User Manual

DA14580 Memory Map and Scatter File

UM-B-011

Abstract

This document contains information about the memory map of the DA14580-01. It also contains information about setting up the proper parameters in order to use the common scatter file provided by Dialog Semiconductor for building a custom application

Contents

Figures

Tables

1 Terms and definitions

BLE	Bluetooth® Low Energy
OTP	One Time Programmable memory
RAM	Random Access Memory
Retention RAM	Memory that is powered when DA14580 is in Deep sleep
RO section	Memory section where code (RO) is placed
ROM	Read Only Memory
RW section	Memory section where global initialized variables are placed
Scatter file	Text file used to specify the memory map of an image to the linker
SPOTAR	Software Patch Over the Air – Receiver
SysRAM	System RAM
ZI section	Memory section where zero initialized variables are placed

2 References

1. DA14580 DataSheet, Dialog Semiconductor
2. UM-B-014, DA14580 Development Kit, Dialog Semiconductor
3. UM-B-006, DA14580 Sleep mode configuration, Dialog Semiconductor

3 Introduction

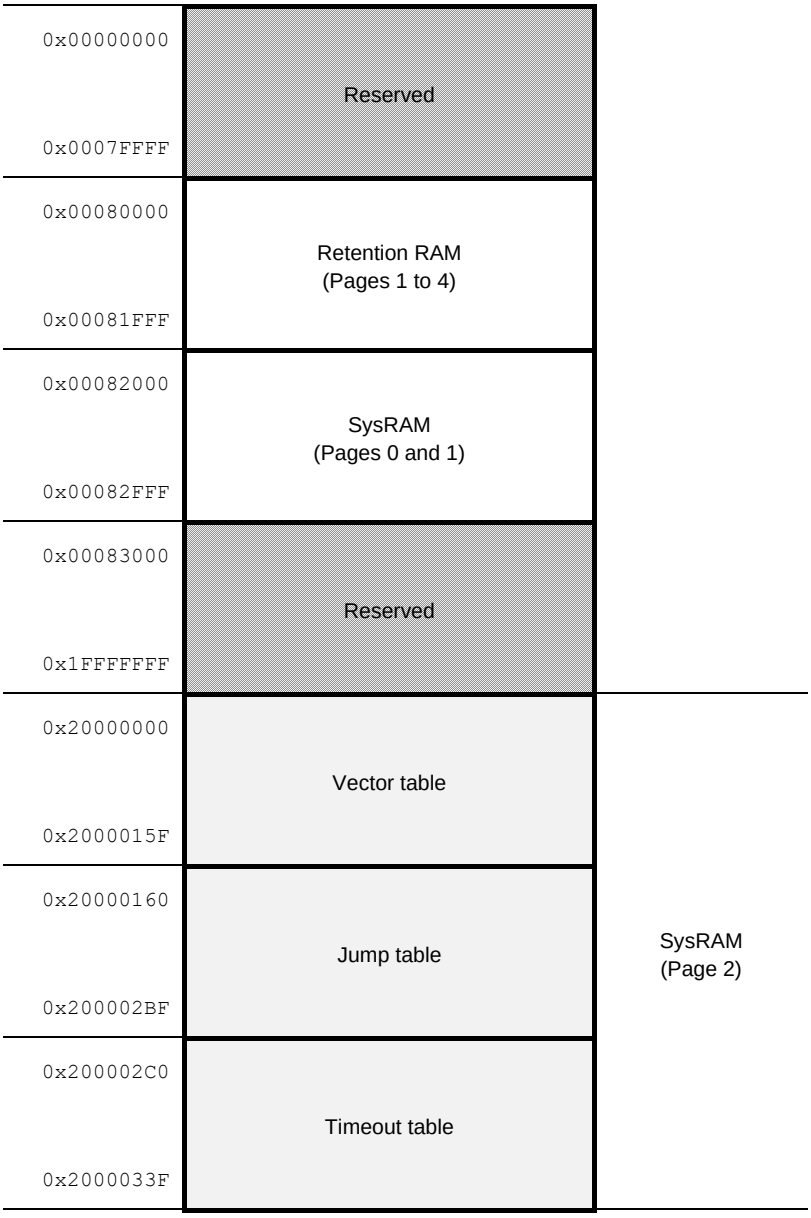
This document describes the memory map of the software running on DA14580 and the structure of the common scatter file provided by Dialog Semiconductor.

It is assumed that “Case 23” is used (EM_MAPPING in BLE_CNTL2_REG – default setting) and no remapping has been executed (application code is placed at the section starting from 0x20000000). If the user wants to use a different configuration (i.e. use remapping or a different memory setup than “Case 23”) then this document can be used as a basis to create a custom scatter file.

The reader interested in how to use the Excel tool and configure the common scatter file can skip sections 4 and 5.

4 Memory map

The memory map of the software running on DA14580 is shown below:



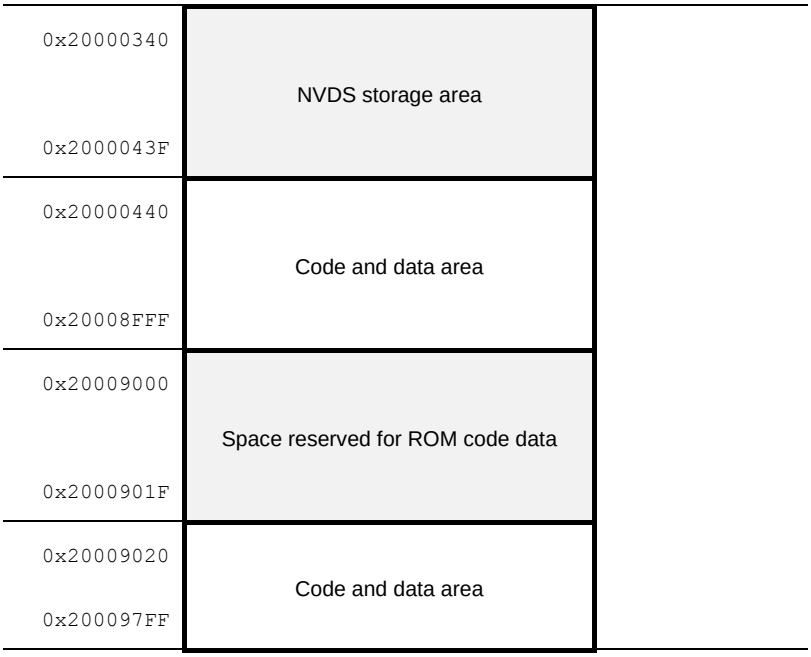


Figure 1: High-level memory map

As shown in Figure 1, areas 0x0 to 0x7FFFF and 0x83000 to 0x1FFFFFFF are reserved and cannot be used. In 0x80000 to 0x81FFF the Retention RAM is located. Above it, at 0x82000 to 0x82FFF there is 4 KB of RAM space that is not retained when the chip goes into Deep Sleep.

From 0x20000000 to 0x200097FF there is 38 KB of RAM space (SysRAM). Some areas in this memory space are reserved, though, and cannot be used by the user. These are:

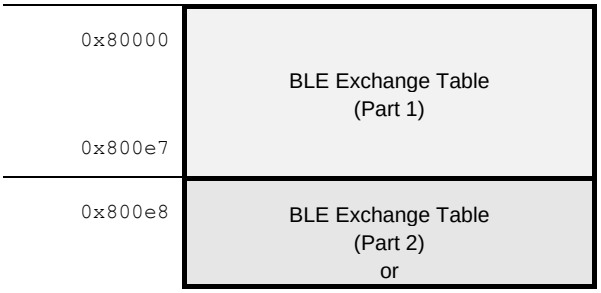
- Vector table, placed at 0x20000000,
- Jump table, placed at 0x20000160,
- Timeout table, placed at 0x200002C0,
- NVDS storage, placed at 0x20000340 and
- ROM code data, placed at 0x20009000.

In the remaining memory space (approximately 37 KB) the application's code and data can be placed. Note that the size of the code and the initialized data is limited by the OTP size, which is 32 KB. There is also space left in the area 0x80000 to 0x82FFF for data storage as well, as will be explained in detail later in the document.

Before describing the procedure with which the various sections of code and data are placed in the respective memory areas, a bigger insight on the memory allocation in the area from 0x80000 to 0x82FFF will be given.

4.1 Memory allocation inside 0x80000 to 0x82FFF

The detailed memory allocation in the areas 0x80000 to 0x81FFF (Retention RAM) and 0x82000 to 0x82FFF (SysRAM pages 0 and 1) is given below:



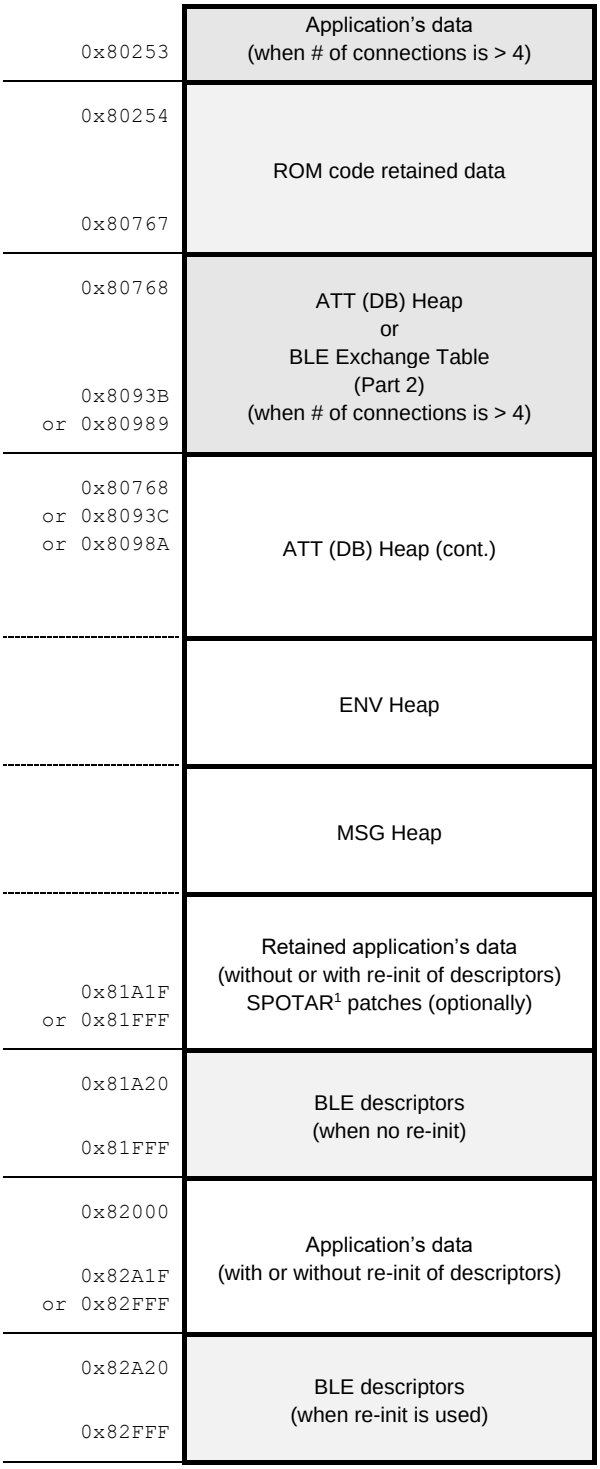


Figure 2: Detailed memory allocations in 0x80000 to 0x82FFF

As shown in the figure above, some memory areas are reserved (marked as grey). More specifically,

- The area 0x80000 to 0x80767 is, in general, reserved. In this are placed the Exchange Table and the ROM code retained variables. When the number of connections the application is built for (BLE_CONNECTION_MAX_USER) is greater than 4, a part of the exchange table is moved outside this area to the area starting from 0x80768. This leaves a "hole" from 0x800e8 to 0x800253 (364 bytes) that can be used for placing retained data the application uses.

