Once you get the pl_all_l1.sav, you can run it like this:

  pl_all_l1, arg1,arg2,arg3

The *Run Command* in the Options/Project is the name of the routine
which will be executed when you 'run' the project.

As to you second question, it works fine when I build the project. I'm
using IDL5.6.

Tian.

sbkim@satrec.kaist.ac.kr (sab) wrote in message
news:<6f9740d5.0311150036.a5da921@posting.google.com>...
> I wish to use IDL routines from C++. To do this,
> I am building a project (.sav) from .pro IDL routines.
> We are using IDL 6.0. There are two problems
> (1) how to handle input commandline arguments? For
> example, IDL routine is as follows:
>
> pro pl_all_l1, arg1, arg2, arg3
> ...
> end
>
> Then we are now building a project (File/Open Project).
> In Project/Options, we see the following:
> Run cmd: pl_all_l1
> Build cmd: pl_all_l1.
> Now the question is: what should we do to enable
> pl_all_l1.sav to accept the commandline arguments?
>
> Also, 'make_rt' converts pl_all_l1.sav to pl_all_l1.exe.
> Then what is the syntax for executing pl_all_l1.exe with
> the commandline arguments?
>
> (2) our IDL routine 'pl_all_l1' contains a common header
> inside it:
>
> pro pl_all_l1, arg1, arg2, arg3
> @commonheader.pro
> ...
> end
>
> "commonheader.pro" is a simple declaration of constants:

\>> ls commonheader.pro
\> YES= 1
\> NO = 0
\>
\> In IDL, pl_all_l1 works fine. But "@commonheader.pro"
\> causes fatal errors during the project building. How
\> can we resolve this?
\>
\> Thanks,
\>
\> Sab