# Atmel SAMD21 MCU/MO Pro

Serial Communication, Analog-to-Digital, and Digital-to-Analog Converter

# Outline

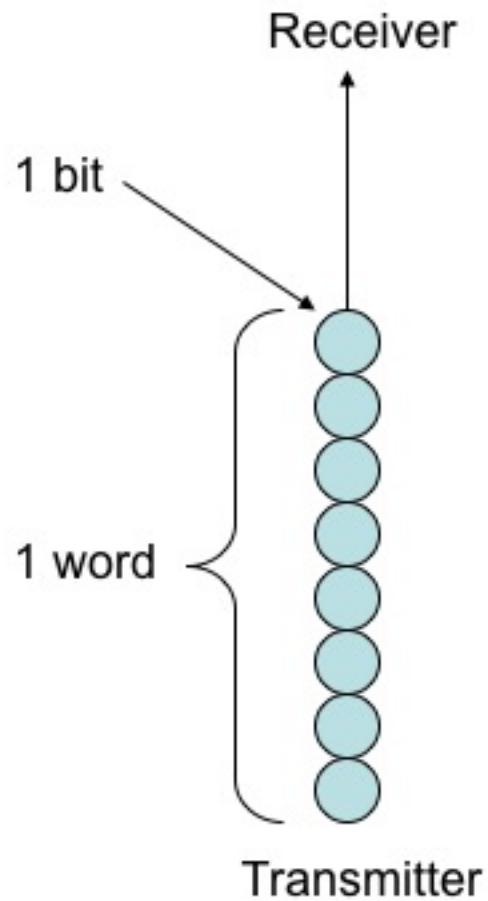
- Serial Communication
- Analog-to-Digital Converter
- Digital-to-Analog Converter

# Serial Communication (Basic Concepts)

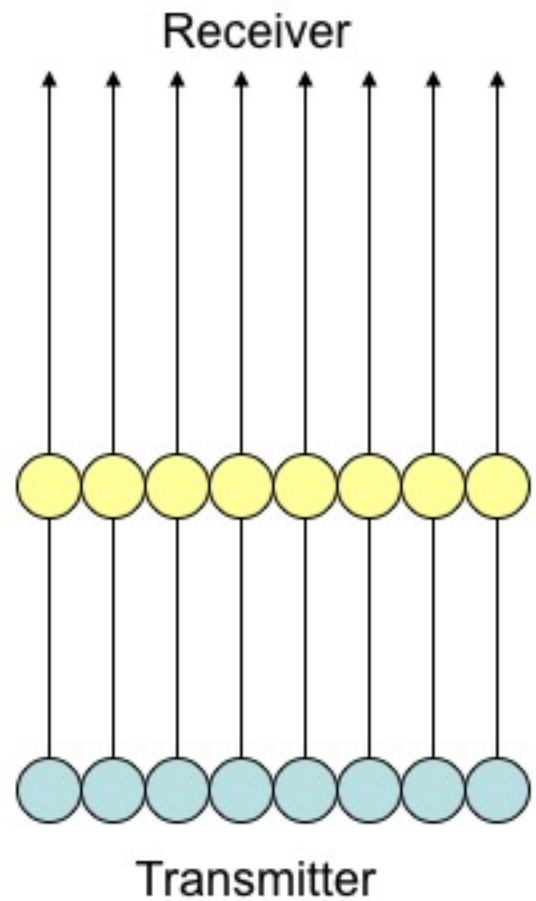
- Slides are adopted from
- http://ume.gatech.edu/mechatronics_course/

# Transmission Illustration

## Serial




## Parallel



# Data Transmission

| | Serial | Parallel |
|---|---|---|
| **Cost** | Cheap | Expensive |
| **Speed** | Slow | Fast |
| **Transmission Amount** | Single bit | 8 bits (8 data lines)<br>Transmitter & Receiver |
| **Transmission Lines** | One line to transmit<br>one to receive | 8 lines for simultaneous transmission |
| **Transmission Distance** | Long distance | Short distance<br>(synchronization) |
| **Example** | Modem | Printer Connection |

# Serial Communications

- Synchronous
  - Synchronous Peripheral Interface (SPI)
- Asynchronous
  - Serial Communication Interface (SCI)

# Synchrononous - SPI

- Constant transmission of data
- Clocks of Transmitter and Receiver must be synchronized
- No safeguard against error or noise
- Data rates depend on clock rates
- Flexible to communication with peripheral devices
  - LCD drivers, A/D converter, other microprocessors
- Simultaneously transmits and receives data
  - Transmission line, Receiving line, and Ground

# Asynchronous - SCI

- Transmission of data through “words”
- Continuous transmission unnecessary
- Built-in safeguards against noise and error
- Transmitter and Receiver operate independently
- Requires start and stop bit for each byte of data
  - Sends constant ‘1’ for idle
  - Sends a ‘0’ for start and “1” for stop bits
- Very reliable data reception

# Bit Types

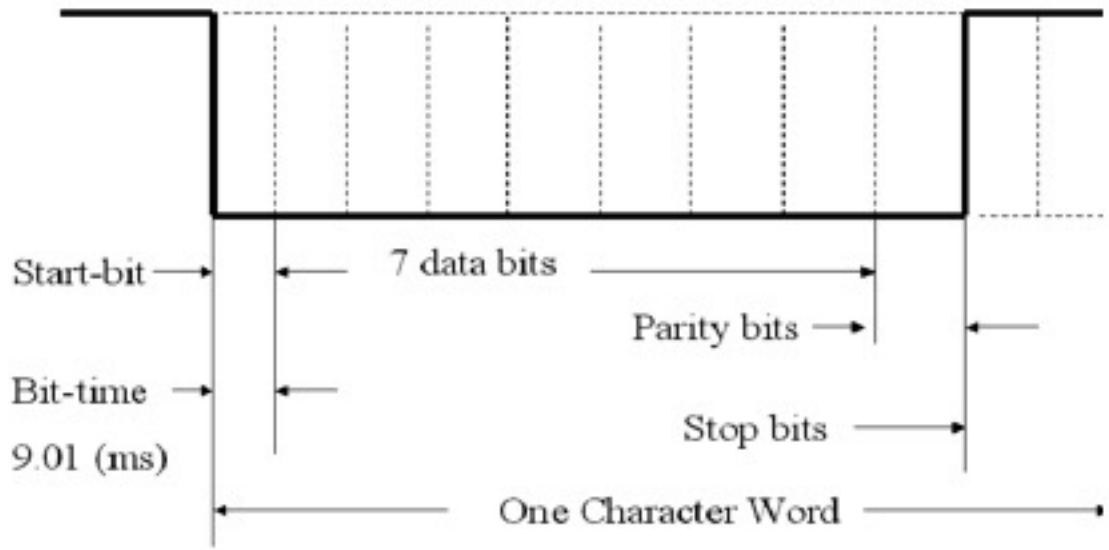



Figure 1. Role of stop, start and parity bits.

- **Start Bit –**
  - **Signals the transmission of a word.**
  - **Transition from "1" to "0". ("Mark-to-space")**
  - **First bit to be transmitted.**

# Bit Types (Cont)

- Stop Bits –
  - Bit at the end of a data word.
  - Bit set to high “1”.
  - Indicates the end of a word.

- Data bits –
  - Data bits to be transmitted.
  - Sender and receiver have to agree in the number of data bits. (Usually 8 or 9)
  - Least significant bit is sent first.
  - Can be low or high.

# Bit Types (Cont)

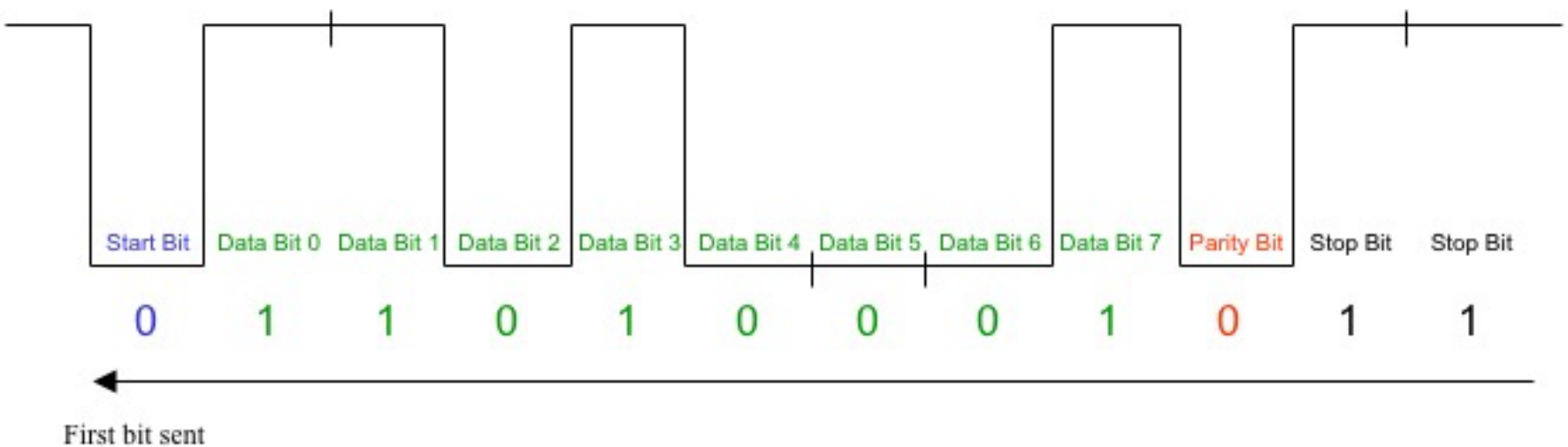
- **Parity bit –**
  - Works as an error check.
  - There are two types: *odd* and *even*
    - Even: if number of 1's in the data word is even.
    - Odd: if number of 1's in the data word is odd.
  - Bit after the data bits and before the stop bit.
  - Can prevent single noise signal, but does not recognize when two bits are altered by noise.
  - Used to prevent noise.

