**DYNAMIC ENGINEERING**
435 Park Dr., Ben Lomond, Calif. 95005
831-336-8891 **Fax** 831-336-3840
**Web Page** http://www.dyneng.com
**E-Mail** sales@dyneng.com
engineering@dyneng.com
support@dyneng.com
Est. 1988

## User Manual

# IP-BiSerial-BA5

Bi-directional Serial Data Interface
IP Module

Revision A
Corresponding Hardware: Revision 2

# IP-BiSerial

version BA5

Digital Serial Interface

IP Module

**Dynamic Engineering**

435 Park Drive

Ben Lomond, CA 95005

831-336-8891

831-336-3840 FAX

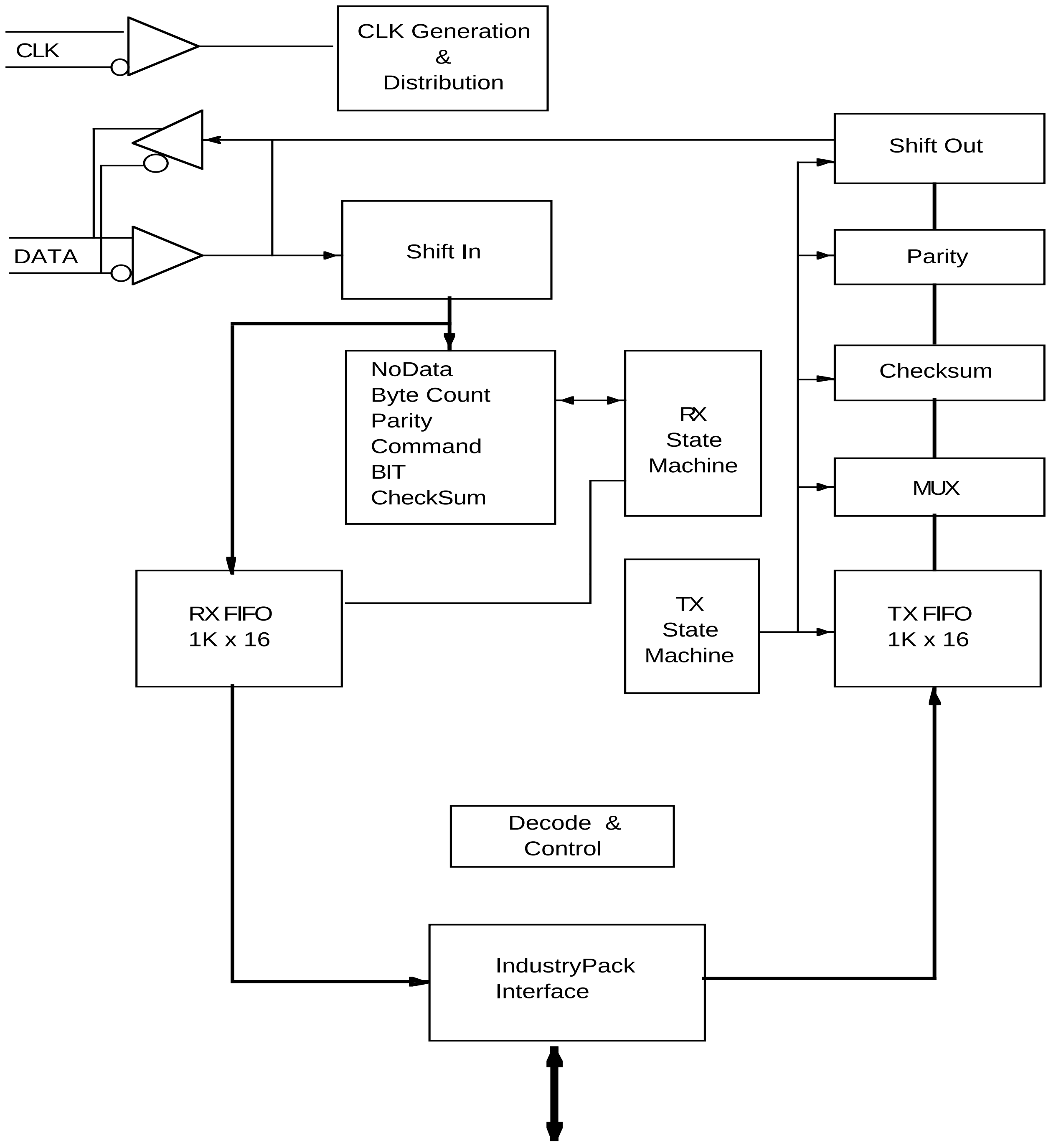
# Table of Contents

Product Description and Operation ............ 5
Address Map ............ 9
Programming ............ 10
Register Definitions ............ 12
BIS_CNTL0 12
BIS_CNTL1 15
BIS_CNTL2 16
BIS_CNTL3 17
BIS_VECTOR 18
BIS_STAT1 19
BISERIAL_RESET 20
BIS_TX_FIFO_0_W 20
BIS_TX_FIFO_1_W 20
BIS_TX_FIFO_0_R 20
BIS_TX_FIFO_1_R 20
BIS_RX_FIFO_0_W 21
BIS_RX_FIFO_1_W 21
BIS_RX_FIFO_0_R 21
BIS_RX_FIFO_1_R 21
Interrupts ............ 22
ID PROM ............ 24
IP Module Logic Interface Pin Assignment ............ 25
IP Module IO Interface Pin Assignment ............ 26
Applications Guide ............ 27
Interfacing ............ 27
Construction and Reliability ............ 28
Thermal Considerations ............ 29
Warranty and Repair ............ 29
Service Policy ............ 30
Out of Warranty Repairs 30
For Service Contact: ............ 30
Specifications ............ 31
Order Information ............ 32
Programming Reference ............ 33
Programmable Almost Empty Flag Example ............ 33

# List of Figures

FIGURE 1 IP-BISERIAL-BA5 INTERNAL ADDRESS MAP 9
FIGURE 2 IP-BISERIAL-BA5 CONTROL REGISTER 0 BIT MAP 12
FIGURE 3 IP-BISERIAL-BA5 CONTROL REGISTER 1 BIT MAP 15
FIGURE 4 IP-BISERIAL-BA5 CONTROL REGISTER 2 BIT MAP 16
FIGURE 5 IP-BISERIAL-BA5 CONTROL REGISTER 3 BIT MAP 17
FIGURE 6 IP-BISERIAL-BA5 STATUS REG 0 BIT MAP 18
FIGURE 7 IP-BISERIAL-BA5 STATUS REG 1 BIT MAP 19
FIGURE 8 IP-BISERIAL-BA5 ID PROM 24
FIGURE 9 IP-BISERIAL-BA5 LOGIC INTERFACE 25
FIGURE 10 IP-BISERIAL-BA5 IO INTERFACE 26

## Product Description and Operation



IP-BISERIAL-BA5 is part of the IP Module family of modular I/O components. The IP-BISERIAL-BA5 is capable of providing multiple serial protocols. The standard protocol implemented provides a Data and Clock interface. The -BA5 version is a custom modification. The main modifications include the use of a free running clock and modeling the data transfer to use a UART protocol. The -BA5 idles in the receive mode and waits for a command to be sent via the data IO. When a command is recognized, the hardware will respond by transmitting the message stored in the transmit FIFO along with status derived from the received message. Once the transmission is complete the -BA5 returns to receive mode and waits for the next message.

