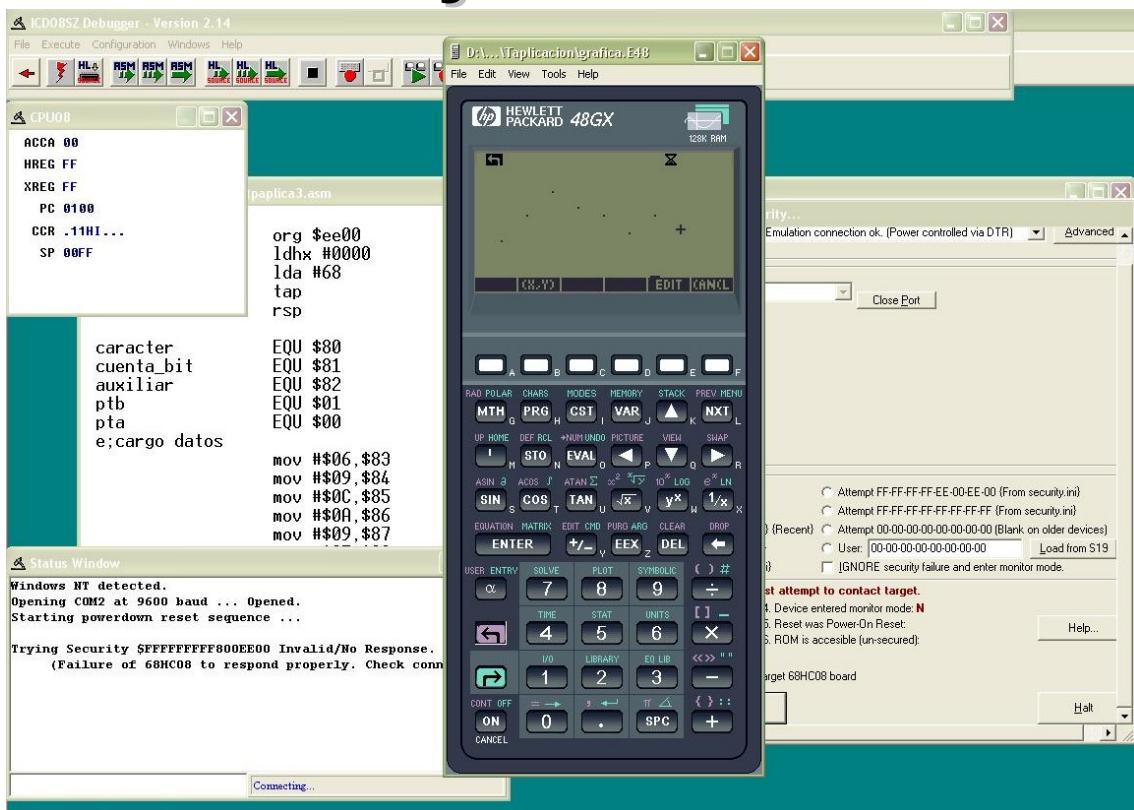


Circuitos digitales y Microprocesadores

Trabajo de Aplicación: Graficador de señales de baja frecuencia



Gruppo
Eduardo Kunysz
Yuichi Inomata
Alfredo Aschkar

Indice.....	1
Abstract.....	3
Distribución de los capítulos:	
Capítulo 1. Introducción	4
Capítulo 2. Investigación.....	7
Capítulo 3. Descripción Saturn	21
Capítulo 4. Programas	32
Capítulo 5. Hardware y puesta en marcha	68
Capítulo 6. Manual del usuario	71
Capítulo 7. Mejoras	77
Capítulo 8. Conclusiones	78
Apéndice	79

Capítulo 1. Introducción

Prestaciones

Diseño y realización

Planificación

Capítulo 2. Investigación

Funciones Osciloscopios

Modos disparo

Modos muestreo

Conversión / Transferencia

Errores

Comunicación serie

Generalidades

Protocolos HP

Operación UART

Conectores

Características eléctricas

Hardware

Microcontrolador

Dispositivo de representación

Comunicación

Monitoreo y menús

Herramientas de programación (CAD)

Win IDE

Debug4x

Capítulo 3. Descripción Saturno

Saturno registros

Introducción

Programming model

Inversión de nibble

Set de instrucciones

Objetos

Comunicación serie

Graficos

Generalidades

Capitulo 4. Programas

Desarrollo transmisión serie

Usando modulo TIM

Diagramas

Desarrollo serie HP (User)

Programas Saturn Assembler

Generico recepcion

Desarrollo graficos

Ubicar pixel

Registro flotante

Desplazamiento lateral

Programas HC908 Assembler

LCD 3 lineas de datos

Menu principal

Captura

Continua

Control

Teclado

Capitulo 5. Hardware y puesta en marcha

Introducción

Esquematico

Placa final

Experienciamon

Calculo errores

Capitulo 6. Manual Usuario

Información sobre el producto

Requisitos

Conectores y controles

Especificaciones

Modos de funcionamiento

Descripción del software

Capítulo 7. Mejoras

Etapa de entrada

Etapa de alimentación

Programas

Capítulo 8. Conclusiones

Apéndice

Abstract

Este documento es un informe detallado sobre el desarrollo del Trabajo de Aplicación de la materia Circuitos Digitales y Microprocesadores, de los alumnos: Yuichi Inomata, Alfredo Aschkar y Eduardo Kunysz

El objetivo de este trabajo es el diseño y construcción del prototipo de un graficador de señales de baja frecuencia. Este será un dispositivo portátil, de bajo costo y funcionara en conjunto con una calculadora del tipo Hewlett Packard. Es decir que requerirá de una calculadora para su funcionamiento y a travez de esta se podrá analizar la información capturada por el equipo para luego poder ser graficada o almacenada. Con este trabajo se pretende explotar al máximo las prestaciones de los microcontroladores 69HC908QY4 de Motorola.

Introducción

Prestaciones básicas:

El graficador deberá tener todas características básicas para poder ser utilizado cómodamente, ya sea desde el punto de vista hardware, como desde el software implementado. La idea es construir un dispositivo de muy bajo costo, que cumpla con las necesidades básicas de un instrumento práctico, portátil y confiable. De fácil adaptación a diversos dispositivos de graficación como de almacenamiento.

Se pretende obtener la máxima frecuencia de muestreo posible y por lo menos dos modos de representación gráfica.

Las características que se desean obtener son:

- Ancho de banda cercano al KHz.
- Tensión de entrada entre 0 y 5V.
- Pequeño, portátil, adaptable y transportable.
- Facilidad en la utilización.

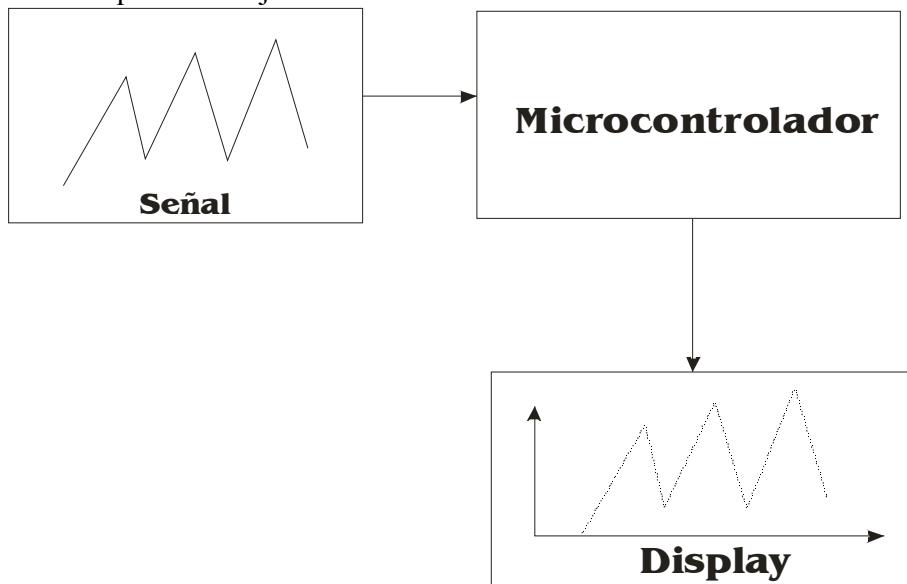
Diseño y realización:

Se trabaja principalmente sobre la idea de explotar al máximo las características de los microcontroladores HC908 de Motorola. Si bien gran parte del trabajo de procesamiento de señales se realizan sobre el dispositivo de representación gráfica, se trata de utilizar todos los módulos del microcontrolador.

Lo que se implementó a nivel hardware, entonces, está basado en un microcontrolador de Motorola, destacándose los siguientes bloques:

- Toma de datos y digitalización
- Entrada de control por matriz (teclado)
- Monitoreo y verificación desde LCD
- Módulo de transmisión de datos

La idea sobre la que se trabaja



Se trabaja sobre dos modos de graficación, que llamamos:

- Captura
- Continua

Uno de los modos de representación consiste en tomar datos durante un cierto tiempo, a intervalos regulados por el usuario.

El otro modo de representación consiste en muestrear en forma continua una señal e ir graficándola en tiempo real. Con la posibilidad de ajustar las frecuencias de muestreo y posición de la imagen.

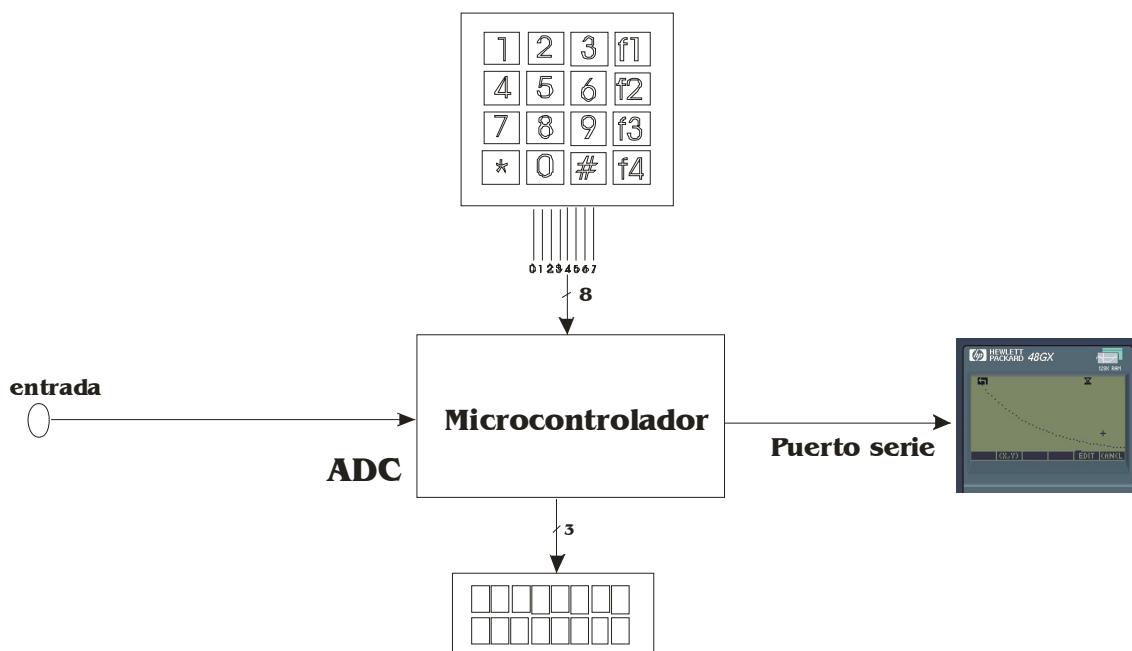
Para poder dar flexibilidad al usuario, se dispone de una matriz de ingreso de datos y un pequeño display donde se especificaran opciones, menues, etc.

El software que se implemento en el dispositivo Hewlett Packard trata de ser lo mas simple posible. Permitiendo dos formas de graficacion uno del modo captura y una forma por el modo continua.

Para el primer modo se trabajo sobre un soft que permite al usuario familiarizado con el sistema operativo de la calculadora, poder manipular la señal desde el propio modo PLOT que posee el dispositivo, de esta forma se puede calibrar a gusto el tamaño de pantalla. Si bien este modo es un poco mas lento, mejora bastante en la presicion de la grafica. El otro metodo de representación no es tan flexible, pero permite realizar graficas a mucha velocidad, bajandose tambien considerablemente la resolucion de la imagen.

El otro modo de graficacion (continua) es poco flexible, pero dada su naturaleza de grafico dinamico, fue posible la manipulación de datos en el microcontrolado, antes de ser enviados a la calculadora.

Diagrama de bloques – implementacion:



Si bien se ha logrado hacer que este prototipo funcione correctamente, hay ciertos puntos que aun no han sido solucionados. Cabe destacar que la etapa de entrada no posee ningun filtro, por lo tanto en las señales cercanas a las del nivel de tierra contienen bastante ruido. Tampoco posee ninguna proteccion, ni aislamiento alguna, de

modo que un sobrepico en la señal podría dañar seriamente el equipo. Tampoco se ha diseñado protección a la salida, pudiendo esto dañar el puerto de comunicaciones serie del dispositivo de representación. Se aclara que esto se debe a qué se ha logrado fabricar este prototipo en poco tiempo y es por ello que ciertas características deseables o incluso necesarias, quedan como pendientes para el futuro.

Planificación:

La primera etapa del proyecto se trabajó principalmente en cumplir ordenadamente las pautas preestablecidas en el pre-proyecto:

1. Utilizar el conversor A/D del microcontrolador 69HC908 para muestrear cierta señal con una dada frecuencia conocida. Se utilizará en primera medida la memoria RAM del dispositivo para almacenar los datos.
2. Desarrollar la interfaz de comunicación serie RS232 en el microcontrolador 68HC908 y establecer la comunicación con el microprocesador Saturn Yorke de la calculadora.
3. Enviar los datos a la memoria RAM de la calculadora y desarrollar un programa en *user RPL* (alto nivel) para graficar la información.
4. Optimizar y mejorar en velocidad la transmisión y comunicación serie configurando en *system RPL* (bajo nivel) el microprocesador Saturn.
5. Graficar en tiempo real la información enviada por el microprocesador HC08 configurando y diseñando la interfaz gráfica en *system RPL*. De esta forma el límite de frecuencia lo impondría el refresco de pantalla. (Así se podría obtener un pseudo Osciloscopio de baja frecuencia).

El primer objetivo se logró bajo simulación utilizando el software Win-Ide In Circuit Simulator

Luego se trabajó sobre la interfaz de comunicación, siendo de los primeros objetivos el más difícil de cumplir, dada la carencia de un módulo de transferencia serie SCI en el microcontrolador 68HC908

Para esta segunda parte se seleccionaron emuladores y soft de transferencia genérico de Windows como Hyper Terminal.

El emulador seleccionado para este trabajo fue el EMU48

Las características principales por las que lo seleccionamos fueron:

- Semejanza con la calculadora real.
- Posibilidad de acceder a puertos e/s ya sea serie como infra red
- Facilidad para escribir programas y compilarlos.
- Simulación en varios niveles de tiempo:
 - Tiempo real de frecuencia de PC
 - Simulación de velocidad de oscilador de calculadora.
- Poder cargar y grabar programas directamente a la PC.

Investigacion

En esta etapa se trabajo sobre toda la información relacionada al proyecto que iba a ser necesario recopilar y analizar. Para luego poder ver su implementacion.

Se trabajo paralelamente sobre dos etapas:

Etapa Hardware.

Etapa Software.

La parte de hardware trato sobre la selección de circuitos y dispositivos fisicos sobre los que se iba a trabajar: selección de componenentes en base a la arquitectura que estabamos utilizando y expansión de posibilidades con logica digital extra como en nuestro caso la utilización de registro de desplazamiento para poder manejar un display con solo tres lineas de datos.

En la parte de software se investigo extensamente los modos de comunicación serie, protocolos, y modos de implementacion para realiza esta comunicación entre dos microcprocesadores de tecnologías diferentes. Tambien se estudio de forma profunda los modos de programación de ambos micropocesadores en lenguaje de ensamblador.

Por otra parte hicimos una pequeña investigación sobre modos de operación de osciloskopios genericos digitales para tener puntos de referencia.

Funciones de osciloscopios

Modos de disparo:

Los osciloscopios tradicionales cuentan con tres modos de disparo (trigger):

- Modo de disparo basico
- Modo de disparo por deteccion de transitorios
- Disparo externo

Nosotros trabajamos sobre la primer opcion de disparo, el modo de disparo basico. Que en general consiste en mantener la imagen fija que se muestra en pantalla. Para ello implementamos un mecanismo de envio de alto de un pin para poder sincronizar y medir captura de elementos paivos como por ejemplo un capacitor.

Modos de muestreo:

-Tiempo real: es la unica forma de capturar señales transitorias. Debe tener como caracteristica que la frecuencia maxima de muestreo debe ser dos veces mas rapida que la de la señal respondiendo al teorema del muestreo:

$$F_m > 2F_s$$

Este método de muestreo es el mas simple de todos y permite digitalizar señales no periódicas y transitorias. Cada muestra y el tiempo en que fue tomada tiene una correspondencia directa con su equivalente en tiempo real. A mayor tasa de muestreo en comparación al ancho de banda de la señal, se obtiene una mayor definición en el resultado.

Conversión y tranferencia:

Para el primer modo de funcionamiento, (captura). El proceso de conversión y toma de datos se realiza integralmente dentro del microcontrolador. Almacenándose el la

memoria RAM del dispositivo. Una vez completada la captura se procede a la transmisión de datos.

El límite de velocidad de proceso lo da el conversor ADC y el bus del microprocesador Ventaja:

- Se puede muestrear a velocidades más altas y por lo tanto se pueden tomar más datos.
- Se pueden utilizar velocidades de transferencia bajas y se puede utilizar el buffer del dispositivo de recepción facilitando y agilizando el software en la calculadora.
- En la calculadora al no requerirse grandes velocidades de proceso se puede trabajar sobre el modo de programación USER RPL que es mucho más sencillo, más seguro y más fácil de modificar para el usuario.

Desventajas:

- Solo se pueden capturar un cierto rango de tiempo y capturas, estando limitados por la capacidad de almacenamiento del dispositivo.

Para el segundo modo de funcionamiento: Continua

Los datos son enviados en forma ininterrumpida desde el microcontrolador a la calculadora, estando limitados en este caso por el método de transferencia de datos.

Ventajas:

- Se pueden graficar señales continuas transitorias.
- El software de la calculadora es muy rápido bastante estable

Desventaja:

- Límite de velocidad muy bajo, se da todo en el puerto de comunicación.
- Programa en la calculadora muy difícil, hubo que trabajar íntegramente en assembler para poder lograr velocidades adecuadas.

Errores:

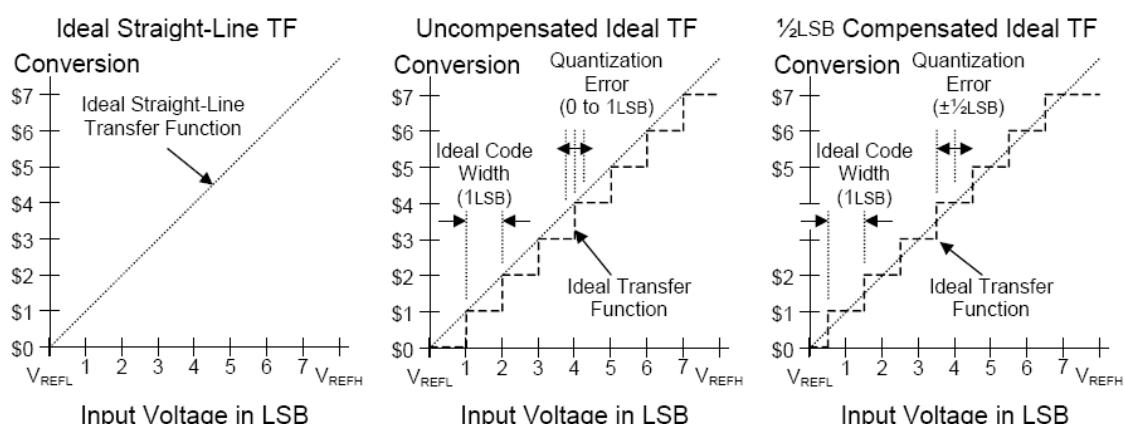
En la digitalización de datos se producen ciertos errores, a continuación se va a hacer una breve descripción de ellos:

Error de cuantificación (EQ):

El camino de la función de transferencia ideal se divide en pasos dependiendo del método de cuantificación que usa el ADC. Existen dos métodos:

1 Cuantificación sin compensación: el error es de 0LSB hasta 1LSB

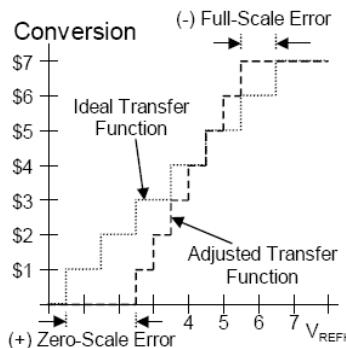
2 Compensación de 1/2 LSB :el error en este caso es de +/-1/2LSB



Error de 0 y fondo de escala:

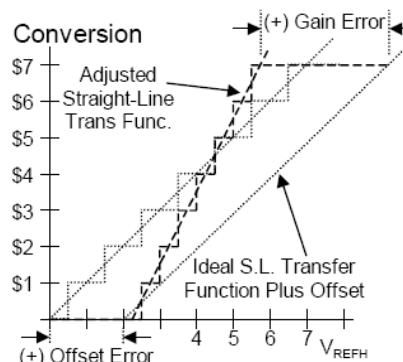
Error de escala en cero: es la diferencia entre la primer transición de tensión y la primer transición ideal (EZS)

Error a fondo de escala: es la diferencia entre la ultima transición de tensión y la transición ideal de tensión.

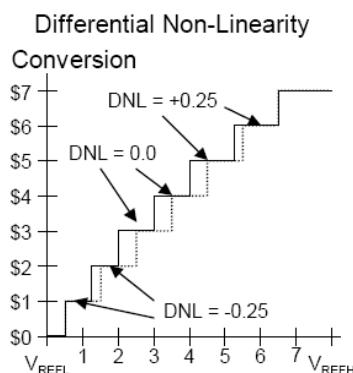


Error de Offset: es la tensión de salida de un ADC con un código de entrada cero, o es el valor de tensión de entrada de un ADC requerido para que la salida sea exactamente el código 0

Error de Ganancia: es el apartamiento respecto del valor de diseño, de la tensión de salida de un ADC para un código de entrada dado, usualmente el de fondo de escala.



Error de linealidad: la no linealidad describe el apartamiento con respecto a una curva de transferencia lineal tanto para un ADC como para un DAC. En un conversor ideal, incrementos iguales de la entrada producen incrementos iguales a la salida.



Error de entrada diferencial, error de ruido:

Las causas de este error se pueden clasificar en:

Acople de referencia: es el ruido que se inyecta en la linea de referencia, esto causa que se acople a otra referencia.

Acople diferencial: se define como el ruido que se inyecta en la entrada o en la referencia, produciendo un acoplamiento con otras.

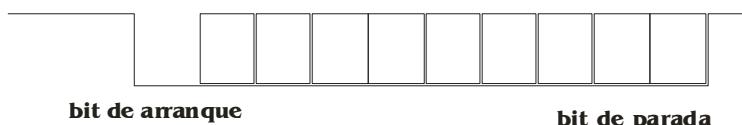
Comunicación serie UART:

La comunicación serie que se utilizo para el proyecto es la del tipo simplex asincronica. Consiste en un dispositivo que envia a otro sin recibir señal de retorno donde no se requiere de un reloj para manejar el sincronismo. Este metodo se utilizo por un tema de velocidad y ahorro de circuitos. Dado que para llevar a cabo una transmisión por este procedimiento se requieren solo dos líneas de datos, una de transmisión llamada TX y una línea de nivel de tierra llamada GND

Para que se pueda establecer una comunicación asincronica es necesario que la linea de comunicación este por defecto en alto, entonces cuando llegue el primer paquete de datos, en su cabeza se detecte el bit de Start. Luego de que la señal se mantuvo un tiempo equivalente a

$$\text{Velocidad de bit} = \frac{1}{\text{Baudrate}} [\text{s}]$$

Se envia el paquete de datos, cuando se recibe el ultimo dato, la señal debe permanecer en alto por lo menos medio tiempo indicando el bit de parada.



