SMX® NXP LPC55S69-EVK
Board Support Notes
IAR

October 20, 2020
Xingsheng Wan

Summary

The NXP LPC55S69-EVK board is built around a 32-bit NXP LPC55S69 ARM microcontroller. It is based on the ARM Cortex-M33 core. Cortex-M33 is an implementation of the ARM-M architecture, which is significantly different from the traditional ARM design (used by ARM7, 9, etc). **Please read the ARM-M section of the SMX Target Guide** for information about this architecture.

The BSP code consists of BSP\ARM\bspm.c and some other files in the directory BSP\ARM\LPC55Sxx. The code in directory BSP\ARMLPC55Sxx\NXP, with a few modifications, is supplied by NXP. The code clock.c is copied from NXP MCUXpresso SDK example.

Notes:

1. There is a separate prefix file (.h) in the CFG directory for each board. For IAR, the one to use is selected in **iararm.h**.

Testing was done on this board:

- BOARD: LPC55S69-EVK, Revision A2
  CPU: LPC55S69JBD100, TR7T0322.000 02, SD19271B Y

Jumpers and Switches
All jumpers are set to defaults. See *LPCXpresso55S69/55S28 Development Boards User Manual*.
Notes:
1. The jumper J10 may also be used as a convenient way to always assert ISP when the LPC55Sxx is reset or powered up.

IRQ Map

IRQ	Description	Interrupt Priority Level	Notes
Int 15	SysTick	0x20 or 0x00	smx tick; all timing based on this
28	USB0	0x80	USB0 interrupt handler
42	SDIO	0xA0	SD/MMC interrupt handler
47	USB1	0x80	USB1 interrupt handler

1. SysTick has no IRQ number because it precedes the range of IRQs. IRQ0 is Int 16. It does not appear in irq_table[] and is hooked and configured specially, not using the usual BSP functions.
2. This processor supports only 3 bits of priority levels (starting at the MSB), so there are only 8 priority levels: 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, 0xE0. Lower number is higher priority. Interrupts with the same priority level are prioritized by their IRQ number. The lower numbered IRQ has higher priority.

3. ISRs that use smx_ISR_ENTER/smx_ISR_EXIT should not be set to a lower value (higher priority) than SB_ARMM_BASEPRI_VALUE (XSMX\xarmm.h) because such level(s) are non-maskable and such ISRs must be maskable. These lower level(s) are reserved for special ISRs by using BASEPRI rather than PRIMASK to disable interrupts. SB_ARMM_DISABLE_WITH_BASEPRI in XSMX\xarmm.h selects which method is used. See section Architectural Notes/ ISR Priority Level in the SMX Target Guide.
4. Interrupt priorities can be changed as desired. However, the priority of SysTick generally should be highest maskable level since all smx timing depends on it.

Memory Map (Debug and ROM targets — Int Flash and Int SRAM)

Base Address	Size	Memory	Sections	Notes
0x00000000	640KB	Flash		Exception Vector Table, Code
0x03000000	128KB	BOOT ROM		
0x04000000	32KB	SRAM X		smxUSBD FS port
0x20010000	256KB	SRAM 0-3		Exception Vector Table, DAR,.noinit, .bss, .data
0x20040000	16KB	SRAM 4		Unused
0x40100000	16KB	Dedicated USB RAM		smxUSBD, smxUSBH HS port

Notes:
1. This memory map comes from Table 4. System memory overview in Chapter 2: LPC55S6x/LPC55S2x/LPC552x Memory Map of the *LPC55S6x/LPC55S2x/LPC552x User Manual*.
2. The Exception Vector Table is copied to SRAM in startup.c.
3. The code runs in internal Flash.
4. This is the default configuration for the Debug and ROM targets and is achieved using the linker command file **lpc55s69evk_irom_iram.icf**.

Memory Map (Debug target — Internal SRAM)

Base Address	Size	Memory	Sections	Notes
0x00000000	640KB	Flash		Unused
0x03000000	128KB	BOOT ROM		
0x04000000	32KB	SRAM X		smxUSBD FS port
0x20000000	64KB	SRAM 0		Exception Vector Table (See notes 5).
0x20010000	192KB	SRAM 1-3		Exception Vector Table, Code, DAR,.noinit, .bss, .data
0x20040000	16KB	SRAM 4		Unused
0x40100000	16KB	Dedicated USB RAM		smxUSBD, smxUSBH HS port

Notes:
1. This memory map comes from Table 4. System memory overview in Chapter 2: LPC55S6x/LPC55S2x/LPC552x Memory Map of the *LPC55S6x/LPC55S2x/LPC552x User Manual*.
2. The Exception Vector Table is copied to SRAM 0 in startup.c.
3. The code runs in SRAM.

