Honeywell

LEVEL 64
GCOS
UNIT RECORD DEVICES USER GUIDE

SERIES 60 (LEVEL 64)

UNIT RECORD DEVICES USER GUIDE

SUBJECT

Description of the Utility Program Used on the Level 64 to Initialize the Unit Record Devices

SOFTWARE SUPPORTED

Level 64 GCOS Software Release 0300

ORDER NUMBER

AQ59, Rev. 0

July 1977

Honeywell

PREFACE

This manual provides a detailed description of the utility program which is used on the Level 64 to initialize the Unit Record Devices.

Each device is treated under a different section. Appendix A covers the error messages and Appendix B the normal and abnormal return codes.

Each section of this document is structured according to the heading hierarchy shown below. Each heading indicates the relative level of the text that follows it.

Level	
1 (highest)	ALL CAPITAL LETTERS, UNDERLINED
2	Initial Capital Letters, Underlined
3	ALL CAPITAL LETTERS, NOT UNDERLINED
4	Initial Capital Letters, Not Underlined
5 (lowest)	ALL CAPITAL LETTERS FOLLOWED BY COLON: Text begins on the same line

A list of the Level 64 document set follows. Many of the manuals may be referenced in this text.

 File No.: 1N13

LEVEL 64 SOFTWARE DOCUMENT SET

Order Number	Abbreviated Text Reference	Full Title
AQ02	Series 100 Program Mode Operator Guide	Series 60 (Level 64) Series 100 Program Mode Operator Guide
AQ09	System Management Guide	Series 60 (Level 64) System Management Guide
AQ10	JCL Reference Manual	Series 60 (Level 64) Job Control Language Reference Manual
AQ13	System Operation Operator Guide	Series 60 (Level 64) System Operation Operator Guide
AQ14	System Operation Console Messages	Series 60 (Level 64) System Operation Console Messages
AQ20	Data Management Utilities	Series 60 (Level 64) Data Management Utilities
AQ21	200 Program Mode User Guide	Series 60 (Level 64) Series 200/2000 Program Mode User Guide
AQ26	Series 100 File Translator	Series 60 (Level 64) Series 100 File Translator
AQ27	Series 200/2000 File Translator	Series 60 (Level 64) Series 200/2000 File Translator
AQ28	Library Management	Series 60 (Level 64) Library Management
AQ52	Program Checkout Facility	Series 60 (Level 64) Program Checkout Facility
AQ53	Communications Processing Facility	Series 60 (Level 64) Communications Processing Facility
AQ55	TDS/64 Standard Site Manual	Series 60 (Level 64) TDS/64 Standard Processor Site Manual
AQ57	TDS/64 Standard Programmer Reference Manual	Series 60 (Level 64) TDS/64 Standard Processor Programmer Reference Manual
AQ63	COBOL User Guide	Series 60 (Level 64) COBOL User Guide
AQ64	COBOL Reference Manual	Series 60 (Level 64) COBOL Language Reference Manual
AQ65	FORTRAN Reference Manual	Series 60 (Level 64) FORTRAN Language Reference Manual
AQ66	FORTRAN User Guide	Series 60 (Level 64) FORTRAN User Guide
AQ67	FORTRAN Math Library	Series 60 (Level 64) FORTRAN Mathematical Library
AQ68	RPG Reference Manual	Series 60 (Level 64) RPG Language Reference Manual
AQ69	RPG User Guide	Series 60 (Level 64) RPG User Guide
AQ70	Series 100 COBOL Translator	Series 60 (Level 64) Series 100 COBOL Translator

LEVEL 64 SOFTWARE DOCUMENT SET

Order Number	Abbreviated Text Reference	Full Title
AQ72	Series 200/2000 COBOL to Level 64 COBOL Translator	Series 60 (Level 64) Series 200/2000 COBOL to Level 64 COBOL Translator
AQ82	BFAS User Guide	Series 60 (Level 64) BFAS User Guide
AQ84	UFAS User Guide	Series 60 (Level 64) UFAS User Guide
AQ85	Sort/Merge Manual	Series 60 (Level 64) Sort/Merge Manual

TABLE OF CONTENTS

TABLES

SECTION I

INTRODUCTION

This manual covers the use of the utility program URINIT (Unit Record INITIalization) on the Level 64. The Operating System (GCOS 64) allows the Unit Record Devices (URDs) to be processed in one of two different ways:

1. Direct Utilization

2. Indirectly, i.e. the program ignores the specific URD Interfaces and uses a spooling facility. This is achieved:

   - In input via the standard or permanent sysin

   - In output via the standard or permanent sysout.

URINIT is a utility program which allows the user to store in a system file all the parameters which are needed by the Unit Record Drivers to initialize the Unit Record Controller. The name of this system file is SYS.URCINIT, and its uses are described in the following section. The records stored in SYS.URCINIT are used by the following devices:

- Line printer

- Card Reader

- Card Punch

(each URD is dealt with in a different section)

Thus, various punching and reading codes may be defined for the Card Readers and Punches; and, on the Printer, different character sets (requiring a change of print belt) and paper-handling mechanisms may be used. It is the responsibility of the user to define in his JCL the format that is required for each.

The System then takes the user's format option and checks whether it is defined in the SYS.URCINIT System file: if so, input/output proceeds using the requested option.

Certain of these format options are pre-defined in the SYS.URCINIT file when the system is delivered:

1. For printers:

   a. The character set which corresponds to the Print belts delivered to the installation.

   b. The physical characteristics of the standard stationery: page depth, margin size etc.

   c. Vertical Format Unit (VFU): which defines the position of the channels used for paper-throwing.

   d. Print Test routine: use of the Print Test button on the printer causes the printing of one line just beneath the paper fold thereby allowing the operator a visual check for the paper alignment.

2. For all Card Readers and Punches: Translation tables for all standard punching and reading codes (e.g., for translating cards punched in H36 code into EBCDIC internal format).

In addition to these standard options supplied with the system, users may define their own format options:

- Special stationery definitions
- Non-standard VFUs
- Special character sets corresponding to special belts
- Special Print Test routines

If any such options are required by a user, it is his responsibility to submit the necessary data to the operations staff who may then include it in the SYS.URCINIT file by using the $URINIT utility. This utility may also be used to modify or delete information in the file.

