bcc	title	prefix/class-number.revision
	CONTROL INPUT/OUTPUT STREAMS	CIOSUC/W-31

checked	authors	approval date	revision date
checked	Larry Barnes	10/2/69	classification: Working Paper
approved	Larry L Barnes	distribution: Company Private	pages: 29

ABSTRACT and CONTENTS

This document is an incomplete version of the description of control input/output streams and the UCALLs for using them. It is being released at this point to finalize the design of the character set, etc. Insofar as the document is complete, it is intended to be accurate.

Control Input/Output Streams

Many programs have two forms of input/output: a low-rate control stream used for commands and error messages, and high-rate or bulk input/output. The normal situation is to use a teletype for the first variety and files for the second. In previous systems it has been necessary to resort to reprogramming or other clumsy solutions to convert the initially independent program from one which is driven by a console to one driven by a file or program. The control input/output component of the standard utility system is designed to drive a program from terminals, files, or sub-processes without changing the driven program. To achieve this desirable goal the driven program must use the UCALLs for control input/output.

There will be an object called a control input/output stream (CIOS) which a sub-process uses 'like' a teletype. The input half of a CIOS may be one of: a terminal, a file, a terminal and a file, or a specified sub-process entry point. The output half may be one of: a terminal, a terminal and a file, a file, a sub-process entry-point, or nothing. In the case of file input and file output, for example, characters read from the input file will be 'echoed' on the output file in exactly the same manner as for a terminal. In addition to the input/output specification a CIOS will have an access lock and a control lock for determining which sub-processes may use the CIOS or change access to it.

Each recursion level of the utility system (see CMP/W-6)

maintains a current CIOS. Thus when the utility is asked to process a command file, it creates a CIOS for this purpose. In order for the user to retain control, the escape interrupt always reconnects the utility to the main, terminal CIOS. If there is no terminal, the utility then makes the process dormant.

Character Set

In this section and appendix A we define the standard 8-bit character set used to store, and transmit symbolic information. The following properties of the internal set are considered desirable:

1) The graphic characters should start at zero and be contiguous in the set.
2) The 96 ASCII graphics should be the first internal graphics.
3) Provision must be made for representing other, device-dependent, graphic characters.
4) As explained in the section on canonical form, there are characters for activating standard functions. In addition there may be device-dependent non-standard functions.
4) For efficiency in storing symbolic files strings of blanks should be encoded.

These properties are embedded in the following character set.

Characters	Internal Codes
64 printing characters of the Model 35 teletype	0-77
32 additional printing characters of the Model 37 teletype	100-137
Non-standard printing characters (device-dependent)	140-177
Multiple blanks (200 plus the number of blanks)	200-277
Standard function characters (see canonical form)	300-337
Non-standard function characters (includes teletype control characters)	340-377

Canonical Form

The idea of a standard or canonical form for the representation of printed lines originally arose in connection with document formatting programs. For any line of text there may be several ways of typing the line. Given that a human being does not usually distinguish between these purely internal differences, no program should either. Further if all lines of text are in a standard form, processing of them is simplified. Thus text will normally be processed and stored in canonical form.

We now proceed to informally define the canonical form of lines. A formal syntactic definition is contained in Appendix B.

1. A line consists either of a new-line character (<NL>), a form feed followed by a new line (<FF> <NL>), or sequences of print positions separated by carriage motion followed by <NL>.

The canonical form has the property that the graphics are ordered as they appear on the line from left to right and top to bottom. Between printing positions appears the motion required to move from one position to the next. Overstruck graphics are stored in a standard form.

2. Carriage motion is either a sequence of spaces (<SP>) or optional spaces followed by vertical motion. The vertical motion is the number of half-line feed characters (<HLR> or <HLF>) required to move from the preceeding print position to the top of the next

position.

Note: If the maximum vertical range for a print position is more than one-half line above the main line (superscript) or below the main line (subscript), the extra half-line feeds appear in escaped form. If necessary, an up:feed is inserted to bring the last print position back to the main line position.

3. A print position consists of some non-zero number of character positions separated by a backspace character (<BS>) and one or two HLF characters.
4. A character position consists of a series of graphic formers separated by backspace characters.

The graphic formers are ordered in ascending order by their internal code value (ie the lowest value graphic is first). Note that all uses of backspace are covered by points 3 and 4.

