STM8L Training Slides

MCD Application

September-October 2009

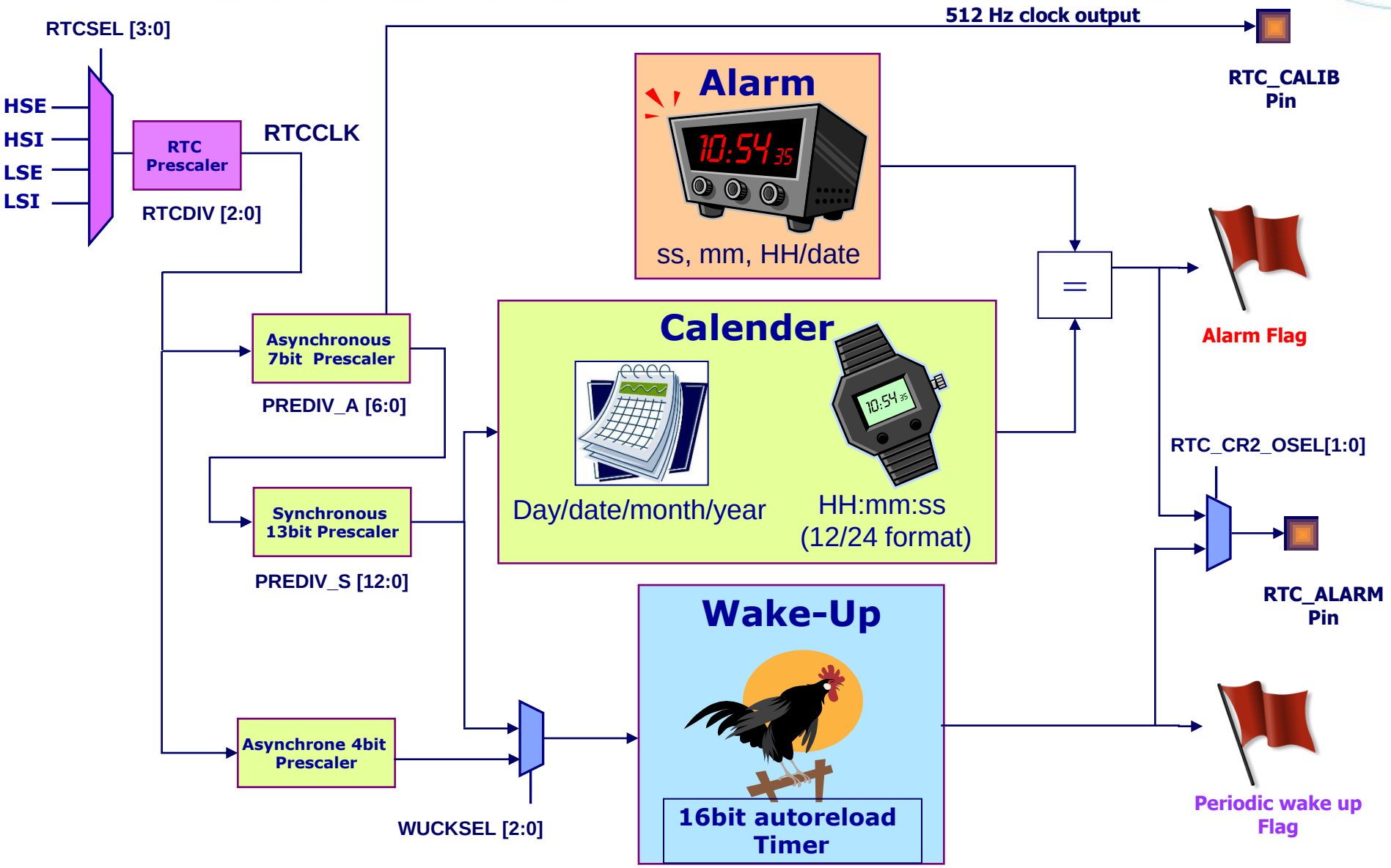
Real Time Clock (RTC)

Available on the STM8L15x
Not on the STM8L101

RTC Features

- **Calendar** with Seconds, Minutes, Hours (12 or 24 format), Day of the week (Day), Day of the Month (Date), Month, and Year coded in BCD format (binary coded decimal) with automatic management for 28, 29 (leap year), 30, and 31 day months.
- **Daylight saving** compensation programmable by software
- **Programmable alarm** with interrupt.
- **Automatic wakeup** interrupt with programmable period to wakeup from Low power modes
- **2 Alternate function outputs:**
  - Output pad providing a 512 Hz clock (with LSE and correct prescaler vallue).
  - Unique pad for both Alarm and Wakeup event.

RTC Block Diagram



RTC registers write protection

- By default and after reset, the RTC registers are write protected to avoid possible parasitic write accesses.

