

# Counter inputs, measuring RPM/impulses (PseDet)

## Abstract

Any digital input of any control system with NOS operation system and digital inputs of the DM(M)-DI24 extension modules can be used for impulse counting. This application note describes the method for using the inputs as counters.

Author: Zbyněk Říha  
Document: ap0017\_en\_05.pdf

## Attachments

File contents: ap0017\_en\_03.zip

|                            |  |
|----------------------------|--|
| counterinputs_p1_en_03.dso | Example #1 – programme counter                                 |
| counterinputs_p2_en_03.dso | Example #2 – programme counter INT, rotation speed measurement |
| counterinputs_p3_en_03.dso | Example #3 – hardware counter                                  |
| counterinputs_p4_en_03.dso | Example #4 – counter of impulses on <b>DM-DI24</b>             |
| counterinputs_p5_en_03.dso | Example #5 – counter of impulses on <b>DMM-DI24</b>            |

## Contents

---

|  |           |
|--|-----------|
| Contents.....  | 2         |
| Revision history.....  | 3         |
| Related documentation .....  | 3         |
| <b>1 Definitions of terms.....</b>                                     | <b>4</b>  |
| <b>2 Counter inputs, measurement of rpm / impulses .....</b>           | <b>5</b>  |
| <b>3 Program counter .....</b>   | <b>6</b>  |
| 3.1 Programme operation.....   | 6         |
| 3.1.1 Impln module .....   | 6         |
| 3.1.2 DImp module.....   | 6         |
| 3.2 Example no. 1 – programme counter .....                            | 6         |
| Quick process, period of 50 ms .....                                   | 7         |
| Proper process, period of 5 ms .....                                   | 7         |
| <b>4 Program counter INT.....</b>                                      | <b>8</b>  |
| 4.1 Interrupt processes.....   | 8         |
| 4.2 Programme operation.....   | 8         |
| 4.2.1 RPM measuring .....  | 9         |
| 4.3 Example no. 2 – programme counter INT, measuring revolutions ..... | 9         |
| Process Interrupt_0.....   | 9         |
| Process Interrupt_1 .....  | 9         |
| Proper process, period of 1 ms .....                                   | 9         |
| <b>5 HW counter .....</b>  | <b>10</b> |
| 5.1 Input modes .....  | 10        |
| 5.2 Properties of counter inputs.....                                  | 11        |
| 5.3 Programme operation.....   | 12        |
| 5.4 Example #3 – hardware counter.....                                 | 12        |
| Process INIT .....   | 13        |
| Proper process, period of 10 ms.....                                   | 13        |
| <b>6 Counting impulses via extension modules .....</b>                 | <b>14</b> |
| 6.1 DM-DI24 extension module .....                                     | 14        |
| 6.1.1 Example #4 – counting impulses on DM-DI24 .....                  | 14        |
| Loading counter values of all inputs .....                             | 14        |
| Setting the value of all counters.....                                 | 15        |
| 6.2 DMM-DI24 extension module .....                                    | 15        |
| 6.2.1 Example #5 – counting impulses on DMM-DI24 .....                 | 15        |
| Reads the value of counters on all inputs. ....                        | 16        |
| Setting the counter values .....                                       | 16        |
| <b>7 Technical support .....</b>                                       | <b>17</b> |
| <b>8 Warning.....</b>  | <b>18</b> |

## Revision history

| Version | Date         | Changes by | Changes  |
|---------|--------------|------------|--|
| 001     | 28. 11. 2008 | Říha Z.    | New document   |
| 002     | 08. 09. 2011 | Říha Z.    | Correction of the max possible value of <b>DM-DI24</b> counter, correction of application. Application example created in DetStudio version 1.7.0.                       |
| 003     | 20. 12. 2012 | Říha Z.    | Added information about the problem with writing counter values in firmware version 1.65.  |
| 004     | 26. 05. 2015 | Říha Z.    | Chapters 4.1, 5, 5.3.1, and 5.3.5 modified.  |
| 005     | 14. 08. 2019 | Říha Z.    | Change of the AP name, update of lists of HW according to current offerings, update of HW in the examples according to current offerings, changes in document structure. |
|         |              |            |  |

## Related documentation

1. Help tab in the PseDet section of the DetStudio development environment  
file: Psedet\_en.chm
2. Operation manuals for AMiT control systems  
file: xxx\_g\_en\_xxx.pdf
3. Application note AP0008 – Communication in MODBUS RTU network (PseDet)  
file: ap0008\_en\_xx.pdf
4. Application note AP0016 – Principles of RS485 interface usage  
file: ap0016\_en\_xx.pdf
5. Application note AP0025 – Communication in ARION network – table definition  
file: ap0025\_en\_xx.pdf

# 1 Definitions of terms

---

## **DetStudio**

Development environment made by AMiT; it is used for control systems parametrisation. The development environment is available for download at [amitautomation.com](http://amitautomation.com).

