

MovingCap Ethernet flatTRACK / turnTRACK Automatisierungsbeispiel

Bearbeiter: oliver.heggelbacher@fullmo.de

Datum: 2025-05-06

1 INHALT

2	Funktionsbeschreibung	2
2.1	Master-Steuerung.....	2
2.2	Parametrier/Einrichtmöglichkeiten für die Antriebsfunktionen	2
2.3	Übersicht IO-Funktionen.....	2
3	Konfiguration flatTRACK: Langsamfahrt zwischen zwei Endpositionen	3
3.1	Konfiguration Fahrprofile über Eingangsfunktionen	3
3.2	Konfiguration Rückmeldungen über Ausgangsfunktionen	4
3.3	Fahrten über Programmierung / TCP auslösen:.....	5
3.4	Rückmeldungs-Ausgänge über Programmierung / TCP auslesen	5
4	turnTRACK 349 mit Getriebe 5:1 Langsame 360° Drehung	6
4.1	Grundeinstellungen - Getriebeübersetzung und Positionsskalierung in Winkelgrad °	6
4.2	Konfiguration Fahrprofile über Eingangsfunktionen	6
5	Anhang: Konfigurationsskripte für MovingCap CODE	7
5.1	MovingCap CODE Python-Skript für flatTRACK Parametrierung.....	7
5.2	MovingCap CODE Python-Skript für turnTRACK 349 Parametrierung.....	8

2 FUNKTIONSBESCHREIBUNG

Beispielhafte Umsetzung von Fahrprofilen für zwei MovingCap Ethernet (MC ETH)-Antriebe

- 1x MovingCap flatTRACK für Linearbewegung zwischen zwei Endpositionen
- 1x MovingCap turnTRACK 349 mit Getriebe für Rotativ/Drehtellerbewegung 360°

2.1 PARAMETRIER/EINRICHTMÖGLICHKEITEN FÜR DIE ANTRIEBSFUNKTIONEN

1. Über die MovingCap Webseite(n)
2. Über Kickdrive-Projekt (.kickpro-Datei), z.B. auf Basis des Grundbeispiels kickdrive_MC349_ETH_IO_Examples.kicktpl wie auf www.movingcap.de veröffentlicht)
3. Über Kickdrive CODE Python-Skript, das im Antrieb hinterlegt wird – siehe Anhang

2.2 ÜBERSICHT IO-FUNKTIONEN

Bezeichnung	Funktionsweise	Applikationsparametrierung
MovingCap flatTRACK		
IN7	IN / Eingang / Kommandierung	Position 0 absolut anfahren
IN8	IN / Eingang / Kommandierung	Position 450 000 (Beispiel) anfahren
IO1	OUT / Ausgang / Rückmeldung	Rückmeldung: Ist in Position 0 (oder das für IN7 definierte Ziel); kein Fehler
IO2	OUT / Ausgang / Rückmeldung	Rückmeldung: Ist in Position 450 000 (oder das für IN8 definierte Ziel); kein Fehler
MovingCap turnTRACK 349		
IN1	IN / Eingang / Kommandierung	um 360° vorwärts drehen
IN2	IN / Eingang / Kommandierung	0°-Marke anfahren
OUT1	OUT / Ausgang / Rückmeldung	Antrieb in Position, kein Fehler

2.3 INTERAKTION / MASTER-STEUERUNG

Die eingerichteten Fahrprofile und Statusmeldungen können alternativ über eine der folgenden Kommunikationswege kommandiert / abgefragt / getestet werden:

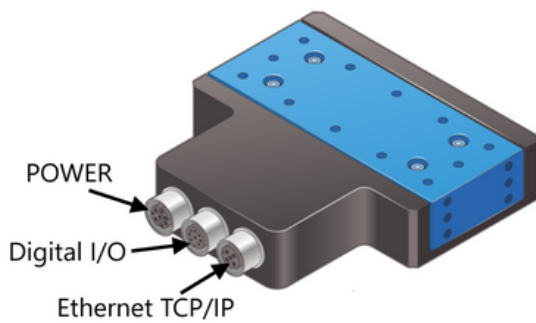
1. Manueller Test über Webseite (siehe Menü IO ← IN, **IN Simulator**)
2. Ansteuerung und Rückmeldung über 24V digitale Eingangs- und Ausgangssignale
3. Ansteuerung über Softwareprotokoll (z.B. REFGO TCP Textprotokoll auf Port 10001, CANopen Objekte incl. CoE CANopen over EtherCAT, Moving Profinet Baustein, oder Kickdrive Windows-Software)
4. In MovingCap hinterlegte Python-Master-Anwendung (MovingCap CODE)

3 KONFIGURATION FLATTRACK: LANGSAMFAHRT ZWISCHEN ZWEI ENDPOSITIONEN

MovingCap flatTRACK IP-Adresse: **192.168.2.150**

Für die Auslösung der Fahrbewegung werden IN7 und IN8 benutzt. Für die Rückmeldung IO1 und IO2. Alle 4 Signale sind am Hybrid Power + Digital I/O Anschluss verfügbar und erlauben die Umsetzung von Ein-Kabel-Lösungen:

Übersicht Anschlussbelegung zu MovingCap flatTRACK Ethernet Antriebsvarianten (flatTRACK 100-650, FATtrack 200, shortTRACK 046)



HYBRID POWER + DIGITAL I/O

Pin	Bezeichnung	Beschreibung	Farbe *1)
1	IN10 (opt. HW_EN)	Digitaleingang 24 Vdc	weiß/orange
2	IO1	Digitalein/-ausgang 24 Vdc	orange
3	IN8	Digitaleingang 24 Vdc	weiß/grün
4	IO2	Digitalein/-ausgang 24 Vdc	grün
5	U_PWR	Leistungsversorgung 24-48 Vdc	blau
6	GND	GND Logik + Power	weiß
7	U_LOGIK	Logikversorgung 24 Vdc	braun
8	IN7	Digitaleingang 24 Vdc	schwarz

*1) Farben der Einzeladern bei Anschlusskabel Phoenix Contact NBC-M12MSY Hybridkabel M12 8-polig, Y-Kodierung

3.1 KONFIGURATION FAHRPROFILE ÜBER EINGANGSFUNKTIONEN

Siehe

https://movingcap.de/webmanuals/eth/i_o-funktionen.html

<https://movingcap.de/webmanuals/eth/inputfct.html>

IN Function Configurator (3511h.xx - 351Ah.xx)

IN1 IN2
 IN7 IN8

OK / Apply / Save

SubIdx	Name	Edit Value	Adjust
01h	Input Function	0400h	Timed Motion Absolute
08h	Target Position	0	Teach
06h	Positioning Velocity	20000	
07h	Acceleration	5000	

IN7 IN8

OK / Apply / Save

SubIdx	Name	Edit Value	Adjust
01h	Input Function	0400h	Timed Motion Absolute
08h	Target Position	450000	Teach
06h	Positioning Velocity	20000	
07h	Acceleration	5000	

3.2 KONFIGURATION RÜCKMELDUNGEN ÜBER AUSGANGSFUNKTIONEN

<https://movingcap.de/webmanuals/eth/outputfct.html>

MovingCap IO ← Digital Output Functions

OUT Function Configurator

OUT1 OUT2 OUT3 OUT4

OK / Apply / Save

SubIdx	Name	Edit Value	Adjust
01h	Output Function	6	Position Reached and Status OK
02h	0: last target reached, 1-10: IN1..10 target reached, 255:bit output: INx reached	7	

MovingCap IO ← Digital Output Functions

OUT Function Configurator

OUT1

OUT2

OUT3

OUT4

OK / Apply / Save

SubIdx	Name	Edit Value	Adjust
01h	Output Function	6	Position Reached and Status OK ▾
02h	0: last target reached, 1-10: IN1..10 target reached, 255:bit output: INx reached	8	
03h	Right Limit Position	0	

