SIEMENS

SIEMENS

A Hands-On Introduction to CAN

SIEMENS

Contents

- **Introduction**
- **Hands-On CAN Training using the C167CR microcontroller**
  - Exercise Overview
  - How to set up the Hard- and Software used
  - Exercises

SIEMENS

Introduction

- **This is a Hands-On CAN Training material, using**
  - An evaluation board with a 16-bit microcontroller with on-chip CAN module (the Phytec kitCON-167 Starter Kit, equipped with the C167CR 16-bit microcontroller)
  - A development environment for the C167CR (Keil uVision; including compiler, assembler, linker/locator, debugger)
  - A 'C'-code generator tool (DAvE, the Digital Application Engineer)
  - An oscilloscope (for visualization purposes).
  - A Windows95 or Windows NT PC (to be able to run DAvE)
  - A CAN-bus analyzer (the Vector CANalyzer) for bus monitoring
- **How to generate CAN software from the scratch:**
  - There are several exercises included, small tasks to be solved using the CAN module of the C167CR.
  - We will create a project for the C167CR in DAvE and configure the device, following the detailed instructions.

SIEMENS

Introduction (cont.)

- After having generated the 'C' code, we will
  - switch to the development environment,
  - create a new project or alter the previous project,
  - include the 'C'-files created by DAvE,
  - compile, assemble, link and locate the project.
- After building the project, we will
  - switch to the debugger,
  - connect to the evaluation board via bootstrap loader (loading the monitor),
  - load, start and debug the example,
  - confirm the working program with an oscilloscope and by watching the trace window of the CAN bus analyzer on the screen.

SIEMENS

Short Introduction to the 'C' code generator tool used (DAvE)

- **DAvE is your Digital Application Engineer from Siemens Microcontrollers.**
- **DAvE can help you compare and evaluate the different members of the Siemens C500 (8-Bit) and C166 (16-Bit) families of microcontrollers and help you find the right chip for your embedded control application.**
- **DAvE can be your one-stop access point to all standard knowledge associated with Siemens embedded technology expertise by offering you context sensitive access to user's manuals, data sheets, application notes etc. directly in your development environment.**
- **DAvE can help you program the Siemens microcontroller you want to use in your project, by offering you intelligent wizards that help you configure the chip to work the way you need it and automatically generate C-level templates with appropriate access functions for all of the on chip peripherals and interrupt controls.**
- **For more information on DAvE, go to http://www.smi.siemens.com/DAvE.html**

SIEMENS

Short Introduction to the development environment used (Keil uVision)

- **Keil uVision:**
  - The µVision/166 integrated development environment provides a Windows-based front end for the Keil 166/167 tools.
  - µVision offers features like a configurable, integrated editor, integrated project manager and make facility, support for external tools, color syntax highlighting, interactive error correction and intuitive setup menus for tool options.
  - µVision makes it easy to set compiler options. The compiler options dialog box provides standard controls for all the major compiler options.
  - Linker options are only a click away. A scrolling command line window lets you see the actual linker options at a glance.

SIEMENS

Short Introduction to the development environment used (Keil uVision) (cont.)

- **C166 Compiler:**
  - The Keil C166 compiler unlocks the features and performance of the Siemens C166 microcontroller family. C166 is easy to use and minimizes the learning curve for starting your C166 projects. Extensions incorporated into the C166 compiler give you full access to all CPU resources and derivatives. C166 Version 3 offers support for all derivatives and full compatibility with all major emulator vendors. The C166 compiler conforms to the ANSI standard and is designed specifically for the C166 microcontroller family.
- **A166 Assembler, L166 Linker/Locator:**
  - The A166 macro assembler and the L166 Linker Locator are part of the C166 compiler package.
- **More information is available from Keil or on the Siemens SPACE Tools CD ROM.**

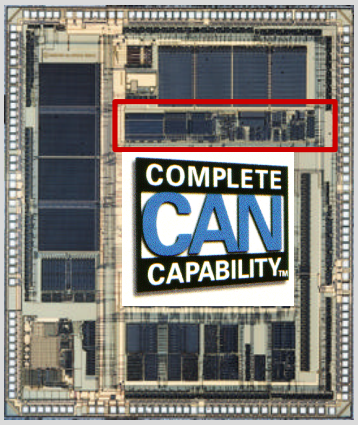
SIEMENS

Short Introduction to the debugger used (Keil dScope)

