Canary - an AVR CAN driver

Christoph Steup

EOS - IVS at FIN - OvGU

March 15, 2010

Sections

Motivation

- AVR is family of micro controllers, therefore:
  - small amount of RAM available
  - small amount of program memory available
  - computational power is limited
  - needs to fullfill realtime requirements in certain enviroments
- to cope with these requirements one need:
  - configurable architecture
  - highly efficient code
  - as much decisions at compile time as possible

Introduction

- Canary - CAN driver provides the fallowing features:
  - adaptive configuration based on C++ template metalanguage
  - small memory footprint
  - small code size
  - full usage of hardware acceleration features of the at90canX[1] series of micro controllers
- Alternative driver CanNoInt, provides:
  - highly deterministic behavior by not using interrupts
  - even smaller footprint than Canary
  - also smaller code size than Canary

Design and Components

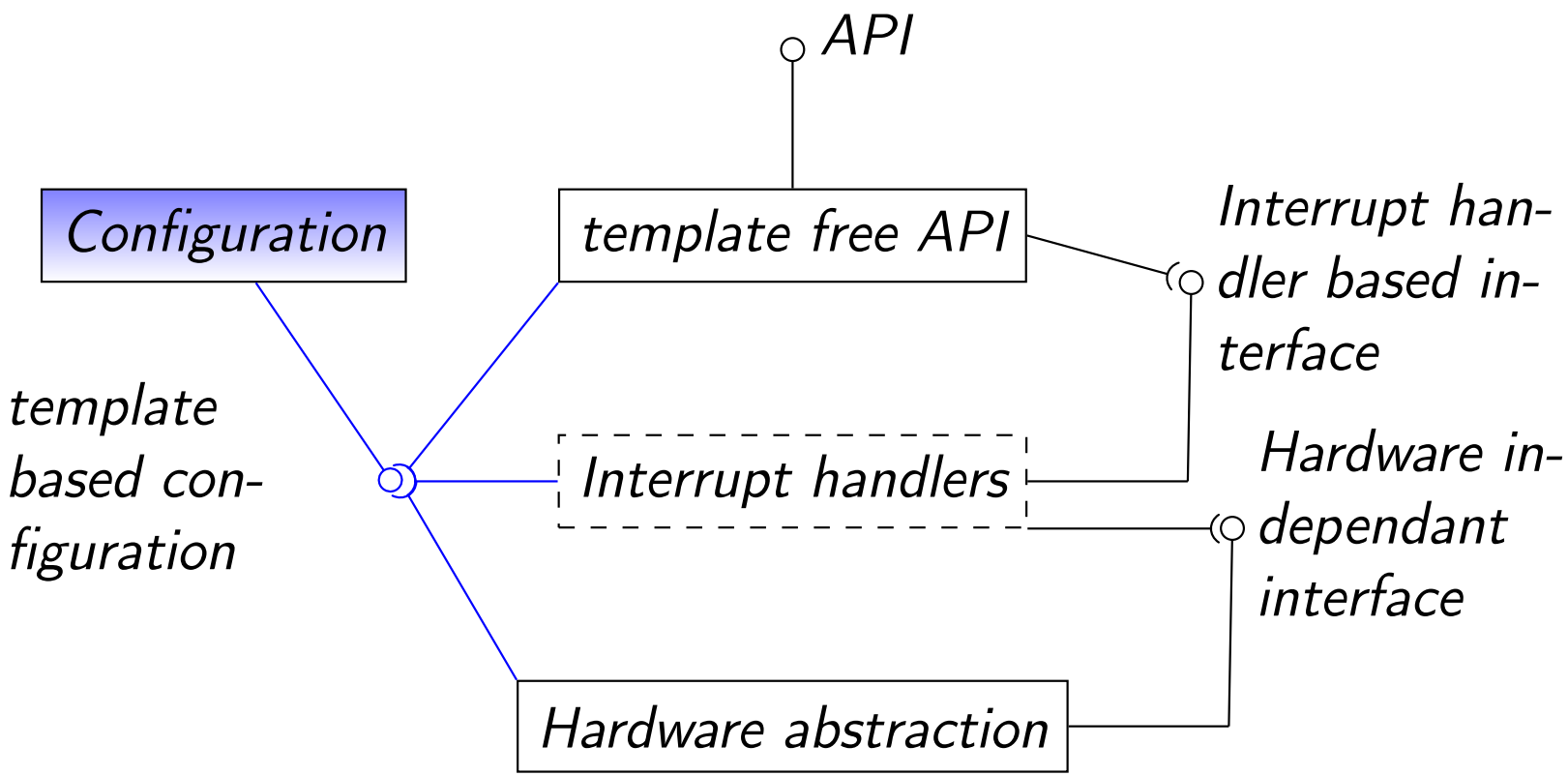

Figure: Diagram of the components of the driver and the interfaces between them.

Basic concepts

- Extension of the avr-halib[2], therefore using similar concepts:
  - register maps
  - delegate style interrupt handling
- some concepts, are taken from FAMOUSO[3]:
  - compile time configuration via templates
  - using template specialization to achive optimal code
- new concept: configurable regmaps, to cope with different register content in CAN 2.0A and CAN 2.0B

Example configuration

```
using namespace avr_halib::canary;

struct CANConfig : defaultCANConfig{
    typedef BaudRateConfig<F_CPU, SPEED_1M, SUBBITS_16>
        baudRate;

    enum Parameters{
        version              = CAN_20B,
        maxConcurrentMsgs    = 4,
        useError             = false,
        useReceive           = false
    };
};

typedef Canary<CANConfig> Can;
typedef Can::MsgSend      CanSendMsg;
```

Listing 1: An example configuration of the Canary driver class

Demo network

- small scale CAN Network - 3 Nodes:
  - Sender: sends 1 message, 2 RTRs
  - Receiver: Receives message
  - RTR-Receiver: Respond to RTRs
- laptop with pcan usb dongle to view CAN traffic
- whole demo interrupt driven
- combining CAN 2.0A and CAN 2.0B
- using receive and reply, as well as auto reply

Evaluation and measurements

application	program size	ram usage
Sending with interrupts	1842	42
Sending without interrupts	1528	42
Receiving with interrupts, no output	2130	41
Receiving with interrupts, LCD output	3602	42
Receiving without interrupts, no output	1614	29
Receiving without interrupts, LCD output	3084	30

Table: Program sizes and used amount of RAM for different example applications

References


at90can documentation is available at http://www.atmel.com/dyn/resources/prod_documents/doc4250.pdf.


the avr-halib is available at https://ivs-pm.ovgu.de/projects/halib.


M. Schulze.
Famouso – eine adaptierbare publish/subscribe middleware für ressourcenbeschränkte systeme.
Electronic Communications of the EASST, 17: Kommunikation in Verteilten Systemen, 2009.