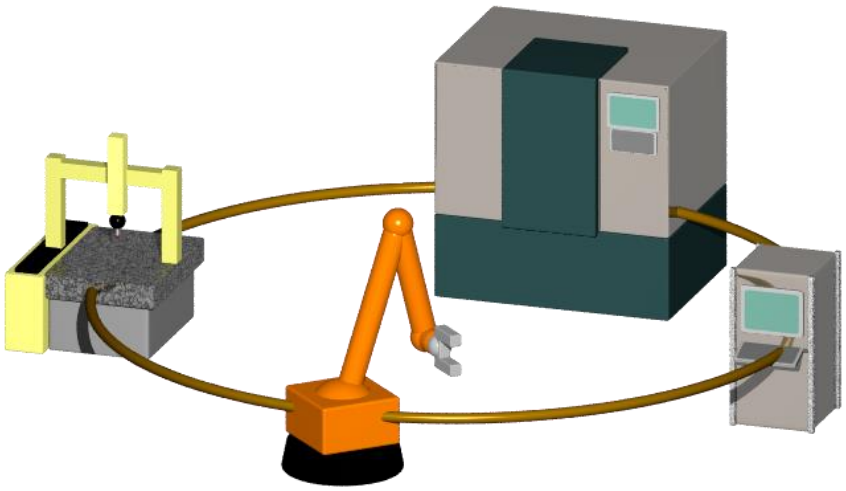

# IO-Communication and Automation

Exchange of Digital Signals between Control Systems
*and*
Integrated CMMs in production lines



Service Information
Author: Thomas Moch

Report version: 3.51
Date: 2019-03-15

# 1 Content

# 2 The Purpose of IO-Communication

IO-Communications allow control systems to communicate with each other through the transfer of electronic information. The communication can be unidirectional as well as bi-directional.

***Examples:***

### Visual Signals with Patrol Lights

*MCOSMOS controls lights using an IO-Card (generally called patrol lights). With one light (e.g. green) it can be shown that the measurement machine is working (that a part program is executing). With another light (e.g. red) it can be shown that the measurement machine is unoccupied or has finished. You can also utilise further electrical signals, for example to indicate whether the measured part is inside or outside of tolerance. Or the CMM is in a failure condition.*

### Loading Management for Pallets

*Information about the state of the system can be communicated, specifying for example, information about where the pallets or parts are, if the machine's working volume is occupied, free or obstructed, and even if the part has failed the tolerance criteria of measurement or not. Where the patrol lights mentioned before only require a signal to be sent, with the Loading Management, you can manage the timing of events, for example, waiting for signals indicating if the loading system has finished loading the part or pallet.*

### Fully Automated Production Line

*This is similar to Loading Management, however many more signals must be managed, loading, placing, etc. In addition, RemoteManager is needed to start different part programs.*
***Please see the special chapter Fully Automated Production Line.***

# 3 Procedure for a New Customer Project

The following questions must be considered for customer projects:

- **Who should the project manager be?**
  *This is the most important and difficult question, since a single person must be responsible for taking on all the project information, requesting the project status and coordinating the timeline. He must know who is doing what, coordinate the work and manage questions between the parties and determine links and connections between the involved parties. This person should normally be part of the team, undertaking the assembly and setting-up of the automated production and measurement machines.*
- **What should be checked?**
  *Switches, Feed Arrangement, Loading Device and/or other electronic features.*
- **What is the main automation workflow?**
  *IO handshaking with the placement system, starting the part program, returning the results and or reports. Please see also following sub-chapter.*
- **Which IO-card should be used?**
  *This depends on what must be checked, and the application environment (computer and operating system).*
- **Who should install the interface connector cables?**
  *Preferably the supplier of the hardware. It would be ideal however, if the project manager could oversee the installation.*
- **Which ports receive which signals, how the switches should be set, and (most importantly) what should be done and when?**
  *Which ports receive which signals and how the switches should be set, may all be freely assigned. Only "when" the IO-Communication will be executed, is determined by the program (for example: start of part program). That means that the program cannot be interrupted by IO-Communication at any time.*
  *Yet, it should be documented, what arrives at which switch and what will be controlled, and this is best documented in the "file header" of the IO-Communication file e.g.:*

```
output - port 1 - switch 1:   "high" - GEOPAK activated (program is running)
output - port 1 - switch 2:   "high" - GEOPAK in idle mode
output - port 1 - switch 3:   "high" - GEOPAK in repeat mode
output - port 1 - switch 4:   "high" - failure (crash) occurred
```

***For tips and hints from the practice, it is recommended to read the chapter "I/O Experience from the UK"***

### 3.1 Example for a main workflow

*Please note that this is only a workflow example for a half automated system in the case that "MCOSMOS Initiates the Placement Process". This example is not a recommended solution.*

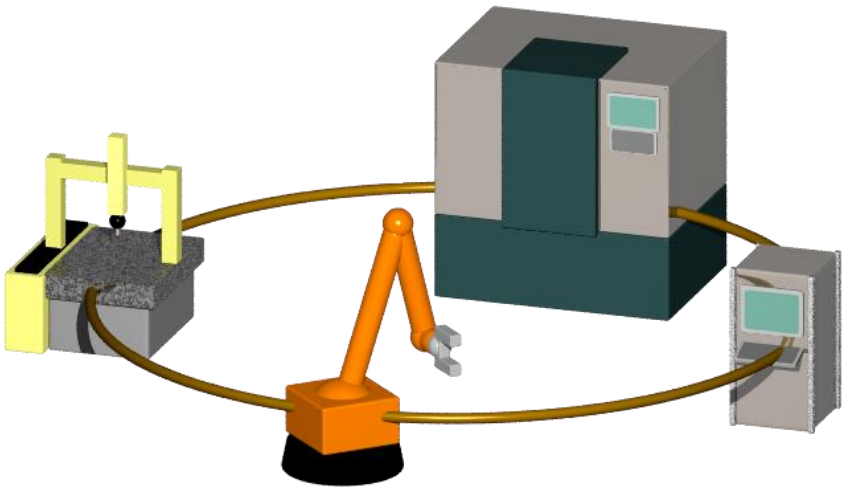
- Blue indicates MCOSMOS part
- Orange indicates the placement device part

1. *CMM is at home position*
2. *Part program started by user via PartManager*
3. *Signal to robot „Free for charging"*
4. *Robot take part from place 1*
5. *Robot put part on CMM*
6. *Robot moves out of CMM working area*
7. *Signal to CMM "Charging finish" and "Out of working area"*
8. *Remove signal to Robot "Free for charging"*
9. *Part is measured.*
10. *Measuring is finish.*
11. *Signal to Robot "Good or Bad" part*
12. *Signal to robot „Free for charging"*
13. *Robot take part from CMM*
14. *Robot moves out of CMM working area*
15. *Robot put part depending from result on table 1 or 2.*
16. *Signal to CMM "Charging finish" and "Out of working area"*
17. *Remove signal to Robot "Free for charging"*

*xx.* *Repeat from point 1 up to 17*

# 4 Fully Automated Production Line

When Mitutoyo 3D CMMs are integrated into automated production lines, the integration process is fairly similar in most cases, with only small details changing. This document aims to make it easier for all Mitutoyo departments to find a good solution for such automation projects.



**MCOSMOS uses the following methods of inter-process communication:**

- Handshaking with the Loading Device is performed with IO-Signals through the IO-Card.
- The external Process Control PC can use RemoteManager on the local PC to start a part program. This is achieved by sending an ASCII-File with a handshake.
- To return the report and/or result data standard part program commands are normally used. These commands are also used for the correction data. Sometimes a program such as CORRECT+ is installed to convert and transfer the data to the production machine.

## 4.1 Components of an Automated Production Line

### Production Machine

*E.g. CNC milling machine or CNC turning machine*
*We don't normally communicate with a production machine directly.*

### Feeding Arrangement/Loading Device

*To place or remove the part and/or pallet on CMM table, a robot and/or Assembly line may be used.*

### An Optional Process Controller

*The Process Controller is normally required to manage the entire automation process.*
*Usually a PLC is used.*

### Mitutoyo 3D CMM

#### + IO-Card or Hilscher FieldBus-Card

*This is crucial for the placement handshake.*

#### + Home-Position-Switch

*The home-position-switch is crucial in ensuring a secure and collision free loading process. It will cost much less to buy and fit this switch than it would to repair a CMM.*
*The switch is placed on the CMM to verify that the CMM is in the Home-Position. It must transmit a high-signal to the IO-Card when the home-position is reached. If required, this signal may also be sent to other external systems like the part placement system. Only when this and other signals (depending on the other IO-Communication-Processes e.g. "Free for placement signal from GEOPAK") are set, is the placement system (e.g. robot) allowed to enter into the space of the CMM. This switch can be installed by KOMEG.*

### Optional Pallet Contact Switch(s)

*To verify that the pallet and its measurement part are correctly placed on the CMM table this switch(s) would be used. Its contacts send a high signal to the IO-Card and/or to the placement device, so that the correct placement of the pallet can be verified. This can contain one or more signals, dependent on the automation hardware. Although there would ordinarily be just be one signal sent from the supporting hardware logic (e.g. PLC), it could be useful however for the placement device to have access to more signals, so that it may, for example, more conveniently advance or prepare placement of the next part. This switch(s) can be installed by KOMEG.*

### Optional Cabin

*A cabin could be needed, e.g. for human protection or climate control during the inspection process. Such a cabin can be built and fitted by KOMEG.*

### Optional Hardware Switch (Automatic / Manual Mode)

*Sometimes the CMM needs to suspend the inspection and placement process; typically for servicing or calibration. For this purpose an additional hardware switch is very effective, and will signal to the placement device that is has no access to the CMM; e.g. for placement or removal.*

### Optional Patrol Lights

*These lights could be required to indicate the status of the machine to the user.*

### Optional Return of Correction Data

*If required, a software solution would typically be used (e.g. CORRECT+).*

## 4.2 Main Workflows

Two main workflows exist, and they are both independent from optional hardware such as pallet, switches, etc.):

- **The process control PC initiates the placement process (recommended).**

- **MCOSMOS initiates the placement process.**

## 4.3 The Process Control PC Initiates the Placement Process

Starting the part program and placing the part are two separate processes.

*Placement:*

- During GEOPAK's idle process, it receives a signal to inform it that the placement system will place or remove the part. An IO handshake process will then occur.

*Starting:*

- The part program may be started either automatically by RemoteManager or manually by PartManager.
- GEOPAK then runs the part program and the results and report are then created.
- If the part program was started by RemoteManager, the completion of the part program will be reported back to RemoteManager.

## 4.4 MCOSMOS Initiates the Placement Process

- The part program may be started by either RemoteManager or PartManager.
- Before starting the part program GEOPAK initiates the placement via IO signals, and performs the handshake.
- GEOPAK runs the part program and the results and report are created.
- When the part program is complete, GEOPAK initiates the removal process via IO signals and performs the handshake.
- If the part program was started by RemoteManager, the completion of the part program will be reported back to RemoteManager.

