KEIL SOFTWARE

C Compilers • Real-Time OS • Simulators • Education • Evaluation Boards

Data Overlaying and Code Banking with A51 Assembler Modules

Application Note 149

Oct 29, 1999, Munich, Germany

by Keil Support, Keil Elektronik GmbH support.intl@keil.com ++49 89 456040-0

This Application Note describes the steps that are required to perform data overlaying and code banking with assembler modules.

Data Overlaying

The BL51 Linker/Locater analyses the program structure of your application. Data segments that are assigned to a function can be overlaid if the functions do not call each other. This data overlaying technique is also known as *compiled-time stack* for variables and parameters.

Code Banking

For function calls into a different code bank, the BL51 Linker/Locater changes the target address of CALL instructions and generates a call to a bank switch table. Therefore the linker needs to distinguish between program code and constant segments.

NOTES
The BL51 linker/locater enables data overlaying and code banking ***only*** *if at least one object module is generated by the C51 Compiler. Therefore your project should contain on C51 source file that is directly translated to an object file with the C51 compiler. In case that you do not have any C sources, you may translate an empty file with C51.*

If you are using code banking in your assembler program you must be aware that the bank switching code might change some or the CPU registers. Please check carefully your bank switching routines to determine the registers that are affected when a bank switch occurs. These registers depend on the configuration of the L51_BANK.A51 module.

Segment Naming Conventions

For correct operation of above features the linker needs to know which parts of your program are program code and which part belongs to constants. Also the function code must be connected to local data segments that can be overlaid. This is done via the segment naming conventions known from the C51 compiler.

Each segment name has a prefix that corresponds to the memory type used for the segment. The prefix is enclosed in question marks (?). The following is a list of the standard segment name prefixes:

Segment Prefix	Data Type	Description
?PR?	code	Executable program code
?CO?	code	Constant data in program memory
?XD?	xdata	External data memory
?DT?	data	Internal data memory
?ID?	idata	Indirectly-addressable internal data memory
?BI?	bit	Bit data in internal data memory
?BA?	bdata	Bit-addressable data in internal data memory
?PD?	pdata	Paged data in external data memory

Each function in a source module is assigned a separate code segment using the ?PR?*function_name*?*module_name* naming convention. For example, the function **error_check** in the file **SAMPLE.C** would result in a segment name of ?PR?ERROR_CHECK?SAMPLE.

Segments for local variables and function parameters that should be overlaid follow the above conventions and have a different prefix depending upon the memory area in which the local variables are stored. Enclosed are the conventions that should be used for program code and local overlay able data segments. Data segments must be defined in A51 with the attribute OVERLAYABLE to enable data overlaying.

Information	Segment Type	Segment Name
Program code	code	?PR?*function_name*?*module_name*
Local DATA variables	data	?DT?*function_name*?*module_name*
Local IDATA variables	data	?ID?*function_name*?*module_name*
Local XDATA variables	data	?XD?*function_name*?*module_name*
Local PDATA variables	data	?PD?*function_name*?*module_name*
Local BIT variables	bit	?BI?*function_name*?*module_name*

Reset and Startup Code

The reset and startup code or your application must be structured the same way as the startup code of the C51 compiler. The following shows the structure of this startup code:

```
?C_C51STARTUP SEGMENT CODE     ; the code segment for the startup code
?STACK        SEGMENT IDATA    ; the segment that reserves stack space

EXTRN CODE (MAIN)              ; the main (start) entry of your application

RSEG ?STACK                    ; ?STACK segment will be place at highest possible address
                               ; to get maximum available IDATA space.  Therefore the DS 1
              DS    1          ; is typically a good choice.  If you want to ensure that
                               ; you have at least 20H bytes free space, you may enter DS 20H.

CSEG  AT  0                    ; absolute segment for reset vector
              LJMP  STARTUP1   ; jump to your startup code

RSEG ?C_C51STARTUP             ; relocateable segment for startup code
STARTUP1:     MOV   SP,#?STACK-1
               :
               :
               :
              LJMP  MAIN       ; jump to start of your application

              END
```

Interrupt Vectors

Each interrupt service routine has its own interrupt vector. For the linker it is important that you define for each interrupt vector an own absolute segment using a CSEG statement. Relocatable sections of your interrupt service routine should go into segment names using the ?PR? naming conventions.

Example:

```
CSEG AT 03H                       ; EXT0 interrupt vector
             SETB   mybit         ; interrupt function code
             RETI

CSEG AT 0BH                       ; Timer 0 interrupt vector
             LJMP   timer0isr

?PR?timer0?isr_module  SEGMENT CODE  ; program code
RSEG ?PR?timer0?isr_module
timer0isr:   :                    ; put your program code here
             :
             :
             RETI

             END
```

Program Example

The following program example shows you the structure for a simple assembler program.

