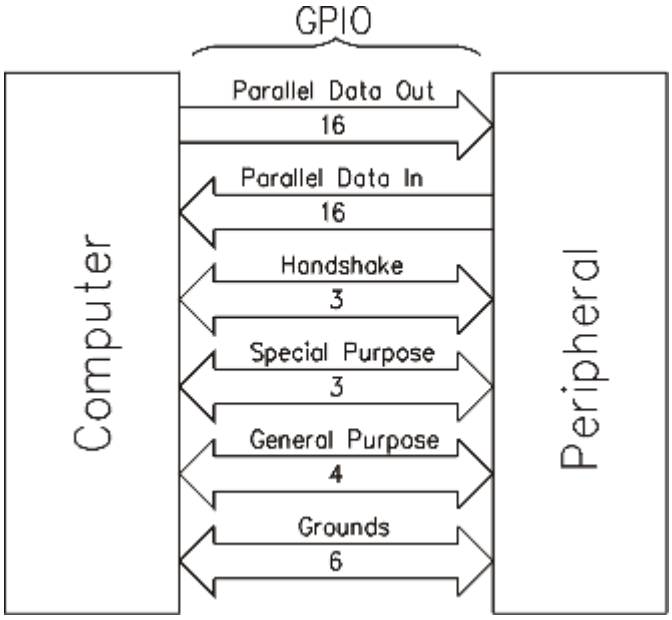
Contents GPIO Interface

{button www.htbasic.com,Inet("www.htbasic.com")}

Chapter 1 Overview and Configuration

The function of the General Purpose Input Output (GPIO) interface is to provide a means of communication between a peripheral device and a computer.



Block Diagram of the GPIO Interface

The GPIO interface uses a 50 pin connector to communicate with the peripheral. Thirty-two lines are used for data input and output; sixteen for output (DO0 - DO15) and sixteen for input (DI0 - DI15).
Three lines are used for handshaking purposes. They are the Peripheral Control line (PCTL), Peripheral Flag line (PFLG), and Input/Output line (IO).

The three Special Purpose lines are External Interrupt Request (EIR), Peripheral Status (PSTS), and Peripheral Reset (PRESET).

The four General Purpose lines are CTL0, CTL1, STI0, and STI1.

Six ground lines are provided for a ground reference and safety.

The last two lines are not connected.

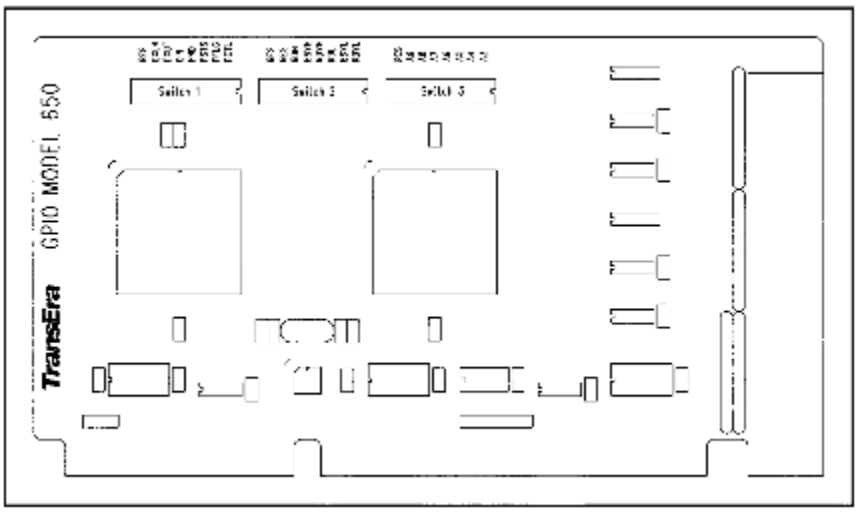
Throughout this manual an output refers to a transfer of data from the computer to the peripheral while an input is a transfer from the peripheral to the computer. This notation applies to the data bus and the signal lines.

This manual has information to configure both the GPIO 650 and the GPIO 600.

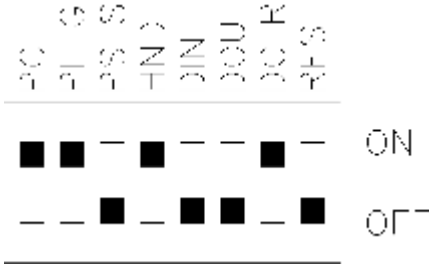
GPIO 650 Hardware Setup

The GPIO 650 board has three switches that are used to configure the board. An understanding of the GPIO interface is necessary to properly set the switches. If you have not previously used the GPIO interface please study the Interface Description section before proceeding with the installation.

Switches labeled RES are reserved and currently unused. The switch is ON when it is toward the PC board and OFF when it is away from the board.



Location of Switches for the GPIO 650



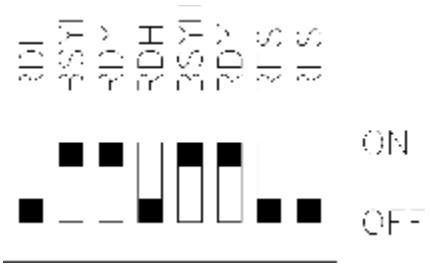
GPIO 650 Switch 1

GPIO 650 Switch One - Interface Settings

This switch selects the signal polarity, type of handshaking and the state of the output data lines after reset. The table shows the effect of each switch setting.

Interface Settings for GPIO 650

Name of Switch Position	PCTL	PFLG	PSTS	HND	DIN	DOUT	DCLR
Function	Set Polarity of PCTL	Set Polarity of PFLG	Set Polarity of PSTS	Full/Pulse Handshake	Set Polarity of Data In	Set Polarity of Data Out	Clear DOUT on Reset
ON	Low = Set	Low = Rdy	Low = OK	Full	Low = 1	Low = 1	Yes
	High = Clr	High = Bsy	High = OK		High = 0	High = 0	
OFF	Low = Clr	Low = Bsy	Low = OK	Pulse	Low = 0	Low = 0	No
	High = Set	High = Rdy	High = OK		High = 1	High = 1	



GPIO 650 Switch 2

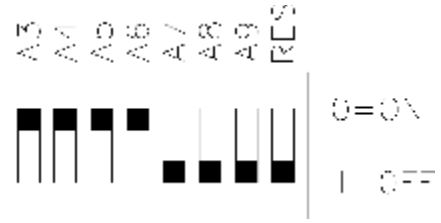
GPIO 650 Switch Two - Input Clock Source

Switch two selects the latch source that will clock the data into the input data buffers. The switch for the **desired clock should be off** with the other two on. The clocks for the high and low bytes do not need to be the same.

The RD clock source causes the computer to clock the input buffers whenever it is reading the data. The BSY clock source causes the input buffers to be loaded on the ready-to-busy transition of PFLG. The RDY clock source uses the busy-to-ready transition of PFLG.

By default these settings select the input clock source. Software can override these settings. If only software control is used for the input clock source selection this switch does not need to be set.

For more information on GPIO input clocking, refer to the Interface Description section for a more detailed explanation.



GPIO 650 Switch 3

GPIO 650 Switch Three - I/O Address

This switch is used to select the I/O address of the board. The I/O address is the base address that the computer uses to communicate with the board. The GPIO 650 board uses 8 address locations above and including the base address. The address space of the GPIO board cannot be shared with any other peripheral. The default address of the GPIO board is 380h.

The following table shows the switch settings with the corresponding hexadecimal board address.

Address Selection for GPIO 650

Switches A9 A3	I/O Address (hex)	Switches A9 A3	I/O Address (hex)
1 0 0 0 0 0 0	2 0 0	1 1 0 0 0 0 0	3 0 0
1 0 0 0 0 0 1	2 0 8	1 1 0 0 0 0 1	3 0 8
1 0 0 0 0 1 0	2 1 0	1 1 0 0 0 1 0	3 1 0
1 0 0 0 0 1 1	2 1 8	1 1 0 0 0 1 1	3 1 8
1 0 0 0 1 0 0	2 2 0	1 1 0 0 1 0 0	3 2 0
1 0 0 0 1 0 1	2 2 8	1 1 0 0 1 0 1	3 2 8
1 0 0 0 1 1 0	2 3 0	1 1 0 0 1 1 0	3 3 0

1 0 0 0 1 1 1	2 3 8	1 1 0 0 1 1 1	3 3 8
1 0 0 1 0 0 0	2 4 0	1 1 0 1 0 0 0	3 4 0
1 0 0 1 0 0 1	2 4 8	1 1 0 1 0 0 1	3 4 8
1 0 0 1 0 1 0	2 5 0	1 1 0 1 0 1 0	3 5 0
1 0 0 1 0 1 1	2 5 8	1 1 0 1 0 1 1	3 5 8
1 0 0 1 1 0 0	2 6 0	1 1 0 1 1 0 0	3 6 0
1 0 0 1 1 0 1	2 6 8	1 1 0 1 1 0 1	3 6 8
1 0 0 1 1 1 0	2 7 0	1 1 0 1 1 1 0	3 7 0
1 0 0 1 1 1 1	2 7 8	1 1 0 1 1 1 1	3 7 8
1 0 1 0 0 0 0	2 8 0	1 1 1 0 0 0 0	3 8 0
1 0 1 0 0 0 1	2 8 8	1 1 1 0 0 0 1	3 8 8
1 0 1 0 0 1 0	2 9 0	1 1 1 0 0 1 0	3 9 0
1 0 1 0 0 1 1	2 9 8	1 1 1 0 0 1 1	3 9 8
1 0 1 0 1 0 0	2 A 0	1 1 1 0 1 0 0	3 A 0
1 0 1 0 1 0 1	2 A 8	1 1 1 0 1 0 1	3 A 8
1 0 1 0 1 1 0	2 B 0	1 1 1 0 1 1 0	3 B 0
1 0 1 0 1 1 1	2 B 8	1 1 1 0 1 1 1	3 B 8
1 0 1 1 0 0 0	2 C 0	1 1 1 1 0 0 0	3 C 0
1 0 1 1 0 0 1	2 C 8	1 1 1 1 0 0 1	3 C 8
1 0 1 1 0 1 0	2 D 0	1 1 1 1 0 1 0	3 D 0
1 0 1 1 0 1 1	2 D 8	1 1 1 1 0 1 1	3 D 8
1 0 1 1 1 0 0	2 E 0	1 1 1 1 1 0 0	3 E 0
1 0 1 1 1 0 1	2 E 8	1 1 1 1 1 0 1	3 E 8
1 0 1 1 1 1 0	2 F 0	1 1 1 1 1 1 0	3 F 0
1 0 1 1 1 1 1	2 F 8	1 1 1 1 1 1 1	3 F 8

GPIO 650 Interrupt and DMA Channels

Interrupts and DMA are used to assist the transferring of data. There are a fixed number of interrupt and DMA channels on a PC and they cannot be shared with other devices. The GPIO 650 enables the channel(s) for use with software so no hardware settings are required.

Interrupt Channels 5, 7, 9, 10, 11, 12, 15

DMA Channels 5, 6, 7

The board can use one interrupt channel and one DMA channel. An interrupt must be selected for DMA to be used. These channels numbers are given to the software driver during the initialization procedure. If there are no interrupt or DMA channels allocated to the board it can still be used to transfer data but it's performance will decrease and interrupts will not be available.

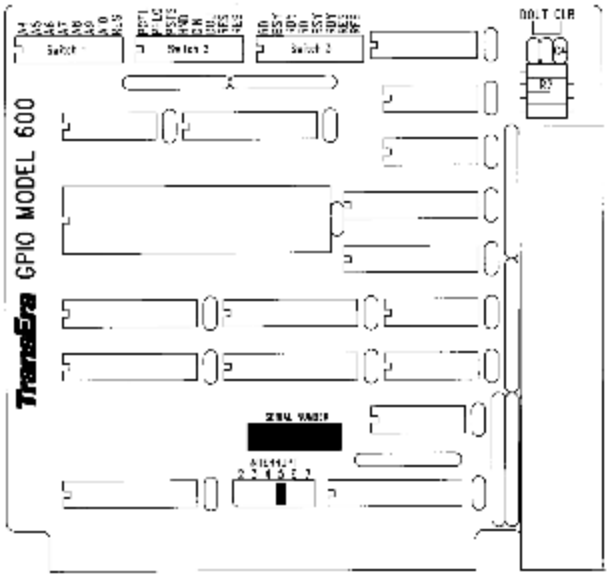
The three DMA channels are the 16-bit DMA channels available on a PC. These channels do not support 8-bit transfers. To handle this limitation the GPIO 650 was designed to pack two 8-bit peripheral handshakes into a single DMA transfer. This not only allows 8-bit peripheral transfers but also doubles the effective 8-bit transfer rate of the GPIO board.

More detailed information on DMA transfers is in Interface Description under DMA Transfers.

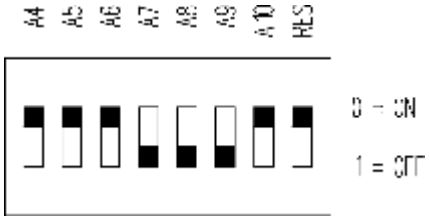
GPIO 600 Hardware Setup

The GPIO 600 board has three switches that are set to configure the board. An understanding of the GPIO interface is necessary to properly set the switches. If you have not previously used the GPIO interface please study the *Interface Description* section before proceeding with the installation.

Switches labeled RES are reserved. The switch is ON when it is toward the PC board and OFF when it is away from the board.



Location of Switches and Jumpers on the GPIO 600



GPIO 600 Switch 1
GPIO 600 Switch One - I/O Address

This switch is used to select the I/O address of the board. The I/O address selected by this switch is the base address that the computer uses to communicate with the board. The GPIO board uses 16 address locations above the base address. The address space of the GPIO board cannot be used by any other peripheral. The default address of the GPIO board is 380h.
The following table shows the switch settings with the corresponding hexadecimal board address.

