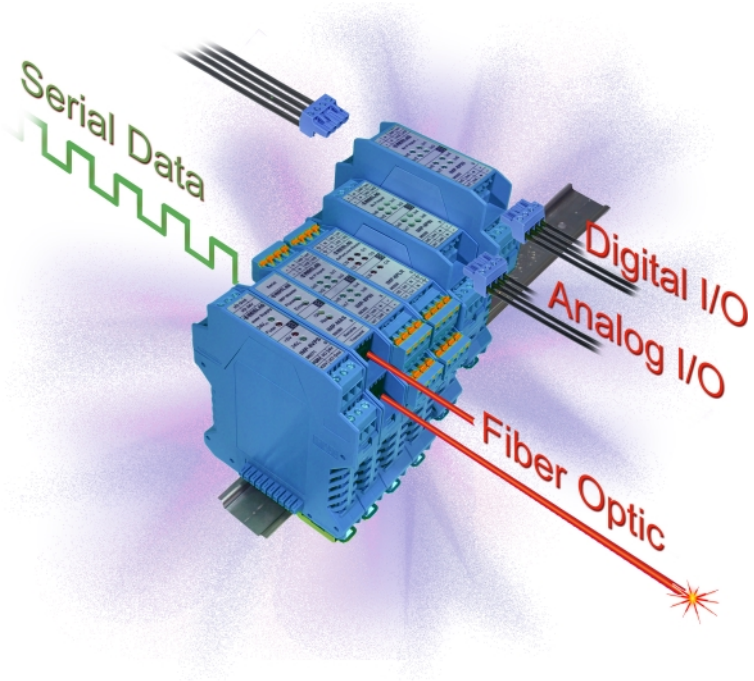
10 Schritte zur laufenden IMP-Steuerung



Version 1.17

Copyright März 2004 by Indel AG, All Rights Reserved

Rev 1.0	10.06.2001-M. Koch	erster Entwurf, erste ausgelieferte Version
Rev 1.1	14.06.2001-M. Koch	Überarbeitet: Achsen, Inco-Server, Not-System, analoge I/Os, pos-channels, Burn, Load
Rev 1.11	24.01.2002-M. Koch	INCO-Explorer
Rev 1.12	19.02.2002-M. Koch	Kapitel 10.12
Rev 1.13	17.06.2002-M. Koch	Stream File .str löschen, Laufzeiten, Zahlenformate Kap. 10. 2, Konfiguration
Rev 1.14	13.09.2002-M. Koch	Zahlenformate Kap. 10. 2 erweitert
Rev 1.15	08.01.2003-M. Koch	Neues IMP-5VPSa
Rev 1.17	22.06.2004-M. Koch	

Inhaltsverzeichnis

IMP - Indel Modular Peripherie

Herzlichen Dank, dass Sie sich für unsere multifunktionale Kleinsteuerung IMP - Indel Modular Peripherie entschieden haben !

Dieses Manual soll Ihnen helfen, möglichst schnell den Einstieg in die Programmierung und den Umgang mit der IMP-Steuerung zu finden. Die ersten 10 Schritte führen nicht nur zu einem ersten lauffähigen Projekt, sondern hoffentlich auch zur erfolgreichen Lösung Ihrer Automatisierungs-Applikation.

Sollten sich Ihre genialen Ideen zur Lösung Ihres Automatisierungs-Problemes nicht oder nur widerspenstig in Tasks für den IMP-Master transferieren lassen, stehen Ihnen unsere erfahrenen Entwicklungsingenieure zur Seite.

Wir legen grossen Wert auf eine gute Zusammenarbeit mit unseren Kunden und bemühen uns entstehende Probleme innert nützlicher Frist zu lösen. Schliesslich haben wir die gesamte Hard- und Software selber entwickelt und können Ihnen dadurch optimalen Support liefern.

Tel. ++41 / (0)44 956 20 00
E-Mail info@indel.ch

1. Hardware Installation

1.2 Netzanschluss

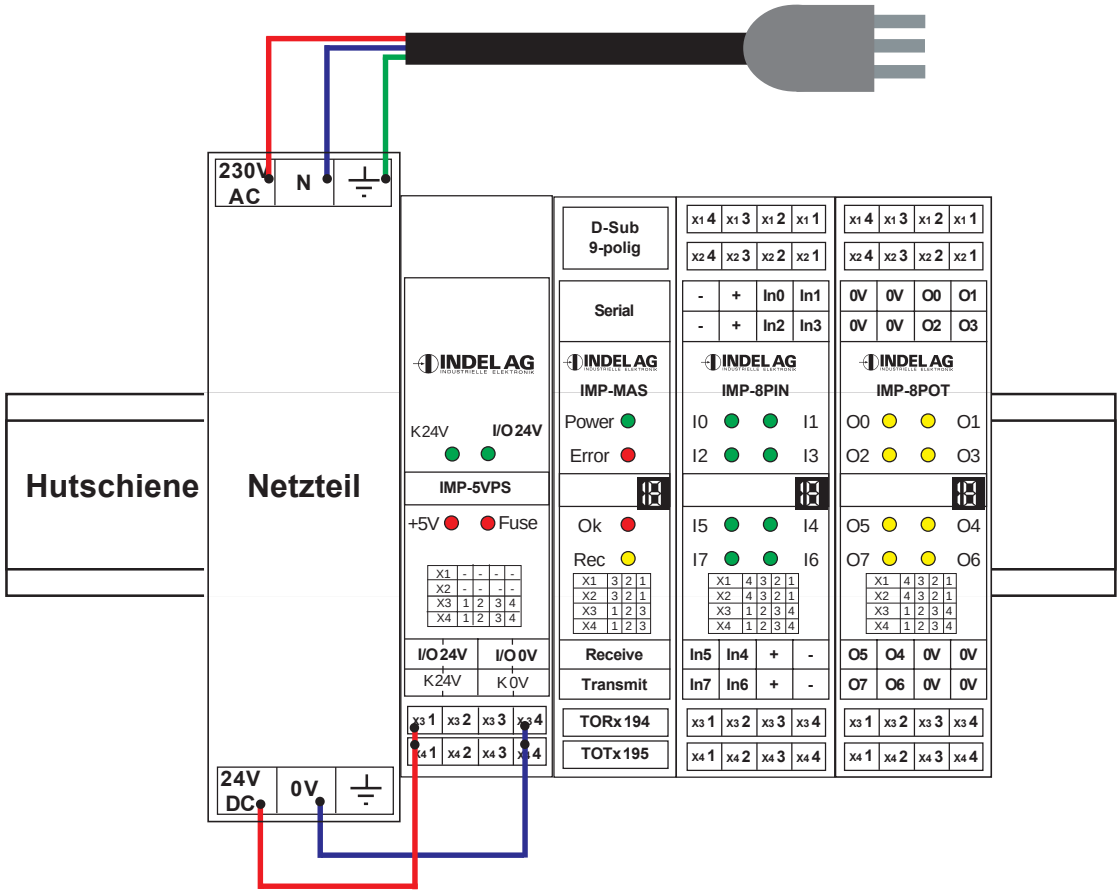


Abb. 1.2.1 Netzanschluss

1.3 Drehschalter Stellung

Jedes IMP-Modul ist mit einem Drehschalter bestückt. Die IMP-Module sind in verschiedene Gruppen mit verschiedenen Karten-Typen eingeteilt. Alle Module des gleichen Typs müssen fortlaufend durchnummeriert werden. Werden mehrere Module des gleichen Typs mit der gleichen Adresse am IMP-Master betrieben, entstehen Adresskonflikte, und die Module können nicht korrekt angesprochen werden. Die Reihenfolge der Module spielt keine Rolle.

Das Modul `IMP-8PIN` gehört zum Karten-Typ `digitale Eingänge`
Das Modul `IMP-8POT` gehört zum Karten-Typ `digitale Ausgänge`

Beide Module müssen für das nachfolgende Beispiel auf Adresse Null eingestellt werden.

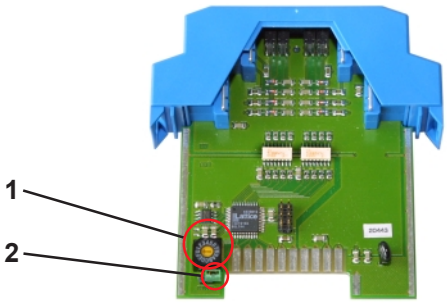


Abb. 1.3.1 Adress Wahlschalter, Drehschalter

1 Adress Wahlschalter für Adresse 0 ... 15
2 Adress Wahl-Jumper für Adresse 16 ... 31

> Für weitere Hilfe siehe CD-ROM, File:
> - `d:\IMP\Hardware\Deutsch\adressierung.pdf`

1.4 Anschluss Belegung

Weitere Informationen zu den einzelnen Modulen finden Sie in der entsprechenden Modul-Beschreibung auf der CD-ROM oder im Internet auf unserer Homepage http://www.indel.ch/

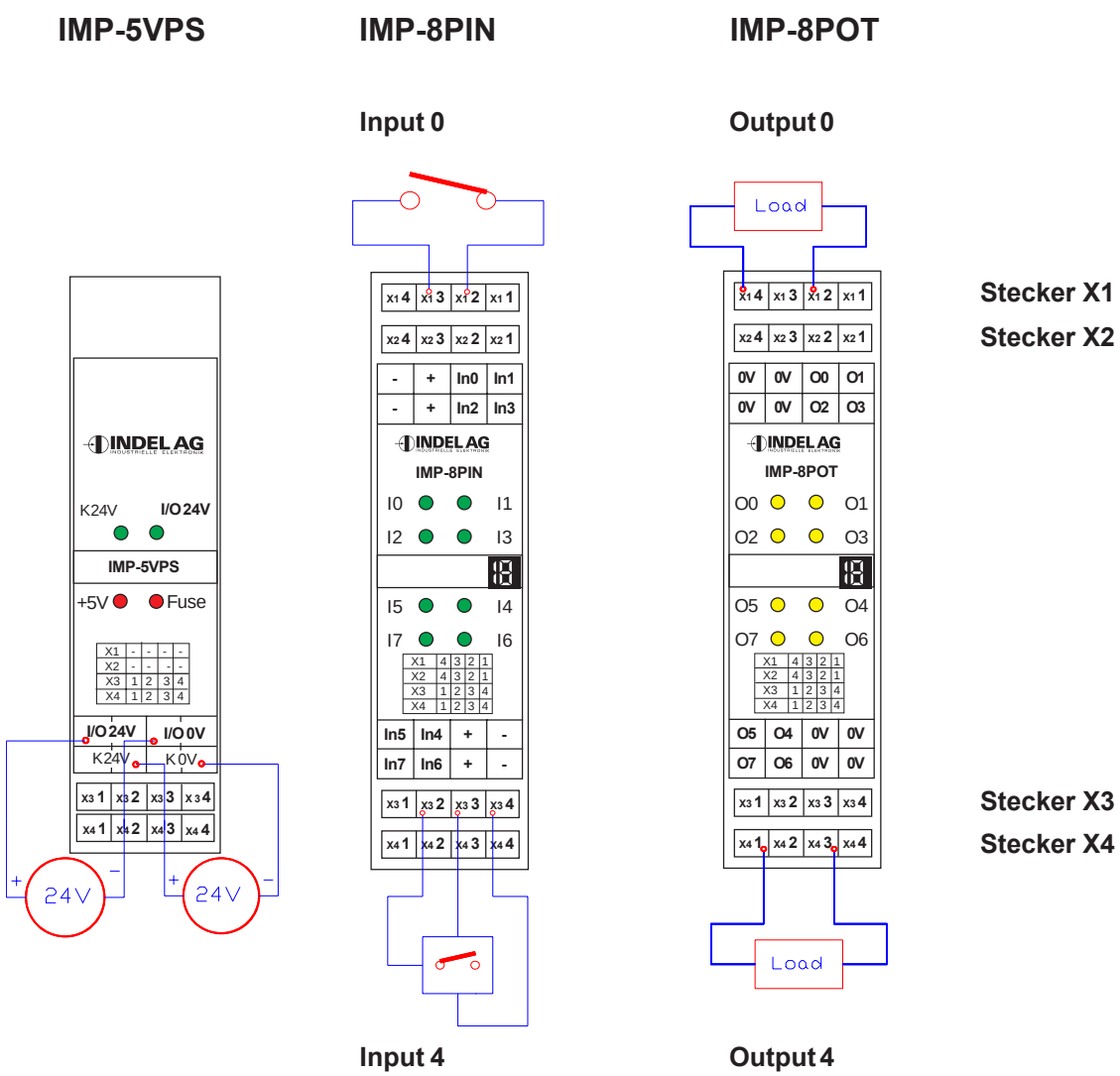


Abb. 1.4.1 Anschlussbelegungen

2. Software Installation

2.1 Installation ab CD-ROM

1

Benützten Sie die Vollversion der Entwicklungsumgebung (CD-ROM "Indel-Steuerungssysteme"). **Starten** Sie die Installation aus dem Verzeichnis auf der CD-ROM:

```
d:\Setup\imd\IMDSetup.exe"
```

Die Installation beinhaltet:

- IMD-Indel Master Desk: Entwicklungsumgebung mit Editor, Cross-Compiler, Debuger, usw
- INCO-Server: Kommunikation zwischen Targets und Visualisierung, netzwerkfähig, PC-Netzwerk, Modem, Internet
- Diverse Tools: Variablen Logger, Variablen Explorer, ACS-Show, Win-Show, Memory-Dump, usw.

2

Wählen Sie die **Sprache** und anschliessend das **Verzeichnis** in dem Sie Indel Master Desk installieren wollen. Default Verzeichnis: `"c:\IMD\"`

Abb 2.1.1 IMD-Setup

Wenn Sie bei "Installationsart" "Benutzer" gewählt haben, können Sie folgende **Komponenten** nach Belieben installieren. Installieren Sie alle Komponenten!

