Additional Instructions

DAC Access

IMPORTANT
READ CAREFULLY BEFORE USE
KEEP FOR FUTURE REFERENCE

1 Introduction

The following document descripts the communication between DAC-Flex and another PLC. To communicate via this interface, it is necessary to change the machine-ID! Further details can be found in the following chapter.

2 Hardware preparation

Prepare communication between DAC control unit and PLC.

2.1 DAC Flex

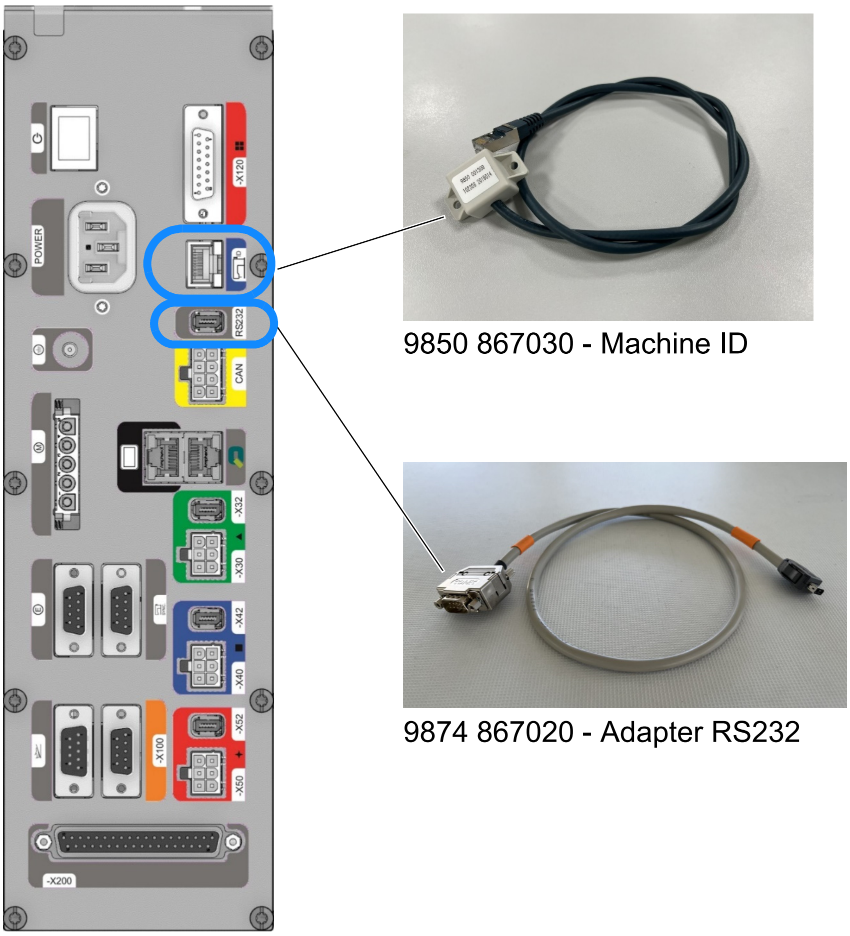
Machine-ID (9850 867030):

- To use DAC Access, you need a special machine-ID (See picture) to enable the serial interface for communication.

Adapter cabel (9874 867020):

- RS232(SUBD9) to Harting IX Typ A

Abb. 1: DAC Flex



2.2 DAC Comfort

2.2.1 Variation 1

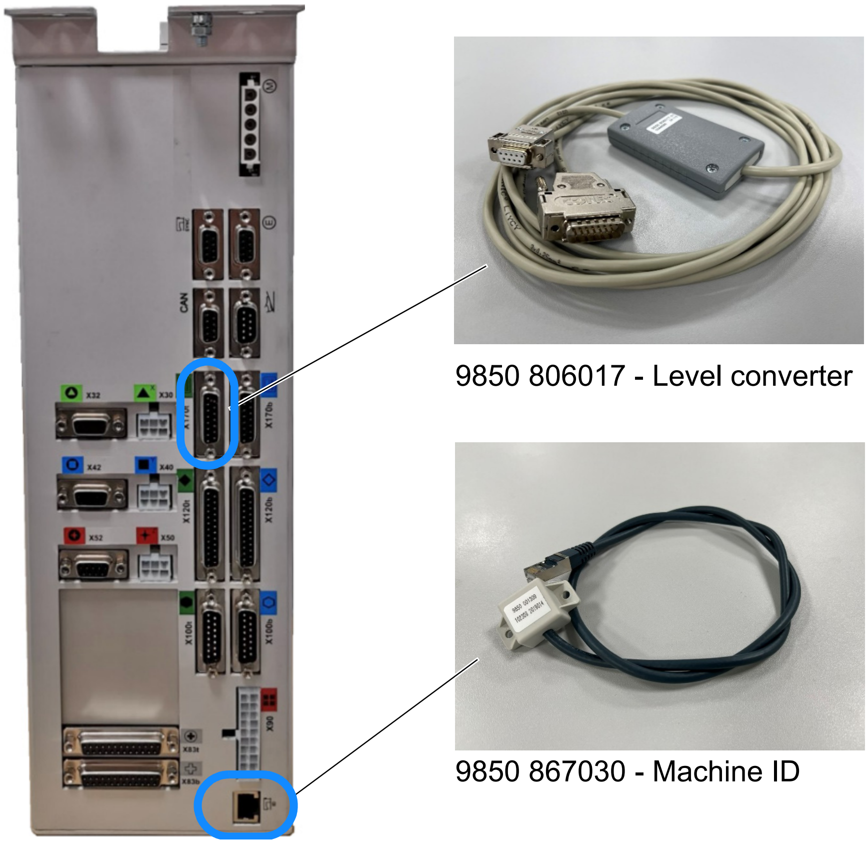
Machine-ID (9850 867030):

- To use DAC Access, you need a special machine-ID (see picture) to enable the serial interface for communication.