Startup Code

```
MACRO ASSEMBLER A51 V6.00
OBJECT MODULE PLACED IN .\start.OBJ
ASSEMBLER INVOKED BY: C:\Keil\C51\BIN\A51.EXE .\start.a51 SET(SMALL) DEBUG EP

LOC  OBJ         LINE     SOURCE

                    1      ?C_C51STARTUP SEGMENT CODE     ; the code segment for the startup code
                    2      ?STACK        SEGMENT IDATA    ; the segment that reserves stack space
                    3      EXTRN CODE (MAIN)              ; entry of your application
                    4
----                5      RSEG ?STACK                    ; ?STACK at highest possible address
                    6      ; to get maximum available IDATA space.  Therefore the DS 1
0000                7      DS    1
                    8
----                9      CSEG  AT  0                    ; absolute segment for reset vector
0000 020000   F    10                    LJMP  STARTUP1   ; jump to your startup code
----               11      RSEG ?C_C51STARTUP             ; relocateable segment for startup code
0000 758100   F    12      STARTUP1:     MOV   SP,#?STACK-1
0003 020000   F    13                    LJMP  MAIN       ; jump to start of your application
                   14
                   15      END
```

Module 1

```
MACRO ASSEMBLER A51 V6.00
OBJECT MODULE PLACED IN .\module1.OBJ
ASSEMBLER INVOKED BY: C:\Keil\C51\BIN\A51.EXE .\module1.a51 SET(SMALL) DEBUG EP

LOC  OBJ         LINE     SOURCE

                    1      ; Module 1 of your application
                    2
                    3      PUBLIC main
                    4      EXTRN CODE (func2)
                    5
                    6      ?PR?main?module1  SEGMENT CODE
----                7      RSEG ?PR?main?module1
0000 120000   F     8      main:             CALL func1
0003 120000   F     9      CALL func2
```

```
0006 80F8         10      SJMP main
                  11
                  12      ?PR?func1?module1 SEGMENT CODE
                  13      ?DT?func1?module1 SEGMENT DATA OVERLAYABLE ; belongs to func1
                  14
----              15      RSEG ?DT?func1?module1
0000              16      func1_var:        DS   10    ; space for local variables in func1
                  17
----              18      RSEG ?PR?func1?module1
0000 F500     F   19      func1:            MOV func1_var,A
0002 22           20      RET
                  21
                  22      END
```

Module 2

```
MACRO ASSEMBLER A51 V6.00
OBJECT MODULE PLACED IN .\module2.OBJ
ASSEMBLER INVOKED BY: C:\Keil\C51\BIN\A51.EXE .\module2.a51 SET(SMALL) DEBUG EP

LOC  OBJ         LINE     SOURCE

                   1      ; Module 2 of your application
                   2
                   3      PUBLIC func2
                   4
                   5      ?PR?func2?module2 SEGMENT CODE
                   6      ?DT?func2?module2 SEGMENT DATA OVERLAYABLE ; belongs to func2
                   7
----               8      RSEG ?DT?func2?module2
0000               9      func2_var:   DS   5            ; space for local variables in func1
                  10
----              11      RSEG ?PR?func2?module2
0000 F500     F   12      func2:       MOV func2_var,A
0002 22           13      RET
                  14
                  15      ?BI?module2 SEGMENT BIT
----              16      RSEG ?BI?module2
0000              17      mybit:            DBIT   1
                  18
----              19      CSEG AT 03H                    ; EXT0 interrupt vector
0003 D200     F   20      SETB   mybit      ; interrupt function code
0005 32           21      RETI
                  22
----              23      CSEG AT 0BH                    ; Timer 0 interrupt vector
000B 020000   F   24      LJMP   timer0isr
                  25
                  26      ?BI?timer0?isr_module  SEGMENT BIT OVERLAYABLE ; bit segment
----              27      RSEG ?BI?timer0?isr_module
0000              28      isrbit:            DBIT   1
                  29
                  30      ?PR?timer0?isr_module  SEGMENT CODE  ; program code
----              31      RSEG ?PR?timer0?isr_module
                  32
0000              33      timer0isr:                     ; put your program code here
0000 D200     F   34      SETB   isrbit     ; interrupt function code
0002 32           35      RETI
                  36
                  37      END
```

Dummy C Module to Enable Data Overlaying and Code Banking

```
C51 COMPILER 6.00, COMPILATION OF MODULE DUMMY
OBJECT MODULE PLACED IN .\dummy.OBJ
COMPILER INVOKED BY: C:\Keil\C51\BIN\C51.EXE .\dummy.c OBJECTEXTEND DEBUG

stmt level    source

   1          /* this is a dummy C51 file to enable
   2             BL51 overlay and banking features */
```

BL51 Linker/Locater Memory Map File (*.M51)