## 4.5 Which Signals Are needed?

**Input Signals (Mitutoyo PC – IO-Card):**

- Automation active (from Process Control) (High = Active)
- Hardware Home Control (from Switch, Sensor) (High = Home)
- Placement device is not in the working area of the CMM (High = Outside)
- Placement device requests placement (High = Request)
- Part placement finished (High = Finished)
- Optional: Manual or automatic mode hardware switch (High = Automatic)
- Optional: Pallet switch(s) for the correct placement (High = Correct)

**Output Signals (Mitutoyo PC – IO-Card):**

- Automation active (from CMM) (High = Active)
- CMM is at stand by (in home position and no part program is running) (High = Stand By)
- After the placement device's request, free for placement (High = Free)
- Irrecoverable error during part program repetition (High = Error)
- Optional: Part program running (High = Running)
- Optional: Result good/bad part (High = Good)
  *Often the Process Control PC or the placement device requires a good or bad part determination after inspection, so it can select the correct manoeuvre for the part.*

## 4.6 Recommended Solutions

### 4.6.1 Process Control PC Initiates Loading Process

Input Signals:

- Automation active (from Process Control) (High = Active)
- Hardware Home Control (Switch, Sensor) (High = Home)
- Loading device is not in the CMM's working area (High = Outside)
- Loading device requests placement (High = Request)
- Loading finished (High = Finished)

Output Signals:

- Automation active (from CMM) (High = Active)
- CMM is at stand by (in home position and no part program is running) (High = Stand By)
- After the loading device's request, available for loading (High = Available)
- Irrecoverable error during part program repetition (High = Error)

The signals can have the following settings (from the IO_COND Header):

```
rem * Input (at CMM, I/O card):
rem * Automation active            DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch      DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area     DI_A2 -> I p1 S3  1 = Out
rem * L.D. request loading         DI_A3 -> I p1 S4  1 = Request
rem * Loading finished             DI_A4 -> I p1 S5  1 = Finished
rem *                              DI_A5 -> I p1 S6
rem *                              DI_A6 -> I p1 S7
rem *                              DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active            DO_A0 -> O p1 S1  1 = Active
rem * CMM is in Idle                DO_A1 -> O p1 S2  1 = Idle
rem * Free for loading             DO_A2 -> O p1 S3  1 = Free
rem * Not Recoverable Error        DO_A3 -> O p1 S4  1 = Error
rem *                              DO_A4 -> O p1 S5
rem *                              DO_A5 -> O p1 S6
rem *                              DO_A6 -> O p1 S7
rem *                              DO_A7 -> O p1 S8
```

When GEOPAK starts and ends, all signals must be set to zero; however automation must be set to high at the start and low at the end.

```
condition 1
rem (RPT: Start of program)
set p1 s10000000
end

condition 20
rem (RPT: End of program)
set p1 s00000000
end
```

If GEOPAK is not executing a part program, it will be waiting in idle mode for the part placement and for the handshake to be performed.

```
condition 16
rem (RPT: Interrupt if not runs)
```

```
rem Check Home Position
if I p1 sx1xxxxxx
  set p1 sx1xxxxxx
if I p1 sx0xxxxxx
  set p1 sx0xxxxxx

rem When no loading request do nothing
if not I p1 s11x1xxxx
  goto EndOfCond

rem CMM free for loading
set p1 sx11xxxxx

rem Wait for loading and L.D.out of area
brk p1 s11x1xxxx
if err
  goto ErrCond

rem Loading no more allowed
set p1 sxx0xxxxx
goto EndOfCond

rem Error
lbl ErrCond
rem Error and not free for loading
set p1 sxx01xxxx

lbl EndOfCond
end
```

A measurement program may now be started, either automatically by RemoteManager or manually by the user. When the part program starts and finishes, it will check the status of the system. For safety reasons the placement device's position is also checked.

```
condition 3
rem (RPT: Before starting a part program repetition)

rem Check again Loading Device
brk p1 s1x1xxxxx

rem CMM not free and loading not allowed
set p1 sx000xxxx
end

condition 10
rem (RPT: After exiting the last part program repetition)
set p1 sx00xxxxx
end
```

If an error occurs during inspection, an error condition is invoked internally, thereby causing the error output signal to be set:

```
condition 8
rem (RPT: Error in part program)
set p1 sxxx1xxxx
end

condition 9
rem (RPT: Reset System)
set p1 sxxx0xxxx
end
```

#### 4.6.1.1 With Additional Pallet and/or Cabin Switches

We need one or both of the following input signals:

- Pallet sensor(s) for correct part placement (High = Placed)
- Cabin Closed Sensor (High = Closed)

In the following example however, the same input signal (marked in red) is used for both the pallet sensor(s) and the cabin closed sensor, because the meaning is identical.

```
rem * Input (at CMM, I/O card):
rem * Automation active             DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch       DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area      DI_A2 -> I p1 S3  1 = Out
rem * L.D. request loading          DI_A3 -> I p1 S4  1 = Request
rem * Loading finished              DI_A4 -> I p1 S5  1 = Finished
rem * Pallet placed and Cabin close DI_A5 -> I p1 S6  1 = placed and Closed
rem *                               DI_A6 -> I p1 S7
rem *                               DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active             DO_A0 -> O p1 S1  1 = Active
rem * CMM is in Idle                DO_A1 -> O p1 S2  1 = Idle
rem * Free for loading              DO_A2 -> O p1 S3  1 = Free
rem * Not Recoverable Error         DO_A3 -> O p1 S4  1 = Error
rem *                               DO_A4 -> O p1 S5
rem *                               DO_A5 -> O p1 S6
rem *                               DO_A6 -> O p1 S7
rem *                               DO_A7 -> O p1 S8
```

Conditions 1, 8, 9, 10 and 20 must not be changed.
It is essential to the placement process to check this additional signal. The pallet must be correctly placed and/or the cabin must be closed.

```
condition 16
rem (RPT: Interrupt if not runs)

rem Check Home Position
if I p1 sx1xxxxxx
  set p1 sx1xxxxxx
if I p1 sx0xxxxxx
  set p1 sx0xxxxxx

rem When no loading request do nothing
if not I p1 s11x1xxxx
  goto EndOfCond

rem CMM free for loading
set p1 sx11xxxxx

rem Wait for finished loading and Loading Device out of area
rem Palett placed and cabin closed
brk p1 s11x11xxx
if err
  goto ErrCond

rem Loading no more allowed
set p1 sxx0xxxxx
goto EndOfCond

rem Error
lbl ErrCond
rem Error and not free for loading
set p1 sxx01xxxx

lbl EndOfCond
```

```
end
```

At the beginning of a part program (condition 3) this signal must be checked:

```
condition 3
rem (RPT: Before starting a part program repetition)

rem Check again Loading Device, Pallet and Cabin
brk p1 s1x1xx1xx

rem CMM not free and loading not allowed
set p1 sx000xxxx
end
```

#### 4.6.1.2 With Additional Good or Bad Part Signal

An output signal is used to communicate that the part is good or bad:

- Good/Bad Part (High = Good)

*I would always recommend using different signals for Good Part and Bad Part. A break in the cable could wrongly be interpreted as a Bad Part.*

This signal can be managed with the help of conditions 21, 22 and 23.

The signal can either be set during the inspection or at the end of it. In the following example, the signal is only set at the end of a part program (marked in red):

```
rem * Input (at CMM, I/O card):
rem * Automation active            DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch      DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area     DI_A2 -> I p1 S3  1 = Out
rem * L.D. request loading         DI_A3 -> I p1 S4  1 = Request
rem * Loading finished             DI_A4 -> I p1 S5  1 = Finished
rem * Pallet fit and Cabin close   DI_A5 -> I p1 S6  1 = Fit and Closed
rem *                              DI_A6 -> I p1 S7
rem *                              DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active            DO_A0 -> O p1 S1  1 = Active
rem * CMM is in idle               DO_A1 -> O p1 S2  1 = Idle
rem * Free for loading             DO_A2 -> O p1 S3  1 = free
rem * Not Recoverable Error        DO_A3 -> O p1 S4  1 = Error
rem * Good Part                    DO_A4 -> O p1 S5  1 = Good
rem * Bad Part                     DO_A5 -> O p1 S6  1 = Bad
rem *                              DO_A6 -> O p1 S7
rem *                              DO_A7 -> O p1 S8
rem
rem * Meaning of used memory BITs:
rem * BIT 11 = in tolerance
rem * BIT 12 = out of tolerance
```

At any start of a part program repetition (condition3) the signal and memory bits must be reset.

```
condition 3
rem (RPT: Before starting a part program repetition)

rem Reset OK/notOK Signals and memory bits
set p1 sxxxx00xx
bit 11 = OFF
bit 12 = OFF

rem Check again Loading Device, Pallet and Cabin
brk p1 s1x1xx1xx

rem CMM not free and loading not allowed
set p1 sx000xxxx
end
```

The tolerance calculation will determine which condition is called. The related conditions must then set the bits accordingly.

```
condition 21
rem (RPT: Feature within control limits)
if not bit 12
  bit 11 = ON
end

condition 22
rem (RPT: Feature within tolerance, but out of control limits)
```

```
same 21
end

condition 23
rem (RPT: Feature out of tolerance)
bit 11 = OFF
bit 12 = ON
end
```

At the end of the part program, the Good/Bad Part signal must be set according to the status of the bits.

```
condition 10
rem (RPT: After exiting the last part program repetition)

rem Set OK signal only if all OK
if BIT 11
  set p2 sxxxx10xx
if BIT 12
  set p2 sxxxx01xx

set p1 sx00xxxxx
end
```

#### 4.6.1.3 With an Additional Manual-Automatic Switch

Just one output signal is used to communicate either “automatic” or “manual”:

- Manual or Automatic Mode (High = Automatic)

For safety reasons all the previous signals must still be managed.

```
rem * Input (at CMM, I/O card):
rem * Automation active            DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch      DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area     DI_A2 -> I p1 S3  1 = Out
rem * L.D. request loading         DI_A3 -> I p1 S4  1 = Request
rem * Loading finished             DI_A4 -> I p1 S5  1 = Finished
rem * Pallet fit and Cabin close   DI_A5 -> I p1 S6  1 = Fit and Closed
rem * Manual/Automatic Mode        DI_A6 -> I p1 S7  1 = Automatic
rem *                              DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active            DO_A0 -> O p1 S1  1 = Active
rem * CMM is in idle               DO_A1 -> O p1 S2  1 = Idle
rem * Free for loading             DO_A2 -> O p1 S3  1 = free
rem * Not Recoverable Error        DO_A3 -> O p1 S4  1 = Error
rem * Good/Bad Part                DO_A4 -> O p1 S5  1 = Good
rem *                              DO_A5 -> O p1 S6
rem *                              DO_A6 -> O p1 S7
rem *                              DO_A7 -> O p1 S8
```

To manage the additional signal only condition 16 must be changed:

```
condition 16
rem (RPT: Interrupt if not runs)

rem Check Home Position
if I p1 sx1xxxxxx
  set p1 sx1xxxxxx
if I p1 sx0xxxxxx
  set p1 sx0xxxxxx

rem Manual Mode?
if I p1 sxxxxxx0x
  end

rem When no loading request do nothing
if not I p1 s11x1xxxx
  goto EndOfCond

rem CMM free for loading
set p1 sx11xxxxx

rem Wait for finished loading and Loading Device out of area
rem Palette placed and cabin closed
brk p1 s11x11xxx
if err
  goto ErrCond

rem Loading no more allowed
set p1 sxx0xxxxx
goto EndOfCond

rem Error
lbl ErrCond
rem Error and not free for loading
set p1 sxx01xxxx

lbl EndOfCond
end
```

