

# Programmering av inbyggda system 2015/2016

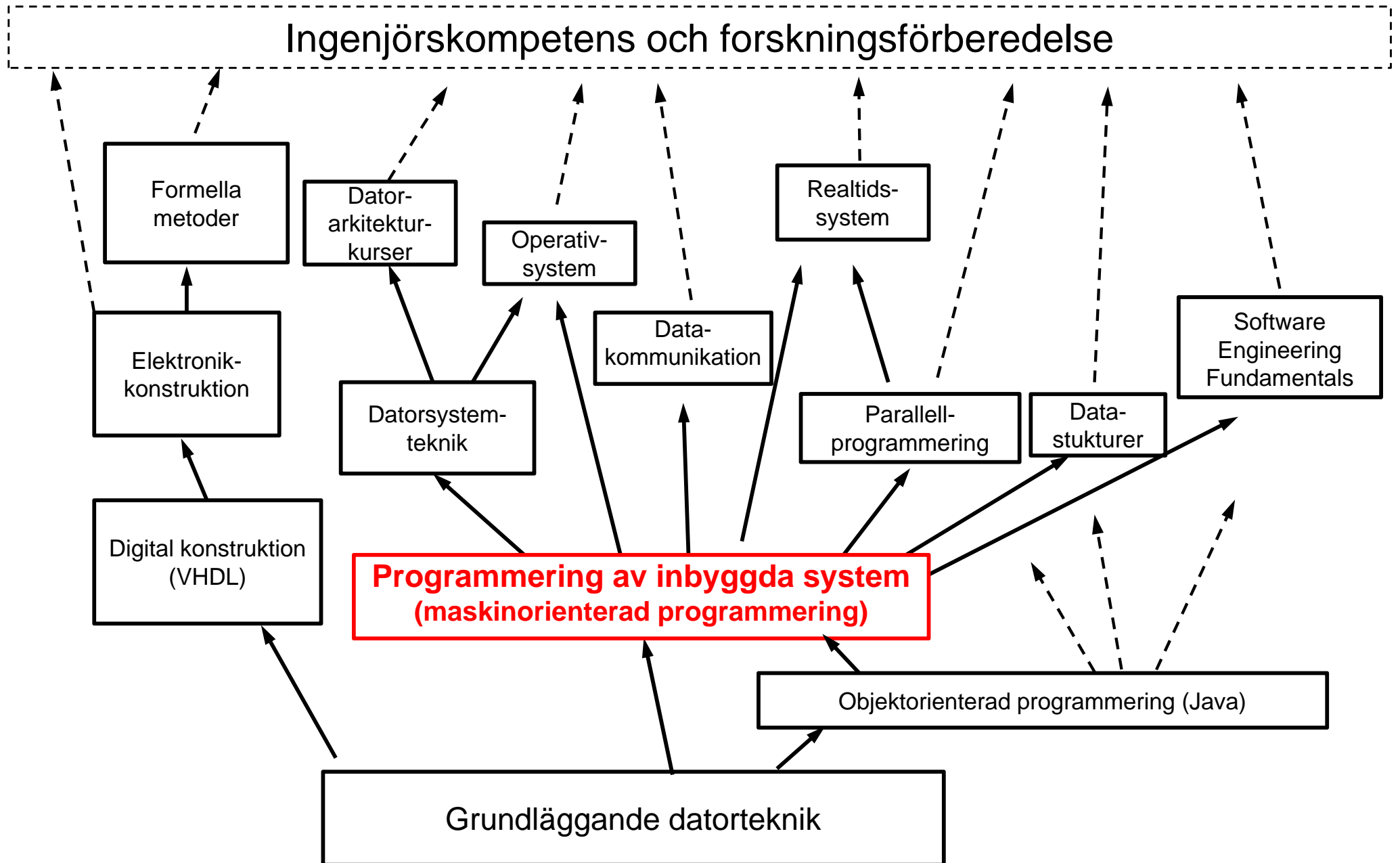
Kursintroduktion  
Roger Johansson

Ur innehållet:

