A SET OF SOFTWARE DRIVERS FOR THE IMAGEN 8/300 PRINTER

A. Balestra, P. Marcucci, R. Smareglia

September 1990

Pubblicazione Osservatorio Astronomico di Trieste n. 1311

Contents

ABSTRACT

Implementations of a driver for the IMAGEN 8/300 laser printer to be integrated in the Astronet Graphics Library (AGL) and of a software interface for alphanumeric printing are described.

1 Introduction

The need of high quality graphic and alphanumeric outputs for scientific purposes guided the authors to build a set of software modules and interfaces in order to exploit all the IMAGEN 8/300 printer capabilities.

The standard scientific outputs can be divided in three main items:

1. alphanumeric outputs (i.e. listings, tables, etc..)
2. graphic outputs (i.e. line drawing)
3. pictorial outputs (i.e. images).

For the first (alphanumeric) a standard output format (typically 80 columns per 66 or 60 rows) has to be choosed, keeping in mind the readability of the printouts and the need to keep more information on a single page as possible.

The second output (graphic) is a more complex object. There is a standard graphic library available in the astronomical community (AGL), that can handle various output devices. IMAGEN 8/300 was not supported, so the only reasonable choice is to build an AGL device driver that supports this printer.

The third type of output (pictorial) is the most difficult to achieve. Laser printers, as the IMAGEN 8/300, are monochromatic devices, while standard astronomical images, generally, are colourful. In the impossibility to print in colors, a convenient way to reproduce the image is to use a suitable gray scale. Such a technique allows the reproduction of a gray scale of n^2 levels using a box of n x n printer pixels for every screen pixel. The number of black dots is directly related to the gray level to print, while the distribution inside the n x n box is provided by a random number generator in order to avoid "artifacts" (unwanted regular patterns within an image).

Image printing is a very device dependent task, and can be achieved via dedicated programs that convert image files, saved with some standard (but unknown) format, in an IMPRESS bitmap. One of these programs is `hcimage`, developed by P. Santin at Trieste Astronomical Observatory. This program is written using the system calls provided by Apollo and so it is not portable to other systems. So, in the impossibility to define a standard for image printing, only two software modules are required: a filter program for ASCII text files and an AGL device driver for graphics.

These two modules produce IMPRESS code suited for the IMAGEN 8/300. IMPRESS is the internal page definition language of this printer and a brief description of it can be found in the "IMPRESS programming manual" [3] shipped with the printer.

The IMPRESS page definition language (PDL) provides a set of commands composed by a stream of bytes where the first byte indicates the command itself, and the following bytes are the

command parameters. The unavailability of an IMPRESS interpreter, and the lacking of documentation, forced the authors to build a suitable program (`ig`) to gather all the informations needed to write a working driver. This program is a little compiler that, starting from a human-readable command list, is able to produce the binary code requested by the image processor. The program can also be used to build IMPRESS data structures in order to perform special tasks as high quality text output, polygon filling and shading.

2 Printer configuration

The configuration selected for the Imagen 8/300 printer is quite standard. The protocol used is a simple serial one, based on asynchronous byte stream communications. This protocol allows the printer to communicate with a host transmitting 7 or 8 bits for a byte.

The connector used is a DB 25 EIA standard for RS 235C. No error checking is provided, while a flow control XON/XOFF is adopted in order to prevent printer buffer overflow. The line speed is setted to 19200 baud, the maximum allowed by the device; no printer emulation is selected. 8 bit data mode, where all 8 bits are taken as is, is enabled.

The EOF (end of file) and the quoting characters are setted to 04 (hex) and 02 (hex) respectively, as suggested by Imagen. The option allowing to ignore any non printing character is not enabled.

The job and document control settings contains informations about the dimensions of the paper sheet, the adopted language, and the operating mode of the printer.

The adopted default format for the paper is normal a4, 3520 pixel in height and 2496 in width (standard Imagen). The language is IMPRESS, which allows great flexibility and the exploitation of special performances. Other types of languages have been tested, (DAISY, PRINTER) but no useful result was achieved. Page reversal is enabled, in order to offer a more user friendly presentation of printed papers.

