VIRGO

INTERNAL MEMORANDUM

From:	Henrich Heitmann, Laser-Optics Group Orsay
To:	B. Mours, Annecy
CC:	N. Man, E. Majorana
Date: 3.5.1998	**Ref:** VIR-MEM-LAS-4100-111

Requirements to be met by a new version of GalaXie

If a new version of GalaXie is released, it should meet the following requirements in order to make it possible to use it at Orsay. The following list results from observations (required improvements and bugs) made during the use of version 7.0 of GalaXie from October 1996 up to now. The version 7.0 has been changed somewhat in Orsay to respond to our requirements. As far as these changes are necessary for being able to work with a future version of GalaXie, they are reported among the "desired changes". A detailed documentation of changes made in Orsay is given in the annex.

<u>Degrees of importance:</u>

[4]	absolutely necessary for use with present configuration (prototype)
[3]	absolutely necessary for use in VIRGO
[2]	strongly desirable
[1]	would be convenient

Desired changes

- include communication with Orsay "gate timing" card. [4]
- include "weighted average" routine used presently in the Orsay version of GalaXie (see annex) [4]
- "reference position" must set the reference position for fine *and* coarse simultaneously, independently of the present status (fine or coarse) [4].
- enable *fast* switching from coarse to fine and vice versa from a second client (e.g. GXlittleClient), without changing the whole configuration; requires small additional message handler in GXserver (see annex for present solution). [4]
- it must be possible to subtract a constant, user-definable pedestal L0 from the luminosity values L read in the zones, such that the value used for the calculations L' is
```
   L' = L-L0 (if L> L0)
   L' = 0    (if L<=L0)
```
  with L=read luminosity, and L0 = user-defined pedestal (e.g. 100 ADC counts). Effect: simple subtraction of pedestal with setting to zero of resulting negative values. This corresponds to the analog pedestal potentiometer in the interface of the

old (EG&G MC9000) camera. Reason: substantial reduction of noise contribution from areas within the zone not contributing to the light spot, which, by the negative pedestal, are set to 0. Allows using large zones without increasing the noise. (feature necessary for use of new camera version) [3]

- Integrate third laser beam (fifth zone) into the fine mode, which will serve for precision measurement of θ_Z. Formulae need to be adapted []. So: *coarse* measurement works with four reference marks, *fine* with two ref. marks and three lasers. [3]
- GXmain: possibility of selecting zones for fine, even when GX is in coarse mode (reason: need to select fine zones while servo is active in coarse mode). This does not mean that the fine zones have to be *read out* in coarse mode (this would take too much CPU time). [3]
- Enable "refresh image", even when 'servo loop' is active (at present: blocks GX). [3]
- optimize as far as possible execution / reading time (CPU/VME), in order to permit bigger zones to be treated (especially for the indirect beam). [2]
- smooth transition between coarse and fine: during a certain time (1 or 2 seconds), the values of x,y,... written into shared memory should be averages between fine and coarse, with shifting weight a(t) :

  value = (a*fine + (1-a)*coarse ...). [1]
- possibility of exiting GXserver by command from the client, or in some other simple way (without `kill -2`). [1]

Errors

- GXserver, `interrupt_handler`: `4+(j-7)` must be replaced by `6+(j-7)`. (see Annex)
- there was a "=" instead of a "+=" in GXSVreadPro, causing some zones not to be read properly. Maybe already repaired, but must be verified!

  ```
  wrong: imagex[ii] = data[0];
         imagey[i]  = data[0];
  right:        imagex[ii]+= data[0];
         imagey[i] += data[0];
  ```
  [4]
- The variable `beam[]` must be defined wherever `GXchoice. ...` is defined (variables have the same function, but sometimes one is used, sometimes the other => creates problems). `beam[0] = GXchoice.selectDirect` etc. [4]
- GX stops (doesn't respond to frame interrupts anymore) after a period of 30 minutes to several hours, when 'servo loop' is on. Must be reactivated by pressing "reference image". (however, comes probably from Orsay timing card or driver) [3]
- It is not possible to define negative angles for the camera view angle. Instead -30 degrees, I use 330 degrees, but upon saving the configuration, GXmain sometimes saves 330 degrees, sometimes +30 degs instead. [3]
- "refresh image" blocks GALAXIE in 'alignment' mode, if 'servo loop' is on. (too much time consumption in CPU) [3]
- "stop server loop" function does not work (important, since otherwise the CPU is overcharged when trying to do e.g. refresh image) [3]
- GXserver cannot be killed with Ctrl-C. One must use kill -2, otherwise Xwindow-session of VME is blocked [2].
- I don't believe in the correct functioning of the presently installed 'pedestal

subtraction' feature in the GXserver: Due to ambiguously defined variables, zero is always subtracted. See email exchange with Christine Drezen. Remove (speed!) or improve. (Don't confuse with the similar feature requested for noise improvement: function is not the same) [2]

