

https://dspace.org/

The implementation of a data acquisition and dissemination system using a micro-computer as a intelligent front-end processor to a multi-user mini computer

Chapman, Mark Jonathon

2014-12-15
http://hdl.handle.net/10539/15983

Downloaded from a DSpace repository. https://dspace.org/

DISSERTATION

NOTE

A routine is defined, for the purposes of this project, as a section of source code which has a single entry point, a single exit point, is easy to comprehend and displays a high degree of functional cohesion within itself and a low degree of functional coupling to other routines.

The design was documented in the Design Specification, most of which has been incorporated into this document and the BADEDAS Technical Manual.

4.4 PROGRAMMING

4.4.1 GENERAL

The benefits of a high level language for programming are well-known. Unfortunately the requirements of a high-level language for the implementation of BADEDAS were rather restrictive. The compiler had to be able to run on the Eclipse under AOS and generate Micronova object code, which did not contain any operating system calls. Further requirements were that the object code had to be fast in execution and very compact in size, it had to be possible for programs to make use of customised device drivers and the language had to be suitable for real-time applications.

After an investigation it became apparent that RTL/2 was the only language available on the Eclipse which come close to meeting these requirements. Unfortunately, RTL/2 did not exist for running under AOS. The possibility of being able to convert the RTL/2 compiler to run under AOS presented itself as a challenging task. However, a minor, but insuperable, technical problem foiled the attempt to achieve this and the programming had to proceed in assembler language. The procedure followed and the problems experienced in the attempt to convert the RTL/2 compiler have been documented, for the benefit of future workers

faced with this problem, as a technical note, which has been included as Appendix A.

Coding was approached in a generally top-down manner, with the higher level routines being coded and stand-alone tested first. In some cases it proved convenient for testing purposes to deviate slightly from this top-down approach.

4.4.2 MICRONOVA ASSEMBLER

Micronova assembler language is not a proper subset of that of the Eclipse. Instructions in the Micronova instruction set which are not compatible with the Eclipse instruction set are:

* instructions which make use of the stack or stack pointers
* the multiply and divide instructions
* the software trap instruction
* I/O and CPU instructions which are dependent on unique aspects of the Micronova hardware

There is a large number of Eclipse instructions which are not supported by the Micronova. The object code generated by these instructions would cause unpredictable results if they were included in a Micronova program.

Since a requirement of BADEDAS is that all software development be carried out on the Eclipse, it was necessary to find a method for assembling Micronova programs on the Eclipse. This could be done with the standard Eclipse assembler, by restricting programs to the subset of the Micronova instructions which is Eclipse compatible. This was, however deemed to be an unsatisfactory solution.

The Eclipse assembler provides a facility by which the table defining the Eclipse instruction set, the semi-permanent symbol table, can be flushed and new instuctions of the Eclipse/Nova instruction format can be defined. A file was created containing the necessary macros and pseudo-instructions to do this. This file, which is called BADMAC, is contained in the source code listings and is further documented in the BADEDAS Technical Manual. By including this file as the first file in an assembly, a program written in Micronova assembler may be assembled on the Eclipse, under AOS. An added advantage of this approach is that the inadvertant use of Eclipse instructions

within a Micronova program is detected by the assembler.

4.5 TESTING AND INTEGRATION

After a routine had been coded and subjected to stand-alone testing, it was integrated into the existing system. Once integrated, the system functions involving that routine were tested. In some cases this required the use of stubs or dummy routines in place of lower level routines called. The testing of BADEDAS-M routines at this level on the Eclipse proved to be awkward, mainly due to the incompatibility of the Micronova stack instructions with those of the Eclipse. So once the down-line loader had been developed, and after initial tests on the Eclipse, the partially integrated system was loaded into Micronova RAM for more thorough testing.

Testing and debugging on the Micronova were carried out from an Eclipse user terminal using the utility MNDEB. MNDEB, developed by Perseus, provides a virtual link from one Eclipse user console port to another. This makes it possible to access the Micronova console debug monitor from an Eclipse console when the Micronova is linked via its TTO port to another Eclipse console port. This utility proved to be a great time-saver, as the alternative would have been to down-line load the Micronova, then unplug the lead forming the Micronova/Eclipse link from the Eclipse and plug it into a CRT terminal for debugging and then to reverse the process for subsequent down-line loading.

The main problems experienced in testing were due to the very limited capabilities of the console debug monitor, especially when dealing with interrupts (see "PROBLEMS" Chapter 5).

4.6 ACCEPTANCE TESTING

Acceptance testing of a system is the process of testing the system in its intended environment using the real input data.

Equipment used in the acceptance tests included two signal generators and a dual channel oscilloscope. One signal generator was used to provide input signals, while the other provided the external clock which determined the sampling and conversion frequencies for the A/D and D/A converters. The oscilloscope was used to display input and output signals. Figure 8 shows the BADEDAS setup for acceptance testing.

