EFM32 Series 0: Serial interfaces

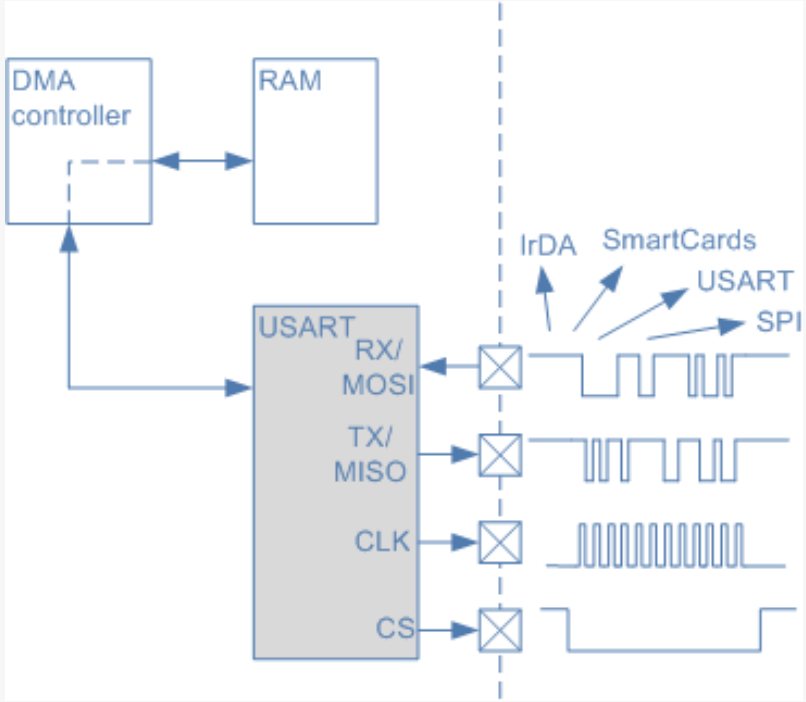
Agenda

- SPI/UART
- I2C
- LEUART
- USB
  - Hardware considerations
  - Device configuration (descriptors)
  - Stack API
  - States and callbacks
- LEUART Hands-On

SPI and UART

SPI and UART Highlights

- Up to 3 USARTs
  - UART/SPI (master/slave)
  - IrDA
  - SmartCards (ISO7816)
  - 8 Mbit/s UART, 16 Mbit/s SPI master
  - I^2S support (ZG, TG, LG, GG and WG)
- Up to 2 UARTs
  - Subset of USART with support for asynchronous communication



ZERO ARM Cortex-M0+ TINY ARM Cortex-M3 GECKO ARM Cortex-M3 LEOPARD ARM Cortex-M3 GIANT ARM Cortex-M3 WONDER ARM Cortex-M4 FPU

LINKS

Baud Rate Calculation

USART Baud Rate

$$br = f_{HFPERCLK}/(oversample \times (1 + USARTn\_CLKDIV/256))$$

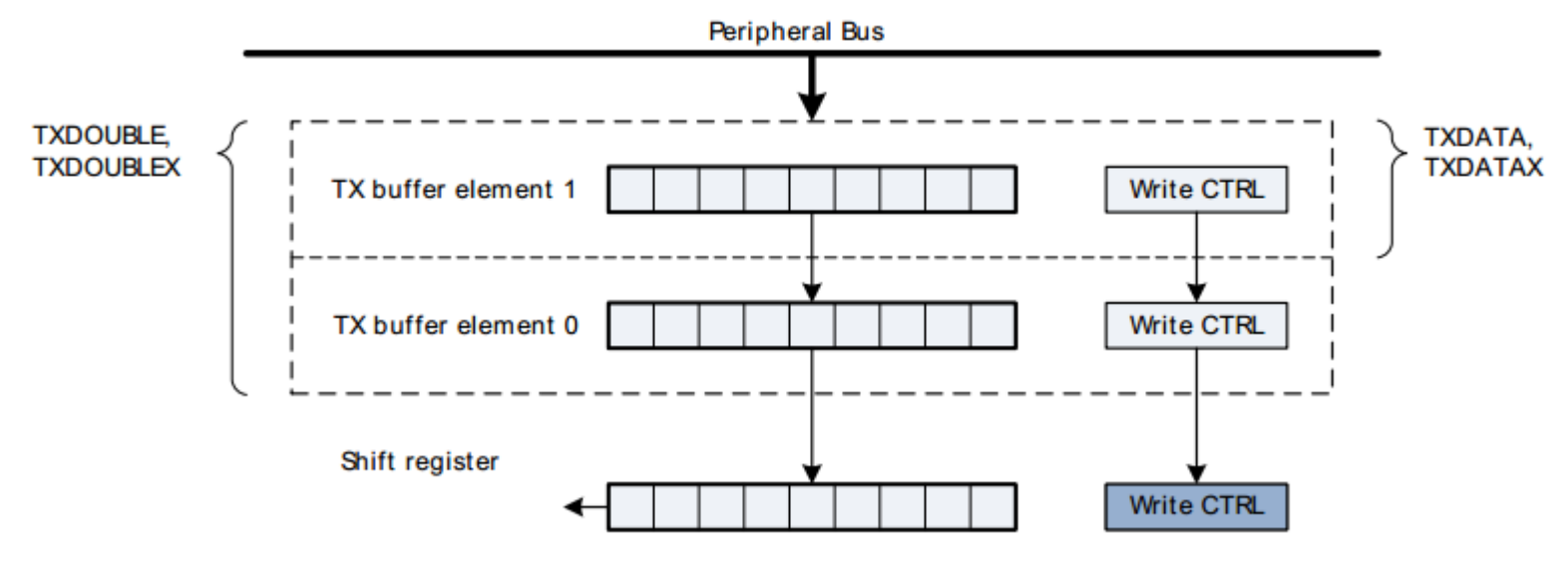
USART Desired Baud Rate

$$USARTn\_CLKDIV = 256 \times (f_{HFPERCLK}/(oversample \times br_{desired}) - 1)$$

```
/* Prepare struct for initializing UART in asynchronous mode*/
uartInit.enable       = usartDisable;   /* Don't enable UART upon intialization */
uartInit.refFreq      = 0;              /* Provide information on reference frequency. When set to 0, the reference frequency is */
uartInit.baudrate     = 115200;         /* Baud rate */
uartInit.oversampling = usartOVS16;     /* Oversampling. Range is 4x, 6x, 8x or 16x */
uartInit.databits     = usartDatabits8; /* Number of data bits. Range is 4 to 10 */
uartInit.parity       = usartNoParity;  /* Parity mode */
uartInit.stopbits     = usartStopbits1; /* Number of stop bits. Range is 0 to 2 */
uartInit.mvdis        = false;          /* Disable majority voting */
uartInit.prsRxEnable  = false;          /* Enable USART Rx via Peripheral Reflex System */
uartInit.prsRxCh      = usartPrsRxCh0;  /* Select PRS channel if enabled */

/* Initialize USART with uartInit struct */
USART_InitAsync(uart, &uartInit);
```

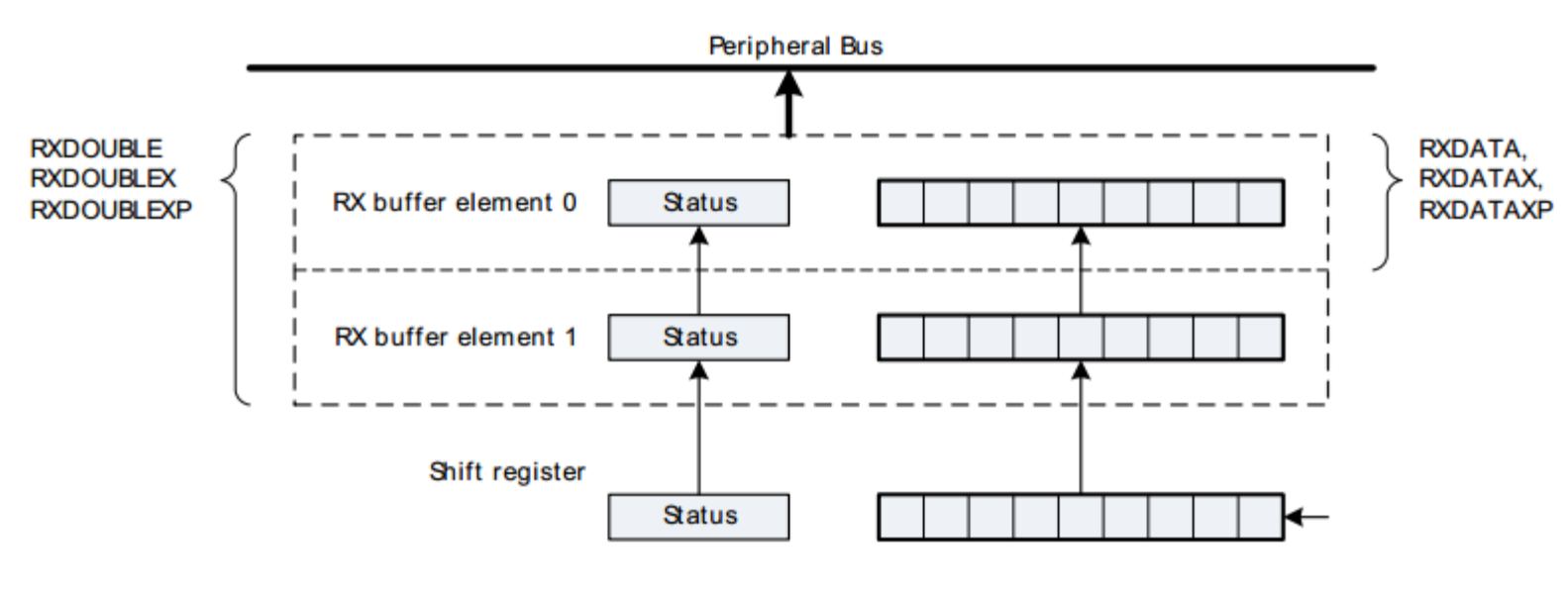
Transmit Buffer

- 2 level FIFO + shift register
- TXDOUBLE writes to both elements
- TXBIL controls TXBL IRQ
  - 0 => IRQ goes high when buffer is empty
  - 1 => IRQ goes high when buffer is half-empty
- IRQs: TXBL, TXOF, TXC



