MIAT_STM32
內部與外部SRAM存取控制實驗

浯陽科技有限公司
WU-YANG
Technology Co., Ltd.

Declared Version

Training Only

Declare	
Document Number	
Document Version	***1.00***
Release Date	
Document Title	MIAT_STM32 內部與外部SRAM存取控制實驗
Exercise Time	■
Platform	■ ***MIAT_STM32.V2*** ■ ***MIAT IOB.V1***
Peripheral	■
Author	■ ***WU-YANG Technology Co., Ltd.***

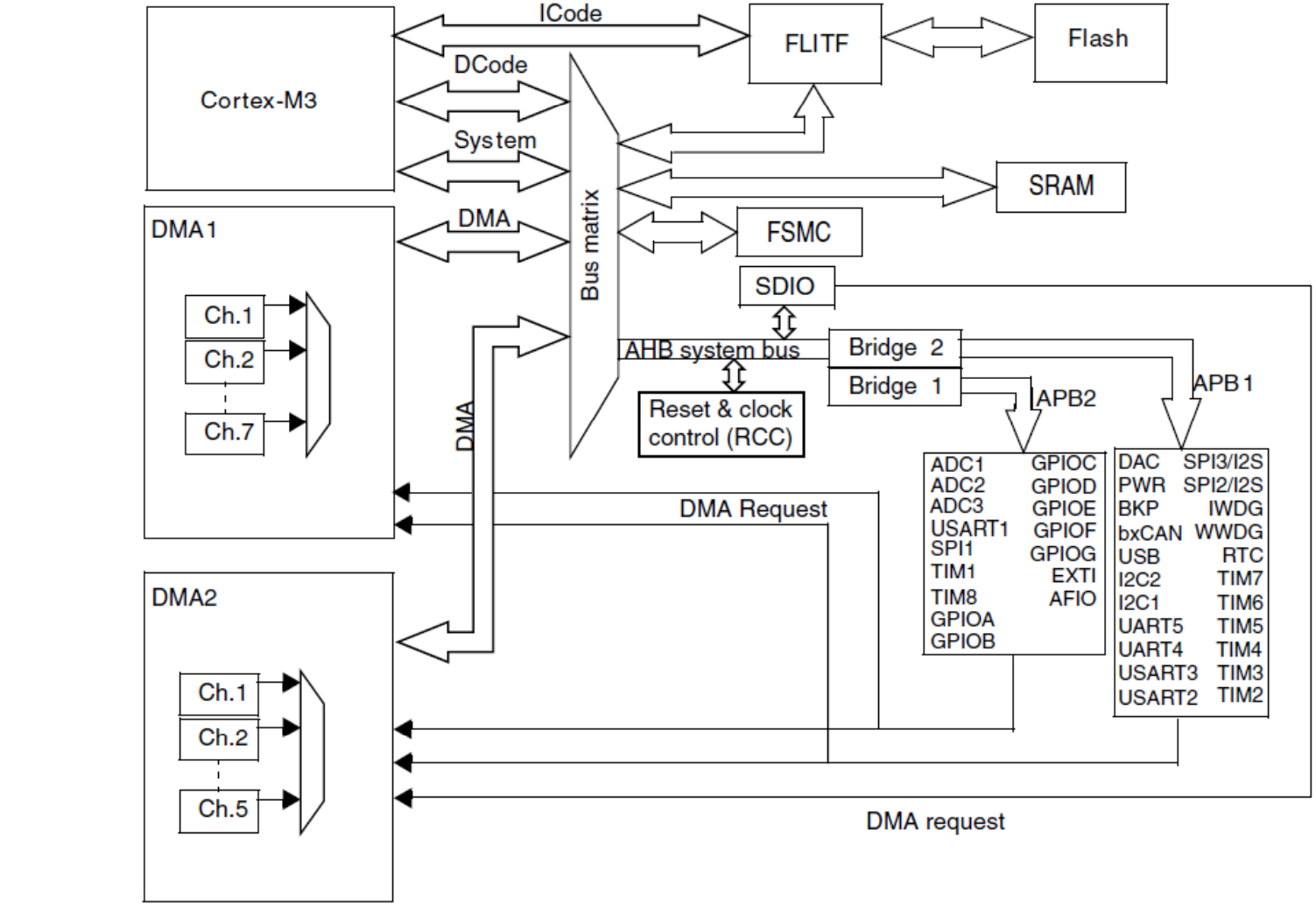
實驗目的(一)

- 使用MIAT_STM32實驗板透過Flexible static memory controller (FSMC)控制內部與外部SRAM進行存取控制實驗，並利用LED確認存取是否正常。

實驗原理

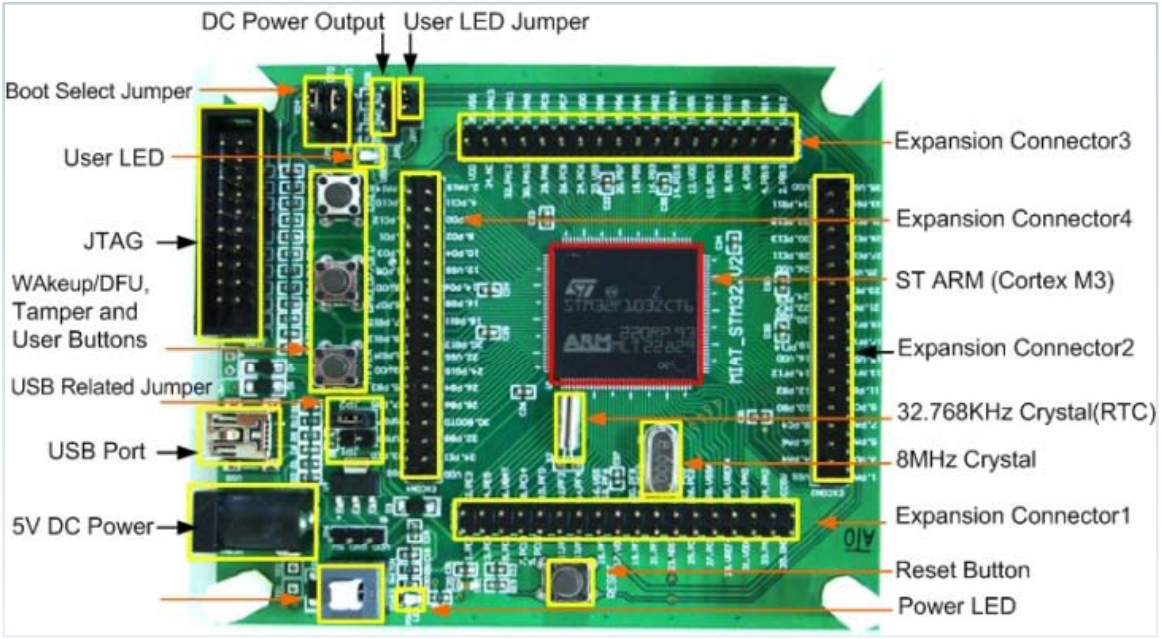
- System architecture
- Embedded SRAM
  - Features
  - Memory map
  - RVMDK環境設定
- External SRAM
  - IS61LV25616AL
  - FSMC (flexible static memory controller)
- Development Flow
- ARM Configure

System architecture

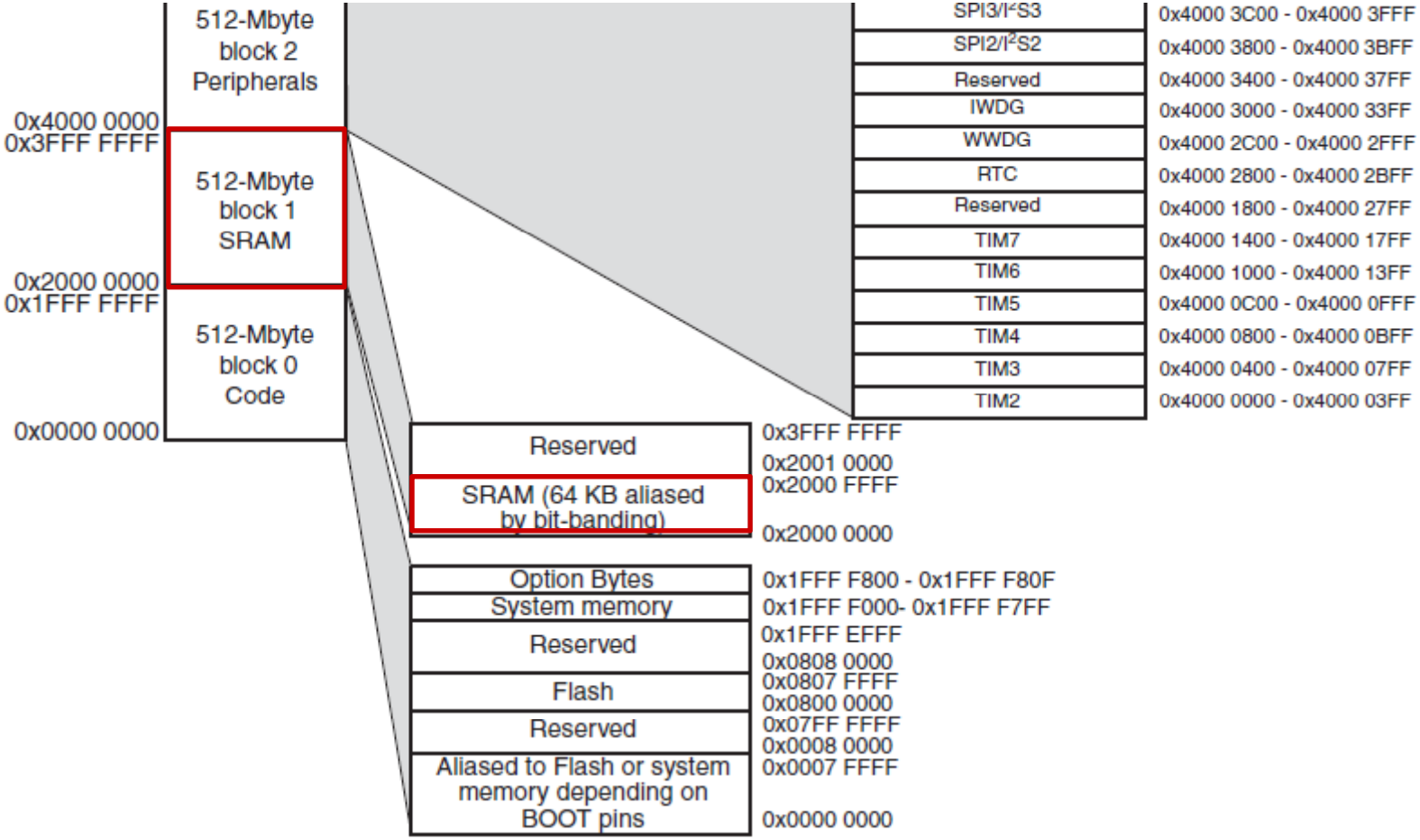


Embedded SRAM

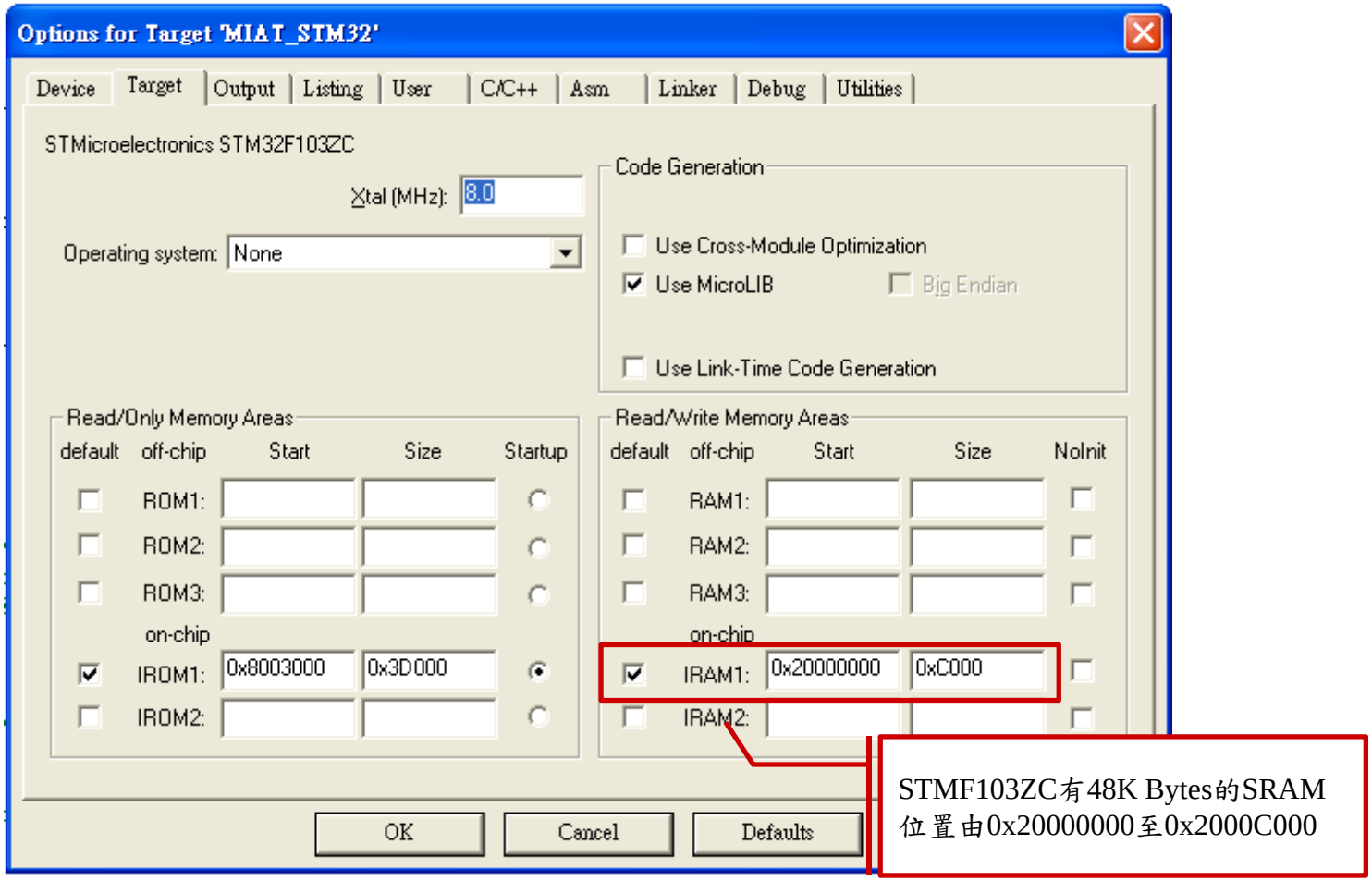
- Features
  - 48Kbytes of embedded SRAM
  - accessed (read/write) at CPU clock speed with 0 wait states



