# CAN Software Example

## AT90CAN32/64/128

# "Illustration by example"

## Develop an Atmel AVR CAN project is:

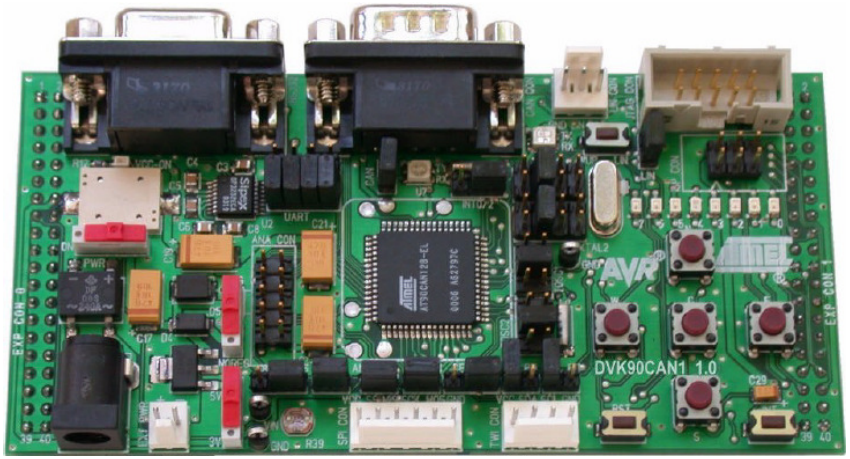
- easy,
- fast &
- efficient.

# Content

- … Example Description
  - … Compiling
  - … Simulation
- … Hardware Debug

# Example Description (1)

- Network
  - 2 Nodes

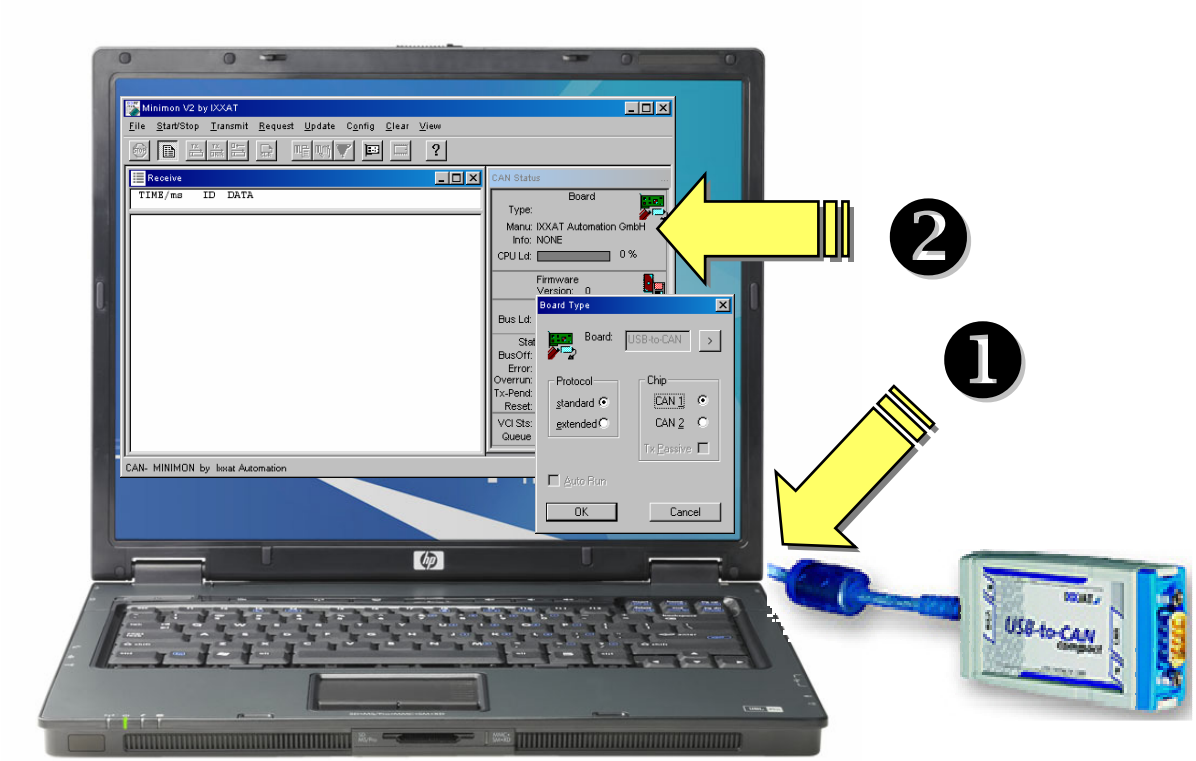



**DVK90CAN1**
**Atmel board**

**USB to CAN**
**iXXAT dongle**

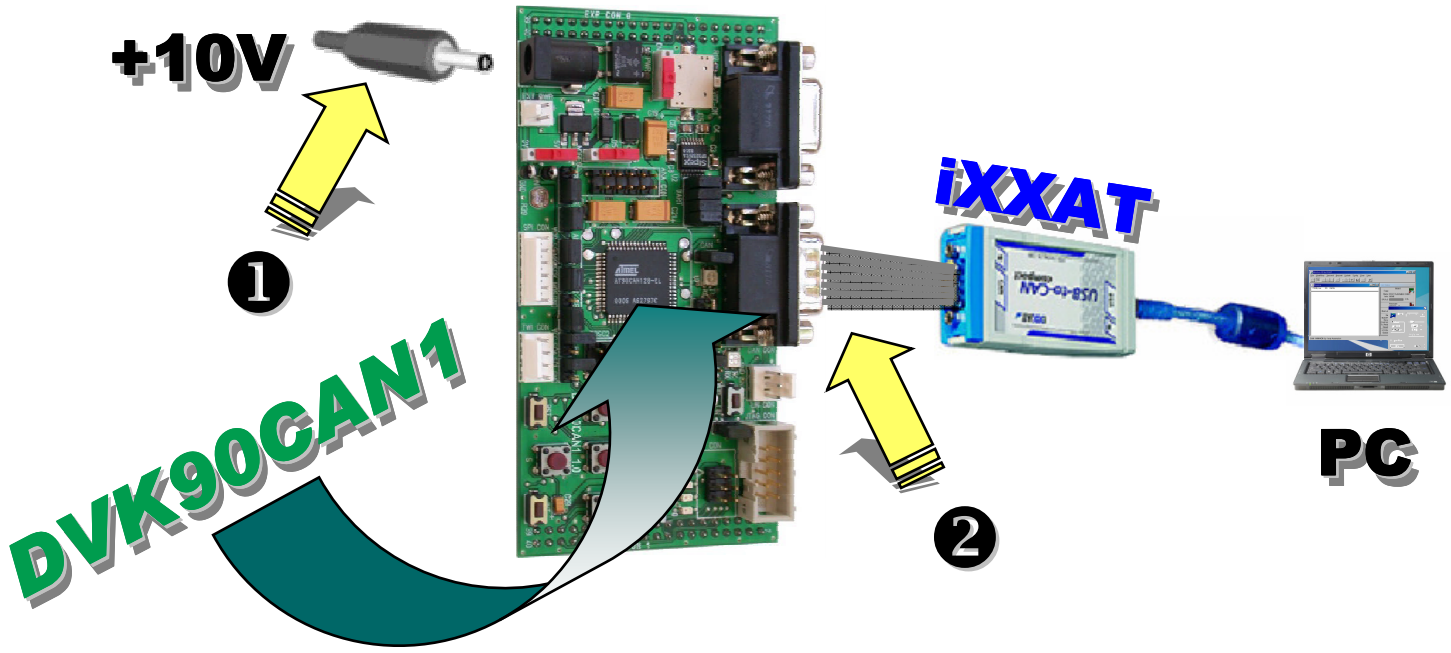
# Example Description (2)

- **IXXAT Dongle**
  - ❶ - Connected to your PC
  - ❷ - Using Minimon V2 (minmon32.exe), IXXAT tool



# Example Description (3)

- **Atmel Board**
  - ❶ - Powered
  - ❷ – Connected to IXXAT Dongle



# Example Description (4)

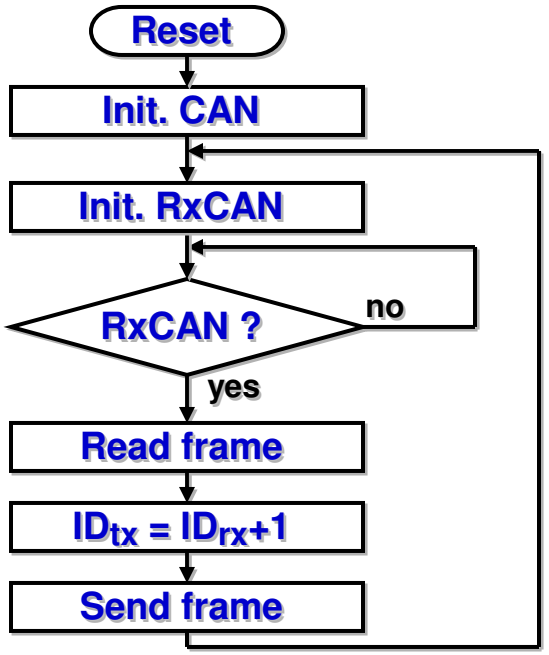
## Program Specification

❶ – **Wait** for CAN frame

❷ – **Read** the CAN frame

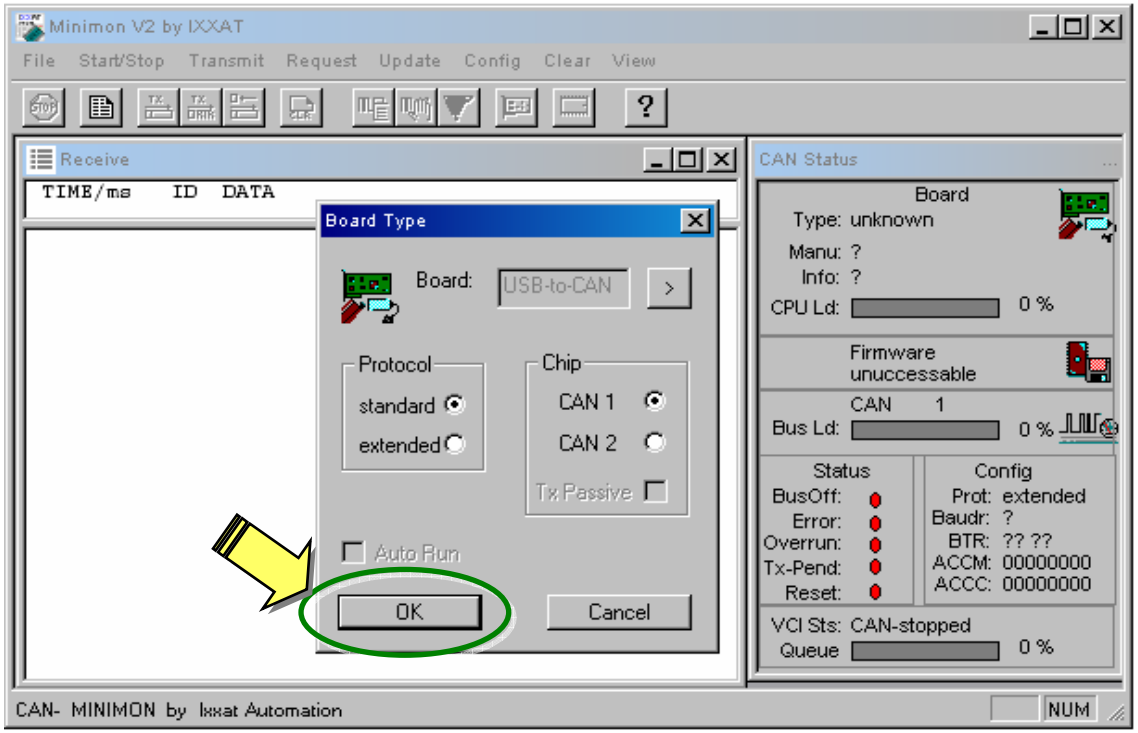
❸ – **Re-send** this CAN frame with $ID_{Tx}=ID_{Rx}+1$

