KIT DE ENTRADAS Y SALIDAS DIGITALES Y ANALÓGICAS PARA CALCULADORAS CIENTÍFICAS PROGRAMABLES HP48G/GX DE HEWLETT PACKARD

Müller, Hugo Jorge – hmuller@hjm.com.ar – http://www.hjm.com.ar
Facultad Regional Paraná
Universidad Tecnológica Nacional
Paraná (Entre Ríos) – ARGENTINA

Ejes Temáticos: 3 y 7 (Industria y Educación)

Palabras Claves: Calculadoras HP48, Entradas y Salidas Digitales y Analógicas, RS232, Scripts USER-RPL, Programación, Automatización Procesos Laboratorio, MICROCHIP DREAM MACHINE 97.

Resumen: Este accesorio especialmente diseñado para las potentes calculadoras científicas programables **HP48G/GX** de Hewlett Packard [1], proporciona el control sobre 32 salidas digitales, 4 canales DAC de tensión, permite leer 32 entradas digitales y 4 canales ADC de 8 bits. El **HP48KIT** básicamente es un dispositivo esclavo implementado con un potente Microcontrolador PIC16C71 [2] de **MICROCHIP** con una interface serial RS-232 por software que bajo el control de la calculadora HP48 ejecuta una serie de comandos. Estos comandos actúan sobre las salidas ya sea digitales o analógicas realizando diversas operaciones que van desde encender o apagar un bit hasta generar pulsos con niveles y tiempos programables de manera totalmente autónoma, también permiten leer las entradas analógicas o digitales a nivel de byte o de bits individuales.

Los comandos son enviados ya sea desde la línea de comando de la HP48 mediante el uso de teclas pre-programadas o desde un programa que el usuario genera en la calculadora. El mismo se puede escribir en el Lenguaje de programación standard de la HP48 llamado User-RPL [3] o bien, si se desea aumentar la velocidad de ejecución, puede optarse por programar en el Lenguaje SYS-RPL [4] o lenguaje de máquina, pero por ahora utilizaremos solo User-RPL, que además de ser un lenguaje estructurado y muy potente, tiene un control muy estricto sobre los errores y permite una rápida y segura programación de subrutinas de control que nos permitan manejar el **HP48KIT**.

Cabe destacar que este trabajo se realizó en 1997 para participar del **CONCURSO DREAM MACHINE 97 Microcontrolador PIC 16/17 8-bit RISC** organizado

conjuntamente por **MICROCHIP TECHNOLOGY Inc.** (USA) y **CIKA ELECTRÓNICA** (Argentina) y obtuvo en dicho certamen el **PRIMER PREMIO DEL CONCURSO** entre un total aproximado de 35 trabajos a nivel nacional. El premio consistió en un equipo emulador en tiempo real PICMASTER con sus correspondientes probes y accesorios y un programador PICSTART PLUS que fuera entregado por directivos de MICROCHIP.

I. INTRODUCCIÓN

El sistema consiste en un **Kit de entradas y salidas digitales y analógicas** para las potentes calculadoras científicas programables Hewlett Packard modelos HP48G y HP48GX. El kit, de aquí en adelante llamado **HP48KIT** permite mediante el uso de una calculadora HP48, controlar un total de 32 salidas digitales con niveles CMOS +5V., leer 32 entradas digitales con niveles CMOS +5V., controlar 4 canales de salida analógicos (DAC con salida de tensión) entre 0 y 5 V. con una resolución de 8 bits y leer 4 canales de entrada también analógicos (ADC) y de características similares.

II. DESCRIPCIÓN GENERAL DEL SISTEMA Y CARACTERÍSTICAS SOBRESALIENTES

Las salidas se hallan agrupadas de a 8 bits en latchs, los latch 0 y 1 (16 bits en total) pueden controlarse como tri-estados, esto es muy útil para conjuntamente con un grupo de 8 ó 16 entradas, operar como un Bus de Datos y permitir leer o escribir dispositivos como memorias RAM, EPROM u otro tipo de periféricos que normalmente trabajan conectados a un Bus bidireccional.

El HP48KIT es totalmente portátil, su consumo es inferior a los 20 [mA] y fue diseñado para trabajar con 4 pilas alcalinas de 1,5 V. tipo AAA o AA. Esto es posible ya que se utiliza para regular los +5V. que alimentan al mismo un regulador de baja caída (LOW DROP-OUT REGULATOR), el LM2936Z-5 de NATIONAL SEMICONDUCTOR® [5].

El **HP48KIT** se conecta a la calculadora HP48 a través de un cable de 4 hilos al conector serial del la misma, proporcionando este puerto una comunicación RS-232 full-duplex a una velocidad máxima de 9600 Baudios (que es la utilizada por el Kit en el modo Half-Duplex).

La calculadora mantiene el control sobre el **HP48KIT** mediante una serie de 103 comandos que el Kit reconoce y pueden ser enviados a través del canal serie. Así mismo el Kit esta habilitado para responder en aquellos comandos que requieren información del mismo, tal es el caso de lectura de entradas digitales o analógicas. Este set de comandos o instrucciones es muy potente y esta orientado tanto a bits individuales como a nibbles o bytes dependiendo de la instrucción de que se trate.

III. EXPLICACIÓN DEL FUNCIONAMIENTO GENERAL DEL SISTEMA

El set de instrucciones ejecutables en la HP48

Desde el punto de vista la calculadora, explicaremos ahora las funciones de cada grupo de comandos y/o de cada comando en particular. Cabe mencionar que estos nomencladores utilizados para los comandos del HP48KIT fueron elegidos para facilitar su memorización, y no corresponden a comandos existentes en la calculadora, sino que son nombres creados para cada subrutina que va a controlar el Kit.

Comando : **CLRLx** (donde x=0,1,2,3). abreviatura de CLeaR Latch x
Código de operación : 00d a 03d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Poner a 0 los 8 bits de un latch en particular.

Comando : **SETLx** (donde x=0,1,2,3). abreviatura de SET Latch x
Código de operación : 04d a 07d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Poner a 1 los 8 bits de un latch en particular.

Comando : **INVLx** (donde x=0,1,2,3). abreviatura de INVert Latch x
Código de operación : 08d a 11d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Invierte los 8 bits de un latch en particular.

Comando : **WRLx** (donde x=0,1,2,3). abreviatura de Write Latch x
Código de operación : 12d a 15d respectivamente. **Parámetros** : Valor a escribir y No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Escribe los 8 bits de un latch en particular con el parámetro “Valor”.

Comando : **INCLx** (donde x=0,1,2,3). abreviatura de INCrement Latch x
Código de operación : 16d a 19d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Incrementa los 8 bits de un latch en particular.

Comando : **DECLx** (donde x=0,1,2,3). abreviatura de DECrement Latch x
Código de operación : 20d a 23d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Decrementa los 8 bits de un latch en particular.

Comando : **INC01** abreviatura de INCrement latch 0 and 1
Código de operación : 24d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Incrementa los 16 bits de la palabra formada por latch1(MSB) + latch0 (LSB).

Comando : **INC12** abreviatura de INCrement latch 1 and 2
Código de operación : 25d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Incrementa los 16 bits de la palabra formada por latch2(MSB) + latch1 (LSB).

Comando : **INC23** abreviatura de INCrement latch 2 and 3
Código de operación : 26d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Incrementa los 16 bits de la palabra formada por latch3(MSB) + latch2 (LSB).

Comando : **DEC01** abreviatura de DECrement latch 0 and 1
Código de operación : 28d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Decrementa los 16 bits de la palabra formada por latch1(MSB) + latch0 (LSB).

Comando : **DEC12** abreviatura de DECrement latch 1 and 2
Código de operación : 29d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Decrementa los 16 bits de la palabra formada por latch2(MSB) + latch1 (LSB).