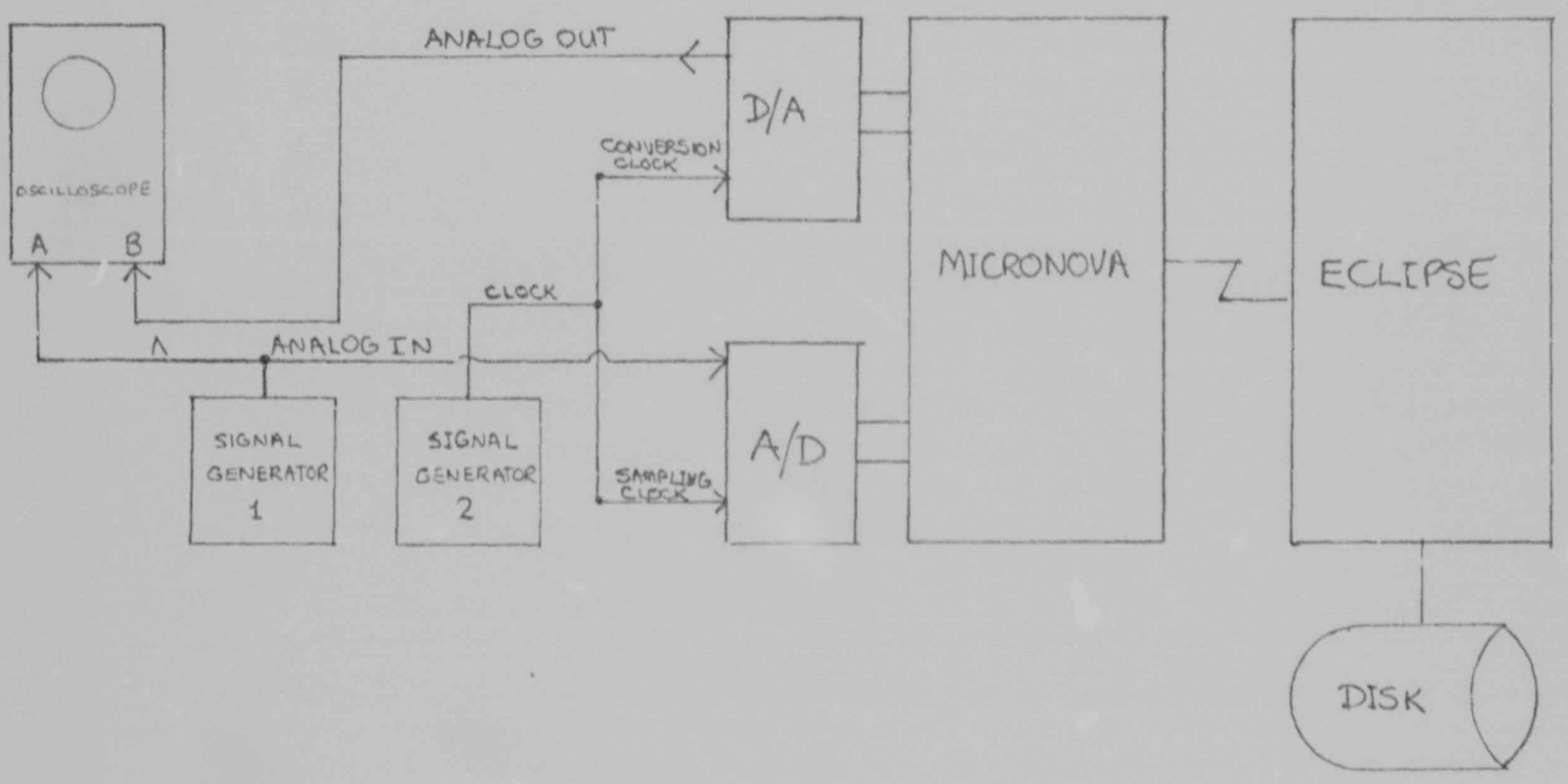


FIGURE 8 : SET-UP FOR ACCEPTANCE TESTING.

A number of simple experiments were conducted which made use of all BADEDAS features. Results obtained were compared to those expected and when the system was found to be functioning correctly it was demonstrated to the project supervisor.

The following is a brief summary of the experiments conducted:

4.6.1 EXPERIMENT 1

The purpose of this experiment was to demonstrate the operation of the BADEDAS initialise and pipe commands and the functioning of the A/D converter.