❹ – Go to ❶



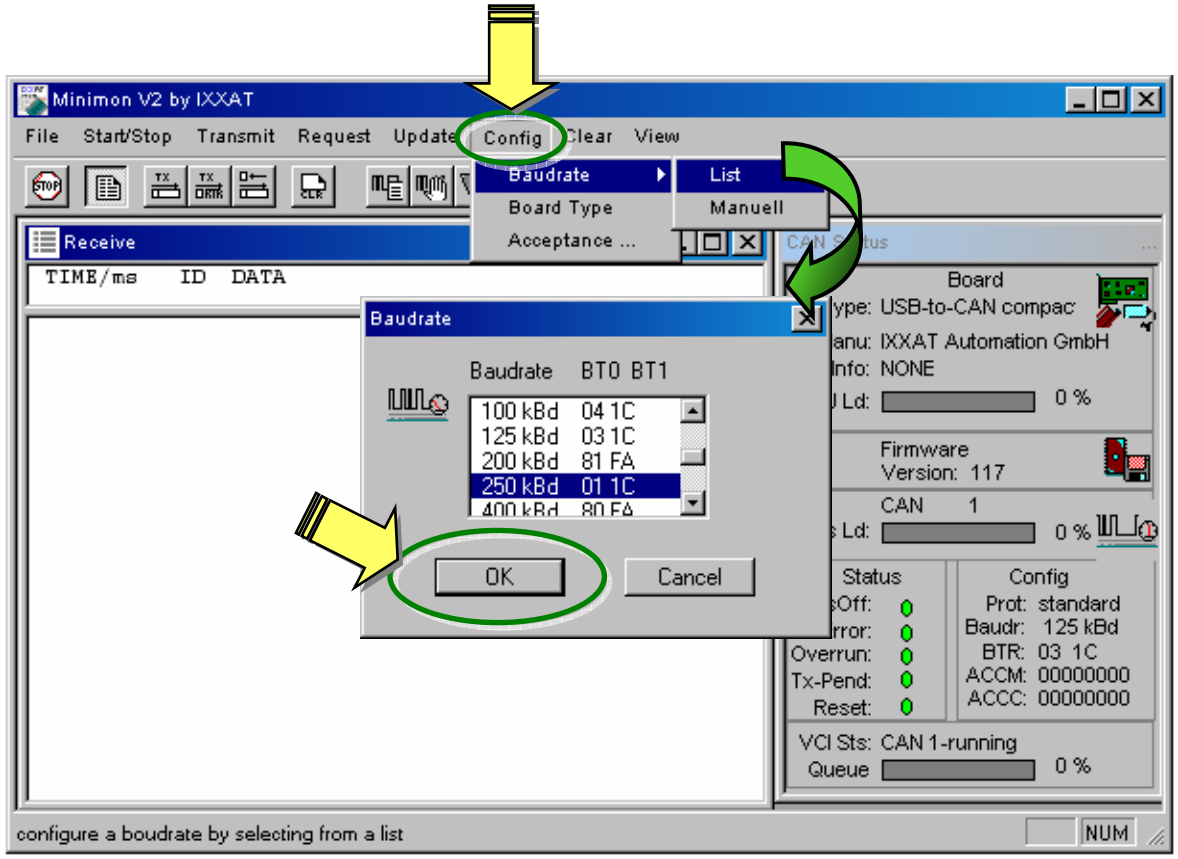
# Example Description (5)

- **Application**
  - **Minimon V2**
    - Initialization



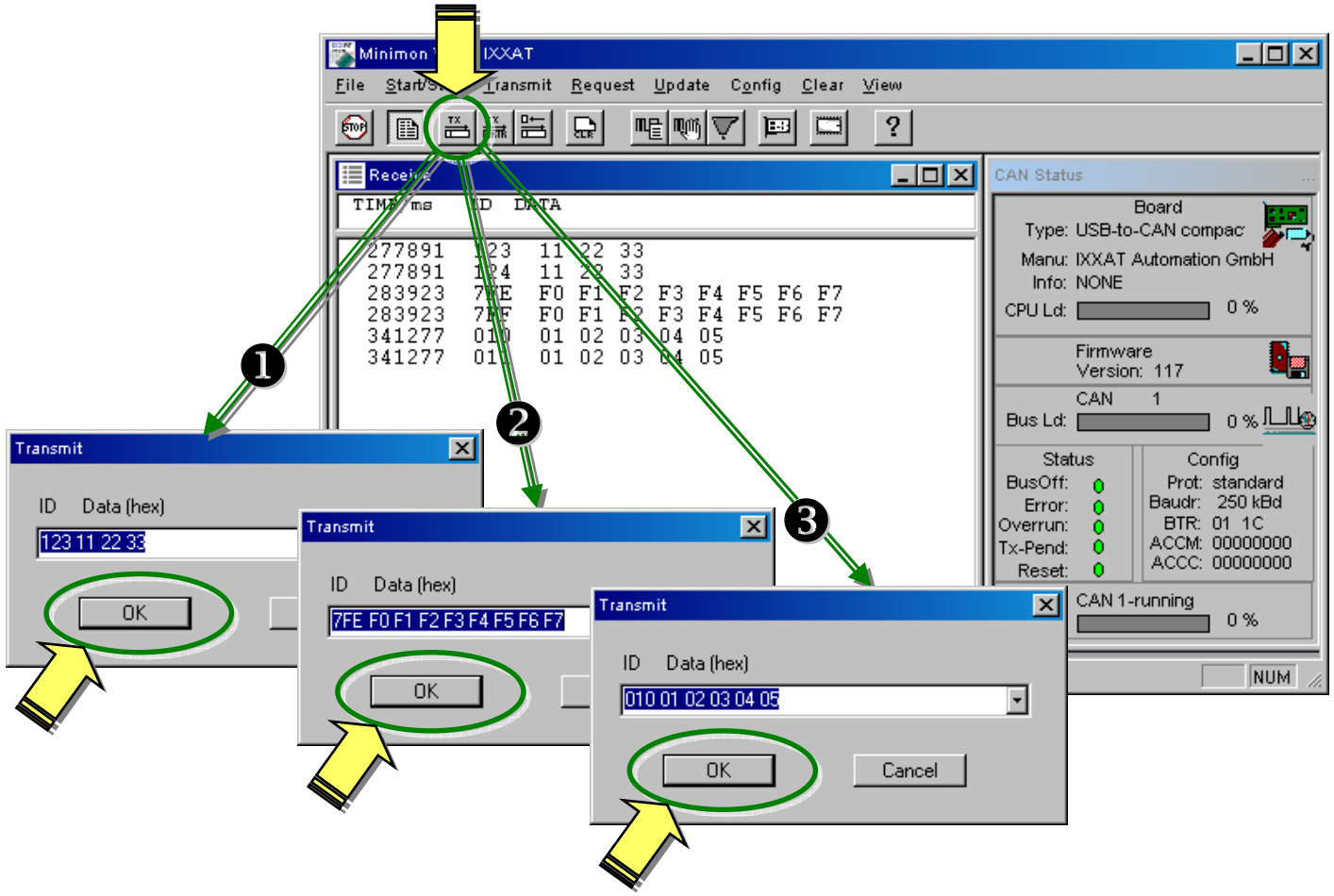
# Example Description (6)

- Baudrate Setting – 250 kBd



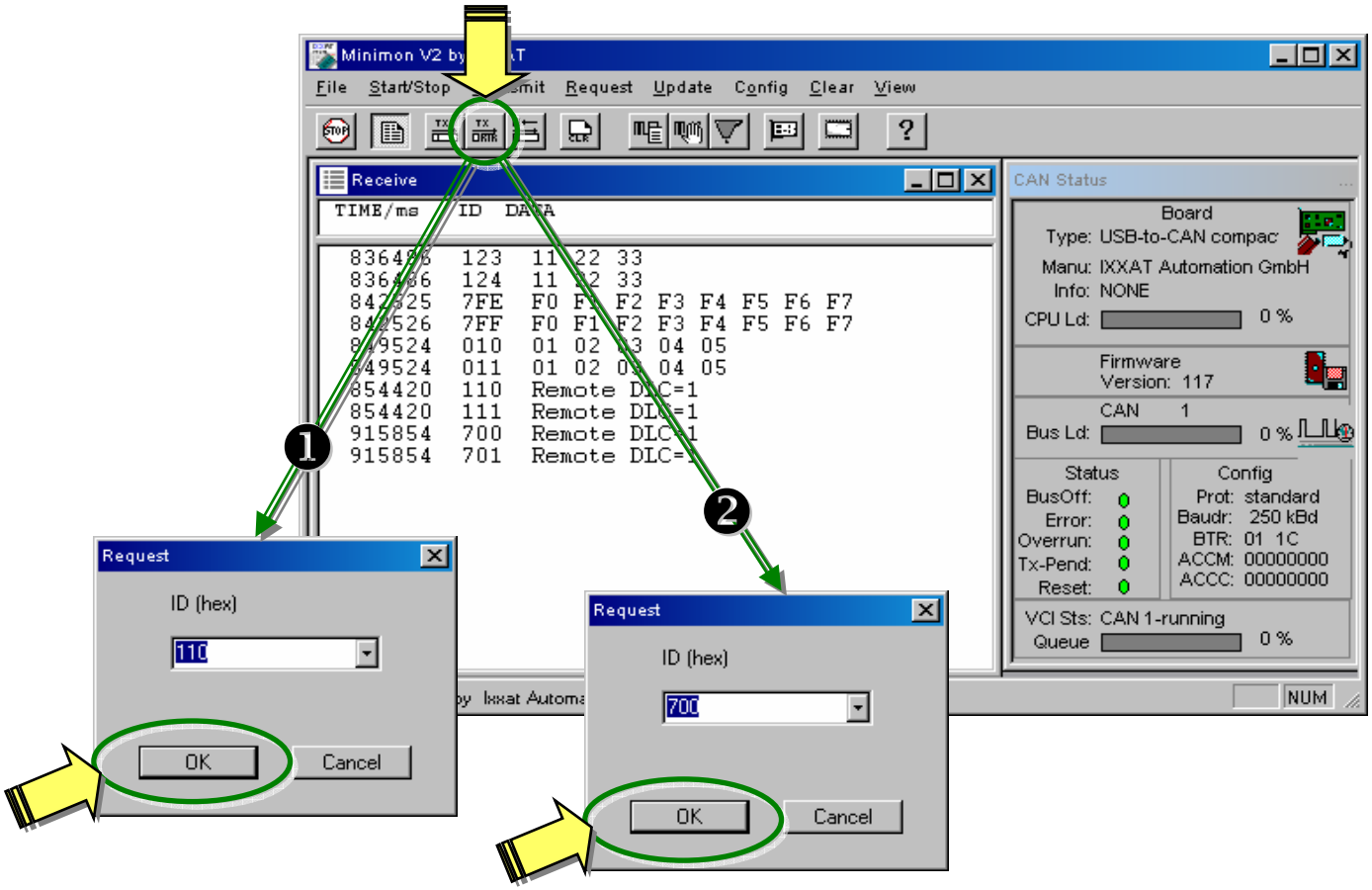
# Example Description (7)

- **CAN Data Frames Generation**



# Example Description (8)

- **CAN Remote Frames Generation**



# Compiling (1)

## AVR Studio® 4 IDE - Freeware

☑ AVR Studio® 4 is a professional **I**ntegrated **D**evelopment **E**nvironment (**IDE**) for writing, programming and debugging AVR® applications in Windows® 9x/NT/2000/XP environments.

☑ AVR Studio® 4 includes an assembler and a simulator.

