TECHNICAL MANUAL

The PACS® Standard

PEAKTRONICS

CONTENTS

SECTION I
The PACS® System

SYSTEM OVERVIEW

Peaktronics Asynchronous Communications System, known as PACS®, is a formerly patented process for a software dependent communications scheme intended for high speed easily managed communication links between electronic equipment. Standardization of the hardware and software elements of the system provide for universal applications while allowing for versatility and easy expansion of the system.

The hardware essentially involves a standard low cost PACS® Cable (see Figure 1) that is used to connect equipment that includes standard PACS® interface circuitry. Establishing a communications link with PACS® requires no settings or synchronizing of any kind. The standard PACS® hardware will automatically interface for the most optimum performance allowed by the two devices connected together and the length and condition of the cable connecting the two devices.

The software is a command set of universal conversation codes that are easily adapted in any computer or peripheral device. PACS® Level 1 is the most basic set of codes that is most useful in industrial controllers, data terminals, keyboards. etc., where code size is usually a concern for the local microprocessor. PACS® Level 1 is upward compatible with other levels of PACS®, thereby eliminating concerns for obsolescence or incompatibility.

The PACS® system can be broken down into two basic elements: the PACS® Master and the PACS® Slave. A better understanding of this will be useful in mastering the PACS® software.

PACS® MASTER

A PACS® Master is any device that controls the communication process within the system. All hardware and software sequences are initiated by the PACS® Master. A PACS® Master is connected to a PACS® Slave via a PACS® Cable. The PACS® Slave is expected to respond in a certain fashion to the sequences initiated by the PACS® Master. A single piece of equipment may have more than one PACS® Master, or port, to allow connections to more than one PACS® Slave at a time.

PACS® SLAVE

A PACS® Slave is any device that responds to the sequences initiated by a PACS® Master. A PACS® Slave is connected to a PACS® Master via a PACS® Cable. The PACS® Slave will respond in a certain fashion to all hardware and software sequences initiated by a PACS® Master. A single piece of equipment may have more than one PACS® Slave, or port, to allow connections to more than one PACS® Master at a time.

PACS® CABLE

The PACS® Cable is a standardized connection system for connecting a PACS® Master to a PACS® Slave. The standard defines the connector type and the cable type with color code (see Figure 1).

The connectors are easily crimped to the cable for a quick and reliable connection. Note that the connectors are inverted on opposite sides of the cable, so that the pinout has the same color code on both ends of the cable. Since both ends of the cable are the same, either end may be connected to the PACS® Master or PACS® Slave.

Conformance to the color code allows for easy identification of wires when the other end of the cable is not accessible; however, the colors could be reversed at both ends and result in the same connection. Should either end of the cable be wired in reverse, the PACS® Standard provides that the PACS® Master and PACS® Slave shall not be damaged by incorrect wiring of the cable.

The PACS® Cable standard is intended to define the interconnection for "external" communication ports. However, for localized PACS® connections (such as board to board) that are not intended for general hookup to other PACS® devices, virtually any four-wire connection scheme could be used. The PACS® Standard provides the means for both sides to adjust to other connection types and lengths. The PACS® Cable shown in Figure 1 is not recommended for lengths of more than 5000 feet.

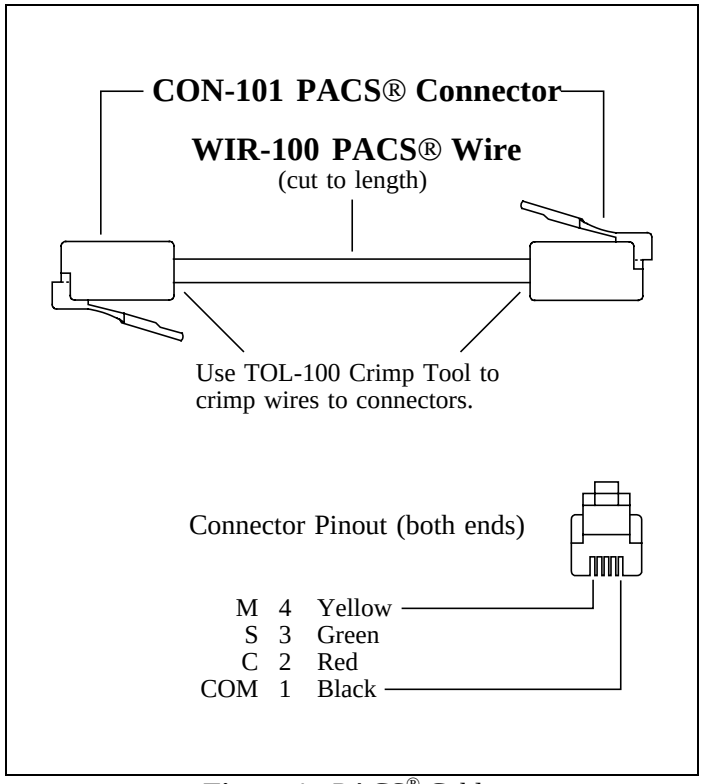

Figure 1 - PACS® Cable

PACS® SEQUENCE

The communication process is achieved through a series of signals called a PACS® Sequence. The PACS® Master will always start the communication sequence and the sequence is terminated according to a predefined format, known as the PACS® Standard. During the sequence, data is verified, acknowledged, and decoded.

The most basic PACS® system would consist of a master connected to a slave where each is responsible for certain signals at certain times in order to complete a PACS® sequence. The master controls the signal sequence while the slave responds to the signals generated by the master. If two slaves are connected together, both will be waiting for the other to start a sequence, and communications will not occur. If two masters are connected together, neither will respond to the other, and again communications will not occur.

In the form of software, a communication sequence is called a "command string". All communication sequences, or command strings, will be a specific instruction or command from the PACS® Master to the PACS® Slave. The PACS® Slave will never return data to a PACS® Master unless the master commands the slave to do so, and then, under the control of the master.

Each command in the PACS® command set defines a communication sequence, also called a command string. For commands that instruct the PACS® Slave to return data, such as the R (READ) command, the returned data is considered to be part of the same PACS® sequence. Therefore, the returning data must be completely received by the master before another sequence can be initiated.

During the PACS® sequence, both the master and slave are required to exchange signals in a predefined fashion such that both devices can detect errors or troubles in the hardware as well as the software. This process is proprietary information at this time. For information on licensing or information on applications, contact Peaktronics, Inc.

Whenever an error condition is detected by the slave, the slave will notify the master and then ignores the command string that resulted in an error. Whenever the master detects an error, the master will instruct the slave to ignore the command string. In either case, the master will know of any error condition, allowing the master the option of attempting to retransmit the sequence.

OFF LINE TIMER

In practice, there are various conditions that could interrupt communications, such as a broken cable or a loss of power to either the master or slave. Due to the asynchronous feature of PACS®, a device would "hang-up" waiting for a response from the other side should the

sequence be interrupted. For this reason, a PACS® device will maintain an off line timer.

The off line timer in a PACS® device will allow a set amount of time for the other device to respond. Should the other device fail to respond within the time period allowed, the sequence will be terminated and ignored; the other device will then be considered to be off line. The amount of time allowed is to be specified by the designer of the PACS® device. The designer is given the flexibility of the time period so as not to restrict any particular application.

