Cornell Local Network (CLONE)*

Interface Unit

Principles of Operation

TR 81-461
May 1981

Department of Computer Science
Upson Hall
Cornell University
Ithaca, New York 14853

Cornell Local Network (CLONE)*

Interface Unit

Principles of Operation

May 25, 1981

ABSTRACT

This document describes the architecture and operation of a CLONE Interface Unit. It is the defining document.

*This work is supported, by NSF Grant MCS-76-22360, MCS-79-01048, an unrestricted "New Science" grant from the IBM Corporation and the Bell Telephone Laboratories One Year On Campus Program.

Preface

The design and implementation of CLONE is a collaborative effort between the Department of Computer Science and the School of Electrical Engineering at Cornell University. The project staff includes: H. Boehm, L. Burley, L. Faust, C. Lee, G. Pepenella and R. Rupp under the supervision of A. Kratzer (EE), A. Demers (CS) and F. Schneider (CS).

Notational Conventions:

All numbers used in this document are decimal.

Hexidecimal constants are denoted: X'ab', where a and b denote nibbles.

Within a word, bit positions are numbered right to left, starting with 0. Thus, bit 0 is always the least-siginificant-bit of the word.

1. Introduction

CLONE is a local computer network that is designed to permit empirical evaluation of multiple access protocols. It facilitates communication among a collection of *hosts*. Hosts can be computers or terminals; a host need not have the capability to directly execute a communications protocol.

A *contention network* is structured around a single, shared communications channel, which provides only one path between any two hosts, as is illustrated in Figure 1.

```
                Shared Communications Channel ("Ether")
<--------.----------------.--------------------------.------>
         |                |                          |
         |                |                          |
   +----^----+      +----^----+                +----^----+
   | Host  1 |      | Host  2 |    .  .  .     | Host  n |
   +---------+      +---------+                +---------+
```

Figure 1 -- A Contention Network

Typically, this channel is a passive medium, such as a coaxial cable, a radio channel or a fiber optic cable. In CLONE, a coaxial cable is used. Each host is connected to the channel by an *interface unit*. The interface unit allows a host to broadcast a message on the shared channel. It also monitors the channel, copying any messages addressed to its host(s). This allows both single-destination and multi-destination messages to be transmitted. Each CLONE Interface unit can support up to four hosts operating in accordance with the RS-232C serial bit transfer protocol, or one host that makes 16 bit parallel transfers.

A *multiple access protocol* is used to arbitrate access to the shared channel. Various multiple access protocols have been developed for different applications. In each, some portion of the channel capacity is used to coordinate access to the channel. A *collision* occurs whenever more than one signal is placed on the channel at a time. Some multiple access protocols attempt to minimize the amount of time spent on collisions, while others exploit the existence of collisions to provide the necessary synchronization for governing access to the channel.

CLONE, as the name suggests, can be used to emulate any of the multiple access protocols that have been described in the literature. This flexibility requires a rather sophisticated interface unit with the following capabilities:

A single transmitter must be able to drive all of the receivers simultaneously.

It must be possible for all CLONE Interface units to transmit simultaneously without damaging the receivers, transmitters or shared channel.

Each interface unit must have a unique, hard-coded identification number.

An interface unit must be able to determine whether the channel is in use.

While transmitting, an interface unit must be able to detect collisions

An interface unit must be able to broadcast a "jam" signal, which can be sensed by all other interface units.

Each interface unit must have sufficient processing power and memory to implement the multiple access protocols we wish to study.

The rest of this document describes the architecture of a CLONE Interface Unit. It is intended to serve as a reference manual for programmers who wish to write multiple access protocols that run in an interface unit. The document does not describe the protocol employed for downloading a program into an interface unit.

2. Interface Unit -- Basic Architecture

The CLONE network interface consists of two identical Processor Units (PUs) and a Shared Bus Unit (SBU), as depicted below.

```
+---------------+                         +---------------+
|   Processor   |                         |   Processor   |
|     Unit      |                         |     Unit      |
|       1       |                         |       2       |
+-------.-------+                         +-------.-------+
        |                                         |
        |                                         |
+---------------------------------------------------------+
|                                                         |
|                   Shared  Bus  Unit                     |
|                                                         |
+---------------------------------------------------------+
```

Figure 2 -- CLONE Interface Unit Organization

The organization of a Processor Unit is shown in Figure 3. The microprocessor is a Motorola 6809E. Each Processor Unit contains some Read-Only Memory (ROM) and a Priority Interrupt Controller (PIC) with associated logic for enabling and disabling interrupts. A Processor Unit also includes an Interval Timer (IT) and an Address Trap Unit (ATU), which are referred to as private periperals in this document. These private peripherals can cause interrupts on the processor to which they are connected.

The ROM is used to store an initial protocol to allow access to the network, and allow other contention protocols to be loaded into the interface unit. The IT allows an interrupt to be scheduled for a specified time in the future. The ATU monitors the processor unit address lines and signals an interrupt if a specified address is generated. This is useful for implementing repetitive control structures.

```
          +---------------+
          | Microprocessor |
          +------.--.------+
                 |  |
Data <------|--^-------.----------.----------.----------.--->
            |     +----^--+        |          |          |
Addr <------^-----|       |------.-|---------.-|---------.-|--->
                  | P I C |      | |         | |         | |
                  |       |   +--^-^--+   +--^-^--+   +--^-^--+
                  |       |   |  ROM  |   |  IT   |   |  ATU  |
                  +-------+   +-------+   +-------+   +-------+
```

Figure 3 -- Processor Unit

The Shared Bus Unit (Figure 4) contains some Random Access Memory (RAM), an Ether Transciever (XCVR), a switch-settable Interface Identification Register (IIDR), an array of 8 Load-and-Set (L/S) registers and a Host-Processor Interface (HPI). The XCVR and HPI will be referred to as shared peripherals in this document. Shared peripherals can cause interrupts on one or both processors.

