

Software Setup for developing realtime application with LEGO mindstorms NXT

Fumitake FUJII* , Dr. Eng.

- * Mechanical Systems Control Laboratory, Graduate School of Science and Engineering, Yamaguchi University

Before you begin

- Please make sure that your environment satisfies the following requirements!
 - You have a notebook PC in which Microsoft Windows is installed.
 - Supported windows version : XP / Vista / 7 / 8
 - You have the broadband internet connection (Some of the software to be installed is several hundreds Mbytes in its size.)
 - At least one USB 2.0 port available.
- If you are prepared, let's start!
 - Open the following internet web site.
<http://lejos-osek.sourceforge.net/index.htm>

Entering nxtOSEK website

- Click “[installation](#)” link on the left of the screen

nxtOSEK/JSP

ANSI C/C++ with OSEK/ μ ITRON RTOS for **LEGO MINDSTORMS NXT**

English / [Japanese](#) / [Simplified Chinese](#)

Click here!

[What is nxtOSEK?](#)
[Downloads](#)
January 2013, nxtOSEK 2.18
[Installation](#)
[How to upload a program](#)



- You’ll then encounter the following screen. Click “[Installation in Windows](#)” link and proceed.

nxtOSEK Installations

Click here!

[Installation in Windows \(XP/Vista/7\)](#)

[Installation in Linux](#)

[Installation in Mac OS X Lion](#)

Follow the instructions given in the page

- What you have to do is simply follow the installation procedure from 1 to 4 **one-by-one** as designated on the webpage.

nxtOSEK Installation in Windows XP/Vista/7

Setup of nxtOSEK application development environment requires several 3rd party software and most of tools is CUI (Command User Interface) based software. So if you are not familiar with those kind of software, it may make you frustrated, but please be patient to follow the below instructions.

[1. Install Cygwin](#)

[2. Install GNU ARM](#)

[3. Setup nxtOSEK program upload software](#)

[4. Set up nxtOSEK](#)

This message is of particular importance!

- Several cautions should be given to you on the 1st (cygwin installation) and the 3rd (program uploader setup) steps.

Tips in Installing cygwin

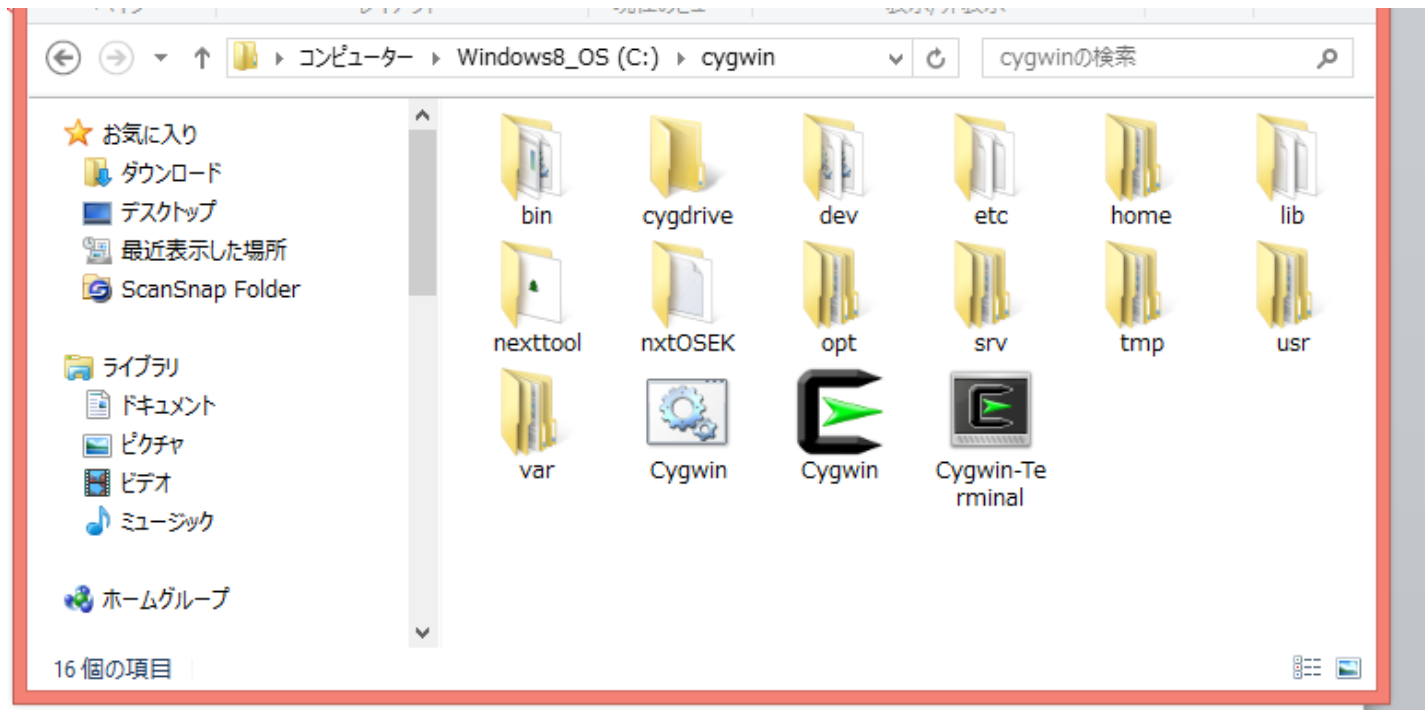
- There might be several steps which require your thoughts and decision while installing cygwin.
 - Read the following Q's and A's carefully before you proceed. Steps which are not mentioned below, just follow the web instruction.
1. Which version of cygwin binaries should I choose ?
 - You will be prompted to select 32bit / 64bit binaries of cygwin to install your PC. **Choose 32 bit version** even if your PC has 64bit OS installed.
 2. Where do I setup the binaries ?
 - I recommend you to install cygwin in the default location (c:\cygwin). We can't support you during class if you have altered this installation location.
 - Never use folder names which contain white space and/or multibyte characters.
 3. Utility versions are different from the screenshots !
 - You might encounter newer binaries during the installation of cygwin (as compared to the screenshots provided in the webpage.)
 - **Never mind!** Install the latest available for download.
 4. What should I do at the end of installation ?
 - Please don't forget to create shortcut of cygwin shell on the desktop.

Choice of uploader and installing nxtOSEK

- In step 3, you have to choose one program uploader from the table.
 - Out legos are equipped with John Hansen's Enhanced NXT firmware.
 - You should then follow the instructions described under "[Set up nxtOSEK program upload software for the Enhanced NXT firmware](#)" and install the following 2 utilities. **You don't need to download Enhanced Firmware.**
 - LEGO MINDSTORMS USB FANTOM Driver
 - John Hansen's NeXTTool. Extract the archive to c:¥cygwin¥nexttool. (if cygwin is installed in the default location)
- The final step is to install "nxtOSEK".
 - First, go back to the top page of nxtOSEK website (displayed in page 3 of this handout) and click "[Downloads](#)" link.
 - Click "nxtOSEK 2.18" link on the top of the page and choose to download "nxtOSEK-v218.zip" from the source-forge web site.
 - Save it to an appropriate location of your hard drive and extract it.
 - Copy nxtOSEK folder to c:¥cygwin.

If your installations are going properly...

- Your “c:¥cygwin” folder now should look like this.



- Check the existence of **nexttool** and **nxtOSEK** folders here.
- If they can't be seen here, you did something wrong during the procedure.

Choice of uploader and installing nxtOSEK

- In addition to 4-step installation of nxtOSEK, we have to set up [sg.exe](#) utility.
 - Show “nxtOSEK installation” page as partly shown in the bottom of p.3 of this handout by clicking “back” button several times in your web browser.

Note: Since nxtOSEK v2.12, [sg.exe](#) binary file (an OSEK OIL parser/code generator) has been removed from nxtOSEK. For more detailed information, please check [Downloads page](#).

Then, click this link.

- Then you see the following screen. Click the link on the bottom of the box.

Note: [sg.exe](#) is removed from nxtOSEK (March 01, 2010)

To keep the terms of Sourceforge.net, I have decided to remove [sg.exe](#) binary file (an OSEK OIL parser and code generator) from nxtOSEK. [sg.exe](#) which was included in nxtOSEK is same as [sg.exe](#) in TOPPERS/OSEK 1.1 at [TOPPERS project](#) where outside of Sourceforge.net. TOPPERS/OSEK 1.1 can be downloaded from :

[osek_os-1.1.lzh](#)

[osek_os-1.1.lzh](#) is compressed as LZH format and it needs to use a lzh supported file extractor, for example, [7-zip](#) which would work in Windows XP/Vista and Linux/Unix.

To use [sg.exe](#) in [osek_os-1.1.lzh](#) for nxtOSEK:

- Extract [osek-os-1.1.lzh](#)
- Copy extracted [/toppers_osek/sg/sg.exe](#) to [nxtOSEK/toppers_osek/sg](#) directory

Click here !

Note that nxtOSEK ver2.18 supports OSEK COM features and it requires [sg.exe](#) in the latest [TOPPERS ATK1](#) (January, 2013)

Grabbing sg.exe

- Then you will see the screen containing the image below. Click the link “ATK1 Release 1.0” in the table.

