
Accessing Class and Section Information from C166 Modules

Application Note 141

August 10, 1999, Munich, Germany

by Reinhard Keil, Keil Elektronik GmbH rk@keil.com ++49 89 456040-13

Extensions in the L166 Linker/Locater Version 4.03 or higher allow you to:

- Access to section start address and section length directly from the C language.
- Access to class start address and class length directly from the C language.

With these extensions you may access class and section information within your C files. These features might be used to for In-System Flash Programming routines as described in the Application Note 138.

Get Section Start Address and Section Length

The L166 Linker/Locater resolves external symbols with the notation ***sectionname_p*** with information from the section. To support access to generated section names within the C language, question mark (?) characters in the section name are replaced with underscore (_) characters. The postfix ***_p*** specifies the information that should be generated.

Examples:

```
_PR_ABC_l_
```

Receives the section length in bytes for the section **?PR?ABC**.

```
_PR_ABC_t_
```

Receives the start or target address for the section **?PR?ABC**.

Get Class Start Address and Class Length

The L166 Linker/Locater resolves external symbols with the notation ***_classname_p*** with information from the class. The postfix ***_p*** specifies the information that should be generated.

NOTE

The start address represents the address of the first section with the specified class name. The length is derived from the end address of the last section with the specified class name. The length includes also the memory that is used by other sections that are located into gaps of the specified memory class.

Examples:

```
_NDATA0_l_
```

Receives the class length in bytes for the class **NDATA0**.

```
_NDATA0_t_
```

Receives the start or target address for the class **NDATA0**.

The file **SR0M.H** contains macro definitions for accessing section related information. These macros are explained in the following:

SR0M_PS

The macro **SR0M_PS** defines all external symbols for a program section that are required to access the SR0M section information.

```
#define SR0M_PS(n) \
extern unsigned char huge _PR_###_s_; /* section source start */ \
extern unsigned char huge _PR_###_l_; /* section len */ \
extern unsigned char huge _PR_###_t_; /* target address */
```

You may use this macro in your programs as follows:

```
SR0M_PS (FLASH) // information definitions for section ?PR?FLASH
```

SR0M_PS_TRG

The macro **SR0M_PS_TRG** returns the storage address for a program code section.

```
#define SR0M_PS_TRG(n) ((void *) & _PR_###_t_)
```

You may use this macro in your programs as follows:

```
void *sec_start;
sec_start = SR0M_PS_TRG (FLASH) // get the start address for section ?PR?FLASH
```

SR0M_PS_LEN

The macro **SR0M_PS_LEN** returns the length for a program code section.

```
#define SR0M_PS_LEN(n) ((unsigned int) & _PR_###_l_)
```

You may use this macro in your programs as follows:

```
int length;
length = SR0M_PS_LEN (FLASH) // get the length for section ?PR?FLASH
```

CLASS_INFO

The macro **CLASS_INFO** defines all external symbols for a memory class that are required to access the class information.

```
#define CLASS_INFO(n) \
extern unsigned char huge _###_l_; /* class len */ \
extern unsigned char huge _###_t_; /* class start address */
```

CLASS_START

The macro **CLASS_START** returns the start address for a memory class.

```
#define CLASS_START(n) ((void *) & _###_t_)
```

You may use this macro in your programs as follows:

```
void *class_start;
class_start = CLASS_START (NDATA0)    // get the start address for NDATA0 memory class
```

CLASS_LEN

The macro **CLASS_LEN** returns the length for a memory class.

```
#define CLASS_LEN(n)    ((unsigned long) &_##n##_l_)
```

You may use this macro in your programs as follows:

```
int length;

length = CLASS_LEN (NDATA0)          // get the length for NDATA0 memory class
```

Example Program

The following example program calculates the checksum of the NCODE memory class.

```
#include <stdio.h>           /* standard I/O .h-file          */
#include <reg167.h>          /* special function register C167    */
#include <srom.h>            /* handling for SECTIONS and CLASSES */

CLASS_INFO (NCODE)          // get class info for NCODE class

/*
 * Calculate Check Sum of NCODE memory class
 */
static unsigned char chksum_ncode (void) {
    unsigned char huge *s;
    unsigned long l;
    unsigned char chksum;

    chksum = 0;
    s = CLASS_START (NCODE);    // get start address of NCODE class
    l = CLASS_LEN (NCODE);      // get length      of NCODE class
    while (l) {
        chksum += *s++;
        l--;
    }
    return (chksum);
}

void main (void) {           /* execution starts here          */
    unsigned char chksum;
    void huge *class_start;

    /* initialize the serial interface */
#ifndef MCB167               /* do not initialize if you use Monitor-166 */
    P3 |= 0x0400;            /* SET PORT 3.10 OUTPUT LATCH (TXD) */
    DP3 |= 0x0400;           /* SET PORT 3.10 DIRECTION CONTROL (TXD OUTPUT) */
    DP3 &= 0xF7FF;           /* RESET PORT 3.11 DIRECTION CONTROL (RXD INPUT) */
    S0TIC = 0x80;            /* SET TRANSMIT INTERRUPT FLAG */
    S0RIC = 0x00;            /* DELETE RECEIVE INTERRUPT FLAG */
    S0BG = 0x40;             /* SET BAUDRATE TO 9600 BAUD */
    S0CON = 0x8011;          /* SET SERIAL MODE */
#endif

    class_start = CLASS_START (NCODE);
    printf ("NCODE Start Addr: %06lX\n", ((unsigned long) class_start));
    printf ("NCODE Length:      %06lX\n", ((unsigned long) CLASS_LEN (NCODE)));

    chksum = chksum_ncode ();
    printf ("NCODE Check Sum:  %02X\n", chksum);
    while (1);               // end of main
}
```