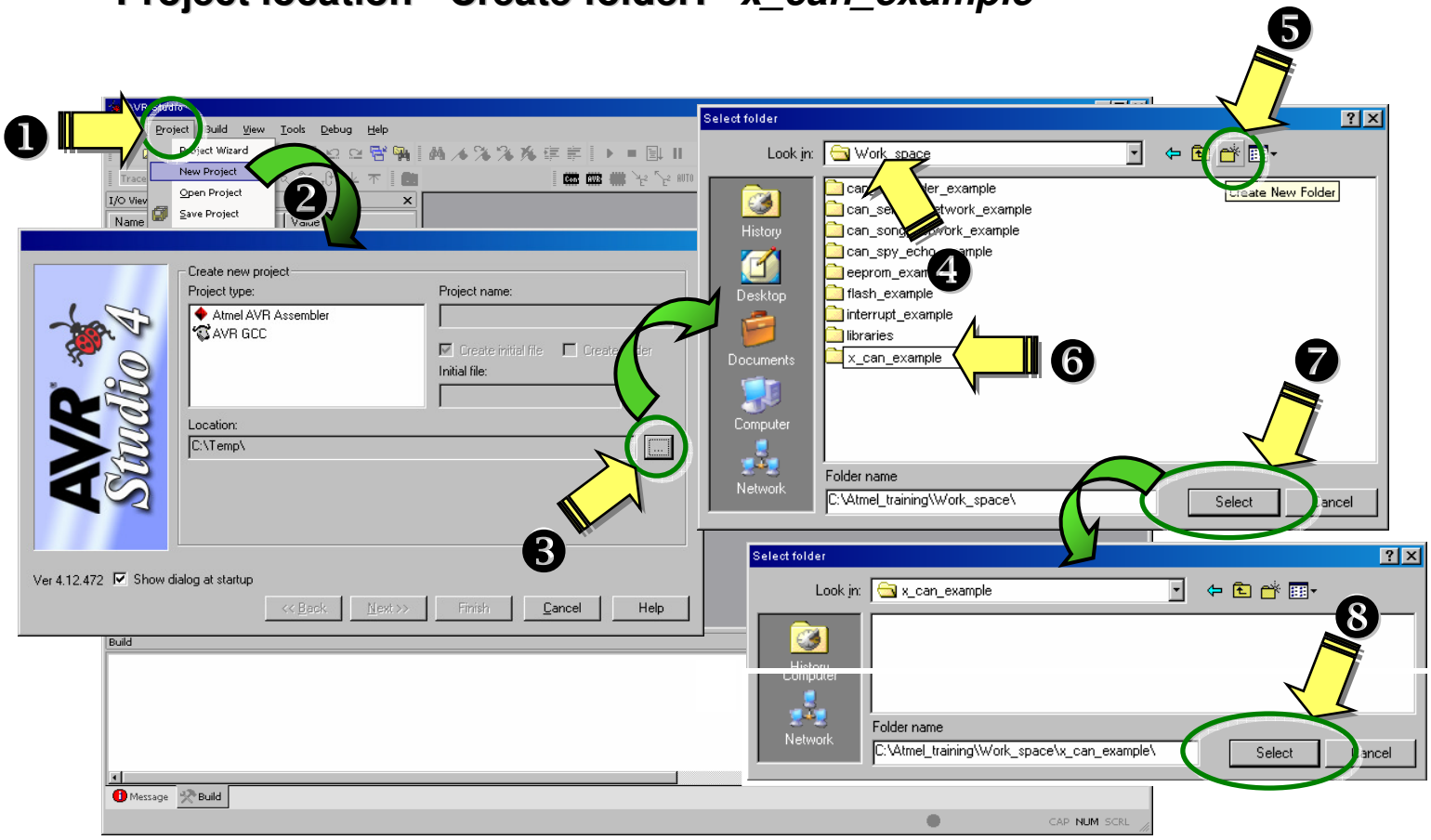
☑ AVR Studio® 4 is also able to include plug-in such as "Compiler plug-in for AVR-GCC" or "CAN plug-in for AT90CANxx family ".

☑ AVR Studio® 4 supportes all AVR development tools such as ICE50, JTAGICE mkII, STK500/501 & AVRISP mkII.

☑ AVR Studio® 4 is a freeware.

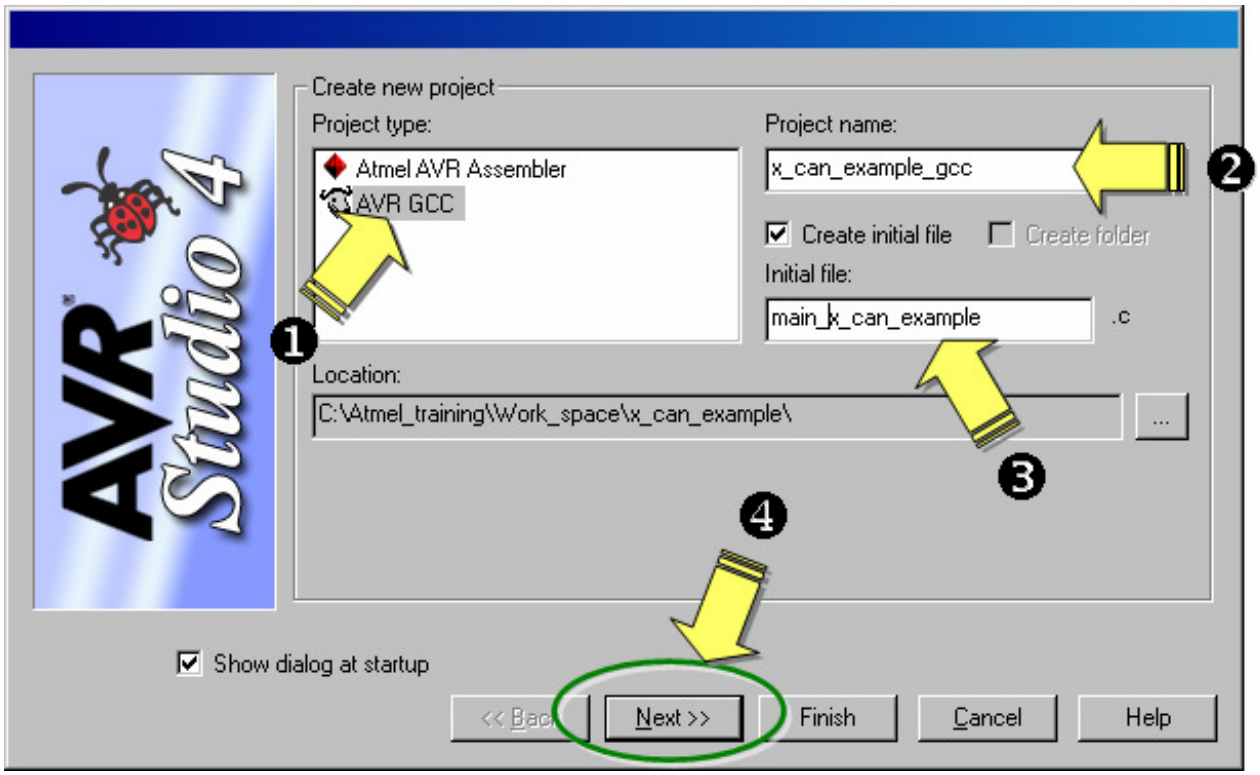
# Compiling (2)

- **Create a new project**
  - **Project location - Create folder: "*x_can_example*"**



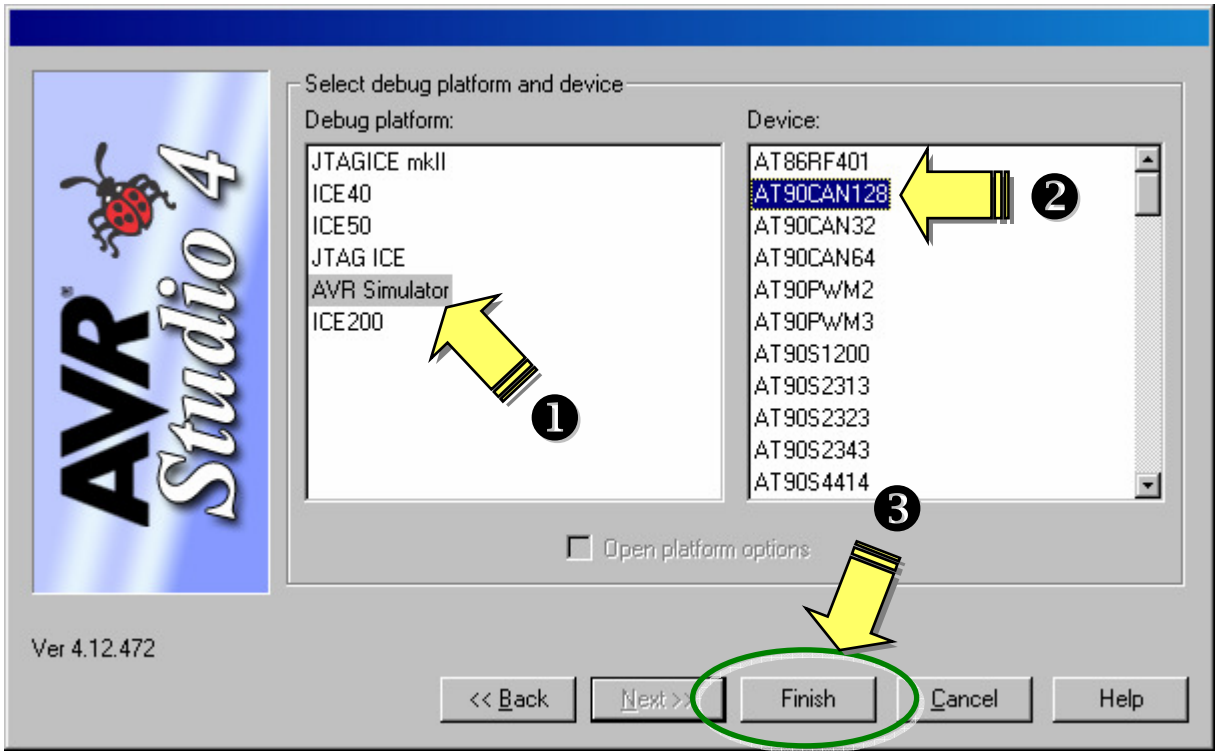
# Compiling (3)

- **Project name: ‘‘*x_can_example_gcc.c*’’**
- **Create initial file: ‘‘*main_x_can_example.c*’’**



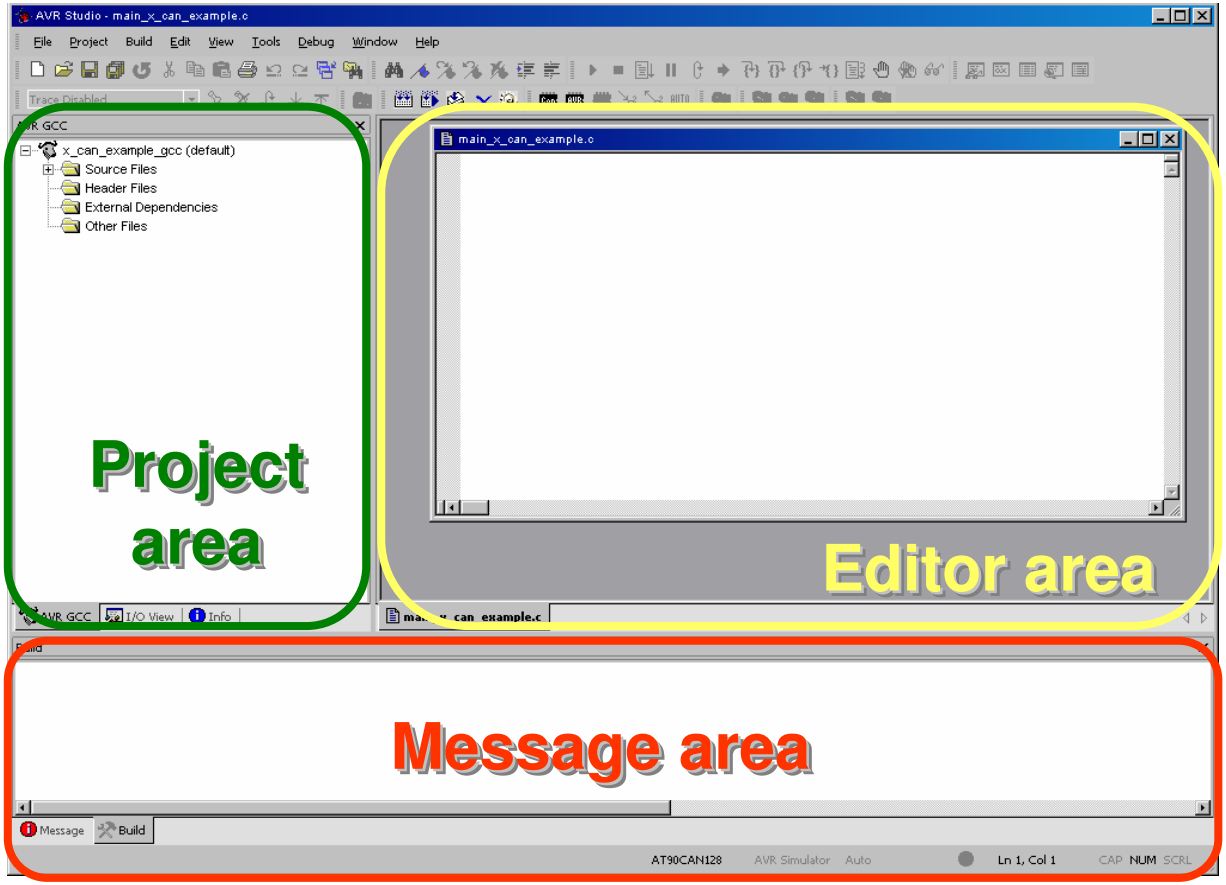
# Compiling (4)

- **Select platform – simulator & device – AT90CAN128**



# Compiling (5)

- Using the Editor
  - Project & editor areas



# Compiling (6)

- Write the main file: "*main_x_can_example.c*"

```
//
//***** main_x_can_example.c ********************************************

#include "config.h"
#include "can_lib.h"

int main (void)
{
st_cmd_t       x_can_message;         // CAN message descriptor
Unsigned char  x_can_buffer[8];       // CAN message buffer
 // Initialization
    x_can_message.pt_data = &x_can_buffer[0];
    can_init(0);
    while(1)
    {
     // RxCAN
        x_can_message.cmd = CMD_RX;     // Enable Rx
        while(can_cmd(&x_can_message) != CAN_CMD_ACCEPTED);
        // Wait for Rx completed
        while(can_get_status(&x_can_message) == CAN_STATUS_NOT_COMPLETED);
     // TxCAN
        x_can_message.id.std++;         // ID Incrementation
        x_can_message.cmd = CMD_TX;     // Enable Tx
        while(can_cmd(&x_can_message) != CAN_CMD_ACCEPTED);
        // Wait for Tx completed
        while(can_get_status(&x_can_message) == CAN_STATUS_NOT_COMPLETED);
    }
    return 0;
}
```

