Module NXT
LEGO MINDSTORMS TUTORIAL

ESWEEK 2009
http://nxtgcc.sf.net
Rasmus Ulslev Pedersen
rup.inf@cbs.dk

Please sit close to the screen: there will be lots of small code shown...

Presentation Overview

Overview	Non-technical overview of LEGO©MINDSTORMS©NXT from a community and user standpoint
HW and SW	Description of NXT hardware and software from a developer perspective
Examples	(1) Using TinyOS as a firmware, (2) an intelligent algorithm
Q & A	There will be time for discussion and future research ideas
Lab	Drawing for the NXT 2.0 and lab exercise using it

Note 1: LEGO has sponsored a NXT 2.0 kit
Note 2: LEGO®, MINDSTORMS®are trademarks of LEGO®. The tutorial contains pictures from Atmel ARM7 documentation, and from LEGO documentation.
Note 3: There is additional material included: Many slides are supplementary and included for future reference.

Module NXT
LEGO MINDSTORMS

ESWEEK 2009
http://nxtgcc.sf.net
Rasmus Ulslev Pedersen
rup.inf@cbs.dk

Outline

Aim

- How the open source universe around Lego Mindstorms NXT is structured
- Talk about a world that stretches from high-schools to adults and from building bricks to programming in assembler
- Emphasis on the overview and the programming aspects
- Participants will be able to see where in the NXT programming spectrum their interest match most
- Use the tutorial to get started with open source programming on NXT

Introduction

- Description of open source hardware and software
- LEGO business models are discussed
- The old RCX
- NXT in several (online) communities
- Overview of how NXT can be programmed in different environments

Some Mindstorms History

- 10 years since Mindstorms was released
- First LEGO Mindstorms was the LEGO RCX: Successful
- LEGO intended it to be a closed source product, but...
- It was soon hacked:-)
- The open source strategy was pursued even more with the present LEGO Mindstorms
- A Goldplated NXT and a limited edition Black NXT was produced in relation to the 10 year anniversary

NXT Development

- Development of NXT started in the 2004
- Initial group was Ralph, Steven, John, and David
- First prototypes were circulated on a limited number basis inside the current MUP group and a select few outside that group
- MUP: Jan 2006, 5 people
- MUP II: Nov. 2005, 11 people
- MDP: 2006
- MCP 2: 31 persons, 100 chosen from 10000 applicants
- MCP 3: 2008-2009
- MCP 4: Next slide

LEGO Design Routines

- Lego has a group of people who has the privilege of talking directly to LEGO staff
- Under non-disclosure agreements (NDA)
- First group of MCPs (MUP) were a select group
- Used as a sounding board for the ideas that went into developing the first LEGO MINDSTORMS NXT.
- MCP 4 is commencing as we speak

Lego Education

- Lego has a website for educational activities
- Strong focus on university segment
- LEGO MINDSTORMS NXT comes in educational sets
- Rechargeable battery and extra building pieces
- Sold with site licences that accompany the NXT
- Different sales curve from retail (slow start, longer product life)

Mindstorms Community

- MINDSTORMS community is large and diverse
- Blogs, forums, large events, books, the MCP program and more
- NXTStep and the NXTasy blogs
- Lugnet is a large LEGO fanclub

Schools

- Many schools participate in First LEGO League
- A tournament where current theme is played out
- The idea is to have adults lead or mentor a team of children aged 9-14 (16 outside the US)
- NXT is used to complete a challenge put out by the FLL management

Business Models

- LEGO earns money on NXT by selling the kit itself
- Licences to educational institutions are also revenue source
- First version of LEGO MINDSTORMS NXT was released in the beginning of 2007
- A second version appeared in the fall of 2009 (biannual release schedule:-)?)
- NXT 2.0 is shipped with a different set of standard sensors (color sensor)
- Spring releases at toys fairs which is then ready for Christmas sale

LEGO SYSTEMS

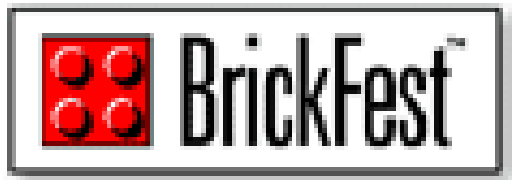
- LEGO partly produce and package NXT on their factory in Billund
- Staff around NXT development is not big
- Firmware development team is in control of the software development
- National Instruments (NI) designs the NXT-G environment
- The careful selection of close partners makes for long-time stability

Firmware Releases

- NXT is a complex product compared to other LEGO products
- Requires focus on quality
- Updated versions released on the official MINDSTORMS website (http://mindstorms.lego.com)
- First firmware had the version number 1.1 (or 1.0...) and the current released version for NXT 2.0 is versioned 1.28

Partners

- Many institutions, companies, groups, and individuals have a stake in NXT
- National Instruments is presumably the most important
- MIT, Tuft, Carnigie Mellon, and others are also active
- Others such as *nxtprograms*
- Number of large events
- This includes `Brickfest`
- There has also been a MINDSTORMS NXT Sumo Competition



NXT Alternatives

- NXT is not completely alone...
- Sun Spots from Sun/Oracle
- Fisher-Technic offers a new robot controller which is is based on an ARM9 processor running at 200MHz
- Various evaluation kits: mbed, IAR, etc.
- However, it is difficult to match dealer network, community, and history of LEGO

Source and Documents by LEGO

NXT Firmware Open Source Complete software for the NXT firmware, which can be compiled with a compliant C compiler

Software Developer Kit A kit with software for controlling NXT remotely from a host computer. It includes a description of the NXT virtual machine.

Hardware Developer Kit Description of the possible sensor and actuator interfaces for NXT. This is extensively used by thirdparty sensor vendors.

Bluetooth Developer Kit It includes the communication protocol for controlling NXT over a Bluetooth connection.

Demo Show PDF files which are important for firmware replacements...

Schematics

- Lego published the schematics
- Can see which pins of the ARM7 MCU is connected to what ports
- Lego uses Orcad for its schematics, which is a commercial program

Analog/Digital Protocols

- NXT features both analogue and digital interfaces to its sensors.
- Sensors are based on an analogue to digital conversion
- Signals are fed into the ARM processor for further processing
- Digital sensors are made with an I2C compliant protocol
- I2C is a two wire protocol that allows for a number of devices on the shared bus
- It can be monitored with a logical probe

NXT-G Blocks

- LEGO provides a proprietary programming language for LEGO MINDSTORMS NXT called NXT-G
- **G** implies it is a graphical programming language
- Places NXT-G programming blocks in a sequence that involves the usual programming constructs such as loops, branches and computations
- Tutorials and small videos with instructions for a given task
- Show Robocup NXT-G program in NXT-G...

Firmware Source

- LEGO MINDSTORMS NXT comes with an open source operating system
- Operating system is written in ANSI C
- Source code fundamentally access the physical layer such as input and output ports of the ARM processor
- Another part provides an abstraction layer to these services (Show the HPL, HAL files...)
- Operating system is a traditional round robin scheduler
- Services each a number of modules

Modules Overview

- Each module is centered on a logical or a physical unit in NXT
- Example: the display or the user interface
- The virtual machine a logical module
- Each time the virtual machine is serviced it can/will execute a number of bytecodes
- A 1 ms period (system tick) services all modules
- Show scheduler...

Fantom PC/Mac Interface

- LEGO has released an interface both for the PC and the Macintosh
- Linux is not supported at this point
- Firmware image in NXT is updated via a USB cable connection
- NXT is programmed from NXT-G either over USB or Bluetooth
- We will revisit this Fantom DLL when GNU debugger is discussed

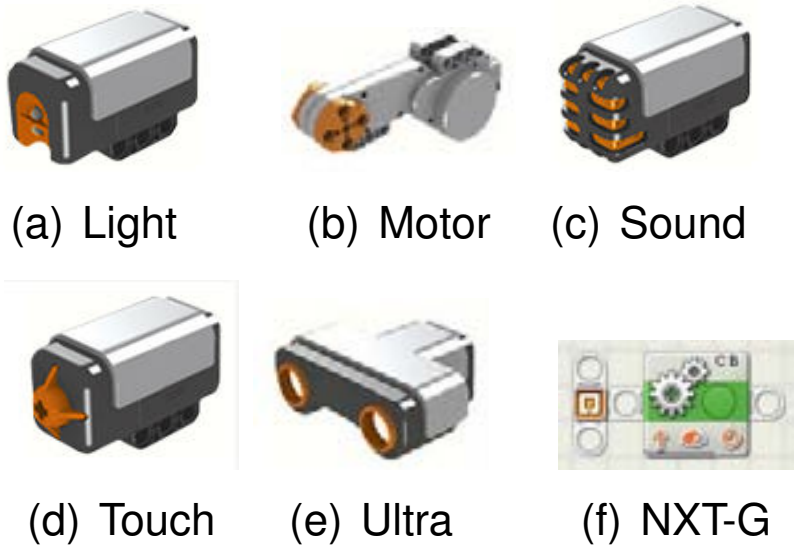
Sensor Partners and Independent Manufactures

- A limited number of third-party sensors exists
- Hitechnic, Mindsensors, and Vernier
- Hitecnic produces its sensors in a standard LEGO sensor housing
- Mindsensors make its sensors available in different shapes and forms
- Vernier specializes in natural sciences experiments
- DCP

Vernier®

Standard Sensors

- Input and output ports feature a 6-wire RJ12 connector
- Ultrasonic distance measurement sensor, a light intensity sensor, a sound sensor, a touch sensor, and motors



(a) Light (b) Motor (c) Sound

(d) Touch (e) Ultra (f) NXT-G

Standard Lego Mindstorms NXT sensors and a NXT-G block

Mindsensors

- Camera
- Realtime Clock
- Acceleration (2,3, and 5 axis reading)
- Dual Infra Red Obstacle Detector
- Motor Multiplexer for NXT
- Magnetic compass
- Pneumatic Pressure Sensor
- High Precision Infrared distance sensor (short, medium, long range)
- Various accessories: port splitter, cables, plugs, sensor kit, tools

Selected NXT sensors from Mindsensors

(g) Multi-axis acc.