In addition to the BA5 version, other custom interfaces are available. Please see our web page for current protocols offered. If you do not find it there, we will redesign the state machines and create a custom interface protocol. That protocol will then be offered as a "standard" special order product. Please contact Dynamic Engineering with your custom application.

The IP-BISERIAL-BA5 supports both 8 and 32 Mhz. IP Bus operation. The IP Clock or external reference is used to derive the reference clocks for the serial operation. Please be sure to select the proper clock divisors and source selector after reset to insure proper operation. Please refer to the programming section for details.

Differential I/O is available on the serial signals. The differential drivers and receivers conform to the RS-485 specification (exceeds RS-422 specification). The RS-485 input signals are terminated with 180Ω.

All configuration registers support read and write operations for maximum software convenience. Word operations are supported (please refer to the memory map).

The IP-BISERIAL-BA5 conforms to the VITA standard. This guarantees compatibility with multiple IP Carrier boards. Because the IP may be mounted on different form factors, while maintaining plug and software compatibility, system prototyping may be done on one IP Carrier board, with final system implementation on a different one. The PCI3IP card makes a convenient development platform in many cases. http://www.dyneng.com/pci_3_ip.html

The serial receive channel is supported by a 1K by 16 bit FIFO. The FIFO supports byte and word reads. Word reads are recommended to keep the FIFOs synchronized at all times. A byte wide write path exists for loop-back testing. The receive channel looks for data in 30 word transfers. The received words are loaded into the FIFOs. The data length loaded is determined and an error generated if an incorrect length message is received. The host can poll or wait for the message complete interrupt.  The message can be read directly from the input FIFO.  The incoming message will have a parity bit attached to each byte. Odd parity is the normal selection.  Parity type is programmable.

The data is all stored into the input FIFO.  The data words 1-29 are used to calculate a checksum which is then compared against the 30th word.  The parity, word count, message ID, checksum status are all reported back with the next transmit word by the hardware.  The status is also available to read by the IP host processor via status register 1.

In addition to the status checks the receiver hardware looks for the presence of the BIT command and if present in the Message ID field and the run bit is set in the next word then the Bit in Progress field is set.  The Bit In Progress bit remains set until the host software sets the Bit Complete control bit and the Bit Results field.  When the hardware is initialized and has not run BIT the BIT in Progress and BIT Completed fields are set to '0' and the status returned is as programmed by the software.  When the BIT command is received then the BIT in Progress bit is set and stays set until the host software sets the bit complete control bit.  When the control bit is set the flags will be updated to indicate that BIT is Complete and not in progress and the status as reported by the carrier software.  The status will remain in this state until a BIT request is received again.  The Bit flags will reverse to show BIT in progress and BIT not complete. The state machine which controls the status flags will reset the carrier control flag once the flag has been recognised.

Once enabled and data is detected, the data is read in and parity checked.  The hardware then searches for the next start bit or the no data condition.  The hardware can receive messages with at most 9 marking bits between messages. When the no data is detected the message capture is terminated.  Each time a new byte is received the byte counter is incremented.  When the no data condition is detected the byte counter is checked to see if all 30 words are received.  Under and over-run errors are captured.  The hardware can operate with 1 stop bit and a continuous message for the 30 words.

The Output channel has a separate 1k x 16 FIFO.  The FIFO can be written as words or bytes. Word writes are recommended to keep the FIFOs synchronized. The upper and lower bytes are read together and sent as part of a 30 data word transfer.  The FIFO can be accessed directly for loop back testing.  The data is available in a byte wide format when looped back.  Parity is added to the data by the hardware.  The parity options are Odd or Even.  The transmit data is composed of the first three data words which are supplied by the internal hardware, words 4-29 which are read from the FIFO and the 30th word which is a checksum.  The checksum is automatically calculated from the words transferred 1-29 and then appended as the 30th word.

The transmitter section sends data as a burst with the minimum of 1 stop bit between messages.  Transmission begins when the received message has been determined to have completed.  Once the "no data" flag is set the state machine switches the data buffer direction and sends a trigger pulse to the transmitter state machine.  The transmitter waits a programmable delay and then starts to send data.  The minimum delay is longer than the minimum of 5 uS from the end of reception.  The maximum delay can be much longer based on the count loaded by software.

The transmitter also has a tx_testmode bit which over-rides the requirement of a received message to transmit data.  By loading data into the FIFO and then enabling the transmitter and receiver with the tx_testmode selected the data can be looped back into the receiver [requires external loop-back connection].

Received data is captured on the rising edge of the reference clock and the transmitted data is referenced to the falling edge of the clock.

Interrupts are supported by the IP-BISERIAL-BA5.  The interrupt occurs at the end of the transmission whether data is received or sent or both.  The interrupts are individually maskable.  The vector is user programmable by a read/write register. The interrupt occurs on IntReq0. The FIFO status is available for the FIFO making it possible to operate in a polled mode.

The IP-BISERIAL-BA5 features a Xilinx FPGA. The FPGA contains all of the registers and protocol controlling elements of the BISERIAL design. Only the drivers, receivers, boot PROM and FIFOs are external to the Xilinx device.

The bus interface to the host CPU is controlled by a logic block within the Xilinx device that contains the decoding and timing elements required to interface to the IP bus interface. The timing is referenced to the 8 or 32 MHz IP logic clock. The IP responds to the ID, INTSEL, MEM and IO selects. The DMA control lines are connected to the Xilinx for future revisions, and are not used at this time. The BISERIAL design requires wait states for read or write cycles to any address. Hold cycles are supported as required by the host processor. Data remains enabled during a read until the host removes the SEL line. Local timing terminates a write cycle prior to ACK being asserted. If no hold cycles are requested by the host, the IP-BISERIAL-BA5 is capable of supporting 16+ MB per second data transfer rate with a 32 Mhz. reference  rate.

