PROYECTO REXPLER

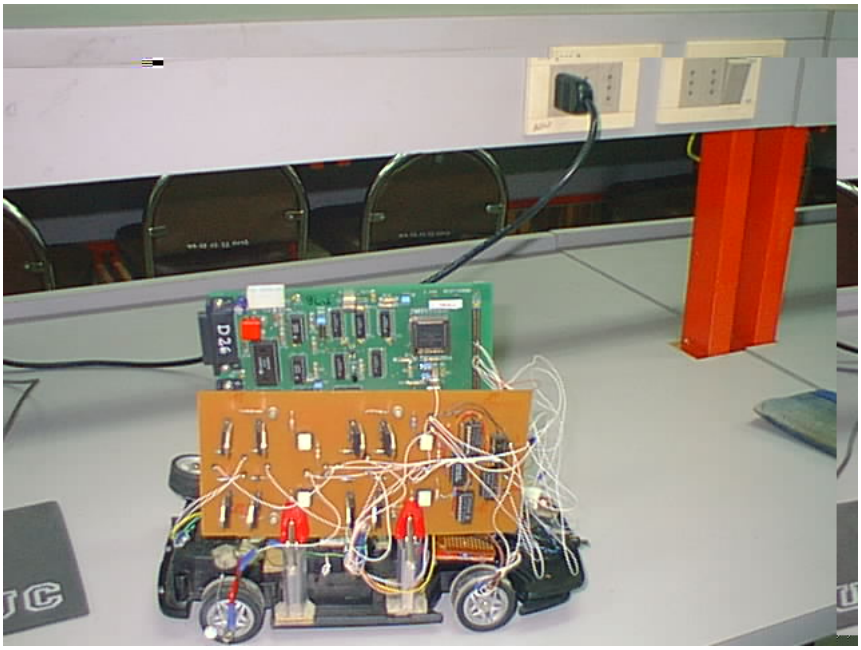
Robot EXPLorador de pERímetros

15 de julio del 2000

Autores: Ariel Guerrero / Luis Jara

https://github.com/aegiloru/rexpler

Prototipo final



UNIVERSIDAD CATOLICA " NUESTRA SENORA DE LA ASUNCION"

MICROPROCESADORES I

Asunción - Paraguay

REXPLER

Robot EXPLorador de pERímetros.

Documentación

- Especificaciones.
- Análisis.
- Implementación hardware
- Implementación software.
- Conclusiones

Recursos

- Download

Autores: Luis Jara / Ariel Guerrero

Fecha: 11/agosto/2000

Cyt | LED | Microprocesares 1

Especificaciones

- Descripción general del sistema.
- Funciones del sistema.
- Descripción de las interfaces.
- Descripción de pruebas
- Descripción de las condiciones de trabajo.
- Descripción de las condiciones ambientales.

Descripción general del sistema

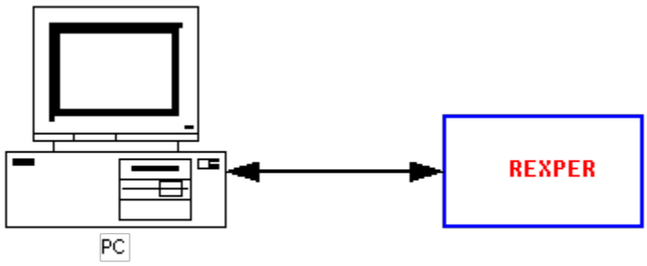


Figura 1: Representación gráfica del sistema.

El trabajo consistirá en el desarrollo de un **sistema de control** para un robot explorador de perimetros **REXPLER**. En la figura 1 se aprecia la representación gráfica del sistema, en la cual, mediante una computadora serán desplegados los datos obtenidos por REXPLER una vez que este haya terminado su misión. Por lo tanto la conexión con la computadora se realizara solo para descarga de datos.

La mision consiste en recorrer los limites de una habitacion vacia, para obtener los datos relativos a su disposicion y a su perimetro. La unidad de medida de longitud utilizada sera el centimetro, y el recorrido se efectuara en sentido horario.

Para dar inicio a una mision la unidad exploradora REXPLER, debera ser colocada de tal forma que quede “mirando” de frente a una de las paredes de la habitacion y a una distancia de por lo menos un metro de la pared mas proxima. La misión termina cuando los limites de la habitacion se haya recorrido por completo y la unidad exploradora descargue los datos de la exploración a la PC.

Funciones del Sistema

1. Controlar el motor de tracción, de tal forma que la unidad avance o retroceda.

2. Controlar el motor de dirección de tal forma que la unidad gire a la izquierda , a la derecha o al frente.

3. Leer los datos provenientes de los sensores.

4. Medir la distancia recorrida.

5. Ofrecer una interfaz sencilla de visualización y comprensión de datos obtenidos por la unidad exploradora al usuario

6. Comunicación con la PC

Descripción de las interfaces.

i) **Interfaz usuario-pc.**

Los datos obtenidos por la unidad REXPLER serán desplegados en pantalla de la PC.

Esta interfaz mostrará en el monitor:

a. El desplazamiento realizado por el explorador desde su posición relativa inicial.

b. El perímetro de dicha configuración.

c. La posición relativa inicial.

La posición relativa inicial es el primer punto en que la unidad encuentra una obstrucción a su desplazamiento.

Esta interfaz mostrara en modo texto cada uno de los vectores recolectados por la unidad exploradora en el formato magnitud, distancia.

Esta interface deberá permitir al usuario "cargar" los datos recolectados y procesados por la unidad exploradora REXPLER.

ii) **Interfaz rexpler usuario**

La unidad REXPLER contará con un panel indicador de estados. Los posibles estados de la unidad exploradora estan especificados en la tabla 1.

Visualizador		Función
Vis1	Vis2	
0	0	Apagado
0	1	Misión
1	0	Comunicación
1	1	Fin de misión

Tabla 1: Estados de la unidad exploradora

iii) **Interfaz REXPLER-pc**

La comunicación de la unidad exploradora con la PC se realizará de acuerdo al protocolo RS-232. La unidad exploradora enviara los datos relativos al resultado de la exploración al la PC.

Descripción de pruebas.

Prueba 1:

Objetivo: Verificar el funcionamiento de la unidad exploradora para una topología sencilla de habitación.

Descripción de la prueba:

a) Colocar a la unidad REXPLER frente a una de las paredes de una habitacion vacia, de dimensiones rectangulares y a una distancia de por lo menos un metro de la pared mas proxima. Cada uno de los lados de la habitacion debe tener como minimo 4 metros de ancho y 6 metros de largo.

b) Verificar constantemente el estado de la unidad exploradora durante el desarrollo de su mision.

c)Cuando el estado sea el de “Comunicación” conectar el RS-232, e iniciar descarga de datos.

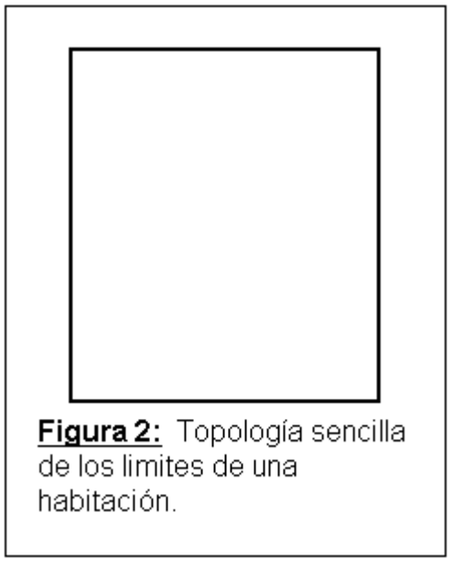


Figura 2: Topología sencilla de los limites de una habitación.

Prueba 2:

Objetivo: Verificar el funcionamiento normal de la unidad REXPER para una topologia ligeramente mas complicada de los limites de una habitacion que la prueba anterior.

Decripcion de la prueba:

a) Ubicandose en la misma habitacion configurar sus limites como en la figura 3 por medio de cajas de carton de una masa de 10 veces mayor a la que tiene la unidad REXPLER como minimo, y de una altura de por lo menos 20 cm. Esto es para hacer que la caja

simule una pared.

b) Repetir el procedimiento realizado durante la prueba uno con esta configuración de limites.

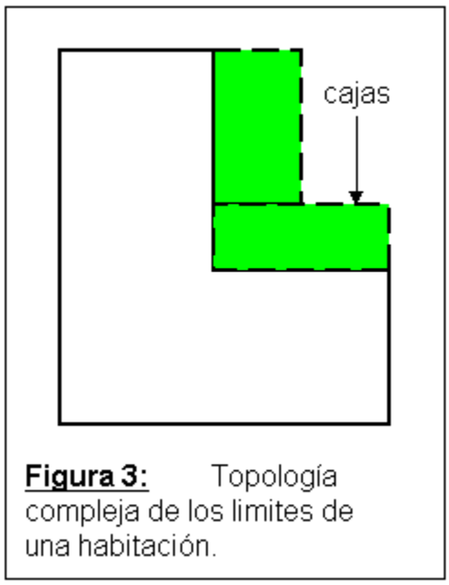


Figura 3: Topología compleja de los limites de una habitación.

Prueba 3:

Objetivos: Verificar el funcionamiento normal de la unidad REXPLER para una topologia compleja de los limites de una habitacion.

Decripcion de la prueba:

a)Utilizando las cajas usadas en la configuración anterior, implementar una topología más compleja que el itém anterio.

b) Repetir el procedimiento realizado durante la prueba uno con esta configuración de limites.

Descripción de las condiciones de trabajo

La habitación a explorar tendrá una superficie plana y lisa.

El contorno debe poseer una forma geometrica tal que pueda ser dividida en rectangulos o cuadrados, siendo las dimensiones de estas figuras geométricas no menor a un metro. La superficie de la habitación no debe exceder los 30 metros cuadrados.

Descripción de las condiciones ambientales.

La unidad exploradora se mantendrá bajo condiciones ambientales de laboratorio.

ANÁLISIS

- Diagrama en bloques del sistema
- Diagrama arboreo general del sistema[DAGS]
- Descripción de los modulos del DAGS

Diagrama en bloques del sistema

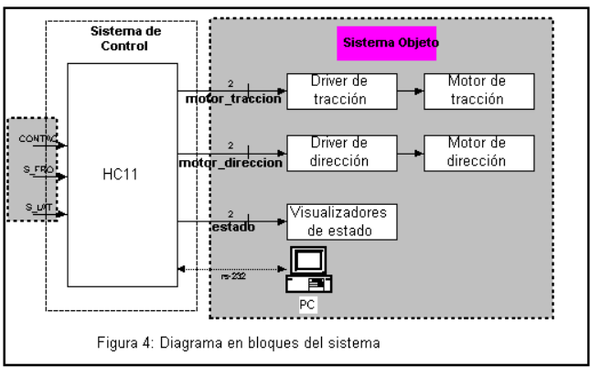


Figura 4: Diagrama en bloques del sistema

El sistema objeto sera una unidad móvil, contara con dos motores y un servomecanismo que serviran para el desplazamiento; además de dos sensores de contacto que determinan la presencia de un objeto en su recorrido.

