An automatic system for data logging and verification of multi-channel, multi-size geophysical data

M. S. SMITH and R. H. DAVIS

Department of Computer Science, Heriot-Watt University, Edinburgh, Scotland

D. BEAMISH

Institute of Geological Science, Edinburgh, Scotland

INTRODUCTION

Geophysical background

William Gilbert announced in 1600 that the Earth is a great magnet. Since that time, numerous methods and techniques have been developed to utilise the many properties of the Earth's natural magnetic field. One such geophysical technique utilises the time changes in the natural geomagnetic field, which occur with frequencies from kilohertz to periods of the order of one day, to investigate the electrical properties of the Earth. When time changes with periods of the order of several seconds to one day are used, the technique is referred to as the magnetotelluric (MT) method.[1]

The MT method is a way of determining the distribution of the Earth's electrical conductivity structure from measurements of the naturally occurring time changes in the electric and magnetic fields on the ground surface.[2]

For a complete description of the geo-electromagnetic field, five orthogonal field components must be recorded. The five data components are referred to as X, Y, H, D and Z, all being functions of time.

The first two data components are the potential differences between two points on the Earth's surface. X is a measure of the induced electric field in the north–south direction, Y is a measure of the electric field in the east-west direction.

H, D and Z represent the three orthogonal components of the magnetic field. H is a measure of the magnetic field in the north–south direction, D is a measure in the east-west direction and Z is the vertical component. In practice, a sixth data channel is also recorded. This channel is temperature which is used to correct the instruments response response to thermal effects.

The six-channel MT instrumentation produces analogue output of the field values. These voltages are sampled at a discrete rate that defines the sampling interval simultaneously on all six channels. A discrete six-channel time series, suitable for computer analysis, is thus formed.

The goal of each field MT system is to obtain such data over a long continuous period of time (say six months) to act as a data base for subsequent analysis. The data base will consist of time variations of the Earth's electric and magnetic fields recorded every 5 s. Five such field systems are arranged in a spatial array, giving 30 data channels of simultaneous magnetotelluric data.

Such an MT-based field monitoring system is to be set up in western Turkey, a region which is seismically active and earthquake prone.

The purpose of the experiment is to use the recorded data base to investigate stress-induced changes in the crust of the region which may be detected by the fields recorded.

Project specification

There is therefore a requirement to measure, capture and verify data from several sets of six-channel magnetotelluric sounding systems deployed in the field, and operating simultaneously.

Up to six field systems may be in operation at any one time over an area of 100 km^2. These six remote field systems will be provided with radio-link telemetry to a common base computer system, which will 'handle' all $6 \times 6 = 36$ possible data sources.

The radio-links are not described here but the assumption is that a standard RS 232-C serial data line exists between each field system and the base computer.

The software development work for the above system concerned the data capture and its subsequent verification by the base computer system. The minimum hardware specification of this system consisted of a DEC PDP-11/23 mini-computer with 128k RAM, a 20 Mbyte Winchester disk for data file storage, a terminal/hard copy listing device, together with a magnetic tape deck.[3]

Incoming data is received on six ports (i.e. $2 \times$ DEC DLV11-J serial line interface boards provide eight such ports), which are configured as part of the system. The incoming data from each field system arrives every 5 s, and for each field system the full data sample consists of $6 \times 2 = 12$ bytes. Each data word (two bytes), consists of 12 bits of data and a four-bit redundancy.

It was required to write: (1) a 'front end' interrupt based data logger in DEC MACRO 11 16-bit assembler code,[4] to service each of the six serial lines (five chosen for the actual field system) as data is received and an interrupt is generated and; (2) a FORTRAN, driver[5] to verify, store and in general handle the logged data.

An interrupt based logging system was chosen in preference to a device handler based system, as the former was considered to be faster, simpler and to offer easier access for the main FORTRAN driver to the logged data.

These two routines in conjunction, formed the Real-Time data collection system.