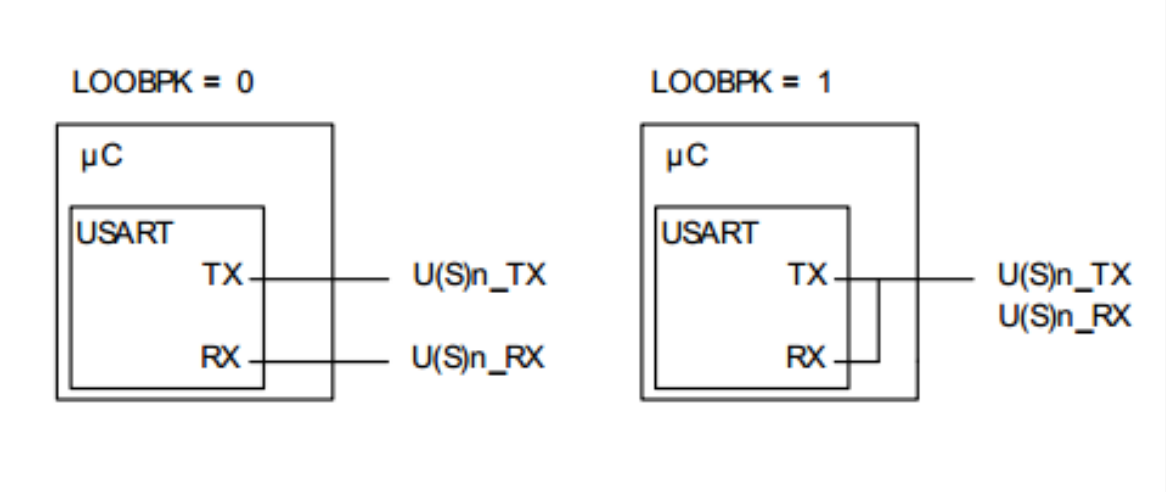
Receive Buffer

- 2 level FIFO + shift register
- RXDOUBLE reads both elements
- RXDATAV IRQ cleared by reading RXDATA/RXDOUBLE
- ‘Peak’ registers leaves contents unchanged

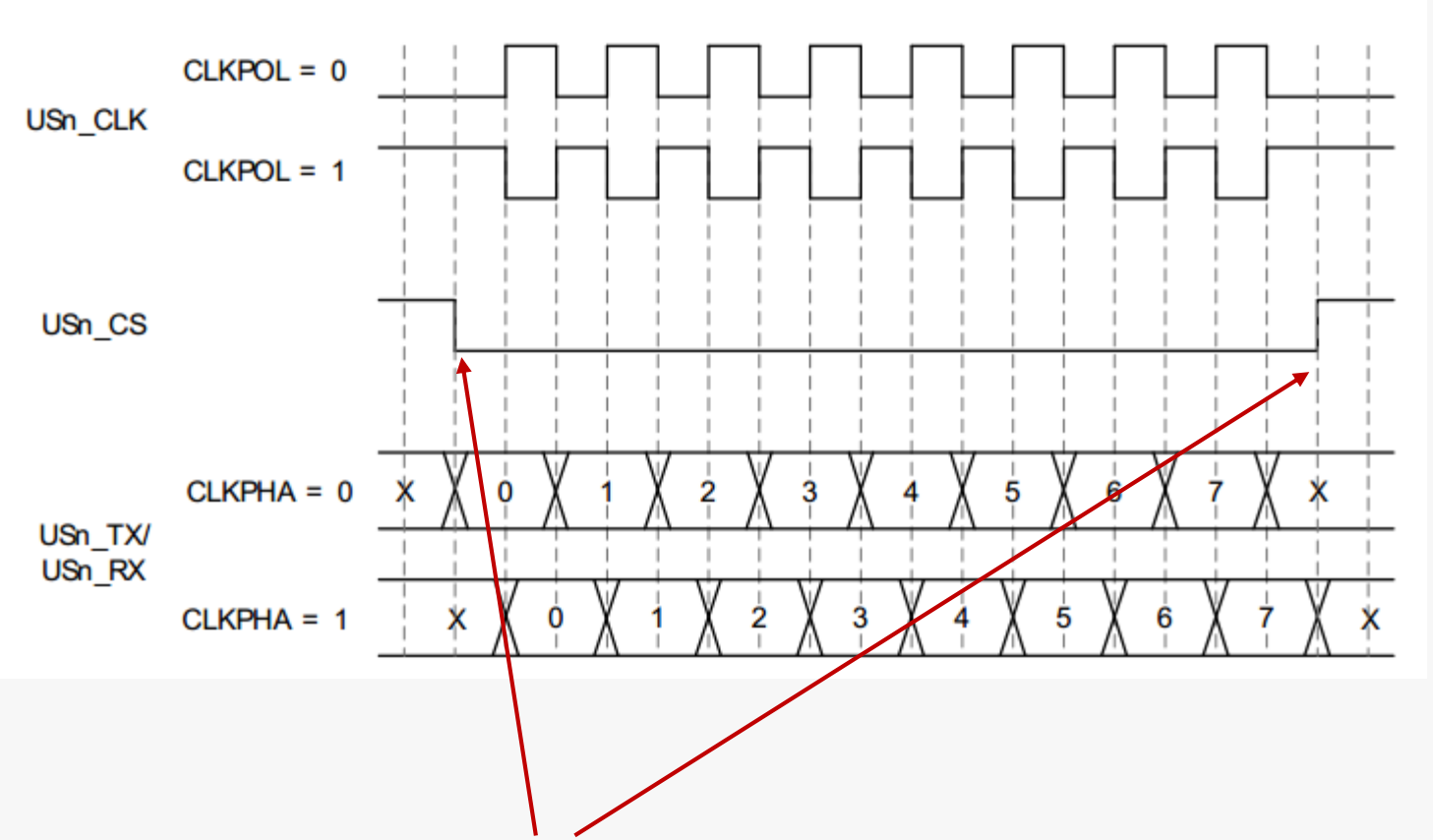


Half-Duplex

- **Allows communication using only one line**
- **LOOPBCK connects RX to TX pin**
- **TXTRIEN tristates transmitter when idle**



