Scheda di interfaccia fra il sistema multiprocessore vettoriale i860 e il dedispersore del sistema Pulsar di Medicina

A.Maccaferri, N.D'amico

Rapporto interno IRA 199/95

- Introduzione
- Descrizione funzionale
- Procedura d'uso
- Schema elettrico
- Realizzazione pratica
- Software di controllo
- Specifiche porta di espansione della scheda i860
- Bibliografia
-

Introduzione

Il sistema di acquisizione dati ed analisi di Pulsar del Radiotelescopio Croce del Nord é costituito da un complesso di apparecchiature come illustrato in fig.1. All'interno di questa architettura gioca un ruolo importante la "Survey Machine" (fig.2), composta da un dedispersore digitale (IRA 198/95) e da 4 schede a processore vettoriale i860 alloggiate all'interno di un host computer i386 con bus ISA. Il dedispersore si occupa di sommare i dati acquisiti in 128 sottobande compensando lo sfasamento in tempo relativo fra le varie sottobande provocato dall'attraversamento del mezzo interstellare del segnale emesso dalle sorgenti Pulsar. I dati "dedispersi" vengono inviati alle schede i860 che applicano l'algoritmo di *pulsar search*. Per ottimizzare le prestazioni del sistema si é reso necessario costruire un' opportuna interfaccia hardware inserita nel bus ISA che permettesse il trasferimento dei dati dedispersi alle 4 schede ed implementare un opportuno software per la gestione del dedispersore e dei 4 processi concorrenti.

Descrizione funzionale

La scheda di interfaccia inserita nel bus ISA dello stesso personal computer in cui sono inserite le 4 schede i860, denominata Mux riceve da un lato i dati dedispersi attraverso un cavo parallelo che costituisce un bus dati ad 8 bit piú il segnale di lettura effettuata (ADD_CK), dall'altro lato si connette direttamente al bus della CPU di ciascuna scheda i860 attraverso una specifica porta di espansione a 64 bit.
La CPU i386 del personal computer programmando un apposito registro sovrintende allo smistamento dei dati selezionando a quale scheda i860 inviarli.
Attraverso questa porta a 64 bit la CPU i860 puó effettuare una lettura ogni 800 nS circa.
Considerando che il dedispersore rende disponibile un dato ad 8 bit ogni 500 nS circa, si ottiene una sensibile riduzione del tempo necessario al trasferimento dati all'i860, impacchettandoli in una word a 16 bit ed implementando un apposito circuito che inserisca alcuni cicli di wait in modo da effettuarne la lettura (2 dati dedispersi ogni word) ogni microsecondo.
In Fig.3 é riportato un diagramma temporale dei principali segnali che intervengono nella sincronizzazione della lettura. Si noti che durante la fase di inizializzazione (precedente a T0), il segnale INIT generato via software, memorizza nel lacth del byte meno significativo il dato di indice 0 della serie di dati dedispersi (segnale ADD_CK) e postincrementa il dedispersore (segnale CLKD). Quando inizia la lettura (al primo FIFOCS ↑) sul bus dati sono presenti il dato 0 memorizzato nel byte meno significativo ed il dato 1 nel byte piú significativo. Effettuata la prima lettura si innesca il processo di temporizzazioni che proseguirá per tutta la lettura. A seguito di ogni lettura (FIFOCS ↑) viene abbassata la linea FIFOHLD, che mantiene in wait l'i860 alla prossima lettura, e vengono generati i segnali ADD_CK-CLKD, che postincrementano il dedispersore e memorizzano nel byte meno significativo il dato proveniente dal dedispersore. Partono inoltre 2 contatori, uno determina dopo quanto tempo memorizzare nel latch il dato successivo per renderlo disponibile alla prossima lettura, e rifare il postincremento; l'altro determina la durata del segnale FIFOHLD per il tempo necessario affinché anche il secondo dato sia disponibile per la lettura.
Il timing e la descrizione dei segnali dell'interfaccia sono riportati nelle specifiche della scheda i860 allegate.

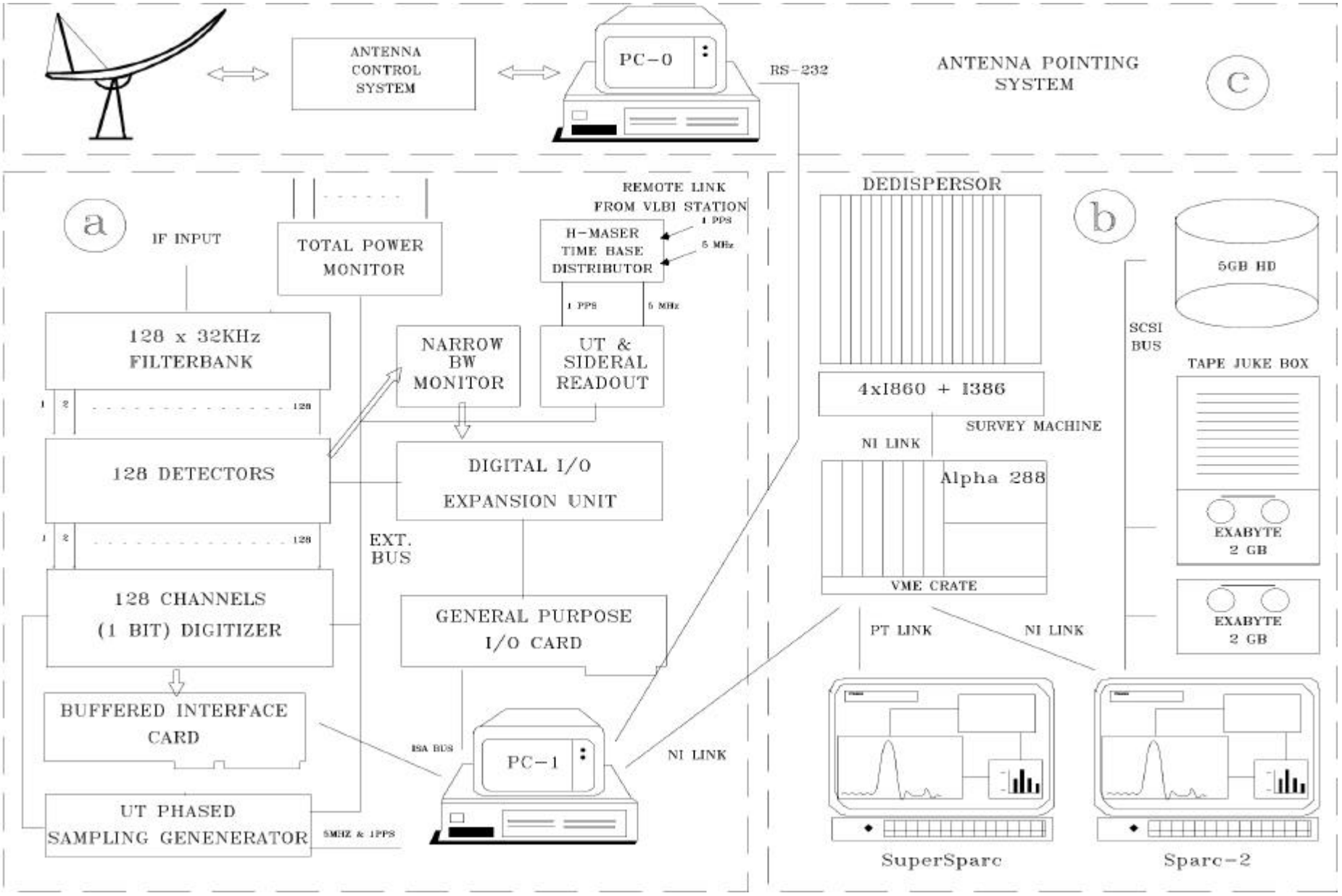


Fig 1 Sistema Pulsar di Medicina. a) Sottosistema di acquisizione dati b) Sottosistema di analisi e memorizzacione c) Sistema di puntamento