Syften, målsättningar, kurslitteratur och genomförande  
Översikt av laborationer

# Syften och målsättningar

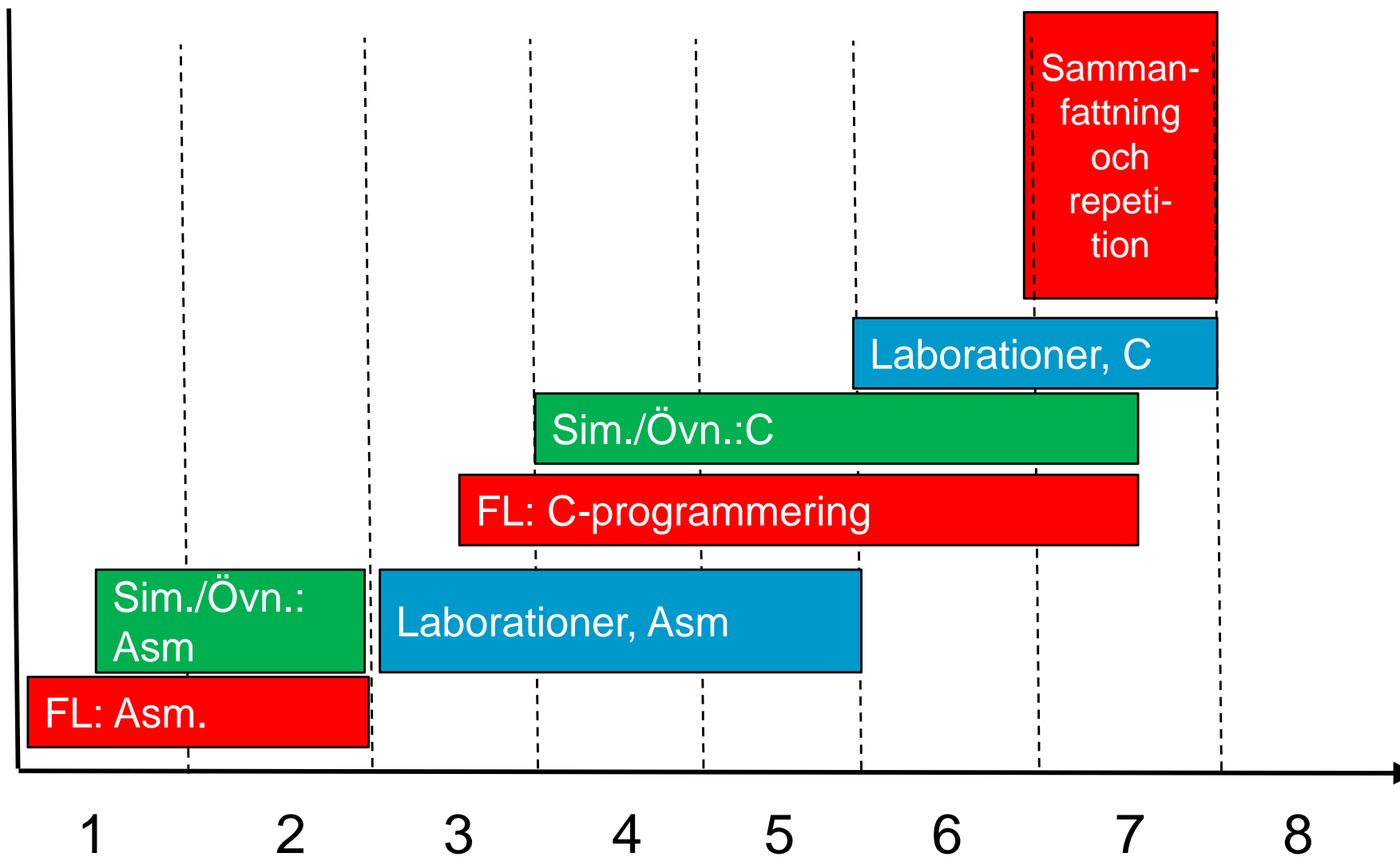
- ❑ Kursens syften är
  - ❑ att vara en introduktion till konstruktion av små inbyggda system och
  - ❑ att ge en förståelse för hur imperativa styrstrukturer översätts till assembler
  - ❑ att ge en förståelse för de svårigheter som uppstår vid programmering av händelsestyrda system med flera indatakällor.
- ❑ 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” (Cremona) alt. “The C programming language”
- “Arbetsbok för MC12” ( Cremona )
  