(h) Magnetic compass

(i) Pneumatic pressure

(j) Infrared distance

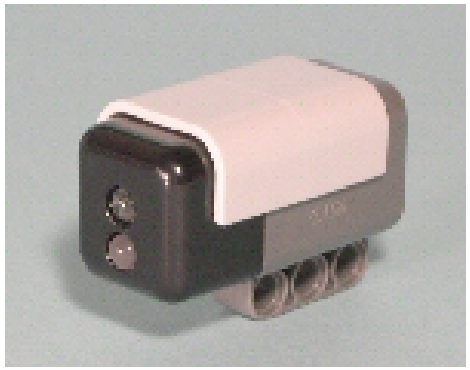
(k) Sensor building kit

(l) Temperature

Hitechnic

- Prototype Board
- Gyro
- IR seeker
- Compass
- Color sensor
- Acceleration/tilt
- Sensor and motor multiplexer
- Accessories: Cables

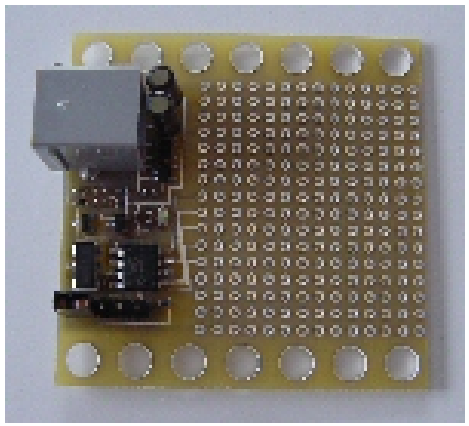
Sensors from Hitechnic



(m) Color

(n) Gyro

(o) Multiplexer



(p) Prototyping board

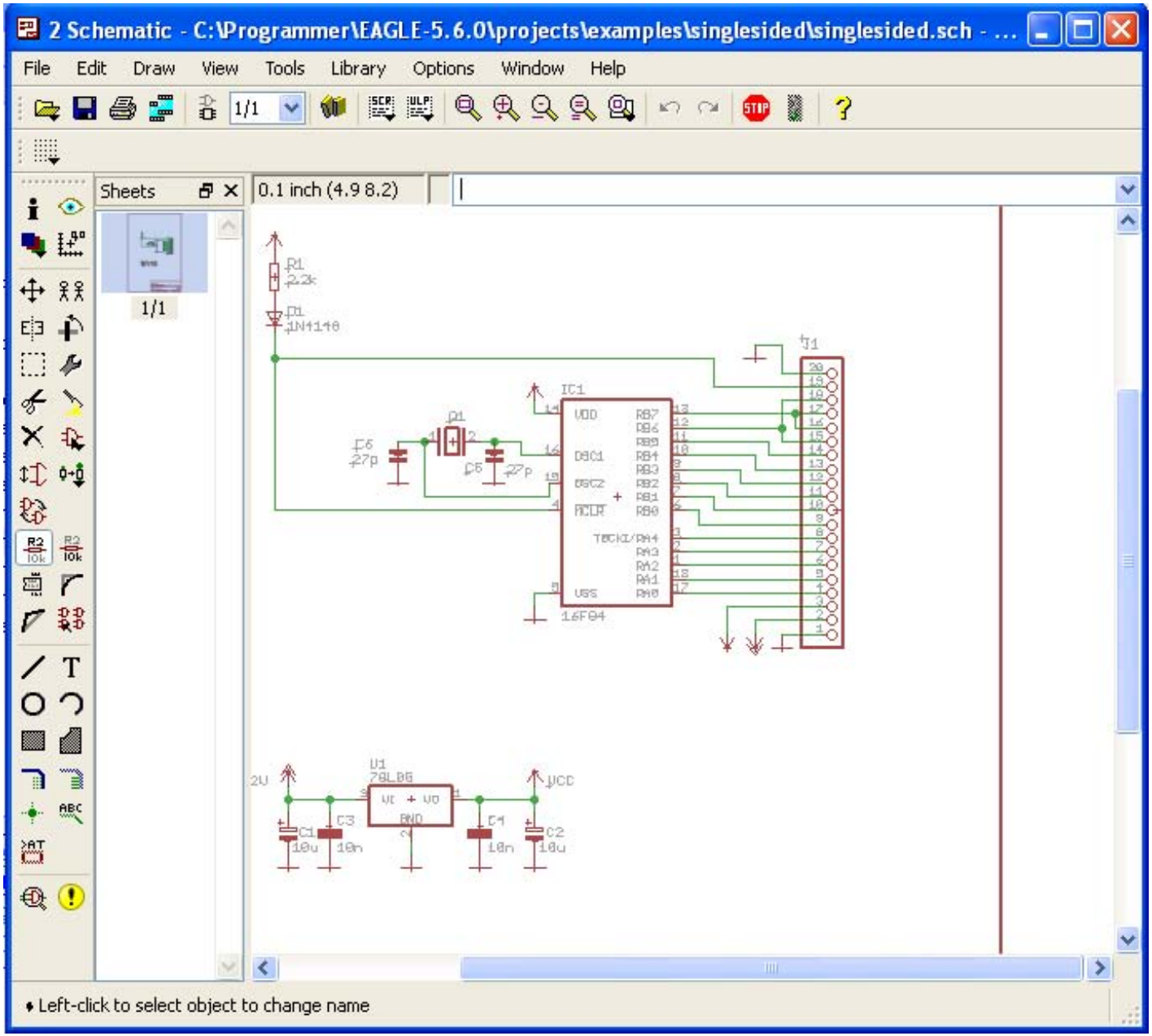
If you don't think "How fast can I get these sensors...", then something is wrong:-)"

Custom Sensors

- It is possible to create custom sensors
- EAGLE is a CAD program for creating specialized PCBs
- The PCBs can be fabricated at places such as Olimex.
- Olimex accepts EAGLE files (probably easier than Gerber files)
- Sensors can be prototyped for under $50

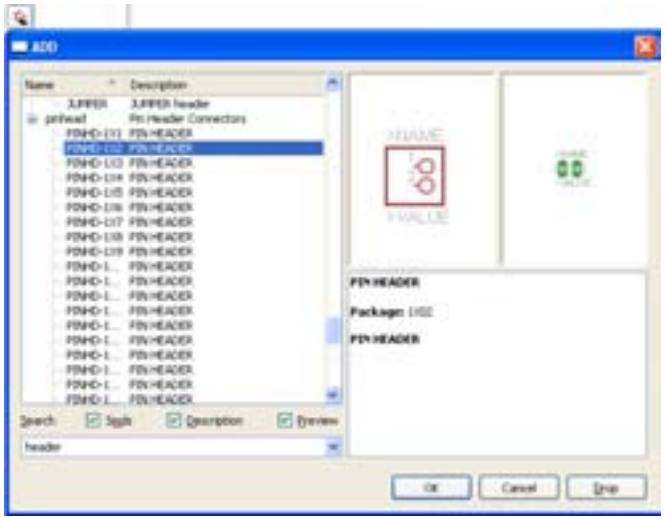
EAGLE

It is easy to modify a schematic in EAGLE



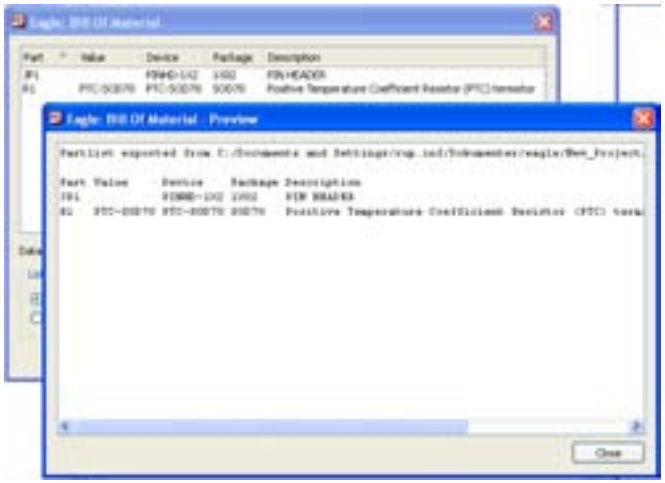
Create a (non-functional...) Demo Temperature Sensor

- Select components
- Take some pin header
- Search for termistor and add it
- Wire it
- Try `File->Run->bom.ulp`



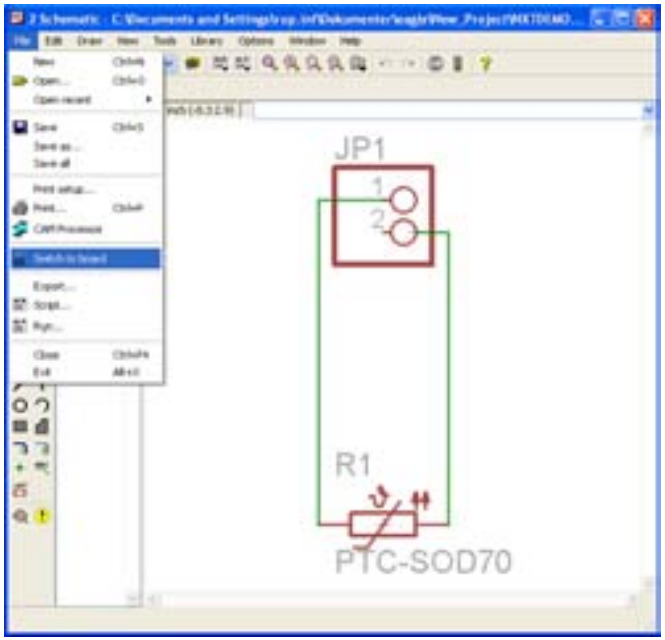
Create a Temperature Sensor

- Try
  `File->Run->bom.ulp`



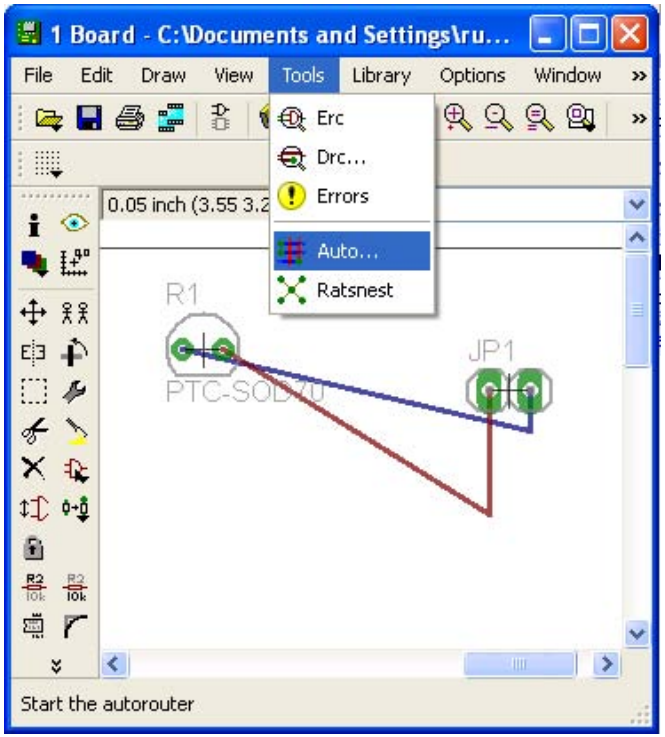
Create a Temperature Sensor

- Create the board
- `File->Switch to board`



Create a Temperature Sensor

- Take `Tools->Auto...`
- Check the layers (`View-Display/hide layers`)
- Perhaps a little overdoing with not using a single layer board...
- Resize the board by using `MOVE` corners



Create a Temperature Sensor

- Make Gerber files
- Run `excellon.cam` from the `File->CAM`
- Use GerbView (or similar to inspect files)
- or Textpad...

