Generalized protocol for parallel-port handshaking

The parallel port is a common vehicle for data I/O for microprocessors. **B T G Tan** compares handshaking protocols for data transfer between parallel ports and proposes a generalized protocol

The parallel port is the most common method of data I/O for microprocessors. This paper compares the handshaking protocols for the most commonly used parallel ports, and shows how these different protocols can be described in terms of a generalized handshaking protocol. The handshaking protocol for the IEEE-488 bus is also treated in terms of this generalized protocol. The generalized protocol proposed may thus be useful for the analysis of data transfer between different types of parallel ports.

microsystems I/O interfacing parallel ports handshaking

The parallel I/O interface integrated circuit is perhaps the most basic of all I/O interface integrated circuits designed for the provision of I/O facilities to the real world by microprocessors. One such widely used integrated circuit is Motorola's 6821 peripheral interface adapter (PIA). In this paper, 'PIA' will be used as a generic term for such integrated circuits.

Most of these devices provide two, and sometimes three, parallel ports, through which several bits may be transferred in parallel to and from the real world. It is of course possible to transfer parallel data directly to and from the microprocessor data bus using buffers and other logic. The parallel port, however, provides a convenient interface for the end user of parallel data without the complication of interfacing with the microprocessor bus, and is thus often provided as a standard feature in microcomputer systems.

Parallel ports are generally 8 bit wide, allowing the transfer of 8 parallel bits simultaneously. For 16- and 32-bit microprocessors, two or more ports can be concatenated for longer data word lengths. Most parallel ports may be used for input and output, though not necessarily simultaneously. In some PIAs, like the 6821, the direction of each individual port data line can be determined by the user; in other PIAs, all data lines have to be defined as input or output lines. The directions of data lines and other aspects of the port are generally controlled by a control register directly addressable by the microprocessor.

Department of Physics, National University of Singapore, Kent Ridge, Singapore 0511

Paper received: 8 October 1988

HANDSHAKING

Most data transfers through parallel ports do not require handshaking. In some cases, a partial handshake is required between the port and the external device to ensure that the data are properly transferred. Perhaps the most common example of this is the Centronics-type parallel printer interface, in which the printer has to send a ready signal to the parallel port before the next character to be printed is output, otherwise the character output rate would be too fast for the printer to handle, resulting in the loss of characters.

Such partial handshakes are used when the external device is much faster or slower than the parallel port. In cases where the speeds of the port and the device are similar and data transfer must be effected accurately, a full or complete handshake with bidirectional signalling between PIA and the external device is necessary. When full handshaking is provided, the data transfer is asynchronous, since the handshaking allows the rate of transfer to be dictated by the speed of the port and the external device.

Most PIAs, except for the simplest ones, provide two control lines per port for full bidirectional handshaking[1-4]. However, there does not appear to be a uniform and consistent approach among different varieties of PIA in the implementation of handshaking signals, with no standard handshaking protocol which would allow ports to be directly interfaced to each other. Each variety of PIA has to be studied anew when considering their handshaking modes, making it confusing for the student and first-time user. In this paper, a generalized handshaking protocol applicable to most PIAs is proposed. The handshaking signals for each variety of PIA can then be more easily understood in terms of this generalized protocol. This would also facilitate the interfacing of different varieties of PIAs using handshaking.

0141-9331/89/09597-10 $03.00

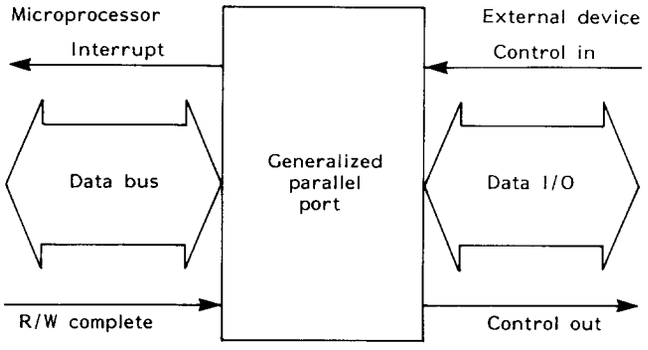

Figure 1. A generalized parallel port

Generalized parallel port

A generalized parallel port is first considered Figure (1). The port has eight data I/O lines for inputting or outputting data, and two conrol lines — control in and control out — for handshaking. It is assumed that the port, which is part of a PIA, is interfaced to a microprocessor via the microprocessor bus. The bus lines of interest are the data bus lines, the interrupt line which tells the microprocessor to commence reading or writing the data, and the R/W complete line through which the microprocessor signals to the port that the data have been read or written. The other lines, such as the address lines, are not shown as they do not participate in the handshaking protocol.

The handshaking protocol is first considered when the port is inputting data from an external device. It is assumed that the port is waiting for the external device to indicate that it has data for the port. When this occurs, the following sequence of events takes place.

1 An input handshake signal from the external device is sent in via the control in line to indicate that data are available for the port to read.
2 The port sends a signal via the interrupt line to the microprocessor to initiate a read routine by the microprocessor.
3 The microprocessor reads the data from an input register connected to the data I/O lines.
4 The microprocessor completes the read operation and indicates this to the port via the R/W complete line.
5 The port sends a signal via the control out line to the external device to indicate that it has accepted the data and is ready for the next set of data.

This may be represented by a time sequence diagram as shown in Figure 2a, in which the port may be in one of two possible states.

- Ready: during this state the port is ready to accept data from the external device.
- Reading: during this state the microprocessor reads the data from the port.

The handshaking signals are as follows:

- RFD: ready for data (signal from the port to the external device to indicate that it is ready for more data)
- DAV: data valid (signal from the external device to the port to indicate that it may start reading the data which is now available)

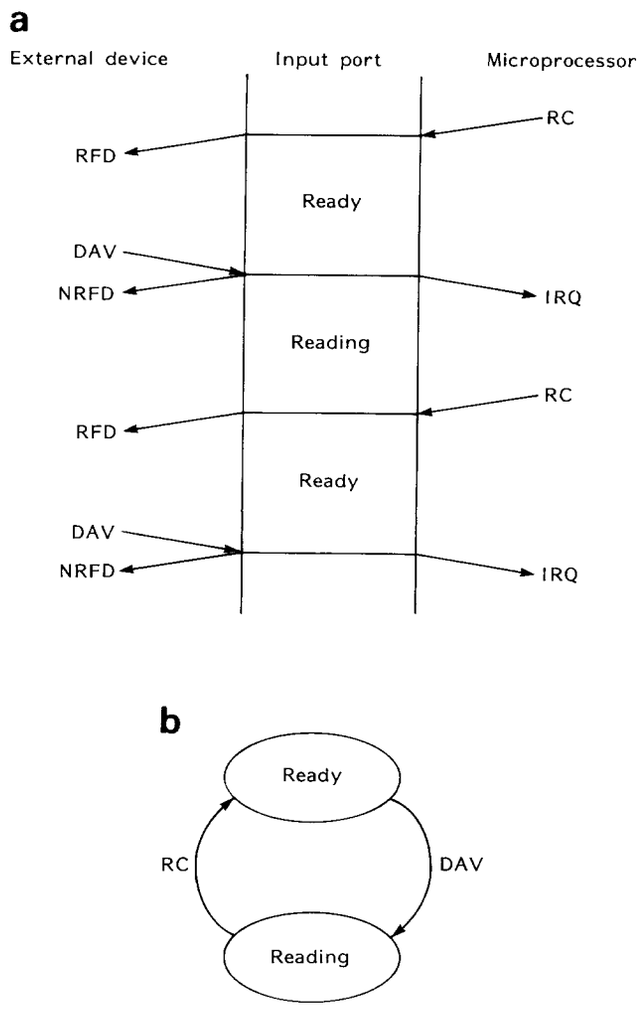

*Figure 2. **a**, generalized input port time sequence diagram and **b**, generalized input port state diagram*

