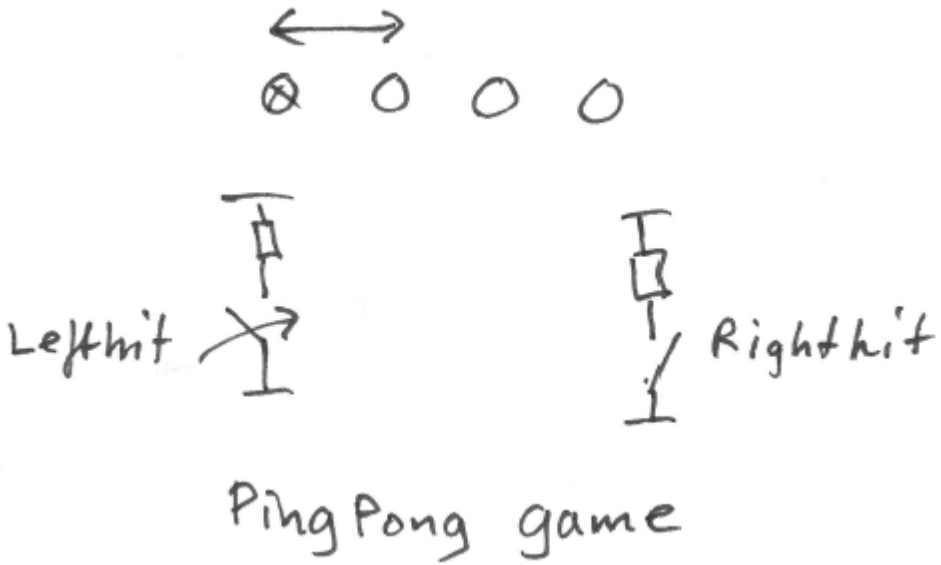
Föreläsning 4 IS1300 Inbyggda system

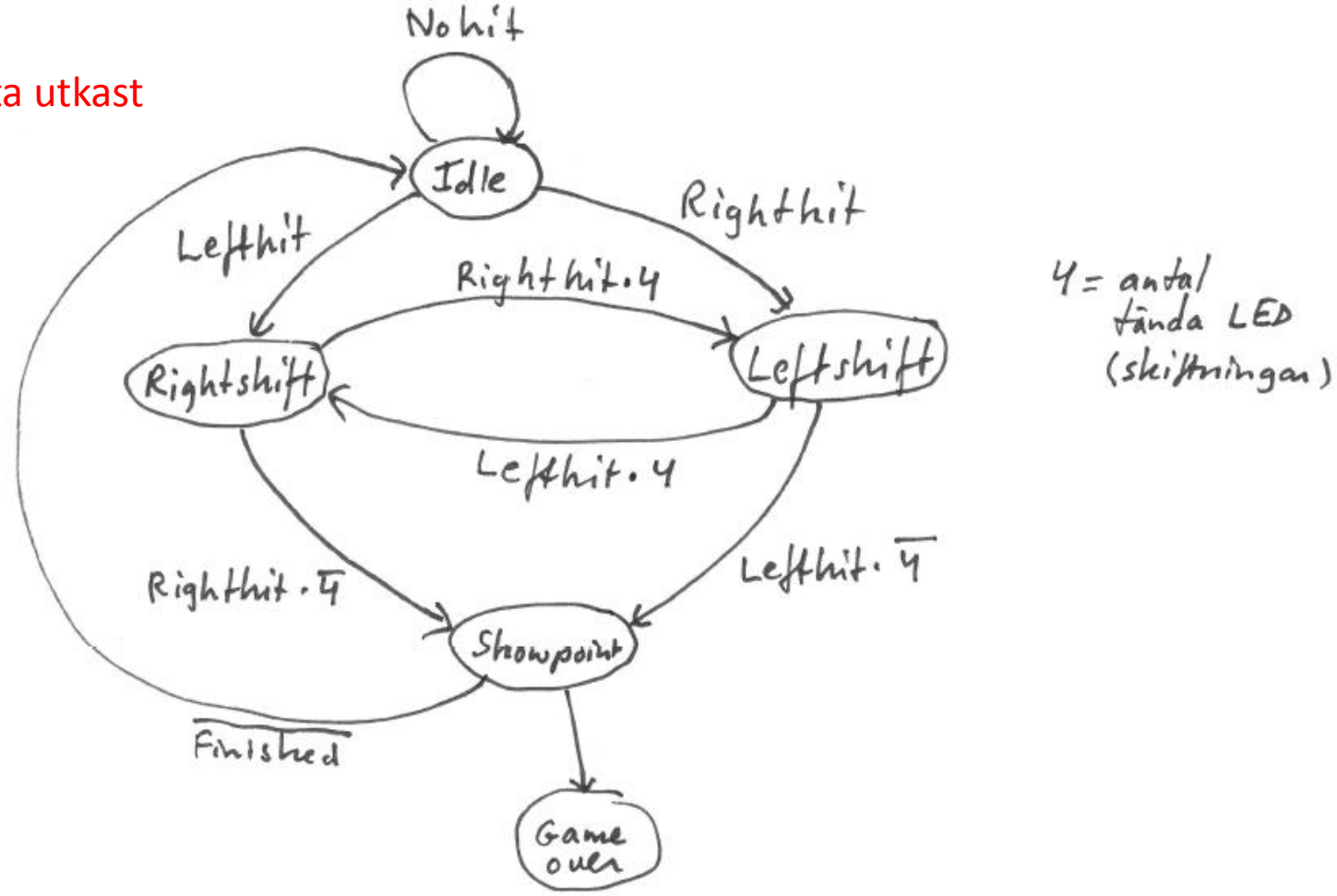
- Programutveckling Exempel PingPong
  - Idé
  - Tillståndsdiagram – State machine
  - Skapa projekt
- Testning av programvara
- Peripheral Library till STM32
- Programmeringsuppgiften RS232