Address Selection for GPIO 600

Switches		I/O Address
A10	**A4**	**(hex)**
0	1 0 0 0 0 0 0	2 0 0
0	1 0 0 0 0 0 1	2 1 0
0	1 0 0 0 0 1 0	2 2 0
0	1 0 0 0 0 1 1	2 3 0
0	1 0 0 1 0 0	2 4 0

0100101	250
0100110	260
0100111	270
0101000	280
0101001	290
0101010	2A0
0101011	2B0
0101100	2C0
0101101	2D0
0101110	2E0
0101111	2F0
0110000	300
0110001	310
0110010	320
0110011	330
0110100	340
0110101	350
0110110	360
0110111	370
0111000	380
0111001	390
0111010	3A0
0111011	3B0
0111100	3C0
0111101	3D0
0111110	3E0
0111111	3F0

GPIO 600 Switch 2

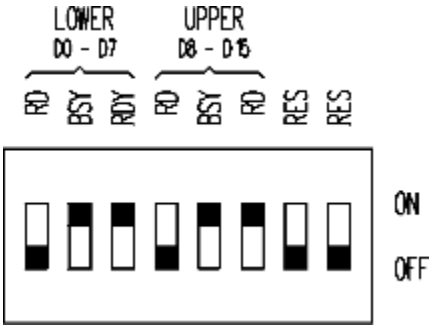
GPIO 600 Switch Two - Interface Settings

This switch selects the signal polarity and the type of handshaking that will be used. The chart

below shows the switch settings for handshaking and polarity.

Interface Settings for GPIO 600

Name of Switch Position	PCTL	PFLG	PSTS	HND	DIN	DOUT	RES
Function	Set Polarity of PCTL	Set Polarity of PFLG	Set Polarity of PSTS	Full/Pulse Handshake	Set Polarity of Data In	Set Polarity of Data Out	Reserved
ON	Low = Set	Low = Rdy	Low = OK	Full	Low = 1	Low = 1	
	High = Clr	High = Bsy	High = OK		High = 0	High = 0	
OFF	Low = Clr	Low = Bsy	Low = OK	Pulse	Low = 0	Low = 0	
	High = Set	High = Rdy	High = OK		High = 1	High = 1	



GPIO 600 Switch 3

GPIO 600 Switch Three - Input Clock Source

This switch selects the latch source that will clock the data into the input data buffers. The switch for the desired clock should be off with the other two on. The clocks for the high and low bytes do not need to be the same.

The RD clock source causes the computer to clock the input buffers whenever it is reading the data. The BSY clock source causes the input buffers to be loaded on the ready-to-busy transition of PFLG. The RDY clock source uses the busy-to-ready transition of PFLG.

For more information on GPIO input clocking, refer to the Interface Description section for a more detailed explanation.

Interrupt Jumper

The GPIO 600 supports interrupts 2 through 7. The default interrupt is 5. The location of the interrupt selector pins can be found on the board diagram at the start of this section. Install the shorting block on the pins that corresponds to the desired interrupt number.

DOUT Clear Jumper

If the DOUT Clear jumper is set, the output data lines will be set low after a board reset. The location of the DOUT Clear jumper can be seen on the board diagram at the start of this section.

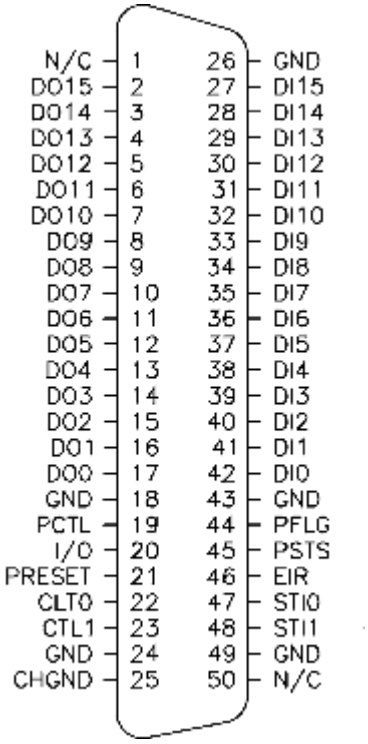
Board Installation

1. Turn off and unplug the computer
2. Remove the computer cover.

3. Locate an empty slot and take off the slot cover by removing the screw. Keep the screw.
4. Eliminate any static electricity by touching the metal computer case.
5. Remove the GPIO board from the static protection pouch and insert it into the selected slot. The board should be handled by its edges only.
6. Replace the screw to hold the board in place.
7. Replace the computer cover.

Chapter 2 Interface Description

This chapter describes the operation of the GPIO interface.



External Connections to GPIO

Signal Descriptions

All of the signal lines of the GPIO board are described below. The numbers in parentheses are the pin numbers on the connector.

Parallel Data Out

Data Output Lines (2-17) - These are the sixteen data output lines. The computer writes to these lines while the peripheral reads them. The logic sense of these lines is user configurable.

Parallel Data In

Data Input Lines (27-42) - These are the sixteen data input lines. The peripheral writes to these lines while the computer reads them. The logic sense of these lines is user configurable.

Handshake

Peripheral Control (19) - PCTL - This is the control line that the computer sets to initiate a transfer of data. The logic sense of this line is user configurable.

Input Output (20) - IO - This line indicates the direction of data flow to the peripheral. A high signal indicates an input while a low signal indicates an output.

Peripheral Flag (44) - PFLG - The peripheral uses this line to acknowledge the transfer. The logic sense of this line is user configurable.

Special Purpose

Peripheral Status (45) - PSTS - This line can be used to indicate the OK or not OK status of the

peripheral. The logic sense of this line is user configurable.

Peripheral Reset (21) - PRESET - This is the reset line. It pulses low for at least 15 microseconds whenever a reset takes place.

External Interrupt Request (46) - EIR - This line allows the peripheral to interrupt the computer. The EIR line is active low.

General Purpose

Control 0 and 1 (22,23) - CTL0, CTL1 - These are general-purpose lines that the computer can set.

Status 0 and 1 (47,48) - STI0, STI1 - These are general-purpose lines that the computer can read.

Grounds

Grounds (18,24,26,43,49) - GND - These are the signal grounds. These pins provide the ground potential for the signal lines.

Chassis Ground (25) - CHGND - This pin is connected to the IO bracket that screws into the PC chassis. It is usually used to connect to the cable shield.

Differences Between GPIO 600 and GPIO 650

The GPIO 600 and 650 are both designed to provide the same interface for communication with digital peripherals. All interaction with the peripheral are essentially the same. The differences come from the interaction between the GPIO board and the host computer. There are also some configuration differences.

The GPIO 650 has a 16-bit interface with the host computer and can use DMA to transfer data. The GPIO 600 uses an 8-bit interface and does not support DMA. The wider data path and the available DMA can allow the GPIO 650 to transfer data significantly faster with fast peripherals than the GPIO 600.

Since the GPIO 650 has a 16-bit interface it has access to the high numbered interrupt sources. These higher sources, greater than 9, are more available than the lower sources making it easier to locate a free interrupt channel.

The input clock source is set on both boards with a dip switch but on the GPIO 650 the software can override these settings.

The PCTL delay time is modified on the GPIO 600 by soldering in a capacitor or a resistor. The PCTL delay time for the GPIO 650 is selected from software. This limits the maximum delay on the GPIO 650 to 1.5μs while the GPIO 600 can be extended to 100's of microseconds with the use of a large capacitor.

The DOUT clear jumper on the GPIO 600 and switch on the GPIO 650 clears the DOUT lines on reset. With the GPIO 600 this forces the output lines low regardless of the polarity set for the output lines. The GPIO 650 sets the lines to a logical 0. This allows the lines to be set low or left high impedance depending on the polarity set for the lines.

Interfacing Circuits

This section gives the electrical requirements for the GPIO interface and some suggestions on interfacing circuits.

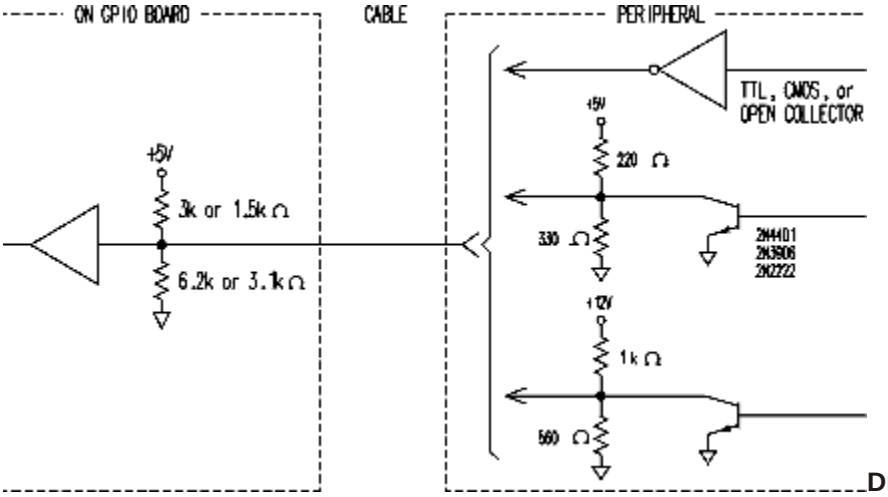
Peripheral Driver Circuit

All signal and data input lines are connected to a TTL input. The data lines are pulled up to 3.4V with a 3k/6.2kΩ resistor divider. The signal lines are pulled up to 3.4V with a 1.5k/3.1kΩ resistor divider. Input

voltages above 5.5V or below -0.5V can result in damage to the GPIO board.

Peripheral Drive Requirements:

Iin low	= 2mA — data lines
	= 4.5mA — signal lines (PFLG, PSTS, STI0, STI1)
Vin max	= 5.5V
Vin high	> 2.0V
Vin low	< 0.7V — data lines
	< 0.6V — signal lines



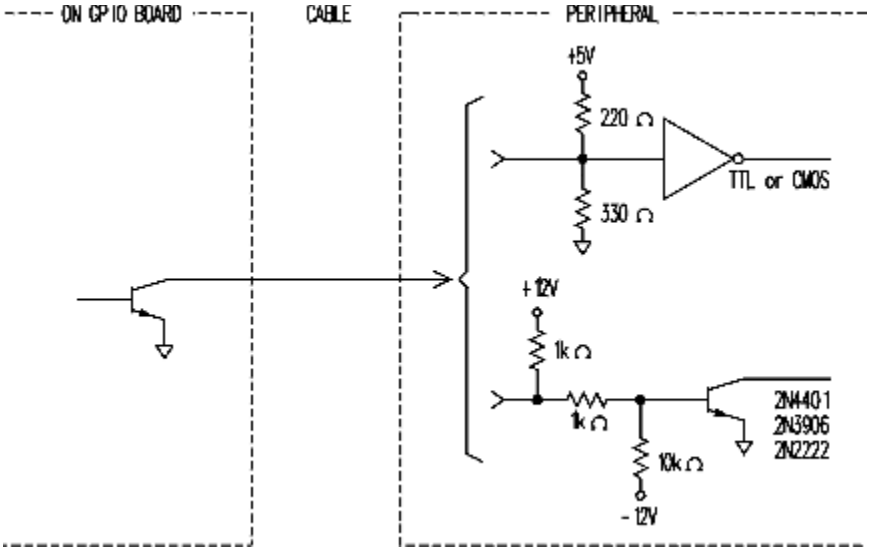
Driver Circuit Suggestions

Peripheral Receiving Circuit

All of the GPIO outputs, both for signals and data, use open collector output buffers. This means that the line is either left free for a high signal, high impedance, or being pulled to ground for a low signal. This has two effects: first, the computer can write to devices running off any voltage up to 30V, second, the driver must be pulled to the high voltage or the peripheral will probably always read a low.

GPIO16 Output Driver Specifications:

Vout low	= 0.4V max while sinking 16mA
	= 0.7V max while sinking 40mA
Vout high	= 30V max
Iout low	= 40mA



Receiver Circuit Suggestions

Data Settling Times

The data settling time is the time from when the peripheral or the computer writes the data until that data is valid. The length of the cable and the type of driver affects the settling time for a particular application. If sufficient time is not provided for the data to settle erratic data errors will occur.

PCTL Delay Time

The PCTL delay time sets the time from when the data is written to the output buffers until the PCTL line is set. The transition of PCTL from clear to set, indicates to the peripheral that the data is valid.

On the GPIO 650 the PCTL delay time is set with software from 0-100ns to 1.4-1.5μs. Refer to Chapters 3 or 5 for information on setting the PCTL Delay time.

With the GPIO 600 a resistor is added to shorten the delay time. A capacitor is added to lengthen the delay time. The resistor or capacitor must be soldered onto the board. The locations for R4 and C2 are shown in the board diagram in chapter 1.

GPIO 600 Equations for PCTL Delay Time

To decrease time constant:

$$R_2 = \frac{3.4e^{-3}}{t_{decrease}} - 4700$$

To increase time constant:

$$C_4 = \frac{t_{increase}}{3290}$$

Peripheral Delay Time

The peripheral must also insure that the data is has written is stable at the GPIO board before it latches the data into the input buffers. The peripheral should delay its response with PFLG until the data has settled.

Data Handshaking

Data handshaking is the process of using signals between the computer and peripheral to inform the other when it is ready to receive or transmit data. This section describes the use of the handshaking lines and demonstrates the data flow that these lines allow.

In the following discussion Full and Pulse Mode Handshaking are shown. The selection between Full and Pulse mode is made with the HND switch described in Chapter 1.