3

Abb 2.1.2 Komponenten

Komponenten:

- ☑ *Indel Master Desk / INCO-Server*
  Entwicklungsumgebung, Kommunikation zwischen Targets und PC-Oberfläche, netzwerkfähig

- ☑ *Tools*
  Variablen Logger, Variablen Explorer, ACS-Show, Win-Show, Memory-Dump, usw.

- ☑ *Samples / Programming*
  Diverse Beispiel-Programme

"Standard" installiert alle Komponenten.

Wählen Sie ein Zielsystem (Target):

Abb 2.1.3 Komponenten

Wählen Sie als Target: **"INCO-SIO"**

⦿ Remote-PC	Beliebiges Target, befindet sich in einem Netzwerk	
⦿ Power-PC Master	Target ist PowerPC-Master:	INFO-PPC, INFO-MASi, lokal
⦿ PCI-Master	Target ist PCI-Master:	INFO-PCI, lokal
⦿ INFO-Master	Target ist INFO-Master:	INFO-PCM, lokal
⦿ PC-Master	Target ist Kupferfeldbusmaster:	EXT-MAS
⦿ IPS	Target ist IPS-System, via serielle Schnittstelle	
⦿ INCO-Sio	Target ist an serieller Schnittstelle:	
	– **IMP-Master:**	**IMP-MAS**
	– Stand Alone Master:	INFO-SAM
	– AC-Servo Regler:	INFO-ACSr

5

Sie haben später die Möglichkeit weitere Targets zu definieren. Benützen Sie dazu den Registrierungs-Editor "`INCO-Registry`" in der Programmauswahl.

Wählen Sie den Target Namen:

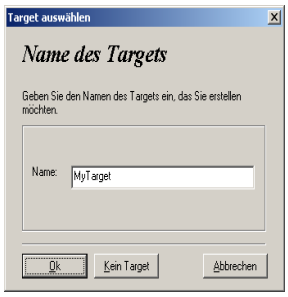


Abb 2.1.4 Target Namen

Geben Sie einen Namen für das Zielsystem an. Ein beliebiger Name ohne Leer- und Sonderzeichen ist erlaubt. Der Name sollte sich auf ihr Projekt (z.B. TestProject) und nicht auf den Kartentyp beziehen. Bei der Installation eines Remote-PC's ist darauf zu achten, dass dieser Name mit dem Target-Namen der Server-Registrierung übereinstimmt.

2.2 Registrierungs-Editor

6

Beantworten Sie die Abfrage "Wollen Sie nun den Registrierungseditor starten?" mit "Ja".

Abb 2.2.1

Bei Targets, die über die serielle Schnittstelle angeschlossen werden, wie IMP-Master, Servo Regler (INFO-HCSr), Stand Alone Master (INFO-SAM), muss der **COM-Port** angegeben werden. Alle übrigen Einträge im Verzeichnis Setup müssen nicht verändert werden. Wählen Sie einen freien COM-Port, der nicht von einem anderen Gerät benützt wird.

7

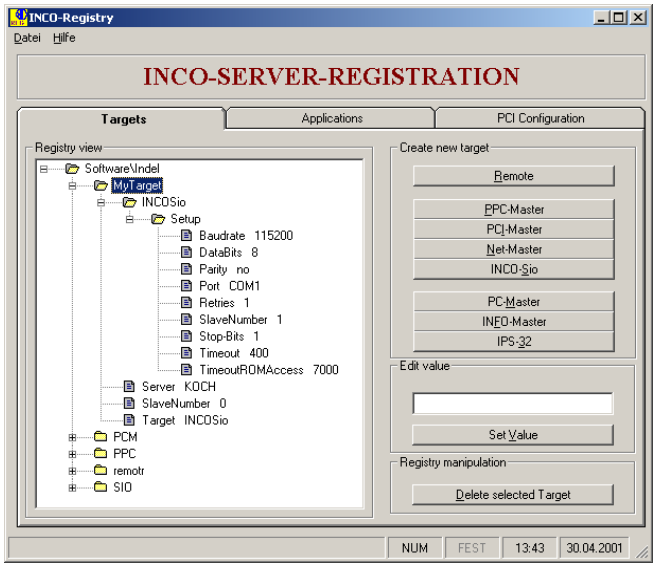


Abb 2.2.2

Registrierungs-Editor: Bei PC-Master-Karten wie INFO-PPC, PC-MAS, PCMAS-32 muss die Adresse im Verzeichnis "Setup" mit der **Adresse**, die auf der Masterkarte eingestellt ist, übereinstimmen. Bei PowerPC kann zwischen Adresse "CE00" bis "D800" gewählt werden! Nur gerade Adressen verwenden (CE, D0, D2, D4...) !

Sie können zusätzliche Targets definieren.

Die Registrierungs-Einträge werden in der Windows-Registry abgelegt. Sämtliche Indel Software Tools greifen auf die Windows-Registry zu.

8

PC neu booten. Nach dem booten erscheint das Symbol für den INCO-Server rechts in der Taskleiste.

3. IMP mit PC verbinden

3.1 Verbindung mit SIO

Verwenden Sie ein 9-poliges Null-Modem Kabel für die Verbindung zwischen IMP-Master und PC.
Distrelec: AT Link-Datenkabel, Länge 3m 67 27 88

Der COM-Port, der in der Registry angegeben worden ist darf von keinem anderen Gerät belegt werden. Z.B. PDA-Dockingstation, Modem, Maus, usw.

Bei jeder Änderung in der INCO-Registry muss der INCO-Server neu gestartet werden, ansonsten werden die Änderungen nicht übernommen!

Bei der Auslieferung ist im IMP-Master ein Notsystem geladen. Die rote Error-LED blinkt.

3.2 Verbindung mit Ethernet

Falls der IMP-MAS2 direkt mit dem PC verbunden werden soll, muss ein gekreuztes Kabel verwendet werden.

Über die SIO kann dem IMP-MAS2 eine IP Nummer eingestellt werden. Die IP muss bei Ihrem Netzwerk-Administrator bezogen werden. (Es darf niemals die gleiche IP-Adresse mehrmals verwendet werden im gleichen Netzwerk.)

Danach muss der IMP-MAS2 als Netzwerk-Target registriert werden. Siehe dazu Kapitel 2. Software Installation.

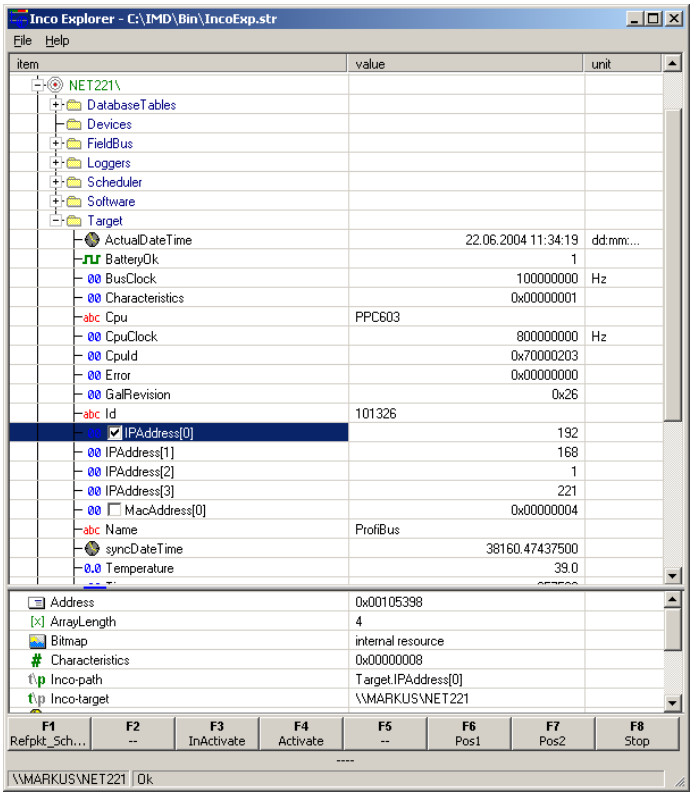


Abb 3.2.1 IP-Adresse

3.3 Verbindung testen

Verwenden Sie das Tool `INCO-Explorer` um die Verbindung zwischen IMP-Master und PC zu testen. Wählen sie das Target aus, das Sie für den IMP-Knoten registriert haben. Das Ergebnis sollte wie folgt aussehen:

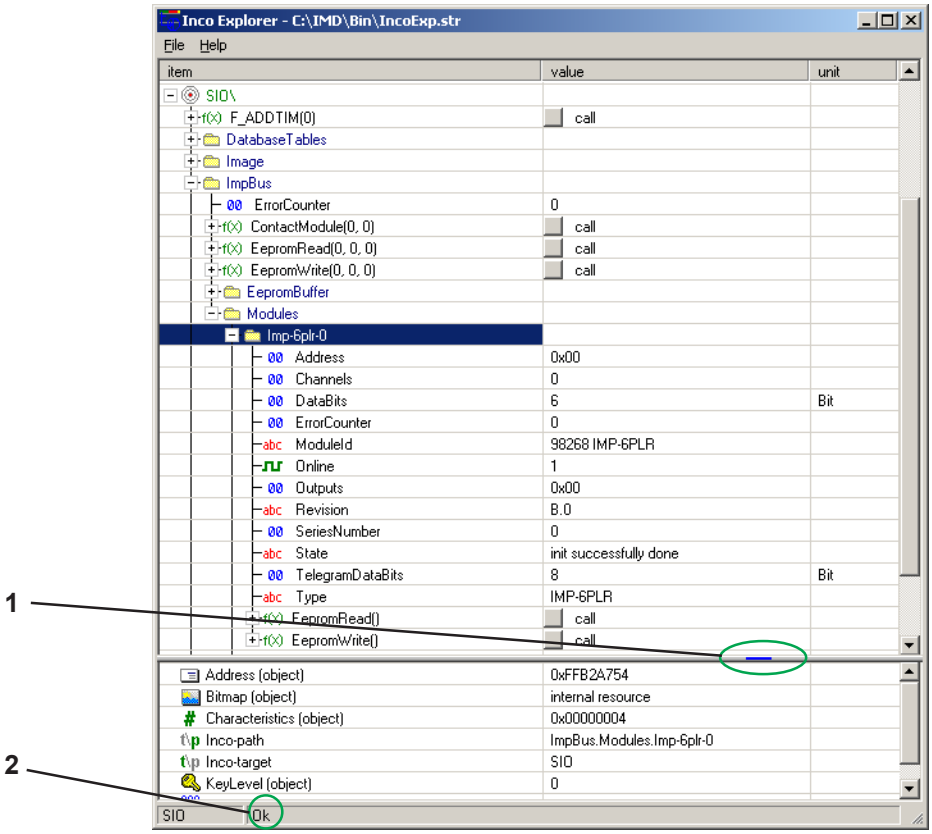


Abb 3.3.1 INCO-Explorer

Der Rollbalken **1** läuft ständig von links nach rechts und wieder zurück, sobald das Target läuft.
Im Fenster **2** werden Fehler aufgezeigt. Schlagen Sie im *Anhang A.1 Fehlermeldungen* nach um die Ursache zu finden.

4. Beispiel Projekt öffnen

Starten Sie die integrierte Entwicklungsumgebung IMD - Indel Master Desk:
`Start > Programme > Indel Master Desk > Indel Master Desk.`

Unter `File > Open` öffnen Sie das Beispiel im Verzeichnis:
`c:\IMD\Program\INOS\Samples\IMP_MAS\FirstStep\IMP.mpd`

Ein Doppelklick auf das File `IMP.CPP.Aplication.Src.FirstStep.cpp` öffnet den ersten Beispieltask.
`c:\IMD\Program\INOS\Samples\IMP_MAS\FirstStep\Applicat\Src\FirstStep.cpp`

Lässt sich das Projekt nicht öffnen, muss das File "FirstStep.str" gelöscht werden. In diesem Fall verwenden Sie eine neuere Version des IMD - Indel Master Desk.

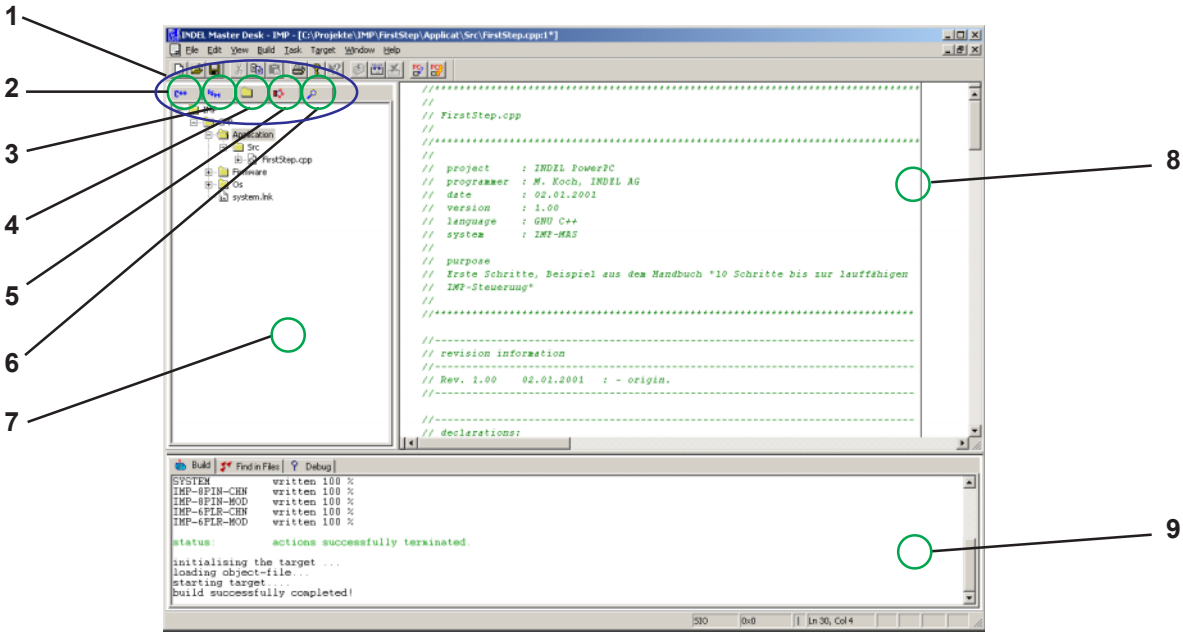


Abb 4.1 IMD - Indel Master Desk

1 Projektverwaltung
2 C++ Task-Fenster: Zustand, Name, Adresse und Programm Counter aller Tasks
3 ISM-Task-Fenster: Zustand, Name, Adresse und Programm Counter aller Tasks
4 Projektstruktur
5 Konfiguration IMP-Module, INFO-Link Module
6 Diagnose Tool (Nur für INFO-Link)
7 Projektstruktur
8 Editier-Fenster
9 Compiler-, Download- Nachrichten

ISM-Tasks
Nebst der Programmiersprache C++ kann die IMP-Steuerung und der INFO-Link in Anweisungsliste namens ISM-6.0 programmiert werden.

