```c
#include "Kommunikation.h"
#include "VBTE-Driver.h"

void getFromKI(struct smflags* sm)
{
        uint8 readByte = 0;
        uint8 manuByte = 0;
        readByte = UART_1_GetChar();
        if(readByte > 0){
        switch( readByte )
        {
            case CMD_INIT :
                UART_1_PutChar(ACK_INIT);
                (*sm).autoflag = 1;
                sm->VBTE1Niveau = writeToVbte(VBTE1Addr, VBTENIVEAU);
                sm->VBTE2Niveau = writeToVbte(VBTE2Addr, VBTENIVEAU);
            break;

            case CMD_AUTO :
                UART_1_PutChar(ACK_AUTO);
                if((*sm).autoflag != 1){
                    (*sm).autoflag = 1;
                    (*sm).vinkelVal = STLEVEL;
                    }
                else
                    break;
            break;

            case CMD_MANU :
                CyDelay(1);
                manuByte = UART_1_GetChar();
                UART_1_PutChar(ACK_MANU);
                convertToValue(manuByte, sm); // vi må lige finde ud af hvilke værdier der
                betyder hvad?
            break;


            case REQ_VBTE1_STATUS :
                UART_1_PutChar(STATUS_VBTE1);
                UART_1_PutChar((*sm).VBTE1Niveau);
                UART_1_PutChar((*sm).VBTE1Status);
                //do stuff
            break;

            case REQ_VBTE2_STATUS :
                UART_1_PutChar(STATUS_VBTE2);
                UART_1_PutChar((*sm).VBTE2Niveau);
                UART_1_PutChar((*sm).VBTE2Status);
            break;
            case REQ_HAELDNING :
                UART_1_PutChar(STATUS_HAELDNING);
                UART_1_PutChar(convertToEnum((*sm).levelVal));
            break;
            case CMD_TERMINATION :
                UART_1_PutChar(ACK_TERMINATION);
                (*sm).autoflag = 0;
                writeToVbte(VBTE1Addr, LUKVENTIL);
```

```
                writeToVbte(VBTE1Addr, LUKVENTIL);

            default :
                UART_1_ClearRxBuffer();
                UART_1_ClearTxBuffer();
                UART_1_PutChar(UNKNOWN);
                //do default stuff
            break;}
        }
}

int convertToEnum(int value)
{
    int i = 0;

    if(value >= STLEVEL && value <= (STLEVEL+16))
        return NUL;

    for(i = SBSTART; i < SBSLUT; i++)
    {
        if( value >= (STLEVEL + (17*i)) && value <= ((STLEVEL+16) + (17*i)))
            return i+SBSTART;
    }
    for(i = BBSLUT; i > BBSTART; i--)
    {
        if( value >= (STLEVEL - (17*(i-BBSTART))) && value <= ((STLEVEL+16) - (17*(i-BBSTART
        ))))
            return i;
    }
    return 255;

}

void convertToValue(int Level, struct smflags* sm)
{
    if( Level == NUL ){
        (*sm).vinkelVal = STLEVEL;
        return;
        }
    if( Level < SBSLUT)
    {
        (*sm).vinkelVal = STLEVEL+(17*(Level-SBSTART));
        return;
    }
    if( Level < BBSLUT )
    {
        (*sm).vinkelVal = STLEVEL-(17*(Level-BBSTART));
        return;
    }
    else
        sm->vinkelVal = STLEVEL; // default hvis enumværdien ikke er kendt

}
```