<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
</table>
| 0    | Mode               | 0 = programmed I/O mode  
1 = data channel mode  
for use in BAFEDAS bit 0 must = 1 |
| 1-2  | Clock              | 01 = auto data channel (max rate)  
10 = internal clock pulse  
11 = external clock pulse |
| 3    | Channel type       | 0 = differential inputs  
1 = single-ended inputs |
| 4-7  | Final channel      | Selects 0 to 17 (octal) as last channel address |
| 8 to 11 | Reserved            | reserved |
| 12 to 15 | Initial channel  | Selects 0 to 17 (octal) as first channel address |

**Notes:**

1. For BAFEDAS, bit 0 of the A/D mode word must always be set to 1, as the A/D may only be used in data channel mode.

2. A clock is needed to supply a constant sampling rate. Normally this clock will be supplied externally by the user. An external clock may be used to accurately control the start of a set of conversions, by using a logic gate to enable the clock the instant (following a pipe command specifying the A/D as the source device) at which sampling is to begin. The internal clock may also be used to set the sampling rate. The period of the internal clock may be set to between 33.3 and 450 microseconds by adjusting a potentiometer on the A/D controller board.

3. The initial and final channel fields allow the user to specify the sampling of a number of signal sources one at a time. Sampling starts at the initial channel and passes sequentially to each of the channels following it in numerical order, until it reaches the final channel, after which it wraps around to the initial channel. The single-ended channels are numbered 0 to 17 (octal) and the differential channels are numbered 0 to 7 (octal) or 10 to
For sampling from multiple sources it is advisable to consult the programmer's reference manual, as there are some peculiarities in the numbering of channels under certain circumstances. When sampling from a single channel, the initial and final channel numbers must both be set to that channel's number.

6.2 Model 4224 Digital To Analog Converter Interface

The model 4224 D/A converter allows the Micronova to output analog signals via two analog output channels. Each channel can drive a load of up to 5 mA. The D/A converter channels can be individually jumper-selected for operation in one of the voltage ranges 0 to 5v, 0 to 10v, -5 to +5v or -10 to +10v. For each of these ranges the D/A converter can supply up to 4096 discrete analog values from digital data with a resolution of 12 bits. The data supplied to the D/A converter may be selected by jumper to be interpreted either as two's complement or offset binary. The format of each word of data to be output is as follows:

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>sign</td>
<td>in two's complement: 0=pos, 1=neg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in offset binary: 1=pos, 0=neg</td>
</tr>
<tr>
<td>1 to 11</td>
<td>magnitude</td>
<td>11 bit digital representation of analog value to be converted</td>
</tr>
<tr>
<td>12 to 13</td>
<td>reserved</td>
<td></td>
</tr>
<tr>
<td>14 to 15</td>
<td>z-pulse</td>
<td>00 = 1.5 volts +/- DC level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01 = 1.0 volts +/- DC level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = 0.5 volts +/- DC level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 = no z-pulse</td>
</tr>
</tbody>
</table>

After start-up, the D/A converter is set up in a default mode. In order to use the D/A converter in a BADEDAS pipe in any mode other than its default mode, it must be initialised. There is only one parameter which can be provided when initialising the D/A converter, the mode word. If no parameter is provided in the initialise command, the default mode word is used.
The mode word has the following format.

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>mode</td>
<td>0 = program I/O mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = data channel mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for use in BADELAS bit 0 must = 1</td>
</tr>
<tr>
<td>1 - 2</td>
<td>clock</td>
<td>0l auto data channel (max rate)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 internal clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 external clock</td>
</tr>
<tr>
<td>3</td>
<td>alternate</td>
<td>1 = alternate between X and Y converters starting with the bit selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>by bit 15</td>
</tr>
<tr>
<td>4</td>
<td>scope mode</td>
<td>0 = z axis pulse generated after conversion by selected converter X or Y</td>
</tr>
<tr>
<td>5</td>
<td>non-store</td>
<td>1 = open collector true low output available (scope control)</td>
</tr>
<tr>
<td>6</td>
<td>write-through</td>
<td>1 = open collector true low output available (scope control)</td>
</tr>
<tr>
<td>7</td>
<td>erase</td>
<td>1 = 2 ms pulse to open collector true low output (scope control)</td>
</tr>
<tr>
<td>8 to 14</td>
<td>-</td>
<td>reserved</td>
</tr>
<tr>
<td>15</td>
<td>D/A</td>
<td>0 = D/A converter X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = D/A converter Y</td>
</tr>
</tbody>
</table>

Notes:

1. The default mode word for the D/A converter is 140000 octal.

2. For BADELAS, bit 0 of the D/A mode word must always be set to 1, as the D/A may only be used in data channel mode.

3. A clock is needed to supply a constant conversion rate. Normally this clock will be supplied externally by the user. The internal clock period is adjustable within the range 15 microseconds to 150 microseconds, using a potentiometer on the D/A controller board.
4. The two D/A converter channels are referred to as X and Y. Either X or Y or alternating channels may be selected, but both channels cannot be selected simultaneously.