Idé

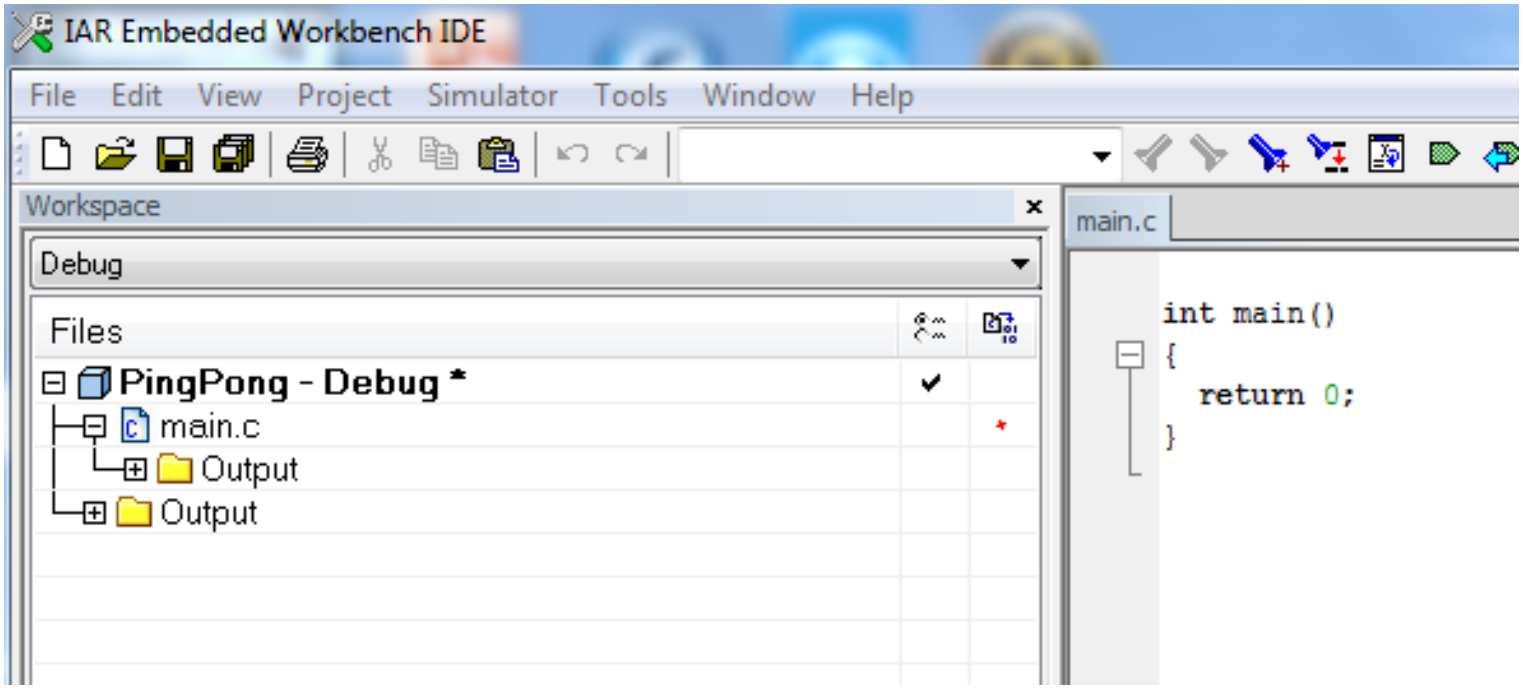


Tillståndsdiagram

Ett första utkast



Skapar projekt i Embedded workbench



Referens: Getting started with embedded workbench, page 37 ...

Skriv kod för State machine

- Skriv en programstruktur
- Använd gärna papper och penna!
- Testa av delar av programmet innan du går vidare
- Bygg programmet med funktioner så att huvudprogrammet blir kort och lätt att följa
- Skriv bra kommentarer

Visa exempel version 0.1 State machine ≈ klar

Hur ska vi testa programmet?

- Traditionellt DLP = Debug Later Programming
- TDD = Test Driven Development
  1. Add a small test
  2. Run all the tests and see how the new one fail
  3. Make the small changes to pass the test
  4. Run all the tests and see the new one pass
  5. Refactor to remove duplication and improve expressiveness

Refactor: Changing structure without changing its behavior. Cleaning!

Bok: Grenning, Test-Driven Development for Embedded C

Ref: http://en.wikipedia.org/wiki/Test-driven_development

Test av C-kod

Även om vi använder DLP bör vi fundera ut i förväg vilka tester vi skall genomföra för att säkra funktionen hos koden

Mitt exempel

- Programmet går nu att köra med debugger i läge simulering