El sistema de control provee las señales de control del motor de dirección y de tracción del sistema objeto, mediante una interfaz de potencia adecuada. Asi mismo posee los canales adecuados para la comunicacion serial con la PC.

El sistema objeto provee las señales del estado de los sensores frontales y laterales al sistema de control.

Descripción de los modulos del DAGS.

A) MOTOR

Descripción:

Este modulo realiza el control de dirección y tracción del sistema objeto. El modulo puede ser descompuesto en los submodulos de tracción y direccion.

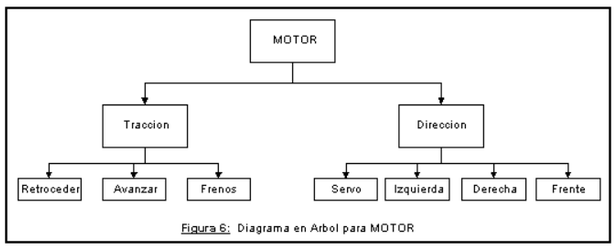


Figura 6: Diagrama en Arbol para MOTOR

A.1 DIRECCION

Descripción:

El sistema objeto contara con un motor de dirección. Este motor nos permitira realizar las maniobras de giro necesarias. El mismo contara con un servomecanismo para este efecto.

En el itém correspondiente a la implementación hardware se detallaran la interacción entre las señales de control de estos módulos y los driver de potencia correspondiente a los mismos.

A.1.1 SERVO

Descripcion:

El servomecanismo de la unidad móvil realimenta señales de la posición actual de las ruedas(IM, DM) de dirección, estas señales son entrada a un modulo combinacional que junto con las entradas del sistema de control(II,DD), determinan cuál sera la nueva dirección(frente, izquierda, derecha). Las salidas del combinacional son las señales I, D. Estas señales son introducidas a una etapa de potencia(DRIVER) que controla el motor de direccion.

Las señales I,D poseen el siguiente comportamiento:

Señal		Función
I	D	
0	0	Frente
0	1	Derecha
1	0	Izquierda
1	1	No permitido

Tabla 2: Señales de control para el combinatorio CC

Implementación: hardware.

A.1.2 FRENTE, IZQUIERDA, DERECHA

Descripción:

Estos modulos permiten al sistema de control establecer las señales II y DD (señal motor_dirección del sistema de control) de manera adecuada, a fin de que el mecanismo de dirección del sistema objeto coloque sus ruedas en las posiciones requeridas.

Implementacion: software

A.2 TRACCION

Descripción:

El sistema objeto contara con un motor de tracción. Este motor nos permite avanzar, frenar o retroceder. Debe poseer una interfaz de potencia(DRIVER) adecuada para interactuar con el sistema de control.

A.2.1 RETROCEDER, AVANZAR, FRENOS.

Descripción de los modulos del DAGS.

Descripción:

Estos modulos permiten al sistema de control establecer las señales FWD, BKWD(señal motor_control del microcontrolador) de manera adecuda para la operación deseada

Señal		Función
Fwd	Bwkd	
0	0	Frenos
0	1	Retroceder
1	0	Avanzar
1	1	No permitido

Tabla 3: Señales de control del modulo de tracción

Implementacion: software

B) SENSOR

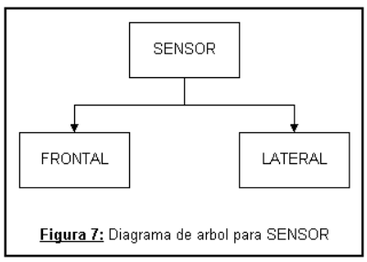


Figura 7: Diagrama de arbol para SENSOR

Descripción:

El sistema objeto contará con sensores de contacto a fin interactuar con el medio y poder determinar la presencia obstaculos en su trayecto.

Las señales provehidas por estos sensores son entradas al sistema de control y activan el estado de las banderas S_FRO, S_LAT, CONTACT, estas banderas constituyen una interface entre los sensores del sistema objeto y el sistema de control.

Referirse al diagrama en bloques del sistema a fin de entender como el sistema se relaciona con estas banderas.

Bandera	Función
S_FRO	Activación del sensor frontal
S_LAT	Desactivación del sensor lateral.
CONTAC	Activación del sensor lateral

Tabla 4: Significado de las banderas del modulo sensor

B.1 FRONTAL, LATERAL

Descripción:

Estos modulos se encargan de establecer de manera adecuada las banderas S_FRO, S_LAT y CONTACT.

Implementación: software

C) ACOPLE

Descripción:

Realiza el procedimiento necesario para que la parte lateral izquierda de la unidad móvil quede apoyada en posición paralela al contorno del lugar a explorar. Dependiendo del sensor activado previamente realiza la maniobra requerida.

Implementación: software.

D) RECOLECCION DE DATOS

Descripción:

Este modulo recogera los datos de magnitud y direccion del tramo recorrido por la unidad movil (sistema objeto).

Descripción de los modulos del DAGS.

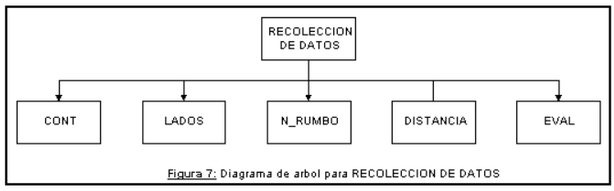


Figura 7: Diagrama de arbol para RECOLECCION DE DATOS

D.1 CONT

Descripcion:

Indicador de magnitud en el tramo recorrido, en la unidad de medida de distancia especificada(ver ESPECIFICACION DEL SISTEMA). Implementación: software

D.2 LADOS

Descripcion:

Indicador de cantidad de tramos recorridos.

Implementación: software.

D.3 N_RUMBO

Descripcion:

Este modulo determina el nuevo rumbo a seguir a partir del rumbo actual y el sensor activado.

Implementación: software.

D.4 DISTANCIA

Descripción:

Este modulo determina el perimetro recorrido por la unidad móvil.

Implementación: software.

D.5 EVAL

Descripción:

Determina si la distancia total recorrida es cero, en cuyo caso indica el fin de recorrido. Esto es indicado en el sistema de control por la bandera FINAL.

Implementación: software.

E) COM

Descripción:

Este modulo se encarga de realizar la comunicación serial entre la unidad móvil y la PC, para transferir los datos recolectados y proveer una interfaz entendible al usuario.

E.1 PCcom

Descripción:

Este modulo permite al usuario cargar los datos recolectados por la unidad móvil a la PC, a fin de poder interpretarlos.

Implementación: software.

E.2 REXPLERcom

Descripción:

Descripción de los modulos del DAGS.

Este modulo transmite los datos recolectados a la PC, via comunicación erial RS-232, del sistema controlador.

Implementación: software y hardware.

Diagrama arboreo general del sistema

Un estudio más detallado de este sistema puede ser descrito mediante el diagrama arboreo general del sistema, en donde se especifican a manera hojas del arbol los módulos que constituyen el sistema.

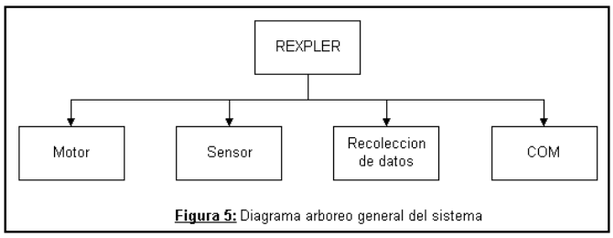


Figura 5: Diagrama arboreo general del sistema

Implementación hardware.

A) SISTEMA DE CONTROL.

El sistema de control es implementado con el microcontrolador M68HC11A8 de MOTOROLA, en particular se utiliza el kit de desarollo M689HC11EVB para proveer las señales de control, adquirir las señales de los sensores y comunicarse serialmente con la PC.

EL diagrama en bloques del sistema, identifica las señales de entrada y salida del sistema de control, en este apartado se asocia el pin del microcontrolador con estas señales.

A.1 Señales de entrada y salida del sistema de control

Señal	Pin	Uso
CONTAC	PA0	ENTRADA
S_LAT	PA1	ENTRADA
S_FRO	PA2	ENTRADA
BKWD	PA5	SALIDA
FWD	PA6	SALIDA
D	PB0	SALIDA
I	PB1	SALIDA
ESTADO	PC0-PC1	SALIDA

Tabla 5: Asignación de pines a las señales de control y señales realimentadas del sistema objeto

Obs.: La descripcion de cada señal se indica en ANALISIS.

A.2 Comunicación serial.

El modulo de comunicación serial COM, utiliza los recurrsos del EVB y del BUFFALO Monitor a fin de implementar el protocolo de parada y espera de comunicación.

Aprovechando que el EVB trabajando en el modo terminal establece una interfaz RS-232 configurado a 9600 baudios, 8 bits de datos, sin paridad y un bit de parada a travez del puerto 2 "TERMINAL I/O PORT CONNECTOR", usando el ACIA MC68B50P.

Se utiliza el mismo conector DB25 - DB9, para conectar el EVB con la PC para la descarga de datos.

Para más detalles consultar el "Evaluation Board User's Manual" del

M68HC11EVB.

B) SISTEMA OBJETO.

El sistema objeto debe ser estudiado a fin de poder implementar los submodulos de TRACCION Y DIRECCION del modulo MOTOR y el modulo de SENSOR. Al efecto se cuenta con dos motores de CD, servomecanimos y ruedas.

B.1 MOTOR

Caracteristicas dinámica de los motores.

Los motores de cd usados para controlar la dirección y tracción de nuestra unidad móvil fueron estudiados bajo diferentes condiciónes de trabajo y alimentación a fin de buscar la configuración de trabajo más adecuada y poder diseñar las interfaces de potencia para los mismos.

Estas medidas fueron obtenidas experimentalmente en laboratorio. La unidad móvil contó con su fuente de alimentación propia de 9[V]. Se medieron la corriente consumida por el motor Im, y la tensión en

sus bornes Vm, y se resumen en la Tabla 5.

Implementación hardware.

Im[A]	Vm[V]	Condición de trabajo
0.146	6.35	Vacio: Las ruedas giran libremente
1	2.5	Rozamiento: Ruedas en contacto con la superficie
1.2	1.5	Traba: Ruedas sujetadas.

Tabla 6: Comportamiento dinámico de los motores de CD.

Hay que destacar que se identfican dos problemas mecánicos que no están resueltos:

1. Los motores de CD a medida que se les entrega más potencia, entran a su zona de saturación, de modo que una potencia inicial de trabajo lo mueve rapidamente, pero una vez que se satura el motor, el mismo se calienta(“perdida de potencia”) y tiene menos “fuerza” por las ruedas, esto implica que la capacidad motora disminuye(velocidad) y las ruedas de dirección pueden “fallar” en al tratar de tomar una dirección.

2. El motor de dirección posee un servomecanimo que controla la recuperación de las ruedas a su posición de equilibrio, este servomecanimo no es muy fiable por poseer un juego, de modo que cuando recupera la dirección de frontal, el mismo varía.

