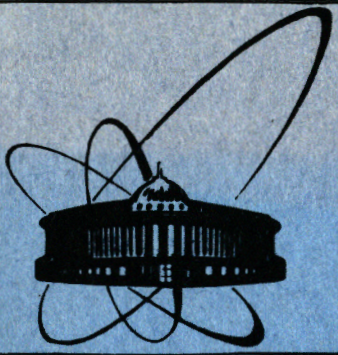

объединенный
институт
ядерных
исследований
дубна

E11-84-363

H.Leich, B.Naumann, V.I.Prikhodko,
M.Rudalics

REALIZATION OF GKS FUNCTIONS ON A MICROPROGRAMMED DISPLAY MODULE

Submitted to the Tenth Symposium on Microprocessing and Microprogramming (Copenhagen, 1984)

1984

1. INTRODUCTION

The extraordinary usefullness of computer graphics is motivated mainly by two reasons: firstly graphical presentation of informational contents is in many occasions not only a very friendly but also the only practicable way of presentation, secondly, only by means of computer graphics a man-machine dialogue is possible, as it may not be realized by other ways. In order to allow application programs involving graphics to be easily portable between different installations and to aid understanding and use of graphics methods by application programmers, standards for basic computer graphics are introduced. Currently at the Laboratory of Computing Techniques and Automation of the Joint Institute for Nuclear Research at Dubna an Intelligent Graphics Terminal (IGT) is in development/1/, which suffices the minimal requirements of a workstation of type OUTIN according to level 2c of the standard proposal "Graphical Kernel System" (GKS)/2/.

2. THE GRAPHICAL KERNEL SYSTEMS CONSIDERATIONS FOR GRAPHICAL INPUT/OUTPUT (AN OUTLINE)

The basic element of graphical output is the so-called graphical output primitive. GKS provides six output primitive types:

- POLYLINE: A set of connected lines defined by a point sequence;
- POLYMARKER: A set of symbols of one type centred at given positions defined by a point sequence;
- TEXT: A character string at a given position;
- FILL AREA: A polygon area which may be hollow or filled;
- CELL ARRAY: An array of pixels;
- GENERALIZED DRAWING PRIMITIVE (GDP): Special geometrical capabilities of a workstation such as drawing of circular arcs and elliptic arcs.

Three types of attribute (geometric, non-geometric and identification) may be assigned to each output primitive. The first two attribute types determine the exact appearance of the output primitive on the screen while the third attribute is used in connection with input. A separate GKS function is provided for each primitive attribute, to allow the application program to specify the value of an attribute without unnecessarily spe-

cifying the values of other attributes. During the creation of an output primitive these values are bound to the primitive and cannot be changed afterwards.

Attributes of the first type control the geometric aspects of primitives; these are aspects which affect the shape or size of a primitive. They are defined separately for each primitive.

The non-geometric aspects of a primitive may be specified in one of two ways, namely via a bundle or individually. For specification of aspects via a bundle, there is one pointer per primitive to a bundle table, each entry of which contains all the non-geometric aspects of the primitive. There are five bundle tables: polyline bundle table, polymarker bundle table, text bundle table, fill area bundle table, and intensity bundle table. For individual specification of aspects, there is a separate attribute for each non-geometric aspect. A further group of attributes, called ASPECT SOURCE FLAGS (ASF's), take the values INDIVIDUAL and BUNDLED to specify the choice. There is one ASF for each non-geometric aspect of each primitive.

There is precisely one attribute of the third type per primitive, namely PICK IDENTIFIER. It is used for identifying a primitive, or a group of primitives, in a segment, when this segment is picked.

Graphical output primitives may be grouped in segments as well as being created out of segments. Each segment is identified by a unique SEGMENT NAME. Segment attributes affect all the primitives in a segment. The segment attributes are:

- SEGMENT TRANSFORMATION: A segment can be scaled, rotated and translated;
- VISIBILITY: A segment is either displayed or not;
- HIGHLIGHTING: A visible segment is either highlighted or not;
- SEGMENT PRIORITY: When a user picks on the crossover of two or more segments, the segment with the higher priority is preferred;
- DETECTABILITY: A segment can either be selected by a pick input device or it cannot.

GKS provides six input classes:

- LOCATOR: provides a position;
- STROKE: provides a sequence of positions;
- VALUATOR: provides a real number;
- CHOICE: provides a non-negative integer number;
- PICK: provides a PICK STATUS, a SEGMENT NAME and a PICK IDENTIFIER;
- STRING: provides a character string.

Each input device can be operated in three modes:

- REQUEST: The application program requests an input and waits until the interaction has been performed, an input message is built and passed to the application program;

- SAMPLE: The application program samples the current input message without waiting for an interaction, each new input destroys the previous input message;
- EVENT: Subsequent input messages enter an event queue, which may be handled by the application program in a FIFO Discipline.