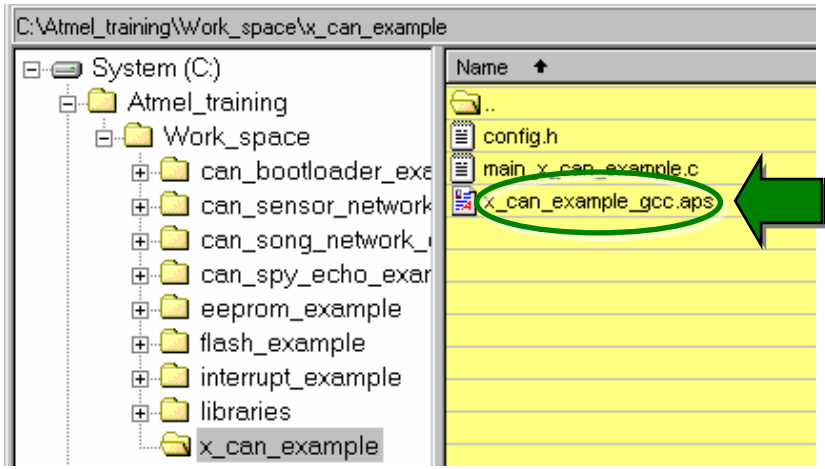
RxCAN

TxCAN

# Compiling (7)

- **Create a new file in the project folder: "*config.h*"**
- **Write the configuration file: "*config.h*"**

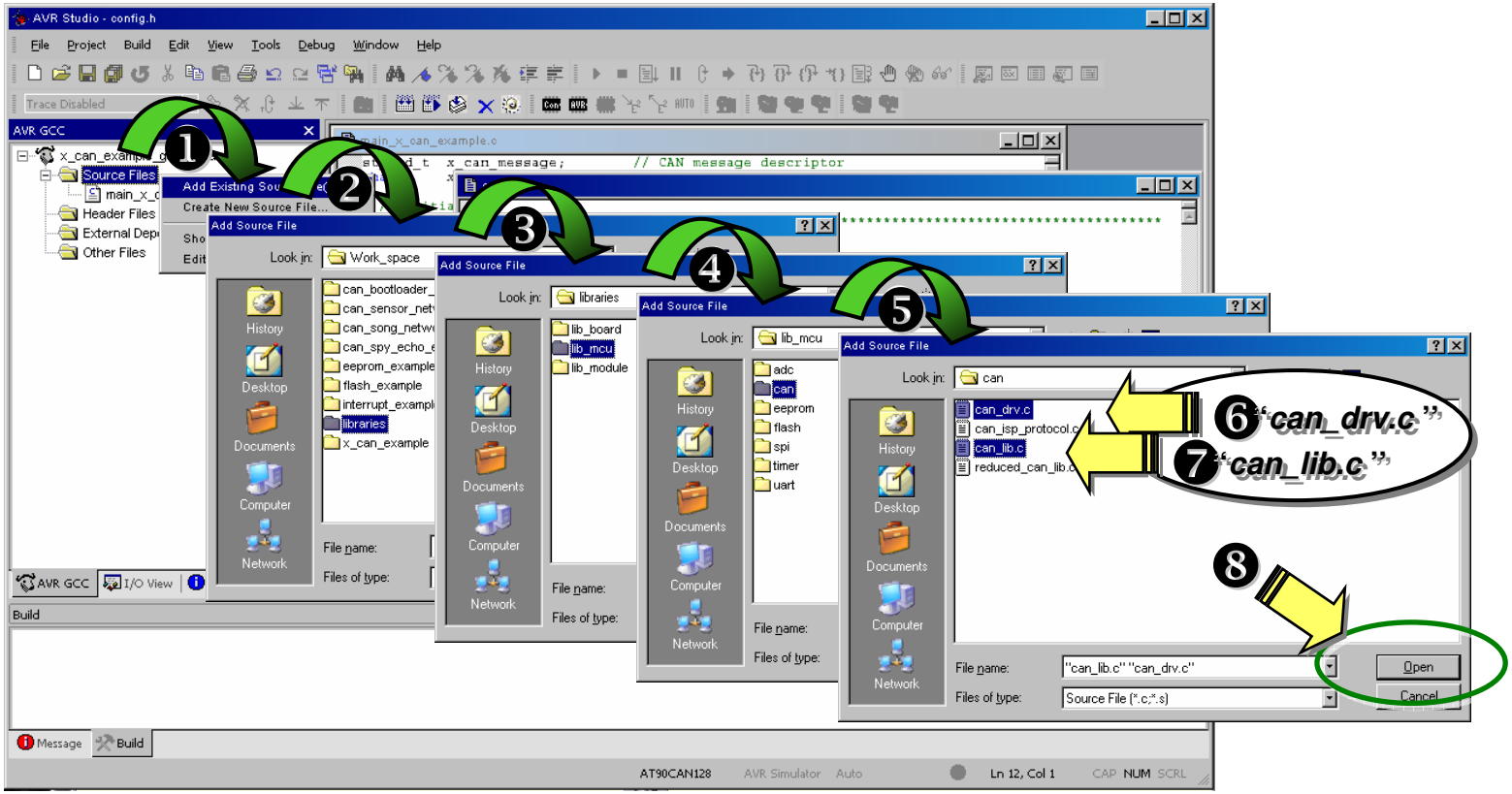
```
//
//***** config.h ***********************************************************

#include <avr/io.h>
#include "compiler.h"

// MCU Configuration
#define FOSC            8000        // 8.000 MHz external crystal
#define F_CPU           (FOSC*1000) // Need for AVR GCC
// CAN Configuration
#define CAN_BAUDRATE    250         // in kBit
```




**Project File**
**(automatic generation)**

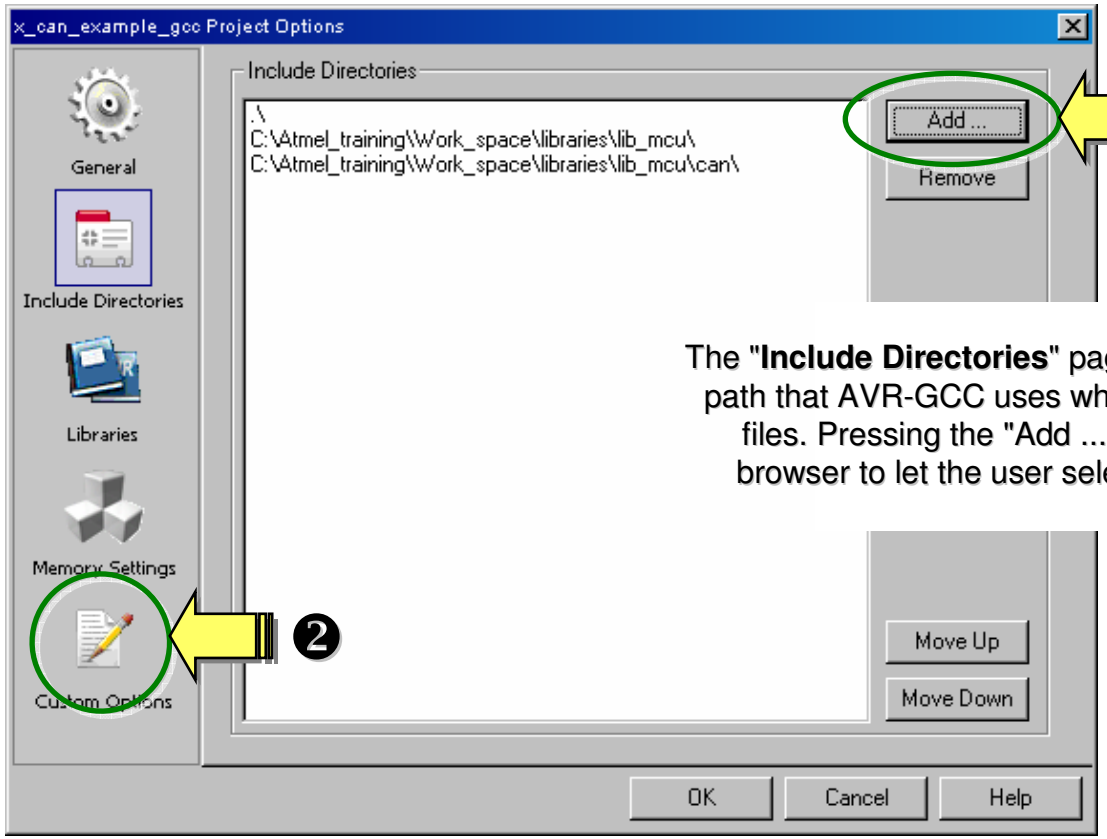
# Compiling (8)

- **Build the project**
  - **Add existing source files – CAN library C-files**



# Compiling (9)

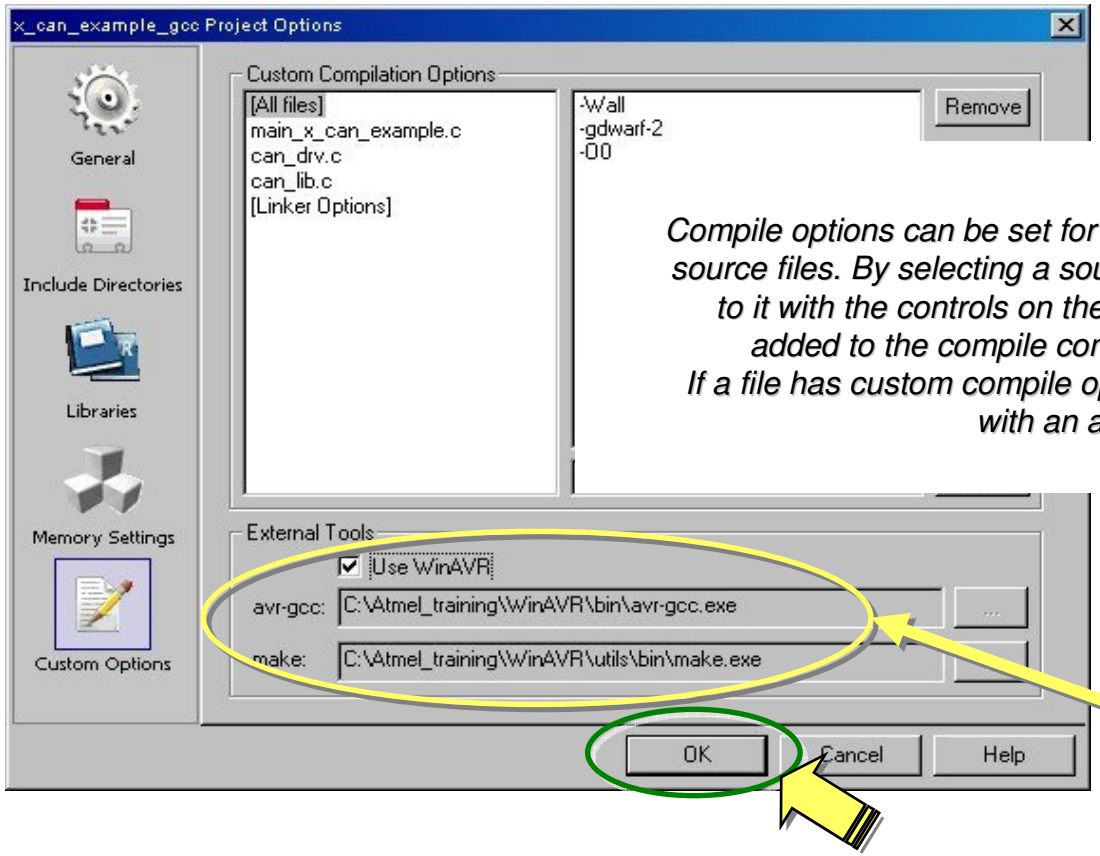
- **Include directories**
  - project, lib_mcu & can directories




The "**Include Directories**" page sets up the search path that AVR-GCC uses when looking up include files. Pressing the "Add ..." button opens a path browser to let the user select a directory to add.