When such a system is established and working, it was intended to establish the MACRO 11 routine in a foreground mode, with the data collected being transferred to array space allocated in a FORTRAN program working in background mode.

In developing the program however, it was later found to be more suitable to load the FORTRAN driver into the foreground as well as the MACRO 11 ISR routines, thereby releasing the background mode for other important tasks. One such task would be, for example, the transfer of logged data files to tape.

With a few exceptions, each incoming data channel can only be fully verified by considering properties of N samples. Typically $N = 720$ would be provided from each channel in 1 h of real-time.

At this point, some specified processing of the data series will take place, and related information is to be listed on the line printer enabling the performance of the field systems to be monitored in a consistent manner. The data processing routines can be considered as standard, and the project requirement includes the implementation of such routines in relation to the real-time data collection system.

In parallel with the above, the raw data series will form file-store, and will be transferred to the mass storage device (Winchester disk). The format of the files will be discussed later on.

The project goal consists therefore of working, verified and tested software made up from the interrelated background program (FORTRAN) and its associated foreground, interrupt driven subroutines (MACRO 11).

The performance of this software system is critical from two main points of view.

The first is that the logger must be sufficiently robust and general as to perform even with loss of data on any or all of the input lines. Fault recovery procedures were also required here.

The second performance function requiring investigation was the 'probability' of simultaneous data arrival, and the established procedures under the DEC system for handling such an event.

The system is to run effectively for periods of the order of months so that these performance viewpoints are considered fundamental. Although the field system will not be unattended, minimal interruption to data capture and storage is a requirement.

Program efficiency and simplicity in helping to reduce any possible timing problems, are also important from a power consumption viewpoint. Problems exist with the local power grid in Turkey and therefore, minimising power consumption and device usage is desirable to reduce the demands made upon the secondary backup power supply.

The potential developments and possible complementary additions to the core data collection system running on the PDP-11/23 are numerous, so that as a core system for possible future enhancement, the implementation of simple, well structured and documented software modules is of importance in facilitating the ease of such development.

The advantages of designing applications programs as aggregates of small program modules are well documented.[6,7] They include ease of development, testing, debugging and modification, as well as facilitating the possibilities of reducing core memory requirements for the active system.

Most of the programs were constructed on the PDP-11/03 Laboratory (Lab) system allocated for software development. The single user DEC O/S used on this machine was RT11SJ – Real-Time 11 Single Job O/S, although in the later stages RT11FB – foreground/background environment was mounted. This equipment was similar in most important aspects to the eventual field system to be purchased.[8, 9]

To supply the test data to develop the logging programs, data generators/transmitters were constructed to run on the two other available machines namely the PDP-11/03 rack mounted low-power field-based system, and a lab-based PDP-11/23 data processing system.

Although the eventual field system is designed for five site operation, a maximum of four only was achievable using these machines. Data array and buffer allocations, however, were made for five sites, to ensure that no space allocation problems would occur when implementing the field program in RT11FB.

One of the data storage devices used was the DEC TU58 dual cartridge tape drive unit on the data generating Field 11/03 system. Test data generators were downloaded from the Lab 11/03 system on to this unit and then run on the Field 11/03.

The other storage device used, was the DEC RX02 dual 8 in floppy disk drive unit. Each of these disks were formatted for double density giving them a capacity of 974 RT11 blocks (1 block = 512 bytes), i.e. approximately 4 M bits each. RT11SJ/FB and its associated editors, compilers etc. occupied almost all of one disk (DY0:), while the other disk (DY1:) carried the developed programs and associated data files.

As can be seen from Fig. 1, most of the Dec machines used interfaced with peripherals through the DLV11-J. This is a four-channel asynchronous serial line unit which links peripherals to the DEC LSI 11 bus over EIA compatible data lines. It uses a UART chip to convert data from parallel to serial and serial to parallel.