## Address Map

| Function | | Offset | Width | Type |
|---|---|---|---|---|
| BIS_CNTL0 | EQU | $00 | word | read/write |
| BIS_CNTL1 | EQU | $02 | word | read/write |
| BIS_CNTL2 | EQU | $04 | word | read/write |
| BIS_CNTL3 | EQU | $06 | word | read/write |
| BIS_VECTOR | EQU | $08 | word | read/write |
| BIS_STAT0 | EQU | $0A | word | read |
| BIS_STAT1 | EQU | $0C | word | read |
| BIS_STAT2 | EQU | $0E | word | read - not used -BA5 |
| BIS_RESET | EQU | $0E | word | write |
| | | | | |
| BIS_TX_FIFO_0_W | EQU | $10 | D15..8 byte or word | write |
| BIS_TX_FIFO_1_W | EQU | $11 | D7..0 byte | write |
| BIS_TX_FIFO_0_R | EQU | $10 | byte on word boundary | read |
| BIS_TX_FIFO_1_R | EQU | $14 | byte on word boundary | read |
| | | | | |
| BIS_RX_FIFO_0_W | EQU | $20 | byte on word boundary | write |
| BIS_RX_FIFO_1_W | EQU | $24 | byte on word boundary | write |
| BIS_RX_FIFO_0_R | EQU | $20 | D15..8 byte or word | read |
| BIS_RX_FIFO_1_R | EQU | $21 | D7..0 byte | read |
| | | | | |
| BISERIAL_IDPROM | EQU | $80 | byte on word boundary | read |
| MEM Space Read any address = RX FIFO Read | | | | |
| MEM Space write any address = TX FIFO Write | | | | |

FIGURE 1 IP-BISERIAL-BA5 INTERNAL ADDRESS MAP

The address map provided is for the local decoding performed within the IP-BISERIAL-BA5. The addresses are all offsets from a base address. The carrier board that the IP is installed into provides the base address and controls the "naming of the bytes". We refer to the bytes following Motorola conventions. i.e. upper is D15-D8 and lower is D7-D0. When byte wide data is located on the lower byte then an odd address results or the use of a word access using only the lower byte of data. We prefer the word oriented approach because it is more consistent across platforms.

## Programming

Programming the IP-BISERIAL-BA5 requires only the ability to read and write data in the host's I/O space. The base address is determined by the IP Carrier board.

In order to receive data the software is only required to enable the RX state machine and FIFOs. If desired, the interrupt can be enabled and the interrupt vector written to the vector register. Data will be loaded into the FIFOs as it is received.

A typical sequence would be to first write to the vector register with the desired interrupt vector. For example, $40 is a valid user vector for the Motorola 680x0 family. Please note that some carrier boards do not use the interrupt vector. The interrupt service routine should be loaded and the mask should be set. When the start bit is set the hardware looks to make sure that a message is not in progress, then begins looking for a message to start. In this manner, the data received is protected from joining mid-message. Once a new message is detected the data loading process begins. When the no data condition is detected the interrupt request is asserted to let the host know that the data is available. The software can read the data from the FIFOs efficiently based on the preprogrammed word count. The transfer status should be checked to determine the validity of the message received.

The end of transmission interrupt will indicate to the software that the message has been started and that the message has terminated. If both the TX and RX interrupts are enabled then the SW needs to read BIS_STAT0 to see which source caused the interrupt. Reading BIS_STAT0 will clear the interrupt status, and the INTACK cycle will clear the actual interrupt. The interrupt status can be read after the INTACK cycle. It is a good idea to read the status register to force the RX_INT and TX_INT bits to 0 before Start is enabled to insure that the RX_INT or TX_INT=1 value read by the interrupt service routine came from the current operation.

Before transmitting data the FIFOs are enabled and the data loaded. The BA5 design can use an internal [32 or 8 MHz] or external 1 MHz.reference. The baud rate selections are used to select the output rate. CLK_HI must be set to the proper level if using the internal rates. CLK_HI selects [or not] the prescaler. Alternate baud rates are available if CLK_HI is set mismatched to the IP reference rate. Once the complete message is loaded and the controls set properly the start bits can be set to cause the transfer to begin.

The -BA5 version of the BiSerial implements Memory Space accesses to allow multiple addresses to be mapped into the FIFOs. This is useful if the processor supports bursts or automatic bus sizing.  A 32 bit write/read with some CPUs will result in two 16 bit accesses to the hardware with automatic incrementing addresses.  The 32 bit access is quite a bit faster than a software loop.  If the host processor supports burst or DMA to the IP MEM space then the resulting incrementing addresses all map into the FIFO location.  This type of access is usually faster than a software loop.

Refer to the Theory of Operation section above and the Interrupts section below for more information regarding the exact sequencing and interrupt definitions.

## Register Definitions

### BIS_CNTL0

$00 BISERIAL Control Register Port read/write

**CONTROL REGISTER 0**

| DATA BIT | DESCRIPTION |
|---|---|
| 15 | spare |
| 14 | spare |
| 13 | TX_xtra_1 |
| 12 | TX_testmode |
| 11 | FAE_EN 1 = Interrupt Enabled |
| 10 | TX_INT_EN |
| 9 | TX_PAR_TYPE |
| 8 | spare |
| 7 | CLK_HI_B |
| 6 | BR2 |
| 5 | BR1 |
| 4 | BR0 |
| 3 | EXT_INT |
| 2 | INT_SET |
| 1 | INT_EN |
| 0 | STRT_TX |

FIGURE 2 IP-BISERIAL-BA5 CONTROL REGISTER 0 BIT MAP

1. All bits are active high and are reset on power-up or reset command. Spare bits are R/W but have no system affect.

2. CLK_HI_B is used to let the HW know which IP reference clock is present to derive the TX Clock rate from. If running at 32 MHz. CLK_HI_B should be set. If not set when the faster IP clock is used then the baud rates will all be off by a factor of 4 [lower than actual]. Depending on cable length the faster rates may work. If set when the IP clock rate is 8 Mhz then the Baud rates will also be a factor of 4 off [faster than actual].

3. EXT_INT is used to select the clock source for the transmitter to be the external clock or one derived from the IP clock. 0 = internal, 1 = external. External = 1 MHz. source for this design. RX_Rdy input signal.

4. BR2, BR1 and BR0 are the bit rate selection bits for generating the external reference clock. and the TX transmit clock.

| **Bit Pattern** | **Divisor** | **clk hi & 32**<br>**or !clk hi & 8** | **1 MHz** |
|---|---|---|---|
| 000 | 1 | default = 8 MHz. | 1 MHz. |
| 001 | 2 | 4 MHz. | .5 MHz. |
| 010 | 4 | 2 MHz. | .25 MHz. |
| 011 | 8 | 1 MHz. | .125 MHz. |
| 100 | 16 | .500 MHz. | .0625 MHz. |
| 101 | 32 | .250 MHz. | .03125 MHz. |
| 110 | 64 | .125 MHz. | .01563 MHz. |
| 111 | 128 | .0645 MHz. | .00781 MHz. |

5. INT_SET is used to create an interrupt for test and software development purposes. Set the bit to cause the interrupt and clear the bit to remove the interrupt.

6. TX_INT_EN is the Interrupt Enable bit for the Transmit channel. The default state is off. If enabled and the master interrupt enable is also enabled then an interrupt is requested when the transmission is complete. The interrupt is cleared by reading the status register.

7. STRT_TX is set to send data. The bit is auto cleared at the end of a transmission.

8. TX_PAR_TYPE defines odd or even parity to be used. 1 = Odd and 0 = even.

9. INT_EN is the master interrupt enable. Default is 0. If set to 1 then either the RX or TX interrupts can occur based on the state machines and the state of the RX and TX interrupt enables. If the master interrupt enable is off [0] then no interrupts will be generated. The status register can see the interrupt requests from the RX and TX state machines to allow polled operation.

10. FAE_EN = 1 to enable the FIFO Almost Empty interrupt. When enabled an interrupt is generated when the data level falls to the programmed level.

11. TX_testmode is set to put the hardware into transmission test mode. The Transmit hardware can transmit without waiting for a received message. Several mods to the receive error checking allow auto incrementing message counts, bypassed second check for BIT etc. to allow for loop-back testing. The data is transmitted from the TX data port and received into the RX data port. The Transmit data port is tri-stated when the receiver switches to response mode. The state machines also have special paths to allow for non-standard termination conditions etc.

12. TX_xtra_1 is another test bit to cause the transmitter to send more than 30

words.  The last word read from the FIFO is resent after the checksum if this bit is set.  For shorter messages place less data into the FIFO.  The transmitter will stop when the FIFO becomes empty and report an error for an underflow.

**BIS_CNTL1**

[$02 BISERIAL Control Register Port read/write

**CONTROL REGISTER 1**

| **DATA BIT** | **DESCRIPTION** |
|---|---|
| 15-8 | Word 1 ID |
| 7 | SPARE |
| 6 | SPARE |
| 5 | SPARE |
| 4 | SPARE |
| 3 | SPARE |
| 2 | SPARE |
| 1 | mbit status |
| 0 | mbit completed |

FIGURE 3 IP-BISERIAL-BA5 CONTROL REGISTER 1 BIT MAP

Mbit Status is set to indicate a "passed Bit" condition. The status is passed as part of the BIT response word.

Mbit completed is set to indicate that the BIT tests have completed. The hardware will automatically reset the bit when the state machine has recognized the completed flag.

Word 1 ID is returned as the ID portion of Word1. The Message count is passed for the other portion.

**BIS_CNTL2**

[$04 BISERIAL Control Register Port read/write

**CONTROL REGISTER 2**

| DATA BIT | DESCRIPTION |
|---|---|
| 15 | SPARE |
| 14 | FAF_EN 1 = Interrupt Enabled |
| 13 | SPARE |
| 12 | SPARE |
| 11 | SPARE |
| 10 | SPARE |
| 9 | SPARE |
| 8 | SPARE |
| 7 | RX_INT_EN |
| 6 | FRX_LD |
| 5 | FTX_LD |
| 4 | CLR_FIFO |
| 3 | TESTMODE |
| 2 | RX_PAR_TYPE |
| 1 | SPARE |
| 0 | STRT_RX |

FIGURE 4 IP-BISERIAL-BA5 CONTROL REGISTER 2 BIT MAP

1. CLR_FIFO is used to reset the FIFOs. The default state is reset. The FIFOs must be taken out of reset to be used to store data. Please refer to FTX_LD and FRX_LD .

2. FRX_LD is tied to the RX FIFO WE2/_LD pin. FTX_LD is tied to the TX FIFO WE2/_LD pin. When the FIFOs are taken out of reset it is possible to set-up the FIFO to accept commands to program the way the programmable almost empty and programmable almost full signals operate. ***In the standard transfer mode these pins are set hi before CLR_FIFO is released to use as a second WE control pin***. If the PAE and PAF flags are used for a different protocol then the FIFOs will require programming.

3. RX_ INT_EN is used to enable the receive interrupt. The default is disabled. If enabled and the master interrupt enable is also enabled, an interrupt is requested when the received message is complete. The interrupt is cleared by reading the status register.

4. STRT_RX is used to enable the receive state machine to receive messages. The start bit is auto-cleared at the end of a transmission.

5. TESTMODE is used to select the reference clock to the RX FIFOs. In test mode

a faster clock is provided to the FIFOs to allow the interface to keep up with the IP Bus requirements.  Default is 0.  Normal operation is 0.  The IP CLOCK hi low bit should be set to low independent of IP clock speed so that the FIFO reference is the IP Clock at 8 or 32 MHz.

6. RX_PAR_TYPE = 1 for odd and = 0 for even parity.

7. FAF_EN = 1 to enable the FIFO Almost Full interrupt for the receive function. The point where the interrupt occurs is programmable.

**BIS_CNTL3**

[$06 BISERIAL Control Register Port read/write

**CONTROL   REGISTER 3**

| **DATA BIT** | **DESCRIPTION** |
|---|---|
| 15-12 | Spare |
| 11-0 | TX Delay Count |

FIGURE 5 IP-BISERIAL-BA5 CONTROL REGISTER 3 BIT MAP

The TX Delay Count is used to add additional delay from the end of the received message to the start of the transmission of data.  The reference clock period times the number loaded into the register will be the length of the delay.

**BIS_VECTOR**

[$08] BISERIAL Interrupt Vector Port
The Interrupt vector for the BISERIAL is stored in this byte wide register. This read/write register is initialized to 'xxFF' upon power-on reset or software reset. The vector is stored in the odd byte location [D7..0]. The vector should be initialized before the interrupt is enabled or the mask is lowered. The interrupt is automatically cleared when the CPU acknowledges the interrupt.

BIS_STAT0
[$0A] BISERIAL Status Port [read only]

| **Data Bit** | **Status** | |
|---|---|---|
| 15 | FAE_INT | FIFO Almost Empty Int 1 = active |
| 14 | FTX_AE0 | PAE channel 0 [MSB] 0 = tx almost empty |
| 13 | FAF_INT | FIFO Almost Full Int 1 = active |
| 12 | FRX_AF0 | PAF channel 0 [MSB] 0 = rx almost full |
| 11 | GND | read 0 |
| 10 | GND | read 0 |
| 9 | GND | read 0 |
| 8 | FE_ERR | 1 = FIFO Empty Error |
| 7 | TX_INT | 0 = no interrupt request 1 = interrupt request |
| 6 | FTX_FF_0 | 0 = full 1 = not full |
| 5 | FTX_MT_0 | 0 = empty 1 = not empty |
| 4 | FTX_MT_1 | 0 = empty 1 = not empty |
| 3 | RX_INT | 0 = no interrupt request 1 = interrupt request |
| 2 | FRX_MT_0 | 0 = empty, 1 = not empty |
| 1 | FRX_FF_0 | 0 = full 1 = not full |
| 0 | FRX_FF_1 | 0 = full 1 = not full |

FIGURE 6 IP-BISERIAL-BA5 STATUS REG 0 BIT MAP

1. The FIFO flags are active low. When the empty bit is low then the FIFO is empty. When the empty flag is high then the FIFO has at least one piece of data stored. When the Full Flag is set [low] the FIFO is full. When not set then the FIFO still has room.