### 4.6.2 MCOMOS Initiates the Placement Process

Input Signals:

- Automation active (from Process Control) (High = Active)
- Hardware Home Control (Switch, Sensor) (High = Home)
- The placement device is not in the working area of the CMM (High = Outside)
- Placement finished (High = Finished)

Output Signals:

- Automation active (from CMM) (High = Active)
- The CMM requests placement (High = Request)
- The CMM requests removal (High = Request)
- Irrecoverable error during part program repetition (High = Error)

The signals can have the following settings (from the IO_COND Header):

```
rem * Input (at CMM, I/O card):
rem * Automation active            DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch      DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area     DI_A2 -> I p1 S3  1 = Out
rem * Un.-/Loading finished        DI_A3 -> I p1 S4  1 = Finished
rem *                              DI_A4 -> I p1 S5
rem *                              DI_A5 -> I p1 S6
rem *                              DI_A6 -> I p1 S7
rem *                              DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active            DO_A0 -> O p1 S1  1 = Active
rem * Request loading              DO_A1 -> O p1 S2  1 = Request
rem * Request unloading            DO_A2 -> O p1 S3  1 = Request
rem * Not Recoverable Error        DO_A3 -> O p1 S4  1 = Error
rem *                              DO_A4 -> O p1 S5
rem *                              DO_A5 -> O p1 S6
rem *                              DO_A6 -> O p1 S7
rem *                              DO_A7 -> O p1 S8
```

When GEOPAK starts and ends, all signals must be set to zero; however automation must be set to high at the start and low at the end.

```
condition 1
rem (RPT: Start of program)
set p1 s10000000
end

condition 20
rem (RPT: End of program)
set p1 s00000000
end
```

An inspection program may now be started, either automatically by RemoteManager or manually by the user. When the inspection program starts and finishes, it will check the status of the system. GEOPAK will request that the part be placed at the start and be removed at the end. For safety reasons the CMM is moved to its home position before and after any placement and removal.

```
condition 3
rem (RPT: Before starting a part program repetition)

rem Drive to secure position
```

```
drv cnc x00??? y00??? z00???
wait cnc x00??? y00??? z00??? t00001

rem Check again CMM and Loading Device
brk p1 s111xxxxx

rem Request loading
set p1 sx10xxxxx

rem Wait loading finished
brk p1 sxx11xxxx

rem re-set loading request
set p1 sxx00xxxx
end

condition 10
rem (RPT: After exiting the last part program repetition)

rem Drive to secure position
drv cnc x00??? y00??? z00???
wait cnc x00??? y00??? z00??? t00001

rem Check again CMM and Loading Device
brk p1 s111xxxxx

rem Request unloading
set p1 sx01xxxxx

rem Wait unloading finished
brk p1 sxx11xxxx

rem re-set unloading request
set p1 sxx00xxxx
end
```

If an error occurs during inspection, an error condition is invoked internally, thereby causing the error output signal to be set:

```
condition 8
rem (RPT: Error in part program)
set p1 sxxx1xxxx
end

condition 9
rem (RPT: Reset System)
set p1 sxxx0xxxx
end
```

#### 4.6.2.1 With Additional Pallet and/or Cabin Switches

We need one or both of the following input signals:

- Pallet sensor(s) for correct part placement (High = Placed)
- Cabin Closed Sensor (High = Closed)

In the following example however, the same input signal (marked in red), may be used for both the pallet sensor(s) and the cabin closed sensor, because the meaning is identical.

```
rem * Input (at CMM, I/O card):
rem * Automation active            DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch      DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area     DI_A2 -> I p1 S3  1 = Out
rem * Un-/Loading finished         DI_A3 -> I p1 S4  1 = Finished
rem * Pallet fit and Cabin close   DI_A4 -> I p1 S5  1 = Fit and Closed
rem *                              DI_A5 -> I p1 S6
rem *                              DI_A6 -> I p1 S7
rem *                              DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active            DO_A0 -> O p1 S1  1 = Active
rem * Request loading               DO_A1 -> O p1 S2  1 = Request
rem * Request unloading             DO_A2 -> O p1 S3  1 = Request
rem * Not Recoverable Error        DO_A3 -> O p1 S4  1 = Error
rem *                              DO_A4 -> O p1 S5
rem *                              DO_A5 -> O p1 S6
rem *                              DO_A6 -> O p1 S7
rem *                              DO_A7 -> O p1 S8
```

Only the start condition must be adapted. The placement process is only complete when the pallet is placed and the cabin is closed.

```
condition 3
rem (RPT: Before starting a part program repetition)

rem Drive to secure position
drv cnc x00??? y00??? z00???
wait cnc x00??? y00??? z00??? t00001

rem Check again CMM and Loading Device
brk p1 s111xxxxx

rem Request loading
set p1 sx10xxxxx

rem Wait loading finished; Pallet fit; Cabin closed
brk p1 sxx111xxx
end
```

#### 4.6.2.2 With Additional Good or Bad Part Signal

A single output signal is used to communicate that the part is good or bad:

- Good/Bad Part (High = Good)

This signal can be managed with the help of the conditions 21, 22 and 23.
The signal can either be set during the inspection or at the end of it. In the following example the signal is only set at the end of a part program (red highlighted lines are new):

```
rem * Input (at CMM, I/O card):
rem * Automation active            DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch      DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area     DI_A2 -> I p1 S3  1 = Out
rem * Un-/Loading finished         DI_A3 -> I p1 S4  1 = Finished
rem * Pallet fit and Cabin close   DI_A4 -> I p1 S5  1 = Fit and Closed
rem *                              DI_A5 -> I p1 S6
rem *                              DI_A6 -> I p1 S7
rem *                              DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active            DO_A0 -> O p1 S1  1 = Active
rem * Request loading              DO_A1 -> O p1 S2  1 = Request
rem * Request unloading            DO_A2 -> O p1 S3  1 = Request
rem * Not Recoverable Error        DO_A3 -> O p1 S4  1 = Error
rem * Good/Bad Part                DO_A4 -> O p1 S5  1 = Good
rem * Bad Part                     DO_A5 -> O p1 S6  1 = Bad
rem *                              DO_A6 -> O p1 S7
rem *                              DO_A7 -> O p1 S8
rem
rem * Meaning of used memory BITs:
rem * BIT 11 = in tolerance
rem * BIT 12 = out of tolerance
```

At any start of a part program repetition (condition 3) the signal and memory bits must be reset.

```
condition 3
rem (RPT: Before starting a part program repetition)

rem Reset OK/notOK Signals and memory bits
set p1 sxxxx00xx
bit 11 = OFF
bit 12 = OFF

rem Drive to secure position
drv cnc x00??? y00??? z00???
wait cnc x00??? y00??? z00??? t00001

rem Check again CMM and Loading Device
brk p1 s111xxxxx

rem Request loading
set p1 sx10xxxxx

rem Wait loading finished Pallet fit and cabin is closed
brk p1 sxx111xxx
end
```

The conditions that calls automatically during tolerance calculation must set the bits.

```
condition 21
rem (RPT: Feature within control limits)
if not bit 12
  bit 11 = ON
end
```

```
condition 22
rem (RPT: Feature within tolerance, but out of control limits)
same 21
end

condition 23
rem (RPT: Feature out of tolerance)
bit 11 = OFF
bit 12 = ON
end
```

At the end of the part program the “Good or Bad Signal” signal must be set according to the status of the bits:

```
condition 10
rem (RPT: After exiting the last part program repetition)

rem Drive to secure position
drv cnc x00??? y00??? z00???
wait cnc x00??? y00??? z00??? t00001

rem Check again CMM and Loading Device
brk p1 s111xxxxx

rem Set OK signal only if OK
if BIT 11
  set p2 sxxxx10xx
if BIT 12
  set p2 sxxxx01xx

rem Request unloading
set p1 sx01xxxxx

rem Wait unloading finished
brk p1 sxx11xxxx
end
```

#### 4.6.2.3 With an Additional Manual-Automatic Switch

We need one additional input signal:

- Manual or Automatic Mode (High = Automatic)

For safety reasons all the previous signals must still be managed.

```
rem * Input (at CMM, I/O card):
rem * Automation active            DI_A0 -> I p1 S1  1 = Active
rem * CMM in home from switch      DI_A1 -> I p1 S2  1 = Home
rem * L.D. not in working area     DI_A2 -> I p1 S3  1 = Out
rem * Un-/Loading finished         DI_A3 -> I p1 S4  1 = Finished
rem * Pallet fit and Cabin close   DI_A4 -> I p1 S5  1 = Fit and Closed
rem * Manual/Automatic Mode        DI_A5 -> I p1 S6  1 = Automatic
rem *                              DI_A6 -> I p1 S7
rem *                              DI_A7 -> I p1 S8
rem
rem * Output (at CMM, I/O card):
rem * Automation active            DO_A0 -> O p1 S1  1 = Active
rem * Request loading              DO_A1 -> O p1 S2  1 = Request
rem * Request unloading            DO_A2 -> O p1 S3  1 = Request
rem * Not Recoverable Error        DO_A3 -> O p1 S4  1 = Error
rem * Good/Bad Part                DO_A4 -> O p1 S5  1 = Good
rem *                              DO_A5 -> O p1 S6
rem *                              DO_A6 -> O p1 S7
rem *                              DO_A7 -> O p1 S8
```

The start and end conditions must be adapted to not request placement or removal of the part.

```
condition 3
rem (RPT: Before starting a part program repetition)

rem Reset OK/notOK Signals and memory bits
set p1 sxxxx0xxx
bit 11 = OFF
bit 12 = OFF

rem Drive to secure position
drv cnc x00??? y00??? z00???
wait cnc x00??? y00??? z00??? t00001

rem Check again CMM and Loading Device
brk p1 s111xxxxx

rem Manual Mode?
if I p1 sxxxxx0xx
  end

rem Request loading
set p1 sx10xxxxx

rem Wait loading finished Pallet fit and cabin is closed
brk p1 sxx111xxx
end


condition 10
rem (RPT: After exiting the last part program repetition)

rem Drive to secure position
drv cnc x00??? y00??? z00???
wait cnc x00??? y00??? z00??? t00001

rem Check again CMM and Loading Device
brk p1 s111xxxxx
```

```
rem Set OK signal only if OK
if BIT 12
  lbl endcond
if BIT 11
  set p2 sxxxx1xxx
lbl endcond

rem Manual Mode?
if I p1 sxxxxx0xx
  end

rem Request unloading
set p1 sx01xxxxx

rem Wait unloading finished
brk p1 sxx11xxxx
end
```

# 5 Other application areas

The IO-Communication functionality can be used in other applications areas, too.

