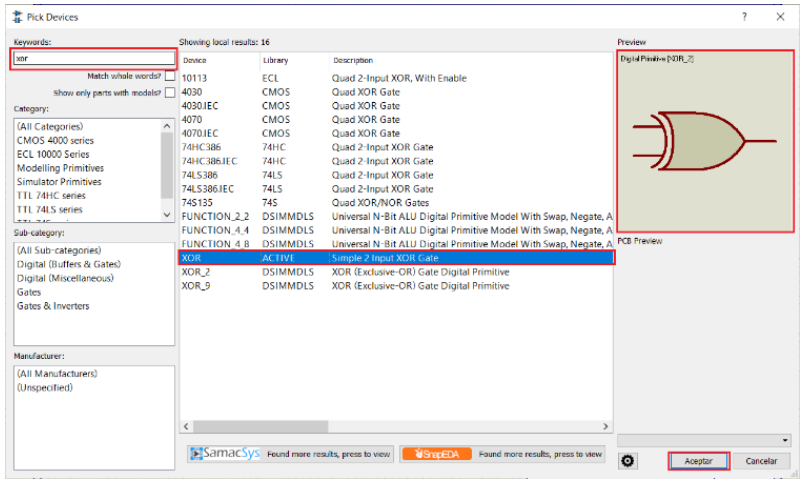


Fig. 3.33 Selección de componente XOR

Seguimos con la selección del componente LED: Animated LED Model (GREEN) Preview (Schematic Model [LEDD]).

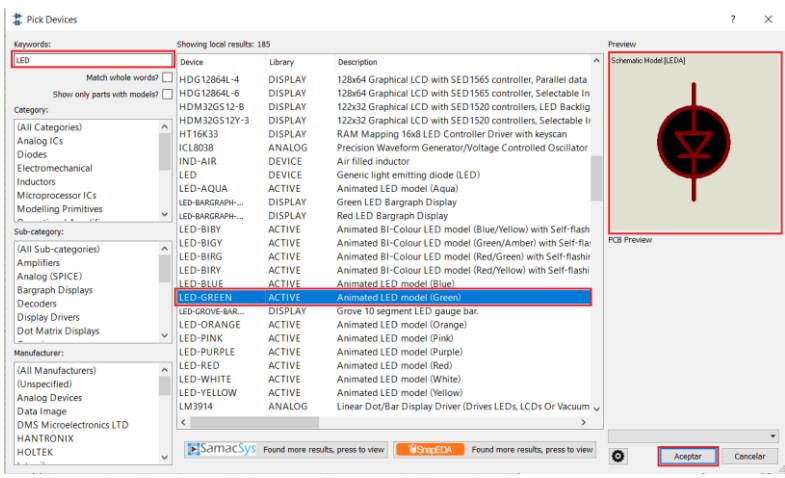


Fig. 3.34 Selección de componente Led Green

Continuamos con una entrada de estado lógico LOGICSTATE: Logic State Source (Latched Action) Preview (Digital rimitive [RTDSTATE]).

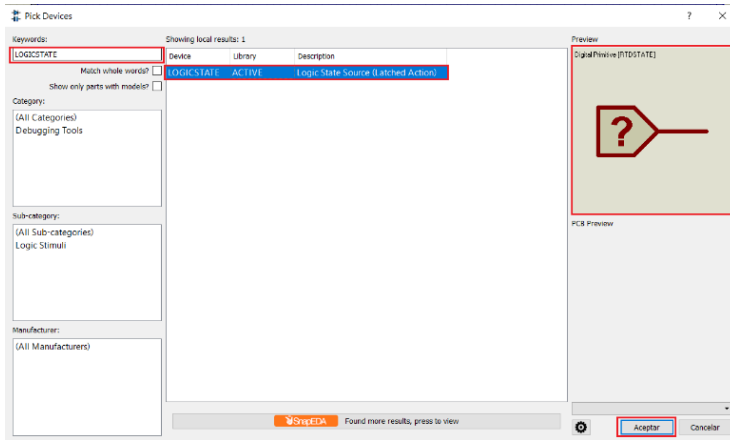


Fig. 3.35 Selección de componente compuerta lógica

Ahora bien, teniendo los siguientes dispositivos: "AND", "OR", "XOR", "LOGISTATE" y un "LED Green", cerramos la ventana de selección de dispositivos (Pick Device). Seleccionamos los dispositivos desde la barra de dispositivos y los arrestamos al área de trabajo de la hoja. Damos clic en LOGICSTATE, ya que será el pulso de estado lógico que ira a las entradas de las compuertas lógicas correspondientes. Vamos posicionando la compuerta AND, que ira conectada cada una de sus entradas a los elementos LOGICSTATE.

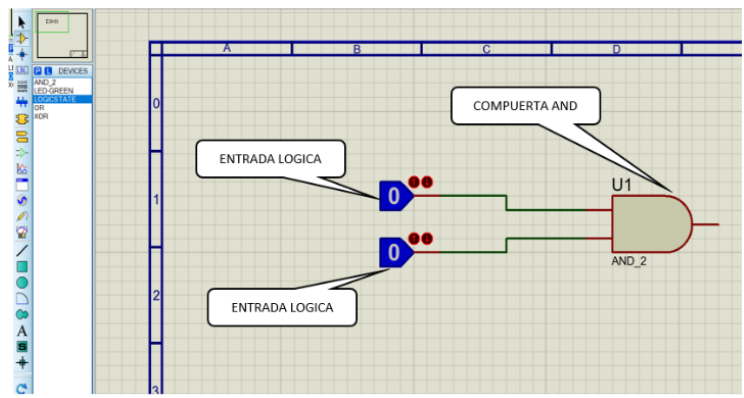


Fig. 3.36 Conexión a compuerta AND

Proseguimos conectando a la compuerta lógica OR, que de igual manera se conectará a cada una de sus entradas de los elementos LOGICSTATE.

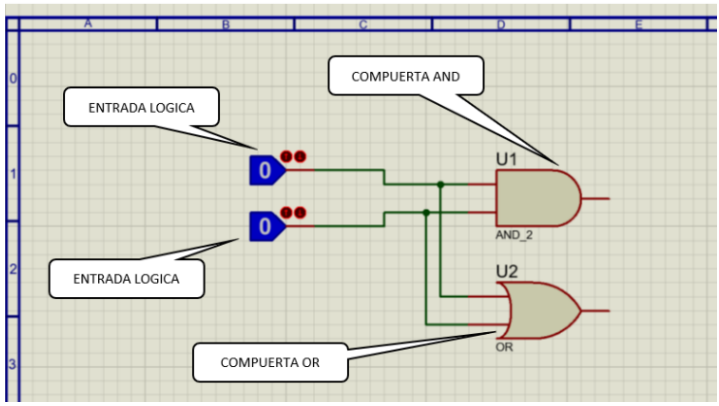


Fig. 3.37 Conexión a compuerta OR

Se procede a conectar cada una de las salidas de las compuertas lógicas AND y OR a las entradas de la compuerta XOR.

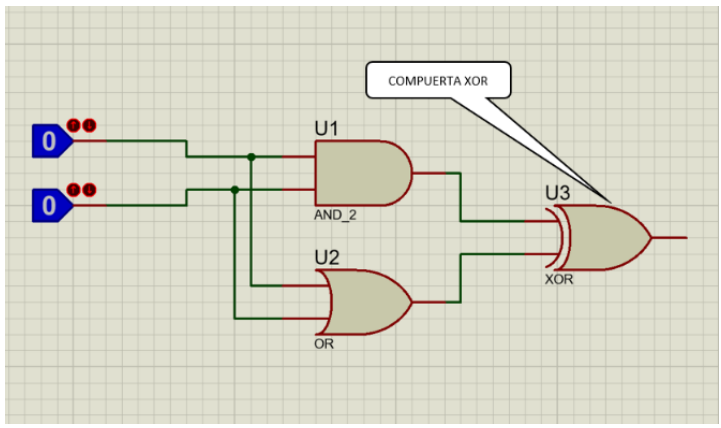


Fig. 3.38 Conexión a compuerta XOR

Para visualizar la salida "Z" de diagrama esquemático del arreglo de compuertas lógicas conectaremos un LED en la salida de la compuerta lógica XOR, $Z = \bar{A} + (B * C)$.

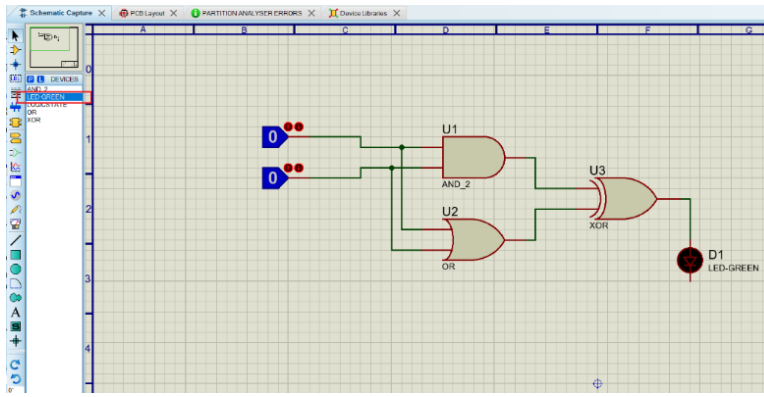


Fig. 3.39 Conexión a Led

Para que el LED encienda al recibir la señal de diagrama esquemático de compuertas lógicas es necesario que cierre circuito al conectar su salida a GROUND, para ello damos clic en el icono de MODO TERMINALES, damos un clic en GROUND y aterrizamos el LED.

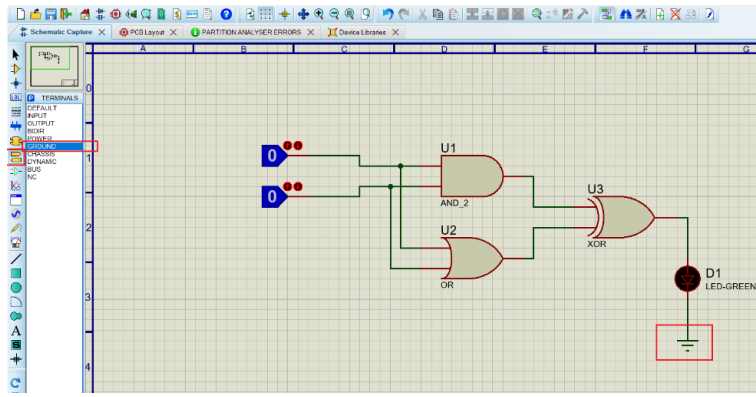


Fig. 3.40 Conexión de Led a puesta a tierra (GND)

Ya podemos decir que esta finalmente terminado y listo para la siguiente fase "SIMULACION", para ello damos clic en el de icono de PLAY (EJECUTAR SIMULACION) que se encuentra en la esquina inferior izquierda.

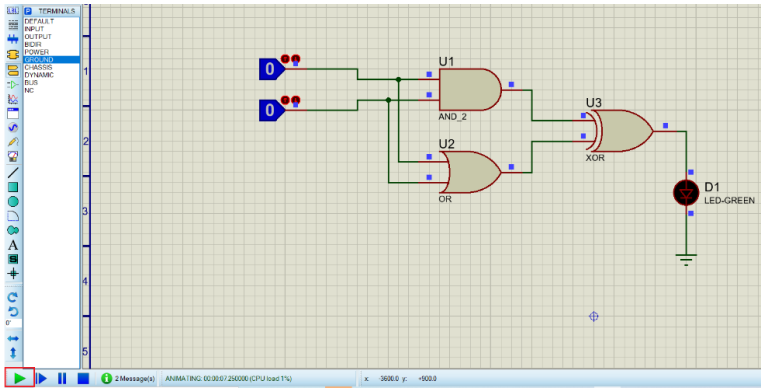


Fig. 3.41 Ejecución de simulación del circuito

En el caso donde el LOGICSTATE A y B estén en 0, la salida será 0. Luego probamos el LOGICSTATE A este en 1 y B esté en 0, la salida será 1, probamos en forma inversa ahora, el LOGICSTATE A esté en 0 y B esté en 1, la salida será 1. Y finalmente si probamos el LOGICSTATE A y B estén en 1, la salida será 0. Podemos ilustrar todos los estados, para armar la tabla de verdad de la compuerta XOR.

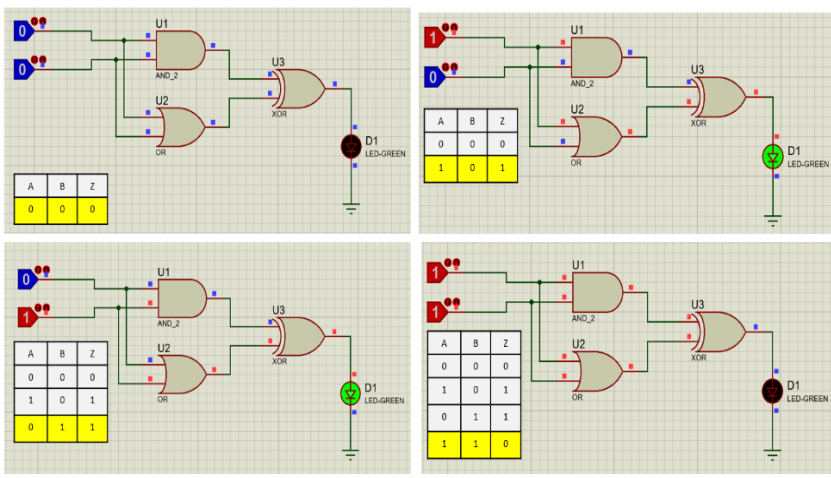


Fig. 3.42 Comprobación de estados y armado de tabla de verdad

De esta manera comprobamos el funcionamiento del circuito lógico, verificando los estados lógicos en la tabla de verdad.

Ares Professional Layout Design.

Incoamos la aplicación Ares, desde la Suite de Proteus por el icono desde la barra de herramientas "PCB Layout".

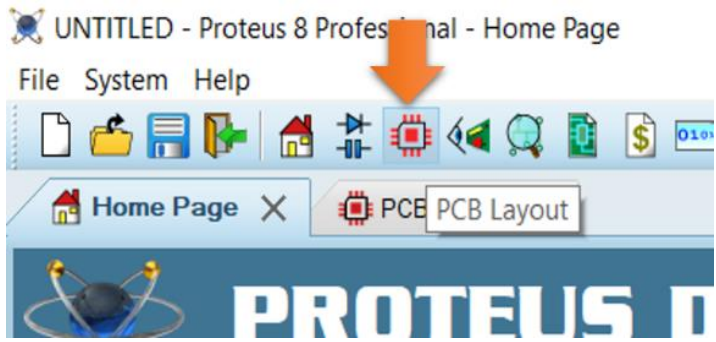


Fig. 3.43 PCB Layout "ARES"

Una vez abierto podemos visualizar la hoja de trabajo, las barras de herramientas y menú como se explicó anteriormente en ISIS.

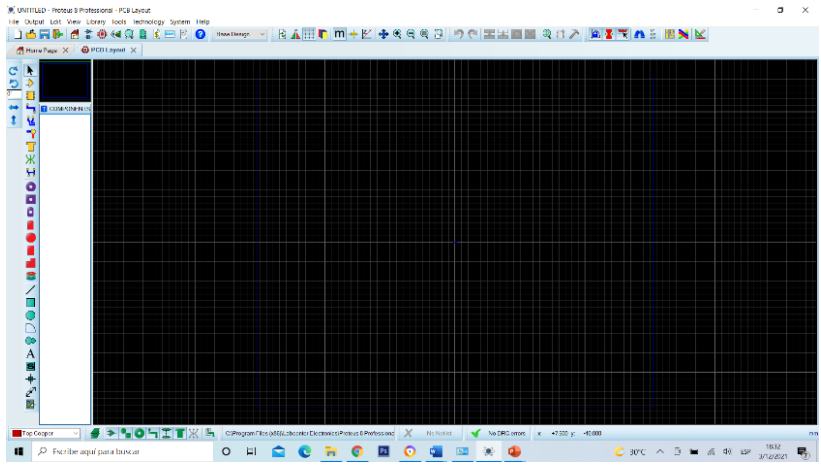


Fig. 3.44 PCB Layout "ARES"

Nos dirigimos a la barra de herramientas ubicada en el lateral izquierdo, allí podemos ver un icono "chips", para poder elegir los componentes de la placa (board).

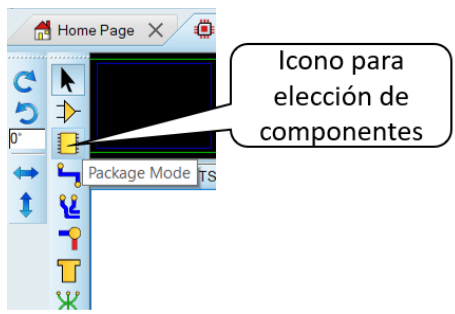


Fig. 3.45 Modo componentes

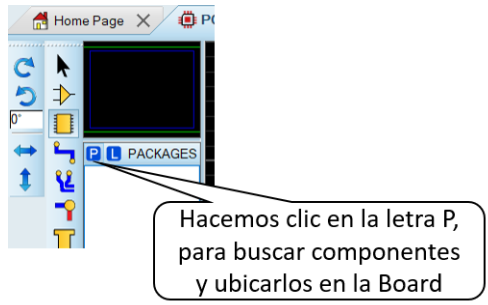


Fig. 3.46 selección de componentes

Escribimos los nombres de los componentes (encapsulados), si deseamos buscar en forma individual.

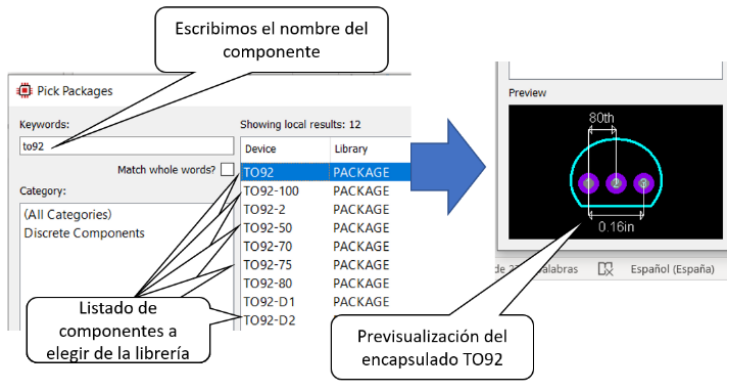


Fig. 3.47 Escribir el componente

Continuando, podrán también modificar el tamaño del grid (espaciamiento entre los puntos de la planilla) para tener más precisión en el posicionamiento de los componentes, así:

Fig. 3.48 Selección de tamaño de grilla

La separación de los puntos que conforman la grilla están en pulgadas o milímetros, se puede optar por cambiar el sistema métrico de medición, de la siguiente forma:

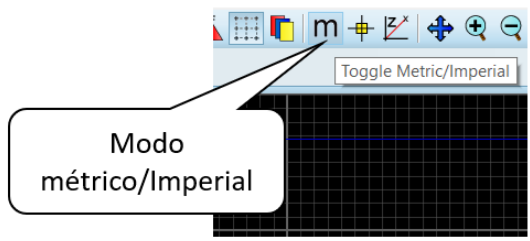


Fig. 3.49 Modo métrico o imperial

Vamos a ver ahora como situar los componentes mediante un sencillo ejemplo:

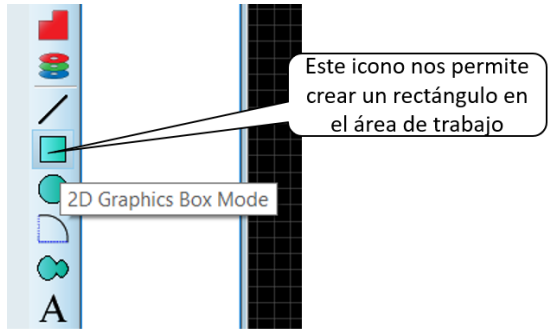


Fig. 3.50 Selección gráfico 2D modo Box

Al seleccionar el icono debe elegirse la opción board edge en el siguiente Combobox, ubicado debajo del área de trabajo inferior izquierdo:

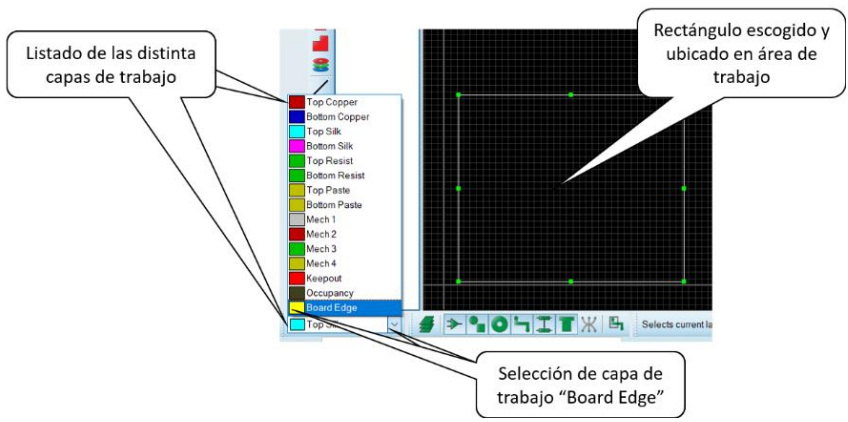


Fig. 3.51 Selección de capa de trabajo "Board Edge"

Una vez hecho esto, se sitúa a los componentes eligiéndolos de la lista del árbol de opciones que está a su izquierda en la pantalla dentro del perímetro elegido.

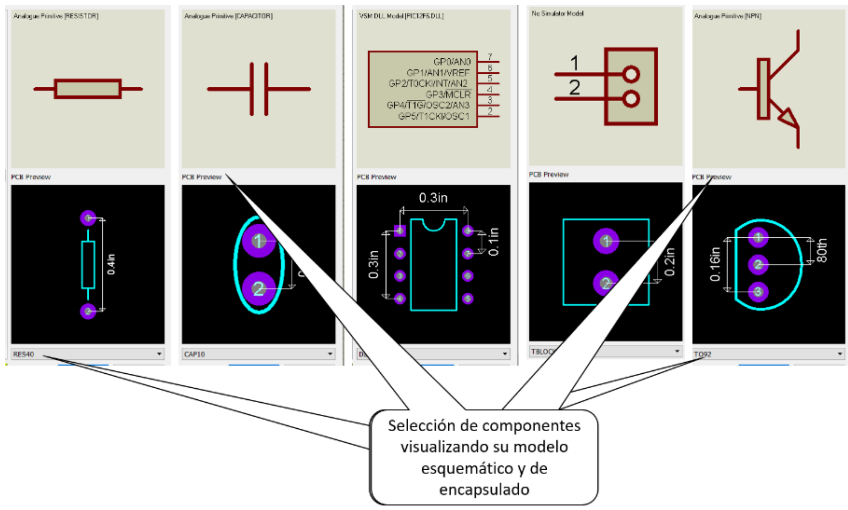


Fig. 3.52 Selección de componentes vista de simbología y encapsulado

De la misma forma que hemos explicado en ISIS, se pueden rotar, duplicar y ubicar los componentes en el área de trabajo.

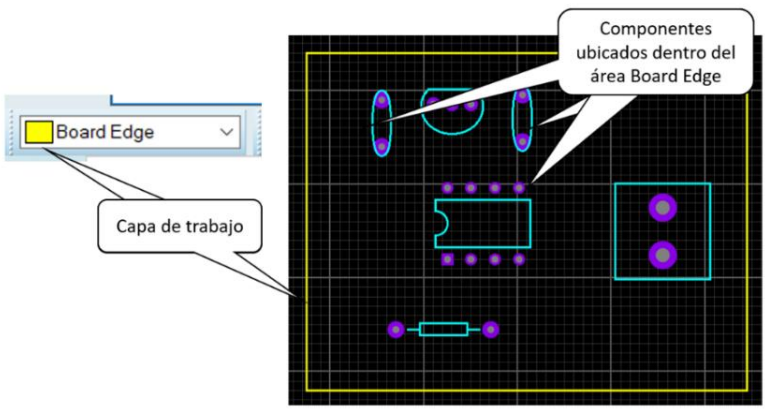


Fig. 3.53 Ubicación de los componentes en área de trabajo

Una vez hecho esto vamos a proceder al unirlos, en el caso que estuviésemos trabajando únicamente en ARES, y no hemos realizado el esquemático en ISIS, sino de lo contrario, las conexiones ya estarían disponibles solo faltaría crearles las rutas de las pistas de cobre.

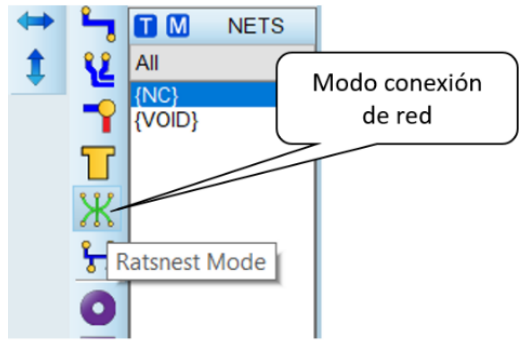


Fig. 3.54 Selección modo conexión de red

Procederemos a seleccionar la opción Bottom Copper (cobre inferior) para efectuar el trazado de las conexiones entre componentes.

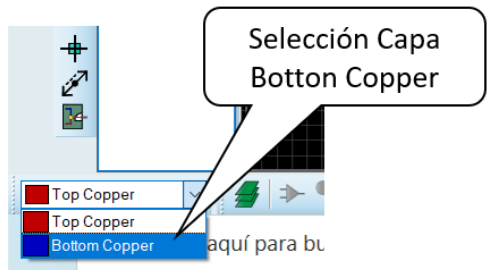


Fig. 3.55 Selección capa Botton Copper

Luego comenzamos a unir los pines de los componentes con las líneas verdes del "rastnest" como corresponda al circuito que se desea implementar.

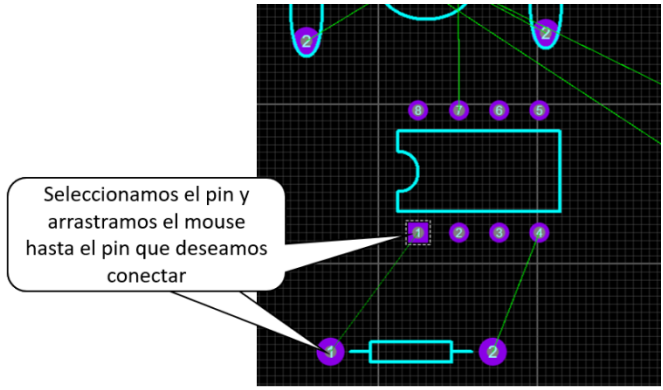


Fig. 3.56 Conexión de pines de componentes en Botton Copper

No importa el orden en el que se encuentran ahora, porque lo que vamos a hacer es situar los componentes de una manera práctica para que el orden nos permita situarlos con el máximo ahorro de espacio dentro de la board; notaran que al mover los elementos las líneas verdes permanecen indicando el sentido de la conexión y que tienden a establecer dicho sentido en función de los nodos en el caso de la unión de más de tres pines de los componentes,; también notarán que al girar los componentes (click derecho rotar) tampoco se altera el sentido de las líneas de conexión.

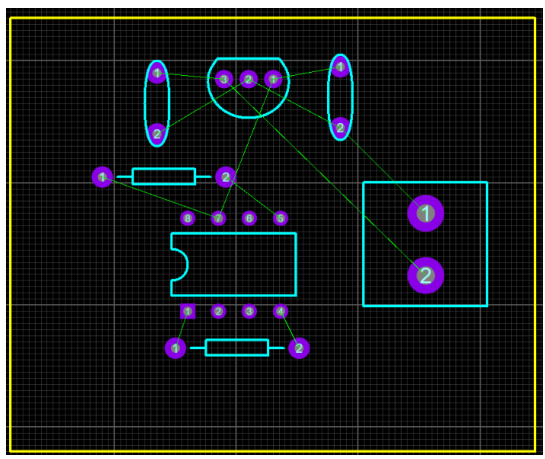


Fig. 3.57 Ubicación de componentes en Botton Copper

Podemos ver como se ha optimizado el espacio para el circuito y como las líneas conservan el sentido, una vez hecho esto se puede proceder a cambiar el tamaño del perímetro de la board y a establecer las conexiones mediante las pistas que harán las conexiones en la board.

Fig. 3.58 Selección Auto-router

Luego, nos aparecerá la siguiente ventana, en Grid debemos elegir el espaciamiento que mejor se adapte al ruteo.

En Routes puede seleccionar que líneas del rastnest quiere rutear

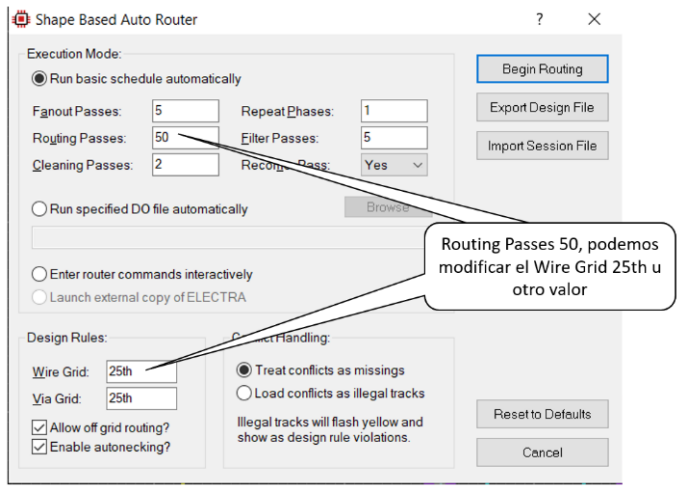


Fig. 3.59 Configuración Auto Router

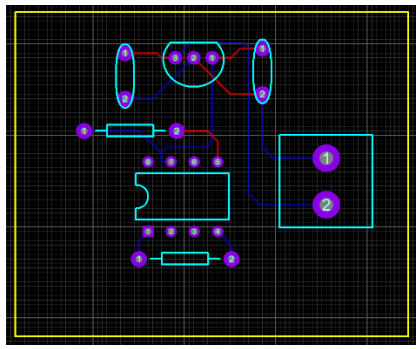


Fig. 3.60 Generación de auto ruteo en Botton Copper y Top Copper lo que no se pudo resolver en la capa seleccionada.

Si la intensión es realizar el circuito en una sola capa, podemos modificar y trasladar las pistas que se encuentran del lado de los componentes (Top Copper) a la del lado del cobre inferior (Botton Copper), con solo seleccionar con el botón derecho del ratón sobre la pista que deseamos cambiar, y en la ventana que nos aparece elegimos la opción: Charger Layer->Botton Copper, y así de esta forma trasladamos la pista de una capa a otra.

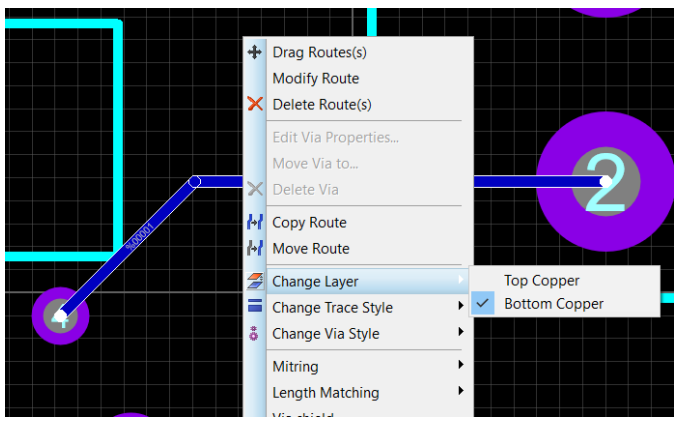


Fig. 3.61 Cambio de pista a Botton Copper

Podemos modificar el ancho de las pistas, haciendo clic derecho sobre la pista que se desea modificar y seleccionamos Charge Trace Style->T40,

para que la pista cambie su espesor del cobre.

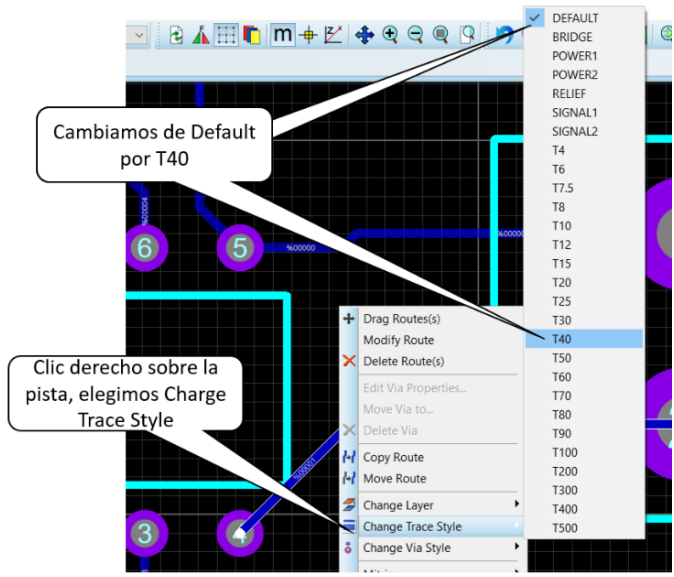


Fig. 3.62 Cambio de espesor de pista

Si deseamos exportar la placa desarrollada a formato PDF, debemos ir a Output->Export Graphic->Export Adobe PDF File. Allí aparece una ventana donde debemos elegir la capa que deseamos imprimir.

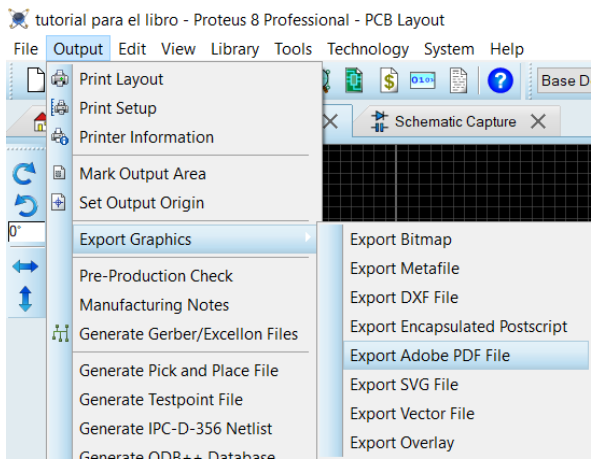


Fig. 3.63 Exportar proyecto a PDF

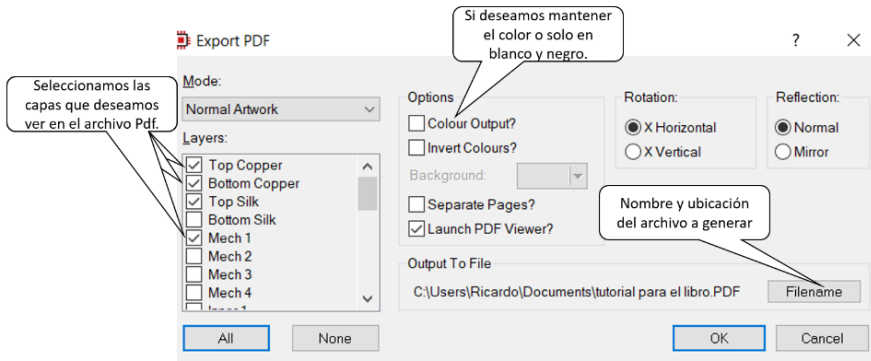


Fig. 3.64 Opciones de exportación de archivo PDF

Si se desea realizar la impresión para ver el tamaño de la placa y ajustar las dimensiones, debemos ir a Output->Print Layout, allí seleccionamos las capas que deseamos imprimir, por ejemplo, Botton Copper y Top Copper. Podemos ubicar la placa en cualquier parte del papel que deseamos que se imprima, por ejemplo, en el centro de la hoja seleccionada.

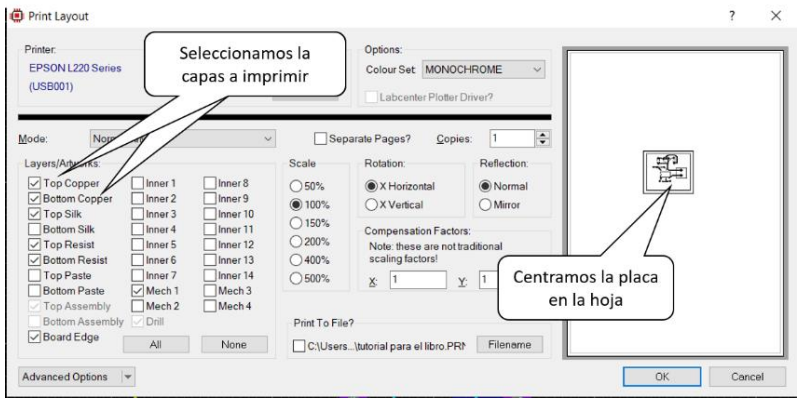


Fig. 3.65 Opciones de impresión del proyecto

Si el propósito del impreso es para un documento puede incrementarse el tamaño del circuito, pero si es para imprimir el papel que llevara las pistas a la placa, el tamaño debe ser del 100%. Como solo se deben Imprimir las pistas se selecciona La opción Bottom Copper y Board Edge; luego sobre otra hoja de papel y seleccionando topsilk, board edge

y mirror se imprimen las siluetas de los componentes.

Para visualizar el proyecto en una simulación en 3D, podemos realizarlo accediendo al icono "3D Visualizer" de la barra de herramientas.

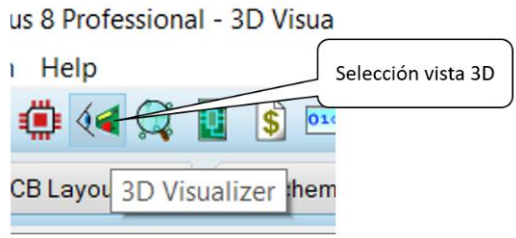


Fig. 3.66 Vista 3D

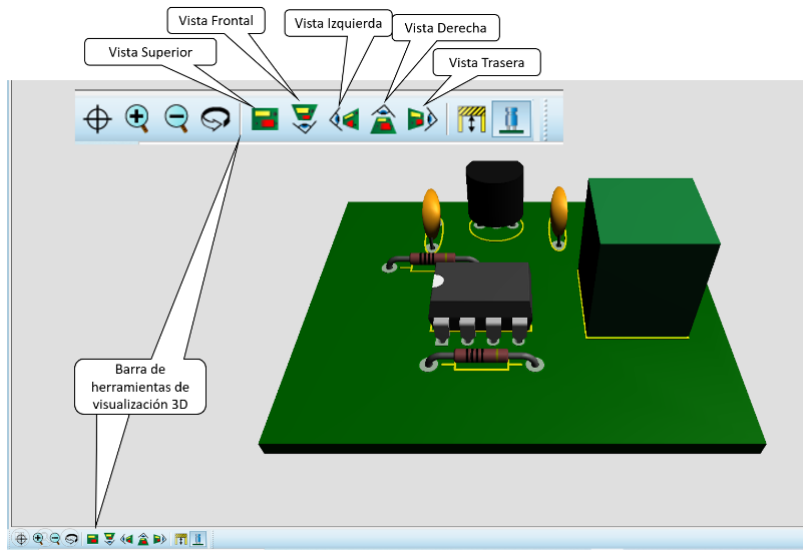
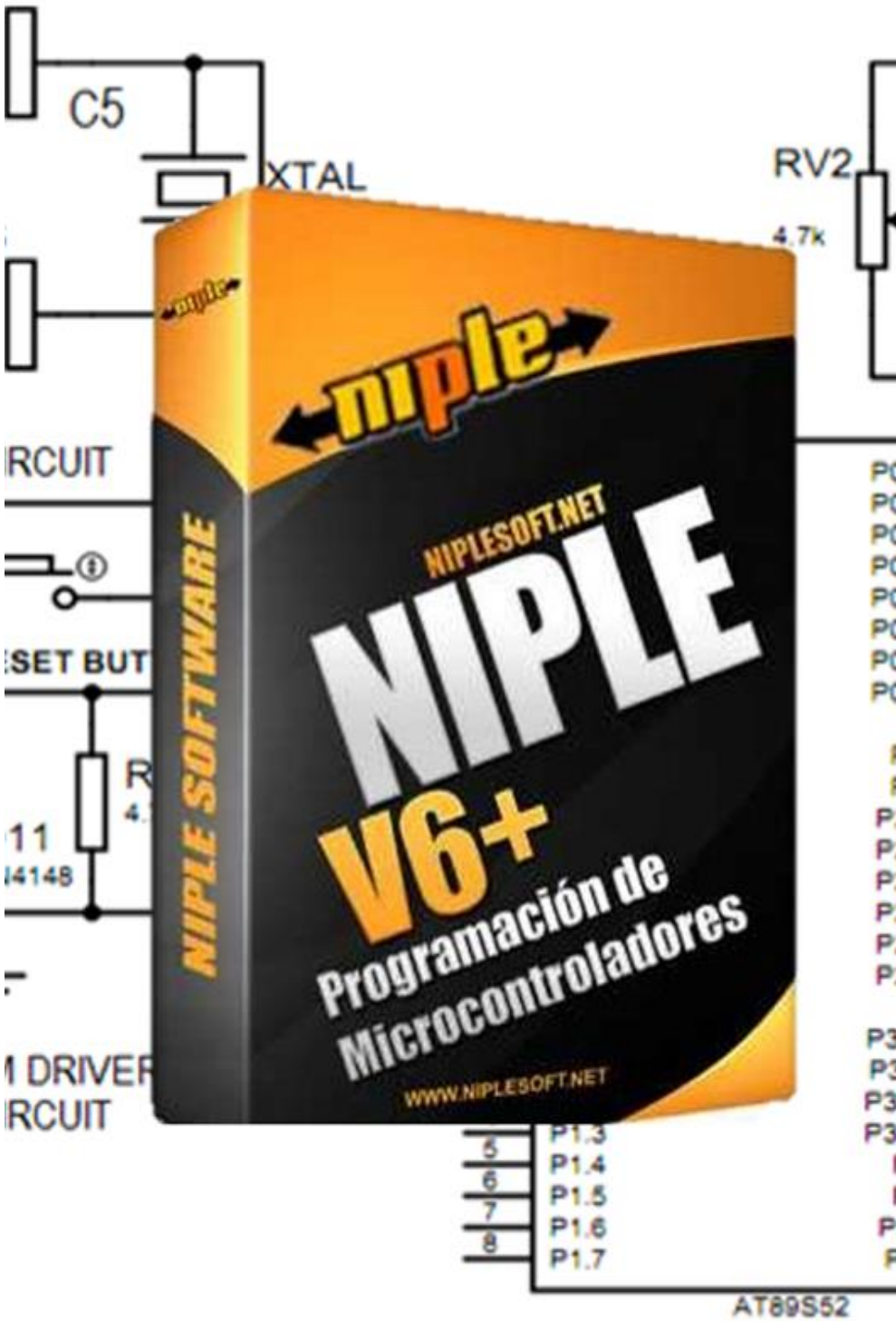


Fig. 3.67 Barra de herramienta para opciones de vista 3D

Para que puedas ejercitar esta demostración, te dare un obsequio, de modo que puedas descargar el poderoso laboratorio virtual o Suite de Proteus 8.11, luego de haberlo descargado, lo podrás instalar siguiendo los pasos que te he explicado anteriormente. Recuerda que la instalación de Proteus es para Windows 7, 8 y 10. No pierdas tiempo, y ponte al día,

para poder dominar y hacer tus propios esquemáticos.

Descarga de Proteus 8.11


C5
XTAL
RV2
4.7k
RCUIT
SET BUT
11
4148
DRIVER
RCUIT
P1.3
P1.4
P1.5
P1.6
P1.7
AT89S52
NIPLESOFT.NET
NIPLE
V6+
Programación de
Microcontroladores
WWW.NIPLESOFT.NET
NIPLE SOFTWARE

Capítulo 4
Metodología del lenguaje visual con Niple

¡Queridos lectores ha llegado el gran momento! Después de haber comprendido como es la lógica de la programación en diagrama de flujo aprendido en el primer capítulo, luego con la presentación de los microcontroladores, su funcionalidad, su estructura y familias, luego enseñarles el funcionamiento del laboratorio virtual o Suite de Proteus, estamos en condiciones de arrancar los motores para comenzar con Niple, y aprender su metodología en la programación visual de microcontroladores PIC.

Antes de comenzar con ello, voy a mostrarle como se realiza la instalación de Niple, como se instala su licencia y cuál es la configuración del sistema para tener preparado el software para empezar a programar.

Instalación de Niple

Vamos a realizar la instalación de Niple en nuestra computadora, para ello, necesitamos ejecutar el instalador que es un archivo.exe, hacemos doble clic al ejecutable. A continuación, se iniciará el asistente que nos guiará paso a paso en este proceso de instalación.

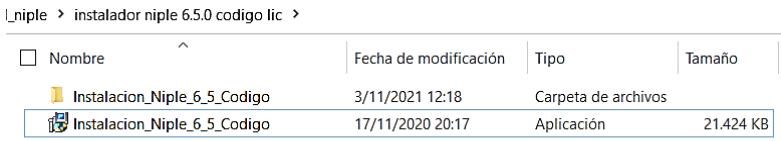


Fig. 4.1 Instalador de Niple 6.5.0

Se abrirá una ventana, dándonos la bienvenida el asistente de instalación de Niple, luego hacemos clic en "Siguiente".

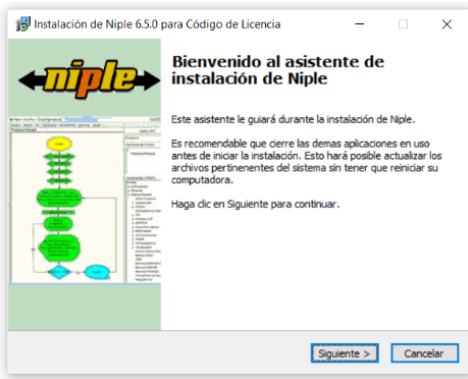


Fig. 4.2 Pantalla de bienvenida

En la próxima pantalla, nos mostrará el acuerdo de licencia, el cual debemos aceptar los términos haciendo clic en "Aceptar".

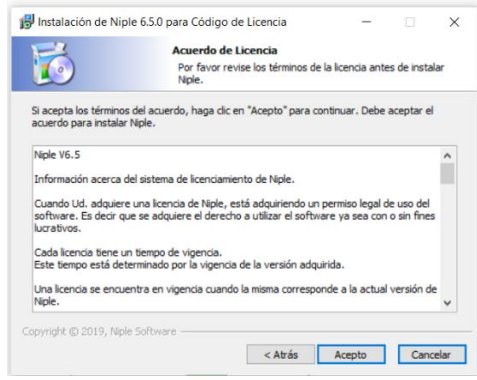


Fig. 4.3 Condiciones de licencia

La instalación nos mostrará la ubicación donde se instalará Niple, para ello, hacemos clic en "Instalar".

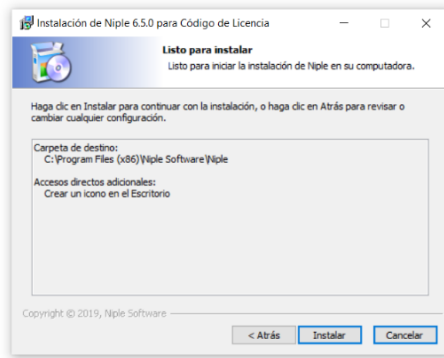


Fig. 4.4 Ubicación de instalación

Comienza la instalación de todos los componentes de Niple. Una vez que el instalador termine de copiar todos los archivos presionamos el botón de "Finalizar".

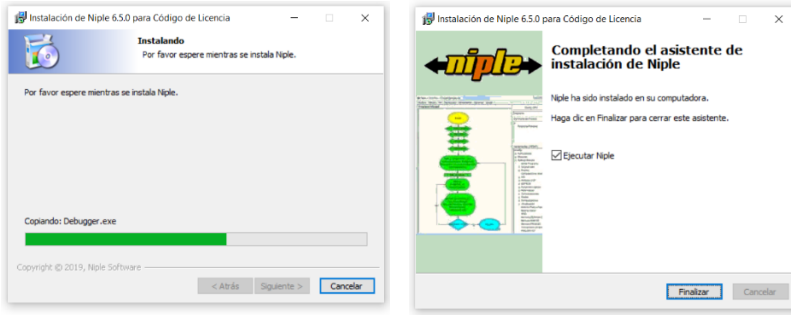


Fig. 4.5 Porcentaje de instalación y finalización

Ahora podemos ejecutar Niple desde el icono que se nos creó en nuestro escritorio, o desde el buscador de la barra de inicio.

Fig. 4.6 Icono de Niple

Instalación Licencia de Niple

La primera vez que ejecutemos el programa, nos aparecerá una ventana con el código de licencia del programa.

Fig. 4.7 Inicializando Niple por primera vez

Podemos visualizar que nos muestra un Código de Licencia alfanumérico, este código lo copiamos, para luego enviárselo al equipo de NipleSoft, quienes nos mandaran el archivo de licencia para nuestro ordenador.

