Using the DT80 with BrainChild IO-16DI Modules

1 Prerequisites

- Nil

2 Required Equipment

- *dataTaker* DT80 range data logger (firmware version 8.08 or above)
- BrainChild IO-16DI Modbus Expansion Module
- Connecting wires

3 Process

3.1 Connect the Module to the dataTaker

The BrainChild module connects to the DT80 Serial Sensor port via RS485. In this exercise we will also power the BrainChild module using the 12V output terminal on the dataTaker.

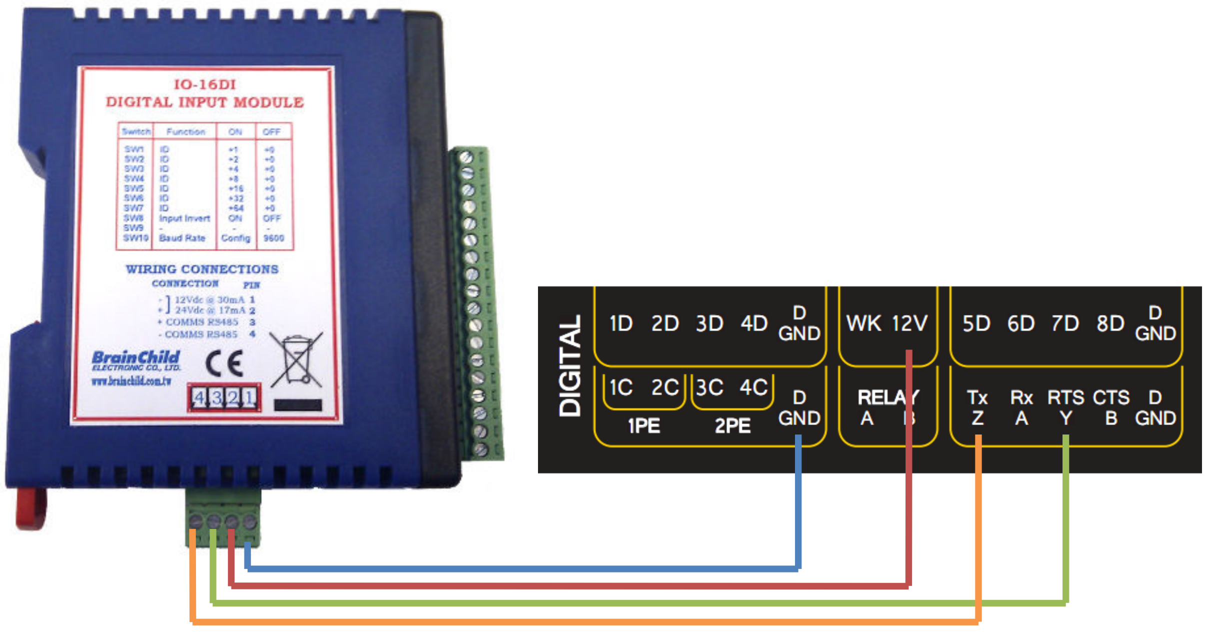

Figure 1 – RS485 Wiring Diagram

IO-16DI Terminal	DT80 Range Logger Terminal
1	**GND**
2	**12V**
3	**RTS Y**
4	**TX Z**

Table 1 – RS485 Connection List

NOTE: If you are powering many modules then you should use a separate power supply, as the dataTaker 12V output can only supply 150mA.

3.2 Set the Modbus Address of the Module

Use the DIP switches on the front of the BrainBhild Module to select a Modbus address. In the example below we have chosen to use address number 1 (DIP1 on, others off).

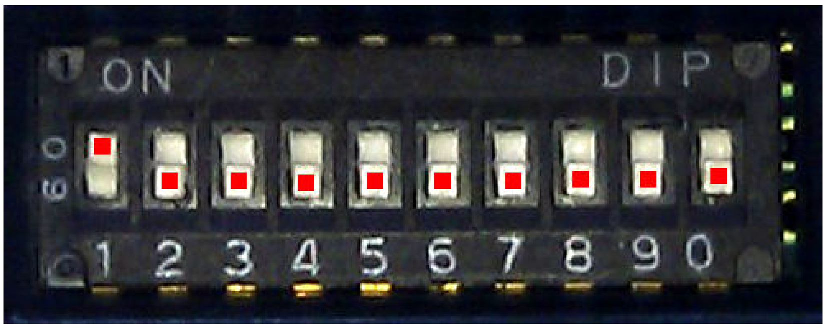

Figure 2 - Setting the Modbus Address (Address 1)

It is essential that each device connected to the same RS485 network is unique, otherwise conflicts will occur and your system will behave in an undesirable manner.

For a complete list of the available addresses and DIP switch settings, please consult the IO-16DI user manual.

3.3 Set Up the DT80 Serial Sensor Port

If DIP0 is switched to off as shown in Figure 2 above then the serial communications for the BrainChild Modules will be set to default (9600,8,1,N). It is thus necessary to configure the DT80 Serial Sensor port profiles to match this. You should also change the Modbus Server port to a known value.

