



M³Real-Time-Systems SS 2017

Prof. Dr. rer. nat. Karsten Weronek Faculty 2
Computer Science and Engineering

Microcontroller

Our Microcontroller in the Exercises



In the exercises we will configure microcontroller.

We use the STR9 by STM.

This microcontroller is based on ARM966-E core at 96MHz.

It is old, inexpensive but has a bunch of components.

ARM has no production plants, they just sell licenses for their design.

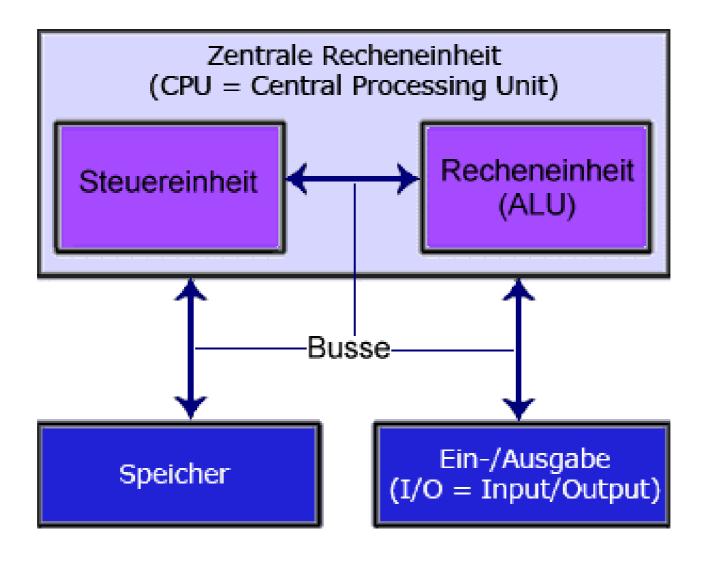
Our chip was produced by ST Microelectronics.

This chip is not based on a "von Neumann Architecture" but on an "Havard Architecture": different Memory for Data and Code

The Havard Architecture is faster!

Von Neumann Architecture

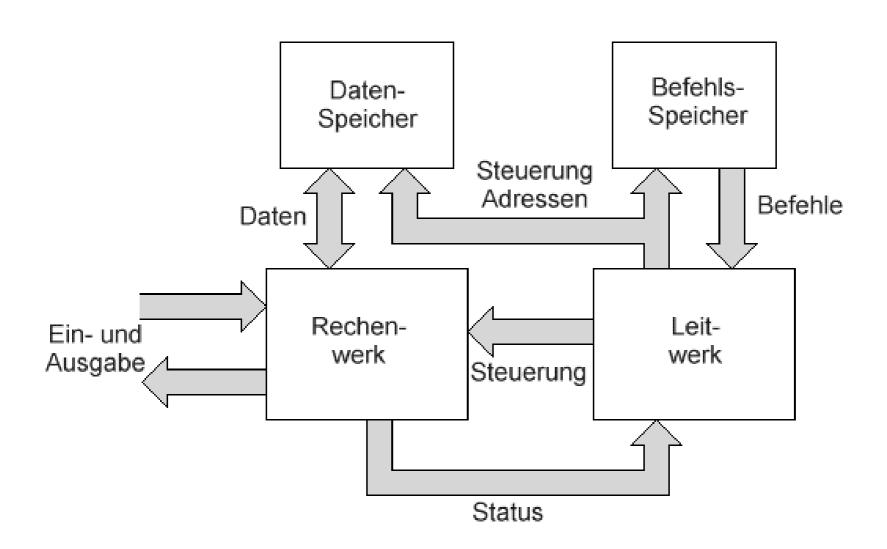




ref.: http://www.netzmafia.de/skripten/mikrocomputer/kap1.html

Harvard Architecture





ref.: http://www.netzmafia.de/skripten/mikrocomputer/kap1.html

Features



16/32-bit 96 MHz ARM9E based MCU (Micro-Controller-Unit)

ARM966E-S RISC core:

Harvard architecture

5-stage pipeline

Tightly-Coupled Memories

- Static Random Access Memory (SRAM)
- Flash-Memory (=EEPROM)