NXTDEMO.dri

```
Generated by EAGLE CAM Processor 5.6.0

Drill Station Info File: C:/Documents and Settings

 Date              : 10-10-2009 16:01:19
 Drills            : generated
 Device            : Excellon drill station

Parameter settings:

 Tolerance Drill +  :  2.50 %
 Tolerance Drill -  :  2.50 %
 Rotate             : no
 Mirror             : no
 Optimize           : yes
 Auto fit           : yes
 OffsetX            : 0inch
 OffsetY            : 0inch
 Layers             : Drills Holes

Drill File Info:

 Data Mode          : Absolute
 Units              : 1/10000 Inch

Drills used:

 Code  Size       used

 T01   0.0276inch    2
 T02   0.0400inch    2

Total number of drills: 4
```


NXT C Programming 101

Just to refresh out C programming skills

- NXT types
- Function pointers
- void pointers

```
typedef   struct
{
  ULONG   ModuleID;
  UBYTE   ModuleName[FILENAME_LENGTH + 1];
  void    (*cInit)(void* pHeader);
  void    (*cCtrl)(void);
  void    (*cExit)(void);
  void    *pIOMap;
  void    *pVars;
  UWORD   IOMapSize;
  UWORD   VarsSize;
  UWORD   ModuleSize;
} __attribute__((__packed__)) HEADER; /*nxtgcc converter*/
```


NXT C Programming 101

- Pointers *
- Address of &
- Dereferencing *

```
ULONG* pUL;
ULONG  someUL;
someUL = 4;
pUL = &someUL;
*pUL = 5;
//Now someUL == 5
```


NXT C Types

- typedef
- struct
- enum
- #define

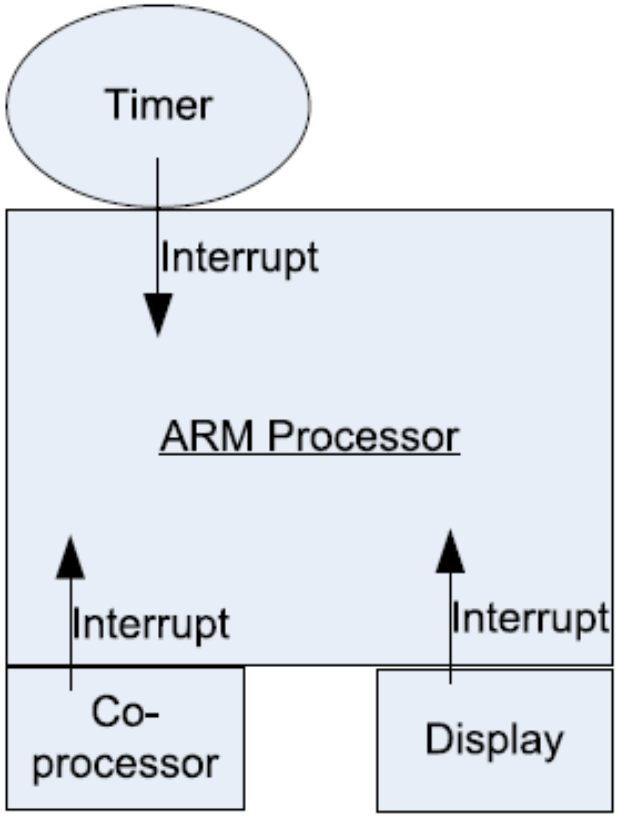
```
typedef   struct
{
  ULONG someUL;
  ...
}SOMESTRUCTYPE;
...
enum
{
  SOMEENUMVAL0,
  SOMEENUMVAL1, SOMEENUMVALEND = SOMEENUMVAL1,
};
...
ifndef   SOME_H_FILE
define   SOME_H_FILE
// code
endif //SOME_H_FILE
```


Embedded Programming

- A central micro processor controls NXT
- A second small micro processor assisting it
- Code is compiled to machine instructions which are executed one at a time
- NXT allows for firmware updates, which is the basis for alternative operating systems

NXT Firmware

- NXT actually consists of two different firmwares
- The main firmware controls the interaction with the user, and radio communication to other NXTs
- An ATmega48 generates the pulse width modulation (PWM) signals for the three motors
- It also provides input regarding the state of the push buttons on the NXT
- Host side SDK which allows for interaction with the NXT via a DLL



NXT Inside

- (a) ARM7 MCU
- (b) ATmega48 MCU
- (c) CSR BlueCore4 Bluetooth radio
- (d) SPI bus and touchpad signals
- (e) high-speed UART behind input port 4
- (f) output (generally motor) port
- (g) USB port, (h) four-button touchpad
- (i) 100x64 LCD display

NXT with important hardware highlighted

ARM to AT48 Communication

- AVR to ARM: Buttons, ADC values, and Battery
- ARM to AVR: Motor control etc.
- Show `m_sched.h` file...

```
typedef   struct
{
  UWORD   AdValue[NOS_OF_AVR_INPUTS];
  UWORD   Buttons;
  UWORD   Battery;
}IOFROMAVR;

typedef   struct
{
  UBYTE   Power;
  UBYTE   PwmFreq;
  SBYTE   PwmValue[NOS_OF_AVR_OUTPUTS];
  UBYTE   OutputMode;
  UBYTE   InputPower;
}IOTOAVR;
```


Firmware Enhancements

- Firmware enhancements are possible only because Lego released the software as open source
- IAR is a commercial compiler used by LEGO
- It was recently released in a free version for Mindstorms
- The other option for programming the existing firmware is to use GCC

Firmware Replacements

- A growing number of firmware replacements available
- Examples: lejos, NQC, pbLua, and RobotC
- A different example is TinyOS
- *leJOS* is based on a small virtual Java machine. It is a firmware replacement that allows for Java programming on Lego Mindstorms NXT. There is extensive support in forms of tutorials, well-developed APIs, and Netbeans/Eclipse programming IDEs at the leJOS website.
- Lua is a scripting language (see Lua website). It is said to perform faster than other scripting languages. Ralph Hempel has ported Lua to Lego Mindstorms NXT and labeled it *pbLua*
- *Not eXactly C* (NXC) is language similar to C. It is supported on Lego Mindstorms NXT by John C. Hansen and available from `http://bricxcc.sourceforge.net/nbc/`. It is built on top of the *Next Byte Codes* (NBC) compiler. It is a firmware replacement.

IAR Toolchains and Firmware Compilers

- The free IAR compiler can be downloaded from IAR's homepage
- It is an industry strength programming environment called *Embedded Workbench*
- It interacts with a JTAG programmer connected to Lego Mindstorms NXT via the IAR C-SPY debugger
- The visualSTATE program interacts with the C-SPY debugger to give a graphical view of the debugging session

GCC

- The GCC compiler can be downloaded from the nxtgcc home page
- It is an *arm-elf-gcc* compiler
- There exists several good toolchains such as WinArm, CodeSurgery etc.
- Show nxtgcc Eclipse project...
- `binsert.c`
- `nxtgcc.rfw`
- `ConvertNxtGcc.java` and `NGU.java`
- `Source-xx` $\Longrightarrow$ `Source` copy
- `m_sched.map`

GCC Compliance Process

- Changes needed to make the source code compile with an arm-elf based GCC
- See details in `ConvertNxtGcc.java` program
- First versions of NXTGCC were error-prone
- IAR supports, and Lego uses, nested flexible array members
- GCC does not support this
- Key was to change the nested flexible array to become a fixed size nested array

Nested Array

Show `Display.txt` in `SourceOrig` and in `Source`

```
typedef   struct
{
  UBYTE   FormatMsb;
  ...
  UBYTE   Data[];
}
BMPMAP;
```

```
typedef   struct
{
  UBYTE   FormatMsb;
  ...
  UBYTE   PixelsY;
  /*nxtgcc converter*/
  UBYTE   Data[1];
}
/*nxtgcc converter*/
__attribute__((_packed__))
BMPMAP;
```


ARM7

- Characterized by a number of modes, exception types, and interrupts
- Associated with each mode is a current program status register (CPSR)
- Each exception and interrupt type is associated with a priority
- 6 privileged modes named `system`, `svc`, `abort`, `fast interrupt`, `interrupt`, and `undefined` mode
- They can all modify the CPSR
- The unprivileged user mode cannot modify the CPSR

The Bible: Atmel doc6175.pdf

The ARM7 processor used in NXT is underlined.

Features

- Incorporates the ARM7TDMI® ARM® Thumb® Processor
  - High-performance 32-bit RISC Architecture
  - High-density 16-bit Instruction Set
  - Leader in MIPS/Watt
  - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
- Internal High-speed Flash
  - 512 Kbytes (AT91SAM7S512) Organized in Two Contiguous Banks of 1024 Pages of 256 Bytes (Dual Plane)
  - 256 kbytes(AT91SAM7S256) Organized in 1024 Pages of 256 Bytes (Single Plane)
  - 128 Kbytes (AT91SAM7S128) Organized in 512 Pages of 256 Bytes (Single Plane)
  - 64 Kbytes (AT91SAM7S64) Organized in 512 Pages of 128 Bytes (Single Plane)
  - 32 Kbytes (AT91SAM7S321/32) Organized in 256 Pages of 128 Bytes (Single Plane)
  - Single Cycle Access at Up to 30 MHz in Worst Case Conditions
  - Prefetch Buffer Optimizing Thumb Instruction Execution at Maximum Speed
  - Page Programming Time: 6 ms, Including Page Auto-erase, Full Erase Time: 15 ms
  - 10,000 Write Cycles, 10-year Data Retention Capability, Sector Lock Capabilities, Flash Security Bit
  - Fast Flash Programming Interface for High Volume Production
- Internal High-speed SRAM, Single-cycle Access at Maximum Speed
  - 64 kbytes (AT91SAM7S512/256)
  - 32 kbytes (AT91SAM7S128)
  - 16 kbytes (AT91SAM7S64)
  - 8 kbytes (AT91SAM7S321/32)
- Memory Controller (MC)
  - Embedded Flash Controller, Abort Status and Misalignment Detection
- Reset Controller (RSTC)

ATMEL®

AT91 ARM Thumb-based Microcontrollers

AT91SAM7S512
AT91SAM7S256
AT91SAM7S128
AT91SAM7S64
AT91SAM7S321
AT91SAM7S32

Periodical Interval Timer

The PIT timer generates the system tick.