The command used to set these control parameters takes the form:

```
@document (language IMPRESS, paper a4, pagereversal on)
```

3 Alphanumeric output

Alphanumeric output is achieved via a script and a filter program that provides an interface between ASCII text files and IMPRESS devices (such as the IMAGEN 8/300).

3.1 prcx

The script file (`prcx`) gets a file name from the UNIX command line and generates an unique temporary file name. This file name is composed using the host name and the process identifier (PID), so providing a method to permit multi user printing on the device.

`prcx` has been written using the C shell programming language available under the UNIX BSD 4.3 operating system.

3.2 pressimg

The ASCII file is then passed to the filter program (`pressimg`) that builds an IMPRESS file (named with the temporary file name) and sends it to the `prf` program for the real print operation. The temporary file is then removed.

This program performs some formatting over the input ASCII text file, using a page format of 80 columns per 65 rows. Lines whose length exceed 80 characters are splitted over two or more rows.

A standard IMPRESS header is added at the beginning of the output file in order to bypass the default one programmed in the printer. This header sets the page dimensions (paper a4) and the order by which the pages will be printed (pagereversal on).

`pressimg` has been written in C language on an Apollo workstation running under the UNIX BSD 4.3 operating system, in order to provide a high degree of portability either between UNIX or VAX/VMS machines. A version under SYS V.3 is also available.

3.3 Use

To print an ASCII text file use the command:

`prcx` *filename*

4 The AGL device driver

An AGL device driver permits the programmer to write code regardless of the physical graphic output device. At the Trieste Astronomical Observatory, a series of printing devices are currently supported by AGL (PostScript, HPGL, Versatec, etc..) but a driver for the IMAGEN 8/300 was lacking due to the poor documentation and software support.

The driver has been written in C language on an Apollo workstation running under the UNIX BSD 4.3 operating system, in order to provide a high degree of portability either between UNIX or VAX/VMS machines. A version under SYS V.3 is also available.

4.1 Description of the implementation

The implementation of the AGL device driver was written following the guidelines described in [5]. Using the driver `nulldrv.c` as a "skeleton", the following procedures have been implemented:

AGLINIM0

this routine performs variuos operation depending on the first parameter passed. With the first parameter set to 0, the routine writes on the output file a header needed by the IMPRESS interpreter contained in the IMAGEN image processor, and performs all the state variables initialization (i.e. scaling factors). Two IMPRESS commands (`SET PEN WIDTH` and `SET PATH UPDATE MODE`) are executed in order to provide the width of the plotting pen (3 in our case, choosed for readability reasons) and the path update mode (0 in our case). With the first parameter set to 4, the routine changes the pen width (in a range from 0 to 4).

AGLERIM0

this routine empties the output data buffer and prints the current page.

AGLESIM0

this function allows the user to pass special instruction (hardware dependent escape sequences) to the output device.

Unsupported.

AGLCUIM0

this routine displays a cursor on the graphic surface (screen).

Unsupported.

AGLMVIM0

this routine sets the current pen position without performing a plot operation.

AGLSEIM0

this routine forces the sending of the output buffer to the device. In this implementation, the output buffer is flushed under the device driver control.

Unsupported.

AGLTEIM0

this routine empties the output data buffer, prints the current page and sends the EOF and the EOT commands to the printer, and thus terminates all plotting operations.

AGLTRIM0

this routine performs the plot operations, starting from the position of the last `AGLMVIM0` or `AGLTRIM0` commands.

4.2 Use

The use of the IMAGEN 8/300 AGL device driver is straightforward. All the entry points follow the AGL conventions described in [5] and can be called from every program using the "`img:`" device identification.

4.3 Comments

The AGL device driver for the IMAGEN 8/300 passed all the standard AGL tests and showed an improvement in the printing speed versus the PostScript and HPGL based printers.

Another feature peculiar to this implementation, is the added control of the buffer size during the draw operation. In our case, a buffer size of 1024 bytes demonstrated to best fit the user normal operations. However, no commands are lost due to the automatic flushing of the buffer to the printer every 1024 draw commands.