The ECHO function provided by GKS has been motivated by the consideration to give the operator the opportunity to recognize immediately, which interaction he has performed. Echoing may be implementation-dependent.

3. THE DISPLAY MODULE OF THE INTELLIGENT GRAPHICS TERMINAL

The IGT hardware is based on a multimicroprocessor system, which controls a vector display in refresh and store mode. The IGT incorporates:

- A monitor module (MM), responsible for communication with the host computer, memory management, and global distribution of functions within the IGT;
- An arithmetic module (AM), capable of autonomously performing transformations of graphical items;
- A display module (DM), which generates the display image and controls the input devices;
- A common memory module;
- A random scan display monitor.

A detailed description has been published in /3/ and /4/. The display module contains the following components:

- I8080 based CPU;
- 12 K PROM;
- 2 K RAM;
- input device interfaces;
- interrupt control;
- I3002 based, microprogrammed graphics processor (GP).

In Fig.1 the structure of the considered DM is presented. Originally this module was designed without having in mind GKS-considerations. Therefore all GKS output-requirements, not supported by the existing hardware, have to be realized by the GP's microprogram.

Because the effectivity of computer graphics essentially depends on the graphical output rate as well as comfortable input techniques, we aspire to realize the whole graphical output and the light-pen input without the help of the CPU. Special attention has to be given to the fact, that the GP may read data from the common memory (which allows GP immediately to interpret a display list stored in this memory) but is not capable of writing data into the memory. For that reason the

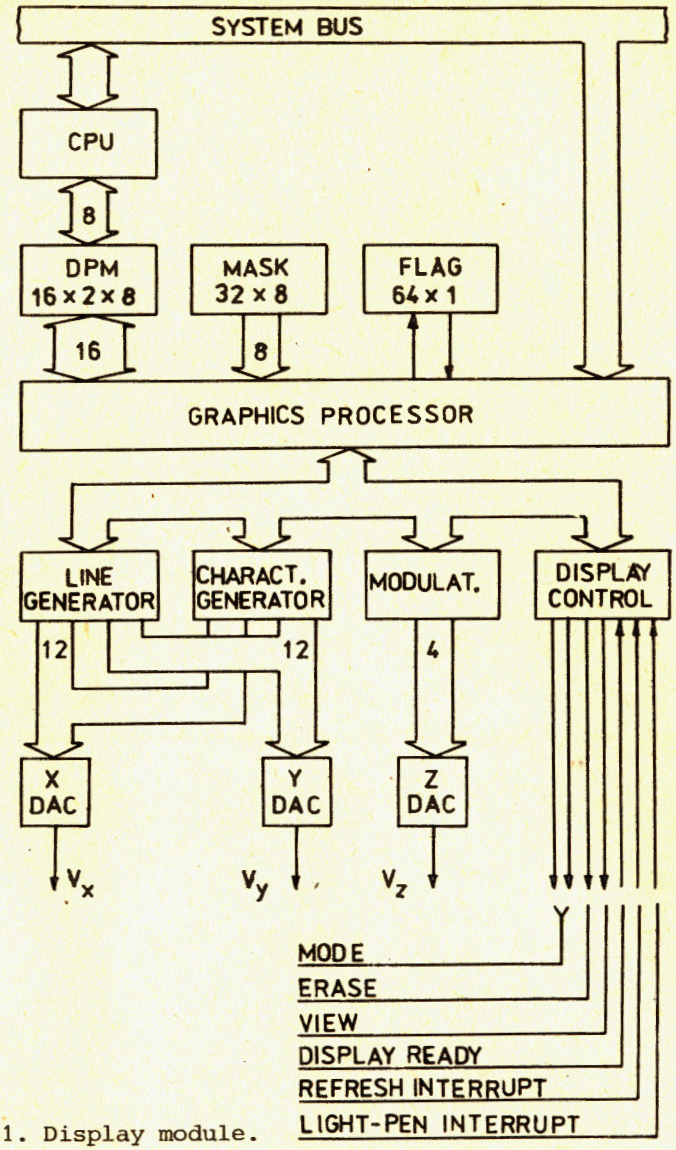

Fig.1. Display module.

light-pen input message has to be passed via the dual-port memory (DPM) to the CPU, which, in its turn, decides where to pass the message for further processing. The DPM is a special register file providing simultaneously access to its contents for the CPU and the GP. Furthermore the GP input data may be masked with constants from a mask file (a PROM). A set of hard-