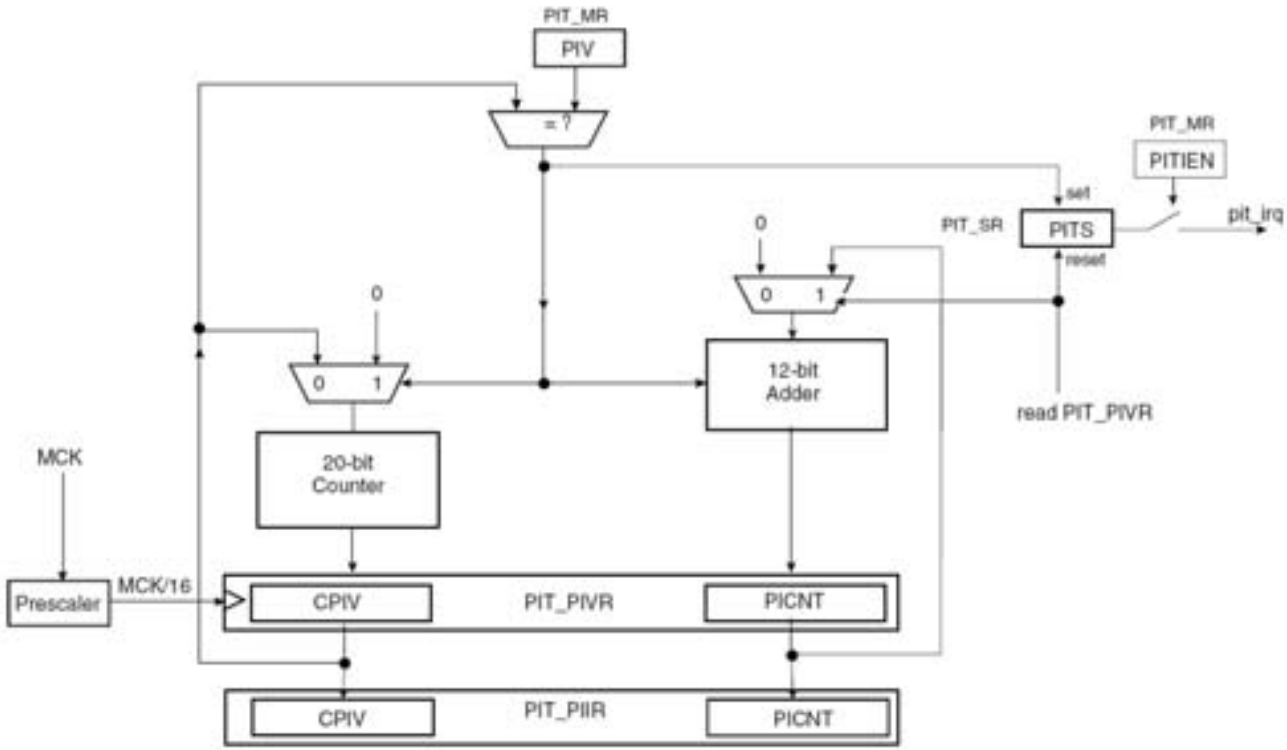
16. Periodic Interval Timer (PIT)

16.1 Overview

The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

16.2 Block Diagram

Figure 16-1. Periodic Interval Timer



Memory Controller

The MC generates the abort exception

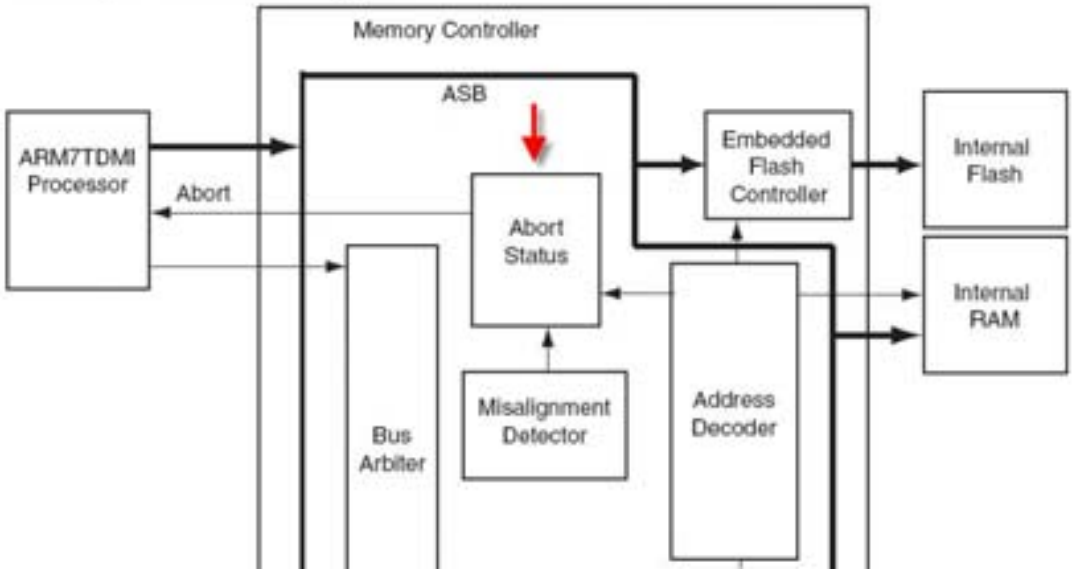
19. Memory Controller (MC)

19.1 Overview

The Memory Controller (MC) manages the ASB bus and controls accesses requested by the masters, typically the ARM7TDMI processor and the Peripheral DMA Controller. It features a simple bus arbiter, an address decoder, an abort status, a misalignment detector and an Embedded Flash Controller.