Fig. 4.8 Código de licencia de la PC

En el siguiente paso, lo que debemos hacer es mandar un correo electrónico a comercial@niple.com, donde solicitan el envió de la licencia para su computadora, copiando y pegando en el correo el código de licencia que figura en pantalla, y así de esta manera, el equipo de NipleSoft les enviaran el archivo de licencia para poder utilizar Niple. Una vez recibido el correo como se ilustra en la siguiente imagen, debemos bajar de dropbox.com el archivo enviado por NipleSoft, para luego poder ejecutarlo.

Instalando la Licencia de Niple

Una vez recibido por correo electrónico la licencia de Niple perteneciente a nuestro usuario registrado, iniciamos el proceso de su instalación. Para ello, debemos bajar la licencia del link que recibimos por correo alojado en Dropbox.com.

Fig. 4.9 Porcentaje de instalación y finalización

Hacemos clic en el enlace para ingresar a Dropbox.com, allí nos dirigimos a donde indica la flecha verde, para poder descargar el archivo.

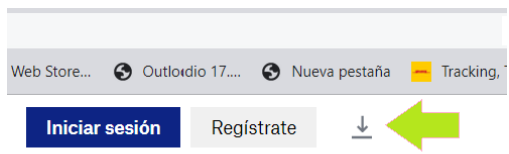


Fig. 4.10 Descarga de archivo de licencia de Dropbox

Posterior a ello, elegimos la opción de "Descarga directa", hacemos clic para descargar el archivo de licencia.

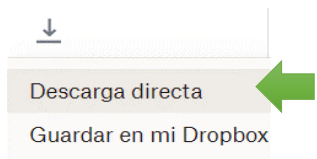


Fig. 4.11 Descarga directa del código de licencia

Finalizada la descarga, vamos a la carpeta donde se encuentra la licencia, hacemos doble clic para ejecutar la instalación de la licencia en nuestra

computadora, de esta forma, ya nos quedó instalado el programa listo para usarse. Cualquier inconveniente o problema en algunos de los procesos de instalación, pueden consultar al equipo de Niplesoft, quienes los asesoraran de manera inmediata.

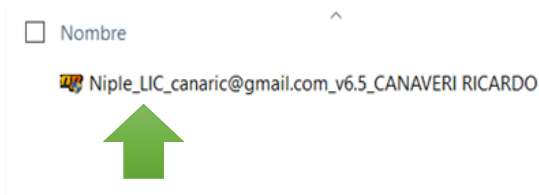


Fig. 4.12 Archivo de licencia de Niple

Al ejecutar el programa, podemos ver en la barra de menú, en la opción Ayuda>Acerca de, visualizamos que se instaló la licencia en forma correcta.

Fig. 4.13 Visualización acerca de la versión de Niple 6.5.0

Configuración del Sistema

Abrimos Niple, en la barra de menú en la opción "Configuración de Sistema", accedemos para realizar algunas configuraciones. En la ventana configuración de sistema, vamos a tener que elegir los directorios donde

se guardaran nuestros archivos.

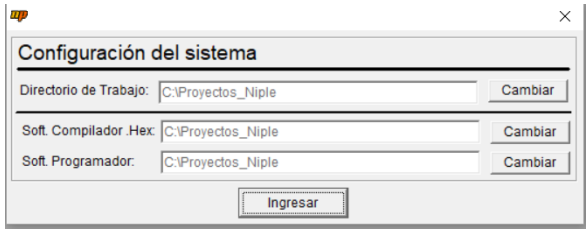


Fig. 4.14 Pantalla de configuración de sistema

Por defecto, se creó un directorio en "C:\Proyectos_Niple", en caso de que el programador desee ubicarlos en otro lugar puede optar por cambiarlos. Hay que tener en cuenta que las rutas elegidas no deben tener espacios entre sus nombres, lo recomendable es agregar un guion bajo para separar las palabras.

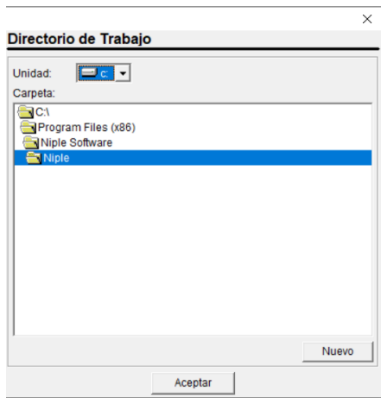


Fig. 4.15 Ubicación de directorio de trabajo

Del mismo modo, para el caso que se cambiar la ubicación de los archivos del software compilador y Software programador.

Metodología de Programación Visual con Niple

En este apartado, comenzare explicando cual es la metodología empleada por Niple en su programación, como lo menciona en el manual de Niple, emplea el diseño de un diagrama de flujo de manera totalmente visual y de alto nivel de programación.

Lo importante a estacar, es que el usuario no debe escribir código en *Assembler* adaptándose al "lenguaje" del micro, sino que, por el contrario, el usuario efectúa el diseño del programa de manera gráfica, seleccionando opciones a través de pantallas claras e intuitivas, y donde el sistema se adapta al "lenguaje humano" y luego "traduce" a código *Assembler*.

El *diagrama de flujo* está formado por un conjunto de bloques "estandarizados". Tanto la forma, como el color y el texto de cada bloque, le están presentando información y datos concretos acerca de cada parte del proceso, lo cual facilita aún más la interpretación del programa.

También, el diagrama de flujo le permite obtener una "visión general" del proyecto.

El proceso de diseño del diagrama de flujo consiste en insertar bloques y vincularlos por medio de flechas.

En el transcurso de éste proceso, Niple controla y advierte de posibles errores, tanto de la lógica del programa como en la configuración del sistema (configuración de puertos, interrupciones, etc.).

En el margen derecho de la pantalla se encuentra el "Árbol de Herramientas". Este consiste en una estructura de árbol donde se desglosan y encuentran todas las funciones que pueden ser insertadas como bloques en el diagrama de flujo, que luego, forman parte del código Assembler generado.

Los bloques se encuentran agrupados en distintos niveles organizados por *categoría* para facilitar la búsqueda de un bloque en particular.

Se mantiene un nivel organizacional por tipo de funciones, de tal forma, que en su primer se encuentran los siguientes ítems:

Instrucciones:

Categoría que incluye todas instrucciones en Assembler. Por otro lado, podemos visualizar una organización por tipo de instrucción:

- Instrucciones orientadas a Registros
- Instrucciones orientadas a Bits
- Instrucciones orientadas a Literales
- Instrucciones orientadas a Control
- Todas las instrucciones ordenadas alfabéticamente

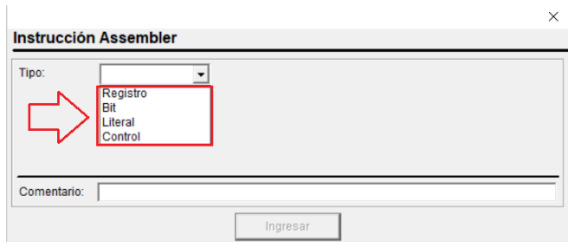


Fig. 4.16 Instrucciones en Assembler

El objetivo de Niple es utilizar lo menos posible los bloques de instrucciones, ya que estos bloques equivalen a programar en código Assembler.

En el segundo nivel se organizan las etiquetas del proyecto, es decir, permite la creación o el salto de etiquetas dentro del diagrama de flujo.

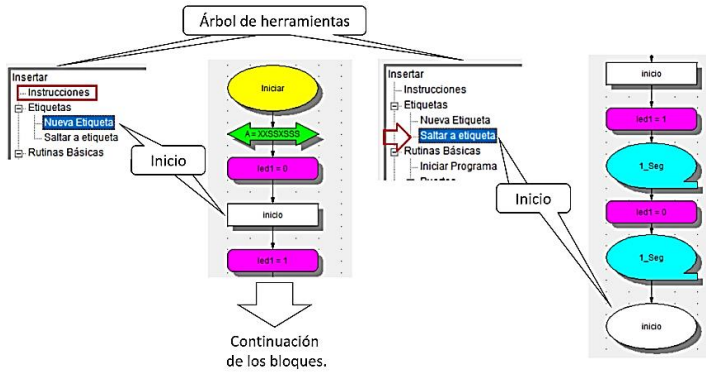


Fig. 4.17 Árbol de herramientas

Rutinas Básicas:

Las rutinas básicas son módulos prediseñados y se encargan de resolver un proceso en particular (una configuración, una comparación, un cálculo, etc.). El lector puede formar estructuras lógicas muy complejas de manera rápida y sencilla a partir de la combinación de rutinas básicas.

Puede utilizar las rutinas básicas para programar dentro de cualquier "Entorno".

Las rutinas básicas están organizadas en categorías como se muestra en la imagen a continuación:

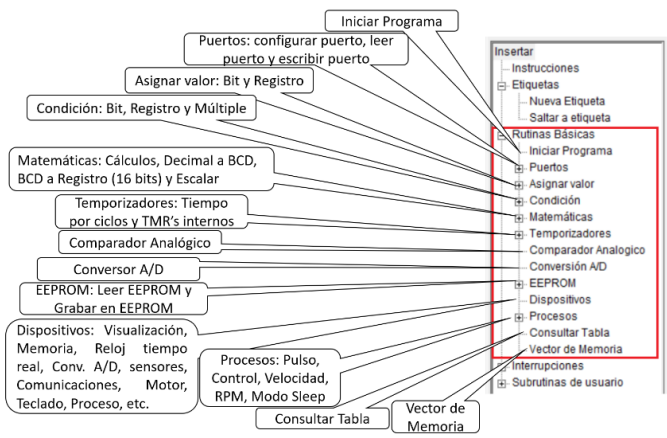


Fig. 4.18 Rutinas básicas

Interrupciones:

Esta categoría contiene todas las funciones de configuración y manejo de las interrupciones. Niple presenta las necesidades disponibles de acuerdo al modelo de PIC que se está utilizando.

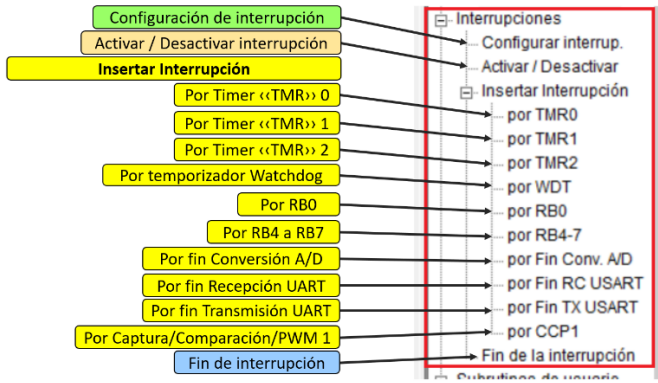


Fig. 4.19 Tipos de Interrupciones

Subrutinas de usuario:

Aquí Niple le propone al usuario "*subrutinas de usuarios",* estas son para diseñar sus propias funciones o procedimientos para solucionar un problema en particular (siempre combinando distintas rutinas básicas) y luego llamarlos desde distintas partes del programa (subrutinas).

Permitiéndole, por un lado, al programador, la creación de funciones o procedimientos personalizados y, por otro lado, la optimización del código *Assembler* generado, dado que, las rutinas de usuario se escriben una vez y luego se insertan "llamadas" a las mismas. Incluyéndose en esta categoría las funciones de declaración e inserción de llamadas a rutinas de usuario.

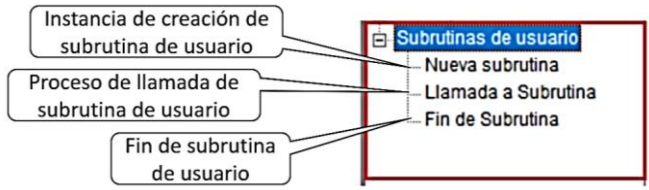


Fig. 4.20 Subrutinas de usuario

Secciones de trabajo

Niple dispone de 2 secciones de trabajo:

- *Sección de Dispositivos*: se encuentran los dispositivos externos que serán utilizados en el proyecto.
- *Sección de Diagrama de Flujo*: se encuentran los diagramas de flujo que conforman el proyecto actual.

Para conmutar entre las secciones de Dispositivos y Diagramas de Flujo, utilice la lista desplegable que se encuentra en la parte superior derecha de la pantalla:

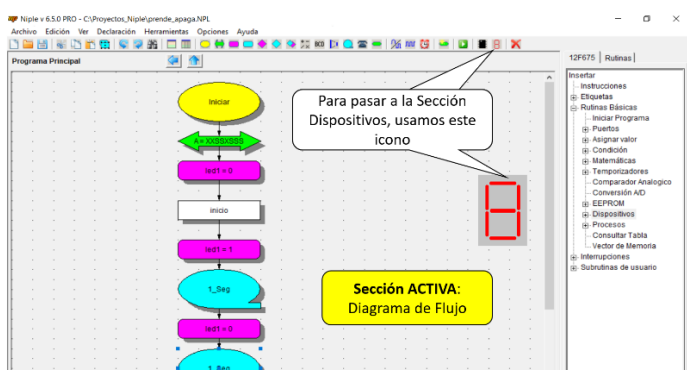


Fig. 4.21 Sección Diagrama de Flujo

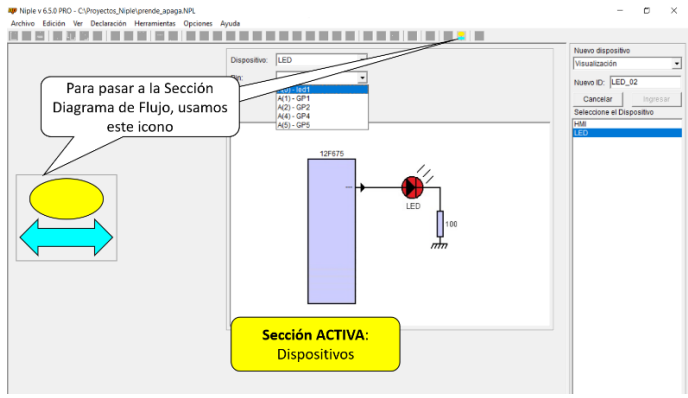


Fig. 4.22 Sección Dispositivos

Al ingresar a la sección Dispositivos puede declarar los dispositivos externos que serán utilizados en el proyecto.

Al seleccionar el tipo de dispositivo a declarar se muestran los diferentes componentes disponibles según el modelo de microcontrolador utilizado.

Cada dispositivo es identificado con un nombre ("ID", por Identificación). Cada dispositivo debe tener un nombre único que lo identifique dentro del proyecto, es decir que no puede haber más de 1 dispositivo con el mismo ID.

Por defecto Niple le propone un Id de manera automática, Ud. Puede cambiar el Id para identificar más fácilmente cada dispositivo dentro del diagrama de flujo.

Una categoría de dispositivos puede estar formado por varios componentes. En este caso se muestra una lista de los componentes disponibles en la parte inferior de la pantalla.

Al seleccionar el componente deseado se muestra la pantalla de configuración y opciones correspondiente.

En la siguiente imagen se muestra un dispositivo de tipo "Memoria", las opciones disponibles en la parte inferior, la pantalla de configuración al seleccionar un dispositivo y el ID asignado por defecto.

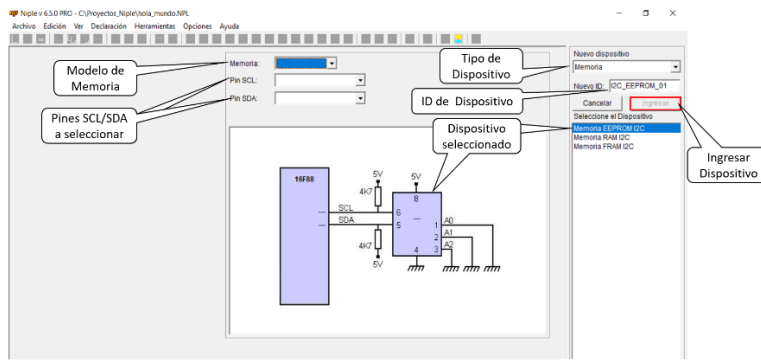


Fig. 4.23 Dispositivo memoria EEPROM I2C

Una vez seleccionadas las características del dispositivo y la conexión de los pines al PIC se termina el proceso de declaración de dispositivo por medio del botón "Ingresar". Este botón estará disponible solo cuando todas las opciones de configuración del dispositivo hayan sido seleccionadas.

En la parte inferior de la pantalla se muestra la lista de dispositivos declarados en el proyecto actual. Al seleccionar un dispositivo se muestran todas las características y opciones de configuración.

Los dispositivos se muestran por su ID. En la siguiente ilustración se visualiza 2 dispositivos de tipo Memoria identificada como "IC2_EEPROM_01" y "IC2_RAM_01".

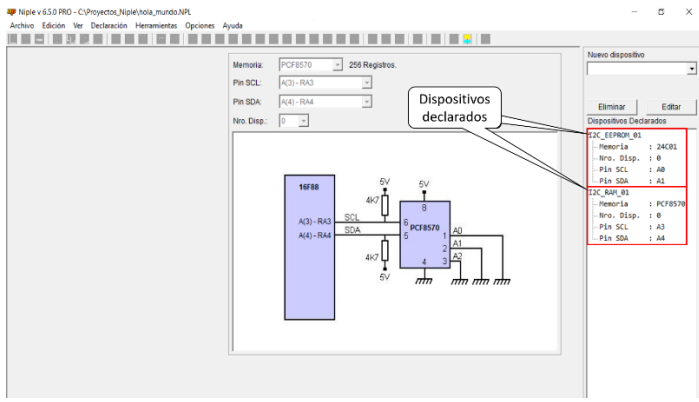


Fig. 4.24 Declaración de 2 dispositivos Memoria EEPROM IC2 (24C01) y Memoria RAM IC2(PCF8570).

Se puedo notar una fácil identificación de los dispositivos a emplear en una implementación, de acuerdo a las funciones que cumplirían cada uno, los cuales están identificados por un ID, permitiéndole la rápida programación dentro del diagrama de flujo ya que se accederá a estas memorias por su ID.

A continuación, se puede visualizar como se utiliza el ID para acceder a cada una de las memorias dentro del diagrama de flujo.

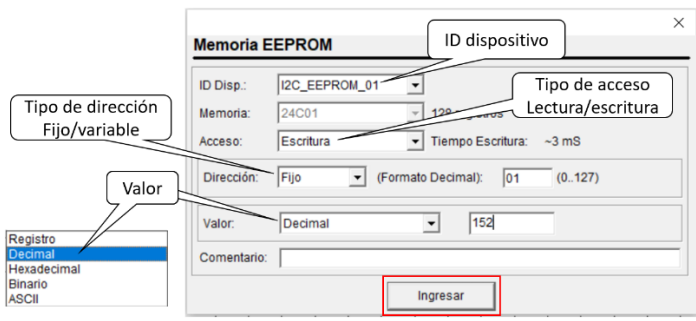


Fig. 4.25 Configuración Memoria EEPROM

En la sección Diagrama de Flujo Ud. programa los procesos o estructuras

lógicas que desea realizar en el proyecto.

Un proyecto esta dividió en diferentes "Partes". A cada una es tas partes se las llama "*Entorno*".

Entornos

Un microcontrolador PIC, en general su programa está organizado en distintas "secciones" con la finalidad de facilitar la compresión al usuario y al mismo tiempo, lograr la optimización del código por medio de la reutilización de rutinas y procesos.

Cada una de estas "secciones", tiene por objetivo resolver un problema en particular, y reciben distintos nombres según la función que cumplen, o el contexto en el cual se ejecutan "Programa Principal", "Subrutinas" e "Interrupciones".

Niple llama "Entorno" a cada una de estas "*Secciones*" que conforman un programa.

El *Entorno* esta de manera perfecta y delimitado por un bloque de "Inicio" y otro de "Fin", con la excepción del entorno del "Programa Principal". Este contiene un bloque de "Inicio" pero no tiene un bloque de final, sino que, se realiza un bucle para ejecutar otro "ciclo" del programa.

Cada proyecto de Niple puede contener varios entornos;

- Entorno del "Programa Principal": éste entorno se crea de manera automática y es el encargado de "coordinar" la ejecución de todo el programa (incluyendo el resto de los entornos). En un

proyecto sólo puede haber un entorno llamado "Programa Principal" y no puede ser borrado.

- Entornos de Subrutinas (rutinas de usuario): Son funciones o procedimientos generados por el usuario. Se le debe asignar un nombre único dentro del proyecto. No tiene un límite máximo en la cantidad de entornos de subrutina que puede incluir en un proyecto. Las subrutinas pueden devolver o no un resultado (Funciones y Procedimientos respectivamente). Para ejecutar una subrutina debe ser "llamada" o invocada.

- Entornos de Interrupciones: Son funciones o procedimientos que se ejecutan al producirse un llamado a interrupción. Cada interrupción tiene un nombre predeterminado y el usuario no puede cambiar éste nombre. La cantidad máxima de "Interrupción" corresponde al número de interrupciones de incluye cada modelo de microcontrolador. En tiempo de ejecución, se accede al entorno de una interrupción de manera automática, al generarse un llamado a interrupción. O sea que no se puede "*Predecir*" el momento exacto en que se producirá una interrupción. Como consecuencia de ello, afecta al sistema de chequeo de errores.

Crear un nuevo Entorno

Un entorno se genera de manera automática al insertar interrupciones o rutinas de usuario en el proyecto. Cuando inserta una interrupción en el proyecto, Niple coloca de manera automática el bloque de inicio de entorno con el texto correspondiente.

Al momento de declarar una *Rutina de Usuario*, Niple genera un nuevo entorno con el nombre que el programador le asignó a la rutina y colocando el prefijo "usr_" (abreviación de la palabra usuario). Inserta de manera automática el bloque de "Inicio de Entorno".

Seleccionar el Entorno actual de trabajo

Para conmutar entre los distintos entornos, se puede realizar explorando el árbol de herramientas ubicado en el lateral derecho, accediendo a la solapa "rutinas", allí podemos encontrar los distintos entornos creados e incluyendo el entorno principal.

En la esquina superior izquierda del área de trabajo se muestra el nombre del entorno actual.

Borrar un Entorno

Para borrar un entorno, seleccione el nombre del entorno a borrar y presione la tecla "Suprimir", nos aparecerá una ventana emergente donde Niple nos pregunta si deseamos borrar el entorno, haciendo clic en "SI", en el caso que se lo desee borrar.

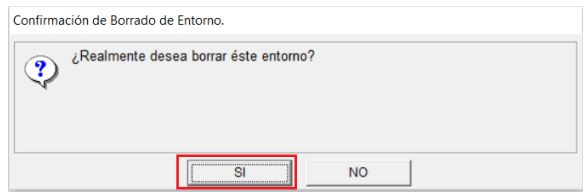


Fig. 4.26 Borrar un entorno

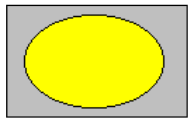
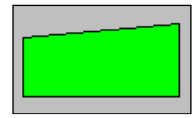
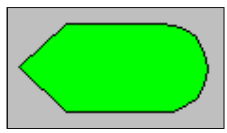
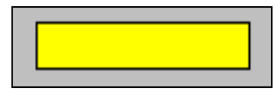
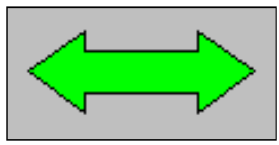
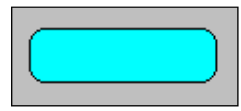
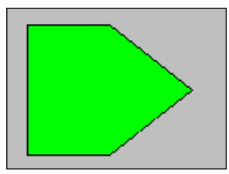
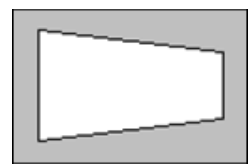
Bloques

Un bloque es un símbolo que representa un proceso (un paso o varios). Cada bloque está representado por su forma según la información, por un color determinado y texto que contiene, e identifican una función determinada.

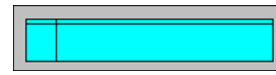
Definición de cada Bloque

Cada forma de bloque está directamente asociada a un tipo de

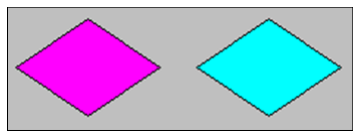
función, esto le facilita la interpretación del programa con sólo echarle un vistazo general:

 Inicio – Fin. Se utiliza como inicio o fin de un entorno	 Ingreso de datos. Leer el valor de un puerto (en paralelo), modo Captura, modo Comparación.
 Salida de datos. Escribir un valor en un puerto (en paralelo, no serial)	 Instrucción. Cualquiera de las 35 instrucciones en código Assember.
 Configuración de un Puerto.	 Asignación. Este bloque representa la asignación de un valor a un registro o un bit.
 Conversión Decimal a BCD ó BCD a Decimal.	 Leer un dato de la memoria EEPROM.

Almacenar un dato en la memoria EEPROM

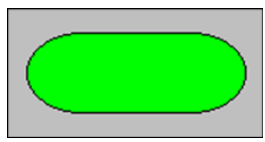


Consultar Tabla.

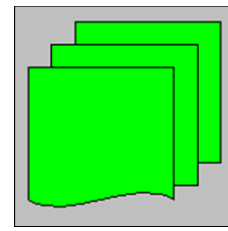


Condición. Evalúa una condición y deriva la ejecución del programa en función del resultado de la evaluación

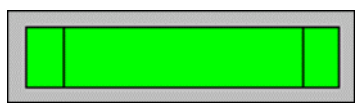
Comunicaciones Seriales RS232. Envío y recepción de un dato por RS232.



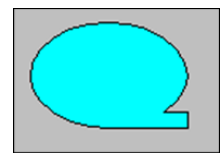
Configuración. Este bloque corresponde a configuración de Interrupción y módulos periféricos



Comunicaciones Seriales RS232. Envío y recepción de un conjunto de datos por RS232 mediante protocolo de usuario diseñado en Niple.



Activar – Desactivar Interrupciones



Temporización por bucle.

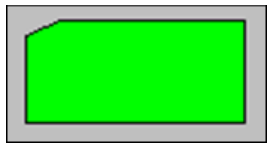
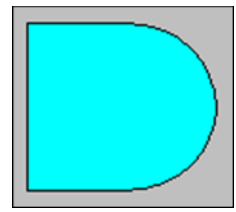
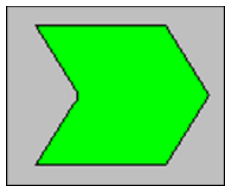
	Calcula y genera los bucles necesarios para esperar un tiempo determinado.
 Cálculo matemático	 Visualizar valores en display de 7 segmentos por el método del multiplexado.
 Registro de desplazamiento. Ingresar un valor y rotar	Llamada a subrutina (Call).

Tabla 4.1 Definición de Bloque de Niple

Selección de Bloques o Vínculos

Llamada a subrutina (Call).

Cuando seleccionamos un bloque, se presentan una serie de 9 nodos (8 alrededor y 1 en el centro del bloque). En el caso de los vínculos se presenta 1 nodo en la mitad de la flecha.

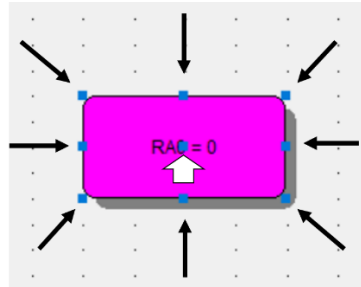


Fig. 4.27. Selección de Bloque o vinculo

- Puede seleccionar uno o más bloques o vínculos de varias maneras.
- Haciendo un clic sobre el bloque o sobre el vínculo.
- Manteniendo presionada la tecla "ALT" o la tecla "CRTL" puede ir incorporando más bloques o vínculos a la selección actual.
- Puede seleccionar varios bloques y vínculos de la siguiente manera:
    - Marque un punto en el área de trabajo (no sobre un bloque)
    - Presione el botón izquierdo del mouse y arrastre sin soltar el botón, se mostrará un rectángulo en línea de trazos.
    - Al soltar el botón del mouse, se seleccionarán todos los bloques y vínculos que son incluidos en el área seleccionada.

Mover un Bloque

Realizar la selección del bloque que desea desplazar y sin soltar el botón izquierdo del mouse y arrastre hasta ubicar el bloque en la nueva posición. Puede mover varios bloques simultáneamente.

Note que todos los bloques que dependen del bloque seleccionado se mueven en conjunto de manera rígida. Esto se debe a que el modo "Arrastrar bloques" se encuentra activado. Si conmuta a modo "Arrastrar bloques" desactivado se desplaza el bloque seleccionado.

Para conmutar entre el modo "Arrastrar bloques" Activado y Desactivado ingrese al menú "Opciones/Diagramas de Flujo/Activar

Arrastre".

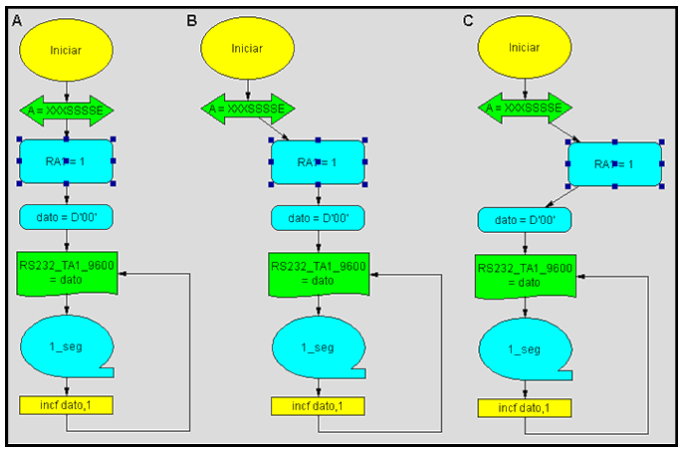


Fig. 4.28. Mover bloque o bloques

a) Posición original.
b) Modo "Arrastrar bloques" Activado.
c) Modo "Arrastrar bloques" Desactivado

Utilice el comando "Edición/Deshacer" para restablecer la posición anterior del bloque.

Estirar un Bloque

Al seleccionar un bloque se muestran 9 nodos (8 alrededor y 1 en el centro del símbolo seleccionado). Para modificar el tamaño del bloque, haga un clic en uno de los 8 nodos periféricos y, sin soltar el botón izquierdo del mouse, arrastre hasta alcanzar el tamaño deseado.

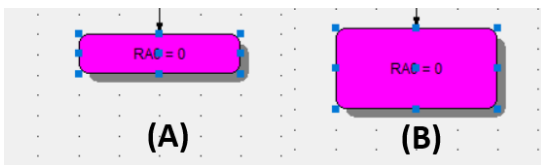


Fig. 4.29. Estirar Bloque

a) Tamaño del bloque original.
b) Bloque estirado

Utilice el comando "Edición/Deshacer" para restablecer el tamaño anterior del bloque.

Borrar Bloques

Tiene 2 formas de eliminar uno o varios bloques:

1) Seleccione el bloque que desea eliminar y presione la tecla "Suprimir".
2) Seleccione el bloque que desea eliminar y presione el botón derecho del mouse. Se desplegará un "menú contextual" y presione la opción "Borrar Símbolo" (o la opción "Borrar" si ha seleccionado varios bloques o vínculos).

Fig. 4.30. Eliminar Bloques

Utilice el comando "Edición/Deshacer" para recuperar los bloques borrados.

Editar Bloques

Muchas veces es necesario cambiar el valor de algún parámetro u opción

a un bloque ya insertado dentro del diagrama de flujo. Para esto tiene 2 opciones:

1. Puede borrar el bloque e insertarlo nuevamente seleccionado los parámetros deseados. Si elige esta opción deberá realizar los vínculos de manera manual.

2. Puede editar el bloque y asignarles nuevos valores a los parámetros solicitados. Para esto no necesita borrar el bloque ni realizar los vínculos nuevamente.

Para editar un bloque, selecciónelo y luego haga un clic con el botón derecho del mouse para que se presente el menú contextual, y elija la opción "Editar", o bien realice doble clic sobre el símbolo a editar y automáticamente entrara al menú de edición del símbolo.

Para que el menú contextual presente la opción "Editar", sólo debe estar seleccionado el bloque a ser editado. Si selecciona más de 1 elemento, ya sean bloques o vínculos, el menú no presentará la opción "Editar" ya que el sistema no puede determinar cuál es el bloque que desea editar.

Se presentará la pantalla correspondiente a el bloque seleccionado. La pantalla será cargada con todos los parámetros seleccionados.

Proceda a cambiar los valores o las opciones y presione el botón "Ingresar".

Con respecto al chequeo de errores, es importante tener en cuenta que rige el mismo criterio que para la inserción de bloques. O sea que, si el bloque a editar está vinculado a otro bloque, se realizará el chequeo de errores "en línea". Mientras que, si el bloque no se encuentra vinculado,

el chequeo no se realizará.

Duplicar Bloques

Al duplicar un bloque se realiza una copia exacta del bloque seleccionado.

Muchas veces, la opción "Duplicar" un Bloque en combinación con la "Edición" de bloques puede ayudarle a ahorrar tiempo de diseño, ya que es común realizar operaciones similares en distintas partes de proyecto, donde sólo cambian algunos parámetros.

Ud. puede realizar una copia de la "estructura lógica" del programa y luego editar los bloques necesarios. Esto le resultara más cómodo que insertar todos los bloques nuevamente.

Para duplicar un bloque, selecciónelo y luego haga un clic con el botón derecho del mouse para que se presente el menú contextual, y elija la opción "Duplicar".

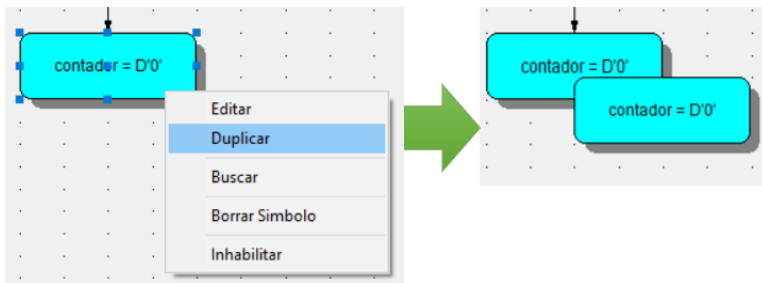


Fig. 4.31. Proceso para duplicar Bloques

Además, puede duplicar uno o más bloques a la vez.

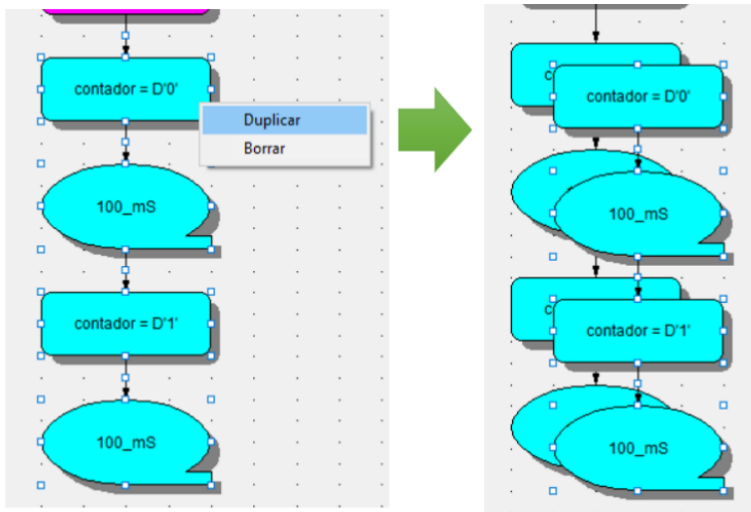


Fig. 4.32. Proceso de selección y duplicación de Bloques

Habilitar e Inhabilitar Bloques

Durante la fase de diseño del proyecto, es habitual que, durante las pruebas, Ud. necesite anular una par- te del proceso par que no sea ejecutada.

Por medio de las opciones "*Habilitar Bloque*" e "*Inhabilitar Bloque*" puede anular la ejecución uno o más bloques sin necesidad de borrarlos o de alterar la lógica del programa.

Para inhabilitar un bloque, selecciónelo y luego haga un clic con el botón derecho del mouse para que se presente el menú contextual, y elija la opción "*Inhabilitar*".

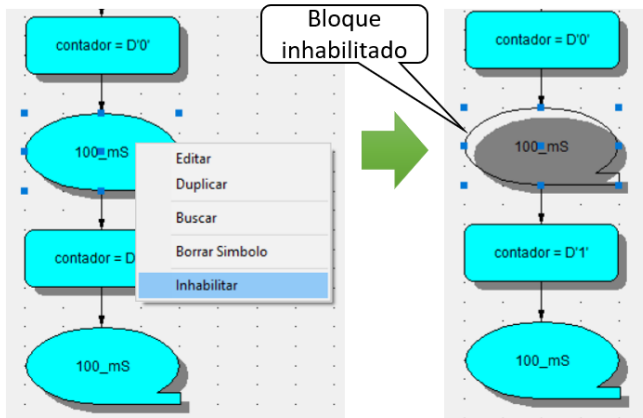


Fig. 4.33. Selección e inhabilitación de Bloque

Un bloque deshabilitado no es considerado en el momento del chequeo de errores "*en línea*" y tampoco genera el código Assembler.

No puede inhabilitar bloque correspondiente a una "Decisión" (símbolo "Rombo") dado que estos bloques tienen 2 salidas.

Para Habilitar un bloque, selecciónelo y luego haga un clic con el botón derecho del mouse para que se presente el menú contextual, y elija la opción "Habilitar".

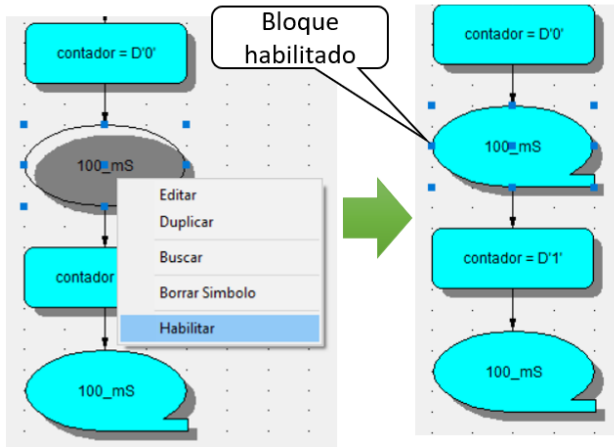


Fig. 4.34. Proceso de selección y habilitación de Bloque

Vínculos (Flechas)

Un vínculo está representado por una *Flecha* que une a 2 o más bloques y determina el flujo del proceso.

Los vínculos determinan la secuencia y el orden de ejecución del programa.

Esto significa en que es muy importante tener en cuenta el sentido en que se genera el vínculo (bloques de origen y de destino).

El bloque desde donde parte el vínculo se denomina "Bloque de Origen" y el bloque al cual llega "Bloque de Destino" y puede identificarlos visualmente porque el vínculo presenta una "Punta de Flecha" en extremo correspondiente al bloque de destino.

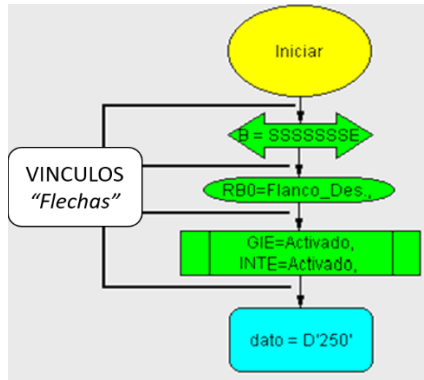


Fig. 4.35. Vínculos o Flechas entre Bloques

Le secuencia de ejecución del programa es; primero se ejecuta el "Bloque de Origen" y a continuación el "Bloque de Destino".

Existe un criterio a tener en cuenta a la hora de realizar vínculos entre bloques y Niple lo supervisa de manera automática:

- Cada Bloque debe tener solo un Vínculo de salida. A excepción de los bloques de Decisión (Rombo) que necesariamente deben tener 2 salidas ("SI" la condición se cumple y si la condición "NO" se cumple).

- Los bloques "Inicio de Entorno" (elipses amarillas) no pueden utilizarse como "Bloques de Destino". O sea que, no pueden tener flechas que "ingresen" a éstos bloques.

- Los bloques "Fin de Entorno" (elipses amarillas) no pueden utilizarse como "Bloques de Origen". O sea que, no pueden tener flechas que "salgan" desde éstos bloques.

Antes de codificar un programa a código Assembler, Niple supervisa que todos los bloques se encuentren correctamente vinculados. En el caso de existir un error de vínculos, Niple lo informa y suspende la codificación del Assembler.

Crear un Vínculo

Para crear un vínculo entre 2 bloques siga los siguientes pasos:

1) Seleccione el "Bloque de Origen".

2) Haga un clic en el Nodo (punto negro) que se encuentra en el centro del bloque.

3) Sin soltar el botón mouse, arrastre hasta el "Bloque de Destino" y suelte el botón del mouse.

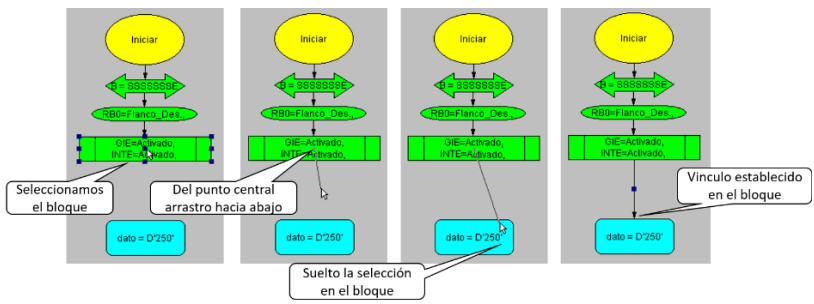


Fig. 4.36. Proceso de creación de un Vinculo

Un vínculo puede adoptar distintas formas de acuerdo a la disposición de los Bloques de Origen y Destino. A estas formas se les llama "Estilo" del vínculo.

Cambiar el estilo de un Vínculo

Para cambiar el estilo de un vínculo;

1) Seleccione el vínculo y presione el botón derecho del mouse.

2) Se mostrará un menú contextual con los distintos estilos que puede adoptar un vínculo.

3) Seleccione el estilo deseado.

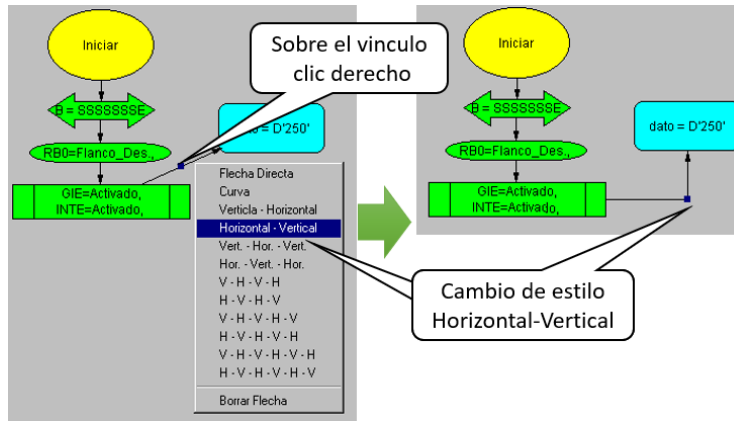


Fig. 4.37. Cambio de estilo de un Vinculo

Borrar Vínculos

Para eliminar un vínculo;

1) Seleccione el vínculo y presione el botón derecho del mouse.

2) Se mostrará un menú contextual.
3) Seleccione la opción "Borrar Flecha".

También puede borrar un vínculo seleccionándolo y presionando la tecla "Suprimir".

Ingreso de Datos

Niple controla de forma automática cada dato ingresado al sistema, ya sean nombres de registro, valores literales u opciones (generalmente numéricas). De esta forma se controlan los errores de "tipeo".

Control de ingreso de nombres de registros

Cuando requiera ingresar el nombre de un registro como parámetro para una función, Niple le presenta una lista con todos los registros declarados, donde Ud. debe seleccionar el nombre del registro deseado. Se debe tener en cuenta al momento de realizar esta operación, como esta configurado el modo de usuario que se explicara mas adelante, ya que dependiendo de este mostrará solo lo permitido de acuerdo a este modo.

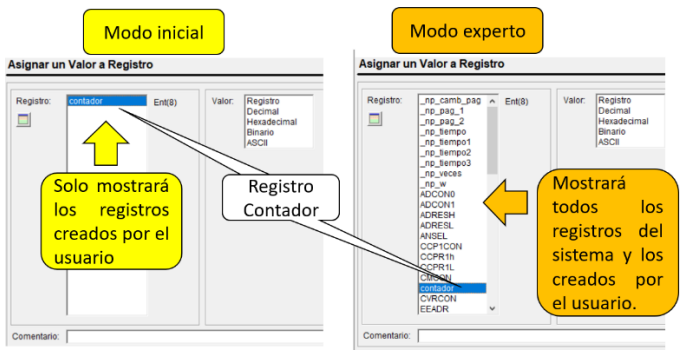


Fig. 4.38. Visualización de registros según modo de usuario

Esto garantiza que el registro a utilizar ha sido previamente declarado y demás previene los posibles errores de "tipeo" por parte del usuario al ingresar el nombre del registro.

Si necesita seleccionar como parámetro, el nombre de un registro "*Predefinido*" y no aparece en lista, consulte el tema "Conceptos Básicos/Modo de Usuario".

En general, junto a la "lista de registros", Niple presenta el Botón "*Nuevo*". Este botón le carga la pantalla de declaración de registros.

De ésta manera, si necesita ingresar como parámetro el nombre de un registro que aún no ha sido declarado, tiene la posibilidad de declararlo y seleccionarlo sin necesidad de cancelar la pantalla actual.

Al cargar la pantalla "*declaración de Registros*" puede declarar más de un registro nuevo antes de volver a la función actual.

Pero al retornar, Niple seleccionará de manera automática en la lista, el nombre del registro seleccionado al momento de cerrar la pantalla.

Control de ingreso de valores Literales

Niple le brinda distintos formatos de ingreso de valores literales, éstos son; formato *decimal*, *hexadecimal, binario y ASCII*.

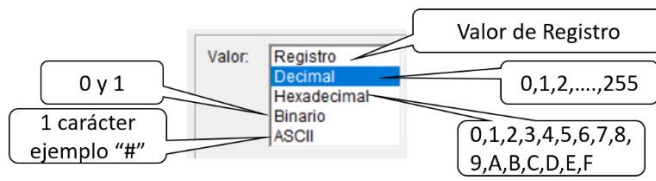


Fig. 4.39. Tipo de valores de ingreso de datos

En general, cuando necesite ingresar un literal, primero debe seleccionar el "formato" en el cual desea ingresar el valor. De ésta manera, Niple supervisa de manera automática el ingreso de datos, y admite sólo los valores permitidos para el formato seleccionado.

Si ingresa un valor no permitido, Niple lo muestra de color rojo y no permite continuar con el proceso actual hasta no corregir el valor ingresado.

Cuando ingresa un número en formato *Binario*, Niple lo acepta como válido sólo cuando se han ingresado los 8 dígitos que conforman el número. Esto garantiza la correcta interpretación del valor ingresado.

Modos de Trabajo "Nivel de Usuario"

Dado que Niple está especialmente diseñado para quienes se inician en la programación de microcontroladores PIC, el sistema ofrece 2 modos de trabajo según el nivel de experiencia o conocimiento técnico del usuario:

- Modo de Usuario Inicial:
  Únicamente se permite el acceso y visualización de los registros y bits declarados por el usuario, y no se tiene acceso a los registros predefinidos. Los únicos bits predefinidos a los que se tiene acceso son los correspondientes a los puertos (RA0-5; RB0-7; RC0-7), y los bits C y Z, dado que estos bits son de uso frecuente. De esta forma, Niple pretende evitar errores al modificar registros de configuración y control del sistema (ya sea por desconocimiento o de manera accidental).

- Modo de Usuario Experto:
  Se permite el acceso a todos los registros y bits del sistema sin ninguna restricción. Niple supone que el usuario tiene pleno conocimiento de las acciones que se está realizando, y no advierte sobre de los riesgos de modificar el valor de los registros de configuración o control.

Por defecto Niple inicia la sesión de trabajo en modo "Usuario Inicial". Para conmutar entre Modo de Usuario Inicial y Experto ingrese al menú "Opciones/Nivel de Usuario".

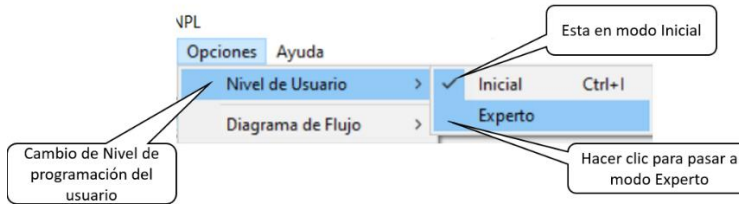


Fig. 4.40. Cambio de modo de usuario (inicial/experto)

Rutinas de Usuario

El usuario puede diseñar sus propias funciones o procedimientos para solucionar un problema en particular (siempre combinando distintas rutinas básicas) y luego llamarlos desde distintas partes del programa (subrutinas). Esto le permite, por un lado, la creación de funciones o procedimientos personalizados y por otro lado, la optimización del código Assembler generado, ya que las rutinas de usuario sólo se escriben una vez y luego se insertan "llamadas" a las mismas. Esta categoría incluye las funciones de declaración e inserción de llamadas a rutinas de usuario.