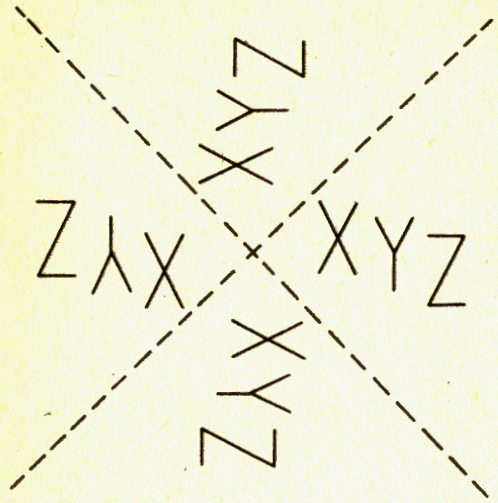

Fig.2. Text directions.

ware-flags supports the execution of the display list. Performing output functions the GP drives a line generator, a character generator, a modulator and controls the display monitor.

The line generator draws a vector from one position (current position) to another (next position). It is based on the constant rate vector generation principle, i.e., the time required to draw a vector is proportional to the length of the vector. The drawing process follows the concept of piecewise approximation/5/. The last position of a POLYLINE and the last position of a POLYLINE part (if a POLYLINE is clipped, it may be composed of more than one part) have to be marked by flags (END OF POLYLINE, END OF POLYLINE PART).

Characters (TEXT) and markers (POLYMARKER) are composed of short vectors (character vectors). These vectors are coded and stored in a 2K PROM. The character generator decodes the current character-vector code and draws the corresponding vector. The last character of a character string has to be marked by a flag (LAST CHARACTER). Two text fonts (latin and cyrillic), four text directions (see fig.2) and sixteen character sizes are provided. Markers are generated as special centred characters.

The modulator controls the intensity (brightness) of the electron beam. Sixteen intensity levels (eight normal and eight highlighted levels) are provided. Six linetypes (SOLID, DASHED, DASHED-DOTTED, DOTTED and their combinations) and BLINKING are supported.

The display may be operated in three modes: refresh, store and write through mode. Working in store mode it is possible to erase (ERASE) and to restore the brightness of the stored image (VIEW). Additionally the display control handles interrupts from the light-pen input device, refresh interrupts and the status line "DISPLAY READY".

The basic instruction set of the GP consists of the I3002 instruction set/6/ and additional instructions (see table 1). Special attention has to be given to the fact, that the GP is capable to perform only fixed-point arithmetics operations (addition, subtraction). The GP-control has an available stack, which allows it to realize a comfortable subroutine technique.

Table 1

Instruction set

```
I3002 instruction set;
jump (JUMP, JUMP ON CONDITION, JUMP ON FLAG, NOP, BRANCH);
call (CALL, CALL ON CONDITION, CALL ON FLAG);
return (RETURN, RETURN ON CONDITION);
wait (WAIT ON CONDITION);
SET FLAG;
EXECUTE MICROOPERATION;
draw (DRAW POLYLINE VECTOR, DRAW CHARACTER VECTOR);
```

Table 2

Condition signals (all condition signals are also inverted available)

```
FALSE;
interrupt (LIGHT PEN INTERRUPT, REFRESH INTERRUPT);
ready (DISPLAY READY, DATA IN READY);
CARRY;
control line generator (END OF POLYLINE VECTOR, END OF PO-
  LYLINE PART, END OF POLYLINE);
control character generator  (END OF CHARACTER VECTOR, END
  OF CHARACTER, LAST CHARACTER);
```

Table 3

Microoperation set

```
load parameter (LOAD X-COORDINATE, LOAD Y-COORDINATE, LOAD
  LINETYPE, LOAD TEXT FONT, LOAD TEXT DIRECTION, LOAD CHA-
  RACTER TYPE, LOAD CHARACTER SIZE, LOAD INTENSITY);
CONTROL LINE GENERATOR;
CONTROL CHARACTER GENERATOR;
control blinking (SET BLINKING, RESET BLINKING);
control display mode  (SET REFRESH MODE,  SET  STORE MODE,
  ERASE, VIEW);
control interrupt  (SET CPU INTERRUPT, RESET LIGHT-PEN IN-
  TERRUPT, RESET REFRESH INTERRUPT);
```

For the conditional execution of instructions (JUMP ON CONDITION, CALL ON CONDITION, RETURN ON CONDITION, WAIT ON CONDITION, JUMP ON FLAG, CALL ON FLAG) sixteen condition lines (see table 2) as well as 64 flags may be examined. By means of the instruction "EXECUTE MICROOPERATION", the so-called microoperations (see table 3) and the draw-instructions (DRAW POLYLINE VECTOR, DRAW CHARACTER VECTOR) the GP performs the control of the line generator, the character generator, the modulator, the interrupt handling, and the display mode.

