AAEC/E601

AAEC/E601

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS RESEARCH LABORATORIES

A UNIX/IDRIS DATAWAY INTERFACE

by

G.W. PEADY

FEBRUARY 1985

ISBN 0 642 59805 3

Australian Atomic Energy Commission

Research Establishment

Lucas Heights Research Establishment

A UNIX/IDRIS DATAWAY INTERFACE

by

G. W. Peady

ABSTRACT

The software interface between a Unix/IDRIS system and the Dataway network at the Lucas Heights Research Laboratories is described. The interface gives complete control over the network without the need for a detailed knowledge of low-level protocols and error recovery mechanisms. Several examples of the use of the interface are given.

National Library of Australia card number and ISBN 0 642 59805 3

CONTENTS

A UNIX/IDRIS Dataway Interface

1. Introduction

This document describes how the Dataway can be controlled from Unix (or IDRIS). It is not intended to explore the folk lore surrounding the use of the Dataway nor does it discuss the hardware. In the following sections, where mention is made of active and passive processes, the reader should be familiar with these concepts and the associated conventions on the network (Peady, 1981 ; Sanger, 1976). The interface has been designed so that the default values satisfy most needs; however, these may be changed as desired to achieve complete control on the network. The user interface for the handler has been designed to be the same from either Unix or IDRIS and, provided the routines described in this paper are used, programs written for the Unix system (to control the network) should be easily transported to the IDRIS system (and vice versa). Some knowledge of Unix, the use of C, and the concept and handling of signals in Unix is assumed.

The terms active and passive are used widely through this report. It is essential that the difference between these two terms is understood for proper operation of the network. In any communication on the network, there are only two computers involved at any one time. The computer which is controlling a session is called active as it determines the sequence of commands which will be issued. The other computer is the passive computer. The passive computer can only request changes of the active computer which may or may not respond to this request. An

example of this active/passive relationship is illustrated by considering a hypothetical session between a user on a terminal and the central computer. The user has the "$" prompt on the terminal and types N<ret># in reply. At this stage, the computer to which the terminal is connected (i.e. the passive computer) sends a request to the central computer (i.e. the active computer). Conventionally this request is called a "Primary Write". This request is considered by the central computer and if resources are available responds by sending a command to print

job: INIT

Following this the active computer requests that some characters be read from the terminal keyboard. The user types these characters and indicates completion by a <ret>. These characters are then sent to the central computer to satisfy the active computer's read request. Note that at this stage the central computer has issued all of the commands except the initial request. The central computer then searches for the requested job and begins to list it by issuing write commands to the passive computer. If the listing is very long then the user may decide that no more is required and hits a '?'. The presence of this character is indicated to the central computer and is interpreted to mean a request to terminate the listing and the listing of the file is stopped. Subsequently a write containing the prompt '-' is issued followed by a

---

<ret> indicates the RETURN key

read to determine the next command from the terminal. The session continues on until the command to terminate the session is read by the central computer. The end is signified by the printing of a '$'. During the whole of this session, the central computer has been in complete control of the commands which have been issued, and has been actively controlling the session. The computer which has the terminal connected to it has only been able to request changes to the direction of the session but it has not been able to initiate them - it has been passive. In this report, the controlling program in the active computer is called the active process and similarly for the passive control program.

For those unfamiliar with some of the terms used a glossary is included as Appendix A.

2. The Unix handler

The Unix handler has been implemented to allow the Dataway to be used like any other device on Unix. It resembles a terminal when used in an active sense and a raw block device when used in a passive sense. To use the Dataway from Unix the user must decide which mode to operate in and then open the appropriate device. For active communication, the devices are usually of the form /dev/tty?? where ? is a single character denoting a Dataway logical unit. On Unix systems, /dev/ttyd is a Dataway terminal device, (on IDRIS /dev/ttyD1), and other Dataway terminal devices can be found as devices with the same major device number (or by asking the system administrator). Note that there are usually several

of these devices already opened as Dataway terminals.

For passive communication the devices are found in the directory /dev/name, where name is the name of the Dataway computer with which communication is required. The names of valid computers can be found in the file dw.h in the standard user include library (/usr/include - Unix, /lib - IDRIS) and are always in upper case(e.g. PDP9L). Appendix B contains a copy of a current dw.h. Note that the file local system should be included in the source program before the use of dw.h. Within the directory for each computer there are entries of the form pbase? where ? corresponds to the relative address to be used from the Unix computer's base address. (i.e. /dev/PDP9L/pbase0 provides a link between the PDP9L and the base address on the user's computer).

The major device number is an index into system tables for handlers. The minor device number is made up of two parts

1. The lower three bits indicate the Dataway address to be used relative to the system Dataway base address.

2. The upper five bits of the byte are used as an index into an array of 32 valid Dataway addresses. The valid (that is non-zero) addresses for the user's system can be found in the standard include directory as file dwadr.h.

This rather unusual structure for the minor device number was chosen to enable the network system interface software to require no operating system changes. This is important if only a binary version of the

operating system is available and no changes can be made (as in the case of IDRIS).

