ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

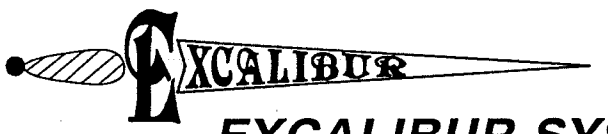
Find the *Excalibur Systems EXC-1553PC/EP* at our website: *Click HERE*

'C' DRIVERS

for the

EXC-1553PCMCIA

User's Manual



EXCALIBUR SYSTEMS, INC.

311 Meacham Avenue • Elmont • N.Y. 11003 • Tel.(516) 327-0000 • Fax(516) 327-4645

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

Introduction 1

1553 General Functions 2

1553 Bus Controller Functions 9

1553 Remote Terminal Functions 20

1553 Bus Monitor Functions 32

Flags ..... Appendix A

A Programming Example (for DOS) ..... Appendix B

A Programming Example (for Windows) ..... Appendix C

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Introduction

The 1553PCMCIA Software Tools Package is a set of 'C' language subroutines designed to aid users of Excalibur EXC-1553PCMCIA cards in writing test programs. These routines provide access to all of the card functions in a structured and straightforward programming environment. The software is designed to operate on a PC compatible running the DOS operating system or the Windows graphical user interface.

The user is expected to be familiar with the functioning of the Mil-Std-1553 bus. Knowledge of the Excalibur card will greatly assist in understanding the rationale of the C driver routines. All structures and flags used by these routines are defined within this document and its Appendices.

Each routine is presented with its formal definition, including data types of all input and output variables. A brief description of the purpose of the routine is provided along with the allowed values for inputs where applicable.

All routines are written as 'C' functions, i.e., they return values. A negative value signifies an error. Full error messages may be printed using the Print_Error routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1 *** General Functions ***

1.1 Init_Socket (int *socket, int *maxmem, int intlevel)

This routine should be the first driver function called by the user. It allows the user to specify the socket that the card is in (allowing multiple cards to be used simultaneously), as well as the amount of host memory to allocate to the card.

inputs:	socket	This is a pointer to an integer containing the socket number in which the card is located (0, 1, 2 or 3). If the card is not found in that socket an error is returned. A value of ANYSOCKET will cause the driver to search all sockets until a card is found.
	maxmem	This is a pointer to an integer containing the maximum window size in Kbytes to assign to the card (16 to 64). In Monitor Mode it is advantageous to have as large a window as possible. If multiple cards are to be supported window size should be limited for each card to leave space for the other cards. Note that the window being allocated is in PC address space between 640K and 1 Meg.
	intlevel	A number between 2 to 15 indicating the desired IRQ line on the PC, which is to be used to send interrupts. (This option consumes 4K of memory). A value of 0 indicates that no interrupts are desired.
outputs:	socket	After the routine returns this value will contain the slot in which the card was found.
	maxmem	After the routine returns this value will contain the amount of memory in Kbytes actually allocated to the card. (This does not include 4K bytes used by an interrupt, if requested). Note that this may be less than the amount requested.
returns:	enocard	no card was found
	enomem	16K window not available
	0	success

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2 Release_Card (void)

This function must be the last driver function to be called by a user program. It releases all system resources used by the EXC-1553PCMCIA and powers down the card. Failure to call this routine will lead to failure in the next call to Init_Socket.

returns: SUCCESS
or an error condition

1.3a Print_Error (int errorcode)

This function is used to print out error messages for debugging. Every driver function returns a value greater than or equal to 0, if it executes without an error condition. If there is an error, it returns a negative error code. Error codes are defined in the file ERROR.H. This function is meant to be used as follows:

```
errorcode = Init_Socket( &socket, &maxmem, NO_INTERRUPTS );
if (errorcode < 0)
   Print_Error (errorcode);
```

inputs: errorcode negative value returned by any driver function

1.3b char far * Get_Error_Msg (int errorcode)

This function is the Windows version of Print_Error. Instead of actually printing, it returns a string containing the error message. This function is meant to be used as follows:

```
errorcode = Init_Socket( &socket, &maxmem, NO_INTERRUPTS );
if (errorcode < 0)
   MessageBox(hwnd, Get_Error_Msg (errorcode), szAppName,
              MB_OK | MB_ICONSTOP);
```

inputs: errorcode negative value returned by any driver function

returns: a far pointer to the error message

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.4 Set_Mode (int mode)

This function sets up the card as a Bus Controller, Remote Terminal or Bus Monitor. It must be called prior to any mode specific routines. **Note:** This routine will always reset the card.

inputs:	mode	BC_MODE	For Bus Controller mode
		RT_MODE	For Remote Terminal mode
		BM_MODE	For Bus Monitor mode
		BM_RT_MODE	For simultaneous Remote Terminal/ Bus Monitor mode

returns:	0	For success
		einval for bad input

1.5 Stop_Card (void)

This routine may be used in any mode to halt operation of the card.

returns:	0	Always

1.6 Set_Broadcast (int flag)

This routine is used to tell the card how to relate to commands sent to RT #31. It has no effect in Bus Controller mode. In RT mode, if BROADCAST is set, messages to RT 31 will be recorded as a broadcast command regardless of the address the RT was set up for. In Bus Monitor mode messages to RT 31 will not be expected to generate a status word on the part of the receiving RT.

input:	flag	BROADCAST	If the card is to treat commands to RT 31 as broadcast messages.
		NOBROADCAST	If the card is to treat commands to RT 31 as regular messages.

returns:	0	For success
	einval	For illegal 'flag' value.

1.7 Get_Broadcast (void)

This routine returns the current value of the broadcast flag.

returns:	BROADCAST	If the card is set up to treat commands to RT 31 as broadcast commands.
	NOBROADCAST	If the card is set up to treat commands to RT 31 as regular commands.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.8 Get_BIT (void)

This routine returns information on the current health of the card.

Bit	Name	Description
14	WRAPF	Wrap Fail. The EXC-1553PCMCIA automatically compares the transmitted word (encoder word) to the reflected decoder word by way of the continuous loop-back feature. If the encoder word and reflected word do not match, the WRAPF bit is asserted. The loop-back path is via the MIL-STD-1553 bus transceiver.
12	BITF	BIT Fail. Assertion of this bit indicates a BIT (Built In Test) failure. Interrogate bit 11 and 10 to determine the specific channel that failed.
11	CHAF	Channel A Fail. Assertion of this bit indicates a BIT test failure in Channel A.
10	CHBF	Channel B Fail. Assertion of this bit indicates a BIT test failure in Channel B.

Ignore all other bits.

1.9 Get_Pending (void)

This routine returns a list of interrupts that occurred since it was last called.