Para poder establecer una correcta comunicación con la calculadora se estudio en detalle cuales son las posibilidades que ofrece el dispositivo HP para la comunicación:

Protocolos de comunicación HP:

Por defecto HP48 utiliza el protocolo de transferencia KERMIT para transferir los datos y corregir los errores de transmisión. En el modelo GX se incorpora el protocolo XMODEM que no realiza verificación de errores de transmisión. Si bien este protocolo suele realizar verificación por el método de redundancia cíclica (CRC), en HP se ha ignorado dicho proceso, consiguiendo un mejor rendimiento.

Asimismo, HP deja abierta la posibilidad de enviar y recibir datos y comandos con dispositivos serie que *no utilicen* el protocolo Kermit. Esto se hace utilizando los comandos (user RPL) de E/S serie generales.

Serial UART operation HP:

Operación de recepción:

Para trabajar con las operaciones de E/S serie generales, el puerto debe estar abierto, esto se consigue ingresando el comando OPENIO.

El sincronismo de todos los bits es relativo al bit de comienzo (extremo). La resolución del sincronismo es 16 veces la del tiempo seteado en baudios. El dato en el pin RX es empujado (shifted) al registro de desplazamiento (shift register) y el tiempo se centra en la mitad de cada bit. Cuando por el pin RX recibe el bit de comienzo, se inicia el clock. Si el pin RX conserva la marca durante un tiempo suficiente (mayor a la mitad del bit, por lo menos), el bit de comienzo es considerado valido y el registro de recepción (shift register) es puesto en enable. Si el pin RX llegare a retornar a la posición anterior antes de cumplido ese tiempo, el reloj de recepción se detiene y la transferencia es abortada. Cuando el registro de desplazamiento hizo un shifted del bit de inicio, ocho bits de datos y uno de stop son recibidos. Luego los 8 bits son transferidos al receive buffer register (RBR), se setean todas las banderas del buffer RBF (receive buffer flag) y se reinicializa el registro de desplazamiento. Si las banderas RBF están seteadas el receive error flag (RER) se setea para indicar overrun. Si el bit de stop no esta en "1" (condición de marca), el RER flag se setea para indicar error y la recepción retorna al estado de espera por un bit valido.

El software de HP48 lee el byte out del RBR y ubica el byte en el buffer de ingreso de 255 bytes. Todas las paridades son leídas por el software una vez ingresado en este buffer.

Para recibir el dato, se debe ejecutar el comando SRECV. El funcionamiento de este comando es el siguiente:

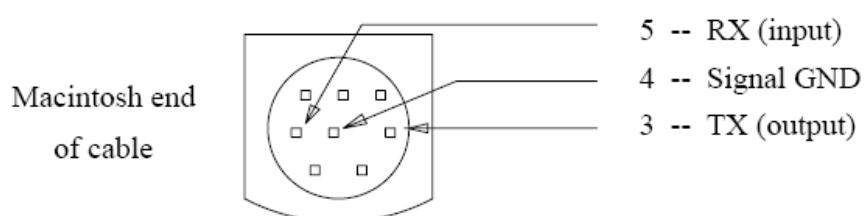
Se debe ingresar primero el número de caracteres que se desea transferir, luego se ejecuta el comando. En una transferencia correcta se devuelven los caracteres al nivel 2 como cadena (char, códigos propios de HP) y se devuelve un 1 al nivel 1. En caso de una transferencia incorrecta, se devuelve una cadena vacía al nivel 2 y un 0 al nivel 1. Las razones por las que se puede dar una transferencia incorrecta son error de paridad, error de comunicación o error de desbordamiento, o se reciben menos caracteres de los especificados antes de que expire el tiempo de espera que por defecto es de 10 segundos. En caso de error se puede ejecutar el comando ERRM para devolver el mensaje de error.

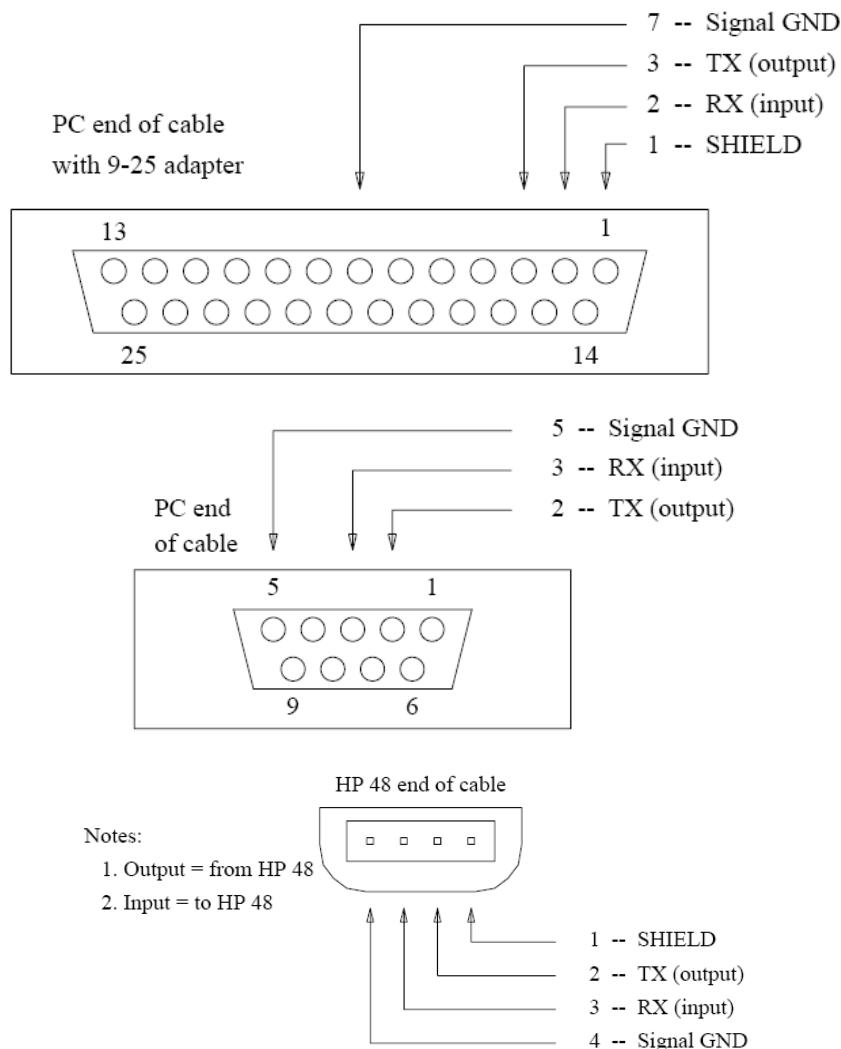
Los tiempos de espera se pueden modificar con el comando STIME.

También es muy útil el comando BUflen que devuelve el número de caracteres del buffer de entrada al nivel 2 y el estado de errores al nivel 2. La maxima velocidad que soportan estos comandos por user RPL es de 9600 baudios

Observación: Estos comandos no verifican la integridad de los datos

Tipos de conectores que se utilizan en el dispositivo HP:





Las especificaciones electricas del puerto serie son:

Signal	Description	Min	Typ	Max	Units
TX	Output voltage swing. Load: 3 kΩ min, 500 pF max	+/-3.0	3.5		
	Bit width tolerance			2.5	%
RX	Input + operating range	1.0		15.0	V
	Input – operating range	-15.0		.03	V
	Input impedance	5.0		7.0	kΩ
	Bit width tolerance			2.5	%
All	Absolute maximum voltage			+/-25	V

Hardware:

Microcontrolador

El hardware que se utilizo se basa principalmente en un microcontrolador 69HC908QY4 de Motorola. Es un microprocesador de 8bits para diseño de sistemas

embebidos de arquitectura Von Newman. Una de las principales características de este microcontrolador son los bajos costos y su alto desempeño. Utiliza un microprocesador denominado CPU08. Otra de las características importantes de este micro es que utiliza memoria flash, cuenta con conversores analógico digitales, módulos de temporización. La placa de desarrollo utilizada es la EvalKit 08

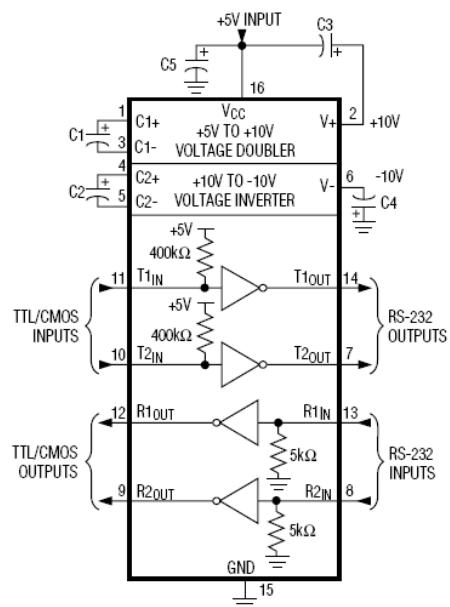
Dispositivo de representación

El otro microprocesador que se utilizó fue el Saturn Yorkee desarrollado por Hewlett Packard en los años 80 para calculadoras programables y fabricado por la empresa NEC. Es un microprocesador basado en una arquitectura de Nibbles con una unidad de núcleo de 4 bits. Los detalles de este microprocesador se detallan más adelante.



Comunicación

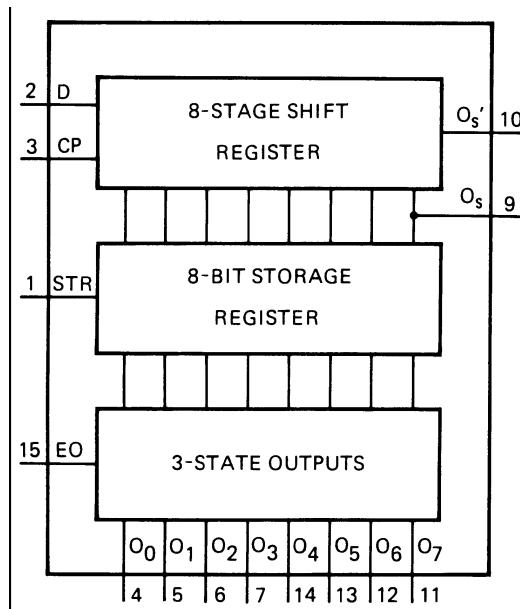
Para interconectar los dos microprocesadores se hizo uso de un chip de estabilización de tensiones para puertos de transmisión serie: MAX232



Monitoreo y menús

Para poder utilizar con mayor eficiencia los puertos de el microprocesador HC908 se utilizó un registro de desplazamientos SIPO: modelo CD4094BE

Con esto se pudo implementar un sistema de control de LCD con 3 líneas de datos



Herramientas de programación:

Para poder escribir los programas en el microprocesador HC908 de Motorola se utilizo el paquete Win Ide que provee la empresa. Esta herramienta tambien se utilizo para simular, y grabar el micro con el evalkit.

Para programar el microprocesador Saturn se utilizo el paquete Debug4x creado por William Graves para poder programar en los diversos lenguajes que soportan toda la linea de microprocesadores Saturn.

Win Ide

1. Comunicar a la PC con la placa para desarrollos

Primero se configuro la placa en modo MONITOR 1 IRQ=Vhigh con la configuración de jumpers de la siguiente manera:

JP2 1-2
 JP3 1-2
 JP4 1-2
 JP5 1-2
 JP6 1-2
 JP7 1-2
 JP8 1-2

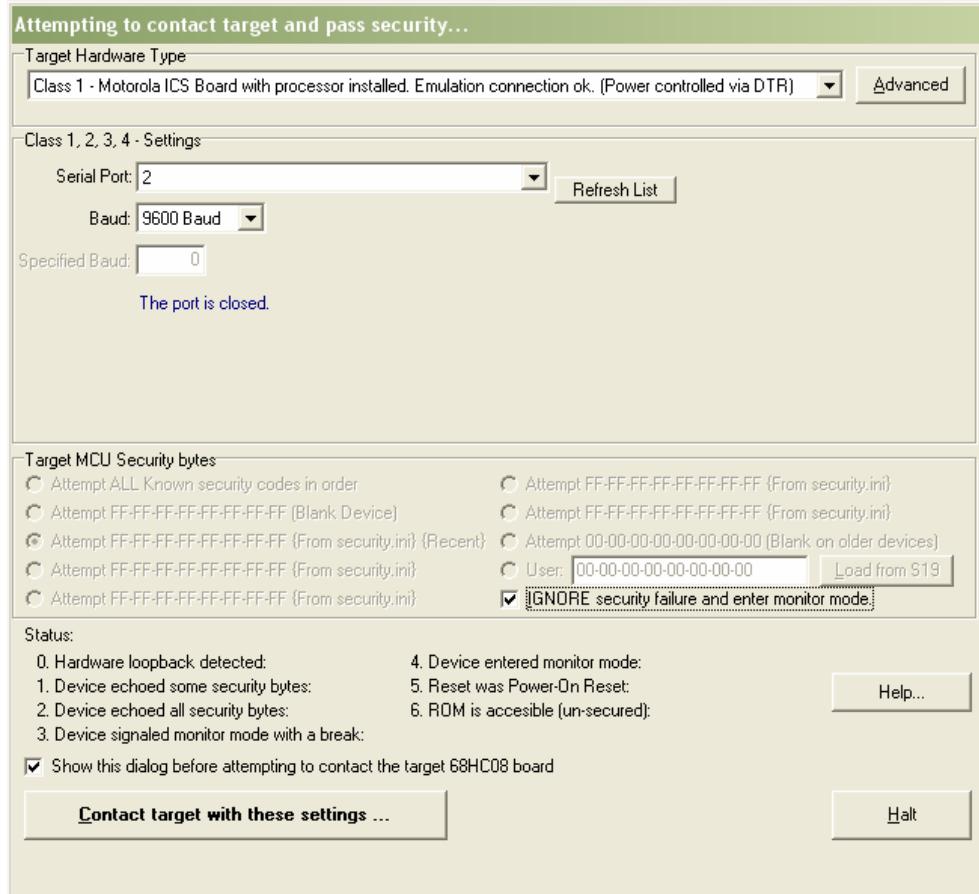
El jumper JP1 se lo configuro en posición 1-2 ->VDD=5Vdc

Se conectaron los conectores RS232 al puerto serie de la PC y la placa respectivamente. Y la alimentación 220. Al ponerse la fuente en ON se encendio la luz roja indicando encendido.

Luego desde el soft WinIDE se cargo corrio el programa del 3er botón de la barra



que llama al archivo PROG08SZ.EXE
Allí se despliega la siguiente ventana que la configuramos como se muestra:



La opción ignore security la utilizamos para evitar que si el micro estuviese grabado tener problemas con la protección de lectura. Las opciones de arriba se cambiaron a Class 1 serial port 2 que era el puerto que estabamos utilizando.

Luego se marcó la opción CONTACT TARGET WITH THESE SETTING...

Aparece una ventana que pide que se ingrese el algoritmo, elegimos el 908_qy4.08p que es el que se utiliza para el HC908QY4

Se produjo un manejo automático del ciclo de alimentación y se encendió la luz amarilla de la tarjeta de simulación.

Luego se accedió a la siguiente ventana



Donde nos indica que estamos conectados con la placa.

2. Borrar cualquier programa que encuentre cargado en la memoria del microprocesador

Para borrar cualquier programa en el micro se utilizó el sexto botón con forma de goma de borrar



3. Grabar en su memoria

Con el programa WinIDE se cargo el programa se lo grabo con un nombre en el rigido de la PC, luego volviendo al PROGRAMMER (PROG08SZ) se utilizo el septimo boton con forma de disco para cargarlo

Luego se utilizo la opcion de LOAD IMAGE CONTAIN FLASH MEMORY DATA Y se verifico el contenido de la memoria entrando a la region de memoria grabada.

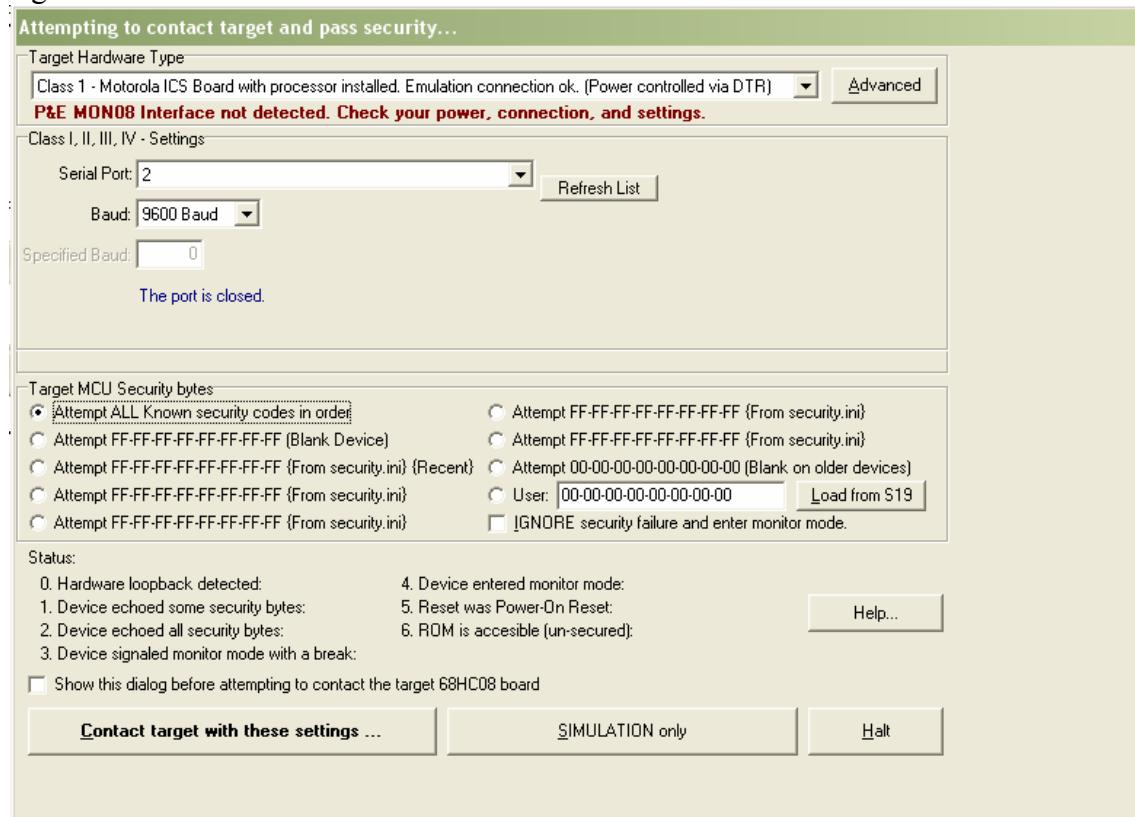
4. Ejecutar, en circuito el programa cargado.

Para ejecutar el programa cargado en circuito se utilizo el segundo boton del WinIDE



Debugger que ejecuta el programa ICS08qtqyZ.EXE

Accediendo a la ventana de configuración y contacto de la placa, la configuramos de la siguiente manera:



Luego se uso la opcion contact target with these settings ... y luego se corrió el programa cargado en la memoria FLASH del micro.

También como prueba ejecutamos el programa de la opción 4 In circuit Simulator donde se corrió el programa desde la memoria RAM de la PC.

Debug4X

Descripción del entorno de desarrollo Debug4x

Debug4x es un kit para el desarrollo de aplicaciones para calculadoras HP. La versión actual es la 2.1 y fue desarrollada por William G. Graves.

Las principales características de éste software son:

- Completo soporte para las calculadoras HP48 y HP49 (También se puede usar para desarrollo de aplicaciones para la HP38, HP39 y HP40 ajustando el objeto binario de salida)
- Automatiza la generación de directorios y librerías.
- Editor de formularios basados en IfMain y DoInputForm.
- Integración con EMU48 para depuración y pruebas de las aplicaciones generadas. Los archivos son automáticamente cargados e instalados en el emulador para los procesos de depuración.
- Depuración gráfica con puntos de interrupción, visualización de pila, variables locales, gráficos y otros parámetros de la calculadora.
- Editor con sintaxis en colores, autocompletación (ctrl.+SPC), diagrama de pila de comandos (ctrl.+J), identificación de rutinas (ctrl.+Shift+B) de código RPL y listados y otras características adicionales.
- Soporte completo para depuración en RPL y ensamblador SATURN.

Tipos de archivo y el directorio Include

Debug4x reconoce los siguientes tipos de archivo:

- Proyectos Debug4x (*.hpp) Los archivos de éste tipo contienen información sobre el tipo de proyecto (objeto, directorio o librería), emulador usado, archivos de código fuente e información sobre los archivos en edición.
- Código fuente (*.s) Contienen el código fuente en ASSEMBLER o RPL. Pueden ser módulos (Archivo con varias declaraciones NULLNAME o xNAME) o fuente incluido (sin estas declaraciones).
- Definiciones (*.h) Contienen definiciones usadas por los módulos del proyecto (llamadas de funciones entre módulos) y definiciones propias para el proyecto en particular.

Los archivos *.E48 y *.E49 son usados para la depuración del código. Cada proyecto puede tener su propia sesión de emulador con contenidos RAM diferentes.

El directorio Include

Es una carpeta ubicada dentro de la carpeta de Debug4x. Contiene los siguientes archivos:

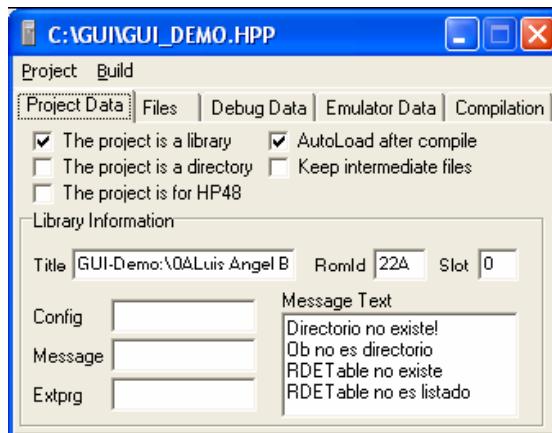
- Suprom48.a y Suprom49.a: Contiene los nombres y direcciones de los comandos usados por la HP48 y HP49 respectivamente.
- Suprom48.stk y Suprom49.stk: Nombres de comandos con su respectivo diagrama de pila. Estos archivos se pueden modificar para añadir nuevos diagramas de pila.

- InformBox.h e InformBox48.h: Definiciones usadas por el editor de formularios. Son incluidos automáticamente al crear un formulario.
 - Header.h: Definiciones estándar. Este archivo es incluido automáticamente al crear un nuevo formulario, pero entra en conflicto con las definiciones dadas en este manual. Cada proyecto puede tener su propio Header.h evitando así usar el archivo del directorio Include; típicamente el contenido de este archivo será INCLUDE MyProject_Header.h.
- Los archivos *.h que sean copiados en ésta carpeta pueden ser usados en cualquier proyecto y entran a ser parte de los comandos habituales de programación.

Crear un proyecto nuevo

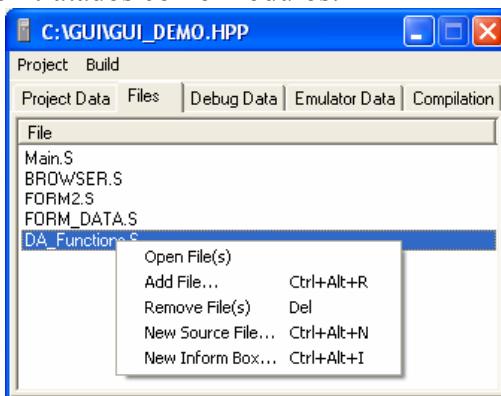
Ejecute Debug4x.exe y seleccione “New Project...” en el menú “Project”; establezca el nombre de archivo y guarde.

Habilitando la opción Autoload after compile la librería es automáticamente cargada e instalada en el emulador, habilitando el uso de puntos de interrupción.



