ECE 4140
Embedded Systems Design

Lecture

Milestone #1 – Setup, RCC and GPIO

Reading:

STM32L4 Reference Manual Chapter 5 Power

STM32L4 Reference Manual Chapter 6 Reset & clock control (RCC)

STM32L4 Reference Manual Chapter 8 GPIO

STM32L452RE datasheet

Tennessee TECH

Coding conventions

- “Code is read much more often than it is written.”

  -- *Guido van Rossum*

- Team-written code should not be distinguishable from code from one person
  - Your southern accent is cool. Be proud!
  - But code is no place for your “accent” to show through
- Conventions mean less time trying to figure out what someone else is doing.
  - In industry, repeated violations of conventions will get you fired!

Coding conventions

- ECE 4140 will use coding conventions for all code
  - ECE 4140 C language coding conventions
    - Available at the class website
  - Python PEP 8
    - Available at *www.python.org*
- Consider adopting a “code-beautifier” in your tool-flow
  - Astyle, PythonTidy, PyLint, etc.

Conventions are not about stifling your creativity.
They are all about increasing your productivity!

Don’t write *C* code like this

Syntactically correct (& functioning) code is not necessarily good code

```
main(int x, char** b)
  {
  while (*b != b] && (*b = b])
       && (b = 0) || (*++b && (**b
       && ((++b)[*b] &&
       (**b <= x[*b] ||
       (**b += x[*b] -=
       **b = x[*b] - **b))
       && --b|| putchar(**b) &&
       ++*b--) || putchar(10))))
              ;
  }
```

C language variable names

Prefix	***Data Type***	***Prefix***	***Data Type***
b	boolean	st	structure
u8	unsigned 8-bit integer	sz	zero-terminated string
u16	unsigned 16-bit integer	fn	function
u32	unsigned 32-bit integer	*Composite prefix examples*	
i8	signed 8-bit integer	pu16	pointer to an unsigned 16-bit integer
i16	signed 16-bit integer	ai8	array of signed 8-bit integers
i32	signed 32-bit integer	pfn	pointer to a function
f	floating point	pst	pointer to a structure
d	double precision float	psz	pointer to a zero-terminated string
p	pointer	apfn	array of "function pointers'"
a	array	apai8	array of pointers to byte arrays

