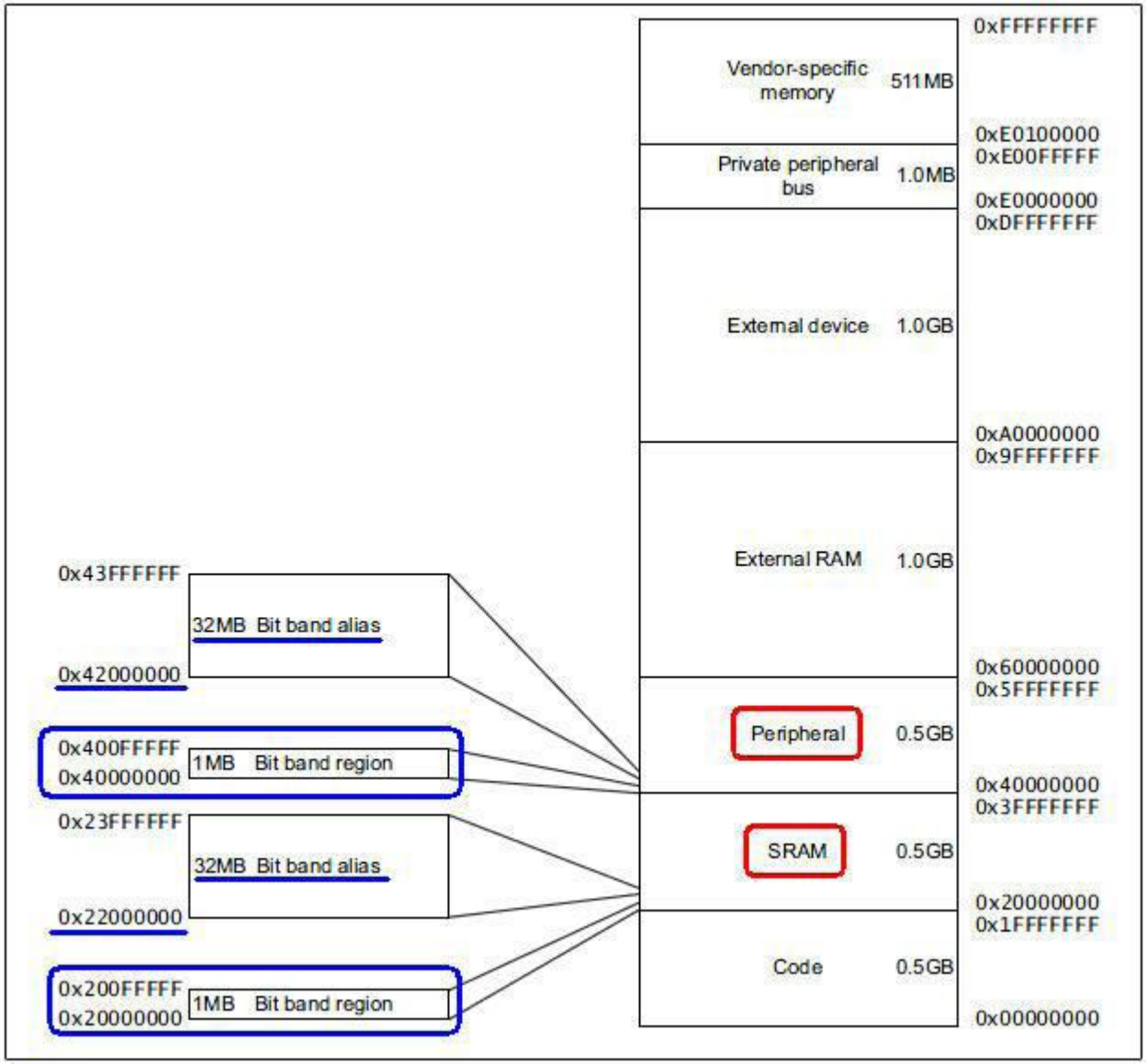
Bit Band

2010.02.08

Bit Banding

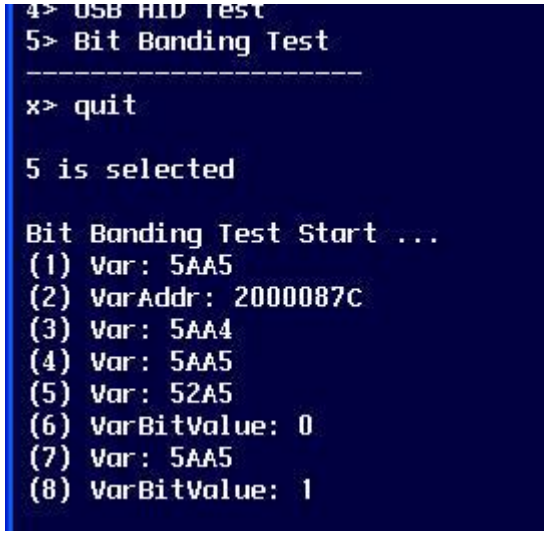

0xFFFFFFFF
Vendor-specific memory
511MB
0xE0100000
0xE00FFFFF
Private peripheral bus
1.0MB
0xE0000000
0xDFFFFFFF
External device
1.0GB
0xA0000000
0x9FFFFFFF
External RAM
1.0GB
0x60000000
0x5FFFFFFF
Peripheral
0.5GB
0x40000000
0x3FFFFFFF
SRAM
0.5GB
0x20000000
0x1FFFFFFF
Code
0.5GB
0x00000000
0x43FFFFFF
32MB Bit band alias
0x42000000
0x400FFFFF
1MB Bit band region
0x40000000
0x23FFFFFF
32MB Bit band alias
0x22000000
0x200FFFFF
1MB Bit band region
0x20000000

main.c macro

```
#define RAM_BASE       0x20000000
#define RAM_BB_BASE    0x22000000
#define Var_ResetBit_BB(VarAddr, BitNumber)   \
        (*(__IO uint32_t *) (RAM_BB_BASE | \
        ((VarAddr - RAM_BASE) << 5) | ((BitNumber) << 2)) = 0)
#define Var_SetBit_BB(VarAddr, BitNumber)     \
        (*(__IO uint32_t *) (RAM_BB_BASE | \
        ((VarAddr - RAM_BASE) << 5) | ((BitNumber) << 2)) = 1)
#define Var_GetBit_BB(VarAddr, BitNumber)     \
        (*(__IO uint32_t *) (RAM_BB_BASE | \
        ((VarAddr - RAM_BASE) << 5) | ((BitNumber) << 2)))
```

- bit_word_addr = bit_band_base + (byte_offset x 32) + (bit_number × 4)
- Var_ResetBit_BB(VarAddr, BitNumber)는 SRAM 영역의 주소값과 비트 번호를 주게 됩니다.
- bit_band_base + (byte_offset x 32) + (bit_number × 4)의 형태로 덧셈을 하는 것과 위 매크로에서 OR 연산을 하는 것은 비트 영역이 서로 겹치는 것이 없기 때문에 동일한 결과를 얻게 됩니다.

실행 결과




```
/* Modify Var variable bit 11 */
Var_ResetBit_BB(VarAddr, 11);
                    /* Var = 0x000052A5 */
printf("(5) Var: %0X\n", Var);
(5) Var: 52A5
```

- 11번 비트를 변경
  - 0x5AA5를 binary로 표현하면 101101010100101
  - 11번 비트는 101**1**01010100101에서 굵게 표시된 앞에서 4번째 숫자 1
  - 이것을 Var_ResetBit_BB을 통해서 0으로 만들고 있는 것이고, 101**0**01010100101이 된다.