## 5.1 Using Patrol Lights as Process Indicator at the end of a measurement

The most used workflow is easy. You need only three output signals:
Output Signals:

- Green Light (High = On), to sign a good part at the end of the measurement
- Yellow Light (High = On), to sign that part program is running
- Red Light (High = On), to sign a bad part at the end of the measurement

```
rem * Output (at CMM, I/O card):
rem * Green Light                DO_A1 -> O p1 S1  1 = On
rem * Yellow Light               DO_A2 -> O p1 S2  1 = On
rem * Red Light                  DO_A3 -> O p1 S3  1 = On
rem *                            DO_A3 -> O p1 S4
rem *                            DO_A4 -> O p1 S5
rem *                            DO_A5 -> O p1 S6
rem *                            DO_A6 -> O p1 S7
rem *                            DO_A7 -> O p1 S8
```

When GEOPAK starts and ends, all signals should be set to zero.

```
condition 1
rem (RPT: Start of program)
set p1 s00000000
end

condition 20
rem (RPT: End of program)
set p1 s00000000
end
```

At the start of a measurement the good / bad signal must be set to “zero” and the Yellow Light must be switch on. A memory bit for the good bad decision must be reset, too.

```
condition 3
rem (RPT: Before starting a part program repetition)
bit 1 = off
set p1 s010xxxxx
end
```

During measurement in the tolerance calculation the IO-Conditions 21-23 are called. We need only the 23.

```
condition 23
rem Feature out of tolerance
BIT 1 = ON
End
```

At the end of a measurement the good / bad signal must now be set depending on the memory bit.

```
condition 3
rem (RPT: Before starting a part program repetition)
bit 1 = off
set p1 s010xxxxx
end
```

Also in this case we must think about the error handling. For this we can add a blinking red light when something happens during measurement.

```
condition 8
rem Error in part-program
set p1 s000xxxxx
blk p1 sxx1xxxxx t1000
end
```

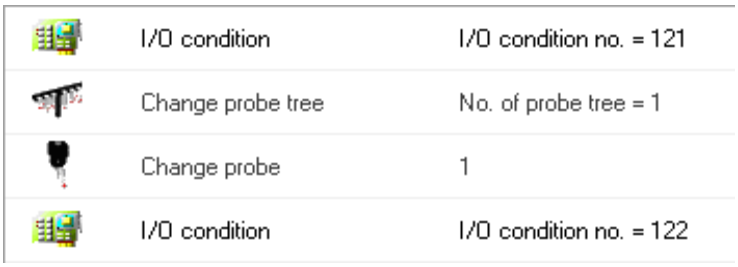
# 6 Using IO within GEOPAK

## 6.1 Part Programmable IO Commands

A part program may have to manage additional hardware. E.g.:

- Variable fixtures (e.g. a pallet which can move its part).
- The inspection program might require that the rack be brought in and the tool be changed.
- A part might require different inspection techniques (e.g. revolver trigger tension).

To use part programmable IO Commands it is necessary to create IO-Conditions that are not reserved.
The following example shows the process of changing the probe. Before the change, the rack must be brought into in the CMM working volume and afterwards it must be retracted again.




We need two input and two output signals:

```
rem * Input (at CMM, I/O card):
rem * Rack placed out                  -> I p2 S1  1 = out
rem * Rack placed in                   -> I p2 S2  1 = in
rem
rem * Output (at CMM, I/O card):
rem * Move Rack out                    -> O p2 S1  1 = Move out
rem * CMM is in idle                   -> O p2 S2  1 = Move in
```

In condition 121 the rack is brought in and in condition 122 it is retracted again. In both conditions, the position of the rack must be awaited.

```
condition 121
rem Move in
set p2 s01xxxxxx
brk p2 s01xxxxxx
end

condition 122
rem Move out
set p2 s10xxxxxx
brk p2 s10xxxxxx
end
```

## 6.2 Using IO Memory Bits in the "Formula Calculation" of GEOPAK

Memory bits may be set in the IO-communication-file, which can then be subsequently queried by the part program. A beneficial scenario would be if there are two possible locations for part placement on a CMM. In this case, the process control PC can set the appropriate memory bits at the start of the inspection process.

In the following example two positions are possible:

```
rem * Input (at CMM, I/O card):
rem * Position Left                    -> I p2 S1  1 = Left
rem * Position Right                   -> I p2 S2  1 = Right
rem
```

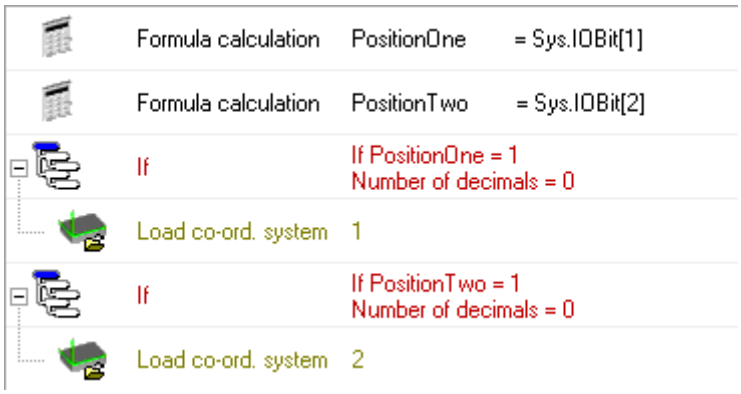
The signals are checked at the start of the part program and the bits are then set:

```
condition 3
rem (RPT: Before starting a part program repetition)
:
rem Reset Position Memory
BIT 1 = OFF
BIT 2 = OFF

rem Check Position Left
if I p2 s10xxxxx
  BIT 1 = ON

rem Check Position Right
if I p2 s01xxxxx
  BIT 2 = ON
:
end
```

The bits can now be queried in the part program and a part can be placed into an available position:



### 6.3 RemoteManager

The RemoteManager needs to start part programs via the Process Control PC. The handshake for this works via ASCII-Files in a shared communication directory.

To start a part program an ASCII-File request could look like the following (“REMOTE.ASK”):

```
EXECUTE_PATH_PART_PROGRAM
D:\PartPrograms\
MyPartName
MyPartProgramName
0
1
1
1
1

STAT
```

On successful start the related answer looks like following (“REMOTE.ANS”):

```
OK
```

When the part program run is finished, the RemoteManager writes a message file (“REMOTE.MSG”):

```
PPEND
```

*For deeper information please read the documentation for the RemoteManager “remotemgr_e.pdf”.*

***Note: MCOSMOS v4.3 will deliver a new RemoteManager with FieldBus support.***

# 7 Digital IO-Cards

## 7.1 Digital IO-Cards and Their Function

- IO-Cards are generally input output cards. In our case, we call them digital input-output cards. That means, per signal there exists only two states, logical "high"(generally high voltage) and logical "low" (generally low voltage).
- The different IO-Cards are suitable to different fields of application:
  - Different circuit-breaking capacities (PLC-24V-150mA, signals-12V-1A)
  - Plug-In Systems (ISA, PCI)
  - Operating systems (Windows XP, Windows 7 …).
- Our IO-Cards have physically separated inputs and outputs. The inputs and outputs are divided into "Ports" and these are further divided into channels (also called switches). Normally, a port has 8 switches, which can represent 0 or 1 ("low" or "high").
- Different IO-Cards may have a various numbers of input and output ports. The "ME-8100-A" card for example has 16 input and 16 output channels, i.e. 2 ports with each 8 switches for the input and 2 ports with each 8 switches for the output.
- Industry usually works with a standard voltage range of 0 to 24V. IO-Cards in this voltage range are usually opto-coupler or relay cards. These cards have galvanically isolated input and output sides; in this case, an internal circuit is actuating an external circuit. That means that the cards need an external power supply, which may then be actuated.
  Opto-coupler cards are used for low power signals. Relay cards are only used for higher power loads (for example lights).

## 7.2 FieldBus-Cards and Their Function

- With a FieldBus card (e.g. with ProfiNet protocol) it is possible to bypass the used ports over an Ethernet connection. So the IO switches are not directly wired. The cards using own internal memory which represented by accessible input and output bytes.
- By default implementation we use one byte per port. This means one byte contains 8 bits which represent the above mention 8 switches. Optional the card can be configure in the way that one port use 8 bytes from the card memory.

## 7.3 List of Supported Cards

### 7.3.1 Digital IO Cards:

a) Meilhaus ME-5810-A/B (Recommended by Mitutoyo up from Win7)
- *PCIe card for Win7 or newer and is supported by the Driver-System "ME-iDS".*
- *16 Opto channel input (up to 36V).*
- *16 Opto channel output (up to 36V, 150mA source driver output).*
- *Compatible with the ME-8100 card.*
- *The B card is supported from v4.1 and upwards giving 32 channels.*

b) Meilhaus ME-8100-A/B (Recommended by Mitutoyo for XP)
- *PCI card for WinNT and Win9x. Win7 if it is supported by the Driver-System "ME-iDS".*
- *16 Opto channel input (up to 36V).*
- *16 Opto channel output (up to 36V, 150mA source driver output).*
- *Features closely compatible with the old EPT16/16 card.*

c) Meilhaus ME-8200-A/B
- *PCI card for Win7 and is supported by the Driver-System "ME-iDS".*
- *16 Opto channel input (up to 36V).*
- *16 Opto channel output (up to 36V, 700mA source driver output).*

d) Meilhaus ME-630
- *PCI card for WinNT and Win7. All types are supported by the Driver-System "ME-iDS".*
- *8 TTL Opto channel input (TTL-level).*
- *8 Opto channel input (up to 36V).*
- *16 Relay channel output (up to 30V, 2A Source driver output).*

e) National Instruments PC-OPDIO-16
- *ISA card for WinNT and Win9x.*
- *8 channel input (up to 36V).*
- *8 channel output (up to 36V, 250µA source driver output).*

The company KOMEG sells test boxes for the cards Meilhaus ME-8100-A, ME-5810-A and ME630 (www.komeg.de).

### 7.3.2 FieldBus Card:

a) Hilscher CIFX 50
- *Supports different FieldBus technologies:*


  *But only ProfiNet and EtherCAT are tested at the moment*
- *ProfiNet card for Win7 or newer and is supported by the Driver-System "cifx".*
- *Free configurable number of input and output ports, but max. 4 supported.*
- *Configure like described below.*

"Hilscher" address: "www.hilscher.com"
"Meilhaus" address: sales@meilhaus.de or www.meilhaus.de.

