Fermi National Accelerator Laboratory

TM-1484

Online Monitoring of Laserpulses Using the GPIB-Interface of a Tektronix 2430 Digital Storage Oscilloscope

Ludo Verluyten
Michael W. Peters
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

October 1987

Operated by Universities Research Association Inc. under contract with the United States Department of Energy

TM-1484

October 1987

ONLINE MONITORING OF LASERPULSES USING THE GPIB-INTERFACE OF A TEKTRONIX 2430 DIGITAL STORAGE OSCILLOSCOPE

Ludo Verluyten

Michael W Peters

One of the monitoringaspects of holography at the 15 foot Bubble Chamber is the online checking of the lightoutput of the JK2000 ruby laser. Ideally the lightoutput should be a squared pulse in time. Spikes, pre- and postlasing (i.e. lasing before and after the squared pulse) can have a negative influence on the holograms.

The scheme for monitoring the lightoutput is shown in fig.1. A EG&G FND-100 photodiode, whose output is stored and digitized by a Tektronix 2430 Digital Storage Oscilloscope, has been placed at the outputstage of the laser. The size of the storage memory is 1 kB. The scope has a horizontal resolution of 50 samples per division and a vertical resolution of 25 digitization levels per division. The scope is externally triggered by a TTL pulse, which opens the Pockel Cell in the oscillator so that lasing can occur. By using a General Purpose Interface Bus (GPIB) an IBM-PC/AT retrieves data from the scope at transferspeeds over 300 kB per second. The GPIB assembly, which is essentially a bidirectional internal data bus, consists of a GPIB-PC II board and a double RF-shielded GPIB extensioncable (both items were purchased from National Instruments). The two devices connected on the GPIB are the IBM-PC and the scope, which has the logical name "dev1" and bus address 1. The PC is the Controller of the bus. It can address the scope to talk (i.e. request it to send data) or to listen. If the scope is in a listen mode, the controller is able to change the settings of the scope over the bus. The GPIB hardware and software setup for the board and the scope are shown in appendix A. The EOI interface line is asserted to mark the end of a message string. The data from the scope are retrieved from channel 1 and are sent in positive integer notation. The program scoperead.c (appendix B) reads the data from the scope. It consists of 4 major parts:

- the main() function to initialize the scope and to print the final results to the file summary.dat located on a fASTdisk D:. This is a software disk residing in extended memory (above 1 MB) to which data can be written or read from at RAM speed.

- the sc_read() function to read the actual curve data from the scope and to store them in a buffer.
- the sc_flt_read() function to read the scope settings.
- the sc_calc() function to calculate the features of the laserpulse.

All GPIB functions, used in the program, address directly the scope and are called device functions. The scope is put in the talk and listen mode. Most of the issued commands in the program only require the talk capability of the scope. The only command which requires the listen capability is "PATH OFF". After each call to the scope the statusword of the bus gets checked for a GPIB error or for a SRQ (service request) issued by the scope. Whenever the scope is improperly addressed, it will assert the RQS bit (requesting service) in the GPIB statusword. To locate the problem, one has to check the statusbyte of the scope by serially polling the device. Checking for scope errors is only useful when the following bits are set on the scope:

- CER (command error)
- EXR (execution error)
- EXW (execution warning)
- INR (internal error)
- DEVDEP (device dependent error)

Describing the characteristics of the lightoutput is essentially a pattern recognition problem. A pulse is defined as each signal that is 8 digitization levels below the bias level (the fast photodiode gives a negative signal output). The main pulse is the pulse with the biggest width. A spike is being defined as each signal in the main pulse larger than two times the average pulseheight. Appendix C shows an erratic laserpulse together with the output from the program scoperead.c. The program, which takes less than 2 seconds to run, describes the features of the laserpulse in a satisfactory way.