# Compiling (10)

- **Custom options**
  - No custom option - verify the WinAVR path

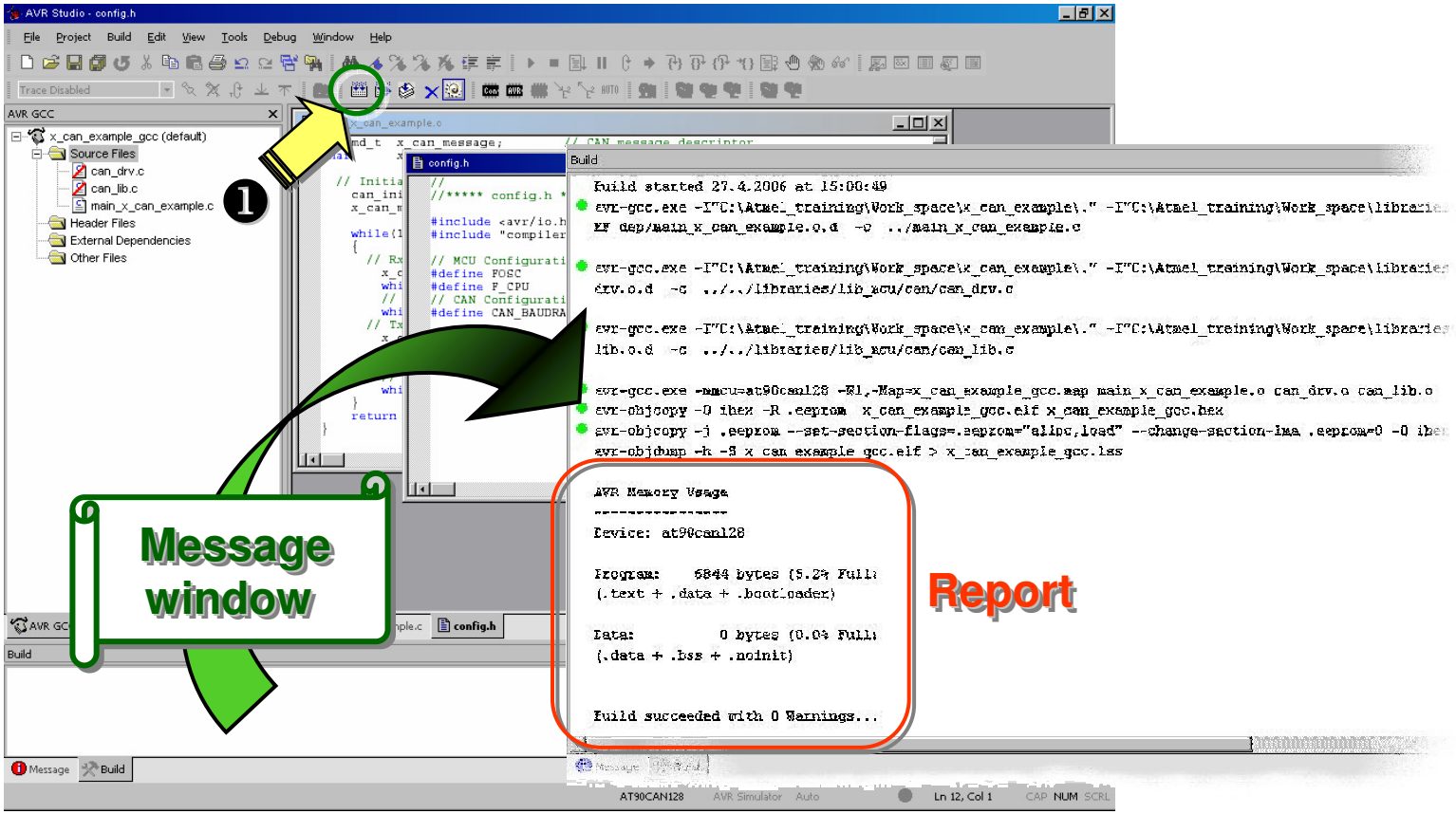



*Compile options can be set for each separate file or for all source files. By selecting a source file and adding options to it with the controls on the right, these options will be added to the compile command for the selected file. If a file has custom compile options set, it will be marked with an asterisk in the list to the left.*

**Verify it before to start !**

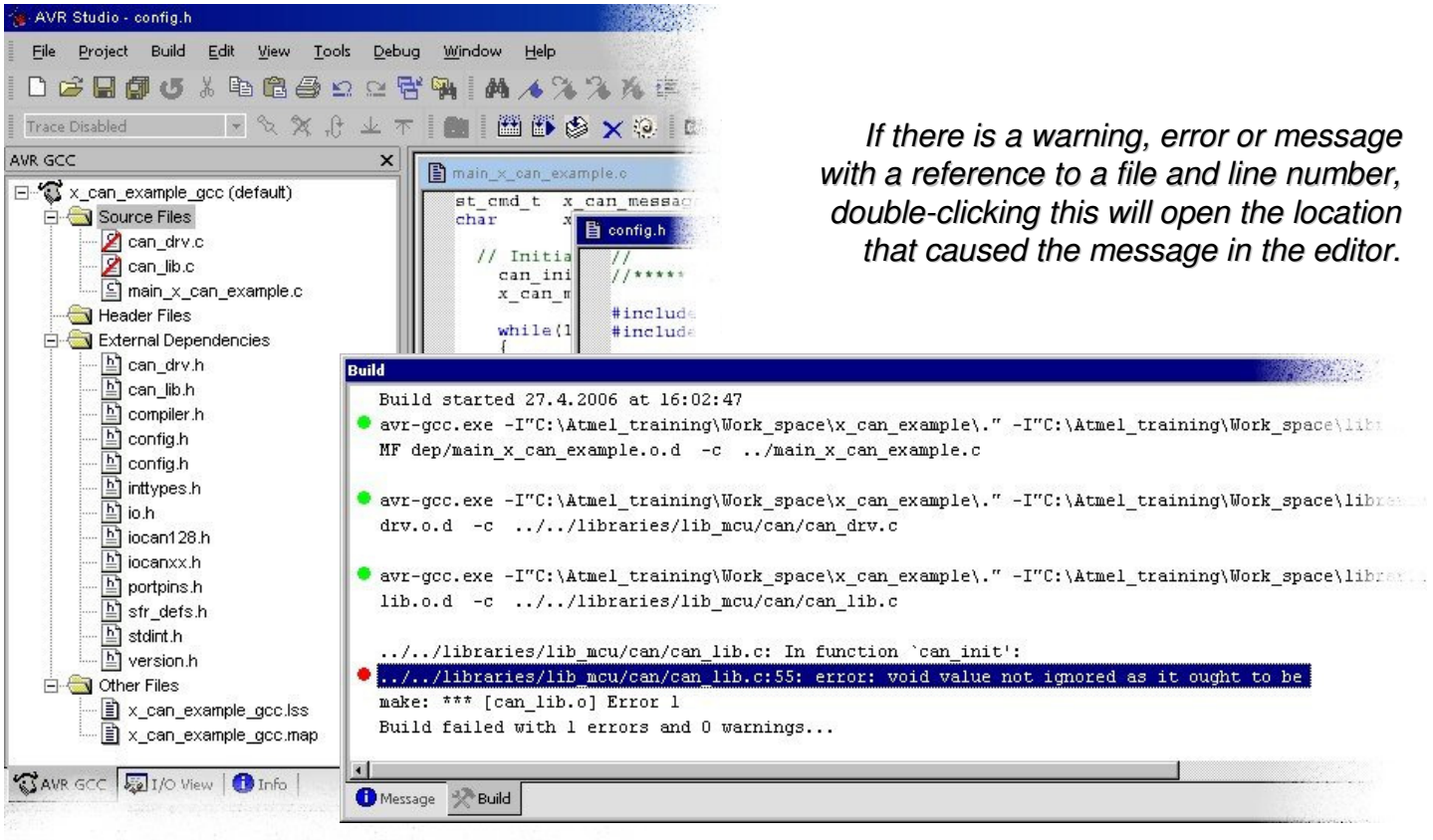
# Compiling (11)

- Invoque GCC
  - Built active configuration



# Compiling (12)

- Warning or error
- (new) Project tree

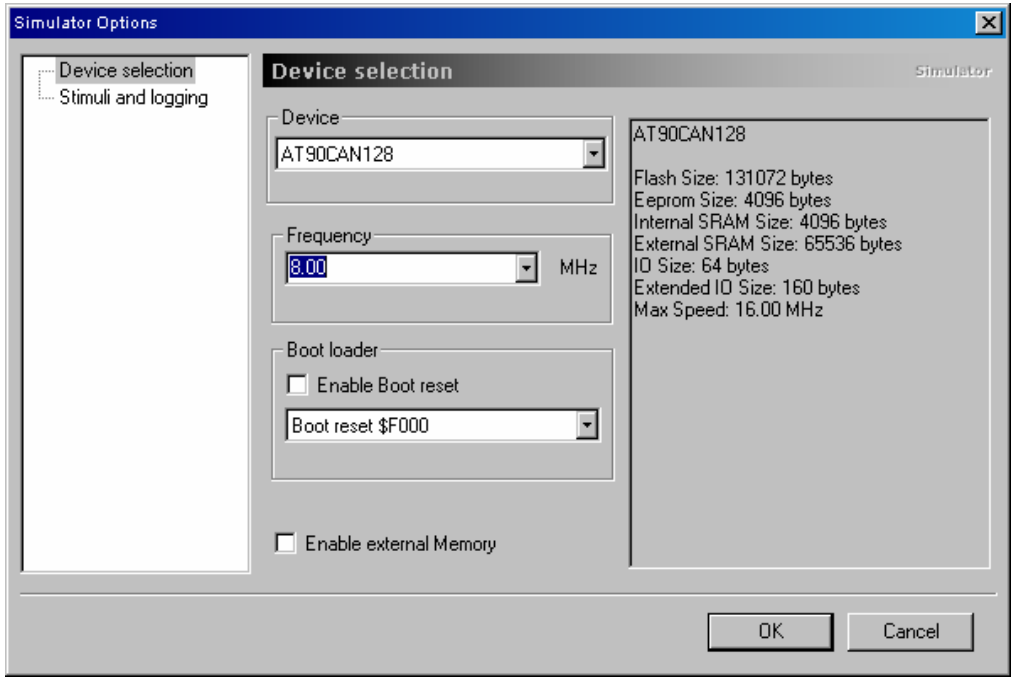



*If there is a warning, error or message with a reference to a file and line number, double-clicking this will open the location that caused the message in the editor.*

# Simulation (1)

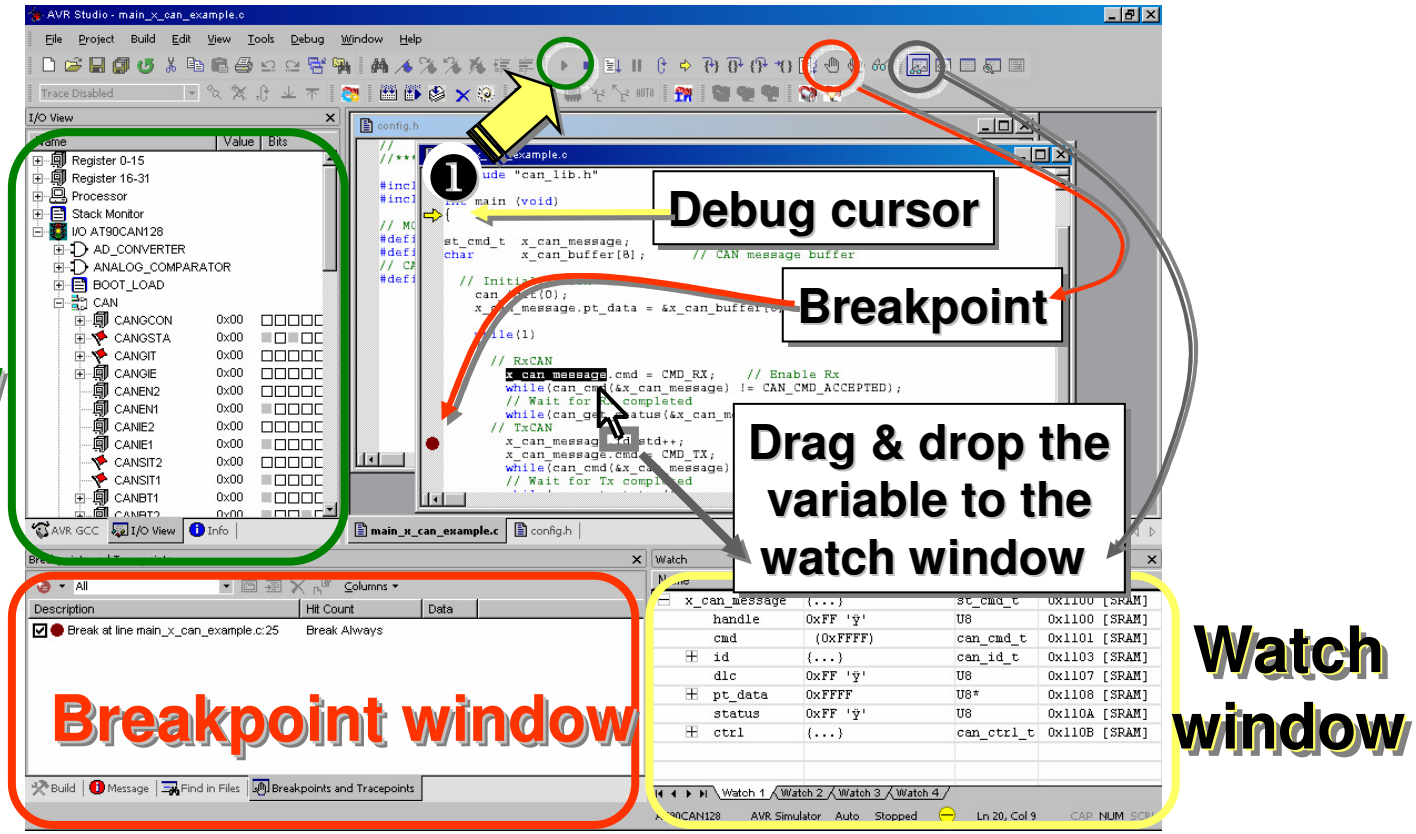
The AVR Simulator simulates the CPU, including all instructions, interrupts and most of the on-chip I/O modules. The AVR Simulator operates within the AVR Studio application as a debug target. This enables the user to use the normal debug commands such as Run, Break, Reset, Single step, set breakpoints and watch variables. The I/O, memory and register views are fully functional using the AVR Simulator.