For inputs the computer reads the contents of the input data buffers on the GPIO board and not the actual input lines. The clock type determines when the data is loaded into these buffers. A RDY clock loads the buffers on a busy-to-ready transition of PFLG. A BSY clock loads the buffers on a ready-to-busy transition of PFLG. The RD clock loads the buffers as it is reading them. The setting of the switches that control the clock selection is described in Chapter 1. Software can override the input clock source switch settings on the GPIO 650 see Chapters 3 and 5 for the registers to modify.

A BSY or RDY clock would normally be chosen for transfers that use handshaking. These sources allow the user to know exactly when the data is going to be stored in the input buffers.

If the buffers are read without handshaking the RD clock source would normally be used. With the RD clock source the data read would be whatever is on the input lines when the computer reads the input buffers. If the clock source is BSY or RDY and PFLG is not toggled the buffer reads would always return the same value regardless of the input lines because the buffers would never be loaded with new data.

Handshaking Steps

A complete handshake involves five separate operations that take place in the order listed. The first two are optional depending on the configuration of the board.

1. A peripheral OK check can be made before the start of a transfer. The peripheral check consists of the computer reading the PSTS line, if it is a logical 1 the peripheral is considered OK.

2. A peripheral ready check can be made before proceeding with the transfer of each word of data. In this case the state of the PFLG line is checked. If PFLG is in the ready state the computer proceeds, if not, the computer waits until the peripheral puts PFLG in the ready state. The PFLG line is checked if Full-Mode handshakes are selected, it is not for Pulse-Mode handshakes.

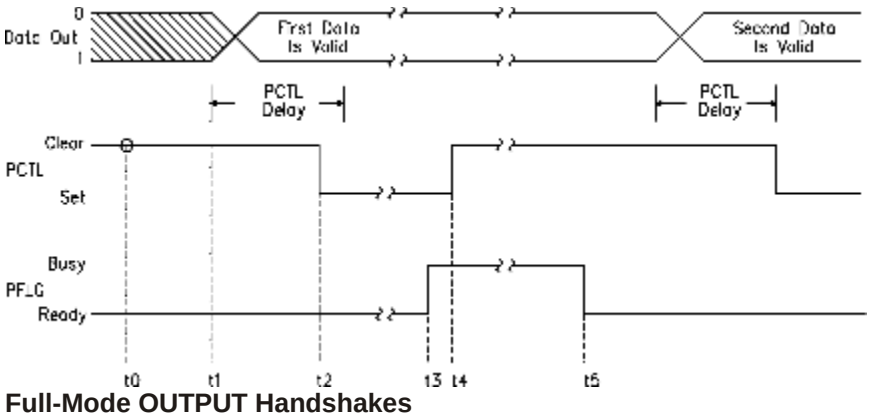
3. The computer initiates the transfer by setting the PCTL and IO lines. The IO line indicates the direction of transfer while PCTL indicates the start of a handshake.

4. The peripheral reads the data output lines or writes information to the data input lines.

5. The peripheral acknowledges that it has read or written the data. It does this by setting or clearing the PFLG line.

Timing Diagrams

In the following pages the possible configurations for the Full and Pulse Mode input and output handshakes are illustrated and described. Since the IO line is always set high for a read and low for a write it will not be shown. The peripheral OK check will not be illustrated.

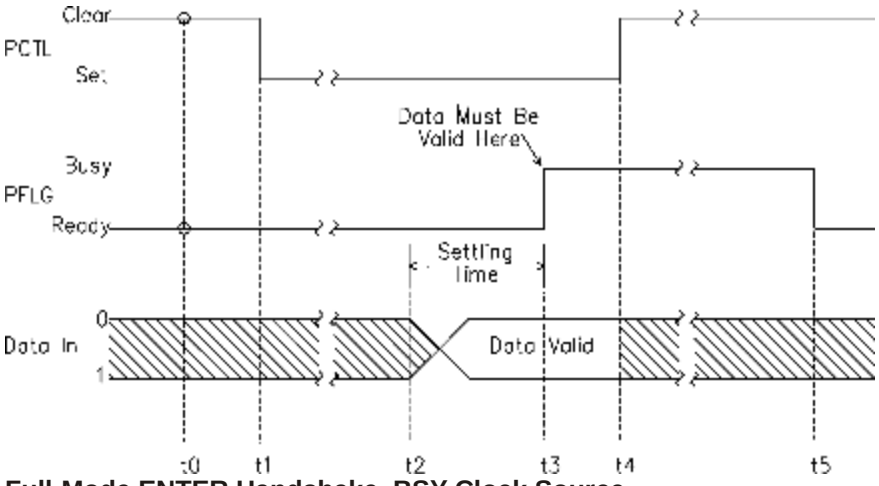


Full-Mode OUTPUT Handshakes

t0 - The computer waits until PCTL is clear and PFLG is ready before starting the transfer.
t1 - The computer puts the data on the output lines and sets the IO line low.
t2 - After the PCTL delay time has expired, PCTL is set. This indicates to the peripheral that there is valid data.
t3 - The peripheral sets PFLG to busy.
t4 - The ready to busy transition on PFLG clears the PCTL line.

t5 - The peripheral sets PFLG to ready.

Because the interface is set to full-mode the peripheral can read the data at any time from the falling edge of PCTL until the busy-to-ready transition of PFLG. The data stays valid since the interface will not change the output data until the interface returns to the ready condition of PCTL clear and PFLG ready.



Full-Mode ENTER Handshake, BSY Clock Source

t0 - The computer checks to see if the peripheral is ready. Since PCTL is clear and PFLG is ready the transfer can proceed.

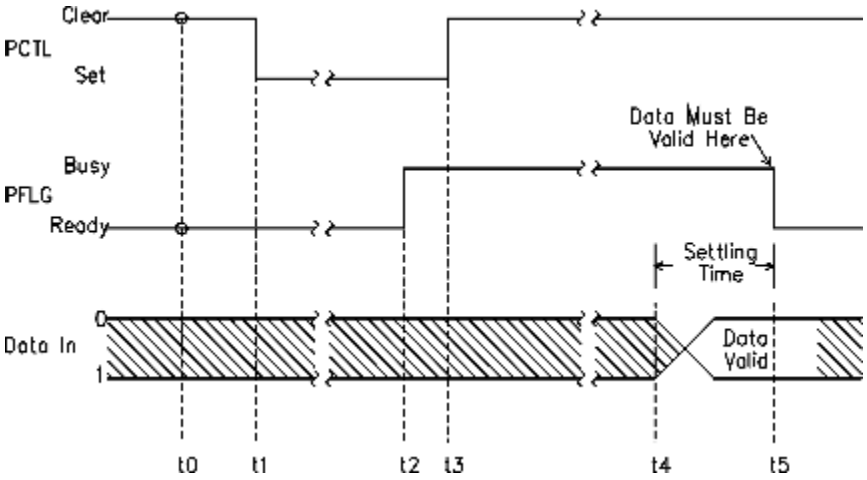
t1 - The computer sets PCTL to start the transfer. The IO line is simultaneously set high.

t2 - The peripheral places the data on the input lines.

t3 - After the data lines have had time to settle the peripheral sets PFLG. The setting of PFLG clocks the data into the input buffers on the GPIO board.

t4 - The PCTL line is cleared automatically after PFLG goes busy.

t5 - The peripheral sets PFLG ready. Once PFLG is ready the peripheral is ready to start a new transfer.



Full-Mode ENTER Handshake, RDY Clock Source

t0 - The computer checks to see if the peripheral is ready. Since PCTL is clear and PFLG is ready the transfer can proceed.

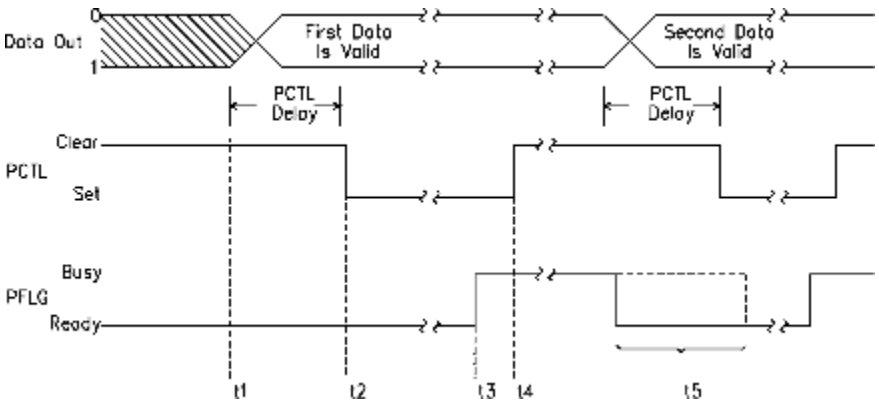
t1 - The computer sets PCTL to start the transfer. The IO line is simultaneously set high if it was not already high.

t2 - The peripheral sets PFLG busy.

t3 - PCTL is cleared in response to PFLG being set to busy.

t4 - The peripheral puts the data on the input lines. This can be done simultaneously with PFLG being set to busy.

t5 - The peripheral sets PFLG ready after the data lines have had time to settle. The transaction of PFLG from busy to ready will clock the data into the buffers on the GPIO board. This also puts the interface in the ready state to start a new transfer.



Busy Pulses, Pulse-Mode OUTPUT Handshakes

With the interface set to pulse-mode transfers, the computer will initiate the transfer if PCTL is clear regardless of the state of PFLG.

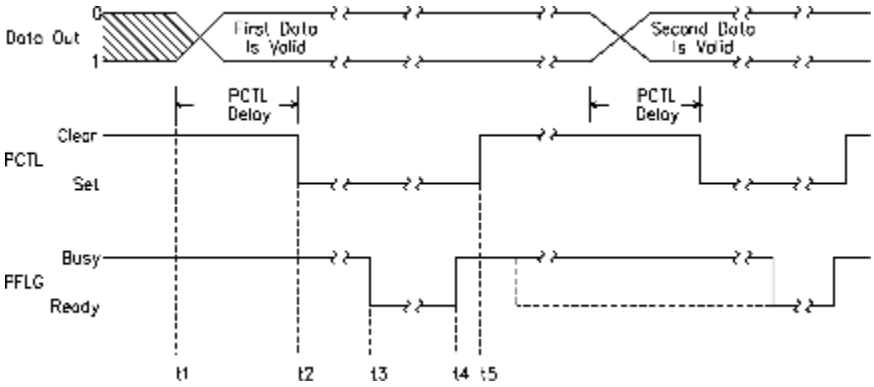
t1 - The computer puts the data on the data out lines and sets the IO line low.

t2 - After the PCTL delay time has expired, the PCTL line is set. This indicates that the data is valid.

t3 - The peripheral reads the data then acknowledges the transfer by setting PFLG to busy.

t4 - The PCTL line is cleared by the PFLG ready-to-busy transition. Once PCTL is clear the computer is ready to start a new transfer.

t5 - It is not important when PFLG goes ready since the computer will start a new transfer even if it is busy.



Ready Pulses, Pulse-Mode OUTPUT Handshakes

The computer will initiate the transfer if PCTL is clear regardless of the state of PFLG.

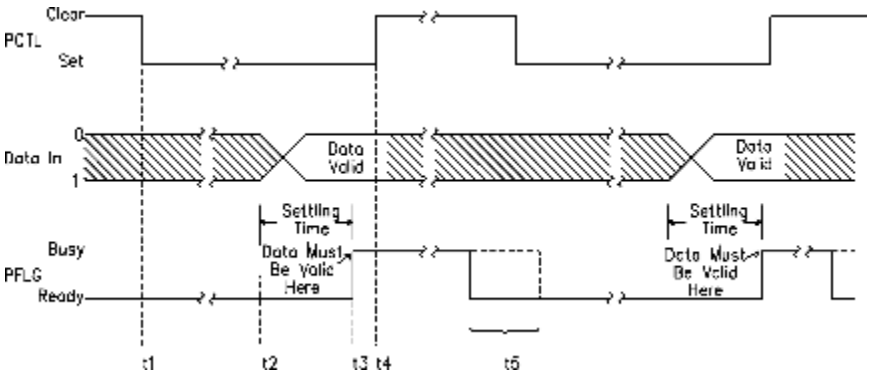
t1 - The computer puts the data on the output lines and sets the IO line low.

t2 - After the PCTL delay time has expired, the PCTL line is set. This indicates the presence of valid data on the output data lines.

t3 - The peripheral puts PFLG in the ready state.

t4 - The peripheral completes the handshake by setting PFLG busy. This ready-to-busy transition clears the PCTL line. The peripheral should read the data before the PFLG ready-to-busy transition.

t5 - Once PCTL is clear the computer is ready to start a new transfer.



Busy Pulses, Pulse-Mode ENTER (BSY Clock Source)

The computer will initiate the transfer if PCTL is clear regardless of the state of PFLG.

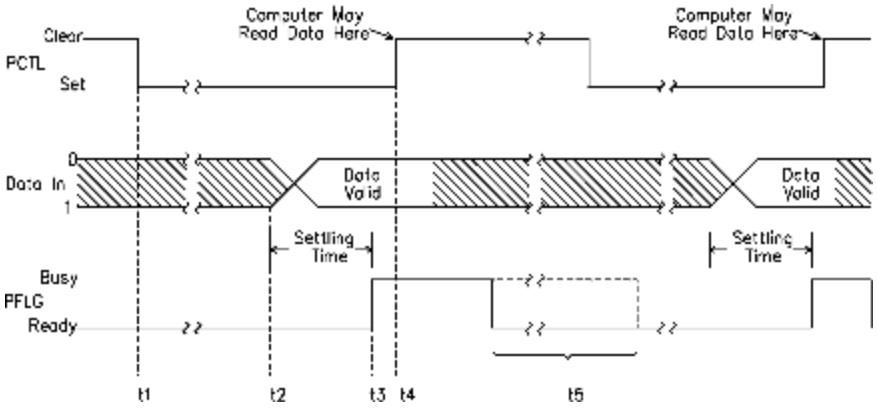
t1 - The computer sets PCTL and the IO line is set high.

t2 - The peripheral writes the data onto the data input lines.

t3 - After a data settling time the peripheral sets PFLG to busy. This clocks the data into the GPIO board buffers.

t4 - The PCTL line goes clear due to the ready-to-busy transition.

t5 - Some time later the PFLG line returns to ready. The next transfer start will not wait for this action.



