INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
INTERNATIONAL CENTRE FOR SCIENCE AND HIGH TECHNOLOGY
c/o INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS 34100 TRIESTE (ITALY) VIA GRIGNANO, 9 (ADRIATICO PALACE) P.O. BOX 586 TELEPHONE 040-224572 TELEFAX 040-224575 TELEX 460449 APH I

SMR/643 - 15

SECOND COLLEGE ON
MICROPROCESSOR-BASED REAL-TIME CONTROL -
PRINCIPLES AND APPLICATIONS IN PHYSICS
5 - 30 October 1992

INTRODUCTION TO MULTIBOARD

A.J. WETHERILT
Scientific & Technical Research Council of Turkey
Marmara Scientific & Industrial Research Centre
P.O. Box 21
41470 Gebze Kocaeli
Turkey

The Multi Function Board : Extensions to ROSY

History : Developed for the 1st Real Time Workshop by Laura Carcione, Alberto Colavita, Pablo Eggarter,Cesar Rosales and Mario Trujillo.

System Hardware :
Memory map

Address	Function
0000 : 3FFF	16k RAM
4000 : 7FFF	16k RAM
8000 : 8001	Communications Port
9000 : 9001	ADC registers
A000 : A001	DAC registers
C000 : FFFF	16k ROM Kernel

Ports and devices :

- **ADC SI8601 :** 8 channel , 8 bit, 25 µsec conversion time,successive approximation, 0-5V input.

ADC Channel Select Reg $9000 W
ADC Data Register $9000 R

A7	A6	A5	A4	A3	A2	A1	A0

ADC Start Conversion $9001 R/W
(will start conversion if S7 is clear)

X	X	X	X	X	X	X	X

- **DAC AD558 :** 8 bit , 0.8 µsec settling time, 0-2.55V output

DAC1 Data Register $A000 W
DAC2 Data Register $A001 W

- **Communications Port :**

Data Exchange Register $8000 W (MULTI)

D7	D6	D5	D4	D3	D2	D1	D0

Status Register $8001 W (MULTI)

S7	X	X	X	X	S2	S1	S0

Meaning of bits

S0 : Data is placed into DER by MULTI. S0 is then set to indicate that data is available to ROSY. When transferring data from ROSY to MULTI S0 is set to indicate that MULTI is ready to accept data. This bit is automatically cleared when ROSY reads (or writes to) the DER.

S1: Set when MULTI is executing a command. Cleared when MULTI is ready to receive a new command or when an erroneous command is received.

S2: When set this bit indicates that MULTI will cause an interrupt on ROSY when next S0 is set.

S7: If clear ADC conversions initiated by read/write to ADC SCR. If set conversions initiated by a High on the Ext_Start pin of the output connector.

Data/Command Register $8000 R (MULTI)

D7	D6	D5	D4	D3	D2	D1	D0

Control Register $8001 R (MULTI)

X	X	X	X	X	C2	C1	C0

Meaning of bits C0: When set this bit indicates that the byte in the DCR is data (if ROSY is sending to MULTI) or that ROSY has read the byte in the DCR (MULTI to ROSY).

C1: Set if the byte in the DCR is a command . If both C0 and C1 set then DCR is treated as a command.

Both C0 and C1 are reset following a read by MULTI.

C2: Indicates that interrupts (IRQ) will be passed from MULTI to ROSY whenever S0 is set by MULTI.

ROSY Port Address : $EF20 / $EF30

Resident Commands : MULTI provides a number of commands resident in the kernel for use by an external user (ROSY). Generally these commands are accessed by placing an appropriate code in the DCR and performing the handshake sequence. Several commands require further bytes to specify extra information. Examples of these commands are :