Up to 96 MIPS directly from flash memory

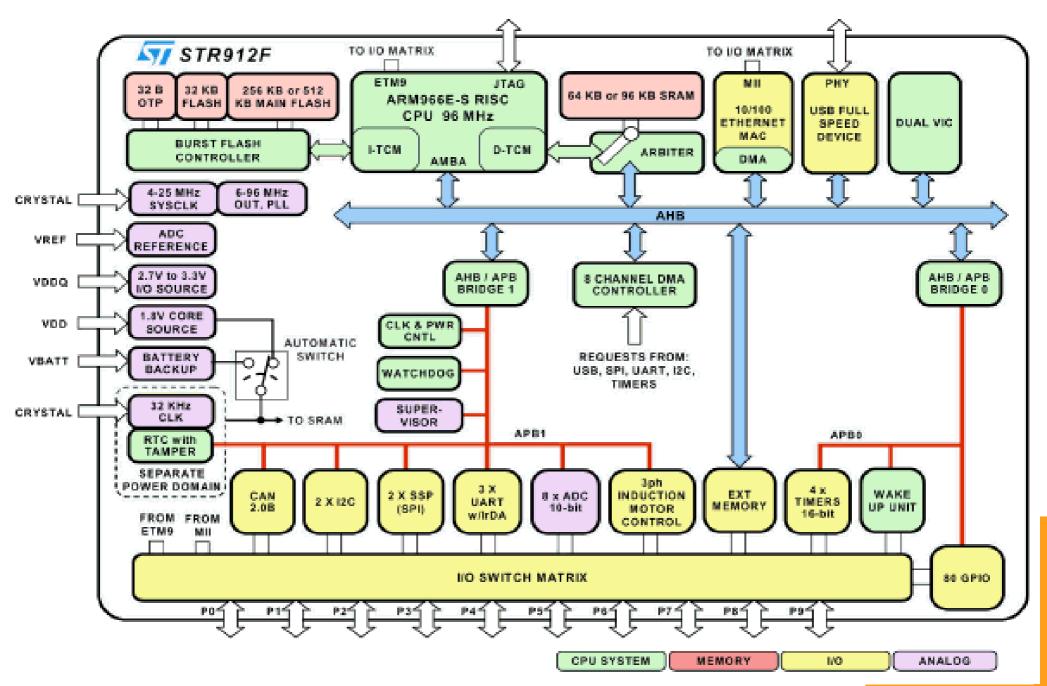
Single-cycle Digital Signal Processor (DSP) instructions supported