[1] Please refer to the *UM-B-007,* "*Software Patch over the Air*" document for more details about SPotA.

- Following the point above, the area 0x80768 to 0x8093B is reserved for the Exchange Table when BLE_CONNECTION_MAX_USER is 5. When it is 6, then the area 0x80768 to 0x80989 is reserved for this purpose.
- BLE's descriptors have to be placed in this memory area. There are two options. One is to place them in the Retention RAM (0x80000 to 0x81FFF) and the other is to place them in the SysRAM pages 0 and 1 (0x82000 to 0x82FFF). When they are not placed in the Retention RAM and the chip goes into Deep Sleep, when exiting from sleep then the descriptors have to be re-initialized before the BLE can use them. This is done automatically and is transparent to the application code writer. If Extended Sleep is used, then all memory is retained during sleep and there is no need for re-initializing the descriptors. The decision which option to use depends on the Retention memory requirements since the re-initialization of descriptors implies a computational overhead. For example, if the size of application's retained data becomes too big or the heaps' sizes increase too much, leaving no space for the application's data, in may be necessary to re-initialize the descriptors.

Inside the remaining part of the Retention RAM the application's retained data are placed, any SPOTAR patches and three heaps that the BLE stack uses. The size of these varies. The size of application's retained data depends on the specific application. The size of each heap depends on the number of supported connections (BLE_CONNECTION_MAX_USER). The following table shows the size of each heap for all possible settings of BLE_CONNECTION_MAX_USER.

Table 1: Sizes of ATT (DB), ENV and MSG heaps

BLE_CONNECTION_MAX_USER	1	2	3	4	5	6
ATT (DB)	1036	1036	1036	1036	1036	1036
ENV	340	668	996	1324	1652	1980
MSG	1324	2028	2732	3436	4140	4844

Based on the above, a table can be constructed that gives the amount or Retention RAM that is left for the application data depending on the number or connections the code is built for and whether re-initialization of BLE descriptors is applied or not. This table is shown below.

Table 2: Available space in Retention RAM for application data

BLE_CONNECTION_MAX_USER		1	2	3	4	5	6
Without re-init	Space in 0x800E8 to 0x80253	0	0	0	0	364	364
	Space in 0x80768 to 0x81FFF	2092	1060	28	*-1004*	*-2504*	*-3636*
With re-init	Space in 0x800E8 to 0x80253	0	0	0	0	364	364
	Space in 0x80768 to 0x81FFF	3596	2564	1532	500	*-1000*	*-2132*

From this table is obvious that in order to support Deep Sleep with 4 connections, re-initialization of BLE descriptors is mandatory. It is also obvious that 5 or 6 connections cannot be supported even with re-initialization of descriptors since the Retention RAM is not big enough to hold all the data[2]. There is a provision for this, though.

Apart from these heaps that have to be placed into the Retention RAM, the BLE Stack uses another heap that is placed in the SysRAM. The purpose is to have a "last-resort", a pool of memory that can be used when there is no memory left in the retained heaps. Of course, if memory is allocated from the "non-retained heap", the chip will not be able to go into Deep Sleep until this memory is freed and

[2] Note that whether Deep sleep will eventually be used by a master with 5 or 6 active connections depends on the application's usage scenario. If the connection interval of all connected devices is big enough to benefit from Deep sleep then the application could enable it. The common scatter file is built to allow this option.

the heap is empty. This is done automatically and is transparent to the application code writer. As mentioned before, there is no such restriction when Extended Sleep is used.

Thus, in order to overcome the lack or Retention RAM for 5 and 6 connections, smaller MSG heap can be used so that it fits in Retention RAM by increasing accordingly the size of the non-retained heap. Since the size of the MSG heap is estimated for the worst case utilization scenario that is quite unlikely to happen, there is a very good chance that the chip will be able to go into Deep Sleep even when 5 or 6 connections are active at the same time.

The above statement can be extended to take into account the application's retained data requirements as well. For example, even if the application is built for up to 6 connections, the size of MSG heap could be set to support only 2 so as to allow for application's data to be placed into the Retention RAM.

Dialog Semiconductor provides an Excel tool that can be used to determine the optimum memory layout based on the application's requirements. This tool is described later in this document.

5 ARM scatter files

The purpose of this section is to provide some basic information about ARM scatter files. Detailed information is provided online at http://www.keil.com.

The scatter file instructs the Linker where to place code and data. It is comprised by Load Region descriptors and Execution Region descriptors as shown below.

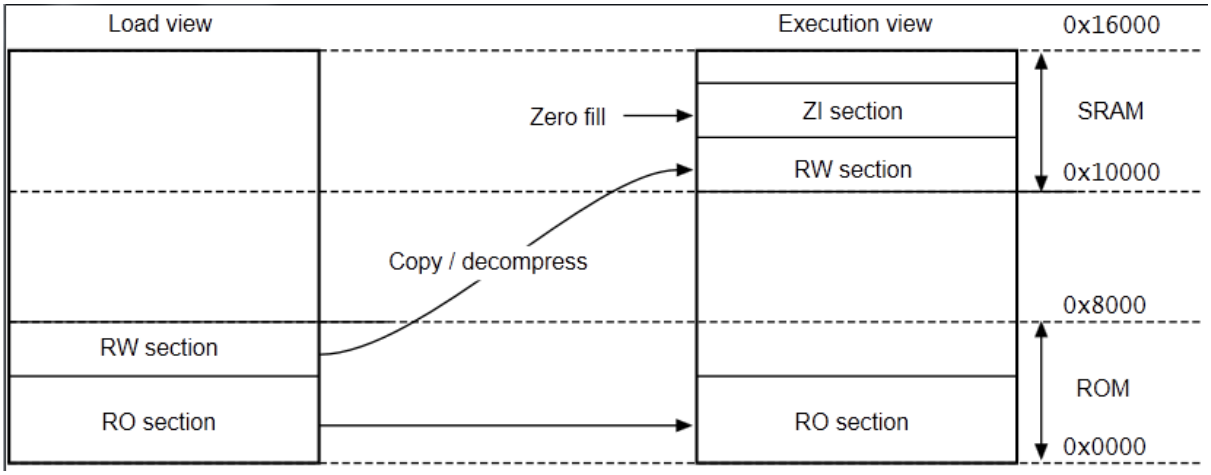


Figure 3: Load and Execution Regions[3]

A Load Region instructs the linker where to place code and data and the initialization code where to load the code or data from. Before the code reaches main_func() in arch_main.c, the initialization code is executed. During initialization code and data will be copied, if necessary, from the Load Regions to the Execution Regions. Thus, if, for example, a FLASH is used to store code and data then this architecture allows for the seamless transfer of data and/or code areas to the system RAM.

For DA14580 this functionality is not required since copying from OTP, which is done during system power-up or when exiting Deep Sleep, is done by the bootrom code and this procedure is completely transparent to the application code. Because of this, Load Regions in DA14580's scatter files match their respective Execution Regions, at least for the memory range 0x20000000 to 0x200097FF. The rest of the Load Regions should contain only zero initialized data. The area 0x80000 to 0x82FFF should be declared as one that contains data that do not have to be initialized (UNINIT). Since this area is zeroed when the chip boots, zero-init data can be placed in here. Data compression must be disabled for all regions.

The structure of a scatter file is shown in the image below.

[3] Image taken from http://www.keil.com/support/man/docs/armlink/armlink_Chdfgiaj.htm

```
LOAD_ROM 0x0000 0x8000       ; Name of load region (LOAD_ROM),
                             ; Start address for load region (0x0000),
                             ; Maximum size of load region (0x8000)
{
    EXEC_ROM 0x0000 0x8000   ; Name of first exec region (EXEC_ROM),
                             ; Start address for exec region (0x0000),
                             ; Maximum size of first exec region (0x8000)
    {
        * (+RO)              ; Place all code and RO data into
                             ; this exec region
    }

    SRAM 0x10000 0x6000      ; Name of second exec region (RAM),
                             ; Start address of second exec region (0x10000),
                             ; Maximum size of second exec region (0x6000)
    {
        * (+RW, +ZI)         ; Place all RW and ZI data into
                             ; this exec region
    }
}
```

Figure 4: Basic structure of a scatter file[4]

There is one Load Region and two Execution Regions. In the area 0x0000 to 0x8000 the code is placed while in the area 0x10000 to 0x16000 RW and ZI data are placed. Since the Load Region and the Execution Region EXEC_ROM have the same address range, no copying of code is done. The code will be executed from the ROM which starts at 0x0000 and ends at 0x8000. Any RW data in there though, will be copied (and, in general, decompressed) to the SRAM Execution Region which starts at 0x10000 and ends at 0x16000.

If the application code intends to place some data at a specific memory area then it can use two approaches:

- Declare the variable to belong to a specific section, i.e. “foo” by using the convention:
  `int variable __attribute__((section("foo"))) = 10; (in file.c)`
  In this case, section “foo” has to be defined in the scatter file as shown below5.

```
FLASH 0x24000000 0x4000000
{
    ...                              ; rest of code

    ADDER 0x08000000
    {
        file.o (foo)                 ; select section foo from file.o
    }
}
```

- Explicitly place it at a specific memory location, i.e. 0x8000, using:
  `int variable1 __attribute__((at(0x8000))) = 10;`

Both approaches are used by the SDK. More specifically,

- BLE descriptors are placed using the “at” attribute. See the declaration of descript in arch_main.c.
- Jump table, Timeout table, NVDS storage, heaps etc. use the first approach. Each one is declared to belong to its own specific section. For example, all the application’s retained data belong to a section named “retention_mem_area0”. Another section named “retention_mem_area1” is declared for the extra space created in 0x820e8 to 0x80253 when BLE_CONNECTION_MAX_USER is 5 or 6. It is the scatter file’s responsibility to place these sections in their respective memory areas.