- Ingen hårdvara ansluten
- Jag kan testa tillståndsmaskinen

Lägg till Peripheral library

- Här finns det olika vägar att gå för att skapa ett eget projekt med rätt namn
  - Anvisningar enligt IAR guide och helpfilen stm32f10x_stdperiph_lib_um.chm
  - Använd mallar från Peripheral library
  - Utgå från ett exempelprojekt till IAR-kortet och lägg till de filer som behövs och kopiera filstruktur och projektinställningar

stm32f10x_stdperiph_lib_um.chm

- Demo av hur man hittar i helpfilen
- Den krets som sitter på IAR-kortet STM32F103ZE har 512 kByte flashminne och är en High density device
- Använd helpfilen för att förstå hur perifera portar konfigureras och API används
- stm32f10x_conf.h bestämmer vilka h-filer som läses in (kommentera bort de som inte behövs)
- De exempel för EVAL-kort som finns beskrivna i help-filen gäller ST-kort, ej IAR-kortet

Versionshantering

- Under utveckling så ändras filer hela tiden och inte alltid till det bättre
- Vi behöver ett system för att hålla ordning på de ändringar som görs
- Nödvändigt när flera jobbar i samma projekt
- I versionshanteringssystemet (VCS = Version Control System) kan du checka in och checka ut kod från en server

