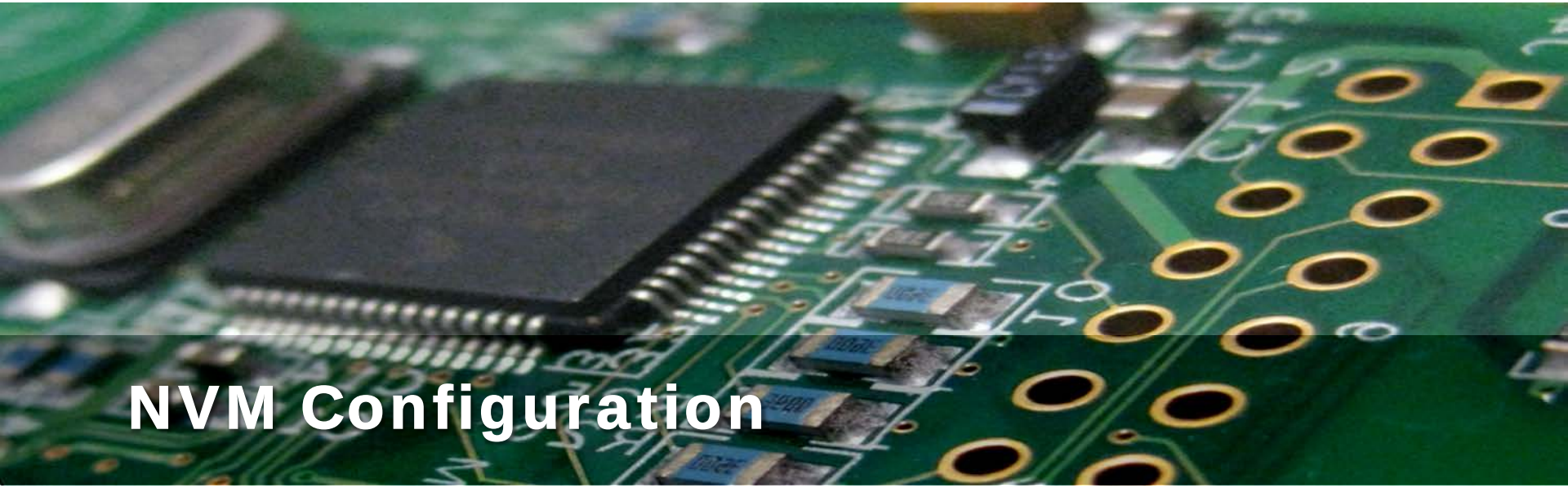
Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur



NVM Configuration

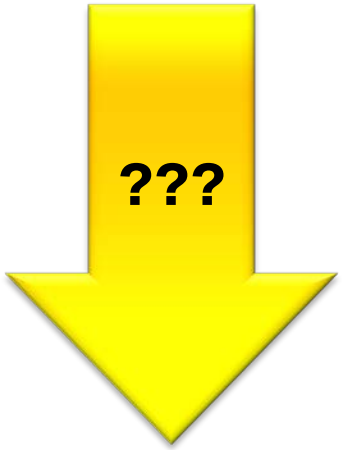
“My data is in RA…. Ahhhrg! Who turned off the power?!?”