The XCVR allows the CLONE interface to transmit and receive messages on a shared coaxial cable. The IIDR provides each interface unit with a software-readable, network unique address. The L/S locations are useful for synchronization of concurrent processes. And, the HPI is used to connect a CLONE interface to a host. Two types of hosts can be accomodated. A 16-bit parallel interface facilitates connection of high-speed hosts to CLONE, a four-port RS-232 serial interface allows connection of slower hosts.

```
Addr <-----.----------.----------.----------.----------.--->
           |          |          |          |          |
Data <---.-|--------.-|--------.-|--------.-|--------.-|--->
         | |        | |        | |        | |        | |
      +--^-^--+  +--^-^--+  +--^-^--+  +--^-^--+  +--^-^--+
      |  RAM  |  | XCVR  |  |  HPI  |  |  L/S  |  | IIDR  |
      +-------+  +-------+  +-------+  +-------+  +-------+
```

Figure 4 -- Shared Bus Unit

3. Processor Unit

3.1. Interrupt Structure

3.1.1. Overview

The interrupt architecture of a processor unit is shown below. Each Processor Unit includes a Priority Interrupt Controller (PIC) to control processing of interrupts from 8 Interrupt Request Input lines labelled IR0, ..., IR7. Only private peripherals and shared peripherals can cause interrupts, as follows. A private peripheral is connected to the PIC IRx line on the processor unit of which it is part. Each shared peripheral is connected to the same interrupt request input line on both PICs.

```
+--------------------+
|                    |
|  Microprocessor    |
|                    |
+-----.---.----------+
      |   |
      |   |
Data  |   +------.----------------.--------------------------------->
      |          |                |
      |          |                |              +-------+
Addr  +--------.-|--------------.-|------------->|   A   |--------->
               | |              | |              |   D   |
            +-----+          +-----+             |   D   |
            | ISR |          | IBR |------------>|   R   |
            +-----+          +-----+             |       |
               | |    _                          |   T   |
               | +--->|\                         |   R   |
               |      | )----------------------->|   A   |
IR0 -----------|----->|_/                        |   N   |
               |       .                         |   S   |
               |       .                         |   L   |
               |       .                         |   A   |
               |      _                          |   T   |
               +----->|\                         |   I   |
                      | )----------------------->|   O   |
IR7 ----------------->|_/                        |   N   |
                                                 +-------+
```

Figure 5 -- CLONE Interrupt Architecture

Each PIC contains an Interrupt Select Register (ISR) and an Interrupt Base Register (IBR). The ISR is used to specify the set of interrupt request inputs that are honored. The IBR specifies the address of a table of addresses -- presumably of interrupt handling routines. If bit x of the ISR is 1, then interrupt request input IRx is enabled; otherwise interrupt request input IRx is disabled.

The interrupt architecture of the Motorola 6809 microprocessor is comprised of three classes of interrupt requests: non-maskable (NMI), fast interrupt request (FIRQ) and interrupt request (IRQ). An interrupt request causes an FIRQ interrupt on the microprocessor if:

i. Interrupt request input IRx is asserted by one or more of the peripherals connected to it -and -
ii. Bit x in the ISR is set.

An interrupt request that has been asserted on a selected IR input will result in a Fast Interrupt ReQuest (FIRQ) to the microprocessor -- the 6809 Interrupt Request (IRQ) and Non-Maskable Interrupt (NMI) inputs are never raised.

3.1.2. Interrupt Select Register (ISR)

Address: X'FFB8'

Reset State:

The ISR contains X'00'. Thus, all interrupts are disabled.

Description:

8-bit read/write.

A 0 in bit x of the ISR means that an asserted interrupt request input on IRx will not cause an FIRQ interrupt on the microprocessor.

A 1 in bit x of the ISR means that an asserted interrupt request input on IRx will cause a processor to take an FIRQ interrupt.

Programming Notes:

(1) FIRQ is not latched in either the PIC or the processor. Hence, clearing the ISR will remove an FIRQ request. However, interrupt request inputs are sometimes latched in the peripherals. (This depends on the device.) Consequently, setting ISR bit x to 1 may cause an immediate FIRQ request. Therefore, an interrupt handling routine must remove the asserted interrupt request, either by clearing the corresponding bit of the ISR or by resetting the interrupting device, prior to re-enabling FIRQ interrupts.

3.1.3. Interrupt Base Register (IBR)

Address: X'FEB9'

Reset State:

The IBR contains X'00'.

Description:

8-bit write-only.

The IBR contains the high-order byte of the address of the Interrupt Table (ITBL), described below. The low-order byte of the ITBL address is always taken as X'00'. When the processor acknowledges an interrupt[1] it fetches a new PC from the ITBL. The high-order byte of the new PC is fetched from byte 0 of the ITBL. The location of the low-order byte in the ITBL is determined by the type and priority of the interrupt according to the following table in Figure 6.