## **Counter inputs**

Are digital inputs that are fitted with quick counters. It is possible to use them to count the inbound impulses via the IRCxxx modules.

## **Channel**

A group of up to sixteen signals (input/output) of the same type (digital/analogue).

## **RS485**

It is a half-duplex serial bus enabling communication of multiple units in a single signal pair. More information is available in the Application note *AP0016 – Principles of RS485 interface usage*.

## **ARION**

It is a serial, half-duplex communication protocol for communication between AMiT control system and extension I/O modules. The total number of inputs/output of the control system can be increased by the I/O modules.

## **MODBUS RTU**

Is an open protocol for intercommunication of various devices allowing data transmission over various networks and buses. It is a client-server (master and slave) based communication.

## **Remote point**

It is a definition of a register/binary (or a group of them) that correlate to inputs/outputs on a device (slave) communicating via the MODBUS RTU protocol.

## **DM-xxx modules**

Modules enabling the user to extend the number of inputs and outputs of the control system using ARION communication network. It is possible to connect up to 63 of these modules into a single ARION network.

## **DMM-xxx modules**

Modules enabling the user to extend the number of inputs and outputs of a device on a master MODBUS RTU network device by using the MODBUS RTU communication network. It is possible to connect up to 63 of these modules into a single MODBUS RTU network.

## 2 Counter inputs, measurement of rpm / impulses

---

All control system developed by AMiT that feature digital inputs can be used for impulse counting / RPM measuring.

Impulse counting and rpm measuring can be implemented – depending on the control system type – by the following means:

- ◆ Program counter
- ◆ Program counter INT
- ◆ HW counter

When the control system is not equipped with digital inputs but is equipped with either RS485 or RS232 communication line (with the RS232 line requiring an RS232/RS485 converter), the impulse counter (up to the frequency of 25 Hz) may be implemented by using the **DM-DI24** (ARION protocol communication) or **DMM-DI24** (MODBUS RTU protocol communication). It is described in chapter 6 “Counting impulses via extension modules”.

### Program counter

Each digital input can be operated in Hi\_x processes. The programme operation can use these inputs as counter inputs. The limitation of the input signal frequency is determined by the programme. It is usually usable to up to the frequency of 250 Hz. Further information is available in chapter 3 “Program counter”.

### Program counter INT

Some digital inputs may generate hardware interrupts. Through software, Interrupt\_x processes can be used as counters or revolution counters. The limitation of the input signal frequency is determined by the programme. We can usually use it up to frequency 10 kHz. Further information is available in chapter 4 “Program counter INT”.

### HW counter

Some control systems are equipped with counter inputs supported by HW, or rather inputs for incremental position sensors. Programme operation is implemented via **IRCxxx** modules. Further information is available in chapter 5 “HW counter”.

## 3 Program counter

---

When the system features digital inputs, it is possible to operate these inputs in fast processes and use them as impulse counters. The system is parametrised in DetStudio IDE through the use of **ImpIn** and **DImp** modules.

The limitation of the input signal frequency is determined by the programme. It is usually usable to up to the frequency of 250 Hz.

Place the **DImp** module in any proper process. Place the **ImpIn** module in a quick process (either Quick or Hi\_x type) the period of which is chosen according to the length of the impulse. The period of the process containing the **ImpIn** module needs to be shorter or equal to a half of the impulse length – i.e. if the impulse is 10 ms long, a period of the quick process needs to be 5 ms or shorter.

### 3.1 Programme operation

---

#### 3.1.1 Impln module

---

The **ImpIn** module handles up to sixteen impulse inputs loaded from a chosen logical DI channel. It doesn't save the processed values into the database but rather into its inner variables. These values are further processed by the **DImp** module. The **ImpIn** module needs to be placed in a quick process (either Quick or Hi\_x type).

Each of the sixteen signals has an active leading or trailing edge, or both edges. In unshaped signals, only the leading edge is usually active.

