The MVS 3.8j Tur(n)key 4- System -- Version 1.00 -- Update 08

Installation

1. It is strongly recommended to create a backup copy of the system. Although the update process has been thoroughly tested, a backup copy comes in handy if an unforeseen error occurs.
2. Update 01, Update 02, Update 03, Update 04, Update 05, Update 06 and Update 07 are prerequisites for Update 08. Make sure Update 01, Update 02, Update 03, Update 04, Update 05, Update 06 and Update 07 have been installed successfully before trying to install Update 08.
3. Make sure that your `tk4-` folder does not contain a subfolder named `update`. If an `update` folder has been left over from the installation of a previous update (i.e. from Update 07) either delete it or rename it.
4. Unzip archive `tk4-_v1.00_update_08.zip` into the `tk4-` folder. Allow your unzip utility to overwrite existing files and to merge into existing folders while unzipping the archive.
5. a) Windows: Open folder `tk4\update` and click (or double click, depending on your settings) `apply_update.bat`.

   b) Linux or OS X: Open a shell window, change directory to the `tk4-/update` folder and run `./apply_update`.
6. You'll be prompted for the credentials of an administrative user (i.e. `HERC01/CUL8TR`), then the system will be IPLed, an update job will be executed and the system will be shut down. Because this update changes `SYS1.LPALIB` the system will then be IPLed a second time to rebuild the link pack area. This second IPL will be followed by an immediate shutdown, concluding the update process.

   Note: Although lots of informational message are displayed during the update process most of the time, there may be update steps that will not display any messages for a couple of minutes. In particular, step "`EXHREST UNZIP`" can take very long on low performance host systems (ARM et al). This must not be misinterpreted as a stall of the update process. Please be patient during these pauses and refrain from manually interrupting the update process.
7. Once the update process has finished check `listing.txt` in the `update` folder for errors. One of the following outcomes is expected:

```
07.55.10 JOB    1  IEF403I UPDATER - STARTED - TIME=07.55.10
07.55.10 JOB    1  IEFACTRT - Stepname  Procstep  Program   Retcode
07.55.10 JOB    1  UPDATER    DELCAT              IEFBR14   RC= 0000
07.55.10 JOB    1  UPDATER    DELVOL              IEFBR14   RC= 0000
07.55.10 JOB    1  UPDATER    ALLOC               IEFBR14   RC= 0000
07.55.10 JOB    1  UPDATER    CHKDONE             IDCAMS    RC= 0004
07.55.10 JOB    1  UPDATER    PREPRC1             IEBCOPY   RC= 0000
07.55.10 JOB    1  UPDATER    PREPRC2             IEFBR14   RC= 0000
07.55.10 JOB    1  UPDATER    PREPRC3             MAWK      RC= 0000
07.55.10 JOB    1  UPDATER    PREPRC4             IEBGENER  RC= 0000
07.55.10 JOB    1  UPDATER    PREPRC5             IEBUPDTE  RC= 0000
07.55.16 JOB    1  UPDATER    PREPRC6             IKJEFT01  RC= 0000
07.55.17 JOB    1  UPDATER    PREPRC7             IKJEFT01  RC= 0000
07.55.17 JOB    1 *IEC501A M 480,UPDATE,SL,6250 BPI,UPDATER,CREDITS
07.55.17 JOB    1  UPDATER    CREDITS             IEBGENER  RC= 0000
07.55.18 JOB    1  UPDATER    EXHIBIT             IEBGENER  RC= 0000
07.55.18 JOB    1  UPDATER    IMON370             IEBGENER  RC= 0000
07.55.18 JOB    1  UPDATER    REVIEW              IEBGENER  RC= 0000
07.55.19 JOB    1  UPDATER    VFPRINTF            IEBGENER  RC= 0000
07.55.19 JOB    1  UPDATER    AFPCNTRL            IEBGENER  RC= 0000
07.55.19 JOB    1  UPDATER    XFERPTCH            IEBGENER  RC= 0000
07.55.19 JOB    1  UPDATER    FTPDRAC             IEBGENER  RC= 0000
07.55.19 JOB    1  UPDATER    MVSDDT              IEBGENER  RC= 0000
07.55.20 JOB    1  UPDATER    SXMACLIB            IEBGENER  RC= 0000
07.55.20 JOB    1  UPDATER    INDFILE             IEBGENER  RC= 0000
07.55.33 JOB    1  UPDATER    ELEMENTS            IEBCOPY   RC= 0000
07.55.33 JOB    1  UPDATER    POSTPRC1            IEBCOPY   RC= 0000
07.55.33 JOB    1  UPDATER    POSTPRC2            IDCAMS    RC= 0000
07.55.33 JOB    1  UPDATER    EXHREST   DELCAT    IEFBR14   RC= 0000
07.55.34 JOB    1  UPDATER    EXHREST   DELVOL    IEFBR14   RC= 0000
07.56.56 JOB    1  UPDATER    EXHREST   UNZIP     MINIUNZ   RC= 0000
```

```
07.57.06 JOB    1  UPDATER    EXHREST   RECV370   RECV370   RC= 0000
07.57.10 JOB    1  UPDATER    EXHREST   DSSREST   DSSREST   RC= 0000
07.57.10 JOB    1  UPDATER    NUCCHECK            IKJEFT01  RC= 0020
07.57.11 JOB    1  UPDATER    NUCBACK             IEBCOPY   RC= 0000
07.57.28 JOB    1  UPDATER    RESTORE   HMASMP    HMASMP    RC= 0004
07.57.28 JOB    1  UPDATER    SVCASM              IFOX00    RC= 0000
07.57.29 JOB    1  UPDATER    SVCLINK             IEWL      RC= 0004
07.57.29 JOB    1  UPDATER    STEP1               IEBGENER  RC= 0000
07.57.31 JOB    1  UPDATER    STEP2               IFOX00    RC= 0000
07.57.31 JOB    1  UPDATER    STEP3               IEBGENER  RC= 0000
07.57.32 JOB    1  UPDATER    STEP4               IFOX00    RC= 0000
07.57.32 JOB    1  UPDATER    STEP5               IEBGENER  RC= 0000
07.57.32 JOB    1  UPDATER    STEP6     HMASMP    HMASMP    RC= 0000
07.57.38 JOB    1  UPDATER    STEP7     HMASMP    HMASMP    RC= 0004
07.58.22 JOB    1  UPDATER    CLEANUP             IEBCOPY   RC= 0000
07.58.22 JOB    1  IEF234E K 480,UPDATE,PVT,UPDATER
07.58.22 JOB    1  IEF404I UPDATER - ENDED - TIME=07.58.22
```

