NOTICE

BMC Communications Corp. reserves the right to change the product described in this document as well as the document itself at any time and without notice.

DISCLAIMER

BMC COMMUNICATIONS CORP. MAKES NO WARRANTIES, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS DOCUMENT OR WITH RESPECT TO THE PRODUCT DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT SHALL BMC COMMUNICATIONS CORP. BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE PRODUCT.

Rev. 3.8 May 2, 2013

TABLE OF CONTENTS

I - INTRODUCTION

DESCRIPTION

The ARTIC board is an ARINC interface board for the PCI, CompactPCI, PMC and PC-1 04.

FEATURES

- **Up to Eight Transmitter Channels.**
- **Up to Eight Receiver Channels.**
- **Two Transmitting Modes (Normal, Loop).**
- **Three Receiving Modes (Normal, Continuous, and Label).**
- **Programmable ARINC Parameters.**
- **Programmable Inter-Message Time Delay.**

SPECIFICATIONS

With the ARTIC boards, the following ARINC parameters are programmable:

ARINC PARAMETERS

1. **Data Rate: Programmable from 1 Mbut to 100 Hz.**
2. **Fixed Baud rate: High (100 Kb) or Low (12.5 Kb).**
3. **No Parity or Parity enable.**
4. **Parity: Odd or Even.**

DIMENSIONS (will vary based on card format ordered)

- **PCI - (6.5 in. x 4.5 in.)**
- **Compact PCI (6.3 in. x 3.9 in.)**
- **PC-I 04 – 16 bit bus (3.56 in. x 3.78 in.)**
- **PMC - (6 in. x 4 in.)**

POWER REQUIREMENTS

- **+ 5 Volts @ 200 mAmp Max**
- **+ 12 Volts 20 mA Max**
- **- I2 VoIts 2OmAMax**

SOFTWARE

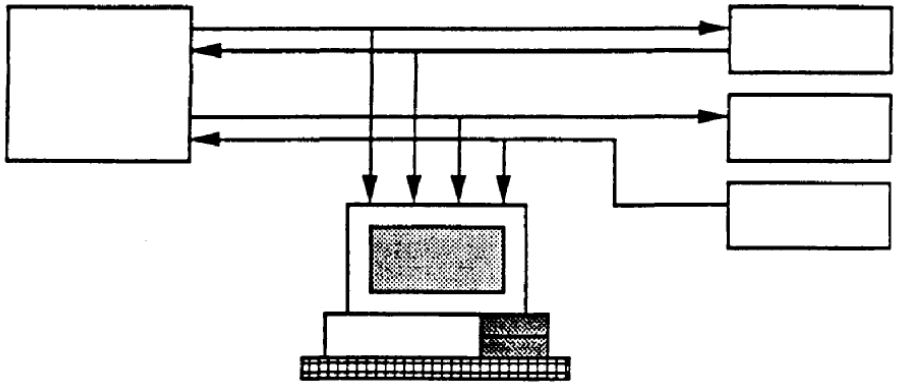
An Application Program Interface (API) library is included. This interface consists of a library of high-level language functions that allow the user to write custom applications programs without having to understand the "nuts and bolts" of the ARTIC system.
BMC Communications Corp. also offers application programs woth source code and DLL fow windows

APPLICATIONS

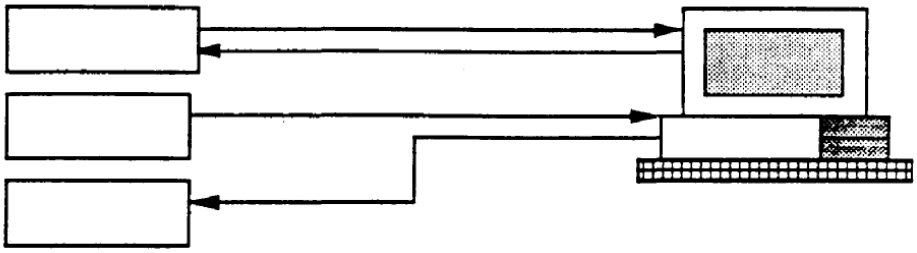
The ARTIC system is typically configured for use in testing and simulations applications. Uses of this system include:

- **Bus Analyzer**
- **Gererator I Analyzer**
- **Data Collection and Analysis**
- **Data Generator**

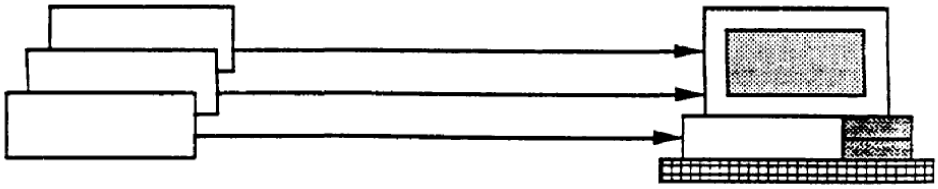
These applications are shown in Figure #1 and are described in the following paragraphs.



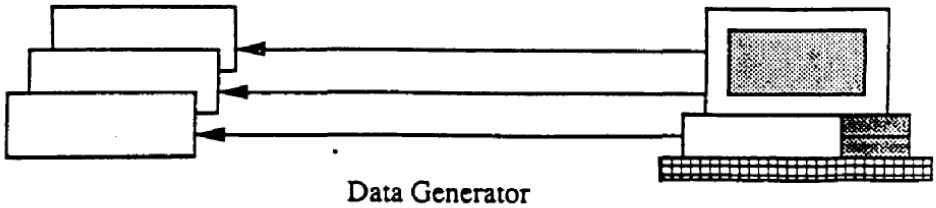
Bus Analyzer - single and bi-directional links



Generator/Analyzer - single and bi-directional links



Data Collection/Analysis



Data Generator

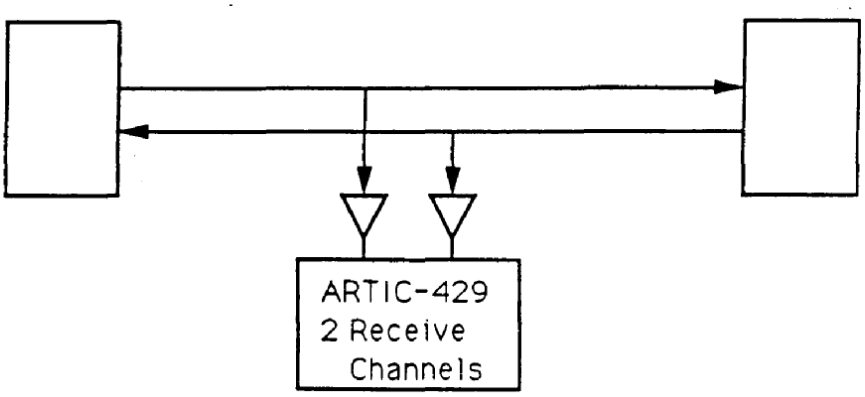
Figure 1 - ARTIC-429 Applications Setups

BUS ANALYZER

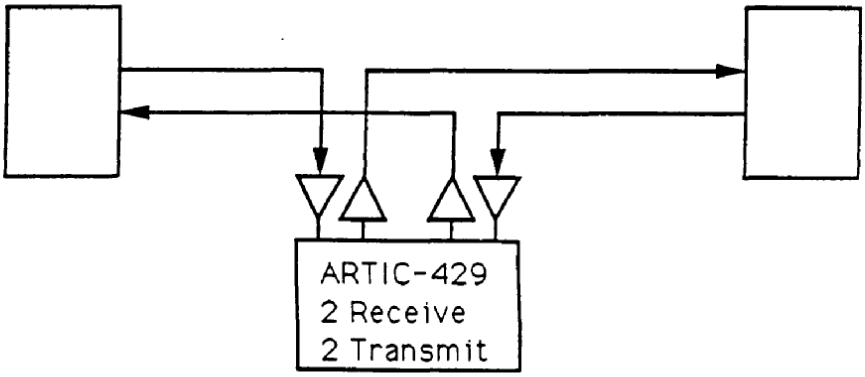
The ARTIC system can be used as either an active or passive bus analyzer. When configured as a Bus Analyzer, the ARTIC system monitors a single or bi-directional ARINC link and can perform several analysis tasks on the received data.

Different levels of analysis can be performed. For example, the received data can be displayed in a raw format, or interpreted and correlated with previously received data. The results of the latter being displayed in a higher-level format, performing statistics, etc.

Using the ARTIC system as an active bus analyzer affects the real-time behavior of the link and is therefore used only in special applications, as stated above. The difference between an active and a passive analyzer is shown in Figure #2.