Bit	Name	Description
14	WRAPF	Wrap Fail Interrupt. The EXC-1553PCMCIA automatically compares the transmitted word (encoder word) to the reflected decoder word by way of the continuous loop-back feature. If the encoder word and reflected word do not match, the WRAPF bit is asserted and an interrupt is generated. The loop-back path is via the MIL-STD-1553 bus transceiver.
13	TAPF	Terminal Address Parity Fail Interrupt. This bit reflects the outcome of the remote terminal address parity check. A logic one indicates a parity failure. When a parity error occurs, the EXC-1553PCMCIA does not begin operation, the TAPF bit is asserted here and in the BIT Word Register, and an interrupt is generated (if not masked). **RT MODE ONLY.**
12	BITF	BIT Fail Interrupt. Assertion of this bit indicates a BIT failure. In BC mode, an interrupt is generated and command processing stops.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

11	MERR	Message Error Interrupt. Assertion of this bit indicates the occurrence of a message error. The EXC-1553PCMCIA can detect Manchester, sync-field, word count, 1553 word parity, bit count, and protocol errors. This bit will be set and an interrupt generated (if not masked) after message processing is complete.
RT MODE ONLY		
10	SUBAD	Subaddress Accessed Interrupt. Assertion of this bit indicates a subaddress, for which Enable_Subaddr_Int has been called with inttype = ACCESS, has transacted a message.
9	BDRCV	Broadcast Command Received Interrupt. Assertion of this bit indicates a subaddress, for which Enable_Subaddr_Int has been called with inttype = BROADCAST, has received a Broadcast message. The EXC-1553PCMCIA suppresses status word transmission.
7	ILCMD	Illegal Command Interrupt. This bit is set to a logic one to indicate the reception of an illegal command by the EXC-1553PCMCIA. Upon receipt of this command, the EXC-1553PCMCIA responds with a status word only; Bit 9 of the status word is set to a logic one.
BC MODE ONLY		
5	EOL	End Of List Interrupt. Assertion of this bit indicates that the EXC-1553PCMCIA is at the end of the command block. An interrupt is generated.
4	ILLCMD	Illogical Command Interrupt. Assertion of this bit indicates that an illogical command (i.e., Transmit Broadcast or improperly formatted RT-RT message) was written into the Command Block. The EXC-1553PCMCIA checks for RT-RT Terminal address field match, RT-RT transmit/receive bit mismatch and correct order, and broadcast transmit commands. If illogical commands occur, the EXC-1553PCMCIA will halt execution.
3	ILLOP	Illogical Opcode Interrupt. Assertion of this bit indicates an illogical opcode (i.e., any reserved opcode) was used in the command block. The EXC-1553PCMCIA halts operation if this condition occurs. An interrupt is generated.
2	RTF	Retry Fail Interrupt. Assertion of this bit indicates all programmed retries failed. An interrupt is generated.
1	CBA	Command Block Accessed Interrupt. Assertion of this bit indicates a command block was accessed (Opcode 1010), if enabled. An interrupt is generated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MONITOR MODE ONLY

0 MBC Monitor Block Counter Interrupt. This bit is set if the EXC-1553PCMCIA's monitor block counter reaches zero (transition from 1 to 0). It should be noted that the Monitor does not discriminate between error-free messages and those messages with errors.

1.10 Set_Protocol (int mode)

This routine sets up the card to meet either Mil-Std-1553A or Mil-Std-1553B specifications. (Note: the default is Mil-Std-1553B).

inputs: mode

MIL_STD_1553A **RT MODE**

* sends status within 7 usec
* ignores T/R bit for mode codes
* subaddr 0 is mode subaddress
* only ME and TF bits defined in status word, all others user programmable via Set_RT_Status
* broadcast of all mode codes except 0 and 2 allowed

BUS CONTROLLER / MONITOR MODE

* will expect RT response within 11 usec
* defines all mode codes without data
* defines subaddress 0 as a mode code

MIL_STD_1553B according to Mil-Std-1553B spec

returns 0 for success
einval for bad input

1.11 Set_Interrupt_Mask (unsigned int intr)

The default of the system is for no interrupts to be generated. If the user wishes to receive interrupts for particular events, Set_Interrupt_Mask must be used to notify the system which events are to cause interrupts.

inputs: intr Each bit represents a type of interrupt to be generated. Bit definitions are as described for Get_Pending. If more than one interrupt type is requested, Get_Pending should be called to find out the cause of the current interrupt.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.12 Reset_Interrupt (void)

Reset_Interrupt should be called immediately upon entry into the user's interrupt service routine. Failure to do so will make it impossible to ever get another interrupt. (This should be the only 1553PCMCIA driver function called within the interrupt service routine).

1.13 Change_Socket (int socket)

This routine enables the user to control more than 1 PCMCIA card within a single program, by switching between two or more active sockets.

For example, to control cards in sockets 0 and 1:

```
socket = 0;
Init_Socket (&socket, &memsize, NO_INTERRUPTS);
/* --- code relating to card in socket 0 */

socket = 1;
Change_Socket(socket);
Init_Socket(&socket, &memsize, NO_INTERRUPTS);
/* --- code relating to socket 1 */

Change_Socket(0);  /* --- may switch back and forth at any time */
```

input: socket number: 0, 1, 2 or 3

returns: previous socket number

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2 *** Bus Controller Functions ***

The Bus Controller operation is based on the manipulation of two data structures. The Command Stack and the Data Block.

Each Command Stack entry (Command Block) represents either a 1553 message or a control instruction such "End of List" or "Skip" (see Opcode Definitions). The user fills in Command Stack entries via the Set_BC_Cmd routine and reads them back via the Get_BC_Cmd routine.

The Data Block is an area of memory associated with a particular Command Stack entry that contains the data associated with that command. This data area must be filled in by the user for RT Receive commands (Load_Datablk) or should be read by the user following an RT Transmit command (Read_Datablk).

Please note that while the following discussion closely follows the discussion in the EXC-1553PCMCIA User's Manual, certain changes have been made to account for the slightly different functionality supplied by the drivers. In particular, no pointers to Dual Port memory are setup by the user. This shields the user from low level memory management considerations.

COMMAND STACK ENTRY

Control Word
Command Word 1
Command Word 2
Status Word 1
Status Word 2
Timer Value
Data Block Number
Branch Entry Number

