Reference Manual

VL-7312
VL-7314
VL-73CT12
VL-73CT14

Dual/Quad RS-232 Serial Card for the STD Bus

VERSALOGIC CORPORATION

Model VL-7312 & VL-7314
Dual / Quad RS-232 Serial Card for the STD Bus

REFERENCE MANUAL

VersaLogic Corporation
3888 Stewart Rd.
Eugene, OR 97402

(503) 485-8575
Fax (503) 485-5712

Doc. Rev. 05/10/93

Table of Contents

Overview

This manual details the installation and operation of VersaLogic's VL-7312 and VL-7314 interface cards. The VL-7312 has two serial channels and the VL-7314 has four serial channels.

Introduction

The VL-7312 and VL-7314 cards provide two and four independent serial RS-232 I/O channels. They operate in asynchronous and synchronous modes, supporting both SDLC and HDLC loop communications. The RS-232 interfaces are jumper selectable for DCE or DTE operation, allowing direct connection to a wide variety of external equipment.

These card is available in standard (VL-7312/7314) and extended temperature (VL-73CT12/73CT14) versions. Throughout this manual "VL-7312" and "VL-7314" will be used to refer to both versions of these boards, unless specifically noted otherwise.

Features

- Two or four full-duplex communication channels.
- Jumper configurable for DTE or DCE operation.
- Four interrupt vectors per channel generated on different status conditions.
- Generates prioritized vectored interrupts from external sources.
- 8 and 10-bit addressability.
- IOEXP supported.
- Asynchronous and synchronous protocols (BISYNC, SDLC, HDLC, CCITT-X.25, and T1).
- Software programmable baud rates to 76.8K baud.
- Local loopback.
- Zilog 8530 type serial communication controller (SCC) chips.
- 6 MHz Z80 or 8 MHz 8088 compatible.
- 8088 or Z80 compatible interrupt timing.
- Extended temperature version available.
- Pro-Log 7312/7314 plug-in replacement.

Data Paths

The six major areas of the Serial Communication Controller (SCC) chips are:

- Transmitter
- Receiver
- Baud rate generator
- Digital phase-locked loop (DPLL)
- Clocking options
- Data encoding

All communication modes are established by programming the write registers. As data is received or transmitted, read register values may change, altering the direction of the data path. These changed values can promote software action or internal hardware action for further register changes.

Transmitter

The transmitter has an 8-bit Transmit Data register (WR8) loaded from the system CPU, and a Transmit Shift register loaded from either WR6, WR7, or the Transmit Data register. In byte-oriented modes, WR6 and WR7 can be programmed with sync characters. In Monosync mode, an 8-bit or 6-bit sync character is used (WR6), whereas a 16-bit sync character is used (WR6 and WR7) in Bisync mode. In bit-oriented synchronous modes, the flag contained in WR7 is loaded into the Transmit Shift register at the beginning and end of a message.

If asynchronous data is processed, WR6 and WR7 are not used and the Transmit Shift register is formatted with start and stop bits shifted out to the transmit multiplexer at the selected clock rate. Synchronous data (except SDLC / HDLC) is shifted to the CRC generator as well as to the transmit multiplexer.

SDLC / HDLC data is shifted to the CRC Generator and out through the zero insertion logic (which is disabled while the flags are being sent). A "0" is inserted in all address, control, information, and frame check fields following five contiguous "1"s in the data stream. The result of the CRC generator for SDLC data is also routed through the zero insertion logic and then to the transmit multiplexer.

Receiver

The receiver has a three deep 8-bit Data FIFO (paired with an 8-bit Error FIFO), and an 8-bit Shift register. This arrangement creates a 3-byte delay time, which allows the CPU time to service an interrupt at the beginning of a block of high-speed data. With each Receive Data FIFO, the Error FIFO stores parity and framing errors and other types of status information. The Error FIFO is readable in Read Register 1.

Incoming data is routed through one of the several paths depending on the mode and character length. In Asynchronous mode, the serial data enters the 3-bit delay if the character of seven or eight bits is selected. If a character length of five or six bits is selected, the data enters the Receive Shift register directly.

In synchronous modes, the data path is determined by the phase of the receive process currently in operation. Asynchronous receive operation begins with a hunt phase in which a bit pattern that matches the programmed sync characters (6-, 8-, or 16-bit is searched).

The incoming data then passes through the Sync register and is compared to a sync character stored in WR6 or WR7 (depending on which mode it is in). The Monosync mode matches the sync character programmed in WR7 and the character assembled in the Receive Sync register to establish synchronization.

Synchronization is achieved differently in the Bisync mode. Incoming data is shifted to the Receive Shift register while the next eight bits of the message are assembled in the Receive Sync register. If

these two characters match the programmed characters in WR6 and WR7, synchronization is established. Incoming data can then bypass the Receive Sync register and enter the 3-bit delay directly.

The SDLC mode of operation uses the Receive Sync register to monitor the receive data stream and to perform zero deletion when necessary; i.e., when five continuous "1"s are received, the sixth bit is inspected and deleted from the data stream if it is "0". The seventh bits is inspected only if the sixth bit equals "1". If the seventh bit is "0", a flag sequence has been received and the receiver is synchronized to that flag. If the seventh bit is a "1", an abort or an EOP (End Of Poll) is recognized, depending on the selection of either the normal SDLC mode or SDLC Loop mode.

The same path is taken by incoming data for both SDLC modes. The reformatted data enters the 3-bit delay and is transferred to the Receive Shift register. The SDLC receive operation begins in the hunt phase by attempting to match the assembled character in the Receive Shift register with the flag pattern in WR7. When the flag character is recognized, subsequent data is routed through the same path, regardless of character length.

Either the CRC-16 or CRC-SDLC cyclic redundancy check (CRC) polynomial can be used for both Monosync and Bisync modes, but only the CRC-SDLC polynomial is used for SDLC operation. The data path taken for each mode is also different. Bisync protocol is a byte-oriented operation that requires the CPU to decide whether or not a data character is to be included in CRC calculation. An 8-bit delay in all synchronous modes except SDLC is allowed for this process. In SDLC mode, all bytes are included in the CRC calculation.

Baud Rate Generator

Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8-bit, Time-Constant registers forming a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output that makes the output a square wave. On start-up, the flip-flop on the output is set High so that it starts in a known state, the value in the Time-Constant register is again loaded into the counter, and the counter begins counting down. When a count of zero is reached, the output of the baud rate generator toggles, an optional interrupt is generated, the value in the Time-Constant register is loaded into the counter, and the process starts over. The time constant can be changed at any time, but the new value does not take effect until the next load of the counter.

No attempt is made to synchronize the loading of a new time constant with the clock used to drive the generator. When the time constant is to be changed, the generator should be stopped by writing to an enable bit in WR14. This ensures the loading of a correct time constant. Page 4-31 shows time constants for several standard baud rates and the formula for determining the time constant for a given baud rate.

Digital Phased-Locked Loop (DPLL)

The SCC contains a digital phase-locked loop that can be used to recover clock information from a data stream with NRZI or FM coding. The DPLL is driven by a clock nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a receive clock for the data. This clock can then be used as the SCC receive clock, the transmit clock, or both.

Clocking Options

The SCC can select several clock sources for internal and external use. Write Register 11 is the Clock Mode Control register for both the receive and transmit clocks.

Write Register 11 also controls the output of the baud rate generator, and the DPLL output.

Data Encoding

The figure below illustrates the four encoding methods used by the SCC. In NRZ encoding, a "1" is represented as an RS-232 −V level and a "0" is represented as an RS-232 +V level. In NRZI encoding, a "1" is represented by no change in level and a "0" is represented by a change in level. In FM1 (more properly termed, biphase mark), a transition occurs at the beginning of every bit cell. A "1" is represented by an additional transition at the center of the bit cell and a "0" is represented by the absence of a transition at the center of the bit cell. In FM0 (more properly termed, biphase space), a transition occurs at the beginning of every bit cell. A "0" is represented by an additional transition at the center of the bit cell and a "1" is represented by the absence of a transition at the center of the bit cell.

In addition to these four methods, the SCC can be used to decode Manchester (biphase level) data using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is from RS-232 +V to RS-232 −V, the bit is "0". If the transition is from RS-232 −V to RS-232 +V, the bit is "1".

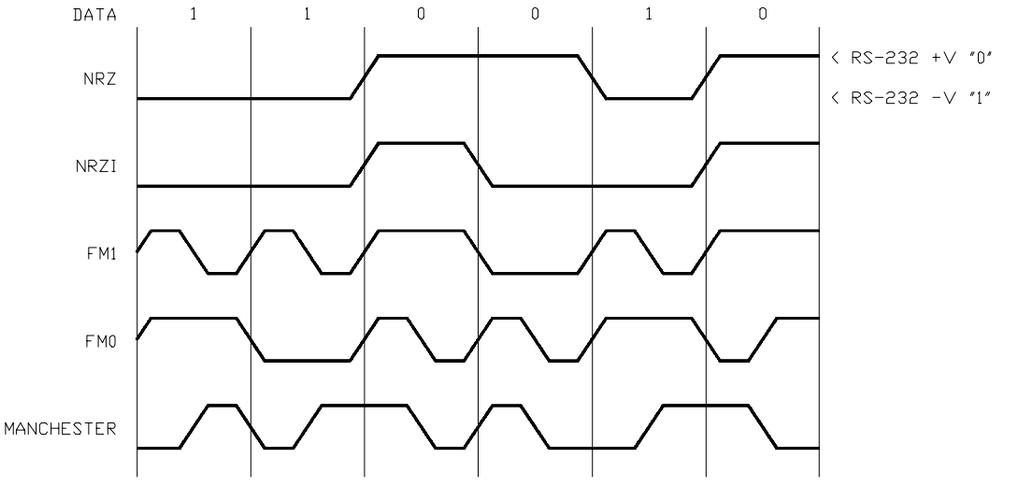


Figure 1-1. Data Encoding Methods

Data Communications Capabilities

The Serial Communication Controller (SCC) on the VL-7312 / VL-7314 handles all asynchronous, byte-oriented synchronous, and bit-oriented synchronous modes of operation.

Asynchronous

The figure below represents a typical asynchronous message format using one start bit, seven data bits, one parity bit, and one stop bit. A start bit is an RS-232 −V to RS-232 +V transition detected by an asynchronous receiver and is actually an information bit notifying the receiver of an incoming character.

The start bit also initiates a clock circuit to provide latching pulses during expected data bit intervals. The parity bit is provided for error checking. The parity bit is calculated in both the receiver and the transmitter; the two results are compared to ensure that the expected and the actual bit values match.

The stop bit returns the message unit to the quiescent marking state; i.e., a constant RS-232 −V state condition lasts until the next start bit indicates an incoming data byte. During reception, the start and stop bits are stripped away and checked for errors, leaving only the working data for CPU interaction.

The number of selected bits for each asynchronous function may differ between the transmitter and the receiver.

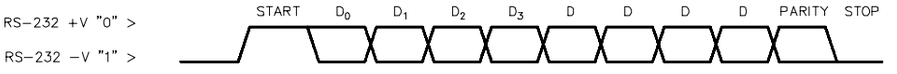


Figure 1-2. Asynchronous Character Format

Monosync Mode

Monosync and Bisync modes require clocking information to be transmitted along with the data either by a method of encoding data that contains clocking information, or by a modem that encodes or decodes clock information in the modulation process.

Start and stop bits are not required in synchronous modes. All bits are used to transmit data. This eliminates the "waste" characteristic of asynchronous communication. The figure below shows the character format for synchronous transmission. For example bits 1-8 might be one character and bits 9-13 part of another character; or bit 1 might be part of one character, bits 2-9 part of a second character, and bits 10-13 part of a third character. The framing (where each character begins) of each character is accomplished by defining a synchronization character, commonly called a "sync character".

The CPU places the receiver in Hunt mode whenever transmission begins (or whenever a data dropout has occurred and the hardware determines that resynchronization is necessary). In Hunt mode, the receiver shifts a bit into the Receive Shift register and compares the contents of the Receive Shift register with the sync character (stored in another register), repeating the process bit-by-bit until a match occurs. When a match occurs, the receiver begins transferring bytes to the Receive FIFO.

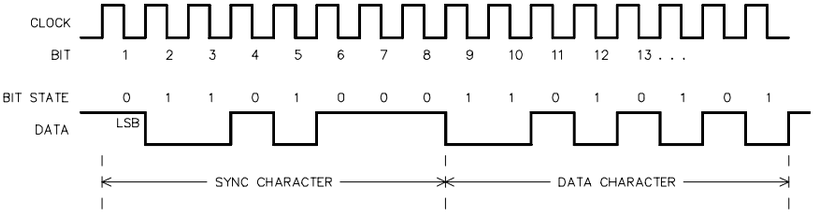


Figure 1-3. Monosync Data Character Format

Bisynchronous Mode

The Bisync mode of operation (see figure below) is similar to the Monosync mode, except that two sync characters are provided instead of one. Bisync attempts a more structured approach to synchronization through the use of special characters as message "headers" or "trailers."

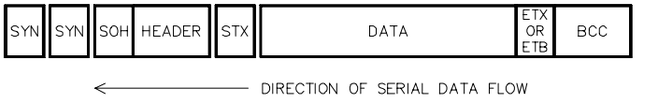


Figure 1-4. Bisynchronous Message Format

External Sync Mode

External Sync Mode (see figure below) eliminates the use of sync characters in the serial data stream by providing an external sync signal to mark the beginning of a data field; i.e., an external input applied to connector J1 waits for an active state change to indicate the beginning of an information field.

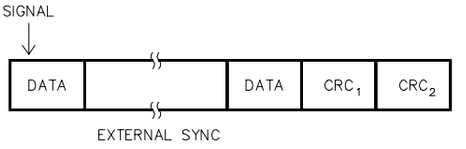


Figure 1-5. External Sync Format

SDLC Mode

Synchronous Data Link Control mode (SDLC) uses synchronization characters similar to Bisync and Monosync modes (such as flags and pad characters), but it is a bit-oriented protocol instead of byte-oriented.

Any data communication link involves at least two stations. The station that is responsible for the data link and issues the commands to control the link is called the "primary station". The other station is a "secondary station". Not all information transfers need to be initiated by a primary station. In SDLC mode, a secondary station can be the initiator.

The basic format for SDLC is a "frame" (see figure below). The information field is not restricted in format or content and can be of any reasonable length (including zero). Its maximum length is that which can be expected to arrive at the receiver error-free most of the time. Hence, the determination of maximum length is a function of communication channel error rates.

The two flags that delineate the SDLC frame serve as reference points when positioning the address and control fields, and they initiate the transmission error check. The ending flag indicates to the receiving station that the 16 bits just received constitute the frame check. The ending flag could be followed by another frame, another flag, or an idle. This means that when two frames follow one another, the intervening flag may simultaneously be the ending flag of the first frame and the beginning flag of the next frame. Since the SDLC mode does not use characters of defined length, but rather works on a bit-by-bit basis, the 01111110 (7EH) flag can be recognized at any time.

To ensure that the flag is not sent accidentally, SDLC procedures require a binary "0" to be inserted by the transmitter after the transmission of five contiguous "1"s. The receiver then removes the "0" following a received succession of five "1"s. Inserted and removed "0s" are not included in the CRC calculation.

The address field is eight bits long and identifies the secondary station to which the commands or data from the primary station are being sent. The control field is eight bits long and is used to initiate all SDLC activities.

The SCC can also serve the High-level synchronization protocol, which is identical to SDLC except for differences in framing.

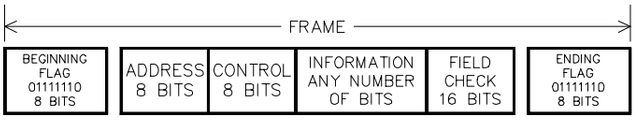


Figure 1-6. SDLC Message Format

SDLC Loop Mode

The SCC supports SDLC Loop mode in addition to normal SDLC. SDLC Loop mode is very similar to normal SDLC but is usually used in application where a point-to-point network is not appropriate (for example, point-of-sale terminals). In an SDLC Loop there is a primary station, called the controller, that manages the message traffic flow on the loop, and any number of secondary stations.

A secondary station in an SDLC loop is always listening to the messages being sent around the loop, and must pass these messages to the rest of the loop by retransmitting them with a one-bit time delay. The secondary station can only place its own message on the loop at specific times. The controller signals that the secondary stations may transmit messages by sending a special character, called an EOP (End Of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and thus is easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it first changes the last "1" of the EOP to a "0" before transmitting it. This turns the EOP into a Flag sequence. The secondary station now places its message on the loop and terminates its message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the upstream station by the same process. All secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop, except upon recognizing an EOP.

There are also restrictions as to when and how a secondary station physically becomes part of the loop. A secondary station that has just powered up must monitor the loop, without the one-bit time delay, until it recognizes an EOP. When an EOP is recognized the one-bit time delay is switched on. This does not disturb the loop, because the line is marking idle between the time the controller sends the EOP and the time that it receives the EOP back. The secondary station that has gone on-loop cannot place a message on the loop until the next time that an EOP is issued by the controller. A secondary station goes off-loop in a similar manner. When given a command to go off-loop, the secondary station waits until the next EOP to remove the one-bit time delay.

To operate the SCC in SDLC Loop mode, the SCC must first be programmed just as if normal SDLC were to be used. Loop mode is then selected by writing the appropriate control word in WR10. The SCC is now waiting for the EOP so that it can go on loop. While waiting for the EOP, the SCC ties TxD to RxD with only the internal gate delays in the signal path. When the first EOP is recognized by the SCC, the Break/Abort/EOP bit is set in RR0, generating an External/Status interrupt (if so enabled). At the same time, the On-Loop bit in RR10 is set to indicate that the SCC is indeed on-loop, and a one-bit time delay is inserted in the Txd to RxD patch.

The SCC is now on-loop but cannot transmit a message until a flag and the next EOP are received. The requirement that a flag be received ensures that the SCC cannot erroneously send messages until the controller ends the current polling sequence and starts another one.

A secondary station on the loop is prohibited from transmitting a message during a polling sequence unless it captures the line at the moment the EOP passes by. The SCC does this automatically. If the secondary station needs to transmit a message, the Go-Active-On-Poll bit in WR10 must be set. If this bit is set when the EOP is detected, the SCC changes the EOP to a flag and starts sending another flag. The EOP is reported in the Break/Abort/EOP bit in RR0 and the CPU should write its data bytes to the SCC, just as in normal SDLC frame transmission. When the frame is complete and CRC has been sent, the SCC closes with a flag and reverts to One-Bit-Delay mode. The last zero of the flag, along with the marking line echoed from RxD, form an EOP for secondary stations further down the loop. If the Go-Active-On-Poll bit is not set at the time the EOP passes by, the SCC cannot send a message until a flag (terminating the current polling sequence) and another EOP are received. While the SCC is actually transmitting a message, the loop-sending bit in RR10 is set.

If SDLC Loop is deselected, the SCC is designed to exit from the loop gracefully. When the mode is terminated by writing to WR10, the SCC waits until the next polling cycle to remove the one-bit time delay. If a polling cycle is in progress at the time the command is written, the SCC finishes sending any message that it may be transmitting, ends with an EOP, and disconnects TxD from RxD. If no message was in progress, the SCC immediately disconnects TxD from RxD. To ensure proper loop

operation after the SCC goes off the loop, and until the external relays take the SCC completely out of the loop, the SCC should be programmed for Mark idle instead of Flag idle. When the SCC goes off the loop, the On-Loop bit is reset.

The SCC allows the user the option of using NRZI in SDLC Loop mode by programming WR20 appropriately. With NRZI encoding, the outputs of secondary stations in the loop may be inverted from their inputs because of messages that they have transmitted. Removing the stations from the loop (removing the one-bit time delay) may cause problems further down the loop because of extraneous transitions on the line. The SCC avoids this problem by making transparent adjustments at the end of each frame it sends in response to an EOP. A response frame from the SCC is terminated by a flag and an EOP. Normally, the flag and the EOP share a zero, but if such sharing would cause RxD and TxD to be of opposite polarity after the EOP, the SCC adds another zero between the flag and the EOP. This causes an extra line transition so that RxD and TxD are identical after the EOP is sent. This extra zero is completely transparent because it only means that the flag and the EOP no longer share a zero. All that a proper loop exit needs, therefore, is the removal of the one-bit time delay.

Configuration

Jumper Options

Various options available on the VL-7312 / VL-7314 cards are selected using removable jumper plugs (shorting plugs). Features are selected or deselected by installing or removing the jumper plugs as noted. The terms "In" or "Jumpered" are used to indicate an installed plug: "Out" or "Open" are used to indicate a removed plug.

Figure 2-1 shows the jumper block locations on the VL-7312 and the VL-7314 cards. They indicate the position of the jumper plugs as shipped from the factory. The function of each jumper block is detailed in Figure 2-2.

VL-7312 / VL-7314 Jumper Block Locations

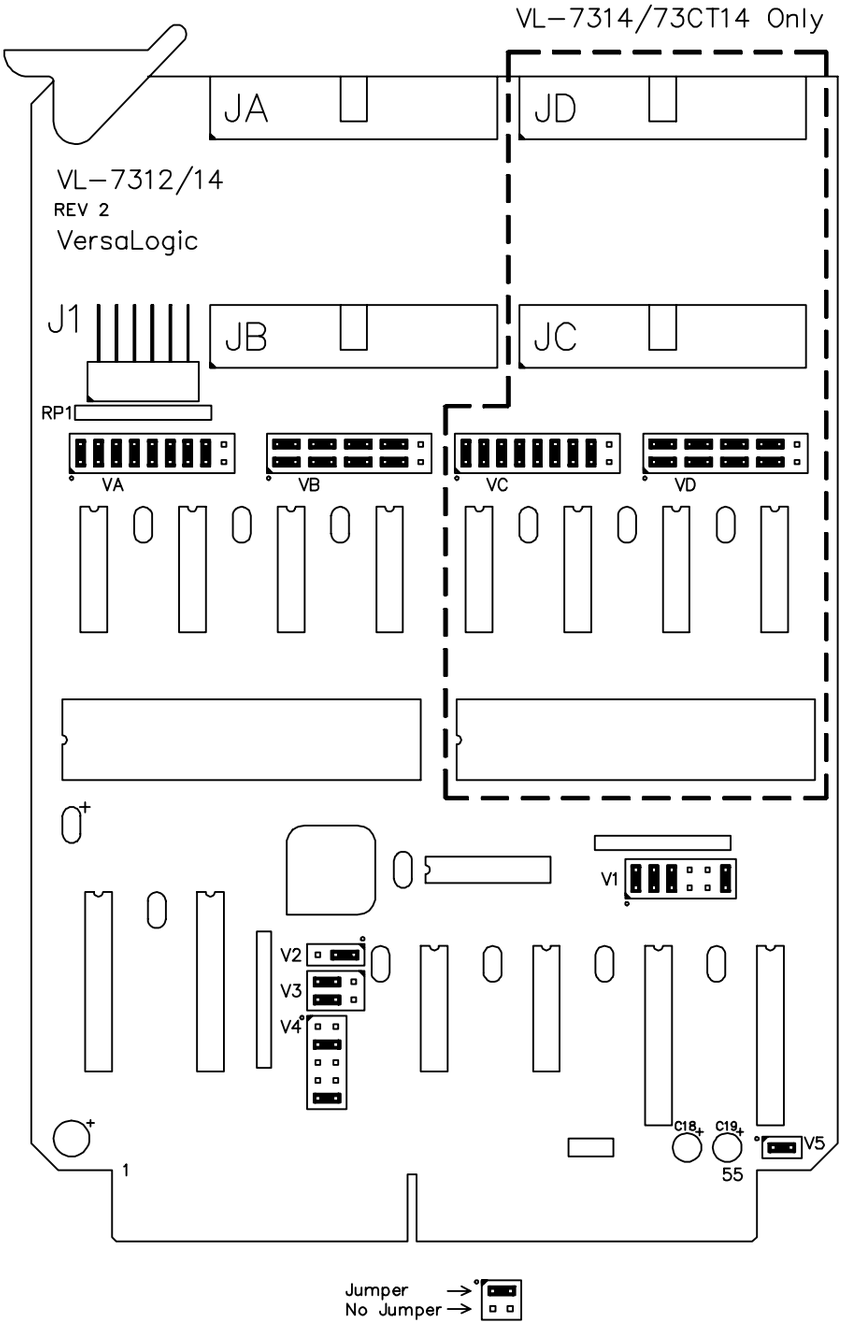


Figure 2-1. Jumper Block Locations for VL-7312/14

Jumper Block	Description	As Shipped	Page
VA	Channel A DCE / DTE configuration	DCE	2-10
VB	Channel B DCE / DTE configuration	DTE	2-10
VC	Channel C DCE / DTE configuration	DCE	2-10
VD	Channel D DCE / DTE configuration	DTE	2-10
$V1_a$	Interrupt vector enable In – Board responds to interrupt acknowledge cycle Out – Board ignores interrupt acknowledge cycle	In	2-7
$V1_b$	Channel C & D interrupt request interconnect In – Channel C & D interrupts merged with channels A & B on INTRQ* Out – Channel C & D interrupts only on INT2* (J1 pin 11)	In	2-7
$V1_c$	Interrupt request enable In – Enables interrupt requests via INTRQ* Out – Disables interrupt requests via INTRQ*	In	2-7
$V1_d$	CPU select In – STD-Z80 interrupt acknowledge cycle Out – STD-8088 interrupt acknowledge cycle	Out	2-9
$V1_e$	Advanced write In – Disable advanced write cycle for nonstandard STD 8088 timing Out – Enable advanced write cycle for STD 8088 timing	Out	2-9
$V1_f$	AUX GND connected to digital ground In – Connect AUX GND to digital ground Out – Separate AUX GND from digital ground	In	2-9
V2	IOEXP select a – Board responds to IOEXP high and low b – Board responds to IOEXP low None – Board responds to IOEXP high	a In, b Out } IOEXP high and low	2-6
$V3_{a-b}$	Address mode selector (8- or 10-bit decoding) a – A9 b – A8	a A9 Low, b A8 Low } 10-Bit decoding	2-4
$V4_{a-e}$	Board address (A3 – A7) a – A7 b – A6 c – A5 d – A4 e – A3	a Out, b In, c Out, d Out, e In } B0 Hex	2-4
V5	PCO/+3.3V Interconnect In – STD-80 Mode. PCO (P51) connected to on-board circuitry. Out – STD-32 Mode. PCO (P51) isolated.	In	2-11

Figure 2-2. Jumper Functions

Board Addressing

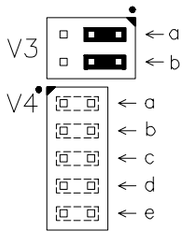
The VL-7312 / VL-7314 cards support both 8- and 10-bit I/O addressing. 8-bit addressing is used with most 8-bit processors (Z80, 8085, 6809, etc.) which provide 256 I/O addresses. 10-bit addressing can be used with 16-bit processors (i.e. 8088, 80188, etc.) to decode up to 1024 I/O port addresses.

Both 8- and 10-bit addressing can be extended (capacity doubled) using the IOEXP signal which is decoded on board.

As shipped the board is configured for 10-bit addressing with a board address of hex 0B0. The VL-7314 occupies eight consecutive I/O addresses (i.e. B0-B7). The VL-7312 also occupies eight consecutive addresses; four are used by the board, and four are not valid.

8-Bit Addressing

To configure the board for an 8-bit I/O address refer to the figure below. Use the table to select the jumpering for the appropriate upper and lower halves of the desired starting address (i.e. "3" and "0" = hex address 30).