5. The D/A converter provides a number of oscilloscope control functions. When selected, these functions provide a means to control beam intensity, storage scope erasing and plotting. In addition, non-store and write-through scope characteristics can be selected. In scope mode, a Z-axis pulse is generated each time a conversion is completed by either the X or Y converter (this is jumper-selectable, on the D/A controller board). The Z-pulse is the scope intensity or brightness pulse occurring 7 microseconds after the conversion.

7.0 USER ROUTINES

Up to three Micronova assembly language user routines may be included in the DAS software for controlling the Micronova. These routines are allocated the DAS mnemonics UKO, U1 and U2. A user routine must have the entry point URTNX, where X is 0, 1 or 2 depending on the DAS mnemonic which is to refer to the user routine.

eg. for the mnemonic U1, the user routine must have the entry point URTN1.

It is also possible for the user to include an initialisation routine for a user routine. The initialisation must have the entry point URIX, where X is again 0, 1 or 2 depending on the mnemonic to be used for the user routine.

A DAS user routine is intended as a facility by which the user can specify a limited transformation to be performed on blocks of input data.

A user routine, which has been included in a DAS pipe, is called whenever a block of data has been input by the pipe's source device. On entry to the routine accumulator 2 will contain the address of the data block. On exit from the user routine accumulator 2 must contain the address of the transformed data block.
The first three words and the last word of the data block are used for storing accounting information about the block. The first bit, bit 0, of the first word, word 0, will always be set to 1, and bits 1 to 15 of this word will contain the total block size in words. Word 1 will contain the data count, which is the number of words of data in the data block. Word 2 and the last word are both reserved.

In DADEDAS a simple transformation is regarded as one in which the result of the transformation can occupy the same memory block as was occupied by the transformation's input data block. When this is the case, the complexity of the user routine is kept very low and little knowledge of the internal functioning of DADEDAS is required. If the transformation results in a larger data block than the input to the transformation, or if the output of the transformation must occupy a memory area different from that occupied by the input data block, the user routine needs to acquire a suitable memory block for the output of the transformation and it needs to release the input data block once the transformation is complete. These aspects are discussed further in "Complex User Routines". An example of a user routine is provided in the appendices.

The user routine initialisation routine is called whenever an initialise command or subroutine call is issued for the user routine. On entry to the initialisation routine, accumulator 2 will point to the command block. Word 0 and word 2 of the command block are reserved and word 1 will contain the number of bytes of command data contained in the command block. Word 3 onwards will contain the command data. The left byte of word 3 will contain the command code, an I for an initialise command, and the right byte will contain the DADEDAS internal code for the user routine being initialised. Following word 3 will be the parameters of the initialise command. Each parameter will occupy a 16 bit word and they will be in the order in which they were entered by the user. Note that number of parameters passed must be obtained from word 1 of the command block. Word 1 is the command byte count and since this count includes the command type and user routine code it will be 2 if no parameters were provided, 4 if 1 parameter was provided (remember each parameter occupies 16 bits = 2 bytes) and so on.

8.0 COMPLEX USER ROUTINES

It should be emphasised that DADEDAS user routines are intended for the performing of simple transformations, if complex transformations need to be performed, they are probably best done on the Eclipse.
User routines which need to acquire blocks of memory are regarded as complex. In order to write such a user routine the user needs to be acquainted with some of the internal functions of BADGEDAS, which are documented in the BADGEDAS Technical Manual. The simplest approach to acquiring a memory block is described below.

To acquire a block of memory, the user must place the negative of the blocksize (number of words) in accumulator 2 and the call USGET.

eg. CALL 0, +1
    USGET

On return from USGET accumulator 2 should contain the word address of the memory block. If on return accumulator 2 still contains the negative blocksize, then the memory request could not be satisfied and the user routine must repeat the call until the request is satisfied. The user should however realise that this could lead to a deadlock situation, especially when the source device is inputting an amount of data which exceeds the pipe output rate. The amount of available memory is displayed on the console monitor after the Micronova has been down-line loaded.