- Laborations-PM,  
delas ut
  
- Övrigt material är på elektronisk form och du kan hämta det via kursens hemsida.

# Genomförande



Vecka

1

2

3

4

5

6

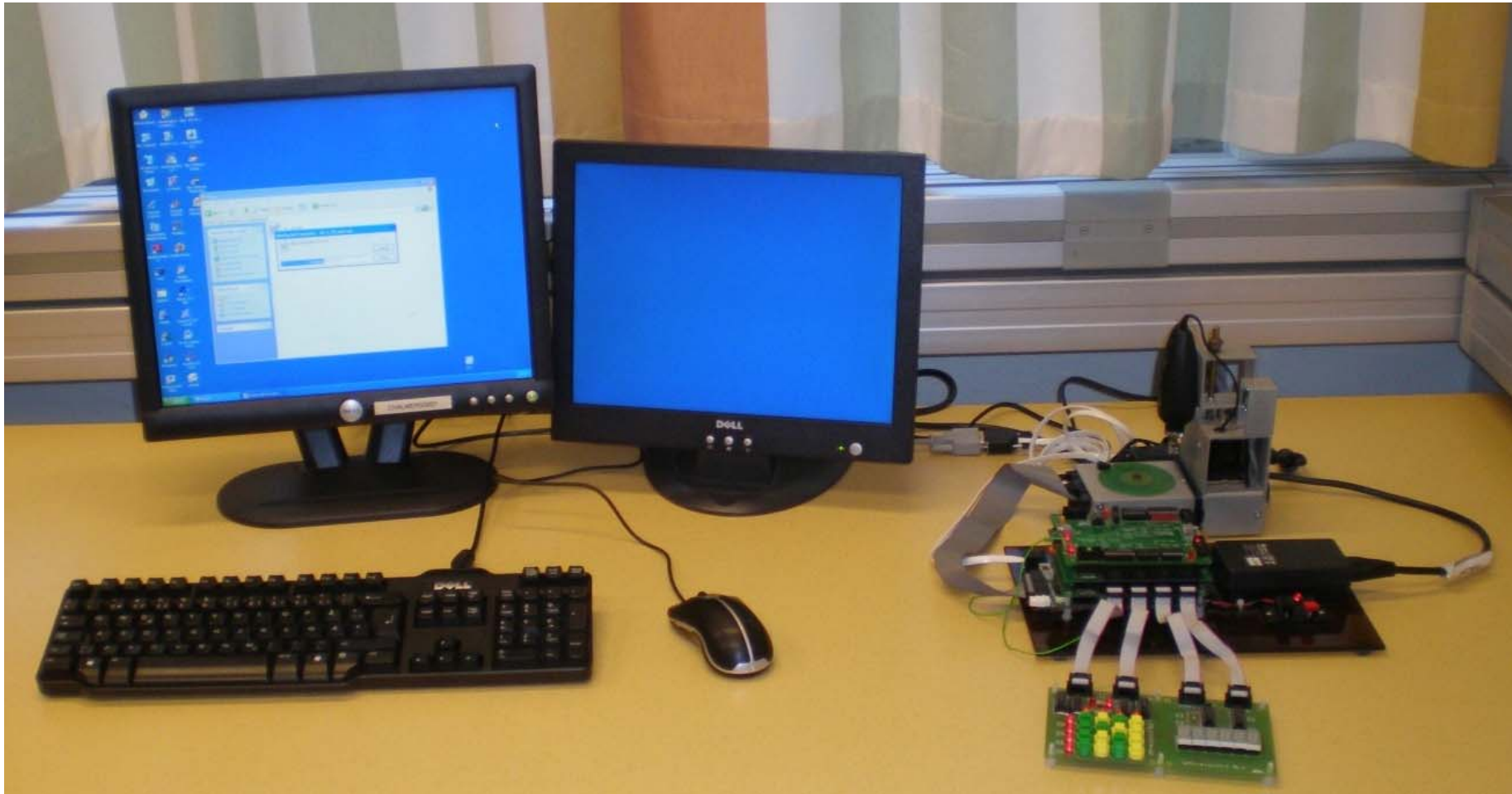
7

8

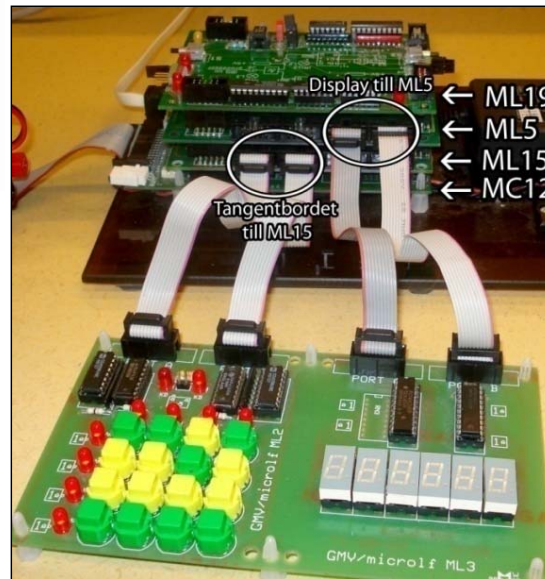
