TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The H. S. CAENET Network is a send and receive half duplex system; It allows asynchronous serial transmissions (1 MBaud rate) of data packets along a simple 50 Ω coaxial cable (RG174). The line must be terminated at both ends. Several devices (H. S. CAENET nodes) are able to share the same media to transmit and receive data. Each node is able to receive the serial data packet and store it automatically in a FIFO (RX FIFO) and transmit the data contained in another FIFO (TX FIFO). Both FIFOs are available in two versions:

- 512 byte deep;
- 4096 byte deep.

The H.S. CAENET Node listens for clear coax before transmitting but it is not able to detect collisions on the cable; for this reason it is important to avoid line contention; i. e., the Nodes should not attempt to transmit at the same time.

Usually transfers between H. S. CAENET nodes take place according to the typical Master/Slaves communication:

- There is a single Master: the H. S. CAENET Controller;

- The Slaves are daisy chained on the network, and are identified by an address code (from 1 to 99);

- The H. S. CAENET Master begins the transmission, all the Slaves receive the data and only the addressed Slave accesses the serial line to transmit the data requested by the Master;

- The transmitted and received data are not fragmented, i. e. they fit in a single packet;

- Each packet is composed of a fixed format Header followed by operation dependent raw Data; the Header consists in 16-bit words in little endian order;

- The maximum data packet length is 4096 bytes.

The address of the H. S. CAENET node (Station #) is selectable from 1 to 99. In this way up to 99 modules may be controlled from a single point via one of the following CAEN H. S. CAENET Controllers:

- A199HS, H. S. CAENET G64 Controller;
- A250 H. S. CAENET Manual Controller;
- A303 H. S. CAENET PC Controller;
- C117B H. S. CAENET CAMAC Controller;
- V288 H. S. CAENET VME Controller.

2. H. S. CAENET NODE

2.1. INTRODUCTION

This chapter contains a description of the Hardware and some technical information for a Hardware/Software Designer that must implement a H. S. CAENET Node on His/Her own device.

Each unit connected to a H. S. CAENET Network must be equipped with a H. S. CAENET NODE BOARD (CAEN Mod. A464).

A detailed description of the Hardware characteristics can be found in § 2.2 through § 2.10.

Some hints on the development of a Software driver for a H. S. CAENET NODE can be found in § 2.4.

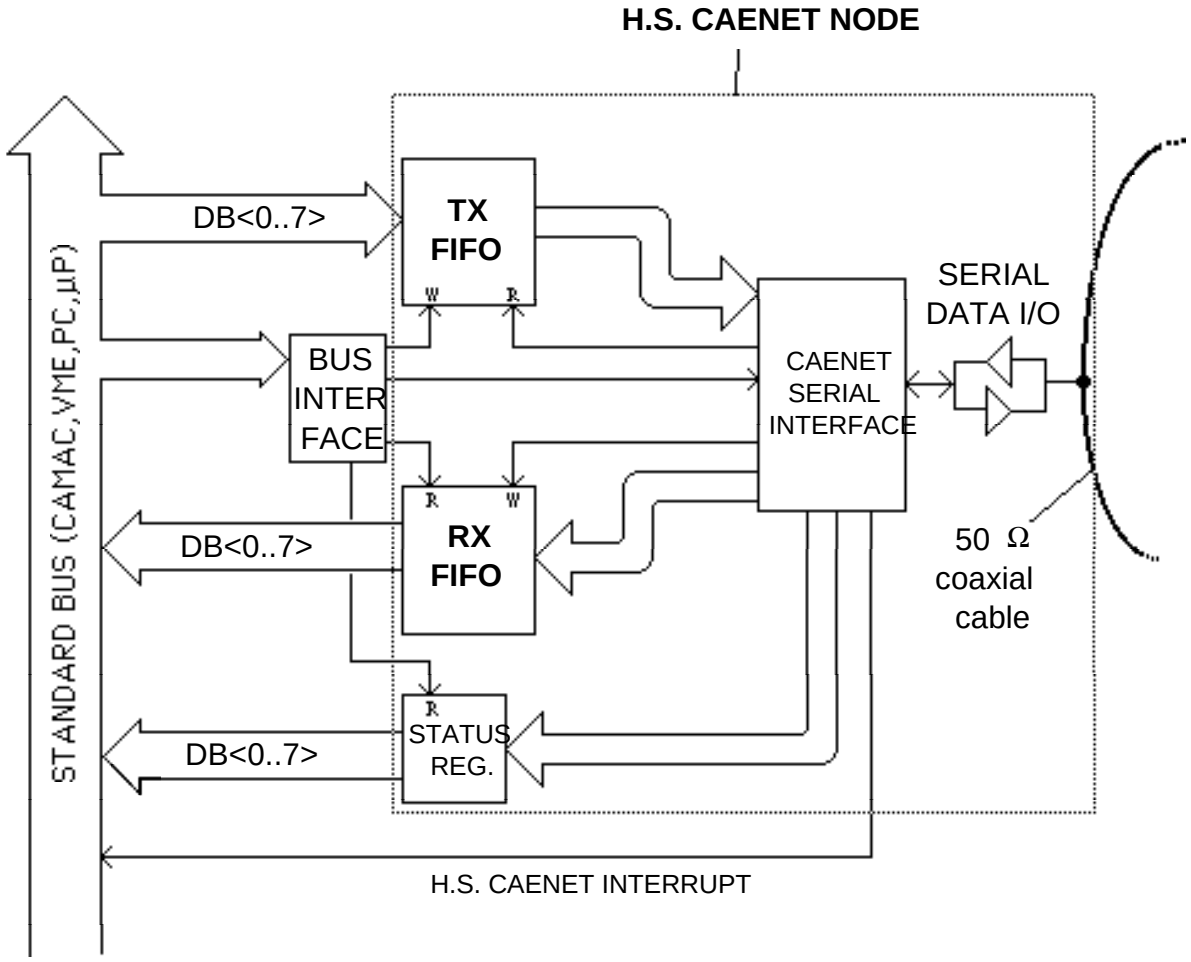


Fig. 2.1: H. S. CAENET Node Block Diagram

2.2. A464 BOARD

The CAEN Mod. A464 is a piggy-back module (size: 74.4x35.5 mm) designed to remotely control and monitor stand-alone systems via a H. S. CAENET communication network and software.