Adapter cable (9850 806017):

- Level converter RS232 (SUBD9) to RS422

Abb. 2: DAC Comfort (1)



2.2.2 Variation 2

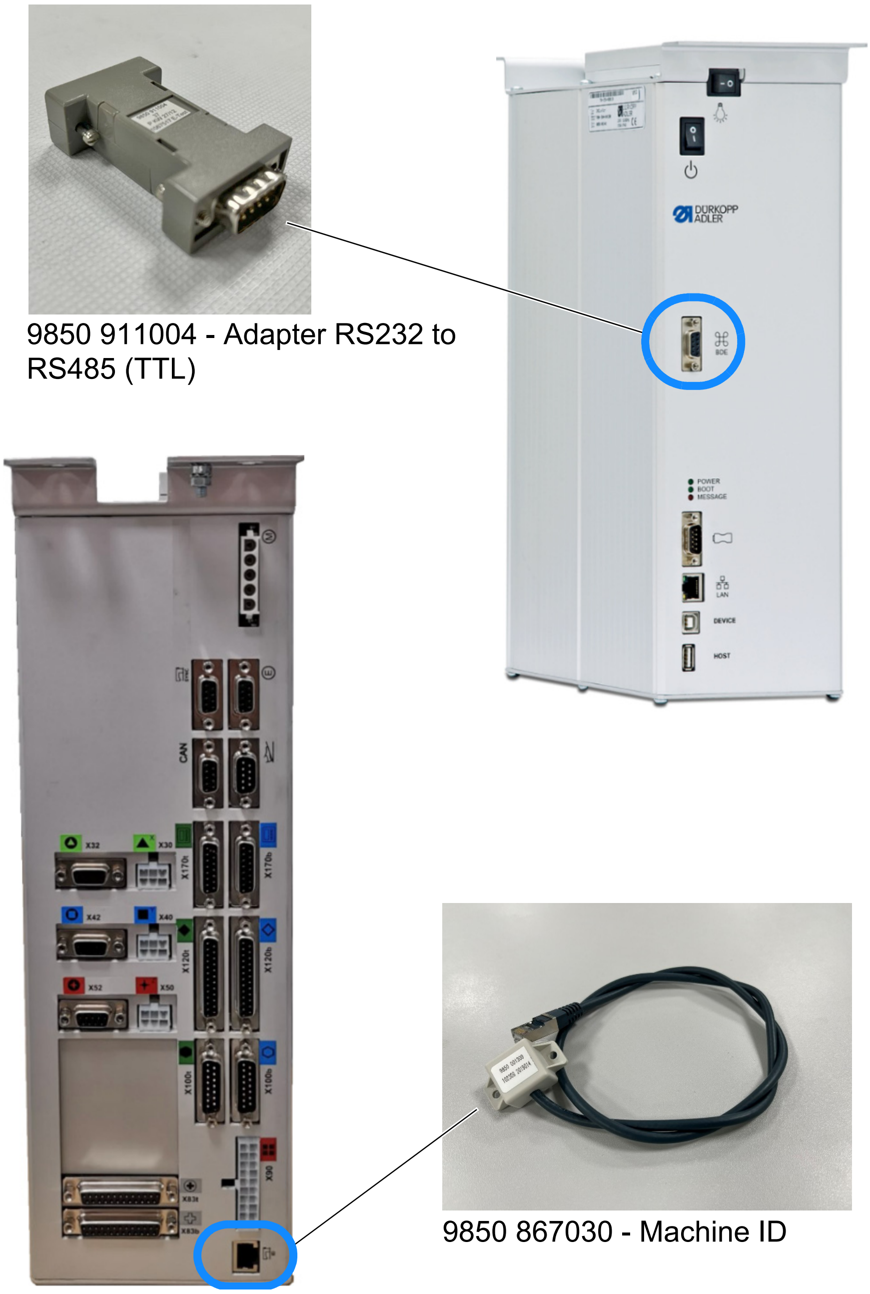
Machine-ID (9850 867030):

- To use DAC Access, you need a special machine-ID (see picture) to enable the serial interface for communication.

Adapter cable (9850 911004):

- Level converter RS232 (SUBD9) to RS485 (TTL)

Abb. 3: DAC Comfort (2)

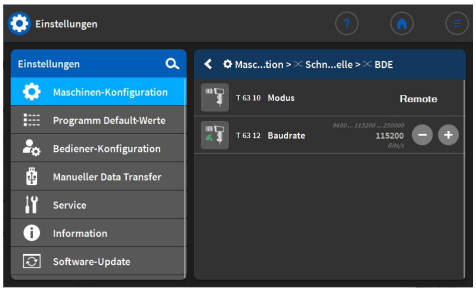


3 Establish Communication

To start communication, parameter T 63 10 must be set to Remote and the baud rate must be 115200 bit/s.

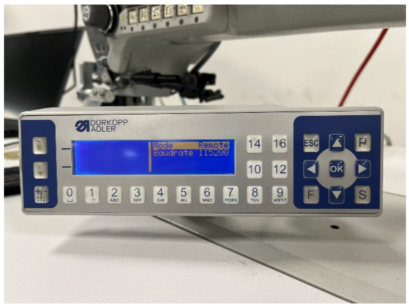
Abb. 4: Establish Communication (1)

OP8100/OP8300(Commander)



Path (Technician level):
Machine configuration/Interface/ BDE

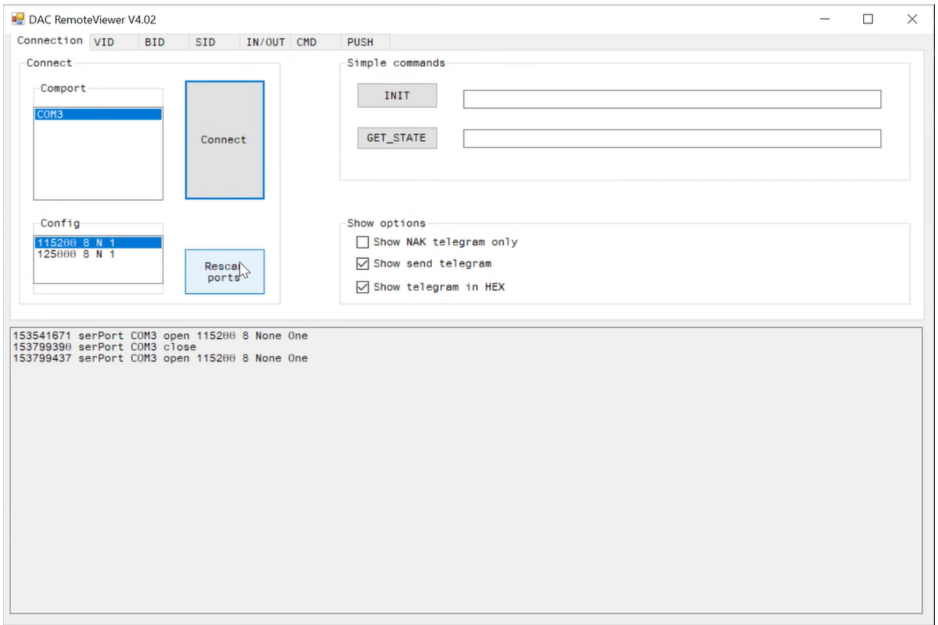
OP3000



Path (Technician level):
User config/Interface(X170t

Now DAC Access is available and commands to read or write parameter can be send. You can test the DAC Access Interface using the DAC RemoteViewer (see following pages):

Abb. 5: Establish Communication (2)



3.1 Telegram frame

All telegrams are send in the following format:

<SOT><CMD><PID><LEN> [<Data 0… N>] <CHK>

SOT	0x01 Start of transmission
CMD	Commando (see Commands)
PID	Packet ID PID of return frame must be equal the PID of request frame PID of two successive requests must be different
LEN	Length of data
Data0..N	Data if length != 0
CHK	Checksum. Sum all bytes from SOT until CHK

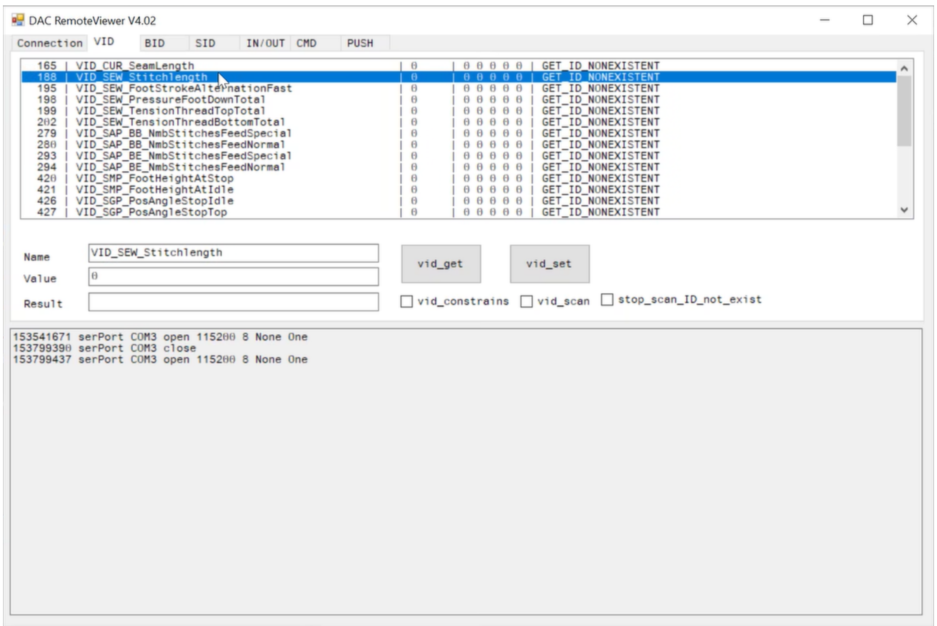
All of returns frames are in the same format, CMD = ACK or NAK and PID = PID of request telegram

Maximal length of telegram 256

Maximal length of data 256 – 6 = 250

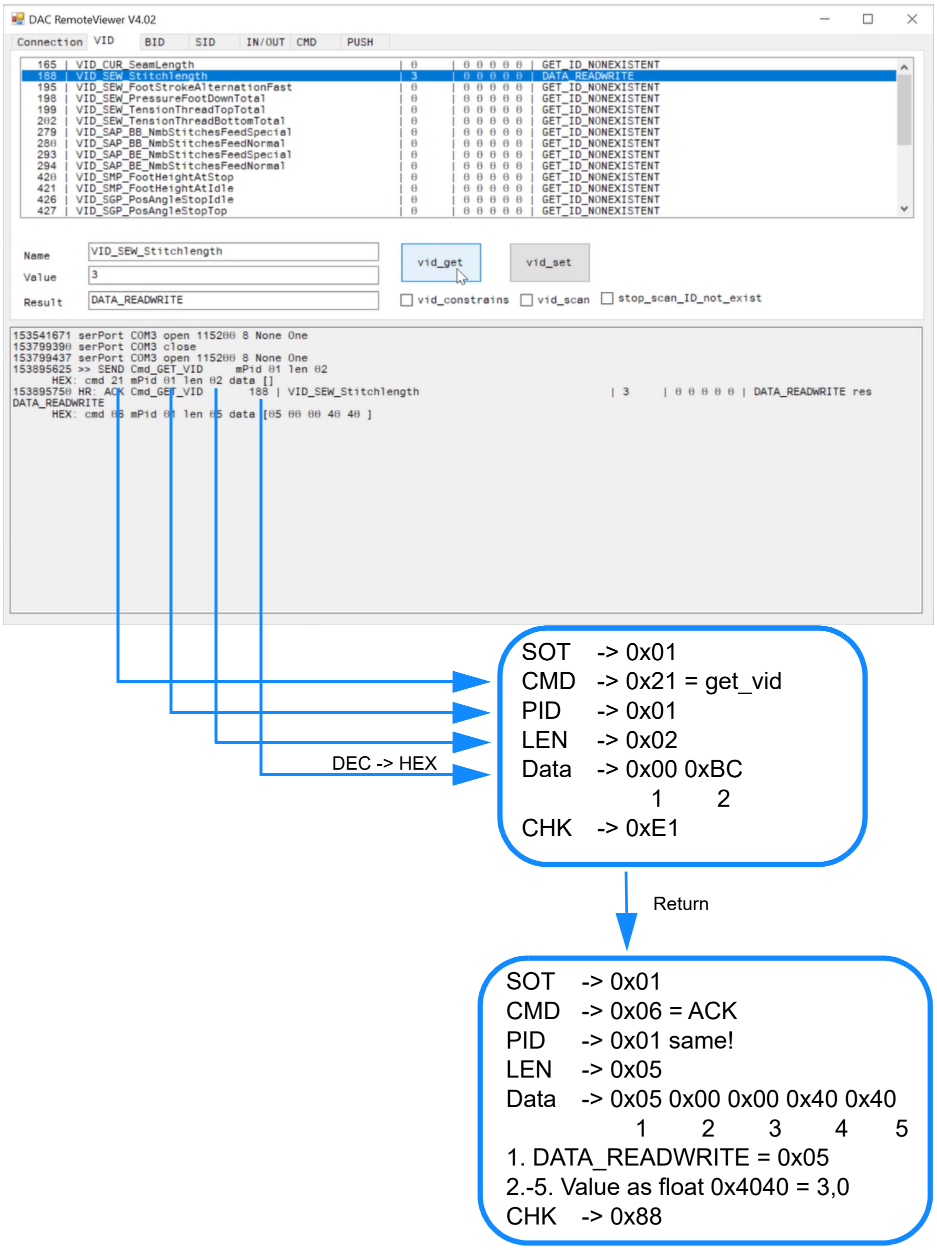
3.1.1 Example 1: Read and write Stitchlength parameter

Abb. 6: Example 1: Read and write Stitchlength parameter (1)



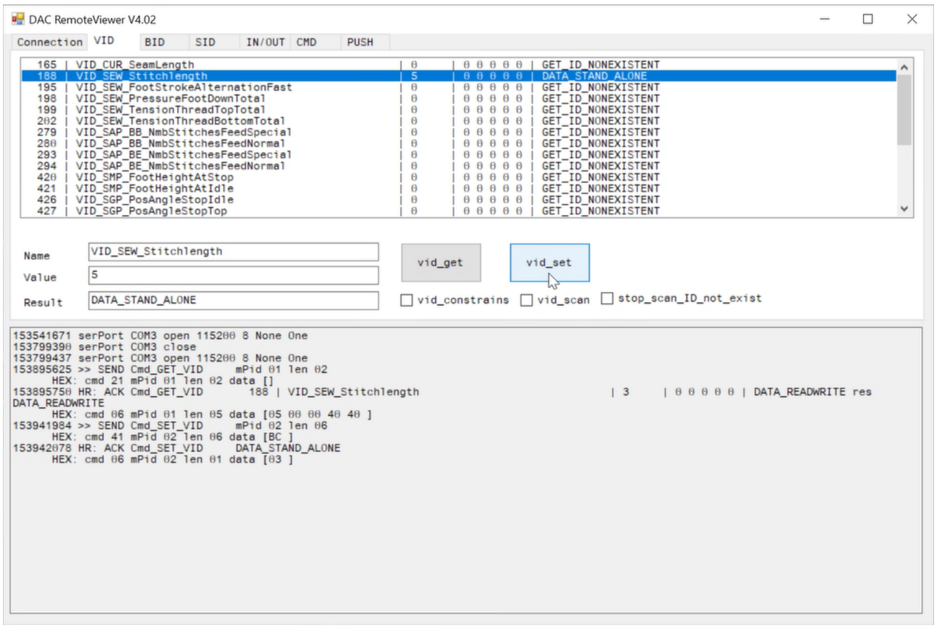
Select the VID parameter of the stitch length.

Abb. 7: Example 1: Read and write Stitchlength parameter (2)



Click on vid_get to read the current parameter from the control.

Abb. 8: Example 1: Read and write Stitchlength parameter (3)



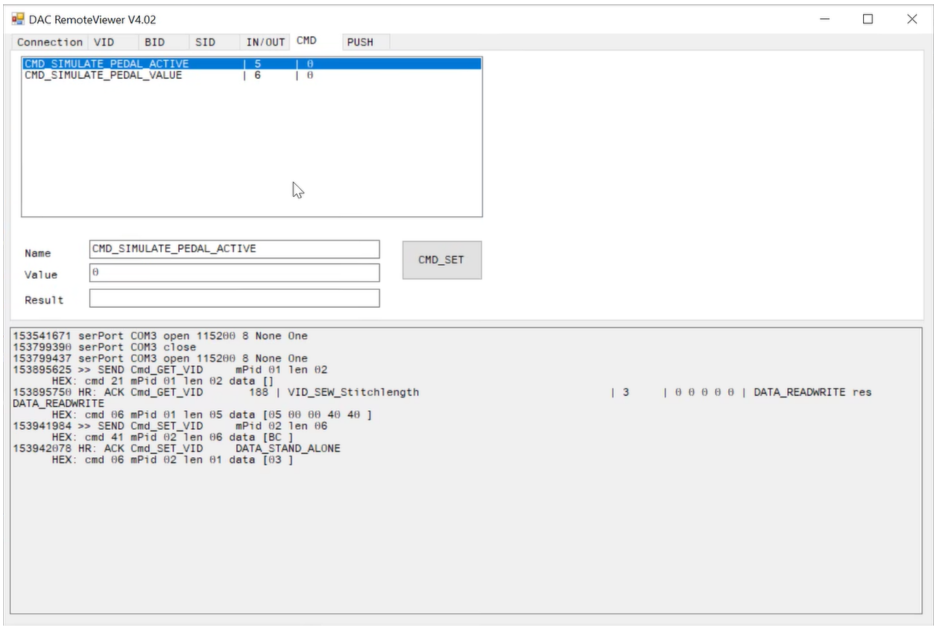
SOT -> 0x01
CMD -> 0x41 = set_vid
PID -> 0x02
LEN -> 0x06
Data -> 0x00 0xBC 0x00 0x00 0x40 0xA0
1 2 3 4 5 6
CHK -> 0xE6

SOT -> 0x01
CMD -> 0x06 = ACK
PID -> 0x02 same!
LEN -> 0x01
Data -> 0x03
1
1. DATA_STAND_ALONE = 0x03
CHK -> 0x0D

To write the VID parameter select it and type in the new value in for example 5 and click on vid_set.

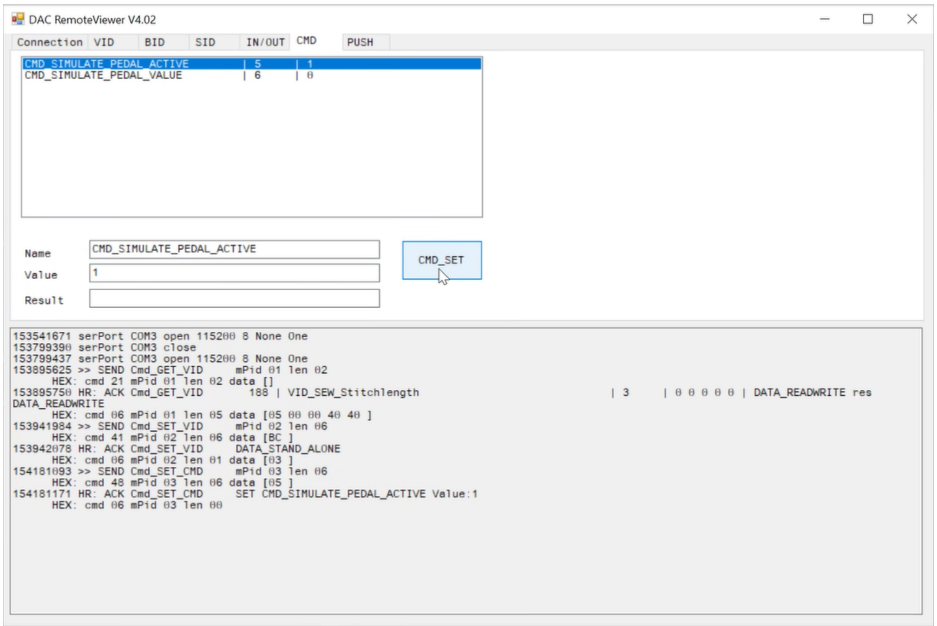
3.1.2 Example 2: Start/Stop sewing machine with simulated pedal value

Abb. 9: Example 2: Start/Stop sewing machine with simulated pedal value (1)



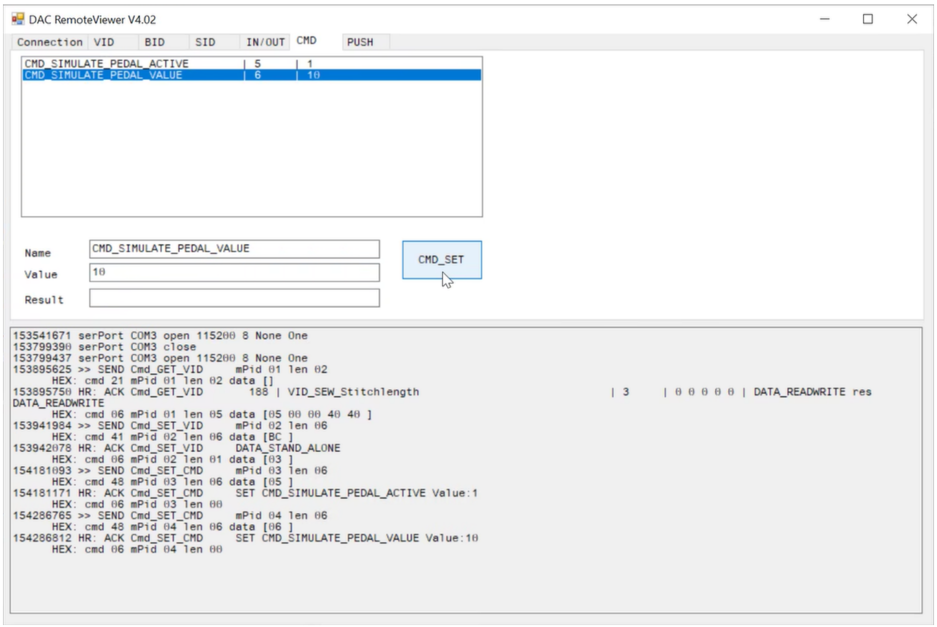
To activate the pedal simulation change the value of the CMD_SIMU-LATE_PEDAL_ACTIVE from zero to one.

Abb. 10: Example 2: Start/Stop sewing machine with simulated pedal value (2)



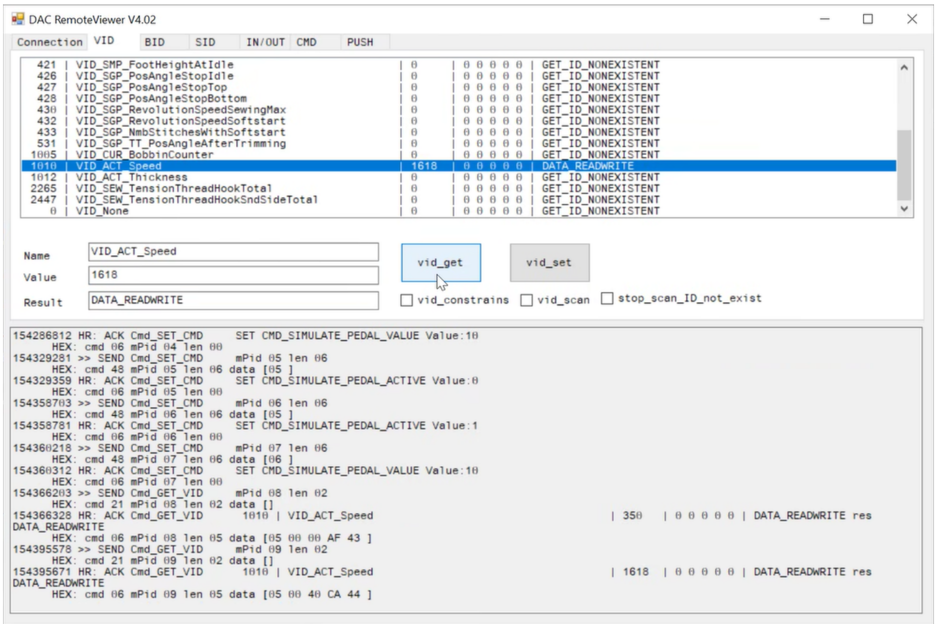
To start sewing change the CMD_SIMULATE_PEDAL_VALUE from zero to a higher value in our example it is ten.

Abb. 11: Example 2: Start/Stop sewing machine with simulated pedal value (3)



If you read now the current sewing speed of the sewing machine you see that it is sewing with $1618\frac{1}{min}$.

Abb. 12: Example 2: Start/Stop sewing machine with simulated pedal value (4)



4 DataTypes

FLOAT 32Bit IEEE single precision
UINT8 unsigned char (8Bit / 1Byte)
UINT16 unsigned char (8Bit / 1Byte) <LSB><MSB>
UINT32 unsigned long (32Bit / 4Byte) <LSBL><MSBL><LSBH><MSBH>
BOOL UINT8
EModelGetResult UINT8 =
typedef enum
{
GET_ID_NONEXISTENT = 0x00, ///< Daten-ID-Nummer nicht vorhanden (bzw. nicht angemeldet und damit nicht verfuegbar)
GET_ACCESS_DENIED = 0x01, ///< Zugriff auf das Objekt nicht erlaubt
DATA_PERM_ABSENT = 0x02, ///< Objekt ist vorhanden, aber während der Systemlaufzeit dauerhaft abwesend -> als Menüpunkt nicht sichtbar, als Kachel nicht konfigurierbar
DATA_TEMP_ABSENT = 0x03, ///< Objekt ist vorhanden, aber während der Systemlaufzeit temporaer abwesend -> als Menüpunkt nicht sichtbar, als Kachel konfigurierbar
DATA_NOT_VALID = 0x04, ///< Objekt ist vorhanden, aber nicht gültig -> als Menüpunkt nicht sichtbar
DATA_READONLY = 0x05, ///< Objekt ist gültig, aber schreibgeschützt -> als Menüpunkt nicht editierbar bzw. nicht betretbar
DATA_READWRITE = 0x06 ///< Objekt ist gültig und auch änderbar -> als Menüpunkt editierbar
}
EModelGetResult; (ModelResults.h)
EModelSetResult UINT8 = typedef enum
{
SET_ID_NONEXISTENT = 0x00, ///< Daten-ID-Nummer nicht vorhanden (bzw. nicht angemeldet und damit nicht verfügbar)
SET_ACCESS_DENIED = 0x01, ///< Zugriff auf das Objekt nicht erlaubt
SETTING_DENIED = 0x02, ///< Objekt war als exklusive Ressource bereits blockiert -> Setzen nicht ausgeführt
DATA_STAND_ALONE = 0x03, ///< Objekt ist unabhängig von anderen Objekten -> Menü muss nicht aktualisiert werden
DATA_INTERACTION = 0x04 ///< Objekt hat ggf. auch andere Objekte geändert -> Menü muss aktualisiert werden
}
EModelSetResult; (ModelResults.h)
PedalValue UINT8
OperationMode UINT8 = enum OperationMode
{
OM_STARTING = 0x00,
OM_REMOTE_IDLE = 0x01,
OM_REMOTE_ACTIVE = 0x02,
OM_SEWING_IDLE = 0x03,
OM_SEWING_ACTIVE = 0x04,
OM_PROGRAMMING = 0x05,

OM_TEACHING_IDLE = 0x06,
OM_TEACHING_ACTIVE = 0x07,
OM_TEACHING_FINISH = 0x08,
OM_SETTINGS = 0x09
}; (OperationAndSewingModes.h)
SewingModel UINT8 = enum SewingMode
{
SEM_MANUAL = 0x00,
SEM_PROGRAM = 0x01,
SEM_SEQUENCE = 0x02,
SEM_DEFAULT = 0x03
}; (OperationAndSewingModes.h)

5 Commands

HEX	Name	Description
0x01	INIT	Initialise of communication
0x06	ACK	Acknowledge
0x15	NAK	Not Acknowledge
0x16	PUSH	Push telegram
0x21	GET_VID	Get VID
0x22	GET_BID	Get BID
0x24	GET_DID	Get DID
0x25	GET_OUTPUT	Get output function state
0x26	GET_INPUT	Get input function state
0x27	GET_STATE	Get pedal state
0x28	GET_SID	Get SID
0x41	SET_VID	Set VID
0x42	SET_BID	Set BID
0x44	SET_DID	Set DID
0x45	SET_OUTPUT	Set output function state
0x46	SET_INPUT	Set input function state
0x47	SET_LOCK_INPUT	Set lock/unlock input function
0x48	SET_CMD	SET_CMD Set pedal lock for example
0x49	SET_SID	Set SID

5.1 INIT

Initialise the communication between DAC and PLC.
CMD = INIT
Data = NULL
Return = ACK with <Version of DAC-Interface. UINT16>

5.2 ACK

Acknowledge command: Operation successful, data include return value (see request description)

5.3 NAK

Not Acknowledge command: Operation error data include the error code (see request description)

5.4 Parameter types

There exist three parameter types (see DataIds.h file)

VID_xxx Number value Ids only numbers with min, max, standard, current value und …

BID_xxx Boolean values Ids only Boolean values

LID_xxx Value lists

DID_xxx Discrete values

SID_xxx String values

5.5 VID get set

5.5.1 Get VID current

Get current value

CMD = GET_VID

Data = <VID UINT16> (DataIDs.h)

Return = ACK with <EModelGetResult UINT8><current value as FLOAT>

5.5.2 Get VID with constrains

Get current value with constrains

CMD = GET_VID

Data = <VID UINT16><FLAG=1 UINT8 > (DataIDs.h)

Return = ACK with < EModelGetResult UINT8><CurrentVal FLOAT><MinVla FLOAT><MaxVal FLOAT><DefaultVal FLOAT><increment FLOAT><decimalPlaces UINT8>

5.5.3 Set VID

Set current value

Data = <VID UINT16><VAL as FLOAT> (DataIDs.h)

Return = ACK or NAK <EModelSetResult UINT8>

5.6 BID get set

5.6.1 Get BID

Get current boolean value

CMD = GET_BID

Data = <BID UINT16> (DataIDs.h)

Return = ACK with <EModelGetResult UINT8><current value UINT8>

5.6.2 Set BID

Set current value

Data = <BID UINT16><VAL as UINT8>

Return = ACK or NAK <EModelSetResult UINT8>

5.7 DID get set

5.7.1 Get DID selected

Get current value element

CMD = GET_DID

Data = <DID UINT16> (DataIDs.h)

