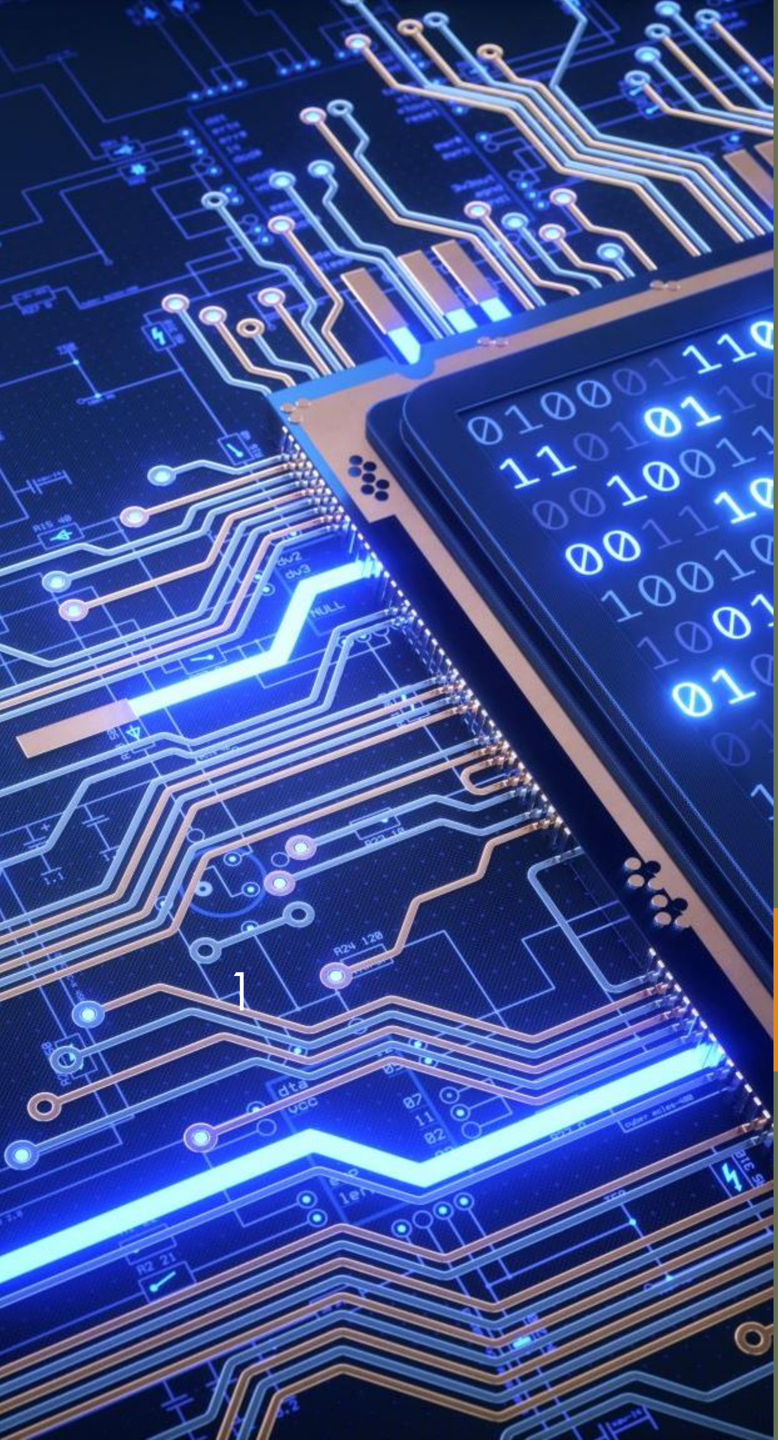


Embedded Controller programming for Real Time Systems: ECE-40097

Lesson 8

Vijay Kumar

Main Topics

RTC

DMA

Power modes

Real Time Clock (RTC)

- RTC is digital clock to provide time and date
- Never stops as long as supply voltage remains in operating range
- Why is this important, especially in connected world
  - To have initial reference of time until connection is established
  - Provides reference time without communication
  - Automatic wakeup to manage all low-power modes
- Plays significant role in embedded system for timing

Features

- Processor support RTC with HW calendar, alarms and calibration
- Automatic wake up interrupt generation
  - Need to conserve power for embedded/IOT devices
- Tamper detection
  - Used for assets safety and tracking
  - When device is in sleep mode
  - Action depends on application or use case
- DST compensation
- Synchronization with external clock
- Programmable alarms