All C language variables should be named

`prefix_somethingDescriptiveAndUseful`

C language variable names

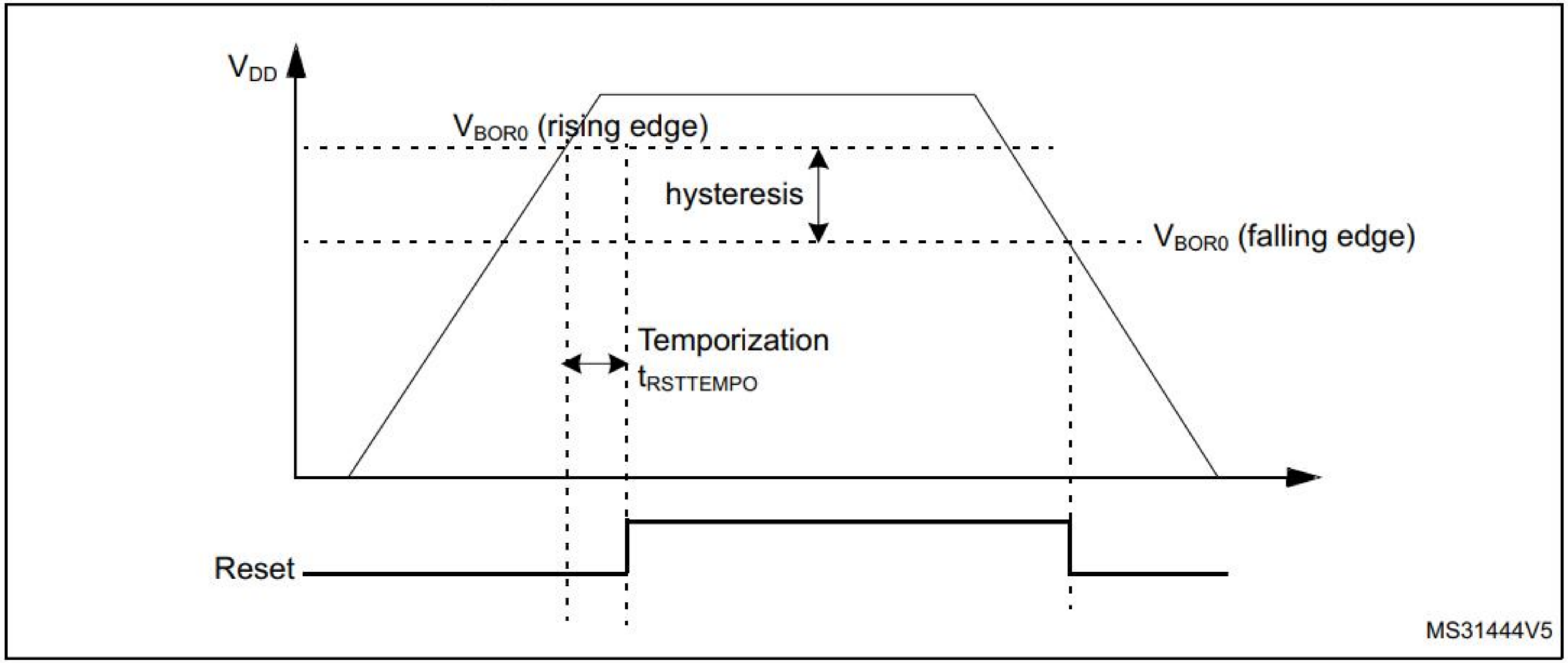
- b_busy
  - boolean, or bit
- u8_numOfApples
  - unsigned 8-bit (byte) count of items
- u32_lightYears
  - unsigned 32-bit value
- i16_temperature
  - signed 16-bit value
- f_priceOfTeaInChina
  - floating-point value
- d_pi
  - double precision floating-point value
- sz_lastName
  - zero-terminated string
- u32_populationWorld
  - unsigned 32-bit integer
- st_timeOfLaunch
  - clock time structure
- fn_fourthRoot
  - function name
- psz_ownerLastName
  - pointer to zero-terminated string
- af_accountBalances
  - array of floating-point values
- pst_bufferDescriptor
  - pointer to a structure
- apst_activeTasks
  - array of pointers of structures

All variables shall be fixed-width data types (in `stdint.h` since C99)

`uint8_t, int16_t, uint32_t, int32_t,` etc.