5 Listings

5.1 prcx

```
#
#     PRCX - Alphanumeric output script on IMAGEN 8/300
#
#        Version 1.0 - Sept. 1990
#
#        A.Balestra,  P.Marcucci,  R.Smareglia
#
##  find PID and define tempfile
#
a=`ps | grep prcx | sed -n '2,2p' | awk '{print $1}'`
name=`hostname`.$a
#
#  fiter
#
/users/uslib/exec/pressimg < $1 > /sys/print/queue/$name
#
#  print file and remove it
#
prf -pr cx -trans /sys/print/queue/$name > /dev/null
echo $1  queued to printer cx
rm /sys/print/queue/$name
#
```

5.2 pressimg

```
/************************************************************

NAME          : pressimg

AUTHORS       : A. Balestra, P. Marcucci, R. Smareglia  -  OAT

TYPE          : program
FILE          : pressimg.c
LANGUAGE      : C
REQUIREMENTS  : none
SIDE EFFECTS  : none

FUNCTIONAL DESCRIPTION :
driver for IMAGEN 8/300 printer; converts ASCII files in
IMPRESS language.

HISTORY       : 900921 -> Creation
              :
              :

************************************************************/

#include <stdio.h>
```

```
#define EOT                    4
#define TABLEN                 7
#define MAXLINES              64
#define MAXCOLS               79
#define SP                   128
#define SET_ABS_H            135
#define SET_ABS_V            137
#define CRLF                 197
#define SET_FAMILY           207
#define SET_IL               208
#define SET_BOL              209
#define SET_SP               210
#define ENDPAGE              219
#define CREATE_FAMILY_TABLE  221
#define EOD                  255

main()

{
   int c, tab;
   int lines = 0;
   int cols = 0;

   /* set document header */
   printf ("@document(language impress, paper a4, pagereversal on)");

   /* set current family to 0 */
   printf ("%c%c", SET_FAMILY, 0);

   /* map family 0 to the resident font courl0 */
   printf ("%c%c%c%ccourl0%c%c", CREATE_FAMILY_TABLE, 0, 1, 0, 0, 0);

   /* set beginning of line at 240 (1 inch from left margin) */
   printf ("%c%c%c", SET_BOL, 0, 240);

   /* set inter line spacing to 48 (20 lines per inch) */
   printf ("%c%c%c", SET_IL, 0, 48);

   /* set inter word spacing to 25 */
   printf ("%c%c%c", SET_SP, 0, 25);

   /* move to H position (1 inch from left margin) */
   printf ("%c%c%c", SET_ABS_H, 0, 240);

   /* move to V position (1 inch from top margin) */
   printf ("%c%c%c", SET_ABS_V, 0, 240);

   /* get character from input file until EOF is reached */
   while ((c = getchar ()) != EOF)
      {
      switch(c)
         {
         /* space case */
         case ' ':
         printf ("%c", SP);
         cols++;
         break;

         /* tab case */
         case '\t':
         for (tab = 0; tab <= TABLEN; tab++, cols++)
            printf ("%c", SP);
         break;
```

```
            /* newline case */
            case '\n':
            printf ("%c", CRLF);
            cols = 0;
            lines++;
            if (lines > MAXLINES)
               /* end of page */
               {
               lines = 0;

               /* print current page */
               printf ("%c", ENDPAGE);

               /* move to H position (1 inch from left margin) */
               printf ("%c%c%c", SET_ABS_H, 0, 240);

               /* move to V position (1 inch from top margin) */
               printf ("%c%c%c", SET_ABS_V, 0, 240);
               }
            break;

            /* print character */
            default:
            putchar(c);
            cols++;
            if (cols > MAXCOLS)
               /* next line when MAXCOLS reached */
               {
               cols = 0;
               printf ("%c", CRLF);
               lines++;
               if (lines > MAXLINES)
                  /* end of page */
                  {
                  /* print current page */
                  printf ("%c", ENDPAGE);

                  /* move to H position (1 inch from left margin) */
                  printf ("%c%c%c", SET_ABS_H, 0, 240);

                  /* move to V position (1 inch from top margin) */
                  printf ("%c%c%c", SET_ABS_V, 0, 240);
                  }
               }
            break;
            }
         }
         /* end of file */
         putchar(c);

         /* end of document */
         printf ("%c", EOD);

         /* end of transmission */
         printf ("%c", EOT);
      }
```