  *** except for the clear of Alarm and Wakeup timer interrupt flags**

- To unlock write protection on all RTC registers
  - 1. Write ‘0xCA’ into the RTC_WPR register
  - 2. Write ‘0x53’ into the RTC_WPR register

➔ **Writing a wrong key reactivates the write protection.**

RTC Clock Sources

- The RTC has two clock sources:
  - ***RTCCLK*** used for RTC timer/counter, can be either the HSE*, LSE, HSI* or LSI clocks.
  - ***SYSCLK*** used for RTC register read/write access.

* When the HSE or HSI clock is selected as RTCCLK source, this clock must be divided to have a maximum of 1 MHz as input for the RTCCLK.

- Before to start using the RTC you have to program the clock controller :
  - Configure the **RTCCLK** source and prescaler in the CLK_CRTCR register
  - Enable **SYSCLK** output to the RTC by setting bit2 in the CLK_PCKENR2 register

RTC in Low Power Modes and in Reset

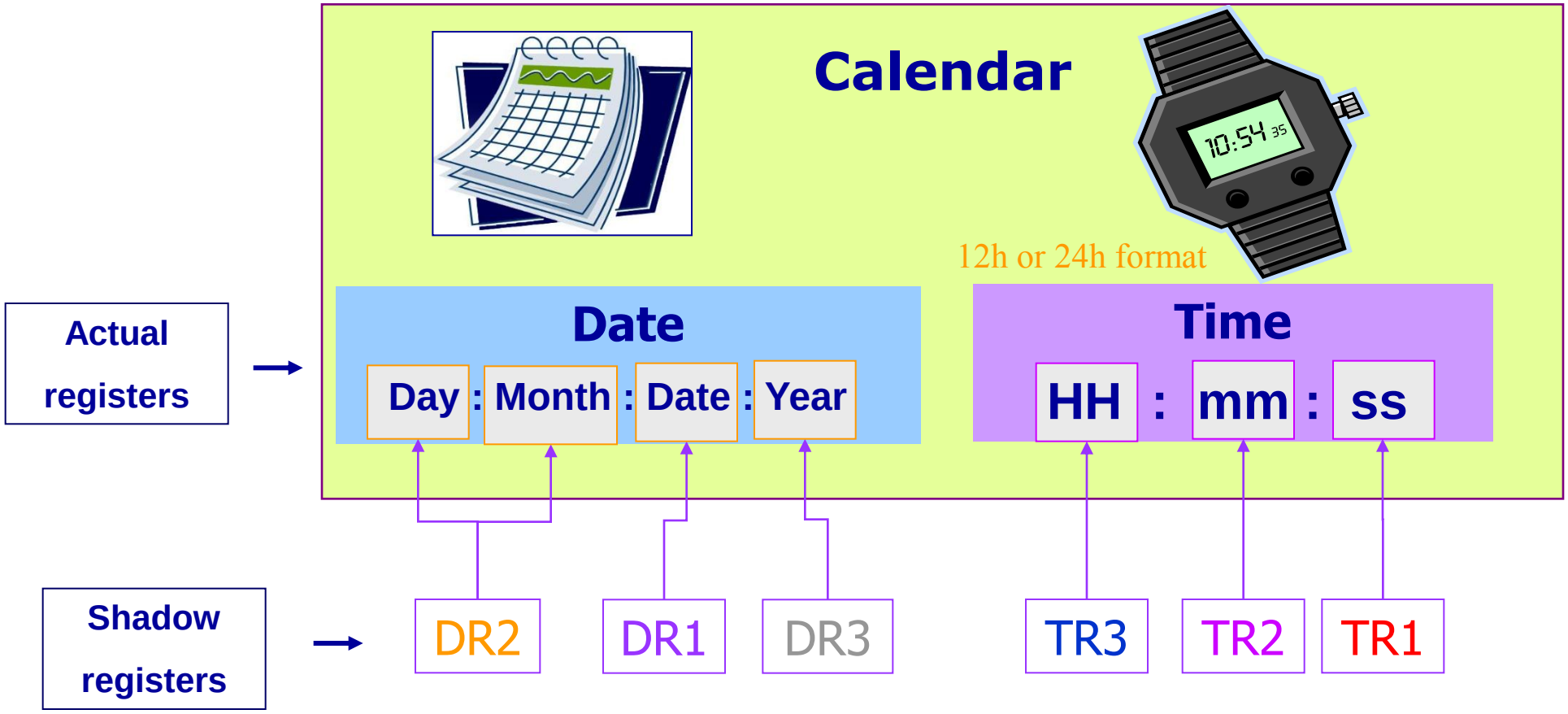
- The RTC remains active what ever is the low power mode
  - Wait, Low Power Wait or Halt (Active Halt)

- When enabled, 2 interrupts can exit the device from low power mode:
  - Alarm A interrupt,
  - Periodic wakeup interrupt

- The RTC remains active under Reset except at Power-on Reset
  - The RTC configuration registers including prescaler programming are not affected by system Reset else than Power-on Reset.
  - The defined clock source is not stopped; the relevant clock controller registers are not affected by system Reset else than Power-on Reset.

