SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 1

The FR80 SPOOL System

issued by
R E Thomas

28 November 1974

DISTRIBUTION: FRAH RB AWB MFC AHF JRG
JWEL PMN DVR JMR RET

This paper concentrates on defining the final SPOOL system, with the various stages required to get the job done. Other papers will describe the systems on the two machines in more detail. However, it is affected considerably by the timescales for the Host software (SPROGS, POLYGRAPHICS etc) and by the personnel involved.

1. FINAL TAPE FORMAT

Eventually, the system will use something akin to the LOAD-GO feature on the FR80. An accounting program will be available (to be written) which can handle accounts and the generation of leader frames (covered in a separate paper). Each job on tape (generated directly, or from SPOOL) will have the following format:

Record 1	Name of FR80 accounts program
Record 2	Monitor commands (probably none)
Record 3	Data for leader frames, zero frame count
Filemark	
Record 1	Name of FR80 program to be selected
Record 2	Monitor commands to perform standard (or required) settings of parameters
Filemark	
Record 1	Data for FR80
etc	

Filemark

Record 1	Name of FR80 accounts program
Record 2	Monitor commands (probably none)
Record 3	Extra accounts details, if required

Filemarks are necessary to ensure the change of program. An extra filemark before the FR80 data ensures that, should LOAD-GO become inoperable, the data can still be obtained in a single program environment. Tapes from other machines, and, probably initially, SD4020 tapes, are likely to have data only, and not to have any accounts record. Such tapes will always have to be dealt with manually.

In the following, 'device' will be used to mean FR80 or SD4020, and 'camera' to include the program in the FR80 which decodes the tape - thus 16 mm output with 4 msec vectors is treated as a different camera from 16 mm output with 8 msec vectors (since different FR80 programs are involved).

2. SPOOL DATA (1906A)

The SPOOL system should aim to provide automatically as much of the red tape as possible, thus saving user time at the expense of system overhead. Each job should consist of a header record, followed by data, and it will be desirable that all records are the same length (since the SPOOL will reside on fixed block discs). The header must include the following information, some of which can only be inserted at the end of the job.

Record Count
Frame Count
Username
Output device
Camera
Data for MONITOR commands
Data for accounts
Data for leader generation

SPOOL should never need to decode the picture data information for any details.

SPOOL can generate all records in the tape format, except for the data records, from information in the header. SPOOL must also separate into different devices and camers, so that all jobs for the same camera are put together on tape. It is assumed that each camera will select one FR80 program. Variable speed options, which need separate programs, can be handled by different camera numbers.

3. EMPTYING SPOOL

To ensure a good turn-round, SPOOL should be emptied reasonably frequently. The current SPOOL provides a fixed number of blocks per user, and so requires no garbage collection.

Output should not be held up in the SPOOL just because there is only one job for a given camera. This ideal may not be reached in practice, and it may in fact be necessary to delay some jobs. The SPOOL area must be large enough to accommodate sufficient number of users to prevent frequent hold-ups. Failed jobs currently lose all their SPOOL output, since the tables are updated at the end of the run.

The delaying of jobs can be lessened to some extent by introducing more than one SPOOL area and using one or the other. On the 1906A, where SPOOL areas will be exofiles, there is no means, other than use of an external semaphore, of telling whether a given exofile is already in use. A reasonable, and extensible, system can be provided by randomly selecting one of the available areas (reading the clock, for example). However, a study of the work handled by the SD4020 now shows that, on average over the last year, 45 jobs are processed per day (29-62), and the daily average of frames is 676 hardcopy (478-765), 6771 16 mm (4613-10062), and 1975 35 mm (351-2962). With this load, it is unlikely that a second SPOOL area is required.

It is not envisaged that any special arrangements need to be made to save SPOOLed output for future plotting on a long term basis. Such jobs should be run to private tapes as before. However, there will be some point in prescribing old SPOOL information on the 1906A when the BSI is linked to VIEW. In this case, a daily well of jobs could be preserved in exofile, which could be interrogated by VIEW. Jobs are likely to remain in the SPOOL for so short a time that directly accessing the SPOOL has minimal value.

FR80 can handle 7 or 9-track tapes. Data should be stored in 7-track form in the SPOOL for space reasons, and 9-track output, if required, generated from SPOOL.

4. STAGES IN WRITING SPOOL

The first stage of SPOOL can provide a system very similar to that currently available on the 1906A. Data is written to disc, and the SPOOL just copies it to tape. However, it should be slightly modified to scan for each camera type and output information on to separate tapes. Such tapes would not use the LOAD-GO system but would be operator controlled, unless the Host package provided the extra records.

There are two ways of proceeding at this stage. Eventually it will be necessary for the SPOOL to output the extra records itself, but it will always be necessary for Host software to generate these records when writing tapes (saved output or long jobs). I believe that it is better for at least one Host system to write the records correctly before any attempt is made to alter the SPOOL (rather than the reverse) for the following reasons:

(1) Format of these records on tape and disc should be the same, and so it is easier for one person to have control over both.

(2) The same program can be used to write tape and disc (as it is now), so the LOAD-GO system will work at the same time for both.

(3) There will be no problems about testing new SPOOL features while this development takes place. Hence all jobs from other packages will still work.

(4) The SPOOL system on the 360/195 is not being written by ACL.

When one Host system has solved the LOAD-GO format problem, the same routines can be placed in SPOOL, and extra records can be generated. Data from the Host will then be written to the header record in the SPOOL rather than generating MONITER commands direct. In this way, less time will be taken with the Host package for the majority of users.

Following the comments about interpreting data in SPOOL, identification frames should be handled initially by the Host, until the appropriate FR80 programs have been written. It should not be necessary for SPOOL ever to add picture data, only parameters and system calls.

5. SPOOL ON THE 360/195

5.1 The Current System

RL73-115 describes the system. It is currently associated with the SCFOR package, and a library containing special versions of IDST,IDEND etc has to be included. Output is written to two data sets; one contains SD4020 orders, and the other contains operator instructions. These data sets are attached by JCL commands, and can be any suitable data set, although in practice, since there are a number of formalities to go through to obtain private data sets, they appear to be restricted to one per group of users.

A data set is given a primary size and an increment. IDEND checks to see if the file has been extended, and, if so, initiates the de-SPOOL job. This job uses IEBGENER utility to generate the SD4020 tape. It is also possible to run another job to force the emptying of the SPOOL system.

Only one job may access a data set at a time. If another program requests the same set, the operating system issues a message asking whether the operators wish to cancel the program or leave it to wait. The message is repeated every 30 sec if waiting. Currently, the SPOOL used by ACL people is sometimes cancelled, but 360/195 operators have been told to leave such jobs to wait.

5.2 Disadvantages with Current System

(1) It is possible to have many SPOOLS, each using this system. Such multiplicity will tend to force extra tapes to be generated.

(2) Since the de-SPOOL is only initiated (or not) at the end of a job, it is possible for output to wait for long periods. Currently, ACL find that they must remember to run a de-SPOOL at least once a day to make the system usable.

(3) With many people using the SPOOL, clashes are more likely, and there will be many wait messages. Operators may be tempted to cancel such jobs in error.

(4) The current SPOOL is linked to the SCFOR package. A more general version would be useable by all 360/195 graphics packages.

5.3 Possible extensions to existing SPOOL

(1) A set of standard data sets be provided, one for each camera. An SD4020 data set can also be provided.

(2) The data sets be made large enough to hold all graphics from normal users over a given time period. Large users should be allowed access to tape directly, as now.

(3) HASP issues a de-spool job automatically on a time basis. This will look at specific named data sets, and so avoid user private SPOOLs being proliferated.

(4) Clash messages for SPOOL areas should invoke an automatic wait. Message should only be printed to the operator if the wait is excessive.

Study of the existing SPOOL programs should indicate whether they can be modified to accommodate this scheme.

5.4 Use of HASP direct

HASP is itself a spooling system. It would therefore seem possible to define some extra slots for each camera, and SPOOL directly to HASP (using SYSOUT). The de-SPOOL system would then be HASP-initiated. In this case, it is probably advisable to write the operation details as a trailer to the data, rather than to a separate SPOOL.

Such a system would be far cleaner from the user viewpoint. However, it does require permission, and it does require someone with a knowledge of HASP.

APPENDIX

Current 1906A SPOOL Layout

Exofile ACLGRAPHICS , 87 cylinders (3145K words)
256-word buckets, all checksummed.

25 users have 502 data buckets + 1 information bucket each.

1 file control bucket, updated only after graphics data completed.

Disc Layout

Bucket	1	File Control
	2-26	User information buckets
	27-30	Not used
	31	Start of data area

File Control Bucket Layout

Word	0	Indicates room for another user, or no room
	1	Date of last emptying
	2	Time of last emptying
	3	Date last used
	4	Time last used
	5	
	6	Number of jobs in SPOOL
	7	Date of last emptying in characters
	8	
	9	Time of last emptying in characters
	10	

User Information Bucket Layout

Word	0-2	Username
	3-5	Jobname
	6-17	Person's name
	18	Date written
	19	Time written
	20	
	21	Date written, in characters
	22	
	23	Time written, in character
	24	
	25	Number of Data Blocks
	26	Number of hardcopy frames

27	Number of microfilm frames
28	Not used
29-34	Time and date entered here when file is emptied
35	Device indicator
36-254	List of data blocks which are followed by filemarks

gm

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 2

Leader Frames for FR80

issued by
R E Thomas

28 November 1974

DISTRIBUTION: FRAH RB AWB MFC AHF JRG
JWEL PMN DVR JMR RET

It is customary to add extra frames to the front of a graphics job, and sometimes to the end as well. These serve the following functions:

(1) Provide some easily recognisable way of separating different jobs in the same reel of film.

(2) Provide a means of identification visible to the naked eye on microfilm, for the owner of the output.

(3) Provide date and time of run.

(4) Provide jobname of job that created it.

(5) Provide registration frame to give some gauge as to the accuracy of the output.

(6) Provide some expendable space between jobs in case the last frame of the last job did not terminate correctly.

Typically, this is provided by (in SPROGS)

Blank Frame (to end any previous unfinished output)	(6)
Frame with cut mark	(1)
Frame with forms flash	(1),(2)
Frame with box and diagonal lines	(5)
Username in large letters	(2)
Jobname, date and time	(3),(4)

The following suggestions deal with the requirements on the FR80. Eventually, it should be possible for FR80 software to generate leader frames automatically.

(1) As long as the first leader frame is easily recognisable, there is no need for a specific cut mark.

(2) The registration frame has little relevance, and is no longer required as a separate item.

(3) There is no need for any trailer frames, since end of job can be detected by end of film or beginning of next job.

(4) It may be useful to include details of which machine and which package generated the output.

(5) The Laboratory name should appear somewhere (advert!).

A possible sequence could be:

Blank frame

Full frame, dark, picture.
The pattern chosen could also serve as some form of registration by issuing lines at different intensities, spot sizes, angles etc.
Operator settings for this job should be included on this frame if possible. Also ACL name.

Frame with username in large letters, together with jobname, date, time.

It is likely that different cameras will require different leaders. On hardcopy, for example, there is likely to be room for amalgamating the two picture frames. The username letter size need not be so large. However, two frames may well be required on 16 mm, to ensure that they are not missed.

gm

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 3

~~MOGS~~ now SMOG

issued by
F R A Hopgood

28 November 1974

DISTRIBUTION: RB AWB MFC AMF JRG
JWEL PMN DVR JMR RET

1. INTRODUCTION

MOGS (Mini Output Graphics System) is a simple and fast graphics package available on both the 1906A and 360/195. The aim has been to provide a minimal set of facilities so that the overhead on users' programs in terms of space should be as small as possible. It is likely that users of this package will, at some stage, wish to use more sophisticated graphics facilities and, consequently, this package will look at the user level, as though it is a subset of SPROGS. The main output device is the FR80 but other output devices can be defined as debugging aids.

2. INITIALISATION

MOGS is a set of FORTRAN routines that can be added to a users' program. Before any graphical output can be produced, the MOGS system must be initialised to define the one output form he wishes to use (only one is allowed in any program run). This is achieved by calling one of the following routines:

(1)	FRC16 :	combined camera, 16mm on FR80
(2)	FRC35 :	combined camera, 35mm on FR80
(3)	FRS16 :	16mm cine camera on FR80
(4)	FRMFCH :	microfiche camera on FR80
(5)	FRHARD :	hardcopy camera on FR80
(6)	TKTRON :	Tektronix 4010 storage tube
(7)	HPACK :	Hewlett-Packard pen plotter
(8)	LPGRAF :	lineprinter in pseudo-graphical form
(9)	DVALL(I):	the argument I indicated which of the above 8 possibilities is to be used.

The routine DVALL gives an alternative method of specifying the output device to be used. However, calling DVALL will cause the routines for all the devices to be loaded whereas calling the routines (1) to (8) will only cause the routines for the specified device to be loaded. The only advantage of DVALL is that it allows the user to choose the device he requires after the program has been loaded. If only a subset of the devices may be required by the user, the other device routines can be replaced by dummy ones by inserting calls to the following routines in the program:

(1)	NOFR :	FR80 routines for all devices
(2)	NOTK :	Tektronix 4010 routines
(3)	NOHP :	Hewlett-Packard routines
(4)	NOLP :	Lineprinter routines

For example, if the user wants to output to either the FR80 16mm cine camera or the Tektronix 4010, the initialisation calls could be:

```
    CALL DVALL(I)
    CALL NOHP
    CALL NOLP
```

The calls to the routines NOFR etc must appear in the program but need not be obeyed.

The initialisation routines set the coordinate system and other properties of the device specified.

3. OUTPUT DEVICES

The MOGS system can produce graphical output which will be plotted on a variety of devices depending on the initialisation routine that has been called. The hardware for each device has its own coordinate system for defining where lines and characters are to be placed. Normally, output is defined with respect to some raster that covers the complete plotting area of the device. For example, the FR80 divides up the complete plotting area into 16383 intervals in both the X and Y directions. A diagonal accross the complete plotting area from bottom-left to top-right would be drawn by specifying its starting position as (0,0) and finishing position as (16383, 16383). The devices own coordinate system is often referred to as 'raster coordinates'. The raster coordinates for the possible output devices are:

Device	XMIN	YMIN	XMAX	YMAX
FR80	0	0	16383	16383
Tektronix 4010	0	0	1023	779
Lineprinter	0	0	119	119
Hewlett-Packard	0	0	9999	9999

On most devices, the plotting area is equivalent to the area on which graphical output can be recorded. However, this is not the case with the FR80 where the square plotting area is the region on the cathode ray tube upon which lines and characters can be drawn. The graphical output is recorded by placing a camera above the tube. In general, the image that can be recorded on film is not the complete plotting area. This is obviously true in the case of 16mm and 35mm cine film as the frame size is not square. It is necessary, therefore, to differentiate between the complete plotting area and the visible plotting area. The latter varies from camera to camera and may even depend on the use to which the output is to be put. For example, the 16mm camera can record so that consecutive frames abut. However, if these frames are viewed using a cine projector, part of the image will not be seen as the standard cine image is defined so that there is a gap between consecutive frames.

4. COORDINATES

The most efficient way of generating graphical output is to define it in terms of the raster coordinates for the device. However, this is not always the most convenient for the user. There is usually a setural coordinate system for the output which reflects the problem. MOGS allows the user either to generate his graphical output in terms of raster coordinates or a coordinate system that he defines before generating any output. MOGS calls the coordinate system for the output device (raster coordinates) 'region 0' and the users' own coordinate system 'region 1'.

The user may define his coordinate system either in terms of the complete plotting area or just the visible part. The two instructions are:

```
    CALL LIMIT(XMIN,YMIN,XMAX,YMAX)

    CALL LIMITV(XMN,YMN,XMX,YMX)
```

These routines define the user's coordinates for the lower left and upper right hand corners of either the whole plotting area (LIMIT) or the visible part (LIMITV). The second routine is probably the more useful. Lines and characters may be drawn anywhere on the plotting area. However, attempts to draw outside the plotting area will cause incorrect graphical output to be produced. Full details of the visible part of the plotting area for each FR80 camera is given in Appendix I. The routine LIMITV gives the cine image for the 16mm and 35mm cameras and the abutment image for hardcopy.

The user may redefine his coordinate system any time during a program run by calling either LIMIT or LIMITV again. Part of his output can be defined in terms of the complete plotting area while another part is defined in terms of the visible part.

The system is initialised to expect output to be generated in terms of region 1. If output is to be defined in terms of raster coordinates, this region must be selected by:

```
    CALL REGION(0.0)
```

Once region 0 has been selected, all subsequent output will be defined in terms of raster coordinates until region 1 is selected by:

```
    CALL REGION(1.0)
```

If the user is defining some of his output in raster coordinates and some in his own coordinate system he must be careful to select region 1 before calling either LIMIT or LIMITV. The system is initialised with region 1 selected. A possible set of instructions is:

```
CALL STSPR
CALL FRS16
CALL REGION(0.0)
CALL VEC(3106.0,4578.0,13278.0,11806.0)
```

The output device is defined as the FR80 16mm cine camera and line drawing is defined in raster coordinates. The call to VEC will produce a diagonal line from one corner to the other of the 16mm cine frame. The same result could have been achieved by:

```
CALL STSPR
CALL FRS16
CALL LIMITV(0.0,0.0,4.0,3.0)
CALL VEC(0.0,0.0,4.0,3.0)
```

The coordinate system has been set up so that the unit lengths in the X and Y directions are approximately the same.

An equivalent result is obtained by defining the output in terms of the complete plotting area:

```
CALL STSPR
CALL FRS16
CALL LIMIT(0.0,0.0,1.0,1.0)
CALL VEC(0.1896,0.2794,0.8104,0.7206)
```

5. LINE DRAWING

Line drawing is defined with respect to the coordinate system currently being used. For a number of the line drawing routines, the line is drawn from the current plotting position which will be defined as (XPOS,YPOS). Each line drawing routine resets the current plotting position to the end point of the line. The possible routines with arguments are given below. All arguments are real.

```
SUBROUTINE VEC(X1,Y1,X2,Y2)
```

This draws a line from (X1,Y!) to (X2,Y2) and sets XPOS=X2,YPOS=Y2

```
SUBROUTINE TOXY(X,Y)
```

This draws a line from (XPOS,YPOS) to (X,Y) and sets XPOS=X, YPOS=Y

```
SUBROUTINE TODXY(DX,DY)
```

This draws a line from (XPOS,YPOS) to (XPOS+DX,YPOS+DY) and sets XPOS=XPOS+DX and YPOS=YPOS+DY.

```
SUBROUTINE TOX(X)
```

This draws a line from (XPOS,YPOS) to (X,YPOS) and sets XPOS=X.

SUBROUTINE TOY(Y)

This draws a line from (XPOS, YPOS) to (XPOS,Y) and sets YPOS=Y

SUBROUTINE TODX(DX)

This draws a line from (XPOS,YPOS) to (XPOS+DX,YPOS) and sets XPOS= XPOS+DX

SUBROUTINE TODY(DY)

This draws a line from (XPOS,YPOS) to (XPOS,YPOS+DY) and sets UPOS=1 YPOS+DY

SUBROUTINE SETXY(X,Y)

This sets the current plotting position to (X,Y)

SUBROUTINE SETX(X)

This sets the current plotting position to (X,YPOS)

SUBROUTINE SETY(Y)

This sets the current plotting position to (XPOS,Y)

SUBROUTINE UPDY(DX,DY)

This sets the current plotting position to (XPOS+DX,YPOS+DY)

SUBROUTINE UPDX(DX)

This sets the current plotting position to (XPOS+DX,YPOS)

SUBROUTINE UPDY(DY)

This sets the current plotting position to (XPOS,YPOS+DY)

6. CHARACTER OUTPUT

To output text and information, the following parameters must be defined:

(1) Current Typing Position : the coordinates of the lower left hand corner of the first character.

(2) Character Height : the height of each character; this will be constant for a string of text whereas the width may vary.

(3) Mono/proportional Spacing : text can be output so that the distance from the left of one character to the left of the next is equal for all characters. This is called mono-spacing. If the characters are allowed to have variable width with an equal spacing between each pair of characters (from the right of one to the left of the next), this is called.proportional spacing. On some devices, only mono-spacing is defined.

(4) Line Space : the distance between two lines of text measured from the bottom of one line to the bottom of the next.

(5; Character Rotation : although characters will normally be output on a horizontal line, it is possible, for some devices, to define characters which have been rotated.

All the character routines define lengths in terms of the currently selected coordinate system. The set of routines is defined below. The argument STR is a Hollierith text while all other arguments have their type indicated by the first letter.

SUBROUTINE STYPE(X,Y)

This defines the current typing position as the point (X,Y)

SUBROUTINE HTEXT(N,STR)

This outputs the N characters of STR so that they are adjacent and to the right of the characters output from the last call of HTEXT or HTYPE as long as there has been no call of STYPE in between. If STYPE has been called, the characters will be output at the current typing position. The text string may run over several lines. The character / is normally used to represent the newline character.

SUBROUTINE NLCHAR(STR)

The one-character string STR defines a character which will represent the newline character in subsequent calls of HTEXT. The routine NLCHAR may be called more than once in a program.

SUBROUTINE MONO(X)

If X has the value 0.0, mono-spacing is defined while X set to 1.0 will define proportional spacing.

SUBROUTINE CHSP(X)

This defines character spacing to be of length X. When characters are output using HTEXT, the value X is the distance from the left of one character to the left of the next if mono-spacing has been defined. If proportional spacing has been defined, it is the distance from the right of one character to the left of the next.

SUBROUTINE LNSP(Y)

This defines the distance between two lines of text from the bottom of one to the bottom of the next.

SUBROUTINE CHOCT(R)

This defines the character rotation to be used for subsequent calls of HTEXT. If R=0.0, no rotation is specified while for R values between 1.0 and 7.0 it defines a rotation of Rx45^{o} anti-clockwise. If no rotations is available on the selected output device, this routine will have no effect.

SUBROUTINE TYPNMB(A,XI,XJ)

The number A will be output as a text string at the current typing position or adjacent to the last string output by HTEXT depending on whether STYPE has been called in between (see HTEXT definition). The format for A is defined as follows:

XI=0,0,	E format, field width XJ+7
XJ=0.0,	I format, field width XI+1
XI,XJ≠0.0,	F format, XI places before decimal point and XJ places after. Field width XI+XJ+2.

SUBROUTINE CHHT(Y)

This routine defines the height of characters to be Y units.

SUBROUTINE HPLOT(R)

This routine outputs the character whose code number is R at the current plotting position (XPOS,YPOS). The character will be output with its lower left hand corner at the current plotting position. No change will be made to (XPOS,YPOS). The code numbers for the complete character set is given in Appendix II. The plotting dot, character number 196, is output so that the dot appears at the position (XPOS,YPOS).

SUBROUTINE HTYPE(R)

This outputs the character whose code number is R so that it is adjacent to and to the right of the characters output from the last call of HTEXT or HTYPE as long as there has been no call of STYPE in between. If STYPE has been called, the character will be output at the current typing position. The code numbers for the complete character set are given in Appendix II.

SUBROUTINE HPAGE

This routine advances the film and sets the current typing position to the top of the page. It is equivalent to:

```
CALL ADVFLM
CALL STYPE(XMN,YMX-Y)
```

where XMN,YMX are the argumnets to the last call of LIMIT or LIMITV and Y is the character height defined by CHHT.

SUBROUTINE HLINES(N)

This routine resets the current typing position to a position at the start of a line and N lines down from the last typing position defined by STYPE or HPAGE. If the last typing position was (SX,SY), the new typing position will be (XMN,SY-N*Y). The value of Y is the line space defined by the last call of LNSP. The value of XMN is defined by the last call of LIMIT or LIMITV.

7. COLOUR AND INTENSITY

Some devices, in particular the FR80 combined 16mm/35mm camera, can produce output at different intensity levels and in different colours. The line and character drawing routines described so far can be used to produce this output depending on the following setting procedures:

SUBROUTINE COLOUR(X)

The argument X specifies the colour of all subsequent line and character drawing until reset by another call of COLOUR. The binary digits of X define different colours and it is possible to use a value of X to specify more than one colour at a time.

Colour output on the FR80 is achieved by a subtractive filter system between the tube and camera. It is possible to place any combination of the three filters Cyan, Yellow and Magenta in the light path. These are specified by the following X values:

X=1.0	CYAN	(blue/green)
X=2.0	YELLOW	(green/red)
X=4.0	MAGENTA	(red/blue)
X=8.0	CLEAR	(blue/green/red)

Each filter other than CLEAR allows two of the three primary colours to pass through. By specifying two filters in the path, it is possible to produce the primary colours themselves. For example:

X=3.0	GREEN
X=5.0	BLUE
X=6.0	RED

The Hewlett-Packard pen plotter can be used with different colour pens. Each binary digit of X is used to indicate a different colour. If a pen change is required, a message will be printed to the operator requesting that pen number I is mounted where I=1,2,3 etc coincides with X values 1.0, 2.0, 4.0 etc.

No colour output is available on either the Tektronix 4010 or the line-printer.

SUBROUTINE INTNST(Y)

Lines and characters may be drawn at different intensity levels on the FR80. The real argument Y which has an integer value in the range 0 to 255 specifies the intensity for all subsequent drawing until a further call of INTNST occurs.

Due to the characteristics of the light output from the FR80 tube, it is likely that different intensity levels may be required for different colours to get a reasonable balance.

No intensity variation is available on the other output devices.

SUBROUTINE COLGL(Z)

This subroutine allows the user to select only part of his graphical output for drawing. If Z is set to -1.0, the system assumes that output is in black and white. Any calls of the subroutine COLOUR will be ignored. If Z is set to any other value, only those colours whose binary digits appear both in Z and X (the argument of COLOUR) will be output. For example:

```
CALL COLGL(1.0)
CALL COLOUR(3.0)
CALL VEC(1.0,1.0,2.0,2.0)
CALL COLOUR(2.0)
CALL VEC(2.0,2.0,3.0,3.0)
CALL COLOUR(1.0)
CALL VEC(3.0,3.0,4.0,4.0)
```

Without the call to COLGL, this program is designed to draw green, yellow and cyan lines in that order. The setting produced by COLGL will cause only those lines drawn through the CYAN filter to be output. The main use of this facility is when the colour output is to be generated photographically from black and white originals. By running the same program three times with an initial call to COLGL having arguments 1.0, 2.0 and 4.0 respectively, three black and white films suitable for photographic processing can be produced. The subroutine COLGL can also be used for debugging purposes.

When generating colour output on the FR80, the subroutine COLGL should be called initially with argument Z set to 15.0. The MOGS system is initialised with Z set to -1.0.

8. MISCELLANEOUS

The following routines complete the set of MOGS facilities.

SUBROUTINE ADVFLM

This routine completes the current frame by advancing the film in a camera or providing some STOP command. Subsequent output will be plotted on a separate frame.

SUBROUTINE ENDSPR

This routine should be called once the graphical output has all been produced. Failing to call ENDSPR may result in the graphical output not being generated.

SUBROUTINE FLEMK

This routine outputs all the graphical information up to this point and then outputs a filemark.

SUBROUTINE RDLIB(PK,NM)

This routine can be used to read standard background graphical output into the program from a file generated by another package. Both arguments are Hollerith text. The first defines the name of the package while the second gives the filename. For example:

```
CALL RDLIB('SPROGS',MYBKGD)
```

The background information is stored in the common block whose name is MOGSLB. This has a minimal size defined in the basic MOGS system. If RDLIB is used, the program must contain a COMMON statement somewhere allocating sufficient store for the background information. For example:

```
COMMON/MOGSLB/DUMMY(5000)
```

The name used in the array declaration can be anything.

The subroutine RDLIB may be called more than once. Each background currently overwrites the earlier one.

SUBROUTINE DRAWNM(MYBKGD)

This routine outputs the background information which has been read into the program by a previous call of RDLIB. The argument to this routine should be the same as the second argument of RDLIB. The routine DRAWNM may be called many times.

APPENDIX I

CAMERA IMAGES

1. FR80

The complete FR80 plotting area has coordinates 0 to 16383 in both the X and Y directions. Each camera is capable of photographing only part of the plotting area - the aperture plate ensures that plotting outside its bounds is not recorded. However, if output is produced over the complete area allowed by the plate, the images on consecutive frames overlap. Consequently, it is usual to restrict plotting to at least an area such that two consecutive frames abut. For 16mm and 35mm cine film, the frame area is even smaller with a gap between consecutive frames. Where appropriate, the three areas given above are defined.

Camera	XWIDTH	YWIDTH	XMIN	YMIN	XMAX	YMAX
8020 combined 35mm						
Aperture	16383+	13376	0	1504	16383	14880
Abutment	16383	12510	0	1937	16383	14447
Cine	16383	12290	0	2047	16383	24337
8020 combined 16mm						
Abutment	6938	5016	4723	5684	11661	10700
Cine	6938	4930	4723	5727	11661	10657
8021 16mm cine						
Aperture	10172	8579	3106	3902	13278	12481
Abutment	10172	7353	3106	4515	13278	11868
Cine	10172	7228	3106	4578	13278	11806
8025 105mm microfiche						
Image	13530	16383	1427	0	14957	16383
8060 hardcopy						
Aperture	16383+	12547	0	1918	16383	14465
Abutment	16383	11866	0	2259	16383	14125

If the limits of the complete plotting area are defined as (0,0) and (1,1), the visible parts of the area for each device are:

Camera	XMN	YMN	XMX	YMX	Ratio Y/X
8020 combined 16mm(cine)	0.2883	0.3495	0.7117	0.6505	0.7106
8020 combined 35mm(cine)	0	0.1249	1	0.8751	0.7501
8021 16mm(cine)	0.1896	0.2794	0.8104	0.7206	0.7106
8025 105mm microfiche	0.0871	0	0.9129	1	1.2109
8060 hardcopy	0	0.1379	1	0.8621	0.7242

2. TEKTRONIX 4010

On the 1906A, the visible part of the plotting area is defined as between (224,0) and (1023,779). This stops GEORGE responses, which appear on the left of the screen, from obscuring the graphical image. The visible part of the plotting area may be defined differently on the 360/195.

3. HEWLETT-PACKARD

The visible part of the Hewlett-Packard plotting area is identical to the complete plotting area.

4. LINEPRINTER

The visible part of the lineprinter area is defined as (0,0) to (60,60). The whole plotting area is visible if output is directed to the lineprinter while only the smaller area is visible if the output is listed on a teletype.

APPENDIX II

MOGS CHARACTER CODE NUMBERS

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
OCT	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
0X 00X	NU	α	β	γ	δ	ϵ	η	θ	λ	μ	ν	π	σ	Σ	φ	ω
1X 02X	Ω	'	‾	•	/	±	ℓ	∂	□	⌑	Δ	←	→	↑	↓	\
2X 04X		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3X 06X	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4X 10X	'	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5X 12X	P	Q	R	S	T	U	V	W	X	Y	Z	[	~	]	^	_
6X 14X	@	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7X 16X	p	q	r	s	t	u	v	w	x	y	z	{	‾	}	|	▯
8X 20X	CO	CO	UN	CO	CO	UN	UN	UN	UN	CO	CO	CO	CO	CO	CO	CO
9X 22X	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN
AX 24X	✓	‡	‡	⇔	⌢	ŏ	δ	∠	≠	≥	⋎	ō	-	—	∼	-
BX 26X	^	℞	%	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN
CX 30X	·	•	⊕	◎	.	UN	UN	UN	∘	∘	○	○	UN	UN	UN	UN
DX 32X	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN
EX 34X	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN
FX 36X	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN

NU-NULL CO-CONTROL UN-UNSPECIFIE

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 4

Some Figures on 1906A Spool

issued by
A W Burraston
13 December 1974

DISTRIBUTION: FRAH RB MFC AHF JRG JWEL
PMN DVR JMR RET PAD

DATE	No of JOBS PER DAY	No of RUNS of OPS-PLOT PER DAY	AVERAGE No of JOBS/RUN	MAX No of JOBS ON SPOOL AT ANY ONE TIME
4/11/74	5	2	2·5	4
5/11/74	6	3	2·0	3
6/11/74	1	1	1·0	1
7/11/74	2	1	2·0	2
11/11/74	5	1	5·0	5
12/11/74	4	1	4·0	4
13/11/74	2	1	2·0	2
14/11/74	2	1	2·0	2
15/11/74	6	2	3·0	5
18/11/74	10	4	2·5	3
19/11/74	13	5	2·6	5
20/11/74	9	3	3·0	4
21/11/74	21	5	4·2	5
22/11/74	3	2	1·5	2
26/11/74	5	2	2·5	3
27/11/74	3	2	1·5	2
28/11/74	5	3	1·7	3
29/11/74	7	4	1·7	2
3/12/74	11	3	3·7	4
4/12/74	10	6	1·7	3
6/12/74	2	1	2·0	2
AVERAGES	6·3	2·5	2·5	3·1

OCCASIONS OF MORE THAN 5 JOBS ON THE SPOOL AT ONE TIME

DATE	No OF JOBS	TIME LAST EMPTIED	TIME EMPTIED	COMMENT
2/4/74	6	15:31	22:50	
16/5/74	6	07:00	23:28	
19/8/74	6	11:02	16:35	
17/9/74	6	07:58	16:01	
15/8/74	12	10:23	22:35	

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 5

Visit to Leeds Computer Centre
9-13 December 1974

issued by
R Brandwood

20 December 1974

DISTRIBUTION: FRAH AWB MFC PAD AHF JRG
JWEL PMN DVR JMR RET (AS)

1. INTRODUCTION

The purpose of the visit was to gain some operating experience of the Leeds COMP80, and generally observe how they organised the workflow, manning of the COMP80, and III 5010 film processor.

2. HARDWARE

2.1 COMP80

16K Store
256K Disc
1 7-track tape deck
1 9-track tape deck
Character Generator
Vector Generator
High speed page print feature
Model 8023 35mm Unsprocketed Camera
8025 105mm Microfiche Camera

2.2 Processor

5010 Film Processor
Hunts Chemicals used (supplied via III)

2.3 Film Types

35mm	Kodak Dacomatic A
	Ilford Ilfodata Com
105mm	Kodak Dacomatic A
	Fuji Microfilm Neg

3. OPERATIONS

3.1 General Manning

Leeds operate a 3-shift system, very similar to the one used at ACL. A shift consists of 20 people whose tasks involve manning the System 4/70 Receipt and Dispatch, Job Assembly, Number Information Service (NIS) and COMP80.

3.2 Manning COMP80

When the machine was installed, approximately 18 months ago, one operator was trained; he gained experience and operated the COMP80 on day shift. The service provided and quality of the output was very good. Two more operators were trained on-site and a 2-shift system introduced - the senior operator then moved onto greener pasture - the service and quality was still very good. Staff changes and introduction of 3-shift working left them short of operators; it was then decided that the manning of the COMP80 was the responsibility of the shift. The quality and service has since declined. They are conscious of this and are trying to rectify it.

3.3 Manning 5010 processor

The processor is manned by photographic grades during the day, no processing is done outside this period. Loading of cassettes is done during the day; suffient quantity is normally left available for the evening and night shift.

4. QAULITY CONTROL

4.1 Cameras and COMP80

A detailed cleaning procedure is layed down for each camera and the tape decks; it is carried out at the start of every shift with the aid of a checklist and recorded when each item is completed.

4.2 5010 Processor

Each day, checks of chemical levels are made and topping up is done as required. All rollers and squeezers are cleaned. The exterior is wiped with a damp sponge to remove any chemicals spilt - (to prevent corrosion). Every 3 weeks the processor is drained, cleaned and refilled with fresh chemicals.

4.3 Film

Each day a test strip is produced and processed before any other work is carried out. The max/min density of the film is established using a MACBETH TD-504 densitometer; the reading must lie within certain bounds before processing can continue. The film is then viewed on a CAPS Page Reproducer, screen size is up to 30" x 20", or NCR microfiche viewer, to check on the focus. A 60X or 100X microscope is also available.

Examples of acceptable and unacceptable film is kept, suitably labelled, to use as a guide.

5. WORK LOAD

The majority of the operators' work was done by the evening and night shift. The time for each job varied but normally they were in excess of 15 mins; it was quite common for a job to have 5-6 data tapes. The work load did not vary a lot from day to day. During the day shift, operations had first call on the machine, but it was mainly used by the Software Support to write new software or from designs for customers.

6. MISCELLANEOUS

6.1 Points of Interest

(1) Very little operating experience was gained during the week, but a better understanding of the general system was achieved.

(2) Selections and training of operators needs to be carefully considered and Leeds experience taken into account.

(3) Layout of the area requires recognition of various factors - space, flow of people and tapes, access to darkroom, storage area for cameras and camera dolly (latter is not highly manoeuverable).

(4) A safety feature incorporated on the Leeds tape decks meant the device powered off when the door was opened; it also meant they could not load a tape whilst the other deck was being used because it occasionally stopped the processor!

(5) The chemicals and some film stock used are purchased through III, it costs Leeds just as much in transport charges as it does for the goods. They are in the process of tendering UK supplies for film and chemicals. ACL should not have this problem.

(6) The routine cleaning and daily test, plus equipment used for quality control are worth considering for implementation.

(7) A modification to the "Thermostatic Control Valve Assembly" will allow the chemicals to be mixed with 'Filtered water' rather than water straight from the mains. AS is arranging this.

(8) Leeds suffered from drop-outs on the operation side, cameras were sometimes loaded with the lens cover left on, the wrong control disc loaded with the microfiche camera, wrong programs loaded.

(9) Friday morning was used for maintenance, this is something we have to arrange with the engineer.

7. SUMMARY

Although little operating experience was gained on the visit, a better understanding of the system was achieved. It appeared that it was small items that caused most concern, these should not be overlooked. ACL should be able to learn from Leeds mistakes.

bb

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 6

FRESCO Output

issued by
F R A Hopgood

31 December 1974

DISTRIBUTION: RB AWB MFC PAD AHF JRG
JWEL PMN DVR JMR RET

The following points have been noticed on a preliminary examination of FRESCO:

(1) No use is made of the ability to input operator commands that change the form of the output.

(2) The High Speed Page Print feature is not used and only mono-spaced characters are output. Text output does go across page boundaries. Even when the output from the Enter Type Mode command is terminated by a 203 control character, it appears that a NOP is output to leave type mode.

(3) The font used in the package has a character height varying from 15 to 370 raster positions. This is larger than any of the fonts available from III.

(4) Neither the Start Job nor End Job commands are used.

(5) All character size changes are controlled by the Set Character Size order even though the height has been specified in raster positions. A conversion table gives the equivalent size number. This means that operator commands are likely to give characters of the wrong size if the image is expanded or reduced.

(6) Film advances are made using the Incremental Film Advance command and specifying the number of pull-downs.

(7) Print statements do not allow more than 256 characters per command. The reason for this is unknown.

(8) The Vector Family command is not used for graticules.

(9) SD4020 overstriking has been converted into a mixture of intensity and spot size changing. A parameter between 1 and 10 causes the following settings:

Parameter	Intensity	Spot Size
1	1	1
2	2	1
3	3	1
4	4	2
5	5	3
6	6	4
7	7	5
8	8	6
9	8	7
10	8	8

(10) The Control Interrupt command is used to output the time to the teletype, send messages to the operator, and generate pause control levels.

(11) The following commands are not used:

Frame Advance
End Job
Start Job
Strip Chart Mode
Repeat
Picture
Enter Justified-Type Mode
Output Page
Select Colour
Plot Current Point
Draw Vector Family
Set X and Y Offsets
Select Character Height
Define Character Set
Output FR80 Supplied Variable
Fiche Title

gm

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 7

Proposal for Camera Schedule

issued by
R Brandwood
13 January 1975

DISTRIBUTION:	FRAH	AWB	MFC	PAD	AHF	JRG
	JWEL	PMN	DVR	JMR	RET	

Because only one camera can be used at any time, and camera changing is not a 2-second job, it is necessary to arrange a schedule. This should allow users to know when they can expect output and when they cannot. It will also allow the operators to prepare the necessary requirements.

1. POINTS TO CONSIDER

1.1 Types of Cameras

TYPE		MODEL
Hardcopy		8060
35mm	B&W	8020
35mm	Colour	8020
16mm	B&W	8020
16mm	Colour	8020
16mm	B&W	8021
105mm	Microfiche	8032

The ordering of 35mm/16mm, B&W and Colour on Models 8020 and 8021 really depends on:

(1) whether present SD4020 cassettes are compatible, or

(2) whether changing from 35mm to 16mm on Model 8020 is easier whilst camera is bolted to light-source, or

(3) whether Model 8021 should be loaded whilst changes are done.