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

Learning Goals

- Goal: Storing persistent configuration Data

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



Configuration Data

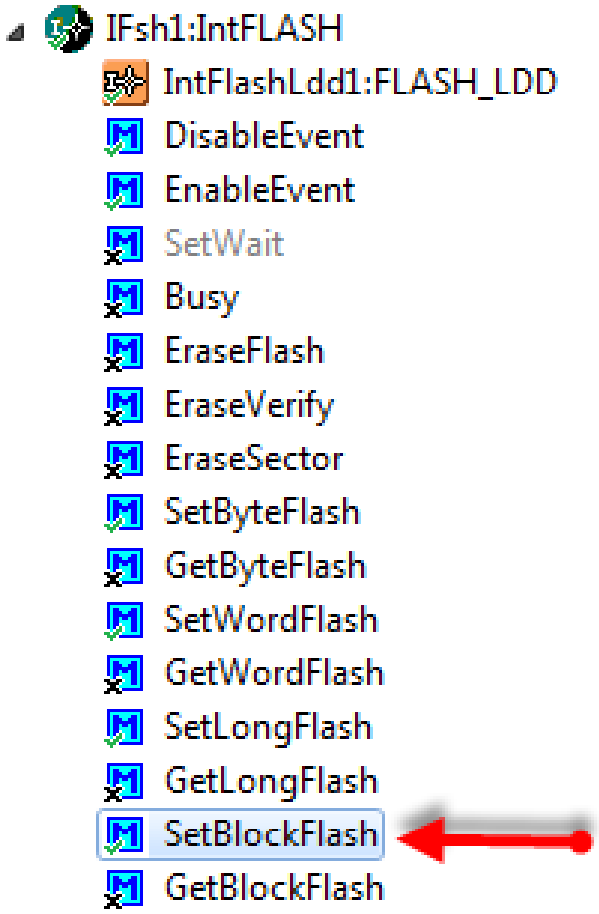
- Data different from system to system
- Sensor calibration values, device ID, application configuration, …
- RAM: lost after power up
- Need to store it in Non-Volatile-Memory (NVM)
- Possible solutions
  - Custom build Image
  - Store Data in NVM

```
#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

Flash Programming with Processor Expert

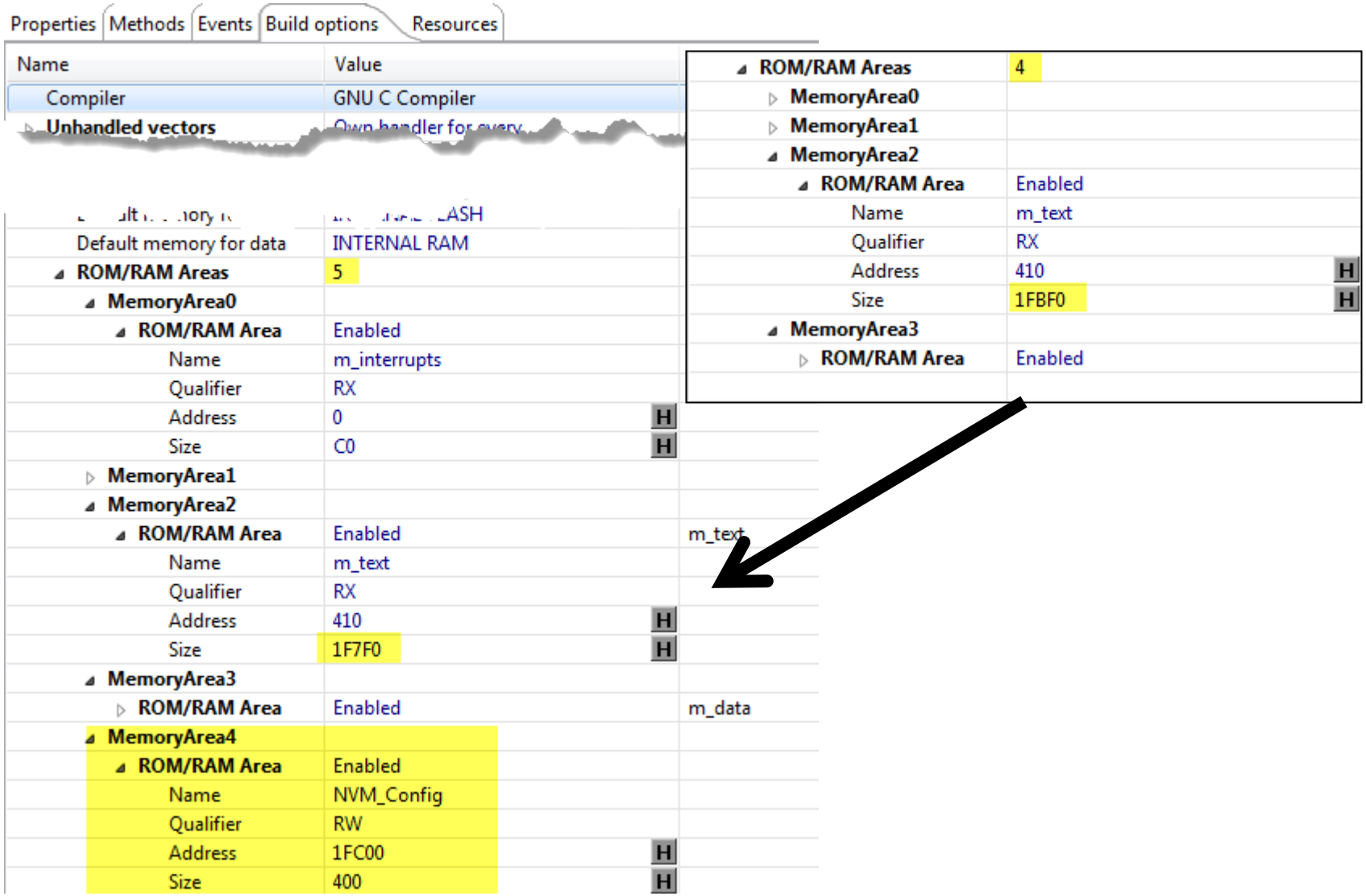


Component name	IFsh1
FLASH	FTFA
FLASH_LDD	FLASH_LDD
Write method	Safe write (with save & erase)
Buffer type	Implemented by the component
Interrupt service/event	Enabled
Command complete interru	
Interrupt	INT_FTFA
Interrupt priority	medium priority
Wait in RAM	yes
Virtual page	Disabled
Initialization	
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

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

CPU Component Memory Map



Implementation Options

- Struct in Flash at fixed address
  - Visible in debugger
  - Can provide default values at compile time
  - Compiler cares about aligment
  - 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

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_SRB
  #define NVMC_FLASH_START_ADDR    0xFBB0
#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 */
#define NVMC_REFLECTANCE_END_ADDR         \
   (NVMC_REFLECTANCE_DATA_START_ADDR+NVMC_REFLECTANCE_DATA_SIZE)
```

Example Usage

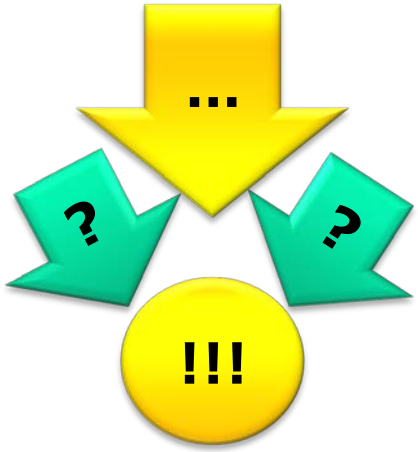
- Consider your own/different Interface

```
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;
}
```

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



Tasks

- 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