In general, PACS® Master devices will tend to have a short off line timer in comparison to a PACS® Slave. This is usually the case since a master will have other tasks to perform depending on the status of the slaves, or peripherals, that it is connected to.

In some cases, a PACS® Slave may have an indefinite off line timer, allowing the master as much time as it may need. Should the master lose power and then power up again, the master would start a new PACS® sequence while the slave is expecting a continuation of the previous sequence; this is called a "sync error" and is handled by the error detection process. As a result, the slave will ignore the sequence that was interrupted by the power loss at the master side, and the master will retransmit the new sequence that was attempted on power up.

Some devices may be intended for general purpose use. In this case, selecting a specific off line timer period may not be suitable for all applications. This type of device, therefore, could have an off line timer that is programmed by the user either through software or hardware.

PACS® INTERCHANGE

The most basic PACS® system would consist of a master connected to a slave; however this would have very limited applications. An easy method for expanding the system would be to provide the master unit with several PACS® master ports for connecting to several peripherals. Also, a peripheral could have several PACS® Slave ports, allowing access by more than one master unit. A more efficient and versatile method of expansion is accomplished with the use of a PACS® Interchange. A PACS® Interchange is a device that allows one or more masters to connect to one or more slaves (see Figure 2).

An interchange is a unit that appears as a single peripheral, or slave, to the master units connected to it. Each slave unit connected to the interchange is represented as a bank of memory that may be accessed by any of the master units. An interchange will automatically handle the communication process to each slave unit. The master unit need only access a "memory bank" (actually a slave unit). This process is specified by the CHAN ID command (see Section II for details).

Since an interchange unit appears as a slave unit to a PACS® Master, any one of the slave units shown in Figure 2 could be another PACS® Interchange unit. Subsequently, that interchange could also connect to an additional interchange, and so on. Accessing each "tier" of interchanges is done with the TIER command (see Section II). While certain limitations arise in actual practice, a combination of TIER and CHAN ID commands could allow a master unit to access literally trillions of slave devices connected to a series of "tiered" interchanges.

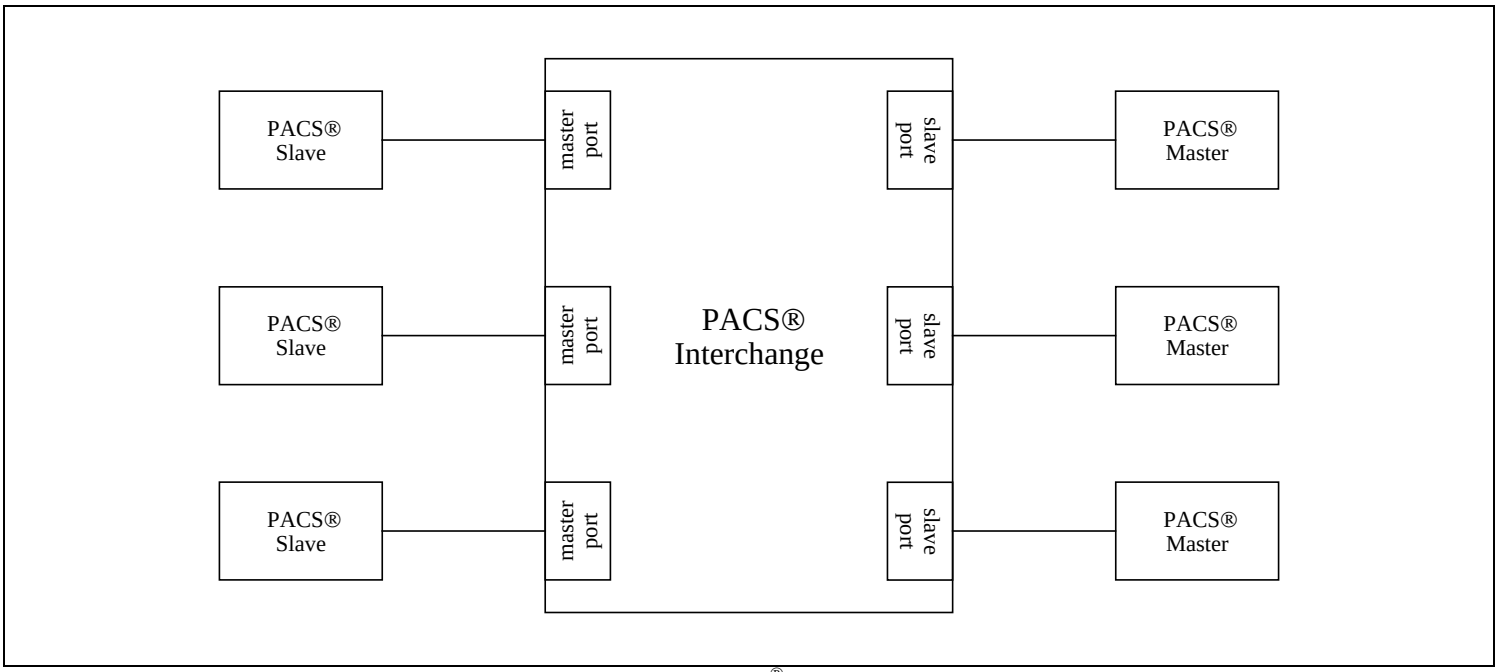

Figure 2 - PACS® Interchange

SECTION II
The PACS® Format

The PACS® software system is designed to be easily adapted by microprocessors of all types. The basic concept of PACS® is to give the master access to the internal memory of a slave; thereby allowing the master to either change or read data in a slave that is pertinent to its operation.

The PACS® format is very straight forward and consistent so that a programmer does not concern himself with how to access a peripheral, but rather, where to access the peripheral's memory. Information detailing the key locations to access in a peripheral should be found within the manual for the peripheral. This discussion will concern itself with the details of how to use the PACS® software.

PACS® SEQUENCE

A complete communication sequence, or PACS® sequence, is called a command string and consists of three pieces of information:

- The first part of a command string represents the COMMAND that tells the slave specifically what operation the master wishes to perform.
- The second portion of the string is the ADDRESS (or memory location) that the master wishes to perform the operation on.
- The third portion of the string represents the DATA that is to operate with the selected address.

A fourth element of a PACS® sequence is primarily a hardware condition, where the slave is required to respond to the transmitted information to confirm that the Command, Address, and Data correlate with each other. To complete a sequence, the master then confirms concurrence to the slave's response. If the master should not concur, the slave will ignore the entire sequence.

Information transmitted across a PACS® Cable is in a binary format. This binary format is inclusive and requires no control characters for proper processing. A command string consists of at least an eight bit command which will define the operation that is to be performed (such as ADD, READ, etc.). Depending on the specific command, a 16 bit address will follow the 8 bit command, and depending on the type of operation, a data word of either 8, 16, or 32 bits will follow the address.

DATA MODE

Each eight bit block is called a byte, so a data word can therefore be either a SINGLE byte, a DOUBLE byte, or a QUAD byte. The size of the data word defines the DATA MODE. In the following example, each byte of a command string is represented by a hexadecimal value,

where 64 is the command byte, 9A21 is the two-byte (16 bit) address, and 2C is a single byte data word:

64 9A21 2C

The above command is said to have a single byte data mode.