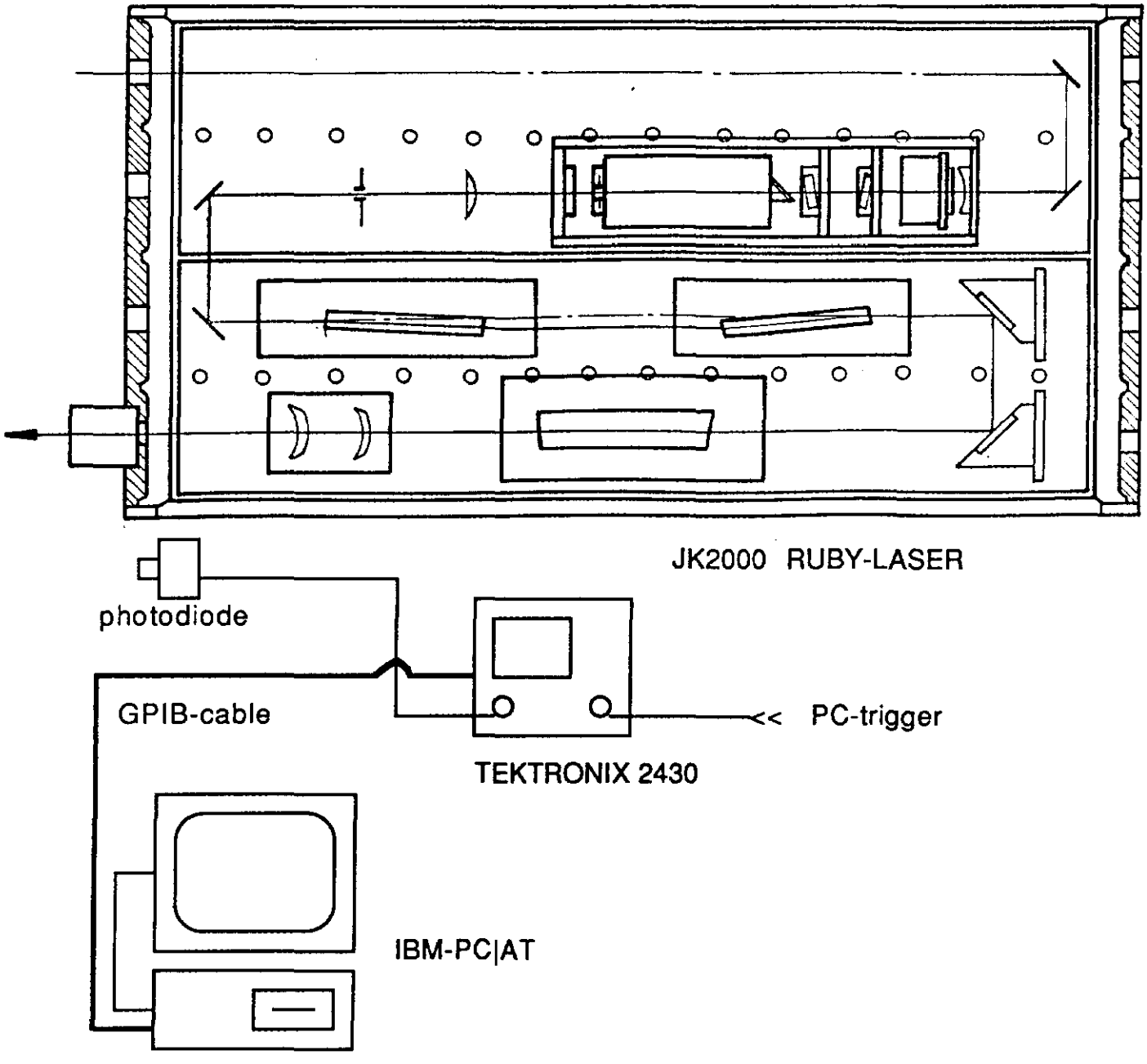
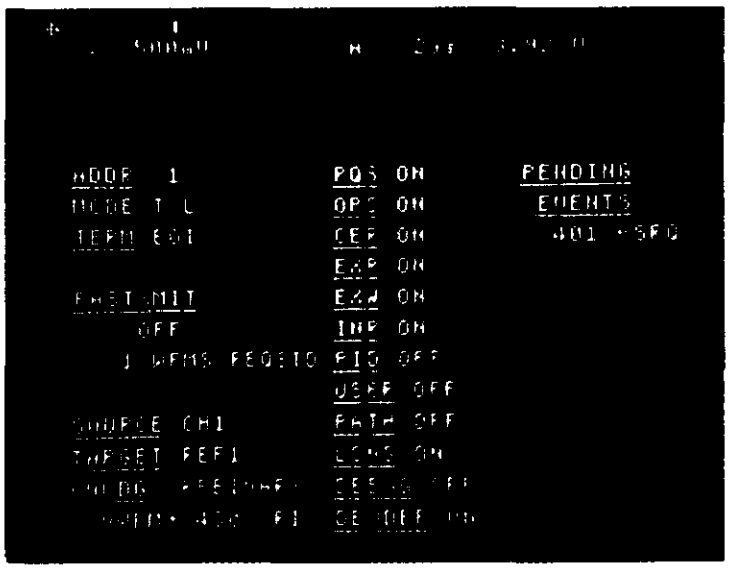


Fig.1: Experimental setup for online monitoring of laserpulses.