TOPPERS/ATK1

TOPPERS/ATK1の最新リリースの配信キットは、以下からダウンロードできます。この配信キットにTOPPERS/ATK1の全ソースファイルが含まれています。ATK1の詳細は「[Automotive Kernel](#)」をご覧ください。

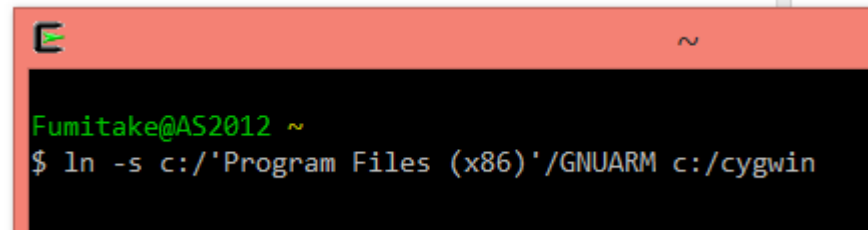
最新リリース

| リリース名 | タイプ | サイズ | リリース日 |
|----------------------------------|-----------------|-------|------------|
| ATK1 Release 1.0 | lzh(SJIS,CR/LF) | 3.7MB | 2008-11-17 |

- Save the file anywhere you like and extract it.
- Then a new folder “atk1-1.0” will appear. Change current folder by following the path `./atk1-1.0/toppers_atk1/sg` and copy sg.exe there to `c:/cygwin/nxtOSEK/toppers_osek/sg` directory.
- You should then go back to nxtOSEK installation and edit the file `c:/cygwin/nxtOSEK/ecrobot/tool_gcc.mak` as instructed in the webpage to suit to your installation.

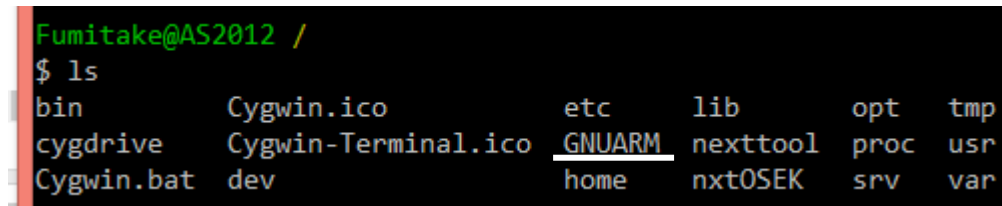
Setup the software link.

- Execute cygwin shell by double-clicking the shortcut icon.
- When cygwin shell window is raised, type the following command in the captured image as it is with your keyboard (This is called **CUI** operation. CUI is an acronym of “**C**haracter **U**ser **I**nterface.”)



```
Fumitake@AS2012 ~  
$ ln -s c:/'Program Files (x86)'/GNUARM c:/cygwin
```

- Then type `cd c:/cygwin` and press enter. After that, type `ls` and press enter.
- If you can see a GNUARM in the list, the `ln -s` command has executed successfully.



```
Fumitake@AS2012 /  
$ ls  
bin          Cygwin.ico          etc          lib          opt          tmp  
cygdrive     Cygwin-Terminal.ico GNUARM     nexttool    proc         usr  
Cygwin.bat   dev                 home        nxtOSEK     srv          var
```

Check software installation.

- If it is all OK, let's try to make an nxtOSEK application here.
- Type in the following command on the cygwin bash shell.
 - `cd /cygdrive/c/cygwin/nxtOSEK/samples_c/helloworld`
 - `make all`
- A number of echo backs will be displayed, but don't worry. You have to troubleshoot something **only when the procedure stops with "Error" in the display.**
- The `make` utility compiles all source files, link necessary libraries and generate program **which can only be executed in LEGO.** If "make all" finishes successfully, you can see following files generated under "helloworld" directory.

```
Fumitake@AS2012 /cygdrive/c/cygwin/nxtOSEK/samples_c/helloworld
$ ls
appflash.sh      helloworld_OSEK_rxe      helloworld_OSEK_rom.elf  kernel_cfg.c
biosflash.sh    helloworld_OSEK_ram.bin  helloworld_OSEK_rom.map  kernel_id.h
build           helloworld_OSEK_ram.elf  helloworld_OSEK_rxe.elf  Makefile
helloworld.c    helloworld_OSEK_ram.map  helloworld_OSEK_rxe.map  ramboot.sh
helloworld.oil  helloworld_OSEK_rom.bin  implementation.oil        rxeflash.sh
```

- This kind of software development environment is called **"cross development."**

You can use eclipse to build nxtOSEK applications.

- nxtOSEK application development can be done with **I**ntegrated **D**evelopment **E**nvironment (**IDE** for short) software called **Eclipse**, if you have successfully installed necessary software as instructed in the previous pages.
- We have successfully tested **the latest version of Eclipse** (**version 4.3: Kepler**) and **Eclipse IDE for C/C++ Developers**.
- SPIED 2013 attendants are strongly encouraged to setup Eclipse and Eclipse C/C++ IDE in their computer.
- Please visit the homepage <http://www.eclipse.org/downloads/> to get the latest version. Both are available from this download page.

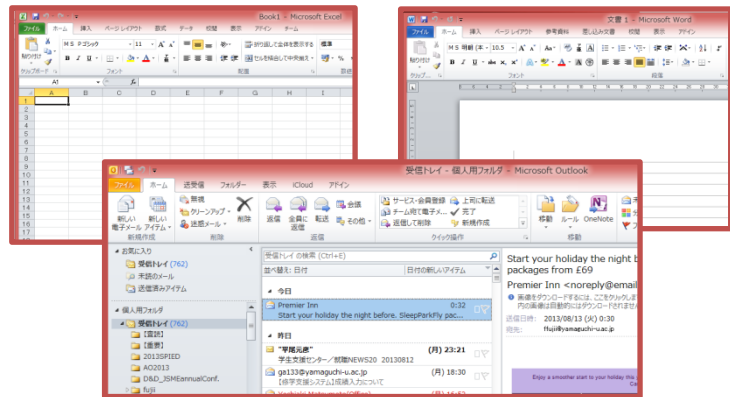
Realtime Programming for Control of LEGO NXT Using nxtOSEK

Fumitake FUJII* , Dr. Eng.

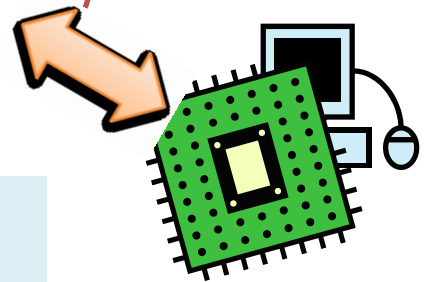
- * Mechanical Systems Control Laboratory, Graduate School of Science and Engineering, Yamaguchi University, Japan

What is RTOS ? (1) – Multitasking

- Multitask OS



High level Software execution



CPU



Low level I/O

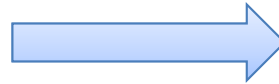
request

True / False Question
 Single core CPU can do multiple jobs **simultaneously**.

What is RTOS ? (2) – Preemptive Multitasking

True / False Question

Single Core CPU can do multiple jobs **simultaneously**.



The answer is ...

- Multitasking in reality

- Multitasking is implemented by **time sharing**

- 1) dividing CPU process time to pieces **by OS** in a **specified manner**.

- 2) Assigning CPU time to applications which are

- waiting in cue and

- have the highest priority

- if CPU is released by some other application** (CPU time is controlled **by Application**)

- Then assigning available CPU time to processes waiting in cue.

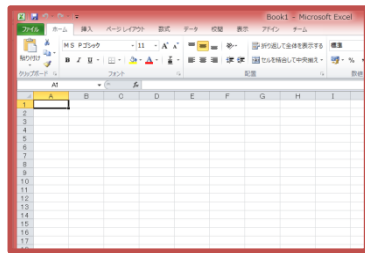
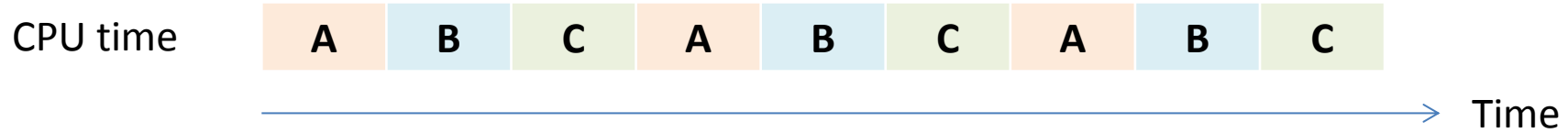
- In this way, a PC is seen as doing multiple jobs simultaneously.

- Strategy 1) --- **Preemptive Multitasking**

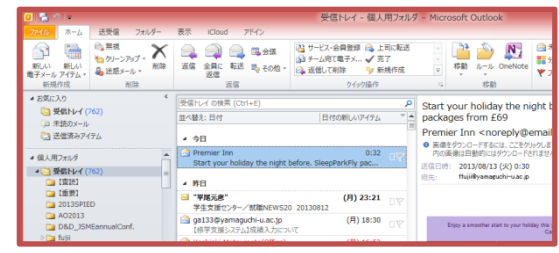
- Strategy 2) --- Non-preemptive Multitasking

Preemptive multitasking

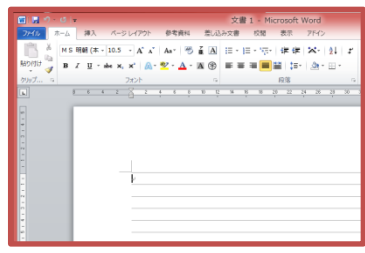
This is an example of round-robin type scheduling. Execution order is dependent on task priority.



task A



task B



task C

Note:

Assigned CPU time to multiple tasks may vary according to the status of tasks. For example, an application will be given less CPU time if it is in idle status and/or it does not require long CPU time.

Enrich your IT vocabulary

- Thread

- A minimum unit of execution of a computer code to which CPU time is given in multitasking / parallel processing.
- Important relation

A thread < An application (in size and required resources)

- Multi-threading

- A time sharing system where thread is used as a unit of execution.
- A single application can be divided into multiple threads.