Die einfach zu erlernende Makrosprache hat aber bis heute ihre Bedeutung und läuft voll kompatibel mit den neuesten Betriebssystemen auf allen neuen Masterkarten.

ISM-6.0 Manual siehe (CD-ROM): `m:\INFO-Link\Software\Deutsch\ISM-Buch.pdf`

Projektstruktur
In der Projektstruktur ist der gesamte Source Code des Projektes aufgelistet:

`Applikation`	kundenspezifische Anwendungen
`Firmware`	hängt von der Konfiguration ab
`OS`	Betriebssystem, der Source-Code des Betriebssystemes liegt ebenfalls bei!

Firmware und Betriebssystem Libraries aller Beispiel-Programme befinden sich zentral im Verzeichnis `\Lib32` im Installationsverzeichnis (Standard: `c:\imd\`).

5. Code editieren

Im Beispiel FirstStep.cpp befindet sich folgender Code in der While-Schleife der Operation `CFirstStep::Action()`:

```
if (m_pInput0->Test()){
     // Input0 has value 1
     m_pLed0->Set();
} // End of if
else {
     // Input0 has value 0
     m_pLed0->Clear();
} // End of if
Relinquish();
```

Bei gedrücktem Eingang "m_pInput0" (Eingang 0, IMP-8PIN Modul 0) wird der Ausgang "m_pLed0" (Ausgang 0, IMP-8POT Modul 0) gesetzt.

6. Projekt Laden

6.1 Umgebungs-Variablen

Bevor das Projekt in den IMP-Master geladen werden kann, muss im IMD noch bekannt gemacht werden, in welches Target das Projekt geladen werden soll.

Target
Dazu muss die Umgebungsvariable `Target` gesetzt werden: Rechter Mausklick in der Projektverwaltung *(Abb 2.1.6, Pos 2)* im Verzeichnis Projektstruktur *(Abb 2.1.6, Pos 4)* öffnet ein Auswahlmenue, wählen Sie `Environment Variables` **(4),** unter `Target` geben Sie den Namen ein, den Sie bei der Installation zuvor für das `SIO-Target` angegeben haben. Mit `Set` wird der Wert übernommen. Im Feld **(6)** ist der aktuelle Target-Name ersichtlich.

Debug-Release
Im Debug-Release `GCC_FLAGS` kann der Optimierungs-Grad angegeben werden:

`-g`: Nicht optimiert, mit Debug-INFO, debuggen ist möglich
`-g, -O2`: Optimiert übersetzt, mit Debug-INFO, debuggen ist nur eingeschränkt möglich

Wichtig
Wird der Code optimiert compiliert, ist er auch schneller und benötigt weniger Speicherplatz! Mit oder ohne Debug-Info hat keinen Einfluss auf Grösse und Geschwindigkeit des Programm-Codes.

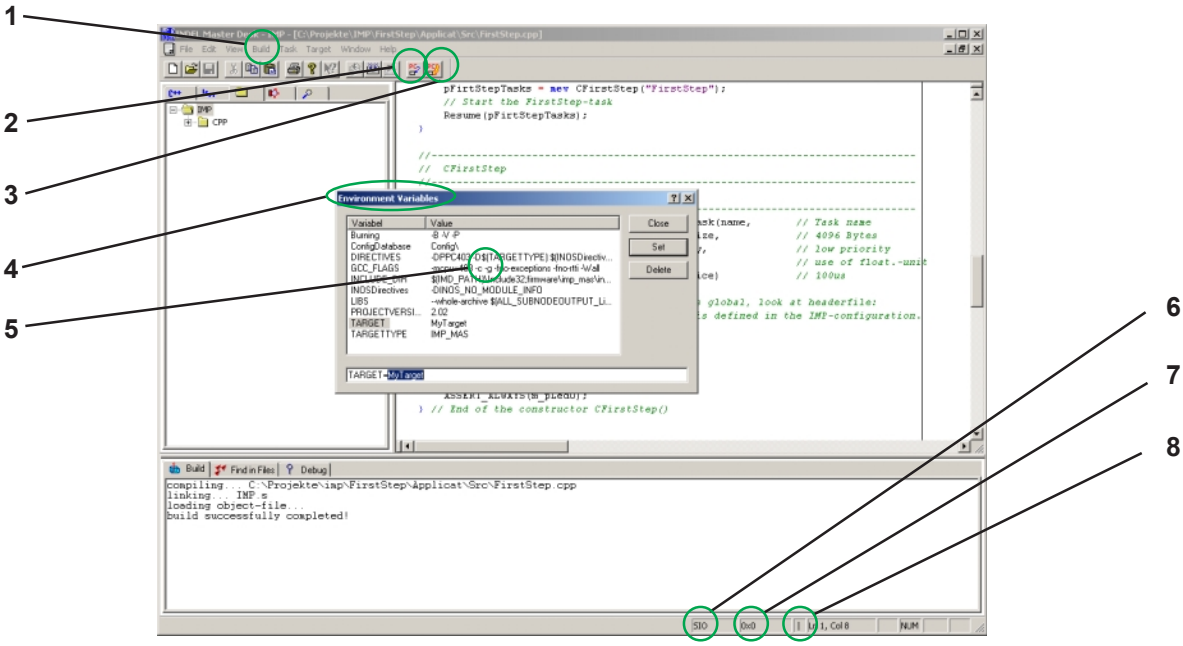


Abb 6.1.1 Umgebungs Variablen, Build, Download

1 Build, Compilieren, Download
2 Load Target, Download Software ins (flüchtige) S-RAM
3 Burn Target, Software downloaden und ins (resistente) Flash-Memory brennen
4 Umgebungs Variablen
5 Debug-Optionen `GCC_FLAGS`
6 Target-Namen (z.B. MyTarget)
7 Fehler-Code (Siehe Online Hilfe -> **F1**)
8 Lebenszeichen vom Target, drehender Balken

6.2 Kompilieren, Download

Jetzt muss das Projekt neu kompiliert werden. Drücken Sie dazu die **F8** Taste um das Projekt zu kompilieren und anschliessend in den IMP-Master zu laden. Im Menue `Build` **(1)** finden Sie alle Kommandos für Compilieren und Download aufgelistet.

`F7`	Build
`Ctrl + F7`	Rebuild Node
`Ctrl + F8`	Load Target
`F8`	Build and Load Target

6.3 Load, Burn Target

`Load Target` *Abb. 6.1.1 Pos. 2* speichert den Programm-Code im flüchtigen S-RAM. Bei Power-down geht das Programm unwiderruflich verloren.

Um den Code dauerhaft im IMP-Master zu speichern, muss `Burn Target` *Abb. 6.1.1 Pos. 3* ausgeführt werden. Damit wird der Code ins nicht-flüchtige Flash-Memory geschrieben, und bleibt auch bei Power-down erhalten. Nach `Burn Target` läuft das System immer noch im S-RAM mit `Hardware Reset` *Abb. 6.3.1 Pos. 1* oder Power-Up startet der IMP-Master mit dem neuen Code.

Damit besteht die Möglichkeit in einer laufenden Anlage jederzeit ein neues Programm zu laden, ohne diese unterbrechen zu müssen.

Wenn das Projekt ordnungsgemäss im IMP-Master läuft und der IMD - Indel Master Desk mit dem Target in Kontakt steht, erscheint ein sich drehender Balken im Fenster von *Abb. 6.1.1 Pos. 8*. Ansonsten schlagen Sie im *Anhang A Fehlermeldungen* nach um die Ursache der Fehler zu finden.

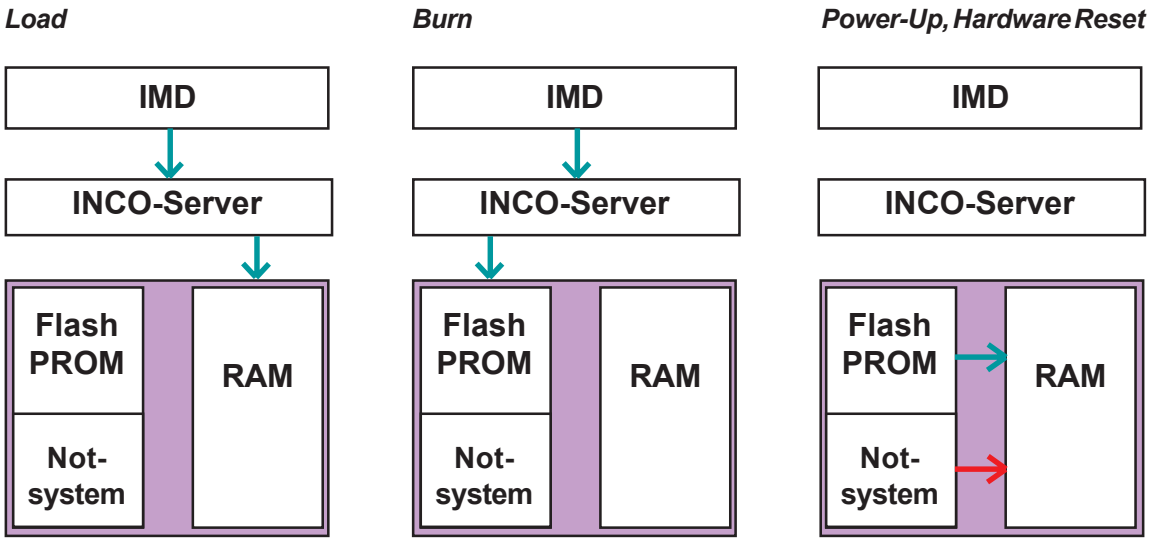


Abb 6.3.1 Load, Burn Target

Rot: Mit dem Notsystem-Stecker wird bei Power-Up nur das Notsystem ins RAM geladen. Siehe auch: *Anhang A.2 "Fehler bei Power-Up der IMP-Steuerung"*.

6.4 Reset Target

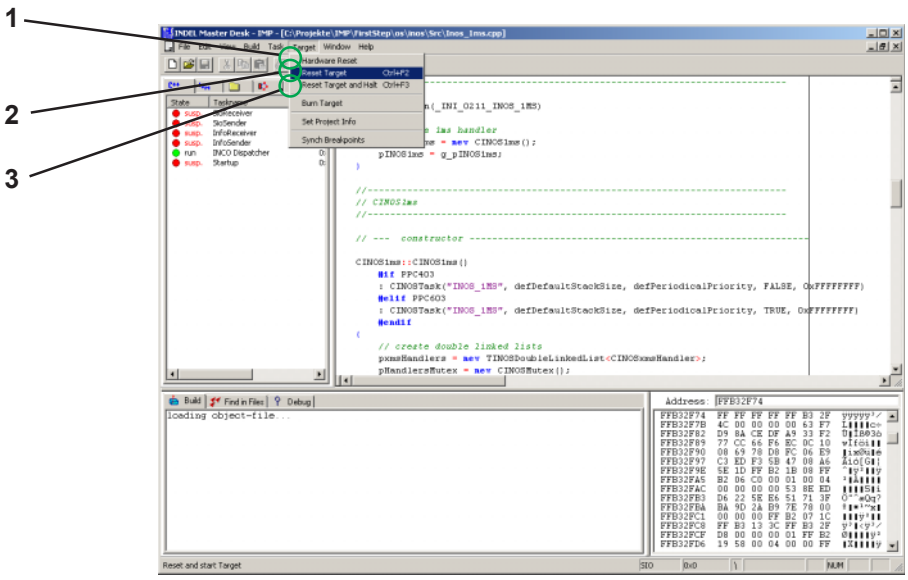


Abb 6.3.1 Reset Target

1 Hardware Reset: Das Target wird zurückgesetzt, der Programmcode wird aus dem Flash-Memory ins S-RAM kopiert und ausgeführt.
2 Reset Target: Das Target wird zurückgesetzt, der Programmcode im S-RAM wird gestartet.
3 Reset and Halt: Das Target wird zurückgesetzt, der Programmcode wird nicht gestartet.