SECTION II

URINIT: FUNCTIONALITIES

The utility URINIT allowes the user to:

- Create a record in the library (according to the type of initialization)
- Delete an existing record
- Update a record
- Rename a record
- List the contents of the library

The allocated space for SYS.URCINIT is 2 cylinders: the delivered contents represents 8% of the file leaving the remaining space available to the user.

The command cards which are recognized by URINIT are listed below: the cards are always punched by starting in column 1 and the slash sign (/) must not be followed by a blank. The use of a command code enables the reading of data cards in a specific mode: the data card deck ends when a new command card is read.

```
/CREATE - to create a new record
/UPDATE - to modify an existing record
/RENAME - to rename an existing record
/DELETE - to delete an existing record
/LIST   - to list the complete file
/END    - to end the update.
```

CREATE

Two records are necessary to create a new item in the file. The first contains /CREATE in the first seven columns and the second contains the data appropriate to the new item. A new directory entry is written at the end of the directory.

UPDATE

Two records are necessary to modify an existing item in the file. The first contains /UPDATE in the first seven columns and the second is identical with the second card of CREATE. Note that when updating an item, it is necessary to include all the unchanged data as well as the modified data.

RENAME

Two records are necessary to rename as existing item in the file. The first contains /RENAME in the first seven columns and the second is arranged as follows:

Card Columns	Data
1-8	original name of item
9	type of initialization
10-17	new name of item

Note: For column 9, see 'Catalogued Items and Contents' in this Section.

DELETE

Two records are necessary to delete an existing item from the catalog. The first contains /DELETE in the first seven columns and the second is arranged as follows:

Card Columns	Data
1-8	name of item
9	type of initialization

Any record which points to a deleted record is not modified.

Note: For column 9, see 'Catalogued Items and Contents' in this Section.

LIST

The complete file may be listed by means of a single record containing /LIST in the first five columns. Each item in the file is

listed in the following format:

INIT_NAME: name of item.

TYPE : type (see below).

LINK-NAME: name of associated item or NONE.

LENGTH : length of item, as input. (in decimal)

DATA : data

The first four characters of "data" are the length in hexadecimal, and may be ignored. The rest of "data" is as input for a paper form or VFU.

The type of the item is one of the following:

C for character set

F for form

V for VFU image

B for belt

P for Print Test Lines

K for initializations or translation tables

END

The last record for updating the file is an end record. This record contains /END in the first four columns.

DATA RECORDS

Formats

Data records have the following format:

Cols 1 - 8 : Name of this initialization entry

Col 9 : Type of initialization : See 'Catalogued Items and Contents' below

Cols 10 - 17 : Name of the next initialization record in the link structure. In rename mode this is the new name given

to the entry named in columns 1-8

Cols 18 - 20 : Length in decimal of the data part of this entry - this is the number of bytes

Cols 21 - 30 : Data part in hexadecimal

If there is too much data to go onto one card, punch the continuation character * (asterisk) at the end of the information on the card and continue. The data string is continued from column 1 on the next card. There is no limit to the number of cards which constitute one record. The last card must be terminated by at least one blank column.

CATALOGUED ITEMS AND CONTENTS

There are 6 types of initialization entry:

1. The character set descriptions (translation tables for card readers and card punches)
2. The form descriptions (printers)
3. The printing test lines descriptions (printers)
4. The VFU image descriptions (printers)
5. The belt segment descriptions or drum images (printers)
6. The initialization records defining the character set codes (printer)

A letter specifies the type of item on the data card:

C for character set

F for form

P for Print Test Lines

V for VFU image

B for belt

K for initialization and translation tables for printer

INITIALIZATION RELATIONSHIPS

There may exist several links between the initialization records: these links are provided by name. Each record in a chain contains in its header the name of the next logical initialization. The last record in a chain contains hexadecimal FF in its link name field.

The logical relationship between the records is defined by the user when creating or updating the library; some relationships are mandatory.

The following diagram shows the relationships between the records.

Link	Type	Content
None	Character set	Translation table (256 bytes)
(0) ↓	Form	Form height, full forms, head of form margin, printing density
↓ (0)	VFU	Vertical Format Unit tape image
↓ ↺ (0)	Printing test	Printing test line image
(r) ↓	Printer initialization	Code folding translation table or empty
↓ ↺ (0)	Printer Belt	Printer belt segment image or drum image

NOTE: (0) = Optional link
(r) = required link

This diagram shows that:

- A VFU may be associated with a form description
- A printer test sequence may be associated with a VFU
- One printing test line can point to another one if several lines compose the printing test sequence
- A printer initialization record always points to a belt segment image or a drum image
- A belt segment image can point to an alternative belt segment image

The updating of the file is a single step job and requires the following JCL statements:

```
$JOB jobid, USER = username, ACCOUNT = accountname;
$URINIT COMFILE = *filename;
$INPUT filename;
card deck for updating file
$ENDINPUT;
$ENDJOB;
```

Where:

jobid is the identification of the job,

username is the name of the user,

accountname is the name of the account,

filename is the name of the INPUT enclosure in DATA format

SECTION III

LINE PRINTER

Level 64 printers are impact printers, connected to GCOS 64 through a Unit Record Processor (URP). They convert digital codes into printed characters; printing is performed by means of hammers and a belt/drum mechanism.

The printer-driver supports the following printers:

- Belt printers

  PRU 1200/1600

- Drum printers

  PRU 0600/0800

DEFINITIONS

BELT	Endless train supporting 480 characters.
CHANNEL	See VFU CHANNEL
CHARACTER SET	Group of different characters printed by a particular group of belts or drums
DENSITY	Vertical print density (6 or 8 lines per inch
DRUM	Rotating drum (PRU 0600/0800 only). Each 'slice' supports all the different characters specific to the drum.
FF1	Full Form 1

FF2	Full Form 2
FORM	Paper page.
FORM NUMBER	Name of a catalogued set of form parameters
FORMHT	Form height
FOOTING	Line printed at each page bottom (SSF only)
HEADER	Characters at the beginning of a record containing information about the data part and/or slew information: the header length may be 0 (SARF), 1 (ASA) or 8 (SSF/ASA),
HEADING	Line printed at each page beginning (SSF only)
HOF	Head of Form level
HPR	Printer driver
LINE TRUNCATION	Partial printing of too large a line: the end characters are lost
LINE FOLDING	Special printing of too long a line: the end characters are printed on consecutive lines (should be avoided)
MARGIN	Number of blank columns skipped at the left of a line
MEDIA	Designates both the belt/drum and the paper forms
NON PRINTABLE CHARACTER	Hexadecimal combination which is undefined
NOSLEW	Slew suppression
REFERENCE LEVEL	Paper position when the paper fold coincides with a red mark located on the tractors.
SEGMENT	Repetitive series of characters on a belt
SLEW	Paper movement
SPACE LINE	Minimum space between two printed lines
STOP LEVEL	VFU channel entry/hole
SUBSTITUTION CHARACTER	Character to be printed instead of invalid combinations (i.e. which are not present in the belt/drum image)
TITLE	see HEADING

TRACTORS	Paper movement mechanism with drive sprockets
VFORM	see VFU
VFU	Vertical Format Unit : set of channels contain predefined stop levels, allowing automatic page editing.
VOLUME	see MEDIA.