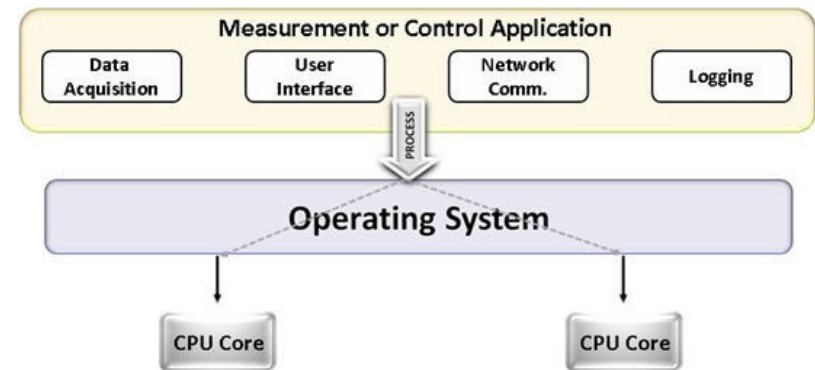
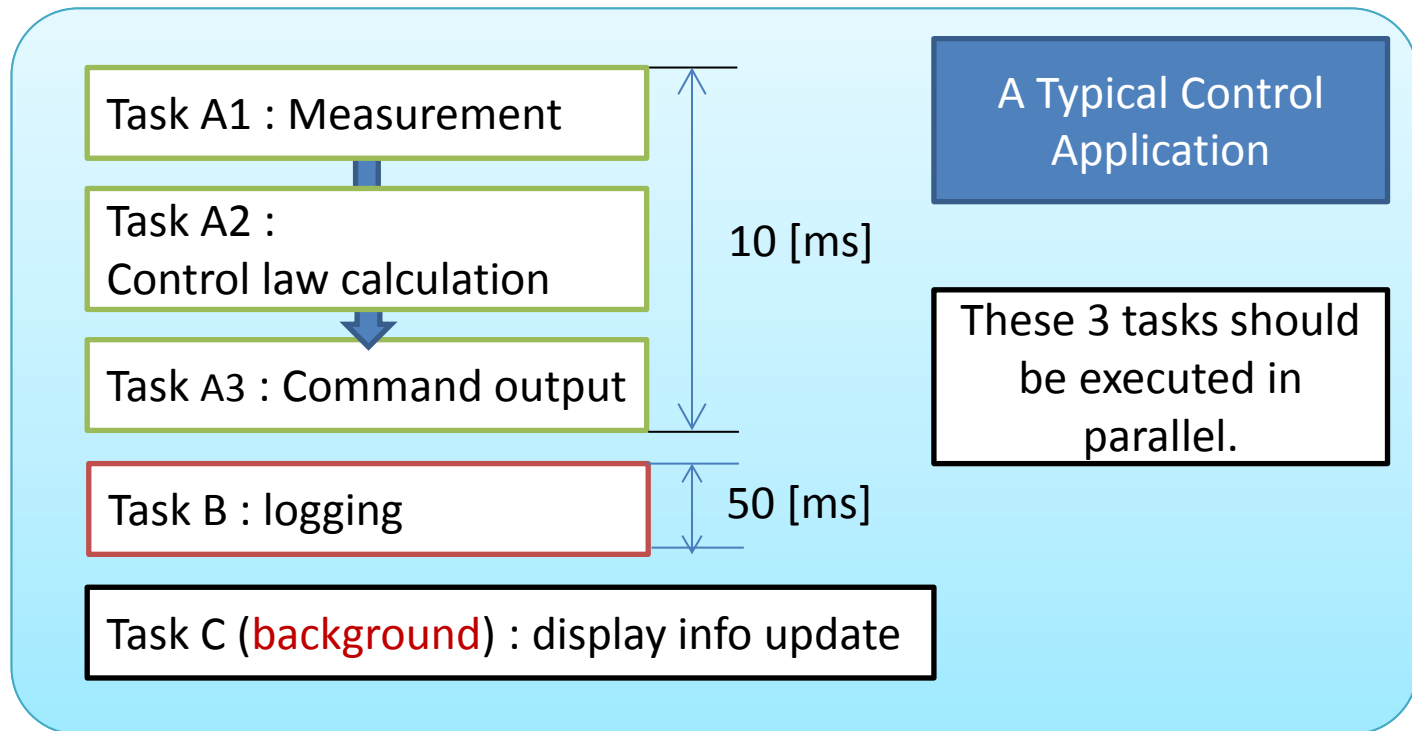


Fig. Multi-threading (courtesy of ti.com)

Finally, what is RTOS ?

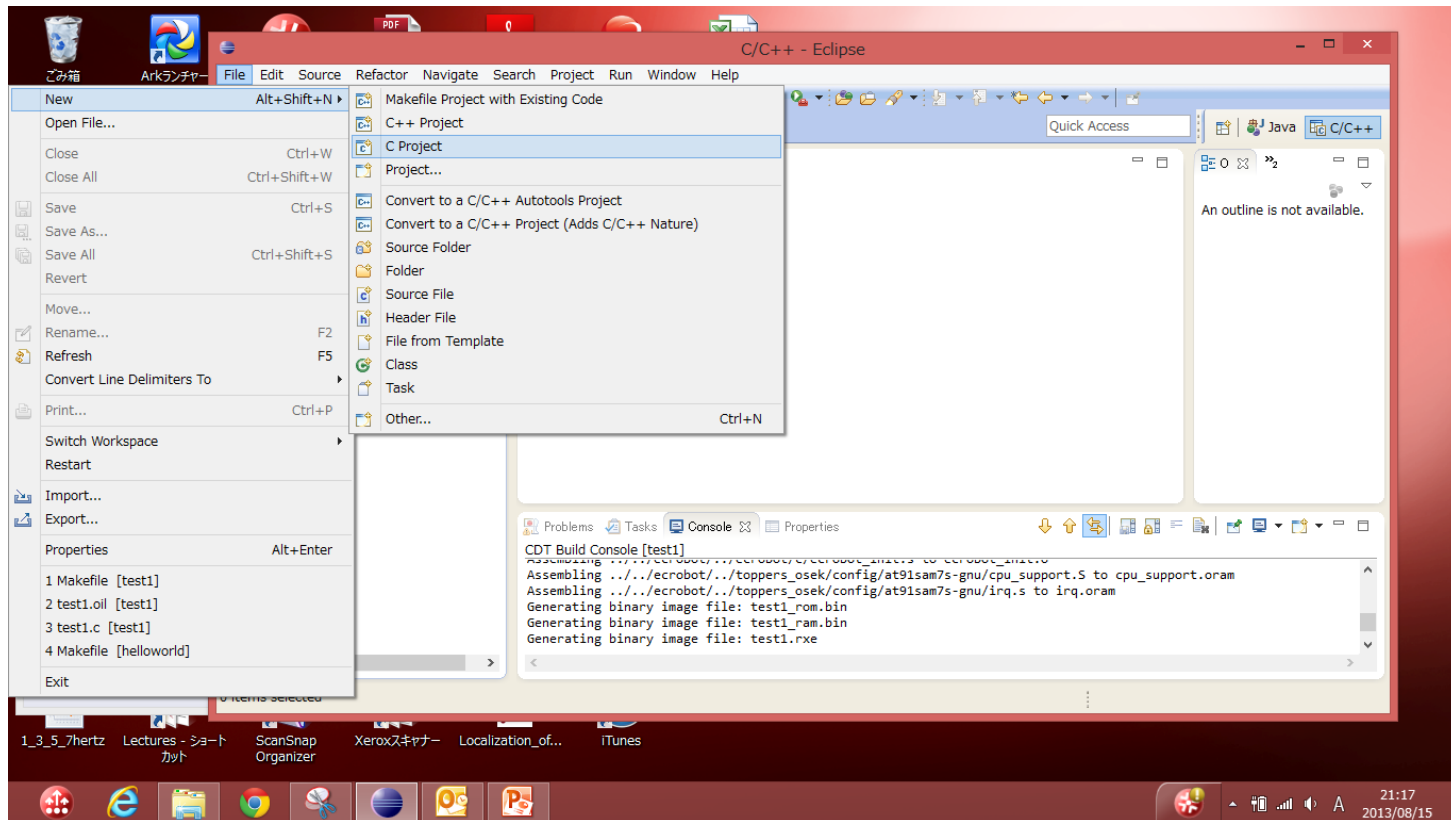
- Definition

- A real time OS (RTOS) is a preemptive multi-threading operating system which can perform CPU time scheduling **in real time**.