[4] Image taken from http://www.keil.com/support/man/docs/armlink/armlink_Chdfgiaj.htm
[5] Taken from http://www.keil.com/support/man/docs/armlink/armlink_BABHIIEF.htm

6 Memory map tool

Dialog Semiconductor provides an Excel tool (DA14580 MemoryMapTool.xlsm) that can be used to determine the optimum placement of the various sections in memory that satisfies the application's requirements.

The tool comprises of 4 sheets. Only the first sheet is of interest to the user. This sheet includes both the section where the user enters the input parameters and the section with the results.

6.1 Input parameters

The image below shows the parameters the user can input to the tool.

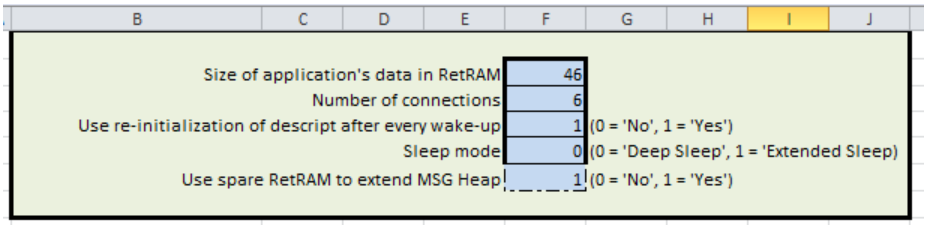


Figure 5: Input parameters to the Excel tool

The user can enter the following parameters:

- The size of the application's data that have to be placed into the Retention RAM and, more specifically, into the section "retention_mem_area0" plus the size of SPOTAR patches, if any. Note that any gaps created due to alignment of data by the Linker have to be counted in as well. It is suggested that the user uses initially a large enough value for this input. After building the application, open file full_emb_sysram.map and find a section that resembles what is shown below.

```
rwip_rf                         0x0008071c   Data          0  rom_symdef.txt ABSOLUTE
cs_area$$Base                   0x00080768   Number        0  arch_main.o(cs_area)
cs_table                        0x00080768   Data        546  arch_main.o(cs_area)
rwip_heap_db_ret                0x0008098c   Data       1036  jump_table.o(heap_db_area)
rwip_heap_env_ret               0x00080d98   Data       1980  jump_table.o(heap_env_area)
rwip_heap_msg_ret               0x00081554   Data       2684  jump_table.o(heap_msg_area)
sys_startup_flag                0x00081fd0   Data          1  arch_main.o(retention_mem_area0)
cal_enable                      0x00081fd4   Data          1  arch_system.o(retention_mem_area0)
lp_clk_sel                      0x00081fd8   Data          4  arch_system.o(retention_mem_area0)
rcx_freq                        0x00081fdc   Data          4  arch_system.o(retention_mem_area0)
dev_bdaddr                      0x00081fe0   Data          6  nvds.o(retention_mem_area0)
disc_envs                       0x00081fe8   Data          4  disc.o(retention_mem_area0)
disc_state                      0x00081fec   Data          6  disc_task.o(retention_mem_area0)
proxm_envs                      0x00081ff4   Data          4  proxm.o(retention_mem_area0)
proxm_state                     0x00081ff8   Data          6  proxm_task.o(retention_mem_area0)
descript                        0x00082a20   Data       1502  arch_main.o(.ARM.__AT_0x00082A20)
__Vectors                       0x20000000   Data          4  boot_vectors.o(RESET)
__Vectors_End                   0x200000a0   Data          0  boot_vectors.o(RESET)
jump_table_base                 0x20000160   Data        352  jump_table.o(jump_table_mem_area)
gap_timeout_table               0x200002c0   Data        100  gapm.o(timeout table area)
```

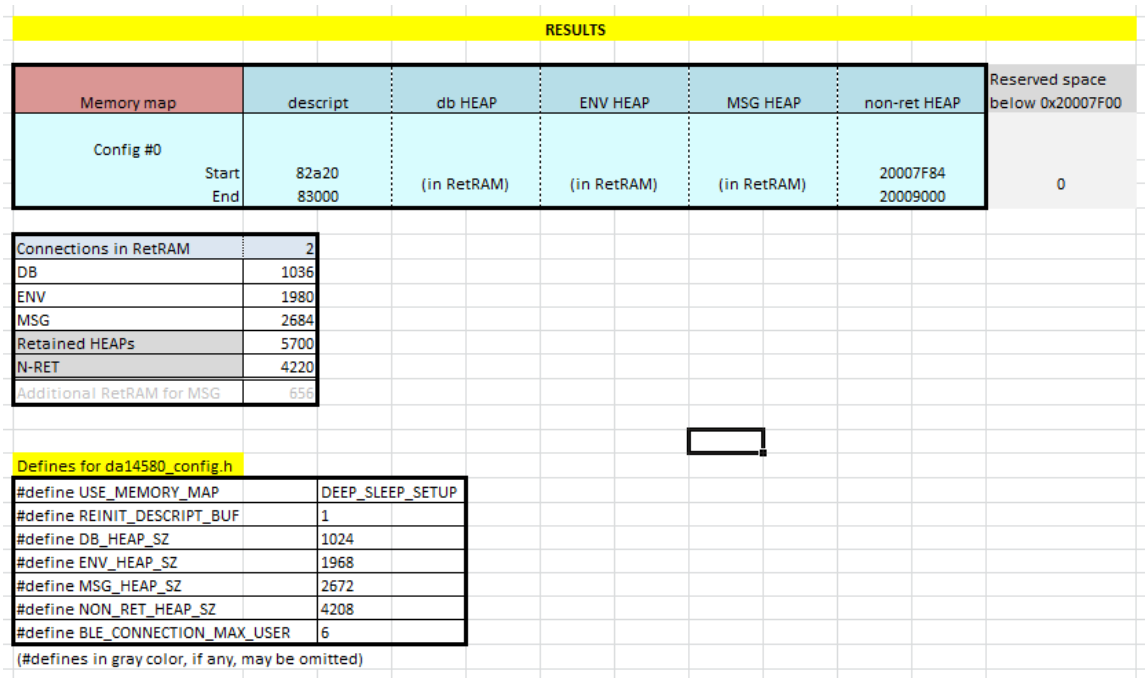
  In this example, "retention_mem_area0" starts at 0x81FD0 and ends at 0x81FF8+6. Thus, the total size is 46 bytes. After determining the correct size, the user can add the size that will be used for SPOTAR patches (if SPOTAR is included) and enter the resulting value to the tool and get the optimum memory configuration.

- The maximum number of connections the application is built for. This is the same as BLE_CONNECTION_MAX_USER. If it is a slave then this value is 1. If it is a master device then it can be from 1 to 6. This parameter affects the sizes of the heaps.
- Whether re-initialization of descriptors will be used or not. Note that the tool will explicitly set this parameter to 1 ("Yes") if there's no possible memory configuration without re-initialization of BLE descriptors that can satisfy the given user parameters. Also note that the tool will ask the user to enable this feature if there is a memory configuration with re-initialization that gives a more optimal setup than without it. In this case, the user may decide depending on whether the overhead of re-initialization is accepted or not. This setting is relevant only if Deep Sleep is used.

- The sleep mode, which is deep or extended sleep. In case of extended sleep no special provisions are required as will be explained later when the respective memory setup is presented.
- In case of Deep Sleep, the resulting setup may use a smaller MSG heap due to the limited Retention RAM size. In this case, the user is offered the option to use any spare memory area inside Retention RAM in order to extend MSG heap (by reducing non-retained heap accordingly). This way the possibility of MSG heap depletion is reduced.

6.2 Results

The image below depicts the area that includes the results of the tool.



RESULTS

Memory map	descript	db HEAP	ENV HEAP	MSG HEAP	non-ret HEAP	Reserved space below 0x20007F00
Config #0						
Start	82a20	(in RetRAM)	(in RetRAM)	(in RetRAM)	20007F84	
End	83000				20009000	0

Connections in RetRAM	2
DB	1036
ENV	1980
MSG	2684
Retained HEAPs	5700
N-RET	4220
Additional RetRAM for MSG	656

Defines for da14580_config.h

#define USE_MEMORY_MAP	DEEP_SLEEP_SETUP
#define REINIT_DESCRIPT_BUF	1
#define DB_HEAP_SZ	1024
#define ENV_HEAP_SZ	1968
#define MSG_HEAP_SZ	2672
#define NON_RET_HEAP_SZ	4208
#define BLE_CONNECTION_MAX_USER	6

(#defines in gray color, if any, may be omitted)

Figure 6: Results of the Excel tool

This area comprises of three sections. The first section shows where the heaps and the BLE descriptors are placed. In this specific example, the descriptors are placed at 0x82A20 to 0x82FFF (so, re-initialization will be done), the retained heaps are placed inside the Retention RAM and the non-retained heap is placed at 0x20007F84 to 0x20008FFF. The next cell shows the memory space reserved below 0x20007F00. This is important since the area 0x20000000 to 0x20007EFF maps directly to the OTP[6]. Thus, any memory reservation for heap data below 0x20007F00 results in reduction of space available for code (RO) and initialized data (RW).

The next section shows the calculated sizes of the heaps and the number of concurrent connections that can be supported while the system enters into deep sleep, with zero probability of heap depletion. In this example this number is 2. In other words, the tool calculated that there's space for MSG heap in the Retention RAM only to support 2 connections, although the user set BLE_CONNECTION_MAX_USER to 6. As was mentioned before, the size of the MSG heap is calculated for the worst case scenario that is quite unlikely to happen. So, it is highly possible that even 6 connections could be supported with Deep Sleep with this configuration. This section also

[6] As was discussed previously, a scatter file instructs the initialization code to copy data from the Load Regions to the respective Execution Regions. Thus, in principal, in the DA14580 case, code could lie outside 0x20000000 to 0x20007EFF. Instead of directly mapping OTP to this area, the initialization code could copy the data to addresses above 0x20007EFF. This is not possible though. OTP copying is not done by the initialization code but from the bootrom code. Furthermore, the initialization code runs only once, during boot. When the system exits deep sleep it resumes execution from the point it was before sleeping and the initialization code is not executed. Thus, if a code relocation scheme using the initialization code was used, this would work only once, during boot. It would fail though at exit from deep sleep.

shows the size of spare space in the Retention RAM that was used to extend MSG heap, if the user has enabled this option. In this example this is 656 bytes.

The last section contains the #defines that have to be entered by the user to the file da14580_config.h in order to build the code with the optimum memory configuration. When Deep Sleep is used, the #defines for heaps' sizes are mandatory and must be added to da14580_config.h. When Extended Sleep is used these may be omitted since their sizes are calculated automatically by the code. For this reason, these #defines are greyed out when Extended Sleep is chosen.