3. Opening a Dataway device

To use the Dataway the usual open system call must be made. For active mode communication it is not generally necessary (or desirable) to specify an address. If, however, an address is really required, then the process does not wait for a primary write, and a system call is available to set the required address (see dwcom, Section 7 and Appendix C). For active mode and no address specified, no reads or writes will be handled until another computer has requested the services of the opening process.

For passive mode communication, it is essential to specify an address. A function in the C library makes the opening for passive control of the Dataway fairly easy. It is

```
dwopen(adr, pw, pwsize)
    int adr;
    char *pw;
    int pwsize;
```

where adr is the the Dataway address for communication, pw is a pointer to the primary write (note that it must point to a word boundary) and pwsize is the size of this write. If the open was successful, dwopen returns a valid file descriptor or -1 otherwise. After a successful open, the primary write is sent and the result of this primary write is

found in the global variable pwdone. With this routine, the first available Dataway address is used and there is no need for any special action. If a particular base address is required, the normal open call must be used on that device. (i.e. to make a connection between the PDP9L and the user's base+3 then the device /dev/PDP9L/pbase3 should be opened - the success of this open depends on the availability of the requested address pair and other system table allocations.)

There are two special addresses supported for access to the central computer - these are IBMPRIM and IBMSEC. If these addresses are used then the allocation of the real path to the central computer, (i.e. via the PDP9L or the Intel micro-computers) is under the control of the Dataway network administrator.

4. Driving an active process

The control of an active process is fairly simple if the user is satisfied with the default values for the Dataway read and write commands; i.e. for a write a 0x01; for a read with echo, a 0x02; and for a read with no echo, a 0x06 command. Note if the Dataway terminal is in the 'no-echo' mode, then the command issued is the current read command or'ed with 0x04. To issue a write command, it is necessary only to write data to the device. However, a read command is not issued immediately a read from the device is requested, but is issued only after an attention is received from the other computer indicating that a character is available. It is issued at this stage only to ensure that the communication path is not blocked waiting for a read when there may

be other writes which could be going out. If the terminal is in raw mode, there will be no read issued at all as the character which generated the attention is available with the command and is placed immediately in the terminal input buffer to be returned with the next read. It is possible to force the issuing of a read command when a read system call is made but this requires the Dataway device to be placed in a special mode (see dwcom, Section 7).

The user who wishes to change the default values can use the dwcom function (see Section 7).

An example of the extended use of the Dataway in an active fashion is given in Appendix D where the program is one which allows remote computers on the network to be loaded from the Unix computer.

5. Driving a passive process

The controlling of a passive process is complicated by the fact that the next command to be sent is unknown. Once the Dataway device has been opened and the primary write sent, the active computer determines which commands are sent. Generally, the arrival of a command from the active computer is indicated by the sending of a signal to the controlling process. By default, the signal is that corresponding to an IOT (ie. signal 6) and arrangements should be made by the process to intercept this type of signal by making the call "signal(6,serviceroutine)" in the main routine, where "serviceroutine" is the name of the function which is going to handle the Dataway signal (be sure to re-enable the

interception of this signal in the service routine). On receipt of this signal, the command can be determined (see Section 6) and action taken according to the type.

If the Dataway command is a read, then the data needed for this read can be acquired and, when ready, a write can be issued specifying the buffer and the length in the normal fashion. Note that a write is needed to fulfil a Dataway read request. If the Dataway command is a write, then a read can be issued to indicate the region for transfer and maximum number of bytes. No further signals will be sent to the process in the case of a Dataway write until the process indicates, with a dwcom type 3 call, that the previous write command has been completely handled. This action is necessary to ensure that another Dataway signal does not interrupt until the handling of the first has been completed.

When no data transfer is required for a particular command, this is indicated by calling dwcom(dwfid, 3, &stat) where dwfid is the file descriptor for the Dataway device and stat contains the status which is to be sent on completion of the command. The normal status which is sent to indicate successful acceptance of the command is 014(0x0c) a Channel End/Device End. However any status may be sent and it can be interpreted by the controlling process in the active computer in whatever manner is desired.

To overcome a problem which occurs occasionally in the PDP11 hardware interface, the length of a buffer for Dataway transfers should be made at least two bytes longer than the maximum number of bytes expected.

Note that all transfers should be on a word boundary and an even number of bytes. If these conditions are not met, an error will be returned for the former and the byte count increased by one for the latter.

On device pbase0, certain commands (0161 (0x71), 0101 (0x41), 0106 (0x46)) will not be passed to the controlling process. These commands are used for network communications and diagnostics. Consequently, any user who really needs these commands should ensure that this address is not used. Note that device pbase0 is not used by the dwopen routine. The dwopen call should be used normally for all passive opens to ensure portability between machines and operating systems.