Incluir módulos de código fuente

En la ventana del proyecto seleccione la ficha Files y usando el menú contextual seleccione New Source File... (Módulo de código) o New Inform Box... (Módulo de formulario). Los archivos mostrados en esta lista son módulos de código fuente. Cada módulo debe tener las archivos *.h necesarios para comunicarse entre ellos. Los formularios son tratados como módulos.



Escribir el código fuente

Debug4x proporciona un completo editor de código fuente. Pueden editarse módulos, archivos incluidos y en general, cualquier tipo de archivo.

La siguiente figura muestra el aspecto de la ventana de edición con parte del código del módulo Main.s. El contenido de este módulo se utilizará para explicar algunos detalles de los procesos de programación, depuración y compilación.

The screenshot shows the Debug4x editor window with the title bar "C:\TPAPLICANT\Aplicacion\grafico.S". The menu bar includes File, Edit, View, Compilation, Breakpoints, and Help. The tabs at the top are "scratch" and "grafico.S | Suprom48.a". The main pane displays assembly code:

```
1 RPL
2
3 :::
4 CLOSEUART
5 RECLAIMDISP
6 TURNMENUOFF
7 ABUFF
8
9 #E
10 #1
11 MAKEGROB
12
13
14 DROP
15 CODEM
16 SAVE
17 C=DAT1 A
● 18 D1=C
19 D1=D1+ 5 $SALTO PROLOGO
20 D1=D1+ 5 $SALTO SIZE
● 21 D1=D1+ 5 $SALTO NUMERO DE FILAS
22 D1=D1+ 5 $SALTO NUMERO DE COLUMNAS
23 C=D1 A
24 RO=C A
25 LC 22 %CONTADOR DE datos (34d=22h)
26 D=C B %CONTADOR DE FIN DE LINEA
27
28 *PRINCIPAL
```

The code includes several comments starting with '\$' and some assembly directives like '#E' and '#1'. Two lines are highlighted with red circles: line 18 ('D1=C') and line 21 ('D1=D1+ 5'). The status bar at the bottom shows "Line 18", "Start 0BFE60", "Stop 0BFFEA", "adr from line 0BFEAB ASM", "CR+LF (Windows)", and "SysRPL".

Declaraciones INCLUDE

La sentencia INCLUDE incluye el contenido del archivo como parte del módulo o fuente donde se haga el llamado. Generalmente se usa INCLUDE con archivos *.h y *.s

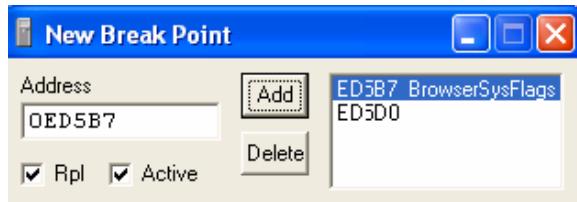
La sentencia INCLUDE también es usada para incluir código fuente dentro del módulo. La línea INCLUDE MsgBoxCicle.s incluye el contenido de MsgBoxCicle.s dentro del archivo actual. Esta estrategia se usa para hacer más claro el código o para incluir otros objetos (Arreglos, grobs, etc) de los cuales no necesitemos ver siempre el código fuente.

Compilación y depuración

En la ficha 'Emulator Data' de la ventana de proyecto seleccione los archivos del emulador a utilizar, habilite la opción 'Autoload after compile' en la ficha 'Project Data', abra el emulador (Ctrl+Alt+E) y a continuación compile el proyecto (F9); el resultado será cargado e instalado en el emulador.

Puntos de interrupción

Permiten detener la ejecución del programa en un punto específico para iniciar la depuración paso a paso. Los puntos de interrupción (Break Point) se establecen en el editor haciendo clic en la parte izquierda de la línea deseada pero solo en los módulos.

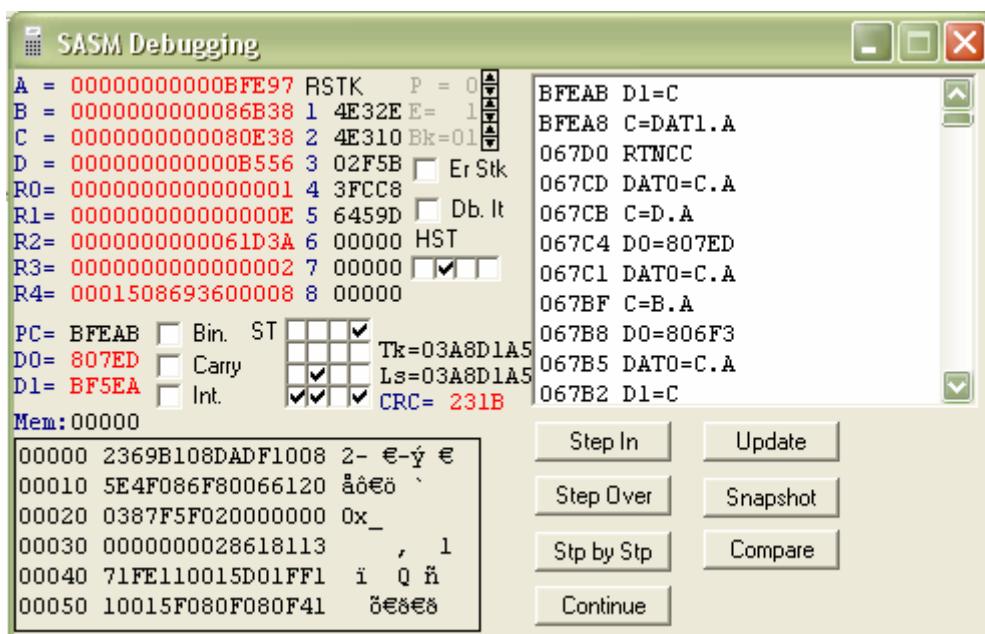


Al hacer doble clic sobre la dirección, se muestra el punto en el editor. Los puntos de interrupción solo están disponibles cuando el emulador está en ejecución. Si el emulador o el proyecto son cerrados, los puntos de interrupción desaparecerán.

Compilar para HP48 y HP49

En general, las calculadoras gráficas HP tienen bastante compatibilidad al nivel de código fuente. Debug4x permite cambiar fácilmente el objeto compilado para éstos dos modelos de calculadora.

Como se ve en el siguiente grafico en el modo de ensamblado se puede ver en detalle una simulación de un archivo compilado, donde se detallan todos los registros.



Descripción del microprocesador Saturn:

Saturn registros

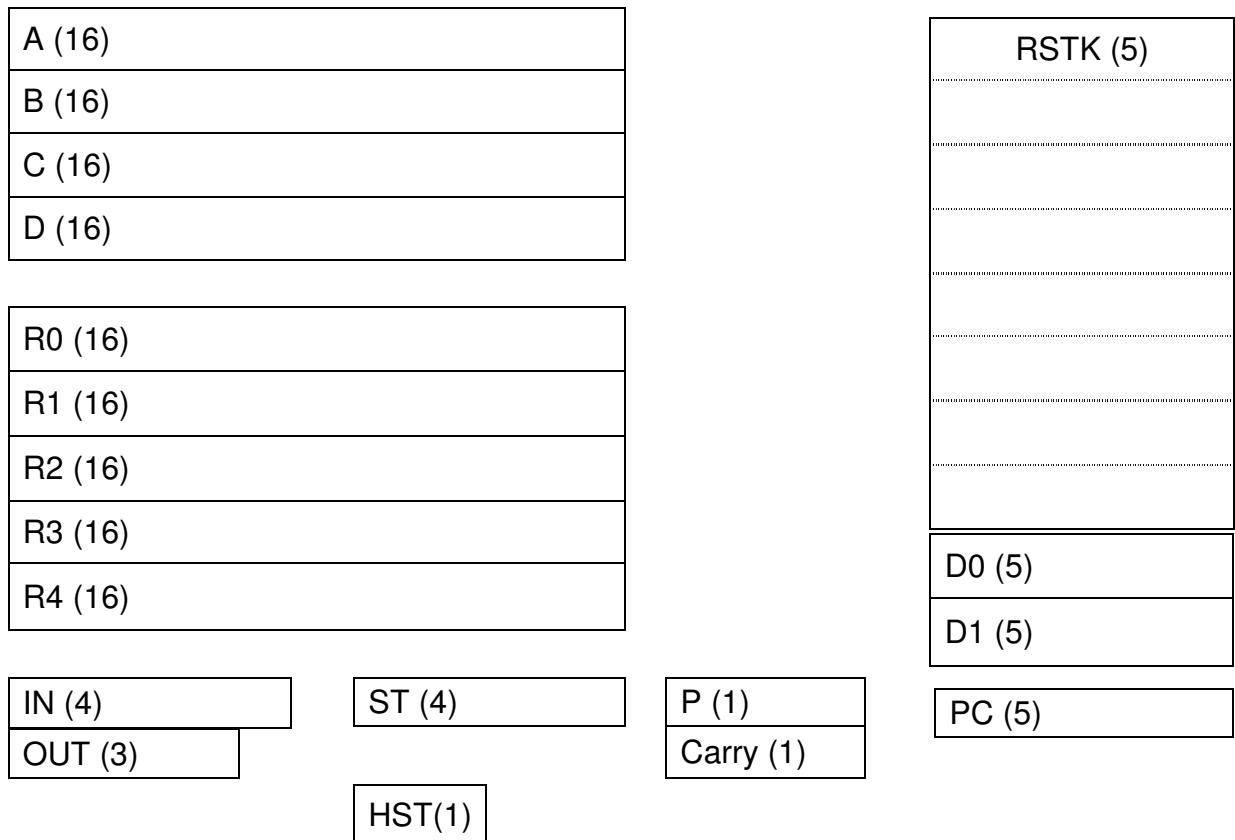
Introducción

El microprocesador Saturn basa su tecnología en una arquitectura de 4 bits (nibbles).

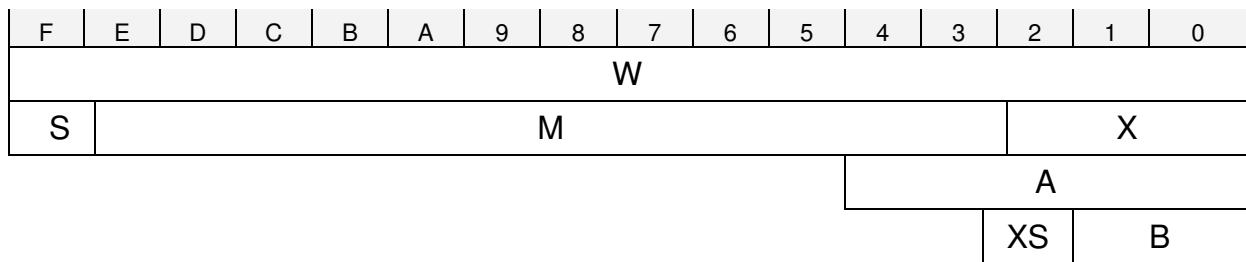
Con un tamaño maximo de dirección de 20bits

Espacio logico de direcciones: 512Kb

Tamaño maximo de registros de 64 bits.



Fields



HST (Hardware Status Bits)

Bit 4	Bit 3	Bit 2	Bit 1
MP	SR	SB	XM
0/1	0/1	0/1	0/1

El **programming model** del Saturn se conforma por los siguientes registros:

Working Registers: son los registros A, B, C, D de 16 nibbles o 64 bits cada uno. A veces llamados registros de calculo, se utilizan como registros de propósitos generales. Son registros directamente relacionados con la Unidad Aritmetica Logica (ALU). De los cuatro registros de trabajo, los dos mas importantes son el A y el C, si bien, se puede cargar datos y borrar a cualquier registro de trabajo directamente, el A y C son los unicos registros a los que se puede acceder directamente con un dato desde la memoria, luego para llevar ese dato a los B y D se mueve el dato desde los registros A y C. Asimismo el registro C es mas rapido que el registro A, por esa razon este registro C se utiliza con fines donde los ciclos de trabajo son importantes, por ejemplo manipulación de objetos graficos.

Save Registers: tambien conocidos como Scratch registers, son 5 registros de resguardo, R0, R1, R2, R3 y R4 de 16 nibbles o 64 bits cada uno. Se utilizan como registros auxiliares para guardar datos de los registros de trabajo en operaciones complejas o para ciertos parametros adicionales de ciertas rutinas de codigo ML.

Punteros: hay tres registros punteros, D0, D1, PC, de 5 nibbles cada uno.

D0 y *D1*: se utilizan para apuntar una direccion de memoria donde se guarda un dato, una subrutina, etc. En algunos casos se puede llegar a utilizar para almacenar algun dato. Por esa razon tienen 5 nibbles, dado que el espacio de memoria direccionable del microprocesador Saturn va desde #00000h hasta #FFFFFh

PC es el conocido Program Counter, se utiliza para apuntar a la proxima direccion de la siguiente instrucción de programa.

Registro P: es un registro de 1 nibble, se utiliza para definir el campo (field) que se va a tomar para trabajar. Con el registro P se puede definir el WP que es el Wide-P, es el tamaño en nibbles desde el primer nibble hasta el nibble n. Mas adelante se vera como se pueden tomar los fields.

RSTK conocido como Return Stack: utiliza el mecanismo LIFO, Last In First Out para trabajar. Es usado por las subrutas. Cuando se llama a una subrutina desde un programa en lenguaje assembler, el procesador graba donde esta (en el RSTK) y luego salta a la subrutina, cuando regresa, llama a la primer direccion del Stack y salta a la posición donde habia comenzado la subrutina.

FLAGS: hay 3 tipos de Flags, cada uno utiliza registros diferentes:

Carry: es el que se habilita cuando se produce un overflow en alguna operación aritmetica o en algun shift con carry. Tambien se habilita cuando se da una condición positiva en algun salto condicional. Es el registro mas utilizado a la hora de realizar saltos relativos, por los pocos ciclos de trabajo que requiere y por la simplicidad.

ST(Status Bits): compuesto por 4 nibbles, tiene 16 bits disponibles para distintos estados del microprocesador. Los codigos entre el 0 y el 11 son de uso interno del micro. Del 12 al 15 tienen operaciones especiales:

Bit 12: Fuerza un Wakeup request

Bit 13: Se setea si ocurre una interrupción

Bit 14: Seteado si hay una interrupción pendiente

Bit 15: Seteado en 1 si las interrupciones estan habilitadas y en 0 deshabilitadas
(se utiliza para enmascarar interrupciones)

HST(Hardware Status Bits): El Saturno utiliza estos 4 bits para eventos especiales.

Bit 4	Bit 3	Bit 2	Bit 1
MP	SR	SB	XM
0/1	0/1	0/1	0/1

XM: eXternal Module missing

SB: Sticky Bit (cuando un MSB pasa a LSB en un shift y se pierde un dato)

SR: Service Request

MP: Module Pulled

El que mas se utiliza es el SB que se habilita cuando en un calculo o en un shift al pasar el un MSB a un LSB se pierde algun dato, se habilita esta bandera para indicar que un dato esta perdido.

OUT e IN: son los registros utilizados para ingresar y extraer datos desde el teclado y para producir sonidos.

Se utilizan los siguientes tamaños:

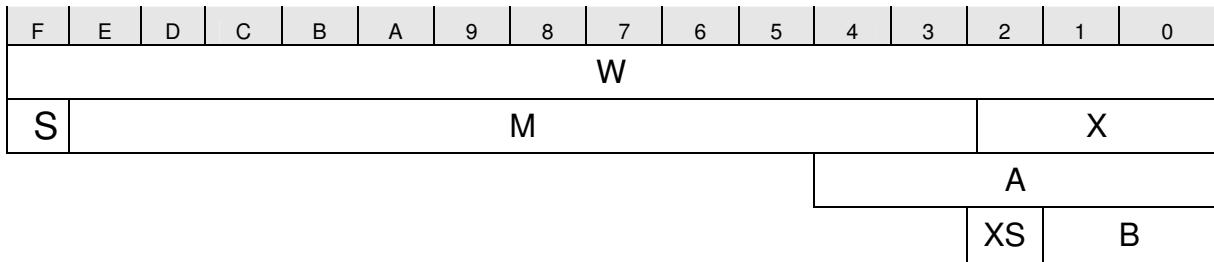
OUT 3 nibbles o 12 bits

IN 4 nibbles o 16 bits.

FIELDS:

Los campos son muy importantes a la hora de seleccionar los datos que se toman para colocar en los registros. No siempre se utilizan los 16 nibbles del registro de trabajo, muchas veces con tomar 5 nibbles por ejemplo para una direccion de memoria alcanza. Esto requeriria muchos ciclos de memoria, para eso se utilizan porciones acotadas de registro llamadas Campos o Fields.

La estructura que se utiliza es la siguiente:



Campo W (Word): toma todos los bits del registro de trabajo: 64bits.

Campo A (Address): es un campo de 5 nibbles, se suele utilizar para direcciones de memoria, va desde el bit 0 hasta el 4.

Campo B (Byte): un byte.

Campo X (Exponent): se suele utilizar para procesar los datos de OUT tambien es el exponente de un numero expresado en punto flotante. 3 nibbles

Campo XS (Exponent Sign): 1 nibble es el MSB del registro X

Campo M (Mantissa): 12 nibbles desde el 3 al 14 se utiliza para representar numeros reales en punto flotante junto con el exponente y el XS.

Campo S (sign): es el ultimo nibble del registro de trabajo, en numero 15.

Inversion de Nibbles

Una curiosidad del microprocesador saturno que puede llegar a complicar a la hora de la programación en lenguaje de maquina (ML) es la inversion de nibbles. El Saturn invierte el orden de los nibbles cuando lee o cuando escribe en memoria. Esto se llama arquitectura “little endian” y es utilizada por microprocesadores con arquitectura x86 PC.

Set de instrucciones

El microprocesador Saturn posee un set de instrucciones CISC con pocas instrucciones, pero repetidas para diversos objetivos y para cada registro.

En la estructura de memoria, nos encontramos con un sector de memoria ROM de 251Kbyte. En este sector de memoria se encuentra almacenado el sistema operativo propio del manejo de la calculadora. Pudiendose acceder a las distintas subrutinas pre programadas por los “enter points” que son las direcciones de memoria donde se encuentran dichas instrucciones. (Llamandose a este sistema de programacion System RPL) En la suma del set de instrucciones basico del Saturno y los comandos disponibles en ROM se tiene una gama muy amplia de posibilidades de programacion.

Obteniendose en algunos casos mejores resultados (en funcion de ciclos de programa) acediendose por los enter points que programando las subrutinas manualmente.

Igualmente la programacion en ML permite mayor flexibilidad.

Set de intrucciones reducido

A continuación se describen algunas intrucciones basicas que se utilizaron en el programa de la comunicación serie:

Carga de datos en los registros de trabajo:

Mnemonic	Opcode	Ciclos
LA n...1	8082 n 1...n	7.5+1.5n
LC n...1	3 n 1...n	3+1.5n

Aca se puede ver que el registro de trabajo C es mucho mas rapido que el A

Cambio del valor de un bit:

Mnemonic	Opcode	Ciclos
ABIT=0 n	8084n	7.5
ABIT=1 n	8085n	7.5
CBIT=0 n	8088n	7.5
CBIT=1 n	8089n	7.5

Dar valores a D0 o D1

Mnemonic	Opcode	Ciclos
D0=yz	19zy	6
D0=wxyz	1Azyxw	9
D0=vwxyz	1Bzyxwv	10.5
D1=yz	1Dzy	6

D1=xyz	1Ezyxw	9
D1=vwxyz	1Fzyxwv	10.5

Recordar que los datos van invertidos vwxyz=>zyxwv

Sumar o restar D0 o D1

Mnemonic	Opcode	Ciclos
D0=D0+ n	16(n-1)	8.5
D0=D0- n	18(n-1)	8.5
D1=D1+ n	17(n-1)	8.5
D1=D1- n	1C(n-1)	8.5

Entre n y el signo hay un espacio.

Copiar A/C a Do/D1

Mnemonic	Opcode	Ciclos
D0=A	130	9.5
D0=C	134	9.5
D1=A	131	9.5
D1=C	131	9.5

Intercambiar field A de A/C con D0/D1

Mnemonic	Opcode	Ciclos
AD0EX	132	9.5
AD1EX	133	9.5
CD0EX	136	9.5
CD1EX	137	9.5

Ex es de Exchange

Leer memoria desde una direccion con field:

Mnemonic	Opcode	Ciclos
A=DAT0 A	142	23.5,3.5
A=DAT1 A	143	23.5,3.5
C=DAT1 A	147	23.5,3.5

Y asi todas las variants

Escribir a una direccion de memoria con field

Mnemonic	Opcode	Ciclos
DAT0=A A	140	19.5
DAT1=A A	141	19.5
DAT1=C A	145	19.5

Saltos:

Incondicionales:

GOTO

Mnemonic	Opcode	Ciclos
GOTO label	6xyz	14

Puede saltar 2048 nibles antes y 2047 despues

GOVLNG

Mnemonic	Opcode	Ciclos
----------	--------	--------

GOVLNG label 8Dvwxyz 18.5
Para saltar a cualquier parte de la memoria.

Subrutinas:

GOSUB

Mnemonic	Opcode	Ciclos
----------	--------	--------

GOSUB label	7xyz	15
-------------	------	----

Puede saltar 2048 nibles antes y 2047 despues

GOSBVL

Mnemonic	Opcode	Ciclos
----------	--------	--------

GOSBVL label	8Fvwxyz	19.5
--------------	---------	------

Puede saltar a cualquier parte de la memoria.

Condicionales

Test

Para realizar un test primero se debe plantear la condicion y luego se llama a un comando de accion: GOYES label saltos condicionales o RTNYES label subrutinas

Comparacion de registros con cero:

Mnemonic	Opcode	Ciclos
?A=0 A	8A8 yz	13.5/21.5
?C=0 A	8A9 yx	13.5/21.5

Testeo de Bit

Mnemonic	Opcode	Ciclos
?ABIT=0 n	8086n yz	12.5/20.5
?ABIT=1 n	8087n yz	12.5/20.5
?CBIT=0 n	808An yz	12.5/20.5
?CBIT=1 n	808Bn yz	12.5/20.5

Para la realizacion del programa se hizo uso de los registros Scratch o Save Registers:

Guardar un registro de trabajo dentro de un save register:

Mnemonic	Opcode	Ciclos
R0=A	100	20.5
R0=C	108	20.5

Desplaza un bit a la derecha con field:

Mnemonic	Opcode	Ciclos
ASRB f	819a0	8.5+n
	819F0 (A)	13.5
CSRB f	819a2	8.5+n
	819F2 (A)	13.5

Donde:

Field	Valor a
P	0
WP	1

XS	2
X	3
S	4
M	5
B	6
W	7

Suma de una constante a un registro de trabajo:

Mnemonic	Opcode	Ciclos
A=A+c f	818F0(c-1) (A)	8+n
	818a0(c-1)	8+n
C=C+c f	818F2(c-1) (A)	8+n
	818a2(c-1)	8+n

Nota: los operadores trabajan en modo decimal, salvo que se especifique lo contrario.

Complemento a2

Mnemonic	Opcode	Ciclos
A=-A f	F8	8
	Bb8	4.5+n

Nota: puede afectar el carry

OBJETOS:

Es la parte esencial del manejo de HP48 y 49. Todo lo que se puede poner en algun nivel de pila es un objeto encodeado de la siguiente manera:

-5 nibbles para un “prologo” que se utiliza para identificar el objeto.

-El objeto en si con todos sus parametros llamado el “cuerpo del objeto”

Para el modo RPL de usuario hay disponibles 28 objetos diferentes. Sin embargo internamente existen otros objetos que usa el propio sistema operativo.

La importancia del manejo e identificacion de objetos radica en el problema de acceder al microprocesador, poder manipularlo y retornar en forma segura al sistema operativo, sin generar una ruptura del codigo principal de operaciones. Esto produciria un reseteo del microprocesador y por consiguiente la perdida total del contenido de la memoria RAM.

Los objetos mas comunes son: Real Number, String, Global Name, Local Name, User binary integer, Graphics Object, Xlib name, Long real, Character, Code object, Library data, Complex number, Array, List, RPL program, Algebraic expression, Unit object, Directory, Library, etc.