Exportar Rutina de Usuario

Una vez creada una rutina de usuario, Niple le ofrece la posibilidad de guardarla en un archivo con extensión .RUT, para luego poder ser utilizada en otros proyectos. De esta manera puede crear una librería de Rutinas de Usuario.
Se genera un archivo para cada rutina.

Para exportar una Rutina de Usuario realice los siguientes pasos:

1) Seleccione el entorno de la rutina de desea.

2) Seleccione el bloque de inicio del entorno. (El bloque con forma de elipse de color amarillo), y presione el botón derecho del mouse

para mostrar el menú contextual y seleccione la opción "Exportar Rutina".

3) Niple solicita el nombre del archivo donde guardar la rutina de usuario. Por defecto, sugiere el nombre de la rutina de usuario.

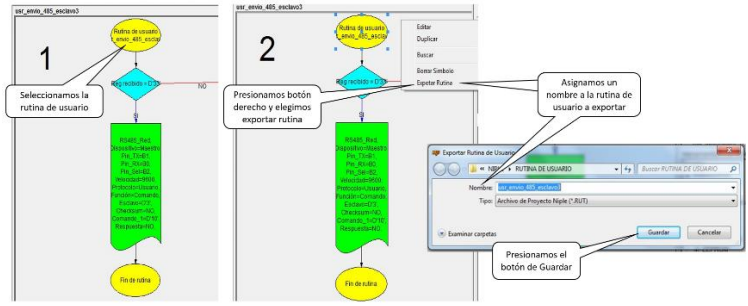


Fig. 4.41. Proceso de exportación de rutina de usuario

Importar una Rutina de Usuario

Niple le permite guardar sus rutinas de usuario en archivos externos para luego ser utilizadas en otros proyectos.
De esta forma, el usuario puede generar su propia librería de rutinas o incluso intercambiar rutinas con otros usuarios.

Para importar una Rutina de Usuario que se encuentra guardada en un archivo, ingrese al menú "Archivo/Rutina de Usuario" y seleccione el archivo de rutina (*.RUT). Niple ingresará la rutina al proyecto actual.

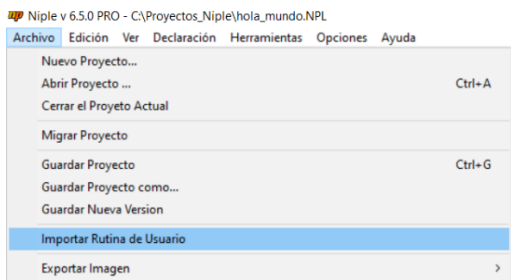


Fig. 4.42. Importar rutinas de usuario

Si el modelo de PIC con el cual fue creada la rutina no es el mismo que el modelo actualmente utilizado, Niple muestra un mensaje de

advertencia, ya que, al generar el código Assembler, pueden ocurrir errores si la rutina utiliza algún módulo que no existe en el modelo de PIC actualmente utilizado.

Por ejemplo, si la rutina fue generada con una versión de Niple para 16F87x y realiza una lectura de un canal A/D y luego la importa a un proyecto de Niple apara 16F6xx, Niple advierte del posible error.

Exportar Imagen

Puede generar un archivo de imagen del diagrama de flujo para realizar la documentación del proyecto. La imagen es generada en formato ".wmf" y corresponde un archivo de tipo vectorial. Este formato es estándar de Windows, o sea que, no requiere de programas especiales para poder visualizarlos o editarlos.

Este tipo de archivos ocupan muy poco espacio de disco.
El archivo de imagen puede ser insertado en un editor de textos para complementar la documentación técnica.
Al ingresar al Menú "Archivo/Exportar Imagen", tiene 2 opciones para generar imágenes:

1. Generar una imagen de "*todo el proyecto*": se generan un archivo de imagen por cada entorno.

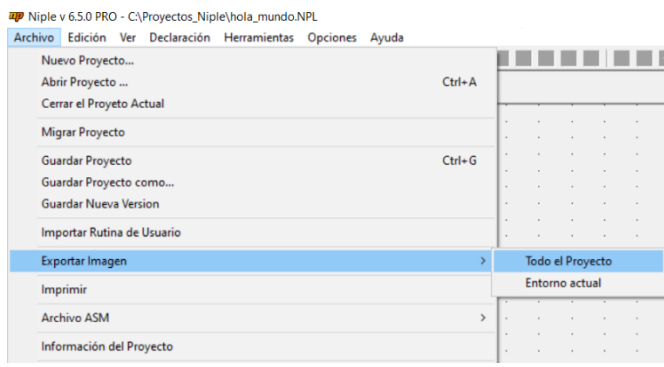


Fig. 4.43. Generación de imágenes de todo el proyecto

2. Generar una imagen solo del "*Entorno actual*": se genera un archivo con el entorno actual.

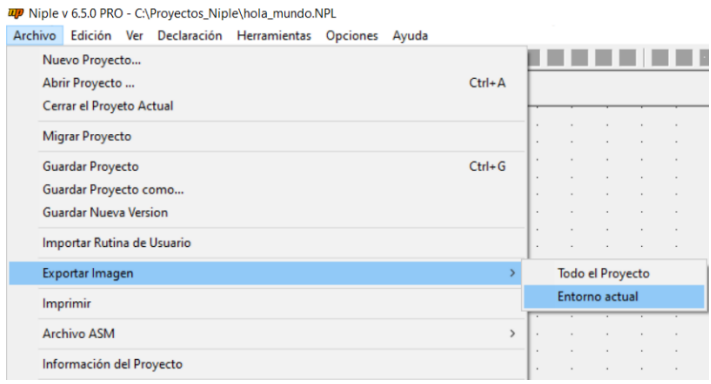


Fig. 4.44. Generación de imagen del entorno actual

Los archivos de imagen se nombran según el siguiente formato: "NombreProyecto_NombreEntorno.WMF".

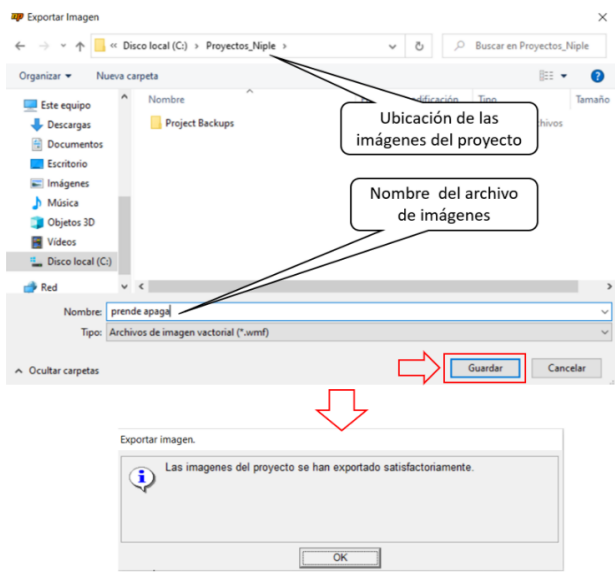


Fig. 4.45 Ubicación de guarda de archivos de imágenes

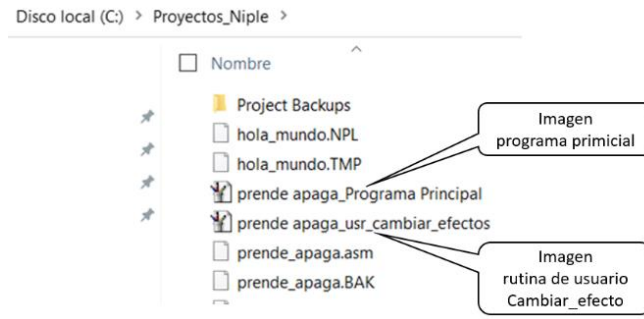


Fig. 4.46 Imágenes guardadas del proyecto

Por esto, Niple solicita el nombre del proyecto y sugiere el nombre actual del proyecto. Si usted, aún no ha ingresado un nombre al proyecto, Niple sugiere "*NombreProyecto*". En el caso de generar archivos de imagen de todo el proyecto, Niple genera un archivo para cada entorno.

Declaración de Registros

Para realizar la declaración de un registro, ingrese al menú "Declaración" y seleccione la opción "Registros". Allí se muestra la ventana de memoria del microcontrolador.

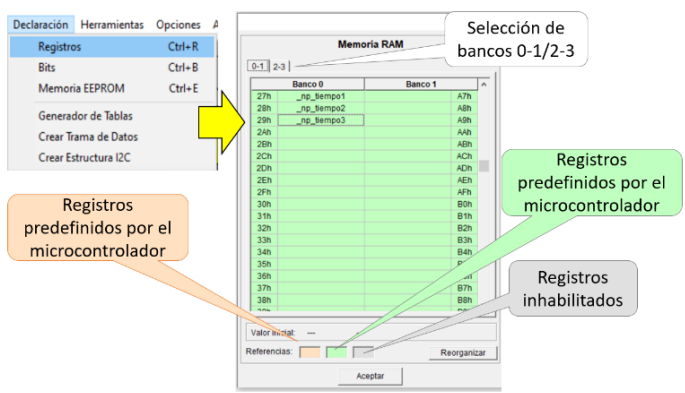


Fig. 4.47 Declaración de registro de usuario

Al hacer clic en una posición de memoria libre (en nuestro caso 2Ah) se carga la ventada de Declaración de nuevo registro:

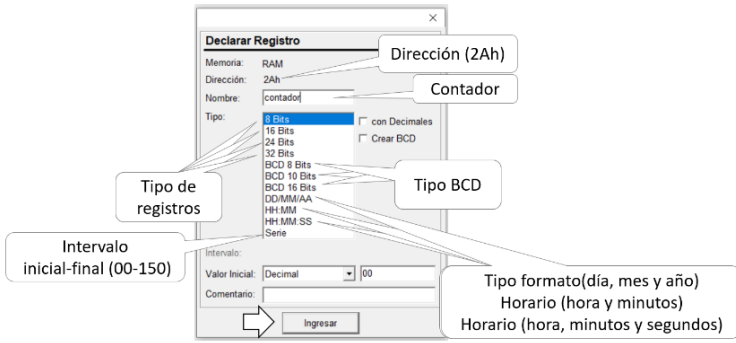


Fig. 4.48 Formatos de declaración de registro

Seleccione el tipo de registro el cual puede ser:

- Registro 8 bits: Se declara un solo registro con el nombre ingresado a continuación. Ejemplo:

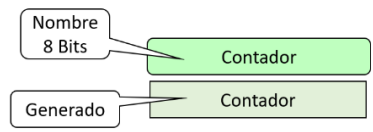


Fig. 4.49 Registro de 8 bits

- Registro 16 Bits H/L (High/Low): Genera 2 bytes con el mismo nombre ingresado agregando los sufijos "_h" y "_l" para poder procesar valores a 16 bits. Ejemplo:

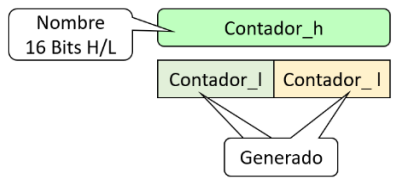


Fig. 4.50 Registro de 16 bits H/L

- Registro 24 Bits HH / H/L: Genera 3 bytes con el mismo nombre ingresado agregando los sufijos "_hh", "_h" y "_l" para poder procesar valores a 24 bits. Ejemplo:
-

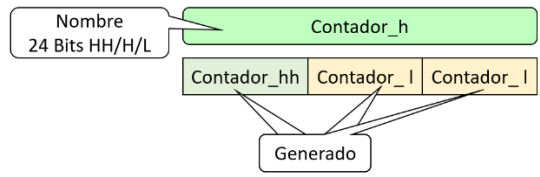


Fig. 4.51 Registro de 24 bits HH/H/L

- 32 Bits HHH / HH / H/L: Genera 4 bytes con el mismo nombre ingresado agregando los sufijos "_hhh", "_hh", "_h" y "_l" para poder procesar valores a 32 bits. Ejemplo:

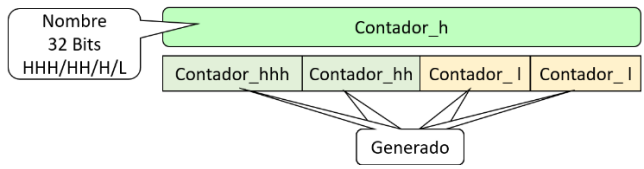


Fig. 4.52 Registro de 32 bits HHH/HH/H/L

- BCD 8 Bits: Genera 3 bytes con el mismo nombre ingresado agregando los sufijos "_uni", "_dec", y "_cen" para poder guardar valores en BCD de registros de 8 bits (entre 0 y 255). Ejemplo:

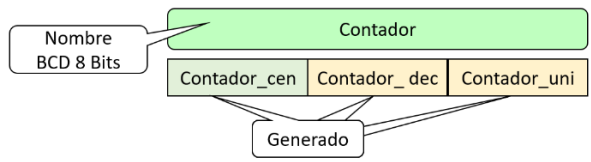


Fig. 4.53 Registro BCD de 8 bits

- BCD 10 Bits: Genera 4 bytes con el mismo nombre ingresado agregando los sufijos "_uni", "_dec", "_cen", y "_u_mil" para poder guardar valores en BCD de registros de 10 bits (entre 0 y 1024). Ejemplo:

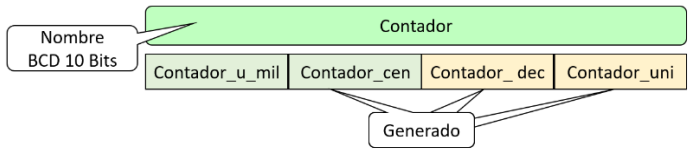


Fig. 4.54 Registro BCD de 10 bits

- BCD 16 Bits: Genera 5 bytes con el mismo nombre ingresado agregando los sufijos "_uni", "_dec", "_cen", "_u_mil" y "_d_mil" para poder guardar valores en BCD de registros de 16 bits. Ejemplo:

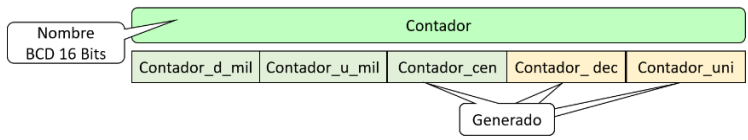


Fig. 4.55 Registro BCD de 16 bits

- Fecha: Genera 2 bytes con el mismo nombre ingresado agregando los sufijos "_d", "_m" para poder guardar valores de fechas. Ejemplo:

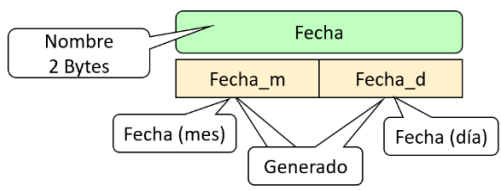


Fig. 4.56 Registro Fecha (2 bytes)

- Hora: Genera 3 bytes con el mismo nombre ingresado agregando los sufijos "_h", "_m" y "_s" para poder guardar valores de horas. Ejemplo:

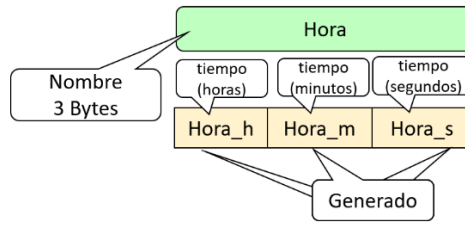


Fig. 4.57 Registro Hora (3 bytes)

- Serie: Permite declarar de manera automática una serie de registros con un prefijo incremental determinado por un intervalo. Ejemplo: se declara una serie de registros llamada "datos" con un intervalo entre "001" y "020".

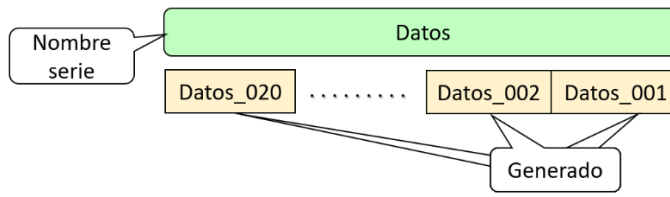


Fig. 4.58 Registro Serie

Luego se ingresa el nombre del nuevo registro y el valor inicial.

La opción "BCD" genera de manera automática los registros correspondientes a la conversión BCD.

Borrar un Registro

Para borrar un registro, debemos seleccionar el registro a borrar y presione la tecla "Suprimir". Sólo puede borrar los registros de usuario. Ejemplo deseamos borrar el registro "Contador2" creado por el usuario.

Fig. 4.59 Borrado de Registro "Contador2"

Puertos

Leer Puerto

Permite obtener el valor actual de un puerto y guardar el resultado en un registro. Es significa que, se obtiene el estado de cada pin correspondiente al puerto seleccionado.

En el campo "Puerto" seleccione el puerto que desea Leer.

Niple supervisa la configuración del puerto seleccionado de manera automática y advierte si el puerto no está completamente como Entrada.

Si bien el hecho de leer el estado de un pin configurado como salida no es considerado un error, igualmente Niple le advierte para asegurarse que Ud. tiene pleno conocimiento de la actual configuración.

Fig. 4.60 advertencia de configuración de puerto

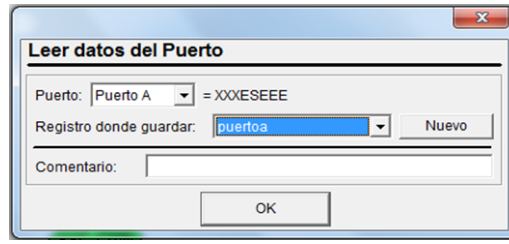


Fig. 4.61 Leer dato de puerto A

Escribir Puerto

Permite escribir un valor en un puerto. Este valor puede ser un valor literal (constante) o el valor de un registro (variable).

Seleccione el tipo de dato desea escribir en el puerto

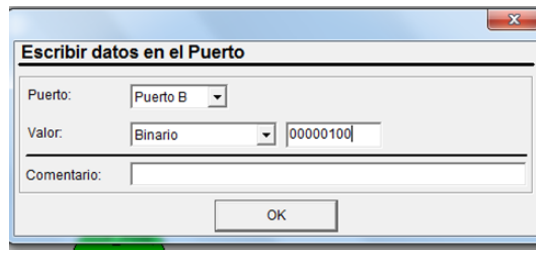


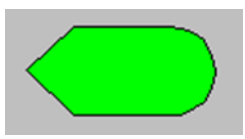
Fig. 4.63 Leer dato de puerto B

Niple supervisa la configuración del puerto seleccionado de manera automática y advierte si el puerto no está completamente como Salida.

Si bien el hecho de escribir un pin configurado como entrada no es considerado un error, igualmente Niple le advierte para asegurarse que Ud. tiene pleno conocimiento de la actual configuración.

Una vez ingresado los datos solicitados presione el botón Ingresar.

Representación gráfica



Sintaxis

Puerto = Valor

Esto se interpreta como: "en 'Puerto' escribir 'Valor'".

Donde:

El parámetro 'Puerto' muestra el nombre del puerto:

A = Puerto A

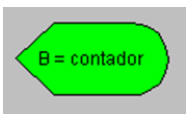
B = Puerto B

C = Puerto C

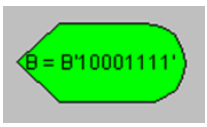
D = Puerto D

El parámetro 'Valor' corresponde a un registro (valor variable) o a un literal (valor constante).

Ejemplos



Escribir el valor que contiene el registro contador en el puerto B.



Escribir el valor constante B'10001111' en el puerto B.

Conversión Analógica/Digital (A/D)
Configuración de los canales A/D

Cuando el micro utilizado dispone de Conversores Analógico/Digital

puede configurarlos desde la configuración del puerto.

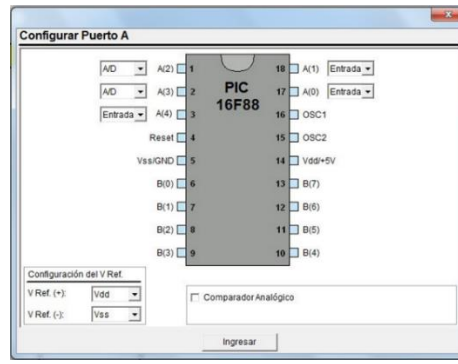


Fig. 4.64 Configuración canales A/D

Seleccione el valor de V Ref (voltaje de referencia) la cual puede ser interna (Vdd y Vss) o externa por medio de determinados pines según el modelo de microcontrolador a utilizar.

Si elige algún voltaje de referencia como "externo", el pin correspondiente es inhabilitado en la pantalla para que no pueda ser utilizado como entrada o como salida.

Una vez seleccionada la configuración deseada, presione el botón "Ingresar".

Antes de insertar el bloque, Niple chequea que la configuración seleccionada corresponda a una de las combinaciones válidas, ya que en algunos modelos de microcontroladores debe emplear las entradas AD con determinada configuración.

Conversión A/D

Una vez configurado el módulo A/D, está en condiciones de realizar una conversión A/D.

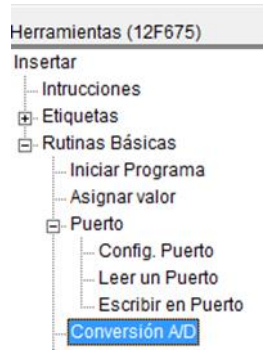


Fig. 4.65 Barra de herramientas conversión A/D

Ingrese al árbol de herramientas y seleccione la opción Rutinas Básicas / Conversión A/D".

Se presenta la pantalla "Conversión Analógica / Digital".

En esta pantalla se muestra toda la información relacionada con el estado actual del módulo de conversión A/D. Esto es:

- Configuración actual del puerto
- Cantidad de canales A/D disponibles

Seleccione el canal A/D que desea utilizar. La lista desplegable llamada "Canal a Leer" contiene todos los canales A/D que se encuentra disponibles.

En esta lista se muestra el número de canal junto al correspondiente nombre de Bit. De esta manera, Niple le facilita la identificación del Pin adecuado.

Para eliminar completamente la posibilidad de seleccionar un pin del Puerto que no hubiera sido configurado como "Entrada A/D", Niple muestra en la lista, sólo los canales que actualmente están disponibles de

acuerdo a la configuración actual del puerto.

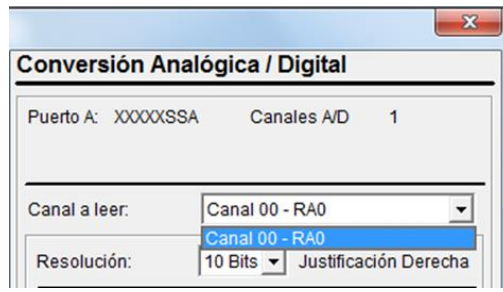


Fig. 4.66 Leer dato de puerto A (Conv A/D)

Una vez seleccionado el canal por el cual realizar la conversión A/D, Niple le solicita que seleccione una de las 2 maneras que tiene de realizar la conversión. Las explicaremos brevemente.

Una conversión A/D puede realizarse de 2 maneras:

1) Mediante un bucle; esperando hasta que finalice la conversión:

El PIC no realiza ninguna otra tarea que no sea "esperar" a que el BIT "go_done" (BIT 2 del registro ad- con0) se ponga en 0, lo cual indica que la conversión ha finalizado.

2) Iniciar la conversión y generar una interrupción al finalizar la conversión A/D:

El PIC inicia la conversión y continúa ejecutando el resto del código mientras se realiza la conversión A/D. Cuando termina de realizar la conversión se produce una interrupción por "Fin de Conversión A/D".

De esta manera si desea realizar la conversión y cuando esta finalice genere una interrupción por fin de conversión tilde la siguiente opción:

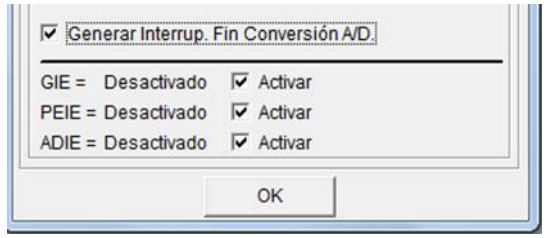


Fig. 4.67 Configuración interrupciones por fin de conversión

Continuación seleccione los registros donde guardar el resultado de la conversión A/D.

Puede realizar conversiones AD a 8 o 10 Bits de resolución.

Si selecciona la resolución a 8 bits, el sistema configura de manera automática la alineación del resulta- do de la medición hacia la izquierda y toma como resultado el registro "Alto", ignorando los 2 bits menos significativos.

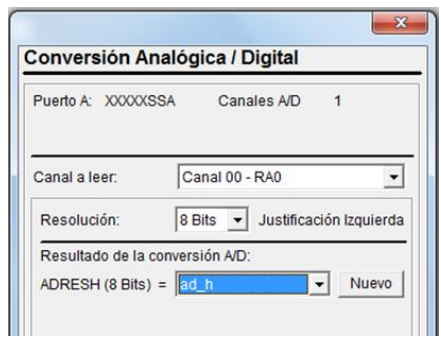


Fig. 4.68 Configuración resultado de conversion A/D

Si selecciona la resolución a 10 bits, el sistema configura automáticamente la alineación del resultado de la medición hacia la derecha y toma como resultado los 2 registros de la conversión A/D.

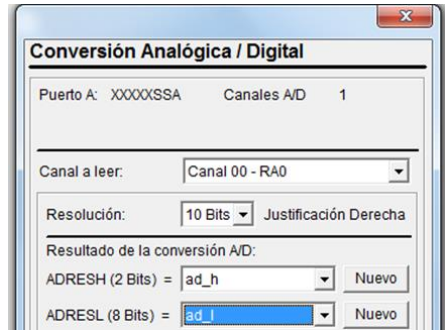


Fig. 4.69 Configuración ADRESH y ADRESL

Niple le permite realizar varias lecturas y promediarlas con lo cual se obtienen mediciones más estables.

Para esto, active la opción "Promedio".

Seleccione la cantidad de mediciones que desea efectuar. Niple efectúa todas las lecturas y realiza el promedio de manera automática.

Niple le da la opción de esperar un tiempo entre las mediciones seleccionadas.

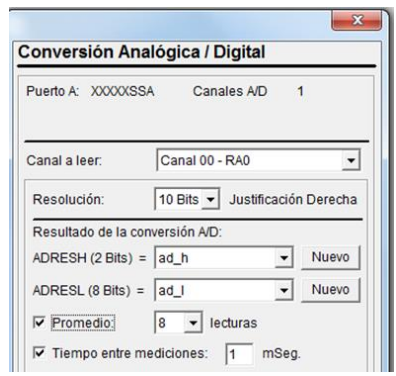


Fig. 4.70 Tiempo de espera entre mediciones

Si elige la opción "Generar una interrupción al finalizar la conversión", Niple chequea de forma automática el estado de los siguientes

habilitadores de interrupciones:

- "GIE" (Habilitador general de interrupciones)
- PEIE (Habilitador general de interrupciones por periféricos, a los cuales pertenece el módulo de conversión A/D).
- ADIE (Interrupción por Fin de Conversión A/D)

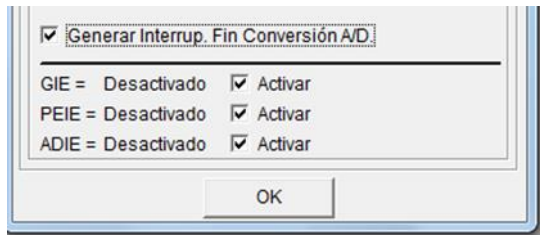


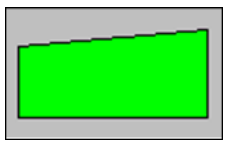
Fig. 4.71 Habilitación de interrupciones

Si alguno de los habilitadores anteriormente mencionados NO se encuentra "Activado", Niple ofrece la opción de activarlo para que la interrupción se produzca normalmente. De lo contrario, la interrupción no se producirá.

El modo de conversión "por interrupción" no permite realizar el promedio automático.

Si decide no activar estos habilitadores, la interrupción no se producirá y, por ende, el fin de la conversión A/D no será "detectado".

Representación gráfica:



Sintaxis:

Registro1, Registro2 = A/D(Nro_Canal)

Donde:

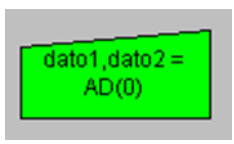
Registro1 = Nombre de un Registro

Registro2 = Nombre de un Registro

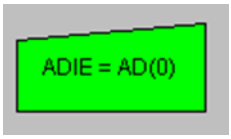
Nro_Canal = Número de canal en el cual se realiza la conversión A/D.

Ejemplos:

A los registros "Dato1" y "Dato2" se asigna el resultado de la conversión A/D.



Genera una interrupción la finalizar la conversión A/D.



EEPROM

Declaración de Registros EEPROM

Puede declarar registros en la memoria EEPROM del PIC para poder acceder a las diferentes dicciones a través de un nombre de registro.

Fig. 4.72 Memoria EEPROM

Se muestra la pantalla de declaración de registros EEPROM.

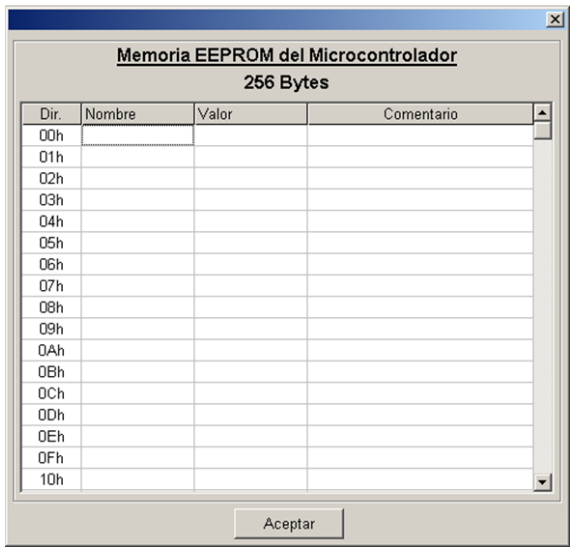


Fig. 4.73 Memoria EEPROM del microcontrolador

La metodología de declaración de registros es la misma que para la declaración de Registros RAM.

Leer EEPROM

Lee un dato de la memoria EEPROM del microcontrolador y lo guarda en un registro. Seleccione entre las opciones de dirección Fija o Variable.

Si elige la opción Fija, seleccione de la lista el registro en donde está contenida la dirección. En la lista "Registro donde guardar el dato", seleccione el registro destino.

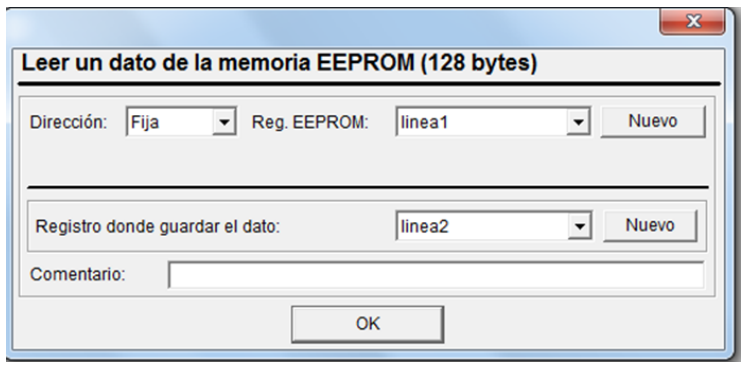


Fig. 4.74 Leer dato de la memoria EEPROM

Presione el botón ingresar.

Si el registro en el que desea guardar los datos no existe, haga clic en *Nuevo*.

Grabar EEPROM

Graba un literal o un registro en la memoria EEPROM del microcontrolador.

En posición de memoria a escribir, seleccione entre las opciones de dirección Fija o Variable. Si elige la opción variable, seleccione de la lista

el registro en donde está contenida la dirección. En Valor a grabar, seleccione entre las opciones un literal o un registro.

Si selecciona la opción un registro, seleccione de la lista el registro que desea guardar en la EEPROM

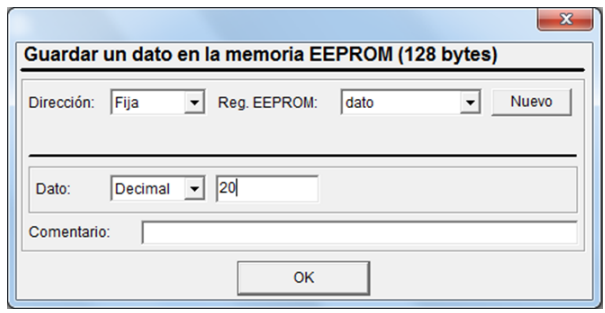


Fig. 4.75 Guarda dato en la Memoria EEPROM

Presione le botón Ingresar.

Si el registro en el que desea guardar los datos no existe, haga clic en Nuevo.

Condición

Las funciones lógicas permiten realizar evaluaciones (comparaciones) de registros y bits con valores lite- rales (constantes) o con los valores contenidos en otros registros y bits (variables).

Comparar Bit

Este bloque evalúa si el valor del BIT seleccionado es 0 o 1 y deriva el flujo del proceso en función del resultado de la evaluación.

En el campo "BIT a evaluar", seleccione el nombre BIT del que desea comparar el valor y a continuación seleccione el valor con el cual comparar el BIT. Un BIT solo puede contener los valores 0 y 1.

Por último, puede ingresar un comentario al bloque.

Una vez seleccionados todos los datos, presione el botón Ingresar.

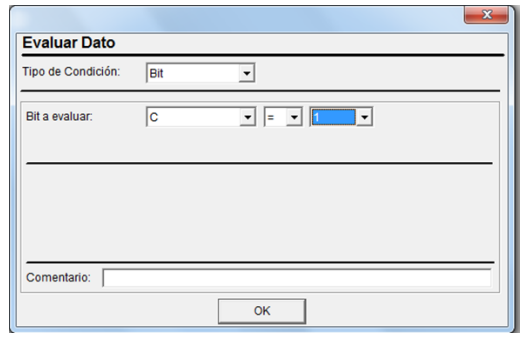


Fig. 4.76 Evaluar dato

Este bloque debe ser vinculado a 2 bloques de salida:

El primer bloque de salida insertado corresponde a la salida "SI", es decir que, en tiempo de ejecución el proceso "tomará" este camino si el resultado de la evaluación es verdadero.

El segundo bloque de salida insertado corresponde a la salida "NO", es decir que, en tiempo de ejecución, el proceso "tomara" este camino si el resultado de la evaluación es falso.

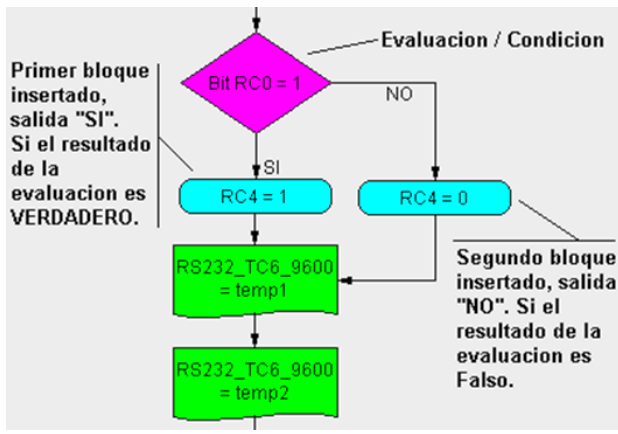


Fig. 4.77 Bloque de Evolución/Condición

Comparar Registro (8 Bits)

Evalúa el valor de un registro, utilizando como valor de comparación un literal o el valor de otro registro.

En la lista "Registro a evaluar", seleccione el registro que desea comparar. A continuación, seleccione el operador de comparación.

Un registro puede ser comparado con un valor literal (constante) o con el valor de otro registro (variable). Seleccione el tipo de valor de comparación marcando en la casilla de opción correspondiente

Si selecciona la opción "un Registro", seleccione de la lista el registro que contiene el valor de comparación, de lo contrario, ingrese el valor literal de comparación.

Por último, puede ingresar un comentario al bloque.

Una vez ingresados todos los datos solicitados, presiones el botón "OK".

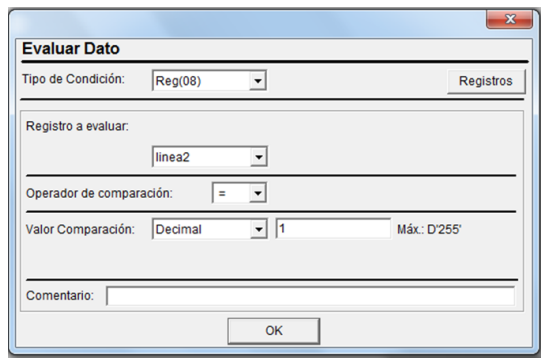


Fig. 4.78 Evaluar dato (Registro 8 bit)

Los operadores de comparación posibles pueden ser =, <, >, <=, >= o <> (distinto de).

Cada vez que debe seleccionar el nombre de un registro y ha olvidado declararlo previamente, Niple le ofrece la posibilidad de declararlo sin necesidad de cancelar la pantalla actual. Para esto presione el botón "Registros" que se encuentra a la derecha de la lista de registros correspondiente.

Este bloque debe ser vinculado a 2 bloques de salida:

El primer bloque de salida insertado corresponde a la salida "SI", es decir que, en tiempo de ejecución el proceso "tomará" este camino si el resultado de la evaluación es verdadero.

El segundo bloque de salida insertado corresponde a la salida "NO", es decir que, en tiempo de ejecución, el proceso "tomara" este camino si el resultado de la evaluación es falso.

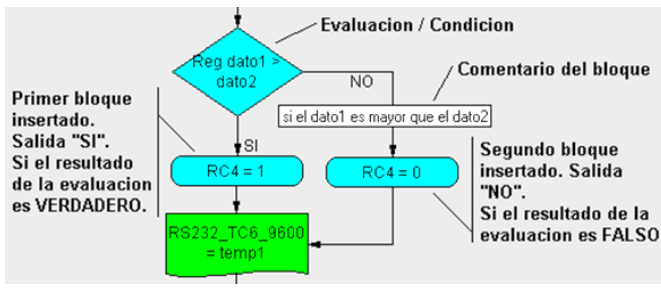


Fig. 4.79 Condición de registro con la salida de NO

Comparar Registro (16 Bits)

Evalúa un valor formado por 2 registros (valor a 16 bits), utilizando como valor de comparación un literal comprendido entre 0 y 65535, o comparándolo con el valor formado por otros 2 registros (16 bits).

En la lista "Registros a evaluar", seleccione los registros que desea comparar. A continuación, seleccione el operador de comparación.

La comparación puede realizarla con respecto a un valor literal

(constante) o con el valor de otros 2 registros (variable). Seleccione el tipo de valor de comparación marcando en la casilla de opción correspondiente.

Si selecciona la opción "Registros", seleccione de la lista los registros que contienes el valor de comparación, de lo contrario, ingrese el valor literal de comparación.

Por último, puede ingresar un comentario al bloque.

Una vez ingresados todos los datos solicitados, presiones el botón "OK".

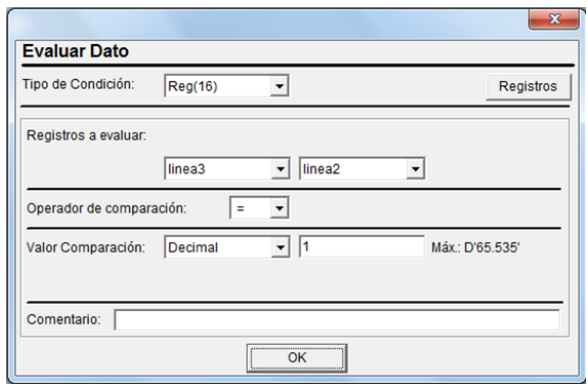
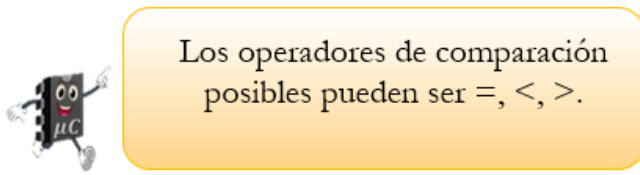


Fig. 4.80 Evaluar dato Registro de 16 bits



Cada vez que debe seleccionar el nombre de un registro y ha olvidado declararlo previamente, Niple le ofrece la posibilidad de declararlo sin necesidad de cancelar la pantalla actual. Para esto presione el botón "Registros" que se encuentra a la derecha de la lista de registros correspondiente.

Este bloque debe ser vinculado a 2 bloques de salida:

El primer bloque de salida insertado corresponde a la salida "SI", es decir que, en tiempo de ejecución el proceso "tomará" este camino si el resultado de la evaluación es verdadero.

El segundo bloque de salida insertado corresponde a la salida "NO", es decir que, en tiempo de ejecución, el proceso "tomará" este camino si el resultado de la evaluación es falso.

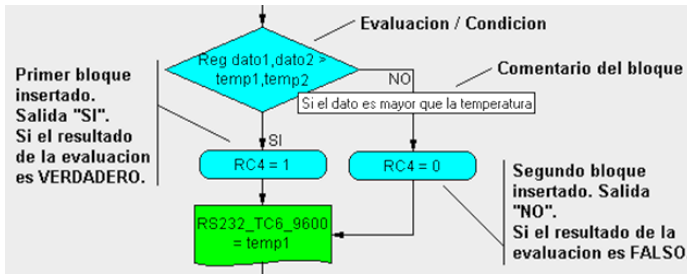


Fig. 4.81 Bloque condicional de registro

Comparar Registros de 24 y 32 Bits

Al igual que comparar registros de 16 Bits Niple en la nueva versión dispone de comparación y manejo de registros de 24 y 32 Bits.

Matemáticas

Puede realizar cálculos de suma, resta, multiplicación y división a 8, 16, 24 y 32 bits. La metodología es la misma para los diferentes tamaños de registro.

Suma

Realiza una operación matemática de Suma, utilizando como sumandos un valor literal o un registro. El resultado es guardado en otro registro.

En Sumando 1, seleccione entre las opciones un literal o el valor de un Registro.

Si seleccionó la opción el valor de un Registro, seleccione de la lista el registro a emplear como Suman- do1. Repita la operación anterior en Sumando 2.

En Resultado, seleccione el registro donde desea guardar el resultado de la operación.

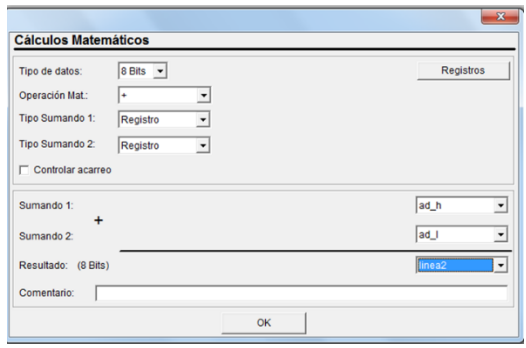


Fig. 4.82 Cálculos Matemáticos

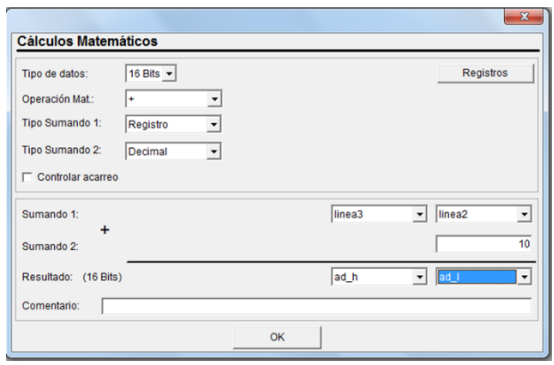


Fig. 4.83. Cálculo matemático registro (16 bits) y un literal Decimal

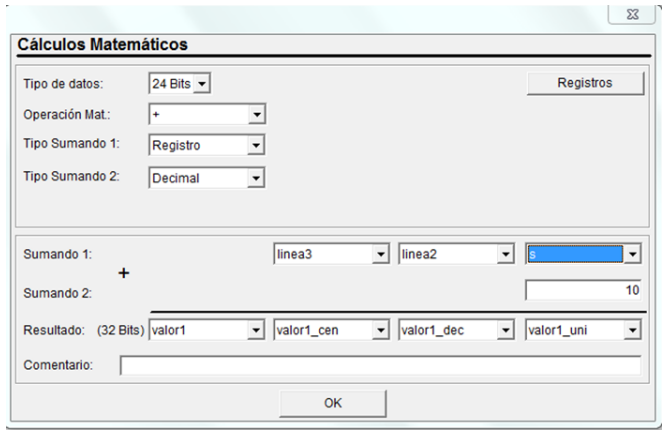


Fig. 4.84 Cálculo matemático registro 32 bits

Resta

Realiza una operación matemática de sustracción, utilizando como minuendo y sustraendo un valor literal o un registro. El resultado es guardado en otro registro.

En minuendo seleccione entre las opciones un literal o el valor de un Registro.

Si seleccionó la opción el valor de un Registro, seleccione de la lista el registro a utilizar como minuendo. Repita la operación anterior en sustraendo.

En Resultado, seleccione el registro donde guardar el resultado de la sustracción. Haga clic en OK.

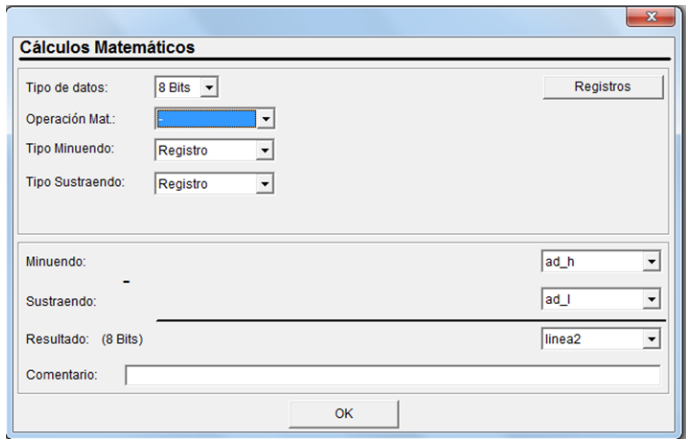


Fig. 4.85 Cálculo matemático resta registro 8 bit

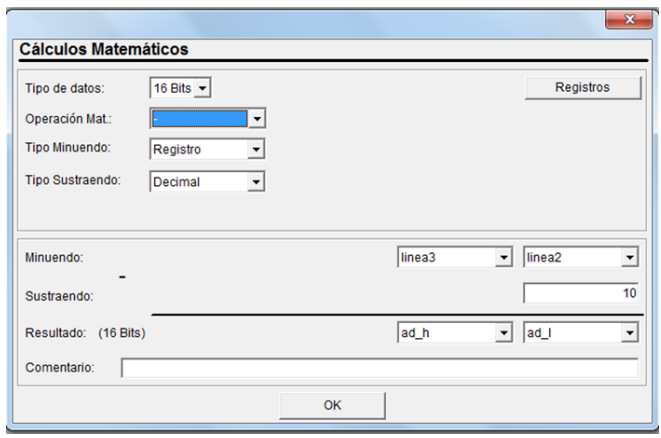


Fig. 4.86 Cálculo matemático resta registro 16 bits

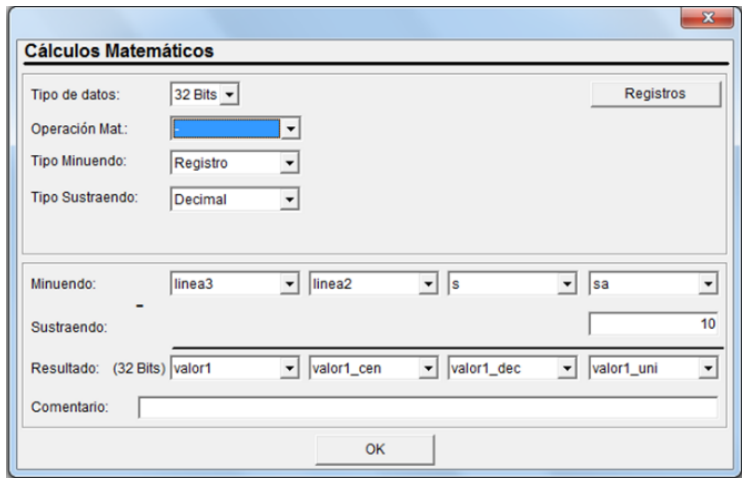


Fig. 4.87 Cálculo matemático resta registro 32 bits

División

Realiza una operación matemática de división, utilizando como divisor valores de un literal o un registro.

El resultado es guardado en otro registro.

En Dividendo seleccione entre las opciones un literal o el valor de un Registro.

Si seleccionó la opción el valor de un Registro, seleccione de la lista el registro a emplear como Dividendo. Repita la operación anterior en Divisor.

En Resultado, seleccione el registro donde guardar el resultado de la división.

