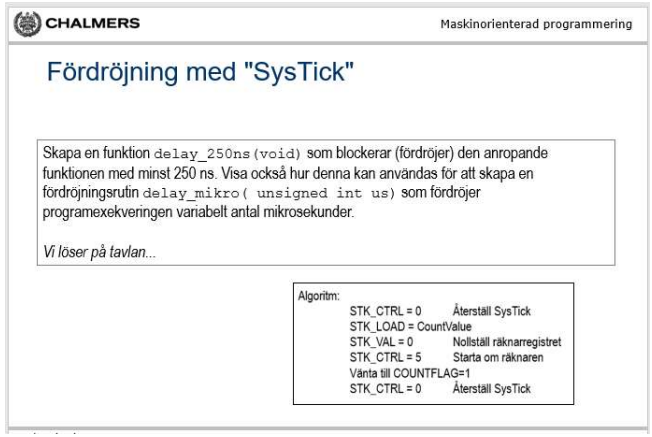

CHALMERS Maskinorienterad programmering

Fördröjning med "SysTick"

Skapa en funktion `delay_250ns(void)` som blockerar (fördröjer) den anropande funktionen med minst 250 ns. Visa också hur denna kan användas för att skapa en fördröjningsrutin `delay_mikro( unsigned int us)` som fördröjer programexekveringen variabelt antal mikrosekunder.

Vi löser på tavlan...

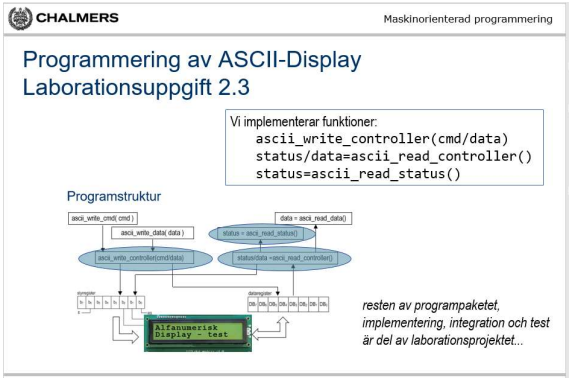
Algoritm:

STK_CTRL = 0	Återställ SysTick
STK_LOAD = CountValue	
STK_VAL = 0	Nollställ räknarregistret
STK_CTRL = 5	Starta om räknaren
Vänta till COUNTFLAG=1	
STK_CTRL = 0	Återställ SysTick

```
#define                 STK_CTRL    ((volatile unsigned int *)(0xE000E010))
#define                 STK_LOAD    ((volatile unsigned int *)(0xE000E014))
#define                 STK_VAL     ((volatile unsigned int *)(0xE000E018))

void delay_250ns( void )
{
     /* SystemCoreClock = 168000000 */
     *STK_CTRL = 0;
     *STK_LOAD = ( (168/4) -1 );
     *STK_VAL = 0;
     *STK_CTRL = 5;
     while( (*STK_CTRL & 0x10000 )== 0 );
     *STK_CTRL = 0;
}

void delay_micro(unsigned int us)
{
#ifdef            SIMULATOR
     us = us / 1000;
     us++;
#endif
     while( us > 0 )
     {
                  delay_250ns();
                  delay_250ns();
                  delay_250ns();
                  delay_250ns();
                  us--;
     }
}
```




```
#define            B_E         0x40        /* Enable-signal */
#define            B_SELECT    4           /* Select ASCII-display */
#define            B_RW        2           /* 0=Write, 1=Read */
#define            B_RS        1           /* 0=Control, 1=Data */

#define GPIO_E     0x40021000       /* MD407 port E */
  ...etc...


void ascii_ctrl_bit_set( unsigned char x  )
{
     char c;
     c = *GPIO_E_ODRLOW;
     *GPIO_E_ODRLOW = B_SELECT | x | c;
}


void ascii_ctrl_bit_clear( unsigned char x )
{
     char c;
     c = *GPIO_E_ODRLOW;
     c = c & ~x;
     *GPIO_E_ODRLOW = B_SELECT | c;
}
```

```
void ascii_write_controller( char c )
{
     ascii_ctrl_bit_set( B_E );
     *GPIO_E_ODRHIGH = c;
     ascii_ctrl_bit_clear( B_E );
     delay_250ns();
}


char ascii_read_controller( void )
{
     char c;
     ascii_ctrl_bit_set( B_E );
     delay_250ns();
     delay_250ns();
     c = *GPIO_E_IDRHIGH;
     ascii_ctrl_bit_clear( B_E );
     return c;
}

char ascii_read_status()
{
     char c;
     *GPIO_E_MODER = 0x00005555;
     ascii_ctrl_bit_set( B_RW );
     ascii_ctrl_bit_clear( B_RS );
     c = ascii_read_controller();
     *GPIO_E_MODER = 0x55555555;
     return c;
}
```