- **dScope Debugger:**
  - dScope-166 is a source-level debugger that lets you debug programs created by the C166 compiler and A166 assembler.
  - dScope includes both a hardware simulator for the C166 family and a target interface for the 166 monitor.
  - dScope simulates all the peripherals of the C166 and its derivatives with loadable CPU drivers.
  - A C-like macro language lets you write functions that simulate complex,external I/O operations.
- **More information is available from Keil or on the Siemens SPACE Tools CD ROM.**

SIEMENS

CAN Hands-On CAN Training



- **Exercise Overview**
- **How to set up the Hard- and Software used**
- **Exercises**
  - How to transmit messages
  - How to receive messages
  - Reacting on an incoming message
  - Transmitting a Remote Frame

SIEMENS

Exercise Overview

- **Exercise 1:**
  - Set up a message object to transmit a standard CAN Data Frame
- **Exercise 2:**
  - Activate the CAN transmit interrupt for this message object and increment the data bytes by one in the interrupt service routine after each transmission
- **Exercise 3:**
  - Set up a message object to receive Data Frames and react on the received Data Frame in the CAN interrupt service routine by reading out the received data bytes
- **Exercise 4:**
  - Write the data bytes from the receive message object into the data bytes of message object 3 (to be configured for transmission) send them “back” to the CANalyzer
- **Exercise 5:**
  - Transmit a Remote Frame and react on the corresponding Data Frame

SIEMENS

How to set up your system: kitCON-167 HW setup

- **Connect your Starter Kit board to your IBM compatible PC using a serial cable (connector P1 to COM1)**
  - Attention Windows 95 users:
    It is recommended to disable the FIFO Buffer in Settings/Control Panel/System/Device Manager/Ports/COM1/Port Settings/Advanced.
- **Connect jumper JP2 at 1+2 (usually a red jumper)**
- **Attach an unregulated power supply with 8V to 12V /500 mA to X5 on the kitCON-167. Double check the correct polarity. The red LED in the power supply area will light to indicate correct power supply.**
- **Press switch S1 to perform a reset.**

SIEMENS

How to set up your system: DAvE installation

- **Insert the DAvE CD in your CD ROM drive.**
- **Run CD ROM\Setup\Setup.exe.**
- **Follow the setup program's instructions.**
- **If you don't have Acrobat Reader installed on your PC, do so by choosing to install Acrobat Reader which is included on the DAvE CD.**
  (Hint: You need Acrobat Reader 3.0 or higher)
- **Please note:**
  You need WINDOWS95 or WINDOWS NT in order to run DAvE!

SIEMENS

How to set up your system: Keil uVision installation

- **Insert the Starter Kit CD into your CD ROM drive.**
- **Start welcome.pdf from the CD ROM (Acrobat Reader is required but also included on the CD: CD ROM\install\reader\)**
- **Go to Third Party Development Tools**
- **Go to Compiler/Assembler**
- **Go to Keil Elektronik**
- **Choose EK166 installation. The following text will assume that Keil tool chain is installed in the default directory c:\c166eval**
- **To enable dScope to connect to the Starter Kit:**
  - Copy the files “Boot” and “Monitor” from d:\cdrom\startkit\ sk_167\Monitor\Keil to your c:\c166eval\bin directory!
- **Instead of the above you could also use the following invocation (from a DOS window, within c:\c166eval\mon166\):**
  - INSTALL PHY7 0 FEA FEC BOOTSTRAP

SIEMENS

How to set up your system: Exercise directory

- **Create a directory**

  c:\can_hot

  on your hard drive.

- **This directory will be used to store the uVision Project as well as the project file and the C-files generated by DAvE.**

SIEMENS

Hints regarding DAvE

- **To create a new Project with DAvE:**
  - Select “Project | New” from the pull down menu
- **To generate code with DAvE:**
  - Select “Project | Generate Code” from the pull down menu
- **To configure a peripheral:**
  - Move your mouse over the peripheral when DAvE shows the block diagram
  - Click the right mouse button
  - Select Configure
- **To get context specific help in DAvE:**
  - Move your mouse over the item you want to find out about
  - Click the right mouse button
  - Make your choice
- **Validate each alpha numeric entry by pressing ENTER**

SIEMENS

Hints regarding DAvE (cont.)