5. A graphic former consists of a possibly zero-length setup sequence of function characters followed by a graphic character.
6. A setup sequence contains an optional color shift character (<RRS> or <BRS>) followed by any other function character in the order in which they were typed. In the absence of a color shift, a line is printed in black.

The complexity of the full definition of the canonical form is more apparent than real. The following simplified definition

of canonical form covers most of the cases arising in practice.

1. A line is either empty (<NL> only) or contains character positions separated by spaces.
2. A character position consists of a single graphic or occasional overstruck graphics.

Thus normally the canonical form differs little from raw input.

Input/Output Subroutines

As mentioned in the first section, CIOS input may come from a sub-process. The purpose of this feature is to provide sophisticated control and error recovery for a driven program by some driving program. The overall implementation of the feature is to buffer the driven program's output either in an internal ring-buffer or on a file.

The driven program operates in the normal manner. The driving program is normally called by the utility when the driven program requests input. When the driving program is called to generate input for the driven program, it may then receive any output previously generated by the driven program. If the internal output buffer is used, the driver will be called if the internal buffer fills up. We now demonstrate the actions of the utility, the driver and driven programs.

1. The driver creates a CIOS for sub-process input/output (figure 1a).
2. The driver calls the driven sub-process, directing it to use the sub-process CIOS.
3. The driven program runs doing apparently normal input-output (figure 1b).
   a) The driven program outputs with calls on the utility at level 1 (UTS_1). The output is buffered in the internal ring buffer or on a file.

If the internal buffer fills up, the output sub-process entry is called (figure 1c). The driver must then collect output with calls on UTS_2. It finally does an SPRETURN to UTS_1.

b) The driven program requests input via calls on UTS_1 (figure 1b). The utility calls the input entry point (figure 1c). The driver receives output from UTS_2, and finally calls UTS_1 with input. The utility at level 2 returns to level 1 and at level 1 returns the input to the driven program.

4. The driven program finally finishes processing and returns to the driver (figure 1a). There may be output left in the buffer.
5. If the driver wants the final output, the driver may call UTS_0 to receive it.
6. The CIOS is either destroyed explicitly or when the driver is destroyed.

Utility Calls for Control Input/Output Streams

As described previously this part of the utility makes a program's control input/output device independent. Use of the calls described below is highly recommended

The subsequent descriptions make reference to the following parameters and variables.

Name	Value	Meaning
NCIOS	8	Number of entries in the CIOS table
DCIOSN	-	Default CIOS number. Its value refers to the normal CIOS number of the calling sub-process. See the description of the utility sub-process table for details.
MXPOS	132	Maximum allowable number of print positions in a line.
MNPOS	40	Minimum number of print positions.

Most of the following UCALLs have one or more failure returns. The failure return is always a three-character error-code and an index to an error message string. (Note: call numbers and message numbers will be assigned later.)

Input/Output Calls

There are seven utility calls for reading and printing characters, numbers and strings. These calls are oriented toward the processing of lines, although character calls are provided. The utility buffers up to a line of output before it is sent to the output media. It keeps track of the print position so that a program can format output precisely.

READ'LINE - Read a Line of Input from the Selected CIOS

Declaration:

STRING FUNCTION READ'LINE(CIOSN, STRING LINE);

Success Return:

RETURN LINE;

Failure Returns:

(1) FRETURN ('CII', ---) unless -1 $\leq$ CIOSN $\leq$ NCIOS.

(2) FRETURN ('CIA', ---) if the caller may not access the CIOS.

(3) FRETURN ('SIZ', ---) if LINE is too small.

Action:

This call collects characters from the designated CIOS input medium until a new line character (∅) is reached. If there is output in the output half of the CIOS, output is initiated first. The input line is converted to canonical form if necessary.

Then the input part of the string is copied to LINE, the line is written on the terminal input file, and the position is set to zero. Line editing conventions appropriate to the terminal are followed. If CIOSN = -1, DCIOSN is used. This is the standard call for command input.

READ'CHAR - Read Character

Declaration:

```
CHARACTER FUNCTION READ'CHAR(CIOSN);
```

Success Return:

```
RETURN CHAR;
```

Failure Returns:

(1) FRETURN ('CII', ---) unless $-1 \leq$ CIOSN $\leq$ NCIOS.

(2) FRETURN ('CIA', ---) if the caller cannot access the CIOS.

Action:

This call collects a character from the designated CIOS and returns it to the caller. It also files the character in the line buffer. If the new line character is input, the buffer is converted to canonical form and output on the appropriate file(s). If CIOSN = -1, DCIOSN is used.