One signal generator was used to provide a 10 kHz sampling frequency for the A/D converter and the other, to provide a 1 kHz square wave to one of the A/D's analog input channels. The A/D converter was set up, using a BADEDAS initialise command, to input 4096 word data blocks from a single input channel.

eg. X BAD/I AD0,4096,1700000

A BADEDAS pipe command was entered specifying the A/D converter as the pipe source, an AOS disk file as the destination and a pipe block count of 1.

eg. X BAD/P AD0,FILE.DAT,1

When the pipe command had run to completion, the contents of the AOS disk file were inspected to ascertain that they did represent the square wave input. The experiment was also performed for pipes involving multiple blocks (ie. a block count of more than 1).

eg. X BAD/P AD0,FILE.DAT,15

4.6.2 EXPERIMENT 2

The purpose of this experiment was to observe the operational limits of BADEDAS. The procedure followed in experiment 1 was repeated using various sampling rates and blocksizes.

As expected, for sampling rates in excess of 25 kHz A/D overrun errors were reported on the monitor console.

When the number of data words input significantly exceeded the total amount of free memory and the sampling rate exceeded the rate at which data was being transferred to the Eclipse, the expected messages, stating that the A/D had been paused while waiting for memory, started to appear on the monitor console.

The same message as in the case above was obtained by setting the block size to be small (ie. less than 100 words) and the sampling frequency high (ie. more than 10 kHz). This demonstrated that it is not efficient to specify small blocksizes when inputting multiple blocks at a high input rate. This is due to the fact that the time taken to acquire a block of memory, typically about 10 milliseconds, exceeds the time taken to input a data block.

Also measured during this experiment was the time taken to down-line load BADEDAS-M. The complete operation, including the down-loading of the bootstrap loader, the binary loader and BADEDAS-M (in excess of 6000 bytes), at a baud rate of 2400, took approximately 30 seconds.

4.6.3 EXPERIMENT C

The purpose of this experiment was to prove the operation of the D/A converter in the BADEDAS system. An oscilloscope was attached to the output of the D/A converter and a 1 kHz conversion clock used. A BADEDAS pipe command was entered specifying the AOS disk file as the data source and the D/A converter as the destination.

eg. X BAD/P FILE.DAT,DA0,15

The previously captured square wave was seen to appear on the oscilloscope.

Also measured during this experiment was the time taken for a 4096 byte file of data samples to be transmitted from the Eclipse to the Micronova. This was found to take less than 19 seconds, at 2400 baud.

4.6.4 EXPERIMENT D

The purpose of this experiment was to prove the operation of a BADEDAS user routine. A simple digital filter routine was written based on the equation:

$$Y = (A/B)*X+(C/D)*Yold$$

Where A/B and C/D are the coefficients, X is the value of an input sample and Yold, which is initially zero, takes the previously calculated value of Y. An initialisation routine was also written to allow the values of the A, B, C and D to be set by the user. Using the initialise command the the coefficients A/B and C/D were set at 1/8 and 7/8 respectively.

eg. X BAD/I UR1 1,8,7,8

Thus giving the low pass filter equation:

$$Y = 0,125\ X + 0,875\ Yold.$$

These routines are repeated in Appendix C.

One signal generator was used to provide a common clock to the A/D and D/A converters. The other signal generator was used to provide a 1 khz square wave input signal to the A/D converter. This signal was displayed on one channel of the oscilloscope and the output of the D/A was displayed on the other channel.

The A/D converter was initialised as in experiment 1 using a BADEDAS command and a BADEDAS pipe command was entered. The pipe command specified the A/D as the pipe source, the filter routine as a user routine, the D/A as the destination and a block count of 100.

DISSERTATION

... X LAD/P ADO,URO,DAO,10(

The effect of the low pass filter routine was clearly visible on the oscilloscope. The experiment was repeated using various other coefficients for the filter.

4.6.5 OTHER EXPERIMENTS

The experiments described above were repeated using small Fortran 5 programs which issued calls to the BADEDAS Fortran 5 interface routines. An example of a Fortran 5 program which achieves the same result as Experiment 1 is given in Appendix D.

CHAPTER 5

CONCLUSIONS

5.1 LIMITATIONS

It is inevitable that a software system occupying a minimal amount of memory and developed in a limited time should exhibit some limitations due to design trade-offs. Some of these limitations are summarised below.

BADEDAS user routines must be written in Micronova assembler. A more satisfactory solution would have been to permit the user to write his user routines in a high level language. If the attempt to get the RTL/2 assembler to run under AOS had been successful, this would have been possible.

BADEDAS has been designed to allow a number of pipes to function concurrently. This was done to satisfy the requirement of providing the ability to add extra device drivers to BADEDAS, without limiting the operation to at most two devices at a time. The provision for concurrent operation had a significant influence on the complexity of the design and as a result certain limitations to the use of this concurrency had to be introduced.