The conventions which exist for communicating on the Dataway should be adhered to when communicating with the central computer (and most others!)[Sanger 1976]. Care should be taken when using the Dataway passively because, if each signal is handled within the signal intercepting routine, a stack overflow may occur as a new signal comes in before the previous signal has been completely handled. There are several ways to avoid this difficulty. One is to use a non-local goto of the form envsave/envrest(See section 2, Thompson and Ritchie, 1976), another is to set a flag in the signal intercept routine and to detect the setting of this flag in a loop in the main routine. The first is the safer and is illustrated in Appendix E, where the program shown implements a remote printer for the central computer on a Unix system.

Another technique considered for use in certain cases was to set the handler into a mode where no signals are generated at all. In this

mode, the commands could be picked up with usual read and write system calls and the use of dwcom. The usual method would be to issue a read system call with a large buffer and byte count; then, if a Dataway write command came, the data would be transferred and the length of the data transfer, returned by the read system call. The type of command could then be determined by using a dwcom type 2 call. If the Dataway command was, in fact, a read, the read system call would return with an error type EWRNGCMN(4), and the type of Dataway command could again be determined in the usual manner before the appropriate write system call was sent. This technique was not used because, once a read system call had been issued, the process would have been unable to do anything else until there was a Dataway command (or a signal) and it would have been possible, if errors occurred in the hardware, to lock up the entire system with processes locked in memory.

6. Closing a device

The close system call can be made at any time to complete a communication, thus tidying up the handler and freeing any addresses which were being used. For active devices using normal conventions, any characters which are still in the output queue are written to the passive computer, as well as a Rewind/Unload command to terminate the session. For a close system call on passive devices, if a Rewind/Unload was the last command to be sent from the active computer, then the handler frees the address it was using. If a Rewind/Unload was not the last command, the handler sends a command to the active computer to

indicate that the close has occurred. This command is in the form of a Unit Check status to an outstanding command or, if none is outstanding, a write to indicate the abnormal termination (a write type 5 with data of 0xcf00, -1, -1).

7. Access to the handler values

Within the handler are data areas which contain status information and values for a particular device . Some of these values can be read and written while others can only be read. The structural form, default values and explanations of each element are shown in Appendix C.

A system call is available to access these values and, by using various versions of this call, information can be extracted or stored in these values for any open Dataway device. The function which implements this system call is dwcom and has three arguments, the first is the file descriptor for the open Dataway device, the second determines the action to be taken, and the third depends on the type of the second. A complete description of the arguments can be found in Appendix C.

The names given to the second argument can be used in a program by including the file dw.h, which also contains many useful definitions for using the Dataway. By using this routine the default values in the handler can be easily changed as frequently as needed. Appendix D lists a routine which uses this call to change the commands used for reads and writes to allow remote computers to be loaded with programs available in Unix. Appendix E is the listing of the program which acts as a remote

printer for the central computer. This program uses the routine to determine the type of command sent by the central computer.

8. Conclusions

This paper describes how the Dataway can be used from Unix/IDRIS for the control of active and passive devices. The interface routines have been used in practice to provide many facilities for the Unix/IDRIS systems. All the plotting performed on the Zeta plotter uses the above routines as do the central computer job submission facilities available from these systems. All network access to the three Unix/IDRIS systems ($2,$3,$4) uses the above facilities, as does network access from these computers.

9. References

Peady, G. W., [1981] - High Level Programming of Network Hardware, Proc. - DECUS Symposium, Griffith University, Brisbane.

Sanger, P. L., [1976] - The AAEC Dataway Terminal Communication System, Proc. 7th Aust. Computer Conf., 2, 610-622.

Thompson, K., Ritchie, D. M.,[1976] - Unix Programmer's Manual, Bell Telephone Laboratories.

10. Appendix A - Glossary of terms

Active	An active computer on the network is the computer which is in complete control of the communications. It decides what commands are issued and will only accept requests for a change in activity from its partner (the passive computer).
Busy status	On completion of a command on the network there is an exchange of status between machines. Some of these statuses are commonly used and Busy is the status which implies that the issuing machine is unable, at this stage, to accept the command. At some later stage it will indicate that it is willing to accept the previous command by issuing a special command, called a Nop/Control which is terminated with a Device End status to indicate the change of state.
Close	A system call to Unix/IDRIS to indicate that the use of the file/device associated with a previous open has been completed.
Major device	The Unix/IDRIS major device number is an index into a table of the available devices. It is used to select which device handler to use when open, close, read and write system calls are made.
Minor device	The Unix/IDRIS minor device number is used by the device handler to differentiate between devices with the same Major device number.
Open	The open system call on Unix/IDRIS is used to allow the program to request the use of a particular file or device.
Passive	The passive computer is the computer which usually initiates a Dataway session and, after that, is completely under the control of the other computer. The passive computer in a communication does not know what the next command will be. It is only able to make certain requests on its partner.
Raw mode	This is a special mode for a Unix/IDRIS terminal in which no processing of input data is performed and each character is sent to the controlling process as it is received. Normally characters are accumulated line by line and editing can be done within a line by using rubouts etc.; on completion of the line the characters are available for the controlling process.

