

# Programmering av inbyggda system 2014/2015

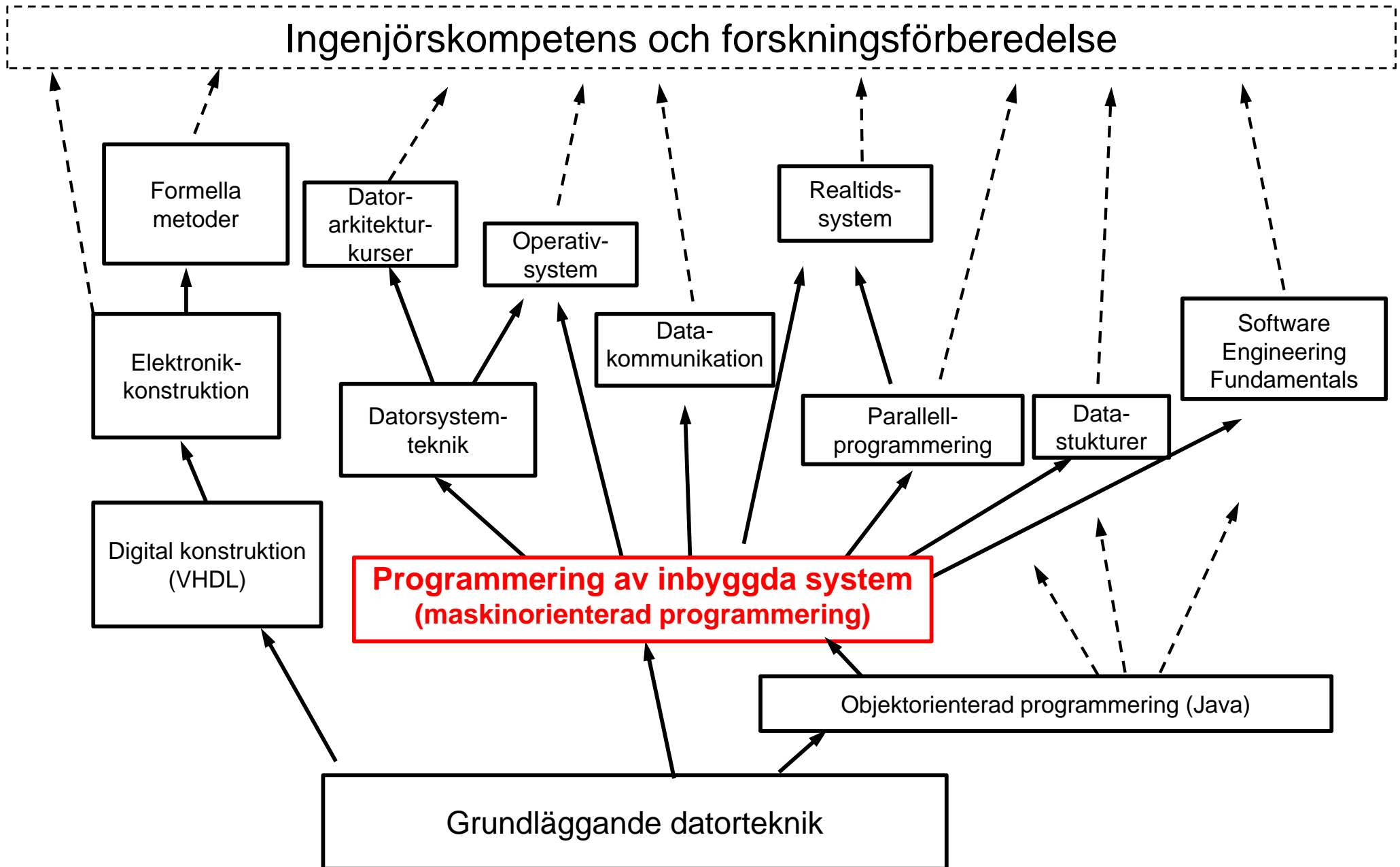
Kursintroduktion  
Roger Johansson

Ur innehållet:

Syften, målsättningar, kurslitteratur och genomförande  
Översikt av laborationer