After acquiring a memory block and completing its transformation, the user routine should perform the following actions:

1. The number of words of data in the new block must be stored in word 1 of the block (the first word of the block is word 0).

2. The last word of the input block must be copied to the last word of the new block. The address of the last word must be obtained by adding bits 1-15 of word 0 (the blocksize) to the address of word 0 and subtracting one. Note that bit 0 of word 0 will always be set on, so it must be masked out to obtain the blocksize. Note also that the blocksize may in some cases be larger than that requested, so it is always necessary to obtain the actual blocksize from word 0.

3. The input block must be released by calling PREMRR, with the address of the block to be released in accumulator 2.
4. On return from the user routine the address of the transformed block should be in accumulator 2.

9.0 ASSEMBLING AND BINDING USER ROUTINES

User routines written for BADEDAS must be assembled on the Eclipse using the AOS CLI macro command N\ASM.

eg. To assemble user routine URTH2 the command line entered is as follows:

MNASM URTH2

This command ensures that the user routine is assembled into pure Micronova object code. Following an error-free assembly, the user routine must be linked with the BADEDAS Micronova software. This is done using the macro command MNBIND.

eg. MNBIND URTH2

This command relieves the user from having to name all the BADEDAS software subsystems in the link command and it replaces the current version of BADEDAS-M.
APPENDIX A

FORTRAN EXAMPLE

The following FORTRAN 5 program makes use of BADEAS subroutine calls to do the following:

* to initialise the A/D converter to operate in data channel mode with an external clock, to sample from channel 1 and to have a input word count of 1024.

* to set up BADEAS pipe between the A/D converter and the host computer, which includes the user routine and sets the block count to 1.

* to read the block input by the A/D converter from the Micronova.

The program finally writes the block into an AOS disk file.

```
INTEGER IARRAY(1024), ARGARRAY(2)
OPEN 20, "FILE.RAT", ATT="RO", LEN=2048
ARGARRAY(1)=1024
ARGARRAY(2)=170000K
CALL BADI("ADO", ARGARRAY, 2)
CALL BADP("ADO", "UR1", R20, 1)
CALL BAUR(IARRAY, 1024, ISIZE)
WRITE BINARY (20) IARRAY
CLOSE 20
END
```

APPENDIX B

EXAMPLE OF BADEPAS USER ROUTINE

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

\[ Y = \left(\frac{A}{D}\right)X + \left(\frac{C}{D}\right)Y_{\text{old}} \]

where \(\frac{A}{B}\) and \(\frac{C}{D}\) are the coefficients (\(\frac{A}{B} + \frac{C}{D}\) must = 1), \(X\) takes on the value of each input sample to be transformed and \(Y_{\text{old}}\), 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 URL.1, 8, 7, 8
TITL: URTH1
JNT: URTH1
NREL
URTH1: ; THIS ROUTINE IS THE IMPLEMENTATION OF THE SIMPLE
; DIGITAL FILTER FUNCTION \( Y = \frac{A}{B}X + \frac{C}{D}YOLD \)
; \( A, B, C \) AND \( D \) HAVE THE DEFAULT VALUES OF 1, 6, 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.

SAVE
STUB ACO,ACO
STUB ACO,YOLD
STUB LDA ACU,1,AC2
STUB SUB ACU
STUB JMP URRET
STUB STA ACO,SCNT
STUB JSR FILR
STUB STA ACO,3,AC2
STUB ICN AC2,AC2
STUB DES SCNT
STUB JMP UR1
STUB URRET,RET

; SAVE ACCUMULATORS
; CLEAR ACO
; CLEAR YOLD
; ACU = NO. OF SAMPLES IN BLOCK
; IS BLOCK EMPTY
; YES = RETURN TO SCHEDULER
; STORE SAMPLE COUNT
; ACO = SAMPLE VALUE
; TRANSFORM SAMPLE VALUE
; STORE TRANSFORMED SAMPLE
; INCREMENT SAMPLE POINTER
; DECREMENT SAMPLE COUNT
; GO AND PROCESS NEXT SAMPLE
; COUNT = ZERO, RETURN
THE ROUTINE IS THE IMPLEMENTATION OF THE SIMPLE DIGITAL FILTER FUNCTION \( Y = \frac{(A/B) \times X + (C/L)}{Y} \).