6.3 Examples

In all examples below SPOTAR is not used.

6.3.1 Deep Sleep with 1 connection

In this example, the user has built an application that has only 46 bytes of data to be placed at the Retention RAM and is for a slave device that is able to enter into Deep Sleep. The user uses the Excel tool so as to find the proper settings.

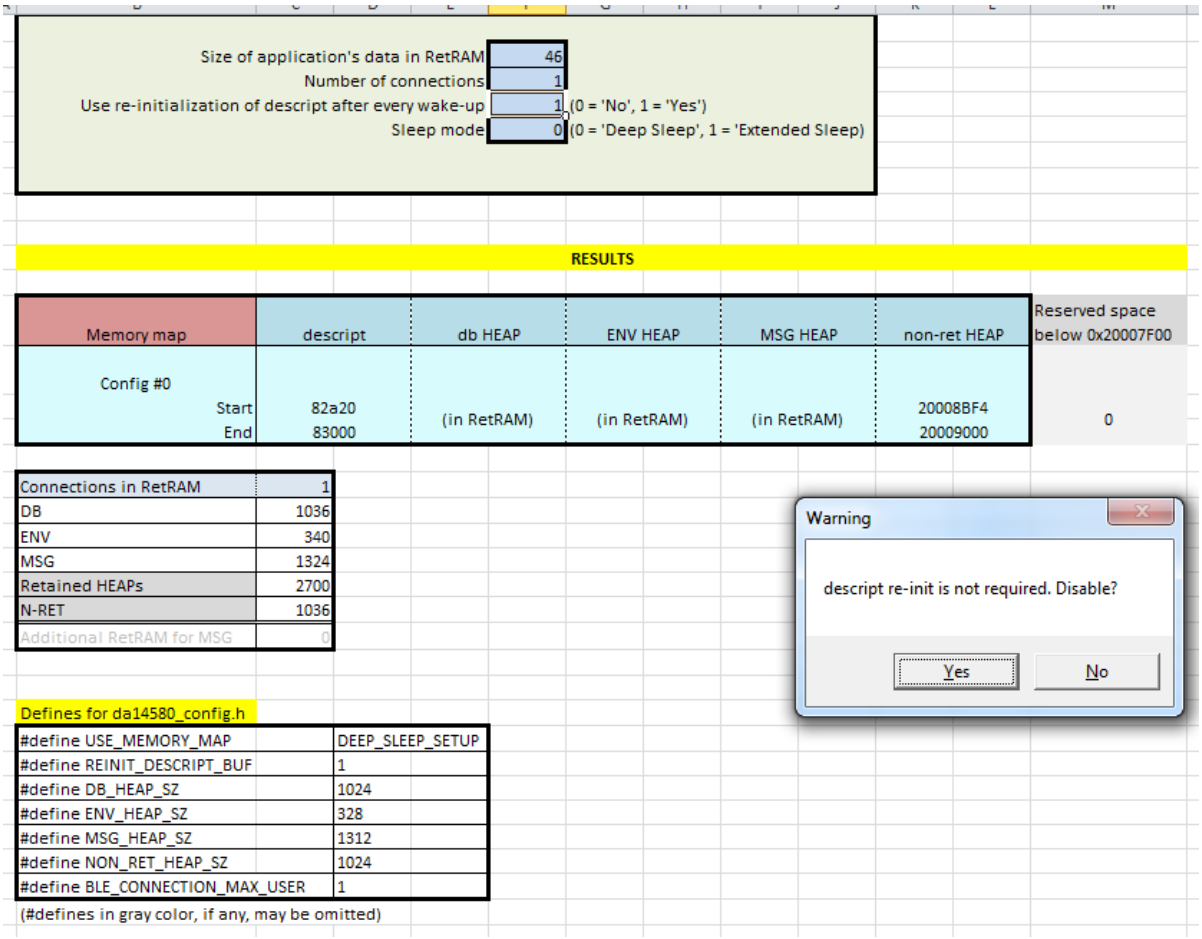
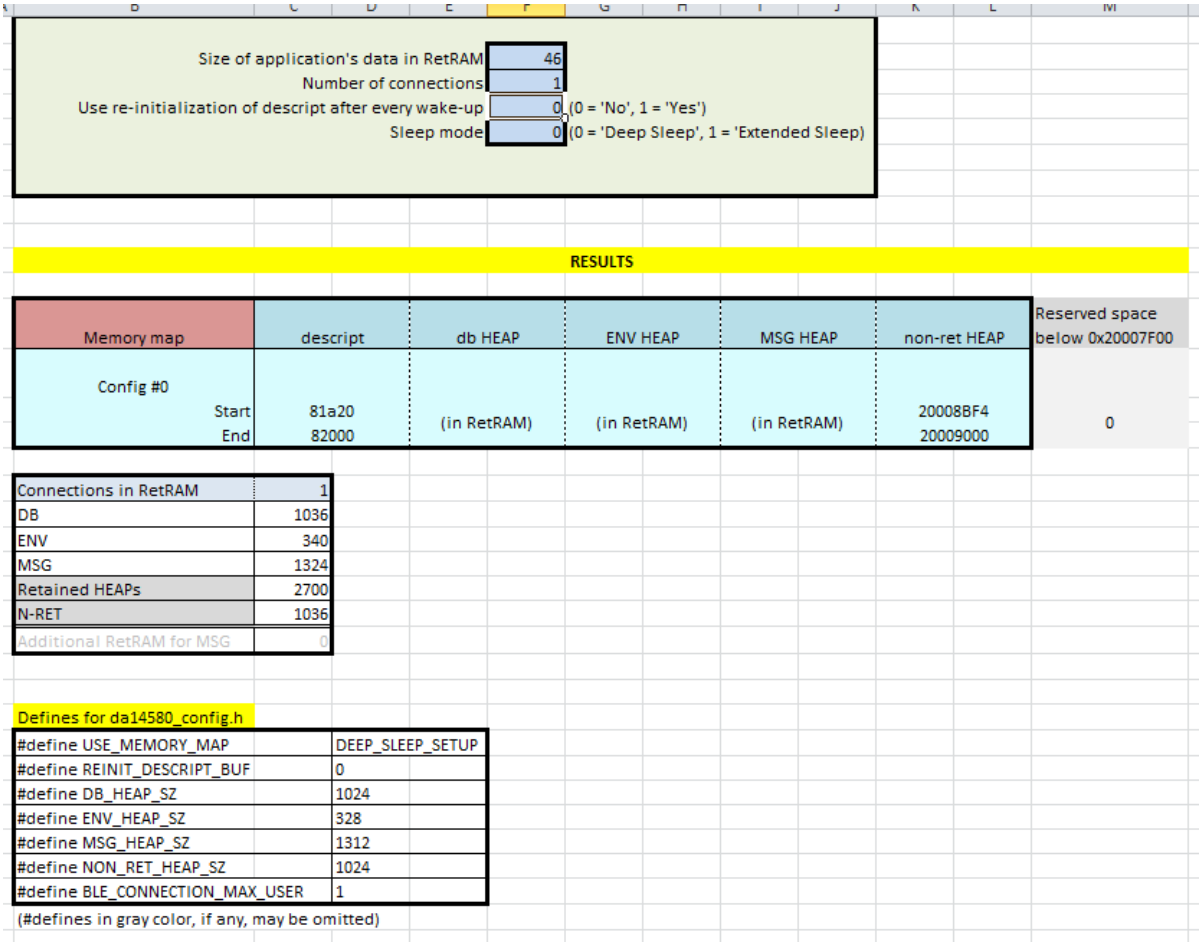


Figure 7: Re-initialization of descriptors is not required

As shown in Figure 7, a pop-up window is displayed automatically because the user has requested re-initialization of BLE descriptors although this is not necessary.

After replying "Yes", the tool calculates the optimum memory configuration. The results are shown below. The user is not offered the option to extend MSG heap by using the spare space of Retention RAM because there's no need for it. The MSG heap has been dimensioned so that it can support the requested number of connections (1 in this case).



Size of application's data in RetRAM	46
Number of connections	1
Use re-initialization of descript after every wake-up	0 (0 = 'No', 1 = 'Yes')
Sleep mode	0 (0 = 'Deep Sleep', 1 = 'Extended Sleep)

RESULTS

Memory map	descript	db HEAP	ENV HEAP	MSG HEAP	non-ret HEAP	Reserved space below 0x20007F00
Config #0						
Start	81a20	(in RetRAM)	(in RetRAM)	(in RetRAM)	20008BF4	0
End	82000				20009000	

Connections in RetRAM	1
DB	1036
ENV	340
MSG	1324
Retained HEAPs	2700
N-RET	1036
Additional RetRAM for MSG	0

Defines for da14580_config.h

#define USE_MEMORY_MAP	DEEP_SLEEP_SETUP
#define REINIT_DESCRIPT_BUF	0
#define DB_HEAP_SZ	1024
#define ENV_HEAP_SZ	328
#define MSG_HEAP_SZ	1312
#define NON_RET_HEAP_SZ	1024
#define BLE_CONNECTION_MAX_USER	1

(#defines in gray color, if any, may be omitted)

Figure 8: Results for Deep Sleep with 1 connection

6.3.2 Deep Sleep with 4 connections

In this example, the user has built an application that has only 46 bytes of data to be placed at the Retention RAM and is for a master device that must support up to 4 concurrent connections and be able to enter into Deep Sleep. The user uses the Excel tool so as to find the proper settings.