Busy Pulses, Pulse-Mode ENTER (RDY Clock Source)

The computer will initiate the transfer if PCTL is clear regardless of the state of PFLG.

t1 - The computer sets PCTL and the IO line is set high.

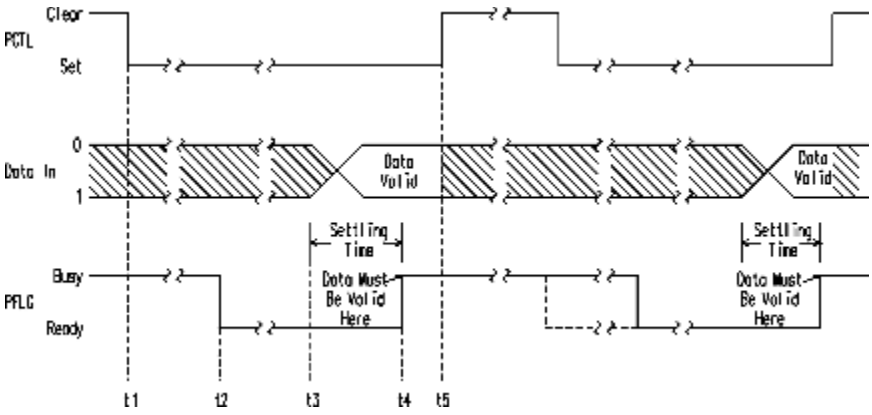
t2 - The peripheral writes the data onto the data input lines.

t3 - The peripheral sets PFLG to busy.

t4 - The PCTL line goes clear due to the PFLG ready-to-busy transition. At this point the computer considers the transaction complete and might read the buffers even though the data has not yet been clocked in.

t5 - The PFLG line returns to ready clocking the data into the GPIO board buffers.

Note: DO NOT USE THIS TRANSFER MODE. There is no way to guarantee that the peripheral can clock the data into the input data buffers before the computer reads the buffers. When the input data buffers are clocked late the data read will be from the previous handshake.



Ready Pulses, Pulse-Mode ENTER (BSY Clock Source)

The computer will initiate the transfer if PCTL is clear regardless of the state of PFLG.

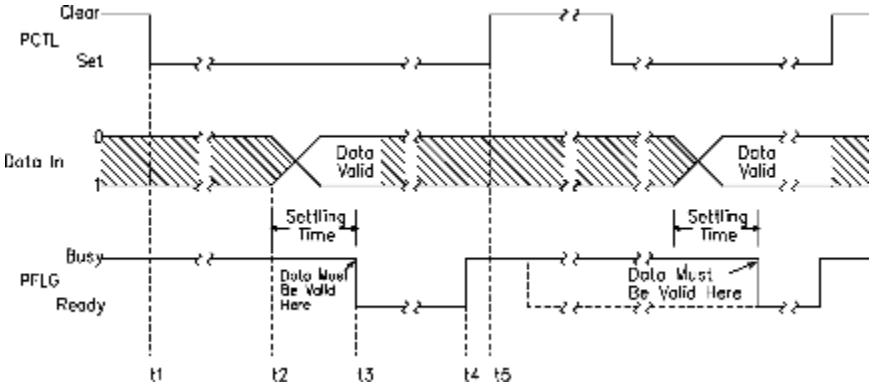
t1 - The computer sets PCTL and the IO line is set high.

t2 - The peripheral sets PFLG to ready.

t3 - The peripheral writes its data to the input lines.

t4 - After the data has had time to settle the peripheral sets the PFLG line to busy. This transaction will clock the data into the input buffers on the GPIO board.

t5 - In response to the ready-to-busy transition on the PFLG line, PCTL is cleared and the interface is ready to initiate another transfer.



Ready Pulses, Pulse-Mode ENTER (RDY Clock Source)

The computer will initiate the transfer if PCTL is clear regardless of the state of PFLG.

t1 - The computer sets PCTL and the IO line is set high.

t2 - The peripheral writes the data to the input lines.

t3 - After the data has had time to settle the peripheral sets the PFLG line to ready. This transaction will clock the data into the input buffers on the GPIO board.

t4 - The peripheral will later set PFLG to busy.

t5 - In response to the ready-to-busy transition on the PFLG line, PCTL is cleared and the interface is

ready to initiate another transfer.

General Purpose IO Lines

These lines are not used for handshaking and are available as aids in working with the peripheral. The usual purpose of each line is described.

PSTS Line

This is the peripheral status line. It can be used to check that the peripheral is still active or at least that the cable is still connected. The typical use is to have the peripheral pull the PSTS line to ground. The PSTS polarity is set so that low represents peripheral OK. If the cable becomes disconnected the line floats high due to the on-board pull-up resistors which causes the PSTS line to report that the peripheral is not OK.

PRESET Line

This is the peripheral reset line. When the board is reset this line pulses low for at least 15μs. This allows both the GPIO 650 and the peripheral to be reset at the same time. A reset is generally performed every time a new program that uses the board is run.

EIR line

This is the external interrupt request line. The board can be configured in software to cause an interrupt whenever this line is low. This lets the peripheral tell the computer when it needs attention. If not used as an interrupt source the EIR line can be used as a general-purpose input.

CTL0 and CLT1 Lines

These are the control 0 and control 1 lines. They have no predefined purpose but can be set and cleared from software as desired.

STI0 and STI1 Lines

These are the status 0 and status 1 lines. They have no predefined purpose but can be read from software as desired.

Interrupts

Interrupts are used to interrupt the computations that the computer is performing so that some other action can take place. The typical action with the GPIO 650 is the start or continuation of the transfer of data. Each interrupt source is enabled and disabled from software.

Interrupt Sources

- Interface Ready
- External Interrupt Request (EIR)
- DMA Terminal Count (GPIO 650 only)

Interface Ready

This interrupt source becomes active whenever the interface is ready to start a new handshake. The interface is ready if PCTL is clear with pulse mode handshake or when PCTL is clear and PFLG is ready with full mode handshake.

Software drivers may use this interrupt when controlling transfers. To avoid conflicting with the software drivers do not enable this source while the driver is performing a transfer.

External Interrupt Request

The EIR line is the source for this interrupt. When the EIR line is low and External Interrupt Request is enabled an interrupt is be generated. This is generally used by the peripheral to inform the computer that it needs attention.

This interrupt source is not used by the software drivers and may be active at any desired time without conflicting with the driver.

DMA Terminal Count

This interrupt source is used by the software driver to control DMA transfers on the GPIO 650. Since it is used exclusively with DMA, TransEra supplied software drivers do not make it available to the user. Information on this interrupt is in Chapter 5 Programming Guide for GPIO 650.

DMA Transfers

DMA stands for Direct Memory Access. DMA is not supported on the GPIO 600. These transfers use a DMA controller on the computer to move the data between the GPIO board and the computers memory without using the computers CPU. DMA transfers are not as subject to the variable delays that exist with Interrupt or Polled (CPU controlled) transfers.

Using DMA is generally the fastest way to perform large transfers of data. Small transfers may be faster with polled transfers because of the time it takes to configure the DMA controller. TransEra supplied software drivers selects the transfer mode that is expected to be the fastest.

When performing 8-bit transfers with the peripheral the GPIO 650 performs two handshakes and then transfers the data with a single 16-bit DMA transfer. This doubles the maximum transfer rate with 8-bit peripherals.

The memory structure of a PC and the limitations of the DMA controller, limits the amount of data that can be transferred with a single DMA controller configuration from 1 to 65536 words. Where in the range a specific transfer starts depends on the physical location in memory that the data is using. When the DMA controller reaches that configuration's limit it needs to be reconfigured. While the DMA controller is being reconfigured transfers with the peripheral are paused. The user will generally not be able to predict when these pauses will take place.

With the exception of the pause mentioned above, the maximum time that the computer should take to get ready for the next handshake is 4μs. With polled and interrupt modes the time for the computer to get ready can vary greatly depending on the speed of the machine and the resources that are in use.

Reset

Before using the GPIO board it is recommended that the board be reset. This puts the board and interface into a known state before accessing the interface. When the board is reset a reset pulse of at least 15μs is put on the PRESET line to inform the peripheral that a reset has taken place.

Chapter 3 HTBasic Software Interface

This chapter explains how to use the GPIO board with HTBasic and the HTBasic software driver. With the introduction of HTBasic for Windows 9.0 Both the GPIO 600 and 650 cards are fully supported under all operating systems supported by the 9.0 release.

The interface is designed to duplicate the operation of the HP GPIO 98622A as close as possible so that few if any software changes need to be made when using either the GPIO 600 or GPIO 650 boards with HTBasic in place of the HP GPIO 98622A with RMB. The differences between the two systems are summarized in the section Differences Between TransEra and HP GPIO.

Differences Between TransEra and HP GPIO

This section gives the differences between the HP GPIO 98622A and the TransEra GPIO Boards.

Setup

TransEra 600 and 650: Board address must be set with dip switches for the PC.
HP: No address needs to be selected.

TransEra 600 and 650: ISC is set in software during the LOAD BIN command.
HP: ISC is selected with a dip switch.

TransEra 600 and 650: The interrupt is selected from software in the LOAD BIN command.
HP: An interrupt priority is selected with a dip switch.

TransEra 600: Identical to HP.
TransEra 650: The PCTL delay time is set from software.
HP: A resistor or capacitor is added to change the PCTL delay time.

TransEra 600: Does not support DMA.
TransEra 650: The DMA channel is selected from software in the LOAD BIN command.
HP: The DMA channel is selected with a DIP switch.

Operation

TransEra: Unused bits in Status and Readio registers are set to zero.
HP: Unused bits in Status registers are set to zero while unused bits in Readio registers are generally set to one.

TransEra 600: Used a DOUT CLEAR jumper but the output lines are set low regardless of the selected polarity.
TransEra 650: When in the on position the DCLR switch causes the data output lines to be set to a logical zero.
HP: Uses the DOUT CLEAR jumper to perform the same function as the GPIO 650.

Registers

TransEra 600: Does not use bits 0-5 of the Interrupt and DMA Status register.
TransEra 650: Does not use bits 1-5 of the Interrupt and DMA Status register and bit 0 reports if DMA is active.
HP: The Interrupt and DMA Status register uses these bits differently.

TransEra 600: Same as HP.
TransEra 650: Defines an Input Clock Source register to override the switch selected input clock sources.
HP: Does not have this register.

Control Registers

This section lists the control registers and explains the purpose of the control bits. These registers should be used instead of writeio registers for board control. All registers are 8-bit unless otherwise noted.

The value associated with each bit is the decimal representation of that bit being set.

Control Registers

Register	Name
0	Interface Reset
1	Set PCTL
2	Peripheral Control
3	Data Out (16-bits)
4	Input Clock Source (GPIO 650 only)
-	Interrupt Enable

Reset Interface

CONTROL Register 0

Writing a non-zero value to this register resets the interface.

Set PCTL

CONTROL Register 1

Writing a non-zero value to this register sets PCTL.

Peripheral Control

CONTROL Register 2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Value=128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1
		0			Report PSTS Error	Set CTL1 Low	Set CTL0 Low

Bit 0: Set CTL0 Low
When this bit is set the CTL0 line is driven low. When clear, the CTL0 output is in a high impedance state.

Bit 1: Set CTL1 Low
When this bit is set the CTL1 line is driven low. When clear the CTL1 output is in a high impedance state.

Bit 2: Report PSTS Error
When set the interface will check the PSTS line before that start of each transfer. When clear the status of the PSTS line is ignored.

Data Out (16 bits)

CONTROL Register 3

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

This is the output data buffer. Each bit is output on the corresponding output line. The logic polarity is user selectable with switch 1.

Interface Parameters

Control Register 4 (not present on GPIO 600 or HP 98622A)

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
		0		Clock Source High Byte		Clock Source Low Byte	

Bits 0,1: Clock Source Low Byte
These bits select the input clock source for the low byte data buffer. These bits are cleared on LOAD BIN or reset. When clear, the switch 2 setting are used.

Bits 1-0	**Clock Source**
00	Switch Setting
01	RDY
10	BSY
11	RD

Bits 2,3: Clock Source High Byte
These bits select the input clock source for the high byte data buffer. These bits are cleared on LOAD BIN or reset. When clear, the switch 2 setting are used.

Bits 3-2	**Clock Source**
00	Switch Setting
01	RDY
10	BSY
11	RD

Interrupt Enable

Interrupt Enable Register

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
			0			Enable Interface Ready Interrupts	Enable EIR Interrupts

Bit 0: Enable Interface Ready Interrupts

When set, an interrupt is requested when the interface is ready. When clear, this interrupt source is disabled.

Bit 1: Enable EIR Interrupts

When set, an interrupt is requested when the EIR line is low. When clear, this interrupt source is disabled.

Status Registers

This section lists the status registers and explains what each bit returns. These registers should be used instead of readio registers for board control. All registers are 8-bit unless otherwise noted.

Status Registers

Register	**Name**

0	Card Identification
1	Interrupt and DMA Status
2	Board Status
3	Data In (16-bit)
4	Interface Ready
5	Peripheral Status

Card Identification

STATUS Register 0

Reading this register returns the ID number of the GPIO driver. The ID number for the GPIO driver is 3.