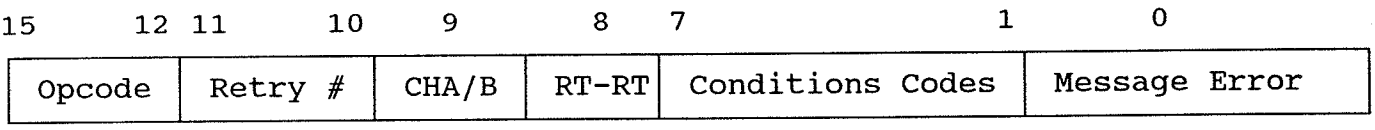
'C' language structure

```
struct CMDENTRY
 {
  unsigned int control;
  unsigned int command1;
  unsigned int command2;
  unsigned int status1;
  unsigned int status2;
  unsigned int timer;
  unsigned int datablk;
  unsigned int gotocmd;
 }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Control Word

The first memory location of each BC Mode Command Block contains the control word. Each control word contains the opcode, retry number, bus definition, RT-RT instruction, condition codes, and the message error bit. The control word is defined below.



Bit Number	Description
15-12	Opcode. These bits define the opcode to be used by the EXC-1553PCMCIA for that particular Command Block. If the opcode does not perform any 1553 function, all other bits are ignored. Each of the available opcodes is defined in the next section.
11-10	Retry Number. These bits define the number of retries for each individual Command Block if the retry opcode is used.

BIT 11	BIT 10	# of Retries
0	1	1
1	0	2
1	1	3
0	0	4

9	Bus A/B. This bit defines on which of the two buses the command will be transmitted. (Logic 1 = Bus A, Logic 0 = Bus B).
8	RT-RT Transfer. This bit defines whether or not the present Command Block is an RT-RT transfer and if the EXC-1553PCMCIA should transmit the second command word. Data associated with an RT-RT is always stored by the EXC-1553PCMCIA.
7-1	Condition Codes. These bits define the condition code the EXC-1553PCMCIA uses for that particular Command Block. Each of the available condition codes will be defined in a subsequent section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

0	Message Error. Assertion of this bit indicates a protocol message error occurred in the RT's response. For this occurrence, the EXC-1553PCMCIA will overwrite this bit prior to storing the Control Word into memory. Noise on the 1553 bus may be one example of such an error.

Opcode Definition

Opcode	Definition
0000	End Of List. This opcode instructs the EXC-1553PCMCIA that the end of the command block has been encountered. Command processing stops and the interrupt is generated if the interrupt is enabled. No command processing takes place (i.e., no 1553).
0001	Skip. This opcode instructs the EXC-1553PCMCIA to load the message-to-message timer with the value stored in timer value location (timer value is $41^2/_3$ nanoseconds per bit). The EXC-1553PCMCIA will then wait the specific time before proceeding to the next command block. This opcode allows for scheduling of specific time between message execution. No command processing takes place (i.e, no 1553).
0010	Go To. This opcode instructs the EXC-1553PCMCIA to "go to" the command block as specified in the branch entry location. No command process takes place (i.e., no 1553).
0011	Built-in Test. This opcode instructs the card chip to perform an internal built-in test. If the device passes the built-in test, then processing of the next command block will continue. However if the device fails the built-in test, then processing stops and an interrupt is generated, if the interrupt is enabled. No command processing takes place (i.e., no 1553).
0100	Execute Block; Continue. This opcode instructs the EXC-1553PCMCIA to execute the current command block and proceed to the next command block. This opcode allows for continuous operations. **This is the standard method for sending out a 1553 message.**
0101	Execute Block; Branch. This opcode instructs the EXC-1553PCMCIA to execute the current command block and unconditionally branch to command block specified in the branch entry location.
0110	Execute Block; Branch on Condition. This opcode instructs the EXC-1553PCMCIA to execute the current command block and branch only if the condition is met. If no conditions are met, the opcode appears as an execute and continue.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

0111 Retry on Condition. This opcode instructs the EXC-1553PCMCIA to perform automatic retries, as specified in the control word, if particular conditions occur. If no conditions are met, the opcode appears as an execute and continue.

1000 Retry on Condition; Branch. This opcode instructs the EXC-1553PCMCIA to perform automatic retries, as specified in the control word, if particular conditions occur. If the conditions are met, the EXC-1553PCMCIA retries. Once all retries have executed, the EXC-1553PCMCIA branches to the location as specified in the branch address location. If no conditions are met, the opcode appears as an execute and branch.

1001 Retry on Condition; Branch if all Retries Fail. This opcode instructs the EXC-1553PCMCIA to perform automatic retries, as specified in the control word, if particular conditions occur. If the conditions are met and all the retries fail, the EXC-1553 PCMCIA branches to the command block specified in the branch entry location. If no conditions are met, the opcode appears as an execute and continue.

1010 Interrupt; Continue. This opcode instructs the EXC-1553PCMCIA to interrupt and continue processing on the next command block. When using this opcode, no 1553 processing occurs.

1011 Call. This opcode instructs the EXC-1553PCMCIA to "go to" the command block specified in the branch entry location without processing this block. The next command block address is saved in an internal register so that the EXC-1553PCMCIA may remember one address and return to the next command block. No command processing takes place (i.e., no 1553).

1100 Return to Call. This opcode instructs the EXC-1553PCMCIA to return to the command block address saved during the Call opcode. No command processing takes place (i.e., no 1553).

1101 Reserved. The EXC-1553PCMCIA will generate an illegal opcode interrupt (if interrupt enabled) and automatically stop execution if a reserved opcode is used.

1110 Load Minor Frame Timer. This opcode instructs the EXC-1553PCMCIA to load the minor frame timer (MFT) with the value stored in the eighth location of the current command block. The timer will be loaded after the previous MFT has decremented to zero. After the MFT timer is loaded with the new value, the EXC-1553PCMCIA will proceed to the next command block. The timer value is interpreted as 64 microseconds per bit. No command processing takes place (i.e., no 1553).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1111 Return to Branch. This opcode instructs the EXC-1553PCMCIA to return to the command block address saved during a Branch opcode. No command processing takes place (i.e., no 1553).

Note: For retries with interrupts enabled, all interrupts are logged after message processing is complete.

Condition Codes

Condition codes have been provided as a means for the EXC-1553PCMCIA to perform certain functions based on the RT's status word. In an RT-RT transfer, the conditions apply to both of the status words. Each bit of the condition codes is defined below.

Bit Number	Description
7	Message Error. This condition will be met if the EXC-1553PCMCIA detects an error in the RT's response, or if it detects no response. (The EXC-1553PCMCIA will wait 15 us in 1553B mode and 11 us in 1553A mode before declaring an RT no response).
6	Status Word Response with the Message Error bit set (Bit time 9 in 1553A mode). This condition is met if the EXC-1553PCMCIA detects that the RT's status word has the Message Error bit set.
5	Status Word Response with the Busy bit set (Bit time 16 in 1553A mode). This condition is met if the EXC-1553PCMCIA detects that the RT's status word has the Busy bit set.
4	Status Word Response with the Terminal Flag bit set (Bit time 19 in 1553A mode). This condition is met if the EXC-1553PCMCIA detects that the RT's status word has the Terminal Flag bit set.
3	Status Word Response with the Subsystem Fail bit set (Bit time 17 in 1553A mode). This condition is met if the EXC-1553PCMCIA detects that the RT's status word has the Subsystem Fail bit set.