5.3 imagdrv.c

```
/*
 * HEADER : imagdrv.c      - Vers .002  - Sep 1990 -  AB, PM, RS
 */
```

```
/*****************************************************************************

/*  AGL imagen 8/300 driver                                                  *

/*                                                                           *
/*  The following entry points have been defined:                            *

/*  Function    Entry point                                                  *
/*                                                                           *
/*  Initialize  AGLINIM0                                                     *
/*  Cursor en.  AGLCUIM0                                                     *
/*  Erase       AGLERIM0                                                     *
/*  Escape      AGLESIM0                                                     *
/*  Move        AGLMVIM0                                                     *
/*  Draw        AGLTRIM0                                                     *
/*  Finish      AGLTEIM0                                                     *
/*  Flush buff. AGLSEIM0                                                     *

#include <stdio.h>

#include "sysdep.h"

#include "aglerror.h"
#include "drvcom.h"
#include "capsflag.h"

/*  device parameters definition                                             *

#define IMFLAGS  SEPALPHA | EXECOMMND
#define IMPLANES 1
#define IMMAXLWIDTH 4
#define IMCLR01  1
#define IMCLR02  1
#define IMCLR03  1
#define IMGLR01  1
#define IMGLR02  1
#define IMGLR03  1


#define INTERACTIVE  1          /* Device interactive flag mask            */
#define RETPIXVAL    2          /* Can return locator pixel value          */
#define PARTERASE    4          /* Partial erase flag mask                 */
#define SEPALPHA     8          /* Separated alpha plane flag mask         */
#define ERASEIT     16          /* The device must be erased on start      */

#define VERS35 35               /* Declare driver as version 3.5           */

#define CHARFACTOR 4.5          /* Character dimension multiplier          */

#define INTPARMS 7
#define FLTPARMS 8

#define NPTMAX 1024             /* Buffer dimension */

#define WOFS  3                 /* Physical width of the logical width 0 */

#define TRUE  1
#define FALSE 0
```

```
static FILE *filpt;
static int id0, id1;
static int rot;
static int nptbuf = 0;
static char penwidth;
static double xfact,yfact,xofst,yofst;
static double xdim,ydim;
static int imgxoff = 150;
static int imgyoff = 250;

unsigned char code;
unsigned char mode;
char          opcode;
short         vcount;
static short  x_pos[NPTMAX];
static short  y_pos[NPTMAX];
int           i;

/*             Portrait mode --+      +----- Landscape mode                  */
/*                             |      |                                       */
/*static double FACTC[2]    = { 1900, 2750 };    */    /* Conversion factors
static double FACTC[2]    = { 2096, 3220 };        /* Conversion factors    */
static double IMGMAXL[2]  = {,21.00, 29.70 };        /* Max lengths           */
static double IMGLENG[2]  = { 21.00, 29.70 };        /* Default lengths       */
static double PIXCM[2]    = { 118,  118  };  /* Pixel/cm (soft.limited)     */



static void showpage()
 {
   code   = (char) 219;              /* ENDPAGE */
   fwrite ( &code, 1, 1, filpt);
 }



static void mcd_path()
 {
     code   = (char) 230;              /* CREATE_PATH */
     vcount = (short) nptbuf;
     fwrite ( &code, 1, 1, filpt);
     fwrite ( &vcount, 1, 2, filpt);

     for(i=0; i<vcount; i++)
        {
