How to Understood .LDF file

Henry Long

Analog Devices Inc.

2017.8.15

A simple Ldf example (1)

1. Here's a basic example of how to map two arrays into two separate memory sections in the LDF file, eg.

   We'd defined two data memory sections, which could be in SRAM or SDRAM depends on the address you defined.

```
MEMORY{
…
MEM_L1_DATA_B { TYPE(RAM) START(0xFF900000) END(0xFF907FFF) WIDTH(16) }
MEM_L1_DATA_A { TYPE(RAM) START(0xFF800000) END(0xFF807FFF) WIDTH(16) }
…
}
```

2. We will map data declared for certain sections to reside in these physical banks:

```
PROCESSOR p0
{           …
        SECTIONS
        {
                L1_Data_A // note that this name is arbitrary and isn't used in the mapping process
                {       INPUT_SECTIONS($OBJECTS(L1_data_a) $LIBRARIES(L1_data_a))
                } > MEM_L1_DATA_A
                L1_Data_B // note that this name is arbitrary and isn't used in the mapping process
                {       INPUT_SECTIONS($OBJECTS(L1_data_b) $LIBRARIES(L1_data_b))
                } > MEM_L1_DATA_B
        }…
 }
```

A simple Ldf example (2)

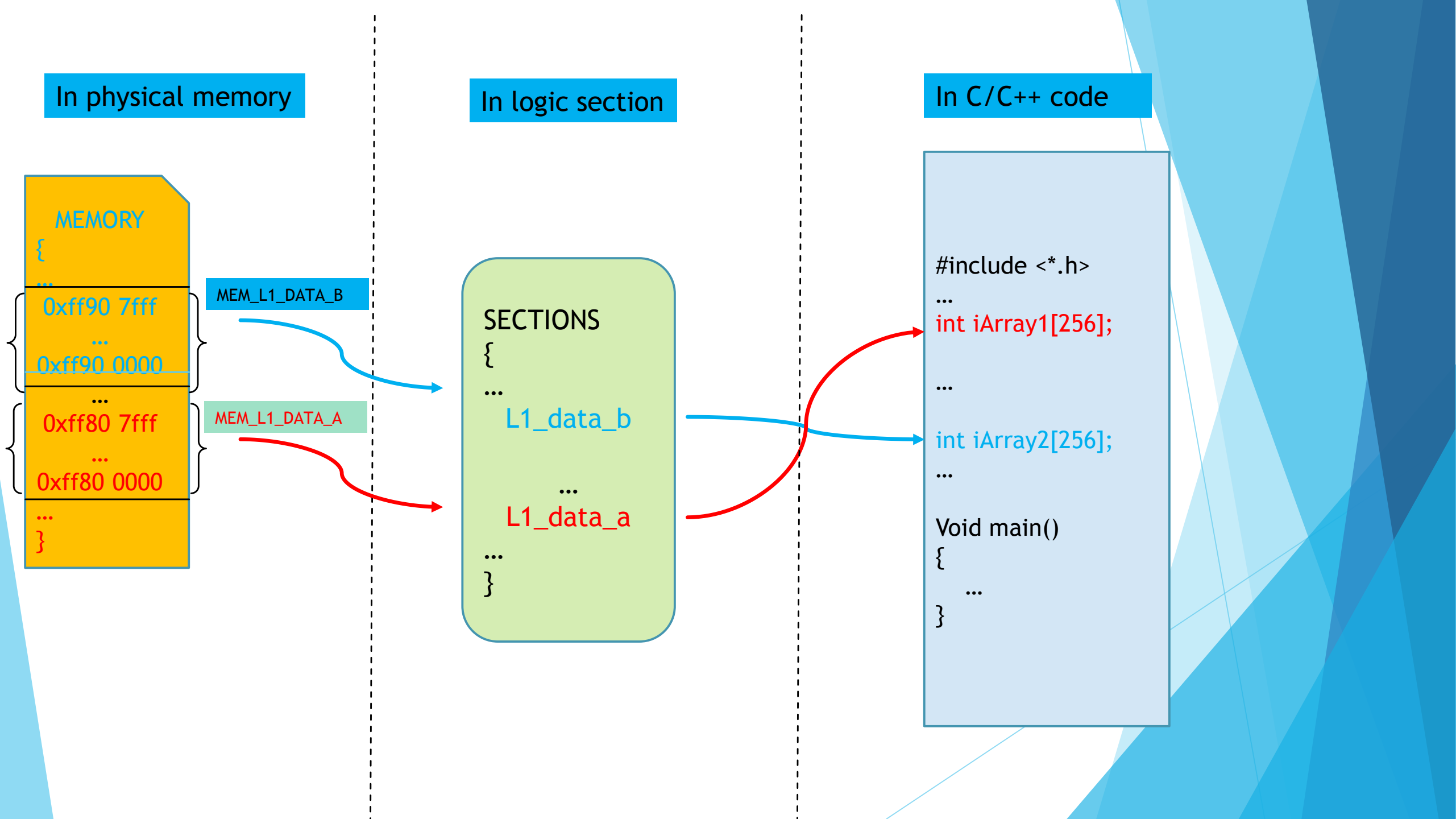
3. **In the C file, you can use the section directive to map variables into your memory sections like so:**

short section("**L1_data_a**") iArray1[256];

short section("**L1_data_b**") iArray2[256];

Note that if you don't specify a section for a variable, it will default to the "data" section. If you open a full LDF file in VDSP, you'll see that data gets mapped into L1_DATA_A default.

In the similar way, you can define your own section in program memory, and allocate a specific function to that physical memory.


In physical memory
In logic section
In C/C++ code
MEMORY
{
...
0xff90 7fff
...
0xff90 0000
...
0xff80 7fff
...
0xff80 0000
...
}
MEM_L1_DATA_B
MEM_L1_DATA_A
SECTIONS
{
...
L1_data_b
...
L1_data_a
...
}
#include <*.h>
...
int iArray1[256];
...
int iArray2[256];
...
Void main()
{
...
}