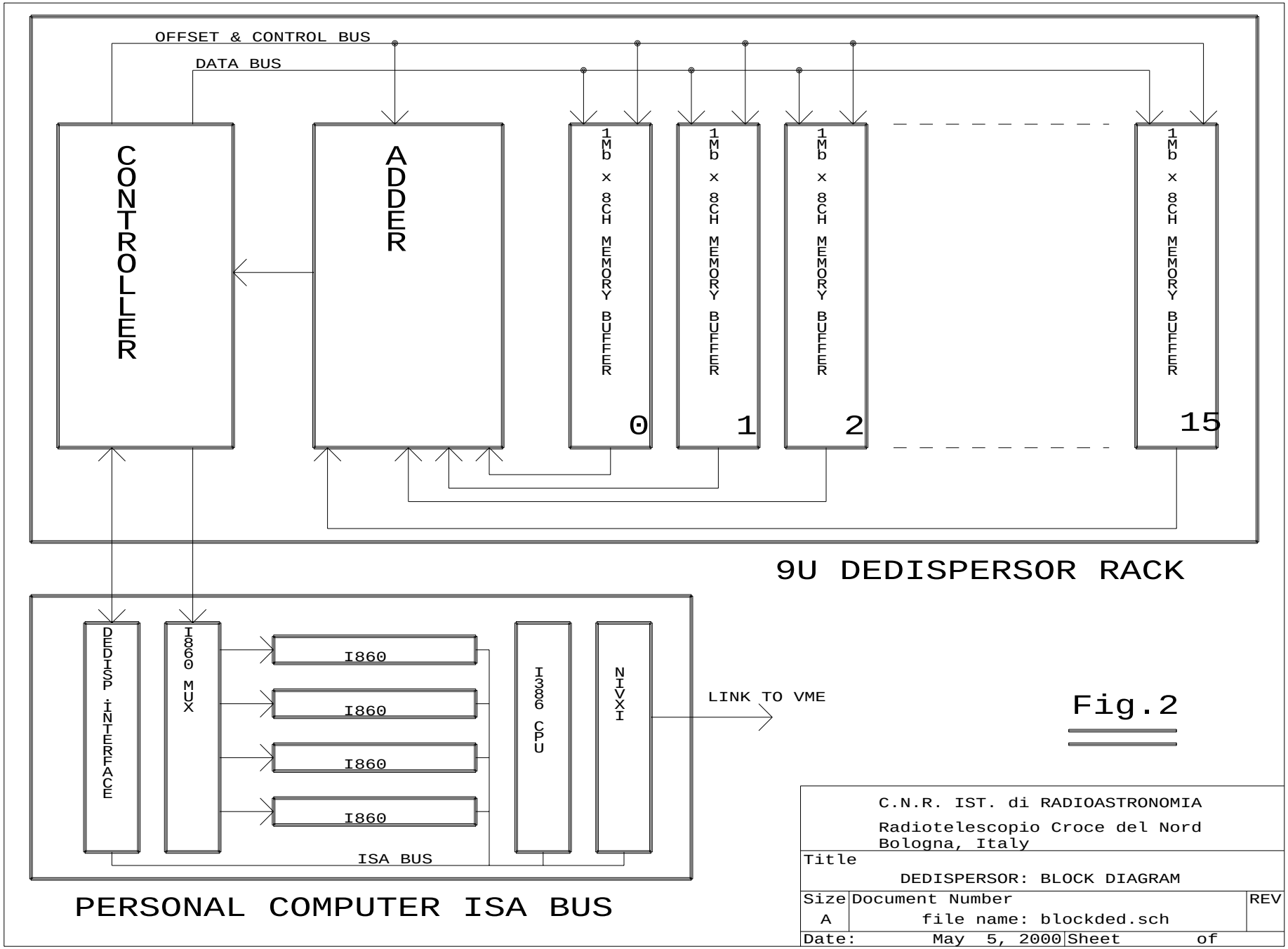


Fig.2

C.N.R. IST. di RADIOASTRONOMIA Radiotelescopio Croce del Nord Bologna, Italy		
Title: DEDISPERSOR: BLOCK DIAGRAM		
Size: A	Document Number: file name: blockded.sch	REV
Date: May 5, 2000	Sheet of	

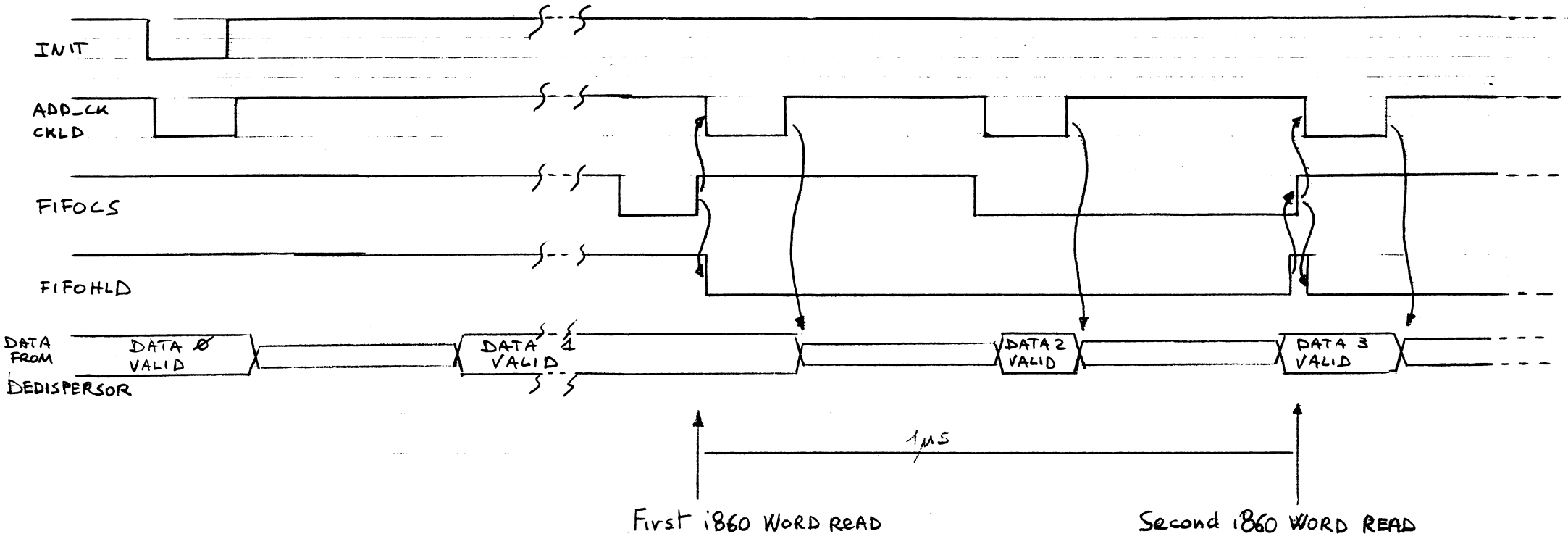


Fig.3 Diagramma temporale dei segnali piú significativi e loro interrelazione


DIGITAL
DEDISPERSOR
4 X I860 VECTOR
PROCESSOR
& Dedispersor
controller
VME CRATE

Procedura d'uso

Le operazioni da effettuare sono:

1) Selezionare la porta (i860) a cui inviare i dati scrivendone l'indirizzo (da 0 a 3) nei 3 bit meno significati del registro di controllo.

2) Configurare il dedispersore per il valore di dedispersione desiderato.

3) Effettuare un impulso 1,0,1, sul segnale INIT, in modo da memorizzare sul latch il valore del 1° dato disponibile, quindi produrre l'impulso ADD_CK di postincremento al dedispersore.

4) Effettuare le letture desiderate attraverso la scheda i860 abilitata al punto 1, tenendo presente che ad ogni lettura si acquisisce una parola composta da 2 dati ad 8 bit dedispersi, di cui il byte meno significativo é il primo dato dedisperso. (ad esempio se leggo 5 parole a 16 bit, avró i dati in questo ordine $t_1.t_0$, $t_3.t_2$, $t_5.t_4$, $t_7.t_6$, $t_9.t_8$).

Schema elettrico

Il circuito implementato é composto da 3 sezioni principali:

- Interfacciamento verso il bus ISA
- Generatore dei segnali di sincronizzazione
- Multiplexer e buffer

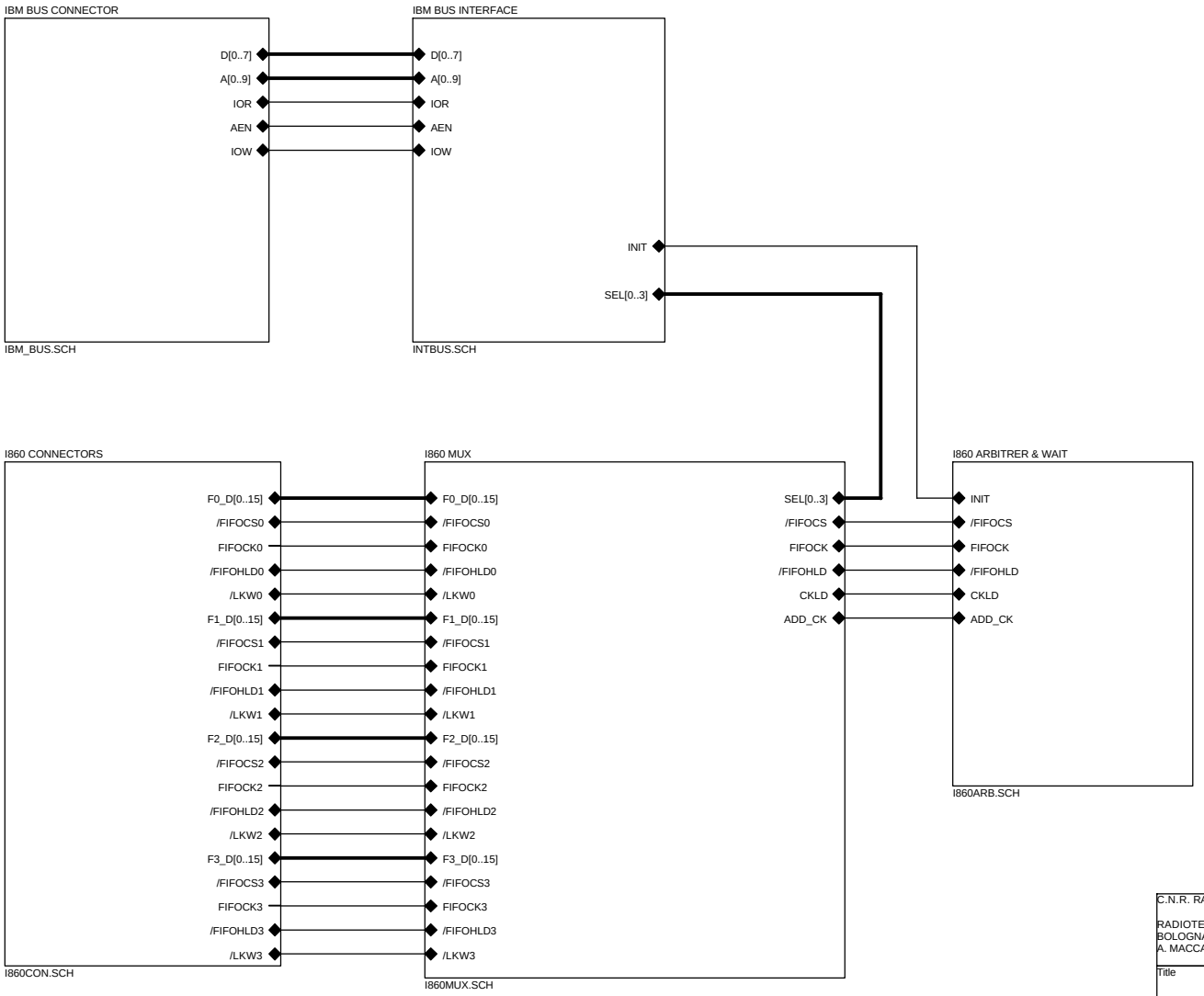
Il circuito di interfacciamento verso il bus ISA é di tipo classico, ed utilizza una sola porta di I/O ad 8 bit. Tutte le linee di indirizzamento sono decodificate, si ha quindi la massima libertá di configurazione dell'indirizzo di I/O, cosí da evitare eventuali conflitti con altre schede.
Il registro U18 memorizza i dati di configurazione. I 3 bit di minor peso, tramite la decodifica U19, attivano la porta i860 alla quale inviare i dati mentre il bit 7 viene utilizzato per generare il clock supplementare della prima lettura.