Embedded SRAM Memory map

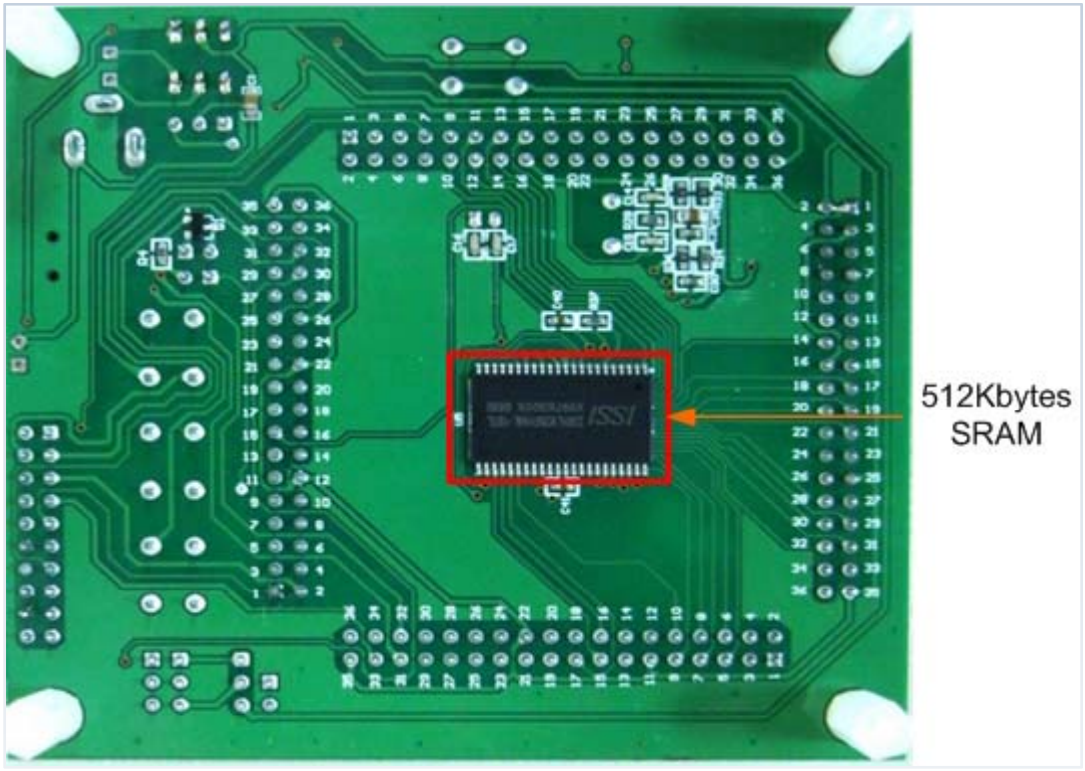


RVMDK環境設定

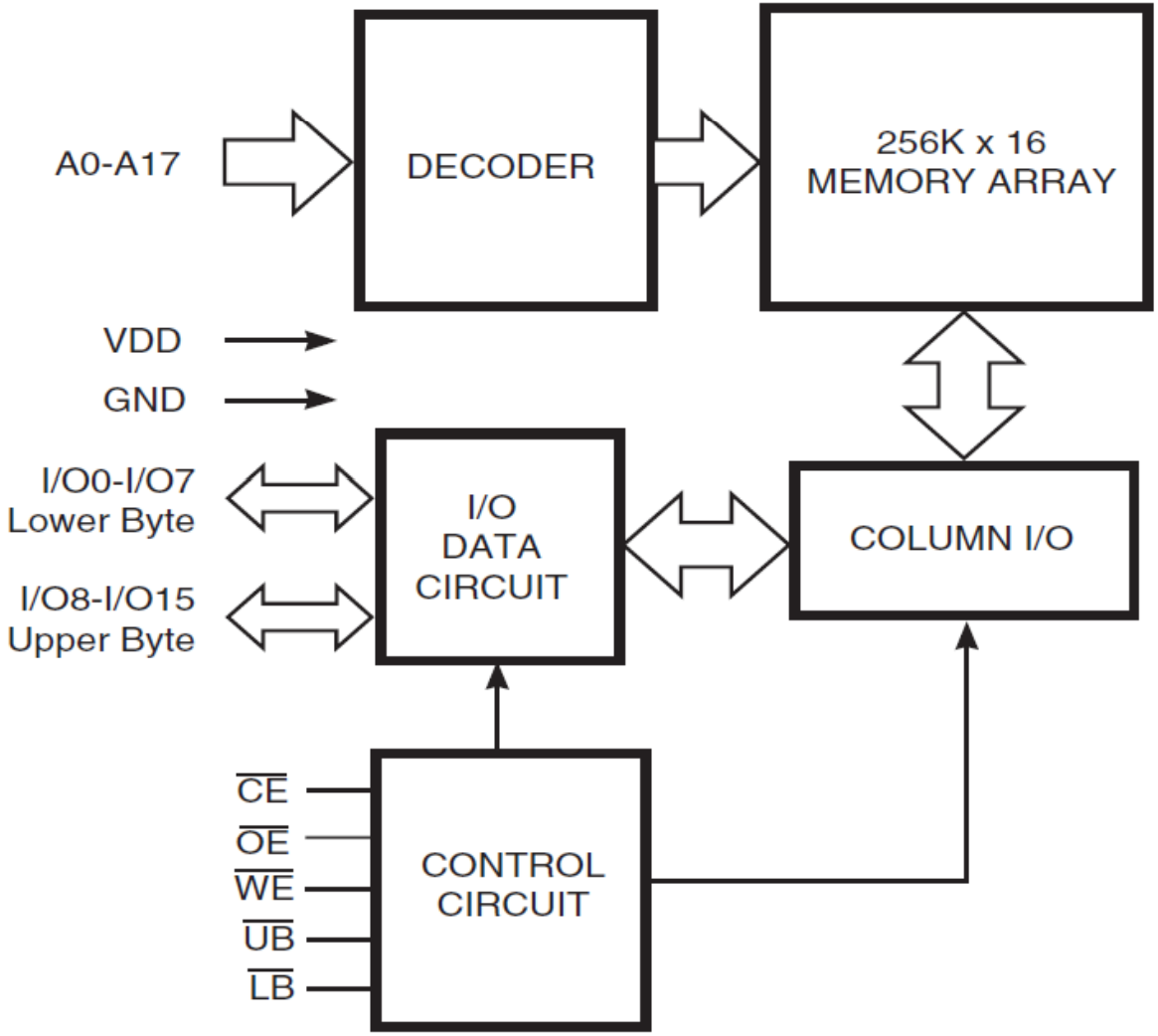

Options for Target 'MIAT_STM32'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
STMicroelectronics STM32F103ZC
Xtal (MHz): 8.0
Operating system: None
Code Generation
Use Cross-Module Optimization
Use MicroLIB
Big Endian
Use Link-Time Code Generation
Read/Only Memory Areas
default off-chip Start Size Startup
ROM1:
ROM2:
ROM3:
on-chip
IROM1: 0x8003000 0x3D000
IROM2:
Read/Write Memory Areas
default off-chip Start Size NoInit
RAM1:
RAM2:
RAM3:
on-chip
IRAM1: 0x20000000 0xC000
IRAM2:
OK
Cancel
Defaults
STMF103ZC有48K Bytes的SRAM
位置由0x20000000至0x2000C000

External SRAM (IS61LV25616AL)

- Features
  - High-speed access time 10 ns
  - 256K x 16



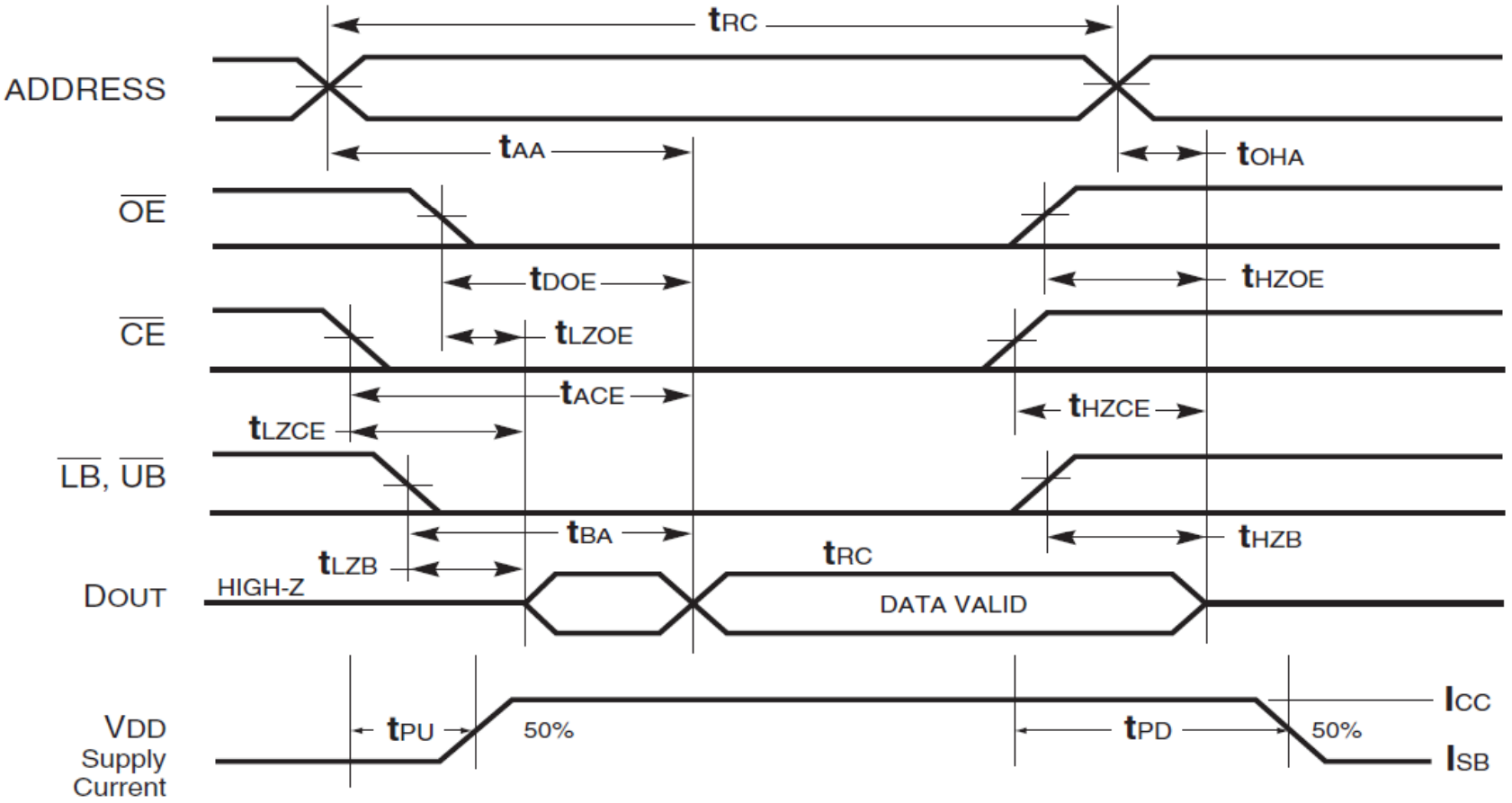
FUNCTIONAL BLOCK DIAGRAM



PIN DESCRIPTIONS

A0-A17	Address Inputs
I/O0-I/O15	Data Inputs/Outputs
$\overline{CE}$	Chip Enable Input
$\overline{OE}$	Output Enable Input
$\overline{WE}$	Write Enable Input
$\overline{LB}$	Lower-byte Control (I/O0-I/O7)
$\overline{UB}$	Upper-byte Control (I/O8-I/O15)
NC	No Connection
V_{DD}	Power
GND	Ground

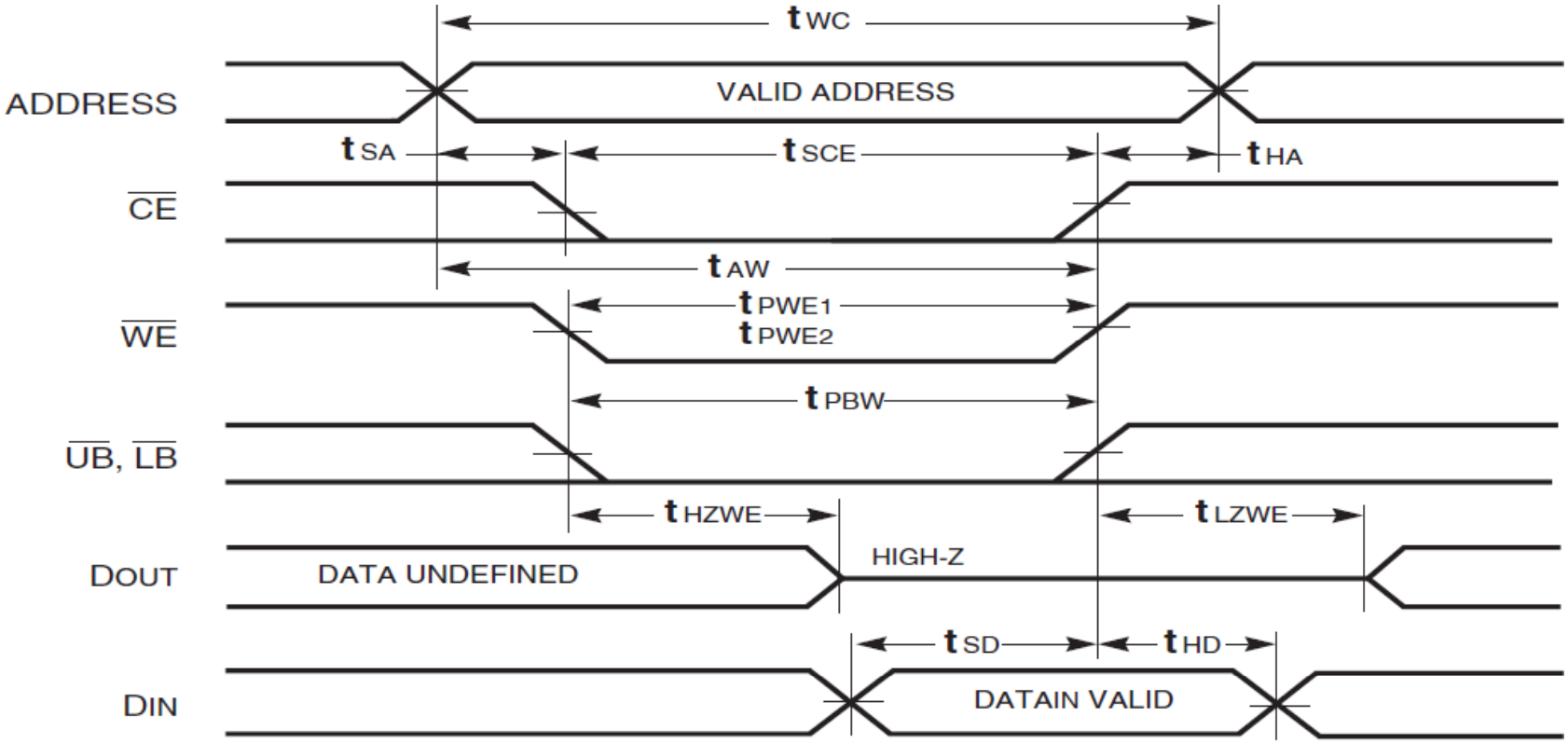
READ CYCLE


tRC
ADDRESS
tAA
tOHA
OE
tDOE
tHZOE
CE
tLZOE
tACE
tHZCE
tLZCE
LB, UB
tBA
tHZB
tLZB
tRC
DOUT
HIGH-Z
DATA VALID
VDD
Supply
Current
tPU
50%
tPD
50%
ICC
ISB

READ CYCLE SWITCHING CHARACTERISTICS

Symbol	Parameter	-10 Min.	-10 Max.
t_{RC}	Read Cycle Time	10	—
t_{AA}	Address Access Time	—	10
t_{OHA}	Output Hold Time	2	—
t_{ACE}	$\overline{CE}$ Access Time	—	10
t_{DOE}	$\overline{OE}$ Access Time	—	4
t_{HZOE}(2)	$\overline{OE}$ to High-Z Output	—	4
t_{LZOE}(2)	$\overline{OE}$ to Low-Z Output	0	—
t_{HZCE}(2	$\overline{CE}$ to High-Z Output	0	4
t_{LZCE}(2)	$\overline{CE}$ to Low-Z Output	3	—
t_{BA}	$\overline{LB}$, $\overline{UB}$ Access Time	—	4
t_{HZB}(2)	$\overline{LB}$, $\overline{UB}$ to High-Z Output	0	3
t_{LZB}(2)	$\overline{LB}$, $\overline{UB}$ to Low-Z Output	0	—
t_{PU}	Power Up Time	0	—
t_{PD}	Power Down Time	—	10

WRITE CYCLE


tWC
ADDRESS
VALID ADDRESS
tSA
tSCE
tHA
CE
tAW
tPWE1
WE
tPWE2
tPBW
UB, LB
tHZWE
tLZWE
DOUT
DATA UNDEFINED
HIGH-Z
tSD
tHD
DIN
DATAIN VALID

WRITE CYCLE SWITCHING CHARACTERISTICS

Symbol	Parameter	-10 Min.	-10 Max.
t_{WC}	Write Cycle Time	10	—
t_{SCE}	$\overline{CE}$ to Write End	8	—
t_{AW}	Address Setup Time to Write End	8	—
t_{HA}	Address Hold from Write End	0	—
t_{SA}	Address Setup Time	0	—
t_{PWB}	$\overline{LB}$, $\overline{UB}$ Valid to End of Write	8	—
t_{PWE1}	$\overline{WE}$ Pulse Width	8	—
t_{PWE2}	$\overline{WE}$ Pulse Width ($\overline{OE}$ = LOW)	10	—
t_{SD}	Data Setup to Write End	6	—
t_{HD}	Data Hold from Write End	0	—
t_{HZWE}[2]	$\overline{WE}$ LOW to High-Z Output	—	5
t_{LZWE}[2]	$\overline{WE}$ HIGH to Low-Z Output	2	—

FSMC

- Features
  - Interfaces with static memory-mapped devices including:
    - Static random access memory (SRAM)
    - Read-only memory (ROM)
    - NOR Flash memory
    - PSRAM (4 memory banks)8- or 16-bit wide databus
  - 8- or 16-bit wide databus
  - Independent chip select control for each memory bank
  - Independent configuration for each memory bank

FSMC

- Programmable timings to support a wide range of devices, in particular:
  - Programmable wait states (up to 15)
  - Programmable bus turnaround cycles (up to 15)
  - Programmable output enable and write enable delays (up to 15)
  - Independent read and write timings and protocol, so as to support the widest variety of memories and timings

NOR/PSRAM address mapping

- NOR/PSRAM bank selection

HADDR[27:26](1)	Selected bank
00	Bank 1 NOR/PSRAM 1
01	Bank 1 NOR/PSRAM 2
10	Bank 1 NOR/PSRAM 3
11	Bank 1 NOR/PSRAM 4