Table 3-01. Line Printer Characteristics

	Speed (lines per minute)	Number of Hammers	Horizontal Printing Density (characters per inch)	Form Height (inch)		Form Width (inch)	
				Min	Max	Min	Max
PRU 1200/1600	1200/1600	136/160	10	4	24	4	22
PRU 0600/0800	600/800	120/132	10	3	17	4.75	18.94

Note: All the printers accept 6 or 8 lines per inch.

Device Naming and Allocating

A printer is a device on which a volume must be mounted at the beginning of a work session. There is one type and one attribute identifying the number of hammers. This attribute is the basis on which Device Management allocates a printer.

Device Type	Attribute	
	naming	meaning
PR	H120	120 print positions
	H132	132 " "
	H136	136 " "
	H160	160 " "

Device Management will allocate a printer having at least as many print positions as specified.

Example: DEVCLASS = PR/H120

A printer having 120 or more print positions will be allocated.

URINIT FILE

The file contains information concerning the following characteristics of the printer:

- Character set and belt - the group of different characters to be printed.
- Paper form - the physical characteristics of the paper.
- Vertical Format Unit (VFU) tape image - this gives the same functional facilities as the standard eight channel physical VFU tape.
- Print Test image - the data that is printed on the printer when the Print Test button is pressed.

CHARACTER SET

Seven character sets are pre-stored in the file. These are the character sets I1, I2, I3, I7, G1, G3 and G7. G1, G3 and G7 are the character sets that should be used when running the 100 Program Mode Software package; and I1 is the standard character set for the PRU 1200/1600 and the PRU 0600/0800 (63-character drum)

I2 is used with the PRU 1200/1600 to print lower cases (E600 belt)
I3 is used with the PRU 0600/0800 drum with 96 characters.

BELT

Associated with each character set is a belt. The following characteristics must be stored for each belt:

Belt code - Name of the belt.

Space code - The character to be used for a space.

Substitution character - The character to be substituted for any invalid character.

Belt image - Gives the correspondence between the binary encoding of a character and its offset on the physical belt segment.

Link name - Specifies an alternative belt; up to seven belts can be linked together. This parameter is optional and can be omitted if there is no alternative belt.

Certain belt images are pre-stored in the file. Associated with character set I1, the belt images E700, E500, E501, E502, and E600 are stored for the PRU1200/1600, and the image EH112-63 is stored for the PRU0600/0800. Associated with the character set G1, the belt images G700, G500, G501, G502, and G600 are stored for the PRU1200/1600, and the image GH112-63 is stored for the PRU0600/0800.

For the belts associated with character set I1, the substitution character is @ (7C hexadecimal) and the space character is 40 (hexadecimal). For belts associated with the character set G1, the substitution character is Δ .

PAPER FORM

A VFU may be associated with each paper form. The linked VFU is automatically referenced each time a reference is made to the appropriate paper form, unless otherwise specified in $OUTPUT(VFORM=). Care must be taken, when updating the file, to maintain this link between a form and its VFU.

The following characteristics must be stored for each paper form:

- Form height - the size in number of lines of the paper form,
- Margin - the number of columns to be skipped at the left of the paper form,
- Head of form - line number from the upper paper form fold, used for jumping to head of page,

- Full form level - line number from the upper paper form fold, used for indicating a full page,
- Printing density - the size of the line; it may be 1/6 inch or 1/8inch,
- Associated VFU - the name of the associated VFU.

Four paper forms are pre-stored in the file. These are the forms GSTD, GALL, 0000 and 001. GSTD is a paper form that can be used when running the 100 Program Mode software package, and 0000 is a standard paper form. The four paper forms are described below in the EXAMPLE.

The identification of the required paper form (and associated VFU) may be referenced by the OCL command NV or by JCL statements. For example, the command:

NV PR01 GSTD

specifies the paper form with the name GSTD.

The Operating System may issue the following message, at the appropriate time:

DV11 * PR01 MOUNT GSTD

After ensuring the correct form is mounted and that it is synchronized, the operator presses the printer button START.

The data is arranged on the card as follows:

Card Columns	Data
1-4	name of paper form
5-8	blank
9	F
10-17	name of associated VFU
18-20	011
21-24	form height (in hexadecimal)
25-28	margin (in hexadecimal)
29-32	head of form (in hexadecimal)
33-36	0000
37-40	full form level (in hexadecimal)
41-42	printing density as 06 for 1/6 inch or 08 for 1/8 inch

Note that the margin should be specified as 0000 for any form that is to be used by the 100 Program Mode software.

VFU

A VFU may be associated with each paper. The linked VFU is automatically retrieved each time a reference is made to the appropriate paper form.

The data for each VFU is arranged in the following manner:

- The channel number - 01 to 12
- The line number from the upper paper form fold to stop level: repeated for each stop level on a particular channel number, specifying the stop levels in increasing order, given in hexadecimal code.
- The characters FF to separate a channel from the following channel.

Example: Channel 1 with stop levels at 10 and 15,
Channel 3 with stop level at 20,
Channel 5 with stop level at 25,
Channel 7 with stop level at 05,
Channel 8 with stop level at 40.