Read	The read system call allows the program to request data and to specify a region and maximum byte count for that data.
Signal	A signal in Unix/IDRIS is a form of inter-process communication. It can be handled by the receiving process if required and appears in that case like an asynchronous interrupt.
Unit Check status	This is a another commonly used status on the network which indicates that something has gone wrong!.

11. Appendix B - Listing of dw.h

Following is a listing of the current version of dw.h on the PDP11/45 system. Note this file should be included in a C program in the following fashion.

```
#include <local_system>
#include <dw.h>
```

```
#define PDP9L           0x40
#define AMC1145         0x48
#define SIS1134         0x6c
#define ACLNOVA         0xc0
#define CEPD1123        0xe0
#define INTELS          0x01
#define NOVA2           0x70
#define ECLIPSE         0xf8
#define IBMPRIM         BASE+1
#define IBMSEC          BASE+2

#ifdef CEPD
#define BASE    CEPD1123
#else
#ifdef AMANDC
#define BASE    AMC1145
#else
#ifdef SIS
#define BASE    SIS1134
#else
==== ***** Local not known !!!!
#endif
#endif
#endif

#define GETSET          0
#define PUTSET          1
#define DWCOM           2
#define RELEASE         3
#define SETSIG          4
#define SENDATTN        5
#define SETATTN         6
#define GETMOTD         7
#define SPECOPEN        8
#define SETTRACE        9
#define SETIBM          10
#define DUMP            99
#define DWINFO          100
```

```
#define OK              0x00
#define UE              0x01
#define UC              0x02
#define DE              0x04
#define CE              0x08
#define CEDE            0x0c
#define BUSY            0x10
#define STATMOD         0x40

#define TIO             0x00
#define WRITEC          0x01
#define READEC          0x02
#define NOPC            0x03
#define PW              0x05
#define ATTN            PW
#define READNOE         0x06
#define RW              0x07
#define CLEARC          0x09
#define UEREAD          0x0a
#define READBRK         0x0e
#define RWUL            0x0f
#define DIAGW           0x41
#define BUFR            0x42
#define DIAGR           0x46
#define MOTD            0x71
#define NOA             0x7f

#define SIGDATAW        6
#define EWRNGCMN        4
#define SWAB            020
#define ATTNCAP         040
```

12. Appendix C - Dataway support routines and default values

There are two routines which are available to help in the control of the network: the first one is

```
dwopen(addr, pwbuf, pwsize)
    int addr;
    char *pwbuf;
    int pwsize;
```

This routine is used to commence communication with the address specified as addr. It is only used when communication is to be passive, the usual open command is used for active communication. The call sends the primary write, starting at pwbuf and of length pwsize, to the specified address. If the open is successful then a file descriptor is returned for later use; on failure a -1 is returned. The success of the primary write is indicated by the contents of a global variable, pwdone, which contains the number of bytes transferred in the primary write.

The second routine which can be used is

```
dwcom(fid, command, value)
    int fid;
    int command;
    char *value;      /* value is really a pointer
                         to different things
                         depending on the command */
```

The first argument is the file descriptor used for communication with the network, i.e. that returned from a successful open call. The second and third arguments and the actions for each are:

Argument 2	Argument 3	Action
GETSET - 0	pointer to a structure of type dwread	Read the current values of the handler into the structure specified.
PUTSET - 1	pointer to a structure of type dwwrite	Write the values specified into the handler values.
DWCOM - 2	pointer to a structure of type dwcmst	Read the last command and status into the structure specified. Note that a command of zero indicates that a large number of attempts have been made to communicate with the address given but none was successful. This usually indicates that one or other of the Dataway control units is not 'on line'.
RELEASE - 3	pointer to a character containing the status	Set the status to be used on completion of the next Dataway command or, if zero status is specified, then the next signal will be released.
SETSIG - 4	pointer to a short integer	Change the signal generated when a Dataway command comes in while in passive mode. The number is the number of the signal which will be sent and it should be between 0 and NSIG(the maximum signal on your system - usually 16). Setting the signal to 0 disables any signals from the handler.
SENDATTN - 5	pointer to a short integer	Indicate that the next command is not in reply to a requested command. Only valid for passive devices. Usually used for sending attentions. The call also performs the same function as SETATTN.

SETATTN - 6	pointer to a character	Allow the system to reply to a 0x09 or 0x0a command (mainly used for ACL access) after an Attention has been sent. (Some computers on the network use these commands to determine the character which caused a Unit Exception status or Attention. Others require an immediate response to these commands and will not allow a Busy reply.)
GETMOTD - 7#	pointer to an array of 80 characters	Read the last message of the day sent on the network.
SPECOPEN - 8	pointer to a short integer	Set the handler into a special mode for use by active processes. In this mode, all read and write system calls generate Dataway read and write commands. If the content of the pointer is greater than 1 then it is used as the address on the Dataway for communication, otherwise the process must wait for a primary write request as normal.
SETTRACE - 9#	pointer to a structure of type dwtrinfo	Set information for tracing Dataway activity
SETIBM - 10#	pointer to an integer containing IBM routing information	Allows selection of route to central computer if the central computer is addressed using IBMPRIM and IBMSEC as network addresses. Note that this information can also be dynamically changed by a global network command from the central computer. The routing is according to the value of the integer as follows: 0-PDP9L-PDP9L, 1-PDP9L-INTEL, 2-INTEL-PDP9L, 3-INTEL-INTEL.