Il generatore dei segnali di sincronizzazione é implementato utilizzando come clock di sistema il clock a 40MHz della scheda i860 attiva, questo ha permesso di evitare i problemi di metastabilitá che altrimenti possono sorgere in applicazioni di questo tipo. Un primo blocco formato principalmente da U26A, U22 ed U23, genera l'impulso di clock supplementare per la lettura e la memorizzazione del dato intermedio fra una lettura e l'altra della word a 16 bit dell'i860.
U26B ed U24 determinano la larghezza dell'impulso ADD_CK che postincrementa il dedispersore a lettura avvenuta a partire dai segnali FIFOCS e dall'impulso supplementare generato dal blocco precedente.
U27A, U31 e U25 generano il segnale FIFOHLD che richiede all'i860 l'inserimento dei necessari cicli di wait in attesa che i due dati impachettati in una word a 16 bit siano disponibili.
Il multiplexer si compone di un blocco (U1 ed U2) che permette di selezionare i segnali di sincronismo FIFOCS, FIFOCLK e FIFOHLD della scheda attiva.
Si possono poi individuare 4 blocchi di buffer, uno per ciascuna scheda i860, che implementano anche la funzione di memorizzazione del dato intermedio, nel byte meno significativo, per l'impacchettamento a 16 bit.
Il buffer U11 rigenera i segnali provenienti dal dedispersore, fornendo un adeguato *fanout* per la circuiteria successiva ed una piú sicura ricezione dei dati grazie agli ingressi di tipo *Smith trigger*. Essendo il percorso fra il dedispersore e l'interfaccia relativamente breve (1 metro), utlizzando un cavo piatto con alternati un segnale ed una massa, é stato possibile garantire una corretta trasmissione dei dati.

Realizzazione pratica

Pur essendo un esemplare unico, per la costruzione della scheda non si é potuto ricorrere alla tecnologia wire-wrap, ma si é dovuto progettare un circuito stampato, perchè essendo i connettori di collegamento con le schede i860 di tipo a passo ridotto, non é possibile montarli in una normale scheda millefori; inoltre in questo modo si occupa un solo slot del bus ISA, permettendo quindi una migliore disposizione delle schede i860 che soffrono facilmente di malfunzionamenti dovuti alla non ottimale ventilazione delle CPU i860. A questo riguardo abbiamo dovuto installare sul coperchio del cabinet industriale in cui il tutto é inserito 4 ventole supplementari.

4x1860-DEDISPERSOR LINK
A.M. '93
CNR Radioastronomia Bo
PORT 3
PORT 0
CONNECTO
P1
WAIT
DELAY
RR4
RR3
RR2
RR1
4K7
74LS244
74LS374
74LS04
74LS00
74F191
74LS85
74LS138
74LS374
74F74
74LS08
74LS04
74LS245
14DIP300
16DIP300
20DIP300
4.7UF
J7
J1
100SIL



