CONTROLLED DATA COMMUNICATIONS INTERFACING WITH REMOTE MICRO COMPUTERS

Preface

"If you can't dazzle them with brilliance, baffle them with something else!" most aptly applies to the title of this paper! What is controlled data communications? How does that apply to an interface? How is a remote micro computer defined?

How do you control datacom and what is it when you get it controlled? Data Communications is a means of transporting data thru the most unreliable and unpredictable means, generally a phone line, for use by the recipient. You control it by establishing reasonable precautions and checks at each end of the communication to affect a 97% (plus) reliability factor. Expect anything and everything when building in your precautions and you will not be disappointed. This paper will show how we did what we did and may give you some ideas for your situation. Beyond that, attend the presentation and ask your questions. As to what you have when you attain control over datacom, it turns out you end up with a decent tool to use to meet today's data processing requirements.

How does controlled datacom apply to an interface? If you can build a controlled datacom network --a reliable one-- you can use it to allow equipment to communicate thereby, easing a manual task or being more effective. If what you are doing is not ending in a more efficient and effective situation, don't do it.

For our purposes, a remote micro computer is described as a micro with limited intelligence situated in an environmentally unstable area, away from trained data processing personnel but accessable via a reliable datacom network. Quite a feat if you can accomplish it.

Section I - Introduction & General Description

E-Z SERVE, INC. AND SUBSIDIARIES

E-Z Serve, a petroleum refining and marketing company whose home offices are located in Abilene, Texas, has retail outlets thru most of the "sunbelt" states. The company got started in 1971 and utilized outside D/P services until 1980 when we purchased our first Hewlett-Packard computer: a series III. Since 1980 we have purchased two more series III's and a series 64 to accomodate our general data processing requirements. The original series III is used for system development and backup in that new software is created on this system and transported to other systems for production.

In 1981, E-Z Serve entered the micro world by virtue of Automated Fueling Systems. At that time few companies developed such equipment and fewer still understood the need for true flexibility in the design of their software. As a result, we developed a system of our own utilizing the Motorola 6809 CPU chip as a base. In order to do this we purchased a Hewlett-Packard 64000 Logic Development System to develop the software and aid in debugging and system analysis.

THE HEWLETT-PACKARD 3000

Our system design called for the HP 3000 to be utilized as a tool in communicating with our Fuel Site Micros. This provided quite a challange as our objectives called for the use of standard products only (see Section II - Objectives). With a fair amount of research we found that this was possible providing some minor allowances were made within the micro to allow flexibility in the data communications area. In doing the research we found that

within the Data Processing industry, Data Communications is the newest and most underspecialized field around today. I strongly recommend the Hewlett-Packard course entitled "Introduction to Data Communications" for anyone doing even a medium amount of work with datacom. Those of my people who have attended this course have shown a much greater understanding of the field upon completion of the course.

THE FUEL SITE MICRO

The Fuel Site Micro is a Motorola 6809 based microcomputer. This unit, located at the fuel site, handles the communication necessary to authorize purchases requested by a cardholder with an acceptable magnetically encoded debit or credit card. Control information is downloaded to the micro and stored transactions are uploaded to the host as a part of the interaction between the micro and the HP 3000. This interaction, what we learned and the sources available for information, make up the subject matter for this paper.

Section II - Objectives.

Our objectives were quite simple. We wanted to create a situation which required little or no manual intervention yet allowed us to control micros, as required, from a host computer. This host was to remain as standard as possible with respect of functioning within manufacturer's unaltered specifications and utilizing generally accepted data processing techniques and procedures. Peripheral equipment was to be utilized in an off-the-shelf fashion and again was to be a device generally accepted to the data processing community in a non-customized state. We recognized that customization cost more and wanted to avoid this unnecessary expense.

The majority of the code was to be done in an easily maintainable language with general subroutines for special functions written in a more detailed language (in this case SPL). The use of intrinsics was to be the exception rather than the rule. We wanted to utilize readily available software tools already on the market to relieve programming effort where possible. This was done thru some of the Quasar products.