- External memory address

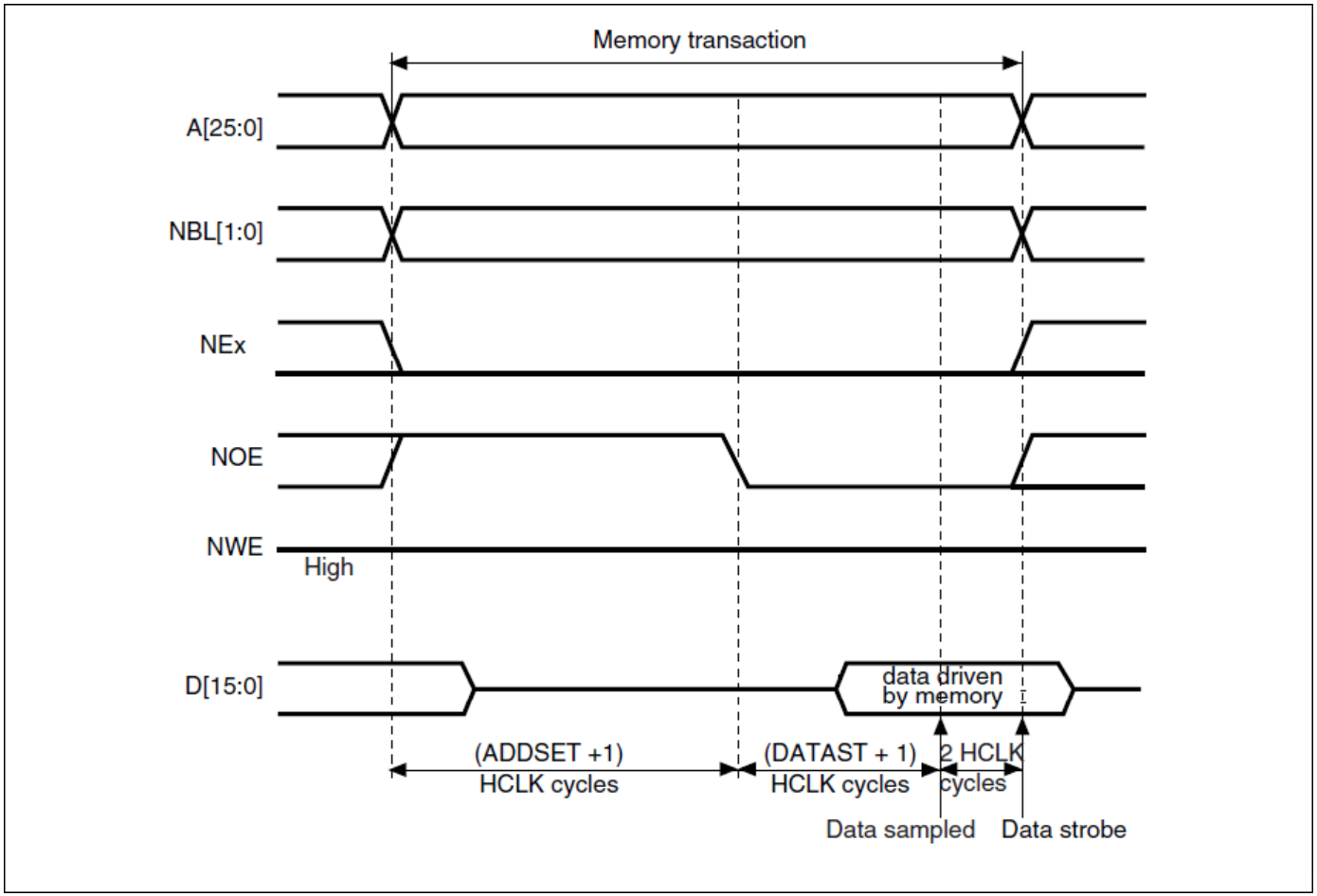
Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit

External memory interface signals

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lowed byte enable (memory signal name: NLB)

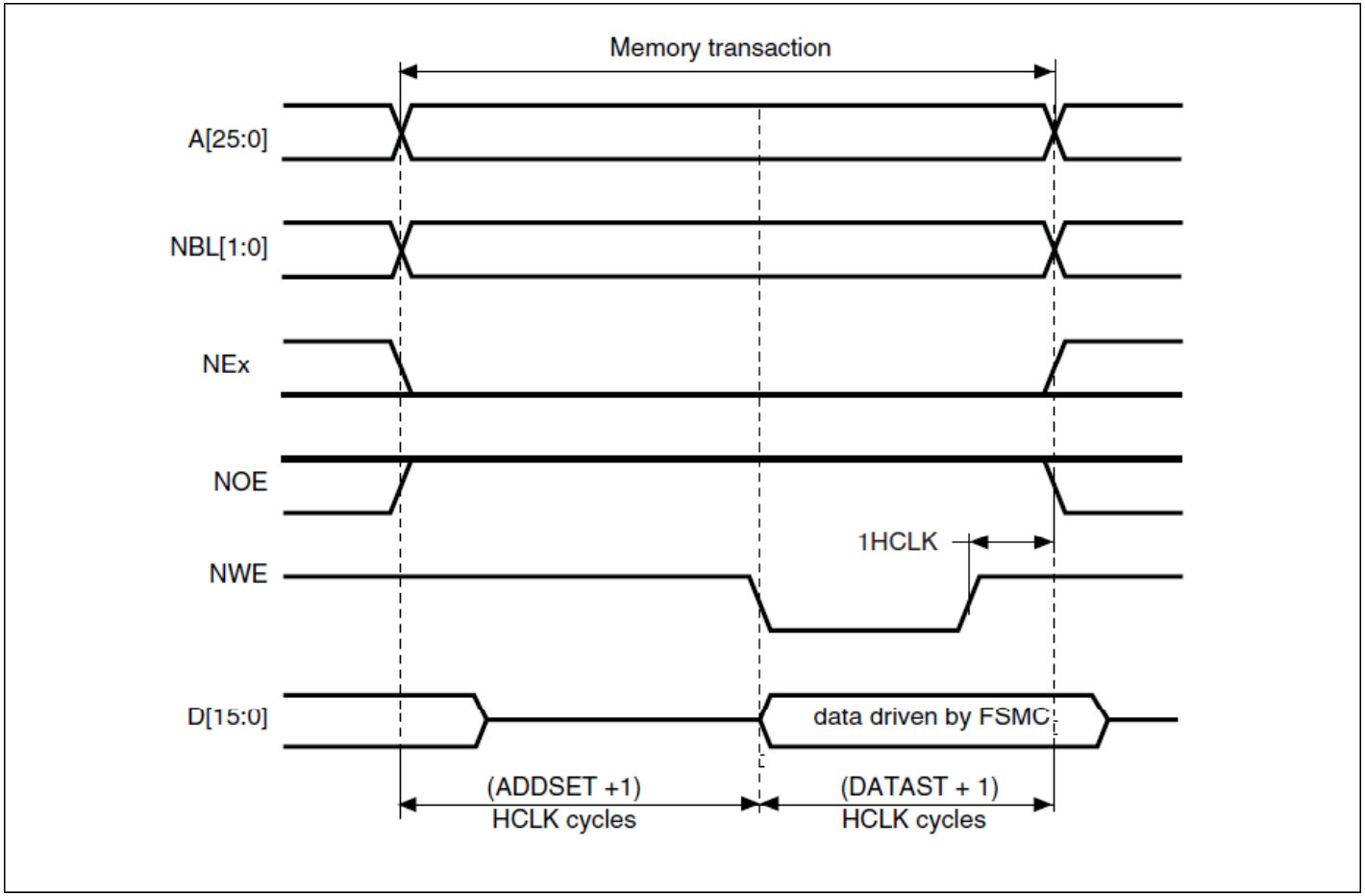
NOR Flash/PSRAM controller timing diagrams

- read accesses

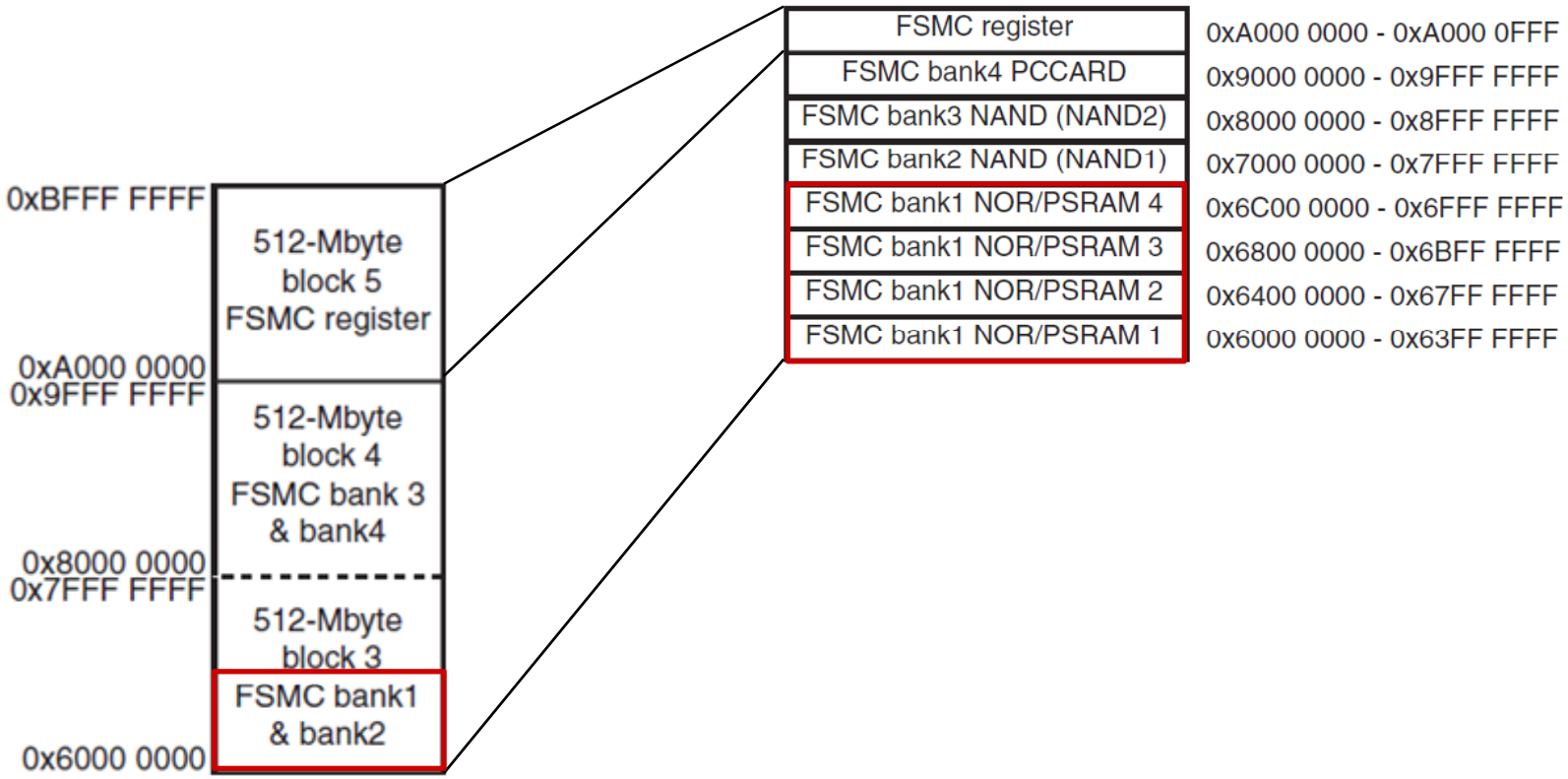


NOR Flash/PSRAM controller timing diagrams

- write accesses

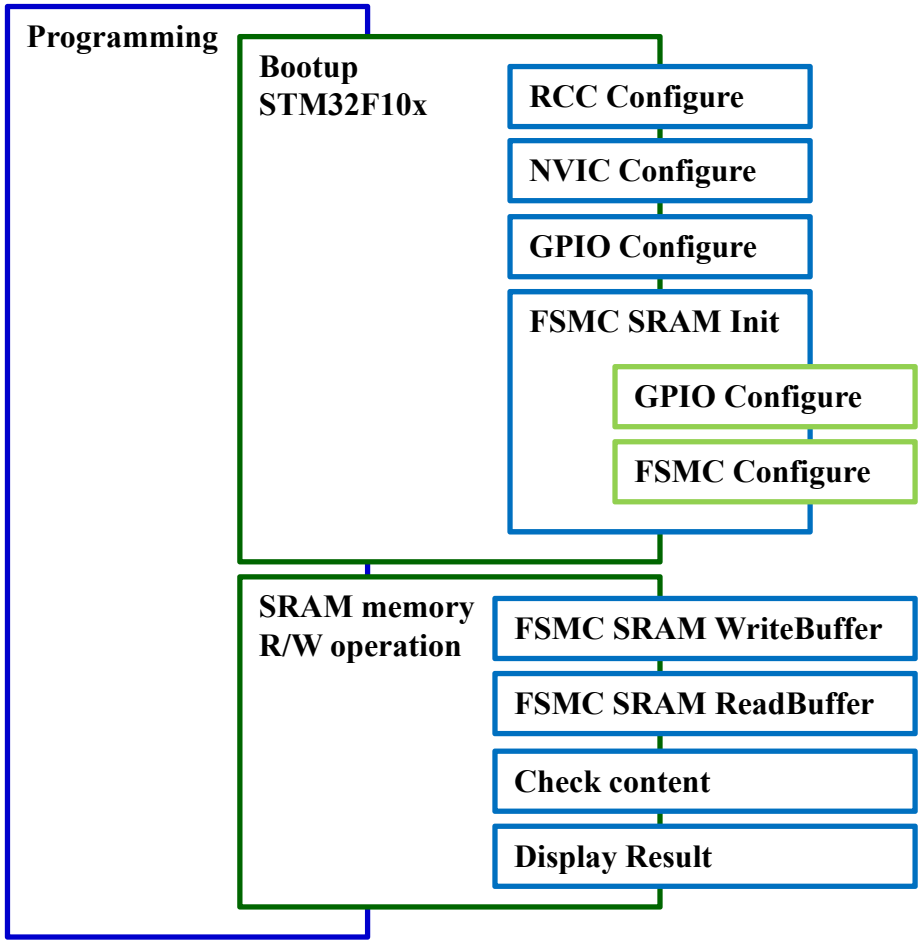


FSMC Memory map



Development Flow

Embedded Software Side



Bootup STM32F10x

```
int main(void)
{
#ifdef DEBUG
  debug();
#endif

  /* System Clocks Configuration */
  RCC_Configuration();

  /* NVIC Configuration */
  NVIC_Configuration();

  /* GPIO Configuration */
  GPIO_Configuration();

/* Write/read to/from FSMC SRAM memory */

  /* Configure FSMC Bank1 NOR/SRAM1 */
  FSMC_SRAM_Init();


……..
```

Configure FSMC IO

GPIO FwLib Functions List

Function name	Description
RCC_APB2PeriphClock Cmd	Enables or disables the High Speed APB (APB2) peripheral clock.
GPIO_Init	Initializes the GPIOx peripheral according to the specified parameters in the GPIO_InitStruct.
RCC_AHBPeriphClock Cmd	Enables or disables the AHB peripheral clock.

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOG |
RCC_APB2Periph_GPIOE | RCC_APB2Periph_GPIOF, ENABLE);

/*-- GPIO Configuration ------------------------------------------------*/
  /* SRAM Data lines configuration */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_8 |
GPIO_Pin_9 | GPIO_Pin_10 | GPIO_Pin_14 | GPIO_Pin_15;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOD, &GPIO_InitStructure);
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7 | GPIO_Pin_8 | GPIO_Pin_9 |
GPIO_Pin_10 | GPIO_Pin_11 | GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 |
GPIO_Pin_15;
  GPIO_Init(GPIOE, &GPIO_InitStructure);
  /* SRAM Address lines configuration */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_12 | GPIO_Pin_13 |
GPIO_Pin_14 | GPIO_Pin_15;
  GPIO_Init(GPIOF, &GPIO_InitStructure);
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_Init(GPIOG, &GPIO_InitStructure);
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_12 | GPIO_Pin_13;
  GPIO_Init(GPIOD, &GPIO_InitStructure);
  /* NOE and NWE configuration */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 |GPIO_Pin_5;
  GPIO_Init(GPIOD, &GPIO_InitStructure);
  /* NE1configuration */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
  GPIO_Init(GPIOD, &GPIO_InitStructure);
  /* NBL0, NBL1 configuration */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
  GPIO_Init(GPIOE, &GPIO_InitStructure);
  /* Enable the FSMC Clock */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC, ENABLE);
```

Configure FSMC

FSMC FwLib Functions List

Function name	Description
RCC_AHBPeriphClock Cmd	Enables or disables the AHB peripheral clock.
FSMC_NORSRAMInit	Initializes the FSMC NOR memory bank according to the parameters specified in FSMC_NORInitStruct.
FSMC_NORSRAMCm	Enables or disables the NOR/SRAM memory bank1.