Para la verificacion de funcionamiento de la conexion serie se utilizo el objeto String, que se detalla a continuacion:

String

Prologo: 02A2C	Epilogo: ninguno
Tamaño: 12 nibbles o mas	Type: 2

Type es el tipo de objeto como se lo identifica en user RPL, se puede obtener el tipo de un objeto realizando lo siguiente:

Ej: se ingresa un código string en el nivel 1

Nivel 1 “Hola”

Luego se escribe el comando Type y se devuelve el tipo de objeto:

Nivel 1 2

Prologo (02A2C)	5 Nibbles
Tamaño	5 Nibbles
Carácter 1	2 Nibbles
...	
Carácter 2	2 Nibbles

El tamaño del String está compuesto por 5 nibbles (del tamaño) más los 2 nibbles por carácter. Hay que tener en cuenta que dentro del tamaño se incluyen los 5 nibbles del campo “tamaño”. Al usarse 2 nibbles por carácter se trabaja con el modo ASCII extendido que soporta $2^8=256$ diferentes valores. Los primeros 32 son operaciones especiales, de comunicación o impresión, son los llamados “caracteres no imprimibles”. Los primeros 128 son estandares y los 128 restantes varían de país a país y de computadora a computadora.

Por ejemplo el string “HP” se encodea de la siguiente manera: #02A2Ch para el string prologo, luego el cuerpo del objeto de la siguiente manera:

Tamaño= 5 nibbles (Tamaño) +2*2 nibbles (2 caracteres)=9 nibbles

Y el código en hexadecimal del carácter: #4850

De esta forma el objeto “HP” visto en memoria queda: #C2A20 90000 8405h

Se observa la inversión de nibbles.

Objetos gráficos

Prologo: 02B1E Epilog: ninguno
Tamaño: 22 nibbles o más Type: 11

Prologo (02B1E)	5 Nibbles
Tamaño	5 Nibbles
Número de filas	5 Nibbles
Número de columnas	5 Nibbles
Cuerpo	Pixeles

GROB (GRaphic OBject): los objetos gráficos contienen datos que representan pixeles. La primera parte del objeto contiene el prologo #02B1Eh, seguido por el tamaño (donde se incluye el propio campo de tamaño), luego el número de filas y columnas, cada uno de 5 nibbles cada uno. Por último el cuerpo representado por pixeles (contracción de “picture element”)

El primer pixel, se representa en la esquina superior izquierda, y el último en la esquina inferior derecha. El dato de los pixeles se representa en bytes, por eso se toman de a dos

nibbles, o nibbles de numero par. Cada bit representa un pixel, siendo un “1” un elemento encendido y un “0” un elemento apagado.

En los registros hay que tener en cuenta que cada byte encodea a los pixeles con el modo de nibble invertido.

Ej: supongamos que queremos tener un pixel off, un pixel on, un pixel off, un pixel on, un pixel off y tres pixeles mas on:

0101 0111

Se tienen 8 pixeles o sea 1 byte o 2 nibbles. Esto se encoda de la siguiente forma:

1010 1110 → #AEh → #AE00

Se tiene 1 fila y 4 columnas, entonces queda (#10000h) y (#40000h)

Si se cuenta todos los nibbles da: 14 nibbles +5 nibbles (size)= 19d = #13h

Por lo tanto el objeto completo quedaria:

#E1B20 31000 10000 40000 AE00

Comunicación serie:

El HP48 UART (Universal Asynchronous Receiver-Transmitter) usa los siguientes registros:

Registro	BITS	DESCRIPCION
#0010D	0-2	3 bits que determinan los baudios 1200 1920 2400 3840 4800 7680 9600 15360 * ¹
.	3	UART clock (R/O)
#00110	All	UART Interrupt control register 0 Enable interrupt on RECV buffer receiving 1 Enable interrupt on RECV buffer full 2 Enable interrupt on XMIT buffer empty * ² 3 Wire serial port enable
#00111	All	UART RECV control register 0 Character present in receive buffer 1 UART receiving carácter 2 Error occurred while receiving 3 Not used
#00112	All	UART XMIT control register
#00113	All	Clear received error if written to (W/O)
#00114	All	LSBs of RECV holding buffer carácter (R/O)
#00115	All	MSBs of RECV holding buffer carácter (R/O)
#00116	All	LSBs of XMIT holding buffer carácter (W/O) * ²
#00117	All	MSBs of XMIT holding buffer carácter (W/O) * ²

*¹ Notese que mediante el control de registros se puede setear 15360

*² XMIT es el comando de transferencia, funciona como el SRECV.

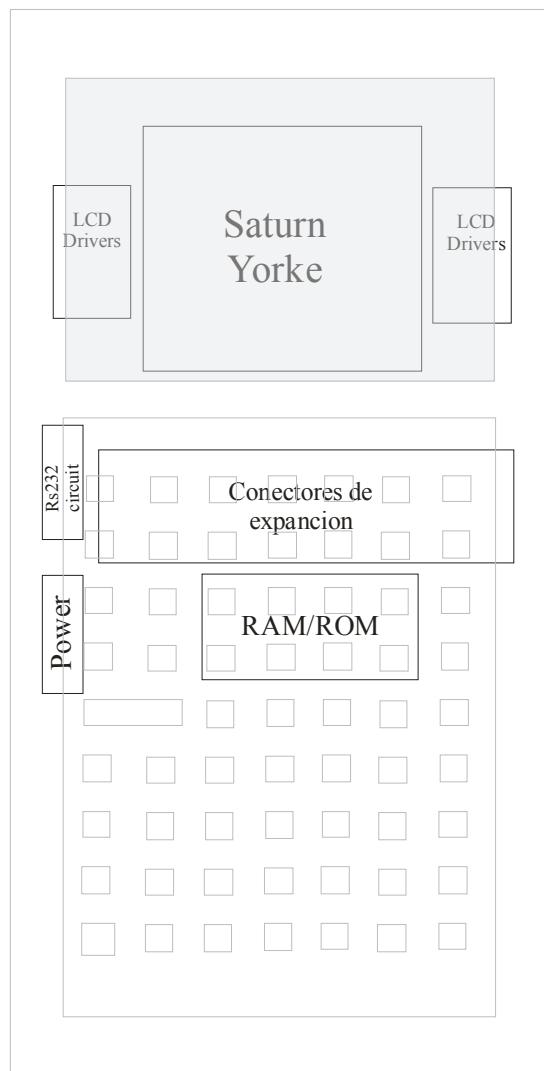
Recepcion:

Las interrupciones asociadas con el puerto serie son controladas por el registro #00110h. Cuando el primer bit de un carácter es recibido, se generara una interrupcion y el bit 0 de #00110 se setea avisando que entro un dato, el bit 1 de #00111 se setea avisando que hay un carácter entrando. Cuando el carácter es recibido, el bit 1 de #00111 se apaga y se setea el bit 0 que informa que hay un carácter presente en el buffer. Tambien se genera otra interrupcion si el bit 1 del #00110 se setea. La rutina de interrupciones captura el carácter recibido y setea el bit 0 de #00111 a cero, indicando que el buffer esta listo para recibir nuevamente.

Graficos HP Saturn

GENERALIDADES

Las calculadoras basadas en un chip Saturn Clarke o Yorke usan un display de cristal liquido (LCD) con una resolucion de 131 por 64 pixeles. El LCD es controlado por el propio microprocesador y dos chips drivers comerciales: SED1181Fla (Japan)



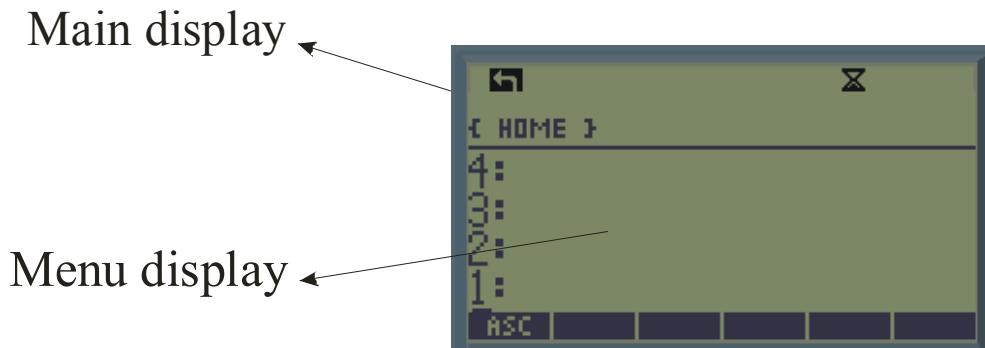
La frecuencia de trabajo normal del microprocesador Saturn ronda los 4096Hz (dependiendo de la temperatura). A esa frecuencia se puede redibujar una linea en la pantalla. Como la pantalla tiene 64 lineas, el refresco de pantalla se da a una frecuencia

de 4096Hz / 64 lineas = 64Hz. Esto daria una refresco de linea aproximadamente de entre 22 a 23 us y 244us de pantalla. Se utiliza una arquitectura UMA (Unified Memory Architecture), esto significa que la memoria RAM del CPU y los controladores de display estan juntos . La ventaja de esto es que el CPU tiene un control total sobre el display, y no se necesitan copiar datos del CPU a una memoria de display. La desventaja es que cuando el display es refrescado, el CPU se detiene garantizando que la memoria RAM se utilizara para el control del display. Esto baja un poco la velocidad del CPU.

El display es controlado por varios registros de E/S mapeados desde #00100 hasta #00134. El display solo trabaja con Objetos Graficos (GROBs).

PRINCIPAL/MENU DISPLAY

El display se divide en dos partes por hardware: area “Main display” y area “Menu display”.



En la parte superior se encuentra el Main display area, en este area el GROB puede ser grande o pequeño. Es posible desplazar el contenido programando el controlador de display cambiando el dato del GROB. En la parte baja se encuentra el Menu display que tiene un ancho de 131 pixeles. Se puede modificar el Main display modificando el area de Menu.

Programas

Desarrollo de la transmisión serie

Calculo de la transferencia serie utilizando el modulo TIM.

Para obtener la velocidad de transmisión se realizo el siguiente calculo:

Frecuencia de oscilador interno del microprocesador HC908: 12.8 MHz

$$\text{Entonces la frecuencia del bus de datos es } \frac{F_{Xtal}}{4} = 3.2 \text{ MHz}$$

Por lo tanto el tiempo de cada ciclo de programa es de $\frac{1}{F_{Bus}} \approx 0,3125 \mu\text{s}$

A 9600 baudios la velocidad de sincronismo de c/bit enviado es:

$$1/9600[\mu\text{s}] = 104,1666 \mu\text{s}$$

Entonces entre cada bit enviado debe haber $104,1666 \mu\text{s} / 0,3125 \mu\text{s} = 333.333$ ciclos

Se redondea a 333T

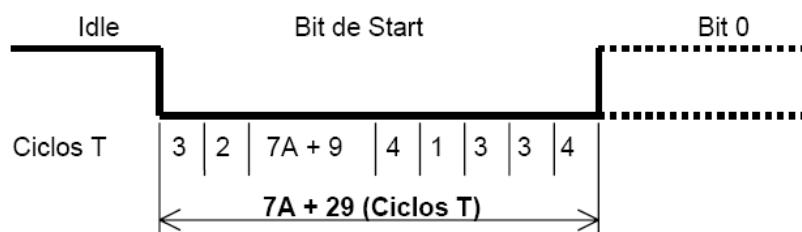
Como se vera mas adelante la cantidad de ciclos que se usan en lineas de programa son de 29 mas 7 ciclos en la subrutina de delay, por lo tanto para calcular el valor del delay exacto se hace: $29+7A=333$ entonces $A = 43.42 \approx \#43d = \#2Bh$

El error absoluto cometido es : $1,041 \mu\text{s}$

El error de esta manera es del 0,99%.

Esto se realizo para generar el delay con instrucciones:

Necesitamos obtener $104,16 \mu\text{s}$ con el modulo tim.



Seleccionando el prescaler en cero, se obtiene un ciclo de $0,3125 \mu\text{s}$.

Para obtener los $104,16 \mu\text{s}$ seteamos el TMODX en 12B.

$$(34\text{ciclos}+299\text{ciclos})*(0,3125 \mu\text{s}) = 104,0625 \mu\text{s}$$

299 en hexadecimal es 12B

El error absoluto en este caso seria de $0,0975 \mu\text{s}$

El error relativo quedaria: 0,0936%

Y para el bit_stop que antes se tomaba dividiendo 2B por 2 para obtener un delay de aproximadamente $52 \mu\text{s}$.

Para este caso con el modulo Tim se utilizo el canal cero para obtener el tiempo.

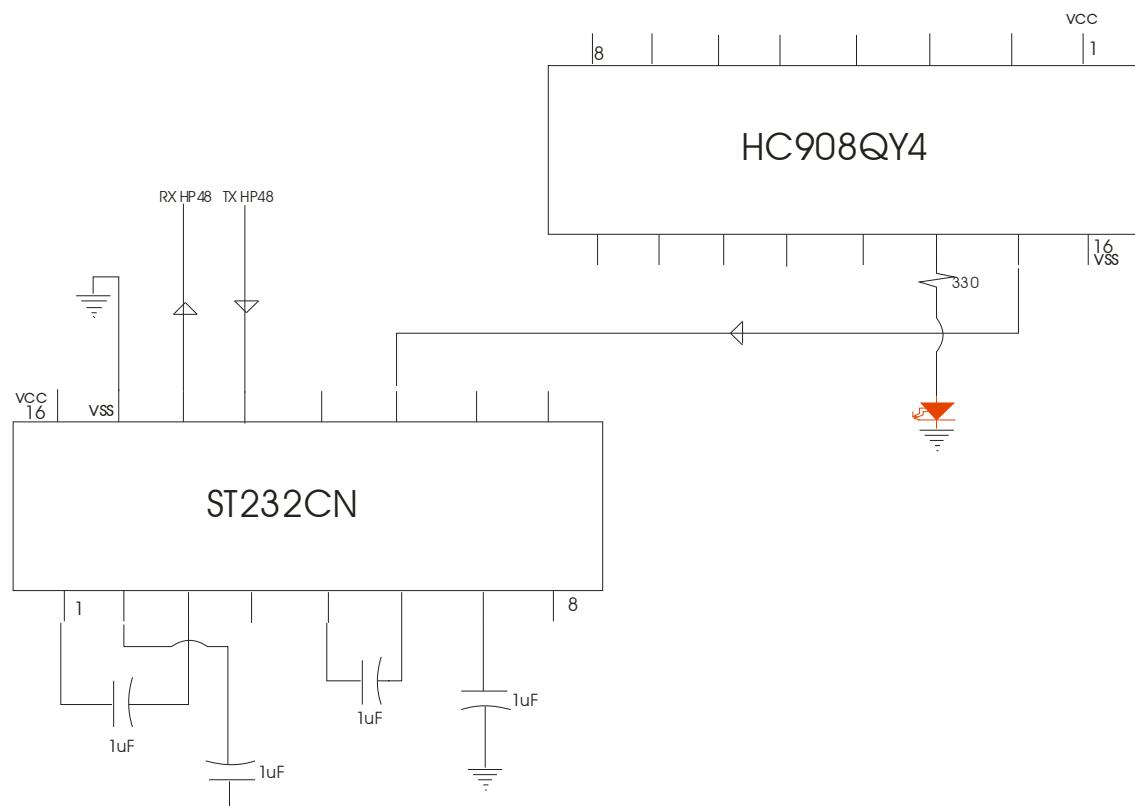
Seteamos el TCH0X en 96(hexa),

El valor ideal seria: $299.333/2$ y el valor usado fue $300/2$ ciclos

cometiendo un error de $0,10417 \mu\text{s}$ que equivale a 0,19%.

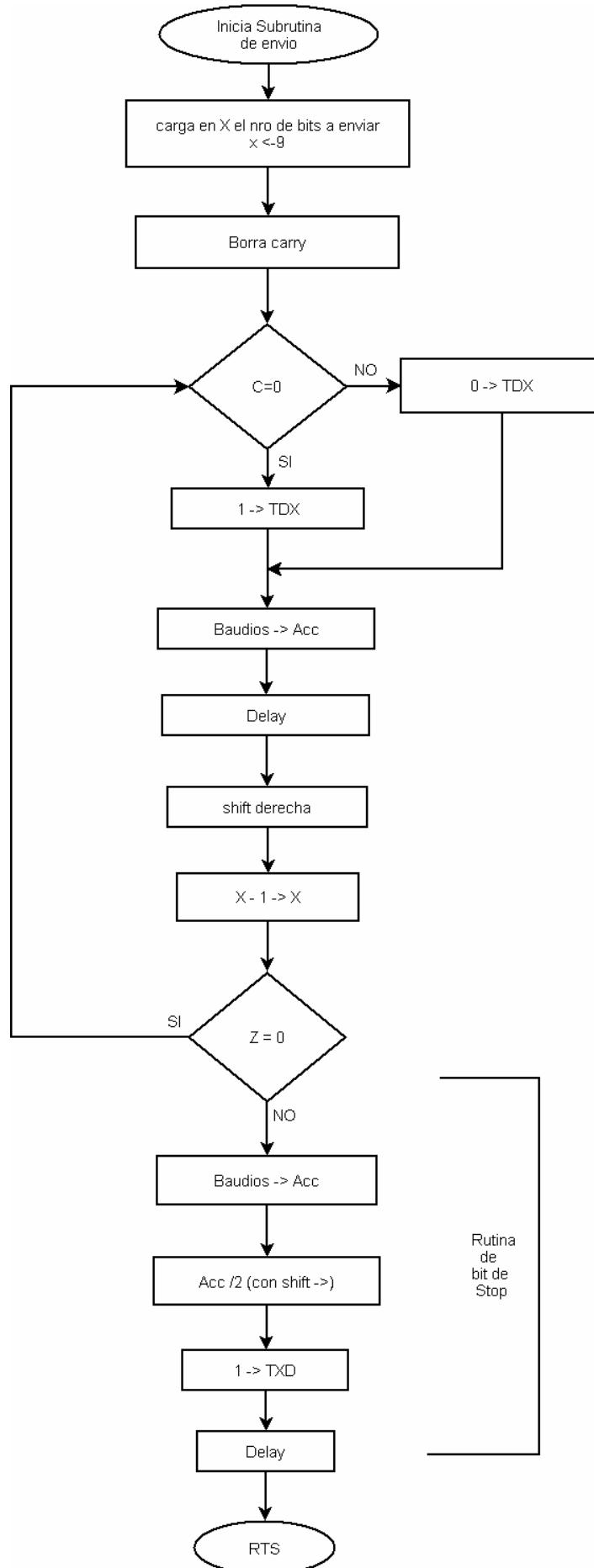
Entonces el error total es: $0,0936\% + 0,19\% = 0,2836\%$

El circuito para realizar las pruebas fue el siguiente:



n es el siguiente:

El diagrama de flujo del programa de transmisión



Programa:

```
;*****SETEO DEL MODULO TIM*****
    mov #$10,$20      ;seleccion preescaler,Tstop y Trst
    mov #$10,$25      ;seteo de output compare
    mov #$11,$23      ;definimos el tmmodh
    mov #$2b,$24      ;def el tmodl
    mov #$0,$26        ;definimos el TCH0H
    mov #$96,$27      ;def el TCH0L

;*****ENVIO PUERTO SERIE*****
saca_caracter:
    stx auxiliar      ;[3] guardo valor de X
    ldx #9            ;[2] bits por caracter (incluido bit start)
    clc              ;[1] borra carry para enviar bit de start
saca_bit_dato:
    bcc saca_0        ;[3] si cy=0, salta para sacar 0      [3]
    bset 3,$01        ;[4] si no saca 1          [4]
    bra otro_bit      ;[3]                                [3]
saca_0:
    bclr 3,$01        ;[4]
    bra otro_bit      ;[3]
otro_bit:
    jsr delay_7a      ;[7a+9]demora=7*31+9=226T      [4]
    ror caracter      ;[4] busca siguiente bit (va al carry) [4]
    decx              ;[1]                                [1]
    bne saca_bit_dato ;[3] 8T vuelvo a sacar el sig bit [3]
saca_bit_stop:
    bset 3,$01        ;[4] __| sube linea para bit de STOP
    jsr delay_7b      ;[4]
    ldx auxiliar      ;[3]recupero valor de X
    rts              ;[4]

;-----SELECCION DE VELOCIDAD DE TRANSMISION-----
delay_7a  bset 4,$20          ;[4]
          bclr 7,$20
dela     brclr 7,$20,dela      ;[299]
          rts ;[4]=7a+4 cycles      [4]
delay_7b  bset 4,$20
          bclr 7,$25
dela1   brclr 7,$25,dela1      ;[4]
          rts
```

Captura y envio de datos HP:

Para realizar el programa de captura y envio de datos en el microprocesador HC908 se utilizo el programa anterior. En el dispositivo HP como primera aproximación se realizo un programa en User RPL (lenguaje de alto nivel).

La configuración del puerto serie que se realizo en el dispositivo HP fue la siguiente:

PORT: WIRE
TRANSFER MODE: BINARIO
BAUD: 9600
PARIDAD: NONE □
CKSUM: 1

TRASLATE □ (intenta acomodar los datos a caracteres validos)

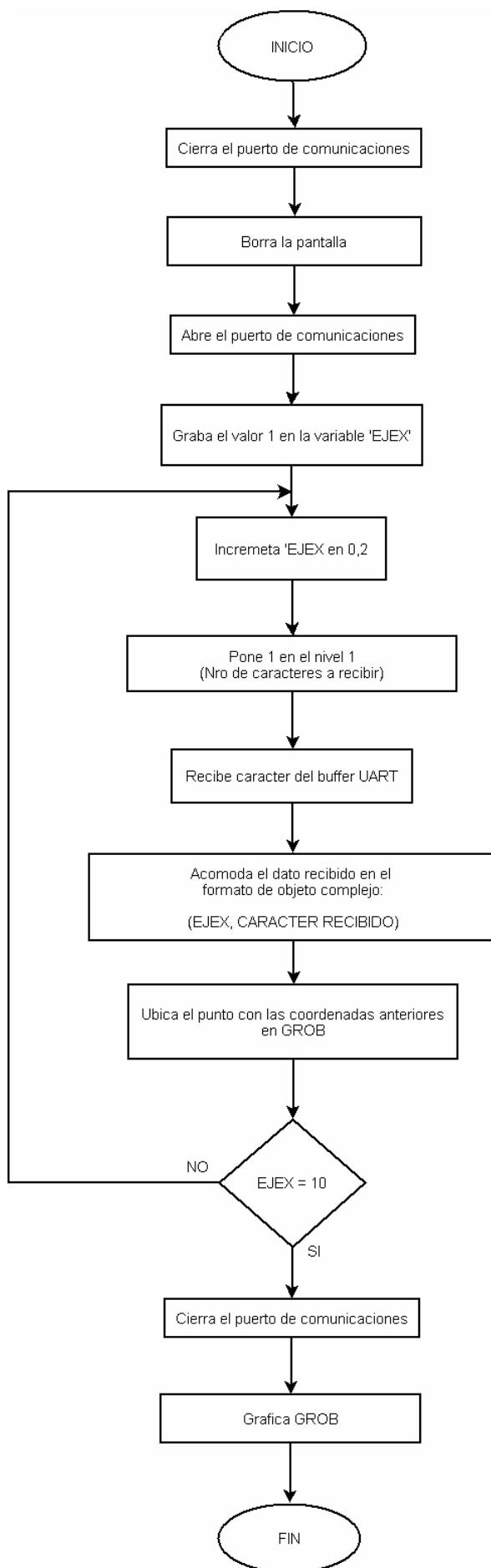
Para realizar la comunicación se realizaron los comandos de E/S serie generales.