# Asynchronous Transmission

**Example 1:**

**Send $8B_{16}$ with one start bit, 8 data bits, even parity, and two stop bits.**

- **$8B_{16}=10001011_2$**



# Asynchronous Transmission

**Example 2:**

**Send $8C_{16}$ with one start bit, 8 data bits, even parity, and two stop bits.**

- **$8C_{16}=10001100_2$**

# Asynchronous Transmission

**Example 3:**

**Send $8C_{16}$ with one start bit, 8 data bits, *odd* parity, and two stop bits.**

- **$8C_{16}=10001100_2$**

# Baud Rate vs. Bit Rate

- Baud rate: number of total information bits transmitted per second (includes start, data, parity and stop bits)
- Bit rate: number of data bits only transmitted per second
- So for us, Baud rate > Bit rate

Note: in modems, multiple transmission voltage levels are used, so multiple bits are encoded with each signal, meaning bit rates can be greater than baud rates

# Baud Rate Example

**What is the bit rate for a 2400 baud rate using a parity bit and two stop bits per data word?**

$$\text{bit rate} = \text{baud rate}\ \frac{8\ \text{data bits/word}}{12\ \text{total bits/word}}$$

$$\text{bit rate} = 2400(2/3) = 1600\ \text{bps}$$

**For HC11 with one start, 8 data, and one stop:**

$$\text{bit rate} = \text{baud rate}\ \frac{8\ \text{data bits/word}}{10\ \text{total bits/word}}$$

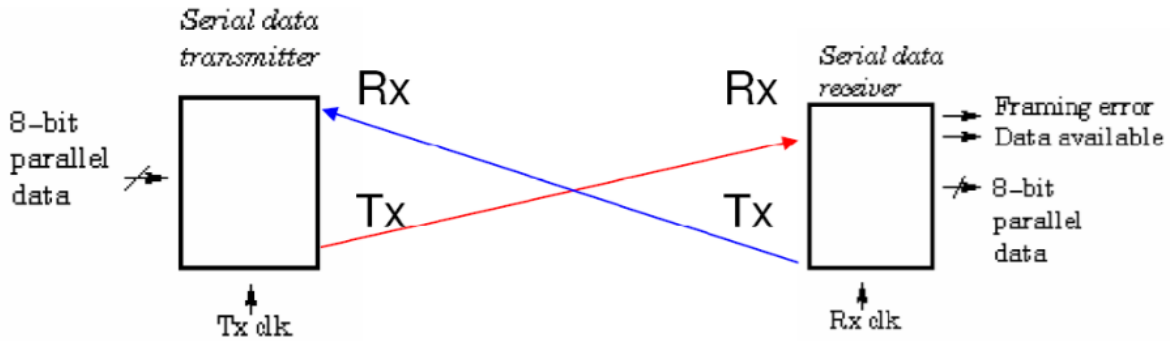
$$\text{bit rate} = 2400(4/5) = 1920\ \text{bps}$$

# Serial Communication (UART, SPI, and $I^2C$)

- Slides are adopted from
- http://www.comm.pub.ro/dicm/C7_Serial_Bus.pdf

# UART

- UART interface has two signals
  - Rx – Receive
  - Tx – Transmit

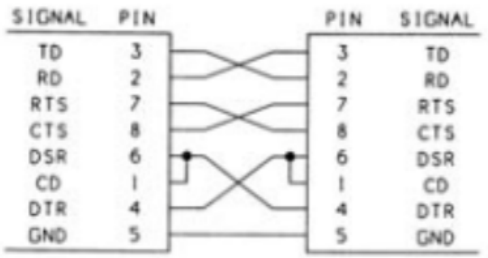



- **USART** has an additional signal, **XCK** (external clock) that can be input or output, and will synchronize the transmission and reception

## RS232

- More than UART: ***handshake***
  - Flow control
  - Equipment status
- RTS - Request to send,
- CTS - Clear to send
- DSR - Data set ready,
- DTR - Data terminal ready
- DCD - Data carrier detect
- RI - Ring indicator


No Handshake

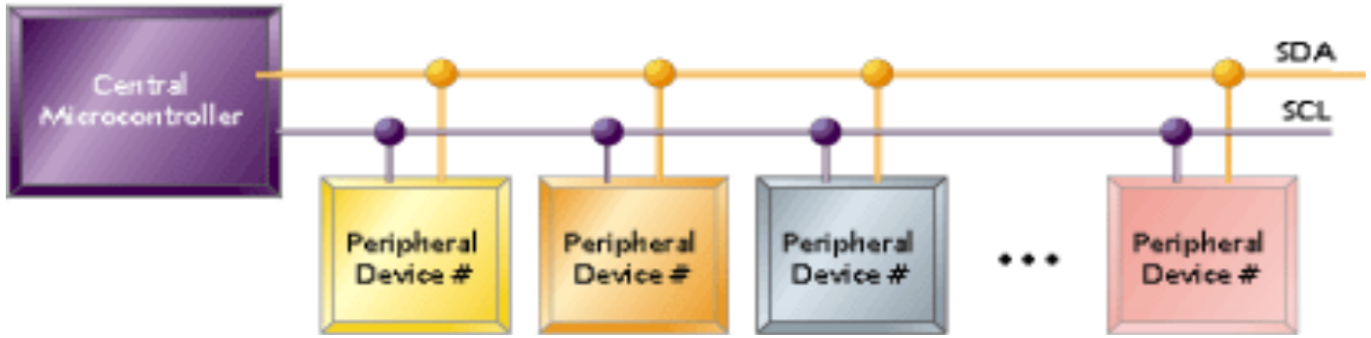



Full Handshake

# $I^2C$

- “Inter-integrated circuit” bus
- Developed for television by Philips Semiconductor, 1980
- $I^2C$ Devices: processors, EEPROMs, sensors, real-time clocks
- Used as a control interface
- $I^2C$ devices can also have separate data interface (digital TV tuners, video decoders, audio processors ...)
- 3 speed $I^2C$ :
  - Slow (under 100 Kbps)
  - Fast (400 Kbps)
  - High-speed (3.4 Mbps) – $I^2C$ v.2.0
- Bus length: typical: inside the equipment, <1m, maximum: few meters

# Bus I²C




- 2-wire (TWI) – Serial data (SDA) and Serial clock (SCL)
- Common ground wire (totally 3 wires)
- Half-duplex, synchronous, multi-master
- Without “chip select” or arbitration
- 1 logic: pull-up resistors
- 0 logic: open-collector or open-drain driven
- Equivalent to wired-AND

# I²C Protocol


1. Master sends “start condition” (S) on SDA and clock is generating on SCL
2. Master sends the slave address (7 bits)
3. Master sends bit read/write (R/W)
   - 0 - slave will receive,
   - 1 - slave will transmit
4. The transmitter (slave or master) sends bit ACK
5. The transmitter sends 1 byte of data

# I²C Protocol (cont.)

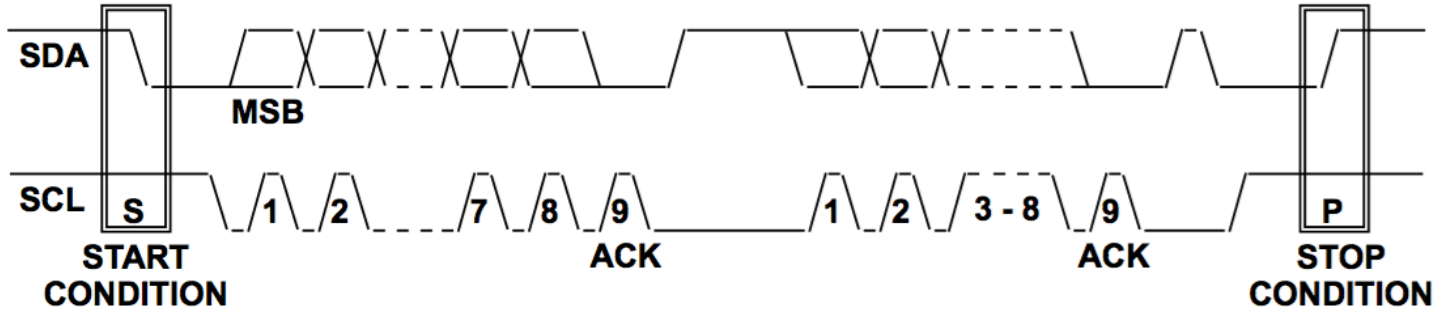

6. The receiver sends an ACK bit for byte received
7. Repeat steps 5 and 6 if more bytes are sent
8. a) for write transaction (master = transmitter) sends stop (P) after the last byte of data
   b) for read transaction (master = receiver), does not send ACK for last byte, just sends stop (P) to signal slaves that transmission ended

# I²C Signals

Sample Signals

- Start (S) – falling edge on SDA when SCL is “1”
- Stop (P) – rising edge on SDA when SCL is “1”
- ACK – the receiver “pull down” SDA to 0 (the transmitter leave SDA=1)
- Data – SDA  when SCL=0; valid transmission when SCL=1