RTC Calendar (1/3)

- The initialization or the reading of the calendar value is done through 6 shadow registers, TRx and DRx.



RTC calendar (2/3)

- RTC initialization :
  - **Enter in initialization phase mode setting INIT bit in ISR1 register**
    - This mode is confirmed with the INITF flag also in ISR1 register
  - Program the 3 prescaler registers (APRER and SPRERx) according to the clock source to get 1Hz clock to the calendar.
  - **Load the initial date values in the 6 shadow registers (TRx, DRx).**
    - And other configuration registers like RTC_CR1 (hour format, …)
  - **Exit the initialization phase clearing INIT bit.**
    - The actual calendar register are then automatically loaded and the counting restarts after few RTCCLK clock periods.

- After reset the check of the INITS flag in ISR1 register indicates if the calendar is already initialized (year not at zero) or not (like after Power-on).

- To manage the “daylight saving” there are 3 bits in CR3:
  - SUB1H or ADD1H to subtract or add one hour to the calendar
  - BCK to memorize above action

RTC calendar (3/3)

- The shadow registers are automatically updated each time the RTCCLK clock is synchronized with System Clock.
  - You can check the RSF flag in ISR1 register to make sure that the calendar value from shadow register is the up-to-date one.

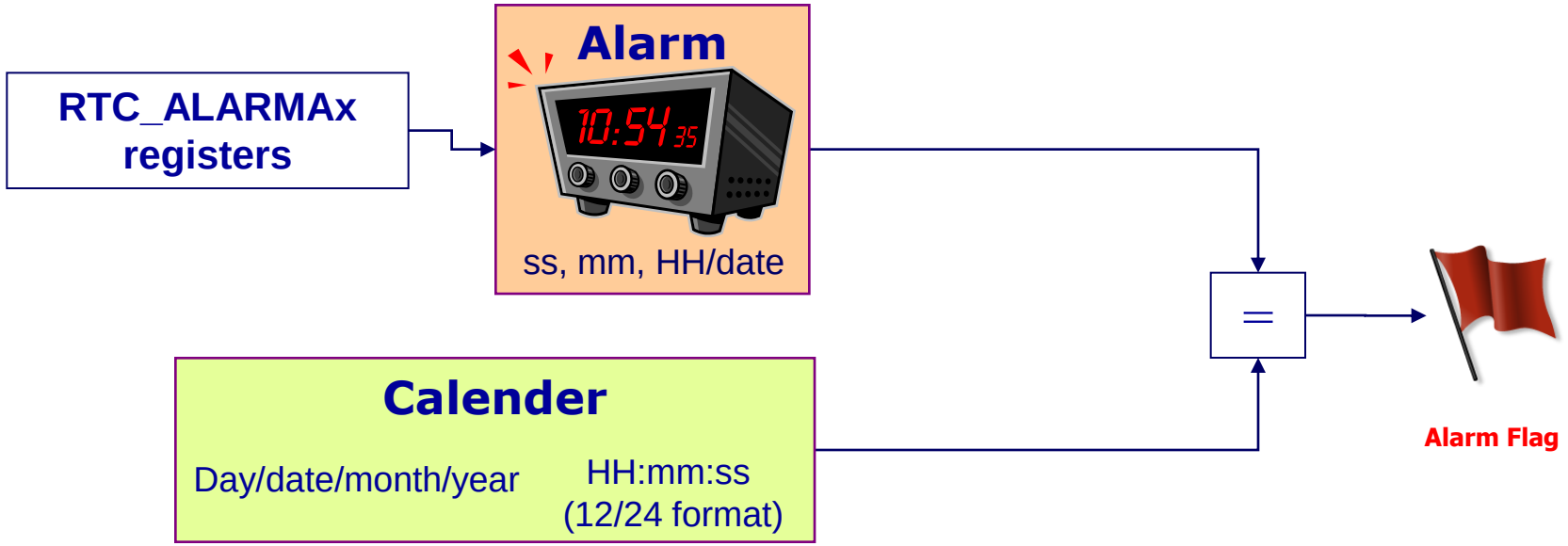
- To ensure the consistency of the calendar fields, update of TR2, TR3, DR1, DR2 and DR3 is frozen **after reading TR1**, and unfrozen when **DR3 is read**.

