Introduction to NXTOSEK

goo.gl/MQLbb

Tizar Rizano

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Introduction

- OSEK is an open standard for embedded system architecture
- NXTOSEK is NXT device drivers + Kernel + OS
- NXTOSEK provides:
  - C and C++ programming environment with GCC tool chain
  - C and C++ API for NXT sensors,actuator and other devices

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Installation

- Open http://lejos-osek.sourceforge.net
- Follow the installation steps for Windows or Linux
- In the end you should have:
  - Installed the compiler: GNU ARM
  - Updated the NXT Firmware to John Hansen's Enhanced NXT firmware
  - Downloaded and setup NXTOsek

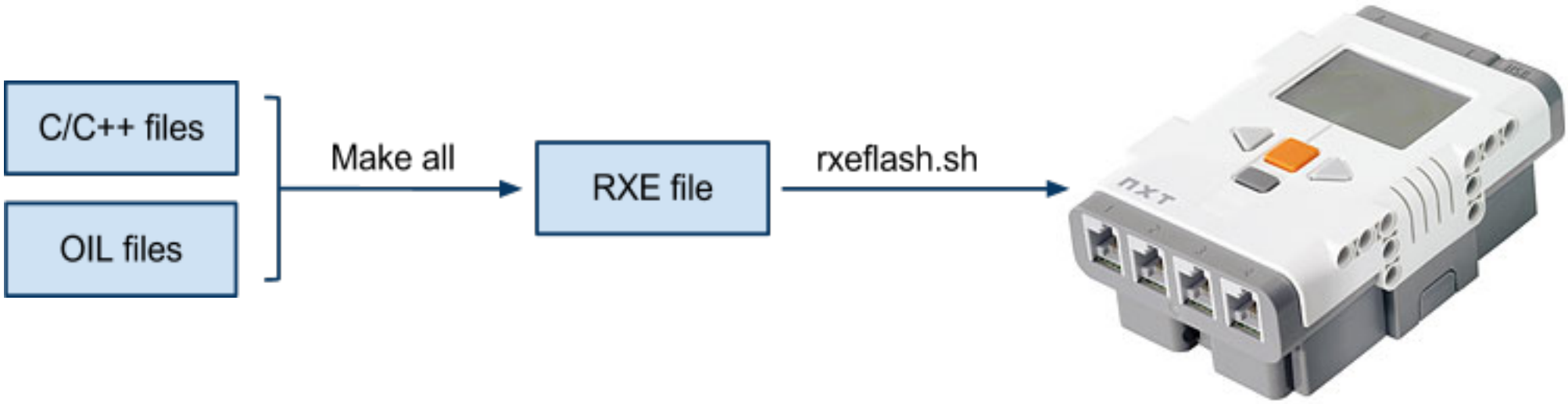
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

NXTOSEK Program

- An NXTOSEK program consists of two parts:
  - an OIL (OSEK Implementation Language) file describing the architectures
    - scheduler
    - tasks : priority, activation, autostart
    - etc
  - C/C++ source codes

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

NXTOSEK Program



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Example 1 : Hello World

OIL file

```
#include "implementation.oil"
CPU ATMEL_AT91SAM7S256 {
  OS LEJOS_OSEK {
    STATUS = EXTENDED;
    STARTUPHOOK = FALSE;
    ERRORHOOK = FALSE;
    SHUTDOWNHOOK = FALSE;
    PRETASKHOOK = FALSE;
    POSTTASKHOOK = FALSE;
    USEGETSERVICEID = FALSE;
    USEPARAMETERACCESS = FALSE;
    USERESSCHEDULER = FALSE;
  };
  APPMODE appmode1{};
  /* Definition of TASK */
  TASK task1 {
    AUTOSTART = TRUE {
      APPMODE = appmode1;
    };
    PRIORITY = 1;
    ACTIVATION = 1;
    SCHEDULE = FULL;
    STACKSIZE = 512;
  };
};
```

C source

```
/* helloworld.c */
#include "kernel.h"
#include "ecrobot_interface.h"

void user_1ms_isr_type2(void){}
TASK(task1){
  while(1){
    ecrobot_status_monitor("Hello, World!");
    systick_wait_ms(500); /* 500msec wait */
  }
}
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Example 1 : Hello World

OIL file