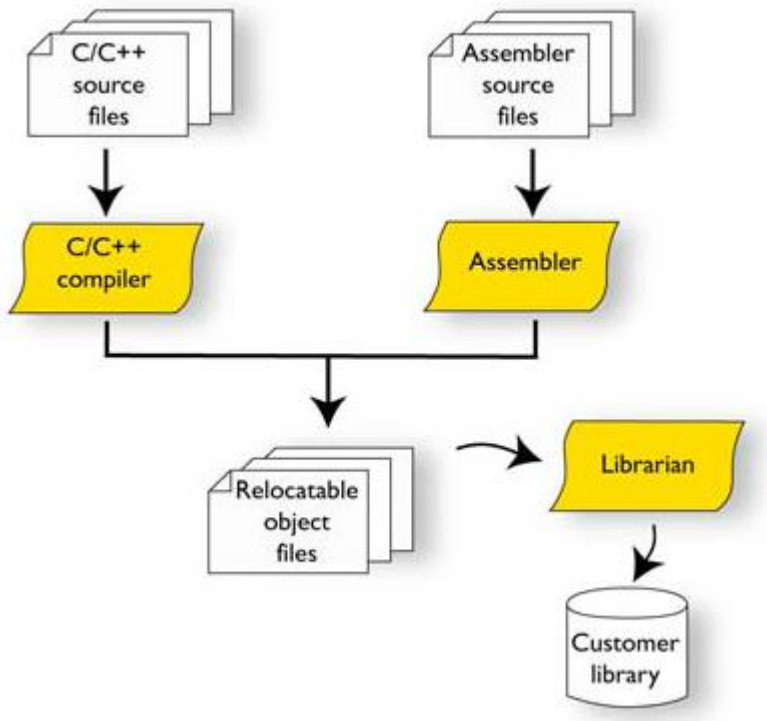
Ref: http://en.wikipedia.org/wiki/Revision_control

Kompilering och länkning

- För att få ett körbart program måste programmet
  - Kompileras
  - Länkas
  - Läggas in i minnet på rätt ställe
  - Initieras

