# Merkzettel, Reminder Sheet

Author: Bernd Heufelder

14. Juni 2019


## Inhaltsverzeichnis

# Teil I
# Pragmatisches

## 1 Verbindungsaufbau mit der Maschine

- PuttY starten
- Maschinenname eingeben, z.B. fc-4438
- Benutzername: ses
  Passwort: dmsu
- Mushell starten
- In der Mushell: Ultimet_startETNE eingeben um Maschine "freizuschalten"
- Im Port-File muss die Maschine definiert sein, dann kann übers Dropdown-Menü die Verbindung/Maschine ausgewählt werden um eine Verbindung herzustellen.

Beispiele für *port.properties* Einträge verschiedener Maschinen. Die Grünen Einträge sind immer gleich und rot ist variabel.

- **FlipChip (fc)/8k8**

  port.8.name=fc-4169 — Beliebiger Name, diser wird im Dropdown Menü angezeigt
  port.8.driver=etn://fc-4169:1160 — Maschinenname, muss mit der Beschilderung auf der Maschine übereinstimmen
  port.8.protocol=ETB
  port.8.accept.0.driver=ETN:0
  port.8.start=auto

- **Lupus Testunit**

  port.8.name=Lupus testunit-401
  port.8.driver=etn://testunit-401:1160
  port.8.protocol=ETB
  port.8.accept.0.driver=ETN:0
  port.8.start=auto

- **Evo/2k2**

  port.8.name=EVO 2-modulig evo-005089 — Beliebiger Name, diser wird im Dropdown Menü angezeigt
  port.8.driver=etn://evo-005089:1160
  port.8.protocol=ETB
  port.8.accept.0.driver=ETN:0
  port.8.start=auto

- **Beliebige Maschine**

```
port.8.name=VCB-02_ssd                  Als Beispiel die VCB-Maschine
port.8.driver=etn://10.10.18.109:1160   Statt Maschinenname kann auch direkt über die
                                        IP-Adresse verbunden werden
port.8.protocol=ETB
port.8.accept.0.driver=ETN:0
port.8.start=auto
```

# 2 Maschineninbetriebnahme

Prinzipiell wird das Phasing aller Achsen durchgeführt. Überprüfung aller Encoder. Können alle Achsen gehomed werden? Die Regelung ist bereits getuned.

## 2.1 Neue Maschine inbetriebnehmen

Noch nicht fertig ...

- Achsen mit neuen Motoren, Encodern oder sonstige Änderungen, die auf die Regelung einspielen neu Inbetriebnehmen (NEW SETTING, ComET-Tools)
- Vorgänger Registerfiles testen falls es keine groben Änderungen sind
- XML-File erweitern
- Servo-File-Packer
- Release New Servo-Bundle
- ...

# 3 ComET Command Prompt

## 3.1 Hilfreiche Kommandos

Rufzeichen ! für alle Achsen und .$x$ führt das Kommando nur für die aktuell ausgewählte Achse aus.

| | |
|---|---|
| **name.x** | als Rückgabewert erhält man die Benennung der Achse mit welchler man gerade verbunden ist |
| **ver.x** | Versionsnummer der gerade gewählten Achse (e.g. 0x316xxxx) -¿ die ersten 3 Ziffern sind die Versionsnummer und die darauf folgenden kennzeichnen die Revision |

## 3.2 Undocumented ETEL Parameters

| | |
|---|---|
| **C9** | MLTI Multiplikator (2=200$\mu s$,3=200$\mu s$,4=800$\mu s$) |
| **K260** | Widerstand terminal to terminal in $mH$ |
| **K261** | Induktivität terminal to terminal in $\Omega$ |

# 4 Maschinenbedienung

**Kommandos und Tastenkombinationen**

| | |
|---|---|
| **Strg+Shift+F2** | Run Application (z.b. zum Start vom Indel-INCO-Explorer )<br><br>• **konquer:** Öffnet File-Explorer des Maschinenpc's<br>+ Motion-Files (Seq., Reg., ...) sind im Pfad *Datacon/share/config/-Servotools* |

# 5 Servo Tools

Erweiterung der Servotool durch Eingabe von *servoToolsDialogExpertMode 2* in der Mushell. Z.B. für den Fall wenn sich Register-Files nicht vom PC auf den Controller updaten lassen, kann dies über die Servotools im Expert-Mode manuell geschehen.

