Lucerne University of Applied Sciences and Arts

HOCHSCHULE LUZERN

Technik & Architektur

What would be your answer?

interrupts are disabled after CPSID
nach MRS R0 sind sie noch nicht disabled!

Hello Erich
i have a question regarding the "fixed" EnterCritical() code.

```
#define CpuCriticalVar() uint8_t cpuSR
#define CpuEnterCritical() \
do { \
asm ( \
"MRS R0, PRIMASK\n\t" \
"CPSID I\n\t" \
"STRB R0, %[output]" \
.....
```

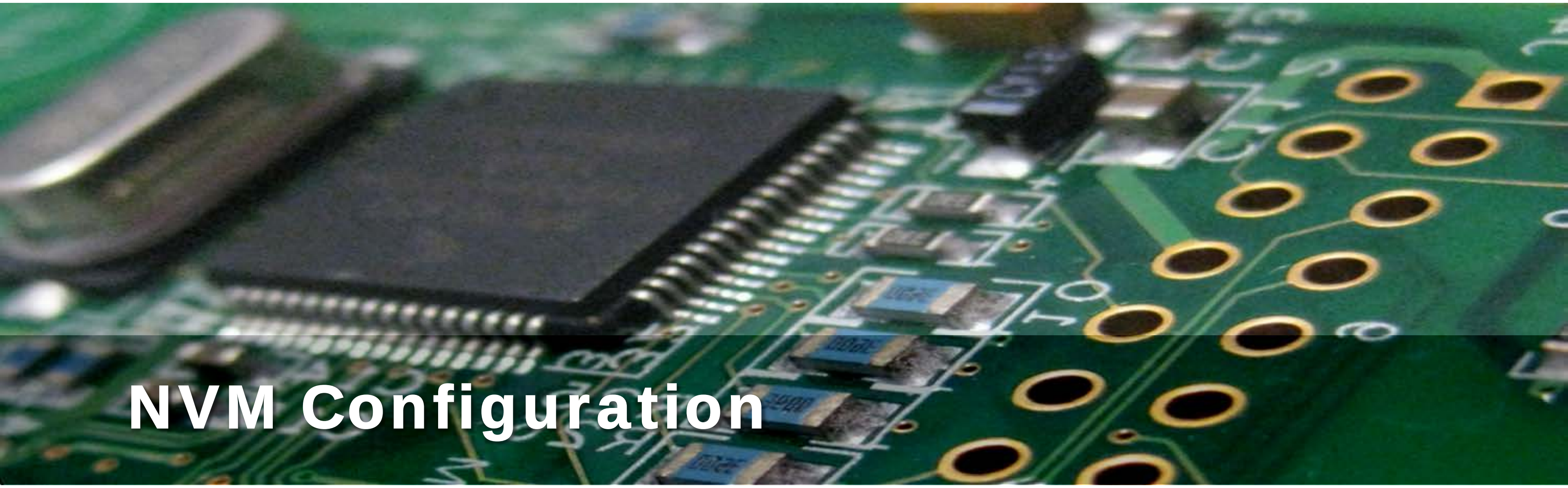
Movie the Primask in to R0 (special kind of move) , save the interrupt status in R0

control processor status interrupt disable (disable interrupts, set the I bit, set the master bit)

store the R0 in the local variable

Imagine the following scenario:
EnterCritical() gets interrupted after storing PRIMASK into R0,
then the ISR modifies R0 and exits (nothing prevents R0 from being modified, or am i wrong?),
then EnterCritical() continues with disabling Interrupts and storing R0 (now with a wrong value) into cpuSR.
Could this scenario not disable all my interrupts and stop my program from working correctly?
...
Best regards,
Dorian

Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN
Technik & Architektur



NVM Configuration

“My data is in RA…. Ahhhrg! Who turned off the power?!?”