> Bei Ziffer 2 und 3: Nicht möglich mit folgender Option in der Konfiguration: ReuseStartupMemory = yes. Siehe auch *Kap. 10.4.1 Optionen*

7. Programm testen

7.1 Status Anzeige der Module

Über das Tool "Variablen Explorer" kann das Innenleben aller Master Karten auf einfachste Art und Weise eingesehen werden. (Beschreibung "Variablen Explorer" siehe Kapitel 9. Tools)

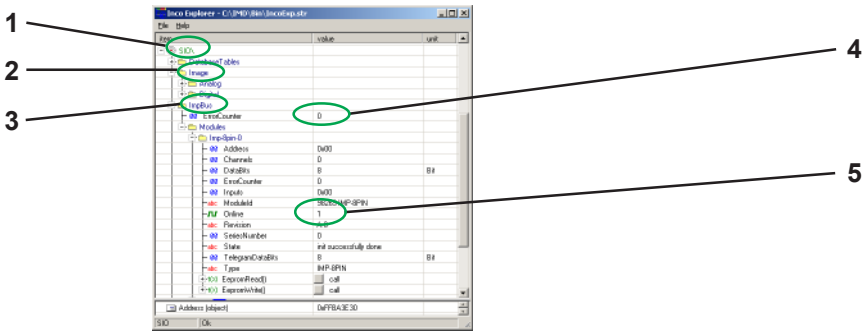


Abb 7.1.1 INCO Explorer

1 Target Name
2 Prozess Abbild
3 Konfigurierte Hardware
4 Fehler Zähler
5 Zustand IMP-Modul

Starten Sie den Variablen Explorer:
`Start > Programme > Indel Master Desk > Variable-Explorer.`

oder im IMD - Indel Master Desk:
`View > Variable-Explorer` bzw. `Alt+V`

Der Variablen Explorer zeigt sämtliche Targets **(1)** an, die auf diesem PC mit dem Registrierungs Editor erstellt worden sind.

Die gesamte Kommunikation zwischen PC und Targets funktioniert über Variablen-Namen und nicht über Adressen. Damit ist der Anwender von der Zuordnung Daten - Adresse befreit, diese Aufgabe übernimmt das Betriebssystem. Dies erleichtert auch den Netzwerkbetrieb wesentlich, da der Zugriff auf Variablen sämtlicher Targets auch über Modem, Internet und PC-Netzwerke erfolgen kann. Diese Möglichkeiten sind standardmässig im Lieferumfang enthalten.

Die Verzeichnisstruktur, wie sie der Variablen-Explorer oben zeigt, wird vom Betriebssystem angelegt. Die wichtigsten Einträge sind das Prozessabbild `Image` **(2)** sowie die Hardware Konfiguration `ImpBus` **(3)**. Selbstverständlich kann der Benutzer eigene Strukturen und Variablen definieren.

Im Verzeichnis `ImpBus` **(3)** wird die gesamte Hardware aufgelistet, die konfiguriert wurde. Jedes Modul wird mit einer Anzahl von Variablen charakterisiert:

`Error Counter`	Übertragungs Fehler des entsprechenden Modules
`Online`	1 = Modul meldet sich, 0 = Modul kann nicht angesprochen werden
`Module ID`	Bestell-Nummer mit Option und Name des Modules
`Revision`	Hardware Version (alphanumerisch) und GAL-Software Version (numerisch)

7.2 I/Os überschreiben

Zu Testzwecken können alle Ein- und Ausgänge aus dem INCO-Explorer überschrieben werden. Dazu muss das `Overwrite-Bit` des entsprechenden Ein- Ausganges auf 1 gesetzt werden.

Für jeden I/O wird zudem der Zustand bei `(t-1)` in der Variablen `Latched` angezeigt. Damit kann die Flanke des I/Os detektiert werden. (t = 0.5, 1, 2, 4 ms)

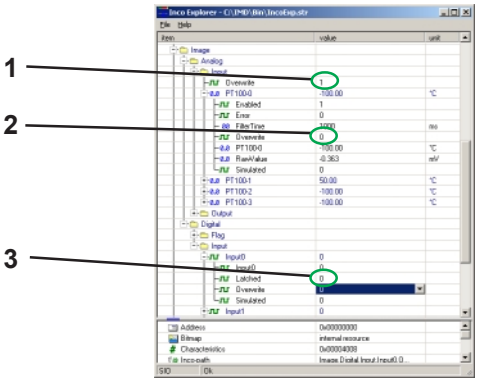


Abb 7.2.1 Overwrite INCO Explorer

1 Overwrite für alle analogen Eingänge
2 Overwrite für analogen Eingang PT100-1
3 Latch für digitale Eingäne

Wird in unserem Beispiel das `Overwrite-Bit` für `INPUT0` gesetzt, kann der Eingang überschrieben werden und der Ausgang geht auf `Ein`.

7.3 Hardware simulieren

Ist die Hardware nicht oder unvollständig vorhanden, wird das Prozessabbild trotzdem erstellt. Die fehlenden Hardware-Komponenten werden in diesem Fall simuliert.
(Siehe *Kapitel 10.4.1 Konfiguration*)

Stimmt die Stellung des Adress-Wahlschalters (Drehschalter) eines Modules nicht mit der Konfiguration überein, werden die betreffenden Kanäle ebenfalls simuliert, d.h. die Software setzt evtl. virtuelle Ausgänge.

Als Kontrolle dient die Variable `Online` im Verzeichnis `ImpBus.Modules.Imp-xy`, die nur 1 wird, wenn das Modul tatsächlich vorhanden ist. Es ist deshalb empfehlenswert, dass die Anwender-Software aus einem Kontroll-Task beim Einschalten die Präsenz der Module testet.
(In einer späteren Version.)

8. Debuggen

8.1 Task-Fenster, Edit-Fenster

Task-Fenster und Edit-Fenster sind im IMD nicht identisch. D.h. Wenn Sie einen Breakpoint im Editier-Fenster setzten, müssen Sie im Task-Fenster einen Doppelklick auf den entsprechenden Task ausführen, damit Sie debugen können. Siehe *Pos. 2, Abb. 4.1.*

Im Menue Task **(1)** finden Sie alle Debug-Funktionen.

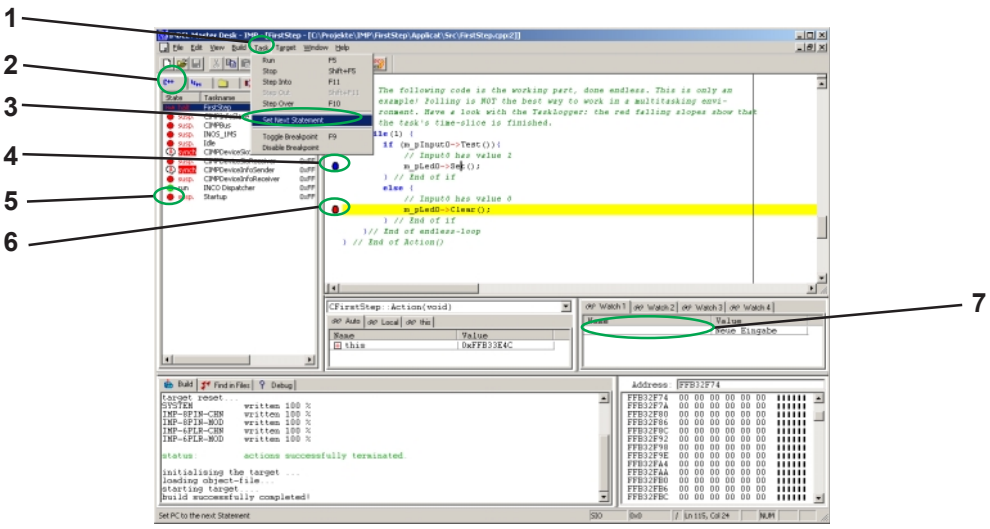


Abb 8.1.1 Debuging

1 Task-Menü
2 Task-Fenster
3 Next Statement
4 Globaler Breakpoint
5 Zustand der Tasks
6 Task spezifischer Breakpoint
7 Watches

8.2 Task-Kontrolle

Breakpoints
Im Source-Code können beliebig viele Breakpoints **(4,6)** gesetzt werden. Alle Interrupts sind in C++ geschrieben und werden als Tasks aufgerufen. Daher können auch in Interrupts Breakpoints gesetzt werden.

Grundsätzlich wird zwischen `Globalen Breakpoints` und `Task Breakpoints` unterschieden. Bei globalen Breakpoints wird immer angehalten. Bei Task Breakpoints muss ein Task-Name aus dem Task-Fenster **(2)** angegeben werden. Der Programm-Code wird nur angehalten, wenn der Code aus dem angegebenen Task angesprungen wird.

Next Statement
Mit `Next Statement` **(3)** springt der PC (Programm-Counter) an die Code-Zeile an der sich der Cursor befindet. Damit können Code-Teile übersprungen oder wiederholt werden.

Zustand der Tasks

> Für weitere Hilfe suche in der Online-Hilfe nach:
> - `Run`
> - `Susp`
> - `Trap`
> - `Sleep`

Stop
Wählen Sie einen Task **(5)** und drücken Sie `Shift + F5`, der Task wird unmittelbar angehalten. Jetzt kann der Task mit `Step Into`, `Step Over` schrittweise abgearbeitet werden.

Step Over, Step Into
`Step Over` **F10** führt Unterprogramme, Funktionen komplett aus, `Step Into` **F11** springt in Funktionen hinein.

Watches
In den max. vier Watches **(7)** können benutzerspezifisch Programm-Variablen angezeigt werden.

8.3 Breakpoint List

Mit `Alt + B, View -> Breakpoint List` erscheint eine Liste mit allen gesetzten Breakpoints und zugehörigen Informationen:

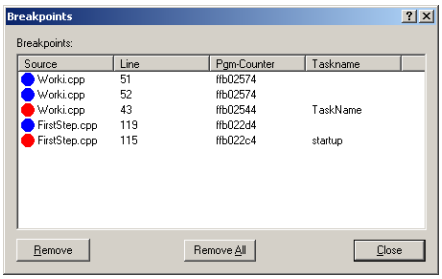


Abb 8.3.1 Breakpoints

9. Software Tools

Sämtliche Tools sind im Lieferumfang mit inbegriffen.

9.1 IMP Konfiguration

Mit der rechten Maustaste können neue IMP-Module eingefügt und konfiguriert werden.

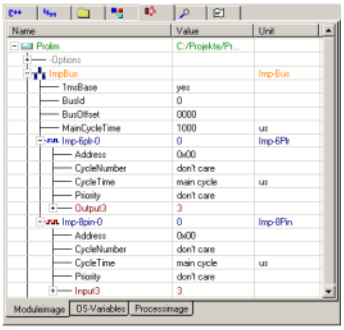


Abb 9.1.1 Konfiguration

Weitere Informationen siehe Kapitel10.4 Konfiguration.

9.2 Variablen Logger

Sämtliche registrierten Variablen können mit dem Variablen-Logger aufgezeichnet und ausgewertet werden. Nebst Variablen können auch Speicherzellen aufgezeichnet werden. Der Logger wird wie ein Oszilloskop bedient. Dem Benutzer stehen acht Kanäle zur Verfügung mit verschiedenen Trigger-Möglichkeiten.

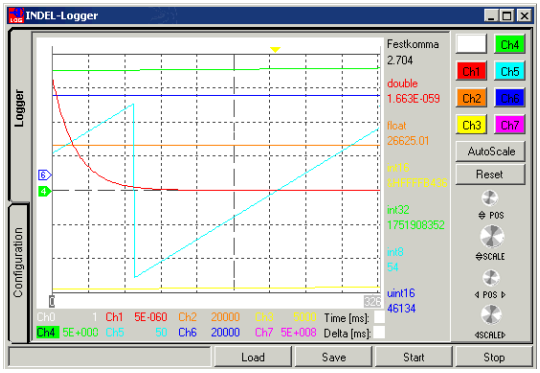


Abb 9.1.1 Variablen Logger

Kurven können in Files oder als Referenz für Vergleiche abgelegt werden. Die Kurven können in verschiedenen Formaten zur Weiterverarbeitung in Mathcad oder Excel abgelegt werden.

Je nach Konfiguration der Zeitbasis können schnelle Ereignisse (<1ms) oder lange Verläufe über mehrere Stunden oder Tage aufgezeichnet werden.

Weitere Informtionen siehe IMD-Hilfesystem unter `Zubehör.OCX.VarLog`

9.3 INCO Explorer

Mit dem INCO Explorer können alle im System registrierten Variablen dargestellt werden. Der INCO Explorer ist netzwerkfähig und kann auf entfernten PCs Einblick in ein System gewähren, auch über Internet und Modem-Verbindungen! Der Variablen-Explorer eignet sich hervorragend als Service-Tool, Inbetriebnahme-Tool und zur Visualisierung.

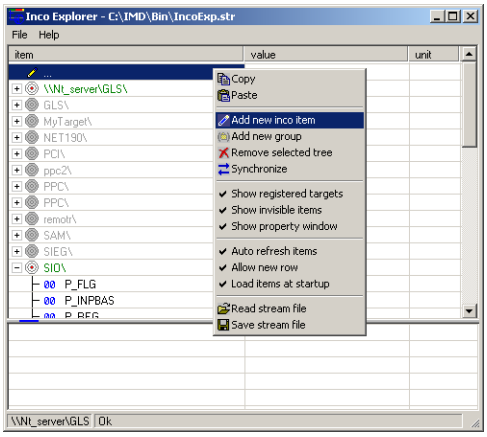


Abb 9.3.1 INCO Explorer

Remote-Target
Ein Remote-Target wird im INCO-Explorer wie folgt installiert:

- Rechter Mausklick in der linken Bildhälfte oben links
- Add new Inco Item
- Netzwerkpfad eingeben: `\\PC-Name\Remote_Target_Name\`
- IP-Adresse eingeben: `\\128.65.56.155\`

jetzt kann auf das Remote-Target zugegriffen werden.

9.4 Win-Show

Das Programm Win-Show bietet die gleichen Funktionen wie das Programm ACS-Show.
Win-Show eignet sich für die Inbetriebnahme aller Indel Achs-Karten: Servo-Regler,Schrittmotor-Indexer und Endstufe, DC-Motor Endstufe, usw.