These limitations occur when the host computer has to perform as a source or destination in more than one pipe. Although the link between the host and the front-end is a full-duplex link, it is effectively restricted to half-duplex operation to prevent the complexity associated with the handling of a simultaneous flow of data packets and acknowledgements in both directions on the link. The result is that, if this type of operation is carried out, performance will drop. It should be noted that BADEDAS was not intended to form part of a real-time signal processing system so, generally speaking, this type of operation should not be necessary. The use of concurrent pipes will probably be most useful for cases where data needs to be

acquired simultaneously from more than one A/D converter.

BADEDAS-M error and status messages are displayed on a device separate from the user console. A more elegant solution would have been achieved if these messages could have been routed to the user console. Two problems prevented this approach. Firstly, in some cases, the program on the Eclipse which initiated the operation in the Micronova for which an error or status message is to be displayed, has terminated by the time the message needs to be displayed. Secondly, the protocol used between the Eclipse an the Micronova, assumes that all communication is initiated by the Eclipse. As a result of this, the program on the Eclipse, BADEDAS-E, has no way of knowing when the program on the Micronova, BADEDAS-M, has a message to display.

Another possible solution could have been to do away with the BADEDAS Eclipse user console and then to allow the user to communicate with the Eclipse via the BADEDAS monitor console. This solution would require the presence of firmware in the Micronova or the ability for the Micronova to boot a program from floppy disk to initiate the down-line loading of BADEDAS-M from the Eclipse.

5.2 PROBLEMS

Apart from the problems experienced with the conversion of the RTL/2 compiler (see Appendix A), the main problems were encountered in the testing and integration phase of the project.

Firstly, the incompatibilities between the Eclipse and Micronova instruction sets, particularly the stack instructions, made the testing of BADEDAS-M software on the Eclipse difficult.

Secondly, testing on the Micronova was done using the console debug monitor, which is obviously not intended as a sophisticated debugging tool. The most bothersome problem experieced with it was due to the fact that it does not disable interrupts immediately a breakpoint is encountered. As a result, it frequently happened that a device interrupt occurred between a breakpoint being encountered and the interrupts being disabled, causing unpredictable results.

acq ·ed simultaneously from more than one A/D converter.

BADEDAS-M error and status messages are displayed on a device separate from the user console. A more elegant solution would have been achieved if these messages could have been routed to the user console. Two problems prevented this approach. Firstly, in some cases, the program on the Eclipse which initiated the operation in the Micronova for which an error or status message is to be displayed, has terminated by the time the message needs to be displayed. Secondly, the protocol used between the Eclipse an the Micronova, assumes that all communication is initiated by the Eclipse. As a result of this, the program on the Eclipse, BADEDAS-E, has no way of knowing when the program on the Micronova, BADEDAS-M, has a message to display.

Another possible solution could have been to do away with the BADEDAS Eclipse user console and then to allow the user to communicate with the Eclipse via the BADEDAS monitor console. This solution would require the presence of firmware in the Micronova or the ability for the Micronova to boot a program from floppy disk to initiate the down-line loading of BADEDAS-M from the Eclipse.

5.2 PROBLEMS

Apart from the problems experienced with the conversion of the RTL/2 compiler (see Appendix A), the main problems were encountered in the testing and integration phase of the project.

Firstly, the incompatibilities between the Eclipse and Micronova instruction sets, particularly the stack instructions, made the testing of BADEDAS-M software on the Eclipse difficult.

Secondly, testing on the Micronova was done using the console debug monitor, which is obviously not intended as a sophisticated debugging tool. The most bothersome problem experieced with it was due to the fact that it does not disable interrupts immediately a breakpoint is encountered. As a result, it frequently happened that a device interrupt occurred between a breakpoint being encountered and the interrupts being disabled, causing unpredictable results.

acquired simultaneously from more than one A/D converter.

BADEDAS-M error and status messages are displayed on a device separate from the user console. A more elegant solution would have been achieved if these messages could have been routed to the user console. Two problems prevented this approach. Firstly, in some cases, the program on the Eclipse which initiated the operation in the Micronova for which an error or status message is to be displayed, has terminated by the time the message needs to be displayed. Secondly, the protocol used between the Eclipse an the Micronova, assumes that all communication is initiated by the Eclipse. As a result of this, the program on the Eclipse, BADEDAS-E, has no way of knowing when the program on the Micronova, BADEDAS-M, has a message to display.

Another possible solution could have been to do away with the BADEDAS Eclipse user console and then to allow the user to communicate with the Eclipse via the BADEDAS monitor console. This solution would require the presence of firmware in the Micronova or the ability for the Micronova to boot a program from floppy disk to initiate the down-line loading of BADEDAS-M from the Eclipse.

5.2 PROBLEMS

Apart from the problems experienced with the conversion of the RTL/2 compiler (see Appendix A), the main problems were encountered in the testing and integration phase of the project.

Firstly, the incompatibilities between the Eclipse and Micronova instruction sets, particularly the stack instructions, made the testing of BADEDAS-M software on the Eclipse difficult.