Milestone 1

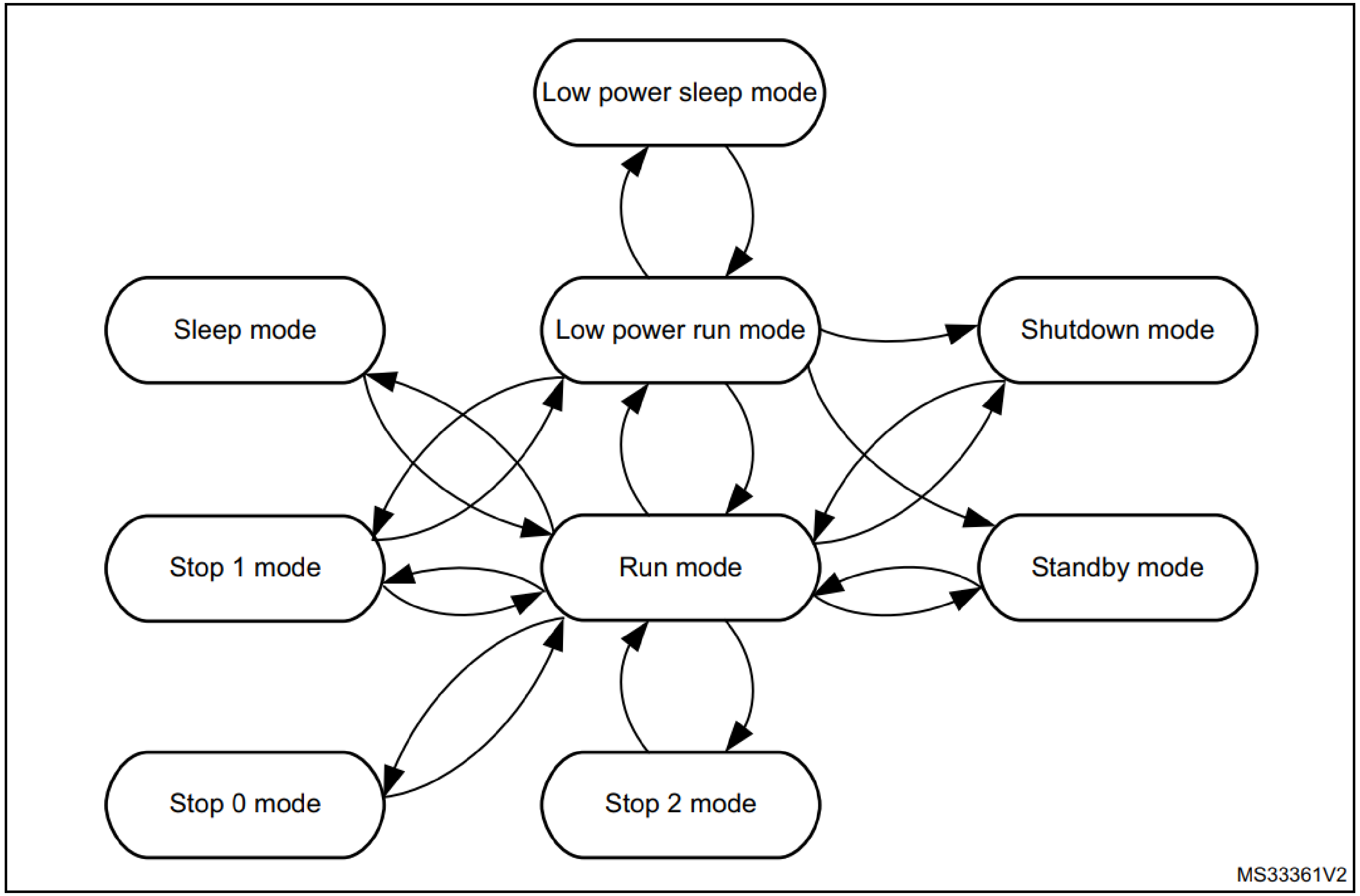
Figure 9. Brown-out reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR lowest threshold (V_{BOR0}).