The DLV11-J allows for the configuration of four independent channels of differing baud rates if required, depending on the associated peripheral. The baud rates on the eventual field system will be identical, but the baud rates used during the laboratory development phase were nearly all different. This, however, caused no major problems although it did necessitate careful consideration in setting up a multichannel test run using all three machines.

Device control/status and I/O buffer register addresses associated with each of the four DLV11-J channels were transparent from the main memory addressing.

THE RT11SJ/FB SYSTEM AND COMPONENTS

RT11, as a modular Real-Time O/S is adapted to user requirements for both software and hardware configurations

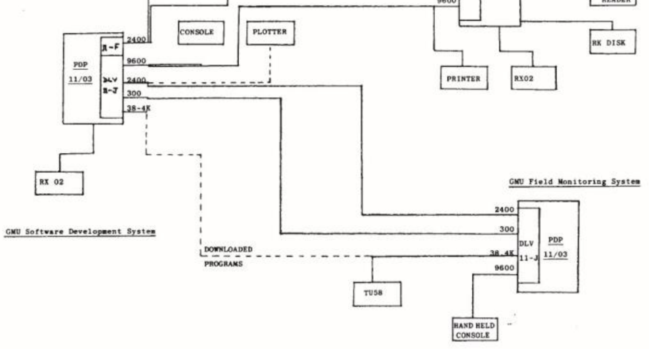

Figure 1

by the SYSGEN system facility. This facility allows the user to create a minimum or maximum system configuration which is usually stored on disk or tape.[9]

In RT11, all files are named with a six-character name, followed by a dot and a three character file type code. Thus for example, PROGRAM.FOR is a FORTRAN source file which when compiled forms a PROGRM.OBJ object file version. This object file is then linked to form either a PROGRM.SAV executable file in RT11SJ/FB, or a PROGRM.REL relocatable file for foreground execution in RT11FB.[10]

The RT11SJ system is booted up from a default system device, automatically on power on, requiring only a simple 'y' confirmation of start up. Once booted, the system's command file is consulted, from which the final (user specified) touches to the system's configuration are added before passing control to the Resident Monitor.

Two problems are worth noting at this point. The first is that with the foreground program running in an infinite loop – as the main data logging program LOG5.FOR does, no time would normally be allocated to the background job.

Consequently, an ISLEEP RT11FB macro call was inserted into LOG5.FOR, to allocate time for any background job to proceed. The amount of time allocated by this system call is a tradeoff between the data collector's responsiveness, input double buffers size and the processing demands made by any background jobs. The ISLEEP value above, was found to suffice.

The second problem is that it is relatively easy to stop the execution of the foreground job by typing 'CONTROL C' twice. This in the field has disastrous implications, and so consequently it is recommended that a method of locking the foreground program in be used.

The ISLEEP system macro call used above is one example of the numerous system macro calls available for use on the RT11FB system.[11]

The following lists and summarises the system macro calls used in programs associated with this project. Most of these were used in direct RT11 block I/O transfers, and in automatic I/O channel and file allocations.

ICLOSE	This subroutine terminates activity on a specified channel and frees it for use in another operation.
IENTER	Allocates space on the specified device and creates a tentative directory entry for the named file.
IFREEC	Returns a specified RT11 channel to the pool of available channels.
IGETC	Allocates an RT11 channel, in the range 0-17 (octal), to be used by other routines and marks it in use so that the FORTRAN I/O system will not access it.
IRAD50	Converts a specified number of ASCII characters to RADIX-50, and returns the number of characters converted. N.B. RADIX-50 is a DEC compressed character code.
IREADW	Transfers a specified number of words from the indicated channel into memory.
ISLEEP	Suspends the main program execution of a job for a specified amount of time. RT11FB only.
IUNTIL	Suspends main program execution of the job until the time of day specified.
IWAIT	Suspends execution of the main program until all I/O operations on the specified channel are complete.
IWRITW	Transfers a specified number of words from memory to the specified channel. Control returns to the user program when the transfer is complete.
LOOKUP	This function associates a specified channel with a device and/or file for the purposes of performing I/O operations.
TIME	This subroutine returns the current time of day as an eight-character ASCII string.