Secondly, testing on the Micronova was done using the console debug monitor, which is obviously not intended as a sophisticated debugging tool. The most bothersome problem experieced with it was due to the fact that it does not disable interrupts immediately a breakpoint is encountered. As a result, it frequently happened that a device interrupt occurred between a breakpoint being encountered and the interrupts being disabled, causing unpredictable results.

Thirdly, a hardware problem was experienced which proved to be intermittent and most evasive. On the Micronova backplane there is a daisy chain for linking the interrupt request lines of all interrupting devices. The purpose of it is to resolve contention between interrupting devices. Unfortunately, this daisy chain had been incorrectly wired, a problem not exposed by the diagnostic programs possessed by the department and which did not seem to affect other software frequently run on the Micronova. The problem manifested itself as a software timing problem and much of the interrupt handling software was rewritten a number of times in an attempt to solve it. Matters were aggravated by the intermittency of the problem. Six weeks after first encountering the problem, a Perseus service technician diagnosed and remedied it.

5.3 CONCLUSION

In spite of the problems encountered, the prcject was successful in meeting the requirements. BADEDAS-M is a compact and yet efficient program which makes optimum use of the Micronova's memory space and CPU. BADEDAS-E provides a user-friendly interface and makes efficient use of AOS for I/O. The protocol used for communication between the Micronova and the Eclipse is both secure and efficient.

In conclusion, I express my gratitude to Mr T Pierce of Software Sciences and Mr I Macleod for their enthusiastic advice and help with the attempt to run the RTL/2 compiler under AOS, Mr M Kagan and Mr B Currer of Perseus, for their help in the attempt to get the AOS Convert utility to work on the problematic RTL/2 files, Dr A Walker for all the knowledge he passed on to me concerning the Micronova, the Eclipse and AOS, and Professor Hanrahan who acted as supervisor for this project.

REFERENCES

Bartlett KA, Scantlebury RA and Wilkinson PT (1969)
"A Note on Reliable Full-Duplex Transmission over Half-Duplex Lines"
CACM vol 12 no. 5

Bourne SR (1978)
"The Unix Shell"
Bell System Journal vol 57 no. 6 part 2

Foster CC (1981)
"Real-Time Programming - Neglected Topics"
Addison-Wesley

Heider G (1982)
"Let Operating Systems Aid in Component Design"
Computer Design Sept 1982

Levi P (1979)
"The System Architecture of Data Acquisition Systems"
Euromicro Journal (5)

Metzger PW (1981)
"Managing a Programming Project" (2nd ed)
Prentice-Hall

Prucnal (1982)
"Experiments in Digital Signal Processing"
IEEE Transactions on Education Vol E-25 no. 3

Rowe LA and Birman K (1982)
"A Local Network Based on the Unix Operating System"
IEEE Transactions on Software Eng. vol SE 8 no. 2

Sommerville I (1982)
"Software Engineering"
Addison-Wesley

Taeymans JR and De Bruyn PF (1982)
"The Use of a Microprocessor as a Front-end Processor for Automatic Measurement"
Microprocessing and Microprogramming" 9 1982

Thompson A (1982)
"Multiprocessor Network of Process Control"
4th year project 1982

Yourdon E (1972)
"Design of On-line Computer Systems"

Prentice-Hall

Yourdon E (1975)
"Techniques of Program Structure and Design"
Prentice-Hall

Ying and Currer B (1978)
"A Multi-Processor End-to-End Communications Package
for Industrial Applications"
SACAC Symposium on Real-Time Software for
Industrial Applications CSIR 1978

Wirth N (1971)
"Program Development by Stepwise Refinement"
CACM vol 14 no. 4

APPENDIX A

RTL/2

A.1 SCOPE

The purpose of this Appendix is to document that part of the BADEDAS effort which was concerned with the attempt to convert RDOS RTL/2 to run under AOS.

A.2 BACKGROUND

The benefits of using a high-level language as opposed to assembler language for programming are well known. Some of these benefits which of relevance to this project are:

* increased programmer productivity
* more reliable programs
* more understandable programs
* easily maintainable programs

In order to take advantage of these benefits, the possibility of using a high level language for the implementing BADEDAS was investigated. The requirements of a high-level language as identified for the implementation of the BADEDAS design are as follows:

* The compiler for the language must be able to run on the Eclipse under AOS.
* The compiler must be able to generate stand-alone object code, ie. object code which does not include calls to AOS.

* The object code must be efficient in memory space used and execution speed.

The language RTL/2 (Real-Time Language 2) was found to come close to fulfilling these requirements. The only exception being that no RTL/2 compiler exists for execution under AOS. An RTL/2 compiler does however exist for another Eclipse operating system, RDOS.