2	Status Word Response with the Instrumentation bit set (Bit time 10 in 1553A mode). This condition is met if the EXC-1553PCMCIA detects that the RT's status word has the Instrumentation bit set.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1 Status Word Response with the Service Request bit set (Bit time 11 in 1553A mode). This condition is met if the EXC-1553PCMCIA detects that the RT's status word has the Service Request bit set.

COMMAND WORDS

The Command Word entries are filled in by the user with 1553 command words. In most 1553 messages, only the first command word needs to be initialized. However, in an RT-RT transfer, the first command word is the Receive Command and the second command word is the Transmit Command.

STATUS WORDS

The Status word entries are filled in by the card with the 1553 Status word(s) returned from the RTs on the bus. As the RT responds to the BC's command, the corresponding status word will be stored in Status Word 1. In an RT-RT transfer, the first status word will be the status of the Transmitting RT while the second status word will be the status of the Receiving RT.

TIMER VALUE

The Timer Value field is used for one of two purposes. First, the value may be used to set up minor frame schedules when using the Load Minor Frame Timer opcode (1110). The MFT counter is clocked by a 64 us internal clock. The MFT counter runs continuously during message processing and must decrement to zero prior to loading the next Minor Frame time value. Second, the value may be used as a message-to-message timer (MMT) when using the Skip opcode (0001). The MMT timer value is 41 2/3 nanoseconds per bit and allows for the scheduling of specific time between message execution. The minimum intermessage gap is 28 microseconds.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DATA BLOCK NUMBER

The Data Block Number is a number from 0 to 750 chosen by the user to be associated with this message. The data block is only used for opcodes that cause 1553 messages to be sent. In an RT-RT transfer, the EXC-1553PCMCIA stores the transmitted data into the data block. The same data block may be associated with more than one command entry block facilitating the transmission of the same data to multiple RTs. The data block number is used in the Read_BC_Datablk and Load_BC_Datablk routines to read and write the data in the data blocks.

BRANCH ADDRESS

The Branch Address location in the BC Mode Command Block contains the Message Stack Entry number of the entry to branch to for those opcodes that alter the command sequence.

COMMAND BLOCK CHAINING

The first Command Block executed will always be Command Block 0. Command Blocks will execute in a contiguous fashion as long as no "go to", "branch", "call", or "return" opcodes are used. With the use of these opcodes, almost any memory configuration is possible. Figures 1 and 2 show how several Command Blocks may be linked together to form a command frame and how branch opcodes may be used to link minor frames. The minimum BC intermessage gap is 28.0 us.

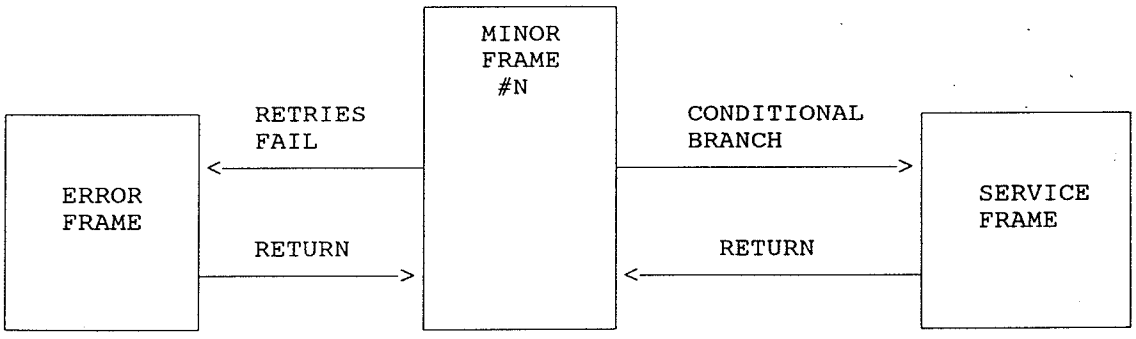


Figure 1: Minor Frame Branching

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

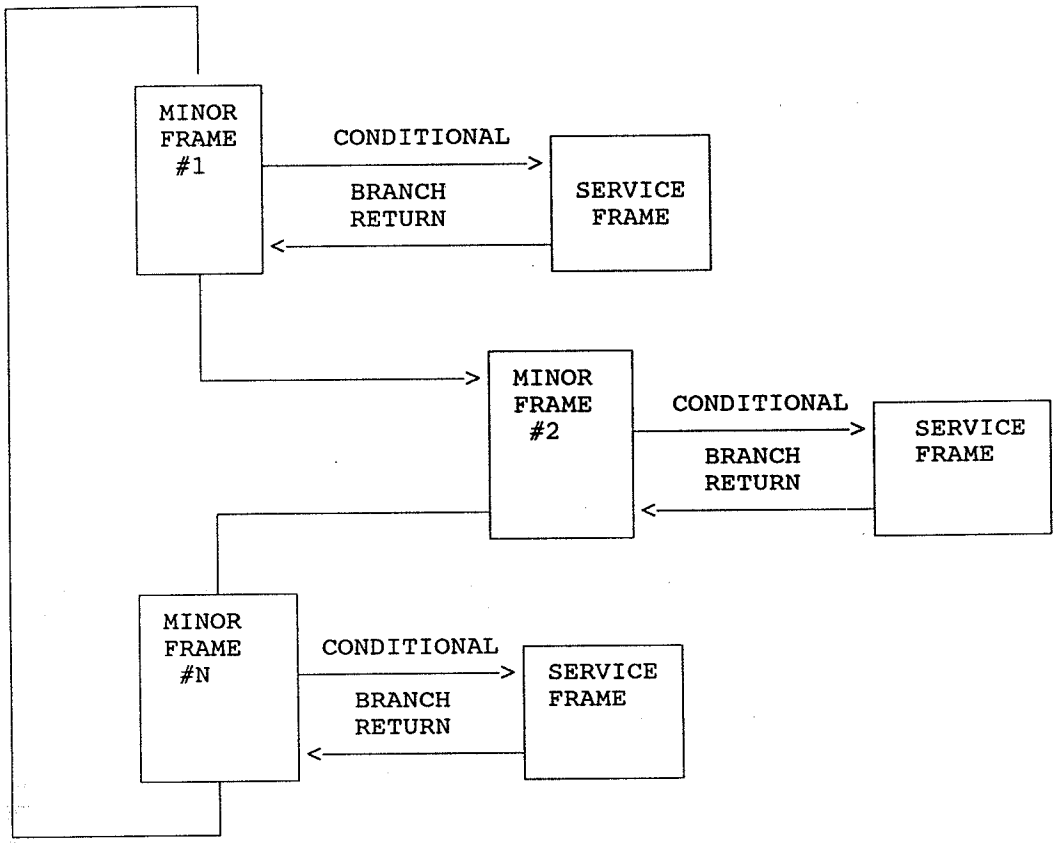


Figure 2: Major Frame Sequencing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.1 Set_BC_Cmd (unsigned int cmdnum, struct CMDENTRY far *entry)

This routine sets up a single command block in the command stack. The user fills in the CMDENTRY structure according to the rules described above. Note that the status word fields are not used for this routine and other fields may or may not be needed according to the opcode inserted in the control word.

inputs:	cmdnum	The command block number. The first command executed is command block 0. Following block 0 blocks will be executed either sequentially or according to the jump sequence requested by the user. The maximum cmdnum value is 1023.
	entry	A pointer to a structure of type CMDENTRY as described in the introduction to this section.
returns:	0	For success
	einval	For bad cmdnum (greater than 1023)
	emode	If Set_Mode was not called with BC_MODE as its argument prior to calling this routine.

2.2 Get_BC_Cmd (unsigned int cmdnum, struct CMDENTRY far *entry)

This routine reads a single command block in the command stack. Most of the data read will be the data that was previously set by the user. The card will fill in values for the status field(s) and the Message Error bit within the control word.

inputs:	cmdnum	The command block number. The first command executed is command block 0. Following block 0 blocks will be executed either sequentially or according to the jump sequence requested by the user. The maximum cmdnum value is 1023.
	entry	A pointer to a structure of type CMDENTRY as described in the introduction to this section.
returns:	0	For success
	einval	For bad cmdnum (greater than 1023)
	emode	If Set_Mode was not called with BC_MODE as its argument prior to calling this routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.3 Load_BC_Datablk (unsigned int blknum, unsigned int wdcnt, unsigned int far *data)

This routine is used to set up the data for an RT receive command.

inputs:	blknum	The data block number. Block numbers run from 0 to 750. This is the same block number as that filled in by the user in the CMDENTRY structure used in the Set_BC_Cmd routine.
	wdcnt	The number of words the user wishes to write to the buffer.
	data	A pointer to the data the user wishes to write. The data buffer must be at least 'wdcnt' words long.
returns:	0	For success
	einval	If blknum is greater than 750 If wdcnt is greater than 32
	emode	If Set_Mode was not called with BC_MODE as its argument prior to calling this routine.

2.4 Read_BC_Datablk (unsigned int blknum, unsigned int wdcnt, unsigned int far *data)

This routine is used to read the data for an RT transmit command or RT to RT command. It may also be used to read back data written by the user with the Load_BC_Datablk routine.

inputs:	blknum	The data block number. Block numbers run from 0 to 750. This is the same block number as that filled in by the user in the CMDENTRY structure used in the Set_BC_Cmd routine.
	wdcnt	The number of words the user wishes to read from the buffer.
	data	A pointer to the data area the user wishes the data to be written to. The data buffer must be at least 'wdcnt' words long.
returns:	0	For success
	einval	If blknum is greater than 750 If wdcnt is greater than 32
	emode	If Set_Mode was not called with BC_MODE as its argument prior to calling this routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.5 Run_BC (void)

This routine should be called only after all 1553 messages have been set up via calls to Set_BC_Cmd. It will start execution of the Message Stack. Get_BC_Cmd, Load_BC_Datablk and Read_BC_Datablk may be called following Run_BC to read and alter data in real time and to check the status of messages.

returns	emode	if card is not in BC mode
	0	(for success) otherwise

2.6 Last_BC_Cmd (void)

This routine returns the number of the last Command Block to be executed. This corresponds to the cmdnum argument in Set_BC_Cmd or Get_BC_Cmd.

returns	emode	if card is not in BC mode
	cmdnum	otherwise

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following sections discuss the data structures that result from command processing. For each complete message processed, the EXC-1553PCMCIA generates a Message Information word and Time-Tag word. These words aid the host in further message processing. The Message Information word contains word count, message type, and message error information. The Time-Tag word is a 16-bit word representing the value of a running counter within the EXC-1553PCMCIA. The precision of the time tag is 64 microseconds per bit. These words are read using the Read_Datablk subroutine in the following sequence.