The previous command string specifically tells the slave to ADD the contents of memory location 9A21 to the value of 2C, and then replace the contents of 9A21 with the result of the addition. Changing the command byte to a value of 84 changes the data mode to a double byte word:

84 9A21 2C10

In the above example, the 16 bit data word, 2C10, is added to the 16 bit value stored at location 9A21 and 9A22, where 9A21 will contain the most significant byte. The 16 bit result will replace the contents of 9A21 and 9A22.

In a similar fashion, changing the command byte to C4, the data mode is changed to a quad byte:

C4 9A21 410A2C10

In this case, the 32 bit data word (41 being the most significant byte) is added to the contents of locations 9A21, 9A22, 9A23, and 9A24 where the most significant byte is 9A21 and the least significant byte is 9A24. As before, the sum replaces the contents of 9A21 through 9A24.

Some commands do not require data to follow the address bytes, or otherwise known as an inherent command. An example of this is the INCR, or increment, command. This command instructs the slave to automatically add one to the value in the selected address.

Although the data is not supplied by the master, the master must still define the data mode:

54 9A21 SINGLE
55 9A21 DOUBLE
56 9A21 QUAD

In the above examples, the command byte 54 instructs the slave to increment the contents of 9A21, while 55 commands the slave to increment the 16 bit value in 9A21 (most significant) and 9A22. Command byte 56 instructs the slave to increment the 32 bit value at 9A21 through 9A24.

ADDRESS MODE

In the previous examples, the command string includes an address (that is, 9A21). This address tells the slave specifically which location the master wishes to operate on. This is called DIRECT ADDRESSING MODE since the master directly tells the slave which address is to be used.

The PACS® Standard provides that a slave will maintain an index pointer which points to the memory location that is to be operated on by a PACS® command. Whenever the master uses Direct Addressing, the slave will set its index pointer to the value stipulated by the address bytes in the command string.

Consider the following command string:

84 9A21 2C10

When the slave has received the command string, the slave will set its index pointer to 9A21. This will guide the slave in performing the addition process dictated by the command byte, 84. At the conclusion of the addition process, the slave will have changed the contents of 9A21 and 9A22 (a double byte word) to the result of the addition, and the index pointer will be set to 9A23. This allows the index pointer in the slave to point to the next available location.

A master unit can take advantage of the knowledge that the slave's index pointer will be set to a predefined value (in this case, 9A23). By using the INDEXED ADDRESSING MODE, the master can avoid transmitting the 16 bit address. Consider the following example:

84 9A21 2C10
15

As discussed before, the 84 command will add the value of 2C10 to the contents of 9A21 and 9A22, and upon the conclusion of the addition, the index pointer is set to 9A23. The following command string is now only one byte, 15. This command will instruct the slave to increment the double byte (double byte data mode) pointed to by the index pointer. In other words, the 16 bit value located at 9A23 and 9A24 is incremented by one. At the completion of the increment process, the slave will set the index pointer to the next successive location (that is 9A25).

The command byte 15 could have been replaced with the following command string:

55 9A23

However, since the master already knew that the index pointer in the slave was pointing to 9A23, the address bytes were not required and therefore Indexed Addressing could be used.

While indexed addressing provides efficiency for the master, certain precautions need to be taken when using Indexed Addressing. In the event that the slave unit should lose power momentarily, the slave will reset its index pointer to location zero when power is returned. The master may want to monitor the slave's power conditions if certain indexed commands are crucial. Information on how

to monitor power conditions should be available in the manual for the slave unit.

If an error occurs during a PACS® sequence, both the slave and master units will detect such a condition. The slave is required to perform no operation in response to an error which means that it will not change its index pointer. However, it is common for an error to occur during the first PACS® sequence after a slave unit powers up in which case the index pointer is reset to zero. Again, the master may need to monitor the slave's power conditions.

Another concern while using Indexed Addressing is to insure that no other master unit (via an interchange, see Section I) has accessed the slave unit, thereby changing the slave's index pointer to an unknown value. To insure expected results from an indexed command, the master unit may have to program special features in the interchange, or have a thorough understanding of the system so as to avoid unexpected results.

READ COMMAND

Some commands have special considerations. One such command is the READ command. A command string for a READ command is unique in the sense that the data word, which can either be single, double, or quad, is supplied by the slave unit. Consider the following:

master transmits:
51 9A21
slave transmits:
1F05
master transmits:
12
slave transmits:
0023A58C

In the above example, the command byte 51 will instruct the slave to return the data located in 9A21 and 9A22 as a double byte word (in this case, 1F05). The actual process of receiving data from a slave is transparent to the user, but it should be understood that the READ command is considered to be inherent (meaning that the master does not supply the data), and that the data which is transferred from the slave to the master is part of the PACS® sequence.

Since the master dictates the data mode, the master can command data transfers that are either 8, 16, or 32 bits long. Additionally, the master still dictates the location(s) to be transferred, and therefore the slave must maintain its index pointer accordingly. This allows for the indexed addressing mode.

In the example above, the master transmits the single byte command, 12, after receiving the data from the slave. This command instructs the slave to return a quad byte using the indexed addressing mode. The master does not transmit the address information; instead the slave will transmit the data stored at the locations pointed to by its index pointer. In this way, the master need only transmit one byte to command the slave to return 4.

CHAN ID COMMAND

Another command that requires special consideration is the CHAN ID command. This command is used to command the slave unit to switch to another memory bank for all subsequent commands. This is useful when a slave has more locations than can be addressed with the basic 16 bit address that follows a command byte.

Memory banks are numbered from 0 to 1023 which allows a slave unit to incorporate very large amounts of memory (up to 67,107,840 bytes) that may be directly accessible by a master. To better understand how this is implemented, consider the following series of commands:

3C 04
62 0500 0A
3C 08
62 0500 01

The 3C command instructs the slave to switch memory banks, the following data byte, 04 tells the slave which memory bank to switch to. The next command, 62, tells the slave to change the contents of location 0500 to a value of 0A. Since the selected memory bank is 04, only location 0500 within memory bank 04 is affected.

To change locations in another memory bank, the CHAN ID command will need to be given again. In this case, 3C 08 instructs the slave to make memory bank 08 accessible to the master. The following CHANGE command, 62 0500 01, tells the slave to change the contents of location 0500 to a value of 01, and since the selected memory bank is 08, only location 0500 within bank 08 is affected. The previous value of 0A is still in location 0500 within bank 04.

Not all slave units are going to be able to make use of the CHAN ID command, and in fact, most slave units would be hard pressed to offer more than the normal 65,535 bytes accessible with the 16 bit address. For this reason, a key PACS® Standard is implemented here to insure compatibility.

All PACS® Slave devices will receive the CHAN ID command as a valid hardware sequence, however, how the slave "executes" the command can be done in one of two ways:

1) The slave can ignore the command, perform no operation at all, and leave the index pointer at its previous value. This option allows for reduced software within the slave and effectively "shadows"

its only memory bank at any bank selected by the master.

2) The slave unit can respond to the command by allowing access to another section, or bank, of memory. The slave can shadow any bank at various CHAN ID selections, or it may respond to unused banks in a predefined way.