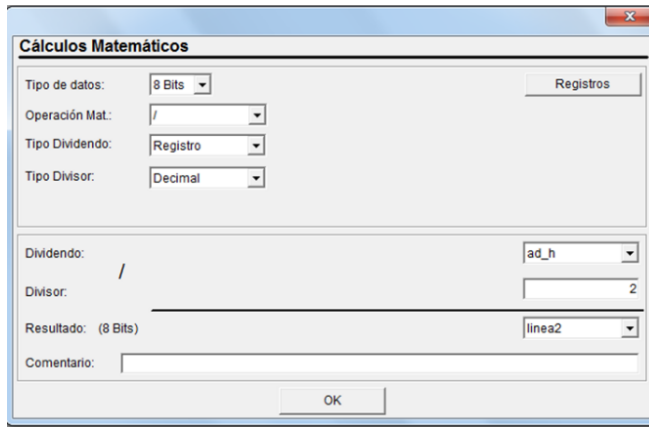


Fig. 4.88 Cálculo matemático división 8 bit.

Multiplicación

Realiza una operación matemática de Multiplicación con registros o valores a 8 bits, utilizando como multiplicador valores de un literal o un registro. El resultado es guardado en un registro.

En Multiplicando seleccione entre las opciones un literal o el valor de un Registro.

Si seleccionó la opción el valor de un Registro, seleccione de la lista el registro a utilizar como Multiplicando.

Repita la operación anterior en Multiplicador.

En Resultado, seleccione el registro donde guardar el resultado de la multiplicación.

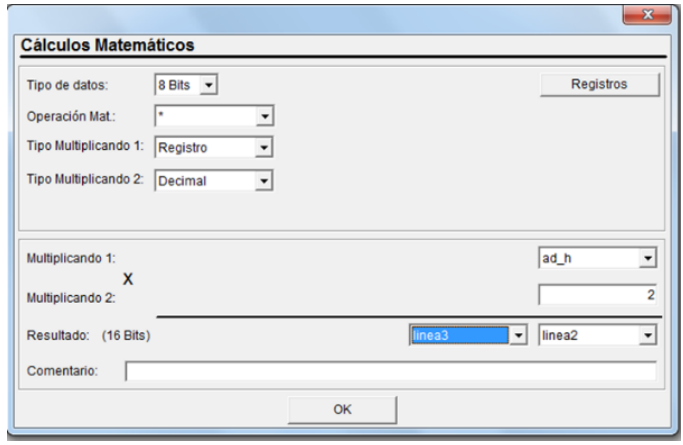


Fig. 4.89 Cálculo matemático multiplicación registro de 8 bit

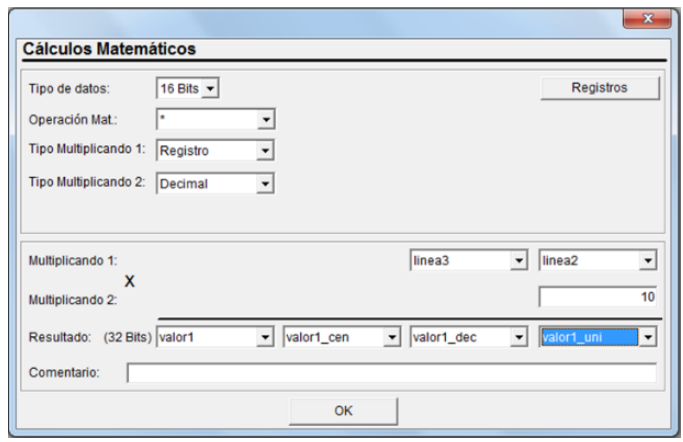


Fig. 4.90 Cálculo matemático multiplicación registro de 16 bits

Incrementar 1

Puede realizar incrementos de registros de manera automática, Niple incrementa el valor del registro y controla el "acarreo" en caso de tratarse de registros mayores a 8 bits.

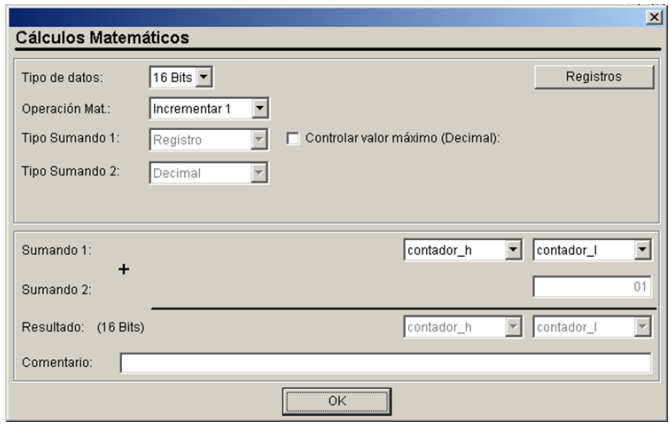


Fig. 4.91 Cálculo matemático incrementar registro 16 bits

Esta opción se trata de un cálculo muy rápido y además ofrece diferentes opciones para control de contadores totalmente automatizado.

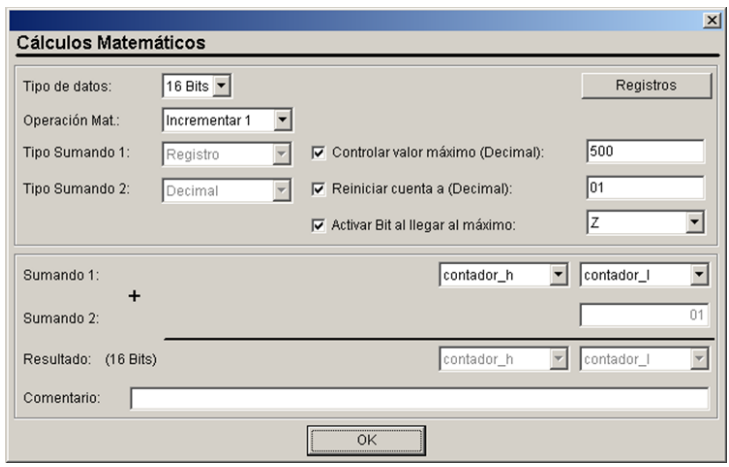


Fig. 4.92 Cálculo matemático incrementar registro de 16 bits

La opción "Controlar valor máximo" le permite automatizar la cuenta hasta un máximo ingresado por el usuario. Una vez alcanzado el valor máximo, ya no sigue incrementando el valor de la cuenta.

La opción "Reiniciar cuenta" le permite volver a inicializar el estado de la cuenta a un valor ingresado por el usuario una vez que se alcanzó el máximo permitido.

La opción "Activar Bit al llegar la máxima" inserta un bloque de comparación donde se evalúa un bit seleccionado por el usuario el cual se pone en "1" cuando la cuenta llego al máximo. De esta manera se puede controlar el flujo del programa según el contador haya llegado al límite máximo o no.

Estas opciones, trabajando en conjunto le permiten realizar estructuras lógicas muy complejas en pocos pasos.

Decrementar 1

Puede realzar decrementos de registros de manera automática, Niple decrementa el valor del registro y controla el "acarreo" en caso de tratarse de registros mayores a 8 bits.

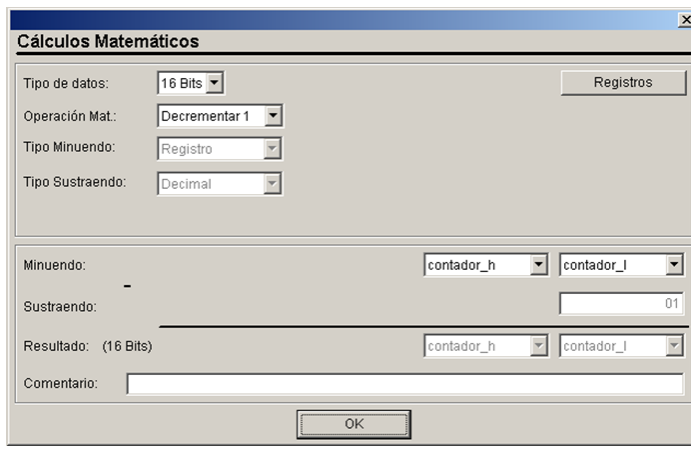


Fig. 4.93 Cálculo matemático decrementar registro de 16 bits

Decimal a BCD

Realiza el cálculo de conversión de un número a BCD, es decir, descomponer el valor contenido en un registro a unidad, decena y centena.

Seleccione el registro que contiene el valor a descomponer.

Seleccione los registros donde desea almacenar el resultado de la descomposición a U, D y C.

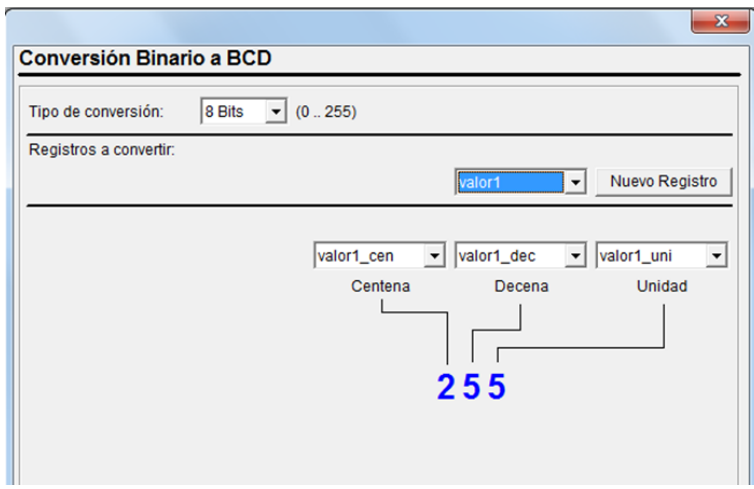


Fig. 4.94 Conversión Binario a BCD 8 bit.

Escalar

La función "Escalar" permite, a partir de un valor dentro de un rango "origen", calcular el valor proporcional correspondiente a un rango "destino".

Por ejemplo: Los registros "Dato1" y "Dato2" contienen el resultado de una conversión A/D, es decir, un valor a 10 bits entre 0 y 1023.

Ahora suponga que desea controlar el ancho de pulso del PWM a 5000 Hz. El ancho de pulso es controlado por medio de un valor a 10 bits contenido en los registros "AnchoPulso1" y "AnchoPulso2". A 5000 Hz el rango válido para el control del ancho de pulso está comprendido entre 0 y 200.

Partiendo de esto, Ud. necesita escalar el valor contenido en los registros "Dato1" y "Dato2" de un rango 0…1023, a un rango 0…200 y asignarlo a los registros "AnchoPulso1" y "AnchoPulso2".

De esta manera se obtiene un valor proporcional, es decir, el mismo porcentaje o relación entre el rango de destino con respecto al rango de origen.

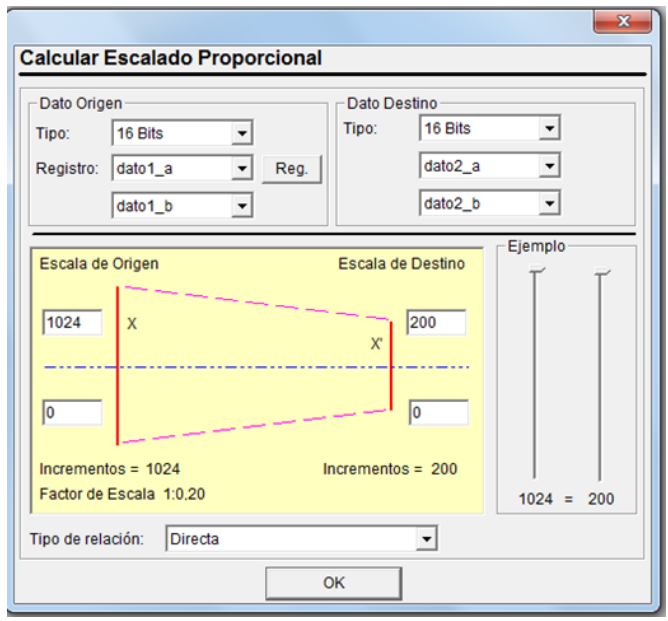
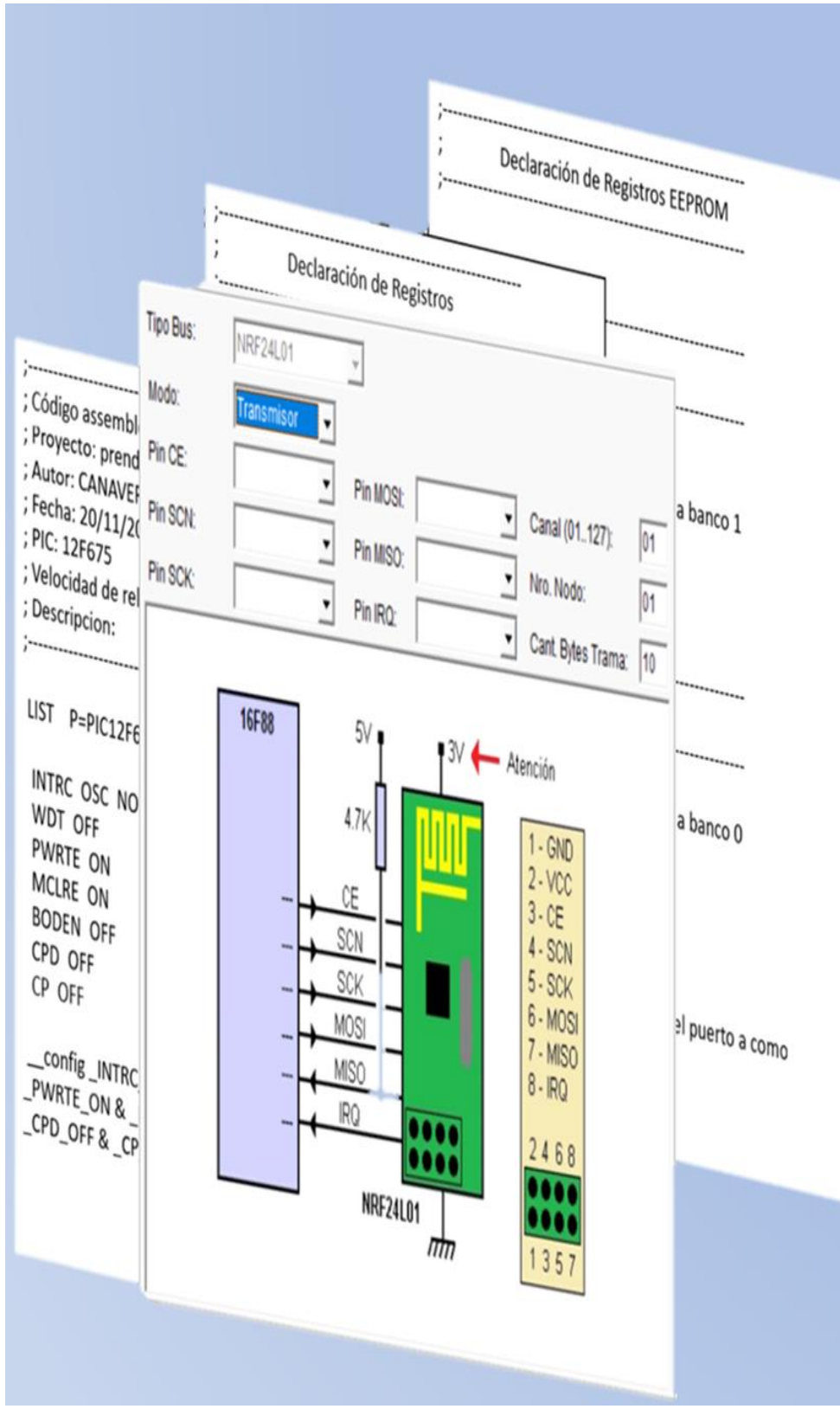


Fig. 4.95 Cálculo escalado proporcional 16 Bits dato origen y destino


Declaración de Registros EEPROM
Declaración de Registros
Tipo Bus:
NRF24L01
Modo:
Transmisor
Pin CE:
Pin SCN:
Pin SCK:
Pin MOSI:
Pin MISO:
Pin IRQ:
Canal (01..127):
01
Nro. Nodo:
01
Cant. Bytes Trama:
10
; Código assembl
; Proyecto: prend
; Autor: CANAVE
; Fecha: 20/11/2
; PIC: 12F675
; Velocidad de rel
; Descripcion:
LIST P=PIC12F6
INTRC OSC NO
WDT OFF
PWRTE ON
MCLRE ON
BODEN OFF
CPD OFF
CP OFF
__config _INTRC
_PWRTE_ON & _
_CPD_OFF & _CP
a banco 1
a banco 0
el puerto a como
16F88
5V
3V
Atención
4.7K
CE
SCN
SCK
MOSI
MISO
IRQ
NRF24L01
1 - GND
2 - VCC
3 - CE
4 - SCN
5 - SCK
6 - MOSI
7 - MISO
8 - IRQ
2 4 6 8
1 3 5 7

Capítulo 5
Programando con Niple

Mi primer programa "Prende_Apaga"

Una vez que tenemos todo configurado del sistema de inicio en Niple, comenzaremos a programar nuestro primer programa, pero antes de ello vamos a tomar unos apuntes para tener presente que vamos a hacer. La primera pregunta que nos hacemos: —¿Qué proyecto haremos? Puedo responderte: —Vamos a hacer un programa para que un microcontrolador a través de uno de sus puertos (elegiremos cualquiera), prenda y apague un led que va a tener a su salida. Entonces en el laboratorio de nuestra Suite de Proteus ¿Qué componentes necesitaremos?, vamos a definir cuáles son:

1. Un microcontrolador (PIC12F675).
2. Una resistencia para conectarla con el Led.
3. Una resistencia para poner un pool up (+5v) para el Master Clear (MCLR).

4. Un Led color verde.
5. El código que vamos a programar (Niple).

Nos preguntamos: —¿Qué hace este programa? Respondiendo: — prenderá un led durante 1 segundo y luego se apagará durante otro segundo, repitiéndose esta acción de forma sucesiva.

Ahora podríamos proponer como pseudocódigo lo siguiente: Sabiendo que utilizaremos el microcontrolador antes mencionado, este deberá realizar la acción de prender y apagar un Led en intervalos de un segundo de manera indefinida a través de uno de sus puertos E/S.

Detallaremos cada línea, formando el literal (líneas) de las acciones a realizar:

1. Inicio.
2. Configurar reloj (4 MHz interno).
3. Configurar todos los puertos como salida.
4. Renombrar el Bit RA0 con el nombre Led1 (para que sea más didáctico y visible a la hora de programar).
5. Inicializar Led1 con valor 0.
6. Bandera inicio.
7. Asignar valor Led1 igual a 1.
8. Espera de 1 segundo.
9. Asignar Led1 con valor 0.
10. Espera de 1 segundo.
11. Bandera inicio.

Se puede observar en el pseudocódigo descrito anteriormente, que no se establece el final del programa, dado que, era una condición de las reglas de negocio establecidas para este problema, de efectuar la acción en forma sucesiva sin interrupción. Esta condición nos permite mostrar que estamos en presencia de una retroalimentación «Loop», quedando el Led1 prendiéndose y apagándose de modo indefinido.

Abrimos Niple, para crear un nuevo proyecto, nos dirigimos al menú,

Archivo>Nuevo proyecto…, luego en la ventana que nos aparece, allí buscamos el microcontrolador PIC12F675, haciendo clic en "Seleccionar".

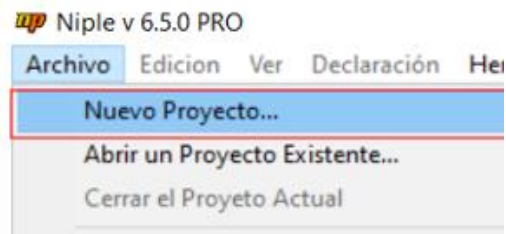


Fig. 5.1 Crear un Nuevo Proyecto en Niple

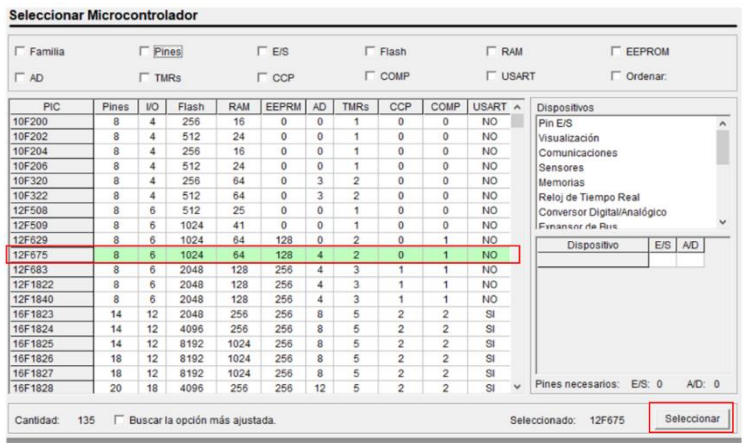


Fig. 5.2 Selección de microcontrolador para el nuevo proyecto

En la siguiente ventana, asignamos como nombre al proyecto: "*prende_apaga*", me permito hacerle una pequeña acotación, los nombres de los proyectos a desarrollar, no deben tener "espacios en blanco", siendo el mismo caso, para el PROMT (carácter o caracteres que forman la línea de comandos) de la ubicación de la carpeta donde se encuentre, en el caso que fuere así, esto ocasionaría un "Error" al momento de compilar el código fuerte. Informando Niple al programador, que se ha detectado un error al momento de compilar el proyecto.

Sabiendo esto, continuamos con asignar el nombre del proyecto, y luego el nombre del archivo. Este tendrá una extensión ".NPL" que identifica al proyecto de Niple. Luego, hacemos clic en "Crear".

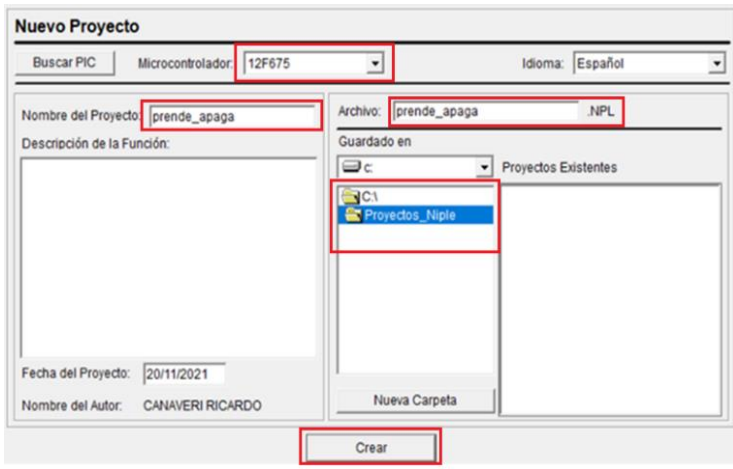


Fig. 5.3 Asignación del nombre de proyecto y del archivo

Pudiendo visualizar ahora, en la pantalla principal de Niple, una nueva ventana de configuración sobre la "Configuración del CPU". Es importante tener presente, que allí debemos cambia el OSC «oscilador» que está por defecto, por un oscilador o Clock interno (para no tener que agregar en el esquemático de Proteus el cristal de 4 MHz y los capacitares cerámicos de 18 pico Faradios). El pin "RESET", lo dejamos, así como se muestra, ya que no lo utilizaremos como pin de "ENTRADA", que es el caso cuando usamos un pulsador.

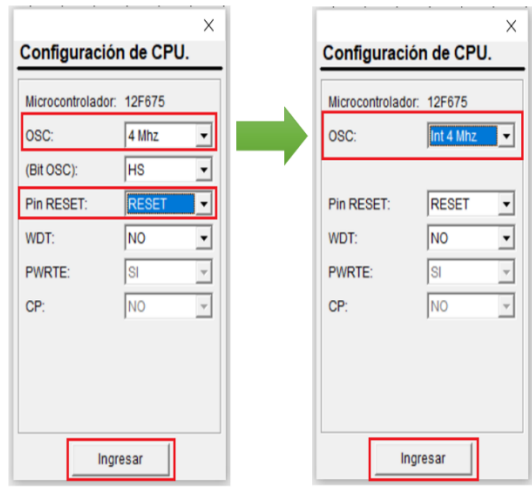


Fig. 5.4 Configuración del CPU (OSC Int 4 Mhz)

Luego, en la siguiente ventana, nos muestra la configuración de los puertos del microcontrolador, debiendo dejar todos los puertos como salida, y hacemos clic en "Ingresar", para dejar esta etapa de configuración.

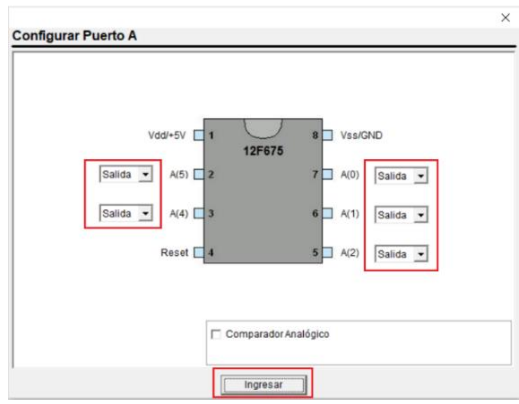


Fig. 5.5 Configuración Puerto A (PIC12F675)

Se puede ilustrar en estos primeros pasos, el icono de "Iniciar" representado en color amarillo, quien se vincula, mediante un conector "flecha hacia abajo" al icono del "Puerto A" que es de color verde. La

configuración del Puerto A queda definida de la siguiente forma: A= XXSSXSSS. Explicando su significado en la siguiente ilustración.

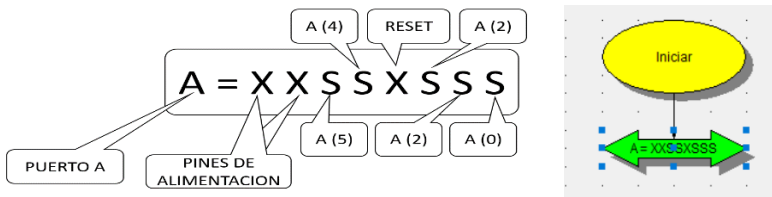


Fig. 5.6 Bloque de configuración Puerto A (PIC12F675)

Una de las facilidades que proporciona Niple a los desarrolladores, en la definición y nombres de registros y bits, es otorgar al programador la posibilidad de cambiar el nombre a los bits de los puertos de los microcontroladores, esto permite, que sea más amigable al momento de la generación del código fuente, y más aún, cuando estamos en presencia de un desarrollo de gran escala, lo que dificultaría al programador tener que utilizar varios bits para la asignación y borrado de valores. De este modo, nuestro programa se hace mucho más amigable y la programación resulta más dinámica y flexible.

Para poder realizar esta acción, de modificar el nombre de un bit en Niple, debemos cambiar la modalidad del nivel de usuario "inicial" y pasarlo al nivel "Experto", ya que estas atribuciones sólo están habilitadas en el modo experto. Nos dirigimos a la barra de menú, en Opciones>Nivel de usuario, debemos cambiar a "Experto", permitiéndonos de esta forma, cambiar el nombre del bit GP0 por "Led1".

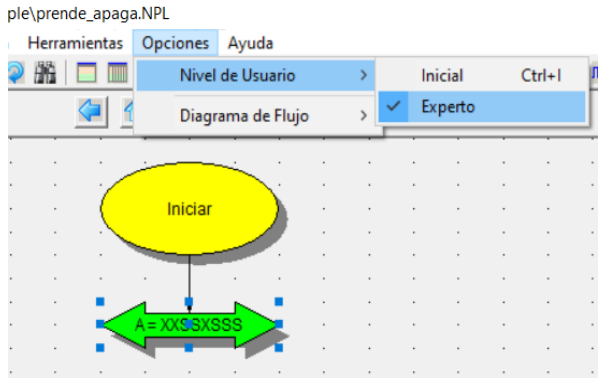


Fig. 5.7 Cambio de modo de usuario (Incial->Experto)

Posteriormente en el menú, Declaración>Bits, ingresamos para cambiar su nombre. Una vez que accedemos a la ventana de "*Memoria de RAM (Bits)*", buscamos el bit GP0, lo seleccionamos y luego damos un "Enter", para cambiarle su nombre por "led1".

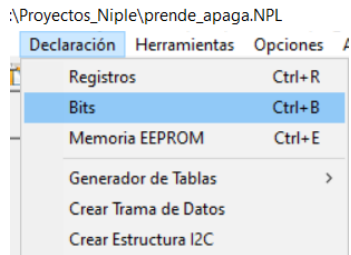


Fig. 5.8 Declaración de Bits

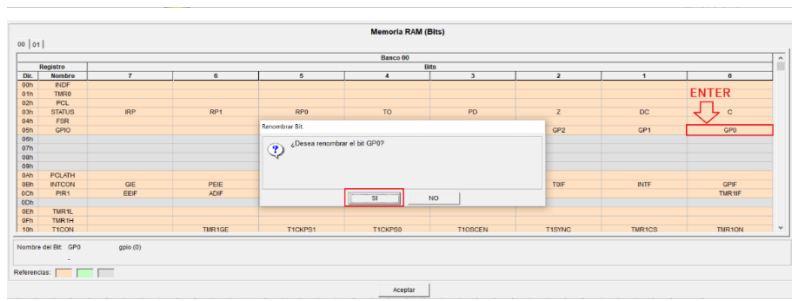


Fig. 5.9 Cambio de nombre del bit GP0

Durante este proceso de cambio nombre, nos sale un mensaje emergente preguntándonos: — ¿Desea que Niple actualice el diagrama de flujo con el nuevo nombre de bit?, eligiendo la opción de "SÍ", luego hacemos clic en "Aceptar" y nos actualizará el nombre del bit GP0 por "led1".

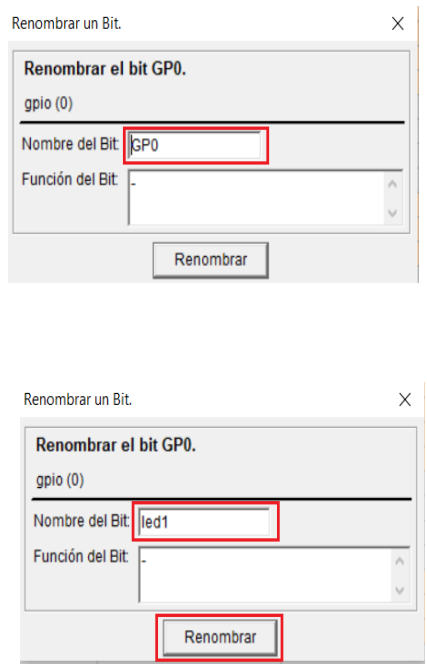


Fig. 5.10 Bits GP0 renombrado por led1

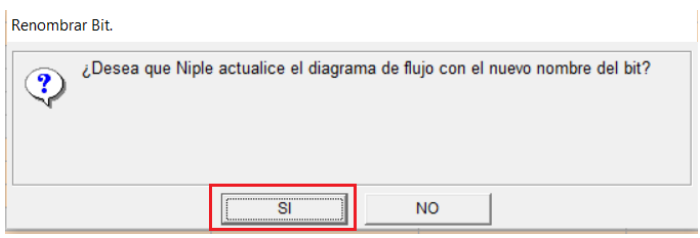


Fig. 5.11 Actualizar nombre de Bits

Utilizamos la barra de herramienta lateral ubicada a la izquierda de la pantalla principal, desplegando Insertar> Rutinas Básicas>Asignar valor

> Bit, usando la barra vertical, buscarnos el bit "led1" y le asignamos el valor "0", luego hacemos clic en "Ingresar".

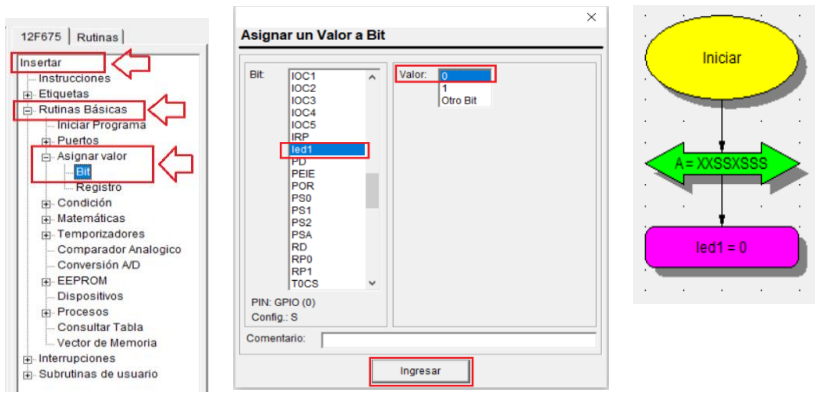


Fig. 5.12 Asignar un valor a Bits

Seleccionando el icono (led1=0), nos permitirá ubicar el comando en la barra de herramientas lateral, para crear una etiqueta (bandera) de "Inicio" de la siguiente forma Insertar>Etiquetas>Nueva Etiqueta, hacemos clic en el botón de "Ingresar".

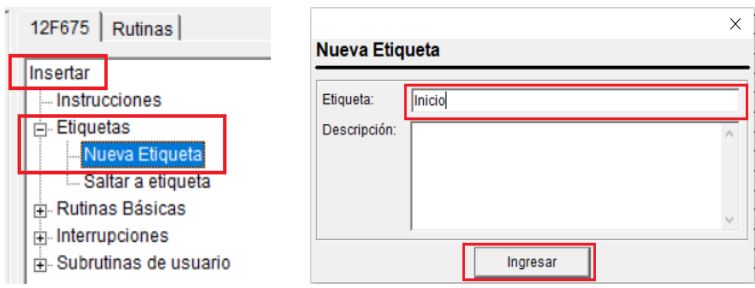


Fig. 5.13 Crear una Nueva Etiqueta

Volvemos a Insertar>Rutinas Básicas>Asignar valor>Bit, para asignar el valor "1" al bit led1, o simplemente, seleccionamos el bloque generado con antelación, presionamos el botón derecho del ratón y elegimos

"duplicar", de esta forma duplicamos el bloque seleccionado, para luego ubicarlo debajo del último bloque, accedemos a este, haciéndole doble clic para cambiarle del bit "led1" el valor "0" por el "1".

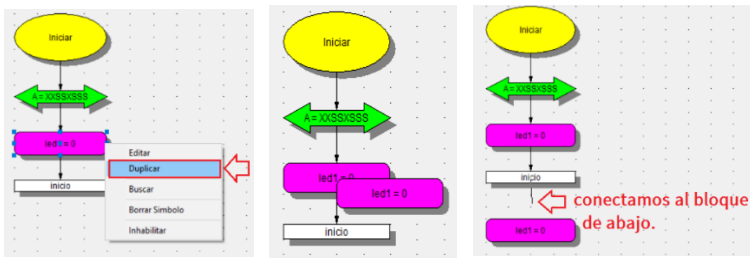


Fig. 5.14 Duplicar bloque (led1)

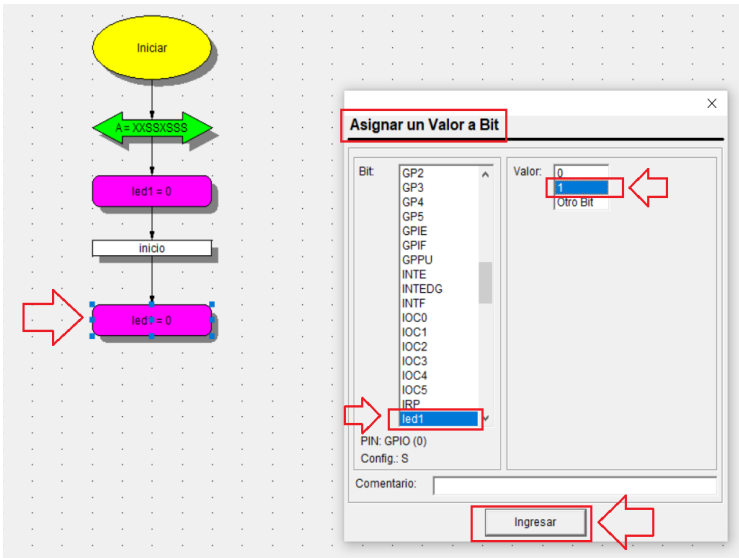


Fig. 5.15 Reasignar un valor "1" al Bit led1.

Para continuar con la secuencia de la programación, debemos asignar un tiempo de espera de 1 segundo, entonces vamos a Insertar>Rutinas Básicas>Temporizadores>Tiempos por ciclos, donde allí, aparecerá una ventana que debemos tildar "Tiempos Fijos", habilitándose dos listas desplegables: en la primera debemos seleccionar la unidad de medición

que es en segundos "Seg.", y en la otra, ingresar el tiempo en formato numérico, introduciendo el "1", luego damos al botón de "Ingresar".

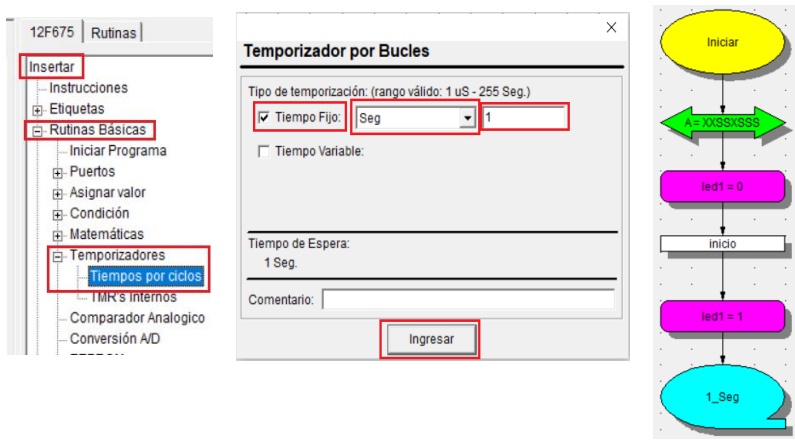


Fig. 5.16 Crear un temporizador por tiempos por ciclos

Repitiendo los pasos previos, de asignación de valor a un bit y de temporizar en 1 segundo, para asignarle en la siguiente etapa su valor correspondiente. También podemos hacerlos ganando tiempo, con la opción de seleccionar "en bloques" los iconos que deseamos duplicar, y luego modificar el valor del bit led1 a "0".

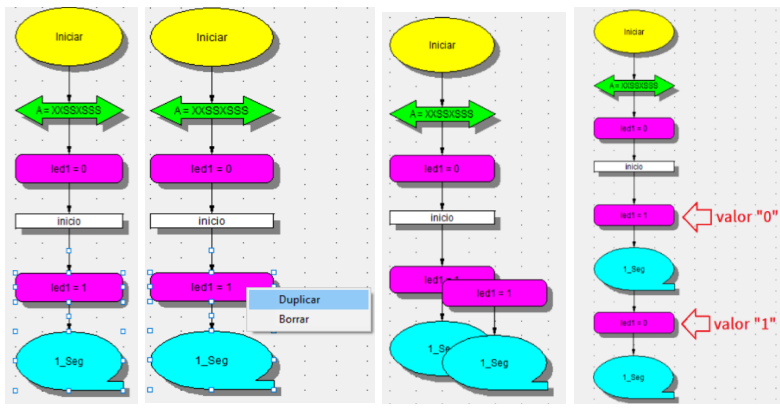


Fig. 5.17 Duplicar bloques para insertar debajo.

Para finalizar, nos faltaría generar un loop (retroalimentación) mediante la etiqueta "Inicio". Vamos a la barra de herramientas lateral, y desde Insertar>Etiquetas>Saltar a etiqueta, hacemos el llamado a la etiqueta "Inicio".

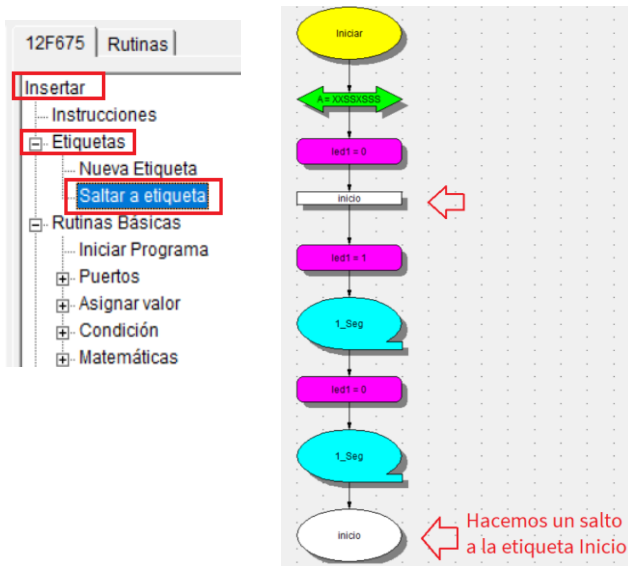


Fig. 5.18 Hacer salto a etiqueta "Inicio".

Guardamos el proyecto, desde Archivo>Guardar Proyecto. Ya lo tenemos guardado con el nombre de "prende_apaga.npl", ahora bien, podemos verificar abriendo el explorador de archivos de Windows, buscando en Mi Equipo>Disco Local (C:)>Proyectos_Niple>, los archivos generados de nuestro primer proyecto.

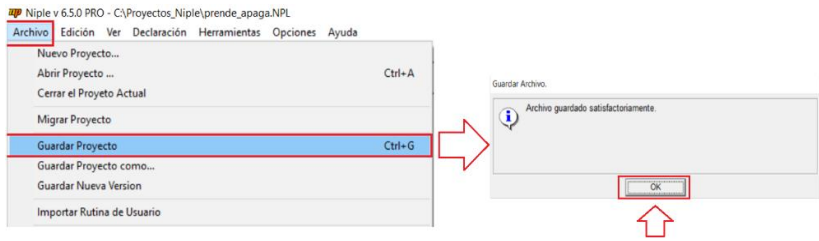


Fig. 5.19 Guardar Proyecto

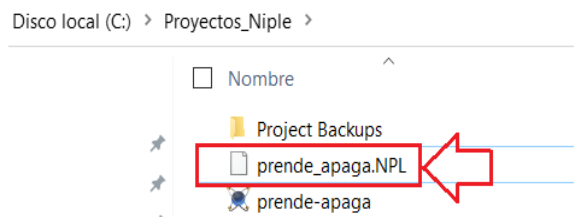


Fig. 5.20 Archivo de proyecto "prende_apaga.NPL"

Pero todavía no hemos finalizado, nos faltaría generar el código ensamblador "Assembler". Para que puedas comprender este proceso, te mostraré en forma gráfica los pasos por los cuales pasa el archivo del proyecto "prende_apaga", en donde, se va transformando hasta llegar al archivo que interpreta el microcontrolador.

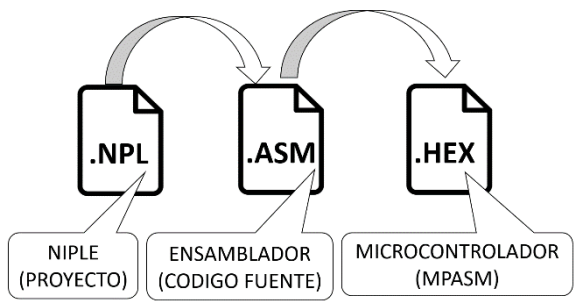


Fig. 5.21 Formatos del archivo de proyecto

Conversión de NPL a ASM

Vamos a la barra de menú, Archivo>Archivo ASM>Generar Código ASM, o sólo presionamos la tecla de acceso rápido (F12).

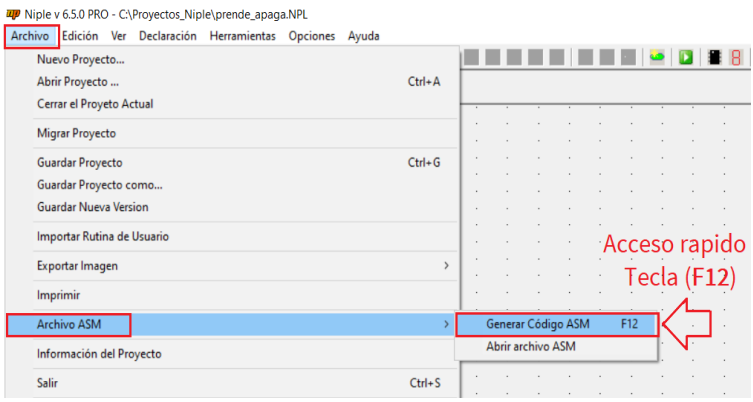


Fig. 5.22 Crear Archivo ASM (Assembler)

Luego se nos abrirá una ventana, mostrando la ubicación del archivo a convertir en ".ASM", el cual se generará en "C:\Proyectos_Niple\prende_apaga.asm", para poder visualizar el código fuente generado debemos tildar "Abrir el archivo ASM generado" y presionamos el botón de "Codificar ASM".

A continuación, en la siguiente ventana "Generación del Código Assembler", nos informa la cantidad de líneas generadas y el porcentaje de la Memoria del microcontrolador utilizada y para finalizar presionamos el botón de "Terminar".

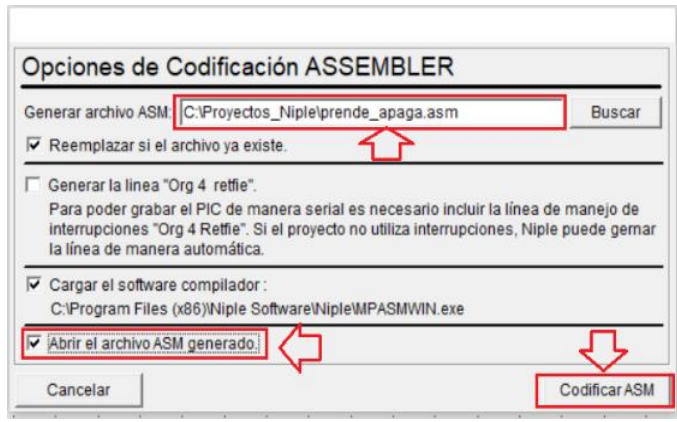


Fig. 5.23 Codificar un archivo NPL a ASM

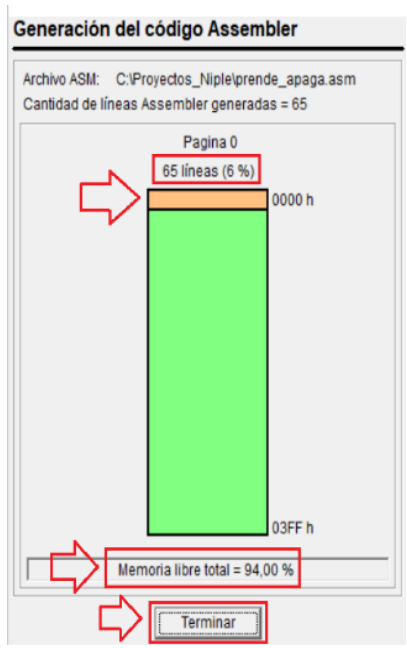


Fig. 5.24 Generación del archivo Assembler

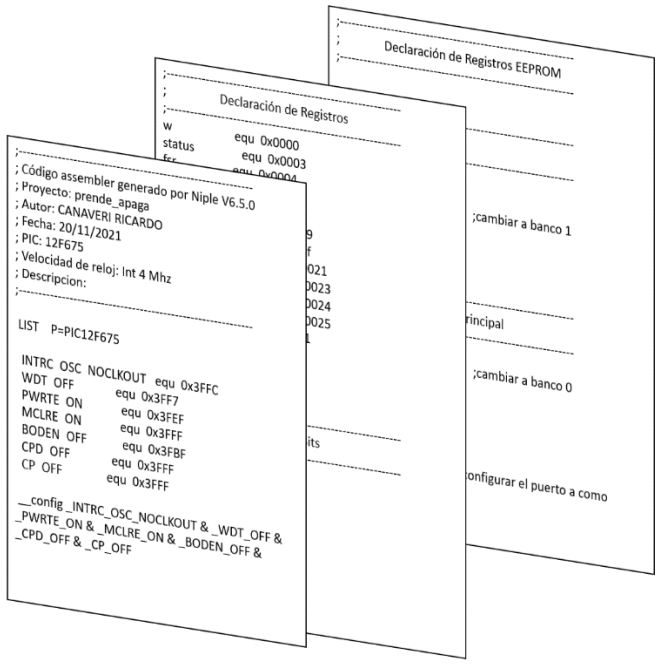


Fig. 5.25 Código Assembler del proyecto "prende_apaga"

Conversión de ASM a HEX

Utilizaremos la herramienta de Microchip MPASM v5.71, que se encuentra incluida en Niple 6.5.0, para ello debemos acceder desde la barra de menú, Herramientas>Software Compilador «.Hex», o mediante la tecla de acceso rápido (F7), allí se abrirá una ventana donde podemos seleccionar el código Assembler y codificarlo a Hexadecimal.