Passive (non-interfering) Bus Monitor/Analyzer



Active (interfering) Bus Monitor/Analyzer

Figure 2 - Bus Analyzer Configurations

GENERATOR/ANALYZER

In the generator / analyzer configuration, the ARTIC system is programmed to transmit data to an avionics device and then analyze the output data of that device. This configuration is shown in Figure #1(2nd illustration).

DATA COLLECTION and ANALYSIS

When configured for data collection and analysis, the ARINC system receives data from the avionics device being tested. During a test flight, for example, aircraft position data for the entire trajectory, provided by a GPS receiver, can be collected and logged. This data can then be stored on disk or sent to a printer.

DATA GENERATOR

The ARTIC system can also be configured as a data generator to test avionics devices, as shown in Figure #1(4th illustration).

2-GETTING STARTED

SYSTEM CONTENTS

Thank you for purchasing BMC Communications Corp.'s ARINC 429/575/561/419/etc. Transmit / Receive Interface Card (ARTIC).
Your basic ARTIC system includes the following:

- **1 ARTIC Board.**
- **1 ARTIC API Library and Example Programs.**

This disk contains standard ARTIC routines that can be incorporated into user-written applications programs. Various sample applications programs are included.

- **1 Set of User Documentation.**
- **User's Guide and AP Library Reference.**
- **DLL for Windows**

HARDWARE and SOFTWARE REQUIREMENTS

In order to use your PCI ARTIC Board, you will need PC operating systems like WINDOWS. (Drivers for non-DOS operating systems are available, contact your BMC Communications Corp. representative for details.)

INSTALLATION

Installing the ARTIC board is as easy as installing any other PC-Bus board; such as an internal modem, display adapter, or expanded memory board.
The user plug a board in a computer and when windows enquire about installation programs it brouse it from the disc under DISTRIBUTION\DLL INSTLATION sub directory.
We strongly recommend, however, that all users read the installation instructions carefully to avoid damaging the board or the computer. After carefully removing the ARTIC board from its package, perform the following steps in the order indicated.

Copy WINDOWS DLL:

ARTIC_PCI.DLL to WINDOWS\SYSTEM32
PLXAPI522.DLL to WINDOWS\SYSTEM32
PCIDRVAPI.DLL to WINDOWS\SYSTEM32
PCI9030.SYS to WINDOWS\SYSTEM32\DRIVERS

Board Address Selection

The PC automatically allocated memory space for the PCI board. The programs from the distribution disk into the following subdirectory:

PC104 BOARD:

Interrupt Request Level Selection

The next step in the installation is selecting the IRQ (Interrupt Request Level — Jumper Block JP5 on the ARTIC board). The appropriate IRQ setting is determined by the software used to drive the ARTIC Board.

If you use any of BMC Communications Corp.'s Applications Software packages, you can select any IRQ (3 to 7). We recommend, however, using the factory-set IRQ, because it matches the default value in the software.

If you write your own application software, you must follow the guidelines provided in the Programming Reference Manual. The ARITC software API library supports both interrupt-driven and polled (no interrupt) operation. If you choose not to use interrupts, select the NO IRQ SELECTED jumper configuration. See Figure 3 for jumper settings.

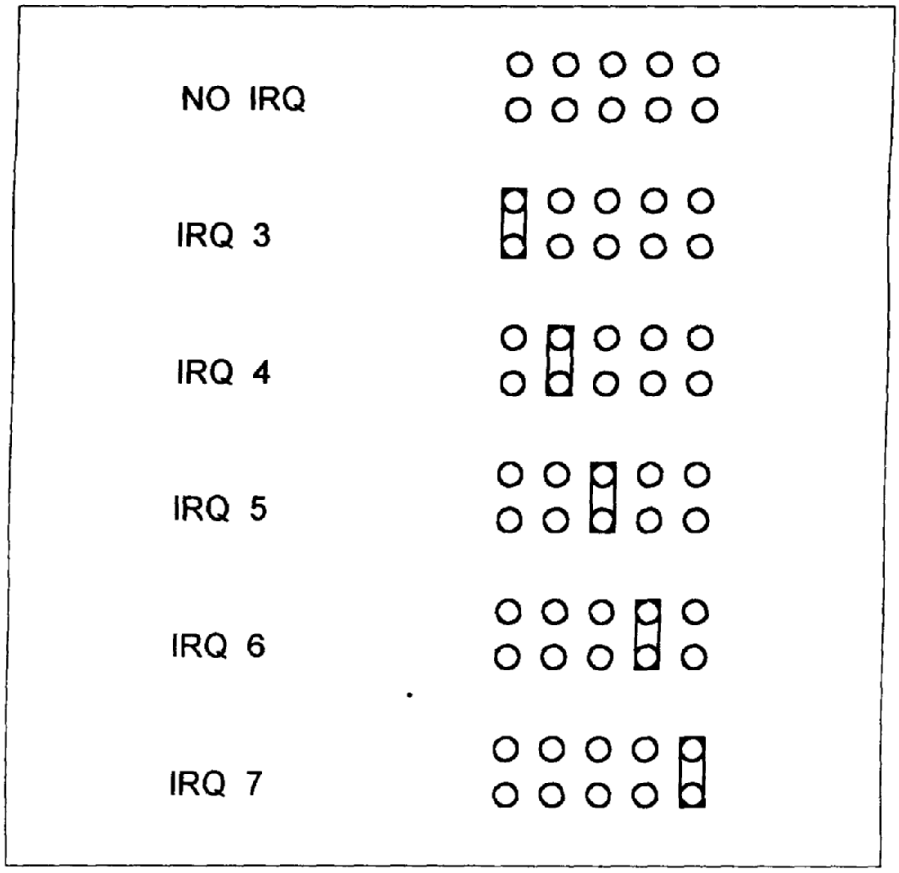


Figure 3. Interrupt Request Level Jumper Setting.

Thank you again for purchasing the ARTIC system. We at BMC Communications Corp. are looking forward to supporting you in any way we can in achieving your ARINC testing and/or development goals.

THEORY of OPERATION

INTRODUCTION

The configuration shown has two receive and two transmit channels. The board has a 16-bit microprocessor that controls all of the board's functions.

COMMUNICATIONS BETWEEN BOARD AND HOST

There are two major ways in which the board can communicate with the host computer. The first is through the normal avenue of shared or common memory; the second is through the use of interrupts. We will now outline each of these methods.

Dual Port Memory

The principal link between the ARTIC board and the host computer is the Dual-Port RAM, which is a memory area 4 Kbytes (PC104) to 32K (PCI, CPCI, VME) in size that is shared by both the ARTIC processor and the host computer processor. This memory is divided into several major sections most of which are further subdivided into individual fields. Each of these fields is dedicated to setting one particular parameter governing one aspect of the board's operation. The ARTIC processor constantly scans the shared memory, waiting for an instruction from the host computer. When such an instruction is found, the ARTIC board attempts to execute it; and then posts a completion result in the Dual-Port RAM to be read by the host processor.

Before giving a ARTIC instruction, the host processor must also enter relevant configuration information in certain fields of the Dual-Port RAM to be read by the ARTIC processor. The ARTIC processor, likewise, communicates back to the host through other dedicated fields, and through result codes; also written to shared memory. These can then picked up by the host processor. This shared memory concept is the method by which the host is able to access information from data transactions occurring on the ARINC data bus, and also enables the host to stay informed about the progress of transmission and board operation, in general. In summary, certain fields in the shared memory are written by the host and read by the board; others are written by the board and read by the host.

Interrupts

In addition to the normal channels of communication through shared memory, the ARTIC processor is capable of sending an IRQ (Interrupt Request) signal to the host computer when certain specified events occur. This signal will interrupt the host processor from whatever task it is doing, and will cause it to branch to a special routine written by the user to handle these events. The host software specifies to the ARTIC processor, by writing the appropriate control information in the shared memory, on which conditions (if any) the ARTIC processor should send an Interrupt Request.

DUAL-PORT MEMORY ORGANIZATION

The Dual-Port Memory is divided into three main sections: the Data Area, the Global System Registers, and the Control Blocks (eight in number, one for each channel). The exact byte-by-byte organization of the Dual-Port Memory can be understood by looking at the structure definitions in the API header file UADI32K.H. A memory map of the DPM is also included in this manual. Please note, however, that is NOT necessary to know exactly where each byte is situated in order to understand the board's operation.

Data Area

The 3840 first bytes (PC104) or 32K (PCI,CPCI,VME) of the Dual Port RAM (address range 0 to xEFF) are used as a general purpose data area. The user is free to allocate Transmit and Receive buffers and locate them at their discretion. This area is usually used for storing the actual contents of messages, rather than ARTIC operational instructions.

Global System Registers (GSR)

The next section of memory in the dual port RAM after the end of the Data Area is the Global Systems Register. This area is used to store general ARTIC configuration information, and to record the progress of the current operation or procedure. The GSR's structure consists of the following:

1. System ID (8 bytes):

The contents of this of this structure is an ASCII string of the form: "ARTICm.n", where m and n are the major and minor version numbers of the board's firmware which should be identical to the ARTIC API library version number.

