

## abi\_c\_cmse\_nonsecure\_call

The tracking issue for this feature is: [#81391](#)

The [TrustZone-M feature](#) is available for targets with the Armv8-M architecture profile ( `thumbv8m` in their target name). LLVM, the Rust compiler and the linker are providing [support](#) for the TrustZone-M feature.

One of the things provided, with this unstable feature, is the `C-cmse-nonsecure-call` function ABI. This ABI is used on function pointers to non-secure code to mark a non-secure function call (see [section 5.5](#) for details).

With this ABI, the compiler will do the following to perform the call:

- save registers needed after the call to Secure memory
- clear all registers that might contain confidential information
- clear the Least Significant Bit of the function address
- branches using the BLXNS instruction

To avoid using the non-secure stack, the compiler will constrain the number and type of parameters/return value.

The `extern "C-cmse-nonsecure-call"` ABI is otherwise equivalent to the `extern "C"` ABI.

```
#![no_std]
#![feature(abi_c_cmse_nonsecure_call)]

#[no_mangle]
pub fn call_nonsecure_function(addr: usize) -> u32 {
    let non_secure_function =
        unsafe { core::mem::transmute::<usize, extern "C-cmse-nonsecure-call" fn() -
> u32>(addr) };
    non_secure_function()
}
```

```
$ rustc --emit asm --crate-type lib --target thumbv8m.main-none-eabi function.rs
```

```
call_nonsecure_function:
    .fnstart
    .save    {r7, lr}
    push     {r7, lr}
    .setfp   r7, sp
    mov      r7, sp
    .pad     #16
    sub      sp, #16
    str      r0, [sp, #12]
    ldr      r0, [sp, #12]
    str      r0, [sp, #8]
    b        .LBB0_1
.LBB0_1:
    ldr      r0, [sp, #8]
    push.w   {r4, r5, r6, r7, r8, r9, r10, r11}
    bic      r0, r0, #1
    mov      r1, r0
```

```
    mov     r2, r0
    mov     r3, r0
    mov     r4, r0
    mov     r5, r0
    mov     r6, r0
    mov     r7, r0
    mov     r8, r0
    mov     r9, r0
    mov     r10, r0
    mov     r11, r0
    mov     r12, r0
    msr     apsr_nzcvq, r0
    blxns   r0
    pop.w   {r4, r5, r6, r7, r8, r9, r10, r11}
    str     r0, [sp, #4]
    b       .LBB0_2
.LBB0_2:
    ldr     r0, [sp, #4]
    add     sp, #16
    pop     {r7, pc}
```