## Simulator options



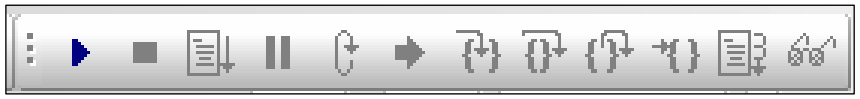
# Simulation (2)

- Debugging (start)
  - Watch window, Code breakpoint & I/O view



# Simulation (3)

- **Debug tools**



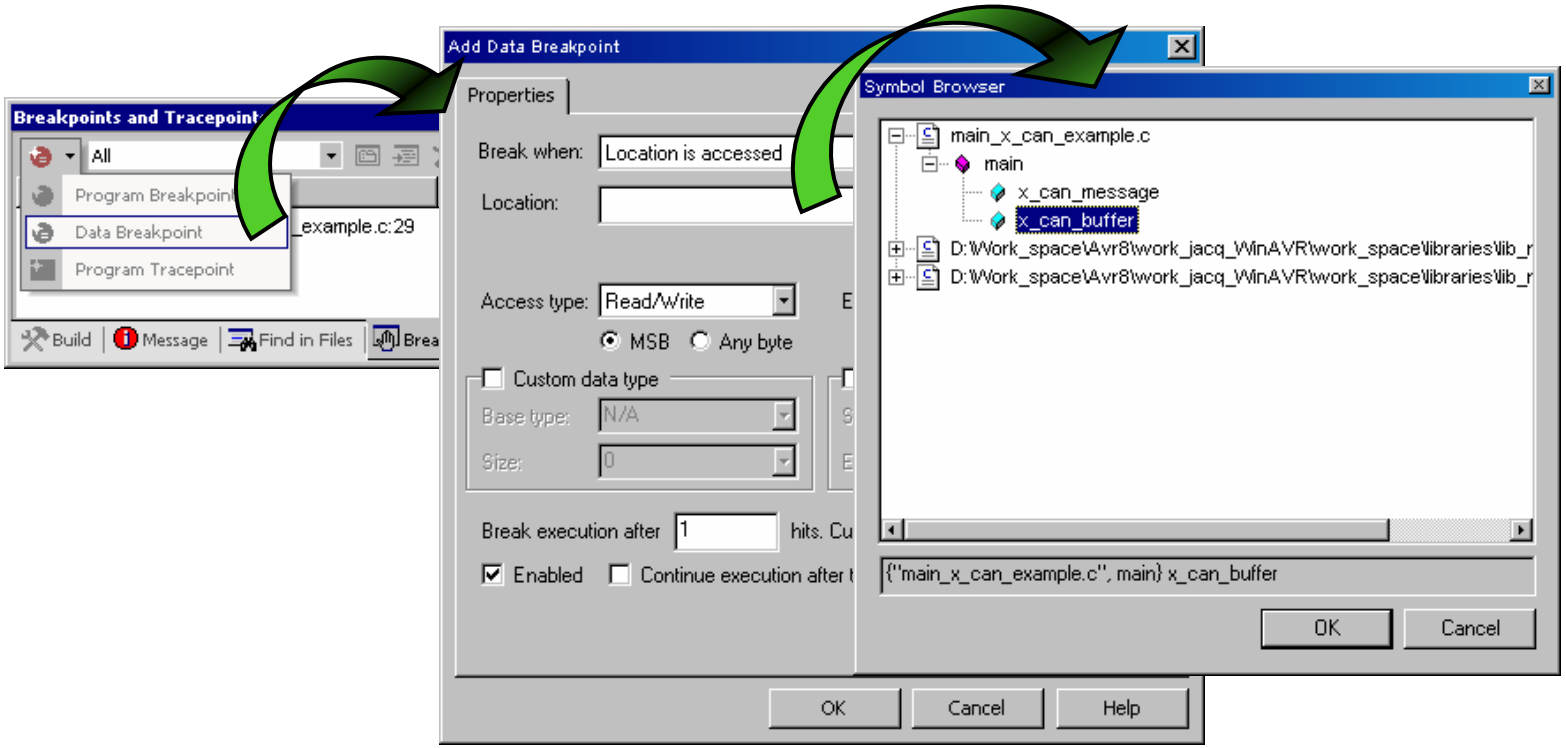
| | |
|---|---|
| Start Debugging | Single step, Trace Into (F11) |
| Stop Debugging | Step Over (F10) |
| Run (F5) | Step Out (SHIFT+F11) |
| Break (CTRL-F5) | Run To Cursor (F7) |
| Reset (SHIFT+F5) | Auto Step |
| Show next statement | Quick watch |

# Simulation (4)

- **Breakpoints**
  - Code breakpoint

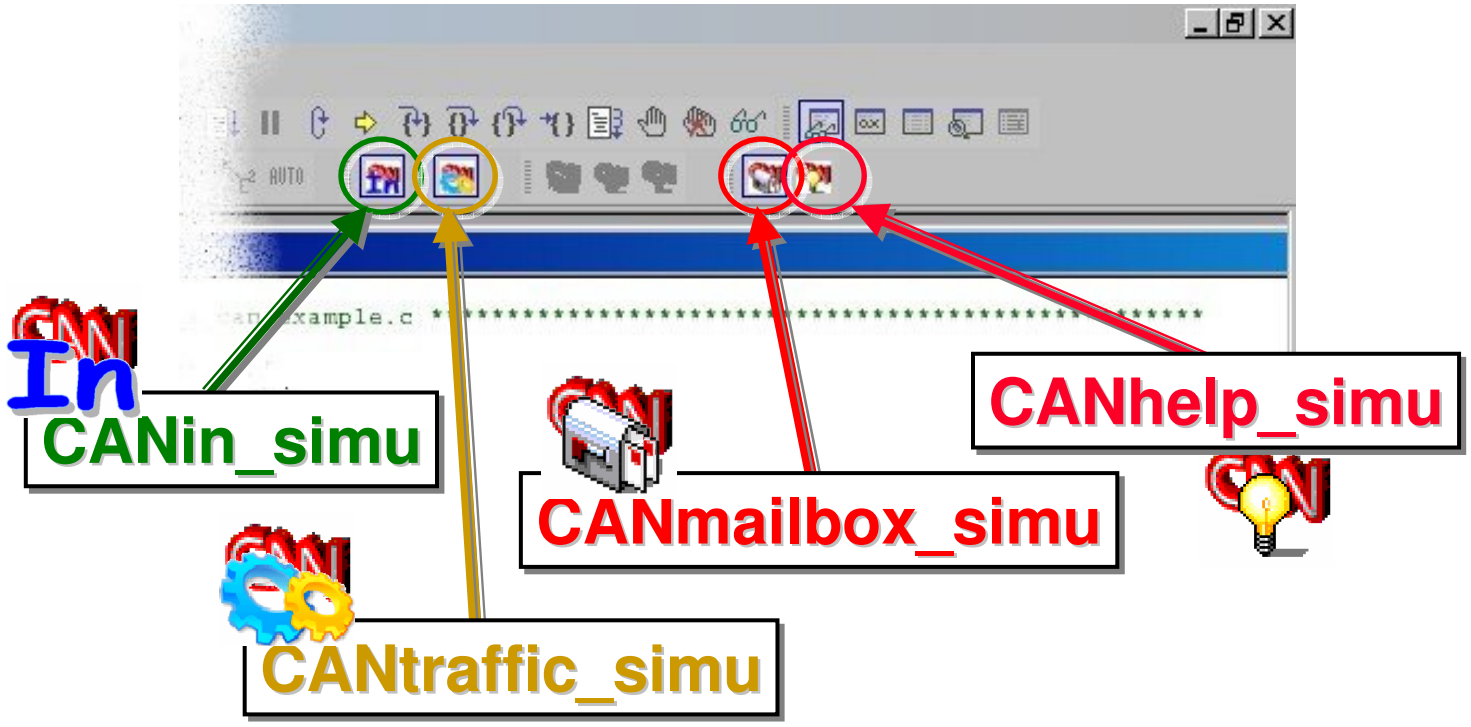
    Toggle program breakpoint (F9)  Clear all program breakpoints

  - Data breakpoint



# Simulation (5)

- CAN plug-in
  - CAN tools



# Simulation (6)

- **CAN simulation**
  - CANin_sim (C:\Program Files\Atmel\AVR Tools\AvrStudioPlugin\AT90CAN128\canin.txt),

CANIN_SIM

| ID | Len | Data | Start | Repeat | Error |
|---|---|---|---|---|---|
| 123 | 4 | 11 22 33 44 | 6000 | 0 | N |
| 7FE | 8 | F0 F1 F2 F3 F4 F5 F6 F7 | 14000 | 0 | N |
| 10 | 5 | 01 02 03 04 05 | 26000 | 0 | N |

  - CANtraffic_sim,

TRAFFIC_SIM

| Dir | Cycle Count | MOb | ID | Len | Data | Stamp |
|---|---|---|---|---|---|---|
| Rx | 6000 | 0 | 123 | 4 | 11 22 33 44 | 0x204 |
| Tx | 12152 | 0 | 124 | 4 | 11 22 33 44 | 0x3cd |
| Rx | 14000 | 0 | 7FE | 8 | F0 F1 F2 F3 F4 F5 F6 F7 | 0x66c |
| Tx | 22622 | 0 | 7FF | 8 | F0 F1 F2 F3 F4 F5 F6 F7 | 0x8ea |

  - & CANmailbox_sim.