DUMP - 99#	pointer to a buffer large enough to store the dump information. This size can be calculated from information provided by the DWINFO call	Get the Dataway trace buffer.
DWINFO - 100#	pointer to a structure of the type dwparams.	Used to obtain information about the Dataway driver parameters.

Note that the entries marked with # can only be used with the special device /dev/dwmotd and errors will be returned if used on normal devices.

The structure of the data which is read and written using dwcom has the following form

```
struc   dwwrite {

        char writecom;      /* the command used when issuing
                               writes */

        char readcom;       /* the command used when issuing
                               reads */

        int pwhdr;          /* header used for detecting a
                               request for use of Unix resources */

        int maxchars;       /* maximum number of characters to
                               be transmitted in one transfer */

        int status;         /* status of Dataway handler */
        };
```

```
struct   dwread {

         struct dwwrite
              dwout;

         int mydwstatus;      /* the last status used for a
                              Dataway activity */

         int hisdwstatus;     /* the last status returned from the
                              other computer */

         char mylastcmnd;     /* the last command used for a
                              Dataway transfer */

         char hislastcmnd;    /* the last command received on a
                              Dataway transfer */

         char mydwadr;        /* the Dataway address being used
                              for this session */

         char hisdwadr;       /* the Dataway address of the other
                              computer */

         };

struct   dwcmst {

         char command;        /* the last Dataway command */

         char sigstat;        /* current state of an internal flag
                              */

         };

struct   dwtrinfo {

         int dwtron;          /* if non-zero tracing is enabled */

         int dwtral;          /* if non-zero tracing is effective
                              on all addresses */

         int dwthisadr;       /* address of other machine for
                              trace */

         int dwtmyad;         /* address of this machine for
                              tracing */

         };
```

```
struct   dwparams {

         int dwbase;          /* base Dataway address for this
                              machine */

         int numpadr;         /* number of passive Dataway
                              addresses available */

         int totpas;          /* total number of concurrent
                              passive processes available */

         int numaproc;        /* number of active processes */

         int tbufsize;        /* size of trace buffer */

         };
```

The default values for these depend on the mode of operation. Thus for active mode the values are:

```
writecom    01
readcom     02        - for a read with echo
            06        - for a read with no echo
maxchars    132
header      070000
```

For passive mode the values are:

```
writecom    01
readcom     02
maxchars    800
```

The variable status has the following significant bits:

Bit	Meaning
0	GO - set when there is a Dataway transfer in progress.
1	OPENED
2	Rewind/Unload received.
3	CLOSING

4# Swab - alternate bytes of data will be swapped on input and output.

5# Attention capability - the passive process will send a command on the next system call.

6# Need PW - the active process needs a primary write to wake it up or a passive process will force a command out.

7+ SETSTAT - a specified status will be sent for the next command.

8 MYTRAN - transfer started by this computer.

9 READ - last command was a read.

10 WRITE- last command was a write.

11 HANGUP - terminal has been hungup - i.e. a ctrl P has been sent.

12 READY - for i/o activity.

13 NEEDRPT - the current activity needs to be repeated if aborted.

Note those bits marked with # are the only bits which can be set by a user directly. The bit marked with + can be set by the user along with an appropriate status using the dwcom function number 3.

Note that all of the values in the structure dwread can be read, but only those in the structure dwwrite may be changed.

13. Appendix D - An example of active use

This appendix contains a description of a program which controls the Dataway in an active manner. It is used to load network machines with data from a computer using the Unix operating system. The program opens an available Dataway terminal address in a single user mode - so that only one process can communicate using it. It is then dormant until a network computer requests the loading of a program. The name of the file to be loaded is contained in this request and is expected to be in the directory /dwload. The file is assumed to be a standard Unix executable file. The usual protocol for a network load, after the primary write, is a Rewind(0x07) command, followed by a 6-byte write(0x05) command, (containing the size, load address and start address in consecutive integers), followed by sufficient write(0x09) commands to transfer the data, followed by a Rewind/Unload(0x0f) command. If the initial request cannot be satisfied then a no-access(0x7f) command should be sent.

When using the handler it is not necessary to send a Rewind at the beginning of the sequence or a Rewind/Unload at the conclusion, as both of these are sent by the handler on open and close system calls respectively.