The corresponding VFU would be:

010A0FFF0314FF0519FF0705FF0828

Notes:
1. The last channel in a VFU is not terminated by FF.
2. The first stop level of channel 7 is head of form. This must always be present for 100 Program Mode.
3. The only stop level of channel 8 is full form level. This must always be present for 100 Program Mode.
4. There is a limit on the number of stop levels for a VFU. This is given by the following formula:

limit = 44 - 2 * number of channels

For example, if only 1 channel is used, it may have 42 stop levels. Whereas, if 4 channels are used, there may only be 36 stop levels between the 4 channels. The data is arranged on the card as follows:

Card Columns	Data
1-8	name of VFU
9	V
10-17	name of associated Print Test or blank
18-20	image length (in decimal) - see below
21-80	VFU layout (in hexadecimal)

The image length is obtained by counting the number of characters in the VFU image and dividing by two.

NOTE: When the amount of data is too great for one card, punch the character asterisk (*) after a channel number, stop level or FF and continue on the following card starting in the first column. There is no limit to the number of cards which may constitute an item. The last card must be terminated by at least one blank column.

PRINT TEST

The standard print test is the printing of one line of E's just beneath the paper fold.

The test is automatically retrieved each time the PRINT TEST button is pressed on the printer. If no print test is specified for a particular VFU, then the default print test H_PRTEST is retrieved. Any other print test must be stored in the SYS.URCINIT file. For each print test, it is possible to give a description of the paper movement before or after printing and the line (or lines) to be printed. The information for each line is given as follows:

- A header (in hexadecimal)
- The characters to print (in hexadecimal).

The header is composed of four bytes:

0000 P1 P2 0000

the bytes P1 and P2 describe the paper movements before or after printing.

P1 is itself divided into two characters such that :

P1 = xy

x may take the values 0 or 8 and the values of y are given in the following table, together with associated values of P2. A value of 0 for x means that the movement takes place before printing and a

value of 8 means that the movement takes place after printing.

Y		P2
0	movement of n lines. n is given by P2 in binary.	00 to FF (or form height)
1	jump to a reference level.	00
2	jump to a line defined by P2, in binary.	00 to FF (or form height)
3	jump to Channel n, given by P2 in binary	01 to 0C
4	As 00.	00 FF (or form height)

Examples: P1P2 = 0001 means a movement of 1 line before the test line is printed.

P1P2 = 8204 means jump to line 4 after the test line has been printed.

The data is arranged on the card as follows:

Card Columns	Data
1 - 8	Name of Print Test
9	P
10-17	name of following line or space (if last line)
18-20	length of description (in decimal) see below.
21-80	description of print line in hexadecimal

The length of description is obtained by counting the number of characters in the description and dividing by two.

NOTE: When the amount of data is too great for one card, punch the character * (asterisk) and continue on the following card starting in the first column. There is no limit to the number of cards which maybe used. The last card must be terminated by at least one blank column.

BELT

The data is arranged on the card as follows:

Card Columns	Data
1-8	name of belt
9	B
10-17	name of alternative belt (optional)
18-20	length of record (in decimal)
21-24	belt code (hexadecimal)
25-26	space code (hexadecimal)
27-28	substitution character (hexadecimal)
29-80	belt image

For ease of verification, it is recommended that 16 codes of the belt image be encoded on each card. Thus for a 240 character belt image, there would be 15 cards specifying the belt image and one initial card.

Note: When the amount of data is too great for one card, punch the character * (asterisk) and continue on the following card starting in the first column. There is no limit to the number of cards which can be used. The last card must be terminated by at least one blank column. Instead of a belt image, a drum image can be stored (for PRU0600/0800 printers). In this case a belt code is replaced by a drum code; this code is only a convenient software means to select the drum image, it does not refer to a physical characteristic of the drum (which is the case for the belt codes against the belt). Its value is 0000 and only one drum image can be stored in a character set.

Examples

1. The pre-stored paper form GSTD and associated VFU (GVFUSTD) are described below:

   the paper form has the following characteristics:

   . Form height = 66 lines

   . Head of form = 5 lines

   . Full form level = 61 lines

. Printing density = 1/6 inch

and the VFU is defined by:

. Stop level 5 on channel 7, for head of form.

. Stop level 61 on channel 8, for full form level.

The card deck for creating this entry in the file is as follows:

```
/CREATE

GSTDFGVFUSTD 011004200000005 0000003D06

/CREATE

GVFU0050705FF083D

/END
```

Note that /END is used to terminate the data for updating the file.

2. The pre-stored paper form 0000 and associated VFU (H_DF_VFU) are described below.

The paper form has the following characteristics:

. Form Height = 66 lines

. Head of Form = 5 lines

. Full form level = 60 lines

. Printing Density = 1/6 inch

and the VFU is defined by:

. Stop level 5 on channel 1

. Stop level 10 on channel 2

. Stop level 40 on channel 3

The card deck for creating this entry in the file is as follows:

```
/CREATE

000FH_DF_VFU011004200000005 0000003C06

/CREATE

H_D0080105FF020AFF0328
```

/END

Remark: Paper 001 is the same as paper 0000 except for the printing density which is 1/8"

3. The pre-stored paper form GALL and associated VFU (GVFUGALL) are described below.

   The paper form is identical with GSTD, but with a VFU in which the channels 1 to 6 correspond to the following lines:

   16,22,28,34,40,46

   Note that the first stop on channel 7 must correspond to head of form and the first stop on channel 8 must correspond to full form level.

   The card deck for creating this entry in the file is as follows:

   /CREATE

   GALL FGALLVFU 011004200000005000003D06

   /CREATE

   GALLVFU V 0230110FF0216FF031CFF0422FF*

   0528FFF062FF0705FF083D

   /END

4. If the file were listed in a subsequent (or the same) run as the above, it would appear as in the table below, where only the forms from the above examples have been included.

   LIST REPORT

   INIT_NAME: GSTD

   TYPE : F

   LINK_NAME: GVFUSTD

   LENGTH : 011

   DATA 000B042 00000005 0000003D 06

   INIT_NAME : GVFUSTD

   TYPE : V

   LINK_NAME : NONE

   LENGTH : 005

```
DATA 00050705 FF083D

INIT_NAME: 0000

TYPE: F

LINK_NAME : H_DF_VFU

LENGTH : 011

DATA : 000B0042 00000005 0000003C 06

UNIT_NAME : H_DF_VFU

TYPE : V

LINK_NAME : NONE

LENGTH : 008

DATA : 00080105 FF020AFF 0328

INIT_NAME : GALL

TYPE : F

LING_NAME : GALL VFU

LENGTH : 011

DATA 00B0042 00000005 0000003D 06

INIT_NAME : GALLVFU

TYPE : V

LINK_NAME : NONE

LENGTH : 023

DATA 00170110 FF0216FF 031CFF04 22FF0528

FF062EFF 0705FF08 3D

Example of a belt image:

1          9          18

/CREATE

11         B          244XXXX4000*

7EF5F6F7   etc.
```

This specifies a belt image of 244 (i.e., 240 plus 4 for belt code, space code, substitution character), with a belt code

XXXX, a space code of 40 (hexadecimal), a substitution character of 00 (hexadecimal), and a belt image which starts 7EF5F6F7.

HOW TO USE THE PRINTER IN DIRECT MODE

At the program level, a file description for the printer has to be defined. At the JCL level, a $ASSIGN card has to be given:

```
$ASSIGN internal file name, DEVCLASS = devclass, MEDIA = name;
```

The system will ask the operator to mount the appropriate form at the initiation of the step by issuing the message:

```
DV * PRO1 MOUNT name FOR Xnn
```

SECTION IV

CARD READER

The Card Reader Driver can support the devices shown below in Table 4-01.

Note that:

- Mark Sense reading and 51-column card reading capabilities have to be asked for at installation time.
- Although the CRU 0600 and CRU 1050 can read both 51-and 80-column cards, the two series of cards cannot both be read at the same time.

Table 4-01. Card Reader Devices

DEVICE	COLUMN CARD		CARDS/MIN	MARK SENSE READING
	80	51		
CRU 0501	yes		500	yes
CRU 0600	yes	yes	600	yes
CRU 1050	yes	yes	1050	yes

The initialization record for the card reader is the character set description. This description is signified by the letter 'C'.

The character set can be defined for the card reader. The character set record consists of:

- A text code
- A translation table

These are stored in one CREATE session: there is no link name.

The text code field indicates the field in which data is transferred to the user.

The value of the text code is:

04 for EBCDIC
05 for ASCII
1E for G100
1F for G100 intermediate

The translation tables that are used are pre-stored in the SYS.URCINIT file. These 9 files are:

- H14G for translation from H14 code to Series 100 internal code,
- H36G for translation from H36 code to Series 100 internal code.
- H36GI for translation from H36 code to Series 100 intermediate code,
- H36E for translation from H36 code to EBCDIC code.
- H14E for translation from H14 code to EBCDIC code.
- H20E for translation from Series 200 Normal Mode to EBCDIC code.
- H20HE for translation from Series 200 Hollerith Mode or Special Mode to EBCDIC code.
- H2CE for translation from Series 200 COBOL Normal Mode to EBCDIC code.
- H2CHE for translation from Series 200 COBOL Hollerith Mode or Special Mode to EBCDIC code.

Definitions:

BLOCK — Amount of data read per card (A block is a physical record)

CARD IMAGE — The mapping of a card, showing which columns have to be transferred in the CPU Memory

CHARACTER SET — A group of different characters, each of them being defined by a punch code of 12 rows (the number of lines on a card)

COMMAND CARD — A File Mark card which is meaningful for the driver processing

FILEMARK — A special character which can be recognized by the device. It is always on the first column of a card

HCR — The handler of the Card Reader (or the card reader

driver)

MARK — A trace which should be made with a single vertical stroke of a pencil and placed inside the mark locators on the card

Data Forms

Two data forms are meaningful: SSF and SARF. However, the value DOF of this parameter is handled as SARF.

The SSF data format allows the following capabilities:

- Job protection
- Data code switching via the HCR command card, but it must be with the restriction of Whole Card reading

SARF data format allows:

- Card image processing

There is, however, no way to process job protection or data code switching. For Data Management the card reader format depends on the data format. In SARF all parameters are fixed at OPEN time and cannot change before CLOSE time. In SSF when the reading mode varies the record format is forced to UNDEFINED and the FD parameter is ignored.

Reading Options

The BINARY parameter defines the reading mode of the cards. Two modes are supported by the Card Reader:
Hollerith or Binary.
When the keyword BINARY is omitted, Hollerith reading mode is assumed.

Hollerith Reading Mode

There is one byte per card column. The 12 rows are packed into 8 bits following this rule:

Punched rows	bits		
	5	6	7
none	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

punched rows	12	11	0	8	9			
	0	1	2	3	4	5	6	7

The resulting code is called Hollerith intermediate code.

Note: No multipunch is allowed on columns 1 to 7 . This is the standard card code in GCOS (Hollerith 36) and the Standard internal code is EBCDIC.

Binary Reading Mode

There are two bytes per card column which are transferred to main memory. The effect of these two bytes on the column depends on the next parameter read.

When the Data Format is SSF, the reading mode can be dynamically switched from one reading mode to another by use of the HCR command card. These command cards are file mark cards which are meaningful to HCR and the descriptions, writing rule and meaning of these cards are given below.

Hol card

<File Mark> HOL <blanks>
:<spaces><char.set><spaces>

with

<File Mark> : (See Subsection)
<char.set> : (See Subsection)
<spaces> : (at least one blank card)

<blanks> : (blank cards only until the card-image card).

This indicates that the following cards are to be read in Hollerith mode.
The Translation Table to the one under which the table to be used has been catalogued. The default value for the table name is H36E.

BIN CARD

<file mark>Bin : <blanks>
:<spaces>NPL<spaces>
:<spaces>H200<spaces>

with

<file mark> : (See Subsection)
<spaces> : (one blank at least)
<blanks> : (only blanks until the end of the card image)

This command indicates that the following cards are to be read in binary (NPL or H200) mode.
Default value is NPL.

URTYPE (Unit Record Type)

This keyword is meaningful only when binary reading (BINARY keyword is present) is chosen.

Each column is represented by a two-byte pattern as follows:

Native

row number	*	*	12	11	0	1	2	3	*	*	4	5	6	7	8	9
bit address	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
byte address	A								A+1							

H200

row number	*	*	9	8	7	6	5	4	*	*	3	2	1	0	11	12
bit address	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
byte address	A								A+1							

For each row in the column the corresponding bit is set to zero if no hole or mark is present; the bit positions marked * are always set to zero.

CHARSET (Character Set)

When using the Hollerith reading mode a second translation normally follows the card reading phase. This translation is done by the firmware by using a 256 byte table. A Translation Table is referenced in a system file by its name, and the standard default value is H36E.