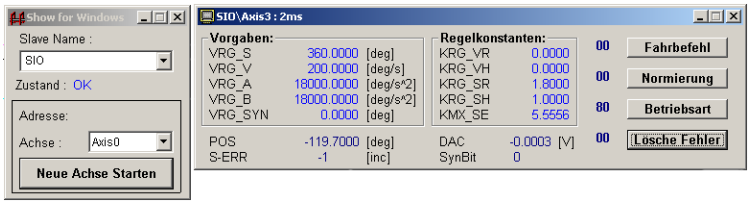


Abb 9.4.1 Win Show

Siehe auch "Regler Manual AC-Servo Regler".

9.5 INCO-Server

Für die netzwerkweite Verbreitung der Feldbusdaten sorgt der INCO-Server. Der INCO-Server wird auf jedem PC in einem Netzwerk installiert, der auf Feldbusdaten zugreift. Aus einer Applikation kann mit einfachen Funktionen aus einer DLL (Dynamic Link Library) auf den Feldbusmaster zugegriffen werden.

Heterogene Netzwerke wie z.B. Verbund 10MBit oder 100MBit beherrscht der INCO-Server ebenso souverän wie Datenaustausch über Internet und Modem-Verbindungen.

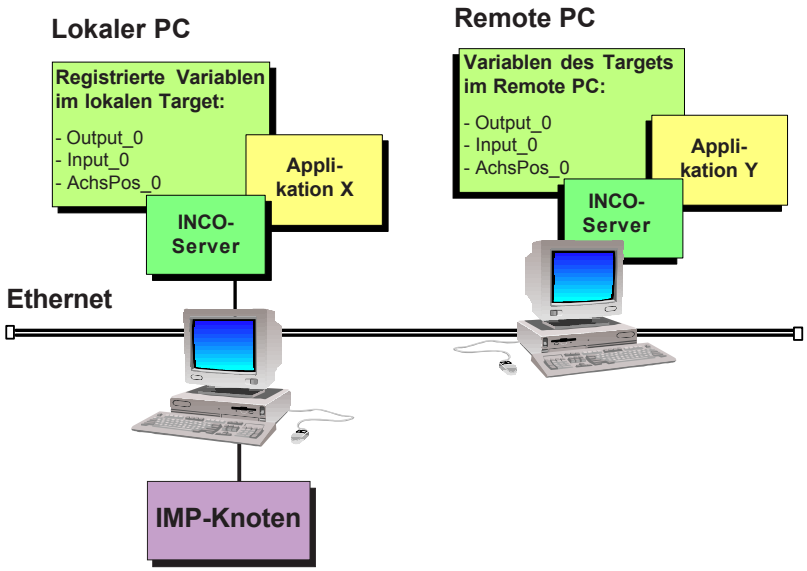


Abb 9.4.1 INCO-Server

Die Kommunikation zwischen PC und Feldbusmaster baut auf einfachen Funktionen auf:

`PutVariable`	schreibe eine Variable in den Master und
`GetVariable`	hole eine Variable vom Master

Weitere Informationen siehe Kapitel "10.1 Aufbau des Betriebssystemes" oder IMD-Hilfesystem unter `Zubehör.Tools.INCO-Server`.

9.6 ACS-Show

Mit dem Programm ACS-Show werden Achsen in Betrieb genommen, ohne eine Zeile Code zu schreiben. Der Logger zeichnet 6 beliebige Parameter wie Phasenstrom, Schleppfehler, Zwischenkreisspannung, usw. auf. Betriebssystem Updates, PID-Parameter abgleichen, usw. werden ebenfalls mit dem ACS-Show vorgenommen.

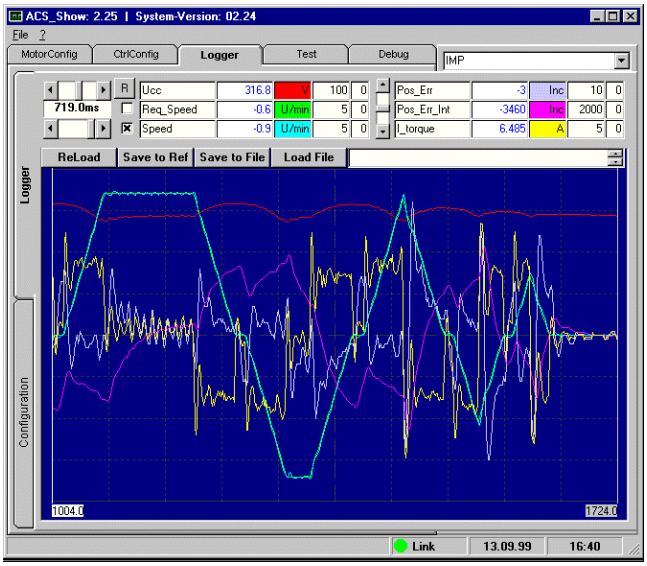


Abb 9.5.1 ACS-Show

Weitere Informtionen siehe File (CD-ROM)
`m:\INFO-Link\Software\Deutsch\acs_manu.pdf`

9.7 Weitere Tools

ACS-Update	Konsolen-Programm für Regler Software und Parameter Update
INCO-Registry	Registrierungs-Editor
INFO-Link Diagnose	Fehler auf einzelnen Segmenten des INFO-Link finden
Map Watch	Rettet Speicher-Inhalt des Targets im Fehlerfall
Memory Dump	Adressbereiche im Zielsystem einsehen
OutWin	Ausgabefenster für Meldungen
Task Logger	Zeitliche Erfassung einer beliebigen Anzahl von Tasks (im IMD)
Task Viewer	Informationen zu C++ Tasks
Trans32	Download Projekt in ein beliebiges Target

Weitere Informtionen siehe IMD-Hilfesystem unter `Zubehör.Tools, bzw. Zubehör.OCX`.

10. Software Lehrgang

10.1 Aufbau des Betriebssystemes

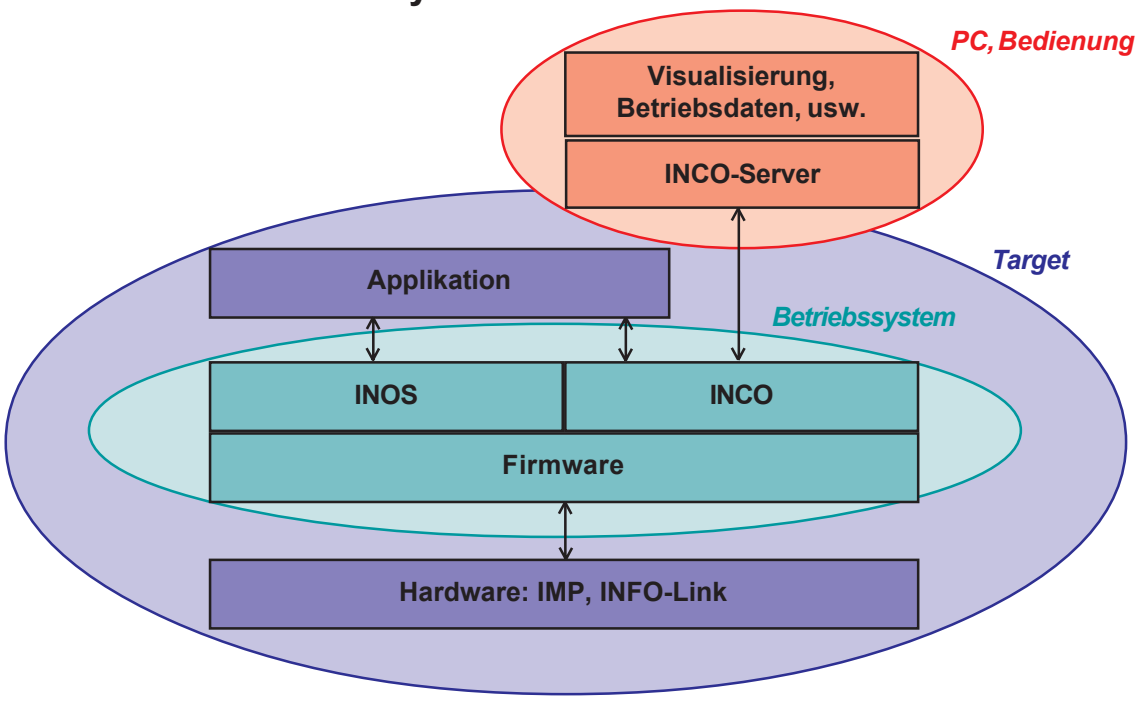


Abb 10.1.1 Aufbau Betriebssystem

Target, Zielsystem
z.B. IMP-Master, INFO-Link Master "INFO-PPC", Remote-Target, Stand-Alone Master mit Ethernet-Anschluss "INFO-SAM".

INCO, INCO-Server, Kommunikation
Indel **Co**nnectivity: Schnittstelle zwischen Visualisierung und Prozessabbild bzw. Applikation.
Die INCO-Schnittstelle ist für alle Targets identisch. d.h. Die Kommunikation zwischen Bedienung und Target ist immer gleich, auch über PC-Netzwerke, Modemverbindungen und Internet.
Der INCO-Server läuft auf dem PC und ist für die netzwerkweite Verfügbarkeit der Feldbus-Daten verantwortlich.

INOS, Betriebssystem
Indel **O**perating **S**ystem: Multitasking-fähiges Echtzeitbetriebssystem. Das INOS Betriebssystem läuft auf allen Indel Targets. D.h. Applikation können ohne grossen Aufwand von einem Target aufs andere portiert werden. Z.B. vom IMP-Master auf den INFO-Link Power PC Master.

Das INOS Betriebssystem arbeitet mit Task-Prioritäten und nach dem Time-Slice Verfahren. Tasks mit höchster Priorität werden nicht unterbrochen. Tasks mit niedriger Priorität können von höher priorisierten Tasks unterbrochen werden, bzw. werden vom Scheduler nach einer bestimmten Zeit unterbrochen, um Tasks mit der gleichen Priorität Rechenzeit zuzuweisen.

Firmware
Targetspezifischer Code, Hardware-Treiber.

Applikation
Kundenspezifischer Code, Anwender-Programm.

10.2 Datentypen und Zahlenformate

10.2.1 Einsatz der Zahlenformate

Float

Der IMP-Master verfügt über keine eigene FPU (Floating Point Unit). Möchte man trotzdem nicht auf Float Zahlen verzichten, können diese emuliert werden. Die Emulation der Float-Zahlen benötigt mehr Speicher-Ressourcen und kann nur in IMP-Mastern ab 1MByte Speicher ausgeführt werden. Ausserdem benötigt die Float-Emulation mehr Rechenleistung. Eine Float Division dauert ca. 14 mal länger als eine Division mit Integer-Zahlen (int32)!

Damit der Compiler Float-Zahlen akzeptiert muss die Compiler-Directive -DUse_Floating eingetragen werden.

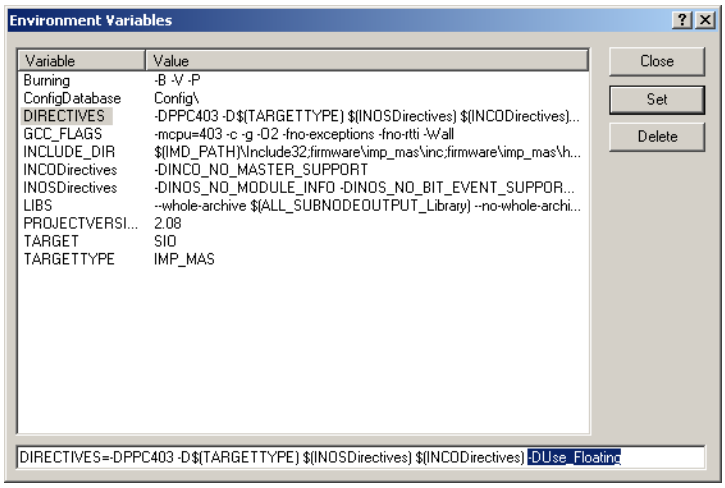


Abb 10.2.1 Environment Variable "Directives"

Wichtig: Nachdem die Compiler-Directive gesetzt worden ist muss das gesamte Projekt neu Kompiliert werden! Sämtliche Code-Stellen im Betriebssystem wo alternativ Float-Zahlen oder Fixed-Zahlen eingesetzt werden, sind zweimal implementiert worden: optimiert für Float-Zahlen bzw. für Fixed-Zahlen. Anhand der Compiler-Directive wird der entsprechende Teil übersetzt.

Fixed

Als Alternative zu den Float-Zahlen wurde das Zahlenformat `Fixed-32` eingeführt.
Eine `Fixed-32` Zahl hat 16-Bit vor dem Komma und 16-Bit hinter dem Komma. Damit ergibt sich ein Zahlenbereich von `+-32760.00000`. Werden Fixed-Zahlen verwendet, muss stets auf Bereichsüberläufe geachtet werden, speziell wenn Multiplikationen und Divisionen ausgeführt werden. Mathematische Operationen können auch ungenau werden, da die Auflösung beschränkt ist. Z.B. Additionen von vielen kleinen Werten << 1.

Mathematische Operationen mit Fixed-32 Zahlen sind sehr viel schneller als emulierte Float-Zahlen.

Werden sehr grosse bzw. kleine Zahlen benötigt, kann auch der `Fixed-64` Datentyp verwendet werden. Beachten Sie die Tabelle, mit den Rechenzeiten der einzelnen Datentypen.

Beispiel Fixed-32
Die Fixed-32 Zahl besteht aus 32Bit; vor dem Komma 16 Bit und hinter dem Komma 16 Bit. Mit den 16 Bit hinter dem Komma können 2^16 Werte eingestellt werden. D.h. der Zahlenbereich von 0 ... 0.999999 kann mit 65537 Werten aufgelöst werden.