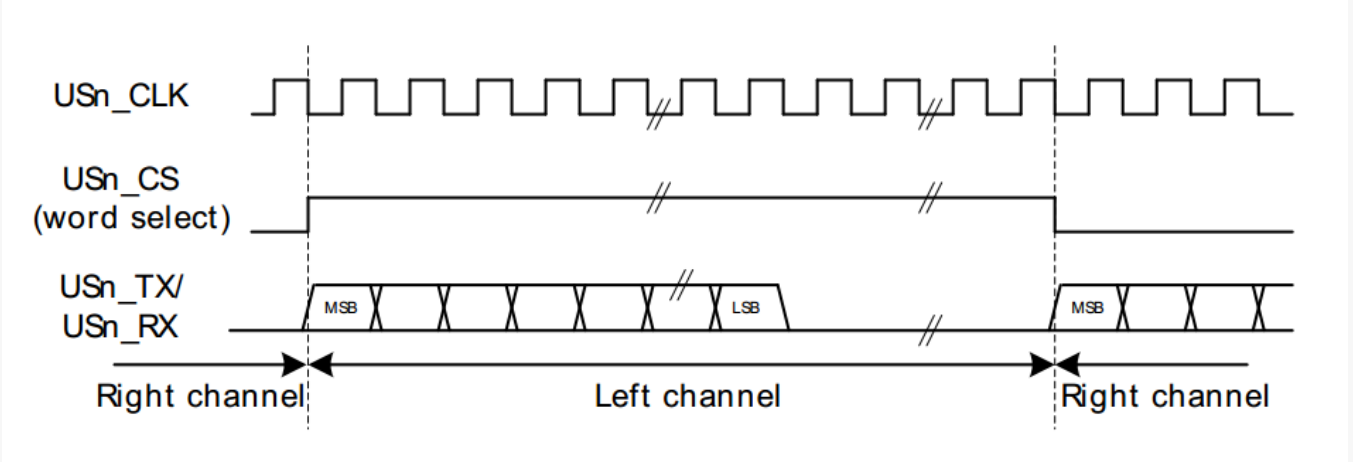
SPI Timing



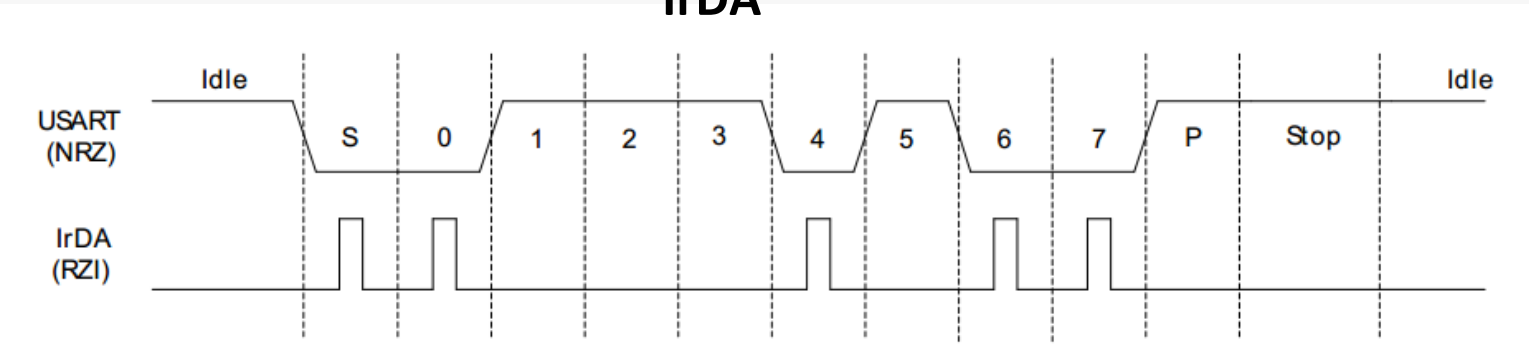
CS hold time cannot be configured with AUTOCS.
Use GPIO if precise control is required.

USART Special Modes

I2S



IrDA

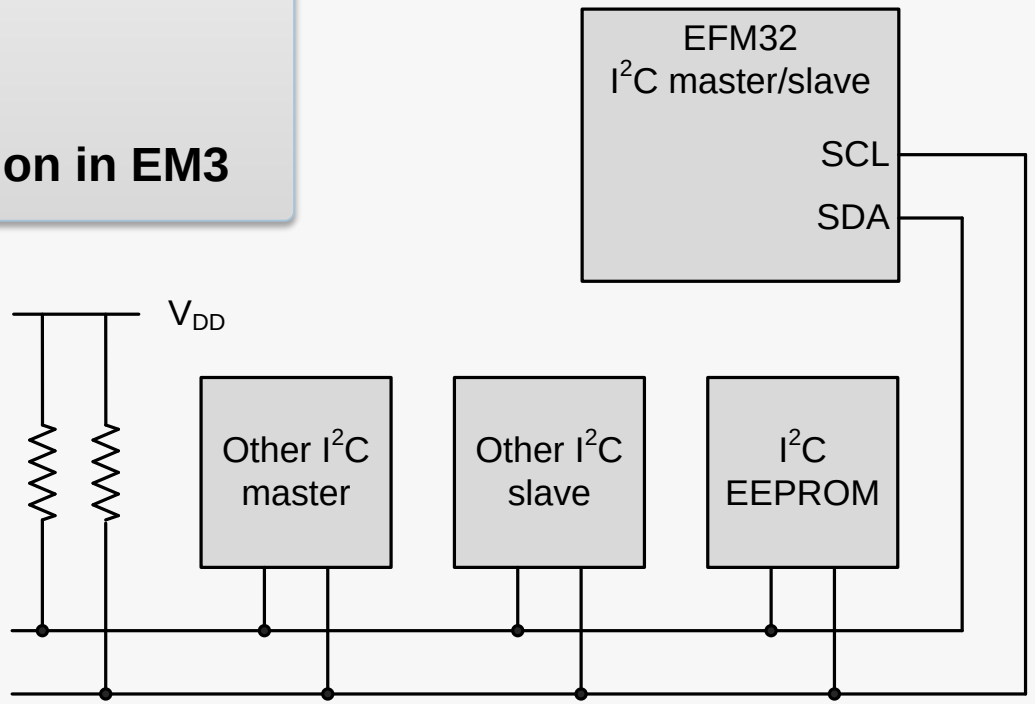


SPI DMA RX

- Problem: SPI Master must transmit in order to receive
- Solution 1: Dummy TX DMA
  - Needs two DMA channels
  - Can cause problem if RX DMA is unable to keep up (TX DMA channel has no knowledge of RXDATAV/RXFULL)
- Solution 2: AUTOTX
  - SPI transmits automatically while space in RX buffer
  - Currently only works on WG (errata LG,GG,TG)
  - Last bytes must be fetched with interrupts. Example in AN0013

I²C Highlights

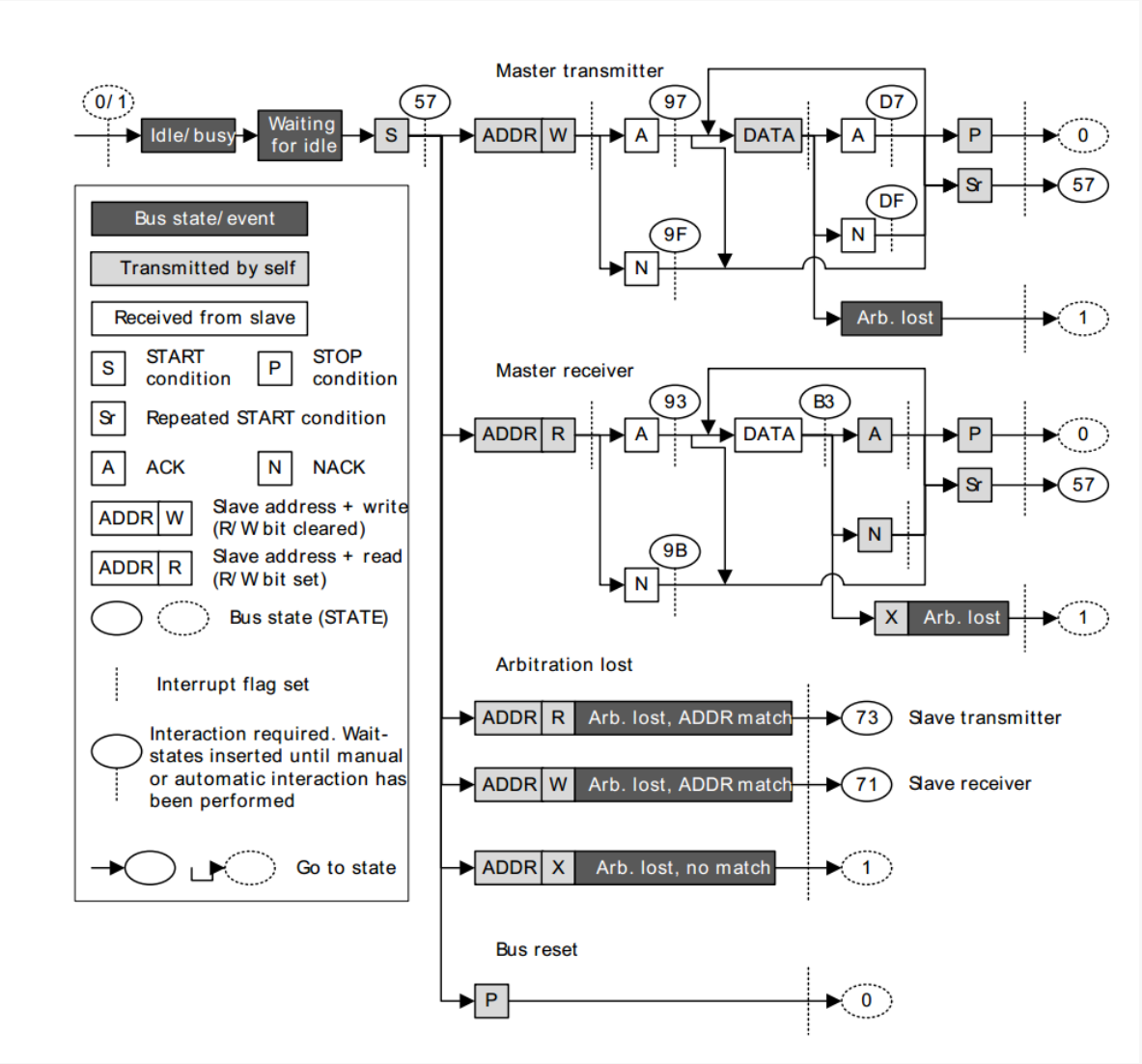
- Up to 2 I²C peripherals included
- I²C and SMBus support
- Data rates up to 1 MBit/s
- Hardware address recognition in EM3



ZERO ARM Cortex-M0+ | TINY ARM Cortex-M3 | GECKO ARM Cortex-M3 | LEOPARD ARM Cortex-M3 | GIANT ARM Cortex-M3 | WONDER ARM Cortex-M4 FPU