Ultimately, the instruction manual for the slave unit should define how the slave will respond to the CHAN ID command. In any case, no matter how the slave executes the command, the slave will NEVER CHANGE THE INDEX POINTER in response to the CHAN ID command. The final responsibility of performance of the CHAN ID command falls on the master's control program. The master's control unit must know what to expect from the slave when using the CHAN ID command.

The term "memory bank" as described above would tend to infer that each bank consists of some type of electronic storage device that is located within the slave. As illustrated in Figure 3, a master can access three different RAM devices, each consisting of some type of storage hardware located inside of the slave. However, any or all of these "memory banks" could be an independent unit outside of the slave. When one or more of these independent units happen to be another PACS® Slave, a PACS® Interchange is created (see Figure 4).

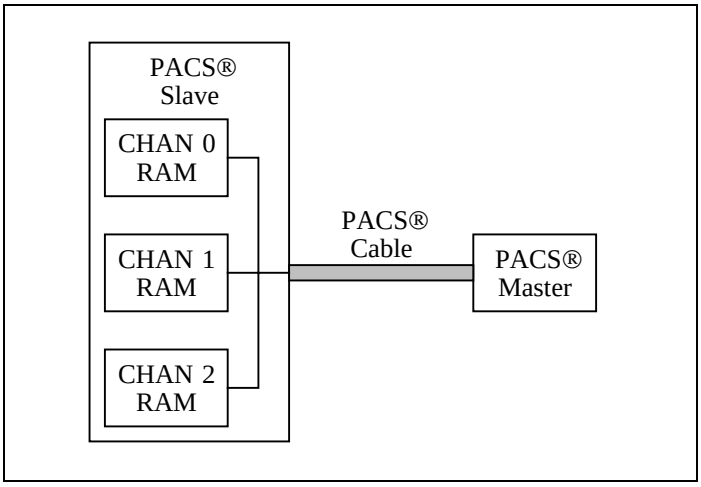

Figure 3 – CHAN ID Accessible Memory Banks

In Figure 4, the master still accesses each bank of memory as it did in Figure 3. The only difference is that the interchange in Figure 4 uses a PACS® Slave to provide the storage device for memory banks 1 and 2. Notice that the interchange uses a local RAM device for CHAN 0. This allows the master to either program features in the interchange, or to monitor key information about the interchange's operation. This process could allow a master to access up to 1024 slave units; this includes the PACS® Interchange which is ALWAYS assigned to CHAN 0.

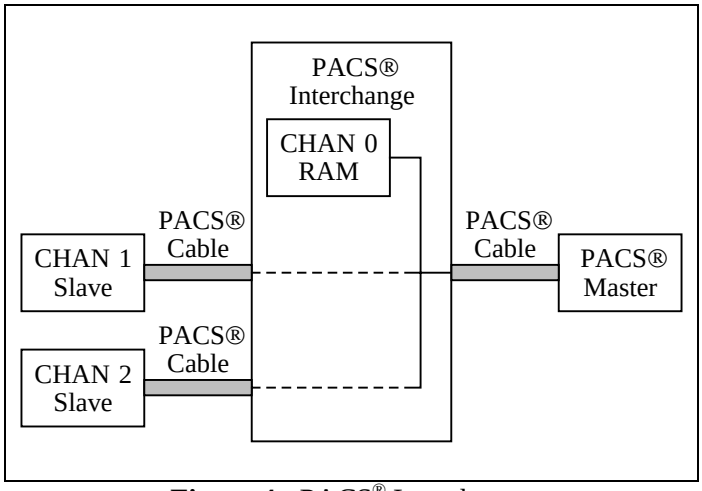

Figure 4 - PACS® Interchange

TIER COMMAND

While the CHAN ID command provides access to a large number of PACS® Slave units, a far greater expansion of the PACS® System is achieved with the TIER command. The PACS® Master in Figure 4 obtains access to the one of the slave units because the PACS® Interchange interprets the CHAN ID command in such a way that all subsequent commands from the master unit are literally passed to the associated slave unit until a new CHAN ID is provided by the master. If one of the slave units happened to be a PACS® Interchange, the master unit would not be able to send a CHAN ID command to the interchange since the first interchange unit (connected directly to the master unit) will interpret the CHAN ID command for itself.

The TIER command essentially instructs the first interchange unit to pass a CHAN ID command to the associated slave unit. Part of the TIER command provides the CHAN ID value that is to be passed on; this eliminates the need to generate a separate CHAN ID command after the TIER command.

Figure 5 illustrates how quickly a large number of slave units can be accessible to a single master unit. The PACS® Master can access the Tier 0 interchange by sending a TIER command with a tier value of 0 along with the desired CHAN ID value. However the same could be accomplished by merely sending the desired CHAN ID command. Once a desired Tier 1 interchange has been selected, the master would then send a TIER command with a tier value of 1 along with a CHAN ID value to select the desired Tier 2 interchange. This would then be followed by a TIER command with a tier value of 2 and a CHAN ID value to select the desired slave unit connected to one of the Tier 2 interchanges. This process can be extended to a group of interchanges connected at Tier 255.

Examining Figure 5, the Tier 0 interchange has the possibility of connecting to 1023 other interchanges. Each of the Tier 1 interchanges can provide access to 1023 interchanges; this results in a total of 1,046,529 possible

Tier 2 interchanges. Each of those can provide access to 1023 salve units, resulting in a possible 1,070,599,167 total slave units. Extending the tier system to its fullest extent of 255 tiers would result in 1023^{255} number of possible slave units.

While it may not be practical to create a fully tiered system, in practice, interchanges and slave units can be tiered in any combination at any tier. Each interchange shown in Figure 5 may assign its CHAN ID values to any combination of internal memory devices or PACS® ports connecting to the next tier, whether they are slave units or interchanges. As always the master unit has finally responsibility for knowing how each of the devices in the system are intended to operate, and each of the slave units and interchanges must specify their functions.