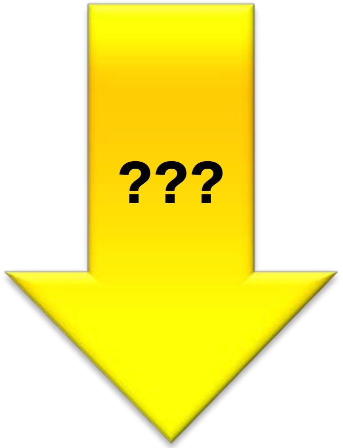
Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN
Technik & Architektur

Learning Goals

- Goal: Storing persistent configuration Data

- Data
    - Calibration
    - Setup
- NVM Options
    - Disk/SD
    - EEPROM
    - FLASH
- Internal Flash
- Implementation Options



Configuration Data

- Data different from system to system
- Sensor calibration values, device ID, application configuration, ...
- RAM: lost after power up
- Need to store it in Non-Volatile-Memory (NVM)
- Possible solutions
  - Custom build Image
  - Store Data in NVM

```
#ifdef USE_FLASH_CONFIG
  static const uint16_t calibValues[3] =
     {0x1247, 0x5579, 0x59AE};
#else /* at runtime */
  static uint16_t calibValues[3];
#endif
```

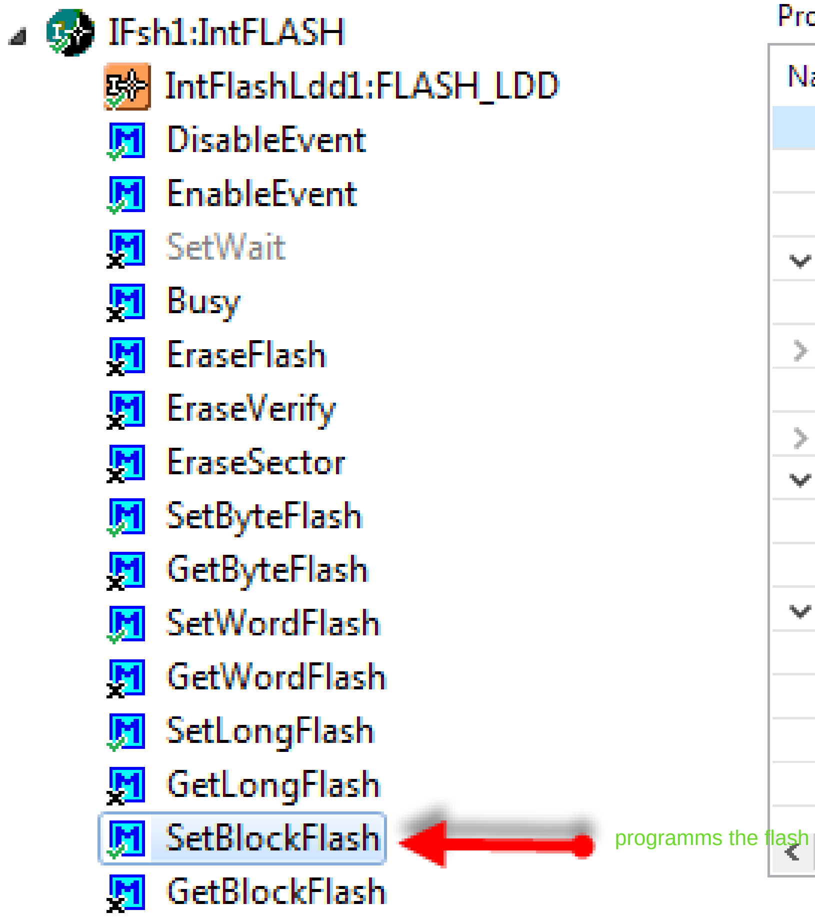
NVM Options

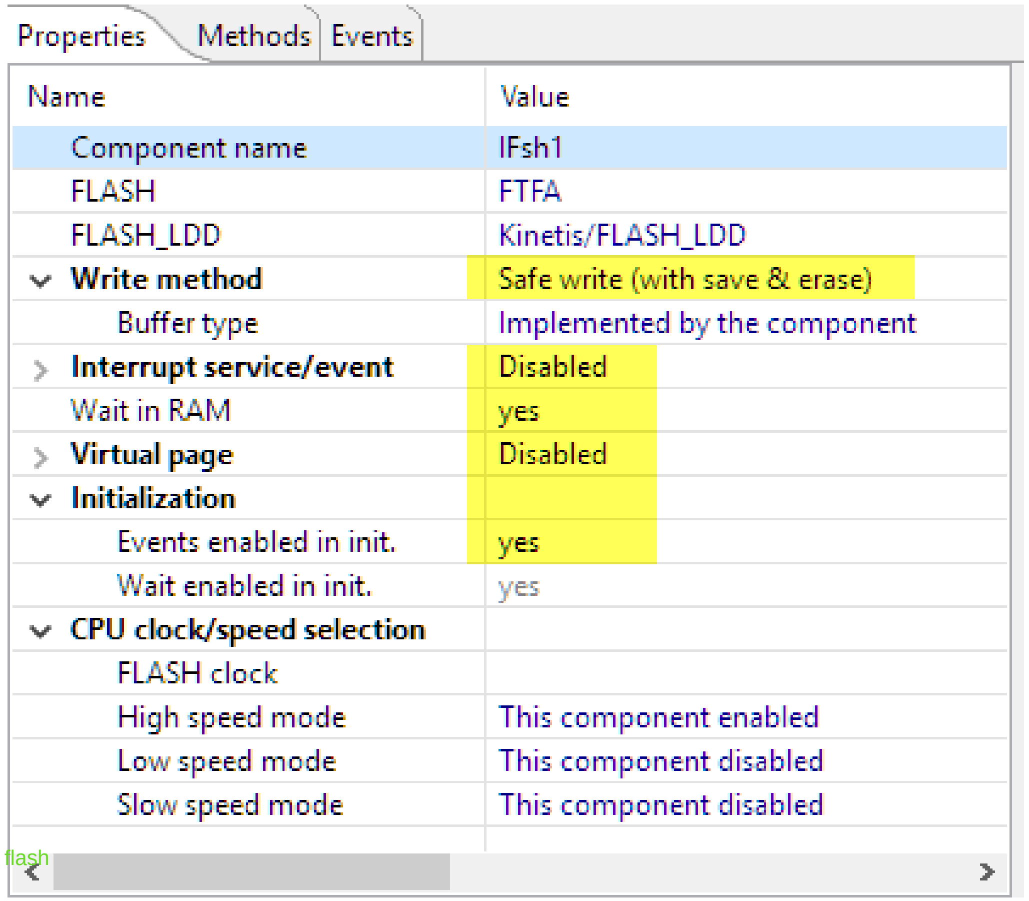
- Battery Buffered SRAM
  - E.g Maxim DS3232, combined with RTC
- SD, disk
  - Raw Block access
  - File System: data exchange with host
  - Consider overhead
  - SD: industrial or not
- External EEPROM/Flash IC
  - SPI, I2C
  - Erase/Program Cycles: ~100k-500k
  - Example: Microchip 24AA
- Internal Microprocessor EEPROM/FLASH
  - No external components
  - Flash Programming Algorithms
  - Erase/Program Cycles: ~50k

Internal Flash Programming

- Part of program flash is reserved for 'reprogramming' by the application
- Flash is Block oriented (1, 2, 4, 8, ... kByte)
    - Erase whole block, reprogram block
    - Erase: bring bits to 1 (0xFF)
- Need 'app'/function to reprogram the flash
    - Optional: Save block content
    - Erase block
    - Program block with new content
- Typically
    - Flash bus is blocked ➔ need to run in RAM!
    - Interrupts disabled