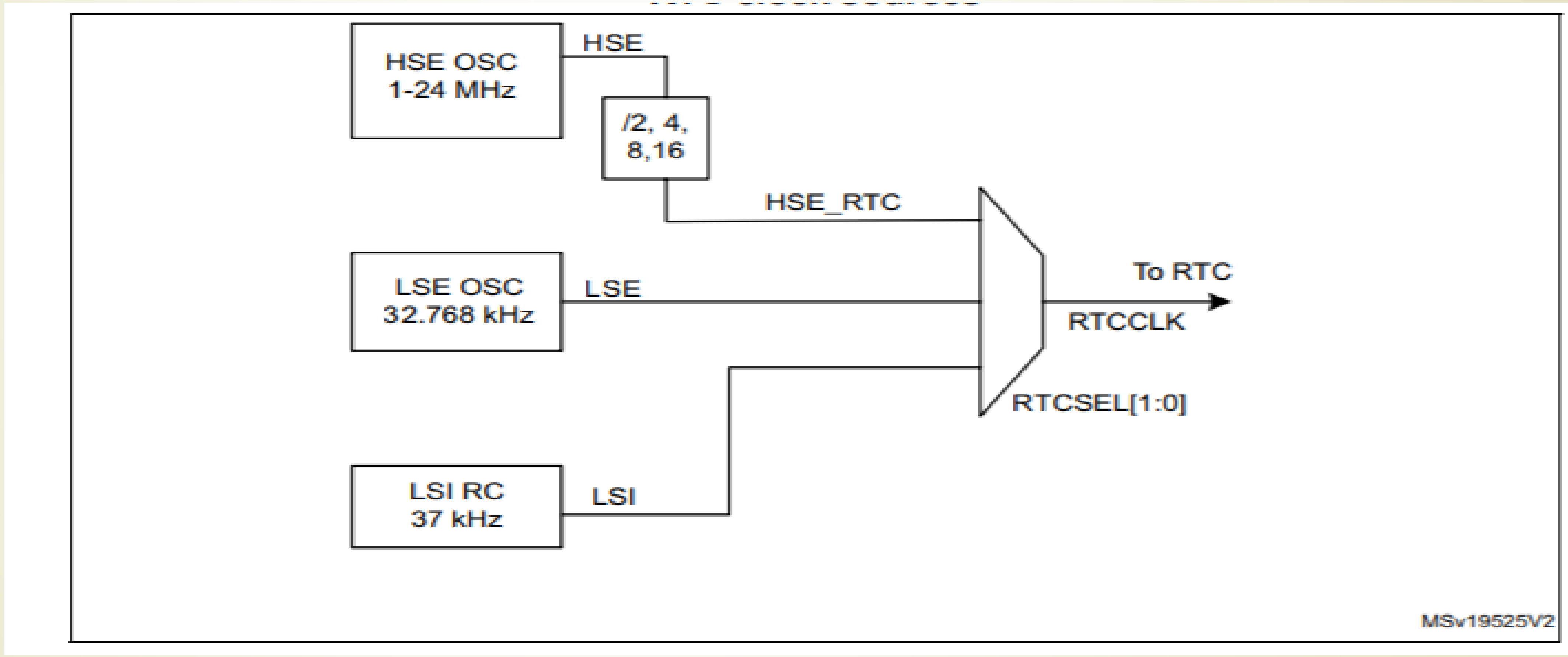
Features

- RTC is designed to minimize the power consumption
  - Real-Time Clock functionality only adds typically 300 nA to the current consumption.
- RTC keeps working in reset mode and its registers are only reset by a VBAT power-on
  - Or it has previously been powered off.
- RTC register values are not lost after a system reset, the calendar keeps the correct time and date until VDD and VBAT power down

RTC State in low power mode

- Sleep mode
- Low-power Run mode
- Low-power Sleep mode
- Stop mode if the RTC clock is provided by LSE or LSI(a)
- Standby mode if the RTC clock is provided by LSE or LSI
- Shutdown mode if the RTC clock is provided by LSE(b)

Clock Source



RTC Requirements

- Low power consumption
- Accuracy of RTC depends usage depends on applications
  - High accurate chips are expensive
- Separately powered by a battery
- Run independent from processor core
- Sometimes omitted to save cost for connected device

UNIX Epoch time

- Number of seconds elapsed since 00:00:00 UTC, Thursday, 1 January 1970
- Uses 32 bits to hold the time
- Holds up to year 2038 and then overflows
  - Known as year 2038 problem
- Fix is to use 64 bits
  - Could be challenge for 32 bits embedded systems

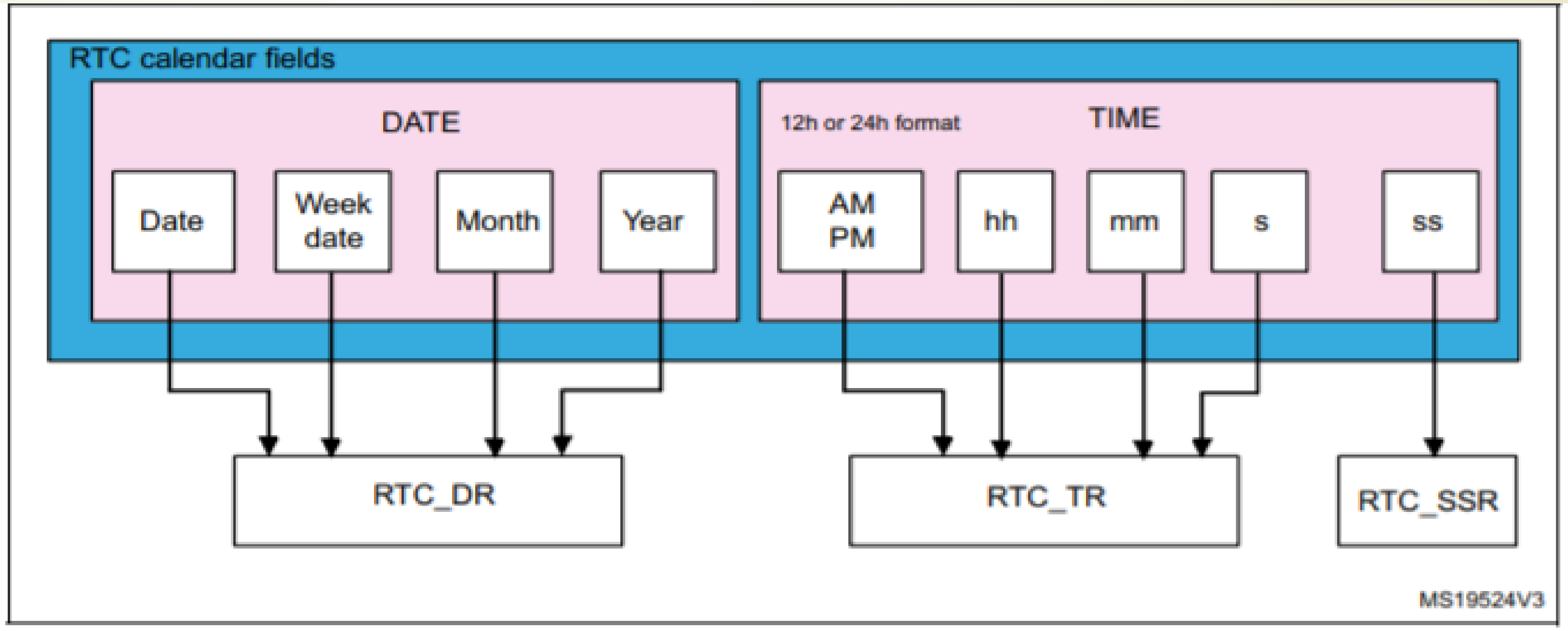
RTC Crystal - PPM

- PPM is parts per million
  - 1 PPM is ± 1.1 seconds per year
  - Typical watch crystal is 22 PPM
- Tends to vary with temperature
  - STM32 RTC is 1000 PPM at 25 degree C
- Gets expensive with lower PPM
- Typical frequency is 32.768 Khz