References

[1] H. Heitmann, Mode cleaner local control with increased θ_z sensitivity, internal note Orsay (3.5.98).

Annex

Changes made in the Orsay version of GalaXie v7.0

This annex results from a comparison (`diff`) between files

	`GXserver.c.orig`	original (?) version 7.0
	`GXserverGaTi.c`	operative version used presently at Orsay
Directory:	`virgo_vme5::/users/home/drezen/GX/v7.0/src/`	

The result was simplified by removing unimportant changes.
Caution: Line numbers may not always be correct.

```
******************************************************************************
***   Added Orsay MTiming interrupt
******************************************************************************

94,95d71
< #include <MTiming.h>
<
596,598d548
< #ifdef GATIMING
<  TiValidateIRQ(MODULE2,OUINONNONNON); /* IRQ sur Trig 1 MD le 24/03/97 */
< #endif
1425,1431c1331,1335
< /*
< Modif 21/05/97 MD
< apres calcul
< #ifdef GATIMING
<  TiValidateIRQ(MODULE2,OUINONNONNON);  IRQ sur Trig 1 MD le 24/03/97
< #endif
< */
---
>  /*ftime(&now);
>  strftime(buf1,64,"%H:%M:%S",localtime(&now.time));
>  sprintf(buf2,".%d",now.millitm);
>  strcat(buf1,buf2);
>  printf("received interrupt from timming board at: %s\n",buf1); */
1673,1680c1583
<    sem_signal(semid);
< /*
< Modif 21/05/97 MD
< Deplacement validation IRQ apres calcul position
< */
< #ifdef GATIMING
<  TiValidateIRQ(MODULE2,OUINONNONNON); /* IRQ sur Trig 1 MD le 24/03/97 */
< #endif
---
>    sem_signal(semid);



******************************************************************************
***   Added handler SetFine for fast & simple changing fine /coarse
******************************************************************************

132d106
< CmMessageStatus handlerSetFine();
269d232
<   CmMessageInstallHandler(handlerSetFine,"SetFine");
2055,2071d1949
<
< /*-----------------------------------------------------------------------*/
< CmMessageStatus handlerSetFine(message,sender)
< CmMessage message;
< char *sender;
< /* modif par HH, 14.11.97: added this function */
< {
<   int    k;
<
<   k = CmMessageGetInt(message);
<   GXchoice.fine   = k?1:0;
<   GXchoice.coarse = k?0:1;
<
<   return(CmMessageOk);
< }
<
```

```
******************************************************************************
***   Simplified GXreadPro, removed inoperative pedestal subtraction (speed)
******************************************************************************
```

```
1187,1197c1122,1144
<   i1 = 0;
<   offset = 0;
---
>   i1 = (GXzoom[0].pt2x-GXzoom[0].pt1x+1)*(GXzoom->pt1y-GXzoom[0].pt1y)
>        + (GXzoom->pt1x-GXzoom[0].pt1x);
>   offset = (GXzoom[0].pt2x-GXzoom->pt2x-1) + (GXzoom->pt1x-GXzoom[0].pt1x+1);
1251a1190
>      i1=  i1+offset;
.......1195,1200
>   ftime(&tps);
>   strftime(buf1,128,"%H:%M:%S:",localtime(&tps.time));
>   sprintf(buf2,"%d",tps.millitm);
>   strcat(buf1,buf2);
>  /* printf("reading signal at : %s\n",buf1);*/
>   strcpy(tempsLecture,buf1);
```

```
******************************************************************************
***   Create alignment info structure only if message to be transferred (speed)
******************************************************************************
```

```
1376,1406d1312
< /* ------------------------------------------------------------------------*/
< void InfoClientInit(CmMessage *p_answer)