4. THE IMPLEMENTATION OF GKS OUTPUT FUNCTIONS ON THE GP

In order to realize the GKS output functions at a high rate, i.e., without complicated arithmetical computations in the GP, the graphical data in the display list should satisfy the following constraints:

- Positions (coordinates) must be transformed to display coordinates (DC). DC are fixed-point integer numbers which may be directly interpreted by the GP;
- Attributes should be transformed to fixed-point integer numbers. Only such attributes will be handled by the GP which may be immediately interpreted by the line generator, character generator, or modulator.

The resulting organization of graphical output data in the display list is as follows:

- Positions (coordinates) and attributes are transformed to fixed-point integer numbers by the AM;
- The POLYLINE attributes LINETYPE (ASF, INDIVIDUAL, BUNDLED), LINE INTENSITY (ASF, INDIVIDUAL, BUNDLED) will be handled directly by the GP, LINE WIDTH is not supported;
- The POLYMARKER attributes MARKER TYPE (ASF, INDIVIDUAL, BUNDLED), MARKER SIZE SCALE FACTOR (ASF, INDIVIDUAL, BUNDLED), MARKER INTENSITY (ASF, INDIVIDUAL, BUNDLED) will be handled directly by the GP;
- Because of the limited attribute handling capabilities of the character generator and the limited register set of the GP the TEXT attributes are handled as follows:
  - Because we only realize the text-precision CHAR provided by GKS the attribute TEXT FONT AND PRECISION is reduced to TEXT FONT (two fonts are supported);
  - The attributes TEXT INTENSITY (ASF, INDIVIDUAL, BUNDLED) will be directly handled by the GP;
  - The attributes CHARACTER EXPANSION FACTOR (ASF, INDIVIDUAL, BUNDLED), CHARACTER SPACING (ASF, INDIVIDUAL, BUNDLED), CHARACTER HEIGHT, CHARACTER UP VECTOR, TEXT PATH, TEXT ALIGNMENT are interpreted by the AM and transformed to the attributes CHARACTER DISPLACEMENT, CHARACTER SIZE, TEXT DIRECTION which may be handled by the GP;
- The output primitives FILL AREA and CELL ARRAY are handled as POLYLINE;
  - The representation of FILL AREA is realized with the interior style HOLLOW, i.e., only the bounding POLYLINE is drawn, the attributes FILL AREA INTENSITY (ASF, INDIVIDUAL, BUNDLED) will be directly handled by the GP;
  - The representation of CELL ARRAY is realized by drawing the bounding POLYLINE with an implementation-specified intensity;
- The output primitive GDP is handled as POLYLINE, circular arcs are provided.

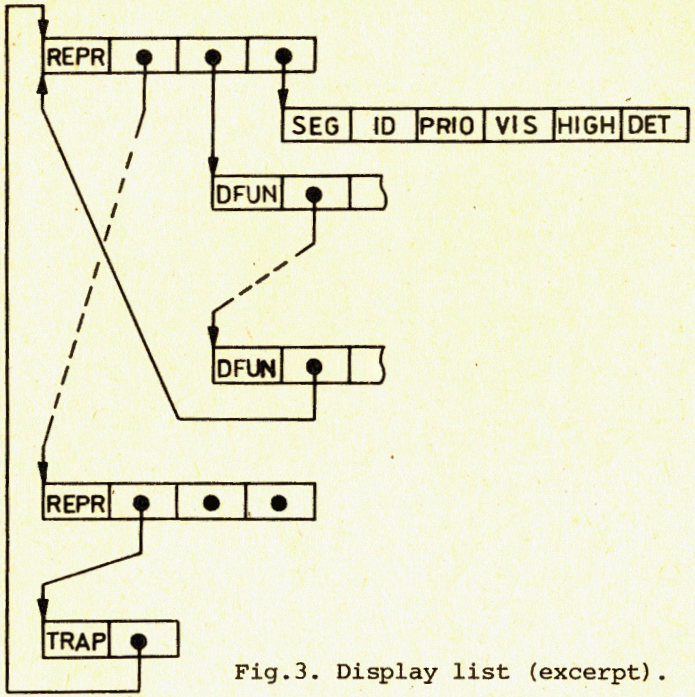

Fig.3. Display list (excerpt).