IMAGE (Card Image)

The card image declares which columns of the card are to be read. This functionality is specifically given for mark sense reading where only one column among two is significant. IMAGE is an 80-bit field with one bit per column. When a bit = 1, it means that the corresponding column is to be used. For the 51 column card reader only the 51 bits on the left are significant.

In SSF mode, all the bits of the card image have to be set to 1 to ensure JOB protection and correct HCR command card processing. If File Mark cards are included with a non-standard card image, only the first column of this card is sent to the user with the number of columns being equal to the number of bits set to 1 in the card image.

OFFSET (Hardware Recovery Option)

This parameter only refers to the following reading error: each column of a card is read twice by the hardware, the two results are different. This mainly occurs when the punched columns of the card are offset from left to right or vice-versa.

If a read error occurs, two kinds of recovery are possible:

Stop Option

- the device is stopped and the operator notified. The card is read again until either a read is successful (the card is passed to user with normal return code) or the device is declared out of order (the user's program receives an abnormal return code)

Offset Option

- the device is not stopped, the erroneously read card is offset in the stacker. The user's program receives the first record of the following card, with the normal return code.

When the keyword is not specifed the stop option is the default value.

MARK SENSE READING

Instead of being punched some columns (or all) may be marked. (This is a hardware facility). If holes and marks are made in the same column, the mark information is lost.

There must be at least one column without a mark between marked columns, but holes can be present beween marked columns.

File Mark Definitions

There are two definitions of the file mark, depending on the reading mode.

Hollerith reading Mode: The file mark value is 53 in internal code. It corresponds to the $ sign in EBCDIC code. This value has to be found in the first column of a card to be considered as a file mark.

Binary Reading Mode : The 53 value found in the first column of a card is considered as a File Mark only if the previous card was a blank card.

File Mark Detection

If the File Mark keyword is present the File Mark is notified to the user by a special return code. If this keyword is absent the File Marks are not recognized and data is sent without special notification.

On Binary Mode Reading, the sequence:

1. Blank Card
2. <File Mark Card>

will be received as:

1. Only one card (with return code) if FM is present
2. Two cards (without return code) if FM is absent.

Detection of Special Cards

Two special cards exist:

JOB card
EOS card

These cards are File Mark cards in binary mode and there is no need for a blank card to permit File Mark detection.

USING THE CARD READER IN DIRECT MODE

The direct use of the card reader is similar to the processing of a non-standard file. At the program level, a file description for the card reader has to be defined. At the JCL level, a $ASSIGN command has to be given, eg

```
$ASSIGN internal file name, DEVCLASS = CD/R/C80, MEDIA = name;
```

The cards to be read are not part of the job stream; they form a separate deck. The system will ask the operator to mount this deck at the initiation of the corresponding step, by issuing the message.

```
DV11 * CD01 MOUNT name FOR Xnn
```

The last two cards of the deck must be a $EOS card (which is the end of file mark) and a blank card. Note that the processing of the deck is the same as that of a non-standard file, the end of which is reached when reaching the $EOS card, and that more than one blank card may be added behind the $EOS card.

SECTION V

CARD PUNCH

The URINIT file contains the translation tables that are used for card code translation. There are three translation tabels pre-stored in the file:

- GH14 for translation from Series 100 internal code to H14 code.
- GH36 for translation from Series 100 internal code to H36 code.
- GIH36 for tanslation from Series 100 intermediate code to H36 code.

80-column cards can be punched with GCOS, and the following devices are supported:

PCU 0120	Card Punch Punches between 100 and 400 CPM, depending on the number of columns to be punched
CCU 0400	Card Reader - Punch punches 100 CPM if all 80 columns are punched

GCOS allows the following to be punched:

- Members of Source Libraries
- Compile Units
- Load Modules
- Data

To Punch	Use
Source Compile Unit Load Module	Library Maintenance (LIBMAINT)
Data	Output Writer or Card Punch in Direct Mode

Standard or Permanent Sysouts

When Standard or Permanent Sysouts are used to punch cards, a $OUTPUT Job Command with OUTDEV = CD/P/C80 and OUTMED = name has to be provided. (The Card Punch Device Class is CD/P/C80)

Example of Punching Data with a Standard Sysout

. JCL

```
$JOB ..... ;
$STEP ..... ;
$ASSIGN PUNCH, SYSOUT;
$OUTPUT PUNCH, OUTDEV = CD/P/C80, OUTMED = CARD;
$ENDSTEP;
$ENDJOB;
```

. Cobol Program

```
SELECT OUTFILE
ASSIGN TO PUNCH - SYSOUT
[ORGANIZATION IS LEVEL 64 SEQUENTIAL]
RECORD CONTAINS 80 CHARACTERS
```

Example of Punching Data with a Permanent Sysout

. JCL

```
$JOB ..... ;
$STEP ..... ;
$ASSIGN PUNCH, CARDFILE, FILESTAT = UNCAT,
```

```
DEVCLASS = MT/T9/D1600, MEDIA = TAPE03;

$OUTPUT PUNCH, OUTDEV = CD/P/C80, OUTMED = CARD;

$ENDSTEP;

$ENDJOB;
```

. Cobol Program

```
SELECT OUTFILE

ASSIGN TO PUNCH_SYSOUT

[ORGNAIZATION IS LEVEL 64 SEQUENTIAL]

RECORD CONTAINS 80 CHARACTERS.
```

Note: Several sysouts can be punched in the same step and a sysout can be printed and punched

OUTMED - Before the Output Writer starts punching the file, the operator will be asked to mount a pack of blank cards in the punch, the name of which is specified by OUTMED.

RECORD SIZE - If the records written on the sysout file (either permanent or standard) are smaller than 80 characters the cards are fitted on the right with blanks. If an attempt is made to write more than 80 characters, the error will be detected at output time by the Output Writer. The Output Writer stops punching cards at the first record greater than 80 characters, and the operator is made aware of this by the message:

OUT_01 WRONG RETURN CODE.....

So ensure that 80 is the maximum number of characters if the file is to be punched.