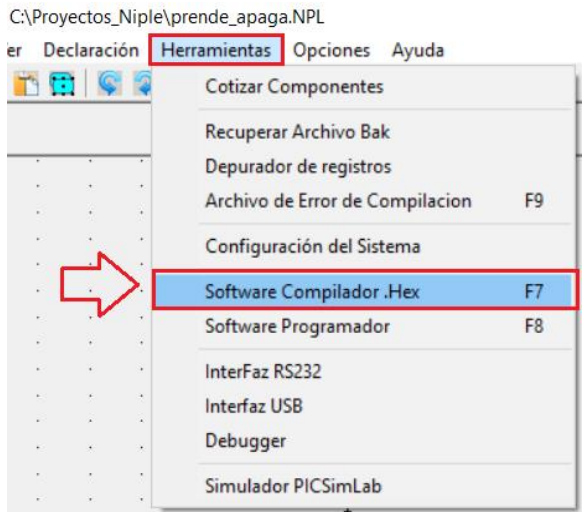


Fig. 5.26 Software Compilador .Hex

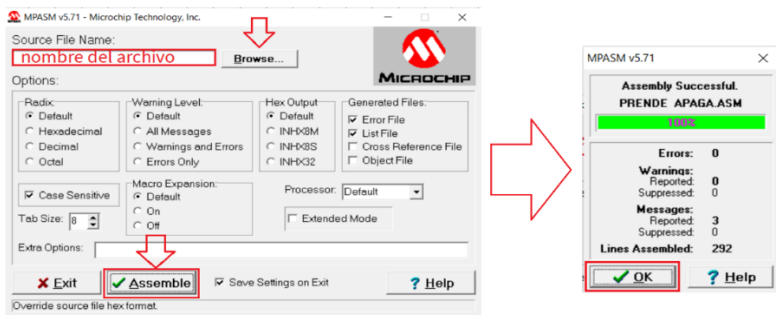


Fig. 5.27 Ensamblado de ASM a .HEX

Podemos visualizar con el explorador de archivos de Windows en la carpeta de proyectos Niple, los archivos generados listos para poder comenzar la simulación en ISIS de Proteus.

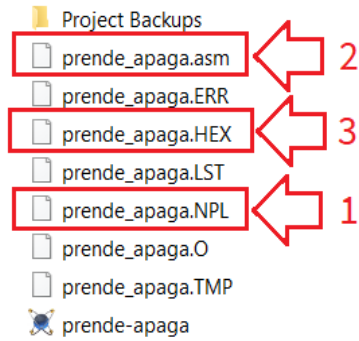


Fig. 5.28 Archivos creados en el proyecto (NPL/ASM/HEX)

Armado del proyecto en ISIS

Para comenzar elegimos los componentes mencionados en este proyecto que son: el microcontrolador PIC12F675, 2 resistencias, 1 led color verde y el código fuente para el microcontrolador. Abrimos Proteus vamos a la barra de herramientas elegimos "Schematic Capture" para diseñar la hoja de trabajo. Nos dirigimos a la herramienta de dispositivos, hacemos clic en la "P" para incorporar los componentes necesarios, en la ventana que nos aparece "Pick Device" escribimos en KEYWORDS la palabra "resistor", y hacemos doble clic en la búsqueda sugerida, de modo de incorporar ese componente en el listado de componentes de trabajo.

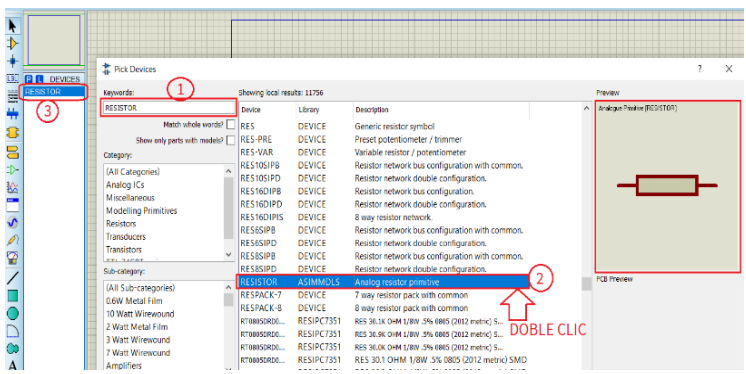


Fig. 5.29 Selección resistor de la herramienta de dispisitivos

Del mismo modo, realizamos la misma acción para el resto de los componentes del proyecto.

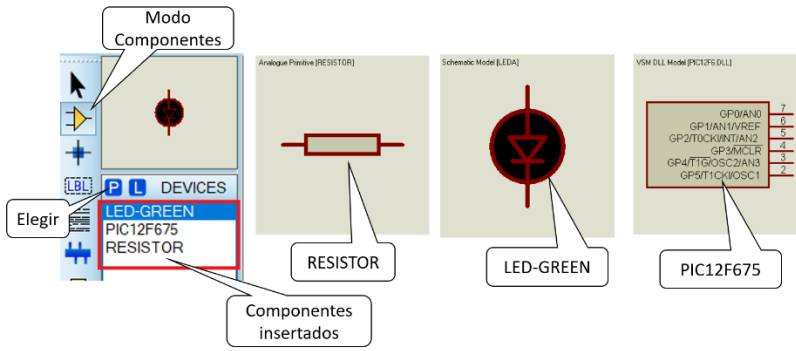


Fig. 5.30 Selección de componentes para el proyecto

Seleccionamos un componente y lo ubicamos en el área de trabajo de la hoja, hacemos clic y se copiará en la hoja. Para el caso del resistor, como inicialmente se encuentra en posición horizontal, antes de ubicarlo en el área de trabajo, podemos usar la herramienta de giro para rotarlo 90º y quede de forma vertical, a los efectos de la practicidad para el diseño.

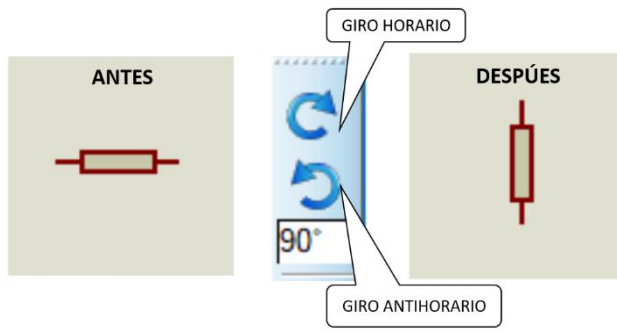


Fig. 5.31 Herramienta de rotación de dispositivos

Luego comenzamos a realizar el conexionado de los alambres (cables) entre cada componente, quedando el esquemático de la siguiente forma:

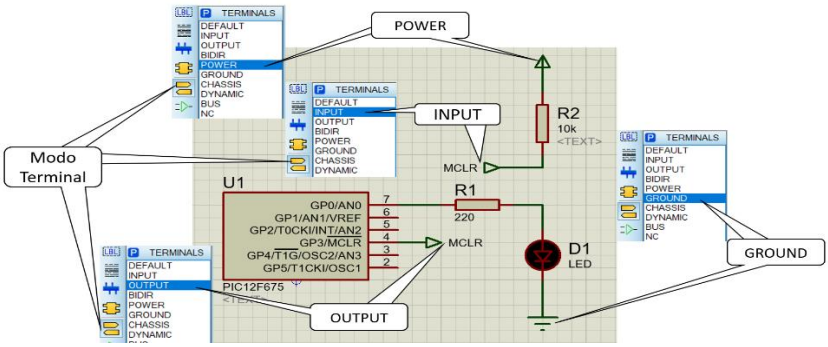


Fig. 5.32 Disposición de componentes del proyecto en ISIS

Es importante personalizar los terminales con nombres, para poder vincular una salida con una entrada. Para esta muestra, agregamos un OUTPUT «salida», desde la barra de herramientas vertical usando el MODO TERMINAL. A ese terminal de salida le hacemos doble clic y podemos acceder a la ventana de propiedades, donde ponemos de nombre "MCLR" y damos a "OK", para guardar esta terminal.

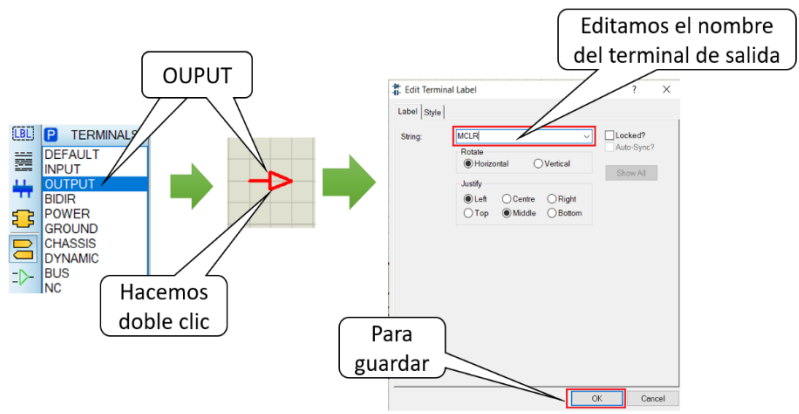


Fig. 5.33 Selección terminal "Output" de Herramienta Modo Terminal

Cuando tenemos que vincular el terminal de salida "MCLR" con un terminal de entrada del mismo nombre (conexión), buscamos en MODO COMPONENTE un terminal de INPUT «entrada», luego le hacemos doble clic y al acceder a la ventana de propiedades en "String" usamos la lista desplegable y elegimos "MCLR" para realizar el conexionado.

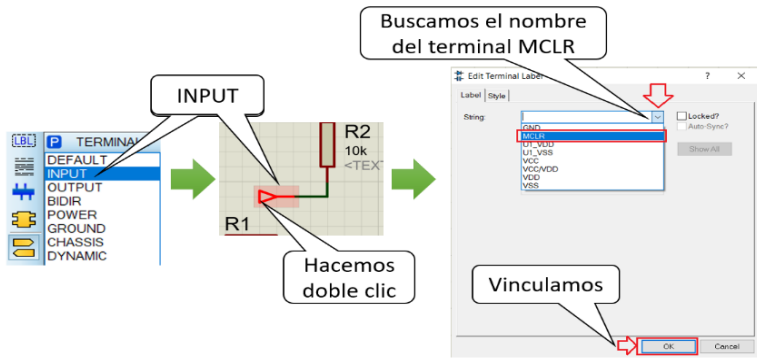


Fig. 5.34 Selección terminal "Input" de Herramienta Modo Terminal

Una vez completado todo el esquema con los componentes, tenemos que tomar la precaución de editar los valores de los resistores, para el caso del pool up (POWER) del puerto *GP3* ***MCLR*** la resistencia será de 10 kΩ (*kilo ohm)* y para el caso del led que se alimenta con una tensión +5v la resistencia debe tener valores entre 150 a 220 ohm, lo cual debemos hacer doble clic a cada una de ellas y editar su valor óhmico.

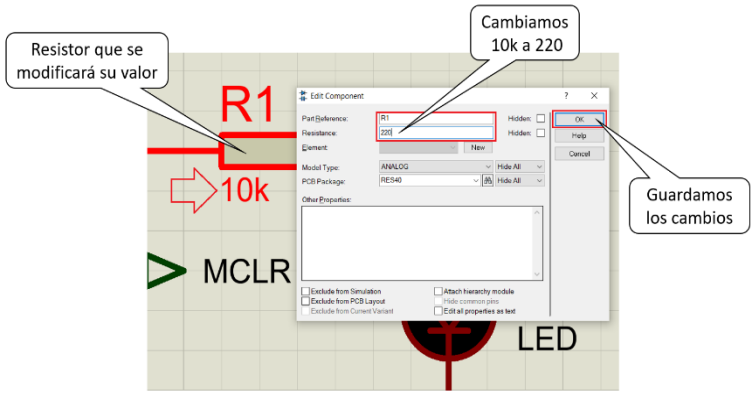


Fig. 5.35 Cambio de valor óhmico en resistor

Nos faltaría cargar el archivo compilado con Niple de nuestro proyecto. Para ello, hacemos doble clic en el microcontrolador PIC12F675, buscamos en "Program File" el archivo guardado "prende_apaga.hex" en la carpeta del proyecto, y hacemos clic en "OK".

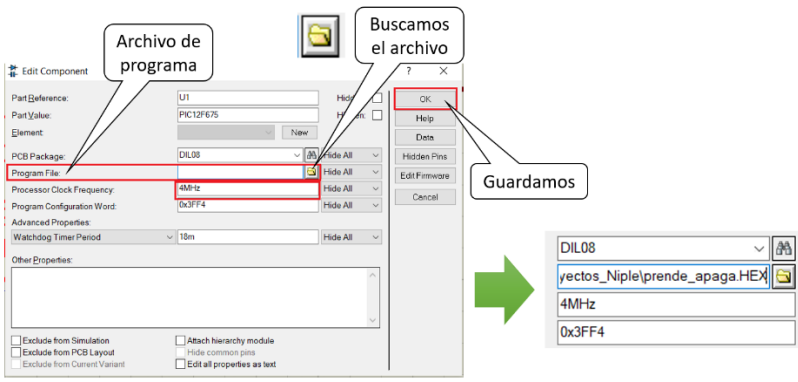


Fig. 5.36 Selección del código fuente (prende_apaga.hex) para el microcontrolador del proyecto.

Utilizando la herramienta de simulación podemos verificar el funcionamiento de nuestro archivo generado. Para ser más intuitivo este aprendizaje pueden acceder a los vídeos que simulamos el

funcionamiento del led prendiéndose y apagándose cada 1 segundo. Luego modificamos el proyecto en Niple cambiando el tiempo de espera por 500 milisegundos y 100 milisegundos, los cuales se pueden apreciar gráficamente en el siguiente acceso.

A continuación, les voy a compartir un video de simulación del primer ejemplo que hemos explicado en este capítulo, para que puedan ver en forma rápida todos los pasos realizados para el cambio de tiempos en el prendido y apagado del led.

25.0°C
TEMPERATURA

Capítulo 6
Declaración de dispositivos al Microcontrolador PIC

Bien, ya hemos transitado por más de la mitad de esta edición, dando los primeros pasos en la programación visual de microcontroladores PIC con Niple, y obtener las simulaciones de los archivos compilados de Niple en la Suite de Proteus, pudiendo observar detalles en la programación para poder corregir y retornar a compilar el archivo para adquirir el código correcto. Sin embargo, todavía tenemos un camino que seguir: −¡Allí vamos! −.

Comunicación RS232

El protocolo RS232 es utilizado para comunicación, con el puerto serial de las computadoras (puertos COM). Este módulo le permite enviar datos, mediante el protocolo de comunicación RS232 por consiguiente, a otros microcontroladores o a una PC.

Para configurar y activar una comunicación RS232, ingrese a la sección "Dispositivos/Comunicaciones". Seleccione de la lista de tipos de protocolos la opción "RS232".

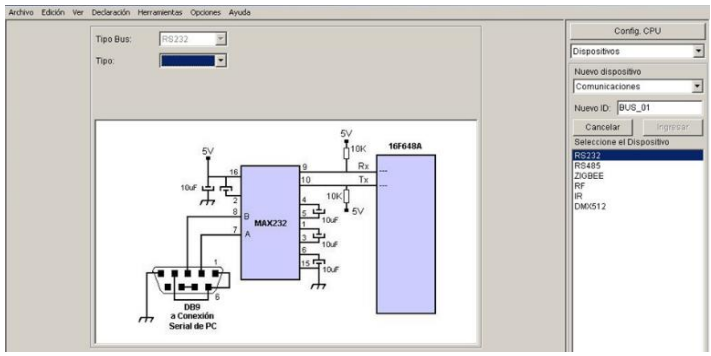


Fig. 6.1 Configuración dispositivo RS232

Seleccione el tipo de comunicación a realizar por lo cual puede ser mediante módulo USART.

La USART es un módulo de comunicaciones que algunos modelos de microcontroladores incluyen. Por medio del módulo USART, todo el procedimiento de envío de los datos se ejecuta de manera automática. Mediante el modo de envío "por código", se genera el código Assembler, necesario para efectuar el envío de los datos.

La diferencia radica, en que el módulo USART prepara el dato e inicia el envío y, luego retorna para seguir ejecutando el resto del código del programa, sin necesidad de hacer todo el proceso de envío del dato, dado que este se ejecuta de manera automática y, en forma paralela a la ejecución del resto del programa. Mientras que si se hace "por código", la ejecución del programa efectúa todo el proceso de envío del dato. Si selecciona el modo de envío "por USART", el sistema selecciona de modo automático el pin de transmisión correspondiente. Este pin no puede ser modificado, dado que el módulo USART utiliza un determinado pin para el envío de datos. Si selecciona el modo de envío "por Código", el sistema le solicita que seleccione el pin de transmisión,

por el cual quiere realizar la comunicación.

A continuación, debe seleccionar la velocidad de comunicación. Por último, seleccione el botón ingresar y ya habrá creado el bus de comunicación RS232.

Para llamar en el programa principal a la rutina enviar datos por RS232, debe ingresar al árbol de herramientas y al apartado de comunicaciones dentro de rutinas básicas.

Luego, debe seleccionar el nombre del bus que creó en la pantalla de dispositivos y configurar la acción.

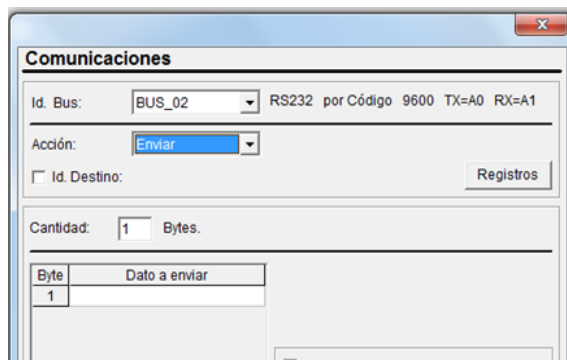


Fig. 6.2 Comunicaciones enviar datos por RS232

En el caso de seleccionar la Acción "Enviar", Niple le mostrará la siguiente ventana donde debe especificar, si desea la identificación del equipo del destino, la cantidad de bytes a enviar y, de que tipo es el dato a enviar ya sea un literal o un registro.

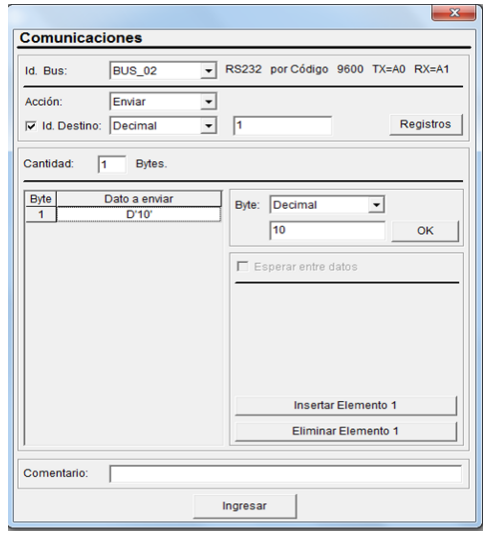


Fig. 6.3 Enviar dato Decimal por RS232

Por último, presione el botón Ingresar.

En el caso, de configurar la comunicación en modo de recepción de datos, debe especificarse parámetros similares al envío de datos y, debe seleccionarse el registro donde guardar los datos recibidos.

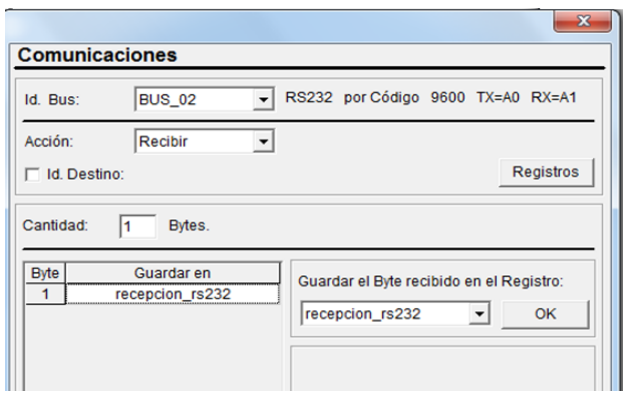


Fig. 6.4 Configuración recepción de datos de RS232

Comunicación RS485

La comunicación RS485 le permite conectar varios microcontroladores "en red". Generalmente, una red de microcontroladores está formada por un micro, denominado "Maestro", es el que se encarga de organizar el funcionamiento de la red, enviando comandos y solicitando información al resto de los micros que son nombrados "esclavos". En general, los dispositivos esclavos, responden enviando los datos solicitados por el maestro o ejecutan comandos enviados por el maestro.

Una red 485 puede estar formada por hasta 256 dispositivos esclavos, a una distancia máxima aprox. de 1200 metros y cada dispositivo esclavo, debe estar identificado por un "número de dispositivos".

En general, el micro maestro es un modelo de micro "grande" y se encarga de realizar los procesos principales (Ingreso de datos al sistema, interfaz con el usuario, LCD, display de 7 segmentos, conexión a PC, etc.) mientras que los dispositivos esclavos, por lo general, son micros más pequeños que realizan los procesos "distribuidos", es decir, los procesos locales propios de cada punto de control y, pueden contar con un alto grado de autonomía.

Niple soporta la interfaz DS75176 para hacer las comunicaciones RS485.

La comunicación RS485 requiere de 3 pines:

- Un pin de Transmisión (TX), configurado como salida
- Un pin de Recepción (RX), configurado como entrada
- Un pin de control de TX/RX de la interfaz DS75176 configurado como salida

Cuando un microcontrolador en modo "Maestro" solicita datos a otro microcontrolador en modo "Esclavo", las interfaces (DS75176) de ambos micros tiene que conmutar a modo "Transmisión" y "Recepción" respectivamente y, si el esclavo debe responder con algún dato al maestro, ambos micros tendrán que modificar el estado de sus interfaces ("Maestro" en modo recepción y "Esclavo" en modo Transmisión)

mediante el pin de control de la interfaz (DS75176). Y cuando termine el envío de datos al maestro, estos volverán a conmutar el modo de sus interfaces al estado inicial. Niple coordina de manera automática el cambio de modo de trabajo (TX/RX) de las interfaces.

A continuación, se muestra un esquema de la estructura de un proyecto típico de implementación de una red RS485.

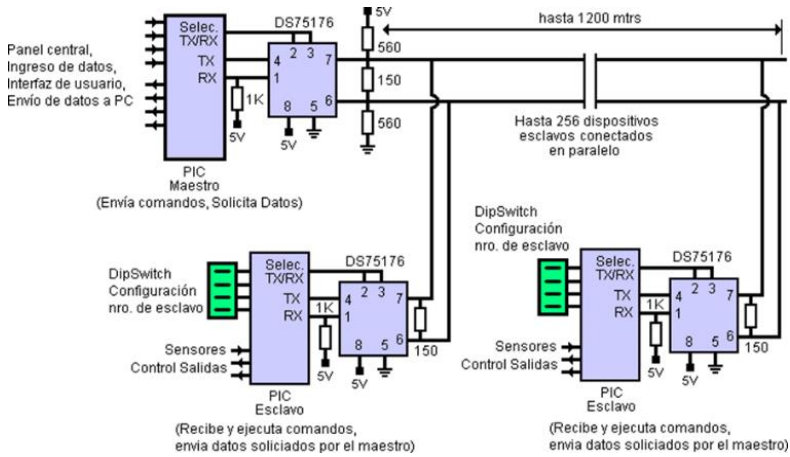


Fig. 6.5 Estructura típico de implementación dispositivo RS485

Modo Maestro

Para ingresar a los módulos de RS485, ingrese a la pestaña dispositivos/comunicaciones. Estando en la pestaña de dispositivos, diríjase a la pestaña comunicaciones:

Seleccione como tipo de BUS RS485.

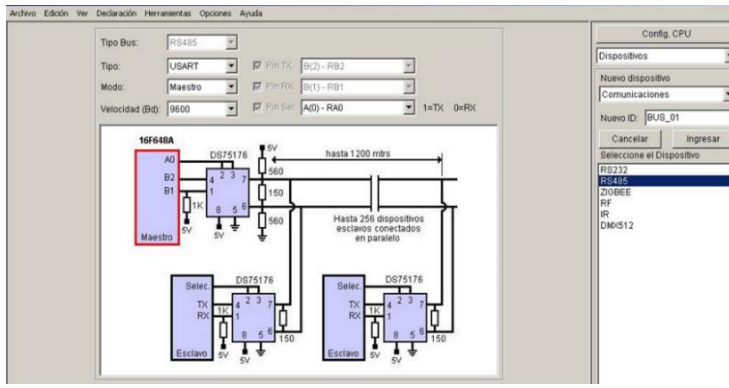


Fig. 6.6 Selección Bus RS485

Seleccione el modo de comunicación, por módulo USART o por código. NO todos los modelos de microcontroladores disponen de un módulo USART.

Si selecciona el modo de comunicación por USART, Niple selecciona de manera automática los pines correspondientes a TX y RX e inhabilita los mismos, porque los mismos ya vienen preestablecidos para ser utilizados por el módulo USART.

Luego seleccione la velocidad de comunicación.

Fig. 6.7 Velocidad de comunicación dispositivo RS485

Ya creado el módulo de comunicación RS485, ingrese al Árbol de herramientas y seleccione comunicaciones, una vez seleccionado pedirá que seleccionemos el tipo de comunicación creado.

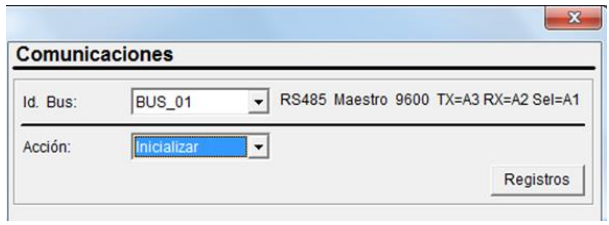


Fig. 6.8 Inicializar Bus de comunicación por RS485

Inicialice el módulo RS485.

Teniendo inicializado el Bus 485, tendremos que configurar, en el caso de elegir la acción, enviar las opciones como:

- Identificación del destino
- Cantidad de Bytes a enviar
- Datos a enviar

Fig. 6.9 Configuración Datos a enviar en RS485

En forma opcional, se puede seleccionar un ID de destino. Esto es ideal para trabajar en una RED 485, donde cada dispositivo esclavo, es identificado por medio de un código ID.
Para interpretar mejor cada una de las funciones disponibles, aclaramos los significados que damos a las siguientes palabras:

- **Enviar**: Enviar una orden a un esclavo, a partir del cual el esclavo realiza una determinada acción, pero esta acción no incluye el envío de una "respuesta" desde el esclavo hacia el maestro.
- **Interrogar**: Solicitar un dato a un esclavo, en general el esclavo responde inmediatamente, sin hacer ningún proceso antes de enviar los datos solicitados.

Habiendo explicado la acción enviar pasamos a la acción Recibir:

Recibir datos: Recibe datos desde un esclavo.

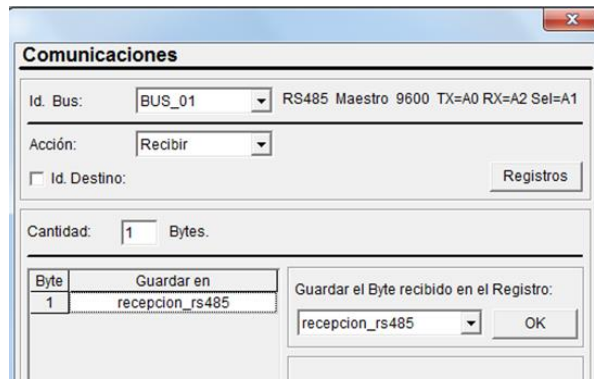


Fig. 6.10 Configuración recepción de datos en dispositivo RS485

Seleccione el registro donde guardaremos los datos y la cantidad de bytes que tiene ese dato.

Interrogar

Como detallamos anteriormente, la función interrogar solicita un dato a un esclavo, en general el esclavo responde inmediatamente, sin realizar ningún proceso antes de enviar los datos solicitados.

Para eso seleccionamos la acción Interrogar.

Fig. 6.7 Comunicaciones acción Interrogar en RS485

Tablas

Una tabla consiste en una metodología de programación, a través de la cual se pueden efectuar consultas, obteniendo un valor de resultado en función de un valor de entrada (índice).

Por ejemplo: La siguiente tabla devuelve el código ASCII correspondiente al valor ingresado como índice.

Indice	Valor Resultado
0	D'48
1	D'49
2	D'50
3	D'51
4	D'52
5	D'53
6	D'54
7	D'55
8	D'56
9	D'57

Tabla 6.1 Devolución en valor ASCII dato ingresado

El procedimiento consiste en sumar al registro contador de programa (registro PCL) el valor correspondiente al índice, esto produce que en la ejecución del programa "salte" la cantidad de posiciones, correspondiente al valor del índice. Para que la tabla devuelva un valor para cada ítem, esta debe estar formada por instrucciones "RETLW"

(retorno de subrutina con un valor en W) y debe ser llamada mediante una instrucción "CALL".

En el siguiente ejemplo, en código Assembler se describe el principio de funcionamiento de una consulta por tabla.

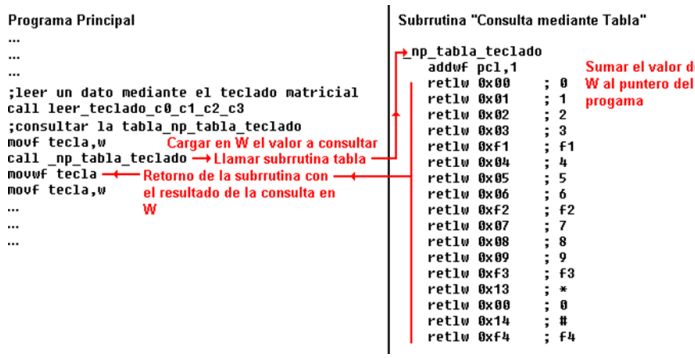


Fig. 6.11 Código Assembler correspondiente a la tabla 6.1

Crear una nueva tabla

Niple incluye un generador de tablas, en el cual Ud. declara la tabla y luego se graba como un archivo con extensión TBL. De esta manera Ud. podrá crear su propia librería de tablas.

Para originar una tabla, ingrese al menú Declaración / Generador de tablas / Tabla estándar.

Niple le muestra la pantalla de "declaración de tablas" y solicita que ingrese el nombre de la tabla. Cada tabla debe tener su correspondiente nombre. Una vez ingresado el nombre, a posterior se procederá a ingresar los valores de los distintos "elementos" que componen la tabla. Los valores de los índices, se incrementan de manera automática, ya que no se puede "saltear" ninguno de ellos.

Para ingresar nuevos elementos a la tabla, presione el botón "Agregar un valor".

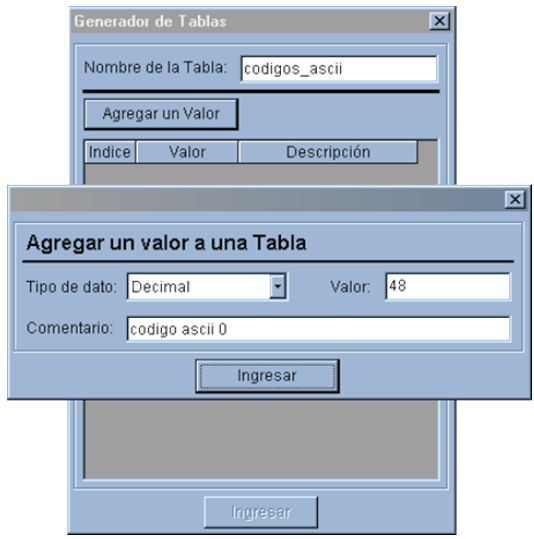


Fig. 6.12 Agregar un valor a una tabla

Si detecta que ha cometido un error al ingresar alguno de los valores de la tabla, puede editarlo haciendo doble clic sobre el elemento a corregir.

Una vez ingresados todos los valores que conforman la tabla, presione el botón "Ingresar". Al presionar el botón "Ingresar" Niple guardará la tabla actual, en un archivo con extensión .TBL

Seguidamente, le solicitará que ingrese el nombre del archivo de tabla.

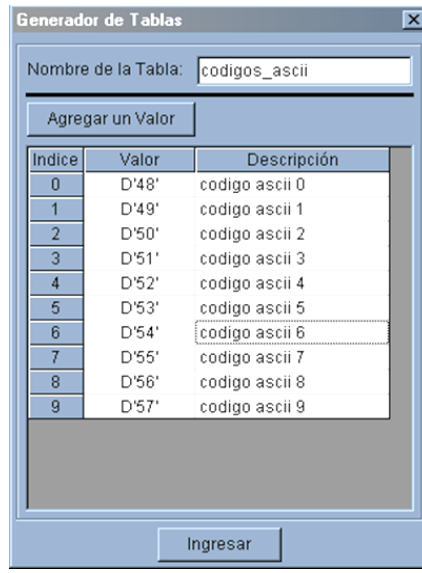


Fig. 6.13 Valores ingresados correspondientes a código ASCII de la tabla

De esta manera, podrá utilizar la tabla recién creada en distintos proyectos, sin necesidad de volver a declararla cada vez que la necesite.

Consultar Tabla

Cargar una tabla, la que deberá ser previamente definida.
Haga clic en Cargar Tabla.
Seleccione el archivo de tabla que desea consultar, haga clic en Abrir.
Aparecerá una pantalla con el detalle de los datos de la tabla.
En Valor a buscar, seleccione el registro que contiene el valor del puntero de la tabla.

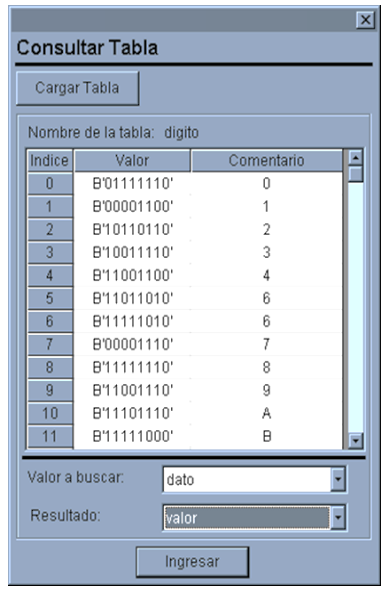


Fig. 6.14 Consulta de tabla

Como resultado, elija el registro en donde desea guardar el dato devuelto por la tabla. Haga clic en Ingresar.

Display de 7 segmentos sin Codificador
Declaración del Dispositivo

Este módulo permite visualizar valores en displays de 7 segmentos mediante conexión a 7 bits, es decir, sin la utilización de un codificador, para generar la combinación binaria correspondiente a cada valor.

Niple le permite utilizar hasta 16 displays conectados en paralelo.

Primero, seleccione el tipo de display a utilizar, esto es: Ánodo Común o Cátodo Común.

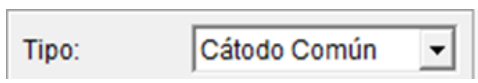


Fig. 6.15 Selección display cátodo común

Luego, se le solicita que seleccione la cantidad de displays a emplear esto es, entre 2 y 16.

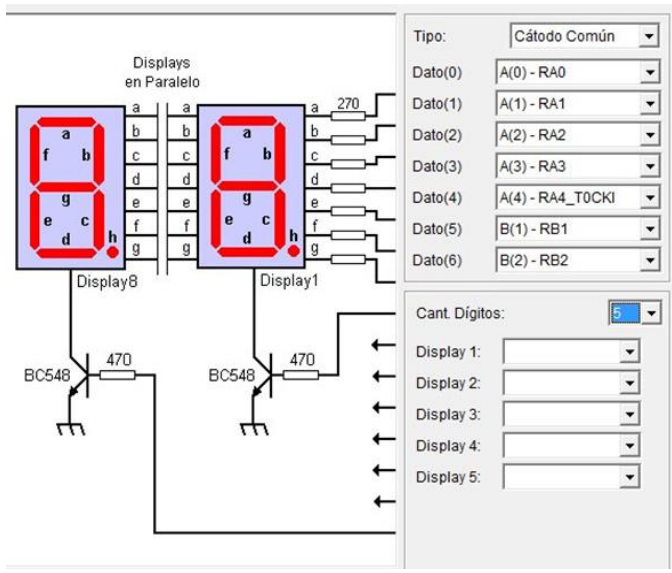


Fig. 6.16 Configuración display de 7 segmentos

Una vez configurado los pines de datos, cantidad de dígitos y los pines de Display, oprimimos el botón ingresar.

Display de 7 segmentos con Codificador
Declaración del Dispositivo

Puede implementar la utilización de Displays de 7 segmentos por medio de un codificador CD4511 para Displays Cátodo Común y el codificador 74LS47 para Displays Ánodo Común.

De esta manera, puede ahorrar 3 pines en el microcontrolador para poder utilizarlos en otras funciones.

Seleccione el tipo de Display a utilizar, los pines correspondientes al bus.

A continuación, seleccione la cantidad de Displays a emplear y sus respectivos pines.

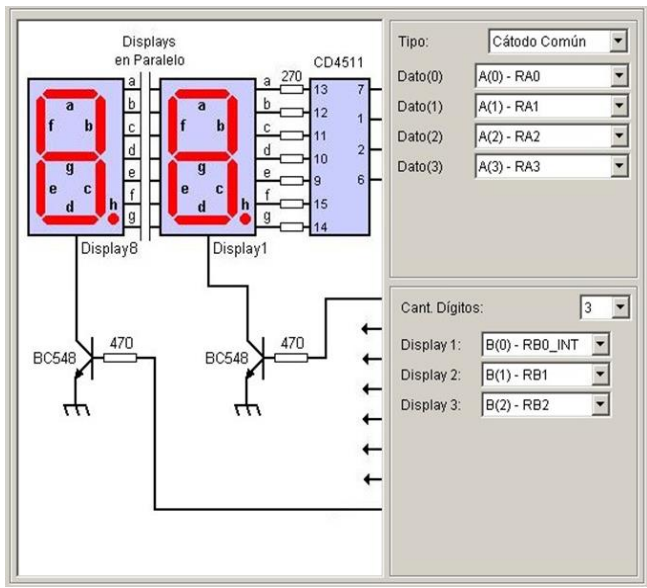


Fig. 6.17 Velocidad de comunicación dispositivo RS485

Una vez seleccionados todos los parámetros de configuración, presione el botón aceptar.

Implementación en el Diagrama de Flujo

Tanto para Displays de 7 segmentos con o sin codificador, la metodología de implementación en el diagrama de flujo es la misma:

Ingrese al árbol de herramientas:

Rutinas/Dispositivos/Visualización/Display de 7 segmentos.

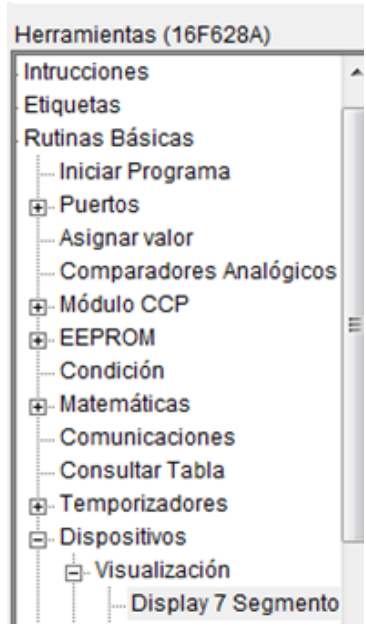


Fig. 6.18 Dispositivo Display de 7 Segmentos

Se muestra la pantalla de visualización en Displays de 7 segmentos.

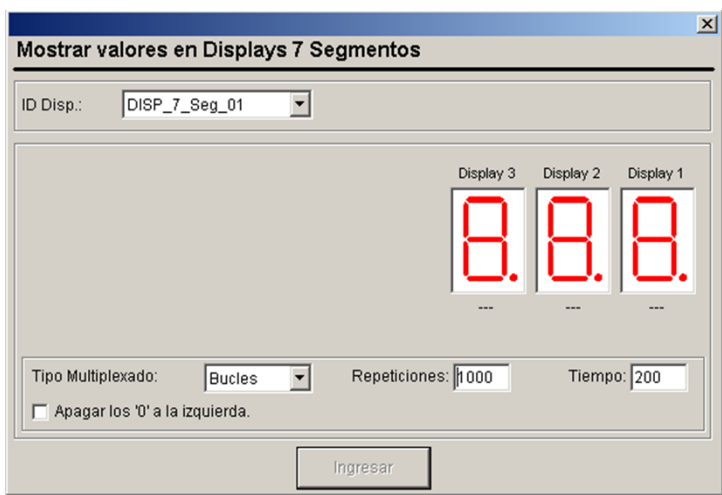


Fig. 6.19 Mostrar valores en Displays 7 Segmentos

En cada Display, se puede visualizar el valor de un registro o un valor literal.

Para seleccionar el valor a visualizar en cada dígito, debe realizar un doble clic en cada imagen de display. Al hacer doble clic en la imagen de un display se presenta la pantalla de "Mostrar un valor en un Display".

En un Display puede visualizar 2 tipos de valores:

1) Puede visualizar el valor que contiene un registro: Se trata de un valor que puede ser variable. Al seleccionar este tipo de dato a visualizar, Niple le ofrece la posibilidad de codificar el valor, mediante una tabla. Si el dato a visualizar requiere codificación mediante tabla, seleccione la tabla.

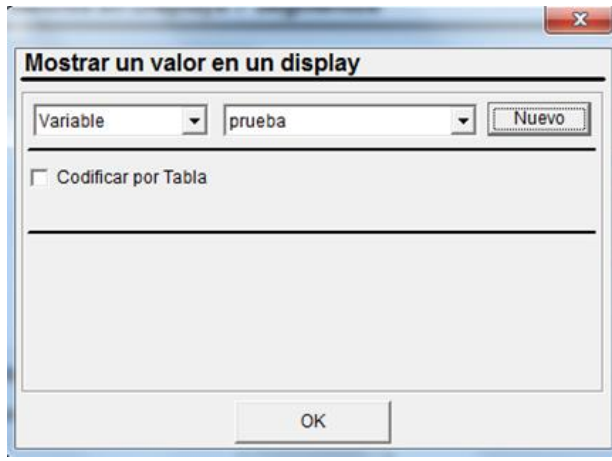


Fig. 6.20 Mostrar valor en Display

2) O puede visualizar un valor constante (un valor literal): en este caso se muestra siempre el mismo valor. Al seleccionar la opción "un valor constante", Niple presenta una imagen del display en el que debe realizar un clic en cada barra que quiere visualizar iluminada. De esta forma puede generar el número o símbolo que desea visualizar. Niple, genera el código binario correspondiente de manera automática. Este código binario

varía, si se trata de un display tipo Ánodo común o Cátodo común, Niple controla el código generado según el tipo de Display.

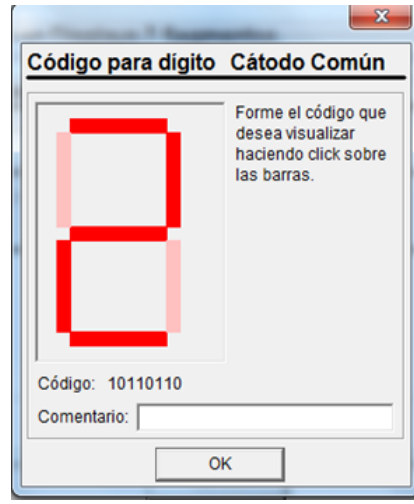


Fig. 6.21 Control de código para el digito generado

Una vez seleccionados los datos a mostrar en los Displays debe seleccionar el tipo de multiplexado que desea efectuar. Tiene 2 opciones: "por Bucles" o "por Paso".

La opción "por Bucles" hace que el microcontrolador se detenga en esta parte del programa y haga todo el proceso de multiplexado completo, mostrando los valores de cada Display y, no continua la ejecución del resto del programa hasta que no haya terminado de hacer este proceso.

Fig. 6.22 Configuración de Bucles en display de 7 segmentos

Las opciones de este tipo de multiplexado son:

a) Repeticiones: corresponde a la cantidad de veces que realiza el ciclo de visualización de todos los Displays.

b) Tiempo: tiempo en uS que demora cada Display encendido en el proceso de multiplexado.

El tipo de multiplexado "Por Paso" muestra el valor correspondiente a uno de los Displays (por ejemplo, el Display 1) y continua con el resto del programa. Cuando vuelve a pasar por este, en el siguiente ciclo del programa activa el siguiente Display y continúa con la ejecución del resto del programa.

Esta opción es ideal para hacer un multiplexado por medio de un temporizador interno (TMR) en el que cada vez que se produce la interrupción por TMR, se ejecuta el bloque y va activando 1 Display en cada interrupción. Así, puede ejecutar un multiplexado mientras hace otros procesos de manera simultánea.

Esta metodología de implementación del multiplexado "Por paso" y temporizando por medio de una interrupción TMR, es la opción recomendada para realizar visualizaciones de manera profesional.
Opcionalmente, Niple le permite "Ocultar" o "Encender" los dígitos con "0" a la izquierda.
Niple, controla automáticamente cuando tiene que mostrar u ocultar un dígito, en función del valor total a mostrar.

Pantalla LCD

Niple le permite controlar más de 1 pantalla LCD, de manera simultánea en el mismo proyecto. Esto se obtiene, conectando las pantallas LCD en paralelo y, los pines de control de cada pantalla se asignan a distintos pines del microcontrolador. Niple, administra las rutinas de control de manera automática. Ud., solamente índica que mensaje desea visualizar,

en cada una de las pantallas LCD.

Como se destaca, en Niple, tanto las configuraciones de las pantallas LCD, las configuraciones y diseños de los de los mensajes y los comandos se procede de manera visual y en alto nivel de programación. Niple, genera en su totalidad, el código Assembler necesario de manera automática.

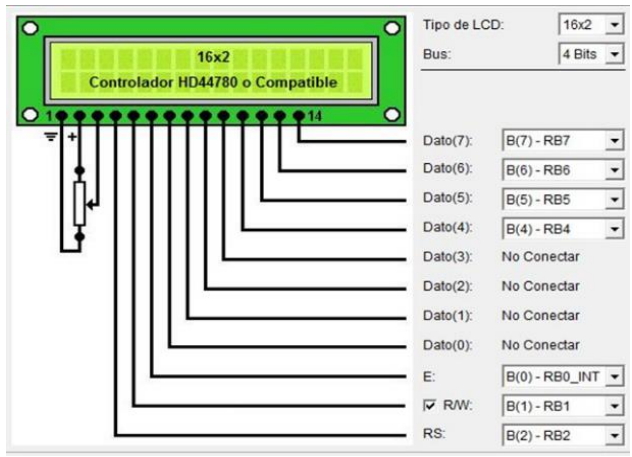


Fig. 6.23 Configuración bits de puerto LCD 16x2

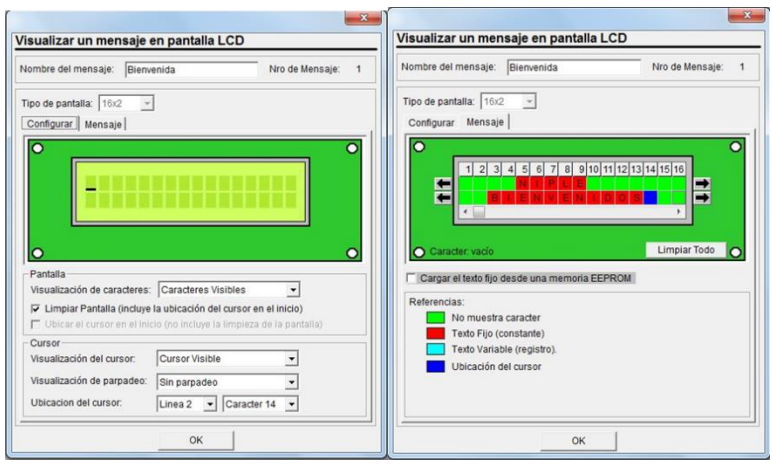


Fig. 6.24 Mostrar un mensaje en LCD 16x2

Principales características:

- Control de múltiples Pantallas LCD de manera simultánea.
- Configuración visual de mensajes.
- Bus de datos a 4 y 8 bits configurable por el usuario.
- Mensajes con valores variables.
- Conversión automática de formato numérico a código ASCII para valores variables.
- Edición y ajuste de mensajes y comandos de manera visual.
- Envío de comandos al LCD:
    - ✓ Apagar la Pantalla
    - ✓ Encender la Pantalla
    - ✓ Limpiar la pantalla
    - ✓ Ubicación del cursor a Inicio
    - ✓ Ubicación del cursor
    - ✓ Configuración del Cursor
    - ✓ Desplazamiento del cursor
    - ✓ Desplazamiento de la Pantalla
- Envío de caracteres al LCD de manera independiente de un mensaje (para ingreso de datos)
- Código Assembler optimizado: Cuando actualiza un mensaje en un LCD, si el mensaje con- tiene valores variables, Niple solo actualiza las variables en lugar de volver a cargar todo el mensaje nuevamente. De esta manera Niple le ahorra tiempo durante la ejecución del programa.

Para utilizar un LCD debe realizar los siguientes pasos:

1. Configurar el Puerto donde conectar el LCD
2. Configurar el LCD
3. Generar el mensaje
4. Visualizar el mensaje
5. Opcionalmente, se pueden enviar comandos o caracteres independientemente de un mensaje.

Configurar LCD

Una vez configurado el puerto en donde conectar el LCD deberá proceder a configurar el LCD en el siguiente orden:
Seleccione el bloque donde va a vincular el nuevo, de configuración de LCD. Ingrese a la pestaña de dispositivos:
Nuevo dispositivo/LCD.
En Niple, la configuración de un LCD consiste en seleccionar de qué manera será conectado el LCD al microcontrolador.
La configuración de un LCD se realiza en forma única, 1 vez.
Dado que Niple puede controlar más de 1 LCD de manera simultánea, en esta pantalla se presenta el número de LCD que está siendo configurado.