-- or --

```
08.05.11 JOB    2  IEF403I UPDATER - STARTED - TIME=08.05.11
08.05.12 JOB    2  IEFACTRT - Stepname  Procstep  Program   Retcode
08.05.12 JOB    2  UPDATER    DELCAT              IEFBR14   RC= 0000
08.05.12 JOB    2  UPDATER    DELVOL              IEFBR14   RC= 0000
08.05.12 JOB    2  UPDATER    ALLOC               IEFBR14   RC= 0000
08.05.12 JOB    2  UPDATER    CHKDONE             IDCAMS    RC= 0000
08.05.12 JOB    2  UPDATER    PREPRC1             IEBCOPY   *FLUSH*
08.05.12 JOB    2  UPDATER    PREPRC2             IEFBR14   *FLUSH*
08.05.12 JOB    2  UPDATER    PREPRC3             MAWK      *FLUSH*
08.05.12 JOB    2  UPDATER    PREPRC4             IEBGENER  *FLUSH*
08.05.12 JOB    2  UPDATER    PREPRC5             IEBUPDTE  *FLUSH*
08.05.12 JOB    2  UPDATER    PREPRC6             IKJEFT01  *FLUSH*
08.05.13 JOB    2  UPDATER    PREPRC7             IKJEFT01  RC= 0000
08.05.13 JOB    2 *IEC501A M 480,UPDATE,SL,6250 BPI,UPDATER,CREDITS
08.05.13 JOB    2  UPDATER    CREDITS             IEBGENER  RC= 0000
08.05.14 JOB    2  UPDATER    EXHIBIT             IEBGENER  RC= 0000
08.05.14 JOB    2  UPDATER    IMON370             IEBGENER  RC= 0000
08.05.14 JOB    2  UPDATER    REVIEW              IEBGENER  RC= 0000
08.05.14 JOB    2  UPDATER    VFPRINTF            IEBGENER  RC= 0000
08.05.15 JOB    2  UPDATER    AFPCNTRL            IEBGENER  RC= 0000
08.05.15 JOB    2  UPDATER    XFERPTCH            IEBGENER  RC= 0000
08.05.15 JOB    2  UPDATER    FTPDRAC             IEBGENER  RC= 0000
08.05.15 JOB    2  UPDATER    MVSDDT              IEBGENER  RC= 0000
08.05.15 JOB    2  UPDATER    SXMACLIB            IEBGENER  RC= 0000
08.05.15 JOB    2  UPDATER    INDFILE             IEBGENER  RC= 0000
08.05.27 JOB    2  UPDATER    ELEMENTS            IEBCOPY   RC= 0000
08.05.27 JOB    2  UPDATER    POSTPRC1            IEBCOPY   *FLUSH*
08.05.27 JOB    2  UPDATER    POSTPRC2            IDCAMS    *FLUSH*
08.05.27 JOB    2  UPDATER    EXHREST   DELCAT    IEFBR14   RC= 0000
08.05.27 JOB    2  UPDATER    EXHREST   DELVOL    IEFBR14   RC= 0000
08.06.47 JOB    2  UPDATER    EXHREST   UNZIP     MINIUNZ   RC= 0000
08.06.58 JOB    2  UPDATER    EXHREST   RECV370   RECV370   RC= 0000
08.07.02 JOB    2  UPDATER    EXHREST   DSSREST   DSSREST   RC= 0000
08.07.02 JOB    2  UPDATER    NUCCHECK            IKJEFT01  RC= 0000
08.07.02 JOB    2  UPDATER    NUCBACK             IEBCOPY   *FLUSH*
08.07.14 JOB    2  UPDATER    RESTORE   HMASMP    HMASMP    RC= 0004
08.07.14 JOB    2  UPDATER    SVCASM              IFOX00    RC= 0000
08.07.15 JOB    2  UPDATER    SVCLINK             IEWL      RC= 0004
08.07.16 JOB    2  UPDATER    STEP1               IEBGENER  RC= 0000
08.07.17 JOB    2  UPDATER    STEP2               IFOX00    RC= 0000
08.07.17 JOB    2  UPDATER    STEP3               IEBGENER  RC= 0000
08.07.18 JOB    2  UPDATER    STEP4               IFOX00    RC= 0000
08.07.18 JOB    2  UPDATER    STEP5               IEBGENER  RC= 0000
08.07.18 JOB    2  UPDATER    STEP6     HMASMP    HMASMP    RC= 0000
08.07.24 JOB    2  UPDATER    STEP7     HMASMP    HMASMP    RC= 0004
08.08.04 JOB    2  UPDATER    CLEANUP             IEBCOPY   RC= 0000
08.08.05 JOB    2  IEF234E K 480,UPDATE,PVT,UPDATER
08.08.05 JOB    2  IEF404I UPDATER - ENDED - TIME=08.08.05
```