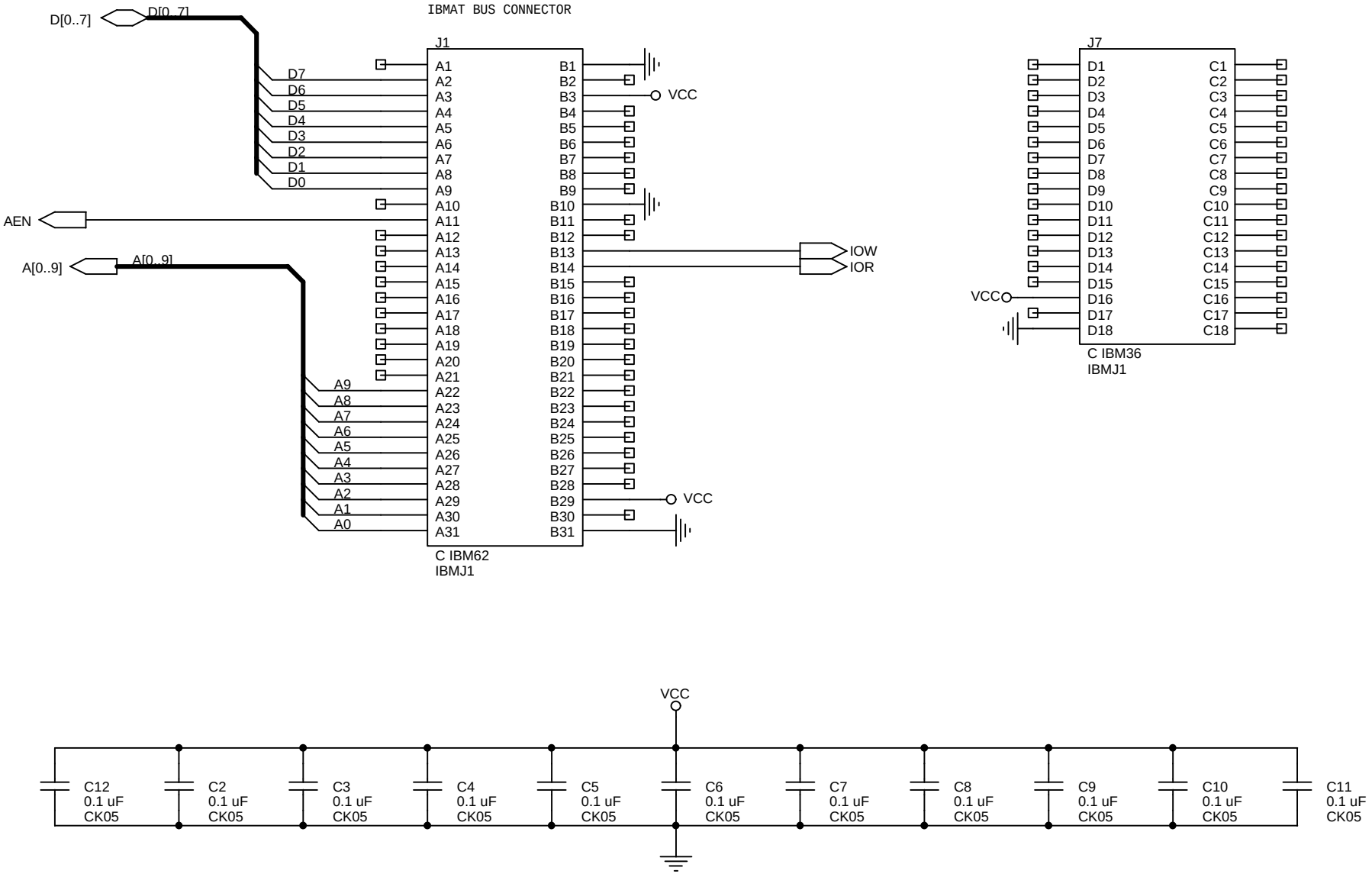
C.N.R. RADIOASTRONOMIA

RADIOTELESCOPIO CROCE DEL NORD
BOLOGNA - ITALY
A. MACCAFERRI

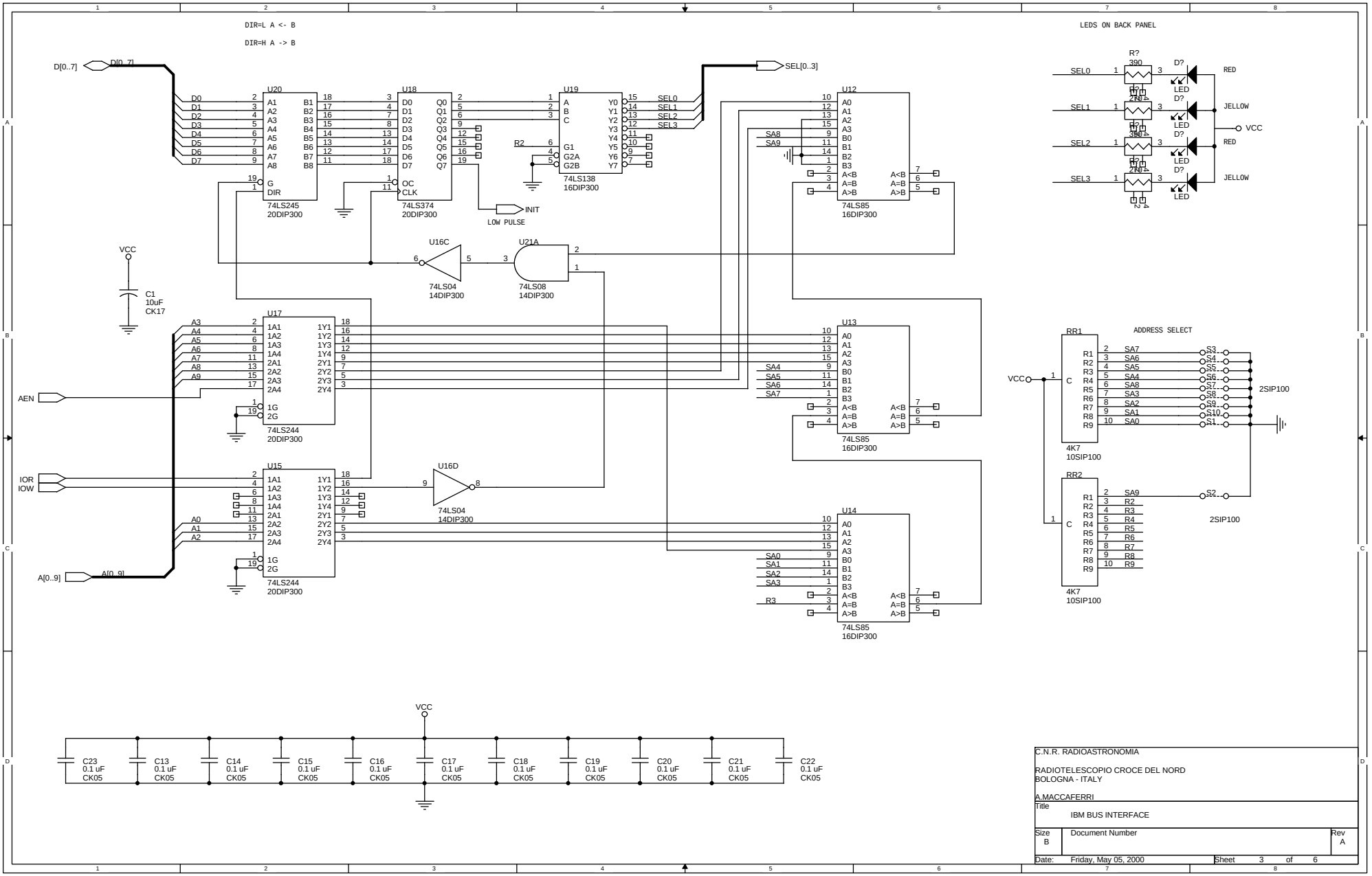
Title: 4xI860 MUX - DEDISPERSOR INTERFACE

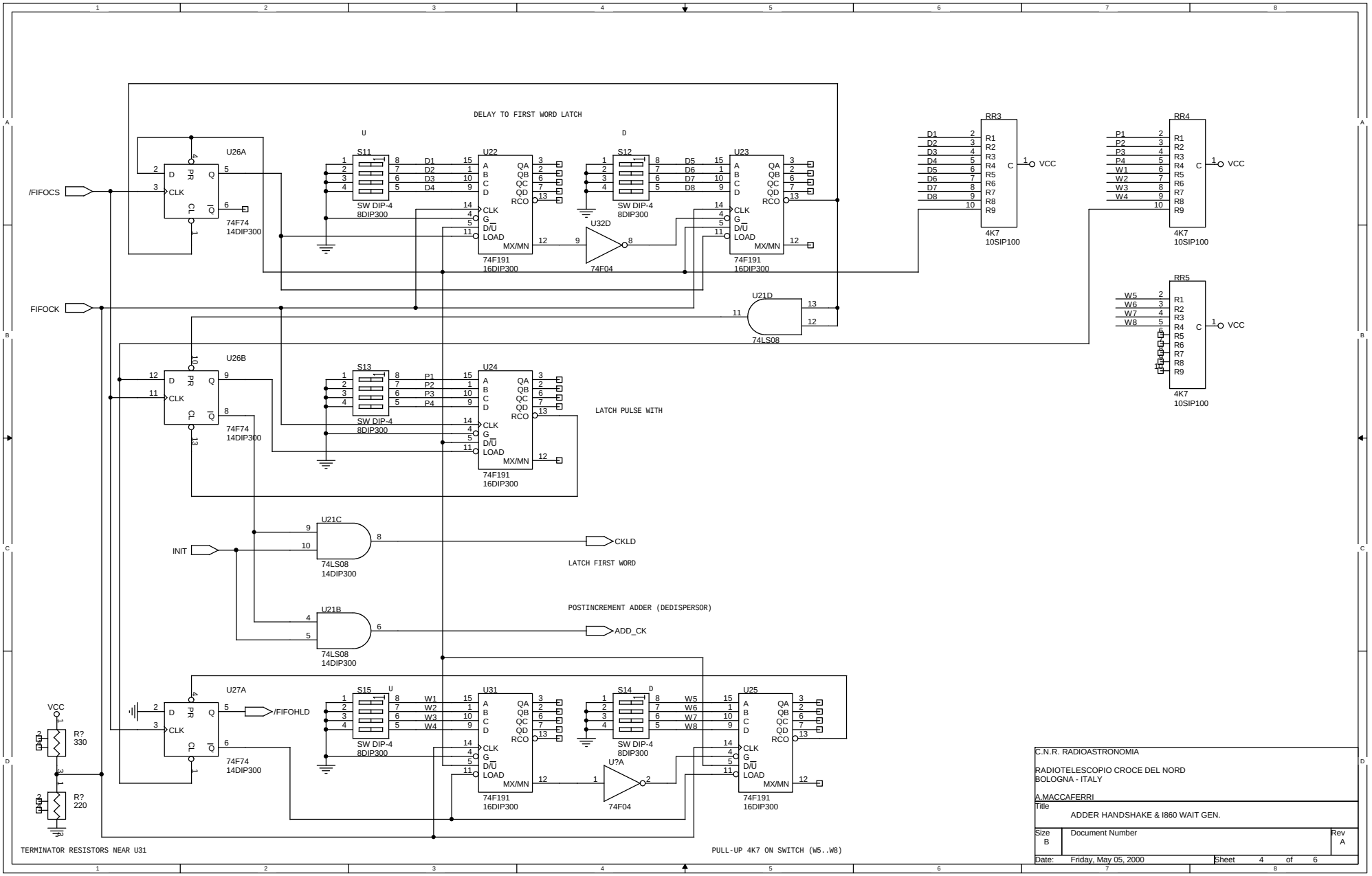
Size	Document Number	Rev
B		A

Date: Friday, May 05, 2000 Sheet 1 of 6



Size	Document Number	Rev
A		
Date: Friday, May 05, 2000	Sheet 2 of 6	


DIR=L A <- B
DIR=H A -> B
LEDS ON BACK PANEL
D[0..7]
SEL[0..3]
U20
74LS245
20DIP300
U18
74LS374
20DIP300
U19
74LS138
16DIP300
INIT
LOW PULSE
U12
74LS85
16DIP300
U16C
74LS04
14DIP300
U21A
74LS08
14DIP300
VCC
C1
10uF
CK17
U17
74LS244
20DIP300
U13
74LS85
16DIP300
AEN
IOR
IOW
A[0..9]
U15
74LS244
20DIP300
U16D
74LS04
14DIP300
U14
74LS85
16DIP300
R?
390
D?
LED
RED
JELLOW
SEL0
SEL1
SEL2
SEL3
RR1
4K7
10SIP100
ADDRESS SELECT
RR2
2SIP100
C23 0.1 uF CK05
C13 0.1 uF CK05
C14 0.1 uF CK05
C15 0.1 uF CK05
C16 0.1 uF CK05
C17 0.1 uF CK05
C18 0.1 uF CK05
C19 0.1 uF CK05
C20 0.1 uF CK05
C21 0.1 uF CK05
C22 0.1 uF CK05
C.N.R. RADIOASTRONOMIA
RADIOTELESCOPIO CROCE DEL NORD
BOLOGNA - ITALY
A.MACCAFERRI
Title
IBM BUS INTERFACE
Size B
Document Number
Rev A
Date: Friday, May 05, 2000
Sheet 3 of 6