Comando : **DEC23** abreviatura de DECrement latch 2 and 3
Código de operación : 30d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Decrementa los 16 bits de la palabra formada por latch3(MSB) + latch2 (LSB).

Comando : **RDLx** (donde x=0,1,2,3). abreviatura de RedD Latch x
Código de operación : 32d a 35d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : Valor leído del latch respectivo. (Ultimo valor enviado) **Función** : Lee el contenido de un latch en particular.

Comando : **RDINx** (donde x=0,1,2,3). abreviatura de ReaD INput x
Código de operación : 36d a 39d respectivamente. **Parámetros** : No. de registro de entrada (implícito en el comando).
Salidas : Valor leído del registro de entrada respectivo. **Función** : Lee el estado de 8 entradas simultáneamente.

Comando : **SWPx** (donde x=0,1,2,3). abreviatura de SWaP nibbles of latch x
Código de operación : 40d a 43d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Intercambia el contenido de los nibbles alto y bajo de un latch en particular.

Comando : **RLCx** (donde x=0,1,2,3). abreviatura de Rotate Left Cyclic latch x
Código de operación : 44d a 47d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Rota el contenido de un latch en particular ciclicamente hacia la izquierda.

Comando : **RRCx** (donde x=0,1,2,3). abreviatura de Rotate Right Cyclic latch x
Código de operación : 48d a 51d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Rota el contenido de un latch en particular ciclicamente hacia la derecha.

Comando : **RVRTx** (donde x=0,1,2,3). abreviatura de ReVeRT latch x
Código de operación : 52d a 55d respectivamente. **Parámetros** : No. de latch (implícito en el comando).
Salidas : ninguna. **Función** : Intercambia los bit 7 y 0, 6 y 1, 5 y 2, 4 y 3 de un latch en particular.

Comando : **TBCDx** (donde x=0,1,2,3). abreviatura de convert To BCD in latch x
Código de operación : 56d a 59d respectivamente. **Parámetros** : "Valor" ser convertido a BCD y No. de latch (implícito en el comando).
Salidas : ninguna.
Función : Convierte el parámetro "Valor" que debe estar en el rango 0-99 a su equivalente BCD y lo deposita en un latch en particular.

Comando : **T7Sx** (donde x=0,1,2,3). abreviatura de convert To 7 Segment in latch x
Código de operación : 60d a 63d respectivamente. **Parámetros** : "Valor" ser convertido a notación 7 segmentos y No. de latch (implícito en el comando).
Salidas : ninguna.
Función : Convierte el parámetro "Valor" que debe estar en el rango 0-15 a su equivalente hexadecimal codificado en 7 Segmentos (b0=A, b1=b, b2=C, b3=d, b4=e, b5=f, b6=g, b7=no usado) y lo deposita en un latch en particular.

Comando : **FBCx** (donde x=0,1,2,3). abreviatura de convert from BCd latch x
Código de operación : 64d a 67d respectivamente. **Parámetros** : No. de registro de entrada (implícito en el comando).
Salidas : "Valor" BCD correspondiente al registro de entrada x.
Función : Lee 8 bits del registro de entrada y los interpreta como 2 dígitos BCD MSB,LSB retornando en "Valor" un numero entre 0-99.

Comando : **NOP** abreviatura de No OPeration
Código de operación : 68d a 79d, 114d, 115d y 124d a 255d. **Parámetros** : ninguno.
Salidas : ninguna. **Función** : No realizan ninguna función, estos códigos son ignorados por el Kit cuando se reciben como comando.

Comando : **PMS01** abreviatura de Pulse Mili Seconds 0 1
Código de operación : 80d **Parámetros** : Tlow : Duración del pulso en milisegundos, Bit : bit que reflejará el pulso.
Salidas : ninguna. **Función** : Genera un pulso de duración programada en milisegundos con estado lógico 0 y luego pasa a estado lógico 1.

Comando : **PMS10** abreviatura de Pulse Mili Seconds 1 0
Código de operación : 81d **Parámetros** : Tlow : Duración del pulso en milisegundos, Bit : bit que reflejará el pulso.
Salidas : ninguna. **Función** : Genera un pulso de duración programada en milisegundos con estado lógico 1 y luego pasa a estado lógico 0.

Comando : **PDS01** abreviatura de Pulse Decimal Seconds 0 1
Código de operación : 82d **Parámetros** : Tlow : Duración del pulso en décimas de segundos, Bit : bit que reflejará el pulso.
Salidas : ninguna. **Función** : Genera un pulso de duración programada en décimas de segundos con estado lógico 0 y luego pasa a estado lógico 1.

Comando : **PDS10** abreviatura de Pulse Decimal Seconds 1 0
Código de operación : 83d **Parámetros** : Tlow : Duración del pulso en décimas de segundos, Bit : bit que reflejará el pulso.
Salidas : ninguna. **Función** : Genera un pulso de duración programada en décimas de segundos con estado lógico 1 y luego pasa a estado lógico 0.

Comando : **PLM01** abreviatura de Pulse Long Miliseconds 0 1
Código de operación : 84d **Parámetros** : THigh, TLow : Duración del pulso en milisegundos 16 bits, Bit : bit que reflejará el pulso.
Salidas : ninguna. **Función** : Genera un pulso de duración programada en milisegundos con estado lógico 0 y luego pasa a estado lógico 1. Tiene una resolución de 16 bits.

Comando : **PLM10** abreviatura de Pulse Long Miliseconds 1 0
Código de operación : 85d **Parámetros** : THigh, TLow : Duración del pulso en milisegundos 16 bits, Bit : bit que reflejará el pulso.
Salidas : ninguna. **Función** : Genera un pulso de duración programada en milisegundos con estado lógico 1 y luego pasa a estado lógico 0. Tiene una resolución de 16 bits.

Comando : **STR01** abreviatura de STRobe 0 1
Código de operación : 86d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Genera un pulso de Strobe o de Clock negativo de corta duración.

Comando : **STR10** abreviatura de STRobe 1 0
Código de operación : 87d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Genera un pulso de Strobe o de Clock positivo de corta duración.

Comando : **SNDL** abreviatura de SEND serially with Low clock
Código de operación : 88d **Parámetros** : "Valor" : valor a enviar, "DataBit" : Bit de datos, "ClockBit" : Bit de clock.
Salidas : ninguna. **Función** : Envía un byte "Valor" serialmente a un dispositivo tipo registro de desplazamiento o SPI que tome los datos durante el descenso del clock.

Comando : **SNDH** abreviatura de SEND serially with High clock
Código de operación : 89d **Parámetros** : "Valor" : valor a enviar, "DataBit" : Bit de datos, "ClockBit" : Bit de clock.
Salidas : ninguna. **Función** : Envía un byte "Valor" serialmente a un dispositivo tipo registro de desplazamiento o SPI que tome los datos durante el ascenso del clock.

Comando : **RCVL** abreviatura de ReCeiVe serially with Low clock
Código de operación : 90d **Parámetros** : "DataBit" : Bit de datos, "ClockBit" : Bit de clock.
Salidas : "Valor" : valor leído del dispositivo serie.

Función : Recibe un byte “Valor” serialmente desde un dispositivo tipo registro de desplazamiento o SPI que actualice los datos durante el descenso del clock.

Comando : **RCVH** abreviatura de ReCeiVe serially with High clock
Código de operación : 91d **Parámetros** : “DataBit” : Bit de datos, “ClockBit” : Bit de clock.
Salidas : “Valor” : valor leído del dispositivo serie.
Función : Recibe un byte “Valor” serialmente desde un dispositivo tipo registro de desplazamiento o SPI que actualice los datos durante el ascenso del clock.