#### 3.1.2 DImp module

---

**DImp** reads data from the impulse input and converts it to a physical dimension. The module cooperates with the module for operating the **ImpIn** impulse inputs. Place the **DImp** module in any proper process (Proc00 to Proc15). When reading impulses from multiple inputs, insert the **DImp** module multiple times.

The **DImp** module processes data from sensors with impulse output where "1 impulse = N physical units". Based on this N value (sensor constant), the module deduces the state of the flow counter (e.g. water/gas/electricity meter counter) and estimates the instantaneous value.

#### **Caution**

*When using a single **ImpIn** module, it is impossible to access the same signal via multiple **DImp** modules!*

### 3.2 Example no. 1 – programme counter

---

Connected to input DI0.0 of the control system, there will be a device sending 100 ms long impulses. Minimum pause between pulses is 100 ms. The physical significance of an impulse is 0.25 MJ. The values of the counter will be loaded every 5 s. The counter will reset when the value reaches 1 GJ.

The **ImpIn** module needs to be in a quick process with a period shorter or equal to a half of the impulse length. For the given example, it is a process with a 50 ms period.

The following code fulfils the required function.

## Quick process, period of 50 ms

---

```
// operation of sexdecuplets (sets of 16) impulse inputs of channel 0
:10000 ImpIn #0, 100, 30, 0xFFFF, 0x0000
```

## Proper process, period of 5 ms

---

```
// loads and converts the impulse input DI0.0
DImp :10000, 0, Delta, Sum, Instant, Constant, SyncIn.0, NONE.0
// when the counter reaches 1 million units, reset it
Let SyncIn = if(Sum >= 1000000, 1, 0)
```

**Sum** is a flow counter; **Delta** is the difference between the last and the current value of the counter; **Instant** is the estimation of the instantaneous value; **Constant** contains the constant of the meter (0.25 MJ/impulse).

The above-mentioned example is included in attachment ap0017\_en\_03.zip as counterinputs\_p1\_en\_xx.dso. This application was created for the **AMiNi4DW2** control system. However, it can be modified to suit any control system fitted with digital inputs via the DetStudio menu ("Tools/Change station...").

## 4 Program counter INT

Some digital inputs may generate hardware interrupts. Through software, Interrupt\_x processes can be used to load impulses via the inputs.

The limitation of the input signal frequency is determined by the programme. It is typically usable to up to 10 kHz.

### 4.1 Interrupt processes

The application supports up to 16 interrupt processes (in the DetStudio IDE marked as Interrupt\_0 to Interrupt\_15) that are called after an interrupt by an external event. This external event means a change in the level of the digital signal received by the control system input. The process may be started based on the leading edge of the signal, the trailing edge of the signal, or both. The number of available interrupt processes and their connection to corresponding inputs depends on the control system type (see the following table).

| Process      | ADiS<br>(AD-CPUW2 + 2× AD-FDI8) | AMiNi-ES |
|--------------|---------------------------------|----------|
| Interrupt_0  | FDI0.0                          | DI0.0    |
| Interrupt_1  | FDI0.1                          | DI0.1    |
| Interrupt_2  | FDI0.2                          | DI0.2    |
| Interrupt_3  | FDI0.3                          | DI0.3    |
| Interrupt_4  | FDI0.4                          | DI0.4    |
| Interrupt_5  | FDI0.5                          | DI0.5    |
| Interrupt_6  | FDI0.6                          | DI0.6    |
| Interrupt_7  | FDI0.7                          | DI0.7    |
| Interrupt_8  | FDI1.0                          | –        |
| Interrupt_9  | FDI1.1                          | –        |
| Interrupt_10 | FDI1.2                          | –        |
| Interrupt_11 | FDI1.3                          | –        |
| Interrupt_12 | FDI1.4                          | –        |
| Interrupt_13 | FDI1.5                          | –        |
| Interrupt_14 | FDI1.6                          | –        |
| Interrupt_15 | FDI1.7                          | –        |

**Note**

With the **ADiS** control system, the processes *Interrupt\_0* to *Interrupt\_7* are tied to the first **AD-FDI8** module in the assembly (**AD-FDI8** closest to CPU). The processes *Interrupt\_8* to *Interrupt\_15* are tied to the second **AD-FDI8** module in the assembly. Assignment of individual logical channels of these modules is insignificant.

**Caution**

When the **ADiS** control system set contains an AD-AO8x module, it is possible to use only 8 interrupt processes. When there are two AD-AO8x modules, no interrupt process can be used.