```
data[0]    ;Message Information Word
data[1]    ;Time-Tag word
data[2]    ;1st  data word
  .
  .
data[32]   ;32nd data word
```

The format of the Message Information word differs slightly from message type to message type. Each message type is described below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RECEIVE INFORMATION (INFO) WORD

The following bits describe the Information Word contents for a non mode code receive command.

Bit	Name	Description
15-11	WC[4-0]	Word Count Bits. These five bits contain word count information extracted from the receive command word bits 15 to 19.
10	reserved	
9	CHA/B	Channel A/B. Assertion of this bit indicates that the message was received on channel A. Conversely, if this bit is set to logic zero, the message was received on channel B.
8	RTRT	Remote Terminal to Remote Terminal Transfer. The command processed was an RT-to-RT transfer.
7	ME	Message Error. Assertion of this bit indicates a message error condition was observed during processing. See bits 0 to 4 for details.
6-5	reserved	
4	ILL	Illegal Command Received. Assertion of this bit indicates the command received was an illegal command.
3	TO	Time-Out Error. Assertion of this bit indicates the EXC-1553PCMCIA did not receive the proper number of data words, i.e., the number of data words received was less than the word count specified in the command word.
2	OVR	Overrun Error. Assertion of this bit indicates the EXC-1553PCMCIA received a word when none was expected or the number of data words received was greater than expected.
1	PRTY	Parity Error. Assertion of this bit indicates the EXC-1553PCMCIA observed a parity error in the incoming data words.
0	MAN	Manchester Error. Assertion of this bit indicates the EXC-1553PCMCIA observed a Manchester error in the incoming data words.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TRANSMIT INFORMATION (INFO) WORD

The following bits describe the Information Word contents for a non mode code transmit command.

Bit	Name	Description
15-11	WC[4-0]	Word Count Bits. These five bits contain word count information extracted from the receive command word bits 15 to 19.
10	reserved	
9	CHA/B	Channel A/B. Assertion of this bit indicates that the message was received on the A bus. Conversely, if this bit is set to logic zero, the message was received on the B bus.
8	reserved	
7	ME	Message Error. Assertion of this bit indicates a message error condition was observed during processing. See bits 0 to 4 for more detail.
6-5	reserved	
4	ILL	Illegal Command Received. Assertion of this bit indicates the command received was an illegal command.
3	reserved	
2	OVR	Overrun Error. Assertion of this bit indicates the EXC-1553PCMCIA received a data word with a Transmit Command.
1-0	reserved	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE CODE RECEIVE INFORMATION (INFO) WORD

The following bits describe the Information Word contents for a mode code receive command.

Bit	Name	Description
15-11	MC[4-0]	Mode Code. These five bits contain the mode code information extracted from the receive command word bits 15 to 19.
10	reserved	
9	CHA/B	Channel A/B. Assertion of this bit indicates that the message was received on the A bus. Conversely, if this bit is set to logic zero, the message was received on the B bus.
8	RTRT	Remote Terminal to Remote Terminal Transfer. Assertion of this bit indicates the command processed was an RT-to-RT transfer.
7	ME	Message Error. Assertion of this bit indicates a message error condition was observed during processing. See bits 0 to 4 for details.
6-5	reserved	
4	ILL	Illegal Command Received. Assertion of this bit indicates the command received was an illegal command.
3	TO	Time-out Error. Assertion of this bit indicates the EXC-1553PCMCIA did not receive the proper number of data words, i.e., the number of data words received was less than the word count specified in the command word.
2	OVR	Overrun Error. Assertion of this bit indicates the EXC-1553PCMCIA received a word when none was expected, or the number of data words received was greater than expected.
1	PRTY	Parity Error. Assertion of this bit indicates the EXC-1553PCMCIA observed a parity error in the incoming data words.
0	MAN	Manchester Error. Assertion of this bit indicates the EXC-1553PCMCIA observed a Manchester error in the incoming data words.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE CODE TRANSMIT INFORMATION (INFO) WORD

The following bits describe the Information Word contents for a mode code transmit command.

Bit	Name	Description
15-11	MC[4-0]	Mode Code. These five bits contain the mode code information extracted from the command word bits 15 to 19.
10	reserved	
9	CHA/B	Channel A/B. Assertion of this bit indicates that the message was received on the A bus. Conversely, if this bit is set to logic zero, the message was received on the B bus.
8	reserved	
7	ME	Message Error. Assertion of this bit indicates a message error condition was observed during processing. See bits 0 to 4 for details.
6-5	reserved	
4	ILL	Illegal Command Received. Assertion of this bit indicates the command received was an illegal command.
3	reserved	
2	OVR	Overrun Error. Assertion of this bit indicates the EXC-1553PCMCIA received a data word with a Transmit Command.
1-0	reserved	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.1 Set_RT_Num (int rtid)

This routine sets the RT address of the EXC-1553PCMCIA. This routine must be called prior to calling Run_RT.

input:	rtid	The RT address of the EXC-1553PCMCIA.
returns:	0	For success
	einval	If rtid is greater than 31
	emode	If Set_Mode was not called with RT_MODE or BM_RT_MODE as its argument prior to calling this routine.

3.2 Run_RT (void)

This routine should be called only after calling Set_RT_Num. It will start execution of the card causing commands to the card's RT address to be responded to.

returns:	0	For success
	emode	If Set_Mode was not called with RT_MODE or BM_RT_MODE as its argument prior to calling this routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.3 Load_Datablk (unsigned int subaddress, unsigned int modecode, unsigned int far *data)

This routine is used to set up the data to be transmitted by the RT for an RT transmit command. Data blocks are loaded on the basis of the subaddress. Separate buffers are used for regular and mode commands.

inputs:	subaddress	The subaddress of the command that is to have the data transmitted in response, or MODECODE if mode code data is to be loaded.
	modecode	If subaddress is MODECODE, this parameter contains the mode code number that the data is to be sent for. (For example, if the data is to be sent with a Send Vector mode command, 'modecode' should be set to 16). If the subaddress is not MODECODE, set this parameter to 0.
	data	A pointer to the data the user wishes to write. The data buffer must be at least 32 words long.
returns:	0	For success
	einval	If subaddress is greater than 31 and not equal to MODECODE If modecode is greater than 31