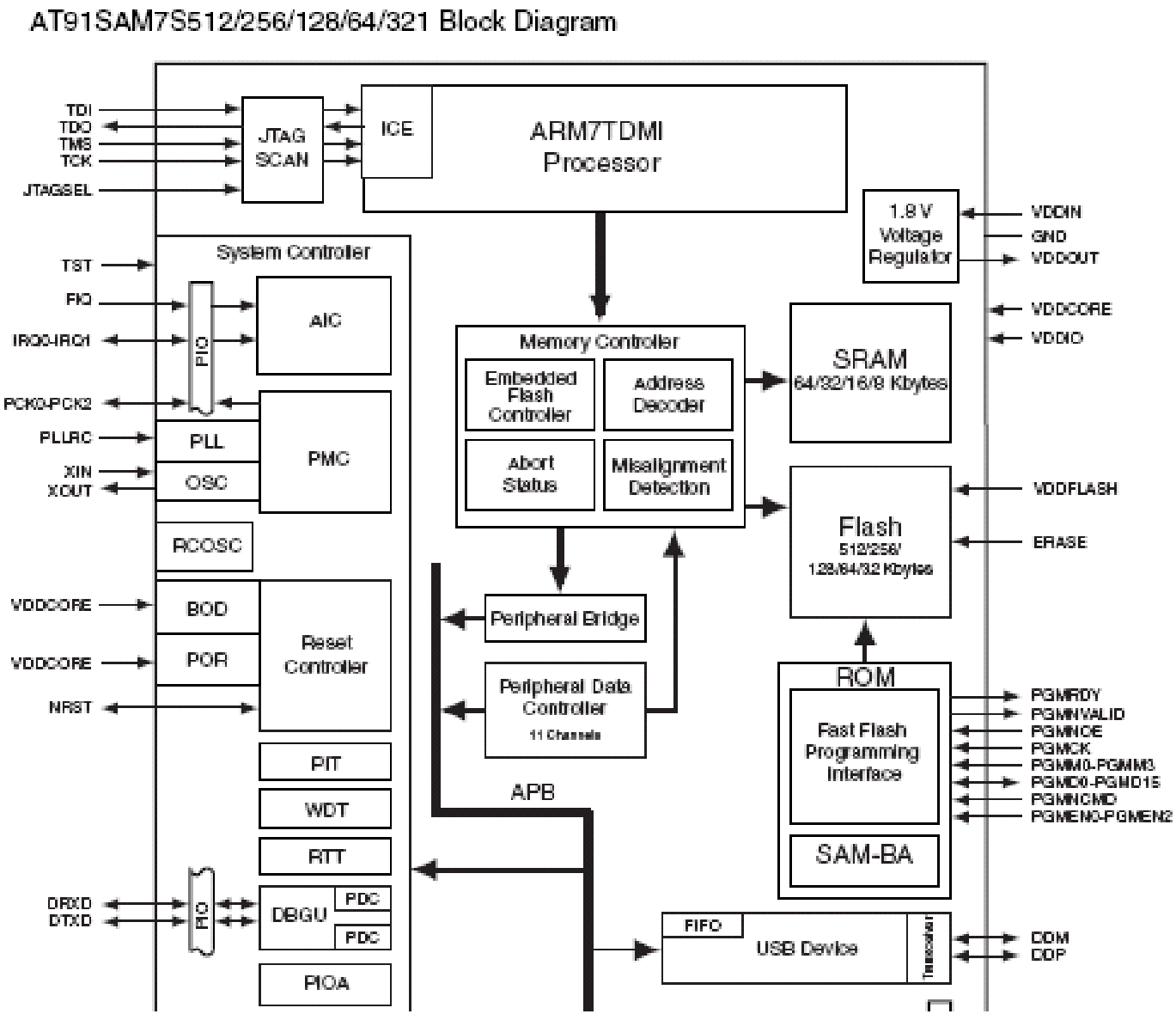
19.2 Block Diagram

Figure 19-1. Memory Controller Block Diagram



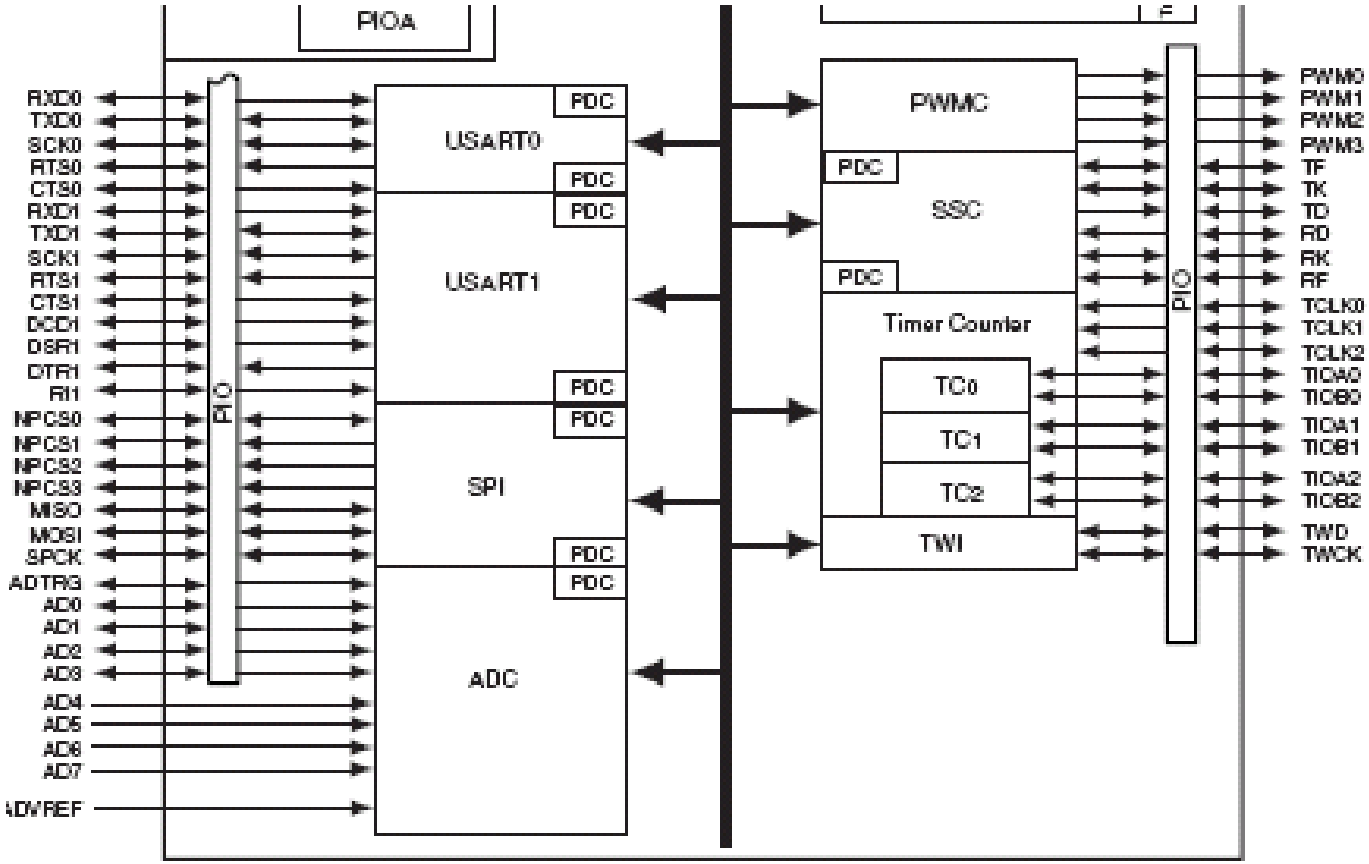
MCU Overview I

SAM-BA, SPI, TWI, AIC, SSC, etc.



MCU Overview II

SAM-BA, SPI, TWI, AIC, SSC, etc.



ARM MCU Features

JTAG, Thumb, ARM

7. Processor and Architecture

7.1 ARM7TDMI Processor

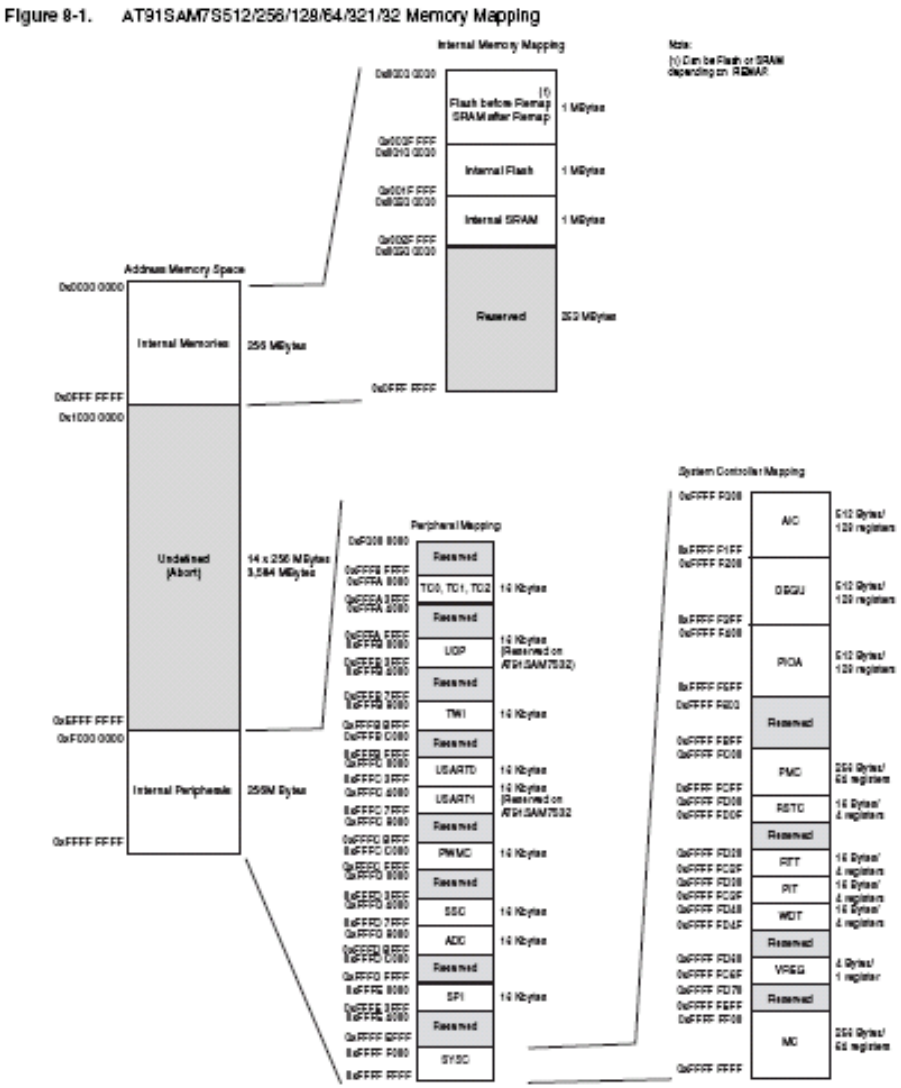
- RISC processor based on ARMv4T Von Neumann architecture
  - Runs at up to 55 MHz, providing 0.9 MIPS/MHz
- Two instruction sets
  - ARM® high-performance 32-bit instruction set
  - Thumb® high code density 16-bit instruction set
- Three-stage pipeline architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)

7.2 Debug and Test Features

- Integrated EmbeddedICE™ (embedded in-circuit emulator)
  - Two watchpoint units
  - Test access port accessible through a JTAG protocol
  - Debug communication channel
- Debug Unit
  - Two-pin UART
  - Debug communication channel interrupt handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on all digital pins