Archivo: INICIO

```
« CLOSEIO      →cierra el puerto
ERASE          →borra pantalla
OPENIO         →abre puerto
1
'EJEX'
STO
GRAFICA        →llama al siguiente archivo
»
```

Archivo: GRAFICA

```
« EJEX
.2
+
'EJEX'
STO
EJEX          →pone en la pila el contenido de la variable
1             →pone 1 (valor n) en el nivel 1
SRECV          →recibe cantidad de caracteres (n) del bufer
DROP           →borra el nivel 1
NUM            →convierte a decimal el contenido del char
|
*
+
→NUM          →pone en formato (x,y) el dato
PIXON
IF 'EJEX==10'
THEN CLOSEIO
PICTURE        →grafica todos los puntos recibidos
END
GRAFICA
»
```



Programas sobre el SATURN ASSEMBLER

Programa tipo de demostración usando Saturn assembler

Se realizo un programa que capture los primeros 4 caracteres recibidos por el puerto serie y luego los escriba en el nivel 1 como carácter. (Por facilidad de programacion se necesita que antes de ejecutarse el codigo haya una cadena de 4 caracteres en el nivel 1). El codigo es el siguiente:

```
::  
CODEM  
GOSBVL =SAVPTR %Guarda Los registros del sistema operativo  
A=DAT1 A      %TRAE EL DATO DE LA DIRECCION DEL STAK  
D1=A          %TRAIGO LA DIRECCION DEL STACK  
D1=D1+ 5      %me paro despues del prologo  
AD1EX  
D1=A  
D1=D1+ 5      %me paro despues del size  
AD1EX  
LC 4  
D=C B  
*OTRO  
$1FD0100      %HACE D1= 0010D QUE ES EL ENTER POINT DE  
               %VELOCIDAD  
C=DAT1 B      %CARGA EL CONTENIDO DE LA DIRECCION DEL SETEO de  
               %VELOCIDAD EN 15360 BAUDIOS  
CBIT=1 0      %SETEA A 1 EL BIT 0  
CBIT=1 1      %SETEA A 1 EL BIT 1  
CBIT=1 2      %SETEA A 1 EL BIT 2  
DAT1=C B      %MUEVE EL VALOR SETEADO A LA DIRECCION DEL EP  
$1F01100      %HACE D1= 00110 UART INTERRUPT REGISTER  
C=DAT1 B      %HABILITA EL RECV BUFFER Y EL WIRE ENABLE SERIAL  
DAT1=C B      %MUEVE EL VALOR A LA DIRECCION  
*BUCLE  
$1F11100      %SE FIJA SI EL BIT 0 DE LA DIRECCION  
               %SE FIJA EN LA DIRECCION 00111 SI EL BIT 0  
               %00111 ESTA ACTIVO, EN ESE CASO ES PORQUE  
               %EL BUFFER TIENE UN DATO  
C=DAT1 B  
?CBIT=0 0  
GOYES BUCLE  
               %ACA SE COPIA EL DATO RECIBIDO AL STACK  
?D=0 B  
GOYES END  
$1F41100      %REGISTRO 00114  
               %TRAIGO DOS NIBLES 00114 Y 00115  
               %SALVO EL STACK  
C=DAT1 B      %TRAE EL DATO RECIBIDO DOS NIBBLES  
D1=A  
DAT1=C B  
D1=D1+ 2  
A=D1 B  
D=D-1 B
```

```

GOTO OTRO

%-----SALIDA A RPL-----
*END
GOSBVL =GETPTR      %Restaura los datos de los registros
GOVLNG =LOOP        %Ratorna al RPL

ENDCODE
;

```

El enter point =SAVptr es una subrutina de la ROM del microprocesador se utiliza para salvar los registros y las posiciones en las que estaba parado el microprocesador antes de entrar en el código. Se utiliza para regresar seguro al código.

Esta subrutina desensamblada sería así:

```

CD0EX    %mueve D0 a C
D0= 8072F %pone D0 en el punto #8072Fh
DAT0=C A  %escribe D0 ahí (lo salva)
D0= 806F8 %pone D0 #806F8
CD1EX
D1=C
DAT0=C A  %salva D1
D0= 806F3
C=B A
DAT0=C A  %salva B(A)
D0= 807ED
C=D A
DAT0=C A  %salva D(A)
RTNCC    %retorna de la subrutina y borra el carry

```

Las direcciones donde se guarda D0, D1, B y D son direcciones reservadas para ese propósito, luego cuando se desea restaurar los registros, se llama a la subrutina: =GETPTR que hace los procesos inversos.

La rutina =LOOP realiza las siguientes operaciones

```

A=DAT0 A
D0=D0+ 5
PC=(A)

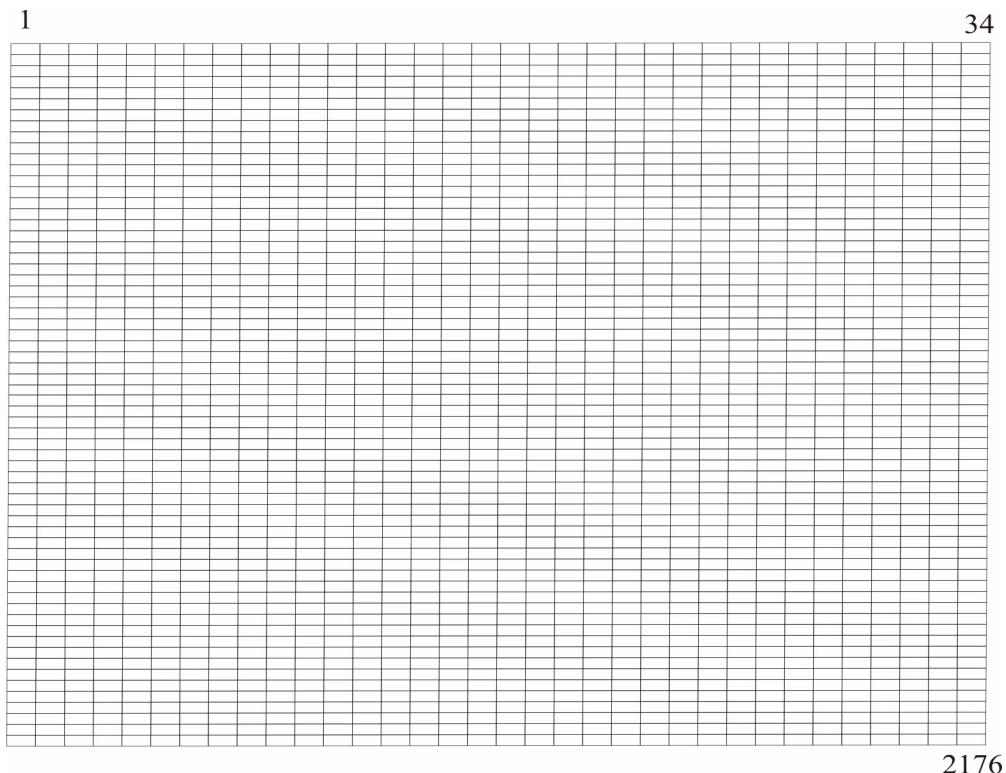
```

Desarrollo del software gráfico en Saturn Assembler

COMO HUBICAR UN PIXEL EN LA PANTALLA

Como se dijo anteriormente la pantalla tiene un ancho de 131 por 64 pixeles, esto sería equivalente a decir $131/8=16,375$ Bytes. Los Bytes a representar deben ser números enteros, por lo tanto se toma el valor inmediato siguiente 17 Bytes. Si se multiplica por dos se obtiene el número exacto de nibbles que hay en una línea: 34 nibbles. La segunda línea comienza con el nibble número 35. Si se tienen en cuenta las 64 filas de pantalla se tendría 34 por 64 nibbles: 2176 nibbles. Si se le suman el prologo, el size, la altura y el ancho (5 por cada uno) daria que el tamaño total de un GROB que ocupe toda la

pantalla es de 2196 nibbles o 1098 Bytes. Si se le restan los 5 nibbles del prologo, se tiene que el size de este objeto es: 2196.



Entonces cada vez que se salta a una nueva linea se debe sumar 34 a la posicion anterior.

Registro flotante:

Para poder imprimir en pantalla un dato recibido por el puerto serie, se diseño un mecanismo de movimiento de puntero de direccion D1 que se llamo Registro Flotante. Cada dato que se recibe desde el microprocesador HC908 representa un valor entre #0h y #FFh, o sea entre 0 y 255.

Entonces se necesitarian 256 filas para representar. Dado que se dispone de 64, entonces habria que dividir a las 256 filas por #4d (#100b). Una forma facil de realizar esto es haciendo dos shift hacia la derecha del dato recibido.

$$\begin{array}{r} 256 = 100000000 \\ \hline 64 = 1000000 \end{array}$$

Este valor deberia ir en la parte mas alta del grafico, sin embargo como se vio antes, el grafico comienza en la esquina superior izquierda, por lo tanto habria que invertir las posiciones entonces se hace:

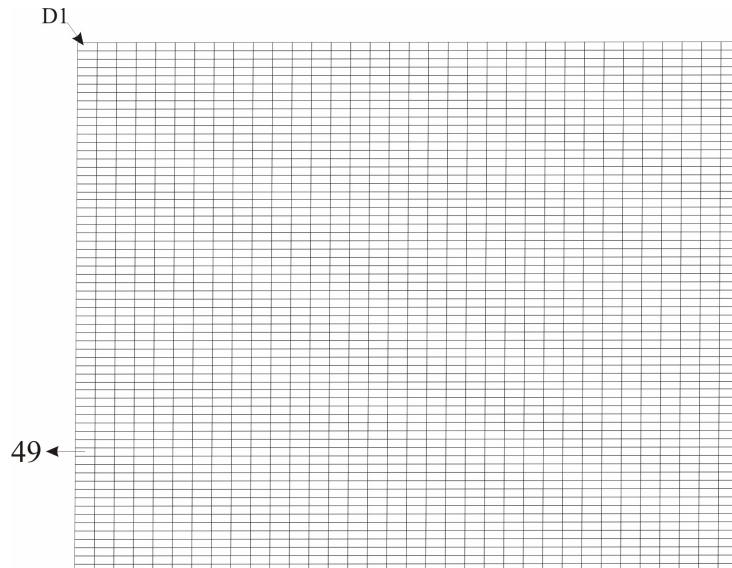
$$64 - \text{posicion} = \text{linea}$$

En este momento el dato recibido tiene un valor entre 0 y 64, para compararlo con el registro flotante D1, se lo debe multiplicar por la cantidad de nibbles que hay en pantalla asi el valor queda entre 0 y 2176 nibbles. A este valor se le suma la posicion del comienzo del GROB y de esta forma se obtiene un valor comparable con D1.

Se carga el registro flotante D1 y se lo compara con el Dato. Si el Dato es diferente al registro, D1 se moverá 34 nibbles para arriba o para abajo hasta acomodarse con el dato, luego se imprime un punto en este lugar y D1 se incrementa en un nibble. De esta forma cuando se vuelve a buscar el nuevo Dato, D1 se moverá 34 veces para arriba o abajo, pero con un incremento de uno. Así se va recorriendo toda la pantalla en forma horizontal.

Por ejemplo, supongamos los dos primeros ciclos, donde entran dos datos 3F y 40 consecutivamente.

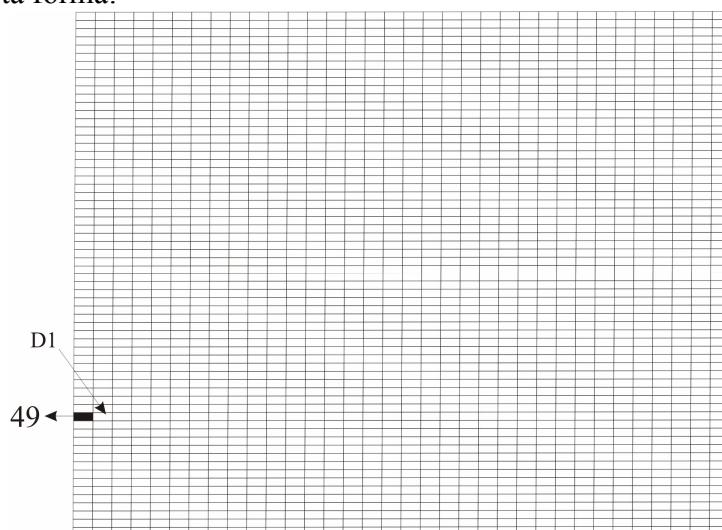
Primero el puntero D1 se pone sobre la posición 0 del body del GROB:



$\#3Fh / 4 = \#63d / 4 = \#15.75d = \#F$ (notese que hay un primer redondeo automático)
Entonces el valor 3F se mapea como F

Se invierte la coordenada: $64 - 15 = 49 \rightarrow$ coordenada Y del GROB

Cuando se compara D1 con 49, $D1 < 49$ entonces se suman 34 tantas veces sean necesarias hasta alcanzar los 49. Luego se imprime el dato y D1 se incrementa en 1 quedando de esta forma:

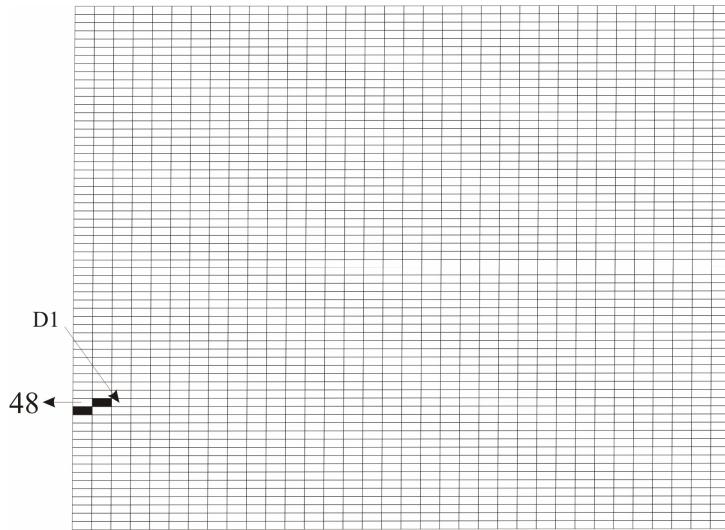


Se vuelve a realizar el mismo procedimiento, ahora quedando el dato en la línea:

$$\#40h / 4 = \#10h = \#16d$$

$$64 - 16 = 48 \rightarrow$$
 coordenada del grob

Cuando se compara D1 con 48, D1 es mayor que el dato, entonces D1 disminuye 34 posiciones e imprime el dato y luego se incrementa en uno, quedando de la siguiente forma:



Para preparar el grafico, se dibuja primero un GROB de 131x64 vacio, y se lo tira en el stack (nivel 1). Cuando el programa comienza busca en D1 donde esta posicionada la primer direccion del nivel 1 donde esta hubicado el GROB.

Diagrama de flujo

3

RPL

```
::
CLOSEUART
RECLAIMDISP
TURNMENUOFF
ABUFF

#E
#1
MAKEGROB

DROP
CODEM
SAVE
C=DAT1 A
D1=C
D1=D1+ 5      %SALTO PROLOGO
D1=D1+ 5      %SALTO SIZE
D1=D1+ 5      %SALTO NUMERO DE FILAS
D1=D1+ 5      %SALTO NUMERO DE COLUMNAS
C=D1 A
R0=C A
LC 22         %CONTADOR DE datos (34d=22h)
```

```

D=C B %CONTADOR DE FIN DE LINEA

*PRINCIPAL
%-----TRANSMISION-----
$1BD0100 %HACE D1= 0010D QUE ES EL ENTER POINT DE
%VELOCIDAD
C=DAT0 B %CARGA EL CONTENIDO DE LA DIRECCION DEL SETEO
CBIT=0 0 %SETEA A 1 EL BIT 0
CBIT=1 1 %SETEA A 1 EL BIT 1
CBIT=1 2 %SETEA A 1 EL BIT 2
DAT0=C B %MUEVE EL VALOR SETEADO A LA DIRECCION DEL %EP
$1B01100 %HACE D1= 00110 UART INTERRUPT REGISTER
C=DAT0 B
CBIT=1 3 %HABILITA EL RECV BUFFER Y EL WIRE ENABLE
%SERIAL
DAT0=C B %MUEVE EL VALOR A LA DIRECCION
*BUCLE
$1B11100 %SE FIJA SI EL BIT 0 DE LA DIRECCION
%SE FIJA EN LA DIRECCION 00111 SI EL BIT 0
%00111 ESTA ACTIVO, EN ESE CASO ES PORQUE
%EL BUFFER TIENE UN DATO

C=DAT0 B
?CBIT=0 0
GOYES BUCLE
$1B41100 %REGISTRO 00114
%TRAIGO DOS NIBLES 00114 Y 00115
%SALVO EL STACK

A=0 W
A=DAT0 B

ASRB B %SHIFT DERECHA UN BIT
ASRB B %SHIFT DERECHA UN BIT
A=-A B %HAGO: (64-POSICION) =LINEA
A=A+64 B

LC 00022 %carga 34
GOSBVL =MUL# %(#03991) B[A] = A[A] * C[A]
C=R0 A %CARGA EL PRINCIPIO DEL BODY DEL GROB
B=B+C A
C=D1 A %CARGA EL VALOR DE LA POSICION DEL ULTIMO
%NIBBLE GUARDADO
?C>=B A
GOYES DECREMENTO
GOSUBL INCREMENTO %Salta a la subrutina INCREMENTO
GOTO LINEA
*DECREMENTO
C=C-34 A
?C<B A
GOYES DECREMENTO
*LINEA
%-----RECOMPONER EL DATO E IMPRIMIRLO-----
D1=C
LC 01
DAT1=C B %imprime el dato en pantalla
D1=D1+ 1
D=D-1 B %FIN DE CADENA
?D=0 B

```

```

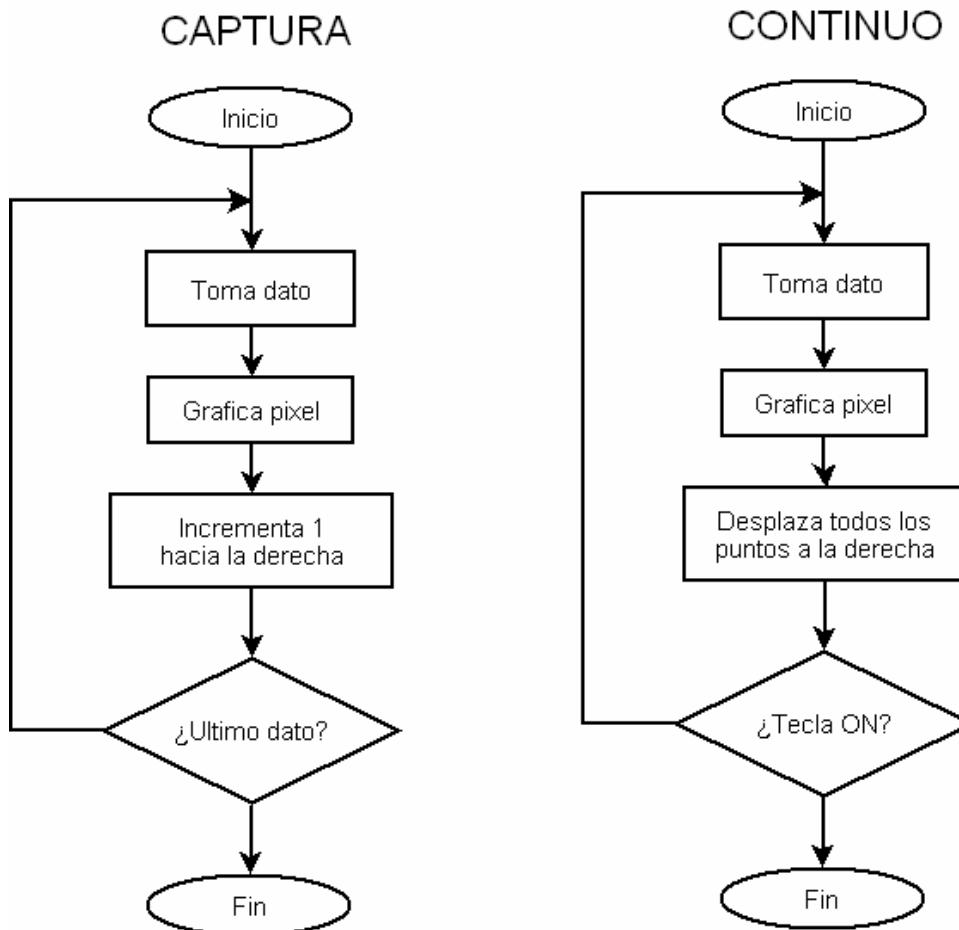
GOYES ON
GOLONG PRINCIPAL
*ON
%-----rutina de tecla ON para salir-----
ST=0 15
*LOOP1
GOSBVL =CINRTN
?CBIT=0 15
GOYES LOOP1
ST=1 15
GOSUB REINT
LOADRPL
*REINT
RTI
*INCREMENTO
?C>=B A
RTNYES
C=C+34 A
GOTO INCREMENTO

ENCODE
DROP
TURNMENUON
RECLAIMDISP
;

```

Desplazamiento lateral

Para realizar el desplazamiento lateral de la pantalla y de esta forma ir avanzando en el muestreo de datos, se diseño una subrutina que reemplaza en el programa anterior de captura el incremento de 1 que se hacia para avanzar sobre el nuevo pixel en la pantalla. De esta forma se logra un efecto de grafico dinamico.

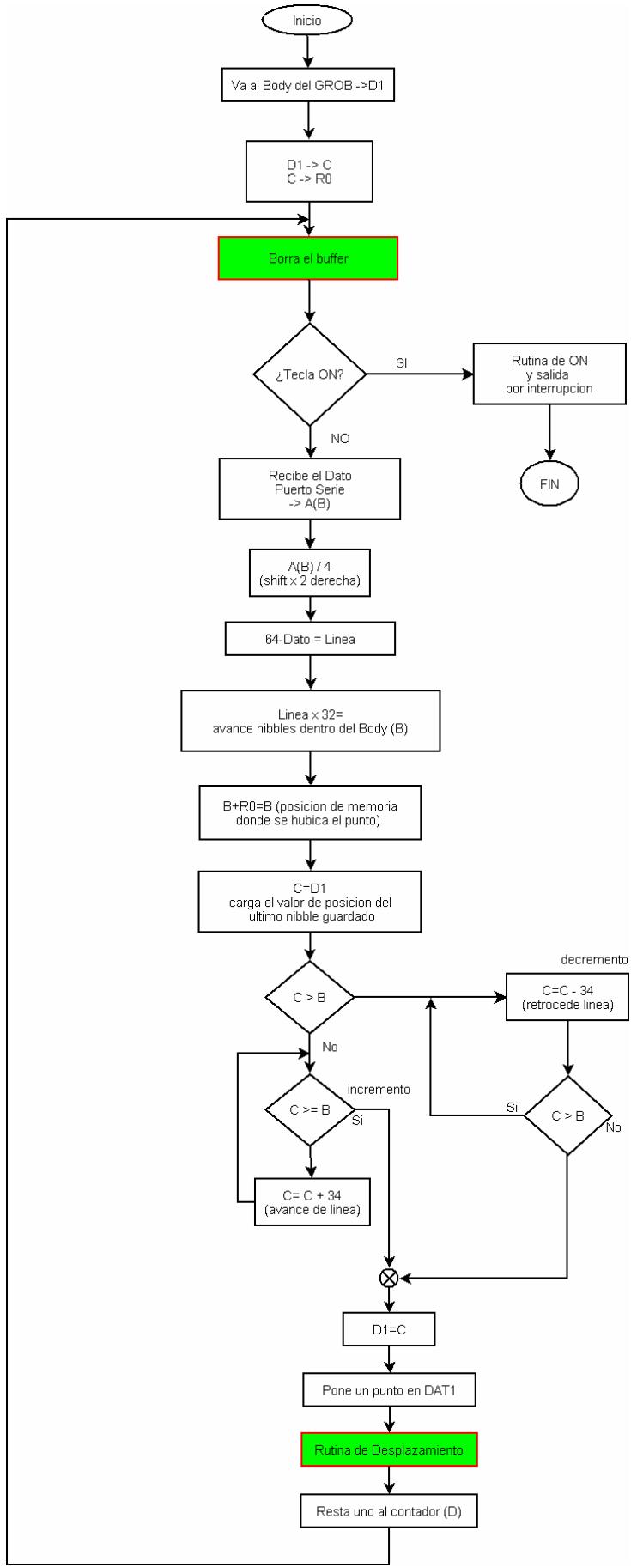


Para realizar esto se aprovecharon los 16 nibbles de los registros de trabajo, que al ser tomados de a 2 veces daria un equivalente a 32 nibbles o 32 pixeles. (casi los 34 de la pantalla).

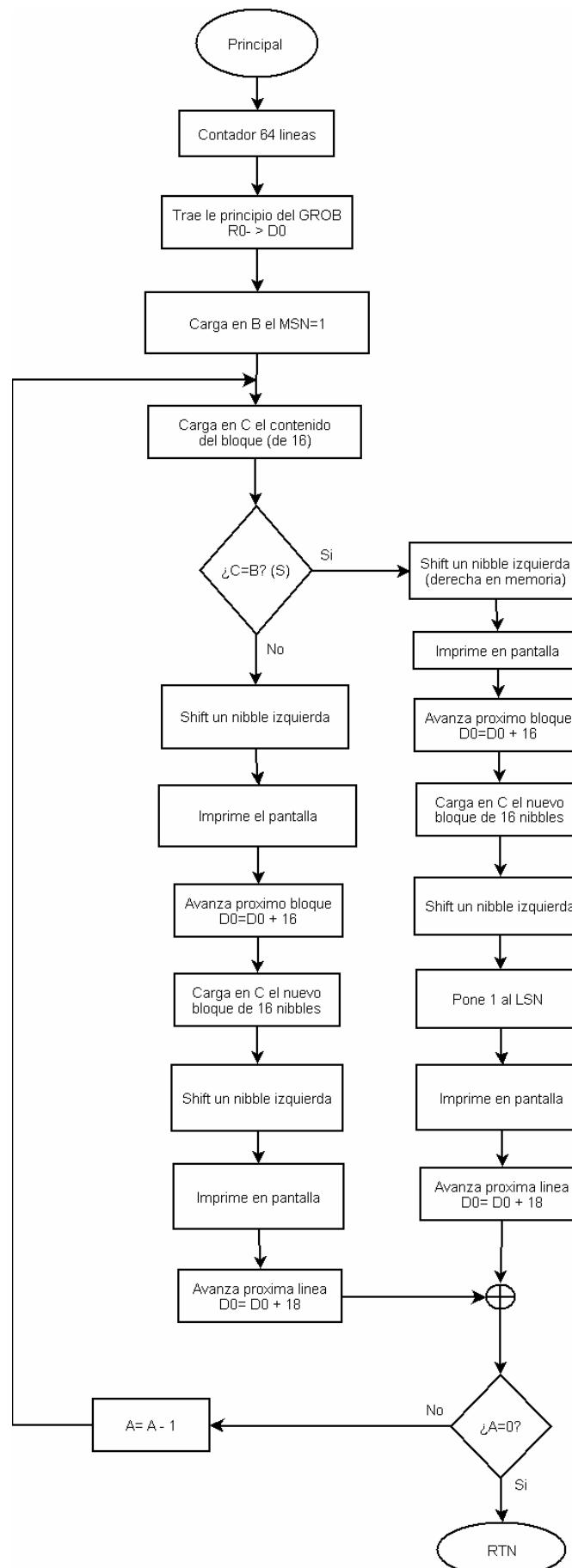
Se toman los primeros 16 nibbles del body y se realiza un desplazamiento (sift) hacia la derecha, si el nibble mas significativo (MSN) (por la inversion de registros) es 0, se pasa al proximo bloque de 16, y se repite la operación. Si el MSN es 1 entonces hay que almacenarlo para ser llevado al proximo bloque y luego realizar el desplazamiento. Esto es porque el microprocesador saturn no posee shft con carry.

Otra de las modificaciones que hay que realizar al programa de captura es la de borrar el buffer cada vez que se lee el puerto serie, dado que los ciclos de programa son mayores que la velocidad de transferencia con la que se trabaja y se produce un delay y un posterior desborde del buffer.

El nuevo programa quedaria de la siguiente forma:



El programa de desplazamiento quedaría de la siguiente forma:



El código queda de la siguiente forma:

RPL

```
:::  
CLOSEUART  
RECLAIMDISP  
TURNMENUOFF  
ABUFF  
  
#E  
#1  
MAKEGROB  
  
DROP  
CODEM  
SAVE  
C=DAT1 A  
D1=C  
D1=D1+ 5 %SALTO PROLOGO  
D1=D1+ 5 %SALTO SIZE  
D1=D1+ 5 %SALTO NUMERO DE FILAS  
D1=D1+ 5 %SALTO NUMERO DE COLUMNAS  
C=D1 A  
R0=C A  
  
*PRINCIPAL  
%TRANSMISION  
%CIERRA PUERTO PARA BORRAR BUFFER  
$1B01100 %HACE D1= 00110 UART INTERRUPT REGISTER  
C=DAT0 B  
CBIT=0 3 %DESHABILITA EL RECV BUFFER DEL WIRE ENABLE SERIAL  
DAT0=C B %MUEVE EL VALOR A LA DIRECCION  
%COMIEZA A RECOGER DATO  
$1BD0100 %HACE D1= 0010D QUE ES EL ENTER POINT DE VELOCIDAD  
C=DAT0 B %CARGA EL CONTENIDO DE LA DIRECCION DEL SETEO  
CBIT=0 0 %SETEA A 1 EL BIT 0  
CBIT=1 1 %SETEA A 1 EL BIT 1  
CBIT=1 2 %SETEA A 1 EL BIT 2  
DAT0=C B %MUEVE EL VALOR SETEADO A LA DIRECCION DEL EP  
$1B01100 %HACE D1= 00110 UART INTERRUPT REGISTER  
C=DAT0 B  
CBIT=1 3 %HABILITA EL RECV BUFFER DEL WIRE ENABLE SERIAL  
DAT0=C B %MUEVE EL VALOR A LA DIRECCION  
*BUCLE %SE FIJA SI EL BIT 0 DE LA DIRECCION  
GOSUBL ON  
$1B11100 %SE FIJA EN LA DIRECCION 00111 SI EL BIT 0  
%00111 ESTA ACTIVO, EN ESE CASO ES PORQUE  
%EL BUFFER TIENE UN DATO  
  
C=DAT0 B  
?CBIT=0 0  
GOYES BUCLE  
$1B41100 %REGISTRO 00114  
%TRAIGO DOS NIBLES 00114 Y 00115  
%SALVO EL STACK  
A=0 W  
A=DAT0 B  
  
ASRB B      %SHIFT DERECHA UN BIT  
ASRB B      %SHIFT DERECHA UN BIT
```

```

A=-A B           %HAGO: (64-POSICION) =LINEA
A=A+64 B
%***** ****
%Para hacer que grafique del 220 al 255 solamente hay que
%sumarle 3 veces mas 64 y sacarle la division de 4 de arriba
%A=A+64 B
%A=A+64 B
%A=A+64 B
%***** ****
LC 00022      %carga 34
GOSBVL =MUL#   %(#03991) B[A] = A[A] * C[A]
C=R0 A          %CARGA EL PRINCIPIO DEL BODY DEL GROB
B=B+C A
C=D1 A %CARGA EL VALOR DE LA POSICION DEL ULTIMO NIBBLE GUARDADO
?C>=B A
GOYES DECREMENTO
GOSUBL INCREMENTO
GOTO LINEA
*DECREMENTO
C=C-34 A
?C>B A
GOYES DECREMENTO
*LINEA
%RECOMONER EL DATO E IMPRIMIRLO
D1=C
LC 01
DAT1=C B      %imprime el dato en pantalla
%desplazamiento lateral
GOSUBL DESPLAZAMIENTO
GOLONG PRINCIPAL

*ON
%***** ****rutina de tecla ON para salir*****
ST=0 15
*LOOP1
GOSBVL =CINRTN
?CBIT=0 15
RTNYES
ST=1 15
GOSUB REINT
LOADRPL
*REINT
RTI
*INCREMENTO
?C>=B A
RTNYES
C=C+34 A
GOTO INCREMENTO
%***** ****rutina de desplazamiento lateral*****
*DESPLAZAMIENTO
LA 40 %CARGA UN CONTADOR DE 64 LINEAS
C=R0 A %TRAE EL PRINCIPIO DEL CUERPO DEL GROB
D0=C
LC 1000000000000000
B=C W
*CUENTA
C=DATO W
?C=B S
GOYES ADD
GOTO SIGUIENTE
*ADD

```

```

CSL W      %SHIFT UN BIT SIN CARRY
DAT0=C W   %IMPRIME LA PRIMER PARTE DEL DATO
D0=D0+ 16  %AVANZO AL PROXIMO MEDIO SECTOR
C=DAT0 W
CBIT=1 0
CSL W
DAT0=C W   %IMPRIME LA SEGUNDA PARTE DEL DATO
D0=D0+ 18  %PASA A LA PROXIMA LINEA (16+18=34)
GOTO CUENTALINE
*SIGUIENTE
CSL W
DAT0=C W
D0=D0+ 16
C=DAT0 W
CSL W
DAT0=C W
D0=D0+ 18

?A=0 B
RTNYES
*CUENTALINE
A=A-1 B  %BAJA EL CONTADOR
GOTO CUENTA

```

```

ENDCODE
DROP
TURNMENUON
RECLAIMDISP
CLOSEUART
;
```

Programas HC908

LCD con 3 lineas de datos

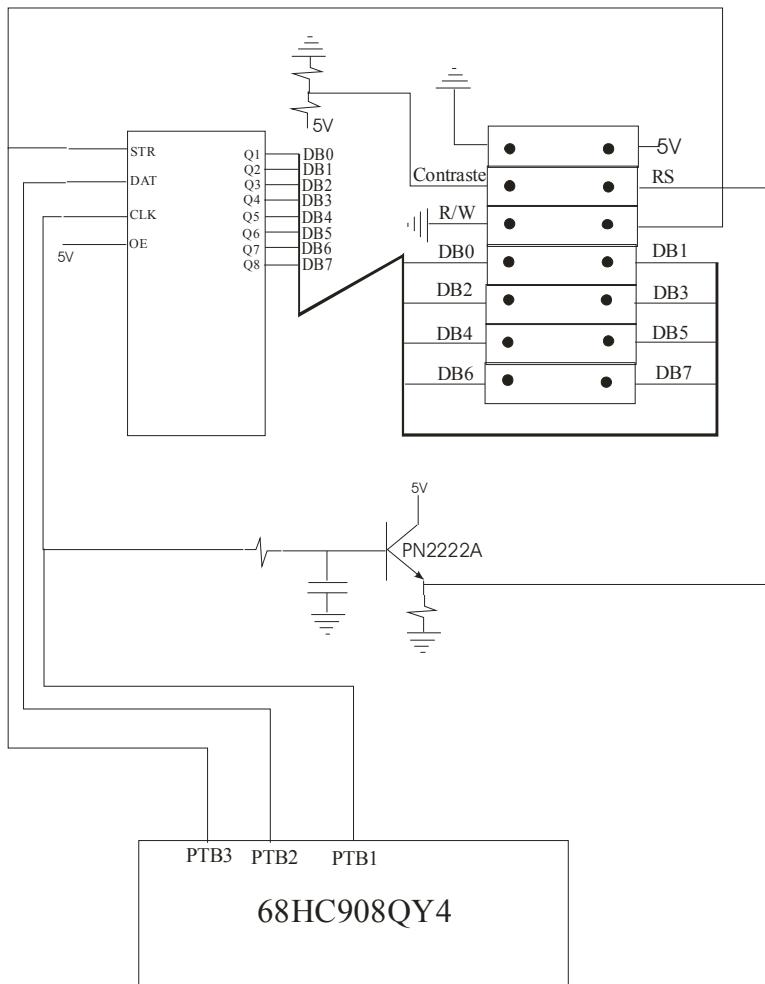
Para poder agregar flexibilidad al dispositivo desde el punto de vista usuario, se necesita incorporar una matriz de entrada de datos y un LCD.

Por los metodos convencionales para realizar esto se hubiese necesitado 8 lineas de datos para la matriz y 10 lineas de datos como minimo para el manejo del display. Dado que para capturar un dato y enviarlo ya se utilizo 2 pines (minimo) del microprocesador, quedan libres 11 pines. Con lo cual no alcanza para. Por eso se recurrio a la utilización de logica adicional con un registro de desplazamiento SIPO. El dispositivo utilizado es: CD4094BE

De esta forma se utilizan 3 lineas de datos para manejar el display y deja 8 lineas mas para el manejo de la matriz.

Por una linea entran los datos en forma serie, con otra linea se controla un reloj para sincronizar el envio de datos y al mismo tiempo el pin RS con la otra se habilita el pin Enable

El circuito quedaria de la siguiente manera:



El programa que se realizo fue el siguiente

```
;***** Seteo inicial *****
org $ee00
ldhx #0000
lda #68
tap
rsp
COM EQU 1
DTASHT EQU 2
CLKSHT EQU 1
STBSHT EQU 3
DATA EQU $80
TEMPA EQU $81
TEMPHX EQU $82
```

```

FLAGS      EQU $83
START      RSP           ;resetea el stack pointer
           mov #$80,$1E ;desconecta el IRQPUD CONFIG 2
           mov #$03,$1F ;dehabilita el modulo COP y habilita el comando
STOP       ;CONFIG 1
           mov #00000111,$05 ;habilito DDRB 1,2,3
           clr $01          ;borro puerto B
;*****Configuraciones de display *****
           jsr DLY50
           mov #$06,DATA
           jsr WCTRL
           jsr DLY50
           mov #$0C,DATA
           jsr WCTRL
           jsr DLY50
           mov #$38,DATA
           jsr WCTRL
           jsr DLY50
           mov #$80,DATA
           jsr WCTRL2
           jsr DLY50
           mov #$02,DATA
           jsr WCTRL
           jsr DLY50
           jsr DLY50
           jsr DLY50
           ldhx #TEXT
PROX      jsr DLY50
           jsr DLY50
           lda 0,x
           beq OTRO
           sta DATA
           bsr WDAT
           aix #1
           bra PROX

OTRO      jsr DLY50
           mov #$C0,DATA
           jsr WCTRL
           ldhx #TEXT2
PROX2     jsr DLY50
           jsr DLY50
           lda 0,x
           beq SALE
           sta DATA
           bsr WDAT
           aix #1
           bra PROX2
SALE      jsr DLY50
           jsr DLY50
           jsr DLY50
           jsr DLY50
           mov #$0E,DATA
           jsr WCTRL
LOOP      bra LOOP

TEXT      DB "Hola!"
           DB 0

TEXT2     DB "Mundo!"
           DB 0
;*****Rutina de DELAY*****
DLY50      sta TEMPA
           sthx TEMPBX
           lda #5
           ldhx #4000t
OUTLP1

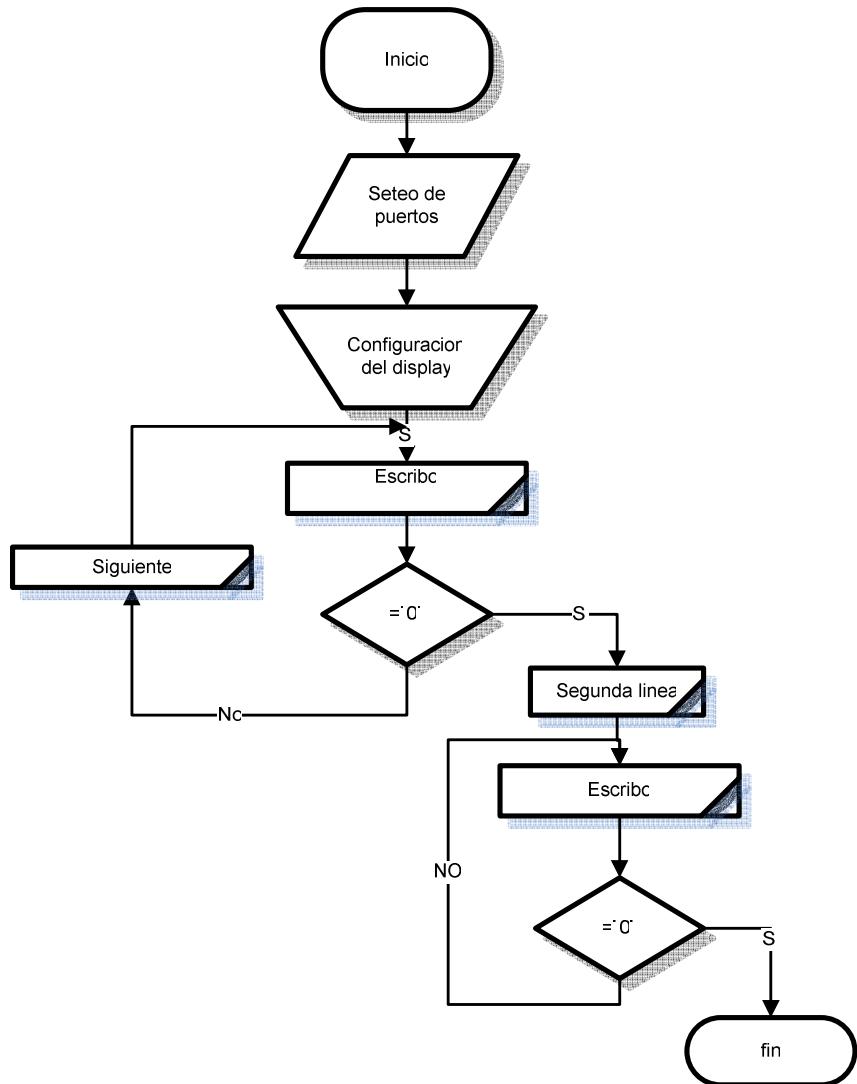
```

```

INNRLP1      aix #-1
              cphx #0
              bne INNRLP1
              deca
              bne OUTLP1
              lda TEMP_A
              ldhx TEMPHX
              rts
;*****
WCTRL        bset COM,FLAGS
              lda DATA
              bsr SHIFT
              bra PULSE
WDAT         bclr COM,FLAGS
              lda DATA
              bsr SHIFT
              bra PULSE
SHIFT        pshx
              ldx #8
NEXT         lsla
              bcs OUT
              bclr DTASHT,$01
              bra CLOCK
OUT          bset DTASHT,$01
CLOCK        brset COM,FLAGS,LCLOCK
              bclr CLKSHT,$01
              bset CLKSHT,$01
              bra SAME
LCLOCK       bclr CLKSHT,$01
              bsr DLY50
              bset CLKSHT,$01
SAME         decx
              bne NEXT
              pulx
              rts
PULSE        bset STBSHT,$01
              bclr STBSHT,$01
              rts
              org $fffe
              dw $ee00
;***** Fin *****

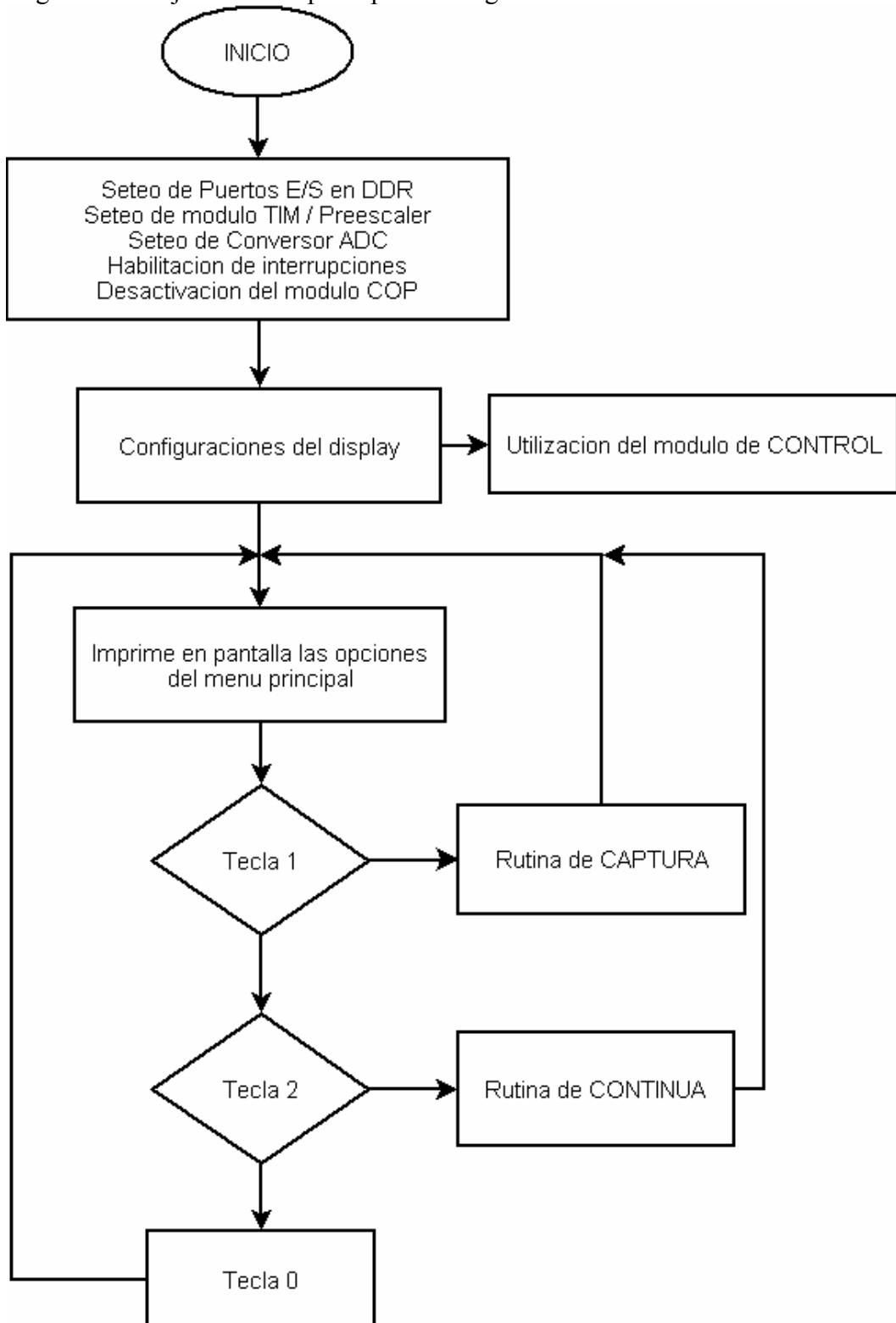
```

El diagrama de flujo general queda de la siguiente manera:



MENU Principal:

El diagrama de flujo del menú principal es el siguiente:



El programa en assembler que se diseño es el siguiente

```
*****
**  

**  

        org $ee00  

        ldx #0000
```

```

        lda #68
        tap
        rsp

COM      EQU 1
DTASHT  EQU 1
CLKSHT  EQU 2
STBSHT  EQU 0
DATA    EQU $80
TEMPA   EQU $81
TEMPHX  EQU $89
FLAGS   EQU $83

START      RSP      ;resetea el stack pointer
           mov #$7F,$05 ;pone ddrb 6-4 como salida teclado y ddr 3-0

como display
           bset 1,$0b ;pone pta 1 como entrada en alto pullup
           bset 3,$0b ;pone pta 3 como entrada en alto pullup
           bset 4,$0b ;pone pta 4 como entrada en alto pullup
           bset 5,$0b ;pone pta 5 como entrada en alto pullup
           mov #$70,$01 ;pone 1 en la salida del ptb 4,5,6
           bset 0,$1f ;cop
           mov #$00,$87
           mov #$01,$1d ;habilita interrupciones
           mov #$40,$1e

           mov #$10,$20 ;seleccion preescaler,Tstop y Trst
           mov #$10,$25 ;seteo de ouputut compare
           mov #$1,$23 ;definimos el tmohd
           mov #$2b,$24 ;def el tmndl
           mov #$0,$26 ;definimos el TCH0H
           mov #$96,$27 ;def el TCH0L

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxDIPLAYxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxx
;*****Configuraciones de display *****
jsr DLY50
mov #$06,DATA
jsr WCTRL

jsr DLY50
mov #$0C,DATA
jsr WCTRL

jsr DLY50
mov #$38,DATA
jsr WCTRL

jsr DLY50
mov #$80,DATA
jsr WCTRL

jsr DLY50
mov #$02,DATA
jsr WCTRL

jsr DLY50
jsr DLY50
jsr DLY50

*****
*****MENU PRINCIPAL*****
PRO      ldhx #TEXTO1
PROX     jsr DLY50
         jsr DLY50