A, B, C and D have the default values of 1, 6, 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
SUB ACO,ACO :SAVE ACUMULATORS
STA ACO,ACO :CLEAR ACO
STA ACO,YOLD :CLEAR YOLD
LOA ACO,1,AC2 :ACO = NO. OF SAMPLES IN BLOCK
SHR ACO :IS BLOCK EMPTY
JMP UNRET :YES = RETURN TO SCHEDULER
STC ACO,SCNT :STORE SAMPLE COUNT
UR1: LDA ACO,3,AC2 :ACO = SAMPLE VALUE
JCR FILTA :TRANSFORM SAMPLE VALUE
STA ACO,3,AC2 :STORE TRANSFORMED SAMPLE
INC AC2,AC2 :INCREMENT SAMPLE POINTER
DSZ SCNT :DECREMENT SAMPLE COUNT
JHP UR1 :GO AND PROCESS NEXT SAMPLE
JUMP; COUNT = ZERO, RETURN
```
FILTA: This subroutine implements the filter function.
Note that 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
LDA AC1,A
JSR MULTP
LDA AC1,B
JSR DVID
MOV AC0,AC2
LDA AC0,YOLD
JSR MULTP
LDA AC1,C
JSR DVID
ADD AC2,AC0
STA AC0,YOLD
RETN AC0

; SAVE ACCUMULATORS
; AC1 = A
; AC0 = A*X
; AC1 = B
; AC0 = (A/B)*X
; AC2 = (A/B)*X
; AC1 = C
; AC0 = C*YOLD
; AC1 = D
; AC0 = (C/D)*YOLD
; AC0 = Y = (A/B)*X+(C/D)*YOLD
; YOLD = Y
; RETURN WITH Y IN AC0
MULTP: ; THIS SUBROUTINE IMPLEMENTS THE MULTIPLY FUNCTION.
; IT MULTIPLIES THE CONTENTS OF ACO BY THOSE OF AC1
; AND RETURNS THE RESULT IN ACO.

MOV ACO, AC2 ; AC2 = MULTIPLICAND
SUB ACO, ACO ; CLEAR ACO
MUL ACO, AC1 ; MULTIPLY
MOV AC1, ACO ; ACO = RESULT
RETN ACO ; RETURN WITH RESULT IN ACO

DIVD: ; THIS SUBROUTINE IMPLEMENTS THE DIVIDE FUNCTION.
; IT DIVIDES THE CONTENTS OF ACO BY THOSE OF AC1
; AND RETURNS THE QUOTIENT IN ACO.

SAV ; SAVE ACCUMULATORS
MOV AC1, AC2 ; AC2 = DIVISOR
MOV ACO, AC1 ; AC1 = DIVIDEND
SUB ACO, ACO ; CLEAR ACO
DIV ACO, AC1 ; DIVIDE
MOV AC1, ACO ; ACO = QUOTIENT
RETN ACO ; RETURN WITH RESULT IN ACO
URI1: ;USER ROUTINE INITIALISE ROUTINE.
;THE ROUTINE IS CALLED BY THE BAREBAS COMMAND
;INTERPRET WHENEVER AN INITIALISE COMMAND IS
;RECEIVED OR URI1 ( USER ROUTINE 0 ).
;ON ENTRY AC2 POINTS TO THE COMMAND BLOCK.

;SAVE ACCUMULATORS
LDA AC3, 1, AC2 ;AC3 = NO. OF BYTES IN COMMAND BLOCK
LDA AC0, FOUK ;ACO = 4
SGE AC3, AC0 ;HAS AT LEAST 1 PARAM BEEN PROVIDED?
JMP U1RET ;NO = SO RETURN
LDA AC0, 4, AC2 ;YES = AC0 = PARM 0
STA AC0, A
LDA AC0, SIX ;AC0 = 6
SGE AC3, AC0 ;HAVE AT LEAST 2 PARAMS BEEN PROVIDED?
JMP U1RET ;NO = SO RETURN
LDA AC0, 5, AC2 ;YES = AC0 = PARM 1
STA AC0, B
LDA AC0, EIGHT ;AC0 = 8
SGE AC3, AC0 ;HAVE AT LEAST 3 PARAMS BEEN PROVIDED?
JMP U1RET ;NO = SO RETURN
LDA AC0, 6, AC2 ;YES = AC0 = PARM 2
STA AC0, C
LDA AC0, TEN ;AC0 = 10
SGE AC3, AC0 ;HAVE AT LEAST 4 PARAMS BEEN PROVIDED?
JMP U1RET ;NO = SO RETURN
LDA AC0, 7, AC2 ;YES = AC0 = PARM 3
STA AC0, D
U1RET: RET
;RETURN
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.