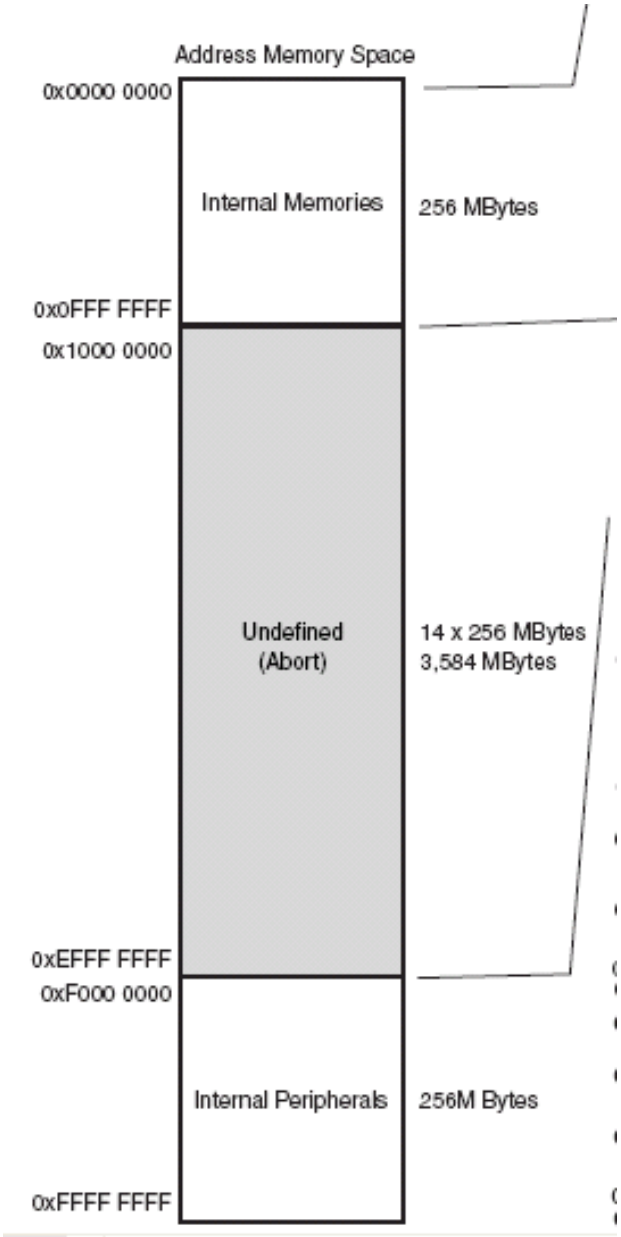
ARM MCU Peripherals and Memory Mappings

The Big (Small...) Picture



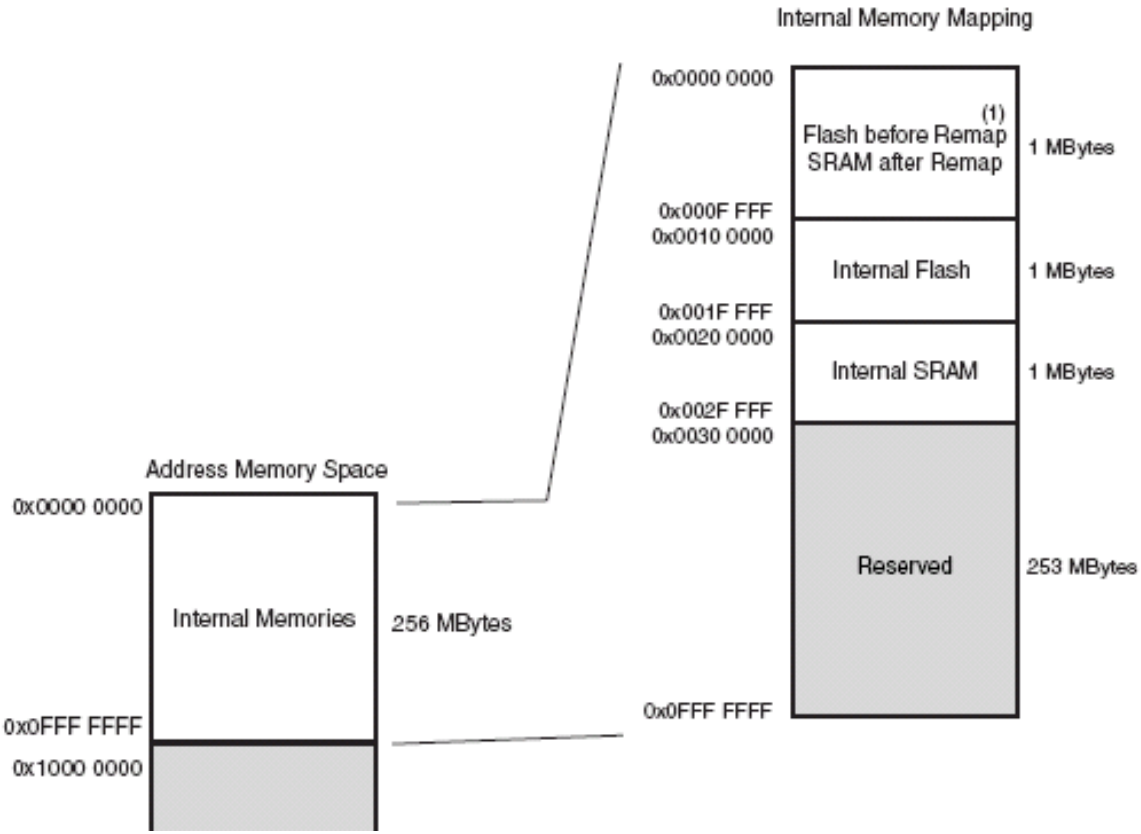
ARM MCU Peripherals and Memory Mappings

Main memory



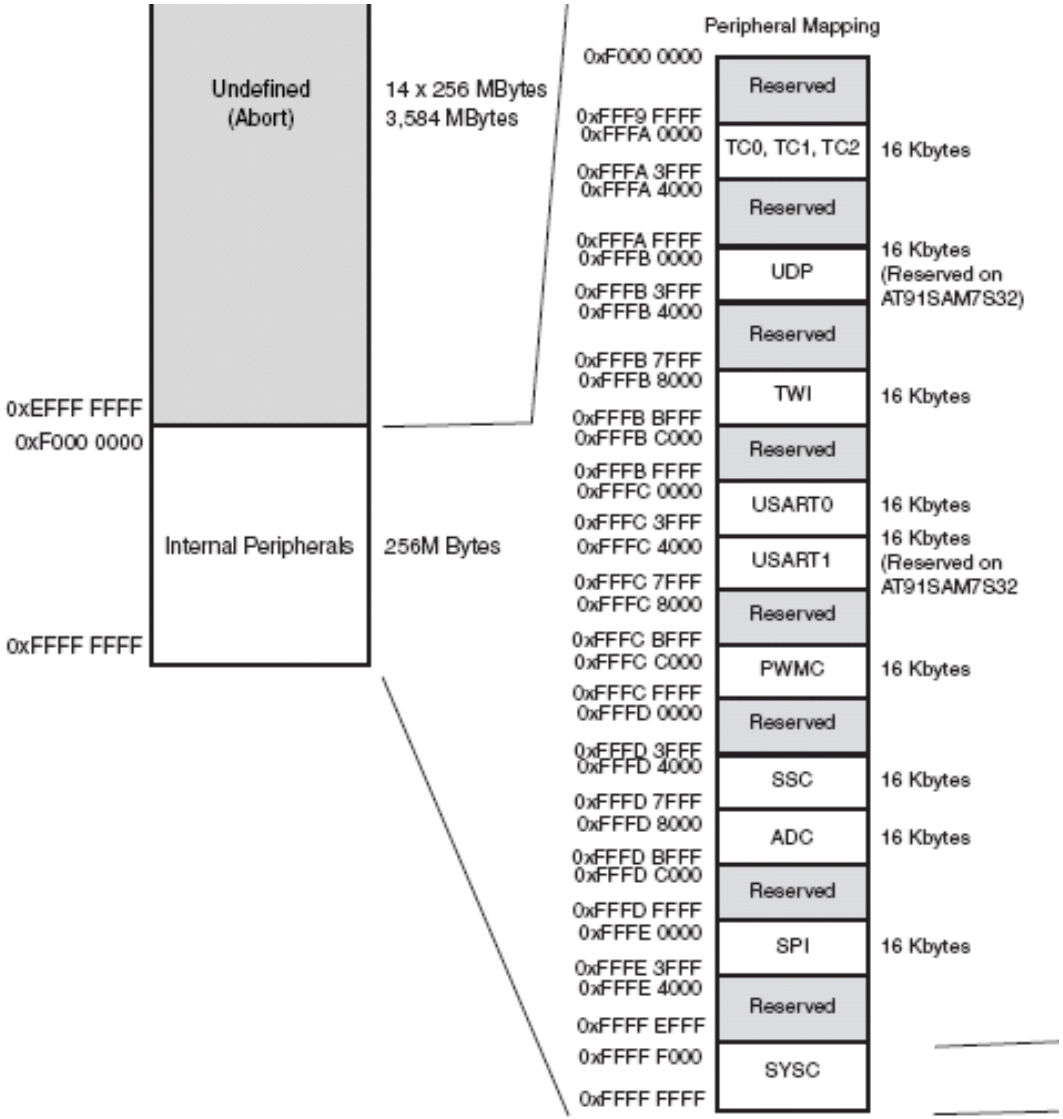
ARM MCU Peripherals and Memory Mappings

Main memory remapping



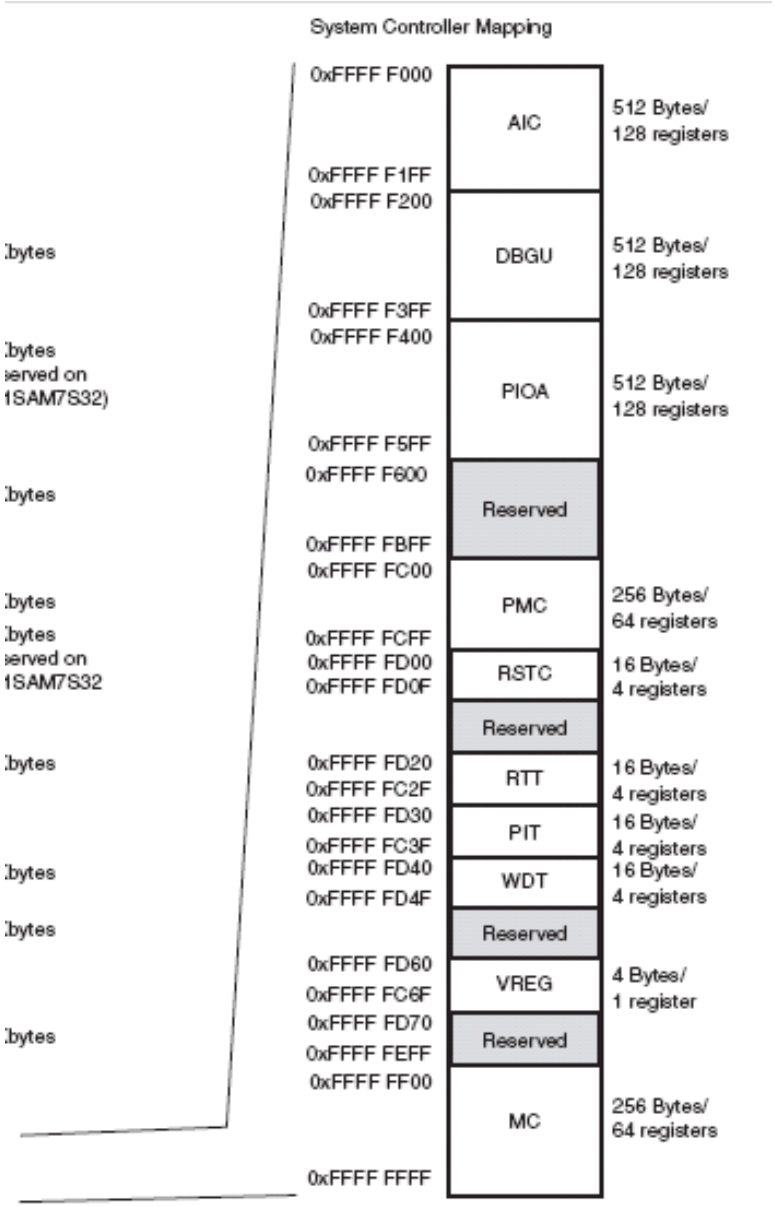
ARM MCU Peripherals and Memory Mappings

Peripherals such as timers



ARM MCU Peripherals and Memory Mappings

System Peripherals such as interrupt controller



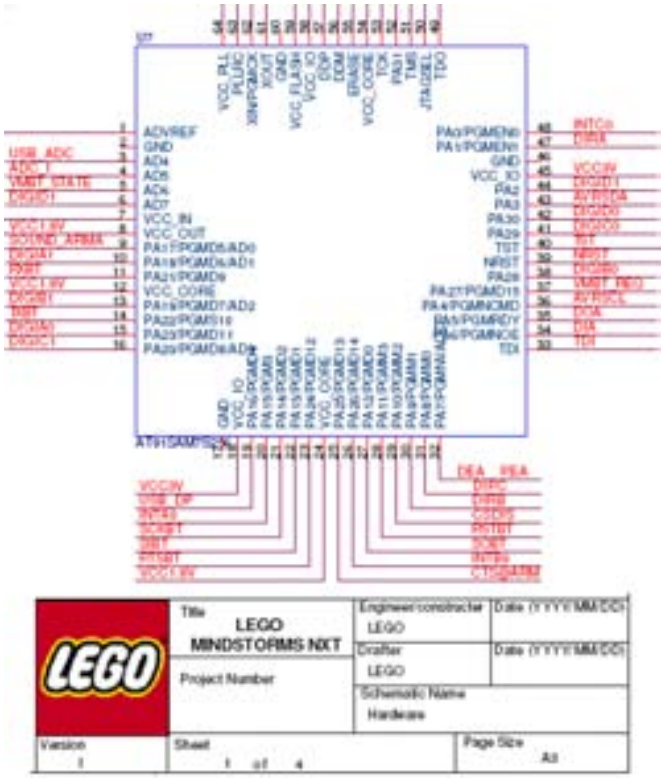
ARM7 Registers

- The CPSR defines the characteristics of the current mode
- It can mask (ie. not allow) interrupts and fast interrupts
- The CPSR includes a flag whether the ARM7 is executing thumb code or arm code
- There is a register file which includes 37 registers
- Each register is 32 bit wide
- Each mode uses 17 registers
- Registers R0-R12 are shared between `usr`, `sys`, `svc`, `abt`, `irq`, and `und` modes
- The `fiq` mode has its own banked version of the R8-R12 registers
- All modes have their own stack pointer and link/return pointer: R13 and R14
- The program counter, `pc`, is placed in R15

ARM and Thumb Instruction Sets

- The ARM instruction set is 32 bit
- Thumb instruction set is 16 bit
- Most of the code in NXT is compiled toward the Thumb instruction set
- Thumb code is more dense than ARM
- Exception and interrupt handlers are compiled to ARM code

Demo ARM compilation.



NXT schematics

ARM Pins

Show PDF with ARM...