# Syften och målsättningar

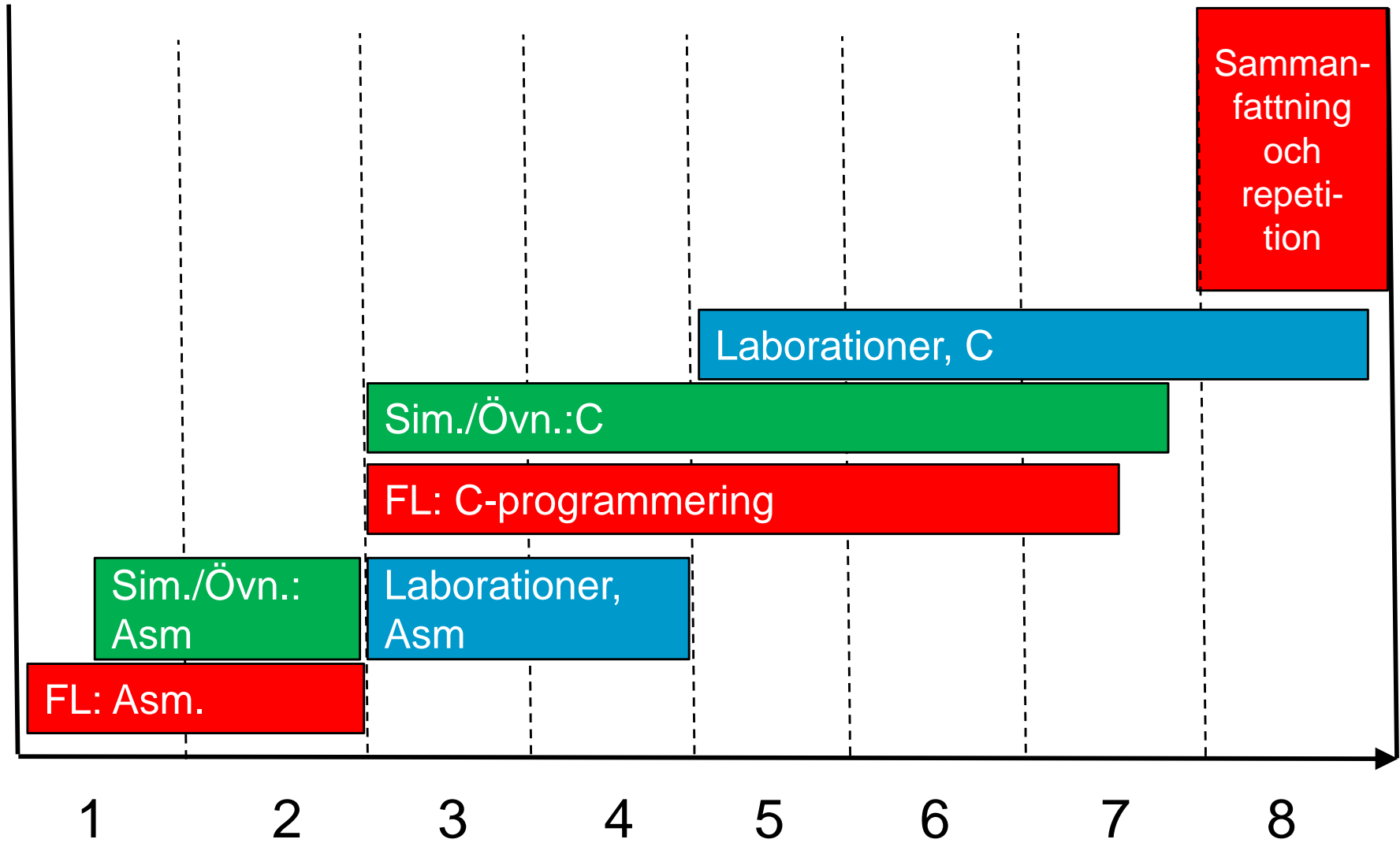
- ❑ Kursens syften är
  - ❑ att vara en introduktion till konstruktion av små inbyggda system och
  - ❑ att ge en förståelse för hur imperativa styrstrukturer översätts till assembler
  - ❑ att ge en förståelse för de svårigheter som uppstår vid programmering av händelsestyrda system med flera indatakällor.
- ❑ Centrala målsättningar är att kunna:
  - ❑ skriva enkla C-program med användande av programspråkets datatyper och styrstrukturer
  - ❑ beskriva motsvarigheten i assembler till typiska programstrukturer i C.
  - ❑ utnyttja de i kursen använda verktygen för programutveckling på ett adekvat sätt
  - ❑ medverka vid konstruktion och programmering av enkla inbyggda system med givna komponenter
  - ❑ konstruera system innefattande olika typer av undantag (interna undantag, avbrott, återstart)
  - ❑ beskriva och exemplifiera några olika typer av digitala kringkomponenter och deras användning.



