3.6 Getting started with ARM cortex-M3 and communication between nodes

This is the exercise where Node 2 will be introduced for the first time. As shown in section 1.1, it consists of the Arduino Due and a shield mounted on top of the Arduino.
NOTE: The Arduino Due uses 3.3V Transistor-Transistor Logic (TTL) levels. Do not connect 5V devices to it directly as this might damage the pins.

3.6.1 Arduino USB Communication

The Arduino Due allows us to use the USB connection as a USB to serial bridge via a separate ATmega chip onboard, ATMEGA16U2. The ATSAM3X8E is connected to the USB bridge through UART via PA8 and PA9. A simple UART and printf library to get you started is available on Blackboard.

3.6.2 Arduino Shield

The custom Arduino Shield sits on top of the Arduino Due providing the following:

- Two 4 mm banana plug receptacles for motor power input
- Two 2 mm banana plug receptacles for motor power output
- One 3x2 pin header for motor encoder input
- One 3x1 pin header for servo power/control output
- One 5V voltage regulator for servo power
- One 2x1 pin header for CAN wires
- One jumper for CAN terminal resistor
- One MCP2562 CAN transciever
- One Allegro A3959 motor driver

Please do not remove the shield.

A thorough description of the shield and its usage in this project may be found in the *TTK4155 Motor Shield* document.

3.6.3 ATSAM3X8E on Linux

For programming the ATSAM3X8E on Linux some additional files will be needed in addition to the Makefile. These files set up the microcontroller and specifies where in memory the program is to be flashed. The files can be found on Blackboard. Include the folder named SAM and the Makefile in the same folder as your C-code.

3.6.4 CAN bus bit timing

All nodes on a CAN bus must have the same bit timing in order to communicate. The CAN *bit time* (the time it takes to eject one bit on the bus) is made up of non-overlapping segments where each segment consist of a given number of *Time Quantas* (TQs). Normally the *CAN bit time* consist of four segments:

1. Synchronisation segment (t_{sync}): This segment is used to synchronise the nodes on the bus.
2. Propagation segment (t_{prop}): This segment compensates for physical delays between nodes.
3. Phase segment 1 and 2 ((t_{seg1}, t_{seg2}): These segments are used to compensate for edge phase errors on the bus. The sampling point is the point between t_{seg1} and t_{seg2}.

An illustration is shown in figure 23.

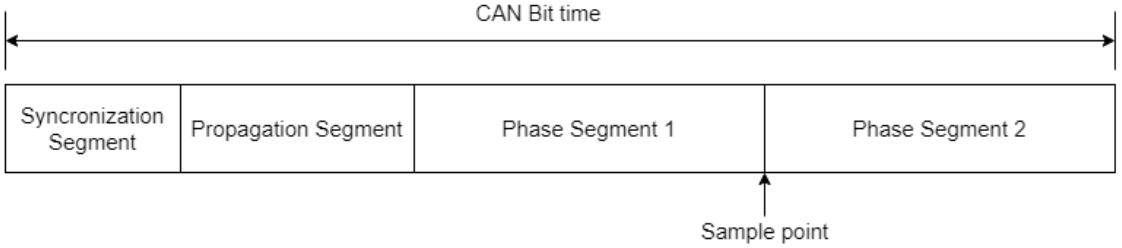

Figure 23: CAN Bit timing

The bit time may be configured by setting registers in the CAN controllers. Make sure that each segment consists of the same number of Time Quantas on every node and that the Time Quantas are equal. The length of a time Quanta can be set by adjusting the Baudrate according to the master oscillator clock frequency.

3.6.5 Exercise

1. Recommended but optional: Write a program which enables (i.e. sets high) the servo header signal pin, and verify your result with an oscilloscope. Think of it as a "hello world" for the Node 2 shield.
2. It is useful to implement serial communication also on Node 2. A simple library is available in Blackboard. Make sure to find/set the baudrate in `uart.c`.
3. Connect the CAN transceiver MCP2551 on Node 1 using the information provided in the datasheet. You can use the 22k resistor for slew-rate limiting.
4. Import the CAN library for node 2 to your project. The library is available in Blackboard. Feel free to adapt it to your needs as the project progresses, e.g. by implementing message ID masks, circular buffers etc.
5. Connect Node 1 and Node 2 together using the CAN bus, conforming to the "AN228: A CAN Physical Layer Discussion" document. **Note:** The `CAN_TERM` jumper on the Node 2 shield *must* be connected at this point.
6. Decide the CAN bus bit-timing by writing to the `CNFx` registers on MCP2515. Make sure to match the configuration in the `CAN_BR` register on ATSAM3X8E.
7. Test the system again, but now with the CAN controller of node 1 in normal mode. Node 2 should be able to reuse the upper level code generated in the previous exercise.
8. Make a joystick driver that can send joystick position from Node 1 to Node 2 over the CAN bus.

3.6.6 Tips

- For UART over USB to work, connect the micro USB-cable to the right USB-port (the one closest to the round power supply connector).
- Remember to enable peripherals in PMC.

- If the microcontroller keeps resetting, make sure to feed the *watchdog timer (WDT)* (See section 15 of the ATsam3X8E datasheet). You can also disable it by setting `WDT->WDT_MR` to `WDT_MR_WDDIS`.
- Sending CAN messages in normal mode will never succeed before there is at least one node that can acknowledge the transmission.
- Remember to terminate the bus in both ends – the shield has built-in termination that can be disabled by removing JP1.
- The order of writing to the CNFx registers in MCP2515 matters. Check that they are correctly configured after writing to all three.
- In this lab we will use JTAG, however the USB to serial bridge on the Arduino can be used to program it. The DTR (Data Terminal Ready) which signals a new connection in RS232, is connected to the Reset line of the ATsam3X8E. Therefore, the microcontroller is reset every time you connect to the COM port in Windows.