- **Assembler Startup File**
  - DAvE also incorporates the Assembler Startup Files of the Keil and the Tasking Compilers.
  - Whenever you do a configuration which affects this Startup file, DAvE is altering the Startup File for you.
  - When starting a real project, be sure to include the modified Assembler Startup File when compiling the C-code created by DAvE. For the examples used in this material, it is not necessary to include the Assembler Startup File.

	Keil C166	Tasking C166
Original Startup File name	START167.A66	CSTART.ASM
Modified Startup File name	START.ASM*	START.ASM

* this will be changed to START.A66 in the next DAvE update.

SIEMENS

Hints regarding the Exercises

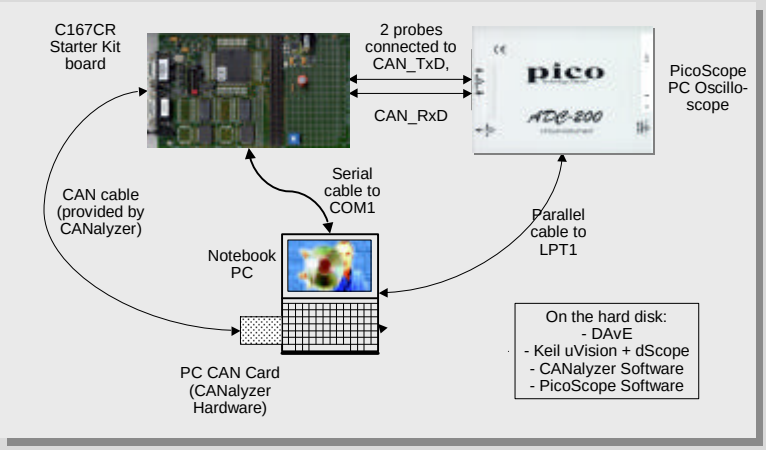
- **If an exercise does not work when running dScope:**
  - Check the oscilloscope connections (right pin?)
  - Check the DAvE / uVision configurations
  - Re-generate the code with DAvE (USER CODE remains!)
  - Recompile the exercise in uVision
  - Reload the exercise into dScope and run it again

SIEMENS

Let's get started now!

SIEMENS

Instructor's Hardware and Software Setup



SIEMENS

Exercise 1 - Transmitting Standard CAN Data Frames

- **Objective:**
  - Generate a Standard CAN (11-bit-Identifier) Message with the on-chip CAN Module
  - Evaluate every identifier bit of incoming messages (only relevant for the following exercises)
  - Use bus speed of 500 kbaud
  - Use Message Object 1
  - Pick an identifier (your choice)
  - Pick a number of data bytes and fill in your data (your choice)
  - Transmit the message every x ms (your choice)
  - Use Timer 6 + the Capture/Reload Register CAPREL of General Purpose Timer Unit 2 (GPT2) to generate this time tick
  - Use the interrupt service routine of Timer 6 to transmit the message

SIEMENS

Exercise 1 - DAvE Configurations

- **Create new Project with microcontroller C167CR**
  - Project name: cantrain
  - Select project path: c:\can_hot\cantrain
- **Project Settings:**
  - General:
    - Select Keil Compiler, SMALL model
  - System Clock:
    - External Oscillator Frequency: Set to 5 MHz
  - Startup Configuration:
    - Bus Type after Reset: Set to 16 bit DEMUX
    - Write Configuration: Pin #WR and #BHE operates as #WRL and #WRH
  - Save & close

SIEMENS

Exercise 1 - DAvE Configurations (cont.)

- **Configure CAN Module:**
  - Control:
    - Global Mask Register: 0x7FF
      (evaluate every bit of incoming identifiers)
  - Baudrate:
    - Baudrate: Enter desired Baudrate: 500 Kbaud
  - Objects:
    - Configure Object 1:
      - Enable Control: Enable Message Object
      - Identifier Selection: Standard 11 bit
      - Message Direction: Transmit Data Frames
      - Data Length Code: Select your choice
      - Data fields: Enter your data bytes
      - Arbitration Register: Enter your 11-bit Identifier
      - Save & close
  - Functions: Include functions CAN_vInit and CAN_vTransmit
  - Save & close

SIEMENS

Exercise 1 - DAvE Configurations (cont.)