2. FE_ERR is set when the TX state machine detects a FIFO Empty condition before completing data transmission. Cleared when read.

5. FAE_INT, FAF_INT are set if the interrupt is enabled and the condition occurs. FAE_INT is set if the FIFO data "level" drops to the programmed level. FAF_INT is

set if the FIFO data "level" increases to the programmed level. Please refer to the Cypress data sheet and the code example in the manual.

## BIS_STAT1

[$0C] BISERIAL Status Port [read only]

| Data Bit | Status | |
|---|---|---|
| 15 | rx_enable | |
| 14 | no_data | |
| 13 | data_dir | |
| 12 | parity error | |
| 11 | count error | |
| 10 | message count error | |
| 9 | cmd error | |
| 8 | checksum error | |
| 7-0 | received message count | byte 1 value in last message received |

FIGURE 7 IP-BISERIAL-BA5 STATUS REG 1 BIT MAP

1. rx_enable, no_data, and data_dir are for hardware status purposes and can be ignored for software operation. The signals reflect the state of key points in the state machine processing.

2. Parity error set indicates that at least one byte from the last message had a parity error detected.

3. Count error set indicates that an improper number of bytes were received.

4. Message count error set indicates that the count has repeated itself.

5. CMD error set indicates that the message command does not fall within the legal values.

6. Checksum error set indicates that the checksum for the message does not match the checksum received at the end of the message.

The error codes 2-6 will be returned in the next transmitted message after being stored. The codes are captured at the end of each received message and held until another message is received.

The received message count is the Byte 1 value received and represents the number of the message received. The value will be sent out with the next transmitted message, or possibly already has been sent out but not yet over-written from the next received message.

The status is stored into a register referenced to the user selected reference clock and not necessarily the IP clock. If read after the Interrupt is set and before the next received message completes the data will be stable. If read while the message is completing and new data is being stored the results may not be stable.

**BISERIAL_RESET**

[$0E] BISERIAL Reset Port
The user can, by accessing this port, cause the BISERIAL to reset all major functions. The Control register, and FIFO's are cleared by a write to this port. Any data pattern can be written.

**BIS_TX_FIFO_0_W**

[$10] BISERIAL FIFO byte 0 write
The BISERIAL supports byte writes to the data FIFOs. By writing a byte to this address only byte_0 is affected. D15..8 are loaded at this address. **Word writes will load both bytes**.

**BIS_TX_FIFO_1_W**

[$11] BISERIAL FIFO byte 1 write
The BISERIAL supports byte writes to the TX FIFOs. By writing a byte to this address only byte_1 is affected. If a word is written to BIS_TX_FIFO_0_W this byte is loaded as well.

**BIS_TX_FIFO_0_R**

[$10] BISERIAL FIFO byte 0 read
A loop-back path is provided for the TX FIFOs to allow the host to read the data stored in the TX FIFOs. Both bytes are read back through the lower byte lane [D7..0]. Reading from this address fetches from the upper FIFO byte. Be sure to set the clock to not be divided [BR=0] and clk_hi set to low. *Once the data is read from the FIFO the data is no longer available for transmission*.

**BIS_TX_FIFO_1_R**

[$14] BISERIAL FIFO byte 1 read
A loop-back path is provided for the TX FIFOs to allow the host to read the data stored in the TX FIFOs. Both bytes are read back through the lower byte lane [D7..0]. Reading from this address fetches from the lower FIFO byte. Be sure to set the clock to not be divided [BR=0] and clk_hi set to low. *Once the data is read from the FIFO the data is no longer available for transmission*.

**BIS_RX_FIFO_0_W**

[$20] BISERIAL FIFO byte 0 write

A loop-back path is provided for the RX FIFOs to allow the host to load data into the RX FIFOs. Both bytes are written through the lower byte lane [D7..0]. Writing to this address loads the upper RX FIFO. This operation competes with and should not be performed during normal operation. Be sure to set the clock to not be divided [BR=0], clk_hi set to low, and testmode set to 1.

**BIS_RX_FIFO_1_W**

[$24] BISERIAL FIFO byte 1 write

A loop-back path is provided for the RX FIFOs to allow the host to load data into the RX FIFOs. Both bytes are written through the lower byte lane [D7..0]. Writing to this address loads the lower RX FIFO. This operation competes with and should not be performed during normal operation. Be sure to set the clock to not be divided [BR=0], clk_hi set to low, and testmode set to 1.

**BIS_RX_FIFO_0_R**

[$20] BISERIAL FIFO byte 0 read
The data stored into FIFO_0 can be accessed through this port. Byte and word accesses are available. A word access will fetch data from both FIFO 0 and FIFO 1.

**BIS_RX_FIFO_1_R**

[$21] BISERIAL FIFO byte 1 read
The data stored into FIFO 1 can be accessed through this port. Only byte wide accesses are supported.

## Interrupts

All IP Module interrupts are vectored. The vector from the IP-BISERIAL-BA5 comes from a vector register loaded as part of the initialization process. The vector register can be programmed to any 8 bit value. The default value is $FF which is sometimes not a valid user vector. The software is responsible for choosing a valid user vector.

The IP-BISERIAL-BA5 state machines generate an interrupt request when a transmission or reception is complete and the INTEN bits in the control registers are set. The transmission is considered complete when the last bit is transmitted. The interrupt is mapped to interrupt request 0. The CPU will respond by asserting INT. The hardware will automatically supply the appropriate interrupt vector and clear the request when accessed by the CPU. The source of the interrupt is obtained by reading BIS_STAT0. The status remains valid until the status register is read. The interrupt status is auto-cleared when the status register is accessed.

Some carrier boards pre-fetch data. If your carrier board pre-fetches the interrupt status, then the status may be cleared when the SW goes to look at it. If this is an issue then be careful with the order of reading the registers to prevent the pre-fetching function from affecting operation.