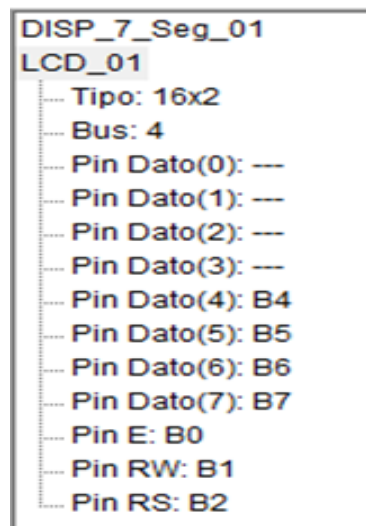


Fig. 6.25 Configuración Pines para LCD

Primero, debemos seleccionar el tipo de display LCD ya sea 16 x 2, 20 x 2 o 20 x 4. Luego, debe seleccionar el tamaño del bus de datos, el cual puede ser de 4 u 8 bits.
Se recomienda utilizar el bus de datos a 4 bits a fin de que la diferencia con el bus a 8 bits es insignificante, tanto en el código generado como en el tiempo de ejecución del programa. Por el contrario, los 4 pines que se

ahorran, pueden ser de gran importancia.

Fig. 6.26 Selección tipo de LCD y Bus

Una vez seleccionado el tamaño del bus de datos, hay que seleccionar los pines en donde se conectara el LCD.
Si selecciona el bus de datos a 4 bits, Niple, le ofrecerá la posibilidad de usar los 4 pines, de mayor o menor peso de un mismo puerto. Esto simplifica el código Assembler generado.

Además, indica en la imagen del LCD los pines que deben ser conectados.

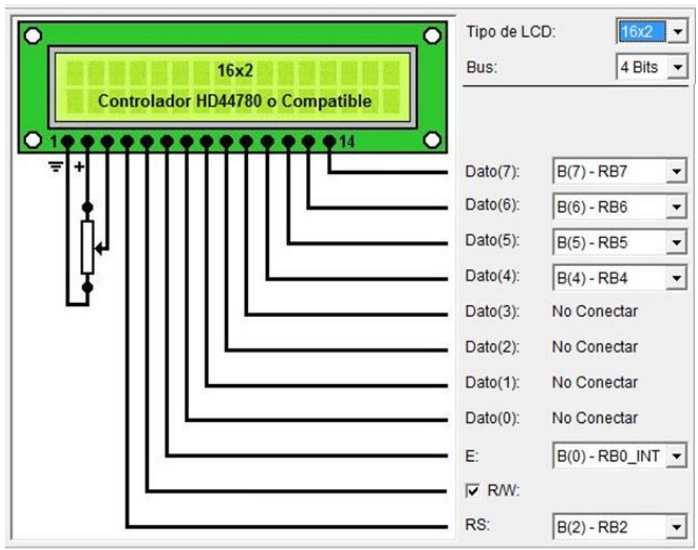


Fig. 6.27 Pines del Puerto B para conexión a LCD

Si selecciona el bus de datos a 8 pines, Niple, emplea un puerto completo.
Si ha seleccionado el bus de datos a 4 bits, Niple, le solicitará que

seleccione los pines correspondientes. En caso de que usted, ha seleccionado el bus de datos a 8 bits, Niple, asignará al bus de datos, todos los pines del puerto seleccionado.
Por último, debe seleccionar los pines del control del LCD. Para esto haga un clic en la casilla de verificación correspondiente a cada pin de control (RS, RW, E).

Fig. 6.28 Pines de control (RS, RW y E)

Una vez ingresados todos los datos solicitados, presione el botón Ingresar. Luego diríjase a la sección diagrama y vaya al árbol de herramientas Rutinas básicas / Dispositivos / Visualización / Pantalla LCD.

La configuración de la pantalla LCD se realiza solo una vez en un proyecto.

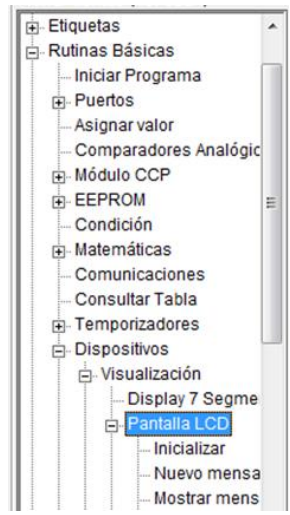


Fig. 6.29 Selección dispositivo Pantalla LCD

Utilizando más de una pantalla LCD

Niple, le permitirá controlar más de una pantalla LCD. Para esto, los pines correspondientes al bus de datos y los pines de control RS y RW de las pantallas a utilizar, deberán estar conectados en paralelo.
Solo debe conectar de manera independiente los pines correspondientes al control "E" de las pantallas. El pin "E" de un LCD es el encargado de habilitar e inhabilitar una pantalla LCD. De esta manera, Niple habilita e inhabilita, las distintas pantallas LCD de manera automática, de acuerdo al programa que Ud. está desarrollando.
Si desea utilizar más de un LCD, el mismo debe ser creado y configurado desde la pestaña Dispositivos/LCD
Recuerde conectar el display LCD en paralelo con los demás LCD y solamente deberá cambiar el pin correspondiente al control "E" para el nuevo LCD.

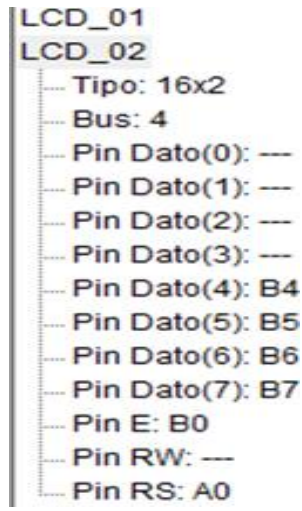


Fig. 6.30 Pines de conexión a LCD

Seleccione el BIT adecuado al pin de control E del LCD. Por último, presione el botón "Ingresar".
Cuando termine de configurar la cantidad de dispositivos LCD, que desee utilizar, debe ingresar al diagrama principal y luego ir al árbol de herramientas, para inicializar las pantallas LCD.

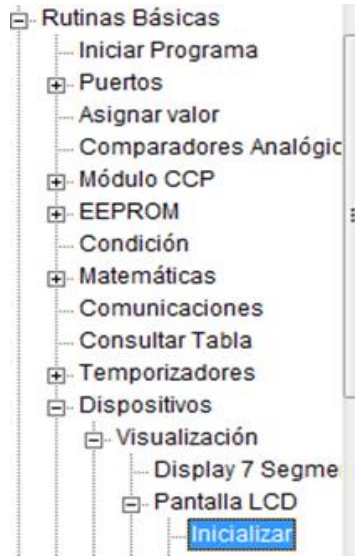


Fig. 6.31 Inicializar LCD

Al inicializar las pantallas se le crearán los siguientes iconos en el programa principal:

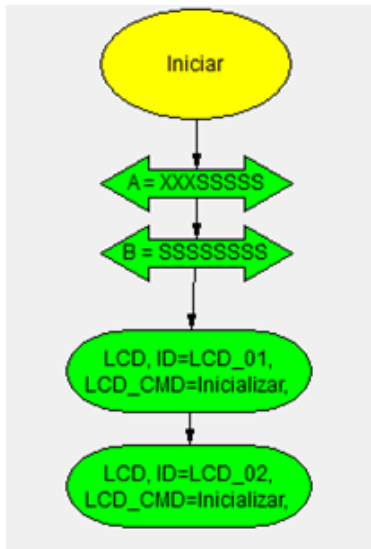


Fig. 6.32 Diagrama de bloque LCD

Generar un nuevo mensaje

La metodología de visualización de mensajes en Niple, consiste en generar un mensaje. Esto, implica configurar la pantalla, el cursor y diseñar el mensaje propiamente dicho.
Una vez "diseñado" el mensaje, puede ser aplicado a cualquier LCD, mediante el módulo de visualización de mensajes.
Para generar un nuevo mensaje, ingrese al árbol de herramientas a la sección:
Insertar/Rutinas Básicas/dispositivos/visualización/Pantalla LCD/Nuevo Mensaje

En Niple, cada mensaje es identificado con un nombre único. Por ejemplo "Bienvenida".
Al cargar la pantalla de nuevos mensajes, lo primero que debe ingresar es el nombre del mensaje.

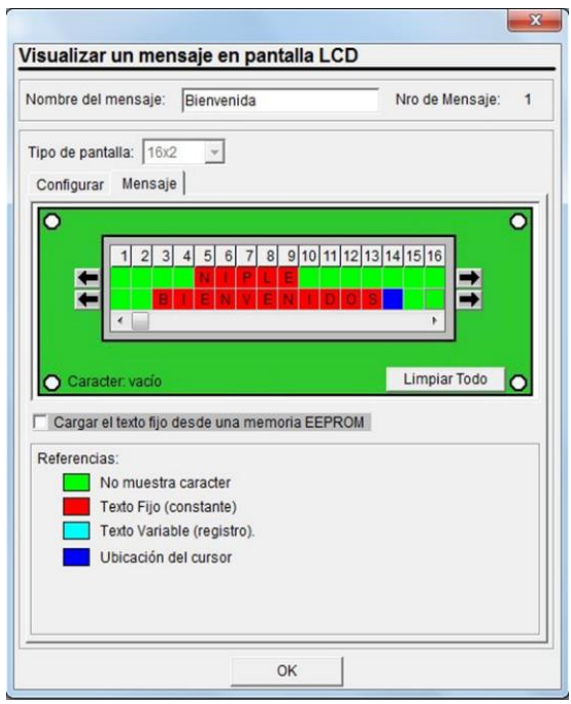


Fig. 6.33 Visualizar nuevo mensaje en LCD

Además, muestra el número de mensaje correspondiente.
Una vez ingresado el nombre, Niple, mostrará las opciones de configuración de pantalla.

Por defecto, la opción "Visualización de Caracteres" se presenta como "Caracteres Visibles". Niple, supone que Ud. quiere que el mensaje sea visible al momento de "llamar" al mensaje.

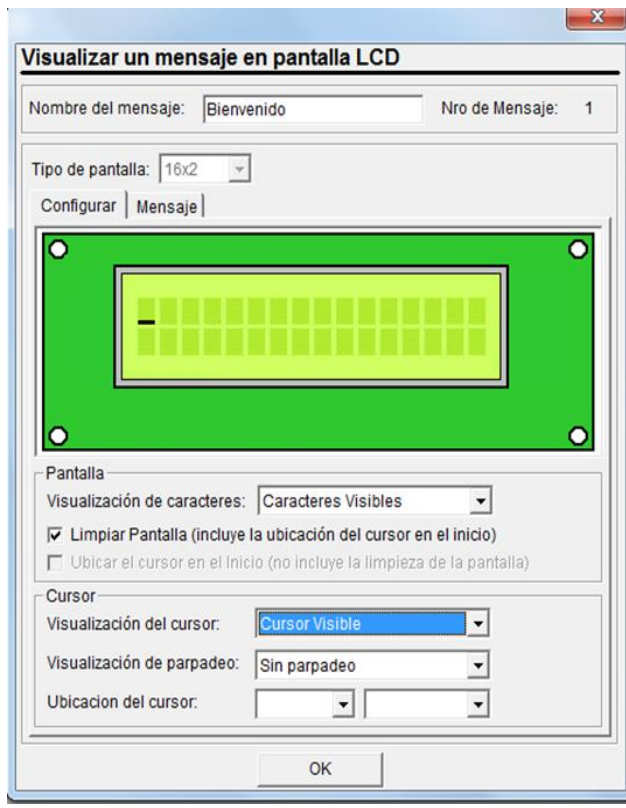


Fig. 6.34 Configuración cursor visible, en mensaje LCD

Si por algún motivo Ud. desea que el mensaje se cargue en el LCD, pero, que permanezca invisible, para luego mostrarlo con el comando "Encender Pantalla" (observar envío de comandos más adelante), seleccione la opción "Caracteres Invisibles".

Además, la opción "Limpiar pantalla" se presentará como activada.
Esta opción realiza una limpieza de la pantalla antes de visualizar el mensaje. Esto significa que será borrado completamente cualquier mensaje que haya sido visualizado antes de mostrar el actual mensaje.
Si esta opción se inhabilita, Niple, no hace la limpieza de la pantalla y existe la posibilidad de "superposición de mensajes".
Al inhabilitar la opción "Limpiar Pantalla" Niple advierte lo anteriormente mencionado, mediante el siguiente mensaje.

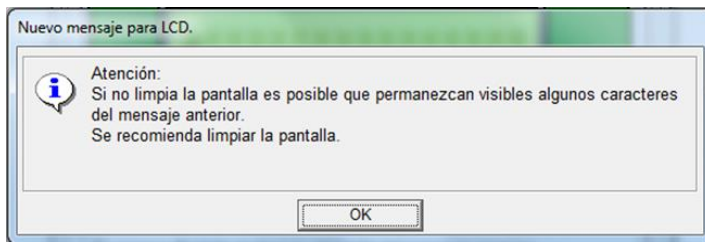


Fig. 6.35 Mensaje emergente de limpiar pantalla

Dado que el objetivo de Niple, es minimizar al máximo la posibilidad de error, ofrece esta opción como habilitada para que proceda a la limpieza de la pantalla, antes de visualizar el mensaje actual.
La limpieza de la pantalla incluye la ubicación del cursor al inicio del LCD, esto es, en la línea 1, carácter
Por esto, si la opción "Limpiar Pantalla" se encuentra activada, entonces, la opción "Ubicar el cursor en el Inicio" se encuentra inhabilitada, dado que la limpieza de pantalla incluye esta función.
Por esto, si la opción "Limpiar Pantalla" se desmarca, se activa la opción "Ubicar el cursor".

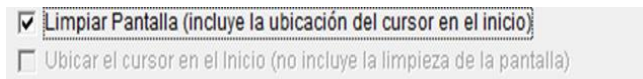


Fig. 6.36 Selección limpiar pantalla en LCD

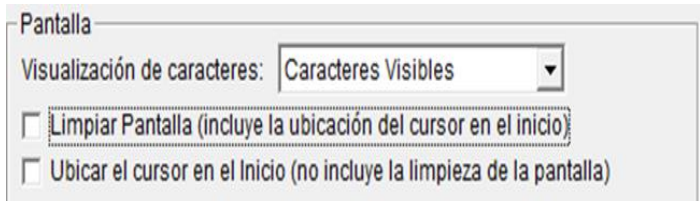


Fig. 6.37 Configuración visualización de caracteres en LCD

A continuación, debe seleccionar la configuración del cursor.

Si selecciona la opción "Cursor Invisible". La opción "Visualización de parpadeo" se configura de manera automática como "Sin parpadeo".

Seleccione la configuración del cursor, el parpadeo y la ubicación del mismo.

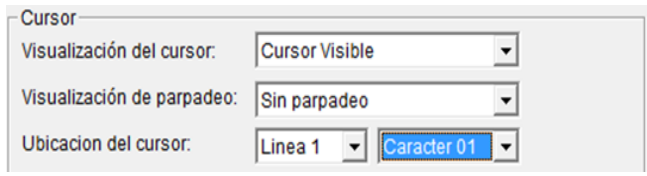


Fig. 6.38 Configuración cursor en LCD

Una vez seleccionadas todas las opciones de la configuración del mensaje, se activa la "Solapa Mensaje".

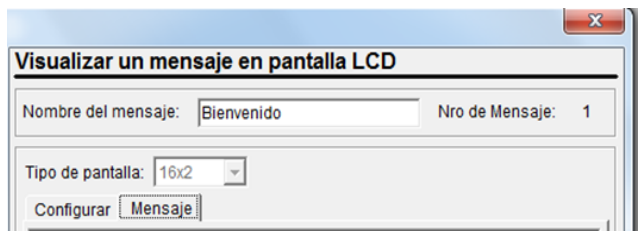


Fig. 6.39 Visualizar mensaje de bienvenida en LCD

Acto seguido, deberá ingresar el mensaje propiamente dicho.

Se presenta una imagen de la pantalla LCD, representada por una grilla de 2 filas, en la cual se enumeran las columnas las que representan a los caracteres.

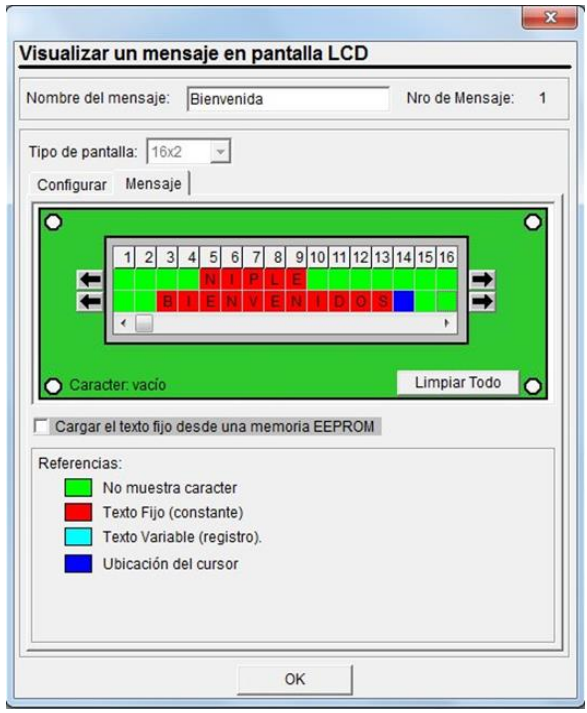


Fig. 6.40 Grilla de visualización de mensaje en LCD

En la grilla, ingrese el mensaje que desea visualizar.

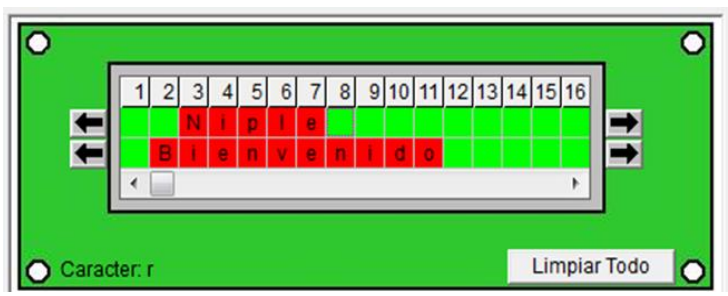


Fig. 6.41 Mostrar mensaje de bienvenida

Como lo indican las referencias, el texto del mensaje se muestra en color rojo.

A la izquierda y a la derecha de la grilla del mensaje, se encuentran botones para desplazamiento del texto ingresado. De esta manera puede ajustar la visualización del texto.

En la siguiente imagen, el texto ha sido desplazado hacia la derecha para ajustarlo al centro de la pantalla.

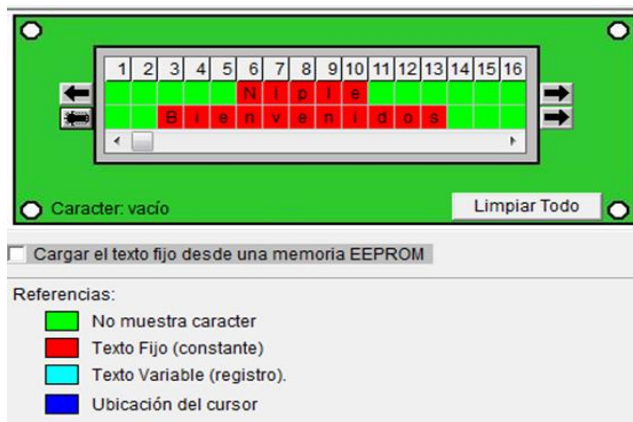


Fig. 6.42 Mensaje en LCD "Niple Bienvenidos"

El botón "Limpiar Todo" le permite limpiar la grilla del texto cuando Ud. desea corregir el mensaje.

Visualizando valores variables en un mensaje

Para ingresar un valor variable en un mensaje, haga doble clic en la celda donde desea visualizar la variable. Se presentará la pantalla de selección de variables:

Esta pantalla le solicita que seleccione el registro donde tiene almacenado el valor variable a visualizar. Si el valor variable corresponde a un número,

antes de ser enviado al LCD debemos convertirlo a su correspondiente código ASCII, ya que el LCD solo puede mostrar caracteres ASCII. Además, si el número es mayor a 9 debemos enviar realizar la conversión a BCD y enviar los valores correspondientes a las unidades, las decenas, las centenas, etc., por separado.

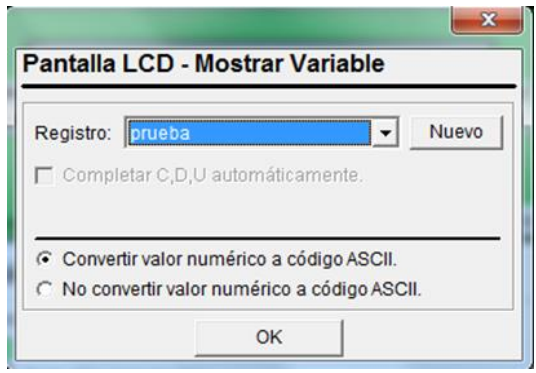


Fig. 6.43 Mostrar mensaje "prueba"

Niple, le ofrece la posibilidad de convertir en número a código ASCII de manera automática. Para esto debe seleccionar la opción "Convertir valor numérico a código ASCII". La cual se encuentra seleccionada de manera predeterminada.

Si lo que desea enviar, no corresponde a un valor numérico, sino a un código ASCII que Ud. ya tiene definido en el programa, debe marcar la opción "No convertir valor numérico a código ASCII".

Al ingresar la variable, la misma se presenta en la grilla de mensaje, como un símbolo de interrogación de color celeste. Se muestra como un símbolo de interrogación, dado que Niple no puede "predecir" que valor será mostrado en la correspondiente variable.

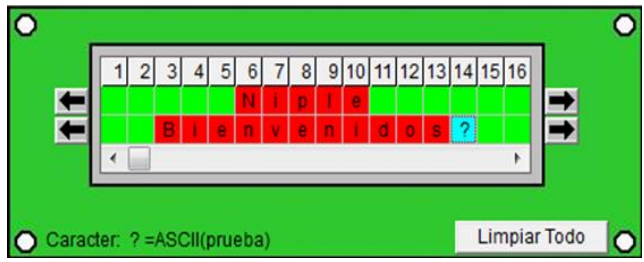


Fig. 6.44 Carácter "?" representa carácter variable en LCD

Si desea consultar qué valor corresponde a una variable. Haga un clic sobre la celda donde se encuentra la variable y, Niple le mostrará a que valor corresponde la misma, como lo muestra la imagen anterior.

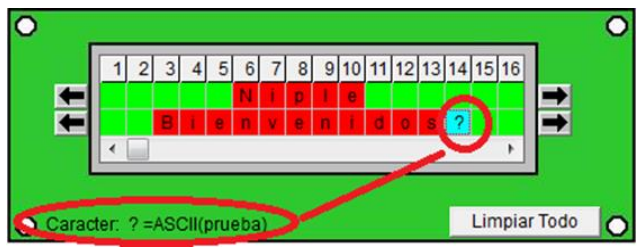


Fig. 6.45 Carácter (?)=ASCII (prueba)

El formato del texto puede ser uno de los 2 siguientes:

1) ? = ASCII(Nombre_Registro): Esto significa que la variable corresponde al carácter ASCII del valor que contiene el registro "Nombre_Registro".
2) ? = Nombre_Registro: Esto significa que la variable corresponde al valor que contiene un registro sin convertirlo a código ASCII.

Por último, presione el Botón "OK".

Al insertar un nuevo Mensaje de LCD, Niple genera un nuevo entorno para el mensaje. De esta manera, el mensaje puede ser tratado como una rutina de usuario.

Al insertarse el mensaje como una rutina de usuario, el mismo puede ser

"llamado" desde cualquier parte del programa y puede ser visualizado en cualquier LCD, en caso de estar utilizando más de 1 LCD.

Además, para minimizar al máximo las posibilidades de errores de visualización, Niple, incluye la configuración de la pantalla junto con el mensaje a visualizar. De esta forma, se garantiza que la pantalla se pueda encontrar configurada, de acuerdo a las necesidades del mensaje a visualizar.

Ud. no debe preocuparse por la configuración de la pantalla, o la configuración y ubicación del cursor, cada vez que desea visualizar el mensaje. Simplemente, muestra el mensaje que necesite visualizar.

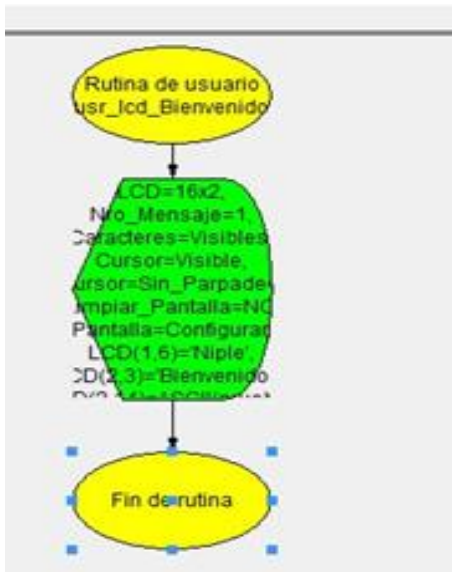


Fig. 6.46 Diagrama de bloque rutina usuario LCD bienvenida

Mostrar Mensaje

Una vez definido el mensaje, puede observarlo en cualquier parte del programa ingresando al árbol de herramientas a:

Rutinas Básicas/dispositivos/visualizacion/Pantalla LCD/Nuevo Mensaje

Se presentará la pantalla "Mostrar un mensaje en LCD".

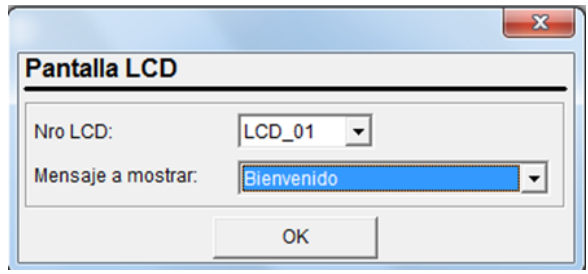


Fig. 6.47 Mostrar mensaje Bienvenido en LCD

Primero debe seleccionar el LCD, en el cual desea visualizar/observar el mensaje y a continuación indique el nombre del mensaje a visualizar. De esta manera, un mismo mensaje puede ser visualizado en cualquier LCD.

Al presionar el botón "OK", Niple, ingresa el bloque del LCD.

Optimización del código Assembler

Niple incluye un algoritmo de optimización para ahorrar recursos en tiempo de ejecución (del PIC). Niple, administra los mensajes que están siendo visualizados en cada LCD y "sabe" en todo momento que mensaje se está visualizando en cada LCD. Incluso si un proyecto incluye solo 1 LCD, Niple sabe qué mensaje está siendo visualizando en el LCD.

Cuando desea visualizar un mensaje en un LCD que ya está siendo visto en dicho mensaje, Niple, actualizara espontáneamente los valores variables (en caso de que el mensaje incluya variables). Es expresar que no vuelve a "cargar" todo el mensaje en el LCD. De esta manera, se ahorra tiempo de ejecución.

Lo mismo ocurre con la configuración de la pantalla y la configuración y ubicación del cursor. Solamente se ejecutan estas configuraciones cuando el mensaje no está siendo visualizado en el LCD actual.

Comandos

Enviar comandos

Niple, le permite enviar una serie de comandos al LCD, indicando el nombre del comando deseado. Para facilitarle aún más el manejo de las pantallas LCD, algunos comandos le permiten realizar reconfiguraciones de pantalla o cursor.

Los comandos disponibles son los siguientes.

- Apagar la pantalla: Este comando apaga los caracteres. No incluye opciones.
- Encender la pantalla: Visualiza los caracteres. Opcionalmente, puede ubicar y reconfigurar el cursor.

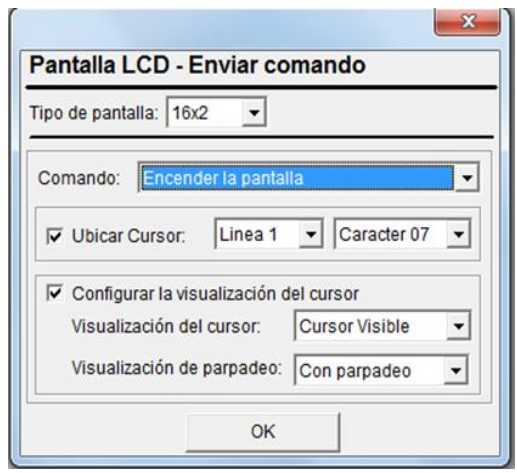


Fig. 6.48 Enviar comando por Pantalla LCD

- Limpiar la pantalla: Limpiar todos caracteres de la pantalla. No incluye opciones.
- Cursor a Inicio: Ubica el cursor al inicio de la pantalla LCD, esto es, al renglón 1, carácter 1. Este comando no incluye opciones.

- Ubicar el cursor: Ubica el cursor en la posición que indique el usuario. Opcionalmente, puede reconfigurar la visualización del cursor.

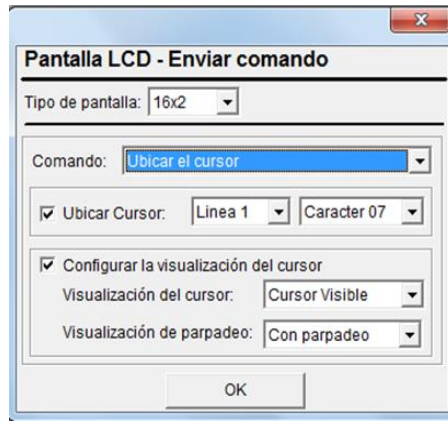


Fig. 6.49 Ubicar curso en Pantalla LCD

- Configurar el cursor: Permite reconfigurar el cursor. Este comando requiere que necesariamente se configure la visualización de los caracteres.

Fig. 6.50 Configuración cursor en Pantalla LCD

- Desplazar el cursor: Permite realizar un desplazamiento del cursor hacia la izquierda o hacia la derecha. Opcionalmente, puede ingresar un retardo entre cada desplazamiento.

Fig. 6.51 Desplazar cursor en Pantalla LCD

- Desplazar la pantalla: Permite hacer un desplazamiento de la pantalla hacia la izquierda o hacia la derecha. Como opción, puede ingresar un retardo entre cada desplazamiento. Este comando produce un efecto visual de deslizamiento del mensaje.

Enviar caracteres

Además de configurar la visualización de mensajes, Niple le permite enviar caracteres independientemente de un mensaje.

Esto es frecuente cuando se efectúan proyectos con ingreso de datos, los cuales son mostrados en la pantalla LCD. Por ejemplo; Ud. diseña el mensaje de solicitud de ingreso de datos. En general, en estos casos, configura el cursor como visible e intermitente y lo ubica en la posición

donde el usuario debe "ingresar el mensaje" y luego espera el ingreso de los datos. Cada vez que el usuario ingresa un dato, el microcontrolador lo captura, por interrupción, por ejemplo, y lo envía a la pantalla LCD donde el usuario puede visualizar los datos que está ingresando.

Este "envío" de datos al LCD, corresponde a la función de Niple de envío de carácter.

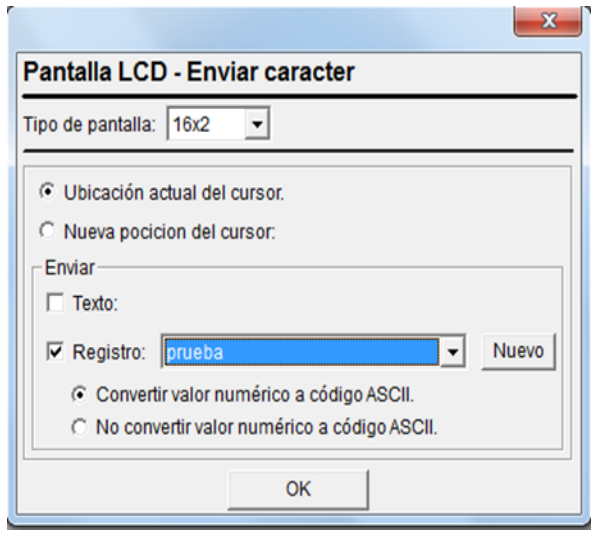


Fig. 6.52 Configuración envió de carácter por Pantalla LCD

Niple le permite enviar un carácter o un texto.

Además, le permite enviar los caracteres a la actual posición del cursor o definir una posición de visualización determinada. Si necesita enviar un valor numérico guardado en un registro, Niple le ofrece la posibilidad de convertirlo a código ASCII de manera automática.

Teclado Matricial

El módulo de teclado matricial permite capturar el valor de la tecla presionada en un teclado matricial de 3x4 o de 4x4 y lo guarda en un registro.

Para ello utiliza interrupciones por cambio de estado en RB4 a RB7.

Niple, verifica que la interrupción por RB4-7 se encuentre activada. Si la interrupción no se encuentra activada, no permite el ingreso del módulo.

Teniendo en cuenta que se utiliza la interrupción por RB4-7, Niple requiere que los pines correspondientes a las columnas del teclado matricial, sean conectados a los pines RB4-7, mientras que le permite seleccionar libremente la conexión de los pines correspondientes a las filas.

Al ingresar a la pantalla de teclado matricial, Niple le solicitará, que seleccione el registro en el cual se guardará el valor de la tecla presionada.

Luego, le solicitará que seleccione el tipo de teclado matricial que desea conectar.

Y por último, que seleccione los pines correspondientes a las filas del teclado matricial.

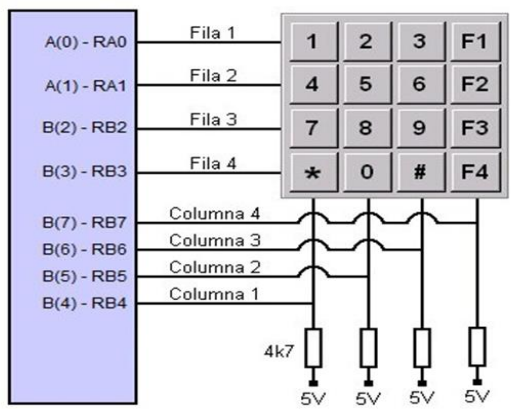


Fig. 6.53 Esquema teclado matricial

Memorias EEPROM I2C

Declaración del Dispositivo

Ingrese la sección "Dispositivos" y seleccione Nuevo dispositivo tipo "Memoria".

Seleccione el modelo de la memoria a utilizar y elija los pines donde conectar la memoria al PIC.

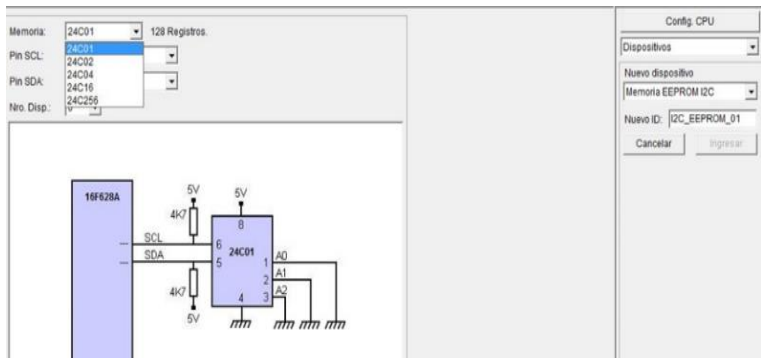


Fig. 6.54 Esquema memoria EEPROM I2C

Ingrese el número de dispositivos y, finalmente, haga clic en el botón "Ingresar" en el panel de la derecha.

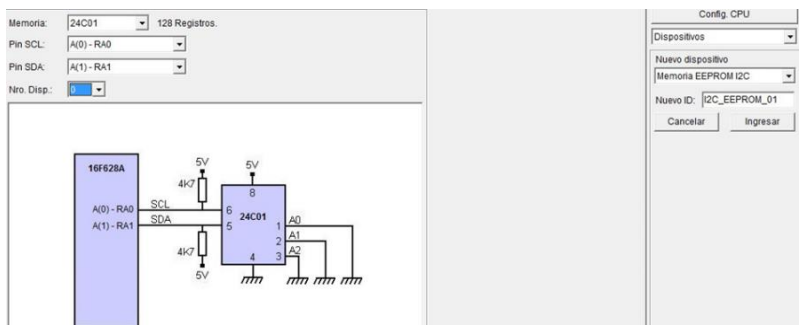


Fig. 6.55 Selección memoria 24C01

Recuerde que puede conectar varios dispositivos en paralelo, formando una red, en la cual, a cada dispositivo se le asigna un número. El número del dispositivo, se configura conectado los pines correspondientes de cada dispositivo. La cantidad de dispositivos que se puede incluir en una red, dependerá del modelo de memoria que está utilizando. Niple, controla de manera automática, la cantidad de dispositivos que se puede conectar en función del modelo seleccionado.

Leer Memoria EEPROM I2C

Ingrese al árbol de herramientas: Rutinas Básicas/Dispositivos/Memoria EEPROM. Se presenta la pantalla "Memorias EEPROM" y selecciona la acción "Lectura"

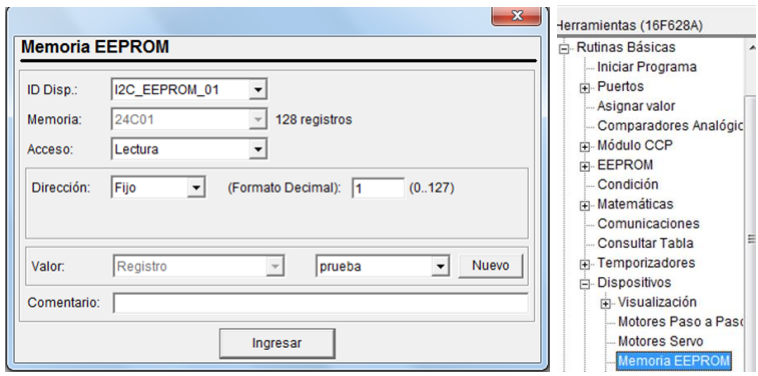


Fig. 6.56 Leer memoria EEPROM

Luego debe seleccionar la dirección de memoria que desea leer desde la memoria EEPROM. Primero, deberá seleccionar el tipo de dirección, la que puede ser "Fija" o "Variable".

Si selecciona el tipo de dirección "Fija" habrá que ingresar, la posición de memoria a leer en formato decimal. Niple, controla que la dirección ingresada no supere el límite máximo, de cantidad de registros, que contiene el modelo seleccionado. Mediante este tipo de acceso (por dirección fija) el microcontrolador accederá a la dirección ingresada, cada

vez que se ejecute el bloque actual.

Si selecciona el tipo de dirección "Variable" tendrá que seleccionar, el registro que contiene, el valor correspondiente a la posición de memoria que desea leer.

En este caso, Niple, no controla si la dirección ingresada no supera el límite máximo, de cantidad de registros que contiene el modelo seleccionado, dado que no se puede predeterminar, que valor tendrá el registro seleccionado al ejecutarse el programa.

Mediante este tipo de acceso (por dirección variable) el microcontrolador accederá a la dirección correspondiente al valor que contiene el registro cada vez que se ejecute el bloque actual.

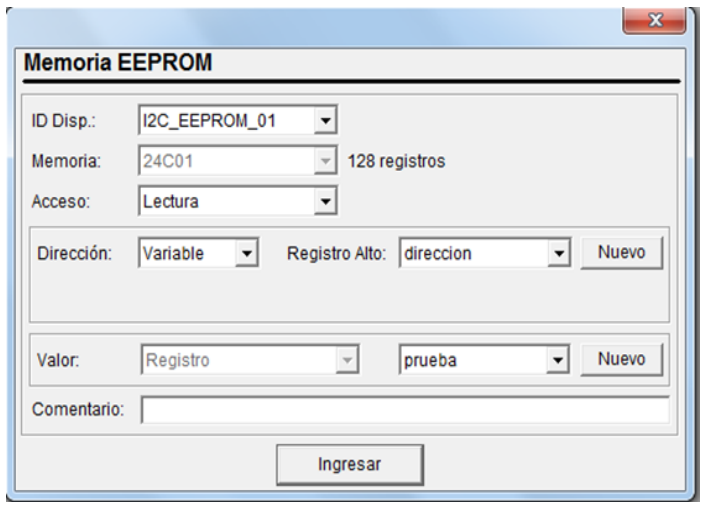


Fig. 6.57 Configuración tipo de acceso a memoria EEPROM

Una vez seleccionada la dirección a leer, debe escoger el registro donde guardar el dato obtenido desde la memoria.

Por último, puede ingresar un comentario.

Escribir Memoria EEPROM I2C

Para guardar datos de memorias EEPROM de la familia 24 Cxxx, ingrese al árbol de herramientas como se muestra en la imagen de abajo:

Al ingresar en Memoria EEPROM, se presenta la pantalla en la que tendrá que seleccionar el tipo de acceso que desea y los parámetros del mismo, ya sea escritura o lectura.

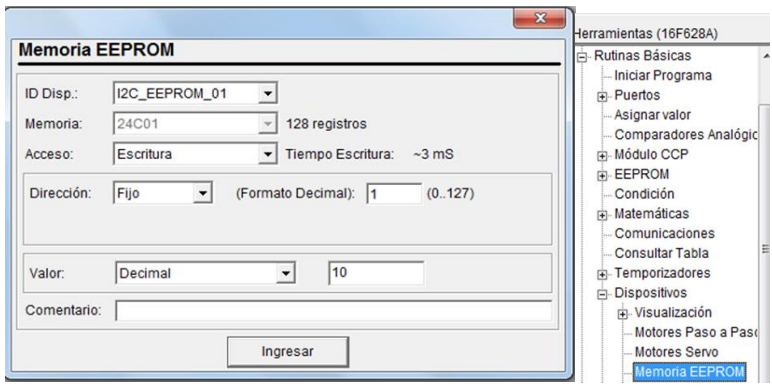


Fig. 6.58 Escribir memoria EEPROM

Seleccione el modelo de memoria que quiera utilizar.

Seleccione el tipo de acceso que desea tener a la memoria EEPROM (ya sea escritura o lectura) Luego debe elegir la dirección de memoria que desee escribir en la memoria EEPROM. Primero, escoja el tipo de dirección, la cual, puede ser "Fija" o "Variable".

Si escogió el tipo de dirección "Fija" ingrese la posición de memoria a escribir en formato decimal.

Si selecciona, el tipo de dirección "Variable" entonces debe seleccionar el registro que contiene el valor correspondiente a la posición de memoria que desea escribir.

Mediante este tipo de acceso (por dirección variable) el microcontrolador

accederá a la dirección correspondiente al valor que contiene el registro cada vez que se ejecute el bloque actual.

Una vez elegida y ubicada la dirección a escribir, Ud. Indicará el dato que crea conveniente grabar en la memoria. Puede ingresar un valor literal o seleccionar un registro, donde contiene el valor que desea grabar. Si selecciono la opción "un Literal", además tendrá que seleccionar el tipo de dato a grabar y luego debe ingresar el valor que desea grabar en la memoria.

Si selecciona la opción "Un Registro", debe seleccionar el registro que contiene el valor a grabar en la memoria.

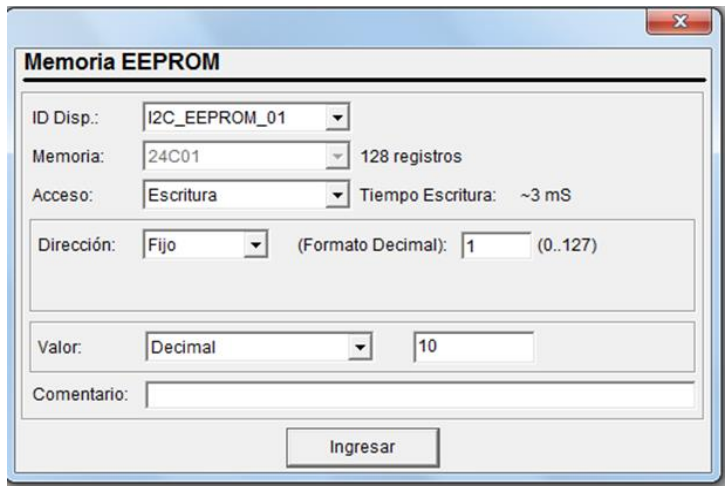


Fig. 6.59 Selección registro de acceso a escritura de memoria EEPROM

Por último, puede ingresar un comentario.

Memorias RAM

Declaración del Dispositivo

Ingrese a la sección Dispositivos, y seleccione el tipo de dispositivo "Memoria RAM I2C".

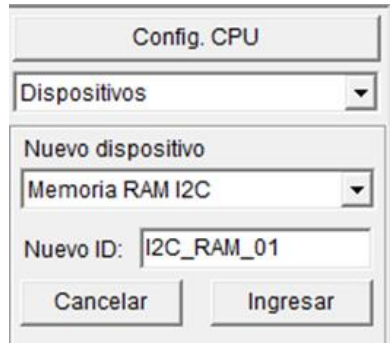


Fig. 6.60 Dispositivo memoria RAM

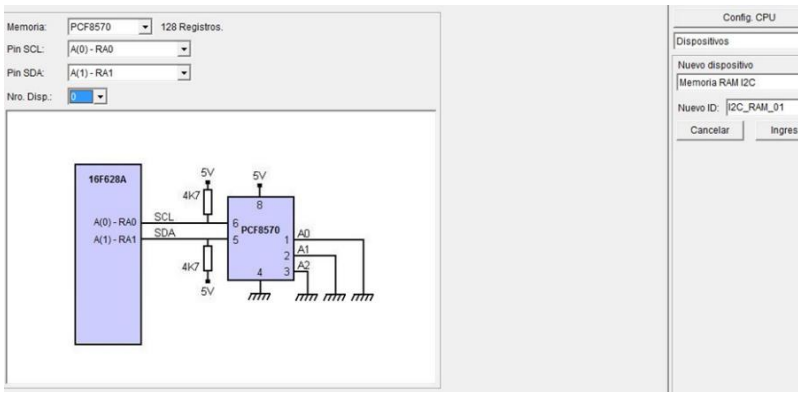


Fig. 6.61 Esquema conexión memoria RAM (PCF8570)

Seleccione el modelo de la memoria a utilizar y los pines correspondientes.
Presione el botón "Ingresar" para terminar de crear el dispositivo.
Recuerde que puede conectar varios dispositivos en paralelo formando una red, en la que cada dispositivo se le asigna un número. Este número de dispositivo se configura, conectando los pines correspondientes de cada dispositivo. La cantidad de dispositivos que puede incluir en una red depende del modelo de memoria que está empleando. Niple controla de manera automática la cantidad de dispositivos que puede conectar en función del modelo seleccionado.

Leer Memoria RAM I2C

Para leer datos de la memoria RAM, ingrese al árbol de herramientas como se muestra en la imagen de abajo:
Seleccione la memoria que desea usar. Al seleccionar el tipo de dispositivo a emplear, Niple le muestra la cantidad de registros de 8 bits que contiene el modelo seleccionado.
Seleccione el tipo de acceso a la memoria en este caso Lectura.
Actualmente, Niple soporta la memoria RAM PCF8570, pero en futuras versiones se irán agregando más modelos.

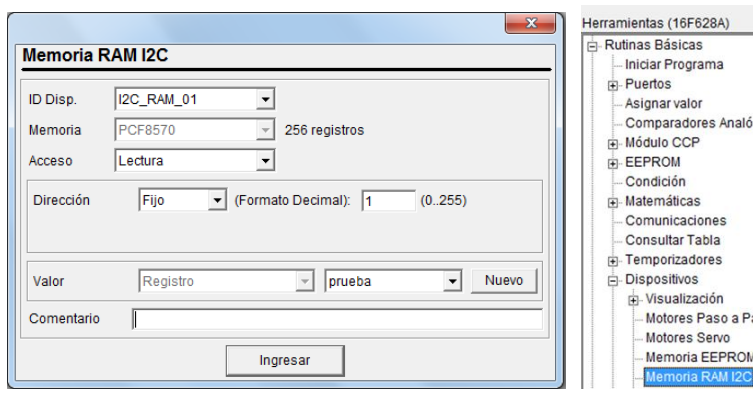


Fig. 6.62 Acceso a lectura memoria RAM I2C

Luego debe seleccionar la dirección de memoria que desea leer desde la memoria RAM. Primero debe selecciona el tipo de dirección, la cual puede ser "Fija" o "Variable".

Si selecciona el tipo de dirección "Fija" debe ingresar la posición de memoria a leer en formato decimal. Niple controla que la dirección ingresada no supere el límite máximo de cantidad de registros que contiene le modelo seleccionado. Mediante este tipo de acceso (por dirección fija) el microcontrolador accederá a la dirección ingresada cada vez que se ejecute el bloque actual.

Si selecciona el tipo de dirección "Variable" debe seleccionar el registro

que contiene el valor correspondiente a la posición de memoria que desea leer.

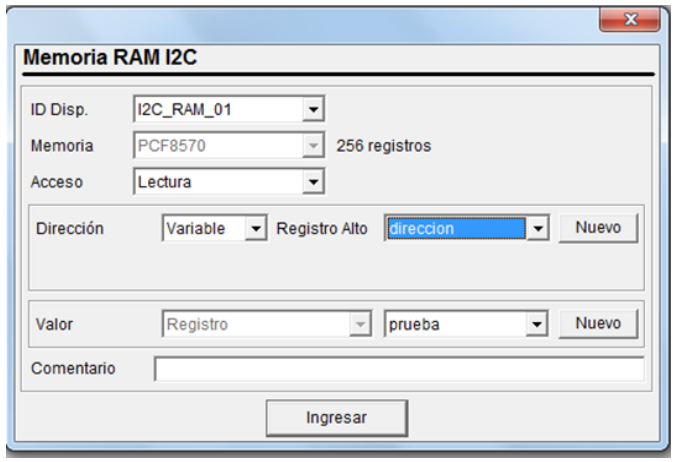


Fig. 6.63 Selección dirección variable de memora RAM I2C

En este caso, Niple no controla que la dirección ingresada no supere el límite máximo de cantidad de registros que con- tiene el modelo seleccionado dado que no se puede predeterminar que valor tendrá el registro seleccionado al ejecutarse el programa.