```
/* Enable the FSMC Clock */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC, ENABLE);

/*-- FSMC Configuration --*/
  p.FSMC_AddressSetupTime = 0;
  p.FSMC_AddressHoldTime = 0;
  p.FSMC_DataSetupTime = 2;
  p.FSMC_BusTurnAroundDuration = 0;
  p.FSMC_CLKDivision = 0;
  p.FSMC_DataLatency = 0;
  p.FSMC_AccessMode = FSMC_AccessMode_A;
  FSMC_NORSRAMInitStructure.FSMC_Bank = FSMC_Bank1_NORSRAM1;
  FSMC_NORSRAMInitStructure.FSMC_DataAddressMux =
FSMC_DataAddressMux_Disable;
  FSMC_NORSRAMInitStructure.FSMC_MemoryType = FSMC_MemoryType_SRAM;
  FSMC_NORSRAMInitStructure.FSMC_MemoryDataWidth = FSMC_MemoryDataWidth_16b;
  FSMC_NORSRAMInitStructure.FSMC_BurstAccessMode =
FSMC_BurstAccessMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalPolarity =
FSMC_WaitSignalPolarity_Low;
  FSMC_NORSRAMInitStructure.FSMC_WrapMode = FSMC_WrapMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalActive =
FSMC_WaitSignalActive_BeforeWaitState;
  FSMC_NORSRAMInitStructure.FSMC_WriteOperation =
FSMC_WriteOperation_Enable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignal = FSMC_WaitSignal_Disable;
  FSMC_NORSRAMInitStructure.FSMC_ExtendedMode = FSMC_ExtendedMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_AsyncWait = FSMC_AsyncWait_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WriteBurst = FSMC_WriteBurst_Disable;
  FSMC_NORSRAMInitStructure.FSMC_ReadWriteTimingStruct = &p;
  FSMC_NORSRAMInitStructure.FSMC_WriteTimingStruct = &p;
  FSMC_NORSRAMInit(&FSMC_NORSRAMInitStructure);

  /* Enable FSMC Bank1_SRAM Bank */
  FSMC_NORSRAMCmd(FSMC_Bank1_NORSRAM1, ENABLE);
```

硬體電路配置

Mapping Table

Num.	SRAM	STM32	Num.	SRAM	STM32	Num.	SRAM	STM32
1	A0	FSMC_A0	14	A13	FSMC_A13	27	I/O8	FSMC_D8
2	A1	FSMC_A1	15	A14	FSMC_A14	28	I/O9	FSMC_D9
3	A2	FSMC_A2	16	A15	FSMC_A15	29	I/O10	FSMC_D10
4	A3	FSMC_A3	17	A16	FSMC_A16	30	I/O11	FSMC_D11
5	A4	FSMC_A4	18	A17	FSMC_A17	31	I/O12	FSMC_D12
6	A5	FSMC_A5	19	I/O0	FSMC_D0	32	I/O13	FSMC_D13
7	A6	FSMC_A6	20	I/O1	FSMC_D1	33	I/O14	FSMC_D14
8	A7	FSMC_A7	21	I/O2	FSMC_D2	34	I/O15	FSMC_D15
9	A8	FSMC_A8	22	I/O3	FSMC_D3	35	CE	FSMC_nNE1
10	A9	FSMC_A9	23	I/O4	FSMC_D4	36	OE	FSMC_nOE
11	A10	FSMC_A10	24	I/O5	FSMC_D5	37	WE	FSMC_nWE
12	A11	FSMC_A11	25	I/O6	FSMC_D6	38	UB	FSMC_NBL1
13	A12	FSMC_A12	26	I/O7	FSMC_D7	39	LB	FSMC_NBL0

實驗步驟

- 範例目錄架構
- 範例說明
- 預設定義說明
- 燒錄MIAT_STM32

範例目錄架構

- 範例目錄
  - 測試映像檔
  - 含括檔
  - 函式庫
  - 專案檔
  - 原始碼

範例說明

Embedded Software Side

SRAM memory R/W operation

- **FSMC SRAM WriteBuffer**
- **FSMC SRAM ReadBuffer**
- **Check content**
- **Display Result**

SRAM memory R/W operation

```
/* Write data to FSMC SRAM memory */
/* Fill the buffer to send */
Fill_Buffer(TxBuffer, BUFFER_SIZE, 0x3212);
FSMC_SRAM_WriteBuffer(TxBuffer, WRITE_READ_ADDR, BUFFER_SIZE);


/* Read data from FSMC SRAM memory */
FSMC_SRAM_ReadBuffer(RxBuffer, WRITE_READ_ADDR, BUFFER_SIZE);

/* Read back SRAM memory and check content correctness */
for (Index = 0x00; (Index < BUFFER_SIZE) && (WriteReadStatus == 0);
Index++)
{
  if (RxBuffer[Index] != TxBuffer[Index])
  {
    WriteReadStatus = Index + 1;
  }
}
```

範例說明

Embedded Software Side

SRAM memory
R/W operation

- **FSMC SRAM WriteBuffer**
- **FSMC SRAM ReadBuffer**
- **Check content**
- **Display Result**

Display Result

```
while (1)
 {
  if (WriteReadStatus == 0)
  {  /* OK Turn on USERLED */
   GPIO_SetBits(GPIOF, GPIO_Pin_11);
  }
  else
  { /* KO Turn off USERLED */
   GPIO_ResetBits(GPIOF, GPIO_Pin_11);
   /* Insert delay */
   Delay(0xAFFFF);
   /* Turn on USERLED */
   GPIO_SetBits(GPIOF, GPIO_Pin_11);
   /* Insert delay */
   Delay(0xAFFFF);
  }
 }
```

如果寫入與讀取Buffer內容相同，外部記憶體使用正常，USERLED紅燈恆亮

如果寫入與讀取Buffer內容不同，外部記憶體使用異常，USERLED紅燈閃爍

預設定義說明

- #define Bank1_SRAM1_ADDR ((u32)0x60000000)
  - 定義SRAM起始點
- #define BUFFER_SIZE 0x400
  - 定義測試資料大小
  - 資料大小必需小於0xC000
- #define WRITE_READ_ADDR 0x8000
  - 定義SRAM寫入起始點
  - 寫入起始點WRITE_READ_ADDR + BUFFER_SIZE 必需小於0x6000C000

燒錄MIAT_STM32

- Rebuilder all target files產生HEX
- DFU File Manager轉換HEX產生DFU
- DfuSe Demonstration燒錄DFU
- Leave DFU mode

內部與外部SRAM存取控制實驗

實驗一

WU-YANG

Technology Co., Ltd.

實驗一練習

- 練習:
  - 修改寫入位置測試是否正常
  - 修改寫入資料大小測試是否正常
  - 取消FSMC_SRAM_Init測試是否正常

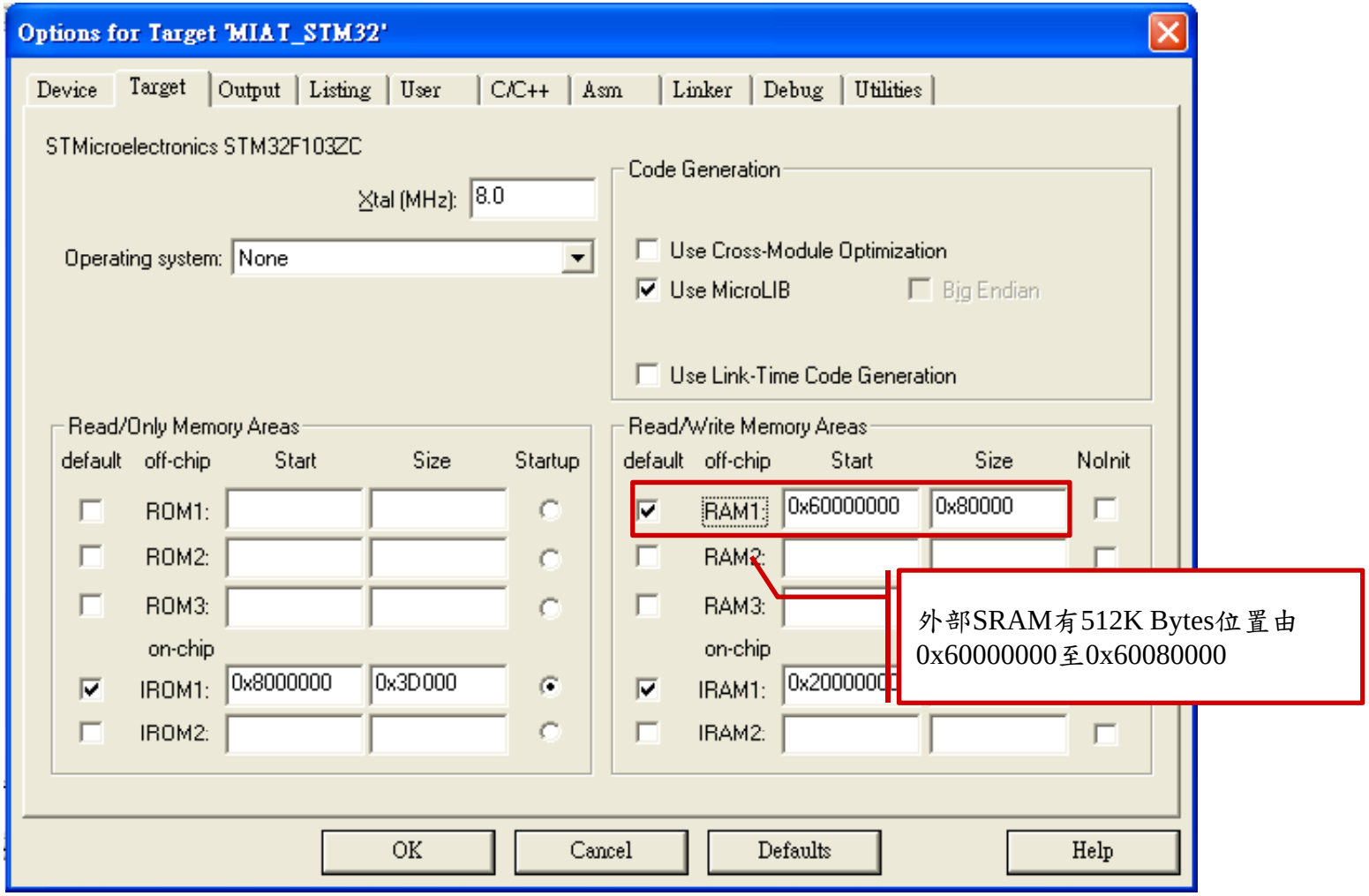
實驗目的(二)

- 使用MIAT_STM32實驗板透過FSMC控制外部SRAM並設定為data memory進行存取控制實驗，同樣利用LED確認存取是否正常。

實驗原理

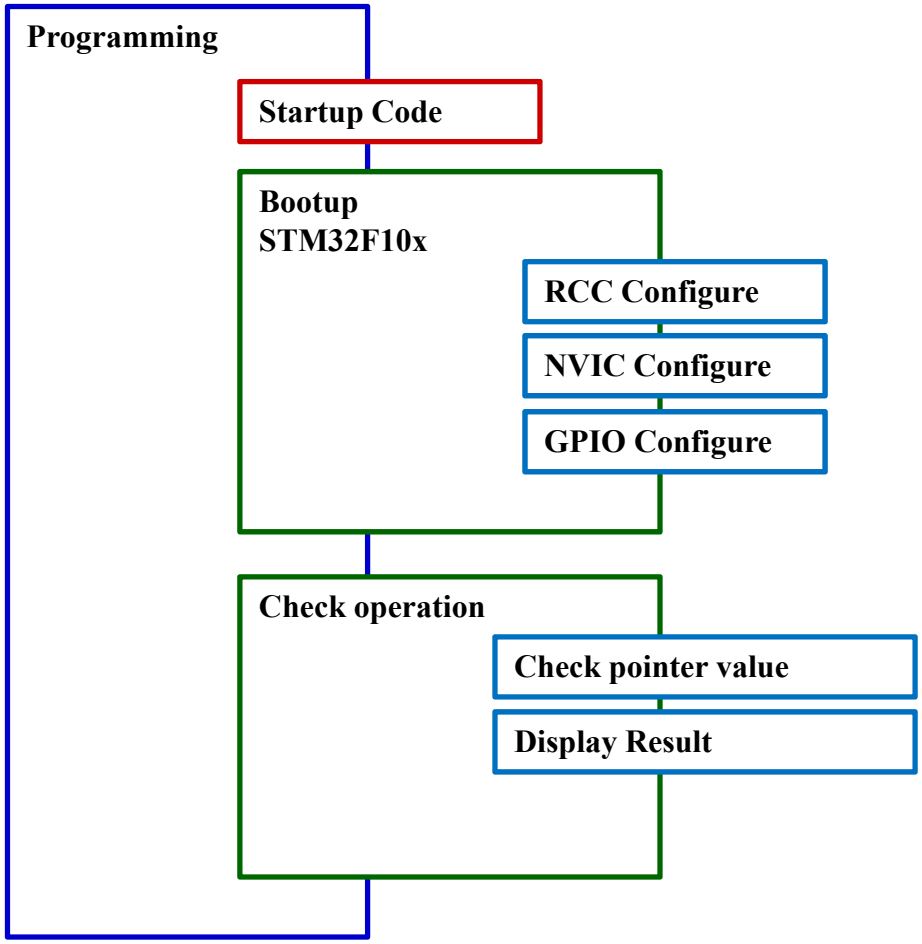
- External SRAM
- RVMDK環境設定
- Development Flow
- ARM Configure
- Startup Code

RVMDK環境設定


Options for Target 'MIAT_STM32'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
STMicroelectronics STM32F103ZC
Xtal (MHz): 8.0
Operating system: None
Code Generation
Use Cross-Module Optimization
Use MicroLIB
Big Endian
Use Link-Time Code Generation
Read/Only Memory Areas
default off-chip Start Size Startup
ROM1:
ROM2:
ROM3:
on-chip
IROM1: 0x8000000 0x3D000
IROM2:
Read/Write Memory Areas
default off-chip Start Size NoInit
RAM1: 0x60000000 0x80000
RAM2:
RAM3:
on-chip
IRAM1: 0x20000000
IRAM2:
外部SRAM有512K Bytes位置由
0x60000000至0x60080000
OK
Cancel
Defaults
Help

Development Flow

Embedded Software Side



Bootup STM32F10x

```
int main(void)
{
#ifdef DEBUG
  debug();
#endif

  /* System Clocks Configuration */
  RCC_Configuration();

  /* NVIC Configuration */
  NVIC_Configuration();

  /* GPIO Configuration */
  GPIO_Configuration();

 for (Index = 0; Index <1024 ; Index++)
  {
    Tab[Index] =Index;
  }
  TabAddr = (u32)Tab; /* should be 0x600000xx */

  /* Get main stack pointer value */
  MSPValue = __MRS_MSP(); /* should be 0x2000xxxx */
……..
```

Startup Code

Register boundary addresses

Boundary address	Peripheral	Bus
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC	AHB

RCC register map

Offset	Register
0x014	RCC_AHBENR
0x018	RCC_APB2ENR

RCC Configure

```
; Enable FSMC clock

LDR R0,= 0x00000114

LDR R1,= 0x40021014

STR R0,[R1]

; Enable GPIOD, GPIOE, GPIOF and GPIOG clocks

LDR R0,= 0x000001E0

LDR R1,= 0x40021018

STR R0,[R1]
```

RCC Register

AHB Peripheral Clock enable register (RCC_AHBENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					SDIO EN	Res.	FSMC EN	Res.	CRCE N	Res.	FLITF EN	Res.	SRAM EN	DMA2 EN	DMA1 EN
					rw		rw		rw		rw		rw	rw	rw

Bit 8 **FSMCEN: FSMC clock enable**
Set and cleared by software.
0: FSMC clock disabled
1: FSMC clock enabled

APB2 peripheral clock enable register (RCC_APB2ENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USAR T1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 8/7/6/5 **IOPGEN: I/O port G/F/E/D clock enable**
Set and cleared by software.
0: I/O port G/F/E/D clock disabled
1: I/O port G/F/E/D clock enabled

Startup Code

Register boundary addresses

Boundary address	Peripheral	Bus
0x4001 2000 - 0x4001 23FF	GPIO Port G	APB2
0x4001 1C00 - 0x4001 1FFF	GPIO Port F	
0x4001 1800 - 0x4001 1BFF	GPIO Port E	
0x4001 1400 - 0x4001 17FF	GPIO Port D	

GPIO register map

Offset	Register
0x00	GPIOx_CRL
0x04	GPIOx_CRH

GPIO Configure

```
; SRAM Data lines, NOE and NWE configuration
; SRAM Address lines configuration
; NOE, NEW, NE1, NBL0, NBL1 configuration
LDR R0,= 0xB4BB44BB
LDR R1,= 0x40011400
STR R0,[R1]
LDR R0,= 0xBBBBBBBB
LDR R1,= 0x40011404
STR R0,[R1]
LDR R0,= 0xB44444BB
LDR R1,= 0x40011800
STR R0,[R1]
LDR R0,= 0xBBBBBBBB
LDR R1,= 0x40011804
STR R0,[R1]
LDR R0,= 0x44BBBBBB
LDR R1,= 0x40011C00
STR R0,[R1]
LDR R0,= 0xBBBB4444
LDR R1,= 0x40011C04
STR R0,[R1]
LDR R0,= 0x44BBBBBB
LDR R1,= 0x40012000
STR R0,[R1]
```

GPIO Register

Port configuration register low

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Port configuration register high

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26,23:22, 19:18, 15:14,11:10, 7:6, 3:2
CNFy[1:0]: Port x configuration bits (y= 0 .. 15)
In input mode (MODE[1:0]=00):
00: Analog input mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] >00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24,
21:20, 17:16, 13:12,
9:8, 5:4, 1:0
MODEy[1:0]: Port x mode bits (y= 0 .. 15)
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

Startup Code

FSMCregister map

Offset	Register
0xA000 0000	FSMC_BCR1
0xA000 0004	FSMC_BTR1

FSMC Configure

```
; FSMC Configuration
; Enable FSMC Bank1_SRAM Bank


LDR R0,= 0x00001011

LDR R1,= 0xA0000000

STR R0,[R1]


