



## TPF 41 System Code Build using MakeTPF V2 for TPF41

Copyright International Business Machines Corporation 2006, 2007. All Rights Reserved.

Note to US Government Users Restricted rights - Use, duplication or disclosure restricted by GSA ADP Schedule contract with IBM Corp.

Note: Before using this information and the product it supports, read the general information under "NOTICES" in this document.

### CONTENTS

This file includes the following information:

- 1.0 INTRODUCTION
- 2.0 PREREQUISITES
- 3.0 SETUP INSTRUCTIONS
- 4.0 BUILD INSTRUCTIONS
- 5.0 NOTICES
- 5.1 Trademarks

### 1.0 INTRODUCTION

This package contains a set of makefiles that can be used to generate the binaries for the TPF 41 system code. These files, and the build procedures that follow, are provided as a sample only and are not intended to be a replacement for the TPF 41 SIP build process.

The procedures below describe the setup and build steps needed to generate a BSS system. The build will be set up in two layers, one containing the source code (and all binary items that cannot be generated by maketpf but are needed for the build), and one layer for the target files generated during the build, including objects, listings, etc.

The intent of this setup is to isolate this build from your existing TPF 41 PDSs and hfs. The instructions that follow will describe what files need to be copied from your existing hfs and PDSs to setup this environment.

For this set of procedures, the source layer will be set up using the names:

- HFS: /u/tpf41/mtpfsrc
- PDS: ACP.\*.MTPFSRC

The target layer will be set up using the names:

- HFS: /u/tpf41/mtpfout
- PDS: ACP.\*.MTPFOUT

It is possible to build only in one layer, but for this test, it is easier to show what is read as input, and what is produced as output when the contents are separated. It also makes it easier to clean up and start over because the cleaning can be done at a directory level rather than file-by-file.

With few exceptions, the output targets, including the BAL objects, C load modules, and listings, will all be written to hfs. The exceptions include:

- STUBs and CLIBs, which must be written to a PDS for the USS "ld" link program to be able to locate them.
- The C load module linkage objects (CSTRTD40, CSTRTL40, CSTDLL40) which must reside in a PDS so that the USS "ld" link command can locate them.

The build is set up using the default for configuration file variable that adds the system name as an automatic suffix on the PDS names (for the system configuration dependent PDSs). The variable is:

```
TPF_USE_SYS_EXT := YES
```

As an example, specifying YES will add the system name as a suffix on all PDSs that have configuration dependencies, like:

```
ACP.SYMACRO.MTPFSRC.BSS
```

The use of the file extension can be turned off, by setting:

```
TPF_USE_SYS_EXT := NO
```

in the maketpf.cfg file. If NO is specified, the resulting PDS names would appear as:

```
ACP.SYMACRO.MTPFSRC
```

However, this will limit you to building only one system or subsystem in the specified set of PDSs.

## 2.0 PREREQUISITES

This sample build requires the MakeTPF41 package, available for download at: <http://www.ibm.com/tpf/download/maketpf.htm>

Performing a full build using these procedures will require approximately 2.0Gb of hfs space.