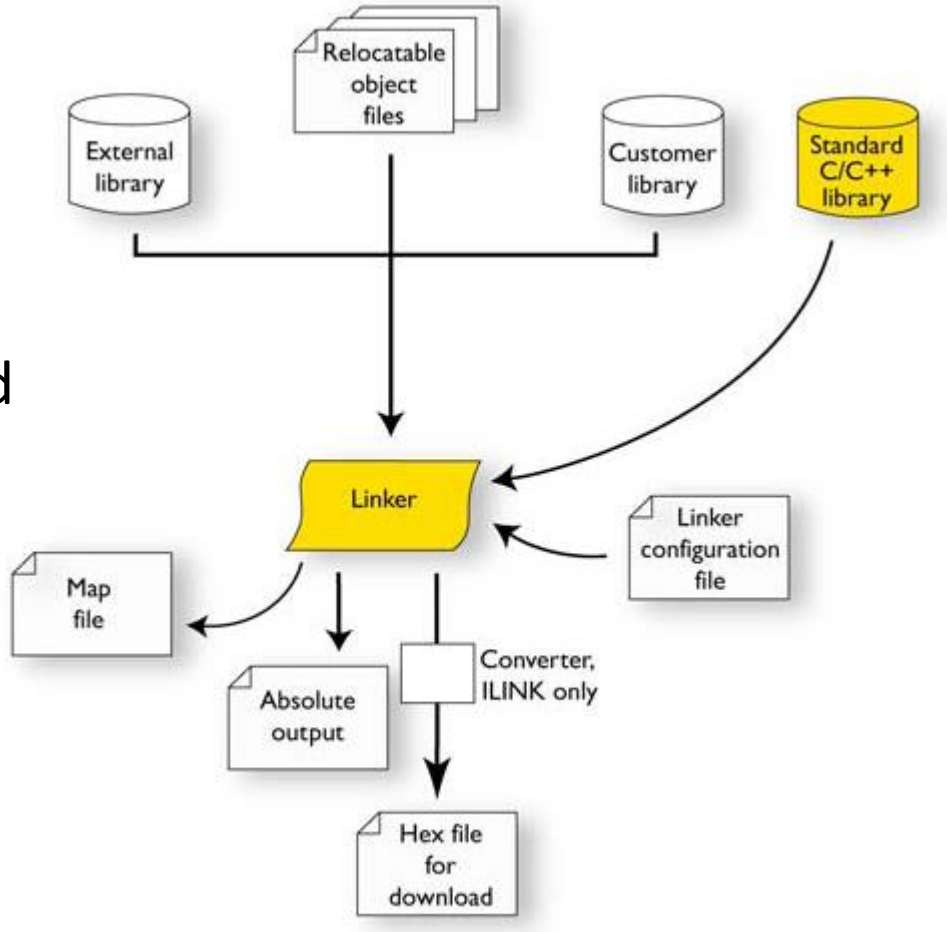
Compiler and assembler

From source code to object code

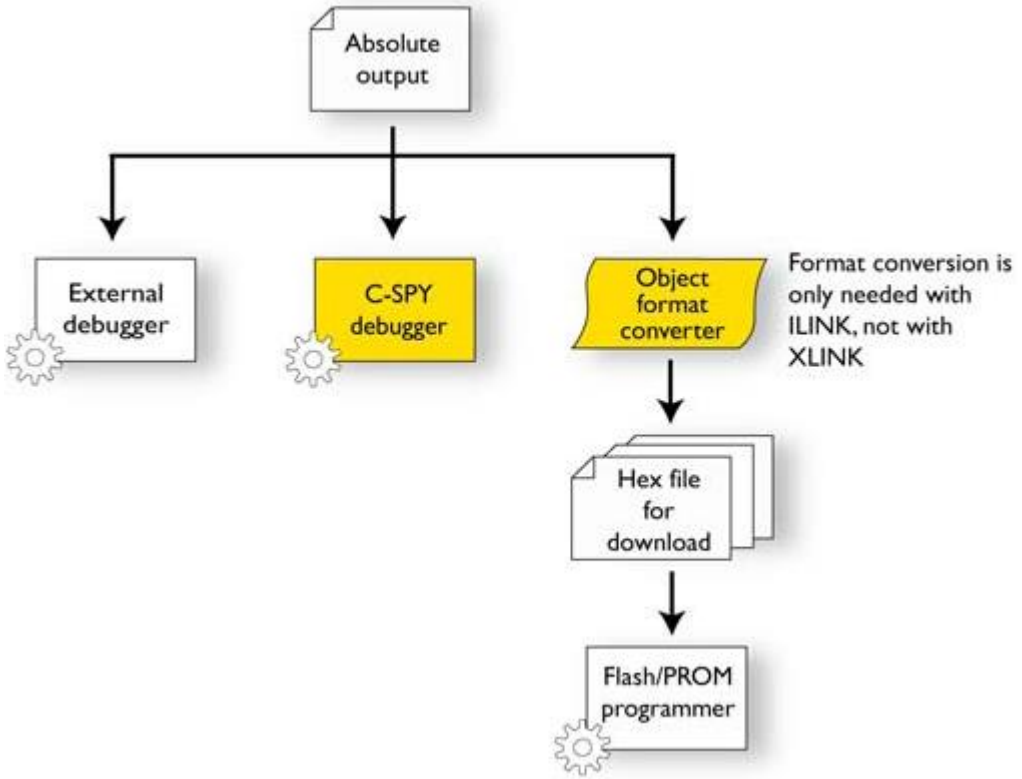


Linker

Linking object files and library files to an executable HEX file for download



Linker output



Initialization phase after reset

Initialization is executed when an application is started (the CPU is reset) but before the main function is entered

- Hardware initialization, for example initializing the stack pointer

The hardware initialization is typically performed by the system startup code Cstartup.