Answers probably are (1) Yes (2) Keep 8020 on until the camera is finished with.

1.2 Expected Workload

This is difficult to forecast, but indications are available from studying present workload and replies to FR80 questionnaires. The added facilities may encourage users to experiment for a while.

Appendix 1	gives a breakdown of the size of jobs run over a 4-month period.
Appendix 2	times when types of jobs run on SD4020.
Appendix 3	answers to FR80 questionnaires.

1.3 Processing Speed

To overlap activities and speed up throughput, film processing has to be taken into account. Each type of film is processed at a different speed, but each has a length of leader on the front and back, approximately 90 ft each end.

SPEEDS

TYPE	FT/MIN	OVERHEAD on LEADER	FRAMES/FT
105mm	9†*	20 mins	400 to 800
35mm	14†*	14 mins	16
16mm	18†	11 mins	40

*speeds used by LEEDS CC, used for assessment
†estimated

Appendix 4 Graph of time taken to process varying length of film.

Hardcopy is processed separately, approximately 20 frames/min.

1.4 Users

It is necessary to consider such things as the user's location, time of last post, courier service to RL, and the fact that the internal user should also be given a good turnround.

1.5 Time to Generate

This is like answering the question "how long is a piece of string?"

A fairly accurate assessment in order of speed would be microfiche, 16mm and 35mm B&W, hardcopy, Colour.

2. CONCLUSION

2.1 Interest

The main interest is in microfiche and colour, with grey levels only a little way behind. Strip charts have only half as much interest as the others.

2.2 Hardcopy

The majority of users use hardcopy producing less than 50 frames and most wish to remain with hardcopy. Those who considered changing opted for microfiche, then 35mm. The two 100 frames+ users wished to remain with hardcopy. The 300 frames+ user (Appendix 1) has not returned a questionnaire.

2.3 35mm

Most users here require less than 100 frames, the 500 frames+ users are :GPRA02 whose work is mainly run in the evening/night shift by arrangement and a 360 user who has an interest in microfiche.

2.4 16mm

Most users here require 100 frames+, with the bulk requiring 300-1000 frames. Two 1000 frames+ users showed an interest in microfiche.

2.5 Film Processing

Microfiche has the largest overhead but is faster for anything over 7200 frames (approximately 400K lines); 35mm is the slowest for anything over 1000 frames. A full cassette of 16mm would take 35 mins to process including leader.

3. ORDERING OF CAMERAS

To allow the 5010 processor to be used as effectively as possible, the first camera should be of a film type, the obvious choice being the microfiche camera because of (1) the overhead involved, and (2) the interest shown in microfiche. This should then be followed by the hardcopy camera; this will then allow plenty of time for the 5010 to be cleared before any 16 and 35mm is generated and allow the hardcopy processing to overlap with any film processing. The combined 16/35mm camera should then be used, the order really being decided by the operator, but suggested B&W, then colour on one size, then change to the other. This will allow B&W processing to be done whilst colour and the next B&W is being generated on the FR80. This then leaves 8020 camera to finish with; the processing of this can be done whilst the next camera is being used.

4. COURIER SERVICE

At present tapes from RL arrive twice a day at 0900 hrs and 1600 hrs. If a good turnround is to be given, especially for microfiche, then these need to be reviewed. It would be ideal if the times could be 0930hrs, 1400 hrs, 1600 hrs, 2000 hrs and 0500 hrs, as this would allow for all work received by 0500 hrs to be returned by 0930 hrs.

5. PRIME SHIFT SCHEDULE

0900	1000	Sys Dev
1000	1020	Microfiche
1020	1110	Hardcopy
1110	1240	Combined 35/16 - B&W and Colour
1240	1310	16mm - 8021
1310	1330	Not scheduled
1330	1430	Hardcopy
1430	1450	Microfiche
1450	1530	Combined 35/16 - B&W only
1530	1600	16mm - 8021
1600	1630	Hardcopy
1630	1800	Sys Dev - if required

No schedule has been attempted for the evening or night shift, as one operator will have processing of B&W, and maybe some colour, as well as the FR80 operating to contend with. Priority here will be given to larger jobs and to ensure a completion of microfiche work by 0900 hrs.

APPENDIX 1

BREAKDOWN OF SIZE OF JOBS

HARDCOPY

FRAMES	OCTOBER	SEPTEMBER	AUGUST	JULY
1 - 25	243	412	421	508
26 - 50	58	88	65	62
51 - 100	23	28	22	24
101 - 300	15	14	6	7
301 - 100	2	1	1	6
TOTAL	341	543	516	607
35mm				
1 - 25	22	38	72	67
26 - 50	25	50	18	10
51 - 100	3	4	9	9
101 - 300	3	7	15	24
301 - 500	0	2	12	13
501 - 1000	8	11	28	48
1000+	9	10	6	2
16mm				
1 - 25	14	7	4	9
26 - 50	2	6	0	5
51 - 100	13	5	2	4
101 - 300	38	32	7	24
301 - 500	43	41	34	59
501 - 1000	58	66	32	134
1000+	10	7	21	24

APPENDIX 2

NUMBER OF JOBS RUN EACH HOUR

	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	HRS	
	0800	0900	1000	1100	1200	1300	1400	1500	1600	
JULY 1974										
HARDCOPY	9	74	93	137	11	95	40	53	85	
35mm	0	17	18	33	1	14	18	12	26	*
16mm	1	12	22	43	10	41	48	22	14	*
AUGUST 1974										
HARDCOPY	0	114	106	80	9	77	42	41	28	
35mm	0	16	23	9	2	20	26	8	20	*
16mm	0	2	3	16	3	15	18	4	5	*
SEPTEMBER 1974										
HARDCOPY	2	163	47	110	23	39	25	65	41	
35mm	0	17	2	13	2	11	15	7	10	
16mm	0	29	21	33	4	32	20	0	6	
OCTOBER 1974										
HARDCOPY	0	71	43	62	55	34	32	30	13	
35mm	0	16	11	10	6	9	7	4	3	
16mm	0	17	23	12	9	20	16	10	17	*

*Processing continued during evening/night shift on some occasions

APPENDIX 3

RETURNS FROM 360 USERS = 34

Interested = 25 Not Interested = 7 Internal = 2

Interested in:

Microfiche	Grey Levels	Colour	Strip Charts
14	8	14	7

If hardcopy slowest change to:

Microfiche	35mm	16mm	Stay with hardcopy
5	1	2	12

Present Usage:

Type	Number of Frames					
	1	10	50	100	500	1000
HARDCOPY	4	12	4			
16mm			1	2		2
35mm			2			1

RETURNS FROM 1906A USERS

Interested = 28 Not Interested = 8 Internal = 27

Interested in:

Microfiche	Grey Levels	Colour	Strip Charts
18	17	20	6

If Hardcopy slowest change to:

Microfiche	35mm	16mm	Stay with hardcopy
10	9	2	16

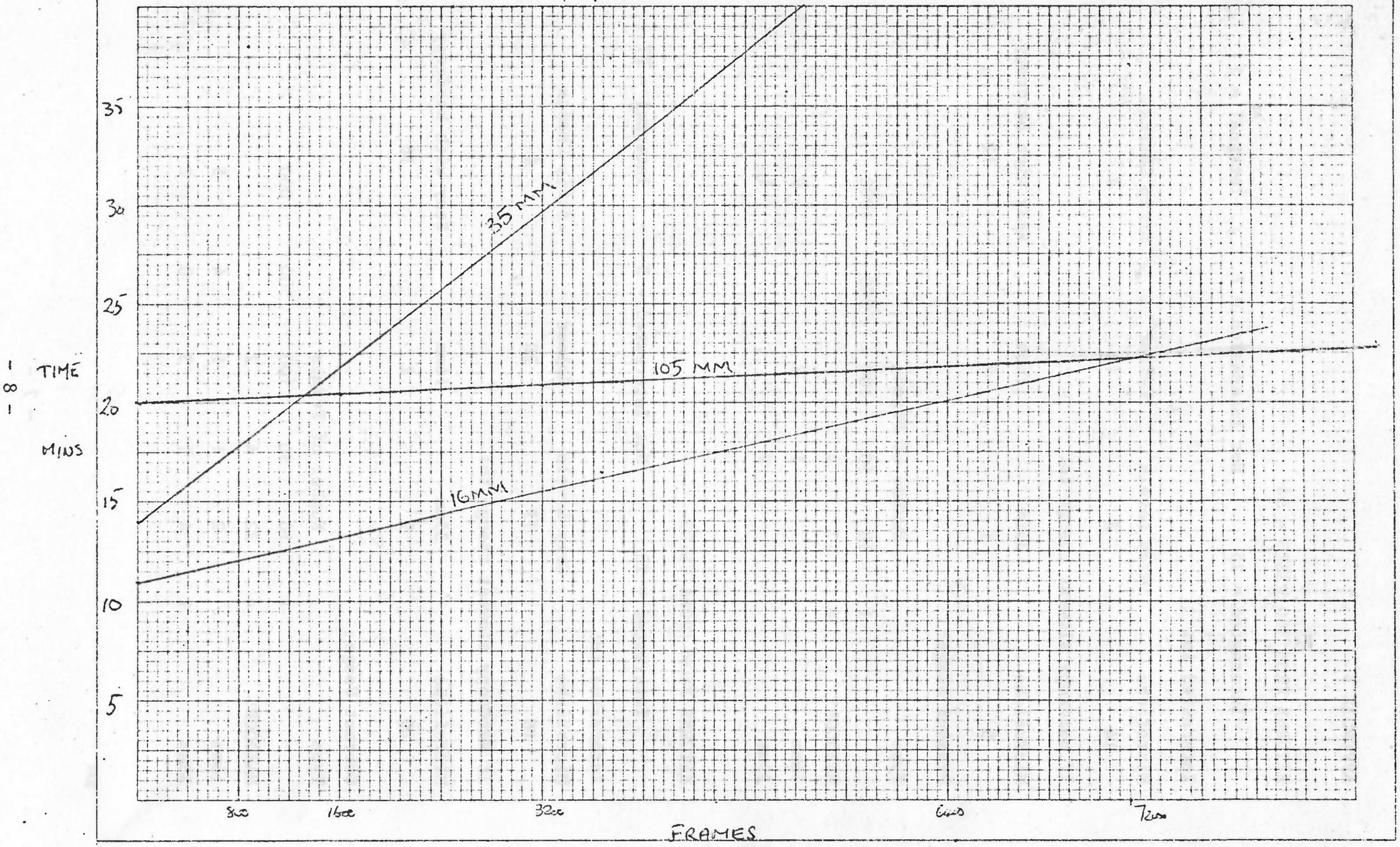
Current Usage:

Type	Number of Frames					
	1	10	50	100	500	1000
HARDCOPY	4	22	4	2		
16mm		2	1	1	4	5
35mm	1	4	1	3	0	2

gm

PROCESSING TIME

APPENDIX 4



SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 8

SD4020 tape analysis

issued by
D V Ralphs

15 January 1975

DISTRIBUTION:	FRAH	RB	AWB	MFC	PAD	AHF
	JRG	JWEL	PMN	JMR	RET	

1. INTRODUCTION

Three types of output were analysed to give an overall view of the savings, if any, in the number of 18-bit words used for an SD4020 tape and the number required to produce the same output on an FR80 tape.

1.1 The SCFOR 'Cut' Frame

This frame used a total of 810 SD4020 words whereas it would only require about 120 words on an FR80 tape. The major saving is due to the SCFOR package outputting the 'Cut' five times, whereas the FR80 would (hopefully!) only need a change in intensity.

1.2 One Block from an Alan Kitching Job

To draw 11 lines, the SD4020 required 334 words whereas the FR80 would only need 40 words.

1.3 A Complete Frame, a Graph by Stu Smith

The SD4020 required 1278 words compared with 698 on the FR80. See if you can spot which of the character strings is software! Hardware character strings on the SD4020 would use significantly fewer words but the SCFOR package appears to generate a lot of redundant code. The vector drawing produces a saving of about 40% and character drawing saves about 50% using the FR80 tape format.

1.4 Conclusions

Major savings occur with the FR80's ability to draw long vectors and its greater range of character sizes (FR80 Fonts). Savings occur with the vector family orders as well. A more detailed breakdown of the results is given overleaf.

ITEM	No. of	FR80 Words			SD4020 Words	COMMENTS
		No. x lw	Extra	Total	Total	
SCFOR CUT FRAME						
Lines in X Direction	13	13 x 3		39	50	SD4020 tape had 5 copies due to overstriking
Lines in Y Direction	8	8 x 3		24	68	
Lines in X + Y	12	12 x 4		48	24	
Dashed Lines in X	1	1 x 3	+ 4	7	20	
Total				118	162	
1 BLOCK ALAN KITCHING						
Lines in X Direction	4	4 x 3		12	136	
Lines in X + Y	7	7 x 4		28	198	
Total				40	334	
STU. SMITH GRAPH						
Lines in X Direction	2	2 x 3		6	52	
Lines in Y Direction	2	2 x 3		6	40	
Connected Lines in X	12	12 x 1	+2	14	24	Extra 2 words posn. beam Overstriking!
Connected Lines in X + Y	106	106 x 2		212	212	
Vertical Family	4	4 x 6		24	72	
Horizontal Family	4	4 x 6		24	72	
Char. String + Terminator	1	3	+ 5	8	14	Apart from * no such saving should have occurred. e.g. ⊛ only 16 SD4020 words were needed.
"	1	2	+ 5	7	8	
"	1	4	+ 5	9	14	
"	1	16	+ 5	21	212 *	
"	1	18	+ 5	23	20	
"	1	21	+ 5	26	22	
"	1	12	+ 5	17	18	
"	1	12	+ 5	17	18	
"	1	21	+ 5	26	22	
"	1	21	+ 5	26	22 ⊛	
"	1	37	+ 5	42	34	
Axis Labelling	9	9 x 5	+ 3	48	140	
"	9	9 x 5	+ 3	48	128	
Single Characters	17	17 x 4	+ 2	70	36	
"	8	8 x 4	+ 2	34	98	
Total				698	1278	

bb

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 9

Viewing of Jobs on FR80 SPOOL

issued by
A W Burraston

23 January 1975

DISTRIBUTION:						
	FRAH	RB	AWB	MFC	PAD	AHF
	JRG	JWEL	PMN	DVR	JMR	RET

The process will be in two parts.

1. PRODUCTION OF A NUMBER OF SMALL FILES EACH CONTAINING 1 FRAME (Background job)

The maximum number of such files produced may be controlled from within the job. Each file will have a file generation number equal to the frame number which it represents and a language code /FRME (files are all called SPOOLFILE).

The parameter string for the job may take two main forms:

(1) jobname,params - the first occurrence of jobname on the spool

(2) n,params - the nth 'file' on the spool is processed.

In each case the job will check for correct ownership of the job.

Those parameters which have been grouped together above by the general name 'params' specify which frames are required. For example:

```
FRAMES 6-10,FRAMES15-16,FRAMES20,FRAMES25-END
```

This would produce frames 6,7,8,9,10,15,16,20,25,26 .....

2. AFTER PART (1) ABOVE HAS BEEN COMPLETED

The user may now look at the frames he has produced via a Tektronix. He could issue the command:

```
LF SPOOLFILE(n/FRME)
```

which would produce the same result as the macro suggested below. The listfile is, however, tedious to keep typing in and a macro should still be provided for the erasing of all these files when the user has finished with them.

Therefore, a macro named FRAME is suggested:

```
FRAME  n,E     -   Lists the file SPOOLFILE(n/FRAME) and erases
                   when finished

FRAME  n       -   Lists the file SPOOLFILE(n/FRAME)

FRAME  ERASE -     Erases old SPOOLFILEs
```

gm

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 10

Proposals concerning the operation of
the FR80/1906A spooling system

issued by
J R Gallop

5 February 1975

DISTRIBUTION: FRAH RB AWB MFC PAD AHF
JRG JWEL PMN DVR JMR RET

The process of operating the spooling system can be considered in stages.

1. INITIATING THE REMOVAL OF JOBS FROM DISC TO TAPE

This will be under the control of the 1906A operators, usually in consultation with the FR80 operators. The following commands should be available to the operators eventually:

(a) DO FR80, *cameracode*

(b) DO FR80FU

(c) DO FR80CP

Format (a) transfers all ready jobs which require the FR80 facilities specified by *cameracode*. The possible *cameracodes* are as follows:

CL16	Combined camera	colour/16mm film
BW16	Combined camera	black and white/16mm film
CL35	Combined camera	colour/35mm film
BW35	Combined camera	black and white film
CINE	16mm cine camera	
FICH	Microfiche camera	
HC	Hardcopy camera	1-up or many-up jobs together on one tape
HC1	Hardcopy camera	1-up jobs only (maybe unnecessary)
HC4	Hardcopy camera	many-up jobs only (in order to exclude 1-up jobs from the prime shift if necessary)
ALL	All jobs in the spoolfile are transferred to tape, except those which are to be retained for viewing purposes. HC1-type and HC4-type jobs will be put onto one tape.	

Format (b) determines which camera has the most ready jobs in the spoolfile and outputs those jobs. Camera HC1 and HC4 are treated as separate cameras. The command is intended to be useful when the spoolfile becomes full or nearly full. The graphics packages will, when the spoolfile is full, display on the operators' console:

```
     ************************************
     FR80 SPOOLFILE IS FULL
     PLEASE TYPE 'DO FR80FU'
     ************************************
```

followed by the question:

```
     HAVE YOU TYPED 'DO FR80FU'
```

If the spoolfile is nearly full (*x*>(maximum-number-of-jobs minus 5) or *x*>15 where *x* is the number of jobs), each graphics package trying to add to the spoolfile will display on the operators' console:

```
     ************************************
     x JOBS IN FR80 SPOOLFILE
     ASK FR80 OPS OR 'DO FR80FU'
     ************************************
```

In this case, the 1906A operators can either startup 'DO FR80FU' or consult the FR80 operator to discover which camera is scheduled next on the FR80. The operators will be able to query the state of the spool for each camera (Section 2), to help in making a decision.

Format (c) should be used if it is suspected that the spoolfile is corrupted; a graphics package will inform the operators of this if one of a number of integrity checks fails. The macro FR80CP (CP stands for corrupt) will transfer all jobs from the spoolfile to tape, whether or not they have been marked as being "to-be-retained". The job will print out vital blocks of the disc to allow system programmers to examine the file. The job will then initialise the file and check the control blocks. If there proves to be a fault, the job will inform the operators of this and steps will have to be taken to have the disc pack tested and to divert subsequent graphics output. The diversion of graphic output can be achieved in one of two ways; either each macro (3) must be changed when the disc fault is discovered or the spoolfile specification should be changed so that it can itself indicate "known to be unusable" (for instance, word 18 of the File Control Block could be 0 if file OK and 1 if known to be unusable). This second solution will only be effective if the disc fails in such a way that the first block can be read.

2. QUERYING THE STATE OF THE DISC

Either the 1906A operators or the FR80 operators may wish to know how many jobs there are in the spoolfile or how many jobs for each camera.

The format of the enquiry request should be:

(a) DO FR80,STATE
(b) DO FR80,STATE,*cameracode*

Format (a) prints out a summary for the whole file:

```
m JOBS IN SPOOLFILE FOR FR80
n JOBS READY TO BE OUTPUT
```

"Ready to be output" means either that no delay is required or that the delay has expired.

In format (b), the *cameracode* must be one of the codes in the list in Section 1. If it is not ALL, then information is printed for the specified camera, for example:

m JOBS IN SPOOLFILE FOR CINE CAMERA
n JOBS READY TO BE OUTPUT

If the *cameracode* is ALL, the information is printed for each camera thus:

m JOBS IN SPOOLFILE FOR FR80
n JOBS READY TO BE OUTPUT
CL16=... BW16=... CL35=... BW35=...
etc

3. INFORMATION PROVIDED FOR THE FR80 OPERATORS:

The readout job will produce a piece of output for the FR80 operators with information about each graphics job. Before the first job, the output should include:

- a top page to ensure that the printout reaches the FR80 operators: "This listing is for the FR80 operators"
- date and time of each job
- name and serial number of magnetic tape
- camera
- "First user's data is preceded by a 1906A header record and 1 filemark"

For each job, the output should include:

- username and jobname and data base details
- date and time job started on the 1906A
- which graphics package
- number of FR80 records transferred to tape
- number of frames
- number of feet (in case of 35mm and 16mm)
- number of subframes (if different)
- which FR80 program
- number of filemarks

After the last job, the output should contain:

- "end of data for camera ...."
- number of jobs
- "last user's data is followed by 2 filemarks"
- totals (FR80 records, frames, feet, subframes)

This describes standard output. If there has been an abnormal event (for example, incomplete data), this will be noted in the information for the job in question, and also the bottom of the whole print-out will include:

"warning: problems with this run ... see above for details"

4. DISPLAYS TO 1906A OPERATORS WHILE TRANSFERRING TO TAPE

The display requesting a tape to be mounted will be put out by the ONLINE command; the job will first check that a tape is required.

At the end of a run for a particular camera, the display will be:

TAPE .... IS FOR FR80, CAMERA ....

At the end of the whole run, an extra display will be:

m JOBS REMAIN IN SPOOLFILE

gm

RWW

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 11

FR80 Spool Tapes

issued by
F R A Hopgood

17 February 1975

DISTRIBUTION: FRAH RB AWB MFC PAD AHF JRG
JWEL PMN DVR JMR WDS RET (PEB EHW)

1. Spool tapes for the FR80 will normally be 7-track although the read-out program should have the ability to switch to 9-track tapes in emergencies or if an operator on the FR80 requires it (9-track not available initially).

2. The read-out program will use a pool tape which it will keep in a directory under :GSIN00. As the read-out job will run under :MANAGER, it must be possible for :MANAGER to take tapes from the pool for other users. PEB will check this. If this is not possible, the tapes must reside under :MANAGER. The advantage of having a user directory, either :GSIN00 or :GSINxx is that a password could be put on it so that only JRG and the operators of the FR80 needed to know it.

3. As there may be a delay between the tape being off-lined and the lineprinter output being produced, the read-out program should issue a display to the operators' console indicating which pool tape (by number) has been used by it. This would allow the FR80 operator to examine the log and find the tape if he needs the tape in a hurry. The 1906A operators can ignore the display.

4. The lineprinter output produced by the read-out job should be placed initially in a filestore file and listed from there. This file would be erased by the next read-out job of this type. This would allow the FR80 operator to list the file if, for some reason, the lineprinter output was unavailable.

5. The FR80 operators will require a job that they can run at a teletype to release spool tapes which they have finished with.

6. It is suggested that the read-out job updates a file containing the names of tapes currently owned by the spool system. This should contain such information as the time the tape was obtained. The names allocated to tapes should reflect the camera required on the FR80 and the program (tape name could be FR80-cameraname where cameraname is the same as the parameter used to de-spool). This corresponds to the

different read-out options. The generation number of the new tape should be one greater than the last one in this series. The read-out program should list the set of tapes currently owned and give some kind of comment for tapes that look exceptional (old or a later generation number has already been deleted, for example). Initially, the listing of tapes owned may just be a directory listing.

7. The job to release tapes can be a normal RUNJOB. It must delete the entry from the file and possibly check for unusual action (deleting a very new tape, for example). Deletion must be by specifying the tape name, generation number and tape number.

8. The interface to the file containing tape names must be sufficiently well-defined so that PEB can add additional spool tapes to it which are listings on microfiche. These can be deleted by the standard release program. A name for the microfiche listing tapes must be defined (possibly FICHE).

9. A problem may occur on a crash when tapes go into query status. It must be possible to reclaim the tapes and update the file to its correct form. Care must be taken that the directory and file are consistent.

10. When the spool tape is produced on the 1906A, it is placed in the rack in the machine room just like any other tape. No special action is taken.

11. The FR80 operator will know the number of the tape, either from the lineprinter output, interrogating the file containing the listing, interrogating the directory or looking up the display on the console. The FR80 operator will expect to find the tape either in the temporary rack in the machine room or in the tape store home for the tape.

12. The FR80 operator will usually keep a tape around the FR80 until correct output for the work is obtained. It will then be moved to the tape store. At some stage the tape will be released by the FR80 operator. The tape must be returned to the tape store before it is released.

13. The 1906A operators will be responsible for an adequate supply of pool tapes.

14. A facility to allow the FR80 operator to put a comment into the file containing the currently-held tapes. This would be next to a particular tape and indicate why this tape was being retained for a particularly long period.

jc

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 12

The Colour Problem and the FR80

issued by
P A Dewar

25 March 1975

DISTRIBUTION: FRAH RB AWB MFC PAD AHF

JRG JWEL PMN DVR JMR WDS

RET RWW

Summary

Principles of colour specification are presented. The subtractive filter system supplied with the FR80 will allow us to use subtractive, additive or a combination of mixings resulting in the release of pigments in processed colour positive paper which additively combine and dyes in colour positive film which subtractively combine to provide tertiary colours (violets, browns etc).

A method of coding by the user is suggested and possible decoding processes emerge as points in an argument in which you are asked to join.

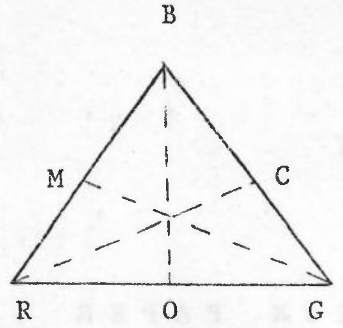
The severe restrictions imposed by the economics of the decoding process enforce us to impose a discrete structure upon the (hue)^2-brightness-saturation continuum (four dimensional but not a vector space) and so restrict the choice to the user. The model proposes sixty "colours", five saturation levels and a linear, five point, grey-scale. Even this model can not be implemented by start-of-service-day since empirical work must be done at the FR80 before the spectral characteristics can be modelled adequately.

Initial service must be the provision of primaries and secondaries with little control over grey level and saturation and with user-improvised tertiary mixing. I propose we say little about the limitations at the outset - the possibilities, after a little work has been done, are quite impressive.

1. PREAMBLE

The colours to which we attach the names "Red", "Blue" and "Green" were demonstrated by Maxwell to be a necessary and sufficient set for the production of any (Subjective) colour as perceived by us. He termed these the "Primary" colours and demonstrated a sort of "Vector Diagram" in the plane - his celebrated "Colour triangle":

Fig 1.1



Additive

This followed from experiments in which lights of the three primary colours were superposed on a white screen in different proportions and it was found that all colours including "white" were obtainable.

Note that the complete absence of Green is denoted by the point of intersection of the BR side and the G altitude in his colour triangle.

This principle forms the basis of the "Additive" colour systems (the R, G, B dot triads on a colour CRT for instance) in which Red, Blue and Green lights are mixed+. The altitude/side interecepts in Maxwell's colour triangle denote three other colours, "Orange", "Magenta" and "Cyan" which may be considered "Secondary" in the sense that they denote the absence in each case of one primary from a "White" source. or the presence of two.

It is instructive to think of:

*Orange	as	"All but Blue"	
Magenta	as	"All but Green"	1.1
Cyan	as	"All but Red"	

This leads to the term "Complementary Colours" for the set (O,C,M) and the representation:

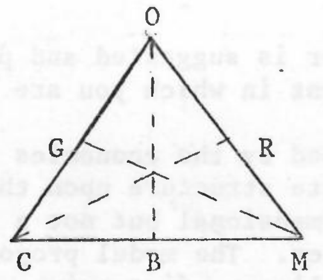


Fig 1.2

Subtractive

which is the inverse of the Maxwell colour triangle in that if we were to mix dyes≠ of the three secondaries (O,C,M) in roughly equal proportions we should obtain on our screen NOT RED + NOT GREEN + NOT BLUE, according to 1.1 and this is, of course, black ie no light at all. The subtractive principle is used in, eg, colour television cameras (sometimes) and is the basis of the FR80 colour system.

The FR80 will be supplied with a set of secondary filters installed (O,C,M). If the tube phosphors are of such a mix that we get an acceptable white with no filters in place then all will be well. If this is not the case then to produce white/grey/black on a standard colour film will require three strikes, each with two filters in place for a given intensity:

+ pigments, which selectively reflect, have the same effect.

* I use "Orange" rather than "Yellow" since "NOT BLUE" is distinctly redder than the standard yellow (Sodium light)

≠ ie filtering agents or filters through which the light passes.

Strike 1 O + M → Red) combine on film to give) but see sec. 6:
Strike 2 O + C → Green) neutral on the intensity) "two-strike
Strike 3 C + M → Blue) scale Black-White) white".)

which is rather cumbersome. (But note at no time will all three secondary filters be in position since this stops all light). If there is a spectral deficiency in the tube face then I suggest we investigate the possibility of installing a permanent filter to correct the "raw" tube light output to 6666^{o}K colour temp (midday high altitude sunlight). This would not affect the operation of the, supplied, "dynamic" filters - indeed it makes if far easier to calculate their effect but, more importantly, it enables us to represent the Grey scale with no filters specified it is on the assumption that we can obtain white light (say 6000^{o}K) from our tube that the following system is suggested.

2. USER INTERFACE - CODING

We have seen that "Colours" are elements in a pseudo-vector space of three dimensions (ie the necessary and sufficient basis consists in three elementary colours - we could liken this to a vector basis). It is therefore necessary to supply only three numbers - an ordered triple (a,b,c) to specify any "Colour" in our perceptual experience. We must not expect the user to do more, given the same basis.

A glance at fig 1.2, though, shows that we are representing the whole "colour space" on a 2-D sheet and, given an origin, a 2-vector will indicate any colour as a displacement from that origin. In a crude model, with no corrections for the spectral characteristics of the whole system (tube face-filters - film - processing of film) we need only two numbers from the user with, say, the cyan vertex as the understood origin.

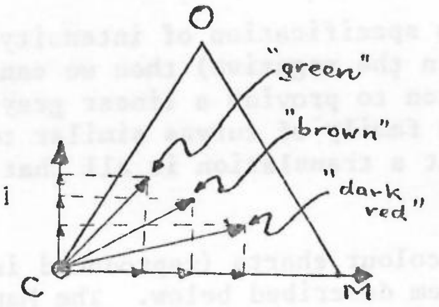


Fig 2.1 Subtractive "two-vector" colour specification

By this means we are obtaining two numbers which can be processed to obtain the quantities "hue" and "saturation" but note that this saturation is the inverse of term used in "The Trade" since the centre of the subtractive colour triangle is black (1.1) and so no "Pastel" shades are possible unless we leave the triangle. (The centre of Maxwell's colour triangle (additive) is white and so a set of nested similar triangles give less saturated colours - pinks, apple green etc.) With our system we must artificially introduce lower saturation by overstriking with white (hence the need for "unfiltered" white).

Hue and saturation are not sufficient, however, for the user he must be able to control the (absolute) intensity of his images and the minimum specification, by the user, will be the triple:

(hue, saturation, intensity) 2.1

and our job is to decode this triple in as simple a way as possible.

3. MACHINE INTERFACE - DECODING

Colour photography is bristling with problems - the Silver halides are only really sensitive to blue light; the exposure process is logarithmic in nature (since exposure of a new plate sensitizes, say n out of the m silver halide molecules; further exposure of the same plate has only m-n molecules to affect → effect is inversely proportional to (integrated) effect to date → log law). These are the two problems that affect us as decoders. Manufacturers build filters into their films in an attempt to get a "pan chromatic" (Sic) response over the visible spectrum - we shall have to correct for their inability to do this satisfactorily. The exposure problem is greater in that we are giving the user the ability to specify intensity but we shall be overstriking whenever he calls for an unsaturated colour. Our control over exposure resides in our ability to specify both intensity of the spot and number of strikes per line element.

For grey level exposure ("black and white") if we attempt to plot exposure vs strikes vs intensity we obtain a stepped surface. In a projection of this surface on to this page we obtain a family of curves:

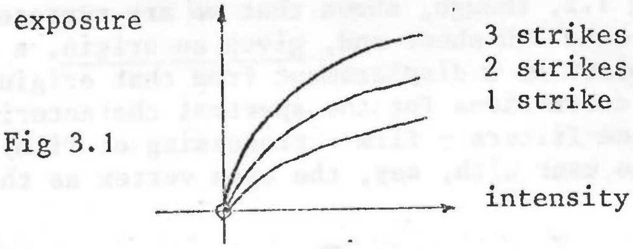


Fig 3.1

If we interpret the user's specification of intensity as exposure (ie the density of silver grains in the negative) then we can calculate the strike/intensity combination to provide a linear grey-scale for him. Of course we must expect a family of curves similar to figure 3.1 for each primary colour, I hope that a translation is all that is necessary to correct for colour.

I suggest that we produce colour charts (reproduced in the User Guide) by means of the model system described below. The Manual then becomes part of the system in that it provides the coding function; closure results if the colour chart reproduction in the Manual is a good one.

4. THE MODEL-CODING

I suggest exploiting the relative insensitivities in our visual perception to:

(i) changes in grey level,

(ii) changes in saturation, and

(iii) the subtler changes in hue

in order to represent these quantities on discrete scales (in the FR80 with the grey-level option these scales are almost continuous!).

The basic representation must abandon the Maxwell triangle in favour of a circular chart. Immediately two possibilities arise:

> Use "additive" formalism+ and have decreasing saturation towards the centre, with white at the centre,
>
> Use "subtractive" formalism+ having constant saturation with, of course, black at the centre.

I suggest five annuli and twelve radii thus partitioning the chart into sixty cells:

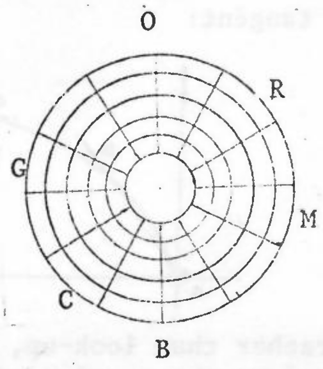


Fig 4.1

The number pair (i,j), iε (1...12), jε (1,...5) will specify the cell.

From the users' point of view 1 and 2 are equivalent although we must abandon one in favour of the other exclusively before publishing.

Possibility 4.1: the number pair represents (hue, saturation) and we must allow the user to "rotate the chart" upon a copy of itself to give him the facility to generate tertiary colours and we calculate the white overstrike/intensity to provide the exposure as specified in the number-triple (2.1) - the user now must send four numbers

(hue1, hue2, saturation, exposure) 4.1

Possibility 4.2: The number pair represents (hue, admixture with other primaries/secondaries) and in a sense is just the inverse with respect to the boundary of the chart and the "chart rotation" is not required by the user but a saturation (white mix) figure is, as is an exposure level. The four numbers the user now sends are:

(hue, mix, saturation, exposure) 4.2

The major criterion for choosing between 4.1 and 4.2 is ease of representation of the system in the User Guide. The decoding of one offers no advantage over that of the other since the colour/exposure space is four-dimensional, however we represent it.

In representing the system to the user: for 4.1 we provide six separate circular charts numbered 1...6 each being the result of the superposition of the basic chart and 0, 1, 2, 3, 4, 5 increments of $\pi/3$ in the rotation of the "overlay"; for 4.2 we provide 5 charts wherein, by white over-striking of the whole pattern in increments from chart to chart we represent decreasing saturation.

(There is a trivial difference in the range of the elements in the (i, j, k, 1) quadruple in that (1,..5) becomes (1,..6) for one element in 4.1.)

+ NB these are purely formal distinctions (psychology-at-work!).

5. THE MODEL - DECODING

Our approach to the final system must be empirical in that we can only set up an experimental model, test it by measurements at camera position (by means of an integrating photometer, say) and upon a small number of users - modifying the system according to the response obtained.

To decode the users' specifications we need a representation of the "System Response Curves" in figure 3.1. A tolerably good approximation to a logarithmic curve is three straight-line segments - best fit lines somewhere between chord and tangent:

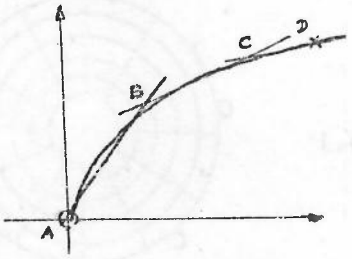


Fig 5.1

If we decide to calculate, rather than look-up, the strike/intensity values then this representation requires the storing of the number pairs A, B, C and D for each curve - 24 numbers in the three-strike - three-line approximation. The intercepts are obtained by interpolation.

I suggest that we work with a five-level logarithmic exposure scale at the outset. We choose which intercept for a given exposure on the precept "Increased intensity is cheaper than a restrike" and so we take the right hand point, eg b, rather than a, in figure 5.2.

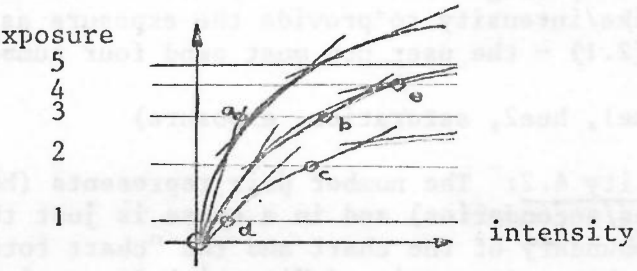


Fig 5.2

By limiting the user's exposure to a small number of discrete levels we allow ourselves the ability to superimpose strikes/intensity for the filters indicated so that the exposure can be closely approached by look-up if necessary (we have the full strike/intensity range at our disposal).

Assuming possibility 4.2 the four numbers: (hue, mix, saturation, exposure) obtained from the coding chart as: (sector, annulus, subchart, required frame brightness), [strictly the first three are on the supplied chart the fourth is artistic licence on a five point log scale!] are decoded as ((filter(s), strike(s), intensity), (fiter(s), strike(s), intensity), (white, overstrike(s), intensity))+, where the first and second filter selections provide the tertiary colour, the white complements these for saturation and the total strike/intensity is calculated or looked-up so as to ensure the correct integrated exposure.

+ or as ((f,s,i), (f,s,i), ("black overstrikes")) (see 6)

The steps in the process are:

$$(\text{hue, mix, saturation}) \rightarrow ((F_1,e_1),(F_2,e_2),(Wh,e_3))$$

$$((F_1,e_1),(F_2,e_2),(Wh,e_3),\text{exposure}) \rightarrow ((F_1,si_1),(F_2,si_2),(Wh,si_3))$$

Where F_i is one or two FR80 secondary filters

e_i is indicated relative exposure

si_j is the calculated or looked-up strike/intensity for that filter and is a function of the other si_j $(i \neq j)$ and the desired exposure.

This is not as formidable as it at first appears, instruction #34 on the FR80 seems to embody the function $F_i \rightarrow i_i$ ie if we ensure that our system is additive with respect to exposure then relative intensity i_i is set whenever filter F_i is selected and this can effect the "linearisation" of the system response (silver deposited) to applied stimulus (required frame "brightness") translation is necessary to make the different sets of curves (3.1) arising from the presence of the filters coincide sufficently well for us to use a simple "Counting" process to arrive at the correct integrated exposure.

Also, since (red, blue, green) are a sufficient set, we could instead of overstriking with white, overstrike with a supplementary filter combination so that some of the already present, primaries are "supplemented" so as to provide the required amount of white. This probably needs more calculation than the former operation since this (former) will be less deeply encoded - we will have produced the charts by white overstriking and we just need to repeat this process.

Our colour service could provide, then, the three options: 2.19, 2.20, 2.21 in the FR80 User Manual, the "nesting" of 2.19 and 2.20 within 2.21 is obvious and suggests storing 2.21 as thirteen triples (α,β,γ) on the basis (O+M, M+C, C+O). The interposing of four glass/air interfaces (in the (O,M,C) filter system to be supplied) for each primary, suggests the need for much overstriking anyway so the inclusion of 2.19 as the (sub-) set (0,0,2), (0,2,0), (2,0,0) and 2.20 as (0,0,2), (0,2,2) etc is probably quite in order. 5.1

Note that the three primaries are never used for any colour in 2.21. Take the third "colour" in the table: blue/blue-green = (0,2,0) + (0,0,1) = (0,2,1) (on the basis (R,B,G) = (O+M, M+C, C+O)) I assume that #34 has embodied the necessary spectral compensation. Now, the addition to this of the triple (1,0,0) gives:

(1,2,1) = (1,1,1) + (0,1,0) [by 5.1 this is: (2,4,2)+(0,2,0)]

= 1.white + 1.blue ie unsaturated (50%) blue - a "pastel" shade.

The addition, on the other hand of, say, (2,0,0) gives (1,1,1) + (1,1,0) ie 50% saturated magenta (note the swings-and-roundabouts situation we are in: in the (O,M,C) system ("ours") this is just 1 strike low intensity white + 1 strike looked-up-intensity magenta), (or 2 Wh + 2.M by 5.1).