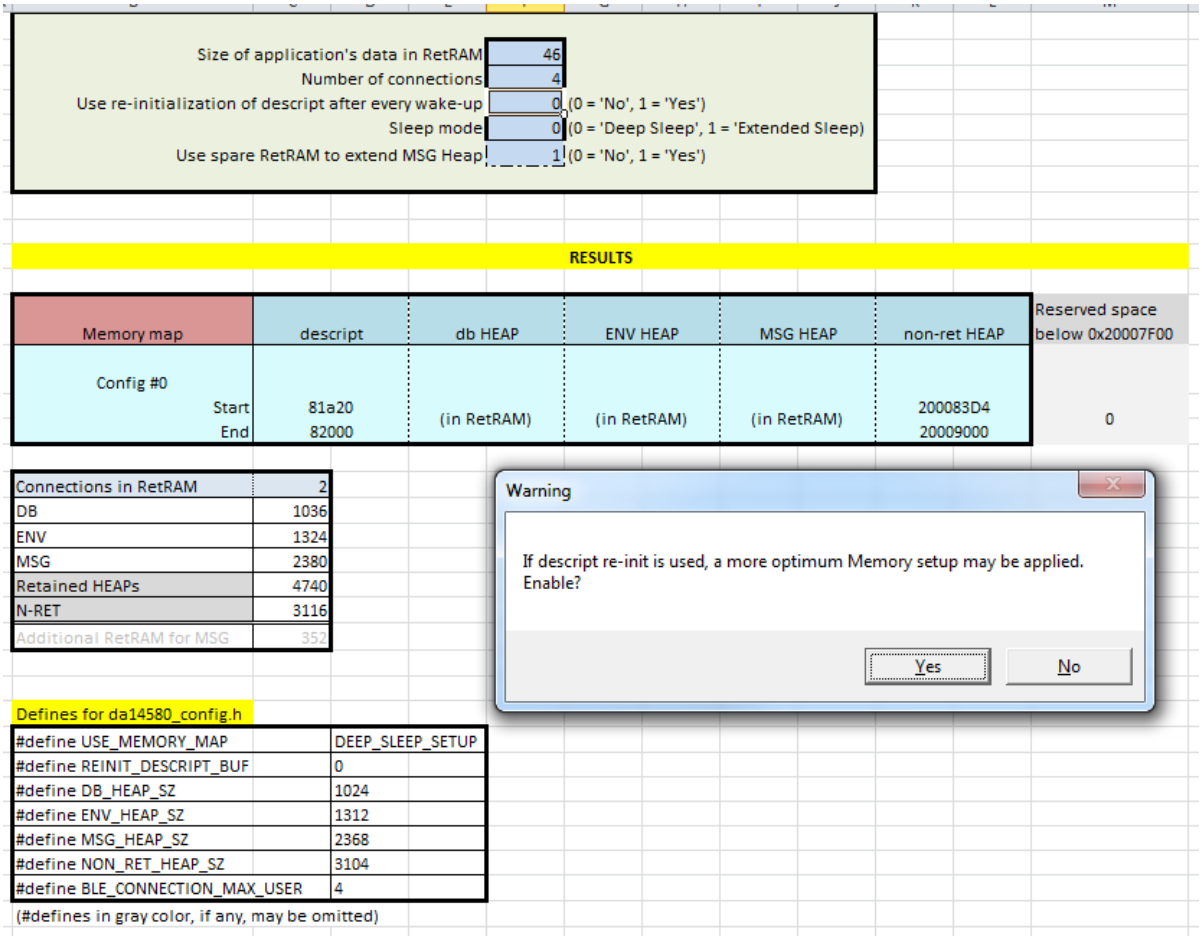
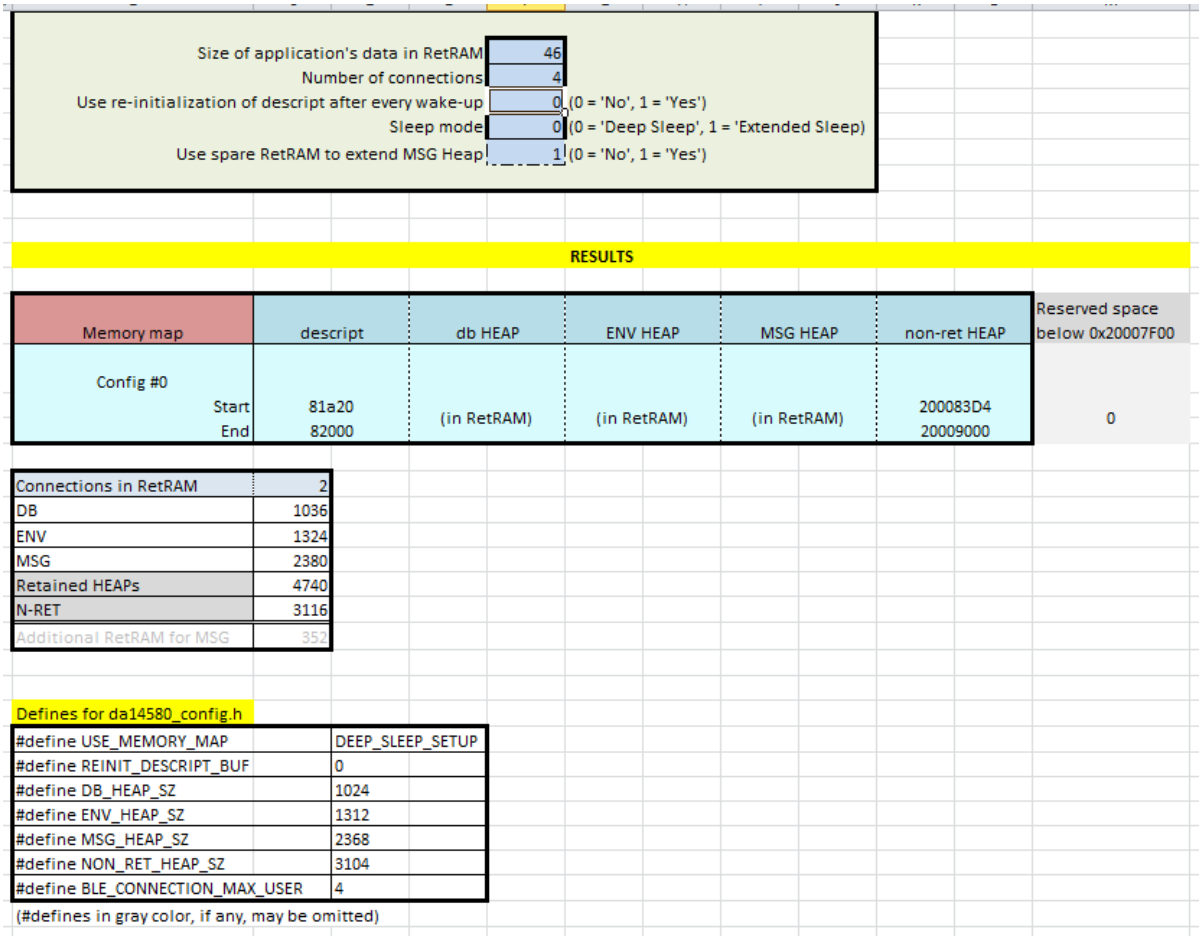


Figure 9: Re-initialization of descriptors is suggested

As shown in Figure 9, the user is offered the option to enable re-initialization of BLE descriptors. Let us see what happens if he chooses not to enable it.



Size of application's data in RetRAM	46	
Number of connections	4	
Use re-initialization of descript after every wake-up	0	(0 = 'No', 1 = 'Yes')
Sleep mode	0	(0 = 'Deep Sleep', 1 = 'Extended Sleep)
Use spare RetRAM to extend MSG Heap	1	(0 = 'No', 1 = 'Yes')

RESULTS

Memory map	descript	db HEAP	ENV HEAP	MSG HEAP	non-ret HEAP	Reserved space below 0x20007F00
Config #0						
Start	81a20	(in RetRAM)	(in RetRAM)	(in RetRAM)	200083D4	0
End	82000				20009000	

Connections in RetRAM	2
DB	1036
ENV	1324
MSG	2380
Retained HEAPs	4740
N-RET	3116
Additional RetRAM for MSG	352

Defines for da14580_config.h

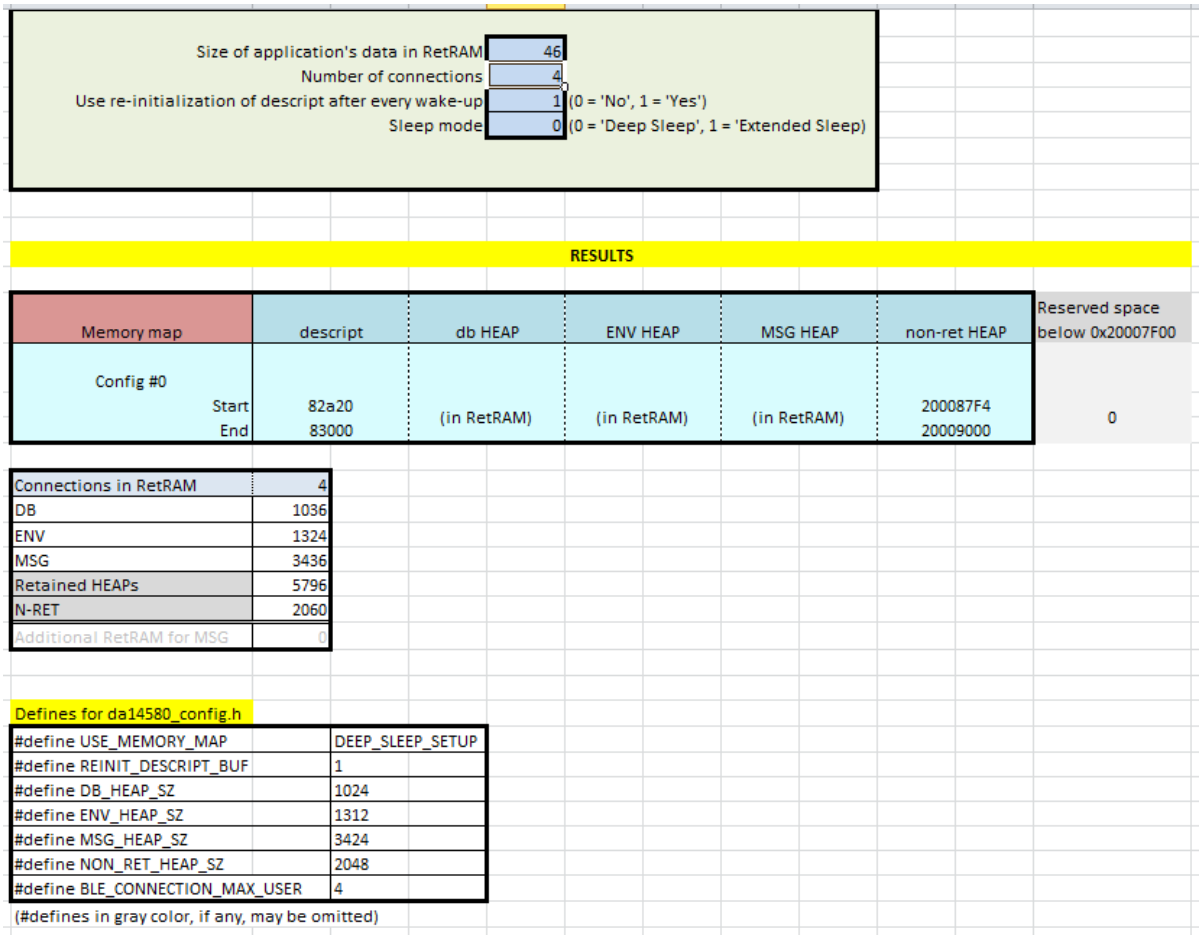
#define USE_MEMORY_MAP	DEEP_SLEEP_SETUP
#define REINIT_DESCRIPT_BUF	0
#define DB_HEAP_SZ	1024
#define ENV_HEAP_SZ	1312
#define MSG_HEAP_SZ	2368
#define NON_RET_HEAP_SZ	3104
#define BLE_CONNECTION_MAX_USER	4

(#defines in gray color, if any, may be omitted)

Figure 10: Master with 4 connections without re-initialization of descriptors

As depicted in the image above, the user chose not to enable the re-initialization of BLE descriptors and to use 352 bytes of spare space in Retention RAM to extend the size of MSG heap. This configuration supports 2 concurrent connections with zero probability of MSG heap depletion.