Die kleinste Zahl, die dargestellt werden kann ist: 1/65537 = 0.00001525902
Die zweitkleinste Zahl: 2 x 1/65537 = 0.00003051804

```
10.5     als  fixed32:   m_rDac_Out = fixed32(10,32767)
35.0     als  fixed32:   m_rDac_Out = fixed32(35,0)
99.999   als  fixed32:   m_rDac_Out = fixed32(10,65469)
```

Fixed-64 analog: 32 Bit vor dem Komma, 32 Bit hinter dem Komma

Real
Soll Code sowohl für ein IMP-System als auch für ein INFO-Link Master geschrieben werden, bietet sich der Real-Datentyp für Fliesskomma-Zahlen an. Bei einem Master mit FPU werden alle als Real32 deklarierten Variablen vom Betriebssystem als Float Zahlen verarbeitet, Real64 analog als Double.

In einem System ohne FPU werden Real Zahlen als Fixed-Zahlen verarbeitet. Auch beim Einsatz von Real-Zahlen muss darauf geachtet werden, dass keine Probleme mit Bereichsüberläufen und zu kleinen Zahlen, die nicht mehr genügend genau sind, entstehen.

```
10.5     als  real32:   m_rDac_Out = real32(10.5)
10.99    als  real64:   m_rDac_Out = real64(10.99)
```

10.2.2 Standard Datentypen

Name	*Vorzeichen*	*Bitbreite*	*Bereich*	
int8	signed	8	-128 ... 127	
uint8	unsigned	8	0 ... 255	
int16	signed	16	-32768 ... 32767	
uint16	unsigned	16	0 ... 65535	
int32	signed	32	-2147483648 ... 2147483647	
uint32	unsigned	32	0 ... 4294967295	
int64	signed	64	-2^63 ... 2^63-1	
uint64	unsigned	64	0 ... 2^64-1	
float	signed	32	-10^38 ... 10^38	Genauigkeit: 8 signifikante Stellen
double	signed	64	-10^308 ... 10^308	Genauigkeit: 15 signifikante Stellen

10.2.3 Fixed32, Fixed64

Für Targets, die keine Floatingpoint-Zahlen unterstützen ist ein Festkomma-Datentyp verfügbar:

Name	Vorzeichen	Bitbreite	Bereich
fixed32	signed	32	-32768.00000 ... 32768.99999
fixed64	signed	64	-2147483648.00000000 ... 2147483647.99999999

Der Dezimalpunkt befindet in der Mitte der Integer-Zahl: Zwischen Bit 15 und 16 bei Fixed32 und zwischen Bit 31 und 32 bei Fixed64.

10.2.4 Real32, Real64

Kompatibilitäts-Datentypen: Soll Software von einem Target ohne FPU auf ein Target mit FPU portiert werden bieten sich die Real-Datentypen an.

Der Compiler ersetzt Real32 Datentpyen wie folgt:

Targets mit FPU (INFO-PCI, INFO-SAM)

```
real32 ->          float
real64 ->          double
```

Targets ohne FPU (IMP-MAS)

```
real32 ->          fixed32
real64 ->          fixed64
```

Targets ohne FPU mit Compiler-Directive -DUse_Floating

```
real32 ->          float
real64 ->          double
```

Bereiche:

Name	Vorzeichen	Bitbreite	Bereich
Real32	signed	32	Target mit FPU siehe float
Real32	signed	32	Target ohne FPU siehe fixed32
Real64	signed	64	Target mit FPU siehe double
Real64	signed	64	Target ohne FPU siehe fixed64

10.2.5 Weitere Datentypen :

Name	Beschreibung
INOSTime	Struktur zur Darstellung von Datum und Uhrzeit.

10.2.6 Code-Beispiel Fixed-Zahlen versus Float-Zahlen

Im Beispiel `Analog.cpp` sind sämtliche Fliesskommazahlen als Real deklariert. Mit aktivierter Compiler Directive `-DUse_Floating` werden aus den Real-Zahlen Float bzw. Double-Werte. Ohne die Directive macht der Compiler fixed32, bzw. fixed64 (Integer) -Werte.

Registrierung von Variablen

Registrierung einer Variable als Real-Wert wird vom Compiler mit und ohne `-DUse_Floating` akzeptiert (Siehe Kapilel "10.11.1 Variablen registrieren":

```
MyVariables->Add(new CINCOreal64("Real64",
     &m_rReal64,                        // Variable
     real64(0.0),                       // Minimum
     real64(1000.0),                    // Maximum
     "°",                               // Unit
     defCharShowFix|SHOW_DIGIT(1)));    // Darstellung
```

Registrierung einer Variable als Double-Wert wird vom Compiler nur mit `-DUse_Floating` akzeptiert:

```
MyVariables->Add(new CINCOdouble("Double",
     &m_Double,                         // Variable
     0.0,                               // Minimum
     1000.0,                            // Maximum
     "°",))                             // Unit
```

10.2.6 Laufzeiten für mathematische Operationen

Gemessene Zeiten im Vergleich. PPC403 (66MHz) und PPC603 (100MHz) für eine Operationen.

Add Mul Div mit int32	**PPC 403 ns**	**PPC 603e ns**
double	53'309	585
fixed64	6'499	4'016
uint64	3'012	1'821
float	18'494	474
int32	681	461
Add	**PPC 403 ns**	**PPC 603 ns**
double	14'700	89
fixed64	-	-
uint64	273	141
float	10'428	89
int32	121	51
Mul	**PPC 403 ns**	**PPC 603 ns**
double	13'184	99
fixed64	-	-
uint64	530	195
float	8'856	94
int32	167	61
Div	**PPC 403 ns**	**PPC 603 ns**
double	16'234	193
fixed64	-	-
uint64	2'679	1'645
float	8'810	146
int32	605	413

Tabelle 10.2.1 Rechenzeit

10.3 Neues Projekt erstellen

Betriebsystem mit Firmware benötigen ca. 150 ... 200 kByte Speicher, je nach angeschlossener Peripherie. Für kleine Projekte, die nur digitale I/Os benötigen, reicht ein IMP-Master mit 256 kByte Speicher.

Unter File > New Project kann ein neues Projekt erstellt werden. Zur Auswahl stehen folgende Varianten:

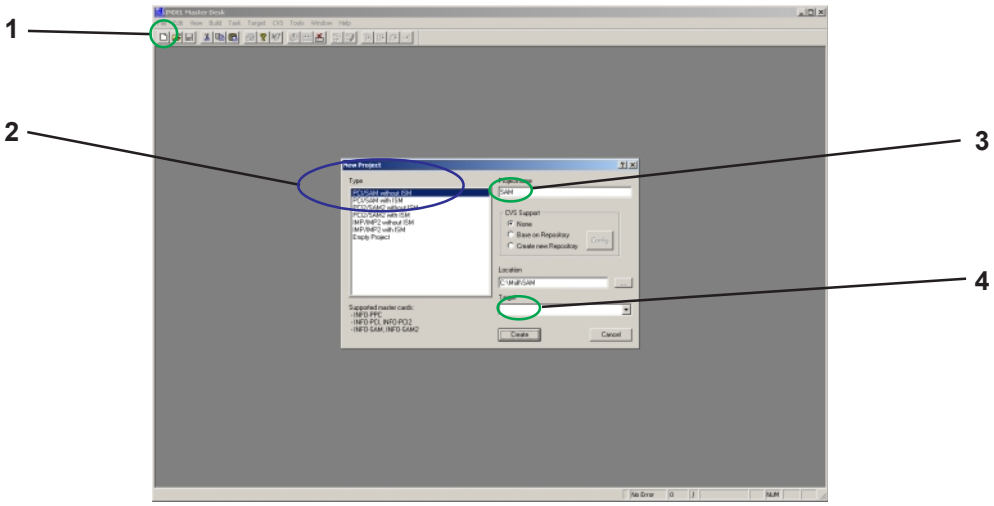


Abb 10.3.1 New Project

1 File, New Project
2 Projekt Auswahl
3 Projekt Name
4 Target Name

PCI/SAM with/without ISM
Projekt für SAM oder PCI-Master. Wahlweise mit oder ohne ISM-Interpreter.

PCI2/SAM2 with/without ISM
Projekt für SAM2 oder PCI2-Master. Wahlweise mit oder ohne ISM-Interpreter.

IMP/IMP2 with/without ISM
Projekt für IMP- oder IMP2-Master. Wahlweise mit oder ohne ISM-Interpreter. Die Wahl zwischen IMP oder IMP2 wird in der Konfiguration (Kapitel 10.4 Konfiguration) getroffen.

10.4 Konfiguration

10.4.1 Optionen

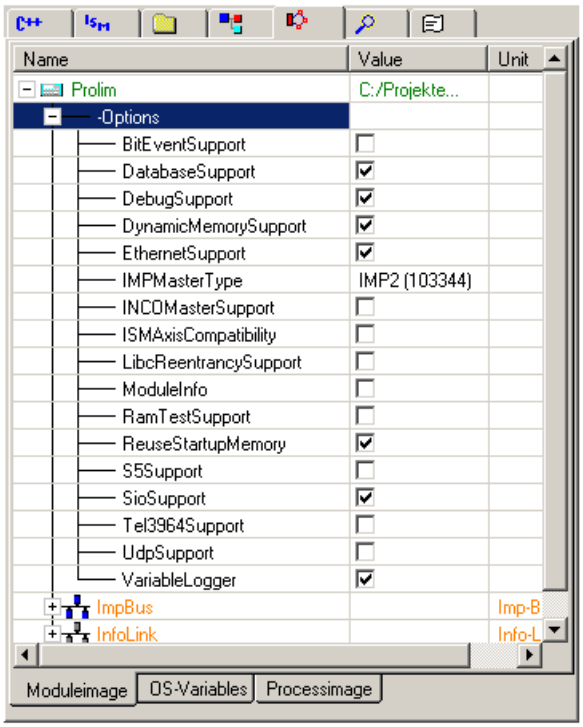


Abb 10.4.1 Konfiguration, Optionen

> Achtung: Wenn eine Options-Einstellung verändert wird, muss immer ein Rebuild auf das gesamte Projekt ausgeführt werden!

Um die Speicher-Ressourcen der IMP-Master optimal ausnutzen zu können, sind viele Optionen konfigurierbar:

Optionen aktueller IMD-Versionen

BitEventSupport	yes	Mit den folgenden Hilfeklassen kann auf ein CINOSBit (Digitaler Eingang, Ausgang oder Flag) gewartet werden bis es gesetzt oder gelöscht wird, ohne die CPU unnötig zu belasten. CINOSBitClearedEvent, CINOSBitSetEvent
DatabaseSupport	yes	Datenbank Funktionen werden unterstütz. Diese Option ist in der Regel immer nötig, da die gesamte Konfiguration mit Datenbanken, bzw. Tabellen realisiert ist.
DebugSupport	yes no	Debugen ist im vollen Umfang möglich. Debugen ist nicht möglich

DynamicMemorySupport	yes	
EthernetSupport	yes	Die Ethernet Schnittstelle wird unterstützt
IMPMasterType		Hier muss die Version des IMP-Masters angegeben werden: IMP-MAS 99280 oder IMP-MAS2 103344
INCOMasterSupport	yes	Sämtliche INCO-Funktionen können zwischen zwei Master verwendet werden. Z.B Serielle Verbindung zwischen zwei IMP-Mastern: Kommunikation mit Put- GetVariable. Kommunikation zwischen IMP-Master und Indel Servo-Regler ebenfalls mit Put- GetVariable, o. ä.
ISMAxisCompatibility	yes	
LibcReentrancySupport	yes	
ModuleInfo	yes	Sämtliche Informationen zu jedem Software-Modul wie: Author, Versions-Nummer, Datum, usw. werden mit Registriet und sind z.B. mit dem INCO-Explorer ersichtlich. Benötigt ca. 30kByte Speicher!
RamTestSupport	yes	Der RAM-Test vergleicht ständig den Inhalt des Flash-PROM mit dem S-RAM
ReuseStartupMemory	yes	Start-Funktionen werden nur einmal beim hochfahren des Systems abgearbeitet. Mit ReuseStartupMemory=yes kann dieser Speicher-bereich wieder freigegeben werden und z.B. für den Variablen Logger verwendet werden. Achtung: Ein Software-Reset ist mit eingeschalteter Option nicht mehr möglich!!
S5Support	yes	Mit dieser Option kann mit einer Siemens S5/S7 über Industrial Ethernet kommuniziert werden. (Nur mit Ethernet Schnittstelle)
SioSupport	yes	Unterstützt die SIO auf dem IMP-Master und die IMP-SIO Karten
Tel3964Support	yes	Das serielle Protokoll 3946R wird unterstützt
UdpSupport	yes	
VariableLogger	yes	Der Variablen-Explorer kann verwendet werden.