Se opta por un sistema en lazo abierto de velocidad, implementado por el driver de potencia que es un puente H, según se explica en el apartado siguiente, teniendo en cuenta que el desempeño del sistema se estabiliza luego de un cierto tiempo, no obstante los problemas citados, según las experiencias de laboratorio hechas.

Driver de Potencia.

La etapa de potencia, tanto para el motor de dirección como el de tracción esta constituido por un puente H, diseñado tomando en cuenta las caracteristicas de trabajo de nuestros motores de tracción y dirección.

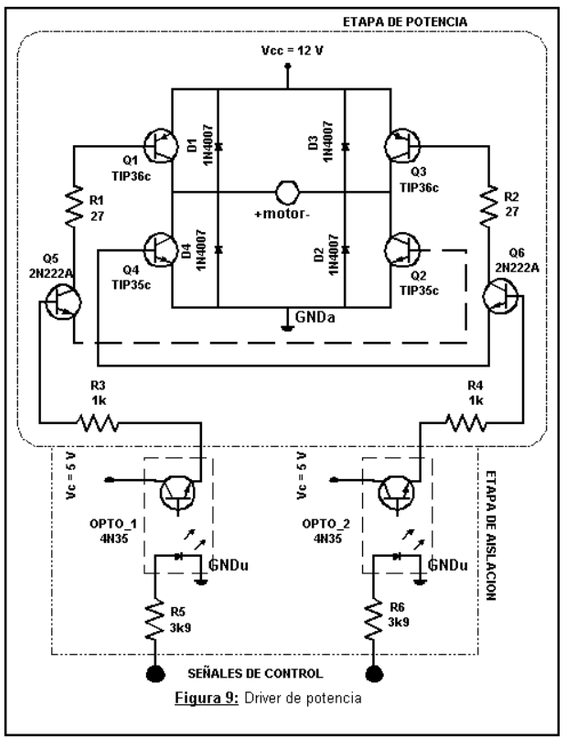


Figura 9: Driver de potencia

EL diseño de esta interfaz provee una corriente maxima de 1[A], con una fuente de 12[V](Vcc) que corresponde a la condición de trabajo maxima de nuestros motores de CD.

En el esquematico se aprecian la disposición de los elementos.

Implementación hardware.

A fin de protejer la parte digital(uC y/o CC) de la carga que pueda proveer la parte analógica, se implementa una etapa de aislación con el optoaislador 4N35, este diseño impone para la parte digital un requerimiento maximo de corriente de 1.5 [mA], a una tensión de trabajo de 5 [V].

Con esto podemos utilizar los pines del microcontrolador o la salida del

combinatorio para comandar el diseño analógico.

El incremento de potencia se controla implementando via software un esquema de modulación por ancho de pulso(pwm).

Motor de tracción.

El driver de potencia diseñado para este motor es un puente H, especificado en el item anterior.

Las señales de entrada a este sistema son Fwd y BKWD(motor_tracción).

Motor de dirección.

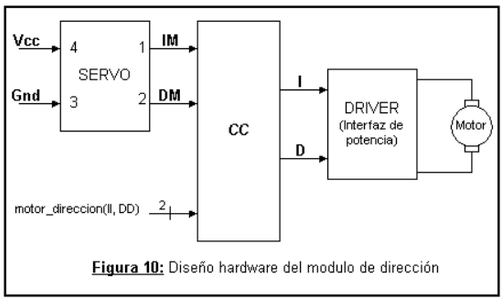


Figura 10: Diseño hardware del modulo de dirección

Este motor cuenta con un mecanismo de servo que nos permite saber el estado actual de las ruedas.

El diseño de la interfaz de potencia es el mismo que el del motor de tracción, excepto que posee un combinatorio CC que recibe las señales IM, DM del servomecanismo e II y DD del microcontrolador. Este combinatorio determina en sus salidas I y D la dirección de las ruedas. Estas salidas son las entradas a la interfaz de potencia.

Experimentalmente se verifico que el mecanismo de servo funciona

de la siguiente manera:

El servomecanismo posee dos entradas y dos salidas, y las mismas

son conectadas entre sí, dependiendo de la posición de las ruedas.

Las entradas estan constituidas por el pin 3 y 4, y las salidaspor el pin 1 y 2.

De manera esquematica se puede sintetizar en 3 modos de funcionamiento.

Modo 1: Ruedas girando totalmente a la izquierda.

Pin 4 conectado al pin 1 y pin 3 conectado al pin 2.

Modo 2: Ruedas al frente.

No existe conexión entre las entradas y las salidas.

Modo 3: Ruedas girando totalmente a la derecha.

Pin 4 conectado al pin 2 y pin 3 conectado al pin 1.

Conectando el pin 4 a Vc(5[V]) y el pin 3 a tierra y analizando las combinaciones posibles de las señales del microcontrolador y del

Implementación hardware.

servo mecanismo obtenemos la siguiente tabla de la verdad:

uC		Servo		Salida	
II	DD	IM	DM	I	D
0	0	0	0	0	0
0	1	0	0	0	1
0	1	0	1	0	0
0	0	0	1	1	0
1	0	0	0	1	0
1	0	1	0	0	0
0	0	1	0	0	1
Otras combinaciones				No permitidas	

Tabla 7: Tabla de la verdad para el driver de dirección CC, asumiendo como entradas el servomecanismo del motor y las ordenes del microcontrolador

El combinatorio CC que resulta de esta tabla es:

I = II . /IM + /DD . /IM . DM

D = DD . /DM + /II . IM . /DM

Este combinatorio es implementado con los IC 7432, 7411, 7414 especificados en el esquemático de la figura 8.

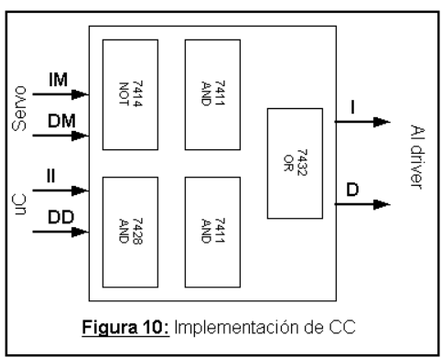


Figura 10: Implementación de CC

B.2 SENSOR

Tanto el sensor lateral como el sensor frontal son sensores de contacto.

Se utiliza los 74244 a fin de proveer niveles logicos TTL a la entrada del microcontrolador asi como antirrobote.

En esquemático de la figura 10 se especifica en forma general este diseño.

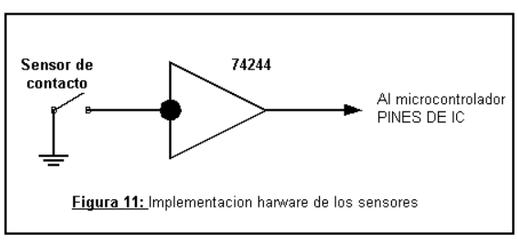


Figura 11: Implementacion harware de los sensores

Implementación software.

- Maquina de estados del sistema
- Rutina principal
- Descripción de las subrutinas principales

Codigo fuente

- Software para el M68HC11A8EVB(asembler)
- Software para 80x86(PC) escrito en C

Maquina de estado del sistema.

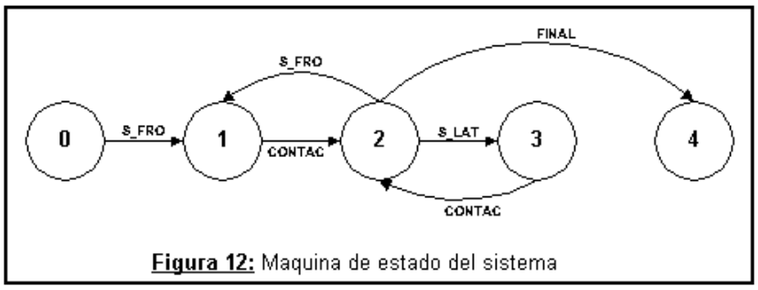


Figura 12: Maquina de estado del sistema

Esta maquina de estado nos permite recorrer el contorno de una habitacion, indicando en cada estado el procedimiento a realizar, resolviendo problemas de las posibles topologias del contorno y la tarea de comunicacion a la pc de los datos recogidos.

Las señales de entrada a la maquina de estado S_FRO, S_LAT, CONTACT, FINAL.

Las primeras señes son obtenidas de los sensores de contacto, la ultima señal indica que la distancia total recorrida es cero lo cual implica que la unidad exploradora a llegado al punto de inicio relativo, y queda esperando la comunicacion con la pc.

Se asume que todos las maniobras de la unidad exploradora son de 90 grados, esto implica que todos los movimientos se caracterizaran por una magnitud que establece la LONGITUD del tramo recorrido y una direccion(NORTE, SUR, ESTE, OESTE) que indica el sentido de recorrido.

La unidad movil realiza siempre un recorrido en el sentido horario.

Estado	Función
0	El robot avanza hasta encontrar un obstaculo,verificando la señal S_FRO.
1	El robot inicia la maniobra de acoplamiento lateral (MANIOBRA_A) a la pared. Como se ilustra en la figura 13, esta rutina termina cuando CONTACT es activado, indicando que la unidad exploradora a quedado paralelo al obstaculo.
2	El robot avanza y al mismo tiempo cuenta la distancia recorrida, buscando siempre la condicion de fin de recorrido(FINAL = ON) pasando al estado de 4. Si recibe alguna señal de los sensores realiza la maniobra correspondiente, guardando en memoria los datos de MAGNITUD y SENTIDO. Si S_LAT se activa, implica que se ha encontrado una discontinuidad de la forma especificada por la Figura 14, su tarea es retomar el acoplamiento (MANIOBRA_B), terminado esto, vuelve al estado 2, cuando CONTAC se activa.

Maquina de estado del sistema.

3 Determinado que se ha satisfecho la condicion de fin de recorrido, se pasa a este estado en donde la unidad movil para y espera a que la unidad de comunicacion PCcom de la pc inicie el protocolo de comunicacion serial para la descarga de datos.

Terminada la comunicacion la unidad indica FIN DE MISION en los VISUALIZADORES.

4

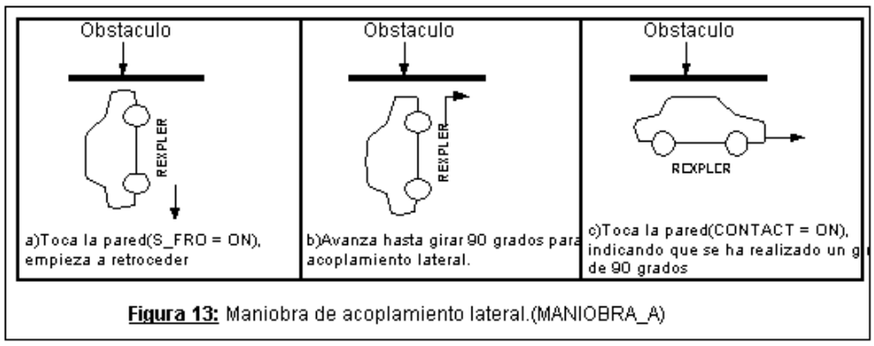


Figura 13: Maniobra de acoplamiento lateral.(MANIOBRA_A)

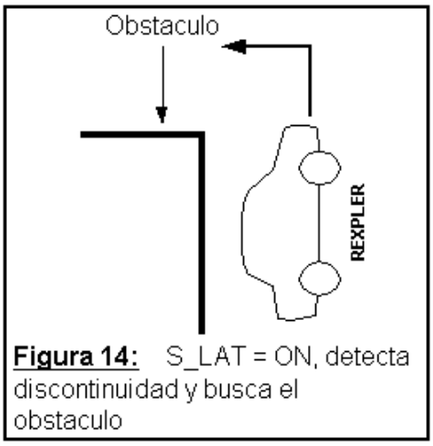


Figura 14: S_LAT = ON, detecta discontinuidad y busca el obstaculo

Rutina principal.