3.3 FAHRTEN ÜBER PROGRAMMIERUNG / TCP AUSLÖSEN:

Siehe

<https://movingcap.de/webmanuals/eth/eingaenge-simulieren.html>

OW3510,2,1796 → schreibt 3510h.2h = 0704h → Führe Funktion 4 Timed Motion mit Parametern aus IN7 aus

OW3510,2,2052 → schreibt 3510h.2h = 0804h → Führe Funktion 4 Timed Motion mit Parametern aus IN8 aus

3.4 RÜCKMELDUNGS-AUSGÄNGE ÜBER PROGRAMMIERUNG / TCP AUSLESEN

https://movingcap.de/webmanuals/eth/i_o-funktionen.html

OR60FE,1

Antwort z.B:

OR60FE,1,131072 → IO2 ist gesetzt, d.h. Rückmeldung „Antrieb hält Position aus IN8-Fahrt und ist fehlerfrei“

OR60FE,1,65536 → IO1 ist gesetzt, d.h. Rückmeldung „Antrieb hält Position aus IN7-Fahrt und ist fehlerfrei“

4 TURNTRACK 349 MIT GETRIEBE 5:1 LANGSAME 360° DREHUNG

MovingCap turnTRACK IP-Adresse: **192.168.2.151**

4.1 GRUNDEINSTELLUNGEN - GETRIEBEÜBERSETZUNG UND POSITIONSSKALIERUNG IN WINKELGRAD °

MovingCap Servo Startup (CiA 402)

Gear & Axis Configuration

Gear In turns (6091h.01h motor revolutions, for example 5)	5
Gear Out turns (6091h.02h shaft revolutions, for example 1)	1
Feed - Positions per turn (6092h.01h feed constant, for example 360)	360

4.2 KONFIGURATION FAHRPROFILE ÜBER EINGANGSFUNKTIONEN

IN Function Configurator (3511h.xx - 351Ah.xx)

IN1 IN2 IN3 IN4 IN5 IN6
 IN7 IN8 IN9 IN10

OK / Apply / Save

SubIdx	Name	Edit Value	Adjust
01h	Input Function	0600h	Timed Motion Relative
08h	Target Position	360	Teach
06h	Positioning Velocity	30	<input type="range"/>
07h	Acceleration	500	<input type="range"/>
09h	Profile deceleration	500	<input type="range"/>

IN Function Configurator (3511h.xx - 351Ah.xx)

IN1 IN2 IN3 IN4 IN5 IN6
 IN7 IN8 IN9 IN10

OK / Apply / Save

SubIdx	Name	Edit Value	Adjust
01h	Input Function	0400h	Timed Motion Absolute
08h	Target Position	0	Teach
06h	Positioning Velocity	60	<input type="range"/>
07h	Acceleration	500	<input type="range"/>
09h	Profile deceleration	500	<input type="range"/>

5 ANHANG: KONFIGURATIONSSKRIPTE FÜR MOVINGCAP CODE

Als Alternative zum manuellen Einrichten über Webseite, oder Schreiben der Parametersätze über Windows-Anwendung Kickdrive, können die Parameterlisten auch über ein vorbereitetes Python-Skript im Antrieb hinterlegt werden.

Beachten Sie die Hinweise im MovingCap-Handbuch zum Hochladen, Ausführen und permanent Speichern von Skripten unter <https://movingcap.de/webmanuals/eth/movingcapcode.html>