If the user enables the re-initialization of descriptors then the tool calculates the following results.



Size of application's data in RetRAM	46	
Number of connections	4	
Use re-initialization of descript after every wake-up	1	(0 = 'No', 1 = 'Yes')
Sleep mode	0	(0 = 'Deep Sleep', 1 = 'Extended Sleep)

RESULTS

Memory map	descript	db HEAP	ENV HEAP	MSG HEAP	non-ret HEAP	Reserved space below 0x20007F00
Config #0						
Start	82a20	(in RetRAM)	(in RetRAM)	(in RetRAM)	200087F4	0
End	83000				20009000	

Connections in RetRAM	4
DB	1036
ENV	1324
MSG	3436
Retained HEAPs	5796
N-RET	2060
Additional RetRAM for MSG	0

Defines for da14580_config.h

#define USE_MEMORY_MAP	DEEP_SLEEP_SETUP
#define REINIT_DESCRIPT_BUF	1
#define DB_HEAP_SZ	1024
#define ENV_HEAP_SZ	1312
#define MSG_HEAP_SZ	3424
#define NON_RET_HEAP_SZ	2048
#define BLE_CONNECTION_MAX_USER	4

(#defines in gray color, if any, may be omitted)

Figure 11: Master with 4 connections with re-initialization of descriptors

As shown in Figure 11, when re-initialization of BLE descriptors is used, 4 concurrent connections can be supported with zero probability of MSG heap depletion. This comes at a cost though. The cost is the overhead required for the re-initialization.

6.3.3 Deep Sleep with 6 connections

In this example, the user has built an application that has only 46 bytes of data to be placed at the Retention RAM and is for a master device that must support up to 6 concurrent connections and be able to enter into Deep Sleep. The user uses the Excel tool so as to find the proper settings.

As was discussed previously, there isn't any possible configuration that can support 6 connections in Deep Sleep without re-initialization of descriptors. Thus, the user is informed that the re-initialization will be enabled, as depicted in Figure 12.

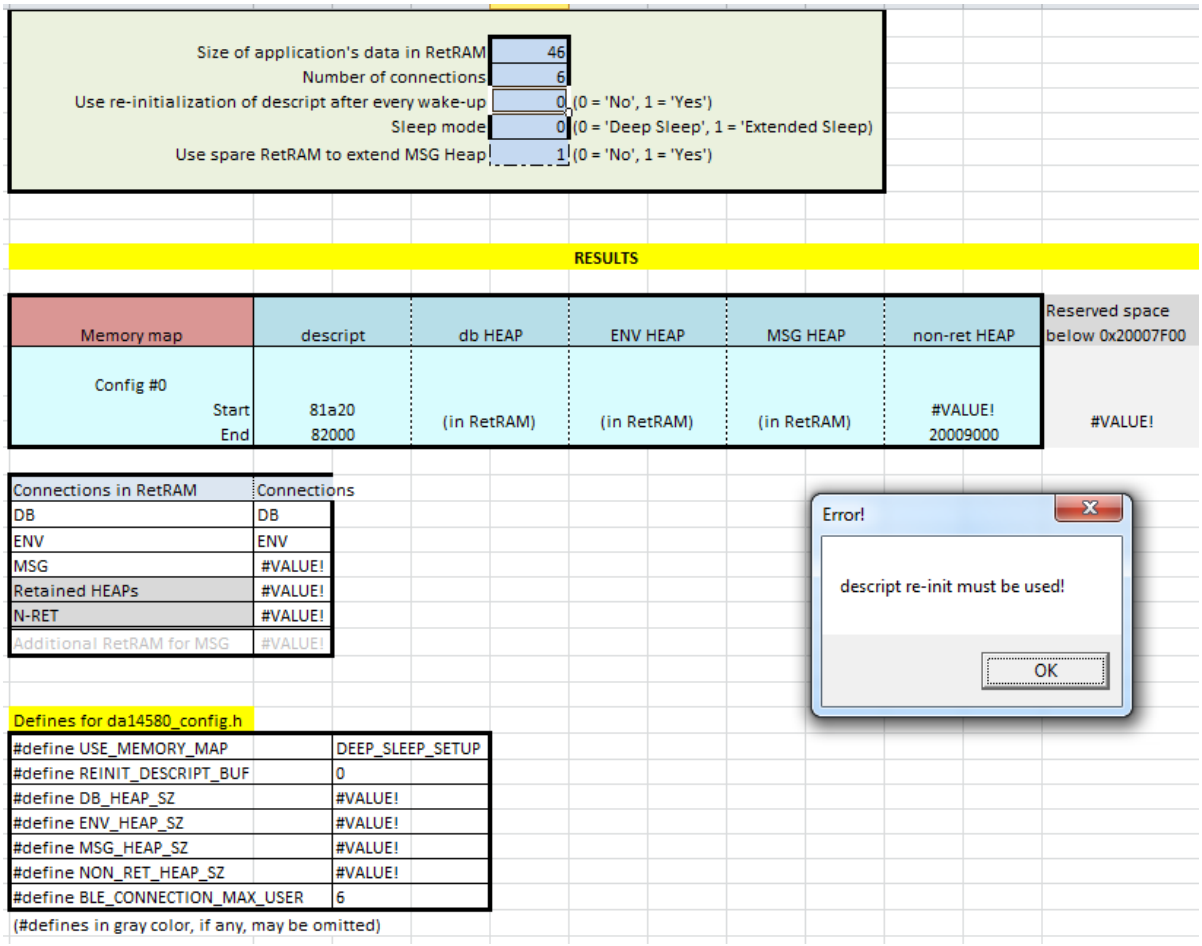


Figure 12: Re-initialization of descriptors is forced

Also, the following “Note” is displayed in the Results’ sheet.

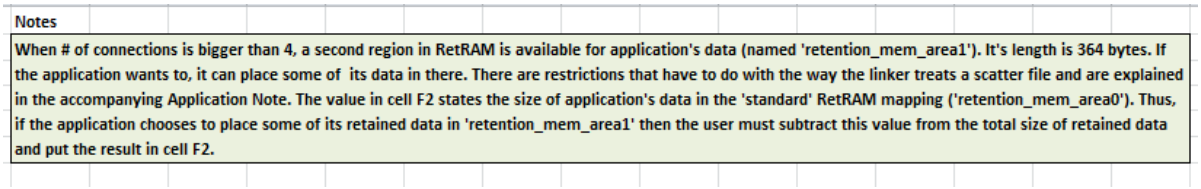


Figure 13: Note informing about “retention_mem_area1”

This note informs the user about the availability of a 364 bytes section (0x800E8 to 0x80253) that can be used to place application’s retained data. The placement of data can be done either by declaring them as belonging to the section “retention_mem_area1” (like the ones that belong to the section “retention_mem_area0”) or by declaring them as belonging too to “retention_mem_area0” but place them explicitly in the respective Execution Region by using in the scatter file statements of the form:

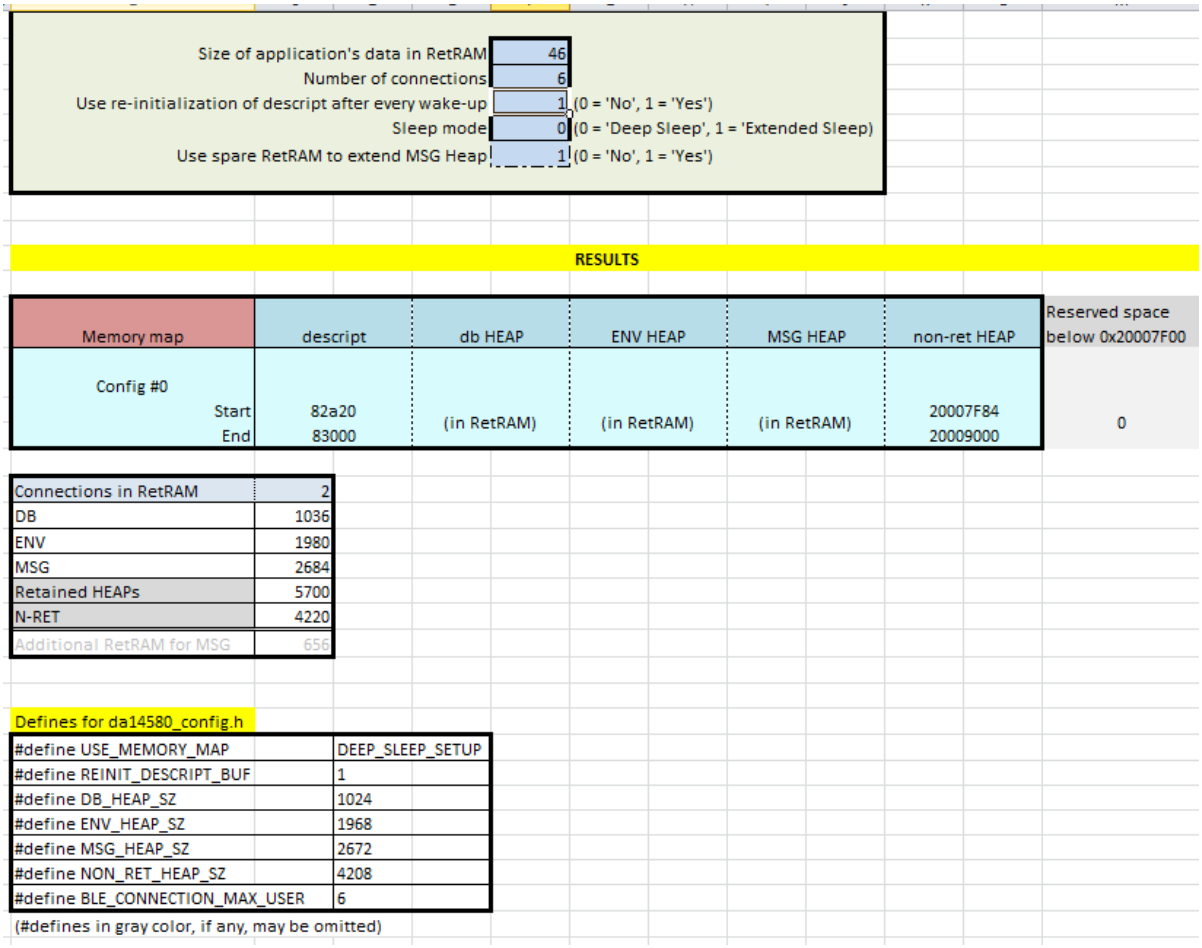
```
app_foo1.o (retention_mem_area0)
app_foo2.o (retention_mem_area0)
...
```

These statements will place any variables declared in the files app_foo1.c and app_foo2.c to belong to the section “retention_mem_area0”, into the respective Execution Region, overriding any statements of the form

```
* (retention_mem_area0)
```

that may exist in the scatter file.