        fwrite(&y_pos[i], 1, 2, filpt);
        fwrite(&x_pos[i], 1, 2, filpt);
        }

     code   = (char) 234;              /* DRAW_PATH */
     opcode = (char) 15;
     fwrite ( &code, 1, 1, filpt);
     fwrite ( &opcode, 1, 1, filpt);

     code   = (char) 232;
     opcode = penwidth;
     fwrite(&code, 1, 1, filpt);
     fwrite(&opcode, 1, 1, filpt);

     nptbuf = 0;
 }
```

```
/*****************************************************************************
/*                          CURSOR READ function                             *

  void AGLCUIM0 (AGLDVCOM)                    /* Read virtual cursor position  *
   struct bufcom *AGLDVCOM;
   {
   ERRCODE=AGLNOERR;
   }


/*****************************************************************************  *
/*                          DEVICE INIT function                             *

void AGLINIM0 (AGLDVCOM)
   struct bufcom *AGLDVCOM;
   {

    static char  * header =
         "@document(language imPRESS,copies 1,jobheader off)\n";

    extern void AG_DMSG();


    char auxbuf[PHNAMLNG];
    static char *filnam = "imagplot";
    char *pt;

    ERRCODE=AGLNOERR;
    switch(IBUFFR(0))                    /* Select function           */
     {
      case 0:                            /* Hardware initialization   */

      strcpy(auxbuf,filnam);
      AG_NEWN(auxbuf);

      if(*auxbuf == '\0')
       {
        ERRCODE=DEVOPNSEV;
        return;
       }

      filpt=fopen(auxbuf,"w");

      if (filpt==NULL)
       {
        AG_DMSG("Open error:",auxbuf);
        ERRCODE=DEVOPNSEV;
        return;
       }

      CHANNEL=0;

      pt=CHARBUF;                                 /* Get USRAUX string         */
      while(*pt++);                      /* Skip device name          */
      while(*pt++);                      /* Skip SYSAUX info          */
```

```
if(TOUPPER(*pt) == 'P')
 {                                        /* select portrait mode          */
  rot = FALSE;
  id0 = 0;
  id1 = 1;
 }
else
 {                                        /* select landscape mode (def.)  */
  rot = TRUE;
  id0 = 1;
  id1 = 0;
 }

strcpy(CHARBUF,auxbuf);                        /* Copy filename to buffer   */
AG_DMSG("Out to:",auxbuf);

fputs (header, filpt);

penwidth = (char) WOFS;                  /* Set pen width = 0 */

code    = (char) 225;                    /* Set path update mode = 0 */
mode = (char) 0;
fwrite(&code, 1, 1, filpt);
fwrite(&mode, 1, 1, filpt);

xfact = (RBUFFR(2)<1.0) ? RBUFFR(2) : 1.0;  /* Get scaling factors   */
yfact = (RBUFFR(3)<1.0) ? RBUFFR(3) : 1.0;
xfact = (xfact<=0.0) ? 1.0 : xfact;
yfact = (yfact<=0.0) ? 1.0 : yfact;

xdim = (RBUFFR(id0)<IMGMAXL[id0]) ? RBUFFR(id0) : IMGMAXL[id0];
xdim = (xdim<=0.0) ? IMGLENG[id0] : xdim;
ydim = (RBUFFR(id1)<IMGMAXL[id1]) ? RBUFFR(id1) : IMGMAXL[id1];
ydim = (ydim<=0.0) ? IMGLENG[id1] : ydim;
xofst = ((IMGMAXL[id0]-xdim)*0.5) * FACTC[id0];
yofst = ((IMGMAXL[id1]-ydim)*0.5) * FACTC[id1];
xfact = xfact * (xdim/IMGMAXL[id0]) * FACTC[id0];
yfact = yfact * (ydim/IMGMAXL[id1]) * FACTC[id1];
xofst *= FACTC[id0];
yofst *= FACTC[id1];
break;

case 1:                                 /* Pass back device characteristics  */


 *(CHARBUF) = '\0';

 RBUFFR(0) = xdim;
 RBUFFR(1) = ydim;

 IBUFFR(1) = IMFLAGS;
 IBUFFR(2) = IMPLANES;
 IBUFFR(3) = xdim*PIXCM[id0];
 IBUFFR(4) = ydim*PIXCM[id1];
 IBUFFR(5) = IMGLR01;
 IBUFFR(6) = IMGLR02;
 IBUFFR(7) = IMGLR03;
 IBUFFR(8) = VERS35;
 IBUFFR(9) = IMMAXLWIDTH;
```

```
        RBUFFR(2) = CHARFACTOR;
        RBUFFR(3) = 0.0;
        RBUFFR(4) = 0.0;
        RBUFFR(5) = 0.0;
        RBUFFR(6) = IMGLENG[id0];
        RBUFFR(7) = IMGLENG[id1];