< /* Prepare "answer" for Cm transfer, using global variables GX... */
< {
<     CmMessage answer;
<     *p_answer = CmMessageNew();
<     answer = *p_answer;
<     CmMessageSetType(answer,"position");
<     CmMessagePutInt(answer,fitflag);
<     CmMessagePutShort(answer,GXchoice.fine);
<     CmMessagePutDouble(answer,GXalignment.Dmeanx);
<     CmMessagePutDouble(answer,GXalignment.Dmeany);
<     CmMessagePutDouble(answer,GXalignment.Fmeanx);
<     CmMessagePutDouble(answer,GXalignment.Fmeany);
<     CmMessagePutDouble(answer,GXalignment.R1meanx);
<     CmMessagePutDouble(answer,GXalignment.R1meany);
<     CmMessagePutDouble(answer,GXalignment.R2meanx);
<     CmMessagePutDouble(answer,GXalignment.R2meany);
<     CmMessagePutDouble(answer,GXalignment.R3meanx);
<     CmMessagePutDouble(answer,GXalignment.R3meany);
<     CmMessagePutDouble(answer,GXalignment.R4meanx);
<     CmMessagePutDouble(answer,GXalignment.R4meany);
<     CmMessagePutDouble(answer,GXalignment.thetaX);
<     CmMessagePutDouble(answer,GXalignment.thetaY);
<     CmMessagePutDouble(answer,GXalignment.thetaZ);
<     CmMessagePutDouble(answer,GXalignment.x);
<     CmMessagePutDouble(answer,GXalignment.y);
<     CmMessagePutDouble(answer,GXalignment.z);
< }
<
```

```
1412c1318
< int   size, i,j,j1,j2,l=1,deltaX,deltaY,colpix,lipix,inited;
---
> int   size, i,j,j1,j2,l=1,deltaX,deltaY,colpix,lipix;
1616a1521,1543
>
>     answer = CmMessageNew();
>     CmMessageSetType(answer,"position");
>     CmMessagePutInt(answer,fitflag);
>     CmMessagePutShort(answer,GXchoice.fine);
>     CmMessagePutDouble(answer,GXalignment.Dmeanx);
>     CmMessagePutDouble(answer,GXalignment.Dmeany);
>     CmMessagePutDouble(answer,GXalignment.Fmeanx);
>     CmMessagePutDouble(answer,GXalignment.Fmeany);
>     CmMessagePutDouble(answer,GXalignment.R1meanx);
>     CmMessagePutDouble(answer,GXalignment.R1meany);
>     CmMessagePutDouble(answer,GXalignment.R2meanx);
>     CmMessagePutDouble(answer,GXalignment.R2meany);
>     CmMessagePutDouble(answer,GXalignment.R3meanx);
>     CmMessagePutDouble(answer,GXalignment.R3meany);
>     CmMessagePutDouble(answer,GXalignment.R4meanx);
>     CmMessagePutDouble(answer,GXalignment.R4meany);
>     CmMessagePutDouble(answer,GXalignment.thetaX);
>     CmMessagePutDouble(answer,GXalignment.thetaY);
>     CmMessagePutDouble(answer,GXalignment.thetaZ);
>     CmMessagePutDouble(answer,GXalignment.x);
>     CmMessagePutDouble(answer,GXalignment.y);
>     CmMessagePutDouble(answer,GXalignment.z);
1683d1585
< inited = 0;
1693d1594
<                 if(!inited) {InfoClientInit(&answer);inited = 1;}
1706c1607
<          if ( !(ind%5) && (infoClient->slowTransfer == 0) && (infoClient->align
== 1))
---
>          if ( (infoClient->slowTransfer == 0) && (infoClient->align == 1))
1711d1611
<               if(!inited) {InfoClientInit(&answer);inited = 1;}
```

```
******************************************************************************
***   Repaired bug 4+(j-7)   ->  6+(j-7) in interrupt_handler
******************************************************************************

1483,1485d1389
<                     /* dernier param.sert pour stockage des valeurs dans un
<                        tableau interne. 6+(j-7) assure que x sont stockes dans
<                        0...5, et y dans 6...11 : */  /* modif.HH */
1487c1391
<                                     GXzoom[j].pt1y,&GXfitY,6+(j-7)) ;
