How to Use smx® L-Kits

September 3, 2014

by David Moore

Overview

The main steps are:

1. Build and run App + esmx (example code).
2. Disable esmx.
3. Do simple experimental modifications to app.c (or skip to step 4).
4. Replace app.c with your code.

See the document smx L-Kit Overview in the root of your Learning Kit for a brief overview of installation and tools.

Hardware Setup

Review the quick start guide provided with your board. Note that we put important details in the BSP notes in the DOC directory of the L-Kit.

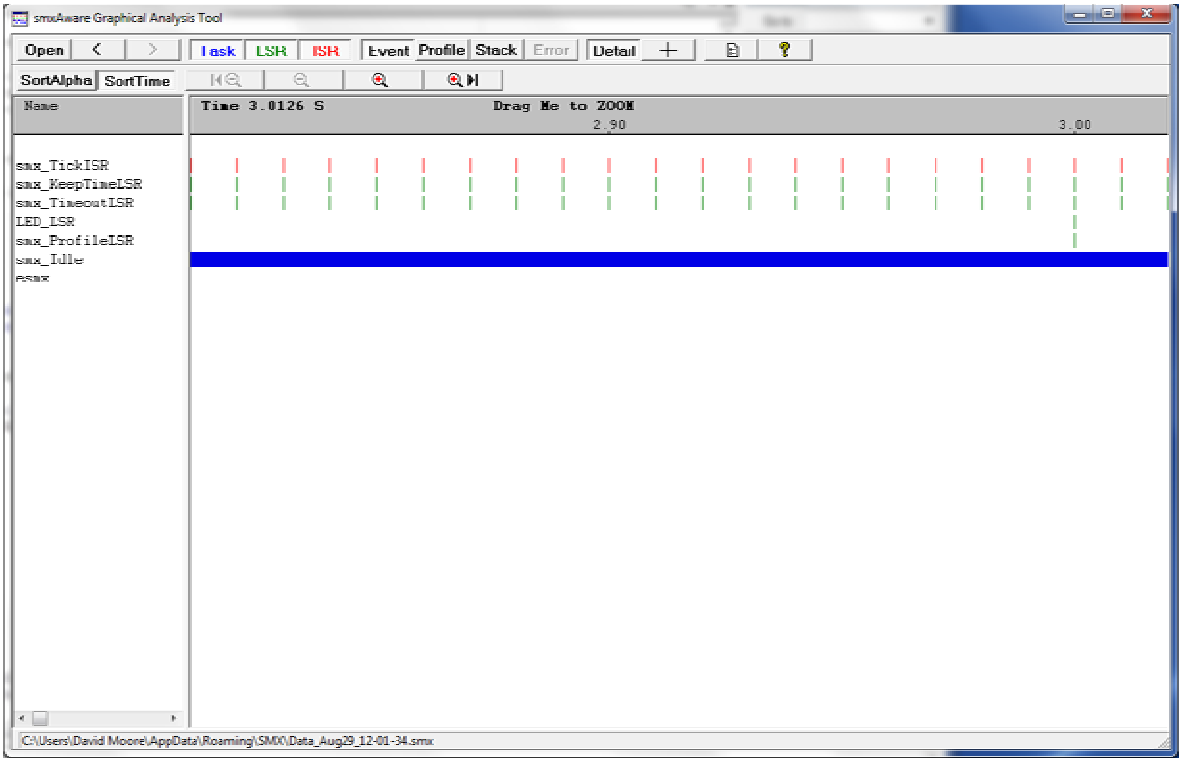
Ultra-low cost DIY boards typically minimize connectors, but some connect a UART via the USB port, which may be shared as the debug port. If your board supports this, connect TeraTerm or similar to see messages printed by our code. For targets that don't support this, the console I/O routines are omitted. We recommend setting the terminal screen to be black.