Esta maquina de estado es implementada como se especifica en el diagrama de flujos de la unidad de control.

Las variables principales son S_FRO, CONTACT, S_LAT, estas senales estan relacionadas con la señales del mismo nombre. El microcontrolador posee rutinas de servicio de interrupcion dedicadas a esos pines(INPUT CAPTURE), que actualizan dichas variables. El programa principal se encarga de verificar el estado de estas variables(pooling) y establece al controlador en alguno de los estados posibles del sistema.

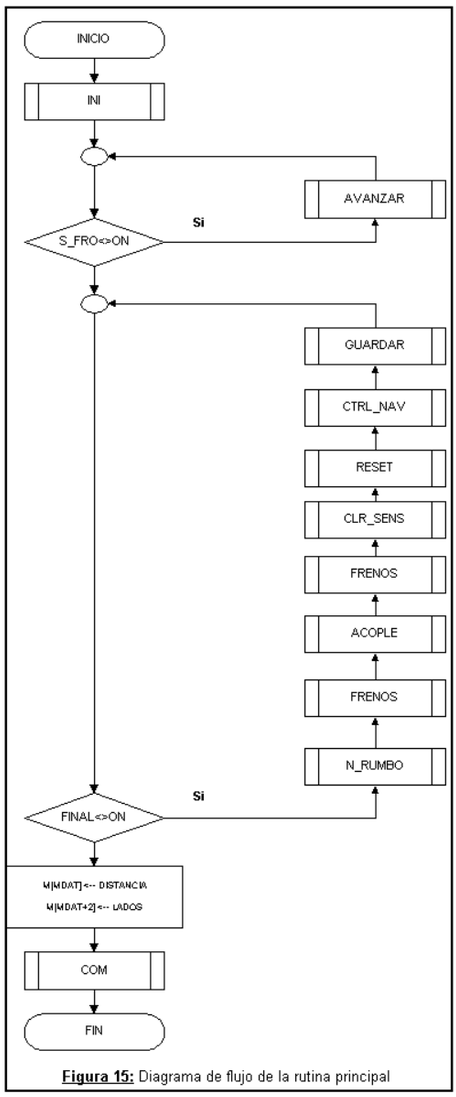


Figura 15: Diagrama de flujo de la rutina principal

Subrutinas Principales.

- INI()
- N_RUMBO()
- AVANZAR()
- FRENOS()
- ACOPLE()
- RESET()
- CTRL_NAV()
- CLR_SENS()
- COM

Subrutina INI.

Esta subrutina inicializa nuestras variables y los registros a usar.

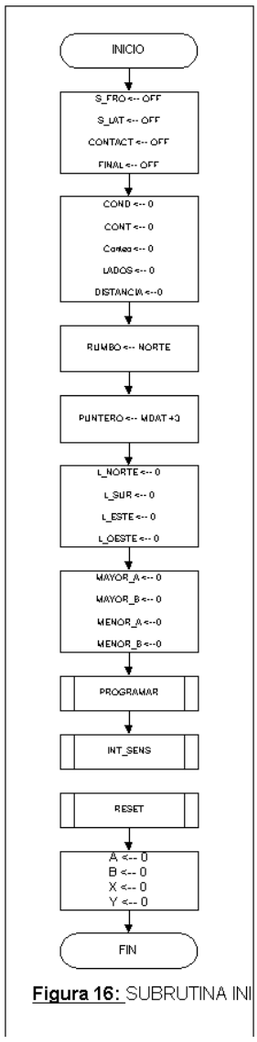


Figura 16: SUBRUTINA INI

Subrutina N_RUMBO.

Determina el nuevo rumbo a seguir a partir del rumbo actual y el sensor activado.

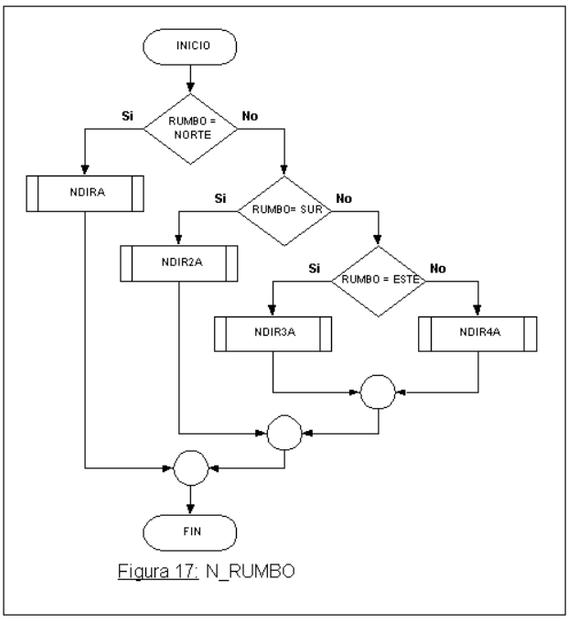


Figura 17: N_RUMBO

Subrutina AVANZAR().

Esta subrutina permite a la unidad exploradora realizar un avance hacia el frente.

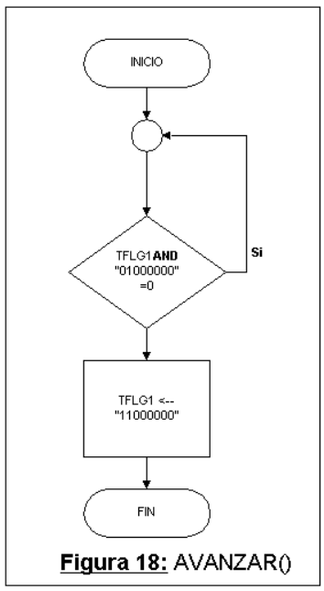


Figura 18: AVANZAR()

Subrutina FRENOS().

Frena el motor de tracción.

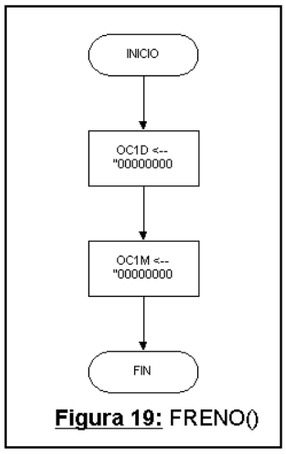


Figura 19: FRENO()

Subrutina ACOPLE().

Dependiendo del sensor activado realiza la maniobra de acople necesaria a fin de que la unidad exploradora permanezca lindando al perimetro.

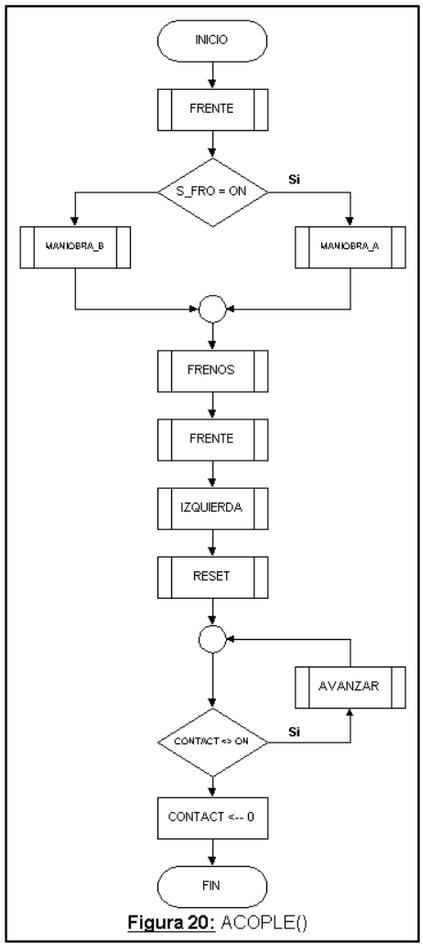


Figura 20: ACOPLE()

Subrutina CTRL_NAV()

Desplaza la unidad exploradora en contacto permanente con la pared, evalua la distancia recorrida a partir del ultimo acople, y verifica la condicion de fin de recorrido.

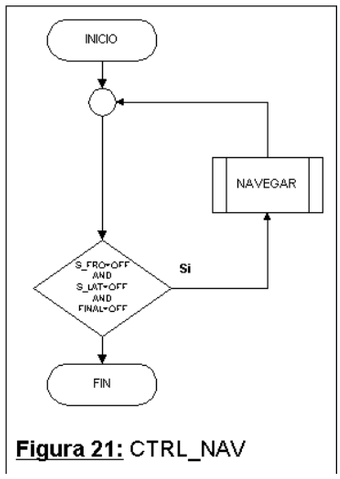


Figura 21: CTRL_NAV

Subrutina RESET().

Configura los registros del TIMER de manera a permitir la maniobra de retroceso.

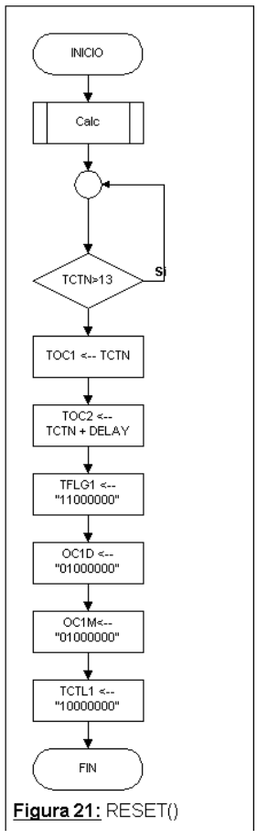


Figura 21: RESET()

Subrutina CLR_SENS().

Restablece los valores de S_FRO, S_LAT y CONTACT a sus valores por defecto en el caso de que hayan sido afectados.

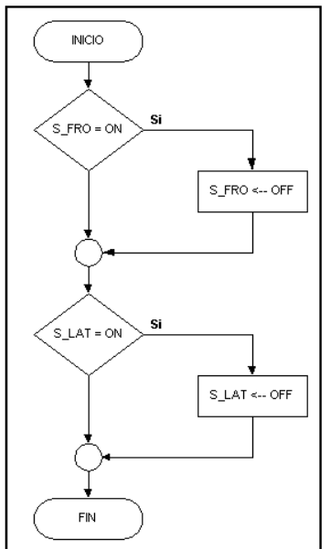


Figura 23: CLR_SENS

Subrutina COM.