Comando : **CLRBT** abreviatura de CLeaRBiT
Código de operación : 92d **Parámetros** : “Bit”: Bit a borrar.
Salidas : ninguna. **Función** : Coloca un bit determinado a 0.

Comando : **SETBT** abreviatura de SETBiT
Código de operación : 93d **Parámetros** : “Bit”: Bit a setear.
Salidas : ninguna. **Función** : Coloca un bit determinado a 1.

Comando : **INVBT** abreviatura de INVertBiT
Código de operación : 94d **Parámetros** : “Bit”: Bit a invertir.
Salidas : ninguna. **Función** : Invierte el estado de un bit determinado.

Comando : **WRBIT** abreviatura de WRiteBIT
Código de operación : 95d **Parámetros** : “Bit”: Bit a escribir, “Estado” : Estado a escribirle (0 o 1)
Salidas : ninguna. **Función** : Escribe un estado lógico “Estado” a un bit determinado.

Comando : **CLRA** abreviatura de CLeaR All
Código de operación : 96d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Coloca todos los latch en 0.

Comando : **SETA** abreviatura de SET All
Código de operación : 97d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Coloca todos los latch en Ffh.

Comando : **INVA** abreviatura de INVert All
Código de operación : 98d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Invierte el estado de todos los latchs.

Comando : **RDOB** abreviatura de ReaD Output Bit
Código de operación : 99d **Parámetros** : “Bit”: Bit a leer
Salidas : “Estado” : Estado lógico del bit de salida leído.
Función : Leer el estado de un bit de salida (de un latch).

Comando : **TEST_TX**
Código de operación : 100d **Parámetros** : ninguno.
Salidas : ninguno. **Función** : Comando reservado para realizar pruebas de transmisión serial RS-232.

Comando : **TEST_RX**
Código de operación : 101d **Parámetros** : ninguno.
Salidas : ninguno. **Función** : Comando reservado para realizar pruebas de recepción serial RS-232.

Comando : **RDIB** abreviatura de ReaD Input Bit
Código de operación : 103d **Parámetros** : “Bit”: Bit a leer
Salidas : “Estado” : Estado lógico del bit de entrada leído.
Función : Leer el estado de un bit de un registro de entrada.

Comando : **SDACx** (donde x=0,1,2,3) abreviatura de Set DAC x
Código de operación : 104d a 107d **Parámetros** : “Valor” : Valor de 8 bits a enviar al DAC correspondiente, Nro.de DAC (implícito)
Salidas : ninguna. **Función** : Setea un DAC con un valor de 8 bits y actualiza la tensión generada inmediatamente.

Comando : **LDACx** (donde x=0,1,2,3) abreviatura de Load DAC x
Código de operación : 108d a 111d **Parámetros** : “Valor” : Valor de 8 bits a enviar al DAC correspondiente., Nro. de DAC (implícito).
Salidas : ninguna. **Función** : Carga un DAC con un valor de 8 bits sin actualizar el la tensión generada.

Comando : **UPDAC** abreviatura de Update DACs
Código de operación : 112d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Setea todos los DACs con los valores cargados en su registros internos y actualiza las tensiones generadas simultáneamente.

Comando : **SDACS** abreviatura de Set DACS
Código de operación : 113d **Parámetros** : “Valor” : Valor de 8 bits a enviar a todos los DACs
Salidas : ninguna. **Función** : Setea todos los DACs con un valor de 8 bits y actualiza la tensión generada inmediatamente.

Comando : **RADCx** (donde x=0,1,2,3) abreviatura de Read ADC x

Código de operación : 116d a 119d **Parámetros** : Nro. de ADC (implícito en el comando).
Salidas : "Valor": Valor de 8 bits leído del ADC correspondiente. **Función** : Lee un canal ADC de 8 bits.

Comando :**ODL0** abreviatura de Output Disable Latch 0
Código de operación : 120d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Deshabilita las salidas del latch 0, poniéndolas en alta impedancia (triestado).

Comando :**OEL0** abreviatura de Output Enable Latch 0
Código de operación : 121d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Habilita las salidas del latch 0, poniéndolas a baja impedancia.

Comando :**ODL1** abreviatura de Output Disable Latch 1
Código de operación : 122d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Deshabilita las salidas del latch 1, poniéndolas en alta impedancia (triestado).

Comando :**OEL1** abreviatura de Output Enable Latch 1
Código de operación : 123d **Parámetros** : ninguno.
Salidas : ninguna. **Función** : Habilita las salidas del latch 1, poniéndolas a baja impedancia.

Set de instrucciones (soportadas por hardware) del HP48KIT y su código mnemónico

A continuación se expone el Set de instrucciones soportados por hardware, con su correspondiente mnemónico y su valor decimal. Se optó por representación decimal, dado que la calculadora nativamente opera en dicha notación y las conversiones insumen ciclos de máquina inútiles si se trabajase en Hexadecimal o binario puro.

Set de instrucciones HP48KIT:

	0	1	2	3
0	CLRL0	CLRL1	CLRL2	CLRL3
4	SETL0	SETL1	SETL2	SETL3
8	INVL0	INVL1	INVL2	INVL3
12	WRL0 (Valor)	WRL1 (Valor)	WRL2 (Valor)	WRL3 (Valor)
16	INCL0	INCL1	INCL2	INCL3
20	DECL0	DECL1	DECL2	DECL3
24	INC01	INC12	INC23	(INCL3)
28	DEC01	DEC12	DEC23	(DECL3)
32	RDL0 -> Valor	RDL1 -> Valor	RDL2 -> Valor	RDL3 -> Valor
36	RDIN0 -> Valor	RDIN1 -> Valor	RDIN2 -> Valor	RDIN3 -> Valor
40	SWP0	SWP1	SWP2	SWP3
44	RLC0	RLC1	RLC2	RLC3
48	RRC0	RRC1	RRC2	RRC3
52	RVRT0	RVRT1	RVRT2	RVRT3
56	TBCD0 (Valor)	TBCD1 (Valor)	TBCD2 (Valor)	TBCD3 (Valor)
60	T7S0 (Valor)	T7S1 (Valor)	T7S2 (Valor)	T7S3 (Valor)
64	FBC0 -> Valor	FBCD1 -> Valor	FBCD2 -> Valor	FBCD3 -> Valor
68	NOP	NOP	NOP	NOP
72	NOP	NOP	NOP	NOP
76	NOP	NOP	NOP	NOP
80	PMS01 (Tlow,Bit)	PMS10 (Tlow,Bit)	PDS01 (Tlow,Bit)	PDS10 (Tlow,Bit)
84	PLM01 (Thigh,Tlow,Bit)	PLM10 (Thigh,Tlow,Bit)	STR01 (Bit)	STR10 (Bit)
88	SNDL (Valor, DataBit, ClockBit)	SNDH (Valor, DataBit, ClockBit)	RCVL (DataBit, ClockBit) -> Valor	RCVH (DataBit, ClockBit) -> Valor
92	CLRBT (Bit)	SETBT (Bit)	INVBT (Bit)	WRBIT (Bit, Valor)
96	CLRA	SETA	INVA	RDOB (Bit) -> Valor
100	TEST_TX	TEST_RX	NOP	RDIB (Bit) -> Valor

	(Reservado)	(Reservado)		
104	SDAC0 (Valor)	SDAC1 (Valor)	SDAC2 (Valor)	SDAC3 (Valor)
108	LDAC0 (Valor)	LDAC1 (Valor)	LDAC2 (Valor)	LDAC3 (Valor)
112	UPDAC	SDACS (Valor)	NOP	NOP
116	RADC0 -> Valor	RADC1 -> Valor	RADC2 -> Valor	RADC3 -> Valor
120	ODL0	OEL0	ODL1	OEL1
124 al 255	NOP	NOP	NOP	NOP

Como se programa un comando del HP48KIT en la calculadora

Todos los comandos que el **HP48KIT** recibe deben ser enviados a través del puerto serie de la calculadora. Previo a utilizar el Kit, debemos configurar el puerto serie en la HP48. Una manera de hacerlo es tipeando **106.01 TMENU** esto no lleva al menú de configuración y allí debemos configurar lo siguiente en la pantalla de la HP:

IR/Wire : wire
ASCII/binary : binary
baud : 9600
parity : none 0
cksum : 1 translate: 3

Luego de configurado el puerto, se creará una variable del sistema **IOPAR** que contendrá lo siguiente:

{ 9600 0 0 0 1 3}

crear esta variable directamente es otra alternativa para configurar el puerto serie.