- In general the system clock frequency must be higher than 2 x RTCCLK frequency to ensure good functionality of the synchronization mechanism.
  - However if user application needs to use the **LS clock as system clock, RTCCLK** has to be **exactly the same clock** (LSE or LSI) and the **RATIO bit in CR1 must be set** to deactivate the synchronization mechanism. In this case RSF flag is always at 1.

RTC Programmable Alarm

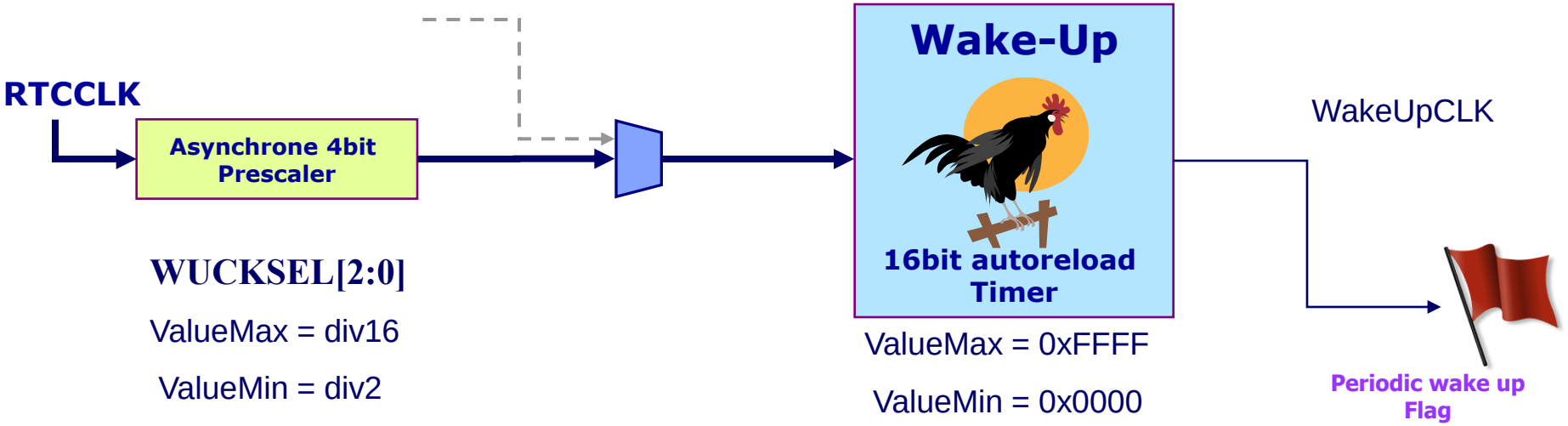
- **Full programmable Alarm**
- Able to **exit** the device from **Active-halt mode**.
- **Alarm event** can also be routed to the **specific output pad** with configurable polarity.

- The **Alarm flag** is set if the calendar seconds, minutes, hours or date match the value programmed in the alarm registers (ALRMARx).
- Calendar seconds, minutes, hours or date fields can be independently selected (maskable or not maskable).



WakeUp configuration (1/3)

- The periodic wakeup flag is generated by a 16-bit programmable binary auto-reload down counter (WUTRH/L registers)
- Able to **exit** the device from **Active-halt mode**.
- The wakeup clock source selection is done via WUCKSEL [2:0] bits in control register RTC_CR1 (to program these bits the auto wakeup must be deactivated, WUTE=0).
- **3 possible cases are possible:**
  - *Case1* **WUCKSEL = 0xx**



