# CHAPTER 11
Z80185 BIDIRECTIONAL CENTRONICS P1284 CONTROLLER

## 11.1 INTRODUCTION

The Centronics P1284 Controller can operate in either the Host or Peripheral role in Compatibility mode (host to printer), Nibble or Byte mode (printer to host), and ECP mode (bidirectional). It provides no hardware support for the EPP mode, although it may be possible to implement this mode by software.

Nine control signals have dedicated hardware pins, and have ±12 mA drive (P1284 Level 2) capability as does the 8-bit data port PIA27-20. **Note:** Signal names listed below are those for the original Compatible mode. The names shown in parentheses represent the same signal, but in a more recent mode. The Z80185 does not include hardware support for the P1284 EPP mode.

The following signals are outputs in a Peripheral mode, inputs in a Host mode:

- Busy (PtrBusy, PeriphAck)
- nAck (PtrClk, PeriphClk)
- PError (AckDataReq, nAckReverse)
- nFault (nDataAvail, nPeriphRequest)
- Select (Xflag)

The following signals are inputs in a Peripheral mode, outputs in a Host mode:

- nStrobe (HostClk)
- nAutoFd (HostBusy, HostAck)
- nSelectIn (P1284Active)
- nInit (nReverseRequest)

Note that, because the Host/Peripheral mode is fully controlled by software, a Z80185-based product can operate as a Host in one system, or as a Peripheral in another, without any change to the hardware. A Z80185-based product could even act as a Host at one time and a Peripheral at another time within the same system, if there is a mechanism to control such alternate use.

In general, the interface architecture automates operations that are seen as performance-critical, while leaving less frequent operations to software control. To achieve top performance, software should assign a DMA channel to the current direction of data flow.

**Note:** The IEEE 1284 Interface should be used with the/IOC bit (bit D5) in the OMCR set to 0. The setting of this bit primarily affects RLE expansion in peripheral ECP forward and host ECP reverse modes.

## 11.2 BIDIRECTIONAL CENTRONICS REGISTERS

Reading the Parallel Controls (PARC) register allows software to sense the state of the input signals per the current mode, plus two or three status flags:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Busy | PError | Select | nFault | nAck | IllOp | DREQ | Idle |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-1. Reading PARC in a Host Mode**
(I/O Address %DA)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| nAutoFd | nStrobe | nSlctIn | nInit | | IllOp | DREQ | Idle |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-2. Reading PARC in a Peripheral Mode**
(I/O Address %DA)

The controller sets IllOp (Illegal Operation) when it detects an error in the protocol, for example, if it’s in Peripheral mode and it detects that the host has driven P1284Active (nSelectIn) Low at a time that mandates an immediate Abort, that is, outside one of the “windows” in which this event indicates an organized disengagement. If “status interrupts” are enabled, such an interrupt is always requested when IllOp is set. Writing PARM with NewMode=1 clears IllOp.

DREQ is the Request presented to the DMA channels, which may or may not be programmed to service this request. If not, an interrupt can be enabled when DREQ is set.

Writing to PARC allows the software to set and clear the output signals per the current mode:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1=drive nAutoFd High | 1=drive nStrobe High | 1=drive nSelctIn High | 1=drive nInit High | 1=drive nAutoFd Low | 1=drive nStrobe Low | 1=drive nSelctIn Low | 1=drive nInit Low |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-3. Writing to PARC in a Host Mode**
(I/O Address %DA)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1=drive Busy High | 1=drive PError High | 1=drive Select High | 1=drive nFault High | 1=drive Busy Low | 1=drive PError Low | 1=drive Select Low | 1=drive nFault Low |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-4. Writing to PARC in a Peripheral Mode**
(I/O Address %DA)

Because there are five outputs in a Peripheral mode, another register, called PARC2, allows software to change the nAck line, rather than the Select line:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1=drive Busy High | 1=drive PError High | 1=drive nAck High | 1=drive nFault High | 1=drive Busy Low | 1=drive PError Low | 1=drive nAck Low | 1=drive nFault Low |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-5. Writing to PARC2 in a Peripheral Mode**
(I/O Address %DB)

The Parallel mode register (PARM) includes the basic mode control of the controller:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| NewMode | IdleIE | StatIE | DREQIE | Mode | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-6. PARM** (I/O Address %D9)

**NewMode = 1** reinitializes the state machine to the initial state for the mode called out by MODE. Never change MODE without writing a 1 in this bit.

**IdleIE = 1** enables interrupts when the controller sets the Idle flag. When software uses a DMA channel to provide data to the P1284 controller, it can be expected that the channel will do so in a timely manner, and thus, that an Idle condition signifies that the channel has finished transferring the block. (Software can also enable an interrupt from the DMA channel, but on the transmit side, such interrupts are not well-synchronized to events on the P1284 controller.) Conversely, if software provides data, Idle may not be grounds for an interrupt.

Some modes set the Idle flag when they are entered. However, such a setting of Idle never requests an interrupt.

**StatIE = 1** enables "status" interrupts that are described separately for each mode.

**DREQIE = 1** enables interrupts when the controller sets DREQ, except that in those modes that set DREQ when they are entered, such setting doesn't request an interrupt.