Note, also, that this table will provide a two-level saturation scale. For a five level scale we should need a table of the form:

$(4,4,4) = \text{wh}.4$
$(4,4,3) = \text{wh}.3+M$

ie "20% saturated" magenta and so on, the triples of the form (α,α,α) always being interpreted as wh.α and obtained from the (corrected) tube face in just α strikes (or fewer if the "clear" filters are completely removed and suitable weights attached to compensate for the dynamic unbalance that results). (I have learned that the filters can be withdrawn.)

So, before launching upon the coding/decoding system which provides a comprehensive colour "service" we could consider this enhanced version of 2.21 (in the user manual) which can provide 25 colours (including some three-colour mixes missing from 2.21) in five possible saturations and which requires a table of 125 triples -375 words - and which could be presented to the user as one 25 sector, five annulus chart. We could allow a "total exposure" number to be specified also - the decoding is trivial excepting that we always extract the white information first by repeatedly subtracting (1,1,1) from the indicated "colour vector" until one of (α,β,γ) is zero and then use clear overstriking for the resulting white component. This could be incorporated into the table or we can omit the five triples (α,α,α) and use jumps on "IF(ALPHA-BETA)" etc.

To return to the "Comprehensive Colour System" we must include a calculation for total exposure since, eg the red could be fully exposed before we overstrike with white (r+b+g, filter compensated); this results in the addition of only b+g to the picture and this may not be the user's intention (we are not supplying exposure charts in this model). From 5.1 we have:

$$((F_1,si_1),(F_2,si_2),(Wh,si_3)) \rightarrow ((F_1',s_1),(F_2',s_2),(Wh',s_3))$$

Where F_i' is the "#34 compensated" F_i.

The store requirement for the colour/saturation decoding is, if we use look-up, 5x12x5 or 5x12x6 triples ie 900 or 1080 in this crude model. I am sure the range will be adequate.

To cope with the exposure number (in this model up to five, log. scaled) we calculate the total exposure for the most saturated colour and scale the others accordingly. The three, relative, exposures for F_1, F_2, Wh are "read off" (interpolated from) the approximations to the strike/intensity curves (5.1), colour and intensity is set and the required number of strikes per (partial) exposure made.

I will now restate the decoding problem in the form of a possible solution, bringing out the major points in a "feasibility" argument and so leave it open to discussion.

6. ARGUMENT

(1) "Many Colour Option"

Suppose that, by use of #34 we have so set up the relative intensities, F'_i, that we obtain a "two strike white". This can be obtained in one of three ways:

1 strike Magenta + 1 strike Green: [(C'+O') + M']
or 1 strike Cyan + 1 strike Red : [(O'+M') + C']
or 1 strike Orange + 1 strike Blue : [(M'+C') + O']

each of these giving a "white" image on the viewed medium - note that both subtractive and additive mixing is going on here!

There is also the "1-strike-white" option with the colour corrected tube face and no other filters. (These could form the basis of the routine testing, incidentally)

(2) "Many Colour Option"

Suppose also that, consistently with 1, above, we have so set the absolute intensities that it takes five strikes fully to expose (saturate) the film, whichever ever filter is set, (by #34). (These settings will be determined by measuring the attenuating effect of the filters and the spectral response of the film so obtaining the exposure vs strikes/intensity curves.)

Then, in the proposed model, for the saturated primaries and secondaries we obtain a five point (logarithmic) exposure range and calculation for the tertiary colours is formidable - we can "linearise" however.

To approximate linearity on a five point exposure range requires, say, a "sixteen-strike-white" intensity setting for each primary and secondary filter selection. The curves are "straightened" by the "built-into-the-software" correspondence: (1,2,3,4,5) → (1,2,4,8,16) between linear exposure and strikes and we have a basis for calculation of exposure for any colour filter combination eg exposure "3" requires 4 strikes of some filter combination (which could be, eg 2C' + R' + B' ie 50% saturated blue at mid exposure). (Note that full exposure should be discouraged) - level 4 being the *advised maximum.

This commits us in the "Full Colour Option" to eight (although users will still be *allowed the "16-strike" option) overstruck images on the brightest parts of a frame. However, this model reduces the colour options as the demanded exposure decreases (at exposure 1 only the primaries and secondaries, fully saturated, are possible).

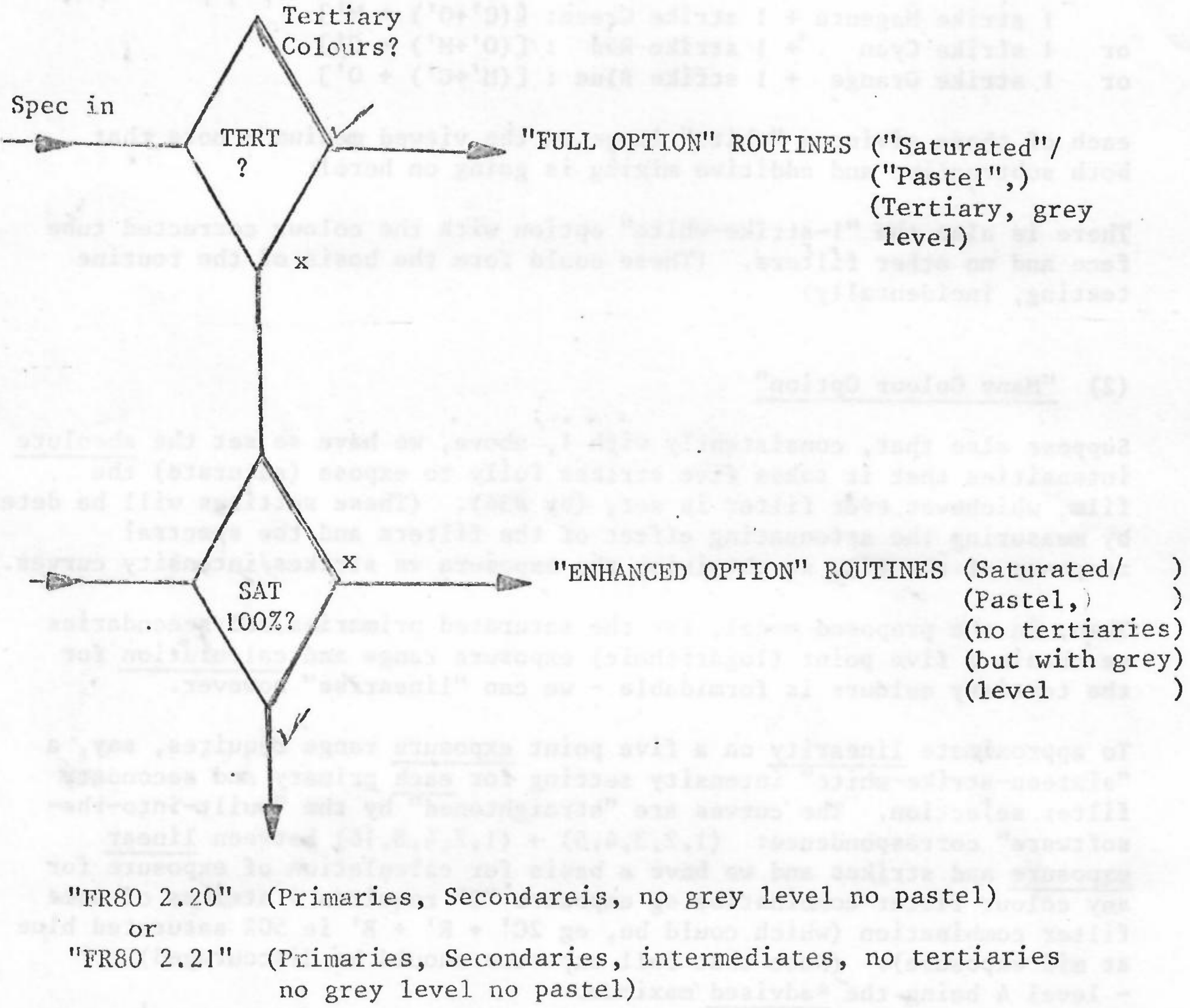
Accordingly we can propose model 2 for decoding wherein:

(a) The maximum number of strikes is always performed when tertiary colours are being specified.

(b) A reduced maximum number, at higher intensity, when only primaries/ secondaries are being selected.

(c) In the case where only fully saturated primary/secondary colours are being selected, The minimum strike number, consistent with a five-point exposure range)not logarithmically over-struck) can be employed.

The decoding in this model requires the following processing flow:


Tertiary Colours?
Spec in
TERT ?
✓
"FULL OPTION" ROUTINES ("Saturated"/ ("Pastel",) (Tertiary, grey level)
x
SAT 100%?
x
"ENHANCED OPTION" ROUTINES (Saturated/) (Pastel,) (no tertiaries) (but with grey) (level)
✓
"FR80 2.20" (Primaries, Secondareis, no grey level no pastel)
or
"FR80 2.21" (Primaries, Secondaries, intermediates, no tertiaries no grey level no pastel)

(This diagram may indicate the (inverse) time order in which we present the service to the user)

For the "Full Option" Model 2 (a) (which is still, remember, very coarse grained indeed but should be sufficient) we alter the correspondence by means of the intensity setting (each strike curve in fig 5.2, eg, goes through the origin - zero intensity is zero exposure, however many the strikes) to:

$$(1,2,3,4,5) \rightarrow (16_1, 16_2, 16_4, 16_8, 16_{16})$$

where the suffices indicate that the intensity is reduced so that $16_1 \equiv$ sixteen strikes at low intensity equivalent to 1 at maximum intensity.

The logarithm is now incorporated into the intensity settings (and these still have to be corrected for spectral response so seven issues of #34 are required at each change of "exposure").

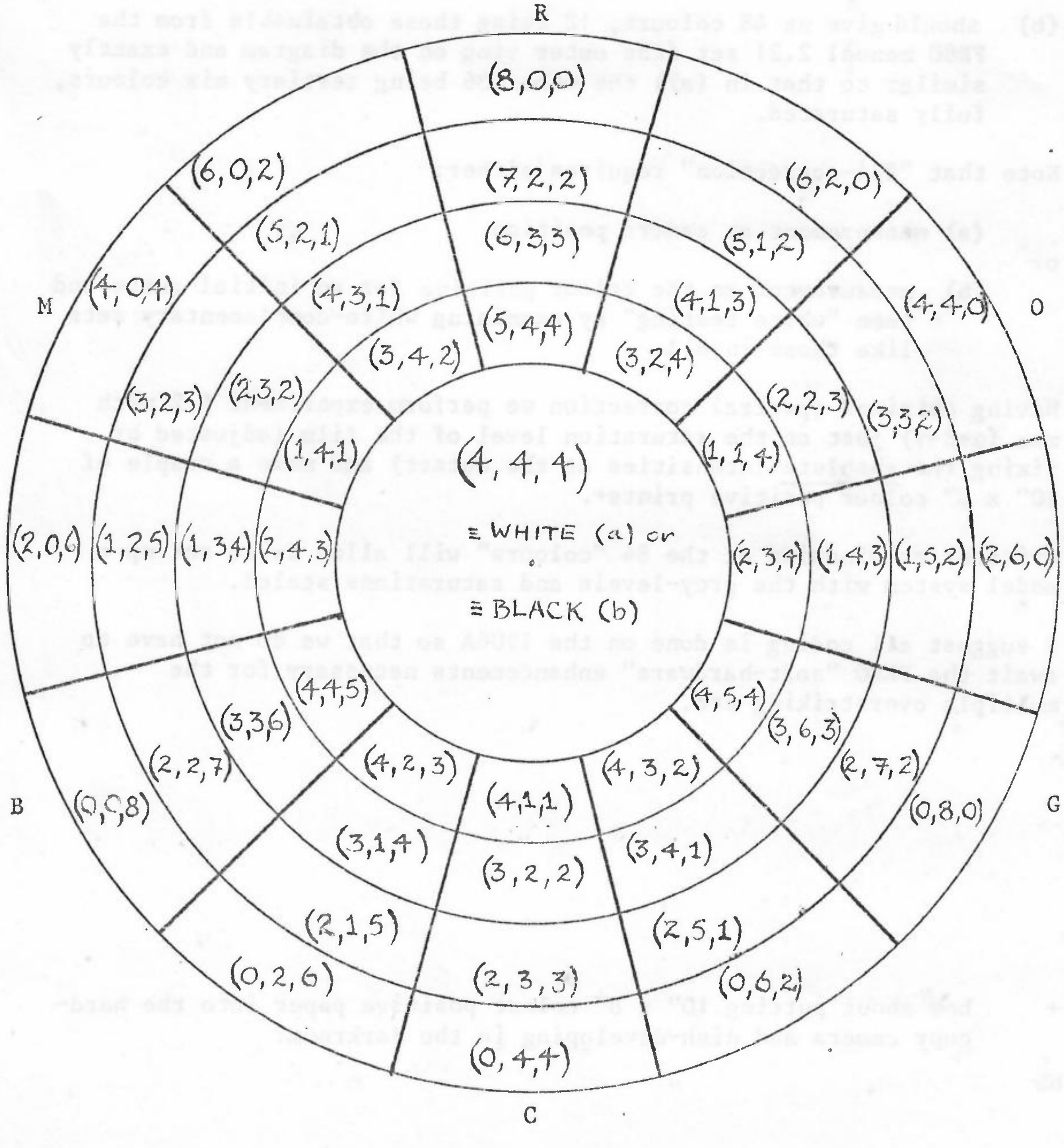
Sixteen now seems an unreasonable range of strikes for merely colour mixing so can we reduce the total strike number consistent with the number of filter changes necessary to achieve the range of saturations/tertiary mixes that we shall offer? An experiment is called for since we are now entering the subjective field of colour/grey level/ saturation level discrimination. I suggest that, as soon as is practicable we generate the following charts:

(a) using (R',G',B') as a basis for the number triples representing strikes

6.2

(b) using (O',C',M') as a basis

Where R' = "#34 - corrected red" etc.



(a) on (R',G',B')

(b) on (O',C',M')

In "service" no triple should be of the form (α,α,β) since we can switch between bases (a) and (b) to avoid this; and every triple should have at least one zero since we can subtract (1,1,1) repeatedly up to n times and then over strike with:

(i) white n times [on (R',G',B') for pastels]

or

(ii) "black n times" ie "n x no strikes-but-maintaining-the-total-"strike"-count [on (O',C',M') for the tertiaries.],

(a) should give us 48 colours, 36 being "pastel shades" since the triples of the form (α,β,γ) $\alpha,\beta,\gamma \neq 0$ can have (1,1,1) subtracted until one of $\alpha,\beta,\gamma = 0$ and so min (α,β,γ) will be the white admixture, reducing saturation

(b) should give us 48 colours, 12 being those obtainable from the FR80 manual 2.21 set (the outer ring on the diagram and exactly similar to that in (a)) the other 36 being tertiary mix colours, fully saturated.

Note that "#34-correction" requires either:

(a) measurement at camera position

or

(b) measurement on the colour positive for an initial guess and then "white testing" by examining white-complementary sets like those in 6.1.

Having obtained spectral correction we perform experiment 6.2 with max $(\alpha+\beta+\gamma)$ just on the saturation level of the film (adjusted by fixing the absolute intensities at the outset) and make a couple of 10" x 8" colour positive prints+.

Critical examination of the 84 "colours" will allow us to set up a model system with the grey-levels and saturations scaled.

I suggest all coding is done on the 1906A so that we do not have to await the FR80 "soft-hardware" enhancements necessary for the multiple overstriking etc.

+ how about putting 10" x 8" colour positive paper into the hard-copy camera and dish-developing in the darkroom?

bb

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 13

SMOG on the 360/195

issued by
M F Chiu
& A W Burraston

15 May 1975

DISTRIBUTION: FRAH RB AWB MFC PAD LOF AHF JRG
JWEL PMN DVR JMR WDS RET RWW MBK
(ME PAM)

REFERENCES

(1) SMOG issued 25 March 1975 by FR80 Project Branch

(2) RL thoughts on control data for spooled graphics files
by C D Osland 8 March 1975

1. INTRODUCTION

The part of SMOG that produces FR80 orders output to magnetic tape (including Logging and LOADGO headers) is now available on the 360/195 in a catalogued library dataset called SYS1.SMOG and may be used for testing purposes. A list of routines is included with this note.

Users should note that the routine DVALL (see Ref 1) has not been included since calling FR80ST with precisely the same argument has the same effect when the FR80 is the only target device.

2. GRAPHICAL OUTPUT (TEMPORARY ARRANGEMENTS)

Since work remains to be done on the graphics spooling system, the relevant I/O routine SP$PUT, described in Ref 2 has been simulated as follows:

Tape data [K=1] and filemarks [K=4] are written via FORTRAN channel 14.

Control information [K=3] is sent to FORTRAN channel 12 in A format, preceded by the message:

```
*** SPOOL CONTROL BLOCK
```

As some work is still required to organise the handling of 9-track IBM standard-labelled tapes from both the system and the FR80 sides, I suggest we stick to 7-track unlabelled tapes, using existing SD4020 facilities for the time being. To arrange for the collating of tapes with the relevant lineprinter monitor, the following temporary procedure has been proposed:

(1) The user should begin his jobname with the letters CL

(2) FORTRAN channel 13 (FR80 tape monitoring information) should be attached to the special forms queue (see below) to be listed at RL.

(3) Mrs Pauline Gammon will then collect the tape and output from the CL pigeon-hole, fill out the appropriate white card and send tape and output via the courier to ACL marked for processing on the FR80.

(4) In case of trouble, the user has a copy of the monitoring information, output on FORTRAN channel 12 (plus a banner).

Once the despooling system is working, this procedure should be superseded.

3. VIEW$

A viewing program (VIEW$) is provided to interpret FR80 data files using MUGWUMP and may be used for testing. In that case, it will probably prove more convenient if SMOG output is routed to a disk dataset to be picked up by VIEW$. The necessary JCL is given below. Optional printing of the FR80 code may be obtained by putting PARM.G='P=1' in the appropriate place on the EXEC card. [We hope soon to include three extra parameters:

S = n to start viewing from frame number n

I = m increment = m , ie view every m(th) frame

E = ℓ stop viewing at frame number ℓ].

VIEW$ uses the right-hand margin of the screen area to "comment" on the plotting process. The comments appear at the time they are encountered in the FR80 data stream. They include:

CHAR - character string encountered
VFM = n - vector family of n interpolated lines
ADV = n - n advance frames requested
SCS = n - set character spacing of n raster points
SCH = n - set character height of n raster points
SLS = n - set text line spacing of n raster points
SCR = n - set character rotation of n x 45^{o} counterclockwise
SI = n - set intensity level to n

last frame only:

MVCS n - no of multiple vector continue orders
UNRC n - no of orders unrecognised by VIEW$
ENJB - end job.

4. JOB CONTROL EXAMPLES

(a) To compile, load and go with a FORTRAN program driving SMOG. Output graphics to FR80 tape.

```
//CLSMOG JOB (AZ05,A3),CHIU

/*SETUP  GRAPHS,TAPE7,BLP,W

// EXEC FHCLG,SYSLIB='SYS1.SMOG',CPRINT=YES

//C.SYSIN DD *

          '
          '
          '
          '
      FORTRAN code
          '
          '
          '
          '

/*

//G.FT14F001 DD UNIT=TAPE7,VOL=SER=GRAPHS,DISP=(,PASS),

// LABEL=(1,BLP),DCB=(RECFM=U,BLKSIZE=1536,DEN=1)

//G.FT14F002 DD UNIT=TAPE7,VOL=SER=GRAPHS,DISP=(,PASS),

// LABEL=(2,BLP),DCB=*.FT14F001

//G.FT14F003 DD UNIT=TAPE7,VOL=SER=GRAPHS,DISP=(,PASS),LABEL=(3,BLP),

// DCB=*.FT14F001

//G.FT14F004 DD UNIT=TAPE7,VOL=SER=GRAPHS,DISP=(,PASS),LABEL=(4,BLP),

// DCB=*.FT14F001

//G.FT12F001 DD SYSOUT=A

//G.FT13F001 DD SYSOUT=(J,,99)

//EXEC GRAFTAPE
//A.FT06F001 DD SYSOUT=(J,,99)
```

NB (i) Because of the filemarks required by the FR80 LOADGO and logging programs, 4 DD cards are required for FORTRAN channel 14. These should all be replaced by the one card:

```
//G.GRAPHICS DD SYSOUT=(G etc)
```

when spooling is available.

(ii) We have catalogued the DCB specification so that:

DCB=XAFR80

may be used in place of all the above.

(b) To compile, load and go with a FORTRAN program driving SMOG. Output to disk dataset for subsequent viewing by VIEW$.

```
//A3TEST JOB (AZ05,A3),CHIU

// EXEC FGCLG,SYSLIB='SYS1.SMOG',REGION=210K

//C.SYSIN DD *

            '
            '
            '
            '
         FORTRAN code
            '
            '
            '
            '

/*

//G.FT14F001 DD DSN=APR.WALTDIS,UNIT=DISK30,VOL=REF=FREEDISK

// DISP=OLD

//G.FT14F002 DD DSN=APR.WALTDIS,UNIT=DISK30,VOL=REF=FREEDISK,DISP=MOD

//G.FT14F003 DD DSN=APR.WALTDIS,UNIT=DISK30,VOL=REF=FREEDISK,DISP=MOD

//G.FT14F004 DD DSN=APR.WALTDIS,UNIT=DISK30,VOL=REF=FREEDISK,DISP=MOD

//G.FT12F001 DD SYSOUT=A.

//G.FT13F001 DD DUMMY
```

where the FREEDISK dataset APR.WALTDIS was previously created with a DCB of (RECFM=U,BLKSIZE=1536)

[or DCB=XAFR80]

(c) To interpret the graphical output produced in example 4 routing the pictures to the MUGWUMP filestore. Lineprinter listing of the FR80 orders is requested.

```
//A3LOOK JOB (AZ05,A3),CHIU,CLASS=X

// EXEC JOBLIB,LIBRARY='SYS1.SMOG',MEMBER=VIEW$,REGION=80K,

// PARM.G'P=1'
```

```
//G.FT14F001 DD DSN=APR.WALTDIS,UNIT=DISK30,VOL=REF=FREEDISK,

// DISP=SHR

//G.MUGWUMP2 DD DSN=MUGWUMP2,DISP=SHR
```

NB Some monitoring is always produced on the lineprinter (G.FT06F001) eg intensity settings, advance frames, non-identified characters etc. The output may be suppressed by inserting:

```
//G.FT06F001 DD DUMMY
```

<u>immediately after</u> the EXEC card(s)

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 14

FR80 Graphics on the 360/195

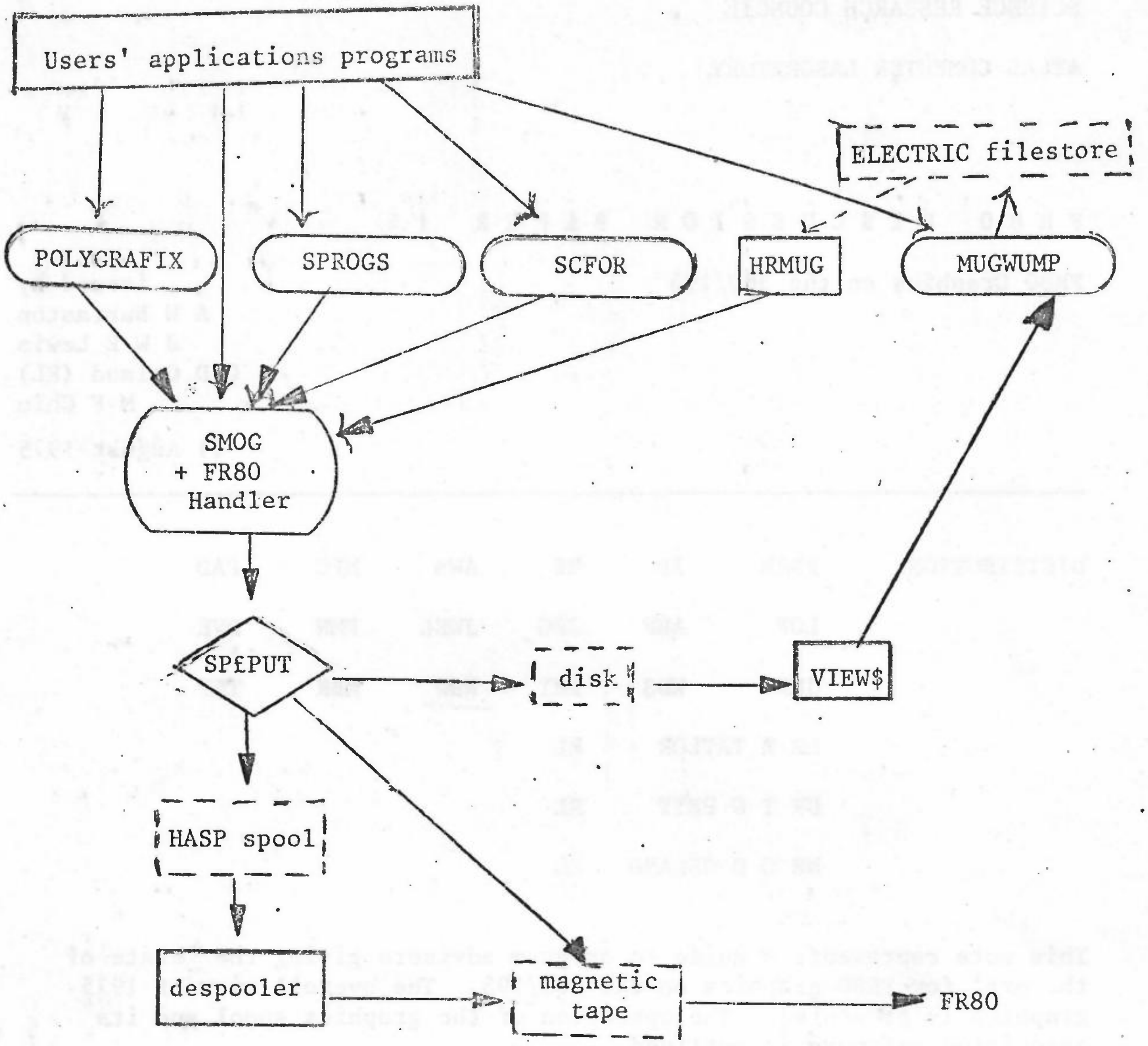
issued by
A W Burraston
J W E Lewis
C D Osland (RL)
M F Chiu

11 August 1975

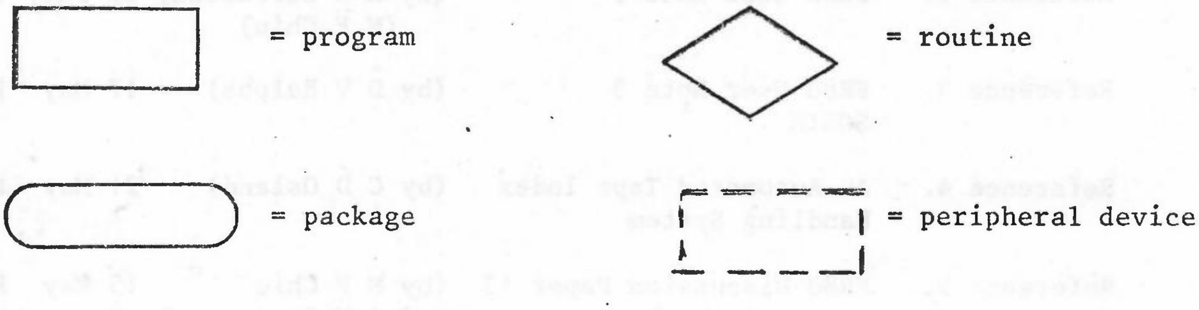
DISTRIBUTION:	FRAH	IB	RB	AWB	MFC	PAD
	LOF	AHF	JRG	JWEL	PMN	DVR
	JMR	WDS	RET	RWW	MBK	TRP
	DR R TAYLOR		RL			
	DR T G PETT		RL			
	MR C D OSLAND		RL			

This note represents a guide to program advisors giving the 'state of the art' for FR80 graphics on the 360/195. The overall view of 1975 graphics is presented. The operation of the graphics spool and its associated software is outlined.

Reference 1.	The SMOG System Manual Appendix 6	(by P A Dewar)	June 1975
Reference 2.	FR80 User Note 9	(by A W Burraston) (M F Chiu)	22 July 1975
Reference 3.	FR80 User Note 5 SCSIM	(by D V Ralphs)	17 May 1975
Reference 4.	An Automated Tape Index Handling System	(by C D Osland)	21 May 1973
Reference 5.	FR80 Discussion Paper 13	(by M F Chiu and A W Burraston)	15 May 1975


Users' applications programs
ELECTRIC filestore
POLYGRAFIX
SPROGS
SCFOR
HRMUG
MUGWUMP
SMOG + FR80 Handler
SPfPUT
disk
VIEW$
HASP spool
despooler
magnetic tape
FR80

Diagram


= program
= routine
= package
= peripheral device

1. INTERCOMMUNICATION OF PACKAGES

The diagram illustrates the projected intercommunication of programs, packages and I/O devices handling graphics on the 360. The main target device is the FR80, but FR80 orders stored on tape or disc may be routed via MUGWUMP to the ELECTRIC filestore using VIEW$ (reference 2). HRMUG (at present available at RL to route pictures to the SD4020) has been modified by Dr T G Pett to send output to the FR80. Users' applications programs may call on the graphical services of the higher level packages (thereby incurring overheads in cpu time and core size) or use the lower level package SMOG direct.

The key routine in SMOG is SP£PUT which may be employed to send graphical orders to HASP spool, magnetic tape or disc. (In the best traditions of IBM FORTRAN, the appropriate device is selected by JCL.) For writing to disc or 7-track unlabelled magnetic tape only, SP£PUT may be replaced by equivalent FORTRAN routines.

2. THE 'STATE OF THE ART'

SMOG The SMOG package is available on the 360 and produces FR80 order code on tape or disc. The routine DVALL (described in the SMOG manual) is not available - ie Tektronix, Hewlett-Packard plotter, etc, are not supported by SMOG. However, the stand-alone program VIEW$ may be used to interpret FR80 orders and output pictures to the ELECTRIC filestore using MUGWUMP (see reference 2).

SCFOR Ralphs' program SCSIM (reference 3) has been modified and implemented on the 360. Only one camera per run can be supported as the SELECT routine will operate only once (NB SELECT in SCSIM is also used differently from SELECT in the old SD4020 package). For users who cannot modify their 4020 programs (eg have lost the source decks) the 360 version of SCSIM allows them to set the number of the FR80 camera (once per run) in the PARM.G field of the EXEC card.

```
eg    //EXECJOBLIB,LIBRARY='ULIB.ATLAS',
      //MEMBER=OLD4020,PARM.G='C=7'
```

for FR80 camera number 7 (1-up hardcopy)

The FR80 cameras allowed (see Section 2.1 of the SMOG Manual) are 4, 5, 7 or 8.

A procedure XASDFRL is being provided so that users can re-linkedit their load modules to incorporate the SCSIM routines in place of the 4020 ones. Having used this procedure to create new load modules, users should be reminded that they will require new JCL.

For overlay programs, the initial version of XASDFRL will require re-specification of the overlay structure to the linkage editor.

The new load module will require about 12K bytes more core.

POLYGRAPHICS With the advent of /360 SMOG, the low level drivers of POLYGRAPHICS has been rewritten and is at present being tested. Initially, only vector drawing facilities will be available and it is then planned to add the hardware characters. A substantial rethink will be required as far as the implementation of the SAVE commands are concerned. At present core store is used but obviously it would be desirable to take advantage of the FR80's facilities in this respect.

SPROGS The FR80 version of this program is in the process of being implemented on the 360 and should be useable (though probably not bug-free) by the beginning of August 1975. It requires disc backing store for most applications and special directions to the linkage editor (on the //SYSLIN stream) to simulate the 1906A's #ELASTIC concept. Tektronix 4010 graphics is supported via MUGWUMP; line-printer graphical is also available. Hewlett-Packard pen plotter and PDP are not supported in this version of SPROGS. In order to facilitate the running of this package, a procedure XASPROGS will be supplied.

3. SPECIAL REQUIREMENT USERS

Some heavy users of graphics may wish for further facilities than those provided by the basic system. This section mentions some of these facilities and suggests how they might be implemented on the 360 without 'ordinary' users incurring the overheads that would normally arise if special features were made part of the lower-level graphics packages.

In order to provide FR80 LOAD-GO header routines it is suggested that FR80ST(I) and FREND be used together with any routines they may call.

A. Output to more than one camera per run

The ordinary SMOG output routine FRIGN is, on the 360, a FORTRAN routine that calls SP£PUT. A user wishing to write output to several cameras might do the following:

(1) Rewrite FRIGN to incorporate FORTRAN formatted WRITE statements as well as a call to SP£Put (to keep 195 graphics logging in order). The user would then switch between channels by setting the appropriate channel number in a COMMON block. The bulk of the graphical output should go through SP£PUT, while FORTRAN WRITE's should be used for subsidiary output to unlabelled 7-track tape.

(2) Save the contents of COMMON/SMOGC/ and COMMON/SMOGFR/ in arrays so that they may be restored and/or updated before accessing a given camera.

(3) To set up the appropriate header information etc for each camera,

(i) If necessary, call FRIGN to clear the buffer for the previous camera.

(ii) Load COMMON/SMOGC/ and /SMOGFR/ from the save areas.

(iii) Call FR80ST(ICAM) where ICAM is the number of the required camera.

(iv) Set a switch so that FRIGN outputs to an appropriate channel.

Graphical orders will then be routed to disc or tape via the required channel.

(4) To change camera:

(i) Call FRIGN to clear the buffer for the previous camera.

(ii) Store COMMON/SMOGC/ and /SMOGFR/ in the save areas for the previous camera.

(iii) Restore COMMON/SMOGC/ and /SMOGFR/ from the appropriate save area.

(iv) Set the switch for FRIGN to output via the new channel.

(5) At close down, (4) should be repeated with a call to FREND for each camera.

If sufficient interest is expressed, routines could be provided by us to carry out these processes.

The scheme outlined above is not completely foolproof especially if the user calls SMOG routines that rely on locally set data initialised constants. In particular, he should try to avoid outputting user-generated stored pictures to more than one channel if they contain advance film orders and he should ensure that output text strings are "sealed" with an end-of-message marker before changing channels.

B. Jobs outputting more than one tape-load for a given camera

A user's job may generate sufficient data to fill two or more reels of magnetic tape. We hope to include Multi-volume processing (analogous to the present OS method) within SP£PUT before long, although the use of 9-track 1600 BPI tape should make the urgency for this requirement less pressing than it will be for 7-track 556 BPI tape.

C. Heavy use of a restricted set of vector orders (using raster coordinates)

A version of the FR80 handler is available, compatible with SMOG, that allows a number of FR80 orders to be generated directly without the overheads of SMOG. Interested parties should contact MFC/Program Advisory Office for further details.

4. HASP SPOOLING

We envisage that most graphical output will be routed to HASP spool via the SP£PUT routine, except that generated on disc for testing purposes in conjunction with VIEW$. The relevant JCL is described in Appendix 6 of the SMOG Manual. Note that there is a different spool queue for each type of camera/output medium.

The Despooler is in the process of being planned and written. It should be run as the result of any of the following conditions:

(i) Operator initiation

(ii) Submission via ELECTRIC

(iii) Automatic submission as the result of triggering by HASP (ie when the spool area has been filled past a given level).

The despooling program itself will be responsible for maintaining a pool of tapes, allotting them and booking them out of RL. A subsidiary program will be used to book them back in again. The necessary software for this housekeeping is already available (reference 4). In addition, the despooler must produce the necessary FR80 operator instructions for each tape.

5. INTERIM USE OF FR80 GRAPHICS

(See also reference 5)

As SP£PUT is still nascent, though imminent, the following JCL examples are supplied to enable users who are "rarin to go" on the 360 to use the packages SMOG amd SCFOR immediately. This JCL is rather bulky compared with that which will replace it when SP£PUT is implemented (see reference 1 for example).

The examples below illustrate the output of FR80 orders to tape or disc only. Examples (a) and (b) are simplified (though still valid) versions of examples 4(a), (b) and (c) in reference 5.

(a) To compile, load and go with a FORTRAN program driving SMOG. Output graphics to FR80 tape.

```
//CLSMOG JOB (AABB,CC),CHIU
/*SETUP   GRAPHS,TAPE7,BLP,W
// EXEC FHCLG,SYSLIB='SYS1.SMOG',CPRINT=YES
//C.SYSIN DD *
                 .
                 .
                 .
              FORTRAN code
/*
//G.FT14F001 DD UNIT=TAPE7,VOL=SER=GRAPHS,LABEL=(1,BLP),
// DCB=(XAFR80,DEN=1)
//G.FT14F002 DD UNIT=TAPE7,VOL=SER=GRAPHS,LABEL=(2,BLP),
// DCB=(XAFR80,DEN=1)
//G.FT14F003 DD UNIT=TAPE7,VOL=SER=GRAPHS,LABEL=(3,BLP),
// DCB=(XAFR80,DEN=1)
//G.FT14F004 DD UNIT=TAPE7,VOL=SER=GRAPHS,LABEL=(4,BLP),
// DCB=(XAFR80,DEN=1)
//G.FT12F002 DD SYSOUT=A
//G.FT13F001 DD SYSOUT=(J,,99)
// EXEC GRAFTAPE,USER=REMOTE
```

(b) To create an appropriate scratch disc data set. Then to compile, load and go with a FORTRAN program driving SMOG. Output to disc dataset to be examined by VIEW$ in the next job step.

```
//A3TEST JOB (AABB,CC),CHIU
// EXEC NULL
//A DD DSN=&&TEMP,UNIT=TEMP14,DISP=(,PASS),DCB=XAFR80,
// SPACE=(CYL,2)
// EXEC FGCLG,SYSLIB='SYS1.SMOG',REGION.G=210K
//C.SYSIN DD *
                 .
                 .
                 .
                 FORTRAN code
/*
//G.FT14F001 DD DSN=&&TEMP,UNIT=TEMP14,VOL=SER=TEMP14,DISP=(OLD,PASS)
//G.FT14F002 DD DSN=&&TEMP,UNIT=TEMP14,VOL=SER=TEMP14,DISP=MOD
//G.FT14F003 DD DSN=&&TEMP,UNIT=TEMP14,VOL=SER=TEMP14,DISP=MOD
//G.FT14F004 DD DSN=&&TEMP,UNIT=TEMP14,VOL=SER=TEMP14,DISP=MOD
//G.FT12F001 DD SYSOUT=A
//G.FT13F001 DD DUMMY
// EXEC JOBLIB,LIBRARY='SYS1.SMOG',MEMBER=VIEW$,REGION.G=80K,
// PARM.G='P=1'
//G.FT14F001 DD DSN=&&TEMP,UNIT=TEMP14,VOL=SER=TEMP14,DISP=SHR
//G.MIGWUMP2 DD DSN=MUGWUMP2,DISP=SHR
```

(c) To compile, load and go with a FORTRAN program driving SCFOR, the JCL is similar to that for SMOG except that SYSLIB should refer to SYS1.SCFOR in place of SYS1.SMOG. Unless he calls SELECT before IDST (the new SCSIM method), the user may specify the FR80 camera number in the PARM.G field of the appropriate EXEC card.

via	C=4	35mm black and white
	C=5	16mm black and white
	C=7	12" x 12" 'single-up' hardcopy
	C=8	"4-up" hardcopy.

An example EXEC card might be:

```
// EXEC FHCLG,SYSLIB='SYS1.SCFOR',REGION.G=210K,
// PARM.G='C=8'
```

(d) To use the procedure XASDFRL to re-linkedit a load module containing the old SC4020 package routines, which will be replaced by new ones from SCFOR. JCL and SYSIN information required is very similar to that normally used in the RL standard procedure FHL (or FGL).