expertMode -¿ Force overwrite

Terminal -¿ Registerwerte auslesen

# 6 Servo-Bundle

Ist ein .zip-File, in welchem sich das XML-File, sämtliche benötigte Register-Files, Sequenzen und Firmware-Files befinden. Auf jeder Maschine befindet sich das Gesamte zip-File. Ein Servobundle kann für alle Maschinenplatformen verwendet werden, da sich die besagten Files für alle Maschinen darin befinden. Jedes Servobundle hat ein version_info.txt, in welchem die Änderungen der einzelnen Versionen gelistet sind.

Sequenzen müssen kompiliert werden bevor sie auf die Maschine überspielt werden. Um ein Kompilieren auf der Maschine zu umgehen wurde der Offline-Sequence-Compiler entworfen. Damit werden die .txt Files in .eseq Files umgewandelt. Im Servo-Bundle befinden sich nur die kompilierten Versionen der Sequenzen.

## 6.1 XML-File

Im XML-File sind für jede einzelne Maschine die Folgenden Zuweisungen für jede Achse definiert

- Auf welchem Controller-Typ hängt die Achse
- Welche Controller-ID hat die Achse
- Mit welcher Firmware soll die Achse laufen
- Welche Sequence läuft für die Achse

- Welches Register-File ist der Achse zugewiesen
- Welches Homing und Phasing soll die Achse verwenden

# 7 Firmenspezifisch

## 7.1 Firmenstruktur

### 7.1.1 Software

**BIB** Softwareabteilung, besthend aus OMA und Betriebssystem für die Maschine (Linux).

**OMA**
Hardwareabstraktion auf Sofwareebene; Applikation erhält möglichst einheitliche Objekte für unterschiedliche Hardware. Die Applikation verwendet diese Hardware dann entsprechend und programmiert damit einen Prozessablauf. Kommunikation der HW über CAN-Bus und PC. OMA ist nicht echtzeitfähig (z.b. Rückmeldung nach Bewegungsende). Bildverarbeitung ist eigenständig und nicht Teil der OMA. Auch wird durch die OMA die Topologie der Hardware, also welche HW an welchem Knoten hängt, verschleiert.

Link zur Doku: https://confluence.besi.com/display/MA/interferometric+systems+RENISHAW

**Applikation** Programiert einen Prozessablauf und seine Bewegungen. Ebenso wird die Benutzeroberfläche auf der Maschine von der Applikation bereitgestellt.

## 7.2 Ansprechpersonen

| | |
|---|---|
| **Simon Holzknecht** | Chef Software Tester, Ansprechpartner zum Reservieren von Maschinen |
| **Michael Kirchler** | Zuständiger System Engineer für 2k2, Umbauarbeiten bei Evo bzw. akutell vorhandene customized Versionen |
| **Bernd Jandl** | Manager Machine Integration Software |
| **Christian Knoll** | Maschinenplanung - Manager System Engineering 2k2 |

# Teil II
# Projektspezifisch

## 8 Common Control Platform (CCP)

### 8.1 Hardware

**Indelprodukte**

- **COP-MAS2:** Das CPU-Board ist die zentrale Kontrolleinheit einer Indel Steuerung. Das CPU-Board koordiniert die Achsen und I/O-Systeme in Echtzeit. Alle Indel CPU-Boards werden mit dem Indel Echtzeitbetriebssystem INOS betrieben, dadurch ist es möglich, die gesamte Maschinenapplikation direkt auf dem CPU-Board zu implementieren. Das INOS Betriebssystem übernimmt die Abstraktion der gesamten Hardware und ermöglicht es damit, die gleiche Applikationssoftware auf beliebigen Indel CPU-Boards auszuführen.
- **SAC:** Endstufe zur Motoransteuerung mit integriertem Netzteil
- **SAM4:** High-End Alternative zum COP-MAS mit einer möglichen closed-loop frequenz bis $64kHz$.