	emode	If Set_Mode was not called with RT_MODE or BM_RT_MODE as its argument prior to calling this routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4 Read_Datablk (unsigned int subaddress, unsigned int modecode, unsigned int far *data, int msgtype, int txorrx)

This routine is used to read the data associated with a particular subaddress. Both data received by the card via receive messages and data set by the user via Load_Datablk may be read using this routine. In addition to the data, Read_Datablk will return a status word and a time tag associated with the last command directed to that datablk.

inputs:

subaddress	The subaddress whose data we are interested in, or MODECODE if mode code data is to be read.
modecode	If subaddress is MODECODE, this parameter contains the mode code number that the data is associated with. (For example, if the user wishes to read the data word associated with the Synchronize With Data mode command, 'modecode' should be set to 17). If the subaddress is not MODECODE, set this parameter to 0.
data	A pointer to the data area the user wishes the data to be written to. The data buffer must be at least 34 words long for a non mode subaddress and at least 3 words long for a mode code subcommand. The first word returned will be the Message Information Word, the second will be the time tag word, and any data will follow as described in the introduction to this section.
msgtype	STANDARD - for non mode, non broadcast message BRDCAST - for non mode, broadcast message MODE - for mode, non broadcast message BROADCAST_MODE for mode, broadcast message
txorrx	TRANSMIT - for transmit message type RECEIVE - for receive message type

returns:

0	For success
einval	If subaddress is greater than 31 and not equal to MODECODE If modecode is greater than 31 If msgtype or txorrx are not one of the values described above
emode	If Set_Mode was not called with RT_MODE or BM_RT_MODE as its argument prior to calling this routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5 Set_Legal_Command (unsigned int subaddress, int brdcst, int txorrx, int ismode, int legal)

The EXC-1553PCMCIA allows the user the option of illegalizing particular subaddresses within the RT. Broadcast commands may be illegalized separately from non broadcast commands. Mode codes may be illegalized separately from non mode codes and Transmit commands may be illegalized separately from receive commands. This routine is used both to set commands legal as well as to set them illegal. For convenience, comments refer to setting commands illegal. The default is for all commands allowed by Mil-Std 1553 (all but Broadcast Transmit) to be legal.

inputs:	subaddress		For 'ismode' = NOMODECODE, the subaddress is to be regarded as illegal. For 'ismode' = MODECODE, the mode code number is to be regarded as illegal.
	brdcst	BROADCAST	If commands to RT 31 are to be illegalized.
		NOBROADCAST	If a command to the card's address is to be illegalized.
	txorrx	TRANSMIT	If a transmit command to 'subaddress' is to be illegalized.
		RECEIVE	If a receive command to 'subaddress' is to be illegalized.
	ismode	MODECODE	If a particular modecode is to be illegalized.
		NOMODECODE	If a non mode subaddress is to be illegalized.
	legal	LEGAL	If the selected command type is to be treated as legal.
		ILLEGAL	If the selected command type is to be treated as illegal.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.6 **Set_RT_Status (int status)**

This routine permits the user to set bits within the 1553 status word returned by the RT. See Set_Status_Clear for the scope of this routine.

For MIL-STD-1553B applications (see Set_Protocol), the following bits may be set:

Bit	Name	Description
9	INS	Instrumentation Bit. This bit assets the Instrumentation bit of the MIL-STD-1553B status word. (Bit 10 of the Status Word).
8	SRQ	Service Request Bit. This bit asserts the Service Request bit of the MIL-STD-1553B status word. (Bit 11 of the Status Word).
7-4	reserved	Should be set to '0'.
3	BUSY	Busy Bit. Assertion of this bit is reflected in the outgoing MIL-STD-1553B status word. Assertion of this bit prevents memory accesses. (Bit 16 of the Status Word).
2	SSYF	Subsystem Flag Bit. This bit asserts the Subsystem Flag bit of the MIL-STD-1553B status word. (Bit 17 of the Status Word).
1	reserved	Should be set to '0'.
0	TF	Terminal Flag. Assertion of this bit is reflected in the outgoing MIL-STD-1553B status word. The EXC-1553PCMCIA automatically asserts this bit if a BIT failure occurs. Inhibit Terminal Flag mode code prevents the assertion by the host. Override Inhibit Terminal Flag Mode Code re-establishes the Terminal Flag option. (Bit 19 of the Status Word).

For MIL-STD-1553A applications (see Set_Protocol) bits 0 through 9 of this word are used to set bits 0 through 9 (bit times 19 through 10) of the status word.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.7 Set_Status_Clear (int flag)

This routine determines the scope of a call to Set_RT_Status. Calls to Set_RT_Status may be effective only for the next status word sent out or until a further call to Set_RT_Status changes the status.

input:	IMMEDIATE	causes calls to Set RT_Status to effect only a single command to the RT
	STATIC	causes calls to Set_RT_Status to effect all commands to the RT until a subsequent Set_RT_Status call is made.

3.8 Enable_Subaddr_Int(unsigned int subaddr, int txorrx, int msgtype, int inttype)

This routine must be called in addition to Set_Interrupt_Mask to cause an interrupt to occur when a specific subaddress or mode code receives a command. An interrupt may be requested either for all commands to a particular subaddress or for Broadcast commands to a particular subaddress. Receive commands and Transmit commands to a subaddress are enabled separately.

inputs:	subaddr	the subaddress to enable for non mode code commands, or the mode code number for mode commands.
	txorrx	TRANSMIT to request an interrupt on Transmit commands to the subaddress, RECEIVE for Receive commands.
	msgtype	MODECODE for mode code commands, or STANDARD for non mode code commands.
	inttype	BROADCAST to interrupt on Broadcast commands of the desired attributes, or ACCESS for any command of the desired attributes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.9 Disable_Subaddr_Int(unsigned int subaddr, int txorrx, int msgtype, int inttype)

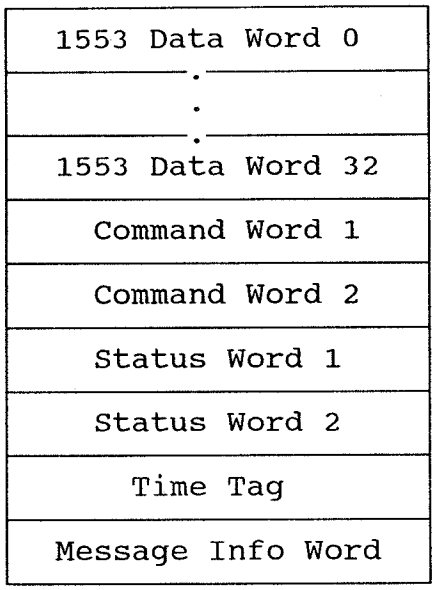
This routine may be called to disable an interrupt requested earlier via a call to Enable_Subaddr_Int.

inputs:	subaddr	the subaddress to disable interrupts for non mode code commands, or the mode code number for mode commands.
	txorrx	TRANSMIT to disable an interrupt on Transmit commands to the subaddress, RECEIVE for Receive commands.
	msgtype	MODECODE for mode code commands, or STANDARD for non mode code commands.
	inttype	BROADCAST to disable interrupts on Broadcast commands of the desired attributes, or ACCESS for any command of the desired attributes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*** Bus Monitor Functions ***

The Bus Monitor operation is based on the following data structure.



'C' language structure