Send the following profile commands to the logger:

```
PROFILE SERSEN_PORT BPS=9600
PROFILE SERSEN_PORT FLOW=NONE
PROFILE SERSEN_PORT DATA_BITS=8
PROFILE SERSEN_PORT STOP_BITS=1
PROFILE SERSEN_PORT PARITY=NONE
PROFILE SERSEN_PORT MODE=RS485
PROFILE SERSEN_PORT FUNCTION=MODBUS_MASTER
PROFILE MODBUS_SERVER SERSEN_ADDRESS=0
```

3.4 Activate the Regulated 12V Power Output

Activating the 12V power output requires one single command:

```
PWR12V=1
```

This command, if used within a program/configuration, should be placed in the immediate schedule (called “on logger activation” in dEX).

3.5 Attaching Sensors to the BrainChild Module

There are many types of sensors which can be connected to the module, however most will either be contact closure or transistor output types. The diagrams below are from the BrainChild IO-16DI user manual:

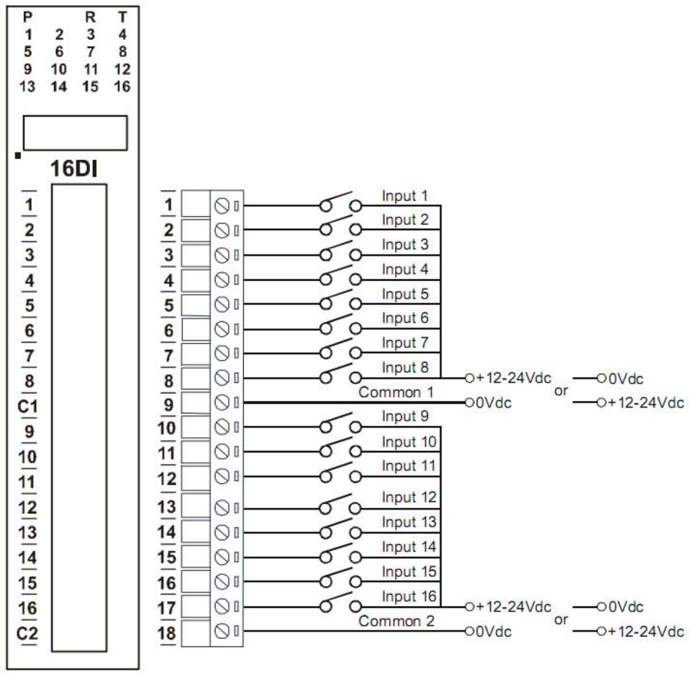

Figure 3 - Contact Closure Type Input Options

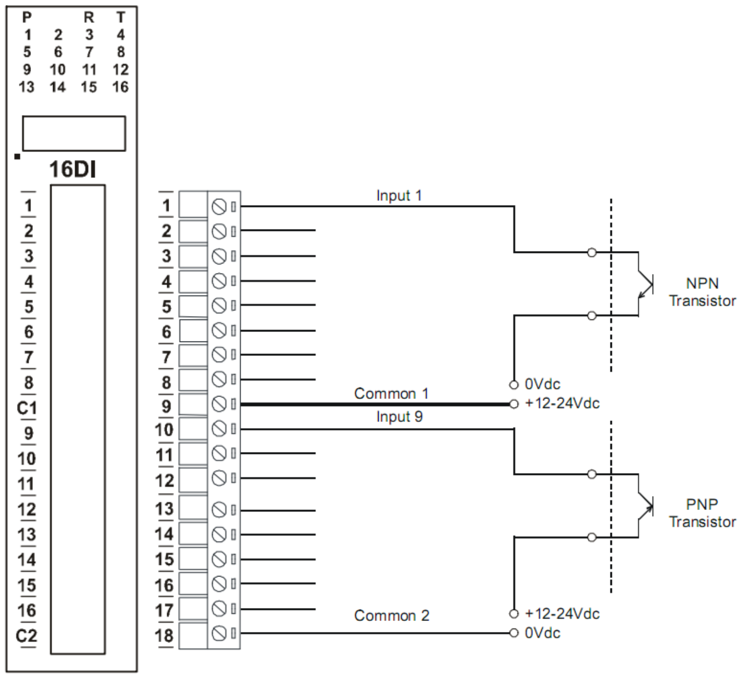

Figure 4 - Transistor Type Input Options

NOTE: The numbers of the shown terminal block do not match the labels on the IO-16DI.