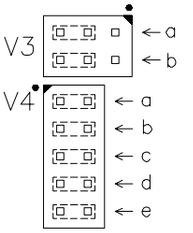
$V4_a$	$V4_b$	$V4_c$	$V4_d$	Upper Digit	$V4_e$	Lower Digit
X	X	X	X	0	X	0
X	X	X	–	1	–	8
X	X	–	X	2		
X	X	–	–	3		
X	–	X	X	4		
X	–	X	–	5		
X	–	–	X	6		
X	–	–	–	7		
–	X	X	X	8		
–	X	X	–	9		
–	X	–	X	A		
–	X	–	–	B		
–	–	X	X	C		
–	–	X	–	D		
–	–	–	X	E		
–	–	–	–	F		

x = Jumper installed.
– = Jumper removed.

Figure 2-3. 8-Bit Address Jumpers

10-Bit Addressing

To configure the board for a 10-bit I/O address refer to the figure below. Use the table to select the jumpering for the appropriate upper, middle, and lower hex digits of the desired address (i.e. "1" and "3" and "0" = hex address 130).



$V3_a$	$V3_b$	Upper Digit	$V4_a$	$V4_b$	$V4_c$	$V4_d$	Middle Digit	$V4_e$	Lower Digit
X	X	0	X	X	X	X	0	X	0
X	–	1	X	X	X	–	1	–	8
–	X	2	X	X	–	X	2		
–	–	3	X	X	–	–	3		
			X	–	X	X	4		
			X	–	X	–	5		
			X	–	–	X	6		
			X	–	–	–	7		
			–	X	X	X	8		
			–	X	X	–	9		
			–	X	–	X	A		
			–	X	–	–	B		
			–	–	X	X	C		
			–	–	X	–	D		
			–	–	–	X	E		
			–	–	–	–	F		

x = Jumper installed.
– = Jumper removed.

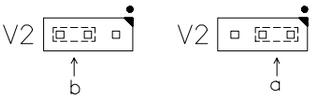
Figure 2-4. 10-Bit Address Jumpers

IOEXP Signal

The IOEXP (I/O expansion) signal on the STD Bus is normally used to select between two different I/O banks or maps. It can be used to double the number of available I/O addresses in the system (by selecting between two banks of I/O boards). The IOEXP signal is usually controlled by (or jumpered to ground on) the system CPU card.

A low IOEXP signal usually selects the standard or normal I/O map. A high IOEXP signal usually selects the secondary or alternate I/O map. Boards that ignore (or do not decode IOEXP) will appear in both I/O maps.

As shipped the IOEXP jumper is configured to ignore the IOEXP signal. The board will be addressed whether the IOEXP signal is high or low. It can be jumpered for two other modes as shown in the figure below.

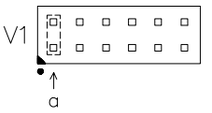


Jumper Block	Description	As Shipped
$V2_a$	Ignore IOEXP (enable high or low)	a – In
$V2_b$	Enable on IOEXP low	b – Out
None	Enable on IOEXP high (no jumpers)	

Figure 2-5. IOEXP Jumper

Interrupt Vector Enable

Each channel can be programmed to generate up to four independent interrupt vector "type codes" or Z80 restart instructions in response to various status conditions. Jumper $V1_a$ is used to enable the vector onto the STD Bus during the interrupt acknowledge cycle (INTAK*). To disable this feature, jumper $V1_a$ is removed.



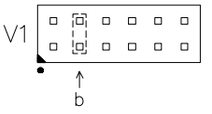
Jumper Block	Description	As Shipped
$V1_a$	Interrupt vector enable	In

Figure 2-6. Interrupt Vector Enable

Interrupt Request Interconnect

The interrupt request signals from each SCC appear separately at connector J1 (pins 9 and 11). By installing jumper $V1_b$ they can be interconnected, the resulting signal appearing at connector J1 pin 9.

This jumper (and jumper $V1_c$) must be installed to enable interrupts from SCC#2 (channel C / D) to the bus.



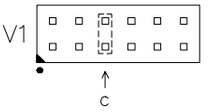
Jumper Block	Description	As Shipped
$V1_b$	Interrupt request interconnect	In

Figure 2-7. Interrupt Request Interconnect

Interrupt Request Enable

If the VL-7312 / VL-7314 is used in systems that implement the STD Bus serial or polled interrupts, jumper $V1_c$ connects the interrupt request signals of both SCCs to the STD Bus signal INTRQ*.

If your interrupt scheme requires interrupts to be handled off the bus, such as a parallel interrupt prioritization scheme, disconnect INTRQ* from the STD Bus by removing $V1_c$. The two SCC interrupt request signals can appear separately at connector J1 (pins 9 and 11) or they can be interconnected by installing jumper $V1_b$.



Jumper Block	Description	As Shipped
$V1_c$	Interrupt request enable	In

Figure 2-8. Interrupt Request Enable

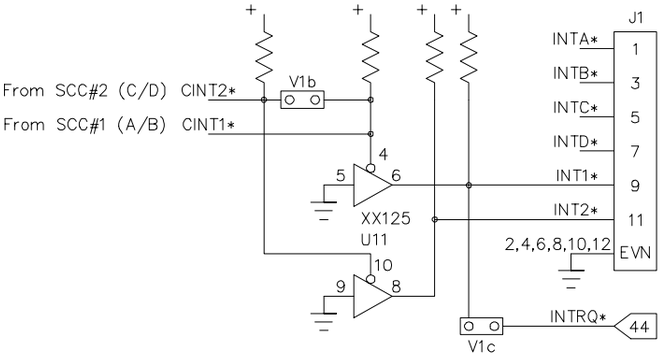


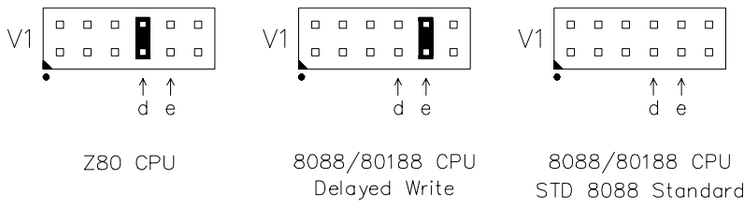
Figure 2-9. Interrupt Interconnect and Request Enable Circuitry

CPU Selection/Write Timing

Jumpers $V1_d$ and $V1_e$ are used to select between STD Z80 and STD 8088 Bus timing, allowing the board to operate properly with either Z80 or 8088/80188 type CPUs.

Jumper $V1_d$ selects between Z80 and 8088 processors, while jumper $V1_e$ selects between advanced (normal STD 8088) or delayed (nonstandard 8088) write signal timing.

As shipped, the board is configured to conform with STD 8088 timing specifications. It can be jumpered in any of three configurations as shown in the figure below.



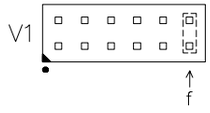
Jumper Block	Description	As Shipped
$V1_d$	Advanced write timing	$V1_d$ Out (STD 8088 timing)
$V1_e$	CPU select	$V1_e$ Out (8088 CPU)

Figure 2-10. CPU Selection / Write Timing

Ground Option

Jumper $V1_f$ connects the system digital (+5V) ground line to the "auxiliary" (±12V) ground line. If jumper $V1_f$ is removed these grounds must be connected together at the power supply or motherboard connection.

As shipped, this jumper is in. In most cases it will not need to be removed.



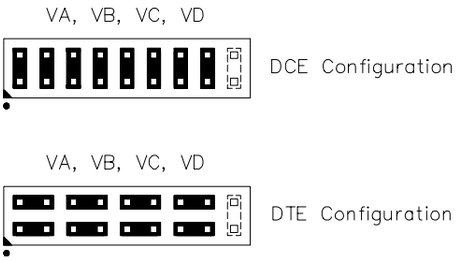
Jumper Block	Description	As Shipped
$V1_f$	Ground option	In (AUXGND tied to GND)

Figure 2-11. Ground Option

RS-232 Interface

Jumper blocks VA, VB, VC, and VD are used to configure each RS-232 port for data communications equipment (DCE) or data terminal equipment (DTE) pinout configuration.

The jumper blocks that control the RS-232 port configuration are shown below. When jumpered for DCE pinout, the port is configured to work directly with an IBM PC serial adapter or standard terminal.



Channel	Connector	As Shipped
Channel A	JA	DCE
Channel B	JB	DTE
Channel C[1]	JC	DCE
Channel D[1]	JD	DTE

[1] Available on VL-7314 only.

Figure 2-12. RS-232 DCE/DTE Jumpers

The jumpers in VA, VB, VC, and VD are grouped in four sections, each controlling a particular set of RS-232 signals.

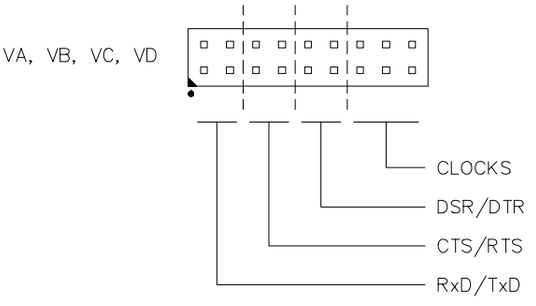


Figure 2-13. RS-232 DCE/DTE Jumper Groups

The figure below shows the relationship the DCE/DTE configuration jumpers have to the RS-232 signals. Numbers in parenthesis indicate standard RS-232-C pin numbers.

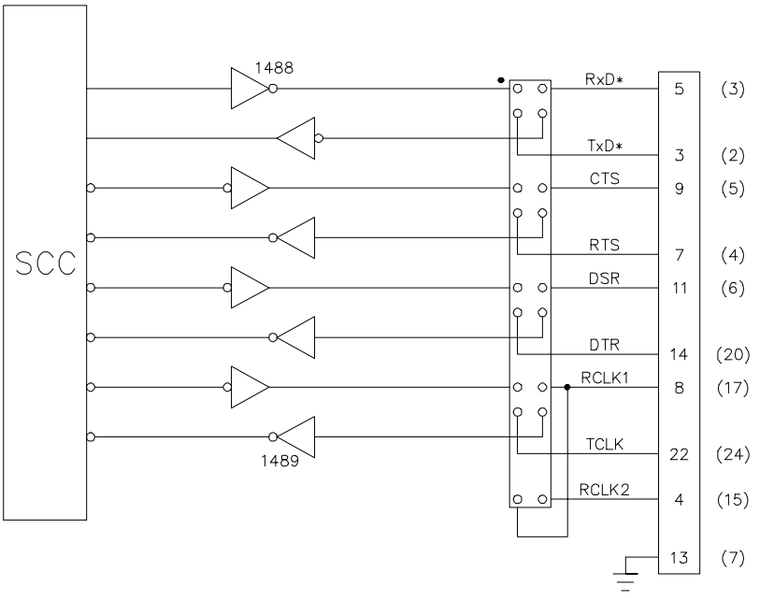


Figure 2-14. RS-232 Interface Circuitry

PCO/+3.3V Interconnect

If the VL-7312/VL-7314 is used in an STD-32 Bus, jumper V5 must be removed. If the card is operated in an STD-80 Bus with a PCI/PCO interrupt chain, jumper V5 must be inserted.

To maintain compatibility with STD-32 specifications which have changed the definition of the PCI (P52) and PCO (P51) signals into +3.3V power distribution lines, jumper V5 was added to protect the PCO output. When V5 is removed, the on-board circuitry which drives PCO is disconnected from the bus.

PCI and PCO are only used in STD-80 systems to support an interrupt priority chain. The VL-7312/VL-7314 does not use +3.3V.

Jumper Block	Description	As Shipped
V5	PCO/+3.3V Interconnect	In (STD-80 Mode)

Figure 2-15. PCO/+3.3V Interconnect

Installation

Handling

CAUTION: The VL-7312 / VL-7314 cards use chips which are sensitive to static electricity discharges. Normal precautions, such as discharging yourself, work stations, and tools to ground before touching the board should be taken whenever the board is handled.

The board should also be protected during shipment or storage by placing it in a conductive bag (such as the one it was received in) or by wrapping it in metal foil.

Installation

The VL-7312 / VL-7314 cards can be installed in any slot of an STD Bus card cage, and should only be used with other standard (TTL) type STD Bus boards. When using CMOS STD Bus CPU boards, use the VL-73CT12 / VL-73CT14.

Priority Chain

The VL-7312 / VL-7314 cards use the STD Bus priority interrupt chain signals PCO and PCI. Priority is based on slot position in the card rack with the highest priority on the right. (Note that some manufacturers make card racks with highest priority on the left.)

The VL-7312 / VL-7314 cards have a fixed priority for the various interrupt sources generated on board but, as a whole, the board is assigned higher or lower priority with respect to other cards by slot position. On VL-7314 cards, SCC #1 (channels A / B) is chained ahead of SCC #2 (channels C / D).

Cards that use the priority chain must be placed next to each other with no unused card slots in between. Cards that do not use the priority chain can be installed between cards using the chain, provided these cards pass on the priority chain by connecting pins 51 and 52 (PCI and PCO) together on board. This is standard practice on all VersaLogic STD Bus boards.

The boards that use the priority chain, as a group, can be placed anywhere in the card rack. Boards which do not use the chain can be freely positioned on either side of the group. They do not have to be placed side by side.

The VL-7312 / VL-7314 can also be used in systems that do not use the STD Bus priority chain. The STD Bus signal INTRQ* is provided at connector J1 pin 9 as INT1*, and can be disconnected from the STD Bus by removing jumper $V1_C$.

External Connections

Connections to the VL-7312 / VL-7314 can be made with standard cable assemblies, or with the following mating connectors:

Connector	Cable Assembly
JA-JD	Use 26-pin socket type connectors such as 3M #3399-7026
J1	Use 2-pin socket type connectors such as AMP #530554-1

Figure 3-1. Cable Assemblies

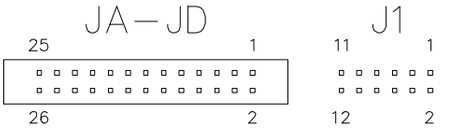


Figure 3-2. Physical Pin Locations

RS-232 Connectors

26-pin header type connectors are used for the RS-232 I/O connectors JA, JB, JC and JD. These connectors are normally converted to DB-25 type connectors using VersaLogic cable #9560 or similar mass terminated cable assembly. Pinouts are included for both the on-board connector and for the DB-25 connector after the port has been converted to this format.

Direct connection to the 26-pin headers can be made using mating connectors such as 3M #3399-7026, AMP #499505-7, or Ansley #609-2641.

JA, JB, JC[1], JD[1] Pin	RS-232 Signal	RS-232 Pin	DCE Signal Direction	DTE Signal Direction
1	–	1	–	–
2	–	14	–	–
3	TxD (BA)	2	IN	OUT
4	RCLK (DB)	15	OUT	IN
5	RxD (BB)	3	OUT	IN
6	–	16	–	–
7	RTS (CA)	4	IN	OUT
8	RCLK (DD)	17	OUT	IN
9	CTS (CB)	5	OUT	IN
10	–	18	–	–
11	DSR (CC)	6	OUT	IN
12	–	19	–	–
13	GND (AB)	7	GND	GND
14	DTR (CD)	20	IN	OUT
15	–	8	–	–
16	–	21	–	–
17	–	9	–	–
18	–	22	–	–
19	–	10	–	–
20	–	23	–	–
21	–	11	–	–
22	TCLK (DA)	24	IN	OUT
23	–	12	–	–
24	–	25	–	–
25	–	13	–	–
26	–	–	–	–

[1] Available on VL-7314 only.

Figure 3-3. RS-232 Connector Pinout

Interrupt Connector

The VL-7314 has four general purpose interrupt inputs and two interrupt outputs accessible through connector J1. (The VL-7312 has only two inputs and one output). A low level signal applied to the general purpose interrupt inputs can initiate unique vectored interrupt requests over the STD Bus. The interrupt output signals are tied to the STD Bus signal INTRQ* through jumpers $V1_b$ and $V1_c$. They are made available at connector J1 pins 9 and 11 for connection to an external interrupt controller card if desired.

J1 Pin	Signal	Description
1	INTA*	General purpose interrupt input A
3	INTB*	General purpose interrupt input B
5	INTC*[1]	General purpose interrupt input C
7	INTD*[1]	General purpose interrupt input D
9	INT1*	Interrupt output from SCC1 (and/or SCC2[1])
11	INT2*[1]	Interrupt output from SCC2
2-12	All even pins grounded	

* Low level signal.
[1] Available on VL-7314 only.

Figure 3-4. Interrupt Connector Pinout

Operation

Introduction

This section includes general information about the use and operation of the VL-7312 / VL-7314 boards. It focuses primarily on the registers and software commands necessary to operate the card.

The VL-7314 card contains two Serial Communication Controller (SCC) chips for a total of four serial channels (A, B, C, and D). The VL-7312 card has only one SCC chip with two serial channels (A and B). Please disregard all references to SCC#2 or channels C and D when programming a VL-7312 card.

All functions of the VL-7312 / VL-7314 are programmed through the on-board SCC chips. These chips contain two full duplex channels, two baud rate generators, and interrupt logic. Associated with each channel are a number of read and write registers for mode control and status information. Modem control signals are monitored and sensed under program control. All of the modem control signals are general purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten Control (write) registers, two Sync Character (write) registers, and four Status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic are a write register for the interrupt vector accessible through either channel, a write only Master Interrupt Control register, and three read registers: one containing the unmodified interrupt vector (channels A and C only), one containing the vector modified with status information (channels B and D only), one containing the Interrupt Pending bits (channels A and C only).

I/O Port Mapping

Output Port	Input Port	Port Address	As Shipped Address
Channel A XMIT Data	Channel A RCV Data	Board Address + 0	B0
Channel A Write Register 0	Channel A Read Register 0	Board Address + 1	B1
Channel B XMIT Data	Channel B RCV Data	Board Address + 2	B2
Channel B Write Register 0	Channel B Read Register 0	Board Address + 3	B3
Channel C XMIT Data	Channel C RCV Data	Board Address + 4	B4
Channel C Write Register 0	Channel C Read Register 0	Board Address + 5	B5
Channel D XMIT Data	Channel D RCV Data	Board Address + 6	B6
Channel D Write Register 0	Channel D Read Register 0	Board Address + 7	B7

Figure 4-1. I/O Port Addresses

SCC Registers

The following table lists the functions assigned to each read and write register. Each SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Write Register	Functions	Page
WR0 [1]	CRC initialize, initialization commands for the various modes, Register Pointer	4-4
WR1 [2]	Transmit/Receive interrupt and data transfer mode definition	4-6
WR2 [2]	Interrupt vector (accessed through either channel)	4-8
WR3 [2]	Receive parameters and control	4-8
WR4 [2]	Transmit/Receive miscellaneous parameters and modes	4-9
WR5 [2]	Transmit parameters and controls	4-11
WR6 [2]	Sync characters or SDLC address field	4-12
WR7 [2]	Sync character or SDLC flag	4-13
WR8 [1,2]	Transmit buffer	4-13
WR9 [2]	Master interrupt control and reset (accessed through either channel)	4-13
WR10 [2]	Miscellaneous transmitter/receiver control bits	4-15
WR11 [2]	Clock mode control	4-16
WR12 [2]	Lower byte of baud rate generator time constant	4-18
WR13 [2]	Upper byte of baud rate generator time constant	4-18
WR14 [2]	Miscellaneous control bits	4-19
WR15 [2]	External/Status interrupt control	4-21

Read Register	Functions	Page
RR0 [1]	Transmit/Receive buffer status and External status	4-22
RR1 [2]	Special Receive Condition status	4-23
RR2 [2,3]	Interrupt vector written into WR2	4-24
RR2 [2,4]	Modified interrupt vector	4-25
RR3 [2]	Interrupt pending bits	4-25
RR8 [1,2]	Receive buffer	4-25
RR10 [2]	Miscellaneous status	4-26
RR12 [2]	Lower byte of baud rate generator time constant	4-26
RR13 [2]	Upper byte of baud rate generator time constant	4-27
RR15 [2]	External/Status interrupt information	4-27

[1] Register directly accessible as I/O port.
[2] Register accessible by writing pointer to WR0 first.
[3] When read from channel A or channel C.
[4] When read from channel B or channel D.

Figure 4-2. Read and Write Register Functions

Accessing SCC Write Registers

Writing to WR0 is straightforward. Since WR0 is mapped as an output port, an output directed to one of the WR0 port addresses (i.e., B3H for serial channel B) is all that is required to update the contents of the selected WR0 register.

Writing to the XMIT Data register is also direct. Since it too is mapped as an output port, an output to the desired XMIT Data register port address (i.e., B2H for serial channel B) is all that is required to update the contents of the Transmit buffer.

Write registers WR1 - WR15 are accessed by indexing. They require two output operations to update their contents. First an output operation is directed to WR0 to index the desired write register. The three least significant bits of WR0, coupled with the "Point High" command (also within WR0), point to the desired write register. A second output operation directed to WR0 actually updates the indexed write register.

The pointer bits within WR0 are automatically cleared after the second output operation so that WR0/RR0 is addressed again.

Note: Special precautions must be taken if your system uses interrupts which access any of the SCC indexed registers WR1-WR15. Since interrupts occur asynchronously with respect to regular, mainline program flow, it is possible for an interrupt to occur between the time a register is indexed (first output to WR0) and the time the register is updated (second output to WR0). Unpredictable results will occur due to the fact that the interrupt service routine (wanting to communicate with an indexed register itself) must also perform two operations; an index and an access cycle. In this situation the SCC chip will get confused; it sees the new register index cycle (within the interrupt service routine) as the second output to WR0, thereby erroneously updating a previously indexed register. Furthermore, when processing returns to the interrupted program, incorrect data will again be written to WR0. Index synchronization will be totally lost. This can be prevented by disabling interrupts while indexing and writing to SCC indexed registers.

Accessing SCC Read Registers

Reading from RR0 is straightforward. Since RR0 is mapped as an input port, an input from the RR0 port addresses (i.e., B3H for serial channel B) is all that is required to fetch the contents of the selected RR0 register.

Reading the RCV Data register is also direct. Since it too is mapped as an input port, an input from the desired RCV Data register port address (i.e., B2H for serial channel B) is all that is required to fetch the contents of the Receive buffer.

Read registers RR1 - RR15 are accessed by indexing. They require both an output and an input operation to fetch their contents. First an output operation is directed to WR0 to index the desired read register. The three least significant bits of WR0, coupled with the "Point High" command (also within WR0), point to the desired read register. The contents of the selected read register is subsequently available by performing an input from RR0 (i.e., B3H for serial channel B).

The pointer bits within WR0 are automatically cleared after the input operation so that WR0/RR0 is addressed again.

Note: Special precautions must be taken if your system uses interrupts which access any of the SCC indexed registers RR1-RR15. Since interrupts occur asynchronously with respect to regular, mainline program flow, it is possible for an interrupt to occur between the time a register is indexed (output to WR0) and the time the register is read (input from WR0). Unpredictable results will occur due to the fact that the interrupt service routine (wanting to communicate with an indexed register itself) must also perform two operations; an index and an access cycle. In this situation the SCC chip will get confused; it sees the new register index cycle (within the interrupt service routine) as the second output to WR0, thereby erroneously updating a previously indexed register. Furthermore, when processing returns to the interrupted program, incorrect data will again be read from WR0. Index synchronization will be totally lost. This can be prevented by disabling interrupts while indexing and reading from SCC indexed registers.

SCC Write Registers

Each SCC contains two sets of write registers (one for each serial channel). The functional personalities (initialization) of the serial channels are programmed by writing to the SCC write registers. Certain run time operational aspects such as handshake and interrupt control are also managed by writing to the SCC write registers.

Write Register 0

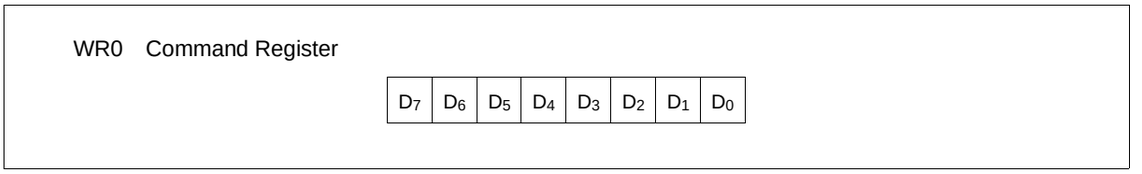


Figure 4-3. Write Register 0

D7, D6 -- CRC Reset Codes.

D7	D6	Action
0	0	Null Code
0	1	Reset Receive CRC Checker
1	0	Reset Transmit CRC Generator
1	1	Reset Transmit Underrun/EOM Latch

Figure 4-4. CRC Reset Codes

(00) Null Code. This command has no effect on the SCC and is used when a write to WR0 is necessary for some other reason other than a CRC Reset command.

(01) Reset Receive CRC Checker. This command is used to initialize the receive CRC circuitry. It is necessary in synchronous modes (except SDLC) if the Enter Hunt Mode command in WR3 is not issued between received messages. Any action that disables the receiver initializes the CRC circuitry. Resetting the Receive CRC Checker command is accomplished automatically in SDLC mode.

(10) Reset Transmit CRC Generator. This command initializes the CRC generator. It is usually issued in the initialization routine and after the CRC has been transmitted. A Channel Reset will not initialize the generator and this command should not be issued until after the transmitter has been enabled in the initialization routine.

(11) Reset Transmit Underrun/EOM Latch. This command controls the transmission of CRC at the end of transmission (EOM). If this latch has been reset, and a transmit underrun occurs, the SCC automatically appends CRC to the message. In SDLC mode with Abort on Underrun selected (WR10, D2 = "1") the SCC sends an abort and a flag after the underrun condition if the Tx Underrun/EOM latch has been reset.

At the start of the CRC transmission, the Tx Underrun/EOM latch is set. The Reset command can be issued at any time during a message. If the transmitter is disabled, this command will not reset the latch. However, if no External Status interrupt is pending, or if a Reset External Status Interrupt command accompanies this command while the transmitter is disabled, an External/Status interrupt is generated with the Tx Underrun/EOM bit reset in RR0.

D5, D4, D3 -- Command Codes.

D5	D4	D3	Command
0	0	0	Null code
0	0	1	Point high
0	1	0	Reset external/status interrupts
0	1	1	Send abort (SDLC)
1	0	0	Enable int on next Rx character
1	0	1	Reset TxINT pending
1	1	0	Error reset
1	1	1	Reset highest IUS

Figure 4-5. Command Codes

(000) Null code. The Null command has no effect on the SCC.

(001) Point High. This command effectively adds eight to the Register Pointer (D2–D0) allowing registers 8 through 15 to be accessed. The Point High command and the Register Pointer bits are written simultaneously.

(010) Reset External/Status Interrupts. After an External/Status interrupt (a change on an input handshake line or a break condition, for example), the status bits in RR0 are latched. This command reenables the bits and allows interrupts to occur again as a result of a status change. Latching the status bits captures short pulses until the CPU has time to read the change. The SCC contains simple queuing logic associated with most of the external status bits in RR0. If another External/Status condition changes while a previous condition is still pending (the Reset External/Status Interrupts command has not yet been issued) and this condition persists until after the command is issued, this second change causes another External/Status interrupt. However, if this second status change does not persist (there are two transitions), another interrupt is not generated. Exceptions to this rule are detailed in the RR0 description (see page 4-22).

(011) Send Abort. This command is used in SDLC mode to transmit a sequence of eight to thirteen "1s". This command always empties the Transmit buffer and sets Tx Underrun/EOM bit in RR0.