Another important factor was data security. This was addressed thru the use of data encryption routines. A sample DES (Data Encryption Standards) routine is included in the Appendix. References are made to other sources available for this information. We also created our own encryption process and my recommendation to anyone is to do the same just for an added measure of protection. Your situation will be different and I recommend you expound these objectives for your use.

Relatively important to us was the issue of backup support. With our multiple HP 3000 processors we are able to duplicate software and data files at separate facilities in order to allow for continuous support in the event of system failures. We utilize DS/3000 to create and maintain the support files necessary at each site. This has allowed us to switch support from one facility to another easily and we can switch back to the original facility just as easily. Because we stipulated the use of standard features we can carry backup tapes to virtually any HP 3000 site with an autodialer modem and be in position to support the micros within 1 hour of arrival. This is important due to the need to provide reliability to the user of the micro.

It is easy to become enchanted with all the nice features available on equipment today. My best advice is to See It Big and Keep It Super Simple (SIBKISS). Keep the overall objective in mind but work at the simplest level possible on each step to that objective.

Section III - Line Control Techniques

Communications with the Fuel Site Micros is done with dial-up, asynchronous phone lines operating at 1200 baud and even parity. The relationship between the HP 3000 and the Fuel Site Micro is such that the HP acts as the master, initiating and directing the data transfer between the systems. The host computer interacts with the micro using ASCII commands and responses. Commands sent to the micro consist of a 'DC3', a two character command and any neccessary parameters followed by a 'DC1'. Since the FREAD intrinsic prompts with a 'DC1' only the characters up to the 'DC1' are written to the device in order to allow the HP to be ready for the response. In the case of multi-record responses the pacing

of each record is accomplished by the micro waiting for a 'null' command, consisting of a 'DC3' and FREAD's 'DC1', before sending the next record. In some cases, this transfer must be interrupted. If this interruption is to come from the host, a simple command is sent to the micro. Usually, an interruption will not be caused by the micro unless the phone connection is lost or the micro system goes down. In this event, the host programs recognize that fact and adjust themselves accordingly. Most of the data transfer is handled in this way since data is primarily sent from the micro to the host. In some instances we must download a large list of numbers to the micro. Transmitting this list can take as long as 2 hours. Since the characteristics of these numbers allows this list to be stored in a bit map, it could also be transmitted in this form. The bit map is sent in blocks of 2048 numbers as follows:

HP: 'DC3' CCnn 'DC1' where 'CC' is a two character command and 'nn' is the block number

micro: D or ? response indicating whether successful or not

HP: 'DC3' 'STX' hhh ... hhh 'ETX' cccc 'DC1' where 'h' is one of 512 hex digits, each one representing a nibble, and 'cccc' is the check digit

micro: D or ? response

This technique transmits the entire list in about 3 to 5 minutes or approximately 3% of the original time required.

During any typical access of the micros, the port is opened and closed several times as the host program calls each location. A method was needed to prevent multiple processes from using this line. One way would be to simply try to open the port; if the open is successful, proceed, if not, terminate the program and try again later. This approach, however, would not work in this case since, as mentioned before, the port would be opened and closed many times during one access and one program could sneak in and open the line while the first program is between phone calls. A more practical method is to use global Resource Identification Numbers (RIN's). Each program that needs the phone line, first trys to lock the RIN, then continues with all of its various line opens and closes, and, once it's finished, unlocks the RIN, freeing up these resources for use by other processes.

Section IV - Use of Hayes Smartmodem 1200

In our application it was neccessary to have requlary scheduled, automatically initiated communication with the Fuel Site Micros. This involved the use of a sleeper job on the 3000 with some kind of auto-dialer to make the calls. After reading a product review in Byte magazine, we decided to try out a Hayes Smartmodem 1200. The Hayes Smartmodem 1200 works with standard RS-232C in full or half-duplex at 1200 or 0-300 baud. It is compatible with Bell 103 and 212A modems in asynchronous mode only. Since this unit is a self-contained, stand-alone system, it required no additional equipment for use other than the computer cable. Connections with phone lines is with a standard modular telephone plug (RJ11, RJ12 or RJ13). One of the really nice features of this modem is the audio monitor included within the cabinet that can be very useful when trying to determine why a call is not being completed. On the front of the enclosure are the usual LED indicators (carrier detect, receive data, modem ready, etc.) with additional ones for auto-answer mode, and off-hook condition. The excellent documentation provided with the modem along with its reasonable price ($500 to $700) made this product very attractive to us.