The A464 requires a single +5 V supply. All the inputs and outputs are TTL level compatible.

The unit gives the complete functionality of a H. S. CAENET Node including the line driver/receiver and the two FIFO buffers for the transmitted and received data packets; both FIFOs are 512 or 4096 bytes deep.

It can be used by the stand-alone system designer as an electronic component and can be driven by the host microprocessor board either in polling or interrupt mode.

The A464 requires an external clock: the clock frequency must be three times the data transfer rate (3.072 MHz for a 1 MBaud rate). The tolerance on this frequency must be ≤1%. Commercially available oscillators usually satisfy this requirement.

The host microprocessor recognizes this board as an 8 bit peripheral device composed of a collection of registers and two memory buffers arranged in FIFO logic (TX FIFO, RX FIFO).

The A464 is able to receive a H. S. CAENET serial data packet and store it automatically in the RX FIFO (Receive mode).

Moreover, by writing in an appropriate register the A464 transmits the data contained in the TX FIFO (Transmit mode).

Interrupts are generated by the Model A464 upon occurrence of any of the following:

- completion of a transmission of a data packet;
- receipt of a data packet;
- complete unloading of the RX FIFO.

The cause of the interrupt can be ascertained by reading the Interrupt status bits in the A464 STATUS REGISTER; this register contains the current status of the H. S. CAENET Node as shown in Table 2.2.

2.3. A464 REGISTERS

The following tables describe the A464 8-bit Internal Registers.

Table 2.1: A464 Registers

Address/ Operation	Register/Buffer	Description
0 Write	TX FIFO	Buffer which is loaded with the data to be transmit.
1 Write	START TX	The Node enters in transmit mode.
3 Write	RESET	Clears TX and RX FIFO and every interrupt pending.
0 Read	RX FIFO	Buffer where the Node stores the received data.
1 Read	STATUS REG.	Contains the status bits of the Node.
2 Read	RESET INTERRUPT	Resets the interrupt.
3 Read	CLEAR RX FIFO	Clears the RX FIFO[1].

Table 2.2: A464 Status Register Configuration

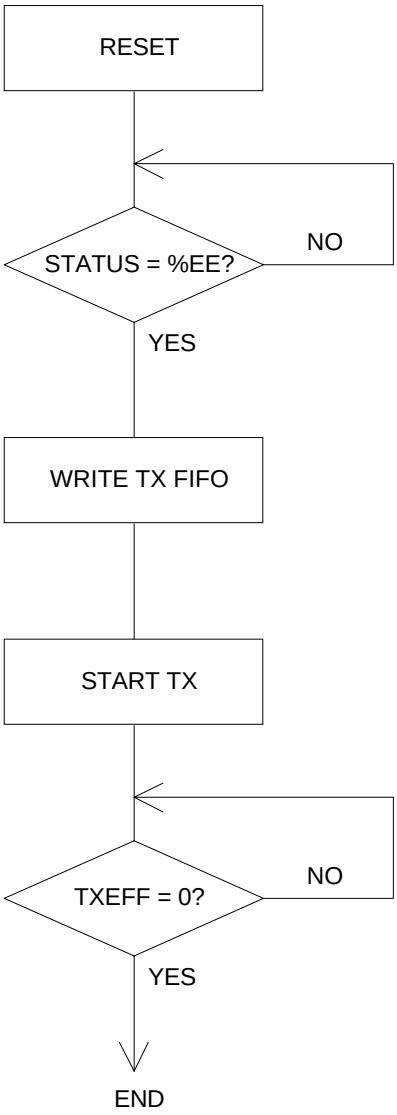
Bit	Name	Description
0	RXFEM	RX FIFO empty: if = 0 indicates that the RX FIFO is empty.
1	RFEFF	RX FIFO empty FLIP FLOP, interrupt status bit: if =0 indicates that the RX FIFO has been completely unloaded.
2	RXEFF	RX END FLIP FLOP, interrupt status bit: if = 0 indicates that the H.S. CAENET Node has received a data packet.
3	RESTART	RESTART mode: if = 0 indicates that the H.S. CAENET Node is in Restart mode.
4	TFEM	TX FIFO empty: if = 0 indicates that the TX FIFO is empty.
5	TXEFF	TX END FLIP FLOP, interrupt status bit: if = 0 indicates that the H.S. CAENET Node has transmitted a data packet.
6	RXACT	RECEIVER ACTIVE: if = 0 indicates that the H.S. CAENET Node is in Receive mode.
7	TXACT	TRANSMITTER ACTIVE: if = 0 indicates that the H.S. CAENET Node is in Transmit mode.

[1]This operation returns also information on the size of the FIFOs on the A464 board: bit 6 is set to 0 in case of a 512 deep FIFO, bit 6 is set to 1 in case of a 4096 deep FIFO.

2.4. FLOW CHARTS

The following flow charts indicate respectively the Data Transmission routine and the Interrupt Handler of a generic A464 Software driver.

The described routines are shown in a simplified version. In order to handle every problem on the network or a module malfunctioning it is recommended to use time-out controls in each testing loop; e. g., if the line is not clear, the node remains indefinitely in restart mode.



For what concerns the behaviour of the caller, there are two distinct cases:

a) The caller is a H. S. CAENET Master.

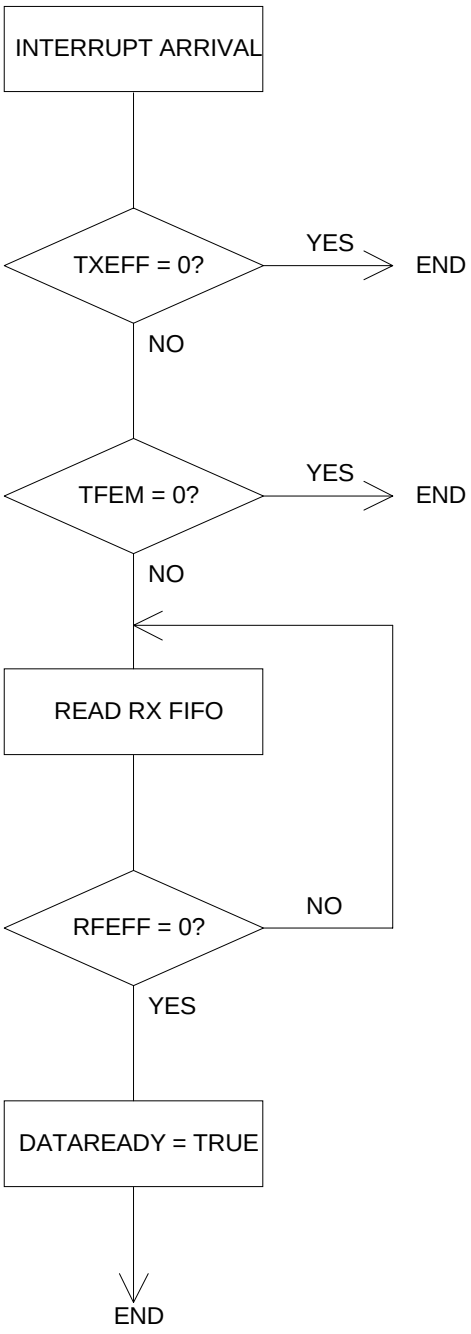
Upon exit from this routine, the caller must take care of starting a timer to be sure that the Slave answers within a reasonable time (e.g., a few msec). This routine represents a H. S. CAENET command.