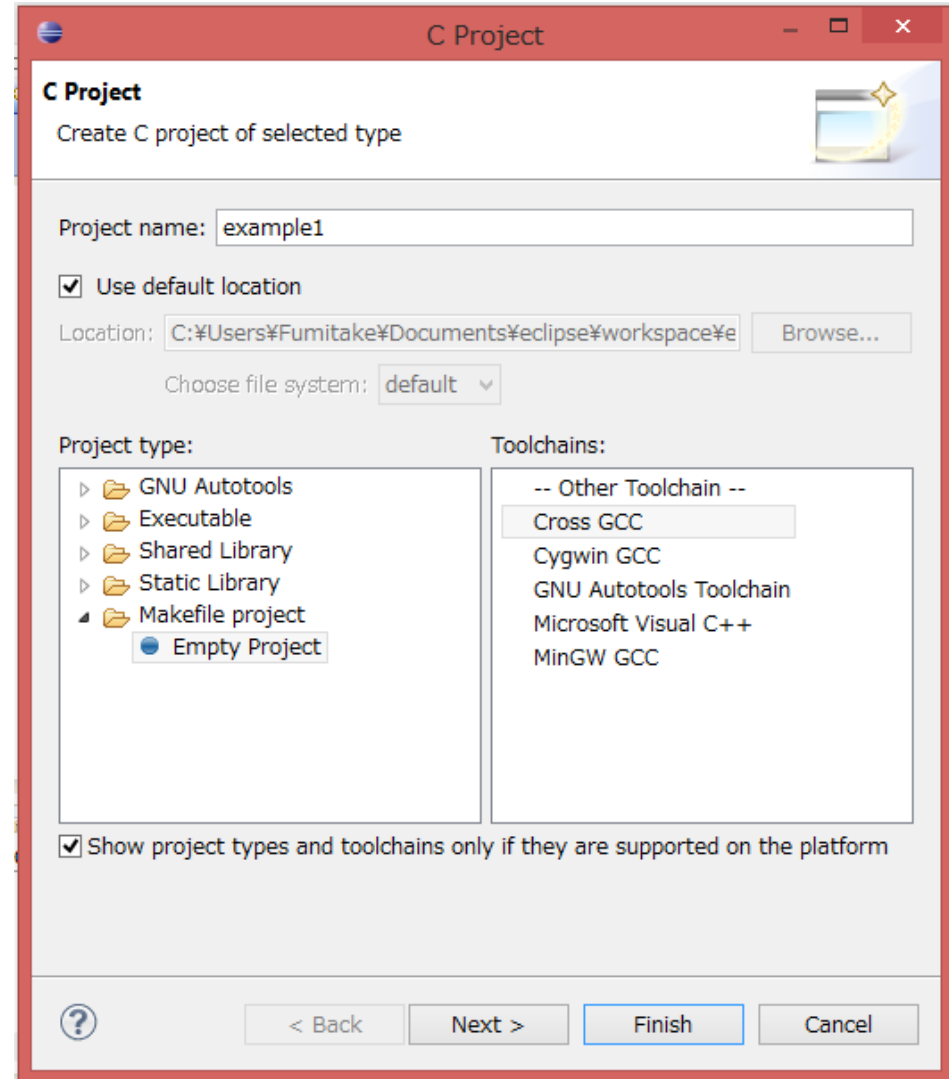
Let's start nxtOSEK programming.

- Let's start our OSEK programming with a fairly simple example.
 - Start your eclipse. Then point “File” → “New” → “C Project” as shown below.



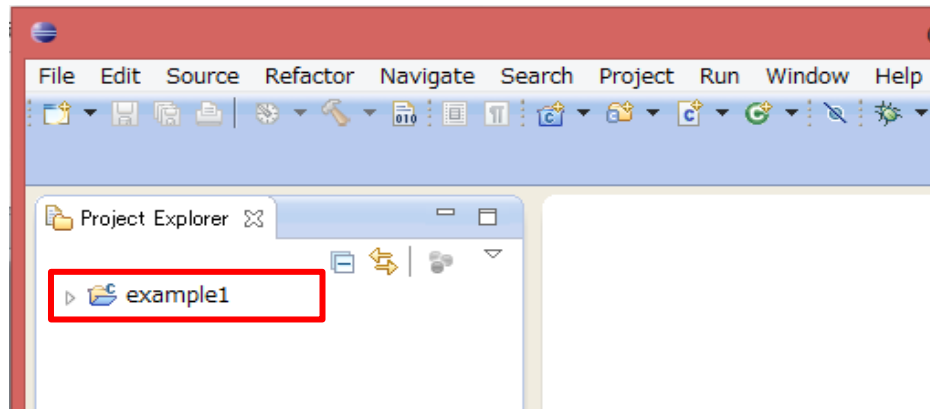
Making a new C project with eclipse

- An window will be raised. Specify following parameters
 1. Project name
 - Specify “example” here.
 2. Project type
 - Choose “Makefile project” & “Empty Project” as shown.
 3. Toolchains
 - Choose “Cross GCC” this time.
- You can alter project location by de-selecting “Use default location” if you would like to do so.
- **However, please be notified that it may affect the descriptions later.**
- ✂ Click “Finish” button to proceed.



Ready for programming : Example 1

- Then you will come up with “example1” project in your “Project explorer” in your eclipse.



- Now you have to edit the following 3 files to be added to this project.
 1. [example1.c](#) (Body of task description written in C-language)
 2. [example1.oil](#) (This file dictates how you control the motion of the task coded in example1.c.)
 3. [Makefile](#) (It specifies how final executable should be generated.)

example1.c

- Add a C-source file “example1.c” to example1 project and edit it.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"

void user_1ms_isr_type2(void){}

DeclareTask(OSEK_Task1) ;

TASK(OSEK_Task1){
    while(1){
        display_clear(0) ;
        display_goto_xy(5,3) ;
        display_string("Task1") ;
        display_update() ;
        systick_wait_ms(1000) ;
    }
}
```

```
        display_clear(0) ;
        display_goto_xy(5,5) ;
        display_string("Task2") ;
        display_update() ;
        systick_wait_ms(1000) ;
    }
}
```

Important Note : All codes are case sensitive. If you make a ‘case-insensitive’ mistake, you will be trapped in a link error which is difficult to find the cause of the trouble.

example1.oil

- Save following text as “example1.oil”.

```
#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;
```

```
};
```

```
/* Definition of application mode */
```

```
APPMODE appmode1{};
```

```
/* Definition of OSEK_Task1 */
```

```
/* Task name OSEK\_Task1 should be consistent  
to the name used in your source file */
```

```
TASK OSEK\_Task1
```

```
{
```

```
AUTOSTART = TRUE
```

```
{
```

```
APPMODE = appmode1;
```

```
};
```

```
PRIORITY = 1; /* lowest priority */
```

```
ACTIVATION = 1;
```

```
SCHEDULE = FULL;
```

```
STACKSIZE = 512;
```

```
};
```

```
};
```

Makefile

- The Makefile

```
# Target specific macros
TARGET = example1

TARGET_SOURCES = ¥
                example1.c

TOPPERS_OSEK_OIL_SOURCE = ./example1.oil

# Don't modify below part

O_PATH ?= build

include /cygdrive/c/cygwin/nxtOSEK/ecrobot/ecrobot.mak
```

- Full-path of “ecrobot.mak” will vary depending on where you put your nxtOSEK installation.
- Do not use MS-DOS type path description.

Compiling the project and send it to LEGO.

- Once you completed preparing these files, try making the binary.
- Build a project
 - Select “example1” project in project explorer window. (This can be done by single clicking the project)
 - “Build” (in the menu) → “Build project”

```
Assembling /cygdrive/c/cygwin/nxtOSEK/ecrobot/./toppers_osek/c
```

```
Generating binary image file: example1_rom.bin
Generating binary image file: example1_ram.bin
Generating binary image file: example1.rxe
```

These 3 statements prove your success in building you project.

```
22:23:42 Build Finished (took 12s.520ms)
```

- Copy it to LEGO NXT
 - Connect you lego with your PC with USB and power on.
 - “Run” (in the menu) → “External tool...” (If you have successfully configured)

```
Executing NeXTTool to upload example1.rxe...
example1.rxe=17328
NeXTTool is terminated.
```

What will happen with your LEGO ?



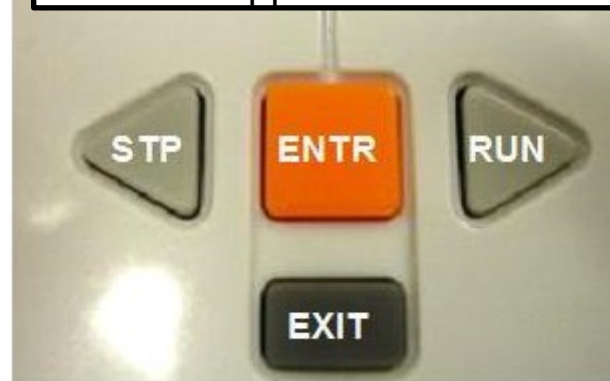
Just after you have power on your brick



A splash – It comes after you have selected a particular OSEK application.



Main screen of main monitor (caution : your application is not running yet.)



How to control execution of your application. (EXIT=shutdown)

The OIL file

- OIL = **O**sek **I**mplementation **L**anguage
 - The OIL file describes OS properties, resource configurations, as well as how your threads(tasks) are being executed.

```
CPU ATMEL_AT91SAM7S256
```

```
{
  OS LEJOS_OSEK
  {
    ....
  };

```

```
  APPMODE appmode1{};

```

```
  TASK Task1
```

```
  {
    ....
  };

```

```
TASK Task2
```

```
{
  ....
};
  COUNTER SysTimerCnt
```

```
{
  ....
};

```

```
  ALARM periodic_alarm1
```

```
{
  ....
};
};

```

Red colored keywords are called “**objects**.”



OBJECTS of OIL file

- Here we summarize possible OBJECTS to be included in an oil file.

| OBJECT Name | Description |
|-------------|--|
| CPU | Specifies CPU to be used. This object is used as a container of all other objects. Should not be altered. |
| OS | Used to define OS properties. Should not be altered. |
| APPMODE | This object defines an application mode. At least 1 application mode is necessary. |
| TASK | This object determines how your corresponding task is executed. An OIL can possibly contain more than 2 TASK objects if your application is multi-task. |
| COUNTER | This objet defines a periodic counter. It is used to define an activation interval of a task which is executed periodically. |
| ALARM | This object defines a periodic counter using COUNTER object as previously defined. |

Details of example1.oil (1)

```
#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;
```

```
};
```

This is a CPU object.

Do not alter this line. This is a CPU used in LEGO NXT.

This is an OS object.

You should not alter/omit these descriptions in your nxtOSEK programming.

example1.oil

```
/* Definition of application mode */
APPMODE appmode1{;
```

This is an APPMODE object.
Use this line as it is.

```
/* Definition of OSEK_Task1 */
/* Task name OSEK_Task1 should be con
to the name used in your source file */
```

This is a TASK object.

This object includes **several important attributes which should be defined here.**

```
TASK OSEK_Task1
```

```
{
  AUTOSTART = TRUE
```

AUTOSTART : Specifies whether the task has been activated on startup or not. If you choose "TRUE," then APPMODE should be specified. If else you choose "FALSE," APPMODE will be specified by the task activator.