Binary compatible with ARM7 code

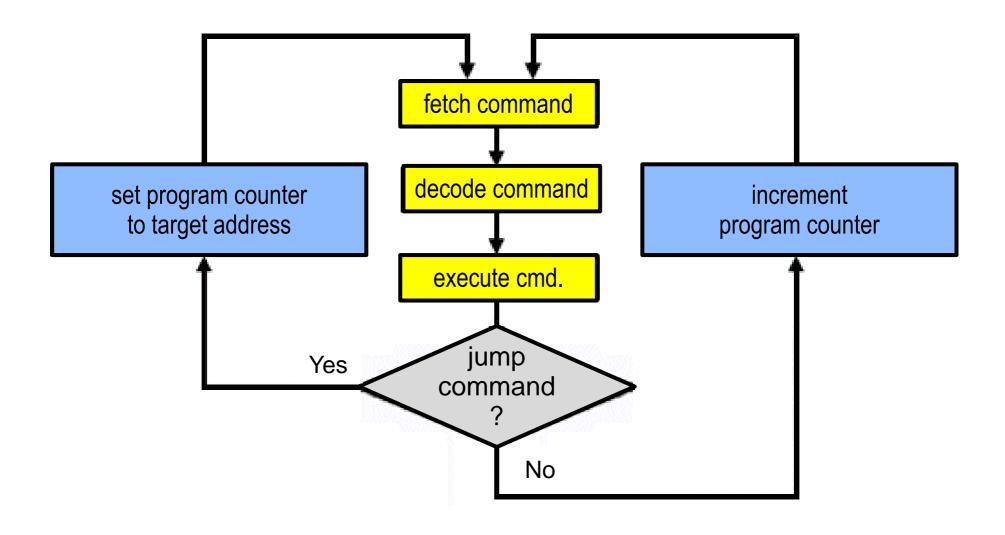
Dual burst Flash memories, 32-bits wide

Achitecture





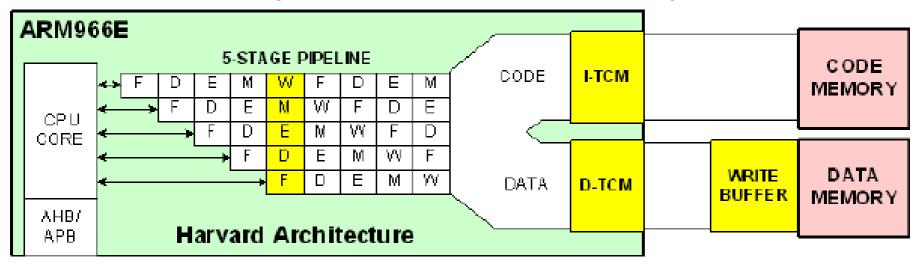


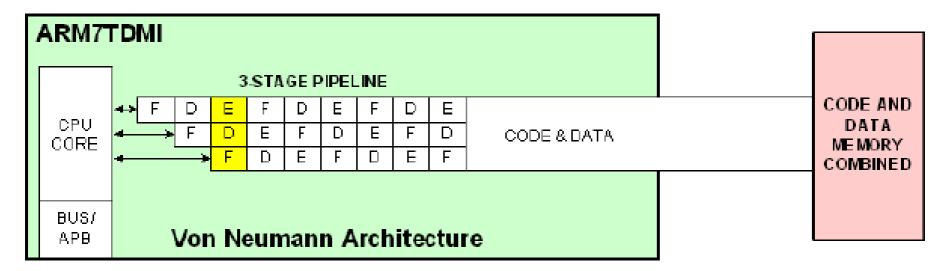


The Pipeline



At the heart of the ARM9 CPU is the instruction pipeline. The pipeline is used to process instructions taken from the program store. On the ARM7 a three-stage pipeline is used.





The ARM9 five-stage pipeline has independent fetch, decode, execute memory (read) and (memory) write stages

Arm STR9 - System Control Unit (SCU)



Peripheral clock gating register 0 (SCU_PCGR0)

Address offset: 14h

Reset value: 0000 00DBh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			MAC	USB 48M	USB	DMA	EXT_MEM _CLK	EMI	VIC	SRAM _ARBITER	SRAM	Res.	PFQ BC	FMI	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		

Peripheral clock gating register 1 (SCU_PCGR1)

Address offset: 18h

Reset value: 0000 0000h

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved								GPIO 8	GPIO 7	GPIO 6	GPIO 5	GPIO 4	GPIO 3	GPIO 2
							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
GPIO 1	GPIO 0	WIU	Res.	ADC	CAN	SSP 1	SSP 0	12C 1	12C 0	UAR T2	UAR T1	UAR T0	МС	TIM2 3	TIM0 1
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Arm STR9 - System Control Unit (SCU)



Peripheral reset register 0 (SCU_PRR0)

Address offset: 1Ch

Reset value: 0000 1053h

15	14	13	12	11	10	9	. 8	7	6	5	4	3	2	1	0
Reserved		RST_ PFQBC_ AHB	RST_ MAC	Res.	RST_ USB	RST_ DMA	Res.	RST_ EMI	RST_ VIC	RST_ SRAM_ ARBITER	Re	es.	RST_ PFQBC	RST_ FMI	
			rw	rw		rw	rw		rw	rw	rw			rw	rw

Peripheral reset register 1 (SCU_PRR1)