Following is a commented source of the loading program.

```
/* routine used to load a Dataway computer from a Unix computer
 * a request in the form of a primary write is received by the
 * Unix computer .
 * this primary write has a header of 0x4000
 * and the 8 bytes of data with the primary write specify
```

```
* the name of the Unix file to be loaded from the directory
* /dwload
*
* the usual Dataway convention for remote loading is:
* a request of the above form,
* followed by a rewind command from the accepting computer
* followed by a 0x05 write command containing the size,
*      the load address and the start address
* the data is then sent with a series of 0x09 write commands
* and the loading is completed with a rewind/unload command
*
* the file must be in the directory /dwload as a standard
* Unix executable file with magic numbers 407,411 or 410
* and the usual Dataway conventions for loading the remote
* machines are used to transfer and execute this file
*
*/


#include <local_system>
#include <dw.h>

#define NULL '\0'
char file[20] "/dwload/";

struct aouthdr
{
    /* this describes the header on a normal Unix executable file
     * and is used to extract information for use when loading
     * the file to the remote machine
     */

    int magn;        /* the magic number */
    int ptext;
    int ds;
    int bss;
    int symbtab;
    int entry;
};

struct dwread dwterms;

main(){
    register int dw,j;
    register char *bp;
    char outbuf[80];
    int  *lo,i;

    while(1){ /*stay there for ever */
        dw=open("/dev/dwl",402);
            /* open Dataway for single use and wait for pw */
```

```
dwcom(dw, GETSET, &dwterms);
    /* read all about it */
dwterms.pwhdr = 010000;
    /* set the header expected for the primary write */
dwcom(dw, PUTSET, &dwterms);
    /* and tell the handler */
bp = &file[8];
i = read(dw, bp, 80);    /*get primary write */
if(i > 8)
    i = 8;  /* only first 8 characters of the
             * primary write are significant
             */
for(j = 0; j < i; j++, bp++ )
{
    *bp = tolower(*bp&0x7f);
    if(*bp == ' '){
        *bp = NULL;
        break;
    }
}
*bp = NULL;
if((i = open(file,00)) > 0){
    dwcom(dw, GETSET, &dwterms);
        /* read about the current settings */
    dwterms.writecom =0x05;
        /* make the first write command a 0x05
         * note that the rewind command has been
         * sent by the handler on acceptance of
         * the primary write
         */

    dwcom(dw, PUTSET, &dwterms);
        /* change the handler settings */

    j = read(i,outbuf,020);    /* read about the file */
    if(j < 0 || !(( j = outbuf->magn) != 0407
        || j != 0411 || j != 0410)) goto noaccess;
    lo = outbuf;
    *lo++ = (outbuf->ptext+outbuf->ds)/2;
        /*size of text */
    *lo++ = 0;
        /*UNIX always starts at 0 */
    *lo = outbuf->entry;
        /*start address */

    write(dw,outbuf,6);
        /* send start address ,size etc. */

    /* change the write command again */
    dwcom(dw, GETSET, &dwterms);
    dwterms.writecom = 9; /* make it 9's now */
```

```
            dwcom(dw, PUTSET, &dwterms);

            /* now send out the data using write 0x09 commands */
            while(( j = read(i,outbuf,80)) > 0)
            {
                write(dw,outbuf,j);
            }
        }
        else{
            /* can't find the file requested indicate this
             * with a 0x7f command
             */
noaccess:
            dwcom(dw, GETSET, &dwterms);
            dwterms.writecom = 0x7f;
                /* conventional invalid request command */
            dwcom(dw, PUTSET, &dwterms);
            write(dw,outbuf,1);
        }
        close(dw);
            /* the close automatically sends a rewind/unload */
    }
}
```