**Table 11-1. Bidirectional Centronics Mode Selection**

| MODE | |
|---|---|
| 0000 | Non-P1284 mode |
| 0001 | Peripheral Compatible/Negotiation mode |
| 0010 | Peripheral Nibble mode |
| 0011 | Peripheral Byte mode |
| 0100 | Peripheral ECP Reverse mode |
| 0101 | Peripheral Inactive mode |
| 0110 | Peripheral ECP Forward mode with software RLE handling |
| 0111 | Peripheral ECP Forward mode with hardware RLE expansion |
| 1000 | Host Negotiation mode |
| 1001 | Host Compatible mode |
| 1010 | Host Nibble mode |
| 1011 | Host Byte mode |
| 1100 | Host ECP Forward mode |
| 1101 | Host Reserved mode |
| 1110 | Host ECP Reverse mode with software RLE handling |
| 1111 | Host ECP Reverse mode with hardware RLE expansion |

A second output register has been added for PIA27-20. Writing to either the Z80181-compatible PIA 2 Data Register (address E3) or the new Alternate PIA 2 Data Register (address EE) writes to the Output Holding Register (OHR). When the PIA27-20 pins are outputs, the outputs of the OHR are the inputs to the second register, which is called the I/O register (IOR), these outputs drive the PIA27-20 pins. When the pins are inputs, they are the inputs to the IOR, which can be read from the PIA 2 Data Register (address E3).

In non-P1284 mode, Host Negotiation mode, Reserved Modes, and in Peripheral Compatible/Negotiation mode when the host drives nSelectIn (P1284Active) High to select negotiation, the direction of the PIA27-20 pins are controlled by the PIA 2 Data Direction register, as on the Z80181. Also in these modes the IOR is loaded on every PHI clock, so that operation is virtually identical to the Z80181. In other modes the controller controls the direction of PIA27-20 and when the IOR is loaded.

A Time Constant Register PART must be loaded by software with the smallest number of PHI clocks that equals or exceeds the "critical time" for the mode selected in PARM. The critical time is 750 ns for Host Compatible mode, 500 ns for most other modes, and the time necessary to indicate DMA completion in Host ECP Forward and Peripheral ECP Reverse modes.

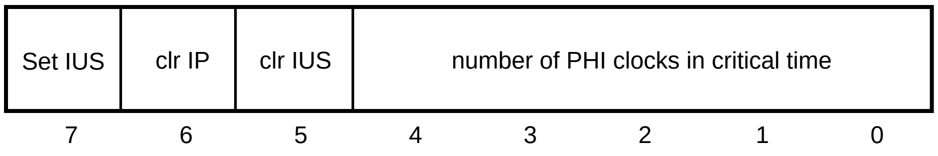

**Figure 11-7. PART Write** (I/O Address %DC)

Reading PART yields the status of the IP and IUS bits, which are described in the Bidirectional Centronics Interface section:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| IUS | IP | 0 | number of PHI clocks in critical time | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-8. PART Read** (I/O Address %DC)

The Vector Register PARV must be loaded by software with the interrupt vector to be used for interrupts from this controller.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Interrupt Vector | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 11-9. PARV** (I/O Address %DD)