```
{
  APPMODE = appmode1;
};
```

PRIORITY : A positive integer should be assigned. Possible value is 1 to 16. Value 1 is the lowest priority. Value 16 is the highest. **It only takes some meaning if there are more than 2 tasks in your application.**

```
PRIORITY = 1; /* lowest priority */
ACTIVATION = 1;
SCHEDULE = FULL;
STACKSIZE = 512;
```

SCHEDULE : If you prefer preemptive multitasking, you should specify "FULL" here; otherwise you should put "NON" here. **Always choose "FULL" during this class.**

ACTIVATION : Number of activations of a task. **This should be "1" unless otherwise stated.**

```
};
```

```
};
```

Summary : What you have specified so far in your OIL

- You now understand the following
 - In example1 application, there is only one task which should be named “OSEK_Task1.”
 - The task “OSEK_Task1” has the lowest priority.
 - However, it does not have any specific meaning because there is only one task defined.
 - The task “OSEK_Task1” is automatically invoked when the application is executed.

example1.c

- Add a C-source file “example1.c” to example1 project and edit it.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"

void user_1m Name of the task which is
identical to OIL definition.

DeclareTask(OSEK_Task1) ;

TASK(OSEK_Task1){
    while(1){
        display_clear(0) ;
        display_goto_xy(5,3) ;
        display_string("Task1") ;
        display_update() ;
        systick_wait_ms(1000) ;
    }
}
```

```
        display_clear(0) ;
        display_goto_xy(5,5) ;
        display_string("Task2") ;
        display_update() ;
        systick_wait_ms(1000) ;
    }
}
```

What it will do:

- This task displays a message “Task1” and “Task2” alternately and endlessly.
- Display flush interval is 1 [s].

Makefile

- Makefile

- It determines the action of the CUI utility “make.” “make” makes executable of an application in a specified way as described in “Makefile.”
- So the name of the file “Makefile” has a special role and meaning. You should not change the name.
- Makefile is automatically recognized by eclipse.

```
# Target specific macros
TARGET = example1

TARGET_SOURCES = ¥
                example1.c

TOPPERS_OSEK_OIL_SOURCE = ./example1.oil

# Don't modify below part

O_PATH ?= build

include /cygdrive/c/cygwin/nxtOSEK/ecrobot/ecrobot.mak
```

You should change red-colored part depending on your program name and software installation.

Example 2 : Driving nxt motors

- Let's drive motors with your OSEK application.



Connect your NXT brick to your PC with USB cable. Plug a motor unit to [port A](#), the other to [port B](#) using the cable supplied with your group's LEGO set.

motor1.c

- Create a new project “motor1” in your eclipse and add the following C source to the project.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"

void ecrobot_device_initialize(){
    nxt_motor_set_speed(NXT_PORT_A, 0, 0);
    nxt_motor_set_speed(NXT_PORT_B, 0, 0);
}

void ecrobot_device_terminate(){
    nxt_motor_set_speed(NXT_PORT_A, 0, 0);
    nxt_motor_set_speed(NXT_PORT_B, 0, 0);
}

void user_1ms_isr_type2(void){}

DeclareTask(Task1);
```

```
TASK(Task1){
    nxt_motor_set_speed(NXT_PORT_A, 70, 0);
    nxt_motor_set_speed(NXT_PORT_B, 70, 0);
    systick_wait_ms(2000); /* wait for 2 seconds */
    nxt_motor_set_speed(NXT_PORT_A, 80, 0);
    nxt_motor_set_speed(NXT_PORT_B, -80, 0);

    systick_wait_ms(2000);

    nxt_motor_set_speed(NXT_PORT_A, 0, 1); /* brake
mode */
    nxt_motor_set_speed(NXT_PORT_B, 0, 1); /* brake
mode */

    display_string("That's all!");
    display_update();

    TerminateTask();
}
```

motor1 : motor1.oil

- OIL : motor1.oil

```
#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;
  }
};
```

```
/* Definition of application mode */
APPMODE appmode1{};

/* Definition of motor1 */
TASK Task1
{
  AUTOSTART = TRUE
  {
    APPMODE = appmode1;
  };
  PRIORITY = 1; /* lowest priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
};
```

motor1 : Makefile

- Only corresponding file names are modified in Makefile description.

```
# Target specific macros
TARGET = motor1

TARGET_SOURCES = ¥
                motor1.c

TOPPERS_OSEK_OIL_SOURCE = ./motor1.oil

# Don't modify below part

O_PATH ?= build

include /cygdrive/c/cygwin/nxtOSEK/ecrobot/ecrobot.mak
```

motor1 : Let's try

When you are ready, flash your program to LEGO.
Then see what will happen when it is executed.

ECRobot API

- What is ECRobot API ?
 - ECRobot API is **a set of library functions which provides programming interface to utilize features available with your LEGO NXT.**
 - As an application programmer, you can integrate sensors, motors into your robot system. It can be controlled by your program **with the use of APIs.**
 - You can also use internal functions of NXT brick (like bluetooth, some communication protocols) to make your system sophisticated by using appropriate APIs.
- API for controlling motor motion.
 - `void nxt_motor_set_speed(U32 PORT, int speed_percent, int brake)`
 - Arguments
 - PORT : Specifies the PORT to which the motor is connected. (NXT_PORT_A, NXT_PORT_B, NXT_PORT_C)
 - speed_percent: -100 to 100 (**Should be an integer**)
 - brake : 0 (float) or 1 (brake)
 - If 0 is chosen, then the motor **gradually loses the velocity (if speed=0)** and will stop after a while.
 - If 1 is specified, then the motor **stops almost immediately.**

Initialize and terminate functions

```
#include "kernel.h"
```

This function is being executed when you start your program.

```
void ecrobot_device_initialize(){
    nxt_motor_set_speed(NXT_PORT_A, 0, 0);
    nxt_motor_set_speed(NXT_PORT_B, 0, 0);
}
```

This function is being executed when you stop your program.

```
void ecrobot_device_terminate(){
    nxt_motor_set_speed(NXT_PORT_A, 0, 0);
    nxt_motor_set_speed(NXT_PORT_B, 0, 0);
}
```

```
void user_1ms_isr_type2(void){}
```

```
DeclareTask(Task1);
```

```
TASK(Task1){
```

```
    nxt_motor_set_speed(NXT_PORT_A, 70, 0);
    nxt_motor_set_speed(NXT_PORT_B, 70, 0);
    systick_wait_ms(2000); /* wait for 2 seconds */
    nxt_motor_set_speed(NXT_PORT_A, 80, 0);
    nxt_motor_set_speed(NXT_PORT_B, -80, 0);
```

```
    systick_wait_ms(2000);
```

```
    nxt_motor_set_speed(NXT_PORT_A, 0, 1); /* brake mode */
    nxt_motor_set_speed(NXT_PORT_B, 0, 1); /* brake mode */
```

```
    display_string("That's all!");
    display_update();
```

```
    TerminateTask();
```

```
}
```


Tips in reading the API reference

- Since nxtOSEK is an programming environment **to develop embedded system application**, there are some dialects which you'd better know.
- Variable type aliases used in the reference.
 - **U8** : unsigned char
 - **S8** : char
 - **U16** : unsigned short
 - **S16** : short
 - **U32** : unsigned int
 - **S32** : int

Example 3 : Multitask

- We will define 2 tasks (named **Task1** and **Task2**) executed in sequel.
- Priorities of the tasks
 - Task 1 --- Assigned lowest priority **1**.
 - Task 2 --- Relatively higher priority **2**.

See what will happen when it is executed.
(C source, OIL and the Makefile are given in the preceding pages)

Example3 : twotasks.c

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"

#define COUNT 501
DeclareTask(Task1);
DeclareTask(Task2);

/* Hooking 1[ms] timer */
void user_1ms_isr_type2(void){}
```

```
TASK(Task1){
    int i;
    for(i = 0 ; i < COUNT ; i++){
        display_goto_xy(0,1);
        display_string("task1 = ");
        display_goto_xy(6,1);
        display_int(i,5);
        display_update();
        systick_wait_ms(10);
    }
    TerminateTask();
}
```

```
TASK(Task2){
    int j;

    for(j = 0 ; j < COUNT ; j++){
        display_goto_xy(0,2);
        display_string("task2 = ");
        display_goto_xy(6,2);
        display_int(j,5);
        display_update();
        systick_wait_ms(10);
    }

    TerminateTask();
}
```

What it will do:

- This task displays a message “Task1” and “Task2” alternately and endlessly.
- Display flush interval is 1 [s].

Example 3 : twotasks.oil

```
#include "implementation.oil"

CPU ATMEL_AT91SAM7S256
{
  OS LEJOS_OSEK
  {
    /* omitting details of OS section */
  };
  /* Definition of application mode */
  APPMODE appmode1{}
```

```
/* Definition of Task1 */
TASK Task1
{
  AUTOSTART = TRUE
  {
    APPMODE = appmode1 ;
  };
};
```

```
PRIORITY = 1; /* lowest priority */
ACTIVATION = 1;
SCHEDULE = FULL;
STACKSIZE = 512;
};
```

```
/* Definition of Task2 */
TASK Task2
{
  AUTOSTART = TRUE
  {
    APPMODE = appmode1 ;
  };
  PRIORITY = 2; /* has higher priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
};
};
```

[Discussion] Which task will be executed first ? Task 1 or Task 2 ?

Discuss with your teammates.



Example 2 : Makefile

```
# Target specific macros
TARGET = twotasks

TARGET_SOURCES = ¥
                twotasks.c

TOPPERS_OSEK_OIL_SOURCE = ./twotasks.oil

# Don't modify below part

O_PATH ?= build

include /cygdrive/c/cygwin/nxtOSEK/ecrobot/ecrobot.mak
```

What was modified as compared to previous Makefile are colored red.

