TPrJob structure

```
#include <PrintTraps.h>

typedef struct TPrJob {            Size  Offset  Description
   short          iFstPage;         2      0     Page number of first page to print
                                                 (1-based)
   short          iLstPage;         2      2     Last page to print
   short          iCopies;          2      4     Number of copies to print
   char           bJDocLoop;        1      6     Print method; 0=draft, 1=spooled
   Boolean        fFromUsr;         1      7     (used internally)
   PrIdleProcPtr  pIdleProc;        4      8     Address of background routine; 0=use
                                                 standard
   StringPtr      pFileName;        4     12     Addr of spool filename; 0=default "\pPrint
                                                 File"
   short          iFileVol;         2     16     Spool volume reference number
                                                 (0=default)
   char           bFileVers;        1     18     Spool file version number (use 0 here)
   char           bJobX;            1     19     (not used)
} TPrJob;                          20

typedef TPrJob *TPPrJob;
```

Notes: TPrJob gets set according to selections made in the print dialogs, especially the job dialog presented via **PrJobDialog**. It is not used directly in any Printing Manager functions, but it defines a subrecord of the TPrint structure (ie, TPrint.prJob) which is used in many Printing Manager calls.

The iFstPage and iLstPage fields define the range of pages which will be printed. Calls to **PrOpenPage** are counted as they are made (starting from 1) and if the current page number falls in this range, it is printed. Otherwise, the page is not output. To avoid time-consuming drawing of pages which will be skipped, you will want to set this range yourself. See **PrOpenPage**.

It is important to check the bJDocLoop flag after you close a document. If it has been set to 1, then the print job has been spooled and you must call **PrPicFile** to actually output the print image (see **PrOpenDoc** for an example).

Background Processing

If you leave pIdle procedure as 0, then the default background task will check for the press of -. (Cmd-period) and abort the printing if that occurs.

You might want to provide a user-friendly dialog window with some sort of Cancel button. Another common enhancement is to display some sort of status as to which page is being printed, etc.

To do so, you must draw the dialog before starting to print. As the last step before printing the first page, store the address of your background routine into pIdleProc (note this gets set back to 0 by **PrValidate**, **PrOpenDoc**, **PrStlDialog**, and **PrJobDialog**). Your routine should be declared as:

```
pascal void MyIdleProc(void)            /* Note:  No parms in or out */
{
    // remember to save the print driver's GrafPort
    // call CurResFile to remember the print driver's resource file
    // if you are loading resources
    //. check for mouseDown in your Cancel button ...
    //. always check for Cmd-period ...
    //. to cancel, call PrSetError( iPrAbort );
    // remember to restore the grafPort (if you changed it)

    // remember to restore the printer driver's GrafPort
    // call UseResFile to restore the print driver's resource file
}
```

Make sure you install your pIdle routine into the print record before calling **PrOpenDoc**. If you do not install your procedure before calling it, the printer driver does not give the application's pIdle routine any time.

Avoid calling **PrError** from within your pIdle routine. Any errors that occur while it is executing are usually temporary, and serve as internal flags for communication with the printer driver. They are not intended for your application. If you absolutely have to call **PrError** from within your pIdle routine, never abort printing within the idle procedure itself. Wait until the last called printing procedure returns, then check to see if the error still remains. Apple has stated that "attempting to abort printing within an idle procedure is a guarantee of certain death".

If you install a procedure for handling requests to cancel printing, with an option to pause the print process, beware of timeout problems when printing to the Laserwriter. Communications between the Macintosh and the LaserWriter must be maintained to prevent a job or wait timeout. If there is not any communication for a period of time (over two minutes), the printer times out and the print job terminates due to a wait timeout. Or, if the print job requires more than three minutes to print, the print job terminates due to a job timeout. There is no good method for determining what kind of printer an application is printing to, so it is probably a good idea to document the possibility of a LaserWriter timing out for a user who chooses to select "pause" for over two minutes.

Some printer drivers do not support pIdle routines. This should not be a problem as long as you do not assume that your pIdle routine is called at print time. You should only use your pIdle routine to display the dialog box and respond to a user pausing, continuing or canceling a print job.

Your custom routine will be called often, especially while the system is waiting for the printer hardware. If you use Quickdraw to display anything in your dialog window, be sure to restore the printing grafPort before returning. See **PrPicFile** for related information.