## 3.0 SETUP INSTRUCTIONS

1. Create the TPF 41 product source code directory in the hfs. For this build setup, we have isolated the source files into a separate directory from where all the generated/build files will reside.
  1. `mkdir /u/tpf41/mtpfsrc`
2. Populate the source code directory with the contents of the TPF product source code. Copy the source code only -- do not copy the binaries, as the intent of this build is to show the binaries can be generated by MakeTPF41. For those binaries that cannot be generated by MakeTPF41 (like OCO objects), they will be copied later in this setup procedure. A put level source tar file is provided with the TPF 41 PUT tapes, and that may be the best option to obtain a level of code to build with. The resulting hfs structure should appear as:
 

```
/u/tpf41/mtpfsrc/rt/base
/u/tpf41/mtpfsrc/macro
/u/tpf41/mtpfsrc/include
etc.
```
3. Install the MakeTPF41 makefiles and maketpf.cntl file into the mtpfsrc directory:
  1. `cd /u/tpf41/mtpfsrc`
  2. `pax -rf MakeTPF41_TPF41_Makefiles.tar.Z -ofrom=ISO8859-1,to=IBM-1047`  
 Once unpacked, the directory structure should look as follows:
 

```
/u/tpf41/mtpfsrc/base
/u/tpf41/mtpfsrc/cb2b
/u/tpf41/mtpfsrc/cdec
/u/tpf41/mtpfsrc/cp
/u/tpf41/mtpfsrc/debug
/u/tpf41/mtpfsrc/oco
/u/tpf41/mtpfsrc/ol
/u/tpf41/mtpfsrc/openssl-0.9.6
/u/tpf41/mtpfsrc/rt
/u/tpf41/mtpfsrc/tpf_mail
```
4. Create the TPF 41 target directory. This will contain all of the objects, listings, etc., generated during the build.
  1. `mkdir /u/tpf41/mtpfout`
5. Create the build directories. These will contain the maketpf.cfg file and the .out and .err files generated while running the bldtpf and maketpf commands. For this sample build, we will run a BSS system build. By convention, we've used the system name as the output directory name:
  1. `mkdir -p /u/tpf41/mtpfout/build/bss`

6. Set up the maketpf.cfg file:
  1. Edit /u/tpf41/mtpfout/build/bss/maketpf.cfg, adding:
    1. TPF\_ROOT := /u/tpf41/mtpfout
    2. TPF\_ROOT += /u/tpf41/mtpfsrc
    3. TPF\_ROOTPDS := ACP.\*.MTPFOUT
    4. TPF\_ROOTPDS += ACP.\*.MPTFSRC
    5. TPF\_BSS\_NAME := BSS
7. Allocate the source PDSs needed for the build. The following PDSs must be allocated, even though some will remain empty. This is because all referenced PDSs must exist, or the "ld" link commands will fail with errors related to being unable to allocate datasets. It is suggested you use your current system PDSs as the template for sizes.
  1. ACP.SYSRCE.MTPFSRC.BSS
    1. This dataset will contain the SIP generated copy files for the BSS system.
  2. ACP.SYMACRO.MTPFSRC.BSS
    1. This dataset will contain the SIP generated macros for the BSS system.
  3. ACP.MACRO.TPFMOD.MTPFSRC
    1. This dataset can be omitted if MAKETPF\_USE\_LOCAL\_MODS := NO is coded in the maketpf.cfg file. If YES is coded (the default) then the dataset must be created and is intended to contain customer mods to TPF macros.
  4. ACP.MACRO.MTPFSRC
    1. This dataset will contain the TPF macros.
  5. ACP.MACRO.TPFDF.MTPFSRC
    1. This dataset can be omitted if DF is not installed and the SIP macro specifies SBTPDF=NO.
    2. This dataset will contain the TPF DF macros and copy files.
  6. ACP.SIPGEN.MTPFSRC
    1. This dataset contains the SIP macros (needed for some offline program assemblies).
  7. ACP.SRCE.CP.MTPFSRC
    1. This dataset will contain the CP copy files.
  8. ACP.SRCE.RT.MTPFSRC
    1. This dataset will contain the RT copy files.
  9. ACP.CLIB.MTPFSRC
    1. This dataset will contain TPF DF clib entries that cannot be built by maketpf.
    2. If TPFDF is not installed, this dataset will be empty. All TPF CLIBs will be generated by maketpf and will be written to ACP.CLIB.MTPFOUT.
  10. ACP.STUB.MTPFSRC
    1. This dataset will be empty. All TPF STUBs will be generated by maketpf and will be written to ACP.STUB.MTPFOUT.
  11. ACP.OBJ.MTPFSRC.BSS
    1. This dataset will be empty.
  12. ACP.LINK.MTPFSRC.BSS
    1. This dataset will be contain TPF MVS offline programs needed during the build.
8. Allocate the target PDSs. As with the source datasets, all of the following PDSs must be allocated, even though some will remain empty. Again, this is because all referenced PDSs must exist, or commands will fail and it is suggested you use your current system PDSs as the template for sizes.
  1. ACP.SYSRCE.MTPFOUT.BSS
    1. This dataset will be empty.
  2. ACP.SYMACRO.MTPFOUT.BSS
    1. This dataset will be empty.
  3. ACP.MACRO.TPFMOD.MTPFOUT
    1. This dataset can be omitted if MAKETPF\_USE\_LOCAL\_MODS := NO is coded in the maketpf.cfg file. If YES is coded (the default) then the dataset must be created but it will remain empty.
  4. ACP.MACRO.MTPFOUT
    1. This dataset will be empty.