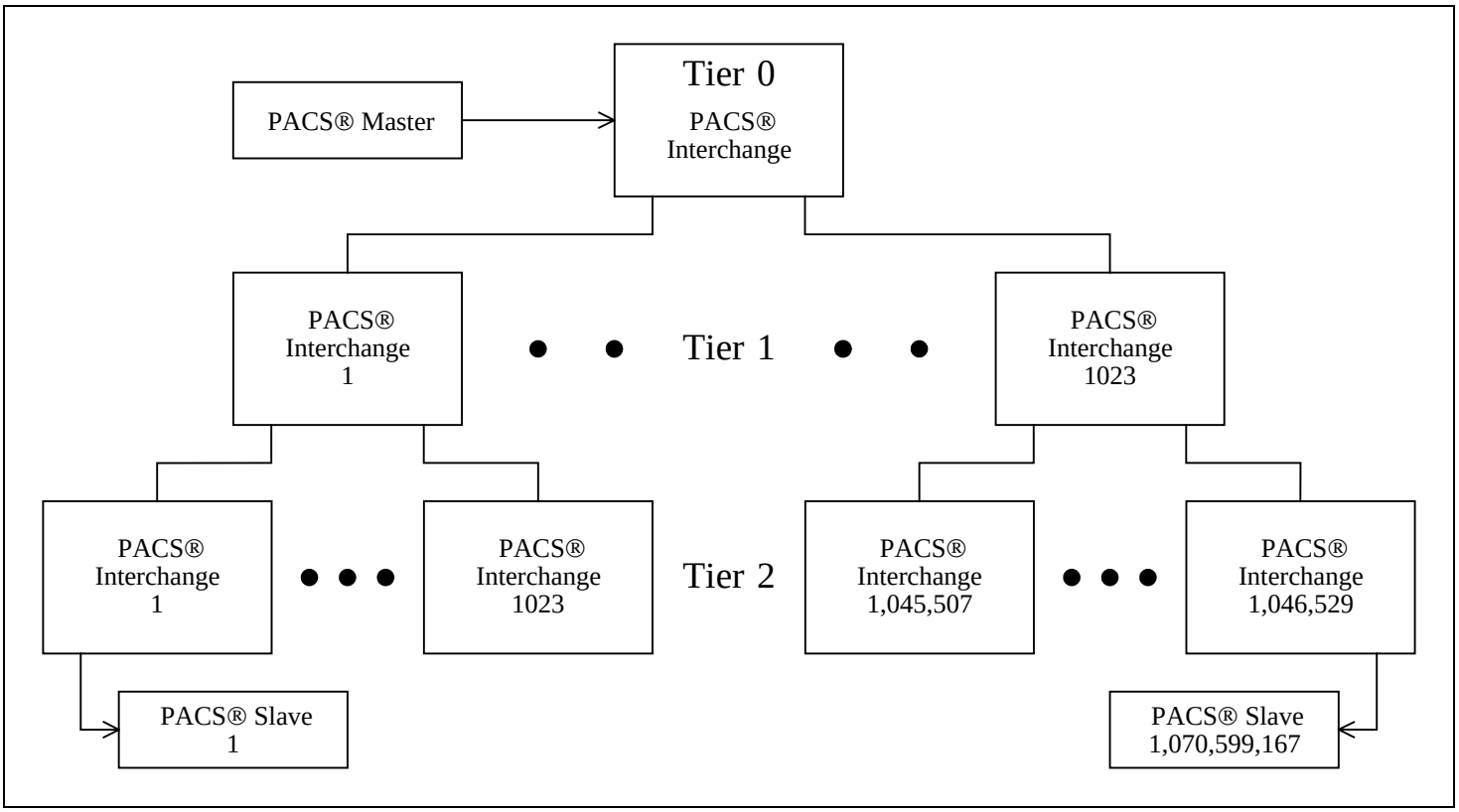

Figure 5 – Tiered Interchanges

LEVEL COMMAND

The LEVEL command instructs a slave unit to return a single byte that represents the level of PACS® incorporated within the slave device. This is useful to a master in order to determine what commands the slave device will execute. Unlike the READ command, the LEVEL command is a direct response from the slave device, and therefore has no address associated with it, and the slave's index pointer is not changed as a result.

Specifically, a Level 1 slave device will perform no operation in response to a CHAN ID or TIER command. Consequently, all following commands from the master will be performed on the only "memory bank" contained in the Level 1 Slave.

A Level 2 slave device will execute an operation in response to a CHAN ID or TIER command. The Level 2 slave must clearly specify what operation is performed for all the possible combinations of the CHAN ID and TIER commands. For example, any one of the PACS® Interchanges shown in Figure 5 could have only 8 channels, instead of the maximum 1023. In this case, the device must specify what operation occurs when it receives a CHAN ID command for channels 9 to 1023.

No PACS® device is considered to be level 0. If the master receives a value of 0, it will treat this as an error and assume the device is only a Level 1 slave. Levels 3 to 254 are reserved for future corresponding levels of PACS®.

Level 255 (FF_{16}) is reserved for a special response from a PACS® Interchange. In a tiered system, such as the one in Figure 5, unnecessary delays could occur if a device goes off line or encounters an error. To avoid such delays, a common practice for interchanges is to return all 1's in response to READ commands for channels that encounter an error, go off line, or simply do not exist. However, a data value with all 1's might be a valid value for the

location being read. The LEVEL command allows the master to determine whether the data received from a previous READ command is suspect, since no PACS® device is considered to be level FF. The LEVEL command can then be used to quickly scan devices in the system to determine when a device is back on line.

Slave devices commonly implement a timer that limits the amount of time the master has between command strings. If the specified time period is exceeded, the slave device will then switch to a specified default operation. The LEVEL command is useful for keeping such timers from timing out, since the LEVEL command does not affect any specific address or the index pointer.

THE NO OPERATION STANDARD

A key to maintaining organization in a communications system is to be able to distinguish good information from bad information. The PACS® hardware already includes a number of methods of discriminating data. These methods provide both master and slave with the capability to know when errors have occurred.

The No Operation Standard derives itself from this process. Simply put, the slave will perform no operation when an error condition is detected and will reestablish communications under the direction of the master; the master, having knowledge of such errors, can take whatever action it deems necessary to compensate for the error, such as retransmitting the command.

In some cases, a slave may receive a command sequence that does not result in any hardware error. However, it may not "understand" the command. For example, a PACS® Level 1 slave may receive a higher level command from the master. To insure that the slave does not perform an undesired operation, the slave will abide by the NO Operation standard. Since the master will not detect an error in this situation, the mater may need to "test" the results of the command prior to assuming that the slave will understand the operation. Alternatively, the master would know that the slave is at a lower level, or may use the LEVEL command to determine the slave's level, and therefore will avoid using the higher level commands. To insure compatibility between various PACS® levels, higher level PACS® command sets are required to include all lower level commands.

A final condition that results in no operation is more deliberate. Several command bytes have been specifically designated as a NOP (No Operation) command. Command byte values of 00, 03, 05, 07, 09, 0B, 0D, and FF are the NOP commands and are typically used for initial communications during a power up sequence in order to "synchronize" the master and slave. The FF command will result in an eight byte sequence, where the additional bytes have no special requirements and are ignored by the slave. The other NOP commands are single byte commands with no additional bytes in the command sequence.

The 00 NOP is inherently the fastest PACS® command and can be useful for keeping slave timers from timing out. Like the LEVEL command, the NOP command does not affect any specific address or the index pointer. Generally, the NOP command is better suited for this purpose when communicating with a Level 1 Slave. Since a Level 2 Slave could be an interchange, the LEVEL command is better suited, since it can also monitor if a given channel is on line.

If the slave performs no operation for any reason, it will not change its index pointer. It is important for the master to know that no operation has occurred whenever the indexed addressing mode is used. The master may choose to transmit a direct address for one command in order to establish the index pointer in the slave. Various commands can be used to change the index pointer without changing data in any location, such as the READ command or by adding zero to a location with the ADD command. The master may also want to test the results of a series of indexed commands. For example, the master could read the last location expected to be changed by a series of indexed commands to confirm that it contains the expected data.

SECTION III
The ASCII Translator

The PACS® format described in the previous section discusses PACS® in its true form as binary codes. While this information is useful to someone who is designing or debugging hardware, it does tend to be cumbersome for someone who needs to write programs in a PACS® system. To aid the programmer, the ASCII Translator format is provided. The same way an assembler aids a programmer in writing programs for microprocessor machine codes, the ASCII Translator provides a standard format that allows programming to be done with abbreviated alphanumeric codes.

