Introduction

µTasker is an operating system designed especially for embedded applications where a tight control over resources is desired along with a high level of user comfort to produce efficient and highly deterministic code.

The operating system is integrated with TCP/IP stack and important embedded Internet services along side device drivers and system specific project resources.

µTasker and its environment are essentially not hardware specific and can thus be moved between processor platforms with great ease and efficiency.

However the µTasker project setups are very hardware specific since they offer an optimal pre-defined (or a choice of pre-defined) configurations, taking it out of the league of “board support packages (BSP)” to a complete “project support package (PSP)”, a feature enabling projects to be greatly accelerated.

This document illustrates how to make use of the SPI EEPROM file system support. It allows saving internal FLASH space in processors with limited resources by offloading the file system to an external EEPROM connected via SPI (Serial Peripheral Interface). Very low cost components allow 16, 32 or 64k of external file system space to be connected.

The necessary circuits are shown and also a test setup for common evaluation and debug boards.

The pros and cons of SPI EEPROM

The µTasker supports a simple file system by default in internal FLASH memory. When there is space available this is the best solution since it requires no external components and access and programming is optimally fast.

However there are often circumstances when using small devices that the program size is limited by the requirement to have also space for the file system. Alternatively the space available may limit the content stored in the file system and larger space would make for improved content or larger data recordings, etc. In this case the SPI EEPROM based file system enables a simple and low cost method of ‘pepping up’ the system.

For example, the NE64 uses by default 48k linear program space and a 16k page for the file system. Should the program require the full 64k space (including paging) it is necessary either to use the 112-pin version with external paged memory to add FLASH or else the SPI EEPROM file system can be activated by using the appropriate project setting. 64k program with 64k web pages can make rather neat and powerful applications suddenly very possible even with such a small device.

I just check the present price for the devices (M95128, M95256, M95512) and they are available for between $2 and $4 for single quantities, the larger memories hardly costing more than the smaller ones….

The drawback of the SPI EEPROM solution is its speed of operation. When reading, each byte has to be collected serially from the external device and when writing the EEPROM write cycles do take somewhat longer that the internal FLASH writes. The SPI clock speed is set as high as possible to keep byte transmissions as short as possible but is limited by the speed of the device itself where between 2MHz and 10MHz (depending on voltage and exact device type) is possible, giving byte access times of around 0,8us to 4µs.

There is also a little overhead involved in addressing the chip contents, this being smaller when larger amounts of data are actually transferred in a linear fashion. Write times of up to 5ms are to be expected when actually programming data.

This is sometimes (slightly) noticeable but many applications, such as web pages from the web server, are not at all critical and so are perfect examples of how such data can be offloaded in this manor. During write cycles the application will be blocked for the write periods and so it should be considered in the design as to whether writes during critical operations could cause problems or not. Data which is set only once, or during servicing, etc. (again - web pages are good examples) will of course not cause any difficulties…

Activating the SPI EEPROM in the µTasker project

This is the simple bit, but you will need to be using the project V1.2.008 or higher for the support to be completely available.

Comment in the define `SPI_FILE_SYSTEM` and comment out the define `FLASH_FILE_SYSTEM`. The FLASH based file system will be shut out and the SPI EEPROM based version will be used instead.

At the time of writing the SPI EEPROM doesn't support sharing the SPI interface with other devices. Therefore the define `SPI_INTERFACE` should not be activated.

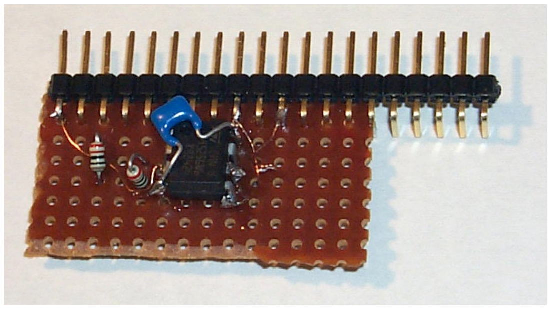
Depending on the device which is to be attached, set the define for the chip size `EEPROM_16K`, `EEPROM_32K` or `EEPROM_64K` as required.