RTC Synchronization

- Synchronizes the time with connected device
  - Using GPS
  - Using NTP server
- Code is written to program RTC with correct time
  - For critical system and hard real time systems
  - Done periodically
- Becomes the source for every source modules
  - Especially true for dual processor board with RTC on one board

RTC Calendar



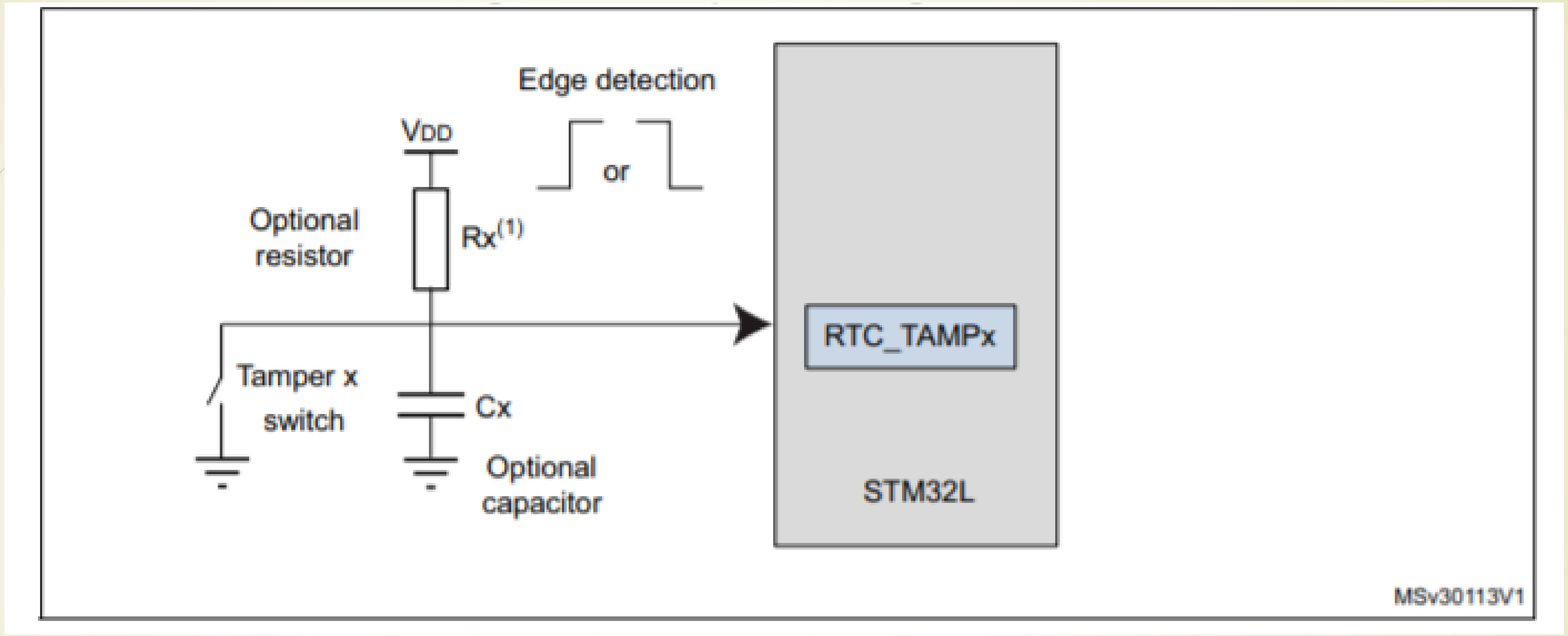
1. RTC_DR, RTC_TR are RTC date and time registers.
2. The sub-second field (RTC_SSR) is the value of the synchronous prescaler counter. This field is not writable.

RTC Date Structure

```
typedef struct
{
  uint8_t WeekDay;  /*!< Specifies the RTC Date WeekDay.
                         This parameter can be a value of @ref RTC_WeekDay_Definitions */

  uint8_t Month;    /*!< Specifies the RTC Date Month (in BCD format).
                         This parameter can be a value of @ref RTC_Month_Date_Definitions */

  uint8_t Date;     /*!< Specifies the RTC Date.
                         This parameter must be a number between Min_Data = 1 and Max_Data = 31
*/

  uint8_t Year;     /*!< Specifies the RTC Date Year.
                         This parameter must be a number between Min_Data = 0 and Max_Data = 99
*/

} RTC_DateTypeDef;
```