5. ACP.MACRO.TPFD.F. MTPFOUT
  1. This dataset can be omitted if DF is not installed and the SIP macro specifies SBTPFD=NO. If created, it will remain empty.
6. ACP.SRCE.CP. MTPFOUT
  1. This dataset will be empty.
7. ACP.SRCE.RT. MTPFOUT
  1. This dataset will be empty.
8. ACP.CLIB. MTPFOUT
  1. This dataset will contain TPF CLIB entries generated during the build.
9. ACP.STUB. MTPFOUT
  1. This dataset will contain TPF STUBs generated during the build.
10. ACP.OBJ. MTPFOUT.BSS
  1. This dataset will contain the C Load module linkage objects (CSTRTD40, CSTRTL40, and CSTDLL40). These objects must reside in a PDS for the ld command to run successfully.
11. ACP.LINK. MTPFOUT.BSS
  1. This dataset will remain empty during the build, but can be populated for loading.

9. Edit the file named "include41/maketpf.rules\_env\_pds\_all", found under the tools directory where the maketpf tools are installed. Update the SYS\_SUFFIX list to include the PDS qualifiers you are using to the list. For this sample build, we have added:

```
SYS_SUFFIX += .MTPFSRC
```

```
SYS_SUFFIX += .MTPFOUT
```

The SYS\_SUFFIX list defines the PDS naming conventions that represent "system" level datasets, as opposed to user level datasets. The difference being that fewer datasets are allocated at the user level, than at the system level. See the maketpf installation instructions for additional information.

10. Populate OCO objects in the source hfs: /u/tpf41/mtpfsrc/obj/oco
  1. Create the directory:
    1. mkdir -p /u/tpf41/mtpfsrc/obj/oco
  2. Copy from your system OCO OBJ PDS (for example, ACP.OBJ.OCO40.PUT19)
    1. cd /u/tpf41/mtpfsrc/obj/oco ; cp "'/'acp.obj.oco.put19"
    2. Rename XXXXXX40 to xxxxxx40.o
    3. typeset -l n ; for N in \* ; do n=\$N ; mv \$N \$n.o ; done
11. If TPF DF is installed, populate the TPFD include files in the source hfs: /u/tpf41/bld41/mtpfsrc/tpfdf/include
  1. Create directory:
    1. mkdir -p /u/tpf41/mtpfsrc/tpfdf/include
  2. Copy from your TPF DF CHDR PDS, for example:
    1. cd /u/tpf41/mtpfsrc/tpfdf/include ; cp "'/'acp.chdr.tpfdf.put20" .
  3. Ensure you rename XXXXXXXX to xxxxxxx.h and rename x@xxxxx.h x\_xxxxx.h
    1. typeset -l n ; for N in \* ; do n=\$(echo \$N | tr '@' '\_' ) ; mv \$N \$n.h ; done
12. Populate CLIB PDS with TPF DF CLIBs.
  1. Copy from your system CLIB PDS. At the time this document was written, the following CLIBs were found to be needed by the TPF system build. For simplicity, all @DB\* files can be copied.
    1. @DBOPN
    2. @DBDEL
    3. @DBADD
    4. @DBCLS
    5. @DBKEY
    6. @DBRED
    7. @DBIFB
    8. @DBDSP
    9. @DBCRE
    10. @CBREP
  2. Copy to: ACP.CLIB.MTPFSRC
13. Populate the SIP generated source files in hfs
  1. Create the directories:
    1. mkdir -p /u/tpf41/mtpfsrc/sysrce/bss

