

```

#include "X10ModtagerDriver.h"

#define Counter 20

/*****
  Definerer X10 Beskeder
*****/

#define HOUSEA "01101001"

#define UNIT1 "01101001"
#define UNIT2 "10101001"
#define UNIT3 "01011001"
#define ON "01011001"
#define OFF "01011010"

#define START "1110"
#define SUFFIX_ADR "01" // Unit Address
#define SUFFIX_CMD "10" // Command Address
#define WAIT "000000" // Six Zero-Crossings

/*****
  Initialiserer arrays/pointers
*****/

char *HouseCodeArray_[2] = {WAIT, HOUSEA};
char *UnitCodeArray_[4] = {WAIT, UNIT1, UNIT2, UNIT3};
char *FunctionCodeArray_[3] = {WAIT, OFF, ON};

/*****
  Implementerer X10 Prototyper
*****/

void InitX10Modtager(void)
{
    DDRD &= 0b11111011;
    MCUCR |= 0b00000001;
    GICR |= 0b01000000;

    //PC3 sættes til 120kHz input
    DDRC &= 0b11110111;

    //nulstiller zero-cross flag
    DDRB = 0x00;
};

char X10ModtagerBit(void)
{
    DDRB = 0x00;

    int setCounter = 0;

    while(DDRB != 0xFF)
    {

    }

    _delay_us(100);

    int i;
    for(i=0; i<Counter; i++)
    {
        _delay_us(20);

        if(PINC & (1<<PC3))
        {
            setCounter++;
        }
    }

    DDRB = 0x00;

    if(setCounter > Counter/2)

```

```
        return '1';
    else
        return '0';
}

char X10ModtagerStby(void)
{
    char StartChk[] = "0000";
    unsigned char StartSamlig;
    int i;

    while(1)
    {
        StartSamlig = 1;

        for(i=0; i<3; i++)
        {
            StartChk[i] = StartChk[i+1];
        }

        StartChk[i] = X10ModtagerBit();

        for(i=0; i<4 && StartSamlig==1; i++)
        {
            if(StartChk[i] != START[i])
                StartSamlig = 0;
        }

        if(StartSamlig == 1)
            return 1;
    }
}

void X10ModtagerPakke(char* DataPakkeModtaget)
{
    int i;

    for(i=0; i<4; i++)
        DataPakkeModtaget[i] = START[i];

    for (i=4; i<94; i++)
        DataPakkeModtaget[i] = X10ModtagerBit();

    for (i=94; i<100; i++)
        X10ModtagerBit();
}

unsigned char X10SamligPakke(char* DataPakkeModtaget, char* DataPakkeSamlig)
{
    char samlig = 1;

    int i;
    for(i = 0; i < 94; i++)
    {
        if (DataPakkeModtaget[i] != DataPakkeSamlig[i])
            samlig = 0;
    }

    if ( samlig == 1 )
    {
        return 1;
    }
    else
        return 0;
}

void InitDataPakke(char* datapakke, char housecode, char unitcode, char FunctionCode)
{
    char *HouseKode_ptr = HouseCodeArray_[housecode];
    char *UnitKode_ptr = UnitCodeArray_[unitcode];
    char *FunctionCode_ptr = FunctionCodeArray_[FunctionCode];

    int i, sted = 0;
```

```
// Send adressen første gang
for(i=0; START[i] != '\0'; i++, sted++)
    datapakke[sted] = START[i];

for(i=0; HouseKode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = HouseKode_ptr[i];

for(i=0; UnitKode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = UnitKode_ptr[i];

for(i=0; SUFFIX_ADR[i] != '\0'; i++, sted++)
    datapakke[sted] = SUFFIX_ADR[i];

// Send adressen anden gang
for(i=0; START[i] != '\0'; i++, sted++)
    datapakke[sted] = START[i];

for(i=0; HouseKode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = HouseKode_ptr[i];

for(i=0; UnitKode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = UnitKode_ptr[i];

for(i=0; SUFFIX_ADR[i] != '\0'; i++, sted++)
    datapakke[sted] = SUFFIX_ADR[i];

// Vent 3 cykler (6 zero-crossing)
for (i = 0; WAIT[i] != '\0'; i++, sted++)
    datapakke[sted] = WAIT[i];

//Send kommando første gang
for (i = 0; START[i] != '\0'; i++, sted++)
    datapakke[sted] = START[i];

for (i = 0; HouseKode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = HouseKode_ptr[i];

for (i = 0; FunctionCode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = FunctionCode_ptr[i];

for (i = 0; SUFFIX_CMD[i] != '\0'; i++, sted++)
    datapakke[sted] = SUFFIX_CMD[i];

// Send kommando anden gang
for (i = 0; START[i] != '\0'; i++, sted++)
    datapakke[sted] = START[i];

for (i = 0; HouseKode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = HouseKode_ptr[i];

for (i = 0; FunctionCode_ptr[i] != '\0'; i++, sted++)
    datapakke[sted] = FunctionCode_ptr[i];

for (i = 0; SUFFIX_CMD[i] != '\0'; i++, sted++)
    datapakke[sted] = SUFFIX_CMD[i];
}

// Zero-crossing interrupt
ISR(INT0_vect)
{
    DDRB = 0xFF;
}
```