b) The caller is a H. S. CAENET Slave.

Upon exit from this routine, the caller has finished because the routine represents the answer to a previous H. S. CAENET command.

N. B. For more detailed information on the H. S. CAENET protocol, see § 3.

Fig. 2.2: Data Transmission Flow Chart



The driver must also have a Read Data routine (not shown here) that returns to the caller the data arrived via H. S. CAENET provided that the Data Ready flag is TRUE.

For what concerns the meaning of the arrived data, there are two distinct cases:

a) The caller is a H. S. CAENET Master.

The data represent the Slave answer to a previous H. S. CAENET command.

b) The caller is a H. S. CAENET Slave.

The data represent a H. S. CAENET command.

N. B. For more detailed information on the H. S. CAENET protocol, see § 3.

Fig. 2.3: Interrupt Handler Flow Chart

2.5. A464 CONNECTION DIAGRAM

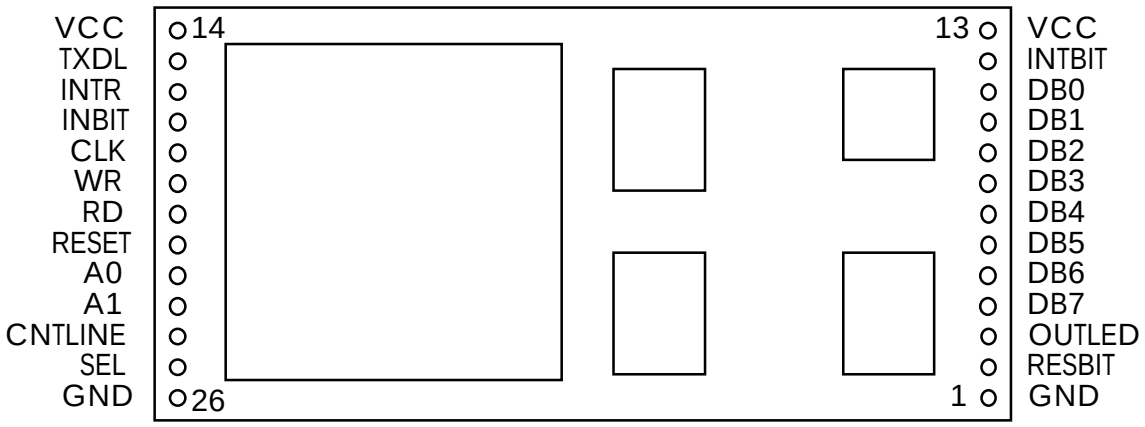


Fig. 2.4: Connection Diagram

2.6. A464 MECHANICAL DATA

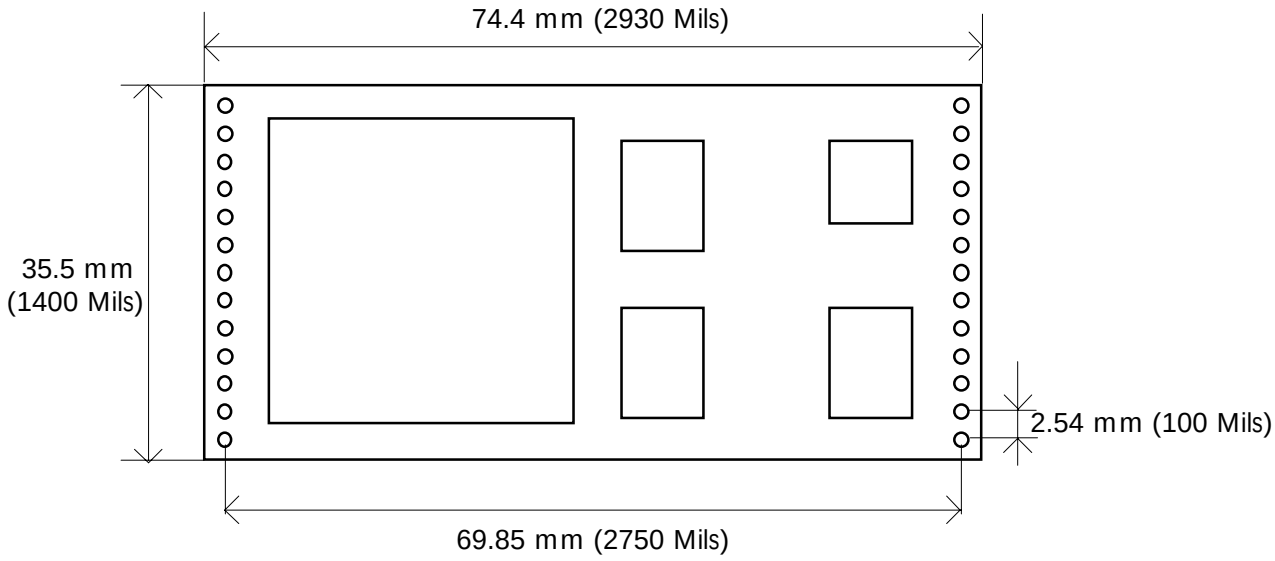


Fig. 2.5: Mechanical Description

2.7. A464 PINS DESCRIPTION

Table 2.3: A464 Pins Description

Symbol	Pin No.	Type	Name and Function
VCC	13, 14	I	Supply: +5 V ±5% Supply.
GND	1, 26	I	Ground.
SEL	25	I	Select: A low on this pin enables the RD and WR operations.
WR	19	I	Write: A low on this pin when SEL is low enables the writing in the TX buffer, the start of transmission, the OUTLED output and the A464 reset command (see Table 2.1).
RD	20	I	Read: A low on this pin when SEL is low enables the readout of the RX buffer and Status Register, the reset of the Interrupt and the clearing of the RX buffer (see Table 2.1).
CLK	18	I	Clock: 3.072 MHz external clock for the module.
A0-A1	22, 23	I	Address lines: these pins act in conjunction with the SEL, WR and RD pins. They are used to decode the operation type (see Table 2.1).
DB0-DB7	11, 4	I/O	Bidirectional Data Bus: TX buffer, RX buffer and Status information is transferred via this bus.
RESET	21	I	Resets the A464 module: active according to the RESBIT input. The minimum pulse width is 1μsec.
RESBIT	2	I	Reset polarity selection: 1= RESET is active high; 0= RESET is active low.