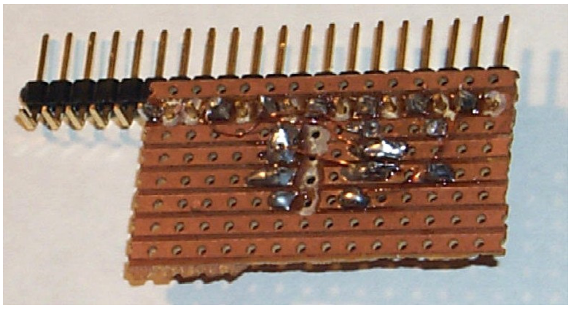
Recompile the project and you will be ready to test. The hardware must of course be available, so let's get on down to building it if it's not yet ready.

Building a test circuit for the Freescale EVAL/DEMO board

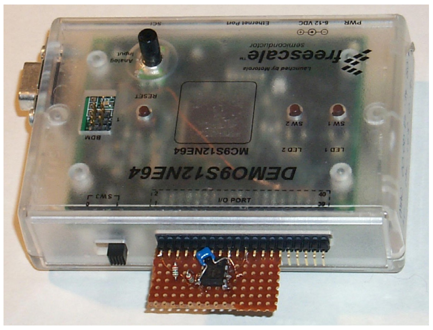
The necessary circuit for the SPI EEPROM is shown in appendix A. It is shown for connecting to a number of Freescale evaluation/demo boards.

Here are a couple of photos of a simple board which I soldered together which fits the freescale demo boards for the NE64 and ColdFire M5223X. It is no great feat of engineering as is clearly visible but it does work well and due to the few connections required, is quite easy to get right.

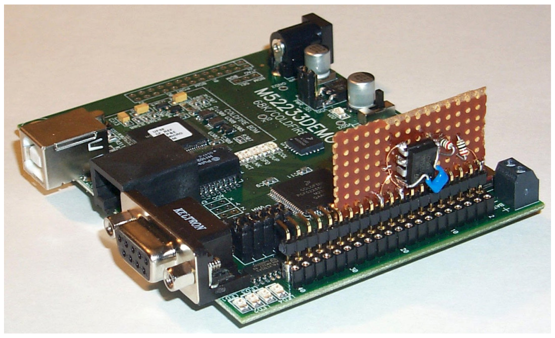

DIP version of M95256 and a single row connector. I diligently added the 100nF cap although it works without it…


The rear view is a bit of an eye-sour but, as I said, it does works fine.

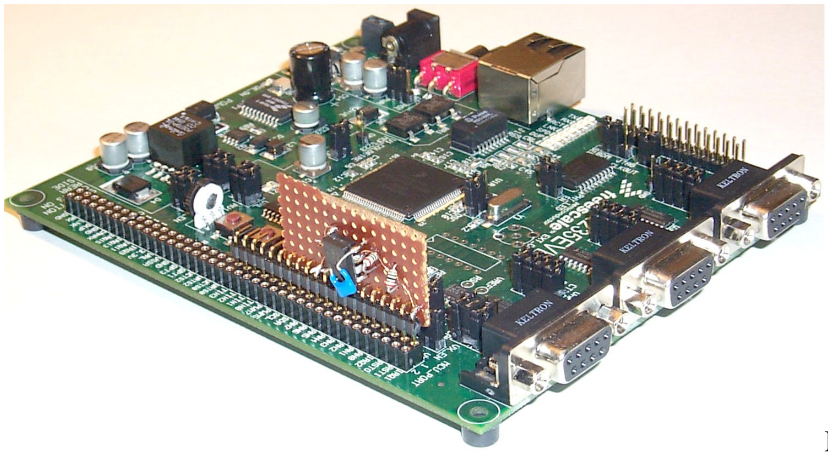
And here it is in action on the DEMO9S12NE64, the M52235EVB and the M52233DEMO. Many thanks to the designers of the freescale boards who have made the pin out compatible - so it fits all perfectly.