IPL the system and verify that your regularly used functionality still works as expected. If it does not, revert to your backup copy and report the problems to the author.

Note 1: The update job adds `EXH.EXHLIB` and `EXH.ESPLIB` on `PUB012` to the list of APF authorized program libraries in `SYS1.PARMLIB(IEAAPF00)`. To provide recoverability, your original `IEAAPF00` list is backed up to `SYS1.PARMLIB.PREUPD08`. Should the automated editing of the list unexpectedly fail, restore your original `IEAAPF00` list and add the two libraries mentioned above manually. `SYS1.PARMLIB.PREUPD08` is not needed for system operations and can safely be deleted, once the contents of `SYS1.PARMLIB(IEAAPF00)` has been verified.

Note 2: Steps `SVCLINK`, `RESTORE` and `STEP7` of the update job modify the system nucleus. To provide recoverability, your original `SYS1.NUCLEUS` library is backed up to `SYS1.NUCLEUS.PREUPD08`. Should the system fail to IPL or otherwise fail to work as expected, reinstate `SYS1.NUCLEUS.PREUPD08`. This should bring the system back to normal operations, but of course at the cost of losing the nucleus changes. Report the results of the `apply_update` run to the author. `SYS1.NUCLEUS.PREUPD08` is not needed for system operations and can safely be deleted, once the integrity of the new nucleus has been verified.

8. Folder `tk4-/update` is not needed to operate the system, once the update has been installed successfully. It is recommended to remove it to avoid interference with future updates.

Fixes

Arbitrary Hercules Aborts During Shutdown Of Dual CPU Systems

On high performance Linux hosts with glibc 2.3.4 or newer and 6 or more CPUs/cores/hyperthreads, Hercules arbitrarily aborts while performing the automated shutdown of a dual processor TK4- MVS 3.8j system. The error is triggered by three `sysclear` commands being issued in very quick succession by the automated shutdown procedure on a dual processor system, which, given enough parallelism provided by the host, leads to multiple frees of the same storage area. The default behavior of glibc in this situation changed with version 2.3.4 from “print a short message and continue” to “print a detailed message and abort”.

The automated shutdown procedure was changed to issue one `sysclear` command only, on single as well as on dual CPU systems. Intensive testing showed that the error doesn’t occur anymore when using the changed shutdown procedure. The root cause of this error most probably is related to thread synchronization, which would need further investigation if it turned out that the changed shutdown procedure doesn’t reliably prevent it from occurring.

It should be noted that setting the environment variable M_CHECK_ACTION to 1 reverts glibc to the pre 2.3.4 behavior, thus also preventing the error from aborting Hercules. While in most cases nothing bad happens when Hercules aborts after having shut down MVS, this cannot be taken for granted, as the abort happens **before** Hercules closes the DASD files. As such, ignoring the error introduces is a certain data corruption risk. For that reason, if you regularly experience this error, please report it to the author.

Web Server Segmentation Faults

A segmentation fault occurring in Hercules when a 3705 Communications Controller was displayed using the web console’s “display” link was fixed in `comm3705.c`.

Segmentation Faults Using NUMCPU=2 On 32-Bit OS X Snow Leopard

This problem was caused by using an invalid build procedure used to create the 32-bit binaries for OS X in Update 07. The problem was solved by using the correct procedure to build the 32-bit OS X binaries distributed with Update 08.

Error 49 When Binding To Specific Interface On OS X Snow Leopard

The reason for this problem was a slightly different behavior of earlier BSD systems (and their forks, like OS X) than Linux, Windows and later BSD systems, when it comes to using a sockaddr structure to bind a socket to a specific interface. Basically, on the older systems, one should initialize all unused fields of the structure to zero before using it. This was fixed for the OS X specific Hercules builds in `tcpip.c`.