(100) Enable Interrupt on Next Rx Character. If the interrupt on the First Received Character mode is selected, this command is used to reactivate that mode after each message is received. The next character to enter the Receive FIFO causes a Receive interrupt. Alternatively, the first previously stored character in the Receive FIFO will cause a Receive interrupt.

(101) Reset Tx Interrupt Pending. This command is used in cases where there are no more characters to be sent; e.g., at the end of a message. This command prevents further transmit interrupts until after the next character has been loaded into the Transmit buffer or until CRC has been completely sent. This command is necessary to prevent the transmitter from requesting an interrupt when the Transmit buffer becomes empty after transmitting the final character of a message.

(110) Error Reset. This command resets the error bits in RR1. If Interrupt on First Rx Character or Interrupt on Special Condition modes is selected and a special condition exists, the data with the special condition is held in the Receive FIFO until this command is issued. If either of these modes is selected and this command is issued before the data has been read from the Receive FIFO, the data is lost.

(111) Reset Highest IUS. This command resets the highest priority Interrupt Under Service (IUS) bit, allowing lower priority conditions to request interrupts. This command allows the use of the internal daisy chain (even in systems without an external daisy chain) and should be the last operation in an interrupt service routine.

D2, D1, D0 -- Register Selection Code. These bits select the register in which a subsequent write to WR0 or read from RR0 will be directed. With the Point High command, registers 8 through 15 are selected.

D2	D1	D0	Selected Register
0	0	0	Register 0
0	0	1	Register 1
0	1	0	Register 2
0	1	1	Register 3
1	0	0	Register 4
1	0	1	Register 5
1	1	0	Register 6
1	1	1	Register 7
0	0	0[1]	Register 8
0	0	1[1]	Register 9
0	1	0[1]	Register 10
0	1	1[1]	Register 11
1	0	0[1]	Register 12
1	0	1[1]	Register 13
1	1	0[1]	Register 14
1	1	1[1]	Register 15

[1] Use Point High command.

Figure 4-6. Register Selection Codes

Write Register 1

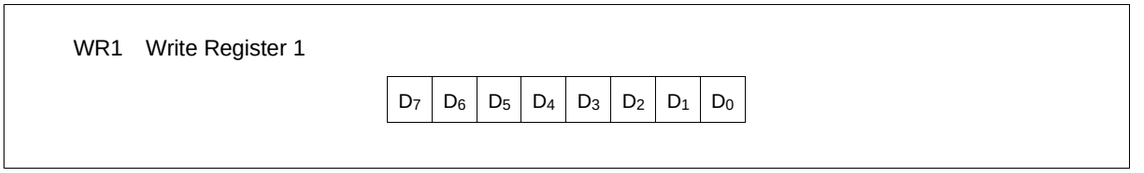


Figure 4-7. Write Register 1

WR1 is the Control register for the various SCC interrupt and Wait/Request modes.

D7 -- WAIT/DMA Request Enable. This bit has no meaning on the VL-7312 / VL-7314.

D6 -- WAIT/DMA Request Function. This bit has no meaning on the VL-7312 / VL-7314.

D5 -- WAIT/DMA Request On Receive Transmit. This bit has no meaning on the VL-7312 / VL-7314.

D4, D3 -- Receive Interrupt Modes. These two bits specify the various character-available conditions that may cause interrupt requests.

D4	D3	Action
0	0	Receive Interrupts Disabled
0	1	Receive Interrupt on First Character or Special Condition
1	0	Interrupt on All Receive Characters or Special Conditions
1	1	Interrupt on Special Conditions Only

Figure 4-8. Receive Interrupt Modes

(00) Receive Interrupts Disabled. This mode prevents the receiver from requesting an interrupt and is normally used in a polled environment where either the status bits in RR0 or the modified vector in RR2 (Channel B) can be monitored to initiate a service routine. Although the receiver interrupts are disabled, a special condition can still provide a unique vector status in RR2.

(01) Receive Interrupt on First Character or Special Condition. The receiver requests an interrupt in this mode on the first available character (or stored Receive FIFO character) or on a special condition. Sync characters to be stripped from the message stream do not cause interrupts.

Special receive conditions are: receiver overrun, framing error, end of frame, or parity error (if selected). If a special receive condition occurs, the data containing the error is stored in the Receive FIFO until an Error Reset command is issued by the CPU.

This mode is usually selected when a Block Transfer mode is used. In this interrupt mode, a pending special receive condition remains set until either an Error Reset command, a channel or hardware reset, or until receive interrupts are disabled.

The Receive Interrupt on First Character or Special Condition mode can be reenabled by the Enable Rx Interrupt on Next Character command in WR0.

(10) Interrupt on All Receive Characters or Special Conditions. This mode allows an interrupt for every character received (or character in the Receive FIFO) and provides a unique vector when a special condition exists. The Receiver Overrun bit and the Parity Error bit in RR1 are two special conditions that are latched. These two bits must be reset by the Error Reset command. Receiver overrun is always a special receive condition, and parity can be programmed to be a special condition.

Data characters with special receive conditions are not held in the Receive FIFO in the Interrupt On All Receive Characters or Special Conditions Mode as they are in the other receive interrupt modes.

(11) Interrupt on Special Conditions Only. This mode allows the receiver to interrupt only on characters with a special receive condition. When an interrupt occurs, the data containing the error is held in the Receive FIFO until an Error Reset command is issued. When using this mode in conjunction with a DMA, the DMA can be initialized and enabled before any characters have been received by the SCC. This eliminates the time critical section of code required in the Receive Interrupt on First Character or Special condition mode; i.e., all data can be transferred via the DMA so that the CPU need not handle the first received character as a special case.

D2 -- Parity is Special Condition. This bit controls whether or not received characters with parity error (those not matching the sense programmed in WR4) give rise to a Special Receive Condition. When this bit is set to "1", the definition of Special Receive Condition is expanded to include all received characters with parity error. When this bit is reset to "0", characters with parity error do not invoke the Special Receive Condition, however, parity status is always available via RR1. When the Vector Includes Status function is enabled via WR9, a Special Receive Condition modifies the interrupt vector stored in WR2. If parity is disabled, this bit is ignored.

D1 -- Transmitter Interrupt Enable. When this bit is set to "1", the transmitter requests an interrupt whenever the Transmit buffer becomes empty. When reset to "0", interrupts from the transmitter are disabled.

D0 -- External/Status Master Interrupt Enable. This bit is the master interrupt enable for the following interrupt sources:

- External interrupts applied to connector J1
- DTR/DSR input handshake signals
- RTS/CTS input handshake signals
- Break/Abort
- CRC transmission (when Tx Underrun/EOM latch is set)
- Baud rate generator zero count

When this bit is reset to "0", all of these interrupt sources are disabled. When set to "1", these interrupts are enabled (provided the corresponding interrupt enable bits within WR15 are set to "1").

Write Register 2

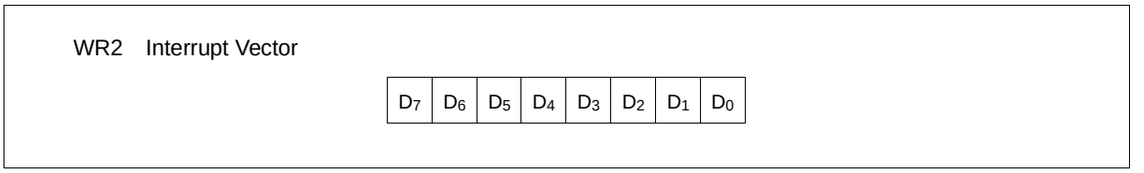


Figure 4-9. Write Register 2

WR2 is the Interrupt Vector register. Only one Vector register exists within in each SCC chip, but they can be accessed by writing to either channel (A/B or C/D). When Vector Includes Status (VIS) and/or Status High/Status Low bits in WR9 are set to "1", the interrupt vector written into WR2 is modified prior to being sent to the CPU. The vector written into WR2 can be read by reading RR2 (access via channels A or C only). If the VIS bit is set to "1", the modified vector can be read by reading RR2 (access via channels B or D only).

Write Register 3

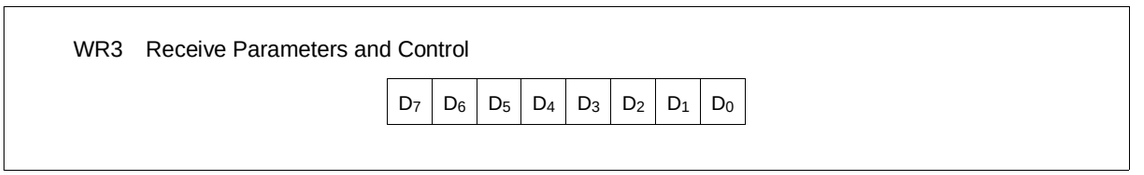


Figure 4-10. Write Register 3

WR3 contains the control bits and parameters for the receiver logic.

D7, D6 -- Receiver Bits Per Character. These bits determine the number of bits to be assembled as a character from the received serial data stream.

D7	D6	Character Length
0	0	5
0	1	7
1	0	6
1	1	8

Figure 4-11. Receiver Bits Per Character

D5 -- Auto Enables. This bit programs the function for the DTR and RTS input handshaking signals (DSR and CTS when jumpered for DTE operation). When this bit is set to "1", RTS (CTS) becomes the transmitter enable and DTR (DSR) becomes the receiver enable. When this bit is reset to "0", the handshaking signals are merely inputs to the corresponding status bits, D3 and D5, in RR0.

D4 -- Enter Hunt Mode. When this bit is set to "1", a comparison is made between assembled receive characters and sync characters (or flags) for the purpose of synchronization. Whenever a flag or sync character is matched, the Sync/Hunt bit in RR0 is reset and, if External/Status Interrupt Enable is set, an interrupt sequence is initiated. The SCC automatically enters the Hunt mode when an abort condition is received or when the receiver is disabled.

D3 -- Receiver CRC Enable. Setting this bit to "1" initiates a CRC calculation at the beginning of the last byte transferred from the Receiver Shift register to the Receive FIFO. This operation occurs independently of the number of bytes in the Receive FIFO. When a particular byte is to be excluded from CRC calculation, this bit should be reset before the next byte is transferred to the Receive FIFO. If this feature is used, care must be taken to ensure that eight bits per character is selected in the receiver because of an inherent delay from the Receive Shift register to the CRC checker.

This bit is internally set to "1" in SDLC mode and the SCC calculates CRC on all bits except inserted zeros between the opening and closing character flags. This bit is ignored in asynchronous modes.

D2 -- Address Search Mode (SDLC). Setting this bit to "1" in SDLC mode causes messages with addresses not matching the address programmed in WR6 to be rejected. No receiver interrupts can occur in this mode unless there is an address match. The address that the SCC attempts to match can be unique (1 in 256) or multiple (16 in 256), depending on the state of the Sync Character Load Inhibit bit. The Address Search Mode bit is ignored in all modes except SDLC.

D1 -- SYNC Character Load Inhibit. In any synchronous mode except SDLC, setting this bit to "1" causes the SCC to compare the byte in WR6 with the byte about to be stored in the Receive FIFO, and it inhibits this load if the bytes are equal. The SCC does not calculate the CRC on bytes stripped from the data stream in this manner. If the 6-bit sync option is selected while in Monosync mode, the compare is still across eight bits, so WR6 must be programmed for proper operation.

If the 6-bit sync option is selected with this bit set to "1", all sync characters except the one immediately preceding the data are stripped from the message. If the 6-bit sync option is selected while in the Bisync mode, this bit is ignored.

In SDLC mode the address recognition logic of the receiver is modified if this bit is set to "1;" i.e., only the four most significant bits of WR6 must match the receiver address. This procedure allows the SCC to receive frames from up to sixteen separate sources without programming WR6 for each source (if each station address has the four most significant bits in common). The address field in the frame is still eight bits long.

In SDLC mode this bit is ignored if Address Search mode has not been selected.

D0 -- Receiver Enable. When this bit is set to "1", receiver operation begins. This bit should be set only after all other receiver parameters are established and the receiver is completely initialized. This bit is reset by a channel or hardware reset command, disabling the receiver.

Write Register 4

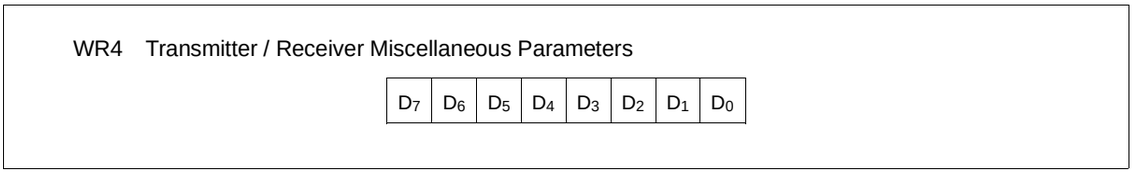


Figure 4-12. Write Register 4

WR4 contains the control bits for both the receiver and the transmitter. These bits should be set in the transmit and receiver initialization routine before writing to registers WR1, WR3, WR6, and WR7.

D7, D6 -- Clock Rate. These bits specify the multiplier between the clock and data rates. In synchronous modes, the 1x mode is forced internally and these bits are ignored unless External Sync mode has been selected.

D7	D6	Action
0	0	1x Mode
0	1	16x Mode
1	0	32x Mode
1	1	64x Mode

Figure 4-13. Clock Rate Factors

(00) 1x Mode. The clock rate and data rate are the same. In External Sync mode, this bit combination specifies that only the INTA*, INTB*, INTC*, and INTD* signals applied at connector J1 can be used to achieve character synchronization.

(01) 16x Mode. The clock rate is 16 times the data rate. In External Sync mode, this bit combination specifies that only the INTA*, INTB*, INTC*, and INTD* signals can be used to achieve character synchronization.

(10) 32x Mode. The clock rate is 32 times the data rate. In External Sync mode, this bit combination specifies that either the INTA*, INTB*, INTC*, and INTD* signals or a match with the character stored in WR7 will signal character synchronization. The sync character can be either six or eight bits long as specified by the 6-Bit/8-Bit Sync bit in WR10.

(11) 64x Mode. The clock rate is 64 times the data rate. With this bit combination in External Sync mode, both the receiver and transmitter are placed in SDLC mode. The only variation from normal SDLC operation is that the INTA*, INTB*, INTC*, and INTD* signals can be used to start or stop the reception of a frame by forcing the receiver to act as though a flag had been received.

D5, D4 -- SYNC Mode.
These two bits select the various options for character synchronization. They are ignored unless synchronous operation is selected in the stop bits field of this register.

D5	D4	Action
0	0	Monosync
0	1	Bisync
1	0	SDLC
1	1	External Sync

Figure 4-14. Sync Modes

(00) Monosync. In this mode, the receiver achieves character synchronization by matching the character stored in WR7 with an identical character in the received data stream. The transmitter uses the character stored in WR6 as a time fill. The sync character can be either six or eight bits, depending on the state of the 6-Bit/8-Bit Sync bit in WR10. If the Sync Character Load Inhibit bit is set, the receiver strips the contents of WR6 from the data stream if received within character boundaries.

(01) Bisync. The concatenation of WR7 with WR6 is used for receiver synchronization and as a time fill by the transmitter. The sync character can be 12 bits or 16 bits in the receiver, depending on the state of the 6-Bit/8-Bit Sync bit in WR10. The transmitted character is always 16 bits.

(10) SDLC Mode. In this mode, SDLC is selected and requires a Flag (01111110) to be written to WR7. The receiver address field should be written to WR6. The SDLC CRC polynomial must also be selected (WR5) in SDLC mode.

(11) External Sync Mode. In this mode, the SCC expects external logic to signal character synchronization via the INTA*, INTB*, INTC*, and INTD* signals.

D3, D2 -- Stop Bit Length / Timing Mode. These bits determine the number of stop bits added to each asynchronous character that is transmitted. The receiver always checks for one stop bit in Asynchronous mode. A Special mode selects Synchronous operation.

D3	D2	Action
0	0	Synchronous mode enable
0	1	Asynchronous mode, 1 stop bit
1	0	Asynchronous mode, 1½ stop bits
1	1	Asynchronous mode, 2 stop bits

Figure 4-15. Stop Bit Length / Timing Mode

(00) Synchronous Modes Enable. This bit combination selects one of the synchronous modes specified by bits D4, D5, D6, and D7 of this register and forces the 1x Clock mode internally.

(01) 1 Stop Bit per Character. This bit combination selects Asynchronous mode with one stop bit per character.

(10) 1½ Stop Bits per Character. This bit combination selects Asynchronous mode with 1½ stop bits per character. This mode can not be used with the 1x clock mode.

(11) 2 Stop Bits Per Character. These bits select Asynchronous mode with 2 stop bits per transmitted character.

D1 -- Parity Even/Odd. This bit determines whether parity on received characters is checked as even or odd. A "1" programmed here selects even parity, and a "0" selects odd parity. This bit is ignored if the Parity Enable bit reset to "0".

D0 -- Parity Enable. When this bit is set to "1", an additional bit position beyond those specified in the bits per character control is added to the transmitted data and is expected in the receive data. The Received Parity bit is transferred to the CPU as part of the data unless eight bits per character is selected in the receiver.

Write Register 5

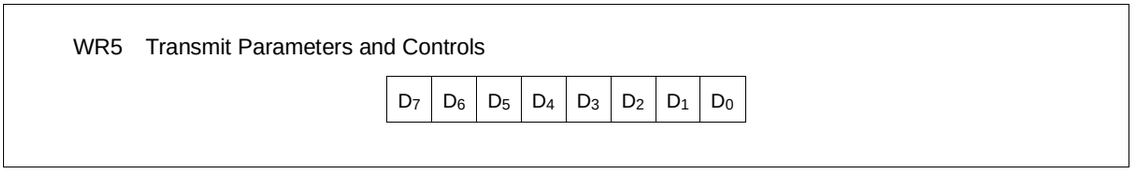


Figure 4-16. Write Register 5

WR5 contains control bits that affect operation of the transmitter. D2 affects both the transmitter and the receiver.

D7 -- Data Terminal Ready. This is the control bit for the DSR output handshaking signal (DTR when jumpered for DTE operation). When this bit is set to "1", DSR (DTR) goes ready, (RS-232 +V); when reset to "0", the signal goes busy (RS-232 −V). This bit is reset to "0" by a channel or hardware reset. See the Handshaking Signals code example on page 5-2 for an example of how to use this bit.

D6, D5 -- Transmitter Bits per Character. These bits control the number of bits in each byte transferred to the Transmit buffer. Bits are assumed to be right justified towards D0. Transmitted data is shifted out least significant bit first. An additional parity bit is sent if enabled within WR4. The Five Or Less mode allows transmission of one to five bits per character; however, the CPU should force the unused upper order bits to 1 or 0 as shown below.

D6	D5	Character Length
0	0	5 or less
0	1	7
1	0	6
1	1	8

Figure 4-17. Transmitter Bits Per Character

D7	D6	D5	D4	D3	D2	D1	D0	Comments
1	1	1	1	0	0	0	D_0	One data bit transmitted
1	1	1	0	0	0	D_1	D_0	Two data bits transmitted
1	1	0	0	0	D_2	D_1	D_0	Three data bits transmitted
1	0	0	0	D_3	D_2	D_1	D_0	Four data bits transmitted
0	0	0	D_4	D_3	D_2	D_1	D_0	Five data bits transmitted

Figure 4-18. Five or Less Character Format

D4 -- Send Break. When this bit is set to "1", the transmit data signal is forced into a spacing condition (RS-232 +V). This condition, called a line break, persists continuously regardless of the data being transmitted at the time. This bit functions whether or not the transmitter is enabled. When this bit is reset to "0", data transmission resumes. This bit is reset by a channel or hardware reset.

D3 -- Transmit Enable. When this bit is set to "1", the transmitter is enabled for normal operation. Data is not transmitted until this bit is set. While reset, the transmit data signal remains in the marking state (RS-232 −V) unless Auto Echo mode or SDLC Loop mode is selected. If this bit is reset in the middle of a transmission, the transmission of data or sync characters is completed prior to transmitter shut-down. An interrupt still occurs when the Transmit Data register becomes empty. Only the character currently being sent is completed. A pending character in the Transmit Data register remains pending. If the transmitter is disabled during the transmission of a CRC character, sync or flag characters are sent instead of CRC. This bit is reset by a channel or hardware reset.

D2 -- SDLC/CRC-16. This bit selects the CRC polynomial used by both the transmitter and the receiver. When this bit is set to "1", the CRC-16 polynomial $X^{16} + X^{15} + X^2 + X^1$ is used; when reset to "0", the SDLC polynomial $X^{16} + X^{12} + X^2 + 1$ is used. When SDLC mode is used, this bit should be reset to "0". The CRC generator and checker can be preset to all "0s" or all "1s", depending on the state of the Preset 1 / Preset 0 bit in WR10.

D1 -- Request To Send. This is the control bit for the CTS output handshaking signal (RTS when jumpered for DTE operation). When this bit is set to "1", CTS (RTS) goes ready, (RS-232 +V); when reset to "0", the signal goes busy (RS-232 −V). This bit is reset to "0" by a channel or hardware reset. See the Handshaking Signals code example on page 5-2 for an example of how to use of this bit.

D0 -- Transmit CRC Enable. When this bit is set to "1", CRC is calculated on a transmit character. If this bit is set at the time the character is loaded from the Transmit buffer to the Transmit Shift register, CRC is calculated on that character. CRC is not automatically sent unless this bit is set when the transmit underrun exists.

Write Register 6

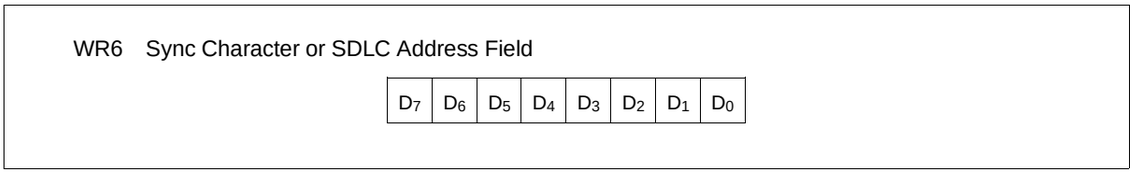


Figure 4-19. Write Register 6

WR6 contains the first 8 bits of a BISYNC sequence. It must contain the check address (if used) for SDLC mode and the sync character in the 8-bit sync mode.

D7	D6	D5	D4	D3	D2	D1	D0	Mode
SYNC7	SYNC6	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Monosync, 8 Bits
SYNC1	SYNC0	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Monosync, 6 Bits
SYNC7	SYNC6	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Bisync, 16 Bits
SYNC3	SYNC2	SYNC1	SYNC0	1	1	1	1	Bisync, 12 Bits
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	SDLC
ADR7	ADR6	ADR5	ADR4	ADR3	x	x	x	(Address Range)

Figure 4-20. BISYNC Sequence

Write Register 7

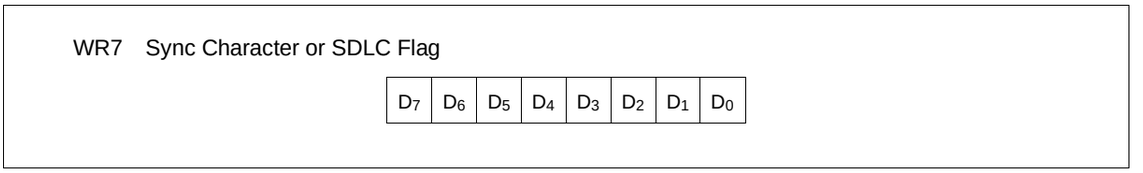


Figure 4-21. Write Register 7

WR7 contains the receive sync character in the Monosync mode, a second byte (the last eight bits) of a 16-bit sync character in the Bisync mode, or a Flag character (01111110) in the SDLC modes. WR7 may hold the receive sync character or a flag if one of the special versions of the External Sync mode is selected. WR7 is not used in Asynchronous mode.

D7	D6	D5	D4	D3	D2	D1	D0	Mode
SYNC7	SYNC6	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Monosync, 8 bits
SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	x	x	Monosync, 6 bits
SYNC15	SYNC14	SYNC13	SYNC12	SYNC11	SYNC10	SYNC9	SYNC8	Bisync, 16 bits
SYNC11	SYNC10	SYNC9	SYNC8	SYNC7	SYNC6	SYNC5	SYNC4	Bisync, 12 bits
0	1	1	1	1	1	1	0	SDLC

Figure 4-22. Sync Character or SDLC Flag

Write Register 8

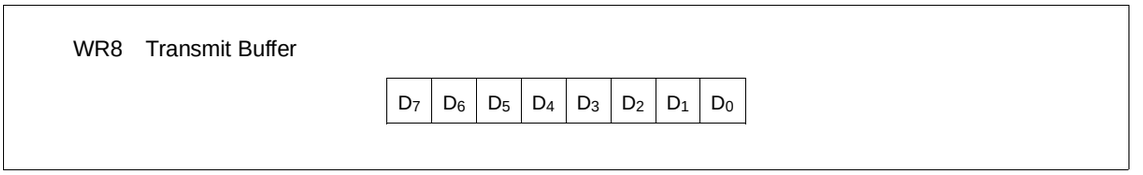


Figure 4-23. Write Register 8

WR8 is the Transmit Buffer register. In most cases, the Transmit Buffer register is accessed by writing directly to the XMIT Data port address assigned to each channel. See page 4-2 for further information.

Write Register 9

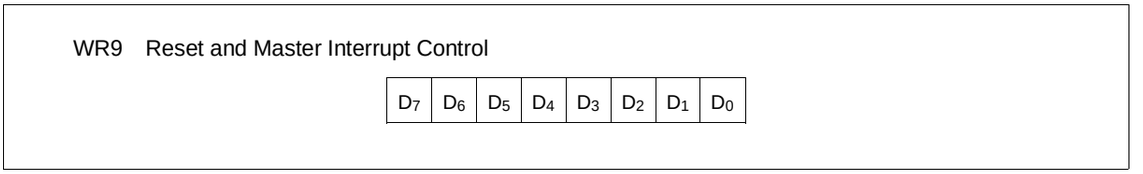


Figure 4-24. Write Register 9

WR9 is the Master Interrupt Control register and contains the Reset command bits. Only one WR9 exists in each SCC and can be accessed from either channel. The interrupt control bits can be programmed at the same time as the Reset command because these bits are only reset by a hardware reset.

D7, D6 -- Reset Command Bits. These bits select one of the reset commands for the SCC. Setting either D6 or D7 disables both the receiver and the transmitter in the corresponding channel, forces the output handshaking signals busy (RS-232 −V), resets the Interrupt Pending (IP), and Interrupt Under Service

(IUS) bits, and disables all interrupts from that channel. Four extra PCLK cycles (814 ns) must be allowed before additional command or control bytes are written to each SCC.

D7	D6	Action
0	0	No reset
0	1	Channel B Reset
1	0	Channel A Reset
1	1	Hardware Reset

Figure 4-25. Reset Commands

(00) No Reset. This command has no effect. It is used when a write to WR9 is necessary for some reason other than an SCC Reset command.

(01) Channel Reset B. This command causes a channel reset to be performed on Channel B.

(10) Channel Reset A. This command causes a channel reset to be performed on Channel A.

(11) Force Hardware Reset. The effects of this command are identical to those of a hardware reset, except the MIE, Status High/Status Low, and Disable Lower Chain bits are reset.

D5 -- Not Used. This bit is not used and must remain "0".

D4 -- Status High/Status Low. This bit controls which vector bits the SCC will modify. When this bit is set to "1", bits V6, V5, and V4 are modified; when reset to "0", V3, V2, and V1 are modified. This bit controls bit modification in both the vector returned during an interrupt acknowledge cycle and the value in RR2 (read from channel B or D only). This bit is reset to "0" by a hardware reset.

