

Basic Communications For The Relay Engineer

by

Graham Clough

GEC Alstom T & D
Vancouver
Washington

Tony Giuliani

GEC Alstom T & D
Hawthorne
New York

Jay Gosalia

GEC Alstom T & D
Hawthorne
New York

Presented to:

20th Annual Western Protective Relay Conference

Spokane, Washington,
October 19, 1993

List of Topics Covered

Basic Terminology
Essentials of Serial and Parallel Channels
Data Encoding
Synchronous and Asynchronous Systems
ISO Seven Functional Layers (Defined)
Formats and Protocols
RS 232 Hardware Specification
RS 485 Hardware Specification and FMO Encoding
HDLC Protocol
IEC 870 -5 FT1.2 Protocol
Data Security
(checksums)
(parity)
(Cyclic Redundancy Checks)
Appendix 1 - Table of ASCII Characters
Appendix 2 - Mechanics of Cyclic Redundancy Checks
Bibliography (Reference)

Introduction

Ten years ago, if a protection engineer had been seen talking to a relay, he would probably asked to make an appointment with a psychiatrist. Today, if you can not talk to a modern relay, then it is using outdated technology. In the space of a few years, protection engineers have begun to use terms previously reserved for *Those Communication People*.

Many excellent papers have been written over the recent past on communications between relays and within substations but most of these assume prior knowledge on behalf of the reader; knowledge which is not always possessed.

This paper tries to clarify some of the terms used and remove some of the mystique from the process of acquiring information from relays by use of digital communication links. It is written not for the communications expert but for the relay engineer, who would like to know more about the basics of this fascinating subject.

Most new power system protective relays possess interfaces to allow communication with computers.

The purpose of the communication is mainly for :

retrieval of : target information
(including fault location)
metering information
setting files
event and disturbance
records

changing settings

controlling circuit breakers

The communication can be local or remote, that is, a computer terminal can be used in the substation and connected directly to the relay, or the relay can be accessed by the use of an office computer and a telephone link by the use of modems.

The term, *modem*, is a contraction of MODulator DEModulator and this device converts the digital signals from the relay or computer to tone signals that can be propagated over a telephone network.

In a *full duplex system*, data can travel in both directions at once, but in *half duplex*, data can only flow in one direction at a time, rather like a polite question and answer session. A *simplex system* allows communication in only one direction.

Digital Communications

Digital communication can be considered to have three main aspects. Firstly, hardware connections are needed. The physical shape of these and the electrical characteristics, of the connection such as voltage levels, must be specified. Secondly, the flow of data must be controlled so that information is not transmitted at a higher rate than a receiver can accept and if data corruption occurs, there is some means of detecting it.

These two aspects are usually called *protocols* or *formats*.

The third aspect involves the coding of the data so information can be intelligently sent. This aspect is known as *language*. All of these are covered in the paper.

Data is transmitted on an electric circuit using a series of pulses or bits. The word Digital is intended to mean a Two State System. Either a signal is there or not. We communicate in a series of 1's and 0's, a 1 being represented one electrical

state and a 0 being represented by another state. These states are usually specific voltage levels and can have positive, negative or zero values. Different formats are mentioned in the paper. The word *bit*, is a contraction of BINARY digiT. A group of eight bits is called a byte.

The two basic arrangements used, are *Serial* and *Parallel*. In a fundamental serial channel, the series of pulses will resemble that shown in figure 1.

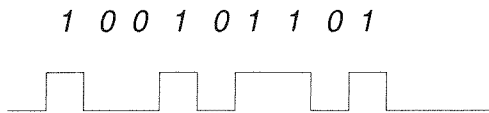


Figure 1. A Train of Serial Digital Pulses

Figure 2 shows the basic configuration of an 8 bit parallel channel. Only the data lines are shown: in practice, various control lines are also needed.

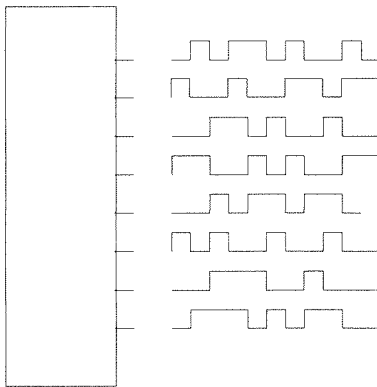


Figure 2. Basic Parallel Communication

The advantage of the parallel interface is speed. Its disadvantage is the number of conductors needed in the connecting cable.

This paper concentrates on serial communication, for, although it is slower than parallel, the fewer number of conductors needed makes it most attractive for distances over a few feet and it can be interfaced with a telephone system by use of a modem.

The minimum number of conductors needed for a full duplex system is three, two data wires and a common. The information in the waveform is arranged in a time dependent series of bits of data each having a specific time duration. The standard formats allocate a time interval known as the *baud period* of each bit. The word *baud* is derived from the name of a Frenchman, M. Baudot, who studied various means of encoding data in the nineteenth century. The baud rate of a simple system can be assumed to be the maximum number of bits that can be transmitted per second.