# Example of a data transmission



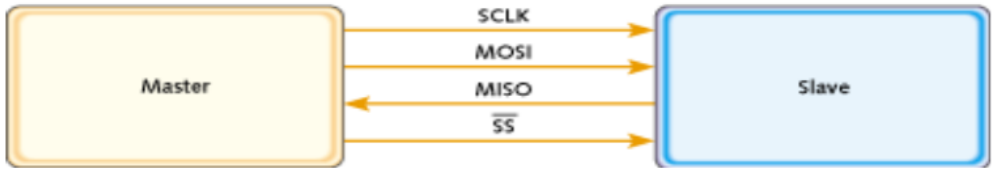
## SPI

- “Serial Peripheral Interface”
- Defined by Motorola for MC68HCxx
- Faster than $I^2C$, (by few Mbps)

Applications:

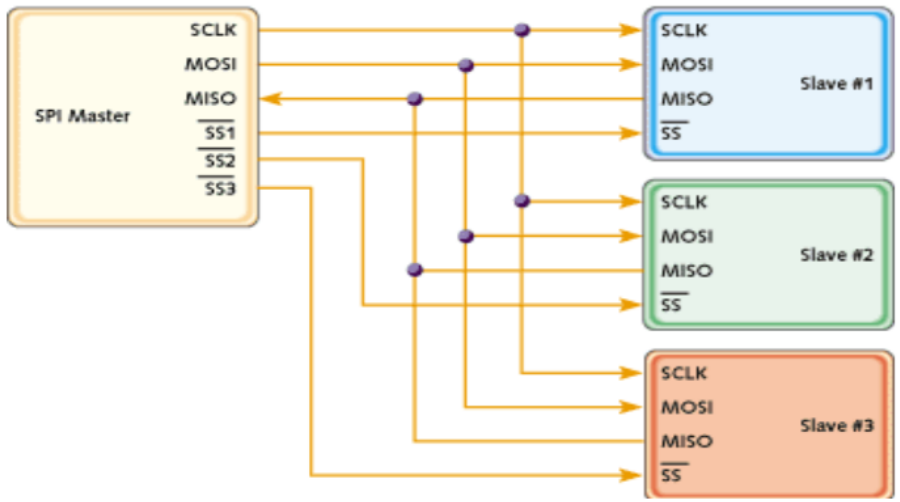
- Like $I^2C$, is used for EEPROM, Flash, real-time clocks...
- Suitable for devices that send continuous flow of data, such as ADC
- Full duplex, suitable for communication between a codec and a signal processor (DSP)

# SPI



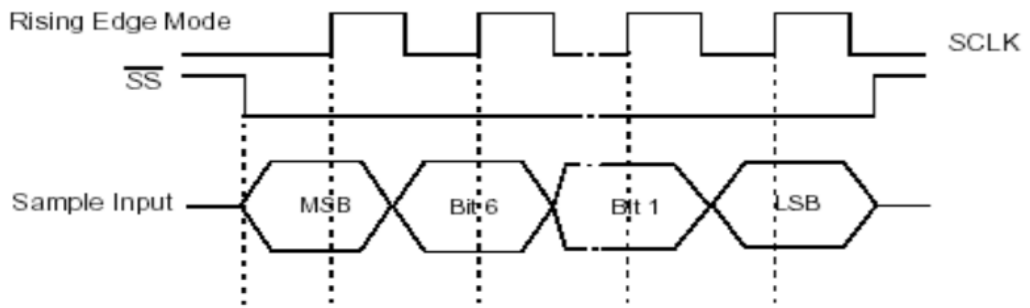

- Synchronous Bus
- Master / slave relationship
- 2 data lines:
  - MOSI – master data output, slave data input
  - MISO – master data input, slave data output
- 2 control lines:
  - SCLK – clock
  - /SS – slave select
- SS means no addressing

# SPI vs. I²C




- For Point-to-point, SPI is more simple and efficient
  - Less overhead than I²C due to no addressing.
- For more slaves, each slave needs a SS line
  - More connections than I²C

# SPI Protocol



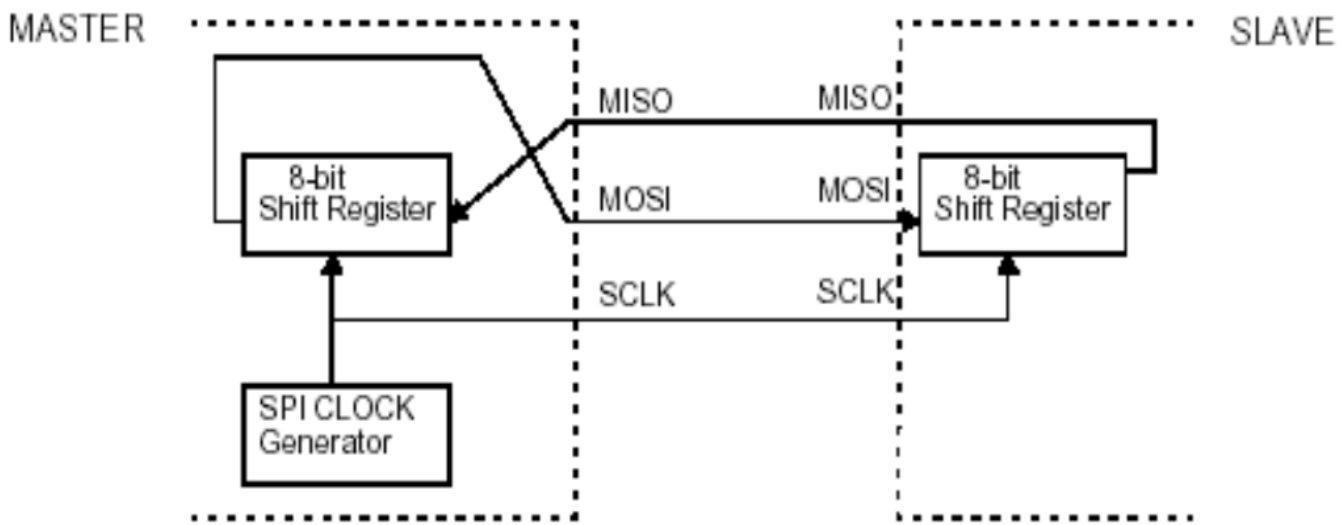

- 2 parameters, Clock Polarity (CPOL) and Clock Phase (CPHA), determine active edge of CLOCK

| CPOL | CPHA | Active edge |
|---|---|---|
| 0 | 0 | Rising |
| 0 | 1 | Falling |
| 1 | 0 | Falling |
| 1 | 1 | Rising |

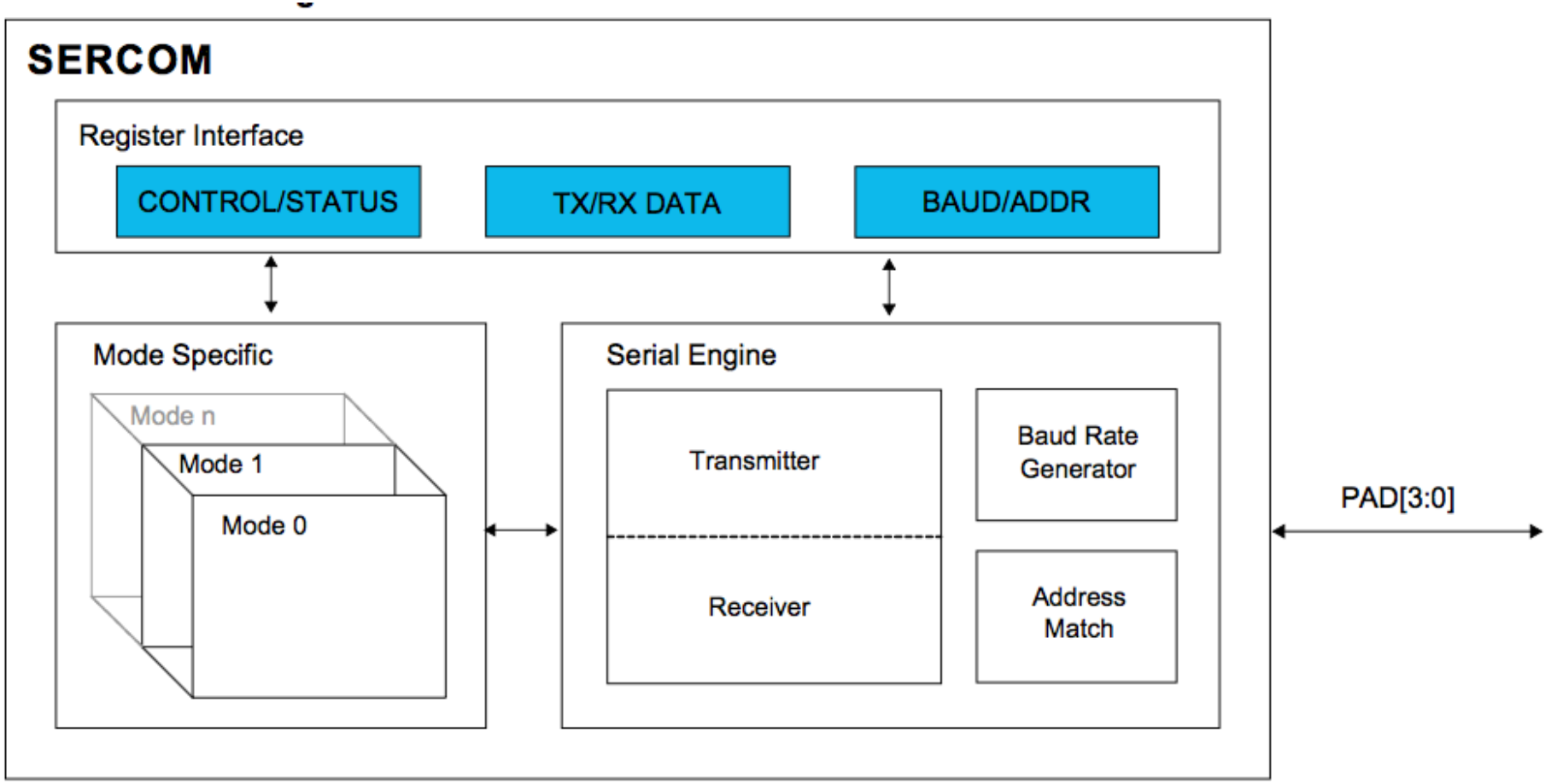
- Master and slave must be configured with the same set of parameters, otherwise can not communicate