14. Appendix E - Remote printer example for passive use

An example of a passive program which acts like a remote printer.

```
/* sprt - Dataway remote printer
 * written by glynn w peady 13/11/79
 */

#include <local_system>
#include <dw.h>

#define UBITS   0x8080


extern int pwdone;
/*  this is the result of the primary write */


int donedw      0;
int nextst      0;

int sprtb[] = {
    0x7100,
    ('S'<<8) | 'P' | UBITS, ('R'<<8) | 'T' | UBITS , UBITS
};
/* note by convention the primary write
 * is in upper case ascii in reverse PDP-11
 * order with the upper bit in each byte a 1
 */

char        cr          '\r';
char        ff[]            "\014\r";
char nwline[] "\n\n\n";
char sprtpid[] "/spool/sprtpid";
char printcom[] "/spool/sprtcom";

int repeat 0;

char prbuf[802];

int dwdes, out, comfid;

struct dwread dataw;


abterm(){
    extern char sprtpid[];

    printf("abnormal termination of printer");
```

```
    close(dwdes);
       /* could be eliminated as a close is done by exit() */
    unlink(sprtpid);
    exit();

}

dwh()
{
    /* the signal received for the Dataway request
     * is handled by this routine
     */

    register int linespc, i;
    int j;
    int prcomcc;
    int comnd;


    signal(6, dwh); /* come and see me again some time */
    /* note the catching of signals must be re-enabled each time */

    dwcom(dwdes, GETCMND, &comnd);    /*get the command */

    comnd =& 0xff;       /* only the low byte is needed */

    if(pwdone <= 0)
    {
        /* something went wrong with the primary write */
        if(comnd == 0)
        {
            /* command = 0 if a repeat or no-reply interrupt
             * was generated - close and try again later
             */
            printf("Dataway unit is not on line?\n");
            close(dwdes);
            sleep(30);
            dwdes = dwopen(PDP9L, sprtb, sizeof( sprtb ));
        }
        else
               /* what else can it  be?? */
            printf("strange Dataway command %o", comnd);
        envrest(&donedw);
    }
    linespc = 0;    /* count the number of line feeds needed */
    switch(comnd)
    {
    case 01:
        linespc++;        /*put cr */
    case 041:
        linespc++;        /*new page */
```

```
case 015:
    linespc++;          /*3 lines */
case 011:
    linespc++;          /*2 lines */
case 05 :
case 0101:                  /* crack!!! */
    linespc++;          /*1 line */
    i = read(dwdes, prbuf, 800);    /* read the data */
    if(i <= 0)printf("Dataway write error\n");
    else{
        swab(prbuf,i);
                    /* swab the bytes from
                     * the central computer
                     */
        for(j=0;j<i;j++)
            prbuf[j] =& 0177;
            /* and take off the top bit */
        write(out, prbuf, i);
        switch(linespc)
        {
        case 5:
            write(out, &cr, 1);  /* just a carriage return */
            break;
        case 4:
            write(out, ff, sizeof( ff ) );
            /* new page please */
        case 0:
            break;
        default:
            /* write out the appropriate  number of lines */
            write(out, nwline, linespc);
            break;
        }
        break;
    }

case 0102:      /* command read */
    /* issued after a UE to determine the printer command */
    comfid = open(printcom, 0);
    prcomcc = read(comfid, prbuf, 800);
    if(prcomcc > 0){
        swab(prbuf,prcomcc);
            /* get the command into a
             * suitable form
             */
        for(i=0;i<prcomcc;i++)prbuf[i] =| 0200;
        write(dwdes,prbuf,prcomcc);
        envrest(&donedw);        /* return to the mainline */
    }
    else
        goto sendcede;
```

```
        /* no data transfer with the following commands
         * hence an immediate CEDE
         */

    case 035:
        write(out, nwline, 3);
        goto sendcede;
    case 031:
        write(out, nwline, 2);
        goto sendcede;
    case 025:
        write(out, nwline, 1);
        goto sendcede;
    case 061:
        write(out, &ff, 1);
        goto sendcede;
    case 0121:
        write(out, &cr, 1);
        goto sendcede;
    case 0177:
        printf("no access \n");
        goto finishoff;
    case 017:
finishoff:
        unlink(sprtpid);
        exit();
    default:
        printf("unknown Dataway command %o\n", comnd);
sendcede:
        if(nextst == 0)nextst = CEDE;
    }
    /* note that all Dataway write commands come to
     * this is used to synchronise the sending of signals
     * nextst == 0 implies a release of the next signal
     * nextst != 0 implies a completion of the next transfer
     * with the status specified in nextst
     */
    envrest(&donedw);
        /* return to mainline to release next signal
         * this procedure is used so that the next
         * signal does not come before the interrupt
         * routine has had a chance to return and clean
         * up its  stack. This in fact forces the
         * clean up.
         */
}


igsignal(sig,addr)
int sig;
int (*addr)();
```

```
{
    if(signal(sig,addr)&1)signal(sig,1);
}

main()
{

    int pid,j;

    signal(6, dwh);
    /* ignore signals if they are already  ignored */
    igsignal(1,abterm);
    igsignal(3,abterm);
    if(!stat(sprtpid,prbuf) || ( j = creat(sprtpid, 0600)) < 0)
        exit();   /*only want one sprt going at any one time */

    pid = getpid();
    write(j, &pid, 2);
         /* write out the pid for use with printer
          * commands
          */
    close(j);
    while((dwdes = dwopen(PDP9L, sprtb, sizeof( sprtb ))) < 0)
    {
        printf("cant open Dataway \n");
        sleep(20);
    }
    out = 1;              /*use standard output */
    while(1)
    {
        if(envsave(&donedw) == 0)
        {
            dwcom(dwdes, RELEASE, &nextst);
            /*release next signal */