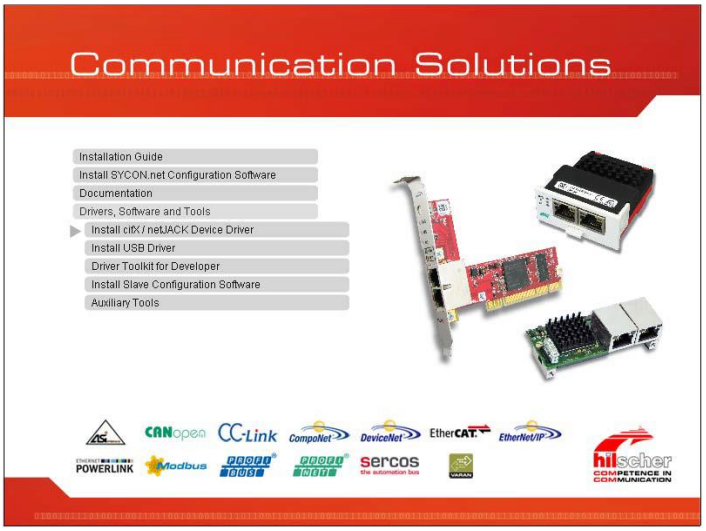
## 7.4 Installation of Hardware (example)

### 7.4.1 Meilhaus ME-8100 or ME-5810 for Windows 7-10.

- Install the ME-IDS driver. Follow installation description from Meilhaus. Do not forget to connect the internal power supply for this card.
  *One knowing problem with this card and Energy settings in Windows 10 (Fast Start-up)*
  - *Take care using Windows10, because the energy settings **(Fast Start-up) will prevent the Meilhaus card from outputting 24V!***
    *What is Windows Fast Start-up*
    https://www.cnet.com/how-to/what-is-windows-10s-fast-startup/

### 7.4.2 Hilscher FieldBus card CIFX 50.

- Start Hilscher Communication Solution CD. Execute “Install cifX Device Driver”:

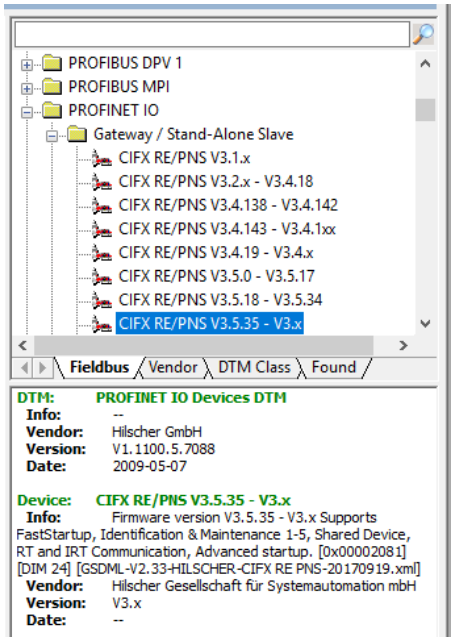



- Current Version of SYCON.NET must be installed (Version on CD/Hilscher Communication Solutions is not the current version):
  - https://kb.hilscher.com/display/SYCON (as of 2019-02-07)

- Get current firmware for cards (netX 500 series, ProfiNet IO-DEVICE) (Current version: V3.13.0.3)
  - https://kb.hilscher.com/display/PNS3V5/PROFINET+IO-Device+V3.13.0.3 (as of 2019-02-07)

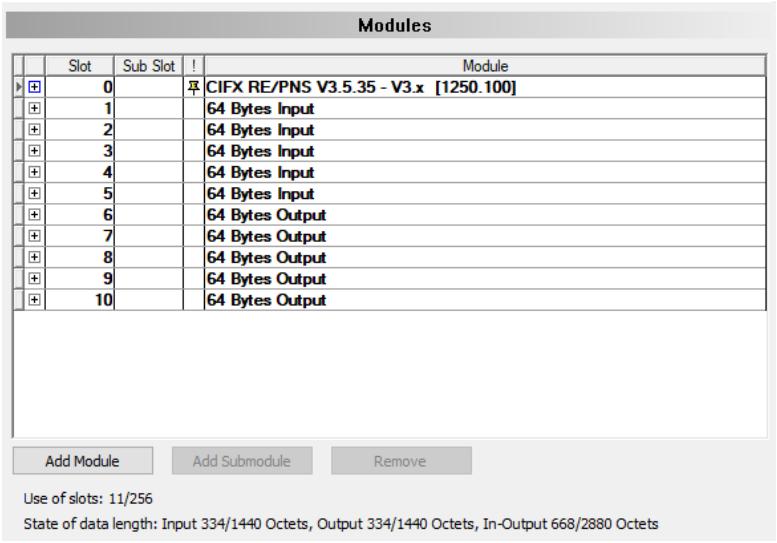
- Configuration of cards:
  - The cards used are of the CIFX 50E-RE type.

- To configure a card, first drag a template from the browser on the left, which corresponds to the firmware that was downloaded in the last step. Currently this is the *CIFX_RE_PNS_V3.5.35_-_V3.x*. Template. It can be found under *PROFINET IO > Gateway / Stand-Alone Slave:*

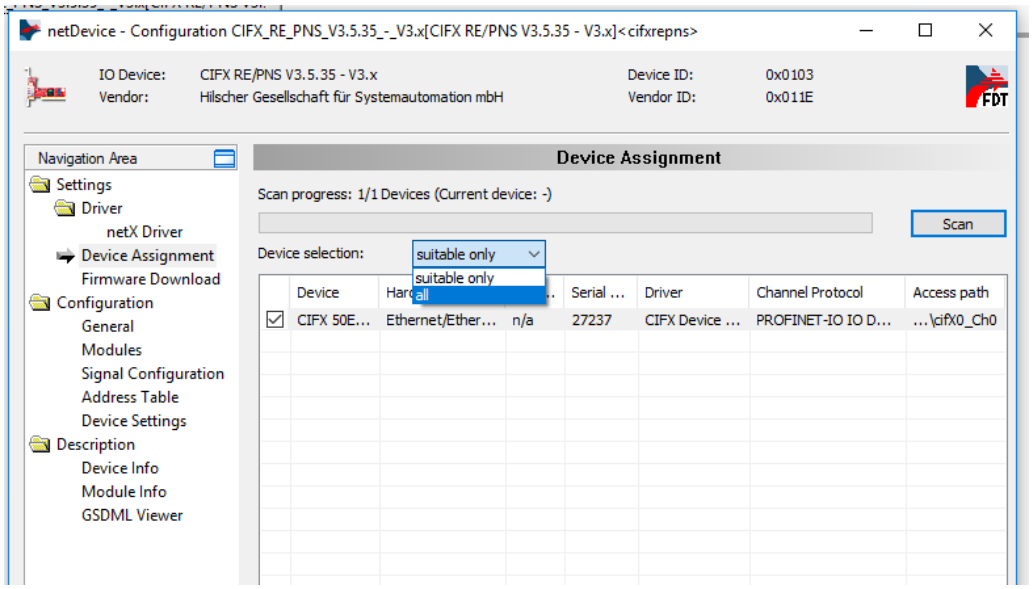



*Drag this template to the main view.*

- Configure the input and output bytes of the card (2-4 bytes needed) (Default is 4):
  Right Mouse click on card icon and select configuration
  Select Modules and set min. 4 input and 4 output bytes, but can be other like following:

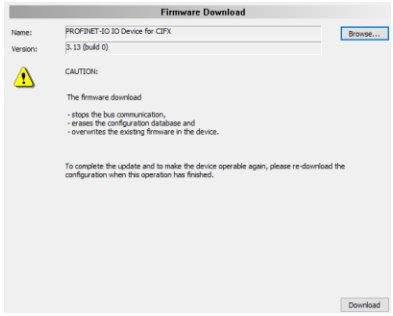



- Once the template has been loaded, open the configuration window from the context menu on the template. Choose the correct device on Device Assignment. If necessary, choose “all” in Device selection.




Apply the selection.

- Next, update the firmware under the “Firmware Download” Tab.




Note: SYCON.NET uses the term Download counterintuitively. Download of Firmware or configuration here means transfer to a card, usually called Upload.

- Figure out the “GSDML” file which is needed for the PLC (and which depends on the used FieldBus protocol):
    - Open configuration of the slave
    - Select GSDML Viewer
    - Locate the used file

# 8 IO-Conditions

## 8.1 Basics to Communicate via a IO-Card

- IO-Cards can be directly controlled and accessed with the use of certain commands. Because there are a variety of IO-Cards, and each can be used for a vast range of tasks, we have established some consistent commands, which can be programmed with a normal ASCII-Editor.
- The so called IO-Communication-File contains the actual communication to our MCOSMOS. At certain locations, so-called IO-Conditions are called and their commands are executed.
- Each IO-Condition has a specified number and is entered into the IO-Communication-File. An IO-Condition begins with the "Condition" statement, including a number and is terminating with "End".
- The IO-Condition commands for reading and writing are distinctly separated in the IO syntax.
- The actual commands of the IO-Conditions for the IO-Cards are:
  - Read instructions: "**Get**", "**Brk**" and "**If I** ..."
  - Write instructions: "**Set**" and "**Blk**"
- All other commands are specified for the IO-Condition flow control or for other output devices (screen, PC-speaker, measurement machine).

⚠ When writing an IO-Condition, never forget the error detection.
⚠ When writing an IO-Condition, you should use a "file header".

## 8.2 Installation of Software (MCOSMOS)

There must be a file called "IO_COND.INI" in the "INI"-directory of MCOSMOS. The default file you can find on the MCOSMOS installation CD (\OPTIONS\IO_COND). In this file you must set the name of the IO-Condition file (default "IO_COND.DAT") and the IO-Card type you want to use.

```
[IO-COND]
Iocondfile=io_customername.dat

;-------------------------------------------------------------------------
; Iocardtype                          Name of an supported Digital IO-Card -
; Iocardtype=NI-PC-OPDIO-16         -> National Instruments PC-OPDIO-16    -
; Iocardtype=ME-ME-8100             -> Meilhaus ME-8100A                   -
; Iocardtype=ME-IDS                 -> Meilhaus Cards with IDS driver      -
; Iocardtype=HI-CIFX                -> Hilscher FieldBus CIFX 50           -
;-------------------------------------------------------------------------
[IO-CARD]
Iocardtype= ME-IDS
```

⚠ It is possible to set no Card Type. Leave the variable empty. Of course, no direct IO function possible with.

⚠ It is possible to set the Card Type as “**ME-IDS**” from v4.1 onwards.

⚠ It is possible to set the Card Type as “**HI-CIFX**” from v4.3 onwards (for Hilscher FieldBus). Other Settings are needed for this type of card if they different from default:

```
;-------------------------------------------------------------------------
; Only for Hilscher FieldBus cifX Driver                                 -
; ! Settings are depend on FielBdBus Card Configuration (e.g. "cifX0")   -
; BoardName=cifX0       -> Name of Card in Configuration                 -
; Channel=0             -> Channel Number; not Used at the moment        -
; NoOfOutPort=4         -> Number of Input Bytes in Configuration        -
; NoOfInPort=4          -> Number of Output Bytes in Configuration       -
; OffOutPort=0          -> Offset to first Input Byte in Configuration   -
; OffInPort=0           -> Offset to first Output Byte in Configuration  -
;-------------------------------------------------------------------------
[HI-CIFX]
BoardName=cifX0
NoOfOutPort=4
NoOfInPort=4
OffOutPort=0
OffInPort=0
```

Now an **IO condition file** must be written. This file must be in the "INI"-Directory, too. Without these files MCOSMOS does not perform any IO-Communication.

For test reasons, you still can write a LOG-file. For this, you must enter in the “IO_COND.INI" file the following:

```
[IO-COND]
Iocondtrap=1
```

Then, an “IO_COND.LOG” file will be written in the MCOSMOS-TEMP directory, which can be as follows:

```
IO-CONDITION LOG-FILE
-------------------------------------------------------------
Time         Inport    Outport   Stat Cmd.
11:37:38.755 S11110000 S11001100 Fin  Cond 3
11:37:38.815 S11110000 S11001100 Fin  Wrt
11:37:38.835 S11110000 S11001100 Fin  Brk  P1 S00001111
11:37:40.046 S11110000 S11110000 Fin  Set  P1 S11110000
11:37:48.418 S11110000 S11001100
```

⚠ This file should only be used for test reasons because it can rapidly become over-sized. When starting GEOPAK, it will be reinitialised.

## 8.3 Syntax of the IO-Condition File

### General Remarks:

- Upper or lower case letters can be used.
- The "words" have to be separated by at least one blank.
- For numbers with a fixed number of digits, leading zeros are necessary.
- The usable "port number(s)" always depend on the card type.
- The "status word" defined by "'S" usually represents the input or output channels of a port. The characters following "S" (in this description denoted "ssssssss") can have these values:
  - S = 0 means "LOW"
  - S = 1 means "HIGH"
  - S = X means disregard, i.e. "do not change" (output) or "do not care" (input).