- IRQ: interrupt request (interrupt signal to the microprocessor to commence the read routine)
- RC: read complete (signal from the microprocessor on completion of the reading of data)

The time sequence diagram also shows an optional NRFD or Not ready for data signal which the port outputs at the start of its reading data. Likewise, the negation of the DAV signal is the NDAV signal.

A state diagram may also be constructed for the input port as shown in Figure 2b.

Three-state latched input ports

Some ports, when inputting data, have a latching capability which means that the port can hold the input data in an input buffer register for the microprocessor to read[5]. If the input port is latched, then the external device need only make the data available until latching is complete, and not until the microprocessor has read the data. A signal may thus be sent to the external device when latching is complete to indicate that the data have been accepted and the external device can remove the data. This is defined as follows:

DAC: data accepted (signal from the port to the external device to indicate that data has been accepted)

The RFD signal is then sent separately when the microprocessor has read the data. The input port thus enters a third state while latching takes place, between the ready and reading states. This state is defined as follows:

Latching: during this state the input port latches the data into an input register from which the microprocessor can read the data.

The DAC signal may be sent out on the control output line as the transition in the opposite direction to the RFD signal; in this case there would be no NRFD signal. If a separate NRFD is required, then two lines must be used, one each for DAC and RFD.

The time sequence and state diagrams for the three-state latched input port are shown in Figures 3a and 3b respectively.

The termination of the latching state is initiated internally within the input port, since it is a purely internal operation of the port. The internal signal to terminate the latching state is named LC ('latch complete') in the state diagram.

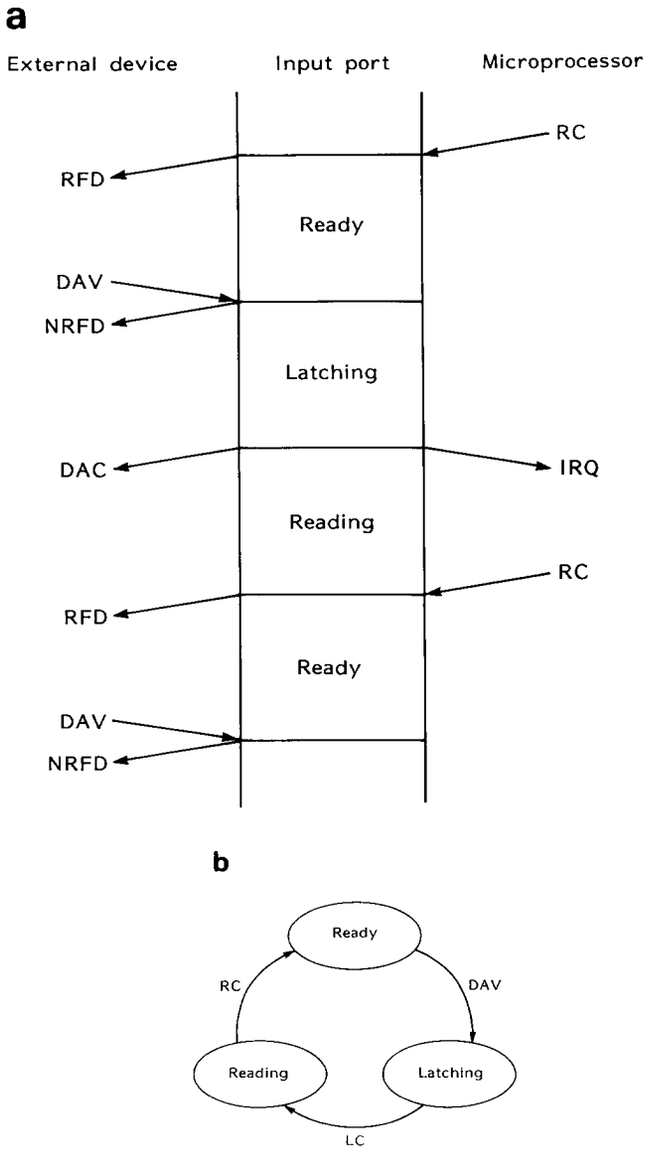

Figure 3. ***a****, generalized latched input port time sequence diagram and* ***b****, generalized latched input port state diagram*

OUTPUT PORTS

The port is now considered as an output device, sending data to an external device. Such an output port may also respond to a handshake signal from the external device which requests data and thus initiates the transfer of data from the microprocessor by a write operation to the port. The following sequence of events, analogous to that for the input port, may thus occur.

1 A handshake signal from the external device sent in via the control input line indicates that the device is ready to receive data.
2 The port sends a signal via the interrupt line to the microprocessor to initiate a write routine by the microprocessor.
3 The microprocessor proceeds with the write operation.
4 The microprocessor completes the write operation and sends a signal via the R/W complete line to indicate this to the port.
5 The port sends a signal via the control out line to the external device to indicate that the data are now valid for the external device to read.

This sequence can also be represented by a time sequence diagram as shown in Figure 4a, in which the output port may be in one of two possible states.

- Valid: during this state the data are available for the external device to read.
- Writing: during this state the microprocessor writes data to the port.

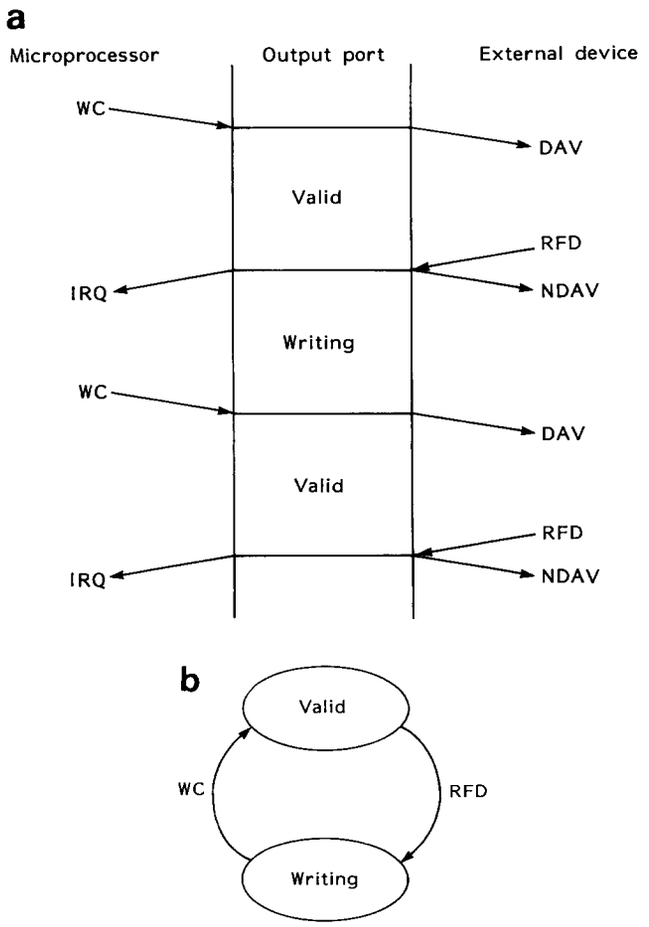

Figure 4. ***a****, generalized output port time sequence diagram and* ***b****, generalized output port state diagram*