To send information along a serial channel, there are two basic requirements.

1. The data must be properly encoded
2. The transmitter and receiver must be time aligned.

Data Encoding

Data is transmitted as a series of pulses or bits which can have a value of either 0 or 1. These bits must be organized in such a way that the receiver can understand what has been sent. Because we are mainly interested in transmitting text, letters, numerals and some symbols, a group of bits can be made to represent a character. When put into groups, the number of discrete characters that can be sent, depends on how many bits represent each character.

If a group contains only five bits, the number of combinations ranges from 00000 to 11111. This represents a range of 32 decimal characters. Other ranges are shown in table 1.

Number of bits representing a character	Possible number of different characters
5	32
6	64
7	128
8	256

Table 1

The most commonly used code for text, is ASCII, or American Standard for the Interchange of Information. This was generated by ANSI as the ANSI X3.4-1968/1977 standard. All of the letters of the alphabet (lower and upper case), all numerals and many symbols are encoded in the first 128 combinations, requiring only seven bits. A listing of these codes is given in appendix 1. Extended ASCII, using eight bits is also used, but the upper 128 characters have not been standardized and vary from one computer manufacturer to another.

Binary Coding

If much numerical data is to be transmitted, ASCII is very inefficient. For example, if the current in a circuit is 225 amps, the number 225 would need three bytes in ASCII code. However if the numbers are sent in binary form, then the decimal number, 225 would be the binary number, 1110 0001, ($2^7+2^6+2^5+2^0$) the least significant bit, (LSB), being on the right. So when many numbers are to be transmitted in a series, the receiver will be instructed by ASCII characters to interpret the next n bytes in binary format and then revert to ASCII.

The use of binary coding makes it very desirable, therefor, to specify eight data bits rather than seven, so that binary numbers

can be sent in complete groups of eight (one byte).

In order to distinguish characters, the receiver must know several things including, where a character starts.

Data can be transmitted using a synchronous or an asynchronous system. In a synchronous system, the bit periods of the transmitter and receiver are controlled by clocks, both of which are synchronized. In an asynchronous system, the clocks in the receiver and transmitter run at **approximately** the same rate.

Asynchronous Systems.

When a character is transmitted in asynchronous systems, an extra bit is put at the front of the group of bits. It is called the *start bit*. It causes the receiver to achieve approximate synchronism with the transmitter for a time long enough for at least ten bits (one eight bit byte, the start bit and the stop bit) of data to be successfully clocked. Each character, or group of bits, is therefor transmitted independently of all other characters.

A slight drift in the transmit and receive frequencies will not be important during the time taken for the transmission of one byte.

During the time between characters when no data is being transmitted, the data channel is usually at a steady high voltage called a *mark*. A *space*, is the converse of a mark, being, for example, zero voltage or a voltage of opposite polarity to the mark. When the transmitter receives the start bit which is a 0, or space, it knows that it must start to look for marks and spaces at a predetermined rate. This is shown in figure 3.

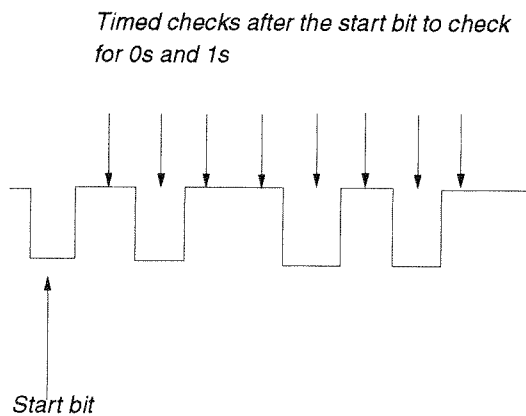


Figure 3. Receiver Timing For A Character

The way that the receiver stays in step with the transmitter is interesting. It is usual for the clock frequency of the receiver and transmitter to be sixteen times the baud rate. When the edge of the start bit is received, the receiver will wait eight clock periods, and then look at the waveform at intervals of sixteen clock periods. The receiver will therefore look at approximately the middle of each bit so even if the receiver clock and the transmitter clocks operate at slightly different frequencies, the receiver can correctly decode the character. This is shown in figure 4.

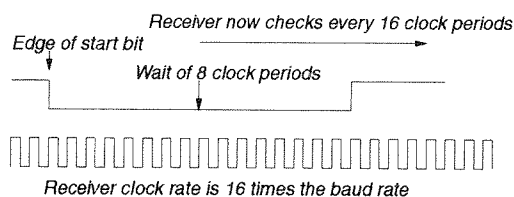


Figure 4. Receiver Synchronization

Format And Protocols

The International Standards Organization, ISO, has identified seven functional *layers* for communication. [3.]. These are not layers in the usual sense of the word, but relate to levels of

complexity and purpose. ANSI is the US representative to ISO.