The interrupt level seen by the CPU is determined by the IP Carrier board being used. The master interrupt can be disabled or enabled through the BIS_CNTL0 register. The individual enables for TX and RX are controllable through BIS_CNTL0 and BIS_CNTL2. The enable operates before the interrupt holding latch, which stores the request for the CPU. Once the interrupt request is set, the way to clear the request is to reset the board, service the request, or disable the interrupt. Toggling the interrupt enable low will clear the interrupt, the interrupt enable can be set back to enabled immediately. TX_INT_EN enables and clears the TX interrupt and RX_INT_EN enables and clears the RX interrupt request.

If operating in a polled mode and making use of the interrupts for status then the master interrupt should be disabled and the Rx or TX or both enabled. When BIS_STAT0 shows an interrupt pending the appropriate FIFO action can take place and the enable toggled to remove the interrupt request then one extra read of the BIS_STAT0 to make sure that the interrupt request is cleared before starting the next transfer. Reading the BIS_STAT0 register does clear the interrupt status, but if the source of the status is still pending [interrupt request] then the status can become set again before the SW has a chance to clear it out. Hense the necessity of one extra read for clearing purposes.

Two additional interrupt sources are available on the -BA5 model.  The FIFO  [TX] Programmable Almost Empty flag [PAE] and [RX] Programmable Almost Full Flag [PAF] can be used to generate interrupts.  Please refer to the register definitions for the enable and interrupt status bit definitions.  The Flags are active low and level sensitive.  The hardware is designed with an edge detector circuit to find the transition from inactive to active.  The interrupt is created when the transition is detected.  The level is held until cleared.  The interrupt is cleared when the status register is read.  The PAF and PAE can be programmed to occur at user specified locations.  The details are in the Cypress data sheet.  Please also make use of our "model" code which we use to test the feature.

The PAE interrupt is useful in that large transfers can be accommodated with an interrupt programmed to let the host know that the FIFO is not empty but close and that a known amount of data can be added.  Similarly, the PAF interrupt can be used to let the host know that the FIFO is almost full and that a known amount of data can be burst out of the FIFO.

Neither PAF nor PAE are likely to be used with this version due to the small messages and protocol implemented.

Power on initialization will provide a cleared interrupt request, interrupts disabled, and interrupt vector of $FF.

## ID PROM

Every IP contains an ID PROM, whose size is at least 32 x 8 bits. The ID PROM aids in software auto configuration and configuration management. The user's software, or a supplied driver, may verify that the device it expects is actually installed at the location it expects, and is nominally functional. The ID PROM contains the manufacturing revision level of the IP. If a driver requires that a particular revision to be present, it may check for it directly.

The location of the ID PROM in the host's address space is dependent on which carrier is used. Normally the ID PROM space is directly above the IPs I/O space, or at IP-base + $80.

Standard data in the ID PROM on the IP-BISERIAL-BA5 is shown in the figure below. For more information on IP ID PROMs refer to the IP Module Logic Interface Specification, available from Dynamic Engineering.

Each of the modifications to the IP-BiSerial-IO board will be recorded with a new code in the DRIVER ID location.  -BA5 is set to '05'.

| Address | Data | |
|---|---|---|
| 01 | ASCII "I" | ($49) |
| 03 | ASCII "P" | ($50) |
| 05 | ASCII "A" | ($41) |
| 07 | ASCII "H" | ($48) |
| 09 | Manufacturer ID | ($1E) |
| 0B | Model Number | ($01) |
| 0D | Revision | ($A0) |
| 0F | reserved | ($00) |
| 11 | Driver ID, low byte | ($05) |
| 13 | Driver ID, high byte | ($00) |
| 15 | No of extra bytes used | ($0C) |
| 17 | CRC | ($CE) |

FIGURE 8 IP-BISERIAL-BA5 ID PROM

## IP Module Logic Interface Pin Assignment

The figure below gives the pin assignments for the IP Module Logic Interface on the IP-BISERIAL-BA5. Pins marked n/c below are defined by the specification, but not used on the IP-BISERIAL-BA5. Also see the User Manual for your carrier board for more information.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| GND | | GND | | 1 | | 26 | |
| | CLK | | +5V | | 2 | | 27 |
| Reset* | | R/W* | | 3 | | 28 | |
| | D0 | | IDSEL* | | 4 | | 29 |
| D1 | | n/c | | 5 | | 30 | |
| | D2 | | MEMSEL* | | 6 | | 31 |
| D3 | | n/c | | 7 | | 32 | |
| | D4 | | IntSel* | | 8 | | 33 |
| D5 | | n/c | | 9 | | 34 | |
| | D6 | | IOSel* | | 10 | | 35 |
| D7 | | n/c | | 11 | | 36 | |
| | D8 | | A1 | | 12 | | 37 |
| D9 | | n/c | | 13 | | 38 | |
| | D10 | | A2 | | 14 | | 39 |
| D11 | | n/c | | 15 | | 40 | |
| | D12 | | A3 | | 16 | | 41 |
| D13 | | IntReq0* | | 17 | | 42 | |
| | D14 | | A4 | | 18 | | 43 |
| D15 | | n/c | | 19 | | 44 | |
| | BS0* | | A5 | | 20 | | 45 |
| BS1* | | n/c | | 21 | | 46 | |
| | n/c | | n/c | | 22 | | 47 |
| n/c | | Ack* | | 23 | | 48 | |
| | +5V | | n/c | | 24 | | 49 |
| GND | | GND | | 25 | | 50 | |

NOTE 1: The no-connect signals above are defined by the IP Module Logic Interface Specification, but not used by this IP. See the Specification for more information.

NOTE 2: The layout of the pin numbers in this table corresponds to the physical placement of pins on the IP connector. Thus this table may be used to easily locate the physical pin corresponding to a desired signal. Pin 1 is marked with a square pad on the IP Module.

FIGURE 9 IP-BISERIAL-BA5 LOGIC INTERFACE

## IP Module IO Interface Pin Assignment

The figure below gives the pin assignments for the IP Module IO Interface on the IP-BISERIAL-BA5. Pins marked. Also see the User Manual for your carrier board for more information.

| | | | |
|---|---|---|---|
| GND | Spare+ | 1 | 26 |
| Spare+ | Spare- | 2 | 27 |
| Spare- | GND | 3 | 28 |
| GND | unused | 4 | 29 |
| Spare+ | GND | 5 | 30 |
| Spare- | unused | 6 | 31 |
| GND | GND | 7 | 32 |
| TX_DATA+ | unused | 8 | 33 |
| TX_DATA- | GND | 9 | 34 |
| GND | unused | 10 | 35 |
| TX_CLK+ | GND | 11 | 36 |
| TX_CLK- | unused | 12 | 37 |
| GND | GND | 13 | 38 |
| spare+ | GND | 14 | 39 |
| spare- | unused | 15 | 40 |
| GND | GND | 16 | 41 |
| CLKIN+ | GND | 17 | 42 |
| CLKIN- | unused | 18 | 43 |
| GND | GND | 19 | 44 |
| DATA_IO+ | GND | 20 | 45 |
| DATA_IO- | unused | 21 | 46 |
| GND | GND | 22 | 47 |
| Spare+ | GND | 23 | 48 |
| Spare- | unused | 24 | 49 |
| GND | GND | 25 | 50 |

NOTE 1: The layout of the pin numbers in this table corresponds to the physical placement of pins on the IP connector. Thus this table may be used to easily locate the physical pin corresponding to a desired signal. Pin 1 is marked with a square pad on the IP Module.