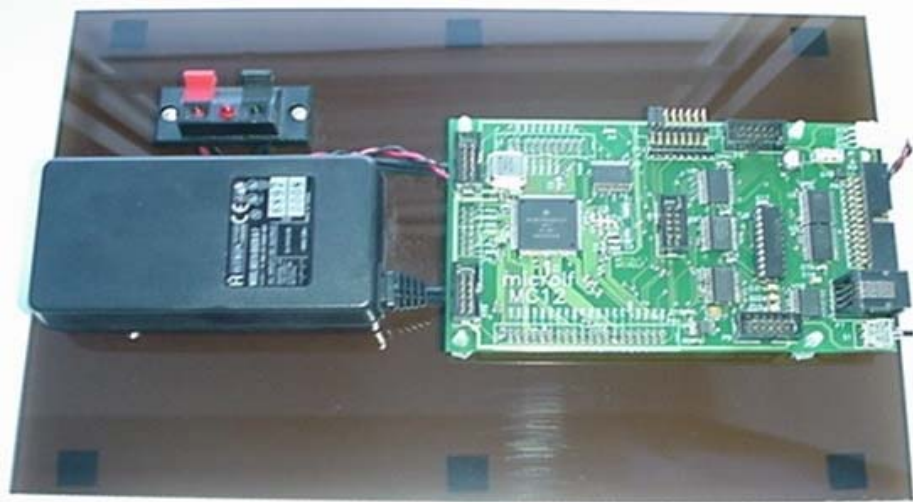
# Laborationsöversikt

- ❑ 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  
"Prioritetskö"
- ❑ Moment 5: Maskinnära programmering i C  
"Övervakning/styrning av bormaskin"