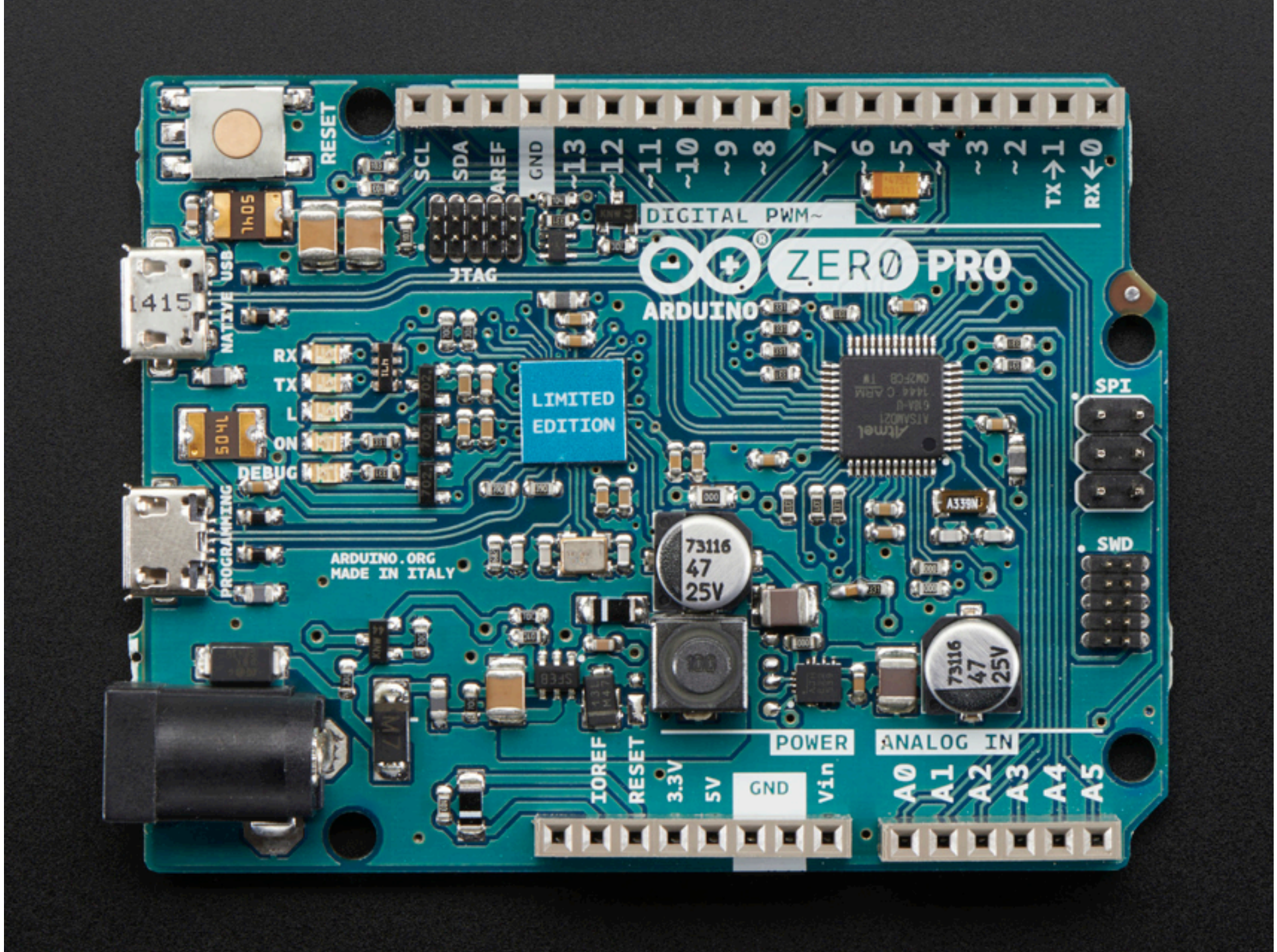
# SPI Protocol (cont.)

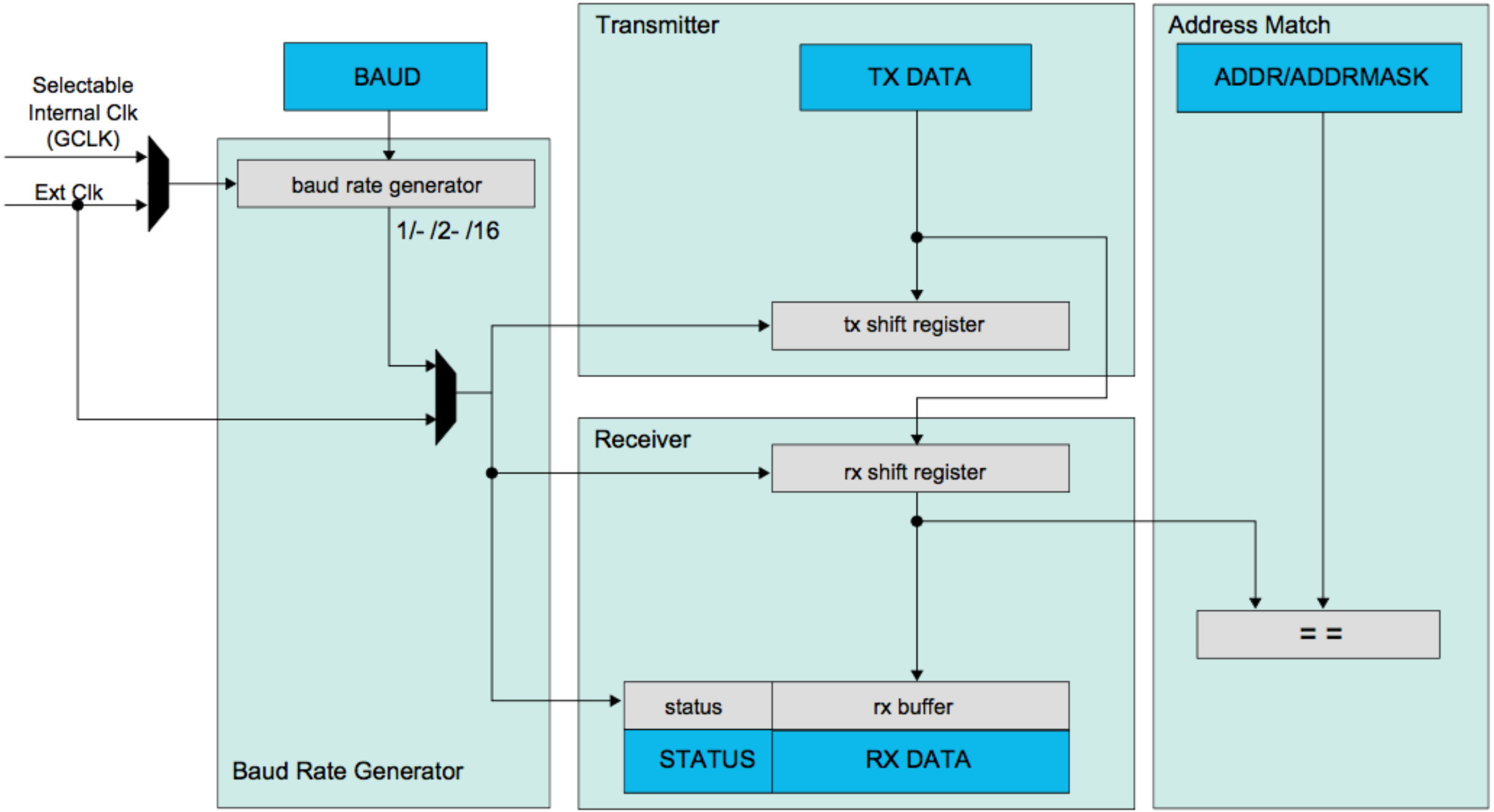
- SPI interface only defines the communication lines and the clock used
- No ACK flow control or ACK mechanisms built in



# Atmel SAMD21 MCU/MO Pro




RESET
SCL
SDA
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TX→1
RX←0
DIGITAL PWM~
JTAG
USB
NATIVE
PROGRAMMING
DEBUG
RX
TX
L
ON
LIMITED
EDITION
ARDUINO
ZERO PRO
ARDUINO.ORG
MADE IN ITALY
SPI
SWD
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
Vin
A0
A1
A2
A3
A4
A5


