TEMPO Program Update Release Notes

RELEASE - July 2007

Contents

1. Installing this Program Update

Please follow the procedure specified in the Program Update email that you received. You will find this in the UPDATE.PDF file.

2. Incompatibilities with the Previous Release

This release is upwardly compatible from the previous release except where noted.

If you use TEMPOW's DDE server, the TPB database item has been renamed to SKIP. You will need to make this syntatic change in your TEMPO protocols. See below for more information.

3. Program Update Organization

The TEMPO/Win and TEMPONET Program Updates organization and installation procedure has changed.

The new installation procedure will be emailed to you along with the relevant FZ files. FZ is a new file format for Program Update files (see below, New Format for Program Update Files - FLIP.EXE).

A Program Update distribution for a single TEMPO system consists of the following FZ files:

TEMPONET

tn.fz	TEMPONET common files (for all clients)
kernel.fz	KERNEL specific files (e.g.: KPED.FZ, KBTD.FZ, etc.)
tnvs.fz	VideoSYNC for TEMPONET (optional)
rdx.fz	RDX Library (optional)

TEMPO/Win

tw.fz	TEMPO/Win common files (for all clients & servers)
kernel.fz	KERNEL specific files (e.g.: KPED.FZ, KBTD.FZ, etc.)
twvs.fz	VideoSYNC for TEMPO/Win (optional)
rdx.fz	RDX Library (optional)

You will receive only one copy of each file, even if the file is used on multiple TEMPO systems (duplicate copies are not sent). If a particular file is to be used on several TEMPO systems, simply copy it to each TEMPO system and use the FLIP program to unpack them.

If you have more than one TEMPO system, you should receive the appropriate files for all TEMPO systems. For example, suppose you have the following TEMPO systems:

TEMPO 1: TEMPONET with KPED, VideoSYNC and the RDX library
TEMPO 2: TEMPO/Win with KBTD
TEMPO 3: TEMPO/Win with KPED and VideoSYNC

You will receive one copy of each of the following files:

File	System	Description
tn.fz	1	TEMPONET common files

kped.fz	1,3	KPED KERNEL specific files
tnvs.fz	1	VideoSYNC for TEMPONET
rdx.fz	1	RDX Library
tw.fz	2,3	TEMPO/Win common files
kbtd.fz	2	KBTD KERNEL specific files
twvs.fz	3	VideoSYNC for TEMPO/Win

Copy these file to the appropriate system:

Your TEMPONET System #1 Files

tn.fz	TEMPONET common files
kped.fz	KPED KERNEL specific files (Same as System 3)
tnvs.fz	VideoSYNC for TEMPONET
rdx.fz	RDX Library

Your TEMPO/Win System #2 Files

tw.fz	TEMPO/Win common files
kbtd.fz	KBTD KERNEL specific files

Your TEMPO/Win System #3 Files

tw.fz	TEMPO/Win common files
kped.fz	KPED KERNEL specific files (same as System 1)
twvs.fz	VideoSYNC for TEMPO/Win

4. Format for Program Update Files - FLIP.EXE

Some customers have had difficulty receiving ZIP files due to their institutions email policy. This has necessitated the creation of a file format for distributing TEMPO Program Updates. This format, called the FLIP format, is quite simple: every bit in a file is "flipped" (0 changed to 1 and 1 changed to 0) from its original value. To restore the original contents of a flipped file, simply flip it a second time. Thus, flipping a file twice restores its original contents.

Files with the .FZ extension are ZIP files that have been flipped once.

The FLIP.EXE program included in this program update (distributed in a ZIP file) is used to flip a TEMPO Program Update files .FZ file. For .FZ files, FLIP will name the flipped file with the .FZ extension and will automatically invoke PKUNZIP.EXE to unzip it.