3.6 Modbus Commands

There are four Modbus commands which will be used in this section:

1. Read a digital state
2. Activate counters
3. Read counters
4. Set/reset counters

3.6.1 *Read a Digital State*

Reading a digital state is straight forward and requires a simple command

```
1MODBUS(ADX,R1:N)
```

where:

- **X** is the address of the BrainChild module
- **N** is the digital input number which is to be read

Example:

```
1MODBUS(AD1,R1:14)
```

(Read the state of digital input #14 on the module with address #1)

3.6.2 *Bulk Reading Digital States into Channel Variables*

It is also possible to read several consecutive states into a list of channel variables. This would be performed for the purpose of speed and sometimes to simplify the program.

```
1MODBUS(ADX,R1:N,=p..qCV)
```

where:

- **X** is the address of the BrainChild module
- **N** is the ***first*** digital input number which is to be read
- **p** is the first channel variable to store a value
- **q** is the last channel variable to store a value

NOTE: The number of values read from the module depends on the range of channel variables specified in the command. This means **q-p+1** values are read from the slave.

Example:

```
1MODBUS(AD1,R1:5,=1..5CV)
```

(Read the state of digital inputs 5-9 into channel variables #1-5)

3.6.3 *Change Counter Mode*

Each input on the IO-16DI can be used as a counter, but these are disabled by default. To change the counter mode we need to write a value to register type 4, number 101.

```
1MODBUS(ADX,R4:101)=N
```

where:

- **X** is the address of the BrainChild module
- **N** is the counter mode, which can be either:
  - 0 = disabled
  - 1 = enabled (standard upward counter)
  - 2 = enabled (up/down counter)

In counter mode 2 the inputs will act as up/down counters. Input 1 will increment counter 1 while input 2 decrements counter1. In the same way, inputs 3 and 4 operate counter 2; inputs 5 and 6 operate counter 3 and so on… A consequence of using this counter mode is halving the total number of counters.

Example:

```
1MODBUS(AD1,R4:101)=1
```

(Activate the standard upward counter on the module with address number 1)

3.6.4 Read Counter

Reading a counter is much like reading a digital input, however the returned value is a 32-bit integer, so its value spans across two registers, which are each 16-bits wide. For this reason we use the MBL channel option, which automatically converts the two 16-bit registers into a single 32-bit value.

NOTE: The module returns a 32-bit ***unsigned*** integer value, which potentially stores numbers up to 4,294,967,296 however the dataTaker uses 32-bit ***signed*** integer values, which means only the lower 31 bits can be used (since the highest bit changes the sign). In short, this means you should reset the counters before they reach a value of 2,147,483,648.

```
1MODBUS(ADX,R4:N,MBL)
```

where:

- **X** is the address of the BrainChild module
- **N** is the register number, which is related to the counter number using the equation ((2xC)+1), where C is the counter number

Example:

```
1MODBUS(AD1,R4:33,MBL)
```

(read counter number 16 [(2 x 16)+1=33])

3.6.5 Set/Reset Counter

Counters may need to be set to a starting value or zeroed. To do this you will use the following command:

```
1MODBUS(ADX,R4:N,MBL)=M
```

where:

- **X** is the address of the BrainChild module
- **N** is the register number, which is related to the counter number using the equation ((2xC)+1), where C is the counter number
- **M** is the value which you would like to assign to the counter

Example:

```
1MODBUS(AD1,R4:31,MBL)=0
```

(Reset counter number 15 [(2 x 15)+1=31] to zero)

4 Putting it together

The example program below will set up the Modbus port, read digital inputs 1-5 every 5 seconds, read a counter on channel 6 every 10 seconds and reset that counter at midnight every day.

```
BEGIN"IO16DI"
'=====================================================
'     DEVICE SET UP
      '--PROFILES--
      PROFILE SERSEN_PORT BPS=9600
      PROFILE SERSEN_PORT FLOW=NONE
      PROFILE SERSEN_PORT DATA_BITS=8
      PROFILE SERSEN_PORT STOP_BITS=1
      PROFILE SERSEN_PORT PARITY=NONE
      PROFILE SERSEN_PORT MODE=RS485
      PROFILE SERSEN_PORT FUNCTION=MODBUS_MASTER
      PROFILE MODBUS_SERVER SERSEN_ADDRESS=0
      '--POWER OUTPUT--
      PWR12V=1
      '--ACTIVATE COUNTERS--
      1MODBUS(AD1,R4:101)=1
'=====================================================

'=====================================================
'  Schedule A (Read digital inputs)
'   - Runs every 5 seconds
'=====================================================
RA("B:",ALARMS:OV:10KB,DATA:OV:1MB)5S LOGONA
      1MODBUS(AD1,R1:1,=1..5CV,W)
      1CV("Digital 1")
      2CV("Digital 2")
      3CV("Digital 3")
      4CV("Digital 4")
      5CV("Digital 5")

'=====================================================
'  Schedule B (Read counter input)
'   - Runs every 10 seconds
'=====================================================
RB("B:",ALARMS:OV:10KB,DATA:OV:1MB)10S LOGONB
      1MODBUS("Counter 6",AD1,R4:13,MBL)

'=====================================================
'  Schedule C (Reset Counter)
'   - Runs every day at midnight
'=====================================================
RC("B:",ALARMS:OV:10KB,DATA:OV:1MB)1D LOGONC
      1MODBUS(AD1,R4:13,MBL)=0
END
```