Arbitrary Connection Failures On FTPD Data Connections In Active Mode

This problem was caused by a logic flaw in FTPD: If, after having issued the PORT command, the ftp client manages to be faster sending the relevant transfer command (LIST, RETR or STOR), than the daemon to complete the connection to the port given, the

transmission may get started although there is no active connection (yet). This is strictly timing dependent and thus could occur on any host OS and in any topology. A patch for `ftpd.c` was created and is applied to the FTPD version distributed with Update 08.

Incorrect Output From JCC Compiled C Programs When Using %p To Print Pointers

Using %p in printf format strings yielded arbitrary output (usually zero). This was a bug in the JCC library. Thanks to Jason Winter for providing a fix!

New or Changed Function

Enhanced Parameterization

The following environment variables have been introduced:

Variable	Default	Description
S37X	#	Set to `ldmod s37x` to enable selected XA, ESA and z/Arch instructions
DYNCRYPT	#	Set to `ldmod dyncrypt` to enable selected z/Arch cryptographic instructions
FSYNC	0	Disable (0) or enable (1) DASD file synchronization with the host's filesystem
GCINT	10	DASD garbage collection interval in seconds

Setting `S37X` or `DYNCRYPT` to the `ldmod` commands shown above enables the instruction set extensions described in section "Instruction Set Extensions". The default setting of `#` executes a comment, resulting in a clean S/370 system without any instruction set extensions.

The `FSYNC` and `GCINT` environment variables control the DASD file synchronization with the host's file system and the garbage collection activity. Enabling `FSYNC` and setting `GCINT` to some low value increases robustness in the event of catastrophic failures (i.e. power outages or other host system crashes) at the cost of a potential performance penalty. The Hercules User Reference Guide recommends: "Specify `FSYNC=1` and `GCINT=5` if you are seriously concerned about your data being lost due to a failure. `FSYNC` will ensure your data on disk is coherent. However, `FSYNC` may cause noticeable performance degradation. Note that an `FSYNC` will not be performed more often than every 5 seconds." These parameters will be applied to the primary TK4- DASD (as defined in conf/tk4-.cnf) only, not to the optional CBT and source DASD, or to other locally configured DASD. The introduction of the `FSYNC` and `GCINT` environment variables is the result of a proposal by David Jackson; thanks for the idea!

For information on how to permanently set environment variables for use with TK4- see `README_MVS_TK4-_v1.00_update_01.pdf` in the `tk4-/doc` folder.

CTCE -- Full Function CTC Adaptor

The Hercules version that comes with TK4- ("TK4- Hercules") has been updated to support the new CTCE device developed by Peter Jansen. The CTC adaptor implementations available until now are only able to support TCP/IP related payloads, they lack the signaling capabilities necessary to support classic 3088 payloads like NJE, SNA, JES3 coupling, XCF, et al. Here the CTCE device comes into play: It supports most of the non TCP/IP related CTC adaptor payloads and thus enables systems to use all the well-known communication methods like JES2 NJE or SNA/NJE, SNA LU 6.x with ACF/VTAM, JES3 clusters, etc.

However, with the exception of JES3, none of these methods are currently available on MVS 3.8j. So, the CTCE device is only one side of the medal: For the first time it makes the re-implementation of some of these communication methods on MVS 3.8j *possible*. But unless someone steps in to *really do it*, there still isn't much use for the CTCE functionality.

Presumably, implementing CTCA NJE support into JES2 would be relatively easy to do, as there is no other missing software, i.e. it is mere coding inside JES2. All the SNA related

stuff, however, would require refurbishing VTAM to add ACF level functionality, which probably comes close to a complete rewrite and is as such much more difficult than just a “simple” NJE implementation in JES2.

In absence of a currently existing CTCE use case on the TK4- MVS 3.8j JES2 system itself, a standalone verification program exercising very basic CTCA functionality has been created. It is completely unrelated to the TK4- MVS 3.8j system and can be found in folder `ctca_demo` of the TK4- distribution.

Thanks to Peter Jansen for making the CTCE device available!

MVSDDT 4.0

MVSDDT Server Version 2.4.1 has been replaced by version 4.0. MVSDDT Version 4.0 is required to be installed on the client system to use the new server version. It can be downloaded from http://mvsddt.altervista.org or from dataset TK4-.SHELBY.MVSDDT.V400.ZIP. Thanks to Shelby Beach for providing this great debug tool! For further information see HELP member MVSDDT.

SXMACLIB

The new macro library SYS2.SXMACLIB is mainly intended to provide support for the instructions emulated by loading the Hercules s37x and dyncrypt modules when running in S/370 mode. For more information, please see the $$S37X member of SYS2.SXMACLIB or HELP member SXMACLIB.

To allow the structured programming macros provided with the original TK3 system to utilize the additional s37x instructions, changes were required to the structured programming macros. In general these changes should be transparent to the user. A few additional features have been implemented in the structured programming macros. These new features along with the capabilities provided by the original macros are fully documented in the $$SPDOC member of SYS2.SXMACLIB or HELP member SXMACLIB.

Note: To avoid conflicts with the TK3 versions of the structured programming macros, it is recommended to place SYS2.SXMACLIB before SYS2.MACLIB in the SYSLIB search order. SYS2.SXMACLIB replaces the previously supplied macro libraries SYS1.ZMACLIB and SYS2.Z9MACLIB, which are now aliases of the new library.