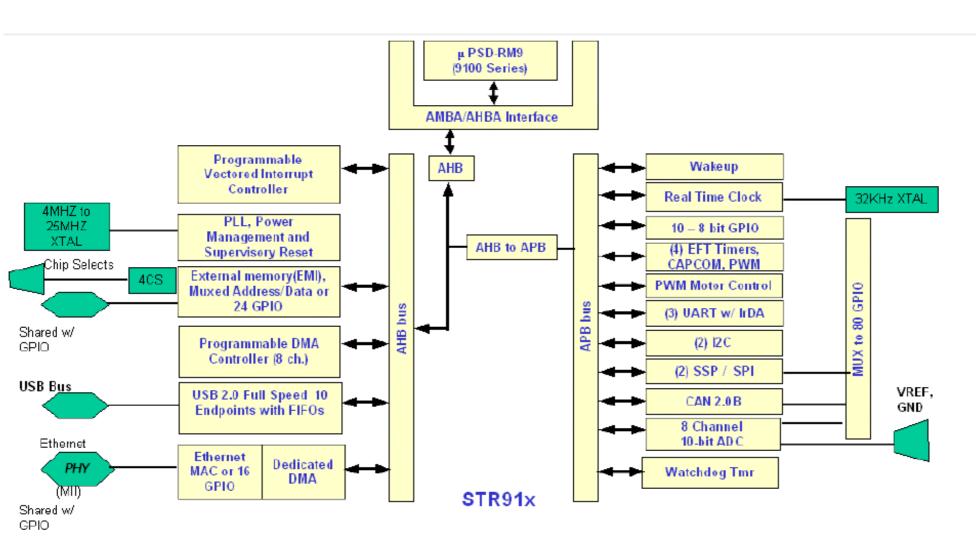
Address offset: 20h

Reset value: 0000 0000h

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RST_GPI09	RST_GPIO8	RST_GPIO7	RST_GPIO6	RST_GPIO5	RST_GPIO4	RST_GPIO3	RST_GPIO2
							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
RST_GPI01	RST_GPIO0	RST_WIU	Reserved	RST_ADC	RST_CAN	RST_SSP1	RST_SSP0	RST_I2C1	RST_I2C0	RST_UART2	RST_UART1	RST_UART0	RST_MC	RST_TIM23	RST_TIM01
rw	rw	rw	•	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

The (bus) architecture of the STR9





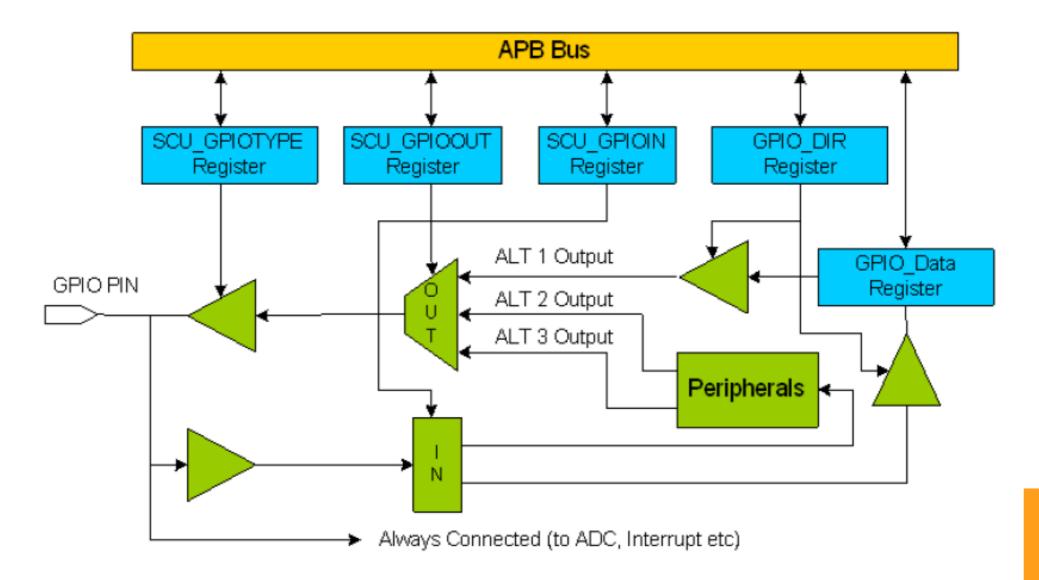
The STR9 has three internal busses. IA high speed bus which connects the CPU to the on-chip memory and complex peripherals the remaining peripherals are connected to two separate peripheral busses.

Arm STR9: General Purpose IO (GPIO)



- GPIO 0 9
- All GPIO pins are 5V tolerant.
- I/O port drivers may be configured as push-pull or as open collector.
- Some peripheral functions have bi-directional functionality such as I2C.
- data and clock lines.
- Only Port 4, bits 0...7 have an ADC input.
- when configured for input mode, all ports are in high impedance.
- Alternate Input functions default to open and on P0 P7 are controlled by SCU GPIOIN control registers.
- Alternate Output functions on P0 P7 are configured via
 SCU GPIOOUT control registers that select from one of 3 output functions.
- The GPIO ports have no internal or programmable pull-up resistors.