Mediante este tipo de acceso (por dirección variable) el microcontrolador accederá a la dirección correspondiente al valor que contiene el registro cada vez que se ejecute el bloque actual.

Una vez seleccionada la dirección a leer, debe seleccionar el registro donde guardar el dato obtenido desde la memoria.

Por último, puede ingresar un comentario.

Una vez ingresados los parámetros requeridos, puede visualizar el esquema de conexión.

Por último, presione el botón "Ingresar" y se insertará el bloque de

Lectura de Memoria RAM en el diagrama de flujo.

Escribir Memoria RAM I2C

Ingrese al árbol de herramientas en la sección "Dispositivos / Memoria RAM I2C".

Se presenta la pantalla "Memoria RAM por I2C". Seleccione el nombre del dispositivo que desea acceder.

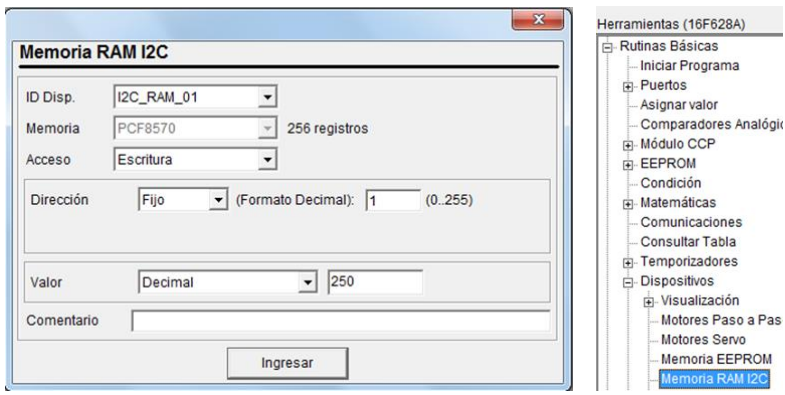


Fig. 6.64 Acceso de escritura a dirección fija de memoria RAM I2C

Recuerde que puede conectar varios dispositivos en paralelo formando una red, en la cual cada dispositivo es asignado por un número. El número del dispositivo, es configurado y conectado a los pines correspondientes de cada conector/dispositivo. La cantidad de dispositivos/conectores, que puede incluir en una red depende del modelo de memoria que se está utilizando. Niple, controla de manera automática la cantidad de terminales/dispositivos, que puede conectar en función del modelo seleccionado.

Luego debe seleccionar la dirección de memoria que desea escribir en la memoria RAM. En primer lugar, se tendrá que seleccionar el tipo de dirección, la que puede ser "Fija" o "Variable".

Si selecciona el tipo de dirección "Fija" deberá ingresar la posición de memoria a leer en formato decimal. Niple, controla que la dirección ingresada no supere el límite máximo, de cantidad de registros, que

contiene el modelo seleccionado. Mediante este tipo de acceso (por dirección fija) el microcontrolador, accederá a la dirección ingresada cada vez que se ejecute el bloque actual.

Si escoge, el tipo de dirección "Variable" tendrá que seleccionar el registro que contiene el valor correspondiente, a la posición de memoria que desea escribir. En este caso, Niple no controla que la dirección ingresada supere el límite máximo, de cantidad de registros que contiene el modelo seleccionado, dado que no se puede predeterminar que valor tendrá el registro seleccionado al ejecutarse el programa.

Mediante este tipo de acceso, (por dirección variable) el microcontrolador accederá a la dirección correspondiente al valor que contiene el registro cada vez que se ejecute el bloque actual.

Una vez seleccionada la dirección a escribir, deberá indicar el dato que desea grabar en la memoria. Puede ingresar un valor literal o seleccionar un registro, el cual contiene el valor que desea grabar.

Si selecciona la opción "Un Literal", debe seleccionar el tipo de dato a grabar y luego ingresar el valor que desee grabar en la memoria.

Si selecciona la opción "un Registro", debe seleccionar el registro que contiene el valor a grabar en la memoria.

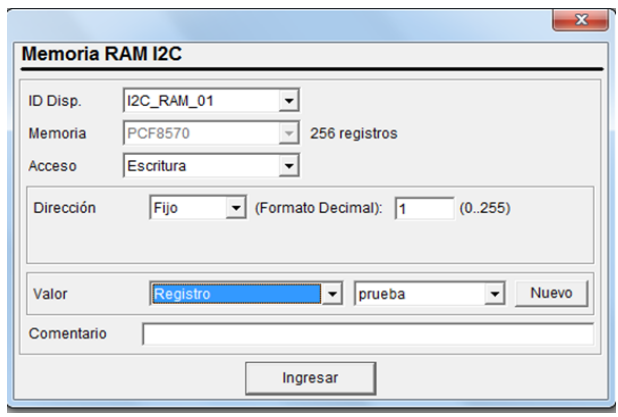


Fig. 6.65 Acceso a escritura de dirección fijo de memoria RAM

Por último, puede ingresar un comentario.

Reloj de Tiempo Real RTC PCF8583

Declaración del Dispositivo

Ingrese a la sección de Dispositivos y seleccione la opción "Reloj Tiempo Real" en la lista de nuevo dispositivo.

Fig. 6.66 Dispositivo reloj de tiempo real PCF8583

Se carga la pantalla de Declaración y Configuración del RTC.

Niple, incluye 2 modelos diferentes de RTC, son los que se incluyen en la lista de dispositivos en la parte superior-izquierda de la pantalla. Seleccione el RTC PCF8583.

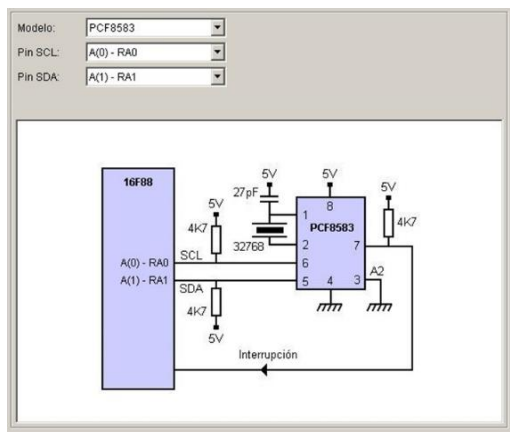


Fig. 6.67 Esquema de conexión RTC PCF8586

El PCF8583 funciona por comunicación I2C. Seleccione los pines correspondientes a las señales SDA y SCL en las listas correspondientes.

Por último, presione el botón "Ingresar" que se encuentra en la parte derecha de la pantalla.

Implementando el PCF8583 en el Diagrama de Flujo.

Ingrese a la sección de "Rutinas Básicas/Dispositivos/Reloj Tiempo Real" en el árbol de herramientas. Al cargarse la pantalla, Seleccione el dispositivo por su nombre (ID).

Las acciones que puede realizar sobre un PCF8583 son:

- Inicialización
- Escritura
- Lectura

Inicialización

El PCF8583 debe ser inicializado, la inicialización asigna un valor de inicio de fecha y hora, poniendo en marcha el reloj/calendario. Por defecto Niple, asigna la fecha y hora actual del proyecto.

Una vez configurado el Reloj de Tiempo Real, se coloca en marcha, activando un bit interno del reloj.

Este bit mantiene el reloj en funcionamiento y Niple, programa la inicialización del reloj para ver este bit y determinar si el reloj ya está funcionando, no se asigna la hora al iniciar el sistema.

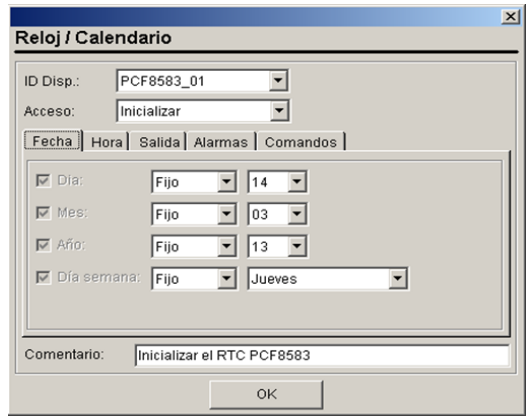


Fig. 6.68 Inicializar el RTC PCF8583

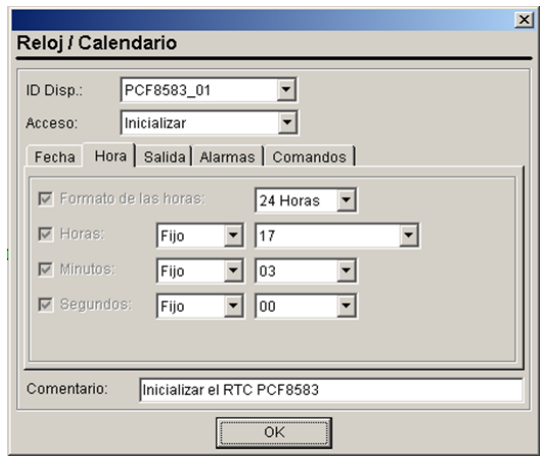


Fig. 6.69 Inicialización Hora del RTC PCF8583

Escritura

Puede escribir datos al Reloj de Tiempo Real y asignar los valores de Fecha y Hora del reloj.

Para configurar la Fecha actual, ingrese a la "Ficha" con el nombre "Fecha".

Seleccione los parámetros que desea configurar (Día, Mes, Año, Día de la Semana) marcado las opciones correspondientes. Cada uno de estos datos puede ser de tipo "Fijo" o "Variable".

Los valores a ingresar pueden ser los siguientes según el tipo de dato solicitado:

- **Día**:
  - Fijo: entre 01 y 31 inclusive.
  - Variable: un registro que contiene el valor del día.
- **Mes:**
  - Fijo: entre 01 y 12 inclusive.
  - Variable: un registro que contiene el valor del mes.
- **Año:**
  - Fijo: entre 00 y 03 inclusive.
  - Variable: un registro que contiene el valor del año.
- **Día Semana**:
  - Fijo: entre domingo y sábado.
  - Variable: un registro que contiene el valor del día de la semana.

La siguiente imagen muestra una configuración de fecha "Fija":

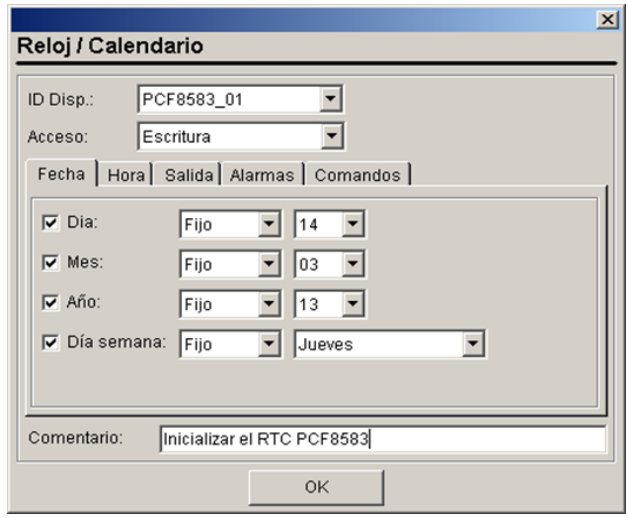


Fig. 6.70 Configuración Fecha Fija en RTC PCF8583

La siguiente imagen indica una configuración de fecha "Variable":

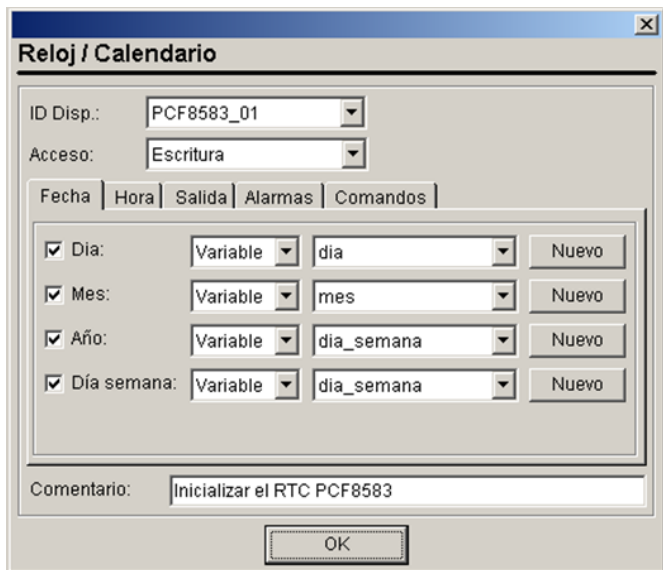


Fig. 6.71 Configuración Fecha Variable en RTC PCF8583

Para configurar la Hora actual, ingrese a la "Ficha" con el nombre "Hora".

Seleccione los parámetros que desea configurar (Formato de Horas, Hora, Minutos, Segundos) marcado las opciones correspondientes. Cada uno de estos datos puede ser de tipo "Fijo" o "Variable".

Los valores a ingresar pueden ser los siguientes, según el tipo de dato solicitado:

- **Formato de Hora:**

  — Fijo: puede seleccionar el formato de 12 o 24 horas.

- **Horas**

  — Fijo: depende del formato de hora seleccionado. Para el formato de 12 horas, puede seleccionar entre las 00 AM..11 AM y 00 PM..11 PM, Y para el formato de 24 horas puede seleccionar entre 0 y 23 inclusive.

— Variable: un registro que contiene el valor de las horas.

- **Minutos**

— Fijo: entre 00 y 59 inclusive.

— Variable: un registro que contiene el valor de los minutos.

- **Segundos**

— Fijo: entre 00 y 59 inclusive.

— Variable: un registro que contiene el valor de los segundos.

La siguiente imagen muestra una configuración de Hora "Fija":

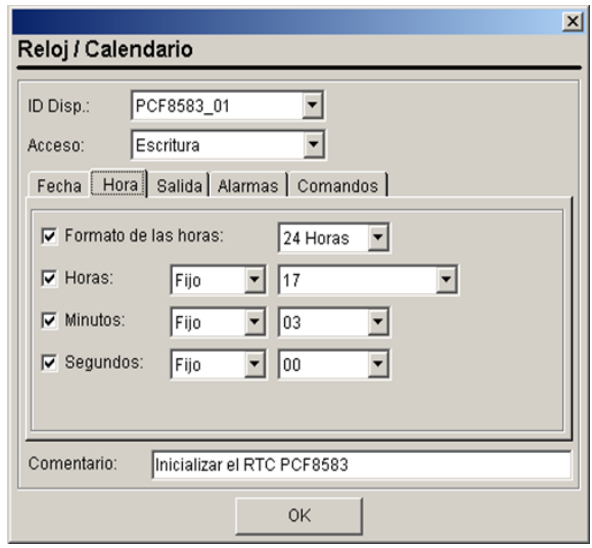


Fig. 6.72 Configuración Hora Fija en RTC PCF8583

La siguiente imagen demuestra una configuración de Hora variable:

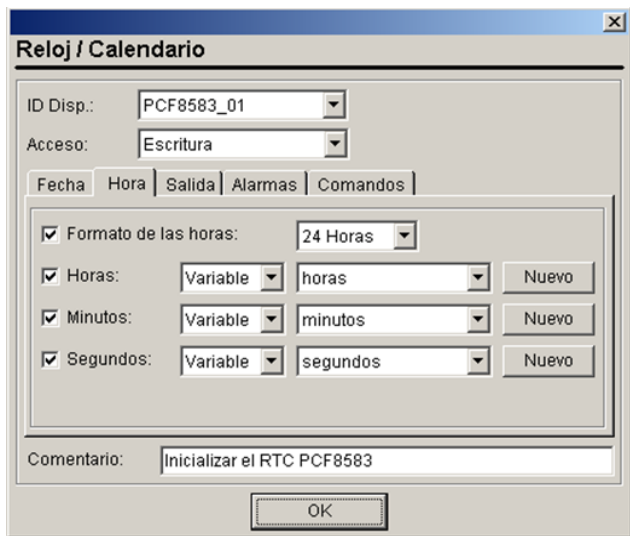


Fig. 6.73 Configuración Hora Variable en RTC PCF8583

Configuración de Alarmas

El PCF8583 incluye un "Pin de Alarma" que el que se ingresa en estado "Bajo" (0 V) cada vez que se produce una alarma. Mientras no se produce una alarma, el pin queda en estado alto (5 V).

Al originarse una alarma en el PCF8583, internamente se activa una bandera (Bit) de alarma y el pin de alarma se pone en estado bajo y se mantiene así mientras la bandera de señal, que permanece activada.

La bandera de señal/alarma se limpiará por software.

El PCF8583 incluye alarmas por Fecha/Hora o por Temporización.

La configuración de alarma por Fecha/Hora le permite generar distintos tipos de alarmas:

- **Alarma Diaria**: esta alarma se activa cada vez que coinciden la hora (horas, minutos y segundos) configurada como alarma con la hora actual del reloj (horas, minutos y segundos actuales).

- **Alarma semanal**: la alarma semanal se activa cuando coincide el día de la semana con él alguno de los 2 días de semana que puede configurar como alarmas.

- **Alarma Fechada**: se activa cada vez que coinciden los datos de Fecha/Hora configurados como alarma con los valores de Fecha/Hora actuales.

Cada una de estas alarmas puede ser configurada con valores Fijos o Variables.

Los valores a ingresar pueden ser los siguientes según el tipo de dato solicitado:

Alarma Diaria:

- **Horas**

- Fijo: depende del formato de hora seleccionado. Para el formato de 12 horas, puede seleccionar entre las 00 AM..11 AM y 00 PM..11PM, Y para el formato de 24 horas puede seleccionar entre 0 y 23 inclusive.
- Variable: un registro que contiene el valor de las horas.

- **Minutos**
  - Fijo: entre 00 y 59 inclusive.
  - Variable: un registro que contiene el valor de los minutos.

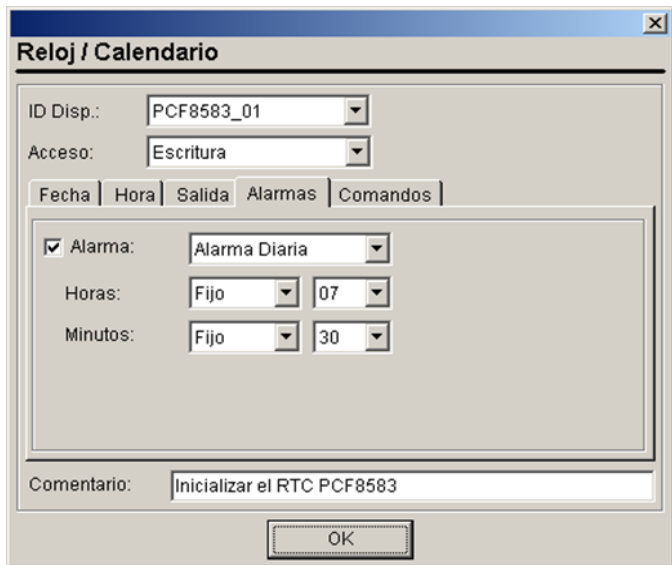


Fig. 6.74 Configuración Alarma Hora Fija en RTC PCF8583

Alarma por Fecha

- Día
  - Fijo: entre 01 y 31 inclusive.

— Variable: un registro que contiene el valor del día seleccionado.

- Mes

  — Fijo: entre 01 y 12 inclusive.

  — Variable: un registro que contiene el valor del mes seleccionado.

- Horas

  — Fijo: depende del formato de hora seleccionado. Para el formato de 12 horas, puede seleccionar entre las 00 AM..11 AM y 00 PM..11PM, Y para el formato de 24 horas puede seleccionar entre 0 y 23 inclusive.

  — Variable: un registro que contiene el valor de las horas.

- Minutos

  — Fijo: entre 00 y 59 inclusive.

  — Variable: un registro que contiene el valor del día seleccionado. Deberá generar el código correspondiente al día. Los códigos correspondientes a los días son los siguientes:

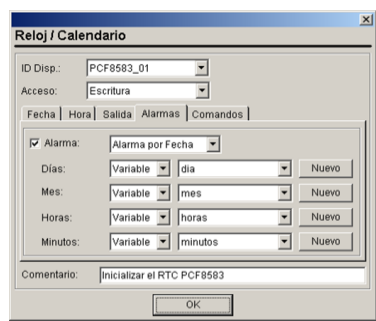


Fig. 6.75 Configuración de Alarma por fecha variable en RTC PCF8583

Lectura

Para leer la Fecha o la Hora actual seleccione los datos que desea consultar (Día, Mes, Año, Día de la Semana, Horas, Minutos, Segundos) y luego seleccione el registro donde desea almacenar cada uno de los datos obtenidos.

No es necesario, que obtenga todos los datos correspondientes a la hora, sino que, puede obtener únicamente los datos que necesita.

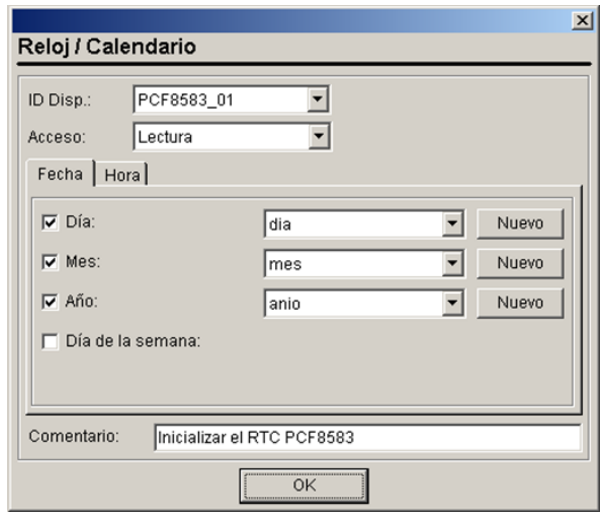


Fig. 6.76 Configuración acceso a lectura Fecha en RTC PCF8583

Utilizando reloj de tiempo real (DS1307)

Siguiendo con el tipo de dispositivos, como el explicado anteriormente, vamos a aprender a utilizar el módulo "reloj de tiempo real" para medir el tiempo. Es un dispositivo ideal para implementar proyectos de todo tipo que necesiten referencia horaria o de calendario.

Sabemos que Niple reúne una serie de dispositivos reloj/calendario de tiempo real, para crear aplicaciones que necesiten llevar un control del tiempo. Son excelentes para llevar a cabo desarrollos de consumo (relojes

tradicionales) o aplicaciones industriales para registrar datos (Dataloggers) en combinación con distintos tipos de sensores.

El DS1307 es un reloj de tiempo real (RTC) que utiliza el bus I2C para comunicarse con el microcontrolador. El reloj/calendario provee segundos, minutos, horas, día, día de la semana, mes y año. Ajusta de forma automática los meses de 31 días y años bisiestos.

Posee circuitería para detectar fallas en la alimentación y conmutar automáticamente al circuito de alimentación de respaldo (vía una celda de litio de 3V o similar), señal de salida programable (onda cuadrada de frecuencia configurable) y su consumo es menor a 500nA cuando se encuentra alimentado por la batería de respaldo.

Configuración de Pines

1. X1: conexión para cristal de cuarzo de 32.768kHz.
2. X2: conexión para cristal de cuarzo de 32.768kHz
3. VBAT: batería de respaldo. Si no se utiliza, este pin debe conectarse a GND.
4. GND: masa.
5. SDA: I2C datos.
6. SCL: I2C reloj.
7. SQW/OUT: salida de onda cuadrada.
8. VCC: alimentación.

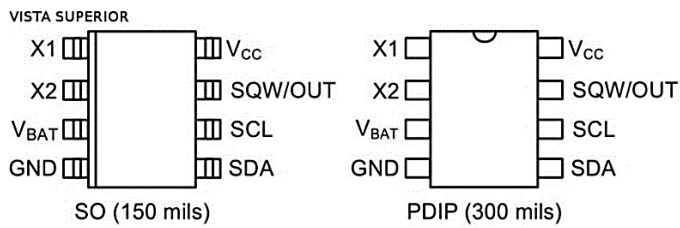


Fig. 6.77 Hoja de datos DS1307

Configuración del Módulo

Para poder utilizar el RTC es necesario declararlo. Para esto seguimos la ruta:

Declaración de Dispositivos

Nuevo Dispositivo ->Reloj Tiempo Real

Los parámetros a configurar son:

Modelo: Niple incorpora varios modelos de RTC. Disponemos de DS1302, DS1307 y PCF8583.

En nuestro ejemplo vamos a emplear el DS1307 PIN SCL: definir pin de reloj I2C PIN SDA: definir pin de datos I2C

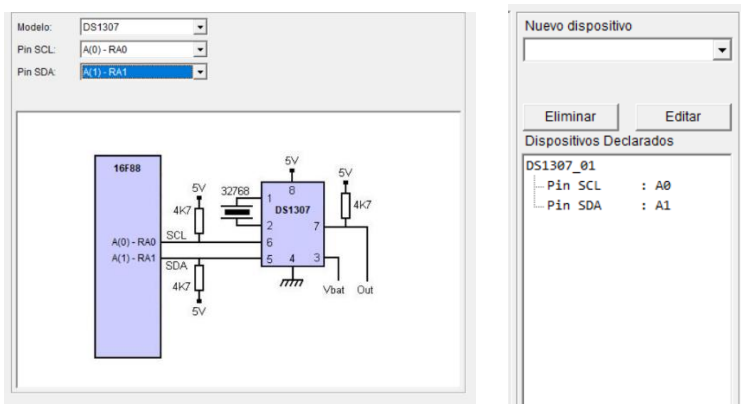


Fig. 6.78 Esquema de RTC DS1307

Inicialización

El primer paso es insertar el bloque de inicialización del dispositivo. Este bloque genera rutinas internas necesarias para el correcto funcionamiento del DS1307.

Por defecto configura el RTC con los datos actuales del sistema operativo (fecha y hora).

Fig. 6.79 Inicialización del RTC DS1307

Escritura

Mediante la opción "Escritura" accedemos a varios aspectos:

- FECHA: configuramos día, mes, año y día semana, tanto de forma fija (ingresando el valor deseado) como variable mediante el uso de registros.
- HORA: configuramos formato de las horas (fijo en 24HS), horas, minutos y segundos, tanto fijo como variable mediante el uso de registros.
- SALIDA: activar pulso de salida (frecuencias admitidas 1Hz 4096kHz, 8192kHz, 32768kHz).

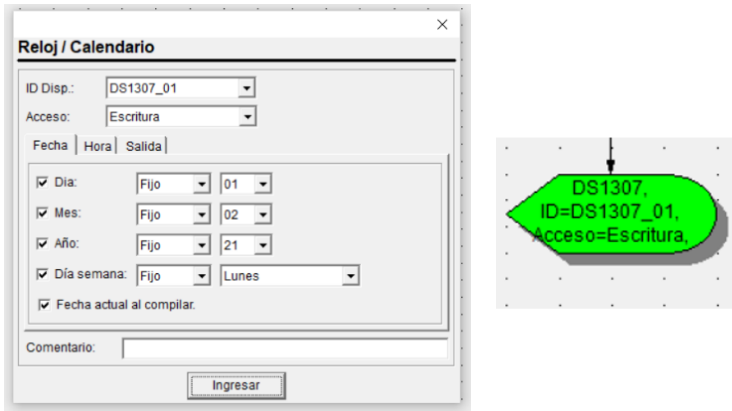


Fig. 6.80 Acceso a escritura de datos del RTC DS1307

Lectura

Mediante la opción "lectura" del bloque RTC podemos leer datos del dispositivo y almacenarlos en variables:

- FECHA: día, mes, año y día semana.
- HORA: horas, minutos y segundos.

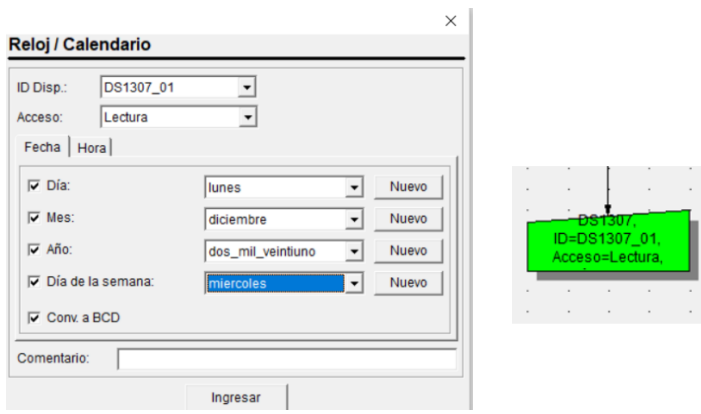


Fig. 6.81 Acceso de lectura de datos del RTC DS1307

Registro Desplazamiento Ingresar valor y rotar

Esta función rota el valor de un BIT en un registro, que puede contener hasta 64 bits de tamaño, el que es tratado, como si fuese un registro común. Cada vez que se produce la condición de reloj, desplaza un 1(uno) o un 0 (cero), dependiendo de la condición de entrada, se cumplirá o no, en el registro.

El usuario deberá declarar BIT en las posiciones que desee, y luego tendrá que programar lo que sucederá cuando estos tengan valor 0(cero) o 1(uno). En el ejemplo de la figura, se utiliza el registro de desplazamiento para el control de una máquina envasadora de estuches, que contiene 64 estuches en todo su largo, y la posición 61 corresponde al pateador de estuches fuera de peso. Se podría decir que cada vez que esté activada la expulsión de un producto fuera de peso y este BIT sea uno, el sistema de expulsión funcionará.

1) En Tamaño del registro, escriba el número de bits a utilizar, con un máximo de 64.

2) En Ingrese los datos de la posición escriba el nombre al BIT que desee y complete el cuadro correspondiente a Función.

3) Repita esta operación hasta declarar la cantidad de bits que desee.

4) Seleccione la señal de Reloj, que producirá la rotación de todos sus bits.

5) Elija si el registro se va a rotar cuando el BIT reloj sea 1(uno) o 0(cero).

6) Seleccione que condición se tiene que cumplir para que sea introducido un 1(uno) al registro.

7) Haga clic en Ingresar.

Las casillas de verificación "Y" o "O" corresponden a las operaciones lógicas AND y OR inclusiva correspondientemente.

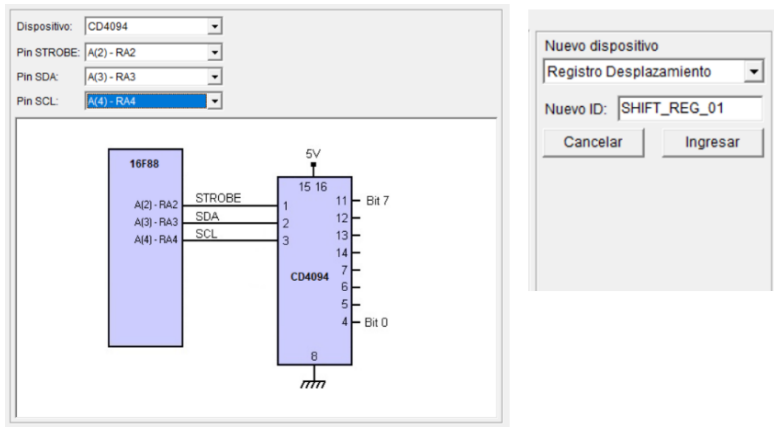


Fig. 6.82 Esquema registro de desplazamiento CD4094

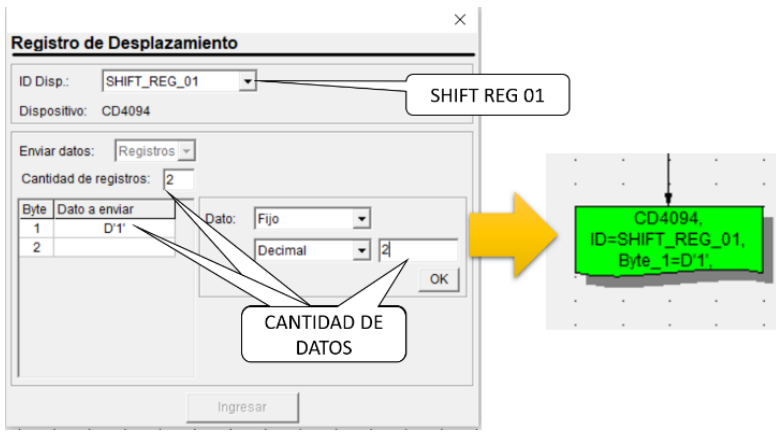


Fig. 6.83 Declaración cantidad de registros en CD4094

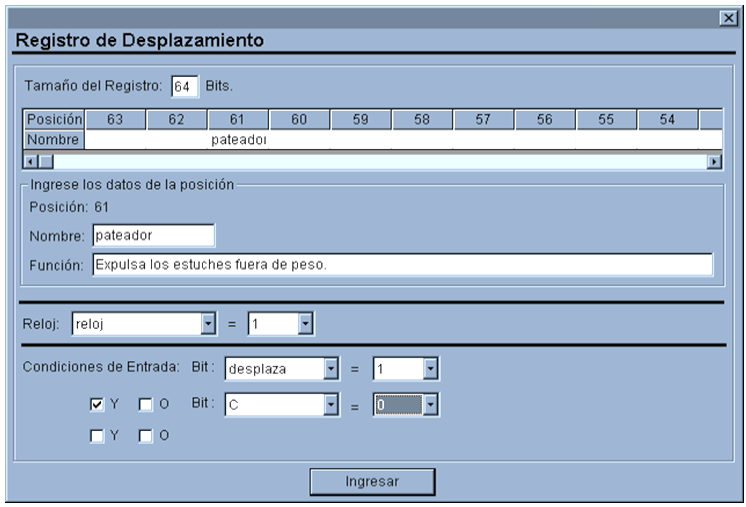


Fig. 6.84 Configuración tamaño del registro de desplazamiento

IMPORTANTE: Procedimiento para programación de los PIC 12F6xx.

Los micros de la familia de PIC 12F6xx requieren un procedimiento especial para ser programados. Estos micros requieren de un valor de calibración ubicado en final de la memoria de programa que permite garantizar la correcta calibración del oscilador interno.

Este procedimiento es necesario realizarlo para todos los tipos de programadores de PIC.

Por ello:

Es necesario recuperar este valor antes de utilizar por utilizar por primera el micro.

A continuación, vamos a utilizar el PIC12F675 y el software de programación IcProg o también, Pickit2 (utilizado por mí). Seleccione el modelo correcto de PIC. En este caso el 12F675.

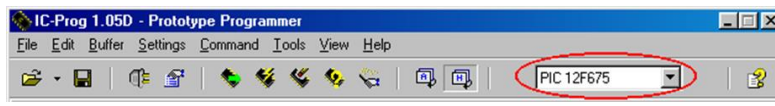


Fig. 6.85 Configuración IC-Prog detección microcontrolador

El valor de calibración puede variar de un chip a otro. Por esto, debe comenzar realizando una lectura de la memoria flash del micro. Para ello, presione el botón para realizar el proceso de lectura

Fig. 6.86 Lectura del microcontrolador

Al finalizar la lectura del micro, se obtiene el contenido de la memoria flash (memoria de programa). Dado que aún el micro no ha sido usado, todas las posiciones de memoria deben contener el valor 3FFF a excepción de la última dirección (0x03FF).

Esta última dirección de memoria flash, contiene el valor de calibración

del microcontrolador (en formato hexadecimal).

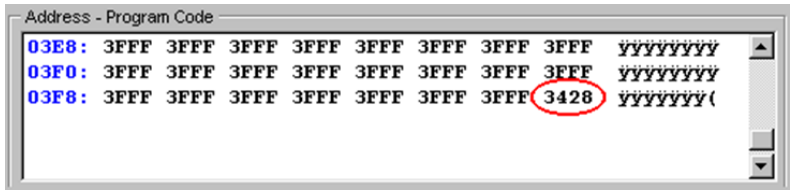


Fig. 6.87 Visualización ultima dirección de memoria flash (3428)

En nuestro ejemplo el valor de calibración del PIC actualmente utilizado es 0x3428.

Es aconsejable tomar nota del valor de calibración. Y mejor aún, puede grabar este valor pegando a un papel en la parte inferior del PIC. Este procedimiento es recomendable cuando trabaja con varios PIC. Recuerde que el código de calibración puede variar de un PIC a otro.

Cargue el archivo .Hex que desea programar en ICProg.

Al realizar esto, el valor de calibración del PIC16F675 fue sustituido por el valor hexadecimal por defecto $3FFF.

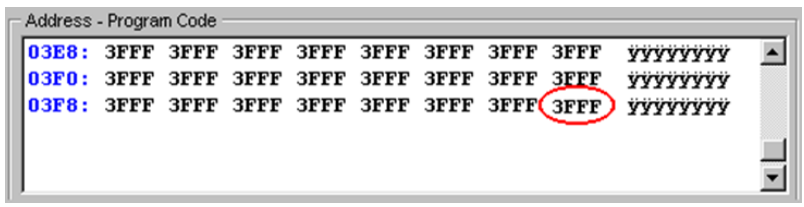


Fig. 6.88 Valor de calibración de sustitución al cargar el archivo (3FFF)

Haga "doble clic" sobre esta última posición de memoria con el fin de asignar el valor de calibración correspondiente.

Fig. 6.89 Editar ultima dirección de memoria (3FFF)

En la zona de valor hexadecimal 3FFF, ingrese el valor de calibración anteriormente obtenido. Para nuestro ejemplo, el valor corresponde a 3428.

Presione el botón "OK".

Ahora el programa está listo para grabar el PIC.

En la fase de programación, ICPROG comienza leyendo la memoria del PIC con el fin de comparar el contenido de la última dirección con el valor de calibración ingresado manualmente.

Si el código de calibración que contiene el PIC es el mismo, que el que se encuentra en la última posición de memoria del archivo a grabar, continúa la grabación y por último, muestra el mensaje que el PIC ha sido grabado correctamente y con esto queda finalizada la grabación del PIC.

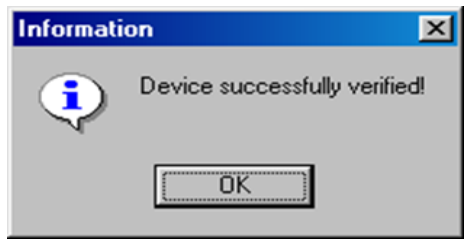


Fig. 6.90 Dispositivo verificado correctamente

Si el código de calibración que contiene el PIC no es el mismo que el que se encuentra en la última posición de memoria del programa a grabar, mostrará un mensaje indicado este problema y preguntará: — ¿Si desea sobrescribir el valor que actualmente se encuentra en el PIC, con el valor contenido en la última posición del archivo a grabar?

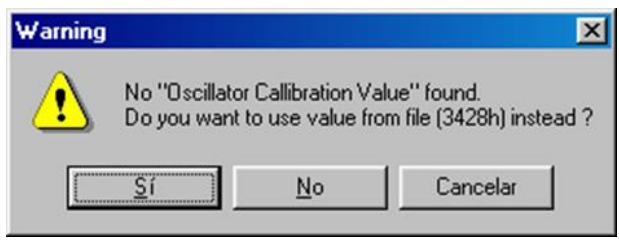


Fig. 6.91 Ventana emergente interrogando el uso de la última dirección de memoria (3428h)

Si responde "SI", se graba el valor de calibración contenido en la última posición del archivo.

Si responde "NO", se graba el programa, pero no se cambia el valor de configuración del PIC.

Si Ud. no está completamente seguro que el valor contenido en la última posición del archivo corresponde al valor de calibración del PIC, seleccione la opción "Cancelar" vuelva a realizar la lectura de la memoria del micro para obtener el valor de calibración.

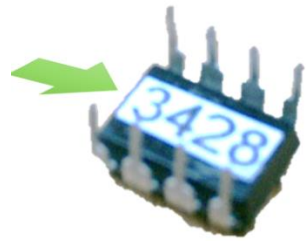
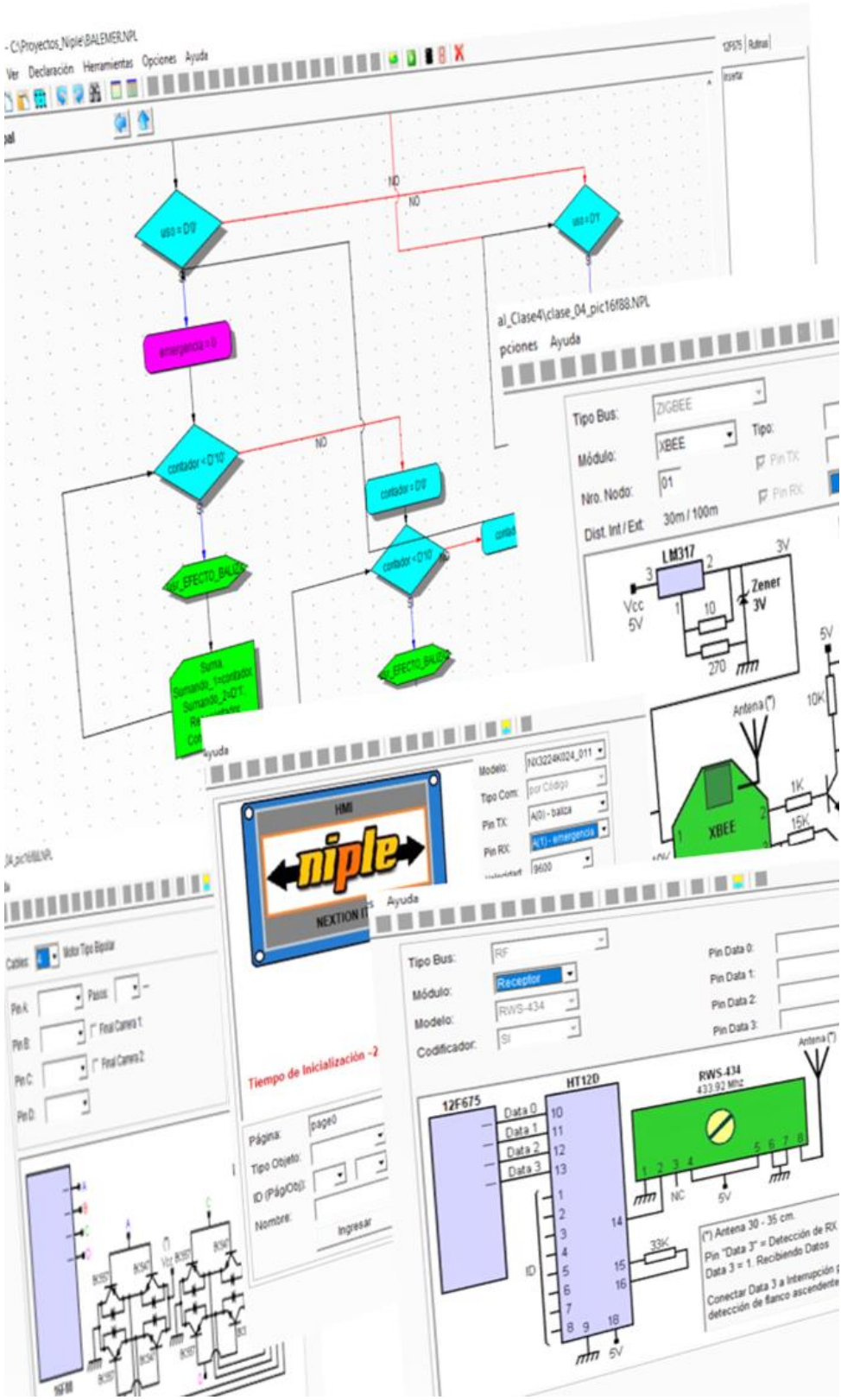


Fig. 6.92 Etiqueta de identificación ultima dirección de calibración de memoria

Para grabar el mismo programa, en distintos microcontroladores, deberá actualizar el valor de calibración de cada PIC en la última posición de memoria del archivo a grabar.


Declaración Herramientas Opciones Ayuda
NO
emergencia = 0
contador < D'10'
contador = D'0'
Suma
Sumando_1=contador
Sumando_2=D'1'
Tipo Bus:
ZIGBEE
Módulo:
XBEE
Nro. Nodo:
01
Dist. Int / Ext:
30m / 100m
LM317
Vcc
5V
Zener
3V
270
Antena (*)
10K
1K
15K
XBEE
HMI
niple
NEXTION IT
Modelo:
Tipo Com:
Pin TX:
Pin RX:
Tiempo de inicialización ~2
Página:
Tipo Objeto:
ID (Pág/Obj):
Nombre:
Ingresar
Tipo Bus:
RF
Módulo:
Receptor
Modelo:
RWS-434
Codificador:
SI
Pin Data 0:
Pin Data 1:
Pin Data 2:
Pin Data 3:
12F675
HT12D
Data 0
Data 1
Data 2
Data 3
RWS-434
433.92 Mhz
NC
5V
33K
ID
(*) Antena 30 - 35 cm.
Pin "Data 3" = Detección de RX
Data 3 = 1: Recibiendo Datos
Conectar Data 3 a Interrupción
detección de flanco ascendente

Capítulo 7
Proyecto. ¡Manos a la obra!

Habiendo visto en el capítulo anterior, podemos decir que hemos pasado por varios de los dispositivos, que se pueden incluir para un proyecto a realizar con microcontroladores PIC. Solamente, hemos ilustrado algunos de ellos, pero tenemos un gran recorrido para observar más adelante y saber todo sobre ellos, sus características técnicas, las prestaciones y funciones que incorpora Niple de ellos, su representación esquemática sugerida por el software, para un buen aprovechamiento de sus puertos.

En este caso, propongo un proyecto que es de fácil armado, donde explicaré paso a paso como montarlo. ¡Bien! ¡Manos a la obra!

Proyecto: Baliza electrónica y luz de emergencia

Este desarrollo es una tarea, poco compleja. Es un diseño sencillo, práctico y muy interesante al mismo tiempo, dado a los múltiples usos,

tanto en cuestiones domésticas como en lo laboral, o a modo de esparcimiento en medio del campo. Pretendo demostrarles de qué manera armar un dispositivo electrónico personalizado, según cada necesidad o criterio particular.

Este es un dispositivo que admite la dualidad de uso, por un lado, es una baliza que les explicaré su funcionamiento, y por otro, es luz de emergencia para asistencia de luminosidad en lugares recónditos o, en situaciones imprevistas. Por ejemplo: sin aviso se corta la luz eléctrica.

Elementos constitutivos

Una baliza debe generar un pulso de luz repetitivo, lo más intenso posible, consumiendo la menor cantidad de energía de una fuente de alimentación totalmente independiente y portátil; como por ejemplo la batería recargable de litio 18650 3.7v 3000 mAh (color azul).

Fig. 7.1 Batería recargable litio 3.7v (3000 mAh)

Led (diodo emisor de luz)

Utilizaremos diodos LED (light emition diode o diodos de emisión luminosa) los cuales no solo tienen un mayor rendimiento, sino que además se presentan en una amplia gama de colores (rojos, verdes, amarillos, azules, etc.) y diferentes tamaños para que el lector pueda personalizar su proyecto.

Fig. 7.2 Leds de 5mm de varios colores

La tensión de trabajo de un LED común es del orden de 1,8 V a una corriente de 20 mA (dependiendo del color a emplear). La mejor manera de construir una fuente de corriente adecuada, y sabiendo que se puede usar varios leds ya sean en serie y en paralelo conforme la construcción más adecuada.

Tansistores (NPN y PNP)

Como ya sabemos, un transistor es un dispositivo activo con tres terminales, y estos tres terminales se conocen como *Emisor*, *Base* y *Colector*. Hay dos tipos principales de transistores: **transistores de unión bipolar** y **transistores de efecto de campo**.
Los transistores **NPN** y **PNP** son transistores de unión bipolar, y es un componente eléctrico y electrónico básico que se utiliza para construir muchos proyectos eléctricos y electrónicos, como el que estamos por diseñar.
Por otro lado, los transistores de unión bipolar los podemos encontrar tanto como partes de circuitos integrados como en componentes discretos. En los transistores PNP, los huecos son los portadores de carga mayoritarios, mientras que en los transistores NPN, los electrones son los portadores de carga mayoritarios.
En este caso, emplearemos un transistor **BC337** para la baliza y otro para la luz de emergencia.

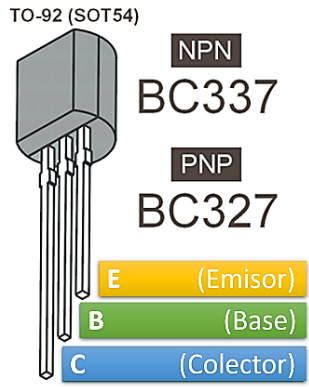


Fig.7.3 Transistor BC337 y BC327

Los transistores BC337 (NPN) son transistores bipolares para usos generales de mediana potencia, muy similares a los BC548 (NPN), pero con una capacidad de corriente de colector mucho mayor, hasta 800 mA, característica que les permite ser usados en muchas aplicaciones de control de media potencia, tales como drivers para pequeños motores, relés y también tiras de leds cortas.