RTC Time Structure

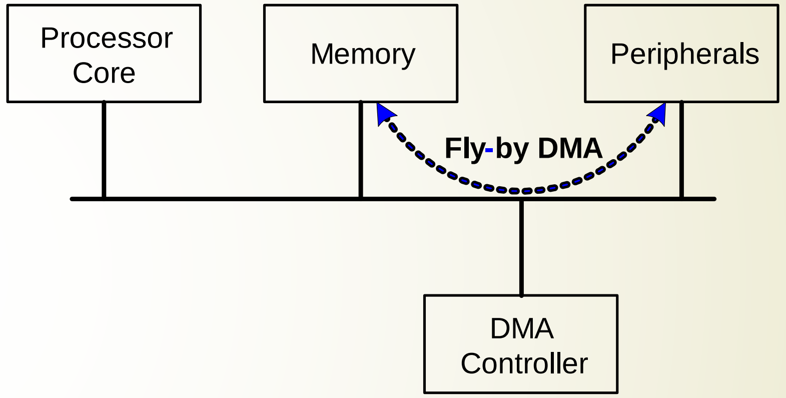
```
typedef struct
{
  uint8_t Hours;          /*!< Specifies the RTC Time Hour.
                               This parameter must be a number between Min_Data = 0 and Max_Data = 12 if the RTC_HourFormat_12 is selected.
                               This parameter must be a number between Min_Data = 0 and Max_Data = 23 if the RTC_HourFormat_24 is selected */

  uint8_t Minutes;        /*!< Specifies the RTC Time Minutes.
                               This parameter must be a number between Min_Data = 0 and Max_Data = 59 */

  uint8_t Seconds;        /*!< Specifies the RTC Time Seconds.
                               This parameter must be a number between Min_Data = 0 and Max_Data = 59 */

  uint8_t TimeFormat;     /*!< Specifies the RTC AM/PM Time.
                               This parameter can be a value of @ref RTC_AM_PM_Definitions */

  uint32_t SubSeconds;    /*!< Specifies the RTC_SSR RTC Sub Second register content.
                               This parameter corresponds to a time unit range between [0-1] Second
                               with [1 Sec / SecondFraction +1] granularity */

  uint32_t SecondFraction;  /*!< Specifies the range or granularity of Sub Second register content
                               corresponding to Synchronous pre-scaler factor value (PREDIV_S)
                               This parameter corresponds to a time unit range between [0-1] Second
                               with [1 Sec / SecondFraction +1] granularity.
                               This field will be used only by HAL_RTC_GetTime function */

  uint32_t DayLightSaving;  /*!< Specifies RTC_DayLightSaveOperation: the value of hour adjustment.
                               This parameter can be a value of @ref RTC_DayLightSaving_Definitions */

  uint32_t StoreOperation;  /*!< Specifies RTC_StoreOperation value to be written in the BKP bit
                               in CR register to store the operation.
                               This parameter can be a value of @ref RTC_StoreOperation_Definitions */
} RTC_TimeTypeDef;
```

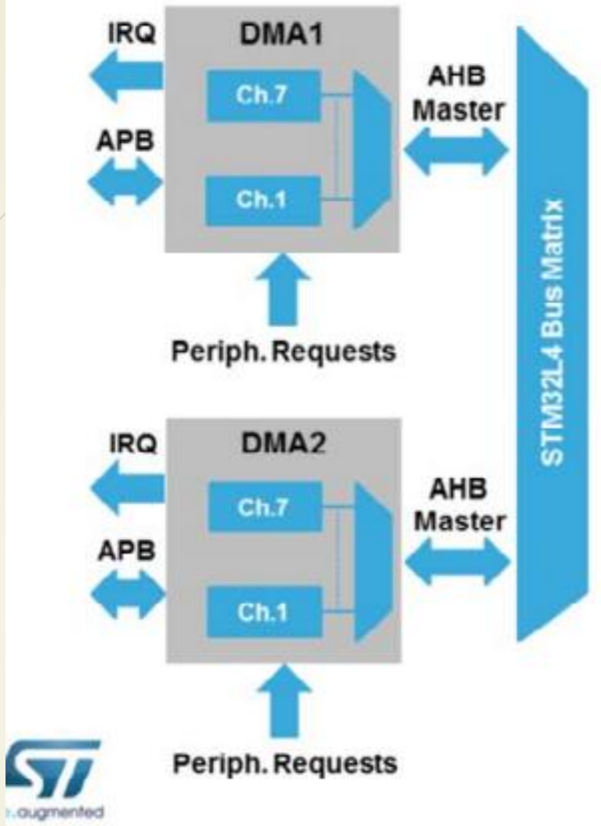


RTC Tamper Detection

What is DMA

- Direct memory Access
- Data transfer between a peripheral to memory and Vice - versa
- Data transfer between a peripheral to another peripheral
- Data transfer from memory to memory
- Releases CPU from moving data
- Uses bus matrix to allow concurrent transfers





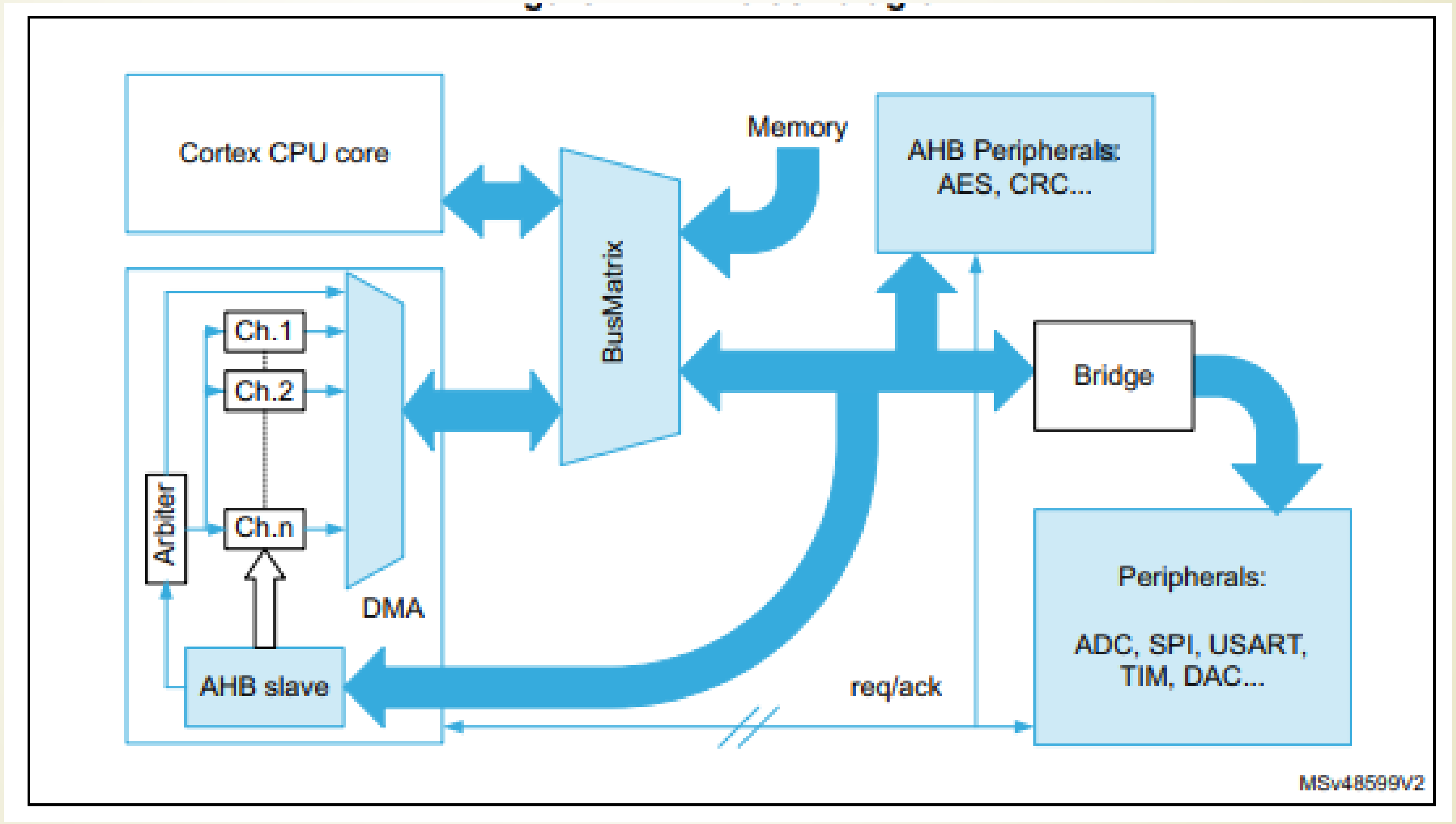
- STM32L4 DMA features
  - Flexible configuration
  - Hardware and software priority management
  - Configurable data transfer modes
    - Peripheral-to-Memory, Memory-to-Peripheral, Peripheral-to-Peripheral, and Memory-to-Memory modes