INTERRUPTS, LANGUAGES AND FORMATS USED

The PDP 11/03 has two levels of priority. However, the PDP 11/23, the eventual field machine, has eight levels of priority ranging from zero, the lowest, to seven, the highest. When the CPU is operating at level seven an external device cannot interrupt it.

Device priority is also influenced by the distance between the device and the processor on the bus. Thus if two devices of equal priority, e.g. the keyboard and the printer, both at priority level four, interrupt simultaneously, the device nearer the processor on the bus will receive the interrupt grant acknowledgement.

The Interrupt Service routines used were constructed to be fast, simple and efficient. The worst case example of five simultaneous interrupts would result in each of the device channels being serviced as above, i.e. nearest to the CPU first. As each ISR takes approximately 40.7 s, with a field data input baud rate of 300, no timing problems are anticipated in using this interrupt driven programmed data transfer.

Each data file will contain all the data collected from one site in a day, i.e. 360 RT11 blocks each. Thus for five sites, approximately 922 K bytes of data will be collected every day and the 20 M byte Winchester storage disk will have to be backed off to tape at least once every three weeks.

Having thus examined the various geological and computing background aspects to this project, we may now move on to examine the development and structure of the data collection and verification system itself.

DEVELOPMENT OF DATA COLLECTION AND VERIFICATION SYSTEM

In the construction of the logging system, it soon became apparent that there were three main developmental areas, whilst a fourth area of ancillary programs emerged later.

These three main areas were Data Generation, Data Collection and Data Verification, and each of these will be examined separately with their associated programs and subroutines within this and the following three sections.

The particular specification supplied for the data logging system, dictated a defensive posture for the system in the sense that all real-time values required by the base computer were externally supplied. Each field site had its own very accurate clock, thereby enabling each of the five field sites to transmit its 12 byte data burst very accurately every 5 s.

Consequently, the data logging routines had to act purely as recipients of the transmitted data, and were designed and constructed with this design imposition.

Several overall design considerations for such externally driven real-time systems need accurate system design as the time scale of events is not controlled by the base computer.[6] In such cases, it is also pointed out that such systems are to function on the basis of random input.

Each of the eight software modules of the final logging suite, and all of the ancillary programs are considered to be self-documented. Listings of them are available on request.

Data generation and the initial development phase

Before any development of the data logging programs could take place, data generators had to be written to supply the test data to develop these programs, after which the next logical task was to develop programs to accept the data transmitted and display it for visual verification.

Thus complementary programs were written as data acceptors to these data generators. They may also be identified as the program precursors of the data logging programs, as they tackled the problem of accepting data on a serial port, storing its byte by byte arrival in a buffer and then displaying the 12-byte buffer contents on a console or line printer for verification.

At the same time, the concept of double buffering was introduced here, so that once a 12-byte data buffer is full and awaiting display, a further 12 bytes can be stored without corrupting the first 12 thereby giving the main program time to deal with the first data buffer.

This standard data logging technique was exercised for the first time at this stage and developed later on for use in the final data collection and verification system.

These data generating and early data accepting programs formed an important base for the development of the final field system. They provided the test data for program development, and the ability to easily check these data sources.

In addition ancillary programs were developed to support the data collection and verification system.

THE LOGGING SYSTEM – OVERVIEW

The data logging system to be described in this section consists of one main co-ordinating program (LOG5.FOR) and six subroutines (SETUP5.MAC, BEEP.MAC, STORE5.FOR, CHNGF5.FOR, FILEX5.FOR and LISTER.FOR). These plus the data verification subroutine make up the actual data collection and verification system.