A.3 THE SUITABILITY OF RTL/2 FOR THE IMPLEMENTATION OF BADEDAS

RTL/2 is a high level language designed for use in real-time control applications and since data acquisition and data dissemination are important activities in real-time control applications, it appeared that RTL/2 would be a suitable language for the implementation of BADEDAS. There are numerous specific reasons for RTL/2 being a suitable language for the implementation of BADEDAS.

A limitation of most high-level languages is that compiled programs need to be executed under the control of an operating system. The main reason for this being that the compiled code makes use of operating system facilities such as the I/O drivers. It is a simple and well defined exercise to create an RTL/2 program which can run as a stand-alone program (ie. without an operating system). When doing this it is necessary to provide a base module and I/O routines for any unusual I/O devices to be used. The base module sets up the run-time environment needed by the program.

RTL/2 compilers often provide cross-compilation facilities for other computers. The RDOS RTL/2 compiler is an example of this as it can generate either Eclipse or Nova object code.

BADEDAS makes use of a high degree of concurrency or multi-tasking. In RTL/2, a multi-tasking program, such as BADEDAS, needs a multi-task monitor to provide the correct environment for it. Apparently, a number of general purpose multi-task monitors written in RTL/2 do exist and BADEDAS could have made use of one of these with little or no tailoring required. This would have substantially reduced the amount of system management software coded for BADEDAS.

A further benefit which RTL/2 would have provided was the ability for user routines to be written in a high-level language instead of assembler.

A.4 THE ATTEMPT TO CONVERT RDOS RTL/2 TO AOS RTL/2

The main problem with using RTL/2 as the implementation language for BADEDAS stems from the fact that an RTL/2 compiler did not exist for ACS based systems. Since RTL/2 appeared to be a highly suitable language for BADEDAS, it was decided to attempt to convert RDOS RTL/2 to run under AOS.

RTL/2 compilers consist, generally, of the same basic parts:

* the front-end - this checks the source and produces a version of the program in a machine independent format
* the back-end - this takes the machine independent module and produces an object code module for the target machine
* the interface - a collection of modules which provide for the communication between the front-end and back-end
* the base - this creates the necessary environment for the compiler, including provision of I/O routines for the file handling of the source and object file, the listing and error messages.

The front-end, back-end and interface programs are written in pure RTL/2 (ie. there are no operating system calls embedded in these programs) and they interface with the operating system through the base program. The base program is the only part of the compiler which is operating system dependent. Fortunately the source code of the compiler base program is provided, so, theoretically, this is all that needs to be altered if one wishes to run the compiler under another operating system.

Mr. I Macleod, senior lecturer in the Department had in his possession a preliminary release of RDOS RTL/2. It was clear that this compiler had the standard RTL/2 compiler format.

The RDOS RTL/2 compiler was developed by Software Sciences Limited of England. Communication was established with Software Sciences and we managed to obtain the standard release of RDOS RTL/2 on magnetic tape. Although the compiler is usually

supplied in absolute file format, Software Sciences were kind enough to provide us with the separate relocatable binary modules which together make up the compiler.

Briefly, the task of getting the compiler to execute under AOS consisted of:

* converting all relocatable binary modules from RDOS format to AOS format
* rewriting the disk I/O routines and overlay routines needed by the compiler
* making several small changes to the base program.

Once these changes had been applied, the RTL/2 compiler would hopefully run under AOS, but because the RDOS RTL/2 compiler back-end had not been altered, the compiler would produce RDOS relocatable binary format object code.

The conversion of RDOS relocatable binary format to AOS relocatable binary format is not a trivial task, as there is a large number of block types in both formats and it is not a simple block-to-block conversion. Fortunately, however, there exists an AOS utility called Convert which performs this conversion.

Unfortunately, a small number of the relocatable binary files would not convert, causing the utility to loop endlessly. Perseus, Data General's South African representatives, were most helpful in the attempt to overcome this problem. Altogether four different releases of the Convert utility were each tried under three different releases of AOS, but to no avail. Finally Perseus claimed that the files could not be in valid RDOS relocatable binary format. Software Sciences claimed that this was incorrect as the files were acceptable as input to the RDOS relocatable binary loader. The large number of different block types and the the size of the files made it virtually impossible to determine whether the offending files did, in fact, contain any blocks of an invalid format.

After further communication with Software Sciences, they offered to send the compiler generated source listings of the offending modules. Unfortunately, in spite of a polite request for these listings, they did not arrive.

DISSERTATION

At an already late stage of the project a decision was made to abandon RTL/2 as the implementation language and with it the many benefits.

APPENDIX B

PROTOCOL

All communication between the host computer and the Micronova is accomplished using sequences of one or more bytes called packets.

There are two types of multibyte packets and a number of single byte packets.