FIGURE 10 IP-BISERIAL-BA5 IO INTERFACE

.

## Applications Guide

### Interfacing

Some general interfacing guidelines are presented below. Do not hesitate to contact the factory if you need more assistance.

**Watch the system grounds**. All electrically connected equipment should have a fail safe common ground that is large enough to handle all current loads without affecting noise immunity. Power supplies and power consuming loads should all have their own ground wires back to a common point.

**Power all system power supplies from one switch.** Connecting external voltage to the IP-BISERIAL-BA5 when it is not powered can damage it, as well as the rest of the host system. This problem may be avoided by turning all power supplies on and off at the same time. Alternatively, the use of OPTO-22 isolation panels is recommended.

**Keep cables short**. Flat cables, even with alternate ground lines, are not suitable for long distances. IP-BISERIAL-BA5 does not contain special input protection.

**We provide the components. You provide the system**. Safety and reliability can be achieved only by careful planning and practice. Inputs can be damaged by static discharge, by applying voltage less than ground or more than +5 volts with the IP powered. With the IP unpowered, driven input voltages should be kept within .7 volts of ground potential.

**Terminal Block**. We offer a high quality 50 screw terminal block that directly connects to the flat cable. The terminal block mounts on standard DIN rails. [ http://www.dyneng.com/HDRterm50.html ]

Many flat cable interface products are available from third party vendors to assist you in your system integration and debugging. These include connectors, cables, test points, 'Y's, 50 pin in-line switches, breakout boxes, etc.

## Construction and Reliability

IP Modules were conceived and engineered for rugged industrial environments. The IP-BISERIAL-BA5 is constructed out of 0.062 inch thick FR4 material.

Through hole and surface mounting of components are used. IC sockets use gold plated screw machine pins. High insertion and removal forces are required, which assists in the retention of components. If the application requires unusually high reliability or is in an environment subject to high vibration, the user may solder the corner pins of each socketed IC into the socket, using a grounded soldering iron.

The IP Module connectors are keyed and shrouded with Gold plated pins on both plugs and receptacles. They are rated at 1 Amp per pin, 200 insertion cycles minimum. These connectors make consistent, correct insertion easy and reliable.

The IP is secured against the carrier with four metric M2 stainless steel screws. The heads of the screws are countersunk into the IP. The four screws provide significant protection against shock, vibration, and incomplete insertion. For most applications, they are not required.

The IP Module provides a low temperature coefficient of 0.89 W/$^{o}$C for uniform heat. This is based upon the temperature coefficient of the base FR4 material of 0.31 W/m-$^{o}$C, and taking into account the thickness and area of the IP. The coefficient means that if 0.89 Watts are applied uniformly on the component side, then the temperature difference between the component side and solder side is one degree Celsius.

### Thermal Considerations

The BISERIAL design consists of CMOS circuits. The power dissipation due to internal circuitry is very low. It is possible to create a higher power dissipation with the externally connected logic. If more than one a Watt is required to be dissipated due to external loading then forced air cooling is recommended. With the one degree differential temperature to the solder side of the board external cooling is easily accomplished.

## Warranty and Repair

Dynamic Engineering warrants this product to be free from defects in workmanship and materials under normal use and service and in its original, unmodified condition, for a period of one year from the time of purchase. If the product is found to be defective within the terms of this warranty, Dynamic Engineering's sole responsibility shall be to repair, or at Dynamic Engineering's sole option to replace, the defective product. The product must be returned by the original customer, insured, and shipped prepaid to Dynamic Engineering.  All replaced products become the sole property of Dynamic Engineering.

Dynamic Engineering's warranty of and liability for defective products is limited to that set forth herein. Dynamic Engineering disclaims and excludes all other product warranties and product liability, expressed or implied, including but not limited to any implied warranties of merchandisability or fitness for a particular purpose or use, liability for negligence in manufacture or shipment of product, liability for injury to persons or property, or for any incidental or consequential damages.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

## Service Policy

Before returning a product for repair, verify as well as possible that the suspected unit is at fault. Then call the Customer Service Department for a RETURN MATERIAL AUTHORIZATION (RMA) number. Carefully package the unit, in the original shipping carton if this is available, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. Dynamic Engineering will not be responsible for damages due to improper packaging of returned items. For service on Dynamic Engineering Products not purchased directly from Dynamic Engineering contact your reseller. Products returned to Dynamic Engineering for repair by other than the original customer will be treated as out-of-warranty.

### Out of Warranty Repairs

Out of warranty repairs will be billed on a material and labor basis. The current minimum repair charge is $100. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the quantity one list price for that unit. Return transportation and insurance will be billed as part of the repair and is in addition to the minimum charge.

## For Service Contact:

Customer Service Department
Dynamic Engineering
435 Park Dr.
Ben Lomond, CA 95005
831-336-8891
831-336-3840 fax
e-mail support@dyneng.com

## Specifications

| | |
|---|---|
| Logic Interface: | IP Module Logic Interface |
| Serial Interface: | RS-485 Data bidirectional, clock input, plus data output for loop-back |
| TX CLK rates generated: | 8 MHz base rates, with multiple divisors, or external reference |
| Software Interface: | Control Registers, ID PROM, Vector Register, Status Ports, FIFO |
| Initialization: | Hardware Reset forces all registers to 0.<br>Software Reset Command resets the control register, and FIFO's. |
| Access Modes: | Word or byte I/O Space (see memory map)<br>Word in ID Space<br>Vectored interrupt |
| Access Time: | back-to-back cycles in 500ns (8Mhz.) or 125 nS (32 Mhz.) to/from FIFO |
| Wait States: | 1 to ID space, 3 to IO or INT space |
| Interrupt: | Tx interrupt at end of transmission<br>Rx interrupt at end of transmission or end of terminal count<br>PAF Interrupt when RX FIFO is Almost Full<br>PAE Interrupt when TX FIFO is Almost Empty |
| DMA: | No Logic Interface DMA Support implemented. Memory Space maps into the FIFOs to allow auto-incrementing accesses |
| Onboard Options: | All Options are Software Programmable |
| Interface Options: | 50 pin flat cable<br>50 screw terminal block interface [HDRterm50]<br>User cable |
| Dimensions: | Standard Single IP Module. 1.8 x 3.9 x 0.344 (max.) inches |
| Construction: | FR4 Multi-Layer Printed Circuit, Through Hole and Surface Mount Components. Programmable parts are socketed. |
| Temperature Coefficient: | 0.89 W/$^{o}$C for uniform heat across IP |
| Power: | Max. **220** mA @ 5V |