This string is extremely .useful as a diagnostic check of correct board operation and is used for that purpose by a number of API functions, If one can properly read this string, then proper board installation and initialization are confirmed.

2. Channel Configuration (1 WORD):

This byte shows the available channels on the board. See Appendix F for bit structure diagram.

3. Channel Active (1 WORD):

This byte shows the active or in use channels on the board. See Appendix F fior bit structure diagram.

4. STATUS_TX_RX Transmit/Receive Indicator (1 word):

This field is set by the board and read by the host. The Following fields are used in this data structure:

A. New Data Received (4 bits, one for each Receive Channel)

B. Receive Buffer Full (4 bits, one for each Receive Channel)

C. Receive Buffer Half-full (4 bits, one for each Receive Channel)

D. Transmit Complete (4 bits, one for each Transmit Channel)

A bit in this word is set by the ARTIC system every time the condition represented by this bit is true. This word is polled by the host in order to identify any relevant events, e.g. receive buffer full, and then cleared by the host to allow new events to

be identified next time. In interrupt mode, the interrupt service routine (ISR) will read this word to identify the event that triggered the interrupt. In polled mode, the application program will read this word occasionally as dictated by the programs flow to check if any relevant event has happened.

The first bit of this field is a transmit flag which is set by the board each time any transmit operation has occurred. The second bit is a receive bit which is likewise set every time a receive operation has occurred. They indicate whether the boards transceivers are transmitting or receiving actual signals to or from the ARINC data bus. Keep in mind that the user must clear these bits every time he reads them or they will remain high (binary '1') forever after the first time they are set by the board. The user will thus be receiving outdated information. Each channel is represented by a pair of bits.

.

5. INTERR_TX_RX Interrupt Selector Register (1 word):
In general, if a user needs to know whether the board has transmitted or received, he can poll the transmit and/or receive flag in the previous field at regular intervals. However, there may be times when the user needs to know immediately should such an event happen. This field allows the user to set the board so that if a transmission or reception occurs it will immediately trigger an interrupt in the host PC.
This word is bit-by-bit identical to the Transmit/Receive Indicator Field above. However, this word is written by the host (PC). By setting any of the bits to a '1' the host may request an interrupt every time the condition represented by that bit occurs. The interrupt service routine (ISR) can then identify the source of the interrupt by reading the above described Transmit/Receive Indicator Field.
See Appendix F for bit structure diagram.

6. PARAM 1-4 channels (1 word each):
This word is used to display and change the active configuration of the ARINC transceivers. The ARINC IC Control Word for PARAM....1,2 corresponds to the configuration ofeach transmit channel 1, 2.
This control word can be used to change the following parameters:
A. Transmit Data Rate
B. Parity

7. TX COUNT (4 words):
Each word corresponds to the transmit counter on a particular channel.
8. RX_COUNT (4 words):
Each word corresponds to the receive counter on a particular channel.

CHANNEL CONTROL BLOCKS:

Transmit Channel Control Block (TCCB)

Each of the four Transmit Channel Control Blocks consists of the following fields:
1. Command (1 byte): the type of operation requested. The value of this field can be as follows:
A. ARTIC_CMD_IDLE - No command requested, idle state
B. ARTIC_TX_NORMAL - Transmit Command: NORMAL mode
C. ARTIC_TX_LOOP - Transmit Command: LOOP mode
D. ARTIC_TX_STOP - Stop transmission

2. Data Start Address (1 word): The offset address (relative to the board address) of the first ARINC word to be transmitted (if applicable).

3. Data Count (1 word): The number of 32-bit ARINC words, starting at the above Start Address, to be transmitted (if applicable).
4. Execution Delay (1 word): Time period in increments of 2 microseconds to elapse before execution of the specified command.
5. Command Status (1 word): All the above fields are set by the host processor. This field is written by the ARTIC processor after receiving a command, within the system's Command Response Time. The value of this field is a command completion status which can be ARTIC_SUCCESS or an error code. All error codes are explained in detail in the following paragraphs. By writing into this register, ARTIC acknowledges the command and releases the Transmit Channel Control Block for a possible next command.

Receive Channel Control Block (RCCB)

Each of the four Receive Channel Control Blocks consists of the foHowing fields:
1. Command (1 byte): the type of operation requested. The value of this field can be as follows:
A. ARTIC_CMD_IDLE - No command requested, idle state
B. ARTIC RX NORMAL - Receive Command: NORMAL mode
C. ARTIC_RXCONTNS - Receive Command: CONTINUOUS mode
D. ARTIC..RX_LABEL - Receive Command: LABEL mode
E. ARTIC_CMD_STOP - Stop receiving
2. Data Start Address (1 word): The address offset of the required receive buffer (if applicable).
3. Buffer Length (1 word): The size in 32-bit ARINC words of the requested receive buffer (if applicable).
4. Label Mask (31 word): This is a string of 256 bits, each representing one ARINC label value. Bit 0 of the first byte represents LABEL 0 and bit 7 of the 32nd byte represents LABEL 255. A bit value of zero means the corresponding label value should be masked i.e. ignored.
5. Command Status (1 word): All the above fields are set by the host processor. This field is wriften by the ARTIC processor after receiving a command, within the system's Command Response Time. The value of this field is a command completion status which can be ARTIC_SUCCESS or an error code. All error codes are explained in detail in the following paragraphs. By writing into this register, ARTIC acknowledges the command and releases the Receive Channel Control Block for a possible next command.

OPERATING MODES

Transmitting Modes

1. NORMAL MODE: The board transmits a data buffer (up to 960 ARINC words). Upon termination, a Status Flag is set and an optional Interrupt Request is posted.

2. LOOP MODE: The board transmits a given buffer (up to 9600ARINC Words). Upon termination, transmission is repeated until a STOP command is received. A Status Flag is set and an optional Interrupt Request is posted every time the last word in the buffer is transmitted.

2. REPEAT MODE: The board transmits a given buffer (up to 9600ARINC Words). Upon termination, transmission is repeated until a STOP command is received or at the end of a predefined transmission number of messages. Status Flag is set and an optional Interrupt Request is posted every time the last word in the buffer is transmitted.

Receiving Modes

In all receiving modes, a user-written 256-bit Label Mask is applied to the received data, enabling it to pre-filter non-relevant labels. Label-filtering is disabled by filling the mask with 1-s. In addition to the operations described below, each time an ARINC data word is received in any of the receiving modes, a Status Flag is set and an optional Interrupt Request is posted. The Last Received Label can be read at all times in all receive modes.

1. NORMAL MODE: Received ARINC data words are stored in a user specified buffer (up to 9600 ARINC words) until the buffer is full or the STOP command is received.
When the buffer is full, a Status Flag is set and an optional Interrupt Request is posted.

2. CONTINUOUS MODE: Received ARINC data words are stored in a user specified buffer.
When the buffer is full, the buffer "wraps around" acting as a circular buffer. Each time the buffer is full or half-full, a Status Flag is set and an optional Interrupt Request is posted. Receiving is terminated using the STOP command.

3. LABEL MODE: Received ARINC data words are stored in a user-specified 256-word long buffer. Each cell in the buffer corresponds to a Label value. Each time a data word is received on the Bus, the board stores it in the corresponding cell. Receiving is terminated using the STOP command.

TRANSMITTER OPERATION

Transmitting is started by sending a Transmit Command to the ARTIC system as follows: the Transmit Channel Control Block (TCCB) is loaded with the appropriate values, the Command code being last. The ARTIC system is awakened by the Command byte being written. The Command byte is then read and cleared back to 0. The Command code is then checked as follows:

ARTIC_TX_NORMAL or ARTIC_TX_LOOP:

If the channel is not busy, the data specified by the Start Address and Data Count fields are moved to the board's private memory (provided they are legal, see error handling below) and the Execution Delay and Transmission Intervals are read. The Status byte is then written with the value ARTIC_SUCCESS, and the TCCB is released (note that at this point the application can also start writing new data into the buffer since its contents have been transferred to the system's private memory). At this point the channel is marked as busy in the Channel Activity byte of the GSR and the channel Transmit Counter in the GSR is set to zero. If the Execution Delay is nonzero, a timer is started. When the timer expires (immediately if the value was zero), the first data word is output to the ARINC channel, followed by the rest of the data buffer contents. After each word is output the system checks whether "interrupt on word transmitted" is requested by reading the Interrupt Condition Word in the GSR. If so, an IRQ is generated. The system also sets the appropriate bit in the Transmit Receive Status Word and the channel's Transmit Data Counter is incremented.