INTR	16	O	Interrupt: active according to the INTBIT input upon occurrence of an Interrupt condition. Remains in the active state until a Reset Interrupt command or a General Reset is performed.
INTBIT	12	I	Interrupt polarity selection: 1= INTR is active high; 0= INTR is active low.
OUTLED	3	O	Output LED driver: this line goes low when at least one datum is present in the TX buffer and upon receipt of a Write access with A1= 1, A0= 0. It goes high after a General or an Interrupt Reset.
CNTLINE	24	I/O	CAENET Line: must be connected to the LEMO CAENET connectors. It is the physical input of the INBIT Receive Buffer shorted with the physical output of the TXDL Transmit Buffer (see Fig. 2.6).
TXDL	15	O	Transmit Data Line: active low. See Note below.
INBIT	17	I	Input Data Line: active low. See Note below.

Note: TXDL and INBIT are normally not used. They can be used whenever the User wishes to utilize its own transceiving logic or wants to opto-isolate the CAENET line. The latter feature is already implemented in the WIENER crates, in the CAEN Mod. A128HS and, optionally, on the SY527 System (Mod. A567). In these cases, the A464 board must be modified by removing the U6 SMD chip (26C32) on the lower face of the board (see Fig. 2.6).

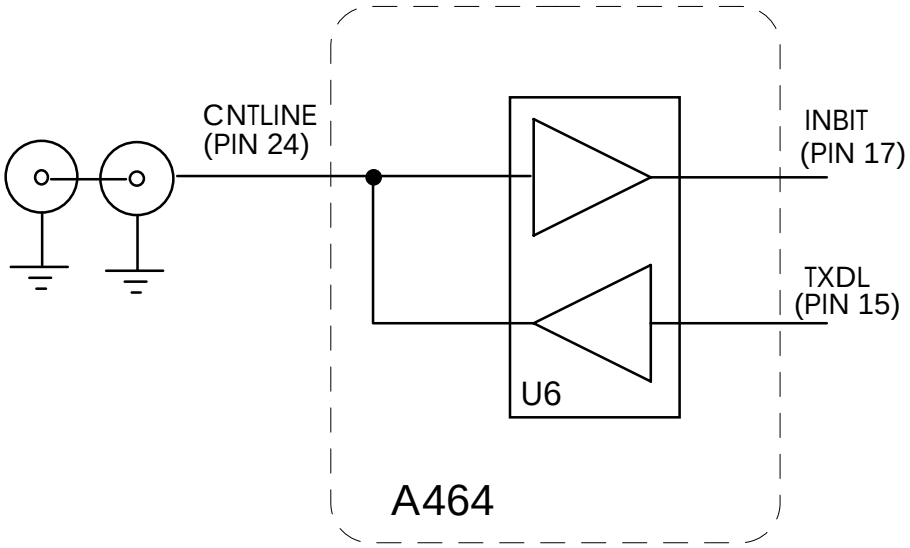


Fig. 2.6: A464 CNTLINE Section

2.8. PHYSICAL LINE AND NODE CAPABILITIES

Output Driver Signal characteristics:	0 to +4 V on 50 Ω Impedance
Input Receiver characteristics:	+1.2 V threshold discriminator
RG174 Cable attenuation:	-6.2 dB/100 m @ 1 MHz
H. S. CAENET Node attenuation:	≈-0.07 dB @ 500 kHz

- It is worth noting that the maximum frequency of the H. S. CAENET signal is 500 kHz (a stream of bits toggling between 0 and 1 transmitted at the rate of 1 MBaud).
- Through the use of appropriate adapters, one can connect the H. S. CAENET nodes via an RG58 cable, that has better attenuation features (1.4 dB/100 m @ 1 MHz) at approximately the same cost.

From the above data it is possible to calculate the maximum number of H. S. CAENET nodes given a certain line length and vice versa, because the two values ase strongly interconnected.

The theoretical maximum number of 100 nodes (1 Master + 99 Slaves) cannot be exceeded.

2.9. POWER REQUIREMENTS

+5 V 250 mA (typical)
500 mA (maximum)