Application benefits

- DMA support for timers, ADC, and communication peripherals
- Offload CPU from data transfer management
- Simple integration

DMA Overview

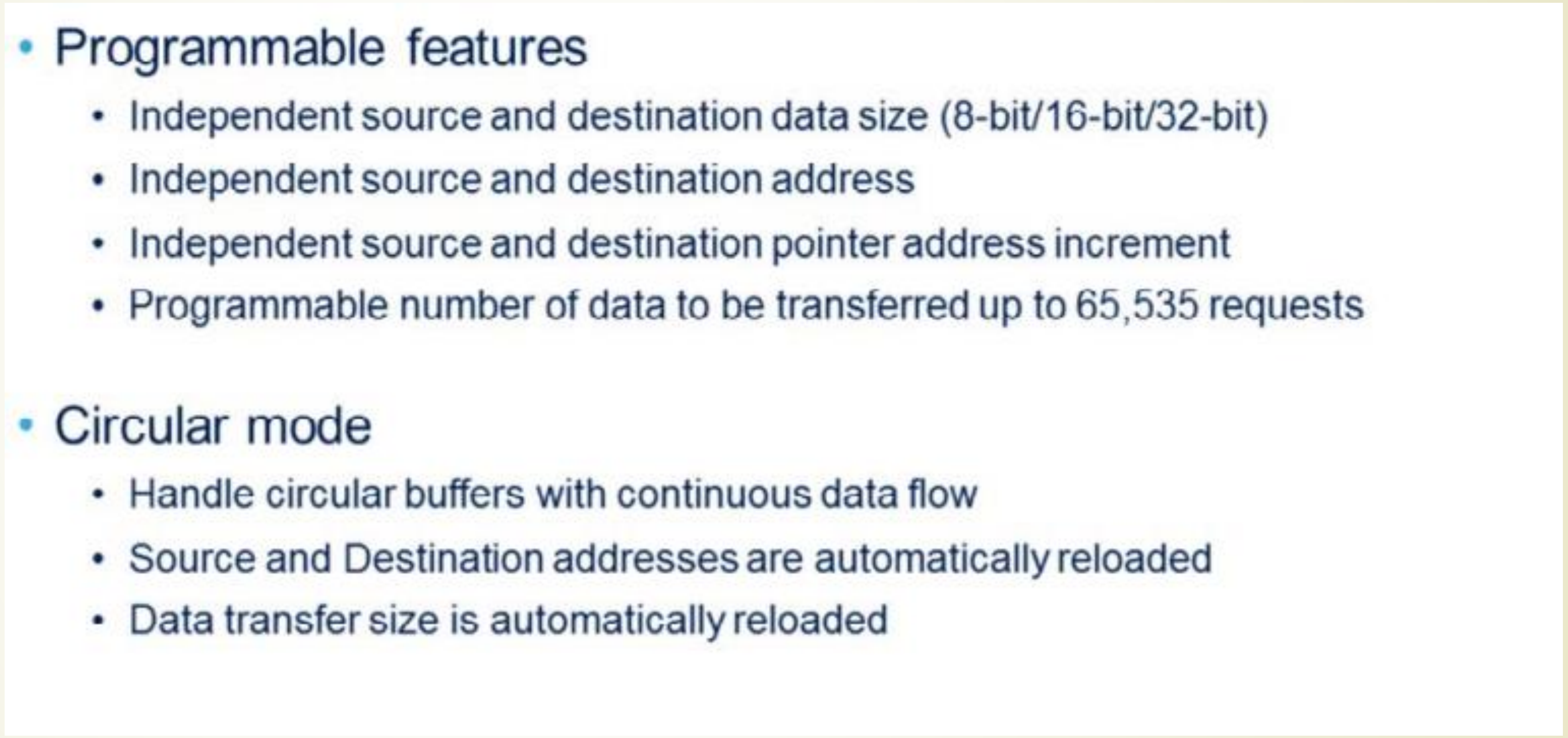
DMA block Diagram



DMA controller

- Two controllers
  - Each with 7 channels - total of 14 channels
  - Each are capable of managing memory access from one or more peripherals
- Only CPU or Controller act as masters
  - All other connected parts are slaves
- No arbitration is needed if masters are not crossing each other
- Memory-to-peripheral or peripheral-to-memory
  - Managed at HW Level
- Memory-to-memory
  - Activated by SW