Esta subrutina se encarga de comunicar la unidad exploradora con la pc, por tanto posee dos implementaciones, un algoritmo de sincronización para la comunicación en la unidad REXPLER, y otro en la PC, que aparte de sincronizar debe proveer una interfaz al usuario a fin de que el mismo pueda interpretar los resultados obtenidos.

Algoritmo implementado en la unidad exploradora REXPLER.

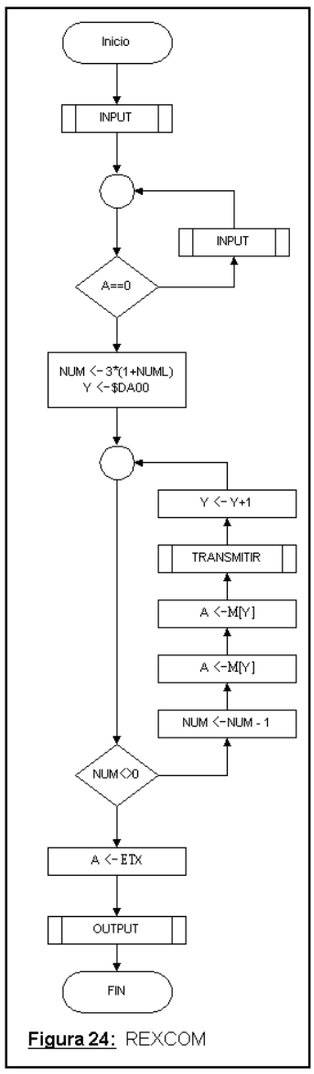


Figura 24: REXCOM

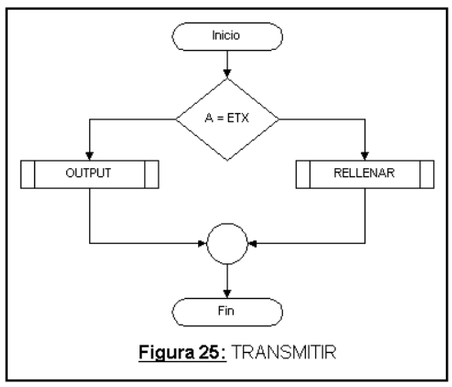


Figura 25: TRANSMITIR

Este algoritmo es la implementacion de un protocolo de parada y espera. La unidad exploradora espera que la PC le indique en que momento transmitir, en el proceso de transmision si encuentra un carácter de control entre los datos a transmitir(ETX), se realiza un procedimiento de rellenado de caracteres a fin de poder distinguir los caracteres de control del dato.

Se utiliza las rutinas de INPUT y OUTPUT del BUFFALO monitor para recibir o transmitir datos del M68HC11A8EVB .

Algoritmo implementado en la PC.

Esencialmente sincroniza la comunicación con la unidad exploradora y formatea los datos recibidos a fin de que sea entendible al usuario humano.

Los datos recibidos se guardan en formato maquina en c:\Rexpler \rexpler.dat y en formato texto en c:\Rexpler\rexpler.txt.

Los datos son mostrados en pantalla.

Subrutina COM.

El algoritmo de sincronización se especifica en el grafico de la figura 26.

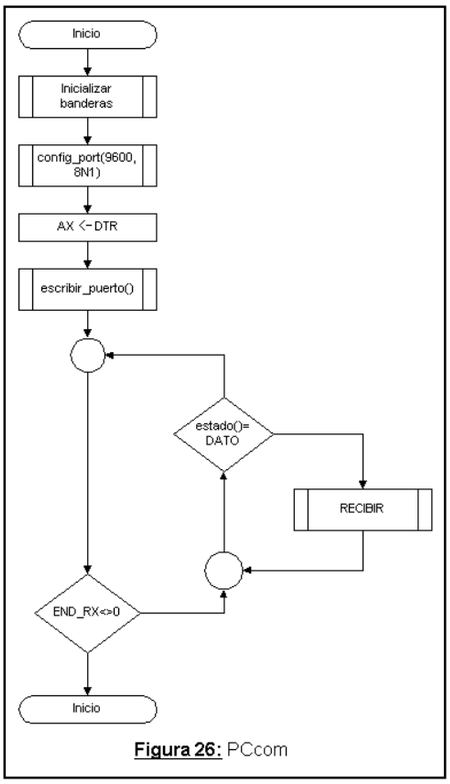


Figura 26: PCcom

El programa que reside en la pc fue desarrollado utilizando BORLAND C 3.1 y se denomina pccom.exe.

Codigo asembler para el M68HC11A8EVB.

```
* ############################################################
*
* REXPLER
* Robot EXPLorador de pERrimetros
* ############################################################
*
* Proposito: Unidad de control de navegación de la unidad
* REXPLER.
*
* Autores: Luis Jara / Ariel Guerrero
*
* Fecha: 15 de julio del 2000.
* ############################################################
* Version Fecha Comentario
* ------- ----- ----------
* 0 15/07/00 Inicio de codificacion
* 1 19/07/00 Prueba de sensores
* 2 21/07/00 Documentacion
* 3 21/07/00 Maniobras de acoplamiento
* 4 28/07/00 Documentacion
* 5 28/07/00 Guardado de datos y condicion de Final
* 6 29/07/00 Modificacion de MANIOBRA_B
* 7 05/08/00 Aplicacion de tres velocidades
* 8 10/08/00 Documentacion final
* ############################################################

* EQUATES
NORTE EQU $4E Direccion Norte de movimiento
SUR EQU $53 Direccion Sur de moviminto
ESTE EQU $45 Direccion Este de movimineto
OESTE EQU $4F Direccion Oeste de movimiento

BITS EQU %01000000 Patron de bits para salida
BITS_R EQU %00100000 Patron de bits para salida en reversa
ALLBITS EQU %01000000 Output bit-6
ALLBITR EQU %00100000 Output bit-5
TOF EQU %10000000 Timer overflow flag
OC1F EQU %10000000 Output compare 1 flag
OC2F EQU %01000000 Output compare 2 flag
OC3F EQU %00100000 Output compare 3 flag
STOP EQU %00000000 Limpia el pin OC2
IC1F EQU %00000100 Activa Int1
IC2F EQU %00000010 Activa Int2
IC3F EQU %00000001 Activa Int3
EDGF EQU %00100000 Imput capture edge (BAJADA)
EDGL EQU %00000100 Imput capture edge (SUBIDA)
EDGLB EQU %00000010 Imput capture edge (BAJADA)
OC2R EQU %10000000 OM2,OL2 to reset bit-6 to zero
OC3R EQU %00100000 OM3,OL3 to reset bit-5 to zero
IZQ EQU %00000010 Gira a la izquierda
FRE EQU %00000000 Al frente
DER EQU %00000001 Gira a la derecha
OUTPORTC EQU %00000011 PC(0) y PC(1) como salida
MISION EQU %00000001 Estado del sistema en Mision
COMUNICACION EQU %00000010 Estado del sistema en Comunicacion
FINISCH EQU %00000011 Estado del Sitema en fin de Mision
CNFI EQU %01000000 Configuracion para el Pulse Acumulator
```

Codigo asembler para el M68HC11A8EVB.

```
UnPorc EQU 655 1 % de 65536 ciclos
NTIMES EQU 93 Numero de TOF's
ON EQU 1 Sensor en estado activo
OFF EQU 0 Sensor en estado de desactivacion
CLEAR EQU 0 Se utiliza para limpiar registros
TOPE EQU 31 Cantidad de periodos del TCTN en 1 seg.
TOPE1 EQU 31 Cantidad de periodos del TCNT por cm. desplazado
CICLU EQU 10 Ciclo util para el DRIVER de Traccion en avance inicial
CICLU1 EQU 10 Ciclo util para el DRIVER de Traccion en maniobras (retroceso)
CICLU2 EQU 10 Ciclo util para el DRIVER de Traccion en maniobras (avance)
CICLU3 EQU 10 Ciclo util para el DRIVER de Traccion en recorrido
AT_MNA EQU 10 Tiempo de retroceso en MANIOBRA_A por primera vez
AT_MNA2 EQU 10 Tiempo de retroceso en MANIOBRA_A las demas veces
AD_MNA EQU 10 Tiempo de avance en MANIOBRA_A por primera vez
AD_MNA2 EQU 10 Tiempo de avance en MANIBRA_A las demas veces
AD_MNB EQU 10 Tiempo de avance en MANIOBRA_B
AT_MNB EQU 10 Tiempo de retroceso en MANIOBRA_B
ADEMNB EQU 10 Tiempo de avance en MANIOBRA_A
PRECISION EQU 10 Error permitido para calcular la distancia total
MIN EQU 4 Cantidad minima de lados de una habitacion
NTIMESS EQU 20 Delay para implementar el antirrebote
PER EQU 12 Desplazamiento en centimetro de una vuelta de la rueda

* Definicion de constantes de memoria
PRG EQU $C000 Inicio de area de codigo
DAT EQU $D000 Inicio de area de datos
STACK EQU $DFFF Inicio de area de pila
MDAT EQU $D0FF Inicio de area de datos recolectados

* Definición de registros I/O
REGS EQU $1000 Direccion de referencia
DDRC EQU $07 Direccion del dato para el puerto C
TFLG2 EQU $25 Registro TFLG2
TCNT EQU $0E TCNT register
TFLG1 EQU $23 TFLG1 offset
TCTL1 EQU $20 Timer control reg 1
TCTL2 EQU $21 Timer control reg 2
TMSK1 EQU $22 Main Timer Interrupt Mask Register
TOC1 EQU $16 TOC1 register
TOC2 EQU $18 TOC2 register
TOC3 EQU $1A TOC3 register
OC1M EQU $0C Output compare 1 mask
OC1D EQU $0D Output compare 1 data
PORTB EQU $04 Puerto B
PORTC EQU $03 Puerto C
PACNT EQU $27 Pulse Acumulator Count Register
PACTL EQU $26 Pulse Acumulat

* ASCII equates
ETX equ 3
NUL equ 0
BEL equ 7

* Constantes equates
uDAT equ $DA00 Inicio DATOS de comunicacion
NUML equ $DA02 Numero de lados recorridos
TRES equ $03
```