Controlling the Hayes is accomplished by commands sent to it over the RS-232 line and an 8 position DIP switch. The modem provides a full compliment of simple ASCII commands with responses in English words or decimal digits. For instance, there are commands to turn the speaker on or off, control echo and half/full duplex, select default dialing modes (pulse or tone), set various configuration register values, and of course dial a phone number. The responses include 'CONNECT', 'NO CARRIER' and 'ERROR' or their numeric equivalents: 1, 3 and 4 (the selection of words or digits is also by a command.)

Using the Hayes with the 3000 is generally quite simple. Rather than requiring an explicit command to hang up the line you can choose to let the HP's dropping of RS-232C DTR signal instruct the modem to hang up the phone when the port is FCLOSEd.

Although the word responses are more explict and less likely to be confused with data, it is neccessary to use the digit responses because of the way the word codes are returned. Each word response includes a carriage return character both in front of and behind the word. This leading carriage return terminates the read of the response and the HP hasn't got

a chance of being ready for another read to accept the rest. The digit responses, however, have a carriage return only after the digit, making them much easier to be read by the HP. Another related item is that the Hayes does not wait for a 'DC1' to send a reply, so if you expect to receive any response you must give it something to do to keep it busy while the 3000 is getting ready for the read. In other words, any configuration commands can and should be included in the same command string as the dial command to give the HP enough time to receive the response.

Section V - Software Tool Development

Before writing application programs, a few tools had to be developed first, mainly subprograms to take care of the actual interface with the micro since most application programs would be written in COBOL. While developing the subroutines, we found a lot of useful information in Ross Scroggs and John Tibbets' articles in the 1982 San Antonio proceedings. Subroutines were written to open and close the line, read from and write to the micro, dial the phone, and logon to the Fuel Site Micro system. The following is a discussion, along with examples, of some of the subroutines developed.

FSMOPEN

The FSMOPEN procedure opens the port and sets up a few line characteristics. To provide flexibilty of use between our different systems and configurations, we used a device class name when opening the port rather than a device number since different systems had device classes assigned to different ports. Also to distinquish between our auto-dial port and our standard dial up port, we gave the auto-dial LDEV an additional device class name (AUTODIAL) along with the common name (DIALUP) (See the configuration listing in the appendix).

```
MOVE FILE'NAME := "FSMFILE ";
MOVE DEV'CLASS := "AUTODIAL ";
FSM'FILE := FOPEN (FILE'NAME,%604,%104,-540,DEV'CLASS);
IF < THEN
BEGIN
 FCHECK (FSM'FILE,FSM'STAT);
 GO TO EXIT;
 END;
```

The only other significant item to point out here is that the port is opened with CCTL in order to use the %320 carriage control parameter in writing to the FSM since the micro does not use carriage returns to delimit commands.

After the port is opened, we must set up the term type, baud rate, parity and echo attributes. A sample of these might be:

```
PARAM := %7411;                    << SET TERM TYPE = 9 & >>
FCONTROL (FSM'FILE,37,PARAM);      << BAUD RATE = 1200    >>
IF < THEN
BEGIN
 FCHECK (FSM'FILE,FSM'STAT);
 GO TO EXIT;
 END;

PARAM := 2;
FCONTROL (FSM'FILE,36,PARAM);      << SET EVEN PARITY >>
IF < THEN ...

FCONTROL (FSM'FILE,24,PARAM);      << ENABLE PARITY CHECKING >>
IF < THEN ...

FCONTROL (FSM'FILE,13,PARAM);      << TURN ECHO OFF >>
IF < THEN ...

FSETMODE (FSM'FILE,4);             << SUPPRESS CR/LF >>
IF < THEN ...                      << AFTER READ     >>
```