LDR R0,= 0x00000200

LDR R1,= 0xA0000004

STR R0,[R1]
```

FSMC Register

SRAM/NOR-Flash chip-select control registers 1 (FSMC_BCR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CBURSTRW	Reserved				EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
												rw					rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 12 **WREN: Write enable bit.**
This bit indicates whether write operations are enabled/disabled in the bank by the FSMC:
0: Write operations are disabled in the bank by the FSMC, an AHB error is reported,
1: Write operations are enabled for the bank by the FSMC (default after reset).
Bits 5:4 **MWID: Memory databus width.**
Defines the external memory device width, valid for all type of memories.
00: 8 bits,
01: 16 bits (default after reset),
10: reserved, do not use,
11: reserved, do not use.
Bit 0 **MBKEN: Memory bank enable bit.**
Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.
0: Corresponding memory bank is disabled
1: Corresponding memory bank is enabled

FSMC Register

SRAM/NOR-Flash chip-select timing registers 1 (FSMC_BTR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

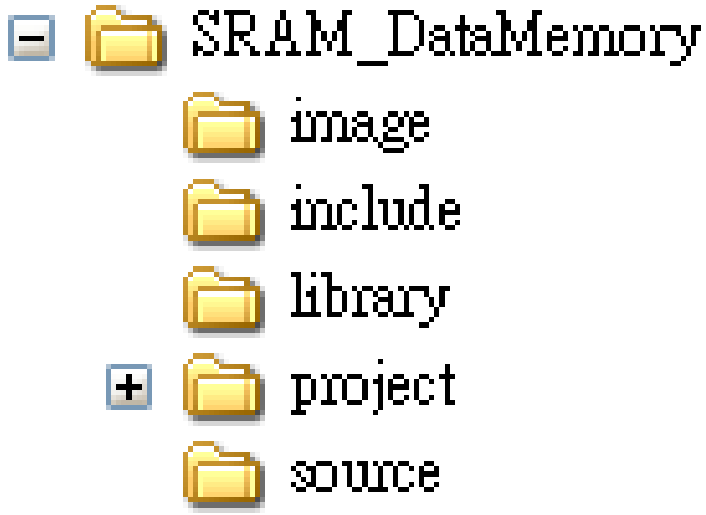
Bits 15:8 **DATAST: Data-phase duration**
0000 0000: Reserved
0000 0001: DATAST phase duration = 2 × HCLK clock cycles
0000 0010: DATAST phase duration = 3 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 256 × HCLK clock cycles (default value after reset)
Bits 7:4 **ADDHLD: Address-hold phase duration**
0000: Reserved
0001: ADDHLD phase duration = 2 × HCLK clock cycle
0010: ADDHLD phase duration = 3 × HCLK clock cycle
...
1111: ADDHLD phase duration = 16 × HCLK clock cycles (default value after reset)
Bits 3:0 **ADDSET: Address setup phase duration**
These bits are written by software to define the duration of the *address setup phase (refer to Figure 162 to Figure 172), used in SRAMs, ROMs and asynchronous NOR Flash:*
0000: ADDSET phase duration = 1 × HCLK clock cycle
...
1111: ADDSET phase duration = 16 × HCLK clock cycles (default value after reset)

實驗步驟

- 範例目錄架構
- 範例說明
- 預設定義說明

範例目錄架構

- 範例目錄
  - 測試映像檔
  - 含括檔
  - 函式庫
  - 專案檔
  - 原始碼



範例說明

Embedded Software Side

Check operation

- **Check pointer value**
- **Display Result**

Display Result