Interrupt and DMA Status

STATUS Register 1

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
Interrupts Are Enabled	Interrupt is Requested			Undefined			DMA is Active

Bit 0: DMA is Active
A DMA transfer is being performed. If clear DMA is not active. This bit is undefined on the GPIO 600.

Bit 6: Interrupt is Requested
If set an interrupt is currently requested. If clear no interrupts are requested.

Bit 7: Interrupts are Enabled
If set at least one interrupt source is enabled. If clear no interrupt sources are enabled.

Transfer Status

STATUS Register 2

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
		Undefined			Handshake In Progress	Interrupts Are Enabled	Transfer In Progress

Bit 0: Transfer in Progress
If set a transfer is taking place. If clear a transfer is not active.

Bit 1: Interrupts are Enabled
This is the same as bit 7 of the Interrupt and DMA Status register. If set at least one interrupt source is enabled. If clear no interrupt sources are enabled.

Bit 2: Handshake in Process
If set a handshake is in progress. With pulse mode handshakes that means that PCTL is set. With full mode handshakes PTCL is set or PFLG is busy. If clear a handshake is not taking place.

Data In (16 bits)

STATUS Register 3

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

This register returns the value stored in the input buffers. The input buffers are updated according to the input clock source selected for each byte.

Interface Ready

STATUS Register 4

If set the interface is ready for a subsequent data transfer. If clear the interface is busy.

Peripheral Status

STATUS Register 5

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
Undefined				PSTS OK	EIR Line Low	STI1 Line Low	STI0 Line Low

Bit 0: STI0 Line Low
If set the STI0 line is low. If clear the line is high.

Bit 1: STI1 Line Low
If set the STI1 line is low. If clear the line is high.

Bit 2: EIR Line Low
If set the EIR line is low. If clear the EIR line is high.

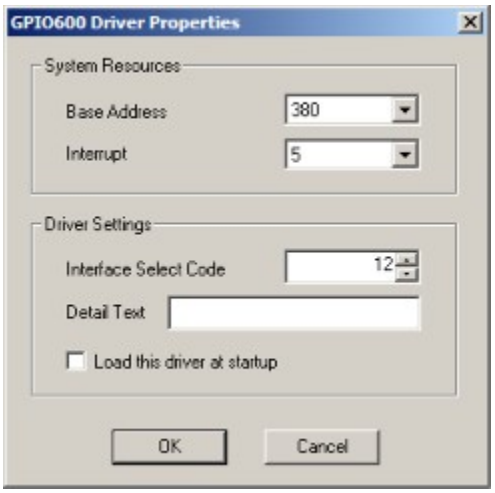
Bit 3: PSTS OK
If set the PSTS line is OK. The polarity of the PSTS line is user selectable with switch 1. If clear the PSTS line is not OK.

Configuring and Loading

The Device Setup dialog is used for configuration and loading both cards. Once a driver is configured it may be loaded using the Device Setup, or by including a line like the following in your AUTOST file:

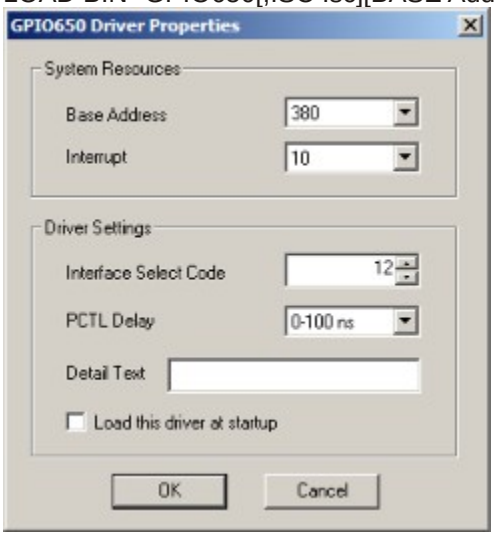
Model 600 Board

LOAD BIN "GPIO600[;ISC Isc][BASE Address][INTERRUPT Int]"



Model 650 Board

LOAD BIN "GPIO650[;ISC Isc][BASE Address][INTERRUPT Int] [DELAY Delay]"



Options

The legal options for the GPIO drivers are:

BASE AddressINTERRUPT InterruptISC IscDELAY Delay (650 only)

Once the driver is configured, it may be necessary to reboot the machine. Both the Device Setup Dialog and the LOAD BIN command load the software driver that HTBasic uses to communicate with the GPIO board. The LOAD BIN command must be executed before any access to the board from HTBasic.

For the DOS version, the driver filenames that the LOAD BIN uses for the GPIO 600 are ‘gpio.d86’ for the PC version and ‘gpio.d36’ for the 386 version. For the GPIO 650 the filenames are ‘gpio16.d86’ for the PC version and ‘gpio16.d36’ for the 386 version. Note: there is no Device Setup necessary in the DOS version.

If the default value is used that parameter does not need to be specified in the LOAD BIN command.

LOAD BIN Options

Parameter	Default Value	Range
BASE	380 (hex)	200-3F0 (hex)
ISC	12	4-32
INT (GPIO 600)	5	0,2,3,4,5,6,7
INT (GPIO 650)	10	0,5,7,9,10,11,12,15
DMA (GPIO 650)	6	0,5,6,7
DELAY (GPIO 650)	1	0-7

The address set with the IO Address switch must match the BASE address value used for LOAD BIN. If these do not agree the LOAD BIN will give a device not found error.

The default ISC for the GPIO board is 12. This is the same ISC number assigned to the second parallel port. If the computer has a second parallel port the ISC of the GPIO board should be changed.

Interrupts may be disabled by using a 0 as the interrupt value. If interrupts are disabled the board will not support TRANSFER commands and ENTER and OUTPUT commands could be slower.

DMA transfers are only supported on the GPIO 650 under DOS. An interrupt channel must be specified for DMA to be used. DMA is not supported on the GPIO 600.

The DELAY option controls the PCTL delay time for the GPIO 650. This delay is the time from when the computer outputs the data onto the GPIO cable until the PCTL line is set. If set to a time shorter than the data settling time there can be data errors on outputs. A resistor or capacitor is added to the board to change the PCTL delay time for the GPIO 600.

Value	Delay Time
0	0-100ns
1	200-300ns
2	400-500ns
3	600-700ns
4	800-9 00ns
5	1.0-1.1μs
6	1.2-1.3μs
7	1.4-1.5 μs

An example LOAD BIN for the GPIO 650 with a base address of 220 hex, ISC number of 13, using interrupt channel 15 and with a PCTL delay of 400-500ns.

```
LOAD BIN "GPIO650; BASE 220 ISC 13 INT 15 DELAY 2"
```

The LOAD BIN for the GPIO 600 would be the same except the DMA and DELAY parameters would be left off and the INT parameter must match the jumper location.

Interface Reset

The interface should be reset before use to put the interface into a known state. It is reset when the CLR I/O or BASIC RESET key is pressed and when the LOAD BIN command is executed. The interface can be also be reset in software by writing a one to Reset Interface (control register 0).

The following occurs during a reset:

- The Peripheral Reset line (PRESET) is pulsed low for at least 15μs.
- The PCTL line is cleared.
- The interrupt enable bit is cleared. This disables interrupts until re-enabled.
- If the DCLR switch is set, the Data Out lines are set to a logic low on the GPIO 650. If the DOUT CLEAR jumper is set on the GPIO 600 the Data Out lines are set low.
- For the GPIO 650, the clock source high and low bits of Interface Parameters (control register 4) are set to 0. This forces the clock source to the switch two settings.

The following lines are unchanged by a reset:

- The CTL0 and CTL1 output lines.
- The IO line.
- The Data Out lines (if DCLR or DOUT CLEAR JUMPER is not set).

Using STATUS and CONTROL Commands for I/O

Reading status register three returns the value of the data buffers in 16-bit 2's complement (-32768 to 32767). The value of these buffers depends on the clock source selected. With the RD source the buffers will be clocked just before reading so that the value read reflects the value on the input data lines. If RDY or BSY source is selected the value will be whatever was present during the previous handshake. The IO line is set high to indicate that a read is taking place. The status command does not use handshaking, therefore, it will not set PCTL to initiate a handshake.

STATUS Example

```
                                        DI15-DI8    DI7-DI0
10    Gpio=12
20    STATUS Gpio,3;A                   2           3
30    STATUS Gpio,3;B                   255         255
```

With the RD clock source A = 515 (i.e.,256×2 +3) and B = -1.

Writing to control register three sets the data lines to the indicated value. The value is represented in 16-bit 2's complement (-32768 to 32767). The IO line will be set low to indicate that an output was performed. No handshaking takes place since PCTL will not be set.

CONTROL Example

```
                                        DO15-DO8    DO7-DO0
10    Gpio=12
20    CONTROL Gpio,3;5*256+11           5           11
30    CONTROL Gpio,3;-2                 255         254
```

Status and Control registers can be used to control user defined handshaking. The following example demonstrates the use of the STATUS and CONTROL statements to perform handshaking with the CTL0, STI0, and IO lines. STI0 will not be able to clock the data into the input buffers since only PFLG is connected to the clocks but it can be used to indicate when there is valid data on the lines to be read. The RD clock source is required so that a status read will clock the data line values into the buffers before reading them.

User Handshake Example

```
10    Gpio=12
20    Num_samples=10
30    DIM Samples(10)
40    STATUS Gpio,3;Tmp     ! Perform a dummy read to set IO high
50    FOR I=1 TO 10         ! This reads 10 data points
60      CONTROL Gpio,2;1    ! Set CTL0 to request communication
70  Loop1: Status GPIO,5;Stat5    ! Wait until peripheral sets STI0
80      IF NOT BIT(Stat5,0) THEN GOTO Loop1
90      STATUS Gpio,3;Samples(I)  ! Read data point
100      CONTROL Gpio,2;0   ! Clear CTL0 to indicate data has been read
110 Loop2: STATUS Gpio,5;Stat5    ! Wait for STI0 to be cleared
120      IF BIT(Stat5,0) THEN GOTO Loop2    ! For a full handshake
130    NEXT I
```

Using OUTPUT and ENTER

Whenever the OUTPUT or ENTER command is used a complete handshake is required. If the handshake does not occur HTBasic gives one of following two responses: 1) If a timeout is set the interface will wait the allotted time then give a timeout error. 2) If the timeout period is not set the computer will appear to have locked up. Pressing the CLR I/O key restores control to the user. The default for the GPIO board is to not have the timeout set.

Most of the examples are of OUTPUT statements. ENTER statements require the same data on the same data lines that the OUTPUT statements produce.

The GPIO interface defaults to an 8-bit transfer unless otherwise specified.

The ASCII representations for A, B, and C are 65, 66, and 67 respectively. Carriage Return is 13 and Line Feed is 10. All values shown are in decimal.

OUTPUT String Example

```
                                        DO15-DO8    DO7-DO0
10    Gpio=12
20    OUTPUT Gpio;"AB"                  0           65
                                        0           66
                                        0           13
                                        0           10
```

Only the low 8 bits are used for default transfers, Carriage Return Line Feed is the default delimiter.

OUTPUT Multiple Strings Example

```
                                        DO15-DO8    DO7-DO0
10    Gpio=12
20    OUTPUT Gpio;"AB";"C"              0           65
                                        0           66
                                        0           67
                                        0           13
                                        0           10
```

Multiple strings are combined before outputting.

OUTPUT Word Default Delimiter

```
                                        DO15-DO8    DO7-DO0
10    Gpio=12
20    OUTPUT Gpio USING "W";5*256+15     5           15
                                        0           13
                                        0           10
```

The default delimiter (byte mode) is still in effect.

OUTPUT Word No Delimiter

```
                                        DO15-DO8    DO7-DO0
10    Gpio=12
20    OUTPUT Gpio USING "W,#";5*256+15   5           15
30    OUTPUT Gpio USING "W,#";-1         255         255
```

Numbers are output in 2's complement 16-bit integers (-32768 to 32767).

OUTPUT Strings With Word Example

```
                                        DO15-DO8    DO7-DO0
10    ASSIGN @Gpio TO 12;WORD,FORMAT ON
20    OUTPUT @Gpio;"A";"BC"             65          66
                                        67          13
                                        10          0
```

This transfer method would normally not be used due to the difficulty of having the peripheral reorganize

the data.

OUTPUT Word Format Off Example

```
                                        DO15-DO8    DO7-DO0
10    ASSIGN @Gpio TO 12;WORD,FORMAT OFF
20    OUTPUT @Gpio;5*256+11             5           11
30    OUTPUT @Gpio;-2                   255         254
```

WORD,FORMAT OFF is equivalent to "W,#". This will execute the transfer of data 5-10 times faster than if the USING "W,#" were used.

OUTPUT Array Example

```
                                        DO15-DO8    DO7-DO0
10    INTEGER I(10)
20    MAT I=(3*256+7)
30    ASSIGN @Gpio TO 12;WORD,FORMAT OFF
40    OUTPUT @Gpio;I(*)                 3           7
                                        3           7
                                        ETC.
```

Arrays can be output with a single statement. By default HTBasic arrays are 0 based. This causes the above example to output 11 I(0)-I(10) words of data.

ENTER statements expect the same data, on the same numbered data input lines that a corresponding output statement produces.

ENTER Array Example

```
                                        DI15-DI8    DI7-DI0
10    INTEGER I(10)
20    ASSIGN @Gpio TO 12;WORD,FORMAT OFF
30    ENTER @Gpio;I(*)                  3           7
                                        3           7
                                        ETC.
```

Eleven handshakes complete the above transfer since INTEGER I(10) creates an array with eleven elements After the transfer, all entries of array "i" will contain 775 (i.e.,3X256 + 7).