When using vectored interrupts on the VL-7314 card, it is recommended that SCC#1 (Channels A and B) be programmed with this bit reset to "0", and SCC#2 (Channels C and D) be programmed with this bit set to "1". Configuration in this manner allows for sixteen unique vectors to be generated.

D3 -- Master Interrupt Enable. Setting this bit to "1" enables interrupts from the SCC; resetting it to "0" disables interrupts. This bit is reset by a hardware reset.

D2 -- Disable Lower Chain. This bit can be used to control the interrupt daisy chain. Setting this bit to "1" prevents lower-priority devices on the daisy chain from requesting interrupts. This bit is reset by a hardware reset. For normal operation this bit should remain reset to "0".

D1 -- No Vector. This bit controls whether or not the SCC will respond to an interrupt acknowledge cycle with a vector. Normal operation of the VL-7312 / VL-7314 requires this bit to be reset to "0". If this bit is set to "1", an undetermined vector will be driven onto the data bus during the interrupt acknowledge cycle.

D0 -- Vector Includes Status. The Vector Includes Status Bit controls whether or not the SCC will include status information in the interrupt vector. If this bit is set to "1", the vector returned in response to an interrupt acknowledge cycle is variable. The variable field depends on the highest-priority Interrupt Pending (IP) bit that is set in RR3 (read from channel A or C only), and the state of the Status High/Status Low bit within WR9. When this bit is reset to "0", the value written into WR2 is returned, unmodified, as the interrupt vector.

V3 V2 V1 V4 V5 V6	Status High/Status Low = 0 Status High/Status Low = 1
0 0 0	Ch B Transmit Buffer Empty
0 0 1	Ch B External/Status Change
0 1 0	Ch B Receive Character Available
0 1 1	Ch B Special Receive Condition
1 0 0	Ch A Transmit Buffer Empty
1 0 1	Ch A External/Status Change
1 1 0	Ch A Receive Character Available
1 1 1	Ch A Special Receive Condition

Figure 4-26. Interrupt Vector Modification

Write Register 10

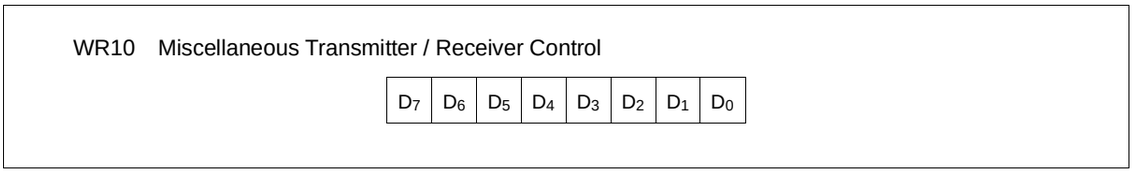


Figure 4-27. Write Register 10

WR10 contains miscellaneous control bits for both the receiver and the transmitter.

D7 -- CRC Presets 1/0. This bit specifies the initialized condition of the receive CRC checker and the transmit CRC generator. If this bit is set to "1", the CRC generator and checker are preset to "1". If this bit is reset to "0", the CRC generator and checker are preset to zero. Either option can be selected with either CRC polynomial. In SDLC mode, the transmitted CRC is inverted before transmission and the received CRC is checked against the bit pattern "0001110100001111". This bit is reset by a channel or hardware reset, and is ignored in Asynchronous mode.

D6, D5 -- Data Encoding. These bits control the coding method used for both the transmitter and the receiver. Normally, operation of the VL-7312 / VL-7314 will use NRZ encoding (00). All of the clocking options are available for all coding methods. The DPLL in the SCC is useful for recovering clocking information in NRZI and FM modes. A hardware reset forces NRZ mode.

D6	D5	Encoding
0	0	NRZ (Used for Asynchronous mode)
0	1	NRZI
1	0	FM1
1	1	FM0

Figure 4-28. Data Encoding

D4 -- Go Active On Poll. When Loop mode is first selected during SDLC operation, SCC connects RxD to TxD with only gate delays in the path. The SCC does not go on loop and insert the 1-bit delay between RxD and TxD until this bit has been set to "1" and an EOP message received. When the SCC is on-loop, the transmitter cannot go active unless this bit is set at the time an EOP message is received. The SCC examines this bit whenever the transmitter is active in SDLC Loop mode and is sending a flag. If this bit is set to "1" at the time the flag is leaving the Transmit Shift register, another flag or data byte (if the Transmit buffer is full) is transmitted. If the this bit is reset to "0" at this time, the transmitter finishes sending the flag and reverts to the 1-Bit Delay mode. Thus, to transmit only one response frame, this bit should be reset to "0" after the first data byte is sent to the SCC but before CRC has been transmitted. If this bit is reset to "0" before the first data is written, the SCC completes the transmission of the present flag and reverts to the 1-Bit Delay mode. After gaining control of the loop, the SCC is not able to transmit again until a flag and another EOP message have been received. Though not strictly necessary, it is good practice to set this bit to "1" only upon receipt of a poll frame to ensure that the SCC does not go on loop without the CPU noticing it.

In synchronous modes other than SDLC with the Loop Mode bit set, the Go Active On Poll bit must be set to "1" before the transmitter can go active in response to a received sync character.

This bit is always ignored in Asynchronous mode and Synchronous modes unless the Loop Mode bit is set. This bit is reset by a channel or hardware reset.

D3 -- Mark/Flag Idle. This bit affects only SDLC operation and is used to control the idle line condition. If this bit is reset to "0", the transmitter sends flags as an idle line. If this bit is set to "1", the transmitter is placed into Mark Idle mode, and sends continuous "1s" after the closing flag of a frame. The idle line condition is selected byte by byte; i.e., either a flag or eight "1s" are transmitted. The primary station in an SDLC loop should be programmed for Mark Idle to create the EOP sequence. Mark Idle must be deselected at the beginning of a frame before the first data is written to the SCC, so that an opening

flag can be transmitted. This bit is ignored in Loop mode, but the programmed value takes effect upon exiting the Loop mode. This bit is reset by a channel or hardware reset.

D2 -- Abort/Flag On Underrun. This bit affects only SDLC operation and is used to control how the SCC responds to a transmit underrun condition. If this bit is set to "1" and a transmit underrun occurs, the SCC sends an abort and a flag instead of CRC. If this bit is reset, the SCC sends CRC on a transmit underrun. At the beginning of this 16-bit transmission, the Transmit Underrun/EOM bit is set, causing an External/Status interrupt. The CPU uses this status, along with the byte count from memory or the DMA, to determine whether the frame must be retransmitted. A Transmit Buffer Empty interrupt occurs at the end of this 16-bit transmission to start the next frame. If both this bit and the Mark/Flag Idle bit are set to "1", all "1s" are transmitted after the transmit underrun. This bit should be set after the first byte of data is sent to the SCC and reset immediately after the last byte of data so that the frame will be terminated properly with CRC and a flag. This bit is ignored in Loop mode, but the programmed value is active upon exiting Loop mode. This bit is reset by a channel or hardware reset.

D1 -- Loop Mode. In SDLC mode, the initial set condition of this bit forces the SCC to connect RxD to TxD and to begin searching the incoming data stream so that it can go on loop. All bits pertinent to SDLC mode operation in other registers must be set before this mode is selected. The transmitter and receiver should not be enabled until after this mode has been selected. As soon as the Go Active On Poll bit is set and an EOP is received, the SCC goes on loop. If this bit is reset after the SCC is on loop, the SCC waits for the next EOP to go off loop.

In Synchronous modes, the SCC uses this bit, along with the Go Active On Poll bit, to synchronize the transmitter to the receiver. The receiver should not be enabled until after this mode is selected. The transmitter is held in the marking state when this mode is selected unless a break condition is programmed. The receiver waits for a sync character to be received and then enables the transmitter on a character boundary. The break condition, if programmed, is removed. This mode works properly with sync characters of 6, 8, or 16 bits. This bit is ignored in Asynchronous mode and is reset by a channel or hardware reset.

D0 -- 6 Bit/8 Bit Sync. This bit selects a Synchronous mode. If this bit is set to "1" in Monosync mode, the receiver and transmitter sync characters are six bits long instead of the usual eight. In Bisync mode, the received sync will be 12 bits and the transmitter sync character will remain 16 bits long. This bit is ignored in SDLC and Asynchronous modes. This bit is reset by a channel or hardware reset.

Write Register 11

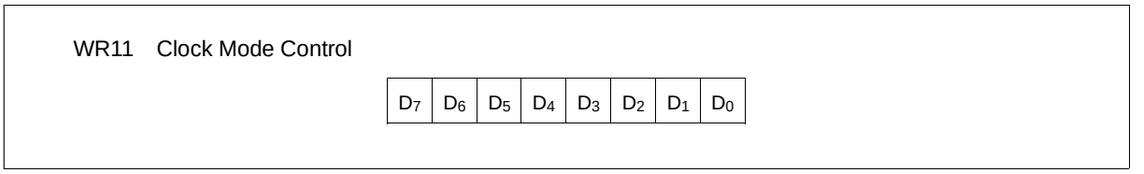


Figure 4-29. Write Register 11

WR11 is the Clock Mode Control register. The bits in this register control the sources of both the receive and transmit clocks.

D7 -- RTxC Input Type. This bit controls the type of input signal the SCC expects to see on the RTxC pin (SCC pin 12 for channels A / C; pin 28 for channels B / D). If this bit is reset "0", the SCC expects a TTL-compatible signal as an input to this pin. If this bit is set to "1", the SCC expects a quartz crystal. Note: This bit must be reset to "0" for correct operation of the VL-7312 / VL-7314.

D6, D5 -- Receive Clock Source. These bits determine the source of the receive clock timebase. They do not interfere with any of the modes of operation in the SCC but simply control a multiplexer just before the internal receive clock circuitry.

D6	D5	Source
0	0	TCLK (RCLK when jumpered for DTE operation). Use for Synchronous modes
0	1	Not used
1	0	Baud rate generator. Use for Asynchronous modes
1	1	Digital phase-locked loop (DPLL)

Figure 4-30. Receive Clock Source

D4, D3 -- Transmit Clock Source. These bits determine the source of the transmit clock timebase. They do not interfere with any of the modes of operation in the SCC but simply control a multiplexer just before the internal transmit clock circuitry. The DPLL output, used by the transmitter in FM modes, lags the output of the DPLL used by the receiver by 90˚. This causes the received and transmitted bit cells to occur simultaneously, neglecting delays.

D4	D3	Source
0	0	TCLK (RCLK when jumpered for DTE operation). Use for Synchronous modes
0	1	Not used
1	0	Baud rate generator. Use for Asynchronous modes
1	1	Digital phase-locked loop (DPLL)

Figure 4-31. Transmit Clock Source

D2 -- Transmit Clock Signal Direction. This bit determines the signal flow direction TRxC (SCC pin 14 for channels A / C; pin 26 for channels B / D). If this bit is set to "1", the signal is configured as an output, carrying the signal selected by D0 and D1 of this register. However, if either the receive or the transmit clock is programmed to come from the TRxC pin, TRxC will be an input, regardless of the state of this bit. The TRxC pin is also an input if this bit is reset to "0". A hardware reset forces this bit to "0".

Note: This bit must be set to "1" for correct operation of the VL-7312 / VL-7314. Any mode which causes TRxC to function as an input is meaningless.

D1, D0 -- Transmit Clock Output Source. These bits determine the signal output on the RCLK pin (TCLK when jumpered for DTE operation). Input modes are not usable on the VL-7312 / VL-7314 due to limitations of the RS-232 driver circuitry.

To reduce ±12V power consumption while running in Asynchronous mode, select the DPLL source. Since the DPLL is not used when running asynchronously, the output will assume a constant RS-232 +V level, thereby reducing switching currents within the 1488 driver chip.

D1	D0	Source
0	0	Not used on VL-7312 / VL-7314
0	1	Transmit clock
1	0	Baud rate generator
1	1	Digital phase-locked loop

Figure 4-32. Transmit Clock Output Source

Write Register 12

WR12 Lower Byte of Baud Rate Generator Time Constant

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Figure 4-33. Write Register 12

WR12 contains the lower byte of the baud rate generator time constant.

Write Register 13

WR13 Upper Byte of Baud Rate Generator Time Constant

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Figure 4-34. Write Register 13

WR13 contains the upper byte of the baud rate generator time constant.
The formulas for calculating time constant and baud rate values are given by:

$$Time\ Constant = \left[\frac{4.9162 \times 10^6}{2 \times (\ baud\ \ rate\ ) \times (\ clock\ mode\ )}\right] - 2$$

or

$$Baud\ Rate = \frac{4.9162 \times 10^6}{(\ 2 \times time\ constant \times clock\ mode\ ) + (\ 4 \times clock\ mode\ )}$$

Where:

clock mode = 1, 16, 32, or 64

Figure 4-35. Baud Rate Formulas

	SYNCHRONOUS (x1 Clock)			ASYNCHRONOUS (x16 Clock)		
Baud Rate	**Decimal**	**Hex**	**Error**	**Decimal**	**Hex**	**Error**
76.8K[1]	30	001E	0	0	0000	0
38.4K[1]	62	003E	0	2	0002	0
19.2K	126	007E	0	6	0006	0
9600	254	00FE	0	14	000E	0
4800	510	01FE	0	30	001E	0
2400	1022	03FE	0	64	0040	0
1800	1361	0551	0.17	83	0053	0.39
1200	2046	07FE	0	126	007E	0
600	4094	07FE	0	254	00FE	0
300	8190	1FFE	0	510	01FE	0
150	16382	3FFF	0	1022	03FE	0
110	22340	5744	.0008	1394	0572	.026

[1] The maximum data rate according to RS-232-C specifications is 20K baud.

Figure 4-35. Baud Rate Generator Time Constants

Write Register 14

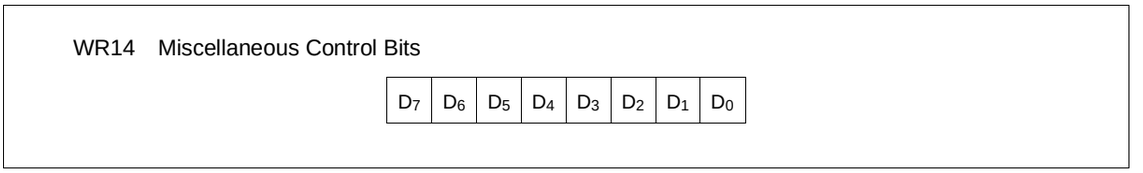


Figure 4-36. Write Register 14

WR14 contains miscellaneous control bits: baud rate generator, phase-locked loop control, auto echo, and local loopback modes.

D7, D6, D5 -- Digital Phase-Locked Loop Command Bits. These three bits encode the eight commands for the Digital Phase-Locked Loop. A channel or hardware reset disables the DPLL, resets the missing clock latches, sets the source to the RTxC pin and selects NRZI mode. The Enter Search Mode command enables the DPLL after a reset.

D7	D6	D5	Command
0	0	0	Null command
0	0	1	Enter search mode
0	1	0	Reset clock missing
0	1	1	Disable DPLL
1	0	0	Set source = BR generator
1	0	1	Set source = RTxC
1	1	0	Set FM mode
1	1	1	Set NRZI mode

Figure 4-37. Digital Phase-Locked Loop Command Bits

(000) Null Command. This command has no effect on the DPLL.

(001) Enter Search Mode. Issuing this command causes the DPLL to enter the Search mode, where the DPLL searches for a locking edge in the incoming data stream. The action taken by the DPLL upon receipt of this command depends on the operating mode of the DPLL.

In NRZI mode, the output of the DPLL is waiting for an edge in the incoming data stream. After the Search mode is entered, the first edge the DPLL sees is assumed to be a valid data edge, and the DPLL begins the clock recovery operation from that point. The DPLL clock rate must be 32 times the data rate in NRZI mode. Upon leaving the Search mode, the first sampling edge of the DPLL occurs 16 of these 32x clocks after the first data edge and the second sampling edge occurs 48 of these 32x clocks after the first data edge. Beyond this point, the DPLL begins normal operation, adjusting the output to remain in sync with the incoming data.

In FM mode, the output of the DPLL is Low while the DPLL is waiting for an edge in the incoming data stream. The first edge the DPLL detects is assumed to be a valid clock edge. For this to be the case, the line must contain only clock edges; i.e., with FM1 encoding, the line must be continuous "0s". With FM0 encoding the line must be continuous "1"s, whereas Manchester encoding requires alternating "1s" and "0s" on the line. The DPLL clock rate must be 16 times the data rate in FM mode. The DPLL output causes the receiver to sample the data stream in the nominal center of the two halves of the bit cell to decide whether the data was a "1" or a "0". After this command is issued, as in NRZI mode, the DPLL starts sampling immediately after the first edge is detected. (In FM mode, the DPLL examines the clock edge of every other bit cell to decide what correction must be made to remain in sync.) If the DPLL does not see an edge during the expected window, the One Clock Missing bit in RR10 is set. If the DPLL does not see an edge after two successive attempts, the Two Clocks Missing bit in RR10 is set and the DPLL automatically enters the Search mode. This command resets both Clock Missing latches.

(010) Reset Clock Missing. Issuing this command resets the Clock Missing latches in RR10, and forces a continuous Search mode state.

(011) Disable DPLL. Issuing this command disables the DPLL.

(100) Set Source = Baud Rate Generator. Issuing this command forces the clock for the DPLL to come from the output of the baud rate generator.

(101) Set Source = RTxC. Issuing this command forces the clock for the DPLL to come from the RTxC pin. This mode is selected by a channel or hardware reset.

(110) Set FM Mode. This command forces the DPLL to operate in the FM mode and is used to recover the clock from FM or Manchester-encoded date. (Manchester is decoded by placing the receiver in NRZ mode while the DPLL is in FM mode.)

(111) Set NRZI Mode. Issuing this command forces the DPLL to operate in the NRZI mode. This mode is also selected by a hardware or channel reset.

D4 -- Local Loopback. Setting this bit to "1" selects the Local Loopback mode of operation. In this mode, the internal transmitted data is routed back to the receiver, as well as to the TxD pin. Whenever a byte is transmitted out, it is also routed to the Receive buffer. The receiver ignores external data at the RxD input and CTS and DCD inputs are disabled. However, a change in CTS or DCD can still generate an interrupt.

D3 -- Auto Echo. When this bit is set to "1", data is received at the RxD input and echoed out the transmit data pin. Only data received at the RxD input is transmitted at the TxD output. CTS is disabled as a transmit enable, but DCD can still be utilized as the receiver enable.

D2 -- DTR Enable. This bit should be reset to "0" to allow the output handshaking signal DSR (DTR when jumpered for DTE operation) to follow the state of the Data Terminal Ready (DTR) bit in WR5.

D1 -- Baud Rate Generator Source. This bit selects the source of the clock for the baud rate generator. If this bit is set to "1", the clock is sourced from the 4.9152 MHz PCLK input. If this bit is reset to "0", the baud rate generator source is the receive clock input.

D0 -- Baud Rate Generator Enable. When this bit is set to "1", the baud rate generator is enabled. When reset to "0", the baud rate generator is disabled. This bit is reset by a hardware reset.

Write Register 15

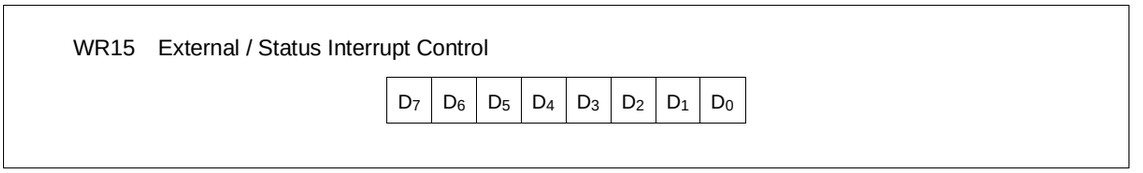


Figure 4-38. Write Register 15

WR15 is the External/Status Source Control register. If the External/Status interrupts are enabled as a group via WR1, bits in this register control which External/Status conditions can cause an interrupt. Only the External/Status conditions that occur after the controlling bits are set to "1" will cause an interrupt. This is true even if an External/Status condition is pending at the time the bit is set.

D7 -- Break/Abort Interrupt Enable. If this bit is set to "1", a change in the Break/Abort status of the receiver causes an External/Status interrupt. This bit is set by a channel or hardware reset.

D6 -- Tx Underrun/EOM Interrupt Enable. If this bit is set to "1", a change of state by the Tx Underrun/EOM latch in the transmitter causes an External/Status interrupt. This bit is set to "1" by a channel or hardware reset.

D5 -- CTS Interrupt Enable. If this bit is set to "1", a change of state on the RTS input handshaking signal (CTS when jumpered for DTE operation) causes an External/Status interrupt. This bit is set by a channel or hardware reset.

D4 -- External/Hunt Interrupt Enable. In Asynchronous modes when this bit is set to "1", an external interrupt at the user interrupt connector (J1) causes an External/Status interrupt. In Synchronous modes if D4 is set, an External/Status interrupt is generated when the hunt bit in the receiver changes state.

D3 -- DCD Interrupt Enable. If this bit is set to "1", a change of state on the DTR input handshaking signal (DSR when jumpered for DTE operation) causes an External/Status interrupt. This bit is set by a channel or hardware reset.

D2 -- Not Used. This bit is not used and must be reset to "0".

D1 -- Zero Count Interrupt Enable. If this bit is set to "1", an External/Status interrupt is generated whenever the counter in the baud rate generator reaches a zero count. This bit is reset to "0" by a channel or hardware reset.

D0 -- Not Used. This bit is has no function in the SCC and must be reset to "0".

SCC Read Registers

The SCC contains seven read registers in each channel. In addition there are two registers which are shared by both channels. The status of these registers is continually changing and depends on the mode of communication, received and transmitted data, and the manner in which this data is transferred to and from the CPU.

Read Register 0

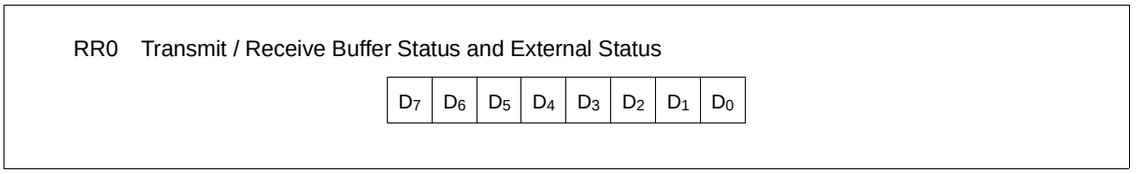


Figure 4-39. Read Register 0

RR0 contains the status of the Receive and Transmit buffers. RR0 also contains the status bits for the six sources of External/Status interrupts.

D7 -- Break/Abort. In the Asynchronous mode, this bit is set when a "break" (i.e., continuous zeros, null character plus framing error) is detected in the receive data stream. This bit is reset when the break condition goes away, leaving a single null character (00H) in the Receive FIFO. This character should be read and discarded. In SDLC mode, this bit is set by the detection of an Abort sequence (seven or more "1s"), then reset automatically at the termination of the Abort sequence. In either case, if the Break/Abort Interrupt Enable bit is set, an External/Status interrupt is initiated. Unlike the remainder of the External/Status bits, both transitions are guaranteed to cause an External/Status interrupt, even if another External/Status interrupt is pending at the time these transitions occur. This procedure is necessary because Abort or Break conditions may not persist.

D6 -- Transmit Underrun/EOM. This bit is set to "1" by a channel or hardware reset and when the transmitter is disabled or a Send Abort command (WR0) is issued. This bit can only be reset by the Reset Tx Underrun/EOM Latch command (WR0). When a Transmit Underrun occurs, this bit is set and causes an External/Status interrupt (assuming the Tx Underrun/EOM Interrupt Enable bit is set within WR15). This bit operates only in Synchronous modes.

D5 -- Clear To Send. This bit shows the status of the RTS input handshaking signal (CTS when jumpered for DTE operation) at the time of the first change of any of the five External/Status bits (DCD, CTS, Sync/Hunt, External Interrupt, Break/Abort, or Tx Underrun/EOM) after power-on, channel reset, or an External/Status interrupt. The change of some of these events latches the status bits within RR0. To get the current state of the RTS/CTS handshaking signal, this bit must be read immediately following a Reset External/Status Interrupt command (command 2, WR0). Any odd number of transitions while another External/Status interrupt is pending also causes an External/Status interrupt condition. A transition of the input handshaking signal causes an interrupt if external interrupts are enabled with bit D0 of WR1. An External/Status vector is sent if the Status Affects Vector bit is programmed with bit D0 of WR9. This bit reads as "1" when the RTS/CTS handshaking signal is active (ready or RS-232 +V); "0" when inactive (busy or RS-232 −V). See the Handshaking Signals code example on page 5-2 for an example of how to use this bit.

D4 -- Sync/Hunt. In Synchronous modes, this bit is reset when character synchronization is achieved and is set by writing the Enter Hunt Mode bit. In Asynchronous mode, this bit is identical in operation to the CTS bit, except it reports the state of the external interrupt signal applied to J1.

D3 -- Data Carrier Detect. This bit shows the status of the DTR input handshaking signal (DSR when jumpered for DTE operation) at the time of the first change of any of the five External/Status bits (DCD, CTS, Sync/Hunt, External Interrupt, Break/Abort, or Tx Underrun/EOM) after power-on, channel reset, or an External/Status interrupt. The change of some of these events latches the status bits within RR0. To get the current state of the DTR/DSR handshaking signal, this bit must be read immediately following a Reset External/Status Interrupt command (command 2, WR0). Any odd number of transitions while

another External/Status interrupt is pending also causes an External/Status interrupt condition. A transition of the handshaking signal causes an interrupt if external interrupts are enabled with bit D0 of WR1. An External/Status vector is sent if the Status Affects Vector bit is programmed with bit D0 of WR9. This bit reads as "1" when the DTR/DSR handshaking signal is active (ready or RS-232 +V); "0" when inactive (busy or RS-232 −V). See the Handshaking Signals code example on page 5-2 for an example of how to use this bit.

D2 -- Transmit Buffer Empty. This bit is set to "1" when the Transmit buffer is empty (ready to accept another character from the CPU) except during CRC transmission in Synchronous mode. If Tx interrupts are enabled, an interrupt is generated when the Transmit buffer goes empty. The interrupt will modify the vector written to WR2 if the Status Affects Vector bit is programmed in WR9. See the Transmit / Receive code example on page 5-2 for an example of how to use this bit.

D1 -- Zero Count. This bit is set when the baud rate generator count reaches zero. An External/Status interrupt is generated if no other interrupts are pending at the time this bit is set. However, if there is another External/Status interrupt condition pending, no interrupt is initiated until interrupt servicing is complete. In polled applications, check the Interrupt Pending bit in RR3 (read from channel A or C only) for a status change and then proceed as though servicing an interrupt.

D0 -- Receive Character Available. This bit is set to "1" when a character is available in the Receive buffer. Each receiver is quadruply buffered (three Storage registers and an Input Shift register) to allow additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. This bit is reset to "0" automatically when the Receive buffers are all empty. Error status reflects the current byte in the Receive buffer. When this bit is set, RR1 should be read and stored to avoid losing any error information. The received data can then be read. Error information does not need to be read when using vectored interrupts, since a modified vector is generated if there is an error. Since the receiver port has three Storage registers, three characters can be contained in the SCC without a read. A fourth character received without a read causes an overrun error in the last Receiver Storage register. See the Transmit / Receive code example on page 5-2 for an example of how to use this bit.