WakeUpCLKmax = RTCCLK/(2 x (0x0001 + 1)) => 122µs

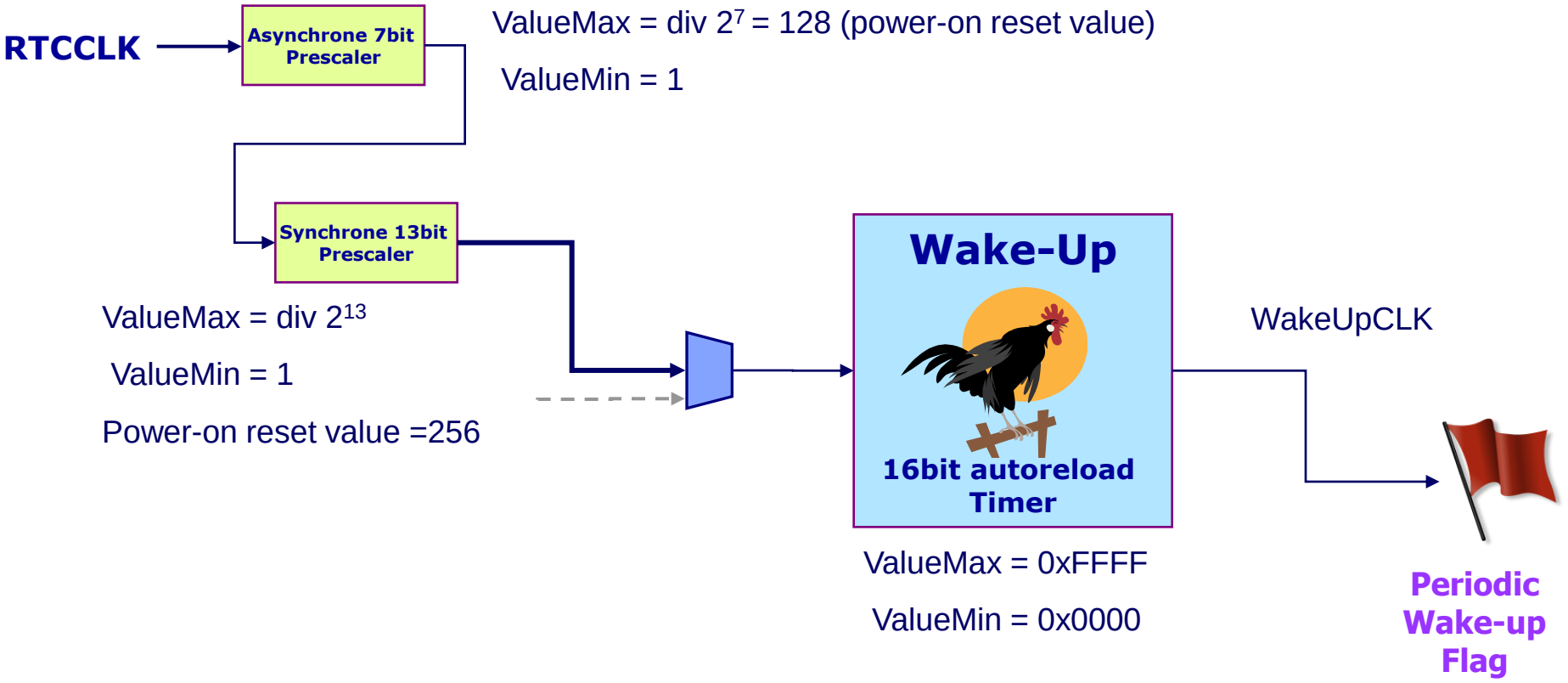
WakeUpCLKmin = RTCCLK/(16 x (0xFFFF + 1)) => 32s

RTCCLK = 32.768KHz

Resolution min=2xRTCCLK=61µs

WakeUp configuration (2/3)