These layers are essentially protocols, or rules for carrying out functions in an effective, efficient and standardized way.

In descending order, these levels or layer, are:

Application (7), Presentation (6) and Session (5) Layers

These are three layers involved with with high level functions.

Transport Layer (4)

This layer ensures high quality network service involving reliable end to end message transport, multiplexing, flow control and message sequencing.

Network Layer (3)

This layer controls the ordering and switching of packets and routing across the network.

Data Link Layer (2)

The data link layer is responsible for controlling the transfer of information from one point to another point in a communication link.

Physical Layer (1)

The physical layer represents conventions or protocols covering the hardware aspects and electrical signals of communications.

Protocols relating to the last two of these, Data Link and Physical, are covered in the paper.

The word *format* tends to be used at character level, and *protocol* at higher levels of character or data block exchange.

Format and *protocol* simply refer to a set of rules and define the essential features of a communication channel. Both the transmitter and receiver must be set to the same protocol values and baud rate for successful communication.

We have seen that in asynchronous systems, the receiver must know where the character starts and this is defined by the start bit. There are other essential things that must be specified. In the simplest format, these are:

- baud rate
- number of stop bits
- number of data bits
- parity

Parity is a simple type of error checking and will be covered later.

During idle time (no data transmission), as stated above, the transmit line is held high. The first zero bit indicates to the receiver that a character is about to be received. The least significant bit is the first bit of the character. After the set number of character bits has arrived, the next bit can be used as the *parity check*. This is a simple error checking system which can detect corruption of the character by interference. Again, the receiver must know in advance whether to expect a parity bit in each character or not. If corruption due to interference is not likely, then parity can be set to off and ignored. This results in a slightly lower overhead, as the number of bits in each message may be less.

If parity is set to be *EVEN*, then if the number of 1's in the byte is an odd number, the parity bit will be 1, making the *total* number of 1's, even and if the total number

of 1's is already even, then the parity bit would be a 0.

Conversely if parity is set to *ODD*, then, if the number of 1's in the byte is an odd number, the parity bit will be 0 and if the number of 1's in the byte is even, then the parity bit will be set to 1, making the total number of 1's in the overall block, an odd number.

An example of a complete character is shown in figure 5. The ASCII code for the letter C, is 67 in decimal and in binary form is 100 0011.

The first 1 (right hand side) is the least significant bit,

$$\text{being } 1 \times 2^0 = 1 \text{ decimal.}$$

The second 1 is $1 \times 2^1 = 2 \text{ decimal}$

The last 1,

in position 7 is $1 \times 2^6 = 64 \text{ decimal}$

Total = 67 decimal

= 043 hexadecimal

Again a choice must be made. In which order will the character be sent?

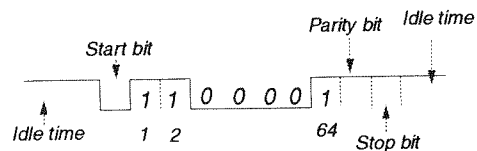


Figure 5. ASCII Representation of the Letter C

The decision, least or most significant bit, is usually made by the manufacturer of the equipment and as long as the transmitter and receiver are compatible, the user need not be concerned.

It can be seen that the parity has been set to even in the example, because the parity bit is 1 and there are three 1's in the

ASCII character. If any one of the bits in the character had been corrupted, then the receiver would detect the incorrect parity. The receiver would then either raise an alarm or send a request to the transmitter to re-send that character. The parity bit is positioned next to the stop bit, so that the receiver can remove it and a correct ASCII character will remain.

The last bit in the message is a *STOP* bit. This indicates to the receiver that the group is complete. In some systems, the number of bits in a group is specified as ten. Where this is so and the choice has been made to not use the parity bit and seven bit ASCII is being used, then two stop bits are needed. The specification for ten bits in a data block results from a convention that expects the transmitter and receiver clocks to time within $\pm 5\%$ of each other. If the tolerance is just better than $\pm 5\%$, then 10 bits of data are the maximum that can be transmitted and received correctly, before timing errors start to occur.

The scheme outlined above is usual for low speed communications involving relays, and is used with the hardware system known as RS 232.

RS 232 Interface

The signal processing in computers and digital relays is carried out using a parallel bus structure. The parallel digital signals are converted into asynchronous serial data streams by the use of a Universal Asynchronous Receiver Transmitter integrated circuit, or *UART*. This chip operates at about +5V dc.

The RS-232E standard, which is probably the most common serial interface in use, was developed by the Electronic Industries Association and published in 1969. This hardware protocol can be used synchronously or asynchronously.