The low order byte fetched is always the one corresponding to highest priority IR input request that is asserted and enabled when the low-order byte of the

---

[1] In response to any of the following: Reset, SWI, FIRQ, NMI or IRQ.

```
IBR----> +--------+
+ X'0':  | NPC hi |
+ X'1':  |  SWI3  |
+ X'2':  |  SWI2  |
+ X'3':  |  FIRQ  |   (default)
+ X'4':  |  IRQ   |   (unused)
+ X'5':  |  SWI1  |
+ X'6':  |  NMI   |   (unused)
+ X'7':  |  Reset |
+ X'8':  |  IR0   |
    .        .
    .        .
    .        .
+ X'F':  |  IR7   |
         +--------+
```

Figure 6 --ITBL and Interrupt Priorities

new PC is actually fetched. This IR input may not be the one that actually caused FIRQ to be asserted. A higher-priority request signal (that is enabled) may have occurred, or in the case of shared peripherals, the other processor may have reset the peripheral since it signalled the interrupt. In fact, it is possible that all interrupt requests have disappeared before the new PC low byte is fetched; in this case, the byte is fetched from the default FIRQ location (offset X'3').

Programming Notes:

(1) Since the low order byte of the ITBL address is always taken as X'00' the ITBL must be aligned on a 256-byte boundary.

(2) Since the IBR is cleared on Reset, the ITBL will initially be in ROM, at locations X'0000' - X'000F'.

(3) More than one peripheral may be assigned the same interrupt request priority. Therefore, an interrupt handler may have to poll the various status registers to determine the source of the interrupt to which it should respond.

3.2. Read Only Meomory (ROM)

Address: X'0000' - X'4000'

Reset State:

Contains CLONE interface unit initialization routine.

Description:

4K bytes of read-only memory.

The ROM contains a program to facilitate loading a contention protocol into a CLONE interface unit.

Programming Notes:

(1) Since the IBR is cleared on Reset, the ITBL will initially be in ROM, at locations X'0000' - X'000F'.

3.3. Address Trap Unit (ATU)

3.3.1. Overview

The ATU can be in either monitor mode or passive mode. When in monitor mode, an interrupt request input is asserted whenever an address generated by the processor matches the contents of the Address Trap Register (ATR)*. A reference to the high-order byte of the ATR places the ATU into passive mode and clears any pending interrupt request. A reference to the low-order byte of the ATR places the ATU into monitor mode.

Interrupt Priority: Switch selectable within interface unit.

---

*This includes addresses (of interrupt routines) that are generated by the PIC in response to an asserted interrupt request.

3.3.2. Address Trap Register (ATR)

Address: X'FFBA'-X'FFBB'

Reset State:

ATR contains 0 and is in passive mode.

Description:

16 bit read/write.

Reference to the high-order byte of this register places the ATU into passive mode and clears any pending interrupt request input from this peripheral. Reference to the low-order byte of this register places the ATU into monitor mode. While in this mode, an interrupt request input is asserted if a valid address on the bus subsequently matches the contents of the ATR.

Programming Notes:

(1) The ATU can be used to write a very fast receive loop, as follows. (A description of the bits in the Receiver Status Register, RSR, appears in Section 4.1.4.2)

```
                DECLARE buffer[0 .. maxsize] of BIT(16);
                ATR := maxsize;
                i := 0;
    waitdata:   IF NOT RSR(r_done) THEN GOTO waitdata;
                buffer[i] := RDR;
                i := i+1;
                GOTO waitdata;
                . . .
       ON address trap OR ether-monitor-interrupt BEGIN;
                ... the entire message has placed in buffer[0] ... buffer[i].
```

Similarly, the ATU can be used to terminate the transmitting loop, as is shown in the following. (A description of the bits in the Transmitter Status Register, XSR, appears in Section 4.1.3.2)

```
              DECLARE buffer[0 .. maxsize] of BIT(16);
              ATR := maxsize;
              i := 0;
   waitxmit:  IF NOT XSR(x_rdy) THEN GOTO waitxmit;
              XDR := buffer[i];
              i := i+1;
              GOTO waitxmit;
              . . .
      ON address trap OR transmitter-interrupt BEGIN;
              x_on := false;
              ... the entire message has been sent, or a collision
                  occurred.
```

(2) The processor does 16 bit writes to this register by writing two 8 bit bytes in succession. First, the high order byte of the ATR is referenced, then the low order byte. Consequently, the ATU will always be in passive mode during the time a new value is being written into the ATR.

3.4. Interval Timer (IT)

3.4.1. Overview

The Interval Timer consists of an 8-bit read/write Interval Timer Register (ITR) that counts down at a rate of 0.5 MHz. When the ITR value reaches 0 an interrupt request is asserted and latched, and counting stops. Writing to the ITR clears any asserted interrupt request.

Interrupt Priority: Switch selectable within interface unit.

3.4.2. Interval Timer Register (ITR):

Address: X'FFBC'

Reset State:

ITR contains 0. No interrupt request is pending.

Description:

8 bit read/write.

When the ITR is non-zero it counts down at a rate of 0.5 MHz. An interrupt request is latched when the ITR counts down to 0. A write reference to the ITR clears any pending interrupt request. In particular, writing 0 to the ITR stops the timer from counting, clears any pending interrupt request, and does not cause a new interrupt request to be generated. Reading the ITR has no effect on the state of the IT.