### Description of Syntax:

For the sake of clarity, the commands (keywords) and fixed identifiers (e.g. X) are given as upper case letters in the following description, the modifying parameters as lower case.

| | |
|---|---|
| **CONDITION nn**<br>**END** | Begin of IO-Condition #nn with 1 <= nn <= 100. From here, the following commands are executed until the END statement is encountered. The number of statements is not limited. The correspondence between the condition-numbers and the action of the program is listed in topic 9 of this report; this is fixed and determined by the programs. |
| **SAME mm** | This means that from here, the condition is the same as mm.<br>E.g.: CONDITION 20<br>SAME 10<br>END means that condition 20 is the same as 10. |
| **REM** | Is just a remark; ignored. |
| **SET Pn Ssssssss** | Sets the IO-signals.<br>Pn means "port number n";<br>Ssssssss specifies how to set the signals.<br>E.g.: SET P2 S1X000001<br>sets switch 1 and 8 to "high", does not change switch 2, and sets the others to "low". |
| **BIT b = ON** | This sets a single "Boolean" variable which can be tested in "IF"-statements. Range of 'b': 0 <=b <=99. |
| **BIT b = OFF** | Resets “Boolean” variable "b" again. Example: BIT 2 = OFF<br>To use the BIT – variable in GEOPAK formula-calculation use the syntax: “Sys.IOBit[b]” |
| **BRK Pn Ssssssss** | Halts the program until the switches of the input port n |

| | |
|---|---|
| | have the pattern defined by Ssssssss.<br>E.g.: BRK P2 S0XXXXXX1<br>Means: wait, until the first switch (input) of port 2 is "low", and the last input is "high", disregarding the others. |
| **WRT (text)** | Writes the "text" on a screen window. All characters within the brackets are written, including blanks. Only one screen window is possible. E.g.: WRT (Hello) writes the message "Hello" into a screen window. |
| **WRT (text1)\<br>WRT (text2)** | The command can be terminated by a "\"; in this case the program does not issue the text immediately, but waits for another.<br>WRT-command with another line of text. Then the two texts are displayed together as one two-line-text. |
| **CLS** | Closes the messages written by WRT. |
| **BEEP Ttttt Fffff** | Makes the computer beep for a duration of tttt milliseconds at the frequency ffff. Under Windows XP T and F are ignored; frequency and length are determined by the system. |
| **HORN Ttttt Fffff** | Causes successive beeps until the operator hits the "OK"-Button. Under Windows XP T and F are ignored; frequency and length are determined by the system; T is used for the break between the beeps. |
| **BLK Pn Sssssssss Ttttt** | The output signals of port n are switched (blinking). The time is defined by tttt in milliseconds; after this interval, the status of the channels is changed again. The bit pattern ssssssss defines the sequence, i.e. the initial setting. The ports are switched until the operator hits the 'OK'-Button. |
| **IF OR IF NOT** | This line contains a condition. If the condition is true, the next line of the file is executed; if not, the next line is ignored. Possible conditions are: |
| **IF I (OR O) Pn Sssssssss** | If the channels of "I"nput or "O"utput port n have the status defined by Ssssssss, this condition is true. |
| **IF ERR** | The error "ERR" is set, if a BRK or WAIT has been overridden by the operator using the 'Cancel'-Button (instead of the "proper" continuation due to the event waited for). |
| **IF BIT b** | Tests for a variable set by "BIT b = ON / OFF"; (cf. above). |
| **LBL llllll...** | Defines a label of up to 20 characters. This label can be used by a GOTO llllll... statement. This causes the program |

| | |
|---|---|
| | to continue at the "LBL"-line immediately ignoring the intermediate lines. It must be one word without blanks. |
| **WAIT KEY** | Waits until "OK" is pressed. |
| **WAIT Ttttt** | Waits tttt milliseconds. The time must always be given with 4 digits. |

**Syntax-Extension for GEOPAK**

The following commands are only valid for GEOPAK; they can be aborted by pressing "Cancel". In this case, the ERR is true, i.e. it can be checked by "IF ERR" (cf. IF-statements).

| | |
|---|---|
| **WAIT CNC Xxxxxx Yyyyyy Zzzzzz Ttttttt** | Waits until the position xxxxx yyyyy zzzzz in machine co-ordinates has been reached within a tolerance of ttttt mm. The co-ordinates have to be entered with 5 digits in mm. |
| **DRV CNC HOME** | Sends the "In home"-command to the CMM (without waiting!). |
| **DRV CNC Xxxxxx Yyyyyy Zzzzzz** | Makes the machine move to xxxxx yyyyy zzzzz without waiting. If the process must wait, the corresponding WAIT commands have to be used. |
| **CNC TOUCH ON** | Start of CMM probe signal. |
| **CNC TOUCH OFF** | End of CMM probe signal. |
| **PAT Lxxx Bxxxxx** | Activates the so-called “Pat-Lights” (Patrol car lights) and the so-called “Buzzer” of the controller<br>Lxxx<br>1. x -> 1. Light<br>2. x -> 2. Light<br>3. x -> 3. Light<br>Bxxxxx<br>1. x -> Buzzer off<br>2. x -> 1. External buzzer<br>3. x -> 2. External buzzer<br>4. x -> 3. External buzzer<br>5. x -> 4. External buzzer |

## 8.4 Standard IO-Condition-Calls

At which places which condition is called?

- ⚠ RPT means GEOPAK repeat mode. Repetitions always handle as multi-repetitions, if only one repetition the first repetition is also the last repetition.
- ⚠ STT means STATPAK (from v1.5.A5).

| Condition | Program | Description |
|---|---|---|
| 1 | RPT | Start of program. |
| 2 | Reserved | |
| 3 | RPT | Start of first repetition or start of single repetition. |
| 4 | RPT | Start of further repetition. |
| 5 | Reserved | |
| 6 | RPT | Called after “8” in case of active “On error goto” handling”. |
| 7 | RPT | Called after “8” in case of deactive “On error goto” handling”. |
| 8 | RPT | Error in part-program. |
| 9 | RPT | Reset System. |
| 10 | RPT | End of last repetition or end of single repetition. |
| 11 | RPT | End of repetition, if another repetition follows. |
| 12..15 | Reserved | |
| 16 | RPT | Called every second in Idle-Mode (No part program runs). |
| 17..19 | Reserved | |
| 20 | RPT | End of program. |
| 21 | RPT | Feature within control limits. |
| 22 | RPT | Feature within tolerance, but out of control limits. |
| 23 | RPT | Feature out of tolerance. |
| 24..30 | Reserved | |
| 31 | STT | Start of STATPAK. |
| 32 | STT | End of STATPAK. |
| 33 | STT | Start of task. |
| 34 | STT | End of task. |
| 35 | STT | Measurement value out of tolerance limits. |
| 36 | STT | Measurement value above upper tolerance. |
| 37 | STT | Measurement value below lower tolerance. |
| 38..39 | Reserved | |
| 40 | STT | At least one measurement value of subgroup out of tolerance limits. |
| 41 | STT | At least one measurement value of subgroup above upper tolerance. |
| 42 | STT | At least one measurement value of subgroup below lower tolerance. |
| 43 | STT | (upper chart) Location of subgroup within control limits. |
| 44 | STT | (upper chart) Location of subgroup out of control limits, but still within tol. |
| 45 | STT | (upper chart) Location of subgroup above upper control limit. |
| 46 | STT | (upper chart) Location of subgroup below lower control limit. |
| 47 | STT | (upper chart) Location of subgroup: Trend. |
| 48 | STT | (upper chart) Location of subgroup: Run. |
| 49 | STT | (upper chart) Location of subgroup: Midthird. |
| 50 | STT | (lower chart) Dispersion of subgroup within control limits. |
| 51 | STT | (lower chart) Dispersion of subgroup out of control limits. |
| 52 | STT | (lower chart) Dispersion of subgroup above upper control limit. |
| 53 | STT | (lower chart) Dispersion of subgroup below lower control limit. |
| 54 | STT | (lower chart) Dispersion of subgroup: Trend. |
| 55 | STT | (lower chart) Dispersion of subgroup: Run. |
| 56..59 | Reserved | |
| 60 | STT | Start of NetworkManager. |
| 61 | STT | End of NetworkManager. |
| 62 | STT | Error during translation. |

| Condition | Program | Description |
|---|---|---|
| 63 | STT | All data are OK, no translation error. |
| 64 | STT | At least one value out of tolerance. |
| 65 | STT | The location of at least one subgroup out of control limits. |
| 66 | STT | The dispersion of at least one subgroup out of control limits. |
| 67 | STT | Other violations of location [run, trend, middle third]. |
| 68 | STT | Other violations of dispersion [run, trend]. |
| 69 | Reserved | |
| 70 | Q-PAK | Q-PAK is started. No signal for door locking. (Not yet programmed). |
| 71 | Q-PAK | Lock door, with check if it is possible to move. |
| 72 | Q-PAK | Unlock door. |
| 73 | Q-PAK | Lock door, without check if it is possible to move. |
| 74 | Q-PAK | The queue is started. No signal for door locking.<br>(Not yet programmed) |
| 75 | Q-PAK | The queue is stopped. No signal for door locking.<br>(Not yet programmed) |
| 76 | Q-PAK | The user logs in. No signal for door locking.<br>(Not yet programmed) |
| 77 | Q-PAK | The user logs out. No signal for door locking.<br>(Not yet programmed) |
| 78 | Reserved | |
| 79 | Q-PAK | Q-PAK is finished. No signal for door locking.<br>(Not yet programmed) |
| 80 | Reserved | |
| 81 | Q-PAK | The queue is started and position 1 is occupied. |
| 82 | Q-PAK | The queue is started and position 2 is occupied. |
| 83 | Q-PAK | The queue is started and position 3 is occupied. |
| 84 | Q-PAK | The queue is started and position 4 is occupied. |
| 85 | Q-PAK | The queue is started and position 5 is occupied. |
| 86 | Q-PAK | The queue is started and position 6 is occupied. |
| 87 | Q-PAK | The queue is started and position 7 is occupied. |
| 88 | Q-PAK | The queue is started and position 8 is occupied. |
| 89 | Reserved | |
| 90..99 | free | Free for user defined IO-Conditions. |
| 100.149 | free | Free for user defined IO-Conditions |
| 150..200 | Reserved | |

### 8.4.1 Explanations for Q-PAK Runs

| Event | Called conditions |
|---|---|
| Q-PAK is called. The door is locked. | 70, 73 |
| Q-PAK is finished. The door is locked. | 73, 79 |
| The user activates the “Open Automated Door” button. This button is only active if the door is locked. | 72 |
| The queue starts after the user pressed the “Start Queue” button. Here, the user is also logged out. | if necessary 81, if necessary 82, if necessary 83, if necessary 84, if necessary. 85, if necessary. 86, if necessary 87, if necessary 88, 71, 74, 77 |
| The queue stops after the user has pressed the “Stop Queue” button (has asked for a break) or because all jobs have been processed. If the user is still logged in, the door is unlocked. | 75, if necessary 72 |
| The user activates the “Lock the Program” button (key symbol). If the door was unlocked, now it is locked. Here, also the user is logged out. | if necessary 73, 77 |
| After the user has effectively logged in. | 76 |