Para enviar datos a través del puerto serie, el Sistema operativo de la calculadora, a través de su lenguaje User-RPL, provee al usuario del comando **XMIT**. El mismo envía la cadena de caracteres del nivel 1 de la pila sin utilizar ningún protocolo especial. Una vez enviada la cadena devuelve 1 al nivel 1 de la pila.

Así, la manera mas sencilla de enviar el comando **INVL0** al HP48KIT sería utilizar el siguiente programa o bien la secuencia de comandos que lo componen en la HP48 :

<< *8 CHR XMIT DROP* >>

donde se introduce el código correspondiente al comando INVL que es el 8, luego con **CHR** se convierte el número 8 a su equivalente en **ASCII**, es decir en una cadena de caracteres, para luego transmitirse por el puerto serie con el comando **XMIT;** este devuelve un 1 a la pila y con **DROP** eliminamos este resultado de la misma.

En el caso de los comandos que requieren leer datos del puerto serie como es el caso de **RDIN1** se utiliza otro comando de la **HP48** que devuelve el contenido del buffer de entrada (que es de 255 bytes de largo) de la UART interna de la calculadora. La codificación del programa que ejecuta el comando **RDIN1** y permite leer el estado de las 8 entradas correspondientes al primer registro es el siguiente:

<< “%” XMIT SRECV DROP NUM >>

en este caso se uso directamente el carácter “por ciento” cuyo código ASCII es 37 (se puede obtener con el comando NUM) para enviar el código de operación correspondiente a **RDIN1** de esta manera nos ahorramos de convertir con el comando **CHR** y se obtiene una mejora en los tiempos de ejecución. El comando **SRECV** espera en la pila un número indicando la cantidad de caracteres que deberá leer del buffer de entrada de la UART, en este caso se deja el 1 que devuelve el comando **XMIT** y entonces **SRECV** devuelve el carácter leído de la *UART* y mediante el comando **NUM** se obtiene el valor decimal del ASCII que se leyó. Este valor corresponde al valor actual del registro de desplazamiento

4021 que se leyó en ese momento en el KIT.

Cada uno de los comandos del kit tiene una codificación similar a los 2 ejemplos anteriormente mostrados. Cuando se desea enviar una serie de comandos y siempre en el mismo orden, se pueden escribir cadenas de caracteres entre comillas con todos los comandos unos a continuación de otros y luego transmitirlos de una sola vez con el comando **XMIT** siempre que estos comandos no tengan operaciones de generación de pulsos o temporizaciones en cuyo caso se deberá esperar el tiempo en que estas se ejecuten para enviar nuevos comandos al KIT.

Un ejemplo de utilización del HP48KIT para una aplicación de voltímetro digital

A continuación vemos un ejemplo de aplicación del **HP48KIT** utilizado como un sencillo voltímetro digital. El programa que realiza la tarea de encuestar los 4 canales ADC y mostrar el valor en la pantalla de la HP48 es el siguiente:

```
<<RCLF    3 FIX    CLLCD DO
   RADV0 1 DISP    RADV1 2 DISP    RADV2
3 DISP    RADV3 4 DISP
    UNTIL KEY END
   DROP  STOF >>
```

Este programa es un claro ejemplo de la potencia del **HP48KIT** y del lenguaje de programación de la HP48 que con tan pocas instrucciones nos permiten visualizar las tensiones en los 4 canales ADC con un rango de 0.000 a 5.000 Volts.

Ahora expliquemos un poco como funciona el programa. El primer comando **RCLF** deja una copia de la configuración de los flags de la HP48 en la pila, luego con **3 FIX** se fijan a 3 el número de decimales para la visualización, con **CLLCD** se borra la pantalla. Luego se inicia un bucle de control estructurado tipo **DO UNTIL END** que sale con la pulsación de una tecla cualquiera. Dentro de este bucle se lee cada uno de los ADC con los comandos **RADV0, RADV1...3**. Estos son comandos que leen un ADC y devuelven el valor en Volts, escalado según una constante que es la tensión de referencia (en el caso del KIT siempre es +5V). Luego mediante **N DISP** se visualiza la tensión en la línea N del display. El ciclo se repite hasta presionar una tecla. Y por último con **STOF** se recuperan los flags de la HP48, que fueran modificados con el comando FIX. Este es un programa sencillo, pero se podría por ejemplo con la tensión leída controlar a su vez un DAC o activar salidas digitales ante cierta condición.

Posibles aplicaciones del HP48KIT

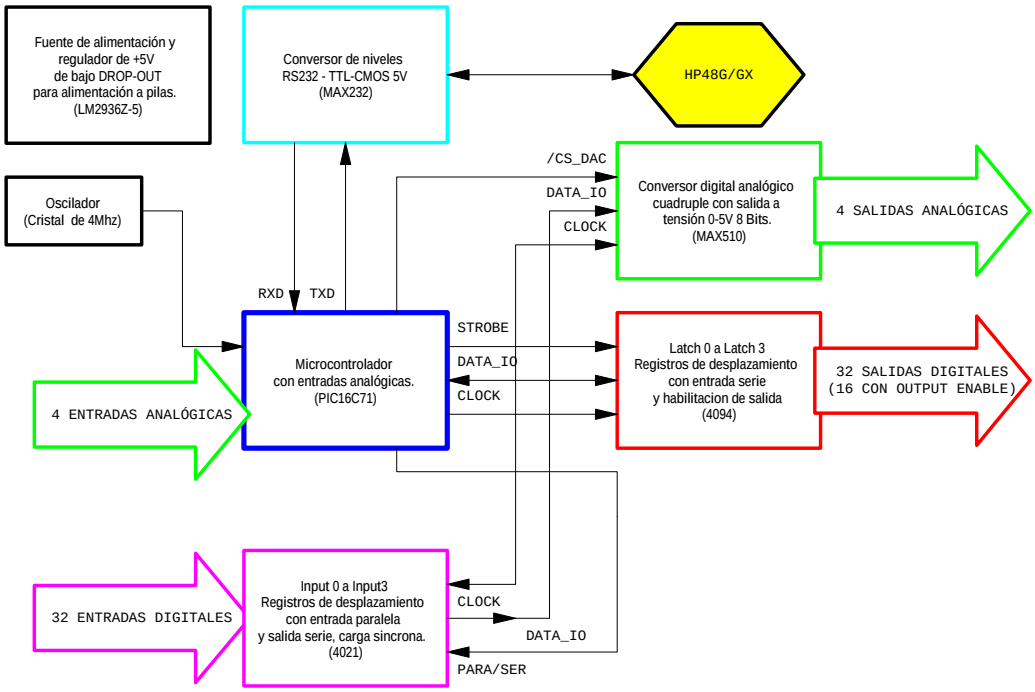
Las Posible aplicaciones del HP48KIT podrían ser: Controlar dispositivos externos como relays, lámparas, LEDs, triacs, leer temperaturas, presiones, pH, voltajes y corrientes, monitorear procesos de laboratorio, registrando datos a períodos regulares, programar memorias Eprom, Eeprom., probar circuitos integrados generando señales complejas y sincronizadas sin glitches y a su vez teniendo la posibilidad de programar todo en subrutinas bajo un lenguaje estructurado y de muy fácil asimilación.