Build and Run App + esmx

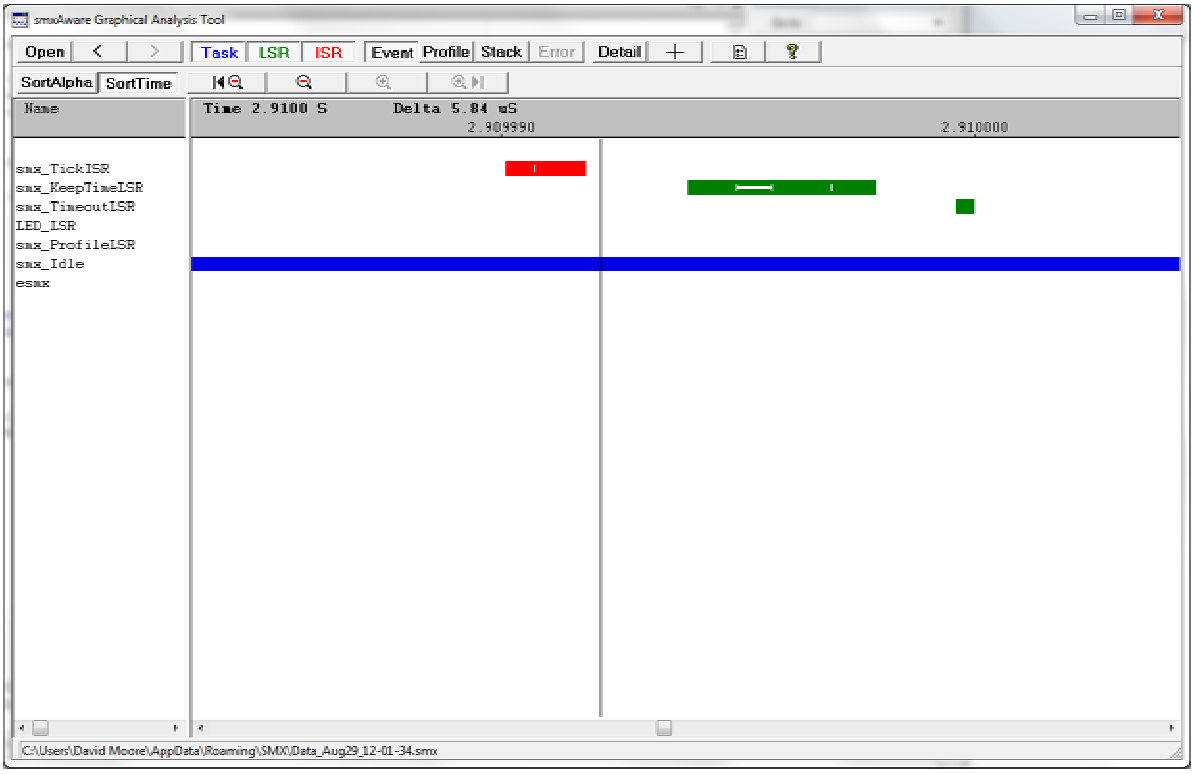
1. Start the EWARM IDE and open the workspace (project) under ESMX\IAR.AM or ESMX\IAR.ARM. Click the Make button.
2. Do the same in the APP directory.
3. Click the Download and Debug button to download the code to flash. When done, the debugger should be at main().
4. Click the Go button. You should see LED(s) blinking (if supported; see BSP notes) and messages on the LCD (if present) and on the terminal (if supported; see Hardware Setup).
5. Kill the debug session and start a new one. You can click the Debug without Downloading button since it was already programmed to flash.
6. Click the file open button and browse to ESMX\esmx.c. Scroll down to the end of the file and put a breakpoint on esmx_main(). This is the main function of the esmx task. From here you can step over and into each example. These come from the smx User's Guide and show how to do common operations. You can follow the Debug Tour document in the ESMX directory for step-by-step instructions to start using esmx.

Add smxAware

1. smxAware extends the IAR debugger to add text and graphical displays helpful for an SMX system. Exit EWARM and copy the files from SMX\SA to the EWARM directory: <ewarm>\arm\plugins\rtos\SMX (create the SMX subdirectory)
2. Restart EWARM.
3. Click the Debug button, and you will see an smxAware item added to the main menu. If not, enable it in the debugger settings: Right-click the top node, Options…, Debugger in left pane, Plugins tab in right pane, check the checkbox for smxAware. The BE choice is for big endian; the other is for little endian. Check only one, usually little endian. (The name of the .dll has “BE” if it is the big endian one.)
4. Run a couple seconds and click the stop button. Select the Graph choice to see timelines showing execution. Note that a small event buffer is configured, to minimize RAM use. You can increase EVB_SIZE in acfg.h to get a longer trace.



You can zoom easily by clicking and dragging in the timescale above the client area:



The reference line was set by right-clicking the mouse in the client area.

5. Open each of the other displays from the smxAware menu.
6. See the smxAware User's Guide for more information about using it.

Disable esmx

1. When you are done stepping through esmx, you can easily disable it by commenting out #define SMX_ESMX in CFG\iararm.h and rebuilding App. If you want to run esmx again later, you can reenable and rebuild or reinstall the L-Kit to a new directory.
2. Make, download, and run again.
3. You can experiment making simple changes to app.c, such as modifying the LED task or LSR to behave differently. You can add code to create and start a simple task.

```
TCB_PTR mytask;

void mytask_main(void)
{
  sb_DEBUGTRAP();
}
```

Add to appl_init():

```
mytask = smx_TaskCreate(mytask_main, PRI_NORM, 0, SMX_FL_NONE, "mytask");
smx_TaskStart(mytask);
```

This example doesn't do much; it just stops the debugger in mytask_main(). You would replace the debug trap with your code.

Replace app.c with Your Code

1. Replace app.c with your code. Start from this:

```
#include “smx.h”

void appl_init(void)
{
}

void appl_exit(void)
{
}
```

2. Add code to appl_init() to create a task, as in the previous section.
3. Increase settings in acfg.h as necessary to accommodate your changes. Generally it is set for a few of each object type (tasks, mutexes, etc.), but these need to be increased as you need more. smx will report an error if you don’t have enough. Put a watch on smx_errno (and sb_errno).

Next Steps

Focus primarily on the smx User’s Guide and smx Reference Manual as you develop your code.

Support

Email lk@smxrtos.com for support or to request to be put on the LK Notification List.