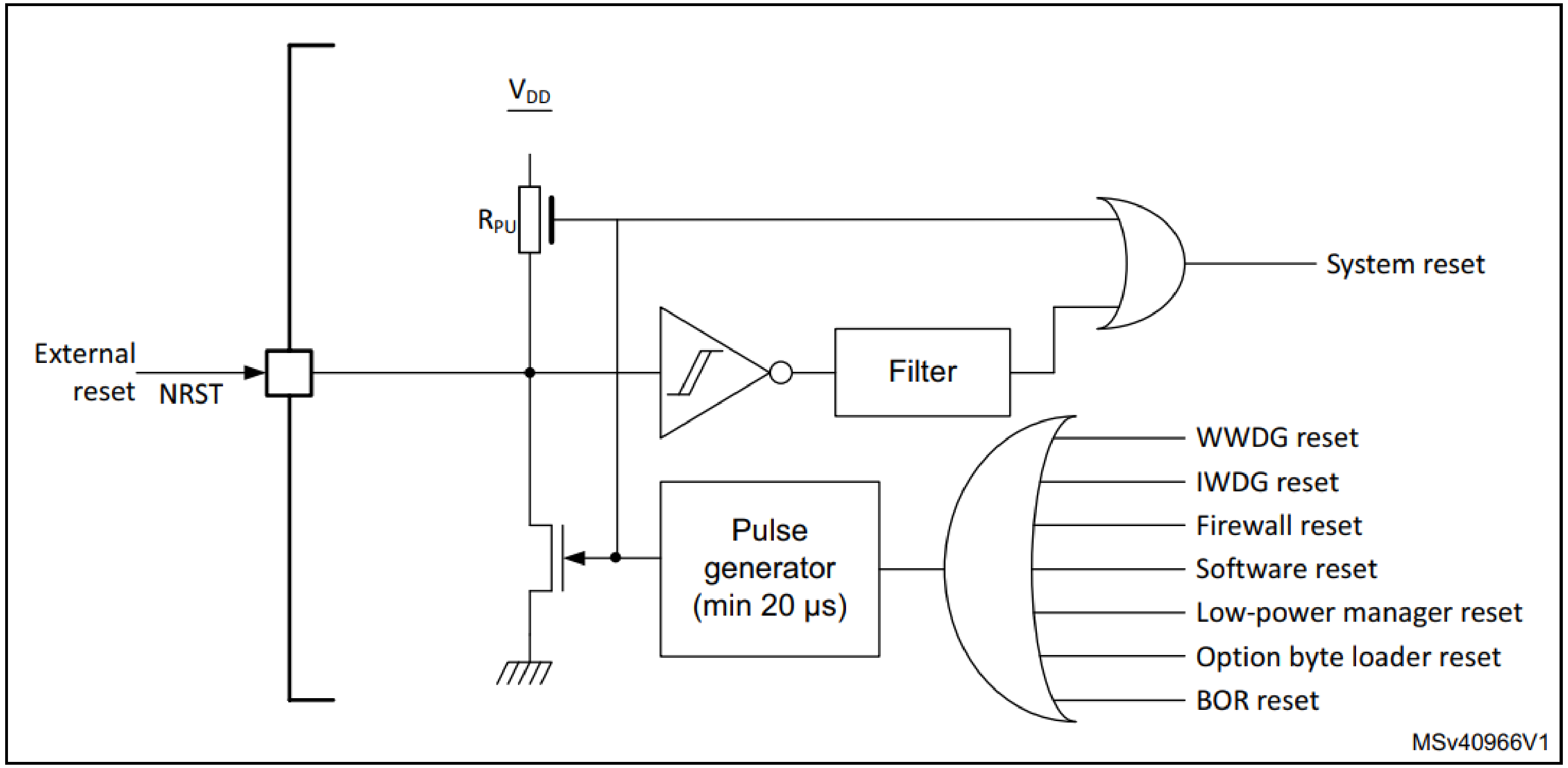
Milestone 1

Figure 11. Low-power modes possible transitions



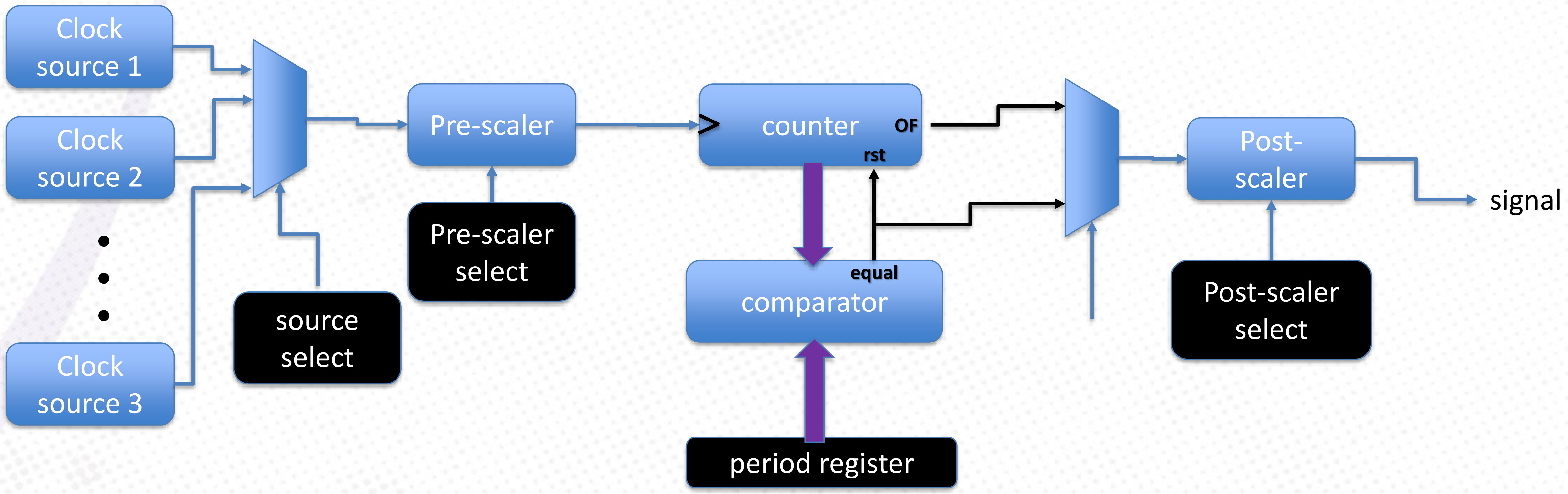
Milestone 1

Figure 12. Simplified diagram of the reset circuit



Review of Timing Signal Creation

- Stylized block diagram of creating arbitrary timing signals in uProcessors



- Pre-scalers and post-scalers can be binary dividers or integer dividers.
  - post-scalers are most often binary dividers

Timing Signals

- Many uP operations are based on timing signals.
- Therefore, many uP peripherals will have variations of the block diagram on the previous page. They will differ in clock sources, capability, and flexibility depending on their application
  - System clock (SYSTICK) and hardware peripheral clocks (RCC)
  - Timers (TIMx)
  - Watchdog timers (WDG)
  - UARTs and Synchronous communications
  - ADCs/DACs
  - Touch controllers, LCD controllers, etc.