### 8.4.2 Explanations GEOPAK Runs

| Event | Called conditions |
|---|---|
| Start of GEOPAK repeat mode. | 1 |
| Before starting a part program repetition. | 3 |
| Before starting a following part program repetition. | 4 |
| Part program error or collision. | 8 |
| After exiting the last part program repetition. | 10 |
| After exiting a preceding repetition. | 11 |
| End of GEOPAK repeat mode. | 20 |

# 9 Training Course

## 9.1 IO-Communication without IO

### 9.1.1 Acoustic Signal in Case of a Collision

In case of a failure (collision) in a Pallet Execution, the customer would like to get issued a continuous acoustic signal that he must confirm before GEOPAK continues.

```
condition 8
horn t0500 f1000
end
```

### 9.1.2 Message at the Beginning and the End

The customer would like to get a message at the beginning and the end of a Pallet Execution to know if he can load, respectively unload the palette.

```
condition 3
wrt (please load palette and press
<enter>!)
wait key
end
```

```
condition 10
wrt (please reload palette and press
<enter>!)
wait key
end
```

### 9.1.3 Load and/or Reload the Part

The customer would like to get a message at the beginning and the end of a repetition to know if he can load and/or unload the part (multiple repetitions).

```
condition 3
wrt ( please load the part and press
<enter>! )
wait key
end

condition 4
same 3
end
```

```
condition 10
wrt ( please reload the part and press
<enter>! )
wait key
end

condition 11
same 10
end
```

### 9.1.4 Good or bad part

At the end of a repetition (no multiple repetition), the customer would like to get a message on the screen, whether the measured part was "good" or "bad".

```
condition 3
bit 1 = off
bit 2 = off
end

condition 10
if bit 1
wrt ( Part is good )
if bit 2
wrt ( Part is bad )
wait key
end
```

```
condition 21
bit 1 = on
end
condition 22
bit 1 = on
end

condition 23
bit 2 = on
end
```

## 9.2 Conditions with IO Functions

### 9.2.1 Optical Signal

In case of a failure (crash) during a part program repetition (no multiple repetition), the customer would like to be given an optical signal, which he must confirm before GEOPAK continues. In addition, at the end of repeat mode, the customer wants to be given an optical signal to confirm whether the part was "good" or "bad".

```
condition 1
set p1 s00000000
end

condition 3
bit 1 = off
bit 2 = off
set p1 s00000000
end

condition 8
wrt ( collision )
blk p1 s00110000 t1000
end

condition 10
if bit 1
set p1 s10000000
if bit 2
set p1 s01000000
end
```

```
condition 20
set p1 s00000000
end

condition 21
bit 1 = on
end

condition 22
bit 1 = on
end

condition 23
bit 2 = on
end
```

### 9.2.2 New Pallet Loading Device

The customer has bought a palette Loading Device. Now, this must run in fully automatic continuous operation. The unit also has a proximity switch.

Signal tracing:

Output - port 1 - switch 1: "high" – GEOPAK activated (program is running)
Output - port 1 - switch 2: "high" - GEOPAK in idle mode
Output - port 1 - switch 3: "high" - GEOPAK in measure mode
Output - port 1 - switch 4: "high" - failure (crash) occurred
Input - port 1 - switch 1: "high" - Loading Device is activated
Input - port 1 - switch 2: "high" - Loading Device is loading
Input - port 1 - switch 3: "high" - Loading Device is Unloading
Input - port 1 - switch 4: "high" – proximity switch
Input - port 1 - switch 5: "high" - failure occurred

⚠ IO-condition, not forget 1 and 20

```
condition 3
wrt ( wait for loading! )
brk p1 s11010000
cls
if not i p1 s11010000
goto error
set p1 s101x0000
end
lbl  error
set p1 s11010000
end

condition 8
set p1 sxxx1xxxx
end
```

```
condition 9
set p1 s10000000
end

condition 10
wrt ( wait for unloading! )
brk p1 s10100000
cls
if not i p1 s10100000
goto error
set p1 s110x0000
end
lbl  error
set p1 s11010000
end
```

### 9.2.3 Cabin Closed?

The customer would like to activate a repetition of the inspection program outside a cabin, which is around the CMM to have an air-conditioned room. At the same time, a colleague should load the part to be measured, close the doors, and press then a switch so that GEOPAK begins to measure. The screened room has self-locking switches, which check if the cabin is closed.

Signal tracing:

Output - port 1 - switch 1: "high" – GEOPAK activated (program is running)
Output - port 1 - switch 2: "high" - GEOPAK in idle mode
Output - port 1 - switch 3: "high" - GEOPAK in repeat mode
Output - port 1 - switch 4: "high" - failure (crash) occurred
Input - port 1 - switch 1: "high" – screened room activated
Input - port 1 - switch 2: "high" – cabin is closed

⚠ IO-condition, not forget 1 and 20

```
condition 3
wrt ( please load the part and close the
cabin! )
brk p1 s11000000
end

condition 4
same 3
end

condition 8
set p1 sxxx1xxxx
end
```

```
condition 9
set p1 s10000000
end

condition 10
wrt ( please unload the part! )
brk p1 s10000000
end

condition 11
same 10
end
```

### 9.2.4 Communication between CMM and Robot

The customer wants to incorporate a measurement machine in his production line that can be feed by a robot. At the end of each repetition the robot must know, whether the measured part was “Good” or “Bad”.

Signal tracing:

Output - port 1 - switch 1: "high" - GEOPAK activated (program is running)
Output - port 1 - switch 2: "high" - GEOPAK in idle mode
Output - port 1 - switch 3: "high" - GEOPAK in repeat mode
Output - port 1 - switch 4: "high" - failure (crash) occurred
Output - port 1 - switch 7: "high" - part is good
Output - port 1 - switch 8: "high" - part is bad
Input - port 1 - switch 1: "high" - Robot is activated
Input - port 1 - switch 2: "high" - Robot is loading
Input - port 1 - switch 3: "high" - Robot is unloading
Input - port 1 - switch 4: "high" - proximity switch
Input - port 1 - switch 5: "high" - failure occurred

⚠ IO-condition, not forget 1 and 20

```
condition 3
bit 1 = off
bit 2 = off
wrt ( wait for loading! )
brk p1 s11010000
cls
if not i p1 s11010000
goto error
set p1 s101x0000
end
lbl  error
set p1 s11010000
err 102
end

condition 4
same 3
end
condition 8
set p1 sxxx1xxxx
end
condition 9
set p1 s10000000

end
```

```
condition 10
if bit 1
set p1 sxxxxxx10
if bit 2
set p1 sxxxxxx01
wrt ( wait for unloading! )
brk p1 s10100000
cls
if not i p1 s10100000
goto error
set p1 s110x0000
end

lbl  error
set p1 s11010000
err 102
end

condition 11
same 10
end

condition 21
bit 1 = on
end

condition 22
bit 1 = on
end

condition 23
bit 2 = on
end
```

# 10 I/O Experience from the UK

*Written by Mike Kuscher*

The following are some suggestions/advice gained from implementing automated systems in the UK.

## 10.1 Home Switches:

For 'bridge machines' it is usually sufficient to fit a single home switch to the Y axis of the CMM. If the Y axis can be verified as 'home' then everything should be clear of the loading area.

For machines such as the Mach 3A, it is necessary to have a 'home signal' for all 3 axes. Fortunately, the machine is available with 'home sensors' already fitted to all 3 axes, but a request needs to be made for the additional relay block that allows these to be 'summed' into a single signal.

For the Ko-Ga-Me machine, it is critical to have home signals for all 3 axes because of the very small volume available, unfortunately, no allowance appears to have been made for this in the design of the machine.

If rotary or index tables are to be used, it is necessary to also have a home signal for this axis as well.

Simply provide the connections as 2 'volt free' wires to the cell integrator. They can send their own signal down the one wire and, if the machine is 'home', they will get their own signal back (less worry for you), because there is no 'At Home' signal available from a CMMC.

## 10.2 24v and 0v supply for I/O cards:

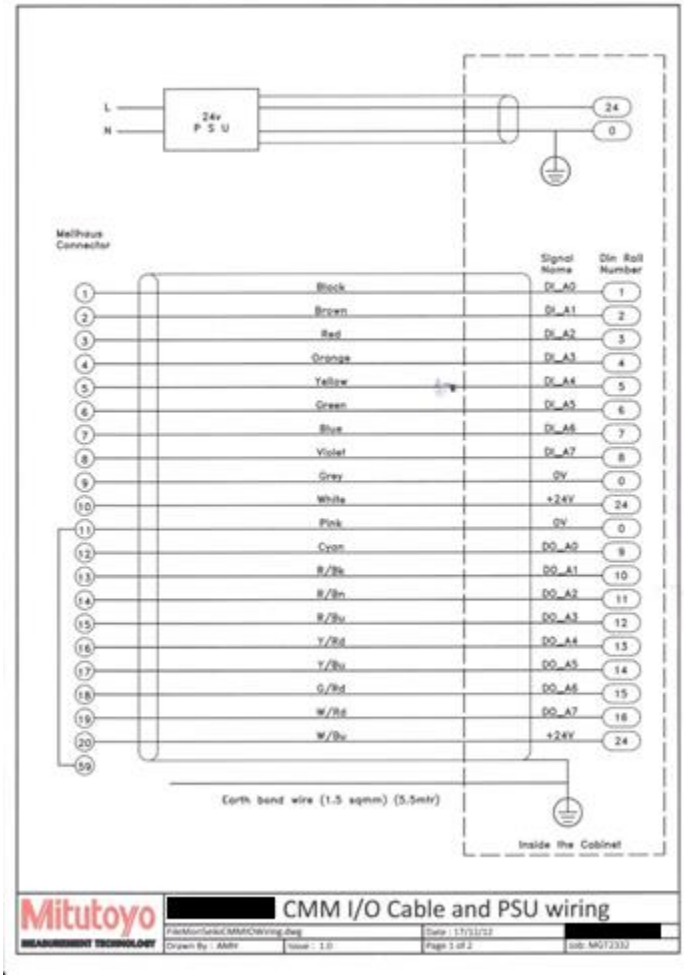
Wherever possible, always try and insist that the cell integrator supplies the 24v and 0v supply for the I/O. This eliminates Mitutoyo personnel from the health and safety implications of using transformers.

Where this is not possible, it is strongly advised to use a sealed power converter (power brick) which are readily available, low cost and already have their own approval.

### 10.3 I/O cables:

These are used to pass the signals between the I/O card in the CMM PC and the cell integrators control system (usually some form of PLC).

It is common that the cell integrator will use a system such as “ProfiBus” or “Siemens”. These are simply devices that convert the individual I/O signals to an Ethernet based signal, over the cell LAN (to reduce wiring) then convert them back to individual signals, at the other end.
The cell integrator would normally be expected to provide these units, as part of the installation.
It is advised that the one end of the cable, that you will supply, is simply terminated, separate, clearly identified cables, for the cell integrators to connect themselves. This eliminates the risks of having to be aware of the different ‘pin outs’ between different systems (Siemens is particularly illogical), but always supply a clear cable connection diagram, as in the following example.