4. Shared Bus Unit

4.1. Ether Transceiver (XCVR)

4.1.1. Overview

The Transceiver can be viewed as three separate logical components: a _transmitter_, a _receiver_ and an _ether monitor_. Associated with each of these components is a (read-only) status register and a (write-only) control register. Each component is assigned a separate interrupt request input. Interrupt requests that are asserted by any of these components are latched. An interrupt request can be cleared by writing into the control register of the component of the XCVR originating the interrupt.

4.1.2. Ether Monitor

4.1.2.1. Overview

At any time, the ether -- a coaxial cable used for communication between CLONE interface units -- is in one of three states: _busy_, _idle_, or _jammed_. The ether is _busy_ if a message transmission is in progress, _jammed_ if some CLONE interface unit is broadcasting a _jam_ signal, and _idle_ if no message transmission or jam is in progress. The ether monitor allows a CLONE interface unit to approximate the state of the ether. Because of the finite signal propogation time through the ether, the ether monitor only reports a past state of the ether. Moreover, there will exist times when the monitor reports the ether to be in no state. This occurs because some time interval must elapse after a CLONE interface unit starts receiving a message transmission, before the receiver circuitry can ascertain that indeed this is a message, and not noise on an idle ether.

The ether monitor consists of an Ether Status Register (ESR) and an Ether Monitor Control Register (EMCR). The EMCR is used to specify which changes in the contents of the ESR cause an interrupt request input to be asserted. The ESR can be read to determine an approximation of the ether state.

Interrupt Priority: Switch selectable within interface unit

4.1.2.2. Ether Status Register (ESR)

Address: X'FFC0'

Reset State:

Determined by state of the ether at the time of Reset.

Description:

8-bit read-only.

The contents of this register reflect an approximation of the state of the ether as follows:

Bit	Name	Meaning
0	e_busy	Message transmission is in progress on the ether.
1	e_idle	Ether is idle.
2	e_jammed	Ether is being jammed.
3-7		Undefined.

At most one of these bits will be 1 at any time. For brief intervals, all bits might be zero.

A change in value from 0 to 1 of any of these bits can cause an interrupt input request to be asserted, if so specified in the EMCR.

4.1.2.3. Ether Monitor Control Register (EMCR):

Address: X'FFC3'

Reset State:

Cleared on Reset. No Ether Monitor interrupt request inputs asserted.

Description:

8-bit write-only.

Bits 0-2 of the EMCR determine which ether states changes cause interrupt request inputs to be asserted, as follows:

Bit	Name	Interrupt is asserted
0	intrpt_busy	if message transmission is started.
1	intrpt_idle	if ether becomes idle.
2	intrpt_jam	if ether becomes jammed.
3-7	*	undefined.

Thus, when bit x of the EMCR is 1, a 1 level in bit x of the ESR causes an ether monitor interrupt request asserted. The request is latched. Writing into the EMCR clears any pending ether monitor interrupt requests. If a value is written into the EMCR and bit x in both EMCR and ESR are 1, then an ether monitor interrupt request input will be immediately asserted.

4.1.3. Transmitter

4.1.3.1. Overview

The Transmitter encodes data and places it on the ether. Optionally, it computes a CRC code on the data being sent, and appends this to the end of transmission. Operation of the transmitter is controlled by two registers: the Transmitter Status Register (XSR) and the Transmitter Control Register (XCR). Data to be transmitted is written into the Transmitter Data Register (XDR).

Interrupt Priority: Switch selectable within interface unit

4.1.3.2. Transmitter Status Register (XSR):

Address: X'FFC1'

Reset State:

XSR contains X'01'.

Description:

8-bit read-only.

Bits of this register reflect the current state of the transmitter.

Bit	Name	Latched?	Meaning
0	x_rdy	no	Data may be written into XDR.
1	x_orun	yes	Overrun -- data was written in XDR when x_rdy was 0.
2	x_uflo	yes	Underflow -- required data was not available in XDR.
3	x_coll	yes	Collision was detected while transmitting.
4-7	*	*	Undefined.

Overflow, underflow and collision are error conditions, and result in the immediate termination of any transmission in progress. Whenever one of these conditions occur, the corresponding bit in the XSR is set (x_orun, x_uflo or x_coll) amd an interrupt request input is asserted. Writing X'01' to the XCR clears any asserted interrupt request input, and resets the x_uflo, x_orun and x_coll bits to 0.

4.1.3.3. Transmitter Control Register (XCR)

Address: X'FFC4'

Reset State:

Cleared on Reset.

Description:

8-bit write-only.

The XCR controls operation of the transmitter, as follows.

Bit	Name	Meaning
0	x_on	Start message transmission.
1	x_crc	CRC should be sent at end of transmission.
2	x_jam	Transmit jam signal on the ether.
3-7	*	Undefined.

Writing a 1 in x_on resets the Transmitter. This involves the following:

i. XDR is reset.
ii. CRC value is reset.
iii. x_orun set to 0.
iv. x_uflo set to 0.
v. x_coll set to 0.

After it has been reset, the transmitter waits until a word of data is written into XDR. This data is broadcast onto the ether, and computation of a CRC value is begun.