Appendix A

1) Scope configuration.



Device: DEV1	Access: GPIB0
Primary GPIB Address	1
Secondary GPIB Address	None
Timeout setting	T10s
EOS byte	00H
Terminate Read on EOS	no
Set EOI with EOS on Write	no
Type of compare on EOS	7-bit
Set EOI w/last byte of Write	yes

2) Hardware and software configuration for the GPIB-PC board.

Board: GPIB0	
Primary GPIB Address	0
Secondary GPIB Address	None
Timeout setting	T10s
EOS byte	00H

Terminate Read on EOS	no
Set EOI with EOS on Write	no
Type of compare on EOS	7-bit
Set EOI w/last byte of Write	yes
GPIB-PC model	PC2
Board is System Controller	yes
Local Lockout on all devices	yes
Disable Auto Serial Polling	no
High-speed timing	yes
Interrupt jumper setting	7
Base I/O Address	2B8H
DMA channel	1
Internal Clock Freq (in MHz)	8

Appendix B

```
/* ****************** Program Scoperead ****************       */
/* This program reads out a Tektronix 2430 digital oscilloscope */
/* by means of a GPIB-PC II board (purchased from National      */
/* Instruments), plugged into one of the slots of an IBM-PC/AT. */
/* ibsta is the statusword of the GPIB.                         */
/* iberr is the errorcode for the GPIB error.                   */
/* ibfind(), ibwrt(,,), ibrd(,,), ibloc() and ibrsp(,) are      */
/* functions supplied with the purchase of the GPIB-PC board.   */
#include <ctype.h>
#include <string.h>
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
#include <stdlib.h>
#include <stdio.h>
#include <v2tov3.h>
extern int ibsta,iberr,ibcnt;
static int itrig,scope;
static double sc_time,sc_volts,sc_trig;
static unsigned char value[1024],spr;
static int index1,pul_width,pul_av,pul_start,pul_index;
static int amp[20],sp_amp[20],time[20];
static float sp_time[20],spwidth[20];
static int index2;
FILE *sum;

main(){
   int i;
   if( (sum=fopen("D:\summary.dat","w")) == NULL ){
     puts("CANNOT OPEN FILE\n");
     exit();
   }
   if( (scope=ibfind("dev1"))<0 ){
     printf("CANNOT OPEN THE SCOPE\n");
     exit();
```

```
    }
    ibwrt(scope,"path off",8);
    if(ibsta<0){
      printf("GPIB ERROR IN WRITING: path off\n");
      printf("STATUSWORD: %6d\n",ibsta);
      printf("ERRORCODE:  %6d\n",iberr);
      ibloc(scope);
      exit();
    }
    if( (ibsta & (1 << 11)) == 0x800 ){
      printf("SCOPE ERROR IN WRITING: path off\n");
      ibrsp(scope,&spr);
      printf("SCOPE STATUSWORD: %6d\n",spr);
      ibloc(scope);
      exit();
    }
    sc_read();
    for(i=0;i<20;i++){
      amp[i]=0;
      time[i]=0;
      sp_amp[i]=0;
      sp_time[i]=0.0;
      spwidth[i]=0.0;
    }
    sc_calc();
    fprintf(sum,"   MAIN PULSE INFORMATION\r\n");
    fprintf(sum,"   ----------------------\r\n");
    fprintf(sum,"   time between pulse and ext trig (us):");
    fprintf(sum," %2f\r\n",sc_time*(pul_start-itrig));
    fprintf(sum,"   pulsewidth (us): %2f\r\n",sc_time*pul_width);
    fprintf(sum,"   average pulseheight (mV): %2f\r\n\n",sc_volts*
pul_av);
    if(index2 != 0){
      fprintf(sum,"   SPIKES IN THE PULSE:\r\n");
      fprintf(sum,"   start (us)     width (us)   max amplitude (mV)
\r\n");
      for(i=0;i<index2;i++){
```

```