```
 /* Infinite loop */
 while (1)
 {
  if (((TabAddr&0xFFFFFF00) == 0x60000000) &&
((MSPValue&0xFFFF0000) == 0x20000000))
  {  /* OK Turn on USERLED */
   GPIO_SetBits(GPIOF, GPIO_Pin_11);
  }
  else
  { /* KO Turn off USERLED */
   GPIO_ResetBits(GPIOF, GPIO_Pin_11);
   /* Insert delay */
   Delay(0xAFFFF);
   /* Turn on USERLED */
   GPIO_SetBits(GPIOF, GPIO_Pin_11);
   /* Insert delay */
   Delay(0xAFFFF);
  }
```

如果Tab位置在0x600000??且Stack pointer位置在0x2000???? ，外部記憶體使用正常，USERLED紅燈恆亮

如果Tab位置不在0x600000??或Stack pointer位置不在0x2000???? ，外部記憶體使用異常，USERLED紅燈閃爍

預設定義說明

- DATA_IN_ExtSRAM EQU 1
  - External SRAM Configuration
    - 0=> DISABLE
    - 1=> ENABLE
- u32 Tab[1024]
  - 宣告於外部SRAM的記憶體

data memory存取控制實驗

實驗二

WU-YANG

Technology Co., Ltd.

實驗二練習

- 練習:
  - 取消RVMDK環境外部SRAM設定測試是否正常
  - 修改DATA_IN_ExtSRAM EQU 0測試是否正常
  - 修改外部SRAM記憶體Tab變數大小測試是否正常

Q & A

WU-YANG

Technology Co., Ltd.

MIAT_STM32
內部Flash存取控制實驗

浯陽科技有限公司
WU-YANG
Technology Co., Ltd.

Declared Version

Training Only

Declare	
Document Number	
Document Version	***1.00***
Release Date	
Document Title	MIAT_STM32 內部FLASH存取控制實驗
Exercise Time	■
Platform	■ ***MIAT_STM32.V2*** ■ ***MIAT IOB.V1***
Peripheral	■
Author	■ ***WU-YANG Technology Co., Ltd.***

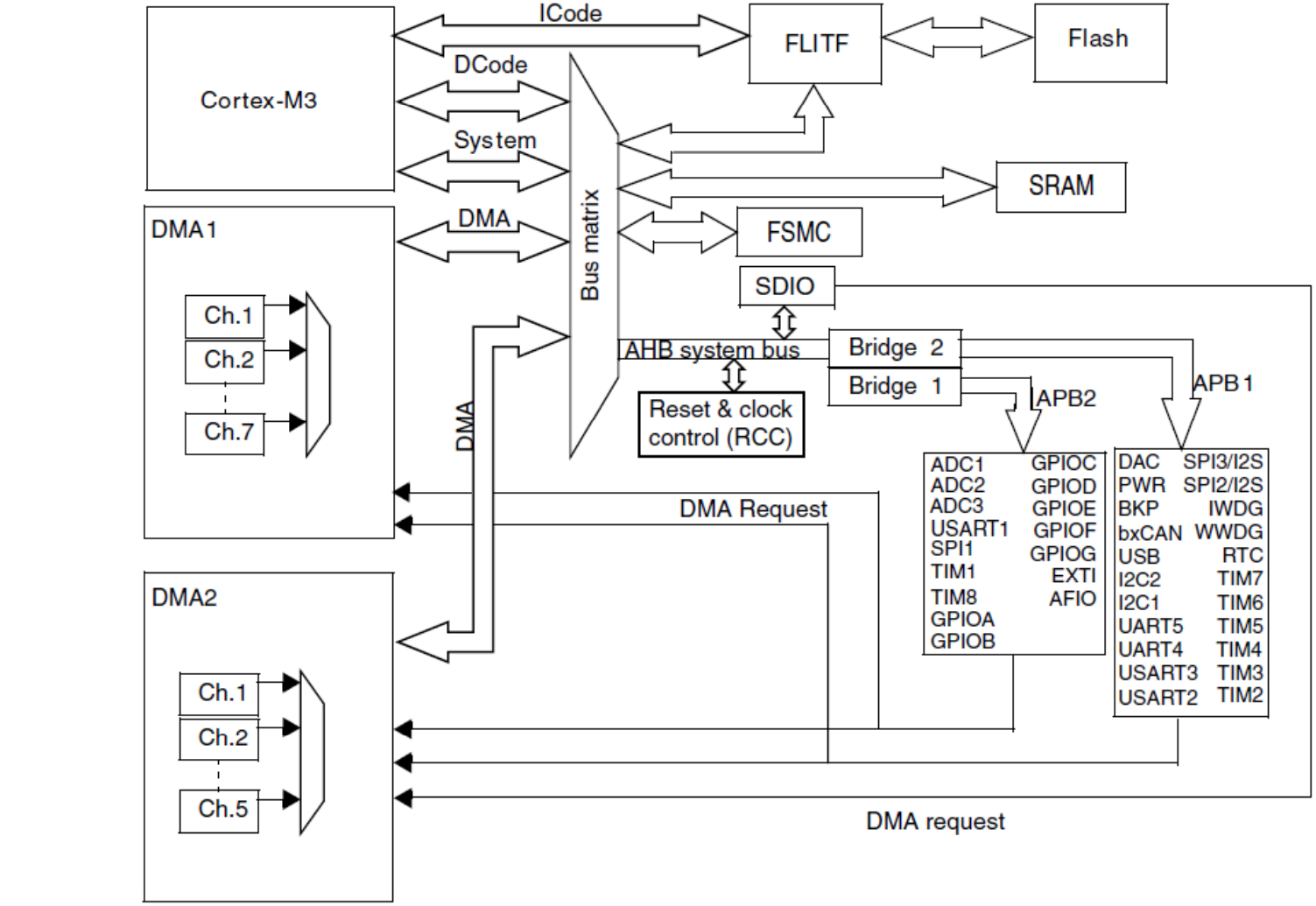
實驗目的(一)

- 使用MIAT_STM32實驗板透過Flash memory interface (FLITF)控制內部Flash進行存取控制實驗，並利用LED確認存取是否正常。

實驗原理

- System architecture
- Embedded Flash
  - Features
  - Memory map
  - RVMDK環境設定
  - FLITF
  - Flash module organization
- Flash library function
- Development Flow
- ARM Configure

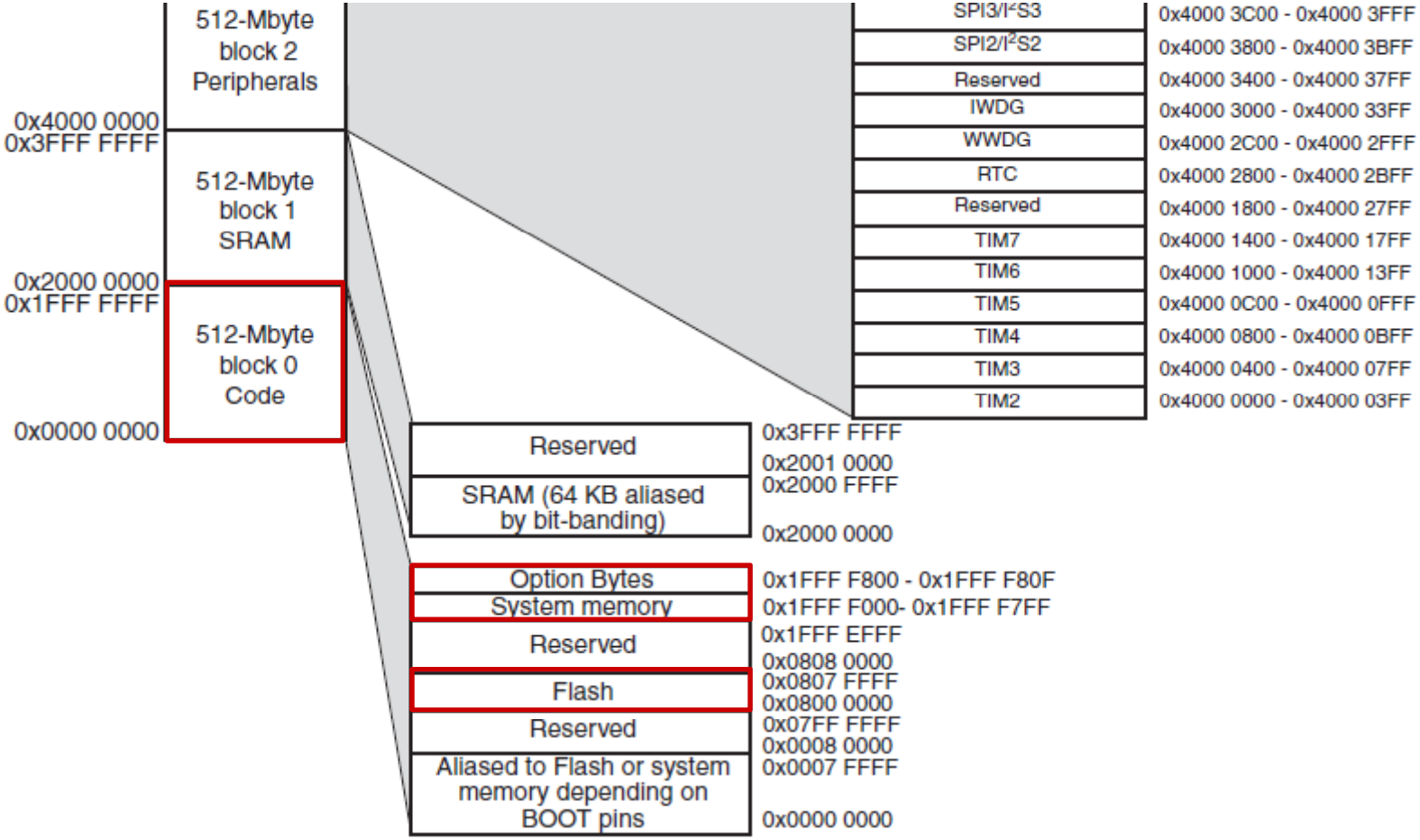
System architecture



Embedded Flash

- Features
- up to 256 Kbytes
- Memory organization: the Flash memory is organized as a main block and an information block:
  - Main memory block of size:
    - up to 32 Kb × 64 bits divided into 128 pages of 2 Kbytes
  - Information block of size:
    - 258 × 64 bits

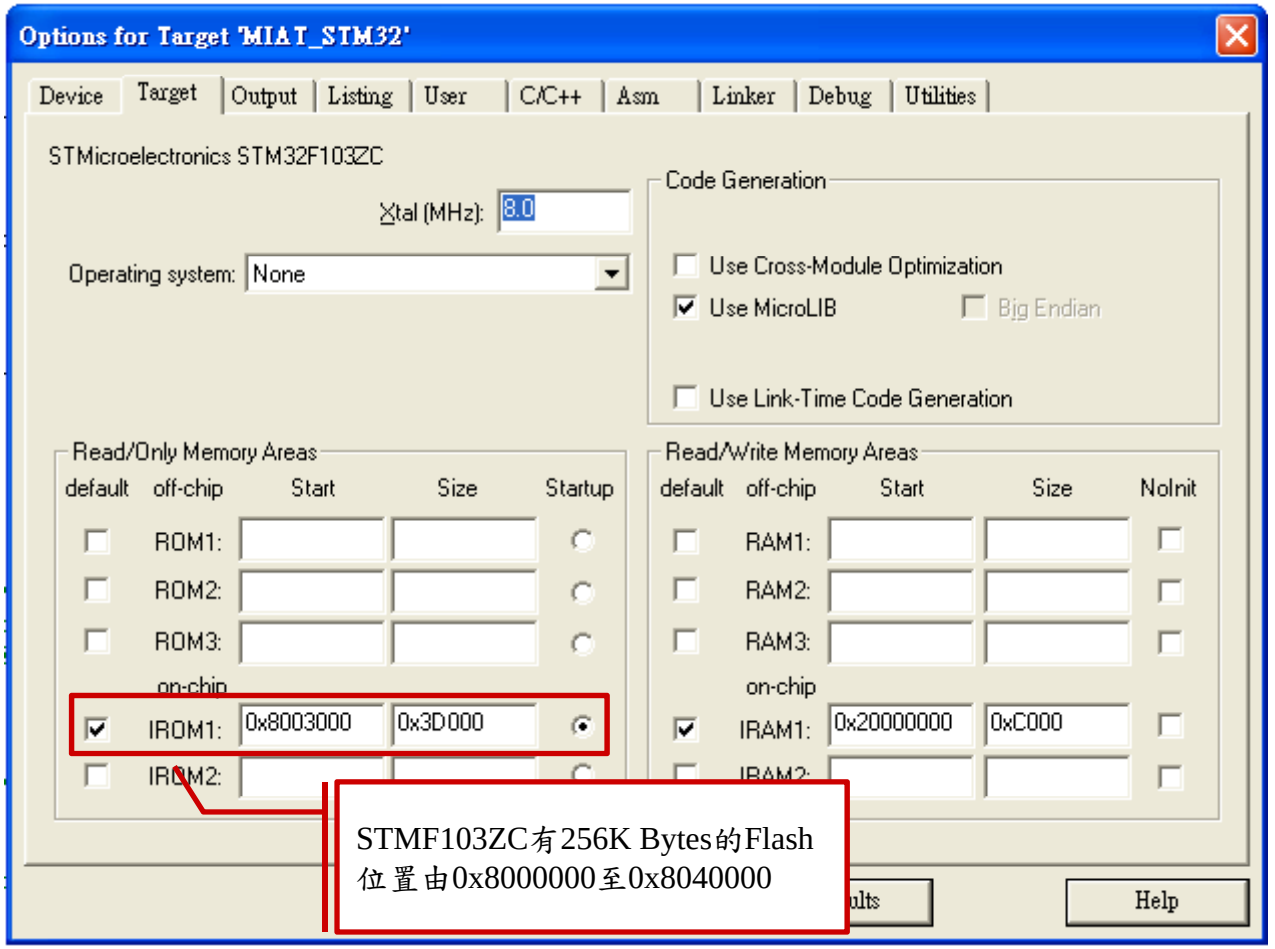
Embedded Flash Memory map



Flash Memory

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 07FF	2 Kbytes
	Page 1	0x0800 0800 - 0x0800 0FFF	2 Kbytes
	Page 2	0x0800 1000 - 0x0800 17FF	2 Kbytes
	Page 3	0x0800 1800 - 0x0800 1FFF	2 Kbytes
	. . .	. . .	. . .
	Page 255	0x0807 F800 - 0x0807 FFFF	2 Kbytes
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16

RVMDK環境設定


Options for Target 'MIAT_STM32'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
STMicroelectronics STM32F103ZC
Xtal (MHz): 8.0
Operating system: None
Code Generation
Use Cross-Module Optimization
Use MicroLIB
Big Endian
Use Link-Time Code Generation
Read/Only Memory Areas
default
off-chip
Start
Size
Startup
ROM1:
ROM2:
ROM3:
on-chip
IROM1: 0x8003000 0x3D000
IROM2:
Read/Write Memory Areas
default
off-chip
Start
Size
NoInit
RAM1:
RAM2:
RAM3:
on-chip
IRAM1: 0x20000000 0xC000
Help
STMF103ZC有256K Bytes的Flash
位置由0x8000000至0x8040000

FLITF

- Features
  - Read interface with prefetch buffer (2x64-bit words)
  - Option byte Loader
  - Flash Program / Erase operation
  - Read / Write protection

Flash memory interface

Block	Name	Base addresses	Size (bytes)
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Flash memory interface register

Flash status register (FLASH_SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EOP	WRPRT ERR	Res.	PG ERR	Res.	BSY
										rw	rw		rw		r

Bits 31:6 Reserved, must be kept cleared.
Bit 5 **EOP: End of operation**
Set by hardware when a Flash operation (programming / erase) is completed. Reset by writing a 1
Note: EOP is asserted at the end of each successful program or erase operation
Bit 4 **WRPRTERR: Write protection error**
Set by hardware when programming a write-protected address of the Flash memory.
Reset by writing 1.
Bit 3 Reserved, must be kept cleared.

Bit 2 **PGERR: Programming error**
Set by hardware when an address to be programmed contains a value different from '0xFFFF' before programming.
Reset by writing 1.
Note: The STRT bit in the FLASH_CR register should be reset before starting a programming operation.
Bit 1 Reserved, must be kept cleared
Bit 0 **BSY: Busy**
This indicates that a Flash operation is in progress. This is set on the beginning of a Flash operation and reset when the operation finishes or when an error occurs.

Flash memory interface register

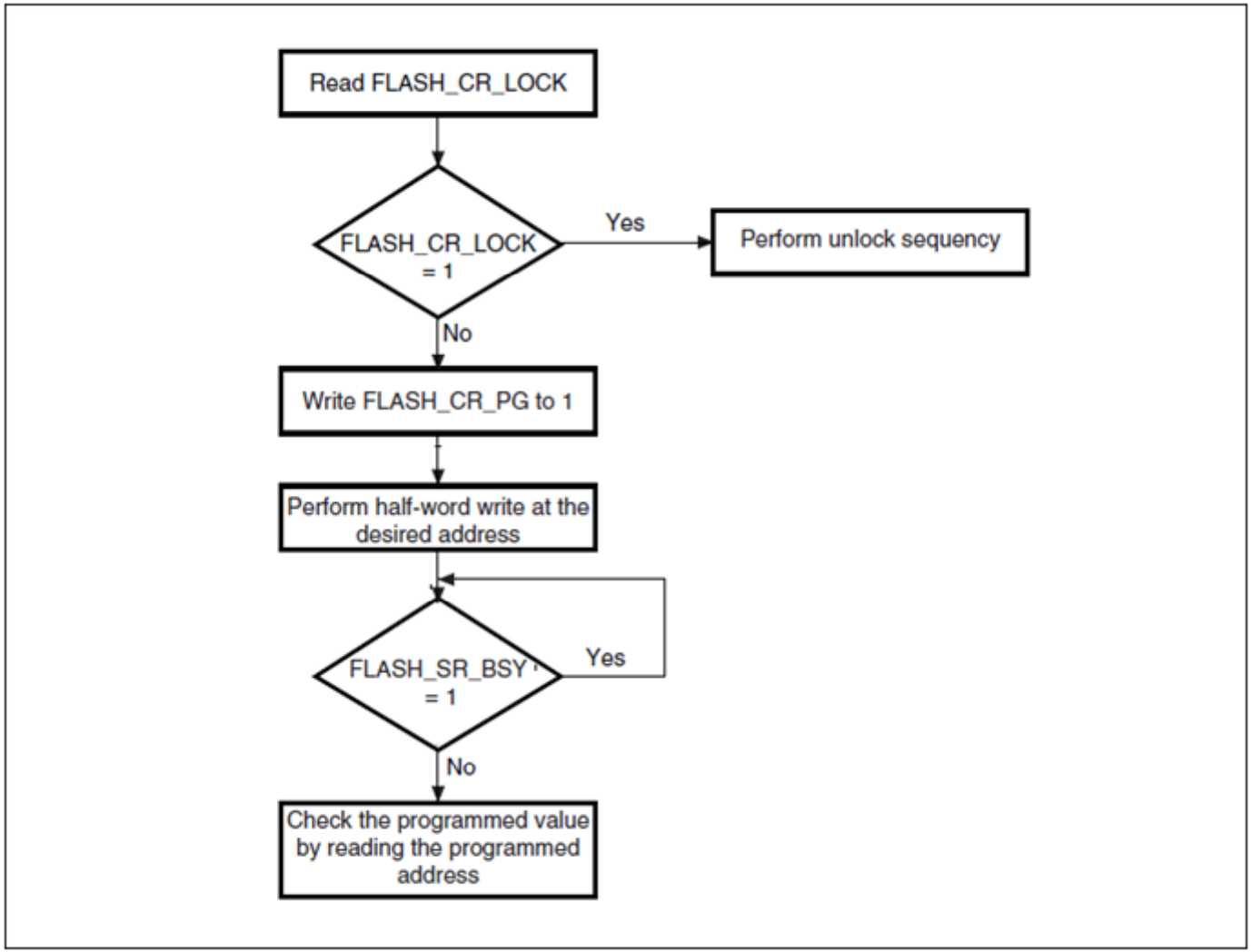
Flash control register (FLASH_CR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			EOPIE	Res.	ERRIE	OPTWRE	Res.	LOCK	STRT	OPTER	OPTPG	Res.	MER	PER	PG
			rw		rw	rw		rw	rw	rw	rw		rw	rw	rw

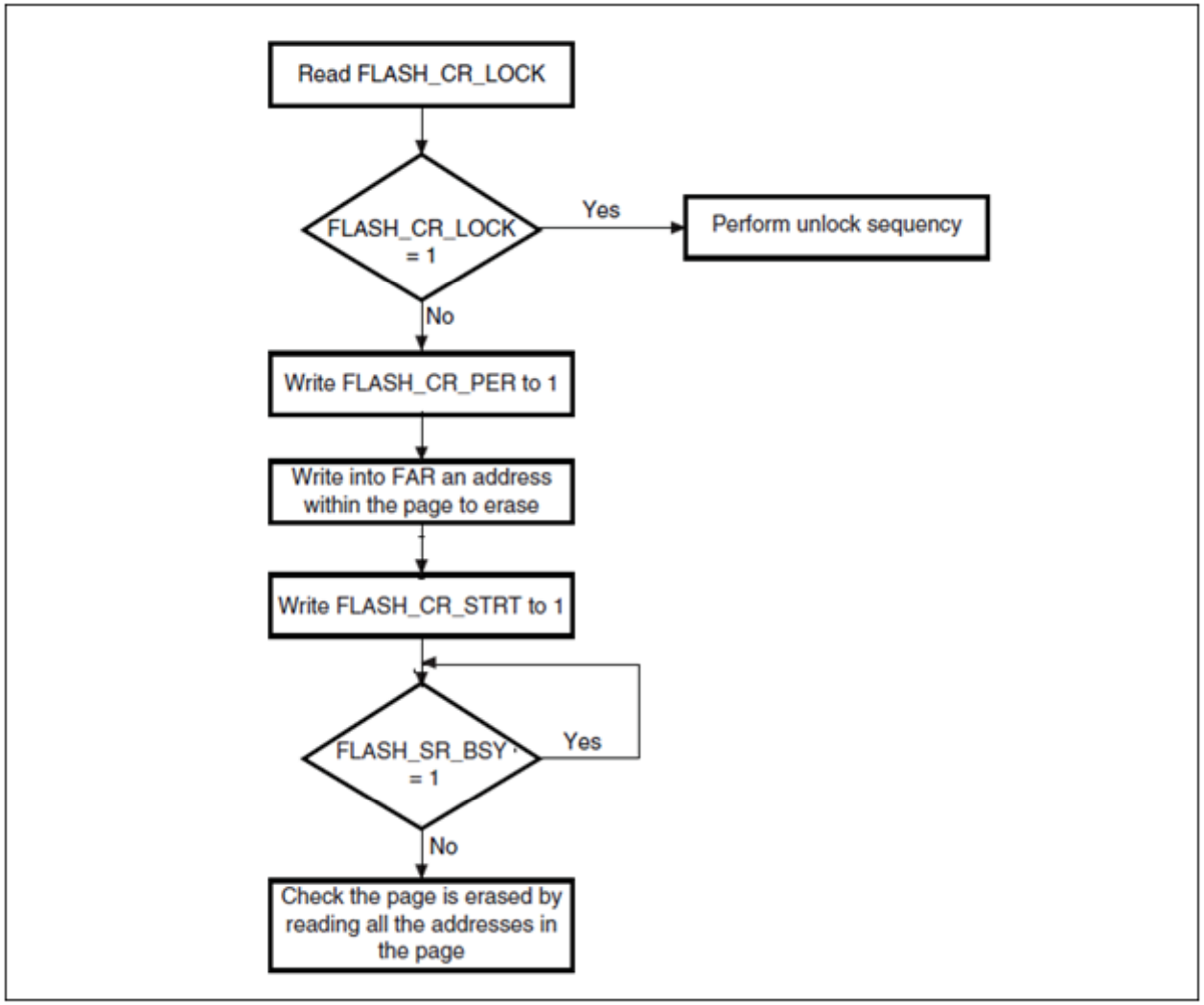
Bit 9 **OPTWRE: Option bytes write enable**
When set, the option bytes can be programmed. This bit is set on writing the correct key sequence to the FLASH_OPTKEYR register.
This bit can be reset by software
Bit 8 Reserved, must be kept cleared.
Bit 7 **LOCK: Lock**
Write to 1 only. When it is set, it indicates that the FPEC and FLASH_CR are locked. This bit is reset by hardware after detecting the unlock sequence.
In the event of unsuccessful unlock operation, this bit remains set until the next reset.

Bit 6 **STRT: Start**
This bit triggers an ERASE operation when set. This bit is set only by software and reset when the BSY bit is reset.
Bit 5 **OPTER: Option byte erase**
Option byte erase chosen.
Bit 4 **OPTPG: Option byte programming**
Option byte programming chosen.
Bit 3 Reserved, must be kept cleared.
Bit 2 **MER: Mass erase**
Erase of all user pages chosen.
Bit 1 **PER: Page erase**
Page Erase chosen.
Bit 0 **PG: Programming**
Flash programming chosen.

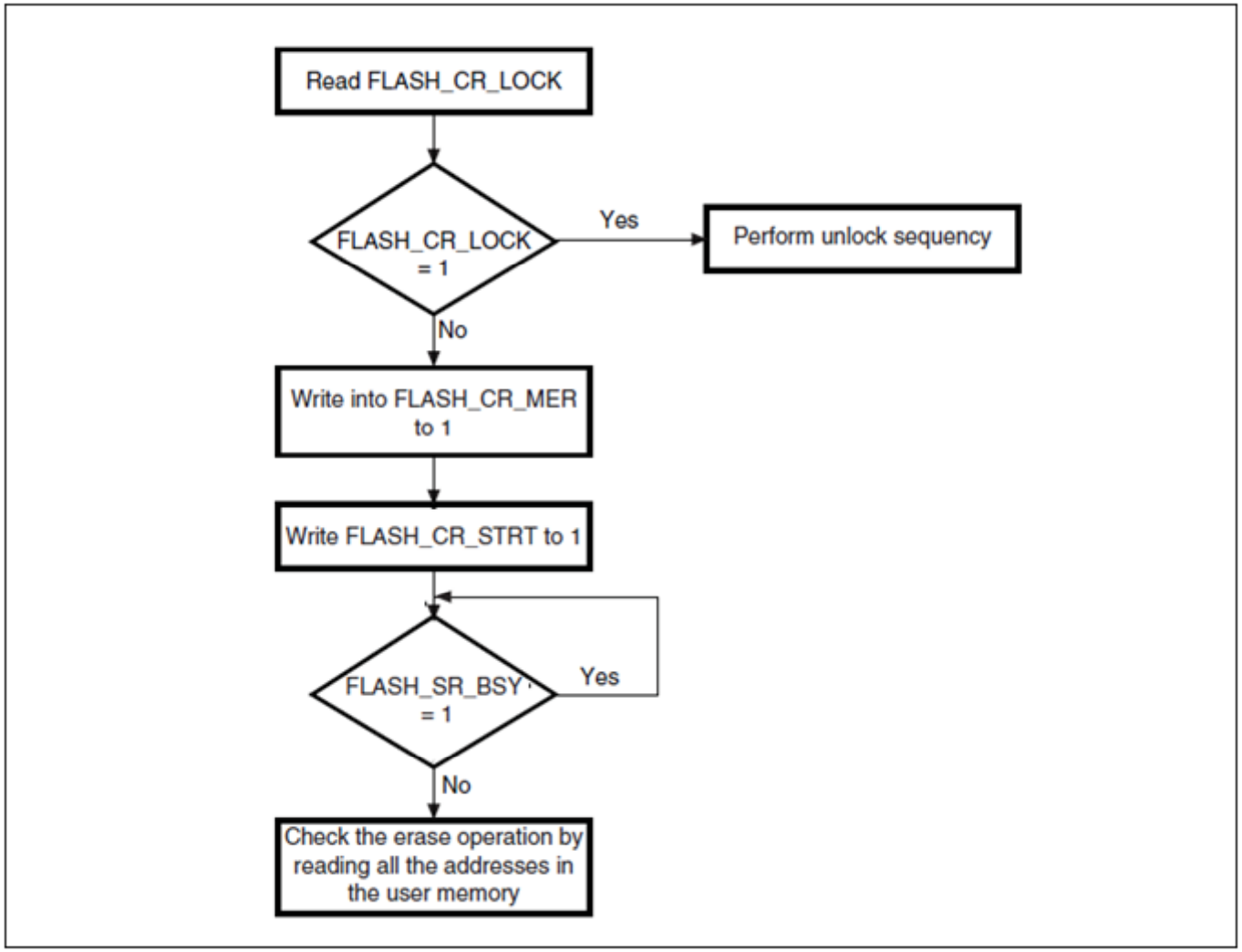
Main Flash memory programming



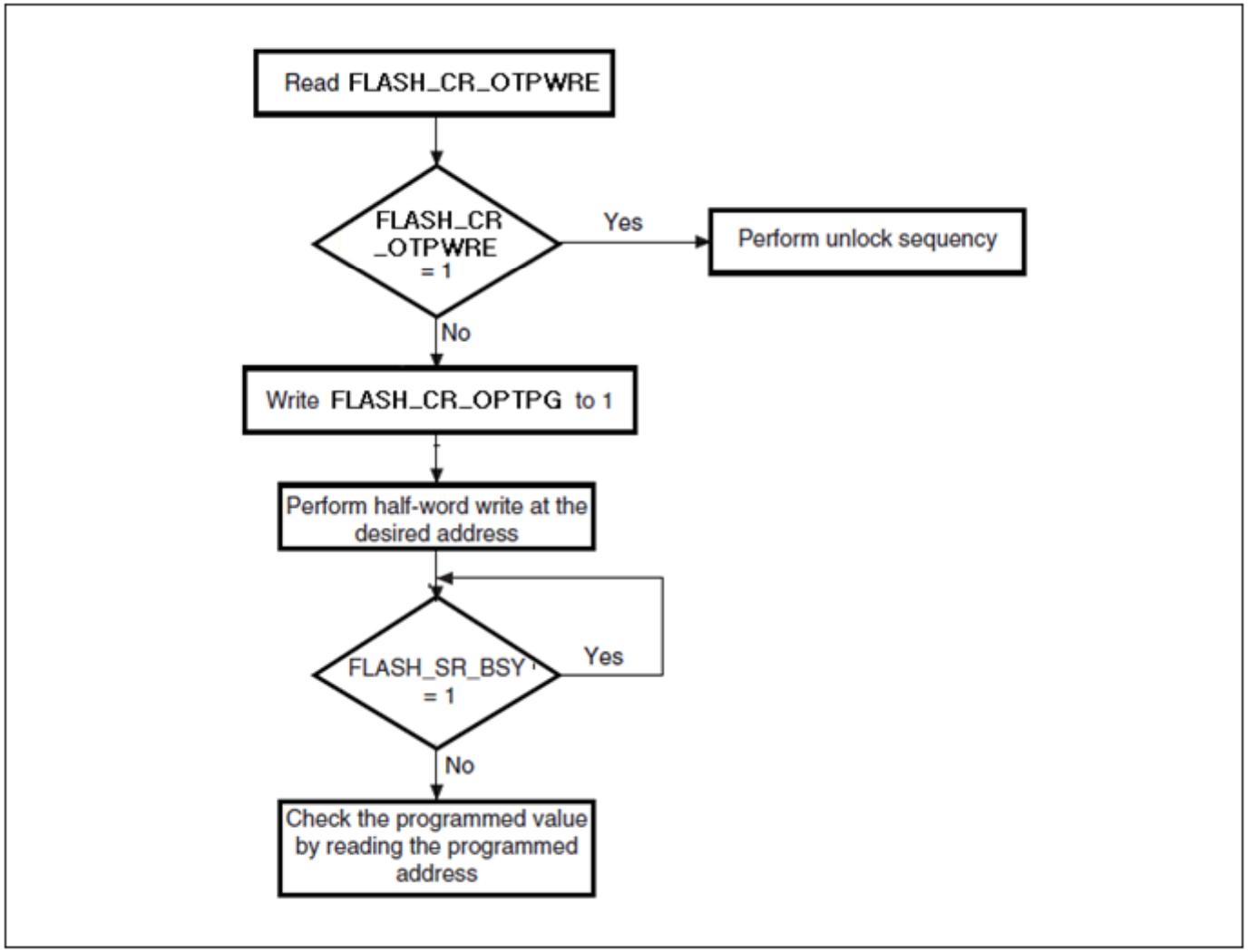
Flash memory Page Erase



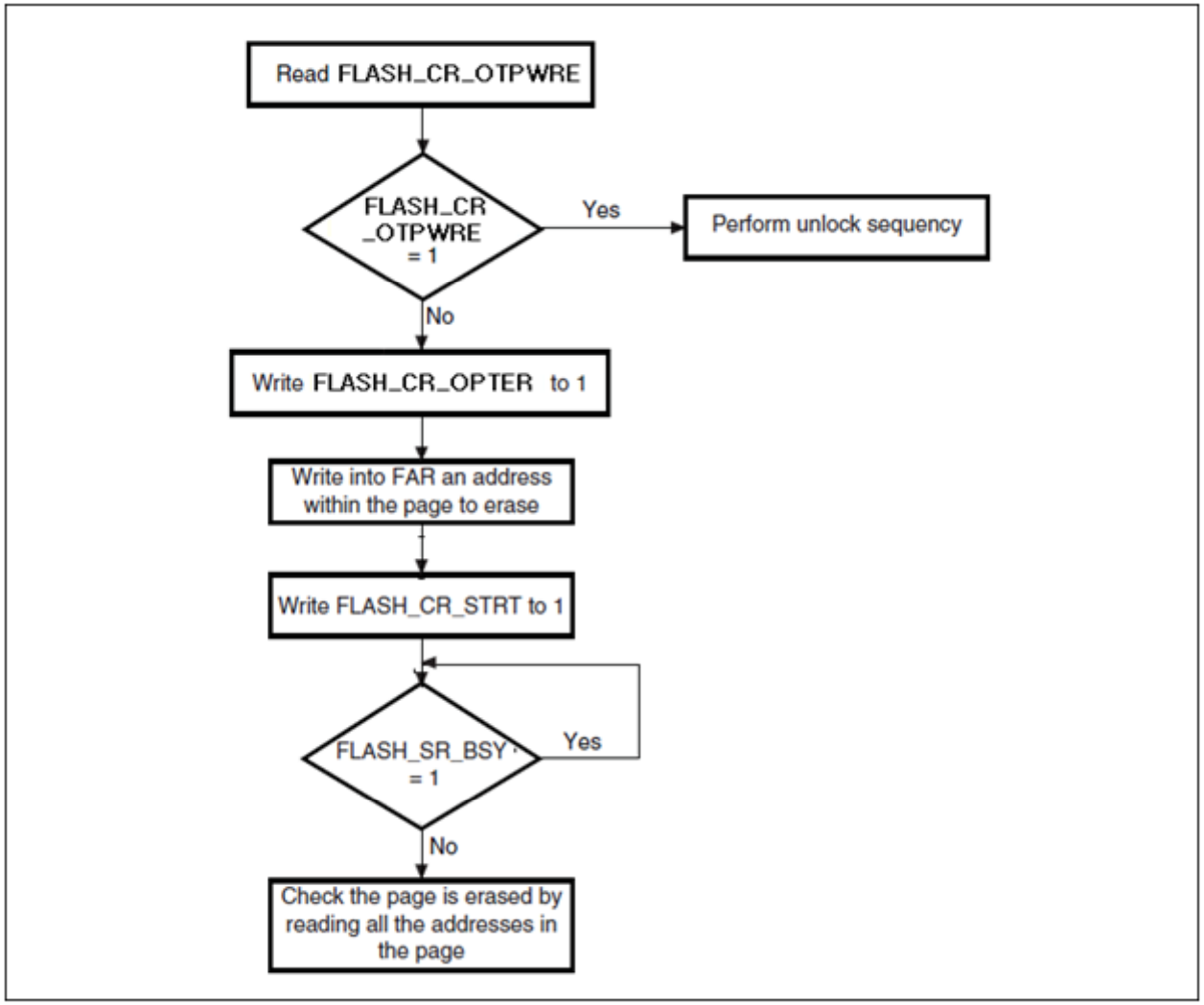
Flash memory Mass Erase



Option byte programming



Option byte Erase



FLASH library function

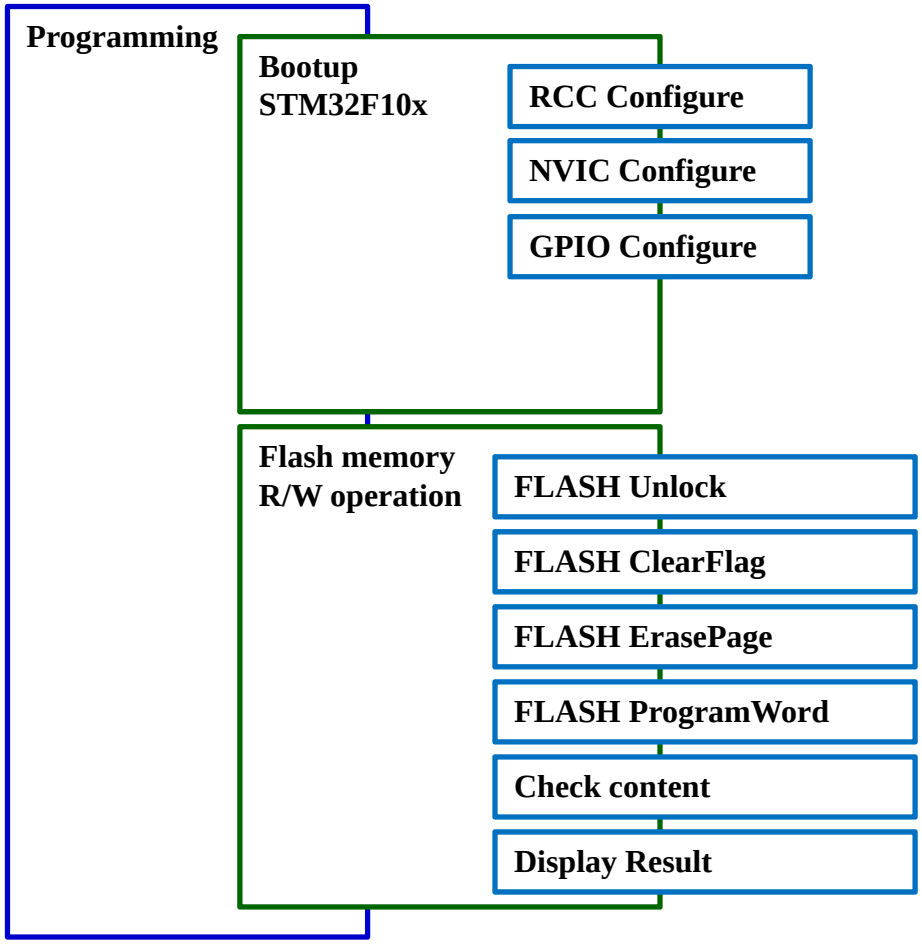
Function name	Description
FLASH_SetLatency	Sets the code latency value.
FLASH_HalfCycleAccessCmd	Enables or disables the Half cycle FLASH access.
FLASH_PrefetchBufferCmd	Enables or disables the Prefetch Buffer.
FLASH_Unlock	Unlocks the FLASH Program Erase Controller.
FLASH_Lock	Locks the Flash Program Erase Controller.
FLASH_ErasePage	Erases a specified FLASH page.
FLASH_EraseAllPages	Erases all FLASH pages.
FLASH_EraseOptionBytes	Erases the FLASH option bytes.
FLASH_ProgramWord	Programs a word at a specified address.
FLASH_ProgramHalfWord	Programs a half word at a specified address.
FLASH_ProgramOptionByteData	Programs a half word at a specified Option Byte Data address.
FLASH_EnableWriteProtection	Write protects the desired pages

FLASH library function

Function name	Description
FLASH_ReadOutProtection	Enables or disables the read out protection.
FLASH_UserOptionByteConfig	Programs the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY.
FLASH_GetUserOptionByte	Returns the FLASH User Option Bytes values.
FLASH_GetWriteProtectionOptionByte	Returns the FLASH Write Protection Option Bytes Register value.
FLASH_GetReadOutProtectionStatus	Checks whether the FLASH Read Out Protection Status is set or not.
FLASH_GetPrefetchBufferStatus	Checks whether the FLASH Prefetch Buffer status is set or not.
FLASH_ITConfig	Enables or disables the specified FLASH interrupts.
FLASH_GetFlagStatus	Checks whether the specified FLASH flag is set or not.
FLASH_ClearFlag	Clears the FLASH pending flags.
FLASH_GetStatus	Returns the FLASH Status.
FLASH_WaitForLastOperation	Waits for a Flash operation to complete or a TIMEOUT to occur.

Development Flow

Embedded Software Side



Bootup STM32F10x

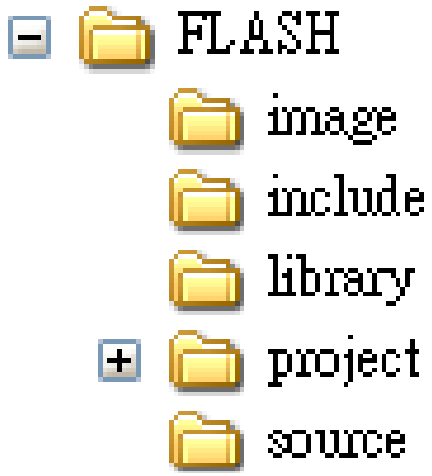
```
int main(void)
{
#ifdef DEBUG
  debug();
#endif