Example 1 member OLD4020 (not an OVERLAY program) on the catalogued dataset ULIB.XAXA is to be re-linkedited to form a new member NEW4020 (driving the FR80) on the same dataset.

```
//A3TRANSL JOB (AABB,CC),CHIU,CLASS=X
// EXEC XASDFRL,LIBRARY='ULIB.XAXA',MEMBER=NEW4020
//L.SYSIN DD DSN=ULIB.XAXA(OLD4020),DISP=SHR
```

Example 2 member OLDPLOT of the dataset SURVEY is to be re-linkedited to form a new member PLOTFR80 on the dataset SCAN. Subroutines S1, S2, S3 in OLDPLOT are overlaid against each other and this overlay structure is required in the new load module.

```
// EXEC XASDFRL,PARM.L='OVLY,MAP,LIST,LET'
//L.SYSLMOD DD DSN=SCAN(PLOTFR80),UNIT=DISK30,
// VOL=SER=XA9999,DISP=OLD
//L.LIB DD DSN=SURVEY,VOL=SER=XA9999,UNIT=DISK30,DISP=SHR
//L.SYSIN DD *
 INCLUDE LIB(OLDPLOT)
 OVERLAY ONE
 INSERT S1
 OVERLAY ONE
 INSERT S2
 OVERLAY ONE
 INSERT S3
 ENTRY MAIN
/*
```

bb

SCIENCE RESEARCH COUNCIL

ATLAS COMPUTER LABORATORY

FR80 DISCUSSION PAPER 15

FR80 DRIVER and the ACL SDF Subset

issued by
R W Witty
26 August 1975

DISTRIBUTION:	FRAH	IB	RB	AWB	MFC	PAD
	LOF	AHF	JRG	JWEL	PMN	DVR
	JMR	WDS	RET	RWW	TRP	

1. DRIVER'S TERMS OF REFERENCE

III software is difficult to work with and is unreliable. This situation stems from a lack of source code and documentation, from the poor quality of the code and from III's inadequate response to problems. ACL has therefore decided to replace the FR80 Displayers and Loadgo system with new software (DRIVER) to be written by ACL. The primary goals of this new software are:

(1) Reliability.

(2) Good documentation.

(3) Easy modification and maintenance.

(4) Easy addition of new facilities such as shading and font changing.

(5) Improved error detection and recovery.

(6) Automatic logging to monitor system performance and produce accounting data.

(7) Reduction of multiple displayers to a single program.

The secondary goals are:

(8) Improved performance via a reduction in console interaction, better I/O routines and the detection and optimisation of system bottlenecks.

(9) Portability to other FR80 sites.

(10) Compatability with existing FR80 software. The new ACL software should be built as a standard FR80 applications program and should interface to the existing FR80 environment by, for example, maintaining the existing disc file structure.

(11) Improved operator interface.

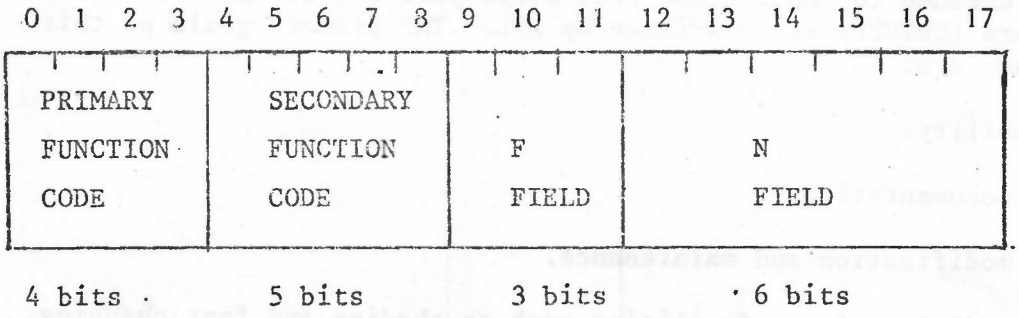
It is envisaged at this early stage that the main benefits of the project will be improved reliability, easier maintenance and the establishment of a firm foundation for future development. Such goals can only be attained by high standards: namely, clear design, disciplined construction and careful documentation.

2. ACL SUBSET OF FR80 SDF ORDERS

The initial version of DRIVER will implement a subset of the FR80 Standard Data Format (SDF) orders as specified below. This will enable graphical jobs produced via the ACL packages to run under the III Displayers and DRIVER without modification.

When this initial stage is complete, further development should take place. Features such as shading and font changing are possibilities. RWW will happily accept comments and suggestions relevant to FR80 software development.

2.1 SDF Instruction Format



(1) Vector orders combine the SFC + F + N fields to give 14-bit operands.

(2) Orders like advance frame combine F + N fields to give 9-bit operands.

2.2 Primary Function Codes

00	III control
04	III continuation
10	III set X, Y abs
14	
20	III variable length commands
24	
30	III set X, Y rel
34	
40	
44	
50	III draw X, Y abs
54	
60	
64	
70	III draw X, Y rel
74	

2.3 Primary and Secondary Function Codes

(1)	000	NO OPERATION
(2)	034	ADVANCE SUB-FRAME
(3)	036	ADVANCE FRAME
(4)	1-- 3--	SET POINT
(5)	5-- 7--	DRAW VECTOR
(6)	204	ENTER TYPE MODE
(7)	212	LINE SPACE
(8)	211	CHAR SPACE
(9)	210	CHAR ROTATE
(10)	224	CHAR HEIGHT
(11)	202	PICTURE
(12)	206	SPOT SIZE
(13)	214	.SELECT COLOUR
(14)	234	SET INTENSITY
(15)	221	CINE/COMIC
(16)	235	PLOT POINT
(17)	222	VECTOR FAMILY

3. SPECIFICATION OF INDIVIDUAL SDF ORDERS

3.1 000 No Operation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	0	0	0	0	0	0	ZEROES								

3.2 034 Advance Sub-Frame

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	0	1	1	1	0	0	n								

Advance 1 sub-frame if n = 0 or 1.

Advance n sub-frames if 1 ≤ n ≤ Smax = manyup x nmax.

Advance sub-frame causes termination of the current 'picture' and initiates a new one, determining the position of the new sub-frame within the current frame, or generating an advance frame as required. For example if manyup is set at four sub-frames per frame then every fourth advance sub-frame order will automatically issue an advance frame.

Note that if advance n sub-frames results in more than nmax (see 036) advance frame orders then the operator must agree to its execution.

3.3 036 Advance Frame

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	0	1	1	1	1	0	n								

Advance 1 frame if n = 0 or 1.

Advance n frames if 1 ≤ n ≤ nmax.

Advance n frames if n > nmax and operator agrees to execute order.

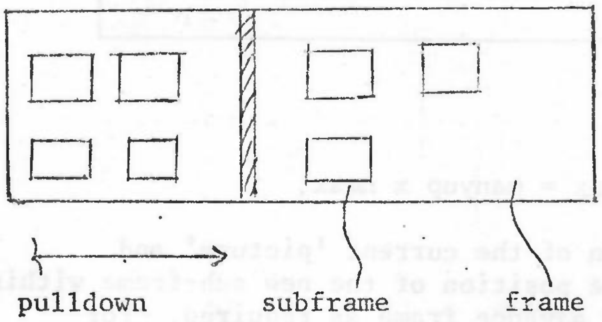
The advance frame order will cause the film to be moved so that a complete blank frame is ready for exposure. The distance moved is the camera pulldown value which is a system variable.

Values of n are checked against an upper limit (nmax), set for each camera, to prevent film wastage. Large amounts of blank film require operator sanction.

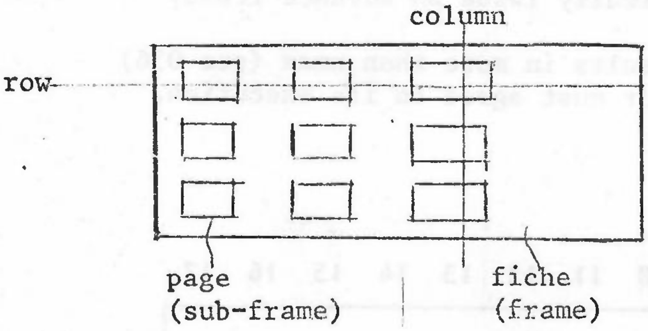
A FRAME is the standard unit of film for a given camera, for example, a 35mm frame or one fiche. The size of the frame is determined by the camera pulldown value.

A SUB-FRAME is a 'picture'. When manyup is in operation several 'pictures' are drawn on the same frame. A sub-frame is indistinguishable from a frame if manyup is off.

When the fiche camera is in operation, the advance frame order causes a new fiche to be started, and advance sub-frame begins a new page within the current fiche.



Two frames with manyup at four sub-frames per frame.



Fiche

3.4 Set Point

3.4.1 100, 300 Set X (one word)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	AR	1	0	14 bit X value													

3.4.2 100, 300 Set Y (one word)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	AR	1	1	14 bit Y value													

3.4.3 100, 300 Set X, Y (two words)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	AR	1	0	14 bit X value													

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	1	14 BIT Y value													

AR = 0	Absolute order
AR = 1	Relative order

Move current point to X,Y without drawing.

3.5 Draw Vector

3.5.1 500, 700 Draw X (one word)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	AR	1	0	14 bit X value													

3.5.2 540, 740 Draw Y (one word)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	AR	1	1	14 bit Y value													

3.5.3 500, 700 Draw X,Y (two words)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	AR	1	0	14 bit X value													

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	1	14 bit Y value													

AR = 0	Absolute order
AR = 1	Relative order

Draw vector from current point to X,Y leaving current point at X,Y.

3.6 204 Enter Type Mode (Nonjustified)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	0	1	0	0	F			ZEROES					

F field	Mode
100	Proportional spacing, regular speed
001	Monospacing, high speed

The first character will be displayed with its lower left corner at the current point. Subsequent characters will be at positions determined by the rotation, spacing, and line feed values. The current position will be unchanged by this command.

Text Format - Text for display on the FR80 will be packed two characters per 18-bit word. They will be in III standard code.

Half-word Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	CHAR. #1									CHAR. #2							
Field 1									Field 2								

If the high-order bit of one of these 9-bit fields is on, the remaining eight bits are a printing character; if the bit is off, the remaining eight bits are a control character. Text will be terminated by an end of message character (203$_8$).

For proportionally spaced text the number entered with the Set Character Spacing (code 211) command is the intercharacter spacing (the distance from the right of one character to the left of the next).

3.7 212 Set Text Line Spacing

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	0	1	0	1	0	Line Spacing	

where 0 ≤ Line Spacing ≤ 510_{10}

OR

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1

0	1	2	3	17
0	0	0	1	Large Line Spacing

where 0 ≤ Large Line Spacing ≤ 16383

The data is a centre line spacing number (from the bottom of one line to the bottom of the next line) to be used in subsequent 204 commands.

3.8 211 Set Character Spacing

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	0	1	0	0	1	Character Spacing	

where 0 ≤ Character Spacing ≤ 510_{10}

OR

Two word format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	1	Large Character Spacing													

where 0 ≤ Large Character Spacing ≤ 16383_{10}

The data is a spacing number (from the left of one character to the left of the next for monospaced text; from the right of one character to the left of the next for proportionally spaced text) to be used in subsequent 204 commands.

3.9 210 Set Character Rotation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	ROTATION		

The data is a rotation number to be used in subsequent text display commands. A rotation value of zero corresponds to upright characters. Successive values are rotated 45° counterclockwise.

3.10 224 Select Character Height

One word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	1	0	1	0	0	Character Height								

where 0 < Character Height ≤ 510_{10}

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1

0	1	2	3	4													17
0	0	0	1	Large Character Height													

where 0 < Large Character Height ≤ 16383_{10}

Note that a character height of zero is not allowed.

3.11 202 Picture

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	0	0	1	0	F			n					

F = 0 define following command sequence as picture n.
F = 3 end definition of picture (n = 77_8) so that end definition is 202377_8.
F = 4 draw picture n.
F = 6 delete definition for picture n.

Definitions may not be nested, but requests to draw a picture may be included within a picture definition. While in process of drawing a picture, the request to draw pictures may be done to a depth of 8. After completing the drawing of a picture, the coordinates (CX and CY) will be reset to the point where they were before the picture was drawn.

The definition of a picture only lasts through the current job.

3.12 206 Set Spot Size

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	0	1	1	0	F			0	0	0	SPOT SIZE		

Bits 15-17 of the data contain the spot size to be used when the filter specified by bits 9-11 is in place. Changing spot size requires about 50 msec.

Where:

F=000 clear; F=001 (unspecified);
F=010 red; F=011 magenta; F=100 green;
F=101 yellow; F=110 blue; F=111 cyan.

3.13 Select Colour

3.13.1 214 Select Secondary Colour (Two word format)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	1	0	0	ZEROES								

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	CYAN COUNT				YELLOW COUNT				MAGENTA COUNT				ZEROES			

3.13.2 214 Select Primary Colour (Two word format)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	1	0	0	ZEROES								

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	BLUE COUNT				GREEN COUNT				RED COUNT				CLEAR COUNT			

3.13.3 214 Select Colour (three word format)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	1	0	0	ZEROES								

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	BLUE COUNT				GREEN COUNT				RED COUNT				CLEAR COUNT			

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	CYAN COUNT				YELLOW COULNT				MAGENTA COUNT				ZEROES			

The counts represent the number of times the film will be exposed with the indicated filter in place. With the subtractive filter system installed, the red, blue and green filters are simulated by inserting two secondary filters simultaneously in the light path.

If the colour select command calls for more than one colour filter per frame, the colour select command acts as a repeat. When a given filter is in place, the intensity and spot size appropriate to that filter, as specified by the 234 and 206 commands, will be in effect.

If the specified count for all filters is zero, then no output will be produced. Multiple hits on the black and white camera are achieved by setting the clear filter hits value to the required number of strikes. The filter counts are initially zero except for clear = 1 which allows black and white jobs to be run on the colour camera.

All lines will be drawn at their specified spot sizes, hits and intensities if a colour job is run on the black and white cameras or hardcopy.

3.14 Set Intensity

3.14.1 234 Set Intensity (one word)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	1	1	0	0	0	INTENSITY							

Bits 10-17 contain the intensity to be used for subsequent commands. This intensity value will be associated with the clear filter. This order is primarily intended for black and white jobs which notionally use the clear filter.

3.14.2 234 Set Intensity (two word format)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	1	1	0	0	1	INTENSITY							

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	ZEROES							F			ZEROES					

F field	Filter
000	Clear
001	Unspecified
010	Red
011	Magenta
100	Green
101	Yellow
110	Blue
111	Cyan

When bit 9 is set, the intensity (bits 10-17) is to be used when the filter specified in the second word is in place. This order is primarily intended for colour jobs. The black and white and hardcopy jobs take their intensity from the clear filter so that they can run successfully using the colour cameras.

3.15 221 Select Cine or Comic Mode

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	0	0	0	1	F			ZEROES					

F field	Mode
100	Cine
000	Comic

3.16 235 Plot Current Point

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	1	1	0	1	ZEROES								

Plot point at current beam position using current intensity and spot size.

3.17 222 Vector Family Order

0	1	2	3	4	5	6	7	8	9 … 17
0	1	0	0	1	0	0	1	0	ZEROES

0	1	2	3	4 … 17
0	0	0	1	No of interpolated lines

Set Point (1 or 2 words, abs or rel)	Optional
Draw vector 1 (1 or 2 words, abs or rel)	
Set Point	Optional
Draw vector 2	

This command conditions the FR80 to interpolate the specified number of lines between the next two vectors produced by SET and DRAW commands. (Those commands may be either one- or two-word type.) Both specified vectors will also be drawn. At the completion of the drawing of these vectors, the current point will be restored to the position it was when the Draw Vector Family command was encountered.

The set points and vectors are positioned as if the Vector Family Order had not been encountered.

bb

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY

ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 16

Graphics Algorithms

issued by
J R Gallop

15 December 1975

DISTRIBUTION:

F R A Hopgood
F Brandwood
I Buchanan
A W Burraston
M F Chiu
P A Dewar
L O Ford
A H Francis

J R Gallop
J W E Lewis
P M Nelson
D Ralphs
D C Sutcliffe
R E Thomas
R W Witty
C D Osland

1. SCOPE OF THE GRAPHICS ALGORITHMS LIBRARY

This paper puts forward proposals concerning the implementation of "graphics algorithms" and comments are invited.

The phrase graphics algorithm will be used loosely to describe routines which implement some technique for visualising data. Such a routine uses a "graphics basic system" like SMOG to actually draw lines and text. Subject areas which have either been implemented or requested include contouring, 3-dimensional objects with hidden-lines removed (Archuleta using Watkins algorithm), projections of world shoreline data, display of a function of 2 variables as histograms or as a grid (hidden-lines removed in both cases), curve-fitting (in 2-dimensions), display of functions of many variables. It will also be useful to add new projections from 3-D to 2-D; at the moment, only perspective is available.

By coordinating these efforts, we can hope to achieve the following:

(a) unified documentation (suggestions in Section 3)

(b) some control over names, where files are stored etc (Section 4)

(c) some control over the content

(d) better use of the special features of the FR80 (for instance, grey-levels, colour, motion)

Special points concerning (c) above. An algorithm should only be included in the library after local testing. Algorithms should, where possible, not use a fixed amount of storage for widely differing problem sizes; the user will as a result be required to specify workspace in call-time argument list; in the case of 2-dimensional arrays (which are to be preferred if the problem is more naturally expressed in this way) an otherwise superfluous argument is necessary in order to dimension the dummy array. A COMMON block can also be used on the 1906A to specify variable storage, but this may cause problems if transferred to a FORTRAN compiler which checks array bounds or if using TRACE 2 on the 1906A.

Any deviations from ANSI FORTRAN should only be made with extreme reluctance and documented in the source prominently. This will greatly ease the transfer of the graphic algorithm library to the future interactive-facility computer and of individual algorithms to the 360/195 on request and to large minis. Two examples: text manipulation will be limited, and no assumptions should be made about initial values of variables except when set in DATA statements.

Exceptions to the standard FORTRAN rules could include:

(1) assembly language routines to perform certain well-defined tasks (for example, the management of an array of bits as logical variables);

(2) dummy subroutine calls (these should be used with care since the intended effects may not be possible on another compiler/loader - a clever optimising compiler may detect that a particular call can never be obeyed and may therefore erase it);

(3) ALGOL!

The question of transmitting information to a remote terminal needs to be considered. It will be necessary to consider ways of compacting the data if the line speed is slow. Two ways are suggested. Firstly, for viewing at a remote terminal, a coarse version of the picture can be sent (the degree of coarseness would be under the user's control). Secondly, a high-level form of the data could be sent (for instance, a list of curve representations) and an intelligent terminal would decode it.

2. SUBJECT AREAS

Of the subject areas listed in Section 1, two will be mentioned here.

Dale Sutcliffe has spent a lot of time assessing contouring algorithms; he has now found several poor ones and one or two good ones. The intention is to implement fairly soon a good contour-following algorithm which uses function values on a rectangular, equally-spaced grid and which can be fairly easily extended to accommodate a skewed grid and/or a grid that is not equally spaced. Contouring between irregularly scattered data points has not been considered yet and the demand for this is uncertain. A routine has been obtained from S Smith of Southampton which assumes that the points are already arranged on a triangular mesh.

Another problem area is how to visualise functions (or data points) which depend on more than 3 independent variables. In particular, the Department of Oceanography at Southampton University wishes to display a function of up to 8 variables derived from experiments on waves and turbulence in the ocean. Any ideas or information about techniques for visualising this sort of data would be appreciated. There are conventional techniques for displaying a function of 2 variables. An extra dimension can be added by motion or by repeating the plots across and down a page (the variable thus expressed would be of very limited range).

Another way is to display small objects apparently embedded in a cuboid of transparent material. The position of each object would represent the 3 independent variables and would be indicated to the viewer by a standard technique like depth-cueing. The data-value at that point would be represented by the size of the object, by its orientation, or by colour.

3. DOCUMENTATION

Documentation should be produced at 2 levels or possibly 3.

The fullest level will contain a description of each subroutine in the library and should enable a user to proceed without further guidance. It will describe all the arguments and common block settings and will provide examples, advice, space and time usage, etc. It should be available in a segmented form since hardly anyone will require the complete set.

The next level will be a summary that could be sent to a wider set of users; its purpose will be to answer "What is available?" instead of "How do I use it?". This level would contain a few lines on every subroutine.

Any further level would simply be publicity and would list the subject areas covered by the library.

4. THE USER INTERFACE

4.1 Library Storage: Names

We should aim to store the algorithms in as few semicompiled libraries as possible (preferably one). It should also be possible to call different algorithms in one row. External names should, therefore, be coordinated, both subroutines and COMMON blocks. There seem to be 2 possible ways.

The first possible method is to allocate systematic (but, as a result, anonymous) names to each algorithm. Within this scheme, we could allocate numbers to routines simply in order of addition to the library, like the GROATS library which starts with GR2(texture) and currently ends at GR15(revcounter). Alternatively, a sophisticated hierarchical classification could be used like the Harwell or the NAG library, but this seems unnecessarily complicated for the somewhat smaller library that we envisage.

The advantage of a systematic scheme is that it is easier for a user to avoid name clashes with both present and future algorithms. If a systematic scheme is preferred, the following is suggested. The naming scheme would include all external names: all subroutines documented and made available to users (the top level), all subroutines called by the top level other than those which already exist in the SMOG, SPROGS or SCSIM library and all COMMON blocks. Each name would be of the form:

$gc_1n_1n_2c_2$

where

c_1 = A top-level user-callable algorithms
B all routines called by the top-level
C COMMON blocks

n_1n_2 = 2-digit number starting from 01

c_2 = a letter or nothing. This would indicate a trivial variant.

In normal operation, 3 lists would be kept corresponding to the possible initial letter-pairs GA, GB and GC. No attempt would be made to keep the 3 lists in step, since the interactions between the routines may make this complicated. So algorithm GA03 may well call subroutines GB05,GB06,GB07 and use COMMON blocks GC10A,GC10B,GC10C. The user would call GA03, but also needs to know that his names should avoid (GA or GB or GC) followed by 2 digits followed by anything else.

The second possibility is to allow names to be chosen by the programmer and the coordinator checks that no new library item clashes with existing contents of the library or with SMOG or SPROGS. Alphabetical lists would have to be maintained and updated in order to keep this process relatively painless. If several different routines do a similar job (for instance, contouring), then these would be systematized (eg CONT1, CONT2, CONT3 etc).

To illustrate the first possible method, the Archuleta package (FR80 User Note 10) would require 21 routine names beginning with GA, over 20 beginning with GB and 27 COMMON block names beginning with GC.

4.2 Options and Defaults

In some graphics algorithms, the number of options provided for the user can be large. For instance, a contouring algorithm can allow the mesh to be skewed or not and the grid to be equally spaced or not; there are many possible labelling options and contours can also be distinguished by line-style (darkness or colour); a border can be drawn and grid intersection points can be marked.

The user should be provided with a clean way of specifying options or of allowing defaults to take effect. The best way would be to use BLOCK DATA to initialise a COMMON block and include this in a library, expecting this load module to be fetched when and only when the COMMON block is referenced. However, this does not happen on the 1906A. There are several other possible ways of specifying defaults.

(1) Use BLOCK DATA and store it in its own semicompiled file; must be quoted as SEMI by the user in his macro in addition to quoting the library itself as LIB. The cost is an unnecessary proliferation of small files.

(2) The user must call a subroutine to initialise the defaults before he calls the algorithm itself. If he does not call the provided initialising subroutine, all the components of the block must be set individually in his program.

(3) Use 2 levels of subroutine, like the Archuleta package. A higher level subroutine automatically assigns values to the COMMON block; one of a lower level puts the responsibility on the user.

(4) One subroutine call with a long argument list. The documentation should make recommendations about options which the user is not concerned about.

(5) Implement the effect of BLOCK DATA in a PLAN subroutine, to which there must be a CALL which is never actually obeyed.

gm

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 17

FR80 DRIVER - Initial Set of Text Processing Orders

issued by
R W Witty

7 March 1977

DISTRIBUTION:

F R A Hopgood
M F Chiu
D A Duce
A H Francis
D Ralphs

A W Burraston
P A Dewar
R W Witty
P M Nelson
K M Crennell

1. INTRODUCTION

This brief note outlines the DRIVER Prototype's text processing orders. Comments and suggestion for further orders are welcome. These orders form an extension to the set detailed in FR80 Discussion Paper 15.

1.1 204 Enter Type Mode (Nonjustified)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
									M	6,9	S	B	8,9				
0	1	0	0	0	0	1	0	0	P	8	F	E	6	0	0	0	0

The first character will be displayed with its lower left corner at the current point. Subsequent characters will be at positions determined by the rotation, spacing, and line feed values. The current position may be changed by this command.

bit 9

1 means output characters with proportional spacing.
0 means output characters with mono spacing.

bit 10

1 means characters in 8 bit format
0 means character in 6 or 9 bit formats (see bit 13)

bit 11

1 means output text at high speed (if high-speed page print option is available).
0 means output text at regular speed.

bit 12

1 means update current position to end of last character printed.
0 means current position remains unchanged.

bit 13

1 means characters in 6 bit format
0 means characters in 8 or 9 bit format (see bit 10)

NOTES:

If bit 12 is set to 1, the justify elements will not be initialised before printing. This means that a single justify command (code 33_8) may be used for multiple piece lines.

With graphic arts characters, the position after print will be the integer portion of the DACs (full scope point values); the fractional portion is lost.

1.2 Text Formats

6 bit format:

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td></tr>
<tr><td colspan="6">Char #1</td><td colspan="6">Char #2</td><td colspan="6">Char #3</td></tr>
</table>

8 bit format:

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td></tr>
<tr><td>0</td><td>0</td><td colspan="8">Char #1</td><td colspan="8">Char #2</td></tr>
</table>

9 bit or Half-word Format:

0	1 2 3 4 5 6 7 8	9	10 11 12 13 14 15 16 17
	CHAR. #1		CHAR. #2
	Field 1		Field 2

If the high-order bit of one of these 9-bit fields is on, the remaining eight bits are a printing character; if the bit is off, the remaining eight bits are a control character. Text will be terminated by an end of message character (203_8).

For proportionally spaced text the number entered with the Set Character Spacing (code 211) command is the intercharacter spacing (the distance from the right of one character to the left of the next).

1.3 Select or Modify Font

Overall structure of this class of orders is

2	3	2	F	N

where F = 0.....Select Font
F = 1.....Set/unset escape-action
F = 2.....Modify Dispatch Base

1.3.1 2320nn Nonstandard Select Font

0 8	9 10 11	12 17
010 011 010	000	N

N is the Font Number to be selected

1.3.2 232000 Font Selection (III SDF)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	TANGENT OF ITALIC ANGLE (Binary point to the left of bit 2)															

This word is optional- the font will be displayed without obliqueness if this word is omitted or if the selected font will not permit italicisation.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	CHARACTER ASPECT RATIO x 10000															

This word is optional. It is used to permit expanding or condensing characters. Values less than 10000 (decimal) result in a condensed font; values more than 10000 result in an expanded font. If equal to 10000, or omitted, or a non-graphic arts font specified, the font will be used as designed.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	FONT NUMBER															

1.3.3 232100 Set/unset escape-actions

Currently there are 13_8 actions which may be triggered by escape characters. They are:

01 .... End-of-text
02 .... Newline
03 .... Line Feed
04 .... Carriage return
05 .... Space
06 .... Backspace
07 .... Null character
10 .... Bad character
11 .... Permanent Modification to Dispatch Base
12 .... Toggle Modification to Dispatch Base
13 .... Escape character, next byte in string is an action number not a character.

Further actions may easily be added. Any suggestions?

Set/unset orders have the following general format:

0 2 3 8 9 10 11 12 17

2	3 2	1	0	0
0	action code		character code	
parameters or null				

Set character code to be escape-action

2 3 2	1	0 0
0	action code	char code

Unset character code from being escape-action

2 3 2	1	0 0
0	0 0	char code

Setup character to cause permanent modification to Dispatch Base

2 3 2	1	0 0
0	1 1	char code
2's complement integer		

Setup character to cause toggle modification to Dispatch Base

2 3 2	1	0 0
0	1 2	
2's complement 18 bit integer		

1.3.4 232200 Immediate permanent modfication to Dispatch Base

2 3 2	2	0 0
2's comp. 18 bit integer		

The parameter will be added to the current value of the Dispatch Base immediately. Reset if param=0

bb

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 18

FR80 DRIVER - Initial set of Monitor Commands

issued by
R W Witty

8 March 1977

DISTRIBUTION:

DISTRIBUTION:	F R A Hopgood	A W Burraston
	M F Chiu	R W Witty
	A H Francis	P M Nelson
	D Ralphs	R Brandwood

1. FR80 DRIVER MK1 MONITOR COMMANDS

The following list is a proposal for the Monitor Command repertoire of the MK1 release of FR80 DRIVER. The commands will operate exactly as specified in the FR80 Operator's Manual. Comments and suggestions will be welcomed.

1. ADVANCE
2. CAMERA
3. CONTINUE
4. CTRL-A
5. CTRL-I
6. END JOB
7. GO
8. HITS
9. MAKE FILM
10. PARITY
11. PULLDOWN
12. REWIND
13. ROTATED
14. SET SIZE
15. SKIP
16. SPEED
17. TAPE TYPE
18. TITLE
19. TRY AGAIN
20. X NUM
21. Y NUM

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 19

FR80 DRIVER - Mark 2 Text Processing Orders

issued by
R W Witty

22 April, 1977

DISTRIBUTION:

F R A Hopgood
M F Chiu
D A Duce
A H Francis
D Ralphs
D C Sutcliffe
A W Burraston
R Brandwood
R W Witty
P M Nelson
K M Crennell
C Osland
C Cooper

1. INTRODUCTION

This note proposes a set of text processing orders for the Mark 2 version of DRIVER. The orders are intended to provide a powerful contribution to the overall text processing facilities of the Laboratory. This is only a proposal; comments and suggestions are welcomed.

The major new features proposed are underlining, automatic emboldening of characters, a powerful super/subscripting mechanism, tabs and a spacing-for-justification mechanism.

Section 2 gives exact details of the SDF orders involved with text processing whilst section 3 outlines very briefly how various features work.

1.1 Summary of Text Processing Facilities

1)	Mono or proportional spacing	
2)	6, 8, 9 bit formats	1,2
3)	Fast or slow speed plotting	1,2
4)	Beam to beginning/end of string	1,2
5)	Software simulator	1,3
6)	Large characters	1
7)	Select font	1
8)	Escape action mechanism	1,2
9)	Bold characters	1,2
10)	Underlining	1,2
11)	Tabs	1,2
12)	Superscript/subscript	1,2
13)	Save/set point	1,2
14)	Large (>256 characters) fonts	1,2
15)	Variable size characters	

Notes

1) Not available in III Displayer
2) Not available in Atlas Displayer
3) Cannot be explicitly selected in Atlas Displayer

2. MK 2 SDF ORDERS

2.1 204 Enter Type Mode (Nonjustified)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	0	1	0	0	M P	6,9 8	S F	B E	8,9 6	S H	0	0	0

The first character will be displayed with its lower left corner at the current point. Subsequent characters will be at positions determined by the rotation, spacing, and line feed values. The current position may be changed by this command.

bit 9

1 means output characters with proportional spacing.
0 means output characters with mono spacing.

bit 10

1 means characters in 8 bit format
0 means character in 6 or 9 bit formats (see bit 13)

bit 11

1 means output text at high speed (if high-speed page print option is available).
0 means output text at regular speed.

bit 12

1 means update current position to end of last character printed.
0 means current position remains unchanged.

bit 13

1 means characters in 6 bit format
0 means characters in 8 or 9 bit format (see bit 10)

bit 14

1 means software simulator used even if height with hardware range so that height used is precise as opposed to intervals 0-63.
0 means will use hardware if possible.

Text Formats

6 bit format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Char #1						Char #2						Char #3					

8 bit format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	Char #1								Char #2							

9 bit or Half-word Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	CHAR. #1									CHAR. #2							
Field 1									Field 2								

If the high-order bit of one of these 9-bit fields is on, the remaining eight bits are a printing character; if the bit is off, the remaining eight bits are a control character. Text will be terminated by an end of message character (203_8).

For proportionally spaced text the number entered with the Set Character Spacing (code 211) command is the intercharacter spacing (the distance from the right of one character to the left of the next).

2.2 212 Set Text Line Spacing

One word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	0	1	0	1	0	Line Spacing								

where $0 \leq$ Line Spacing $\leq 510_{10}$

OR

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1

0	1	2	3														17
0	0	0	1	Large Line Spacing													

where $0 \leq$ Large Line Spacing ≤ 16383

The data is a centre line spacing number (from the bottom of one line to the bottom of the next line) to be used in subsequent 204 commands.

2.3 211 Set Character Spacing

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	0	1	0	0	1	Character Spacing	

where 0 ≤ Character Spacing ≤ 510_{10}

OR

Two word format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	1	Large Character Spacing													

where 0 ≤ Large Character Spacing ≤ 16383_{10}

The data is a spacing number (from the left of one character to the left of the next for monospaced text: from the right of one character to the left of the next for proportionally spaced text) to be used in subsequent 206 commands.

2.4 210 Set Character Rotation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	ROTATION		

The data is a rotation number to be used in subsequent text display commands. A rotation value of zero corresponds to upright characters. Successive values are rotated 45^{o} counterclockwise.

2.5 224 Select Character Height

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	1	0	1	0	0	Character Height	

where 0 ≤ Character Height ≤ 510_{10}

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1

0	1	2	3	4	17
0	0	0	1	Large Character Height	

where 0 ≤ Large Character Height ≤ 16383_{10}

Note that a character height of zero is not allowed.

2.6 232 Select or Modify Font

Overall structure of this class of orders is

2	3	2	F	N

where F = 0.....Select Font
F = 1.....Set/unset escape-action
F = 2.....Modify Dispatch Base
F = 3.....Reset Dispatch Base
F = 4.....Set Escape-action parameters

2.6.1 232000 Font Selection (III SDF)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	TANGENT OF ITALIC ANGLE (Binary point to the left of bit 2)															

This word is optional- the font will be displayed without obliqueness if this word is omitted or if the selected font will not permit italicisation.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	CHARACTER ASPECT RATIO x 10000															

This word is optional. It is used to permit expanding or condensing characters. Values less than 10000 (decimal) result in a condensed font; values more than 10000 result in an expanded font. If equal to 10000, or omitted, or a non-graphic arts font specified, the font will be used as designed.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	FONT NUMBER															

2.6.2 2320nn Nonstandard Select Font

0 8	9 10 11	12 17
010 011 010	000	N

N is the Font Number to be select N>0.

This is a non standard order. It should not be used. It has been included only for compatability with an Atlas modification to the III Displayer.

2.6.3 232100 Set/unset escape-actions

The following actions may be triggered by escape characters. They are:

01 .... End-of-text
02 .... Newline
03 .... Line Feed
04 .... Carriage return
05 .... Space
06 .... Backspace
07 .... Null character
10 .... Bad character
11 .... Permanenet Modification to Dispatch Base
12 .... Toggle Moddification to Dispatch Base
13 .... Escape character, next byte in string is an action number not a character
14 .... (reserved for system use, actually end-of-buffer action)
15 .... Bold on/off
16 .... Underline on/off
17 .... Tab to next stop
20 .... Superscript
21 .... Subscript
22 .... Super/subscript lock on/off
23 .... Save current position
24 .... Set point

Set/unset orders have the following general format:

<table>
<tr><th>0 2</th><th>3 8</th><th>9</th><th>10 11</th><th>12</th><th>17</th></tr>
<tr><td>2</td><td>3 2</td><td colspan="2">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>6 bit action code</td><td>0</td><td colspan="3">8 bit character code</td></tr>
</table>

Unset Character code from being escape-action

2 3 2	1	0 0
0	0 0	char code cc

The Dispatch Table entry for character code cc will be modified to return cc to be a normal plotting character. An unset must only be issued for a character code which actually has a Character Stroke Definition associated with it.

Action 01. End-of-text

2 3 2	1	0 0
0	0 1	cc

The character code cc causes the plotting of text to cease, ie cc must be the last byte in the string. The rest of the word in which the end-of-text cc resides will be ignored.

Action 02. Newline

2 3 2	1	0 0
0	0 2	cc

The occurrence of cc in a text string causes a newline to be started, determined by the 212 order and the initial starting position of the text.

Action 03. Line Feed

2 3 2	1	0 0
0	0 3	cc

The occurrence of cc in a text string causes the beam to be moved down onto the next line, determined by the 212 order.

Action 04. Carriage Return

2	3	2	1	0	0
0	0	4	cc		

The occurrence of cc in a text string causes the beam to be returned to the start of the current line.

Action 05. Space

2	3	2	1	0	0
0	0	5	cc		

The occurrence of cc in a text string causes a space to the plotted. The width of the space is set by the 211 order.

Action 06. Backspace

2	3	2	1	0	0
0	0	6	cc		

The occurrence of cc in a text string will cause the beam to be backspaced to the beginning of the last character plotted. Any number of backspaces may be successfully performed if the text is monospaced. Under proportional spacing multiple backspaces will not give the intuitive result as only the width of the last character plotted is remembered. Should multiple backspaces under proportional spacing be necessary then this feature could be incorporated at an added cost in Driver code.

Action 07. Null Character

2	3	2	1	0	0
0	0	7	cc		

The occurrence of cc in a text string is completely ignored. Driver does have a mechanism to count the number of null characters plotted however.

Note that the current III Displayer has a bug when using null characters with proportional spacing - it proportionally spaces null character so plotting a null character will actually advance the beam by an amount set by the 211 order.

Action 10. Bad Character

2	3	2	1	0	0
0	1	0	cc		

The occurrence of cc in a text string will cause a small rectangular box to be plotted which is proportional to the current character height. This mechanism allows bad characters to be spotted in the output. Driver also has a mechanism to count the number of bad characters plotted.

Action 11. Permanent modification to the Dispatch Base

2	3	2	1	0	0
0	1	1	cc		

The occurrence of cc in a text string will cause the permanent modification parameter to be added to the current value of the Dispatch Base, thus causing all successive bytes to access a new section of the Dispatch Table. This feature should be used with great care. The Permanent Modification parameter is set up by the 232406 order.

Action 12. Toggle Modification to the Dispatch Base

2	3	2	1	0	0
0	1	2	cc		

The first and following odd occurrences of cc in a text string will cause the Toggle Modification parameter to be added to the current value of the Dispatch Base. Second and even occurrences cause the subtraction of the parameter. This mechanism is intended to facilitate case changing. The Toggle Modification Parmeter is set up by the 232407 order.

Action 13. Escape Character

2 3 2	1	0 0
0	1 3	cc

The occurrence of cc in a text string indicates that the next byte is not a plottable character but is the integer number of an action code, ie if the user does not wish to dedicate a specific character code for each individual action he can nominate one character to act as an escape character which will be followed by which ever action code he requires.

Action 14. End-of-buffer

This action is used internally by Driver and must not be used by the user.

Action 15. Bold

2 3 2	1	0 0
0	1 5	cc

The occurrence of cc in a text string causes the emboldening of characters to be toggled. The initial setting is OFF. The parameters associated with emboldening are set by the 232404 order.

Action 16. Underline

2 3 2	1	0 0
0 1 6		cc

The occurrence of cc in a text string causes the underlining of characters to be toggled. The initial setting is OFF. The parameters associated with underlining are set by the 232403 order.

Action 17. Tab

2 3 2	1	0 0
0	1 7	cc

The occurrence of cc in a text string causes the beam to be moved to the next tab position as set up by the 232402 order.

Action 20. Superscript

2 3 2	1	0 0
1	2 0	cc

The occurrence of cc in a text string causes the beam position to be moved to the next superscript position. See section 3.2 for fuller description.

Action 21. Subscript

2 3 2	1	0 0
0	2 1	cc

The occurrence of cc in a text string causes the beam position to be moved to the next subscript position. See section 3.2 for fuller description.

Action 22. Super/Subscript lock

2 3 2	1	0 0
0	2 2	cc

The occurrence of cc in a text string causes the super/subscript lock to be toggled. The initial setting is OFF. See section 3.2 for a fuller description.

Action 23. Save current position

2 3 2	1 0 0	
0	2 3	cc

The occurrence of cc in a text string causes the current beam coordinates to be saved as Set Point orders in the Position Table. (See section 3.2); the table position is given by the next byte.

Action 24. Set Point

2 3 2	1	0 0
0	2 4	cc

The occurrence of cc in a text string causes the current beam position to be moved according to the Set Point orders contained in the Position Table (see section 3.2), according to the value of the next byte.

2.6.4 232200 Immediate permanent modification to Dispatch Base

2 3 2	2	0 0
2's comp. 18 bit integer		

The parameter will be added to the current value of the Dispatch Base immediately.

2.6.5 232300 Reset Dispatch Base

2 3 2	3	0 0

Dispatch Base is reset to its initial value ie the address of first entry in the Dispatch Table.

2.6.6 232400 Setup Escape-action parameters

This class of orders sets up the parameters for those escape-action mechanisms which allow text driven modifications to the character plotting. Their overall structure is:

2 3 2	4	N

where:

N = 01 ...... Set Point
N = 02 ...... Tab Stops
N = 03 ...... Underline
N = 04 ...... Bold
N = 05 ...... Super/subscripting
N = 06 ...... Permanent Dispatch Modification
N = 07 ...... Toggle Dispatch Modification

232401 Set Point Parameters

2 3 2	4	0 1
index		
Set Point X abs/rel		
Set Point Y abs/rel		

The two Set Point orders will be entered into Position Table (index). The orders may be absolute or relative out must be single word orders only.