The letters RS mean Recommended Standard and the E is the latest revision letter. The standard specifies the hardware connections shown in figure 6 and also specifies what voltage levels represent 0's and 1's. RS 232 is a full duplex system, that is one in which data can be transmitted in both directions at once.

RS 232 PIN DEFINITION	COMPUTER OR RELAY PIN NUMBER (DTE)	MODEM PIN NUMBER (DCE)
Signal Ground	1	1
Transmit data	2	3
Receive data	3	2
Request to send	4	5
Clear to send	5	4
Data set ready	6	20
Chassis Ground	7	7
Carrier detect	8	8
Data terminal ready	20	6

Figure 6. Common RS 232 Connections

The RS 232 document also specifies that the term *Data Terminal Equipment*, DTE, refers to what is connected at each end of the communication path, sink or source and *Data Circuit-terminating Equipment*, DCE, is equipment for establishing, maintaining, terminating or for performing signal conversion or coding.

The knowledge of this does not help in achieving good communications but may help in understanding some of the literature. More information is in reference 6.

Although nine terminals are commonly used and twenty five have been specified: communication can, however, take place with only three connections, transmit,

receive and signal ground. The other lines are designed to control the flow of data when, for example, a modem is connected. The process of controlling the flow of data is called *handshaking*.

Handshaking

Data is stored in a buffer in the transmitter, before it is sent and the receiver will have a similar buffer. If the data are transmitted faster than the receiver can accept, then its buffer will start to fill up and if no action is taken an overflow will occur and data will be lost.

Handshaking allows the receiver to tell the transmitter that its buffer is close to full, in which case the transmitter will stop transmitting until the receiver tells it to recommence.

The handshaking can be done using the hardware lines, Clear To Send (CTS) etc., or the control can be in software using Xon/ Xoff control. Again the transmitter and the receiver must both be set to operate in these modes.

An example of Xon/Xoff is a relay which is receiving data from a computer and its buffer is say, 70% full. At this point the relay might send a signal to the transmitter, Xoff. This is coded in ASCII as decimal code 19 (binary 0010011) and can be generated on a keyboard by ^Q (control and Q). When the buffer contents have fallen to say 30% full, the receiver will be ready for more data and it then informs the transmitter by sending the character Xon. This is ASCII character 17 (binary 0010001) and can be generated on the keyboard by ^S (control and S). Other techniques can also be used, such as character and line pacing, where delays between lines of text or characters are intentionally included, to ensure that the receiver buffer will not overflow. The

number of conductors can be three (send, receive and ground - see figure 6.) .

RS 232 Hardware

The usual hardware interface is a 25 pin D (DB 25) connector, but frequently a 9 pin D connector is used, when the functions provided by all of the control lines are not needed. (see figure 6). However, use of the 25 pin connector, does allow +12 and - 12 volt power supplies to be connected between equipment and indeed in the actual standard, all 25 pins have functions ascribed to them. However the ± 12 volt connections and the DB 25 connector are NOT to the RS 232 standard. Strictly speaking, any 25 pin connector can be used. [3]. The signal levels for the RS 232 interface are specified as:

RS 232 Signal	Representative Voltage Level
Logic level 1 (data)	-3 to - 15 volts
Logic level 0 (data)	+3 to +15 volts

The control lines use positive logic, that is logic 1's are positive and logic 0's are negative. The higher voltage levels, ± 15 volts rather than the +5 and 0 volt output voltage of the standard UART, allow transmission distance to be greater and give more noise immunity. An RS 232 data link is rated for only 50 feet at a maximum baud rate of 19,200 but can be extended for over 100 feet in practice, especially, at lower baud rates. Further limitations are the 2,500pF capacitance specification and the driver chip slew rate of 30 V per micro second. Slew rate is the maximum rate of change of output voltage that the actual driver chips are capable of.

Summary of RS 232 Characteristics

(From the EIA Standard)

Number of Transmitters and Receivers allowed on one line	1 Transmitter 1 Receiver
Mode of operation	DC Coupling
Maximum distance	50 feet
Maximum Data Rate	20,000 bits/sec
Transmitter voltage	± 15 V max ± 5 V min
Driver Slew rate	30 volts per micro sec
Receiver Sensitivity	3 volts

One of the main limitations is distortion of the pulses due to the capacitance of the transmitting cable. Figure 7 shows how the shape of square voltage pulses changes with increased distance (cable capacitance).

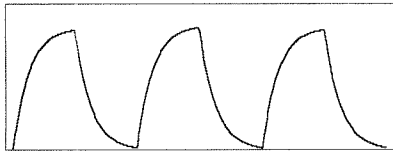


Figure 7. Distortion Of Square Waves Due To Cable Capacitance

A useful alternative to RS232, is the RS485 format.

RS 485 Interface

The RS 485 interface is very different from the RS 232. Only four conductors are specified for full duplex operation. With two conductors, half duplex operation only is possible. Relays that use this sort of communication must use a question and answer, technique, i.e., a request for information is transmitted and the

transmitter then waits for a reply. This appears, at first sight, to be a disadvantage, but this specification has many strengths.