---
>                                     GXzoom[j].pt1y,&GXfitY,4+(j-7)) ;
```

```
******************************************************************************
***   Enable Fine and Coarse reference position to be read simultaneously
******************************************************************************

2076,2077d1953
< /* modif par HH, 31.1.97: lire zones "fine" et "coarse",
<    et determiner pos.ref. pour les deux a la fois */
2079c1955
<   int    size,j,k,fine_sav;
---
>   int    size,j,k;
2080a1957
>   int    xCP[NPIX],yCP[NPIX];
```

```
2092,2112c1968,1989
<    CmConnect dummy;
<    /* modif HH: if fine, read coarse points and vice versa: */
<    fine_sav = GXchoice.fine;
<    GXchoice.fine   = fine_sav?0:1;
<    GXchoice.coarse = fine_sav?1:0;
<    interrupt_handler(dummy);
<    GXchoice.fine   = fine_sav?1:0;
<    GXchoice.coarse = fine_sav?0:1;
<    /* ----- */
<    GXalignment.DmeanxOld = GXalignment.Dmeanx;
<    GXalignment.DmeanyOld = GXalignment.Dmeany;
<    GXalignment.FmeanxOld = GXalignment.Fmeanx;
<    GXalignment.FmeanyOld = GXalignment.Fmeany;
<    GXalignment.R1meanxOld = GXalignment.R1meanx;
<    GXalignment.R1meanyOld = GXalignment.R1meany;
<    GXalignment.R2meanxOld = GXalignment.R2meanx;
<    GXalignment.R2meanyOld = GXalignment.R2meany;
<    GXalignment.R3meanxOld = GXalignment.R3meanx;
<    GXalignment.R3meanyOld = GXalignment.R3meany;
<    GXalignment.R4meanxOld = GXalignment.R4meanx;
<    GXalignment.R4meanyOld = GXalignment.R4meany;
---
>   if (GXchoice.fine == 1)
>       {
>       GXalignment.DmeanxOld = GXalignment.Dmeanx;
>       GXalignment.DmeanyOld = GXalignment.Dmeany;
>       GXalignment.FmeanxOld = GXalignment.Fmeanx;
>       GXalignment.FmeanyOld = GXalignment.Fmeany;
>       GXalignment.R1meanxOld = GXalignment.R1meanx;
>       GXalignment.R1meanyOld = GXalignment.R1meany;
>       GXalignment.R2meanxOld = GXalignment.R2meanx;
>       GXalignment.R2meanyOld = GXalignment.R2meany;
>       }
>   else
>       {
>       GXalignment.R1meanxOld = GXalignment.R1meanx;
>       GXalignment.R1meanyOld = GXalignment.R1meany;
>       GXalignment.R2meanxOld = GXalignment.R2meanx;
>       GXalignment.R2meanyOld = GXalignment.R2meany;
>       GXalignment.R3meanxOld = GXalignment.R3meanx;
>       GXalignment.R3meanyOld = GXalignment.R3meany;
>       GXalignment.R4meanxOld = GXalignment.R4meanx;
>       GXalignment.R4meanyOld = GXalignment.R4meany;
>       }
```

```
******************************************************************************
***   Added definition of beam[] wherever GXchoice. ... is defined
******************************************************************************

2410c2296
<      beam[0] = GXchoice.selectDirect = n; /* modif. HH 3.12.97 */
---
>      GXchoice.selectDirect = n;
2412c2298
<      beam[1] = GXchoice.selectFolded = n; /* modif. HH 3.12.97 */
---
>      GXchoice.selectFolded = n;
2414c2300
<      beam[2] = GXchoice.selectRef1 = n; /* modif. HH 3.12.97 */
---
>      GXchoice.selectRef1 = n;
2416c2302
<      beam[3] = GXchoice.selectRef2 = n; /* modif. HH 3.12.97 */
---
>      GXchoice.selectRef2 = n;
2418c2304
<      beam[4] = GXchoice.selectRef3 = n; /* modif. HH 3.12.97 */
---
>      GXchoice.selectRef3 = n;
2420c2306
```

```
<      beam[5] = GXchoice.selectRef4 = n; /* modif. HH 3.12.97 */
---
>      GXchoice.selectRef4 = n;
```