Transmitter
Address Match
BAUD
TX DATA
ADDR/ADDRMASK
Selectable
Internal Clk
(GCLK)
Ext Clk
baud rate generator
1/- /2- /16
tx shift register
Receiver
rx shift register
= =
status
rx buffer
STATUS
RX DATA
Baud Rate Generator

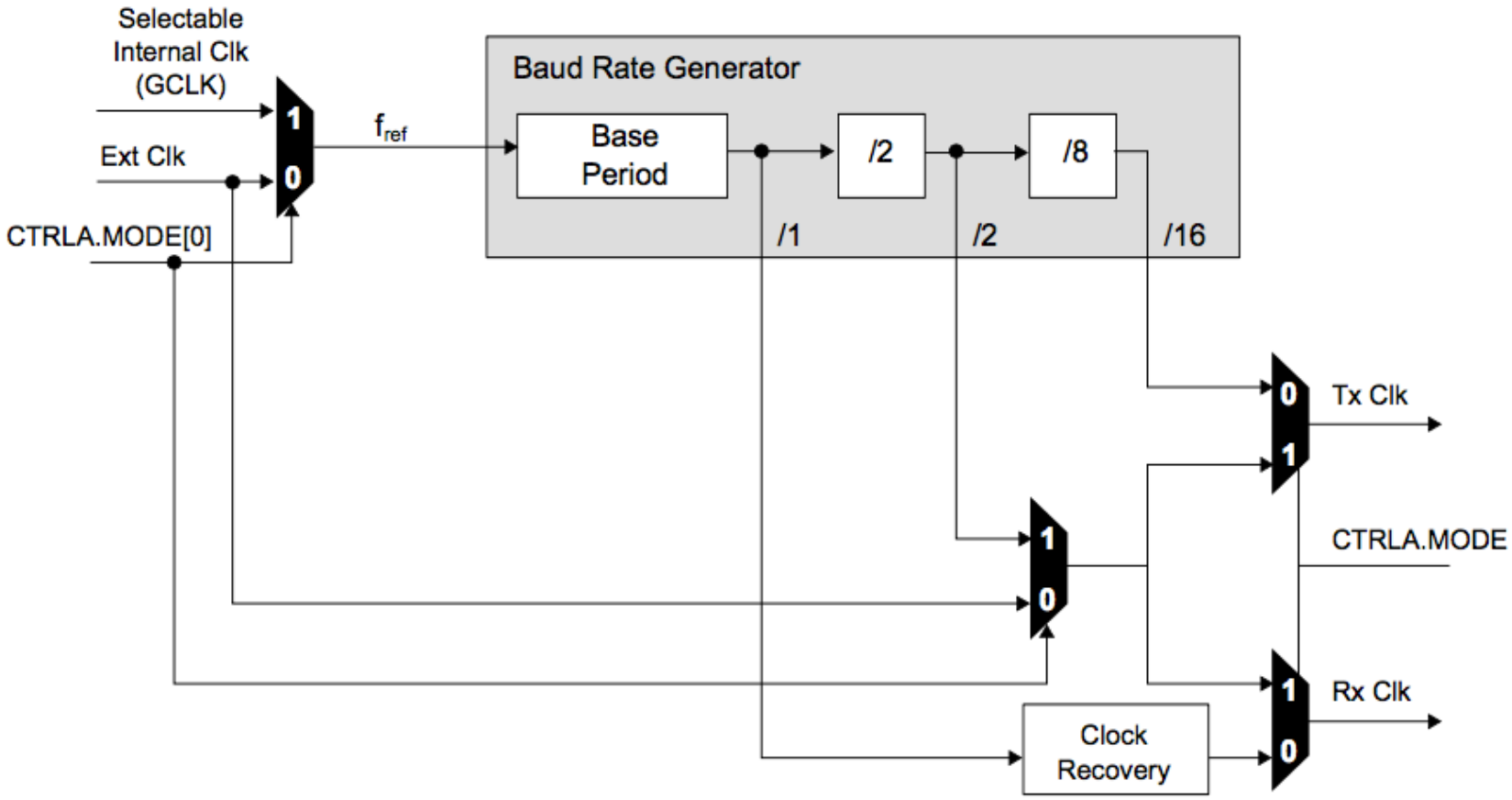
# Basic Operation

- Initialization
  - The SERCOM must be configured to the desired mode by writing to the Operating Mode bits in the Control A register (CTRLA.MODE)
- Enabling, Disabling and Resetting
  - The SERCOM is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The SERCOM is disabled by writing a zero to CTRLA.ENABLE
- Clock Generation – Baud-Rate Generator

# Mode

| CTRLA.MODE | Description |
|---|---|
| 0x0 | USART with external clock |
| 0x1 | USART with internal clock |
| 0x2 | SPI in slave operation |
| 0x3 | SPI in master operation |
| 0x4 | $I^2C$ slave operation |
| 0x5 | $I^2C$ master operation |
| 0x6-0x7 | Reserved |

# Baud-Rate Generator

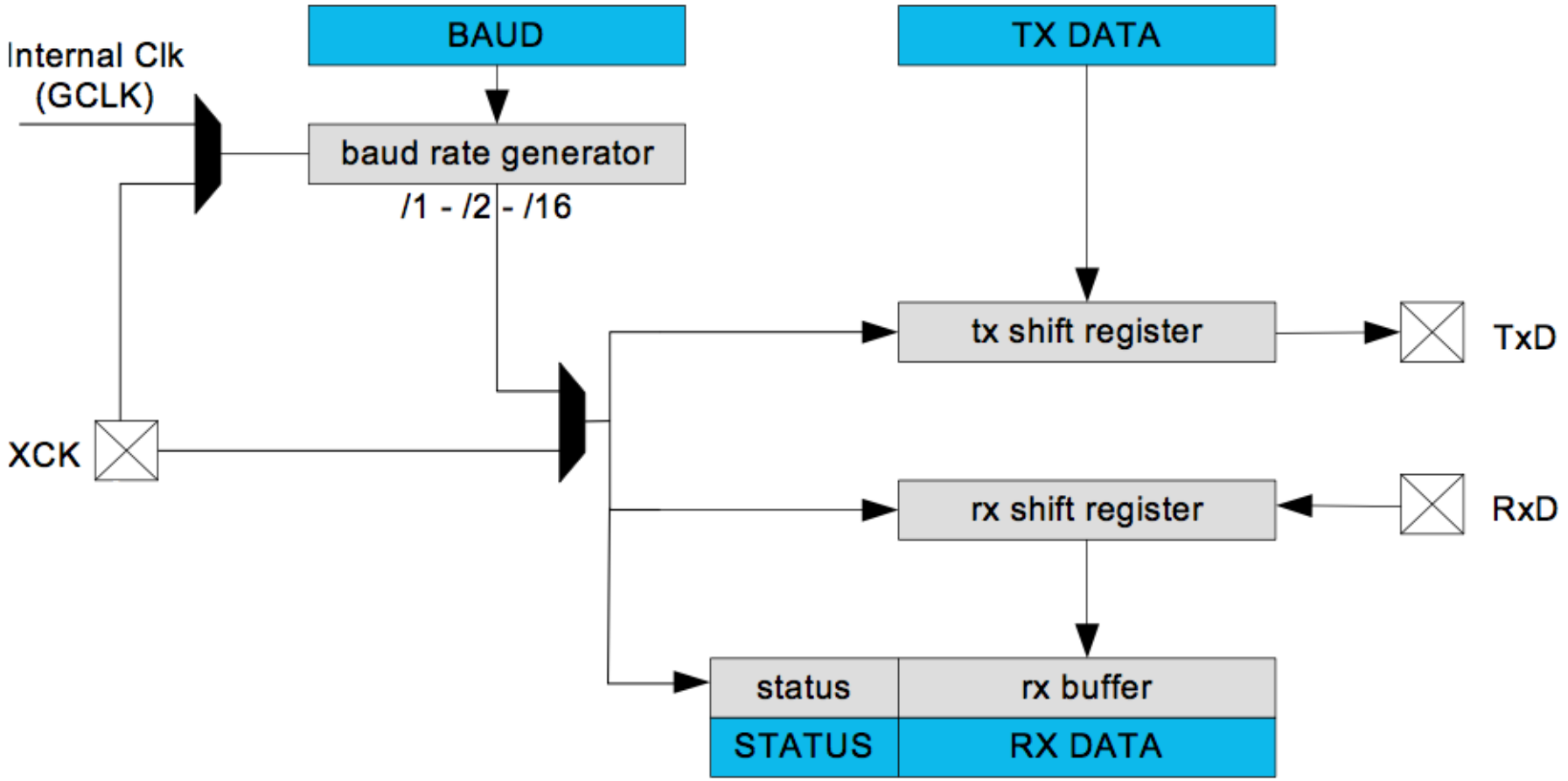


# Baud-Rate Generator

| Operating Mode | Condition | Baud Rate (Bits Per Second) | BAUD Register Value Calculation |
|---|---|---|---|
| Asynchronous Arithmetic | $f_{BAUD} \leq \frac{f_{REF}}{S}$ | $f_{BAUD} = \frac{f_{REF}}{S}(1 - BAUD / 65{,}536)$ | $BAUD = 65{,}536\left(1 - S\frac{f_{BAUD}}{f_{REF}}\right)$ |
| Asynchronous Fractional | $f_{BAUD} \leq \frac{f_{REF}}{S}$ | $f_{BAUD} = \frac{f_{REF}}{S(BAUD + (FP/8))}$ | $BAUD = \frac{f_{REF}}{S \times f_{BAUD}} - \frac{FP}{8}$ |
| Synchronous | $f_{BAUD} \leq \frac{f_{REF}}{2}$ | $f_{BAUD} = \frac{f_{REF}}{2(BAUD + 1)}$ | $BAUD = \frac{f_{REF}}{2 f_{BAUD}} - 1$ |

S – Number of samples per bit. Can be 16, 8, or 3.