ARTIC_TX_STOP command is received by the system at any time during transmission, the system will stop the transmission and mark the channel as not busy in the Channel Activity byte.

If the entire specified data was transmitted, the system will check again the Command code that started the transmission. If it was ARTIC_TX_LOOP, it will reset the channel's Transmit Counter to zero and restart the transmission at the first data word, not before the Execution Delay is introduced (if nonzero).

If the Command code was ARTIC_TX_NORMAL, the system will set the channel's "Transmit Complete" bit in the Interrupt Status Block. If this interrupt is enabled in the Interrupt Condition Block an IRQ will be generated at this point and the channel will be marked as "Not Busy" in the Channel Activity byte of the GSR.

RECEIVIER OPERATION

Receiving is started by sending a Receive Command to the ARTIC system as follows: the Receive Channel Control Block (RCCB) is loaded with the appropriate values, the Command code being last. The ARTIC system is awakened by the Command byte being written. The Command byte is then read and cleared back to 0. The Command code is then checked as follows:

ARTIC_RX_NORMAL or ARTIC_RX_CONTNS:

If the channel is not busy, the contents of the entire RCCB are moved to the board's private memory. The Status byte is then written with the value ARTIC_SUCCESS (provided Receive Buffer range is valid - see Error Handling Paragraph), and the RCCB is released. At this point the channel is marked as busy in the Channel Activity byte of the GSR and the channel Receive Counter in the GSR is set to zero. The receiver is then enabled and the system waits for data on the ARINC channel. When a data word is received, the bit corresponding to its label value in the Label Mask is checked. If its value is 0 (masked), the received data is discarded and no further action is taken. Otherwise (the label is enabled), the system stores the data in the next location in the receive buffer. It then checks whether "interrupt on data received" is requested by reading the Interrupt Condition Word in the GSR. If so, an IRQ is generated. The system also sets the channel's DATA RECEIVED bit in the Transmit/Receive Status Word. The same status bit I IRQ operation is performed if the Half-buffer Full condition is met, i.e. the flow of received data just passed the half-size mark of the receive buffer. The word's Label value is written into the channel's Last Received Label byte of the GSR and the channel's Receive Data Counter is incremented.

If an ARTIC_RX_STOP is received by the system at any time during receive, the system will stop receiving in this channel and mark the channel as not busy in the Channel Activity byte.

If a received data word was stored in the last location of the receive buffer, the system sets the BUFFER FULL bit in the Transmit/Receive Status Word and if the corresponding bit is set in the Interrupt Condition Word, the IRQ is generated. At this point, if the Command was ARTIC_RX_NORMAL, the receive operation is complete and the channel is marked as not busy in the Channel Activity byte in the GSR.

In the Continuous Mode (ARTIC_RX_CONTNS), the channel will continue accepting data and the buffer will "wrap around", i.e. if a data word is received after a BUFFER FULL situation, the data word will be stored at the beginning of the buffer and the channel's Receive Counter will be set to 1. The only way to stop receiving in this mode is via a ARTIC_RX_STOP command.

ARTIC_RX_LABEL:

If the Receive Command code was ARTIC_RX_LABEL, the system uses a fixed length 256x3 byte-long receive buffer (768 bytes). Each of the 256 3-byte cells in this buffer represents an ARINC Label. When a data word is received and it passes the Label Mask test, the system stores it in its corresponding place in the buffer, e.g. Label 0 in the 1st cell and so on. In addition, the next 31 word (256 bits) after the end of the buffer are used by the ARTIC processor to flag newly received labels. The only way to stop receiving in this mode is via a ARTIC_RX_STOP command.

4- APPLICATION PROGRAM INTERFACE

GENERAL

The supplied Application Programs Interface (API) is a collection of high-level language functions which activate all of the board's functions, so the application programmer is not required to have an intimate knowledge of the low-level description of the shared memory interface protocol.
However, the entire interface between the ARTIC board and the Host Computer is documented in this user's manual to allow for a deeper understanding of the system, if such is desired by the user. The entire source code for the API is included and can be used to learn and understand the board's operation.
The Applications Program Interface (API) Reference contains descriptions and syntax for standard ARTIC system routines that can be used by programmers to write custom application software for the ARTIC system. Correct use of the API requires understanding of software concepts and programming language.

PROGRAMMING LANGUAGE

The API was written in C It is therefore most convenient for applications written in MS-C. However, keep in mind that an application programmer uses the API, they do not write it. If you use any programming language, you can easily call the API functions using the mixed-language conventions. If you use a language other than Microsoft's, you can still interface to the API. In this case, however you will have less documentation available. Registered users will receive updates periodically including programming tips and programming examples.

COMPILER CONSIDERATIONS

Memory Model

You can use any of the memory models supported by your compiler

DEFINITIONS

Before using the Application Program Interface, it is necessary to become familiar with the following terms:

1. Channel Number

2. Label Mask

Channel Number:

A number from I to 8 specifying one of the four possible channels on the board. Whether it is a Receive or a Transmit channel must be understood by the context. For example the API function rx_count(channel) returns the number of received data words in the specified Receiver Channel.

A function using a channel as a parameter will return the ARTIC_INV_CHANNEL error code if the specified channel is not configured on the board (for example channel 2 on a 1TIR board)

COMPILER CONSIDERATIONS

Memory Model

You can use any of the memory models supported by your compiler,

DEFINITIONS

Before using the Application Program Interface, it is necessary to become familiar with the following terms:

1. Channel Number

2. Label Mask

Channel Number:

A number from I to 8 specifying one of the four possible channels on the board. Whether it is a Receive or a Transmit channel must be understood by the context. For example the API function rx_count(channel) returns the number of received date words in the specified Receiver Channel.

A function using a channel as a parameter will return the ARTIC_INV_CHANNEL error code if the specified channel is not configured on the board (for example channel 2 on a 1T1R board)

Label Mask:

Label Masks have been described in the Theory of Operation chapter. For the API user, they are arrays of 31 word (256 bits) which you have to allocate in your program. Once a label mask is allocated, you can enable I disable any label using the convenient artic_IabeLxxx API functions. However, if you work in the straightforward mode in which all labels are enabled, you can use the API's pre-allocated "all-enabled" Label Mask _ARTIC_ALL_LABELS and pass it to artic_rx_cmd, the API function which sends Receive Commands to ARTIC.

API STRUCTURE

The ARTIC-429 Application Interface is a Collection of C-language functions that can be called from any language (programs written in languages other than C must follow the compiler's mixed language conventions). All function names start with "artic_". The API includes also several global variables whose names start all with _ARTIC_.

Global Variables

_ARTIC_BOARD: this is a variable of type ARTIC_429 far * This variable must be set by the API function artic_set_board before any other operation on the board. This function loads the selected board address into _ARTIC_BOARD and from then on all operations will be performed on this board. This variable was made a global variable in order to allow multi-board operation while not having to pass the board address to each and every API function.

API Functions

The API functions constitute the main body of the ARTIC—429 API library. A detailed description of each function can be found in the API FUNCTION REFERENCE MANUAL, which is probably the document you will use most while developing ARTIC applications.

Reset Control

The final section of the dual port memory is a one byte field called Reset Control located at address offset FF8 (hex). This field will generate a software reset of the CRAM Board that will, as a result, clear all fields in the board to '0'.

MEMORY MAP

It is extremely useful for anybody attempting to write his own API functions, or needing to diagnose a problem with a board that appears to be malfunctioning or simply wanting to understand on a deeper level the theory of operation of the board, to have a list of all of the ARTIC fields and their exact locations in hardware.

5- ARTIC BM MODE (BUS MONITOR)

This system is oriented to ARINC systems Bus Monitoring Applications. In order to maintain compatibility with the regular ARTIC system, the same API functions can be used. The only difference is the following:

1. When receiving in RXfiORMAL or RXCNTNS mode, the ARTIC-429/BM starts storing the collected data ALWAYS starting at offset 0000, regardless of the start address value passed in the Receive Command. The total receive buffer size is 3800 bytes.

2. The ARTIC-429 BM board stores the receive data in packets of 10 bytes for each received ARINC word:

A. The ARINC word (4 bytes)

B. Time Tag (4 bytes representing a long word in increments of 2 microseconds)

C. Channel (1 word the channel from which data was received)

3. The board maintains one common buffer (starting at 000, as described above) for ALL channels. Sending a Receive command to a channel enables the board to monitor the channel. The board will store sequentially all words received from all active channels. However, it still maintains a separate counter for each channel that can be read using the API function artic_rx_count(channel). The buffer will wrap around after the maximum number of 10 byte packets that can be stored in 3800 bytes has been received. You can still enable interrupt on buffer full and half-buffer full as in normal mode, and the same bits in the TransmitlReceive Status Word will be updated.

