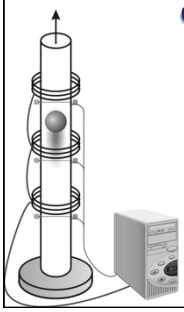
Operating Systems for Embedded Computing

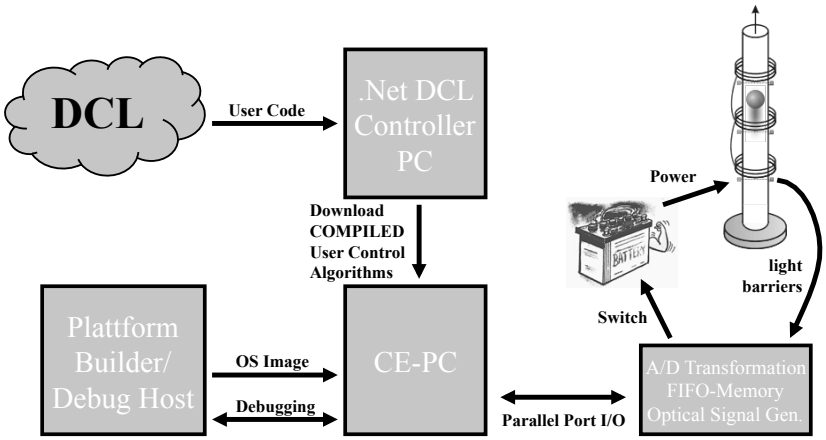
# The Higher Striker Experiment

## „Higher Striker“ – „Hau den Lukas“

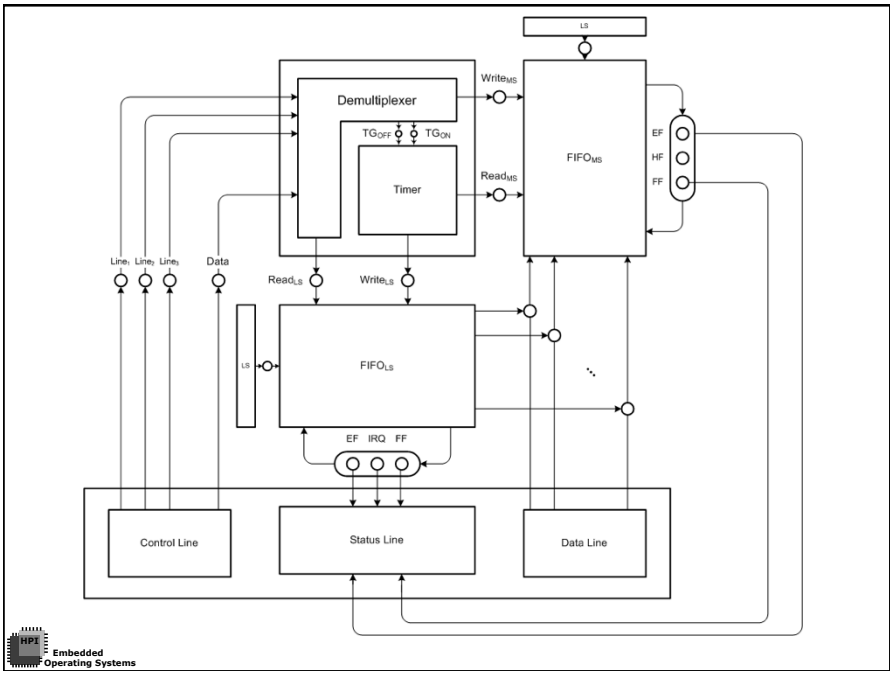
- Similar hardware to pendulum experiment
  - Parallel I/O / 38 kHz sample rate / 256 Byte buffer
- Use of Real Time OS
  - Smaller Buffers, Higher Sampling Frequency
  - Short control delay
  - COTS x86 PC
    - Intel Celeron 633 MHz, 128 MB RAM (max 64 MB usable)
    - 10 Mbit/s LAN (NE 2000 PCI)
      - Combination of non-RT .Net and RT application
      - CE-PC Windows Ce.Net 4.2



# "Higher Striker" - Architecture




HPI Embedded Operating Systems




# Controlling the Experiment

- Abstract from Direct Hardware Programming
- Provide Higher Level API to the experiment
- 2 possible control scenarios :
  - Calculation before runtime, analysis of result after experiment execution for next run
  - Calculation of control signals during runtime – hard deadlines
- Investigation of various operating systems
- .Net Compact Framework and real time