232402 Tab Stops

2 3 2	4	0 2
index		
X coordinate (abs)		

The absolute value of the X coordinate will be entered into Tab Stop Table (index).

232403 Underline

2 3 2	4	0 3
X offset		
Y offset		
S 0 0 0 0 0		

S = 4 ...... Full underlining (including inter character spacing)
S = 0 ...... Intercharacter spacing not underlined

The starting position of the underline is the starting position of the character offset by (X,Y) as specified by the 241300 order. The line will always extend the full width of the character, but may optionally underline the intercharacter space if proportional spacing is active. Underlining is controlled by action 16.

232404 Bold

2 3 2	4	0 4
repeat factor RF		
rel X offset		
rel Y offset		

Each character will be drawn RF times, being incrementally offset by (X,Y) each time. Emboldening is controlled by action 15.

232405 Super/subscripting

2 3 2	4	0 5
index		
rel Y offset		
ch height		
ch space		

The parameters are entered into Subscript Table (index). See section Super/subscripting is controlled by actions 20, 21, 22.

232406 Permanent Dispatch Base Modification

2 3 2	4	0 6
2's complement integer		

An integer value which will be added to the current value of the Dispatch Base. See action 11.

232407 Toggle Dispatch Base Modification

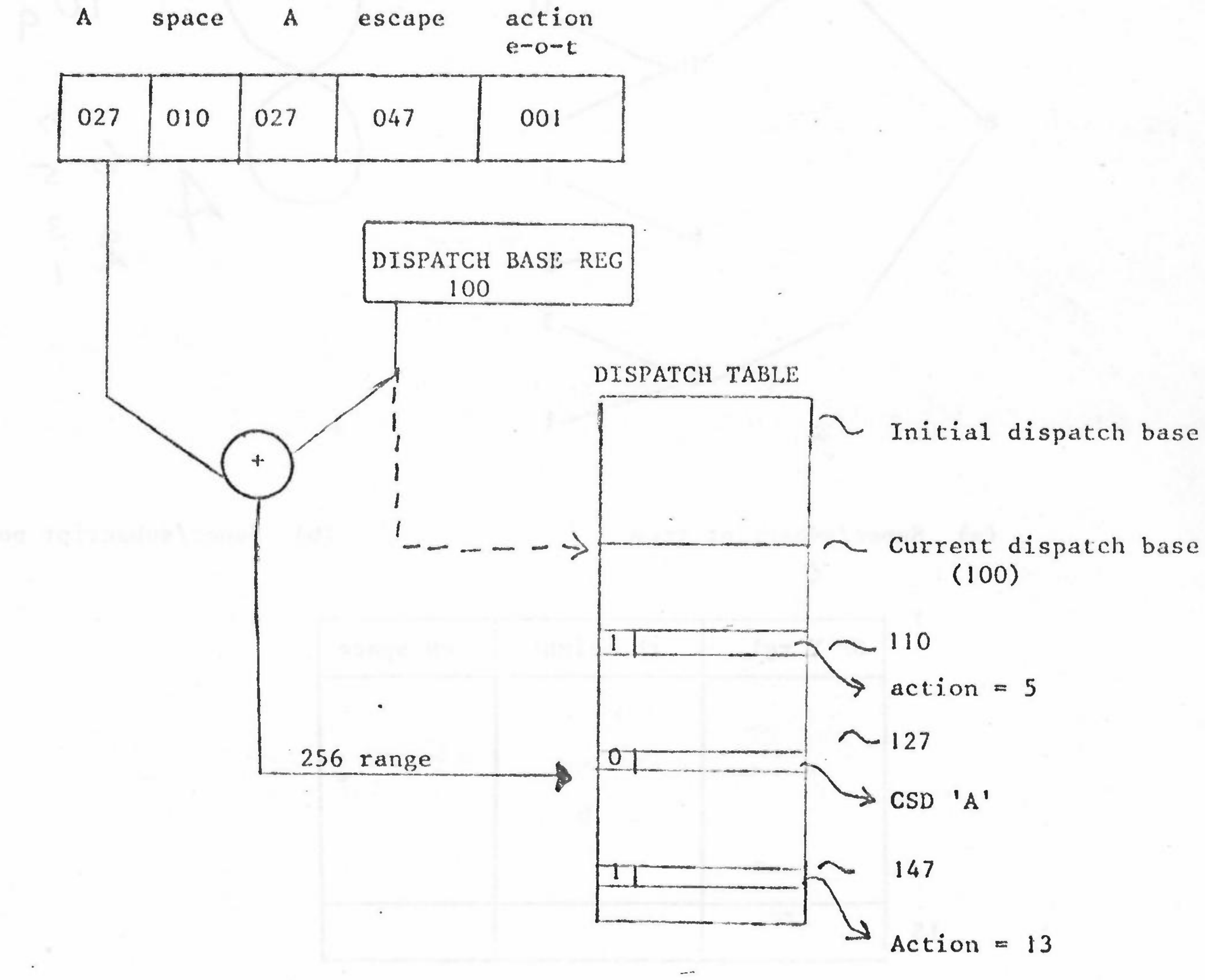
2 3 2	4	0 7
2's complement integer		

An integer value which will be alternately added and subtracted from the current value of the Dispatch Base on encountering an action 12 code.

3. BRIEF EXPLANATORY NOTES

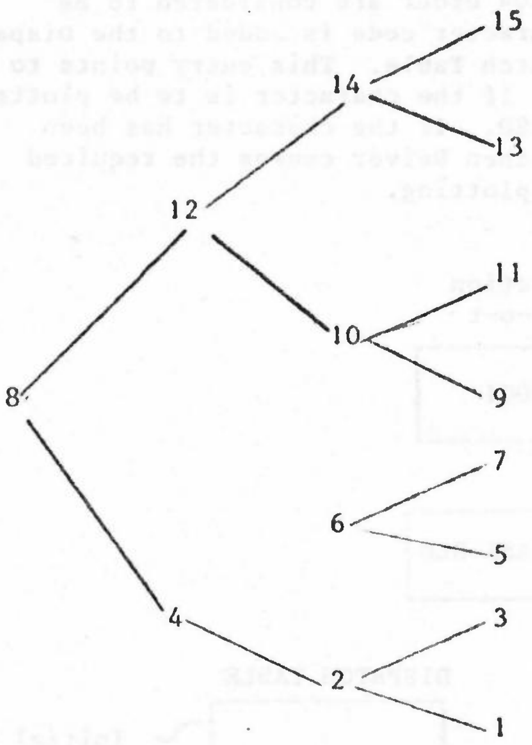
3.1 The character plotting mechanism

All words immediately following a 204 order are considered to be plottable characters. A byte's character code is added to the Dispatch Base to find its entry in the Dispatch Table. This entry points to the Character Stroke Definition (CSD). If the character is to be plotted, then the hardware interpretes the CSD. If the character has been nominated as an escape action code then Driver causes the required action to occur and then continues plotting.

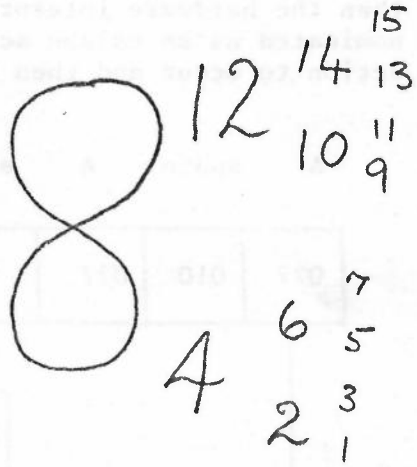


Result: A A

3.2 Super/subscripting, Tabs Set Points



(a) Super/subscript tree



(b) Super/subscript positions

1	SP Y rel	ch height	ch space
15			

(c) Super/subscript parameter table.

1	abs SPX rel	abs SPY rel
n		

(d) Save/Set Point Table

1	SPX abs
P	

(e) Tab Stops Table

A general super/subscripting mechanism is proposed here. Sequences of action codes 20 and/or 21 (denoted ↑ and ↓ respectively) cause the beam to be moved according to the Set Point Y relative order stored in the corresponding position in the super/subscript parameter table (c). Sequences containing ≤ 3 ↑ ↓ may preceed a character. It is envisaged that the table (c) will be set up so that the ↑ ↓ actions step through the notional binary tree (a) thus giving the possible super/subscripts detailed in (b). The limit of depth 3 will be a Driver compile time parameter. When the next plottable character succeeding the ↑ ↓ sequence is found it is plotted at the new point with the corresponding character height and spacing supplied in the table (c) in raster units. Immediately after plotting the beam position will be 'brought down' to the string's base line ie the X coordinate will be unchanged but the Y coordinate will be reset to its value previous to the ↑ ↓ sequence. If a string of characters forms a super/subscript then the super/subscript lock action (22) may be used to inhibit the returning of the Y coordinate value. Toggling the lock will return the Y coordinate to the base level. The values in the Super/subscript parameter table are set by the 232405 order.

The Save Point mechanism (action 23) allows the user to save the current position in the Save/Set Point table (d). Any time later action 24, the Set Point mechanism, may be used to restore this position, thus allowing superscripts to be plotted directly above subscripts (see examples). The number of available entries in the Save/Set Point table will be a Driver compile time option.

Entries in the Save/Set Point table may be preset by use of the 232401 order. As these orders may be relative as well as absolute, this mechanism can be used to allow several different width spaces. If the table has several entries with SPX relative and SPY relative = 0 orders then different spacings may be obtained using the action 24 code.

Actions 23 and 24 are two bytes long in a text string. The first byte is the character code which has been nominated to cause the action and the second byte is the index of the required table entry. In the examples Save Point is represented by SV and Set Point by SP.

The 232402 order may be used to set up a series of absolute X coordinates in raster units to be used as Tab Stops, table (c). Action 17 will cause the current position to be moved to the next greatest X coordinate in the table, such that tab stop$_n$ ≤ C.P. < tab stop$_{n+1}$ causes the C.P. to be moved to tab stop$_{n+1}$. The Y coordinate is unchanged. If tab stop$_{n+1}$ does not exist the beam will not be moved and a warning message will be issued.

Subscripting examples

$$2^{a^{3}_{max}}_{b^{2}_{min}}$$

$2 \quad \textcircled{SV}\ 1 \quad \downarrow b \quad \downarrow\uparrow 2 \quad \downarrow\downarrow\textcircled{L}\ \text{min}\ \textcircled{L}$

$\textcircled{SP}\ 1 \quad \uparrow a \quad \uparrow\uparrow 3 \quad \downarrow\downarrow\textcircled{L}\ \text{max}\ \textcircled{L} \quad \textcircled{eot}$

$$A = \int \frac{(I_A I_B)^{1/2}}{I_{A_i} o(t)} \ at$$

$A\ \nabla\ =\ \nabla \int \nabla \quad \textcircled{SV}\ 1$

$\textcircled{U}\ \ \textcircled{SP}\ 2\ (I\ \ \downarrow A\quad I\ \ \downarrow B)\ \uparrow\textcircled{L}\ 1/2\ \textcircled{L} \qquad \textcircled{U}$

$\textcircled{SP}\ 1\ \ \textcircled{SP}\ 3\ \ I\ \ \downarrow A\quad \downarrow\downarrow i\quad \uparrow\uparrow o\quad (t)\nabla\nabla\nabla\nabla\quad \textcircled{SP}\ 4\ \nabla dt\quad \textcircled{eot}$

bb

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 20

FR80 Driver - MK 2 Super/Subscripting

issued by
R W Witty

10 May 1977

DISTRIBUTION:

F R A Hopgood
M F Chiu
D A Duce
A H Francis
D Ralphs
D C Sutcliffe
A W Burraston
R Brandwood
R W Witty
P M Nelson
K M Crennell
C Osland
C Cooper

1. MK 2 SUPER/SUBSCRIPTING AMENDMENT

A disadvantage of the parameterised super/subscripting proposed in FR80 Discussion Paper 19 is that the mechanism is not independent of character height. This means that a sensible default cannot be established and that the parameters must be reset each time a new character height is selected. To overcome these problems a new parameter system based on relative heights and spacings is proposed below. The absolute value parameters set by the 232405 order (page 16) are replaced by fractions which multiply the currently selected character height or space to produce absolute values. The format of the 232405 order thus becomes an eight word order:

2 3 2 4 0 5
.index
rel. Y offset top
bottom
ch. height top
bottom
ch. space top
bottom

For example, if the currently selected character height is 100 rasters and rel.Y offset top = +2 and rel Y offset bottom = -3 then the super/subscript at this level would appear (100*(+2))/(-3) = -66 rasters from the current position ie 66 rasters below the base line. Super/Subscript offsets and heights will be a proportion of the currently selected character height and super/subcript spacing will be a proportion of the current character spacing value.

bb

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

See over

FR80 DISCUSSION PAPER 21

Rationalisation of accounting (SYSLOG)
numbers for the FR80

issued by
D Ralphs

30 June 1977

DISTRIBUTION:

F R A Hopgood
A H Francis
D C Sutcliffe
A W Burraston
R Brandwood
P C Thompson
A T Lea
M F Chiu
D A Duce
D Ralphs
C D Osland
R W Witty
P M Nelson
R E Thomas

1. INTRODUCTION

The FR80 logging/accounting codes for the host machine and for the generating package are not fully utilised yet. These codes are part of the GSTART record sent to the FR80, see FR80 Technical Paper 4. The two major machines generating FR80 jobs (the RL 1906A and the RL 360/195) differ in the few codes that they currently send, hence the following proposals to rationalise these codes.

When these codes are implemented in; the various packages, SYSLOG for FR80 Driver Mark 2 and the FR80 accounting program, it should be possible to determine where the FR80 work is generated.

2. HOST MACHINE NUMBERS

2.1 As Defined in FR80 Technical Paper 4

0	RL 1906A
1	RL 360/195
-1	'Other'

2.2 Proposed Codes

-1	'Other'
0	RL 1906A
1	RL 360/195
2	ACD Prime
3	ACD 4070
4	Edinburgh PDP10
5	UMIST PDP10
6	Oxford 1906A
7	Reading 1904S
8	Sheffield 1906S
9	Harwell 370/168
10	Daresbury 370/165
11	Cambridge 360/..
12	RL 4080

3. PACKAGE NUMBERS

3.1 As Defined in FR80 Technical Paper 4

0	SMOG
1	SPROGS
2	GROATS
-1	'Other'

3.2 Proposed Codes

-1	'Other'
0	SMOG
1	SPROGS
2	GROATS
3	SCFOR
4	MUGWUMP
5	HPLOT/HBOOK
6	GINO-F
7	POLYGRAPHICS
8	FLIST
9	DRAW
10	SLIPS
11	COM (SYSOUT=M)
12	FINGS
13	GHOST
14	Harwell FR80 Package
15	DRPLOT

1b

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

amended

FR80 DISCUSSION PAPER 21

Rationalisation of accounting (SYSLOG) numbers for the FR80

issued by
D Ralphs

30 June 1977

DISTRIBUTION:

F R A Hopgood
A H Francis
D C Sutcliffe
A W Burraston
R Brandwood
P C Thompson
A T Lea

M F Chiu
D A Duce
D Ralphs
~~C D Osland~~
R W Witty
P M Nelson
R E Thomas

1. INTRODUCTION

The FR80 logging/accounting codes for the host machine and for the generating package are not fully utilised yet. These codes are part of the GSTART record sent to the FR80, see FR80 Technical Paper 4. The two major machines generating FR80 jobs (the RL 1906A and the RL 360/195) differ in the few codes that they currently send, hence the following proposals to rationalise these codes.

When these codes are implemented in; the various packages, SYSLOG for FR80 Driver Mark 2 and the FR80 accounting program, it should be possible to determine where the FR80 work is generated.

2. HOST MACHINE NUMBERS

2.1 As Defined in FR80 Technical Paper 4

0	RL 1906A
1	RL 360/195
-1	'Other'

2.2 Proposed Codes

-1	'Other'
0	RL 1906A
1	RL 360/195
2	ACD Prime
3	ACD 4070
4	Edinburgh PDP10
5	UMIST PDP10
6	Oxford 1906A
7	Reading 1904S
8	Sheffield 1906S
9	Harwell 370/168
10	Daresbury 370/165
11	Cambridge 370/165
12	~~RL 4080~~ C + A Graphics 4080/A
13	" " " /B

3. PACKAGE NUMBERS

3.1 As Defined in FR80 Technical Paper 4

0	SMOG
1	SPROGS
2	GROATS
-1	'Other'

3.2 Proposed Codes

-1	'Other'
0	SMOG
1	SPROGS
2	GROATS
3	SCFOR
4	MUGWUMP
5	HPLOT/HBOOK
6	GINO-F
7	POLYGRAPHICS
8	FLIST
9	DRAW
10	SLIPS
11	COM (SYSOUT=M)
12	FINGS
13	GHOST
14	Harwell FR80 Package
~~15~~	~~DRPLOT~~
15	GPGS

1b

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 22

FR80 DRIVER - Mark 2 Text Processing Orders
Second Iteration

issued by
R W Witty
25 May 1977

DISTRIBUTION:

F R A Hopgood
M F Chiu
D A Duce
A H Francis
D Ralphs
D C Sutcliffe

A W Burraston
R Brandwood
R W Witty
P M Nelson
K M Crennell
C Osland
C Cooper

1. INTRODUCTION

This note proposes a set of text processing orders for the Mark 2 version of DRIVER. The orders are intended to provide a powerful contribution to the overall text processing facilities of the Laboratory. This is only a proposal; comments and suggestions are welcomed.

The major new features proposed are underlining, automatic emboldening of characters, a powerful super/subscripting mechanism, tabs and a spacing-for-justification mechanism.

Section 2 gives exact details of the SDF orders involved with text processing whilst section 3 outlines very briefly how various features work.

1.1 Summary of Text Processing Facilities

1)	Mono or proportional spacing	
2)	6, 8, 9 bit formats	1,2
3)	Fast or slow speed plotting	1,2
4)	Beam to beginning/end of string	1,2
5)	Software simulator	1,3
6)	Large characters	1
7)	Select font	1
8)	Escape action mechanism	1,2
9)	Bold characters	1,2
10)	Underlining	1,2
11)	Tabs	1,2
12)	Superscript/subscript	1,2
13)	Save/set point	1,2
14)	Large (>256 characters) fonts	1,2
15)	Variable size characters	

Notes

1) Not available in III Displayer
2) Not available in Atlas Displayer
3) Cannot be explicitly selected in Atlas Displayer

2. MK 2 SDF ORDERS

2.1 204 Enter Type Mode (Nonjustified)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	0	1	0	0	M	6,9	S	B	8,9	H	0	0	0
									P	8	F	E	6	S			

The first character will be displayed with its lower left corner at the current point. Subsequent characters will be at positions determined by the rotation, spacing, and line feed values. The current position may be changed by this command.

bit 9

1 means output characters with proportional spacing.
0 means output characters with mono spacing.

bit 10

1 means characters in 8 bit format
0 means character in 6 or 9 bit formats (see bit 13)

bit 11

1 means output text at high speed (if high-speed page print option is available).
0 means output text at regular speed.

bit 12

1 means update current position to end of last character printed.
0 means current position remains unchanged.

bit 13

1 means characters in 6 bit format
0 means characters in 8 or 9 bit format (see bit 10)

bit 14

1 means software simulator used even if height with hardware range so that height used is precise as opposed to intervals 0-63.
0 means will use hardware if possible.

Text Formats

6 bit format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Char #1						Char #2						Char #3					

8 bit format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	Char #1								Char #2							

9 bit or Half-word Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	CHAR. #1									CHAR. #2							
Field 1									Field 2								

If the high-order bit of one of these 9-bit fields is on, the remaining eight bits are a printing character; if the bit is off, the remaining eight bits are a control character. Text will be terminated by an end of message character (203_8).

For proportionally spaced text the number entered with the Set Character Spacing (code 211) command is the intercharacter spacing (the distance from the right of one character to the left of the next).

2.2 212 Set Text Line Spacing

One word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	0	1	0	1	0	Line Spacing								

where 0 ≤ Line Spacing ≤ 510_{10}

OR

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1

0	1	2	3														17
0	0	0	1	Large Line Spacing													

where 0 ≤ Large Line Spacing ≤ 16383

The data is a centre line spacing number (from the bottom of one line to the bottom of the next line) to be used in subsequent 204 commands.

2.3 211 Set Character Spacing

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	0	1	0	0	1	Character Spacing	

where 0 ≤ Character Spacing ≤ 510_{10}

OR

Two word format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	1	Large Character Spacing													

where 0 ≤ Large Character Spacing ≤ 16383_{10}

The data is a spacing number (from the left of one character to the left of the next for monospaced text: from the right of one character to the left of the next for proportionally spaced text) to be used in subsequent 206 commands.

2.4 210 Set Character Rotation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	ROTATION		

The data is a rotation number to be used in subsequent text display commands. A rotation value of zero corresponds to upright characters. Successive values are rotated 45° counterclockwise.

2.5 224 Select Character Height

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	1	0	1	0	0	Character Height	

where 0 ≤ Character Height ≤ 510_{10}

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1

0	1	2	3	4	17
0	0	0	1	Large Character Height	

where 0 ≤ Large Character Height ≤ 16383_{10}

Note that a character height of zero is not allowed.

2.6 232 Select or Modify Font

Overall structure of this class of orders is

2	3	2	F	N

where F = 0.....Select Font
F = 1.....Set/unset escape-action
F = 2.....Modify Dispatch Base
F = 3.....Reset Dispatch Base
F = 4.....Set Escape-action parameters

2.6.1 232000 Font Selection (III SDF)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0

0	1	2	3	4	5	6,	7	8	9	10	11	12	13	14	15	16	17
0	1	TANGENT OF ITALIC ANGLE (Binary point to the left of bit 2)															

This word is optional- the font will be displayed without obliqueness if this word is omitted or if the selected font will not permit italicisation.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	CHARACTER ASPECT RATIO x 10000															

This word is optional. It is used to permit expanding or condensing characters. Values less than 10000 (decimal) result in a condensed font; values more than 10000 result in an expanded font. If equal to 10000, or omitted, or a non-graphic arts font specified, the font will be used as designed.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	FONT NUMBER															

2.6.2 2320nn Nonstandard Select Font

0 8	9 10 11	12 17
010 011 010	000	N

N is the Font Number to be select N>0.

This is a non standard order. It should not be used. It has been included only for compatability with an Atlas modification to the III Displayer.

2.6.3 232100 Set-unset excape-actions

The following actions may be triggered by escape characters. They are:

01	End-of-text
02	Newline
03	Line feed
04	Carriage return
05	Space
06	Backspace
07	Null Character
10	Bad character
11	Permanent modification to Dispatch Base, implied index = 1
12.......	" " " " ", indexed
13	Reset Dispatch Base
14	Toggle modification to Dispatch Base, implied index = 1
15	" " " " ", implied index = 2
16	" " " " ", indexed
17	Escape character, next byte is action code
20-27	Save/Set Points (tabs etc)
30	Bold on/off
31	Underline on/off
32	Select font
33	Superscript, up a level
34	Subscript, down a level
35	Restore previous super/subscript level and previous (X,Y)
36.......	Restore previous super/subscript level but only previous Y
37	Super/subscript local X fraction modification
40	" " " Y " "
41	" " " HT " "
42	" " " SP " "
43	" " " level " "
44	Overline on/off
45-67	Not yet allocated
70-77	Reserved for system use (77 = end of buffer)

Set/unset orders have the following general format:

<table>
<tr><td>0   2</td><td>3   8</td><td>9</td><td>10   11</td><td>12</td><td>17</td></tr>
<tr><td>2</td><td colspan="2">3   2</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>6 bit action code</td><td>0</td><td colspan="3">8 bit character code</td></tr>
</table>

Unset Character code from being escape-action

2	3	2	1	0	0
0	0	0	char code cc		

The Dispatch Table entry for character code cc will be modified to return cc to be a normal plotting character. An unset must only be issued for a character code which actually has a Character Stroke Definition associated with it.

Action 01. End-of-text

2	3	2	1	0	0
0	0	1	cc		

The character code cc causes the plotting of text to cease, ie cc must be the last byte in the string. The rest of the word in which the end-of-text cc resides will be ignored.

Action 02. Newline

2	3	2	1	0	0
0	0	2	cc		

The occurrence of cc in a text string causes a newline to be started, determined by the 212 order and the initial starting position of the text.

Action 03. Line Feed

2 3 2	1	0 0
0	0 3	cc

The occurrence of cc in a text string causes the beam to be moved down onto the next line, determined by the 212 order.

Action 04. Carriage Return

2 3 2	1	0 0
0	0 4	cc

The occurrence of cc in a text string causes the beam to be returned to the start of the current line.

Action 05. Space

2 3 2	1	0 0
0	0 5	cc

The occurrence of cc in a text string causes a space to the plotted. The width of the space is set by the 211 order.

Action 06. Backspace

2 3 2	1	0 0
0	0 6	cc

The occurrence of cc in a text string will cause the beam to be backspaced to the beginning of the last character plotted. Any number of backspaces may be successfully performed if the text is monospaced. Under proportional spacing multiple backspaces will not give the intuitive result as only the width of the last character plotted is remembered. Should multiple backspaces under proportional spacing be necessary then this feature could be incorporated at an added cost in Driver code.

Action 07. Null Character

2 3 2	1	0 0
0	0 7	cc

The occurrence of cc in a text string is completely ignored. Driver does have a mechanism to count the number of null characters plotted however.

Note that the current III Displayer has a bug when using null characters with proportional spacing - it proportionally spaces null character so plotting a null character will actually advance the beam by an amount set by the 211 order.

Action 10. Bad Character

2 3 2	1	0 0
0	1 0	cc

The occurrence of cc in a text string will cause a small rectangular box to be plotted which is proportional to the current character height. This mechanism allows bad characters to be spotted in the output. Driver also has a mechanism to count the number of bad characters plotted.

Actions 11, 12. Permanent modification to the Dispatch Base

2 3 2	1	0 0
0	1 1	cc

The occurrence of cc in a text string will cause a permanent modification parameter to be added to the current value of the Dispatch Base, thus causing all successive bytes to access a new section of the Dispatch Table. This feature should be used with great care. The Permanent Modification parameter are set up by the 232406 order. Action 11 uses the 1st entry in the Permanent Modification parameter table, ie it has an implied index of 1. Action 12 takes the next byte as an index value.

Action 13. Escape Character

2 3 2	1	0 0
0	1 3	cc

The occurrence of cc in a text string causes the Dispatch Base to be reset to its initial value.

Actions 14, 15, 16. Toggle Modification to the Dispatch Base

2 3 2	1	0 0
0	1 4	cc

The first and following odd occurrences of cc in a text string will cause a Toggle Modification parameter to be added to the current value of the Dispatch Base. Second and even occurrences cause the subtraction of the parameter. This mechanism is intended to facilitate case changing. The Toggle Modification Parameters are set up by the 232407 order. Action 14 uses the first entry in the Toggle Modification parameter table, ie it has an implied index of 1. Action 15 has an implied index of 2. Action 16 takes the next byte as an index value.

Action 17. Escape Character

2 3 2	1	0 0
0	1 7	cc

The occurrence of cc in a text string indicates that the next byte is not a plottable character but is the integer number of an action code, ie if the user does not wish to dedicate a specific character code for each individual action he can nominate one character to act as an escape character which will be followed by which ever action code he requires.

Actions 20-27. Save/Set Points

2	3	2	1	0	0
0	2	0	cc		

Details of these action codes are given in section 3. Save/Set points are intended for use as horizontal and vertical tabs, save and set points, and special spacings.

Action 30. Bold

2	3	2	1	0	0
0	3	0	cc		

The occurrence of cc in a text string causes the emboldening of characters to be toggled. The initial setting is OFF. The parameters associated with emboldening are set by the 232404 order.

Action 31. Underline

2	3	2	1	0	0
0	3	1	cc		

The occurrence of cc in a text string causes the underlining of characters to be toggled. The initial setting is OFF. The parameters associated with underlining are set by the 232403 order.

Action 32. Select Font

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>3 2</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string causes the next byte to be interpreted as a font number. This font will then be selected.

Action 33,34. Superscript/Subscript

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>3 3</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string causes the current position and text parameters to be saved on a stack, and a new set of text parameters to be set according to the values set by the 232405 order.

Action 33 (34) sets up the next set of Supercript (subscript) parameters.

Action 35. Restore previous super/subscript level and previous CX,CY

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>3 5</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string causes the current position and text parameters to be 'unstacked' and the previous position and parameters to be restored.

Action 36. Restore previous super/subscript level and previous CY

2	3	2	1	0	0
0	3	6	cc		

The occurrence of cc in a text string leaves the current X coordinate unchanged but restores the previous Y coordinate as well as restoring the previous text parameters.

Actions 37-43. Super/Subscript local fraction modification

2	3	2	1	0	0
0	3	7	cc		

The occurrence of cc in a text string will cause one of the super/subscripting fractions in the currently active set to be modified. This modification will be lost when the corresponding action 35 causes unstacking of this parameterset, hence the phrase 'local modification' as the values set by the 232405 order remain unchanged. The layouts of the actions in the text strings are:

<cc→37>	<X top> <X bottom>	X fraction
<cc→40>	<Y top> <Y bottom>	
<cc→40>	<Y top> <Y bottom>	Y fraction
<cc→41>	<HT top> <HT bottom>	height fraction
<cc→42>	<SP top> <SP bottom>	spacing fraction
<cc→43>	<level>	halt level

Action 44. Overline on/off

2	3	2	1	0	0
0	4	4	cc		

The occurrence of cc in a text string causes the overlining of characters to be toggled. The initial setting is OFF. The parameters associated with overlining are set by the 232410 order.

2.6.4 232200 Immediate permanent modification to Dispatch Base

2 3 2	2	0 0
2's comp. 18 bit integer		

The parameter will be added to the current value of the Dispatch Base immediately.

2.6.5 232300 Reset Dispatch Base

2 3 2	3	0 0

Dispatch Base is reset to its initial value ie the address of first entry in the Dispatch Table.

2.6.6 232400 Setup Escape-action parameters

This class of orders sets up the parameters for those escape-action mechanisms which allow text driven modifications to the character plotting. Their overall structure is:

2 3 2	4	N

where:

N = 01 ...... Set Point Order Entry
N = 02 ...... Save/Set Point Action Setup
N = 03 ...... Underline
N = 04 ...... Bold
N = 05 ...... Super/subscripting
N = 06 ...... Permanent Dispatch Modification
N = 07 ...... Toggle Dispatch Modification
N = 10 ...... Overline

232401 Set Point Order Entry

2 3 2	4	0 1
index		
Set Point X abs/rel		
Set Point Y abs/rel		

The two Set Point orders will be entered into Position Table (index). The orders may be absolute or relative but must be single word orders only.

232402 Save/Set Point Action Setup

2 3 2	4	0 5
mode	action #	
start posn	end posn	
pointer initial value		
pointer increment		

action number .... one of 20-27

mode	001	....	save point	indexed
	002	....	" "	auto-increment
	003	....	" "	search X
	004	....	" "	search Y
	005	....	set point	indexed
	006	....	" "	auto-increment
	007	....	" "	search > X
	010	....	" "	search > Y
	011	....	" "	search < X
	012	....	" "	search < Y

The specified action number will cause the Save/Set Point action determined by the associated mode. References to the Position Table will be between the Start and End entries. Those modes using an implicit pointer will have their initial value and increment set by this order. For those modes which have explicit indexes these two words will be ignored.

232403 Underline

2 3 2	4	0 3
rel X offset		
rel Y offset		
S 0 0 0 0 0		

S = 4 ...... Full underlining (including inter character spacing)
S = 0 ...... Intercharacter spacing not underlined

The starting position of the underline is the starting position of the character offset by (X,Y) as specified above. The line will always extend the full width of the character, but may optionally underline the intercharacter space if proportional spacing is active. Underlining is controlled by action 31.

232404 Bold

2 3 2	4	0 4
repeat factor RF		
rel X offset		
rel Y offset		

Each character will be drawn RF times, being incrementally offset by (X,Y) each time. Emboldening is controlled by action 32.

232405 Super/Subscript

2 3 2	4	0	5
halt depth	S	0	0
XM			
XD			
YM			
YD			
HTM			
HTD			
SPM			
SPD			

S = 0 .... Superscript parameters
S = 1 .... Subscript parameters

Halt depth is the super/subscripting level number at which the scaling parameters are no longer applied ie all fractions are 1/1. This prevents characters being reduced to invisibility after many levels of super/subscripting.

X and Y fractions determine the starting position of the next level of super/subscripting. HT and SP fractions determine the character height and spacing. See section 3 for more details.

232406 Permanent Dispatch Base Modification

2 3 2	4	0 6
0 0 0	0	index
2's complement integer		

The integer is placed in the Permanent Modification parameter table at the position denoted by 'index'. This index value will correspond to an index value associated with actions 11 and 12.

232407 Toggle Dispatch Base Modification

2 3 2	4	0 7
0 0 0	0	index
2's complement integer		

The integer is placed in the Toggle Modification parameter table at the position denoted by 'index'. This index value will correspond to an index value associated with actions 14, 15 and 16.

232410 Overline

2 3 2	4	1 0
rel x offset		
rel y offset		
S 0 0 0 0 0		

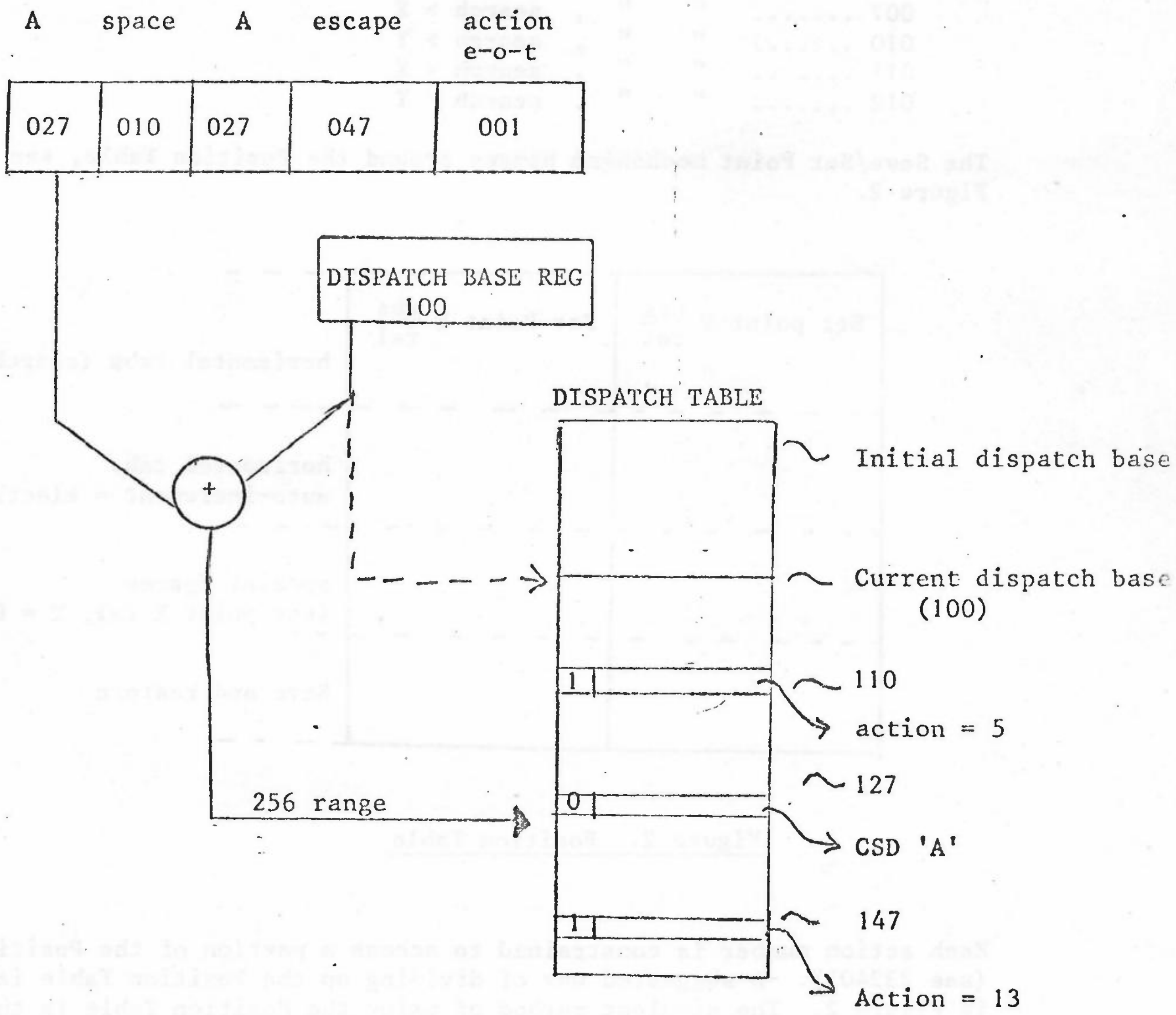
S = 4 ... Full overlining (including intercharacter spacing)
S = 0 ... Intercharacter spacing not overlined.

The starting position of overlining is the starting position of the next character (ie the current position) offset by (X+current character height,Y). The line will always extend the full width of the character, but may optionally overline the intercharacter gap if proportional spacing is active. Overlining is controlled by action 44. Overlining is the complement of underlining.

3. BRIEF EXPLANATORY NOTES

3.1 The character plotting mechanism

All words immediately following a 204 order are considered to be plottable characters. A byte's character code is added to the Dispatch Base to find its entry in the Dispatch Table. This entry points to the Character Stroke Definition (CSD). If the character is to be plotted, then the hardware interpretes the CSD. If the character has been nominated as an escape action code then Driver causes the required action to occur and then continues plotting.



Result: A A

Figure 1

3.2 Save/Set Point Mechanism

There are eight Save/Set Point action codes (20-27), each of which may operate in any one of ten modes defined by the 232402 order. The modes are:

001	Save Point,	indexed
002	" " ,	auto-increment
003	" " ,	search X
004	" " ,	search Y
005	Set Point,	indexed
006	" " ,	auto-increment
007	" " ,	search > X
010	" " ,	search > Y
011	" " ,	search < X
012	" " ,	search < Y

The Save/Set Point mechanism hinges around the Position Table, see Figure 2.

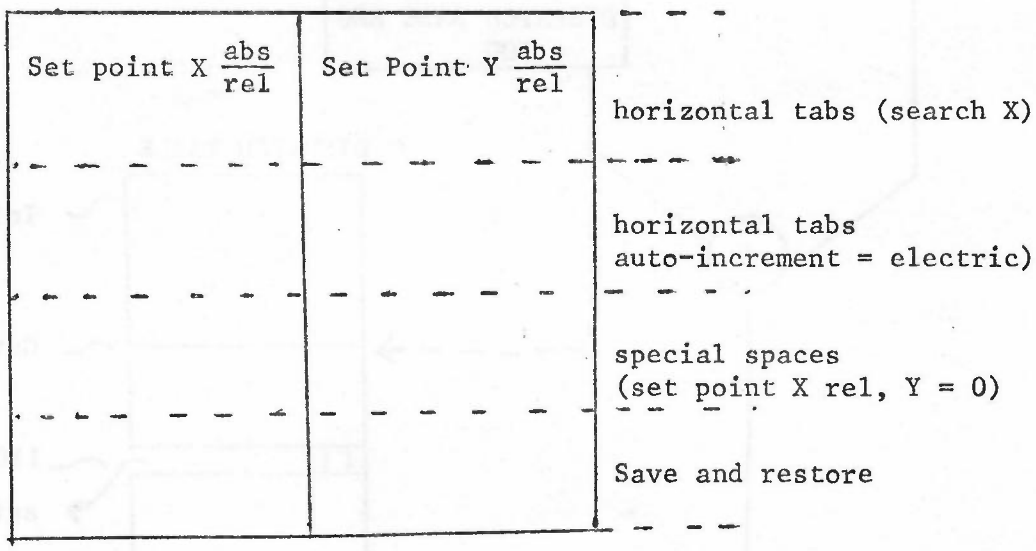

Set point X $\frac{\text{abs}}{\text{rel}}$
Set Point Y $\frac{\text{abs}}{\text{rel}}$
horizontal tabs (search X)
horizontal tabs
auto-increment = electric)
special spaces
(set point X rel, Y = 0)
Save and restore

Figure 2. Position Table

Each action number is constrained to access a portion of the Position Table (see 232402). A suggested way of dividing up the Position Table is shown in Figure 2. The simplest method of using the Position Table is the indexed mode in which an action (mode = index) code in the text string is followed by an integer which is used as an index to select a particular entry in the Position Table. Suppose a Save Point, index = 3 in encountered: The absolute X and Y coordinates of the current position will be entered in slot 3 of the table. A Set Point, index = 3, will cause the beam to be moved to the absolute coordinates stored in slot 3. The auto-increment mode has an implicit index which is initialised by the 232402 order. Each time an auto-increment Save/Set Point is encountered the implicit index is altered, see Figure 3. If the Save Point increment is +1 and the Set Point increment is -1 then a stacking mechanism is achieved.