– *Case2* **WUCKSEL = 10x**

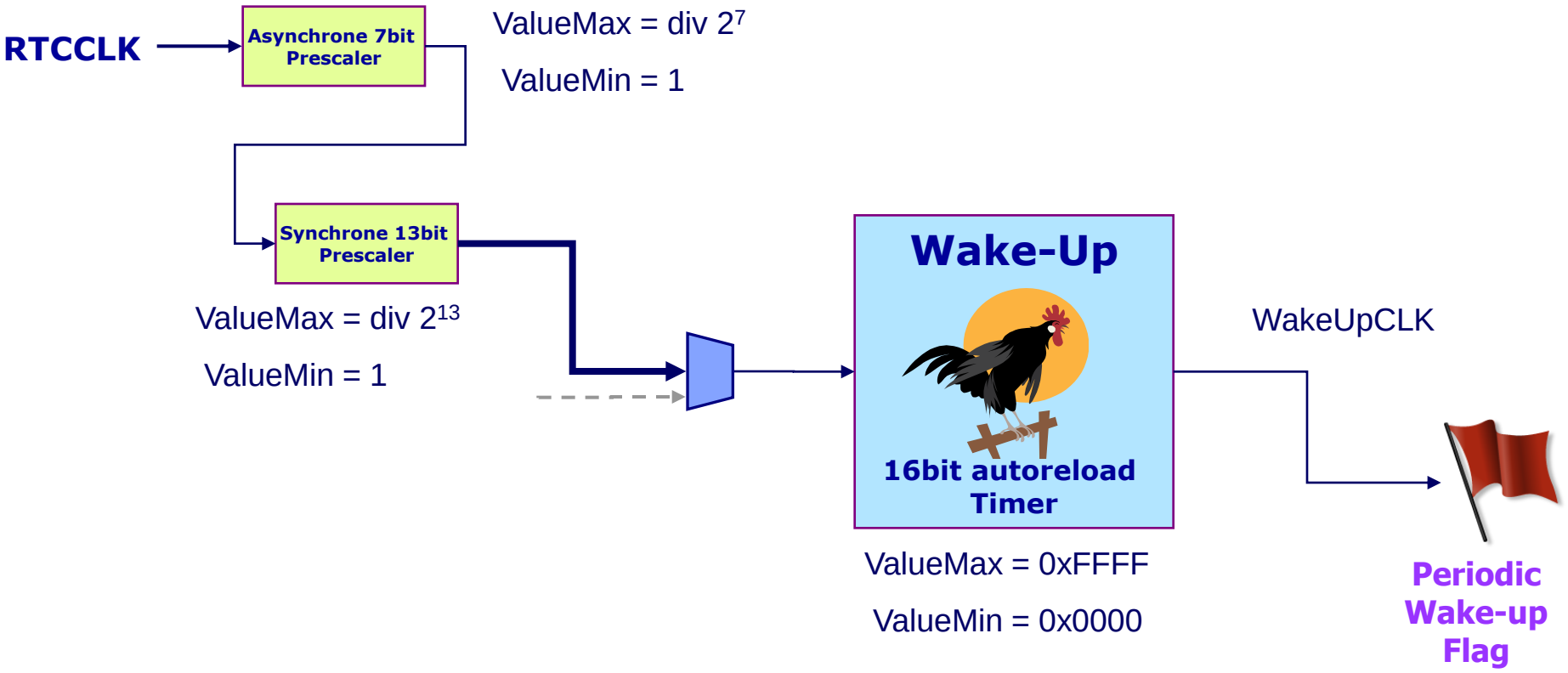


WakeUpCLKmax = RTCCLK/(1 x (0x0000 + 1))

WakeUpCLKmin = RTCCLK/(2^{20} x (0xFFFF + 1))

WakeUp configuration (3/3)

– *Case3* **WUCKSEL = 11x**



WakeUpCLKmax = RTCCLK/(1 x (0x10000 + 1))

WakeUpCLKmin = RTCCLK/(2^{20} x (0x1FFFF + 1))

Auto Wake-Up (AWU)

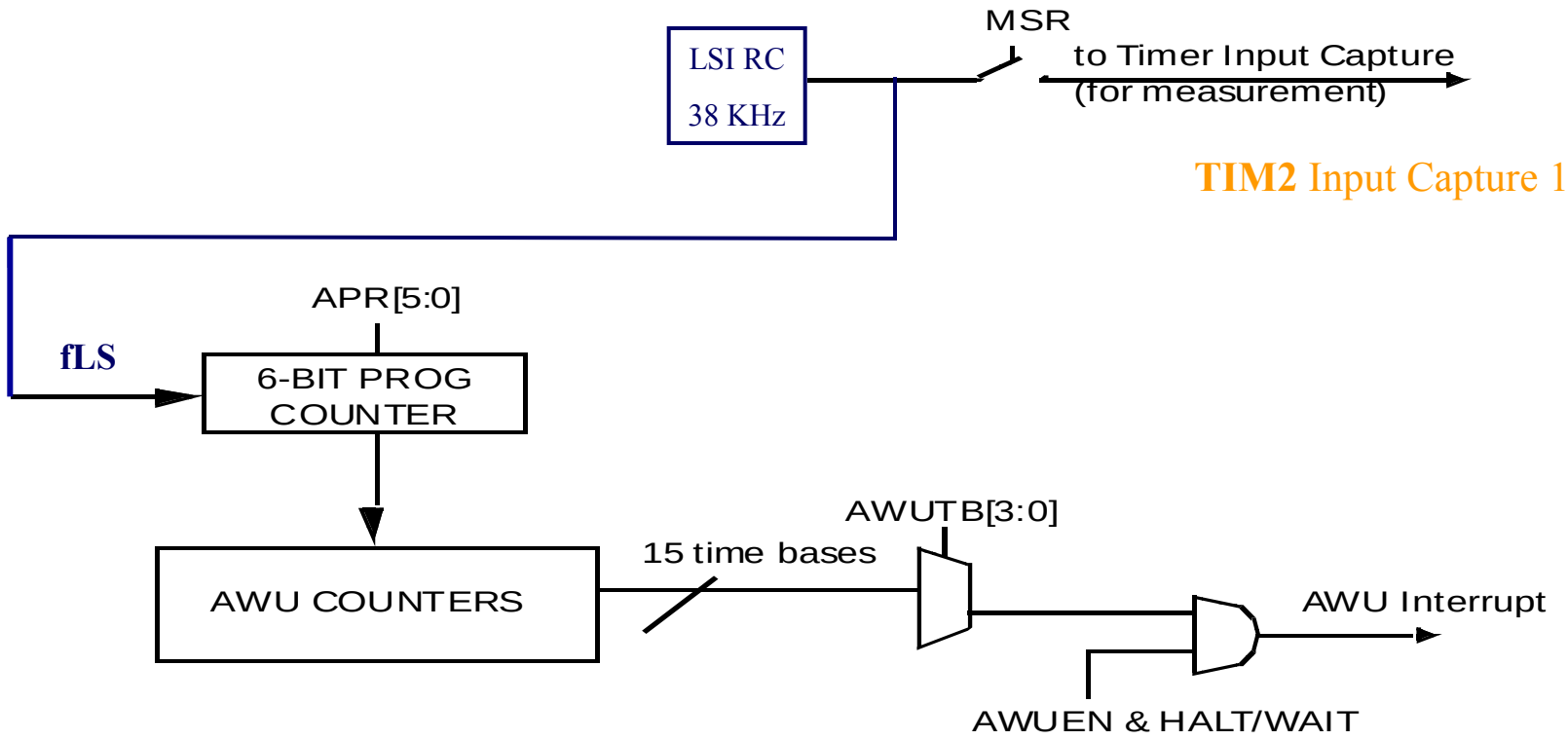
Available on the STM8L101
Not on the STM8L15x