Embedded
Operating Systems

# „Higher Striker“ - Event List - API

```
0;1;
10;0;
20;2;
25;0;
50;4;
80;0;
```

→

```
1111111111000000000
0222220000000000000
0000000000000000888
8888888888888888888
8888880000000000000
```

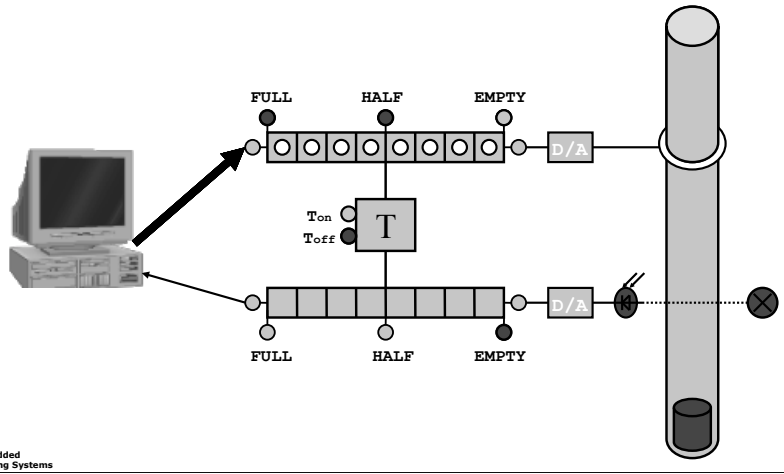
- Definition before runtime
- Transformation into byte stream before runtime
- Simple checks possible (temperature of magnet)
- Generation of event list after runtime for analysis for next experiment runs
- Not flexible – but simple to implement



# Minimal Program (Pseudo Code)

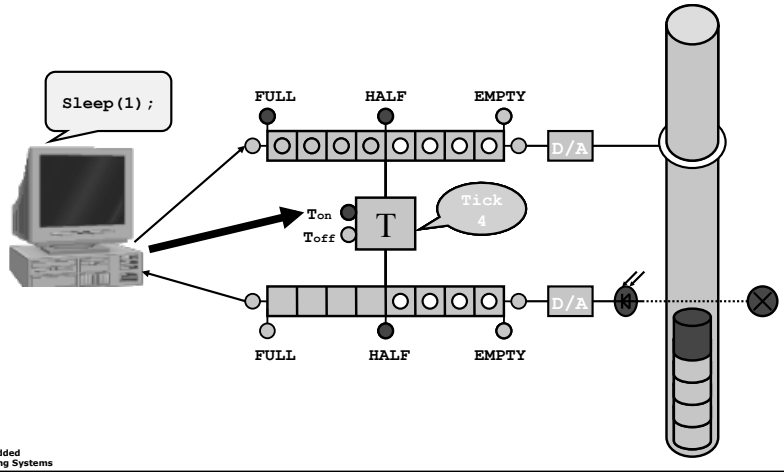
```
INITIALIZE();
do
{
  READ();
  WRITE(buffer);
  GETSTATUS();
  if(EMPTY_FLAG_LS) Sleep(1);
}
while(!EOF(buffer))
```