5.1 MOVINGCAP CODE PYTHON-SKRIPTE FÜR FLATTRACK PARAMETRIERUNG

```
#id flatTRACK_IO_left_right_Example_CODE.py 2025-05-06 oh

import sys
import mcdrive as mc

FLATTRACK_IO_LEFT_RIGHT_APP = """
050.3511h.01h,unsigned16,0 # IN1 function
050.3512h.01h,unsigned16,0 # IN2 function
050.3513h.01h,unsigned16,0 # IN3 function
050.3514h.01h,unsigned16,0 # IN4 function
050.3515h.01h,unsigned16,0 # IN5 function
050.3516h.01h,unsigned16,0 # IN6 function
050.3517h.01h,unsigned16,1024 # IN7 function
050.3517h.05h,integer32,0 # IN7 timer
050.3517h.06h,integer32,20000 # IN7 velocity
050.3517h.07h,integer32,5000 # IN7 acceleration
050.3517h.08h,integer32,0 # IN7 target pos
050.3517h.09h,integer32,0 # IN7 deceleration
050.3517h.0ah,integer32,1000 # IN7 max current / torque
050.3518h.01h,unsigned16,1024 # IN8 function
050.3518h.05h,integer32,0 # IN8 timer
050.3518h.06h,integer32,20000 # IN8 velocity
050.3518h.07h,integer32,5000 # IN8 acceleration
050.3518h.08h,integer32,450000 # IN8 target pos
050.3518h.09h,integer32,0 # IN8 deceleration
050.3518h.0ah,integer32,1000 # IN8 max current / torque
050.3519h.01h,unsigned16,0 # IN9 function
050.351ah.01h,unsigned16,0 # IN10 function
050.3611h.01h,unsigned16,6 # OUT1 function
050.3611h.02h,unsigned16,7 # OUT1 configuration
050.3611h.03h,integer32,0 # OUT1 right position
050.3611h.04h,integer32,0 # OUT1 left position
050.3611h.05h,integer32,1000 # OUT1 right pos. distance
050.3611h.06h,integer32,1000 # OUT1 left pos. distance
050.3612h.01h,unsigned16,6 # OUT2 function
050.3612h.02h,unsigned16,8 # OUT2 configuration
050.3612h.03h,integer32,0 # OUT2 right position
050.3612h.04h,integer32,0 # OUT2 left position
050.3612h.05h,integer32,1000 # OUT2 right pos. distance
050.3612h.06h,integer32,1000 # OUT2 left pos. distance
050.3613h.01h,unsigned16,0 # OUT3 function
050.3614h.01h,unsigned16,0 # OUT4 function
050.3614h.02h,unsigned16,0 # OUT4 configuration
050.6067h.00h,unsigned32,100 # Target Reached Window [user units]
050.6068h.00h,unsigned16,50 # Target Window Time [ms]
"""
```

```

def WriteObjectsFromKickdriveList(kickdriveTxtExportString):
    """
    This function takes a Kickdrive Object Editor text list and sets the parameters

    Parameters:
    kickdriveTxtExportString (str): A string containing KickDrive export data.
    Each line represents a CANopen object as comma separated list format
    "ID,Type,Value".
    IDs contain NodeId, index and subindex in format 050.3401h.03h (for example)
    Values are in decimal format. Everything after '#' is a comment / ignored.

    Returns:
    None. The function prints a message for each object written to the drive.
    """
    # Create a list of parameters
    params = [line.split(',') for line in kickdriveTxtExportString.split('\n') if
line.strip()]
    # Iterate through each parameter
    for param in params:
        # Skip lines without an ID
        if len(param) < 2:
            continue
        id = param[0].split('.')
        index = int(id[1].rstrip('h'), 16)
        subindex = int(id[2].rstrip('h'), 16)
        value = int(param[2].split('#')[0])
        mc.WriteObject(index, subindex, value)
        print ("Writing object " + hex(index) + ", " + hex(subindex) + " = " +
str(value))

WriteObjectsFromKickdriveList(FLATTRACK_IO_LEFT_RIGHT_APP)