Codigo asembler para el M68HC11A8EVB.

```
* Vectores de interrupcion
IC1V EQU $00E8 IC1 interrupt vector
IC2V EQU $00E5 IC2 interrupt vector
IC3V EQU $00E2 IC3 interrupt vector

*BUFFALO equtes
OUTPUT equ $E3B3 Rutinas de I/O del buffalo
INPUT equ $E387


* Definicion de variables
org DAT
S_LAT rmb 1 Sensor lateral desactivado
S_FRO rmb 1 Sensor frontal activado
FINAL rmb 1 Indicador de fin de mision
CONTACT rmb 1 Sensor lateral activado
CONT rmb 2 Contador de distancia
COUNTER rmb 1 Para el retardo en TRESEG
RUMBO rmb 1 Indicador de direccion de movimiento
Ciclo rmb 2 Indica el ciclo util
Cuant rmb 2 Variable auxiliar en Calc
DELAY rmb 2 Retardo para poder implementar el ciclo util
CONTD rmb 1 Contador bandera en NAVEGAR
Conteo rmb 1 Igual a CONTD pero para EVALRN
PUNTERO rmb 2 Apunta la direccion en donde se guardaran los datos
DELAY1 rmb 1 Retardo para retroceder en MANIOBRA_A
DELAY12 rmb 1 Retardo para retroceder en MANIOBRA_A
DELAY2 rmb 1 Retardo para avanzar MANIOBRA_A
DELAY22 rmb 1 Retardo para avanzar MANIOBRA_A
DELAY3 rmb 1 Retardo para avanzar MANIOBRA_B
DELAY4 rmb 1 Retardo para retroceder en MANIOBRA_B
DELAY5 rmb 1 Retardo para avanzar en MANIOBRA_B
RETAR1 rmb 1 Variable auxiliar para retroceder en MANIOBRA_A
RETAR12 rmb 1 Variable auxiliar para retroceder en MANIOBRA_A
RETAR1A rmb 1 Variable auxiliar para retroceder en MANIOBRA_A
RETAR2 rmb 1 Variable auxiliar para avanzar en MANIOBRA_A
RETAR22 rmb 1 Variable auxiliar para avanzar en MANIOBRA_A
RETAR2A rmb 1 Variable auxiliar para avanzar en MANIOBRA_A
RETAR3 rmb 1 Variable auxiliar para avanzar en MANIOBRA_B
RETAR4 rmb 1 Variable auxiliar para retroceder en MANIOBRA_B
RETAR5 rmb 1 Variable auxiliar para avanzar en MANIOBRA_B
BANDERA rmb 1 Bandera en MANIOBRA_A que identifica la primera maniobra de las demas
LADOS rmb 1 Cantidad de lados de la habitacion
L_NORTE rmb 2 Distancia recorrida en la direccion NORTE
L_SUR rmb 2 Distancia recorrida en la direccion SUR
L_ESTE rmb 2 Distancia recorrida en la direccion ESTE
L_OESTE rmb 2 Distancia recorrida en la direccion OESTE
MAYOR_A rmb 2 Variable utilizada para evaluuar la condicion de fin
MENOR_A rmb 2 Variable utilizada para evaluuar la condicion de fin
MAYOR_B rmb 2 Variable utilizada para evaluuar la condicion de fin
MENOR_B rmb 2 Variable utilizada para evaluuar la condicion de fin
DISTANCIA rmb 2 Indica la distancia total recorrida
Demora rmb 1 Delay introducido para evitar el rebote
PAVIEJO rmb 1 Valor antiguo del registro PACNT
NUM rmb 2
CNTcom rmb 1


* ############################################################
```

Codigo asembler para el M68HC11A8EVB.

```
* Principal
* ###########################################################
org PRG
lds #STACK

* Inicilizacion de variables y registros
jsr INI

* Estado del sistema en Mision
ldx #REGS
ldaa #MISION
staa PORTC,x
*
* LOOP1
* Mientras (!S_FRO)
* AVANZAR();
*
LOOP1 tst S_FRO
bne END_L1

jsr AVANZAR
bra LOOP1

END_L1 nop

*
* LOOP2
* Mientras (!FINAL)
* N_RUMBO();
* FRENOS();
* ACOPLE();
* FRENOS();
* CLR_SENS();
* RESET();
* CTRL_NAV();
* GUARDAR();
*

LOOP2 tst FINAL
bne END_L2

jsr N_RUMBO
jsr FRENOS
jsr ACOPLE
jsr FRENOS
jsr CLR_SENS
jsr RESET
jsr CTRL_NAV
jsr GUARDAR

bra LOOP2

END_L2 jsr FRENOS

* Se guardan los datos de distancia total recorrida y cantidad de lados
* Los primeros dos bytes para la distancia, y el sgt. para la cant. de lados
ldx #MDAT
ldy DISTANCIA
sty 0,x
```

Codigo asembler para el M68HC11A8EVB.

```
ldaa LADOS
staa 2,x

* Estado en Comunicacion con la PC
ldx #REGS
ldaa #COMUNICACION
staa PORTC,x

* Esperando comunicacion
jsr COM

* Fin de Mision
ldx #REGS
ldaa #FINISCH
staa PORTC,x
* ############################################################
swi fin programa
* ############################################################




* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
* SUBRUTINAS
* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!




* ************************************************************
* INI
* ************************************************************
* Proposito: Inicializa variables y registros.
*
* ************************************************************
INI
* Inicializacion de los estados de los sensores
clr S_FRO
clr S_LAT
clr CONTACT

* Inicializacion de variables en general
clr FINAL
clr CONTD
clr Conteo
clr LADOS
clr BANDERA

* Inicializacion de la variable PUNTERO
ldx #MDAT
ldab #3
abx
stx PUNTERO

* Inicializacion de variables de 2 BYTES
ldx #CLEAR
stx CONT
stx L_NORTE
stx L_SUR
stx L_ESTE
stx L_OESTE
```

Codigo asembler para el M68HC11A8EVB.

```
stx MAYOR_A
stx MAYOR_B
stx MENOR_A
stx MENOR_B
stx DISTANCIA

* Definicion de la primera direccion de movimiento
ldaa #NORTE
staa RUMBO RUMBO <- NORTE

* Cofiguracion para el pulse acumulator
ldx #REGS
ldaa #CNFI
staa PACTL,x

* Se limpia el registro PACNT y la variable PAVIEJO
clr PACNT,x
clr PAVIEJO

* Inicializa el ciclo util y los tiempos de retardo
jsr PROGRAMAR

* Inicializa la interrupcion para los sensores
jsr INT_SENS

* Espera el encenndido
LOOP3 ldaa S_FRO
cmpa #OFF
beq LOOP3

* Introduce una espera para prevenir rebotes
jsr TRESEG

* Restablece el valor de S_FRO
ldaa #OFF
staa S_FRO

* Configura los registros del TIMER para el avance
jsr RESET

* Inicializacion de variables de retardo
ldaa DELAY1
staa RETAR1
ldaa DELAY12
staa RETAR12
ldaa DELAY2
staa RETAR2
ldaa DELAY22
staa RETAR22
ldaa DELAY3
staa RETAR3
ldaa DELAY4
staa RETAR4
ldaa DELAY5
staa RETAR5

* Cofiguracion del Bit 0 y 1 del Puerto C para Salida
ldx #REGS
ldaa #OUTPORTC
```

Codigo asembler para el M68HC11A8EVB.

```
staa DDRC,x

* Limpieza de registros de proposito general
clra
clrb
ldx #CLEAR
ldy #CLEAR

rts
*-------------------------------------------------------------
* PROGRAMAR
*-------------------------------------------------------------
* Proposito: Programa el ciclo util de la senhal de exitacion
* para el DRIVER de Traccion. Ademas determina los tiempos
* de retroceso y avance de las diferentes maniobras
*
*-------------------------------------------------------------
PROGRAMAR

* Determina los retardos en las maniobras
ldaa #AT_MNA
staa DELAY1

ldaa #AT_MNA2
staa DELAY12

ldaa #AD_MNA
staa DELAY2

ldaa #AD_MNA2
staa DELAY22

ldaa #AD_MNB
staa DELAY3

ldaa #AT_MNB
staa DELAY4

ldaa #ADEMNB
staa DELAY5

* Determina el ciclo util inicial
ldy #CICLU
sty Ciclo

rts
*-------------------------------------------------------------
* INT_SENS
*-------------------------------------------------------------
* Proposito: Se encarga se inicializar el imput capture IC1,
* IC2 y IC3 para el sensor frontal y lateral
*
*-------------------------------------------------------------
INT_SENS

ldx #REGS

* Se configura para capturar los flancos de bajada (IC1) y
* de subida (IC2) y bajada (IC3)
```

Codigo asembler para el M68HC11A8EVB.

```
ldaa #EDGF|EDGL|EDGLB
oraa TCTL2,x
staa TCTL2,x

* Se setean las banderas del imput capture IC1F, IC2F, IC3F
ldaa #IC1F|IC2F|IC3F
oraa TFLG1,x
staa TFLG1,x

* Se habilita el imput capture para IC1, IC2 e IC3
ldaa #IC1F|IC2F|IC3F
oraa TMSK1,x
staa TMSK1,x

* Se habilitan las interrupciones
cli


rts

* ************************************************************
* TRESEG
* ************************************************************
* Proposito: Introduce un retardo de 3 segundos de retraso
* ************************************************************
TRESEG
ldx #REGS
* Se limpia primeramente el TOF
ldaa #TOF
staa TFLG2,x

* Se inicializa el contador y se espera por NTIMES
ldaa #NTIMES
staa COUNTER

* Se espera mientras el TOF no es seteado
SPIN1 ldaa TFLG2,x
bita #TOF
beq SPIN1

* Despues de que TOF=1; se vuelve a setear
ldaa #TOF
staa TFLG2,x

* Y se decrementa el contador
dec COUNTER

* IF COUNTER != 0 Bifurca
bne SPIN1
rts
* ************************************************************
* AVANZAR
* ************************************************************
* Proposito: La unidad REXPLER avanza en direccion al
* frente.
* ************************************************************
AVANZAR

ldx #REGS
```

Codigo asembler para el M68HC11A8EVB.

```
spin brclr TFLG1,x OC2F spin

* Reset the OC1F, OC2F
ldaa #OC1F|OC2F
oraa TFLG1,x
staa TFLG1,x
rts
*****************************************************************
* FRENOS
*****************************************************************
* Proposito: Esta rutina es utilizada para frenar los motores
* traccion
*****************************************************************
FRENOS
ldx #REGS
ldaa #STOP
staa OC1D,x
staa OC1M,x
rts
* ***************************************************************
* ACOPLE
* ***************************************************************
* Proposito: Segun S_FRO o S_LAT, se realiza una de las
* maniobras de acoplamiento lateral (tipo A o B).
* ***************************************************************
ACOPLE

* Establece velocidad para retroceso para las maniobras
ldy #CICLU1
sty Ciclo

* Pone las ruedas de direccion al frente
jsr FRENTE

*
* IF S_FRO = ON THEN
* MANIOBRA_A()
*
* ELSE IF
* MANIOBRA_B()
*
ldaa #ON
cmpa S_FRO
bne SN_AC

jsr MANIOBRA_A
bra FS_AC

SN_AC jsr MANIOBRA_B

* Empieza a buscar la pared
FS_AC jsr FRENOS
jsr FRENTE
jsr IZQUIERDA
jsr RESET

* Mientras no encuentra la pared...avanza
M_AC ldaa #ON
```