      fprintf(sum,"    %2f        %2f",sp_time[i],spwidth[i]);
      fprintf(sum,"      %2f\r\n",sp_amp[i]*sc_volts);
      }
   }
   fprintf(sum,"\n\n");
   if( index1 > 1 ){
     fprintf(sum,"   PRE- AND/OR POSTLASING\r\n");
     fprintf(sum,"   ----------------------\r\n");
     fprintf(sum,"   time (us)      max amplitude (mV)\r\n");
     for(i=0;i<index1;i++){
        if(i != pul_index)
         fprintf(sum,"   %2f         %2f\r\n",time[i]*sc_time,
amp[i]*sc_volts);
     }
   }
} /* end main */

sc_read(){
   /* This function reads data from the scope. The first argument*/
   /* in the function sc_flt_read() is part of the scope command-*/
   /* language.                                                  */
   double val;
   /* read out the external triggerposition */
   if(sc_flt_read("atr?pos",2,&val)<0) ibloc(scope);
   itrig=32*val;
   /* read out the time scale of the scope */
   if(sc_flt_read("hor?ase",4,&sc_time)<0) ibloc(scope);
   /* read out the voltage scale of the scope */
   if(sc_flt_read("ch1?vol",4,&sc_volts)<0) ibloc(scope);
   sc_time=sc_time*1e+6/50.0;
   sc_volts=sc_volts*1e+3/25.0;
   sc_trig=itrig*sc_time;
   ibwrt(scope,"curv?",5);
   if(ibsta<0){
     printf("GPIB ERROR IN WRITING: curv?\n");
     printf("STATUSWORD: %6d\n",ibsta);
     printf("ERRORCODE: %6d\n",iberr);
```

```
  ibloc(scope);
  exit();
}
if( (ibsta & (1 << 11)) == 0x800 ){
  printf("SCOPE ERROR IN WRITING: curv?\n");
  goto scoperror;
}
ibrd(scope,value,3);   /* return ignored */
if(ibsta<0){
  printf("GPIB ERROR IN READING FIRST 3 BYTES FROM CURVE\n");
  printf("STATUSWORD: %6d\n",ibsta);
  printf("ERRORCODE: %6d\n",iberr);
  ibloc(scope);
  exit();
}
if( (ibsta & (1 << 11)) == 0x800 ){
  printf("SCOPE ERROR IN READING FIRST 3 BYTES FROM CURVE\n");
  goto scoperror;
}
ibrd(scope,value,1024);
if(ibsta<0){
  printf("GPIB ERROR IN READING CURVE DATA\n");
  printf("STATUSWORD: %6d\n",ibsta);
  printf("ERRORCODE: %6d\n",iberr);
  ibloc(scope);
  exit();
}
if( ( ibsta & (1 << 11)) == 0x800 ){
  printf("SCOPE ERROR IN READING CURVE DATA\n");
  goto scoperror;
}
ibloc(scope);
return;
scoperror:
  ibrsp(scope,&spr);
  printf("SCOPE STATUSWORD: %6d\n",spr);
  ibloc(scope);
```

```
      exit();
} /* end sc_read */

sc_flt_read(sc_cmd,nread,result)
   char *sc_cmd[];
   int nread;
   double *result;
{
   /* This function does the actual readout of the scope. */
   char buf[80];
   ibwrt(scope,sc_cmd,strlen(sc_cmd));
   if(ibsta<0){
     printf("GPIB ERROR IN WRITING DATA MESSAGE: %7s\n",sc_cmd);
     printf("STATUSWORD: %6d\n",ibsta);
     printf("ERRORCODE: %6d\n",iberr);
     ibloc(scope);
     exit();
   }
   if( (ibsta & (1 << 11)) == 0x800){
     printf("SCOPE ERROR IN WRITING DATA MESSAGE: %7s\n",sc_cmd);
     goto scoperror;
   }
   ibrd(scope,buf,nread);
   if(ibsta<0){
     printf("GPIB ERROR IN READING DATA FROM DATA MESSAGE: %7s\n",
sc_cmd);
     printf("STATUSWORD: %6d\n",ibsta);
     printf("ERRORCODE: %6d\n",iberr);
     ibloc(scope);
     exit();
   }
   if( (ibsta & (1 << 11)) == 0x800){
     printf("SCOPE ERROR IN READING DATA FROM DATA MESSAGE: %7s\n",
sc_cmd);
     goto scoperror;
   }
   buf[nread]='\0';
```