### 4.2 Programme operation

The Interrupt\_x process can be easily used as an impulse counter – the impulses will be fed to the input of the control system where the interrupts originate (that includes the corresponding Interrupt\_x process, see table above). The counting of inbound impulses itself will be handled by this process.



## 4.2.1 RPM measuring

---

To measure RPM via the impulse sensor, use the **RPM1** module. The module measures RPM on a digital input (one that creates interrupts). When measuring RPM on multiple inputs, insert the **RPM1** module multiple times. Insert the module into an `Interrupt_x` process tied to the input where the RPM measurement is required. The language of the process needs to be either LA or RS.

In the `Interrupt_x` process parameters, define whether the process (and the **RMP1** module) should start at the leading edge or the trailing edge of the signal. Except for rare cases, it is strongly advised against setting the `Interrupt_x` process start to both edges – it would undesirably affect measuring accuracy. It is recommended to avoid further modules in the `Interrupt_x` process with the **RPM1** module.

The frequency of impulses notwithstanding, the content of the output variable is updated every 10 ms. The output variable may be read in any process.

The total of impulses per time unit for all **RPM1** modules used in the application mustn't exceed 80,000/minute. That being said, it is necessary to realise that anything above 40,000/minute may cause serial-communication failures due to the process taking too much CPU time. The above-mentioned numbers may differ depending on the control system type.

## 4.3 Example no. 2 – programme counter INT, measuring revolutions

---

Connected to input DI0.0 of the control system, there will be a device sending impulses. The physical significance of an impulse is 1 m<sup>3</sup>. The counter will reset when the value reaches 1 million m<sup>3</sup>.

Furthermore, the signal from the impulse RPM counter will be fed to input DI0.1. The sensor generates 8 impulses during a single rotation.

The following code fulfils the required function.

### Process Interrupt\_0

---

```
// counting impulses fed to input DI0.0 with the leading edge active
IncDec count_value, 1, 1.000
```

The total of loaded impulses (or rather their value in m<sup>3</sup>) is in the `count_value` variable.

### Process Interrupt\_1

---

```
// RPM measuring from impulses fed to input DI0.1 with the leading edge active
RPM1 0x0808, 10.000, Revolutions
```

Moving average of the last eight impulse lengths is used to eliminate vibrations and inaccuracies when measuring the placement of the eight marks on the revolutions meter disc. Revolutions start to be measured at 10 RPM. The measured value is saved into the `Revolutions` variable.

### Proper process, period of 1 ms

---

```
// evaluation (potentially a reset) of the loaded value of the counter
Let count_value = if(count_value > 1000000, 0, count_value)
```

The above-mentioned example is included in attachment `ap0017_en_03.zip` as `counterinputs_p2_en_xx.dso`. This application was created for the **AMiNi-ES** control system.

## 5 HW counter

The **AMiNi-Es** control system is equipped with HW counters. The control system is equipped with three counter inputs (numbered 0 to 2). Signal voltage levels need to be at 24 V DC. The max. counting frequency is 160 kHz. These are fast, HW supported counter inputs, or rather inputs for incremental position sensors. For each counter input, it is possible to select a specific usage purpose or a sensor type connected to it.

### 5.1 Input modes

The counter input reacts differently to different combinations of input signals (F1 and F2) and edges according to the selected mode.

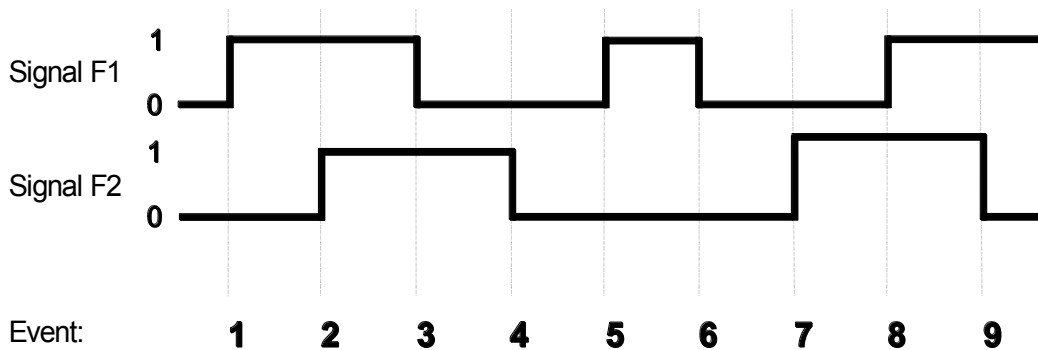


Fig. 1 – Event definition

The “+” in the next table means value increases, “-” means value decreases and the empty field means that the counter does not react to the event.