The ASCII Translator can take on different forms. For example, the translator may be part of a development system where programs are encoded into PACS® codes and then placed on disk or EPROM for its final application. In other cases, the ASCII Translator may be part of an interface module that accepts another form of communications, such as RS-232, and converts incoming ASCII characters into PACS® codes for an outgoing PACS® Master port. This method however is less efficient due to the number of ASCII characters, or bytes, that are required to generate more efficient PACS® code.

PACS® COMMAND

Information processed by the ASCII Translator must follow a certain format and uses certain control characters to identify the meaning of characters. The general structure of a command sequence is illustrated in Figure 6. The PACS® command is represented by a one or two character abbreviation. A complete list of the ASCII commands can be found in APPENDIX A. The translator will convert the one or two character code into a single command byte.

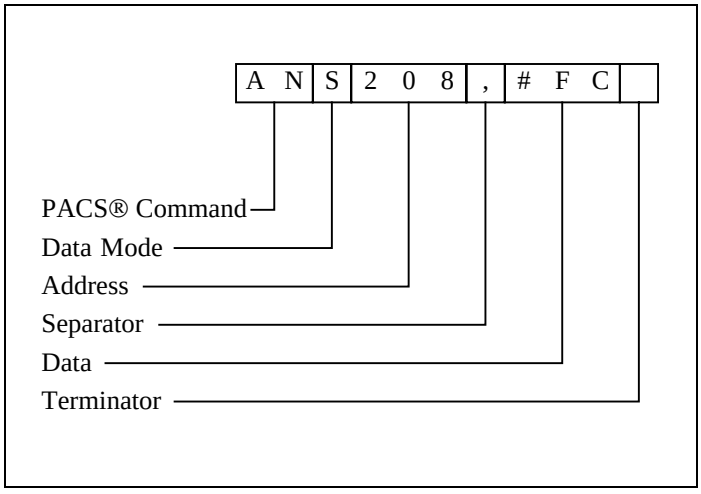

Figure 6 – ASCII Translator Format

DATA MODE

The character that follows the command characters defines the data mode for the translator. The translator will use this character to further define the command byte. So,

the translator will actually convert two or three ASCII characters into a single PACS® command byte. To define the data mode, one of three characters is used:

S - SINGLE byte
D - DOUBLE byte
Q - QUAD byte

ADDRESS

Once the PACS® Command and data mode are defined, further definition of the PACS® Command is required; the address mode must be defined. Whether a command will be direct addressing or indexed addressing depends on whether or not the address information is supplied. If direct addressing is desired, then the address is supplied after the data mode character. If indexed addressing is desired, then the address information is not supplied.

The address information can be provided in one of two formats, decimal or hexadecimal. Decimal address information can be any value from 0 to 65535. This means that the address could be one to five ASCII characters in length. To distinguish between decimal and hexadecimal addresses, the ASCII character "#" will follow the data mode character to denote that the following characters will represent a hexadecimal address. The address can be any value from 0 to FFFF. Hexadecimal addresses, therefore, can be two to five characters long. Leading zeros can be supplied; however, the translator will ignore them. In any case the translator is not required to accept more than five characters for the address field.

SEPARATOR

Since address values can vary in length, a comma is used to mark the end of the address characters. The comma is called a Separator. The Separator will separate characters that define an address from characters that define the data (see Figure 6). Basically all characters that precede the separator are used to generate the PACS® Command byte and the two address bytes (if needed) for the command sequence. When the translator sees the separator, the previous characters are evaluated and the PACS® Command and address bytes are generated.

DATA

Following the separator, data characters (if required) should be supplied. As with the address information, the data can be supplied in two different formats, decimal and hexadecimal. To designate a hexadecimal data value, the ASCII character "#" will precede the actual data (as illustrated in Figure 6).

When supplying the data characters, it should be noted that the data mode has already been established; therefore, the provided data must correlate with the command used. The table below shows the value limits for the various data modes – the commas and spaces are not part of the data value, but are only shown for easier reading of the larger values. Leading zeroes can be supplied, but as with the address value, they are ignored. The translator in this case is not required to accept more than ten ASCII characters for a data value.

DATA MODE	DECIMAL DATA LIMITS	HEXADECIMAL DATA LIMITS
SING	0 to 255	0 to FF
DOUB	0 to 65,535	0 to FF FF
QUAD	0 to 4,294,967,295	0 to FF FF FF FF

Some commands do not require data. In this case, no data characters are supplied. Since there are no data characters, the separator is not needed, and therefore is omitted. The READ command does not require the programmer to supply data; however, the programmer will expect data to be returned. To specify the returned data in hexadecimal form, the separator is used, followed by the ASCII character "#". If the separator is omitted, then the returned data will be in decimal form.

TERMINATOR

The terminator clearly marks the end of the ASCII string that is to be converted into a PACS® Sequence. The terminator can either be an ASCII "space" character or a "CR" (carriage return) character. The character that follows the terminator will be interpreted by the translator as the next PACS® Command character. Commands that do not require data will have the terminator character follow the address value.

CHAN ID COMMAND

Since the CHAN ID command does not use address information, a special format is provided for an ASCII Translator. The format simply requires the user to provide the CHAN ID number desired followed by a terminator. Since all other commands start with an alpha type character, the translator will interpret any command starting with a numeric character as a CHAN ID command. The CHAN ID value given can be any decimal value from 0 to 1023. The translator is not required to accept hexadecimal values.

Leading zeroes can be supplied, but will be ignored; however, the translator is not required to accept more than four characters for the CHAN ID value. As

with all other commands, a terminator must follow the CHAN ID value to mark an end to the ASCII command string.

TIER COMMAND

The TIER command is also formatted differently from other commands since it does not utilize address or data information. The ASCII command code for the TIER command is the character "T" which is followed by the desired tier number (a value from 0 to 255). The comma separator follows the tier number to distinguish the tier number from the desired CHAN ID value which follows the separator. As with all commands, a terminator character must follow the CHAN ID value to end the command string.

The translator will ignore leading zeroes for the tier and CHAN ID values, but is not required to accept more than three characters for the tier value and four characters for the CHAN ID value. The translator is not required to accept hexadecimal characters for either value.

SYNTAX ERRORS

The format described in this section explains the basic requirements for an ASCII Translator. Whenever the translator is confronted with a character string that does not conform to the format, a syntax error occurs. In the event of such errors, the translator will ignore the entire string, and usually will provide a means of letting the user know of such errors. Syntax errors should be debugged from a user's program prior to implementing, especially in applications where indexed addressing is used. Following are some examples of syntax errors.

1. Invalid command character(s)
   BD123,34 ADS12,#0A
   should be:
   SD123,34 ANS12,#0A

2. Invalid/missing data mode character
   ANT12,0 AN12,#10D8
   should be:
   ANS12,0 AND12,#10D8

3. Missing "#" for hexadecimal address or data
   AD3F0,5 SD82,3C
   should be:
   AD#3F0,5 SD82,#3C

4. Data supplied with an inherent command
   DD45,12
   should be:
   DD45

5. Missing terminator between two strings. In this example, both commands result in an error.
   ANS80,#AASS81,2
   should be:
   ANS80,#AA SS81,2

Some errors are programmer errors where the translator will accept the string; however, the user obtains incorrect results. Examples of this type of error are shown below:

1. Missing address causing the command to be interpreted as indexed addressing.
   CD,34 should be **CD12,34**

2. Incorrect data mode causing the incorrect locations to be affected.
   CD12,1 should be **CS12,1**

SECTION IV
PACS® Command Set

This section is a complete listing of each PACS® command. Each variation of the command along with a summary of its operation is provided. Various symbols used in this section are defined below. PACS® codes are given in hexadecimal.