- 1) RS 485 operates in a differential mode and with suitable data encoding, FM0 or FM1, allows the interface with the relay to be made by a small transformer.
- 2) 32 transmitter/ receiver pairs can share the same data channel.
- 3) It can operate over a distance of about 4000 feet.
- 4) The maximum data rate is 10 million bits per second
- 5) Only a shielded pair of conductors is needed.

Summary of the Salient Characteristics of the RS 485 Standard

Number of Transmitters and Receivers allowed on one line	32 Transmitters 32 Receivers
Mode of operation	Differential
Maximum distance	4000 feet
Maximum Data Rate	10 million bits/sec
Transmitter voltage	1.5V min
Receiver Sensitivity	300 millivolts

Bit Detection by the RS485 Receiver With FM0 Coding

Full use of the RS 485 hardware specification can be made by employing a format known as FM 0. This is shown in figure 8. The 1's are transmitted effectively at half the frequency of 0's. The value of the bit, 0 or 1, is determined by the timing of the transition of the signal. If the change occurs at the end of a baud period, then the bit is a 1 and if the change is in the middle of the baud period, then the bit is a 0.

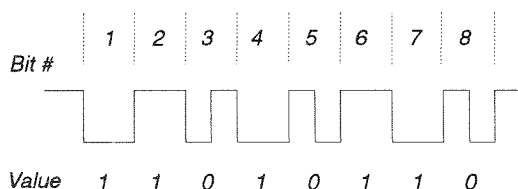


Figure 8. Reading Bits in FM0 Format

FM1 format is the converse of FM0: the bit edge for a 1 is in the middle of the baud period.

Data can be transmitted using the RS 485 specification in the same way as the RS232 system, asynchronously, as previously described but transmission rates can be much higher if data is transmitted synchronously and there is no limitation on device slew rate in the RS 485 standard.

RS 232 and RS 485 are known as *Physical Layer Protocols*.

Synchronous Systems

When synchronous systems are used, data can be sent in large blocks as well as in single characters. Synchronizing is done on RS 232 systems either by sending periodic synchronizing characters or using separate lines which carry a clock signal.

When synchronizing characters are used on either RS 232 or RS 485, a circuit known as a phase locked loop, or some other frequency tracking system, locks the frequency of the receiver clock to the transmitter clock. Data can now be sent in much longer continuous blocks. Thus *framing bits* are added to raw data bits in **asynchronous transmission** and *framing characters* are added to blocks of data for **synchronous transmission**.

Synchronous transmission has two main advantages, the actual transmission speed of the data can approach the baud rate more closely because it is sent in much larger blocks and the security of the data can be substantially improved. The implication of sending data in larger blocks is that the overhead of synchronizing bits for each character is replaced by a smaller overhead of synchronizing bytes for a large group of characters.

ISO Data Link Layer (Protocols and Languages)

A common problem experienced by users of relays which have the ability to communicate is the lack of a standard language to be used within a protocol. The *protocol* allows for the controlled transport of data between relays and computers and the *language* is what is contained within the data, the format of the data, how information is requested, how information is returned etc.. At the present time, each manufacturer has his own language and in many cases his own protocol.

The international organizations IEEE, IEC and CIGRE are considering the requirement of a standard language but it will be five to ten years before a standard emerges. Two protocols are described here, the IEC 870-5 and the HDLC.

These protocols are known as **Data Link Layer Protocols**.

The simple character oriented protocol previously described fits into the category of the data link layer but it is slower and less efficient than the frame oriented protocols described below.

The IEC 870-5 FT1.2 Protocol

The IEC 870-5 protocol can be used synchronously or asynchronously and the asynchronous structure, FT1.2 is described here.

A typical data packet consists of:

Start byte 0110 1000 (68 hex)
Length byte Number of bytes in the data
 field
Length byte repeated
Start byte repeated
Control byte
Address byte
Data packet up to 253 bytes of data
Checksum byte
End Byte 0001 0110 (16 hex)

The frame therefor consists of 261 bytes or approximately 2088 bits. The IEC 870-5 standard specifies RS232 as the physical layer, so at a rate of 19.2 kbits per second, a maximum of nine frames can be transmitted per second.

Because the IEC 870-5 FT1.2 structure is asynchronous, each byte of data in the data field is contained between a start bit, and a parity bit and stop bit. The start bit, as previously described, ensures time alignment between the receiver and transmitter for transmission time of one byte of information. This clearly slows the speed of data transmission but the structure has two advantages over simple character transmission. Firstly, the space, or delay time between bytes is controlled and

consistent and thus transmission is faster than simple character transmission, and secondly data security can be substantially increased using a two coordinate parity check or a checksum. This is described in the section on Data Security. When a system is running at a speed where there is no delay between the stop bit of one character and the start bit of the next, the system is said to be running at its *Cadence Speed*.