This library brings assembler programming on MVS 3.8j to an exciting new level. Thanks to Shelby Beach for providing SYS2.SXMACLIB!

RFE/REVIEW 46.6

RFE/REVIEW release 45.6 has been replaced with release 46.6. Thanks to Greg Price for maintaining and continuously improving RFE/REVIEW and thus bringing capabilities similar to those of ISPF/PDF to MVS 3.8j! For further information see HELP member RFE.

IMON/370 Build 16.01.02

IMON/370 build 12.01.07 has been replaced with build 16.01.02. Thanks to Greg Price for bringing his famous system monitor to MVS 3.8j! For further information, press PF1 while in IMON/370 to read the help pages.

ZP60023 Rework 2016-08-06

The primary purpose of usermod ZP60023 is to provide DAS at the task level through an update of the program check first level interrupt handler (PCFLIH). Previous reworks, however, also hooked into the PCFLIH to emulate a few frequently used XA, ESA and z/Arch instructions. All of these instructions are now emulated on the TK4- Hercules layer when the s37x dynamic module is loaded. Thus the instruction emulation part of ZP60023 is no longer needed and was removed. Thanks to Greg Price for providing MVS 3.8j DAS capabilities!

Note: The previously installed rework 2012-01-07 is retained as SYS1.UMODCNTL(ZP60023@). If the MVS component of TK4- Update 08 is to run on a Hercules platform not providing full s37x support (namely the Spinhawk and Hyperion lines), ZP60023@ can be applied instead of ZP60023 to reactivate the PCFLIH based instruction emulation support.

Additional Type 3/4 SVCs

The original TK4- SVC table, which had been taken over from TK3 without modifications, has only one unused type 3 and one unused type 4 non APF, non preemptible SVC generated. These types are the most common ones used by application programs when it comes to performing authorized processing. Given that MVSDDT 4.0 takes the single free type 3 slot (SVC 233), some of the many unused type 2 slots were repurposed: SVCs 220-226 are newly generated as type 3, SVCs 227-229 are newly generated as type 4, which now gives a total of 10 additional type 3/4 slots. The source of this change is can be found in SYS1.SYSGEN.CNTL, member IOGEN. This member always holds the current configuration of the TK4- MVS 3.8j system.

EXHIBIT

The EXHIBIT system presents the operating system status at a glance and provides many useful functions to interactively modify about any setting one can think of. Thanks to Gerhard Postpischil for making EXHIBIT available to the MVS 3.8 community! For further information see HELP member EXHIBIT.

IND$FILE 2.0.5

IND$FILE release 1.1.1 has been replaced with release 2.0.5. Thanks to Mike Rayborn for making IND$FILE file transfer functionality for 3270 terminal emulations available to the MVS 3.8 community! For further information run the IND$FILE command without arguments from a TSO READY prompt, or from RFE or RPF menu 6.

RAC Based Authentication and Authorization for FTPD

To allow running the FTP daemon on internet accessible systems or on multi user systems at a reasonable risk, the most critical security weaknesses of Jason Winter's original FTPD implementation have been hardened using a minimalistic integration into the MVS Resource Access Control (RAC) framework. Given the considerable complexity it adds to the FTPD configuration, using this security enhanced FTPD version will not make much sense on the typical single user TK4- system, as long as it is not accessible from the internet. For that reason, TK4- Update 8 still comes with Jason's original unsecured FTPD being configured. HELP member FTPD-RAC has all the information needed to activate and configure the security enhanced version of FTPD, its source can be found under HLQ JCC.FTPD-RAC.

CHAT, A Socket Programming Example

The EMAIL program provided by Jason Winter as a socket programming example with his original distribution of the TCP/IP instruction performs an SMTP communication using data read from SYSIN. As most SMTP servers didn't require authentication back in 2002, this program was a meaningful example, allowing sending e-mail from MVS to an arbitrary recipient on the internet. Nowadays, however, no real e-mail server accepts SMTP dialogs without prior authentication. This basically makes the EMAIL program useless, even as an example.

Quite a few support questions around TCP/IP support showed that there is a need for a usable "low complexity" socket programming example (the "high complexity" one would of course still be FTPD). For that reason CHAT, a modification of the EMAIL program, was created by replacing the hardcoded SMTP dialog with an interactive dialog between a TSO terminal session and a network-cat tool (ncat, netcat, socat, etc.) running elsewhere. CHAT

is installed ready to use in SYS2.CMDLIB. Its source can be found in JCC.TCPIP.SRC. See HELP member CHAT for more details.

Enhanced JRP Printer Translate Table

Depending on the code pages in use by the 3270 terminal emulation and application programs like REVIEW, code point BA or AD is displayed as left bracket, and code point BB or BD is displayed as right bracket. The EBCDIC character set defines BA/BB as brackets, while some newer compilers (namely JCC and GCC) allow or even require using AD/BD. As delivered, the JRP utility prints only BA/BB as brackets, leading to ugly output when the file to be printed uses AD/BD for brackets. The JRP internal printer translate table in module JRP300 has been ZAPped using job SYS2.CNTL(JRP300$) to print both variants as brackets.