```

```

        lda 0,x
        beq OTRO
        sta DATA
        jsr WDAT
        aix #1
        bra PROX

OTRO          jsr DLY50
               mov #$C0,DATA
               jsr WCTRL

               ldhx #TEXTO2
PROX1         jsr DLY50
               jsr DLY50
               lda 0,x
               beq SALE1
               sta DATA
               jsr WDAT
               aix #1
               bra PROX1

SALE1         jsr DLY50
               mov #$C0,DATA
               jsr WCTRL
               jsr DLY50
               jsr DLY50
               jsr DLY50
               jsr DLY50

               jsr DLY50
               mov #$01,DATA
               jsr WCTRL

               ldhx #TEXTO3
PROX2         jsr DLY50
               jsr DLY50
               lda 0,x
               beq SALE2
               sta DATA
               jsr WDAT
               aix #1
               bra PROX2

SALE2         jsr DLY50
               jsr DLY50
               jsr DLY50
               jsr DLY50
               jsr DLY50
               mov #$0E,DATA
               jsr

```

***** Selección de tecla *****

```

PRINCIPAL      mov #$00,$87
               jsr TECLADO
               lda $87
               beq PRINCIPAL
               cmp #$01
               beq CAPTURA1
               cmp #$02
               beq CONTINUAL
               cmp #$10
               beq menu
               jmp PRINCIPAL
menu          jsr DLY50
               mov #$01,DATA
               jsr WCTRL

```

```

        jmp PRO
CAPTURA1      jmp CAPTURA ;porque no da el offset
CONTINUA1     jmp CONTINUA ;porque no da el offset
*****
*
***** TEXTOS *****

TEXT01         DB '1_CAPT'
               DB 0
TEXT02         DB '2_CONT'
               DB 0
TEXT03         DB '0_MENU'
               DB 0
TEXT04         DB 'CAPTURANDO'
               DB 0
TEXT05         DB '0)FREC:'
               DB 0
TEXT06         DB '1)50Hz'
               DB 0
TEXT07         DB '2)785Hz'
               DB 0
TEXT08         DB '3)12,5KHz'
               DB 0
TEXT09         DB '4).21MHz'
               DB 0
TEXT010        DB 'Ok'
               DB 0
*****
*****
$Include 'CONTROL.inc'
$Include 'CAPTURA.inc'
$Include 'CONTINUA.inc'
$Include 'TECLADO.inc'
$Include 'SERIAL.inc'

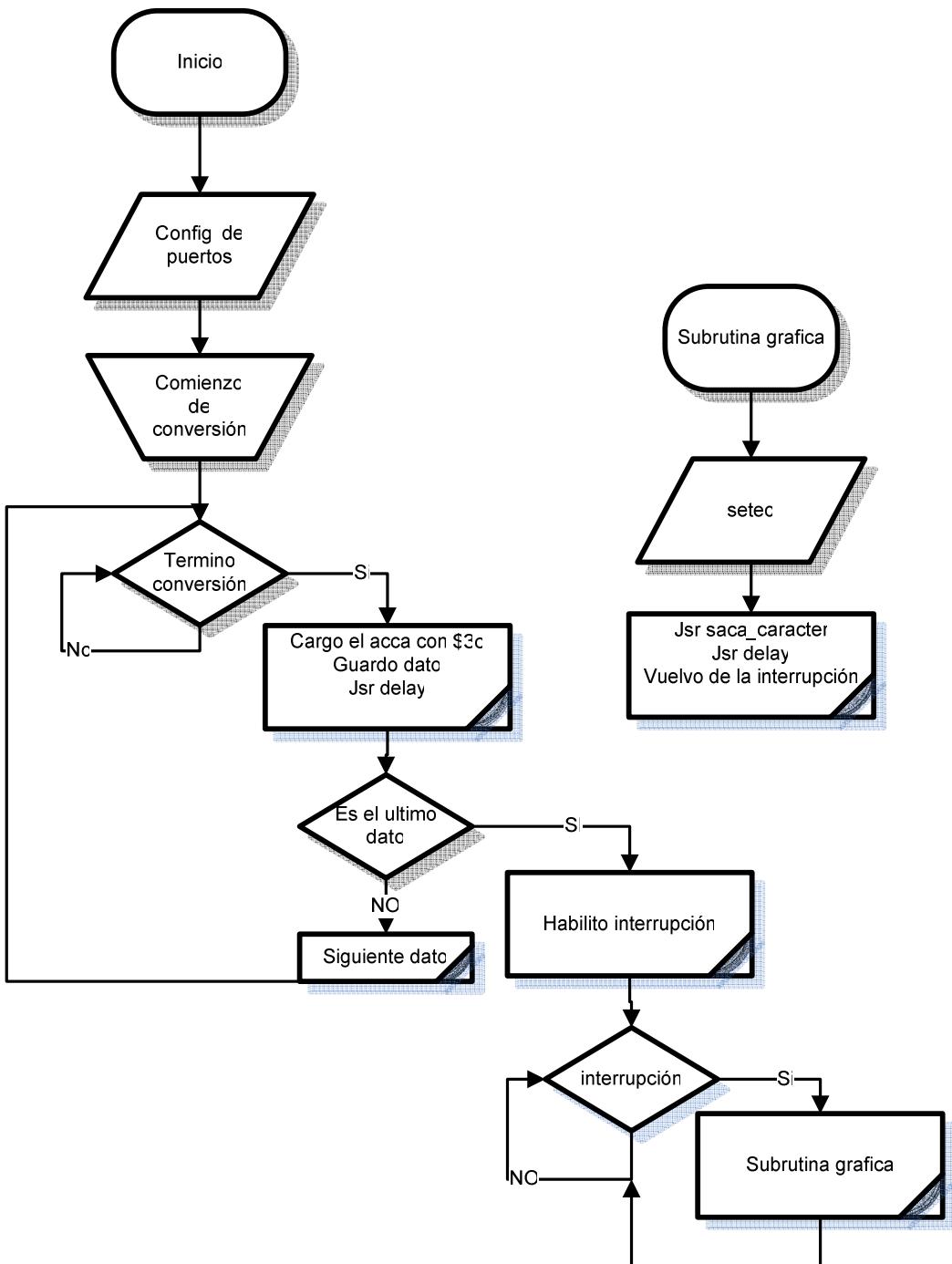
        org $ffffa
        dw  INTERUPCION
        dw $ee00
        dw $ee00

;xxxx Fin xxx

```

Archivos incluye

CAPTURA.INC



CAPTURA

```

caracter EQU $9D
cuenta_bit EQU $9E
auxiliar EQU $9F
ptb EQU $01
pta EQU $00
bset 7,$05 ;habilito como salida ptb2
bclr 7,$01 ;pongo en bajo el ptb7

;*****DISPLAY*****
OPCIONESCAP1      mov #$01,DATA
                    jsr WCTRL
                    ldhx #TEXTO6

```

```

PROXT1      jsr DLY50
            jsr DLY50
            lda 0,x
            beq OPCIONESCAP2
            sta DATA
            jsr WDAT
            aix #1
            bra PROXT1
OPCIONESCAP2  mov #$C0,DATA
            jsr WCTRL
            ldhx #TEXTO7
PROXT2      jsr DLY50
            jsr DLY50
            lda 0,x
            beq OPCIONESCAP3
            sta DATA
            jsr WDAT
            aix #1
            bra PROXT2
OPCIONESCAP3  jsr DLY50
            jsr DLY50
            jsr DLY50
            jsr DLY50
            mov #$01,DATA
            jsr WCTRL
            ldhx #TEXTO8
PROXT3      jsr DLY50
            jsr DLY50
            lda 0,x
            beq OPCIONESCAP4
            sta DATA
            jsr WDAT
            aix #1
            bra PROXT3
OPCIONESCAP4  mov #$C0,DATA
            jsr WCTRL
            ldhx #TEXTO9
PROXT4      jsr DLY50
            jsr DLY50
            lda 0,x
            beq MENUCAPT
            sta DATA
            jsr WDAT
            aix #1
            bra PROXT4

MENUCAPT    jsr DLY50
            jsr DLY50
            jsr DLY50
            jsr DLY50
            mov #$01,DATA
            jsr WCTRL
            ldhx #TEXTO5
PROXC1      jsr DLY50
            jsr DLY50
            lda 0,x
            beq TECLADOCAPTURA
            sta DATA
            jsr WDAT
            aix #1
            bra PROXC1

;*****Opciones de teclado*****
TECLADOCAPTURA  mov #$00,$87      ;borra el teclado
            jsr TECLADO
            lda $87
            beq TECLADOCAPTURA

```

```

        cmp #$01
        beq FREC1
        cmp #$02
        beq FREC2
        cmp #$03
        beq FREC3
        cmp #$04
        beq FREC4
        jmp OPCIONESCAP1 ;cualquier otra opcion muestra el menu

FREC1      mov #$FF,$F3
            mov #$FF,$F4
            bra CAPT
FREC2      mov #$0F,$F3
            mov #$FF,$F4
            bra CAPT
FREC3      mov #$00,$F3
            mov #$FF,$F4
            bra CAPT
FREC4      mov #$00,$F3
            mov #$0F,$F4
;*****Programa de captura*****
;*****Captura
            jsr DLY50
            jsr DLY50
            mov #$01,DATA
            jsr WCTRL
            ldhx #TEXTO10
PROXCAPTURA jsr DLY50
            jsr DLY50
            lda 0,x
            beq INICIOCAPTURA
            sta DATA
            jsr WDAT
            aix #1
            bra PROXCAPTURA
INICIOCAPTURA
;*****inicio del programa *****
;bset 3,$05    ;habilito el puerto b
;bset 3,$01    ;pongo la salida de RS232 a 1
;mov #$20,$3c ; Se setea la conversion ADC continua y el canal cero (PTA0)
;clra
;*****Comienzo de conversion*****
;*****espera
            cli
            mov #$00,$F8
espera      lda $F8
            cmp #1    ;mecanismo para detectar que se realizo la interrupcion
            bne espera
            jmp pro

INTERRUPCION ;comienzo de la interrupcion
            ldhx #$0000
            mov #$01,$F8

            bset 7,$01
inicio      brclr 7,$003c,inicio ; * Mientras el bit que indica la conversion
            ; * completa esta en cero va a seguir en el bucle.
            ldx #$A0
proximo    lda $003e      ; * Cargo en el acumulador el Registro de Dato ADC.
            sta ,x       ;guarda en la proxima direccion donde apunta el indice
            jsr delay
            cmpx #$F2
            incx
            bmi proximo

```

```

;*****GRAFICA*****
otrol          ldx #$A0           ;apunto el indice al primer dato
               lda ,x             ;lo levanto al acumulador
               sta caracter        ;lo grabo en la direccion de
transmision    jsr saca_caracter
               jsr delay           ;genera un retardo entre dato y dato a
enviar         cmpx #$F2           ;se fija si esta al fin de la cadena de
datos          incx
               bmi otrol           ;si no esta al final va al proximo
               bclr 7,$05           ;deshabilito el puerto ptb7
               rti
;*****


;-----  

---  

; SELECCION DE VELOCIDAD DE TRANSMISION  

---  

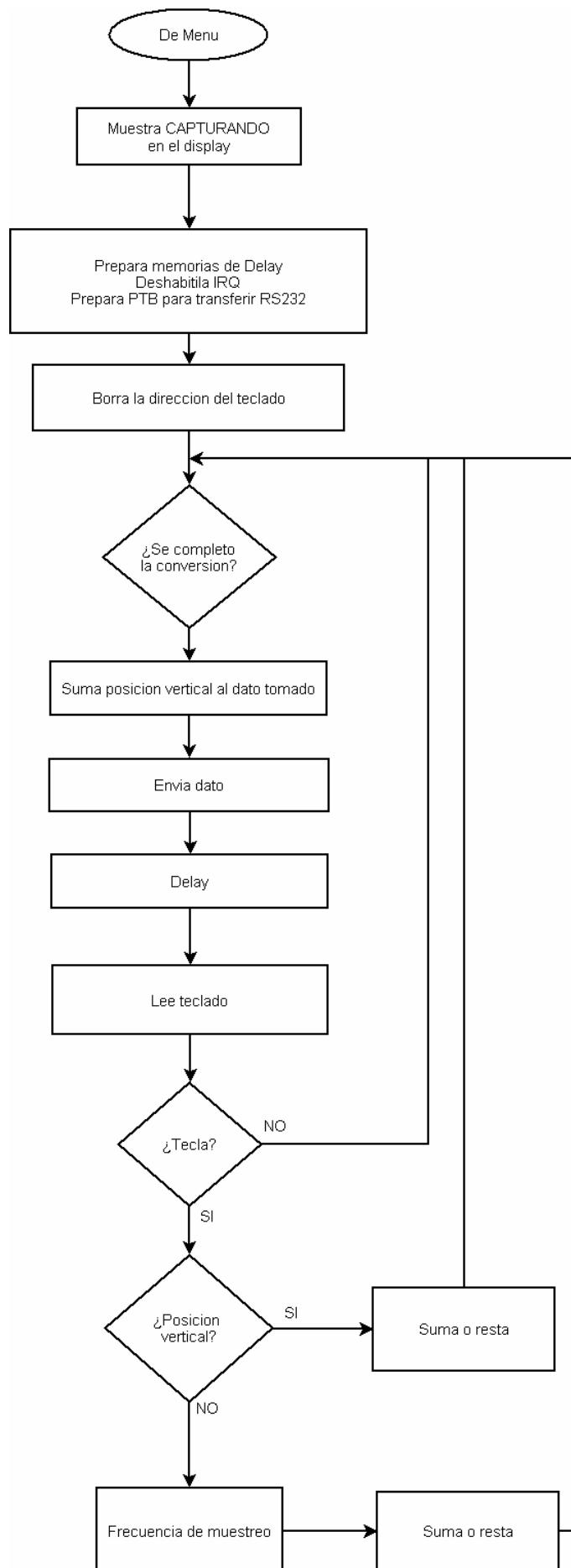
---  

delay_7a        bset 4,$20
               bclr 7,$20
dela           brclr 7,$20,dela
               rts      ;[4]=7a+4 cycles
delay_7b        bset 4,$20
               bclr 7,$25
delal          brclr 7,$25,delal
               rts

;*****rutina de delay de frecuencia de muestreo*****
delay:          sta TEMPA
               sthx TEMPBX
               ldx #$FF
del2           aix #-1
               cphx #0
               bne del2
               lda TEMPA
               ldhx TEMPBX
               rts

```

CONTINUA.INC



CONTINUA

```
;*****DISPLAY*****
        mov #$02,DATA
        jsr WCTRL
        ldhx #TEXTO4
PROXC    jsr DLY50
        jsr DLY50
        lda 0,x
        beq CAPTC
        sta DATA
        jsr WDAT
        aix #1
        bra PROXC
CAPTC

;*****inicio del programa *****
        mov #$00,$F5      ;valor que se suma a la posicion del
dato
        mov #$FF,$F4      ;parte baja del delay
        mov #$00,$F3      ;parte alta direccion del DELAY
        mov #$01,$1d      ;/IRQ
        mov #$40,$1e      ;Desconecta el pullup interno del irq
        cli
;*****Configuracion Puertos y Conversor Analogo Digital
        bset 0,$1f
        bset 3,$05          ;habilito como salida ptb2
        bset 3,$01          ;pongo a 1 el puerto b para salida
RS232    mov #$20,$3c      ;Se setea la conversion ADC continua
        clra
;*****Comienzo de conversion*****
borrateclado    mov #$00,$87      ;borra la direccion de teclado

inicio3     brclr 7,$003c,inicio3 ;* espera conversion en el bucle

        lda $003e    ;* Cargo en el acumulador el Registro de Dato ADC.
        adc $F4      ; sumo el valor de la posicion vertical
        sta $9D      ; guarda el dato en la direccion de envio

        jsr saca_caracter
        jsr delay
;*****Selector de frecuencia por teclado*****
        mov #$00,$87
        jsr TECLADO
        lda $87
        beq inicio3
        cmp #$04
        beq decremento
        cmp #$06
        beq incremento
        cmp #$02
        beq subelinea
        cmp #$08
        beq bajalinea
        cmp #$10      ;tecla "0"
        beq volver

decremento   dec $F3
incremento   inc $F3
subelinea   jmp borrateclado
bajalinea   inc $F4
                jmp borrateclado
                dec $F4

        jmp borrateclado
```

```

        jmp borrateclado
volver           jmp pro

```

CONTROL.INC (para controlar las acciones del LCD con 3 lineas de datos)

```

WDAT          bclr COM, FLAGS
              lda DATA
              bsr SHIFT
              bra PULSE

;*****Rutina de DELAY*****
DLY50          sta TEMPA
              sthx TEMPHX
              lda #5
OUTLP1         ldhx #4000t
INNRLP1        aix #-1
                cphx #0
                bne INNRLP1
                deca
                bne OUTLP1
                lda TEMPA
                ldhx TEMPHX
                rts
;*****


WCTRL         bset COM, FLAGS
              lda DATA
              bsr SHIFT
              bra PULSE

SHIFT         pshx
              ldx #8
NEXT          lsla
              bcs OUT
              bclr DTASHT,$01
              bra CLOCK

OUT           bset DTASHT,$01

CLOCK         brset COM, FLAGS, LCLOCK
              bclr CLKSH,$01
              bset CLKSH,$01
              bra SAME

LCLOCK        bclr CLKSH,$01
              bsr DLY50
              bset CLKSH,$01

SAME          decx
              bne NEXT
              pulx
              rts

PULSE         bset STBSHT,$01
              bclr STBSHT,$01
              rts

```

TECLADO.INC

```

TECLADO        mov #$00,$84    ;conta 4
              lda #$77

```

```

loop1

    rola
    sta $85
    and #$70
    ora #$0f
    sta $01
    jsr entrada
    inc $84
    lda $84
    cmp #$04
    beq salidateclado
    lda $85
    jmp loop1
salidateclado    rts
;xxxxxxxxxxxxxx Subrutina xxxxxxxxxxxxxxxxxxxxxxxx

entrada

    lda $00
    sta $86
    and #$3A
    cmp #$3A ;0 por 1
    bne decod
    rts
;xxxxxxxxxxxxxx Decodificacion xxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxx Fila xxxxxxxxxxxxxxxxxxxxxxxx

decod

    brclr 5,$86,pfi
    brclr 4,$86,sfi
    brclr 3,$86,tfi
    brclr 1,$86,cfi

;xxxxxxxxxxxxxx Columna xxxxxxxxxxxxxxxx

pfi      brclr 6,$85,numeral
         brclr 5,$85,cero
         brclr 4,$85,aste
sfi      brclr 6,$85,nueve
         brclr 5,$85,ocho
         brclr 4,$85,siete
tfi      ;brclr 7,$85,f2
         brclr 6,$85,seis
         brclr 5,$85,cinco
         brclr 4,$85,cuatro
cfi      brclr 6,$85,tres
         brclr 5,$85,dos
         brclr 4,$85,uno

;xxxxxxxxxxxxxx Tecla seleccionada xxxxxxxxxxxxxxxx

uno       mov #$01,$87
         jmp delay2

dos       mov #$02,$87
         jmp delay2

tres     mov #$03,$87
         jmp delay2

cuatro   mov #$04,$87
         jmp delay2

cinco   mov #$05,$87
         jmp delay2

```

```

seis      mov #$06,$87
          jmp delay2

siete     mov #$07,$87
          jmp delay2
ocho      mov #$08,$87
          jmp delay2
nueve    mov #$09,$87
          jmp delay2
cero      mov #$10,$87
          jmp delay2
aste      mov #$2A,$87
          jmp delay2
numeral   mov #$23,$87
          jmp delay2
          clr $88
incre     inc $88
          lda #$ff
          cbeq $88,sigue
          jmp incre
sigue    rts
delay2
          clr $88
incre2   inc $88
          lda #$ff
          cbeq $88,sigue2
          jmp incre2
sigue2   rts

```

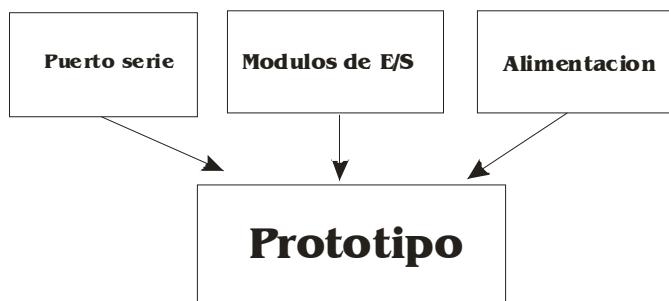
El SERIAL.INC fue el primero que se explico al comienzo del capitulo.

Hardware y puesta en marcha

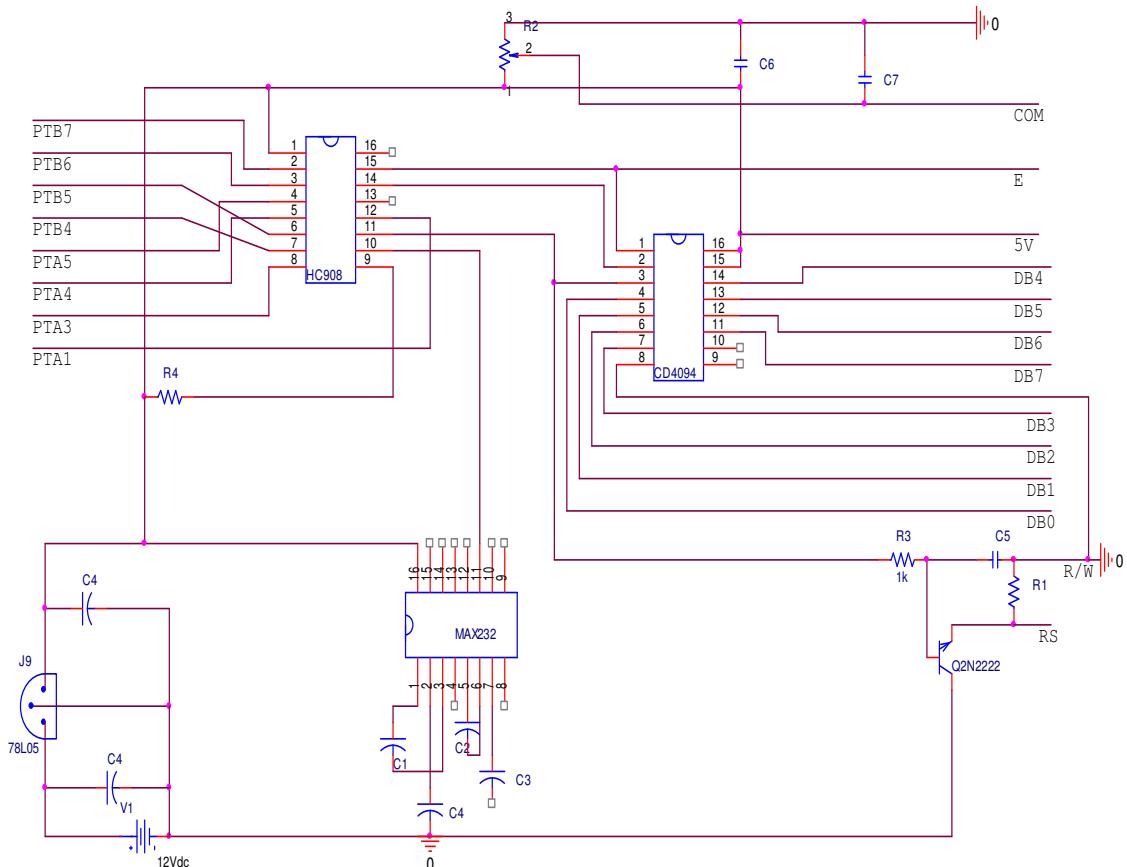
Para diseñar todo el hardware se fue haciendo todo por partes probandolo en una placa de prueba protoboard.

Se trabajo sobre tres etapas principales de hardware:

- Comunicación serie RS232 (MAX232).
- Manejo de LCD con tres líneas de datos y teclado.
- Alimentacion.