The HDLC Protocol

HDLC stands for High Level Datalink Control and this protocol is used with synchronous systems.

The basic HDLC packet consists of:

Opening flag 0111 1110
Address field 8 or more bits
Control field 8 or 16 bits
Data field any number of bits
Frame check 16 bits
Close flag 0111 1110

One function of the opening flag is to maintain synchronism between the transmitter and receiver. When no data is being transmitted, flag bytes are continuously transmitted.

The address byte is used to differentiate between two ends of a system so that it is possible to distinguish commands and responses [2].

The control field will contain commands which the receiver uses to keep track of the frames that it receives and to control the flow of information.

The data field can contain up to 1024 bytes of information. This information will be ASCII text, binary numbers, etc., in a language specified by the user.

The frame check field is dealt with in appendix 2.

Zero Insertion

An interesting technique is used in HDLC and other protocols, to allow discrimination between the opening and close flags and any data in the packet which is of the form 0111 1110. It is called *zero insertion*. The transmitter knows when it sends a flag, so when it detects more than 5, 1's in a row, it inserts a 0 after the fifth 1. When the receiver sees six 1's in a row, it knows that this represents a flag. When it sees five 1's in a row, it simply removes the next bit, if it is zero.

If part of the data is 011 1110 then the transmitter will transform this to 0111 1100, and when the receiver removes the first zero after the fifth 1, the data will be again correct.

Data Security

There are two main areas of concern relating to security. One is the corruption of data by noisy channels and the other is unauthorized access.

Detection of Data Corruption

Where relay communication is for the purpose of only gathering information, the resistance of data to corruption by interference is not vital. The parity bit in each byte of data, as shown in figure 5, is probably good enough. If parity is incorrect, then either an alarm is raised or the modem or relay can automatically request that the data is resent. Where relay settings can be changed remotely or the relay can trip a circuit breaker or perform other control functions, on receipt of a remote signal, more security is needed.

The resistance of data to corruption is called the *Hamming Distance*. This term honors the mathematician Mr. R W Hamming, who was the first person to really formalize error detection in communications.

A simple parity check has a hamming distance of 2, because it only detects corruption of 1 bit. The hamming distance is the minimum number of bits that must be changed to make the data acceptable again.

Two Coordinate Parity Check

In figure 9, ten, seven bit characters have been shown arranged in a vertical table. Each character has an additional parity bit shown in column P.

This parity bit is known as the *horizontal* parity bit, the *transverse* parity bit, the *lateral* parity bit or the *row* parity bit. Note that the parity is odd.

For the Block Check character at the bottom of the table, bits 1 to 7 constitute the parity bits respectively for columns 1 to 7 of the preceding ten characters. For example, the block check bit 1 is the parity bit for all the bit 1's of the preceding ten characters. Block check bit 2 is the parity bit for all the bit 2's of the preceding ten characters etc.. These parity bits are often called the *longitudinal* parity bits, the *column* parity bits or the *vertical* parity bits.

Note, in this case, even parity has been chosen for the column parity bits 1 to 7. Block check parity bit in column P is the row parity bit for the seven bits of the block check character. This row parity bit is odd because all the previous row parity bits were odd.

When the ten characters have been transmitted the block check character is sent.

Figure 9 shows how the block check character helps to detect additional errors.

In this figure a single bit error in character 2 $\{^1\}$ would be detected by both the horizontal and vertical parity checks.

The two errors in character 9 $\{^2\}$ would not be detected by a horizontal parity check but is picked up by a vertical parity check. However, the four bit error in characters 6 and 7 $\{^3\}$ would get through undetected.

The system is not foolproof but it does allow more errors to be detected. Since the maximum error detection capability is 3, its hamming distance is 4. Two coordinate parity checking is frequently used because it is relatively cheap to implement in electronic circuitry. Its main disadvantage is that it involves an increased amount of overhead in the form of one bit for each character plus an extra character at the end of each block of characters.

Bit #	P	7	6	5	4	3	2	1	Character #
	1	0	1	0	1	1	0	1	1
$\{^1\}$	1	0	0	0	1	0	0	0	2
	0	1	0	0	0	1	0	1	3
	0	0	1	1	1	0	0	0	4
	0	1	0	1	1	0	1	1	5
$\{^2\}$	1	1	1	1	0	0	0	1	6
$\{^2\}$	0	1	1	0	1	1	1	0	7
	1	0	0	1	0	1	1	1	8
$\{^3\}$	0	0	0	0	1	0	1	1	9
	0	0	1	0	0	1	1	0	10
	1	1	0	1	0	1	1	0	Block check

Figure 9. Detection of Errors by Two Coordinate Parity

Checksums

The checksum is generated by summing the value of each byte of data from the control byte to the end of the free data. This technique can be used in synchronous transmission where two coordinate parity cannot be used. That sum can of course be a very large number. For example if each byte consisted of 1's (255 decimal) and there were 255 bytes, the answer would be 65025 decimal or 11111110 00000001 binary. In practice, the eight least significant bits only are taken, giving a checksum of 00000001 for the example.