# UART Example



# Basic Operation

- Initialization
- Enabling, Disabling and Resetting
- Clock Generation and Selection
- Data Register
  - Data Transmission
  - Data Reception

# Initialization

- USART mode with external or internal clock must be selected first by writing 0x0 or 0x1 to the Operating Mode bit group in the Control A register (CTRLA.MODE)
- Communication mode (asynchronous or synchronous) must be selected by writing to the Communication Mode bit in the Control A register (CTRLA.CMODE)
- SERCOM pad to use for the receiver must be selected by writing to the Receive Data Pinout bit group in the Control A register (CTRLA.RXPO)
- SERCOM pads to use for the transmitter and external clock must be selected by writing to the Transmit Data Pinout bit in the Control A register (CTRLA.TXPO)
- Character size must be selected by writing to the Character Size bit group in the Control B register (CTRLB.CHSIZE)

# Initialization (Cont.)

- MSB- or LSB-first data transmission must be selected by writing to the Data Order bit in the Control A register (CTRLA.DORD)
- When parity mode is to be used, even or odd parity must be selected by writing to the Parity Mode bit in the Control B register (CTRLB.PMODE) and enabled by writing 0x1 to the Frame Format bit group in the Control A register (CTRLA.FORM)
- Number of stop bits must be selected by writing to the Stop Bit Mode bit in the Control B register (CTRLB.SBMODE)
- When using an internal clock, the Baud register (BAUD) must be written to generate the desired baud rate
- The transmitter and receiver can be enabled by writing ones to the Receiver Enable and Transmitter Enable bits in the Control B register (CTRLB.RXEN and CTRLB.TXEN)

# Enabling, Disabling and Resetting

- The USART is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The USART is disabled by writing a zero to CTRLA.ENABLE
- The USART is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the USART, except DBGCTRL, will be reset to their initial state, and the USART will be disabled. Refer to the CTRLA register for details

# Clock Generation and Selection

- Synchronous mode is selected by writing a one to the Communication Mode bit in the Control A register (CTRLA.CMODE) and asynchronous mode is selected by writing a zero to CTRLA.CMODE. The internal clock source is selected by writing 0x1 to the Operation Mode bit group in the Control A register (CTRLA.MODE) and the external clock source is selected by writing 0x0 to CTRLA.MODE.
- When CTRLA.CMODE is zero, the baud-rate generator is automatically set to asynchronous mode and the 16-bit Baud register value is used. When CTRLA.CMODE is one, the baud-rate generator is automatically set to synchronous mode and the eight LSBs of the Baud register are used.

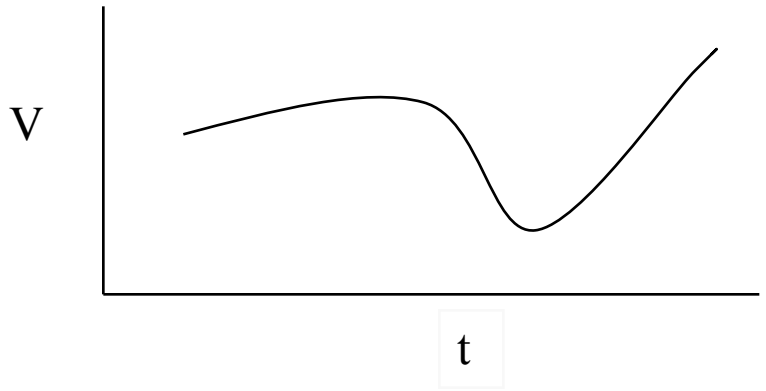
# Data Register (Tx)

- The USART Transmit Data register (TxDATA) and USART Receive Data register(RxDATA) share the same I/O address, referred to as the Data register (DATA). Writing the DATA register will update the Transmit Data register. Reading the DATA register will return the contents of the Receive Data register.
- A data transmission is initiated by loading the DATA register with the data to be sent. The data in TxDATA is moved to the shift register when the shift register is empty and ready to send a new frame. When the shift register is loaded with data, one complete frame will be transmitted.
- The Transmit Complete interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) is set, and the optional interrupt is generated, when the entire frame plus stop bit(s) have been shifted out and there is no new data written to the DATA register.
- The DATA register should only be written when the Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is set

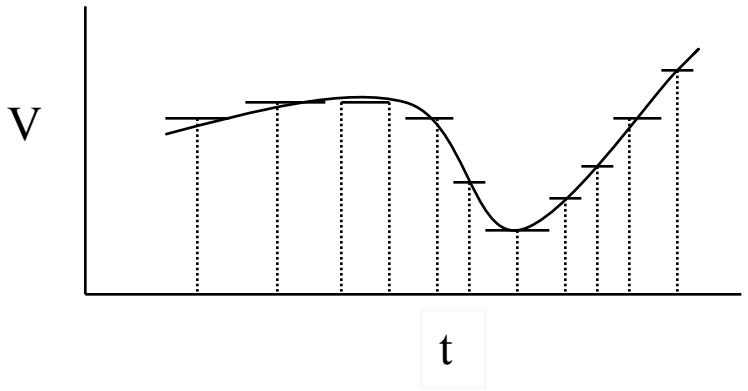
# ADC and DAC (Basic Concepts)

- Slides are adopted from
- http://harding.edu/jmackey/physics350/adc/adc_dac%20converter.ppt