- **Configure GPT2:**
  - Core Timer 6:
    - Timer 6 Mode: Timer mode
    - Up/Down Control: Count down
    - Input Selection: Select your prescaler value
    - Timer Register: Load T6 with your time tick value (time ticks * selected resolution = time tick value)
    - Timer Start: Start T6 after initialization
    - Reload Mode: Enable Reload from CAPREL
    - Interrupt Control: Enable Timer 6 interrupt

SIEMENS

Exercise 1 - DAvE Configurations (cont.)

- **Configure GPT2 (cont.)**
  - Capture/Reload:
    - CAPREL Register: Set to time tick value (same as above)
  - Interrupts:
    - Select an interrupt level for GPT2 Timer 6 interrupt
  - Functions:
    - Include GPT2 initialization function GT2_vInit
  - Save & close
- **Generate Code**

SIEMENS

Exercise 1 - uVision Configurations

- **Create new Project: c:\can_hot\can_hot.prj**
- **Edit Project:**
  - Add GT2.C (from c:\can_hot\)
  - Add CAN.C (from c:\can_hot\)
  - Add MAIN.C (from c:\can_hot\)
  - Open all, Save
- **Assembler Options:**
  - Object:
    - Include Debug Information
    - Segmented Mode
    - enable 80C167 Instructions
  - Miscellaneous:
    - Set: enter memory model: `SMALL`

SIEMENS

Exercise 1 - uVision Configurations (cont.)

- **Compiler Options:**
  - Listing: Include Symbols
  - Object:
    - Include Debug Information
    - enable 80C167 Instructions
  - Memory: Memory Model SMALL
- **Linker Options (these are just example values):**
  - Sections:
    - Locate variable initialization information to address 200h:
      Enter `?C_INITSEC (200H)` in section 1 box
  - Location:
    - Reserve interrupt vector locations for Non Maskable Interrupt, Serial Interface Receive Interrupt and Timer 2 Interrupt for monitor use:
      Enter `08h-0Bh, 088h-08Bh, 0ACh-0AFh`
      in Reserve 1 box

SIEMENS

Exercise 1 - uVision Configurations (cont.)

- **Linker Options (cont.)**
  - Classes:
    - Locate startup code for the C main function to address 1000h:
      Enter `ICODE(1000h-09FFFh)` in a class field
    - Locate near function code to address 1000h:
      Enter `NCODE(1000h-09FFFh)` in a class field
    - Locate constant near objects to address 1000h:
      Enter `NCONST(1000h-03FFFh)` in a class field
    - Locate near objects without initialization to address 4000h:
      Enter `NDATA(4000h-07FFFh)` in a class field
    - Locate near objects with initialization to address 4000h:
      Enter `NDATA0(4000h-07FFFh)` in a class field

SIEMENS

Exercise 1 - uVision Configurations (cont.)

- **Environment Pathspecs:**
  - Uncheck Box “Automatically determine Environment Pathspecs”
  - BIN file Directory: C:\C166EVAL\BIN
  - INC file Directory: C:\C166EVAL\INC (filled in by uVision)
  - LIB file Directory: C:\C166EVAL\LIB (filled in by uVision)
  - TMP file Directory: C:\TEMP
    - Be sure to create the directory C:\TEMP on your hard drive or adjust the setting “TMP file directory” to the temporary directory that you are using.
- **Make Options:**
  - After Compile: Run L166 Linker and OH166 (if hex file is needed)

SIEMENS

Exercise 1 - uVision Configurations (cont.)

- **Edit MAIN.C:**
  - include endless loop in main():

```
// USER CODE BEGIN (Main,2)
while(1) {};
// USER CODE END
```

- **Edit GT2.C:**
  - In Timer 6 Interrupt Service Routine (GT2_viIsrTmr6(void)): Transmit Message Object 1:

```
// USER CODE BEGIN (GT2_IsrTmr6,1)

CAN_vTransmit(1);
// Send out CAN message 1

// USER CODE END
```


SIEMENS

Exercise 1 - Running the Program

- ➔ **uVision: Build all**
- ➔ **Run dScope Debugger (from within uVision with the button on the very right of the toolbar)**
- ➔ **File | Load CPU Driver: mon166.dll**
  this will load the Keil Monitor into the kitCON-167's RAM via bootstraploader
- ➔ **File | Load Object file: Load c:\can_hot\cantrain**

SIEMENS

Exercise 1 - Running the Program (cont.)