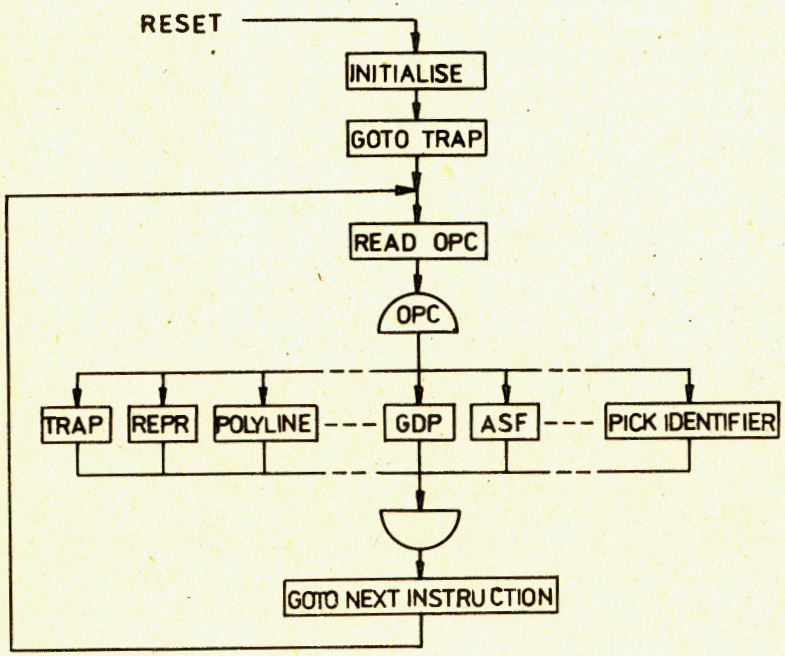

Fig.4 Main loop.

After transformation by the AM all GKS functions, which may be handled by the GP, are arranged in a list structure: the display list. The GP scans the display list, which contains all necessary information to generate the display image/7/. In fig.3 a simplified part of this data structure is presented. The first field of each element of the structure contains the operation code (OPC). Segments (SEG) are designated by SEGMENT NAME (ID), PRIORITY (PRIO), VISIBILITY (VIS), HIGHLIGHTING (HIGH) and DETECTABILITY (DET). The output of segments is performed by means of representations (REPR) and display functions (DFUN). Exactly one representation corresponds to each segment to be displayed. The representations are arranged according to the priority of the corresponding segment. The advantage of this arrangement is the following: the first segment, which causes an impulse in a pick input device placed on the crossover of two or more segments is definitely the segment with the highest priority of all segments causing a pick impulse. It may be selected without the necessity to take into consideration other segments. To each representation a set of display functions, which contain the data to be displayed, is subordinated. There are two groups of display functions:

- Output primitive functions: To each graphical output primitive type corresponds an output primitive function (POLYLINE, POLYMARKER, TEXT, FILL AREA, CELL ARRAY, GDP);
- Load attribute functions: To each attribute handled by the GP corresponds a load attribute function (ASF, LINETYPE INDIVIDUAL, LINE INTENSITY INDIVIDUAL, POLYLINE ATTRIBUTES BUNDLED, MARKER TYPE INDIVIDUAL, MARKER SIZE INDIVIDUAL, MARKER INTENSITY INDIVIDUAL, POLYMARKER ATTRIBUTES BUNDLED, CHARACTER SIZE, TEXT DIRECTION, CHARACTER DISPLACEMENT, TEXT FONT INDIVIDUAL, TEXT INTENSITY INDIVIDUAL, TEXT ATTRIBUTES BUNDLED, FILL AREA INTENSITY INDIVIDUAL, FILL AREA INTENSITY BUNDLED, PICK IDENTIFIER).

In refresh mode all representations with their corresponding display functions should be processed in each refresh cycle. The refresh cycle starts and stops with a TRAP instruction, which realizes the necessary synchronization to obtain a stable display image.The processing of primitives out of segments (not discussed here) is also performed in the TRAP-command.

Fig.4 illustrates the GP's main program loop, table 4 presents the corresponding microprogram. The CPU initializes the DPM-register TRAP×ADDR and starts the GP (by means of a flag in the DPM-register which is initially cleared and then interrogated by the GP). The GP scans the display list; the actual address handled by the GP is stored in CURR×ADDR. After reading the first field of a display list element the GP performs a branch according to the actual operation code to a subroutine which realizes the desired function. At the start point each

Table 4

Main loop (microprogram)

```
.
START:       CALL   'INITIALISE'; .initialization
             LDTE   'TRAP*ADDR';  .T:=TRAP*ADDR
             LDA    'T';          .AC:=T
             STA    'CURR*ADDR';  .CURR*ADDR:=AC
NEXT*INSTR:  CALL   'READ*BYTE';  .AC:=byte(CURR*ADDR)
                                  .CURR*ADDR:=CURR*ADDR+1
             BRANCH 'OPC*TABLE';  .go to OPC*TABLE+(AC)
.
OPC*TABLE:   JMP 'TRAP';      .go to TRAP
             JMP 'REPR';      .go to REPR
             JMP 'POLYLINE'; .go to POLYLINE
             .
             JMP 'GDP';       .go to GDP
             JMP 'ASF';       .go to ASF
             .
             JMP 'PICK*ID';  .go to PICK*ID
.
LOAD*NEXT*ADDR: LDA 'NEXT*ADDR';  .AC:=NEXT*ADDR
                STA 'CURR*ADDR';  .CURR*ADDR:=AC
                JMP 'NEXT*INSTR'; .go to NEXT*INSTR
```