Como se puede apreciar, estas aplicaciones son especialmente útiles en el área educativa, para realizar mediciones de laboratorio donde se requieran implementar scripts o automatizaciones rápidas de procesos donde haya que leer valores, efectuar todo tipo de cálculos (aquí es donde la HP48 se luce con su potencia de cálculo), almacenarlos, y actuar en consecuencia.

El potente set de instrucciones que posee el HP48KIT, sumado a los innumerables comandos propios de la HP48 y la posibilidad de programar eventos de Alarma que ejecuten procedimientos a horas específicas o a períodos regulares, hacen de este

conjunto, una herramienta muy útil para cualquier aficionado a la electrónica, diseñador, estudiante, docente o investigador que sea usuario de la HP48 y quiera explorar el mundo de la electrónica con su calculadora personal.

IV. DIAGRAMA DE BLOQUES EXPLICATIVO DEL SISTEMA :



V. DESCRIPCIÓN FUNCIONAL DEL CIRCUITO

El circuito consta básicamente de seis bloques funcionales. El más importante es la unidad de control que gobierna los demás circuitos, está formada por un potente microcontrolador **PIC16C71 (*U1 en el esquema eléctrico*)** corriendo a 4 Mhz. Se eligió este microcontrolador debido a que el mismo posee 4 entradas analógicas integradas en el mismo chip además de 1K de memoria de programa y 36 bytes de Ram para propósitos generales que resultan suficientes para el sistema operativo **HP48KIT** y todo esto a un bajo costo. Para la interface serial RS232 se prefirió implementarla por software, ya que solo se requiere una comunicación Half-Duplex a 9600 baudios y esto puede realizarse perfectamente a 4 Mhz. en estos micros debido a su arquitectura RISC y su alta velocidad de ejecución.

El segundo bloque consiste en el conversor de niveles **RS232** a TTL-CMOS-5V, para esto se utiliza el conocido **MAX232 (*U2*)** [6] con sus cuatro capacitores (***C1 a C4***) de soporte. Este bloque permite, a través de un conector **DB9 Macho (*CN1*)** la interconexión del **HP48KIT** con la calculadora **HP48G/GX** a través de un cable serie que normalmente se utiliza para conectar la misma a una computadora personal para intercambiar programas.

El tercer bloque es la fuente de alimentación del sistema, donde se utilizó el regulador **LM2936Z-5 (*U12*)** de National Semiconductor que con su baja caída de tensión serie (típicamente 200 mV@ 50 mA) permite obtener los +5 Volts. que alimentan a todos los circuitos integrados a partir de 4 pilas de 1,5 Volts. tipo AAA o AA. Esto junto al hecho de

que la calculadora trabaja con 3 pilas AAA. le otorga al **HP48KIT** un extraordinaria portabilidad.

El cuarto bloque funcional lo forma el **MAX510 (U3)** [7] que es un cuádruple conversor digital analógico serial de 8 bits con salida a tensión en todo el rango de alimentación (0-5V). Este integrado de la firma **MAXIM®** proporciona 4 salidas de tensión buffereadas a través de una interface serie compatible con el protocolo **SPI**, también permite variar las tensiones de referencia en grupos de 2.

Otro bloque, y uno de los más importantes, es el encargado de manejar las 32 salidas digitales. Este está formado por 4 registros de desplazamiento seriales tipo **CD4094 (*U4-U7*)** [8] que se encuentran enganchados unos con otros y manejados por el microcontrolador a través de un flujo de 32 bits que luego se latchea en las salidas mediante una señal de STROBE, las líneas de CLOCK y DATA_IO se comparten con los demás periféricos series, esto permite con solo 13 líneas de entradas-salidas en el 16C71 manejar muchos periféricos y con un considerable simplificación en el circuito impreso. La limitación en la velocidad de ejecución de los comandos del **HP48KIT** está impuesta por los 9600 Baudios de la interface serie de la calculadora y no habría mayores ventajas en este sentido si se controlaran las entradas y salidas con un protocolo paralelo en vez de serial.

Y por último está el sexto bloque funcional consistente en 4 registros de desplazamiento con carga paralela y salida serie tipo **CD4021 (*U8-U11*)** [9] que también se controlan serialmente. La línea de control **PARA/SER** permite cargar los registros internos de manera paralela, para luego serialmente leer los 32 bits en un flujo continuo de datos entre los **4021** y el **PIC16C71**.

VI. DESCRIPCIÓN FUNCIONAL DE TODAS LAS SUBRUTINAS DEL SOFTWARE.

Con respecto a las restantes subrutinas, no menos importantes, por razones de espacio solo me limitaré a explicar su función.

transmit : Transmite el contenido del registro w a 9600 BPS por la linea _tx.
transmit_reg : Ídem anterior pero toma el valor a enviar de la variable “dato”.
receive : Espera un byte del puerto serie, linea _rx y lo devuelve en W y en “dato”.
load_latch : Toma las variables latch0 a latch3 y con estos valores actualiza las salidas del 4094.
read_shift_reg : Lee los registros de los 4021 y guarda los valores leídos en sreg0 a sreg3.
intercambio : Efectúa una llamada a las dos subrutinas anteriores.
inicio : Rutina que se ejecuta luego de un Reset o Power_On y llama al programa principal.
init_ram : Parte de inicio que limpia a 0 todas las posiciones de memoria.
wait_command : Lazo principal del programa que interpreta los comandos seriales de la HP48.
grupo1 : Parte de la rutina wait_command, ejecuta los comandos comprendidos entre 0 y 79d.
grupo2 : Parte de la rutina wait_command, ejecuta los comandos entre 80 y 123d.
fin_comando : Se utiliza para ejecutar el NOP y volver al lazo principal de espera de comando.
cmd_clear : Pone a 00h un latch especificado en los bits 0 y 1 del comando.
cmd_set : Pone a FFh un latch especificado en los bits 0 y 1 del comando.
cmd_inv : Invierte el contenido de un latch especificado en los bits 0 y 1 del comando.
cmd_write : Escribe un valor recibido como parámetro en el latch especificado.
cmd_inc8 : Incrementa un latch individual.
cmd_dec8 : Decrementa un latch individual.
cmd_inc16 : Incrementa dos latchs como un word de 16 bits.
cmd_dec_16 : Decrementa dos latchs como un word de 16 bits.
cmd_readlatch : Devuelve el contenido previamente enviado al latch especificado por puerto serie.