Optionen von früheren IMD-Versionen

1msSupport	*yes*	*1ms Handler wird unterstüzt. (IMD Rev. 1.34)*
ExceptionHandling	*yes*	*Das ANSI C/C++ Exception Handling wird unterstütz. Siehe GNU-Hilfe (IMD Rev. 1.34)*
InfoLinkConnection	*yes*	*Wird verwendet wenn der IMP-Knoten als Slave im INFO-Link betrieben wird. (IMD Rev. 1.34)*

10.4.2 OS-Variablen

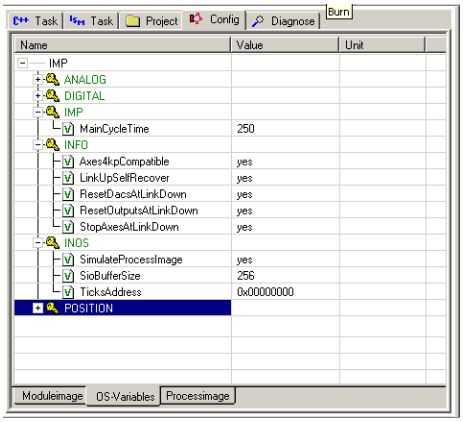


Abb 10.4.2 Konfiguration, OS-Variablen

MainCycleTime	250, 500, 1000µs	Zykluszeit für den IMP-Bus. Innerhalb diser Zeit werden alle IMP-Teilnehmer einmal angesprochen. Beliebige Zykluszeiten bis 0.050ms sind möglich, evtl. mit gewissen Einschränkungen.
Axes4kpCompatible	yes/no	Achs-Module wie IMP-SMI (Schrittmotor) werden wie INFO-4KP Karten übertragen. Option = no -> Achsen werden wie Servo-Regler übertragen. -> Entsprechende Karte konfigurieren im INFO-Master.
LinkUpSelfRecover	yes/no	Nach einem Link Down werden automatisch sämtliche Ausgänge des IMP-Knoten (analoge, digitale, Achsen,...) mit den Werten vom INFO-Master überschrieben. Option = no -> Nach einem Link Down werden die Zustände der Ausgänge beibehalten. Die Software im INFO-Master muss aktiv bestimmen ab welchem Zeitpunkt wieder die Werte vom INFO-Master gelten. Diese Option wird benötigt, wenn der IMP-Knoten eigene Intelligenz ausführt und im Falle eines Link-Downs unabhängig vom INFO-Master weiterarbeitet.
ResetDacsAtLinkDown	yes/no	Bei Link Down werden alle DAC = 0 gesetzt. Muss no sein wenn der IMP-Master eigene Intelligenz ausübt.
ResetOutputsAtLinkDown	yes/no	Bei Link Down werden alle Ausgänge = 0 gesetzt. Muss no sein wenn der IMP-Master eigene Intelligenz ausübt.
Stop AxesAtLinkDown	yes/no	Bei Link Down werden alle Achsen angehalten. Muss no sein wenn der IMP-Master eigene Intelligenz ausübt.
SimulateProcessImage	yes/no	Ist ein Peripherie-Modul physisch nicht vorhanden und wird ein "Find" auf einen Ein-Ausgang dieses Modules gemacht, kommt die Funktion trotzdem erfolgreich zurück. Wird benötigt um Software zu testen, ohne dass die benötigte Hardware vorhanden ist. Wird die Option = no gesetzt können z.B. Maschinen-Optionen mit dem Anschliessen der entsprechenden Peripherie in der Software aktiviert werden.
SioBufferSize	yes/no	Grösse des Buffers der seriellen Schnittstelle am IMP-Master.
TicksAddress	yes/no	Hardware-Adresse des Millisekunden-Timers. Kann bei Bedarf auf eine feste Adresse gelinkt werden.

10.5 Start Funktion

Für jeden Task können sog. Start-Funktionen ausgeführt werden. Die Namen aller Start-Funktionen werden vom Task `Startup` (Betriebssystem-Task) alphabetisch sortiert und bei Systemstart nach deren Reihenfolge aufgerufen. Im Normalfall setzt sich der Name folgendermassen zusammen:

`_INI_xxxx_MyModulname` wobei xxxx eine 4-stellige Zahl bedeutet

Mit diesem Hilfsmakro kann das System nach einer genau definierten Reihenfolge aufgestartet werden. Unterhalb von `_INI_1000` werden Betriebssystem Funktionen aufgestartet. In der Regel folgen alle Benutzerspezifischen Start-Funktionen ab `_INI_1000`. Breakpoints können erst nach `_INI_0200_INOS` gesetzt werden. Ein vorheriges Debuggen ist nicht möglich.

Das Starten eines Tasks geschieht ebenfalls in der Startfunktion.

Beispiel

```
#include <inos_mod.h>

bool MyVariable;

StartFunction(_INI_1000_MyStartFunction)
{
     // create task
     g_pMyWorkerTask = new CMyWorkerTask("Worki");
     // put task to READY state
     Resume( g_pMyWorkerTask );

     MyVariable = 0;
}
```

> Für weitere Hilfe suche in der Online-Hilfe nach:
> - `StartFunction`

10.6 Neuer Task

Das IMP-Betriebssystem ist echtzeit- und multitaskingfähig. Multitaskingfähig heisst, es können mehrere Programmteile "parallel" ablaufen. In Wirklichkeit läuft natürlich nur immer ein Task zu einem bestimmten Zeitpunkt. Das Betreibssystem ordnet jedem Task eine gewisse Zeit an Rechenleistung zu. Danach wird der Task unterbrochen und der nächste Task erhält Rechenleistung. Dieses Verfahren wird als "Preemptives Multitasking" bezeichnet.

Zusätzlich kann jedem Task noch eine Priorität zugesprochen werden. Wird, z.B. durch einen Interrupt ein Task mit höherer Priorität gestartet, unterbricht dieser Task einen andern mit niedrigerer Priorität, auch vor Ablauf seines "Time-Slices". Normale Applikations-Tasks laufen in der Regel mit niedrigster Priorität.

10.6.1 Source-Files importieren

Das Beispiel `Worki.cpp` zeigt, wie ein Task aufgesetzt wird. Binden Sie den Worki-Task wie folgt in Ihr neues Projekt ein:

- Klicken Sie mit der rechten Maustaste auf das Verzeichnis `Applikation` in der Projektstruktur *(siehe: Pos. 4, Abb. 4.1)*
- Fügen Sie einen Ordner mit Namen `SRC` ein: `Add Node -> Folder` für Programm Code
- Fügen Sie einen Ordner mit Namen `INC` ein: `Add Node -> Folder` für Header-Datei
- Wechseln Sie in das Verzeichnis `SRC`
- Fügen Sie das File `c:\IMD\Program\INOS\Samples\Imp_mas\FirstStep \Applicat\Src\Worki.cpp` ein: `Add Node -> File`

Mit **F8** wird das Projekt übersetzt und in den IMP-Master geladen. Im C++ Task-Fenster *(siehe: Pos. 2, Abb. 4.1)* erscheint der Worki Task nach erfolgreichem Download.

> Für weitere Hilfe suche in der Online-Hilfe nach:
> - `CINOSTask,`
> - `CINOSTask ClassMembers`

10.6.2 Task Starten, Stoppen, Rechenleistung abgeben

Resume()
Startet den Task.

Suspend()
Stoppt den Task. Der Task kann jederzeit mit Resume() wieder gestartet werden.

Relinquish()
Gibt vorübergehend Rechenleistung ab. Dieser Mechanismus ermöglicht es die Rechenzeit des Prozessors unter mehreren Tasks mit der selben Priorität zu verteilen. Ruft ein Task diese Funktion auf, werden alle READY-Tasks mit der selben Priorität ausgeführt bevor der ursprüngliche Task wieder Rechenzeit erhält. Diese Funktion kann z.B. in Schleifen aufgerufen werden.

Sleep()
Gibt für eine bestimmte Zeit Rechenleistung ab. z.B. Sleep(500) hat zur Folge, dass dieser Task 500ms lang nicht mehr aufgerufen wird. Sleep(0) entspricht Relinquish().

> Für weitere Hilfe suche in der Online-Hilfe nach:
> - `Resume`
> - `Suspend`
> - `Relinquish`
> - `Sleep`
> - `Ready`

10.7 1ms Handler

Das Betriebssystem erzeugt beim Hochstarten einen Task der exakt jede Millisekunde aktiviert wird. Der Anwender hat nun die Möglichkeit, diesem Task verschiedene Aufgaben zu übergeben. So ist es z.B. möglich, Funktionen zu registrieren, die alle x ms aufgerufen werden. Diese Funktionen laufen alle synchron zum zentralen 1ms Tasks.

Wichtig
Da das Betriebssystem diesen Mechanismus auch verwendet (z.B. von CINOSTask::Sleep( )) muss darauf geachtet werden, dass innerhalb dieser Funktionen nicht auf ein Ereignis gewartet wird, da ansonsten der ganze 1ms Task blockiert wird und nachfolgende Funktionen nicht mehr aufgerufen werden.

Member Funktionen registrieren

```
void CMy1msTask::My1ms()
{

}

// constructor
DeclareClassMemberPointer(CMy1msTask, My1ms);
pINOS1ms->AddHandler(ClassMemberPointer(CMy1msTask, My1ms), this,2);
```

Die Funktion `My1ms` wird alle 2ms aufgerufen.

Variable jede Millisekunde dekrementieren

```
int32 g_MyVariable;

// decrement (-1) variable g_MyVariable each ms until it's 0
g_pINOS1ms->AddTimer(&g_MyVariable);
```

Die Variable `g_MyVariable` wird jede Millisekunde um eins dekrementiert, bis sie 0 ist.

Das Beispiel `1ms.cpp` zeigt die Anwendung der Memberfunktionen der Klasse `CINOS1ms`.

> Für weitere Hilfe suche in der Online-Hilfe nach:
> - `CINOS1ms`

10.8 Module ansprechen

10.8.1 Digitale Ein- Ausgänge ansprechen

Mit der Klasse `CINOSBit` können einfach einzelne Ein- und Ausgänge oder Flags verändert werden. Sie wird für den Zugriff auf die digitalen Ein- / Ausgänge sowie auf die Flags verwendet. Ein weiterer Vorteil besteht darin, dass auf eine Signal-Flanke ohne Belastung der CPU gewartet werden kann. Siehe auch Beispiel `FirstStep.cpp`.

```
#include <CINOSProcessImage.h>

// Suchen des digitalen Einganges „MyInput"
CINOSBit *pMyInput = g_pInputs->Find(„MyInput");
// Test auf die Gültigkeit des Einganges
ASSERT_ALWAYS(pMyInput);
// Nun warten bis der Eingang gesetzt wurde
pMyInput->WaitForSet();
//...
pMyInput->WaitForCleared();
```

Für weitere Hilfe suche in der Online-Hilfe nach:
- `CINOSBit`
- `g_pInputs, g_pOutputs`
- `CINOSBit class members`

10.8.2 Analoge Ein- Ausgänge ansprechen

Analog zu den digitalen Ein-Ausgängen werden die analogen I/Os angesprochen. Dazu wird die Klasse `CINOSDacChannel` und `CINOSAdcChannel` verwendet. Die globalen Pointer auf das Prozessabbild heissen: `g_pDacChannels, g_pAdcChannels`.
Siehe auch Beispiel `Analog.cpp`.

```
#include <CINOSProcessImage.h>

// Suchen des analogen Ausganges „MyDAC"
CINOSDacChannel *pMyDAC = g_pDacChannels->Find(„MyDAC");
// DAC Ausgang setzten
pMyDAC->SetValue(Value);

// Suchen des analogen Einganges „MyADC"
CINOSAdcChannel *pMyADC = g_pAdcChannels->Find("Input0");
ASSERT_ALWAYS(pMyADC);
// ADC Ausgang lesen
real32 value = pMyADC->GetValue();
```

Für weitere Hilfe suche in der Online-Hilfe nach:
- `CINOSDacChannel, CINOSAdcChannel`
- `g_pAdcChannels, g_pDacChannels`

10.8.3 Positions Kanäle ansprechen

Positionskanäle verwenden die Klasse `CINOSPosChannel`. Der globale Pointer auf das Prozessabbild heisst: `g_pPosChannels`.

```
#include <CINOSMotors.h>
// Suchen des Positions-Kanals „MyPosition"
CINOSPosChannel *pMyPosition = g_pPosChannels->Find("Position0");
ASSERT_ALWAYS(pMyPosition );
// Positions Kanal lesen
int32 Pos = pMPosition->GetPosition();
pMPosition->SetPosition(Pos,true);
```

> Für weitere Hilfe suche in der Online-Hilfe nach:
> - `CINOSPosChannel`
> - `g_pPosChannels`

10.9 Timer

Die CINOSTime Klasse wird vorwiegend zur Zeitmessung verwendet. Siehe auch Beispiel `Timer.cpp`.

```
#include <inos_tim.h>
// Erzeuge das Messobjekt MyTime
CINOSTime MyTime;

// Die Messzeit wird hier gestartet
MyTime.InitTime();

// Zeitmessung für 1'000 Additionen
for(n=0;n<1000;n++)
    Sum = Sum +n;

// Holen der zeit
MeasureTime = MyTime.ElapsedUS();

// Die Messzeit wird hier wieder gestartet
MyTime.InitTime();

// Zeitmessung für 1'000'000 Divisionen
for(n=1;n<1000001;n++)
    Sum = Sum / 200.123;

// Holen der Zeit in us
MeasureTime = MyTime.ElapsedMS();
```

Während der Messung müssten die Interrupts gesperrt werden.

10.10 Achsen

Mit Hilfe dieser Klasse können Motoren aller Art angesprochen werden. Sie bietet eine einheitliche Schnittstelle für Schrittmotor- und Servo-Achsen wie INFO-HCSr, IMP-SMC, IMP-SMI und IMP-DAC mit IMP-INC.

Die meisten Funktionen können synchron oder asynchron aufgerufen werden, d.h. man kann mit Hilfe eines Parameters definieren, ob die Funktion sofort zurückkehrt (asynchron) oder erst wenn der Befehl abgearbeitet wurde (synchron).

Entscheidet man sich für die asynchrone Version, liefert die Funktion einen Zeiger auf ein Sync-Object zurück, mit Hilfe dessen auf das Funktionsende gewartet werden kann.