➔ **Hit Go!**

➔ **Verification:**

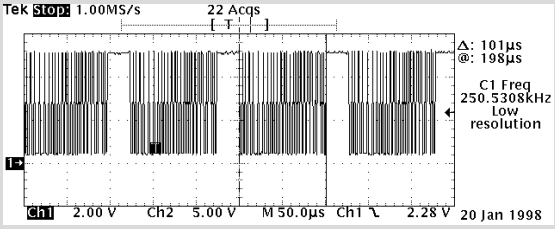
- Connect Oscilloscope to
  - P4.6 / CAN_TxD (connector X3 pin 40, called A22/TXDC)
  - P4.5 / CAN_RxD (connector X3 pin 43, called A21/RXDC)
- Watch your message in the CANalyzer trace window: Verify your identifier and your data bytes
- Verify your time tick value

➔ **If you hit Stop, the application stops in the endless loop in main().**

SIEMENS

Exercise 1 - Screenshot: CAN Message (1Mbit/s)

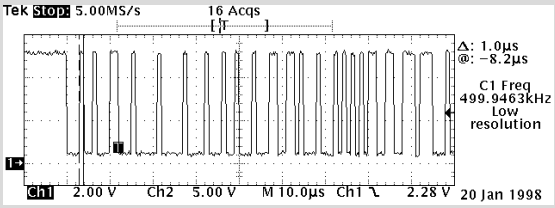
P4.6 / CAN_TxD



50.0 us / Div

situation if no acknowledge received

P4.6 / CAN_TxD



10.0 us / Div

1 us per bit
= 1 Mbit/s

SIEMENS

Exercise 2 - Load new data into a Message Object using the Transmit Interrupt

- **Objective:**
  - Modify Exercise 1
  - Go back to DAvE and enable the CAN module's global interrupt as well as the transmit interrupt of message object 1, so that an interrupt is generated every time a message is transmitted out of message object 1.
  - Increment the data bytes by one in the interrupt service routine after each successful transmission.

SIEMENS

Exercise 2 - DAvE Configurations

- **Open Project from Exercise 1 (should still be open)**
  - (Project path: c:\can_hot\cantrain)
- **Configure CAN Module:**
  - Control:
    - Interrupt Control: Enable CAN interrupt
  - Objects:
    - Configure Object 1:
      - Interrupt Control: Enable Transmit Interrupt
      - Save & close
  - Interrupts:
    - Select an interrupt level for the CAN interrupt
  - Save & close
- **Generate Code**

SIEMENS

Exercise 2 - uVision Configurations

- **uVision Configurations:**
  - Open Project c:\can_hot\cantrain.prj (should still be open)
  - uVision will notify you that another program has modified the open files (DAvE has regenerated the code)
    - say “yes” to reload the files
  - Edit CAN.C: declare loop variable:

```
// USER CODE BEGIN (CAN_General,1)
   unsigned int i;
// USER CODE END
```


SIEMENS

Exercise 2 - uVision Configurations (cont.)

- **uVision Configurations (cont.)**
  - Edit the CAN interrupt service routine in CAN.C (CAN_vIIsr(void) interrupt XP0INT):
    - Increment data bytes in the transmit interrupt part for message 1:

```
case 3: // Message Object 1 Interrupt
// The transmission of the last message
// was successful.
// USER CODE BEGIN (CAN_IsrTxOk,1)
   for (i=0;i<8;i++) CAN_OBJ[0].Data[i]++;
// USER CODE END
```

  - Build all
  - Close opened dScope session (stop the program execution first), then press reset button on board
  - Open new dScope session (icon in uVision)

SIEMENS

Exercise 2 - Running the Program

- **dScope:**
  - Load Keil Monitor into kitCON-167's RAM via bootstraploader:
    File | Load CPU Driver: mon166.dll
  - Load new object file:
    File | Load Object file: c:\can_hot\cantrain
  - Click Go

- **Verification:**
  - Watch your message in the CANalyzer trace window:
    Data bytes should be counting up each message