[Optional] What priority means to task execution

- In class assignment
 - Modify the priority setting in “exercise2.oil” file as shown below, and try to execute the application.
 - Priority of Task1 \rightarrow 2
 - Priority of Task2 \rightarrow 1
 - See what happens. Carefully observe the changes.

This page is intentionally
left blank.

Example 4 : Realtime Multitask

- We will define 2 tasks (named **Task1** and **Task2**) executed in **parallel**.
- Priorities of the tasks
 - Task 1 --- Assigned lowest priority **1**.
 - Task 2 --- Relatively higher priority **2**.
- Periodic execution
 - Task 1 --- Executed in every 20 [ms].
 - Task 2 --- Executed in every 20 [ms].

See what will happen when it is executed.
(C source, OIL and the Makefile are given in the preceding pages)

Exercise 4 : Real Multitasking (para2tasks.c)

```

#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"

#define COUNT 501

DeclareCounter(SysTimerCnt);
DeclareTask(Task1);
DeclareTask(Task2);

/* Hooking 1[ms] timer */
void user_1ms_isr_type2(void)
{
    SignalCounter(SysTimerCnt); /* Increment cnt */
}

TASK(Task1){
    static int i=0;

    if(i <= COUNT){
        display_goto_xy(0,1);
        display_string("task1 = ");
        display_goto_xy(8,1);
        display_int(i,5);
        display_update();
        i++;

```

```

    } else {
        display_goto_xy(0,4);
        display_string("TASK1 Terminated");
        display_update();
    }
    TerminateTask();
}

TASK(Task2){
    static int j=0;

    if(j <= COUNT){
        display_goto_xy(0,2);
        display_string("task1 = ");
        display_goto_xy(8,2);
        display_int(j,5);
        display_update();
        j++;
    } else {
        display_goto_xy(0,5);
        display_string("TASK2 Terminated");
        display_update();
    }
    TerminateTask();
}

```

Example 4 : para2tasks.oil

```

#include "implementation.oil"
CPU ATMEL_AT91SAM7S256
{
  OS LEJOS_OSEK
  {
    // omitted hence it's completely same as before
  };

  /* Definition of application mode */
  APPMODE appmode1{ };

  TASK Task1 /* Definition of Task1 */
  {
    AUTOSTART = FALSE ;
    PRIORITY = 1; /* lowest priority */
    ACTIVATION = 1;
    SCHEDULE = FULL;
    STACKSIZE = 512;
  };

  TASK Task2 /* Definition of Task2 */
  {
    AUTOSTART = FALSE ;
    PRIORITY = 2; /* lowest priority */
    ACTIVATION = 1;
    SCHEDULE = FULL;
    STACKSIZE = 512;
  };

```

```

COUNTER SysTimerCnt
{
  MINICYCLE = 1 ;
  MAXALLOWEDVALUE = 10000 ;
  TICKSPERBASE = 1 ; /* 1[tick] = 1[ms] */
};

ALARM periodic_alarm1
{
  COUNTER = SysTimerCnt ;
  ACTION = ACTIVATETASK {
    TASK = Task1 ;
  };
  AUTOSTART = TRUE {
    ALARMTIME = 1 ;
    CYCLETIME = 20 ;
    APPMODE = appmode1 ;
  };
};

ALARM periodic_alarm2
{
  COUNTER = SysTimerCnt ;
  ACTION = ACTIVATETASK {
    TASK = Task2 ;
  };
  AUTOSTART = TRUE {
    ALARMTIME = 1 ;
    CYCLETIME = 20 ;
    APPMODE = appmode1 ;
  };
};
};
};

```



Example 4 : Makefile

- Only corresponding file names are modified in Makefile description.

```
# Target specific macros
TARGET = para2tasks

TARGET_SOURCES = ¥
                para2tasks.c

TOPPERS_OSEK_OIL_SOURCE = ./para2tasks.oil

# Don't modify below part

O_PATH ?= build

include /cygdrive/c/cygwin/nxtOSEK/ecrobot/ecrobot.mak
```

Example 4 : Test run.

- There are 2 tasks defined in the C source and the oil file.
- Please be aware that C source is different from the one used in the previous example 3.

Seeing is believing!

See what will happen when it is executed.

Example 4 : para2tasks.oil

```

#include "implementation.oil"
CPU ATMEL_AT91SAM7S256
{
  OS LEJOS_OSEK
  {
    // omitted hence it's completely same as before
  };

  /* Definition of application mode */
  APPMODE appmode1{};

TASK Task1 /* Definition of Task1 */
{
  AUTOSTART = FALSE ;
  PRIORITY = 1; /* lowest priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
};

TASK Task2 /* Definition of Task2 */
{
  AUTOSTART = FALSE ;
  PRIORITY = 2; /* relatively high priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
};

```

```

COUNTER SysTimerCnt
{
  MINICYCLE = 1 ;
  MAXALLOWEDVALUE = 10000 ;
  TICKSPERBASE = 1 ; /* 1[tick] = 1[ms] */
};

ALARM periodic_alarm1
{
  COUNTER = SysTimerCnt ;
  ACTION = ACTIVATETASK {
    TASK = Task1 ;
  };
  AUTOSTART = TRUE {
    ALARMTIME = 1 ;
    CYCLETIME = 20 ; /* in [ms] */
    APPMODE = appmode1 ;
  };
};

ALARM periodic_alarm2
{
  COUNTER = SysTimerCnt ;
  ACTION = ACTIVATETASK {
    TASK = Task2 ;
  };
  AUTOSTART = TRUE {
    ALARMTIME = 1 ;
    CYCLETIME = 20 ;
    APPMODE = appmode1 ;
  };
};
};

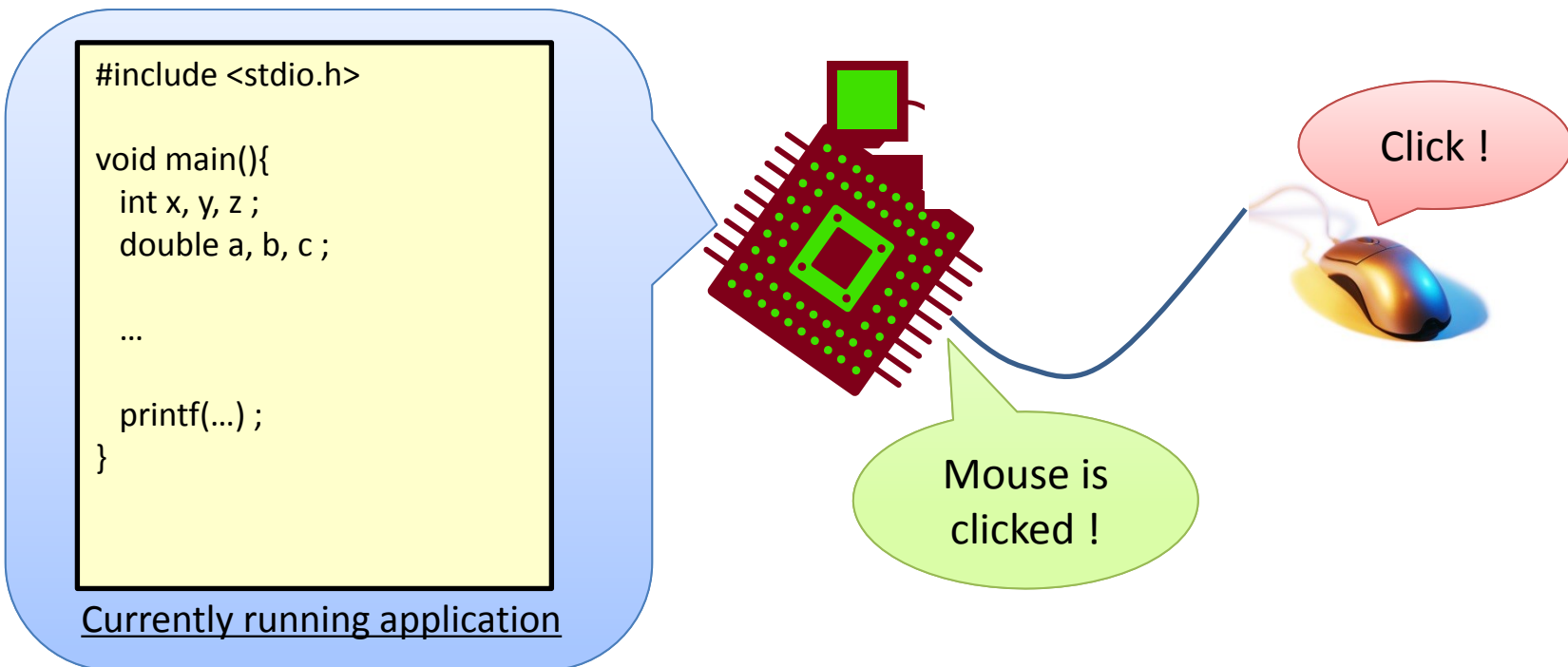
```

Task 1 in invoked by **periodic_alarm1** in every 20[ms].

Task 2 in invoked by **periodic_alarm2** in every 20[ms].