Each of these software modules is described using Blackman's[12] approach, is considered to be self-documented, is available on request, and is consistent with other geomagnetic software.[13,14]

Low level macro 11 routines

Two MACRO 11 subroutines are used in the data logging system. These are SETUP5.MAC, the main assembly language routine which sets up the interrupt based multichannel data logging service routines and BEEP.MAC, a simple routine to attract operator attention to error conditions detected by the logging system.

SETUP5.MAC is the interrupt driven DEC assembler routine, which sets up five ISR's to service each of five serial input lines from the two DLV11-J I/O boards. The basic data unit used is the 8-bit byte. Each byte is stored from the input buffer associated with a channel, into a storage buffer location in the double buffer. Each double buffer is 3456 bytes consisting of two single 1728 byte buffers. Once one of these single buffers is full for a channel, a flag is set and the main program is thereby prompted to clear the full buffer.

The second MACRO 11 routine used in the data collection system is BEEP.MAC. BEEP simply loops ten times sending 10 ASCII '7' (BEEP) characters to the console. It is called only when a system error message has been generated and printed.

These then are the two MACRO 11 routines developed for the data collection system. We may now move on to examine the FORTRAN IV program and subroutines comprising the rest of the system.

HIGHER LEVEL FORTRAN IV ROUTINES

Description of these routines which form the remainder of the logging system can be divided into four sub-task areas:

(1) Logical control of the logging operation – LOG5.FOR.
(2) Continuous data storage – STORE5.FOR.
(3) Data file handling – CHNGF5.FOR and FILEX5.FOR.
(4) Error handling – LISTER.FOR.

Logic control of the logging operation

As an automatic system, barring equipment failures, LOG5 and the data collection and data verification system will log day files of data indefinitely as specified. The only compulsory operator requirements are a backing off to tape of the data files at least once every three weeks, and an occasional resetting of the PDP 11/23 base computer internal clock if required.

LOG5 consists of three main sections of code: declaration and initialisations, logging control and status checking.

Continuous data storage.

STORE5.FOR is the subroutine called from within LOG5, to partially process, load and store the full data buffers for each channel onto the 20 Mbyte Winchester disk. In addition to this data storage function, it is also responsible for updating block and hour counters and for summing the data series components (X, Y, H, D and Z) to enable hourly statistical verification to occur.

Data file handling

The two subroutines included in this third sub-task area are CHNGF5.FOR and FILEX5.FOR. Both use specific RT11 system macro calls to open and close data files on an RT11 block structured disk. These DEC system macros have been summarised previously and include IWAIT, ICLOSE, IFREEC, IGETC, IRAD50 and IENTER.

The function of CHNGF5.FOR is to close an old data file named with the parameters passed in from LOG5, rename and open a new data file on the disk with the next data file name in the series. To do this latter task, FILEX5.FOR is called from within CHNGF5.

The second subroutine used in handling the data files is FILEX5.FOR. Although its function could be considered as a simple extension to CHNGF5, is specific internal functions associated with renaming and opening a new data file on the DEC system coupled with its need to be called from within LOG5 as well as CHNGF5, justified its existence as a separate subroutine.

Error handling

The subroutine concerned with printing meaningful error messages for the whole data collection and verification system is LISTER.FOR.

DATA VERIFICATION

Two approaches to data verification were used in the development of the logging system. The first approach as mentioned in the project specification, involved verifying the data by considering properties of *N* samples of the data series using statistical means. The second approach, however, involved the more arbitrary verification of the data received and logged by the system.

As each magnetotelluric site produces 144 data integers for each of the five MT components evern 12 minutes, an hourly check of $5 \times 144 = 720$ samples for each component was incorporated into the logging system for continuous data verification.

CONCLUSIONS

The ISR's, necessarily in machine code, are externally driven. Data from each site enters the system via an asynchronous serial line device. Each data byte generates an interrupt which the ISR services to collect and store successive data bytes (i.e. multiple-channel) from each site. A sample sequential store mode also preserves the sequential nature of the channels.