2. Copy from your SYSRCE PDS, for example:
  1. `cd /u/tpf41/mtpfsrc/sysrce/bss ; cp "'/acp.sysrce.rlse.bss"` .
3. Rename XXXXXXXX to xxxxxxxx.asm
  1. `typeset -l n ; for N in * ; do n=$N ; mv $N $n.asm ; done`
14. Populate the SIP generated include files in hfs
  1. Create the directories:
    1. `mkdir -p /u/tpf41/mtpfsrc/sychdr/bss`
  2. Copy from your SYSCHDR PDS, for example:
    1. `cd /u/tpf41/mtpfsrc/sychdr/bss ; cp "'/acp.sychdr.rlse.bss"` .
  3. Rename XXXXXXXX to xxxxxxxx.h
    1. `typeset -l n ; for N in * ; do n="$N" ; mv "$N" "$n.h" ; done`
15. Promote TPF macros from hfs to the MTPFSRC PDSs (required by hlasm)
  1. Copy from: `/u/tpf41/mtpfsrc/macro`
  2. To: ACP.MACRO.MTPFTEST
    1. `cd /u/tpf41/mtpfsrc/macro ; for n in * ; do cp $n "'/acp.macro.mtpfsrc(${n%%.*})"` ; done
16. Promote TPF SIP macros from hfs to the MTPFSRC PDS (required by hlasm).
  1. Copy from: `/u/tpf41/mtpfsrc/macro/sip`
  2. To: ACP.SIPGEN.MTPFSRC
    - for n in \* ; do cp \$n "'/acp.sipgen.mtpfsrc(\${n%%.\*})"
17. Promote the TPF local mod macros from hfs to PDS (if any)
  1. From: `/u/tpf41/mtpfsrc/local_mod/macro`
  2. To: ACP.MACRO.TPFMOD.MTPFSRC
    - for n in \* ; do cp \$n "'/acp.macro.tpfmod.mtpfsrc(\${n%%.\*})"
18. Promote the CP CPY files from hfs to PDS
  1. From:
    - `/u/tpf41/mtpfsrc/cp/base`
    - `/u/tpf41/mtpfsrc/cp/mpif`
    - `/u/tpf41/mtpfsrc/cp/hpo`
  2. To: ACP.SRCE.CP.MTPFSRC
    - `cd /u/tpf41/mtpfsrc/cp`
    - `find . -name "*.cpy" | while read n ; do m=${n##*/} ; m=${m%%.*} ;`
    - `cp $n "'/acp.srce.cp.mtpfsrc($m)"` ; done
19. Promote the RT CPY files from hfs to PDS
  1. From `/u/tpf41/mtpfsrc/rt/base`
  2. To: ACP.SRCE.RT.MTPFSRC
    - `cd /u/tpf41/mtpfsrc/rt`
    - `find . -name "*.cpy" | while read n ; do m=${n##*/} ; m=${m%%.*} ;`
    - `cp $n "'/acp.srce.rt.mtpfsrc($m)"` ; done
20. Promote the local\_mod copy files from hfs to PDS (if any)
  1. From: `/u/tpf41/mtpfsrc/local_mod`
  2. To: ACP.MACRO.TPFMOD.MTPFSRC
    - `cd /u/tpf41/mtpfsrc/local_mod`
    - `find . -name "*.cpy" | while read n ; do m=${n##*/} ; m=${m%%.*} ;`
    - `cp $n "'/acp.macro.tpfmod.mtpfsrc($m)"` ; done
21. Populate the SIP generated macros in your MTPFSRC PDSs:
  1. Copy all members from your system level SYMACRO PDS
  2. To: ACP.SYMACRO.MTPFSRC.BSS
22. Populate the MTPFSRC LINK PDS with the offline programs needed for build:
  1. Copy the following member from your system LINK PDS
    1. DB2PP
  2. Copy to: ACP.LINK.MTPFSRC.BAS
23. If you have TPFDF installed, populate the TPF DF macros in your MTPFSRC PDSs
  1. Copy all members from your system level TPFDF macro PDS
  2. To: ACP.MACRO.TPFDF.MTPFSRC
24. Populate `/u/tpf41/mtpfsrc/export` directory with the export files that cannot currently be generated by maketpf (CXML, CTLF, CDWF). These are required for linking other TPF programs, so they must be added to the hfs.
  1. From your existing system IMPORTS PDS
  2. To: `/u/tpf41/mtpfsrc/export`
    1. The file names in hfs must be changed to: CXML.x, CTLF.x, and CDWF.x
    2. The files must be copied to hfs in "binary" mode, to avoid truncation of some lines that exceed 80 characters.
    3. Note that the .x files may exist in your existing system hfs and can