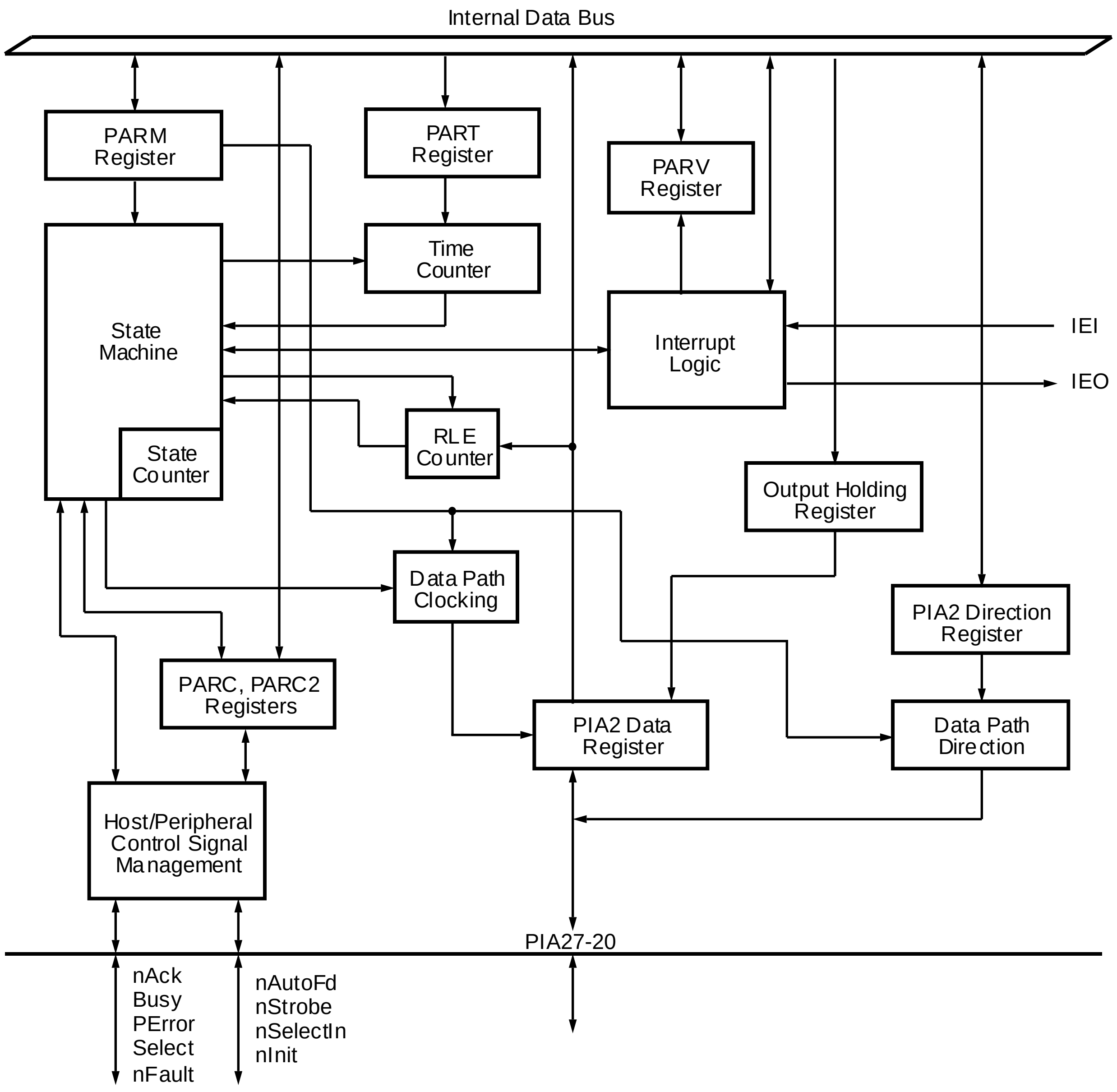

**Figure 11-10. Bidirectional Centronics P1284 Controller Functional Block Diagram**

## 11.2.1 Interrupts

As in other Zilog peripherals, the controller includes an interrupt pending bit (IP), and an interrupt under service bit (IUS). The controller is part of an on-chip interrupt acknowledge daisy-chain that extends from the IEI pin, through the EMSCC, CTC, and this controller in a programmable priority order, and from the lowest-priority of these devices to the IEO pin. The interrupt request from the controller is logically ORed with /INT0 and other on-chip interrupt requests to the processor.

The controller sets its IP bit whenever any of three conditions occurs:

PARM4 is 1, and the controller sets the DREQ bit. This does not include when the controller forces the DREQ bit to 1, when software first places the controller in Peripheral Nibble, Peripheral Byte, Peripheral ECP Reverse, Host Compatible, or Host ECP Forward mode.

PARM5 is 1, and a mode-dependent “status interrupt” condition occurs. The following sections describe the status interrupt conditions (if any) for each mode.

PARM6 is 1, and the controller sets the Idle bit, except when the controller forces the Idle bit to 1, when software first places the controller in Peripheral Nibble, Peripheral Byte, Peripheral ECP Reverse, Host Compatible, or Host ECP Forward mode. The following sections describe when Idle is set in each mode.

Once IP is set, it remains set until software writes a 1 to PART6.

The controller will begin requesting an interrupt of the processor whenever IP is set, its IEI signal from the on-chip daisy-chain is High/true, and its IUS bit is 0. Once it starts requesting an interrupt, the controller will continue to do so until /IORQ goes Low in an interrupt-acknowledge cycle, or IP is 0, or IUS is 1.

The controller drives its IEO output High, if its IEI input is High, and its IP and IUS bits are both 0. A Z80 interrupt acknowledge cycle is signalled by /M1 going Low, followed by /IORQ going Low. The controller, and all other devices in the daisy-chain, freeze the contribution of their IP bits to their IEO outputs while /M1 is Low, which prevents new events from affecting the daisy-chain. By the time/IORQ goes Low, one and only one device will have its IEI pin High and its IEO pin Low — this device responds to the interrupt by providing an interrupt vector, and setting its IUS bit. This controller also clears its IP bit when it responds to an interrupt acknowledge cycle.

The interrupt service routine, that is initiated when the interrupt vector value identifies an interrupt from this controller, should save the processor context and then proceed as follows:

If the ISR does not allow nested interrupts, it can clear the IP and IUS bits by writing hex 60, plus the “critical time” value to the PART, then read the status from PARC and proceed based on that status. Near the end of the ISR it should re-enable processor interrupts.

If the ISR allows nested interrupts, it can re-enable processor interrupts, clear IP by writing hex 40 plus the “critical time” value to the PART, and then read the status from PARC and proceed based on that status. At the end of the ISR it should clear IUS to allow further interrupts from this controller and devices lower on the daisy-chain, by writing hex 20 plus the “critical time” value to the PART.

## 11.2.2 Operating Modes

The remainder of this section describes the operation of the various PARM register modes that can be selected.

## 11.2.3 Non-P1284 Mode

The Z80185 defaults to this mode after a Reset, and this mode is compatible with the use of PIA27-20 on the Z80181. The directions of PIA27-20 can be controlled individually by writing to register E2, as on the Z80181. The state of outputs among PIA27-20 can be set by writing to register E3, and the state of all eight pins can be sensed by reading register E3. The Busy, nAck, PError, nFault, and Select pins are tri-stated in this mode, while nStrobe, nAutoFd, nSelectIn, and nInit are inputs. There are no status interrupts in this mode.

## 11.2.4 Peripheral Inactive Mode

This mode operates identically to Non-P1284 mode as described above, except that the Busy, nAck, PError, nFault, and Select pins are outputs that can be controlled via the PARC and PARC2 registers, and status interrupts can occur in response to any edge on nAutoFd, nStrobe, nSelectIn, or nInit. This mode differs from Peripheral Compatibility/Negotiation mode with nSelectIn (P1284 Active) High, only in that the controller will not operate in Compatibility mode if nSelectIn goes Low.

### 11.2.5 Host Compatible Mode

1. Setting this mode configures PIA27-20 as outputs regardless of the contents of register E2. When entering this mode, the controller sets the Idle and DREQ bits, but these settings do not request an interrupt.