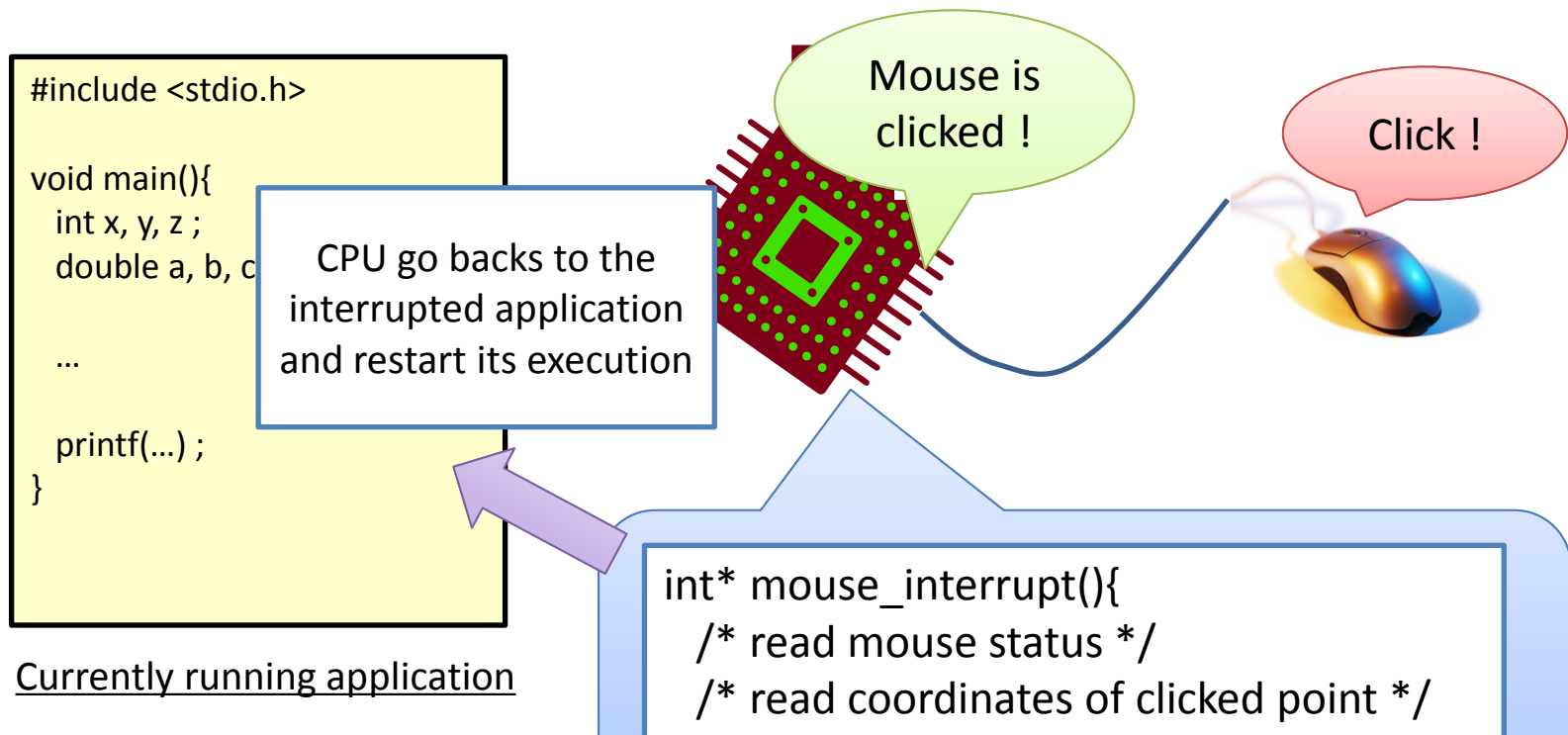
Interrupt handler and hook (1)

- Every CPU has signal pins labeled “intXX.” This is a logical signal line to tell CPU that external devices requires immediate service.
- This logical scheme is called “**interrupt**”. How to handle interrupt request should also be provided as a program. The program to serve for interrupt request is specifically called “**interrupt handler**.”



Interrupt handler and hook (2)

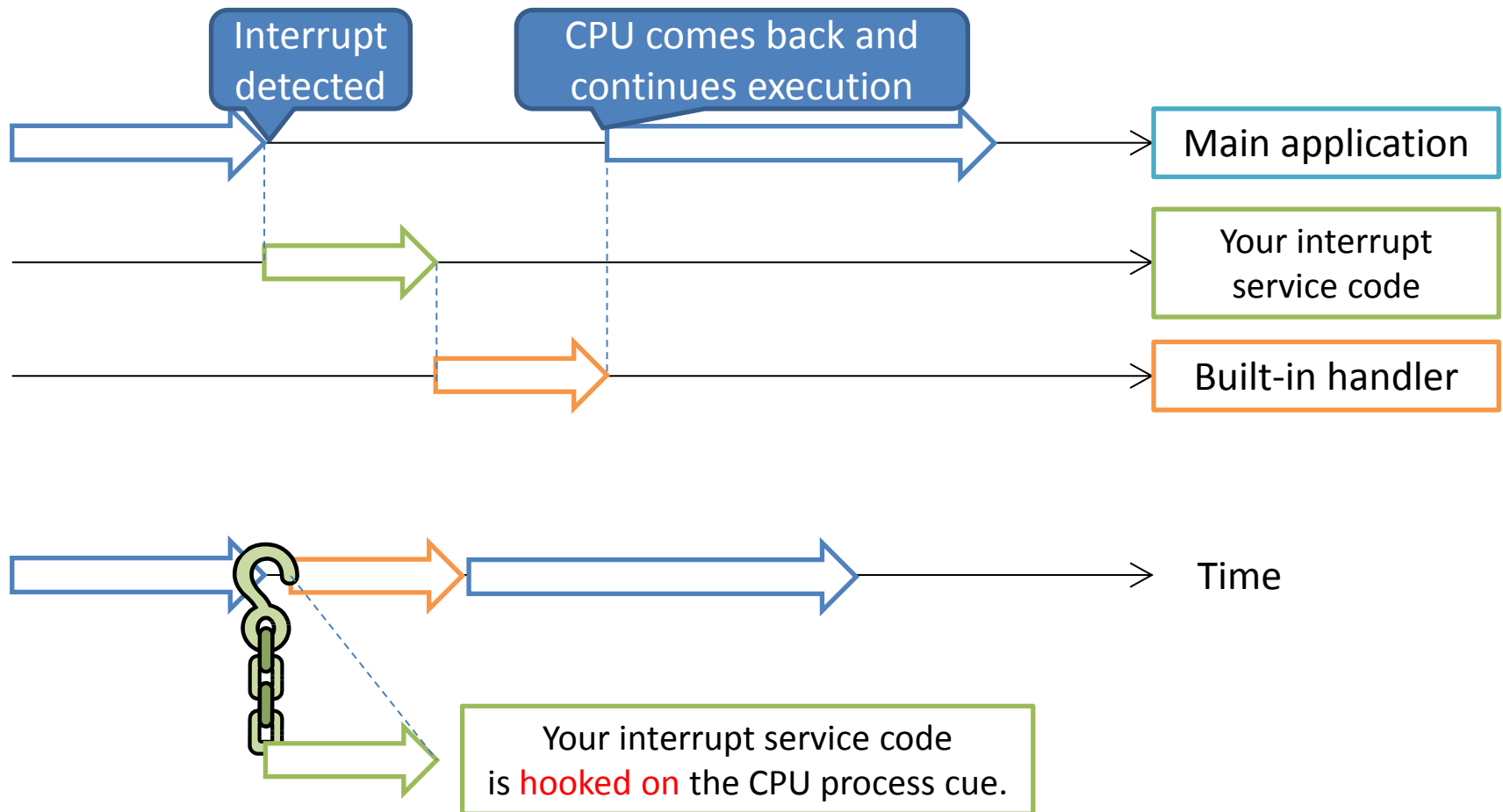
- High level operating system (MS windows / MacOS / linux ...) **has its own interrupt handler for every possible interrupt.**



Is it possible to execute your original code when interrupt is detected ?
The answer is YES!

Interrupt handler and hook (3)

- To insert your original code before the built-in interrupt handler, you have to **hook your code** a built-in handler, as can be depicted in the figure.



Three hooks available in nxtOSEK

- `void ecrobot_device_initialize(void)`
 - This is a start up hook.
 - Codes inside this function will be executed at startup of NXT brick.
 - Should contain initialization of sensors and connections.
- `void ecrobot_device_terminate(void)`
 - This is a close down hook.
 - Codes inside this function will be executed at closing down of NXT brick.
 - Should contain inactivation and termination of connections.
- `void user_1ms_isr_type2(void)`
 - This is a hook for 1[ms] timer interrupt.
 - Task cycletime will be generated by this function. Use [SignalCounter](#) function to increment system timer count.

Exercise 4 : Real Multitasking (para2tasks.c)

```

#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"

#define COUNT 501

DeclareCounter(SysTimerCnt);
DeclareTask(Task1);
DeclareTask(Task2);

/* Hooking 1[ms] timer */
void user_1ms_isr_type2(void)
{
    SignalCounter(SysTimerCnt); /* Increment cnt */
}

TASK(Task1){
    static int i=0;

    if(i <= COUNT){
        display_goto_xy(0,1);
        display_string("task1 = ");
        display_goto_xy(8,1);
        display_int(i,5);
        display_update();
        i++;
    }
}

```

This routine is hooked on built-in timer interrupt routine.

Necessary to increase system timer count

```

} else {
    display_goto_xy(0,4);
    display_string("TASK1 Terminated");
    display_update();
}
TerminateTask();
}

TASK(Task2){
    static int j=0;

    if(j <= COUNT){
        display_goto_xy(0,2);
        display_string("task1 = ");
        display_goto_xy(8,2);
        display_int(j,5);
        display_update();
        j++;
    } else {
        display_goto_xy(0,5);
        display_string("TASK2 Terminated");
        display_update();
    }
    TerminateTask();
}
}

```

How periodic tasks are implemented in nxtOSEK.

```

COUNTER SysTimerCnt
{
  MINCYCLE = 1 ;
  MAXALLOWEDVALUE = 10000 ;
  TICKSPERBASE = 1 ; /* 1[tick] = 1[ms] */
};
ALARM periodic_alarm1
{
  COUNTER = SysTimerCnt ;
  ACTION = ACTIVATETASK {
    TASK = Task1 ;
  };
  AUTOSTART = TRUE {
    ALARMTIME = 1 ;
    CYCLETIME = 20 ; /* in [ms] */
    APPMODE = appmode1 ;
  };
};
ALARM periodic_alarm2
{
  COUNTER = SysTimerCnt ;
  ACTION = ACTIVATETASK {
    TASK = Task2 ;
  };
  AUTOSTART = TRUE {
    ALARMTIME = 1 ;
    CYCLETIME = 20 ;
    APPMODE = appmode1 ;
  };
};
};
};

```

There are some important information for coding realtime application in nxtOSEK.