The timing diagram also indicates an optional NDAV signal at the end of the valid state to indicate that the data are no longer valid.

It may be seen in the timing diagram that the output port responds to an RFD signal by initiating the interrupt signal to the microprocessor and sends out a DAV signal when the write process is completed. RFD and DAV have the same meaning as in the time sequence diagram for the input port. IRQ then initiates a microprocessor data write routine, while WC indicates the completion of the writing of the data to the port.

It is also assumed that the output port is latched, i.e. the data which appear at the output port will be held there in an output buffer register even after the microprocessor has removed the data from its data bus.

A state diagram for the output port is given in Figure 4b.

INITIATING THE HANDSHAKE

In the case of the input port, the DAV signal from the external device initiates the reading of data by the microprocessor; in the case of the output port, the external device sends an RFD signal to initiate the writing of data by the microprocessor. In most manufacturers' data sheets describing the handshake protocols, the output handshake is described as being initiated by the port when it sends out the DAV signal to tell the external device that the data are available. For consistency between the input and output handshakes, the output handshake has been described as being initiated by an RFD signal from the external device.

In this way, both handshakes appear to be similar, as in both cases an external signal initiates an interrupt routine to transfer data between microprocessor and port. In any case, since the handshakes are cyclic in nature, the selection of a starting point is quite arbitrary.

REAL-WORLD PARALLEL PORTS AND SIGNALS

The generalized parallel port and handshaking protocols which have been described can serve as a useful model for the description of the parallel ports in actual use. Each of the real-world ports which will be discussed has two or more ports, each port having a control in line and a control out line for handshaking signals. A signal may be conveyed on a handshaking line in one of three ways:

- by a single pulse
- by a low-to-high transition
- by a high-to-low transition.

When a single pulse is used, then the line generally conveys only one type of signal. If transitions are used for handshaking signals, two types of signals may be conveyed on one line. The two possible transitions may each be used to convey a signal and its negation such as RFD and NRFD, or two completely different signals such as RFD and DAC.

The PIAs that will be considered are the following.

- The Motorola 6821 and the functionally identical MOS Technology 6520 peripheral interface adapter (PIA).
- The MOS Technology 6522 versatile interface adapter (VIA).
- The Intel 8255 programmable peripheral interface (PPI).
- The Zilog Z-80 parallel I/O interface (PIO).

The ports provided by each of these chips are similar to the generalized port already described. Not all of these real-world ports can provide both an input and output handshake. All the real-world ports have the signal lines defined in the generalized port, although the lines are given different names by the manufacturers. The corresponding lines for each interface chip[6, 7] are shown in Table 1.

THE 6821 PIA

The handshaking modes of the 6821 and 6520 PIAs[6, 7] are now considered. As they are virtually identical, references to the 6821 should be taken to apply to the 6520 as well, the only difference being that the 6821 refers to the clock signal as E (enable) while the 6520 calls it ϕ_2. The 6821 PIA has two ports, A and B, each of which has a handshaking mode. However, port A's handshaking mode is only for data input, while port B's handshaking mode is only for data output. As in the generalized ports, each port has a control in and a control out line (known as CA1 and CA2 respectively), in addition to the eight data I/O lines of the port. Each port has its own interrupt line, $\overline{\text{IRQA}}$ for Port A and $\overline{\text{IRQB}}$ for port B, which can initiate a microprocessor interrupt.

One major feature of the 6821 is that it is controlled by an external clock signal with its own input line to the PIA, the E signal line, which is derived from the microprocessor clock. This E line is functionally equivalent to the R/W complete line of the generalized model.

The input handshaking protocol for port A is considered first. The CA1 line will recognize either a high-to-low transition or a low-to-high transition as a DAV signal; the type of transition recognized may be selected by the user. On receipt of a DAV signal from the external device on CA1, CA2 goes high and the PIA sends a signal on the $\overline{\text{IRQA}}$ line, serving as the IRQ signal, to the microprocessor

Table 1. Handshake lines for real-world parallel ports

Line	Port									
	6821 PIA		*6520 PIA*		*6522 VIA*		*8255 PPI*		*Z-80 PIO*	
	In	Out	In	Out	In	Out	In	Out	In	Out
Control in	CA1	CB1	CA1	CB1	CA1	CB1	$\overline{\text{STB}}$	$\overline{\text{ACK}}$	$\overline{\text{STB}}$	$\overline{\text{STB}}$
Control out	CA2	CB2	CA2	CB2	CA2	CB2	IBF	$\overline{\text{OBF}}$	RDY	RDY
Interrupt	$\overline{\text{IRQA}}$	$\overline{\text{IRQB}}$	$\overline{\text{IRQA}}$	$\overline{\text{IRQB}}$	$\overline{\text{IRQ}}$	$\overline{\text{IRQ}}$	INTR	INTR	$\overline{\text{INT}}$	$\overline{\text{INT}}$
R/W complete	E	E	$\phi 2$	$\phi 2$	$\phi 2$	$\phi 2$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$

to start reading the data. Completion of the read routine is signalled to port A via the E signal. When the microprocessor completes a read operation, the negative edge of the next E clock pulse acts as an RC (read complete) signal, and the CA2 line undergoes a high-to-low transition to serve as an RFD signal.

The low-to-high transition made by CA2 in response to DAV on CA1 may be considered as the negation of RFD, i.e. NRFD to show to the external device that the port is now not ready to accept new data.

Port B of the 6821 provides a handshaking protocol for data output. The protocol is very similar to that for data input as provided by port A. The handshaking sequence is initiated by an input signal on CB1, which serves as an RFD signal from the external device which is to receive the data from the port. CB2 goes high in response to signify NDAV and $\overline{\text{IRQA}}$ is pulled low to act as the IRQ signal to initiate the microprocessor write routine which sends data to the PIA. Completion of the write routine is indicated by the next E pulse (in a manner similar to the completion of the read routine for port A), and CB2 undergoes a high-to-low transition to serve as a DAV signal to the external device, which may then proceed to accept the data.