SIEMENS

Exercise 3 - Receiving Data Frames

- **Objective:**
  - Modify Exercise 2
  - Go back to your DAvE project and configure message object 2 for the reception of standard CAN data frames (11-bit ID) with the identifier 0x120
  - Enable the receive interrupt of message object 2, so that an interrupt is generated every time a message is received in message object 2.
  - React on the received Data Frame in the CAN interrupt service routine by setting a breakpoint to the interrupt service routine

SIEMENS

Exercise 3 - DAvE Configurations

- **Open Project from Exercise 2 (should still be open)**
  - (Project path: c:\can_hot\cantrain)
- **Configure CAN Module:**
  - Objects:
    - Configure Object 2:
      - Enable Control: Enable Message Object
      - Identifier Selection: Standard 11 bit
      - Message Direction: Receive Data Frames
      - Data Length Code: Select 8 data bytes
      - Arbitration Register: Enter the 11-bit Identifier 0x120
      - Interrupt Control: Enable Receive Interrupt
      - Save & close
  - Functions:
    - Include additional function CAN_vGetMsgObj
  - Save & close
- **Generate Code**

SIEMENS

Exercise 3 - uVision Configurations

- **uVision Configurations:**
  - Open Project c:\can_hot\cantrain.prj (should still be open)
  - uVision will notify you that another program has modified the open files (DAvE has regenerated the code)
    - say “yes” to reload the files
  - Edit CAN.C: declare software message object of type TCAN_Obj:

```
// USER CODE BEGIN (CAN_General,2)
   unsigned int i;
   TCAN_Obj message_2;
// USER CODE END
```

SIEMENS

Exercise 3 - uVision Configurations (cont.)

- **uVision Configurations (cont):**
  - Edit the CAN interrupt service routine in CAN.C (CAN_vIsr(void) interrupt XP0INT):
    - Download message object 2 into the software message object message_2 and include a nop-instruction in the receive part for message 2:

```
else
{
// The CAN controller has stored a new
// message into this object.
// USER CODE BEGIN (CAN_IsrRxOk,2)
CAN_vGetMsgObj(2,&message_2);
_nop_();          // for Breakpoint
// USER CODE END
}
```

  - Build all
  - Close opened dScope session (stop program first), reset board
  - Open new dScope session (icon in uVision)

SIEMENS

Exercise 3 - Running the Program

- **dScope:**
  - Load Keil Monitor into kitCON-167's RAM via bootstraploader:
    File | Load CPU Driver: mon166.dll
  - Load new object file:
    File | Load Object file: c:\can_hot\cantrain
  - Set Breakpoint on reception of message 2:
    - In the debug window:
      select Commands | Select source module
    - select the can software module and click close
    - scroll down in the code until you reach your inserted NOP
    - Double click this line to set a breakpoint on it
  - Watch the software message object:
    - Click Setup | Watchpoints
    - Enter Expression: message_2
    - Click Define Watch
    - Click Close
    - Click View | Watch window, resize it so you can see the data
  - Click Go

SIEMENS

Exercise 3 - Running the Program (cont.)

- **Verification:**
  - Watch your debug window in dScope: The breakpoint gets hit (it turns red) and the programs stops every time the message with the identifier 0x120 is sent by the CANalyzer on the instructor's desk (your identifier stops appearing in the CANalyzer trace)
  - Watch the Watch Window in dScope: Data bytes get updated in the structure message_2

SIEMENS

Exercise 4 - Load a Transmit Message Object

- **Objective:**
  - Modify Exercise 3
  - Go back to your DAvE project and configure message object 3 for the transmission of extended CAN data frames (29-bit ID) with an identifier of your choice
  - React on the received Data Frame in the CAN interrupt service routine by setting a breakpoint to the interrupt service routine
  - In the interrupt service routine for message object 2, load the data bytes of the transmit message object 3 with the data bytes of receive message object 2 and send them "back" to the CANalyzer

SIEMENS

Exercise 4 - DAvE Configurations

- **Open Project from Exercise 3 (should still be open)**
  - (Project path: c:\can_hot\cantrain)
- **Configure CAN Module:**
  - Objects:
    - Configure Object 3:
      - Enable Control: Enable Message Object
      - Identifier Selection: Extended 29 bit
      - Message Direction: Transmit Data Frames
      - Data Length Code: Select 8 data bytes
      - Arbitration Register: Enter your 29-bit Identifier
      - Save & close
  - Functions:
    - Include additional function CAN_vLoadData
  - Save & close
- **Generate Code**

SIEMENS

Exercise 4 - uVision Configurations

- **uVision Configurations:**
  - Open Project c:\can_hot\cantrain.prj (should still be open)
  - uVision will notify you that another program has modified the open files (DAvE has regenerated the code)
    - say “yes” to reload the files

SIEMENS

Exercise 4 - uVision Configurations (cont.)