Each STR9 GPIO pin may be configured as input, output or two additional alternate outputs can connect the external pin to on chip peripherals

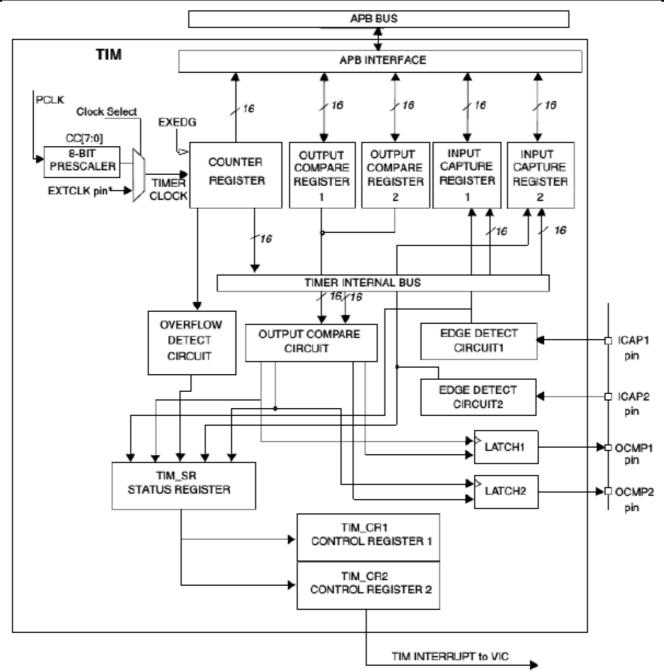
Arm STR9: Timer



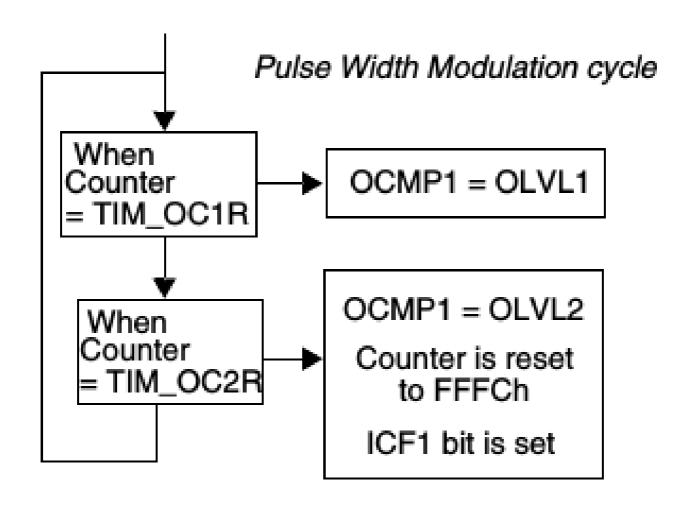
- Programmable prescaler: f_{PCLK} divided by 1 to 256 in steps of 1
- Overflow status flag and maskable interrupt
- External clock inputs with the choice of active edge
- Output compare functions with:
 - 2 dedicated 16-bit registers
 - 2 dedicated programmable signals
 - 2 dedicated status flags
 - 1 maskable interrupt
- Input capture functions with:
 - 2 dedicated 16-bit registers
 - 2 dedicated active edge selection signals
 - 2 dedicated status flags
 - 1 maskable interrupt
- Pulse Width Modulation output mode (PWM)
- One Pulse mode (OPM)
- PWM input mode (PWMI)
- 5 alternate functions
- DMA support

Timer Architecture











Pulse width modulation mode

Pulse Width Modulation (PWM) mode enables the generation of a signal with a frequency and pulse length determined by the value of the TIM_OC1R and TIM_OC2R registers.

The Pulse Width Modulation mode uses the complete Output Compare 1 function plus the TIM_OC2R register.