IntFLASH Component

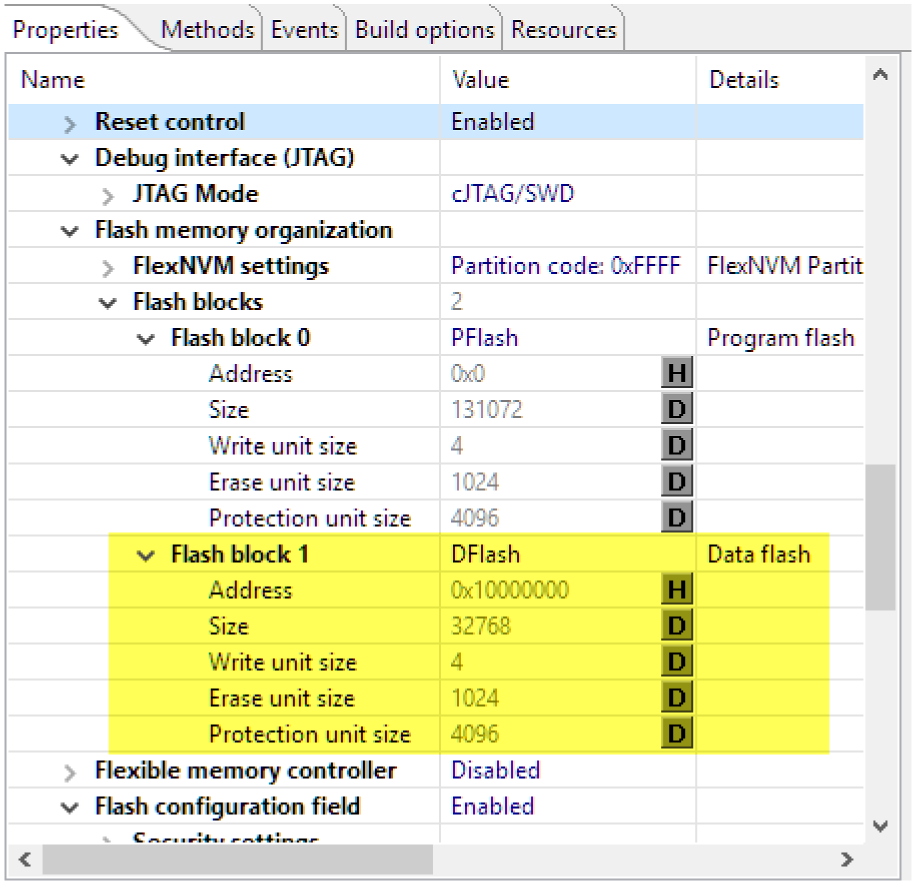




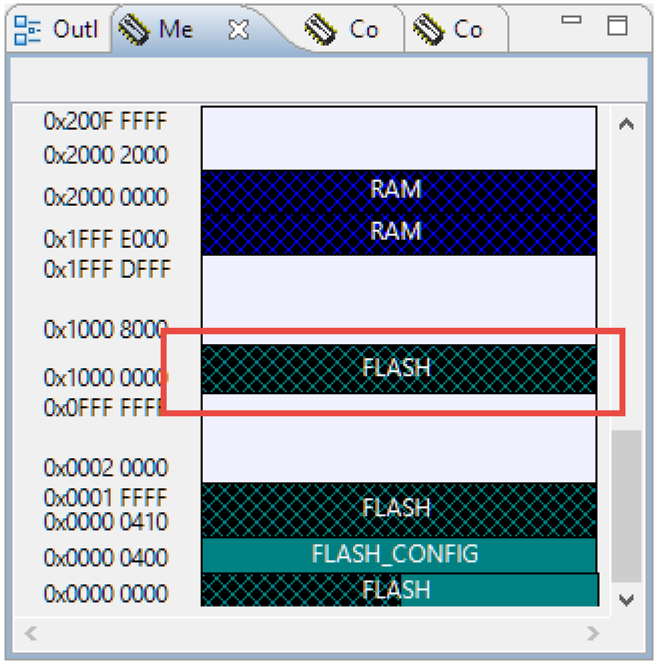
Name	Value
Component name	IFsh1
FLASH	FTFA
FLASH_LDD	Kinetis/FLASH_LDD
Write method	Safe write (with save & erase)
Buffer type	Implemented by the component
Interrupt service/event	Disabled
Wait in RAM	yes
Virtual page	Disabled
Initialization	
Events enabled in init.	yes
Wait enabled in init.	yes
CPU clock/speed selection	
FLASH clock	
High speed mode	This component enabled
Low speed mode	This component disabled
Slow speed mode	This component disabled