## Order Information

| | |
|---|---|
| IP-BISERIAL-BA5 | IP Module with 1 Tx and 1 Rx serial channel, Programmable data rates Standard protocol support, RS-485 drivers and receivers 16 bit IP interface |
| Tools for IP-BISERIAL-BA5 | IP-Debug-Bus - IP Bus interface extender with testpoints, isolated power and quickswitch technology to allow hot swapping of IPs or power cycling without powering down the host. http://www.dyneng.com/ipdbgbus.html |
| | IP-Debug-IO II - IndustryPack IO connector breakout with testpoints, ribbon cable headers, and locations for user circuits. http://www.dyneng.com/ipdbgio.html |
| | HDRterm50 - Ribbon cable compatible 50 pin header to 50 screw terminal header. Comes with DIN rail mounting capability. http://www.dyneng.com/HDRterm50.html |
| | PCI3IP - 1/2 length PCI card with 3 IP slots. http://www.dyneng.com/pci_3_ip.html |

**All information provided is Copyright Dynamic Engineering**

# Programming Reference

## Programmable Almost Empty Flag Example

```
/****************************************************
 TX FIFO Test adapted for programmable flag tests
 slot a
 board 0
 check status and read - write patterns

 *  kvl 12-28-97
 *  updated for spartan implementation 5-25-99 KVL
 *  updated for flag tests 7/6/99
 *  Dynamic Engineering
 *  Copyright 1997,1999 Dynamic Engineering All Rights Reserved.
 *  435 Park Dr.
 *  Ben Lomond, Ca. 95005
 *  408-336-8891
 *  engineering@dyneng.com
 *  http://www.dyneng.com
 ****************************************************/
/*  at this point AE has been checked to work with the default value          */
/* check again with a new value                                                */
/*  reset the FIFOs and place into the two enable mode.                        */
/*   program the PAE flag to trigger at 16 instead of 7                        */
/*    load fifo to default check point and test then to new check point        */
/*     and test again                                                          */
/* set LD control low with reset hi
     then set reset low with LD low
      then set reset hi  to clear the FIFO with the WE control in the dual mode */

data_out = 0x0058; //clr fifo hi, LD lo for TX, testmode
putword(base1,(pci40_ioa_st + bisS_cntl2),hWinRT,iWinRTlength,data_out);
data_out = 0x0048; //clr FIFO lo, ld lo, testmode
putword(base1,(pci40_ioa_st + bisS_cntl2),hWinRT,iWinRTlength,data_out);
data_out = 0x0058; //clr FIFO hi, ld lo, testmode
putword(base1,(pci40_ioa_st + bisS_cntl2),hWinRT,iWinRTlength,data_out);

/*  should now be reset into the dual WE mode */

// check status         to see if empty
data_in = 0x4070 & getword(base1,(pci40_ioa_st + bisS_stat0),hWinRT,iWinRTlength);
//read pattern back and check if correct empty, almost empty and not full
if(0x40 != (short)data_in)
   {
  error = 1;
  testdata[1] = 0x40;
  testdata[2] = data_in;
  putword(base1,(pci40_ioa_st + bisS_cntl0),hWinRT,iWinRTlength,cntl0_copy);
  putword(base1,(pci40_ioa_st + bisS_cntl2),hWinRT,iWinRTlength,cntl2_copy);
  WinRTCloseDevice(hWinRT);
  return(error);
  }

/*  now program the PAE to be at the new value */
/* first write with LD low = LSB of PAE, second = upper 5 bits of PAE
     third = LSB of PAF and 4th = upper 5 bits of PAF */

/* flag comes from  upper byte ...lower data does not matter */
data_out = 0x1000;
putword(base1,(pci40_ioa_st + bisS_ftx_0_w),hWinRT,iWinRTlength,data_out);

/* read from output port with LD low to read back new value */
data_in = 0x00ff & getword(base1,(pci40_ioa_st + bisS_ftx_0_r),hWinRT,iWinRTlength); //read
pattern back and check if correct
/* put into data mode by re-writing with reset hi and LD hi. */

data_out = 0x0078; //clr FIFO hi, ld lo, testmode
putword(base1,(pci40_ioa_st + bisS_cntl2),hWinRT,iWinRTlength,data_out);


// Load both bytes
data_out = 0x00;
for(j=0; j<8; j++) {           //check that AE flag does not change to not AE, still not full
putword(base1,(pci40_ioa_st + bisS_ftx_0_w),hWinRT,iWinRTlength,data_out);
data_out = 0xffff & (data_out + 1);
}

// check status         check if upper & lower bytes are not empty
data_in = 0x4070 & getword(base1,(pci40_ioa_st + bisS_stat0),hWinRT,iWinRTlength);
//read pattern back and check if correct not empty, not almost empty and not full flags
if(0x0070 != (short)data_in)               // should not have PAE flag cleared yet
   {
  error = 2;
  testdata[1] = 0x0070;
  testdata[2] = data_in;
  putword(base1,(pci40_ioa_st + bisS_cntl0),hWinRT,iWinRTlength,cntl0_copy);
  putword(base1,(pci40_ioa_st + bisS_cntl2),hWinRT,iWinRTlength,cntl2_copy);
  WinRTCloseDevice(hWinRT);
  return(error);
  }
// now low with more data and see if flag is set at the proper time
data_out = 0x00;
for(j=0; j<0xa; j++) {       //check that AE flag changes to not AE, still not full
  putword(base1,(pci40_ioa_st + bisS_ftx_0_w),hWinRT,iWinRTlength,data_out);
// fill FIFO with pattern
  data_out = 0xffff & (data_out + 1);
  }

data_in = 0x4070 & getword(base1,(pci40_ioa_st + bisS_stat0),hWinRT,iWinRTlength);
//read pattern back and check if correct not empty, not almost empty and not full flags
if(0x4070 != (short)data_in)               // should have PAE flag cleared
  {
```

```
error = 2;
testdata[1] = 0x4070;
testdata[2] = data_in;
putword(base1,(pci40_ioa_st + bisS_cntl0),hWinRT,iWinRTlength,cntl0_copy);
putword(base1,(pci40_ioa_st + bisS_cntl2),hWinRT,iWinRTlength,cntl2_copy);
WinRTCloseDevice(hWinRT);
return(error);
}
```