2. If software, or a DMA channel, writes eight bits to the Output Holding Register (OHR) when Idle is set, the controller transfers the byte to the Input/Output Register and negates DREQ only momentarily, so as to request another byte from software or the DMA channel.

3. In this mode, the nAutoFd line is not under control of the PARC register, but rather under control of which register the software uses to write data to the OHR. Each time the controller transfers a byte from the OHR to the Input/Output Register, it sets nAutoFd High if the byte was written to address E3, and Low if the byte was written to the "alternate" address EE. In a DMA application all of the bytes transferred from one output buffer will have the same state of nAutoFd, but this state can be changed from one buffer to the next by changing theI/O address used by the DMA channel. In non-DMA applications software can set the state of nAutoFd for each character, by writing data to the two different register addresses.

4. When a data byte has been valid on PIA27-20 for 750 ns (as controlled by the PART register), and the Busy and PError lines are Low and the Select, nAck, and nFault lines are High, the controller drives nStrobe Low. After the controller has held nStrobe Low for 750 ns it drives nStrobe back to High. Then it waits for 750 ns of data hold time to elapse. If software or a DMA channel has written another byte to the Output Holding Register (thus clearing DREQ) by the time this wait is satisfied, the controller transfers the byte from the Output Holding Register to the Input/Output Register, sets DREQ again, and returns to the event sequence at the start of this paragraph. Otherwise, it sets Idle and returns to the event sequence at the start of paragraph #2.

Status interrupts in this mode include rising and falling edges on PError, nFault, and Select.

### 11.2.6 Host Negotiation Mode

Setting this mode puts PIA27-20 under control of registers E2 and E3, as on the Z80181.

Software has complete control of the controller, and can either revert to Host Compatibility mode, or set one of the following Host modes, depending on how the peripheral responds to the Negotiation value(s).

Status interrupts in this mode include rising and falling edges on PtrClk (nAck), nAckReverse (PError), and nPeriphRequest (nFault). nFault is not used during actual P1284 negotiation, but is included because these events are significant during Byte and ECP mode idle times.

### 11.2.7 Host Reserved Mode

This mode differs from Host Negotiation mode only in that there are no status interrupts in this mode.

### 11.2.8 Peripheral Compatible/Negotiation Mode

In this mode, if P1284Active (nSelectIn) is Low, the controller sets PIA27-20 as inputs, regardless of the contents of register E2; when P1284Active (nSelectIn) is High, PIA27-20 are under the control of registers E2 and E3. On entry to this mode, the controller sets the Idle bit, if DREQ is set from a previous mode.

If, in this mode, nStrobe goes (is) Low, P1284Active (nSelectIn) is Low, and DREQ is 0, indicating that any previous data has been taken by the processor or DMA channel, the controller captures the data on PIA27-20 into the Input/Output Register, sets DREQ to notify software or the DMA channel to take the byte, drives the Busy line High, and one PHI clock later drives nAck Low. When at least 500 ns (as controlled by the PART register) have elapsed, the controller drives nAck back to High. One PHI clock later, if the CPU or DMA has taken the data and thus cleared DREQ, the controller drives Busy back to Low, otherwise it sets Idle.

Select, PError and nFault are under software control in this mode, and nAutoFd can be sensed by software, but has no other effect on operation.

In this mode, software should monitor for the condition P1284Active (nSelectIn) High, and nAutoFd Low simultaneously. If software detects this state, it should participate in a Negotiation process. Software should read the value on PIA27-20 and set PError, nFault, XFlag, and nAck as appropriate for the data value. As long as P1284Active (nSelectIn) remains High in this mode, software is in complete control of the controller. After the host has driven nStrobe Low and then High again for an acceptable value, software should reprogram the MODE field to the appropriate one of the following Peripheral modes.

Status interrupts in this mode include rising and falling edges on P1284Active (nSelectIn) and nInit, and rising and falling edges on HostBusy (nAutoFd) and HostClk (nStrobe) while P1284Active (nSelectIn) is High.

### 11.2.9 Host Nibble Mode

1. If, during Host Negotiation mode, software has placed the value 00 or 04 on the data lines, and received a positive response on Xflag (Select) and a Low on nDataAvail (nFault) at a rising edge of PtrClk (nAck), then after optionally programming a DMA channel to store data, it should set this mode.

2. For each byte in this mode, the controller drives HostBusy (nAutoFd) Low and waits until DREQ is cleared, indicating that the CPU or DMA has taken any previous data, and the peripheral has driven PtrClk (nAck) Low. At this point it samples the other four status lines from the peripheral into the less-significant four bits of the Input/Output Register as follows:

**Table 11-2. Nibble Mode Bit Assignments**

| Signal | First Data Bit | Second Data Bit |
|---|---|---|
| Busy | 3 | 7 |
| PError | 2 | 6 |
| Select | 1 | 5 |
| nFault | 0 | 4 |

The controller then drives HostBusy (nAutoFd) back to High, and waits for the peripheral to drive PtrClk (nAck) back to High. Then it drives HostBusy (nAutoFd) back to Low and waits for the peripheral to drive PtrClk (nAck) Low. At this point it samples the four status lines from the peripheral into the most-significant four bits of the Input/Output Register, as shown above. Then it drives HostBusy (nAutoFd) back to High, sets the DREQ bit, and waits for the peripheral to drive PtrClk (nAck) back to High. When this occurs, if the peripheral is driving nDataAvail (nFault) Low, indicating more data is available, the controller then returns to the event sequence at the start of paragraph #2.