Procedure

To use pulse width modulation mode select the following in the CR1 register:

- Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with OC1R register.
- Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with OC2R register.
- Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
- Set the PWM bit.
- Select the timer clock (ECKEN) and the prescaler division factor (CC[7:0]).
- Load the OC2R register with the value corresponding to the period of the signal.
- Load the OC1R register with the value corresponding to the length of the pulse if (OLVL1= 0 and OLVL2 = 1).

If OLVL1= 1 and OLVL2 = 0 the length of the pulse is the difference between the OC2R and OC1R registers.

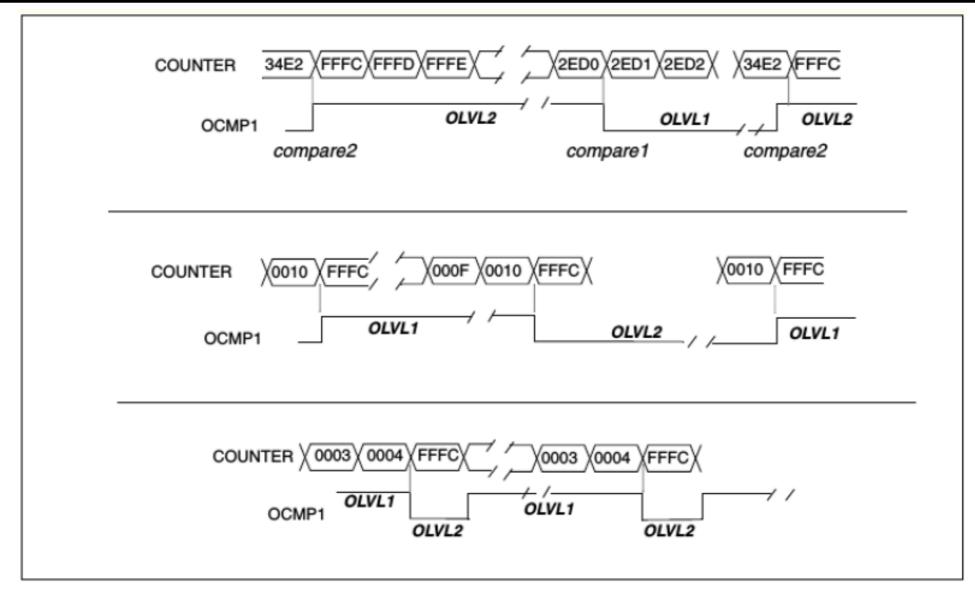
The OCiR register value required for a specific timing application can be calculated using the following formula:

OC/R Value =
$$\frac{t \cdot f_{PCLK}}{t_{PRESC}} - 5$$

Prof. Dr. rer. nat. Karsten Weronek

STR9: PWM: Example





- In the top part of Figure 47, OC1R = 2ED0h, OC2R = 34E2, OLVL1 = 0, OLVL2 = 1.
- 2. In the middle part of Figure 47, OC1R = OC2R = 0010h, OLVL1 = 1, OLVL2 = 0.
- In the bottom part of Figure 47, OC1R = FFFCh, OC2R = 0004h, OLVL1 = 1, OLVL2 = 0.

How to work with STR9



- it is not possible to develop software directly on the microcontroller
- you have to develop the software on a PC
- you have to compile/build the software on a PC
 - therefore you need a so called cross-compiler
- then you have to deploy the executable onto the microcontroller
 - this is called "to flash"

To do this the following tool chain is provided

Tool chain



For the cross compile gcc is used

- the compiler runs on the PC (x86 von Neumann architecture) as the build platform but
- the target platform is a STR9 with ARM7 (Havard) architecture

To build the software we use make

- a makefile is a file that defines in which order all SW-components have to be compiled and have to be linked together
- in the exercises, you just type make in the correct directory where the sources and the make-file is available.

To connect to the chip

 The tool openood is provided to connect to the chip and to enable to flash the using str9flash.pl

Inline Assembler



part of the exercise code:

```
asm(" \
    MRS r0, cpsr \n \
    ORR r0, r0, #0x00000040 \n \
    AND r0, r0, #0xffffffff \n \
    MRR cpsr_c, r0 \n \
");
```



M³Real-Time-Systems SS 2017

Prof. Dr. rer. nat. Karsten Weronek Faculty 2
Computer Science and Engineering

Microcontroller