**Besi-Produkte:**

- **Hammer S3 mit Connection-Board:** Äquivalente Endstufe zum SAC von ESEC selbstentwickelt.
- **UCB:** Notfalleinheit falls COP-MAS ausfällt übernimmt das UCB-Board die Steuerung

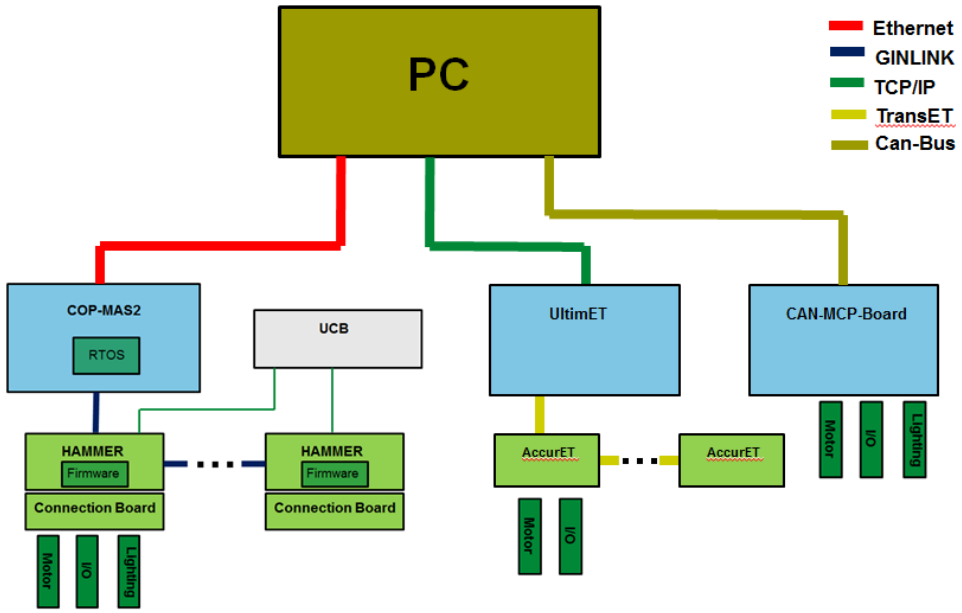


Abbildung 1: Gemischte Struktur aus Indel und ETEL Controller

### 8.1.1 Kompatibilität Hammer und Connection-Board

Beim Starten werden die ID's der Hammer-Firmware und die ID des Connection-Boards (CB) verglichen. Die ID des CB's ist im Flashspeicher auf dem Board gespeichert. Das Board T128 benötigt dazu die Firmware N128 auf dem Hammer. Die Identification und Variant kann über

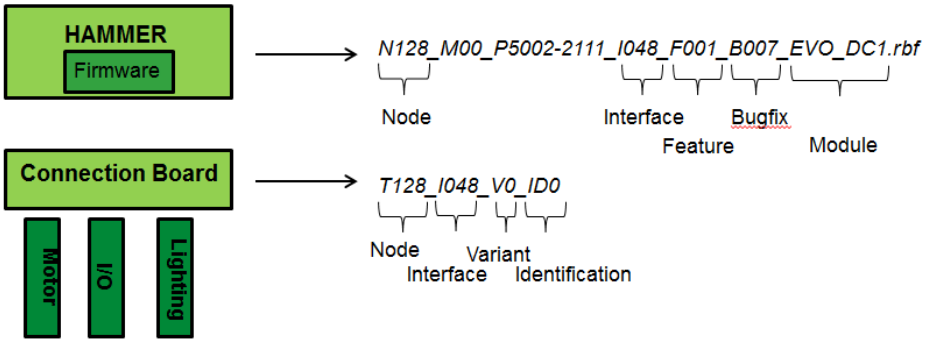


Abbildung 2: ID Übereinstimmung

Dip-Switches am CB geändert werden.

## 8.2 Konfiguration

Die Konfiguration eines Hammer-Boards geschieht anhand von zwei *.dt2* Files, die auf dem MaschinenPC liegen.
Im Source-Code ist der Pfad zu diesen Files hinterlegt. Diese *.dt2* Files sind von Indel und werden direkt vom COP-MAS gelsen.

Um den Konfigurationsaufwand zu erleichtern werden die Beiden *.dt2* Files von Besi aus mehreren *.dtx* Files erzeugt. Es wird also ein neues Front-End geschaffen, das handlicher zu ändern sein wird.