```
#include "implementation.oil"
CPU ATMEL_AT91SAM7S256 {
  OS LEJOS_OSEK {
    STATUS = EXTENDED;
    STARTUPHOOK = FALSE;
    ERRORHOOK = FALSE;
    SHUTDOWNHOOK = FALSE;
    PRETASKHOOK = FALSE;
    POSTTASKHOOK = FALSE;
    USEGETSERVICEID = FALSE;
    USEPARAMETERACCESS = FALSE;
    USERESSCHEDULER = FALSE;
  };
  APPMODE appmode1{};
  /* Definition of TASK */
  TASK task1 {
    AUTOSTART = TRUE {
      APPMODE = appmode1;
    };
    PRIORITY = 1;
    ACTIVATION = 1;
    SCHEDULE = FULL;
    STACKSIZE = 512;
  };
};
```

Enable waiting state e.g. stop the task to wait for a certain event

Hook routines flag. Enable/disable user defined function in the OS

Enable/disable resource scheduler

Lower number means lower priority

Only a single activation is permitted

This task is preemptable

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Example 2 : Sensors and Display

OIL file

```
#include "implementation.oil"
CPU ATMEL_AT91SAM7S256 {
 OS LEJOS_OSEK {
  STATUS = EXTENDED;
  STARTUPHOOK = FALSE;
  ERRORHOOK = FALSE;
  SHUTDOWNHOOK = FALSE;
  PRETASKHOOK = FALSE;
  POSTTASKHOOK = FALSE;
  USEGETSERVICEID = FALSE;
  USEPARAMETERACCESS = FALSE;
  USERESSCHEDULER = FALSE;
 };
 APPMODE appmode1{};
 TASK task1 {
  AUTOSTART = TRUE {
   APPMODE = appmode1;
  };
  PRIORITY = 1;
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
 };
};
```

C source

```
#include "kernel.h"
#include "ecrobot_interface.h"
#define PORT NXT_PORT_S3
void user_1ms_isr_type2(){}
void ecrobot_device_initialize() {
   ecrobot_set_light_sensor_active(PORT);
}
void ecrobot_device_terminate(void) {
 ecrobot_set_light_sensor_inactive(PORT);
}
TASK(task1) {
 while(1) {
  if (ecrobot_is_ENTER_button_pressed()){
   display_clear(0);
   display_goto_xy(0,0);
   display_int(ecrobot_get_light_sensor(PORT),0);
   display_update();
  }
 }
}
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Example 3 : Periodic Task

OIL file

```
#include "implementation.oil"
CPU ATMEL_AT91SAM7S256
{
 OS LEJOS_OSEK {
  /*...*/
 };
 APPMODE appmode1{};
 TASK task1 {
   AUTOSTART = FALSE;
   PRIORITY = 1;
   ACTIVATION = 1;
   SCHEDULE = FULL;
   STACKSIZE = 512;
 };
 COUNTER SysTimerCnt {
   MINCYCLE = 1;
   MAXALLOWEDVALUE = 10000;
   TICKSPERBASE = 1;
 };
 ALARM cyclic_alarm1 {
   COUNTER = SysTimerCnt;
   ACTION = ACTIVATETASK {TASK = task1;};
   AUTOSTART = TRUE {
     ALARMTIME = 1000; /* Offset */
     CYCLETIME = 200;  /* Period */
     APPMODE = appmode1;
   };
 };
};
```

C source

```
/* periodic.c */
#include "kernel.h"
#include "ecrobot_interface.h"
DeclareTask(task1);
DeclareCounter(SysTimerCnt);
int ctr=0;
void user_1ms_isr_type2(void)
{
  StatusType ercd;
  ercd = SignalCounter(SysTimerCnt);
  if (ercd != E_OK){ShutdownOS(ercd);}
}

TASK(task1)
{
    display_clear(0);
    display_goto_xy(0,0);
    display_int(ctr, 3);
    ctr++;
    display_update();
    TerminateTask();
}
```