Table 5

Microprogram subroutines (ASF, LINETYPE*INDIVIDUAL, LINE*INTENSITY*INDIVIDUAL, POLYLINE*ATTRIBUTES*BUNDLED)

```
.
ASF:
 CALL 'SAVE*NEXT*ADDR'; .NEXT*ADDR:=word(CURR*ADDR)
                        .CURR*ADDR:=CURR*ADDR+2
 CALL 'LOAD*ASF';       .load ASF's
                        . (LINE*TYPE*ASF, LINE*INTEN*ASF,
                        .  MARK*TYPE*ASF, MARK*SIZE*ASF,
                        .  MARK*INTEN*ASF,
                        .  TEXT*FONT*ASF, TEXT*INTEN*ASF,
                        .  FILL*AREA*INTEN*ASF)
 JMP  'LOAD*NEXT*ADDR'; .go to LOAD*NEXT*ADDR
.
LINETYPE*INDIVIDUAL:
 CALL 'SAVE*NEXT*ADDR'; .NEXT*ADDR:=word(CURR*ADDR)
                        .CURR*ADDR:=CURR*ADDR+2
 CALL 'LOAD*LINE*TYPE*INDIV';
                        .LINE*TYPE*INDIV:=byte(CURR*ADDR)
 JMP  'LOAD*NEXT*ADDR'; .go to LOAD*NEXT*ADDR
.
LINE*INTENSITY*INDIVIDUAL:
 CALL 'SAVE*NEXT*ADDR'; .NEXT*ADDR:=word(CURR*ADDR)
                        .CURR*ADDR:=CURR*ADDR+2
 CALL 'JUMP*BUNDL';     .CURR*ADDR:=word(CURR*ADDR)+7
 CALL 'LOAD*LINE*INTEN*INDIV';
                        .LINE*INTEN*INDIV:=byte(CURR*ADDR)
 JMP  'LOAD*NEXT*ADDR'; .go to LOAD*NEXT*ADDR
.
```


Table 5 (continued)

```
POLYLINE*ATTRIBUTES*BUNDLED:
 CALL 'SAVE*NEXT*ADDR'; .NEXT*ADDR:=word(CURR*ADDR)
                        .CURR*ADDR:=CURR*ADDR+2
 CALL 'JUMP*BUNDL';     .CURR*ADDR:=word(CURR*ADDR)+7
 CALL 'LOAD*LINE*TYPE*BUNDL';
                        .LINE*TYPE*BUNDL:=byte(CURR*ADDR)
                        .CURR*ADDR:=CURR*ADDR+1
 CALL 'JUMP*BUNDL';     .CURR*ADDR:=word(CURR*ADDR)+7
 CALL 'LOAD*LINE*INTEN*BUNDL';
                        .LINE*INTEN*BUNDL:=byte(CURR*ADDR)
 JMP  'LOAD*NEXT*ADDR'; .go to LOAD*NEXT*ADDR
.
```

Table 6

Microprogram subroutines (POLYLINE)

```
.
POLYLINE:
 CALL 'SAVE*NEXT*ADDR';
      .NEXT*ADDR:=word(CURR*ADDR), CURR*ADDR:=CURR*ADDR+2
 CALL 'LOAD*LINETYPE';
      .if LINETYPE*ASF=INDIV
      .   then LINETYPE:=LINETYPE*INDIV
      .   else LINETYPE:=LINETYPE*BUNDL
 CALL 'LOAD*LINE*INTEN';
      .if LINE*INTEN*ASF=INDIV
      .   then INTEN:=LINE*INTEN*INDIV+HIGHL
      .   else INTEN:=LINE*INTEN*BUNDL+HIGHL
 CALL 'DRAW*POLYLINE';
 JMP  'LOAD*NEXT*ADDR'; .go to LOAD*NEXT*ADDR
.
DRAW*POLYLINE:
DRLI1: CALL 'MOVE*ABSOL';
DRLI2: JNF  'END*POLYLINE*FLAG,DRLI3';
            .if END*POLYLINE*FLAG=0 then go to DRLI3
       RET; .return
DRLI3: JOF  'END*POLYLINE*PART*FLAG,DRLI1';
            .if END*POLYLINE*PART*FLAG=1 then go to DRLI1
       CALL 'DRAW*POLYLINE*VECTOR';
       JMP  'DRLI2'; .go to DRLI2
.
MOVE*ABSOL:
 CALL 'READ*WORD';
      .AC:=word(CURR*ADDR), CURR*ADDR:=CURR*ADDR+1
 STA  'CURR*POSIT*X'; .CURR*POSIT*X:=AC
 CALL 'READ*WORD';
      .AC:=word(CURR*ADDR), CURR*ADDR:=CURR*ADDR+2
 STA  'CURR*POSIT*Y'; .CURR*POSIT*Y:=AC
 CALL 'LOAD*END*POLYLINE*FLAG';
      .END*POLYLINE*FLAG:=w1a2(AC)
 CALL 'LOAD*END*POLYLINE*PART*FLAG';
      .END*POLYLINE*PART*FLAG:=w1a3(AC)
 CALL 'MOVE'; .move beam to CURR*POSIT
 RET; .return
```