MARGIN - The first character of the record is punched in the first column of the card. If it is to be punched in column number i it is specified by $OUTPUT MARGIN = i - 1

Example: Standard sysout with a record 50 characters long to be punched in columns 10 to 60

```
$STEP      ;
$ASSIGN PUNCH, SYSOUT;
$OUTPUT PUNCH, OUTDEV = CD/P/C80,
 OUTMED = CARD, MARGIN = 9;

$ENDSTEP;
```

The record size has to verify:

MARGIN VALUE + RECORD SIZE < 80

Otherwise the same error occurs as trying to write a record which is too long without MARGIN (See RECORD SIZE)
The MARGIN facility can only be used when the Output Writer is used, i.e. when a $OUTPUT at job or step level is specified.

MEDIA CONVERSION

BFAS or UFAS sequential file output can be punched by using the $OUTPUT Command at job level. The parameters to be specified are:

OUTDEV = CD/P/C80 and OUTMED: _

Example 1:

```
$JOB ..... ;
$OUTPUT = CARD, DEVCLASS = MS/M400, MEDIA =
C018, OUTDEV = CD/P/C80, OUTMED = DECK;
$ENDJOB;
```

Example 2: How to punch permanent file in SARF

```
$JOB..... ;
$OUTPUT CARD, DEVCLASS = MS/M400, MEDIA = C018,
              OUTDEV = CD/P/C80, OUTMED = DECK,
              DATAFORM = SARF;
$ENDJOB;
```

CARD PUNCH IN DIRECT

When the Card Punch is in direct mode, a file description for the Card Punch has to be written in the program along with a $ASSIGN job card. The Card Punch device class has to be specified (DEVCLASS = CD/P/C80) and the blank card deck has to be mounted (MEDIA = name) The system will request the mounting of the named card deck at the initiation of the corresponding step.

Example:

Cobol program PROGA to punch a card deck OUTDECK.

JCL

```
$JOB..... ;
$STEP PROGA..... ;
$ASSIGN CARDFILE, DEVCLASS = CD/P/C80,
 MEDIA = OUTDECK;
$ENDSTEP;
$ENDJOB;
```

Cobol program

```
SELECT CARD
ASSIGN TO CARDFILE[-CARD-PUNCH]
[ORGANIZATION IS LEVEL 64 SEQUENTIAL]
FD CARD
BLOCK CONTAINS 1 RECORD
RECORD CONTAINS 80 CHARACTERS
LABEL RECORD IS OMITTED
OPEN OUTMED CARD
      .
      .
      .
WRITE name
      .
      .
      .
      .
CLOSE CARD
```

At PROGA step initiation the operator will receive a mounting request [MESSAGE DV11]

Remarks

$ASSIGN - The only parameters taken into account by the system are:

- The internal file name
- The device class

. The media name

. The Pool/No Pool option.

Any other parameters specified will be ignored.

COBOL PROGRAM

FILE DESCRIPTION - The Clause BLOCK CONTAINS is not significant: it will always be one card per buffer. If a variable lengh record is specified in the program it will be ignored. The system will work as if a fixed length record was specified.

OPEN - The specified programming mode has to be Output. If another processign Mode is specified the program will abort at OPEN time with the abnormal return code=WRONG PMD (1C05)

PUNCHED DECK - No banner is punched. The first card punched corresponds to the first record of the file.

CARD PUNCH INCIDENTS

If an incident occurs whilst punching a card deck:

. The operator recovers it and the user is not aware of it (see Note) or

. The operator cannot recover it.

If an irrecoverable incident occurs while the Output Writer is punching a deck, the Output Writer stops and it is the operators responsibility to HOLD or CANCEL the output. When a program is punching cards it is aborted at WRITE time with the abnormal return code MDNAV (0A03)

NOTE: Sometimes after an incident several cards can be repunched. These cards will be marked by the operator acording to installation instructions.

BINARY PUNCHING

Binary Code Definition

In binary Mode a punched column corresponds to two bytes in memory; therefore an 80-column punched card corresponds to a 160-byte area in memory.

Code Translation

The 12-bit Hollerith card code is translated into the 8-bit internal machine code in two steps:

Step 1

The Hollerith card code is translated into an 8-bit intermediate code

Step 2

The intermediate code is translated into an 8-bit internal machine code by means of a translation table stored in the SYS.URCINIT system file. Several standard translation tables are provided.

PUNCHING

Standard Sysout, Permanent Sysout and Media Conversion can be used to punch 80-column cards in Binary Mode. Binary Cards cannot be punched directly from a Cobol program. To punch Binary cards, the following parameter has to be mentioned in the $OUTPUT command:

MODE = BINARY (Hollerith is the default value)

The binary mode is chosen with the parameter URTYPE (NATIVE MODE is the default value)

STANDARD SYSOUT

To punch cards in NATIVE binary mode:

$JOB..... ;

```
$STEP..... ;

$ASSIGN PUNCH, OUTFILE, FILESTAT = UNCAT,
 DEVCLASS = MT/T9, MEDIA = U15;
$OUTPUT PUNCH, OUTDEV = CD/P/C80,
 OUTMED = CARD,
 MODE = BINARY;

$ENDSTEP;

$ENDJOB;
```

PERMANENT SYSOUT

To punch cards in H200 Binary mode:

```
$JOB..... ;

$STEP..... ;

$ASSIGN PUNCH, OUTFILE, FILESTAT = UNCAT,

 DEVCLASS = MT/T9, MEDIA = TU16;

$OUTPUT PUNCH, OUTDEV = CD/P/C80, OUTMED = CARD,
 MODE = BINARY, URTYPE = H200;

$ENDSTEP;

$ENDJOB;
```

To punch a permanent sysout file created in another job:

```
$JOB..... ;

$OUTPUT OUTFILE, DEVCLASS = MT/T9, MEDIA = TU16,
 OUTDEV = CD/P/C80, OUTMED = CARD,
 MODE = BINARY;

$ENDJOB
```

Permanent Sequential File

To punch a permanent file in SARF

```
$JOB..... ;

$OUTPUT OUTFILE, DEVCLASS = MT/T9, MEDIA = T016,
 OUTDEV = CD/P/C80, OUTMED = CARD,
```

```
 DATAFORM = SARF,
 MODE=BINARY;
$ENDJOB;
```

Remarks: The first two bits of each byte of the area in memory which are to be punched have to be equal to 0 or punching is stopped with the abnormal return code (i/o) FAIL.

If the record size is less than 160 bytes long (168 if SSF) the Output Writer stops punching at the first record longer than 160 (168 if SSF) and the operator is given the warning message OUT_01.