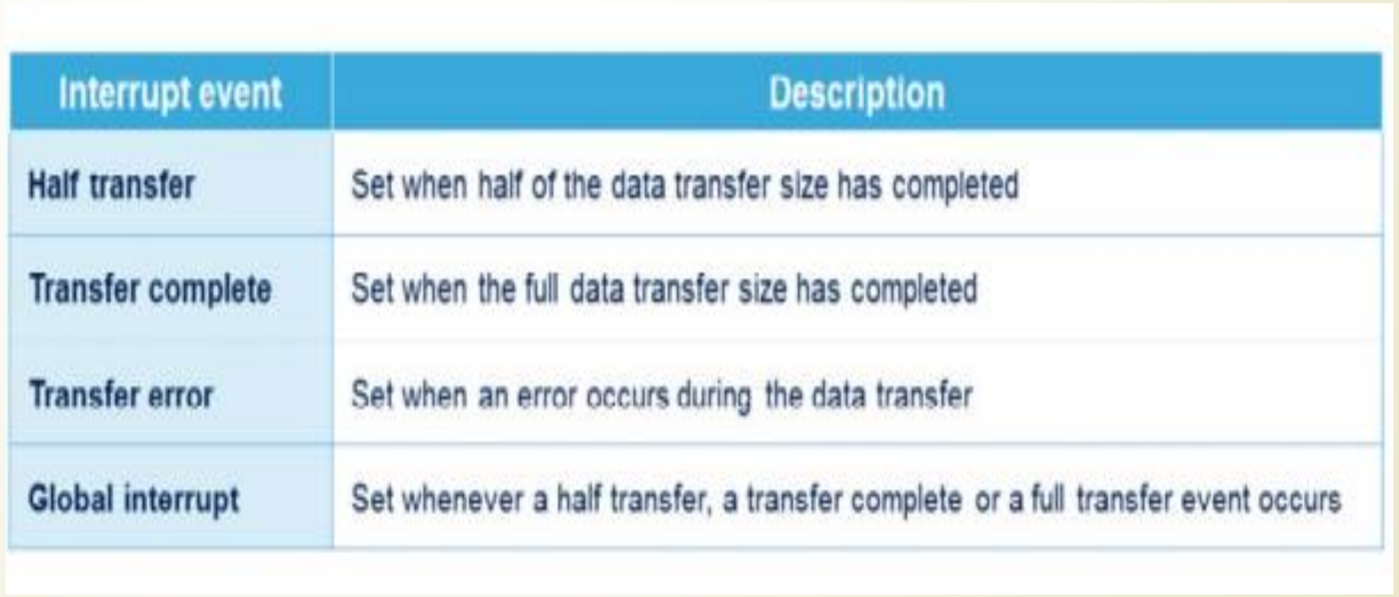
Channel Flexibility



- Programmable features
  - Independent source and destination data size (8-bit/16-bit/32-bit)
  - Independent source and destination address
  - Independent source and destination pointer address increment
  - Programmable number of data to be transferred up to 65,535 requests
- Circular mode
  - Handle circular buffers with continuous data flow
  - Source and Destination addresses are automatically reloaded
  - Data transfer size is automatically reloaded

DMA Controller

- Software programmable priorities
  - Very high, High, Medium or Low
- Independent channel interrupt flags
  - DMA Half Transfer
  - DMA Transfer complete
  - DMA Transfer Error
  - Global Interrupt
- Channel is automatically disabled in case of Bus Error

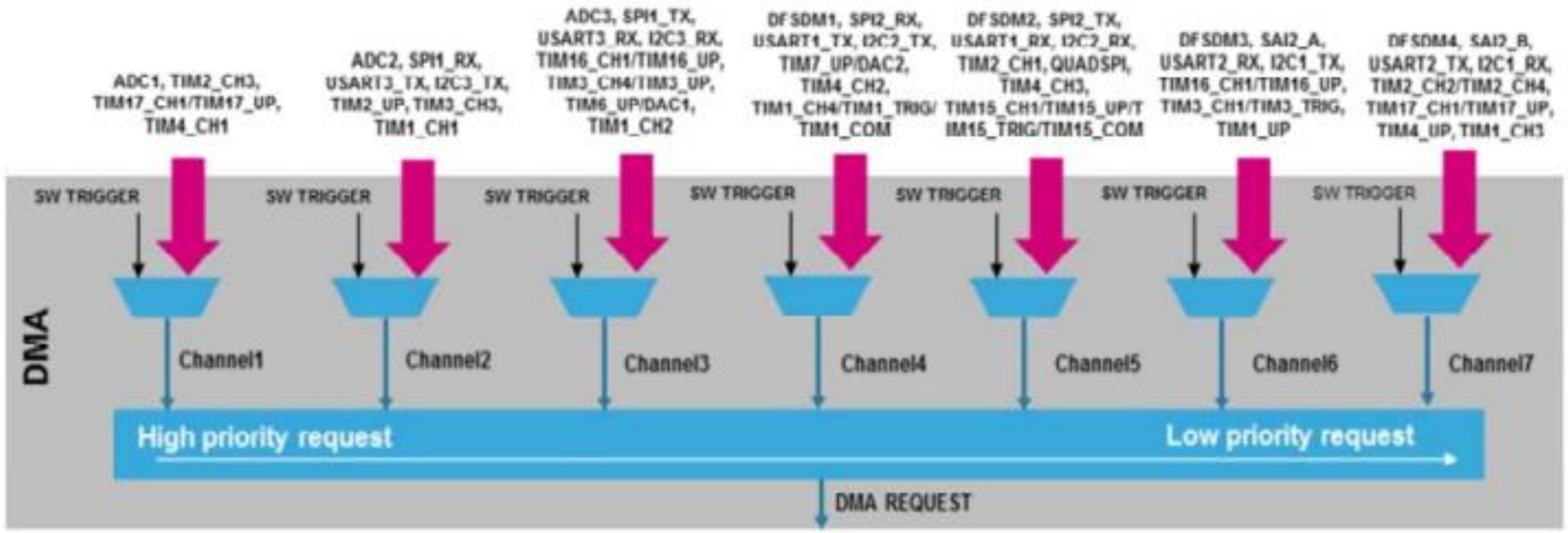


Interrupt event	Description
Half transfer	Set when half of the data transfer size has completed
Transfer complete	Set when the full data transfer size has completed
Transfer error	Set when an error occurs during the data transfer
Global interrupt	Set whenever a half transfer, a transfer complete or a full transfer event occurs