Read Register 1

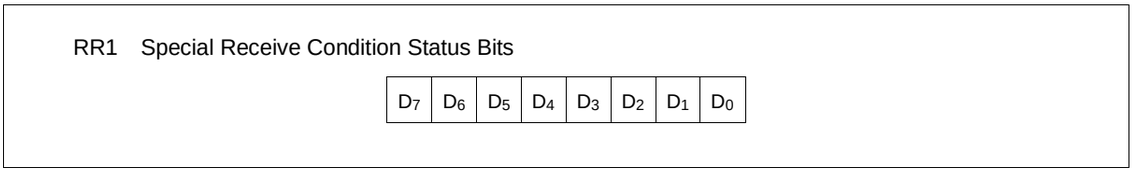


Figure 4-40. Read Register 1

RR1 contains the Special Receive condition status, Residue Codes for the I-field in SDLC mode, and various error conditions.

D7 -- End of Frame (SDLC). This bit is used only in SDLC mode and indicates that a valid closing flag has been received and that the CRC Error bit and residue codes are valid. This bit can be reset by issuing the Error Reset command (command 6, WR0). It is also updated by the first character of the following frame. This bit is reset in any mode other than SDLC.

D6 -- CRC/Framing Error. In Asynchronous mode if a framing error occurs this bit is set (but not latched) for the receive character in which the framing error occurred. Detection of a framing error adds an additional one-half bit to the character time so that the framing error is not interpreted as a new Start bit. In Synchronous and SDLC modes, this bit indicates the result of comparing the CRC checker to the appropriate check value. This bit is reset by issuing an Error Reset command (command 6, WR0), but the bit is never latched. Therefore, it is always updated when the next character is received. When used for CRC error status in Synchronous or SDLC modes, this bit is usually set since most bit combinations, except for a correctly completed message, result in a non-zero CRC.

D5 -- Receiver Overrun Error. This bit indicates that more than three characters have been received without a read from the CPU. Only the character that has been written over is flagged with this error,

when the overrun character is read this bit is latched to "1", until reset by the Error Reset command (command 6, WR0). If the Status Affects Vector bit is enabled (D0, WR9), the character that has been overrun interrupts with the Special Receive Condition vector.

D4 -- Parity Error. When parity is enabled (D0, WR9), this bit is set for the characters whose parity does not match the programmed sense (even/odd). This bit is latched so that once an error occurs, it remains set until the Error Reset command (command 6, WR0) is issued. If the Parity Is Special Condition (D2, WR1) bit is set, a parity error causes a Special Receive Condition vector to be returned on the character containing the error and on all subsequent characters until the Error Reset command is issued.

D3, D2, D1 -- Residue Codes. In those cases in SDLC mode where the received I-Field is not an integral multiple of the programmed character length, these three bits indicate the length of the I-Field and are meaningful only for the transfer in which the end of frame bit is set. This field is set to "011" by a channel or hardware reset and is forced to this state in Asynchronous mode. These three bits can leave this state only if SDLC is selected and a character is received.

D3	D2	D1	I-Field Bits in Last Byte	I-Field Bits in Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

Figure 4-41. I-Field Bit Selection (8 Bits Only)

I-Field bits are right justified in all cases. If a receive character length other than eight bits is used for the I-Field, the table below is used.

D3	D2	D1	Bits/Character
0	1	1	8
0	0	0	7
0	1	0	6
0	0	1	5

Figure 4-42. Residue Bits/Character

D0 -- All Sent. This bit is set when all characters have completely cleared the transmitter. Interrupts are not caused by transitions of this bit. This bit is always set in Synchronous and SDLC modes.

Read Register 2 (Channels A and C only)

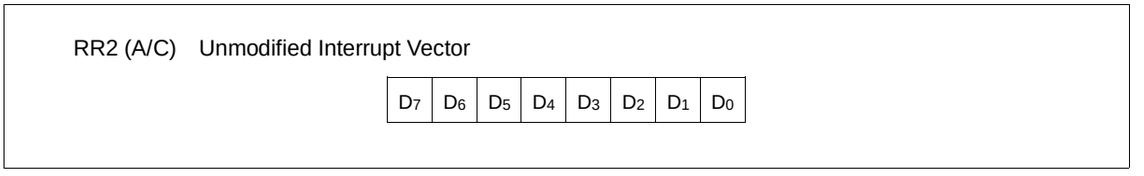


Figure 4-43. Read Register 2 (A/C)

When read from channel A or C, this register contains the same interrupt vector as written into WR2.

Read Register 2 (Channels B and D only)

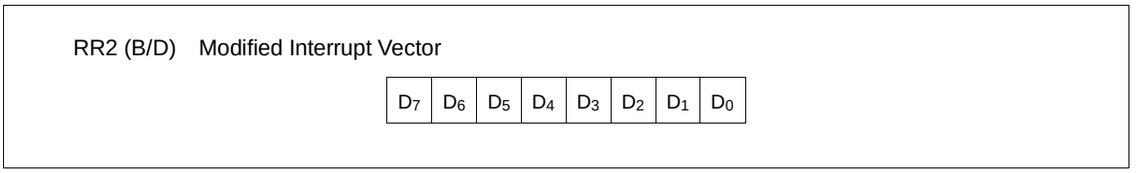


Figure 4-44. Read Register 2 (B/D)

When read from channel B or D, RR2 contains the modified interrupt vector. When the Status Affects Vector bit is set, the base interrupt vector written into WR2 is modified to reflect changing status conditions within the SCC. This modified vector allows direct hardware vectoring to unique interrupt handling routines for various SCC conditions. The vector address can represent a restart instruction (for Z80 CPUs) or index an interrupt vector table in memory that contains the actual starting address location of the interrupt service routine. RR2 can be read to support polled interrupt schemes.

Read Register 3

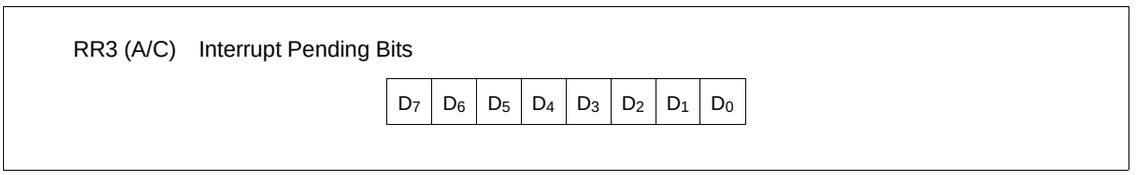


Figure 4-45. Read Register 3 (A/C)

RR3 contains the status of each Interrupt Pending bit. If this register is read from channels B or D, all zeros are returned.

Bit	Interrupt Source Pending
D7	0
D6	0
D5	Channel A/C Receive
D4	Channel A/C Transmit
D3	Channel A/C External/Status
D2	Channel B/D Receive
D1	Channel B/D Transmit
D0	Channel B/D External/Status

Figure 4-46. Interrupt Pending Bits

Read Register 8

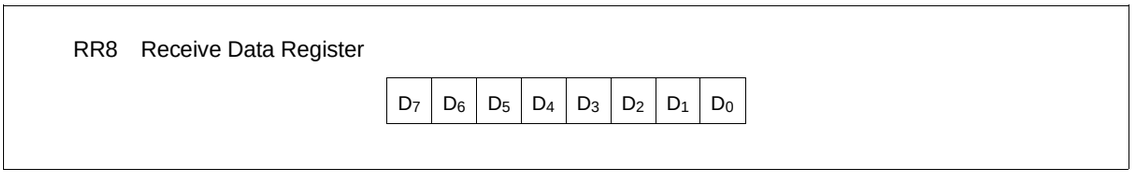


Figure 4-47. Read Register 8

RR8 is the Receive Data register. In most cases, the Receive Data register is accessed by directly reading the RCV Data port address assigned to each channel. See page 4-3 for further information.

Read Register 10

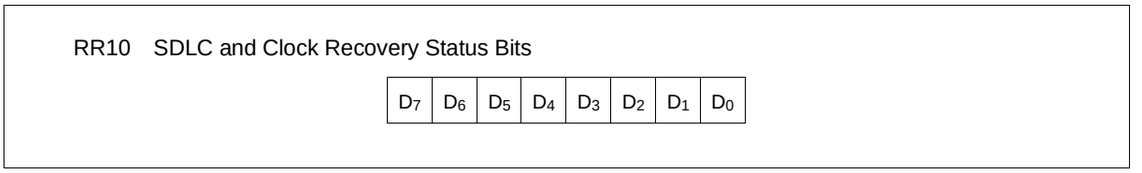


Figure 4-48. Read Register 10

RR10 contains some miscellaneous status bits. Unused bits are always "0".

D7 -- One Clock Missing. While operating in the FM mode, the DPLL sets this bit to "1" when it does not see a clock edge within the window where it expects one. This bit is latched until reset by a Reset Missing Clock or Enter Search Mode command in WR14.

D6 -- Two Clocks Missing. While operating in the FM mode, the DPLL sets this bit to "1" when it does not see a clock edge in two successive tries. At the same time the DPLL enters the Search mode. This bit is latched until reset by a Reset Missing Clock or Enter Search Mode command in WR14. In the NRZI mode of operation and while the DPLL is disabled, this bit is always "0".

D5 -- Not used. This bit is not used, it always reads "0".

D4 -- Loop Sending. This bit is set to "1" in SDLC Loop mode while the transmitter is in control of the Loop, that is, while the SCC is actively transmitting on the loop. This bit is reset at all other times. It can also be polled in SDLC mode to determine when the closing flag has been sent.

D3 -- Not used. This bit is not used, it always reads "0".

D2 -- Not used. This bit is not used, it always reads "0".

D1 -- On Loop. This bit is set to "1" while the SCC is actually on loop in SDLC Loop mode. This bit is set to "1" in the X.21 mode (Loop mode selected while in monosync) when the transmitter goes active. This bit is "0" at all other times. It can also be polled in SDLC mode to determine when the closing flag has been sent.

D0 -- Not used. This bit is not used, it always reads "0".

Read Register 12

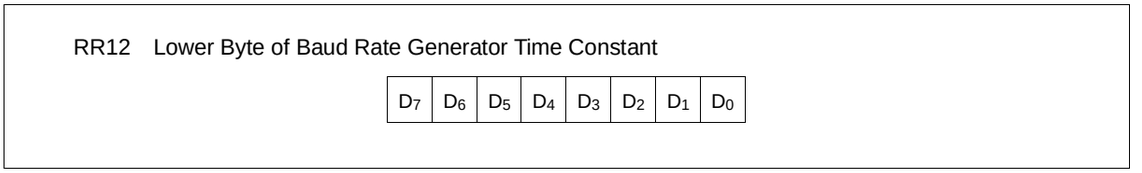


Figure 4-49. Read Register 12

RR12 returns the value stored in WR12, the lower byte of the time constant for the baud rate generator.

Read Register 13

RR13 Upper Byte of Baud Rate Generator Time Constant

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Figure 4-50. Read Register 13

RR13 returns the value stored in WR13, the upper byte of the time constant for the baud rate generator.

Read Register 15

RR15 External / Status Interrupt Enable Bits

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Figure 4-51. Read Register 15

RR15 reflects the value stored in WR15, the External/Status Interrupt Enable bits. The two unused bits, D0 and D2 are always returned as "0".

External / Status Bit	Interrupt Source
D7	Break/Abort
D6	Tx Underrun/EOM
D5	CTS
D4	SYNC/Hunt
D3	DCD
D2	0
D1	Zero Count
D0	0

Figure 4-52. External / Status Interrupt Enable Bits

SCC Operation

Initialization

This section describes general initialization concepts. Also shown are specific initialization examples for operation in Asynchronous and Synchronous modes.

Generic Initialization

The SCC is initialized in three distinct stages; first, basic operating modes are defined (i.e. Asynchronous mode, data bits, parity, etc.), this is followed by hardware configuration (transmitter enable, receiver enable, baud rate, etc.). Operating modes should be initialized before hardware functions are enabled. In the third stage, interrupts, if used, are enabled. The following table shows the sequence in which SCC registers should be initialized. See the code example on page 5-1 for further information.

	Register	D7	D6	D5	D4	D3	D2	D1	D0	Comments
Stage 1.	**Modes and Constants**									
	WR9	X	X	0	0	0	0	0	0	Channel reset
	WR4	X	X	X	X	X	X	X	X	Tx/Rx control, Async/Sync
	WR2	X	X	X	X	X	X	X	X	Interrupt vector (optional)
	WR3	X	X	X	X	X	X	X	0	Receiver disable
	WR5	X	X	X	X	0	X	X	X	Transmitter disable
	WR6	X	X	X	X	X	X	X	X	Sync character
	WR7	X	X	X	X	X	X	X	X	Sync character
	WR9	0	0	0	X	0	X	0	X	Interrupt control. Keep interrupts disabled at this time
	WR10	X	X	X	X	X	X	X	X	Miscellaneous control (optional)
	WR11	0	X	X	X	X	1	X	X	Clock control
	WR12	X	X	X	X	X	X	X	X	Baud rate lower byte
	WR13	X	X	X	X	X	X	X	X	Baud rate upper byte
	WR14	X	X	X	X	X	X	X	0	Miscellaneous control. Keep baud rate generator disabled at this time
	WR14	X	X	X	S	S	S	S	0	This register requires multiple writes if more than one command is used
Stage 2.	**Enables**									
	WR3	S	S	S	S	S	S	S	1	Receiver enable
	WR5	S	S	S	S	1	S	S	S	Transmitter enable
	WR0	1	0	0	0	0	0	0	0	Reset Transmit CRC generator
	WR14	0	0	0	S	S	S	S	1	Enable baud rate generator
Stage 3.	**Interrupt Enables (optional)**									
	WR15	X	X	X	X	X	0	X	0	Enable external interrupts
	WR0	0	0	0	1	0	0	0	0	Reset External/Status twice
	WR0	0	0	0	1	0	0	0	0	Reset External/Status twice
	WR1	0	0	0	X	X	0	X	X	Enable receive, transmit, and external interrupts
	WR9	0	0	0	S	1	S	0	S	Enable Master Interrupts

1 Set to "1".
0 Reset to "0".
X Varies.
S Same as previously programmed.

Figure 4-53. Generic SCC Initialization Sequence

Asynchronous Initialization

The following is the initialization sequence for polled mode (non-interrupt), 16x baud rate generator clocked, Asynchronous operation at 9600 baud, even parity, 7 data bits, and 1 stop bit. All output handshaking signals are set to their active state (RS-232 +V), input handshaking signals are ignored.

	Register	D7	D6	D5	D4	D3	D2	D1	D0	Comments
Stage 1.	**Modes and Constants**									
	WR9	1	0	0	0	0	0	0	0	Reset channel A
	WR4	0	1	0	0	0	1	1	1	16x clock mode 1 stop bit Parity enable Even parity
	WR3	0	1	0	0	0	0	0	0	Receive 7 bits/character Prevent input handshaking signals from affecting receiver or transmitter Keep receiver disabled for now
	WR11	0	1	0	1	0	1	1	1	Use baud rate generator as transmitter and receiver timebase Don't output Tx clock signal, this reduces ±12V power
	WR12	0	0	0	0	1	1	1	0	Baud rate lower byte for 9600 baud @ 16x clock mode
	WR13	0	0	0	0	0	0	0	0	Baud rate upper byte for 9600 baud @ 16x clock mode
	WR14	0	0	0	0	0	0	1	0	Allow control of the DSR/DTR output handshaking signal via WR5 Use the 4.9152 MHz oscillator (PCLK input) as the timebase for BR gen. Keep baud rate generator disabled for now
Stage 2.	**Enables**									
	WR3	0	1	0	0	0	0	0	1	Receiver enable
	WR5	1	0	1	0	1	0	1	0	Set DSR/DTR output handshaking signal ready (RS-232 +V) Transmit 7 bits/character Do not send line break Transmitter enable Set CTS/RTS output handshaking signal ready (RS-232 +V)
	WR14	0	0	0	0	0	0	1	1	Enable baud rate generator

Figure 4-54. Asynchronous Initialization Sequence

Synchronous Initialization

The following is the initialization sequence for 9600 baud, synchronous SDLC operation using transmit and receive interrupts.

Register	D7	D6	D5	D4	D3	D2	D1	D0	Comments
Stage 1. Modes and Constants									
WR9	1	0	0	0	0	0	0	0	Reset channel A
WR4	0	0	1	0	0	0	0	0	1x clock mode SDLC mode No parity
WR2	x	x	x	x	x	x	x	x	Interrupt vector (user selected)
WR3	1	1	0	0	1	1	0	0	Receive 8 bits/character Prevent input handshaking signals from affecting receiver or transmitter Do not enter hunt mode Enable CRC on received characters Enable address search mode Keep receiver disabled for now
WR5	0	1	1	0	0	0	0	0	Set DSR/DTR output handshaking signal busy RS-232 −V) Transmit 8 bits/character Do not send line break Keep transmitter disabled for now Use SDLC / CRC polynomial Set CTS/RTS output handshaking signal busy (RS-232 −V) Disable transmission of CRC
WR6	x	x	x	x	x	x	x	x	SDLC address (user selected)
WR7	0	1	1	1	1	1	1	0	SDLC flag character
WR9	0	0	0	0	0	0	0	1	Modify interrupt vector with status information
WR10	1	0	0	0	0	0	0	0	Preset CRC checker / generator with 1's Use NRZ data encoding When line is idle send flags Send CRC on transmit underrun Don't use SDLC loop mode
WR11	0	0	0	1	0	1	1	0	Rx clock sourced from TCLK (RCLK when jumpered for DTE operation) Transmit clock sourced from baud rate generator Output BR gen. on RCLK (TCLK when jumpered for DTE operation)
WR12	1	1	1	1	1	1	1	0	Baud rate lower byte for 9600 baud @ 1x clock mode
WR13	0	0	0	0	0	0	0	0	Baud rate upper byte for 9600 baud @ 1x clock mode
WR14	0	0	0	0	0	0	1	0	Disable local loopback Disable auto echo Allow control of the DSR/DTR ouput handshaking signal via WR5 Use the 4.9152 MHz oscillator (PCLK input) as the timebase for BR gen. Keep baud rate generator disabled for now
Stage 2. Enables									
WR3	1	1	0	0	0	0	0	1	Enable receiver
WR5	0	1	1	0	1	0	0	1	Enable transmitter Enable transmission of CRC
WR0	1	0	0	0	0	0	0	0	Reset transmit CRC generator
WR14	0	0	0	0	0	0	1	1	Enable baud rate generator

Figure 4-55. Synchronous Initialization Sequence (continued next page)

Stage 3. Interrupt Enables

WR15	0	1	0	0	0	0	0	0	Enable Tx underrun / EOM interrupts
WR0	0	0	0	1	0	0	0	0	Reset External / Status interrupts twice
WR0	0	0	0	1	0	0	0	0	Reset External / Status interrupts twice
WR1	0	0	0	1	0	0	1	0	Enable receive interrupts Enable transmit interrupts

Figure 4-56. Synchronous Initialization Sequence (Cont.)

Baud Rate Selection

The individual baud rates for each channel are selected by programming the WR12 and WR13 time constants. For example, to operate Channel A in Asynchronous mode at 9600 baud, write 0EH to WR12, and write 00H to WR13.

The formulas for calculating time constant and baud rate values are given by:

$$Time\ Constant = \left[\frac{4.9162 \times 10^6}{2 \times (baud\ rate) \times (clock\ mode)}\right] - 2$$

or

$$Baud\ Rate = \frac{4.9162 \times 10^6}{(2 \times time\ constant \times clock\ mode) + (4 \times clock\ mode)}$$

Where:

baud rate = desired baud rate
clock mode = 1, 16, 32, or 64

Figure 4-57. Baud Rate Formulas

The table below gives time constants for several standard baud rates.

	SYNCHRONOUS (x1 Clock)			ASYNCHRONOUS (x16 Clock)		
Baud Rate	**Decimal**	**Hex**	**Error**	**Decimal**	**Hex**	**Error**
76.8K[1]	30	001E	0	0	0000	0
38.4K[1]	62	003E	0	2	0002	0
19.2K	126	007E	0	6	0006	0
9600	254	00FE	0	14	000E	0
4800	510	01FE	0	30	001E	0
2400	1022	03FE	0	64	0040	0
1800	1361	0551	0.17	83	0053	0.39
1200	2046	07FE	0	126	007E	0
600	4094	07FE	0	254	00FE	0
300	8190	1FFE	0	510	01FE	0
150	16382	3FFF	0	1022	03FE	0
110	22340	5744	.0008	1394	0572	.026

[1] The maximum data rate according to RS-232-C specifications is 20K baud.

Figure 4-58. Baud Rate Generator Time Constants

External Interrupts

When operated in Asynchronous mode, vectored interrupt requests can be generated by applying low level TTL signals to the INTA*, INTB*, INTC*, or INTD* inputs on connector J1 (pins 1, 3, 5, or 7). In Synchronous modes, these inputs serve as character synchronization inputs. To use external interrupts the following steps should be taken.

Enables and initialization

- Program a base interrupt vector into WR2.
- Enable external interrupts by setting bit D4 within WR15.
- Reset External/Status interrupts twice. WR0, command 2.
- Set the Master Interrupt Enable bit for External/Status interrupts by setting bit D0 within WR1.
- Set the SCC Master Interrupt Enable bit by setting bit D3 within WR9.

Interrupt servicing

The interrupt service routine for the external interrupt request should at least perform the following actions.

- Reset External/Status Interrupt command. WR0, command 2.
- Reset Highest IUS. WR0, command 7.
- Any action which returns the requesting signal back to a high level.

Software Examples

This section shows some software examples to assist you in constructing your own software routines. These examples were assembled with the Microsoft Macro Assembler 5.0 for use with VersaLogic's 80188 CPU card, VL-188.

Asynchronous Code Example

Initialization

The following example initializes channel A for asynchronous operation at 9600 baud, 8 data bits, even parity, and 1 stop bit.

The key user subroutine is:

INIT : Initializes channel A.

```
0000              code    segment                 ;Microsoft Macro Assembler 5.0
0000                      org     0000h
                          assume  cs:code
                                                  ;PORT ADDRESSES
= 00B0            a_data  equ     00B0h           ;Chan. A (SCC#1) Data register
= 00B1            a_cmd   equ     00B1h           ;Chan. A (SCC#1) read/write register 0

0000              init:                           ;CHANNEL A INITIALIZATION
0000  EB 18 90              jmp     transfer      ;Jump past table

0003  16          table   db      offset eot - offset table-1
0004  04 47                db      04,047h         ;WR4  x16, 1 Stop bit, Even Parity
0006  02 00                db      02,000h         ;WR2  Base interrupt vector = 00H
0008  03 E0                db      03,0E0h         ;WR3  Auto enables, Rx = 8 bits
000A  05 E2                db      05,0E2h         ;WR5  Tx = 8 bits, DTR & RTS = On
000C  0B 56                db      11,056h         ;WR11 Clock sources
000E  0C 0E                db      12,00Eh         ;WR12 Baud rate low  = 9600 Baud
0010  0D 00                db      13,000h         ;WR13 Baud rate high = 9600 Baud
0012  0E 02                db      14,002h         ;WR14 PCLK = baud rate timebase
0014  03 E1                db      03,0E1h         ;WR3  Enable Rx
0016  05 EA                db      05,0EAh         ;WR5  Enable Tx
0018  0E 03                db      14,003h         ;WR14 Enable baud rate generator
001A              eot     label   byte            ;End of table

001A              transfer:                       ;Issue channel reset command
001A  BA 00B1               mov     dx,a_cmd
001D  B0 09                 mov     al,09h          ;Point to WR9 via WR0
001F  EE                    out     dx,al
0020  B0 80                 mov     al,80h          ;Channel A reset command
0022  EE                    out     dx,al
0023  FC                   cld                     ;Direction flag = forward
0024  BE 0003 R            mov     si,offset table
0027  2E: AC                lods    table           ;Fetch table length into CX
0029  8A C8                 mov     cl,al
002B  B5 00                 mov     ch,00h
002D  BA 00B1               mov     dx,a_cmd        ;DX = Port address of WR0
0030  2E: AC      repeat:   lods    table           ;Fetch table entry
0032  EE                    out     dx,al
0033  E2 FB                 loop    repeat
0035  C3                    ret
```

Transmit / Receive

The following example illustrates low-level asynchronous transmission and reception routines for polled-mode character input/output through channel A.

The key user subroutines are:

READ_CHAR : Fetches received character to AL register.

RX_STATUS : Used to determine if new characters need to be read using Read_Char subroutine, results returned in Z flag.

WRITE_CHAR : Transmits character in AL register.

```
                                                ;Microsoft Macro Assembler 5.0

0036                read_char:                  ;Exit with received character
                                                ;in AL register.  Flag register
                                                ;is altered

0036  E8 0042 R             call    rx_status      ;See if character has arrived
0039  74 FB                 jz      read_char      ;If not... keep testing
003B  52                    push    dx
003C  BA 00B0               mov     dx,a_data
003F  EC                    in      al,dx          ;Read character, return in AL
0040  5A                    pop     dx
0041  C3                    ret

0042                rx_status:                  ;Checks to see if a received
0042  52                    push    dx             ;character is waiting to be
0043  BA 00B1               mov     dx,a_cmd       ;read.  Z=1 if no character,
0046  EC                    in      al,dx          ;Z=0 if character is waiting
0047  A8 01                 test    al,00000001b
0049  5A                    pop     dx
004A  C3                    ret

004B                write_char:                 ;Transmit character in AL
                                                ;register.  Flag register is
                                                ;altered
004B  52                    push    dx
004C  50                    push    ax             ;Save character for later output

004D  BA 00B1               mov     dx,a_cmd       ;Check to see if Transmit Buffer
0050  EC            wc1:    in      al,dx          ;is empty
0051  A8 04                 test    al,00000100b
0053  74 FB                 jz      wc1            ;If not... keep testing

0055  BA 00B0               mov     dx,a_data      ;Restore output character
0058  58                    pop     ax             ;and transmit
0059  EE                    out     dx,al
005A  5A                    pop     dx
005B  C3                    ret

005C                code    ends
                            end
```

Handshaking Signals

The following code example shows how to read the RTS and DTR handshaking signals (CTS and DSR when jumpered for DTE operation). Also shown are examples of how to assert CTS (RTS) and DSR (DTR).

Note: When manipulating the bits D1 and D7 within WR5, care must be taken not to modify other bits within WR5 which control other aspects of the serial channel. Since WR5 cannot be read (there is no RR5), a binary image of WR5 should be maintained in RAM in order to reference existing WR5 contents. This reserved variable (labeled PATTERN in the example below) must be updated whenever WR5 is updated. Bit manipulation is performed on this byte-variable, and a copy of it is then output to WR5. Do not forget to assign an initial value to this variable when the channel is initialized.

The key user subroutines are:

RTS_STATUS : Returns logic state of Channel A RTS input signal (CTS when jumpered for DTE operation), results reflected in zero flag.

DTR_STATUS : Returns logic state of Channel A DTR input signal (DSR when jumpered for DTE operation), results reflected in zero flag.

CTS_HI : Asserts an active signal level (ready or RS-232 +V) on the Channel A CTS output signal (RTS when jumpered for DTE operation).

CTS_LO : Asserts an inactive signal level (busy or RS-232 −V) on the Channel A CTS output signal (RTS when jumpered for DTE operation).

DSR_HI : Asserts an active signal level (ready or RS-232 +V) on the Channel A DSR output signal (DTR when jumpered for DTE operation).