3. If nDataAvail (nFault) is High at a rising edge of nAck in this mode, indicating that the peripheral has no more data, the controller sets Idle and waits for software to program it back to Host Negotiation mode. Software can then select the next mode (reference IEEE P1284 specification).

If host software is programmed not to select all the data that a peripheral has available, it should first disable the DMA channel, if one is in use, then wait for DREQ to be 1 and PtrClk (nAck) to be High. If nDataAvail (nFault) is Low at this point, the controller will have already driven HostBusy (nAutoFd) Low to solicit the next byte. Software should then program the controller back to Host Negotiation mode, read the IOR to get the current byte, and take the next byte from the peripheral under software control. After the peripheral drives nAck High after the second nibble, software can drive P1284Active (nSelectIn) Low to tell the peripheral to leave Nibble mode.

There are no status interrupts in Host Nibble mode.

### 11.2.10 Peripheral Nibble Mode

1. Software shouldn't set this mode until there is reverse data available to send. In other words, it should implement the P1284 "reverse idle mode" via software in Peripheral Compatibility/Negotiation mode. After software has driven nDataAvail (nFault), AckDataReq (PError), and Xflag (Select) all Low to signify that data is available, then driven PtrClk (nAck) High after 500 ns, and if requested programmed a DMA channel to provide data to send, when it sees HostBusy (nAutoFd) Low to request data, software should set this mode.

Setting this mode sets DREQ and Idle, but these settings do not request an interrupt. The PIA27-20 pins remain configured for data input but are not used. Instead, four of the five control outputs are driven with the LS and MS four bits of the Input/Output Register, as shown in Table 11-2, while PtrClk (nAck) serves as a handshake/clock output. On entering this mode the hardware begins routing bits 3-0 of the IOR to these lines.

2. If software, or a DMA channel, writes a byte to the Output Holding Register when Idle is set, the controller immediately transfers the byte to the IOR and clears Idle, and negates DREQ only momentarily to request another byte from software or the DMA channel.

3. After data has been valid on the four control outputs for 500 ns (as controlled by the PART register), the controller drives the PtrClk (nAck) line Low. Then it waits for the host to drive the HostBusy (nAutoFd) line back to High, after which it drives PtrClk (nAck) back to High, switches the four control lines to bits 7-4 of the IOR, and begins waiting for the host to drive HostBusy (nAutoFd) back to Low. When bits 7-4 have been valid for 500 ns and the host has driven HostBusy (nAutoFd) Low, the controller drives PtrClk (nAck) Low again and begins waiting for the host to drive HostBusy (nAutoFd) High. When HostBusy (nAutoFd) has been driven High, the controller returns the four control outputs to the state set by software in PARC. At this point, if software or a DMA channel has not yet written another byte to the Output Holding Register (thus clearing DREQ), the controller sets Idle and waits for software to do so. If/when software or a DMA channel has written a new byte to the OHR, the controller transfers the byte to the IOR, sets DREQ, and clears Idle if it had been set. Then, when the control outputs have been valid for 500 ns, the controller drives PtrClk (nAck) to High. It then waits for the host to drive HostBusy (nAutoFd) back to Low, at which time it switches the four control lines back to bits 3-0 of the IOR and returns to the event sequence at the start of this paragraph.

If there is no more data to send, when the controller sets Idle, software should modify PARC to make nDataAvail (nFault) and AckDataReq (PError) High, and then change the mode to Peripheral Compatible/Negotiation. Then (after 500 ns) software should set PtrClk (nAck) back to High in PARC and enter Reverse Idle state.

Status interrupts in Peripheral Nibble mode include rising and falling edges on P1284Active (nSelectIn) and nInit. The controller sets the IllOp (Illegal Operation) bit if P1284Active (nSelectIn) goes Low in this mode, before it drives nAck High for the status states on the four control lines, or after the host drives HostBusy Low thereafter, in which case software should immediately enter Peripheral Compatibility/Negotiation mode. If P1284Active goes Low, but IllOp stays 0, indicating that the Host negated P1284Active in a legitimate manner, software should enter Peripheral Inactive mode for the duration of the "return to Compatibility mode", and then enter Peripheral Compatibility/Negotiation mode.

### 11.2.11 Host Byte Mode

1. When in Host Negotiation mode the software has presented the value hex 01 or 05 on PIA27-20, it has been acknowledged by the peripheral, and the peripheral has driven nDataAvail (nFault) and AckDataReq (PError) to Low to indicate data availability and then driven PtrClk (nAck) back to High, software should set this mode. This sets PIA27-20 as inputs regardless of the contents of register E2, and clears the Idle flag. The controller then waits 500 ns (as controlled by the PART register) before proceeding.

2. For each byte, the controller drives HostBusy (nAutoFd) Low to indicate readiness for a byte from the peripheral. Then it waits for PtrClk (nAck) to go Low, at which time it captures the state of PIA27-20 into the Input/Output Register; sets the DREQ bit to request software, or the DMA channel to take the byte, and drives HostBusy (nAutoFd) High and HostClk (nStrobe) Low. When software, or the DMA channel, has taken the byte (thus clearing DREQ) and the peripheral has driven PtrClk (nAck) back High, and at least 500 ns after driving HostClk (nStrobe) Low, the controller drives HostClk (nStrobe) back to High, and samples nDataAvail (nFault). If it is still Low, the controller returns to the event sequence at the start of this paragraph, otherwise it sets the Idle flag.

In response to Idle, software should enter Host Negotiation mode. Thereafter, it can set HostBusy (nAutoFd) Low, to enter Reverse Idle state, or enter Host Compatible mode (reference IEEE P1284 specification), or conduct a new negotiation.

If software is programmed not to accept all the data that a peripheral has available in this mode, it should first disable the DMA channel, if one is in use, and then wait for DREQ to be 1 and nAck to be 1. Then it should reprogram the controller back to Host Negotiation mode, read the last byte from the IOR, drive HostClk (nStrobe) back to High, and then drive P1284Active (nSelectIn) Low to instruct the peripheral to leave Byte mode.

There are no status interrupts in Host Byte mode.

## 11.2.12 Peripheral Byte Mode

1. Software should not set this mode until there is reverse data available to send — that is, it should implement the P1284 “reverse idle mode” via software in Peripheral Compatibility/Negotiation mode. The exact sequencing among PtrClk (nAck), nDataAvail (nFault), and AckDataReq (PError) differs according to whether this mode is entered directly from Negotiation or from reverse idle phase, and is controlled by software. But in either case, before software sets this mode, it should set nDataAvail (nFault) and AckDataReq (PError) to Low, then after 500 ns, set PtrClk (nAck) to High. When it detects that the host has driven HostBusy (nAutoFd) Low to request data, software should set this mode, which sets the DREQ and Idle flags.

2. In this mode, as long as P1284Active (nSelectIn) remains High, the controller drives PIA27-20 as outputs, regardless of the contents of register E2. When software, or a DMA channel, writes the first byte to the Output Holding Register, the controller immediately transfers the byte to the Input/Output Register, clears Idle but negates DREQ only momentarily, to request another byte from software, or the DMA channel.

3. After each byte is transferred to the IOR, the controller waits 500 ns data setup time (as controlled by the PART register) before driving PtrClk (nAck) Low, and thereafter waits for the host to drive HostBusy (nAutoFd) High. When this occurs, if software, or the DMA channel, has not written more data to the Output Holding Register, that is, if DREQ is still set, the controller sets the Idle flag and waits for software or the DMA channel to do so. If software, or the DMA channel, then writes data to the Output Holding Register, the controller clears DREQ and Idle. When there is data in the OHR and DREQ is 0, this guarantees that it is appropriate to keep nDataAvail (nFault), and AckDataReq (PError) Low to indicate that more data is available, and the controller drives PtrClk (nAck) back to High. The controller then waits for a rising edge on HostClk (nStrobe), and then for the host to drive HostBusy (nAutoFd) Low, at which time it transfers the byte from the OHR to the Output Register, sets DREQ, and then it returns to the event sequence at the start of this paragraph.

While this mode is in effect, software should monitor the interface for two conditions:

**Case 1:** Idle set and no more data to send, or

**Case 2:** P1284Active (nSelectIn) Low.

In Case #1, the software should write zero to register E3 to keep PIA27-20 outputs momentarily, and then set the mode back to Peripheral Compatibility, so that the interface is fully under software control, set nDataAvail (nFault) and AckDataReq (PError) High to signify no more data, wait 500 ns, and set PtrClk (nAck) back to High. When HostBusy goes back to Low, the software should set PIA27-20 back to inputs.

In Case #2, if a falling edge on P1284Active happens any time other than between a rising edge on HostClk (nStrobe), and the next falling edge on HostBusy (nAutoFd), the controller sets the IllOp bit to notify software that an immediate Abort is in order, in which case software should immediately enter Peripheral Compatibility/Negotiation Mode. If P1284Active goes Low, but IllOp is not set, meaning that the Host negated P1284Active in a “legal” manner, software should enter Peripheral Inactive Mode for the duration of the “return to Compatibility Mode”, and then enter Peripheral Compatibility/Negotiation Mode.

Status interrupts in Peripheral Byte Mode include rising and falling edges on P1284Active (nSelectIn) and nInit.

### 11.2.13 Host ECP Forward Mode

1. After a negotiation for ECP mode, "host" software should remain in Negotiation mode so that it has complete control of the interface, until one of two situations occurs. If software has data to send, it should optionally program the DMA channel to provide the data, and then set this mode. Alternatively, if software has no data to send and it detects that nPeriphRequest (nFault) has gone Low, indicating the peripheral is requesting reverse transfer, it should set PIA27-20 as inputs, wait 500 ns, drive nReverseRequest (nInit) to Low to indicate a reverse transfer, and then set Host ECP Reverse mode. In other words, software should handle all aspects of ECP mode, other than active data transfer sequences.

2. Setting this mode configures PIA27-20 as outputs regardless of the contents of register E2. On entry to this mode, the controller sets Idle and DREQ to request a byte from software or a DMA channel, but these settings do not cause an interrupt request.

3. If software, or a DMA channel, writes data to the Output Holding Register while the Input/Output Register is empty, the controller immediately transfers the byte to the IOR, clears Idle, and negates DREQ only momentarily, to request another byte.

4. In this mode, the alternate address for the Output Holding Register allows software to send a "channel address" or an RLE count value. Such bytes are typically written by software rather than a DMA channel. Writing to the alternate address loads the OHR and clears DREQ, like writing to the primary address, but clears a ninth bit that is set when software, or a DMA channel, writes to the primary address. A similar ninth bit is associated with the Input/Output Register, from which it drives the HostAck (nAutoFd) line.

5. As each nine bits arrive in the IOR and thus out onto PIA27-20 and HostAck (nAutoFd), the controller waits one PHI clock and then drives HostClk (nStrobe) to Low. It then waits for the peripheral to drive PeriphAck (Busy) to High, after which it drives HostClk (nStrobe) back to High. Then it waits for the peripheral to drive PeriphAck (Busy) back to Low. When this has happened, if software or a DMA channel has written a new byte to the Output Holding Register, and thus cleared DREQ, the controller transfers the byte to the IOR, sets DREQ again, and returns to the event sequence at the start of this paragraph. Otherwise, it returns to the event sequence at the start of paragraph #3. If software, or a DMA channel, does not provide a new byte for the time indicated in the PART register, the controller sets the Idle flag.

6. While this mode is in effect, software should monitor for the condition "Idle and no more data left to send", and/or nPeriphRequest (nFault) Low. Host software has complete freedom as to whether to honor the peripheral's reverse request on nFault while it has data to send. When there is no more data, software can set Host Negotiation mode to have full control of the interface, and if requested can drive P1284Active (nSelectIn) to Low in order to terminate ECP mode, or can set Host ECP Reverse mode, wait 500 ns, and drive nReverseRequest (nInit) to Low.

Status interrupts in Host ECP Forward mode include rising and falling edges on nPeriphRequest (nFault).

### 11.2.14 Peripheral ECP Forward Modes

1. After a negotiation for ECP mode, "peripheral" software should remain in Compatibility/Negotiation mode with P1284Active (nSelectIn) High, so that it has complete control of the interface, though when it detects the host drive HostAck (nAutoFd) Low for the second time, it should then set nAckReverse (PError) High. If software has data to send, it should drive nPeriphRequest (nFault) Low at the same time, and optionally program a DMA channel to provide the data. Whether or not it has data to send, software should then set one of the two ECP Forward modes.

2. In these modes, the controller configures PIA27-20 as inputs regardless of the contents of register E2. On entry to one of these modes, the controller clears the Idle bit, if it had been set.

3. For each byte, the controller waits for the host to drive HostClk (nStrobe) to Low. When HostClk (nStrobe) is Low and software, or the DMA channel, has taken any previous byte and thus cleared DREQ, operation diverges into four cases depending on the state of HostAck (nAutoFd), the mode, the MSbit of the data, and the state of an internal 7-bit Run-Length Encoding (RLE) counter.

If HostAck (nAutoFd) is High, indicating that this byte is neither an RLE value, nor a Channel Address, the controller captures the data from PIA27-20 into the Input/Output Register, sets DREQ to request software, or the DMA channel, to take this byte, and drives PeriphAck (Busy) High. If the RLE counter is zero, the controller waits (if necessary) for the host to drive HostClk (nStrobe) back to High, after which it drives PeriphAck (Busy) back to Low and returns to the event sequence at the start of paragraph #3. If the RLE counter is non-zero, the controller waits for software, or a DMA channel, to read the byte from the Input/Output Register, negates DREQ only momentarily, and decrements the RLE counter. It does this until the RLE counter is zero, at which point it proceeds as described above. Thus an RLE value of "n" results in the next byte being provided to software, or a DMA channel "n+1" times.

4. If HostAck (nAutoFd) is Low and the MS bit of the byte is zero (PIA27 is Low), the byte is an RLE repeat count. If the mode is "hardware RLE expansion," the controller transfers (the seven LS bits of) it to the RLE counter, leaves DREQ cleared, and drives PeriphAck (Busy) High.

5. Thereafter, the controller waits for the host to drive HostClk (nStrobe) back to High, at which time it drives PeriphAck (Busy) back to Low, and returns to the event sequence at the start of paragraph #3.

6. If HostAck (nAutoFd) is Low, and PIA27 is High, the byte is a "channel address." In this case, or when PIA27 is Low and the mode is "software RLE handling," the controller captures the data from PIA27-20 into the Input/Output Register, leaves DREQ cleared to keep a DMA channel from storing the byte, and sets the Idle bit, which it does not otherwise set while in this mode. Software should respond to this condition by reading the byte from the PIA 2 data register E3. Software can then do whatever else is needed to handle the situation, and then set Busy High. Thereafter the controller clears Idle, waits (if necessary) for the host to drive HostClk (nStrobe) back to High, and then drives PeriphAck (Busy) back to Low and returns to the event sequence at the start of paragraph #3.

While this mode is set, if data to send becomes available, software should drive nPeriphRequest (nFault) Low to alert the host of this fact. Also software should monitor the controller for either of two conditions:

a. If the host drives nReverseRequest (nInit) Low in response to nPeriphRequest (nFault) Low, software should drive nAckReverse (PError) Low, optionally program a DMA channel to provide the data, and set Peripheral ECP Reverse mode.

b. If P1284Active (nSelectIn) goes Low, the controller sets the IllOp bit in PARC, if this occurs between the time the host drives HostClk (nStrobe) Low, and when the controller subsequently drives PeriphAck (Busy) back to Low, in which case software should immediately enter Peripheral Compatibility/Negotiation mode. If P1284Active goes Low, but IllOp stays zero, indicating a "legal" termination, software should enter Peripheral Inactive mode and sequence the nAckReverse (PError), PeriphAck (Busy), PeriphClk (nAck), nPeriphRequest (nFault), and Xflag (Select) lines to leave ECP mode.

Status interrupts in Peripheral ECP Forward mode include rising and falling edges on P1284Active (nSelectIn) and nReverseRequest (nInit).