XDR must then be filled with a new word within 18 μ-sec after x_rdy becomes 1, until the end of the transmission. The end of a transmission is signified by writing no word into XDR and clearing x_on within 18 μ-sec after x_rdy becomes 1.

The x_rdy bit (in XSR) is used for synchronizing the deposit of data into XDR with the actual transmission of that data. Data may be written into XDR only when x_rdy is 1. Writing into XDR when x_rdy is 0 will result in setting x_orun and an interrupt request input is asserted. If after x_rdy becomes 1, the processor does not write into XDR or clear x_on within 18 μ-sec, x_uflo will be set.

When the x_on bit is cleared:

(i) if x_crc is 0, transmission stops immediately after sending the last data word.
(ii) if x_crc is 1, transmission stops after sending the last data word and then the accumulated CRC value.

Setting the x_jam bit to 1 causes the transmitter to stop transmitting immediately, and a jam signal is broadcast on the ether. The CRC value is not transmitted, regardless of the setting of x_crc. The jam signal is braodcast until the x_jam bit is cleared. However, a hardware timeout mechanism prevents the transmitter from jamming the ether for more than one second.

4.1.3.4. Transmitter Data Register (XDR)

Address: X'FFC6'-X'FFC7'

Reset State: Undefined.

Description:

16-bit write-only.

Data written into XDR is transmitted on the ether. Data may be written into XDR only if x_on (in XCR) and x_rdy (in XSR) are both 1.

x_orun is set (overrun error) if an attempt is made to write into XDR when x_rdy is 0.
x_uflo is set (underflow error) if more than 18 μ-sec elapse after x_rdy becomes 1 and the processor neither writes into XDR or clears x_on.

4.1.4. Receiver

4.1.4.1. Overview

The Receiver receives and decodes transmissions from the ether. It also computes a CRC for each message as it is received. Operation of the receiver is controlled by the Receiver Control Register (RCR) and the Reciever Status Register (RSR). Data is read from the Receiver Data Register (RDR).

Interrupt Priority: Switch selectable within interface unit

4.1.4.2. Receiver Status Register (RSR)

Address: X'FFC2'

Reset State:

RSR contains 0.

Description:

8-bit read-only.

Bits of this register reflect the current status of the receiver, as follows.

Bit	Name	Latched?	Meaning
0	r_valid	no	RDR contains valid data.
1	r_uflo	yes	Underflow -- attempt to read RDR when r_done = 0.
2	r_orun	yes	Overrun -- valid data in RDR was lost.
3	r_perr	yes	Parity error -- data word in RDR has a parity error.
4	r_crc	yes	The accumulated output of the CRC checker. 0 ==> no CRC error 1 ==> CRC error found
5-7	*	*	Undefined.

r_valid is set to 1 whenever a data word is placed by the receiver into RDR. Reading RDR resets the r_valid bit. RDR retains its value until the next word is decoded from the ether and placed in RDR.

An interrupt request input is asserted whenever an error condition (underflow, overrun, parity error) is detected. The corresponding bit in the RSR (r_uflo, r_orun or r_rperr) is set. A subsequent reference to the RCR clears r_uflo, r_orun, r_perr and any asserted interrupt request input from the receiver .

The r_crc bit saves the output of the CRC checker from the last time a data word was transferred to RDR by the receiver. Valid CRC is indicated by a 0 in this bit after reception of a complete message.

4.1.4.3. Receiver Control Register (RCR)

Address: X'FFC5'

Reset State:

Cleared on Reset.

Description:

8-bit write-only.

The RCR is used to control operation of the receiver, as follows.

Bit	Name	Meaning
0	r_on	Values in RSR are meaningful.
1-7	*	Undefined.

An attempt to write to RCR resets the receiver. This involves the following.

i. RDR is reset.
ii. RSR is reset.

4.1.4.4. Receiver Data Register (RDR)

Address: X'FFC6'-X'FFC7'

Reset State:

RDR contains an undefined value after Reset.

Description:

16-bit read-only.

Data received is placed into the RDR. The RDR may be read only if r_valid is 1.

r_uflo is set (underflow error) if an attempt is made to read RDR when r_valid is 0.

r_orun is set (overrun error) if the processor does not read RDR within 18 μ-sec after r_valid is set, except for the last word of the message.

4.2. Host Processor Interface (HPI)

4.2.1. Quad Serial Interface (QSI)

4.2.1.1. Overview

The Quad Serial Interface implements four RS-232 ports to facilitate connection of up to four hosts to a CLONE interface unit. The baud rate at which all four ports will operate can be set by a switch inside the CLONE interface. Associated with each port is a a Control [illegible] (QCR), a Status Register (QSR), a Transmit Data Register (QTDR), and a Receive Data Register (QRDR),

Interrupt Priority: Switch selectable from within interface unit

4.2.1.2. Control Register (QCR)

Addresses: X'FFD2', X'FFD4', X'FFD6', X'FFD8'

Reset State:

QCR contains an undefined value after reset.

Description:

8-bit write-only.

A QCR controls the operation of its associated RS-232 port, as follows:

Bit	Meaning
0,1	Baud rate control and port reset.
2-4	Word select bits (number of bits, parity, number of stop bits).
5,6	Transmitter control bits.
7	Receiver interrupt request select bit.