DSR_LO : Asserts an inactive signal level (busy or RS-232 −V) on the Channel A DSR output signal (DTR when jumpered for DTE operation).

```
                                                ;Microsoft Macro Assembler 5.0

0000                    data    segment
0000                            org     0000h           ;Allocate a variable to hold the
0000  00                pattern db      00h             ;initial value of WR5 established
0001                    data    ends                    ;during channel initialization

0000                    code    segment
                                assume  cs:code
0000                            org     0000h

                                                        ;PORT ADDRESSES
= 00B0                  a_data  equ     00B0h           ;Chan. A (SCC#1) Data register
= 00B1                  a_cmd   equ     00B1h           ;Chan. A (SCC#1) read/write
                                                        ;register 0

                        init:   .
                                .
                                .
                                mov     pattern,0EAh    ;At end of channel initialization
                                .                       ;save a copy of the binary value
                                .                       ;sent to WR5
                                .
                                ret

0000                    rts_status:                     ;Read Channel A RTS
                                                        ;Zero flag = logic state of
                                                        ;RTS upon exit.  Use conditional
                                                        ;jump instructions JZ or JNZ as
                                                        ;appropriate to control program
                                                        ;flow after calling this
                                                        ;subroutine.
                                                        ;Zero=0 (JNZ) RTS ON (RS-232+12V)
                                                        ;Zero=1 (JZ)  RTS OFF(RS-232-12V)
```

```
0000  50                      push    ax              ;Save registers
0001  52                      push    dx

0002  BA 00B1                mov     dx,a_cmd        ;Reset External/Status Interrupts
0005  B0 10                   mov     al,10h          ;via WR0 causing RTS status to
0007  EE                      out     dx,al           ;be latched into RR0

0008  EC                      in      al,dx           ;Read CTS bit in RR0
                                                      ;to fetch RTS status

0009  A8 20                   test    al,00100000b    ;Test bit position D5

000B  5A                      pop     dx              ;Restore registers
000C  58                      pop     ax

000D  C3                      ret

000E                  dtr_status:                     ;Read Channel A DTR
                                                      ;Zero flag = logic state of
                                                      ;DTR upon exit.  Use conditional
                                                      ;jump instructions JZ or JNZ as
                                                      ;appropriate to control program
                                                      ;flow after calling this
                                                      ;subroutine.
                                                      ;Zero=0 (JNZ) DTR ON (RS-232+12V)
                                                      ;Zero=1 (JZ)  DTR OFF(RS-232-12V)

000E  50                      push    ax              ;Save registers
000F  52                      push    dx

0010  BA 00B1                mov     dx,a_cmd        ;Reset External/Status Interrupts
0013  B0 10                   mov     al,10h          ;via WR0 causing DTR status to
0015  EE                      out     dx,al           ;be latched into RR0

0016  EC                      in      al,dx           ;Read DCD bit in RR0
                                                      ;to fetch DTR status

0017  A8 08                   test    al,00001000b    ;Test bit position D3

0019  5A                      pop     dx              ;Restore registers
001A  58                      pop     ax

001B  C3                      ret

001C                  cts_hi:                         ;Channel A CTS signal true.
                                                      ;Causing it to assume RS-232 +12V

001C  50                      push    ax              ;Save registers
001D  52                      push    dx

001E  80 0E 0000 R 02         or      pattern,02h     ;Set bit D1 in variable

0023  BA 00B1                mov     dx,a_cmd        ;Point to WR5 via WR0
0026  B0 05                   mov     al,05h
0028  EE                      out     dx,al

0029  A0 0000 R               mov     al,pattern      ;Output variable to WR5
002C  EE                      out     dx,al

002D  5A                      pop     dx              ;Restore registers
002E  58                      pop     ax

002F  C3                      ret

0030                  cts_lo:                         ;Channel A CTS signal false.
                                                      ;Causing it to assume RS-232 -12V
```

```
0030  50                      push    ax              ;Save registers
0031  52                      push    dx

0032  80 26 0000  FD          and     pattern,0FDh    ;Reset bit D1 in variable

0037  BA 00B1                 mov     dx,a_cmd        ;Point to WR5 via WR0
003A  B0 05                   mov     al,05h
003C  EE                      out     dx,al

003D  A0 0000 R               mov     al,pattern      ;Output variable to WR5
0040  EE                      out     dx,al

0041  5A                      pop     dx              ;Restore registers
0042  58                      pop     ax

0043  C3                      ret

0044                  dsr_hi:                         ;Channel A DSR signal true.
                                                      ;Causing it to assume RS-232 +12V

0044  50                      push    ax              ;Save registers
0045  52                      push    dx

0046  80 0E 0000 R 80         or      pattern,80h     ;Set bit D7 in variable

004B  BA 00B1                 mov     dx,a_cmd        ;Point to WR5 via WR0
004E  B0 05                   mov     al,05h
0050  EE                      out     dx,al

0051  A0 0000 R               mov     al,pattern      ;Output variable to WR5
0054  EE                      out     dx,al

0055  5A                      pop     dx              ;Restore registers
0056  58                      pop     ax

0057  C3                      ret

0058                  dsr_lo:                         ;Channel A DSR signal false.
                                                      ;Causing it to assume RS-232 -12V

0058  50                      push    ax              ;Save registers
0059  52                      push    dx

005A  80 26 0000 R 7F         and     pattern,7Fh     ;Reset bit D7 in variable

005F  BA 00B1                 mov     dx,a_cmd        ;Point to WR5 via WR0
0062  B0 05                   mov     al,05h
0064  EE                      out     dx,al

0065  A0 0000 R               mov     al,pattern      ;Output variable to WR5
0068  EE                      out     dx,al

0069  5A                      pop     dx              ;Restore registers
006A  58                      pop     ax

006B  C3                      ret

006C                  code    ends
                              end
```

Interrupts

The VL-7312 / VL-7314 is capable of generating interrupts in response to a wide variety of events. Both SCC chips can be programmed to provide unique interrupt vectors for each of the possible interrupt sources. The interrupt vectors can be used as "type codes" for 8088 class CPUs, or "mode 2 vectors" for Z80 CPUs.

Interrupt Code Example

The following code example shows how to operate the VL-7312 / VL-7314 using interrupts. This example utilizes vectored interrupts to handle asynchronously received and transmitted characters. Ring buffers are used to temporarily hold characters until they can be processed. The code example includes routines to manage interrupts using VersaLogic's VL-188 CPU card.

The key user subroutines are:

RX_STATUS : Used to determine if new characters need to be read using Read_Char subroutine, results returned in Z flag.

READ_CHAR : Fetches received character to AL register.

TX_STATUS: Used to determine if new characters can be written using Write_Char subroutine, results returned in Z flag.

WRITE_CHAR: Transmits character in AL register.

```
                                                    ;Microsoft Macro Assembler 5.0

                                                    ;VL-7312/14 I/O PORTS
= 00B0                          a_data  equ     00B0h           ;Channel A (SCC#1) Data register
= 00B1                          a_cmd   equ     00B1h           ;Channel A (SCC#1) read/write reg. 0

                                                    ;VL-188 CPU I/O PORTS
= FF22                          eoi     equ     0FF22h          ;80188 EOI register
= FF3A                          int1    equ     0FF3Ah          ;80188 INT1 Control register

0000                            stack   segment page stack      ;Reserve 256 bytes for stack
0000  0100[ ?? ]                        db      100h dup (?)
0100                            stack   ends

0000                            data    segment page

0000  0100[ 00 ]                rx_buf  db      100h dup (00h)  ;Rx buffer = 256 bytes
0100                            rx_end  label   byte            ;End of buffer label
0100  0100[ 00 ]                tx_buf  db      100h dup (00h)  ;Tx buffer = 256 bytes
0200                            tx_end  label   byte            ;End of buffer label

0200  0000                      rx_in   dw      0000h           ;Rx buffer insertion pointer
0202  0000                      rx_out  dw      0000h           ;Rx buffer extraction pointer
0204  0000                      tx_in   dw      0000h           ;Tx buffer insertion pointer
0206  0000                      tx_out  dw      0000h           ;Tx buffer extraction pointer
0208  00                        asleep  db      00h             ;Flag tracks Tx interrupt enable state

0209                            data    ends
```

```
0000                            vector segment page at 0        ;CPU Interrupt Vector Table
0080                                    org     0080h
0080  ????????????????         vec20   dq      ?               ;Channel B Tx buffer empty
0088  ????????????????         vec22   dq      ?               ;Channel B External/Status
0090  ????????????????         vec24   dq      ?               ;Channel B Rx character ready
0098  ????????????????         vec26   dq      ?               ;Channel B Special Rx condition
00A0  ????????????????         vec28   dq      ?               ;Channel A Tx buffer empty
00A8  ????????????????         vec2A   dq      ?               ;Channel A External/Status
00B0  ????????????????         vec2C   dq      ?               ;Channel A Rx character ready
00B8  ????????????????         vec2E   dq      ?               ;Channel A Special Rx condition
00C0                            vector  ends

0000                            code    segment
                                        assume  cs:code,ds:data

0000  E8 0036 R                 start:  call    point           ;Initialize buffer pointers
0003  E8 021A R                         call    install         ;Install CPU interrupt vectors
0006  E8 0059 R                         call    init            ;Reset and initialize channel A
0009  E8 020E R                         call    un_msk          ;Enable STD Bus INTRQ* interrupts
000C  FB                                sti                     ;Enable CPU interrupt flag

000D  E8 001F R                         call    alphabet        ;Fast write alphabet to Tx buffer

0010  E8 0139 R                 hold:   call    rx_status       ;Wait here until character arrives
0013  74 FB                             jz      hold
0015  E8 009A R                         call    read_char       ;Fetch the character
0018  E8 00AB R                         call    write_char      ;Echo character back out
001B  EB F3                             jmp     hold

001D  EB FE                     halt:   jmp     halt

001F                            alphabet:                       ;Fast write alphabet to Tx buffer
001F  B0 41                             mov     al,'A'          ;followed by CR/LF
0021  B9 001A                           mov     cx,26
0024  E8 00AB R                 tloop:  call    write_char
0027  FE C0                             inc     al
0029  E2 F9                             loop    tloop
002B  B0 0D                             mov     al,0Dh
002D  E8 00AB R                         call    write_char
0030  B0 0A                             mov     al,0Ah
0032  E8 00AB R                         call    write_char
0035  C3                                ret

0036                            point:                          ;Initialize pointers, regs & flags
0036  50                                push    ax

0037  B8 ---- R                         mov     ax,seg data     ;Initialize DS and ES registers
003A  8E D8                             mov     ds,ax
003C  8E C0                             mov     es,ax

003E  8D 06 0000 R                      lea     ax,rx_buf       ;Receive buffer:
0042  A3 0200 R                         mov     rx_in,ax        ;Point insertion pointer to head
0045  A3 0202 R                         mov     rx_out,ax       ;Point extraction pointer to head

0048  8D 06 01F0 R                      lea     ax,tx_buf+0f0h  ;Transmit buffer:
004C  A3 0204 R                         mov     tx_in,ax        ;Point insertion pointer to head
004F  A3 0206 R                         mov     tx_out,ax       ;Point extraction pointer to head

0052  C6 06 0208 R 01                   mov     asleep,1        ;Initialize Tx interrupt flag

0057  58                                pop     ax
0058  C3                                ret
```

```
0059                            init:                           ;Channel A initialization
0059  EB 24 90                          jmp     transfer        ;Jump past table

005C  22                        table   db      offset eot - offset table-1
005D  04 47                             db      04,047h         ;WR4  x16, 1 Stop bit, Even Parity
005F  02 20                             db      02,020h         ;WR2  Base interrupt vector = 20h
0061  03 80                             db      03,080h         ;WR3  No auto enable, Rx = 7 bits
0063  05 A2                             db      05,0A2h         ;WR5  Tx = 7 bits, DTR & RTS = On
0065  09 01                             db      09,001h         ;WR9  VIS=1, Status Low, Lower chain
0067  0B 56                             db      11,056h         ;WR11 Clock sources
0069  0C 0E                             db      12,00Eh         ;WR12 Baud rate (low)  9600 baud
006B  0D 00                             db      13,000h         ;WR13 Baud rate (high) 9600 baud
006D  0E 02                             db      14,002h         ;WR14 Use PCLK for baud rate timebase
006F  0E 03                             db      14,003h         ;WR14 Enable baud rate generator
0071  03 81                             db      03,081h         ;WR3  Enable Rx
0073  05 AA                             db      05,0AAh         ;WR5  Enable Tx
0075  0F 00                             db      15,000h         ;WR15 All ext/status interrupts = off
0077  00 10                             db      00,010h         ;WR0  Reset external/status twice
0079  00 10                             db      00,010h         ;WR0  Reset external/status twice
007B  01 12                             db      01,012h         ;WR1  Enable Tx & Rx interrupts
007D  09 09                             db      09,009h         ;WR9  Master interrupt enable
007F                            eot     label   byte            ;End of table

007F                            transfer:                       ;Issue channel reset command
007F  BA 00B1                           mov     dx,a_cmd        ;Point to WR9 via WR0
0082  B0 09                             mov     al,09h
0084  EE                                out     dx,al

0085  B0 80                             mov     al,80h          ;Channel A reset command
0087  EE                                out     dx,al

0088  BE 005C R                         mov     si,offset table ;Fetch table length into CX
008B  2E: AC                            lods    table
008D  8A C8                             mov     cl,al
008F  B5 00                             mov     ch,00h

0091  BA 00B1                           mov     dx,a_cmd

0094  2E: AC                    repeat: lods    table           ;Fetch table entry
0096  EE                                out     dx,al
0097  E2 FB                             loop    repeat

0099  C3                                ret

009A                            read_char:                      ;Exit with received character
                                                                ;in AL register.  Returns old
                                                                ;data if new data has not been
                                                                ;received.
009A  56                                push    si

009B  E8 0139 R                         call    rx_status       ;See if there's new data
009E  74 03                             jz      old_data        ;If not... return old data
00A0  E8 0195 R                         call    bump_rx_out     ;Else... increment extraction pointer
00A3                            old_data:
00A3  8B 36 0202 R                      mov     si,rx_out       ;Point to new character
00A7  8A 04                             mov     al,[si]         ;Fetch character into AL register

00A9  5E                                pop     si
00AA  C3                                ret                     ;Return to caller
```

```
00AB                            write_char:                     ;Character in AL register is
                                                                ;written to Tx buffer.

                                                                ;If buffer is full, character
                                                                ;is lost and carry flag is set.

00AB  52                                push    dx              ;Save registers
00AC  56                                push    si

00AD  E8 0167 R                         call    tx_status       ;Is Tx buffer full?
00B0  72 13                             jc      wc_exit         ;If yes... Loose character & exit

00B2  E8 01DD R                         call    bump_tx_in      ;Store character in next available
00B5  8B 36 0204 R                      mov     si,tx_in        ;location of Tx buffer
00B9  88 04                             mov     [si],al

00BB  80 3E 0208 R 01                   cmp     asleep,1        ;Were Tx interrupts disabled?
00C0  75 03                             jne     wc_exit

00C2  CD 28                             int     28h             ;If yes... Send character directly
                                                                ;to UART causing Tx interrupts to
                                                                ;reenable.

00C4  F8                                clc                     ;Clear carry flag

00C5                            wc_exit:
00C5  5E                                pop     si              ;Restore
00C6  5A                                pop     dx

00C7  C3                                ret

00C8                            rx_isr:                         ;Rx Interrupt Service Routine

                                                                ;As each character arrives it
                                                                ;is placed into Rx buffer indexed
                                                                ;with the insertion pointer.

                                                                ;The incoming character is lost
                                                                ;if the buffer is full.

00C8  50                                push    ax              ;Save registers
00C9  52                                push    dx
00CA  56                                push    si
00CB  1E                                push    ds

00CC  B8 ---- R                         mov     ax,seg data
00CF  8E D8                             mov     ds,ax

00D1  BA 00B0                           mov     dx,a_data       ;Point to RR0
00D4  EC                                in      al,dx           ;Read character from hardware

00D5  24 7F                             and     al,7Fh          ;Mask high order bit
                                                                ;when receiving 7-bit data

00D7  E8 0139 R                         call    rx_status       ;Is buffer full?
00DA  72 09                             jc      rx_isr_exit     ;If yes... Loose character & exit

00DC  E8 01AD R                         call    bump_rx_in      ;Else... Point to next buffer
00DF  8B 36 0200 R                      mov     si,rx_in        ;location

00E3  88 04                             mov     [si],al         ;Store character in buffer

00E5                            rx_isr_exit:

00E5  BA FF22                           mov     dx,eoi          ;Issue a Non-Specific End-Of-Interrupt
00E8  B8 8000                           mov     ax,8000h        ;command to 80188 interrupt controller
00EB  EF                                out     dx,ax

00EC  BA 00B1                           mov     dx,a_cmd        ;Issue Reset Highest IUS command
```

```
00EF  B0 38                           mov     al,38h          ;via WR0
00F1  EE                              out     dx,al

00F2  1F                              pop     ds              ;Restore registers
00F3  5E                              pop     si
00F4  5A                              pop     dx
00F5  58                              pop     ax

00F6  FB                              sti                     ;Reenable CPU interrupts
00F7  CF                              iret                    ;and return to caller

00F8                          tx_isr:                         ;Tx Interrupt Service Routine

                                                              ;When UART wants another character,
                                                              ;one is fetched from Tx buffer
                                                              ;using extraction pointer.

                                                              ;Tx interrupts are disabled if
                                                              ;the UART asks for one when the
                                                              ;buffer is empty.

00F8  50                              push    ax              ;Save registers
00F9  52                              push    dx
00FA  56                              push    si
00FB  1E                              push    ds

00FC  B8 ---- R                       mov     ax,seg data
00FF  8E D8                           mov     ds,ax

0101  E8 0167 R                       call    tx_status       ;Is buffer empty?
0104  75 0E                           jnz     not_empty       ;If not... Process character

0106  BA 00B1                         mov     dx,a_cmd        ;Else... Disable Tx interrupts
0109  B0 28                           mov     al,28h          ;Using Reset TxINT Pending
010B  EE                              out     dx,al           ;command via WR0

010C  C6 06 0208 R 01                 mov     asleep,1        ;Flag indicating that Tx
                                                              ;interrupts have been shut down
0111  EB 13 90                        jmp     tx_isr_exit

0114                          not_empty:
0114  C6 06 0208 R 00                 mov     asleep,0        ;Flag indicating that Tx
                                                              ;interrupts are enabled

0119  E8 01C5 R                       call    bump_tx_out     ;Increment extraction pointer
011C  8B 36 0206 R                    mov     si,tx_out       ;Fetch character into AL register
0120  8A 04                           mov     al,[si]

0122  BA 00B0                         mov     dx,a_data       ;Transmit character
0125  EE                              out     dx,al

0126                          tx_isr_exit:
0126  BA FF22                         mov     dx,eoi          ;Issue a Non-Specific End-Of-Interrupt
0129  B8 8000                         mov     ax,8000h        ;command to 80188 interrupt controller
012C  EF                              out     dx,ax

012D  BA 00B1                         mov     dx,a_cmd        ;Issue Reset Highest IUS command
0130  B0 38                           mov     al,38h          ;via WR0
0132  EE                              out     dx,al

0133  1F                              pop     ds              ;Restore registers
0134  5E                              pop     si
0135  5A                              pop     dx
0136  58                              pop     ax

0137  FB                              sti                     ;Reenable CPU interrupts
0138  CF                              iret                    ;and return to caller
```

```
0139                            rx_status:                      ;Evaluate condition of Rx buffer.
                                                                ;Zero and Carry flags altered to
                                                                ;reflect current status:
                                                                ;Z=0  Rx buffer not empty   (jnz)
                                                                ;Z=1  Rx buffer empty       (jz)
                                                                ;C=0  Rx buffer not full    (jnc)
                                                                ;C=1  Rx buffer full        (jc)

0139  50                                  push    ax            ;Save registers
013A  52                                  push    dx

013B  8B 16 0200 R                        mov     dx,rx_in      ;Test for empty condition
013F  3B 16 0202 R                        cmp     dx,rx_out     ;By comparing in with out

0143  9F                                  lahf                  ;Copy flags to accumulator
0144  8A C4                               mov     al,ah
0146  24 40                               and     al,40h        ;Mask everything but Z to zero

                                                                ;Test for full condition
                                                                ;by comparing in+1 with out

0148  81 FA 00FF R                        cmp     dx,offset rx_end-1 ;Pointing to last buffer cell?
014C  74 04                               jz      rs_head       ;If yes... Repoint to head
014E  42                                  inc     dx            ;Else increment copy of pointer
014F  EB 04 90                            jmp     rs_full
0152                            rs_head:
0152  BA 0000 R                           mov     dx,offset rx_buf
0155                            rs_full:
0155  3B 16 0202 R                        cmp     dx,rx_out     ;Compare in+1 with out pointer
0159  9F                                  lahf                  ;Fetch flags into AH register
015A  D0 C4                               rol     ah,1          ;Rotate Z into C position
015C  D0 C4                               rol     ah,1
015E  80 E4 01                            and     ah,1          ;Mask everything but C to zero
0161  0A E0                               or      ah,al         ;Merge Z and C flags
0163  9E                                  sahf                  ;Store back into flag register

0164  5A                                  pop     dx            ;Restore registers
0165  58                                  pop     ax
0166  C3                                  ret

0167                            tx_status:                      ;Evaluate condition of Tx buffer.
                                                                ;Zero and Carry flags altered to
                                                                ;reflect current status:
                                                                ;Z=0  Tx buffer not empty    (jnz)
                                                                ;Z=1  Tx buffer empty        (jz)
                                                                ;C=0  Tx buffer not full     (jnc)
                                                                ;C=1  Tx buffer full         (jc)

0167  50                                  push    ax            ;Save registers
0168  52                                  push    dx

0169  8B 16 0204 R                        mov     dx,tx_in      ;Test for empty condition
016D  3B 16 0206 R                        cmp     dx,tx_out     ;By comparing in with out

0171  9F                                  lahf                  ;Save Z flag state in accumulator
0172  8A C4                               mov     al,ah
0174  24 40                               and     al,40h        ;Mask everything but Z to zero

                                                                ;Test for full condition
                                                                ;by comparing in+1 with out

0176  81 FA 01FF R                        cmp     dx,offset tx_end-1 ;Pointing to last buffer cell?
017A  74 04                               jz      ts_head       ;If yes... Repoint to head
017C  42                                  inc     dx            ;else increment copy of pointer
017D  EB 04 90                            jmp     ts_full
0180                            ts_head:
0180  BA 0100 R                           mov     dx,offset tx_buf
0183                            ts_full:
0183  3B 16 0206 R                        cmp     dx,tx_out     ;Compare in+1 with out pointer
```

```
0187  9F                              lahf                    ;Fetch flags into AH register
0188  D0 C4                          rol     ah,1            ;Rotate Z into C position
018A  D0 C4                          rol     ah,1
018C  80 E4 01                       and     ah,1            ;Mask everything but C to zero
018F  0A E0                          or      ah,al           ;Merge Z and C flags
0191  9E                             sahf                    ;Store back into flag register

0192  5A                             pop     dx              ;Restore registers
0193  58                             pop     ax
0194  C3                             ret

0195                            bump_rx_out:                 ;Increment Rx buffer extraction
                                                             ;pointer.  Roll to beginning
                                                             ;if it runs off the end
0195  50                             push    ax

0196  A1 0202 R                       mov     ax,rx_out       ;Pointing to last buffer cell?
0199  3D 00FF R                       cmp     ax,offset rx_end-1
019C  74 07                          jz      ro_head         ;If yes... Repoint to head
019E  FF 06 0202 R                   inc     rx_out          ;Else... Increment pointer
01A2  EB 07 90                       jmp     ro_exit
01A5                            ro_head:
01A5  C7 06 0202 R 0000 R            mov     rx_out,offset rx_buf
01AB                            ro_exit:
01AB  58                             pop     ax
01AC  C3                             ret

01AD                            bump_rx_in:                  ;Increment Rx buffer insertion
                                                             ;pointer.  Roll to beginning
                                                             ;if it runs off the end
01AD  50                             push    ax

01AE  A1 0200 R                       mov     ax,rx_in        ;Pointing to last buffer cell?
01B1  3D 00FF R                       cmp     ax,offset rx_end-1
01B4  74 07                          jz      ri_head         ;If yes... Repoint to head
01B6  FF 06 0200 R                   inc     rx_in           ;Else... Increment pointer
01BA  EB 07 90                       jmp     ri_exit
01BD                            ri_head:
01BD  C7 06 0200 R 0000 R            mov     rx_in,offset rx_buf
01C3                            ri_exit:
01C3  58                             pop     ax
01C4  C3                             ret

01C5                            bump_tx_out:                 ;Increment Tx buffer extraction
                                                             ;pointer.  Roll to beginning
                                                             ;if it runs off the end
01C5  50                             push    ax

01C6  A1 0206 R                       mov     ax,tx_out       ;Pointing to last buffer cell?
01C9  3D 01FF R                       cmp     ax,offset tx_end-1
01CC  74 07                          jz      to_head         ;If yes... Repoint to head
01CE  FF 06 0206 R                   inc     tx_out          ;Else... Increment pointer
01D2  EB 07 90                       jmp     to_exit
01D5                            to_head:
01D5  C7 06 0206 R 0100 R            mov     tx_out,offset tx_buf
01DB                            to_exit:
01DB  58                             pop     ax
01DC  C3                             ret

01DD                            bump_tx_in:                  ;Increment Tx buffer insertion
                                                             ;pointer.  Roll to beginning
                                                             ;if it runs off the end
01DD  50                             push    ax

01DE  A1 0204 R                       mov     ax,tx_in        ;Pointing to last buffer cell?
01E1  3D 01FF R                       cmp     ax,offset tx_end-1
01E4  74 07                          jz      ti_head         ;If yes... Repoint to head
01E6  FF 06 0204 R                   inc     tx_in           ;Else... Increment pointer
```

```
01EA  EB 07 90                  jmp     ti_exit
01ED                            ti_head:
01ED  C7 06 0204 R 0100 R       mov     tx_in,offset tx_buf
01F3                            ti_exit:
01F3  58                        pop     ax
01F4  C3                        ret

01F5                            error_isr:
01F5  50                        push    ax              ;This interrupt service routine does
01F6  52                        push    dx              ;nothing.

01F7  BA 00B1                   mov     dx,a_cmd        ;Issue Error Reset command via WR0
01FA  B0 30                     mov     al,30h
01FC  EE                        out     dx,al

01FD  BA FF22                   mov     dx,eoi          ;Issue a Non-Specific End-Of-Interrupt
0200  B8 8000                   mov     ax,8000h        ;command to 80188 interrupt controller
0203  EF                        out     dx,ax

0204  BA 00B1                   mov     dx,a_cmd        ;Issue Reset Highest IUS command
0207  B0 38                     mov     al,38h          ;via WR0
0209  EE                        out     dx,al

020A  5A                        pop     dx
020B  58                        pop     ax
020C  FB                        sti
020D  CF                        iret

020E                            un_msk:
020E  50                        push    ax
020F  52                        push    dx
0210  BA FF3A                   mov     dx,int1         ;Unmask STD Bus INTRQ* interrupts
0213  B8 0037                   mov     ax,0037h
0216  EF                        out     dx,ax
0217  5A                        pop     dx
0218  58                        pop     ax
0219  C3                        ret

021A                            install:                ;Install service routine addresses
021A  50                        push    ax              ;into CPU Interrupt Vector Table
021B  1E                        push    ds
021C  B8 ---- R                 mov     ax,seg vector
021F  8E D8                     mov     ds,ax
                                assume  ds:vector
0221  C7 06 0080 R 01F5 R       mov     word ptr vec20[0],offset error_isr
0227  C7 06 0082 R ---- R       mov     word ptr vec20[2],seg    error_isr
022D  C7 06 0088 R 01F5 R       mov     word ptr vec22[0],offset error_isr
0233  C7 06 008A R ---- R       mov     word ptr vec22[2],seg    error_isr
0239  C7 06 0090 R 01F5 R       mov     word ptr vec24[0],offset error_isr
023F  C7 06 0092 R ---- R       mov     word ptr vec24[2],seg    error_isr
0245  C7 06 0098 R 01F5 R       mov     word ptr vec26[0],offset error_isr
024B  C7 06 009A R ---- R       mov     word ptr vec26[2],seg    error_isr
0251  C7 06 00A0 R 00F8 R       mov     word ptr vec28[0],offset tx_isr
0257  C7 06 00A2 R ---- R       mov     word ptr vec28[2],seg    tx_isr
025D  C7 06 00A8 R 01F5 R       mov     word ptr vec2A[0],offset error_isr
0263  C7 06 00AA R ---- R       mov     word ptr vec2A[2],seg    error_isr
0269  C7 06 00B0 R 00C8 R       mov     word ptr vec2C[0],offset rx_isr
026F  C7 06 00B2 R ---- R       mov     word ptr vec2C[2],seg    rx_isr
0275  C7 06 00B8 R 01F5 R       mov     word ptr vec2E[0],offset error_isr
027B  C7 06 00BA R ---- R       mov     word ptr vec2E[2],seg    error_isr
0281  1F                        pop     ds
0282  58                        pop     ax
0283  C3                        ret

0284                            code    ends
                                end     start
```

Reference

Specifications

Size: Meets all STD Bus mechanical specifications

Storage Temperature: −40˚ to +85˚ C

Free Air Operating Temperature:

VL-7312:	0˚ to +65˚ C
VL-73CT12:	−40˚ to +85˚ C
VL-7314:	0˚ to +65˚ C
VL-73CT14:	−40˚ to +85˚ C

Power Requirements:

VL-7312:	5V	±5% at 312 ma typ.
	±12V	±10% at 25 ma typ.
VL-73CT12:	5V	±5% at 85 ma typ.
	±12V	±10% at 25 ma typ.
VL-7314:	5V	±5% at 460 ma typ.
	±12V	±10% at 50 ma typ.
VL-73CT14:	5V	±5% at 94 ma typ.
	±12V	±10% at 50 ma typ.