  FLASHStatus = FLASH_COMPLETE;
  MemoryProgramStatus = PASSED;
  Data = 0x15041975;

/* RCC Configuration */
  RCC_Configuration();

  /* NVIC Configuration */
  NVIC_Configuration();

  /* GPIO Configuration */
  GPIO_Configuration();


……..
```

實驗步驟

- 範例目錄架構
- 範例說明
- 預設定義說明
- 燒錄MIAT_STM32

範例目錄架構

- 範例目錄
  - 測試映像檔
  - 含括檔
  - 函式庫
  - 專案檔
  - 原始碼



範例說明

Flash FwLib Functions List

Function name	Description
FLASH_Unlock	Unlocks the FLASH Program Erase Controller.
FLASH_ClearFlag	Clears the FLASH pending flags.
FLASH_ErasePage	Erases a specified FLASH page.
FLASH_ProgramWord	Programs a word at a specified address.

```
/* Unlock the Flash Program Erase controller */
FLASH_Unlock();

/* Define the number of page to be erased */
NbrOfPage = (EndAddr - StartAddr) / FLASH_PAGE_SIZE;

/* Clear All pending flags */
FLASH_ClearFlag(FLASH_FLAG_BSY | FLASH_FLAG_EOP | FLASH_FLAG_PGERR |
FLASH_FLAG_WRPRTERR);

/* Erase the FLASH pages */
for(EraseCounter = 0; (EraseCounter < NbrOfPage) && (FLASHStatus ==
FLASH_COMPLETE); EraseCounter++)
{
  FLASHStatus = FLASH_ErasePage(StartAddr + (FLASH_PAGE_SIZE * EraseCounter));
}

/*  FLASH Word program of data 0x15041979 at addresses defined by StartAddr and
EndAddr*/
Address = StartAddr;

while((Address < EndAddr) && (FLASHStatus == FLASH_COMPLETE))
{
  FLASHStatus = FLASH_ProgramWord(Address, Data);
  Address = Address + 4;
}
```

範例說明

Embedded Software Side

Flash memory
R/W operation

Check content

Display Result

Display Result

```
/* Check the corectness of written data */
 Address = StartAddr;

 while((Address < EndAddr) && (MemoryProgramStatus != FAILED))
 {
  if((*(vu32*) Address) != Data)
  {
    MemoryProgramStatus = FAILED;
  }
  Address += 4;
 }

 while (1)
 {
  if (MemoryProgramStatus == PASSED)
  {  /* OK Turn on USERLED */
    GPIO_SetBits(GPIOF, GPIO_Pin_11);
  }
  else
  { /* KO Turn off USERLED */
    GPIO_ResetBits(GPIOF, GPIO_Pin_11);
    /* Insert delay */
    Delay(0xAFFFF);
    /* Turn on USERLED */
    GPIO_SetBits(GPIOF, GPIO_Pin_11);
    /* Insert delay */
    Delay(0xAFFFF);
  }
 }
```

如果寫入與讀取Flash內容相同，
Flash記憶體使用正常，
USERLED紅燈恆亮

如果寫入與讀取Flash內容不同，
Flash記憶體使用異常，
USERLED紅燈閃爍

預設定義說明

- #define StartAddr ((u32)0x08008000)
  - 定義Flash使用起始點
  - 使用起始點必需大於0x8003000 + Code Size
    - 附註: 0x8000000~0x8003000為DFU程式區塊，使用此區塊將造成無法燒錄程式
    - Code Size可由產生的HEX檔得知
- #define EndAddr   ((u32)0x0800C000)
  - 定義Flash使用結束點
  - 使用起始點必需小於0x8040000
- Data = 0x15041975;
  - 寫入資料
  - 32Bit
- #define FLASH_PAGE_SIZE   ((u16)0x800)
  - 每一個Page有2KByte

Intel HEX Format

:	//	aaaa	tt	dd	cc

field	Description
:	the colon that starts every Intel HEX record.
//	the record-length field that represents the number of data bytes (**dd**) in the record.
aaaa	the address field that represents the starting address for subsequent data in the record.
tt	the field that represents the HEX record type, which may be one of the following: **00** - data record **01** - end-of-file record **02** - extended segment address record **04** - extended linear address record
dd	a data field that represents one byte of data. A record may have multiple data bytes. The number of data bytes in the record must match the number specified by the // field.
cc	the checksum field that represents the checksum of the record. The checksum is calculated by summing the values of all hexadecimal digit pairs in the record modulo 256 and taking the two's complement.

HEX Example

```
:020000040800F2
:10300000181400204531000 8D9340008CD340008D8
……
:1038200042370008000000000000000000000017
:0C383000000000000000000000008C
:04000005080031318D
:00000001FF
```

- 資料填入起始位置為0x08003000
- 每一行有0x10(16)個Bytes
- 最後一筆資料為 :04000005080031318D
- 資料結束點為0x08033830
- (0x08003830 -0x08000000)/0x800 = 7
- 需以0x800為單位，所以使用的位置起點為0x08004000

燒錄MIAT_STM32

- Rebuilder all target files產生HEX
- DFU File Manager轉換HEX產生DFU
- DfuSe Demonstration燒錄DFU
- Leave DFU mode

內部Flash存取控制實驗

實驗一

WU-YANG

Technology Co., Ltd.

實驗一練習

- 注意:
  - 請使用預設0x08008000之後的位置，避免覆蓋DFU與使用者程式碼區塊
- 練習:
  - 修改存取位置測試是否正常
  - 修改寫入資料測試是否正常
  - 修改寫入資料後取消FLASH_Unlock測試是否正常
  - 修改寫入資料後取消FLASH_ErasePage測試是否正常

實驗目的(二)

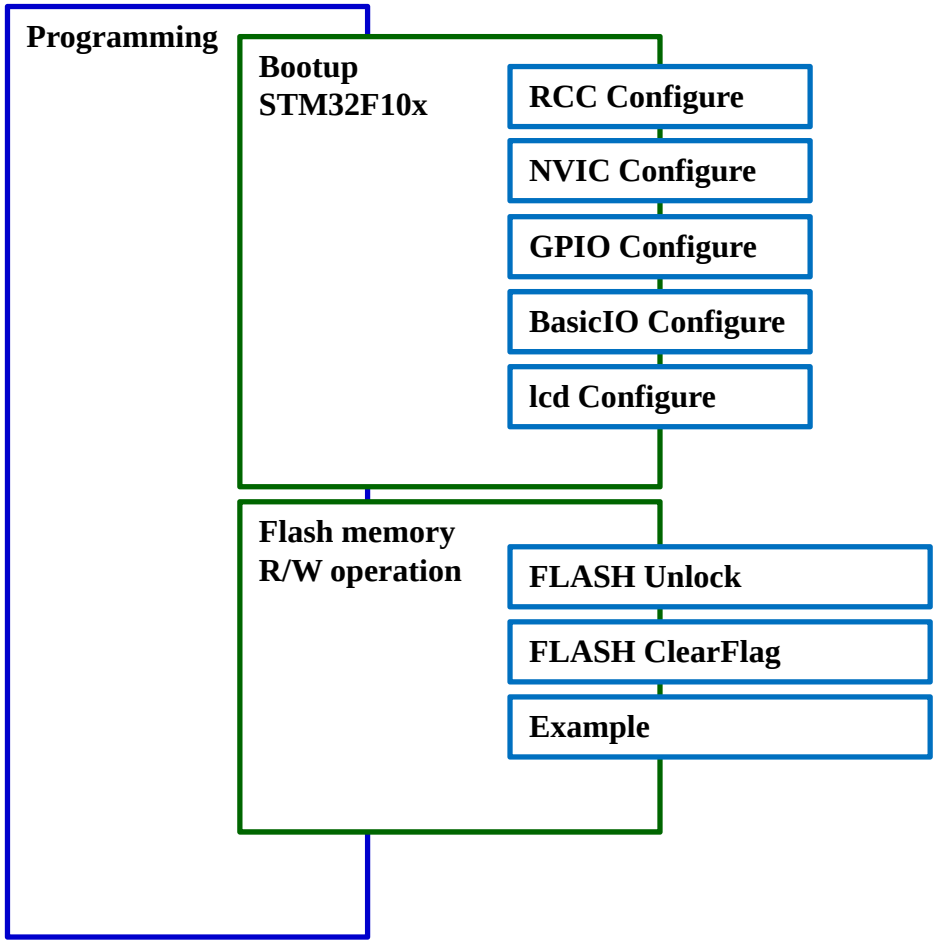
- 使用MIAT_STM32實驗板透過Flash memory interface (FLITF)控制內部Flash進行存取控制實驗，並利用SW、KEY決定資料寫入與讀取，LCD顯示狀態。

實驗原理

- System requirement
  - Embedded Flash
  - LCD
  - KEY
  - SW
- Development Flow
- ARM Configure

Development Flow

Embedded Software Side



Bootup STM32F10x