Command	Code	# of post bytes	Function
RESET	$00	0	Performs a software reset of the system (useful for installing new commands). If successful S1 and S2 cleared with error code in DER
STOP	$0F	0	Halt any task currently being executed
RECEIVE	$01	4	Instructs MULTI to send data to ROSY. First 2 bytes give address in MULTI where data is stored . 3rd and 4th bytes give # of bytes to be transferred.
SEND	$02	4	Instructs MULTI to take a number of bytes of data and to store the block at the given address. First 2 extra bytes specify the loading address and last 2 bytes the # of bytes to be transferred.
ONE_ADC	%00101xxx	0	MULTI will perform a single ADC conversion through channel xxx and will place the result in the DER and setting S0. S1 will be set while the conversionis in progress.
SEV_ADC	%00110xxx	4	Instructs MULTI to perform a number of conversions through channel xxx and to store the resulting data in consecutive locations starting at the address given by the first 2 extra bytes.

RUN	%03	2	MULTI will execute code previously loaded into its memory. The two postbytes give the address of the first statement to be executed.

Additional commands are provided for configuring MULTI as either a Multichannal Analyser or Digital Oscilloscope. The relevant manual should be refered to for further details of these functions.

Possible error codes returned by the above functions are :

succesful_execution	$0
undefined_command	$03
stop_executed	$04
reset_executed	$05
to_many_bytes	$07
illegal_address	$10

Using the Multi Function board under OS9

In order to take advantage of the real time features of ROSY and OS9 a device driver has been prepared for MULTI. The device driver enables the user to treat MULTI as a SCF and hence the following functions are supported :

- init — initialise device
- read — read a single character
- write — write a single character
- getsta — get device status
- setsta — set device status
- term — terminate device

These functions can be used to build more complicated C routines enabling MULTI to provide hardware functions for real time systems.

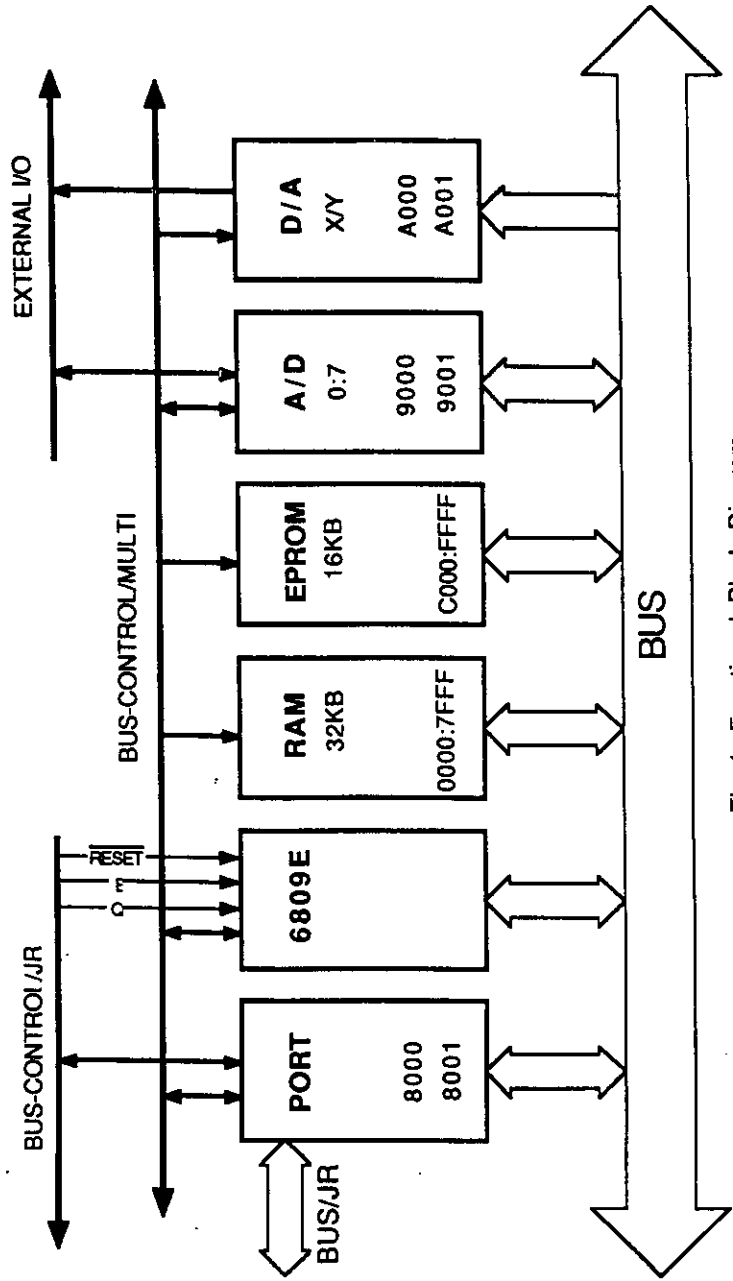
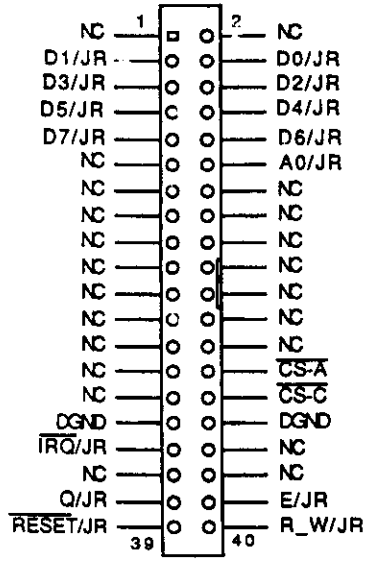