# Phase 1 - Init



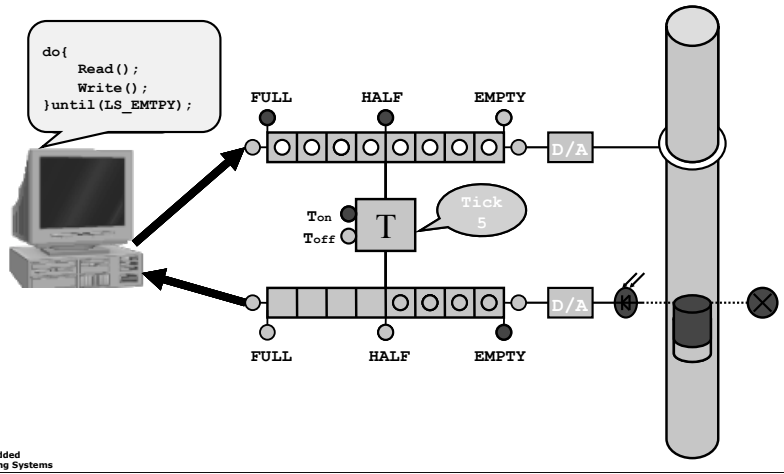

Embedded Operating Systems

# Phase 2 – Running
# Using full buffers - Sleeping



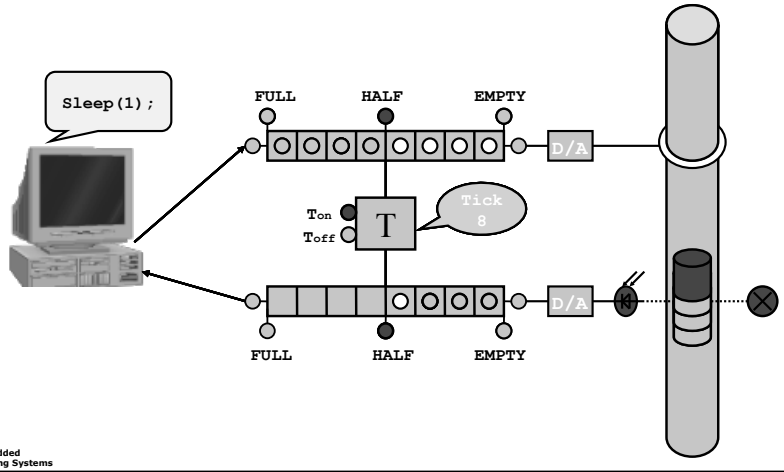

Embedded Operating Systems

# Phase 2 – Running
# Using full buffers – I/O



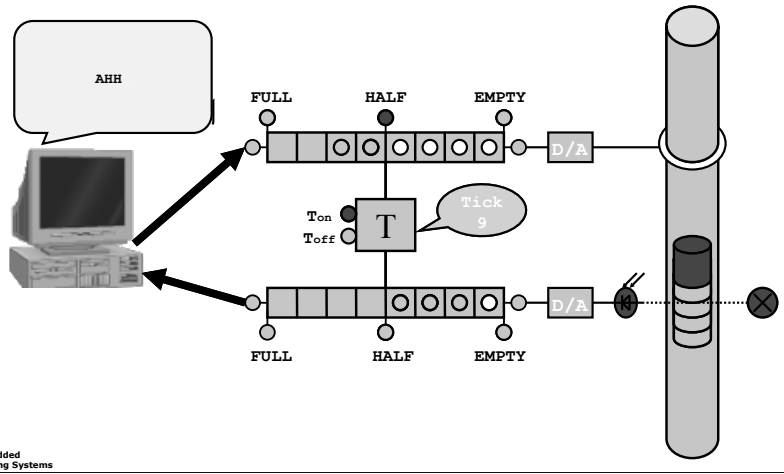

Embedded Operating Systems

# Phase 2 – Running
# Using full buffers – Event




Embedded Operating Systems

# Phase 2 – Running
# Using full buffers – React on Event




Embedded Operating Systems

# Problem

- Time to react on event : buffer length + 1
- 6,7 ms + time for calculation: too long to enable magnet on time
- Cylinder could fly about 4 cm

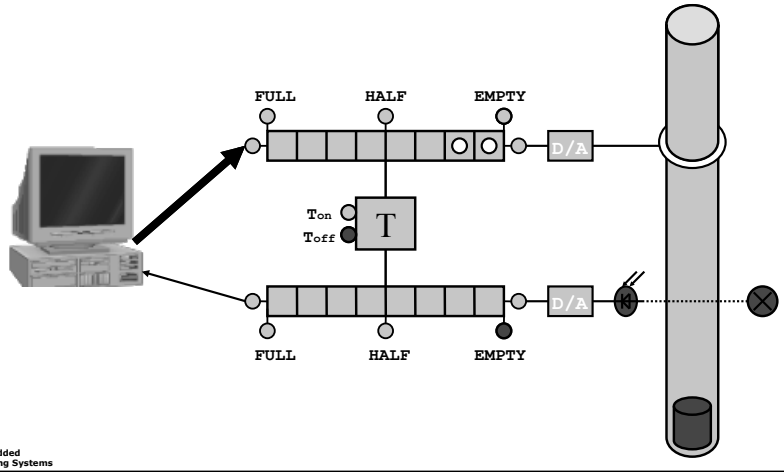
Embedded Operating Systems

# Solution 1

- Decrease buffer write ahead
  - Initial size of outgoing buffer
- Deadline much harder
- Sleep as short as possible
- Atomic Read / Write necessary because only Empty Flag of LS-FIFO usable
- Reaction time : (write_ahead +1) / frequency
- ! Reaction time > minimal Sleep