Codigo asembler para el M68HC11A8EVB.

```
cmpa CONTACT
beq FM_AC

jsr AVANZAR
bra M_AC

* Se restablece el valor de CONTACT
FM_AC clr CONTACT

* Direccion al frente para seguir la pared
jsr FRENTE

rts
*-------------------------------------------------------------
* MANIOBRA_A
*-------------------------------------------------------------
* Proposito: Esta rutina se encarga de realizar los procedimientos
* necesarios de girar 90 grados a la derecha
*-------------------------------------------------------------
MANIOBRA_A
*
* IF BANDERA=ON THEN
* RETAR1A <-- RETAR12
* RETAR2A <-- RETAR22
* ELSE
* RETAR1A <-- RETAR1
* RETAR2A <-- RETAR2
* BANDERA <-- ON
*
ldaa #OFF
cmpa BANDERA
beq SN_BAN

* Aqui entra la segunda vez en adelante
ldaa RETAR12
staa RETAR1A
ldaa RETAR22
staa RETAR2A
bra FS_BAN

* Aqui entra la primera vez solamente
SN_BAN ldaa RETAR1
staa RETAR1A
ldaa RETAR2
staa RETAR2A

* Se activa una bandera para indicar que ya paso por este tramo
ldaa #ON
staa BANDERA

FS_BAN nop

*
* REVERSA();
* Conteo <-- 0
*
* Se configura los registros del TIMER para el retroceso
jsr REVERSA
```

Codigo asembler para el M68HC11A8EVB.

```
* Se reinicializa esta variable para el retardo en EVALRN
clr Conteo

*
* MIENTRAS RETAR1A<>0 HACER
* RETROCEDER();
* EVALRN();
*
M1_MNA tst RETAR1A
beq FM1_MNA
jsr RETROCEDER
ldaa RETAR1A
jsr EVALRN
staa RETAR1A

bra M1_MNA

FM1_MNA nop
*
* FRENOS()
* DERECHA()
* RESET()
* Conteo <-- 0
*
* Se prepara para girar a la derecha
jsr FRENOS
jsr DERECHA
jsr RESET
clr Conteo
*
* MIENTRAS RETAR2A<>0 HACER
* AVANZAR()
* EVALRN()
*
M2_MNA tst RETAR2A
beq FM2_MNA

jsr AVANZAR
ldaa RETAR2A
jsr EVALRN
staa RETAR2A
bra M2_MNA

FM2_MNA nop

rts
*////////////////////////////////////////////////////////
* REVERSA
*////////////////////////////////////////////////////////
* Proposito: Inicializa los registros del TIMER para
* marcha atras
*////////////////////////////////////////////////////////
REVERSA
jsr Calc
ldx #REGS

* Grab the value of the TCNT register tal que sea < a 14
loopr ldd TCNT,x
cpd #14
```

Codigo asembler para el M68HC11A8EVB.

```
bhs loopr

std TOC1,x
*Set TOC3 to compare DELAY cycles later
addd DELAY
std TOC3,x

* Reset output compare flags
ldaa #OC1F|OC3F
oraa TFLG1,x
staa TFLG1,x

* Initialize the data register
ldaa #BITS_R
oraa OC1D,x
staa OC1D,x

* Initialize the mask register
ldaa #ALLBITR
oraa OC1M,x
staa OC1M,x

* Set up OC2 to reset the bit
ldaa #OC3R
oraa TCTL1,x
staa TCTL1,x
rts
*////////////////////////////////////////////////////////
* RETROCEDER
*////////////////////////////////////////////////////////
* Proposito: Esta rutina hace posible que el ROBOT retroceda
*
*////////////////////////////////////////////////////////
RETROCEDER
spinR brclr TFLG1,x OC3F spinR

* Reset the OC1F, OC3F
ldaa #OC1F|OC3F
oraa TFLG1,x
staa TFLG1,x
rts
*////////////////////////////////////////////////////////
* EVALRN
*////////////////////////////////////////////////////////
* Proposito: Evalua los retardos para las diferentes maniobras
*
*////////////////////////////////////////////////////////
EVALRN
ldab Conteo
cmpb #TOPE
bne SN_EVR

deca
clr Conteo
bra FS_EVR

SN_EVR inc Conteo

FS_EVR rts
```

Codigo asembler para el M68HC11A8EVB.

```
*////////////////////////////////////////////////////
* FRENTE
*////////////////////////////////////////////////////
* Proposito: Posiciona las ruedas de direccion al frente
*////////////////////////////////////////////////////
FRENTE
ldx #REGS
ldaa #FRE
staa PORTB,x
rts
*////////////////////////////////////////////////////
* DERECHA
*////////////////////////////////////////////////////
* Proposito: Posiciona las ruedas de direccion a la derecha
*////////////////////////////////////////////////////
DERECHA
ldx #REGS
ldaa #DER
staa PORTB,x
rts
*////////////////////////////////////////////////////
* IZQUIERDA
*////////////////////////////////////////////////////
* Proposito: Posiciona las ruedas de direccion a la izquierda
*////////////////////////////////////////////////////
IZQUIERDA
ldx #REGS
ldaa #IZQ
staa PORTB,x
rts
*-------------------------------------------------------------
* MANIOBRA_B
*-------------------------------------------------------------
* Proposito: Esta rutina se encarga de realizar los procedimientos
* necesarios de girar 90 grados a la izquierda
*-------------------------------------------------------------
MANIOBRA_B

*
* RESET();
* Conteo <-- 0
*
* Se configuran los registros del TIMER para el avance
jsr RESET

* Se inicializa la variable conteo para el retardo
clr Conteo
*
* MIENTRAS RETAR3<>0 HACER
* AVANZAR()
* EVALRN()
*

M2_MNB tst RETAR3
beq FM2_MNB

jsr AVANZAR
ldaa RETAR3
jsr EVALRN
```

Codigo asembler para el M68HC11A8EVB.

```
staa RETAR3
bra M2_MNB

FM2_MNB nop

* Restablece la variable RETAR3
ldaa DELAY3
staa RETAR3
* Se prepara para retroceder
jsr FRENOS
jsr DERECHA
jsr REVERSA
clr Conteo

*
* MIENTRAS RETAR4<>0 HACER
* RETROCEDER()
* EVALRN()
*
M1_MNB tst RETAR4
beq FM1_MNB
jsr RETROCEDER
ldaa RETAR4
jsr EVALRN
staa RETAR4
bra M1_MNB
FM1_MNB nop

* Restablece la variable RETAR4
ldaa DELAY4
staa RETAR4

* Se preparara para avanzar
jsr FRENOS
jsr FRENTE
jsr RESET
clr Conteo
*
* MIENTRAS RETAR5<>0 HACER
* AVANZAR()
* EVALRN()
*
M5_MNB tst RETAR5
beq FM5_MNB
jsr AVANZAR
ldaa RETAR5
jsr EVALRN
staa RETAR5
bra M5_MNB
FM5_MNB nop

* Restablece la variable RETAR5
ldaa DELAY5
staa RETAR5

rts
* ************************************************************
* N_RUMBO
* ************************************************************
```

Codigo asembler para el M68HC11A8EVB.

```
* Proposito: Determina el proximo rumbo a partir del punto
* de acople.
* ************************************************************
N_RUMBO

ldaa RUMBO
cmpa #NORTE
beq NDIR1
cmpa #SUR
beq NDIR2
cmpa #ESTE
beq NDIR3
cmpa #OESTE
beq NDIR4

FIN_NR nop
rts

NDIR1 jsr NDIR1A
jmp FIN_NR

NDIR2 jsr NDIR2A
jmp FIN_NR

NDIR3 jsr NDIR3A
jmp FIN_NR

NDIR4 jsr NDIR4A
jmp FIN_NR

* ------------------------------------------------------------
* NDIR1A
* ------------------------------------------------------------
* Proposito: Asume que la direccion actual es norte y
* determina el nuevo rumbo cuando el sensor lateral lo
* indique
* ------------------------------------------------------------
NDIR1A
ldaa S_LAT
cmpa #ON
beq DIRN_O
ldaa #ESTE
staa RUMBO

FIN_ND1A nop
rts

DIRN_O ldaa #OESTE
staa RUMBO
bra FIN_ND1A
* ------------------------------------------------------------
* NDIR2A
* ------------------------------------------------------------
* Proposito: Asume que la direccion actual es sur y
* determina el nuevo rumbo cuando el sensor lateral lo
* indique
* ------------------------------------------------------------
NDIR2A
ldaa S_LAT
```

Codigo asembler para el M68HC11A8EVB.

```
cmpa #ON
beq DIRS_E
ldaa #OESTE
staa RUMBO

FIN_ND2A nop
rts

DIRS_E ldaa #ESTE
staa RUMBO
bra FIN_ND2A

* ------------------------------------------------------------
* NDIR3A
* ------------------------------------------------------------
* Proposito: Asume que la direccion actual es este y
* determina el nuevo rumbo cuando el sensor lateral lo
* indique
* ------------------------------------------------------------
NDIR3A
ldaa S_LAT
cmpa #ON
beq DIRE_N
ldaa #SUR
staa RUMBO

FIN_ND3A nop
rts

DIRE_N ldaa #NORTE
staa RUMBO
bra FIN_ND3A

* ------------------------------------------------------------
* NDIR4A
* ------------------------------------------------------------
* Proposito: Asume que la direccion actual es oeste y
* determina el nuevo rumbo cuando el sensor lateral lo
* indique
* ------------------------------------------------------------
NDIR4A
ldaa S_LAT
cmpa #ON
beq DIRO_S
ldaa #NORTE
staa RUMBO

FIN_ND4A nop
rts

DIRO_S ldaa #SUR
staa RUMBO
bra FIN_ND4A

* ************************************************************
* RESET
* ************************************************************
* Proposito: Configura los registros del TIMER para el
* retroceso
```

Codigo asembler para el M68HC11A8EVB.

```
* ************************************************************
RESET

jsr Calc

ldx #REGS

* Grab the value of the TCNT register tal que sea < a 14
loop ldd TCNT,x
cpd #14
bhs loop

std TOC1,x

* Set TOC2 to compare DELAY cycles later
addd DELAY
std TOC2,x

* Reset output compare flags
ldaa #OC1F|OC2F
oraa TFLG1,x
staa TFLG1,x

* Initialize the data register
ldaa #BITS
oraa OC1D,x
staa OC1D,x

* Initialize the mask register
ldaa #ALLBITS
oraa OC1M,x
staa OC1M,x

* Set up OC2 to reset the bit
ldaa #OC2R
oraa TCTL1,x
staa TCTL1,x



rts

* ------------------------------------------------------------------------------
* Nombre de la Subrutina: Calc
*
* Parametros de entrada: Ciclo ( 2 bytes) - Representa el ciclo util,
* su rango de variacion es de 0 a 100
*
* Parametros que devuelve: DELAY (2 bytes) - representa la cantidad de ciclos
* de reloj que durara el ciclo util
*
* Esta sybrutina se encarga de multiplicar la cte. UnPorc por el contenido
* de la direcc. dada por Ciclo. El resultado lo guarda en la direcc. DELAY
* ------------------------------------------------------------------------------
* DELAY <-- UnPorc * Ciclo



Calc

ldx #UnPorc
```