## 8.3 File-Schlacht

### 8.3.1 _Objects

Dieses File wird automatisch von der Software erstellt, deshalb kann es nicht so einfach ins AxisDefinition integriert werden.

### 8.3.2 AxisDefinition

Hier wird z.b. beim MHPusher auf des Registerfile und die Speedsets dieser Achse verwiesen. Diesen liegen im Subfolder *./Axis*.

### 8.3.3 ObjectsDefinitions

Es gibt 3 verschiedene Typen an ObjetsDefinition-files:

- ObjectsDefinitions
- ObjectsDefinitions-MotX-DC

```
[SLAVE]
_Ref = "T129_I064_MH"
#Name = "TestUnit"
Name = "T129_I064_MH"
Identification = 0
Variant = 0

# ---

[AXIS-SERVO]
_Ref = "MHPusher"
Name = "MHPusher"
Identification = 0
Variant = 0

[AXIS-SERVO]
_Ref = "MHY"
Name = "MHY"
Identification = 0
Variant = 0

[AXIS-SERVO]
_Ref = "MHZ"
Name = "MHZ"
Identification = 0
Variant = 0

[AXIS-DC]
_Ref = "MHBeltIn"                    #Reference to the generic motor port
Name = "MHBeltIn"       #Identification name for the required motor
Identification = 0
Variant = 0

[AXIS-DC]
_Ref = "MHBeltOut"                   #Reference to the generic motor port
Name = "MHBeltOut"      #Identification name for the required motor
Identification = 0
Variant = 0
```

Abbildung 3:

- ObjectsDefinitions-MotX-Servo
- ObjectsDefinitions-MotX-Stepper

In den MotX-Files sind die auf den Motor bezogenen INPUT und OUTPUT Bits den jeweiligen Variablen zugeordnet, wie sie im IO_Sheet hinterlegt sind. Im *ObjectsDefinitions* sind die Bits für sonstige IO-Signale definiert.

```
#####  Axes  #####

[AXIS-SERVO_REDEF]
_Ref = "Mot1Servo"          #Reference to the generic motor port
_Id = "MHY"              #Identification name for the required motor
_Files = "axis/MHY.dt2, axis/_machine_11_MODULE.MHY.lua"
#!RequiredCurControlVersion = "1.0"
!AxisType = "linear"      # "linear|rotation"

[AXIS-SERVO_REDEF]
_Ref = "Mot2Servo"          #Reference to the generic motor port
_Id = "MHZ"              #Identification name for the required motor
_Files = "axis/MHZ.dt2, axis/_machine_11_MODULE.MHZ.lua"
#!RequiredCurControlVersion = "1.0"
!AxisType = "linear"      # "linear|rotation"

[AXIS-SERVO_REDEF]
_Ref = "Mot3Servo"              #Reference to the generic motor port
_Id = "MHPusher"                #Identification name for the required motor
_Files = "axis/MHPusher.dt2, axis/_machine_11_MODULE.MHPusher.lua"
#!RequiredCurControlVersion = "1.0"
!AxisType = "linear"          # "linear|rotation"

[AXIS-DC_REDEF]
_Ref = "Mot4Dc"
_Id = "MHBeltIn"                #Identification name for the required motor
_Files = "axis/MHBeltIn.dt2, axis/_machine_11_MODULE.MHBeltIn.lua"
#!RequiredCurControlVersion = "1.0"
!AxisType = "linear"          # "linear|rotation"

[AXIS-DC_REDEF]
_Ref = "Mot5Dc"
_Id = "MHBeltOut"               #Identification name for the required motor
_Files = "axis/MHBeltOut.dt2, axis/_machine_11_MODULE.MHBeltOut.lua"
#!RequiredCurControlVersion = "1.0"
!AxisType = "linear"          # "linear|rotation"
```

Abbildung 4:

```
######################
#####    Axis    #####
######################

[AXIS-DC_DEF]
_Id = "Mot3Dc"
PortNumber = 2

######################
#####   Input    #####
######################

[POS_DEF]
_Id = "Mot3DcEncPos"
_Parent = "Mot3Dc"
PortNumber = 2
StartBit = 0
Bits = 14

[INP_DEF]
_Id = "Mot3DcInit"
_Parent = "Mot3DC"
PortNumber = 2
Options = 0x8
BitNumber = 30
!Pin = ""

######################
#####   Output   #####
######################

[DAC_DEF]
_Id = "Mot3DcMotVolt"
_Parent = "Mot3Dc"
Unit = "%"
PortNumber = 2
Options = 0x0
StartBit = 0
Bits = 16
Characteristics = 0x0000003C
Offset = 0.0
Gain = 327.68
Minimum = -100 #Apeak
Maximum = 100 #Apeak

[OUT_DEF]
_Id = "Mot3DcMotEn"
_Parent = "Mot3Dc"
PortNumber = 2
BitNumber = 31
Options = 0x8
```