K20 (Remote)

- Enable Data Flash @ 0x1000’0000



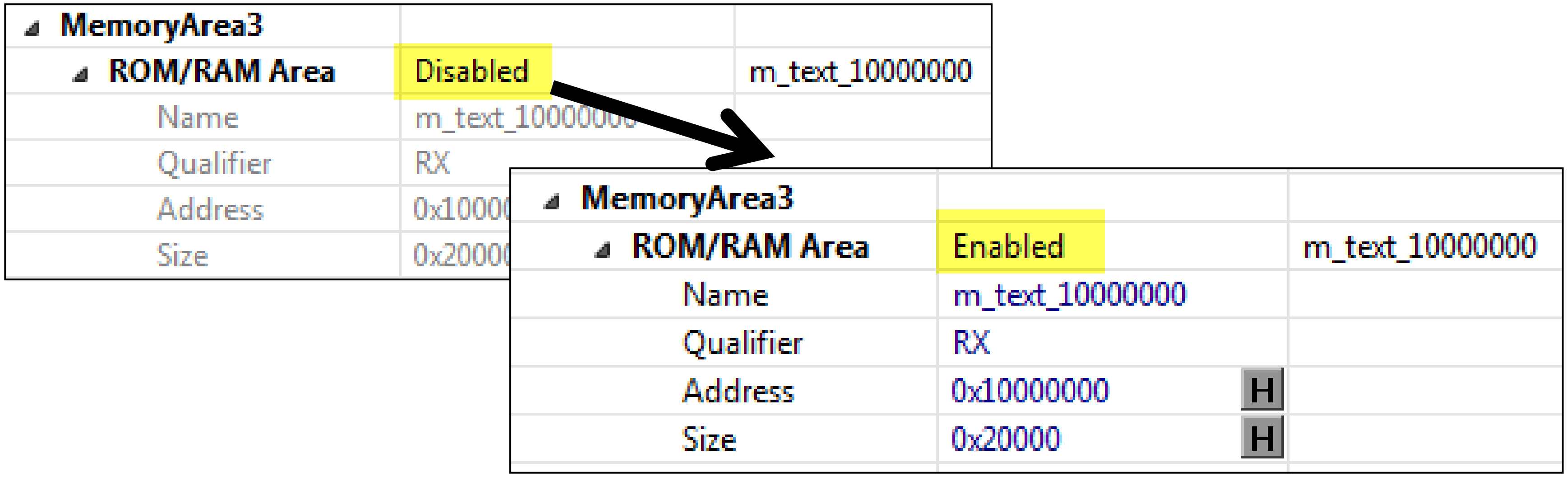
Name	Value		Details
Reset control	Enabled		
Debug interface (JTAG)			
JTAG Mode	cJTAG/SWD		
Flash memory organization			
FlexNVM settings	Partition code: 0xFFFF		FlexNVM Partit
Flash blocks	2		
Flash block 0	PFlash		Program flash
Address	0x0	H	
Size	131072	D	
Write unit size	4	D	
Erase unit size	1024	D	
Protection unit size	4096	D	
Flash block 1	DFlash		Data flash
Address	0x10000000	H	
Size	32768	D	
Write unit size	4	D	
Erase unit size	1024	D	
Protection unit size	4096	D	
Flexible memory controller	Disabled		
Flash configuration field	Enabled		



Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN
Technik & Architektur

CPU Build Options (K22FX512)

- Enable Flex Memory area (0x1000'0000)
- Block size 0x1000! unit of 4kB
- Alignment!