cmd_readinp : Devuelve el byte leído del registro externo 4021 que se especifique.
indexa : Apunta el FSR a la memoria intermedia del latch o registro especificado en B0, B1.
cmd_swapnibble: Toma el contenido de un latch especificado, ejecuta un Swap y lo actualiza.
cmd_rotatelc : Rota ciclicamente el contenido de un registro 4094 a la izquierda.
cmd_rotaterc : Rota ciclicamente el contenido de un registro 4094 a la derecha.
cmd_revert : Intercambia los bits altos por los bits bajos de un latch especificado.
cmd_tobcd : Convierte el parámetro a BCD y con este valor actualiza el latch especificado.
cmd_to7seg : Ídem pero convierte a notación de 7 segmentos.
cmd_frombcd : Lee un registro de entrada, interpreta como bcd y devuelve este valor vía serie.
cmd_clrall : Coloca todos los latchs 4094 a 00h.
cmd_setall : Ídem anterior pero a FFh.
cmd_invall : Invierte todos los bits de todos los latchs 4094.
cmd_readobit : Lee el estado de un bit de un latch y devuelve este valor serialmente.
cmd_readbit : Ídem pero lee el estado real de un Pin de entrada de algún 4021 especificado.
cmd_clrbit : Pone a 0 el estado de un bit especifico, es decir de una salida de algún 4094.
cmd_setbit : Ídem anterior pero a 1.
cmd_invbit : Invierte el estado de un bit especifico, es decir de una salida de algún 4094.
bit_process : Subrutina interna que lee un byte del puerto serie y genera máscaras para leer bits.
calc_mask : Tabla para enmascarar un bit específico.
readbit_w : Similar al cmd_readbit pero opera sobre bit indicado en w y no espera dato serie.
clrbit_w : Similar pero opera sobre bit indicado en w.
setbit_w : Ídem.
cmd_writebit : Escribe un valor pasado como parametro en un bit también pasado como tal.
cmd_p8mseg01 : Genera un pulso 01 sobre un Pin de salida especificado, controla la duración en mseg.
cmd_p8mseg10 : Ídem pero es un pulso 10.
cmd_p8dseg01 : Ídem pero espera el tiempo en décimas de segundos. (Resolución 8 bits)
cmd_p8dseg10 : Ídem pero pulso 10.
cmd_p16mseg01: Opera de manera similar pero con una resolución de 16 bits y temporiza en mseg.
cmd_p16mseg10: Ídem pero 10.
cmd_strobe01 : Genera un clock breve 01 sobre el Pin especificado en el parametro serie.
cmd_strobe10 : Ídem pero clock breve 10.
cmd_sndsrl : Envía un byte a un dispositivo ext. en formato serie compatible SPI, clock negativo.
cmd_sndsrh : Ídem pero clock positivo.
cmd_rcvsrl : Recibe un byte de un dispositivo externo esclavo en formato serie SPI, clock negativo.
cmd_rcvsrh : Ídem pero clock positivo.
wait_8_mseg : Demora tantos milisegundos como indique la variable “pausa”.
wait_8_dseg : Demora tantas décimas de segundos como indique la variable “pausah”.
wait_16_mseg : Demora tantos mseg. como indiquen “pausah*256+pausal”
rutina_test_tx : Rutina para probar el correcto funcionamiento del puerto serie en transmisión.
rutina_test_rx : Ídem pero para recepción.
cmd_sdac0-3 : Carga un valor a un de los 4 dacs y actualiza la tensión generada inmediatamente.
cmd_ldac0-3 : Ídem pero no actualiza hasta recibir un instrucción UPDAC.
cmd_updac : Actualiza todos las tensiones que se hubieran cargado simultáneamente.
cmd_radc0-3 : Lee un canal ADC y devuelve este valor por el puerto serie a la HP48.
cmd_odl0 : Deshabilita la salida OE del primer 4094, dejando las salidas 0-7 en alta impedancia.
cmd_oel0 : Similar pero habilita la salida OE.
cmd_odl1 : Deshabilita la salida OE del segundo 4094, dejando las salidas 8-15 en alta impedan.
cmd_oel1 : Similar pero habilita la salida OE.

VII. PRINCIPALES RUTINAS DEL FIRMWARE DEL SISTEMA

Las principales subrutinas del firmware del sistema son:
Estas son:

a) Rutina de inicialización
b) Rutina de escritura de latchs
c) Rutina de lectura de registros de desplazamiento
d) Rutina o lazo principal del programa
e) Rutina de transmisión RS232 a 9600 BPS

f) Rutina de recepción de RS232 a 9600 BPS con “**Muestreo múltiple para reducir tasa de errores”.**

Diagrama de bloques de algunas de las principales rutinas del software del sistema

A continuación observamos los diagramas de flujo realizados para las rutinas principales del Kit:

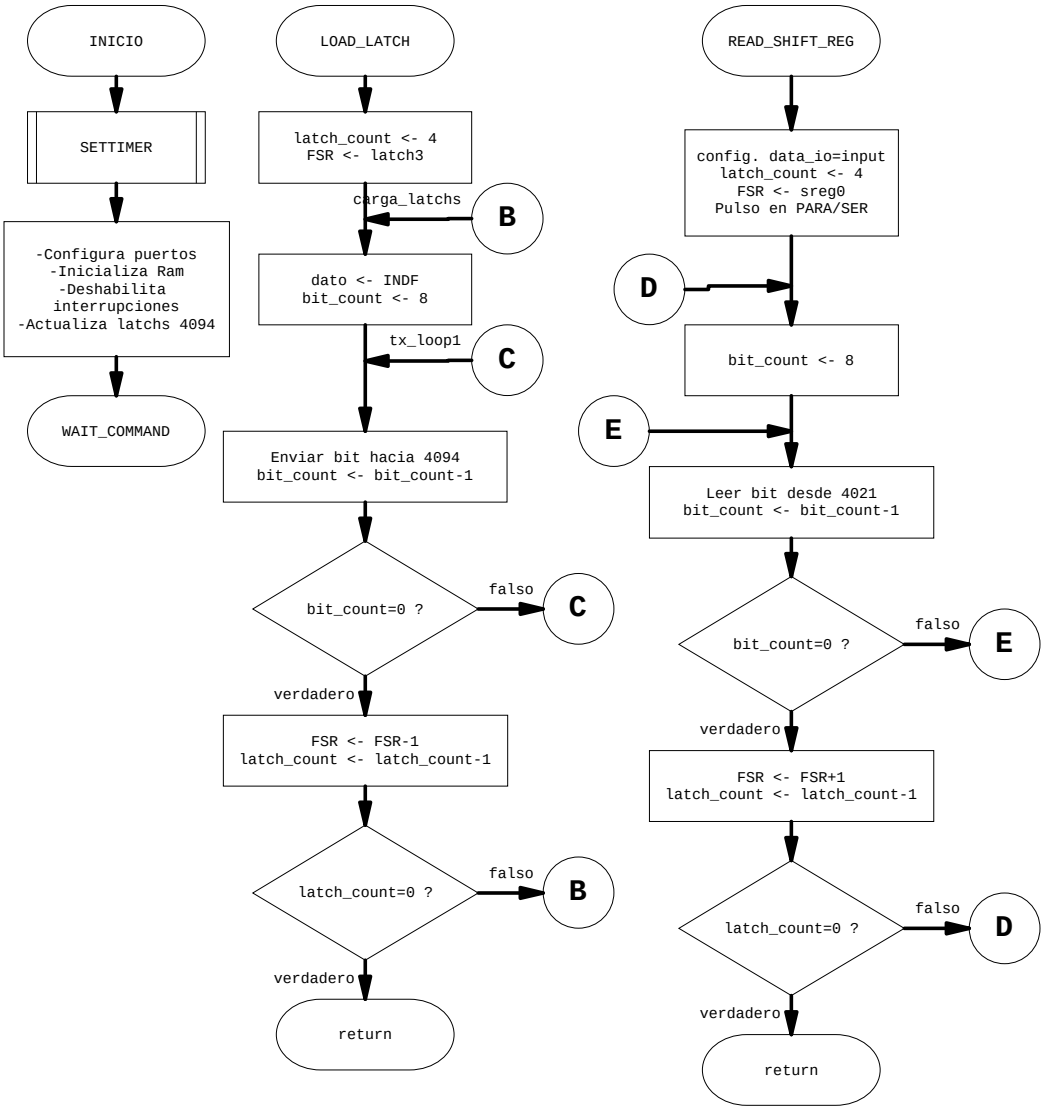


Diagrama 1.- Rutina de inicialización, escritura de latchs y lectura de registros de desplazamiento.