FSMCLOSE

All this procedure does, other than closing the line, is pause for the five seconds during which DTR is down after the file is closed before proceeding to the next phone call.

```
FCLOSE (FSM'FILE,0,0);
IF < THEN ...

INTERVAL := 5.0;
PAUSE (INTERVAL);
```

FSMREAD

When reading a programmatically controlled device you should take precautions to prevent your read from hanging, should the device fail to respond. The easiest, if not the only, way to do this is with timed reads.

```
TIME'LIMIT := 2;
FCONTROL (FSM'FILE,4,TIME'LIMIT);
IF < THEN ...

LEN := FREAD (FSM'FILE,FSM'BUF,-540);
IF > THEN
BEGIN
 FSM'STAT := 1000;       << INDICATES EOF TO CALLING PROGRAM >>
 GO TO EXIT;
 END;
IF < THEN ...
```

If the FREAD does time out, the calling program will be given the file system error (22) to handle as it needs to.

FSMWRITE

The calling program will pass to FSMWRITE the command minus the leading and trailing delimiters. The trailing 'DC1' comes from the following FREAD but the leading 'DC3' is inserted by this procedure before transmission. Along with the buffer to be sent, the calling program passes a length which follows the same rules as the length parameter in FWRITE intrinsic (positive -- word count; negative -- byte count).

```
IF FSM'LEN < 0 THEN      << CONVERT TO POSITIVE BYTE COUNT >>
 LEN := \FSM'LEN\
ELSE
 LEN := FSM'LEN * 2;

MOVE B'BUF := %23,2;
MOVE *     := B'FSM'BUF,(LEN);

FWRITE (FSM'FILE,FSM'BUF,-(LEN+1),%320);
IF <> THEN ...
```

FSMDIAL

The procedure to dial the Fuel Site Micro using the Hayes Smartmodem 1200 accepts a phone number string from the calling program. Setting the modem for half-duplex and decimal digit responses as required can be done with commands but we chose to use the internal DIP switches since this configuration would always be used. The phone numbers used for each location called are stored in a controlling file in the exact format as will be passed to the dialing procedure including 1 + area code for any long distance calls. The string 'AT T D' does several things: 'AT' is the character sequence telling the Hayes that this is a command for it to execute, the 'T' instructs the Hayes to default to tone dialing mode, and the 'D' begins the actual dial command. If the number must be dialed using pulses, a 'P' can be inserted in front of the number stored in the control file. Examples of phone number dialing sequences are:

```
AT T D 555-1234

AT T D 1 (808) 555-1234
```

```
AT T D P 555-1234
```

Spaces, parentheses and hyphens passed to the Hayes are ignored and can be added to improve readabilty. The procedure will set up the phone number dialing sequence, pass it to the Hayes, and return a response indicating results.

```
FRWRITE (FSM'FILE,L'BUF,0,0);       << SEND INITIAL CR >>

MOVE B'BUF := 256 (" ");            << CLEAR BUFFER >>
MOVE B'BUF := "AT T D ",2;          << BUILD COMMAND >>
MOVE *     := B'FSM'PHNUM,(20);

LEN := 28;
DO LEN := LEN - 1                   << FIND LENGTH OF          >>
 UNTIL LEN < 6                      << SIGNIFICANT CHARACTERS >>
 OR B'BUF (LEN) <> " ";
IF LEN = 6 THEN                     << NO PHONE # PASSED >>
BEGIN
 FSM'STAT := 1001;
 GO TO EXIT;
 END;

FWRITE (FSM'FILE,L'BUF,-(LEN+1),0);

MOVE B'BUF := 256 (" ");

TIME'LIMIT := 30;
FCONTROL (FSM'FILE,4,TIME'LIMIT);

LEN := FREAD (FSM'FILE,L'BUF,-256);    << GET RESPONSE >>
IF > THEN
BEGIN
 FSM'STAT := 1000;                  << INDICATES EOF TO  >>
 GO TO EXIT;                        << CALLING PROGRAM   >>
 END;
IF < THEN
BEGIN
 FCHECK (FSM'FILE,FSM'STAT);
 IF FSM'STAT = 22 THEN
  FSM'STAT := 1002;
 GO TO EXIT;
 END;

IF B'BUF <> "1" THEN                << DIALING FAILED >>
BEGIN
 FSM'STAT := 1002;
 GO TO EXIT;
 END;

FSM'STAT := 0;
```

FSMLOGON

In order to secure the Fuel Site Micro from unauthorized access, a logon procedure was established which included a password stored with the control file for each location. The procedure will not be shown here for security reasons but the actual code did not involve anything complicated.