Abbildung 5:

# Teil III
# Theorie

## 9 Encoder

### 9.1 SinCos-Encoder

Im Abtastkopf leuchtet eine Infrarot-LED seitlich auf eine Maßverkörperung mit reflektierenden und nicht-reflektierenden Bereichen. Das Licht wird von den reflektierenden Bereichen durch ein transparentes Phasengitter zurückreflektiert.
In der Erkennungsebene des Abtastkopfes entstehen hierdurch sinusförmige Interferenzstreifen. Das optische System integriert aus mehreren Teilungsperioden einen Durchschnittswert und filtert Signale heraus, die nicht von der Maßverkörperung reflektiert werden. Dadurch wird die Stabilität des Signals auch dann gewährleistet, wenn die Maßverkörperung verschmutzt oder leicht beschädigt ist.
Durch die einzigartige Optikkonstruktion des Abtastkopfes sowie die Funktionen Auto Gain Control (AGC) und Auto Offset Control (AOC) des REF Interface sind geringe Kurzzeitfehler gewährleistet, die typischerweise einen zyklischen Fehler (SDE) von weniger als +/-0,05 um ergeben.
Das REF Interface verfügt über eine Einstell-LED, die durch ein grünes Licht das Erreichen der optimalen Einstellung anzeigt.
Eine Referenzmarke oder ein einfacher Endschalter sind bei diesem Abtastkopf erhältlich. Die Referenzmarke dient als Referenz oder Nullpunkt für den Abtastkopf, während der Endschalter das Ende des Verfahrbereiches signalisiert.

### 9.2 EnDat Encoder

Es gibt ein grobes Lineal in welchem die absolute Position kodiert ist und ein feineres für die inkrementalen Schritte (feinschritt). Die Digitalisierung des Positionswertes erfolgt oft direkt am Encoderkopf.

## 10 Bussysteme

### 10.1 EtherCAT

EtherCAT (Ethernet for Control Automation Technology) ist ein von der Firma **Beckhoff Automation** initiiertes Echtzeit-Ethernet.

EtherCAT unterscheidet sich wesentlich von anderen Industrial Ethernet Lösungen. Während bei diesen der vom Master versendete Standard Ethernet Frame (gemäß IEEE 802.3) in jeder Anschaltung zunächst empfangen, dann interpretiert und die Prozessdaten weiterkopiert werden, entnehmen beim EtherCAT die EtherCAT Slave-Geräte die für sie bestimmten Daten, während das Telegramm das Gerät durchläuft. Ebenso werden Eingangsdaten im Durchlauf in das Telegramm eingefügt.

Zykluszeiten $\leq 100\mu s$ mit niedrigem Jitter.