- 1) Periods of periodic tasks are generated by the internal timer of ARM7 cpu.
- 2) In order to code a realtime task, you have to define **COUNTER** object in OIL.
- 3) You have also to define **ALARM** object in your oil file. Your ALARM object can be more than 2 as this example shows.
- 4) The minimum CYCLETIME (Execution interval) is 1[ms]. However, **please keep it in mind** that most of the CPU time might be occupied by 1[ms] task, which can be a cause of trouble.



This page is intentionally
left blank.

Challenge 1 : Accelerating and Decelerating the motor

- Build a **motor velocity control application** which satisfies the following technical specification.
- [Specification]
 1. Connect a motor to NXT brick. Use port A unless port A is not functioning properly .
 - Increase or decrease the speed of motor rotation depending on the status of push sensors.
 - Use the API `nxt_motor_set_speed()`.
 2. Connect 2 touch sensors to PORT S1 and S2.
 - In case your program detects that S1 is pushed, increase the speed of the motor to some amount .
 - In case your program detects that S2 is pushed, decrease the speed of the motor to some amount.
 - Use the API `ecrobot_get_touch_sensor()`. You should consult your API handout for details of the function.
 3. Display the current power setting to brick's LCD.

Give Thoughts on CPU time distribution

- As you understand, there are 3 distinct tasks to be executed.

Task 1

Reading the status of two push sensors.

Task 2

Modifying the speed of the motor.

Task 3

Display the current motor speed.

- **[Discussion]**
 - Which task should be executed periodically (not restricted to only 1 task) ?
 - Which task should be given the shortest cycletime ? (In other words, which task should be most frequently executed ?)
 - Which task is the least important of all ? (It is adequate to execute this task in back ground. No realtime execution is necessary.)
 - **[Advanced Question]**
 - What is the suitable cycletime for tasks which you think they should be executed in real time.

Start Coding!

- Start coding with API reference beside you.
 - Just try to start coding.
 - You have to make 3 files.
 - The C source
 - OIL file
 - The Makefile.
 - Don't fear for making mistakes!
 - Your LEGO brick will not be broken even if you made some **serious** mistake!
 - **Code it ! Try it ! And debug it!**
 - To get the status of push sensor
 - API function : `U8 ecrobot_get_touch_sensor(U8 port_id)` will help.

Challenge 2 ! Continuous velocity control

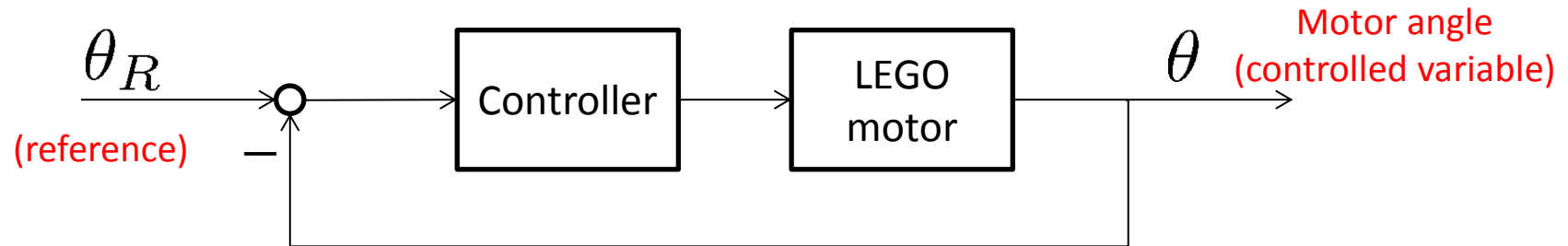
- In the previous challenge, we use two touch sensors in order to take acceleration/deceleration action.
- However, there are several sensors included in your LEGO set.
 - 2 touch sensors
 - 1 sound sensor (returns a value reversely proportional to loudness of a sound)
 - 1 sonar sensor (returns a distance between the sensor surface and the obstacle in [cm])
 - 1 light sensor (returns a value proportional to reflection rate of a light)
 - It returns a large value if the target is dark, and *vice-versa*.
- All these sensors are supported by erobot API. **Try using one of these (except touch sensor) to implement smooth velocity control.**
 - Discuss with your teammates which sensor to use and create an application.

That's all

- Thank you for your attendance to this class.
- I hope all teams will create *amazing and innovative systems based on your original idea and LEGO programming technique.*

[Addendum] Set Point Tracking (servo) of motor angle

- Construct a servo control system of motor angle



- Program specification
 - Control motor angle so that $\theta \rightarrow \theta_R (t \rightarrow \infty)$ in a specified time.
 - Reference angle can be given by push button manipulation.
- ECRobot APIs used in this example
 - `nxt_motor_set_speed` (\leftarrow This is the control input)
 - `nxt_motor_set_count` / `nxt_motor_get_count` (feedback signal acquisition)
 - `ecrobot_get_touch_sensor` (reference signal)

servo.c (C source for servo control) (1)

```

#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include <math.h>

#define POSITIVE NXT_PORT_S1
#define NEGATIVE NXT_PORT_S2
#define Kg 0.8

DeclareCounter(SysTimerCnt);
DeclareTask(MotorTask);
DeclareTask(TouchSensorTask);

int ref_deg = 0, motor_pow = 0, deg;
F32 int_err = 0.0;

void ecrobot_device_initialize(){
    nxt_motor_set_speed(NXT_PORT_A, 0, 0);
    nxt_motor_set_count(NXT_PORT_A, 0); /* initialize */
}

void ecrobot_device_terminate(){
    nxt_motor_set_speed(NXT_PORT_A, 0, 0);
}

```

```

void user_1ms_isr_type2(void){
    SignalCounter(SysTimerCnt);
}

TASK(MotorTask){
    int_err += (ref_deg - deg) * 0.01;
    deg = nxt_motor_get_count(NXT_PORT_A);
    motor_pow = (int)(Kg * (ref_deg - deg) + 0.5 * int_err);
    nxt_motor_set_speed(NXT_PORT_A, motor_pow, 0)

    TerminateTask();
}

TASK(TouchSensorTask){
    int go_positive, go_negative;

    go_positive = ecrobot_get_touch_sensor(POSITIVE);
    go_negative = ecrobot_get_touch_sensor(NEGATIVE);

    ref_deg += 60 * (go_positive - go_negative);
    TerminateTask();
}

TASK(DisplayTask){

```

servo.c (C source for servo control) (2)

```
TASK(DisplayTask){  
  
    while(1){  
        display_goto_xy(1,2) ;  
        display_string("power = ") ;  
        display_goto_xy(6,2) ;  
        display_int(motor_pow, 5) ;  
        display_goto_xy(1,3) ;  
        display_string("deg = ") ;  
        display_goto_xy(6,3) ;  
        display_int(deg, 5) ;  
        display_goto_xy(1,4) ;  
        display_string("ref = ") ;  
        display_goto_xy(6,4) ;  
        display_int(ref_deg, 5) ;  
        display_update() ;  
    }  
  
    TerminateTask() ;  
}
```

[Note]

If you are familiar with classical control theory, you might understand that this program is an implementation of **PI-control** about motor angle.

servo.oil (1)

```
#include "implementation.oil"
```

```
CPU ATMEL_AT91SAM7S256
```

```
{
  OS LEJOS_OSEK
  {
    /* omitted since it is the same as before */
  };
}
```

```
/* Definition of application mode */
```

```
APPMODE appmode1{};
```

```
/* Definition of MotorTask */
```

```
TASK MotorTask
{
  AUTOSTART = FALSE ;
  PRIORITY = 2; /* lowest priority */
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
};
```

```
/* Definition of TouchSensorTask */
```

```
TASK TouchSensorTask
```

```
{
  AUTOSTART = FALSE ;
  PRIORITY = 3;
  ACTIVATION = 1;
  SCHEDULE = FULL;
  STACKSIZE = 512;
};
```

```
TASK DisplayTask
```

```
{
  AUTOSTART = TRUE
  {
    APPMODE = appmode1 ;
  };
  PRIORITY = 1 ;
  ACTIVATION = 1 ;
  SCHEDULE = FULL ;
  STACKSIZE = 512 ;
};
```

servo.oil (1)

```

COUNTER SysTimerCnt
{
    MINCYCLE = 1 ;
    MAXALLOWEDVALUE = 10000 ;
    TICKSPERBASE = 1 ; /* 1[tick] = 1[ms] */
};

ALARM periodic_alarm1
{
    COUNTER = SysTimerCnt ;
    ACTION = ACTIVATETASK
    {
        TASK = TouchSensorTask ;
    };
    AUTOSTART = TRUE
    {
        ALARMTIME = 1 ;
        CYCLETIME = 100 ;
        APPMODE = appmode1 ;
    };
};

```

```

ALARM periodic_alarm2
{
    COUNTER = SysTimerCnt ;
    ACTION = ACTIVATETASK
    {
        TASK = MotorTask ;
    };
    AUTOSTART = TRUE
    {
        ALARMTIME = 1 ;
        CYCLETIME = 15 ;
        APPMODE = appmode1 ;
    };
};

```

I believe you can write Makefile by yourself.