Features

- **AWU function with Low Speed clock**
  - The AWU has been designed to implement the Active Halt low power mode.
  - Dedicated AWU interrupt vector

- **LSI frequency measurement**
  - It is possible thanks to LSI connection to one internal timer input capture. This operation allows to define precise time base for AWU and to make a calibration to keep Beeper frequency at 1K, 2K or 4K.

AWU Block Diagram



AWU description

- **AWU operation**
  1. LSI measurement if needed *(see dedicated slide)*
  2. Find the AWUTB[3:0] and APR[5:0] according to targeted wake-up interval *(see dedicated slide)*
  3. AWUEN=1 to enable the AWU interrupt and LS clock source to the counters (if LSI RC is selected it will start automatically)
  4. HALT
- **In Halt mode (after Halt instruction):**
  - the counters start to run when entering this mode
  - the AWU interrupt wakes-up the CPU at the regular programmed intervals :

This is the Active Halt mode

- **Power consumption reduction:** AWUTB[3:0] = “0000”
  - Puts AWU in stand-by: time-base OFF (default value)
- **APR[5:0] different from 3Fh (reset setting) to get counter running**

Time base selection (1/3)

- **The Time base selection is done with the determination of 2 parameters:**
  - APR[5:0], it gives the prescaler division factor APR_{DIV}
  - AWUTB[3:0], it gives the counter output rank
- **The chosen Time base depends on precise LSI frequency.**
- **We propose to define first 15 non overlapped ranges of intervals according to counter output rank.**
  - 0001 > $2/f_{LS}$ – $64/f_{LS}$, APR_{DIV} moving from 2 to 64
  - 0010 > $2x32/f_{LS}$ - $2x2x32/f_{LS}$, APR_{DIV} moving from 32 to 64 (std)
  - 0011 > $2x64/f_{LS}$ - $2x2x64/f_{LS}$, APR_{DIV} moving from 32 to 64 (std)
  - …
  - 1101 > $2^{11}x64/f_{LS}$ - $2^{11}x128/f_{LS}$, APR_{DIV} moving from 32 to 64 (std)
  - 1110 > $2^{11}x130/f_{LS}$ - $2^{11}x320/f_{LS}$, APR_{DIV} moving from 26 to 64
  - 1111 > $2^{11}x330/f_{LS}$ - $2^{12}x960/f_{LS}$, APR_{DIV} moving from 11 to 64
- **Important to measure LSI to get precise interval ranges**

Time base selection (2/3)

- **The targeted Time base is in one of the above ranges.**
  - It gives the TB value
- **Then the choice of APR_{DIV} is the result of one of the equations:**
  - Interval = $2^n x APR_{DIV}/f_{LS}$ where n depends on the TB value
  - Interval = $5x2^{11}xAPR_{DIV}/f_{LS}$
  - Interval = $30x2^{11}xAPR_{DIV}/f_{LS}$
- **The result is not necessarily an integer. It means that the targeted interval is reached with an error.**
- **The maximum error is half of the step in a given range; it is relatively more important for lowest targeted values.**
- **For standard range the maximum error is**
  - **0.5xstep/min range value -> 1/64 = 1.5625%**

Time base selection (3/3)

- **Example with f_{LS}= 38KHz**
- **Target time interval is 3s**

- **The appropriate interval range is:**
  - 1100 : $2^{10}x64/f_{LS}$ - $2^{10}x128/f_{LS}$ = 1.72s –3.45s