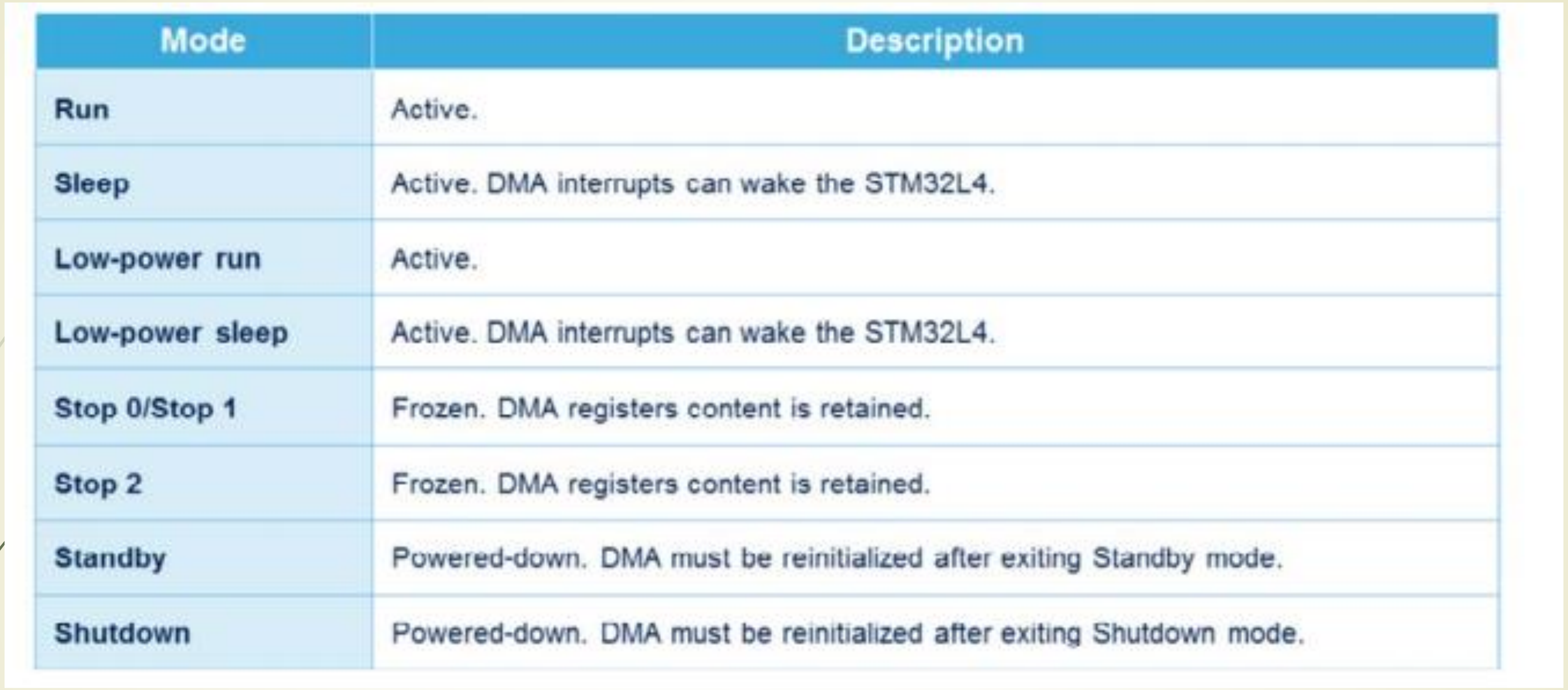
Data Transfer

- Four steps are required to perform a DMA data transfer.
  - The first step is the arbitration for the bus access
  - When successful, address computation follows
  - The third step is a single data transfer itself.
  - The fourth and final step is the acknowledge handshake
- Latency
  - A latency describes the delay between the request activation of the DMA data transfer and the actual completion of the request.
  - The request is either hardware-driven from a peripheral or is software-driven when channel is enabled

- DMA1 controller provides access to 7 channels
  - New: Peripheral requests are mapped through a multiplexer (Not OR gate)
  - Independent software trigger for each channel


ADC1, TIM2_CH3, TIM17_CH1/TIM17_UP, TIM4_CH1
ADC2, SPI1_RX, USART3_TX, I2C3_TX, TIM2_UP, TIM3_CH3, TIM1_CH1
ADC3, SPI1_TX, USART3_RX, I2C3_RX, TIM16_CH1/TIM16_UP, TIM3_CH4/TIM3_UP, TIM6_UP/DAC1, TIM1_CH2
DFSDM1, SPI2_RX, USART1_TX, I2C2_TX, TIM7_UP/DAC2, TIM4_CH2, TIM1_CH4/TIM1_TRIG/ TIM1_COM
DFSDM2, SPI2_TX, USART1_RX, I2C2_RX, TIM2_CH1, QUADSPI, TIM4_CH3, TIM15_CH1/TIM15_UP/T IM15_TRIG/TIM15_COM
DFSDM3, SAI2_A, USART2_RX, I2C1_TX, TIM16_CH1/TIM16_UP, TIM3_CH1/TIM3_TRIG, TIM1_UP
DFSDM4, SAI2_B, USART2_TX, I2C1_RX, TIM2_CH2/TIM2_CH4, TIM17_CH1/TIM17_UP, TIM4_UP, TIM1_CH3
SW TRIGGER
SW TRIGGER
SW TRIGGER
SW TRIGGER
SW TRIGGER
SW TRIGGER
SW TRIGGER
DMA
Channel1
Channel2
Channel3
Channel4
Channel5
Channel6
Channel7
High priority request
Low priority request
DMA REQUEST