DEMO9S12NE64



M52233DEMO

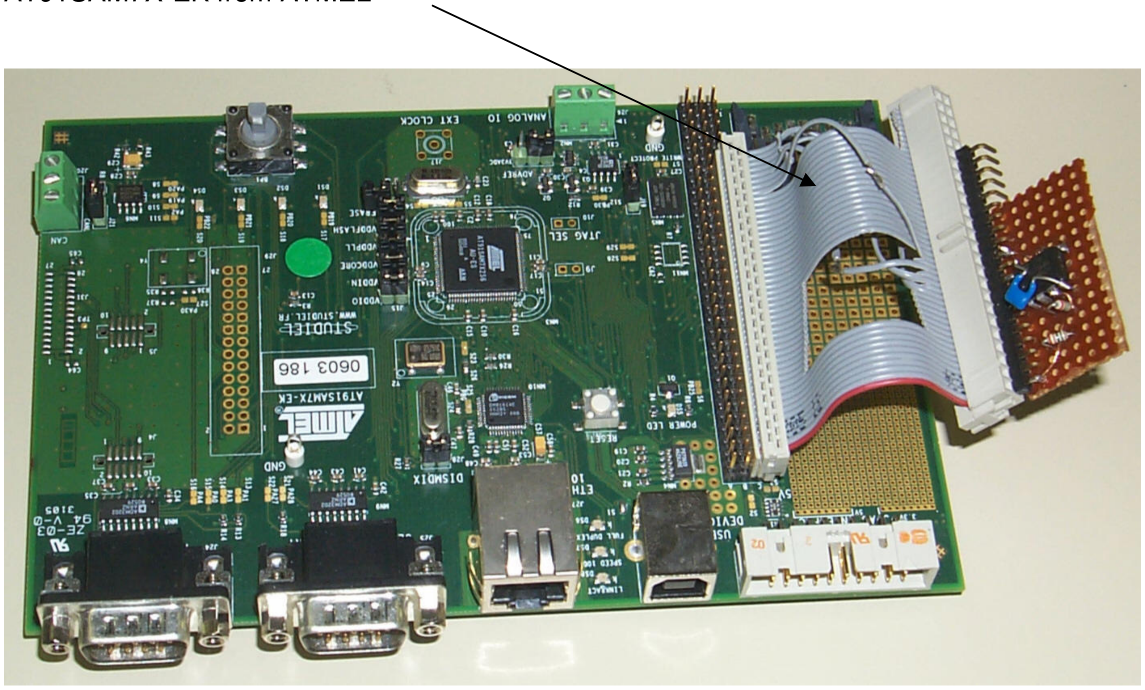


M52235EVB

Building a test circuit for the SAM7X EVAL board

The necessary circuit for the SPI EEPROM is shown in appendix B.

The same circuit board used for the Freescale boards was used for testing. As can be seen in the following photo, a cable was used to adapt the pin-out to the extension socket on the AT91SAM7X-EK from ATMEL



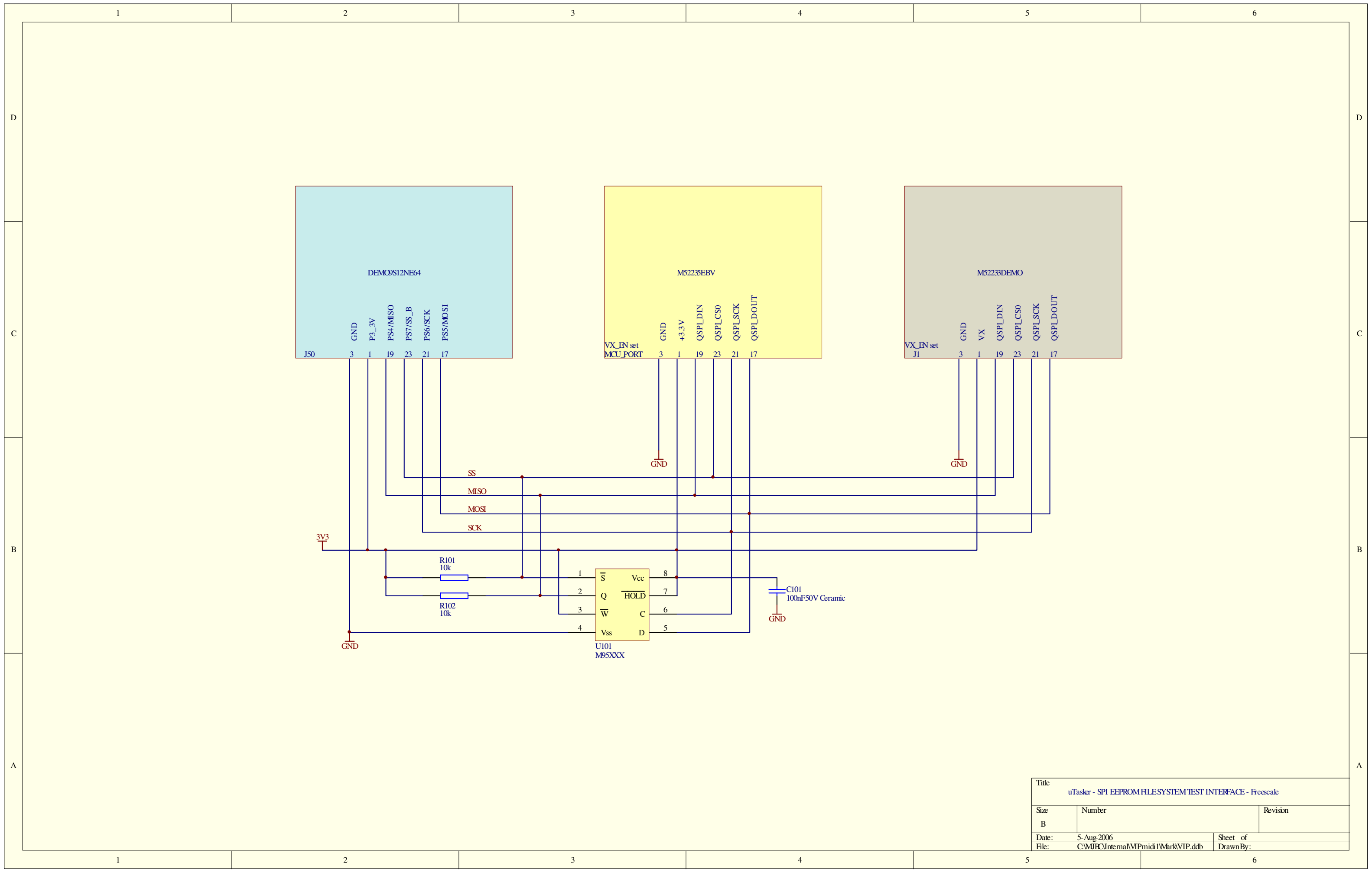
FLASH emulation

The SPI EEPROM operates in a manor which emulates the FLASH used in the FLASH file system. This means that the user interface and the file system itself remain identical. The only difference is the actual access to the memory itself where instead of reading directly from FLASH memory or writing using IAP (In-Application Programming) techniques, the reads and writes are via the SPI interface, respecting the requirements of the SPI EEPROM devices.

Modifications

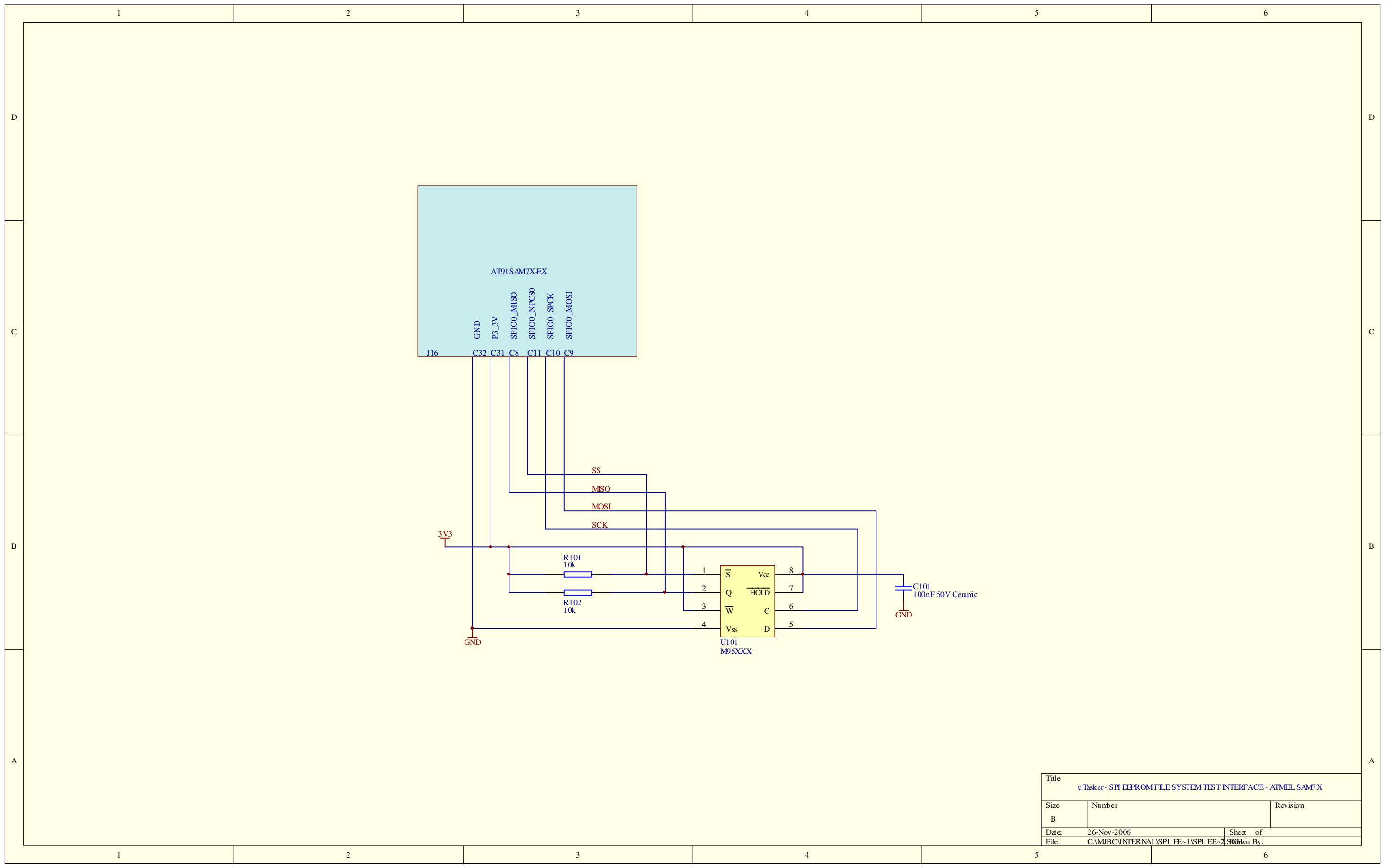
0.02 26.11.2006 SAM7X support added

Appendix A



Test adapter for NE64 and M5223X demo / evaluation boards

Appendix B



Test adapter for SAM7X evaluation board