The .FZ file format is used for all of the TEMPO Program Update files. For example, to flip and unzip the TN.FZ (TEMPONET Program Files) from a Command Prompt, type:

```
flip tn.fz
```

You should see output similar to the following:

```
FLIP0 - 3.0 Copyright (C) 2005 Reflective Computing.  All Rights Reserved.
FLIP0 - Flipping 'C:\usr\email\tn.fz' to 'tn.zip' ..
FLIP0 - 340536 bytes flipped from 'C:\usr\email\tn.fz' to 'tn.zip'.
FLIP0 - Unzip output file 'tn.zip' [YES/NO/QUIT]?y
FLIP0 - Running 'pkunzip -d -o tn.zip >>FLIP0.log'..
```

```
FLIP0 - Delete output file 'tn.zip' [YES/NO/QUIT]?y
FLIP0 - Deleting 'tn.zip'
```

FLIP flips the TN.FZ file and creates the TN.ZIP file. It then asks if you want to unzip the TN.ZIP file. If you answer *yes*, it invokes PKUNZIP to unzip TN.ZIP. FLIP then asks if you want to delete the TN.ZIP file. Answering *yes* will delete TN.ZIP.

The FLIP program includes a small help page which you can access by typing FLIP at the Command Prompt without any arguments.

Before flipping FZ files, you must install FLIP.EXE in your TEMPO directory. For users who don't have FLIP.EXE, you should receive FLIP.ZIP, which contains FLIP.EXE. After copying FLIP.ZIP to your TEMPO directory, use PKUNZIP to unzip it by typing at a Command Prompt:

```
cd \tempo
pkunzip -d -o flip.zip
```

If you have difficulty receiving the FLIP.ZIP file (i.e., because your email software has filtered it out), please contact Reflective Computing. We recommend that you always use the latest anti-virus software to scan files you receive from the internet. FLIP should only be used with .FZ files that you receive from Reflective Computing.

5. New DDE Sample Program - TEMPOEXP.C

The TEMPOEXP.C program provides an example (in C) for retrieving the value of a TEMPO expression using TEMPOW's DDE (Dynamic Data Exchange) interface.

The TEMPEXP.* files are found in the TEMPO\DDE subdirectory. TEMPOEXP.C contains detailed documentation and a brief help page on its use. The TEMPOEXP.EXE utility program provides a convenient way to retrieve expression values from a command line.

6. New VideoSYNC Command: IS - Save Screen to PCX File

The new IS VideoSYNC command is used to save a rectangular region of the current RW page (see the RW command) to a PCX file. The IS command has the following syntax:

```
IS ulx, uly, lrx, lry, pcxFilename
```

The ulx, uly are the upper left corner and lrx, lry are the lower right corner of the rectangular region of the current RW page that is saved to the PCX file. The pcxFilename is the name of the PCX file and can contain a complete path prefix.

If a path is not specified in the pcxFilename, the PCX file is stored on the RAM drive created by the VideoSYNC boot diskette (for TEMPO/Win users, the default location is the current hard drive directory at the time you ran the VideoSYNC program).

For an example on how to use the IS command, see the PROWIN\acquire.pro protocol's saveVSscreen() process.

7. New PCL function: writefx()

The writefx() PCL function build a single line of text to be written to a file in a 255 byte buffer in the kernel from multiple calls to writefx().

The syntax for the writefx() function is:

```
int writefx(formatString, args, ...)
```

The format string and args use the same conventions as the printf() function.

The first word (the characters up to the first space) are treated as the name of the file. The file name may include the full path to the file. If no path is specified, the current TEMPOW directory is used (see the CD TEMPO command).

The writefx() function returns the current number of commands in the L3COMMAND queue. Note that this is different from the return value for writef(), which returns the number of characters written.

One or more calls to writefx() cause characters to be stored in a 255 byte buffer, which is private to the writefx() function. When a newline character ('\n') is stored, the writefx() function generates one or more L3COMMANDS (using the WRITE TEMPO command) to write the text in the buffer to the specified file.

For example, the following code writes to FILE.TXT a single line that looks like: "x=14, y=16, z=2\n".

```
writefx("file.txt x=%d, ", x);
writefx("y=%d, ", y);
writefx("z=%d\n", z);            // The newline causes writef to output
                                 // a WRITE command to the L3COMMAND queue
```

When using multiple calls to writefx() to build one output line of text, the filename is specified only in the first writefx() call. See below for how this differs from the writef() function.