#### Reactions to events

| Input mode      | Active edge settings   | Reaction to an event: |   |   |   |   |   |   |   |   |
|-----------------|------------------------|-----------------------|---|---|---|---|---|---|---|---|
|                 |                        | 1                     | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| IRC F1-F2       | Not taken into account | +                     | + | + | + | + | - | - | - | - |
| IRC F2-F1       | Not taken into account | -                     | - | - | - | - | + | + | + | + |
| Direction + / - | Leading                |                       | - |   |   |   |   | + |   |   |
|                 | Trailing               |                       |   |   | + |   |   |   |   | - |
|                 | Both edges             |                       | - |   | + |   |   | + |   | - |
| Direction - / + | Leading                |                       | + |   |   |   |   | - |   |   |
|                 | Trailing               |                       |   |   | - |   |   |   |   | + |
|                 | Both edges             |                       | + |   | - |   |   | - |   | + |
| Upwards         | Leading                |                       | + |   |   |   |   | + |   |   |
|                 | Trailing               |                       |   |   | + |   |   |   |   | + |
|                 | Both edges             |                       | + |   | + |   |   | + |   | + |
| Downwards       | Leading                |                       | - |   |   |   |   | - |   |   |
|                 | Trailing               |                       |   |   | - |   |   |   |   | - |
|                 | Both edges             |                       | - |   | - |   |   | - |   | - |

## 5.2 Properties of counter inputs

Parameters and modes of counter inputs of the **AMiNi-ES** control system:

| <b>Input "0"</b>   |   |   |
|--|---|---|
| Number of the input  | 0   |   |
| Signal F1  | DI0.1   |   |
| Signal F2  | DI0.0   |   |
| Range of the inner counter   | 32 bits (-2,147,483,648 to 2,147,483,647)   |   |
| Inputs for the position marks (indexes) in the automatic mode      | None. Use any free digital input for semi-automatic operation mode of the position marks. |   |
| <b>Supported combinations of modes and active edge selections:</b> |   |   |
| Mode   | Edges   | Note  |
| IRC F1-F2  | Not taken into account  | A single period of input signals corresponds to 4 unit changes of the inner counter. (The counter changes with every edge of any of the input signals). |
| IRC F2-F1  | Not taken into account  |   |
| Direction + / -  | Leading   | Input signal F1 doesn't have any impact on the activity of the input. The corresponding digital input may be used as usual.                             |
|  | Trailing  |   |
| Both edges   | Leading   |   |
|  | Trailing  |   |
| Direction - / +  | Leading   |   |
|  | Trailing  |   |
| Both edges   | Leading   |   |
|  | Trailing  |   |
| Upwards  | Leading   | Input signal F1 doesn't have any impact on the activity of the input. The corresponding digital input may be used as usual.                             |
| Downwards  | Trailing  |   |
|  | Both edges  | Leading   |
| Both edges   |   | Trailing  |

| <b>Input "1"</b>   |   |      |
|--|---|------|
| Number of the input  | 1   |      |
| Signal F1  | DI0.3   |      |
| Signal F2  | DI0.2   |      |
| Range of the inner counter   | 32 bits (-2,147,483,648 to 2,147,483,647)   |      |
| Inputs for the position marks (indexes) in the automatic mode      | None. Use any free digital input for semi-automatic operation mode of the position marks. |      |
| <b>Supported combinations of modes and active edge selections:</b> |   |      |
| Mode   | Edges   | Note |
| The same as for input "0".   |   |      |

| Input "2"   |   |      |
|---|---|------|
| Number of the input   | 2   |      |
| Signal F1   | DI0.5   |      |
| Signal F2   | DI0.4   |      |
| Range of the inner counter                                    | 32 bits (-2,147,483,648 to 2,147,483,647)   |      |
| Inputs for the position marks (indexes) in the automatic mode | None. Use any free digital input for semi-automatic operation mode of the position marks. |      |
| Supported combinations of modes and active edge selections:   |   |      |
| Mode  | Edges   | Note |
| The same as for input "0".                                    |   |      |

### Note

The signals from inputs DI0.0 to DI0.5 are accessible in corresponding logical channels DI0 and DI0AC (numbers 0 and 1) regardless of the usage (or the lack of) of function modules for using those inputs as counter inputs/inputs for incrementing the position sensors. Nothing stands in the way of using these signals as conventional digital inputs.