- Software C/C++ system initialization

Typically, this includes making sure that every global (statically linked) C/C++ object receives its proper initialization value before the main function is called.

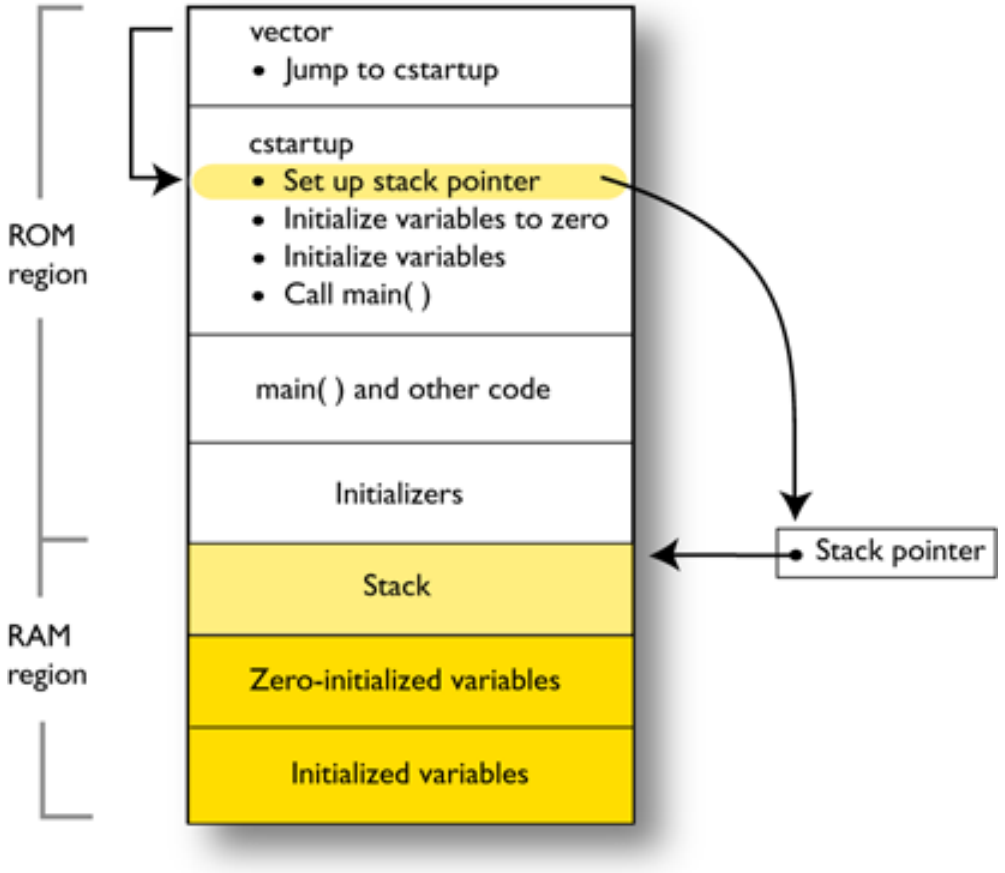
- Application initialization

For a bare-bone application, it can include setting up various interrupts, initializing communication, initializing devices, etc.

For a ROM/flash-based system, constants and functions are already placed in ROM. All symbols placed in RAM must be initialized before the main function is called. The linker has already divided the available RAM into different areas for variables, stack, heap, etc.

