STM32 Null Pointer Detection

The NULL Pointer Problem

A common cause of software bugs is null pointers. Pointers may be NULL if they are un-initialized and un-checked. The use of NULL pointers almost always results in something bad happening. Often, NULL pointer access can cause error exceptions and or diagnostic crashes. But on MCUs that have valid address decoding at address 0x0000:0000, the use of NULL pointers may not cause a crash at all but may, instead, cause strange behaviors that can sometimes be difficult to debug.

Cortex-M Memory

The Cortex-M family (Cortex-M0, M3, and M4) are such MCUs. They have their interrupt vectors positioned at address zero. Because of this, NULL pointer accesses will not necessarily cause crashes. Instead, the NULL pointers will access memory in the vicinity of the vector table and who knows what will happen next?

STM32 Memory Aliasing

The STMicro STM32 family of Cortex-M3/4 MCUs do things a little differently. FLASH is physically addressed at address 0x0800:0000; the STM32 vector table is then physically located at 0x0800:0000 instead of 0x0000:0000. If the STM32 hardware is configured to boot from FLASH, then the the STM32 will remap the FLASH memory so that is aliased at address 0x0000:0000. In that way, the STM32 can boot from FLASH or external memory or any other memory region that it is capable of mapping.

In the NuttX linker scripts, the applications are linked to execute from the physical FLASH region at address 0x0800:0000. All valid FLASH memory access will then access memory in the 0x0800:0000 FLASH address range. But illegal NULL pointer access will access the aliased copy of FLASH beginning at 0x0000:0000. So we still have the problem.

The Cortex-M Memory Protection Unit

The Memory Protection Unit (MPU) is an optional component of a Cortex-M implementation. Most popular Cortex-M3/4 MCUs do support the MPU. The MPU can be used to protect regions of memory so that if there is any attempted, unauthorized access to certain memory regions, then a memory protection violation exception will occur and the system will detect the illegal access.

See the ARM website for more information about the Cortex-M3/4 families and the Cortex-M3/4 MPU. See, for example "2.2. Memory Protection Unit (MPU)".

Using the MPU to Detect Null Pointer Usage

So, for the STM32, one thing that we can do is to program the MPU to prohibit software access to the memory region beginning at address 0x0000:0000. Petteri Aimonen recently posted a code snippet on the NuttX Forum showing how to do this. Here is Petteri's post:

```c
/* Catch any null pointer dereferences */
int region = 0;
putreg32(region, MPU_RNR);
putreg32(0, MPU_RBAR);
putreg32(MPU_RASR_ENABLE | MPU_RASR_SIZE_LOG2(20) | (0xFF << MPU_RASR_SRD_SHIFT) | MPU_RASR_AP_NONO, MPU_RASR);
mpu_control(true, false, true);
```
From:
http://nuttx.org/ - NuttX Real-Time Operating System

Permanent link:

Last update: 2018/04/16 00:48