PCB	Port/pin	ARM7 pin	ARM7 PIO
DIGIA0	1/5	15	PA23/PGMD11
DIGIA1	1/6	10	PA18/PGMD6/AD1
DIGIB0	2/5	38	PA28
DIGIB1	2/6	13	PA19/PGMD7/AD2
DIGIC0	3/5	41	PA29
DIGIC1	3/6	16	PA20/PGMD8/AD3
DIGID0	4/5	42	PA30
DIGID1	4/6	6	AD7

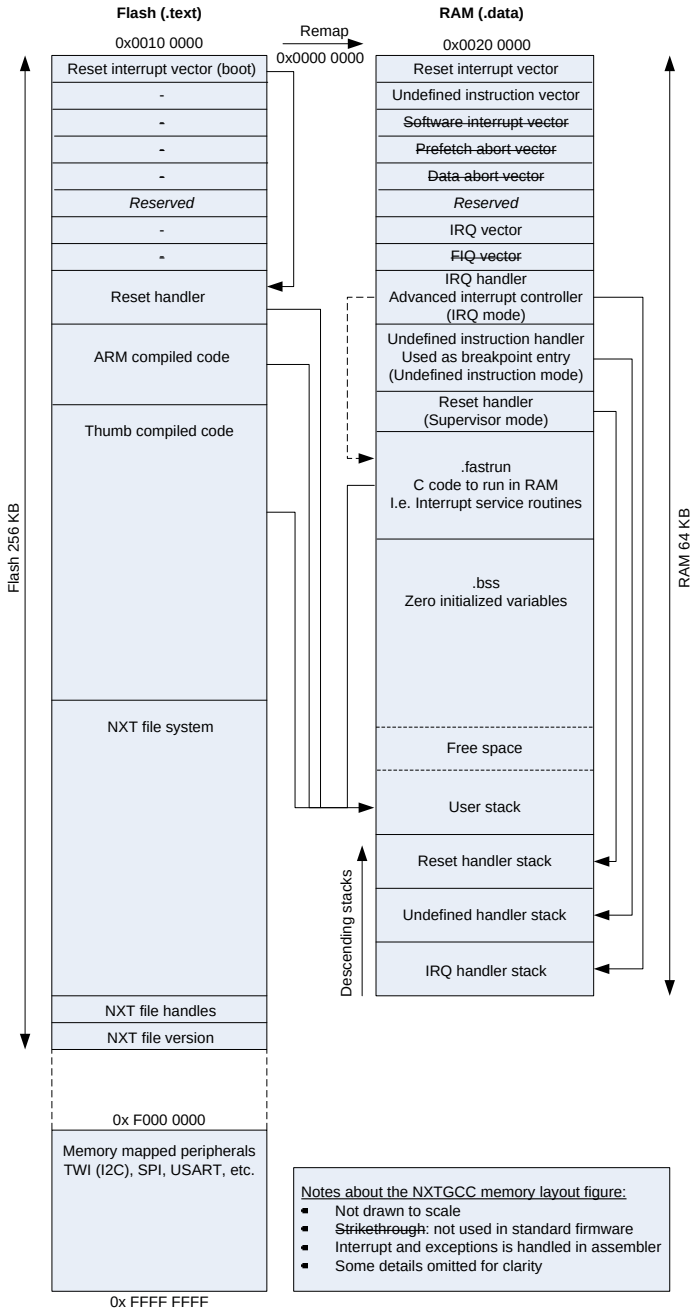
- Port 1 through 4 are mapped to general purpose input output pins
- Most ARM7 pins are multiplexed
- Pin 6 on inout port 1-3 both can serve as GPIO pins and as an analog to digital converter (ADC)
- The *Lowspeed* module use the GPIO pins to *bit-bang* an I2C protocol

Memory Layout and Dynamics

- Flash memory is initially mapped to address 0x0000 0000
- Flash is also accessible at address 0x0010 0000
- NXT immediately branches to the reset handler that is defined in CStartup.S
- It uses a temporary stack in the RAM area
- The reset handler sets up the stacks for the interrupt handler (IRQ), the undefined instruction handler (UND) stack, and the user stack (USR)
- The USR stack is used when the normal code executes
- IRQ stack is obviously used when the IRQ handler is invoked: it branches to the correct interrupt service routine (ISR)
- Controlled by writing the address of that ISR into the advanced interrupt controller (AIC) source vector register (SVR)
- Example: each of the three timers (TC0, TC1, TC2) have an ID, and that ID is used as index in the AIC SVR vector

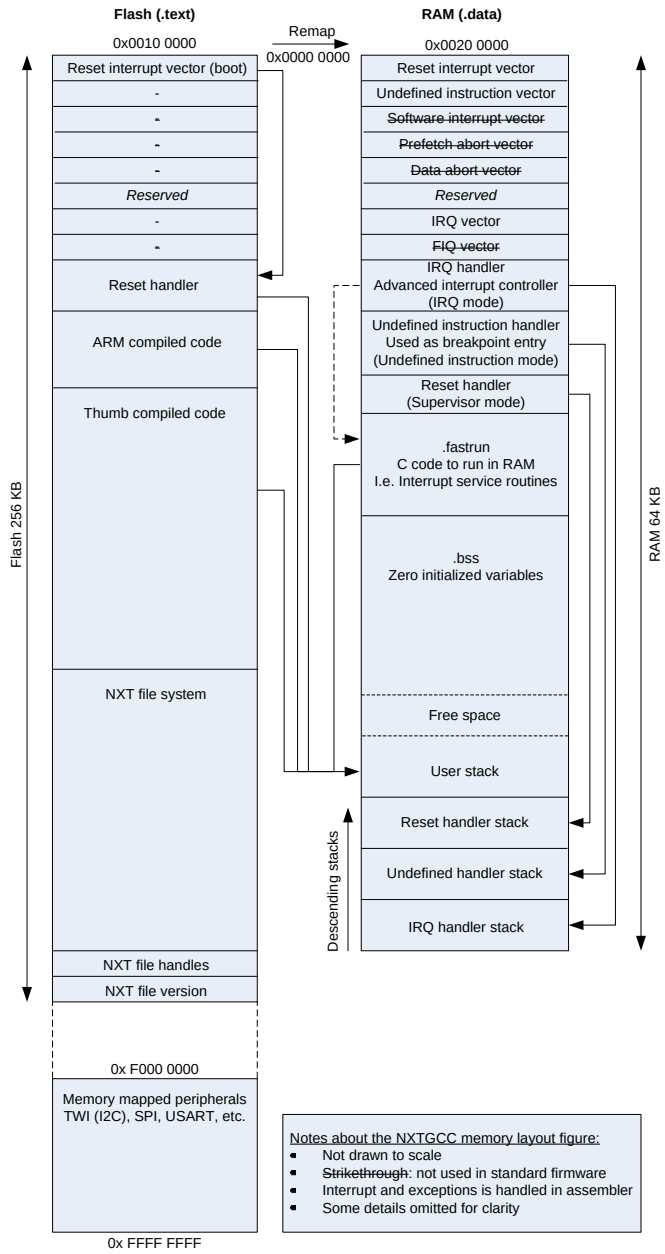
Next: Memory Layout

Memory Layout



Memory Layout

- The stacks on NXT (i.e. ARM7) are fully descending stacks
- Using the instructions STMFD and LDMFD
- Postfix FD depicts that we are working with a full descending stack



TODO: Show code with descending stacks

Undefined Instruction Stack Setup

- Undefined stack setup in Cstartup.S
- The illegal memory access (ABT) stack is set up before the undefined instruction stack

```
/* Set up Undefined Instruction Mode
   and set UND Mode Stack*/
.EQU UND_STACK_TOP, (ABT_STACK_TOP - ABT_STACK_SIZE)
  msr     CPSR_c, #ARM_MODE_UND | I_BIT | F_BIT
  mov     sp, r0  /* Init stack UND */
  sub     r0, r0, #UND_STACK_SIZE
```


Post Boot

- Remapping of address 0x0000 0000 such that it is now the RAM and not the flash that is accessible from address 0x0000 0000
- Main reason for this is that we need to service interrupts even when the NXT flash-based file system is being written to
- Remapping the memory ensures that the interrupt vector table is access in RAM
- All interrupt service routines are linked into the .fastrun segment

Show segments in file.

NXT Running

- Mostly taking place in the Thumb compiled code
- Interrupt is asserted from one of the ARM7 peripheral sources then that interrupt service routine takes over
- Most peripherals in NXT are associated with a software module

Modules

- Each module is built from an header file, a C file, and .iom file
- Layered in a hardware physical layer (HPL), and a hardware abstraction layer (HAL)
- HAL files start with the letter c followed by an underscore, while the HPL files start with the letter d
- Show a module...

NXT Software Modules and ARM7 HW Peripherals

Name	ARM Peripherals	Note
Scheduler	-	Main control
Button	AVR via I2C/TWI	4 buttons
Display	SPI	LCD display
CMD	-	Virtual machine
Input	GPIO, PWM, AD	4 input ports
IOCtrl	I2C, AIC	AVR communication
Loader	MC	File management
Lowspeed	GPIO, PWM	I2C emulation
Output	AVR via I2C	Motor control
Sound	SSC	Loadspeaker control
UI	-	Menu system
Timer	PIT	Operating system tick
Bluetooth	SPI, GPIO	Communication control
Communication	UDP, UART, SPI	USB, RS485, and BT

Abbreviations: I2C:Inter-integrated circuit, TWI:Two-wire interface, PWM:Pulse width modulation, AIC:Advanced interrupt controller, MC:Memory controller, SSC:Synchronious serial controller, PIT:Periodic interval timer, SPI:Serial peripheral interface, UDP:USB device port, and UART:Universal asynchronous receiver tranceiver.

NXT Modules

Scheduler This is the main scheduler: m_sched. It works in a round robin fashion with a 1 ms system *tick*. It will call each of the modules listed below using function pointers. There is an initialization phase in which each module's cInit method is called. Then the system starts calling the cCtrl methods. Termination is done via calls to CExit.

Button It handles the four buttons on top of NXT. Principle: It uses the AD converter peripheral Button 0 is read as? Button 1-3 is read as an AD converted value.

CMD This is the Lego virtual machine. It reads the executable files stored in flash and creates a RAM space for runtime information. The rxe file contains bytecode instructions, bytecode scheduling, and run-time data.

Comm It takes care of the BT, USB and the Highspeed port. Peripherals used: SPI to BT, USB, and USART in RS485 mode.

NXT Modules

Display It is code for accessing the pixels of the 100 x 64 pixel display. It is possible to both set pixels, write characters, and strings (it is a matter of abstraction). There is a font included with the NXT. There are eight text lines on the display. It accesses the serial peripheral interface (SPI) hardware peripheral. The SPI works by writing data to a slave (the display in this case) over the master out slave in (MOSI) line. It receives data from the slave over the master in slave out line (MISO). The data flow is synchronous and is regulated by a serial clock (SPCK) line. The display receives a command from the ARM7 specifying which line is to be written, and then it receives the actual data for that line.

