

p.n: 991 03509

RCSL No: 31-D617

Edition: October 1980

Author: Ejvind Lynning

Title:

PASCAL80 Driver Conventions

Keywords:

PASCAL80, driver programming.

Abstract:

This manual contains a standard for the interface between a PASCAL80 driver and process(es) using the driver.

(16 printed pages)

Copyright © 1982, A/S Regnecentralen af 1979

RC Computer A/S

Printed by A/S Regnecentralen af 1979, Copenhagen

Users of this manual are cautioned that the specifications contained herein are subject to change by RC at any time without prior notice. RC is not responsible for typographical or arithmetic errors which may appear in this manual and shall not be responsible for any damages caused by reliance on any of the materials presented.

<u>TABLE OF CONTENTS</u>	<u>PAGE</u>
1. INTRODUCTION	1
2. PROCEDURES	2
3. FORMATS	3
3.1 Header Message	4
3.1.1 Driver Message	4
3.1.2 Answer	6
3.2 Data Buffer	7
4. DRIVER MANUALS	10

1. INTRODUCTION

1.

This note deals with the procedures and formats used for communication between a driver and one or more process incarnations in a PASCAL80 system which use(s) the driver to provide i/o functions.

The standard described in this note applies to drivers written in PASCAL80 as described in the PASCAL80 Report (RCSL No 52-AA964) as it stands at the present date. Future revisions of the PASCAL80 language will most likely entail revisions of the driver conventions as well.

The conventions described here apply not only when the driver process and the process(es) using the driver are programmed completely in PASCAL80, but also when one or more of the processes are programmed in machine code, as long as they communicate by using the PASCAL80 message exchange mechanisms (signal, wait, etc.).

By definition a PASCAL80 driver is (an incarnation of) a process containing a CHANNEL statement. This definition indicates that a driver performs i/o functions, but it does not characterise the amount of processing, such as e.g. error correction or data structuring, which is performed. The functions related to i/o may therefore in different cases be distributed differently between the driver and the process incarnation(s) which make use of the driver. In order that the present conventions should not introduce undesirable restrictions on this functional distribution they have been kept fairly minimal.

Procedures and formats which need not be adhered to in all cases, but which should be followed whenever they are appropriate, are included as recommendations.

The note also contains some hints on the structure and contents of driver manuals.

2. PROCEDURES

2.

By convention each driver incarnation has a unique request semaphore. A request to the driver is made in the form of a message, called a driver message, signalled to this semaphore.

When a driver has processed a request the message is returned (i.e. signalled to its answer semaphore). When returned, the message is referred to as an answer.

If a driver does not process and answer requests in the same order as the driver messages are received this should be explained in the driver manual.

When a driver maintains internal request (buffer) queues it shall in general be possible to recall the pending requests by means of a "regret" request.

3. FORMATS

3.

In the following we use the term i/o-data to mean data which is actually transferred to or from an input or output device (notice that a communication line is also considered an i/o device). The term "data buffer" is used with the same meaning as "message data" in the PASCAL80 Report.

The "user" bytes u1, u2, and u3 of the header of a driver message/answer are used to specify the function requested from a driver and after processing of a request to provide result information. When u1-u3 are not sufficient to hold the request/-result information, part of this information may also be placed in the data buffer.

I/o data, including address information to indicate the precise amount of data, are held in the data buffer of the driver buffer/answer.

The general format is illustrated below:

	<u>driver message</u>	<u>answer</u>
	message header	message header
u1	function	unchanged
u2	no standard use	result
u3	device address or data byte if needed	data byte if needed
	data buffer with output data	data buffer with input data
byte 0-1	first	unchanged
byte 2-3	last	unchanged
byte 4-5	(lastw)	next
(from	function or address information	result/status information)
(byte 6	not held in u1-u3	not held in u1-u3)
remaining		)
bytes	output data	input data

3.1 Header Message

3.1

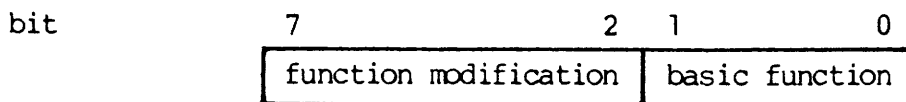
The "user" bytes u1-u4 are used in driver messages and answers as described in this section.

3.1.1 Driver Message

3.1.1

u1: function

The byte u1 is used to specify the function to be performed by the driver as shown below (bit 7 is most significant, bit 0 least significant), i.e. $u1 = \text{basic function} + 4 * \text{function modification}$.



The basic function is coded as follows:

<u>value</u>	<u>meaning</u>
--------------	----------------

- | | |
|---|---|
| 0 | control operation, i.e. any operation that does not involve an actual transfer of data, |
| 1 | read (receive) data operation, |
| 2 | write (transmit) data operation, |
| 3 | write and read data operation, i.e. write followed by read using the same buffer. |

There is no standard for the coding of the function modification. When necessary, this field is used in a driver dependent way to distinguish between different function requests with the same basic function code. The following coding is recommended (bf=basic function, fm=function modification):

- bf=0, fm=0: get device status,
- bf=0, fm=1: initiate/connect/open device,
- bf=0, fm=2: terminate/disconnect/close device,
- bf=0, fm=3: regret request,
- bf=0, fm=5: pause (release channel to test program),
- bf>0, fm=0: block transfer of binary data to/from data buffer.

u2: not used

Normally u2 is not used to hold information to a driver.

u3: device address or single byte i/o data

There are two alternative uses of the u3 byte.

