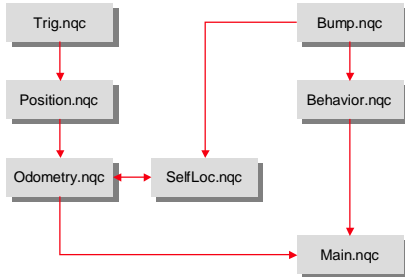


Hilfe zu Übungsblatt 4 – Module



Übungsblatt 4 – Include-Dateien

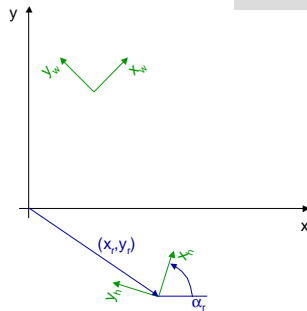
- **Funktionsweise**
 - Include-Dateien werden einfach mitkompiliert
 - Einbinden mit `#include "Dateiname.nqc"`
- **Vorteile**
 - Erhöhung der Übersichtlichkeit
- **Nachteile**
 - Im RCX Command Center muss beim Kompilieren immer die Hauptdatei ausgewählt sein.
 - Das RCX Command Center zeigt Fehler in Include-Dateien nicht an
 - Daher empfiehlt sich teilweise die Nutzung von NQC auf der Kommandozeile
 - Aufruf unter Windows: `\Programme\RCxCC\NQC -TRCX2 main.nqc`
- **Module einklammern in**

```

        #ifndef _DATEINAME_
        #define _DATEINAME_
        // ...
        #endif
    
```

Übungsblatt 4 – Metrische Positionen

► Addition

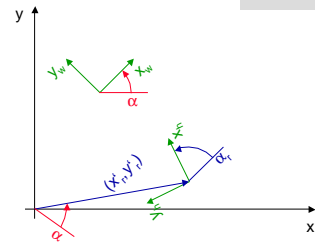


Übungsblatt 4 – Metrische Positionen

► Addition

$$x'_r = x_r \cos \alpha - y_r \sin \alpha$$

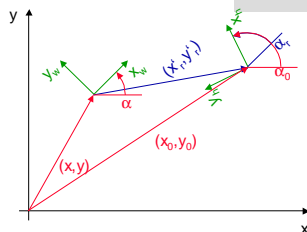
$$y'_r = x_r \sin \alpha + y_r \cos \alpha$$



Übungsblatt 4 – Metrische Positionen

► Addition

- $x'_r = x_r \cos \alpha - y_r \sin \alpha$
- $y'_r = x_r \sin \alpha + y_r \cos \alpha$
- $x_0 = x + x'_r$
- $y_0 = y + y'_r$
- $\alpha_0 = \alpha + \alpha_r$



Übungsblatt 4 – Trig & Position

```

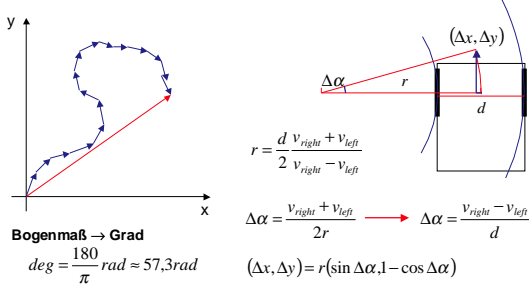
    Trig.nqc
    void sincos(int angle,int& sin,int& cos);
    void atr(int dist,int& angle)

    Position.nqc
    #include "Trig.nqc"
    void posAdd(int& x1,int& y1,int& a1,
               int x2,int y2,int a2) {
        int sin,cos;
        sincos(a1,sin,cos);
        x1 += (x2 * cos - y2 * sin) / 100;
        y1 += (x2 * sin + y2 * cos) / 100;
        a1 += a2;
        posAngleFix(a1);
    }

    void posAngleFix(int& a) {
        if(a >= 180)
            a -= 360;
        else if(a < -180)
            a += 360;
    }

    void posSub(int& x1,int& y1,int& a1,
               int x2,int y2,int a2) {
        int sin,cos;
        sincos(a2,sin,cos);
        x1 -= x2;
        y1 -= y2;
        int temp = (x1 * cos + y1 * sin) / 100;
        y1 = (y1 * cos - x1 * sin) / 100;
        x1 = temp;
        a1 -= a2;
        posAngleFix(a1);
    }
  