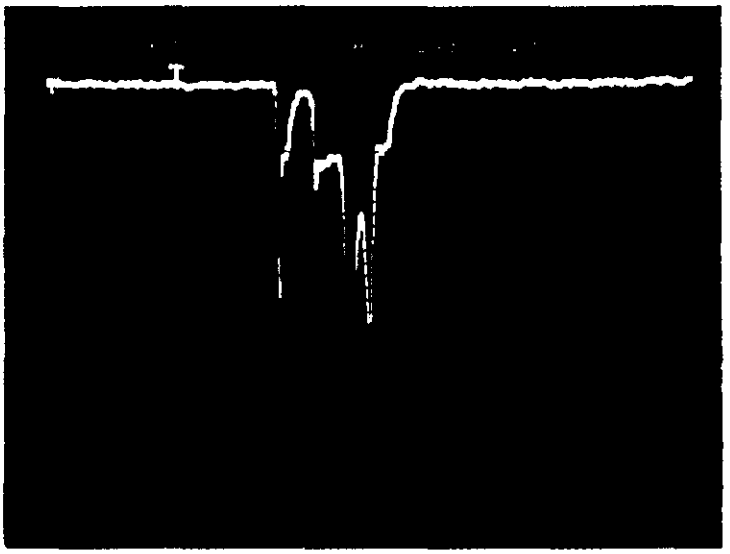
```
   *result=atof(&buf[0]);
   return;
   scoperror:
     ibrsp(scope,&spr);
     printf("SCOPE STATUSWORD: %6d\n",spr);
     ibloc(scope);
     exit();
} /* end of sc_flt_read */

sc_calc(){
   /* This function calculates:                                    */
   /*      - the width of the laserpulse,                          */
   /*      - the time the laserpulse starts after the Pockel Cell*/
   /*        opening,                                              */
   /*      - the average amplitude of the laserpulse,              */
   /*      - the width and maximum amplitude of pre- and           */
   /*        postlasing,                                           */
   /*      - the time, width and maximum amplitude of any          */
   /*        spikes in the pulse.                                  */
   int i,j;
   int average,work,workbef,thresh;
   int width,start,stop,pul_stop;
   int sp_width,sp_start;
   average=0;
   pul_width=0;
   for(i=0;i<10;i++) average+=value[i];
   average/=10;
   thresh=8;
   workbef=0;index1=0;
   for(i=0;i<1023;i++){
        work=average-value[i];
        if(work>=thresh){
          if(workbef>=thresh){
            width++;
            amp[index1]=max(amp[index1],work);
          }
          else{
```

```
          start=i;
          width=0;
          amp[index1]=0;
        }
      }
      else if(workbef>=thresh){
        time[index1]=start+width/2-itrig;
        if( (width>=0) && (width<6) ){
          time[index1]=0;
          index1--;
          }
        if(width>pul_width){
          pul_width=width;
          pul_start=start;
          pul_stop=start+width;
          pul_index=index1;
        }
        index1++;
      }
      workbef=work;
}
if( (index1==0) || (pul_width==0) ) return;
pul_av=0.;
for(i=pul_start;i<=pul_stop;i++) pul_av+=average-value[i];
pul_av/=pul_width;
thresh=2*pul_av;
workbef=0;index2=0;
for(i=pul_start;i<=pul_stop;i++){
      work=average-value[i];
      if(work>=thresh){
         if(workbef>=thresh){
           sp_width++;
           sp_amp[index2]=max(sp_amp[index2],work);
         }
         else{
           sp_start=i;
           sp_width=0;
```

```
            sp_amp[index2]=0;
          }
        }
        else if( (workbef>=thresh) && (sp_width > 0) ){
          sp_time[index2]=(sp_start-pul_start)*sc_time;
          spwidth[index2]=sp_width*sc_time;
          index2++;
        }
        workbef=work;
    }
    return;
} /* end sc_calc */
```

Appendix C



MAIN PULSE INFORMATION

---

time between pulse and ext trig (us): 4.320000

pulsewidth (us): 2.600000

average pulseheight (mV): 1060.000000

SPIKES IN THE PULSE:

start (us)	width (us)	max amplitude (mV)
1.120000	0.200000	4060.000000

PRE- AND/OR POSTLASING

---

time (us)	max amplitude (mV)
3.440000	1720.000000