After the user presses “Ok”, the tool calculates the optimum configuration. The results are shown below.



Size of application's data in RetRAM	46	
Number of connections	6	
Use re-initialization of descript after every wake-up	1	(0 = 'No', 1 = 'Yes')
Sleep mode	0	(0 = 'Deep Sleep', 1 = 'Extended Sleep)
Use spare RetRAM to extend MSG Heap	1	(0 = 'No', 1 = 'Yes')

RESULTS

Memory map	descript	db HEAP	ENV HEAP	MSG HEAP	non-ret HEAP	Reserved space below 0x20007F00
Config #0						
Start	82a20	(in RetRAM)	(in RetRAM)	(in RetRAM)	20007F84	0
End	83000				20009000	

Connections in RetRAM	2
DB	1036
ENV	1980
MSG	2684
Retained HEAPs	5700
N-RET	4220
Additional RetRAM for MSG	656

Defines for da14580_config.h

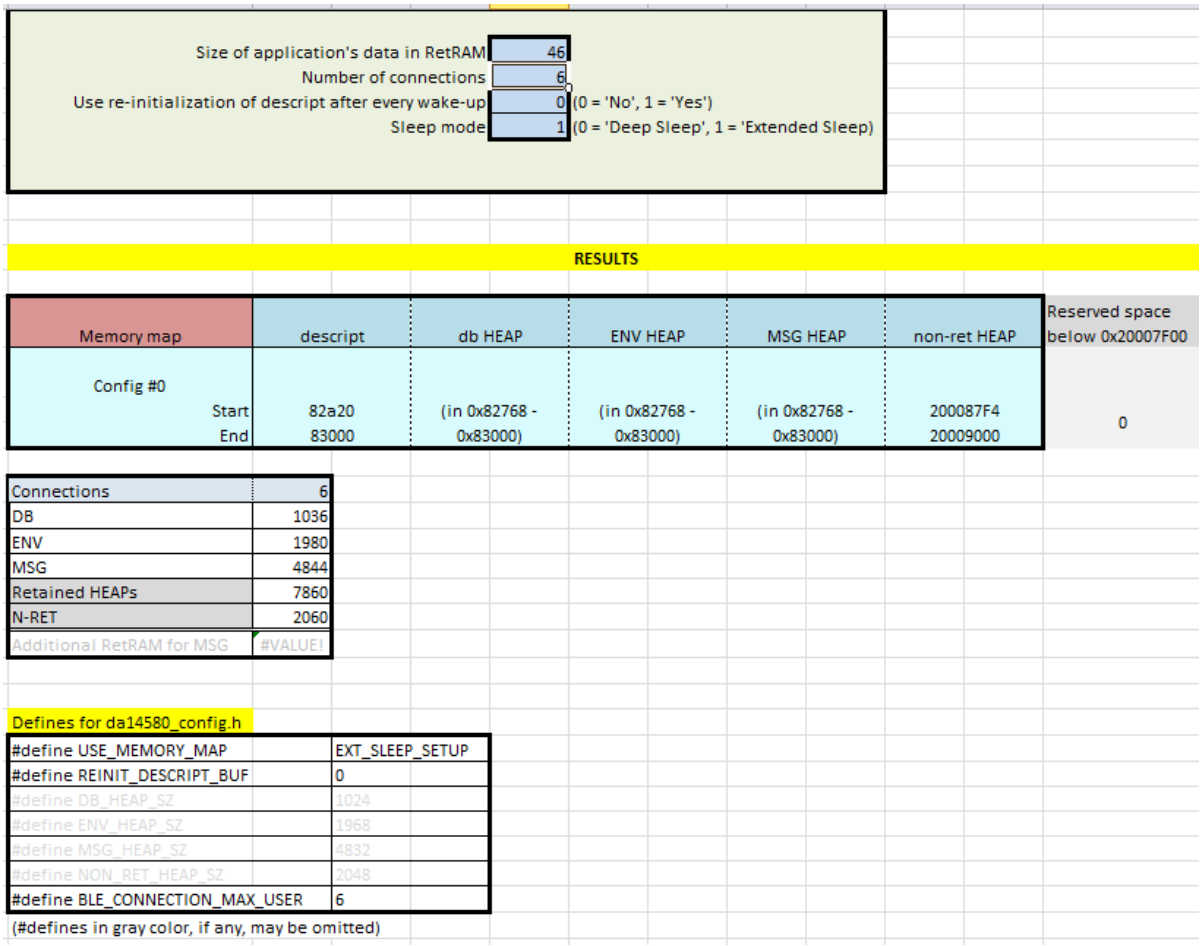
#define USE_MEMORY_MAP	DEEP_SLEEP_SETUP
#define REINIT_DESCRIPT_BUF	1
#define DB_HEAP_SZ	1024
#define ENV_HEAP_SZ	1968
#define MSG_HEAP_SZ	2672
#define NON_RET_HEAP_SZ	4208
#define BLE_CONNECTION_MAX_USER	6

(#defines in gray color, if any, may be omitted)

Figure 14: Master with 6 connections

6.3.4 Extended Sleep

In this example, the user has built an application that is for a master device that must support up to 6 concurrent connections and be able to enter into Extended Sleep. The user uses the Excel tool so as to find the proper settings.



Size of application's data in RetRAM	46	
Number of connections	6	
Use re-initialization of descript after every wake-up	0	(0 = 'No', 1 = 'Yes')
Sleep mode	1	(0 = 'Deep Sleep', 1 = 'Extended Sleep)

RESULTS

Memory map	descript	db HEAP	ENV HEAP	MSG HEAP	non-ret HEAP	Reserved space below 0x20007F00
Config #0						
Start	82a20	(in 0x82768 -	(in 0x82768 -	(in 0x82768 -	200087F4	0
End	83000	0x83000)	0x83000)	0x83000)	20009000	

Connections	6
DB	1036
ENV	1980
MSG	4844
Retained HEAPs	7860
N-RET	2060
Additional RetRAM for MSG	#VALUE!

Defines for da14580_config.h

#define USE_MEMORY_MAP	EXT_SLEEP_SETUP
#define REINIT_DESCRIPT_BUF	0
#define DB_HEAP_SZ	1024
#define ENV_HEAP_SZ	1968
#define MSG_HEAP_SZ	4832
#define NON_RET_HEAP_SZ	2048
#define BLE_CONNECTION_MAX_USER	6

(#defines in gray color, if any, may be omitted)

Figure 15: Master with 6 connections using Extended Sleep

As shown above, there's no need for re-initialization of descriptors since the memory in never shut-off. Furthermore, all heaps have their proper sizes. This is the reason that the corresponding #defines are greyed indicating that it is not required to be included in da14580_config.h.

Note that the note shown in Figure 13 is still shown even in the Extended Sleep case. The user may use this section for data storage but this is not expected to happen since the whole memory is available and is large enough.

7 File structure and scatter file configuration

The common scatter file that is provided by Dialog Semiconductor is named scatterfile_common.sct and is located in dk_apps/scatterfiles/.

This file includes two header files:

- da14580_config.h: among others that are irrelevant to the scatter file, BLE_CONNECTION_MAX_USER, USE_MEMORY_MAP and the other #defines that are calculated by the Excel tool are included in this file.
- da14580_scatter_config.h: this file includes definitions of the sizes of the various heaps and some defines that are required by the scatter file in order to place the various sections at the proper positions in memory. The user should not alter this file.

Furthermore, in the file da14580_stack_config.h the definitions of DEEP_SLEEP_SETUP and EXT_SLEEP_SETUP are included.

In order to configure the application to use the common scatter file the user must make sure that scatterfile_common.sct is selected in Keil (Project Options → Linker → Scatter file). Also, the sections of the file da14580_config.h that are marked in red in Figure 16, must have the values that

are calculated by the Excel tool. The only difference from what shown in the figure is when Extended Sleep is used, where the definitions of heaps' sizes can be omitted.

```
/*Maximum user connections*/
#define BLE_CONNECTION_MAX_USER 1

/*Build for OTP or JTAG*/
#define DEVELOPMENT__NO_OTP     0                   //0: code at OTP, 1: code via JTAG

#ifdef CFG_PRINTF
#define HAS_PRINTF              1
#else
#define HAS_PRINTF              0
#endif

/*Low power clock selection*/
#define CFG_LP_CLK              0x00                //0x00: XTAL32, 0xAA: RCX20, 0xFF: Select from OTP Header

/*
 * Scaterfile: Memory maps
 */
#define USE_MEMORY_MAP          DEEP_SLEEP_SETUP
#define REINIT_DESCRIPT_BUF     0                   //0: keep in RetRAM, 1: re-init is required (set to 0 when Extended Sleep is used)
#define DB_HEAP_SZ              1024
#define ENV_HEAP_SZ             328
#define MSG_HEAP_SZ             1312
#define NON_RET_HEAP_SZ         1024
```

Figure 16: Scatter file configuration via da14580_config.h

The structure of the scatterfile_common.sct will not be analysed in this document. The reader is encouraged to review it.

The memory layouts the scatterfile_common.sct creates for Deep and Extended Sleep are presented in the following sub-sections.

7.1 Deep Sleep memory layout

The memory layout in the range 0x20000440 to 0x200097FF for various configurations is shown below.