This file shows the memory structure of your application. Within this map file you find the OVERLAY MAP that shows you the program structure as seen by the linker/locater.

```
BL51 BANKED LINKER/LOCATER V4.00a, INVOKED BY:
C:\KEIL\C51\BIN\BL51.EXE module1.obj, module2.obj, start.obj, dummy.obj TO app149 RAMSIZE (256)

MEMORY MODEL: SMALL

INPUT MODULES INCLUDED:
  module1.obj (MODULE1)
  module2.obj (MODULE2)
  start.obj (START)
  dummy.obj (DUMMY)


LINK MAP OF MODULE:  app149 (MODULE1)


            TYPE    BASE      LENGTH    RELOCATION   SEGMENT NAME
            -----------------------------------------------------

            * * * * * * *   D A T A   M E M O R Y   * * * * * * *
            REG     0000H     0008H     ABSOLUTE     "REG BANK 0"
            DATA    0008H     000AH     UNIT         _DATA_GROUP_
                    0012H     000EH                  *** GAP ***
            BIT     0020H.0   0000H.1   UNIT         ?BI?MODULE2
            BIT     0020H.1   0000H.1   UNIT         _BIT_GROUP_
                    0020H.2   0000H.6                *** GAP ***
            IDATA   0021H     0001H     UNIT         ?STACK

            * * * * * * *   C O D E   M E M O R Y   * * * * * * *
            CODE    0000H     0003H     ABSOLUTE
            CODE    0003H     0003H     ABSOLUTE
            CODE    0006H     0003H     UNIT         ?PR?FUNC1?MODULE1
                    0009H     0002H                  *** GAP ***
            CODE    000BH     0003H     ABSOLUTE
            CODE    000EH     0008H     UNIT         ?PR?MAIN?MODULE1
            CODE    0016H     0006H     UNIT         ?C_C51STARTUP
            CODE    001CH     0003H     UNIT         ?PR?FUNC2?MODULE2
            CODE    001FH     0003H     UNIT         ?PR?TIMER0?ISR_MODULE


OVERLAY MAP OF MODULE:   app149 (MODULE1)

SEGMENT                            BIT_GROUP          DATA_GROUP
  +--> CALLED SEGMENT           START    LENGTH     START    LENGTH
------------------------------------------------------------------
?PR?TIMER0?ISR_MODULE           0020H.1  0000H.1    -----    -----

*** NEW ROOT ***************************************************

?C_C51STARTUP                   -----    -----      -----    -----
  +--> ?PR?MAIN?MODULE1

?PR?MAIN?MODULE1                -----    -----      -----    -----
  +--> ?PR?FUNC1?MODULE1
  +--> ?PR?FUNC2?MODULE2

?PR?FUNC1?MODULE1               -----    -----      0008H    000AH

?PR?FUNC2?MODULE2               -----    -----      0008H    0005H



SYMBOL TABLE OF MODULE:  app149 (MODULE1)

  VALUE           TYPE          NAME
  ----------------------------------

  -------         MODULE        MODULE1
  C:000EH         SEGMENT       ?PR?MAIN?MODULE1
  C:0006H         SEGMENT       ?PR?FUNC1?MODULE1
  D:0008H         SEGMENT       ?DT?FUNC1?MODULE1
  C:000EH         PUBLIC        MAIN
  C:0006H         SYMBOL        FUNC1
  D:0008H         SYMBOL        FUNC1_VAR
  C:000EH         LINE#         8
  C:0011H         LINE#         9
  C:0014H         LINE#         10
  C:0006H         LINE#         19
  C:0008H         LINE#         20
  -------         ENDMOD        MODULE1

  -------         MODULE        MODULE2
  C:001CH         SEGMENT       ?PR?FUNC2?MODULE2
  D:0008H         SEGMENT       ?DT?FUNC2?MODULE2
  B:0020H.0       SEGMENT       ?BI?MODULE2
  B:0020H.1       SEGMENT       ?BI?TIMER0?ISR_MODULE
  C:001FH         SEGMENT       ?PR?TIMER0?ISR_MODULE
  C:001CH         PUBLIC        FUNC2
  D:0008H         SYMBOL        FUNC2_VAR
  B:0020H.1       SYMBOL        ISRBIT
  B:0020H.0       SYMBOL        MYBIT
  C:001FH         SYMBOL        TIMER0ISR
  C:001CH         LINE#         12
  C:001EH         LINE#         13
  C:0003H         LINE#         20
  C:0005H         LINE#         21
  C:000BH         LINE#         24
  C:001FH         LINE#         34
  C:0021H         LINE#         35
  -------         ENDMOD        MODULE2

  -------         MODULE        START
  C:0016H         SEGMENT       ?C_C51STARTUP
  I:0021H         SEGMENT       ?STACK
  D:0081H         SYMBOL        SP
  C:0016H         SYMBOL        STARTUP1
  C:0000H         LINE#         10
  C:0016H         LINE#         12
  C:0019H         LINE#         13
  -------         ENDMOD        START

  -------         MODULE        DUMMY
  C:0000H         SYMBOL        _ICE_DUMMY_
  -------         ENDMOD        DUMMY

LINK/LOCATE RUN COMPLETE.  0 WARNING(S),  0 ERROR(S)
```