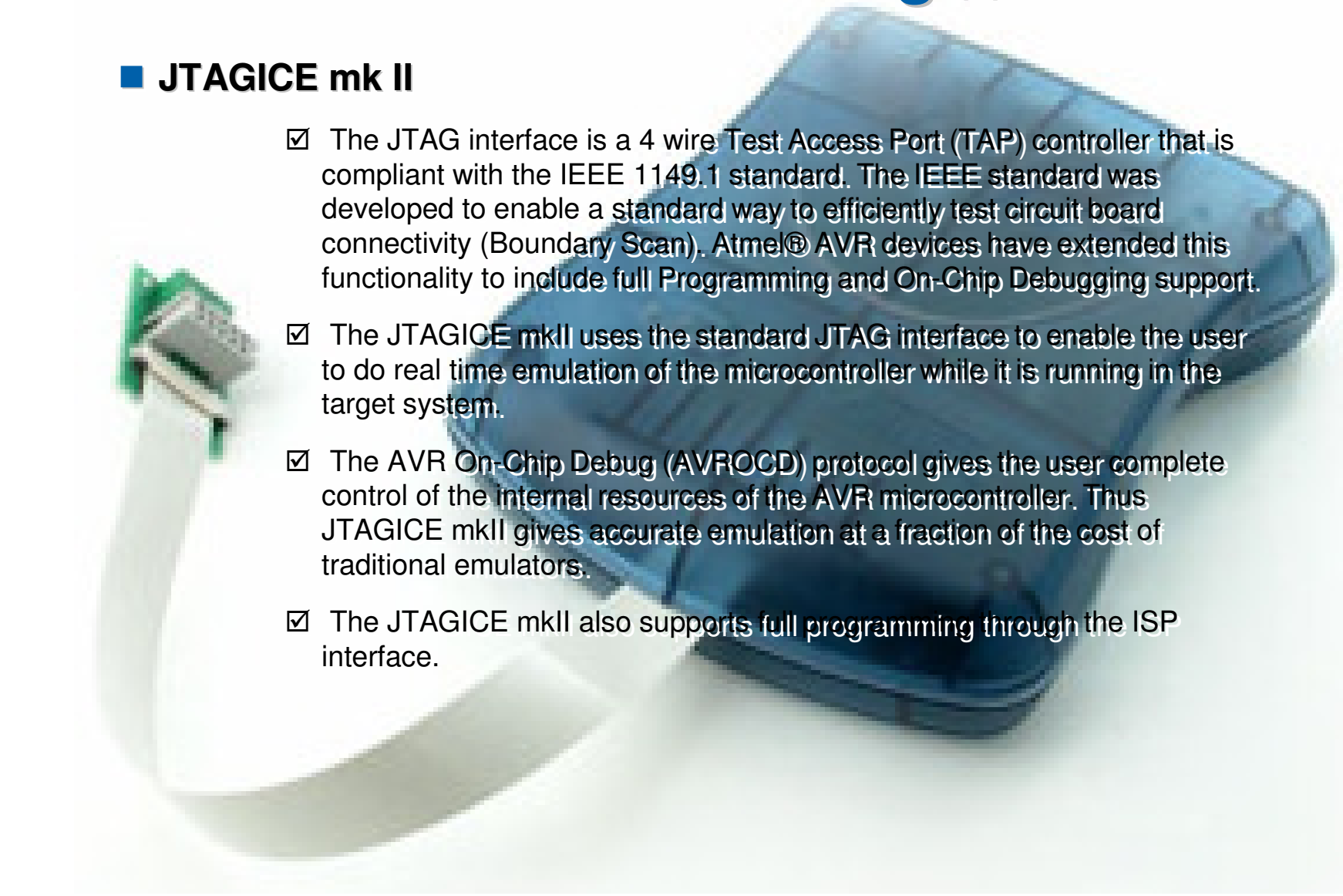
MAILBOX_SIM

| MOb | Mode | Status | ID Tag | ID Mask | Len | Data | Stamp |
|---|---|---|---|---|---|---|---|
| 0* | Tx | 23% processed | 7FF | 0 | 8 | F0 F1 F2 F3 F4 F5 F6 F7 | 066C |
| 1 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 2 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 3 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 4 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 5 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 6 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 7 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 8 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 9 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 10 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 11 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 12 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 13 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |
| 14 | Dis | Done | 0 | 0 | 0 | 00 00 00 00 00 00 00 00 | 0000 |

# Hardware Debug (1)

## JTAGICE mk II

- ☑ The JTAG interface is a 4 wire Test Access Port (TAP) controller that is compliant with the IEEE 1149.1 standard. The IEEE standard was developed to enable a standard way to efficiently test circuit board connectivity (Boundary Scan). Atmel® AVR devices have extended this functionality to include full Programming and On-Chip Debugging support.
- ☑ The JTAGICE mkII uses the standard JTAG interface to enable the user to do real time emulation of the microcontroller while it is running in the target system.
- ☑ The AVR On-Chip Debug (AVROCD) protocol gives the user complete control of the internal resources of the AVR microcontroller. Thus JTAGICE mkII gives accurate emulation at a fraction of the cost of traditional emulators.
- ☑ The JTAGICE mkII also supports full programming through the ISP interface.

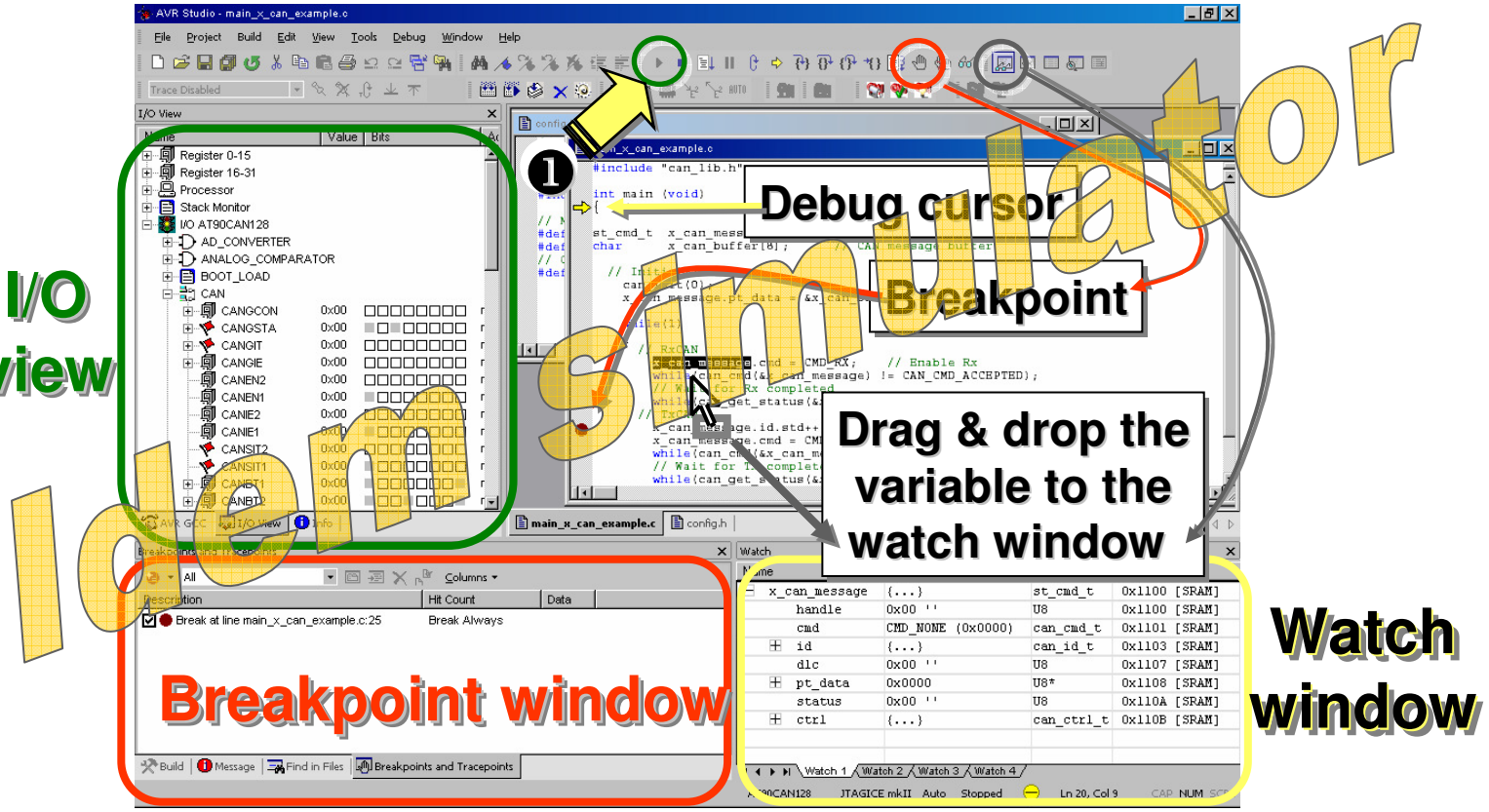


# Hardware Debug (2)

- DVK90CAN1 & JTAGICE mk II
  - Connections

# Hardware Debug (3)

- **Debugging with JTAGICE mk II (start)**
  - **Watch window, Code breakpoint & I/O view**



# Hardware Debug (4)

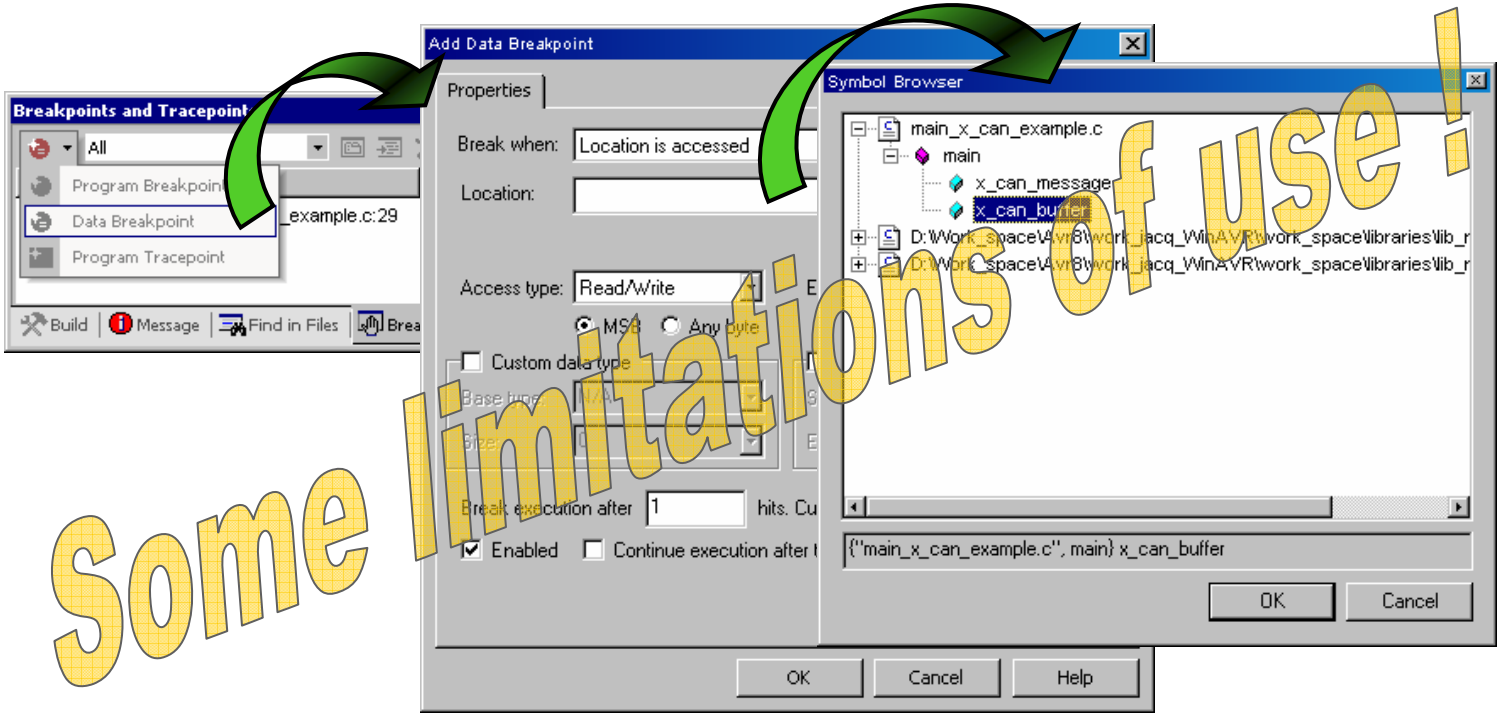
- **Debug tools**

Start Debugging
Stop Debugging
Run (F5)
Break (CTRL-F5)
Reset (SHIFT+F5)
Show next statement
Single step, Trace Into (F11)
Step Over (F10)
Step Out (SHIFT+F11)
Run To Cursor (F7)
Auto Step
Quick watch