When a driver services more than one device, u3 is used to hold device address information.

When data is transferred in single byte mode, i.e. a driver request causes the transfer of only one byte, u3 is used to hold this byte, so that a header message (message without data buffer) may be used.

u4: not used

The byte u4 must not be used in driver messages.

Notes

These conventions do not rule out the use of u3 for other purposes than described, nor the use of u2, in a driver message. However, such use is not standardised.

When the bytes u1-u3 are not sufficient to hold a driver request, additional information may be placed in the data buffer. In the case of a data transfer request, i.e. when the data message is used for i/o data, such information should start in byte 6, otherwise there are no restrictions. Using the PASCAL80 LOCK statement such information can freely be given a suitable type-definition (as opposed to u1-u3 which are restricted to the type 0..255 by the language definition).

3.1.2 Answer

3.1.2

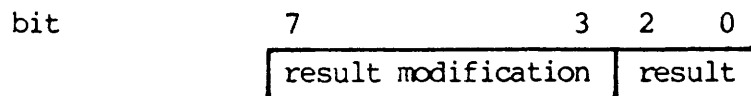
u1: unchanged

When a driver message is returned as an answer u1 is unchanged.

u2: result

The byte u2 is used to give result information from a driver to the requesting incarnation (actually the incarnation to which the request message is returned) as shown below, i.e.

$u2 = \text{result} + 8 * \text{result modification}$



The result is coded as follows:

<u>value</u>	<u>meaning</u>
0	processed successfully,
1	not processed because of a previous error not yet repaired (used with multi-buffering),
2	transient error, i.e. error which may be corrected by the driver; accompanying data may contain errors, e.g. parity errors.
3	persistent error, i.e. error which must be corrected by operator intervention and/or a suitable driver request to reset the device,
4	illegal function, i.e. unintelligible driver message,
5-6	reserved, not used presently.
7	not used; this value may be used to indicate that a message does not contain an answer from a driver.

The result modification is used in a driver dependent way to provide additional information needed to distinguish different results, e.g. different kinds of transient error requiring different repair. As a general recommendation, only distinctions which will be useful for the requesting process incarnation should be provided.

u3: unchanged if device address or single byte i/o data

When u3 is used in a driver message to hold a device address it should not be changed by the driver.

If data is transferred in single byte mode u3 is used to hold data read from a device.

Other uses of u3 are not ruled out, but they are not standardised.

u4: not used

The byte u4 must not be changed by a driver. This allows the user of a driver to use u4 for purposes other than communication with the driver.

Note

Result information which cannot be placed in u1-u3 may be held in the data buffer of an answer in the same way as request information in the data buffer of a driver message.

3.2 Data Buffer

3.2

The format of the data buffer is standardised for block data transfer requests, i.e. when the buffer is used to hold i/o data. Other uses of the data buffer, e.g. for device status information are not standardised.

When all request/result information needed in a driver message/answer can be held in ul-u3 the data buffer (of size n bytes) is treated by a driver as a record of the type:

```
data_buffer=RECORD
    first:      INTEGER;
    last:       INTEGER;
    lastw/next: INTEGER;
    (* when necessary, additional request/result *)
    (* information is placed here *)
    data:       ARRAY(6..n-1) OF byte (* i/o data *)
END(*RECORD*);
```

The locations ("sub-array") data(first..last) is called the data area of the buffer.

The output data for a write/transmit operation are taken from the data area. Note that in the case of a write and read data operation lastw (not last) is used as the index of the last data byte to be written. If the write operation completes normally, next=last+1 (or lastw+1) afterwards.

Similarly the input data of a read/receive operation are placed in the data area. If this area is not sufficient for the received data, buffer overrun will occur. If the read/receive operation completes normally, next, in the answer, will be the index of the location following the last received byte. Thus next indicates the length of the block that was actually received into memory.

A driver must not change first or last.

Some drivers may accept a data buffer stack. As a general rule each data buffer in the stack is formatted as described above, and the concatenation of the data areas of the buffers, from top to bottom, is considered one logical i/o data record. Only the top message header is interpreted according to the conventions spelled out in section 3.1. When a driver supports stacked data buffers it should be explicitly mentioned in the driver manual.

When request/result information in addition to what can be held in u1-u3 is needed, such information should be placed between lastw/next and the data array; the min bound of this array (6) should be incremented accordingly. The formatting of such request/result information is not standardised.

4. DRIVER MANUALS

4.

The following contents are suggested for driver manuals.

Section 1: Introduction

Section 2: Functions Supported by the xxx Driver

This section contains a thorough description of the functions provided by the driver.

Section 3: Driver Interface

This section contains explicit specifications of all formats and codes used in driver messages as well as the answers that may occur for each type of request. The actions caused by a particular request need not be explained in detail, but references to section 2 should be given whenever appropriate.

The material in this note need not be repeated in all driver manuals, a reference is sufficient.

Section 4: Parent Process Responsibilities

This section contains information on how to declare, link, create, and start the driver. Specifically all process parameters of the drivers should be described. Suggested values of the storage and priority parameters needed for create and start should also be given.

RETURN LETTER

Title: PASCAL80 Driver Conventions

RCSI. No.: 31-D617

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

Do you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

..... Fold here

..... Do not tear - Fold here and staple

Affix
postage
here

RC International

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark