Norhof LN2 microdosing systems
Galileilaan 33U
6716BP Ede, Holland
tel (00-31)-(0)85-1045715
info@norhof.com

version dec 2022
not a complete manual, but additional information to start with.

#915 pump, driving by RS232 commands, or the driver.

The Norhof LN2 pumps can be connected to a PC by RS232 interface.
The supplied monitor program installs the driver (Pump913drv.exe) which communicates directly to the RS232 port of the pump. This driver is linked as a class module in the Monitor program, but can be used simultaniously for other programs.
For user defined programs, it is highly recommended to make use of this driver.
This driver can be used as class module in Visual Basic, but also in other program languages, like C++.
Big advantage is that during development also the Monitor program can run, so incomming data can be verified on the screen of the PumpMonitor.

The driver polls on certain moments data from the pump, and stores this data to make it easy accessable from the user software. The user program only needs to read the data from the driver, while the driver itself takes care of the communication between itself and the pump. Also in the driver all values are calculated into mBars, degrees Celcius and Fahrenheit, Volts, etc., including all offsets, scaling and calibration values.
The timing of the data polling is determined in a way that there is fast access for faster changing values, and slower acces for data which may change slowly.
Ofcourse the questioning to the pump for data will interupt the processor in the pump, so too fast data request may cause the processor in the pump to react too slow on regulation of pumping itself.
For steering the pump (in mode E and F) 3 registers are available to switch the pump ON and OFF, give parameters for pumping pressure and temperature.

Ofcourse direct driving of the RS232 interface is also possible, but in that case our Monitor program can not work at the same time. Also, you yourself should take care of reading the RAM locations in the pump which corresponds with all data. The data in the pump are all in steps of the ADC converter, or numbers for all working functions. For most values an offset, calibration and scaling factors should be used to convert this data to usable values.
Reading of the RAM locations is not dangereous, but writing into RAM (to steer the pump), can be dangereous, because when writing accidentally to a wrong address, may cause the processor to work with wrong values, reacting in perhaps steering the heater in the pump wrong, or steering the electronics in the pump in a way it was not designed for, so in worse case it could damage something.
Also, all calibration values, setpoints and other fixed parameters are stored in EEprom. If the processor crashes, theoreticly the processor could write in its own EEprom, making the pump unusable.
In case of damaged EEprom the pump should be returned to us to fix this, because there is no way of reprogramming the pump on distance. (there is no room enough to include a firmware update program in the pump)
It may be clear that using the pump in direct RS232 access there are some riscs.

Using the driver for user defined programs.

On the CDrom supplied with the #915pump, there is a map "915utils". In this map is a setup which installs the DataRecorder program, but it also installs some Visual Basic files, from which NorhofPumpDrive.vbp includes the NorhofPumpDrive.frm.

With this form, all communication with the driver can be set up.
For C++ users, this source code can be easily changed to C++ to make it to work in C++
In the beginning text of this form is some more information to link the driver as a class module to this form.
If this form runs without errors, this form can be used as start of your application program.
The driver automaticly generates a sequence of reading commands to the pump, every 0.1, 0.5 and 5 seconds.
The PumpReady routine is called contineously with different values, to indicate that data in the driver may have been changed. Reading the prCommand type defines what command was put in the pump.
Reading the mBuffer gives the data which was returned by the pump at that moment.
The result was stored in the driver on the correct place, and the (possible) updated value is available.
For having the result, the mBuffer could be analised, but better is to read the arrays or registers in the driver to read the data including offsets and scaling.
Example:
The Processor has 8 analogue inputs, from which the result comes in ram &H80 to &H8F.
To read this analogue values, the driver send a prcAnalogValues (= "rm 80 10") command into the pump every 100 mSec, meaning " Read Memory from &H80, number of values &H10,
returning RAM adress &H80 to &H8F (16 values, is 8 x 2 bytes) in the mBuffer.
The 4th set of bytes is from the MAIN external sensor, so address 86 lsb and 87 msb.
Value is from 00 to 3ff, but there is an offset, calibration and scale to convert it to degrees Celcius.
If a prcAnalogValues command was returned, in stead of analising the 16 returned values in the mBuffer, you could read the value gDrv.ADC(snMain, rsTemprature) , which gives the temperature of the main (external) sensor in degrees Celcius, which was just updated.
In this way the calibration and the scale of the used PT100 element is used in the result.

In the NorhofPumpDrive.frm is most of the commands needed for making an application.
If you run the complete NorhofPumpDrive.vbp, including the grafiekform.frm, you can see in the grafiekform how data from the first form can be used in a second form.
If you need to know any more commands, please let me know

Direct access by RS232 commands:

COMparameters = 19200,N,8,1
To communicate with the pump, there are 5 types of commands.
"i" terminated with CRLF returns info from this pump.
"rm" command reads RAM
"wm" command write in RAM
"re" command reads EEprom
"we" command write EEprom
commands are echoed back with "Ready", or "Wrong command"
command should be send as "rm 010 10", terminated with CRLF

re 010 = read eeprom &H10, 1 byte
re 010 2 = read eeprom &H10, 2 bytes
re 010 10 = read eeprom &H10, 16 bytes

rm 010 = read memory &H10, 1 byte
rm 010 2 = read memory &H10, 2 bytes
rm 010 10 = read memory &H10, 16 bytes

wm 238 3a = write memory &H238, 3a
all writing command must be sent double , so two times directly after each other before accepted.

from place &H235 to &H23F is not used, so here you could experiment to change the data.

rm 23a ' no idea what is there now, probaly F0 or 0D , because memeory is filled with F0 0D before intitialising.
wm 23a aa ' first time will not be eccepted
wm 23a aa ' second time will be accdepted if it is equal as the first time
rm 23a ' should give aa

To read the pressure:

Pressure is the value from the ADC ,

```
; AD results space 0x080 (ADCAVEST)
; ---------------------------------------------------------------------
.equ ADCVes = ADCAVEST +$0 ; Vesel sensor value pa0 (Low,High)
.equ ADCTMB = ADCAVEST +$2 ; TMB sensor value pa1 (Low,High)
.equ ADCEX1 = ADCAVEST +$4 ; External sensor 1 value pa2 (Low,High)
.equ ADCEX2 = ADCAVEST +$6 ; External sensor 2 value pa3 (Low,High)
.equ ADCPre = ADCAVEST +$8 ; Presure sensor value pa4 (Low,High)
.equ ADCEFL = ADCAVEST +$A ; External flow value pa5 (Low,High)
.equ ADCIFL = ADCAVEST +$C ; Internal flow value pa6 (Low,High)
.equ ADCTem = ADCAVEST +$E ; Int/ext temp. value pa7 (Low,High)
.equ ADCPow = ADCAVEST +$10; Power supply voltage (low,High)

.equ POWERtal = ADCAVEST +$12; Power vermen. factor byte

.equ MinSLEEP = ADCAVEST +$13; Sleep timer value (LSB)
.equ MinSLEEPH = ADCAVEST +$14; Sleep timer value (MSB)
```

so ADCAVEST +$8 = 088

"rm 088 2" will give 23 00 to 2A 00 when the pump is not in LN2, then the dewar pressure is zero mBar.
there is an offset which is in EEprom on

```
.org EELN2STRT ;$0010 ;cold offset, or low values
;                   Vess empt        TMB LN2        2nd sens
Eln2List: .DB 0xed, 0x00,     0x87, 0x00,     0x8f, 0x00
;                    Main sens        Vess room       TMB heat
Eln2List1: .DB 0x8f, 0x00,        0xa0, 0x01,     0x0f, 0x01
;                    Press low          precom2
Eln2List2: .DB 0x23, 0x00 ,       0x07 ,0x00
```

so "re 01c 2" will give the calibrated offset, probably 23 00 (or 24 00, or 25 00, or 2a 00)
(value can be different after calibration)

the actual pressure is "rm 088 2" minus "re 01c 2" times the scale
the scale is 5.42888 mBar per step

The external sensors are
.equ ADCEX1 = ADCAVEST +$4 ; External sensor 1 value pa2 (Low,High)
.equ ADCEX2 = ADCAVEST +$6 ; External sensor 2 value pa3 (Low,High)

NB in all code is SENSOR1 the 'extra' sensor, and SENSOR2 the 'main' sensor.

the offsets (calibration -196) and offsets is in table above,

the (calibration +30) warm values is:

```
; Default Warm Temp values
;-------------------------------------------------------------------------
.org EEWARMCAL ;$0090; calibratie if is +30 degrees
;                       Vess empt         TMB LN2         2nd sens
EWCalList: .DB   0x7D, 0x02,       0x50, 0x01,   0xDD, 0x01
;                       Main sens         Vess room       TMB heat
EWCalList1: .DB 0xDD, 0x01,      0x00, 0x00,      0x00, 0x00
;                        PressWa           Preco3
EWCalList2: .DB   0x00, 0x00,       0x00, 0x00
```

calibration values depend on how the sensor is used, on 33 mA current (selfheating, for level detection), or on 1 mA current (for temperature measurement)

the actual temperature of sensor 1 is rm 084 2 minus re 014 2
the actual temperature of sensor 2 is rm 086 2 minus re 016 2

this is in steps of the ADC.
The ADC steps is a more or less logaritmic scale corresponding to the PT100 characteristics (in the driver)

Calculating into degrees is done in the driver

Pumping mode SLEEP, or STANDBY, or PUMPING is in the pumpstatus register

```
; Pump status register (R25)
; ----------------------------------------------------------------------
.def PStatus = R25      ; Pump Status register
.equ rs_recv = 0        ;=1 ; 1 RS232 character received
.equ Pumping = 1        ;=2 ; 2 Pump request
.equ Valve_on = 2       ;=4 ; 4 Valve closed
.equ TMB_on = 3         ;=8 ; 8 TMB in heating mode
.equ Active = 4         ;=16 ; 16 Pump status : active
.equ Sleeep = 5         ;=32 ; 32 Pump status : sleep
.equ Warn_on = 6        ;=64 ; 64 Pump status : warning
.equ Alarm_on = 7       ;=128 ; 128 pump status : alarm
```

memory place is &H19 (decimal 25)
so "rm 19" gives this status , is 20 and 28 in sleep (sometimes the TMB is in heating mode in sleep)
when pumping, in active mode gives 1E and 16

never write in register 19

Better is to steer Pumping or Sleep with the RS232 registers or the pon/pof command.
Sending a "pon" command (PumpingON) will set the pump in STANDBY, sending a "pof" command (PumpingOFf) will set the pump to SLEEP.

The warning bit in the pumpstatus register is set when the dewar is almost empty (less than 5 liters left)

Also here you can read the status for the alarms.

If there is an alarm, the alarm register is

```
RMSTRT = 060
.equ Alarm = RMSTRT + 2      ; Alarm Flag register  (rm 62 1)
.equ HOT_AL_ON = 0           ; Vesel to warm (ROOM TEMP)
.equ VES_AL = 1              ; Vesel alarm flag (ALARM_CHECK)
.equ TMB_AL = 2              ; TMB alarm flag flag (ALARM_CHECK)
.equ EX1_AL = 3              ; EX1 alarm flag flag (ALARM_CHECK)
.equ EX2_AL = 4              ; EX2 alarm flag flag (ALARM_CHECK)
.equ FROZ_AL = 5             ; rise pipe frozen alarm (or press tube frozen)
; = 6                        ; container warmer than -140 (only model 605 pump)
equ  FILL_AL_MA =7           ;  filling too long
```

the alarm mask depends on in what mode is the pump, and how many sensors
there are present.

```
.equ AlarmMask = RMSTRT + 3 ; Alarm Mask Flag register (rm 63 1)
.equ VES_AL_RO = 0 ; Enable Vessel Room temp check
.equ VES_AL_MA = 1 ; Enable Vessel alarm check
.equ TMB_AL_MA = 2 ; Enable TMB alarm check
.equ EX1_AL_MA = 3 ; Enable EX1 alarm check
.equ EX2_AL_MA = 4 ; Enable EX2 alarm check
```

so in (rm 062)AND(rm 063) is >0 then there is an alarm.

If there is a FROZEN alarm (bit 5 mem 62), also alarmregister2 should be checked

```
.equ  Alarm2reg  = $219   ; second alarm register (rm 219 1)
   .equ  Froz1_ON       = 0      ; 1  frozen measuring tube 1
   .equ  Froz2_ON       = 1      ; 2  frozen measuring tube 2
   .equ  NoPress3_ON    = 2      ; 4  no pressure building
   .equ  Froz4_ON       = 3      ; 8  not used
   .equ  FrozInjec      = 4      ; 16 frozen injector (only model 611 pump)
```

FROZEN alarm (bit 5 mem 62) and mem219 bit 0 and 1 = 0, then it is rise pipe frozen alarm
FROZEN alarm (bit 5 mem 62) and mem219 bit 0 or 1 = 1, then it is frozen measuring tube

If there is a FILLTOOLONG alarm (bit 7 mem 62), also alarmregister2 should be checked

FILLTOOLONG alarm(bit 7 mem 62) and mem219 bit 2 = 0 then it is filling too long
FILLTOOLONG alarm(bit 7 mem 62) and mem219 bit 2 = 1 then it is no pressure building alarm

appendix 1

Level of LN2 in the dewar reading
For calculating the level, you need only two values.
The pressure of the pressure sensor when the pump is not pumping, corresponds to the level height in the dewar, because there is a tube going to the bottom of the dewar.
reading
.equ DewarlevelL = $CE ; LN2level in dewar (LSB)
.equ DewarlevelH = $CF ; (MSB)
gives the pressure in steps
reading
.equ PressCoL = $C8 ;&h0c8 0mBar pressure cal.point (LSB)
.equ PressCoH = $C9 ; (MSB)
gives the calibration for 0 mBar, so this should be subtracted from the first value
so:
dewarlevelLN2 = (Dewarlevel - PressCo) ' in steps
real level in cm = dewarlevelLN2 * stepsize / weight LN2 +0,8 cm
cmlevel! = CInt(dewarlevelLN2 * 0.542888 / 0.808 * 10 + 8) / 10
(the tube does not go to the bottom, but is ab. 8 mm from the bottom)

During pumping it is not possible to accurate updating, but we decrease this value a little depending on how much power the heater gives. But this is not realy accurate, but a guess.
But after pumping, the value is updated to the correct level again.

pressure in dewar reading

Pressure in the dewar (at the botom) is at &H88 and 89 so "rm 88 2"
Offset for 0 mBar is at &HC8 and C9
so "rm C8 2" (gives between 23 00 and 40 00)
Steps for the pressure is 5.42888 mBar
So pressure is ("rm 88 2" minus "rm C8 2") times 5.42888

Reading if PUMPING is busy

pumpstatus is in PumpStatus register. is in register &H19

```
.equ rs_recv = 0 ;=1 ; 1 RS232 character received
.equ Pumping = 1 ;=2 ; 2 Pump request
.equ Valve_on = 2 ;=4 ; 4 Valve closed
.equ TMB_on = 3 ;=8 ; 8 TMB in heating mode
.equ Active = 4 ;=16 ; 16 Pump status : active
.equ Sleeep = 5 ;=32 ; 32 Pump status : sleep
.equ Warn_on = 6 ;=64 ; 64 Pump status : warning almost empty dewar
.equ Alarm_on = 7 ;=128 ; 128 pump status : alarm, no LN2, or a sensor broken
```

So "rm 19" gives the status, in which bit 4 and 5 is the pumping status
bit 4 and 5 "01" is SLEEP
"10" is ACTIVE
If in ACTIVE, bit 1 defines PUMPING or ACTIVE
if bit 1 = "0" status is ACTIVE
if bit 1 = "1" status is PUMPING
In fact reading bit 1 is enough for knowing when it is pumping

Reading EXTRA external sensor

reading the temperature from the sensor (sensor "extra") is at address &H84 (and &H85 MSB)
so "rm 84 2" CRLF will give back two values, LSB and MSB .
Offset for -196 value for this sensor is at EEprom at &H14 and 15
so "re 14 2" CRLF (gives 89 00)
Calibration for +30 Celsius is in EEprom at &H94 and 95
so "re 94 2" CRLF (gives D6 01)
Low switchpoint of this sensor is at &HB8 and 9
so "rm B8 2" (gives 95 00 for -192) (AF 00 for -181)
High switchpoint is at &HBA and BB
so "rm BA 2" (gives BC 00)

The value of the sensor is "rm 84 2" which is the part of 5volt / 3ff as result of the PT100 characteristic connected with 235 ohm to +5 Volt. For correcting lead resistance and Voltage correction, the calibration for -196 and +30 degrees should be included. So the real temperature is not easy to calculate without the driver.

Switching to PUMPING in direct RS232 mode

For driving the #915 pump, there are 3 special registers in the pump to switch the pump from OFF to ACTIVE, to set adjusted temperature, and to set adjusted flowrate. (in mode E and F)

When used in mode 3, only the OFF and ACTIVE register is needed.

When using the driver PUMP913drv.exe as class module (in Visual Basic, or other language), calling SetContr with a 0 or 1 or 3 will set the pump Sleep or ACTIVE or pumping

```
Public Sub SetContr_Click() '
   If ContrSet = "" Then ContrSet = 0
   If gDrv.RS232Set(rscControl, ContrSet) = False Then
      MsgBox ("wrong value sent for pump control !!!")
   End If
End Sub
```