The above three lines are equivalent to using a single call to writefx():

```
writefx("file.txt x=%d, y=%d, z=%d\n", x, y, z);
```

or a single call to the writef() function:

```
writef("file.txt x=%d, y=%d, z=%d\n", x, y, z);
```

Note that when building a line of text with multiple calls, there is a difference between the writefx() and writef() functions. When using multiple calls to writef() function, each call to writef() generates one L3COMMAND command. Each call to writef() also requires that you specify the name of the file as the first word:

```
writef("file.txt x=%d, ", x);
writef("file.txt y=%d, ", y);
writef("file.txt z=%d\n", z);
```

Compare the above three lines with the corresponding three lines using writefx() above.

Both writef() and writefx() automatically break the line into one or more L3COMMANDs, if necessary. (Each L3COMMAND command has a limit of 78 bytes). This does not affect the result that is stored in the output file but long lines may consume more then one entry in the L3COMMAND queue.

Note that the internal buffer in writefx() is 255 bytes. Writefx() generates an error if a single output line exceeds this limit; the output line is truncated to 255 bytes. (Because writef() function outputs one or more L3COMMANDs with each call, it does not need to maintain an internal buffer.)

The fact that writefx() maintains an internal buffer could cause undesirable results if more than one PCL process is making calls to writefx(). If multiple processes are each making multiple calls to writefx(), the text from each process could become interleaved in the file or text could be written to the wrong file. Therefore, it is recommended that if you use multiple calls to writefx() in a process, the last call should also contain the newline character at the end of the text. This will force writefx() to emit the text to the L3COMMAND queue and flush its internal buffer.

8. PLOAD Displays More PCL Compiler Error Messages

When the PCL compiler encounters an error in your protocol, it writes messages to the .ERR file (same base file name as your protocol). TEMPOW's PLOAD command now displays in the Message Window the first 8 lines of text from the .ERR file. These lines are also written to buger for diagnostic purposes.

9. TPB Item Removed from TEMPO and COBALT DDE Server

The TPB item has been removed from TEMPO and COBALT's DDE server and is no longer available. Users should switch to using the SKIP item, as documented in the TEMPO Reference Manual, which returns precisely the same value.

10. PCL WAIT Instructions Optimized

The six PCL wait instructions (WAIT, NEXTTICK, WSON, WSOF, WCON and WCOF) have been optimized. Each of these instructions now compile to 4 bytes of INSTRUCTION memory instead of 14 bytes.

The KERNEL process control block (PROCESSES) has increased in size from 24 bytes to 36 bytes.

11. PCL SPAWN and SPAWNWAIT Instruction Optimized

The PCL SPAWN and SPAWNWAIT instructions have been optimized to execute significantly faster than before.

12. TEMPO Huge Protocol Support - XMS Instruction Caching

The KERNEL executes compiled protocol code in conventional memory. This imposes a limit based on the fact that conventional memory is a limited resource.

For protocols that require more space than will fit in conventional memory, the KERNEL now supports Extended Memory (XMS) Instruction Caching. XMS memory does not have the limitations of conventional memory: The KERNEL can access all of the memory (RAM) in the server computer as XMS memory. When XMS Instruction Caching is enabled, the compiled instructions for each process are stored in XMS memory. The KERNEL imposes a limit of about 64 Mb for compiled protocol instructions in XMS memory (most protocols are 20-50 K bytes of compiled code).

Process instructions can not be directly executed in XMS memory. Before a process can be executed, the KERNEL must copy the processes instructions to conventional memory. When your protocol attempts to

execute (i.e., with a SPAWN statement) a process that is not currently loaded in conventional memory, the KERNEL automatically loads it by copying that processes instructions from XMS memory. If the conventional memory area is full, the KERNEL first unloads one or more processes from the conventional memory to make room for the desired process. This activity of loading and unloading process instructions, called *instruction caching*, is performed automatically by the KERNEL and is completely transparent to your protocol. XMS Instruction Caching does not require you to change your protocol in any way and your protocol will, in general, execute in exactly the same was as it does when XMS Instruction Caching is disabled.