2.10. A464 WAVEFORMS AND TIMINGS

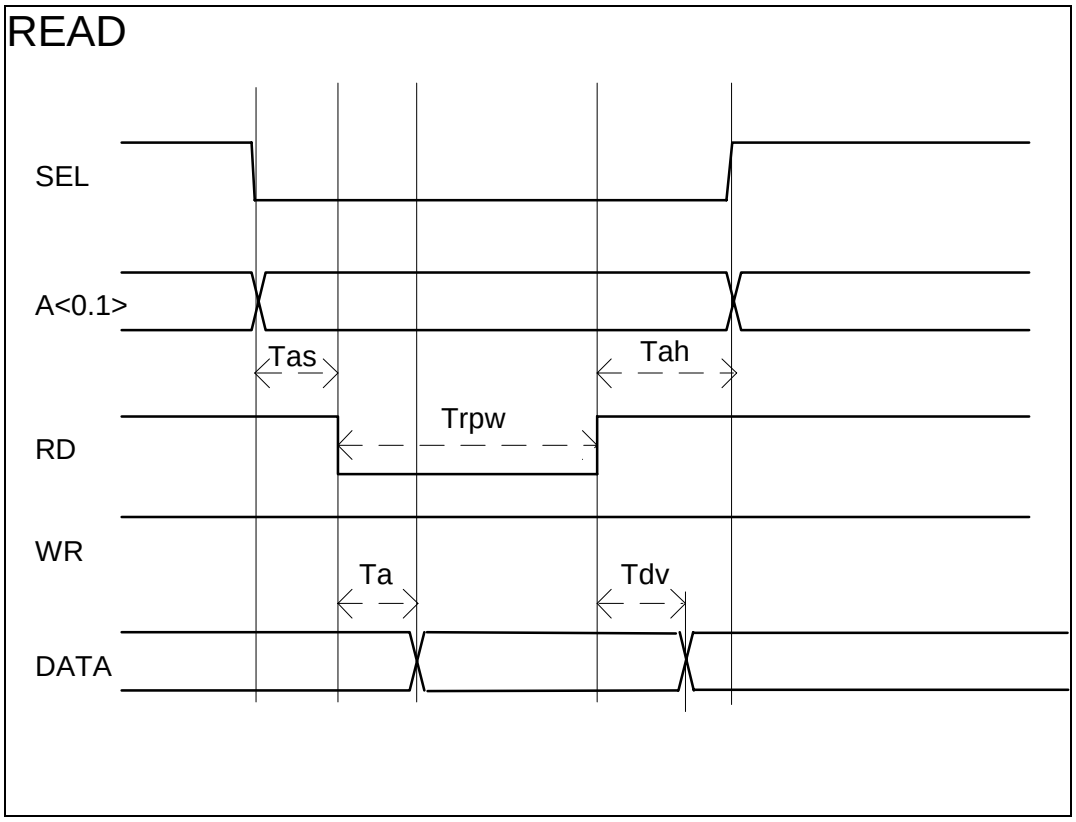


Fig. 2.7: READ Waveforms

Table 2.4: A464 READ Timings

	MIN	MAX	
Tas	10 ns		
Tah	10 ns		
Tdv	5 ns		In any case
Ta		105 ns	RX BUFFER case
Trpw	80 ns		RX BUFFER case
Ta		50 ns	RD STATUS REG case
Trpw	50 ns		RD STATUS REG case RESET INTERRUPT case CLEAR RX BUFFER case

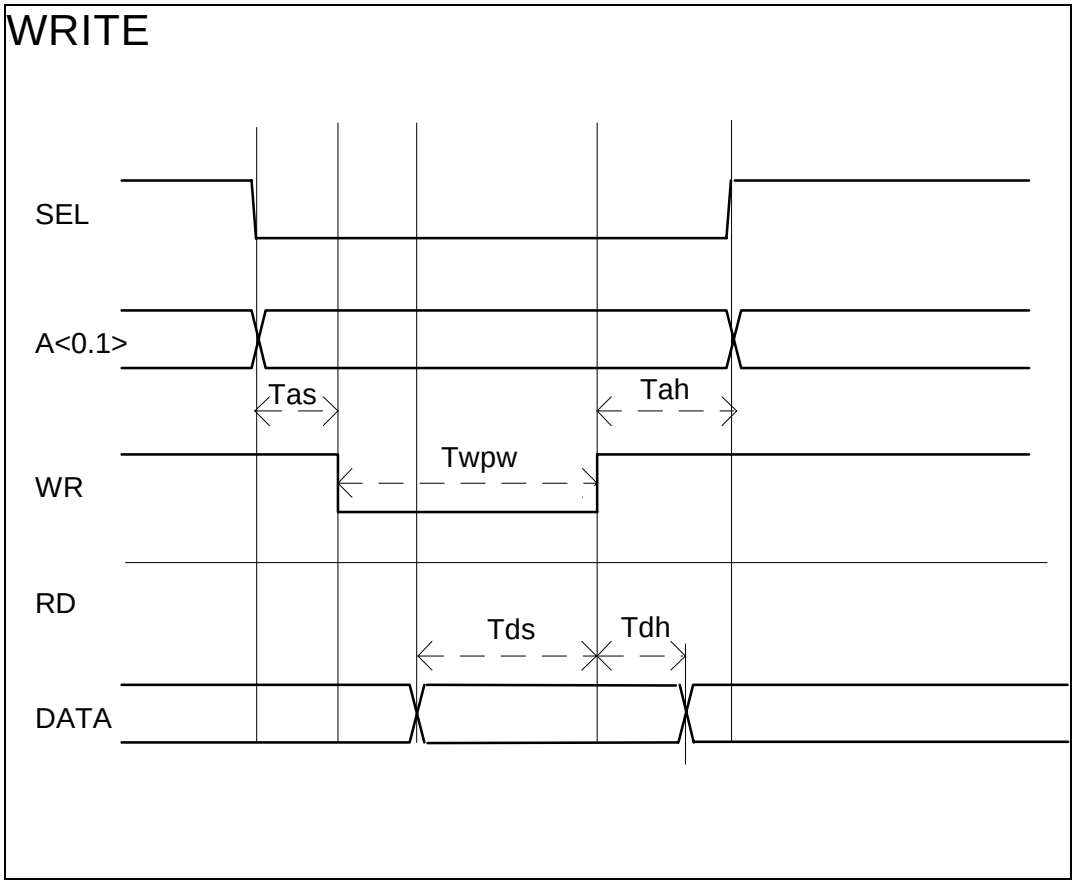


Fig. 2.8: WRITE Waveforms

Table 2.5: A464 WRITE Timings

	MIN	
Tds	40 ns	
Tdh	35 ns	
Tas	10 ns	
Tah	10 ns	
Twpw	80 ns	TX BUFFER case
Twpw	25 ns	START TX case OUTLED ON case RESET case

3. H. S. CAENET PROTOCOL

3.1. INTRODUCTION

This Chapter describes the software protocol to be placed upon the H. S. CAENET Node. The protocol is completely independent from the hardware specifications of the H. S. CAENET Node itself, nonetheless it is mandatory in order to control the existing H. S. CAENET Nodes.

Usually, the transfers between H. S. CAENET nodes take place according to the typical Master/Slaves communication:

- There is a single Master: the H. S. CAENET Controller;
- The Slaves are daisy chained on the network, and are identified by an address code (from 1 to 99);
- The H. S. CAENET Master begins the transmission, all the Slaves receive the data and only the addressed Slave accesses the serial line to transmit the data requested by the Master;
- The transmitted and received data are not fragmented, i. e. they fit in a single packet;
- Each packet is composed of a fixed format Header followed by operation dependent raw Data; the Header consists in 16-bit words in little endian order;
- The maximum data packet length is 4096 bytes.