After the subroutines had been developed and testing began a program was needed that would emulate the Fuel Site Micro's system console. This program's initial purpose was only to assist in testing of other programs using the communications interface to the micro. This program later proved to be very useful for both development of new application programs and for setup and maintenance of the Fuel Site Micros. As new programs were needed to load or collect a particular set of data, this program, along with a data line monitor, could be used to observe exactly how this was to take place, including such things as errors that can occur, responses, terminating the transfer, etc. Also, as stations were brought up or reconfigured, the console emulator could be used to download

a command file of setup parameters and on a day to day basis, monitor and maintain these parameters from the host as if the operator were at that location.

As mentioned earlier, a sleeper program was needed to initiate the data collection program. At first, we wrote a quick and dirty sleeper that simply streamed a job and went to sleep for a specified time interval. This program worked well enough, but we later decided that it would be necessary to automatically stream a few other jobs at different time intervals than that of the data collection job. After checking the contributed library catalog, we found the SLEEPER programs. These programs consists of a sleeper that actually performs the functions and a maintenance program that updates the control file. The use of a control file makes this system very flexible and easy to use as it can stream jobs, run programs, execute MPE commands and files of commands. Our testing and use of these programs showed them to be very useful and flexible.

Section VI - Error Checking

Although there is a certain amount of error detection and correction built into the modems, there are still many transmission errors that can occur. With the addition of a few simple procedures, error detection, and a certain degree of correction, could be greatly increased. The primary method of doing this was by making use of check sums. Almost all data transfer that occurs automatically is numeric, consisting of either decimal or hexidecimal digits, and includes a check sum that is computed and compared by the host. Another area of checking was passed to MPE itself by enabling parity checking. If a parity error or check sum error occurs, the host requests that same record from the micro, up to three times, until no errors occur. Also, since there is a possiblity of missing records or collecting the same records twice due to an abnormal previous collection, each data record collected is assigned a request or transaction number by the micro. The HP keeps track of which records were received as of the last collection and reports any discrepancies (missing or duplicated records) which may require manual intervention. These few simple procedures greatly enhance those built in and provide an adequate amount of error checking.

Section VII - Remote System Backup

We have several HP 3000's in various locations connected via DS lines so we decided to design some features of the system in such a way as to make remote system backup as easy as possible. First of all, each of our four systems has at least one dial-up port configured as device class 'AUTODIAL' (See configuration listing in appendix) with a Hayes Smartmodem 1200 attached. Also, all software and at least a skeleton data base is duplicated at each of the sites. The controlling of the automatic communication is done, in part, by a location file. Each of our locations for each of our systems has an entry in this file with flags indicating such things as whether or not to call, what data to collect or down load, etc. By controlling our programs from this file, along with a minimal amount of other data in the data base, it became quite simple to switch data communication tasks from one system to another. To make the switch usually involves nothing more than changing a few flags, and adding a new task to the sleeper control file.

Summary

It is possible to affect a reasonably reliable data communications technique without doing anything exotic in the hardware or software.

This paper contains a very simple and basic approach which we found acceptable. All sloppy procedures and inefficient coding are probably oversights and should be treated as such. No one is trying to state that the indicated techniques are the best or only way of accomplishing the end results. If better is possible, good is not good enough! So by all means improve on this information in any way you can.

BEST OF LUCK TO ALL OF YA'LL!

DISCLAIMER

The ideas and material presented in this paper have worked for us but it is entirely the reader's responsibility to assure that any material utilized by the reader is functional for the reader's intended purpose. The authors disclaim any and all responsibility or liability which may arise directly or indirectly from the use in whole or in part of any of the ideas, concepts, examples or other material presented herein. The authors are presenting this paper as individuals seperate and apart from any company affiliation.