```
struct BM_MSG
 {
  unsigned int data[32];
  unsigned int command1;
  unsigned int command2;
  unsigned int status1;
  unsigned int status2;
  unsigned int timetag;
  unsigned int msginfo;
 }
```

1553 Data Words

The data structure always allows space for 32 data words regardless of the number of data words actually associated with the message. The actual number of words may be obtained from the Command word or the Message Information Word.

Command Words

The next two locations in the BM_MSG structure are for command words. In non-RT-RT 1553 messages, only the first command word will be stored. However, in an RT-RT transfer, the first command word is the Receive Command and the second command word is the Transmit Command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Status Words

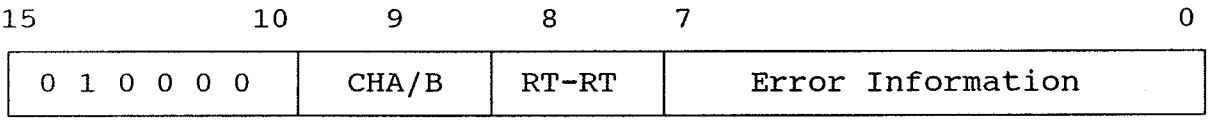
The next two locations in the Monitor Block are for status words. As the RT responds to the BC's command, the corresponding status word will be stored in Status Word 1. In an RT-RT transfer, the status of the Transmitting RT will be placed in status1, while the status of the Receiving RT will be placed in status2.

Time-Tag

The next location in the Monitor Block is the time-tag associated with the message. The time-tag is stored into this location at the end of message processing (i.e., captured after the command is validated). The time-tag's resolution is 64 microseconds per bit.

Message Information Word

The last memory location of each Monitor Block contains the message information word. Each information word contains the bus definition, RT-RT Transfer, and the error information.



Bit Number	Description
15-10	These bits will always be set to 010000.
9	Channel A/B. This bit defines on which of the two buses the command was received. (Logic 1= Bus A, Logic 0= Bus B).
8	RT-RT Transfer. This bit defines whether or not the message associated with this Monitor Block was an RT-RT transfer and whether the EXC-1553PCMCIA saved the second command word. This bit will be set only if the EXC-1553PCMCIA is instructed to monitor the Receive RT.
7-0	Error Information. These bits define the conditions of the message received by the EXC-1553PCMCIA for that particular Monitor Block. Each of the message information bits is defined in the following section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Error Information Bits

Error information bits are provided as a means to supply more data on the message. In an RT-RT transfer, the information applies to both of the status words. Each error information bit is defined below.

Bit Number	Description
7	Message Error. This bit will be set if the monitor detects an error in either the command word, data words, or the RT's status.
6	Mode Code without Data. This bit will be set if the monitor detects that the command being processed is a mode code without data words.
5	Broadcast. This bit will be set if the monitor detects that the command being processed is a broadcast message.
4	Reserved.
3	Time-out Error. This bit will be set if the RT did not receive the proper number of data words, e.g., the number of data words received was less than the word count specified in the command word.
2	Overrun Error. This bit will be set if the RT received a word when none was expected or the number of data words received was greater than expected.
1	Parity Error. This bit will be set if a parity error has occurred on the data words or the RT's status word.
0	Manchester Error. This bit will be set if a Manchester error has occurred on either the data words or the RT's status word.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.1 Run_BM (void)

This routine starts execution of the card causing commands on the bus to be recorded in the cards memory.

returns:	0	For success
	emode	If Set_Mode was not called with BM_MODE or BM_RT_MODE as its argument prior to calling this routine.

4.2 Read_Next_Message (struct BM_MSG *msg)

This routine returns a single message from the card's memory. Messages are returned in the sequence they were read by the card. The card has the capacity for holding up to 750 messages in a circular buffer (depending on the value of maxmem in Init_Socket), allowing the user considerable leeway in terms of getting the data out before it is overwritten.

input:	msg	A pointer to a structure of type BM_MSG defined in the introduction to this section. The user must allocate space for a structure of type BM_MSG and then supply a pointer to that structure to this routine. The structure will then be filled in by this routine.

returns:	0	For success
	emode	If Set_Mode not called with BM_MODE or BM_RT_MODE as its argument prior to calling this routine.
	enomsg	If there are no unread messages on the card.

4.3 Clear_Monitor_RT (int rtid)

This routine can be used to cause the Monitor to ignore all messages to the specified subaddress.

input:	rtid	the subaddress to ignore.

output:	0	for success
	emode	if not in Monitor mode
	ebadid	for rtid > 31

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.4 Monitor_RT (int rtid)

This routine can be used to cause the Monitor to commence monitoring subaddresses disabled via Clear_Monitor_RT.

input:	rtid	the subaddress to ignore.
output:	0	for success
	emode	if not in Monitor mode
	ebadid	for rtid > 31

4.5 Monitor_All_RT (void)

This routine can be used to cause the Monitor to commence monitoring all subaddresses disabled via multiple calls to Clear_Monitor_RT.

output:	0	for success
	emode	if not in Monitor mode

4.6 Get_Last_Cmd (void)

This routine returns the last Command word that was monitored on the bus.

output:	emode	if not in Monitor mode
	last command received, otherwise	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Mode defines */
#define MSEL         0x0300   /* status word bits for mode selection */
#define BC_MODE      0x0000
#define RT_MODE      0x0100
#define BM_MODE      0x0200
#define BM_RT_MODE   0x0300

/* RT mode defines */
#define DESCRIPTORSIZE   4
#define TRANSMIT         1
#define RECEIVE          0
#define LEGAL            1
#define ILLEGAL          0
#define TRANSMIT_SECTION 64
#define MODECODE         32
#define NOMODECODE       0
#define STANDARD         0
#define BRDCAST          1
#define MODE             2
#define BROADCAST_MODE   3

/* BC defines       */
#define MAXCMDS       1024    /* maximum commands to fit in a 16k bank   */
#define MAXDATABLKS   750     /* rounded maximum data blocks in 3 banks  */
#define BLOCKSINBANK  256     /* number of 32 word data blocks in a bank */
#define BC_CMD_BANK   3       /* bank 3 used for commands                */
#define DATABLKSIZE   32      /* 32 words per data block                 */
#define CMDENTRYSIZE  8       /* 8 words per command entry               */
#define CMDOFFSET     0x6000  /* command block start in bank 3           */

/* RT define */
#define STATIC    0
#define IMMEDIATE 1

/* Control word opcode defines */
#define END_OF_LIST  0x0000 /* opcode for stop transmitting */
#define SENDMSG      0x4000 /* opcode for sending a message */
#define GOTO_MSG     0x2000 /* for infinite loops           */

/* Control word defines */
#define CHANNEL_A    0x0200 /* send message on channel A */
#define CHANNEL_B    0x0000 /* send message on channel B */
#define RT2RT        0x0100 /* message is RT to RT       */
#define RT_ERROR     0x0001 /* error in RT response      */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* Control Register */
#define START_EXECUTION 0x8000  /* STEX bit in Control register */
#define RESET           0x4000  /* Reset Summit                 */
#define CHAEN           0x1000  /* Channel A enable             */
#define CHBEN           0x0800  /* Channel B enable             */
#define BROADCAST       0x0010  /* set to make RT31 Broadcast   */
#define NOBROADCAST     0x0000  /* RT 31 is normal RT           */

/* Set Interrupt Mask */
#define MERR            0x0800  /* message error (bit, word, parity) */
#define SUBAD           0x0400  /* subaddress accessed          */
#define ACCESS          0x0001  /* interrupt on any access (must be
                                different from value of BROADCAST*/

/* Protocol */
#define MIL_STD_1553A           1
#define MIL_STD_1553B           2

/* Misc */
#define ANYSOCKET 100
#define NO_INTERRUPTS 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*** Appendix B ***