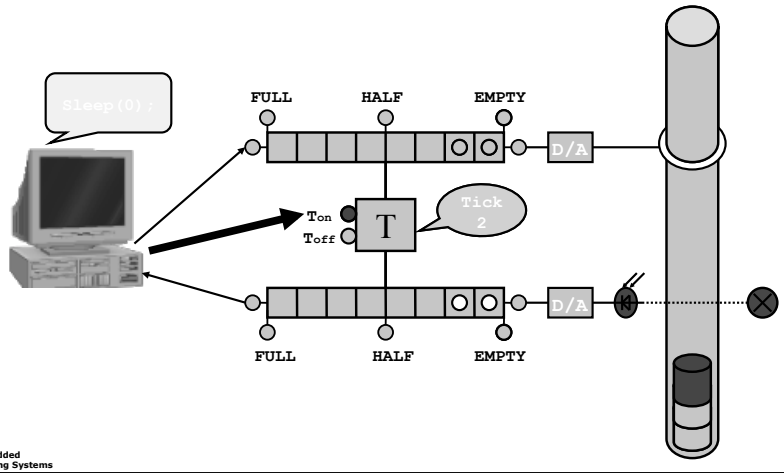
Embedded Operating Systems

# Phase 1 – Init (small write ahead)



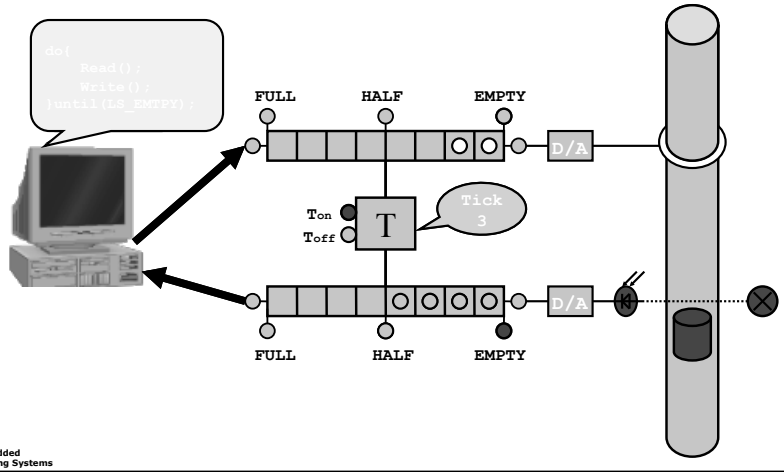

Embedded Operating Systems

# Phase 2 – Running
# Using small buffers - Sleeping




Embedded Operating Systems

# Phase 2 – Running
# Using small buffers – I/O




Embedded Operating Systems

# Why do we have to sleep ?

- Several parallel tasks have to be performed
  - In context of DCL : monitoring / observation / replacement of user control components
  - Recording of experiment runtime data
  - Update of system clock
  - Networking support to upload new user control components
  - Logging
  - Kitl Debugging Connection
  - Watch dog signalling



# Simple I/O – Accessing the Hardware

# The parallel port interface

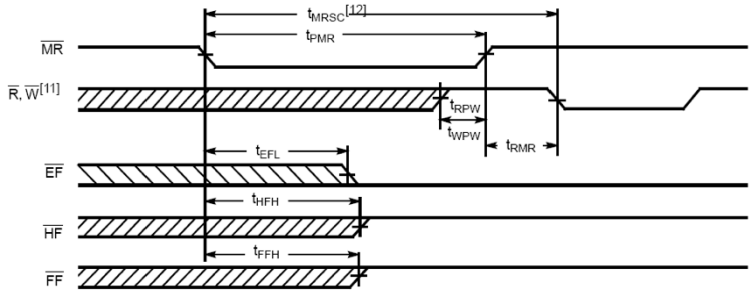

- 25-Pin Connector connects experiment hardware
- Experiment Setup:
  - DATA register 0x03BC
  - STATUS register 0x03BD
  - CONTROL register 0x03BE

```
inline void OUTP( short port, unsigned char value )
{
        _asm mov dx, port
        _asm mov al, value
        _asm out dx, al
}
```