LINKS

I2C State Machine



I2C

- emlib handles I2C protocol state machine
- Polled transfer:

```
while (I2C_Transfer(I2C0) == i2cTransferInProgress);
```

- Interrupt driven transfer:

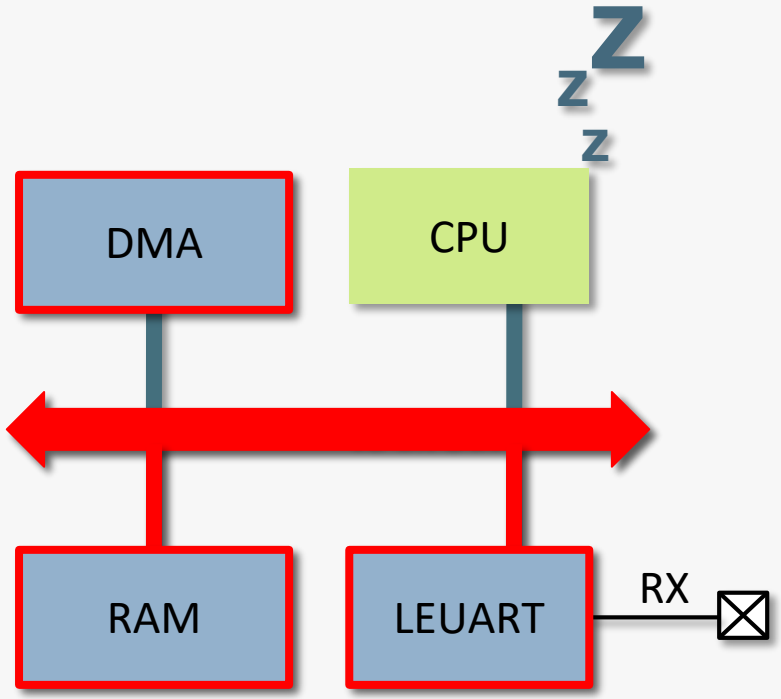
```
volatile I2C_TransferReturn_TypeDef i2cStatus;

void I2C0_IRQHandler(void)
{
  i2cStatus = I2C_Transfer(I2C0);
}

bool i2cActive = true;
i2cStatus = I2C_TransferInit(i2c, &seq);

while ( i2cActive )
{
  INT_Disable();
  if ( i2cStatus == i2cTransferInProgress ) {
    EMU_EnterEM1();
  } else {
    i2cActive = false;
  }
  INT_Enable();
}
```

DMA example in AN0013

Low Energy UART

LEUART Highlights

- Up to 2 LEUARTs
- Full UART with 32 kHz clock
- Available even in Deep Sleep
- 150 nA at 9600 baud/s
- DMA support
- Valid wake-up packet



Baud rate

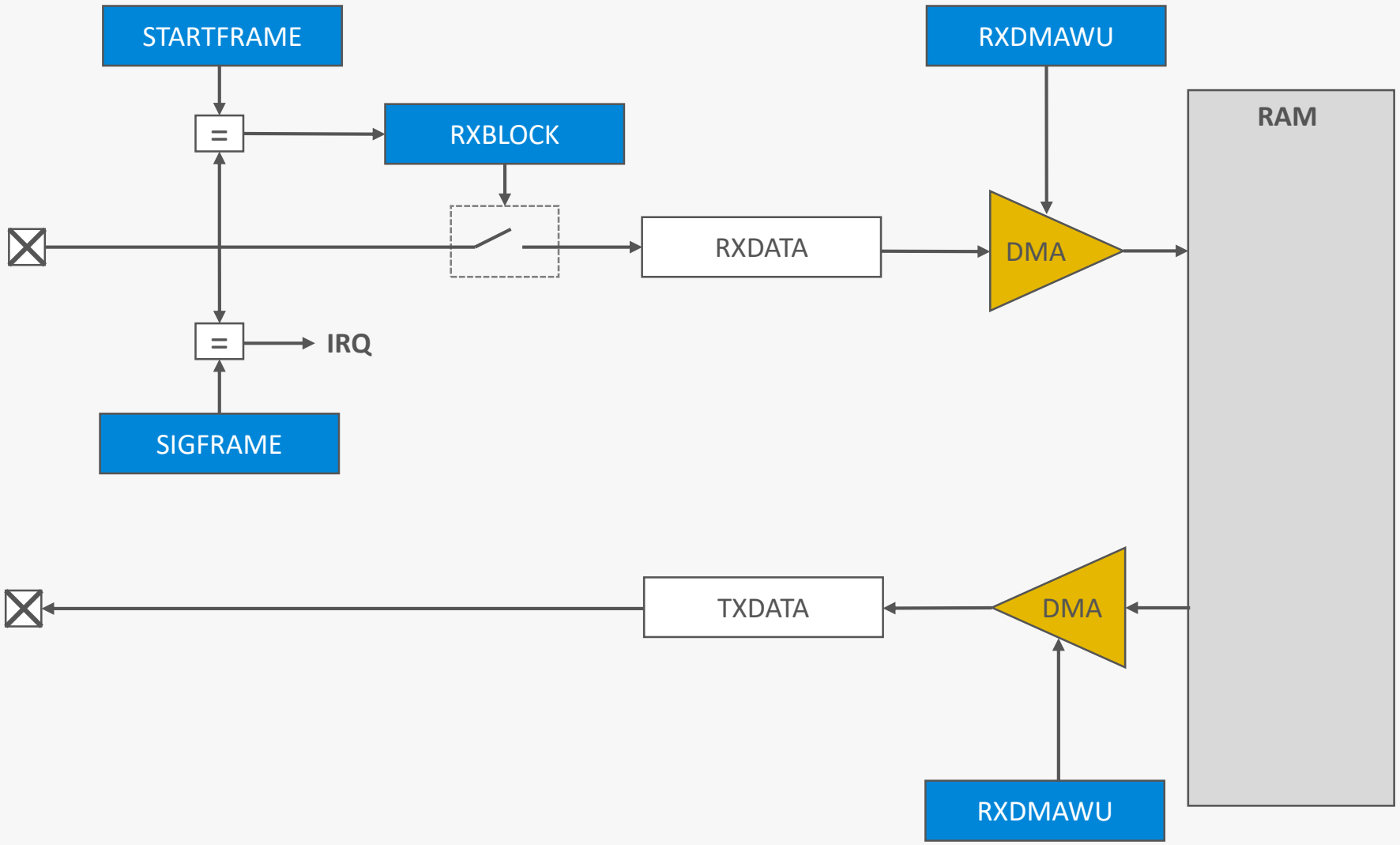
LEUART CLKDIV Equation

$$LEUARTn\_CLKDIV = 256 \times (fLEUARTn/br_{DESIRED} - 1)$$

Desired baud rate [baud/s]	LEUARTn_CLKDIV	LEUARTn_CLKDIV/256	Actual baud rate [baud/s]	Error [%]
300	27704	108,21875	300,0217	0,01
600	13728	53,625	599,8719	-0,02
1200	6736	26,3125	1199,744	-0,02
2400	3240	12,65625	2399,487	-0,02
4800	1488	5,8125	4809,982	0,21
9600	616	2,40625	9619,963	0,21