Table 6 (continued)

```
DRAW×POLYLINE×VECTOR:
 CALL   'READ×WORD';
        .AC:=word(CURR×ADDR), CURR×ADDR:=CURR×ADDR+2
 STA    'NEXT×POSIT×X'; .NEXT×POSIT×X:=AC
 CALL   'READ×WORD';
        .AC:=word(CURR×ADDR), CURR×ADDR:=CURR×ADDR+2
 STA    'NEXT×POSIT×Y'; .NEXT×POSIT×Y:=AC
 CALL   'LOAD×END×POLYLINE×FLAG';
        .END×POLYLINE×FLAG:=w1a2(AC)
 CALL   'LOAD×END×POLYLINE×PART×FLAG';
        .END×POLYLINE×PART×FLAG:=w1a3(AC)
 CALL   'INIT×LINE×GENERATOR'; .initialise line generator
 WNC    'DISPLAY×READY'; .if DISPLAY×READY=0 then wait
 DRAWL;.draw POLYLINE VECTOR from CURR×POSIT to NEXT×POSIT
 CALL   'LOAD×NEXT×POSIT'; .CURR×POSIT:=NEXT×POSIT
 RET;   .return
```

subroutine saves the address of the next list element (NEXT× ADDR). After processing of the actual list element the address of the next list element is loaded and this element is processed.

A simplified part of the display list and the corresponding subroutines are presented at fig.5 and tables 5,6. It is shown how the subroutines ASF, LINETYPE×INDIVIDUAL, LINE×INTENSITY× INDIVIDUAL, POLYLINE×ATTRIBUTES×BUNDLED and POLYLINE implement the desired functions..

5. CONCLUSION

At present the design of graphics systems is more and more influenced by the trend for standardization. The Graphical Kernel System standard proposal defines a set of basic elements for graphical output/input. The implementation of the standard's requirements by means of simulation of functions by software on an existing hardware, which was developed without having in mind these requirements, may cause some difficulties and may significantly decrease the system's throughout. In the paper the realization of GKS output-functions on a microprogrammed special processor for display control is described. Using some additional circuitry in the microprogrammed processor and selecting suitable representation of graphical data in the display list a high output rate as well as smart response characteristics for light-pen input are achieved.

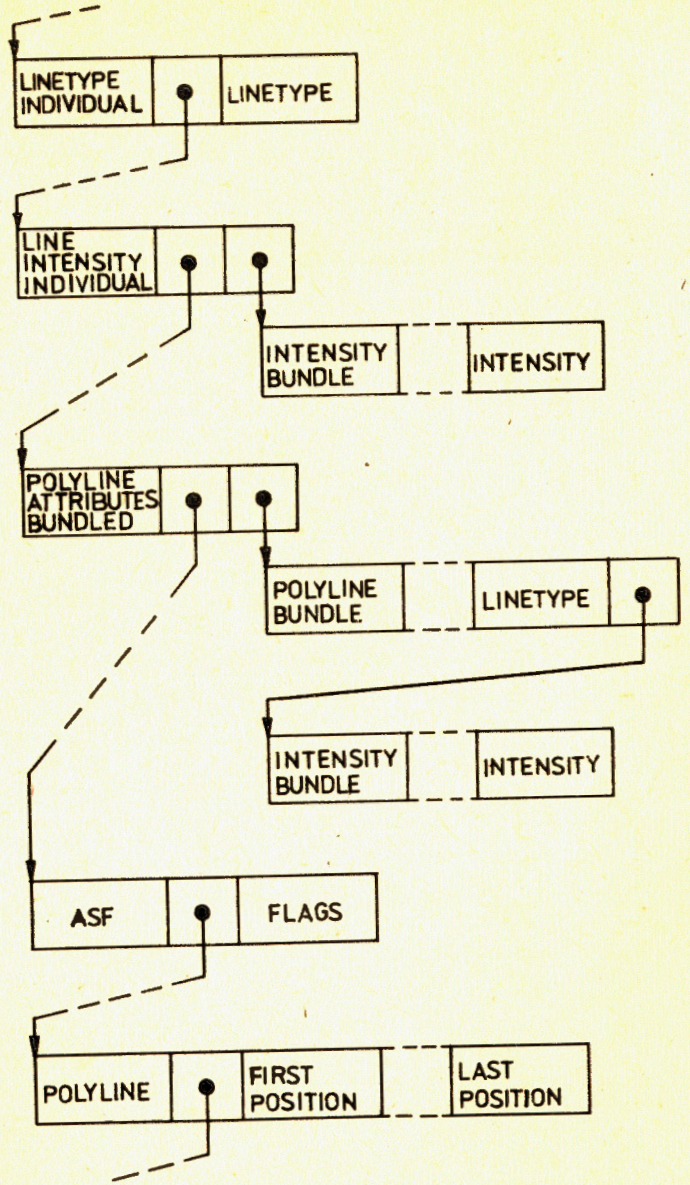

Fig.5. Display list (excerpt).

REFERENCES

1. Leich H., Levchanovsky F.U., Prikhodko V.I. Microprocessing and Microprogramming, 1983, 12, 3/4, pp. 175-180.

2. Draft International Standard ISO/DIS 7942. Information Processing Graphical Kernel System (GKS), Functional Description, Version 7.2, NI-5.9/1-83, Nov., 1982.
3. Leich H., Levchanovsky F.U., Prikhodko V.I. JINR, P10 83-7, Dubna, 1983 (in Russian).
4. Leich H., Levchanovsky F.U., Prikhodko V.I. JINR, P11-83-385, Dubna, 1983 (in Russian).
5. Kamae I. et al. IEEE Trans. on Compt. C-28, 1979, 10, pp.728-738.
6. INTEL SERIES I3002 REFERENCE MANUAL. Intel Publication 98-221A, Intel Corporation, Santa Clara.
7. Rudalics M. JINR, E11-83-393, Dubna, 1983. In: Proc. of of EUROGRAPHICS'83. North-Holland Pub., 1983, pp.383-392.

Received by Publishing Department
on May 24, 1984.

SUBJECT CATEGORIES OF THE JINR PUBLICATIONS

Index	Subject
1.	High energy experimental physics
2.	High energy theoretical physics
3.	Low energy experimental physics
4.	Low energy theoretical physics
5.	Mathematics
6.	Nuclear spectroscopy and radiochemistry
7.	Heavy ion physics
8.	Cryogenics
9.	Accelerators
10.	Automatization of data processing
11.	Computing mathematics and technique
12.	Chemistry
13.	Experimental techniques and methods
14.	Solid state physics. Liquids
15.	Experimental physics of nuclear reactions at low energies
16.	Health physics. Shieldings
17.	Theory of condenced matter
18.	Applied researches
19.	Biophysics

Лайх Х. и др. E11-84-363

Реализация GKS-функций в микропрограммируемом дисплейном модуле

В настоящее время в ЛВТА ОИЯИ разрабатывается интеллектуальный графический терминал, который удовлетворяет минимальным требованиям к рабочей станции типа OUTIN и соответствует уровню 2С стандарта Graphical Kernel System (GKS). Выполнение требований стандарта на существующей аппаратуре, которая не была ориентирована на его использование, может вызвать некоторые трудности и значительно уменьшить пропускную способность системы. В данной работе рассматривается реализация GKS-функций в микропрограммируемом специальном процессоре, предназначенном для управления дисплеем. Путем использования некоторых дополнительных приемов и выбора подходящего представления графических данных удалось достичь высокой скорости вывода информации и быстрой реакции на прерывания от светового карандаша.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1984

Leich H. et al. E11-84-363

Realization of GKS Functions on a Microprogrammed Display Module

Currently at the Laboratory of Computing Techniques and Automation of the Joint Institute for Nuclear Research an Intelligent Graphics Terminal is in development, which suffices the minimal requirements of a workstation of type OUTIN according to level 2c of the standard proposal Graphical Kernel System (GKS). The implementation of the standards requirements on an existing hardware, which was developed without having in mind these requirements, may cause some difficulties and may significantly decrease the systems throughout. The realization of GKS functions on a microprogrammed special processor for display control is described. Using some additional circuitry and selecting suitable representation of graphical data a high output rate as well as smart response characteristics for light-pen input are achieved.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1984