AFPCNTRL Utility -- Control AFP (Additional Floating Point) Register Availability

The S/370 architecture originally provides four floating point registers, numbered 0, 2, 4, and 6. Later architectures or features provide additional floating point (AFP) registers, numbered 1, 3, 5, 7, and 8 to 15. The new AFPCNTRL Utility can be used to enable or disable the AFP registers. Note that enabling the AFP registers on MVS 3.8j has a potential system integrity impact, when those registers are used by more than one job at a time. Thus it is strongly recommended to enable them only when needed and disable them after usage. See HELP member AFPCNTRL for details.

Instruction Set Extensions

TK4- Hercules has been updated to support over 300 instructions from newer architectures in S/370 mode. Together with the assembler support in SYS2.SXMACLIB this allows to assemble and run many programs written for newer systems (namely MVS/XA, MVS/ESA, OS/390 and z/OS) on MVS 3.8j, as long as they don't need any architectural features (namely AMODE 31 and ATL storage) or operating system support specific to the newer system. The same holds true for load modules created on these newer systems, i.e. many of them can just run out of the box on MVS 3.8j.

These instructions are enabled or disabled by loading or unloading dynamic modules s37x (everything except cryptographic instructions) and dyncrypt (cryptographic instructions only). As the presence of these instructions contradicts the original TK4- concept of providing a clean S/370 system, they are disabled by default. They must be explicitly enabled to use them.

To enable the instruction set extensions for the current TK4- run only, enter

```
ldmod s37x
```

and

```
ldmod dyncrypt
```

at the Hercules console or at the Hercules web console.

To enable the instruction set extensions permanently use the parameterization described in "Enhanced Parameterization".

Remarks on the instruction set extensions:

- Some of the floating point instructions that get enabled by the instruction set extensions require the AFP registers to be available, even if they don't use them. See "AFPCNTRL Utility -- Control AFP (Additional Floating Point) Register Availability" for information on the AFP registers.
- The instruction set extensions enable the EPSW instruction. It should be noted that this problem state instruction may yield unexpected or undesirable results when used

in a virtual machine under VM/370 (which, however, is out of scope for TK4-). Basically one could even argue, that making this instruction available in S/370 mode breaks the S/370 claim, that it can be 100% virtualized in software -- which surely isn't intended. This argument came in late. To prevent delays releasing TK4- Update 08, the ESPW instructions remains enabled for the time being. In case any problems would arise from this, a separate set of Hercules binaries will be provided which excludes the instruction. Should any need for both cases (instruction available or not available) come up, the presence of the instruction will be made externally configurable in a later TK4- Update.

The following table lists all instructions that get enabled by the instruction set extensions:

ADB	ADD (long BFP)
ADBR	ADD (long BFP)
AEB	ADD (short BFP)
AEBR	ADD (short BFP)
AFI	Add Immediate
AGSI	Add immediate long storage
AHI	Add Halfword Immediate
AHIK	add distinct halfword immediate
AHY	add halfword y
ALC	Add logical with carry
ALCR	Add logical with carry register
ALFI	Add Logical immediate
ALGSI	Add logical with signed immediate long
ALHSIK	add logical distinct signed halfword immediate
ALRK	add logical distinct register
ALSI	Add logical with signed immediate
ALY	add logical y
ARK	add distinct register
ASI	Add immediate storage
AXBR	ADD (extended BFP)
AY	add y
BASSM	Branch and Save and Set Mode
BRAS	Branch Relative and Save
BRASL	Branch Relative and Save Long
BRC	Branch Relative on Condition
BRCL	Branch Relative on Condition Long
BRCT	Branch Relative on Count
BRXH	Branch relative on index high
BRXLE	Branch relative on index low or equal
BSM	Branch and Set Mode
CDB	COMPARE (long BFP)
CDBR	COMPARE (long BFP)
CDFBR	Convert from fixed (32 to long BFP)
CDFR	Convert fixed to float long register
CEB	COMPARE (short BFP)
CEBR	COMPARE (short BFP)
CEFBR	Convert from fixed (32 to short BFP)
CEFR	Convert from fixed to float short register
CFC	Compare and form codeword
CFDBR	Convert to fixed (long BFP to 32)
CFDR	Convert from float long to fixed register
CFEBR	Convert to fixed (short BFP to 32)
CFER	Convert from float short to fixed register
CFI	Compare Immediate
CFXBR	Convert to fixed (extended BFP to 32)
CFXR	Convert from float extended to fixed register
CGHSI	Compare halfword immediate long storage
CHHSI	Compare halfword immediate halfword storage
CHI	Compare Halfword Immediate
CHRL	Compare halfword relative long
CHSI	Compare halfword immediate storage
CHY	compare halfword y
CIB	Compare immediate and branch
CIJ	Compare immediate and branch relative
CIT	Compare immediate and trap
CKSM	Checksum
CLCLE	Compare logical long extended
CLCLU	Compare logical long unicode
CLFHSI	Compare logical immediate fullword storage
CLFI	Compare Logical Immediate
CLFIT	Compare logical immediate and trap fullword
CLGHSI	Compare logical immediate long storage