3.2. MASTER-TO-SLAVE DATA COMPOSITION

The Master-to-Slave data have to be written in the Node TX FIFO by performing subsequent write accesses as follows.

Table 3.1: Master-to-Slave Data Composition

Order	Datum (HEX)	Meaning
1	%000n	H. S. CAENET Controller Identifier Code
2	%00XX	Node Number
3	Code	First word of the operation Code to be performed
4 to 2048	Code/Set	Any subsequent words of the Code or Set values

- The n in the Identifier Code can be 0 in case of an A250 Manual Controller or 1 for all other Controllers.
- The Operation Codes may be some words in length and may be followed by several set values. Particular care must be taken onto Code %0000. It has the same meaning for all Slaves. The format of the response is explained in § 3.3.2.

As soon as the data packet has been stored in the TX FIFO, it can be transmitted on the cable.

After any transmission, the Master reads the Slave response.

3.3. SLAVE-TO-MASTER DATA COMPOSITION

The answer data coming from the Slave Node is automatically stored into the RX FIFO and therefore is available to the Master.

The following Table shows the structure of the Slave-to-Master data packet:

Table 3.2: Slave-to-Master Data Composition

Order	Datum	Meaning
1	%000n	H. S. CAENET Controller Identifier Code
2	Error Code	Error code
3 to 2048	value	Any following Parameter value

The n in the Identifier Code can be:

- 0 in case of an answer to a A250 Manual Controller;
- 1 in case of an answer to all other Controllers when packet length ≤ 256 words;
- 2 in case of an answer to all other Controllers when 256 < packet length ≤ 2048 words;

3.3.1. ERROR CODES DESCRIPTION

All Error Codes are coded as negative values (MSB set to 1). The generic Error codes are described in the following Table. For the Controller-dependent Error Codes refer to the single Controllers User's Manuals.

The H. S. CAENET protocol allows the use of any Slave-dependent Error Code providing that is not in the following table and is a negative value.

Table 3.3: Error Codes

Datum (Hex)	Meaning
%0	Successful operation.
%FF00	Module Busy; it has tried to effect an operation while the module is performing a previous operation.
%FFFD	No data to be transmitted; it has tried to start a transmission with the Transmit data Buffer empty (H. S. CAENET Controller error message).
%FFFE	The H. S. CAENET Controller identifier is incorrect (H. S. CAENET Controller error message).
%FFFF	The addressed module does not exist. This message are generated after a period of 500 msec (H. S. CAENET Controller error message).

3.3.2. NODE IDENTIFIER PACKET
(Response To Code %0000)

All the Slaves must answer to this Code. The response contains in the low byte the ASCII code of the string of characters identifying the name of the Slave. Table 3.4 shows a practical example for a real CAEN Model (SY527 Universal Multichannel Power Supply System).

Table 3.4: Module Identifier Data Packet Structure

Word	Contents	
	db15..8	db7..0
3	0	"S"
4	0	"Y"
5	0	"5"
6	0	"2"
7	0	"7"
8	0	" "
9	0	"V"
10	0	"2"
11	0	"."
12	0	"1"
13	0	"6"

4. SOFTWARE EXAMPLES

4.1. INTRODUCTION

This Chapter contains two code samples that allow an application program to control a H. S. CAENET Network via a CAEN Mod. V288 and a CAEN Mod. A303 respectively.

4.2. USING THE V288 CONTROLLER

The details of using the Mod. V288 to communicate with the Mod. SY127 are explained by means of examples:

- VMECAENET.H: Declaration for communication via VME with the Mod. V288;

- VMCAENET.C: Caenet Package for V288 Module.