Input The input sensors are plugged into the 4 input ports on NXT. NXT will know which type of sensor is plugged into a port. It can be a temperature sensor, sound, lowspeed, and so on. The lowspeed sensor is an I2C/TWI compliant sensor.

NXT Modules

IOCtrl This module controls the booting and power down sequences of NXT. It will write to the AVR what the power should be and what the PWM for the motors should be. The IOCtrl module uses the I2C(TWI) hardware peripheral to relay this information to the AVR. It also uses the advanced interrupt controller (AIC), by having it call an I2C handler routine.

Loader File management is achieved by the loader module. It reads, writes, creates, and deletes files in the NXT flash. It resides in the upper part of the flash memory starting from the value of the memory address STARTOFUSERFLASH in d_loader.h. It uses the memory controller (MC) peripheral module of the ARM7. No read while writing: the methods that perform erase and write of flash pages are placed in RAM. This is achieved by marking the methods with __ramfunc, which tags the method to belong to a section called .fastrun. The .fastrun section are placed in the DATA=RAM area.

NXT Modules

Lowspeed An I2C compliant device can be connected to one of the four input ports. The lowspeed module handles the communication and recognizes the connected sensor as either the standard ultrasonic device, or a custom device. It uses an IRQ handler that is invoked via the AIC hardware peripheral: It utilizes the pulse width modulation controller (PWM) peripheral to generate the clock.

Output Motors are connected to NXT through the three output ports: A, B, and C. The module communicates the motor output to the AVR via the I2C bridge. It is managed by struct, which is passed to the AVR from the ARM7 that controls this communication line. The information that controls the motors are an output mode and a PWM frequency for each motor. Thus, the output module make use of the hardware peripherals TC0, TC1, TC2, to monitor motor C, A, and B respectively. Interrupt handlers: the tacho count bookkeeping.

NXT Modules

Sound A loudspeaker with 8 bit resolution is controlled by the sound module. It plays sounds which are files residing in the NXT file system. Each time cCtrl method of the sound module is called, then it either plays a sound, continues to play a sound, or closes the sound file that has been played. The module utilizes the synchronous serial controller (SSC) hardware peripheral. With the SCC providing the AC signal ((SOUND_ARMA from PA17) to the SPY0030A audio driver, NXT can play sounds. It uses the peripheral DMA controller (PDC) to handle the transmission of the bitstream to the speaker.

UI The module manages the menu system, selected files, icons, animations, etc. on NXT. It responds when an user press on of the buttons on NXT.

Timer NXT is programmed around a system tick of 1 ms. This property is supported by the timer module that use the periodic interval timer (PIT) hardware peripheral to maintain a 1 ms counter.

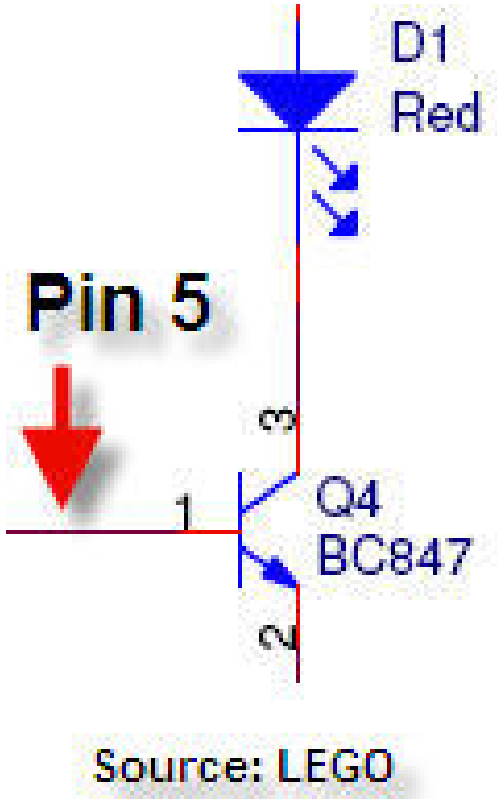
Debugging Overview

- Easiest way to debug NXT is to use the light sensor
- We can find the memory location that controls the GPIO pin
- Other choices is to use a JTAG debugger

Light Sensor Port

```
//Turn on an input port
#define ON(port)
{
  *AT91C_PIOA_SODR = PORT_TO_PIN(port);
}
```

Turn on a light sensor on an input port for low-level debugging...
Show section *15.4.4 Output Control* in ARM PDF...



Light sensor schematics: Show how to turn on the light sensor in code

Display Debugging

Example (from c_cmd.c):

```
  {// LCD DEBUG START
  cnt++; // a static counter
  UBYTE DebugString[50];
  sprintf(DebugString,"Debug info: %d", cnt);
  if(cnt==1)
  {
    cDebugString(&DebugString[0]);
  }
}//LCD DEBUG END
```

Showing a debug string in the display

GDB Debugging

- We may think (initially) that we have stopped the processor
- A break point can be inserted into the code using the GCC inline assemble syntax

```
/*
 *  Trigger a breakpoint exception.
 */
__ramfunc void bsp_breakpoint(void) {
  //asm volatile (".word 0xE7FFDEFE");
  NXTGCCBREAK;
}
```


GDB

- GDB is the debugging system that is connected to GCC
- GDB system offers a remote debugging system
- USB driver to use is the Fantom SDK
- Fantom C++ files, there is a section with ëxtern C"
- Read and write methods for arbitrary bytes
- Allows for an implementation of the GDB remote debugging protocol
- Debugger is arm-elf-gdb

GDB Remote Protocol

- It includes a one-byte checksum
- Comes after the # in the packet
- Problem is that the Fantom DLL does not like if we try to read from the USB if there is not something to read
- We read two characters (checksum)
- NXTGCC Fantom GDB wrapper (discussed later) writes and especially reads one character at a time to/from NXT, even though the USB endpoints are in bulk mode

Show GDB files here...

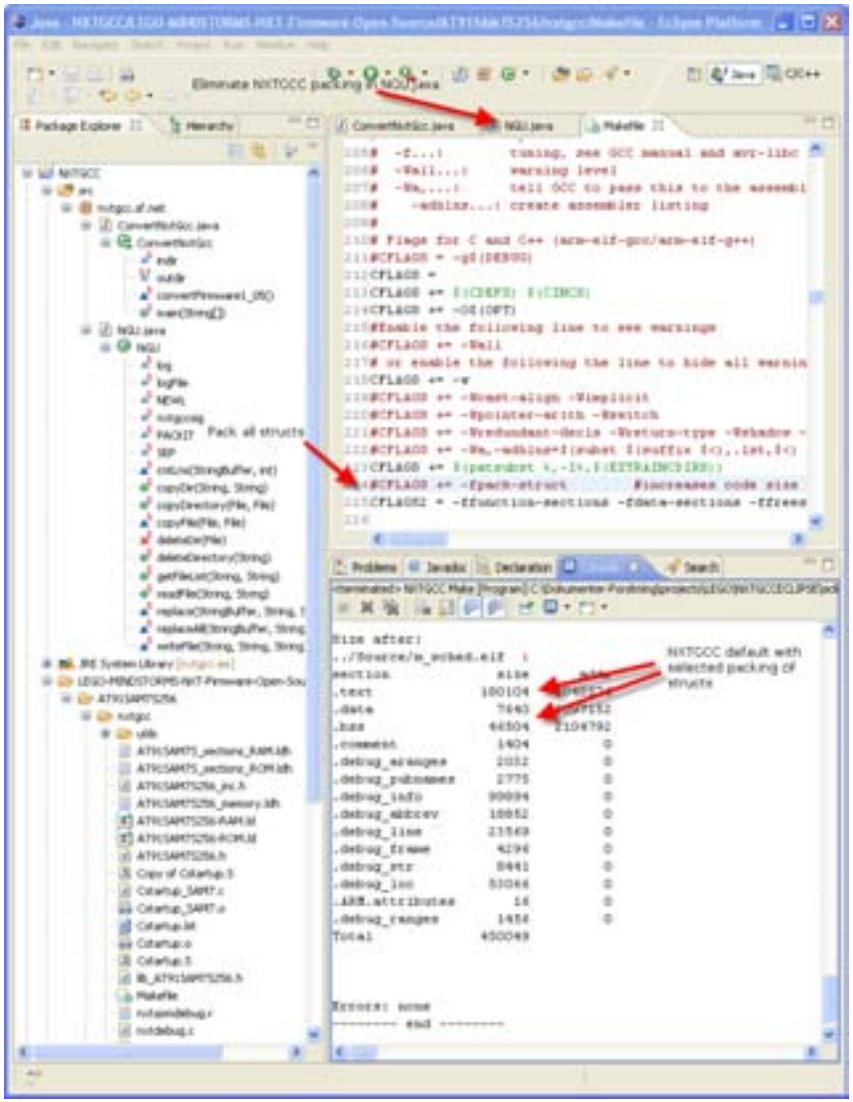
Hardware Debugging

- It is possible to create one using a 1.27 mm pitch row
- Each of the eight pins from J17 can be connected to a standard 20 port JTAG like this: $1 \rightarrow 9$, $2 \rightarrow 7$, $3 \rightarrow 13$, $4 \rightarrow 15$, $5 \rightarrow 5$, $6 \rightarrow 4$, $7 \rightarrow$ *not connected*, and $8 \rightarrow 1$.
- Use Segger, IAR or some open source project like OPENOCD

Using a JLink to flash an ARM7 Eval. board, while warming up for NXT to arrive in the summer of 2006.

Compiler Performance

- Eclipse is one possibility as an programming environment
- Output from compiling the firmware has been shown
- Size and memory addresses of the different sections are displayed after a successful build



Reading the compilation statistics in Eclipse

Comparison

Comparison of GCC and IAR Code Size

Compiler	Packing structs	Size of .text (bytes)	Size of .data (bytes)
GCC	All	202052	54548
NXTGCC	Selected	180104	54144
GCC	None	162136	54228
IAR	(speed)	127552	49831
IAR	(size)	122440	49763

GCC compiler (arm-elf-gcc version 4.2.2) that NXTGCC uses, creates a larger code footprint than the IAR compiler does. For the flash we can compare the GCC compiler's approx. 175 KB to the IAR compiler's approx. 120 KB. Regarding RAM, the picture is approx. 53 KB compared to approx. 49 KB in favor of IAR.

Code Speed

Comparison of GCC and IAR Code Speed

Compiler	Speed # loops
NXTGCC (selected structs)	615
IAR (size optimized)	640

Benchmark code: Steven Hassenplug

Documentation

Lego, Atmel, and the NXTGCC project provide comprehensive information about NXT. This includes hardware schematics, explanation of important protocols, and documentation for programming NXT:

- **Lego:** Hardware Developer Kit
- **Lego:** Software Developer Kit
- **Atmel:** AT91SAM7S256 aka ARM7 hardware description
- **NXTGCC** `http://nxtgcc.sourceforge.net`