The ISR's are of modular construction i.e. each asynchronous line (i.e. site) is serviced by equivalent code but distinguished by individual entry points and hardware addresses. ISR functionality is divided into two parts:

(1) Interrupt initialisations – which enables data collection to begin at a specified real-time.
(2) Interrupt service – once initialised, the ISR collects and stores data generating interrupts in the hardware addresses set up in (1).

In order to service the ISR's, a FORTRAN driving program and associated sub-routines is operated in real-time. The basic function of this software is to create a logical progression of file store from the data collected by the ISR's. This driving program accesses the systems line-time clock and thus manages the real-time aspect of data collection.

Each data integer is 16 bits (double byte), and arrives from the UART in the form of two consecutive bytes (MSB first). The size of the double buffers used has a real-time significance for the system, in that they represent 12 min of real-time. With five sites, a maximum of 15 RT11 blocks are written to Winchester storage disk every 12 min, which is considered to be a minimal disk access demand.

This main FORTRAN driver (LOG5) was designed to be simple in its logical structure with specific value parameters passed into the commonly used subroutines. This increased modularity,[7] and eased the documentation of the module interfaces.

The dynamic considerations of systems status (LOG5) and data verification (STATS5) have made it more flexible and informative. Concern about possible corruption of the systems simple integrity by the addition of data verification code, has been dispelled by the robust nature of STATS5, and the dual role has eased and simplified the code overheads required for real-time data verification.

The common use of the error listing subroutine LISTER and indeed the common use of all data handling routines, greatly eases system expansion and possible adaptation to other analogous applications and tasks.

In conclusion, the development of a modular, tested data collection and verification system for the I.G.S. has been satisfactorily achieved, and the main body of this report details how this has been done. Full listings of the programs are available on request. The system is at present being applied in the field in Turkey, and commenced in March 1984.

ACKNOWLEDGEMENT

We are indebted to J. Riddick and J. MacDonald (Geomagnetism Unit) and C. Fyfe (Global Geigmology Unit) of the Institute of Geological Science, Edinburgh for their advice, support and practical assistance during this work.

REFERENCES

1 Sharma, P. V. *Geophysical Methods in Geology*, Elsevier Scientific Publ., Amsterdam, 1976, Ch. 4
2 Vozoff, K. 'The magnetotelluric method in the exploration of sedimentary basins, *Geophysics*, 1972, 37(1), 98
3 Digital Equipment Corporation. (Dec.). *Memories and Peripherals Hand-book*, 1978/9, Ch. 1–2, DLV11J (pp. 2-147–2-173)
4 Southern, R. W. *PDP-11 Programming Fundamentals*, Southcroft Publ., Canada, 1972
5 McCracken, D. D. *A Guide to FORTRAN IV Programming* (2nd edn), Wiley and Sons, Chichester, 1972
6 Pritchard, J. A. *An Introduction to On-Line Systems*, NCC Publ., 1976
7 Allworth, S. T. *Introduction to Real-Time Software Design*, MacMillan Press Limited, London, 1981
8 DEC. *RT11 System User's Guide*, Vol. 2A, 1980
9 DEC. *Software Support Manual*, Vol. 3B, 1980
10 DEC. *RT11 FORTRAN IV Manual*, 1980
11 DEC. *Programmers Reference and MACRO 11 Language Manual*, *RT11*, Vol. 3A, 1980
12 Blackman, M. *The Design of Real-Time Applications*, Wiley Interscience Publ., London, 1981
13 Kirkpatrick, I. J. Software Development for a Programmable A.D.C. Device Internal Report
14 Riddick, J., Forbes, A. and Green, C. 'The recording and processing of digital magnetic data from the UK observatories', *G.M.U. Report No. 27*, N.E.R.C., I.G.S., Edinburgh.