XMS Instruction Caching is disabled by default and all protocol instructions are loaded into conventional memory.

During protocol development, you may find that you need to increase INSTRUCTIONS to accomodate the larger protocol. If you find that you can not increase INSTRUCTIONS due to the conventional memory limitation, you can enable XMS Instruction Caching.

To enable XMS Instruction Caching, use the following syntax for the SETUPTN parameter TEMPOServer/Protocol/INSTRUCTIONS:

INSTRUCTIONS=*nConventionalInstructions* , *nXMSInstructions*

where

nConventionalInstructions	is the size of of the conventional memory table.
nXMSInstructions	is the size of the XMS memory table.

Both nConventionalInstructions and nXMSInstructions are in units of 18 bytes.

If nXMSInstructions is not specified or is zero, XMS Instruction Caching is disabled.

For example, to allocate 36,000 bytes (2000 x 18) for the conventional memory table and 360,000 bytes (20,000 x 18) for the XMS table, set INSTRUCTIONS to:

INSTRUCTIONS=2000,20000

If you need more XMS space, you can increase nXMSInstructions.

The limit for nXMSInstructions is 3,500,000 (about 64 Mb).

When XMS Caching is enabled, the smallest allowed value for nConventionalInstructions is 920. If nConventionalInstructions is zero and nXMSInstructions is non-zero, the KERNEL defaults to nConventionalInstructions 1820 (about 32 K b). The conventional memory area must be at least large enough to hold the largest process.

If you don't need to use XMS Instruction Caching, we recommend that you leave it disabled. If you do need to use it, we recommend you set nXMSInstructions to 2 times the size of your largest protocol and set nConventionalInstructions to 0.

There is a slight time penalty to loading a process (times are dependent on your computer's CPU speed and the size of the process) of about 1-25 uSec (the SHOW P display shows the load time for each process). The larger the nConventionalInstructions is, the more processes it can hold and the less

frequently the KERNEL has to load/unload processes from XMS. Making nConventionalInstructions as *large* as posssible reduces the number of times the KERNEL will have to load/unload processes.

To determine how much space your protocol consumes, CSAVE your protocol's PCF file and examine the table in the .PCF file.

13. TEMPOW's PCF File Contains Process Specific Information

When you CSAVE your protocol configuration file, TEMPOW writes additional information about the resources your protocol consumes.

Additional detailed information about each process has been added to the .PCF file. You can examine your .PCF files with any text editor. This is the same information as in the SHOW P. Refer to SHOW P documentation for more details.

14. TEMPOW's SHOW P Displays XMS Instruction Cache Information

TEMPOW's SHOW P display has been enhanced to display detailed diagnostic information about XMS caching. Additional columns have been added to SHOW P to display the number of times the process has been loaded and the locations in conventional memory and in XMS Memory of the processe's instructions.