Figure 17a: Non-retained heap has size less than 4096 bytes

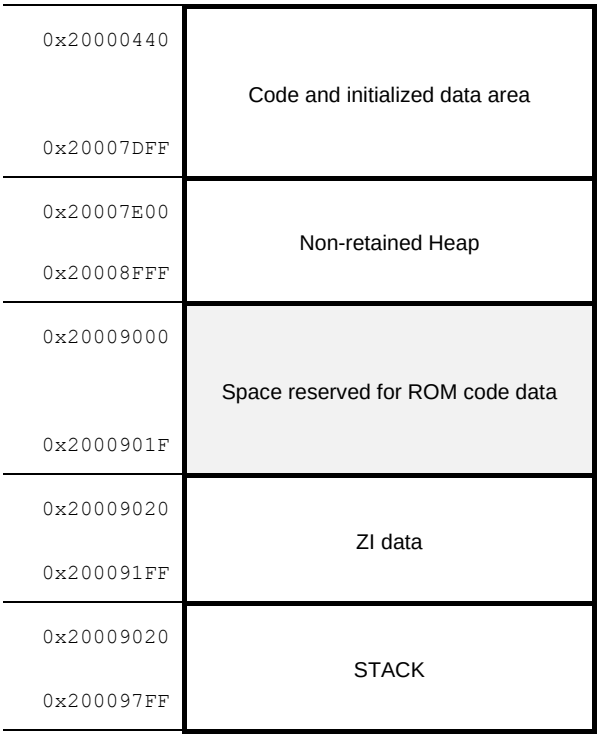


Figure 17b: Non-retained heap has size equal to 4608 bytes

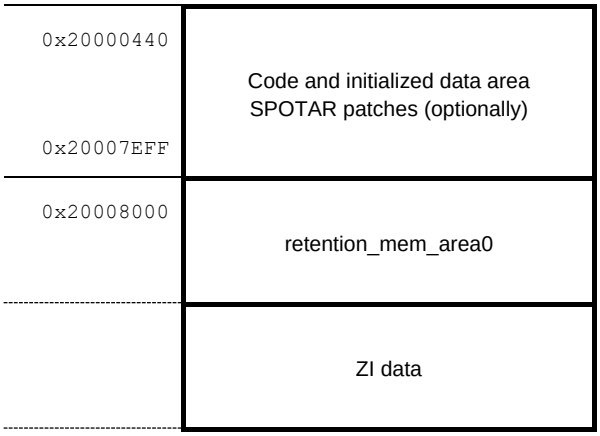
In the figures above, two layouts are shown. In the first one, the non-retained heap's size is small. In this case, the area 0x20000000 to 0x20007FFF that the OTP is mapped to, is left intact. From 0x20008000 and until the beginning of the non-retained heap, Zero-Init data are placed. Then the heap follows. Another section for ZI data is from 0x20009020 to 0x200091FF. Above this, the STACK is located.

In the second layout, the non-retained heap's size is large. It does not fit in 0x20008000 to 0x20008FFF, so the code and initialized data section is reduced from 0x20000440 to 0x20007EFF to 0x20000440 to 0x20007DFF. Of course, since the whole section is used for non-retained heap, there's no place left there for any ZI data. This rest of this layout is the same with the other one.

Regarding the layout in the range 0x80000 to 0x82FFF the possible cases where discussed in Section 4.1.

7.2 Extended Sleep memory layout

The memory layout in the range 0x20000440 to 0x200097FF is shown below.



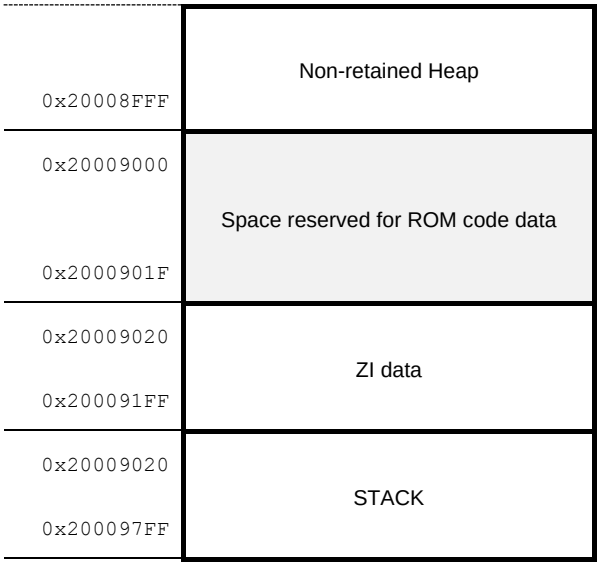
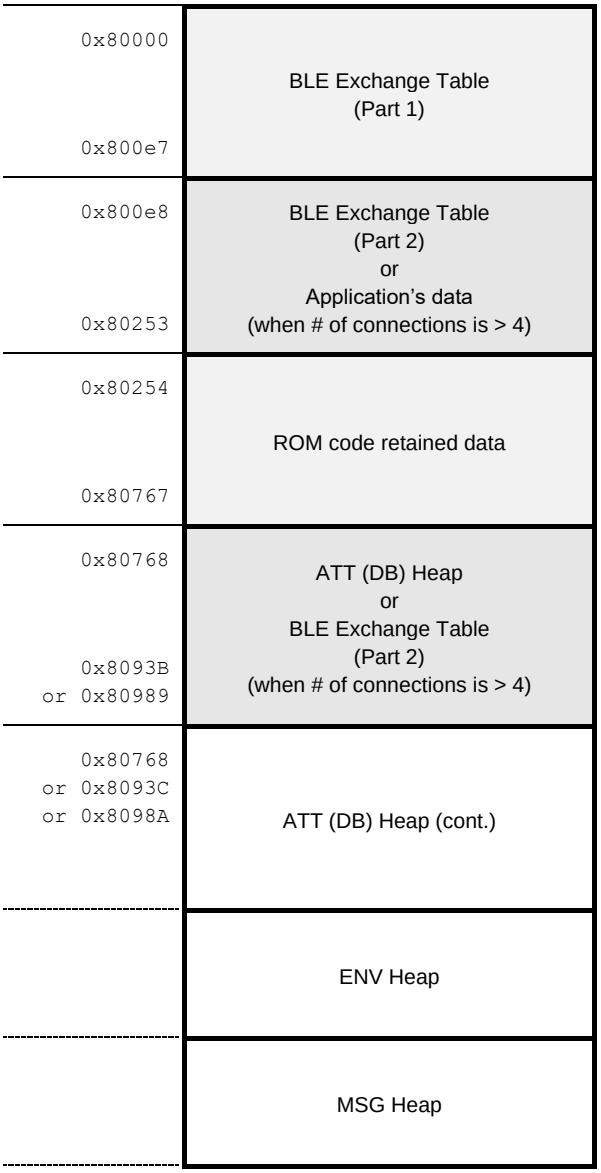


Figure 18: Extended Sleep layout in 0x20000440 to 0x200097FF

The memory layout in the range 0x80000 to 0x82FFF is shown below.



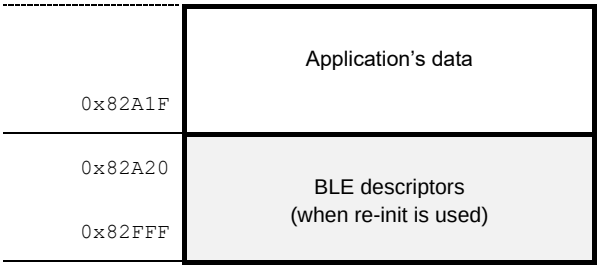


Figure 19: Extended Sleep layout in 0x80000 to 0x82FFF

8 Building the application – Keil's status report

The status report from Keil when an application that supports less than 5 connections is built is shown below.

```
Build Output
Build target 'Full_emb_cortex_M0'
linking...
Program Size: Code=21172 RO-data=2756 RW-data=44 ZI-data=7668
FromELF: creating hex file...
".\out\full_emb_sysram.axf" - 0 Error(s), 0 Warning(s).
```

Figure 20: Successful built of an application with 1 to 4 connections

If an application that supports 5 or 6 connections is built then Keil will most likely output the following status.

```
Build Output
compiling findl_task.c...
linking...
..\..\..\scatterfiles\scatterfile_common.sct(256): warning: L6314W: No section matches pattern *(retention_mem_area1).
Program Size: Code=11100 RO-data=1128 RW-data=0 ZI-data=13772
Finished: 0 information, 1 warning and 0 error messages.
FromELF: creating hex file...
".\out\full_emb_sysram.axf" - 0 Error(s), 4 Warning(s).
```

Figure 21: Successful built of an application with 5 or 6 connections

As shown in Figure 21, a warning is issued by the Linker because there's a section declared in scatterfile_common.sct which is named "retention_mem_area1" that has no data. This happens because the used did not place data in this section using any of the procedures described in 6.3.3. This warning may be ignored.

Any other outcome indicates that something is wrong. Two possible cases are shown below.

```
..\..\..\scatterfiles\scatterfile_common.sct(256): warning: L6314W: No section matches pattern *(retention_mem_area1).
..\..\..\scatterfiles\scatterfile_common.sct: Error: L6220E: Execution region ZI_RET00 size (8124 bytes) exceeds limit (6296 bytes).
    Region contains 9 bytes of padding and 0 bytes of veneers (total 9 bytes of linker generated content).
Not enough information to produce a SYMDEFs file.
Finished: 1 information, 1 warning and 1 error messages.
".\out\full_emb_sysram.axf" - 1 Error(s), 4 Warning(s).
Target not created
```

Figure 22: Build fails when data do not fit in Retention RAM

```
.\out\full_emb_sysram.axf: Error: L6985E: Unable to automatically place AT section arch_main.o(.ARM.__AT_0x00082A20)
    with required base address 0x00082a20. Please manually place in the scatter file using the --no_autoat option.
Not enough information to produce a SYMDEFs file.
Not enough information to list image symbols.
Finished: 2 information, 0 warning and 1 error messages.
".\out\full_emb_sysram.axf" - 1 Error(s), 3 Warning(s).
Target not created
```

Figure 23: Build fails when too much data in Retention RAM prevents descriptors to be placed

Possible causes of this problem are:

- The size of the section that contains the application's retained data ("retention_mem_area0") is larger than the one declared in the Excel tool and used for the calculations.

- The #defines in da14580_config.h have not the values that the Excel tool calculated.

The following errors may also occur:

- *Memory map error! Device configured for Deep Sleep but the chosen Memory Map is not correct.*

  The application is configured to use Deep Sleep but EXT_SLEEP_SETUP was declared in da14580_config.h

- *Config error: the number of supported connections ranges from 1 to 6.*

  The user set BLE_CONNECTION_MAX_USER to a value outside the range 1 to 6.

Finally, the following warnings may be issued:

- *Memory map: Device configured for Extended Sleep but the chosen Memory Map applies for Deep Sleep. Please check if Deep Sleep can be used!*

  The application is configured to use Extended Sleep but DEEP_SLEEP_SETUP was declared in da14580_config.h. Since the code can successfully be built using a memory layout that supports Deep Sleep, this warning instructs the user to check whether Deep Sleep could be used in his application. Of course, there might be other restrictions, that are application dependant, that could exclude the usage of Deep Sleep.

- *Memory map: re-init of descript is not necessary in Extended Sleep mode.*

  REINIT_DESCRIPT_BUF was set to 1 is da14580_config.h although EXT_SLEEP_SETUP was defined for USE_MEMORY_MAP. The re-initialization of descriptors is not required when Extended Sleep is used and will not be done by the system anyway. This warning is issued because failing to use the proper value for this parameter might indicate that also the rest of the #defines in da14580_config.h that are set by the Excel tool have not been set correctly. The user should re-check them for safety.

9 Revision history

Revision	Date	Description
1.0	27-Mar-2014	Initial version for DA14580
1.1	17-Jul-2014	Correct range of the “Code and initialized data” section.
1.2	25-Jan-2022	Updated logo, disclaimer, copyright.

Status definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.