ACKNOWLEDGEMENTS

This paper was co-authored by Mike Shelton. I would like to express my appreciation to Mike not only for his assistance on this paper but also for doing a superb job in assisting in creating a CONTROLLED DATA COMMUNICATION INTERFACE TO REMOTE MICRO COMPUTERS for our shop.

Other acknowledgements are as follows:

Hewlett/Packard	for superb development gear
Ross Scroggs	for related papers and work
Chris Gindorf	for professional help and direction
Tommy Sorrells	for efforts beyond the call
Judi Pianta	for beginning footsteps
Sharon Ashby	for being part of my life
Randy Nicholson	for the challenge and opportunity

GLOSSARY OF SELECTED TERMS

Bit Map: a representation of data where a particular bit within a byte has positional value rather than numerical value; the on/off status of a bit represents a condition; examples: bit mapped graphics where each bit represents the on/off status of a pixel on the screen; table of numbers where each bit represents the absence or presence of a number by its being off or on.

CCTL: Carriage control

Check Sum: a sum of the numerical values of each character in a particular buffer.

DC1: ASCII character; Device Control 1; decimal equivalent: 17.

DC3: ASCII character; Device Control 3; decimal equivalent: 19.

DTR: Data Terminal Ready; pin 20 of RS-232C interface.

ETX: ASCII character; End of Text; decimal equivalent: 3.

FCLOSE: HP system intrinsic or subroutine used to close a file.

FREAD: HP system intrinsic used to read from a file.

STX: ASCII character; Start of Text; decimal equivalent: 2.

REFERENCES

Computer Data Protection, Federal Register, Commerce Department/ National Bureau of Standards, March 17, 1975, pp. 12134-12139.

McEntire, Norman C. "Hayes's Stack Smartmodem." Byte, March 1983, pp. 282-290.

Meushaw, Robert V. "The Standard Data Encryption Algorithm, Part 1: An Overview." Byte, March 1979, pp. 66-74.

Meushaw, Robert V. "The Standard Data Encryption Algorithm, Part 2: Implementing the Algorithm." Byte, April 1979, pp. 110-130.

MPE Intrinsics Manual, Hewlett-Packard Company, January 1981

Scroggs, Ross. "Everything You Wanted to Know About Interfacing to the HP3000, Part I.", Proceedings of the 1982 HP 3000 IUG San Antonio Conference, February 1982, pp. 6-40-1 to 6-40-10.

Smartmodem 1200 Owner's Manual, Hayes Microcomputer Products, Inc., 1982.

Tibbets, John J. "Everything You Wanted to Know About Interfacing to the HP3000, Part II.", Proceedings of the 1982 HP 3000 IUG San Antonio Conference, February 1982, pp. 6-40-11 to 6-40-17.

LOG DEV NUM	DRT NUM	U N I T	C H A N	T Y P E	SUB TYPE	TERM TYPE (DEV	SPEED TYPE)	REC WIDTH	OUTPUT DEV	MODE	DRIVER NAME	DEVICE CLASSES
1	25	0	0	3	8	(7935)		128	0		HIOMDSC2	DISC SPOOL SYSDISC
.												
.												
.												
52	8	32	0	16	0	10	240	40	52	JAID	HIOTERM1	TERM
53	8	33	0	16	1	10	240	40	53	JAID	HIOTERM1	AUTODIAL DIALUP
54	8	34	0	16	1	10	240	40	54	JAID	HIOTERM1	DIALUP
55	8	35	0	16	0	10	240	40	55	JAID	HIOTERM1	MUX

DAVID L. ASHBY is a Texas native with 10 years data processing experience. This experience spans banking, college, service bureau, and petroleum industries with a good mix of applications within each.

David is currently Data Processing Manager for E-Z Serve in Abilene, Texas and has been in that job since early in 1980. Among the unique experience is his overseeing and developing of an Automated Fueling System on Micro Computers. These units are interfaced into HP 3000 computers -- thus the base for this paper and associated presentation.