A portion of ACQUIRE.PCF is shown below for the ACQUIRE protocol when XMS Caching is enabled. In this example, seven of the ten processes have been loaded into conventional memory. Processes BOXOFF JUICE and FAILURE are not currently loaded in conventional memory (they have not yet executed).

```
; PROTOCOL PROCESS CONTROL BLOCKS (XMS Instructions Caching ENABLED)
; INSTRUCTIONS=900,10012
; Memory (bytes):  49144/16200 Conventional, 49144/180224 XMS
; Server CPU Mhz  Empirical 331.1    Theoretical 333.0
;
;    STATE PROCESS  Longest  0.365ms   PC INST  MAX LONG     MS  MaxMS LongMS Bytes  Conventional        XMS      Loaded  uSec
;   1 DONE MAIN                        50    0    7    7  0.000  0.108  0.108    50       2-59          2-51          1    25
;   2 DONE INIT                       172    0   14   14  0.000  0.068  0.068   172      60-239         52-223         1    12
;   3 RUN  TRIAL_LOOP                 206    1   10    6  0.006  0.064  0.064   276     946-1229       224-499         1    16
;   4 DONE SETUP                       54    0    5    0  0.000  0.030  0.000    54    1230-1291       500-553         1    16
;   5 RUN  TRIAL                      184    1   16    0  0.006  0.071  0.000   442    1292-1741       554-995         1    15
;   6 RUN  WATCHEYE                   694   53   53   51  0.062  0.096  0.096   698     240-945        996-1693        1    24
;   7 DONE BOXON                      184    0   14    0  0.000  0.045  0.000   184    1742-1933      1694-1877        1    14
;   8 DONE BOXOFF                      20    0    2    0  0.000  0.031  0.000    20    1934-1961      1878-1897        1    17
;   9 DONE JUICE                      158    0    4    0  0.000  0.042  0.000   158    1962-2127      1898-2055        1    17
;  10 SUSP FAILURE                      0    0    0    0  0.000  0.000  0.000   108    not loaded     2056-2163        0     0
;  11 SUSP HUGE1                        0    0    0    0  0.000  0.000  0.000 16144    not loaded     2164-18307       0     0
;  12 SUSP HUGE2                        0    0    0    0  0.000  0.000  0.000 14692    not loaded    18308-32999       0     0
;  13 SUSP HUGE3                        0    0    0    0  0.000  0.000  0.000 16144    not loaded    33000-49143       0     0
```

The new columns in SHOW P and the PCF file are:

- The Bytes column is the number of bytes of instruction space required by the process.
- The Conventional and XMS columns indicate the location of the process's instructions in these memory areas.
- The Loaded column indicates the number of times the process has been loaded from XMS memory.
- The uSec column indicates the longest amount of time, in microseconds, it took to load the process.

15. TEMPOW's SHOW S Displays Server Mhz

The SHOW S display has been enhanced to display the Empirical and Theoretical Mhz of the server computer. The Empirical Mhz is the actual observed rate of the server CPU. The Theoretical Mhz is the rate the CPU is supposed to operate at.

16. PROCESSES Control Block Size Increased

Due to the addition of XMS Instruction Caching, the size of a Process Control Block has increased from 24 bytes to 42 bytes. The number of Process Control Blocks allocated by the kernel is determine by the TEMPOServer/Protocol/Processes parameter.

17. DDESRV TPB Renamed to SKIP

TEMPOW and COBALT's DDE item TPB (under Topic DB) has been renamed to SKIP.

The TPB field has been obsoleted and removed from the documentation for several years but we left it in the software until now. As of this release, TPB has been removed from TEMPOW and COBALT's DDE server. If your protocols referenced this DDE item, you will see an error message like:

DDESRV - Topic 'DB' Item 'TPB 2' Invalid ITEM field or databse number.

when you run your protocol. Simply change all TPB references in your protocol to SKIP and the CLOAD the protocol to correct the problem.

18. SETUPTN - Network Software Component

The SETUPTN program has been modified to load the Network Software components from the file NETWORK.FZ file in the C:\TEMPO\TEMPONET\NETWORK directory.

19. KSTART - Improved Diagnosis of Problems

The KSTART.BAT file been improved to provide more precise reporting of problems when a problem occurs during the loading of TEMPO or COBALT servers. Suggestions are provided to the user when the kernel server fails to load because of network problems or when the kernel's protocol parameters result in the kernel consuming so much memory that the kernel server is unable to load.

20. SPAWN Statement Optimized for Speed

The SPAWN statement has been optimized and now executes several times faster then it did previously. This results in faster execution of the protocol.

21. Current Software Versions and Set Numbers

The TEMPO software is distributed as a matched set. When installing a Program Update, insure that the TEMPO software on all computers are updated from the same distribution.

TEMPOW	11.7 Set 40.13
COBALT	2.7 Set 40.13
SYMPHONY	4.4
SETUPTN	5.26
KSRV,KSRVB	11.3 Set 40.13
All KERNELS	11.02 Set 40
PCL	11.9 Set 40
KINFO	10.10 Set 40
HTB	10.1

VDOSYNC	11.9
BUGER	10.3
All DIAGNOSTIC	13.18 Set 40