- **uVision Configurations (cont):**
  - Edit the CAN interrupt service routine in CAN.C (CAN_vilsr(void) interrupt XP0INT):
    - Load data bytes of message object 3 with the data bytes from the software message object message_2 (replace the NOP) and transmit message object 3:

```
// The CAN controller has stored a new
// message into this object.
// USER CODE BEGIN (CAN_IsrRxOk,2)
CAN_vGetMsgObj(2,&message_2);
CAN_vLoadData(3,&message_2.ubData[0]);
CAN_vTransmit(3);
// USER CODE END
```

  - Build all
  - Close opened dScope session (stop program first), reset board
  - Open new dScope session (icon in uVision)

SIEMENS

Exercise 4 - Running the Program

- **dScope:**
  - Load Keil Monitor into kitCON-167's RAM via bootstraploader: File | Load CPU Driver: mon166.dll
  - Load new object file: File | Load Object file: c:\can_hot\cantrain
  - Click Go
- **Verification:**
  - Make sure that your extended message is sent on reception of the 0x120 message from the CANalyzer (verify identifier)
  - Verify whether the data bytes have the right contents

SIEMENS

Exercise 5 - Transmitting a Remote Frame and Receiving the corresponding Data Frame

- **Objective:**
  - Modify Exercise 4
  - Go to uVision and change the Timer 6 interrupt service routine to transmit message 2 instead of message 1
  - This will generate a remote frame to be transmitted out of message object 1 with the identifier 0x120
  - React on the received corresponding Data Frame in the CAN interrupt service routine by setting a breakpoint to the interrupt service routine

SIEMENS

Exercise 5 - uVision Configurations

- **uVision Configurations:**
  - Open Project c:\can_hot\cantrain.prj (should still be open)
  - Edit GT2.C:
    - In Timer 6 Interrupt Service Routine (GT2_viIsrTmr6(void)): Transmit Message Object 2:

```
// USER CODE BEGIN (GT2_IsrTmr6,1)
   CAN_vTransmit(2);
// Send out CAN message 2 (Remote Frame)
// USER CODE END
```

SIEMENS

Exercise 5 - uVision Configurations

- **uVision Configurations:**
  - Edit the CAN interrupt service routine in CAN.C (CAN_vIlsr(void) interrupt XP0INT):
    - insert NOP (for breakpoint) into receive interrupt service routine for message 2:

```
// USER CODE BEGIN (CAN_IsrRxOk,2)
CAN_vGetMsgObj(2,&message_2);
CAN_vLoadData(3,&message_2.ubData[0]);
CAN_vTransmit(3);
_nop_();
// USER CODE END
```

  - Build all
  - Close opened dScope session (stop program first), reset board
  - Open new dScope session (icon in uVision)

SIEMENS

Exercise 5 - Running the Program

- **dScope:**
  - Load Keil Monitor into kitCON-167's RAM via bootstraploader:
    File | Load CPU Driver: mon166.dll
  - Load new object file:
    File | Load Object file: c:\can_hot\cantrain
  - Set Breakpoint on reception of message 2:
    - In the debug window:
      select Commands | Select source module
    - select the can software module and click close
    - scroll down in the code until you reach your inserted NOP
    - Double click this line to set a breakpoint on it
  - Click Go

SIEMENS

Exercise 5 - Running the Program (cont.)

- **Verification:**
  - Watch your debug window in dScope: The breakpoint gets hit (it turns red) and the programs stops, every time the remote frame with the identifier 0x120 is sent and answered by the CANalyzer on the instructor's desk (your identifier stops appearing in the CANalyzer trace)
  - If you want you can additionally watch the data bytes get updated in the structure message_2 (in the Watch Window)
  - Watch your messages in the CANalyzer trace window: There should be your remote frame, the answer from the CANalyzer and your extended frame.
  - Please note: With this exercise we are violating the CAN protocol because we have several stations that transmit the same identifier (0x120). This is not allowed in a real system because it can lead to bus collisions.