The hamming distance for simple checksums is 2.

Cyclic Checks

Cyclic codes are very much more complex than parity and checksums so only a simple example will be given here. In addition, although this process can be implemented in software, its complexity usually demands that hardware implementation in the form of a special chip, is used. Hamming distances of greater than 6 can be obtained but the proof of this demands complex mathematics.

The basic principle of a cyclic code is to divide the value of the bits to be checked by a constant. This will result in a remainder. This remainder is transmitted with the data. The receiver adds the remainder to the value of the number of bits and divides the result by the same constant: there will be no remainder if the message has not been corrupted. For those who are interested, the process is described in appendix 2 and covered in detail in reference 5 in the bibliography.

Protection against Unauthorized Access

A simple way of providing protection against unauthorized access is to use passwords. The relay will not respond if the correct password is not used. A further level of protection can be provided by the dial back modem. Modems are available, that can be set up so that no flow of data can be sent by the modem to the relay unless the modem has initiated the phone call. The procedure here, is to call the remote modem. This modem will then hang-up and call a specific telephone number and secure communication can then take place. Of course, if the language used within the communicated data is not known to an unauthorized person, it will be extremely difficult for him to obtain information and, more importantly, exercise remote control over substation equipment.

Appendix 1. Breakdown of the ASCII Character Set

It is probable that the original thinking in the generation of the ASCII code was to cover the twenty six lower case letters, twenty six uppercase letters and ten numerals. This gave sixty two different symbols. As this is close to the number of combinations provided by a group of 6 bits (64) and no punctuation symbols are included, it was probably decided to use a group of seven bits, giving 128 combinations so that lower case letters would not have to be sacrificed and room would be available for many punctuation symbols and characters that could be used for control purposes.

A table of the standard US ASCII characters is shown in figure 11. The character block shown in figure 10, below relates the positions of the characters in the table to their appearance in a bit stream.

The initials MSB indicate *most significant bit* and the initials LSB mean *least significant bit*. There are variations in ASCII for other countries which take account of special needs such as currency symbols and accents.

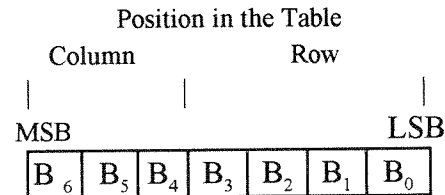


Figure 10 - ASCII Character Block

ASCII characters are divided into two major groups, graphic characters and control characters. The term graphic, means that the character is printable and a control character initiates, modifies or stops an action, that affects the transmission or interpretation of data. A detailed description of the ASCII character set is given in reference 3 in the bibliography.

Least significant bits (B ₃ B ₂ B ₁ B ₀)				Most significant bits (B ₆ , B ₅ , B ₄)							
				000	001	010	011	100	101	110	111
0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(8	H	X	h	x
1	0	0	1	HT	EM)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[k	{
1	1	0	0	FF	FS	,	<	L	\	l	
1	1	0	1	CR	GS	-	=	M]	m	}
1	1	1	0	SO	RS	0	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	-	o	DEL

Figure 11. A Table of Standard ASCII Characters

Appendix 2. Cyclic Redundancy Code

A stream of data bits can be represented in the form of a polynomial. For example the bits 11001 would be represented by

$$x^4 + x^3 + 1$$

and 10010 would be $x^4 + x$

The polynomial simply represents the powers of 2 that are present,

1	1	0	0	1
x^4	x^3	-	-	x^0
2^4	2^3	-	-	2^0

We will call this the *data polynomial* $D(x)$. We will define the polynomial which we will use for checking as the *checking polynomial* $C(x)$ and let it be $x^3 + x^2 + 1$.

If we shift the data polynomial by the degree of the checking polynomial, then divide the checking polynomial into the result, we will get the quotient $Q(x)$ plus a remainder, i.e.,

$$\frac{D(x).x^c}{C(x)} = Q(x) + \frac{R(x)}{C(x)} \quad \dots\dots\dots (i)$$

The remainder will be one degree less than the checking polynomial.

$$D(x).x^c = Q(x).C(x) + R(x) \quad \dots\dots\dots(ii)$$

Since addition and subtraction are equivalent in modulo 2 arithmetic,

$$D(x).x^c + R(x) = Q(x) . C(x) \quad \dots\dots\dots(iii)$$

If $D(x).x^c + R(x)$ is transmitted, and the receiver divides it by $C(x)$, if no corruption has occurred, the result will be $Q(x)$ with no remainder.