4. This is an alternate configuration for the Debug target and is achieved using the linker command file **lpc55s69evk_iram.icf**.
5. The data in SRAM 0 are changed by code in Flash after Reset while debugging Debug target. So we do not run the code in 0x20000000 for Debug target.

Memory Configuration

The memory maps are shown above. These are achieved with the linker command files (.icf) in APP\IAR.AM\LPC55Sxx.

Peripherals Available on the NXP LPC55S69-EVK Board

1. SysTick – Used to generate the smx system tick. See bspm.c in BSP\ARM.
2. Timers (UTICK, MRT CTIMER) – Available.
3. USART #0 – Initialized to 115200Kbps / no parity / 8 bits / 1 stop bit. Used as the SMX information port. This includes Protosystem/demo output messages and SMX errors. This is the USB Debug probe connector (P6) (CMSIS-DAP / J-Link) on the board.
4. USART #1-7 – Not connected on the board.
5. RGB LED (D8) – Supported by our led.c. See LED_task or LED_LSR in app.c.
6. USB Host controller – TBD
7. USB Device controller – Supported by smxUSBD. Driver is XUSBD\DCD\udip3511.c.
8. SDMMC – Supported by smxFS. Driver is XFS\fdmsdio_lpc313x.c.
9. ADC, Analog, CASPER, CRC, DSP, I2C, INPUTMUX, Inter-CPU Mailbox, ISP & IAP, PLU, PWM, SPI, WWDT – not supported.

Clock (PLL) Configuration

The clock speed is controlled by the crystal and PLL configuration, which is set in clock.c. Also see PLL0 and PLL1 functional description section inChapter 4: LPC55S6x/LPC55S2x/LPC552x SYSCON of the *LPC55S6x/LPC55S2x/LPC552x User Manual*.

Crystal / CLKIN	Core (CPU)	AHB_CLK
16.000 MHz	150 MHz	150 MHz

Timer Configuration

SysTick is used as the smx system tick, upon which all smx timing is based. The other timers are available for your use. Note that SysTick is part of the interrupt controller (NVIC), which is part of the ARM-M core, so it is documented in the section for the processor core.

UART Configuration

UART #0 is used for console I/O, in polled mode. It is connected to the Link2 Debug probe connector (P6) (CMSIS-DAP / J-Link) on the board. UART #1-7 are not connected to a COM port. Messages written by the Protosystem and demos using sb_ConWriteString(), etc are displayed to an ANSI terminal connected to top port on the board. Initialization is done in sb_PeripheralsInit() in bspm.c. It is set to 115200-N-8-1. See the code in BSP\ARM\ LPC55Sxx\uart.c.

Note: VCOM device will be found while connecting the Link2 Debug probe connector (P6) (CMSIS-DAP / J-Link) to PC. To get the VCOM driver and others, get this and install it before plugging board: www.nxp.com/lpcscrypt (or uninstall it from Device Manager if it shows VCOM with error).

USB Device Configuration (smxUSBD)

There two USB controllers/ports user can use for device function. Basically, it is the same USB IP. One supports HS, and the other supports FS. SUD_IP3511_FS is used to select which controller smxUSBD will use. SUD_IP3511_FS is 1 will use FS port, SUD_IP3511_FS is 0 will use HS port.

For FS controller, endpoint buffer offset needs to be 64-byte aligned but can be any offset within the 4M range (64K*64). The data buffer base must be 4M aligned. The buffer start address can be any SRAM area but not across 4M boundary. The size of the RAM used depends on the configuration of your device, the endpoint's max packet size and endpoint number. We are using the SRAM X located at 0x04000000, but it can be relocated to any free SRAM space that does not cross 4M boundary.
SUD_IP3511_DATA_BUF_BASE SUD_IP3511_DATA_BUF_START and
SUD_IP3511_EP_LIST_BASE can be used to adjust these settings.

For HS controller, it needs to use dedicated USB RAM located at 0x40100000. The endpoint buffer offset needs to be 64-byte aligned. The size of the RAM used depends on the configuration of your device, the endpoint's max packet size and endpoint number. The dedicated USB RAM is 16KB and it should be enough for a normal configuration.