4. The board will honor the number of data words specified in a ARTIC_RX_NORMAL command, i.e. if it will stop monitoring a channel from which the specified data count has been already received, while other channels can still remain active.

The following library functions are related with bus monitor opeartion:

artic_ena_monitor artic_dis_monitor
artic_ena_monful I artic_dis_monful
artic_ena_monhaif artic_dis_monhalf
artic_mon_halfbuf_ful
artic_mon_buiffull
artic_bm_available
artic_bm_mode
artic_rx_mon_count

SYSTEM_ID = "ARTICX.X"

ARINC MEMORY CONTROL SEGMENT

X7E00	**GSR**	**X7E00**	**System_ID (8)**	**BYTE**
		X7E08	**Channe_cfg**	**WORD**
ARINC		**X7E0A**	**STATUS_TX_RX**	**WORD**
GENERAL		**X7E0C**	**INTERR_TX_RX**	**WORD**
REGISTERS		**X7E0E**	**PARAM_1**	**WORD**
		X7E10	**PARAM_2**	**WORD**
		X7E12	**PARAM_3**	**WORD**
		X7E14	**PARAM_4**	**WORD**
		X7E16	**TX_COUNT[4]**	**WORD**
		X7E1E	**RX_COUNT[4]**	**WORD**
		X7E26	**RESERVED [2]**	**WORD**
X7E28	**Tx_ctl [4]**	**X7E28**	**Command**	**WORD**
		X7E2A	**Start_Addr**	**WORD**
ARINC	**CHANNEL 0**	**X7E2C**	**Word_Count**	**WORD**
TRANSMIT		**X7E2E**	**Delay**	**WORD**
CONTROL		**X7E30**	**Status**	**INT.**
REGISTERS		**X7E32**	**Command**	**WORD**
		X7E34	**Start_Addr**	**WORD**
	CHANNEL1	**X7E36**	**Word_Count**	**WORD**
		X7E38	**Delay**	**WORD**
		X7E3A	**Status**	**INT.**
		X7E3C	**Command**	**WORD**
	CHANNEL2	**X7E3E**	**Start_Addr**	**WORD**
		X7E40	**Word_Count**	**WORD**
		X7E42	**Delay**	**WORD**
		X7E44	**Status**	**INT.**
		X7E46	**Command**	**WORD**
	CHANNEL3	**X7E48**	**Start_Addr**	**WORD**
		X7E4A	**Word_Count**	**WORD**
		X7E4C	**Delay**	**WORD**
		X7E4E	**Status**	**INT.**
X7E50	**Rx_ctl[4]**	**X7E50**	**Command**	**WORD**
		X7E52	**Start_Addr**	**WORD**
ARINC	**CHANNEL 0**	**X7E54**	**Buffer_Size**	**WORD**
RECEIVE		**X7E56**	**Label_Mask[16]**	**WORD**
CONTROL		**X7E76**	**Status**	**INT.**

REGISTERS		**X7E78**	**Command**	**WORD**
		X7E7A	**Start_Addr**	**WORD**
	CHANNEL 11	**X7E7C**	**Buffer_Size**	**WORD**
		X7E7E	**Label_Mask[16]**	**WORD**
		X7E8E	**Status**	**INT.**
		X7E90	**Command**	**WORD**
	CHANNEL 12	**X7E92**	**Start_Addr**	**WORD**
		X7E94	**Buffer_Size**	**WORD**
		X7E96	**Label_Mask[16]**	**WORD**
		X7EA6	**Status**	**INT.**
		X7EA8	**Command**	**WORD**
	CHANNEL 13	**X7EAA**	**Start_Addr**	**WORD**
		X7EAC	**Buffer_Size**	**WORD**
		X7EAE	**Label_Mask[16]**	**WORD**
		X7EBE	**Status**	**INT.**

X7EC0	**MONITOR_CTL**	**X7EC0**	**COUNTER**	**WORD**
ARINC BUS		**X7EC2**	**STATUS**	**WORD**
MONITOR		**X7EC4**	**INTERR_COND**	**WORD**
REGISTERS		**X7EC6**	**RESERVED**	**WORD**

ARINC 708

INTRODUCTION:

Arinc 453/708 is a protocol which transmit data frames of 1600 bits. The transmission rate is 1 Mbit. The data is preceeed by a three microsecond width. SYNC PULSE. The pulse has a transition in the middle. The data is a Manchester code structure; a transition in the mioddle of the one microsecond. If the transition is from low to high data is logic ZERO. If the transition is from high to low data is logic ONE.

ARINC 708 BUS
The ARINC 708 allows two types of data bus interface techniques:

1. **Direc Coupling**
2. **Transformer Coupling**

The purpose of the data bus coupler is to prevent a short in the main data bus. The ARTIC 708 board has two set of 4x2 headers for the two output buses. The headers with four jumpers are located close to the on board transformers.The user select between direct coupling connecting jumpers located in the corners (1-2 and 7-8). Transformer coupling is selected connecting the internal jumpers (3-4 and 5-6). The cable to be used must have chararcteristic impedance from 70 to 85 ohms.

Data Area
The 3840 first bytes (PC104) or 32K (PCI,CPCI,VME) of the Dual Port RAM) are used as a general purpose data area. The user is free to allocate Transmit and Receive buffers and locate them at their discretion. This area is usually used for storing the actual contents of messages, rather than ARTIC operational instructions.
One of the transmit control register define the number of words to be transmitted. ANY VALUE DIFFERENT THAN 100 WORDS (1600 BITS) CREATES AN ERROR.
The transmission data start from the LSB- Least Significant bit – BIT 0 to the MSB – Most Significant Bit-BIT 15. Due to the fact the Arinc 708 is bit oriented the user should be carefull with the bit loction in the frame.

Global System Registers (GSR)
The next section of memory in the dual port RAM after the end of the Data Area is the Global Systems Register. This area is used to store general ARTIC 708 configuration information, and to record the progress of the current operation or procedure. The GSR's structure consists of the following:

1-. Channel Configuration (1 WORD):
This words shows the active or in use channels on the board.

2. STATUS_TX_RX Transmit/Receive Indicator (1 word):
This field is set by the board and read by the host. The Following fields are used in this data structure:
A. Transmit Complete – bits 0
B- New Data Received - bit 1
C. Receive Buffer Full – BIT 2
D. Receive Buffer Half-full – BIT 3

A bit in this word is set by the ARTIC system every time the condition represented by this bit is true. This word is polled by the host in order to identify any relevant events, e.g. receive buffer full, and then cleared by the host to allow new events to be identified next time. In interrupt mode, the interrupt service routine (ISR) will read this word to identify the event that triggered the interrupt. In polled mode, the application program will read this word occasionally as dictated by the programs flow to check if any relevant event has happened.
The first bit of this field is a transmit flag which is set by the board each time any transmit operation has occurred. The second bit is a receive bit which is likewise set every time a receive operation has occurred. They indicate whether the boards transceivers are transmitting or receiving actual signals to or from the ARINC data bus. Keep in mind that the user must clear these bits every time he reads them or they will remain high (binary '1') forever after the first time they are set by the board. The user will thus be receiving outdated information. Each channel is represented by a pair of bits.

.
3. INTERR_TX_RX Interrupt Selector Register (1 word):
In general, if a user needs to know whether the board has transmitted or received, he can poll the transmit and/or receive flag in the previous field at regular intervals. However, there may be times when the user needs to know immediately should such an event happen. This field allows the user to set the board so that if a transmission or reception occurs it will immediately trigger an interrupt in the host PC.
This word is bit-by-bit identical to the Transmit/Receive Indicator Field above. However, this word is written by the host (PC). By setting any of the bits to a '1' the host may request an interrupt every time the condition represented by that bit

occurs. The interrupt service routine (ISR) can then identify the source of the interrupt by reading the above described Transmit/Receive Indicator Field. See Appendix F for bit structure diagram.

4. PARAM :

The first two bits; bits 0-1, defined the channel:
0 - channel A
1- channel B
2- reserved
3- TTL output

Bits 2-3 defined data transmition error injection
Bit 2 – SYNC error injection
Bit 3 – Data Manchester Code error injection.

5. TX COUNT:
Corresponds to the transmit messagecounter

6. RX_COUNT (4 words):
Corresponds to the receive message counter.

CHANNEL CONTROL BLOCKS:

Transmit Channel Control Block (TCCB)
Transmit Channel Control Blocks consists of the following fields:
1. Command (1 byte): the type of operation requested. The value of this field can be as follows:
A. ARTIC_CMD_IDLE - No command requested, idle state
B. ARTIC_TX_NORMAL - Transmit Command: NORMAL mode
C. ARTIC_TX_LOOP - Transmit Command: LOOP mode
C. ARTIC_TX_REPEAT - Transmit Command: REPEAT mode
D. ARTIC_TX_STOP - Stop transmission

2. Data Start Address (1 word): The offset address (relative to the board address) of the first ARINC word to be transmitted (if applicable).
3. Data Count (1 word): The number of words, starting at the above Start Address, to be transmitted (if applicable).
4. Execution Delay (1 word): Time period in increments of 1 MILISECOND to elapse before execution of the specified command.
5. Command Status (1 word): All the above fields are set by the host processor. This field is written by the ARTIC processor after receiving a command, within the system's Command Response Time. The value of this field is a command completion status which can be ARTIC_SUCCESS or an error code. All error codes are explained in detail in the following paragraphs. By writing into this

register, ARTIC acknowledges the command and releases the Transmit Channel Control Block for a possible next command.

