ReadMe.txt V3.3
PL1UC, A Must For All Active Group Members

Introduction

First of all I say thank you to Markus Loew. His tries and testing allowed finally to make the most important changes in PL1UC. We got aware that without this "hidden" tool we would not be able to have a beneficial exchange of code brought to us as conversation attachments, or as files in the group's file section.

Many of the comments in the original ReadMe.txt (see page 5) still apply so I will not repeat them here.

There are 2 ways to obtain PL1UC. The first is download from the **https://groups.io/g/pl1f-and-mvs38j/** file section folder "#UTILITY PL1UC". The other way is to clone from **https://github.com/oldbuzzard54/PL1UC-Preprocessor.**

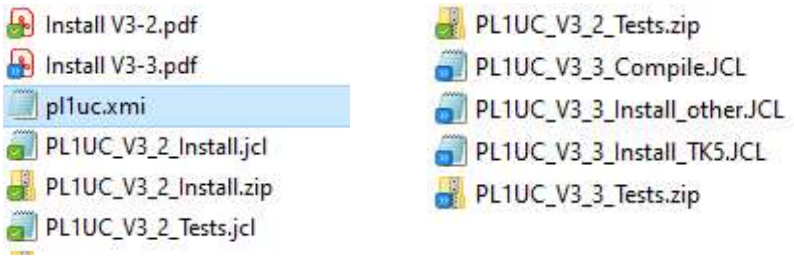
There is a new install package for PL1UC V3.3. Be aware that the JCL files (install and test) are now based on the TK5- MVS V1 configuration. Some minor changes may be required to install on TK4- or TK3. With minor exceptions, the new version of PL1UC should be transparent to existing users.

Installation

For simplicity, unless otherwise given, it is assumed that all files are prefixed with PL1UC_3_3 or simply PL1UC on the PC files. On TSO, it is assumed the high level qualified is HERC01.

From the groups.io file section:

- Download and unzip the PL1UC_V3_3.zip file. The zip file will contain the following files:

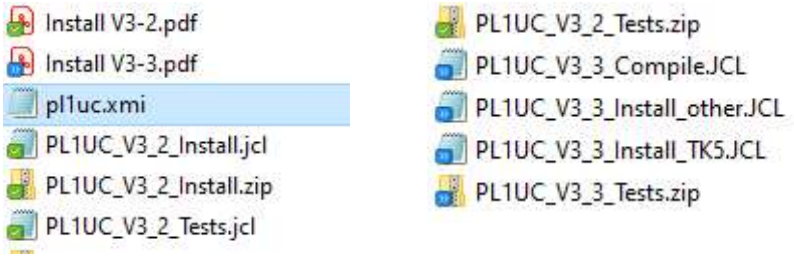


- BINARY upload the PL1UC.xmi file to MVS as HERC01.PL1UC.XMI.
- Submit the PL1UC_V3_3_TK5.install.jcl or PL1UC_V3_3_other.jcl to MVS. This will install PL1UC on your system.

The job should end with return code 0000 for all steps. The job is coded so if any step has a non-zero return code, the install will not complete.

From the github.com:

- Download (or clone) from github.com as described above.



- BINARY upload the PL1UC.xmi file to MVS as HERC01.PL1UC.XMI.
- Submit the PL1UC_V3_3_TK5.install.jcl or PL1UC_V3_3_other.jcl to MVS. This will install PL1UC on your system.

The job should end with return code 0000 for all steps. The job is coded so if any step has a non-zero return code, the install will not complete.

Post Install

To be sure that everything works well you may submit the test job in PL1UC_V3_2_TESTS.JCL which should end with return code 0000 for all steps. No changes were made to the JCL so the V3_2 is still valid. Please note the purpose of these tests is to insure the procs are set up correctly. Samples of the test jobs output can be found in PL1UC_V3_3_TESTS.ZIP.

If your install and test jobs do not end as expected then please post a conversation message within the topic "PL1UC, A Must For All Active Group Members" . It is also important to send your message saying that it works correctly on your system. Thanks a lot!

Changes for PL1UC V3.3

V3.3 is a bug fix as well as a couple small enhancements.

The bug was in the routines for processing %include statement. It resulted in nested comments being generated and code following the include being dropped.

Enhancements include:

1. In the processed code, includes are surrounded with start/end comments.
2. A comment line is generated. If the first line of the input program does not contain a comment, the comment line will be the first in the output. If a comment is found in the first line of the program, the comment line will be the second in the output.

Enhancements for PL1UC V3.2

Support was added to process %includes encountered. This means included code can be done in upper and lower case if so desired. Additional JCL is required to use this new feature. An EXEC statement parm of 'PLUS' was added to request this. If no parm is given, the default is 'CLASSIC'. In CLASSIC mode, no includes will be processed and PL1UC V3.2 will function the same as V3.1a.