The timing waveforms for the 6821 PIA input and output handshakes are shown in Figures 5a and 5b respectively, labelled with the generalized signals as described above.

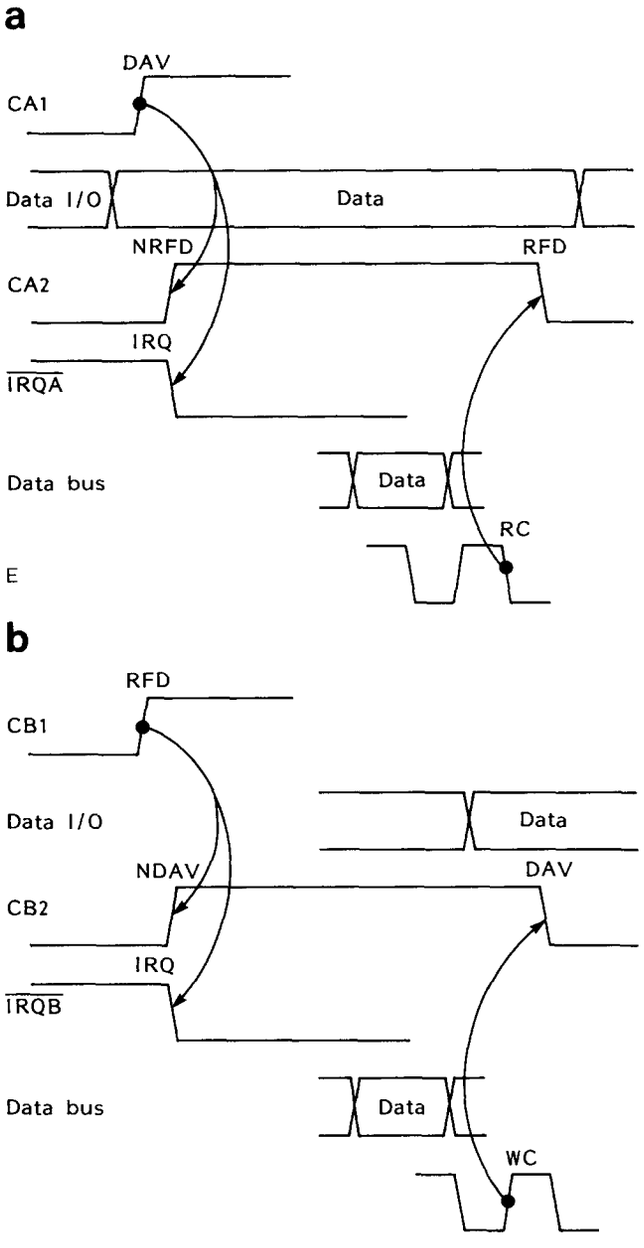

Figure 5. ***a****, 6821 PIA input handshake protocol timing waveforms and* ***b****, 6821 PIA output handshake protocol timing waveforms*

THE 6522 VIA

The 6522 versatile interface adapter (VIA), derived directly from the 6821 PIA, has a handshaking protocol which is identical with that of the 6821[6,7]. The major difference is that port A of the 6522 can provide both input and output handshaking signals. This is possible because CA2 will undergo a high-to-low transition on completion of either a read or a write operation by the microprocessor; the transition could then be interpreted as either RFD or DAV respectively. This is useful when some lines of port A are defined as inputs and the remainder as outputs. Both the 6821 and the 6522 can have port lines defined individually as input or output lines.

In addition, both ports A and B, when operated as input ports with handshaking, can latch the input data. This capability is switched on by a bit in the peripheral control register. With port A, for example, the DAV signal, which is received by CA1, will immediately initiate latching of the data on the input data lines. The raising of CA2 which follows may then be considered to be a DAC signal, since CA2 goes up simultaneously with the IRQ signal to the microprocessor after the data have been properly latched into the port.

Another minor difference between the 6821 and the 6522 occurs in the interrupt lines to the microprocessor; the 6522 has only one line, $\overline{\text{IRQ}}$, which is shared by both the A and B ports.

The output modes of both ports A and B behave identically with the output modes of the 6821 PIA.

The timing waveforms for the 6522 VIA latched input handshake are shown in Figure 6, labelled with the generalized signals.

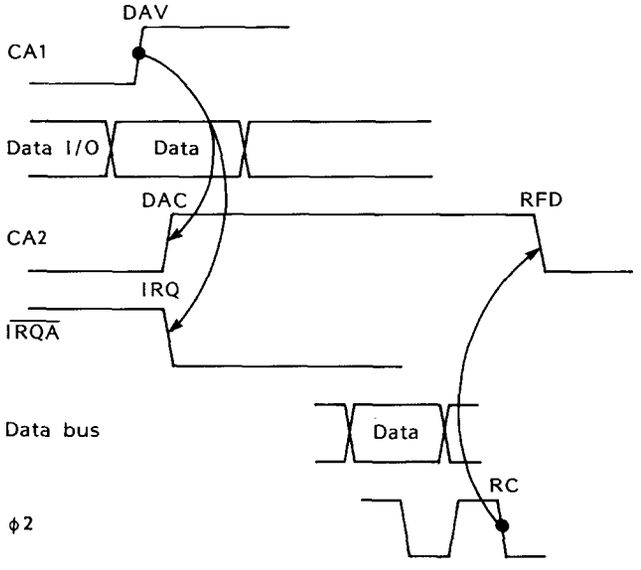

Figure 6. 6522 VIA input handshake protocol timing waveforms

THE 8255 PPI

The 8255 programmable peripheral interface (PPI) has three 8-bit ports labelled A, B and C[6,7]. In the handshaking mode, the C port is not used as such but its lines are used as handshaking lines for ports A and B. Both ports A and B can act as input or output ports. For both ports A and B, all eight lines of each port have to be defined as all input or all output. Both ports can do input and output handshaking. In addition, port A is capable of bidirectional operation, in which data may flow in either direction. Handshaking is also provided for this bidirectional mode.

By considering first either ports A or B as inputs, the control input line is the strobe or $\overline{STB}$ line, the control output line is the input buffer full or IBF line, the IRQ line is the INTR line and the R/W complete line is the $\overline{RD}$ line. When the ports are configured as outputs, the corresponding lines are $\overline{ACK}$, $\overline{OBF}$, INTR and $\overline{WR}$.

Ports A or B are considered first in mode 1 of the PPI in which they behave as latched input ports with handshaking. The external device sends a negative-going pulse via the $\overline{STB}$ line which serves as a DAV signal. Both pulse edges have a role to play: the leading edge causes the port to latch the data into its input buffer and the trailing edge casues the IRQ signal to occur on the INTR line. The $\overline{STB}$ pulse length must be long enough for the latching to occur. The IBF line goes high to indicate that the input buffer of the 8255 port is full; this is equivalent to an NRFD signal. The microprocessor interrupt routine then reads the data and empties the input buffer. The completion of the read is indicated by the RC signal on the $\overline{RD}$ line. This causes the IBF line to go low which serves as an RFD signal, thus signifying that the input buffer is empty.

