HP-82166A manual supplement
HP-IL Module ROM (PRG+PRDM5)
Introduction etc.

HEWLETT PACKARD

Hewlett-Packard Interface Loop Fact Sheet

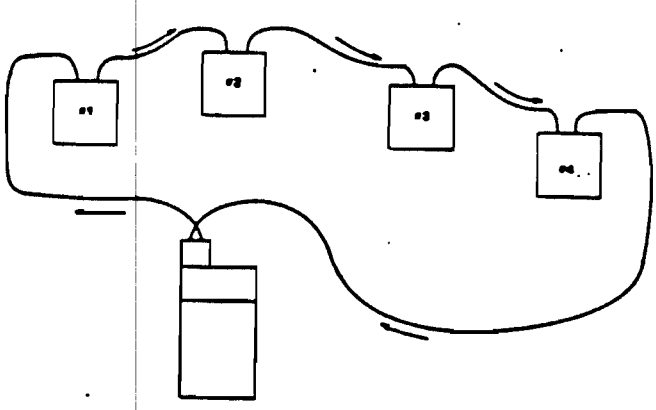
HP-IL

Why a New Interface?

Hewlett-Packard is committed to designing highly portable computational and information management systems. In these systems, power consumption, size, unit cost, and obsolescence must be minimized. HP-IL combines low power, small size, and low cost with an HP commitment to continued support and new product development.

What is HP-IL?

The Hewlett-Packard Interface Loop (HP-IL), is a bit serial interface designed for small, low cost, battery-operable systems. It is intended for use in field portable and simple bench-top systems which use highly-portable controllers, such as the HP-41.



In HP-IL systems, devices are connected by two-wire cables leading from the output port of one device to the input port of the next, until all devices form a closed loop.

This loop structure includes three unique new features: auto address assignment, device identification, and power ON/OFF control. The first two functions allow the controller to assign addresses to devices and to identify device capabilities. The third allows the controller to manage power utilization in battery-operated systems.

HP-IL vs. HP-IB

Although HP-IB and HP-IL serve the same basic function—interfacing controllers, instruments and peripherals—they differ in many respects.

1. Because of HP-IL's lower power consumption, it is usable with portable, battery-powered systems. Generally speaking, HP-IB is not.
2. HP-IL system components will generally be low cost and have moderate performance. HP-IB system components are at the medium- to high-end of the performance spectrum and generally cost more.
3. HP-IL systems work at relatively low data rates. HP-IB systems work at relatively high data rates.
4. HP-IL allows device separations of up to 100 meters with shielded, twisted pairs (10 meters with zip cord). HP-IB requires extender hardware for long distance connections.

HP-IL is not intended as a replacement for HP-IB, but rather as a low cost, low power alternative extending below the traditional scope of HP-IB in price and performance.

HP-IL adds a new dimension to Hewlett-Packard's instrumentation and computing capability. New HP-IL controllers, peripherals and instruments will be added to HP's product line on a continuing basis.

5953-1980 11/81

Printed in U.S.A.

HEWLETT PACKARD

Hewlett-Packard Interface Loop Fact Sheet

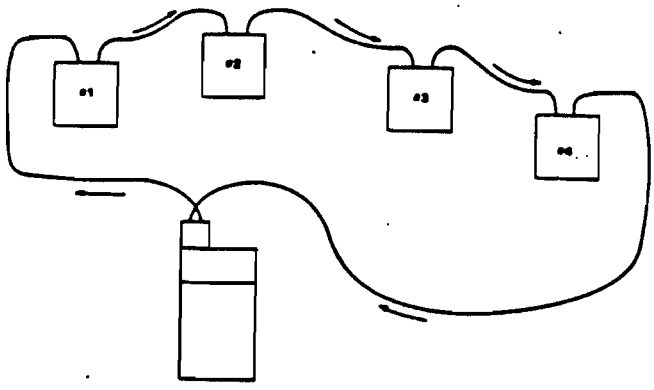
HP-IL

Why a New Interface?

Hewlett-Packard is committed to designing highly portable computational and information management systems. In these systems, power consumption, size, unit cost, and obsolescence must be minimized. HP-IL combines low power, small size, and low cost with an HP commitment to continued support and new product development.

What is HP-IL?

The Hewlett-Packard Interface Loop (HP-IL), is a bit serial interface designed for small, low cost, battery-operable systems. It is intended for use in field portable and simple bench-top systems which use highly-portable controllers, such as the HP-41.



In HP-IL systems, devices are connected by two-wire cables leading from the output port of one device to the input port of the next, until all devices form a closed loop.

This loop structure includes three unique new features: auto address assignment, device identification, and power ON/OFF control. The first two functions allow the controller to assign addresses to devices and to identify device capabilities. The third allows the controller to manage power utilization in battery-operated systems.

HP-IL vs. HP-IB

Although HP-IB and HP-IL serve the same basic function—interfacing controllers, instruments and peripherals—they differ in many respects.

1. Because of HP-IL's lower power consumption, it is usable with portable, battery-powered systems. Generally speaking, HP-IB is not.
2. HP-IL system components will generally be low cost and have moderate performance. HP-IB system components are at the medium- to high-end of the performance spectrum and generally cost more.
3. HP-IL systems work at relatively low data rates. HP-IB systems work at relatively high data rates.
4. HP-IL allows device separations of up to 100 meters with shielded, twisted pairs (10 meters with zip cord). HP-IB requires extender hardware for long distance connections.

HP-IL is not intended as a replacement for HP-IB, but rather as a low cost, low power alternative extending below the traditional scope of HP-IB in price and performance.

HP-IL adds a new dimension to Hewlett-Packard's instrumentation and computing capability. New HP-IL controllers, peripherals and instruments will be added to HP's product line on a continuing basis.

5953-1980 11/81

Printed in U.S.A.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

converter GPIO

HP 82166A/HP 82165A MANUAL SUPPLEMENT

The information in this supplement applies to the HP 82166A HP-IL Converter and to the HP 82165A HP-IL/GPIO Interface. Additional technical information not contained in the owner's manuals is provided here to assist in the proper application of these two products. This document also serves as an addendum to correct those errors which have been found in the owner's manuals.

Because these two products are nearly identical functionally, the information in this supplement refers to both unless explicitly stated otherwise. The term "converter" refers to both products throughout this supplement. Page references to the HP 82166A HP-IL Converter Technical Manual (82166-90002, Nov. 1981) will be given when appropriate followed by the corresponding page reference to the HP 82165A HP-IL/GPIO Interface Owner's Manual (82165-90002, Feb. 1982) contained in parentheses, for example, page 12 (14). Any previous addendums for these two products are superseded by this manual supplement, since that information is also included here. If information contained herein conflicts with information in the owner's manuals, this supplement takes precedence.

IMPORTANT NOTE FOR HP-41 OWNERS

The HP 82160A HP-IL Module for the HP-41 Handheld Computer does not by itself provide functions to access the converter control registers. This means that the converter can only be used in its power-on default operating mode: eight-bit bidirectional data bus, full three-line handshake, positive (high true) data bus logic, negative (low true) handshake line logic, and service request for manual service request only. Additional operations which cannot be performed with the HP-IL module alone are the converter/HP-IL test, the clear transfer buffer command, and operation using service requests. This default operation mode is sufficient for many applications.

The HP 82183A Extended I/O Module (available March 1983), when used with the HP-IL module, provides the additional functions needed to access the other converter operating modes for those applications which require them. While it was not designed with this purpose in mind, the 00041-15043 HP-IL Development Module (available December 1982) also contains a set of functions which can provide this extra capability when used with the HP-IL module.

Additional information regarding the use of the HP-41 and the HP-IL module with the converter is contained in Appendix D, "Using the HP-41 As a Controller," page 33 (34), as well as in the HP-IL module owner's manual. Information on the extended I/O module and the HP-IL development module can be found in their respective owner's manuals.

THE TRANSFER BUFFER

All the data that passes through the converter is stored temporarily in the transfer buffer. There is only one transfer buffer. It can contain data bytes being sent from HP-IL to the GPIO device or bytes being sent from the GPIO device to HP-IL, but not both simultaneously. If the buffer contains data received from the GPIO device and HP-IL performs a write operation to the converter, the data from the GPIO device will be lost. If the buffer has data from HP-IL, the GPIO device cannot write into the converter buffer since the RDYO line will not go true. This will be a problem only in applications where data must pass in both directions. Loss of data from GPIO can be avoided by reading the converter status before writing data from HP-IL into the buffer.