B.1 MULTIBYTE PACKETS

Multibyte packets are either command or data packets. Command packets are used to convey command blocks from the host to the Micronova. A command block never requires more than a single packet to be transmitted. One or more data packets are required for transmitting each data block between the host and the Micronova.

Command and data packets have the same format:

BYTE	0	SOM (start-of-packet code)
BYTE	1	packet ID
BYTE	2	packet type
BYTE	3	packet source
BYTE	4	length of packet contents-1
BYTES	5 to N	packet contents
BYTE	N+1	ETB or EOM (end-of-block or end-of-packet code)
BYTE	N+2	checksum

NOTES:

* The following codes are used in multibyte packets:
  SOM 02H start-of-packet
  EOM 04H end-of-packet
  ETB 17H end-of-transmission block

* The packet ID serves to distinguish packets. Each packet is allocated an ID different from the previous packet from the same source. This avoids duplication of packets following the non-arrival or corruption of an acknowledgement.

* Packet type is "C" for a command packet and "D" for a data packet.

* The length field of a packet contains the packet byte length less one. A packet may therefore contain up to 256 bytes of data or command.

* The contents of a packet may consist of any sequence of 8 bit binary codes.

* The second last byte of a command packet will be the ETB code, to signify that the packet contains a complete transmission block.

* The second last byte of all data packets except the last packet of a data block will be the EOM code.

* The second last byte of the last packet of a data block will be the ETB code, to signify that this is the end of a transmission block.

* The last byte of the packet will contain a checksum, which is an XOR function of all the other bytes in the packet.

B.2 SINGLE BYTE PACKETS

There eight single byte packets, whose job it is to manage the flow of multibyte packets between the host and the Micronova.

ENQ	05H	enquiry
SACK	06H	sync-acknowledge
XON	11H	continue transmission
ACK0	12H	acknowledge 0
XOFF	13H	hold transmission
ACK1	14H	acknowledge 1
NAK	15H	negative acknowledge
SYNC	16H	synchronise

B.2.1 ACK0, ACK1 And NAK

All multibyte packets sent between the host and the Micronova must be acknowledged, either positively or negatively, by the receiver. Negative acknowledgement is achieved by sending an NAK code and positive acknowledgement by sending either ACK0 or ACK1.

ACK0 and ACK1 are used to implement alternating acknowledgement. Using this scheme positive acknowlegement alternates between ACK0 and ACK1, with the first positive acknowledgement following synchronisation with the host being ACK0.

The alternating acknowledgement scheme prevents the sender from getting out of synchronism with the receiver following a time-out. Since AOS buffers packets for BADEDAS-E, the possibility exists that more than one acknowledgement is buffered following a time-out retransmission, thus resulting in the acknowledgment read by BADEDAS-E not being the response to the packet just sent.

B.2.2 SYNC And SACK

When BADEDAS-E wishes to commence communication with BADEDAS-M, it transmits a SYNC code. BADEDAS-M must then clear the variables containing the packet-ID's of the most recent packets sent and received and it must set the next acknowledge to be sent and the next acknowledge expected each to ACK0. Once this has been done, BADEDAS-M responds with a SACK, sync-acknowledge, code.

B.2.3 ENQ

All communication between BADEDAS-E and BADEDAS-M is solicited by BADEDAS-E. When BADEDAS-E is expecting data from BADEDAS-M, it sends an ENQ code to prompt the transmission of that data.

B.2.4 XON And XOFF

All data blocks input by BADEDAS source devices occupy memory until the block has been output and the memory area released. If the rate at which data blocks are being input exceeds that at which they are being output, then it is likely that BADEDAS-M will run out of available memory. If BADEDAS-E attempts to communicate with BADEDAS-M when there is not sufficient available memory to buffer the incoming packet, then BADEDAS-M will send a XOFF code to BADEDAS-E in order to signal that it must hold transmission. BADEDAS-E must then wait until BADEDAS-M sends a XON code before it resumes transmission.

B.3 TIMEOUTS

A significant aspect of this protocol is the timeout mechanism. Two types of time-out are used. An intercharacter timeout and an acknowledgement timeout.

If more than the specified intercharacter timeout period elapses between the receipt of two bytes of the same packet, then the packet received this far is abandoned. This timeout guards against the receiver becoming confused by the receipt of incomplete packets.

If after a packet has been transmitted, more than the specified acknowledgement timeout period elapses without the receipt of an acknowledgement, then the packet is retransmitted. This timeout guards against a deadlock occurring after an acknowledgement fails to arrive or gets corrupted in transmission. If the receiver happened to receive the previous packet correctly, it will detect that the the retransmitted packet is a duplicate and will repeat the latest acknowledgement sent.