### 11.2.15 Host ECP Reverse Modes

1. In these modes the controller configures PIA27-20 as inputs, regardless of the contents of register E2. On entry to one of these modes, the controller clears the Idle bit, if it had been set.

2. For each byte, the controller waits for the peripheral to drive PeriphClk (nAck) Low. When this happens, and software, or the DMA channel, has taken any previous byte from the Input/Output Register and thus cleared DREQ, operation diverges into four cases, depending on the state of PeriphAck (Busy), the mode, the LS bit of the data, and the state of an internal 7-bit RLE counter.

   If PeriphAck (Busy) is High, indicating that this byte is neither an RLE value nor a Channel Address, the controller captures the data from PIA27-20 in the IOR, sets DREQ to notify software, or the DMA channel to take the byte, and drives HostAck (nAutoFd) High. If the RLE counter is zero, the controller then waits (if necessary) for the peripheral to drive PeriphClk (nAck) back to High, after which it drives HostAck (nAutoFd) back to Low and returns to the event sequence at the start of paragraph #2. If the RLE counter is non-zero, the controller waits for software, or the DMA channel, to read the byte from the IOR, negates DREQ only momentarily, and decrements the RLE counter. It does this until the RLE counter is zero, at which point it proceeds as described above. Thus an RLE value of “n” results in the next byte being provided to software or a DMA channel “n+1” times.

3. If PeriphAck (Busy) is Low, and the MSbit of the byte is zero (PIA27 is Low), the byte is an RLE repeat count. If the mode is “hardware RLE expansion,” the controller transfers (the seven LSbits of) it to the RLE counter, leaves DREQ cleared, and drives HostAck (nAutoFd) High. Thereafter the controller waits for the peripheral to drive PeriphClk (nAck) back to High, at which time it drives HostAck (nAutoFd) back to Low and returns to the event sequence at the start of paragraph #2.

4. If PeriphAck (Busy) is Low, and the MSbit of the byte is 1 (PIA27 is High), the byte is a “channel address”. In this case, or when the LSbit is zero, but the mode is “software RLE handling,” the controller captures the data from PIA27-20 in the IOR, leaves DREQ cleared, to keep a DMA channel from storing the byte, and sets Idle, which it does not otherwise set in this mode. Software should respond to this condition by reading the byte from the PIA 2 data register E3, reprogramming a DMA channel, if necessary, and doing whatever else is needed to handle the channel address, and finally setting HostAck (nAutoFd) High. Thereafter the controller clears Idle, waits for the peripheral to drive PeriphClk (nAck) back to High, and then drives HostAck (nAutoFd) back to Low, and returns to the start of the event sequence in paragraph #2 above.

5. If data has become available to be sent while this mode is in effect and software elects to send it, it should drive nReverseRequest (nInit) to High, set Host Negotiation mode to take full control of the interface, wait for nAckReverse (PError) to go High, and then set PIA27-20 as outputs.

6. Status interrupts in Host ECP Reverse mode include rising and falling edges on nPeriphRequest (nFault). nPeriphRequest carries a valid “reverse data available” indication during Reverse ECP mode. If so, enable status interrupts during this mode; if not, disable them.

### 11.2.16 Peripheral ECP Reverse Mode

1. In this mode, as long as nReverseRequest (nInit) is Low, and P1284Active (nSelectIn) is High, the controller drives the contents of the Input/Output Register onto PIA27-20, regardless of the contents of the E2 register. On entry to this mode, the controller sets Idle, and sets DREQ to request data from software, or a DMA channel.

2. If software, or a DMA channel, writes data to the Output Holding Register while the Input/Output Register is empty, the controller immediately transfers the byte to the IOR, clears Idle, and negates DREQ only momentarily, to request another byte.

3. In this mode, an alternate address for the Output Holding Register allows software to send a "channel address" or an RLE count value. Such bytes are not typically written by a DMA channel. Writing to this alternate address loads the OHR and clears DREQ, the same as writing to the primary address, but clears a ninth bit set when software, or a DMA channel, writes to the primary address. A similar ninth bit is associated with the IOR, and drives the PeriphAck (Busy) line in this mode.

4. As each nine bits arrive in the IOR, and thus out onto PIA27-20 and PeriphAck (Busy), the controller waits one PHI clock, and then drives PeriphClk (nAck) Low. It then waits for the host to drive HostAck (nAutoFd) High, after which it drives PeriphClk (nAck) back to High. The controller then waits for the host to drive HostAck (nAutoFd) back to Low. When this has happened, if software, or the DMA channel, has written a new byte to the Output Holding Register, and thus cleared DREQ, the controller transfers the byte to the IOR, sets DREQ again, and returns to the start of the event sequence in this paragraph. Otherwise, it returns to the event sequence at the start of paragraph #2. If software, or the DMA channel, doesn't provide new data within the time indicated by the PART register, the controller sets the Idle bit.

5. While this mode is in effect, software should monitor whether the host drives nReverseRequest (nInit) High. If it detects this, it should set the mode back to Peripheral ECP Forward, wait 500 ns and then drive nAckReverse (PError) back to High, before proceeding as described for Peripheral ECP Forward mode above.

6. Status interrupts in Peripheral ECP Reverse mode include rising and falling edges on P1284Active (nSelectIn) and nReverseRequest (nInit). Since there are no "legal terminations" during the time this mode is set, the controller sets IllOp for any falling edge on P1284Active (nSelectIn) in this mode.

Zilog, Inc. 210 East Hacienda Ave.
Campbell, CA 95008-6600
Telephone (408) 370-8000
FAX 408 370-8056
Internet: http://www.zilog.com