Codigo asembler para el M68HC11A8EVB.

```
stx Cuant
ldd #0
ldy Ciclo

M cpy #0
beq FinM
addd Cuant
dey
bra M

FinM std DELAY
rts
*****************************************************************
* CLR_SENS
*****************************************************************
* Proposito: Restablece los valores de S_FRO , S_LAT y
* CONTACT en el caso de que fueron afectados por los sensores
*
*****************************************************************
CLR_SENS

*
* IF S_FRO <> OFF THEN
* S_FRO <-- OFF
*

ldaa #ON
cmpa S_FRO
bne FS1CS

* Se restablece el sensor frontal
ldaa #OFF
staa S_FRO



*
* IF S_LAT <> OFF THEN
* S_LAT <-- OFF
*

FS1CS ldaa #ON
cmpa S_LAT
bne FS2CS

* Se restablece el sensor lateral
ldaa #OFF
staa S_LAT
FS2CS nop
*
* IF CONTACT <> OFF THEN
* CONTACT <-- OFF
*
ldaa #ON
cmpa CONTACT
bne FS3CS
* Se restablece CONTACT
ldaa #OFF
staa CONTACT
```

Codigo asembler para el M68HC11A8EVB.

```
FS3CS nop

rts
* ************************************************************
* CTRL_NAV
* ************************************************************
* Proposito: Realiza el desplazamiento en contacto perma-
* -nente con con la pared, evalua la distancia recorrida a
* partir del ultimo acople y verifica la condicion fin de
* recorrido
* ************************************************************
CTRL_NAV

* Se utiliza un nuevo valor para el ciclo util
ldy #CICLU3
sty Ciclo

* Variable que produce un retardo para el antirrebote
ldaa #NTIMESS
staa Demora

*
* LOOP21
* MIENTRAS (!(S_FRO or S_LAT or FINAL))
* NAVEGAR();
*

LOOP21 tst S_FRO
bne END_L21
tst S_LAT
bne END_L21
tst FINAL
bne END_L21

jsr NAVEGAR

bra LOOP21

END_L21 rts

* ------------------------------------------------------------
* NAVEGAR
* ------------------------------------------------------------
* Proposito: Avanza, cuenta la distancia recorrida y
* evalua la condicion de fin
* ------------------------------------------------------------
NAVEGAR

jsr AVANZAR
jsr MEDIDA
jsr EVAL

rts
*------------------------------------------------------------
* MEDIDA
*------------------------------------------------------------
* Proposito: Mide la distancia recorrida
*------------------------------------------------------------
MEDIDA
```

Codigo asembler para el M68HC11A8EVB.

```
ldx #REGS
* IF Demora = 0 THEN
ldaa Demora
cmpa #0
bne SNREB

* IF PAVIEJO <> PACTN
ldaa PACNT,x
cmpa PAVIEJO
beq FSREB

* DIST <-- DIST +1
ldy DISTANCIA
ldab #ON
aby
sty DISTANCIA
* CONT <-- CONT +1
ldy CONT
ldab #ON
aby
sty CONT
* PAVIEJO <-- PACNT
ldaa PACNT,x
staa PAVIEJO
bra FSREB

* Demora <-- NTIMES
ldaa #NTIMES
staa Demora
bra FSREB
* ELSE
SNREB dec Demora
FSREB rts



* ------------------------------------------------------------
* EVAL
* ------------------------------------------------------------
* Proposito: Calcula el desplazamiento neto, si el des-
* -plazamiento neto es cero hemos llegado al punto de inicio.
* ------------------------------------------------------------
EVAL
*
* IF RUMBO = NORTE THEN
* L_NORTE <-- L_NORTE +1
*
ldaa #NORTE
cmpa RUMBO
bne FSEVL1

ldy L_NORTE Incrementa L_NORTE
iny
sty L_NORTE
bra FSEVL4
*
* IF RUMBO = SUR THEN
* L_SUR <-- L_SUR +1
```

Codigo asembler para el M68HC11A8EVB.

```
*
FSEVL1 ldaa #SUR
cmpa RUMBO
bne FSEVL2

ldy L_SUR Incrementa L_SUR
iny
sty L_SUR
bra FSEVL4
*
* IF RUMBO = ESTE THEN
* L_ESTE <-- L_ESTE +1
*
FSEVL2 ldaa #ESTE
cmpa RUMBO
bne FSEVL3

ldy L_ESTE Incrementa L_ESTE
iny
sty L_ESTE
bra FSEVL4
*
* IF RUMBO = OESTE THEN
* L_OESTE <-- L_OESTE +1
*
FSEVL3 ldaa #OESTE
cmpa RUMBO
bne FSEVL4

ldy L_OESTE Incrementa L_OESTE
iny
sty L_OESTE
FSEVL4 nop
*
* IF L_NORTE > L_SUR THEN
* MAYOR_A <-- L_NORTE
* MENOR_A <-- L_SUR
*
ldy L_SUR
ldx L_NORTE
cpx L_SUR
blo ELSEIF1

stx MAYOR_A
sty MENOR_A
bra ENDIF1
*
* ELSE
* MENOR_A <-- L_NORTE
* MAYOR_A <-- L_SUR
*
ELSEIF1 stx MENOR_A
sty MAYOR_A
*
* IF L_ESTE > L_OESTE THEN
* MAYOR_A <-- L_ESTE
* MENOR_A <-- L_OESTE
*
```

Codigo asembler para el M68HC11A8EVB.

```
ENDIF1 ldy L_OESTE
ldx L_ESTE
cpx L_OESTE
blo ELSEIF2

stx MAYOR_B
sty MENOR_B
bra ENDIF2
*
* ELSE
* MENOR_B <-- L_ESTE
* MAYOR_B <-- L_OESTE
*

ELSEIF2 stx MENOR_B
sty MAYOR_B

ENDIF2 nop

* IF LADOS > 2 THEN
ldaa LADOS
cmpa #MIN
blo ENDIF3

* IF PRECISION+MENOR_A > MAYOR_A
ldab #PRECISION
ldx MENOR_A
abx
cpx MAYOR_A
blo ENDIF3

* IF PRECISION+MENOR_B > MAYOR_B
ldx MENOR_B
abx
cpx MAYOR_B
blo ENDIF3
* FINAL <-- ON
ldaa #ON
staa FINAL

ENDIF3 nop



rts

* ************************************************************
* GUARDAR
* ************************************************************
* Proposito: Guardar en la direccion de memoria indicada
* por el puntero la distancia recorrida y el rumbo.
* ************************************************************
GUARDAR
ldy CONT
ldx PUNTERO
sty 0,x
ldaa RUMBO
staa 2,x
```

Codigo asembler para el M68HC11A8EVB.

```
ldx PUNTERO
ldab #3
abx
stx PUNTERO

ldx #CLEAR
stx CONT

inc LADOS
rts

*
*******************************************************************************
* COM
* MODULO DE PRUEBA DE COMUNICACION
*
*******************************************************************************
* Autor: Ariel Guerrero / Luis Jara
* Fecha: Sab 22/jul/2000
*
*******************************************************************************
* Fecha Version Comentario
* 22/jul 1 Imprime un caracter con OUTSTR
* 22/jul 2 Se reimplementa OUTSTR con OUTPUT()
* 22/jul 3 Se utiliza input
* 27/jul 4 Implementacion p/ HC11
* 28/jul 5 Mejoras en el stuffing
* 29/jul 6 Pruebas de com
* 01/ago 7 Se transmite los caracteres en un solo bloque,
* rellenado de caracteres si dato= caracter_ctrl
*
*******************************************************************************
* MAPA DE MEMORIA A TX

* |-------------------|
* | Perimetro | $DA00
* |-------------------|
* | Num. de Lados | $DA02
* |-------------------|
* | RECORD(i) | $DA03
* |-------------------|
* | RECORD(i+1) | $DA06
* |-------------------|
* |
* v

COM jsr ESPERAR

jsr DELAYc

* Formato del bloque de datos 1(BD1)
* ------------------------
* | BD | ETX |
* ------------------------
* BD : Bloque de datos a transmitir
* ETX : Terminacion del bloque de datos
* Calculo del Numero de bytes a transmitir
* NUM = TRES + NUML*TRES

ldaa NUML
inca
```

Codigo asembler para el M68HC11A8EVB.

```
ldab #TRES
mul D <- TRES * (1 + NUML)
* ldd #22
std NUM

* LP1:
* Mientras( NUM <> 0 )
* Transmitir_BD;

ldy #uDAT

LP1 ldd NUM
* subd #1
subd #0

beq FIN_LP1
std NUM

ldaa 0,Y A <- M{x}

* SI(A==ETX) RELLENAR1;
cmpa #ETX
beq RELLENAR1
NXT1 jsr OUTPUT
iny
ldd NUM
subd #1
std NUM

bra LP1
* FIN de TX de comunicacion
FIN_LP1 ldaa #ETX
jsr OUTPUT

rts
RELLENAR1 jsr STUFFING
bra NXT1
* ---------------------------------------------------------------
* STUFFING
* ---------------------------------------------------------------
* Proposito: Rellenado de caracteres de control
* Fecha: 27/jul
* Autor: AG/LJ
* ---------------------------------------------------------------
STUFFING jsr OUTPUT
rts

* ---------------------------------------------------------------
* ESPERAR
* ---------------------------------------------------------------
* Proposito: Espera cualquier caracter para sincronizar
* Fecha: 27/jul
* Autor: AG/LJ
* ---------------------------------------------------------------
ESPERAR jsr INPUT
* Mientras(No haya dato) esperar_para_sincronizar;
cmpa #NUL
beq ESPERAR
rts
```

Codigo asembler para el M68HC11A8EVB.

```
* ---------------------------------------------------------
* DELAYc
* ---------------------------------------------------------
* Proposito: provee un retardo para sincronizacion
* Fecha: 29/jul
* Autor: AG/LJ
* ---------------------------------------------------------
DELAYc clr CNTcom * Inicializar contador
ldaa #255
staa CNTcom
delay1c nop
nop
nop
nop
nop
dec CNTcom
tst CNTcom
bne delay1c
rts


*************************************************************
*************************************************************
*************************************************************
*************************************************************
rs_IC1
* Se establece el sensor frontal a ON
ldaa #ON
staa S_FRO

* Se setea la bandera
ldx #REGS
ldaa #IC1F
oraa TFLG1,x
staa TFLG1,x
rti
*************************************************************
rs_IC2

* Se restablece el sensor frontal a ON
ldaa #ON
staa S_LAT

* Se setea la bandera
ldx #REGS
ldaa #IC2F
oraa TFLG1,x
staa TFLG1,x

rti
*************************************************************
rs_IC3

* Se restablece la variable CONTACT a ON
ldaa #ON
staa CONTACT

* Se setea la bandera
```

Codigo asembler para el M68HC11A8EVB.

```
ldx #REGS
ldaa #IC3F
oraa TFLG1,x
staa TFLG1,x

rti
*****************************************************************
*****************************************************************
*****************************************************************
* Rutina de servicio de interrupcion del IC1
org IC1V
jmp rs_IC1

* Rutina de servicio de interrupcion del IC2
org IC2V
jmp rs_IC2

* Rutina de servicio de interrupcion del IC3
org IC3V
jmp rs_IC3
*****************************************************************
*****************************************************************
*****************************************************************
```