If you do not want to use the driver, this could be done by writing "0" or "1" or "3" in register &H114
so "wm 114 0" CRLF, will switch the pump OFF(=SLEEP)
"wm 114 1" crlf, will switch the pump ACTIVE,
"wm 114 3" crlf, will switch the pump to PUMPING,
with writing to RAM 110 and 111 you can set the temperature setpoint for mode E.
with writing to RAM 112 and 113 you can set the pumping pressure for mode E and F.

```
.equ RSTmpSetL = RSREG + $0        ;(&h110) RS232 temperature setpoint      (LSB)
 .equ RSTmpSetH = RSREG + $1       ;                                        (MSB)
 .equ RSFlwSetL        = RSREG + $2 ;(&h112) RS232 flow setpoint             (LSB)
 .equ RSFlwSetH        = RSREG + $3 ;                                        (MSB)
 .equ RSConSet= RSREG + $4 ;(&h114) RS232 pump on/off control
 .equ  RSCSACT         = 0          ;=1 RS232 control active bit
 .equ  RSCSPUMP        = 1          ;=2 RS232 control Pumping bit
 .equ  RSCSEXAL        = 2          ;=4 RS232 control Extra sensor alarm mask
 .equ  RSCSMAAL         = 3         ;=8 RS232 control Main sensor alarm mask
 .equ  RSCSEXLR        = 4          ;=10 RS232 control Extra sensor led red
 .equ  RSCSEXLG        = 5          ;=20 RS232 control Extra sensor led green
```

```
.equ  RSCSMALR     = 6          ;=40 RS232 control Main sensor led red
.equ  RSCSMALG     = 7          ;=80 RS232 control Main sensor led green
```

Hardware of the Pump

In the pump is a PCB with the following hardware:

- Supply voltage of 12 to 30 Volt, AC or DC, 4 to 2 Ampere
  We supply a 12 Volt AC transformer in most cases.
- Pump heater and solenoid valve both work on direct supply voltage (12 Volt default)
  Is supply voltage is higher, both the heater and solenoid have PWM to let it work on the higher voltage,
  Almost all other components work on 5 Volt
- 5 Volt regulator to create 5 Volt power supply
- Atmel Mega16 microcontroller. Working on 5 Volt with the external Xtal on 8.0 mHz..
- 2 internal (set) of PT100 sensors, the vessel sensor(s) and the TMB sensor. Work on 33 mA.
- 2 possible external sensors, mostly on 33 mA, but in some working modes on 1 mA.
  named MAIN sensor (sensor 2) and EXTRA sensor (sensor 1)
  (when set by jumpers on 1 mA, the signal is amplified by the opamp)
- The beeper
- 6 leds on the front of the PCB
- The pressure sensor (0-500 mBar) 0,5 to 4,5 Volt
- Mode select switch (16 positions) mode 0 to F (900 pumps only)
- On/Off pushbutton
- 2 Potmeters for adjusting temperature and pressure. (900 pumps only)
- Several possible input signals, digital and/or analogue 0-5 Volt
- Alarm and external heater outputs by optocouplers.

The firmware source code is written in assembler.
The ATmega16 processor has $45F (1119) RAM places, 512 bytes EEprom and $1fff (8191) Flash memory

The pump is connected to the PC with the RS232 interface.
The RS232 port works on 19800,N,8,1
All commands should be terminated with chr$13 and chr$10 (0D, 0A)

Remote ON and OFF

All the #600, #800 and #900 models can be remotely switched OFF and ON (= in SLEEP or STANDBY), in 4 ways:
1 With a 5 volt signal on pin 4 of the 25p subDconnector, the pump can be switched to SLEEP.
0 Volt = SLEEP, 5 Volt is no-operation.
With a 5 volt signal on pin 5 of the 25p subDconnector, the pump can be switched to STANDBY.
0 Volt = STANDBY, 5 Volt is no-operation.
Connecting these pins to ground (pin 17=18=19=20), is also enough to make the signals switch.
So two pushbuttons could do the job also. SLEEP overrides the STANDBY.
(pull up resistor is 1 kohm, with 100nF capacitor)

2 With the optocoupler signals (see below)

For programmers:
3 If NO monitor software is standby, so the serial port is free for your own application software, the serial port can be opened at 19200,N,8,1
is Baudrate 19200, No parity, 8 databits, 1 stopbit.
Sending a "pon" command (PumpingON) will set the pump in STANDBY, sending a "pof" command (PumpingOFf) will set the pump to SLEEP.
Sending "rm 19" will give back the pump status register. result and 2 = pumping, result and 16 = standby, result and 32 = sleep, result and 64 = dewar level lowar than 5 liters, result and 128 = alarm

4 If the monitorsoftware is running, so the port is occupied, there is a way to tell the monitor software to switch the pump. This is not a very gentle solution, but the only easy way.
The monitor software will check continuously in his working directory (c:\Program Files\NorhofPumpMonitor as default) for a (empty) file "PON.txt" or "POF.txt" (or "pon.txt" or "pof.txt") . If found, the pump will switch ON or OFF, and delete the file.
creating a "RM19.txt" file gives back a "RM19.dat" file with the pumpstatus result.