# Laborationsplats



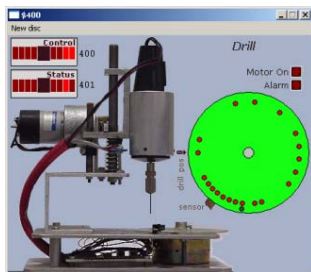
# Laborationssystem





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
        PSHX

        CMPA  #MAX
        BHI  COMEX
                
```

**MC12 Visual Simulator**

Current target setup: drill

**Control**

Exception handling  
 Exception handling  
 ROM write E/D

**Interrupts** i x  
 Activate   
 Service

**Status**  
 IO break  
 IRQ break  
 Running

**Stack**

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

**Program**

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

**Registers**

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

Apply

**\$400**

Control 400

Status 401

Motor On

Alarm

Drill machine interface showing a green circular sensor area.

**\$9C0**

Interface Interrupts

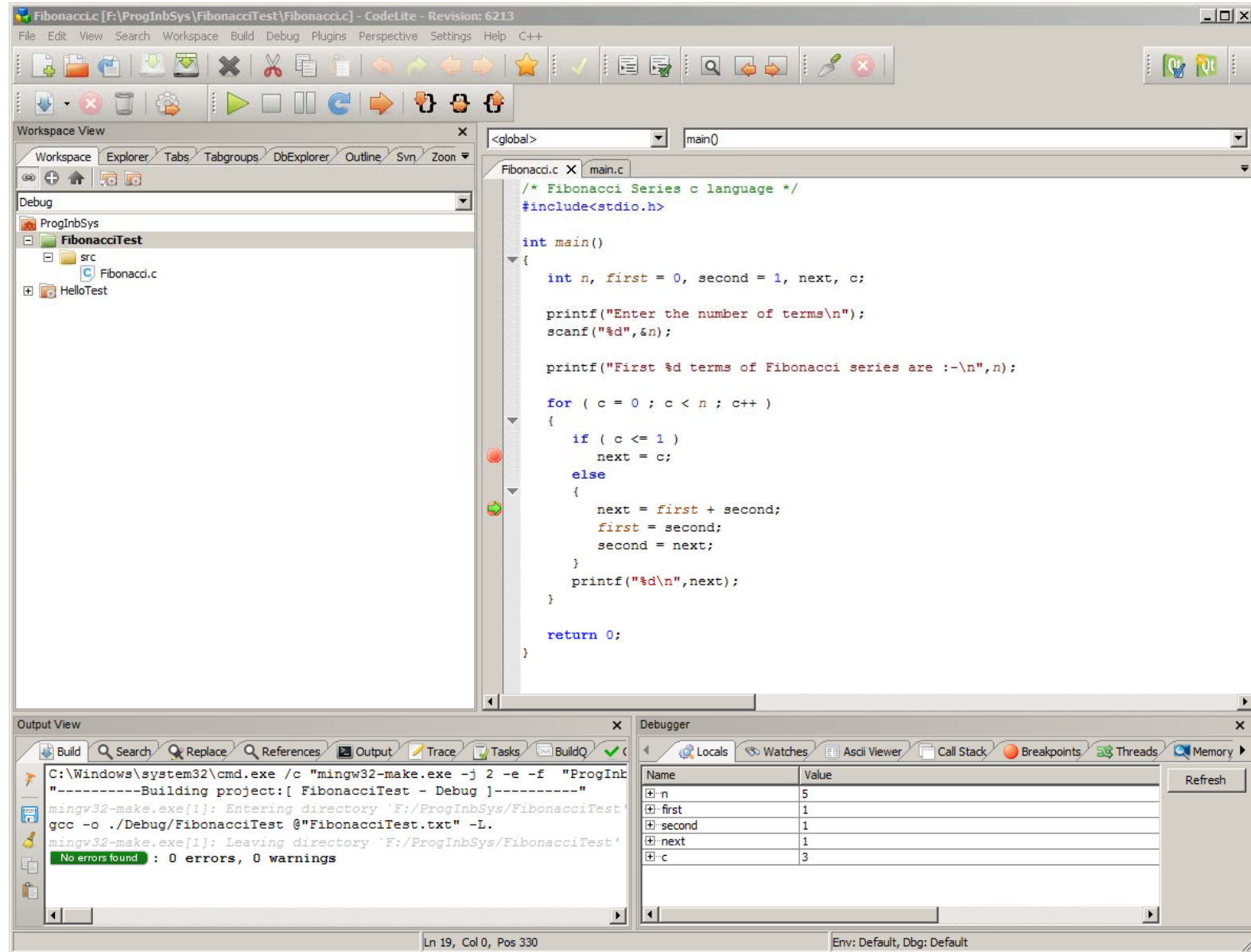
PCMT 0 PCMT 1

CMVstator f RL.2

Hardware interface board with various pins and buttons.

Load Done Ln 42 Col 14

# CodeLite för moment 4 "Prioritetskö".



The screenshot displays the CodeLite IDE interface. The main editor shows the following C code in `main.c`:

```
/* 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 compilation output:

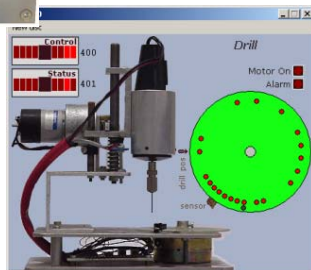
```
C:\Windows\system32\cmd.exe /c "mingw32-make.exe -j 2 -e -f "ProgInk
"-----Building project:[ FibonacciTest - Debug ]-----"
mingw32-make.exe[1]: Entering directory `F:/ProgInSys/FibonacciTest'
gcc -o ./Debug/FibonacciTest @"FibonacciTest.txt" -L.
mingw32-make.exe[1]: Leaving directory `F:/ProgInSys/FibonacciTest'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

The Debugger window shows the following local variables:

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

## Moment 5:

# XCC12 för Simulator och laborations- system



**- GMV XCC12 - for MC68HC512 (MC12)**

File Edit Project Debug Windows Help

Freescall HC(S)12(X) Visual Simulator (MC9512DG256)

Current target setup: hc12-drill-m115

**Control**

- Exception handling
- ROM write E/D
- C 0000288 Cycles
- C 0000165 Bytes

**Interrupts**

- Activate
- Service

**Status**

- IO break
- IRQ break
- Running

**Stack**

- SP (SP)
- 2FCB 12
- 2FCC A4
- 2FCD 00
- 2FCE 80
- 2FCF 12
- 2FD0 D0
- 2FD1 00
- 2FD2 07

**Program**

Address	Instruction	Comment
12B4	LEAS	-44, SP
12B7	LEAX	2, SP
12B9	IDY	#\$1549
12BC	LDD	#\$002A
12BF	LSRD	
12C0	MOVW	2, Y+, 2, X+
12C4	DBNE	D, \$12C0
12C7	JSR	\$102C
12CA	JSR	\$10F6
12CD	JSR	\$129A
12D0	STD	0, SP
12D2	LDD	0, SP

**Registers**

- 0000 A:B (D)
- 2FFD X
- 1573 Y
- 12C7 PC
- 2FD1 SP
- SXHNZVC
- 11010000 CCR

**Debug: 'steg4main.c'**

```

/* Uppgift 6.4 */
#include "drill.h"
#include "keyboardM15.h"
#include "clock.h"

void main() {
    int t[] = { 0,1,1,1,1,1,1,1,2,0 };
    init_clock();
    drinit();
    while(1) {
        int n = get_key();
        switch (n) {
            case 0: start();
                    break;
            case 1: stop();
                    break;
            case 2: down();
                    break;
            case 3: up();
                    break;
            case 4: step();
                    break;
            case 5: drill();
                    break;
            case 6: refpo();
                    break;
            case 7:
                    refpo();
                    auto_drill(t);
                    break;
        }
    }
}
    
```

**ctrl\_shadow**

Name	Value
ctrl_shadow	0

**Auto symbols**

Name	Value
t	0x2FD3
[0]	0
[1]	1
[2]	1
[3]	1
[4]	1
[5]	1
[6]	1
[7]	1
[8]	2
[9]	1

**Drill**

Motor On

Alarm

drill pos ↓

sensor

**Interface Interrupts**

GMV/Motor/E R.2

Ready

Ln 12 Col 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)**