This is an example of how to use the driver functions to access the EXC-1553PCMCIA card in a DOS programming environment.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "summit.h"
#include "proto.h"

struct BM_MSG msgbuf;
struct BM_MSG *msgptr = &msgbuf;
char message[80];
/*
* This program sets up the Bus Monitor of the EXC-1553PCMCIA and writes
* messages received to the screen.
*/


int main (void)
{
unsigned int i;
int error=0;
int socket;
int maxmemsize;
printf("Exc-1553PCMCIA Bus Monitor sample program rev 1.0\n\n");
while (error == 0)
  {    /*
       *  initialize card. Init_Socket will check each slot until it finds
       *  the EXC-1553PCMCIA
       */
        socket = ANYSOCKET;
        maxmemsize = 64;
        error = Init_Socket(&socket, &maxmemsize, NO_INTERRUPTS);
        if (error != 0)
       {
        printf("Init_Socket FAILURE\n");
        Print_Error( error );
        exit (EXIT_FAILURE);
       }
        printf("socket = %u,  memsize = %u\n",socket,maxmemsize);
        /* set up as Bus Monitor */
        error = Set_Mode(BM_MODE);
        if (error != 0)
       {
        printf("Set_Mode FAILURE\n");
        Print_Error( error );
        break;
       }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        /* Start card runing */
        error = Run_BM();
        if (error != 0)
       {
        printf("Run_BM FAILURE\n");
        Print_Error( error );
        break;
       }
        i = 0;
        while(1)
       {
        if (kbhit())
           break;
        /*
         *  read next monitored message. The msgptr structure also
         *  contains the data associated with the message though we
         *  don't print it here.
         */
        error = Read_Next_Message(msgptr) ;
        if (error == 0)
          {
           printf("%5u cmd1 %4x  stat1 = %4x  ttag = %4x   msginfo=%x\n",++i,
           msgptr->command1,msgptr->status1,msgptr->timetag,msgptr->msginfo);
          }
       }
       error = 1;
     }
  /* Release Card resources */
  error = Release_Card();
  if (error != 0)
    {
     printf("Release_Card FAILURE\n");
     Print_Error( error );
    }

return (0);
} /* end main */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

*** Appendix C ***

This is an example of how to use the driver functions to access the EXC-1553PCMCIA card in a Windows programming environment.

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <conio.h>
#include "summit.h"
#include "proto.h"
#include "error.h"


#define Print_n_Exit( title, error )                          \
{                                                             \
  if (error != 0)                                             \
      {                                                       \
      strcpy(Ctitlebuf, title);                               \
      MessageBox(hwnd, lstrcat(Ctitlebuf, Get_Error_Msg (error)), \
                 szAppName, MB_OK | MB_ICONSTOP);             \
      Release_Card();                                         \
      exit(0);                                                \
      }                                                       \
}

#define ID_TIMER 1

struct BM_MSG msgbuf;
struct BM_MSG *msgptr = &msgbuf;
char Ctitlebuf[120];
int i = 0;
static char szAppName[] = "PCMCIA Bus Monitor" ;


long FAR PASCAL _export WndProc (HWND, UINT, UINT, LONG) ;

int PASCAL WinMain (HANDLE hInstance, HANDLE hPrevInstance,
              LPSTR lpszCmdParam, int nCmdShow)
  {
  HWND         hwnd ;
  MSG          msg ;
  WNDCLASS     wndclass ;
  int          error, slot, memsize;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
if (!hPrevInstance)
             {
             wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
             wndclass.lpfnWndProc   = WndProc ;
             wndclass.cbClsExtra    = 0 ;
             wndclass.cbWndExtra    = 0 ;
             wndclass.hInstance     = hInstance ;
             wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
             wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
             wndclass.hbrBackground = GetStockObject (WHITE_BRUSH) ;
             wndclass.lpszMenuName  = NULL ;
             wndclass.lpszClassName = szAppName ;

             RegisterClass (&wndclass) ;
             }

hwnd = CreateWindow (szAppName,      // window class name
"PCMCIA Monitor Display",            // window caption
WS_OVERLAPPEDWINDOW,                 // window style
CW_USEDEFAULT,                       // initial x position
CW_USEDEFAULT,                       // initial y position
CW_USEDEFAULT,                       // initial x size
CW_USEDEFAULT,                       // initial y size
NULL,                                // parent window handle
NULL,                                // window menu handle
hInstance,                           // program instance handle
NULL) ;                              // creation parameters
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*************************************************************************
/ Code for PCMCIA
/*************************************************************************/


  MessageBox(hwnd,  "Bus  Monitor  Test  Program",  szAppName,  MB_OK  ¦
MB_ICONINFORMATION);
  /* Initialize the board */
  slot = ANYSOCKET;
  memsize = 64;
  error = Init_Socket(&slot, &memsize, NO_INTERRUPTS);
  Print_n_Exit( "", error );

  /* set up as Bus Monitor */
  error = Set_Mode(BM_MODE);
  Print_n_Exit( "Set_Mode FAILURE:  ", error );

  error = Clear_Monitor_RT(2);
  error = Monitor_All_RT();
  error = Get_BIT();
  error = Get_Pending();

  /* start the board running */
  error = Run_BM();
  Print_n_Exit( "Run_BM FAILURE:  ", error );

  if (!SetTimer (hwnd, ID_TIMER, 5, NULL))
    {
      if (IDCANCEL == MessageBox (hwnd, "Too many clocks or timers!",
          szAppName, MB_ICONEXCLAMATION ¦ MB_ICONEXCLAMATION))
      {
      Release_Card();
      return FALSE;
      }
    }

  ShowWindow (hwnd, nCmdShow) ;
  UpdateWindow (hwnd);

  while (GetMessage (&msg, NULL, 0, 0))
      {
      TranslateMessage (&msg) ;
      DispatchMessage (&msg) ;
      }
  return msg.wParam ;
  }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
long FAR PASCAL _export WndProc (HWND hwnd, UINT message, UINT wParam, LONG
lParam)
     {
                char Cmessage[100];
                int error;
                HDC hdc;
                PAINTSTRUCT ps;
                RECT rect;

                switch (message)
                    {
                    case WM_PAINT:
                    hdc = BeginPaint (hwnd, &ps);
                    sprintf(Cmessage,"%5u: cmd1 = %4x,  stat1 = %x,
                            ttag = %4x,  msginfo = %4x", i++,
                            msgptr->command1, msgptr->status1,
                            msgptr->timetag, msgptr->msginfo);
                    GetClientRect (hwnd, &rect);
                    SelectObject (hdc, GetStockObject(SYSTEM_FIXED_FONT) );
                    DrawText (hdc, Cmessage, -1, &rect,
                              DT_SINGLELINE ¦ DT_CENTER ¦ DT_VCENTER) ;
                    EndPaint (hwnd, &ps);
                    return 0;

                    case WM_TIMER:
                    error = Read_Next_Message(msgptr);
                    if (error != 0)
                    {
                    if (error != enomsg)
                       {
                       MessageBox(hwnd,"Read_Next_Message FAILURE ",
                                  szAppName, MB_OK ¦ MB_ICONSTOP);
                       MessageBox(hwnd, Get_Error_Msg (error), szAppName,
                                  MB_OK ¦ MB_ICONSTOP); \
                       DestroyWindow(hwnd);
                       }
                    }
                    else
                    InvalidateRect (hwnd, NULL, FALSE);
                    return 0;

                    case WM_DESTROY:
                    KillTimer (hwnd, ID_TIMER);
                    Release_Card();
                    PostQuitMessage (0);
                    return 0;
                    }

                return DefWindowProc (hwnd, message, wParam, lParam) ;
     }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The information contained in this document is believed to be accurate. However, no responsibility is assumed by Excalibur Systems, Inc. for its use and no license or rights are granted by implication or otherwise in connection therewith. Specifications are subject to change without notice.

Nov 1995 Rev. B-2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP