# Tutorial - Getting started with CAN

This document is a quick tutorial over;

1. How to use CANking.
2. How to get started with CAN in AVR32 Studio.
3. How to connect the external CAN controller (MCP2515) with the EVK1100 board.
4. How to use the Scania dashboard.

CANking

- Start CANking, click in "Ok, I know what I'm doing", choose "template" and click "Ok". Then choose CAN Kingdom Basic.
- Under CAN Controller you need to set the Bus Speed. It has to be the same for all the boards that you are using. Under CAN Channel, look that the Kvaser USBcan II HS/HS is chosen. Now click on "Bus Statistics" and click on "Go On Bus".
- You should now be able to send and receive messages.
- In the Output Window you can see if any message has been received.
- To send messages go to Messages->Universal->Universal.
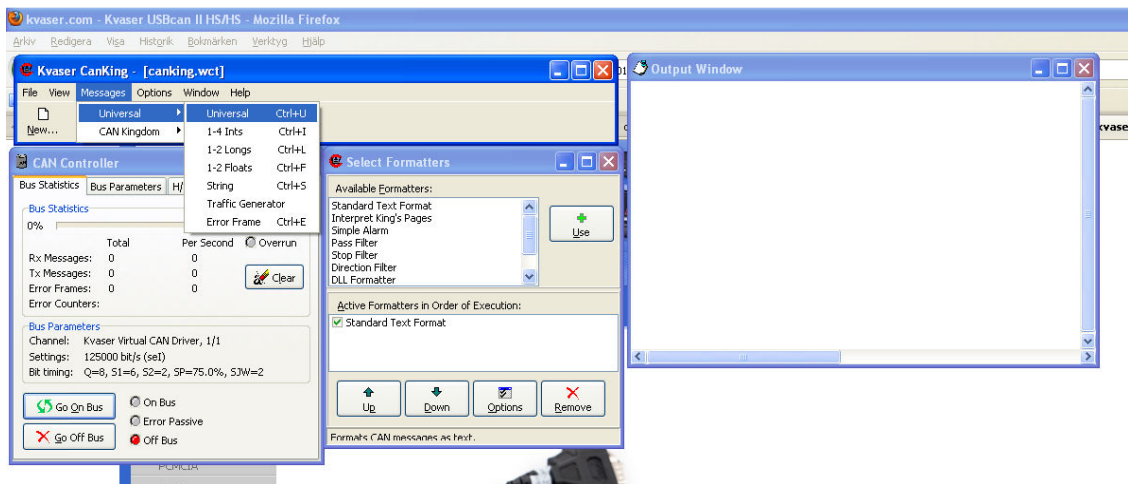
- Here you can send messages with different Id, length and data content.
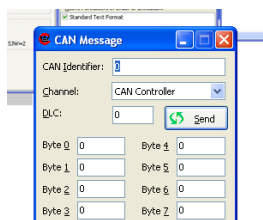
MCP2515

Step 1

Start with creating a new AVR 32 C Project From Template. Then copy the content from "main.c" from the homepage to the main.c file created in your project.

Step 2

This main.c file contains some includes that have to be included from the SOFTWARE_FRAMEWORK. To be able to use dip204.h you have to include "CPU-cycle counter" in the SOFTWARE_FRAMEWORK and then click "Next" to go to the second page and then check the box "Display (dip204) component".
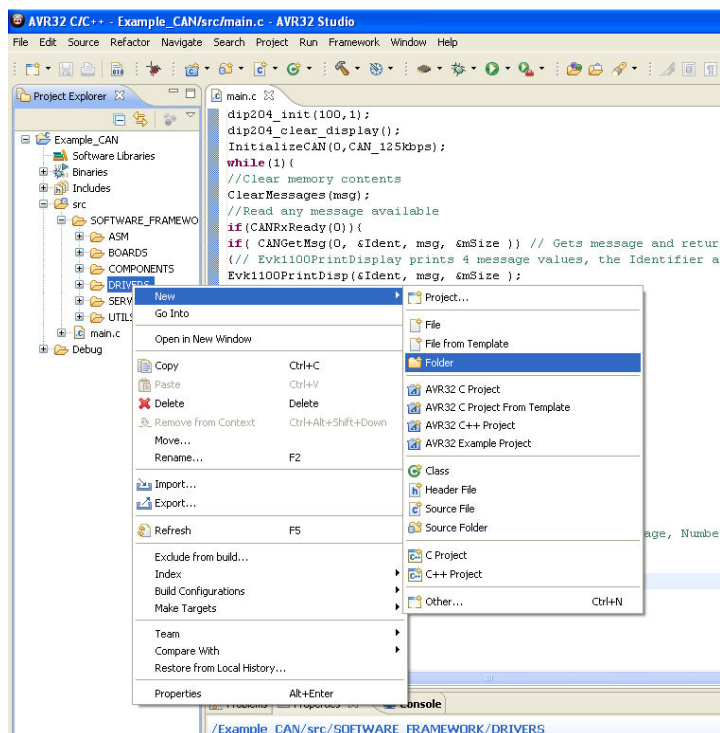
The includes that have to be added in the SOFTWARE_FRAMEWORK are;

```c
#include "board.h"
#include "can.h"
#include "spi.h"
#include "gpio.h"
#include "pm.h"
#include "dip204.h"
#include "intc.h"
#include "delay.h"
#include "string.h"
```

Step 3

To be able to get the CAN module to work we have to add the can.c, can.h and regs2515.h files that are provided on the homepage. To include these files you can do like this.

Right-click on "DRIVERS" under "SOFTWARE_FRAMEWORK". Created a new folder called CAN.

Embedded Systems for Mechatronics 1, MF2042
Tutorial – Getting started with CAN
version 2011-10-04

Step 4

Right-click on the CAN folder you just have created and create a new source file called can.c. Copy
the content from can.c on the homepage to the file you just have created.
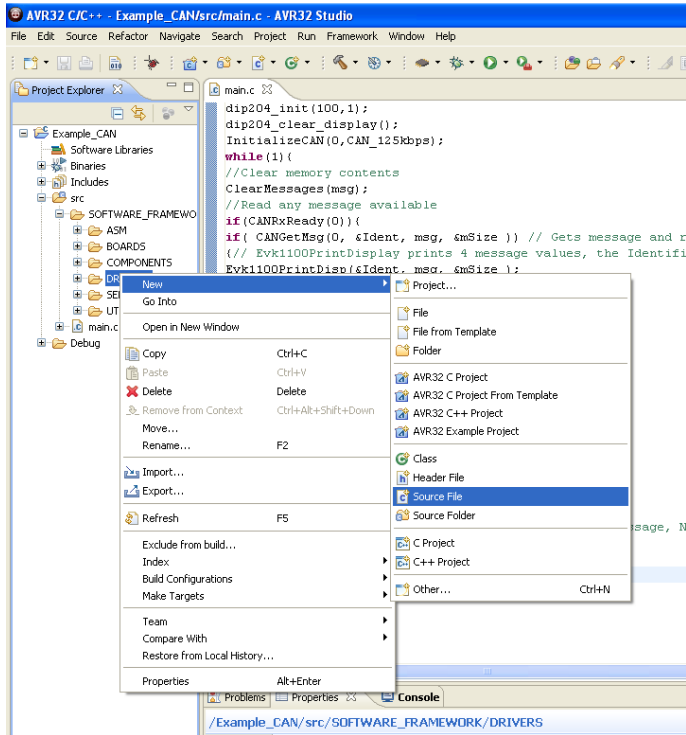
Right-click again on the CAN folder and create two header files called can.h and regs2515.h. Copy the
content on can.h and regs2515.h on the homepage to the files you just have created.
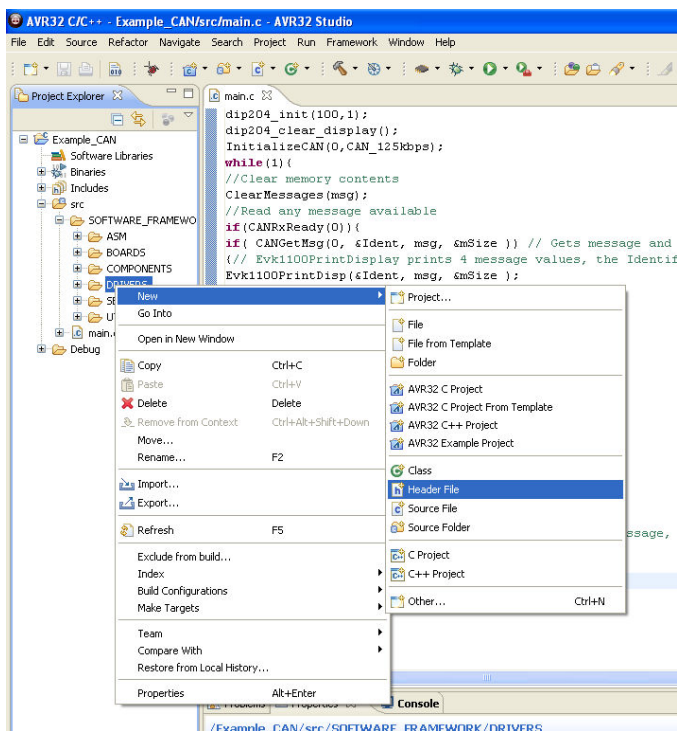
Step 5

For the compiler to be able to find the new files we have added, we need to add a link to the folder. Go to project-> properties. Click on "C/C++ Build" and then on "Settings". Under "AVR 32/GNU C Compiler" click on "Directories". Click on "add" and then click on workspace and click on the folder that you just have created (CAN). Now everything should work!
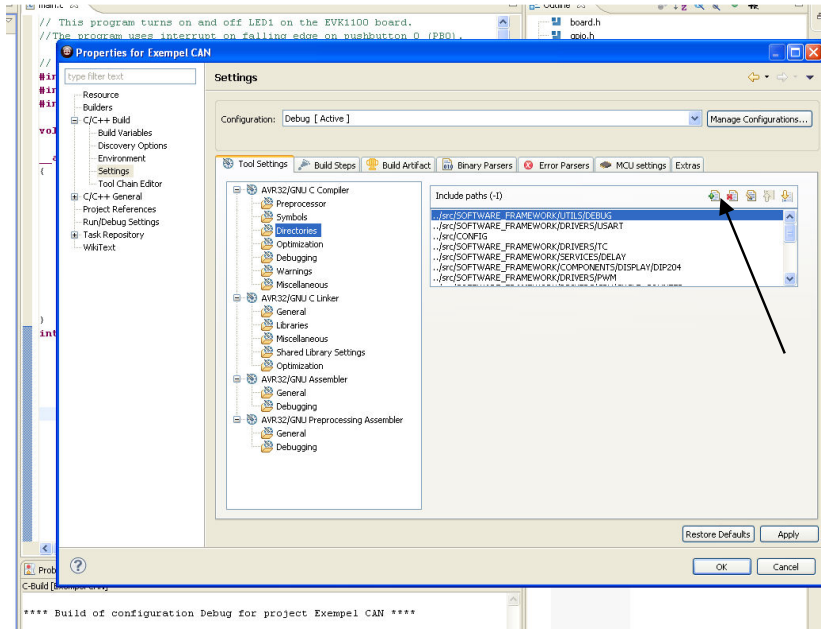
Now you should be able to compile your project.