If PLUS is requested, additional DD statement(s) must be added to the JCL. The default DDNAME is PL1LIB. It should be added before the SYSIN DD statement.
Example:

```
//MYSTEP   EXEC     PL1UFCLG,MODE='PLUS',
//   PARM.PL1L='...',
//   PARM.LKED='...',
/    REGION.GO=...
//PL1U.SYSPRINT  DD    SYSOUT=... adapted source code print output
//PL1U.PL1LIB   DD   DSN=…. Your include library
//PL1U.SYSIN     DD    DSN=....   source code input
//PL1L.SYSPRINT  DD    SYSOUT=... compiler print output
//LKED.SYSLMOD   DD    DSN=...
//LKED.SYSPRINT  DD    SYSOUT=...
//GO.....
```

If you want PL1UC to search more than 1 PDS library for members, concatenated datasets are supported. For example, you have a testing and production library, you might use something like this:

```
//PL1U.PL1LIB   DD   DSN=…. Your TEST include library
//            DD  DSN=….   Your Production include library
```

In this case, PL1UC will search for the %INCLUDEd member first in the first library listed. If not found, it will search the next library listed. If not found in any of the libraries, it will be listed as a "not found" in the report.

Since %INCLUDEs can specify DDNAME, additional JCL for each DDNAME must be supplied. The best way to explain this is with examples. Below are a PL/1 code fragments to be processed by PL1UC.

Example 1:

```
%INCLUDE RECORDA;
%INCLUDE MACROB;
This would require the DD to be added to the JCL.

//PL1U.PL1LIB   DD   DSN=…. Your include library that contains RECORDA and MACROB
```

Example 2: Testing changes to RECORDA with production MACROB

```
%INCLUDE RECORDA;
%INCLUDE MACROB;
This would require the DD to be added to the JCL.

//PL1U.PL1LIB   DD   DSN=…. Your include library that contains RECORDA
//     DD DSN=….              Your production library that contains MACROB
```

Example 3: Testing changes to RECORDA with production MACROB

```
%INCLUDE TESTLIB(RECORDA);
%INCLUDE PRODLIB(MACROB);
This would require the DDs to be added to the JCL.

//PL1U.TESTLIB   DD   DSN=…. Your include library that contains RECORDA
//PL1U.PRODLIB  DD  DSN=….   Your production library that contains MACROB
```

As you can see, there are multiple ways to accomplish the same results. As a matter of style, examples 2 and 3 will produce the same results. Example 2 would be the recommended way to handle it. When RECORDA is moved to production, no changes would be required to the code. With example 3, the include would need to be changed. Stuff like this contributes to installation problems.

JCL Procdures

Five JCL procs will be installed in SYS2.PROCLIB. These procs all start with PL1UF. They are copies of procs starting with PL1LF with the addition of the PL1UC step. Of course you can still use the "old" procs if you like.

No changes were made to the procs for V3.3

ReadMe.txt V3.3
PL1UC, A Must For All Active Group Members

Original ReadMe.txt
PL1UC, A Must For All Active Group Members

First of all I say thank you to John James. His tries and testing allowed finally to make the most important changes in PL1UC. We got aware that without this "hidden" tool we would not be able to have a beneficial exchange of code brought to us as conversation attachments, or as files in the group's file section.

Many of us experienced the problem that PL/I(F) does not work, as soon as the source code contains OR operators '|', CONCAT operators '||', or NOT operators '¬'. It depends from the code page you are using in Hercules. It is even more confusing: In most cases the bad operators do not show up in the compiler listing, because they have been replaced by spaces.

If you want to download source code from other members for testing, or your PL/I(F) compiler does not work properly, then I recommend to read the following instructions:

The most convenient way to make accessible PL/I(F) is to install PL1UC which you find in the group's file section folder "#UTILITY PL1UC". Be aware that the JCL files (install and test) are based on the TK4- MVS original configuration. May be that you have to adapt them, in order to make them work in your Hercules/OS.

For TK4- users it's very simple. Download the JCL Files PL1UC_INSTALL-F.JCL and PL1UC_TESTS-F.JCL. Submit the install job. For people who alredy use PL1UC the job should end with return code 0000 for all steps, for new users IDCAMS will end with return code 0008 because there was no existing version to be deleted. To be sure that everythig works well you may submit the test job which should end with return code 0000 for all steps. If your install and test jobs do not end as expected then please post a conversation message within the topic "PL1UC, A Must For All Active Group Members" . It is also important to send your message saying that it works correctly on your system. Thanks a lot!

After the installation please do not use the procedures PL1LFC, PL1LFCG, PL1LFCL, and PL1LFCLG anymore, but use PL1UFC, PL1UFCG, PL1UFCL, and PL1UFCLG, particularly when source code to be compiled comes from somewhere else.

Example:

instead of

```
//MYSTEP   EXEC    PL1LFCLG,PARM.PL1L='...',PARM.LKED='...',REGION.GO=...
//PL1L.SYSPRINT  DD    SYSOUT=... compiler print output
//PL1L.SYSIN     DD    DSN=....   source code input
//LKED.SYSLMOD   DD    DSN=...
//LKED.SYSPRINT  DD    SYSOUT=...
```

```
//GO.....
```

please write

```
//MYSTEP   EXEC     PL1UFCLG,PARM.PL1L='...',PARM.LKED='...',REGION.GO=...
//PL1U.SYSPRINT  DD    SYSOUT=... adapted source code print output
//PL1U.SYSIN     DD    DSN=....   source code input
//PL1L.SYSPRINT  DD    SYSOUT=... compiler print output
//LKED.SYSLMOD   DD    DSN=...
//LKED.SYSPRINT  DD    SYSOUT=...
//GO.....
```

From now on you will not have to bother about NOT's and OR's anymore. You may compile source code coming from anyone w/o modifications. And the PL/I language may be written in lower and upper case letters. The only annoyance you may encounter is the OR '|' maybe looking more like '×', and the CONCAT '||' like '××' in the compiler listing.