CONV'LINE - Convert Line to Canonical Form

Declaration:

STRING FUNCTION CONV'LINE(STRING LINE);

Success Return:

RETURN LINE;

Failure Returns:

None.

Action:

The LINE is converted to canonical form, including compression of multiple blanks. The canonical line is written over the input line.

PRINT'STRING - Print a String

Declaration:

```
FUNCTION PRINT'STRING(CIOSN, STRING STR);
```

Success Return:

```
RETURN;
```

Failure Returns:

(1) FRETURN ('CIT', ---) if CIOSN < -1 or CIOSN > NCIOS.

(2) FRETURN ('CIA', ---) unless the caller has access to CIOSN.

(3) FRETURN ('CIL', ---) if the position would be greater than 'LNG'.

(4) FRETURN ('CIO', ---) if the line buffer overflows.

Action:

The string STR is appended to the buffer. If it contains new line characters, each line is converted to canonical form and output. DCIOSN is used if CIOSN = -1.

PRINT'CHAR - Print a Character

Declaration:

FUNCTION PRINT'CHAR(CIOSN, CHARACTER CHAR);

Success Return:

RETURN;

Failure Returns:

Same as for PRINT'STRING.

Action:

CHAR is appended to the line buffer. If it is a new line character, the buffer is converted to canonical form and output. If CIOSN = -1, DCIOSN is used.

Creation and Control of a CIOS

The following UCALLs are used to create, initialize, read and delete control input/output streams. In general a sub-process must control the CIOS in order to change the access or device information. There is no restriction on reading the CIOS table.

CREATE'CIOS - Create and Initialize a New CIOS

Declaration:

```
INTEGER FUNCTION CREATE'CIOS(CIOSN), FRETURN;
```

Success Return:

```
RETURN CIOSN;
```

Failure Returns;

(1) FRETURN ('CII', ---) unless $-1 \leq$ CIOSN $\leq$ NCIOS.

(2) FRETURN ('CIF', ---) if

(a) CIOSN = -1 and there are no free entries, or

(b) CIOSN = -1 and the specified entry is not free.

Action:

If CIOSN = -1, the table is searched for a free entry and CIOSN is set to its value. Then the entry is cleared and the control lock is set to the name of the calling sub-process and the line length is set to 72.

READ'CIOS'FIELD - Read CIOS Field

Declaration:

INTEGER FUNCTION READ'CIOS'FIELD(FLD, CIOSN);

Success Return:

RETURN VAL;

Failure Returns:

(1) FRETURN ('CII', ---) unless $-1 \leq$ CIOSN $\leq$ NCIOS.

(2) FRETURN ('ARG', ---) if FLD is not acceptable (see below).

Action:

CIOSN is used to select a CIOS entry. If CIOSN = -1, DCIOSN is used. FLD selects one of fifteen fields within the selected CIOS entry in accordance with the following table.

FLD	VAL
Ø or 'CL'	Control Lock
1 or 'AL'	Access Lock
2 or 'POS'	Line position
3 or 'LEN'	Line length
4 or 'EST'	Echo On or Off
5 or 'IPT'	Input Type (3 character value)
6 or 'OPT'	Output Type (3 character value)
7 or 'SFN'	Saved input File Number
8 or 'OFN'	Output File Number
9 or 'IPN'	Input Number (line number, file number, or sub-process and entry number)

10 or 'OPN'	Output Number
11 or 'BWS'	Break strategy (see SET'CIOS'FIELD)
12 or 'DVT'	Device type (see CHIO interface manual)
13 or 'QIT'	Quit character
14 or 'ESC'	Escape character

SET'CIOS'FIELD - Set CIOS Field

Declaration:

INTEGER FUNCTION SET'CIOS'FIELD(FLD, DATA, CIOSN);

Success Return:

RETURN;

Failure Returns:

(1) FRETURN ('CII', ---) unless -1 $\leq$ CIOSN $\leq$ NCIOS.

(2) FRETURN ('CIC', ---) unless the caller controls the CIOS and control is required (see below).

(3) FRETURN ('CIA', ---) if the caller neither controls or has access to the CIOS.

(4) FRETURN ('FLD', ---) if the value of FLD is invalid (see below).

(5) In addition to the three general errors just described, errors specific to the value of FLD may occur. These are listed below by field.

