


Iniciar sesión para crear y calificar contenido y para seguir, marcar y compartir contenido con otros miembros.



Startup code and interrupt handlers

Pregunta realizada por **LPCware Support** el 31-mar-2016

 Me gusta • 0

 Comentario • 0

Overview

When you create a project using the LPCXpresso IDE project wizard, the startup code generated (`cr_startup_*.c` or `cr_startup_*.cpp`) includes a vector table containing a set of default interrupt handlers, dependent upon the target MCU. For example, the following is taken from the vector table generated for the LPC11U6x:

```
extern void (* const g_pfnVectors[])(void);
__attribute__((section(".isr_vector")))
void (* const g_pfnVectors[])(void) = {
    &_vStackTop,                // The initial stack pointer
    ResetISR,                   // The reset handler
    NMI_Handler,                // The NMI handler
    HardFault_Handler,         // The hard fault handler
    0,                          // Reserved
    0,                          // Reserved
    0,                          // Reserved
    0,                          // Reserved
    0,                          // Reserved
    0,                          // Reserved
    0,                          // Reserved
    0,                          // Reserved
    SVC_Handler,               // SVC call handler
    0,                          // Reserved
    0,                          // Reserved
```

```
PendSV_Handler,           // The PendSV handler
SysTick_Handler,         // The SysTick handler

// LPC11U6x specific handlers
PIN_INT0_IRQHandler,     // 0 - GPIO pin interrupt 0
PIN_INT1_IRQHandler,     // 1 - GPIO pin interrupt 1
PIN_INT2_IRQHandler,     // 2 - GPIO pin interrupt 2
:
:
```

The first entry is the value to be loaded automatically into the stack pointer register (r13) at reset. (The symbol normally used for this is generated by the linker script generated by LPCXpresso.)

The second entry is the address of the reset handler, `ResetISR`, which is located elsewhere in the startup file. This address will be loaded into the program counter at reset (i.e. it will be where the MCU starts executing from).

There are then a set of handlers, starting with `NMI_Handler` and ending with `SysTick_Handler` which are the standard default handlers for Cortex-M. The exact list will depend upon which Cortex part is being used, Cortex-M3/M4 implement more than Cortex-M0/M0+. For more details, please see ARM's Cortex documentation.

These default handlers are defined higher up the startup file as weak:

```
WEAK void SVC_Handler(void) ;
```

so that if your main application code provides a (non-weak) implementation of a handler function **with exactly the same name**, this will be used instead of the default version contained in the startup code.

Finally, the vector table contains a list of MCU specific handlers. The exact number of these, and the interrupts that these implement will depend upon what MCU is being used. For details, please see the NXP User Manual for the MCU that you are using.

Forward declaration of the MCU specific interrupt handlers are again provided for these higher up the startup file, for example:

```
void PIN_INT0_IRQHandler (void) ALIAS(IntDefaultHandler) ;
```

These are weak and also aliased to the `IntDefaultHandler` lower down the startup code - which is a "forever" loop.

Again, if your main application code provides a (non-weak) implementation of a particular interrupt handler function with exactly the same name, this will be used, rather than the `IntDefaultHandler` from the startup code.

Interrupt Handlers in C++ applications

One thing to be careful of is that if your application is a C++ one, then any interrupt handlers defined in C++ files within in your main application will need to have C linkage rather than C++ linkage. To do this, make sure that you are using `extern "C" { }` around the interrupt handler within your main application code.

For more details, please see you favorite C++ text book, or an internet search will bring up lots of useful links - for example:

http://en.wikipedia.org/wiki/Compatibility_of_C_and_C%2B%2B#Linking_C_an...

Nadie más tiene esta pregunta

Visibilidad:  LPCXpresso IDE FAQs • 546 vistas

Modificado por última vez el 17-may-2016 4:57

Etiquetas: startup lpcxpresso startup_code interrupt_handlers

Categorías: Compiler / Assembler / Linker

0 Respuestas

Contenido relacionado

Example for the FlexTimer Module with interrupts on the FRDM-K20D50

how to initialize the EMC for using off-chip SRAM in startup.s file

Jumping from Bootloader to Application

how to initialize the EMC for using off-chip SRAM in startup.s file

how to initialize the EMC for using off-chip SRAM in startup.s file

Contenido recomendado

Want to interface customized camera in CSI

VSCP를 사용한 간편한 IoT 구현: 1 - 프로젝트 및 VSCP 소개

Most accurate way to get elapsed time (in microseconds) on iMX6?

Are the UBOOT sources available for SCM-IMX6DQ?

DMA on kinetis k64

ABOUT NXP

[Investors](#)

[Partners](#)

[Careers](#)

FOLLOW US



NEWS 11 Jan 2016

Photo Advisory -- US Secretary of Transportation, Anthony Foxx, Meets With NXP CEO Rick Clemmer at CES 2016

[Read More](#)

RESOURCES

[Mobile Apps](#)

[Press, News, Blogs](#)

[Contact Us](#)



[Privacy](#) | [Terms of Use](#) | [Terms of Sale](#) | [Feedback](#)

©2006-2016 NXP Semiconductors. All rights reserved.

[Inicio](#) | [Parte superior de la página](#) | [Ayuda](#)

© 2016 Jive Software | Con tecnología de **jive**