VL-7312 / VL-7314 Jumper Block Locations

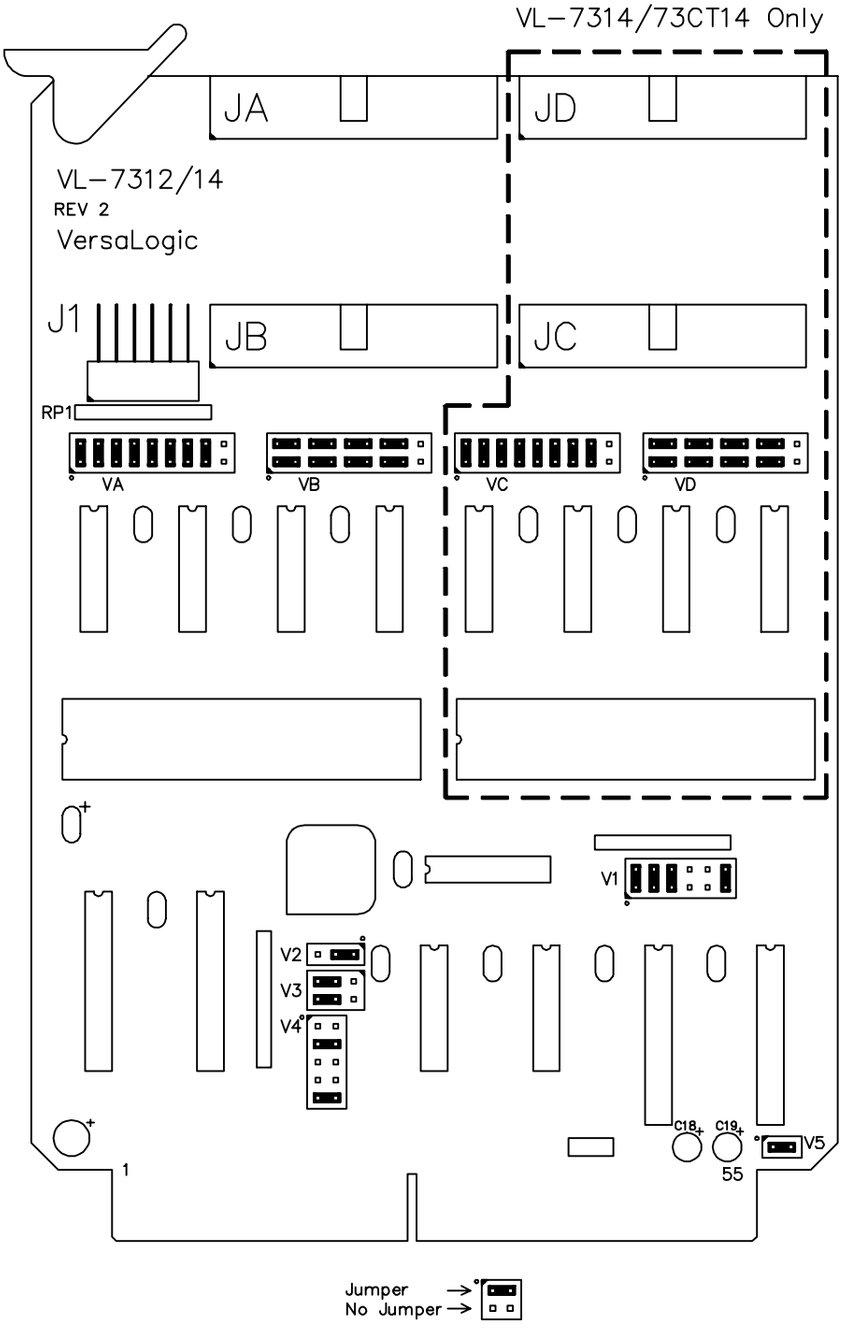


Figure 6-1. Jumper Block Locations for VL-7312/14

Jumper Options

Jumper Block	Description	As Shipped	Page
VA	Channel A DCE / DTE configuration	DCE	2-10
VB	Channel B DCE / DTE configuration	DTE	2-10
VC	Channel C DCE / DTE configuration	DCE	2-10
VD	Channel D DCE / DTE configuration	DTE	2-10
$V1_a$	Interrupt vector enable In – Board responds to interrupt acknowledge cycle Out – Board ignores interrupt acknowledge cycle	In	2-7
$V1_b$	Channel C & D interrupt request interconnect In – Channel C & D interrupts merged with channels A & B on INTRQ* Out – Channel C & D interrupts only on INT2* (J1 pin 11)	In	2-7
$V1_c$	Interrupt request enable In – Enables interrupt requests via INTRQ* Out – Disables interrupt requests via INTRQ*	In	2-7
$V1_d$	CPU select In – STD-Z80 interrupt acknowledge cycle Out – STD-8088 interrupt acknowledge cycle	Out	2-9
$V1_e$	Advanced write In – Disable advanced write cycle for nonstandard STD 8088 timing Out – Enable advanced write cycle for STD 8088 timing	Out	2-9
$V1_f$	AUX GND connected to digital ground In – Connect AUX GND to digital ground Out – Separate AUX GND from digital ground	In	2-9
V2	IOEXP select a – Board responds to IOEXP high and low b – Board responds to IOEXP low None – Board responds to IOEXP high	a In, b Out } IOEXP high and low	2-6
$V3_{a-b}$	Address mode selector (8- or 10-bit decoding) a – A9 b – A8	a A9 Low, b A8 Low } 10-Bit address decoding	2-4
$V4_{a-e}$	Board address (A3 – A7) a – A7 b – A6 c – A5 d – A4 e – A3	a Out, b In, c Out, d Out, e In } B0 Hex	2-4
V5	PCO/+3.3V Interconnect In – STD-80 Mode. PCO (P51) connected to on-board circuitry. Out – STD-32 Mode. PCO (P51) isolated.	In	2-11

Figure 6-2. Jumper Functions

I/O Port Mapping

All communications with the VL-7312 / VL-7314 take place through the following I/O ports:

Output Port	Input Port	Port Address	As Shipped Address
Channel A XMIT Data	Channel A RCV Data	Board Address + 0	B0
Channel A Write Register 0	Channel A Read Register 0	Board Address + 1	B1
Channel B XMIT Data	Channel B RCV Data	Board Address + 2	B2
Channel B Write Register 0	Channel B Read Register	Board Address + 3	B3
Channel C XMIT Data	Channel C RCV Data	Board Address + 4	B4
Channel C Write Register 0	Channel C Read Register 0	Board Address + 5	B5
Channel D XMIT Data	Channel D RCV Data	Board Address + 6	B6
Channel D Write Register 0	Channel D Read Register 0	Board Address + 7	B7

Figure 6-3. I/O Port Addresses

SCC Registers

The following table lists the functions assigned to each read and write register. Each SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Write Register	Functions	Page
WR0 [1]	CRC initialize, initialization commands for the various modes, Register Pointer	4-4
WR1 [2]	Transmit/Receive interrupt and data transfer mode definition	4-6
WR2 [2]	Interrupt vector (accessed through either channel)	4-8
WR3 [2]	Receive parameters and control	4-8
WR4 [2]	Transmit/Receive miscellaneous parameters and modes	4-9
WR5 [2]	Transmit parameters and controls	4-11
WR6 [2]	Sync characters or SDLC address field	4-12
WR7 [2]	Sync character or SDLC flag	4-13
WR8 [1,2]	Transmit buffer	4-13
WR9 [2]	Master interrupt control and reset (accessed through either channel)	4-13
WR10 [2]	Miscellaneous transmitter/receiver control bits	4-15
WR11 [2]	Clock mode control	4-16
WR12 [2]	Lower byte of baud rate generator time constant	4-18
WR13 [2]	Upper byte of baud rate generator time constant	4-18
WR14 [2]	Miscellaneous control bits	4-19
WR15 [2]	External/Status interrupt control	4-21
Read Register	**Functions**	
RR0 [1]	Transmit/Receive buffer status and External status	4-22
RR1 [2]	Special Receive Condition status	4-23
RR2 [2,3]	Interrupt vector written into WR2	4-24
RR2 [2,4]	Modified interrupt vector	4-25
RR3 [2]	Interrupt pending bits	4-25
RR8 [1,2]	Receive buffer	4-25
RR10 [2]	Miscellaneous status	4-26
RR12 [2]	Lower byte of baud rate generator time constant	4-26
RR13 [2]	Upper byte of baud rate generator time constant	4-27
RR15 [2]	External/Status interrupt information	4-27

[1] Register directly accessible as I/O port.
[2] Register accessible by writing pointer to WR0 first.
[3] When read from channel A or channel C.
[4] When read from channel B or channel D.

Figure 6-4. Read and Write Register Functions

SCC Write Registers

Write Register 0

D7	D6	
0	0	Null Code
0	1	Reset Receive CRC Checker
1	0	Reset Transmit CRC Generator
1	1	Reset Transmit Underrun / EOM Latch

D5	D4	D3	
0	0	0	Null code
0	0	1	Point high
0	1	0	Reset external / status interrupts
0	1	1	Send abort (SDLC)
1	0	0	Enable int on next Rx character
1	0	1	Reset TxINT pending
1	1	0	Error reset
1	1	1	Reset highest IUS

D2	D1	D0	
0	0	0	Register 0
0	0	1	Register 1
0	1	0	Register 2
0	1	1	Register 3
1	0	0	Register 4
1	0	1	Register 5
1	1	0	Register 6
1	1	1	Register 7
0	0	0[1]	Register 8
0	0	1[1]	Register 9
0	1	0[1]	Register 10
0	1	1[1]	Register 11
1	0	0[1]	Register 12
1	0	1[1]	Register 13
1	1	0[1]	Register 14
1	1	1[1]	Register 15

[1] Use Point High command

Write Register 1

D7 – $\overline{\text{WAIT}}$ / DMA Request Enable
D6 – $\overline{\text{WAIT}}$ / DMA Request Function
D5 – WAIT / DMA Request On Receive / $\overline{\text{Transmit}}$

D4	D3	
0	0	Receive interrupt disable
0	1	Receive Interrupt on First Character or Special Condition
1	0	Interrupt on All Receive Characters or Special Condition
1	1	Receive Interrupt on Special Condition Only

D2 – Parity is Special Condition
D1 – Transmitter Interrupt Enable
D0 – External / Status Master Interrupt Enable

Write Register 2

D7	D6	D5	D4	D3	D2	D1	D0	
								Interrupt Vector

Write Register 3

D7 D6
0 0 Rx 5 Bits / Character
0 1 Rx 7 Bits / Character
1 0 Rx 7 Bits / Character
1 1 Rx 8 Bits / Character

D5 – Auto Enables
D4 – Enter Hunt Mode
D3 – Receiver CRC Enable
D2 – Address Search Mode (SDLC)
D1 – SYNC Character Load Inhibit
D0 – Receiver Enable

Write Register 4

D7 D6
0 0 1x Clock Mode
0 1 16x Clock Mode
1 0 32x Clock Mode
1 1 64x Clock Mode

D5 D4
0 0 Monosync (8-bit sync character)
0 1 Bisync (16-bit sync character)
1 0 SDLC
1 1 External Sync

D3 D2
0 0 Synchronous mode enable
0 1 Asynchronous mode, 1 stop bit
1 0 Asynchronous mode, 1½ stop bits
1 1 Asynchronous mode, 2 stop bits

D1 – Parity Even / $\overline{\text{Odd}}$
D0 – Parity Enable

Write Register 5

D7 – DTR

D6 D5
0 0 Tx 5 or less Bits / Character
0 1 Tx 7 Bits / Character
1 0 Tx 6 Bits / Character
1 1 Tx 8 Bits / Character

D4 – Send Break
D3 – Transmit Enable
D2 – $\overline{\text{SDLC}}$ / CRC-16
D1 – RTS
D0 – Transmit CRC Enable

Write Register 6

D7	D6	D5	D4	D3	D2	D1	D0	Mode
SYNC7	SYNC6	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Monosync, 8 Bits
SYNC1	SYNC0	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Monosync, 6 Bits
SYNC7	SYNC6	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Bisync, 16 Bits
SYNC3	SYNC2	SYNC1	SYNC0	1	1	1	1	Bisync, 12 Bits
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	SDLC
ADR7	ADR6	ADR5	ADR4	ADR3	x	x	x	SDLC (Address Range)

Write Register 7

D7	D6	D5	D4	D3	D2	D1	D0	Mode
SYNC7	SYNC6	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	Monosync, 8 bits
SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	x	x	Monosync, 6 bits
SYNC15	SYNC14	SYNC13	SYNC12	SYNC11	SYNC10	SYNC9	SYNC8	Bisync, 16 bits
SYNC11	SYNC10	SYNC9	SYNC8	SYNC7	SYNC6	SYNC5	SYNC4	Bisync, 12 bits
0	1	1	1	1	1	1	0	SDLC

Write Register 8

D7 D6 D5 D4 D3 D2 D1 D0 Transmit Buffer

Write Register 9

D7	D6	
0	0	No reset
0	1	Channel B Reset
1	0	Channel A Reset
1	1	Hardware Reset

D5 – Not Used
D4 – Status High / $\overline{\text{Status Low}}$
D3 – Master Interrupt Enable
D2 – Disable Lower Chain
D1 – No Vector
D0 – Vector Includes Status

Write Register 10

D7 – CRC Presets 1 / $\overline{0}$

D6	D5	
0	0	NRZ (Used for Asynchronous mode)
0	1	NRZI
1	0	FM1
1	1	FM0

D4 – Go Active On Poll
D3 – Mark / $\overline{\text{Flag}}$ Idle
D2 – Abort / $\overline{\text{Flag}}$ On Underrun
D1 – Loop Mode
D0 – 6-Bit / 8-Bit Sync

Write Register 11

D7 – RTxC XTAL / $\overline{\text{NO XTAL}}$ (**NOTE:** Must be "1" for proper operation of VL-7312 / VL-7314 card)

D6	**D5**	
0	0	TCLK (RCLK when jumpered for DTE operation). Use for Synchronous modes
0	1	Not used
1	0	Baud rate generator. Use for Asynchronous modes
1	1	Digital phase-locked loop (DPLL)

D4	**D3**	
0	0	TCLK (RCLK when jumpered for DTE operation). Use for Synchronous modes
0	1	Not used
1	0	Baud rate generator. Use for Asynchronous modes
1	1	Digital phase-locked loop (DPLL)

D2 – TRxC Output / $\overline{\text{Input}}$ (**NOTE:** Must be "0" for proper operation of VL-7312 / VL-7314 card)

D1	**D0**	
0	0	Not used on VL-7312 / VL-7314
0	1	Transmit clock
1	0	Baud rate generator
1	1	Digital phase-locked loop (DPLL)

Write Register 12

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0	Lower byte of baud rate time constant

Write Register 13

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
TC_{15}	TC_{14}	TC_{13}	TC_{12}	TC_{11}	TC_{10}	TC_9	TC_8	Upper byte of baud rate time constant

Write Register 14

D7	**D6**	**D5**	
0	0	0	Null command
0	0	1	Enter search mode
0	1	0	Reset clock missing
0	1	1	Disable DPLL
1	0	0	Set source = BR generator
1	0	1	Set source = RTxC
1	1	0	Set FM mode
1	1	1	Set NRZI mode

D4 – Local Loopback
D3 – $\overline{\text{Auto Echo}}$
D2 – $\overline{\text{DTR Enable}}$ / Request Function
D1 – Baud Rate Generator Source
D0 – Baud Rate Generator Enable

Write Register 15

D7 – Break / Abort Interrupt Enable
D6 – Tx Underrun / EOM Interrupt Enable
D5 – CTS Interrupt Enable
D4 – External / Hunt Interrupt Enable
D3 – DCD Interrupt Enable
D2 – Not used
D1 – Zero Count Interrupt Enable
D0 – Not used

SCC Read Registers

Read Register 0

D7 – Break / Abort
D6 – Transmit Underrun / EOM
D5 – CTS
D4 – Sync / Hunt
D3 – DCD
D2 – Transmit Buffer Empty
D1 – Zero Count
D0 – Receive Character Available

Read Register 1

D7 – End of Frame (SDLC)
D6 – CRC / Framing Error
D5 – Receiver Overrun Error
D4 – Parity Error
D3 – Residue Code 2
D2 – Residue Code 1
D1 – Residue Code 0
D0 – All Sent

Read Register 2 (Read from channels A and C only)

D7 D6 D5 D4 D3 D2 D1 D0 Interrupt vector as written into WR2

Read Register 2 (Read from channels B and D only)

D7 D6 D5 D4 D3 D2 D1 D0 Modified interrupt vector (sent to CPU)

Read Register 3 (Read from channels A and C only)

D7 – 0
D6 – 0
D5 – Interrupt pending from channel A / C Receive
D4 – Interrupt pending from channel A / C Transmit
D3 – Interrupt pending from channel A / C External / Status
D2 – Interrupt pending from channel B / D Receive
D1 – Interrupt pending from channel B / D Transmit
D0 – Interrupt pending from channel B / D External / Status

Read Register 8

D7 D6 D5 D4 D3 D2 D1 D0 Receive Data Register

Read Register 10

D7 – One Clock Missing
D6 – Two Clocks Missing
D5 – Not used
D4 – Loop Sending
D3 – Not used
D2 – Not used
D1 – On Loop
D0 – Not used

Read Register 12

D7	D6	D5	D4	D3	D2	D1	D0	
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0	Lower byte of baud rate time constant

Read Register 13

D7	D6	D5	D4	D3	D2	D1	D0	
TC_{15}	TC_{14}	TC_{13}	TC_{12}	TC_{11}	TC_{10}	TC_9	TC_8	Upper byte of baud rate time constant

Read Register 15

D7 – Break / Abort interrupt enable
D6 – Tx Underrun / EOM interrupt enable
D5 – CTS interrupt enable
D4 – SYNC / Hunt interrupt enable
D3 – DCD interrupt enable
D2 – 0
D1 – Zero count interrupt enable
D0 – 0

Write Registers Affecting Interrupts

Register	D7	D6	D5	D4	D3	D2	D1	D0	Comments
WR0	0	0	0	1	0	0	0	0	Reset external/status interrupts
WR0	0	0	1	0	0	0	0	0	Enable interrupt on next Rx character
WR0	0	0	1	1	1	0	0	0	Reset highest IUS bit
WR1	X	X	X	X	X	X	X	1	External/Status interrupt enable
WR1	X	X	X	X	X	X	1	X	Transmitter interrupt enable
WR1	X	X	X	X	X	1	X	X	Parity is special condition
WR1	X	X	X	0	0	X	X	X	Rx interrupt disable
WR1	X	X	X	0	1	X	X	X	Rx interrupt on 1st character or special condition
WR1	X	X	X	1	0	X	X	X	Rx interrupt on all characters or special condition
WR1	X	X	X	1	1	X	X	X	Rx interrupt on special condition only
WR2	X	X	X	X	X	X	X	X	Base interrupt vector (or restart instruction)
WR9	X	X	X	X	X	X	0	1	Enable interrupt vector modification
WR9	X	X	X	X	X	X	1	X	Suppresses vector in response to interrupt acknowledge
WR9	X	X	X	X	X	1	0	X	Disable lower interrupt daisy chain
WR9	X	X	X	X	1	X	0	X	Master interrupt enable
WR9	X	X	X	1	X	X	0	X	Status high/status low
WR15	X	X	X	X	X	X	1	X	Zero count interrupt enable
WR15	X	X	X	X	1	X	X	X	DCD interrupt enable
WR15	X	X	X	1	X	X	X	X	External/hunt interrupt enable
WR15	X	X	1	X	X	X	X	X	CTS interrupt enable
WR15	X	1	X	X	X	X	X	X	Tx underrun/EOM interrupt enable
WR15	1	X	X	X	X	X	X	X	Break/abort interrupt enable

Figure 6-5. Write Registers Affecting Interrupts

Read Registers Affected By Interrupts

Register	D7	D6	D5	D4	D3	D2	D1	D0	Comments
RR0	X	X	X	X	X	X	1	X	Zero count status
RR0	X	X	X	X	X	1	X	X	Transmit buffer empty status
RR0	X	X	X	X	1	X	X	X	Data carrier detect status
RR0	X	X	X	1	X	X	X	X	Sync/hunt status
RR0	X	X	1	X	X	X	X	X	Clear to send status
RR0	X	1	X	X	X	X	X	X	Transmit underrun/EOM
RR0	1	X	X	X	X	X	X	X	Break/abort status
RR1	X	X	1	X	X	X	X	X	Receiver overrun status
RR2	X	X	X	X	X	X	X	X	(Ch. A/C) Same interrupt vector in WR2
RR2	X	X	X	X	X	X	X	X	(Ch. B/D) Modified interrupt vector
RR3	0	0	X	X	X	X	X	1	Interrupt pending from Ch. B/D external/status
RR3	0	0	X	X	X	X	1	X	Interrupt pending from Ch. B/D transmit
RR3	0	0	X	X	X	1	X	X	Interrupt pending from Ch. B/D receive
RR3	0	0	X	X	1	X	X	X	Interrupt pending from Ch. A/C external/status
RR3	0	0	X	1	X	X	X	X	Interrupt pending from Ch. A/C transmit
RR3	0	0	1	X	X	X	X	X	Interrupt pending from Ch. A/C receive

Figure 6-6. Read Registers Affected By Interrupts

Interrupt Vectors

The table below shows all possible interrupt vectors and the conditions which cause them. The resulting vector depends upon the state of the Vector Includes Status and the Status High/Status Low bit within WR9 (see page 4-13).

D7	D6	D5	D4	D3	D2	D1	D0	Comments
Vector Includes Status = 1 Status High/Status Low = 0								
V_7	V_6	V_5	V_4	0	0	0	V_0	Channel B/D Transmit Buffer Empty
V_7	V_6	V_5	V_4	0	0	1	V_0	Channel B/D External/Status Change
V_7	V_6	V_5	V_4	0	1	0	V_0	Channel B/D Receive Character Available
V_7	V_6	V_5	V_4	0	1	1	V_0	Channel B/D Special Receive Condition
V_7	V_6	V_5	V_4	1	0	0	V_0	Channel A/C Transmit Buffer Empty
V_7	V_6	V_5	V_4	1	0	1	V_0	Channel A/C External/Status Change
V_7	V_6	V_5	V_4	1	1	0	V_0	Channel A/C Receive Character Available
V_7	V_6	V_5	V_4	1	1	1	V_0	Channel A/C Special Receive Condition
Vector Includes Status = 1 Status High/Status Low = 1								
V_7	0	0	0	V_3	V_2	V_1	V_0	Channel B/D Transmit Buffer Empty
V_7	1	0	0	V_3	V_2	V_1	V_0	Channel B/D External/Status Change
V_7	0	1	0	V_3	V_2	V_1	V_0	Channel B/D Receive Character Available
V_7	1	1	0	V_3	V_2	V_1	V_0	Channel B/D Special Receive Condition
V_7	0	0	1	V_3	V_2	V_1	V_0	Channel A/C Transmit Buffer Empty
V_7	1	0	1	V_3	V_2	V_1	V_0	Channel A/C External/Status Change
V_7	0	1	1	V_3	V_2	V_1	V_0	Channel A/C Receive Character Available
V_7	1	1	1	V_3	V_2	V_1	V_0	Channel A/C Special Receive Condition
Vector Includes Status = 0								
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0	All interrupt sources

V_n = Bit as programmed in WR2.

Figure 6-7. Interrupt Vectors

Physical Pin Locations

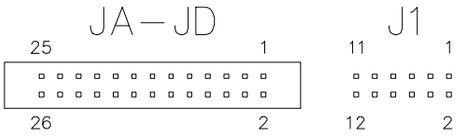


Figure 6-8. I/O Connector Physical Pin Locations

RS-232 Connectors

26-pin header type connectors are used for the RS-232 I/O connectors JA, JB, JC and JD. These connectors are normally converted to DB-25 type connectors using VersaLogic cable #9560 or similar mass terminated cable assembly. Pinouts are included for both the on-board connector and for the DB-25 connector after the port has been converted to this format.

Direct connection to the 26-pin headers can be made using mating connectors such as 3M #3399-7026, AMP #499505-7, or Ansley #609-2641.

JA, JB, JC[1], JD[1] Pin	RS-232 Signal	RS-232 Pin	DCE Signal Direction	DTE Signal Direction
1	–	1	–	–
2	–	14	–	–
3	TxD (BA)	2	IN	OUT
4	RCLK (DB)	15	OUT	IN
5	RxD (BB)	3	OUT	IN
6	–	16	–	–
7	RTS (CA)	4	IN	OUT
8	RCLK (DD)	17	OUT	IN
9	CTS (CB)	5	OUT	IN
10	–	18	–	–
11	DSR (CC)	6	OUT	IN
12	–	19	–	–
13	GND (AB)	7	GND	GND
14	DTR (CD)	20	IN	OUT
15	–	8	–	–
16	–	21	–	–
17	–	9	–	–
18	–	22	–	–
19	–	10	–	–
20	–	23	–	–
21	–	11	–	–
22	TCLK (DA)	24	IN	OUT
23	–	12	–	–
24	–	25	–	–
25	–	13	–	–
26	–	–	–	–

[1] Available on VL-7314 only.