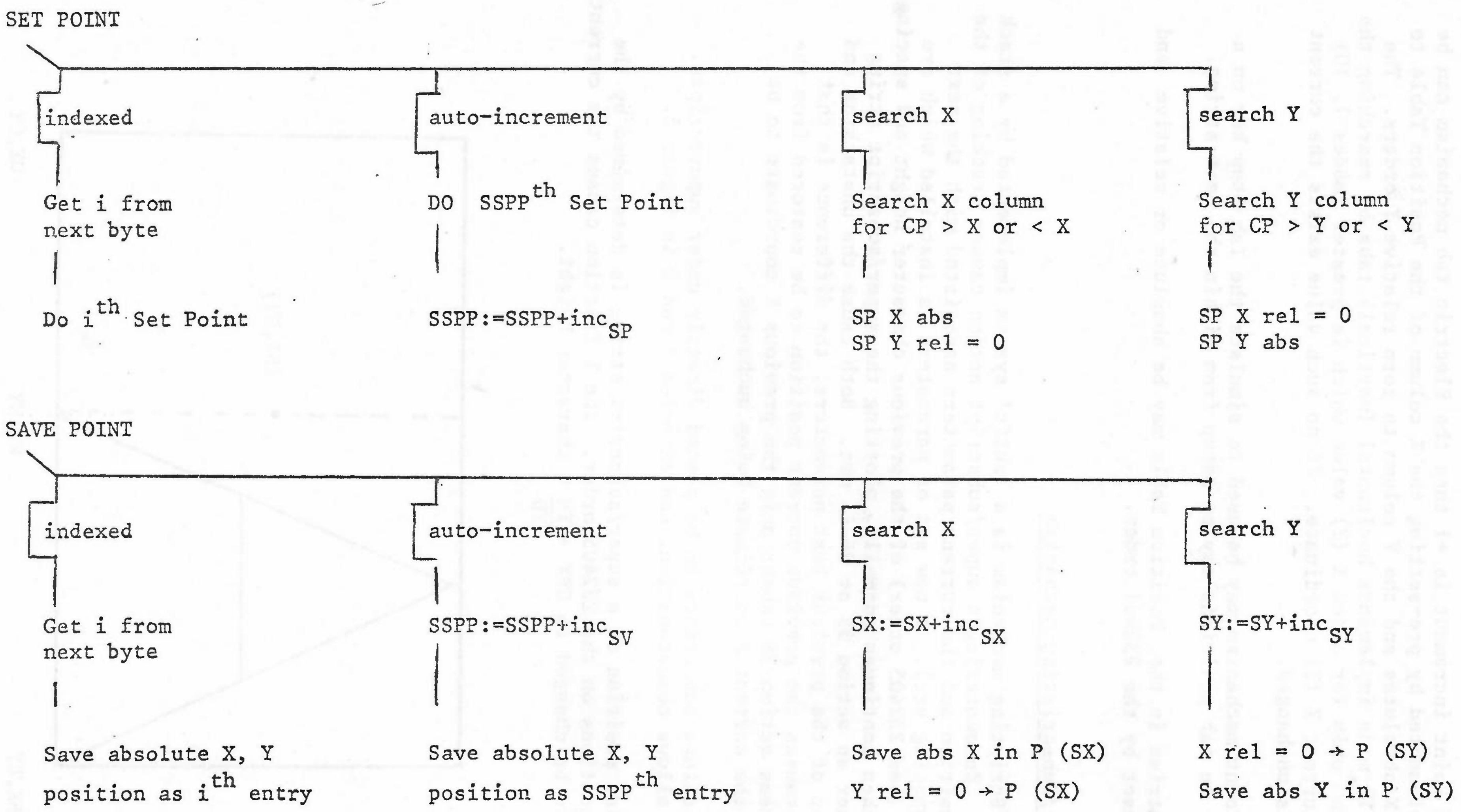

SET POINT
indexed
auto-increment
search X
search Y
Get i from next byte
DO SSPPth Set Point
Search X column for CP > X or < X
Search Y column for CP > Y or < Y
Do ith Set Point
SSPP:=SSPP+inc$_{SP}$
SP X abs
SP Y rel = 0
SP X rel = 0
SP Y abs
SAVE POINT
indexed
auto-increment
search X
search Y
Get i from next byte
SSPP:=SSPP+inc$_{SV}$
SX:=SX+inc$_{SX}$
SY:=SY+inc$_{SY}$
Save absolute X, Y position as ith entry
Save absolute X, Y position as SSPPth entry
Save abs X in P (SX)
Y rel = 0 → P (SX)
X rel = 0 → P (SY)
Save abs Y in P (SY)

Figure 3

If the Set Point increment is +1 then the Electric tab mechanism can be easily implemented by pre-setting the X column of the Position Table to Set Point X absolutes and the Y column to zero relative Y orders. The search X (Y) mode implements horizontal (vertical) tabs by searching the X (Y) column only for a new X (Y) value which is greater (modes 7, 10) than the current X (Y) coordinate. If no such value exists the current position is unchanged.

The Save Point mechanism may be used to simulate the Tab Stop key on a typewriter ie tab positions may be setup from within the text string.

All the entries in the Position Table may be absolute or relative and may be preset by the 232401 order.

3.3 Super/Subscripting mechanism

The Super/Scripting mechanism is a 'shift' system implemented by a stack in DRIVER. Encountering a super/subscript action causes stacking of the current position and the current parameters associated with the text (height, spacing etc). A new set of parameters is installed which are fractions (see 232405 order) of the previous character height and spacing. Plotting then continues normally, plotting the super/subscript string until either an action 35 or 36 is met. Both cause the unstacking and restoration of the previous text parameters; the difference is that action 35 causes the previous current position to be restored from the stack whereas action 36 causes only the previous Y coordinate to be restored, the current X coordinate being unchanged.

Action 35 allows subscripts to be placed directly under superscripts. Action 36 allows concatenation, see examples 1 and 2 in figure 5.

The starting position of a super/subscript string is determined by the X and Y fractions on the 232405 order. The Y fraction causes the current position to be changed to CPY + $\frac{YM}{YD}$ * character height.

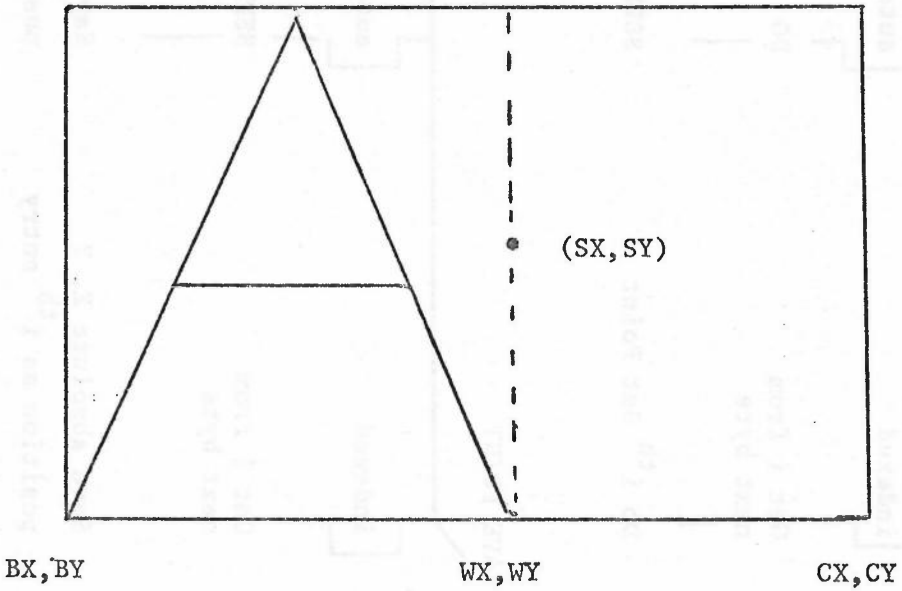


Figure 4

The X coordinate will be calculated as:

$$X' := CX - (\text{backspace } X) + \frac{XM}{XD} * \text{character width.}$$

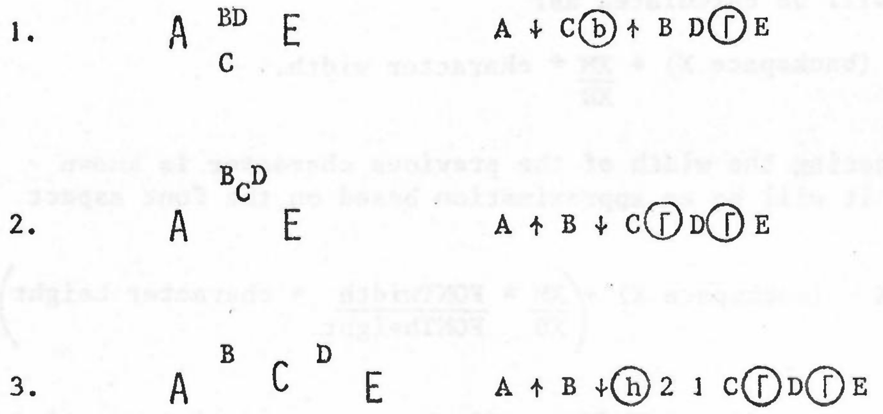
In proportional spacing the width of the previous character is known exactly. In mono it will be an approximation based on the font aspect ratio:

$$\text{ie} \quad X' := CX - (\text{backspace } X) + \left(\frac{XM}{XD} * \frac{\text{FONTwidth}}{\text{FONTheight}} * \text{character height} \right)$$

For example, if XM/XD = 1/1 and YM/YD = 1/2 then a superscript starting position (SX, SY) would be as shown in Figure 4. (CX, CY) - (WX, WY) represents the intercharacter spacing value.

Local modifications may be made to the fractions controlling Super/Subscripting by using actions 37-43. These modifications last only until an action 35 or 36 unstacks the particular entry, see example 3 in Figure 5.

Repeated application of the fractions could cause the character height to be reduced below a visible threshold. To prevent this a limiting depth may be set in the 232405 order. Super/subscripting fractions beyond this depth limit will be set to 1/1 so no further reductions can occur.



Key: ↑ .... action 33 - superscript

↓ .... action 34 - subscript

ⓑ .... action 35 - restore CX, CY

$\textcircled{\Gamma}$ .... action 36 - restore CY only

ⓗ .... action 41 - local mod to height

Figure 5

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 23

FR80 DRIVER - MARK 2 Text Processing Orders Third Iteration

issued by
R W Witty
D C Sutcliffe

16 November 1977

DISTRIBUTION:

F R A Hopgood
M F Chiu
D A Duce
A H Francis
K M Crennell
C Osland

A W Burraston
R Brandwood
R W Witty
P M Nelson
D C Sutcliffe
C Cooper

1. INTRODUCTION

This note is the third version of a proposed set of text processing orders for the Mark 2 version of DRIVER. It replaces FR80 Discussion Paper 22, issued 25 May 1977. One major change is made in this paper, namely the introduction of the Edge of Page concept. This allows text formatting features like tabs, newlines and special spaces to be specified as being relative to a notional Edge of Page. The Edge of Page concept enables text output to be drawn at all 8 rotation angles with consistent results. Discussion Paper 22 failed to consider the question of outputting text at 45° angles. A description of the Edge of Page idea is given in Section 3 of this paper.

Remember that this is a proposal, not a specification for Mark 2 DRIVER; your comments and suggestions are most welcome.

1.1 Summary of Text Processing Facilities

1)	Mono or proportional spacing	
2)	6, 8, 9 bit formats	1,2
3)	Fast or slow speed plotting	1,2
4)	Beam to beginning/end of string	1,2
5)	Software simulator	1,3
6)	Large characters	1
7)	Select font	1
8)	Escape action mechanism	1,2
9)	Bold characters	1,2
10)	Underlining	1,2
11)	Tabs	1,2
12)	Superscript/subscript	1,2
13)	Save/set point	1,2
14)	Large (>256 characters) fonts	1,2
15)	Variable size characters	

Notes

1) Not available in III Displayer
2) Not available in Atlas Displayer
3) Cannot be explicitly selected in Atlas Displayer

2. MK 2 SDF ORDERS

2.1 204 Enter Type Mode (Nonjustified)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	0	1	0	0	M	6,9	S	B	8,9	H	0	0	0
									P	8	F	E	6	S			

The first character will be displayed with its lower left corner at the current point. Subsequent characters will be at positions determined by the rotation, spacing, and line feed values. The current position may be changed by this command.

bit 9

1 means output characters with proportional spacing.
0 means output characters with mono spacing.

bit 10

1 means characters in 8 bit format
0 means character in 6 or 9 bit formats (see bit 13)

bit 11

1 means output text at high speed (if high-speed page print option is available).
0 means output text at regular speed.

bit 12

1 means update current position to end of last character printed.
0 means current position remains unchanged.

bit 13

1 means characters in 6 bit format
0 means characters in 8 or 9 bit format (see bit 10)

bit 14

1 means software simulator used even if height with hardware range so that height used is precise as opposed to intervals 0-63.
0 means will use hardware if possible.

Text Formats

6 bit format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Char #1						Char #2						Char #3					

8 bit format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	Char #1								Char #2							

9 bit or Half-word Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	CHAR. #1									CHAR. #2							

Field 1 Field 2

If the high-order bit of one of these 9-bit fields is on, the remaining eight bits are a printing character; if the bit is off, the remaining eight bits are a control character. Text will be terminated by an end of message character (203_8).

For proportionally spaced text the number entered with the Set Character Spacing (code 211) command is the intercharacter spacing (the distance from the right of one character to the left of the next).

2.2 212 Set Text Line Spacing

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	0	1	0	1	0	Line Spacing	

where 0 ≤ Line Spacing ≤ 510_{10}

OR

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1

0	1	2	3		17
0	0	0	1	Large Line Spacing	

where 0 ≤ Large Line Spacing ≤ 16383

The data is a centre line spacing number (from the bottom of one line to the bottom of the next line) to be used in subsequent 204 commands.

2.3 211 Set Character Spacing

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	0	1	0	0	1	Character Spacing	

where $0 \le$ Character Spacing $\le 510_{10}$

OR

Two word format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	1	Large Character Spacing													

where $0 \le$ Large Character Spacing $\le 16383_{10}$

The data is a spacing number (from the left of one character to the left of the next for monospaced text: from the right of one character to the left of the next for proportionally spaced text) to be used in subsequent 206 commands.

2.4 210 Set Character Rotation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	ROTATION		

The data is a rotation number to be used in subsequent text display commands. A rotation value of zero corresponds to upright characters. Successive values are rotated 45^{o} counterclockwise.

2.5 224 Select Character Height

One word format

0	1	2	3	4	5	6	7	8	9	17
0	1	0	0	1	0	1	0	0	Character Height	

where $0 \le$ Character Height $\le 510_{10}$

Two word format

0	1	2	3	4	5	6	7	8	9								17
0	1	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1

0	1	2	3	4	17
0	0	0	1	Large Character Height	

where 0 ≤ Large Character Height ≤ 16383_{10}

Note that a character height of zero is not allowed.

2.6 232 Select or Modify Font

Overall structure of this class of orders is

2	3	2	F	N

where F = 0.....Select Font
F = 1.....Set/unset escape-action
F = 2.....Modify Dispatch Base
F = 3.....Reset Dispatch Base
F = 4.....Set Escape-action parameters

2.6.1 232000 Font Selection (III SDF)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	TANGENT OF ITALIC ANGLE (Binary point to the left of bit 2)															

This word is optional- the font will be displayed without obliqueness if this word is omitted or if the selected font will not permit italicisation.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	CHARACTER ASPECT RATIO x 10000															

This word is optional. It is used to permit expanding or condensing characters. Values less than 10000 (decimal) result in a condensed font; values more than 10000 result in an expanded font. If equal to 10000, or omitted, or a non-graphic arts font specified, the font will be used as designed.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	FONT NUMBER															

2.6.2 2320nn Nonstandard Select Font

0 8	9 10 11	12 17
010 011 010	000	N

N is the Font Number to be select N>0.

This is a non standard order. It should not be used. It has been included only for compatability with an Atlas modification to the III Displayer.

2.6.3 232100 Set-unset excape-actions

The following actions may be triggered by escape characters. They are:

01	End-of-text
02	Newline
03	Line feed
04	Carriage return
05.......	Space
06	Backspace
07	Null Character
10	Bad character
11	Permanent modification to Dispatch Base, implied index = 0
12	" " " " " " " = 1
13	" " " " " indexed
14	Toggle modification to Dispatch Base, implied index = 0
15	" " " " " " = 1
16	" " " " " indexed
17	Escape character, next byte is action code
20-27	Save/Set Points (tabs etc)
30	Bold on/off
31,45.....	Underline on/off
32	Select font
33.......	Superscript, up a level
34	Subscript, down a level
35	Restore previous super/subscript level and previous (X,Y)
36	Restore previous super/subscript level but only previous Y
37	Super/subscript local X fraction modification
40	" " " Y " "
41	" " " HT " "
42	" " " SP " "
43	" " " level
44,46.....	Overline on/off
47	Reset dispatch base
50-67.....	Not yet allocated
70-77.....	Reserved for system use (77 = end of buffer)

Set/unset orders have the following general format:

0 2	3 8	9	10 11	12		17
2	3 2		1	0		0
0	6 bit action code	0	8 bit character code			

Unset Character code from being escape-action

2	3 2		1	0 0
0	0 0		char code cc	

The Dispatch Table entry for character code cc will be modified to return cc to be a normal plotting character. An unset must only be issued for a character code which actually has a Character Stroke Definition associated with it.

Action 01. End-of-text

2	3 2	1	0 0
0	0 1	cc	

The character code cc causes the plotting of text to cease, ie cc must be the last byte in the string. The rest of the word in which the end-of-text cc resides will be ignored.

Action 02. Newline

2	3 2	1	0 0
0	0 2	cc	

The occurrence of cc in a text string causes a newline to be started, determined by the 212 order and the initial starting position of the text.

Action 03. Line Feed

<table>
<tr><td>2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0 0 3</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string causes the beam to be moved down onto the next line, determined by the 212 order.

Action 04. Carriage Return

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>0 4</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string causes the beam to be returned to the start of the current line.

Action 05. Space

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>0 5</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string causes a space to the plotted. The width of the space is set by the 211 order.

Action 06. Backspace

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>0 6</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string will cause the beam to be backspaced to the beginning of the last character plotted. Any number of backspaces may be successfully performed if the text is monospaced. Under proportional spacing multiple backspaces will not give the intuitive result as only the width of the last character plotted is remembered. Should multiple backspaces under proportional spacing be necessary then this feature could be incorporated at an added cost in Driver code.

Action 07. Null Character

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>0 7</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string is completely ignored. Driver does have a mechanism to count the number of null characters plotted however.

Note that the current III Displayer has a bug when using null characters with proportional spacing - it proportionally spaces null character so plotting a null character will actually advance the beam by an amount set by the 211 order.

Action 10. Bad Character

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>1 0</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string will cause a small rectangular box to be plotted which is proportional to the current character height. This mechanism allows bad characters to be spotted in the output. Driver also has a mechanism to count the number of bad characters plotted.

Action 11, 12, 13 Permanent modification to the Dispatch Base

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>1 1</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string will cause a permanent modification parameter to be added to the current value of the Dispatch Base, thus causing all successive bytes to access a new section of the Dispatch Table. This feature should be used with great care. The Permanent Modification parameter are set up by the 232406 order. Action 11 (12) uses the 1st (2nd) entry in the Permanent Modification parameter table, ie it has an implied inded of 0(1). Action 13 takes the next byte as an index value.

Action 32. Select Font

2	3	2	1	0	0
0	3	2	cc		

The occurrence of cc in a text string causes the next byte to be interpreted as a font number. This font will then be selected.

Action 33,34. Superscript/Subscript

2	3	2	1	0	0
0	3	3	cc		

The occurrence of cc in a text string causes the current position and text parameters to be saved on a stack, and a new set of text parameters to be set according to the values set by the 232405 order.

Action 33 (34) sets up the next set of Supercript (subscript) parameters.

Action 35. Restore previous super/subscript level and previous CX,CY

2	3	2	1	0	0
0	3	5	cc		

The occurrence of cc in a text string causes the current position and text parameters to be 'unstacked' and the previous position and parameters to be restored.

Action 36. Restore previous super/subscript level and previous CY

2 3 2	1	0 0
0	3 6	cc

The occurrence of cc in a text string leaves the current X coordinate unchanged but restores the previous Y coordinate as well as restoring the previous text parameters.

Actions 37-43. Super/Subscript local fraction modification

2 3 2	1	0 0
0	3 7	cc

The occurrence of cc in a text string will cause one of the super/subscripting fractions in the currently active set to be modified. This modification will be lost when the corresponding action 35 causes unstacking of this parameterset, hence the phrase 'local modification' as the values set by the 232405 order remain unchanged. The layouts of the actions in the text strings are:

<cc→37>	<X top> <X bottom>	X fraction
<cc→40>	<Y top> <Y bottom>	
<cc→40>	<Y top> <Y bottom>	Y fraction
<cc→41>	<HT top> <HT bottom>	height fraction
<cc→42>	<SP top> <SP bottom>	spacing fraction
<cc→43>	<level>	halt level

Actions 44,46 Overline on/off

2 3 2	1	0 0
0	4 4	cc

The occurrence of cc in a text string causes the overlining of characters to be toggled. The initial setting is OFF. The parameters associated with overlining are set by the 232410 order. Action 44 controls full overlining. Action 46 controls broken overlining.

Action 47. Reset dispatch base

<table>
<tr><td colspan="2">2 3 2</td><td>1</td><td>0 0</td></tr>
<tr><td>0</td><td>4 7</td><td colspan="2">cc</td></tr>
</table>

The occurrence of cc in a text string causes the Dispatch Base to be reset to its initial value.

2.6.6 232400 Setup Escape-action parameters

This class of order sets up the parameters for those escape-action mechanisms which allow text driven modifications to the character plotting. Their overall structure is:

2 3 2	4	N

where:

N = 01 ...... Set Point Order Entry
N = 02 ...... Save/Set Point Action Setup
N = 03 ...... Underline
N = 04 ...... Bold
N = 05 ...... Super/subscripting
N = 06 ...... Permanent Dispatch Modification
N = 07 ...... Toggle Dispatch Modification
N = 10 ...... Overline
N = 11 ...... Set/unset Edge of Page & enter/leave Mk 2 Text Mode

232401 Set Point Order Entry

2 3	2	4 0 1
index		
Set Point X rel		
Set Point Y rel		

Two Set Point Relative orders will be entered into by Position Table at slot <index>. Only relative orders may be used. Only the single word version may be used. The orders specify a position relative to the current Edge of Page axes, see section 3.

232402 Save/Set Point Action Setup

2 3 2	4	0 5
mode	action #	
start posn	end posn	
pointer initial value		
pointer increment		

action number ...... one of 20-27

mode	001	...... save point	indexed	
	002	...... " "	auto-increment	
	003	...... " "	search X	
	004	...... " "	search Y	
	005	...... set point	indexed	
	006	...... " "	auto-increment	
	007	...... " "	search > X	
	010	...... " "	search > Y	
	011	...... " "	search < X	
	012	...... " "	search < Y	

The specified action number will cause the Save/Set Point action determined by the associated mode. References to the Position Table will be between the Start and End entries. Those modes using an implicit pointer will have their initial value and increment set by this order. For those modes which have explicit indexes these two words will be ignored.

232403 Underline

2 3 2	4	0 3
Set Point rel Y offset		
S 0 0 0 0 0		
number of underlines		
gap between lines		

S = 4 ...... Full underlining (including inter character spacing)
S = 0 ...... Intercharacter spacing not underlined (broken underlining)

The starting position of the underline is the starting position of the character offset by Y as specified above. The line will always extend the full width of the character, but may optionally underline the intercharacter space if proportional spacing is active. Underlining is controlled by action 31, 45. The Y offset is a single word Set Point Relative order.

232404 Bold

2 3 2	4	0 4
repeat factor RF		
Set Point rel X offset		
Set Point rel Y offset		

Each character will be drawn RF times, being incrementally offset by (X,Y) each time. Emboldening is controlled by action 32. The (X,Y) offset is a pair of single word Set Point Relative Orders.

232405 Super/Subscript

2 3 2	4	0 5	
halt depth	S	0	0
XM			
XD			
YM			
YD			
HTM			
HTD			
SPM			
SPD			

X see discuss paper 20

S = 0 .... Superscript parameters
S = 1 .... Subscript parameters

Halt depth is the super/subscripting level number at which the scaling parameters are no longer applied ie all fractions are 1/1. This prevents characters being reduced to invisibility after many levels of super/subscripting.

X and Y fractions determine the starting position of the next level of super/subscripting. HT and SP fractions determine the character height and spacing. See section 3 for more details.

232406 Permanent Dispatch Base Modification

2 3 2	4	0 6
0 0 0	0	index
2's complement integer		

The integer is placed in the Permanent Modification parameter table at the position denoted by 'index'. This index value will correspond to an index value associated with actions 11 and 12.

232407 Toggle Dispatch Base Modification

2 3 2	4	0 7
0 0 0	0	index
2's complement integer		

The integer is placed in the Toggle Modification parameter table at the position denoted by 'index'. This index value will correspond to an index value associated with actions 14, 15 and 16.

232410 Overline

2 3 2	4	1 0
Set Point rel Y offset		
S 0 0 0 0 0		
number of overlines		
gap between lines		

S = 4 ... Full overlining (including intercharacter spacing)
S = 0 ... Intercharacter spacing not overlined (broken overlining).

The starting position of overlining is the starting position of the next character (ie the current position) offset by (Y + current character height + relative Y offset). The line will always extend the full width of the character, but may optionally overline the intercharacter gap if proportional spacing is active. Overlining is controlled by actions 44,46. Overlining is the complement of underlining. The Y offset is a single word Set Point Relative order.

232411 Set/unset Edge of Page & Mk 2 Text Mode

2	3	2	4	1	1
S	0	0	0	0	0

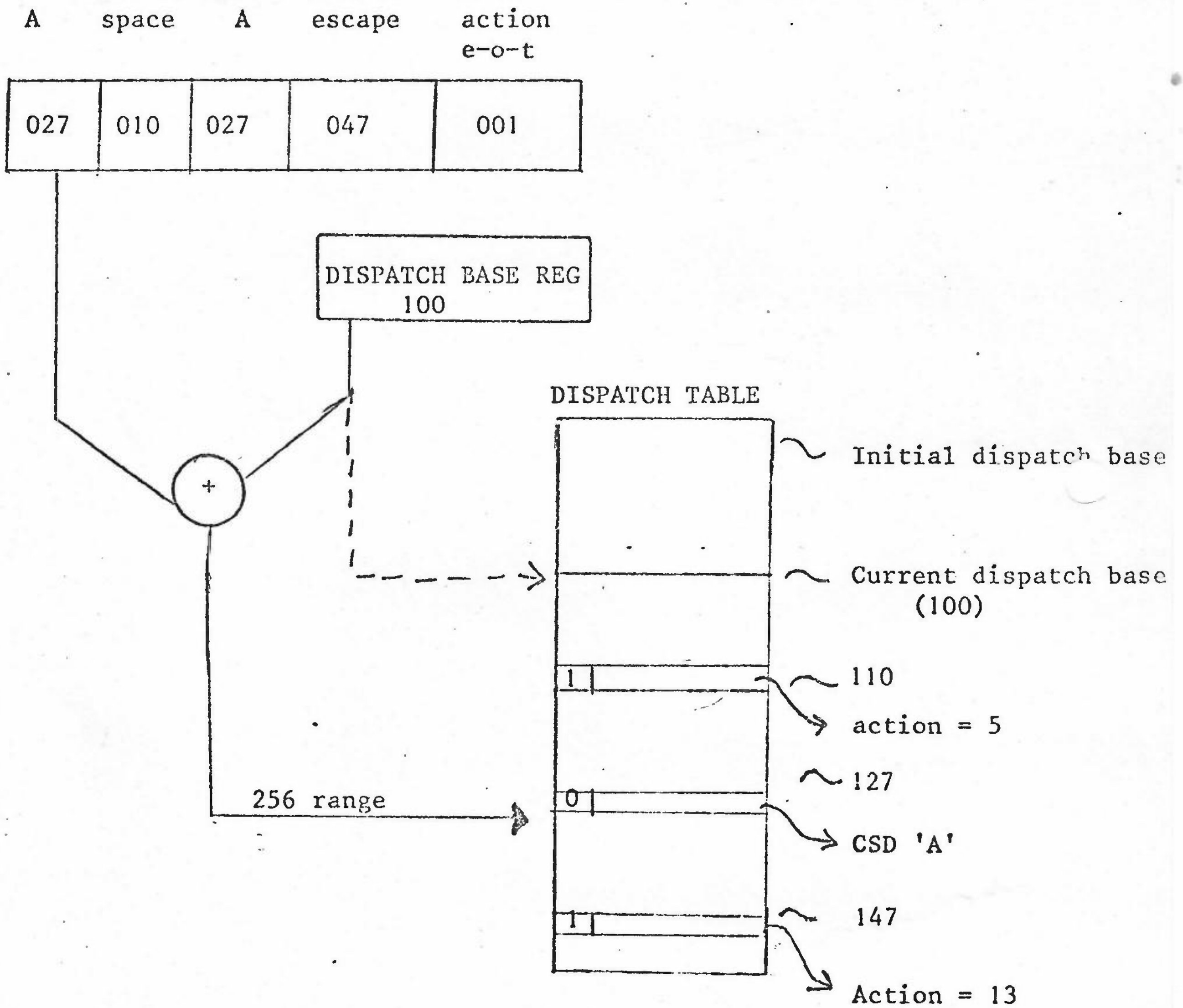
S = 0 ...... Set Edge of Page system at current point and enter Mk 2 text mode (see section 3).

S = 4 ...... Unset Mk 2 Text mode ie remove Edge of Page.

3. BRIEF EXPLANATORY NOTES

3.1 The character plotting mechanism

All words immediately following a 204 order are considered to be plottable characters. A byte's character code is added to the Dispatch Base to find its entry in the Dispatch Table. This entry points to the Character Stroke Definition (CSD). If the character is to be plotted, then the hardware interpretes the CSD. If the character has been nominated as an escape action code then Driver causes the required action to occur and then continues plotting.



Result: A A

Figure 1

3.2 The Edge of Page concept

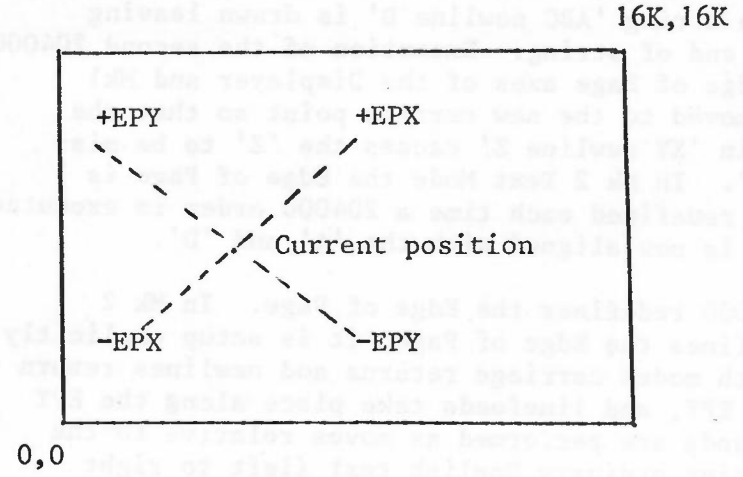
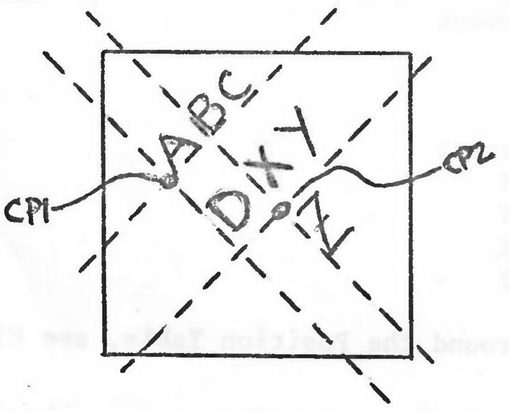


Figure 6. Edge of Page axes for text rotation = 1 (45° anticlockwise)

A new order, 232411, causes Mk 2 DRIVER to enter/leave Mk 2 Text Mode from/to Mk 1 Text Mode. Entering Mk 2 Text Mode sets up the Edge of Page axes at the current point. These axes define the logical Edge(s) of Page for text layout commands. The axes are rotated about the current point according to the current value of text rotation (210000 order). All text layout commands are relative to the Edge of Page axes. This allows a clean workable definition of text layout at all 8 rotations. The system outlined in Discussion Paper 22 was inadequate for 45° rotations.



210001

204040

A B

C newline

D eot

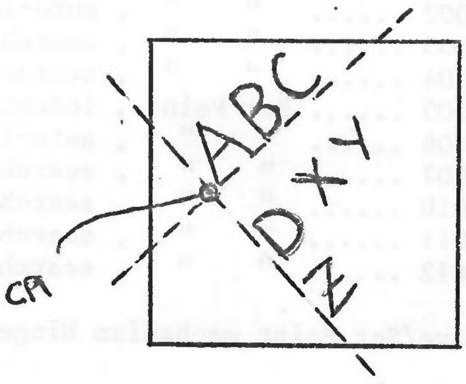
204000

X Y

newline Z

eot

Figure 7. Mk 1 Text Mode



210001

232411

000000

204040

A B

C newline

D eot

204000

X Y

newline Z

eot

232411

400000

Figure 8. Mk 2 Text Mode

Figures 7 and 8 are an example of the difference between Mk 1 and Mk 2 Text Modes. In figure 7 the string 'ABC newline D' is drawn leaving the current position at the end of string. Execution of the second 204000 order causes the implicit Edge of Page axes of the Displayer and Mk1 DRIVER to be redefined and moved to the new current point so that the second newline encountered in 'XY newline Z' causes the 'Z' to be misaligned with the 'A' and 'D'. In Mk 2 Text Mode the Edge of Page is explicitly setup and is not redefined each time a 204000 order is executed so that in figure 8 the 'Z' is now aligned with the 'A' and 'D'.

In Mk 1 Text Mode every 204000 redefines the Edge of Page. In Mk 2 Text Mode 204000 never redefines the Edge of Page, it is setup explicitly by the 232411 order. In both modes carriage returns and newlines return to the Edge of Page Y axis, EPY, and linefeeds take place along the EPY axis. All text layout commands are performed as moves relative to the Edge of Page axes. For setting ordinary English text (left to right working) spacing and tabs will involve positive (+EPX) relative moves and line feeds and vertical tabs will require negative (-EPY) relative moves. Conceptually the axes are the left hand margin and top line of the 'page' in this example.

3.3 Save/Set Point Mechanism

There are eight Save/Set Point action codes (20-27), each of which may operate in any one of ten modes defined by the 232402 order. The modes are:

```
001 ...... Save Point, indexed
002 ......  "     "  , auto-increment
003 ......  "     "  , search X
004 ......  "     "  , search Y
005 ...... Set Point , indexed
006 ......  "     "  , auto-increment
007 ......  "     "  , search > X
010 ......  "     "  , search > Y
011 ......  "     "  , search < X
012 ......  "     "  , search < Y
```

The Save/Set Point mechanism hinges around the Position Table, see Figure 2.

Set point X rel	Set Point Y rel	horizontal tabs (search X)
		horizontal tabs auto-increment = electric)
		special spaces (set point X rel, Y = 0)
		Save and restore

Figure 2. Position Table

Each action number is constrained to access a portion of the Position Table (see 232402). A suggested way of dividing up the Position Table is shown in Figure 2. The simplest method of using the Position Table is the indexed mode in which an action (mode = index) code in the text string is followed by an integer which is used as an index to select a particular entry in the Position Table. Suppose a Save Point, index = 3 in encountered: The X and Y coordinates of the current position will be entered into slot 3 of the table as relative moves from the edge of page axes. A Set Point, index = 3, will cause the beam to be moved to the edge of page origin plus the relative moves stored in slot 3. The auto-increment mode has an implicit index which is initialised by the 232402 order. Each time an auto-increment Save/Set Point is encountered the implicit index is altered, see Figure 3. If the Save Point increment is +1 and the Set Point increment is -1 then a stacking mechanism is achieved.

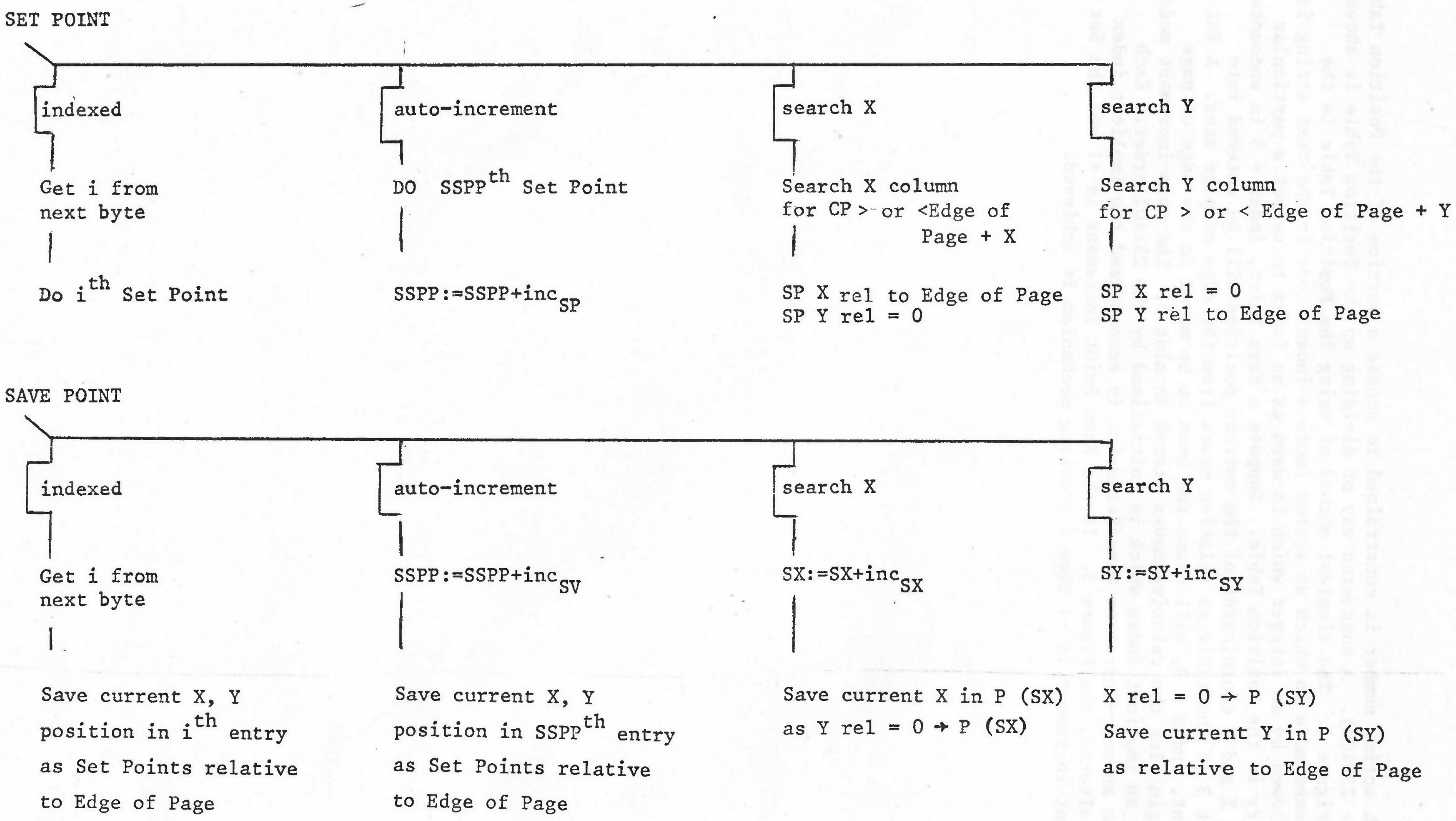

SET POINT
indexed
Get i from next byte
Do i^{th} Set Point
auto-increment
DO $SSPP^{th}$ Set Point
$SSPP:=SSPP+inc_{SP}$
search X
Search X column for CP > or <Edge of Page + X
SP X rel to Edge of Page
SP Y rel = 0
search Y
Search Y column for CP > or < Edge of Page + Y
SP X rel = 0
SP Y rel to Edge of Page
SAVE POINT
indexed
Get i from next byte
Save current X, Y position in i^{th} entry as Set Points relative to Edge of Page
auto-increment
$SSPP:=SSPP+inc_{SV}$
Save current X, Y position in $SSPP^{th}$ entry as Set Points relative to Edge of Page
search X
$SX:=SX+inc_{SX}$
Save current X in P (SX) as Y rel = 0 → P (SX)
search Y
$SY:=SY+inc_{SY}$
X rel = 0 → P (SY)
Save current Y in P (SY) as relative to Edge of Page

Figure 3

If the Set Point increment is +1 then the Electric tab mechanism can be easily implemented by pre-setting the X column of the Position Table to Set Point X relatives and the Y column to zero relative Y orders. The search X (Y) mode implements horizontal (vertical) tabs by searching the X (Y) column only for a new Edge of Page X (Y) value which is greater (modes 7, 10) than the current X (Y) coordinate. If no such value exists the current position is unchanged.