Receive Channel Control Block (RCCB)
Each of the four Receive Channel Control Blocks consists of the foHowing fields:
1. Command (1 byte): the type of operation requested. The value of this field can be as follows:
A. ARTIC_CMD_IDLE - No command requested, idle state
B. ARTIC RX NORMAL - Receive Command: NORMAL mode
C. ARTIC_RXCONTNS - Receive Command: CONTINUOUS mode
D. ARTIC_CMD_STOP - Stop receiving
2. Data Start Address (1 word): The address offset of the required receive buffer (if applicable).
3. WORD COUNT – The Buffer Length is 128 words
4. Command Status (1 word): All the above fields are set by the host processor. This field is wriften by the ARTIC processor after receiving a command, within the system's Command Response Time. The value of this field is a command completion status which can be ARTIC_SUCCESS or an error code. All error codes are explained in detail in the following paragraphs. By writing into this register, ARTIC acknowledges the command and releases the Receive Channel Control Block for a possible next command.

OPERATING MODES

Transmitting Modes

1. NORMAL MODE: The board transmits a data buffer (up to 960 ARINC words). Upon termination, a Status Flag is set and an optional Interrupt Request is posted.

2. LOOP MODE: The board transmits a given buffer (up to 9600ARINC Words). Upon termination, transmission is repeated until a STOP command is received. A Status Flag is set and an optional Interrupt Request is posted every time the last word in the buffer is transmitted.

2. REPEAT MODE: The board transmits a given buffer Upon termination, transmission is repeated until it reaches the value set at transmit repeat register

Receiving Modes

1. NORMAL MODE: Received ARINC data words are stored in a user specified buffer.
A Status Flag is set and an optional Interrupt Request is posted.

2. CONTINUOUS MODE: Received ARINC data words are stored in a user specified buffer.
When the buffer is full, the buffer "wraps around" acting as a circular buffer. Each time the buffer is full or half-full, a Status Flag is set and an optional Interrupt Request is posted. Receiving is terminated using the STOP command.

TRANSMITTER OPERATION

Transmitting is started by sending a Transmit Command to the ARTIC system as follows: the Transmit Channel Control Block (TCCB) is loaded with the appropriate values, the Command code being last. The ARTIC system is awakened by the Command byte being written. The Command byte is then read and cleared back to 0. The Command code is then checked as follows:

ARTIC_TX_NORMAL or ARTIC_TX_LOOP or ARTIC_TX_REPEAT:

If the channel is not busy, the data specified by the Start Address and Data Count fields are moved to the board's private memory (provided they are legal, see error handling below) and the Execution Delay and Transmission Intervals are read. The Status byte is then written with the value ARTIC_SUCCESS, and the TCCB is released (note that at this point the application can also start writing new data into the buffer since its contents have been transferred to the system's private memory). At this point the channel is marked as busy in the Channel Activity byte of the GSR and the channel Transmit Counter in the GSR is set to zero. If the Execution Delay is nonzero, a timer is started. When the timer expires (immediately if the value was zero), the first data word is output to the ARINC channel, followed by the rest of the data buffer contents. After each word is output the system checks whether "interrupt on word transmitted" is requested by reading the Interrupt Condition Word in the GSR. If so, an IRQ is generated. The system also sets the appropriate bit in the Transmit Receive Status Word and the channel's Transmit Data Counter is incremented.

ARTIC_TX_STOP command is received by the system at any time during transmission, the system will stop the transmission and mark the channel as not busy in the Channel Activity byte.

If the entire specified data was transmitted, the system will check again the Command code that started the transmission. If it was ARTIC_TX_LOOP, it will reset the channel's Transmit Counter to zero and restart the transmission at the first data word, not before the Execution Delay is introduced (if nonzero).

If the Command code was ARTIC_TX_NORMAL, the system will set the channel's "Transmit Complete" bit in the Interrupt Status Block. If this interrupt is enabled in the Interrupt Condition Block an IRQ will be generated at this point and the channel will be marked as "Not Busy" in the Channel Activity byte of the GSR.

RECEIVIER OPERATION

Receiving is started by sending a Receive Command to the ARTIC system as follows: the Receive Channel Control Block (RCCB) is loaded with the appropriate values, the Command code being last. The ARTIC system is awakened by the Command byte being written. The Command byte is then read and cleared back to 0. The Command code is then checked as follows:

ARTIC_RX_NORMAL or ARTIC_RX_CONTNS:

If the channel is not busy, the contents of the entire RCCB are moved to the board's private memory. The Status byte is then written with the value ARTIC_SUCCESS (provided Receive Buffer range is valid - see Error Handling Paragraph), and the RCCB is released. At this point the channel is marked as busy in the Channel Activity byte of the GSR and the channel Receive Counter in the GSR is set to zero. The receiver is then enabled and the system waits for data on the ARINC channel. When a data word is received, the bit corresponding to its label value in the Label Mask is checked. If its value is 0 (masked), the received data is discarded and no further action is taken. Otherwise (the label is enabled), the system stores the data in the next location in the receive buffer. It then checks whether "interrupt on data received" is requested by reading the Interrupt Condition Word in the GSR. If so, an IRQ is generated. The system also sets the channel's DATA RECEIVED bit in the Transmit/Receive Status Word. The same status bit I IRQ operation is performed if the Half-buffer Full condition is met, i.e. the flow of received data just passed the half-size mark of the receive buffer. The word's Label value is written into the channel's Last Received Label byte of the GSR and the channel's Receive Data Counter is incremented.

If an ARTIC_RX_STOP is received by the system at any time during receive, the system will stop receiving in this channel and mark the channel as not busy in the Channel Activity byte.

If a received data word was stored in the last location of the receive buffer, the system sets the BUFFER FULL bit in the Transmit/Receive Status Word and if the corresponding bit is set in the Interrupt Condition Word, the IRQ is generated. At this point, if the Command was ARTIC_RX_NORMAL, the receive operation is complete and the channel is marked as not busy in the Channel Activity byte in the GSR.

In the Continuous Mode (ARTIC_RX_CONTNS), the channel will continue accepting data and the buffer will "wrap around", i.e. if a data word is received after a BUFFER FULL situation, the data word will be stored at the beginning of the buffer and the channel's Receive Counter will be set to 1. The only way to stop receiving in this mode is via a ARTIC_RX_STOP command.

ARTIC_RX_LABEL:

If the Receive Command code was ARTIC_RX_LABEL, the system uses a fixed length 256x3 byte-long receive buffer (768 bytes). Each of the 256 3-byte cells in this buffer represents an ARINC Label. When a data word is received and it passes the Label Mask test, the system stores it in its corresponding place in the buffer, e.g. Label 0 in the 1st cell and so on. In addition, the next 31 word (256 bits) after the end of the buffer are used by the ARTIC processor to flag newly received labels. The only way to stop receiving in this mode is via a ARTIC_RX_STOP command.