Bits 0 and 1 can be used to change the baud rate at which communication with the host occurs. The "standard baud rate" referred to below is switch selectable from within the interface box. The following encoding is used:

Bit 0	Bit 1	Function
0	0	Do not use.
0	1	Standard baud.
1	0	Standard baud/16.
1	1	Master reset.

Setting these bits has an effect only after a master reset has been issued.

Bits 2, 3, and 4 are used to select the number of data and stop bits, and the parity with which communication with the connected host occurs. The following encoding is used:

Bit 2	Bit 3	Bit 4	Meaning
0	0	0	7 data bits; even parity; 2 stop bits.
0	0	1	8 data bits; no parity; 2 stop bits.
0	1	0	7 data bits; even parity; 1 stop bit.
0	1	1	8 data bits; even parity; 1 stop bit.
1	0	0	7 data bits; odd parity; 2 stop bits.
1	0	1	8 data bits; no parity; 1 stop bit.
1	1	0	7 data bits; odd parity; 1 stop bit.
1	1	1	8 data bits; odd parity; 1 stop bit.

Bits 5 and 6 are used to control the action taken when the QTDR empites. The following encoding is used.

Bit 5	Bit 6	Meaning
0	0	Transmitter: active; interrupt request disabled.
0	1	Transmitter: off; interrupt request disabled.
1	0	Transmitter: active; interrupt request enabled.
1	1	Transmitter: active; interrupt request disabled and a Break Level (space) is transmitted.

If bit 7 is set, a quad receiver interrupt request is signalled when QRDR becomes full or an overrun occurs. *** QSR

4.2.1.3. Status Register (QSR)

Addresses: X'FFD2', X'FFD4', X'FFD6', X'FFD8'

Reset State:

QSR is cleared on reset.

Description:

8-bit read-only.

The status of the RS-232 port is reflected in this register, as follows:

Bit	Name	Latched?	Meaning
0	qr_valid	no	QRDR contains unread and valid data.
1	tx_rdy	no	Data may be stored into QTDR.
2	qr_cd	no	Data Carrier Detect. If 0 then there is a logical connection to the host.
3	qr_cts	no	Clear to Send. If 0 then it is OK to send to host.
4	qr_frm_err	yes	Framing error in last transfer.
5	qr_orun	yes	Overrun; valid data in QRDR was lost.
6	qr_p_err	yes	Parity error; data in QRDR has a parity error
7	qr_int	yes	An interrupt request is being signalled by this port.

Programming Notes:

(1) Bits 2 and 3 will rarely (if ever) be useful.

(2) If a framing error occurs, it means that the last received character is improperly framed by start and stop bits.

4.2.1.4. Transmit Data Register (QTDR)

Addresses: X'FFD3', X'FFD5', X'FFD7', X'FFD9'

Reset State:

QTDR contains an undefined value after reset.

Description:

8-bit write-only.

Data written into QTDR is transferred to the host connected to the port. Data may be written into QTDR only if tx_rdy in the corressponding QSR is set. If data is written into QTDR while tx_rdy is not set, the old value of QTDR is lost.

4.2.1.5. Receive Data Register (QRDR)

Addresses: X'FFD3', X'FFD5', X'FFD7', X'FFD9'

Reset State:

QRDR contains an undefined value after reset.

Description:

8-bit read-only.

Data from the connected host is placed into QRDR. The contents of this register may be read at any time. If qr_valid is set, then the present contents of QRDR have never been read. If new data is transferred into QRDR by the host before QRDR is read, an overrun occurs, and qr_orun will be set.

4.2.2. Parallel Interface (PI)

4.2.2.1. Overview

The Parallel Interface allows 16-bit parallel data transfers between a host and a CLONE interface unit. It was designed for connection to a Digital Equipment Corporation DR11-B DMA parallel interface, but should be suitable for use with other, similar devices. The description below assumes the host connection is to a DR11-B. We assume the reader is familiar with the operation of this device, as described in DEC publication EK-DR11B-TM-004.

The Parallel Interface consists of three 8-bit parallel input ports and three 8-bit parallel output ports. The input ports receive data and status information from the host. They are designated Input Data Low (PIIDL), Input Data High (PIIDH) and STatus (PIST). The output ports send data and control information to the host. They are designated Output Data Low (PIODL), Output Data High (PIODH) and CoMmanD (PICMD).

Each 8-bit port comprises three registers: a Data Buffer (DB), a Data Direction register (DD), and a Control register (C). Thus, the complete Parallel Interface consists of 18 registers, organized as follows.

```
+-------+-------+  +-------+-------+  +-------+-------+
|PIIDLDB|PIODLDB|  |PIIDHDB|PIODHDB|  |PISTDB |PICMDDB|
+-------+-------+  +-------+-------+  +-------+-------+
|PIIDLDD|PIODLDD|  |PIIDHDD|PIODHDD|  |PISTDD |PICMDDD|
+-------+-------+  +-------+-------+  +-------+-------+
|PIIDLC |PIODLC |  |PIIDHC |PIODHC |  |PISTC  |PICMDC |
+-------+-------+  +-------+-------+  +-------+-------+

  PIIDL   PIODL      PIIDH   PIODH      PIST    PICMD
```

This register configuration directly reflects the organization of the 6520 integrated circuits from which the CLONE parallel interface is built. Users wishing to connect CLONE interfaces to devices other than DEC DR11-B's can consult the manufacturer's documentation for a complete description of the capabilities of these integrated circuits.