Embedded Operating Systems

# Accessing the Hardware Reset

Master Reset



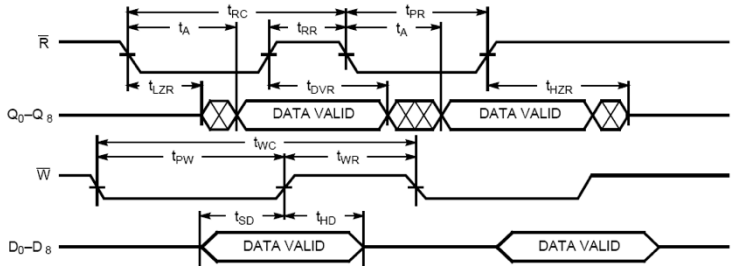

```
Sleep(1);
OUTP(CONTROL, 0x02 | 1);
Sleep(1);
OUTP(CONTROL, 0x02);
```

```
#define RESET 0x02
#define READ_LS 0x0C
#define WRITE_MS 0x6
#define TIMER_STOP 0x00
#define TIMER_START 0x0E
```

Embedded Operating Systems

# Accessing the Hardware Reading/ Writing Data

Asynchronous Read and Write




```
OUTP( CONTROL, RECEIVING | DISABLE_IRQ | READ_LS | 1 );
rvalue = INP( DATA );
OUTP( CONTROL, RECEIVING | DISABLE_IRQ | READ_LS | 0 );
```

Embedded Operating Systems

# Initializing the Hardware

```
// Stop Timer
    SendCommand(STOP_TIMER);
// Reset FIFO
    SendCommand(RESET);
// Pre-fill WriteBuffer ( Magnet control )
    for(i=0; i<255; i++)
    {
        // do nothing
        WRITEMS(0x00);
    }
// Finally start the Timer
    SendCommand(START_TIMER);
```

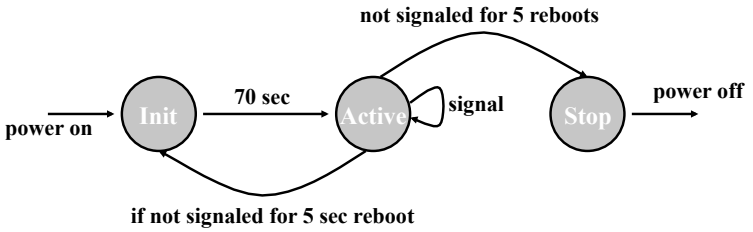
Embedded Operating Systems

# Higher Striker Control API

```
int HDLInitialize ( );
int HDLInitialize ( unsigned char Writeahead,
                    unsigned char Initial[]);

int HDLStartClock( );
int HDLStopClock( );

int HDLPerform( unsigned char * Write,
                unsigned char * Read );

int HDLGetStatus();
int HDLGetError( int * LSError, int * MSError );
```

Embedded Operating Systems

# Higher Striker – Watch Dog




- Hardware Watchdog connected / signaled via serial communication interface of control PC
- In case of system hang-up control PC will be rebooted
- Atmel AVR 8-Bit Tiny 12 microcontroller

Embedded Operating Systems

# Solution 2 – using interrupts

- Half Full Flag triggers interrupt
- Interrupt handler empties buffers
- Problem : at least half full buffers must be used

Embedded Operating Systems

# Solution 3
# Real-Time Linux and Periodic Threads

- rtLinux can schedule Threads up to 40 kHz periodically / low jitter
- Buffers are read/written each period
- Used write-ahead buffer :

| Iterationen / Periode | Busy waiting | 13 µs | 26 µs | 260 µs |
|---|---|---|---|---|
| 100000 ~ 2s | 1 | 1 | 41 | 48 |
| 1000000 ~ 26s | 1 | 40 | 50 | 59 |
| Andere Prozesse | Beinahe nicht aktiv | Sehr zäh | langsam | Fast normal |

Embedded Operating Systems

J.Gressmann, B. Kaufmann 2004

## Some Phyiscs

Distance between 2 magnets = 18 cm
Distance between 2 ligth barriers = 18 cm
Light barrier in the middle of 2 magnets
Height of cylinder = 5 cm
Cycling frequency = 38,4 kHz
Weight of cylinder = 4,8g :-)

## Programming in C

- Include string #usec# in a comment of your code
- Programm against Windows Ce.Net 4.2 operating system
- Use C / C++
- Debug output: The file result.txt will be made available as result in the Web interface
- Constants are provided in hio.h
  - Parallel interface ports numbers
  - Light barrier / magnet coding
  - Demultiplexer codings

HPI Embedded Operating Systems

# How to Start

- Read Lab Assignment 1
- Start with programming basic I/O
  - Start with the evaluation of the status register
  - Write 0 into magnet buffer
  - Check error flags
  - Change initial buffer size
- Evaluate light barriers
  - Output speed at each light barrier
  - Calculate on-/off-time for next magnet

Embedded Operating Systems