Connecting the MCP2515 board to the EVK1100 board
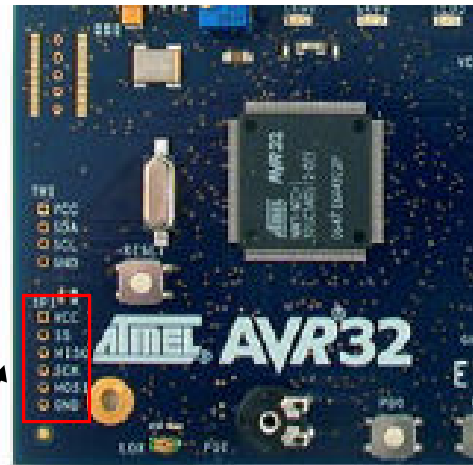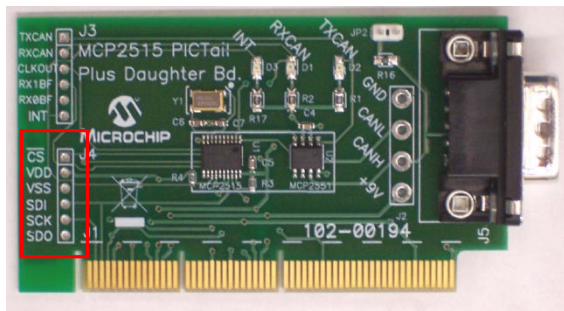
To connect the MCP2515 board to the EVK1100 board the following pins have to be connected.

MCP2515		EVK1100
CS (Chip Select)	←————→	SS (Slave Select)
VDD = External 5v (see **Error! Reference source not found.**)		VCC (not connected)
VSS = ground (common with the EVK1100)	←————→	GND (Common with the MCP2515 board)
SDI (Serial Data In)	←————→	MISO (Master Input Slave Output)
SCK (Serial Clock)	←————→	SCK (Serial Clock)
SDO (Serial Data Output)	←————→	MOSI (Master Output Slave Input)

The power supply for the MCP2515 board is taken from the power supply next to your computer (see Figure 1: Power supply). Make sure that the MCP2515 and the EVK1100 board uses common ground. Cables can be found in the lab.
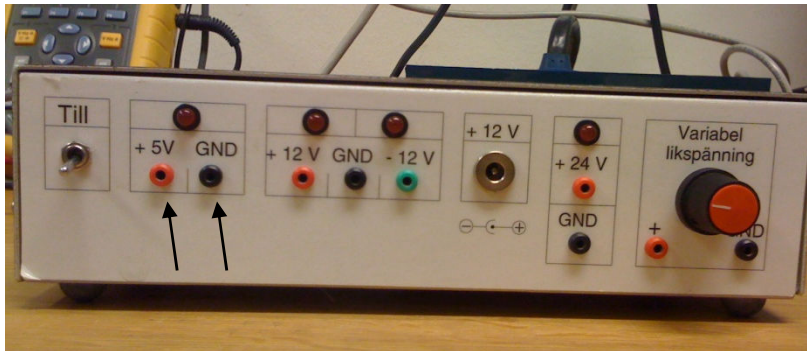

Figure 1: Power supply

CAN bus

In the lab there are three parallel CAN busses installed on the walls. A red, a yellow and a green bus that can be seen in Figure 2: CAN bus termination resistors   Figure 3: CAN bus sockets. At each end of every CAN bus there is one 120 ohm resistor connected, (see Figure 2: CAN bus termination resistors   Figure 3: CAN bus sockets. This is because that the CAN bus needs to be terminated.

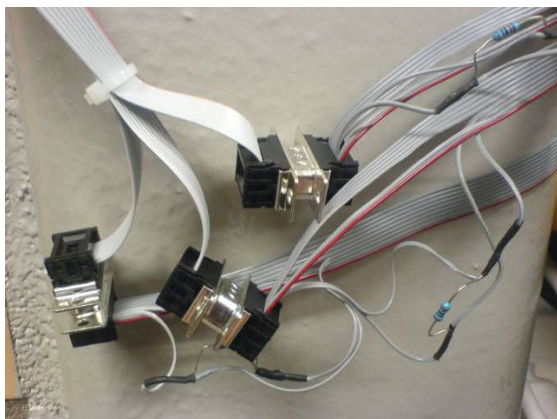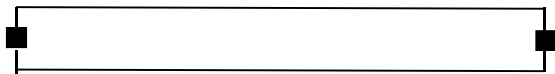120 Ω   120 Ω
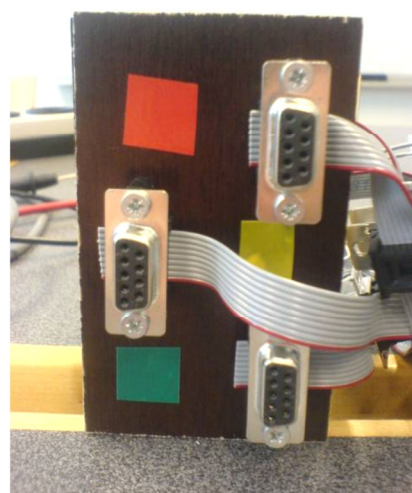