En la siguiente ilustración se muestra la conexión a un circuito de control para tiras de led (cortas) que emplea un BC337. El valor de R depende de la lógica de control, sin embargo, se puede utilizar un valor de 3,3KΩ.

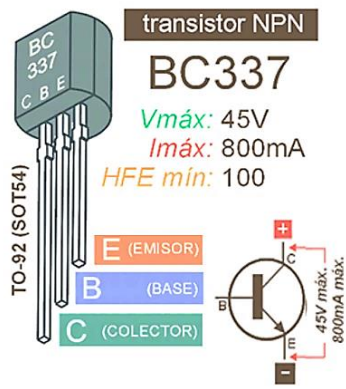


Fig.7.4 Datos técnicos Transistor BC337

Cálculos de resistencias para los Leds rojo y blanco:

Para comenzar con los cálculos, contamos con una tensión total de batería de 7,2 volts (2 baterías en serie), debemos tener en cuenta que difiere la tensión de trabajo para el led de color blanco, respecto del led de color rojo. Expresando este detalle, realizaremos los cálculos de las resistencias respectivas.

En el caso del Led rojo, tiene una tensión de trabajo de 1,8 volts, lo que nos permite, de esta forma, sumar dos leds en serie y colocar una resistencia limitadora de corriente de 82 ohm. En cambio, para los leds de color blanco existe una tensión de trabajo de 3,6 volts, posibilitando a un solo led por cada rama a conectar en el transistor, pudiendo tener una resistencia limitadora de corriente de 180 ohm. A continuación, se ilustran los cálculos realizados, conforme a la hoja de trabajo de cada led.

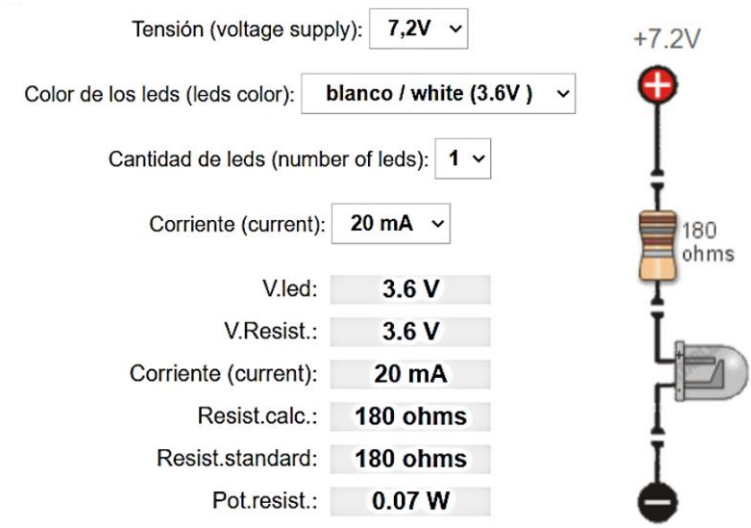


Fig. 7.5 Calculo resistencia para led blanco

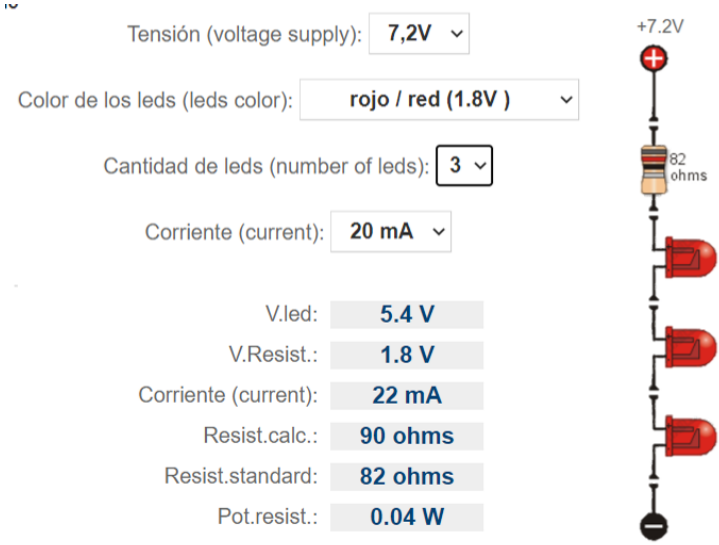


Fig. 7.6 Calculo resistencia para led rojo

Programando en Niple

Se crea un nuevo proyecto en Niple, escogiendo el microcontrolador PIC 12F675, luego seleccionar.

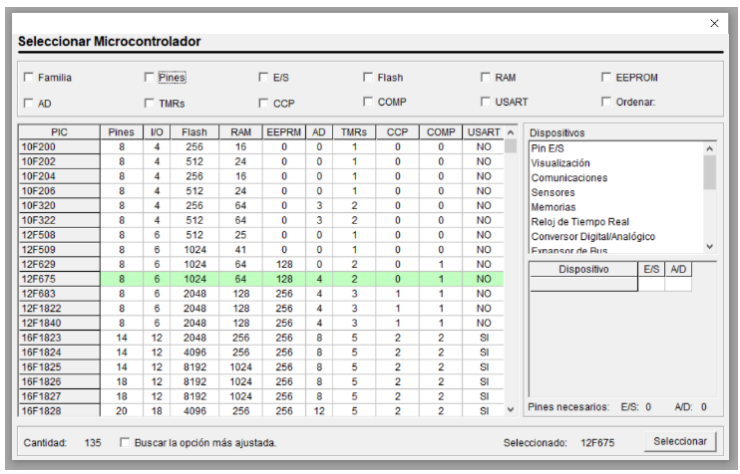


Fig. 7.6 Selección microcontrolador PIC 12F675

A posteriori, en la siguiente ventana, se asigna el nombre del proyecto y el del de archivo "baliza_emergencia. NPL", en el directorio asignado.

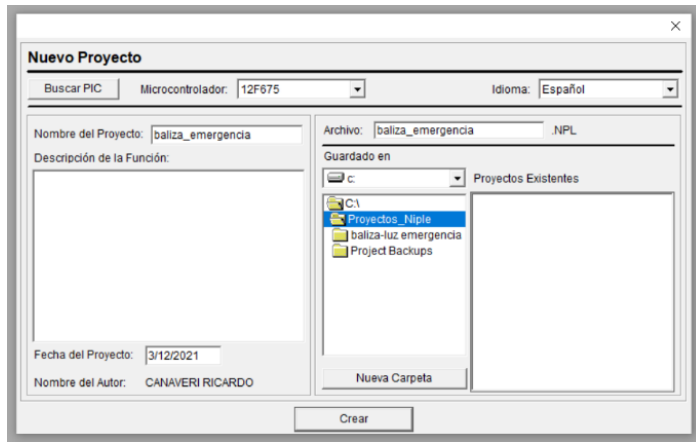


Fig. 7.7 Asignación de nombre del archivo y proyecto

Se sigue con la configuración del CPU, en este punto se usa el cristal interno de 4 Mhz, como OSC, el Pin de RESET lo establecemos como RESET.

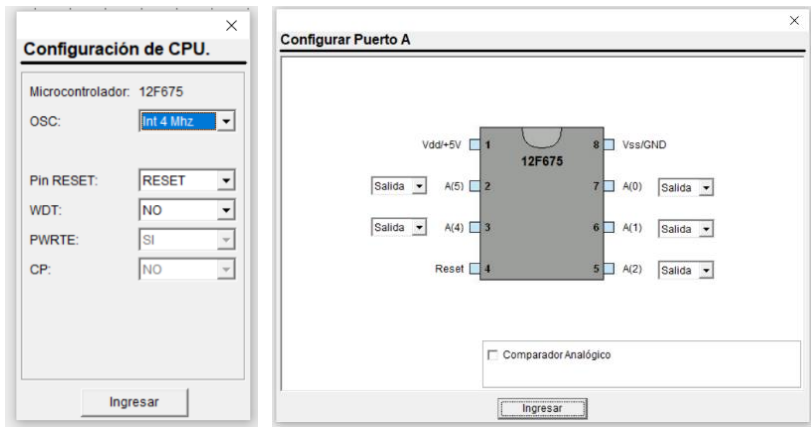


Fig. 7.8 Configuración CPU y pines del Puerto A

En esta instancia, se verifican los nombres establecidos en el

esquemático, para reemplazar los originales de los bits GP0 y GP1, por los siguientes: baliza y emergencia.

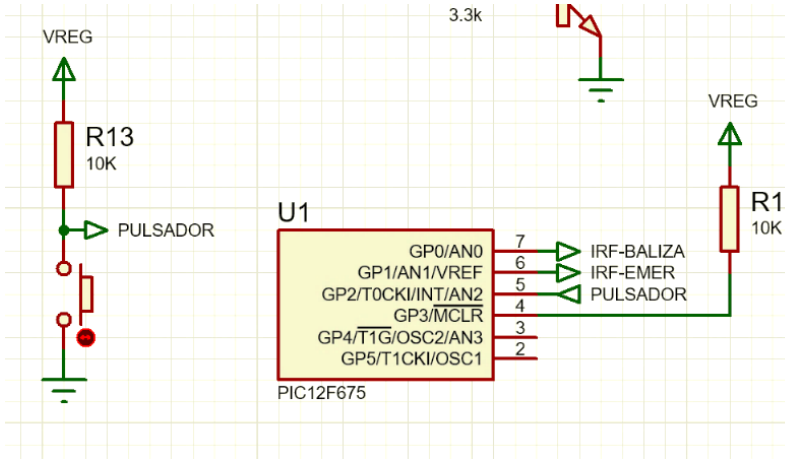


Fig. 7.9 Esquemático PIC 12F675 en ISIS

Para poder realizar esta actividad, se deberá ir al menú, y cambiar el modo de usuario, para pasar de usuario "inicial" a usuario "experto", así se podrá modificar los nombres de los bits designados para baliza y para luz de emergencia del proyecto.

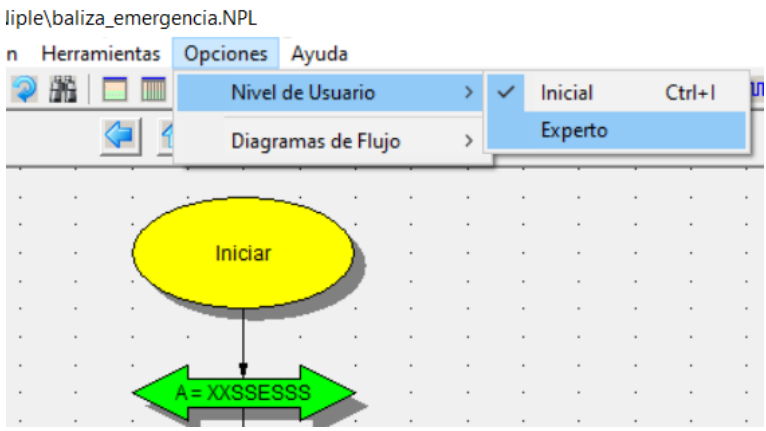


Fig. 7.10 cambio de modo de usuario

Se procede con el cambio de nombres de los bits, para ello hay que dirigirse a menú. Declaración de Bits, seleccionamos GP0, se da ENTER, entonces Niple preguntará si se desea cambiar el nombre al bit seleccionado.

Fig. 7.11 Cambio de nombre del bit GP0 por "baliza"

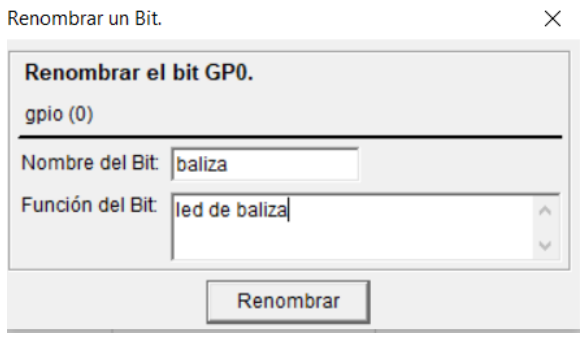


Fig. 7.12 Renombrar un Bit

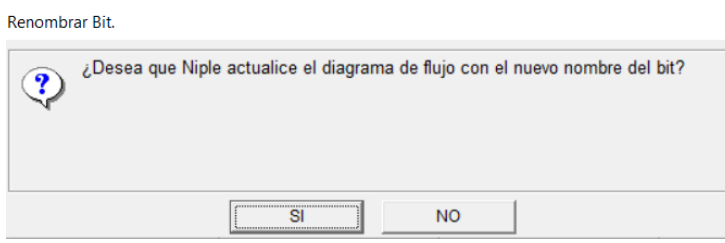


Fig. 7.13 Renombrar Bit.

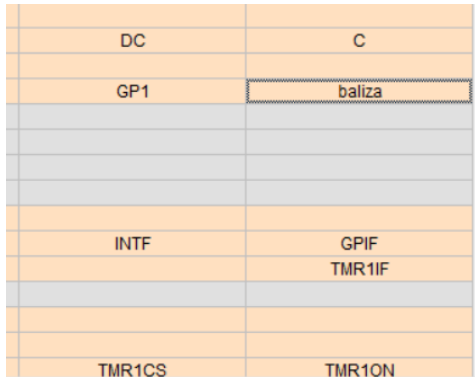


Fig. 7.14 Renovación bit GP0 por baliza

Se establece el nuevo nombre al segundo bit GP1 por led emergencia, de la misma forma que el anterior.

Fig. 7.15 Renovación bit GP1 por led_emergencia

A continuación, se puede apreciar en la memoria RAM de bits los nombres reasignados.

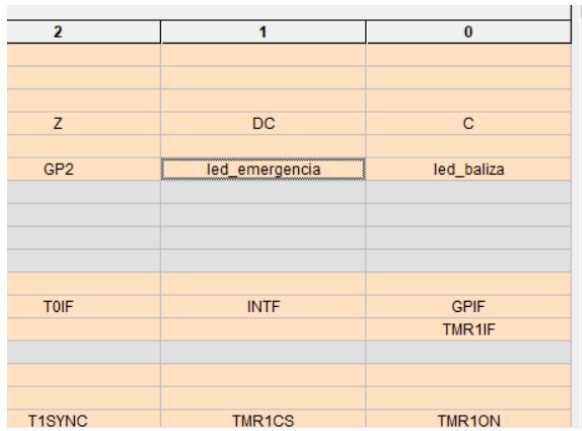


Fig. 7.16 Memoria RAM de bits

Luego, se procede a ir a la barra de herramientas, para asignar un valor a un bit, se buscan los bits renombrados y se asigna un valor decimal 0, para inicializarlos.

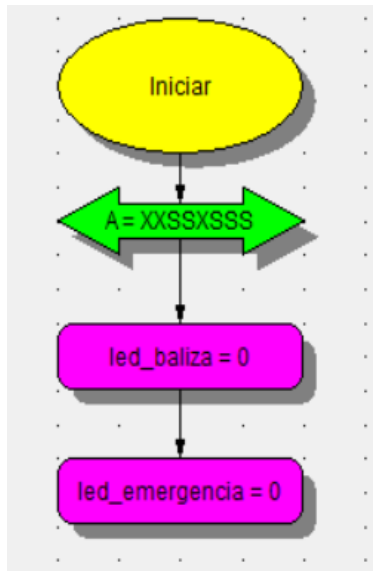


Fig. 7.17 Diagrama de Bloques

Configuramos una nueva interrupción, recordando que en el

esquemático optamos por poner un pulsador en el bit del puerto A GP2, por consiguiente, se tienen que inicialmente configurar la interrupción y luego activarla. Verificamos que GP2 esté configurado como bit de "Entrada", del modo contrario, Niple nos alertará con un mensaje emergente informándonos que el bit GP2 no está configurado como bit de entrada.

De esta manera hay que ir a Configuración de Puerto A, y realizar la modificación por "Entrada"

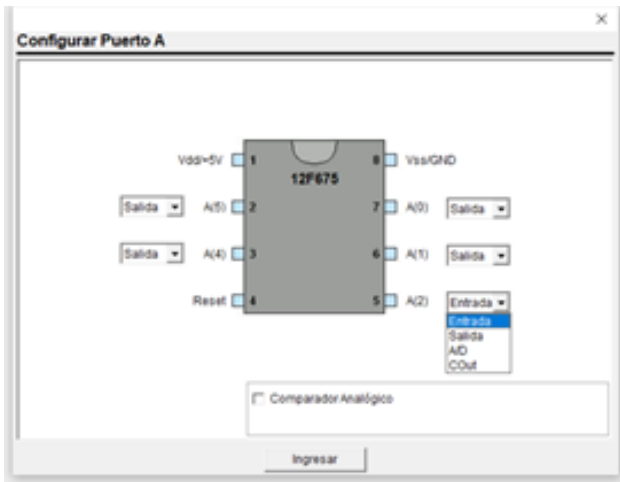


Fig. 7.18 Cambio de configuración GP2

Acá se realiza la habilitación general de las interrupciones "Activar", luego tildamos la casilla de Interrupción de detección de flanco en GP2 "Activar", dando a finalizar.

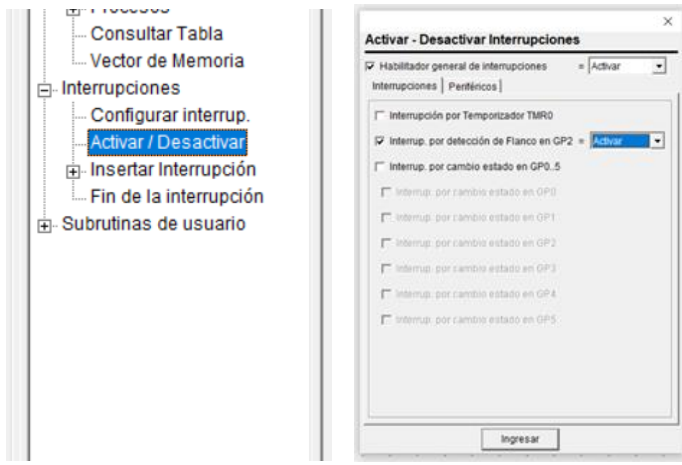


Fig. 7. 19 configuración Interrupción GP2

Una vez habilitadas las interrupciones generales, se busca en el árbol de herramientas de Interrupciones y se elige la interrupción "Por GP2", de esta forma se abrirá una ventana con el inicio de configuración de la interrupción GP2.

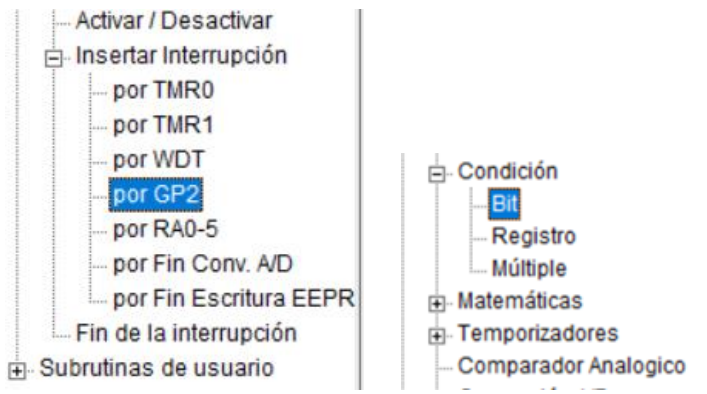


Fig. 7.20 Interrupción por GP2

Atención: es necesario tener presente que esta interrupción es para atender los cambios de estados del pulsador, que están conectados en el puerto GP2. Como se encuentra a +5v con una resistencia de 10 kΩ,

ósea que, está siempre en estado lógico de "1", ahora bien, si al presionar el pulsador, existe una tendencia o hacemos que cambie ese estado lógico por un estado a "0", esta condición nos permitirá establecer un condicional en la interrupción de GP2, consultando el estado del bit GP2 si es igual a cero, de esta manera existe una garantía que se está presionando el pulsador. Otra cosa más, hay que tener presente, cuando se presionan el botón en los contactos metálicos que forma parte del dispositivo que los pulsadores, se producen micro contactos, y estos generan una sucesiva cantidad de cambios continuos de estados lógicos, producto de esos micro contactos establecidos. Para evitar que esto suceda, se desarrolla una rutina anti rebote que van a evitar los rebotes indeseados, lo bueno de esto, es que Niple en las recientes versiones soluciona estos inconvenientes, agregando una opción para los pulsadores, de activar una rutina embebida, posibilitando elegir la opción de tildar el casillero que nos dice: hasta que se suelte el pulsador.

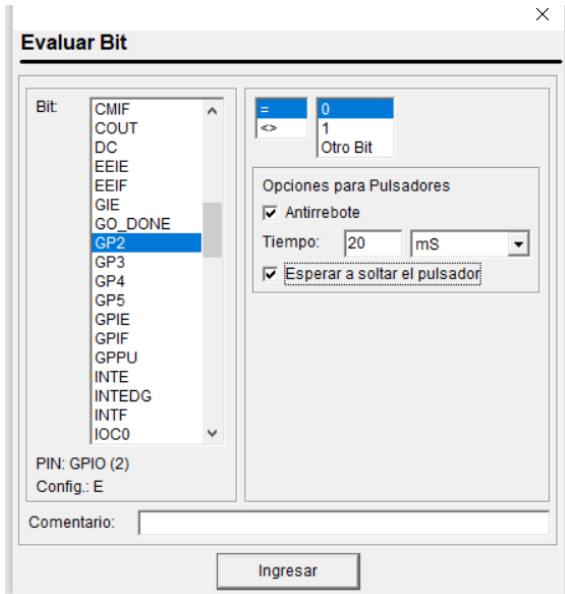


Fig. 7.21 Cuadro de dialogo de comparación de bit

En este aspecto se crearán dos subrutinas de usuario, una que es la

subrutina para el funcionamiento como baliza, y la otra, el funcionamiento como luz de emergencia. Entonces primero creamos la subrutina de usuario "Baliza"

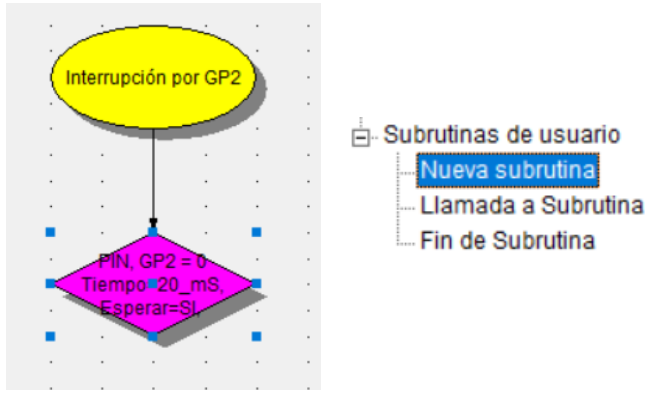


Fig. 7.22 Bloques de interrupción por GP2

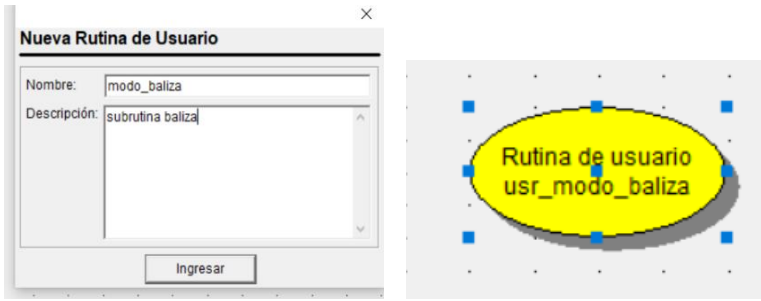


Fig. 7.23 Creación de subrutina modo baliza

Se puede utilizar una instrucción Assembler de control, para generar un paso que no haga nada, que realiza una sola operación, consumiendo un ciclo de programa, es muy utilizada para efectuar retroalimentaciones o loop, similares a los que se emplean con las ·Etiquetas. Esta instrucción se llama NOP.

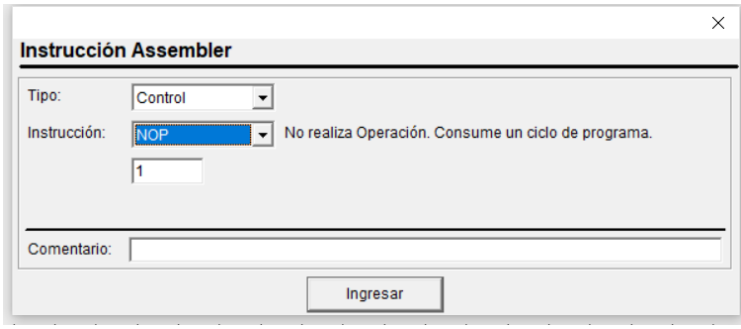


Fig. 7.24 Cuadro de dialogo de comparación de bit

Posteriormente, se debe crear un registro "Contador", para ser evaluado en cada condicional, que se desee generar un efecto distinto de frecuencia de encendido y apagado de los leds. Este registro es de 8 bits, valor "Decimal" inicializado en cero.

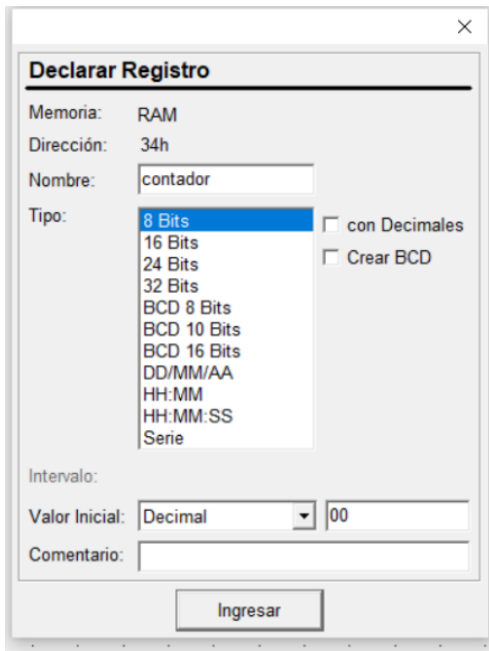


Fig. 7.25 Cuadro de dialogo de comparación de bit

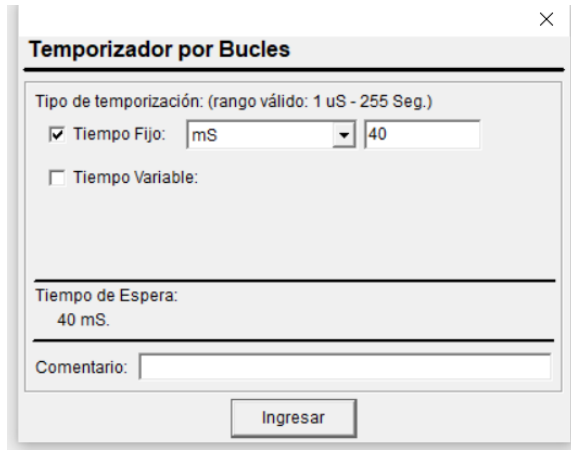


Fig. 7.26 Cuadro de dialogo de comparación de bit

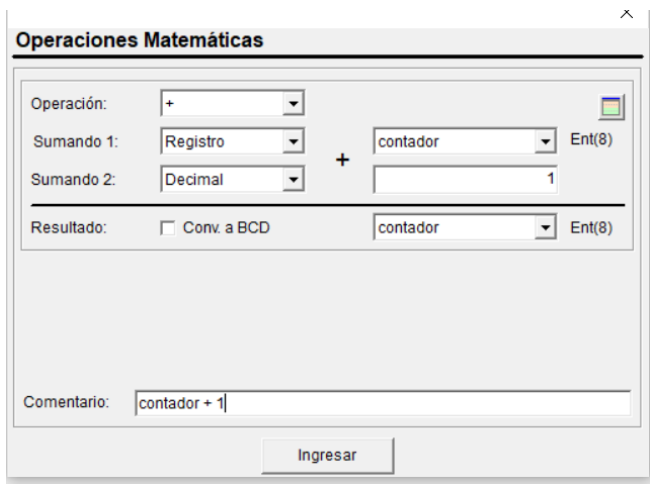


Fig. 7.27 Cuadro de Operaciones Matemáticas (Contador o sumador)

La lógica, empleada en el registro contador es con el valor inicial "0" en cuanto sea menor a 10, Ingresará y realizará el efecto baliza, luego se incrementará en un valor decimal. Cuando la condición no se cumpla, sale de la función baliza y pasa a la modalidad luz de emergencia.

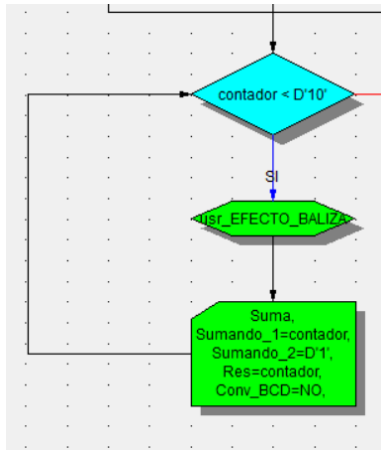


Fig. 7.28 Función de matemáticas para realizar suma o acumuladores

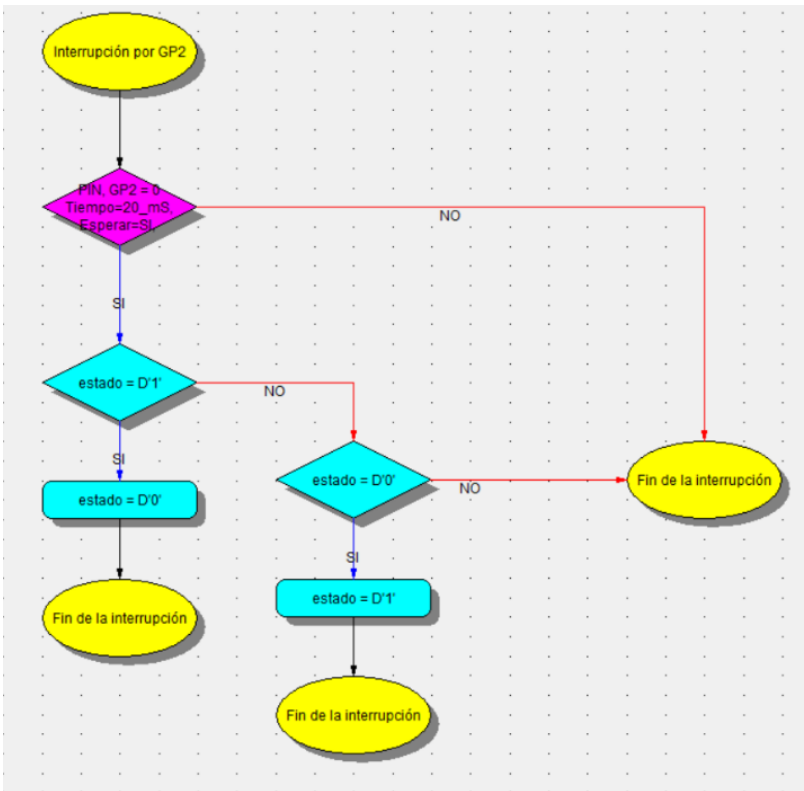


Fig. 7.29 Interrupción por Gp2 completa

Se crea la nueva rutina de usuario modo_luz_emergencia, este modo es sencillo porque sólo lo que tenemos que hacer es cambiar el estado lógico del bit del puerto A GP1.

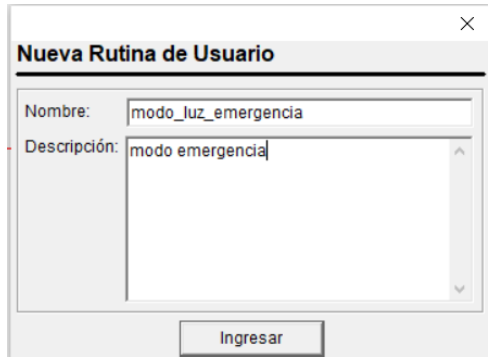


Fig. 7. 30 subrutina de usuario luz de emergencia

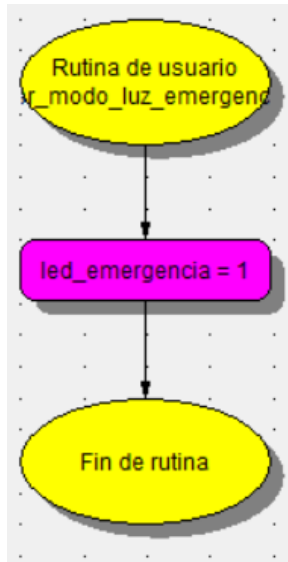


Fig. 7.31 Rutina de usuario modo_luz_emergencia

Guardamos el proyecto, y luego generar archivo ASM. Para verificar su no hay algún problema en la programación.

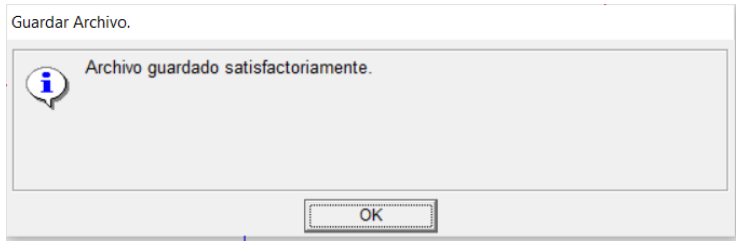


Fig. 7.32 Guardar archivo de Proyecto

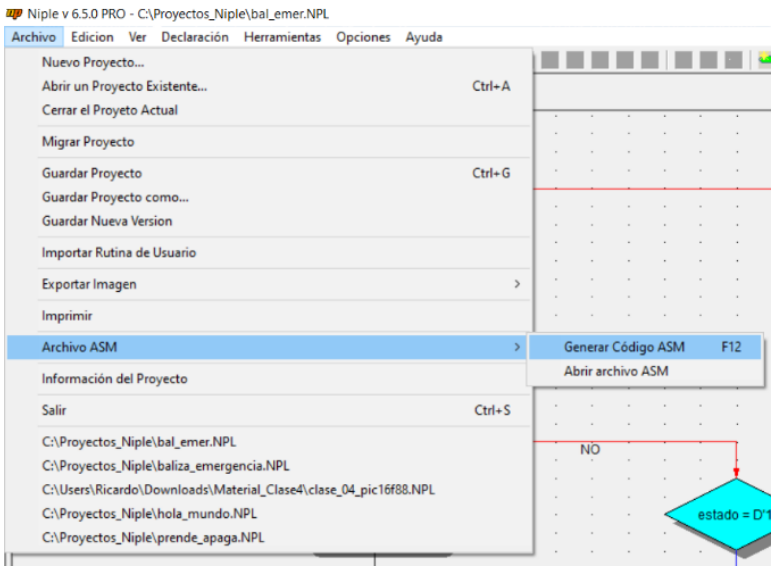


Fig. 7.32 Generar Código ASM

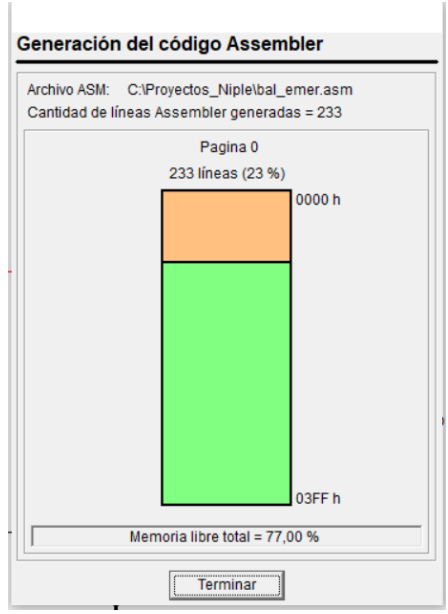


Fig. 7.33 Generación del código Asamblea (23%)

Luego de haberse generado el código Assembler, se abre el software ensamblador para generar el archivo. Hex del proyecto. Podemos invocar al mismo, desde Menú->Herramientas->Software Compilador, o mediante la tecla de acceso rápido "F7". Buscamos el archivo .ASM que deseamos compilar y hacemos clic en Assemble.

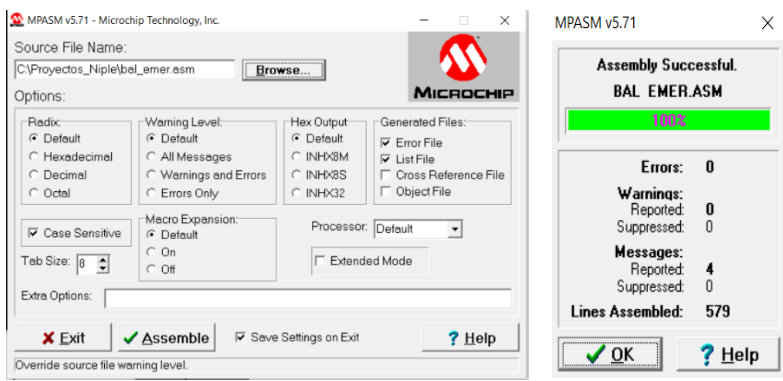


Fig. 7.34 MPASM v5.71 Asembler.

Si nos vamos a la barra de herramientas e ingresamos a la solapa de "Rutinas" se visualiza el árbol de entornos generados, en el que se incluyen:

- Programa principal.
- Interrupción por GP2
- usr_EFECTO_BALIZA (efecto lumínico N°1)
- usr_EFECTO_BALIZA2 (efecto lumínico N°2)
- usr_MODO_EMERGENCIA.

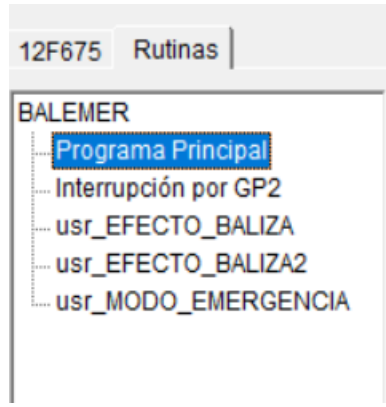


Fig. 7.35 Entornos generados en el proyecto

Explicaré como diseñé la "interrupción por GP2". En primer lugar, se ubica en el bloque de inicio de la interrupción "Interrupción por GP2" (de color amarillo), vinculado a este, se ubica el bloque comparar un bit, para este caso, el bit que se compara es GP2, donde se evalúa si GP2=0, para saber si el pulsador fue presionado, en caso contrario su valor será GP2=1.

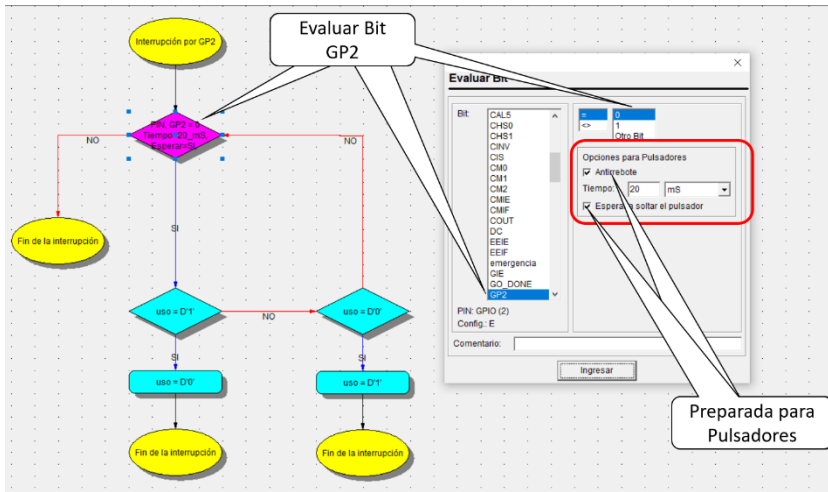


Fig. 7. 36 Detalle de la interrupción por Gp2, con la incorporación de la rutina anti rebote para pulsadores.

Una vez evaluado el bit GP2, en caso de ser igual a cero, se cumple la condición por "SI" y allí se hallara un segundo condicional que evalúa el registro "uso". Si Uso=1, el funcionamiento es en "modo baliza", caso contrario, si Uso=0; el funcionamiento es en modo luz de emergencia.
Lo interesante en esta lógica, es que al cumplir cualquiera de las condiciones, al registro se le modifica su valor decimal (si vale 1 se cambia por 0 y viceversa), de esta forma se garantiza en el código, que cada vez que se presione el pulsador, se establecerá un modo de funcionamiento (baliza o luz de emergencia).

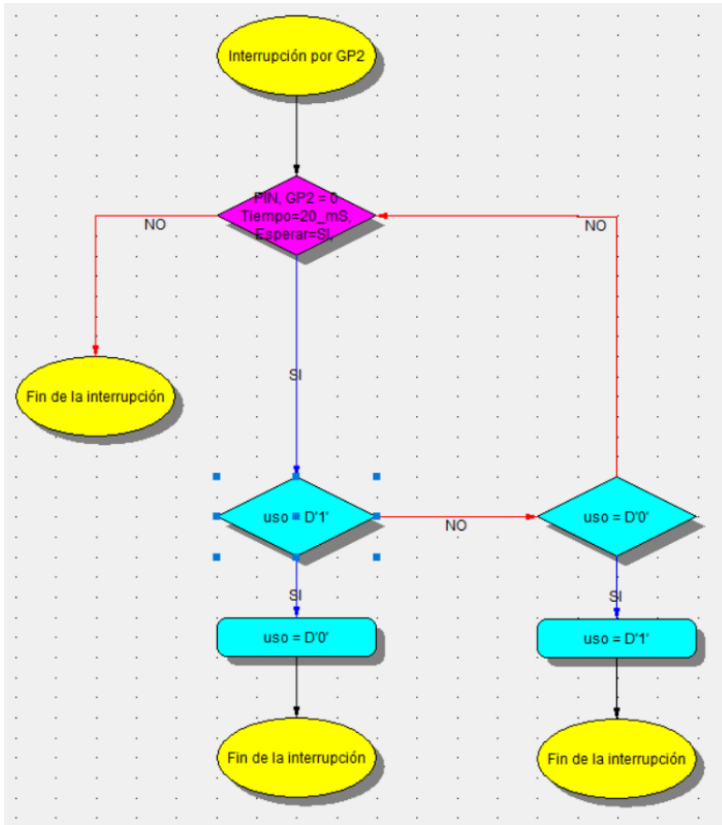


Fig. 7.37 Interrupción por GP2

En este caso mostraré como está diseñado el esquemático completo del proyecto, identificando la ubicación de cada parte del mismo. Donde se puede observar, que se dividió en partes o etapas de bloques:

- Zona de led de la baliza
- Zona de led de la luz de emergencia
- Zona de potencia (transistores BC337)
- Etapa de control (Microcontrolador PIC 12F675)
- Etapa de regulación 5 volt.
- Pulsador (etapa cambio de estados)

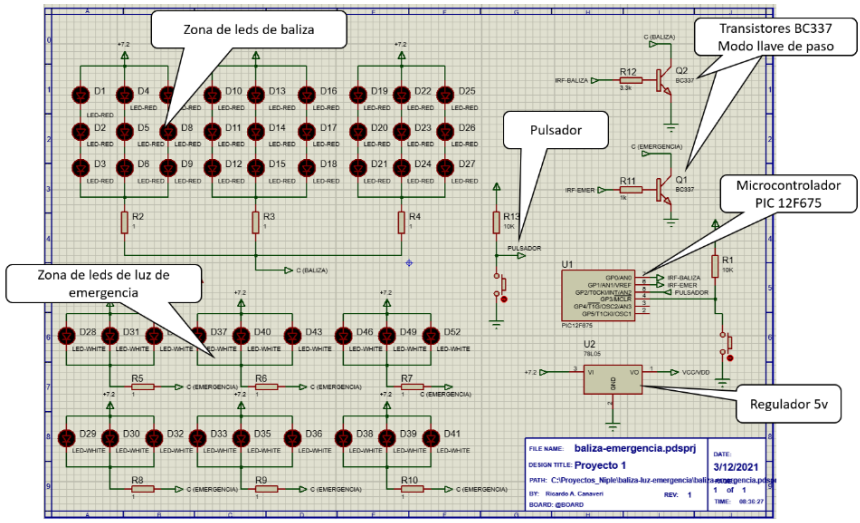


Fig. 7.38 Esquemático completo del proyecto

Ahora, dejaré los links de acceso del material, el proyecto en Niple, los esquemáticos de Proteus y los archivos compilados para realizar las simulaciones.

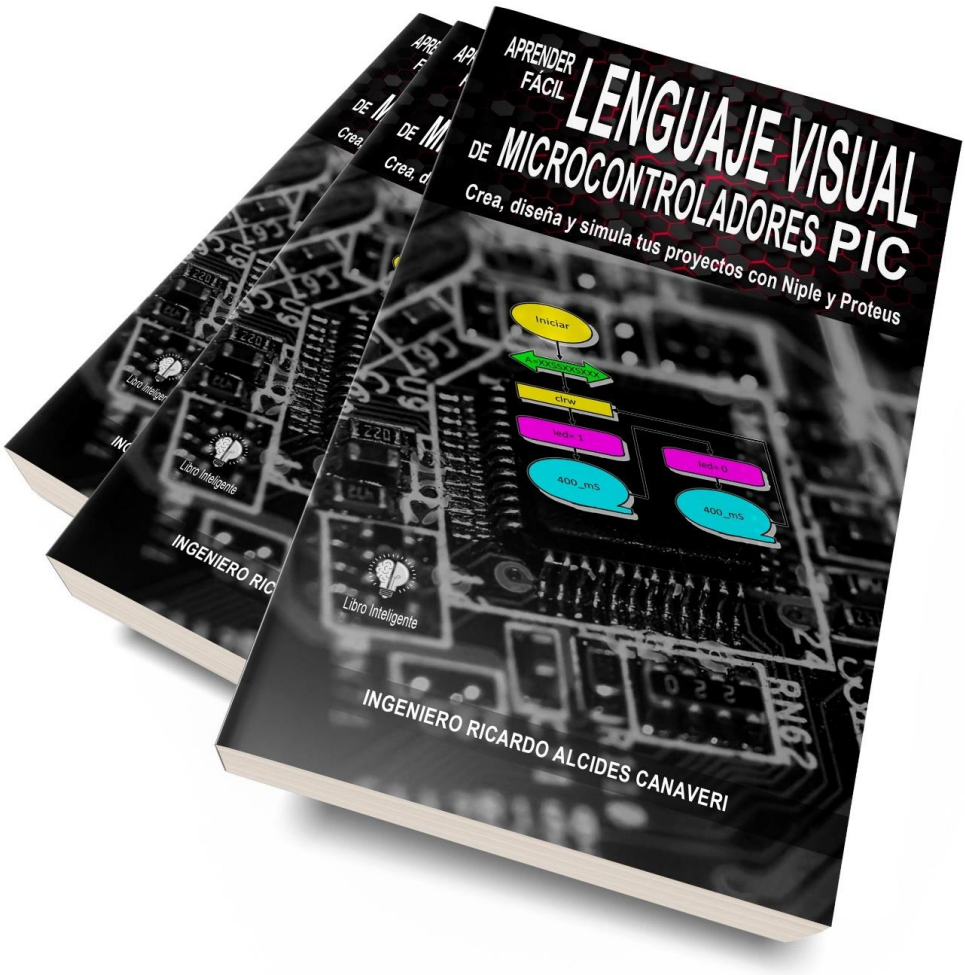
Mensaje para el Lector

Aprender Fácil, nos permite leer, es encender un fuego y cada sílaba que se deletrea es una chispa de magia cuya duración es la vida misma. Esta edición es la fábrica de la imaginación, diseño y compromiso. Si quieres aventurarte, en este facinante camino de aprendizaje, llevado por la lectura.

¡Felicitaciones! Se ha efectuado un gran recorrido, en el que pude transmitir mis experiencias y conocimientos sobre el lenguaje visual de microcontroladores PIC, por supuesto que hay muchas cosas por aprender y experimentar. El objetivo, de esta primera edición, fue que se conozca el emocionante mundo de la programación de microcontroladores PIC, la facilidad que otorga Niple, en resolver en forma fácil y visual la programación, economizando tiempo que es un recurso muy valioso para nuestras vidas, y esperando que les haya gustado, haber simplificado muchos conocimientos de varios años en estas casi 350 páginas de aprendizaje. Agradezco a todos los lectores que hayan leído esta primera edición, quedando en plena disponibilidad de cada interesado. Confiando que a futuro se pueda realizar nuevas ediciones, para ampliar y especificar temáticas más profundas en detalles, lograr una mayor ilustración y mejor comprensión de los conocimientos que pretendo compartir en este *Académico Didáctico*. Deseando a cada lector y aprendiz un ¡Hasta la próxima Edición! Trabajen y hagan lo que más les apasione en sus vidas, recuerden que el tiempo es un recurso limitado y tenemos que aprovecharlo al máximo.

"El recuerdo que deja un libro es más importante que el libro mismo"

Sinceramente Ricardo.


APRENDER FÁCIL
LENGUAJE VISUAL
DE MICROCONTROLADORES PIC
Crea, diseña y simula tus proyectos con Niple y Proteus
Iniciar
400_mS
400_mS
Libro Inteligente
INGENIERO RICARDO ALCIDES CANAVERI

Enlaces de contacto con Ricardo Alcides Canaveri