Fig 1. Functional Block Diagram

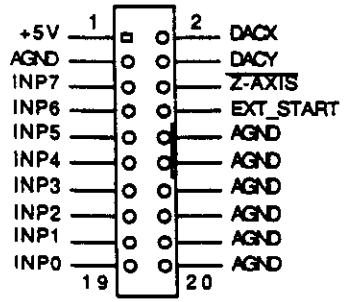
APPENDIX A

Connector Specifications
Top view

a)40-Pin ROSY-JR to MULTI connector



b)20-Pin MULTI output connector



(7)

```
*
* Device descriptor for Multifunction Board for Rosy Jr.
*

      nam dmult
      ifp1
      use defsfile
      endc


      ttl Device descriptor for Multi
      mod MLEND,MLNAM,DEVIC+OBJCT,REENT+1,MLMNGR,MLDRV
      fcb READ.+WRITE.
      fcb $FF
      fdb $EF30   Port address
      fcb 0       No options
MLNAM  fcs /dmult/
MLMNGR fcs /SCF/
MLDRV  fcs /mult/
      emod
MLEND  equ *
```

```
*
*
*
*    Device driver for Multifunction Board for Rosy Jr.
*
*
*

         nam mult
         ifp1
         use defsfile
         endc

         ttl Driver for Multi
         mod MLEND,MLNAM,DRIVR+OBJCT,REENT+1,MLENT,MLVAR
         fcb EXEC.+UPDAT.
MLNAM    fcs /mult/
         fcb 1                  Revision number

         org V.SCF
*
* Variable area
*
Micsr    rmb 2                  address of control/status register
Errd     rmb 1                  error or data retrieved by interrupt routine
Int      rmb 1                  flag for interrupt request aborted
Sem1     rmb 1                  semaphore to protect command execution
Sem2     rmb 1                  semaphore for Multi,only one function at a time
*                               can be exec.
User     rmb 1                  last process that sent a command to Multi
MLVAR    equ .
*
*
E$MNI    equ  99                Multi is not installed
E$NCM    equ 100                Not controlling Multi
E$NVO    equ 101                Not a valid operation
E$NCP    equ 102                No command being performed
E$WTD    equ 103
E$CAB    equ 104                Command was aborted
*
*
MLENT    lbra Init
         lbra Read
         lbra Write
         lbra Getstat
         lbra Setcon
         lbra Term              ... if it can be executed

MLMASK   fcb 0                  (flip none)
         fcb $04                IRQ Polling Mask ... there is probably an
*                               interrupt if bit 2 is set
         fcb 1                  Now,assign a low priority
*
*
```

```
* Subroutine Init: install interrupt service routine for Multi
*                  into polling table
*
*
* Passed: Y  address of the device descriptor module
*         U  address of the device static storage
*
Init     ldd V.Port,u
         addd #1
         std Micsr,u            store address of status register
         ldd #2                 check if Multi is installed sending a reset
         std [V.Port,u]
         ldx #100   Wait ...
         os9 F$Sleep
         lda [V.Port,u]         check exit error
         cmpa #5
         beq Cini
         ldb #E$MNI
         orcc #1
         bra Exini
Cini     lda #1                 initialize semaphores
         sta Sem1,u
         sta Sem2,u
         clr Int,u
         ldd Micsr,u
         leax MLMASK,pc pointer to mask for polling
         leay irqsvc,pc pointer to Irq service routine
         os9 F$Irq  install
Exini    rts


*
*
* Subroutine Read: receives a byte of data from Multi
*
* Passed : Y  address of path descriptor
*       U  address of device static storage
*       A  char to be read
*
Read     bsr Ckoper             check if it is a valid operation
         bcs Exre