DEFINITION OF SYMBOLS

a - ASCII address value
d - ASCII data value
DIR - direct addressing mode
DOUB - double byte data mode (16 bit word)
IND - indexed addressing mode
QUAD - quad byte data mode (32 bit word)
SING - single byte data mode (8 bit word)

PACS® COMMANDS

ASCII CODE	COMMAND NAME
A	Add
AN	And
C	Change
D	Decr
EO	Ex Or
I	Incr
L	Level
OR	Or
R	Read
S	Sub
none	NOP
x	Chan ID *
T	Tier *

* Level 2 only

A ADD

This command instructs the PACS® Slave to **add** the binary value of the transmitted data to the binary value at the selected address. The result of the addition becomes the new value at the selected address. Carry, overflow, and two's complement addition are the responsibility of the user.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	64	2	1	ASa,d
DOUB	DIR	84	2	2	ADa,d
QUAD	DIR	C4	2	4	AQa,d
SING	IND	25	0	1	AS,d
DOUB	IND	45	0	2	AD,d
QUAD	IND	85	0	4	AQ,d

AN AND

This command instructs the PACS® Slave to logically "**and**" each bit of the transmitted data with the corresponding bit at the selected address. The result of the operation becomes the new value at the selected address.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	68	2	1	ANSa,d
DOUB	DIR	88	2	2	ANDa,d
QUAD	DIR	C8	2	4	ANQa,d
SING	IND	29	0	1	ANS,d
DOUB	IND	49	0	2	AND,d
QUAD	IND	89	0	4	ANQ,d

C CHANGE

This command instructs the PACS® Slave to **change** the value at the selected address to the value of the transmitted data.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	62	2	1	CSa,d
DOUB	DIR	82	2	2	CDa,d
QUAD	DIR	C2	2	4	CQa,d
SING	IND	23	0	1	CS,d
DOUB	IND	43	0	2	CD,d
QUAD	IND	83	0	4	CQ,d

D DECR

This command instructs the PACS® Slave to **decrement** the value at the selected address by one. A value of zero is decremented to all logical ones.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	58	2	0	DSa
DOUB	DIR	59	2	0	DDa
QUAD	DIR	5A	2	0	DQa
SING	IND	18	0	0	DS
DOUB	IND	19	0	0	DD
QUAD	IND	1A	0	0	DQ

EO EX OR

This command instructs the PACS® Slave to logically "**exclusive-or**" each bit of the transmitted data with the corresponding bit at the selected address. The result of the operation becomes the new value at the selected address.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	6C	2	1	EOSa,d
DOUB	DIR	8C	2	2	EODa,d
QUAD	DIR	CC	2	4	EOQa,d
SING	IND	2D	0	1	EOS,d
DOUB	IND	4D	0	2	EOD,d
QUAD	IND	8D	0	4	EOQ,d

I INCR

This command instructs the PACS® Slave to **increment** the value at the selected address by one. A value of all logical ones is incremented to zero.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR		2	0	ISa
DOUB	DIR		2	0	IDa
QUAD	DIR		2	0	IQa
SING	IND		0	0	IS
DOUB	IND		0	0	ID
QUAD	IND		0	0	IQ

L LEVEL

This command instructs the PACS® Slave to return a single byte value that represents its PACS® Code level. The index pointer value remains unchanged.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	none	1C	none	1	L

DATA BYTES indicates the number of bytes returned by the slave.

OR OR

This command instructs the PACS® Slave to logically "**or**" each bit of the transmitted data with the corresponding bit at the selected address. The result of the operation becomes the new value at the selected address.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	6A	2	1	ORSa,d
DOUB	DIR	8A	2	2	ORDa,d
QUAD	DIR	CA	2	4	ORQa,d
SING	IND	2B	0	1	ORS,d
DOUB	IND	4B	0	2	ORD,d
QUAD	IND	8B	0	4	ORQ,d

R READ

This command instructs the PACS® Slave to return the value at the selected address. The value at the selected address remains unchanged.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	50	2	1	RSa,#
DOUB	DIR	51	2	2	RDa,#
QUAD	DIR	52	2	4	RQa,#
SING	IND	10	0	1	RS,#
DOUB	IND	11	0	2	RD,#
QUAD	IND	12	0	4	RQ,#

DATA BYTES indicates the number of bytes returned by the slave.
ASCII codes omit the separator and # character for decimal values.

S SUB

This command instructs the PACS® Slave to **subtract** the binary value of the transmitted data from the binary value at the selected address. The result of the subtraction becomes the new value at the selected address. Borrow, overflow, and two's complement subtraction are the responsibility of the user.

DATA MODE	ADDRESS MODE	PACS® CODE	ADDRESS BYTES	DATA BYTES	ASCII
SING	DIR	66	2	1	SSa,d
DOUB	DIR	86	2	2	SDa,d
QUAD	DIR	C6	2	4	SQa,d
SING	IND	27	0	1	SS,d
DOUB	IND	47	0	2	SD,d
QUAD	IND	87	0	4	SQ,d

NOP NOP

This command instructs a PACS® Slave to perform no operation. The purpose of the NOP command is to allow the master to operate the PACS® lines without performing a specific operation. The seven data bytes that must follow the FF NOP have no significant meaning and may be any value desired. Note that there is no ASCII code assigned to this command.

PACS® CODE	DATA BYTES
00	0
03	0
05	0
07	0
09	0
0B	0
0D	0
FF	7

x (PACS® Level 2 only) CHAN ID

This command instructs a PACS® Slave or Interchange to enable the selected memory bank or communication **channel** to another PACS® Slave. A PACS® Slave may perform no operation in response to this command.

PACS® CODE	DATA BYTES	OPERATION
3C	1	Data byte = selected bank or channel
3D	1	Data byte + 256 = selected bank or channel
3E	1	Data byte + 512 = selected bank or channel
3F	1	Data byte + 768 = selected bank or channel

For ASCII code, any decimal value from 0 to 1023 that appears between two terminators will represent the selected bank or channel.

T (PACS® Level 2 only) TIER

This command instructs a PACS® Interchange to pass a CHAN ID command to its currently active communication channel. The second byte of the command sequence represents the selected bank or channel in the desired slave or interchange. The third byte of the command sequence, designated as **t**, represents the targeted tier where the desired slave or interchange is connected. A standard PACS® Slave may either perform no operation or use the tier and CHAN ID values to select additional memory banks. An interchange will respond to the TIER command as follows:

1. If **t** = 0, the interchange will execute the CHAN ID command defined by the 2nd byte (see table below).

2. If **t** > 0 and the active channel is 0, the interchange cannot pass the tier command on to itself; therefore, it may either perform no operation, or it may use the tier and CHAN ID values to select additional internal memory banks.