DMA1 request mapping



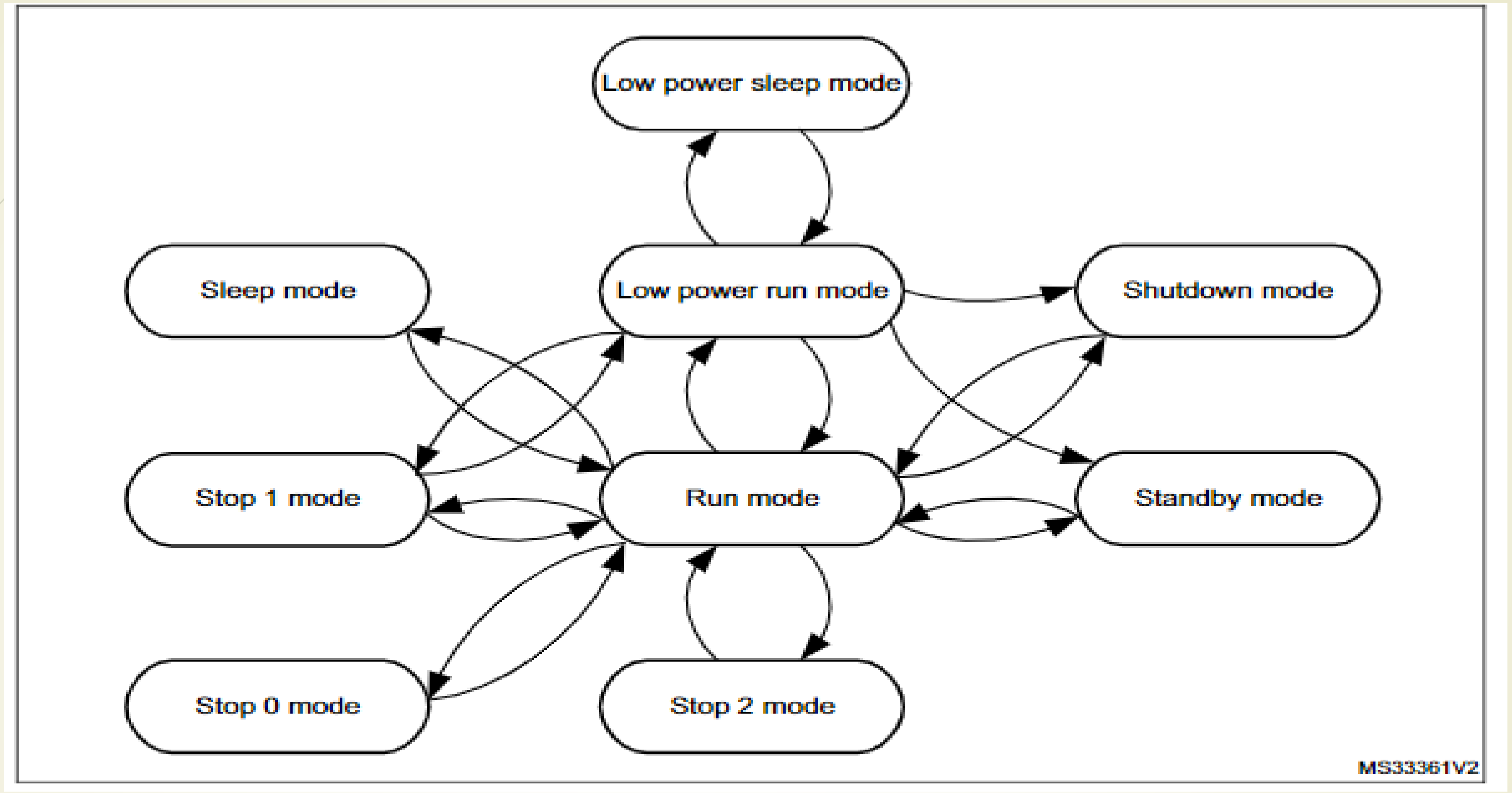
Mode	Description
Run	Active.
Sleep	Active. DMA interrupts can wake the STM32L4.
Low-power run	Active.
Low-power sleep	Active. DMA interrupts can wake the STM32L4.
Stop 0/Stop 1	Frozen. DMA registers content is retained.
Stop 2	Frozen. DMA registers content is retained.
Standby	Powered-down. DMA must be reinitialized after exiting Standby mode.
Shutdown	Powered-down. DMA must be reinitialized after exiting Shutdown mode.

DMA in Low Power Mode

DMA Advantage

- Without DMA
  - CPU has to execute many load and store instructions
  - leading to slower performance

- With DMA
  - Makes an automatic data transfer when received a DMA request without involving CPU
  - Accelerates the overall performance.


Low power sleep mode
Sleep mode
Low power run mode
Shutdown mode
Stop 1 mode
Run mode
Standby mode
Stop 0 mode
Stop 2 mode
MS33361V2

Power Mode

Power Modes

- Stop Mode
  - In these Stop modes all the high speed oscillators (HSE, MSI, HSI) are stopped, while the low speed ones (LSE, LSI) can be kept active.
- Standby Mode
  - Wakeup from this mode could be done by RTC, if clocked by the low-speed oscillators (LSE or LSI)
- Shutdown Mode
  - Shutdown mode is implemented in the STM32L4xx devices in order to lengthen even more the battery life of battery-powered applications.
  - This mode allows the lowest consumption, by switching off the internal voltage regulators, and by disabling the voltage power monitoring.
  - Wakeup from this mode could be done by RTC

Power Modes

- Could transition to run mode from any state except Low power sleep mode
- Transition to shutdown mode is from run mode (low power)
- Could be simplified with 3 modes
  - Run, sleep (suspend) and shutdown (power off)
  - From sleep to run mode is nothing but Resume
  - Transition to shutdown after few cycles (pre configured) of suspend and resume

Power Modes

- Challenge is to wakeup the unit from shutdown
  - Will require external trigger for automatic wakeup
    - E.g ignition on Vehicles
    - Key to turn on the device
- Timers or RTC could be used to wake up from Sleep/suspend periodically
- Use every occasion to shutdown or power cycle the device
  - With minimal or no impact on performance or data integrity

Power conservation

- Every effort should be made to conserve the power
- It is very specific to type of embedded device
- Any communication chips are good source of power drain
  - Minimize the number of messages
  - Turn off or put the device in suspend mode (if supported)
  - May be completely turn off the device (if HW supports) in suspend mode
- Highly recommend to do through testing

Next Lesson Topic

- Embedded Software design
- Future development on board
- Feedback from you
- Course Wrap up