         lda [V.Port,u]         read data exchange register
         ldb #$01               send acknowledgment
         stb [Micsr,u]
         inc Sem2,u
         clrb
Exre     rts


*
* Write routine: transfers a byte of data to Multi
```

```
*
* Passed: A char to write
*       Y address of path descriptor
*       U address of device static storage
*
*
Write   bsr Ckoper          check if it is a valid operation
        bcs Exwt
        ldb #$01
        std [V.Port,u]      write into data command register
        inc Sem2,u
        clrb
Exwt    rts


Ckoper  ldb User,u
        cmpb V.Busy,u       is it the process that sent the last command?
        bne Cko10           no ... error
        lsr Sem2,u          controlling Multi,yet?
        bcs Cko20           yes
Cko10   comb
        ldb #E$NCM          not controlling Multi
        bra Excko           exit with error
Cko20   ldb [Micsr,u]
        andb #%00000011     check acknowledgement mask and busy flag
        cmpb #%00000011
        beq Cko30           it is a valid operation
        comb
        ldb #E$NVO          not a valid operation,error
        bra Excko
Cko30   clrb
Excko   rts



*
*
* Subroutine Getstat: returns Multi's error into least
*                     significant part of register x
*                     if the most significant part is zero
*                     otherwise the most significant part
*                     contains the value of status register
*
*
* Passed : A function code
*       Y address of path descriptor
*       U address of device static storage
*
Getstat tst Sem1,u          is Sem1 locked?
        beq Cpr             yes ... continue
        comb
        ldb #E$NCP          no command is being performed now!
        bra Exge
Cpr     ldb User,u          is it the process that sent the command?
        cmpb PD.CPR,y       no ... error
        bne G0
        ldx PD.RGS,y
        lsr Sem2,u
        bcs G00
G0      comb
        ldb #E$NCM          not controlling Multi
        bra Exge
G00     ldb [Micsr,u]       read status register
        andb #03
        beq G02
        cmpb #1
        bne G01
        ldb [V.Port,u]      read error
        lda #0
        bra G03
G01     tfr b,a
        ldb #0
        std R$X,x
        bra G04
G02     ldb Errd,u          error was read by the interrupt routine
        lda #0
G03     std R$X,x
        inc Sem1,u          free semaphore
G04     clrb
        inc Sem2,u
Exge    rts

*
*
* Subroutine Setcon: sends command to Multi
*                    a byte is sent into user's x register
*                    followed by flags to be written into control reg
*                    returned 'error code' if interrupt was enabled
*
* Passed: A function code
*       Y address of path descriptor
*       U address of device static storage
*
Setcon  equ *
        ldx PD.RGS,y        Retrieve parameters
        ldd R$X,x
        bitb #%00000010     Is it a command?
        beq Isdat           no...
        bitb #%10000000     Will this command ignore semaphore 1
        beq Cksem           no ... check semaphore 1
        bsr Wsem2           wait semaphore 2
        clr Sem1,u          new owner of semaphore 1
        orcc #IntMasks
        tst V.Wake,u        Is there any process waiting for an interrupt?
```

```
        beq Ncint       no...