Figure 6-9. RS-232 Connector Pinout

Interrupt Connector

The VL-7314 has four general purpose interrupt inputs and two interrupt outputs accessible through connector J1. (The VL-7312 has only two inputs and one output). A low level signal applied to the general purpose interrupt inputs can initiate unique vectored interrupt requests over the STD Bus. The interrupt output signals are tied to the STD Bus signal INTRQ* through jumpers $V1_b$ and $V1_c$. They are made available at connector J1 pins 9 and 11 for connection to an external interrupt controller card if desired.

J1 Pin	Signal	Description
1	INTA*	General purpose interrupt input A
3	INTB*	General purpose interrupt input B
5	INTC*[1]	General purpose interrupt input C
7	INTD*[1]	General purpose interrupt input D
9	INT1*	Interrupt output from SCC1 (and/or SCC2[1])
11	INT2*[1]	Interrupt output from SCC2
2-12	All even pins grounded	

* Low level signal.
[1] Available on VL-7314 only.

Figure 6-10. Interrupt Connector Pinout

STD Bus Pinout

COMPONENT SIDE				SOLDER SIDE			
Pin	**Signal**	**Flow**	**Description**	**Pin**	**Signal**	**Flow**	**Description**
1	+5V	In	+5 volt power	2	+5V	In	+5 volt power
3	GND	In	Digital ground	4	GND	In	Digital ground
5	VBATT	—	Battery Power	6	DCPWRDWN*	—	DC Power Down
7	D3	I/O	Data bus	8	D7	I/O	Data bus
9	D2	I/O	Data bus	10	D6	I/O	Data bus
11	D1	I/O	Data bus	12	D5	I/O	Data bus
13	D0	I/O	Data bus	14	D4	I/O	Data bus
15	A7	In	Address bus	16	A15	—	Address bus
17	A6	In	Address bus	18	A14	—	Address bus
19	A5	In	Address bus	20	A13	—	Address bus
21	A4	In	Address bus	22	A12	—	Address bus
23	A3	In	Address bus	24	A11	—	Address bus
25	A2	In	Address bus	26	A10	—	Address bus
27	A1	In	Address bus	28	A9	In	Address bus
29	A0	In	Address bus	30	A8	In	Address bus
31	WR*	In	Write strobe	32	RD*	In	Read strobe
33	IORQ*	In	I/O address select	34	MEMRQ*	In	Memory address select
35	IOEXP	In	I/O expansion	36	MEMEX	In	Memory expansion
37	INTRQ1*	—	Interrupt Request 1	38	MCSYNC*	In	Machine cycle sync
39	STATUS1*	In	CPU status	40	STATUS0*	—	CPU status
41	BUSAK*	—	Bus acknowledge	42	BUSRQ*	—	Bus request
43	INTAK*	In	Interrupt acknowledge	44	INTRQ*	Out	Interrupt request
45	WAITRQ*	Out	Wait request	46	NMIRQ*	—	Non-maskable interrupt
47	SYSRESET*	In	System reset	48	PBRESET*	—	Push button reset
49	CLOCK*	In	CPU clock	50	CNTRL*	—	AUX timing
51	PCO	Out	Priority chain out	52	PCI	In	Priority chain in
53	AUXGND	In	±12 volt ground	54	AUXGND	In	±12 volt ground
55	AUX+V	In	+12 volt input	56	AUX-V	In	–12 volt input

* Denotes an active low signal.

Figure 6-11. STD Bus Pinout

Decimal / Hex / ASCII Conversion Chart

The chart below is useful for both ASCII and decimal / hex conversion. The "^" symbol denotes control characters. "^A" represents control A, etc.

Dec.	Hex	ASCII	Dec.	Hex	ASCII	Dec.	Hex	ASCII	Dec.	Hex	ASCII
0	00	NUL	32	20	SPACE	64	40	@	96	60	‘
1	01	^A SOH	33	21	!	65	41	A	97	61	a
2	02	^B STX	34	22	"	66	42	B	98	62	b
3	03	^C ETX	35	23	#	67	43	C	99	63	c
4	04	^D EOT	36	24	$	68	44	D	100	64	d
5	05	^E ENQ	37	25	%	69	45	E	101	65	e
6	06	^F ACK	38	26	&	70	46	F	102	66	f
7	07	^G BEL	39	27	’	71	47	G	103	67	g
8	08	^H BS	40	28	(	72	48	H	104	68	h
9	09	^I HT	41	29	)	73	49	I	105	69	i
10	0A	^J LF	42	2A	*	74	4A	J	106	6A	j
11	0B	^K VT	43	2B	+	75	4B	K	107	6B	k
12	0C	^L FF	44	2C	,	76	4C	L	108	6C	l
13	0D	^M CR	45	2D	-	77	4D	M	109	6D	m
14	0E	^N SO	46	2E	.	78	4E	N	110	6E	n
15	0F	^O SI	47	2F	/	79	4F	O	111	6F	o
16	10	^P DLE	48	30	0	80	50	P	112	70	p
17	11	^Q DC1	49	31	1	81	51	Q	113	71	q
18	12	^R DC2	50	32	2	82	52	R	114	72	r
19	13	^S DC3	51	33	3	83	53	S	115	73	s
20	14	^T DC4	52	34	4	84	54	T	116	74	t
21	15	^U NAK	53	35	5	85	55	U	117	75	u
22	16	^V SYN	54	36	6	86	56	V	118	76	v
23	17	^W ETB	55	37	7	87	57	W	119	77	w
24	18	^X CAN	56	38	8	88	58	X	120	78	x
25	19	^Y EM	57	39	9	89	59	Y	121	79	y
26	1A	^Z SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Figure 6-12. Decimal / Hex / ASCII Conversion Chart

VL-7312 Parts Placement Diagram

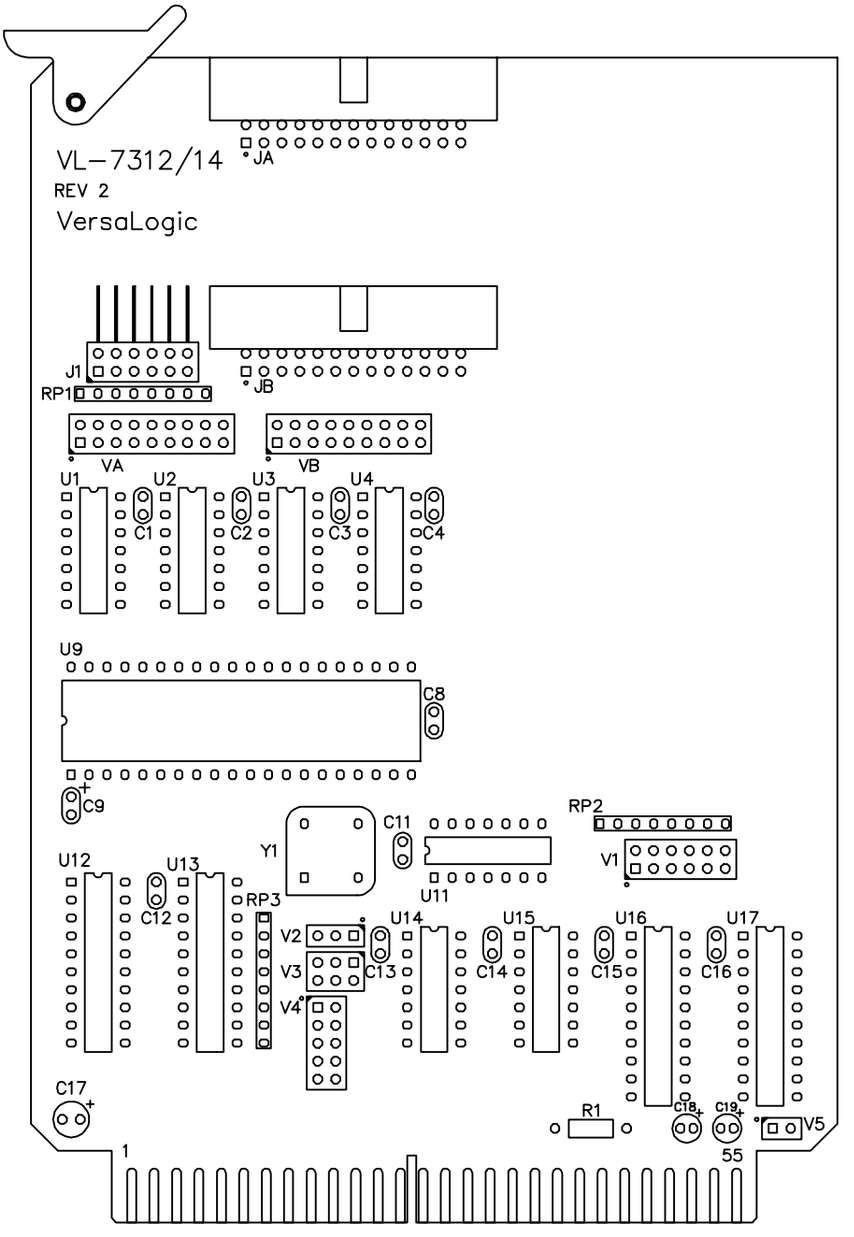


Figure 6-13. VL-7312 Parts Placement Diagram

VL-7314 Parts Placement Diagram

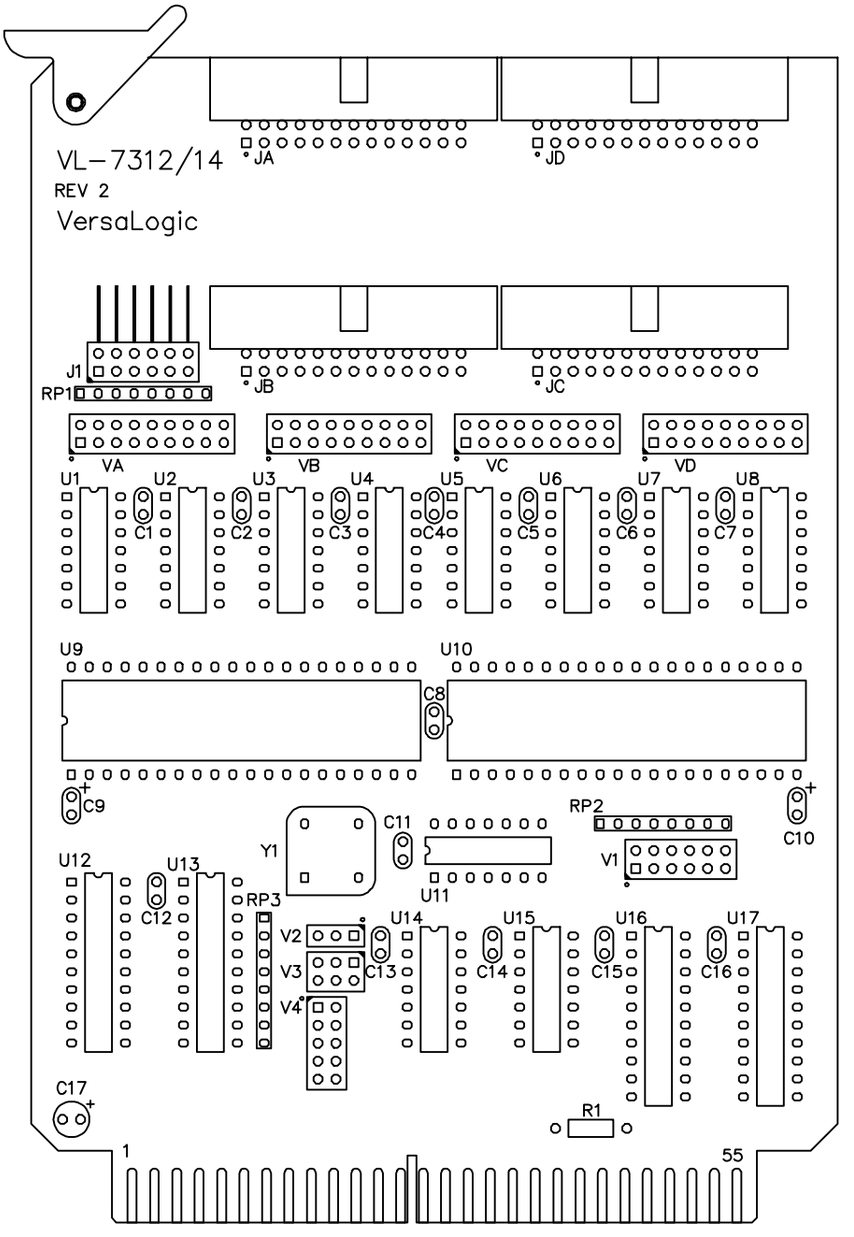
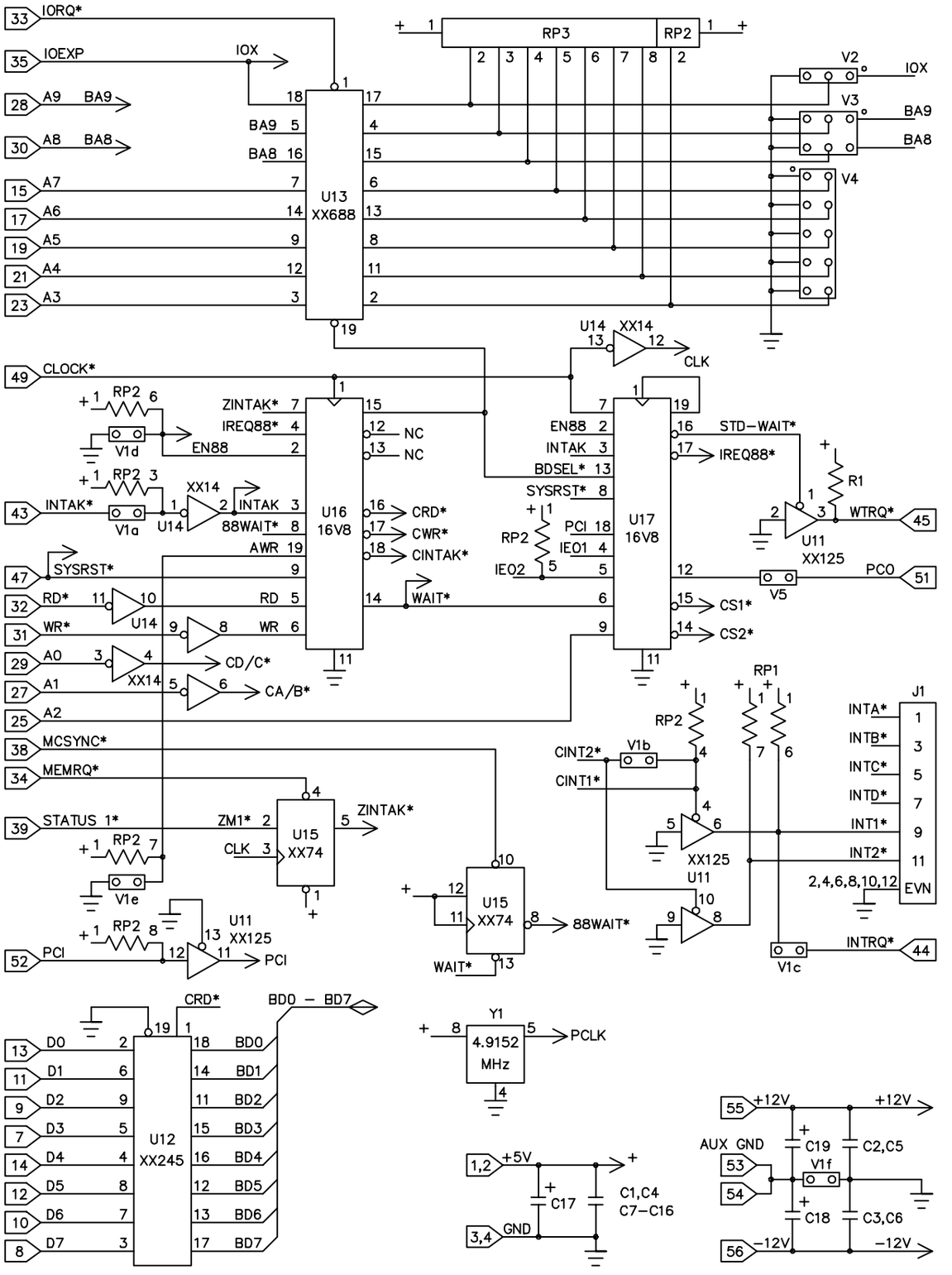


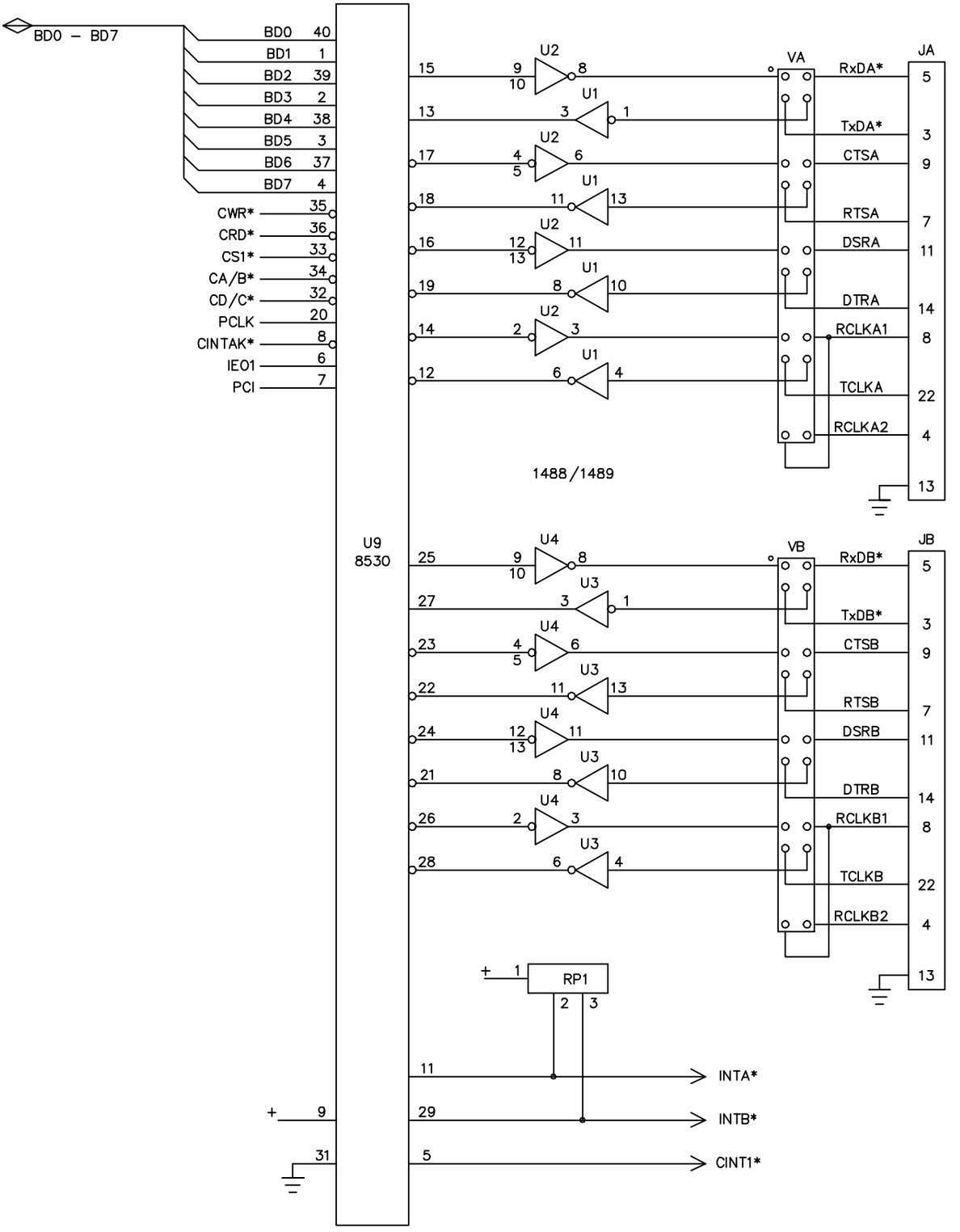
Figure 6-14. VL-7314 Parts Placement Diagram

VL-7312/14 Schematic

REV 2

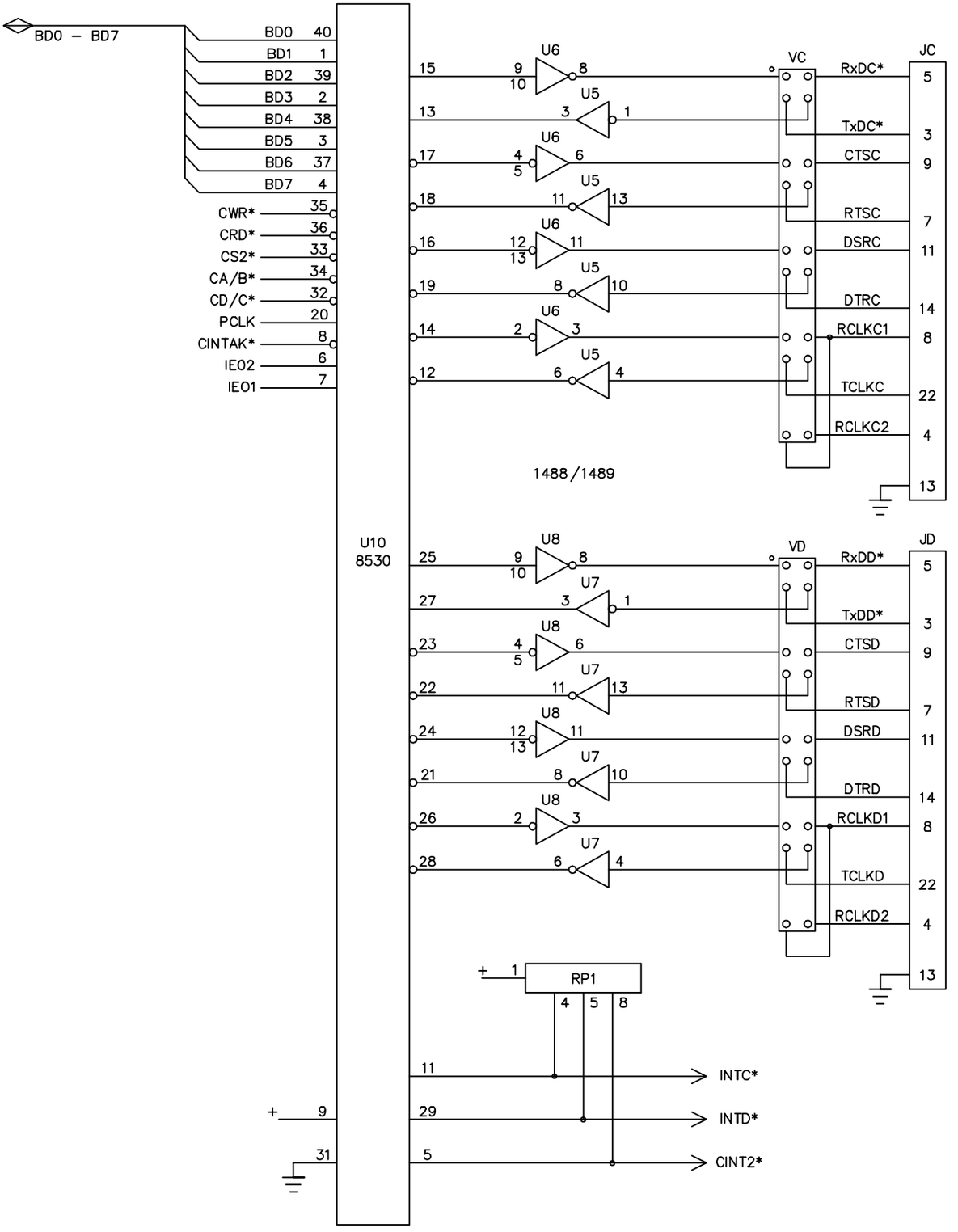


VL-7312/14 Schematic



VL-7314 Schematic

VL-7314 Only



VL-7312 Parts List

Rev. 2.00

Capacitors

C1-C4, C8, C11-C16,	.01 µf Z5U
C9	1 µf tantalum
C17	22 µf electrolytic radial
C18, C19	2.2 µf electrolytic radial

Integrated Circuits

U1, U3	14C89
U2, U4	14C88
U9	8530 6MHz
U11	74LS125
U12	74LS245
U13	74HCT688
U14	74LS14
U15	74LS74
U16	16V8-25QP P7312-A Rev. 3.00
U17	16V8-25QP P7312-B Rev. 3.00

Resistors

R1	2.2K Ω, 5%, ¼ W
RP1, RP2, RP3	10K Ω, 7 res. SIP

Semiconductor

Y1	Crystal oscillator 4.9152 MHz

Miscellaneous

JA, JB	26-pin R/A header
J1	12-pin R/A header (no housing)

VL-73CT12 Parts List

Rev. 2.00

Capacitors

C1-C4, C8, C11-C16,	.01 µf ceramic
C9	1 µf tantalum
C17	22 µf electrolytic radial
C18, C19	2.2 µf electrolytic radial

Integrated Circuits

U1, U3	14C89
U2, U4	14C88
U9	85C30 6MHz
U11	74ACT125 or 74AHCT125
U12	74ACT245
U13	74HCT688
U14	74HCT14
U15	74HCT74
U16	16V8-25QPI P7312-A Rev. 3.00
U17	16V8-25QPI P7312-B Rev. 3.00

Resistors

R1	2.2K Ω, 5%, ¼ W
RP1, RP2, RP3	10K Ω, 7 res. SIP

Semiconductor

Y1	Crystal oscillator 4.9152 MHz

Miscellaneous

JA, JB	26-pin R/A header
J1	12-pin R/A header (no housing)

VL-7314 Parts List

Rev. 2.00

Capacitors

C1-C8, C8, C11-C16,	.01 µf ceramic
C9, C10	1 µf tantalum
C17	22 µf electrolytic radial
C18, C19	2.2 µf electrolytic radial

Integrated Circuits

U1, U3, U5, U7	14C89
U2, U4, U6, U8	14C88
U9, U10	8530 6MHz
U11	74LS125
U12	74LS245
U13	74HCT688
U14	74LS14
U15	74LS74
U16	16V8-25QP P7312-A Rev. 3.00
U17	16V8-25QP P7312-B Rev. 3.00

Resistors

R1	2.2K Ω, 5%, ¼ W
RP1, RP2, RP3	10K Ω, 7 res. SIP

Semiconductor

Y1	Crystal oscillator 4.9152 MHz

Miscellaneous

JA, JB, JC, JD	26-pin R/A header
J1	12-pin R/A header (no housing)

VL-73CT14 Parts List

Rev. 2.00

Capacitors

C1-C8, C8, C11-C16,	.01 µf ceramic
C9, C10	1 µf tantalum
C17	22 µf electrolytic radial
C18, C19	2.2 µf electrolytic radial

Integrated Circuits

U1, U3, U5, U7	14C89
U2, U4, U6, U8	14C88
U9, U10	85C30 6MHz
U11	74ACT125
U12	74ACT245 or 74AHCT245
U13	74HCT688
U14	74HCT14
U15	74HCT74
U16	16V8-25QPI P7312-A Rev. 3.00
U17	16V8-25QPI P7312-B Rev. 3.00

Resistors

R1	2.2K Ω, 5%, ¼ W
RP1, RP2, RP3	10K Ω, 7 res. SIP

Semiconductor

Y1	Crystal oscillator 4.9152 MHz

Miscellaneous

JA, JB, JC, JD	26-pin R/A header
J1	12-pin R/A header (no housing)