Interrupt Priority:

Three separate conditions in the parallel interface generate interrupt requests: GO, END CYCLE, and READY. The GO and END CYCLE requests share a single priority level; the READY request is made at a different level. Both priority levels are jumper selectable.

4.2.2.2. Parallel Interface Input Data Low Data Buffer (PIIDLDB)

Address: X'FFD8', if PIIDLC bit 2 is 1

Reset State:

The value of PIIDLDB is undefined after reset.

Description:

8-bit read-only.

Reading this register returns the current value of bits 0 thru 7 of the host's DR11-B output data buffer (DRDB).

4.2.2.3. Parallel Interface Input Data Low Data Direction (PIIDLDD)

Address: X'FFD8', if PIIDLC bit 2 is 0

Reset State:

PIIDLDD is cleared on reset.

Description:

8-bit read/write.

Each bit of PIIDLDD determines whether the corresponding bit of PIIDLDB operates as an input (bit value = 0) or an output (bit value = 1). This register must contain X'00' for communication with a DR11-B.

4.2.2.4. Parallel Interface Input Data Low Control (PIIDLC)

Address: X'FFDA'

Reset State:

PIIDLC is cleared on reset.

Description:

8-bit read/write.

This register controls PIIDL as follows:

Bit	Meaning
0-1	Unused.
2	0 - PIIDLDD selected. 1 - PIIDLDB selected.
3-7	Unused.

4.2.2.5. Parallel Interface Output Data Low Data Buffer (PIODLDB)

Address: X'FFDC', if PIODLC bit 2 is 1

Reset State:

The value of PIODLDB is undefined after reset.

Description:

8-bit read/write.

The value in this register appears in bits 0 thru 7 of the host's DR11-B input data buffer (DRDB).

4.2.2.6. Parallel Interface Output Data Low Data Direction (PIODLDD)

Address: X'FFDC', if PIODLC bit 2 is 0

Reset State:

PIODLDD is cleared on reset.

Description:

8-bit read/write.

Each bit of PIODLDD determines whether the corresponding bit of PIODLDB operates as an input (bit value = 0) or an output (bit value = 1). This register must contain X'FF' for communication with a DR11-B.

4.2.2.7. Parallel Interface Output Data Low Control (PIODLC)

Address: X'FFDE'

Reset State:

PIODLC is cleared on reset.

Description:

8-bit read/write.

This register controls PIODL as follows:

Bit	Meaning
0-1	Unused.
2	0 - PIODLDD selected. 1 - PIODLDB selected.
3-7	Unused.

4.2.2.8. Parallel Interface Input Data High Data Buffer (PIIDHDB)

Address: X'FFD9', if PIIDHC bit 2 = 1

Reset State:

The value of PIIDHDB is undefined after reset.

Description:

8-bit read-only.

Reading this register returns the current value of bits 8 thru 15 of the host's DR11-B data buffer (DRDB). In addition, reading this register causes a CYCLE REQUEST signal pulse to be sent to the host DR11-B, initiating a DMA cycle (See programming notes below).

4.2.2.9. Parallel Interface Input Data High Data Direction (PIIDHDD)

Address: X'FFD9', if PIIDHC bit 2 is 0

Reset State:

PIIDLDD is cleared on reset.

Description:

8-bit read/write.

Each bit of PIIDHDD determines whether the corresponding bit of PIIDHDB operates as an input (bit value = 0) or an output (bit value = 1). This register must contain X'00' for communication with a DR11-B.

4.2.2.10. [illegible] Interface Input Data High Control (PIIDHC)

Address: X'FFDB'

Reset State:

PIIDLC is cleared on reset.

Description:

8-bit read/write.

This register controls PIIDH as follows:

Bit	Meaning
0-1	Unused.
2	0 - PIIDHDD selected. 1 - PIIDHDB selected.
3-5	Must be B'101' for use with DR11-B
6-7	Unused.

4.2.2.11. Parallel Interface Output Data High Data Buffer (PIODHDB)

Address: X'FFDD', if PIODHC bit 2 = 1

Reset State:

The value of PIODHDB is undefined after reset.

Description:

8-bit read/write.

The value in this register appears in bits 8 thru 15 of the host's DR11-B input data buffer (DRDB). Writing this register causes a CYCLE REQUEST signal pulse to be sent to the host DR11-B, initiating a DMA cycle (See programming notes below).

4.2.2.12. Parallel Interface Output Data High Data Direction (PIODHDD)

Address: X'FFDD', if PIODHC bit 2 is 0

Reset State:

PIODHDD is cleared on reset.

Description:

8-bit read/write.

Each bit of PIODHDD determines whether the corresponding bit of PIODHDB operates as an input (bit value = 0) or an output (bit value = 1). This register must contain X'FF' for communication with a DR11-B.

4.2.2.13. Parallel Interface Output Data High Control (PIODHC)

Address: X'FFDF'

Reset State:

PIIDLC is cleared on reset.

Description:

8-bit read/write.

This register controls PIODH as follows:

Bit	Meaning
0-1	Unused.
2	0 - PIODHDD selected. 1 - PIODHDB selected.
3-5	Must be B'101' for use with DR11-B
6-7	Unused.

4.2.2.14. Parallel Interface STatus Data Buffer (PISTDB)

Address: X'FFD4', if PISTC bit 2 is 1

Reset state:

The value of PISTDB is undefined after reset.