Return = ACK with <EModelGetResult UINT8>< value as FLOAT >

5.7.2 Get DID elements enabled

CMD = GET_DID

Data = <DID UINT16><INDEX=FFFF UINT16>

Return = ACK with <NUMBER of did elements as UINT16><FLAGS UINT8 * (NUMBER+7)/8> <MinVal FLOAT><MaxVal FLOAT><DefaultVal FLOAT><increment FLOAT><decimalPlaces UINT8>

5.7.3 Get DID

Get did (discrete id) values

CMD = GET_DID

Data = <DID UINT16><value UINT16> (DataIDs.h)

Return = ACK with <EModelGetResult UINT8> <INDEX=selected element as UINT16>

5.7.4 Set DID

Set discrete id value

Data = <DID UINT16><VALUE FLOAT> (DataIDs.h)

Return = ACK or NAK <EModelSetResult UINT8>

5.8 SID get set

5.8.1 Get SID

Get current string value

CMD = GET_SID

Data = <SID UINT16> (DataIDs.h)

Return = ACK with < EModelGetResult UINT8><length UINT8><string-array char>

5.8.2 Set SID

Set current string value

Data = <SID UINT16><length as UINT8><string-array char>

Return = ACK or NAK <EModelSetResult UINT8>

5.9 CMD set

5.9.1 Set CMD

Set current value

CMD = SET_CMD

Data = < CMD UINT16><VAL1 as SINT32> (The number of values depends of the command) (CommandIDs.h)

Return = ACK or NAK (check CommandIDs.h)

5.10 Input functions

5.10.1 Get Input functions count

Get intput count

CMD = GET_INPUT

Data = NULL

Return = ACK with <MAX_INPUT_FUNCTIONS_IDS UINT8>

5.10.2 Get Input function value

Get input value

CMD = GET_INPUT

Data = <INPUT_FUNCTION_ID UINT8>

Return = ACK with <STATE UINT8>

Input functions

5.10.3 Set Input value

Set input value

CMD = SET_INPUT

Data = <INPUT_FUNCTION_ID UINT8><STATE UINT8>

Return = ACK or NAK if Blocked with <STATE UINT8>

5.10.4 Set lockage/unlock Input Function

Set input value

CMD = SET_LOCK_INPUT

Data = <INPUT_FUNCTION_ID UINT8><STATE UINT8>

Return = ACK

5.10.5 Get Output functions count

Get output count

CMD = GET_OUTPUT

Data = NULL

Return = ACK with <MAX_OUTPUT_ID UINT8>

5.10.6 Get Output value

Get output value

CMD = GET_OUTPUT

Data = <OUTPUT_FUNCTION_ID UINT8>

Return = ACK w ith <STATE UINT8>

5.10.7 Set Output value

Set output value

CMD = SET_OUTPUT

Data = <OUTPUT_FUNCTION_ID UINT8><STATE UINT8>

Return = ACK

5.11 Get current operation, state, pedal value and mode

Get current operation modes

CMD = GET_STATE

Data = NULL

Return = ACK <PedalValue UINT8> <OperationMode UINT8> <Sewing-Mode UINT8>

5.12 Push telegrams

Push telegrams are sent only from DAC to PLS

5.12.1 Telegram types

PT_MSG	0x01
PT_OUTPUT	0x02
PT_INPUT	0x03
PT_VAL	0x04
PT_CmdFinish	0x05

5.12.2 Messages error warning and information

CMD = PUSH

Data = <PUSH_TYPE=PT_MSG><MSG_ID UINT32><MSG_TYPE UINT8>

Return = no return necessary

5.12.3 Output function

CMD = PUSH

Data = <PUSH_TYPE=PT_OUTPUT><OUTPUT_FUNCTION_ID UINT8><State BOOL>

Return = no return necessary

5.12.4 Output function config enable/disable

CMD = PUSH

Data = <PUSH_TYPE=PT_CONFIG><PUSH_TYPE=PT_OUTPUT> <OUTPUT_FUNCTION_ID UINT8><Enable BOOL>

Return = ACK

5.12.5 Input function

CMD = PUSH

Data = <PUSH_TYPE=PT_INPUT><INPUT_FUNCTION_ID UINT8><State BOOL>

Return = no return necessary

5.12.6 Input function config enable/disable

CMD = PUSH

Data = <PUSH_TYPE=PT_CONFIG><PUSH_TYPE=PT_INPUT> <INPUT_FUNCTION_ID UINT8><Enable BOOL>

Return = ACK

5.12.7 Current values

CMD = PUSH

Data = <PUSH_TYPE=PT_VAL> <VALUE_ID UINT16><VALUE FLOAT>

Return = no return necessary

5.12.8 CurrentValues config enable/disable

CMD = PUSH

Data = <PUSH_TYPE=PT_CONFIG><PUSH_TYPE= PT_VAL > <VALUE_ID UINT16><Enable BOOL><minUpdateTime UINT16>

Return = ACK

5.12.9 Command Finish

CMD = PUSH

Data = <PUSH_TYPE= PT_CmdFinish ><COMMAND_ID UINT16>

DÜRKOPP ADLER GmbH
Potsdamer Str. 190
33719 Bielefeld
Germany
Phone: +49 (0) 521 925 00
E-Mail: service@duerkopp-adler.com
www.duerkopp-adler.com