```

Übungsblatt 4 – Odometrie



Übungsblatt 4 – Odometry

```

#include "Position.nqc"
#define AXLE_WIDTH 90
#define ODO_TO_CM 10 / 88
#define CM_TO_ODO 88 / 10

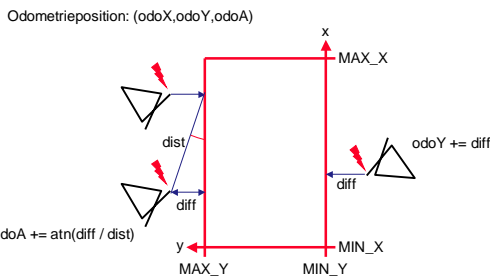
int odometryX = 0,
    odometryY = 0,
    odometryA = 0,
    odoX = 0, // private vars
    odoY = 0,
    odoA = 0;

#include "SelfLoc.nqc"

task odometry() {
    SetSensor(SENSOR_1,SENSOR_ROTATION);
    SetSensor(SENSOR_3,SENSOR_ROTATION);
    int odoLeft = SENSOR_1,
        odoRight = SENSOR_3;

while(true) {
    int diffLeft = (odoLeft - SENSOR_1),
        diffRight = (odoRight - SENSOR_3);
    odoLeft -= diffLeft;
    odoRight += diffRight;
    // TODO: calculate dx, dy, da ...
    posAdd(odoX,odoY,odoA,dx,dy,da);
    selflocExecute();
    dx = odoX * ODO_TO_CM;
    odometryX = dx; // direct assignment
    dy = odoY * ODO_TO_CM;
    odometryY = dy; // direct assignment
    odometryA = odoA;
    Wait(1);
}
}
    
```

Übungsblatt 4 – SelfLoc



Übungsblatt 4 – SelfLoc

```

#include "Bump.nqc"
#include "Odometry.nqc"

// map
#define MIN_X 0
#define MAX_X 55 * CM_TO_ODO
#define MIN_Y 0
#define MAX_Y 32 * CM_TO_ODO

#define THRESHOLD 5 * CM_TO_ODO

void selflocCorrect(int x,int y) {
    // TODO: calc minimum distance from walls in map.
    // if minimum is in x-direction, correct odoX
    // else correct odoY
    // if the same wall was hit at least twice with a
    // minimum distance in between, also correct odoA.
}

void selflocExecute() {
    int x = odoX,
        y = odoY;
    a = odoA;
    if(bumpLeftLatch) {
        // local bumper position is, e.g., at (1cm, 8.5cm)
        // TODO: calc global bumper position in (x,y,a)
        selflocCorrect(x,y);
        bumpLeftLatch = 0;
    }
    else if(bumpRightLatch) {
        // local bumper position is, e.g., at (1cm, -8.5cm)
        // TODO: calc global bumper position in (x,y,a)
        selflocCorrect(x,y);
        bumpRightLatch = 0;
    }
}
    
```

Übungsblatt 4 – Main

```

#include "Odometry.nqc"
#include "Behavior.nqc"

int printValue = 0;

task print() {
while(true) {
    printValue = odometryX;
    SetUserDisplay(printValue,0);
    Wait(50);
    printValue = odometryY;
    SetUserDisplay(printValue,1);
    Wait(50);
    printValue = odometryA;
    SetUserDisplay(printValue,2);
    Wait(50);
}
}

task main() {
    SetSensor(SENSOR_2,SENSOR_LIGHT);
    SetGlobalDirection(OUT_A,OUT_FWD);
    SetGlobalDirection(OUT_C,OUT_REV);
    SetPower(OUT_A | OUT_C,OUT_HALF);

    start odometry;
    start print;
    start bump;
    start behavior;

    startBehavior(LEFTWALL);

while(true)
    Wait(10);
}
    
```