The actual polynomial used in the HDLC protocol is $x^{16} + x^{12} + x^5 + 1$.

Example of a CRC Check

The rules governing the division of binary numbers are based on exclusive OR logic (modulo 2). These are:

$$\begin{aligned} 1 + 1 &= 0 \\ 1 + 0 &= 1 \\ 0 + 1 &= 1 \\ 0 + 0 &= 0 \end{aligned}$$

If we multiply $x^4 + x^3 + 1$ by x^3 (shift it 3 places) we get:

$$x^7 + x^6 + x^3$$

If this is divided by $x^3 + x^2 + 1$,

The remainder will be:

$$\begin{array}{r} x^4 + x \\ x^3 + x^2 + 1 \overline{) x^7 + x^6 + + x^3} \\ \underline{x^7 + x^6 + x^4} \\ + x^3 \\ \underline{ + x^4 + x^3} \\ + + x \\ \text{Remainder} \quad x \end{array}$$

Thus, if x is added to $x^7 + x^6 + x^3$ and the result is divided by the checking polynomial $x^3 + x^2 + 1$, the remainder will be zero.

$$\begin{array}{r} x^4 + x \\ x^3 + x^2 + 1 \overline{) x^7 + x^6 + + x^3 + x} \\ \underline{x^7 + x^6 + x^4} \\ + x^3 + x \\ \underline{ + x^4 + x^3 + x} \\ + + + \\ \text{Remainder} \quad 0 \end{array}$$

This process is usually implemented in hardware. The chip for implementation will contain a shift register containing

exclusive OR gates (XOR).

Figure 12 shows the diagram of a circuit for the checking polynomial $x^3 + x^2 + 1$.

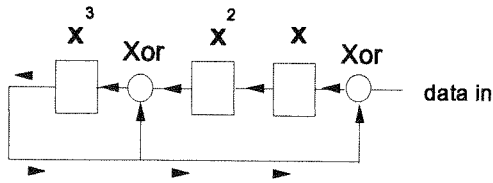


Figure 12. A Circuit for Generating the Polynomial $x^3 + x^2 + 1$

Figure 13 shows the data 11001

shifted by $x^3 = 11001000$

plus $x^1 = 10$

$= 11001010$

being clocked through the circuit shown in figure 12 and no remainder appearing. The logic that is implemented in each step of the process is as follows.

1. From the start to step 1, The content of cell c is clocked into cell b.
2. The content of cell a is XORed with the previous content of cell b and the result is put into cell a.
3. The content of cell a is XORed with bit d and the result is put into cell c.
4. The content of the input cell is shifted one bit to the left.

This process is carried out until all of the input bits have been clocked into the CRC circuit.

For each step in figure 13, the OR operation for cells a and b and for cell a and character d is shown. When the whole string of data bits has been clocked into the

CRC circuit, a check is made to see if there is a remainder. If a remainder is found, then corrective action is taken, for example a request is made by the receiver for retransmission of that packet of data.

This process is carried out on each packet of data as it is received, the CRC circuit being reset to 0 after each packet of data is processed.

	a	b	c	d	input cell
start	0	0	0	0	11001010
0 or 0 = 0 0 or 1 = 1					
step 1	0	0	1	0	1001010
0 or 0 = 0 0 or 1 = 1					
step 2	0	1	1	0	001010
0 or 1 = 1 0 or 0 = 0					
step 3	1	1	0	0	01010
1 or 1 = 0 1 or 0 = 1					
step 4	0	0	1	0	1010
0 or 0 = 0 0 or 1 = 1					
step 5	0	1	1	0	010
0 or 1 = 1 0 or 0 = 0					
step 6	1	1	0	0	10
1 or 1 = 0 1 or 1 = 0					
step 7	0	0	0	0	0
0 or 0 = 0 0 or 0 = 0					
step 8	0	0	0	0	

Figure 13. Operation of the CRC Polynomial $x^3 + x^2 + 1$

Bibliography

- 1) The IBM PC from the Inside Out. Murray Sargent III & Richard L Shoemaker
- 2) A Practical Guide to Computer Communications and Networking. Richard Deasington. - Ellis Horwood Publishers
- 3) Data Communications for Microcomputers. Nichols, Nichols and Musson. - McGraw Hill Book Company.
- 4) RS-232 Made Easy. Martin D. Sayer - Prentice Hall
- 5) Digital, Analog and Data Communication. William Sinnema - Reston Publishing Company.
- 6) Technical Aspects of Data Communication. John McNamara - Digital Press
- 7) Introduction to Data Communications and LAN Technology. Ed de Silva, BSP Professional Books
- 8) Related Papers - Data Collection and Control for Protective Relays, Garrity and Hedding , MIPSYCON 1990 and Integrating a Dedicated Relay Communication Network into a SCADA System, Driggans, Garity and Hedding, MIPSYCON 1992.