CL

FRETURN ('DAT', ---) if the exclusive or of DATA and VAL anded with the complement of the caller's key is non-zero.

LEN

FRETURN ('DAT', ---) unless MNPOS $\leq$ DATA $\leq$ MXPOS or the value of POS is non-zero.

BWS

FRETURN ('DAT', ---) unless DATA is one of the values described below.

Action:

In general this call sets the value of the specified field to DATA. It also returns the old value of the field. As usual if CIOSN equals -1, DIOSN is used. The action taken for each value of FLD is:

FLD	Action
Ø or 'CL'	Sets CL to DATA; requires control of the CIOSN. If the control lock is set to Ø, the CIOS is deleted and any files associated with the CIOS are closed. Similarly terminals will be freed.
1 or 'AL'	Sets AL; requires access. Any value of data is acceptable.
3 or 'LEN'	Requires access.
4 or 'EST'	Turns echoing on if DATA is non-zero, off if zero.
11 or 'BWS'	Sets the break and wakeup strategy according to the following table. This call has effect only for terminal input.

DATA	Break on
Ø or 'NEV'	never break
1 or 'CTL'	break only on controls
2 or 'PUN'	break on control and punctuation characters
3 or 'ALL'	break on all characters

12 or 'DVT'	Sets the device type; requires control. See

the CHIO manual for settings.

13 or 'QIT' Sets the QUIT character. Requires access to the CIOS and the QUIT interrupt.

14 or 'ESC' Sets the ESCAPE character. Requires access to both the CIOS and the ESCAPE interrupt.

The other fields of a CIOS except for POS are set by the following calls.

SET'CIOS'INPUT - Set the CIOS Input Medium

Declaration:

FUNCTION SET'CIOS'INPUT(CIOSN, CODE, INDEX, FILE)

Success Return:

RETURN;

Failure Returns:

(1) FRETURN ('CII', ---) unless $-1 \leq$ CIOSN $\leq$ NCIOS.

(2) FRETURN ('CIC', ---) if the caller does not control the CIOS.

(3) FRETURN ('FLD', ---) if CODE is invalid.

(4) FRETURN ('DAT', ---) if INDEX or FILE are not legal values.

Action:

This call initializes the input half of the CIOS. If CIOSN = -1, DCIOSN is used. CODE is either a three character abbreviation for the input medium, or the code value. The interpretation of the values of INDEX and FILE is summarized by the following table.

CODE	INDEX	FILE	Meaning
Ø or 'NUL'	Ignored	Ignored	No Input Source
1 or 'TRM'	CHIO line number	Ignored	Normal terminal input
2 or 'FIL'	OFT index	Ignored	File Input
3 or 'TSF'	CHIO line number	OFT index	Terminal input Saved on File
4 or 'SPE'	Sub-process entry	Ignored	

The value of INDEX for 'SPE' is twelve bits. The first four bits are the sub-process number and the last eight are the entry number.

SET'CIOS'OUTPUT - Set the CIOS Output Medium

Declaration:

FUNCTION SET'CIOS'OUTPUT(CIOSN, CODE, INDEX, FILE)

Success Return:

RETURN;

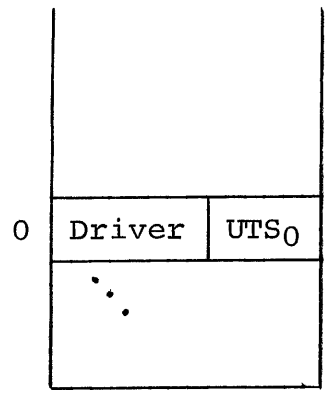
Failure Returns:

Identical to SET'CIOS'INPUT.

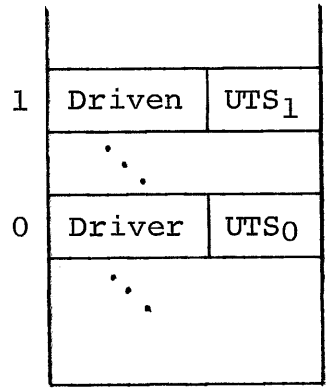
Action:

This call initializes the output half of a CIOS. It is symmetric with SET'CIOS'INPUT. The following table gives the interpretation of the parameters.

