
Subject: Re: AXP VMS IDL problem with Control_C?

Posted by [rivers](#) on Sat, 25 Feb 1995 23:59:26 GMT

[View Forum Message](#) <> [Reply to Message](#)

>
>> We are having problems with the Control_C interrupt in VMS IDL running under
>> AXP OpenVMS 6.1. The first time one calls Control_C it works properly,
>> as a keyboard interrupt returning one to the IDL prompt. But on subsequent
>> calls, it behaves like Control_Y and aborts one to the \$ prompt.

> For what it's worth, this behavior seems to follow the second ^C in
> any OpenVMS AXP application, including DEC ones. I wonder if it's an
> (intentional) feature....
>

That is certainly not universally true. All the applications I have written, which originally ran on VAX/VMS and now run on AXP/VMS, handle ^C just fine. I did not recode anything when recompiling on AXP. You have to be aware of the (documented) way the ^C handler works. You need to redeclare the handler after each ^C. For those interested I am appending the routines I use. Routine set_control_c() must be called again each time from the ast_routine if it is desired to treat multiple ^Cs the same way.

What DEC applications treat ^C differently under AXP/VMS from VAX/VMS?

Mark Rivers	(312) 702-2279 (office)
CARS	(312) 702-9951 (secretary)
Univ. of Chicago	(312) 702-5454 (FAX)
5640 S. Ellis Ave.	(708) 922-0499 (home)
Chicago, IL 60637	rivers@cars3.uchicago.edu (Internet)

```
SUBROUTINE clear_control_c
IMPLICIT NONE
CALL set_control_c(0)
RETURN
END
```

```
SUBROUTINE set_control_c(ast_routine)
```

```
c Routine to declare an AST routine to be entered on ^C.
c Modified by MLR 17-NOV-1991 to return harmlessly if SYS$INPUT is not a
c terminal.
IMPLICIT NONE
INCLUDE '($iodef)
```

```
INCLUDE '($DVIDEF)
INTEGER istat, channel, sys$assign , sys$quiow, ast_routine
INTEGER LIB$GETDVI
```

```
LOGICAL first_time/.true./
LOGICAL terminal_device
```

C

```
IF (first_time) THEN
  first_time = .false.
  istat = sys$assign('sys$input', channel,,)
  IF (.NOT. istat) CALL lib$signal(%val(istat))
```

C

```
Find out if SYS$INPUT is a terminal
  istat = LIB$GETDVI(DVI$_TRM,, 'SYS$INPUT', terminal_device)
  IF (.NOT. istat) CALL LIB$SIGNAL(%val(istat))
END IF
```

C

```
IF (.NOT. terminal_device) RETURN
IF (ast_routine .NE. 0) THEN
  istat = sys$quiow
  &      ( ,
  &      %val (channel),
  &      %val (io$_setmode .OR. io$m_ctrlcast),
  &      ,
  &      ,
  &      ,
  &      ast_routine,
  &      ,
  &      ,
  &      ,, )
  ELSE
  istat = sys$quiow
  &      ( ,
  &      %val (channel),
  &      %val (io$_setmode .OR. io$m_ctrlcast),
  &      ,
  &      ,
  &      ,
  &      %val(0),
  &      ,
  &      ,
  &      ,, )
  END IF
```

C

```
IF ( .NOT. istat ) CALL lib$signal(%val(istat))
RETURN
END
```