For both FS and HS case, we put the EP List at the beginning of the SRAM range.
SUD_IP3511_DATA_BUF_BASE SUD_IP3511_DATA_BUF_START and
SUD_IP3511_EP_LIST_BASE, in udport.h, can be used to adjust those settings.

USB Host Configuration (smxUSBH)

TBD

SDMMC Configuration (smxFS SD/SDHC)

Only one microSD card slot is provided on this board. The driver is similar to LPC313x and LPC43xx but this processor has a dedicated internal DMA engine to transfer data between FIFO and application memory. Our driver uses this internal DMA to transfer data. Highspeed mode is also supported and enabled by default. SD bus speed is set to 50MHz.

Each DMA descriptor has size limitation up to 2*4096 bytes. We allocate DMA descriptors statically. We assume max data buffer is 32KB which should be enough for default smxFS setting. Size is set in the driver:

```
#define MAX_BUF_SIZE (32*1024)    /* configurable */
```

Debugging with IAR C-SPY

It is necessary to change the project settings to specify and configure it for the JTAG unit you are using. This is documented in the IAR EWARM User's Guide in Part 6: C-SPY Hardware Debugger Systems/ Hardware Specific Debugging. Also, see the documents about this linked by the IAR release notes (readme.htm). In the table of documents at the bottom of the release notes there are links to IAR C-SPY driver notes files.

A .mac file is provided in the project directory that the debugger runs to configure the board before downloading code. The project file is already set to point to this file.

Debugging with Onboard LPC4322 Link2 Debug probe

On board LPC4322 Link2 Debug Probe (CMSIS-DAP / J-Link) is connected to micro USB port P6. Connect to PC directly with a USB A-Male to Micro B Cable.

The on board LPC4322 Link2 Debug Probe is flashed CMSIS-DAP Firmware by default. **We tried to use CMSIS-DAP to download and debug the project, but it always failed.**

We successfully programmed J-Link firmware to LPC4322 Link2 Debug Probe and downloaded/debugged with LPC4322 Link2 Debug Probe. All settings are in project files. Please refer to section **Programming J-Link Firmware to LPC4322** below for steps to do this.

Note: Sometimes IAR gives these errors on attempt to download program:

```
Download error at 0x00000000: downloading into non-writable memory.
Download error at 0x00000000: memory write failed.
```

ISP boot does not solve this problem. (To do ISP boot, hold ISP button while press/release Reset button.)

Debugging with IAR I-jet

We could not make I-jet work. It seemed to successfully download the image to the target, but when attempting to run or step it immediately gets a Hard Fault.

Debugging with IAR J-Link

We could not make J-Link work since our J-Links are too old to support ARMv8.

Programming J-Link Firmware to LPC4322

1. Download the lpcscrypt utility from www.nxp.com/lpcscrypt. We only used the Windows version.
2. Install lpscrypt. It will install in folder C:\NXP\LPCScrypt_<version>.
3. Close jumper **J4 DFU**, and connect the board to PC via port P6 (Debug Link).
4. Open a CMD prompt window and change to directory **C:\NXP\LPCScrypt_<version>\scripts**.
5. Run script **program_JLINK.cmd** and press any key to start programing J-Link firmware.
6. When programming J-Link firmware is done, press CTRL-C to quit the script.
7. Remove jumper **J4 DFU** and unplug/replug the USB cable to reboot the board.
8. Set IAR debug connection to J-Link and click the Debug button.

Boot and Startup Code

The same startup code is linked to the Protosystem for all build targets, so we don't rely on any boot code in flash to bring up the board. Summary of startup:

__iar_program_start -> __cmain -> __low_level_init -> main() -> smx_Go() -> smx_SchedRunTasks() -> ainit() -> tasks

__iar_program_start is in the IAR run time library, DLIB, in cstartup_M.c. It calls __cmain in cmain.s, which calls __low_level_init, which is our startup code in BSP\ARM\IMXRT\startup.c, and then it calls other init routines before calling main(). main() is in APP\main.c.

At startup, the main stack is used. smx_Go() changes to the dual-stack model so that tasks use the process stack pointer (PSP), and ISRs, LSRs, and the scheduler use the main stack pointer (MSP). There is a brief time during startup before the first task runs that both pointers are writing to the same area of memory, corrupting it, but that is ok because it is never used.

Further Reading

Refer to the release notes in the DOC directory and the ARM section of the SMX Target Guide.