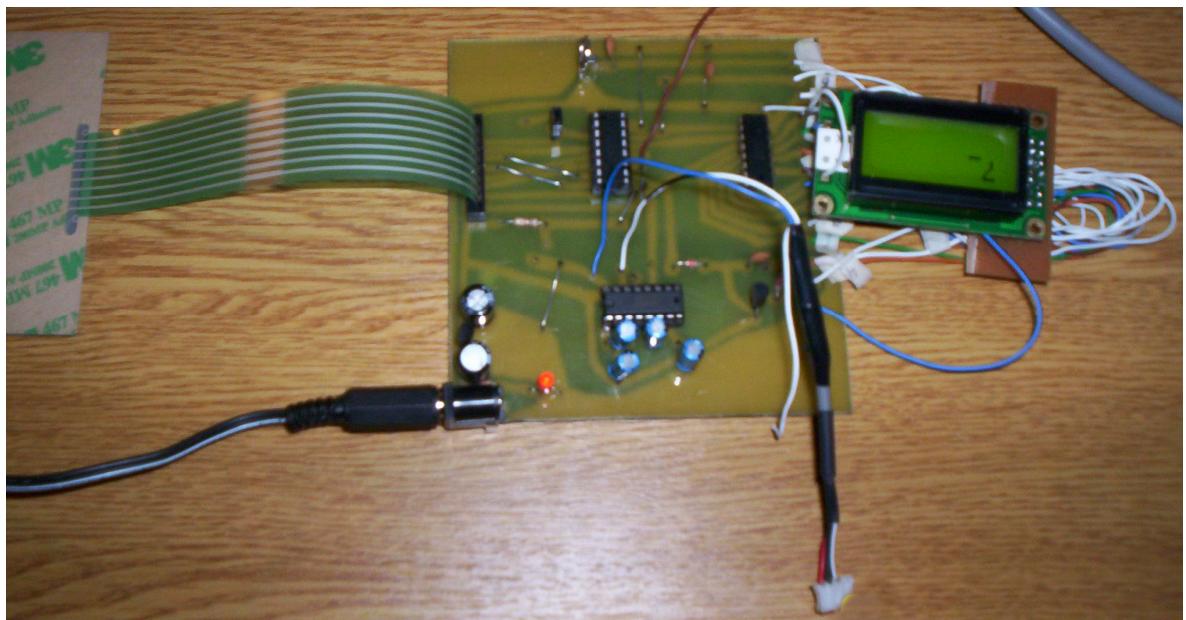
Esquematico:



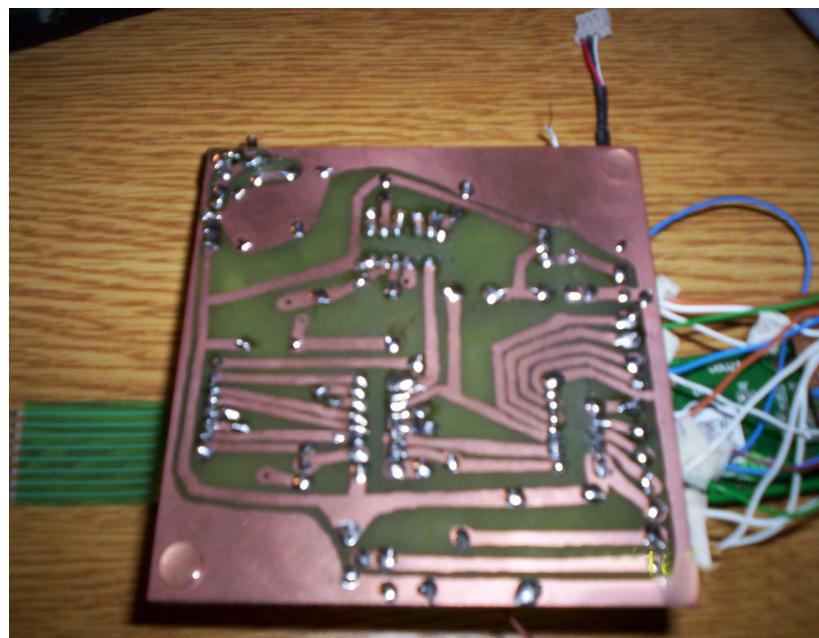
La placa final

que se obtuvo fue la siguiente:

Frente:



Atrás:



Se realizaron los siguientes experimentos:

Se realizo una prueba de medición de la carga de un capacitor de $10\mu F$ en serie con una resistencia de 330Ω y se obtuvo el siguiente grafico:



El tiempo estimado de envío+grafica es de unos 45 a 50 segundos.
Se tomaron distintas medidas y se calculó el error cometido en todas las mediciones que fue de +/-1 pixel teniendo en cuenta que el fondo de escala sería 255 entonces el error relativo que se obtuvo fue de 0,39%

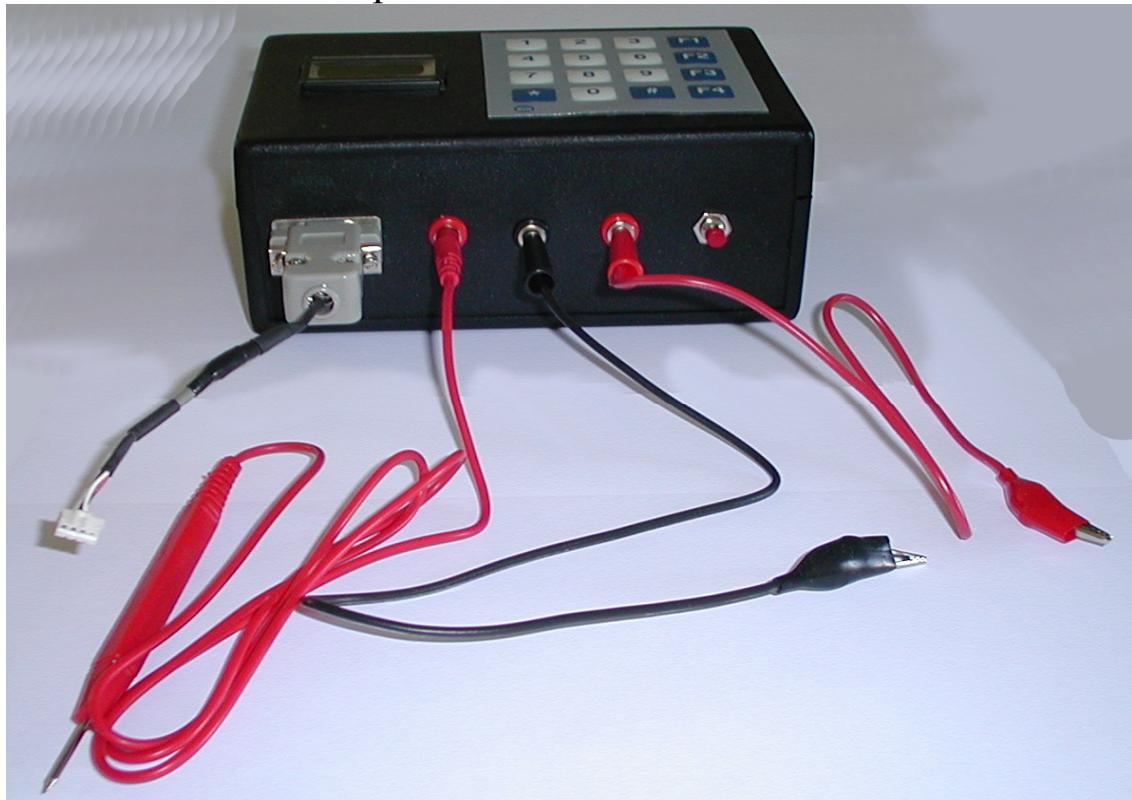
En el caso del modo continua y modo captura a gran velocidad el error aumentó teniendo una incertidumbre extra de la posición de la línea para casos continuos. Esto se debe a que el programa de graficación de la calculadora busca un dato menor, si encuentra uno mayor le suma 34, en caso contrario le resta y nunca encuentra el dato exacto, por lo tanto la curva fluctúa entre +/-1 por sobre la línea real, esto implicaría otro 0,39% a fondo de escala, si a esto le sumamos el error anterior estaría dando un error total de 0,78%

Cuando se toman valores cercanos al nivel de tierra el error aumenta debido a el acople de ruido de nivel de referencia. En algunos casos llegándose a tener hasta un error de +/-5 pixeles, esto daría un error relativo de 1,96%. En las notas de mejoras que se pretenden hacer al dispositivo están algunas de las soluciones propuestas para resolver este inconveniente.

Manual del Usuario:

Introducción: En este capítulo se presenta el manual de usuario del graficador, el cual incluye una descripción de sus características y la guía de como utilizar el software.

Información sobre el producto:



Requisitos:

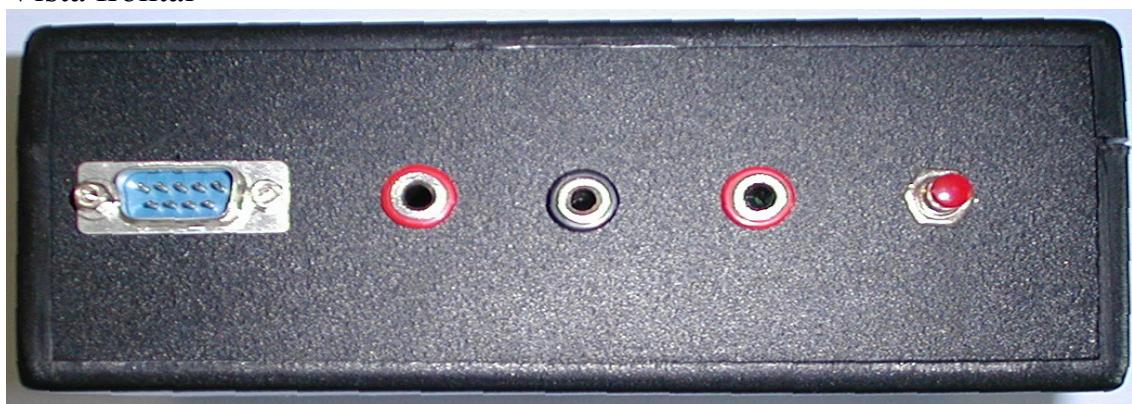
Para el funcionamiento del graficador se necesita una calculadora graficadora con las siguientes características:

Helett Packard modelos: 48SX, 48G, 48GX, 48G+, 49

Las distintas distribuciones que se haran del software dependen del modelo para su compilación.

Conectores y controles del graficador:

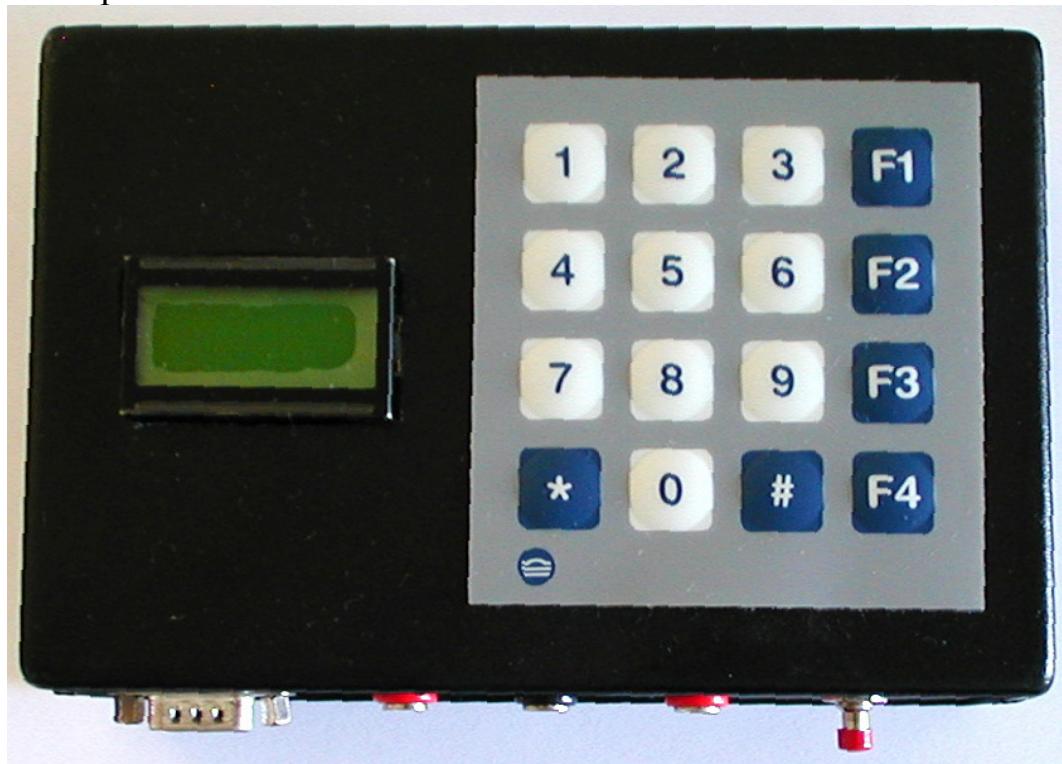
Vista frontal



Descripción de izquierda a derecha:

Conecotor	Descripcion
RS232	Puerto de comunicación serie
Chanel	Toma de datos de ADC
GND	Tierra
Escalon	Generador de funciones escalon
Disparador	Disparador del modo CAPTURA

Panel Superior



Descripción:

Dispositivo	Descripción
LCD	Display de monitoreo / menues
Matriz	Teclado para comandar menues y opciones.

KIT de cables:



Descripción:

Cable	Descripción
RS232	Conector entre el graficador y la calculadora
Punta	Punta de medición, para toma de datos
Rojo	Cable simple para conexión escalon
Negro	Cable simple para tierra

Especificaciones: al final del informe

Modos de funcionamiento:

El graficador de señales posee dos modos de funcionamiento. Cuando conecte por primera vez el dispositivo a una fuente de 9V se encenderá el Display indicando dos posibles opciones:

- 1-Captura
- 2-Continua
- 0-Menu

Si desea volver a ver el slide del menú solo aprete la opción 0

Modo Captura.

Cuando seleccione la opción 1 del menú, uds ingresara en el modo CAPTURA. Este modo es el método que se utiliza para graficar señales cortas o estáticas en el tiempo. Se podrá seleccionar entre distintos tipos de frecuencias de interés bastante generales para tomar los datos.

El despliegue de opciones es el siguiente:

- 1)50Hz
- 2)785Hz
- 3)12,5KHz
- 4)0.21MHz

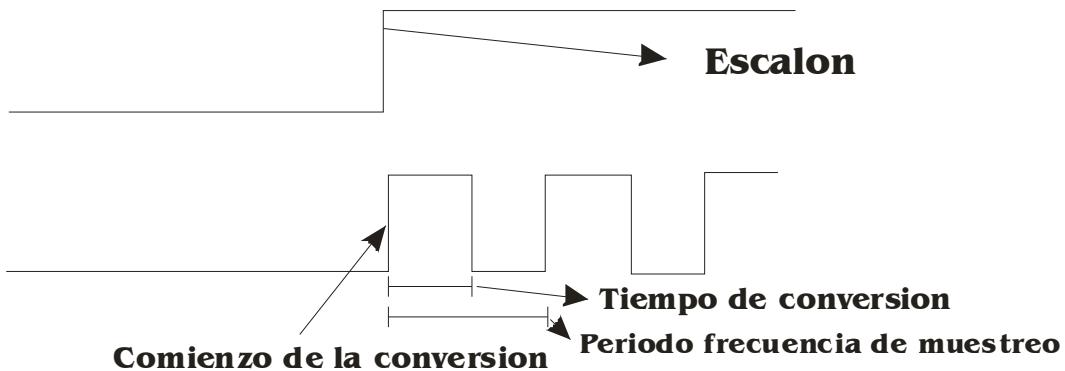
Volviendo a repetir tocando la opcion 0

Una vez seleccionada la frecuencia de muestreo aparecera un cartel “OK” eso significa que el dispositivo esta listo para disparar.

En caso de que se necesite medir por ejemplo la carga de un capacitor, el dispositivo cuenta con un conector que genera escalones. Es la 4 ficha contando de izquierda a derecha. Este mecanismo funciona de la siguiente manera:

En el momento en que se aprieta el disparador, se genera un escalon por el conector 4, inmediatamente se comienza con la toma de datos.

Este mecanismo sirve para poder sincronizar con presicion el inicio de una exitacion.

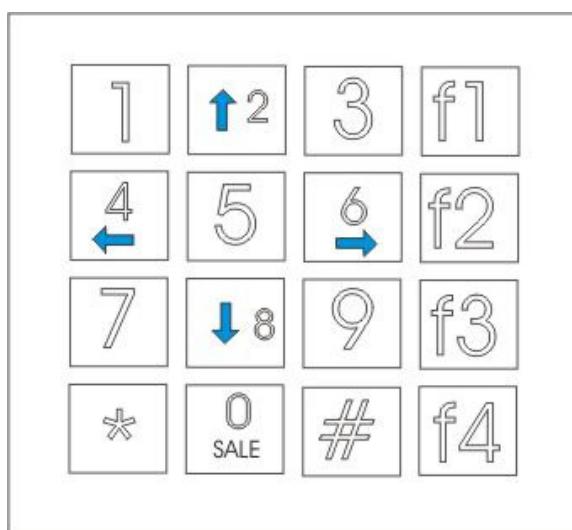


Luego de que se ejecute el disparo, el dispositivo comenzara a transmitir los datos a la calculadora. Volviendo luego al menú principal.

Modo CONTINUA

Si en el menú principal elije la opcion 2 entrara en el modo CONTINUA.

A continuación se detallan las opciones incorporadas en el teclado en este modo:



Las opciones que estan sobre las teclas 4 y 6 sirven para cambiar la frecuencia de muestreo.

Las opciones que estan sobre las teclas 2 y 8 sirven para subir o bajar la grafica en caso de que se desee centrarla.

La opcion 0 sirve para salir y volver al menú principal.

Descripción del software HP

Para el dispositivo HP se cuenta con 3 programas. Dos para el modo de captura de la señal y uno para el modo de Continua.

Programas del modo captura:

Alta resolucion: INICIO

Uno de alta resolucion, bastante flexible para el usuario, donde se puede modificar las caracteristicas graficas directamente modificando las opciones del plot.

Para hacer una captura con este programa se deben seguir los siguientes pasos:

1. Configurar el puerto de comunicaciones de la siguiente manera:

PORT: WIRE

TRANSFER MODE: BINARIO

BAUD: 9600

PARIDAD: NONE

CKSUM: 1

TRASLATE

2. Verificar en la opcion de plot que este encuadrado dentro de los siguientes valores:

Eje X entre 1 y 10

Eje Y entre 0 y 255 (fondo de escala equivalente a 5V)

3. En el directorio principal donde se guardo el programa, ejecutar el programa INICIO y luego presionar el disparador para realizar la captura.

¡Importante!

Desde el momento en que se ejecuta el programa INICIO la calculadora esperara 10 segundos hasta que se efectue el disparo, si en ese periodo no se realizo la comunicación serie, el programa saldra dando un mensaje de error.

Tiempo estimado de la captura+grafica es de 30 a 40 segundos

Baja resolucion: CAPTURA

Este programa permite ver rapidamente una grafica en pantalla sin tener que configurar puertos ni ploter.

El procedimiento para ejecutarlo es el siguiente:

1. Abra el directorio donde se encuentra el programa y ejecute CAPTURA
2. Dispare con el disparador del dispositivo. Obtendra la grafica
3. Una vez realizada la captura sale del programa tocando la tecla ON

Nota

En caso de haber ejecutado previamente el programa INICIO, es recomendable previo a la ejecución de este programa ejecutar el comando CLOSEIO para prevenir que haya quedado abierto el puerto de comunicaciones y se produzca una falla. En caso de producirse falla la forma de desatascar la calculadora es apretando al mismo tiempo las teclas ON+C

Programas de modo continua:

Para realizar una grafica en este modo, solo hay que ejecutar el programa en la calculadora, llamado: CONTINUA y luego seleccionar la opcion en el graficador, los controles de esta opcion son los descriptos anteriormente.

Para salir de este programa se utiliza la tecla ON en cualquier momento.

En caso de que la grafica quede estatica, significa que se detuvo el envio de datos, simplemente vuelva a elegir la opcion en el menu, o salga con la tecla ON.

Mejoras

Dado los tiempos en que se desarrollo este proyecto y su envergadura, era de esperar en un principio que quedaran temas pendientes. Aquí se detallan algunos de estos temas:

Etapa de entrada

Cuando se realizaron las primeras medidas, en modo continua sobretodo, se observaron grandes errores a la hora de medir el nivel de tierra. Esto es producido por un acople de la señal con otras referencias de tierra, en el capitulo 2 en la investigación del error se menciono este problema.

Para solucionar este problema se planteo en las exposiciones intermedias del proyecto la posibilidad de introducir un filtro a base de amplificadores operacionales del tipo mas comun: 741. Y de esta forma poder solucionar este problema.

Etapa de alimentación

En la etapa de alimentación nos encontramos con la dificultad de hacer funcionar correctamente la luz del display. Esto fue debido a que el consumo es muy grande y por lo tanto bajaba el rendimiento general del circuito. Si se pudiera mejorar esta etapa se podria conectar esta opcion.

Programas

Se han pensado y analizado varios posibles programas para implementar a futuro y de esta forma mejorar las prestaciones del prototipo.

Una vez solucionado el problema de la etapa de entrada se podria hacer un programa en el que tocando una tecla se pueda obtener un zoom de la imagen en el modo Continua y asi poder obtener mas detalles de la señal. La forma de implementarlo seria de la misma forma que se hizo el control de posición vertical, pero en este caso en vez de sumar una direccion de memoria habria que multiplicarla.

Otra de las mejoras posibles seria incorporar un sistema que detecte la amplitud de la señal en modo Captura y envie en los dos primeros paquetes por el puerto serie estos datos, luego, la calculadora calibraria con estos dos datos la resolucion de la pantalla y mostraria de esta forma un modo automatico de grafico.

Esto se podria lograr, buscando entre los datos capturados, el mayor y menor, para luego ubicarlos en las primeras direcciones de envio.

Conclusiones

Despues de realizar investigaciones y luchar con los programas, hemos podido conseguir un dispositivo que cumpla minimamente con lo que habiamos supuesto o esperado.

Nos encontramos con muchas dificultades, sobretodo a la hora comunicar los dos microprocesadores de tecnologías tan diferentes. Si bien los conocimientos adquiridos en la materia fueron utiles y facilitaron en gran medida el estudio de un microprocesador totalmente diferente al que veniamos utilizando, no fue facil adaptarse a la simultaneidad del desarrollo en lenguaje de maquina.

De a pequeñas etapas pudimos conseguir los objetivos planteados en el preproyecto y en tiempos inmejorables.

Asimismo creemos que el dispositivo final que obtuvimos tiene muy buenas caracteristicas, de adaptabilidad, incluso para un posible uso en PC. Si bien estamos lejos de un instrumento de medicion de presicion, esta fue una muy buena aproximación a ellos.

APENDICE

Especificaciones mecanicas:

Dimensiones: 110 x 160 x 65 mm

Peso:

Especificaciones electricas:

Rango de entrada de tension: 6 a 35V CC

Toma de datos ADC:

Maxima corriente: +/-25mA

Maxima tension 5.5V

Bibliografias:

-Guia de Usuario HP

-Guide to the Saturn Processor (whith HP48 Applications) by Matthew Mastracci Rev 1b

-HP 48 I/O Technical Interfacing Guide

-HP 48 series Advanced User's reference manual

-técnicas de comunicación serie para microcontroladores de la flia HC908 que no poseen modulo SCI

-“Introduction to Saturn Assembly Lenguage” Gilbert Fernandes

-“Article 5921 of comp.sys.handhelds” Jan Brittenon

-“HP original SASM document for programming the saturn using HP”

-Archivos ayuda de Debu4x

-MC68HC908QY4/D rev 5 Freescale

-AN2438/D Freescale

-Apuntes de la catedra de circuitos digitales y microprocesadores

-Apuntes Conversores A/D D/A catedra Introducción a los sistemas logicos y digitales

-CPU08 Central proces Unit Referente Manual Freescale

-Programacion de interfaces graficas (GUI) en system RPL de Luis Angel Barahona Burgos