3. If **t** > 0 and the active channel is not 0, the interchange will decrement the value of **t** by one and then pass the modified command to the active channel.

PACS® CODE	OPERATION
5C	2nd byte = selected bank or channel
5D	2nd byte + 256 = selected bank or channel
5E	2nd byte + 512 = selected bank or channel
5F	2nd byte + 768 = selected bank or channel

The ASCII format for this command is **Tt,*x***, where **t** represents the targeted tier, and ***x*** represents the selected bank or channel.

APPENDIX A

PACS® Reference Guide

	PACS® CODE	ASCII	DATA MODE	ADDR MODE
A ADD	64	ASa,d	SING	DIR
	84	ADa,d	DOUB	DIR
	C4	AQa,d	QUAD	DIR
	25	AS,d	SING	IND
	45	AD,d	DOUB	IND
	85	AQ,d	QUAD	IND
AN AND	68	ANSa,d	SING	DIR
	88	ANDa,d	DOUB	DIR
	C8	ANQa,d	QUAD	DIR
	29	ANS,d	SING	IND
	49	AND,d	DOUB	IND
	89	ANQ,d	QUAD	IND
C CHANGE	62	CSa,d	SING	DIR
	82	CDa,d	DOUB	DIR
	C2	CQa,d	QUAD	DIR
	23	CS,d	SING	IND
	43	CD,d	DOUB	IND
	83	CQ,d	QUAD	IND
D DECR	58	DSa	SING	DIR
	59	DDa	DOUB	DIR
	5A	DQa	QUAD	DIR
	18	DS	SING	IND
	19	DD	DOUB	IND
	1A	DQ	QUAD	IND
EO EX OR	6C	EOSa,d	SING	DIR
	8C	EODa,d	DOUB	DIR
	CC	EOQa,d	QUAD	DIR
	2D	EOS,d	SING	IND
	4D	EOD,d	DOUB	IND
	8D	EOQ,d	QUAD	IND
I INCR	54	ISa	SING	DIR
	55	IDa	DOUB	DIR
	56	IQa	QUAD	DIR
	14	IS	SING	IND
	15	ID	DOUB	IND
	16	IQ	QUAD	IND

	PACS® CODE	ASCII	DATA MODE	ADDR MODE
L LEVEL	1C	L	SING	
OR OR	6A	ORSa,d	SING	DIR
	8A	ORDa,d	DOUB	DIR
	CA	ORQa,d	QUAD	DIR
	2B	ORS,d	SING	IND
	4B	ORD,d	DOUB	IND
	8B	ORQ,d	QUAD	IND
R READ	50	RSa,#	SING	DIR
	51	R#a,#	DOUB	DIR
	52	RQa,#	QUAD	DIR
	10	RS,#	SING	IND
	11	R#,#	DOUB	IND
	12	RQ,#	QUAD	IND
S SUB	66	SSa,d	SING	DIR
	86	SDa,d	DOUB	DIR
	C6	SQa,d	QUAD	DIR
	27	SS,d	SING	IND
	47	SD,d	DOUB	IND
	87	SQ,d	QUAD	IND
NOP	00	no ASCII		
	03	no ASCII		
	05	no ASCII		
	07	no ASCII		
	09	no ASCII		
	0B	no ASCII		
	0D	no ASCII		
	FF	no ASCII		
x CHAN ID	3C	0-255		
	3D	256-511		
	3E	512-767		
	3F	768-1023		
T TIER	5C	Tt,x (channels 0-255)		
	5D	Tt,x (channels 256-511)		
	5E	Tt,x (channels 512-767)		
	5F	Tt,x (channels 768-1023)		

APPENDIX B

ASCII Conversion Table

HEX	BINARY	ASCII	HEX	BINARY	ASCII	HEX	BINARY	ASCII
00	0000 0000	NUL	2B	0010 1011	+	55	0101 0101	U
01	0000 0001	SOH	2C	0010 1100	,	56	0101 0110	V
02	0000 0010	STX	2D	0010 1101	-	57	0101 0111	W
03	0000 0011	ETX	2E	0010 1110	.	58	0101 1000	X
04	0000 0100	EOT	2F	0010 1111	/	59	0101 1001	Y
05	0000 0101	ENQ				5A	0101 1010	Z
06	0000 0110	ACK	30	0011 0000	0	5B	0101 1011	[
07	0000 0111	BEL	31	0011 0001	1	5C	0101 1100	\
08	0000 1000	BS	32	0011 0010	2	5D	0101 1101	]
09	0000 1001	TAB	33	0011 0011	3	5E	0101 1110	^
0A	0000 1010	LF	34	0011 0100	4	5F	0101 1111	_
0B	0000 1011	VT	35	0011 0101	5			
0C	0000 1100	FF	36	0011 0110	6	60	0110 0000	\
0D	0000 1101	CR	37	0011 0111	7	61	0110 0001	a
0E	0000 1110	S0	38	0011 1000	8	62	0110 0010	b
0F	0000 1111	S1	39	0011 1001	9	63	0110 0011	c
			3A	0011 1010	:	64	0110 0100	d
10	0001 0000	DEL	3B	0011 1011	;	65	0110 0101	e
11	0001 0001	DC1	3C	0011 1100	<	66	0110 0110	f
12	0001 0010	DC2	3D	0011 1101	=	67	0110 0111	g
13	0001 0011	DC3	3E	0011 1110	>	68	0110 1000	h
14	0001 0100	DC4	3F	0011 1111	?	69	0110 1001	i
15	0001 0101	NAK				6A	0110 1010	j
16	0001 0110	SYN	40	0100 0000	@	6B	0110 1011	k
17	0001 0111	ETB	41	0100 0001	A	6C	0110 1100	l
18	0001 1000	CAN	42	0100 0010	B	6D	0110 1101	m
19	0001 1001	EM	43	0100 0011	C	6E	0110 1110	n
1A	0001 1010	SUB	44	0100 0100	D	6F	0110 1111	o
1B	0001 1011	ESC	45	0100 0101	E			
1C	0001 1100	FS	46	0100 0110	F	70	0111 0000	p
1D	0001 1101	GS	47	0100 0111	G	71	0111 0001	q
1E	0001 1110	RS	48	0100 1000	H	72	0111 0010	r
1F	0001 1111	US	49	0100 1001	I	73	0111 0011	s
			4A	0100 1010	J	74	0111 0100	t
20	0010 0000	SP	4B	0100 1011	K	75	0111 0101	u
21	0010 0001	!	4C	0100 1100	L	76	0111 0110	v
22	0010 0010	"	4D	0100 1101	M	77	0111 0111	w
23	0010 0011	#	4E	0100 1110	N	78	0111 1000	x
24	0010 0100	$	4F	0100 1111	O	79	0111 1001	y
25	0010 0101	%				7A	0111 1010	z
26	0010 0110	&	50	0101 0000	P	7B	0111 1011	{
27	0010 0111	'	51	0101 0001	Q	7C	0111 1100	|
28	0010 1000	(	52	0101 0010	R	7D	0111 1101	}
29	0010 1001	)	53	0101 0011	S	7E	0111 1110	~
2A	0010 1010	*	54	0101 0100	T	7F	0111 1111	DEL