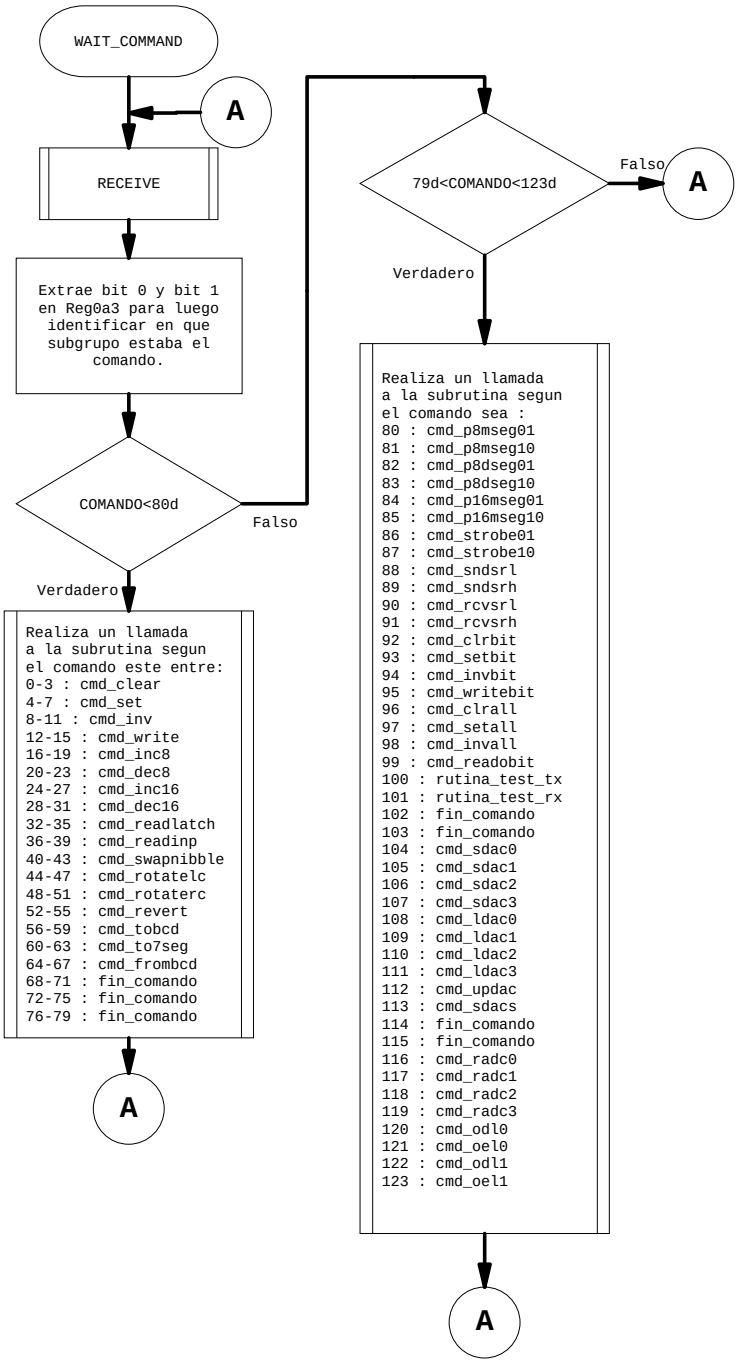


Diagrama 2.- Rutina o lazo principal de programa

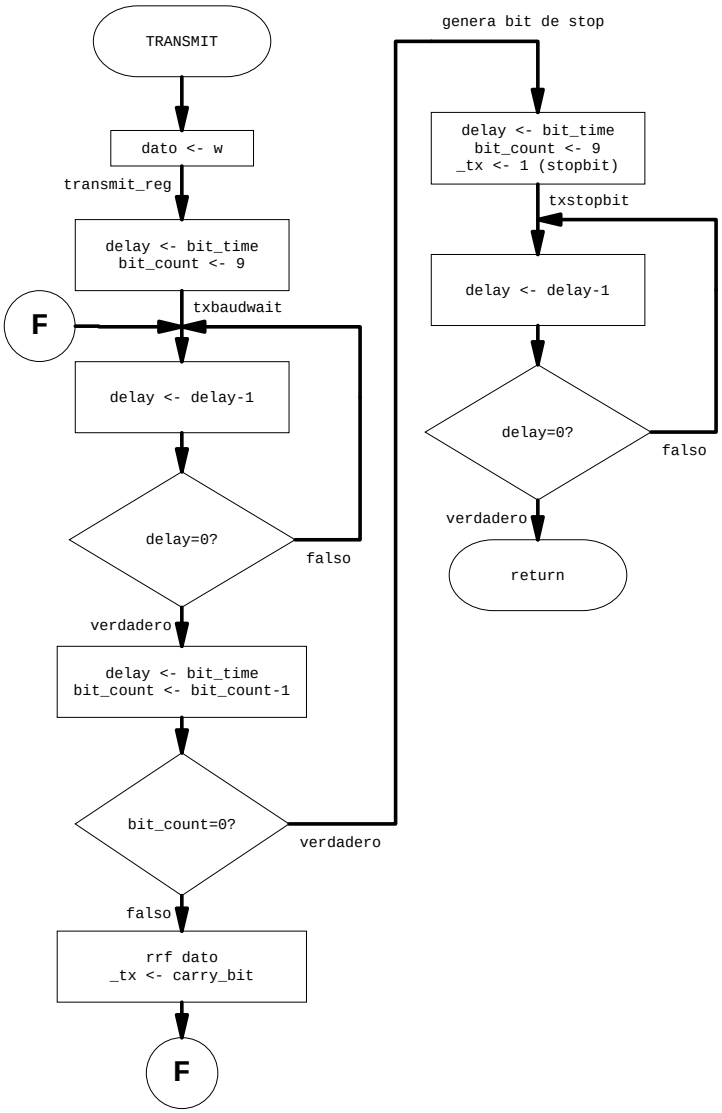


Diagrama 3.- Rutina de transmisión RS232 a 9600 BPS.