## 5.3 Programme operation

To handle the counter inputs, or rather inputs for incremental position sensors, use the following modules:

- ◆ **IRCMoDe** – For setting the mode and the HW counter input constants.
- ◆ **IRCPreset** – For setting the default value (position marker) of the HW counter input.
- ◆ **IRCSet** – For setting the value of the HW counter input.
- ◆ **IRCIIn** – For reading the value of the HW counter input.

Detailed description of modules is in the DetStudio/PseDet help.

The counter inputs, or rather inputs for incremental position sensors, are defined by two input signals (F1 and F2) or by two inputs for incremental position, see chapter 5.2 "Properties of counter inputs".

## 5.4 Example #3 – hardware counter

Connected to the the control system, there will be a device sending signals (a single signal). Values will be loaded from the counter every 10 s. At the same time, it will be necessary to calculate the difference between the current count of impulses and the previously measured one. The counter will be reset when it reaches 5 million or with the (re)start of the control system.

### Counter parameters

The selected control system features three HW supported counter inputs. For counting the impulses from a single connected signal, the best approach is to use the "upwards" mode – with every active edge (leading/trailing/both) on input F2 causing the counter to increment its value. Input signal F1 has no impact on the counter operation in this mode. It is possible to use the mode for both counter inputs. Use counter "0" – the signal from the device is fed to input F2, i.e. input DI0.0.

The programme itself is implemented using the following code.

## Process INIT

---

Counter “0” in process INIT is set to “upwards”. The counter input will react to the leading edge. It will increase by 1 with each incoming impulse. If the setup of the counter mode finishes successfully, the 0th bit of the `set` variable is set to 1. Furthermore, the default value of the counter is for this process set to 0.

```
// initialisation of the counter input mode
IRCMoDe 0, 4, 1, 0.000, 1.000, set.0
// reset counter
IRCSeT 0, 0.000
Let count_old = 0
```

## Proper process, period of 10 ms

---

In this process, the counter value is read. It is then used to calculate the difference between the current and the previous counter value. When the counter exceeds the predefined value, it is reset.

```
// load the counter value
IRCIn 0, count_value
// evaluation of the read value
Let ok = if ((count_old < count_value) AND (count_value < 5000000), 1, 0)
If ok.0
    // difference calculation
    Let count_delta = count_value - count_old
    Let count_old = count_value
Else
    // reset counter
    IRCSeT 0, 0.000
    Let count_old = 0
EndIf
```

The above-mentioned example is included in attachment `ap0017_en_03.zip` as `counterinputs_p3_en_xx.dso`. This project was created for the **AMiNi-ES** control system.

## 6 Counting impulses via extension modules

When the control system features communication line RS485 or RS232, it is possible to use the extension modules for impulse counting. Extension modules **DM-DI24** and **DMM-DI24** support the use of function for counting incoming impulses on any of their inputs. This function partially solves problems with short impulse detection. However, it is crucial to take the following properties into account when using this function:

- ◆ Max. value of the counter is 16,383 (number 14-bit) for **DM-DI24** and 32,767 (number 15-bit) for **DMM-DI24**. After another impulse is added, the counting starts again from zero.
- ◆ Maximum frequency of incoming impulses is 25 Hz. With a higher frequency, there is no guarantee that all incoming impulses are recorded.
- ◆ The module's internal counter resets when the power-supply voltage is disconnected; it is also possible to reset it in the programme.
- ◆ It is necessary to tend to the internal counter overflow in terms of programming.

### 6.1 DM-DI24 extension module

Communication with the **DM-DI24** extension module occurs via the ARION protocol. More information is available in the Application note *AP0025 – Communication in the ARION network – table definition*.

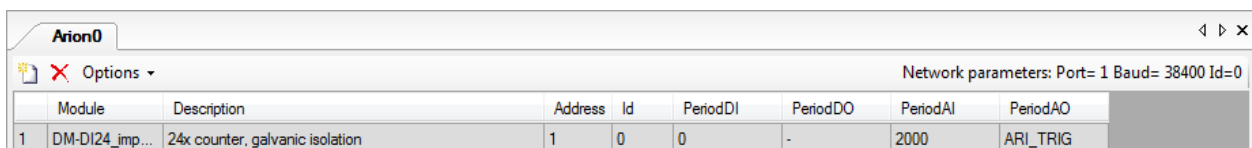
**Note**

The function for counting the incoming impulses on any of the inputs of the **DM-DI24** is available since firmware version 1.65.