File Allocation:

PERMANENT SYSOUT: RECORD SIZE = 168
PERMANENT FILE:
SARF - RECORD SIZE = 160
SSF - RECORD SIZE = 168

Hollerith Punching: Code Translation

The standard GCOS punching mode is to punch EBCDIC memory data into H36 card code. GCOS can also punch EBCDIC into any hollerith card code by use of Translation Tables. These Tables are stored in the SYS.URCINIT file and they translate the 8-bit internal machine code into an intermediate Hollerith code, and this intermedate code is then translated into the 12-bit Hollerith card code.

Several standard Translation Tables are provided, and users can store permanent Translation Tables into the SYS.URCINIT file.

The Translation Tables can only be used when the punching is being done by the Output Writer. When a Cobol program uses the card punch directly the internal machine code is assumed to be EBCDIC and the card code H36.

When the punching mode is not standard (EBCDIC to H36) the name of the Translation Table to be used has to be given in the CHARSET parameter of $OUTPUT:

CHARSET = Translation Table name (< 8 characters)

Example: To punch cards in EH14

```
$JOB..... ;

$STEP..... ;
```

```
$ASSIGN PUNCH, SYS.OUT;

$OUTPUT PUNCH, OUTDEV = CD/P/C80,
 OUTMED = CARD, CHARSET = EH14;

$ENDSTEP;

$ENDJOB
```

EH14 is the name of the Standard Translation Table EBCDIC→GH14

If the Translation Table is not in the URINIT file the operator is informed with the message OU05; he can Hold or Cancel the Output.

SSF

A file created by a Cobol program is in SSF in the following cases:

- SYSOUT option specified in the Cobol ASSIGN Clause (Standard or Permanent SYSOUT files)
- PRINTER option specified in the Cobol ASSIGN clause
- WITH SSF specified in the file description in all the other cases, the file is in SARF

When the output file is SSF, at WRITE time the cobol program adds an 8-byte SSF header to the data area specified in the WRITE verb.

When cards are punched, the output records have to be in SSF when binary and hollerith cards are punched in the same output deck.

DOF (Device Oriented Format)

When data is sent to a Unit Record Device, both SARF and SSF records are transformed into DOF by the access method. To avoid this overhead, the records can be built directly in DOF. In this case, device independence is lost because the DOF is specific to each unit record type. A file in DOF for a card punch cannot be printed and vice versa.

The specific data format for the records of a DOF file destined for a card punch is:

```
L1  L2  MBZ  MBZ  MB2      DATA
0   1   2    3

        Record Size
```

L1 L2 - record length in binary (must be greater than 3)

Byte #2+3- must be equal to 0 in binary

Each record is composed of a data part preceded by a 4-byte DOF header.

The DOF header is part of the record and has to be included in the record size of the file when it is allocated.

USING THE CARD PUNCH IN DIRECT MODE

At the program level, a file description for the card punch has to be defined. At the JCL level, a $ASSIGN command card has to be given:

$ASSIGN internal file name, DEVCLASS = CD/P/C80, MEDIA = name;

The system will ask the operator to mount the deck of cards at the initiation of the corresponding step, by issuing the message:

DV11 * CD01 MOUNT name FOR Xnn

No banner is punched for the card deck; the first card punched corresponds to the first record of the file.

APPENDIX A

URINIT ERROR MESSAGES

CODE UNKNOWN, CARD BYPASSED

Cause: The code (col. 9) is invalid.

COMMAND CARD MISPLACED

Cause: A command card was read when a continuation of data was expected. The last command is ignored. This command is taken into account.

COMMAND CARD MISSING

Cause: The first card is not a command card, i.e. /CREATE, /UPDATE, /RENAME, /DELETE, /LIST or /END.

/END CARD MISSING

Cause: Filemark read before /END card.

END OF PROCESSING

Cause: A /END card has been read or end of file has been found when reading user input.

END OF THE CURRENT SESSION

Cause: A new command has been taken into account.

ENTRY ALREADY EXISTANT

Cause: In create mode the file contains a record with the name given for a new record. The command for the new entry is ignored.

ILLEGAL HEXA STRING INGNORED

Cause: Illegal character in hexadecimal string.

ILLEGAL LENGTH FIELD IGNORED

Cause: The length of the data is not equal to the length given in columns 18-20 of the command.

INIT NAME MISSING, CARD BYPASSED

Cause: Cols. 1 - 8 blank.

NEW NAME ALREADY EXISTANT

Cause: In rename mode, the file already contains a record with the new name given,

NEW NAME MISSING, CARD BYPASSED

Cause: In rename mode, cols. 10 - 17 are blank.

NO DATA ON THE CARD READER = ABORT

Cause: End of file is found at first card read. The job is terminated.

RECORD UNKNOWN

Cause: The name (cols. 1 - 8) is not in the library.

WRONG COMMAND CARD

Cause: Invalid command (cols. 2 - 7).

WRONG LENGTH, STATEMENT IGNORED

Cause: The length of the data is not equal to the length given in the command (cols. 18 - 20).

XXXX ERROR NUMBER = YY RETURN CODE VALUE = ZZZZ

Cause: Software error.

APPENDIX B

RETURN CODES

Normal Return Codes

Return Code Name	Explanation of Cause	Recommended Action
CARDSKIP	Only if offset option required. The preceding card read has not been sent to the user because of a misreading (badly punched...)	Depending of user program: continue or close file.
DATALIM	End of file was reached ($EOS card).	Close file.
DONE	Correct Execution May also mean a dummy file was assigned and primitive is without action.	Continue Processing
DUMMY	File was assigned as dummy and primitive is without action.	Continue Processing.
FILEMARK	User's file mark detected and recognition required	Continue Processing.
JOBCMD	$JOB detected with $JOB protection required and standard reading parameters (Hollerith with H36 to EBCDIC translation).	Continue Processing.

Abnormal Return Code (Error Detection)

- If an I/O error is detected, the user is notified through a return code, but, in every case it can continue its processing. Data Management does not look upon everything as an I/O error.

- If a logical error occurs, the user is notified through a return code. Depending on the error type, Data Management has a synchronization that promises some return action (e.g., if a wrong HCR command card is detected and job protection is required, the user can try to read the next job card to resynchronize it).

- If a system error occurs, Data Management looks at its processing on that error and the only thing to do is to close the file.