Although the 8255 is able to latch its input data, it does not put out a DAC signal. The transition of IBF which responds to the $\overline{STB}$ signal is an NRFD and not a DAC signal, since it occurs before the latching operation is completed. The $\overline{STB}$ pulse is made long enough to ensure that latching is complete before the microprocessor interrupt occurs. Hence a DAC signal is not really needed, since the external device 'knows' when latching is complete so that it can remove the data.

The A and B ports can also operate as output ports under mode 1. The handshaking protocol is very similar to that for input, except that the $\overline{ACK}$ line from the external device (instead of the $\overline{STB}$ line) initiates the data transfer by sending a pulse which acts as an RFD signal. The leading edge of this pulse causes the $\overline{OBF}$ line to go high as an NDAV signal, while the trailing edge triggers an IRQ signal on the INTR line to start a microprocessor interrupt write routine. Completion of the write operation is signalled by the WC on the $\overline{WR}$ line, causing the $\overline{OBF}$ to go low which serves as a DAV signal, showing that the output buffer of the port is now fully loaded with the output data.

In its mode 2, the 8255 permits port A to be used as a bidirectional port, with all eight lines able to handle both input and output data. In this mode, port A has both the input and output handshaking lines — $\overline{STB}$, IBF, $\overline{ACK}$ and $\overline{OBF}$, with INTR serving both input and output of data. The handshaking protocols are identical with the separate input and output protocols, except that both protocols may occur as required on port A.

Figures 7a and 7b show the timing waveforms, labelled with the generalized signals, for the 8255 PPI input and output handshakes respectively.

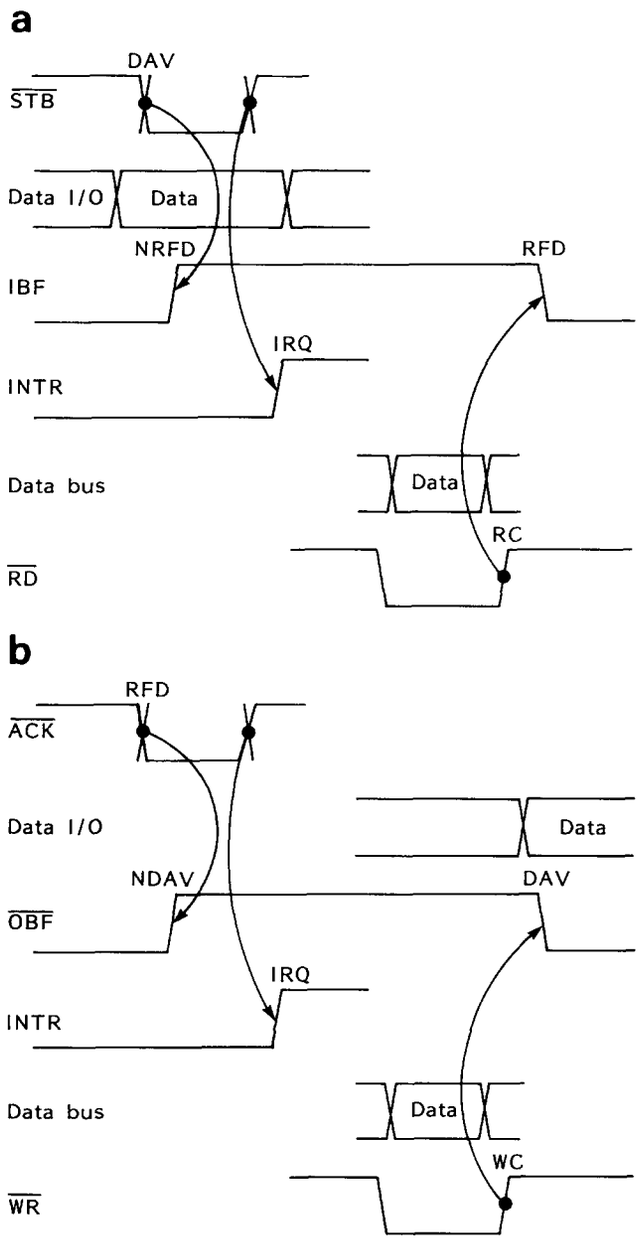

Figure 7. ***a****, 8255 PPI input handshake protocol timing waveforms and* ***b****, 8255 PPI output handshake protocol timing waveforms*

THE Z-80 PIO

The Z-80 parallel I/O interface (PIO), which has two parallel ports A and B, has some similarities to the 8255 PPI[6,7]. Each of the A and B ports can act as outputs in mode 0 or inputs in mode 1. Port A can, as in the 8255 PPI, act as a bidirectional port in mode 2. In each port the strobe ($\overline{STB}$) line is equivalent to the control input line and the ready (RDY) line is equivalent to the control output line. Both ports have a common interrupt line, $\overline{INT}$, equivalent to IRQ while the $\overline{RD}$ and $\overline{WR}$ lines for the input and output modes respectively are equivalent to the R/W complete line.

The input handshaking protocol is initiated by an external device via the STB line, which sends a negative-going pulse signal indicating DAV. Like the 8255, the Z-80 PIO can also latch its input data; the leading edge of the $\overline{STB}$ pulse initiates the latching of the data into the input

port, and should be long enough to enable the latching to be complete.

The trailing edge of the pulse will thus occur after latching is complete. This trailing edge causes the RDY line to go low, serving as a DAC signal and thus signifying the completion of latching, and also causes the $\overline{INT}$ line to go low to serve as an IRQ signal, initiating a microprocessor interrupt to read the incoming data. The $\overline{RD}$ line goes low while a read operation takes place, and goes high to serve as an RC signal when the read operation is completed. This in turn causes the RDY line to undergo a low-to-high transition, serving as an RFD signal to the external device.

The output handshaking of mode 0 is similar in principle. The external device initiates the sequence by sending a negative-going pulse on the $\overline{STB}$ line as an RFD signal. The trailing edge of this pulse causes the RDY line to go low, serving as an NDAV signal, and the $\overline{INT}$ line also to go low, serving as an IRQ signal. The IRQ initiates a microprocessor interrupt routine to write new data to the port, during which the $\overline{WR}$ line is pulled down. When the write operation is complete, the $\overline{WR}$ line goes up to serve as a WC signal, thus causing a positive-going transition on the RDY line, which serves as a DAV signal to the external device.

The Z-80 PIO input and output handshake timing waveforms labelled with the generalized signals are given in Figures 8a and 8b.

It has thus been shown that the generalized protocol signals can be used to label the handshaking signals of the 6821, the 6522, the 8255 and the Z-80 PIO. This helps to clarify the relationship between their parallel ports and shows that their handshaking protocols are basically similar in principle. As a result, these ports can be more easily interfaced to each other for data transfer with handshaking. For example, in interfacing the 6821 as an output port to the 8255 as an input port, the DAV signal from the 6821 must be converted into a negative-going pulse which can be recognized as an $\overline{STB}$ signal by the 8255.

Generalized handshaking protocol

A generalized handshaking protocol between two generalized parallel ports is now considered. It is assumed that the control in line of each port is connected to the control out line of the other port, and the data I/O lines of both ports are connected to each other. The objective is to effect the transfer of data from the memory of one microprocessor to the memory of another microprocessor via the two parallel ports. A time sequence diagram showing the handshaking protocol for this asynchronous data transfer process can be obtained by matching corresponding handshaking signals between the ports, as is shown in Figure 9.

It will be seen that each port initiates the interrupt of the other port's microprocessor in turn, giving a symmetrical and orderly interlocked sequence in which data are transferred from the first microprocessor to the output port, from the output port to the input port, and then to the other microprocessor. It is assumed that the two microprocessors are properly set up so that their respective IRQ signals will initiate a corresponding write and read routine. For example, a number of contiguous memory locations of the output port's microprocessor could be

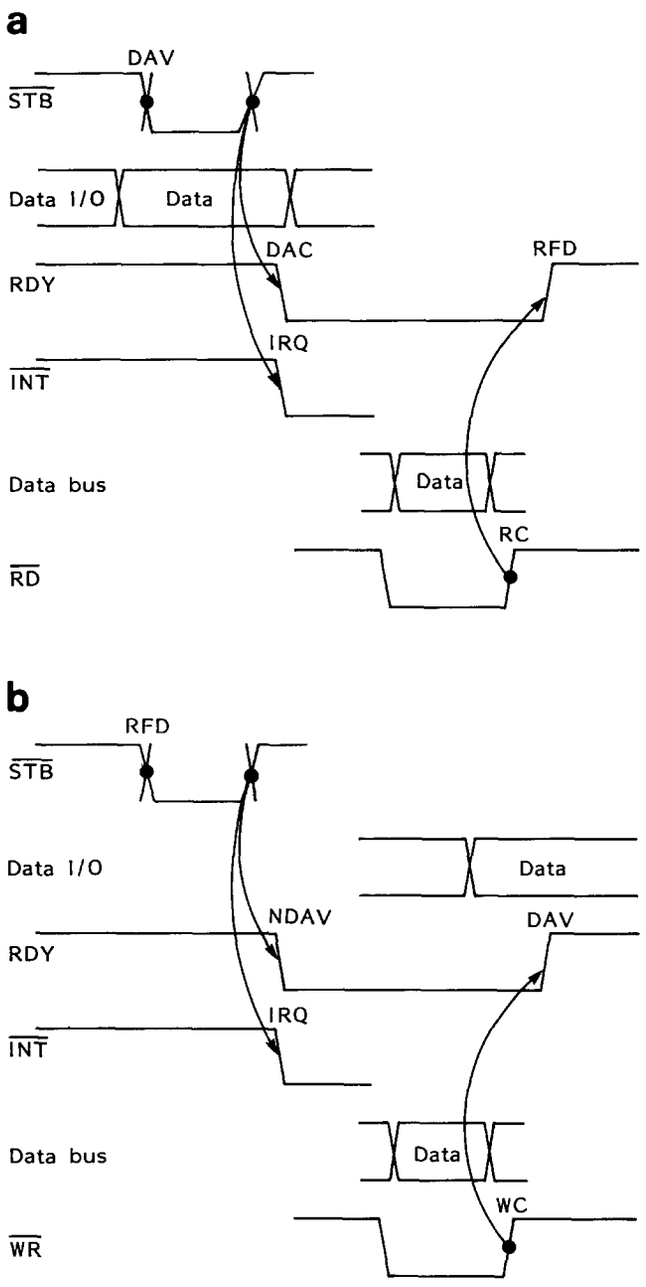

*Figure 8. **a**, Z-80 PIO input handshake protocol timing waveforms and **b**, Z-80 PIO output handshake protocol timing waveforms*

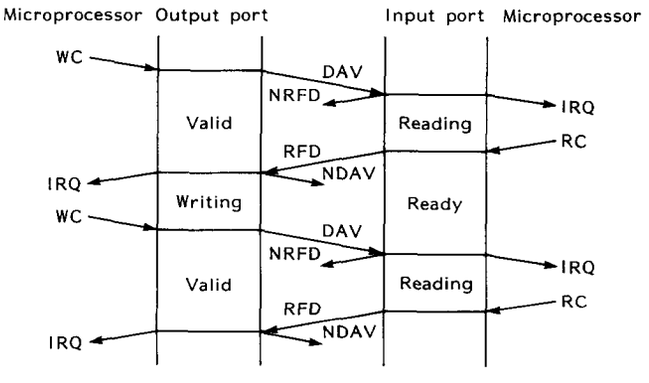

Figure 9. Time sequence diagram for handshake protocol between generalized input and output ports

accessed sequentially to send a series of data bytes to a contiguous set of memory locations of the input port's microprocessor.

The entire sequence should be initiated by either microprocessor by sending an RFD or a DAV signal, as appropriate, to initiate an interrupt in the other microprocessor. The first microprocessor then waits in a loop while the other microprocessor executes its interrupt routine. When this interrupt routine is complete, the second microprocessor sends a signal to initiate an interrupt routine in the first microprocessor, before going into a loop to await its next interrupt. This entire sequence could be terminated by a predetermined 'end-of-data' character which would enable both microprocessors to exit their respective loops. Instead of waiting in a do-nothing loop, each microprocessor could do useful work while waiting to be interrupted by the other microprocessor.

This process closely resembles the passing of control between two co-routines, except that in this case the two interrupt routines are running on different microprocessors.

In the case of the 6821 PIA, for example, the read and write routines need not be initiated by interrupt requests. The microprocessors could check the appropriate status bit in the status registers of their respective ports to see if a DAV or an RFD signal, as the case may be, had occurred. If so, the microprocessors could take the appropriate action to read or write the data as required, without going through an interrupt sequence.

THREE-STATE INPUT PORT HANDSHAKING

If a three-state latched input port were to be interfaced with the two-state output port, the output port should simply ignore the DAC signal and use the RFD signal to initiate the writing state. The DAC signal should not be used to initiate the writing state. If it were used to do so, the writing state would then proceed in parallel with the input port's reading state and it would not be possible to predict which would terminate first, thus upsetting the protocol sequence.

If the DAC signal is to be made use of, it could initiate a third state in the output port, which is interposed between the valid and writing states. During this state, which will be called the erasing state, the data may be removed by the output port from their data lines. The output port then waits for the RFD signal to initiate the writing state so that it can transfer new data from its microprocessor. The time sequence diagram for this modified protocol is shown in Figure 10.

The state diagram for the modified three-state output port is shown in Figure 11a.

The advantage of this protocol is that the data may be removed from the data I/O lines earlier. In a system where the data lines constitute a bus used by more than one output port, the removal of the data frees the data bus for use by another output port, thus maximizing the throughput of data.

IEEE-488 handshaking protocol

The latching of input data by the input port, however, could be exploited more effectively. If the output port could initiate the writing of new data from its microprocessor as soon as the latching is complete, then the throughput of data from output to input port could be increased. However, as has been pointed out, using the DAC signal to initiate the writing state in the output port could lead to ambiguities in the protocol, since both the reading and the writing states would then proceed in parallel, and it would not be possible to predict which one would terminate first. Consequently, it would not be possible to predict whether the ready state or the valid state was initiated first, and the protocol sequence would become indeterminate.