#### 6.1.1 Example #4 – counting impulses on DM-DI24

An example of implementation of communication via the ARION protocol (38,400 bps, port 1) with **DM-DI24** (address 1).

ARION network with a **DM-DI24** extension module is defined according to the following picture.



| Module | Description                                       | Address | Id | PeriodDI | PeriodDO | PeriodAI | PeriodAO |
|--------|---|---------|----|----------|----------|----------|----------|
| 1      | DM-DI24_imp...<br>24x counter, galvanic isolation | 1       | 0  | 0        | -        | 2000     | ARI_TRIG |

Fig. 2 Definition of ARION network with a **DM-DI24** extension module

To use the **DM-DI24** module in impulse counting mode, it is required to define a module named DM-DI24\_impulse in the Arion0 table. It is further required to set the value of parameter PeriodDI to 0 and parameter PeriodAI to the required value. At the same time, the module can be used to load states of the digital inputs. In this case, set parameters PeriodDI and PeriodAI to required values. It is then possible to use the **ARI\_DigIn** module to read the current state of DI inputs of the **DM-DI24** extension module and the **ARI\_AnIn** module to read the state of DI inputs counter states.

#### Loading counter values of all inputs

```
ARI_AnIn 1, 0, 24, DataCount_all[0,0], NONE[0,0], 16384.0, 0.0, 16383.0, 0.0, 16383.0
```

**Note**

It is necessary to prevent counter overflow! Counter range is 0 to 16383. I.e. a counter generates a row of numbers 0; 1; 2; ...; 16,382; 16383; 0; 1; 2; ... Therefore, it will be necessary to count the

number of overflows “*p*” of the counter, and the resulting sum of pulses will therefore be equal to:  
**sum = *p* × 16,384 + counter.**

## Setting the value of all counters

```
ARI_AnOut 1, 0, 24, SetCount[0,0], NONE[0,0], 16384.0, 0.0, 16383.0, 0.0, 16383.0
// setting the counter values
If Set_up.0
    // issuing the transfer into DM-DI24
    ARI_Trig 1, 1
    // resetting the flag for setup
    Let Set_up = 0
EndIf
```

To set the values into the buffer through the `ARI_AnOut` module, it is required to trigger the transfer itself via the `ARI_Trig` module because, in case of AO node of **DM-DI24**, the values are not transferred periodically.

### Caution!

**In firmware version 1.65, there was an issue with writing the required value into individual counters.** The value written into AO0 is instead written into AO7, value written into AO1 is instead written into AO6, and so on. This problem occurs in octuplets, i.e. the value written to AO8 is instead written into AO15, the value written to AO16 is instead written into AO23, and so on. This issue is solved as of firmware version 1.66. **When replacing the DM-DI24 module, it is necessary – depending on the DM-DI24 firmware version – to also adjust the control system SW!**

The above-mentioned example is included in attachment ap0017\_en\_03.zip as counterinputs\_p4\_en\_xx.dso. This project was created for control system **AMiNi4DW2**. However, it can be modified to suit any control system fitted with a serial communication line using the DetStudio menu “Tools/Change station”.

## 6.2 DMM-DI24 extension module

Communication with the **DMM-DI24** extension module occurs via the MODBUS RTU protocol. More information is available in the Application note *AP0008 – Communication in MODBUS RTU network (PseDet)*.

### 6.2.1 Example #5 – counting impulses on DMM-DI24

An example of implementation of communication via the MODBUS RTU protocol (38,400 bps, even parity, one stop-bit, port 1) with **DMM-DI24** (address 1).

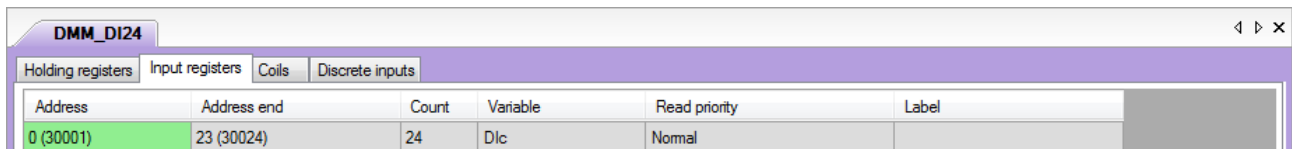
Positions of remote points (registers/binaries) in individual modules are always determined by the number of the given input/output of the **DMM-DI24** module.