Beispiel

```
#include <CINOSMotors.h>

// try to find motor with name 'XAchse'
CINOSMotor* motor = pINOSMotors->Find(„XAchse");
// stop if not found
ASSERT_ALWAYS(motor)
// get pointer to motor contorller
CINOSMotorController* controller = motor->GetController();

// activate controller
controller->Activate();

// do synchronous move to absolute position 10.0
controller->Absolute();
controller->Move(10.0, false);

// do asynchronous move to absolute position 20.0
CINOSSync* sync = controller->Move(20.0, true);
// wait until we are there
sync->Wait();
```

Für weitere Hilfe suche in der Online-Hilfe nach:
- CINOSMotorController
- Kapitel 10.12.1.2 Achen bewegen
- Kapitel 6 im "Regler Manual AC-Servo Kontroller"

10.11 Variablen, Funktionen registrieren

10.11.1 Variablen registrieren

Der Zugriff aus der Visualisierung oder aus Indel-Tools auf Werte und Variablen im Zielsystem geschieht über ihren Namen. Der Benutzer muss sich nicht mehr um Adressen oder Memory Bereiche kümmern. Damit kann der Programmierer der Visualisierung die Prozessvariablen, die der Steuerungsprogrammierer definiert hat, sofort und ohne Umwandlungsprozeduren direkt auf der Visualisierungsseite übernehmen.

Für die netzwerkweite Verbreitung der Prozessdaten sorgt der sog. INCO-Server, der auf jeder beliebigen Station im Netzwerk installiert werden kann.

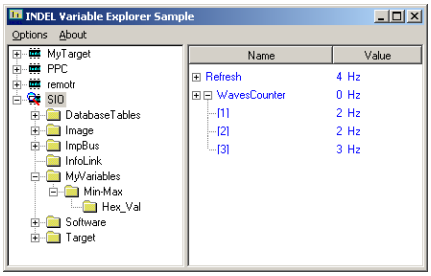


Abb 10.11.1 Variablen registrieren

Damit Variablen und Parameter aus dem Variablen-Explorer oder aus der Visualisierung betrachtet und verändert werden können, müssen diese *registriert* werden. CINCOItem ist die Basisklasse aller Variablen, welche im Variablenexplorer dargestellt werden können.

Siehe auch Beispiel `Variable.cpp`.

```
#include <Inco_itm.h>

// Register WavesCounter (array) with unit "Hz", range 0-1000000
// The variable appears in the "root directory"
pINCOItems->Add(new CINCOuint32("WavesCounter", m_uWavesCounter,
         0,1000000,"Hz",defCharShowDec,0,ARRAY_SIZE(WavesCounter)));

// Add a new directory "My Variables"
CINCOObject* MyVariables = new CINCOObject("MyVariables", this);
pINCOItems->Add(MyVariables);

// Add a sub-directory "Min-Max"
CINCOObject* Min_Max = new CINCOObject("Min-Max", 0);
MyVariables->Add(Min_Max);

// Register Variables
MyVariables->Add(new CINCOint32("Value 0",&m_uValue,0,0,65000,"mV"));
```

Für weitere Hilfe suche in der Online-Hilfe nach:
- `pINCOItems`
- `defCharShowExp`
- `SHOW_DIGIT`
- `CINCOuint32`

10.11.2 Funktionen registrieren

Nebst Variablen können auch Funktionen registriert, und über den Variablen-Explorer aufgerufen werden. Ein Doppelklick auf den "Function-Value" aktiviert die Funktion.

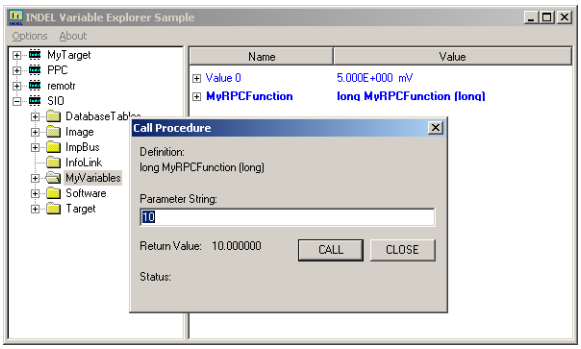


Abb 10.11.2 Funktionen registrieren

Siehe auch Beispiel `Variable.cpp`.

```
int32 MyRPCFunction(int32 aCommand)
{
     // Ausführen der Funktion // ...
     return aCommand;
}

StartFunction(_INI_1000_RPCFunction)
{
     // Registriert die globale Funktion MyRPCFunction
     pINCOItems-Add(new CINCOProcedure
                         (REG_TYP_FCT(int32,MyRPCFunction,(int32))));
}
```

Für weitere Hilfe suche in der Online-Hilfe nach:
- `pINCOprocedure`

Achtung
In Funktionen die von der Visualisierung aufgerufen werden, dürfen keine Breakpoints gesetzt werden. Diese Funktionen sind auch nicht debugbar! Ansonsten wird das gesamte System angehalten!

10.12 Visualisierung, INCO-Schnittstelle

Die Kommunikation zwischen PC und Feldbusmaster baut auf einfachen Funktionen auf:

`PutVariable`	schreibe eine Variable in den Master und
`GetVariable`	hole eine Variable vom Master

Sämtliche Funktionen, die es ermöglichen auf ein Target zuzugreifen, sind in der INCO-DLL (INCO_32.DLL, Dynamic Link Library) enthalten. Alle gängigen Visualisierungs-Tools können auf Indel Targets zugreifen, sofern diese eine DLL einbinden können.

Das INFO-Link Steuerungssystem unterstützt diverse gängige Visualisierungs-Systeme:

- Lab View (National Instruments)
- InTouch (Wonderware)
- Intellution (Intellution)
- Visual Basic (Microsoft)
- Visual C++ (Microsoft)
- Viscontrol (Viscom AG)
- Delphi (Borland)

10.12.1 Beispiel in Visual Basic

10.12.1.1 Eingänge lesen

Um die Zugriffszeit auf die Eingänge von 8 IMP-8PIN Karten zu optimieren, können alle Eingänge mit einem GetBlock32() auf einmal (anstatt jeder Eingang einzeln) gelesen werden.
Konfiguration:

- INCO-Server installiert und gestartet
- PowerPC mit geladenem Projekt (IO-Karten konfiguriert)
- Alle Eingangskarten liegen im Speicher hintereinander

> Für weitere Hilfe suche in der Online-Hilfe nach:
> - `Inco-Server`
> - `VB-Beispiel`

VB-Beispiel (Verwendung Klassenmodul INCO_32.BAS)

```
' Code in beliebiger Form reagiert auf das Ereignis Click eines Buttons
Private Sub GetInputs_Click()
     On Error GoTo ERROR_HANDLER
     Dim lInputs(1 To 2) As Long
     Dim lError As Long, lInputAddress As Long
     Dim sTargetName As String
     sTargetName = „PPC"
     lError = 0

     ' holen der Basisadresse der digitalen Eingänge
     lError = GetVBVariable(sTargetName, „Digital.Inputs!Address",
                                                          lInputAddress)

     ' alles in Ordnung? Fehlerfrei?
     If lError Then GoTo ERROR_HANDLER

     ' holen der ersten 64 (2*32-Bit-Karten) Eingänge
     lError = GetBlock32(sTargetName, lInputAddress, lInputs(1), 2)

     ' alles in Ordnung? Fehlerfrei?
     If lError Then GoTo ERROR_HANDLER

     MsgBox „Inputs from the second 32-bit card:
                              0x" & Hex(lInputs(2)), vbOKOnly, „Message"
     Exit Sub

ERROR_HANDLER:
     Dim sErrorMsg As String

     If lError Then
          ' holen des Fehlertextes
          Call GetVBErrorDescription(sTargetName, lError, sErrorMsg, 100)
     End If

     ' Ausgabe des Fehlertextes
     MsgBox sErrorMsg + Err.Description, vbExclamation + vbOKOnly, „Error"
End Sub
```

Achtung

Wegen der Little- / Big-Endian Problematik funktioniert obiges Beispiel nur mit der GetBlock32() Funktion. Bei den Funktionen GetBlock8(), GetBlock16() und GetBlock64() stimmt die Reihenfolge der Bytes nicht!

10.12.1.2 Achsen bewegen

Aus dem Variablen-Explorer oder dem INCO-Explorer können alle Achen in Betrieb genommen werden. Unter `"Axis.MyAxis"` sind alle konfigurierten Achsen aufgelistet. Die Erklärung dieser Schnittstelle finden Sie im File `"Regler Manual"`.

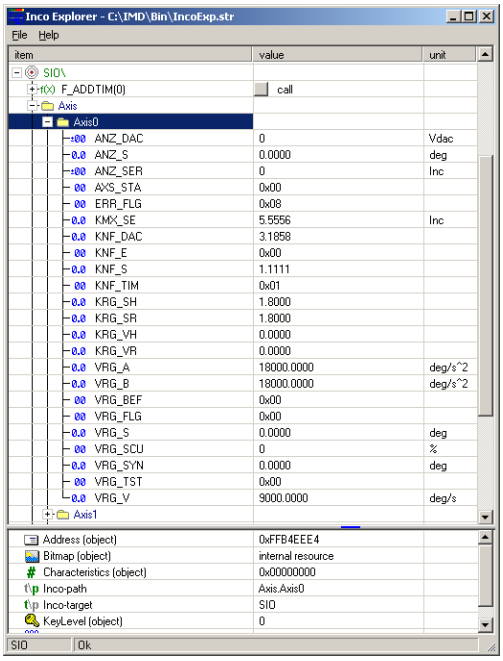


Abb 10.12.1 Interface für Achsen

Diese universelle Schnittstelle kann auch benützt werden um aus der Visualisierung die Achsen im System zu bedienen.

Mit dem Programm "Win-Show" bzw. "ACS-Show" (für Servo-Regler) können die Achsen ebenfalls bedient werden. Beide Programme sind in VB geschrieben und greifen über diese Schnittstelle auf die Achsen zu.

Siehe auch Kapitel 6 im "Regler Manual AC-Servo Kontroller"

10.12.3 Beispiele in C++

Siehe Beispiele im Verzeichnis `\IMD\Program\VisualC++`

10.12.3 Beispiele in Delphi

Siehe Beispiele im Verzeichnis `\IMD\Program\Delphi`

A Fehlermeldungen

A.1 Fehlerquellen beim Downloaden

0x06BA: `Contacting MyTarget failed, error: INCO-Server not found`

Problem
Der INCO-Server muss gestartet sein: `INCO-Server` in der Programm-Auswahl.

0x70001: `Contacting MyTarget failed, error: Error initialysing COM`

Problem
Der INCO-Server versucht den COM-Port, der im Registrierungs Editor dem Target `MyTarget` zugeordnet wurde, zu öffnen.

Ursache
- Der COM-Port existiert physikalisch nicht.
- Der COM-Port wird bereits von einem anderen Gerät benützt.

0x70002: `Contacting MyTarget failed, error:`

Problem

Ursache

0x10003: `Contacting MyTarget failed, error: Target not defined`

Problem
Der INCO-Server sucht ein Target namens `MyTarget` in der Registrierung, kann es aber nicht finden. (Der Name `MyTarget` wurde in den Environment Variables als Target für das aktuelle Projekt eingetragen.)

Ursache
- Es existiert kein Target namens `MyTarget`. Verwenden Sie den Registrierungs Editor `INCO-Registry` in der Programm-Auswahl.

0x70004: `Contacting MyTarget failed, error: Timeout reading from COM`

Problem
Der INCO-Server versucht mit einem Target namens `MyTarget` über die konfigurierte serielle Schnittstelle Kontakt aufzunehmen, kann es aber nicht finden.

Ursache
- Serielles Schnittstellenkabel ok? Nullmodem-Kabel verwenden.
- Serielles Kabel am richtigen COM-Port eingesteckt?
- Speisung eingeschaltet?

A.2 Fehler bei Power-Up der IMP-Steuerung

Bei Power Down während dem Brennen der Flash-Prom werden nicht alle Sektoren ordnungsgemäss beschrieben. Danach kann der IMP-Master das geladene Programm nicht ausführen.

Damit der IMP-Master trotzdem wieder gestartet werden kann, befindet sich in einem Sektor des Flash-Prom ein Notsystem, das minimale Funktionen, z.B. Kommunikation zum PC unterstützt.

Das Notsystem prüft den Code im Flash-Prom mit einer Check-Summe auf Fehler bevor er gestartet wird. Damit wird sichergestellt, dass ein unvollständig gebranntes Programm nicht gestartet wird. Ist dies der Fall verweilt der IMP-Master im Notsystem.

Kann aus irgend einem Grund ein fehlerhaftes Programm nicht erkannt werden, und der IMP-Master startet trotztdem stürtzt der Master ab. In diesem Fall kann der Master mit einem Kurzschluss-Stekker gezwungen werden nur das Notsystem zu starten.

Folgende Verbindungen müssen am Seriellen Stecker auf dem IMP-Master hergestellt werden:

Verbindungen:	Signale	Pin	
	RxD, TxD	2, 3	Drahtbrücke
	DSR, DTR	6, 4	Drahtbrücke

Nachdem der IMP-Master aufgestartet ist, kann der Kurzschlussstecker entfernt werden und das serielle Kabel zum PC kann wieder eingesteckt werden.

B Mitgeltende Unterlagen

Online Hilfe
IMD-Hilfesystem: C/C++ Hilfe, Klassen, Applikationsbeispiele, Tools, usw.

Hardware Ordner
Dokumentation der IMP-Module
Steckerbelegung, Adressierung, Anschlussbeispiele, usw.

INDEL Verdrahtungs-Richtlinie:
Vorschriften für eine EMV-Gerechte Verdrahtung und Installation in Maschinen und Anlagen

INDEL Aufbaurichtlinie
Vorschriften für eine EMV-Gerechte Verdrahtung der IMP-Module

C Online Dokumentation

Sämtliche Dokumentationen sind im Internet unter

http://www.indel.ch/

publiziert. Wählen Sie die Seite "Dokumentation" und danach "IMP". Sie können die neuesten Doku-Versionen als kostenloses PDF-File herunterladen.

Dieses Handbuch befindet sich auch auf der CD-ROM:

```
"d:\IMP\Software\Deutsch\10Schritte.pdf"
```

Index

R

S

T

U

V

W

Z