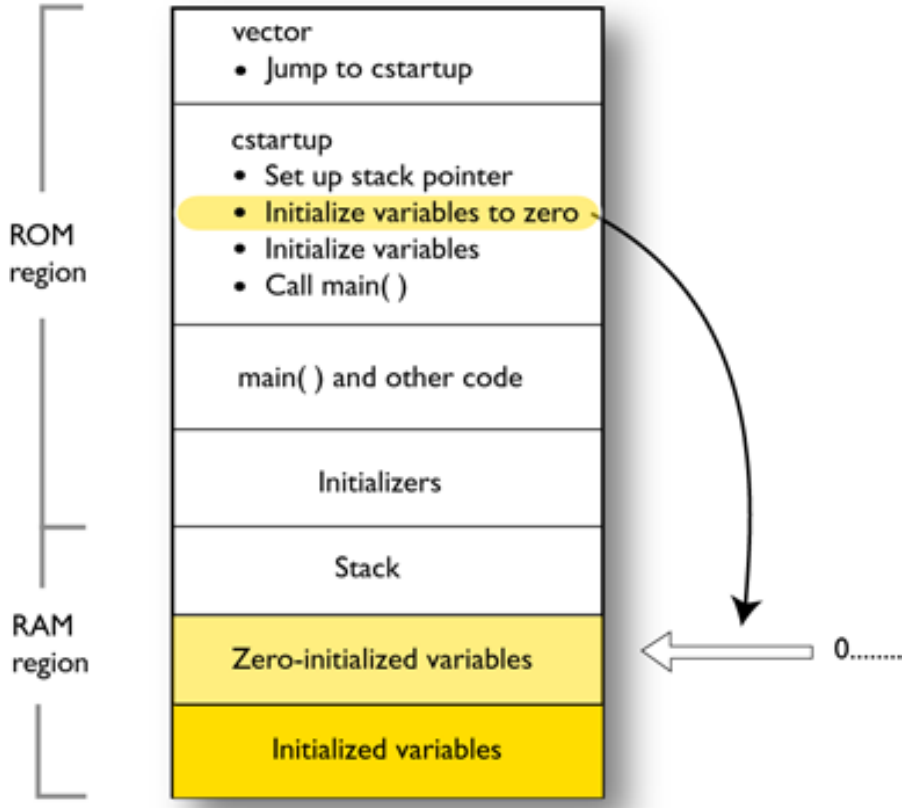
1 Initialize stack pointer

When an application is started, the system startup code first performs Hardware initialization, such as initialization of the stack pointer to point at either the start or the end—depending on your microcontroller—
of the predefined stack area.



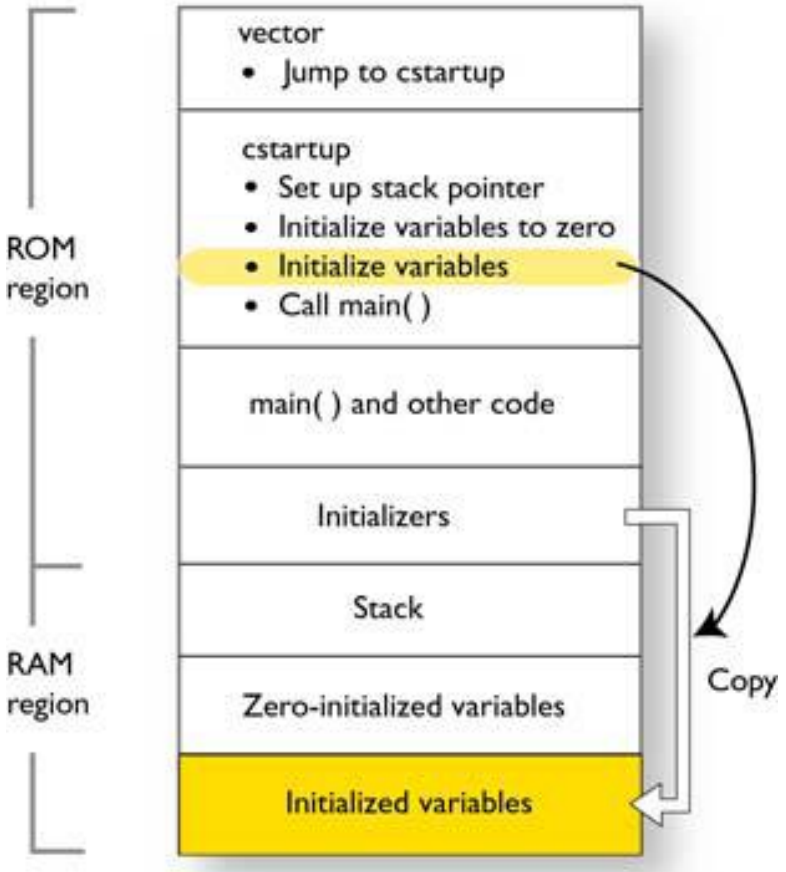
2 Zero-initialized variables

Then, memories that should be zero-initialized are cleared, in other words, filled with zeros. Typically, this is data referred to as zero-initialized data; variables declared as,
for example, int i = 0;