Using the TRANSFER Command

The TRANSFER command allows data to be exchanged (similar to INPUT and OUTPUT commands) with the GPIO through I/O paths. The main difference between TRANSFER and INPUT/OUTPUT statements is TRANSFER can allow other HTBasic commands to be executed as it is executing. A TRANSFER can be thought of as a "background" process. A complete handshake is required when using TRANSFER. A timeout will only occur if the WAIT parameter is specified in the TRANSFER statement.

The following program reads 8 bytes using the TRANSFER statement at line 90. Every 4 bytes read will cause the subroutine at line 170 to be executed. After 2 records are read the transfer will be terminated and the subroutine at line 190 will be executed. Notice that HTBasic continues to execute after the TRANSFER statement is started.

Input TRANSFER Example

```
10    Gpio=12
20    INTEGER Xx,I,Cnt,Done
30    DIM X$[8] BUFFER     ! Create the buffer
```

```
40    ASSIGN @Buf TO BUFFER X$;FORMAT OFF
50    ASSIGN @Dev TO Gpio
60    Done=0
70    ON EOR @Dev GOSUB 170 ! Set up end of record gosub
80    ON EOT @Dev GOSUB 190 ! Set up end of transfer gosub
85    ! Transfer 2 four byte records
90    TRANSFER @Dev TO @Buf;RECORDS 2,EOR (COUNT 4)
100    IF Done=0 THEN 100
110    STATUS @Buf,4;Cnt     ! Read number of bytes in buffer
120    FOR I=1 TO Cnt
130      ENTER @Buf USING "#,B";Xx  ! Read data from the buffer
140      PRINT USING "#,K,B,K,/";"[",Xx,"]"
150    NEXT I
160    PAUSE
170    PRINT "END OF RECORD"
180    RETURN
190    PRINT "END OF TRANSFER"
200    Done=1
210    RETURN
220    END
```

The following program outputs 8 bytes (4 words) to the GPIO. The ON EOR statement at line 90 will cause the subroutine at line 160 to be executed every 4 bytes transferred. The ON EOT statement at line 100 causes the subroutine at line 180 when the transfer is done. The data will be output one word (16 bits) at a time (4 handshake transfers). The first transfer will output 49 ("1") on D0-D7 and 50 ("2") on D8-D15.

Output TRANSFER Example

```
10    Gpio=12
20    INTEGER Bytes,I,Done
30    DIM X$[8] BUFFER       ! Create the buffer
40    Bytes=8
50    ASSIGN @Buf TO BUFFER X$;FORMAT OFF
60    ASSIGN @Dev TO Gpio;WORD
65    ! Load the buffer with a string
70    OUTPUT @Buf USING "#,K";"12345678"
80    Done=0
90    ON EOR @Dev GOSUB 160  ! Set up the end of record gosub
100    ON EOT @Dev GOSUB 180  ! Set up the end of transfer gosub
110    PRINT "START TRANSFER"
115    ! Transfer 2 four byte records
120    TRANSFER @Buf TO @Dev;EOR (COUNT 4)
130    WHILE Done=0
140    END WHILE
150    PAUSE
160    PRINT "END OF RECORD"
170    RETURN
180    PRINT "END OF TRANSFER"
190    Done=1
200    RETURN
210    END
```

Maximizing the Speed of I/O Transfers

For many users the transfer rate of the GPIO board is not relevant since they are passing relatively small amounts of data and the board is fast enough. In other cases transfer rate can be critical. This section

lists some ideas to help increase the transfer rate. These techniques are not always applicable but when used can increase performance 2-10 times verses other methods that perform the same task.

Assign a DMA channel to the GPIO interface if using the GPIO 650. When the board is allowed to use DMA the transfer will be able to move data at a rate of 250K word/second. Although pauses will occur in the transfer every 64K words to reinitialize the DMA controller it is still the fastest way to transfer with most machines. Note that timeouts are disabled during DMA transfers.

Avoid the use of USING statements with OUTPUT and ENTER commands. When USING is defined HTBasic handles the transfer in a way that allows it to reformat the data if necessary. This occurs even if it does not reformat the data. This handling adds a large amount of overhead to the transfer. For an example, assigning FORMAT OFF, WORD to the GPIO device will execute outputs about 10 times faster than if USING "W,#" were specified in the OUTPUT statement. In both cases the output data will be the same.

The use of delimiting for ENTER and TRANSFER strings is usually required but the overhead of checking every byte to see if it is the delimiting character slows the transfer significantly. When the length of the data is known use COUNT with TRANSFERs. With ENTERs, ALLOCATE the array to the size of the data and then ENTER into that array.

HTBasic is very flexible when it comes to changing the type of data that is being used. This ability to change the data type can also result in significant overhead. When possible define arrays as integers and use the FORMAT OFF mode to transfer numeric data to the peripheral. If not specified the HTBasic default is FORMAT ON which converts the data into its ASCII representation before outputting it. This results in a large speed penalty.

These techniques are not limited to the exact situations described but the ideas of using DMA, not delimiting, using FORMAT OFF and avoiding data type changes can increase the data rate for many applications.

GPIO Timeouts

Timeouts are used to prevent the system from appearing to lock up when communication with the peripheral is not functioning.

The timeout period starts when PCTL is set and continues until the interface is ready. With a Pulse-Mode transfer it continues until PCTL is clear. For a Full-Mode transfer it continues until PCTL is clear and PFLG is ready.

When a timeout occurs the interface is reset. Refer to Interface Reset for a list of changes on reset. If the error is not trapped with an ON TIMEOUT instruction the program will terminate and display a timeout error.

When using DMA, timeouts are disabled. If the interface stops communicating during a transfer the computer can appear to lock up. Pressing the BASIC RESET key will terminate the program and return control to the user. If this is unacceptable DMA must be disabled by selecting DMA channel 0 when performing the LOAD BIN statement.

General Purpose Lines

There are four general-purpose lines that are available for any use. CTL1 and CTL0 are output lines and STI1 and STI0 are input lines. The lines are read and written to through status and control statements.

Peripheral Control

The CTLx lines are set by writing to Peripheral Control (control register 2). When writing to this register all the bits are modified. To avoid accidentally clearing other bits a software copy of the register should be kept. When a CTLx line it to be changed, bit modify the software copy and then write it to the control

register. Note that CTLx has low-true polarity (1 = low). This polarity is not selectable.

CTLx Control Example

```
10    Gpio=12                        ! 12 = Interface Select Code
20    Pcontrol=BINIOR(Pcontrol,1     ! Set CTL0
25    ! Clear a bit by ANDing the register with the compliment
30    Pcontrol=BINAND(Pcontrol,BINCMP(2))    ! Clear CTL1
40    CONTROL Gpio,2;Pcontrol        ! Write out value
```

This sets the CTL0 bit and clears the CTL1 bit in line 130. Since the outputs are inverted the CTL0 line is then forced low and the CTL1 line goes to high impedance.

The STIx lines are read by reading Peripheral Status (status register 5). The STIx lines have low-true polarity (1 = low). This polarity is not selectable.

Check STIx Status Example

```
10    Gpio=12                  ! 12 = Interface Select Code
20    STATUS Gpio,5;Stat5      ! Read Peripheral Status
30    Sti0=BIT(Stat5,0)        ! Check the STIx bits
40    Sti1=BIT(Stat5,1)
```

Using PSTS

The PSTS line is generally used to indicate if peripheral communication is operational. The interface can be enabled to check it before every OUTPUT, ENTER and TRANSFER command by setting the PSTS Error bit of the Peripheral Control Register. Its status can be read at any time, whether or not PSTS Checking is enabled, by reading the PSTS OK bit of Peripheral Status.

If the PSTS input is left floating (not connected to anything), it will read a high. Therefore it is recommended to set the polarity to low = OK. This way a disconnected cable will return a PSTS not OK signal.

Activate PSTS Checking Example

```
10    Gpio=12                         ! 12 = Interface Select Code
20    Pcontrol=BINIOR(Pcontrol,4)     ! Set PSTS Check
```

Deactivate PSTS Checking Example

```
10    Gpio=12
15    ! Clear a bit by ANDing the register with the compliment
20    Pcontrol=BINAND(Pcontrol,BINCMP(4))    ! Clear CTL1
30    CONTROL Gpio,2;Pcontrol                ! Write out value
```

Read PSTS Value Example

```
10    Gpio=12
20    STATUS Gpio,5;Pstatus             ! Read Peripheral Status
30    Psts=BIT(Pstatus,2)               ! Check Psts bit
```

HTBasic GPIO Interrupts

The GPIO interface supports Ready interrupts and External Interrupt Request (EIR) interrupts. Both of these interrupts are level-sensitive, this means that the signal must remain until it can be serviced. The service routine must be able to distinguish between the interrupts as they both call the same interrupt service routine.

Enabling Interrupts

After LOAD BIN or reset the interrupts are disabled. They are enabled by writing a new mask into the Interrupt Enable Register. The mask identifies which interrupts will be allowed to cause an interrupt. Set each desired interrupt bit to enable it.

Interrupt Mask Example

```
10    Gpio=12                 ! 12 = Interface Select Code
20    ON INTR Gpio GOSUB Int
30    ENABLE INTR Gpio;1      ! Set the mask to 1 for EIR
```

Interface Ready

When the interface ready bit is set an interrupt occurs whenever the interface becomes ready. In Full-Mode Handshake the interface is ready whenever PCTL is clear and PFLG is ready. In Pulse-Mode Handshake the interface is ready when PCTL is clear regardless of the state of PFLG.

External Interrupt Request

When the EIR bit is set an interrupt occurs whenever the EIR line goes low. The polarity of this line cannot be changed. This line must be kept low until it is inside the service routine or it may be missed by the program.

Interrupt Service Routine

The interrupt service routine is the code that the computer executes if an interrupt occurs. A typical application is using an EIR interrupt to signal the computer that it has data available to be read. Interrupts are disabled after every call to the interrupt service routine. If the application requires multiple interrupts, the interrupts must be re-enabled during the interrupt service routine.

If both EIR and Interface Ready interrupts are enabled the service routine needs to be able to identify what caused the interrupt. This is accomplished by reading Interface Ready (status register 4) to see if the interface is ready or the EIR Line Low bit of Peripheral Status (status register 5) to check for an EIR interrupt.

The following example sets up an interrupt handler called `Gpio_int` to read one character each time EIR goes low. Eleven integer values will be read and then stored in the `Samples` array. In this example the peripheral needs to clear EIR when the sample point is read.

Interrupt Controlled Input

```
 10    Gpio=12
20    Num_samples=10
30    Count=0
40    INTEGER Samples(10)
50    ON INTR Gpio GOSUB Gpio_int
60    ENABLE INTR Gpio;1         ! Enable the EIR interrupt
70    ! The loop below can be replace by useful code while waiting
80    ! for the transfer to be completed
90   Loop: DISP TIME$(TIMEDATE)
100    IF Count<Num_samples THEN GOTO Loop    ! Wait for transfer
110    DISP "Transfer is complete"
120    STOP
200    Gpio_int:        ! This is the interrupt handling routine
210    STATUS Gpio,5;Stat5
220    IF BIT(Stat5,2) THEN    ! Check if a valid gpio interrupt exists
240      ENTER Gpio USING "W,#";Samples(Count)
```

```
250       Count=Count+1
260       ENABLE INTR Gpio    ! Peripheral needs to clear EIR before
265                     ! this statement
270     END IF
280     RETURN
290     END
```

READIO and WRITEIO Registers

This section describes the GPIO Interface's READIO and WRITEIO registers. These registers are made to be as compatible as possible with the registers from the HP GPIO 98622A. They are not the physical registers that exist on either of the the GPIO boards. To use these registers the driver must be loaded. The actual physical registers are listed in the Programming Guide chapters of this manual.

In some cases both the HP and TransEra boards have similar bits but operation is different particularly with DMA operation. In these cases the control bits are left out of the READIO and WRITEIO registers.

READIO and WRITEIO registers should generally not be used. Most functions are available from STATUS and CONTROL register. Accessing WRITEIO registers can have undesirable effects by overriding settings that the software driver had set.

GPIO WRITEIO Registers

WRITEIO Register 0–Set PCTL
WRITEIO Register 1–Reset Interface
WRITEIO Register 2–Interrupt Mask
WRITEIO Register 3–Interrupt and DMA Enable
WRITEIO Register 4–MSB of Data Out
WRITEIO Register 5–LSB of Data Out
WRITEIO Register 6–Undefined
WRITEIO Register 7–Set Control Output Lines

Set PCTL

WRITEIO Register 0

Writing any non-zero numeric value to this register places PCTL in the Set state; writing zero causes no action.

Reset Interface

WRITEIO Register 1

Writing any non-zero numeric value to this register resets the interface.

Interrupt Mask

WRITEIO Register 2

Bit 7 **Value=128**	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
			0			Enable Interface Ready Interrupts	Enable EIR Interrupts

Interrupt and DMA Enable

WRITEIO Register 3

Bit 7 **Value=128**	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
Enable Interrupts				0			

MSB of Data Out

WRITEIO Register 4

Bit 7 **Value=128**	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

LSB of Data Out

WRITEIO Register 5

Bit 7 **Value=128**	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

Set Control Output Lines

WRITEIO Register 7

Bit 7 **Value=128**	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
			0			Set CTL1 (1= Low; 0= High)	Set CTL0 (1= Low; 0= High)

GPIO READIO Registers

Register 0−Interface Ready
Register 1−Card Identification
Register 2−Reserved
Register 3−Interrupt Status
Register 4−MSB of Data In
Register 5−LSB of Data In
Register 6−Reserved
Register 7−Peripheral Status

Interface Ready

READIO Register 0

A 1 indicates that the interface is Ready for subsequent data transfers, and 0 indicates Not Ready.

Board ID register

READIO Register 1

This register always contains 3, the identification for GPIO interfaces.