# Kurslitteratur

- “Vägen till C” ( DC ) alt. “The C programming language”
- “Arbetsbok för MC12” ( DC )
- Laborations-PM och inlämningsuppgift,  
finns att plocka upp på DC
- Övrigt material är på elektronisk form och du kan hämta det via kursens hemsida.

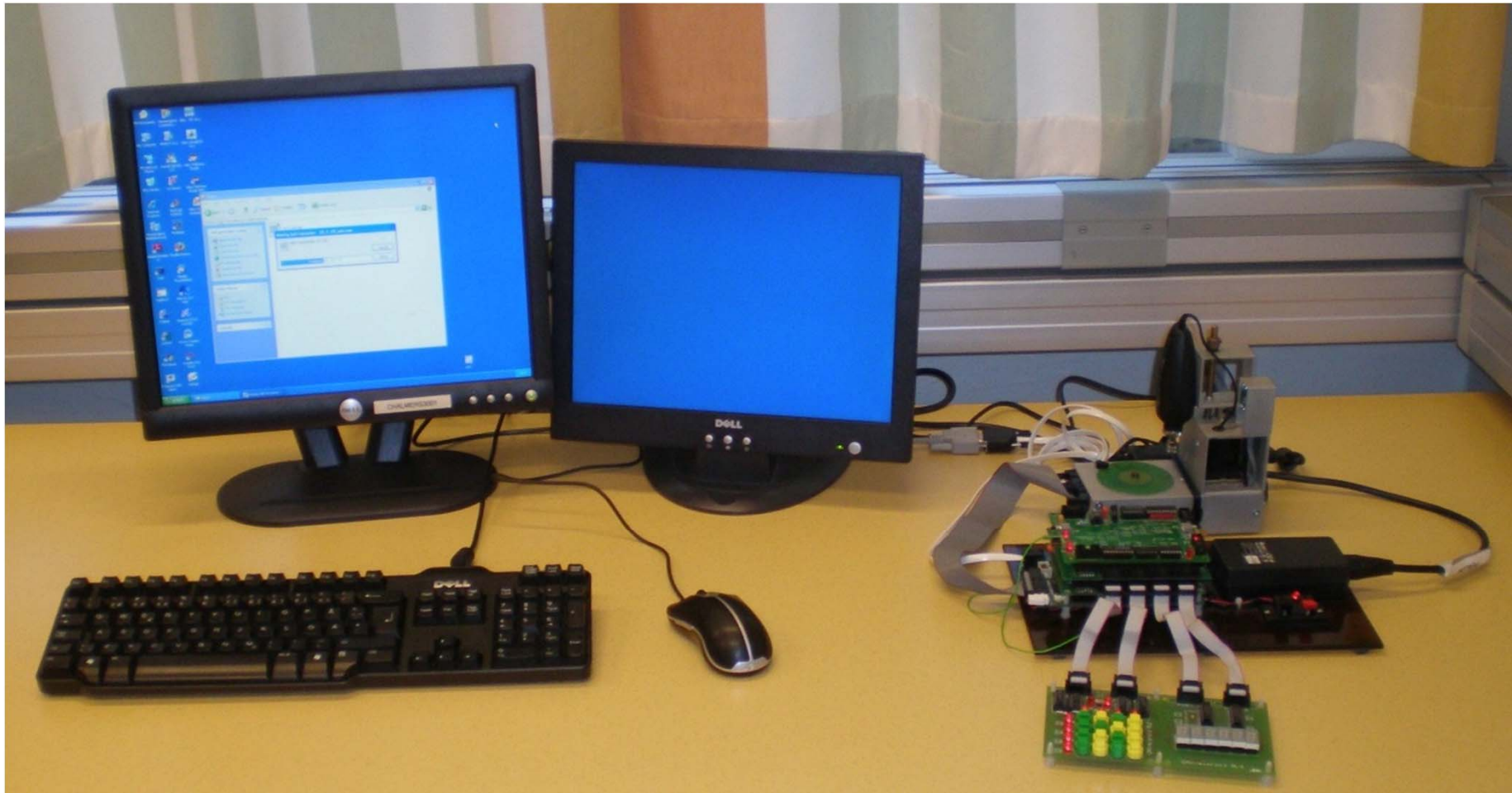
# Genomförande



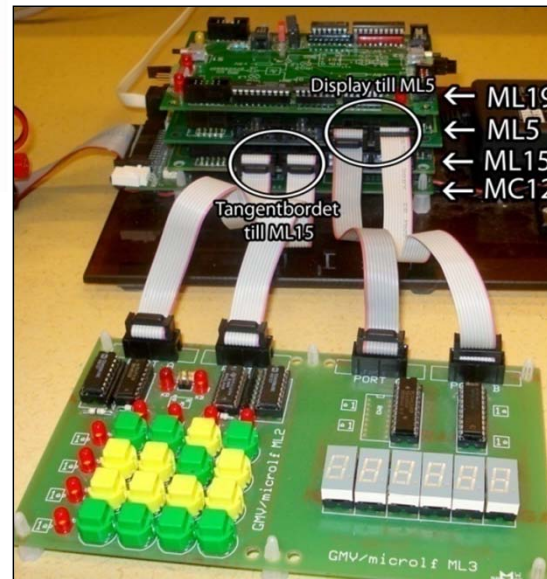
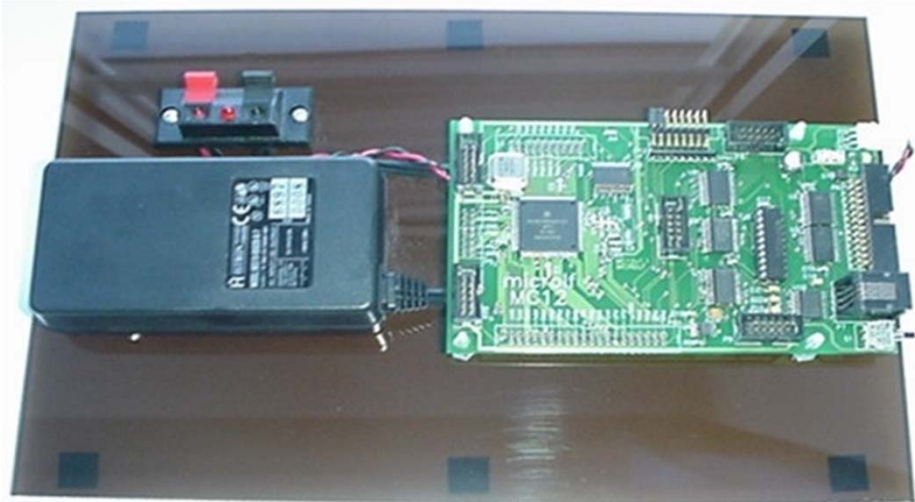
# Laborationsöversikt

- ❑ Moment 1: Inledande programmering i assembler  
"Introduktion till laborationssystemet"
- ❑ Moment 2: Programutveckling i assembler  
"Övervakning/styrning av bormaskin"
- ❑ Moment 3: Programutveckling i assembler  
"Pseudoparallell exekvering"
- ❑ Moment 4: Programutveckling i C  
"Goldbach hypotes" och "Prioritetskö"
- ❑ Moment 5: Maskinnära programmering i C  
"Övervakning/styrning av bormaskin"

# Laborationsplats



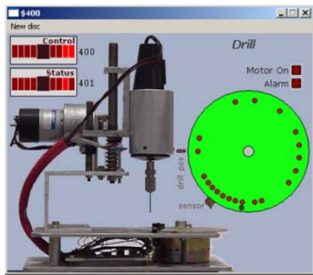
# Laborationssystem





Moment 1, 2 och 3:

## ETERM för Simulator och laborationssystem



GMV ETERM 6 for MC12

File Edit Debug Windows Help

```

main7.s12
*
#define RUNFAST

USE      IODEFS.S12

ORG      Start
LDS      #BOS

* Calm the drill...
MOVW    #0,DCtrl
* .. reflect drill status
MOVW    #0,DCCopy

Loop:
JSR     KEYB1      ; wait for
NOP
JSR     COMMAND ; do instruction
BRA     Loop

*****
*SUBROUTIN COMMAND
*Beskrivning: Rutinen avgör vilken
*kommandosubrutin som skall anropas och anropar
*denna.
*Anrop:      JSR     COMMAND
*Indata:     Kommandonummer i reg A
*Utdata:     Inga
*Reg-påverkan: Ingen
*Anrop subr: SUBO - SUB7
*****

MAX EQU 7
COMMAND PSHA
        PSWX

        CMPA #MAX
        BHI COMEX
            
```

MC12 Visual Simulator

Current target setup: drill

**Control**

Exception handling  
 ROM write E/D

Interrupts  i  X  
 Activate   
 Service

**Status**

IO break  
 IRQ break  
 Running

**Program**

Address	Instruction	Step
1000	LDS #3B00	
1003	MOVW #00,\$0F00	
1008	MOVW #00,\$119C	
100D	JSR \$1037	
1010	NOP	
1011	JSR \$1016	
1014	BRA \$100D	
1016	PSHA	
1017	PSWX	
1018	CMPA #07	
101A	BHI \$1024	
101C	ASLA	

**Stack**

Address	Value
SP (SP)	
3C7A	00
3C7B	00
3C7C	00
3C7D	00
3C7E	00
3C7F	00
3C80	00
3C81	00

**Registers**

Register	Value	Label
A:B (D)	0000	X
Y	0000	
PC	1000	
SP	3C80	
SXHINZVC	11010000	CCR

