19589 - PS Telematik-Projekt: Wireless Embedded Systems
First Steps

Bastian Blywis
Department of Mathematics and Computer Science
Institute of Computer Science
15. April, 2009
1. Embedded Systems

2. MSP430f1612

3. ScatterWeb²

4. Hardware

5. First Steps

6. Tasks
- Slow CPU
- Small RAM and (Flash) ROM Memory
- Often times no MMU
- Limited energy resources
- FPU might be missing
- Specialized OS
- Difficult to debug
- Small operation system for the MSB-430 . . .

- . . . but more or less a collection of source code

- Event driven

- No (preemptive) multitasking

- Microcontroller in low power mode most of the time (99%)

- User space applications use kernel functions via libscatterweb

- Documentation needs work
- Hardware: Core module MSB-430
- Extension: MSB-430S
- OS kernel with drivers for core module . . .
- . . . and extension drivers (optional)
- Applications using kernel API (that’s what you will develop)
- Applications using hardware directly (rare)
Interrupt Handling

Event based system

1. System initially in low power mode
2. HW interrupt, e.g. voltage applied to I/O pin
3. CPU interrupts current program (if running)
4. registered interrupt service routine (ISR) in IVT gets called
5. ISR sets flag (depending on interrupt source)
6. ISR wakes up CPU
7. superloop detects set flag
8. registered event handler gets called → event handled
   - in kernel and/or
   - in user application
9. CPU reenters low power mode again
- Limited
- Via JTAG interface (but doesn’t work this well with MSPGCC)
- Write messages to serial out
- Configure MSB-430 via terminal commands

Example, logging:

```c
    printf( /* some text */ , /* some arguments */ );
    LOG_LOW( "PONG from %u", args->netheader->from );
```

Example, terminal command:

```c
    COMMAND(RST, 0, cmdargs) {
        System_reset(WDS_SWRESET);
    }
```
Terminal commands can be invoked

- locally
- remote

Example, local:

$ id
[id] 57

Example, remote:

$ @123 id
[123] 123
Hardware

Core Module Connector
Ground
Power via USB
Switch
JTAG Connector
Power via Battery
Serial Connector

MSB-430T base module

FTDI USB-to-Serial cable
JTAG Interface (USB)

MSB-430
JTAG Interface (parallel)
Stacked MSB Modules
1. Open Cygwin Bash Shell

2. Download Source (do not use a windows network folder!!!)

   $ cd SOME_DIR
   $ mkdir SOME_NEW_DIR
   $ svn co https://svn.mi.fu-berlin.de/scatterweb/...
   ...MSB-430/lab-branches/SS09/lab{1,2,3,4,5,6,7,8,9}

3. Make copy of [EMPTY] application

   $ cd lab{1,2,3,4,5,6,7,8,9}/Applications
   $ svn cp [EMPTY] first_steps

4. Compile application

   $ cd first_steps
   $ make

5. Flash to MSB-430

   $ make flash
1. Open a terminal emulator (e.g. TerraTerm, HyperTerminal, hterm)
   - 115200 Baud
   - Data Bits = 8
   - Parity = none
   - Stop Bits = 1
   - Flow Control = None
   - Send line ends with line feed (\r\n or cr-lf)
   - Local Echo = on

2. Reset device (use the hardware switch)

3. Toggle red LED
   $ led 1
   $ led 0
1. Open src/ScatterWeb.Process.c

2. Write a new terminal commando
   
   ```c
   COMMAND(HelloMSB, CMDFLAG_SERIAL, cmdargs) {
       printf("Hello User");
   }
   ```

3. Implement a call counter
   
   ```c
   uint8_t counter = 0;

   COMMAND(HelloMSB, CMDFLAG_SERIAL, cmdargs) {
       printf("Hello User - called \%u times\r\n", counter++);
   }
   ```
- Parse a parameter
  ```c
  uint8_t counter = 0;
  ```
  ```c
  COMMAND(HelloMSB, CMDFLAG_SERIAL, cmdargs) {
      uint16_t n = String_parseUint16(cmdarg->args, NULL);
      while(n--) {
          printf("Hello User - called %u times", counter++);
      }
  }
  ```
- You can use every function in the stdlib...
- ...unfortunately MSPGCC provides only a reduced stdlib
1. Call a function once 5 s after system start
   
   ```c
   void callMe() { /* some action */}
   
   void Process_init() {
      System_registerCallback( C_RADIO, (fp_vp)Process_radioHandler );
      System_registerCallback( C_SENSOR, (fp_vp)Process_sensorHandler );
      Timers_add(5*1024, callMe, 0xFFFF);
   }
   ```

2. Call a function every 5 s
   
   ```c
   void callMe() {
      print("foo bar");
      Timers_add(5*1024, callMe, 0xFFFF); // 1024 ticks == 1 second
   }
   ```

3. View timer table with command `tmr`

4. Toggle the led in your periodic function
1. Create header

```c
netpacket_send_args_t npsargs;
Net_sendArgsInit(&npsargs);
```

2. Set necessary values

```c
npsargs.netheader.to = ???; // choose a destination id
npsargs.netheader.type = USERDEFINED_PACKET+???; // choose a number
npsargs.netheader.flags = 0;
```

Note: USERDEFINED_PACKET is defined in
Libraries/libscatterweb/include/ScatterWeb.Net.PacketTypes.h
Add your lab-number to the value to get a unique type id.

3. Add some data

```c
uint8_t* data = "Hello world!";
npsargs.payload[0].buffer = (uint8_t*) data;
npsargs.payload[0].size = strlen(data)+1; // include terminating char
```

4. Send frame/packet

```c
Net_send(&npsargs);
```
1. Goto Process_radioHandler()

2. Include case for your type

```c
switch ( args->netheader->type ) {
    case PING_REQUEST_PACKET:
        Net_sendPong(args->netheader->from);
        break;
    case PING_REPLY_PACKET:
        LOG_LOW( "PONG from %u", args->netheader->from );
        break;
    case USERDEFINED_PACKET+???:
        // do something
        break;
    default:
        return false;
}
```

3. Test the sending and receiving of data with your two nodes
Traffic Generator
- Write a command that can initiate and terminate the periodic sending of data.
- The interval shall be configurable as parameter.
- Limit the number of sent packets by another parameter.

Example: `genPackets start 0.5 1000`
(send a packet every 0.5 seconds and stop after 1000 packets)

Dump Network Traffic
- Write a command that switches on/off the dumping of received packets.
- Decode the header fields in your output.
- Enable promiscous mode; receive all packets and not only multicasts and unicast destined for your node

Example: `dump start`

From: 123
   To:  12
   Id:  3
   ...

From: 123
   To:  12
   Id:  3
   ...

...
End of the First Steps