Interrupt Status

READIO Register 3

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
Interrupts Are Enabled	Interrupt is Requested			Undefined			DMA is Active

MSB of Data In

READIO Register 4

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

LSB of Data In

READIO Register 5

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

Peripheral Status

READIO Register 7

Bit 7 Value=128	Bit 6 Value = 64	Bit 5 Value = 32	Bit 4 Value = 16	Bit 3 Value = 8	Bit 2 Value = 4	Bit 1 Value = 2	Bit 0 Value = 1
Undefined				PSTS OK	EIR Line Low	STI1 Line Low	STI0 Line Low

Chapter 4 Programming Guide for GPIO 600

This chapter provides the board level information necessary to write a device driver that can control the GPIO board without TransEra supplied software. This is difficult and not recommended for most users. If a driver is being used, writing to any of the hardware registers directly can cause undesirable results due to conflicts with the driver.

Throughout this chapter setting a bit refers to changing the bit to a logical one. Clearing a bit refers to changing the bit to a logical zero.

The control or status register number is the offset added to the base address. This gives the physical IO address required to access the board. The base address is selected with switch number one. The GPIO 600 has an 8-bit interface so it only uses 8-bit registers.

Hardware Register Description

A short explanation of the registers is given. Data is written to control register and read from status registers. The Control or Status register number is the offset from the base address of that register.

Register Summary

Control Registers

Offset	**Name**
0	Set PCTL
1	Reset Interface
2	Interrupt Mask
3	Enable Interrupts
7	Set CTLx
8	Low Data Output
9	High Data Output
10	High Data Output and Set PCTL After Delay

Status Registers

Offset	**Name**
0	Interface Ready
1	Board ID
2	Interrupt Mask
3	Interrupt Status
7	Peripheral Status
8	Low Byte Data Buffer
9	High Byte Data Buffer
10	Start Read

Set PCTL

Control Register 0
Writing any value to this register location causes PCTL to be set.

Reset Interface

Control Register 1
Writing any value to this register location resets the interface.
The following occurs during a reset:

- The PRESET line pulses low for at least 15ms.
- The PCTL line is cleared.
- The output data lines are cleared if the DOUT Clear jumper is intalled.
- Interrupts are disabled.

Interrupt Mask

Control Register 2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		0				Enable Interface Ready Interrupts	Enable EIR Interrupts

Bit 0: Enable EIR Interrupts
When set EIR interrupts are requested when the EIR line goes low as long as the Enable Interrupts bit (register 3 bit 3) is set. When clear EIR interrupts are disabled.

Bit 1: Enable Interface Ready Interrupts
When set an interrupt is generated whenever the board is ready and the Enable Interrupts bit (register 3 bit 3) is set.

Enable Interrupts

Control Register 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	0			Enable Interrupts		0	

Bit 3: Enable Interrupts
This it the board level enable interrupts. When set the interrupt mask determines which interrupts are valid. When clear all interrupts are disabled.

Set CTLx

Control Register 7

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			0			Set CTL1 Low	Set CTL0 Low

Bit 0: Set CTL0 Low
When set the CTL0 line is forced low. When clear the CTL0 line goes in a high impedance state.

Bit 1: Set CTL1 Low
When set the CTL1 line is forced low. When clear the CTL1 line goes in a high impedance state.

Low Data Out

Control Register 8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

The data written to this register is output to the DO0-DO7 data lines (pins 17-10) and the IO line is forced low. The polarity of the data depends on the polarity selected by the DOUT switch.

High Data Out

Control Register 9

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

The data written to this register is output to the DO8-DO15 data lines (pins 9-2) and the IO line is forced low. The polarity of the data depends on the polarity selected by the DOUT switch.

High Data Out With PCTL

Control Register 10

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

The data written to this register is output to the DO8-DO15 data lines (pins 9-2) and the IO line is forced low. The polarity of the data depends on the polarity selected by the DOUT switch. Writing to this register instead of register 9 causes the PCTL line to be automatically set after the PCTL delay time. This is used to initiate output handshakes.

Interface Ready

Status register 0

Reading this register returns a 1 if the interface is ready. A 0 indicates that the interface is not ready. The conditions that result in interface ready are described in Chapter 2.

Board ID

Status register 1

This register always returns a 3. It is the board ID for the GPIO 600.

Interrupt Mask

Status register 2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Undefined				Interface Ready Interrupts Enabled	EIR Interrupts Enabled

This register always returns the interrupt mask. It is a copy of what was written to control register 2.

Interrupt Status

Status Register 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Undefined			Interrupts are Enabled	Interrupt is Requested	Undefined	

Bit 2: Interrupt is Requested
When set an interrupt is currently requested by the GPIO board. When clear no interrupts are beign requested by the board.

Bit 3: Interrupts are Enabled
This returns the value written to the Enable Interrupt bit (register 3 bit 3).

Pperipheral Status

Status Register 7

Bit 0: STI0 is Low
When the STI0 line (pin 47) is low this bit is set to one. When the STI0 line is high it is clear.

Bit 1: STI1 is Low
When the STI1 line (pin 48) is low this bit is set to one. When the STI1 line is high it is clear

Bit 2: EIR is Low
If the EIR line (pin 46) is low this bit is set. When the EIR line is high it is clear.

Bit 3: PSTS is OK
This bit is set if the PSTS line (pin 45) is in the OK state. The polarity of OK is user selectable with switch 2.

Low Data In

Status Register 8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

This register reads the low byte of the input data buffers. It also sets the IO line to a high impedance state.

High Data In

Status Register 9

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

This register reads the high byte of the input data buffers. It also sets the IO line to a high impedance state.

Start Read

Status Register 10
Reading this register sets the IO line to a high impedance state and sets PCTL to initiate an input read handshake.

IO Line Control

This section discusses the control or access of the IO lines.

CTL0

CTL0 is set or cleared by writing to the Set CTL0 bit (register 7 bit 0). When the CTL0 bit is set the line is driven low.

CTL1

CTL1 is set or cleared by writing to the Set CTL1 bit (register 7 bit 1). When the CTL1 bit is set the line is driven low.

EIR

The status of EIR is found by reading the EIR is Low bit (register 7 bit 3). When EIR is Low returns one the line is low.

IO

Conditions to set IO line
- Reading the Start Read register (register 10).
- Reading Low Data In (register 8) or High Data In (register 9) registers.

Conditions to clear IO line
- Writing to Low Data Out (register 8), High Data Out (register 9), or High Data Out With PCTL (register 10).

PCTL

Conditions to set the PCTL line
- Writing to Set PCTL (register 0).
- Starting a read cycle by reading Start Read (register 10).
- Writing to High Data Out With PCTL (register 10). This sets PCTL after the PCTL delay time.

Conditions to clear the PCTL line
- A ready-to-busy transition on PFLG.
- Writing to Reset Interface (register 1).

PRESET

The PRESET line is set low for at least 15μs by writing to Reset Interface (register 1).

PSTS

The PSTS line is checked by reading PSTS OK (register 7 bit 3). The polarity of the line is user selectable.

STI0

The status of STI0 is found by reading the STI0 is Low bit (register 7, bit 0). When set, the line is low.

STI1

The status of STI1 is found by reading the STI1 is Low bit (register 7, bit 1). When set, the line is low.

Interrupts

The two interrupt sources are controlled by the mask in Interrupt Mask and the global interrupt enable in Enable Interrupts. An interrupt source is selected when its corresponding bit is set in the Interrupt Mask. The selected source(s) can cause an interrupt when the Enable Interrupt bit of the Enable Interrupt register is set.

Which if any interrupt is active can be found by reading the Interrupt is Requested bit (register 3 bit 2). When both interrupts are enabled the source of the interrupt can be determined by checking the status of Interface Ready (register 0) and EIR is Low (register 7 bit 2).

Since the interrupt controller on an ISA bus is edge triggered the interrupt source must be cleared at some point in the interrupt handler before exiting. If it is never cleared the controller will not re-interrupt. This can be done by either clearing and then resetting the Enable Interrupts bit before exiting or reading Interrupt is Requested and looping back through the interrupt handler until it is clear.

Transferring Data

This section explains the steps required to transfer data between the board and peripheral. In all of the following transfer methods handshaking is required. If no handshaking is used, output and input single

bytes or words using the Output Data and Input Data registers.

Input Transfer with Polling

This is the simplest input transfer method. It cannot be run as a background process.

Step 1. Set PCTL and force the IO line high by writing to the Start Read register.

Step 2. Repetitively check the Interface Ready register until it is set to one. This signals the end of the handshake.

Step 3. Read the input data. If an 8-bit transfer read the Low Data In only. For a 16-bit data transfer also read the High Data In register to get the high data byte.

Step 4. If more data is to be input return to step 1, otherwise, the transfer is completed.

Output Transfer with Polling

This is the simplest output transfer method. It cannot be run as a background process.

Step 1. Output the low byte of data to the Low Data Out register. If it is a 16-bit transfer output the high byte to the High Data Out With PCTL register. For an 8-bit transfer, output a 0 to the High Data Out With PCTL register. PCTL will be set automatically after the PCTL Delay time.

Step 2. Repetitively check the Ready Interface register until set to one. This signals the end of the handshake.

Step 3. If more data is to be output return to step 1, otherwise, the transfer is completed.

Interrupt Transfer

An interrupt transfer performs the same steps as a polled transfer with two key differences. During the transfer, instead of waiting for the Ready bit to go high the program can continue until a ready interrupt occurs. This lets the transfer proceed as a background process. The second difference is that the input and output of data and control of IO and PCTL is all done in the interrupt handler.

The ready interrupt is enabled by setting Enable Ready Interrupts (register 2 bit 1) and Enable Interrupts (register 3 bit 3). Clearing either bit disables ready interrupts. After the desired amount of data has been transferred the interrupt handler should disable ready interrupts as part of completing the transfer.

Chapter 5 Programming Guide for GPIO 650

This chapter provides the board level information necessary to write a device driver that can control the GPIO board without TransEra supplied software. This is difficult and not recommended for most users. If a driver is being used, writing to any of the hardware registers directly can cause undesirable results due to conflicts with the driver.

Throughout this chapter setting a bit refers to changing the bit to a logical one. Clearing a bit refers to changing the bit to a logical zero.

The Action register (register 0) is so named because setting a bit causes an action to take place. When a one is written to a bit in this register the specified action takes place and the bit setting is discarded. Writing zeros to bit locations in the Action register has no effect. All other control registers are latched registers.

Every bit in a latched register assumes the written value. Keep a software copy of latched registers to prevent bits from being inadvertently cleared. The best way to modify a latched register is to change the software copy and then send the software copy to the board.

The GPIO board has a mix of 8 and 16 bit registers. For proper operation the data with the correct width must be sent to the registers. The registers that are indicated as 8 or 16 bits will correctly accept both data widths.

Hardware Register Description

A short explanation of the registers is given. Data is written to control register and read from status registers. The Control or Status register number is the offset from the base address of that register.

Register Summary

Control Registers

Offset	Name	Width
0	Action	8-bit
1	Board Control	8-bit
2	Board Setup	16-bit
4	Output Data	8/16-bit
5	Output High Data	8-bit
6	Handshake Data Output	16-bit

Status Registers

Offset	Name	Width
0	Board Status	16-bit
2	Board ID	8-bit
4	Input Data	8/16-bit
5	Input High Data	8-bit

Action

Control Register 0 (8-bit)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		0		Clear TC	Start Read	Reset Peripheral	Set PCTL

Bit 0: Set PCTL
Writing 1 to this bit causes the PCTL line to be set. Writing 0 does nothing.

Bit 1: Reset Peripheral
Writing 1 to this bit causes the PRESET line to pulse low for about 15us. PCTL is also put in the cleared state. Writing 0 does nothing.

Bit 2: Start Read
Writing 1 to this bit causes the board to start a read handshake cycle. It sets IO high and sets PCTL. Writing 0 does nothing.

Bit 3: Clear TC
Writing 1 to this bit clears the terminal count latch for channel A. Writing 0 does nothing.

Board Control

Control Register 1 (8-bit)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Word Transfers	0	Enable DMA Outputs	Enable DMA Inputs	Enable Interrupts	Enable TC Interrupts	Enable Ready Interrupts	Enable EIR Interrupts

Bit 0: Enable EIR Interrupts
When set, and if interrupts are enabled, an interrupt will be requested when EIR is low. If clear, the EIR line cannot cause an interrupt.

Bit 1: Enable Ready Interrupts
When set, and if interrupts are enabled, an interrupt will be requested when the interface is ready. If clear, ready interrupts will not be generated.

Bit 2: Enable TC Interrupts
When set, and if interrupts are enabled, an interrupt will be requested when a DMA terminal count is latched by the board. If clear, latched terminal counts cannot cause an interrupt.

Bit 3: Enable Interrupts
This is the general board level interrupt enable bit. When set, any enabled interrupts can take place. When clear, no interrupts will be requested.

Bit 4: Enable DMA Inputs
When set, the board attempts to perform DMA input transfers. When clear input DMA transfers are disabled.

Bit 5: Enable DMA Outputs
When set, the board attempts to perform DMA output transfers. When clear output DMA transfers are disabled.

Bit 7: Word Transfers
This controls the data width for DMA transfers only. When set the board outputs 16-bit data to the peripheral during DMA transfers. When clear 8-bit data are used with the high byte forced to a logical zero.

Board Setup

Control Register 2

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Clock high			Clock Low		PCTL Delay Time		0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

0	DMA Select	Interrupt Select	Ctl1 Line Low	CTL0 Line Low

Bit 0: CTL0 Line Low
When set, the CTL0 line (pin 22) is driven low. When clear, the CTL0 line output is set to a high impedance state.

Bit 1: Ctl1 Line Low
When set, the Ctl1 line (pin 23) is driven low. When clear, the Ctl1 line output is set to a high impedance state.