```

5.2 MOVINGCAP CODE PYTHON-SKRIPT FÜR TURNTRACK 349 PARAMETRIERUNG

```

#id turnTRACK349_IO_360degree_Example_CODE.py 2025-05-06 oh

import sys
import mcdrive as mc

# basic gear & feed setup first
MC349_BASIC_SETUP_GEAR5TO1_360DEG = """
050.3401h.15h,unsigned16,1000 # Acc. Torque [0.1%]
050.3401h.16h,unsigned16,1000 # Dec. Torque [0.1%]
050.3401h.18h,unsigned16,1000 # Stall Torque [0.1%]
050.6073h.00h,unsigned16,1000 # Max. Current/Torque [0.1%]
050.6091h.01h,unsigned32,5 # Motor revolutions
050.6091h.02h,unsigned32,1 # Shaft revolutions
050.6092h.01h,unsigned32,360 # Feed
050.6092h.02h,unsigned32,1 # Shaft revolutions
050.6067h.00h,unsigned32,2 # Target Reached Window [user units]
050.6068h.00h,unsigned16,50 # Target Window Time [ms]
050.607dh.01h,integer32,0 # Softlimit - Min Pos [user units]
050.607dh.02h,integer32,360 # Softlimit - Max Pos [user units]
050.60f2h.00h,unsigned16,128 # Positioning option code
"""

MC349_IO_TURN349DEG_APP = """
050.3511h.01h,unsigned16,1536 # IN1 function
050.3511h.05h,integer32,0 # IN1 timer
050.3511h.06h,integer32,30 # IN1 velocity
050.3511h.07h,integer32,500 # IN1 acceleration
050.3511h.08h,integer32,360 # IN1 target pos

```

```

050.3511h.09h,integer32,500 # IN1 deceleration
050.3511h.0ah,integer32,1000 # IN1 max current / torque
050.3512h.01h,unsigned16,1024 # IN2 function
050.3512h.02h,unsigned16,0 # IN2 txPDO
050.3512h.05h,integer32,0 # IN2 timer
050.3512h.06h,integer32,60 # IN2 velocity
050.3512h.07h,integer32,500 # IN2 acceleration
050.3512h.08h,integer32,0 # IN2 target pos
050.3512h.09h,integer32,500 # IN2 deceleration
050.3512h.0ah,integer32,1000 # IN2 max current / torque
050.3513h.01h,unsigned16,0 # IN3 function
050.3514h.01h,unsigned16,0 # IN4 function
050.3611h.01h,unsigned16,6 # OUT1 function
050.3611h.02h,unsigned16,0 # OUT1 configuration
050.3611h.03h,integer32,0 # OUT1 right position
050.3611h.04h,integer32,0 # OUT1 left position
050.3611h.05h,integer32,2 # OUT1 right pos. distance
050.3611h.06h,integer32,2 # OUT1 left pos. distance
050.3612h.01h,unsigned16,0 # OUT2 function
"""

```

```

def WriteObjectsFromKickdriveList(kickdriveTxtExportString):

```

```

    """

```

```

    This function takes a Kickdrive Object Editor text list and sets the parameters

```

```

    Parameters:

```

```

    kickdriveTxtExportString (str): A string containing KickDrive export data.

```

```

    EEach line represents a CANopen object as comma separated list format

```

```

    "ID,Type,Value".

```

```

    IDs contain NodeId, index and subindex in format 050.3401h.03h (for example)

```

```

    Values are in decimal format. Everything after '#' is a comment / ignored.

```

```

    Returns:

```

```

    None. The function prints a message for each object written to the drive.

```

```

    """

```

```

    # Create a list of parameters

```

```

    params = [line.split(',') for line in kickdriveTxtExportString.split('\n') if
line.strip()]

```

```

    # Iterate through each parameter

```

```

    for param in params:

```

```

        # Skip lines without an ID

```

```

        if len(param) < 2:

```

```

            continue

```

```

        id = param[0].split('.')

```

```

        index = int(id[1].rstrip('h'), 16)

```

```

        subindex = int(id[2].rstrip('h'), 16)

```

```

        value = int(param[2].split('#')[0])

```

```

        mc.WriteObject(index, subindex, value)

```

```

        print ("Writing object " + hex(index) + ", " + hex(subindex) + " = " +
str(value))

```

```

# the order is important here:

```

```

# first set up the gear ratio & position scaling

```

```

WriteObjectsFromKickdriveList(MC349_BASIC_SETUP_GEAR5TO1_360DEG)

```

```

# Now write the IO configuration with target positions scaled in degrees

```

```

WriteObjectsFromKickdriveList(MC349_IO_TURN349DEG_APP)

```