ARINC 708 MEMORY CONTROL SEGMENT

X7D90	**GSR708**	**X7D90**	**Channe_cfg**	**WORD**
		X7D92	**STATUS_TX_RX**	**WORD**
ARINC		**X7D94**	**INTERR_TX_RX**	**WORD**
708		**X7D96**	**PARAM**	**WORD**
GENERAL		**X7D98**	**TX_COUNT**	**WORD**
REGISTERS		**X7D9A**	**RX_COUNT**	**WORD**
		X7D9C	**RESERVED [2]**	**WORD**
X77DA0	**TX708_ctl**	**X7DA0**	**Command**	**WORD**
		X7 DA2	**Start_Addr**	**WORD**
ARINC		**X7DA4**	**Word_Count**	**WORD**
TRANSMIT		**X7 DA6**	**Delay**	**WORD**
CONTROL		**X7 DA8**	**Repeat**	**WORD**
REGISTERS		**X7 DAA**	**RESERVED [2]**	**WORD**
		X7 DAE	**Status**	**INT.**
X7E50	**RX708_ctl**	**X7E50**	**Command**	**WORD**
		X7E52	**Start_Addr**	**WORD**
RECEIVE	**CHANNEL 0**	**X7E54**	**Buffer_Size**	**WORD**
CONTROL		**X7E56**	**Label_Mask[16]**	**WORD**
REGISTERS		**X7E76**	**Status**	**INT.**
	ANNEL1	**X7E78**	**Command**	**WORD**
		X7E7A	**Start_Addr**	**WORD**
		X7E7C	**Buffer_Size**	**WORD**
		X7E7E	**Label_Mask[16]**	**WORD**
		X7E8E	**Status**	**INT.**
		X7E90	**Command**	**WORD**
		X7E92	**Start_Addr**	**WORD**
	CHANNEL3	**X7E94**	**Buffer_Size**	**WORD**
		X7E96	**Label_Mask[16]**	**WORD**
		X7EA6	**Status**	**INT.**
		X7EA8	**Command**	**WORD**
		X7EAA	**Start_Addr**	**WORD**
		X7EAC	**Buffer_Size**	**WORD**
		X7EAE	**Label_Mask[16]**	**WORD**
		X7EBE	**Status**	**INT.**

ARINC 717

INTRODUCTION:

Arinc 573/717 is a protocol used on flight data acquisition and recording system.
Arinc 717 replaces the older 573 keeping the same letrical characteristics "Harvard bi-phase encoding".
The protocol defines an ARINC 717 word as 12 BIT.
An ARINC 717 frame are composed of 4 blocks, each with the same number of 12 bits words.
The board has a 16 bit oriented BUS. The last 4 bits, bits 12-13-14- & 15 are ignored.
The user MUST store the correct information on the first 12 bits on each address word.

Bit Rates:
ARINC 717 data is sent at a nominal rate of 768 bits per/second.
The board provides programmable output rates of 64 or 128 or 256 or 512 ARINC 717 words p/second.

Frame Formats:
ARINC 717 words consist of 12 bits sent a nominal rate of 768 bits p/second.
Data words are formatted into four sub-frames of 64 words.
The complete frame repeats every four seconds and consist of four sub-frames.
The first word of each sub-fram provides a synchronization pattern. There is a unique synchronization for each sub-fram as show bellow:

SUB-FRAME	PATTERN (octal)
1	1107
2	2670
3	5107
4	6670

Specific data is identified as "time slot" addresses.
The asssigment of data parameters to word in the frames is application-specific and is determined by the system designer.
Data it is stored into the dual port memory and follow a command the unit transmit the frame.
The unit is software programmable to select data received between Harvard bi-phase or NRZ input signals.
The unit transmit a pre defined number of words, which can be from 1 to 65k, and intermessage delay using artic_tx_command
(ARTIC USERS API document)

ARINC 717 BUS

The ARINC 717 has two differential outputs. Signal are complement, Harvard bi-phase encoding and NRZI, with output voltage from ground to +-5VDC.
The signals cable should be shielded with insulating jacket.

Data Area

The 32K BYTES (PCI,CPCI, PMC) of the Dual Port RAM are used as a general purpose data area. The user is free to allocate Transmit and Receive buffers and locate them at their discretion. This area is usually used for storing the actual contents of messages, rather than ARTIC operational instructions.

The transmission data start from the LSB- Least Significant bit – BIT 0 to the MSB – Most Significant Bit-BIT 12. Due to the fact the Arinc 717 is bit oriented the user should be carefull with the bit loction in the frame.

The unit allocates a fixed 1k words for the receiver buffer starting from the receive start address register. Receive data is stored into a circular buffer. The Dual Port memory is structured as 16 bits p/ word address, ARINC 717 has 12 bits only. The Most Significant Nibble on the DPM memory has always zero value. The information has bits 0-11 ARINC 717 value abd bits 12-15 are filled with zeros.

Global System Registers (GSR)

The next section of memory in the dual port RAM after the end of the Data Area is the Global Systems Register. This area is used to store general ARTIC 717 configuration information, and to record the progress of the current operation or procedure. The GSR's structure consists of the following:

1-. Channel Configuration (1 WORD):
This words shows the active or in use channels on the board.
The first two bits; bits 0-1, defined the channel operational status:

- **0 - channel 1 receive operating**
- **1- channel 1 transmit operating**
- **8 - channel 2 receive operating**
- **9- channel 2 transmit operating**

2. STATUS_TX_RX Transmit/Receive Indicator (1 word):
This field is set by the board and read by the host. The Following fields are used in this data structure:
A. Transmit Complete – bits 0
B- Reserved - bit 1
C. Receive Buffer Full – BIT 2
D. Receive Buffer Half-full – BIT 3

A bit in this word is set by the ARTIC system every time the condition represented by this bit is true. This word is polled by the host in order to identify any relevant events, e.g. receive buffer full, and then cleared by the host to allow new events to be identified next time. In interrupt mode, the interrupt service routine (ISR) will read this word to identify the event that triggered the interrupt. In polled mode, the application program will read this word occasionally as dictated by the programs flow to check if any relevant event has happened.
The first bit of this field is a transmit flag which is set by the board each time any transmit operation has occurred. The second bit is a receive bit which is likewise set every time a receive operation has occurred. They indicate whether the boards transceivers are transmitting or receiving actual signals to or from the ARINC data bus. Keep in mind that the user must clear these bits every time he reads them or they will remain high (binary '1') forever after the first time they are set by the board. The user will thus be receiving outdated information. Each channel is represented by a pair of bits.

.
3. INTERR_TX_RX Interrupt Selector Register (1 word):
In general, if a user needs to know whether the board has transmitted or received, he can poll the transmit and/or receive flag in the previous field at regular intervals. However, there may be times when the user needs to know immediately should such an event happen. This field allows the user to set the board so that if a transmission or reception occurs it will immediately trigger an interrupt in the host PC.
This word is bit-by-bit identical to the Transmit/Receive Indicator Field above. However, this word is written by the host (PC). By setting any of the bits to a '1' the host may request an interrupt every time the condition represented by that bit occurs. The interrupt service routine (ISR) can then identify the source of the interrupt by reading the above described Transmit/Receive Indicator Field.
This field is set by the board and read by the host. The Following fields are used in this data structure:
A. Transmit Complete – bits 0 -4
B- Reserved - bit 1 -5
C. Receive Buffer Full – BIT 2 -6
D. Receive Buffer Half-full – BIT 3 - 7

.

4. PARAM :

The second nibble - first two bits; bits 4-5 for channel 1 and fourth nibble - first two bits; bits 12-13 for channel, defined the channel operational status:

Channel 1:

4-5- defines baud rate:

00 – 64 words p/second
01 – 128 words p/second
10 – 256 words p/second
11 – 512 words p/second

2 reserved

3 - selection Harvard bi-phase or NRZ output signals

1- defines receive data source:
0- external receive data
1- data transmitted by the board - internal loop back

6-7 reserved

Channel 2:

12-13- defines baud rate:

00 – 64 words p/second
01 – 128 words p/second
10 – 256 words p/second
11 – 512 words p/second

10 reserved

11 - selection Harvard bi-phase or NRZ output signals

1- defines receive data source:
0- external receive data
1- data transmitted by the board - internal loop back

12-15 reserved

5. TX COUNT:

Corresponds to the frame transmit message counter. It is incremented every end of transmit block message.

6. RX_COUNT:

Corresponds to the number of data receive message counter, incremented every received ARINC 717 word.

CHANNEL CONTROL BLOCKS:

Transmit Channel Control Block (TCCB)

Transmit Channel Control Blocks consists of the following fields:

1. Command (1 byte): the type of operation requested. The value of this field can be as follows:

A. ARTIC_CMD_IDLE - No command requested, idle state

B. ARTIC_TX_NORMAL - Transmit Command: NORMAL mode

C. ARTIC_TX_LOOP - Transmit Command: LOOP mode

C. ARTIC_TX_REPEAT - Transmit Command: REPEAT mode

D. ARTIC_TX_STOP - Stop transmission

2. Data Start Address (1 word): The offset address (relative to the board address) of the first ARINC word to be transmitted (if applicable).

3. Data Count (1 word): The number of words, starting at the above Start Address, to be transmitted (if applicable).

4. Execution Delay (1 word): Time period in increments of 65,512 MILISECOND between every transmission frame on LOOP or REPEAT commands.

5. Command Status (1 word): All the above fields are set by the host processor. This field is written by the ARTIC processor after receiving a command, within the system's Command Response Time. The value of this field is a command completion status which can be ARTIC_SUCCESS or an error code. All error codes are explained in detail in the following paragraphs. By writing into this register, ARTIC acknowledges the command and releases the Transmit Channel Control Block for a possible next command.