Description:

8-bit read-only.

Reading this register returns the values of certain control bits in the host DR11-B status register (DRST), as follows:

Bit	Name	DRST Name	Meaning
0	READY	READY	Cleared by DR11-B at the beginning of a DMA data transfer. Cleared at the end of the transfer, or when an error condition exists.
1	BUSY	BUSY	Set by the DR11-B to indicate that a (1-word) UNIBUS transfer sequence is in progress.
2	FNCT1	FNCT1	
3	FNCT2	FNCT2	All FNCT bits are under host program control.
4	FNCT3	FNCT3	
5-7			Unused.

4.2.2.15. Parallel Interface Status Data Direction (PISTDD)

Address: X'FFD4', if PISTC bit 2 is 0

Reset State:

PISTDD is cleared on reset.

Description:

8-bit read/write.

Each bit of PISTDD determines whether the corresponding bit of PISTDB operates as an input (bit value = 0) or an output (bit value = 1). This register must contain X'00' for communication with a DR11-B.

4.2.2.16. Parallel Interface Status Control (PISTC)

Address: X'FFD5'

Reset State:

PISTC is cleared on reset.

Description:

8-bit read/write.

This register controls PIST and enabling of GO and END CYCLE interrupts, as follows:

Bit	Meaning
0	1 - An interrupt is requested when a GO pulse is generated by the host DR11-B. 0 - GO interrupts disabled.
1	Must be 1 for communication with DR11-B.
2	1 - PISTDB selected. 0 - PISTDD selected.
3	1 - An interrupt is requested when an END CYCLE pulse is generated by the host DR11-B. 0 - END CYCLE interrupts disabled.
4	Must be 1 for communication with DR11-B.
5	Must be 0 for communication with DR11-B.
6	Latches END CYCLE pulses from the host DR11-B. Cleared whenever PISTC is read.
7	Latches GO pulses from the host DR11-B. Cleared whenever PISTC is read.

4.2.2.17. Parallel Interface CoMmanD Data Buffer (PICMDDB)

Address: X'FFD6', if PICMDC bit 2 is 1

Reset state:

The value of PICMDDB is undefined after reset.

Description:

8-bit read/write.

The PICMDDB is used to set control signals that must be supplied to the host DR11-B. Several of these signals appears in the host DR11-B status register (DRST).

Bit	Name	DRST Name	Meaning
0	XFERDIR		1 - DMA transfer direction is to the host. 0 - DMA transfer direction is to CLONE unit.
1	ATTN	ATTN	Used by CLONE interface unit to request an interrupt in the host CPU.
2	STAT1	STAT1	
3	STAT2	STAT2	All STAT bits appear as read-only bits in DRST.
4	STAT3	STAT3	
5-7			Unused.

4.2.2.18. Parallel Interface CoMmanD Data Direction (PICMDDD)

Address: X'FFD6', if PICMDC bit 2 is 0

Reset State:

PICMDDD is cleared on reset.

Description:

8-bit read/write.

Each bit of PICMDDD determines whether the corresponding bit of PICMDDB operates as an input (bit value = 0) or an output (bit value = 1). This register must contain X'FF' for communication with a DR11-B.

4.2.2.19. Parallel Interface CoMmanD Control (PICMDC)

Address: X'FFD7'

Reset State:

PICMDC is cleared on reset.

Description:

8-bit read/write.

This register controls PICMD and enabling of READY interrupts, as follows:

Bit	Meaning
0	1 - An interrupt is requested when the READY signal is asserted by the host DR11-B. 0 - READY interrupts disabled.
1	Must be 1 for communication with DR11-B.
2	1 - PICMDDB selected. 0 - PICMDDD selected.
3-6	Unused.
7	Latches the READY signal from the host DR11-B. Cleared whenever PISTC is read.

4.3. Load and Set Locations (L/S)

4.3.1. Overview

There are 8 Load-and-Set (L/S) locations. They are used to provide synchronization between the two processors. Each L/S is an 8-bit register that contains either X'00' or X'FF'. A read reference to an L/S returns its current value and sets the L/S to X'FF' as one indivisible action. A write reference to an L/S sets it to X'00'.

4.3.2. Technical Details

Addresses: X'FFC8', X'FFC9', X'FFCA', X'FFCB', X'FFCC', X'FFCD'

X'FFCE', X'FFCF'

Reset State:

Contains X'00'.

Description:

8-bit read/write.

Indivisibly set by read access, cleared by write access.

4.4. Interface ID (IIDR)

4.4.1. Overview

The IIDR is a 16-bit read-only register whose value is switch-selectable. This value allows each CLONE Interface unit to have a unique address. It is also used during intial protocol loading to determine whether a protocol is to be obtained from the connected host, of through the network.

4.4.2. Technical Details

Address: X'FFD0'-X'FFD1'

Reset State:

Value is switch-selected.

Description:

16-bit read-only.

By convention, the least significant 8 bits contain a unique identification for this CLONE Interface Unit. The most significant 8 bits allow the specification of options to the protocol running in the interface unit.

4.5. RAM

4.5.1. Overview

The RAM provides up to 16K of storage that is accessable to both PUs. No performance penalty is incurred by referring to this shared memory.

4.5.1.1. Technical Details

Address: X'4000'-X'8000'

Reset State:

The contents of the RAM are undefined after a reset.

Description

16 bit read/write.