C.N.R. RADIOASTRONOMIA

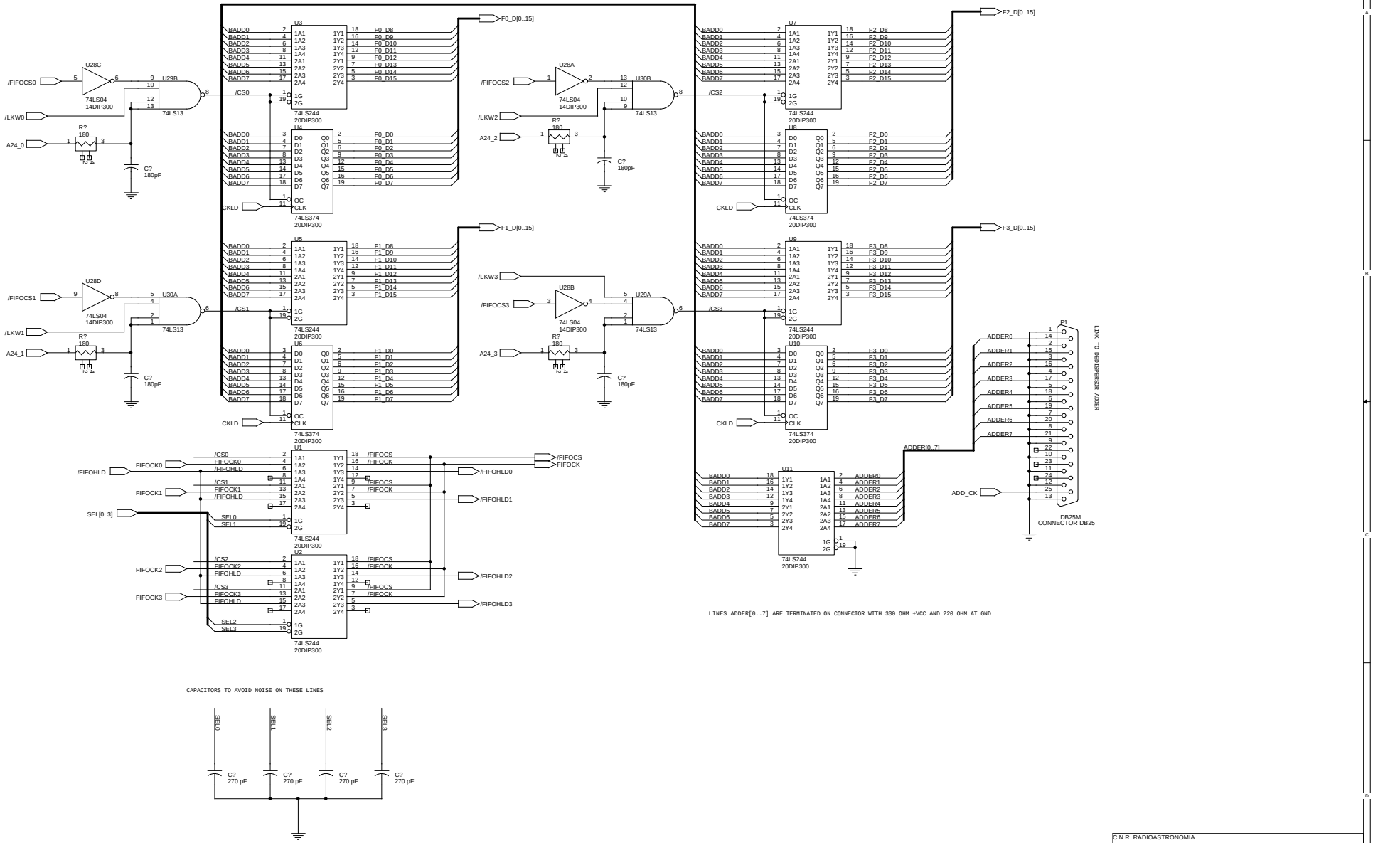
RADIOTELESCOPIO CROCE DEL NORD
BOLOGNA - ITALY

A.MACCAFERRI

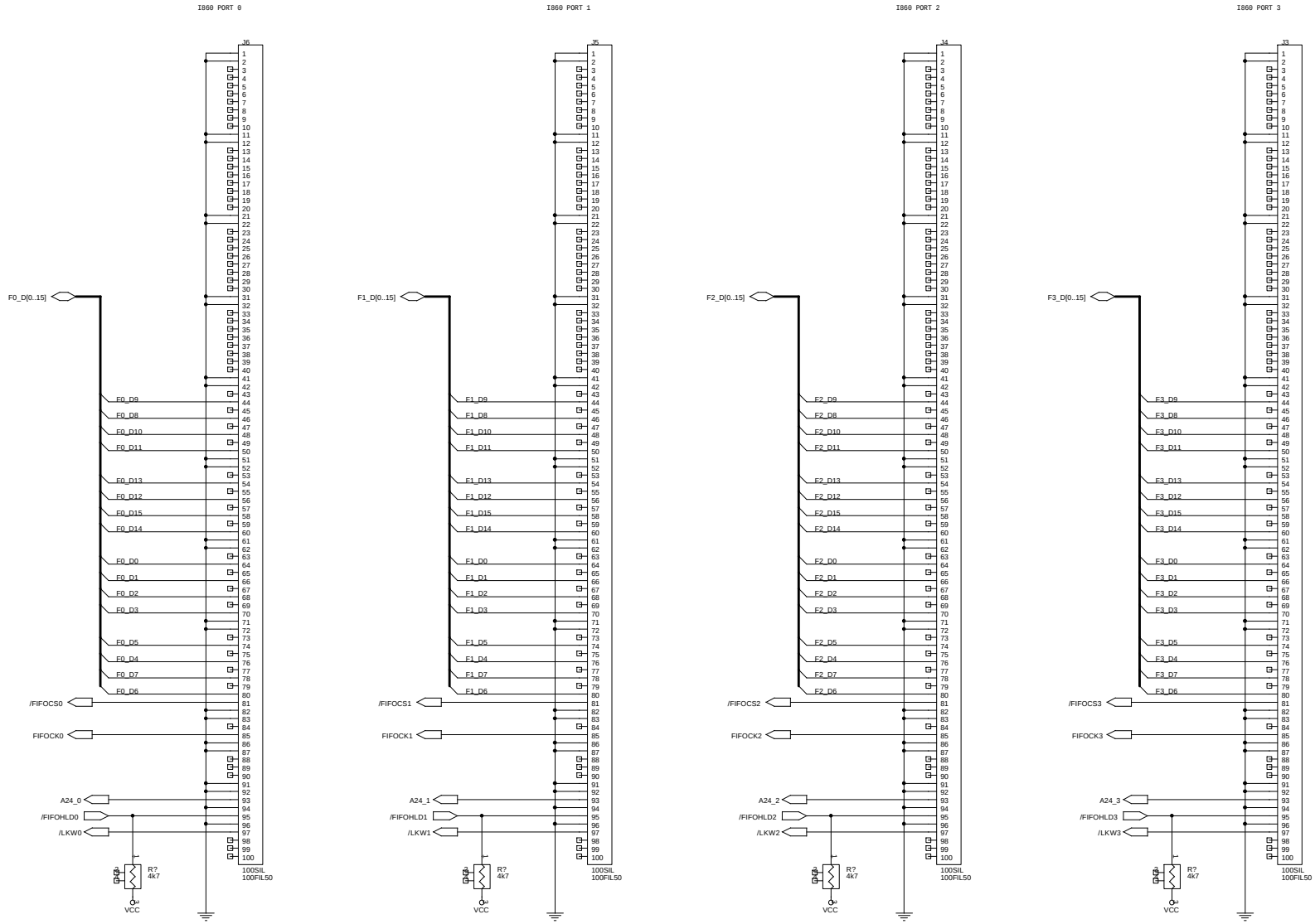
Title: ADDER HANDSHAKE & I860 WAIT GEN.

Size	Document Number	Rev
B		A

Date: Friday, May 05, 2000 Sheet 4 of 6



C.N.R. RADIOASTRONOMIA	
RADIOTELESCOPIO CROCE DEL NORD BOLOGNA - ITALY	
A.MACCAFERRI	
Title	I860 MUX
Size C	Document Number
Rev A	
Date: Friday, May 05, 2000	Sheet 5 of 6



C.N.R. RADIOASTRONOMIA RADIOTELESCOPIO CROCE DEL NORD BOLOGNA - ITALY		
Title: LINKS TO 4 I860 FIFO PORT		
Size: C	Document Number	Rev: A
Date: Friday, May 05, 2000	Sheet 6 of 6	

Software di controllo

La scheda di interfaccia é chiamata a distribuire i dati in uscita dal dedispersore alle 4 schede i860, é stato quindi predisposto un opportuno insieme di routine software, che attraverso il colloquio fra ciascuna delle 4 schede i860 e la CPU i386 host, gestisse le varie fasi del processo:

- Trasferimento di un nuovo set di dati al dedispersore, quando tutte le varie dedispersioni sono state analizzate.
- Rendere disponibile in successione il dedispersore ad ognuna delle 4 schede.
- Configurare il dedispersore per un dato fattore di dedispersione, su richiesta dell'i860 che attende i dati.
- Abilitare la corrispondente porta della scheda di interfaccia.
- lettura dei dati da parte dell'i860

A titolo di esempio sono stati inclusi in questa nota tecnica i listati di alcune routine utilizzate per l'analisi dati *PULSAR*. Le parti di interesse sono state evidenziate in grassetto.
Nel file SERVICE.C (**i386**) viene creata la tabella dei vettori di interrupt utilizzata per indirizzare le richieste d'IRQ degli i860.
Nel file IRQHN5.FOR risiede la parte di software eseguita dalla **CPU i386** che colloquia con le schede i860 rispondendo alle sollecitazioni che queste effettuano tramite le richieste di interrupt hardware (IRQ). Ogni scheda ha un prorio numero di IRQ che la identifica. Tramite questo IRQ vengono effettuate diversi tipi di richieste, distinte da un byte di controllo.

i860_IRQ_INIT	Inizializza il gestore di IRQ
I860_IRQ3	Identifica la scheda richiedente. Verifica la disponibilitá del dedispersore, se libero procede, altrimenti mette in attesa l'i860. Riceve dall'i860 i dati analizzati della dedispersione precedente. Alloca e configura il dedispersore, inizializza la scheda di interfaccia e ne seleziona la porta connessa alla scheda richiedente.
I860_IRQ13	Identifica la scheda richiedente. Dealloca il dedispersore

Nel file DEDISP.FOR (**CPU i386**) é presente oltre alla parte di configurazione del dedispersore, l'inizializzazione e la configurazione della scheda di interfaccia (MUX i860).`

Nel file NCPAR.F é elencata la parte di codice eseguita dalle **CPU i860** che sovrintende al colloquio con l'host i386.
Le subroutine di interesse sono:

NC_DATA	Effettua la richiesta di IRQ tramite la routine SIGNAL_HOST. Se la richiesta e' servita invia al i386 i risultati della dedispersione precedente e legge i nuovi dati dedispersi, altrimenti attende un tempo prestabilito, poi rieffettua la richiesta.