            nextst = 0;
            /*make it zero in case reset by cancel signal */
        }

        else
            sleep(30);        /*wait for prince charming */
    }
}

/* sprt - Dataway remote printer
 * written by glynn w peady 13/11/79
 */

#include <local_system>
#include <dw.h>

#define UBITS   0x8080
```

```
extern int pwdone;
/*  this is the result of the primary write */


int donedw      0;
int nextst      0;

int sprtb[] = {
    0x7100,
    ('S'<<8) | 'P' | UBITS, ('R'<<8) | 'T' | UBITS , UBITS
};
/* note by convention the primary write
 * is in upper case ascii in reverse PDP-11
 * order with the upper bit in each byte a 1
 */

char cr '\r';
char ff[] "\014\r";
char nwline[] "\n\n\n";
char sprtpid[] "/spool/sprtpid";
char printcom[] "/spool/sprtcom";

int repeat 0;

char prbuf[802];

int dwdes, out, comfid;

struct dwread dataw;


abterm(){
    extern char sprtpid[];

    printf("abnormal termination of printer");
    close(dwdes);
       /* could be eliminated as a close is done by exit() */
    unlink(sprtpid);
    exit();

}

dwh()
{
    /* the signal received for the Dataway request
     * is handled by this routine
     */

    register int linespc, i;
    int j;
    int prcomcc;
```

```
int comnd;


signal(6, dwh); /* come and see me again some time */
/* note the catching of signals must be re-enabled each time */

dwcom(dwdes, GETCMND, &comnd);    /*get the command */

comnd =& 0xff;       /* only the low byte is needed */

if(pwdone <= 0)
{
    /* something went wrong with the primary write */
    if(comnd == 0)
    {
        /* command = 0 if a repeat or no-reply interrupt
         * was generated - close and try again later
         */
        printf("Dataway unit is not on line?\n");
        close(dwdes);
        sleep(30);
        dwdes = dwopen(PDP9L, sprtb, sizeof( sprtb ));
    }
    else
           /* what else can it  be?? */
        printf("strange Dataway command %o", comnd);
    envrest(&donedw);
}
linespc = 0;    /* count the number of line feeds needed */
switch(comnd)
{
case 01:
    linespc++;        /*put cr */
case 041:
    linespc++;        /*new page */
case 015:
    linespc++;        /*3 lines */
case 011:
    linespc++;        /*2 lines */
case 05 :
case 0101:               /* crack!!! */
    linespc++;        /*1 line */
    i = read(dwdes, prbuf, 800);    /* read the data */
    if(i <= 0)printf("Dataway write error\n");
    else{
        swab(prbuf,i);
                  /* swab the bytes from
                   * the central computer
                   */
        for(j=0;j<i;j++)
            prbuf[j] =& 0177;
```

```
            /* and take off the top bit */
        write(out, prbuf, i);
        switch(linespc)
        {
        case 5:
            write(out, &cr, 1);  /* just a carriage return */
            break;
        case 4:
            write(out, ff, sizeof( ff ) );
            /* new page please */
        case 0:
            break;
        default:
            /* write out the appropriate  number of lines */
            write(out, nwline, linespc);
            break;
        }
        break;
    }

case 0102:      /* command read */
    /* issued after a UE to determine the printer command */
    comfid = open(printcom, 0);
    prcomcc = read(comfid, prbuf, 800);
    if(prcomcc > 0){
        swab(prbuf,prcomcc);
            /* get the command into a
             * suitable form
             */
        for(i=0;i<prcomcc;i++)prbuf[i] =| 0200;
        write(dwdes,prbuf,prcomcc);
        envrest(&donedw);        /* return to the mainline */
    }
    else
        goto sendcede;

    /* no data transfer with the following commands
     * hence an immediate CEDE
     */

case 035:
    write(out, nwline, 3);
    goto sendcede;
case 031:
    write(out, nwline, 2);
    goto sendcede;
case 025:
    write(out, nwline, 1);
    goto sendcede;
case 061:
    write(out, &ff, 1);
```

```
        goto sendcede;
    case 0121:
        write(out, &cr, 1);
        goto sendcede;
    case 0177:
        printf("no access \n");
        goto finishoff;
    case 017:
finishoff:
        unlink(sprtpid);
        exit();
    default:
        printf("unknown Dataway command %o\n", comnd);
sendcede:
        if(nextst == 0)nextst = CEDE;
    }
    /* note that all Dataway write commands come to
     * this is used to synchronise the sending of signals
     * nextst == 0 implies a release of the next signal
     * nextst != 0 implies a completion of the next transfer
     * with the status specified in nextst
     */
    envrest(&donedw);
        /* return to mainline to release next signal
         * this procedure is used so that the next
         * signal does not come before the interrupt
         * routine has had a chance to return and clean
         * up its stack. This in fact forces the
         * clean up.
         */
}


igsignal(sig,addr)
int sig;
int (*addr)();
{
    if(signal(sig,addr)&1)signal(sig,1);
}

main()
{

    int pid,j;

    signal(6, dwh);
    /* ignore signals if they are already  ignored */
    igsignal(1,abterm);
    igsignal(3,abterm);
    if(!stat(sprtpid,prbuf) || ( j = creat(sprtpid, 0600)) < 0)
        exit();   /*only want one sprt going at any one time */
```

```
    pid = getpid();
    write(j, &pid, 2);
         /* write out the pid for use with printer
          * commands
          */
    close(j);
    while((dwdes = dwopen(PDP9L, sprtb, sizeof( sprtb ))) < 0)
    {
        printf("can't open Dataway \n");
        sleep(20);
    }
    out = 1;                /*use standard output */
    while(1)
    {
        if(envsave(&donedw) == 0)
        {
            dwcom(dwdes, RELEASE, &nextst);
            /*release next signal */
            nextst = 0;
            /*make it zero in case reset by cancel signal */
        }

        else
            sleep(30);          /*wait for prince charming */
    }
}
```