        clr V.Wake,u    tell it that control was lost...
        ldb #1
        stb Int,u
Ncint   andcc #^IntMasks
        bra Scom
Cksem   bsr Wsem1
Scom    pshs b
        ldb PD.CPR,y    store current process as new user of Multi
        stb User,u
        puls b
        bra Oper
Isdat   bitb #%00000001 is it data ?
        bne D10         yes
        comb
        ldb #E$WTD      exit with an error
        bra Exse
D10     pshs b
        ldb User,u      is it the process that sent the command?
        cmpb PD.CPR,y
        puls b
        bne D20         no ... error
        lsr Sem2,u
        bcs D30         continue
D20     comb
        ldb #E$NCM      exit with an error
        bra Exse
D30     bsr Ckakn       check acknowledgment mask
        bcs Exse        exit if error
Oper    bsr Wreg        write Multi's registers
        bcs Exse
        clrb
Exse    rts

Wsem1   lsr Sem1,u
        bcs Exs1
        ldx #3
        os9 F$Sleep
        bra Wsem1
Exs1    clr Sem2,u      now take Sem2
        rts

Wsem2   lsr Sem2,u
        bcs Exs2
        ldx #2
        os9 F$Sleep
        bra Wsem2
Exs2    rts

Ckakn   pshs a
        lda [Mlcsr,u]
        anda #%00000011
        cmpa #%00000011
        puls a
        beq Excka
        comb
        ldb #E$NVO
        inc Sem2,u
Excka   rts

Wreg    bitb #%00000100 interrupts requested?
        beq Wint        no ...
        pshs a          be prepared to receive the interrupt
        lda User,u
        sta V.Wake,u
        puls a
        std [V.Port,u]
        inc Sem2,u      free semaphore 2
Sleep   ldx #10
        os9 F$Sleep     wait for a signal
        tst V.Wake,u    has the process received the Irq interrupt?
        bne Sleep       no ... suspend process again
        tst Int,u       was the command aborted?
        beq Wr0         no ... exit
        clr Int,u       clear interrupt flag
        comb
        ldb #E$CAB      command was aborted
        bra Exwr
Wint    std [V.Port,u]
        inc Sem2,u
Wr0     clrb
Exwr    rts

*
*
*
* Subroutine Term: removes device from polling table
*
Term    ldd Mlcsr,u
        ldx #0
        os9 F$IRQ
        rts

*
*
* IRQ service routine : it reads error and wakes up
*               the sleeping process;
*
*
Irqsvc  lda [Mlcsr,u]
        anda #%00000001 was an interrupt issued by Multi?
        beq Err         no ... error
        lda [V.Port,u]  read byte from data register
        sta Errd,u
```

```
        lda V.Wake,u
        beq Exirq
        ldb #S$Wake
        clr V.Wake,u        clear this flag
        os9 F$Send          send signal to driver
Exirq   clrb
        rts
Err     comb
        rts


        emod
MLEND   equ *
```

```
/*
        Definitions for Multi library - Multi.h
*/

#define MULTI           "/dmult"
#define NULL            0
#define ERROR           -1
#define CONT            0
#define SET             0x00FF
#define RESET           0x0006
#define STOP            0x0f82
#define RECEIVE         0x0102
#define SEND            0x0202
#define RUN             0x0302
#define ADC0            0x2802
#define S_ADC0          0x3002
#define MULTI0          0x3802
#define DISP_F          0x1002
#define DISP_XY         0x1202
#define DISP_XY2        0x9302
#define V_CURSOR        0x9002
#define H_CURSOR        0x9102
#define Z_POS           0x0402
#define Z_NEG           0x0502
```