```
int main(void)
{
#ifdef DEBUG
  debug();
#endif

  FLASHStatus = FLASH_COMPLETE;

/* RCC Configuration */
  RCC_Configuration();

  /* NVIC Configuration */
  NVIC_Configuration();

  /* GPIO Configuration */
  GPIO_Configuration();

  Init_BasicIO();

  lcd_init();                          // LCD Initialization

……..
```

硬體電路配置

Mapping Table

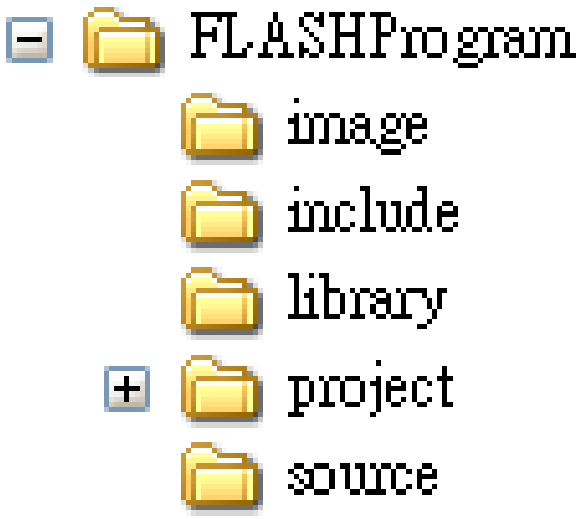
Num.	MIAT_STM32V2	MIAT_IOBV1	Num.	MIAT_STM32V2	MIAT_IOBV1
1	PC8 (3.26)	SW1	10	PE6 (1.5)	LCD_EN
2	PC9 (3.27)	SW2	11	PF6 (1.18)	LCD_R/W
3	PC10 (4.3)	SW3	12	PF7 (1.19)	LCD_RS
4	PC11 (4.4)	SW4	13	PF8(1.20)	LCD_D4
5	PB5 (4.27)	KEY1	14	PF9 (1.21)	LCD_D5
6	PB6 (4.28)	KEY2	15	PF10 (1.22)	LCD_D6
7	PB7 (4.29)	KEY3	16	PF11 (2.13)	LCD_D7
8	PB8 (4.31)	KEY4	17	VDD (2.36)	VCC3.3V
9	VCC5V (1.36)	VCC5V	18	GND (1.35)	GND

實驗步驟

- 範例目錄架構
- 範例說明
- 預設定義說明

範例目錄架構

- 範例目錄
  - 測試映像檔
  - 含括檔
  - 函式庫
  - 專案檔
  - 原始碼



範例說明

Embedded Software Side

Flash memory
R/W operation

- **FLASH Unlock**
- **FLASH ClearFlag**
- **Example**

FLASH memory R/W operation

```
lcd_clear();
lcd_print ("MIAT_STM32 DEMO ");

/* Unlock the Flash Program Erase controller */
FLASH_Unlock();


/* Clear All pending flags */
FLASH_ClearFlag(FLASH_FLAG_BSY | FLASH_FLAG_EOP | FLASH_FLAG_PGERR |
FLASH_FLAG_WRPRTERR);

while(1)
{

 KEY_Buffer=Key_Scan();
 set_cursor (0, 1);
 lcd_print ("SW Value = 0x");
 lcd_putchar(((SW&0x7)+0x30));
 if(KEY_Buffer==1)
 {
  Address=(StartAddr|(SW<<12));
  set_cursor (0, 0);
  lcd_print ("Set Addr = 0x");
  lcd_putchar(((SW&0x7)+0x30));
  lcd_print ("  ");
 }
```

掃描KEY是否按下與顯示SW數值，LCD Line2顯示
SW Value = 0x?(SW)

如果KEY1按下，記錄SW數值至Address ，LCD Line1顯示
Set Adde = 0x?(Address)

範例說明

Embedded Software Side

SRAM memory
R/W operation

Example

FLASH memory R/W operation

```
else if(KEY_Buffer==2)
   {
    /* Erase the FLASH pages */
    FLASHStatus = FLASH_ErasePage(Address);

    /*  FLASH Word program of data at addresses defined by SW*/
    FLASHStatus = FLASH_ProgramWord(Address, (SW&0x7));

    set_cursor (0, 0);
    lcd_print ("Write 0x");
    lcd_putchar(((SW&0x7)+0x30));
    lcd_print (" at 0x");
    lcd_putchar((((Address>>12)&0x7)+0x30));
   }
   else if(KEY_Buffer==3)
   {
    Data=(u16 *)Address;
    set_cursor (0, 0);
    lcd_print ("Addr 0x");
    lcd_putchar((((Address>>12)&0x7)+0x30));
    lcd_print (" = 0x");
    lcd_putchar(((*Data&0x7)+0x30));
    lcd_print ("  ");
   }

  }
```

如果KEY2按下， 清除Address所處區塊並寫入SW數值 ，LCD Line1顯示
Write 0x?(SW) at 0x?(Adddress)

如果KEY3按下， 讀取Address紀錄數值 ，LCD Line1顯示
Addr 0x?(Adddress) = 0x?

預設定義說明

- #define StartAddr ((u32)0x08020000)
  - 定義Flash使用起始點
  - 使用起始點必需大於0x8003000 + Code Size
    - 附註: 0x8000000~0x8003000為DFU程式區塊，使用此區塊將造成無法燒錄程式
    - Code Size可由產生的HEX檔得知
- u32 Address = 0x08020000;
  - 定義Address初始值
  - 初始值必需與Flash使用起始點相同
- Address=(StartAddr|(SW<<12));
  - Example: StartAddr = 0x08020000 SW = 1 Address = 0x08021000
    StartAddr = 0x08020000 SW = 7 Address = 0x08027000
- StartAddr|(7<<12)必需小於0x8040000

使用者界面與Flash存取控制實驗

實驗二

WU-YANG

Technology Co., Ltd.

實驗二練習

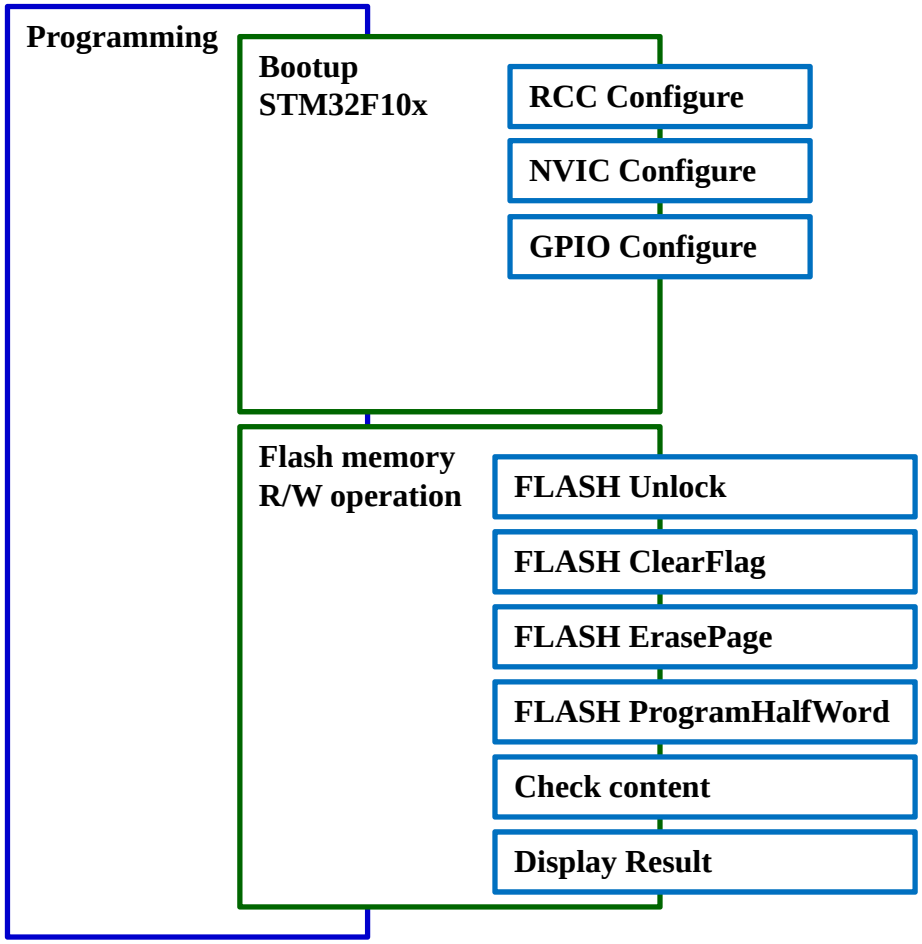
- 注意:
  - 請使用預設0x08020000之後的位置，避免覆蓋DFU與使用者程式碼區塊
- 練習:
  - 利用SW與KEY測試Flash寫入與讀取是否正常
  - Flash寫入後電源關閉，再開啟後寫入資料是否仍存在
  - 修改存取位置測試是否正常

實驗目的(三)

- 使用MIAT_STM32實驗板透過Flash memory interface (FLITF)控制內部Flash進行存取控制實驗，並藉由填入Option Bytes 設定寫入鎖定，保護Page鎖定不被覆蓋。

Development Flow

Embedded Software Side



Bootup STM32F10x

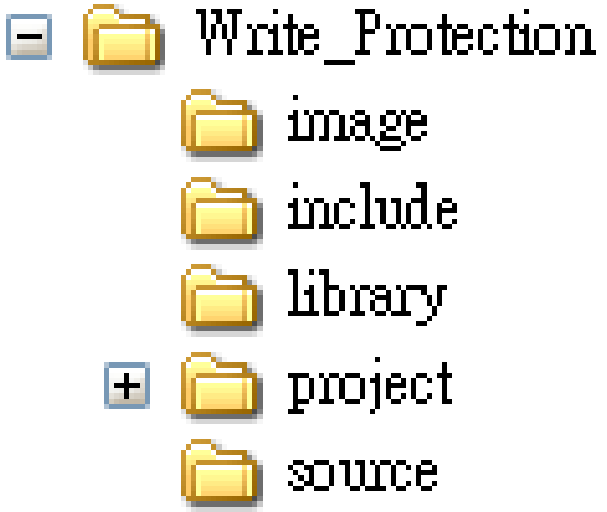
```
int main(void)
{
#ifdef DEBUG
  debug();
#endif

  FLASHStatus = FLASH_COMPLETE;
  MemoryProgramStatus = PASSED;
  Data = 0x1753;

/* RCC Configuration */
  RCC_Configuration();

  /* NVIC Configuration */
  NVIC_Configuration();

  /* GPIO Configuration */
  GPIO_Configuration();


.....................................................
```

實驗步驟

- 範例目錄架構
- 範例說明
- 預設定義說明

範例目錄架構

- 範例目錄
  - 測試映像檔
  - 含括檔
  - 函式庫
  - 專案檔
  - 原始碼



範例說明

Flash FwLib Functions List

Function name	Description
FLASH_Unlock	Unlocks the FLASH Program Erase Controller.
FLASH_ClearFlag	Clears the FLASH pending flags.
FLASH_EraseOptionBytes	Erases the FLASH option bytes.
FLASH_EnableWriteProtection	Write protects the desired pages
NVIC_GenerateSystemReset	Generate a system reset.

```
  /* Unlock the Flash Program Erase controller */
  FLASH_Unlock();

  /* Define the number of page to be erased */
  NbrOfPage = (EndAddr - StartAddr) / FLASH_PAGE_SIZE;

  FLASH_ClearFlag(FLASH_FLAG_BSY | FLASH_FLAG_EOP|FLASH_FLAG_PGERR
|FLASH_FLAG_WRPRTERR);

  /* Get pages write protection status */
  WRPR_Value = FLASH_GetWriteProtectionOptionByte();
  ProtectedPages = WRPR_Value & 0x000000C0;

#ifdef WriteProtection_Disable
  if (ProtectedPages == 0x00)
  {/* Pages are write protected */
    /* Disable the write protection */
    FLASHStatus = FLASH_EraseOptionBytes();
    /* Generate System Reset to load the new option byte values */
    NVIC_GenerateSystemReset();
  }
#else
 #ifdef WriteProtection_Enable
  if (ProtectedPages != 0x00)
  {/* Pages not write protected */
    FLASHStatus = FLASH_EraseOptionBytes();
    /* Enable the pages write protection */
    FLASHStatus = FLASH_EnableWriteProtection(FLASH_WRProt_Pages12to13
|FLASH_WRProt_Pages14to15);
    /* Generate System Reset to load the new option byte values */
    NVIC_GenerateSystemReset();
  }
 #endif
#endif
```

範例說明

Flash FwLib Functions List

Function name	Description
FLASH_ErasePage	Erases a specified FLASH page.
FLASH_ProgramHalfWord	Programs a half word at a specified address.

```
 if (ProtectedPages != 0x00)
  {
    /* Clear All pending flags */
    FLASH_ClearFlag(FLASH_FLAG_BSY |
FLASH_FLAG_EOP|FLASH_FLAG_PGERR |FLASH_FLAG_WRPRTERR);

    /* erase the FLASH pages */
    for(EraseCounter = 0; (EraseCounter < NbrOfPage) && (FLASHStatus ==
FLASH_COMPLETE); EraseCounter++)
    {
      FLASHStatus = FLASH_ErasePage(StartAddr + (FLASH_PAGE_SIZE *
EraseCounter));
    }

    /* FLASH Half Word program of data 0x1753 at addresses defined by  StartAddr
and EndAddr */
    Address = StartAddr;

    while((Address < EndAddr) && (FLASHStatus == FLASH_COMPLETE))
    {
      FLASHStatus = FLASH_ProgramHalfWord(Address, Data);
      Address = Address + 2;
    }
```

範例說明

Embedded Software Side

Flash memory
R/W operation

Check content

Display Result

Display Result

```
/* Check the corectness of written data */
  Address = StartAddr;

  while((Address < EndAddr) && (MemoryProgramStatus != FAILED))
  {
   if((*(vu16*) Address) != Data)
   {
    MemoryProgramStatus = FAILED;
   }
   Address += 2;
  }

 while (1)
 {
  if (MemoryProgramStatus == PASSED)
  {  /* OK Turn on USERLED */
   GPIO_SetBits(GPIOF, GPIO_Pin_11);
  }
  else
  { /* KO Turn off USERLED */
   GPIO_ResetBits(GPIOF, GPIO_Pin_11);
   /* Insert delay */
   Delay(0xAFFFF);
   /* Turn on USERLED */
   GPIO_SetBits(GPIOF, GPIO_Pin_11);
   /* Insert delay */
   Delay(0xAFFFF);
  }
 }
```

如果寫入與讀取Flash內容相同，
Flash記憶體可以寫入，
USERLED紅燈恆亮

如果寫入與讀取Flash內容不同，
Flash記憶體不能寫入，
USERLED紅燈閃爍

預設定義說明

- #define StartAddr ((u32) 0x08006000)
  - 定義Flash使用起始點
  - 使用起始點必需大於0x8003000 + Code Size
    - 附註: 0x8000000~0x8003000為DFU程式區塊，使用此區塊將造成無法燒錄程式
    - Code Size可由產生的HEX檔得知
- #define EndAddr   ((u32) 0x08008000)
  - 定義Flash使用結束點
  - 使用起始點必需小於0x8040000
- Data = 0x1753;
  - 寫入資料
  - 16Bit
- #define FLASH_PAGE_SIZE   ((u16)0x800)
  - 每一個Page有2KByte

預設定義說明

- #define WriteProtection_Enable
  - Uncomment this line to Enable Write Protection
- #define WriteProtection_Disable
  - Uncomment this line to Disable Write Protection
- FLASH_EnableWriteProtection
  - 設定影響2個Page
    - FLASH_WRProt_Pages0to1
    - FLASH_WRProt_Pages1to2 ……
    - FLASH_WRProt_Pages60to61
  - 設定影響多個Page
    - FLASH_WRProt_Pages62to255

FLASH_EnableWriteProtection

- Example:

位置0x08006000 ~ 0x08007000

每一個Page有0x800Byte

StartPage = (0x08006000-0x08000000)/0x800 = 12

EndPage = ((0x08007000-0x08000000)/0x800)-1 = 13

執行FLASH_EnableWriteProtection(FLASH_WRProt_Pages12to13)

將影響Pages12、 Pages13

內部Flash寫入鎖定存取控制實驗

實驗三

WU-YANG

Technology Co., Ltd.

實驗三練習

- 注意:
  - 請使用預設0x08006000之後的位置，避免覆蓋DFU與使用者程式碼區塊
  - 更改寫入位置測試寫入鎖定時需同時更改FLASH_EnableWriteProtection鎖定之Page
- 練習:
  - 打開#define WriteProtection_Enable測試是否可寫入
  - 打開#define WriteProtection_Disable測試是否可寫入
  - 修改存取位置測試是否正常
  - 修改寫入資料測試是否正常

Q & A

WU-YANG

Technology Co., Ltd.