

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/

BITS	NAME	FUNCTION
0	mode	0 = programmed I/O mode 1 = data channel mode for use in BADEDAS 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
12 to 15	initial channel	selects 0 to 17 (octal) as first channel address

Notes:

1. For BADEDAS, 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

17 (octal). 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:

BITS	NAME	FUNCTION
0	sign	in two's complement: 0=pos, 1=neg in offset binary: 1=pos, 0=neg
1 to 11	magnitude	11 bit digital representation of analog value to be converted
12 to 13		reserved
14 to 15	z-pulse	00 = 1,5 volts +/- DC level 01 = 1,0 volts +/- DC level 10 = 0,5 volts +/- DC level 11 = no z-pulse

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.

BITS	NAME	FUNCTION
0	mode	0 = program I/O mode 1 = data channel mode for use in BADEDAS bit 0 must = 1
1 - 2	clock	01 auto data channel (max rate) 10 internal clock 11 external clock
3	alternate	1 = alternate between X and Y converters starting with the bit selected by bit 15
4	scope mode	0 = z axis pulse generated after conversion by selected converter X or Y
5	non-store	1 = open collecter true low output available (scope control)
6	write-through	1 = open collecter true low output available (scope control)
7	erase	1 = 2 ms pulse to open collecter true low output (scope control)
8 to 14	-	reserved
15	D/A	0 = D/A converter X 1 = D/A converter Y

Notes:

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

2. For BADEDAS, 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 BADEDAS software for controlling the Micronova. These routines are allocated the BADEDAS mnemonics UR0, UR1 and UR2. A user routine must have the entry point URTNX, where X is 0, 1 or 2 depending on the BADEDAS mnemonic which is to refer to the user routine.

eg. for the mnemonic UR1, 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 BADEDAS 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 BADEDAS 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 BADEDAS 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 BADEDAS 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 BADEDAS 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 BADEDAS 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 BADEDAS, which are documented in the BADEDAS 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    @.+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 approaches the amount of available Micronova memory at an input rate 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 FREBLK, with the address of the block to be released in accumulator 2.

```
        JSR     @.+1
                FREBLK
```

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 MNASM.

eg. To assemble user routine URTN2 the command line entered is as follows:

```
MNASM URTN2
```

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 URTN2
```

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

BADEDAS FORTRAN EXAMPLE

The following Fortran 5 program makes use of BADEDAS 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 BADEDAS pipe between the A/D converter and the host computer, which includes the user routine 1, 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.DAT",ATT="RO",LEN=2048
ARGARRAY(1)=1024
ARGARRAY(2)=170000K
CALL BADI("AD0",ARGARRAY,2)
CALL BADP("AD0","UR1",HD0,1)
CALL BADR(IARRAY,1024,ISIZE)
WRITE BINARY (20) IARRAY
CLOSE 20
END
```

APPENDIX B

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 takes 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
```

```
            .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 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                     ;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
```

```
URI1:   ;USER ROUTINE INITIALISE ROUTINE.
        ;THIS ROUTINE IS CALLED BY THE BADEDAS COMMAND
        ;INTERPRETER WHENEVER AN INITIALISE COMMAND IS
        ;RECEIVED FOR URI ( USER ROUTINE 0 ).
        ;ON ENTRY AC2 POINTS TO THE COMMAND BLOCK.
        ;
        SAV                     ;SAVE ACCUMULATORS
        LDA     AC3,1,AC2       ;AC3 = NO. OF BYTES IN COMMAND BLOCK
        LDA     AC0,FOUR        ;AC0 = 4
        SGE     AC3,AC0         ;HAS AT LEAST 1 PARM BEEN PROVIDED?
        JMP     UIRET           ;NO - SO RETURN
        LDA     AC0,4,AC2       ;YES - AC0 = PARM 0
        STA     AC0,A           ;A = PARM 0
        LDA     AC0,SIX         ;AC0 = 6
        SGE     AC3,AC0         ;HAVE AT LEAST 2 PARMS BEEN PROVIDED?
        JMP     UIRET           ;NO - SO RETURN
        LDA     AC0,5,AC2       ;YES - AC0 = PARM 1
        STA     AC0,B           ;B = PARM 1
        LDA     AC0,EIGHT       ;AC0 = 8
        SGE     AC3,AC0         ;HAVE AT LEAST 3 PARMS BEEN PROVIDED?
        JMP     UIRET           ;NO - SO RETURN
        LDA     AC0,6,AC2       ;YES - AC0 = PARM 2
        STA     AC0,C           ;C = PARM 2
        LDA     AC0,TEN         ;AC0 = 10
        SGE     AC3,AC0         ;HAVE AT LEAST 4 PARMS BEEN PROVIDED?
        JMP     UIRET           ;NO - SO RETURN
        LDA     AC0,7,AC2       ;YES - AC0 = PARM 3
        STA     AC0,D           ;D = PARM 3
UIRET:  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.