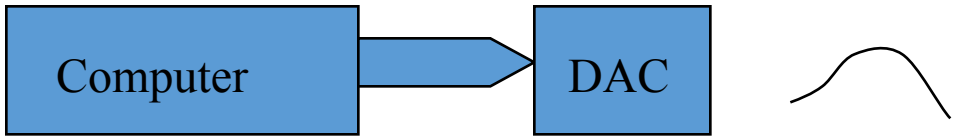
Real world (lab) is analog



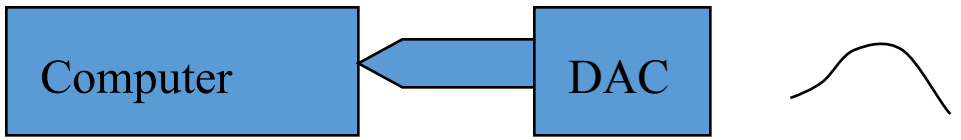

Computer (binary) is digital



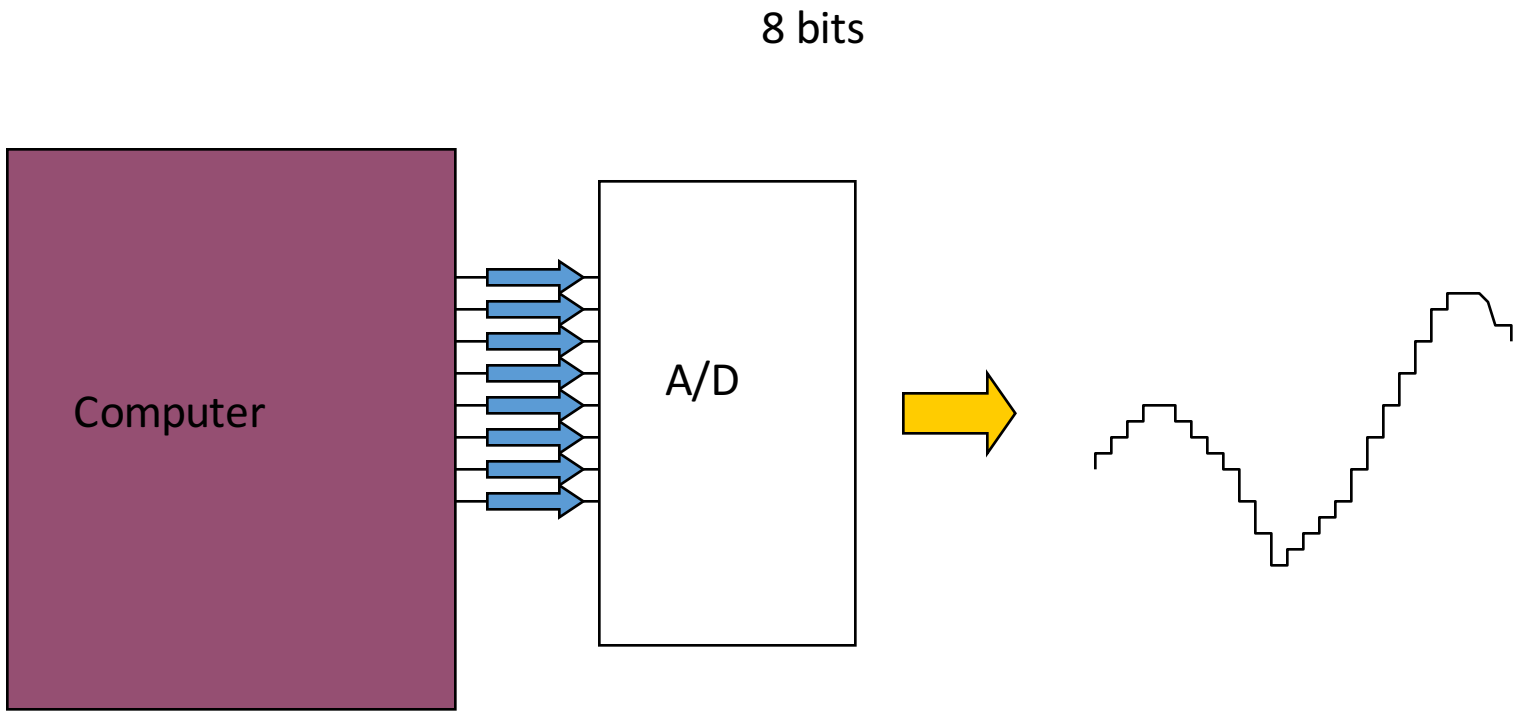

D/A Conversion



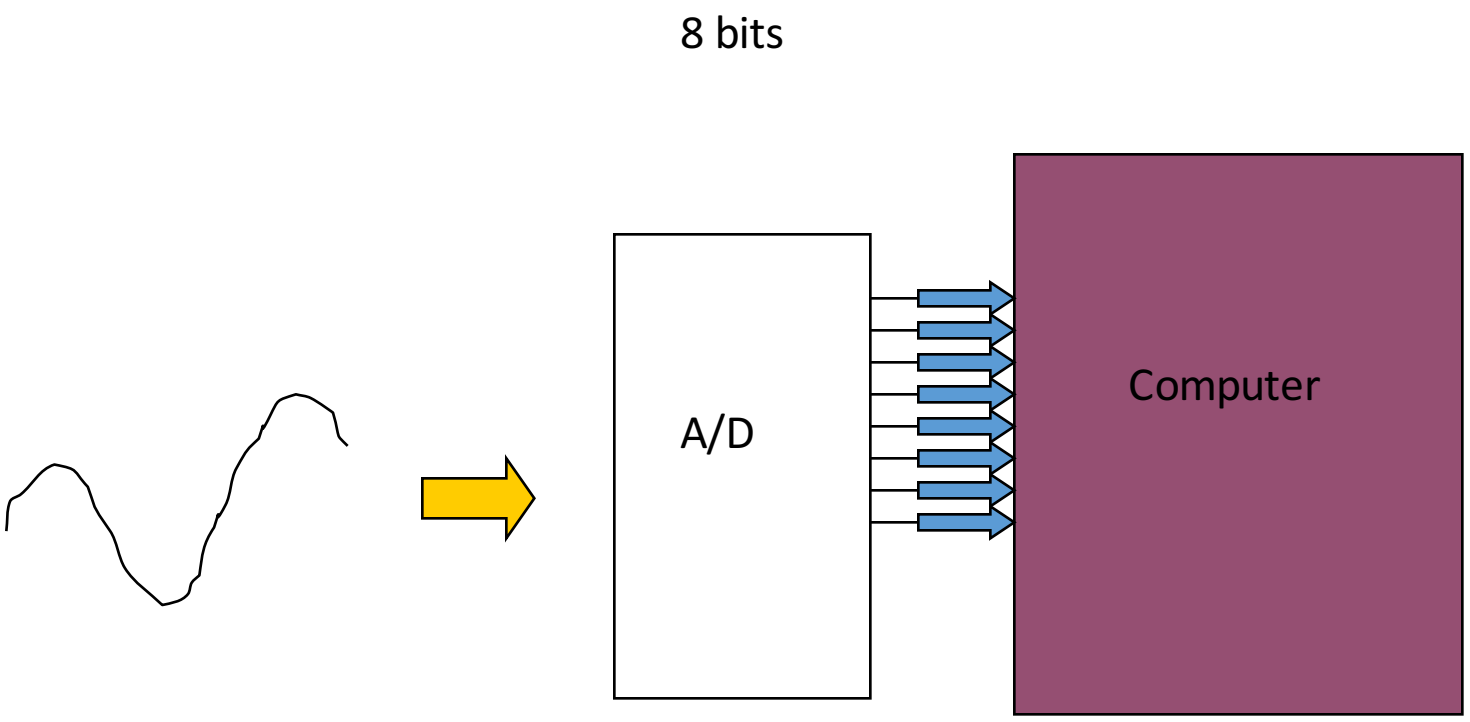

A/D Conversion



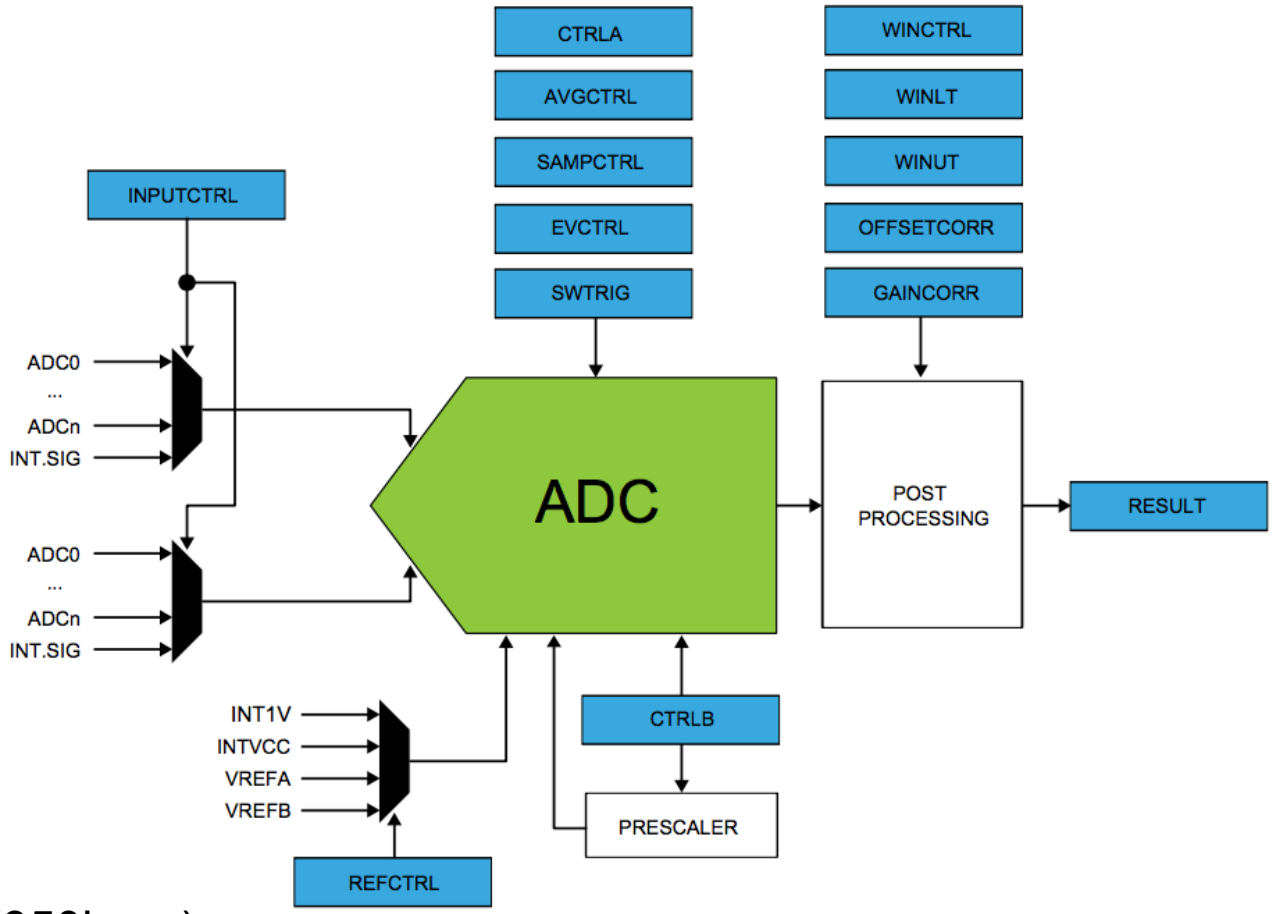
Digital to Analog Conversion (DAC or D/A)