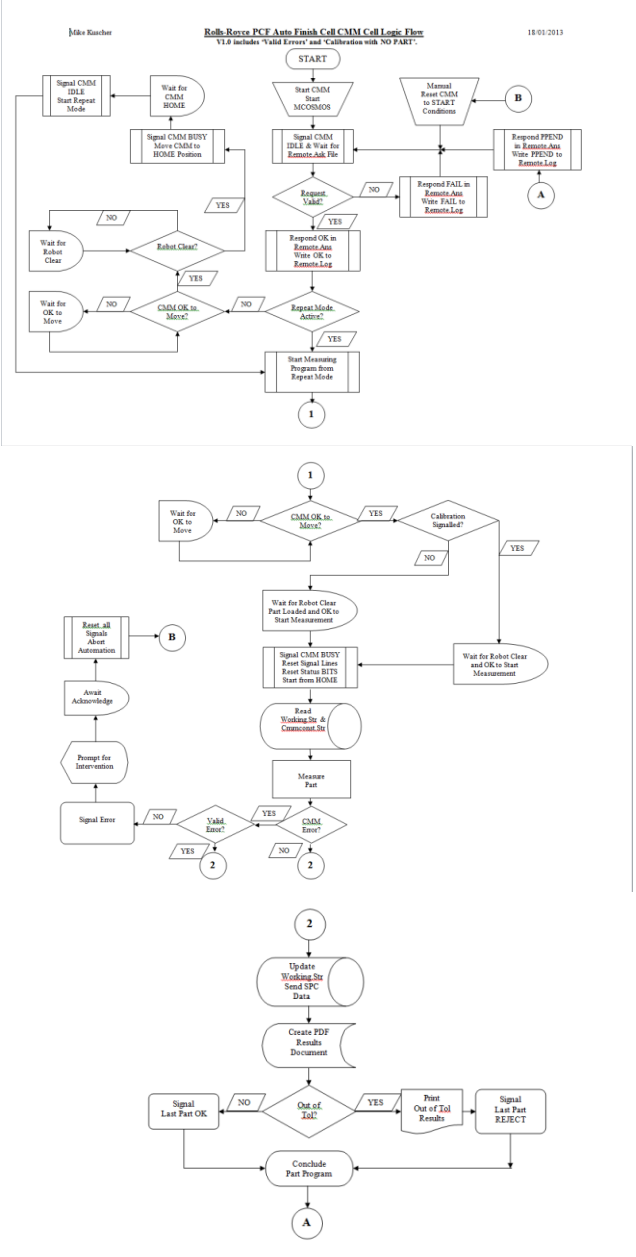


### 10.4 Planning the project:

When the project is about to start, always produce a document of how you understand the system will work and circulate this to all of the project partners (customer, cell integrator, machine tool supplier, robot supplier etc.) for agreement. It is surprising how often this simple action shows up potential issues that no one else has thought of.

XOverview of [redacted] Auto Finish Cell Operation.doc

The next stage is to produce the whole process as a flow chart. Again, this should be circulated as it often identifies further potential issues that would become problems much later (example follows).



## 10.5 I/O Signals:

You usually have 16 input and 16 output lines available – USE THEM!
If there are spare lines, wire them in and label them as “spare”, there is often one or more extra signals needed when testing starts.

Never rely on just one signal being High or Low, always use 2 signals in a ‘toggle’ system, for example…
Use 1 signal for “CMM BUSY” and another signal for “CMM IDLE”. The reason is simple.

- CMM BUSY could be High because the CMM IS busy, or there could be a short circuit giving a false signal.
- CMM IDLE could be Low because the CMM IS Idle, or a wire could be disconnected.

So, if the machine IS Busy, send CMM BUSY High AND send CMM IDLE Low, if the CMM is not Busy, send CMM BUSY Low and CMM IDLE High then look at both signals. This is safer, because it is very unlikely that you have a short circuit on one signal line and a disconnected wire on the other signal line, both at the same time.

The same rule applies to inputs, for example ROBOT IN AREA and ROBOT CLEAR.

If you are going to be generating ‘corrective feedback’, you will need to know if the machine tool is in ‘Setup Mode’ or ‘Production Mode’ to calculate the correct amount of feedback.

## 10.6 I/O Truth Table:

Always draw up and supply a copy of the I/O, detailing what signals you will send, what signal you need to receive, what I/O Bits you will use and what ‘user specified’ I/O conditions you have created. This is invaluable for everyone involved, including whoever is going to write the part programs. Example below…

Mike Kuscher

**RR Filton GCI I/O LINES v1.3**

11/11/2013

**CMM Output Lines:** Status/Safety signals from each CMM to Cell control

| Channel 1 - Line No: | DO_A0 | DO_A1 | DO_A2 | DO_A3 | DO_A4 | DO_A5 | DO_A6 | DO_A7 |
|---|---|---|---|---|---|---|---|---|
| Usage: | CMM IDLE | CMM BUSY | CMM ERROR | Last Part OK | Last Part REJECT | Last Part REWORK | Corrections Available | Operator Call |
| Channel 2 - Line No: | DO_A8 | DO_A9 | DO_A10 | DO_A11 | DO_A12 | DO_A13 | DO_A14 | DO_A15 |
| Usage: | CMM in Automation | CMM in Manual | Spare | Spare | Spare | Spare | Spare | Spare |

CMM ‘HOME’ & Index Table ‘HOME’ signals provided by ‘volt free’ contacts, direct to ‘Cell Control’ PLC.

**CMM Input Lines:** Status/Safety signals to each CMM from Cell control

| Channel 1 - Line No: | DI_A0 | DI_A1 | DI_A2 | DI_A3 | DI_A4 | DI_A5 | DI_A6 | DI_A7 |
|---|---|---|---|---|---|---|---|---|
| Usage: | Robot IN DANGER AREA | Robot CLEAR | Confirm Part Present | Confirm CMM START | CMM DO NOT RUN | Run with NO PART | Vending Magerle in Setup Mode | Spare |
| Channel 2 - Line No: | DI_A8 | DI_A9 | DI_A10 | DI_A11 | DI_A12 | DI_A13 | DI_A14 | DI_A15 |
| Usage: | Spare | Spare | Spare | Spare | Spare | Spare | Spare | Spare |

**Internal Status Flags:** Internal status signalling between I/O and MCOSMOS

| BIT No: | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Usage: | Start Automation ONLY | Valid Error | IN Tolerance | OUT Tolerance | REWORK | CORRECTIONS | Vending Magerle in Setup Mode |

In all of the above tables, the condition is TRUE when the logic is HIGH (1) and FALSE when the logic is LOW (0)

**Part Program I/O Calls:** User declared I/O condition calls available to part programmer

| I/0 Condition | 90 | 91 | 92 | 93 | 94 | 99 |
|---|---|---|---|---|---|---|
| Usage: | Call before using ‘On error goto’ | Call after using ‘On error goto’ | Call before reapplying tolerances | Call to signal Rework needed | Call to signal corrections available | Operator CALL signal for cell control |

### 10.7 Why write special I/O conditions?

There are many reasons, here are just 2 examples…

1. If an error occurs, during a part program, this will activate I/O condition 8 for error handling. Often, during 'conditional programming' the programmer will use "On Error Goto". Trouble is, I/O will still see this as an error. Solution?
    a. Declare an I/O Bit for identification to the I/O if "On Error Goto" will be used.
    b. Create and I/O condition (e.g. 90) to set this Bit to 1
    c. Create another I/O condition (e.g. 91) to set this Bit to 0
    d. Test inside condition 8 to see if this Bit is 1 or 0
    e. Tell the programmer to call condition 90 just before using "On Error Goto"
    f. Tell the programmer to call condition 91 after using "On Error Goto"
    g. Now, in condition 8, if the Bit is 1 then there is no error, it is just "On Error Goto". If the Bit is 0, then it is really an error.
2. In an automated cell, there will be no one watching the CMM, what happens if 'operator input' is required? It can just 'sit there' waiting.
    a. Create an I/O condition (e.g. 99) to send an "operator call" I/O line high
    b. Tell the programmer to call condition 99 just before asking for operator input
    c. This signals at the HMI that the operator is needed at CMM#x
    d. When the operator attends he first acknowledges condition 99, which switches the I/O line Low again
    e. Now the operator attends to the input, process continues, no delays.

### 10.8 Test, Test And Test Again:

When a robot hits a CMM, the results are dramatic and expensive!
You just need a box, with 16 input LEDs, 16 output LEDs a 24v supply, I/O cable and 16 switches to simulate the inputs. They are easy to make, or you can purchase one.
Keep simulating your system, in virtual, doing all of the things they should do AND all of the things they should not do.
When the automation system is finally built, things WILL be wrong.
When this happens, they WILL say the fault is yours, because it is easier for them.
If you know, beforehand, that your system is correct AND you can prove it, it saves you a lot of time and wasted arguments/explanations.

### 10.9 Keep multiple copies:

Of all files and documents you produce, they will be needed for copies or similar cells, updates and recovery.

### 10.10 Finally:

Ensure you have a system in place to handle any arising queries…

# 11 History

| Report | What | For Version |
|---|---|---|
| V3.51 | Problem with ME 5810 and Windows10 Fast Start-up | - |
| | Better description of Hilscher FieldBus card installation | |
| V3.50 | Support Hilscher FieldBus CIFX 50 (e.g. for ProfiNet) | MCOSMOS v4.2.R3 |
| V3.10 | Example for a main workflow | - |
| | Other application areas | - |
| | I/O Experience from the UK | - |
| V3.01 | Correct Signal reset in condition 3 and 10 | - |
| V3.00 | Add Full Automation chapter and reorder/overwork old one | - |
| V2.61 | Support of Meilhaus ME-5810 B IO-Card | MCOSMOS v4.1.R1 |
| V2.60 | Support of Meilhaus ME-5810 IO-Card | MCOSMOS v3.4.R1 |
| V2.51 | Correct Buzzer explanation | - |
| V2.50 | Support of Meilhaus ME-8200 IO-Card | MCOSMOS v3.4.R1 |
| v2.40 | Condition call 16 | v3.1.E13 |
| v2.30 | Condition call 6 + 7 | v3.0.E1 |
| v2.21 | Correction | - |
| v2.20 | Pat-Lights PAT Lxxx Bxxxx | v2.4.E1 |
| v2.10 | Condition call 85 – 88 | v2.3.E1 |
| | New card ME-630 | v2.3.E1 |
| v1.51 | Command “CNC TOUCH ON / OFF” | v1.5.E8 |
| v1.50 | Explanations to runs | - |
| | New conditions for Q-Pak | - |
| | Max. conditions set on 200 | v1.5.E5 |
| v1.33 | Correction | - |
| v1.32 | PIN-port scheme of ME-8100-A-PCI | - |
| v1.31 | Description of use of IO-Bit in GEOPAK | - |
| v1.30 | Examples: IO-communication with IO | - |
| | Examples: IO- communication without IO | - |
| | Procedure with “Customer Project” | - |
| v1.20 | Call condition 31 – 70 | STATPAK v1.5.A5 |
| | Insert the whole paragraph | - |
| | Insert the whole paragraph | - |
| | Insert the whole paragraph | - |
| v1.10 | Support of Meilhaus-ME-8100-A IO-Card | MCOSMOS v1.4.R2 |
| v1.02 | Support of Meilhaus-ID-100 IO-Card | MCOSMOS v1.3.R1 |
| v1.01 | Call condition 1, 21, 21, 22, 23 | GEOPAK v1.3.B8 |