LIBRARY FUNCTIONS FOR MULTIFUNCTION BOARD

```
#include <os9.h>
#include <stdio.h>
#include <modes.h>
#include <errno.h>
#include "multi.h"


int pmult;

/*
    Initialize Multi to be used
    returns errno if it was impossible to access Multi
    0 if not
*/
int init()
{
    if ((pmult=open(MULTI,S_IREAD+S_IWRITE))==ERROR)
        return(errno);
    return(0);
}


/*
    Send a control to Multi
    receives: byte || control,
    returns ERROR if it was not possible to access Multi
    CONT if operation was succefully performed
*/
int control(code)
unsigned code;
{
    struct registers reg;
    reg.rg_a=pmult;
    reg.rg_b=200;
   /*
    code contains in its most significant part the code of the
    command and in its least significant part the control
   */
    reg.rg_x=code;
    if (_os9(I_SETSTT,&reg)==ERROR)
        return((int)reg.rg_b);
    return(CONT);
}


/*
    Read status register
    returns ERROR if access to Multi was not possible
        SET if Multi is busy
    Multi's error otherwise
*/
int rea_stat()
{
    struct registers reg;
    reg.rg_a=pmult;
    reg.rg_b=200;
    if (_os9(I_GETSTT,&reg)==ERROR)
        return((int)reg.rg_b);
    if (reg.rg_x&0xff00)
        return(SET);
    return(reg.rg_x);
}


/*
    Reset command using interrupts
    returns error returned by control
*/
int reset()
{
    int error;
    if ((error=control(RESET))!=CONT)
        return(error);
    return(rea_stat());
}


/*
    Stop command
*/
int stop()
{
    int err;
    if ((err=control(STOP))!=CONT)
        return(ERROR);
    while ((err=rea_stat())==SET) tsleep(2);
    return(err);
}


/*
    Send command; it receives as arguments
    a pointer to a char array
    the number of bytes to be transfered
    the address in Multi's memory
*/
int send(data,nbytes,address)
char *data;
unsigned nbytes,address;
{
    int err;
```

```
    /*
      Send command and check possible error
    */
      if ((err=control(SEND))!=CONT)
         return(err);
    /*
      Send arguments
    */
      if (write(pmult,&address,2)!=2)
         return(errno);
      if (write(pmult,&nbytes,2)!=2)
         return(errno);
    /*
      Check busy flag before transfering data
    */
      if ((err=rea_stat())!=SET)
         return(err);
      if (write(pmult,data,nbytes)!=nbytes)
         return(errno);
      return(rea_stat());
}


/*
    Receive command; it receives as arguments the pointer to a
    char array were data will be stored; the number of bytes to be
    transfered and the address from where data will be read
*/
int receive(data,nbytes,address)
char *data;
unsigned nbytes,address;
{
    int err;
    /*
      Send command and check possible error
    */
      if ((err=control(RECEIVE))!=CONT)
         return(err);
    /*
      Send arguments
    */
      if (write(pmult,&address,2)!=2)
         return(errno);
      if (write(pmult,&nbytes,2)!=2)
         return(errno);
    /*
      Check busy flag before reading data
    */
      if ((err=rea_stat())!=SET)
         return(err);
      if (read(pmult,data,nbytes)!=nbytes)
         return(errno);
    /*
      Read final error
    */
      return(rea_stat());
}


/*
    Run command without interrupts
*/
int run(address)
unsigned address;
{
    int error;
    if ((error=control(RUN))!=CONT)
       return(error);
    if (write(pmult,&address,2)==ERROR)
       return(errno);
}


/*
    Adc_several command
    address: Multi's buffer address,
    nconv: number of samples
*/
int adc_sev(address,nconv)
unsigned address,nconv;
{
    int error;
    if ((error=control(S_ADC0))!=CONT)
       return(error);
    if (write(pmult,&address,2)!=2)
       return(errno);
    if (write(pmult,&nconv,2)!=2)
       return(errno);
    while ((error=rea_stat())==SET) tsleep(2);
    return(error);
}


/*
    Multichannel command with interrupt
    It receives as arguments the total number of counts
    (simple way)
*/
multichan(count)
long count;
{
    char *p;
    int error;
    unsigned  code;
    if ((error=control(MULTI0))!=CONT)
       return(error);
```