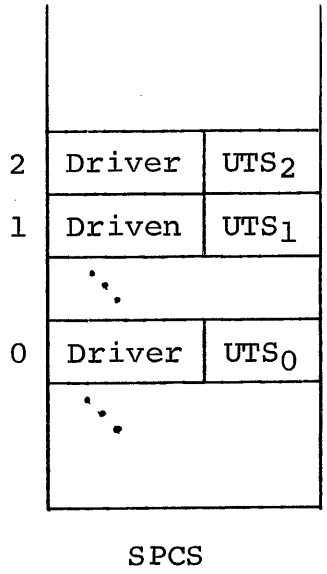
CODE	INDEX	FILE	Meaning
Ø or 'NUL'	Ignored	Ignored	Delete Output
1 or 'TRM'	CHIO line number	Ignored	Normal Terminal output
2 or 'FIL'	OFT index	Ignored	File Output only
3 or 'TOF'	CHIO line number	OFT index	Terminal Output to file
4 or 'SPE'	Sub-process entry	Ignored	



a. Driver Running at Level Ø



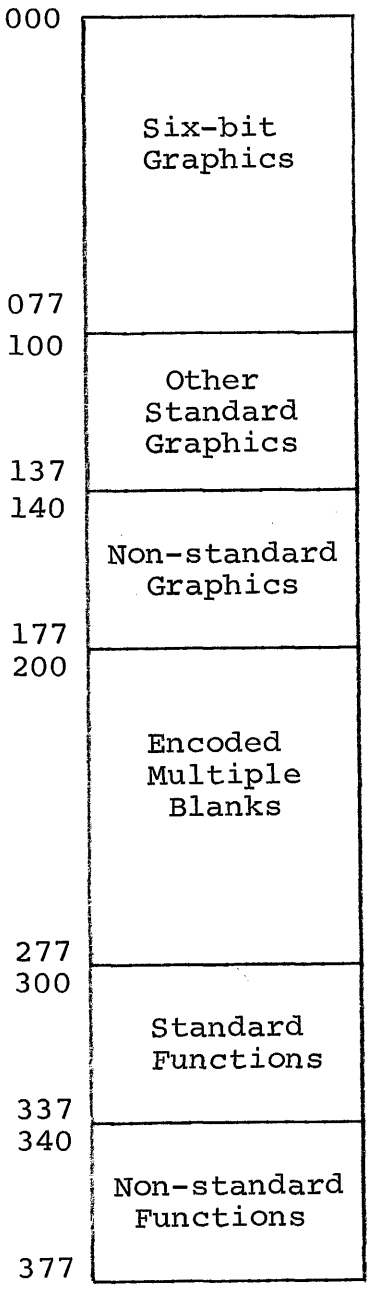
b. Driven Program Running at Level 1



c. Driver Called to Process Input (Output) at Level 2

Figure 1. Sub-Process Call Stack During Sub-Process Input/Output

Appendix A - Internal Character Set and Octal Equivalents



Parts of Internal Character Code

	0	1	2	3	4	5	6	7
000	↛	!	"	#	$	%	&	'
010	(	)	*	+	,	-	.	/
020	ø	1	2	3	4	5	6	7
030	8	9	:	;	<	=	>	?
040	@	A	B	C	D	E	F	G
050	H	I	J	K	L	M	N	O
060	P	Q	R	S	T	U	V	W
070	X	Y	Z	[	\	]	↑	←

Six-bit Graphics

	0	1	2	3	4	5	6	7
100	_	a	b	c	d	e	f	g
110	h	i	j	k	l	m	n	o
120	p	q	r	s	t	u	v	w
130	x	y	z	{	|	}		

Other Standard Graphics

	0	1	2	3	4	5	6	7
300	NUL							BEL
310	BS		NL		FF		RRS	BRS
320			HLF		HLR			
330								

Standard Functions

Appendix B - Syntax of Canonical Form

1. line = [motion] 1$ ppos $(motion 1$ ppos) [up:feed] ⟨NL⟩ /
   ⟨FF⟩ ⟨NL⟩ / ⟨NL⟩;
2. motion = $⟨SP⟩ (⟨SP⟩ / V:motion);
   V:motion = up:feed / down:feed;
   up:feed = ⟨HLR⟩$⟨HLR⟩;
   down:feed = ⟨HLF⟩$⟨HLF⟩;
3. print:pos = char:pos $(⟨BS⟩ down:feed char:pos)
4. char:pos = former $(⟨BS⟩ former)
5. former = [setup] graphic
6. setup = color $ function
   color = ⟨RRS⟩ / ⟨BRS⟩