

# Portable *smx* (*psmx*)

## PRODUCT BRIEF

*psmx* is the portable version of *smx*. It comes with a concise, well-written porting guide that makes porting *smx* to a non-supported or custom processor a simple process.

*psmx* is delivered with full *smx* and *smxBSP* source code, *smx* manuals, a porting guide, an x86 reference port, and a pre-built, PC-version of *pmEasy* for the reference port.

*psmx* complements other portable Micro Digital products such as *smxFile*, *smxNet*, and *smxPEG*, and it may be used with them.

### Target Port

*psmx* is written almost entirely in C. Hence the first step in porting *psmx* is to get a clean compile of the *psmx* library and of the *smx* Protosystem, using the cross-compiler selected for the target processor. This is normally easy to do since *psmx* has very clean C code – it compiles without errors or warnings for several commercial compilers. Note: *psmx* does require a minimal C++ compiler. This usually is not a problem. (If it is, contact us.)

The next step is to write assembly macros for time-critical and processor-dependent operations. there are 12 of these:

- DISABLE() interrupts
- ENABLE() interrupts
- ENTER\_ISR()
- EXIT\_ISR()
- BUILD\_NEW\_STACK()
- SWITCH\_TO\_NEW\_STACK()
- ADJUST\_NEW\_STACK()
- SAVE\_SP\_TO\_TCB()
- PUSH\_SSR\_REGS()
- POP\_SSR\_REGS()

- CALL\_EXIT\_ROUTINE\_CPP
- CALL\_ENTRY\_ROUTINE\_CPP

As is apparent, these are basic operations, which should pose no problem for any reasonable processor. Most can be implemented with just a few assembly instructions and should be an easy job for someone experienced with the target processor. For more information about the assembly macros see the *smx* Porting Guide on our website.

Once the assembly macros are written and debugged, *psmx* is ready to run on the target processor. The final stop is to alter the *smx* Board Support Package, *smxBSP*, for the specific board.

### *smxBSP*

Of the many *smxBSP*'s for supported boards, we will include the closest match to your board. The *smxBSP* API provides the hardware interface to *smx* and other *smx* products. Supporting new hardware usually requires only small modifications to the *smxBSP* routines. See the *smxBSP* brochure for more information.

### Reference Port

Part of the *psmx* package is an x86 reference port that can be run on any ordinary PC. This is good as a confidence test and it is good for becoming familiar with *smx*. Tools required to do this are:

- Microsoft Visual C++ 32-bit V5 or V6
- MASM V6.1x

Make files are provided to make the x86 versions of the *psmx* library and the *smx* Protosystem. The Protosystem includes startup and example code and serves as the basis for future applications. The make file links it with the *smx86* reference library and binds it with *pmEasy* into an EXE file.

Having rebooted the computer to DOS, the EXE file can be loaded and started from DOS. *pmEasy* runs first. It switches the processor into protected mode, and then loads the actual application, which in this case is the Protosystem. The demo part of the Protosystem will display run-time information on the screen. This completes the confidence test.

## Using WinBase

Another option is to purchase WinBase with *psmx*. WinBase permits building and testing *psmx* and the Protosystem using project files under Microsoft Visual C++ Developer Studio. This is useful not only for the confidence test, but also for learning *smx* and doing actual application code development while the target hardware is being developed.

WinBase also supports *smx++* and PEG, but not *smxFile* and *smxNet*. See the WinBase Product Brief for more information.