The Save Point mechanism may be used to simulate the Tab Stop key on a typewriter ie tab positions may be setup from within the text string.

All the entries in the Position Table must be Set Point relative orders and may be preset by the 232401 order.

3.4 Super/Subscripting mechanism

The Super/Scripting mechanism is a 'shift' system implemented by a stack in DRIVER. Encountering a super/subscript action causes stacking of the current position and the current parameters associated with the text (height, spacing etc). A new set of parameters is installed which are fractions (see 232405 order) of the previous character height and spacing. Plotting then continues normally, plotting the super/subscript string until either an action 35 or 36 is met. Both cause the unstacking and restoration of the previous text parameters; the difference is that action 35 causes the previous current position to be restored from the stack whereas action 36 causes only the previous Y coordinate to be restored, the current X coordinate being unchanged.

Action 35 allows subscripts to be placed directly under superscripts. Action 36 allows concatenation, see examples 1 and 2 in figure 5.

The starting position of a super/subscript string is determined by the X and Y fractions on the 232405 order. The Y fraction causes the current position to be changed to $CPY + \frac{YM}{YD} *$ character height.

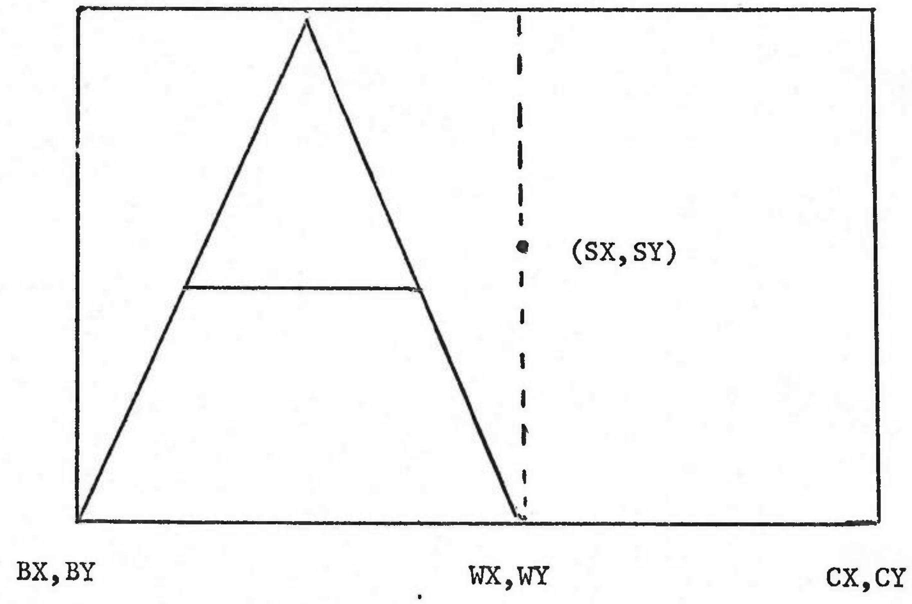


Figure 4

The X coordinate will be calculated as:

$$X' := CX - (\text{backspace } X) + \frac{XM}{XD} * \text{character width.}$$

In proportional spacing the width of the previous character is known exactly. In mono it will be an approximation based on the font aspect ratio:

$$\text{ie} \quad X' := CX - (\text{backspace } X) + \left(\frac{XM}{XD} * \frac{\text{FONTwidth}}{\text{FONTheight}} * \text{character height} \right)$$

For example, if XM/XD = 1/1 and YM/YD = 1/2 then a superscript starting position (SX, SY) would be as shown in Figure 4. (CX, CY) - (WX, WY) represents the intercharacter spacing value.

Local modifications may be made to the fractions controlling Super/Subscripting by using actions 37-43. These modifications last only until an action 35 or 36 unstacks the particular entry, see example 3 in Figure 5.

Repeated application of the fractions could cause the character height to be reduced below a visible threshold. To prevent this a limiting depth may be set in the 232405 order. Super/subscripting fractions beyond this depth limit will be set to 1/1 so no further reductions can occur.

1. $A^{BD}_{C}\ E$ $\qquad A \downarrow C \textcircled{b} \uparrow B\ D \textcircled{\Gamma} E$

2. $A^{B_{C}D}\ E$ $\qquad A \uparrow B \downarrow C \textcircled{\Gamma} D \textcircled{\Gamma} E$

3. $A^{B}\ C^{D}\ E$ $\qquad A \uparrow B \downarrow \textcircled{h}\ 2\ 1\ C \textcircled{\Gamma} D \textcircled{\Gamma} E$

Key: $\uparrow$ .... action 33 - superscript

$\downarrow$ .... action 34 - subscript

$\textcircled{b}$ .... action 35 - restore CX, CY

$\textcircled{\Gamma}$ .... action 36 - restore CY only

$\textcircled{h}$ .... action 41 - local mod to height

Figure 5

bb

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 24

FR80 DRIVER - Mk1 and Mk2 Scaling, Offsetting and Rotation

issued by
D C Sutcliffe
R W Witty

16 November 1977

DISTRIBUTION: F R A Hopgood
D A Duce
A H Francis
D C Sutcliffe
R W Witty

1. INTRODUCTION

It is proposed to enhance the scaling, offsetting and rotation SDF orders and monitor commands, of the FR80 Displayer, in DRIVER. DRIVER Mk1 will provide compatability with the present Displayer whilst Mk2 will incorporate additional orders to exploit these features. It is intended to be able to offset, scale, and rotate at any of three levels - see figure 1

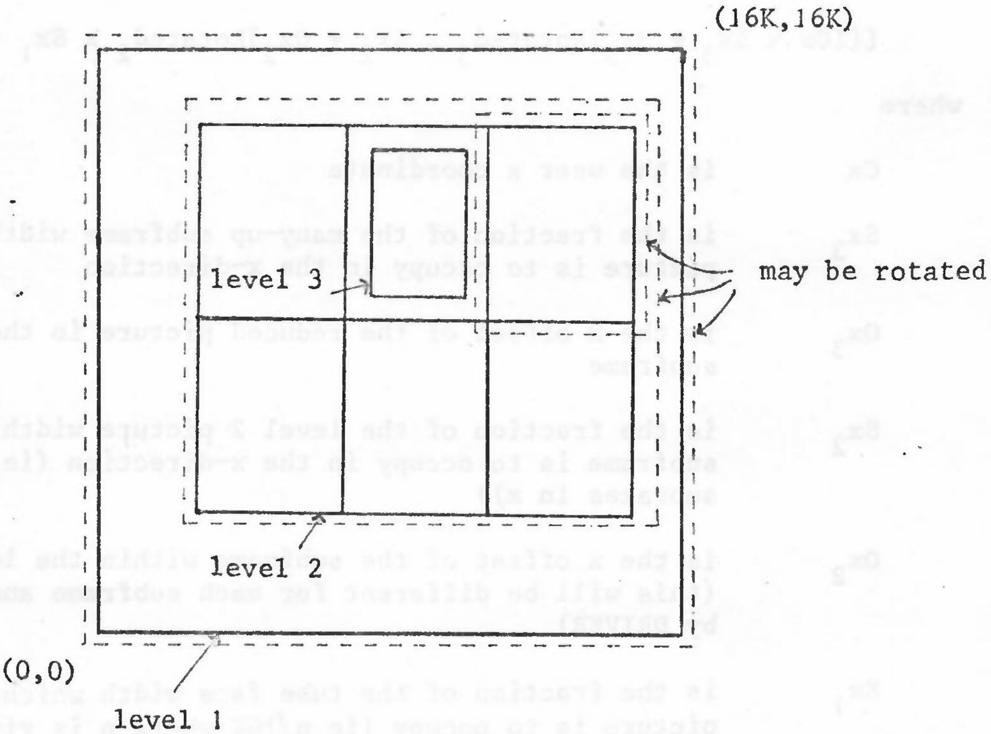


Figure 1.

At the first level the user's picture defined over the 16K x 16K raster may be scaled (differently in x and y), offset, and the whole scaled offset picture rotated by 0, 90, 180, or 270 degrees. At present it may only be scaled the same in x and y, offset, and rotated by 0 or 90 degrees (COMIC or CINE).

At the second level the level 2 picture may be split into several abutting subframes (not necessarily the same number of subframes in x and y), each addressed as 16K x 16K points. Each has a different offset from the origin calculated by DRIVER based on the number of subframes, in each direction, per frame. The whole set of subframes, the level 2 picture, may be rotated by any of the four angles above.

The third level of modification allows the user to specify the fraction of the subframe (separately in x and y so that a square region in a non-square subframe may be selected or vice versa) which is to be used for the picture (since some users do not want the subframes to abut). The offset of this area from the subframe origin may also be specified. The subframe itself may be rotated by any of the four angles.

The present Displayer combines subsets of these last two features to produce non-abutting square subframes which cannot be individually rotated.

At whichever level the user's picture is defined it will be defined on a 16K x 16K raster (even if this does not correspond to a square area on the PLS).

2. TRANSFORMATION FROM USER TO ACTUAL COORDINATES

The transformation from user's x coordinates to actual (PLS) x coordinates is given by:

$$[[[Cx \times Sx_3 + Ox_3]Rotated_3 \times Sx_2 + Ox_2]Rotated_2 \times Sx_1 + Ox_1]Rotated_1$$

where

Cx is the user x coordinate

Sx_3 is the fraction of the many-up subframe width which the picture is to occupy in the x-direction

Ox_3 is the x offset of the reduced picture in the many-up subframe

Sx_2 is the fraction of the level 2 picture width which each subframe is to occupy in the x-direction (ie 1/(number of subrames in x))

Ox_2 is the x offset of the subframe within the level 2 picture (this will be different for each subframe and is calculated by DRIVER)

Sx_1 is the fraction of the tube face width which the level 2 picture is to occupy (ie n/16K where n is given by the SETSIZE command)

Ox_1 is the x offset of the level 2 picture to produce the level 1 picture

Rotated_i (i = 1,2,3) is the transformation (involving both x and y coordinates of the point) necessary to rotate the picture by 0, 90, 180 or 270 degrees about the origin and shift the picture back on top of itself. This is achieved by rotating the picture modulo 16K.

Each Sx_i must be less than or equal to 1. (This will be calculated as n/d, where n and d are single length, by extending n to double length with zeros, dividing out, rounding the quotient if necessary (by comparison of the remainder and denominator), and always using the quotient remembering the binary point is at the left hand end.)

A corresponding formula holds for the transformation of user y coordinates to actual y coordinates.

If the hardcopy camera is being used then the final image needs reflecting in y immediately before plotting - see Section 4.

3. TRANSFORMATION FROM ACTUAL TO USER COORDINATES

It is necessary to be able to carry out the reverse transformation to that described in Section 2 (eg after a character string has been plotted by hardware, the actual endpoint can be found but the user endpoint is required).

The reverse transformation for x is given by:

$$([[([Ax]\text{Unrotated}_1-Ox_1)/Sx_1]\text{Unrotated}_2-Ox_2)/Sx_2]\text{Unrotated}_3-Ox_3)/Sx_3$$

where

Ax is the actual x coordinate

Ox_i (i=1,2,3)
are as in Section 2
Sx_i (i=1,2,3)

Unrotated_i (i=1,2,3) is the transformation necessary to rotate the picture by 0, -90, -180 or -270 degrees (as appropriate) about the origin and shift the picture back on top of itself.

Again a corresponding formula holds for the transformation of actual y coordinates to user y coordinates.

If the hardcopy camera is being used then a reflection in y must be performed before the reverse transformation is applied - see Section 4.

4. HARDCOPY REFLECTION

All the FR80 cameras except the hardcopy camera have an optical system which reduces the size of the image on the PLS before it is recorded on the film. The hardcopy camera, however, has an optical system which enlarges the image before it is recorded on the film. This difference is necessary because the size of the frame on all film types except hardcopy is smaller than the tube face whereas the size of the frame on hardcopy is larger.

The optical system of the hardcopy camera, unlike those of the other cameras, reflects the image in both x and y. It is not necessary, though, to reflect the image in both x and y before plotting as another difference exists. With all film types except hardcopy the image is drawn on the back of the film and viewed from the front. With hardcopy the image is drawn on the same side as it is viewed, which gives a reflection in x. Hence, it is only necessary to reflect the image in y before it is plotted.

5. CHARACTER ROTATION

If hardware characters are being used, then rotation of characters by one of the angles 0, 45, 90, 135, 180, 225, 270, or 315 degrees is achieved by the hardware.

However, if the software simulator is being used, then rotation of characters must be done by software. Since the output of the software simulator always consists of relative moves, the rotation of characters becomes simplified. Each end point must be rotated θ degrees (where θ is one of the above eight angles) about the start point. Since the endpoint is relative to the start point, the problem reduces to one of rotating the given coordinates by θ degrees about their origin. That is, if x,y are the given end points, then the rotated end points x^1,y^1 are given by:

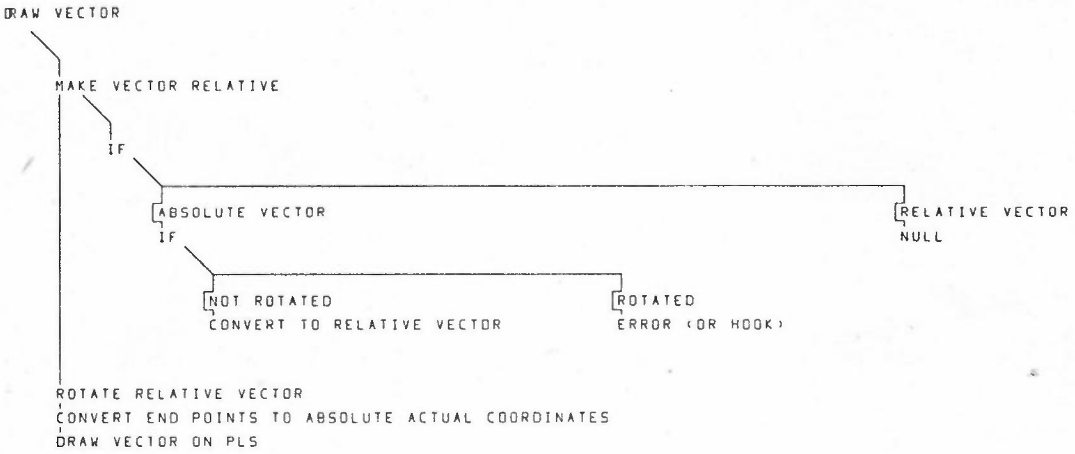
$$x^1 = x \cos \theta - y \sin \theta$$

$$y^1 = x \sin \theta + y \cos \theta$$

By storing the sines and cosines of the eight angles (as binary fractions if necessary) this can be easily executed.

A vector drawing routine is proposed which will draw possibly rotated relative vectors. This would take a relative vector, rotate it by one of the eight angles above (ie including 0 degrees meaning no rotation), convert the new end points to absolute actual coordinates and draw the vector. It will handle all the output from the character drawing software simulator.

However, an absolute vector is a relative vector from the current point with no rotation and so the above routine could be extended to draw all vectors. Thus:



would be sufficient to draw all vectors. If at a later stage it is found to be necessary to draw rotated absolute vectors then this can be incorporated into this routine.

1b

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 25

FR80 DRIVER - Picture and Text Rotations

issued by
D C Sutcliffe
R W Witty

28 November 1977

DISTRIBUTION: F R A Hopgood
D A Duce
A H Francis
D C Sutcliffe
R W Witty

1. INTRODUCTION

This note discusses the difference between the rotation of a whole picture and the rotation of a text string. Pictures are defined over a square coordinate system and mapped onto a square PLS area. It is sensible, therefore, only to allow 90° rotations of whole pictures. Text strings may be rotated by any angle though in practice this will be restricted to 45° angles to match the hardware features of the FR80. Rotation of a whole picture involves 'shifting' the picture back onto the PLS. Text rotation does not require shifting.

2. PICTURE ROTATION

The general equation for rotation through an angle θ is:

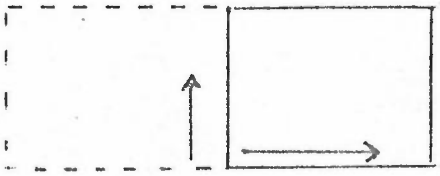
$$x' = x \cos\theta - y \sin\theta$$

$$y' = x \sin\theta - y \cos\theta$$

Picture rotation may occur through 0°, 90°, 180°, and 270°. These are special cases as the sin and cos values are always 0 and ±1.

For $\theta = 0^\circ$ clearly $x' = x$, $y' = y$

for $\theta = 90^\circ$



The equation is $x' = -y$, $y' = x$

For example: $-(16k-1) \leq x \leq 0$, $0 \leq y \leq (16k-1)$

After rotation the picture is 'outside' the PLS area and must be shifted back onto the PLS. The shift for $\theta = 90$ is (16K-1) in x ie $x' = -y + (16K - 1)$.

The full set of rotation and shift equations is thus:

0^{o} $\quad x' = x \qquad y' = y$

90^{o} $\quad x' = -y + (16K - 1) \qquad y' = x$

180^{o} $\quad x' = -x + (16K - 1) \qquad y' = -y + (16K - 1)$

270^{o} $\quad x' = y \qquad y' = -x + (16K - 1)$

3. TEXT ROTATION

Text rotation may be at 0^{o}, 45^{o}, 90^{o}, 135^{o}, 180^{o}, 225^{o}, 270^{o} and 315^{o} angles. Rotated coordinates may be found from the general formula:

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta - y \cos \theta$$

Text strings must wraparound the PLS area when drawn 'off the edge' so all arithmetic must be modulo 16K. This ensures that parts of a character drawn along the X axis ($y = 0$) when rotated by 90^{o} appear along the Y axis ($x = 0$). The rest of such a character drawn at $y>0$ must wraparound to produce values $0 \leqslant x \leqslant (16K - 1)$. Examples of text rotation equations are:

0^{o} $\quad x' = x \quad y' = y$

45^{o} $\quad x' = [x(.7071) - y\,(.7071)]_{16K} \quad y' = x[(.7071) + y\,(.7071)]_{16K}$

90^{o} $\quad x' = [-y]_{16K} \quad y' = x$

Where $[\ ]_{16K}$ indicates modulo 16K arithmetic.

There is no shifting involved in text rotation.

gm

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 26

FR80 DRIVER - Mk2 Monitor Commands

issued by
R W Witty

December 1977

DISTRIBUTION:

F R A Hopgood
A W Burraston
P M Nelson
D C Sutcliffe
A H Francis
R W Witty
R Brandwood
D A Duce

1. INTRODUCTION

This paper proposes a set of Monitor Commands for Mk2 DRIVER. Those commands marked with an asterisk are new commands not available in the III Displayer; all other commands are the same as their III counterparts. This is a proposal - your comments and suggestions are welcome.

* ACTUAL CAMERA
ADVANCE
BACK
* BROADCAST
BY COLUMNS
BY ROWS
CAMERA
CLEAR
CONTINUE
CTRL A
CTRL D
CTRL I
* CTRL P
CURRENT PAGE
END JOB
FOCUS
FRAME
GO
HITS
IMAGES-FICHE
MAKE FILM
* MANYUP
* MESSAGE
* MOVE TO FRAME
PARITY
OFFSET
PITCH-MARGIN
PULLDOWN
* REEL
REWIND
ROTATION
SET SIZE
SIZE OF TITLE
SKIP
SPEED OF VECTOR
* STATISTICS
TAPE TYPE
TITLE
TRY AGAIN
USE
XNUM
YNUM

2. MK2 MONITOR COMMANDS

* ACTUAL CAMERA/n←

Operator only command specifying the number of the actual camera loaded. Used only when camera changed or system disk reloaded.

n = 1 - CL16

2 - BW16

3 - CL35

4 - BW35

5 - PR16

6 - MFCH

7 - HCS

8 - HCM

ADVANCE/[n]←

Advances the film n frames. Each frame advance is equal to the number of steps set with the PULLDOWN command. If omitted, n = 1. With 105 mm film, the camera will advance n fiche.

BACK/[n]←

Backspaces the data to the start of the current file if no parameter is entered. If a number is entered, the tape will backspace that number of records. If already at the beginning of a file, the tape will backspace to the beginning of the previous file. If at loadpoint, it does nothing. When backspacing records, file marks stop the tape. Thus, BACK/9999← sends the tape to the end of the previous file.

* BROADCAST/N, text←

Allows the operator to set up a 'message of the day' which will be inserted into the users' ID frames. 'n' specifies the line number and text is any string of less than 60 characters terminated by a carriage return. In this way n lines of text may form the message, entered by n BROADCAST commands.

If n = 0 the current broadcast message is deleted so nothing will appear in ID frames.

BY COLUMNS See section 3

BY ROWS See section 3

CAMERA/n←

Selects a camera so that the program can limit the area of the light source which you can use in order to be compatible with the aperture of the camera being used. The program will also set the number of steps required to complete the CLEAR command. Select n as follows:

Parameter	Camera
3	35mm sprocketed
4	16mm sprocketed
16	hardcopy
* 105	microfiche 105mm

CLEAR/←

Advances the camera sufficiently to move all exposed film into the takeup magazine.

CAM/3	=	ADVANCE/36	frames
CAM/4	=	ADVANCE/100	frames
CAM/16	=	ADVANCE/6	frames
CAM/105	=	ADVANCE/5 fiche	frames

CONTINUE/[n]←

Restarts the program at the location where it was last interrupted. The film is not advanced. n specifies the number of frames to be made. If n is omitted, the job will continue until normal completion.

CTRL A — Interrupts the program after the next frame advance and return control to MONITOR.

CTRL D — Interrupts the program instantly and transfers control to DEBUG.

CRTL I — Interrupts the program instantly and returns control to MONITOR.

* CTRL P — Causes printing out on teletype of a summary of job's progress so far ie elapsed time, number of subframes, frames, magnetic tape blocks etc.

END JOB/←

Completes the titling of a partial fiche at the end of a job.

FOCUS/[m][,n]←

Causes the focusing image to appear on the monitor screen and the light source. You may select an intensity (m) and spot size (n). The range for both is from 0 to 7. If omitted, m = 7 and n = 0.

In some programs the FOCUS command may be used to make notations on film. Follow these steps:

(a) Activate the FOCUS command.

(c) Type any character. This will stop the display of the focusing image and advance the film one frame. If this feature is not available in the program being used, the program will return control to MONITOR.

(c) At this point, characters typed on the Teletype will flash on both the monitor screen and the light source, exposing the film . A ← may be used to create additional lines.

(d) Type CTRL I. The program will advance the film one frame and return control to MONITOR.

FRAME/n←

Sets the starting frame count number to n.

GO/[m][,[n][,o]]←

Advances the film one frame and starts the program. The program will record m frames or n files. When recording multiple files, it will advance o frames between files.

If no parameters are entered, the job will continue until normal completion. If m is specified, the program will record m frames or until normal completion, whichever comes first. If n is specified, the program will record n files or to a DOUBLE END OF FILE, whichever comes first. If omitted, o = 1.

HITS-CHAR,VEC,PTS,TITLE,CUT/[m][,[n][,[o][,[p][,q]]]]←

Sets the program to hit characters, vectors, points, fiche titles, and fiche cut marks m,n,o,p, and q times, respectively. Omitted parameters remain unchanged.

IMAGES-FICHE

See section 3.

MAKE FILM/n←

Sets the program to either the make film (n = 1) or the view only (n = 0) mode. The view only mode suppresses the film advance, eliminates some typeouts, and blanks the light source so that no film will be exposed.

* MANYUP/pictures per frame←

XMPY,XDIV,XOFF,YMPY,YDIV,YOFF,ROT←

Several pictures may be drawn on one frame. For each picture a set of scaling, offset and rotation parameters must be supplied. This allows 'free format' placement within the sub-frame.

* MESSAGE/n,text←

Same parameters are BROADCAST. The message is placed in the current job's trailing ID frame only.

* MOVE TO FRAME/n←

The tape will be positioned at the start of frame n, and control returned to MONITOR. No film will be exposed or advanced.

* PARITY/m,n←

Sets the program to read a data tape with either even (m = 0) or odd (m = 1) parity data blocks and even (n = 0) or odd (n = 1) file marks.

OFFSET X,Y/[m][,n]←

Moves the image in the X (m) and/or Y (n) axes. m and m, which may be positive or negative, are in scope points. An OFFSET command encountered on the data tape will override this command. Omitted parameters remain unchanged.

PITCH-MARGIN

See section 3.

PULLDOWN/n←

Sets the program to do n steps for each frame advance. For sprocketed cameras, n must be a multiple of four.

* REEL/m[,n]←

Indicates that this tape/job is the mth of n. n may be omitted except for the last reel which must have both parameters present and equal in value. The first reel is REEL/1.

Reel numbers >1 cause suppression of ID frames and frame advances so that multiple jobs may be 'abutted'. When m = n the trailing ID frame will be produced. When m = 1 and n > 1 then the leading ID frame will be produced and the trailing ID frame suppressed.

REWIND/[n]←

Rewinds the magnetic tape on unit #n. If omitted, n is equal to the current parameter of the USE command.

ROTATION/[n]←

Sets the program to record in either the upright (n = 0) or the rotated (n = 1) mode. In the rotated mode the image is rotated counterclockwise 90 degrees. If omitted, n = 0.

SET SIZE/n←

Sets the image size. n is the number of scope points to be used to represent 16,384 scope points.

SIZE OF TITLE

See section 3

SKIP/[n]←

Positions the data tape at the start of the next file if no parameter is entered. If a negative number is entered, the tape will skip that number of files. If a positive number is entered, the tape will skip that number of records. When skipping records, file marks stop the tape.

SPEED OF VECTOR/n

n = 1, 2, 4 or 8 msec vector speed.

* STATISTICS/m,n←

If m = 1 then statistics about the job will appear on the trailer frame.
If n = 1 then statistics about the job will appear on the teletype.
m,n=0 will turn off this facility
Default n=n=0.
"Statistics" means a detailed account of the frequencies of individual SDF orders, CPU timings per task etc to allow detailed performance analysis of user jobs/DRIVER.

TAPE TYPE - 2, 5, 8 OR 9/n←

Sets the program to read a data tape which is either 7- or 9-track and is either 200, 556, 800 or 1600 bpi. Select n as follows:

PARAMETER	BPI	TRACK
2	200	7
5	556	7
8	800	7
9	1600	9
*10	800	9

TITLE See section 3

TRY AGAIN/[n]←

Makes repeated attempts to read after there has been an indicated tape error. n specifies the number of attempts you wish to make. This number will be used as the new re-try count. If omitted, n remains unchanged.

USE/n← Sets the program to read data from unit n.

X NUMBER UP/n

Sets the program to record n images per frame in the X axis.

Y NUMBER UP/n

Sets the program to record n images per frame in the Y axis.

3. SPECIAL INSTRUCTIONS FOR CREATING MICROFICHE

The microfiche camera is a multiformat microimage recorder which operates with moving film and a fixed lens. Figure 3-1 shows typical formatting. Note that the film is stepped across its width row by row from A1 to A6. After recording has been completed in the last row (A6), the film returns to the first row while simultaneously advancing one column. This brings the film to position B1.

The stepper motor system controls the distance between the center lines of the columns. The servomotor and control disk control the distance between the center lines of the rows. The number of columns for the margin setting is:

left margin + right margin + (column pitch/2)

The formula for the total length of a fiche is:

(columns per fiche x column pitch)
+ left margin + right margin

The desired image size is acquired in the normal manner, ie through the use of the operator commands available to the particular program being used.

When recording on 16mm film in linear mode, use the modified 16-1 control disk. The frame advance distance is set in one-quarter millimeters with the PULLDOWN command.

TABLE 3-11 shows operating parameters for standard microfiche formats. The format may be selected by the operator using the commands in TABLE 3-10.

BY COLUMNS/←

Sets the program to step vertically down a column of images before stepping across to the next column when there are multiple images in each frame.

BY ROWS/←

Sets the program to step horizontally across a row of images before stepping down to the next row when there are multiple images in each frame.

IMAGES-FICHE/m,n←

Sets the program to produce a fiche with m columns and n rows (including titles). n must be equal to or less than the maximum number of rows which it is possible to record with the control disk being used.

PITCH-MARGIN/m,n←

Sets the program to produce a fiche with a column pitch of m and a margin setting of n (see Par. 3.4).

SIZE OF TITLE/m,n←

Specifies the horizontal (m) and veritcal (n) dimensions of an image area. These parameters are used by the program to calculate character size, spacing, and line feed values for titling. The formulas are:

m = column pitch in mm x scope pints per mn

n = row advance in mm x scope points per mm

NOTE

Changing m and n will not affect a title which has already been entered into the program; the title must be reentered.

TITLE/titling information←

Specifies the format and content of fiche titles to be recorded on all subsequent fiche until another titling command is encountered. See Par. 3.5.

X - NUMBER UP/n←

Sets the program to record n images per frame in the X axis.

Y - NUMBER UP/n←

Sets the program to record n images per frame in the Y axis.

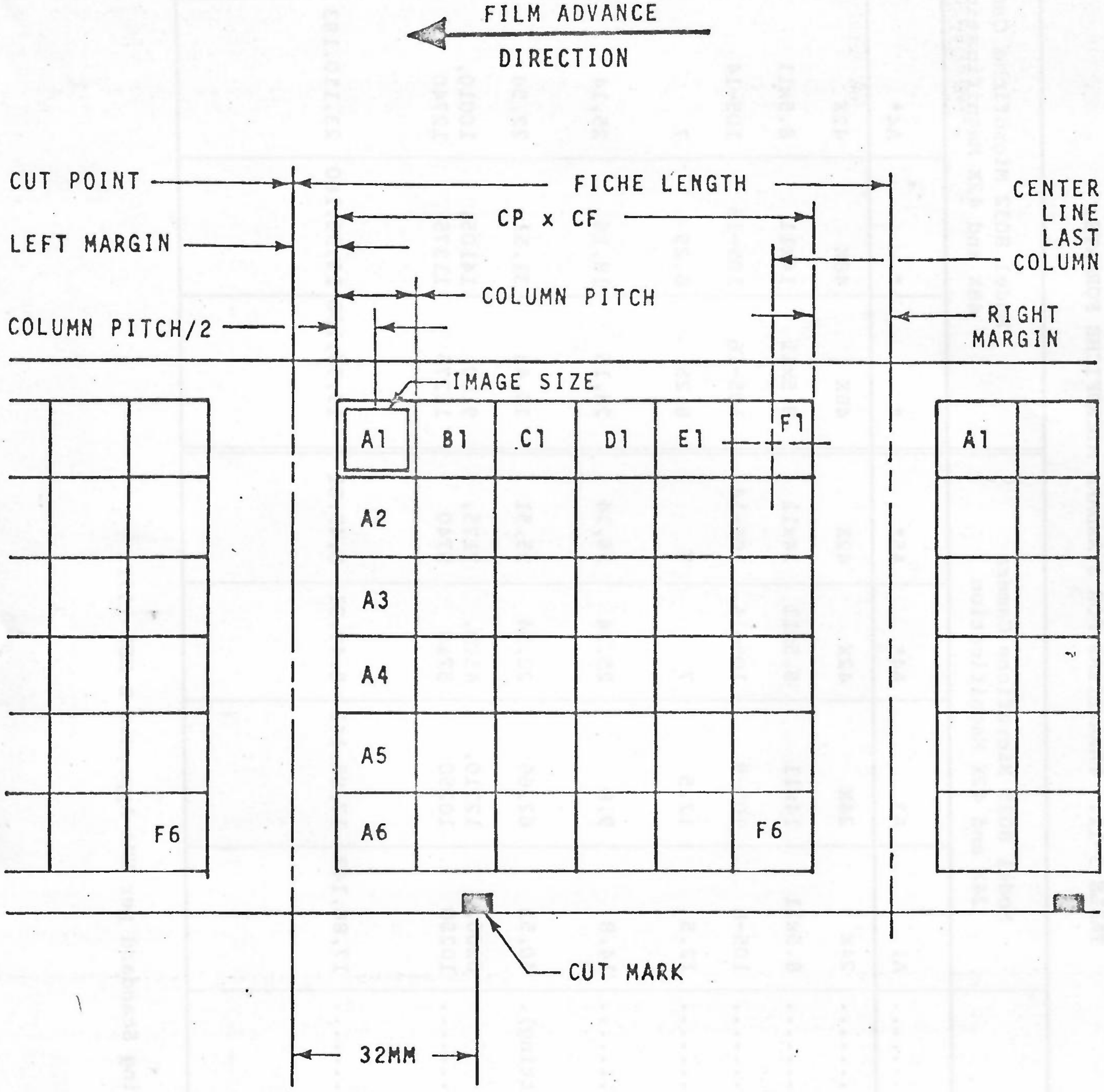


Figure 3-1. Microfiche Format.

TABLE 3-11. PARAMETERS FOR STANDARD MICROFICHE FORMATS.

	Model 8025 Microfiche Camera 24X and 42X Magnification				Model 8032 Microfiche Camera 48X and 42X Magnification			
NMA format	A1	A3	A4*	A5*	*	*	A4*	A5*
Reduction ratio	24X	24X	42X	42X	48X	48X	42X	42X
Page size (inches)	8.5x11	14x11	8.5x11	14x11	8.5x11	14x11	8.5x11	14x11
Control disk	105-8	105-8	105-14	105-14	105-16	105-16	105-14	105-14
Row advance (mm)..............	12.5	12.5	7	7	6.25	6.25	7	7
IMAGES-FICHE (columns,rows)	14,8	9,8	25,14	16,14	28,16	18,16	25,14	16,14
PITCH-MARGIN (column pitch,margin setting).	40,53	62,66	22,54	35,51	20,43	31,51	22,54	35,51
SIZE OF TITLE (X,Y)	8200, 10250	12710, 10250	4510, 5740	7175, 5740	9100, 11375	14105, 11375	10010, 12740	15925, 12740
(triple-I 15 computer) CHARACTERS - SIZE,SPACE,FEED	17,86,143	17,86,143	8,49,81	8,49,81	19,96,160	19,96,160	23,110,183	23,110,183

*Not NMA Format and Coding Standard per NMA Appendix B MS2-1971.

3.5 FICHE TITLING FORMAT

A fiche title may occupy any number of rows across the top of a fiche. (In the example, two rows are used.)

An integral number of title character positions (horizontally and vertically) may be assigned to each fiche image area in the title rows. The character size used for the title is determined by the number of characters specified horizontally per image. (In the example, the words "FICHE TITLE" have four character positions horizontally and two line positions vertically.)

Up to 128 characters (not including spaces) may be displayed in the title. Up to 16 different messages may be put into the title area.

The input for the titling may be either the Teletype (see TITLE, TABLE 3-10) or records on the data tape. In either case, the format is the same. On a print tape, a comma in the carriage control position causes characters from the printing positions to be interpreted as titling information.

Titling information will be in the following format:

a. The first character will be:

T	Beginning of title. (This title will supersede the previous title.)
C	Continuation of title information.
space	(from tape only) Step to next fiche with same title. (All subsequent characters will be ignored.)

b. The second character will be ignored unless the first character is a T, in which case it will be a digit telling how many rows will be used for the title.

c. Subsequent characters are in a free format, specifying titling parameters for the following titling message. Blanks are ignored. Each parameter specification consists of a letter followed by one or two numbers (separated by commas). The letter designates which parameter(s) will be set and the number the value to be assigned. A $ indicates that parameter assignments continue in the next record.

Parameter input will be terminated by a /, which initiates input of text for this title message.

d. Parameters:

Cn Number of character positions into which an image area will be divided (horizontally).

Hm,n Horizontal position for the first character of a message.

m = column number

n = character position within column (left character = 1)

Ln Number of line positions into which an image area will be divided (vertically).

Vm,n   Vertical position for the first character of a message.

m = row number

n = line position within row (top line = 1)

e. Text for a message may be any character in the printing set of the particular program being used. The $ is used as an escape code and the following character is a control code. Control characters are:

$ Put $ into title message.

L   Line return; return to initial character position of this message one line position lower.

C   Text continues in the next record beginning with the third character.

M   End of message; parameter assignments for the next message follow.

T   End of title.

a digit   The digit represents the number of characters to be used as a fiche number. The following n digits will be used as the initial fiche number (leading digits may be spaces). The fiche number will be incremented by one on each subsequent fiche.

Examples:

a. The following example will produce the title shown in Figure 3-2:

```
T2C4H1,1L2V1,1/FICHE TITLE$M$C

C C6H1,1L2V2,1/OCTOBER 12, 1970$M$C

C C8H4,1L4V1,3/RECORDED$L$L ON THE$C

C $L$L FR 80$M$C

C C4H5,2L2V1,1/NO. $3  1$T
```

b. The following example will produce a title with two lines of n characters in the top row. There will be five character positions across each image area:

```
T1C5H1,1L2V1,1/(n characters)$C

C $L(n characters)$T
```

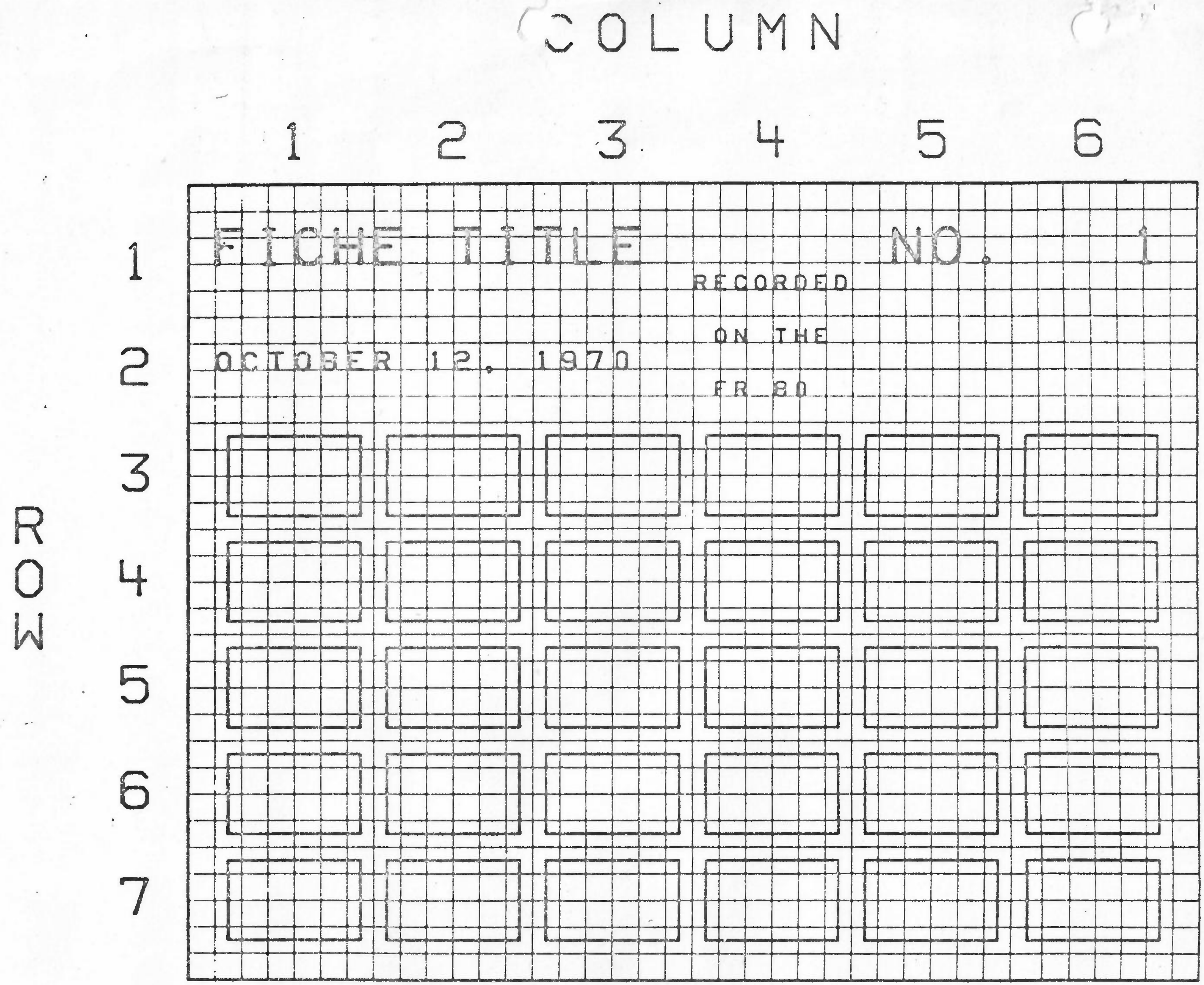


Figure 3-2. Sample Fiche Titling.