These two listings describe the function and general design of a driver for the Mod V288; all the possible errors are handled, included the VME Buserror.

```
/*************************************************************************/
/*                                                                       */
/*             ------       C . A . E . N .   SpA       ------           */
/*                                                                       */
/*    VMCAENET.H - Declarations for communication with V288 Module       */
/*                                                                       */
/*************************************************************************/

#ifndef   uchar
#define   uchar                        unsigned char
#endif
#ifndef   ushort
#define   ushort                      unsigned short
#endif

/* Constants for vme_cycles routines */
#define   BYTE                                    1
#define   WORD                                    2
#define   LWORD                                   4

/*
  Errors returned by caenet_read and caenet_write; the positive ones
  are depending from V288 Module and not from CAENET network
*/
#define   TUTTOK                                  0
#define   E_NO_Q_IDENT                            1
#define   E_NO_Q_CRATE                            2
#define   E_NO_Q_CODE                             3
#define   E_NO_Q_DATA                             4
#define   E_NO_Q_TX                               5
#define   E_NO_Q_RX                               6
#define   E_LESSDATA                              7
#define   E_BUSERR                                8

/* Number of iterations before deciding that V288 does not answer */
#define   TIMEOUT                                -1

#define   Q                           (ushort)0xfffe
#define   V288                                    1

/* Registers of V288 Module */
#define   STATUS                     (v288addr+0x02)
#define   TXMIT                      (v288addr+0x04)

#define   LOBYTE(x)                (uchar)((x)&0xff)
#define   HIBYTE(x)       (uchar)(((x)&0xff00) >> 8)

/*
  Interface between the user program and V288; these functions are defined
  in file Vmcaenet.c
*/
int  caenet_read();
int  caenet_write();
int  read_caenet_buffer();

/* Declarations of Global Variables defined in the user program */
extern  unsigned       v288addr,craten;
extern  ushort                   code;
```

```
/*************************************************************************/
/*                                                                       */
/*             ------       C . A . E . N .   SpA       ------           */
/*                                                                       */
/*    VMCAENET.C - Caenet Package for V288 Module                        */
/*                                                                       */
/*************************************************************************/

#include "vmcaenet.h"


/***-----------------------------------------------------------------------

  Read_data

    -----------------------------------------------------------------------***/
int read_data(datovme)
ushort *datovme;
{
ushort q=0;
vme_read(v288addr,datovme,WORD);
vme_read(STATUS,&q,WORD);
return((q == Q) ? TUTTOK : TIMEOUT);
}

/***-----------------------------------------------------------------------

  Wait_resp

    -----------------------------------------------------------------------***/
int wait_resp(datovme)
ushort *datovme;
{
int i=0;
ushort q=0;
while(i!=TIMEOUT && q!=Q)
     {
      vme_read(v288addr,datovme,WORD);
      vme_read(STATUS,&q,WORD);
      i++;
     }
return((i == TIMEOUT) ? TIMEOUT : TUTTOK);
}

/***-----------------------------------------------------------------------

  Send_comm

    -----------------------------------------------------------------------***/
int send_comm(vmeaddress,datovme)
unsigned int vmeaddress;
ushort datovme;
{
int i=0;
ushort q=0;
while(i!=TIMEOUT && q!=Q)
     {
      if(!vme_write(vmeaddress,&datovme,WORD))
         return E_BUSERR;
      vme_read(STATUS,&q,WORD);
      i++;
     }
return((i == TIMEOUT) ? TIMEOUT : TUTTOK);
}
```

```
/***-----------------------------------------------------------------------

  Caenet_read: Called by user programs to load "byte_count" bytes from
               CAENET into the buffer pointed by "*dest_buff".

               The VME address of V288, the CAENET crate number and the
               CAENET code are found in global variables.

               Caenet_read returns TUTTOK = 0 if everything has worked;
               It returns one from seven different errors (defined as
               positive constants in Vmcaenet.h) if it has received one
               error which strictly depends from V288 Module;
               It returns a negative error (depending from the CAENET slave
               module) if the CAENET communication has not worked.

     Remember: Module V288 can return three "general" negative errors
               related to the CAENET network that this routine does not
               handle separately from the "slave specific" ones.

    ------------------------------------------------------------------***/
int caenet_read(dest_buff,byte_count)
uchar *dest_buff;
int byte_count;
{
int i,esito;
ushort mstident=V288,datatemp;
short dato;

if((esito=send_comm(v288addr,mstident)) == TIMEOUT)
   return E_NO_Q_IDENT;
else if(esito == E_BUSERR)
   return esito;

/* Transmit Crate Number */
if((esito=send_comm(v288addr,(ushort)craten)) == TIMEOUT)
   return E_NO_Q_CRATE;
else if(esito == E_BUSERR)
   return esito;

/* Transmit Code       */
if((esito=send_comm(v288addr,(ushort)code)) == TIMEOUT)
   return E_NO_Q_CODE;
else if(esito == E_BUSERR)
   return esito;

/* Start Transmission        */
if((esito=send_comm(TXMIT,mstident)) == TIMEOUT)
   return E_NO_Q_TX;
else if(esito == E_BUSERR)
   return esito;

if(wait_resp(&dato) == TIMEOUT)
   return E_NO_Q_RX;

if(dato == TUTTOK)                          /* Test on the operation     */
   for(i=0;i<byte_count;i+=2)
      {
       if(read_data(&datatemp) == TIMEOUT && i<byte_count-1)
          return E_LESSDATA;
       dest_buff[i]   = HIBYTE(datatemp);
       dest_buff[i+1] = LOBYTE(datatemp);
      }
return dato;
}

/***-----------------------------------------------------------------------
```

```
 Caenet_write: Called by user programs to transfer "byte_count" bytes to
               CAENET from the buffer pointed by "*source_buff".

               The VME address of V288, the CAENET crate number and the
               CAENET code are found in global variables.

               Caenet_write returns TUTTOK = 0 if everything has worked;
               It returns one from seven different errors (defined as
               positive constants in Vmcaenet.h) if it has received one
               error which strictly depends from V288 Module;
               It returns a negative error (depending from the CAENET slave
               module) if the CAENET communication has not worked.

     Remember: Module V288 can return three "general" negative errors
               related to the CAENET network that this routine does not
               handle separately from the "slave specific" ones.

    ------------------------------------------------------------------***/
int caenet_write(source_buff,byte_count)
uchar *source_buff;
int byte_count;
{
int i,esito;
ushort mstident=V288,datatemp;
short dato;

if((esito=send_comm(v288addr,mstident)) == TIMEOUT)
   return E_NO_Q_IDENT;
else if(esito == E_BUSERR)
   return esito;

/* Transmit Crate Number */
if((esito=send_comm(v288addr,(ushort)craten)) == TIMEOUT)
   return E_NO_Q_CRATE;
else if(esito == E_BUSERR)
   return esito;

/* Transmit Code       */
if((esito=send_comm(v288addr,(ushort)code)) == TIMEOUT)
   return E_NO_Q_CODE;
else if(esito == E_BUSERR)
   return esito;

/* Transmit data       */
for(i=0;i<byte_count;i+=2)
   {
    datatemp=(ushort)source_buff[i]<<8 | source_buff[i+1];
    if((esito=send_comm(v288addr,datatemp)) == TIMEOUT)
       return E_NO_Q_DATA;
    else if(esito == E_BUSERR)
       return esito;
   }

/* Start transmission                */
if((esito=send_comm(TXMIT,mstident)) == TIMEOUT)
   return E_NO_Q_TX;
else if(esito == E_BUSERR)
   return esito;

if(wait_resp(&dato) == TIMEOUT)
   return E_NO_Q_RX;

return dato;
}

/***-------------------------------------------------------------------
```

```
  Read_caenet_buffer: Called by user programs to load "byte_count" bytes from
                      CAENET buffer into the buffer pointed by "*dest_buff".

    ------------------------------------------------------------------***/
int read_caenet_buffer(dest_buff,byte_count)
uchar *dest_buff;
int byte_count;
{
int    i;
ushort datatemp;

for(i=0;i<byte_count;i+=2)
   {
    if(read_data(&datatemp) == TIMEOUT && i<byte_count-1)
       return E_LESSDATA;
    dest_buff[i]   = HIBYTE(datatemp);
    dest_buff[i+1] = LOBYTE(datatemp);
   }
return TUTTOK;
}
```

4.3. USING THE A303 CONTROLLER

4.3.1. INTRODUCTION

The following pages describe an example of interfacing to the A303: the DOS device driver CAENET.SYS.

CAENET.SYS permits an application program to control every CAENET device via an A303 PC Controller for H. S. CAENET. It must be invocated in file CONFIG.SYS by the classical call:

```
device = CAENET.SYS
```

If the call is successfull, the following startup message appears during the PC boot phase:

```
Mod. A303 CAENET Driver Installed              Version 1.0
```

The version here described permits the control of A303 only in Memory Space. A custom version which permits the access to A303 in I/O Space is available on request.