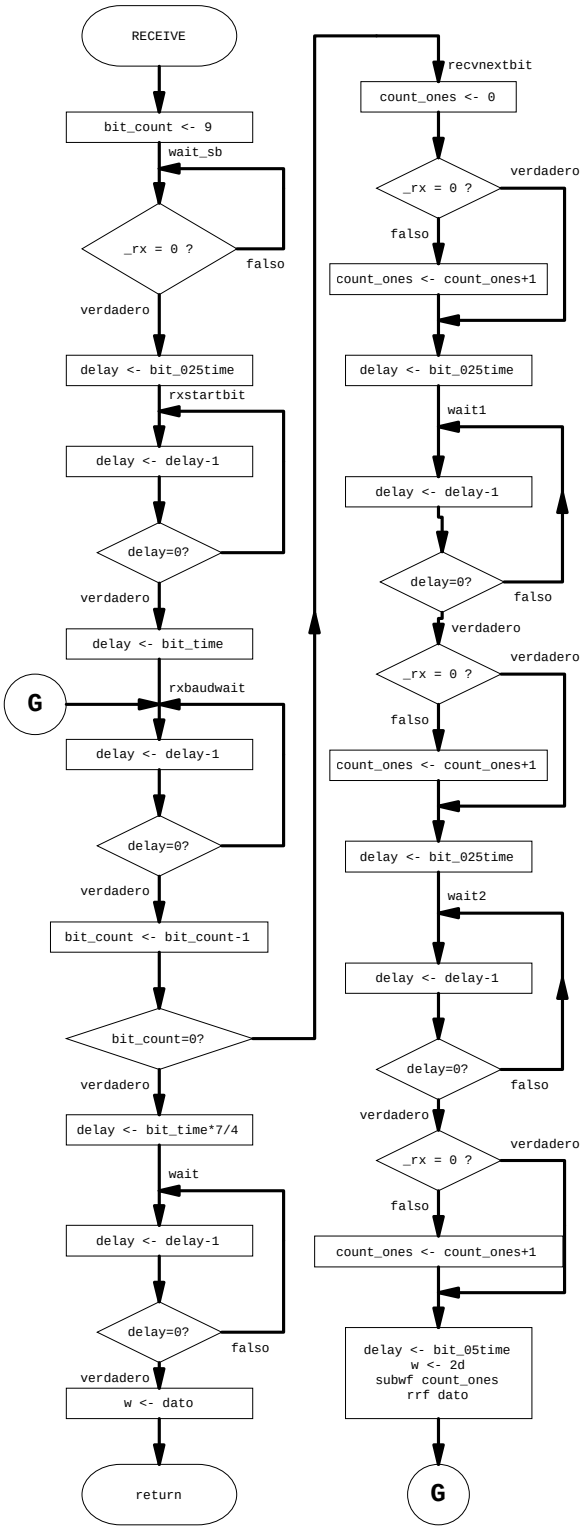
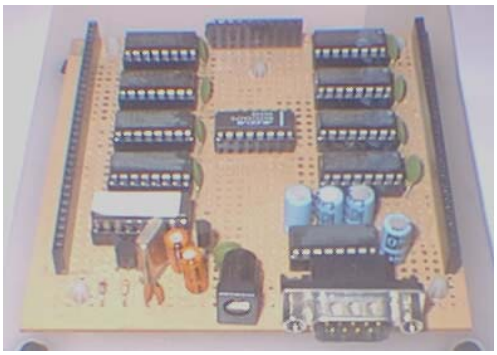


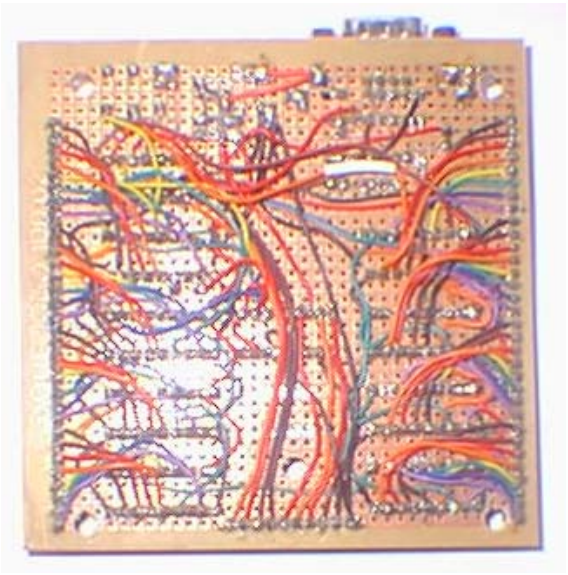
Diagrama 4.- Rutina de recepción de RS232 con "Muestreo múltiple para reducir tasa de errores".

VIII. PROTOTIPO

Descripción del prototipo

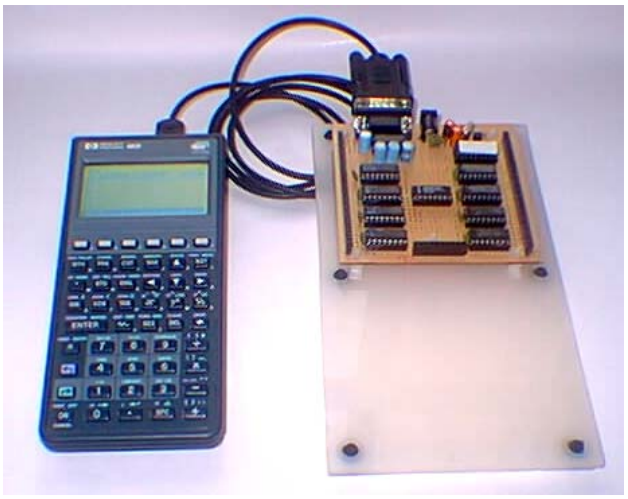
El prototipo se armó primeramente en un protoboard, en su fase de diseño, luego de algunos ajustes, el mismo se montó sobre un circuito impreso universal con soldadura (Las fotografías corresponden a esta fase de diseño). Sobre el circuito impreso del prototipo se halla un conector DB9 macho para PCB a 90 grados, para conexión a la HP48, un jack tipo DC para la alimentación de una fuente externa o 4 pilas de 1,5Volts. Todos los circuitos integrados se montaron en zócalos. Las 32 salidas digitales, 32 entradas digitales, 4 salidas analógicas y 4 entradas analógicas se extraen por medio de headers con una separación de 0,1 ‘’. En el modelo definitivo, se planea utilizar conectores DB37 a 90 grados para circuito impreso para la parte digital y conectores DB15 para la parte analógica.





Fotografías del prototipo

A continuación se observan algunas fotografías del prototipo funcional armado:



IX. DISTINCIONES RECIBIDAS CON EL PROYECTO

Cabe destacar que este trabajo se realizó en 1997 para participar del **CONCURSO DREAM MACHINE 97 Microcontrolador PIC 16/17 8-bit RISC** organizado conjuntamente por **MICROCHIP TECHNOLOGY Inc.** (USA) y **CIKA ELECTRÓNICA** (Argentina) y obtuvo en dicho certamen el **PRIMER PREMIO DEL CONCURSO** entre un total aproximado de 35 trabajos a nivel nacional.

El slogan del concurso era "*Un concurso de diseño para los que ven las cosas diferentemente*". Unos de los criterios de puntuación fue "Aquellos trabajos que manifiesten la Innovación y Eficiencia de los microcontroladores PIC en el diseño". Respecto de la puntuación, el trabajo obtuvo el 100% del puntaje, y unos de los puntos destacados por el jurado compuesto por Ingenieros de Desarrollo de Microchip fue que el trabajo explotaba al máximo la utilización de los recursos del microcontrolador PIC16C71 (Por aquella época uno de los mas grandes en 16 pines, y el único con memoria "flash" era el 16C84), generando una USART altamente eficiente por software, logrando con su bajo pin-out controlar 64 i/o y aparte destacaron la alta relación costo/beneficio del proyecto.

El premio consistió en un equipo emulador en tiempo real PICMASTER con sus correspondientes probes y accesorios y un programador PICSTART PLUS que fuera entregado por directivos de MICROCHIP.

X. REFERENCIAS

[1] http://www.hpcalc.org/
[2] http://ww1.microchip.com/downloads/en/DeviceDoc/30272a.pdf
[3] HP 48G Series Advanced User's Reference Manual
http://www.hpcalc.org/hp48/docs/books/aur.html
[4] 1. rplman (hp)
http://www.hpcalc.org/hp48/docs/programming/rpl-pdf.zip
RPL Programming Guide from Goodies Disk 4 in Adobe Acrobat PDF format. By Hewlett-Packard (http://www.hp.com/calculators/).
2. Programming in System Rpl - Kalinowski Dominik
http://www.hpcalc.org/hp49/docs/programming/progsysrpl_pdf.zip
http://staff.science.uva.nl/~dominik/hpcalc/
[5] http://www.dhmicro.com/Datasheet/LM2936-5.0.pdf
[6] http://www.datasheetcatalog.org/datasheet/texasinstruments/max232.pdf
[7] http://datasheets.maxim-ic.com/en/ds/MAX509-MAX510.pdf
[8] http://www.datasheetcatalog.net/es/datasheets_pdf/C/D/4/0/CD4094.shtml
[9] http://www.datasheetcatalog.com/datasheets_pdf/C/D/4/0/CD4021B.shtml