mem: 0000-0030

\$9C0

Interface Interrupts

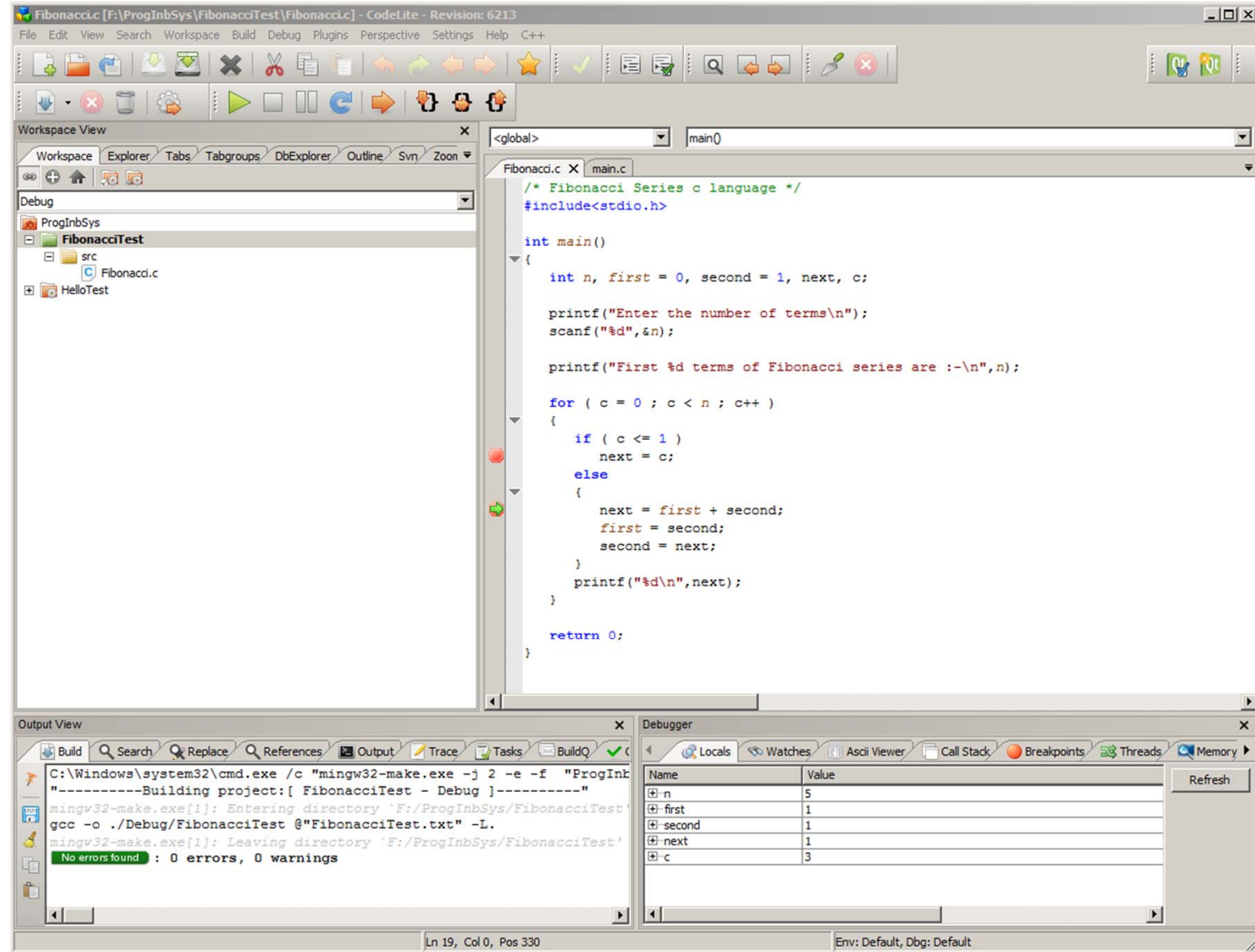
\$400

New disc

**Control**

Motor On   
 Alarm

CodeLite för  
moment 4  
"Morsealfabetet"  
och "Prioritetskö".



The screenshot shows the CodeLite IDE interface. The main editor displays a C++ program for calculating the Fibonacci series. The program includes `<stdio.h>` and defines a `main()` function that prompts the user for the number of terms, reads the input, and prints the first `n` terms of the Fibonacci series. The program is compiled and executed, as shown in the Output View at the bottom.

```
/* Fibonacci Series c language */
#include<stdio.h>

int main()
{
    int n, first = 0, second = 1, next, c;

    printf("Enter the number of terms\n");
    scanf("%d", &n);

    printf("First %d terms of Fibonacci series are :-\n", n);

    for ( c = 0 ; c < n ; c++ )
    {
        if ( c <= 1 )
            next = c;
        else
        {
            next = first + second;
            first = second;
            second = next;
        }
        printf("%d\n", next);
    }

    return 0;
}
```