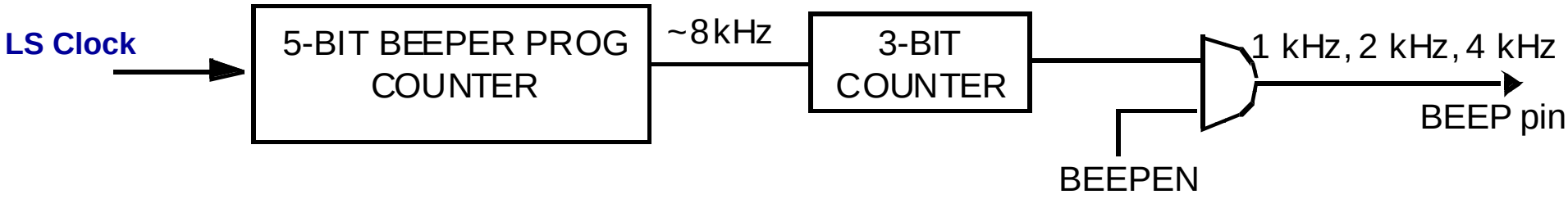
- 3s = $2x2^{10}xAPR_{DIV}/f_{LS}$ => APR_{DIV} = $3xf_{LS}/2x2^{10}$ = 55.7

- The reached time interval is :
  - $2^{11}x56/f_{LSE}$ = **3.02s** (or $2^{11}x55/f_{LSE}$ = **2.96s**)
  - This is not precisely 3s
  - Error = (3.02-3)/3 = 0.6%

Beeper (BEEP)

Available on both STM8L101 and STM8L15x

Beeper Block Diagram



- **BEEP pin = PA0 on both products**
- **LS Clock = LSI (38KHz) or LSE (32.768KHz) on STM8L15x**
  - **Possibility to measure the frequency using TIM2 input capture**
  - **Connection is done with MSR bit in BEEP_CSR1 register**
- **LS Clock = LSI (38KHz) only on STM8L101**
  - **Frequency measurement done at AWU level**

Beeper description

- **Beeper operation**
  1. LSI measurement and calibration if needed *(see dedicated slide)*
  2. Define BEEPDIV[4:0] values to get right $BEEP_{DIV}$ division factor
  3. Choice of BEEPSEL to select respectively: $f_{LS}/(8xBEEP_{DIV})$, $f_{LS}/(4xBEEP_{DIV})$ and $f_{LS}/(2xBEEP_{DIV})$
  4. BEEPEN =1

- **BEEPDIV[4:0] must be different from 1Fh (reset setting) to get counter running.**

LSI Calibration for Beeper

- **The LSI frequency is around 38KHz**
- **If you still need to adjust it to provide more accurate Beeper output at 1KHz, 2KHz or 4KHz the procedure below must be followed:**
- **Calibration procedure**
  1. MSR=1 to connect LSI to TIM2 input capture 1
  2. Measure LSI frequency with TIM2
  3. Write the calibration value BEEPDIV[4:0] as follows:

     A and x being the integer and fractional part of $f_{LSI}/8$ in KHz the calibration value is:
     - BEEPDIV[4:0] = A-2 when x is less than or equal to A/(1+2*A)
     - BEEPDIV[4:0] = A-1 otherwise

Calibration (2/2)

- **Example with f_{LSI} = 35KHz:**

  $f_{LSI}/8$ = 4.375KHz
  -> A = 4
  -> A/(1+2*A) = 0.444--

  -> x = 0.375
  -> x is less than A/(1+2*A) therefore:
  BEEPDIV[4:0] = A-2 = 2 = 02h

- The prescaler output frequency is
  - 35/4 = 8.75KHz
- Clock is calibrated for Beeper at 1.094KHz, 2.19KHz or 4.375KHz

Quiz

- Why the access to the RTC are protected after a reset ?
  - to avoid possible parasitic write accesses.
- What are the calendar fields which can be selected as alarm reference.
  - Each calendar field can be independently selected (maskable or not maskable).
- Is RTC able to exit the device from low power modes?
  - The RTC alarm can exit the device from Active-halt mode .
- What are the interrupt flags of the whole RTC IP?
  - Periodic wakeup flag/ alarm flag.
- What is the use of RSF flag?
  - To know when the RTC calendar shadow registers are updated.

CONFIDENTIALITY OBLIGATIONS:

THIS DOCUMENT CONTAINS SENSITIVE INFORMATION.

IT IS CLASSIFIED “MICROCONTROLLERS, MEMORIES & SECURE MCUs (MMS) RESTRICTED AND ITS DISTRIBUTION IS SUBMITTED TO ST/MMS AUTHORIZATION

AT ALL TIMES YOU SHOULD COMPLY WITH THE FOLLOWING SECURITY RULES:

- DO NOT COPY OR REPRODUCE ALL OR PART OF THIS DOCUMENT
- KEEP THIS DOCUMENT LOCKED AWAY
- FURTHER COPIES CAN BE PROVIDED ON A “NEED TO KNOW BASIS”, PLEASE CONTACT YOUR LOCAL ST SALES OFFICE OR DOCUMENT WRITTER.