# Analog-to Digital Conversion (ADC or A/D)

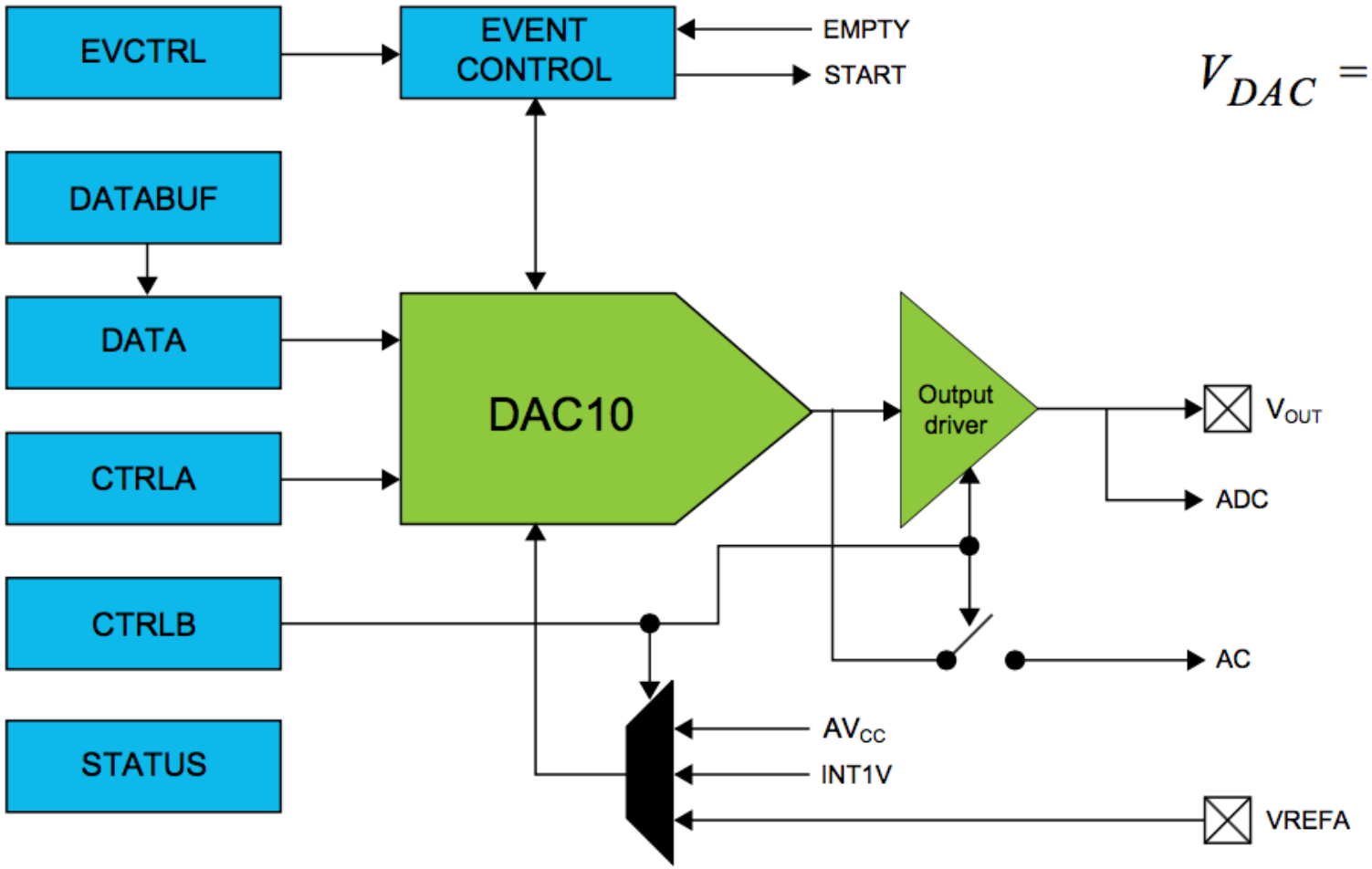


# Atmel SAMD21 MCU/MO Pro




8-, 10- or 12-bit resolution
Up to 350,000 samples per second (350ksps)
1/2x to 16x gain

# Atmel SAMD21 MCU/MO Pro




$$V_{DAC} = \frac{DATA}{0x3FF} \cdot VREF$$

```
/*
 *
 * + Pin number +  ZERO Board pin  |  PIN   | Label/Name      | Comments (* is for default peripheral in use)
 * +------------+------------------+--------+-----------------+------------------------------
 * |            | Digital Low      |        |                 |
 * +------------+------------------+--------+-----------------+------------------------------
 * | 0          | 0 -> RX          |  PA11  |                 | EIC/EXTINT[11] ADC/AIN[19] PTC/X[3] *SERCOM0/PAD[3] SERCOM2/PAD[3] TCC1/WO[1] TCC0/WO[3]
 * | 1          | 1 <- TX          |  PA10  |                 | EIC/EXTINT[10] ADC/AIN[18] PTC/X[2] *SERCOM0/PAD[2] TCC1/WO[0] TCC0/WO[2]
 * | 2          | ~2               |  PA08  |                 | EIC/NMI ADC/AIN[16] PTC/X[0] SERCOM0/PAD[0] SERCOM2/PAD[0] *TCC0/WO[0] TCC1/WO[2]
 * | 3          | ~3               |  PA09  |                 | EIC/EXTINT[9] ADC/AIN[17] PTC/X[1] SERCOM0/PAD[1] SERCOM2/PAD[1] *TCC0/WO[1] TCC1/WO[3]
 * | 4          | ~4               |  PA14  |                 | EIC/EXTINT[14] SERCOM2/PAD[2] SERCOM4/PAD[2] TC3/WO[0] *TCC0/WO[4]
 * | 5          | ~5               |  PA15  |                 | EIC/EXTINT[15] SERCOM2/PAD[3] SERCOM4/PAD[3] TC3/WO[1] *TCC0/WO[5]
 * | 6          | ~6               |  PA20  |                 | EIC/EXTINT[4] PTC/X[8] SERCOM5/PAD[2] SERCOM3/PAD[2] TC7/WO[0] *TCC0/WO[6]
 * | 7          | ~7               |  PA21  |                 | EIC/EXTINT[5] PTC/X[9] SERCOM5/PAD[3] SERCOM3/PAD[3] TC7/WO[1] *TCC0/WO[7]
 * +------------+------------------+--------+-----------------+------------------------------
 * |            | Digital High     |        |                 |
 * +------------+------------------+--------+-----------------+------------------------------
 * | 8          | ~8               |  PA06  |                 | EIC/EXTINT[6] PTC/Y[4] ADC/AIN[6] AC/AIN[2] SERCOM0/PAD[2] *TCC1/WO[0]
 * | 9          | ~9               |  PA07  |                 | EIC/EXTINT[7] PTC/Y[5] DC/AIN[7] AC/AIN[3] SERCOM0/PAD[3] *TCC1/WO[1]
 * | 10         | ~10              |  PA18  |                 | EIC/EXTINT[2] PTC/X[6] SERCOM1/PAD[2] SERCOM3/PAD[2] *TC3/WO[0] TCC0/WO[2]
 * | 11         | ~11              |  PA16  |                 | EIC/EXTINT[0] PTC/X[4] SERCOM1/PAD[0] SERCOM3/PAD[0] *TCC2/WO[0] TCC0/WO[6]
 * | 12         | ~12              |  PA19  |                 | EIC/EXTINT[3] PTC/X[7] SERCOM1/PAD[3] SERCOM3/PAD[3] *TC3/WO[1] TCC0/WO[3]
 * | 13         | ~13              |  PA17  | LED             | EIC/EXTINT[1] PTC/X[5] SERCOM1/PAD[1] SERCOM3/PAD[1] *TCC2/WO[1] TCC0/WO[7]
 * | 14         | GND              |        |                 |
 * | 15         | AREF             |  PA03  |                 | *DAC/VREFP PTC/Y[1]
 * | 16         | SDA              |  PA22  |                 | EIC/EXTINT[6] PTC/X[10] *SERCOM3/PAD[0] SERCOM5/PAD[0] TC4/WO[0] TCC0/WO[4]
 * | 17         | SCL              |  PA23  |                 | EIC/EXTINT[7] PTC/X[11] *SERCOM3/PAD[1] SERCOM5/PAD[1] TC4/WO[1] TCC0/WO[5]
 * +------------+------------------+--------+-----------------+------------------------------
 * |            |SPI (Legacy ICSP) |        |                 |
 * +------------+------------------+--------+-----------------+------------------------------
 * | 18         | 1                |  PA12  | MISO            | EIC/EXTINT[12] SERCOM2/PAD[0] *SERCOM4/PAD[0] TCC2/WO[0] TCC0/WO[6]
 * | 19         | 2                |        | 5V0             |
 * | 20         | 3                |  PB11  | SCK             | EIC/EXTINT[11]                *SERCOM4/PAD[3] TC5/WO[1] TCC0/WO[5]
 * | 21         | 4                |  PB10  | MOSI            | EIC/EXTINT[10]                *SERCOM4/PAD[2] TC5/WO[0] TCC0/WO[4]
 * | 22         | 5                |        | RESET           |
 * | 23         | 6                |        | GND             |
 * +------------+------------------+--------+-----------------+------------------------------
 * |            | Analog Connector |        |                 |
 * +------------+------------------+--------+-----------------+------------------------------
 * | 24         | A0               |  PA02  |                 | EIC/EXTINT[2] *ADC/AIN[0] PTC/Y[0] DAC/VOUT
 * | 25         | A1               |  PB08  |                 | EIC/EXTINT[8] *ADC/AIN[2] PTC/Y[14] SERCOM4/PAD[0] TC4/WO[0]
 * | 26         | A2               |  PB09  |                 | EIC/EXTINT[9] *ADC/AIN[3] PTC/Y[15] SERCOM4/PAD[1] TC4/WO[1]
 * | 27         | A3               |  PA04  |                 | EIC/EXTINT[4] *ADC/AIN[4] AC/AIN[0] PTC/Y[2] SERCOM0/PAD[0] TCC0/WO[0]
 * | 28         | A4               |  PA05  |                 | EIC/EXTINT[5] *ADC/AIN[5] AC/AIN[1] PTC/Y[5] SERCOM0/PAD[1] TCC0/WO[1]
 * | 29         | A5               |  PB02  |                 | EIC/EXTINT[2] *ADC/AIN[10] PTC/Y[8] SERCOM5/PAD[0] TC6/WO[0]
 * +------------+------------------+--------+-----------------+------------------------------
 * |            | LEDs             |        |                 |
 * +------------+------------------+--------+-----------------+------------------------------
 * | 30         |                  |  PB03  | RX              |
 * | 31         |                  |  PA27  | TX              |
```