The IEEE-488 or GPIB bus is an example of a parallel port system which has latched input ports, and which uses

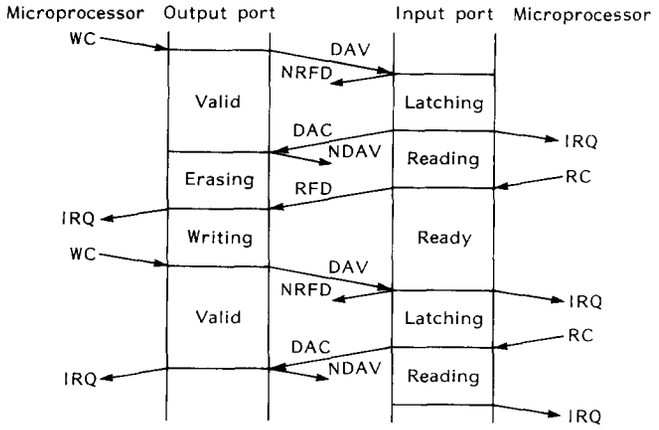

Figure 10. Time sequence diagram for handshake protocol between generalized latched input port and three-state output port

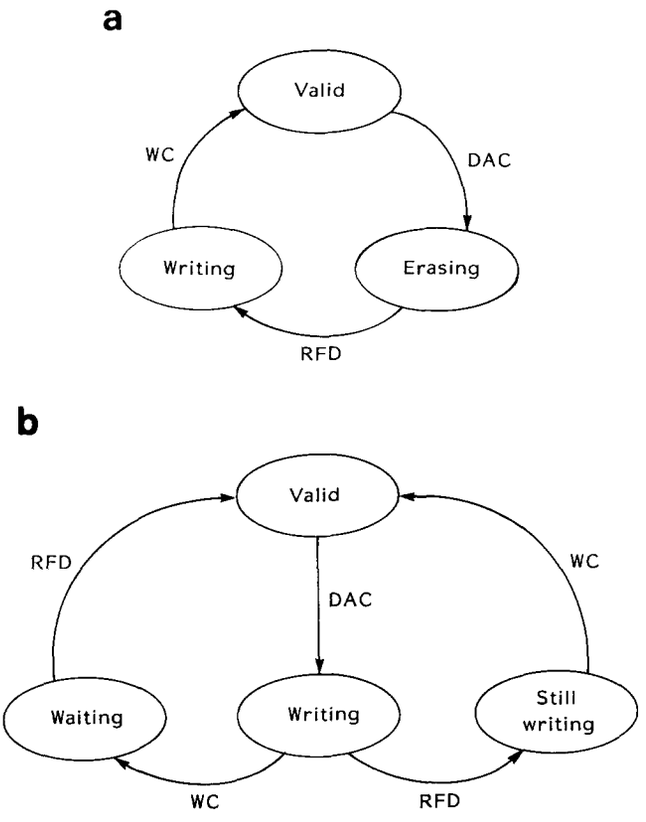

*Figure 11. **a**, Three-state output port state diagram, **b**, IEEE-488 talker state diagram*

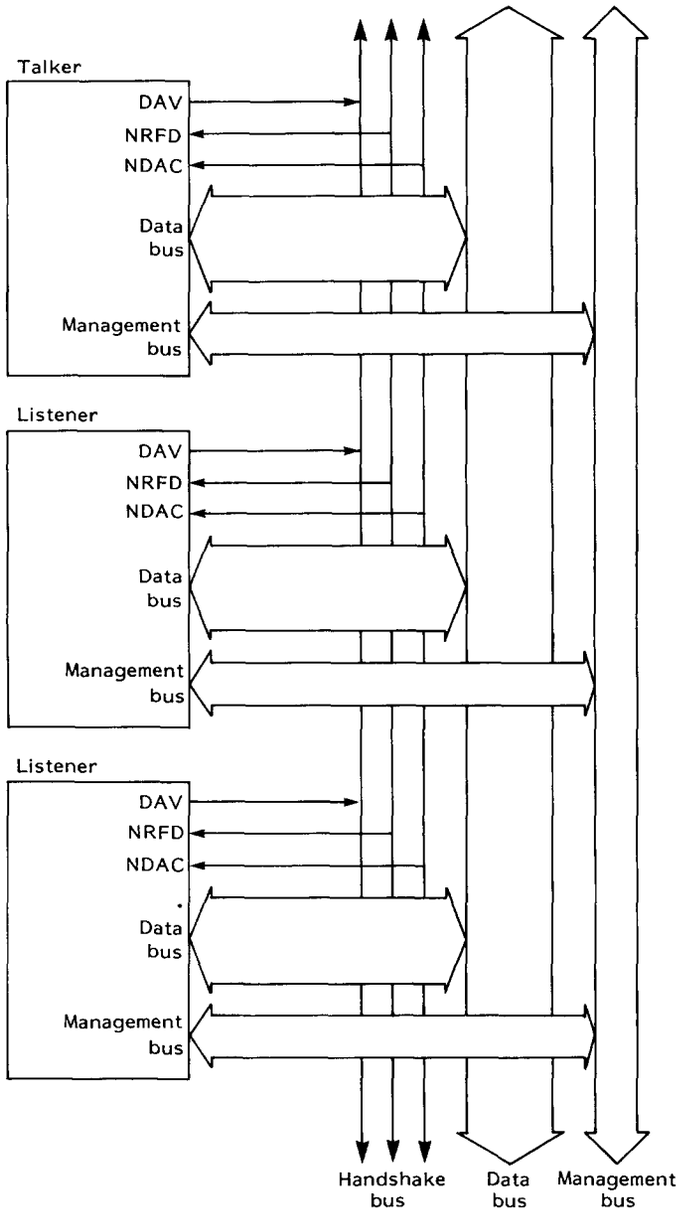

Figure 12. IEEE-488 system block diagram

the DAC signal to initiate the renewal of data in its output port[4,8]. The IEEE-488 handshake protocol uses three handshaking signals each with their own dedicated lines, NRFD and NDAC, which are effectively the same as RFD and DAC respectively, and DAV. These three lines enable an IEEE-488 output port or talker to transfer data to several input ports or listeners which read the data at varying rates. After latching the input data, each listener will output its DAC signal. The combined NDAC line will go low to indicate DAC to the talker only if all listeners have sent out their DAC signals. Similarly, each listener may send out its own RFD signal, but the NRFD line will go low to indicate RFD to the talker only if all listeners have sent out their RFD signals. In this way, listeners of widely differing speeds can be accommodated.

The block diagram for a typical IEEE-488 system given in Figure 12 shows, in addition to the handshake lines, the eight-line data bus and the five-line management bus, which manages the transfer of data on the data bus. The IEEE-488 protocol is illustrated in Figure 13 which shows the timing waveforms for the IEEE-488 handshake[8].

The IEEE-488 handshake may also be represented by the time sequence diagram of Figure 14. IRQ, RC and WC may now represent internal signals within listeners and talker. It should be noted that the DAC signal, and not the RFD signal, initiates the writing of new data in the talker. In the IEEE-488 protocol, the initiation of the talker's valid state is made conditional on both the completion of the writing of new data and the receipt of the RFD signal from the listener[8]. In this way, the valid state always starts after the ready state, no matter how long the writing state lasts. In Figure 14, the writing state is short enough to terminate before the RFD signal. The talker thus goes into a waiting state, in which the new data are still not valid for the listener, until the RFD signal is received. Only then does the talker go into its valid state and send out its DAV signal.

On the other hand, the writing state may be so long that the RFD signal is received before the writing of new data in the talker is complete. In this case, the talker goes into a 'still writing' state on receipt of RFD, until the completion of the writing of the new data. The talker then enters its valid state and sends out its DAV signal.

The state diagram for the IEEE-488 three-state talker handshaking protocol is given in Figure 11b (the state diagram for the listener protocol is similar to that already given for the latched input port). The diagram shows the transition from the writing state to the valid state for the

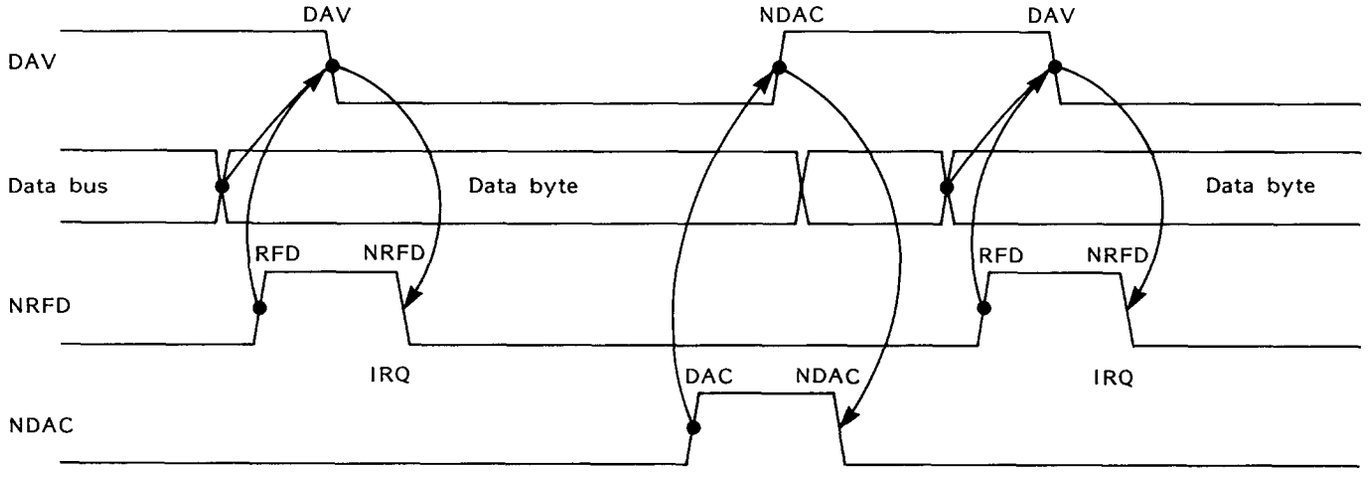

Figure 13. IEEE-488 handshake protocol timing waveforms

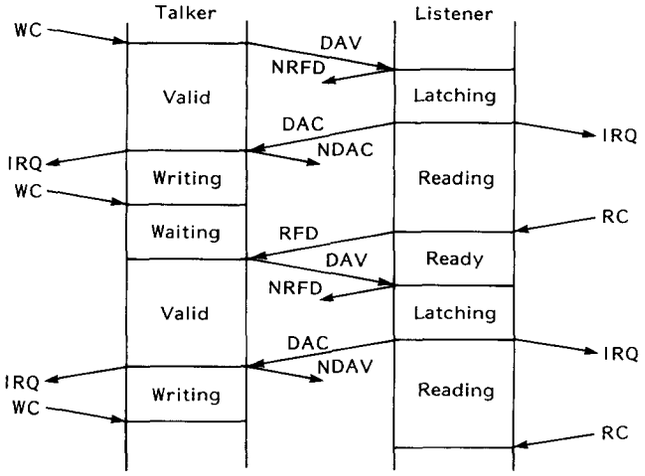

Figure 14. IEEE-488 handshake protocol time sequence diagram

two alternative situations given above; the transition to the valid state is shown to be dependent on the receipt of RFD.

None of the real-world parallel ports that have been considered can perform this kind of handshaking protocol. This is due to the fact that all their output handshaking protocols automatically send out a DAV signal on completion of the writing of data from the microprocessor. However, where an output port's control lines are programmable, such as the CB2 control line of the 6821/6520/6522, it would be possible to emulate the handshaking behaviour of the IEEE-488. This could be done, for example, by keeping CB2 high after the completion of a write operation until after an RFD signal had been sensed on the CB1 control line.

It was noted that, in the three-state protocols with separate and distinct DAC and RFD signals, the three key signals, RFD, DAV and DAC and the two negative signals, NRFD and NDAV, always proceed in the following sequence:

RFD ⟶ DAV ⟶ NRFD ⟶ DAC ⟶ NDAV

CONCLUSION

This paper has attempted to elucidate the relationships between the various handshaking protocols for data transfer used by different parallel I/O chips as well as the IEEE-488 interface. It is hoped that the generalized handshaking protocol described and applied to real-world protocols will prove to be a useful method of analysing data transfer between parallel ports of all types.

REFERENCES

1 **Davis, D J** 'Microprocessor systems' in *Advances in Electronics and Electron Physics 1981* Academic Press, New York, NY, USA (1981)
2 **Stone, H S** *Microcomputer Interfacing* Addison-Wesley, Reading, MA, USA (1983)
3 **Hill, F J and Peterson, G R** *Digital Systems Hardware Organization and Design* John Wiley, New York, NY, USA (1987)
4 **Hall, D V** *Microprocessors and Digital Systems* McGraw-Hill, New York, NY, USA (1983)
5 **Hill, F J and Peterson, G R** *Digital Logic and Microprocessors* John Wiley, New York, NY, USA (1984)
6 **Osborne, A** *An Introduction to Microprocessors, Vol II Some Real Products* Sybex, Paris, France (1976)
7 **Hofacker, W (ed.)** *Microcomputer Hardware Handbook* Elcomp, Pomona, CA, USA (1982)
8 **Krutz, R L** *Interfacing Techniques in Digital Design with Emphasis on Microprocessors* John Wiley, New York, NY, USA (1988)

Bernard Tan graduated in 1965 with an honours degree in physics from the University of Singapore, and in 1968 with a DPhil from Oxford University, UK. He is a chartered engineer and a member of the IEE and IEEE. Since 1968 he has taught at the National University of Singapore where he is now an associate professor in physics. His research interests include electronic properties and surface analysis of materials, digital acoustal and image processing and microprocessor applications.