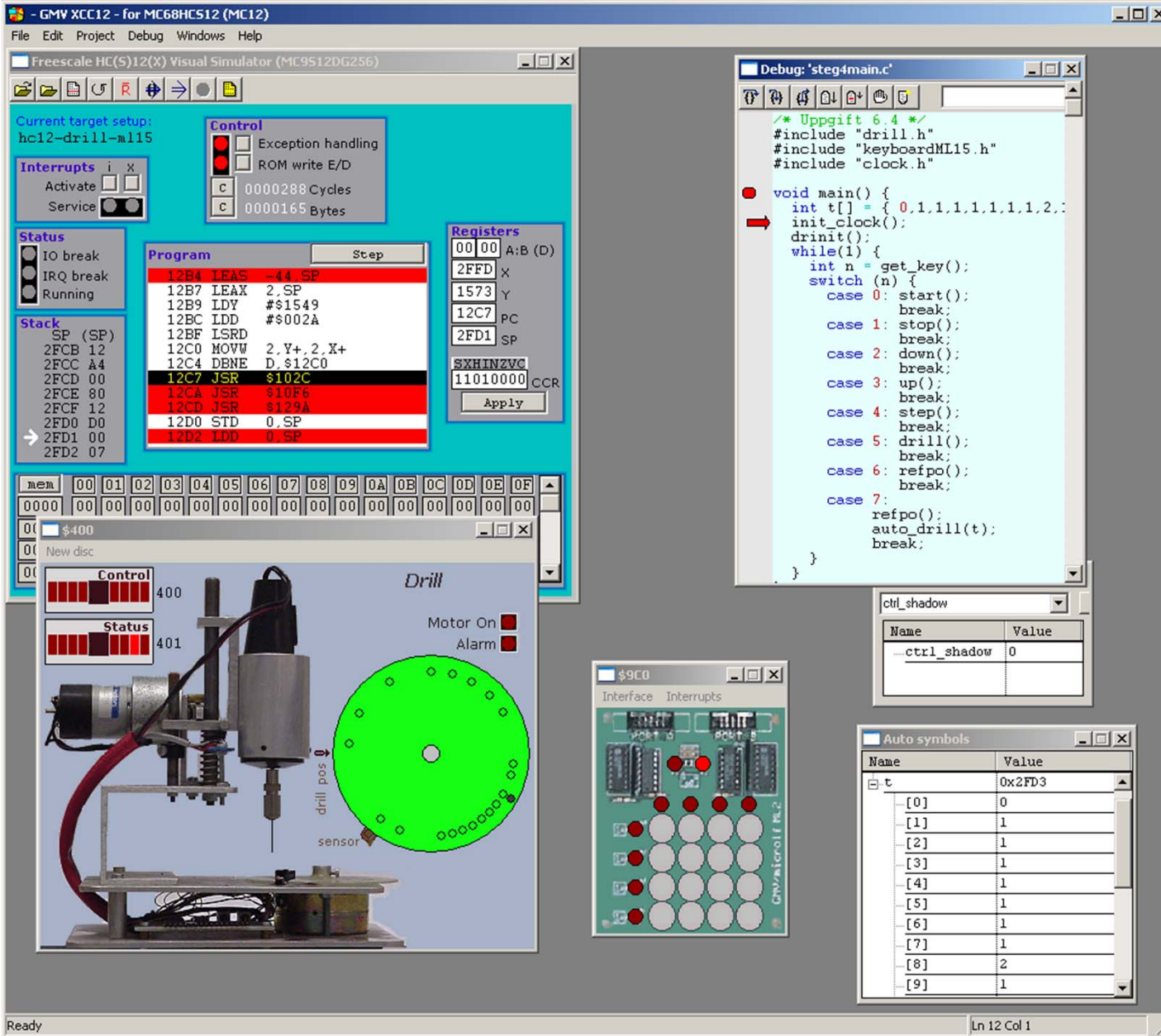
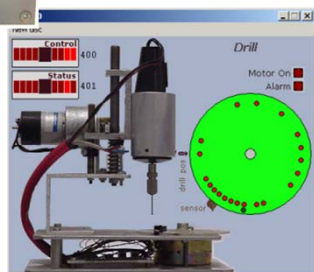
The Output View shows the following text:

```
C:\Windows\system32\cmd.exe /c "mingw32-make.exe -j 2 -e -f "ProgInk
"-----Building project:[ FibonacciTest - Debug ]-----"
mingw32-make.exe[1]: Entering directory `F:/ProgInbSys/FibonacciTest'
gcc -o ./Debug/FibonacciTest @"FibonacciTest.txt" -L.
mingw32-make.exe[1]: Leaving directory `F:/ProgInbSys/FibonacciTest'
No errors found : 0 errors, 0 warnings
```

The Debugger window shows the following variables and their values:

Name	Value
n	5
first	1
second	1
next	1
c	3

## Moment 5:

XCC12  
för Simulator  
och laborations-  
system

The screenshot displays the Freescale Visual Simulator interface for the MC9S12DG256. The main window shows the program execution environment with the following components:

- Control Panel:** Shows exception handling and ROM write E/D options. Cycles: 0000288, Bytes: 0000165.
- Program Window:** Lists assembly instructions with addresses and comments. The current instruction is `12C7 JSR $102C`.
- Registers Window:** Shows register values: A:B (D) 0000, X 2FFD, Y 1573, PC 12C7, SP 2FD1, SXHINZVC, and CCR 11010000.
- Code Editor:** Displays the source code for `steg4main.c`. The code includes headers for `drill.h`, `keyboardML15.h`, and `clock.h`. The `main` function initializes a timer array `t` and enters a `while` loop that processes key events and calls `drill()` and `refpo()` functions.
- Hardware Interface:** Shows a keyboard interface and a drill system with a motor, alarm, and sensor. The drill system is currently in a "Drill" state.
- Auto symbols Window:** Lists symbols and their values, including `t` at `0x2FD3` and an array of values from 0 to 1.

# Inför laborationerna

- ❑ Laborationerna måste vara väl förberedda innan laborationstillfället
- ❑ Utveckling och test kan göras med simulatorer
- ❑ Använd kodnings-/simuleringsövningar och hemarbete för förberedelserna
- ❑ ETERM, CodeLite och XCC12 finns på kursens "resurssida", hämta och installera omgående
- ❑ OBS: Laborationerna börjar i läsvecka 3  
**ANMÄL ER SENAST ONSDAG LV2  
(via kursens hemsida i PingPong)**