MemoryArea3		
ROM/RAM Area	Disabled	m_text_10000000
Name	m_text_10000000	
Qualifier	RX	
Address	0x10000	
Size	0x2000	

MemoryArea3		
ROM/RAM Area	Enabled	m_text_10000000
Name	m_text_10000000	
Qualifier	RX	
Address	0x10000000 H	
Size	0x20000 H	

Implementation Options

- Struct in Flash at fixed address
  - Visible in debugger
  - Can provide default values at compile time
  - Compiler cares about alignment
  - Dependency to other modules
  - Need to make sure it is properly allocated by Linker

- Blocks in Flash
  - Start Address + Size
  - Raw flash blocks
  - Using absolute addresses
  - Programmer needs to care about alignment
  - Simple dependency to other modules

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

Constant Struct

```
/* NVM_Config.h */
typedef struct {
  …
  int16_t pressureOffsetMbar;     /* pressure data offset */
  uint8_t pressureSamplingFreqHz; /* pressure measurement frequency */
  …
} NVMC_DataType;

extern const NVMC_DataType NVMC_Data;

#define NVMC_GetPressureOffsetMbar()            (NVMC_Data.pressureOffsetMbar)

uint8_t NVMC_SavePressureOffsetMBar(int16_t offset);
```

```
/* NVM_Config.c */
const NVMC_DataType NVMC_Data =
{
…
    800,            /* mbar offset */
    10,             /* pressure frequency */
…
};
```

Blocks: Address and Size

```
#if PL_IS_FRDM
  #define NVMC_FLASH_START_ADDR    0x1FC00
#elif PL_IS_ROBO
  #define NVMC_FLASH_START_ADDR    0x10000000
#else
  #error "unknown target?"
#endif
```

```
#define NVMC_REFLECTANCE_DATA_START_ADDR  \
   (NVMC_FLASH_START_ADDR)
#define NVMC_REFLECTANCE_DATA_SIZE        \
   (6*2*2) /* 6 sensors (min and max) 16bit each */
        min  max
#define NVMC_REFLECTANCE_END_ADDR        \
   (NVMC_REFLECTANCE_DATA_START_ADDR+NVMC_REFLECTANCE_DATA_SIZE)
```

Example Usage

- Consider your own/different Interface

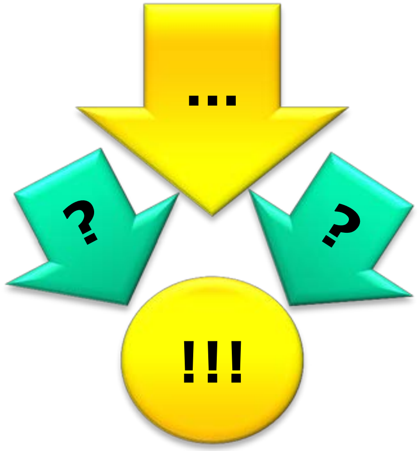
```
uint8_t NVMC_SaveReflectanceData(void *data, uint16_t dataSize) {
  if (dataSize>NVMC_REFLECTANCE_DATA_SIZE) {
    return ERR_OVERFLOW;
  }
  return IFsh1_SetBlockFlash(data,
    (IFsh1_TAddress)(NVMC_REFLECTANCE_DATA_START_ADDR), dataSize);
}

void *NVMC_GetReflectanceData(void) {
  if (isErased((uint8_t*)NVMC_REFLECTANCE_DATA_START_ADDR,
         NVMC_REFLECTANCE_DATA_SIZE)
      )
  {
    return NULL;
  }
  return (void*)NVMC_REFLECTANCE_DATA_START_ADDR;
}
```

if erased -> everything is FFFFF..

Summary

- Needs for NVM (for configuration data)
- Many options
    - Battery buffered SRAM
    - D/File System
    - Internal or external
    - Erase/Program Cycles
- Flash Applet
    - Run in RAM
    - Blocks
    - Constant structs
    - Constant Memory Pointers



Lab: NVM Config

- Add NVM Configuration Module
- Save your configuration data
  - Reflectance array
  - Sensor calibration data
  - Any other application data
- Note:
  - Detect if flash is erased or not

- NOTE: Segger J-Link does not a full FLASH erase! It only erases blocks defined by application!