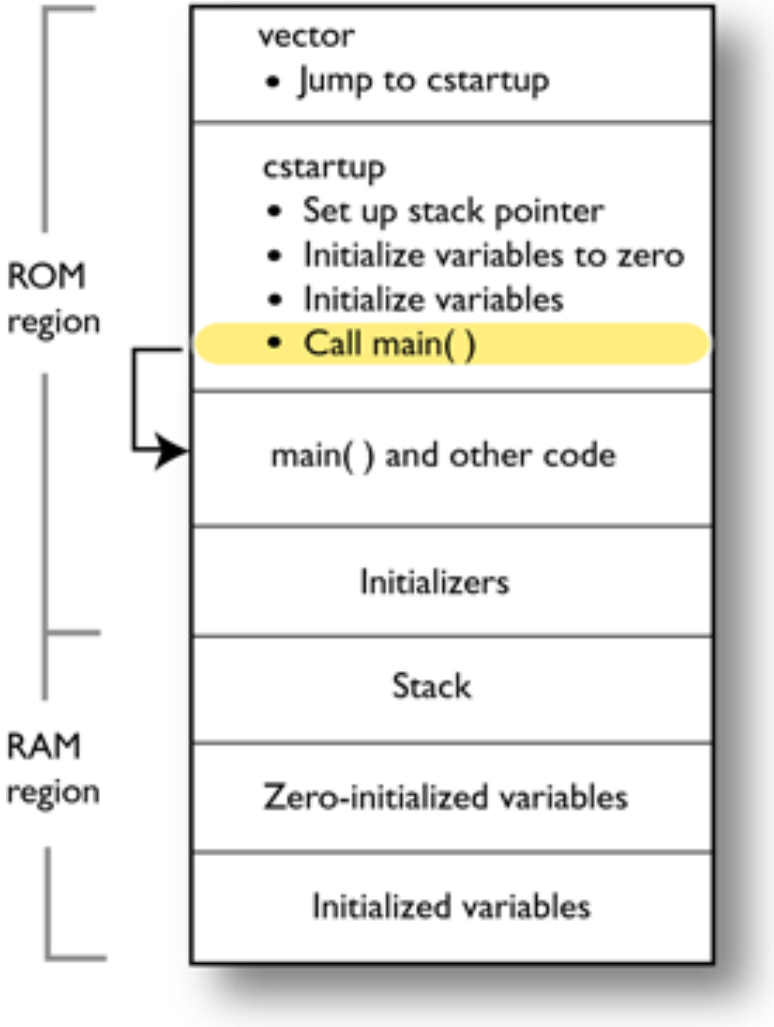
3 Initialize data

For *initialized data*, data declared with a non-zero value, like int i = 6;, the initializers are copied from ROM to RAM.



4 Finally, main is called

Finally, the main function is called.



Programmeringsuppgiften

- Seriell kommunikation via RS232 från PC
- Seriella tecken som överförs skall tolkas och aktivera något på kortet samt returnera kvittens eller värden
- Redovisning sker vecka 10 med skriftlig rapport och muntlig examination. Var beredd att kunna förklara hur du har gått tillväga för att utveckla programmet.

Seriell port - RS232

- Seriell port

  http://en.wikipedia.org/wiki/Serial_port

- RS232

  http://en.wikipedia.org/wiki/RS-232

I vår microcontroller används USART för asynkron seriell kommunikation

http://en.wikipedia.org/wiki/USART