To read/write counter values from/to the **DMM-DI24** module, it is possible to use the following MODBUS functions.

| Functions | Usage with DMM-DI24                          |
|-----------|--|
| 3         | Reading counter values.                      |
| 4         | Reading counter values (same as function 3). |
| 6         | Writes the value of a single counter.        |
| 16        | Writes the value of all counters.            |

## Reads the value of counters on all inputs

The **DMM-DI24** module supports reading the counter states both via the input registers and via the holding registers. In terms of the application note, the input registers will be used for reading, see the following picture.



| Address   | Address end | Count | Variable | Read priority | Label |
|-----------|-------------|-------|----------|---------------|-------|
| 0 (30001) | 23 (30024)  | 24    | Dlc      | Normal        |       |

Fig. 3 – Definition of reading the values of all counters

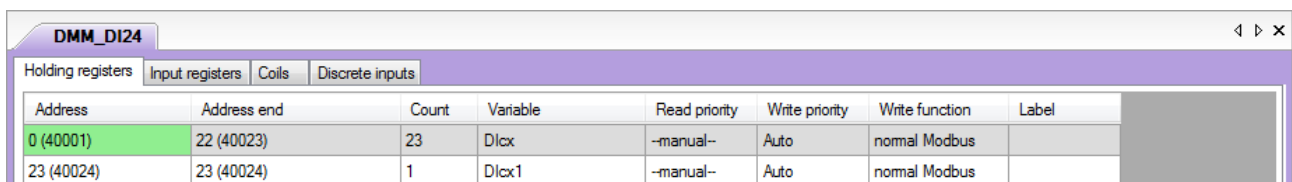
**Dlc** is a  $1 \times 24$  MI-type matrix. This construction ensures that values of all counters of the extension module are read with a period of 1 s (“Normal” priority).

### Note

*It is necessary to pay attention to counter overflow! Counter range is 0 to 32767. Therefore, a counter generates a row 0; 1; 2; ...; 32,766; 32,767; 0; 1; 2; ... Therefore, it will be necessary to count the number of overflows “p” of the counter, and the resulting sum of pulses will be therefore equal to:  $sum = p \times 32,768 + counter$ .*

## Setting the counter values

Setup of required counter values can be defined via the holding registers according to the following picture.



| Address    | Address end | Count | Variable | Read priority | Write priority | Write function | Label |
|------------|-------------|-------|----------|---------------|----------------|----------------|-------|
| 0 (40001)  | 22 (40023)  | 23    | Dlcx     | -manual-      | Auto           | normal Modbus  |       |
| 23 (40024) | 23 (40024)  | 1     | Dlcx1    | -manual-      | Auto           | normal Modbus  |       |

Fig. 4 – Definition of the setup of values to multiple counters and one counter

**Dlcx** is a  $1 \times 23$  MI-type matrix. Writing into any cell of the matrix triggers writes of all values in the matrix to counters on inputs DI0 to DI22. **Writes are performed even when the same value is written into the cell (i.e. the value “doesn’t change”).**

**Dlcx1** is a scalar Integer variable. Writing into the variable triggers a write of the variable value into the counter at DI23. **The write is performed even when the same value is written into the variable (i.e. the value “doesn’t change”).**

The above-mentioned example is included in attachment ap0017\_en\_03.zip as counterinputs\_p5\_en\_xx.dso. This project was created for the control system **AMiNi4DW2**. However, it can be modified to suit any control system fitted with a serial communication line using the DetStudio menu “Tools/Change station”.



## 7 Technical support

---

The AMiT Technical Support Department provides all information regarding counter inputs. The Technical support is best contacted via e-mail at **support@amit.cz**.

## 8 Warning

---

In this document, AMiT, spol. s r. o. provides information as it is, and the company does not provide any warranty concerning the contents of this publication and reserves the right to change the documentation content without any obligation to inform anyone or any authority about it.

This document can be copied and redistributed under the following conditions:

1. The whole text (all pages) must be copied without making any modifications.
2. All redistributed copies must retain the AMiT, spol. s r. o. copyright notice and any other notices contained in the documentation.
3. This document must not be distributed for profit.

The names of products and companies used herein may be trademarks or registered trademarks of their respective owners.