GXlibCalcul.c:

```
******************************************************************************
***   This is the "Weighted average" function presently used at Orsay
******************************************************************************

void GXLCcalculAverage(zCP,nbptmax,size,pt1,fit,idata)
struct GXarrayDouble   *zCP;
int                    nbptmax;  /* length of pixel array (256) */
double                 size;     /* pixel size (40um) */
int                    pt1;      /* beginning of zone (px.number) */
int                    idata;    /* number of data set (not used) */
struct GXarrayDouble   *fit;

#define Ndata 20     /* max. number of data sets to be treated */

/*-- Takes the projection (in X or Y) of the contents of a selected
     CCD zone (zCP->value[i]), and calculates the weighted mean value
     of X or Y (*fit). First and last element (row of pixels) are
     discarded (not reliable?). The min. value (average of two neighboring
     elements, in order to compensate effect of even/odd pixel noise) is taken
     as offset and subtracted from all values. This assures that even for
     pathologic cases the sum of weights is positive.
        If the light spot leaves the zone, then the last valid position is
     returned as result, until the spot enters the zone again (=> stabilizes
     feedbacks with integrators).
     . To an index i corresponds a position (pt1+i-nbptmax/2)*size.
     . Length of zone: zCP->size.--*/

 {
  int     i,imin,imax,ldebug;
  double  u,v,z,zmin,zmax,vmax,sumuv,sumv,offset;

  static int debug=0,idatamax=-1;
  static int ncyc[Ndata]      ={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
  static double PosSave[Ndata]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
                                0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},
                vmaxmax[Ndata]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
                                0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};

  ldebug = (debug%6001==0) && LDEBUG;
  if(idata<0 || idata>Ndata) {
     if(ldebug) printf("Error(GXLCcalculAverage): idata=%d\n",idata);
     idata=Ndata-1;
     }
  imin = 1; imax = zCP->size-2; /* discard first and last element */
  zmin = 3e30; zmax = -3e30;
  for(i=imin;i<=imax;i++)
     {
     z = 0.5*(*(zCP->value+i)+*(zCP->value+i+1));
     if(z<zmin) zmin=z;
     if(z>zmax) zmax=z;
     }
  offset=zmin;
  vmax = zmax-offset;
  if(vmax>vmaxmax[idata])
     { vmaxmax[idata]=vmax; }    /* Max. in zone over several measurements */
  if(idata>idatamax) idatamax = idata; /* max.index actually used (save time) */
  ncyc[idata] = 0;
  for(i=0;i<=idatamax;i++)  {
     /* Reset max, if idata after >> one cycle still unused.
        (i.e. reset fine if changed to coarse etc) */
     if(ncyc[i]>100) vmaxmax[i] = 0.;
     else {
        ncyc[i]++;
        vmaxmax[i] *= 0.99998; /* Slowly adapt max. to changing conditions */
        }
     }
  if(ldebug)
```

```
   { printf("zmin %lf,zmax %lf,vmax %lf,vmaxmax[idata] %lf\n",
             zmin,zmax,vmax,vmaxmax[idata]); }

/* ptr1 from 0 or 1? ######### */
sumuv=sumv=0.0;
if(vmax<0.5*vmaxmax[idata])       /* spot more than 1/2 outside zone */
   {
   fit->value[1]=PosSave[idata];  /* use saved position */
   putchar('.'); fflush(stdout);
   }
else {     /* calculate position from actual data */
   for(i=imin;i<=imax;i++) /* Pix.relative to beginning of zone */
      {
      u = size*(i+pt1-nbptmax/2+0.5); /* pix.pos.(mm);camera center=0*/
      v = *(zCP->value+i)-offset;     /* corrected pixel contents */
      if(ldebug)
        { printf("%3d:%6.3lf:%8.2lf|",i+pt1,u,v); }
      sumuv += u*v;
      sumv  += v;
      }
   fit->value[1]=sumv==0.0 ? -1000.0 : sumuv/sumv;
   PosSave[idata] = fit->value[1];     /* save position */
   }
if(ldebug)
   {
   printf("\nAvg:%9.5lf,sum=%lf.2|%.2lf\n",fit->value[1],sumv,sumuv);
   printf("nbptmax:%d,size:%g,pt1:%d,length:%d,idata:%d\n\n",
           nbptmax,size,pt1,zCP->size,idata);
   }
debug++;
return;
}
```