APPENDIX C

EXAMPLE OF BADEDAS USER ROUTINE

The following is an example of a user routine written for use in BADEDAS. The routine is an implemention of a simple digital low pass filter based on the equation

Y = (A/B)*X+(C/D)*Yold

where A/B and C/D are the coefficients ((A/D)+(C/D) must = 1), X take on the value of each input sample to be transformed and Yold, initially = zero, always equals the previously calculated value of Y. A, B, C and D may be initialised by the user.

eg. X BAD/I UR1,1,8,7,8

```
        .TITL   URTN1
        .ENT    URTN1
        .NREL
URTN1:  ; THIS ROUTINE IS THE IMPLEMENTATION OF THE SIMPLE
        ; DIGITAL FILTER FUNCTION Y = (A/B)*X+(C/D)*YOLD
        ; A, B, C AND D HAVE THE DEFAULT VALUES OF 1, 8, 7 & 8
        ; RESPECTIVELY. THEIR VALUES MAY BE CHANGED USING THE
        ; INITIALISE COMMAND.
        ; ON ENTRY ACCUMULATOR 2 POINTS TO THE BLOCK OF SAMPLES
        ; TO BE TRANSFORMED.
        ;
        SAV                     ;SAVE ACCUMULATORS
        SUB     AC0,AC0         ;CLEAR AC0
        STA     AC0,YOLD        ;CLEAR YOLD
        LDA     AC0,1,AC2       ;AC0 = NO. OF SAMPLES IN BLOCK
        SNEZ    AC0             ;IS BLOCK EMPTY
        JMP     URRET           ;YES - RETURN TO SCHEDULER
        STA     AC0,SCNT        ;STORE SAMPLE COUNT
UR1:    LDA     AC0,3,AC2       ;AC0 = SAMPLE VALUE
        JSR     FILTA           ;TRANSFORM SAMPLE VALUE
        STA     AC0,3,AC2       ;STORE TRANSFORMED SAMPLE
        INC     AC2,AC2         ;INCREMENT SAMPLE POINTER
        DSZ     SCNT            ;DECREMENT SAMPLE COUNT
        JMP     UR1             ;GO AND PROCESS NEXT SAMPLE
URRET:  RET                     ;COUNT = ZERO, RETURN
```

```
FILTA:  ; THIS SUBROUTINE IMPLEMENTS THE FILTER FUNCTION
        ; NOTE A/B AND C/D ARE EFFECTIVELY RECALCULATED FOR
        ; EACH VALUE OF Y. THIS IS DONE TO AVOID THE NEED
        ; TO DEAL IN FRACTIONS.
        SAV                         ;SAVE ACCUMULATORS
        LDA     AC1,A               ;AC1 = A
        JSR     MULTP               ;AC0 = A*X
        LDA     AC1,B               ;AC1 = B
        JSR     DIVID               ;AC0 = (A/B)*X
        MOV     AC0,AC2             ;AC2 = (A/B)*X
        LDA     AC0,YOLD            ;AC0 = YOLD
        LDA     AC1,C               ;AC1 = C
        JSR     MULTP               ;AC0 = C*YOLD
        LDA     AC1,D               ;AC1 = D
        JSR     DIVID               ;AC0 = (C/D)*YOLD
        ADD     AC2,AC0             ;AC0 = Y = (A/B)*X+(C/D)*YOLD
        STA     AC0,YOLD            ;YOLD = Y
        RETN    AC0                 ;RETURN WITH Y IN AC0
```

```
MULTP:  ;THIS SUBROUTINE IMPLEMENTS THE MULTIPLY FUNCTION.
        ;IT MULTIPLIES THE CONTENTS OF AC0 BY THOSE OF AC1
        ;AND RETURNS THE RESULT IN AC0
        ;
        MOV     AC0,AC2         ;AC2 = MULTIPLICAND
        SUB     AC0,AC0         ;CLEAR AC0
        MUL                     ;MULTIPLY
        MOV     AC1,AC0         ;AC0 = RESULT
        RETN    AC0             ;RETURN WITH RESULT IN AC0

DIVID:  ;THIS SUBROUTINE IMPLEMENTS THE DIVIDE FUNCTION.
        ;IT DIVIDES THE CONTENTS OF AC0 BY THOSE OF AC1
        ;AND RETURNS THE QUOTIENT IN AC0.
        ;
        SAV                     ;SAVE ACCUMULATORS
        MOV     AC1,AC2         ;AC2 = DIVISOR
        MOV     AC0,AC1         ;AC1 = DIVIDEND
        SUB     AC0,AC0         ;CLEAR AC0
        DIV                     ;DIVIDE
        MOV     AC1,AC0         ;AC0 = QUOTIENT
        RETN    AC0             ;RETURN WITH RESULT IN AC0
```

Author Chapman M J
Name of thesis The implementation of a data acquisition and dissemination system using a microcomputer as an intelligent front-end processor to a multi-user minicomputer 1984

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed, or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.