256 x (32768 / 9600 – 1) = 617.813333

LEUART + DMA in EM2

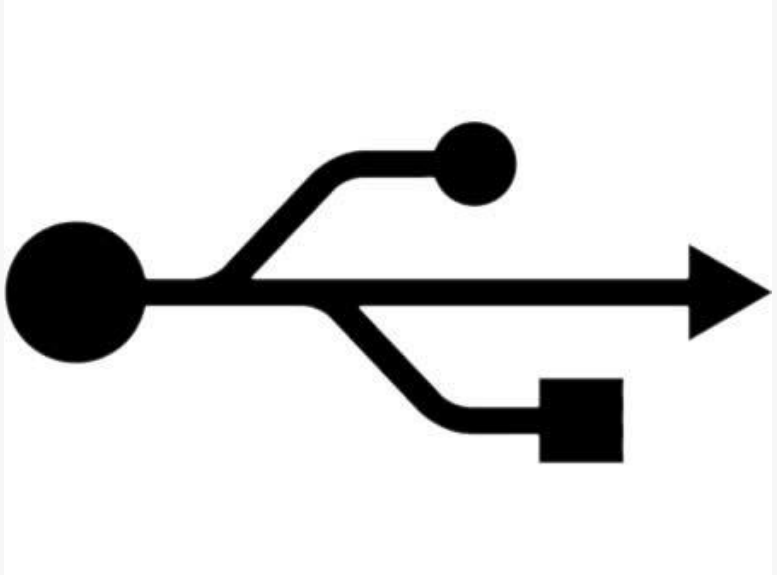


USB

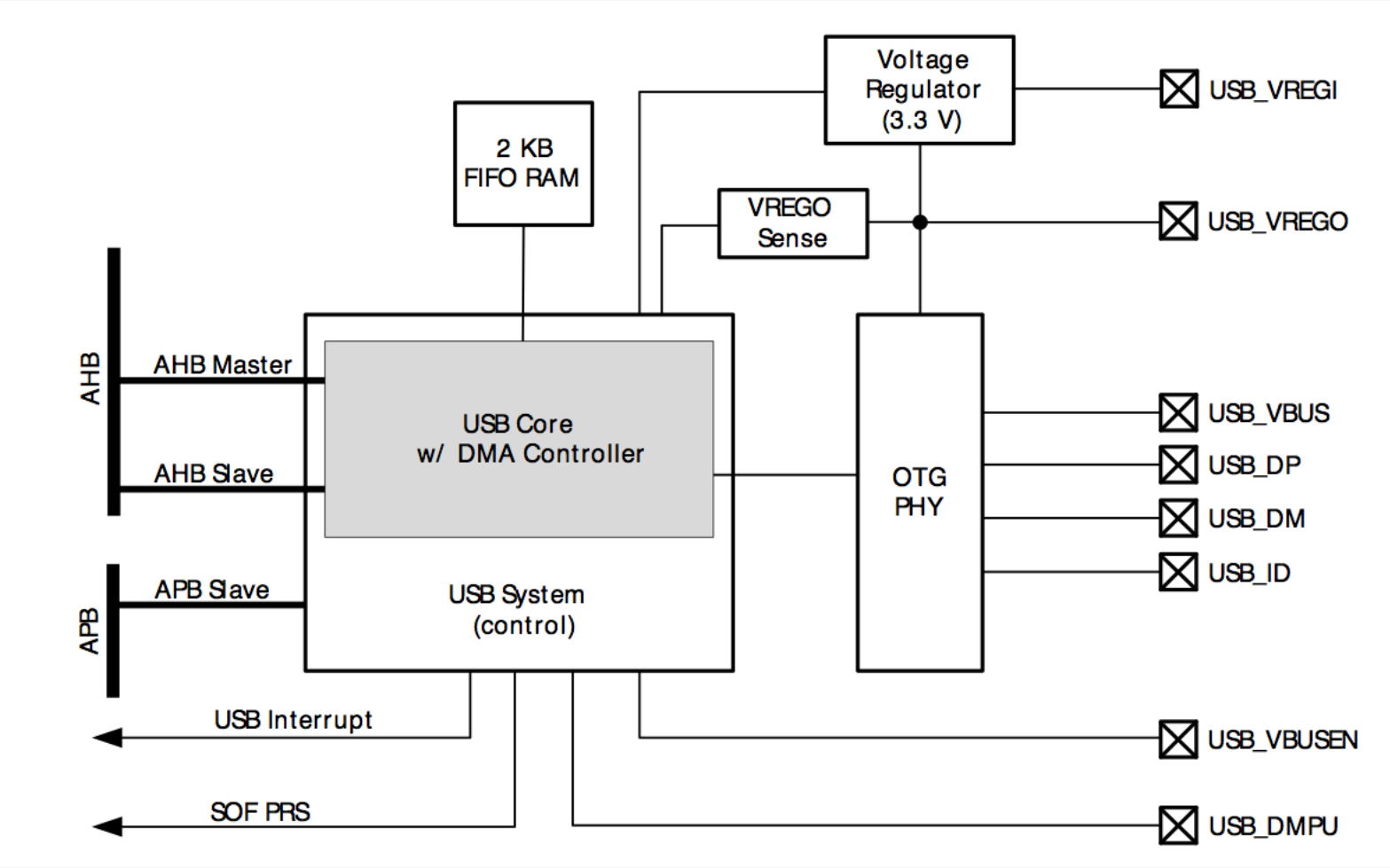
Universal Serial Bus Controller

USB Highlights

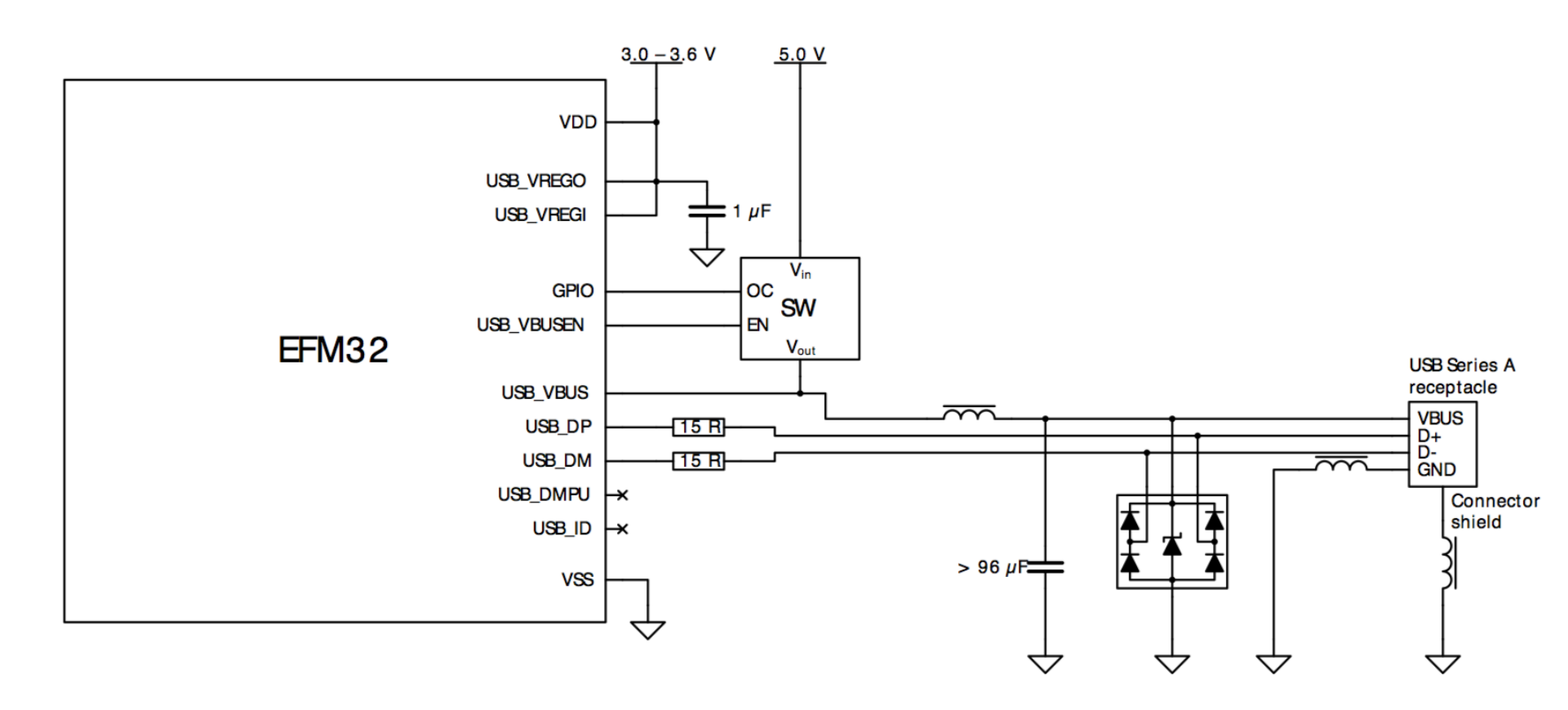
- USB 2.0 compliant
- Support for USB Device and Host
- Full speed (12 Mbit/s)
- 14 endpoints (2 KB buffers)
- Integrated 3.3V regulator (up to 100 mA)
- Dedicated DMA for USB
- Pre-programmed USB device bootloader
- Free stack in Simplicity Studio
  - Mass Storage Host/Device
  - Human Interface Host/Device
  - Vendor Unique Device
  - Communication Class Device (USB-to-RS232)