FILE NAME: SERVICE.C

```
#include "run860.h" /* to compile: cl /c /G2 /AL service.c */
#include "stdio.h"
extern void _far i860_irq3(char i);

extern void _far i860_irq13(char i);

int service()
{
     _860_services[3] = i860_irq3;
     _860_services[4] = i860_irq3;
     _860_services[5] = i860_irq3;
     _860_services[6] = i860_irq3;

     _860_services[13] = i860_irq13;
     _860_services[14] = i860_irq13;
     _860_services[15] = i860_irq13;
     _860_services[16] = i860_irq13;

     _enableints = 1;
     return 1;
}
```

FILE NAME: IRQHN5.FOR

```
$DEBUG
*================================================================
*  Interrupt handlers for i860 Microway board and VME devices.

*  i860 Interrupt id code IRQ$NR:

*  Interrupt id code 3,4,5 & 6 allocate the desipersor
*  interface to the i860 process requesting service. In this
case
*  the interrupt id identifes uniquely the i860 process.

*  Interrupt request 13,14,15 & 16 deallocate the dedispersor
*  interface to the corresponding i860 process. In this case
*  the interrupt id-10 identifies the i860 process.


*  WARNING: This is a FORTRAN code running in a language "C"
*  environment, so strings and disk I/O follows language "C"
*  rules.

*  History
*================================================================
*  v1.00 N. D'Amico, Nov 1992 - Dedispersed data read passed
*                               by the host.
```

```
*  v1.10 N. D'AMico, Jan 1993 - VME interrupt handler included.

*  v2.00 N. D'AMico, Feb 1993 - Dedispersed data read by i860
*                               using a the FIFO map.
*  v3.00 N. D'Amico, Mar 1995 -
*================================================================

     INTERFACE TO SUBROUTINE FREEVME
     END

     INTERFACE TO SUBROUTINE NEWBEAM(nbeam,bname)
     integer*4 nbeam[reference]
     character*(*) bname
     END

     INTERFACE TO SUBROUTINE MOVETOVME(irqnr,buf[reference])
     integer*1 irqnr[reference]
     integer*1 buf(*)
     END

     INTERFACE TO INTEGER*2 FUNCTION SEND860
     &  [FAR,C,alias:'_send860'] (word[reference],wordlen)
     integer*1 word(*)
     integer[C] wordlen[value]
     END

     INTERFACE TO INTEGER*2 FUNCTION GET860
     &  [FAR,C,alias:'_get860'] (word[reference],wordlen)
     integer*1 word(*)
     integer[C] wordlen[value]
     END
*================================================================
     Subroutine I860_IRQ_INIT

     implicit none

     include 'vmemem.inc'
     integer*4  dmx(4)

     logical    master(4)           ! Dedispersor allocation
flags
     logical    justend(4)          ! i860 just ended

     logical    dm0done             ! DM 0 step done
     logical    dm0alloc            ! DM 0 step allocated for
processing
     logical    VME                 ! Servicing VME interrupt

     integer*4  current_process_nr
     integer*4  current_dm_nr
```

```
      integer*4  nbeam
      integer*4  priority(4),times(4)
      common/pr/priority,times,dmx,current_process_nr,
      &            current_dm_nr,nbeam,justend,master,
      &            dm0alloc,dm0done,VME

      logical    pending
      integer*2  controller
      integer*2  level
      integer*4  statusid
      integer*4  j

      character time0*12

      integer*1 hinc(640)
      character*16 b_name
      common/beam/b_name,hinc

      common/time/time0

      nbeam=0
      current_process_nr=1
      current_dm_nr=0

      time0=' '
      do j=1,4
       priority(j)=0
       master(j)=.false.
       justend(j)=.false.
       dmx(j)=0
      enddo
      priority(1)=3
      dm0done=.false.
      dm0alloc=.false.

      return
      end
*=============================================================
      Subroutine I860_IRQ3 [C] (IRQ$NR)

      implicit none

      include 'c:\i860\param\param.inc'
      include 'vmemem.inc'

      integer*1  IRQ$NR               ! Interrupt req #
      integer*2  send860              ! i860 interface routine
      integer*2  get860
      integer*2  dummy                ! i860 return value
      integer[C] length               ! nr of bytes to send
```

```
     integer*1  dmbyteveto(20)
     integer*1  dmbyte(20)           ! array for DM value
transfer
     integer*4  dm                   ! current DM value for
transfer
     integer*4  dmveto
     integer*4  dmx(4)

     integer*4  dstatus              ! Return satus from
Dedispersor
     logical    justend(4)
     logical    master(4)

     logical    dm0alloc             ! DM 0 step allocated for
processing
     logical    dm0done              ! DM 0 step done
     logical    VME                  ! Servicing VME interrupt

     integer*4  current_process_nr
     integer*4  current_dm_nr
     integer*4  ntd
     integer*4  ddm
     real*4     dval(4,64)
     integer*4  sh(128),shh(128)
     integer*2  bandmask

     integer*4  nbeam

     character*16 bveto,bname

     integer*1  b64k(12288)
     character*8 check
     equivalence (check,b64k(1))
     integer*4  mm,priority(4),times(4)
     common/pr/priority,times,dmx,current_process_nr,
     &            current_dm_nr,nbeam,justend,master,
     &            dm0alloc,dm0done,VME

     common/dedispersor/ddm,dval,sh,bandmask,shh

     integer*1 hinc(640)
     character*16 b_name
     common/beam/b_name,hinc

     equivalence(dmbyteveto(1),dmveto)
     equivalence(dmbyte(1),dm)

     logical sendflag
     character*16 dummyf
     equivalence(dummyf,b64k(1))

     equivalence(dmbyteveto(5),bveto)
```

```
equivalence(dmbyte(5),ntd)

logical    pending
integer*2  controller
integer*2  level
integer*4  statusid

integer*4  j,m
character*22 msgg(4)
character*20 mess,mess2,errmess*30,messkip*25
character char*12,strtime
character time0*12,stime*12
common/time/time0
data errmess/' PRIORITY ALLOCATION FAILURE! 'C/
data mess/' IRQ Received ! 'C/
data mess2/' I am here 'C/
data msgg/' Received data plot 1 'C,
&            ' Received data plot 2 'C,
&            ' Received data plot 3 'C,
&            ' Received data plot 4 'C/
data messkip/' Skipping dummy block ...'C/
data bveto/'='/
call IRQ_DI

do j=1,4
 if(master(j))then
  dmveto=-1
  length=4
  do m=1,4
   if(priority(m).eq.IRQ$NR)then
    dummy=send860(dmbyteveto,20)
    call IRQ_EN
    return
   endif
  enddo
  do m=1,4
   if(priority(m).eq.0)then
    priority(m)=IRQ$NR
    dummy=send860(dmbyteveto,20)
    call IRQ_EN
    return
   endif
  enddo
  write(6,10)errmess
  dummy=send860(dmbyteveto,20)
  call IRQ_EN
  return
 endif
enddo

if(priority(1).eq.0)priority(1)=IRQ$NR
```

```
     if(priority(1).ne.IRQ$NR)then
      dmveto=-1
      length=4
      dummy=send860(dmbyteveto,20)
      call IRQ_EN
      return
     endif

     master(IRQ$NR-2)=.true.

     if(justend(IRQ$NR-2))then
       length=4
       dmveto=-1
       master(IRQ$NR-2)=.false.
       priority(1)=priority(2)
       priority(2)=priority(3)
       priority(3)=priority(4)
       priority(4)=0
       dummy=send860(dmbyteveto,20)
       call IRQ_EN
       return
      endif
      dmx(IRQ$NR-2)=dmx(IRQ$NR-2)+1
      dm=dmx(IRQ$NR-2)
      ntd=ndmtot
      call DEDISPERSOR_ALLOCATE(IRQ$NR,dstatus,dm)
      if(dstatus.eq.0)then
       length=4
       dummy=send860(dmbyte,20)
       call sendinfo

* Send header to calling i860
       dummy=send860(hinc,640)

* Get the (previous) summary block
       dummy=get860(b64k,12288)
       if(dummyf(1:10).eq.'DUMMY_PLOT')then
          write(6,'(a)')messkip
       else
          call movetovme(IRQ$NR,b64k)
       endif
       call bandtovme(IRQ$NR,b64k)
       call IRQ_EN
       return
      else
       write(6,1)
 1        format(' Unexpected status from Dedispersor ')
       pause' Hit CR to stop'
       call IRQ_EN
       stop
      endif
```

```
      write(6,6)
 6       format(' Unexpected counter value ')
      pause' Hit CR to stop'
      call IRQ_EN
      stop

      call IRQ_EN
      return
 10     format(a)
      end

*=============================================================
      Subroutine I860_IRQ13 [C] (IRQ$NR)

      implicit none

      include 'vmemem.inc'
      include 'c:\i860\param\param.inc'

      integer*1  IRQ$NR               ! Interrupt req #
      integer*2  send860              ! i860 interface routine
      integer*2  dummy                ! i860 return value
      integer[C] length               ! nr of bytes to send

      integer*1  dmbyteveto(4)
      integer*1  dmbyte(4)            ! array for DM value
transfer
      integer*4  dm                   ! current DM value for
transfer
      integer*4  dmveto
      integer*4  dmx(4)

      integer*4  dstatus              ! Return satus from
Dedispersor
      logical    justend(4)
      logical    master(4)
      logical    end

      logical    dm0alloc             ! DM 0 step allocated for
processing
      logical    dm0done              ! DM 0 step done
      logical    VME                  ! Servicing VME interrupt

      integer*4  current_process_nr
      integer*4  current_dm_nr

      character*30 message
      character*30 mess2

      integer*4  ddm
      real*4     dval(4,64)
```

```
      integer*4  sh(128),shh(128)
      integer*2  bandmask
      common/dedispersor/ddm,dval,sh,bandmask,shh
      integer*4  nbeam
      equivalence(dmbyteveto(1),dmveto)
      equivalence(dmbyte(1),dm)

      logical    pending
      integer*2  controller
      integer*2  level
      integer*4  statusid

      integer*4  j,m

      character*16 b_name
      integer*1 hinc(640)
      common/beam/b_name,hinc

      integer*4  priority(4),times(4)
      common/pr/priority,times,dmx,current_process_nr,
     &            current_dm_nr,nbeam,justend,master,
     &            dm0alloc,dm0done,VME

      data mess2/' i860_IRQ:: request ok  'C/
      data message/' i860_IRQ:: bad request 'C/

      call IRQ_DI
      if(.not.master(IRQ$NR-10-2))then
       write(6,10)message
       call IRQ_EN
       stop
      endif

*  Deallocate Dedispersor Interface

      call DEDISPERSOR_DEALLOCATE(IRQ$NR,dstatus,dm)

      if(dstatus.eq.0)then
       if(priority(1).ne.IRQ$NR-10)then
        write(6,2)
 2        format(' Unexpected Priority setting ! ')
       else
        priority(1)=priority(2)
        priority(2)=priority(3)
        priority(3)=priority(4)
        priority(4)=0
       endif
      else
       write(6,1)
 1       format(' Unexpected status from Dedispersor ')
       pause' Hit CR to stop'
       call IRQ_EN
```

```
      stop
     endif

     if(dmx(IRQ$NR-10-2).eq.ndmtot)then
      dmx(IRQ$NR-10-2)=0
      justend(IRQ$NR-10-2)=.true.
      end=.true.
      do j=1,4
       if(dmx(j).ne.0)then
        end=.false.
       endif
      enddo
      if(end)then
       nbeam=nbeam+1
       call newbeam(nbeam,b_name)
       call FREEVME
       current_process_nr=current_process_nr+1
       do j=1,4
        justend(j)=.false.
       enddo
      endif
     endif

     master(IRQ$NR-10-2)=.false.
     length=4
     dummy=send860(dmbyte,4)

     call IRQ_EN

     return
 10     format(a)
     end
C ***************************************************************
```