be copied from there if they do, but that depends on if you are at PUT19 or later.

25. Ensure the function switch variables used by MakeTPF to match your system configuration. This should have been done as part of the MakeTPF41 tools installation, but is included again here for completeness.

The function switches are used in the MakeTPF control file to indicate whether or not programs are to be included in the build, based on the value of the function switch. A default function switch of TPF\_SBALL indicates the program is always built. A specific function switch, like TPF\_SBTDFDF, indicates that a program should only be built, if the value of the function switch is YES. These switches are similar to the FUNC= coding found in the SPPGML macro.

The value of the switches can be set in either of the two ways described below. The values for each of the switches on your system can be obtained from the SIP generated header file named "c\$func.h". For each switch listed in that header, there is a corresponding variable in MakeTPF, with a TPF\_ prefix. For example, if TPFDF is enabled on your system, the entry in the header file for the TPFDF switch would appear as:

```
"SBTDFDF",1,
```

In MakeTPF, this must be mapped to a variable as follows (noting the prefix of TPF\_):

```
TPF_SBTDFDF := YES
```

The supported values for each MakeTPF switch are NO (if 0 in the header) and YES (if 1 in the header).

To modify the values known to MakeTPF, you can use either of the following methods.

1. Edit

**<installpath>/include41/maketpf.rules\_functionswitch\_defaults**

and update the default value of the function switches in that file. For example, modify the TPF\_SBTDFDF setting to be YES, if you have the TPFDF code installed on your system. This will cause the MACRO.TPFDF PDS to be included in the MACLIB for assemblies. The default value is NO.

2. Edit **<installpath>/include41\_user/<sys>.function\_switches** and set the values for each of the function switches in that file.

#### 4.0 BUILD INSTRUCTIONS

To complete a full build of the TPF 41 code provided in the MakeTPF41\_TPF41\_Makefiles package:

1. Change to the working build directory. All .out and .err files generated by the MakeTPF tools will be written to this directory:  
cd /u/tpf41/mtpfout/build/bss
2. Ensure the maketpf and bldtpf commands are installed and in your PATH.
  1. Issue the command:  
whence maketpf  
This should return the location where the maketpf command is installed. If it does not, add the MakeTPF41 tools directory to the PATH:  
PATH=<MakeTPF41 installdir>/tools:\$PATH  
and then run the whence command again to verify.
3. To build all real-time, CIMR, KPT, and CP programs, issue the following build command. This will run the bldtpf system build against the control file, redirecting error and output messages to a file named bldtpf\_bss.out, and will simultaneously echo those messages to the screen:  
bldtpf -tpf /u/tpf41/mtpfsrc/base/cntl/tpf.cntl 2>&1 | tee bldtpf\_bss.out
4. To build all offline programs, issue the following build command. This will run the bldtpf system build against the control file, redirecting error and output messages to a file named bldtpf\_bss\_offline.out, and will simultaneously echo those messages to the screen:  
bldtpf -ol /u/tpf41/mtpfsrc/base/cntl/tpf.cntl 2>&1 | tee bldtpf\_bss\_offline.out

#### 5.0 NOTICES

IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product,

program, or service. IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Department 830A  
Mail Drop P131  
2455 South Road  
Poughkeepsie, NY 12601-5400  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee. Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

## **5.1 Trademarks**

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM  
MVS  
OS/390

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.