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 27

FR80 DRIVER - Scaling, Wraparound and the Character Generator

issued by
R W Witty

5 January 1978

DISTRIBUTION:	
F R A Hopgood	A W Burraston
M F Chiu	R Brandwood
D A Duce	R W Witty
A H Francis	P M Nelson
K M Crennell	D C Sutcliffe
C D Osland	

1. A PROBLEM WITH PLOTTING SCALED TEXT

The standard III Displayer allows Line Feed and Character Spacing values in the range 0-511 (9 bits). Line Feeds can only proceed 'down the page' and Character Spacing 'left to right' ie the normal English directions. FR80 DRIVER improves over the Displayer by allowing Line Feed and Character Spacing values in the range 0-16383 (14 bits) thus allowing Line Feeds to be either 'upwards' or 'downwards' or 'right to left'. This increased generality, needed for Chinese, Hebrew etc, complicates both the user-interface and the programming of DRIVER due to the subtlety of the implicit module 16K arithmetic which manifests itself as 'wraparound'.

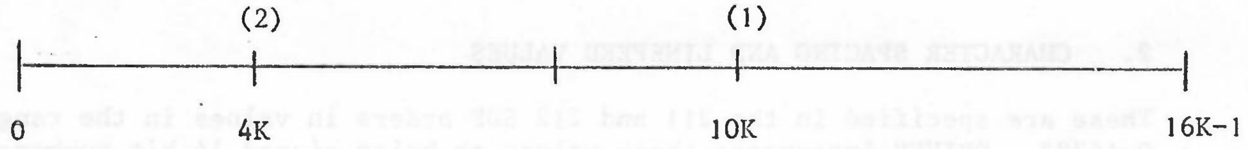


Figure 1. Unscaled, spacing = 10K

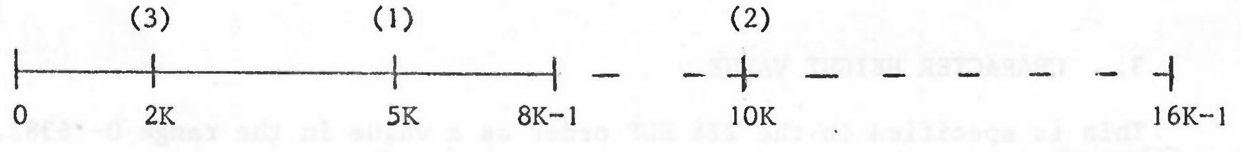


Figure 2. Scale factor = 1/2. Spacing (Scaled) = 5K

Consider figure 1. If the current position of the beam is the origin, then issuing a Character Space of magnitude 10K will cause the beam to move to position (1) ie (10K,0). If a second Character Space occurs the beam will move to position (2) ie (4K,0). This is because position (2) is, in X, (10K+10K) mod 16K, ie wraparound has occurred. Because of wraparound the Character Spacing can be thought of as either +10K (left to right) or -6K (right to left - see section 2). Consider the same sequence of events but scaled by a half. The Character Space is now actually 5K. The first occurrence of Character Spacing moves the beam to position (1) in figure 2 ie (5K,0). The second Character Space moves the beam to (5K+5K) mod 16K ie position (2), (10K,0). This is not what happened in figure 1 scaled down. What the user expects is that the beam will have moved to position (3) ie (2K,0) The reason for the discrepency is that the scale factor was only applied to the Spacing value and not to the modulus of the arithmetic used to calculate new positions. That is position (2) in figure 2 should have been calculated, in X, as (5K+10K/2) mod (16K/2). The FR80 hardware is fixed to wraparound at 16K and cannot vary as the scale factor varies, so the DRIVER software must simulate the variable modulus arithmetic system required to ensure correct scaling of pictures.

The FR80 has special hardware features to do Line Feed and Character Spacing modulo 16K. It is easy for DRIVER to simulate these features when scale factors other than unity are in operation for Line Feed and proportional spacing as these two amount to Set Point instructions which go through the software scaling routines. If the user plots mono-spaced characters with the high speed character generator then the Character Spacing is done by the hardware and so will be modulo 16K irrespective of the scale factor. This will cause manyup pictures and single-up pictures to give different results if the text being plotted 'goes off the edge'. The single-up will wraparound as expected. The manyup will not wraparound the sub-frame border as expected but only at the frame border, causing possible overwriting of one manyup frame by another. Perhaps the hygenic solution to this problem is for DRIVER to forcably change all high-speed mono-spaced text orders in manyup sub-frames into slow-speed orders in which characters are plotted one at a time and the spacing carried out by software, via the scaling routines similar to proportional spacing. This would ensure that all manyup subframes were scaled down replicas of their single-up equivalents.

2. CHARACTER SPACING AND LINEFEED VALUES

These are specified in the 211 and 212 SDF orders in values in the range 0-16383. DRIVER interprets these values as being signed 14 bit numbers in the range -8K to +8K-1 with spacing being left to right and linefeeds being downwards. Hence if the user wishes to specify an upwards linefeed of magnitude 6K he should should specify a downward linefeed of -6K which is expressed as 16K-6K ie 10K to be modulo 16K.

3. CHARACTER HEIGHT VALUE

This is specified in the 224 SDF order as a value in the range 0-16383. DRIVER interprets this value as an unsigned 14 bit number in the range 0-16383 representing the magnitude of the character height.

4. CONCLUSION

FR80 pictures are defined over a 16K raster. The user exploits modulo 16K arithmetic to produce his picture. When the FR80 DRIVER software scales down a user's picture it implicitly scales down the 16K modulus of the arithmetic too. Since the hardware only operates modulo 16K there is a mismatch between unscaled and scaled subframes if full use is made of FR80 hardware features because the hardware fails to wraparound at the scaled subframe boundary. To ensure correct scaled wraparound the high speed option of the Character Generator cannot be used with manyup or scaling.

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 28

FR80 DRIVER - Proposed Operator Interface

issued by
R W Witty

12 January 1978

DISTRIBUTION:

F R A Hopgood
A W Burraston
P M Nelson
D C Sutcliffe
D Daniels
G Jones

A H Francis
R W Witty
R Brandwood
D A Duce
B Jeeves

1. INTRODUCTION

This paper outlines a design for the operator interface to FR80 DRIVER. This is a proposal - your comments and suggestions are welcome. It is intended that the DRIVER operator interface will be essentially the same as the Displayer in philosophy and operation. DRIVER will handle a superset of the Displayer Monitor Commands. Described below is a list of those aspects where DRIVER differs from the Displayer. If something is not mentioned we have probably overlooked it so tell us! The individual monitor commands are described in Discussion Paper 26. A list of the proposed DRIVER monitor commands is given in Section 9 and changes to Discussion Paper 26 in Section 10.

2. MONITOR SCREEN DISPLAY

It is proposed that the DRIVER display be essentially the same as the Displayer ie a list of all the available monitor commands together with their respective current parameter values where appropriate. The display will not be shown when processing Monitor Commands from tape ie loadgo mode. It will be shown continuously when the command input is from the teletype.

DRIVER will possess one additional feature on the Monitor Screen Display, namely a display of the current line being input from the teletype. This will take account of character deletes (a new DRIVER feature - Section 3) so that the operator can see exactly what is being input.

DRIVER will also possess a command to allow the display character height to be altered (zoomed) as is currently available on some III software products but not the Displayer.

3. PROMPTING AND INPUT OF CHARACTERS BY THE OPERATOR

DRIVER will issue the prompt character '←' whenever it seeks operator input similar to the 1906A. 'CNTRL-X' will delete the whole of the line input so far, and '←' will delete the last character also similar to the 1906A.

4. COMMAND NAMES

As with the Displayer the operator will have the choice of typing in the shortest string of characters which uniquely identifies a command, or the full name, or a character string in between these two extremes. All characters input as part of a command name will be checked.

A command name may be terminated by one of three characters specified below:

Note that DRIVER's ':' is equivalent to the Displayer's '/' and that DRIVER's '/' does not print out the command in full thus speeding up operator communication

(1) '/' ..... terminates the name. Input parameters are then expected immediately.

(2) ':' ..... terminates the name. DRIVER then prints out the rest of the name in full and waits for input parameters.

(3) '=' ..... terminates the name. DRIVER then prints out the rest of the name in full followed by the current values of the associated parameters, and a carriage return, linefeed sequence.

5. INPUT PARAMETERS

Parameters are input as with the Displayer ie a series of values separated by commas and terminated by a carriage-return. Two adjacent commas signifies that the corresponding parameter is to remain unchanged. If less values are input than are associated with the command then the unspecified values remain unchanged.

6. ERRORS

Values input by the operator will be checked immediately and, if appropriate, the message '*ERR' followed by a number will be output. This short error message will speed operator communication in the case of well-known errors. If, after an error message, the character '!' is typed then DRIVER will type an explanatory message in English.

Errors detected in monitor commands on the tape will cause the short error message to be printed on the teletype and control to be transferred to monitor for operator action.

7. CNTL-A,-I,-P

The commands CNTL-A,-I,-P (see Discussion Paper 26) may only be used during plotting. They may not be issued when in monitor mode.

8. SWITCH 5

Data switch 5 will have the same effect as with the Displayer ie if 'GO' command is executed when switch 5 is on then control will pass to monitor and await operator action. If the operator types 'GO' when switch 5 is up then control will remain with monitor and the job will not begin plotting.

9. DRIVER MK2 MONITOR COMMANDS

42X
48X
ACTUAL CAMERA
ADVANCE
BACK
BROADCAST
BY COLUMNS
BY ROWS
CAMERA
CLEAR
CONTINUE
CNTRL A
CNTRL D
CNTRL I
CNTRL P
CURRENT PAGE
END JOB
FOCUS
FRAME
GO
HITS
IMAGES PER FICHE

MAKE FILM
MANY UP
MESSAGE
MOVE TO FRAME
PARITY
OFFSET
PITCH-MARGIN
PULLDOWN
REEL
REWIND
ROTATION
SETSIZE
SIZE OF TITLE
SKIP
SPEED OF VECTOR
STATISTICS
TAPE TYPE
TITLE
TRY AGAIN
USE
X NUMBER UP
Y NUMBER UP
?

10. AMENDMENTS TO FR80 DISCUSSION PAPER 26

1. Two new commands, '42X' and '48X', will cause the PITCH-MARGIN, IMAGES PER FICHE and SIZE OF TITLE to be set to appropriate default values for 42X and 48X magnification fiche.

2. The command '?' will cause the Monitor Screen Display to be printed, in full, on the teletype. This command exists in the Displayer but was missed out of Discussion Paper 26.

3. The parameters to the MANY UP command have been changed. The number of subframes per frame is controlled by 'XNUM' and YNUM' so 'pictures per frame' is deleted from MANYUP. The parameters to the MANYUP command now specify the scaling, offset and rotation of the picture within any sub-frame.

4. The FOCUS command's parameters should be $0 \leqslant m \leqslant 255$ and $0 \leqslant n \leqslant 7$

5. A whole paragraph is missing from Section 3, Special Instructions For Creating Microfiche. It is in the III Manual.

6. TAPE TYPE may now only take the values 2, 5 or 8.

gm

SCIENCE RESEARCH COUNCIL

RUTHERFORD LABORATORY
ATLAS COMPUTING DIVISION

FR80 DISCUSSION PAPER 29

FR80 DRIVER - Proposed Image Processing Formats

issued by
P M Nelson
8 March 1978

DISTRIBUTION:

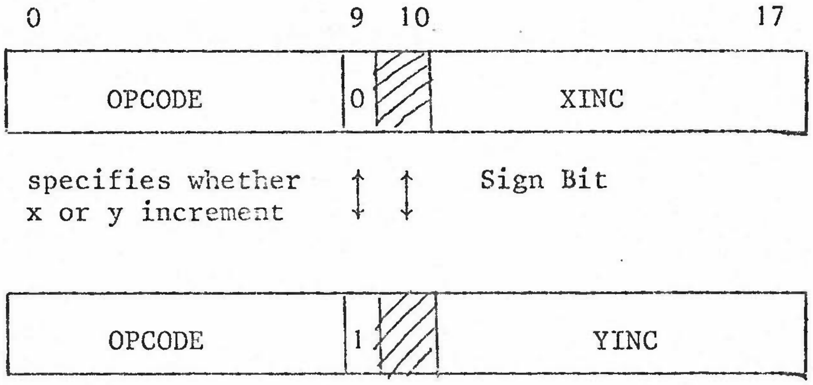
F R A Hopgood
D A Duce
C D Osland
D C Sutcliffe
A W Burraston
R Brandwood
R W Witty
P M Nelson

In order to reconstruct an image in colour and with a range of intensities using the FR80 Plot Current Point command, a relatively large amount of data is required using the FR80 Displayer format. By employing a system somewhat similar to the FRSCAN (SMOG) routine with regard to data compaction, but without the density per unit area system (or half-tone effect), the amount of tape written by the host computer should be reduced by a factor of 10 as should the FR80 processing time. This will make the FR80 attractive to the Image Processing groups who are presently discouraged by finding that a full frame image takes more than one hour to plot and more than one tape to hold the data.

FR80 Commands required

1. Use 'X Move Absolute' and 'Y Move Absolute' to position start.

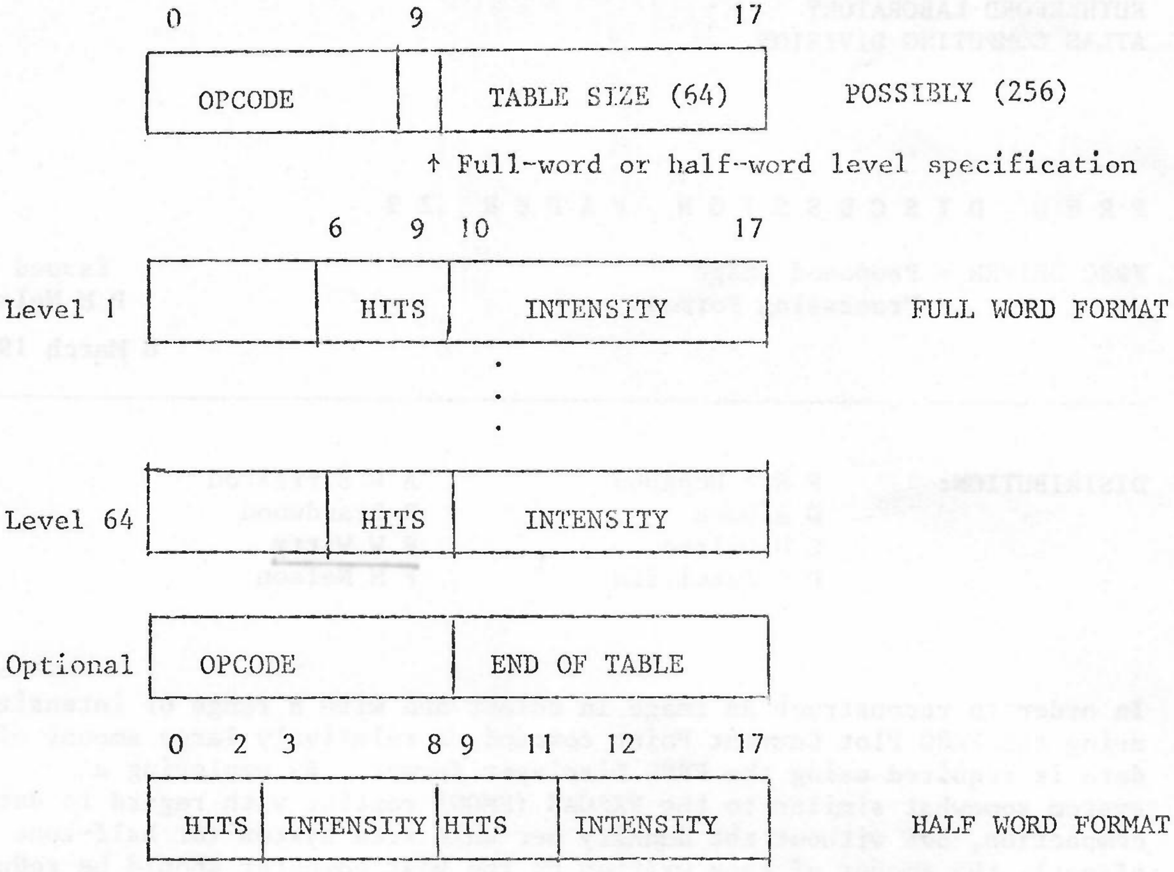
2. New command: SET X OR Y INCREMENT



Possible SMOG routines:

SETXI(XINC) and SETYI(YINC)

3. New command: LOAD TRANSLATION TABLE

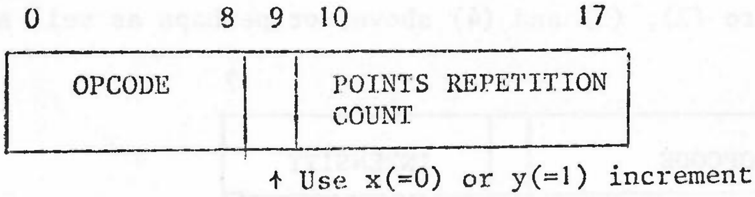


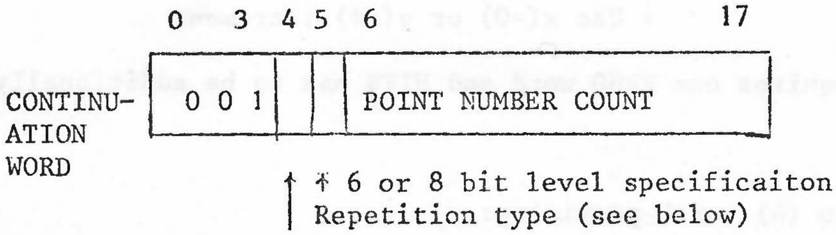
Possible SMOG routine:

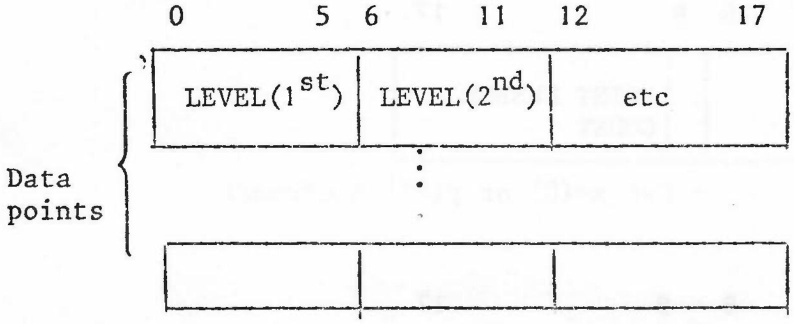
LOADTT(ARRAY,SIZE,WORDS)

where ARRAY is an array containing the Levels specification, SIZE is the number of levels or table entries and WORDS indicate full or half-word specifications for the table entries, ie WORDS=SIZE or WORDS=SIZE*0.5.

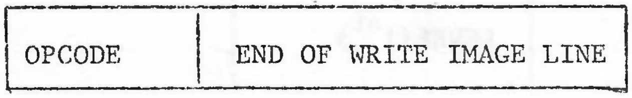
4. New Command: WRITE IMAGE LINE



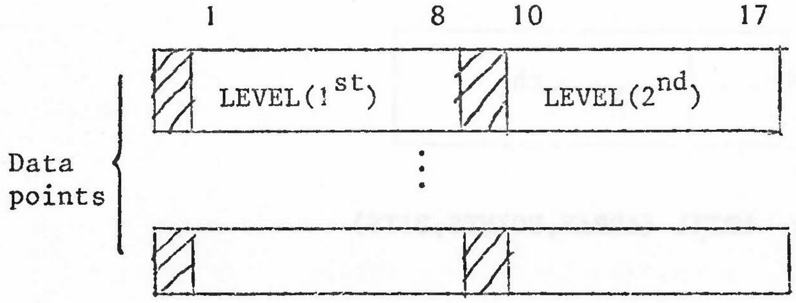




Optional:



Alternatively, for 8-bit data



Possible SMOG routine: WRTIML (ARRAY, PNTREP, POINTS, BITS)

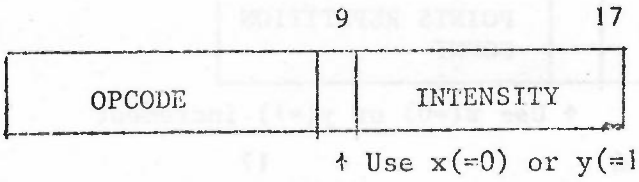
Where ARRAY contains the data points and where PNTREP is the number of times the total data point sequence is repeated ie

[LEVEL(1st),LEVEL(2nd)etc]*PNTREP
or LEVEL(1st)*PNTREP,LEVEL(2nd)*PNTREP etc

depending on bit 4 in the continuation word. POINTS is the number of data points and BITS is either 6.0 or 8.0 depending on the data. The last argument may be superfuous in that it should relate to the table size in LOADTT.

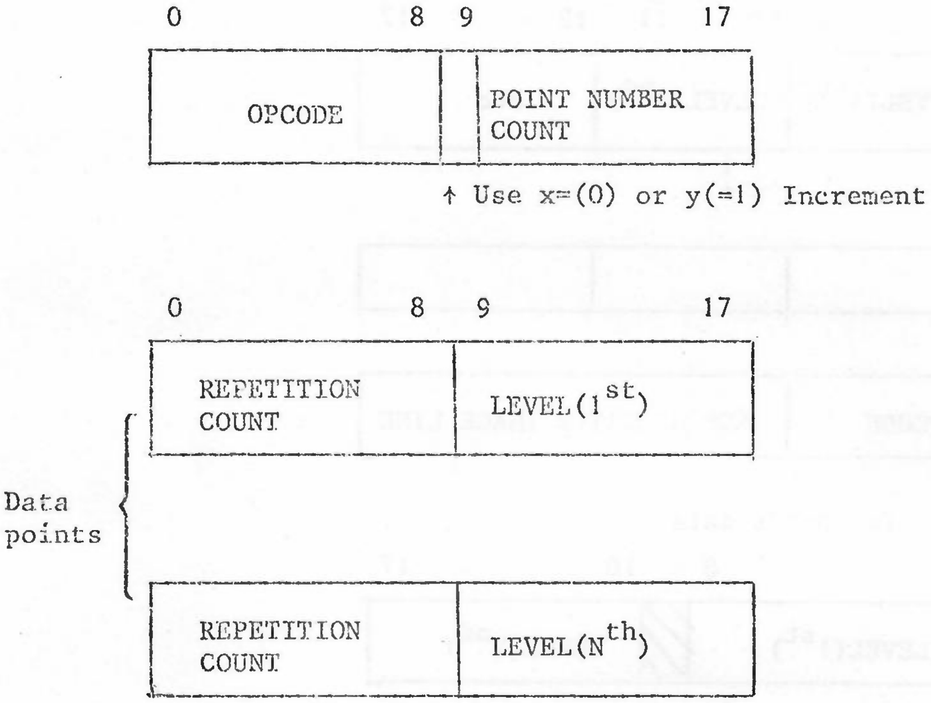
It would be useful if all eight data tracks on a 9-track tape could be used for the points data. Although a new tape block could be forced at the start of such data, if necessary, some limitations may exist on the FR80 to make this unattractive.

An alternative to (2), (3) and (4) above, or perhaps as well as:



Where each point requires one FR80 word and HITS has to be additionally specified.

Another variation to (4) worth providing:



Possible SMOG routine: WRTIL (ARRAY,POINTS,BITS)

Where ARRAY contains the data, each point with its associated repetition count. POINTS is the number data points and BITS is either 6.0 or 8.0 as before.

R. W. Witty

SCIENCE RESEARCH COUNCIL
RUTHERFORD LABORATORY
ALTAS COMPUTER DIVISION

FR80 DISCUSSION PAPER 30

Scaling, Wrapround and Modular Arithmetic

H K F YEUNG
27 NOVEMBER 1978

DISTRIBUTION F R A Hopgood
R E Thomas
D C Sutcliffe
H K F Yeung
R W Witty

1 INTRODUCTION

The problem of wrapround with respect to scaling has been described in Discussion paper 27. Here we shall look at it again in terms of Modular Arithmetic.

Methods adopted by the current version of DRIVER and the problems that it may have are discussed.

Modifications that we would like to make in light of the modular arithmetic discussed so far are suggested in section 7. The suggestion is aimed at both single and many - up pictures.

2 MODULAR ARITHMETIC

All we need is the following property :

$$n = n+m \ [\text{mod } m]$$

this is true even when n is negative, for example :

$$-2 + 2 = 0$$

$$\text{but } -2=m-2 \ [\text{mod } m] = m-2$$
$$\text{and } ((m-2)+2) \ [\text{mod } m] = 0$$

3 NOTATIONS

In the following discussions some symbols are used which have fixed meaning, and they are listed below :

c = current value
d = value to be added to c
f = c + d
k = scale factor
c' = k*c
d' = k*d
f' = c'+d'
m = modulo value
m' = k * m
a = a very small positive value (used to simplify notation)

4 RULES

Listed below are rules that we use to manipulate the modular arithmetic. They reflect how the FR80 hardwares work with respect to this problem. The rules are inadequate as can be seen from the examples later. By the time we have studied all the examples, we shall be in a position to add more rules which in turn reflect the software needed.

(R1) $0 \leqslant c \leqslant m$

(R2) d can be positive or negative, but we can assume $|d| < m$

(R3) $k < 1$, but to simplify our arithmetic, we let $k=1/2$ in all examples

(R4) the modular arithmetic is applied to d (or d') before it is added to c, that is,
d [mod m] or d'[mod m]

(R5) the modular arithmetic is also applied to the result of the addition before the final value is delivered, that is,
f [mod m] or f' [mod m]

5 EXAMPLES

The best way to study this problem is by looking at examples. There are four cases :

(E1) d is postive and d+c < m

(E2) d is positive and d+c > m

(E3) d is negative and d[mod m]+c < m

(E4) d is negative and d[mod m]+c > m

In the first set of examples (suffixed by a), values of d and c are not scaled. It is known that results obtained in this case are correct.

In the second set of examples (suffixed by b), values of d and c are scaled by 1/2. The third set is similar to the second one , but the arithmetic is carried out differently according to the new rules.

(E1a)

let c = m/2

d = m/4

f = 3m/4

(E2a)

c = m/2
d = m/2 + a
f = (m+a)[mod m] = a

(E3a)

c = m/2
d = (-m/4) = 3m/4 (Rule R4)
f = (m/2+3m/4)[mod m] = m/4

(E4a)

c = m/2
d = -3m/4 = m/4 (Rule R4)
f = m/2 + m/4 = 3m/4

In the second set of examples, c' = c/2 , d' = d/2 and f' should be f/2 accordingly.

(E1b)

c' = m/4
d' = m/8
f' = 3m/8

(E2b)

c' = m/4
d' = m/4 + a/2
f' = m/2 + a/2

COMMENT : the answer is wrong

(E3b)

$c' = m/4$
$d' = -m/8 = m--m/8 = 7m/8$ (Rule R4)
therefore, $f' = m/4 + 7m/8 = m/8$

(E4b)

$c' = m/4$
$d' = -3m/8 = 5m/8$
therefore, $f' = m/4 + 5m/8 = 7m/8$

COMMENT : the result is wrong

6 CURRENT STATE OF DRIVER WITH RESPECT TO THIS PROBLEM

Before we propose additional rules, let us look at the way DRIVER handles this problem. DRIVER imposes an extra rule which states :

if $d > m/2$ (or $d' > m'/2$), d should be modified as follow
$d = d - m$ (or $d' = d' - m'$

Suppose $d=m/2+a$ (a is a small value, say), according to the rule

$d = m/2 + a - m$

$= -m/2 + a$

$= m - m/2 + a$ (Rule R4)

$= m/2 + a$

so d is unchanged, but

$d' = m/4 + a/2$

$= m/4 + a/2 - m'$

$= -m/4 + a/2$

$= 3m/4 + a/2$ (Rule R4)

(D1a)

$c = m/2$
$d = m/2 + a$
therefore, $f = m/2 + m/2 + a$

$= a$

(D1b)

$c' = m/4$
$d' = m/4 + a/2 = 3m/4 + a/2$ (see above)
$f' = m/4 + 3m/4 + a/2$ ~~= 3m~~

$= a/2$

(D2a)

$c = m/100000$ (that is, a small value)
$d = m/2 + a$
therefore,
$f = m/2 + m/100000 + a$

(D2b)

$c' = m/200000$
$d' = m/2 + a/2 = 3m/4 + a/2$
therefore, $f' = 3m/4 + m/200000 + a/2$

COMMENT : the result is not correct.

(D3a)

$c = 3m/4$
$d = m/2 - a$
$f = 3m/4 + m/2 -a$

$= m/4 - a$

(D3b)

$c' = 3m/8$
$d' = m/4 - a/2$ (special rule does not apply)
therefore, $f' = 5m/8 -a/2$

COMMENT : again, this is wrong.

7 ADDITIONAL RULES

It is known that the modulo value for d' and f' should be m' instead of m. But there is nothing we can do because it is dictated by the FR80 hardwares. The additional rules below suggests what softwares are needed to cure this problem.

(A6) this rule replaces (R4) and it states :

if $d'<0$, it is to be modified as

$d' = m' + d'$

(A7) if $d'+c' > m'$, d' is modified as follows :

$$d' = d' + (m-m')$$

(E1c)

$c' = m/4$
$d' = m/8$
$f' = 3m/8$

(E2c)

$c' = m/4$
$d' = m/4 + a/2$

$= (m/4 + a/2) + (m - m')$ (Rule A7)

$= 3m/4 + a/2$
$f' = (m/4+3m/4+a/2)[\text{mod } m]$

$= a/2$

(E3c)

$c' = m/4$
$d' = -m/8$

$= m' -m/8 =m/2 -m/8$ (Rule A6)

$= 3m/8$
but $c'+d' > m'$, therefore, $d' = 3m/8 + (m-m')$

$= 7m/8$
$f' = m/4 + 7m/8 = m/8$ (Rule R5)

(E4c)

$c' = m/4$
$d' = -3m/8$

$= m' - 3m/8$ (Rule A6)

$= m/8$
$f' = m/4 + m/8$

$= 3m/8$

(D1c)

$c' = m/4$
$d' = m/4 + a/2$

but $c' + d' > m'$, therefore,

$$d' = m/4 + a/2 + (m-m') = 3m/4 + a/2$$

$$f' = (m/4+3m/4+a/2) \ [\text{mod } m] = a/2$$

(D2c)

$$c'=m/200000$$

$$d' = m/4 + a/2$$

$$f' = m/4 + m/200000 + a/2$$

(D3c)

$$c' = 3m/8$$

$$d' = m/4 -a/2$$

but $c' + d' > m'$, therefore

$$d' = m/4 - a/2 + m - m'$$

$$= 3m/4 - a/2$$

$$f' = (3m/8+3m/4-a/2)[\text{mod } m] = m/8 -a/2$$

8 LIMITATION

There is one limitation (perhaps more if we actually implement it) and it is :

> If characters are generated by the high speed character generator all characters in the same string (after the second one) must be in the same side of the second character.

For example, if we want to generate A,B,C,D,D,E, ... (in this order), we can have :
(P1) A B C D E ...
OR
(P2) ...E D C B A
BUT NOT,
(P3) D E ... A B C
NOR,
(P4)C B A ... E D

In cases P3 and P4, there will be no problem if there is a carriage return between C and D.

9 CONCLUSION

It can be seen that the softwares required to make up for the shortcomings of the hardwares are easy to implement and the solution is applicable to both single and many-up pictures. Though we have not include offsets in our model, they should cause no more problem than complicating the arithmetic.

R. W. Witty

SCIENCE RESEARCH COUNCIL
RUTHERFORD LABORATORY
ALTAS COMPUTER DIVISION

FR80 DISCUSSION PAPER 31

FONT FACILITIES IN DRIVER

H K F YEUNG
18 JUNE 1979

F R A Hopgood	R E Thomas
R W Witty	P M Nelson
C D Osland	K M Crennell
D C Sutcliffe	H K F Yeung

ABSTRACT

This paper describes how to prepare new fonts on host machines and how to access them from user jobs. Facilities described here will not be available till release 4.

1. PREPARING A SOURCE FILE FOR A NEW FONT

A source file for a new font should consist of
(a) control records
(b) a dispatch table
(c) character definitions

1.1 CONTROL RECORDS

All control records should begin with a '/' and there are 3 types:

(a) identification record
this should be the first record of a source file and contains information about the font number and user name of that font. The format is:

```
/FONT=a-number USER=a-name
```

Allocation of font numbers will be a matter to be discussed by all parties concerned and will not be pursued further here.

(b) end of dispatch table record
the format is:

```
/END OF DISPATCH TABLE
```

this should be put immediately after a dispatch table.

(c) end of definition record
the format is:

```
/END OF DEFINITION
```

this should be the last statement of a file.

EXAMPLE (incomplete)

```
/FONT=100 USER=ME

<a-dispatch-table>

/END OF DISPATCH TABLE

<character-definitions>

/END OF DEFINITION
```

1.2 DISPATCH TABLE

A dispatch table is a list of variable names (to be interpreted as addresses) such that the definition of the character of code zero begins at the address designated by the zeroth element of the list and so on. Entries in the dispatch table need not be distinct. However a definition must exist for each distinct entry. Though a variable name can be a string of any letter and/or digits, the following format is recommended:

CCnnnn

where n is a digit from 0 to 7.

The maximum number of entries in a dispatch table is 256 (decimal), but there is no minimum from the users' point of view.

1.3 CHARACTER DEFINITION

Each definition should be preceded by a variable name (label) corresponding to those entered in the dispatch table. A variable name must be separated from the rest of the definition by a comma. A rest-of-the-definition is a string of 6-digits. Two digits are required to form one stroke command and they are listed below:

> NOTE : UNLESS SPECIFIED OTHERWISE, ALL COMMANDS ARE IN OCTALS, AND ALL MEASUREMENTS ARE IN DECIMALS.

COMMANDS	DESCRIPTIONS
00-07	terminates character with 1-8 spaces
10	provided 8 spaces and continues
11	bean off
12	beam on
13	terminates without spaces
14	draw east
15	draw north-east (slope=1)
16	draw north-east (slope=10/14 or 14/7)
17	draw north-east (slope=7/14)
20	draw north-east (slope=5/14)
21	draw north
22	draw north-west (slope=-5/14)
23	draw north-west (slope=-7/14)
24	draw north-west (slope=-10/14 or -14/7)
25	draw north-west (slope=-1)
26	draw west
27	draw south-west (slope=1)
30	draw south-west (slope=10/14 or 14/7)
31	draw south-west (slope=7/14)
32	draw south-west (slope=5/14)
33	draw south
34	draw south-east (slope=-5/14)
35	draw south-east (slope=-7/14)
36	draw south-east (slope=-10/14 or -14/7)
37	draw south-east (slope=-1)
76-41	repeat following stroke 2-31 (decimal) times

when there are two options in the slopes, users can specify either of them by an appropriate command (see section 4)

1.4 COMPLETE EXAMPLE

```
/FONT=100 USER=ME
```

```
CC0000
CC0001
CC0000
CC0002

/END OF DISPATCH TABLE

CC0000,
     141516
     172000
CC0001,
     202122
     732600
CC0002,
     110000

/END OF CHARACTER DEFINITION
```

1.5 CONTROL CHARACTERS

Control characters such as line feed and carriage return are handled by special routines, and there is no way a user can provide proper definitions for them. However, as described in Discussion paper 23, it is possible to norminate any characters to be control characters at run time. So there is no need to worry about them in preparing new fonts. All that is required is a slot in a dispatch table and a dummy definition (such as CC0002 above).

2. TRANSFERRING A FONT SOURCE FILE FROM HOST MACHINE TO FR80

Software is required not only to copy font source files from disk to tape, but also to patch files to the required format. Using the example in section 1.4, the final output will probably look like:

```
/FONT=100 USER=ME
.NAME ATL;FON100 BINARY
72000/

$CC0000
$CC0001
$CC0000
$CC0002
...
/END OF DISPATCH TABLE

      0
```

```
       0
CC0000,
       141516
       172000
       0
       0
CC0001,
       202122
       732600
       0
       0
CC0002,
       110000

/END OF CHARACTER DEFINITIONS

START
```

It is hoped that the program MOVEIT (currently in Prime 400) is suitable for this purpose after some minor modifications.

3. PROCESSING A NEW FONT ON FR80

A new font needed to be assembled on FR80 before it can be used in user jobs. The commands required are (assuming the file is already in disk, otherwise use R$J to load it).

```
A$J
ATL;FONnnn
```

where n is a digit from 0 to 7.

4. FONT CHANGING

There are 3 ways to request a new font:

(a) use a monitor command (in the same way as we use CAMERA/), the command is:

```
FONT/i,j,k,m
```

where i is a font number in octal which must be present, the other 3 parameters can be absent. Imagine a character is drawn within a box, j is the number of strokes required to go from the left lower corner of the box to the upper left corner, k is the number of strokes required to go from the left lower corner to the lower right corner. The default values of j and k are 14 and 10 respecively (decimal). If m is zero (default), the slope of stroke commands 16, 24, 30 and 36 are 10/14, -10/14, 10/14 and -10/14 respectively. If m is one, those will be 14/7, -14/7, 14/7 and -14/7.

example : FONT/100,16,8,1

(b) use a graphic order, which consists of at least 2 words

first word : 232000
second word : 600000+j

where j is a font number in octal, for example to select the font in example 1.4, we need

```
232000
600100
```

(c) use an escape-action, see Discussion paper 23

This section is included here to give the readers a preview on what the commands might be. A separate paper will be issued to discuss them in more detail as we get closer to release 4. It is expected that (a) will be the most popular method.

5. TESTING NEW CHARACTER DEFINITIONS

It is appreciated that new character definitions needed to be tested before they can be finally incorporated into a new font. The best way to do it is to simulate the stroke definitions (as it is done in DRIVER for large characters) by drawing vectors according to the information below:

length of one stroke (v) = character height/number of strokes

(default value for number-of-strokes is 14 (decimal))
(or it can be set by parameter j in command FONT)

COMMANDS	CHANGE IN X	CHANGE IN Y
14	v	0
15	v	v
16	v*10/14 or v	v or v*7/14
17	v*7/14	v
20	v*5/14	v
21	0	v
22	-v*5/14	v
23	-v*7/14	v

24	-v*10/14 or -v	v or v*7/14
25	-v	v
26	-v	0
27	-v	-v
30	-v*10/14 or -v	-v or -v*7/14
31	-v*7/14	-v
32	-v*5/14	-v
33	0	-v
34	v*5/14	-v
35	v*7/14	-v
36	v*10/14 or v	-v or -v*7/14
37	v	-v

once again there are two options for commands 16, 24, 30 and 36 as explained in previous sections.

6. CONCLUSION

Simplicity and security are the two main themes of our proposal here. Simplicity is achieved by hiding away all the hard work in the transferring program and, of cource, DRIVER. As for security, knowing that the FR80 is primitive in this respect, we have carefully included the FR80 operators in our scheme. The transferring program is also designed to perform all the necessary checkings as one would normally expect.