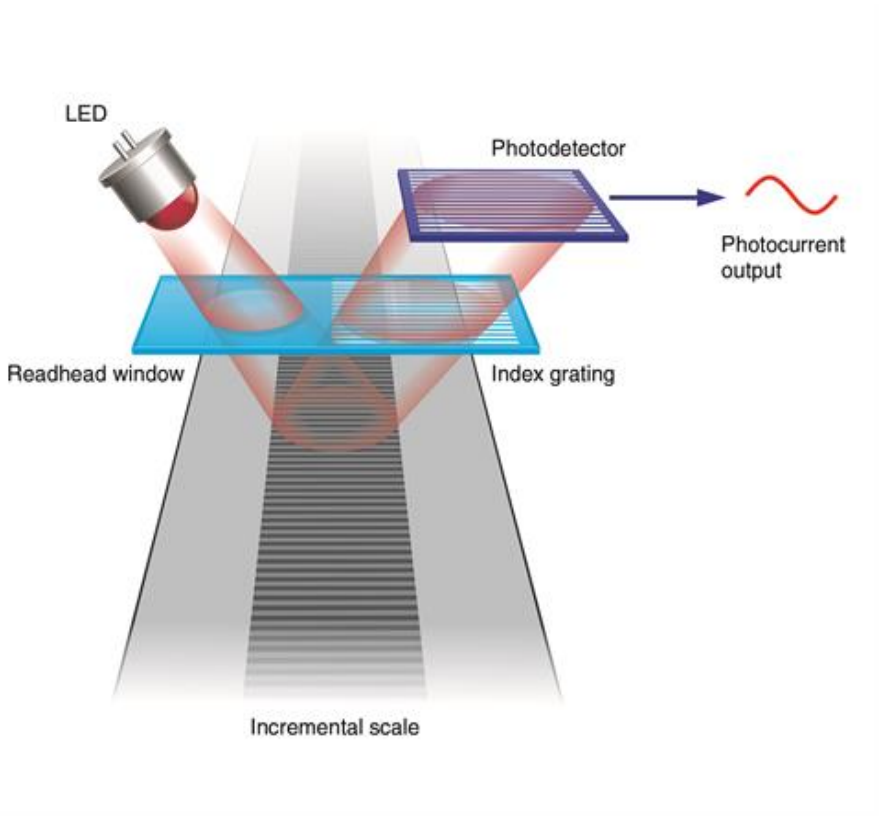


Abbildung 6: SinCos Encoder Hardwareaufbau des Encoderkopfs mit Lineal. Das Index Grating ist nur eine transparente Platte.

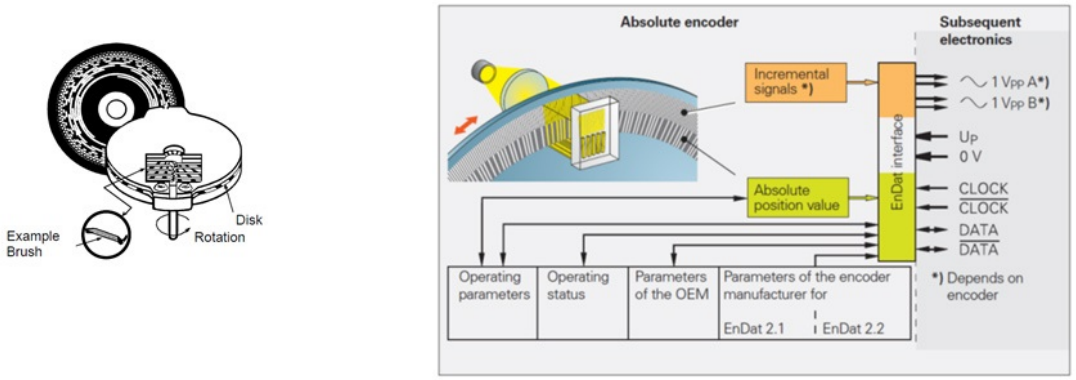


Abbildung 7: Absolutencoder

## 10.2 ETEL Systeme

### 10.2.1 TransnET

Für Echtzeit-Anwendungen mit mehreren Achsen. Die TransnET Synchronisation mit Jitter im Nanosekunden-Bereich ist die richtige Lösung für deterministische und zuverlässige Kommunikation innerhalb von Maschinen.

### 10.2.2 Turbo-ETEL-Bus

Multi-axis fast bus mit 100Mbps

## 10.3 CAN-Bus

Der CAN-Bus (Controller Area Network) ist ein serielles Bussystem und gehört zu den Feldbussen. Feldbus ist ein Bussystem zur Kommunikation für Sensoren, Aktoren in einer Anlage bzw. Maschine.

**Funktionsprinzip**: Der CAN-Bus arbeitet nach dem „Multi-Master-Prinzip“ d. h., er verbindet mehrere gleichberechtigte Steuergeräte. Ein CSMA/CR-Verfahren löst Kollisionen (gleichzeitiger Buszugriff) auf, ohne dass die gewinnende, höher priorisierte Nachricht beschädigt wird. Der CAN-Bus ist nicht echtzeitfähig, da nicht garantiert werden kann ob der Bus zu gegebenen Zeitpunkt gerade frei ist.

## 10.4 Ginlink

Busverbindung vom COPMAS zum Hammer.