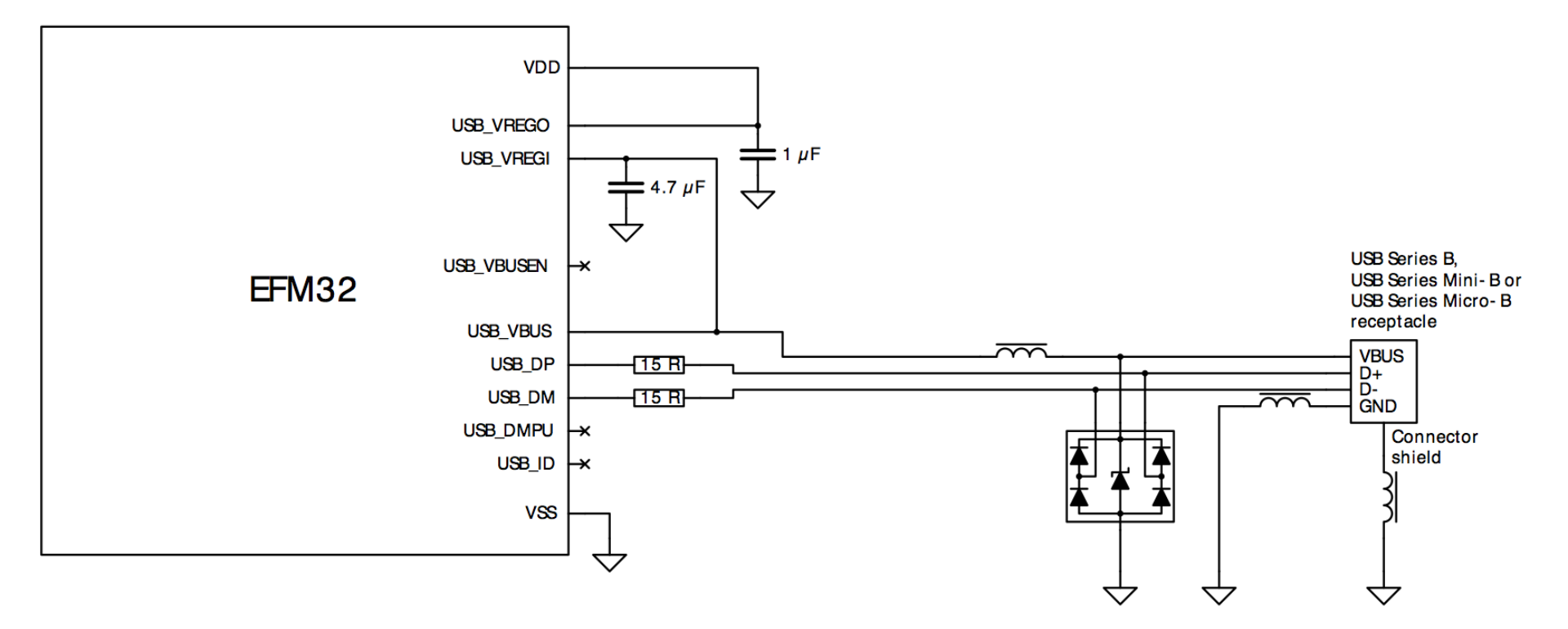
USB Overview



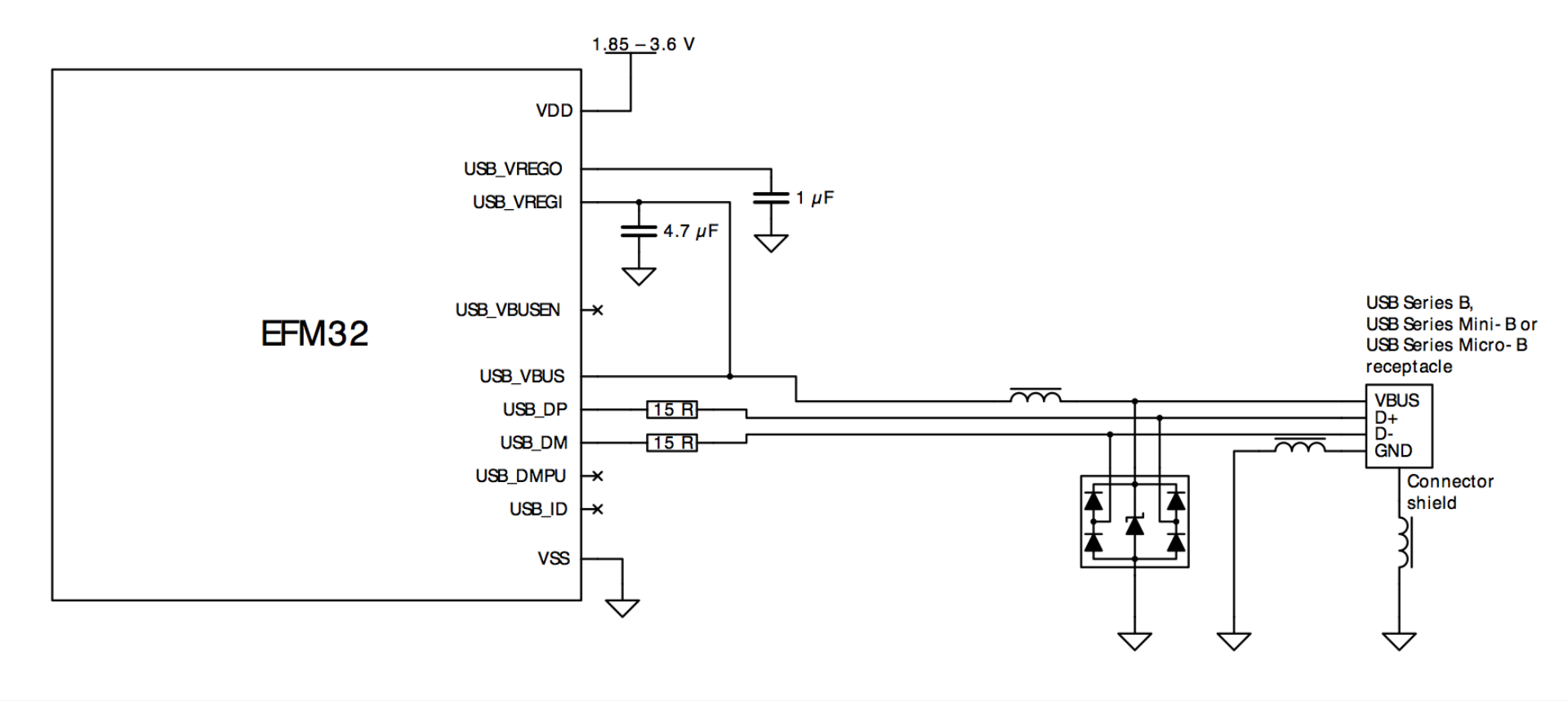
USB Host


3.0 – 3.6 V
5.0 V
VDD
USB_VREGO
USB_VREGI
1 µF
GPIO
USB_VBUSEN
OC
EN
SW
Vin
Vout
EFM32
USB_VBUS
USB_DP
USB_DM
USB_DMPU
USB_ID
VSS
15 R
15 R
> 96 µF
USB Series A receptacle
VBUS
D+
D-
GND
Connector shield