CLHHSI	Compare logical immediate halfword storage
CLHRL	Compare logical halfword relative long
CLIB	Compare logical immediate and branch
CLIJ	Compare logical immediate and branch relative
CLRB	Compare logical and branch register
CLRJ	Compare logical and branch relative register
CLRL	Compare logical relative long
CLRT	Compare logical and trap register
CLST	Compare logical string
CLY	compare logical y
CMPSC	Compression call
CPSDR	copy sign fpr long reg
CRB	Compare and branch register
CRJ	Compare and branch relative register
CRL	Compare relative long
CRT	Compare and trap register
CSST	Compare and swap and store
CU14	Convert UTF-8 to UTF-32
CU24	Convert UTF-16 to UTF-32
CU41	Convert UTF-32 to UTF-8
CU42	Convert UTF-32 to UTF-16
CUSE	Compare until substring equal
CUTFU	Convert UTF-8 to unicode
CUUTF	Convert unicode to UTF-8
CVBY	convert to binary y
CVDY	convert to decimal y
CXBR	COMPARE (extended BFP)
CXFBR	Convert from fixed (32 to extended BFP)
CXFR	Convert from fixed to float extended register
CXR	Compare floating point extended register
CY	compare y
DDB	Divide (long BFP)
DDBR	Divide (long BFP)
DEB	Divide (short BFP)
DEBR	Divide (short BFP)
DIDBR	Divide to integer (long BFP)
DIEBR	Divide to integer (short BFP)
DL	Divide logical
DLR	Divide logical register
DXBR	Divide (extended BFP)
EFPC	Extract FPC
EPSW	Extract PSW
EXRL	Execute relative long
FIDBR	Load FP integer (long BFP)
FIDR	Load FP integer floating point long register
FIEBR	Load FP integer (short BFP)
FIER	Load FP integer floating point short register
FIXBR	Load FP integer (extended BFP)
FIXR	Load FP integer float extended register
IILF	Insert Immediate
IILH	Insert Immediate
IILL	Insert Immediate
IPM	Insert Program Mask
KDB	Compare and signal (long BFP)
KDBR	Compare and signal (long BFP)
KEB	Compare and signal (short BFP)
KEBR	Compare and signal (short BFP)
KIMD	compute intermediate message digest
KLMD	compute last message digest
KM	cipher message
KMAC	compute message authentication code
KMC	cipher message with chaining
KMCTR	cipher message with counter
KMF	cipher message with cipher feedback
KMO	cipher message with output feedback
KXBR	Compare and signal (extended BFP)
LAA	load and add
LAAL	load and add logical
LAN	load and and
LAO	load and or
LARL	Load Address Relative Long
LAX	load and exclusive or
LB	Load Byte
LBR	Load Byte
LCDBR	Load complement (long BFP)
LCDFR	load complement fpr long reg
LCEBR	Load complement (short BFP)
LCXBR	Load complement (extended BFP)
LCXR	Load complement float extended register

LDE	Load lengthened floating point short to long
LDEB	Load lengthened (short to long BFP)
LDEBR	Load lengthened (short to long BFP)
LDER	Load length float short to long register
LDXBR	Load rounded (extended to long BFP)
LEDBR	Load rounded (long to short BFP)
LEXBR	Load rounded (extended to short BFP)
LEXR	Load rounded float extended to short register
LFAS	load fpc and signal
LFPC	Load FPC
LHI	Load Halfword Immediate
LHR	Load halfword register
LHRL	Load halfword relative long
LHY	load halfword y
LLC	Load Logical Character
LLCR	Load Logical Character
LLH	Load Logical Halfword
LLHR	Load Logical Halfword
LLHRL	Load logical halfword relative long
LLILF	Load Logical Immediate
LLILH	Load Logical Immediate
LLILL	Load Logical Immediate
LNDBR	Load negative (long BFP)
LNDFR	load negative fpr long reg
LNEBR	Load negative (short BFP)
LNXBR	Load negative (extended BFP)
LNXR	Load negative floating point extended register
LOC	load on condition
LOCR	load on condition register
LPD	load pair disjoint
LPDBR	Load positive (long BFP)
LPDFR	load positive fpr long reg
LPEBR	Load positive (short BFP)
LPXBR	Load positive (extended BFP)
LPXR	Load positive floating point extended register
LRL	Load relative long
LRV	Load Reversed
LRVH	Load Reversed Halfword
LRVR	Load reversed register
LT	Load and Test
LTDBR	Load and test (long BFP)
LTEBR	Load and test (short BFP)
LTXBR	Load and test (extended BFP)
LTXR	Load and test floating point extended register
LXD	Load lengthened floating point long to extended
LXDB	Load lengthened (long to extended BFP)
LXDBR	Load lengthened (long to extended BFP)
LXDR	Load length float long to extended register
LXE	Load lengthened float short to extended
LXEB	Load lengthened (short to extended BFP)
LXEBR	Load lengthened (short to extended BFP)
LXER	Load length float short to extended register
LXR	Load (Extended)
LY	load y
LZDR	Load Zero (long)
LZER	Load Zero (short)
LZXR	Load Zero (extended)
MAD	Multiply and add floating point long
MADB	Multiply and add (long BFP)
MADBR	Multiply and add (long BFP)
MADR	Multiply and add floating point long register
MAE	Multiply and add floating point short
MAEB	Multiply and add (short BFP)
MAEBR	Multiply and add (short BFP)
MAER	Multiply and add floating point short register
MAY	Multiply and add unnorm long to extended FP
MAYH	Multiply and add unnorm long to extended high
MAYHR	Multiply and add unnorm long to ext high reg
MAYL	Multiply and add unnorm long to ext low FP
MAYLR	Multiply and add unnorm long to ext low register
MAYR	Multiply and add unnorm long to ext register
MDB	Multiply (long BFP)
MDBR	Multiply (long BFP)
MDEB	Multiply (short to long BFP)
MDEBR	Multiply (short to long BFP)
MEE	Multiply floating point short
MEEB	Multiply (short BFP)
MEEBR	Multiply (short BFP)
MEER	Multiply floating point short register