Figure 2: CAN bus termination resistors   **Figure 3: CAN bus sockets**

If you are using the MCP2515 board there are a 120 Ω resistor integrated on the board that can be used when you want to directly send messages to CANking, without using the CAN busses on the wall. To use this resistor use the jumper shown in Figure 4: MCP2515.

Figure 4: MCP2515

Scania dashboard

One fun way of testing CAN is to try to control the Scania dashboard in the lab (see Figure 5: Scania dashboard). The Scania dashboard datasheet can be found on the homepage, and shows the different Id:s, which byte that should be set and the data range. The Scania dashboard is connected to the yellow bus (middle) and uses 250 kbps bus speed.
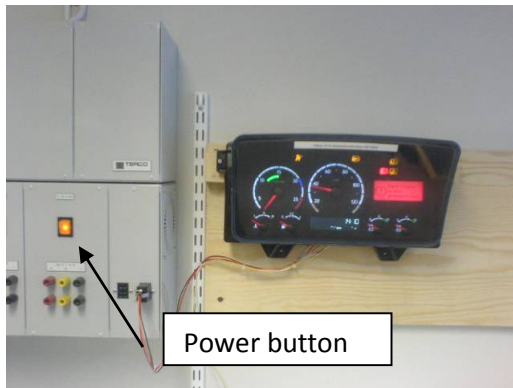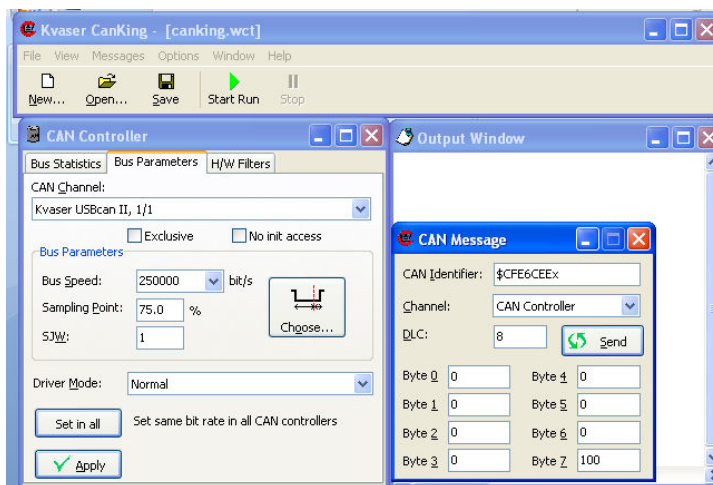

Power button

Figure 5: Scania dashboard

CANking settings:

Make sure that you put an "x" after the Identifier in the CAN Identifier field, since the Scania dashboard uses Extended Id.

MCP2515 example:

For example the Velocity indicator can be controlled by sending one message to the right identifier (0X0CFE6CEE). The 7th byte is set to 50 and the velocity indicator should now show 50 km/h.

```
msg[0]=0;
msg[1]=0;
msg[2]=0;
msg[3]=0;
msg[4]=0;
msg[5]=0;
msg[6]=0;
msg[7]=50;

// Channel, Identifier (max 0x1fffffff (29 bits)), Message, Number of bytes, R or 0 (Remote
//frame or no remote frame).
CANSendMsg( 0, 0x0cfe6cee, msg, 8, 0 );
delay_ms(1000);
dip204_clear_display();
```

If you want to control the other instruments this can be done in the same way.