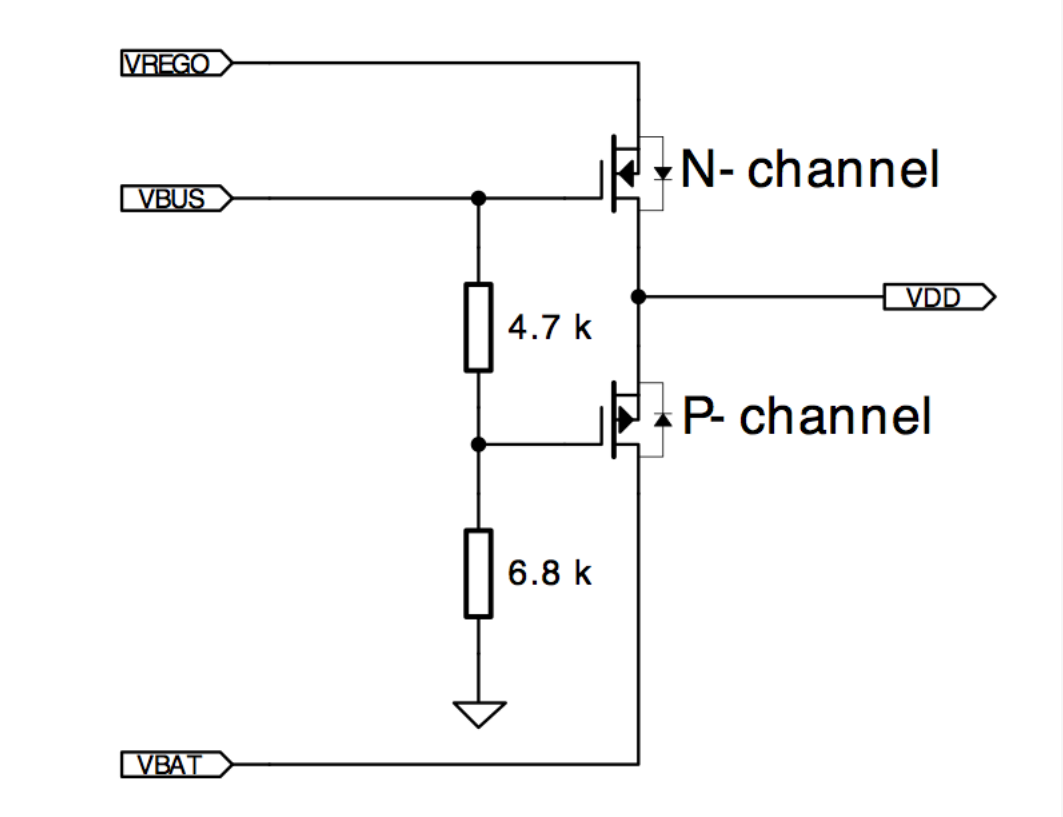
Bus Powered Device



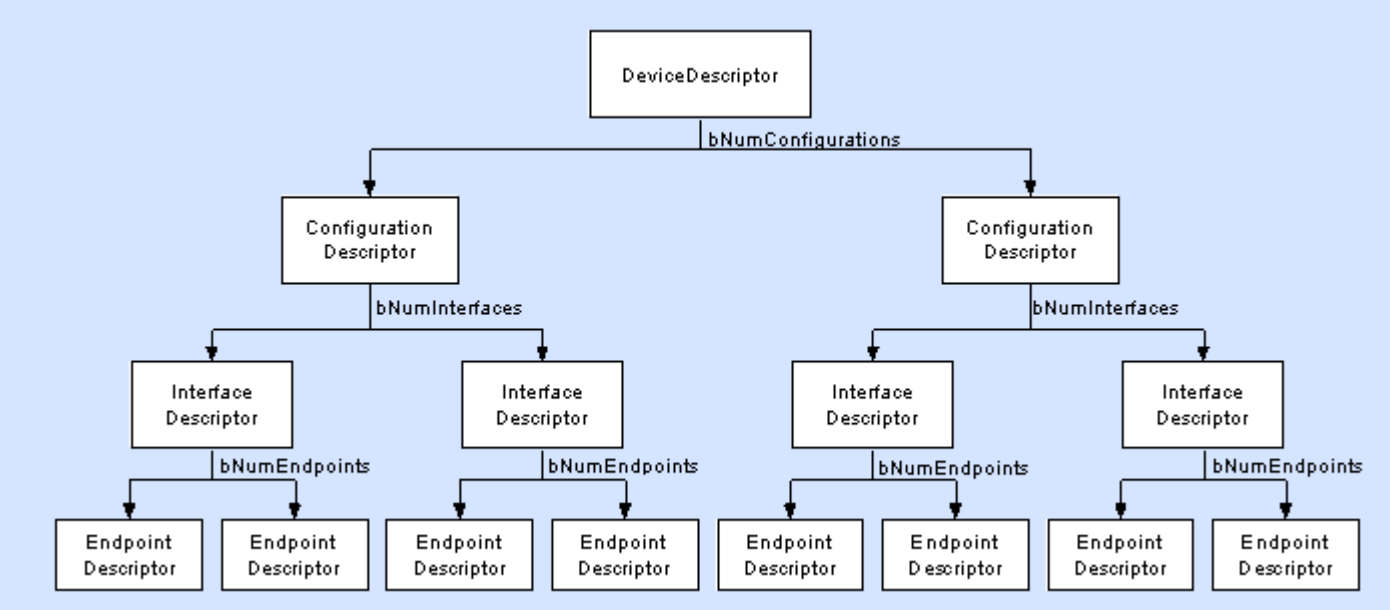
Self Powered Device



Power Switch



USB Descriptors



http://www.beyondlogic.org/usbnutshell/usb5.shtml

Device Descriptor

```
EFM32_ALIGN(4)
static const USB_DeviceDescriptor_TypeDef deviceDesc
{
  .bLength            = USB_DEVICE_DESCSIZE,
  .bDescriptorType    = USB_DEVICE_DESCRIPTOR,
  .bcdUSB             = 0x0200,
  .bDeviceClass       = 0,
  .bDeviceSubClass    = 0,
  .bDeviceProtocol    = 0,
  .bMaxPacketSize0    = USB_EP0_SIZE,
  .idVendor           = 0x2544,
  .idProduct          = 0x0002,
  .bcdDevice          = 0x0000,
  .iManufacturer      = 1,
  .iProduct           = 2,
  .iSerialNumber      = 3,
  .bNumConfigurations = 1
};
```

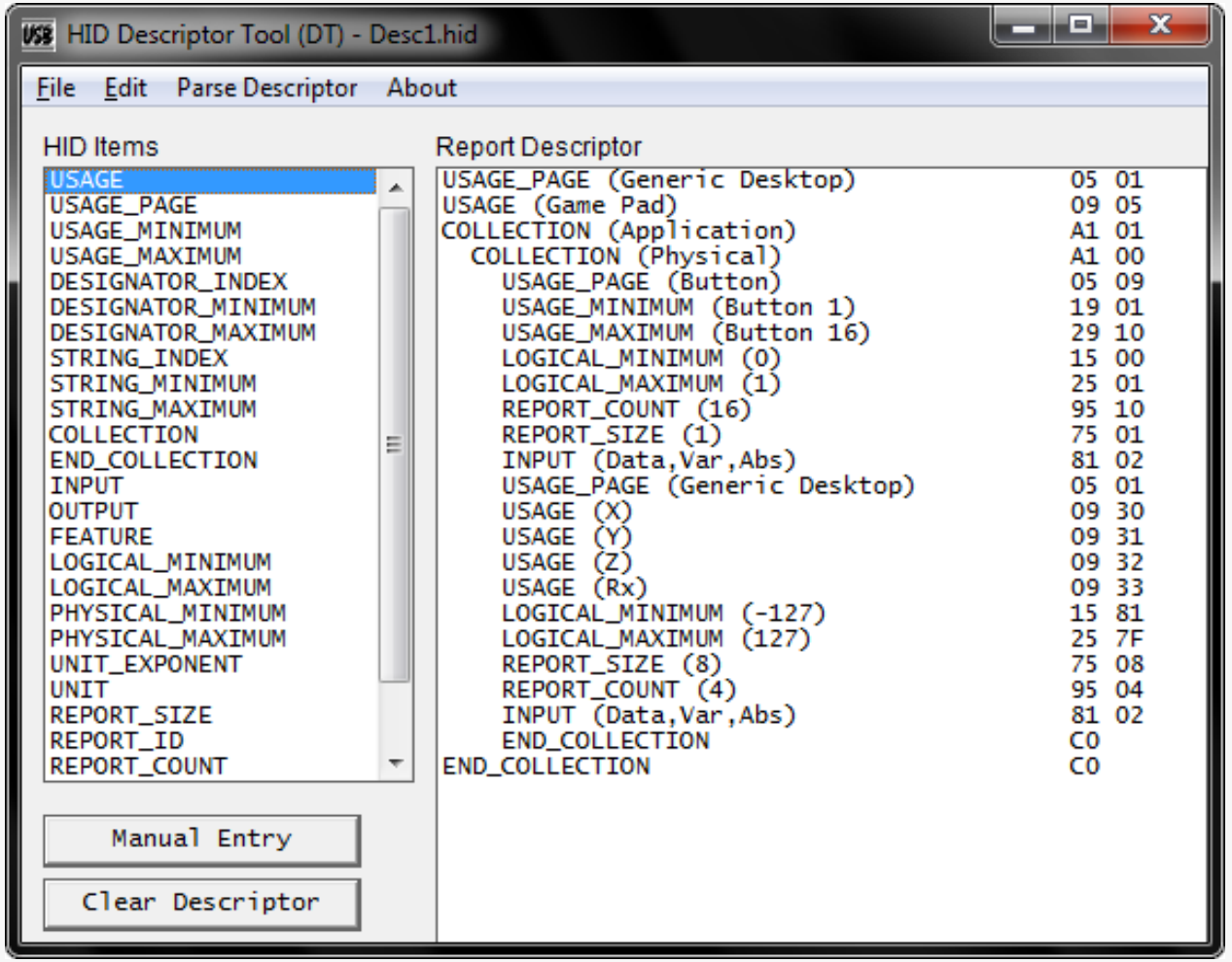
Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of the Descriptor in Bytes (18 bytes)
1	bDescriptorType	1	Constant	Device Descriptor (0x01)
2	bcdUSB	2	BCD	USB Specification Number which device complies too.
4	bDeviceClass	1	Class	Class Code (Assigned by USB Org) If equal to Zero, each interface specifies it's own class code If equal to 0xFF, the class code is vendor specified. Otherwise field is valid Class Code.
5	bDeviceSubClass	1	SubClass	Subclass Code (Assigned by USB Org)
6	bDeviceProtocol	1	Protocol	Protocol Code (Assigned by USB Org)
7	bMaxPacketSize	1	Number	Maximum Packet Size for Zero Endpoint. Valid Sizes are 8, 16, 32, 64
8	idVendor	2	ID	Vendor ID (Assigned by USB Org)
10	idProduct	2	ID	Product ID (Assigned by Manufacturer)
12	bcdDevice	2	BCD	Device Release Number
14	iManufacturer	1	Index	Index of Manufacturer String Descriptor
15	iProduct	1	Index	Index of Product String Descriptor
16	iSerialNumber	1	Index	Index of Serial Number String Descriptor
17	bNumConfigurations	1	Integer	Number of Possible Configurations

Configuration Descriptor

```
EFM32_ALIGN(4)
static const uint8_t configDesc[] __attribute__ ((aligned(4)))=
{
  /*** Configuration descriptor ***/
  USB_CONFIG_DESCSIZE,    /* bLength                                */
  USB_CONFIG_DESCRIPTOR,  /* bDescriptorType                        */

  USB_CONFIG_DESCSIZE +   /* wTotalLength (LSB)                     */
  USB_INTERFACE_DESCSIZE +
  USB_HID_DESCSIZE +
  (USB_ENDPOINT_DESCSIZE * NUM_EP_USED),

  (USB_CONFIG_DESCSIZE +  /* wTotalLength (MSB)                     */
  USB_INTERFACE_DESCSIZE +
  USB_HID_DESCSIZE +
  (USB_ENDPOINT_DESCSIZE * NUM_EP_USED))>>8,

  1,                      /* bNumInterfaces                         */
  1,                      /* bConfigurationValue                    */
  0,                      /* iConfiguration                         */

#if defined(BUSPOWERED)
  CONFIG_DESC_BM_RESERVED_D7,    /* bmAttrib: Bus powered           */
#else
  CONFIG_DESC_BM_RESERVED_D7 |   /* bmAttrib: Self powered          */
  CONFIG_DESC_BM_SELFPOWERED,
#endif

  CONFIG_DESC_MAXPOWER_mA( 100 ), /* bMaxPower: 100 mA             */
```

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor in Bytes
1	bDescriptorType	1	Constant	Configuration Descriptor (0x02)
2	wTotalLength	2	Number	Total length in bytes of data returned
4	bNumInterfaces	1	Number	Number of Interfaces
5	bConfigurationValue	1	Number	Value to use as an argument to select this configuration
6	iConfiguration	1	Index	Index of String Descriptor describing this configuration
7	bmAttributes	1	Bitmap	D7 Reserved, set to 1. (USB 1.0 Bus Powered) D6 Self Powered D5 Remote Wakeup D4..0 Reserved, set to 0.
8	bMaxPower	1	mA	Maximum Power Consumption in 2mA units

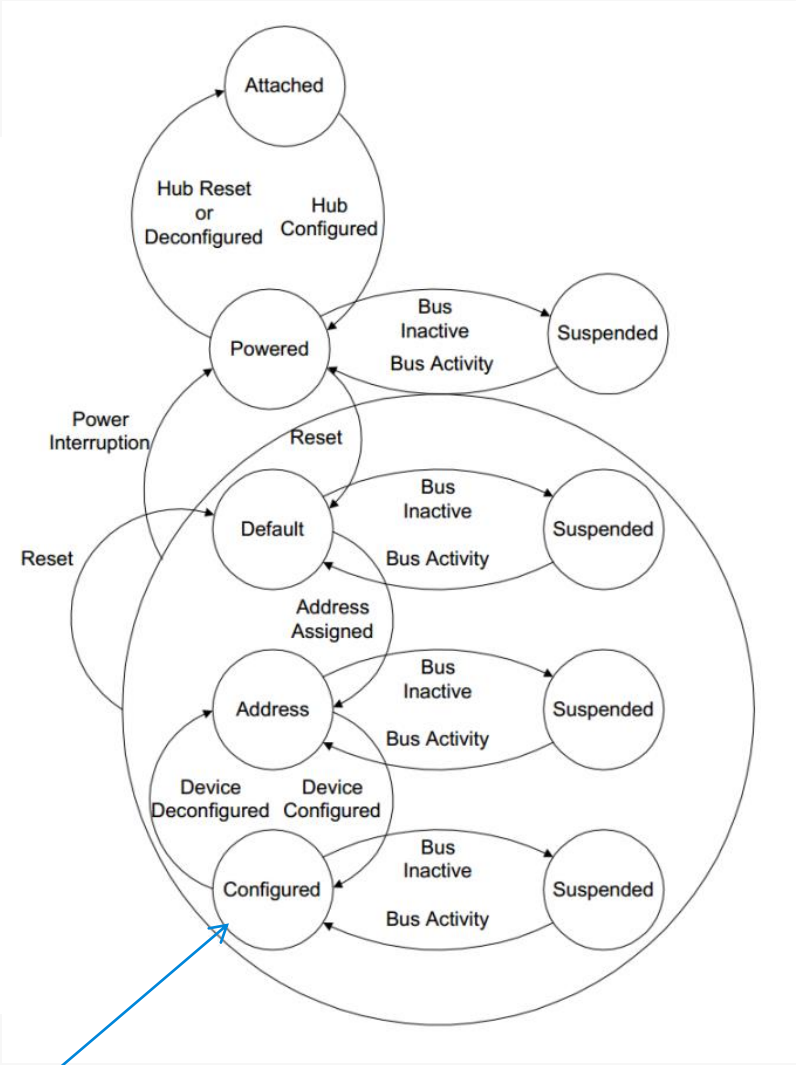
HID Descriptors



USB State Machine

Called for every state change

```
static void StateChange(USBD_State_TypeDef oldState,
                        USBD_State_TypeDef newState)
{
  if (newState == USBD_STATE_CONFIGURED)
  {
    /* We have been configured, start HID functionality ! */
    if (oldState != USBD_STATE_SUSPENDED)   /* Resume ?   */
    {
      keySeqNo        = 0;
      keyPushed       = false;
      lastReportIndex = -1;
      idleRate        = DEFAULT_IDLE_RATE / 4;     /* Unit is 4 millisecond.
      GPIO_PinOutSet(ACTIVITY_LED);
      QueueInit();
    }
    USBTIMER_Start( SCAN_TIMER, SCAN_RATE, ScanTimeout);
    if ( idleRate )
    {
      USBTIMER_Start( IDLE_TIMER, idleRate * 4,  IdleTimeout);
    }
  }

  else if ((oldState == USBD_STATE_CONFIGURED) &&
           (newState != USBD_STATE_SUSPENDED))
  {
    /* We have been de-configured, stop HID functionality */
    USBTIMER_Stop(SCAN_TIMER);
    USBTIMER_Stop(IDLE_TIMER);
    GPIO_PinOutClear(ACTIVITY_LED);
  }
```



Communication only possible in this state

Transmit Functions and Buffers

- API functions only initiate transfer
  - USBD_Read()
  - USBD_Write()
- Application receives callback when transfer is complete (or fails)

- All read/write buffers must be WORD (32-bit) aligned
- Buffer sizes should be rounded up to the next WORD boundary
- Buffers must be statically allocated, NOT on the stack
- Macros exist
  - STATIC_UBUF()
  - UBUF()
- Receive buffers should be rounded up to nearest maxpacket size IF host will send more data than device expects

Callbacks

- Application receives callbacks when
  - Transfers complete or fail
  - State changes
  - Connection established or lost
  - Control message received
  - Start-of-Frame received
  - Line reset received
- Only callbacks that are used needs to be implemented

Using EM2

- USB can run off LF clock when suspended or disconnected
- Application can use `USBD_SafeToEnterEM2()` to determine when it is safe to enter EM2
- Optionally the stack can enter EM2 automatically (SLEEPONEXIT)

```
/**************************************************************
 * Power saving configuration. Select low frequency
 * clock and power saving mode.
 *************************************************************/

/* Select the clock used when USB is in low power mode */
#define USB_USBC_32kHz_CLK   USB_USBC_32kHz_CLK_LFXO

/* Select the power saving mode. Enter power save on Suspend and
 * when losing power on VBUS. Let the application handle when to
 * enter EM2 */
#define USB_PWRSAVE_MODE (USB_PWRSAVE_MODE_ONSUSPEND | USB_PWRSAVE_MODE_ONVBUSOFF)
```

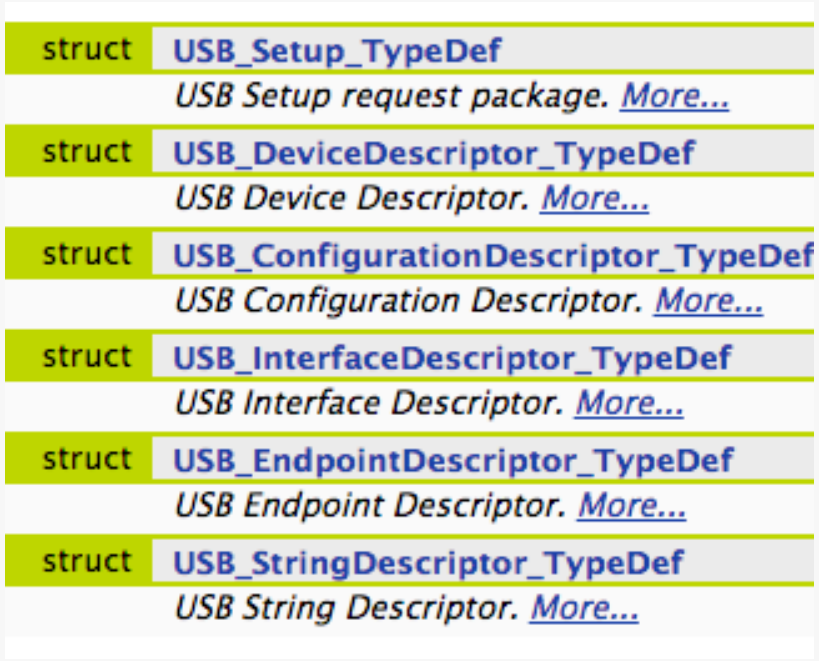
Configuration

- USB stack is configured in
  - usbconfig.h
  - descriptors.h
- USB Stack can output debug information over UART
  - Uses retargetio.c and retargetserial.c

```
/* Enable debug output from the stack */
#define DEBUG_USB_API

/* Enable printf calls in stack */
#define USB_USE_PRINTF

/* Function declaration for the low-level printing of a
 * character. This function must be implemented by the
 * application. */
int RETARGET_WriteChar(char c);
#define USER_PUTCHAR  RETARGET_WriteChar
```

Documentation

- Doxygen usb documentation
- AN0065 EFM32 as USB Device
- AN801 EFM32 as USB Host
- AN0046 USB Hardware Guidelines
- AN0042 USB-UART Bootloader
- Device and Host examples in SS
  - usbdcdc, usbhidkbd, usbdmsd …



Demo

Demo USB HID Keyboard

Hands-on LEUART

- Open training_leuart
- Set TFT in ‘EFM’ mode
- Fill in the missing code
  - Missing code is marked with «TODO»
- Connect USB-UART adapter
  - TX (Orange) to PC7
  - GND (Black) to GND (pin zero on any port header)
- Verify that code works
  - Only bytes from start frame to CR (enter) is printed to TFT
- Missing code

1. Enable DMA wakeup by LEUART on RX data
2. Define a start frame (byte)
3. Enable RX unblock on start frame
4. Enable RX block (two places)