c **FILE NAME: DEDISP.FOR**

```
$DEBUG
     Subroutine DEDISPERSOR_SET_DM(sh,mask_cs,port,shh)

     implicit none

     integer*4 val4,control_a,control_b,sh(0:127),shh(0:127)
     integer*2 value,address,mask_cs,sh2(0:127)
     integer*4 j,ii,kd,kp,n,port,m
     equivalence (val4,value)

c       master reset
     val4=239
     address=648
```

```
     call out$W (value,address)
     val4=255
     address=648
     call out$w (value,address)

c       master control request
     val4=251
     address=648
     call out$W (value,address)
     val4=255
     address=648
     call out$W (value,address)

c      abilito bus dedispersor factor & mem data in, reset ded.
autoincr.
       val4=31
       address=648
       call out$w (value,address)
       val4=159
       address=648
       call out$w (value,address)

c  scrivo  128 valori di dedispersione
      n=0
      do n=0,127
C             val4=n
C             address=646
C             call out$w(value,address)
          val4=sh(n)
          shh(n)=sh(n)
          sh2(n)=value
C             address=642
C             call out$W(value,address)
      enddo
      address=642
      n=128
      call outp$b (sh2,n,address)

c        write(6,'(''  Hit CR to proceed '')')
c        read(5,*)

c       seleziona blocchi canali attivi

     address=644
     call out$W (mask_cs,address)

c       inizializza scheda mux i860 a servire 860 voluto
c       e genera primo clock per parola 0 (low byte)
     val4=port+128
     address=656
     call out$w (value,address)
     val4=port
```

```
      address=656
      call out$w (value,address)
      val4=port+128
      address=656
      call out$W (value,address)

c       disabilito bus dedispersor factor & memory input
        val4=255
        address=648
        call out$w (value,address)
        return
        end
```

Technical Specification of the Number Smasher-860 FIFO Interface

V 1.0

INTRODUCTION

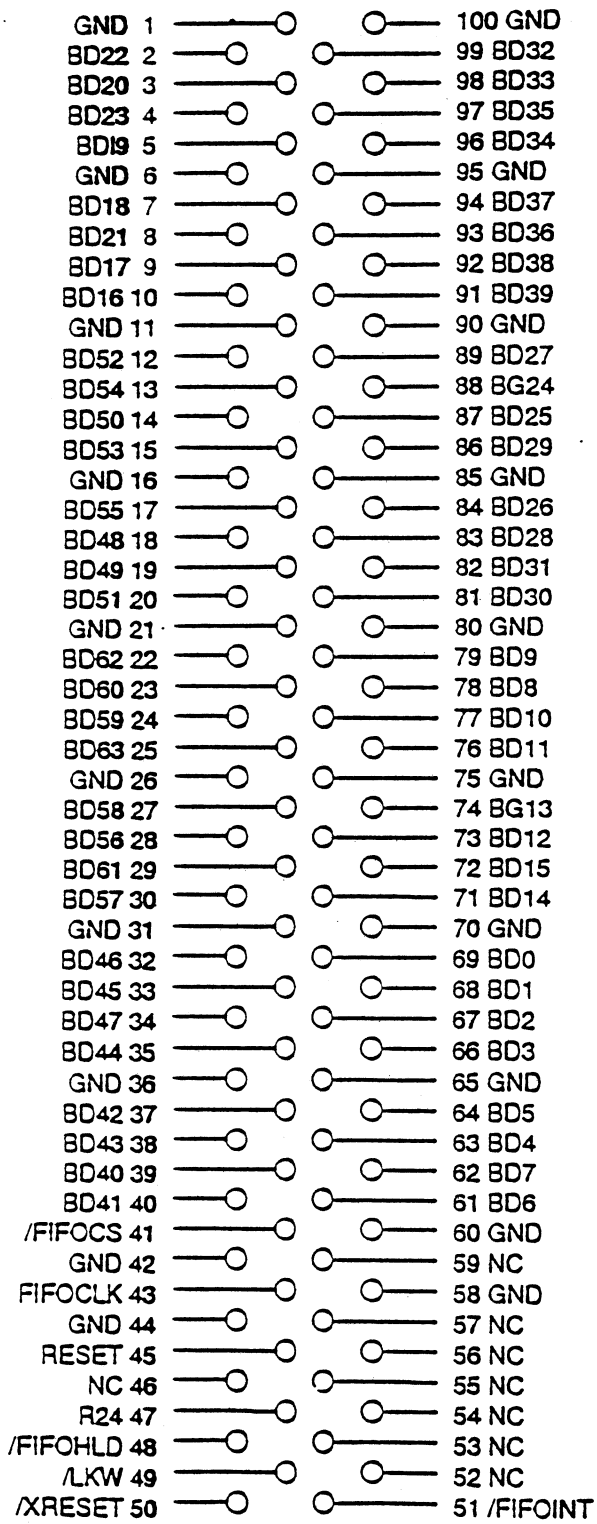
This document describes the 100-pin FIFO interface connector on Microway's Number Smasher-860, and some details on designing PC adapter boards that use this interface. The FIFO interface is used to connect Microway's FIFO I/O boards (both ISA and EISA versions) to the Number Smasher-860. This specification is provided to assist hardware designers who are interested in developing custom peripherals to the Number Smasher-860 using the interface. While this description is considered accurate, no guarantee is made by Microway. This specification and the hardware it describes are subject to change without notice.

SIGNAL DESCRIPTIONS

/XRESET	(Input)	Active low board reset input; has 3.3K pullup. Asserting /XRESET will cause RESET, below, to assert.
/FIFOHLD	(Input)	Active low "wait" signal; inserts wait states during FIFO data transfer cycles.
/FIFOINT	(Input)	Active low interrupt request to the i860. OR'ed with all other i860 hardware interrupt signals to drive i860 INT/CS8 pin. Has 3.3K pullup.
RESET	(Output)	Active high CPU reset. This signal drives the i860 RESET pin. The i860 can be reset from either the on-board link interface or the FIFO /XRESET signal.
FIFOCLK	(Output)	Reference clock, same frequency as i860 (33 MHz or 40 MHz).
/FIFOCS	(Output)	Active low. On-board PALS drive /FIFOCS low during FIFO read or write transfers. /FIFOCS's active width can be increased by asserting /FIFOHLD.
/LKW	(Output)	Active low. With /FIFOCS, indicates a FIFO write cycle.
A24	(Output)	This is a direct connection to the i860 A24 address line. A24 is used to select whether the FIFO data port (A24=1) or control port (A24=0) is being accessed.
BD0-BD63	I/O	This is the buffered i860 data bus. 74F245 transceivers are used on the Number Smasher-860 board to drive this data bus. The i860 is directly connected on the opposite side of the transceivers.
NC		No Connection on the current Number Smasher-860. RESERVED for future definition by Microway.
GND		Tied to the PCB ground plane (0V).

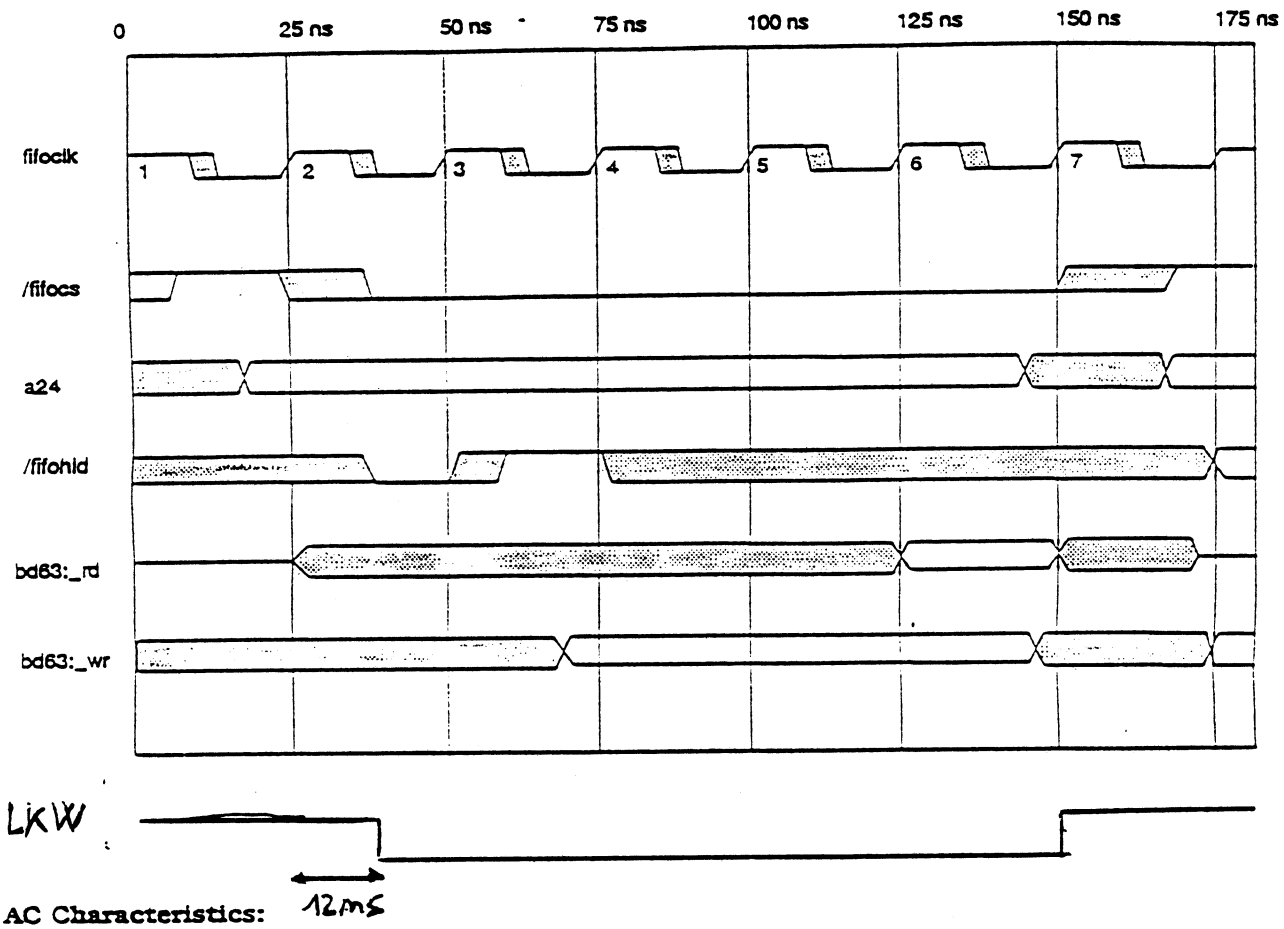
CONNECTOR PINOUT

The 100-pin connector's signal assignment is shown below. This is a view of the connector's hole pattern on the component side of the Number Smasher-860 printed circuit board (PCB).



..C. SPECIFICATION & TIMING DIAGRAM

Timing Diagram



AC Characteristics:

All figures below are in nanoseconds.

40MHz clock speed is assumed. Designing to these specs will also ensure compatibility with 33MHz product.

Description	Edge	Time	Notes
All cycles:			
FIFOCS active delay:	2	12 ns Max	(Notes 1,2)
FIFOCS inactive delay:	7	19 ns Max	
FIFOHLD setup:	3.4	12 ns Max	(Notes 1,2)
FIFOHLD hold:	3.4	2 ns Min	
LKW valid delay:	2	12 ns Max	
LKW invalid delay:	7	1 ns Min	
A24 valid delay:	1	18 ns Max	
A24 invalid delay:	7	-4.5 ns Min	

Description	Edge	Time	Notes
Read cycles:			
DATA setup:	7	24 ns Min	
DATA hold:	7	0 ns Min	
FIFOCS to DATA low impedance:	n/a	0 ns Min	(Note 3)
DATA high impedance delay:	7	21 ns Max	
Write cycles:			
DATA valid delay:	3	21 ns Max	
DATA invalid delay:	7	-3 ns Min	

Notes:

1. If FIFOHLD is tied inactive, the cycle length is five CPU clocks, and FIFOCS is active for four clocks. Add one CPU clock for each P2 CLK rising edge on which FIFOCS and FIFOHLD are both active. The diagram shows one such extra clock.

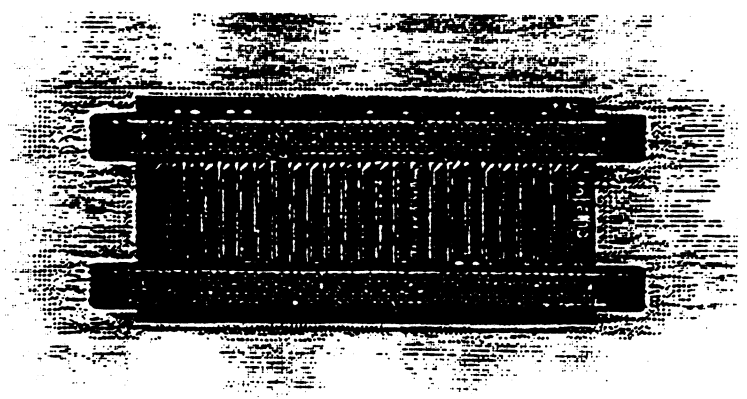
2. FIFOCS valid delay (max) and FIFOHLD setup (min) are such that it is not realistically possible to drive FIFOHLD active in response to FIFOCS active. Therefore, if cycles longer than 5 CPU clocks are desired, the idle state of FIFOHLD (between cycles) should be active. In this case, the fastest cycle will be 6 CPU clocks.

3. FIFOCS to data low impedance delay is a signal-to-signal spec. All others are clock-to-signal or signal-to-clock. If the data bus driver is enabled in response to FIFOCS, or some clock edge thereafter, this spec will automatically be met.

4. All valid delays, setup & hold times are relative to the rising edge of the clock on the P2 connector. This is not the same clock as the CPU clock, nor is it the same as the clock driving the PALs on the 860 board. Worst-case skews have already been incorporated in these figures.

INTERCONNECT BOARD

The Interconnect Board shown below is used to connect the Number Smasher-860 and a FIFO I/O adapter. This board and all of the connectors shown are available from Microway. Contact Microway for pricing.



Bibliografia

1) **"Il rivelatore a 128 canali del sistema pulsar di Medicina"**
A.Cattani, C.Bortolotti,,A.Cattani, N.D'Amico, A.Maccaferri, S.Montebugnoli
IRA 169/92

2) **"Il digitalizzatore ad 1 bit - 128 canali del sistema pulsar di Medicina"**
A.Cattani, S.Montebugnoli, N.D'Amico, A.Maccaferri
IRA 170/92

3) **"Scheda di interfaccia per acquisizione dati ad alta velocitá su bus ISA"**
A.Maccaferri, N.D'Amico
IRA 154/92

4) **"Pulsar astronomy at the Northern Cross"**
N.D'Amico, C.Bortolotti, A.Cattani, F.Fauci, G.Grueff, A.Maccaferri, S.Montebugnoli, M.Roma, G.Tomassetti.
IRA 137/90

5) **"Il dedispersore digitale del sistema "*Pulsar*" di Medicina"**
A.Maccaferri, N.D'Amico
IRA 198/95

6) **"Microway NDP Number Smasher 860 technical reference"**

7) **"The IBM PC from the inside out"**
Murray Sargent III & Richard L.Shoemaker
Addison Wesley Publishing Company, Inc. 1986