# Hardware Debug (5)

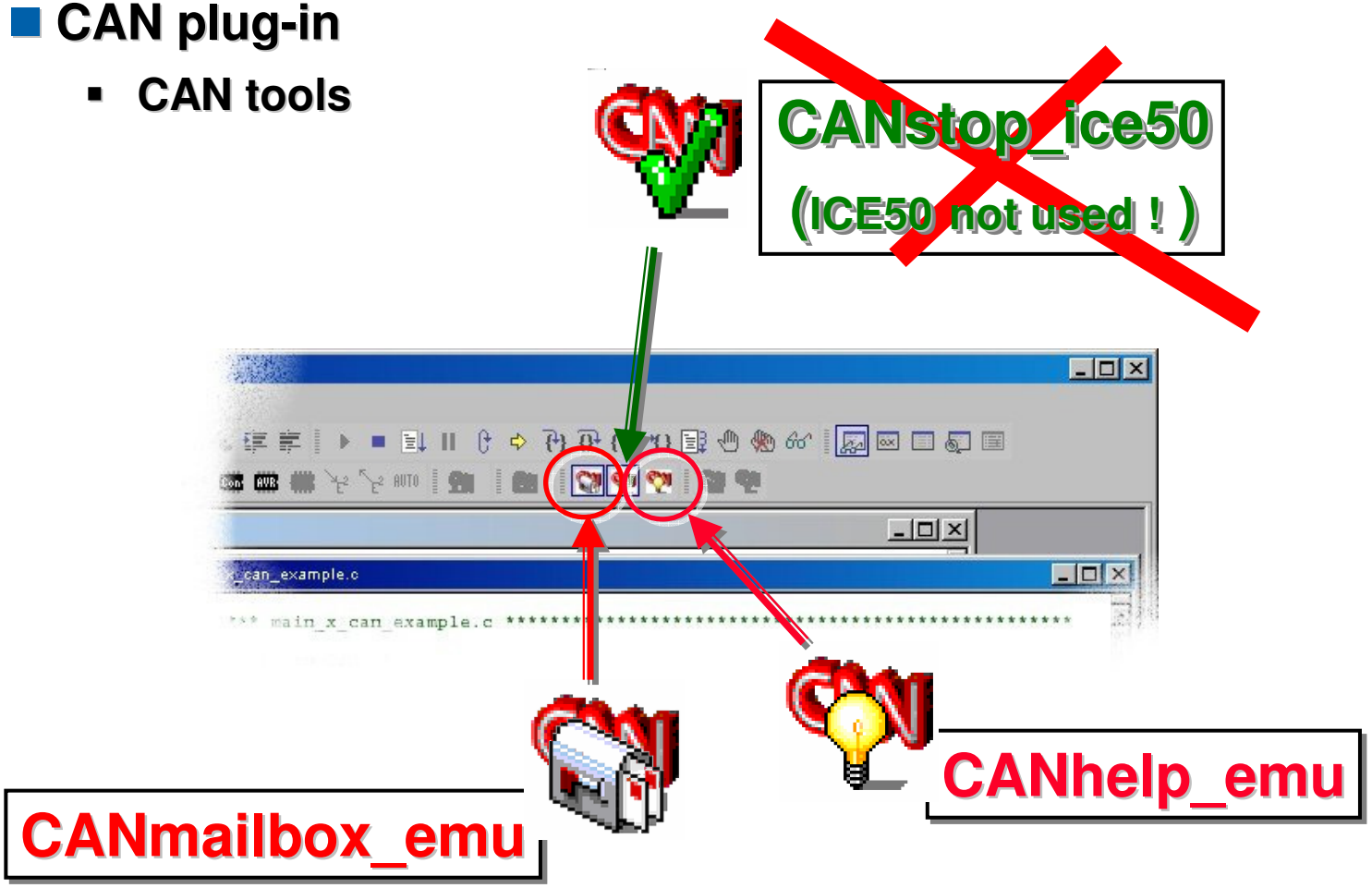
- **Breakpoints**
  - Code breakpoint

    Toggle program breakpoint (F9)  Clear all program breakpoints
  - Data breakpoint



# Hardware Debug (6)

- CAN plug-in
  - CAN tools



# Hardware Debug (7)

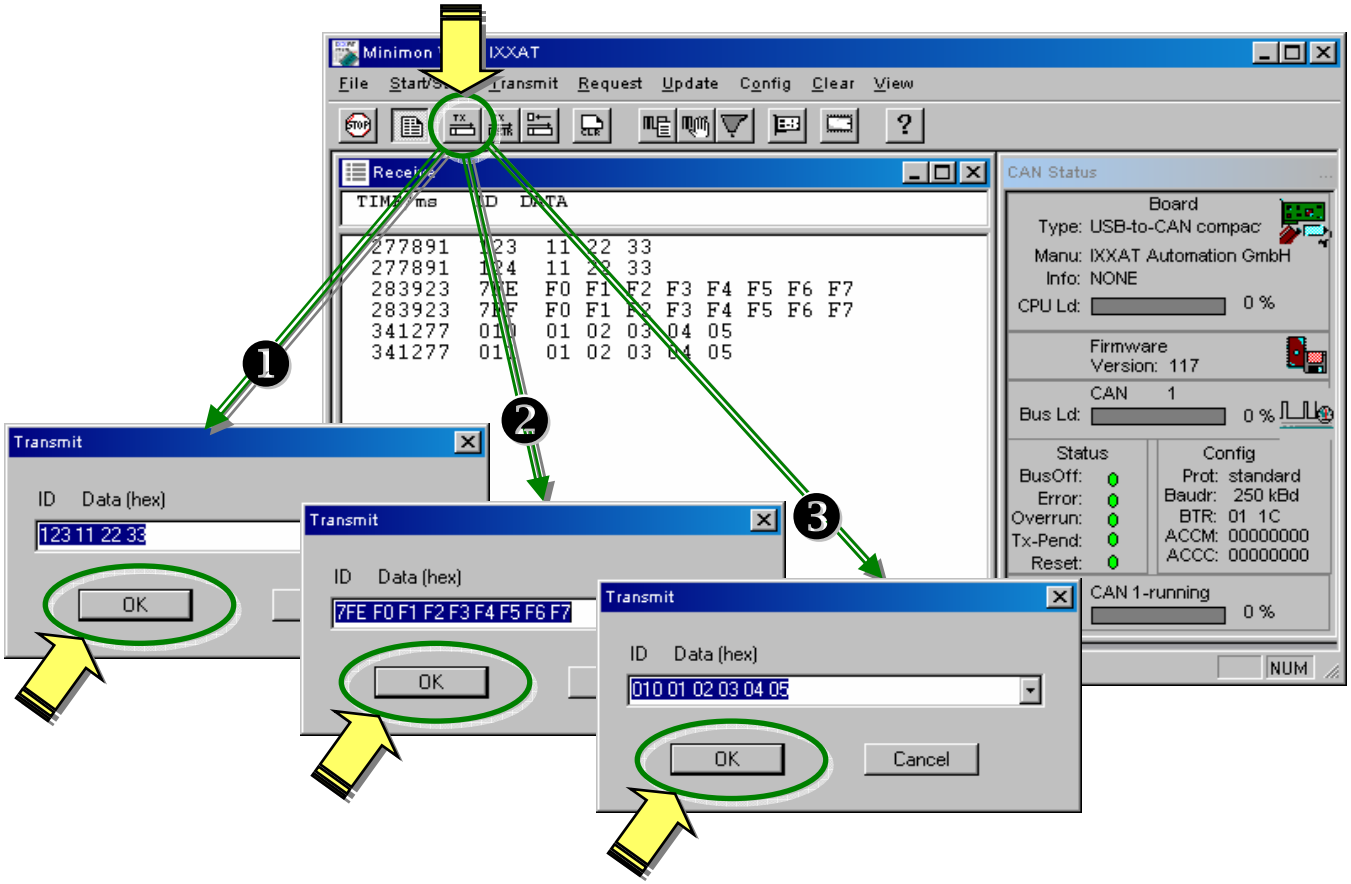
- **CAN hw. debugging**
  - & CANmailbox_emu.

CANICE

| MOb | Mode | ID Tag | . | Len | Data | Stamp |
|---|---|---|---|---|---|---|
| 0 * | Tx | 124 | 0 | 4 | 11 22 33 44 C3 00 00 00 | 4CCB |
| 1 | Dis | 0 | 0 | 0 | F7 BF 9C F8 BB 7B F7 2A | 9D25 |
| 2 | Dis | 0 | 0 | 0 | 42 D0 2C FA B7 64 C5 03 | 91BD |
| 3 | Dis | 0 | 0 | 0 | 4F 37 BF F1 0E D2 92 FA | 5719 |
| 4 | Dis | 0 | 0 | 0 | E7 E9 3E 4E 68 56 B9 D2 | E2CF |
| 5 | Dis | 0 | 0 | 0 | 29 EF 00 67 EB DC 7F F6 | 9225 |
| 6 | Dis | 0 | 0 | 0 | DD 09 58 2A F4 9D 31 E3 | 36E4 |
| 7 | Dis | 0 | 0 | 0 | 00 00 47 06 67 10 F2 10 | 0561 |
| 8 | Dis | 0 | 0 | 0 | 60 59 5E AA F5 E2 FC 0D | E733 |
| 9 | Dis | 0 | 0 | 0 | 30 7E 12 B9 C3 FC D7 32 | 890B |
| 10 | Dis | 0 | 0 | 0 | 1F DF E7 8E C6 79 0D 13 | 2207 |
| 11 | Dis | 0 | 0 | 0 | 32 26 E9 38 54 2A FB 74 | 282F |
| 12 | Dis | 0 | 0 | 0 | 2F CB 37 E7 DB D6 FD C0 | 7BF8 |
| 13 | Dis | 0 | 0 | 0 | FE 06 78 11 A4 AB F6 3A | 4BC2 |
| 14 | Dis | 0 | 0 | 0 | B2 BC 35 2D AD B0 41 23 | 4D53 |

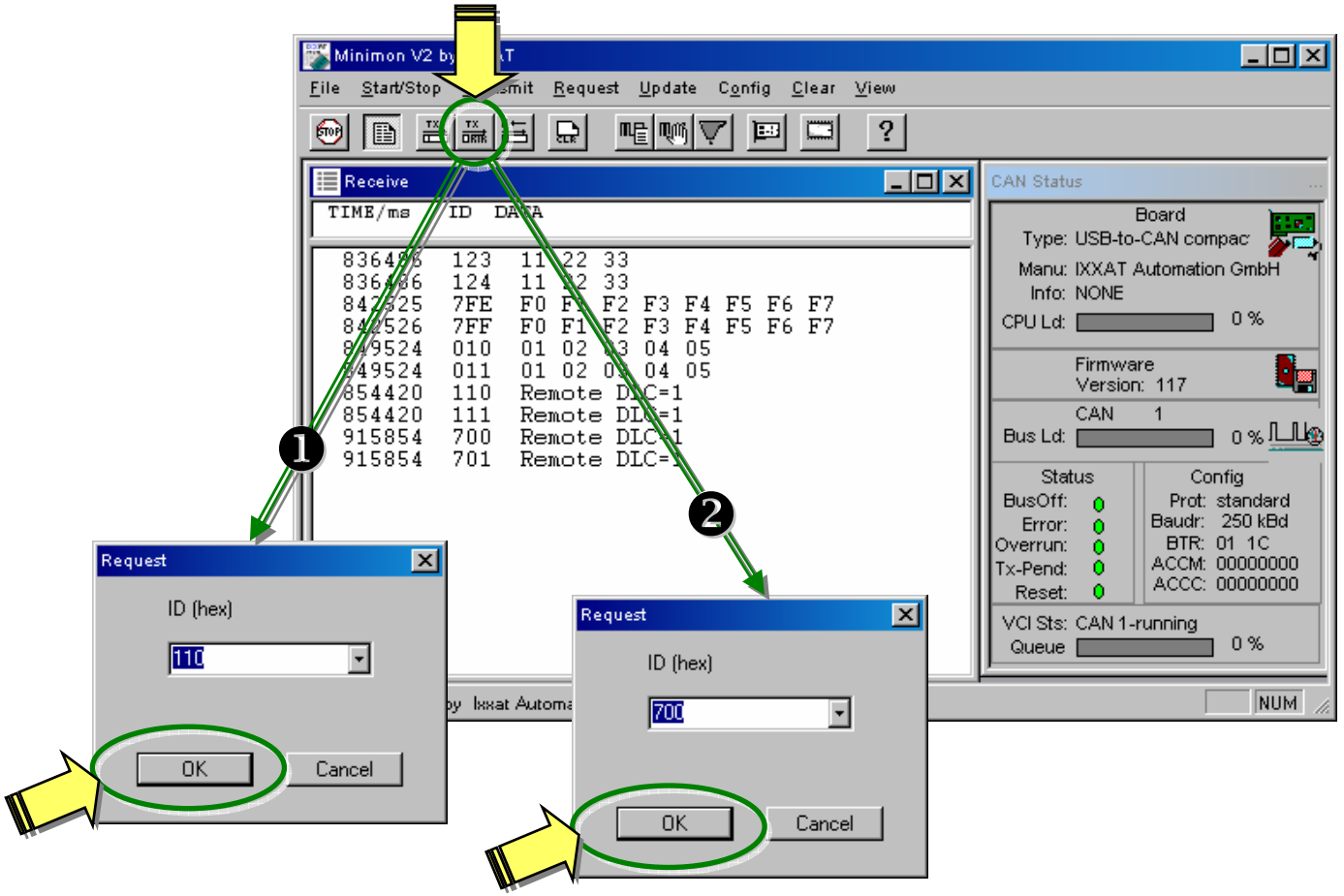
# Hardware Debug (8)

- **Application with Minimon V2**
  - **CAN Data Frames Generation**



# Hardware Debug (9)

- **CAN Remote Frames Generation**



# Conclusion

Simulator ≡ Hw. Debug

… or almost !

Develop an
Atmel AVR CAN project is:
easy, fast & efficient.

# Atmel contact

**Corporate Headquarters**
2325 Orchard Parkway
San Jose, CA 95131,
USA
TEL: (1)(408) 441-0311
FAX: (1)(408) 487-2600

**Asia**
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
TEL: (852) 2721-9778
FAX: (852) 2722-1369

**Product Contact**
La Chantrerie
BP 70602
44306 Nantes Cedex 3
France
TEL: (33) 2 40 18 18 18
FAX: (33) 2 40 18 19 60

# Atmel customer support

**E-mail**
avr@atmel.com

# Atmel products

**Web site**
www.atmel.com