MFY	Multiply (Long Displacement)
MHI	Multiply halfword immediate
MHY	Multiply halfword (Long Displacement)
ML	Multiply logical
MLR	Multiply logical register
MS	Multiply single register
MSD	Multiply and subtract floating point long
MSDB	Multiply and subtract (long BFP)
MSDBR	Multiply and subtract (long BFP)
MSDR	Multiply and subtract floating point long reg
MSE	Multiply and subtract floating point short
MSEB	Multiply and subtract (short BFP)
MSEBR	Multiply and subtract (short BFP)
MSER	Multiply and subtract floating point short register
MSFI	Multiply single immediate fullword
MSR	Multiply single register
MSY	multiply single y
MVCLE	Move long extended
MVCLU	Move long unicode
MVGHI	Move long from halfword immediate
MVHHI	Move halfword from halfword immediate
MVHI	Move fullword from halfword immediate
MVST	Move string
MXBR	Multiply (extended BFP)
MXDB	Multiply (long to extended BFP)
MXDBR	Multiply (long to extended BFP)
MY	Multiply unnormalized long to extended FP
MYH	Multiply unnormalized long to extended high FP
MYHR	Multiply unnormalized long to ext high FP reg
MYL	Multiply unnormalized long to extended low FP
MYLR	Multiply unnormalized long to ext low FP reg
MYR	Multiply unnormalized long to extended register
NILF	And Immediate
NILH	And immediate
NILL	And Immediate
NRK	and distinct register
NY	and y
OILF	Or Immediate
OILH	Or Immediate
OILL	Or Immediate
ORK	or distinct register
OY	or y
PCC	perform cryptographic computation
PCKMO	perform cryptographic key management operation
PFD	Prefetch data
PFDRL	Prefetch data relative long
PKA	Pack ASCII
PKU	Pack unicode
RLL	Rotate left single logical
SAM24	Set Addressing Mode - 24 bit addressing
SAM31	Set Addressing Mode - 31 bit addressing
SDB	Subtract (long BFP)
SDBR	Subtract (long BFP)
SEB	Subtract (short BFP)
SEBR	Subtract (short BFP)
SFASR	set fpc and signal
SFPC	Set FPC
SHY	subtract halfword y
SLAK	shift left single distinct
SLB	Subtract logical with borrow
SLBR	Subtract logical with borrow register
SLFI	Subtract logical immediate
SLLK	shift left single logical distinct
SLRK	subtract logical distinct register
SLY	subtract logical y
SQD	Square root floating point long
SQDB	Square root (long BFP)
SQDBR	Square root (long BFP)
SQDR	Square root floating point long register
SQE	Square root floating point short
SQEB	Square root (short BFP)
SQEBR	Square root (short BFP)
SQER	Square root floating point short register
SQXBR	Square root (extended BFP)
SQXR	Square root floating point extended register
SRAK	shift right single distinct
SRK	subtract distinct register
SRLK	shift right single logical distinct
SRNM	Set BFP rounding mode (2 bit)

SRST	Search string
SRSTU	Search string unicode
STFPC	Store FPC
STHRL	Store halfword relative long
STOC	store on condition
STRL	Store relative long
STRV	Store Reversed
STRVH	Store Reversed Halfword
STY	store y
SXBR	Subtract (extended BFP)
SY	subtract y
TAM	Test Addressing Mode
TBDR	convert float long to bfp long reg
TBEDR	convert float long to bfp short reg
TCDB	Test data class (long BFP)
TCEB	Test data class (short BFP)
TCXB	Test data (extended BFP)
THDER	convert bfp short to float long reg
THDR	convert bfp long to float long reg
TMLH	Test Under Mask High
TMLL	Test Under Mask Low
TP	Test Decimal
TRE	Translate Extended
TROO	Translate One to One (ETF2 installed)
TROT	Translate One to Two (ETF2 installed)
TRTE	Translate and test extended
TRTO	Translate Two to One (ETF2 installed)
TRTR	Translate and Test Reverse
TRTRE	Translate and test reverse extended
TRTT	Translate Two to Two (ETF2 installed)
UNPKA	Unpack ASCII
UNPKU	Unpack unicode
UPT	Update tree
XILF	Exclusive Or Immediate
XRK	exclusive or distinct register
XY	exclusive or y