In the following we suppose that the application program is written in C Language (in particular the code examples are in Borland Turbo C/C++), but the same considerations are applicabile to programs written in Assembler, Pascal or BASIC provided that the compiler library contains a function which executes a DOS call (e.g. MsDos procedure of Turbo Pascal).

4.3.2. SOFTWARE INTERFACE

Once the driver is in memory, the user program can communicate with it. First one must verify that it responds:

```
#include <fcntl.h>

int dev_handle;

dev_handle = open("CAENETXX", O_RDWR);
if( dev_handle == -1 )
    puts("CAENET.SYS not found");
```

Afterwards the user program must use the DOS IOCTL standard protocol. It is activated by the DOS call %0x44 (please consult the MSDOS Programmer's Reference Manual to get detailed information about the IOCTL DOS protocol). An example of communication follows:

```
#include <io.h>
#include <dos.h>

#define  CAENET_READ  2
#define  CAENET_WRITE 3

typedef struct cnet_fr
{
 unsigned char far  *addr;      /* A303 Memory Space Address  */
 int                craten;     /* CAENET Module Crate Number */
 int                cmd;        /* CAENET Code sent           */
```

```
 int                dir;        /* Can be READ or WRITE       */
 unsigned char far  *iobuf;     /* I/O Data Buffer            */
 int                buflen;     /* Buffer Length              */
 int                exitus;     /* How things went            */
} CNET_FRAME;

CAENET_FRAME  caenet_frame;

... some initialization of caenet_frame

/* To read some data */
ioctl(dev_handle,CAENET_READ,&caenet_frame,sizeof(caenet_frame));

/* To write some data */
ioctl(dev_handle,CAENET_WRITE,&caenet_frame,sizeof(caenet_frame));
```

The application program must set adequately the various fields of CAENET_FRAME structure (except for the last one), before the IOCTL call. In fact the first 6 fields are inputs to the driver, while the 7-th field is an output from it. In particular, the exitus field can take the following values:

0	Operation Successfully Completed
1	Wrong A303 Memory Space Address
2	Transmit Timeout
3	No Slave Module Response
4	Less Data Transferred

Codes 1 to 4 correspond to malfunctionings detected by the A303 itself. A negative value corresponds instead to an error code signalled by the Slave module with which the last CAENET operation has taken place.

Possible settings of the CAENET_FRAME structure follow:

addr	Default Factory Setting is 0xD0010000L
craten	A value between 1 and 99
cmd	CAENET Command, Slave Module dependent
dir	if = 0 the following is a read operation, if = 1 the following is a write operation
iobuf	buffer address in the form: Segment:Offset. It can be NULL if the CAENET Command is not followed by data
buflen	buffer length in bytes. It can also be 0, in the case as above. When cmd implies setting of a value, it corresponds to the sizeof(variable which contains the value)
exitus	set by driver. See previous paragraph

For those compilers that do not have an ioctl routine, the DOS Call in explicit form follows:

```
AX = 4402h (Read), 4403h (Write)
BX = dev_handle
CX = sizeof(CAENET_FRAME)
DX = offsetof(CAENET_FRAME)
DS = segmentof(CAENET_FRAME)

CALL MSDOS
```

4.3.3. EXAMPLE

In the following we include the code of a C wrapper for the driver.

```
/************************************************************************/
/*                                                                      */
/*            ------    C. A. E. N.   S.p.A.    ------                  */
/*                                                                      */
/*   CAENDRV.C                                                          */
/*                                                                      */
/*                                                                      */
/*   Aut: C.Raffo                                                       */
/*                                                                      */
/************************************************************************/
#include <io.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>

typedef struct cnet_fr
{
 unsigned char far  *addr;
 int                craten;
 int                cmd;
 int                dir;
 unsigned char far  *iobuf;
 int                buflen;
 int                exitus;
} CNET_FRAME;

#define   CWRITE            1
#define   CREAD             0
#define   IM_A_DEVICE  0x0080
#define   IOCTL_OK     0x4000
#define   YES               1
#define   NO                0

static int         dev_handle;
static CNET_FRAME  caenet_frame;

extern unsigned char far *address;
extern int               cratenum,code;

/*
  --- CAENET_READ ---
*/
int caenet_read(unsigned char *dest, int byte_count)
{
caenet_frame.addr=address;
caenet_frame.craten=cratenum;
caenet_frame.cmd=code;
caenet_frame.dir=CREAD;
```

```
caenet_frame.iobuf=MK_FP(_DS,dest);
caenet_frame.buflen=byte_count;
ioctl(dev_handle,2,&caenet_frame,sizeof(caenet_frame));
return caenet_frame.exitus;
}

/*
  --- CAENET_WRITE ---
*/
int caenet_write(unsigned char *source, int byte_count)
{
caenet_frame.addr=address;
caenet_frame.craten=cratenum;
caenet_frame.cmd=code;
caenet_frame.dir=CWRITE;
caenet_frame.iobuf=MK_FP(_DS,source);
caenet_frame.buflen=byte_count;
ioctl(dev_handle,3,&caenet_frame,sizeof(caenet_frame));
return caenet_frame.exitus;
}

/*
  --- DRV_INSTALLED ---
*/
int drv_installed(void)
{
size_t  d_info;

dev_handle=open("CAENETXX",O_RDWR);

if(dev_handle == -1)
   return NO;

d_info=ioctl(dev_handle,0);

if((d_info & IM_A_DEVICE) == 0 || (d_info & IOCTL_OK) == 0)
   return NO;

return YES;
}
```

5. ACKNOWLEDGEMENTS

The present document has been produced with the support of the following people:

Giovanni Grieco / CAEN S.p.A.	Marketing Division
Claudio Landi / CAEN S.p.A.	R&D Division
Maurizio Lippi / CAEN S.p.A.	R&D Division
Claudio Raffo / CAEN S.p.A.	R&D Division

For all informations and requests, our contact address is:

C.A.E.N. S.p.A. Costruzioni Apparecchiature Elettroniche Nucleari

Via Vetraia, 11
55049 Viareggio (LU), ITALY

Tel. (0584) 388398 Fax (0584) 388959
E-Mail Address: CAEN@VM.CNUCE.CNR.IT

APPENDIX A: ELECTRICAL DIAGRAMS

APPENDIX B: COMPONENT LIST AND LOCATIONS