        RBUFFR(8) = IMGMAXL[id0];
        RBUFFR(9) = IMGMAXL[id1];
        break;

      case 2:
        ERRCODE=UNSFEATINF;
        break;                                  /* Unsupported              */

      case 3:
        ERRCODE=AGLNOERR;                       /* Not used since vers. 3.3 */
        break;

      case 4:
        ERRCODE=AGLNOERR;                       /* Set pen width */
        penwidth = (char) (IBUFFR(1) + WOFS);
        break;

      case 5:
        ERRCODE=AGLNOERR;                       /*  TBD */
        break;

      case 6:
        break;

    }
  }

/*****************************************************************************
/*                          ERASE SCREEN function                            *

void AGLERIM0 (AGLDVCOM)
  struct bufcom *AGLDVCOM;
  {
    mcd_path();
    showpage();
    ERRCODE=AGLNOERR;
  }



/*****************************************************************************  *
/*                          ESCAPE function                                  *

void AGLESIM0 (AGLDVCOM)
  struct bufcom *AGLDVCOM;
  {
   ERRCODE=UNSFEATINF;
  }


/*****************************************************************************  *
/*                          MOVE function                                    *
```

```
void AGLMVIM0 (AGLDVCOM)
   struct bufcom *AGLDVCOM;
   {
    ERRCODE=AGLNOERR;
    if ( nptbuf > 1 )
       mcd_path();
    x_pos[0]=(short)(RBUFFR(0)*xfact+imgxoff);
    y_pos[0]=(short)(RBUFFR(1)*yfact+imgyoff);
    nptbuf = 1;
   }

/*****************************************************************************
/*                              BUFFER SEND function                         *

void AGLSEIM0 (AGLDVCOM)
   struct bufcom *AGLDVCOM;
    {
     ERRCODE = AGLNOERR;
    }


/*****************************************************************************  *
/*                              TERMINATE Function                           *

void AGLTEIM0 (AGLDVCOM)
   struct bufcom *AGLDVCOM;
   {
   extern void AG_DMSG();

   mcd_path();
   showpage();
   code   = (char) 255;                /* EOF */
   fwrite(&code, 1, 1, filpt);

   code   = (char) 4;                  /* EOT */
   fwrite(&code, 1, 1, filpt);

   fclose(filpt); filpt=NULL;
   AG_DMSG("Out file","closed");
   ERRCODE=AGLNOERR;
   CHANNEL=(-1);
   }


/*****************************************************************************
/*                              DRAW function                                *

void AGLTRIM0 (AGLDVCOM)
   struct bufcom *AGLDVCOM;
   {

   x_pos[nptbuf]=(short)(RBUFFR(0)*xfact+imgxoff);
   y_pos[nptbuf]=(short)(RBUFFR(1)*yfact+imgyoff);
   nptbuf++;

   if ( nptbuf >= NPTMAX )
      {
      mcd_path();
      x_pos[0] = x_pos[NPTMAX-1];
      y_pos[0] = y_pos[NPTMAX-1];
      nptbuf = 1;
      }
```

```
    ERRCODE=AGLNOERR;
    }
```

5.4 imagdef

```
/*
 * HEADER : imagdef.c       - Vers .001  - Feb 1990 -  AB, PM, RS
 *
 *  Driver entry point table for the IMAGEN Driver
 *
 */

{
DEF(AGLCUIM0);
DEF(AGLERIM0);
DEF(AGLESIM0);
DEF(AGLINIM0);
DEF(AGLMVIM0);
DEF(AGLSEIM0);
DEF(AGLTEIM0);
DEF(AGLTRIM0);
(void)setdev("imagen",AGLCUIM0,AGLERIM0,AGLESIM0,AGLINIM0,
                    AGLMVIM0,AGLSEIM0,AGLTEIM0,AGLTRIM0);
}
```

6 References

1. IMAGEN 8/300 Page Printer User's Guide version 2.1
   December 1984

2. IP-II Languages Reference Manual version 2.1
   December 1984

3. imPRESS Programmer's Manual version 2.1
   August 1984

4. IP-II Image Processor Disk Manager Program
   September 1985

5. GUIDE TO WRITING AN AGL DEVICE DRIVER Vers. 3.x
   L. Fini, February 1989

6. AGL 3.5 Reference Manual
   L. Fini, June 1989