Receive Channel Control Block (RCCB)

Each of the four Receive Channel Control Blocks consists of the foHowing fields:

1. Command (1 byte): the type of operation requested. The value of this field can be as follows:

A. ARTIC_CMD_IDLE - No command requested, idle state

B. ARTIC RX NORMAL - Receive Command: NORMAL mode

C. ARTIC_RXCONTNS - Receive Command: CONTINUOUS mode

D. ARTIC_CMD_STOP - Stop receiving

2. Data Start Address (1 word): The address offset of the required receive buffer (if applicable).

3. WORD COUNT – The Buffer Length SHOULD BE 64 or 128 or 256 or 512 ARINC 717 words. PLEASE NOTE incorrect value will create incorrect transmission. The transmission rate is defined in the 717 baud rate command.

4. Command Status (1 word): All the above fields are set by the host processor. This field is wriften by the ARTIC processor after receiving a command, within the system's Command Response Time. The value of this field is a command completion status which can be ARTIC_SUCCESS or an error code. All error codes are explained in detail in the following paragraphs. By writing into this register, ARTIC acknowledges the command and releases the Receive Channel Control Block for a possible next command.

OPERATING MODES

Transmitting Modes

1. NORMAL MODE: The board transmits a data buffer. Upon termination, a Status Flag is
set and an optional Interrupt Request is posted.
2. LOOP MODE: The board transmits a given buffer. Upon termination, transmission is repeated until a STOP command is received. A
Status Flag is set and an optional Interrupt Request is posted every time the last word in the buffer is transmitted. There is an intermessage gap if the delay register is different than zero.
2. REPEAT MODE: The board transmits a given buffer Upon termination, transmission is repeated until it reaches the value set at transmit repeat register. There is an intermessage gap if the delay register is different than zero

Receiving Modes

1. NORMAL MODE: Received ARINC data words are stored in a user specified buffer.
A Status Flag is set and an optional Interrupt Request is posted.

2. CONTINUOUS MODE: Received ARINC data words are stored in a user specified buffer.
When the buffer is full, the buffer "wraps around" acting as a circular buffer. Each time the buffer is full or half-full, a Status Flag is set and an optional Interrupt Request is posted. Receiving is terminated using the STOP command.

TRANSMITTER OPERATION

Transmitting is started by sending a Transmit Command to the ARTIC system as follows: the Transmit Channel Control Block (TCCB) is loaded with the appropriate values, the Command code being last. The ARTIC system is awakened by the Command byte being written. The Command byte is then read and cleared back to 0. The Command code is then checked as follows:

ARTIC_TX_NORMAL or ARTIC_TX_LOOP or ARTIC_TX_REPEAT:

If the channel is not busy, the data specified by the Start Address and Data Count fields are moved to the board's private memory (provided they are legal, see error handling below) and the Execution Delay and Transmission Intervals are read. The Status byte is then written with the value ARTIC_SUCCESS, and the TCCB is released (note that at this point the application can also start writing new data into the buffer since its contents have been transferred to the system's private memory). At this point the channel is marked as busy in the Channel Activity byte of the GSR and the channel Transmit Counter in the GSR is set to zero. If the Execution Delay is nonzero, a timer is started. When the timer expires (immediately if the value was zero), the first data word is output to the ARINC channel, followed by the rest of the data buffer contents. After each word is output the system checks whether "interrupt on word transmitted" is requested by reading the Interrupt Condition Word in the GSR. If so, an IRQ is generated. The system also sets the appropriate bit in the Transmit Receive Status Word and the channel's Transmit Data Counter is incremented.

ARTIC_TX_STOP command is received by the system at any time during transmission, the system will stop the transmission and mark the channel as not busy in the Channel Activity byte.

If the entire specified data was transmitted, the system will check again the Command code that started the transmission. If it was ARTIC_TX_LOOP, it will reset the channel's Transmit Counter to zero and restart the transmission at the first data word, not before the Execution Delay is introduced (if nonzero).

If the Command code was ARTIC_TX_NORMAL, the system will set the channel's "Transmit Complete" bit in the Interrupt Status Block. If this interrupt is enabled in the Interrupt Condition Block an IRQ will be generated at this point and the channel will be marked as "Not Busy" in the Channel Activity byte of the GSR.

RECEIVIER OPERATION

Receiving is started by sending a Receive Command to the ARTIC system as follows: the Receive Channel Control Block (RCCB) is loaded with the appropriate values, the Command code being last. The ARTIC system is awakened by the Command byte being written. The Command byte is then read and cleared back to 0. The Command code is then checked as follows:

ARTIC_RX_NORMAL or ARTIC_RX_CONTNS:

If the channel is not busy, the contents of the entire RCCB are moved to the board's private memory. The Status byte is then written with the value ARTIC_SUCCESS (provided Receive Buffer range is valid - see Error Handling Paragraph), and the RCCB is released. At this point the channel is marked as busy in the Channel Activity byte of the GSR and the channel Receive Counter in the GSR is set to zero. The receiver is then enabled and the system waits for data on the ARINC channel. When a data word is received, the bit corresponding to its label value in the Label Mask is checked. If its value is 0 (masked), the received data is discarded and no further action is taken. Otherwise (the label is enabled), the system stores the data in the next location in the receive buffer. It then checks whether "interrupt on data received" is requested by reading the Interrupt Condition Word in the GSR. If so, an IRQ is generated. The system also sets the channel's DATA RECEIVED bit in the Transmit/Receive Status Word. The same status bit I IRQ operation is performed if the Half-buffer Full condition is met, i.e. the flow of received data just passed the half-size mark of the receive buffer. The word's Label value is written into the channel's Last Received Label byte of the GSR and the channel's Receive Data Counter is incremented.

If an ARTIC_RX_STOP is received by the system at any time during receive, the system will stop receiving in this channel and mark the channel as not busy in the Channel Activity byte.

If a received data word was stored in the last location of the receive buffer, the system sets the BUFFER FULL bit in the Transmit/Receive Status Word and if the corresponding bit is set in the Interrupt Condition Word, the IRQ is generated. At this point, if the Command was ARTIC_RX_NORMAL, the receive operation is complete and the channel is marked as not busy in the Channel Activity byte in the GSR.

In the Continuous Mode (ARTIC_RX_CONTNS), the channel will continue accepting data and the buffer will "wrap around", i.e. if a data word is received after a BUFFER FULL situation, the data word will be stored at the beginning of the buffer. The only way to stop receiving in this mode is via a ARTIC_RX_STOP command.

ARINC 717 MEMORY CONTROL SEGMENT

The units has up to two independent ARINC 717 transmit/receive channels.

X7D50	**GSR717**	**X7D50**	**Channe_cfg**	**WORD**
		X7D52	**STATUS_TX_RX**	**WORD**
ARINC		**X7D54**	**INTERR_TX_RX**	**WORD**
717		**X7D56**	**PARAM**	**WORD**
GENERAL		**X7D58**	**TX_COUNT [2]**	**WORD**
REGISTERS		**X7D5C**	**RX_COUNT [2]**	**WORD**
X7D64	**TX717_ctl**	**X7D60**	**Command**	**WORD**
		X7D62	**Start_Addr**	**WORD**
ARINC	**Channel 1**	**X7 D64**	**Word_Count**	**WORD**
TRANSMIT		**X7D66**	**Repeat**	**WORD**
CONTROL		**X7 D68**	**Delay**	**WORD**
REGISTERS		**X7D6A**	**Status**	**INT.**
	Channel 2	**X7 D6C**	**Command**	**WORD**
		X7D6E	**Start_Addr**	**WORD**
		X7 D70	**Word_Count**	**WORD**
		X7D72	**Repeat**	**WORD**
		X7 D74	**Delay**	**WORD**
		X7 D76	**Status**	**INT.**
X7D7C	**RX717_ctl[4]**	**X7E30**	**Command**	**WORD**
		X7E32	**Start_Addr**	**WORD**
RECEIVE	**CHANNEL 0**	**X7E34**	**Buffer_Size**	**WORD**
CONTROL		**X7E36**	**Label_Mask[16]**	**WORD**
REGISTERS		**X7E76**	**Rec_Index**	**INT.**
	CHANNEL1	**X7E78**	**Status**	**WORD**
		X7E7A	**Start_Addr**	**WORD**
	Channel 2	**X7E7C**	**Buffer_Size**	**WORD**
		X7E7E	**Label_Mask[16]**	**WORD**
		X7EBE	**reserved**	**INT.**
		X7E80	**Command**	**WORD**
		X7E82	**Start_Addr**	**WORD**
	CHANNEL3	**X7E84**	**Buffer_Size**	**WORD**
		X7E96	**Label_Mask[16]**	**WORD**
		X7EA6	**Status**	**INT.**
		X7EA8	**Command**	**WORD**
		X7EAA	**Start_Addr**	**WORD**
		X7EAC	**Buffer_Size**	**WORD**
		X7EAE	**Label_Mask[16]**	**WORD**
		X7EBE	**Status**	**INT.**