The manual states that the converter is able to hold 32 bytes of data. This is not correct. If the GPIO device is not accepting data, HP-IL can put 30 bytes of data into the transfer buffer. The 31st byte will be held by the converter and not retransmitted on the loop. When the GPIO device accepts some data, thereby making room in the buffer, this byte will be retransmitted on the loop (and loaded into the buffer) and the transmission of data can continue. If the GPIO device does not respond, the converter will hold the 31st byte indefinitely or until the controller times out. In this situation, the controller normally aborts the data transmission and displays an error message.

If the GPIO device is sending data to HP-IL, the converter buffer can contain up to 31 bytes of data from the GPIO device. In this condition, the RDYO line will not go true, so the GPIO device cannot write any more data into the converter. When HP-IL reads one or more bytes from the buffer, the RDYO line will once again go true, and the GPIO device can continue sending its data.

Information on the transfer buffer is found on page 6 (8), 11-13 (13-15), 16 (18), and 27-28 (28-29).

THE STATUS BYTE AND SERVICE REQUESTS

An application which only sends data to the GPIO device may not require status information. If the GPIO device is faster than the sourcing device, the buffer will never have more than one or two bytes in it. The data transmission will take place without interruption at the rate of the sourcing device. On the other hand, if the GPIO device is slower than the sourcing device, the buffer may fill up. When this occurs and another byte is sourced, the converter simply holds up the loop transfer until the GPIO device accepts a byte, thereby making room in the buffer. In this case, the data transmission takes place with brief interruptions at the rate of the GPIO device.

When the GPIO device must send data to HP-IL, when data must flow both ways, or when I/O operations need to be interleaved, it is important to know the converter status so the controller can determine what needs to be done next. Most of the status indications are related to the transfer buffer.

If the decimal value of the returned status byte is 1, the converter is ready to receive data from HP-IL. This could mean the buffer is empty, or it could mean that the buffer still contains some data from HP-IL which is flowing steadily out to the GPIO device and more data can be accepted from HP-IL immediately. If the status value is 8, the buffer has some data from HP-IL, but the data is not flowing out to the GPIO device. This usually means that the GPIO device did not set the RDYI handshake line true. If the DAVO timeout option is enabled in the converter control registers, it could mean that although the RDYI line was true, the GPIO device did not accept the data byte by driving the DACI line true before the timeout period expired.

However, in most situations the convertor returns a value of 1 even though the GPIO device is not taking the data. If the Interface Clear command is sent to the converter just prior to reading its status, it will update the status internally and return the correct value, either 8 or 1. For example, the HP-41 sends this command by executing the [STOPIO] function. Note that the Interface Clear command does not change the data in any way.

When data is being sent from HP-IL to GPIO, the status will always be either 1 or 8. Even though the buffer is full (30 bytes), the buffer full status value of 4 will not be indicated. Furthermore, data cannot be sent from the GPIO device while data from HP-IL is in the buffer since the RDYO line will not go true, so that the buffer busy status value of 16 will also never be generated.

A status value of 2 indicates that the buffer contains some data from the GPIO device and is ready to send it to HP-IL. When the status byte is 6, the transfer buffer has received 31 bytes of data from the GPIO device and will accept no more until some of the data is read out of the buffer by HP-IL.

On the HP 82165A only, the front panel MSRQ (manual service request) key simply connects the $\overline{\text{MSRQ}}$ line (Pin 13) to ground while it is pressed. On both products, the status value of 32 (MSRQ) is returned only if status is read while the $\overline{\text{MSRQ}}$ line is grounded. The MSRQ condition is not automatically retained until the status is read.

Each of the status conditions (as well as the manual service request condition) can be used to generate a service request on the loop (sets bit CO in Data and Identify messages). The power-on default condition is that MSRQ will generate a service request, while the status conditions do not. This default can be changed by setting

or clearing the appropriate bits in control register R00. If bit 7 is set to enable status service requests, either bit 5 can be set to enable any change in status to cause a service request, or bit 5 can be left clear and combinations of bits 4 through 0 can be set to cause particular status conditions to send a service request.

Information on the converter status and service requests is found on page 12-13 (14-15) and 27-28 (28-29).

THE HANDSHAKE LINES

Before the operation of the handshake lines can be thoroughly understood, some familiarity with the microprocessor program structure is needed. All of the I/O operations of the converter are done by polling rather than by interrupts. The converter's internal program simply runs in a continuous loop checking to see if an HP-IL or GPIO operation needs to be done. The operation of this program loop is the same regardless of whether the converter is the talker or a listener on HP-IL. Only the transfer buffer and the handshake lines affect this firmware loop.

At the top of the loop, HP-IL is polled to see if a message has been received. If it has, the appropriate action is taken, which might include retransmitting the received message on the loop, or perhaps sending a data byte from the transfer buffer to the loop.

Next, the microprocessor checks to see if there is data in the buffer waiting to be sent to the GPIO device. If there is, the state of the RDYI line is checked. If the GPIO device is not ready (RDYI false), the firmware simply goes back to the top of the loop. If the GPIO device is ready (RDYI true), the converter sends the byte on the data bus lines and sets the DAVO line true. Then the converter waits until the GPIO device accepts the data by driving the DACI line true. When this occurs, the microprocessor goes back to the top of the loop and continues.

If the GPIO device does not respond by driving the DACI line true, the converter will wait indefinitely. This means that HP-IL will not function during this time, since the converter cannot respond to a received message until it gets back to the top of the loop. If necessary, the DAVO timeout option can be enabled to handle this situation. By appropriate settings for control register R02-3,0 and control register R03, the converter can be made to wait only the specified amount of time in this state and then remove the data from the data bus lines and drive DAVO false if the GPIO device has not responded. The data byte is retained in the converter buffer and the converter firmware will attempt to output the byte again the next time through the loop.

The output (HP-IL -> GPIO) timing numbers in the table on page 19 (20) are misleading and should not be used. The following paragraphs provide the appropriate information for interfacing in any one of the four output handshake modes. Refer to the diagrams on page 15 (17).

In the full handshake mode (default) the GPIO device may set the RDYI line true at any time to indicate that it is ready to receive data. The GPIO device must hold the RDYI line true until the converter responds by driving the data bus lines and the DAVO line true. If there is no data in the buffer, the GPIO device should simply wait until there is. If there is data in the buffer, the delay from RDYI true to DAVO true may range from about 250 microseconds to one millisecond. As soon as the DAVO line is true, the GPIO device may accept the data and drive the DACI line true to indicate this. If the GPIO device does not drive the DACI line true immediately the converter will simply wait until it does (unless the DAVO timeout option is enabled, as described previously). The GPIO device now holds the DACI line true until the converter removes the data and sets the DAVO line back to its false state (about 80 microseconds). Then the GPIO device can return the RDYI and DACI lines to the false state, thus completing the cycle. This should be done within about 50 microseconds after the DAVO line goes false so that the converter will not try to send a second byte on the same cycle.

In the valid/accepted handshake mode the RDYI line is not used. When there is data in the buffer the converter simply drives the data bus lines and the DAVO line true. The converter now waits until the GPIO device signals acceptance of the byte by setting the DACI line true (the DAVO timeout option may be used here, also). The GPIO device holds the DACI line true until the converter removes the data and drives the DAVO line false again (about 80 microseconds). The GPIO device must return the DACI line to its false state within about 50 microseconds after the DAVO line goes false so that a second byte will not be sent on the same cycle. Note that if the GPIO device simply holds the RDYI line always true in the full handshake mode, the result is the same as the valid/accepted mode just described.

The DACI line is not used in the ready/valid handshake mode. The GPIO device simply sets the RDYI line true when it is ready to accept data. If there is data in the buffer there will be a delay (250 microseconds to one millisecond) and then the converter will drive the data bus lines and the DAVO line true. The DAVO line will remain true for a period of time determined by control registers R02-3 and R03 (there is no DAVO timeout option in this mode). The GPIO device should return the RDYI line to its false state within about 50 microseconds after the DAVO line goes false to prevent a double byte transfer. If the GPIO device holds the DACI line always true in the full handshake mode, the result is the same as the ready/valid mode except for the length of the DAVO pulse.

Neither the RDYI line nor the DACI line are used in the strobed output mode. When there is data in the buffer the converter simply drives the data bus lines and the DAVO line true for a length of time determined by control register R02-3 and R03 (there is no DAVO timeout option here, either). If the GPIO device holds the RDYI line and the DACI line always true in the full handshake mode, the result is the same as the strobed output mode except for the length of the DAVO pulse.

Important Note: On the HP 82165A only, the GPIO device should not use the leading edge of the DAVO line to latch data from the converter. The data is not valid until approximately 50 nanoseconds after the leading edge of the DAVO pulse. This does not apply to the HP 82166A.

If there is no data to be sent to the GPIO device, the microprocessor checks to see if the GPIO device wants to send data to HP-IL. It does this by driving the RDYO line true and then waiting a short time to see if the GPIO device drives the DAVI line true to indicate that the data on the data bus is valid. If the GPIO device does not respond, the converter returns the RDYO line to its normal false state and goes back to the top of the loop. Consequently, when no data is being transferred, a series of pulses will be transmitted on the RDYO line, one pulse each time the microprocessor passes through the polling loop. If there is data in the buffer to be sent to the GPIO device, there will be no RDYO pulses and the converter will not be able to receive data from the GPIO device.

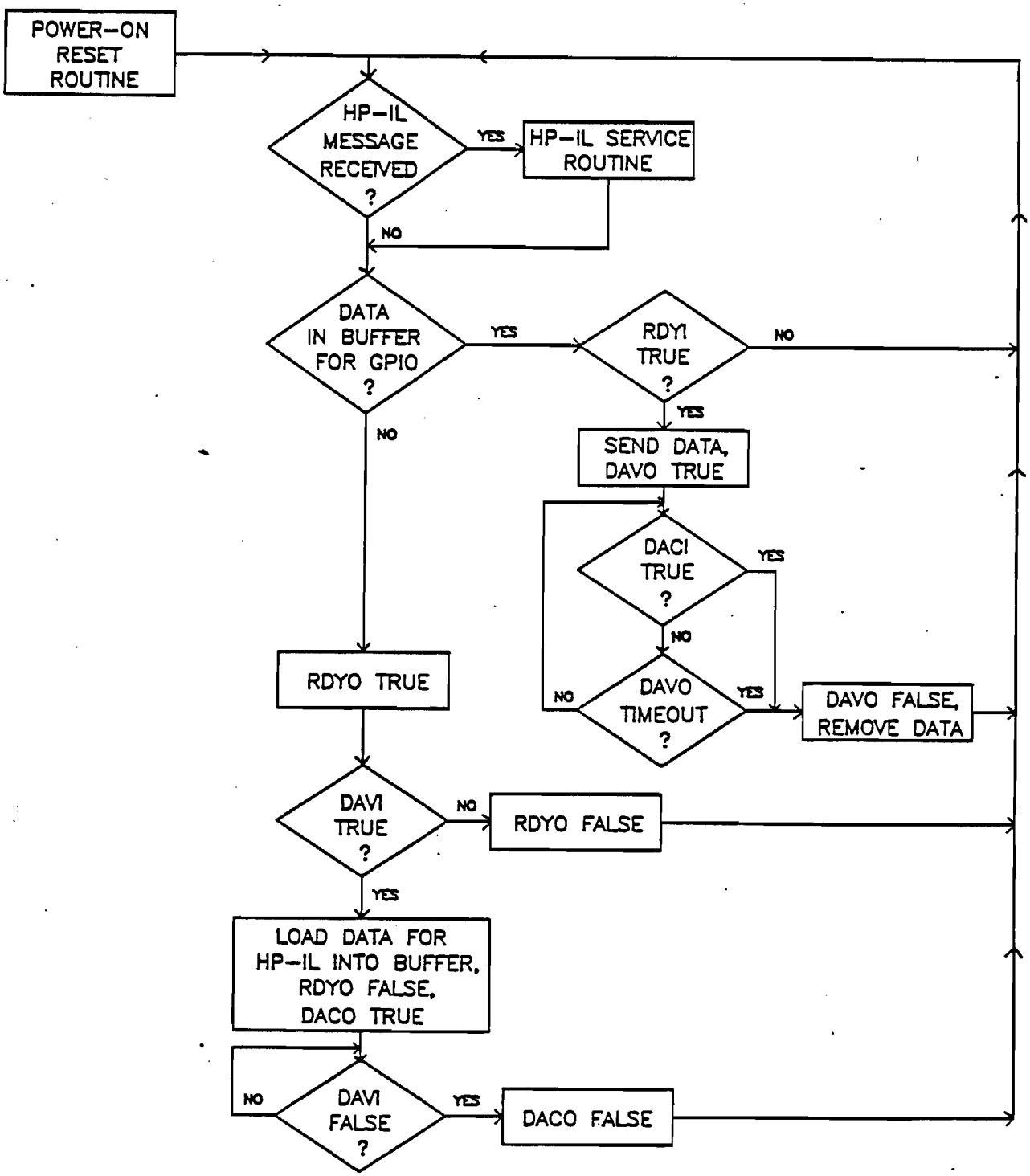
After the GPIO device puts data on the bus and sets the DAVI line true in response to the RDYO pulse, the converter will load the data into the buffer and set the DACO line true and the RDYO line false. The GPIO device now returns the DAVI line to its false state and removes the data from the bus. At this point, the converter resets the DACO line to its false state and goes back to the top of the loop.

In order for the converter to recognize that the GPIO device wants to send data, the DAVI line must be true sometime during a time window starting at the leading edge of the RDYO pulse (about 60 microseconds). If the device does not set the DAVI line quickly enough, it will simply need to wait for the next RDYO pulse, holding the data and the DAVI line true until the converter drives the DACO line true.

The external device must use the RDYO line in order to know whether or not the converter can receive data. If the external device responds quickly enough to the RDYO pulse and carefully controls the length of time that it drives the data and the DAVI lines true, then it need not use the DACO line. However, it will generally be easier for the device to simply drive the data and the DAVI lines true when the RDYO line goes true and remove the data and set the DAVI line false when the DACO line goes true. This information should clarify the paragraphs near the bottom of page 7 (9), "GPIO Input Handshake Lines (RDYO, DAVI, DACO)" and on page 14 (17), "For HP-IL <- GPIO operations, ..." Refer to the input (HP-IL <- GPIO) timing characteristics in the table on page 19 (20).

Note that the microprocessor waits until the GPIO device removes its data and drives the DAVI line false before it returns the DACO line to its normal false state. If the GPIO device holds the DAVI line true and does not return it to the false state, the converter will wait indefinitely for this to happen. As before, HP-IL will not be functional while the GPIO device holds the converter in this state. There is no timeout option for this situation.

The following flowchart summarizes the previous information:



In the diagram at the bottom of page 15 (17), "HP-IL <- GPIO Operation", the blue arrow from the trailing edge of the DAVI pulse to the trailing edge of the RDYO pulse should be drawn from the leading edge of the DAVI pulse to the trailing edge of the RDYO pulse. Also, all the diagrams on page 15 (17) indicate the logical state of the handshake lines--not the voltage level. If negative handshake logic is active (the default condition), the true state (shown as high in the diagrams) corresponds to zero voltage, and the false state (low in the diagrams) corresponds to a positive voltage. If positive handshake logic is active, the logic levels shown in the diagram correspond to the actual voltage levels.

A lock-up condition can occur if a GPIO device using positive handshake logic is connected to the converter and is powered on before or at the same time as the converter. The GPIO device will power on with the DAVI line driven false (its normal condition), which is low. The converter powers on reading the low DAVI line as a true condition, however, and stops in the loop waiting for the DAVI line to go high (false). The control registers cannot be modified to remove the lock-up, since the program cannot return to the HP-IL service routine.

In order to use positive handshake logic, it is necessary to first set the control registers in the converter and then connect the GPIO device. An alternative is for the GPIO device to hold the DAVI line high until the converter control registers are properly set from HP-IL. Because of this, use of positive handshake logic may cause one invalid data byte to be accepted from the GPIO device at power-on only.

One other situation which can arise is best illustrated by considering two converters connected back-to-back on GPIO. If HP-IL writes, say, 40 bytes of data into the first converter, GPIO will transfer 31 bytes into the second converter's buffer, and 9 bytes will remain in the first converter's buffer to be sent later. Now if HP-IL writes some data into the second converter, the 31 bytes will be destroyed, and both converters will be locked up, waiting to send data to the other on GPIO. Neither can accept the other's data since the HP-IL data in the buffer prevents either converter from driving its RDYO line true. A clear buffer command or power-on reset is needed to recover from this situation. Unusual circumstances might also cause a similar condition to occur with a GPIO device and a converter. HP-IL continues to function in this case.

ADDITIONAL INFORMATION

The following information applies only to the HP 82165A manual:

On page (6) the box in the diagram should be labeled "HP-IL/GPIO Interface" rather than "HP-IL/RS-232 Interface".

On page (9) under "Power Supply," add the sentence "The interface is isolated from earth ground by the AC adapter."

On page (9) under "GPIO Output Handshake Lines (RDYI, DAVO, DACI)," the phrase " when DAVO is false, the data bus lines are high." should read " when DAVO is false, the data bus lines are undefined."

On page (10) under "HP-IL Interfacing Output Line ($\overline{\text{GETO}}$)," the sentence "An active low signal on the GETO line sets a Group Execute Trigger message." should read, "The GETO line is normally driven high by the interface. When an HP-IL Group Execute Trigger (GET) command is received, the interface pulses this line low briefly."

On page (11) the entry in the table next to "Ready for Command" should be "No response" rather than "Executes a pending Loop Power Down message."

On page (16) add the following note to the table at the top of the page, "In the 8-bit unidirectional configuration, data bus A receives data from the external device and data bus B sends data to the external device."

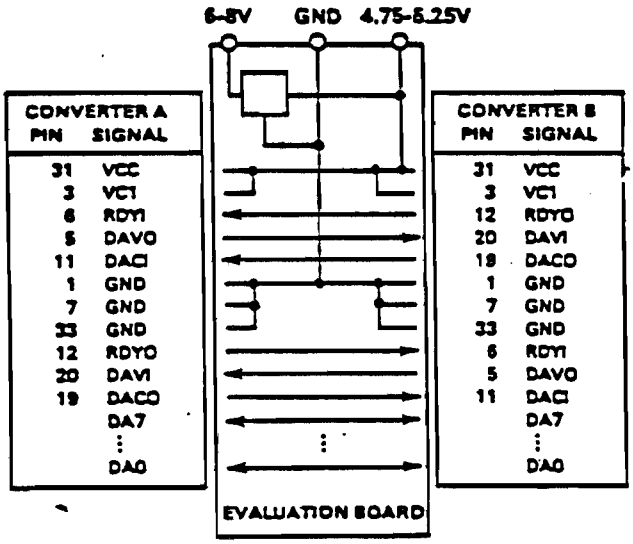
On page (19) under "Electrical Characteristics," the entry for "Voltage to Any Pin" should include a minimum of -0.3 V. Also, delete the last four lines of the table, beginning with "Output Current, High Level."

The following information applies only to the HP 82166A manual:

On page 11 in the bottom table, the response for the Device Dependent Talker 0 message (Send Control Registers) should include the sentence, "The transfer buffer is cleared." The response for the Device Dependent Talker 2 message (Enable End-Of-Line) should include the sentence, "The GPIO end-of-line sequence must be the last data bytes in the buffer."

On page 18, under "Electrical Characteristics," the specification for "Supply Ripple Voltage" should be deleted.

On page 29 under "Converter to Converter," the evaluation board has been changed to provide more convenient operation. Change the supply voltages in the second paragraph to 4.75 to 5.25 Vdc and 6 to 8 Vdc. Delete the caution--the evaluation board won't be damaged by operating it with less than two units connected. Revise the diagram of the board as shown below.



The following information applies to both manuals:

On page 13 (15) in the table, the designations "Bit 1", "Bit 2", etc. should be changed to "D0", "D1", etc.

On page 19 (20) in the lower timing diagram, the vertical line extending through the trailing edges of both RDYO and DACO should be two separate lines. The end of the RDYO pulse is not related to the end of the DACO pulse.

On page 36 (37) the last four lines in the program example should be numbered consecutively with the previous line.

Version #1

HPIL MONITOR ROM

TABLE OF CONTENTS

1.0 PURPOSE OF THE ROM

THE PURPOSE OF THIS 4K ROM IS TO PROVIDE AN EASY TO USE AND INEXPENSIVE DIAGNOSTIC TOOL FOR ALL THE PRESENT AND FUTURE PIL DEVICES. A WAY TO PROVIDE THIS TOOL IS TO MAKE ALL THE PIL COMMAND AND DATA TRANSFER CAN BE EXECUTED BY THE USER LEVEL LANGUAGE. THEREFORE ANY PIL TEST PROGRAM CAN BE WRITTEN IN USER LEVEL LANGUAGE. IT WILL BE EASY TO DEVELOP AND EASY TO MODIFY. ANY FUTURE PIL PRODUCT CAN USE THE SAME DIAGNOSTIC ROM TO DEVELOP ITS OWN TEST PROGRAM TOO.

ANOTHER INTENTION OF THIS ROM IS TO MAKE THE 41C BECOMING A PIL LOOP MONITOR. USE THE 41C AS A SCOPE TO SHOW OR BUFFER THE FRAME IN THE LOOP. WITH A PRINTER PLUG-IN, FRAMES CAN BE PRINTED TOO.

```
PPC ROM 2C
X<>ROM
Y<>T
Z<>T
Y<>Z
BCDBIN
GTOROM
RXR
RXL
X+Y
SXL
NSTO
NRCL
TP
VP
CODE
DECODE
EN
DE
CLPV
XROMTST
MELBROM 1A
PSIZE
PPACK
DIS
2-D
1-D
KEY
PKEY
X>Y??
CHR$
NUM
RDEL
R$
L$
LDEL
RHASTO
CLRKEY
```

```
-MONITOR 1A
RFRM
RREG
WFRM
WREG
CF33
SF33
FRAV?
FRNS?
IFCR?
ORAV?
SRQR?
AAU
CMD
DDL
DDT
GET
GTL
IFC
LAD
LPD
REN
SDC
TAD
UNL
UNT
AAD
NRD
SAI
SDA
SDI
SST
TCT
IDY
A-BUF
A=BUF?
A=BUFX?
AIPT
BSIZE?
BUF-AX
BUF-RGX
BUF-XA
BUF-XB
INBIN
INBUFX
IOFSX?
MIPT
MONITOR
OUTBIN
OUTBINY
OUTBUFX
PT=
PT?
PRBYTES
PRFRMS
RG-BUFX
RG=BUF?
ROMCHKX
SCOPE
SETBUFX
X-BUF
X=BUF?
X<>FLAG
```

2.0 DIRECT READ/WRITE TO PIL CHIP REGISTERS

THE PIL CHIP HAS 8 SINGLE BYTE REGISTERS. EACH OF THIS REGISTER CAN BE READ OR WRITTEN THROUH THE 41C X REGISTER.

WREG : WRITE TO A GIVEN REGISTER
Y IS TAKEN AS THE REGISTER NUMBER AND X IS TAKEN AS A DECIMAL NUMBER (0-255) TO WRITE TO THE REGISTER.

RREG : READ FROM A GIVEN REGISTER
X IS TAKEN AS THE REGISTER NUMBER TO READ FROM. THE CONTENT OF THE REGISTER WILL BE CONVERTED INTO A DECIMAL NUMBER AND PUSHED INTO THE STACK. SO AFTER THE READ Y= REG NUMBER, X=CONTENT OF THE REG

WFRM : WRITE FRAME (SENDING A FRAME)
Y IS TAKEN AS A DECIMAL NUMBER PRESENTING THE 8 DATA BITS. X IS TAKEN AS A NUMBER PRESENTING THE 3 CONTROL BITS. EXAMPLE: SENDING THE COMMAND FRAME "IFC" X = 4, Y = 144. "WFRM" WILL CHECK "ORAV" AND "FRNS" BEFORE SENDING THE FRAME. IF "FRNS" WAS SET, IT WILL GENERATE "FRNS ERR" MESSAGE. IF "ORAV" IS NOT SET WITHIN 10 SECONDS, IT WILL GENERATE "ORAV = 0" MESSAGE. "WFRM" WILL NOT WAIT FOR FRAME COMES BACK NOR WILL IT DO THE ERROR CHECKING AFTERWARD.

RFRM : READ FRAME
"RFRM" WILL READ RSGISTER 1&2 AND PUT THEM INTO STACK WHERE X = A DECIMAL NUMBER PRESENTING THE 3 CONTROL BITS AND Y = A DECIMAL NUMBER PRESENTING THE 8 DATA BITS OF THE FRAME. "RFRM" WILL NOT CHECK "ORAV" & "FRNS" PRIOR TO READ REGISTER 1 & 2. IT WOULD JUST READ IT.

THE FOLLOWING 5 FUNCTIONS ARE FOR TESTING THE LOWER 5 BITS OF REGISTER 1. IF IT IS EXECUTED FROM PROGRAM EXECUTION, IT WILL SKIP THE NEXT STEP IF THE ANSWER IS YES. IF IT IS EXECUTED FROM KEY BOARD, IT WILL ONLY DISPLAY "YES" OR "NO".

IFCR? : IFC RECEIVED ?

SRQR? : SERVICE REQUEST RECEIVED ?

FRAV? : FRAME AVAILABLE ?

FRNS? : FRAME RETURN NOT AS SEND ?

ORAV? : OUTPUT REGISTER AVAILABLE ?

THERE ARE TWO POTENTIAL PROBLEMS WHEN USE THIS DIAGNOSITC ROM WITH PIL BASIC MODULE. ONE IS THAT IF THE PIL CHIP IS NOT AUTO SOURCING "IDY", IT WILL BE TURN OFF WHEN THE 41C GOES TO LIGHT SLEEP. ANOTHER ONE IS THAT IF THE 41C IN AUTO I/O MODE, IT WILL CONSTANTLY SOURCE FRAMES TO LOOK FOR THE PRINTER. AND THIS WILL CHANGE THE REGISTER 0,1,2,3 OF THE PIL CHIP. IN ORDER TO PREVENT THE 41C GOING TO LIGHT SLEEP AND TO PREVENT THE BASIC MODULE MESSING AROUND WITH THOSE REGISTERS, WE PROVIDE THE FOLLOWING 3 FUNCTIONS :

SF33 : SET USER FLAG 33
WHEN USER FLAG 33 IS SET, NO PIL FUNCTION WILL BE EXECUTED BY THE BASIC MODULE. OF COURSE, IT WILL NOT LOOK FOR THE PRINTER EITHER. WHILE FLAG 33 IS SET, TRY TO EXECUTE ANY PIL BASIC FUNCTION WILL RESULT IN TRANSMIT ERROR. THE USER FLAG 31-35 CAN NOT BE SET OR CLEARED BY THE MAINFRAME "SF" AND "CF" FUNCTION. AND IT WILL NOT BE CLEARED ON TURN ON EITHER. THESE CAN BE ONLY CLEARED BY "MEMORY LOST" OR BY SOME SPECIAL FUNCTION LIKE THE NEXT FUNCTION "CF33".

CF33 : CLEAR USER FLAG 33

ON : STAY ON
ACTUALLY THIS IS A MAINFRAME FUNCTION TO PREVENT THE 41C GOING FROM LIGHT SLEEP TO DEEP SLEEP(OFF). THE "ON" FUNCTION ALSO SETS THE SYSTEM FLAG 44. WHEN FLAG 44 SET, THIS DIAGNOSTIC ROM WILL PREVENT THE 41C GOING INTO LIGHT SLEEP SO WE WE CAN MAKE THE PIL CHIP STAY ON ALL THE TIME. BUT THIS WILL MAKE THE 41C CONSTANTLY AWAKE AND DRAIN THE POWER IN A MUCH FASTER RATE TOO .

3.0 SENDING A FRAME BY ITS MNEMONIC

FOR CONVENIENCY, THERE IS A NUMBER OF FRAMES CAN BE SEND BY THEIR MNEMONIC. PRIOR TO SOURCE A FRAME, IF "ORAV" NOT GET SET WITHIN 10 SECONDS, IT WILL PRODUCE AN ERROR AND THE FRAME WILL NOT BE SENT. AFTER SENDING THE FRAME, IT WILL WAIT FOR THE FRAME COMES BACK AND DO THE ERROR CHECKING.

3.1 SENDING COMMAND FRAMES

THE FOLLOWING FUNCTIONS CAN BE USED TO SOURCE A COMMAND FRAME. SINCE ONLY CONTROLER CAN SOURCE COMMAND FRAME, SO "CA"(CONTROLER ACTIVE) IS AUTOMATICALLY SET TO 1 PRIOR SOURCING THE COMMAND FRAME. PRIOR TO SOURCE A COMMAND FRAME, IT ALWAYS CHECK "ORAV". IF "ORAV" NOT SET WITHIN 10 SECONDS, IT WILL GENERATE "ORAV = 0" ERROR. AFTER SENDING THE COMMAND FRAME, THE "RFC" WILL BE SOURCED AUTOMATICALLY BY THE CHIP. IF THE "RFC" DOES NOT RETURN WITHIN 10 SECONDS, IT WILL GENERATE "TIME OUT" ERROR.

AAU : (HEX 49A) AUTO ADDRESS UNCONFIGURE

CMD : SEND COMMAND
THE DIO8,DIO7,,,DIO1 OF THE THE COMMAND IS TAKEN FROM X-REG AS A DECIMAL NUMBER(0-255). FOR EXAMPLE: TO SEND COMMAND "DCL"(DEVICE CLEAR), SET X=20.

DDL : (HEX 4A0-4BF) SEND DEVICE DEPENDABLE LISTEN COMMAND
THE LOWER 5 BITS OF DDL COMMAND IS TAKEN FROM X-REG (0-31). IF X >= 32, WILL GENERATE "ADR ERR" MESSAGE. FOR EXAMPLE: TO SEND DDL 3, SET X = 3.

DDT : (HEX 4C0-4DF) SEND DEVICE DEPENDABLE TALK COMMAND
SAME AS DDL

GET : (HEX 408) GROUP EXECUTE TRIGGER

GTL : (HEX 401) GO TO LOCAL

IFC : (HEX 490) INTERFACE CLEAR
ONLY SYSTEM CONTROLER CAN SOURCE IFC, SO "SC" WILL BE SET TO 1 UPON EXECUTION OF THIS FUNCTION.

LAD : (HEX 420-43E) LISTEN ADDRESS
THE CONTENT OF X IS TAKEN AS THE ADDRESS NUMBER(<=30).

LPD : (HEX 49B) LOOP POWER DOWN

REN : (HEX 492) REMOTE ENABLE

SDC : (HEX 404) SELECTED DEVICE CLEAR

TAD : (HEX 440-45E) TALK ADDRESS
THE CONTENT OF X IS TAKEN AS THE ADDRESS NUMBER (<=30).

UNL : (HEX 437) UNLISTEN

UNT : (HEX 457) UNTALK

3.2 SENDING READY FRAMES
"ORAV" WILL BE CHECKED BEFORE SOURCING A READY FRAME AS IT DOES IN SOURCING A COMMAND FRAME. BUT THE TIME OUT AND ERROR CHECKING ARE NOT NECESSARILY PERFORMED, IT WILL DEPENDS ON WHAT FRAME IS BEING SOURCED."CA" IS ALWAYS SET TO PRIOR TO SEND A READY FRAME.

ASP : (HEX 580-59F) AUTO SIMPLE PRIMARY
THE CONTENT OF X IS TAKEN AS THE STARTING ADDRESS NUMBER(<=30). THE RETURN ADDRESS NUMBER IS PUSHED INTO STACK. "CA" IS SET TO 1 PRIOR TO SOURCE ASP.

THE FOLLOWING 4 FUNCTIONS "SDA", "SAC", "SID", "SSP" ARE CALL THE "SOT" FRAME (START OF TRANSMITION). IT ASSUME THE DEVICE IS A ADDRESSED TALKER AND WILL EAT THE "SOT" FRMAE AND START TO SOURCE DATA FRAME, IF THEY RECOGNIZED THESE FRAME. IF THE ADDRESSED TALKER DOES NOT RECOGNIZE THEM, IT SHOULD RETRANSMIT THE "SOT" FRAME.

SDA : (HEX 560) SEND DATA
THIS FUNCTION WILL ONLY SEND THE READY FRAME "SDA" BUT WILL NOT WAIT FOR ANY FRAME COMING BACK. "CA" IS SET TO 1 PRIOR TO SOURCE THE "SDA"

SAC : (HEX 563) SEND ACCESSORY BYTE
SET CA=1 AND SEND READY FRAME "SAC". THEN IT WILL READ THE INCOMMING DATA FRAMES UNTIL A NON-DATA FRAME COMES IN. ALL THE DATA FRAME WILL BE TREATED AS ONE BINARY NUMBER AND CONVERTED INTO A DECIMAL NUMBER AND PUSHED INTO STACK. IF THE LAST NON-DATA FRAME IS NOT THE "ETO", IT WILL GENERATE "EOT ERR". IF THE "SAC" BEEN RETRANSMITED BACK BY THE DEVICE, IT WILL GENERATE "NO RESPONSE" ERROR. IF NO FRAME COMES BACK WITHIN 10 SECONDS, WILL GENERATE "TIME OUT" THIS FUNCTION ASSUME THE DEVICE ALREADY BEEN ADDRESSED AS A TALKER. ERROR.

SID : (HEX 562) SEND DEVICE ID
SIMILER TO THE "SAC" FUNCTION EXCEPT THE DEVICE ID WILL BE STORED IN ALPHA REGISTER.

SSP : (HEX 561) SEND SERIEL POLL
SIMILER TO THE "SAC" FUNCTION EXCEPT THE READY FRAME SEND IS "SSP".

TCT : (HEX 564) TAKE CONTROL
THE WAY PASS CONTROL WORKS IS THAT THE ACTIVE CONTROLLER FIRST ADDRESS THE POTENTIAL CONTROLLER AS A TALKER, AND THEN SEND IT THIS "TCT" READY FRAME. IF THE POTENTIAL CONTROLLER WILL TAKE THE CONTROL, IT WILL EAT THE "TCT" FRAME AND START TO SOURCE ITS FIRST COMMAND FRAME. IF THE POTENTIAL CONTROLLER RETRANSMIT THE "TCT" READY FRAME, THAT MEANS IT WILL NOT TAKE THE CONTROL.

THE "TCT" FUNCTION ASSUMES THE POTENTIAL CONTROLLER ALRAEDY BEEN ADDRESSED AS A TALKER, SO IT START WITH SENDING THE "TCT" FRAME. THEN IT WILL WAIT FOR A INCOMMING FRAME. IF THE INCOMMING FRAME IS A COMMAND FRAME, IT WILL PUT THE COMMAND FRAME IN X & Y

REGISTER AS IT DOES IN "RFRM" FUNCTION. IF THE "TCT" COMES BACK OR NO FRAME COMES IN WITHIN 10 SECONDS, IT WILL GENERATE "NO RESPONSE" ERROR.

NRD : (HEX 542) NOT READY FOR DATA
THIS FUNCTION WILL SEND A SERIES OF FRAMES AND DO THE ERROR CHECKING. IT WILL FIRST READ THE DATA FRAME SUPPOSEDLY IN REGISTER 2 AND SAVE IT. THEN SOURCE THE NRD FRAME. WHEN THE NRD COMES BACK, IT WILL RETRANSMIT THE SAVED DATA FRAME, AND THEN EXPECT AN ETO/ETE COMES BACK. IF THE LAST FRAME RETURN IS NOT AN ETO, IT WILL GENERATE "EOT ERR".

3.3 SENDING IDY FRAME

1. IDY : SEND OUT IDY FRAME WITH DIO8-DIO1 EQUAL TO ZEROS.
"CA" IS SET TO 1 PRIOR SOURCING THE IDY FRAME. WHEN IT COMES BACK, DIO8-DIO1 ARE CONVERTED INTO A DECIMAL NUMBER AND PUSHED INTO STACK.

3.4 SEND/READ DATA FRAMES

OUTBIN : OUTPUT BINARY IN X
CONVERT THE NUMBER IN X TO BINARY AND SEND IT OUT AS DATA FRAMES. THE NUMBER OF BYTES TO SEND IS THE MINIMUM NUMBER OF BYTES WHICH CAN COVER THE X IN BINARY. IF THE CONTENT OF X IS A STRING, IT WILL ONLY SEND THE NON-NULL BYTES. IT ASSUMES THE DEVICE ALREADY BEEN ADDRESSED AS A LISTENER. "TA" IS SET TO 1 AND CHECK "ORAV" & "FRNS" PRIOR SOURCING THE DATA FRAMES. IF TRANSMIT ERROR IS DETECTED, IT WILL PRODUCE AN ERROR. AGAIN, THE TIME OUT IS SET TO 10 SECONDS. NO ETO/ETE WILL BE SENT AT THE END OF TRANSMITION.

OUTBINY : OUTPUT BINARY IN X AND COUNTED BY Y
SAME AS OUTBIN EXCEPT THE NUMBER OF BYTES TO SEND IS CONTROL BY THE NUMBER IN Y. BUT THE MAXIMUM NUMBER OF BYTES IS 7.

INBIN : INPUT BINARY TO X
SET LA=1 AND WILL SEND READY FRAME "SDA" PRIOR TO READ ANY FRAME. THIS FUNCTION WILL KEEP READING THE DATA FRMAE UNTIL A NON-DATA FRAME COMES IN. BUT IT WILL ONLY CONVERT THE LAST 7 BYTES OF DATA FRAME TO A DECIMAL NUMBER AND PUSH IT INTO STACK. IF THE LAST FRAME IS NOT AN "ETO", IT WILL PRODUCE "EOT ERR" MESSAGE. BUT THE X-REG WILL STILL BE CHANGED EVEN AN ERROR IS DETECTED. IF NO FRAME COMES IN WITHIN 10 SECONDS, WILL GENERATE "TIME OUT" ERROR.

4.0 HOW THE ERROR WILL AFFECT THE PROGRAM EXECUTION

EOT

THERE ARE FIVE KINDS OF ERROR : "TIME OUT", "FRNS ERR", "EIO ERR", "NO RESPONSE" AND "ORAV = 0" ARE HANDLED DIFFERENTLY FROM THE WAY THE 41C HANDLES ITS ERROR. IT CAN BE DESCRIBED BY THE FOLLOWING TABLE :

	FLAG 25 SET	FLAG 25 CLEAR
EXECUTE FROM KEY BOARD	X=NEGATIVE ERROR CODE DISPLAY ERROR MESSAGE WILL NOT CLEAR FLAG 25	X=NEGATIVE ERROR CODE DISPLAY ERROR MESSAGE
EXECUTE FROM PROGRAM EXEC-UTION OR SST	X=NEGATIVE ERROR CODE WILL NOT CLEAR FLAG 25 PROGRAM CONTINUE	X=NEGATIVE ERROR CODE DISPLAY ERROR MESSAGE STOP PROGRAM EXECUTION

THE FIVE ERROR CODES ARE :

-1 : TIME OUT ERROR. TIME OUT IS ALWAYS SET TO 10 SECONDS

-2 : FRNS ERROR(FRAME RETURN NOT AS SEND)

-3 : EOT ERROR(END OF TRANSMITION ERROR)

-4 : NO RESPONSE ERROR. DEVICE DOES NOT RESPOND TO "SAC", "SID", "SSP", OR "TCT".

-5 : ORAV = 0 ERROR. PRIOR TO SOURCE A FRAME, "ORAV" DOES NOT GET WITHIN 10 SECONDS.

5.0 BUFFER OPERATION

A ONE DIMENSIONAL BYTE ARRAY CAN BE SET UP AS A BUFFER TO READ/WRITE A LARGER NUMBER OF FRAMES. A POINTER WILL BE USED AS THE INDEX OF THE BUFFER, WHICH CAN POINT TO ANY BYTE IN THE BUFFER.

SETBUFX : SET UP THE BUFFER BY X

THE CONTENT OF X IS TAKEN AS THE BUFFER SIZE IN BYTES. IF X = 0 THE EXISTING BUFFER, IF THERE IS ONE, WILL BE DISLOCATED. THE BUFFER IS CREATED AS AN I/O BUFFER IN 41C'S SPARE USER MEMORY. SO THE BUFFER SIZE IS LIMITED TO 1771 BYTES. BESIDES, IF THE BUFFER SIZE IS NOT EXACTLY AN MULTIPLE OF 7, IT WILL BE FILLED UP TO MULTIPLE OF 7 BYTES. IF THE 41C DOES NOT HAVE THE ENOUGH MEMORY TO ALOCATE THE BUFFER, IT WILL PACK THE MEMORY AND ASK THE USER TO TRY AGAIN.

PT= : SET THE POINTER TO A GIVEN BYTE NUMBER.

THE CONTENT OF X IS TAKEN AS THE POINTER VALUE. THE POINTER IS STARTING FROM ZERO. IF THE POINTER IS ASKED TO SET BEYOND THE END OF BUFFER, IT WILL PRODUCE AN ERROR.

PT=? : ASK THE PRESENT VALUE OF THE POINTER

THE PRESENT VALUE OF THE POINTER WILL BE PUSHED INTO STACK.

AIPT : AUTO ADVANCE POINTER AFTER EACH BUFFER OPERATION

THE BUFFER POINTER CAN BE AUTOMATICALLY ADVANCED AFTER EACH BUFFER OPERATION BY SELECTING AUTO ADVANCE MODE. THIS MODE WILL STAY IN EFFECT UNTIL THE OTHER FUNCTION "MIPT" IS EXECUTED.

MIPT : MANUAL ADVANCE POINTER

WHILE IN MANUAL ADVANCE MODE, POINTER WILL STAY WHERE IT WAS AFTER EACH BUFFER OPERATION. THE POINTER WILL ONLY BE MOVED BY THE FUNCTION "PT=". WHEN THE BUFFER FIRST SET UP, IT IS ASSUMED IN MANUAL ADVANVE MODE.

X-BUF : STORE X TO BUFFER

THE CONTENT OF X WILL BE CONVRETED INTO BINARY AND STORED INTO BUFFER STARTING FROM WHERE THE POINTER IS. THE NUMBER OF BYTES IS DETERMINED BY THE VALUE OF X. IF X HAS A STRING, IT WILL SUPRESS NULL BYTES AND NO CONVERSION WILL BE DONE.

BUF-XA : CONVERT ASCII CODED DECIMAL NUMBER FROM BUFFER TO X

THE NUMBER IN X-REG IS TAKEN AS THE NUMBER OF BYTS TO CONVERT. STARTING FROM WHERE THE POINTER IS, CONVERT X NUMBER OF BYTES, DECIMAL NUMBER IN ITS ASCII FORM, INTO A DECIMAL NUMBER AND PUSH IT INTO STACK. IT WILL BE TERMINATED BY "CR" &"LF".

BUF-XB : CONVERT BINARY TO DECIMAL FROM BUFFER

SIMILAR TO BUF-XA, EXCEPT THE BYTES PICK UP FROM BUFFER ARE TREATED AS A BINARY NUMBER.

A-BUF : STORE ALPHA REGISTER TO BUFFER

STORE THE CONTENT OF ALPHA REGISTER TO BUFFER STARTING FROM WHERE THE POINTER IS. IF BUFFER OVERFLOWED, IT WILL PRODUCE AN ERROR.

BUF-AX : RESTORE ALPAH REGISTER FROM BUFFER BY X

RESTORE X NUMBER OF BYTES FROM BUFFER TO ALPHA REGISTERE STARTING FROM THE POINTER. THE CONTENT OF X IS TAKEN AS BYTE COUNT. THIS

FUNCTION WILL ALSO BE PRE-TERMINATED BY "CR","LF" TOO. IT WILL NOT CLEAR ALPHA REGISTER BEFORE EXECUTION.

INBUFX : INPUT DATA FRAMES INTO BUFFER BY X
X IS TAKEN AS BYTE COUNT. SET LA=1 AND WILL SEND "SDA" PRIOR TO READ. READS X NUMBER OF DATA FRAMES AND STORE THEM TO BUFFER STARTING FROM THE CURRENT POINTER. THE EXECUTION WILL BE TERMINATED AS FOLLOWAS:

1). BY A NON-DATA FRAME, BUT IF IT IS NOT AN "EIO" WILL GENERATE AN ERROR.

2). READ IN X NUMBER OF BYTES. AND IT WILL SEND THE "NRD" AFTER LAST BYTE.

3). REACHED END OF BUFFER. AN ERROR WILL BE GENERATED. TIME OUT IS SET TO 60 SECONDS FOR THIS FUNCTION, BUT ANY KEY DOWN WILL SHORT CIRCUIT THE TIME OUT AND GENERATE "TIME OUT" ERROR.

OUTBUFX : OUTPUT DATA FRAME FROM BUFFER BY X
X IS TAKEN AS BYTE COUNT. SET TA=1 PRIOR TO EXECUTION. THEN SENDS X NUMBER OF DATA FRAMES FROM BUFFER STARTING FROM THE CURRENT POINTER POSITION. THIS FUNCTION WILL HANDLE "NRD" PROPERLY IN MIDDLE OF THE TRANSMITION AND WILL SEND AN EIO/EIE AT THE END OF TRANSMITION.

RG-BUFX : COPY REGISTERS TO BUFFER BY X
THE CONTENT OF X IS TAKEN AS THE REGISTER INDEX IN THE bbb.eee FORM, WHERE bbb IS THE BEGINNING REGISTER NUMBER AND eee IS THE ENDING REGISTER NUMBER. THE COPY WILL START FROM WHERE THE POINTER IS. IF THE BUFFER IS OVERFLOWED, IT WILL PRODUCE AN ERROR. IF ALL OR PARTIAL OF THE DATA REGISTER IS NOT EXIST, IT WILL PRODUCE AN ERROR. THE REGISTER WILL BE COPIED EXACTLY THE WAY IT IS, NO CONVERSION WILL BE MADE.

BUF-RGX : COPY BUFFER TO REGISTERS BY X
SAME AS "RG-BUF" EXCEPT THE DIRECTION IS REVERSED.

X=BUF? : COMPARE X WITH BUFFER
IF X HAS A NUMBER, IT WILL CONVERT THE ABS(X) INTO A BINARY NUMBER AND COMPARE THE SAME NUMBER OF BYTES WITH THE BUFFER STARTING FROM THE CURRENT POINTER. IF X HAS A STRING, IT WILL COMPARE ALL THE NON-NULL BYTES WITH THE SAME NUMBER OF BYTES IN BUFFER. LIKE THE OTHER COMPARE FUNCTION, IT WILL DISPLAY "YES" OR "NO", OR SKIP THE NEXT STEP.

A=BUFX? : COMPARE ALPHA REGISTER TO BUFFER BY X
COMPARE X BYTES OF ALPHA REGISTER WITH BUFFER STARTING FROM POINTER. IF X IS LARGER THAN THE LENGTH OF ALPHA REGISTER, IT WILL ONLY COMPARE TO THE END OF ALPHA REGISTER.

A=BUF? : COMPARE ALPHA REGISTER TO BUFFER
SAME AS "A=BUFX?" EXCEPT IT ALWAYS COMPARE FROM THE BEGINNING TO THE END OF THE ALPHA REGISTER WITH THE SAME NUMBER OF BYTES IN BUFFER.

RG=BUF? : COMPARE REGISTERS TO BUFFER
THE CONTENT OF X IS TAKEN AS REGISTER INDEX IN THE bbb.eee FORM.

THE BLOCK OF REGISTERS WILL BE COMPARED WITH THE SAME NUMBER OF BYTES IN THE BUFFER STARTING FROM THE CURRENT POINTER POSITION. EVERY REGISTER IS DIRECTLY COMPARED WITH 7 BYTES IN THE BUFFER, NO CONVERSION WILL BE DONE.

6.0 SCOPE MODE

6.1 SCOPE MODE FUNCTIONS

SCOPE : PUT THE 41C INTO SCOPE MODE
IN SCOPE MODE, THE 41C WILL NO LONGER SOURCE FRAMES, BUT IT CAN DISPLAY OR BUFFER ANY FRAME PASSING BY. WHEN ENTERING THE SCOPE MODE, "TA" & "LA" ARE SET TO 1. ALL THE FRAME WILL BE SHOWN IN ITS MENUMONIC. THE DELAY OF EACH FRAME CAN BE CONTRCLED BY THE DIGIT KEYS 0,1,2 AND 3. BUT THE IDY FRAME WILL ALWAYS USE ZERO DELAY. WHILE ANY KEY IS DOWN, THE TRANSMITION WILL BE HALT UNTIL THE KEY IS UP AGAIN.

MONITOR : PUT 41C INTO MONITOR MODE
THE MONITOR MODE IS LITTLE DIFFERENT FROM THE SCOPE MODE. IN MONITOR MODE THE 41C CAN STILL BE USED AS A REGULAR 41C. ALL THE MONITOR MODE DOES IS SET TA & LA TO 1 AND NEVER LET 41C GO INTO LIGHT SLEEP, AND IT CONSTANTLY CHECK IF THERE IS ANY FRAME COMES IN. WHEN THERE IS A FRAME COMES IN, IT WILL READ IT AND PUT IT TO 41C X & Y REGISTER LIKE THE "RFRM" FUNCTION DOES AND IT WILL SHOW THE FRANE IN THE DISPLAY. BUT THE FRAME WILL NOT BE RETRANSMITED AUTOMATICALLY. THE USER WILL HAVE TO EXECUTE THE "WFRM" FUNCTION TO RETRANSMIT THE FRAME. TO GET OUT OF THE MONITOR MODE, THE USER CAN EITHER CLEAR TA OR LA, OR WRITE 00 TO REGISTER 5 OF THE PIL CHIP.

PRFRMS : PRINT FRAMES IN BUFFER
PRINT THE FRAMES IN BUFFER STARTING FROM WHERE THE POINTER IS. EVERY FRAME IS USED TWO BYTES TO STORE IN BUFFER, SO THE POINTER NEED TO BE SET TO THE FIRST BYTE OF A FRAME.
THE PRINTING WILL BE STOPPED BY EITHER REACHING THE END OF BUFFER OF BY HITTING THE "R/S" KEY. IF ANY OTHER KEY IS PUSHED DURING THE PRINTING WILL SLOW DOWN THE PRINTING TO ABOUT 2 SECONDS PER FRAME. SO IF THERE IS NO PRINTER PLUG-IN, THE USER CAN USE THIS FEATURE TO SLOW DOWN THE DISPLAY TO A READABLE RATE.

PRBYTES : PRINT BYTES IN BUFFER
SIMILAR TO THE "PRFRMS" EXCEPT IT PRINTS EVERY SINGLE BYTE IN THE BUFFER STARTING FROM THE CURRENT POINTER.

6.2 SCOPE MODE KEY BOARD

WHEN GOES INTO SCOPE MODE, THE ROM WILL TAKE THE FULL CONTROL OF THE KEY BOARD. THE 41C WILL NEVER GO TO LIGHT SLEEP NOR THE CONTROL WILL PASS BACK TO THE 41C UNTIL EXIT FROM SCOPE MODE. ONLY THE FOLLOW KEYS WILL BE ENABLED IN SCOPE MODE:

1. "STO" KEY
THE STO KEY IS USED AS A TOGGLE KEY TO GO IN OR OUT OF BUFFER MODE. WHEN IT IS IN BUFFER MODE, ALL THE FRAMES PASSED BY WILL BE STORED INTO BUFFER STARTING FROM THE POINTER. TWO BYTES ARE REQUIRED FOR EVERY FRAME. WHILE IN BUFFER MODE, THE FRAME WILL STILL BE SHOWN AND THE DELAY SET PREVIOUSLY WILL STILL VALID. WHEN FILL UP TO THE LAST BYTE IN BUFFER, A MESSAGE "BUFFER FULL" WILL BE SHOWN IN DISPLAY, THE TRANSMITION WILL BE HALT AT THIS POINT. TWO THINGS THE USER CAN DO TO RESUME THE TRANSMITION:

1). HIT THE "STO" KET TO GET OUT OF BUFFER MODE.
2). HIT "BACK ARROW" KEY TO RESET THE POINTER TO ZERO.

2. DIGIT KEY "0"
SET THE FRAME DELAY TO ZERO. IN ORDER TO SHOW THE TRANSMITION THE FRAME WILL STILL BE ROLLING INTO THE DISPLAY, BUT IT WOULD JUST TOO FAST TO TELL WHAT IT IS.

3. DIGIT KEY "1"
SET FRAME DELAY TO 1 SECOND

4. DIGIT KEY "2"
SET FRAME DELAY TO 2 SECONDS

5. DIGIT KEY "3"
SET FRAME DELAY TO 3 SECONDS

6. "BACK ARROW" KEY
RESET THE BUFFER POINTER TO ZERO

7. "SST" KEY
MOVE THE POINTER TO NEXT FRAME AND DISPLAY IT. A MESSAGE "END OF BUFFER" WILL BE SHOWN WHEN REACH THE LAST FRAME IN THE BUFFER. FOR EVERY "SST", THE POINTER IS ADVANCED TWO BYTES.

8. "BST" KEY (SHIFT,SST)
MOVE THE POINTER BACK ONE FRAME AND DISPLAY IT

9. "R/S" KEY
EXIT FROM SCOPE MODE. WHEN EXIT FROM SCOPE, TA & LA ARE SET TO ZERO. IF THERE IS NO KEY DOWN AND NO FRAME TRANSMITING FOR 10 MINUTES, THE SCOPE MODE IS AUTOMATICALLY TERMINATED. WHENEVER EXIT THE SCOPE MODE, THE BUFFER POINTER WILL PROPERLY UPDATED IF IT IS IN POINTER AUTO ADVANCE MODE.

10. "OFF" KEY
CALCULATOR OFF

7.0 RECEIVING SERVICE REQUEST

THERE IS A WAY PROVIDED BY THIS ROM TO HANDLE SERVICE REQUEST AS AN INTERRUPT. THE 41C IS CAPABLE OF AUTOMATICALLY SOURCING IDY FRAME WHILE IT IS IN LIGHT SLEEP. IF THE 41C IS WAKEN UP FROM LIGHT SLEEP BY A SERVICE REQUEST, THE PRINTER CODE AT LOCATION @60000 WILL HAVE THE FIRST CHANCE TO SEE IF IS THE PRINTER REQUESTING SERVICE. IF IT IS, IT WILL SERVICE THE PRINTER. IF IT IS NOT THE PRINTER REQUESTING SERVICE, THEN THE FOLLOWING PLUG-IN ROM WILL HAVE THE CHANCE TO HAVE THE CONTROL. WHENEVER THIS ROM HAVE A CHANCE TO SERVICE A SERVICE REQUSET, IT WILL FIRST LOOK AT THE INTERRUPT ENABLE FLAG WHICH IS THE USER FLAG 18. IF THE USER FLAG 18 IS SET, THEN IT WILL TRY TO EXECUTE THE SERVICE ROUTINE. IF IT FINDS THE SERVICE ROUTINE, IT WILL JUMP TO START EXECUTING THE SERVICE ROUTINE. THE FIRST LINE OF THE SERVICE ROUTINE HAS TO AN ALPHA LABEL "SRQR". THAT IS THE UNIQUE LABEL IT WILL BE SEARCHING FOR. SO IF EITHER THE USER FLAG 18 IS NOT SET OR THE LABEL "SRQR" NOT FOUND, IT WILL IGNORE THE SERVICE REQUSET.

7.0 UTILITY FUNCTIONS

IOFSX? : TEST 41C CPU IO/FLAG BY X

THERE ARE 14 (0-13) HADRWARE IO/FLAG IN 41C CPU WHICH CAN BE TEST FOR SET OR CLEAR BY THIS FUNCTION. IF EXECUTE FROM KEY BOARD, IT WILL DISPLAY "YES" OR "NO". IF EXECUTE FRUM PROGRAM, IT WILL SKIP NEXT THE NEXT STEP IF THE FLAG IS SET. THE CONTENT OF X IS USED TO INDICATE WHICH FLAG TO TEST.

ROMCHKX : VERIFY PLUG-IN ROM CHECKSUM BY X

X IS TAKEN AS THE ROM ID OF A PLUG-IN ROM. FOR EXAMPLE: THE ROM ID OF THE PIL MODULE IS 28 & 29. WHILE COMPUTING THE ROM CHECKSUM, THE DISPLAY WILL SHOW "DD CC-NN TST" WHERE DD IS THE ROM ID, CC-NN IS THE ROM LABEL. WHEN THE CHECKSUM COMPUTATION IS DONE, THE DISPLAY WILL CHANGE TO "DD CC-NN OK" OR "DD CC-NN BAD". IF THE FUNCTION WAS EXECUTED FROM PROGRAM EXECUTION, IT WILL SKIP THE NEXT STEP IF THE CHECKSUM IS GOOD.

X<>FLAG : TAKE THE INT(X) AND EXCHANGE IT WITH USER FLAG 0-7

CONVERT INTEGER PART OF X (0-255) AND SAVE IT IN USER FLAG 0-255, THEN CONVERT THE USER FLAG 0-7 AS A DECIMAL NUMBER AND PUSH IT INTO STACK.