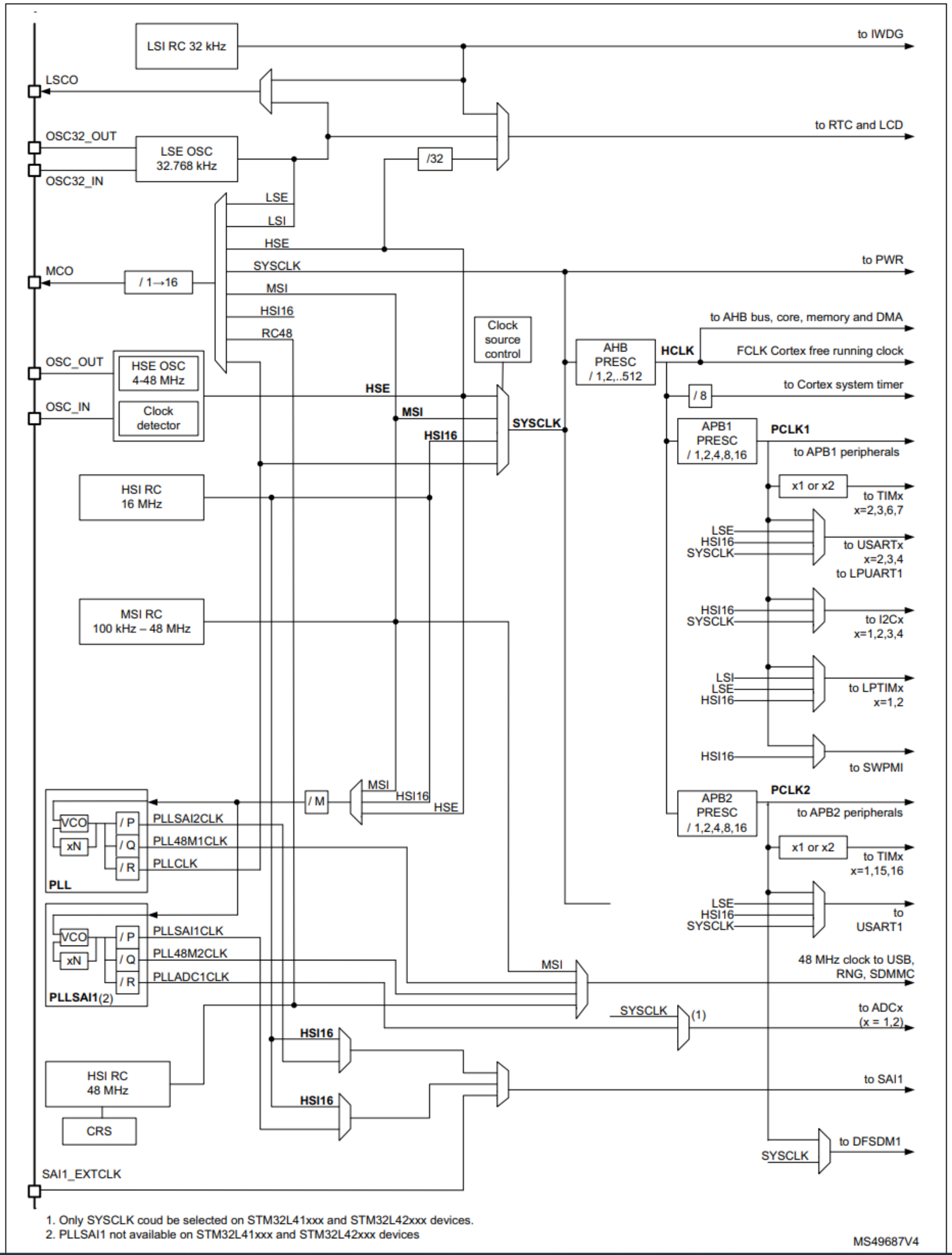
- Give a timing signal creation block diagram, know how to calculate
  - period of the timing signal given the controlling parameters
  - controlling parameters to (nearly) create a desired period
  - percentage error of your created timing signal w.r.t the desired timing signal

STM32L4xx Clocks

- Four different clock sources can be used to drive the system clock (SYSCLK):
  - HSI16 (high speed internal)16 MHz RC oscillator clock
  - MSI (multispeed internal) RC oscillator clock
  - HSE oscillator clock, from 4 to 48 MHz
  - PLL clock
- The devices have the following additional clock sources
  - 32 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes.
  - 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the realtime clock (RTCCLK).
  - RC 48 MHz internal clock sources (HSI48) to potentially drive the USB FS, the SDMMC and the RNG.
- Each clock source can be switched on or off independently when it is not used, to optimize power consumption.
- Several prescalers can be used to configure the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB, the APB1 and the APB2 domains is 80 MHz.

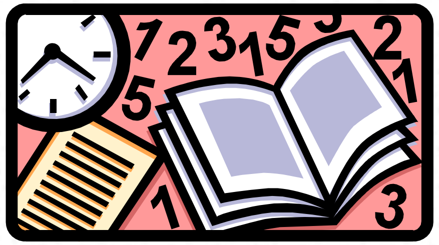
Milestone 1

Figure 13. Clock tree



1. Only SYSCLK coud be selected on STM32L41xxx and STM32L42xxx devices.
2. PLLSAI1 not available on STM32L41xxx and STM32L42xxx devices

Reading Assignment



- STM32L452RE datasheet
- STM32L4 reference manual chapter 8 on GPIO
- Other readings:
  - RB&J, Chapter 8
  - J. Ganssle, *Art of Designing Embedded Systems 2/e*, Chapt. 5
- *www.libopencm3.org*

Milestone 1