Bits 2-4: Interrupt Select
These bits select the physical PC interrupt that the board will use.

Bits 4-2	**Interrupt**
000	Disabled
001	IRQ5
010	IRQ7
011	IRQ9
100	IRQ10
101	IRQ11
110	IRQ12
111	IRQ15

Bits 5-6: DMA Select
These bits select the physical PC DMA channel that the boards DMA channel will use.

Bits 6-5	**DMA Channel**
00	Disabled
01	DMA5
10	DMA6
11	DMA7

Bits 9-11: PCTL Delay
These bits control the settling time for the output data. It is the time from the writing of the data using Handshake Data Output (register 6) until PCTL is set.

Bits 11-9	**Delay Time**
000	0-100ns
001	200-300ns
010	400-500ns
011	600-700ns
100	800-900ns
101	1.0-1.1μs
110	1.2-1.3μs
111	1.4-1.5μs

Bits 12-13: Clock Low
These bits select the clock source for the low data byte.

Bits 13-12	**Clock Source**
00	Switch Setting
01	RDY
10	BSY
11	RD

Bits 14-15: Clock High
These bits select the clock source for the high data byte.

Bits 15-14	Clock Source
00	Switch Setting
01	RDY
10	BSY
11	RD

Output Data

Control Register 4 (8/16-bits)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

An 8-bit write is output to the DO0-DO7 lines (pins 17-10). A 16-bit write controls both the Output Low Data Register and the Output High Data Register (register 5). The low byte is output to the DO0-DO7 lines while the high byte is output to the DO8-DO15 lines (pins 9-2).

Output High Data

Control Register 5 (8-bits)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

Writes to this register are output to the DO8-DO15 lines (pins 9-2). 16-bit writes to register 4 also control this register.

Handshake Data Output

Control Register 6 (16-bit)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

Writes to this register are output to the DO0-DO15 lines (pins 17-2). PCTL is then set automatically after the PCTL delay time to start the handshake.

Board Status

Status Register 0 (16-bit)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Undefined	Word Transfers	Undefined		DMA Terminal Count	Undefined	Active DMA	Interrupt is Requested

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC Interrupt	EIR Interrupt	Ready Interrupt	PSTS OK	EIR is Low	STI1 is Low	STI0 is Low	Ready

Bit 0: Ready
If set, the interface is ready for an IO transfer. If clear, an IO transfer is in progress.

Bit 1: STI0 is Low
If set, the STI0 line (pin 47) is low. If clear, the STI0 line is high.

Bit 2: STI1 is Low
If set, the STI1 line (pin 48) is low. If clear, the STI1 line is high.

Bit 3: EIR is Low
If set, the EIR line (pin 46) is low. If clear, the EIR line is high.

Bit 4: PSTS OK
If set, the PSTS check reports OK. The actual line polarity is controlled by the PSTS switch. If clear, PSTS check is not OK.

Bit 5: Ready Interrupt
If set, a ready interrupt is requested. If clear, a ready interrupt is not requested.

Bit 6: EIR Interrupt
If set, an EIR interrupt is requested. If clear, an EIR interrupt is not requested.

Bit 7: TC Interrupt
If set, a DMA terminal count interrupt is requested. If clear, a DMA terminal count interrupt is not requested.

Bit 8: Interrupt is Requested
If set, an interrupt is requested. This is a logical OR of the three specific interrupt sources. If clear, no interrupt is requested.

Bit 9: Active DMA
If set, DMA is currently active. If clear, DMA is not active.

Bit 11: DMA Terminal Count
The DMA controller has sent a terminal count signal to the board while the board was performing a DMA transfer. This bit remains set until cleared by a Clear TC strobe (register 0 bit 3). If clear, a valid terminal count has not occurred.

Bit 14: Word Transfers
This returns the value of the Word Transfers bit (register 1, bit 7).

Board ID

Status Register 2 (8-bits)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	1	0	0	0	1	1

This register returns the board ID number of 99 (63 hex).

Input Data

Status Register 4 (8/16-bits)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

An 8-bit read returns the value of the low byte of the input data buffer that is connected to DI0-DI7 lines (pins 42-35). A 16-bit read returns the value of input low data buffer as the least significant byte and the value of the input high buffer as the most significant byte.

Input High Data

Status Register 5 (8-bits)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

Returns the value stored in the input high data buffer that is connected to pins DI8-DI15 (pins 34-17).

IO Line Control

This section discusses the control or access of the IO lines.

CTL0

CTL0 is set or cleared by writing to the Set CTL0 bit (register 2 bit 0). When the CTL0 bit is set the line is driven low.

CTL1

CTL1 is set or cleared by writing to the Set CTL1 bit (register 2 bit 1). When the CTL1 bit is set the line is driven low.

EIR

The status of EIR is found by reading the EIR is Low bit (register 0, bit 3). When EIR is Low returns one the line is low.

IO

Conditions to set IO line
- Writing a one to Start Read (register 0 bit 2).
- Reading Input Data (register 4) or Input High Data (register 5) registers.
- Enabling an input DMA cycle by setting the Enable DMA Inputs bit (register 1, bit 4).

Conditions to clear IO line
- Writing to Output Data (register 4), Output High Data (register 5), or Handshake Data Output (register 6).
- Enabling an output DMA cycle by setting the Enable DMA Outputs bit (register 1, bit 5).

PCTL

Conditions to set the PCTL line
- Writing a 1 to Set PCTL (register 0 bit 0).
- Starting a read cycle by writing a 1 to Start Read (register 0 bit 2).
- Writing to Handshake Data Output (register 6). This sets PCTL after the PCTL delay time.

Conditions to clear the PCTL line
- A ready-to-busy transition on PFLG.
- Writing a 1 to Reset Peripheral (register 0 bit 1).

PRESET

The PRESET line is set low for about 15us by writing a 1 to the Reset Peripheral bit (register 0, bit 1).

PSTS

The PSTS line is checked by reading PSTS OK (register 0 bit 4). The polarity of the line is user

selectable.

STI0

The status of STI0 is found by reading the STI0 is Low bit (register 0, bit 1). When set, the line is low.

STI1

The status of STI1 is found by reading the STI1 is Low bit (register 0, bit 2). When set, the line is low.

Initializing the Board

Before using the board the input clock source, PCTL delay time, interrupt number, DMA channel(s) and transfer width should be selected.

Input Clock Source

The input clock source for the high and low byte is controlled by the Clock High and Clock Low bit fields (register 2 bits 12-15).

Bits 13-12	Clock Source
00	Switch Setting
01	RDY
10	BSY
11	RD

Bits 14-15: Clock High
These bits select the clock source for the high data byte.

Bits 15-14	Clock Source
00	Switch Setting
01	RDY
10	BSY
11	RD

PCTL Delay Time

The PCTL delay time is set with the PCTL Delay Time bit field (register 2, bits 8-10).

Bits 11-9	Delay Time
000	0-100ns
001	200-300ns
010	400-500ns
011	600-700ns
100	800-900ns
101	1.0-1.1us
110	1.2-1.3us
111	1.4-1.5us

Interrupt Number

The board's interrupt number is selected using the Interrupt Select bit field (register 2 bits 2-4).

Bits 4-2	Interrupt
000	Disabled
001	IRQ5
010	IRQ7
011	IRQ9

100	IRQ10
101	IRQ11
110	IRQ12
111	IRQ15

DMA Channel

The DMA channels are selected with the DMA Channel bit field (register 2 bits 5-6).

Bits 6-5	DMA Channel
00	Disabled
01	DMA5
10	DMA6
11	DMA7

Interrupts

The three interrupt sources are controlled by four bits in the DMA and Interrupt Control register (register 1). Enable EIR Interrupts, Enable Ready Interrupts and Enable TC Interrupts enable each of these interrupt sources individually. They make up the interrupt mask. The Enable Interrupt bit is the global enable that must be set for any interrupt to be enabled.

Which if any interrupt is active can be found by reading Board Status (register 0). Interrupt is Requested (bit 8) indicates that at least one interrupt is active. TC Interrupt, EIR Interrupt and Ready Interrupt are set if that particular interrupt is active.

Since the interrupt controller on an ISA bus is edge triggered the interrupt source must be cleared at some point in the interrupt handler before exiting. If it is never cleared the controller will not re-interrupt. This can be done by either clearing and then resetting Enable Interrupts before exit or reading Interrupt is Requested and looping back through the interrupt handler until it is clear.

Transferring Data without DMA

This section explains the steps required to transfer data between the board and peripheral. In all of the following transfer methods handshaking is required. If no handshaking is used, output and input single bytes or words using the Output Data and Input Data registers.

The required board information for controlling transfers with the GPIO board is included. The information necessary to program the DMA controller, interrupt controller, and properly allocate memory is beyond the scope of this manual.

Input Transfer with Polling

This is the simplest input transfer method. It cannot be run as a background process.

Step 1. Set PCTL and force the IO line high by writing a one to Start Read (register 0 bit 2).

Step 2. Repetitively check Ready (register 0, bit 0) until set. This signals the end of the handshake.

Step 3. Read the input data. Use an 8-bit read of Input Data Register (register4) for byte transfers or a 16-bit read of the same register for word transfers.

Step 4. If more data is to be input return to step 1, otherwise, the transfer is completed.

Output Transfer with Polling

This is the simplest output transfer method. It cannot be run as a background process.

Step 1. Output the data, set PCTL and set IO low by performing a 16-bit write to the Handshake Data Output register (register 6). If performing a byte transfer the high byte should be set to zero.

Step 2. Repetitively check Ready (register 0, bit 0) until set. This signals the end of the handshake.

Step 3. If more data is to be output return to step 1, otherwise, the transfer is completed.

Interrupt Transfer

An interrupt transfer performs the same steps as a polled transfer with two key differences. During the transfer, instead of waiting for the Ready bit to go high the program can continue until a ready interrupt occurs. This lets the transfer proceed as a background process. The second difference is that the input and output of data and control of IO and PCTL is all done in the interrupt handler.

The ready interrupt is enabled by setting Enable Ready Interrupts (register 1 bit 1) and Enable Interrupts (register 1 bit 3). Clearing either bit disables ready interrupts. After the desired amount of data has been transferred the interrupt handler should disable ready interrupts as part of completing the transfer.

DMA Transfers

DMA transfers are performed by the DMA controller on the computer. This allows a background transfer that is faster and more deterministic than interrupt or polled transfers.

The DMA hardware on the board controls handshaking during the DMA cycles. The IO line is forced high when Enable DMA Inputs (register 1 bit 4) is set and low when Enable DMA Outputs (register 1 bit 5) is set. This will override any attempts to change the IO line setting by other software means. The PCTL line is set as needed to carry out the transfer.

All DMA transfers are 16-bit transfers. If the interface is set for 8-bit transfers the board hardware will perform two 8-bit handshakes for every 16-bit DMA transfer. In this mode the low byte is first in the 8-bit data stream. This allows the bytes to be stored in memory in the same order as the handshakes.

DMA Transfer Steps

If only one DMA controller configuration is required all references to interrupts may be ignored.

Step 1. Make sure that DMA is disabled by clearing Enable DMA Inputs (register 1 bit 4) and Enable DMA Outputs (register 1 bit 5). The interface should be set for the desired byte or word transfers with the Word Transfer bit (register 1 bit 7).

Step 2. Configure the DMA controller.

Step 3. Verify that no old terminal counts are latched by writing a one to Clear TC (register 0 bit 3).

Step 4. Start the DMA transfer and enable TC interrupts by setting Enable DMA Inputs (register 1 bit 4) for input transfers, Enable DMA Outputs (register 1 bit 5) for output transfers, Enable TC Interrupts(register 1 bit 2), and Enable Interrupts (register 1 bit 3).

The DMA controller will cause a TC interrupt each time the controller has completed the programmed transfer. The interrupt handler must then reset the DMA controller to transfer more data or clear the bits set in step 4 to end the transfer. The interrupt handler is discussed in the DMA Interrupt Handler section.

When terminating an input DMA handshake clear Enable DMA Input before clearing the TC latch. When input DMA is enabled the board automatically performs input handshakes until TC is latched. If cleared in the wrong order the board will attempt to perform one extra handshake in word mode and two extra in byte mode.

When inputting or outputting odd numbers of byte data the last byte cannot be output with DMA. Since each DMA transfer moves two bytes of data there is no way to prevent an extra handshake for the second

not needed byte. To avoid this problem set up the DMA controller to stop before the last byte is output or entered. The last byte can then be easily handled by transferring it from the interrupt handler after DMA has been disabled.

When performing a DMA transfer an potential interrupt is generated each time the DMA controller has finished transferring the configured amount of data. By interrupting on this event the controller can be reconfigured to transfer more data or the board's DMA control bits can be cleared to end the DMA transfer.

DMA Interrupt Handler

Step 1. Verify that it is a TC interrupt by checking to see if TC Interrupt (register 0 bit 7) is set. If it is not a TC interrupt take care of the interrupt source and leave the interrupt handler. If more data is to be transferred proceed to step 2 otherwise go to step 4.

Step 2. Reprogram the DMA controller to transfer from/to the next block of memory.

Step 3. Clear the terminal count by writing a one to Clear TC (register 0, bit 3). Goto step 5.

Step 4. Clear either Enable DMA Outputs (register 1 bit 5) or Enable DMA Inputs (register 1 bit 4), Enable TC Interrupts (register 1 bit 2) and, if no other interrupts are active, Enable Interrupts (register 1 bit 3).

Step 5. Check that no interrupts are requested by reading the Interrupt is Requested bit (register 0 bit 8). If an interrupt is still requested handle it before exiting.

Step 6. Reinitialize the interrupt controller.

Step 7. Exit the interrupt handler.

{ewl RoboEx32.dll, WinHelp2000, }