

BACKGROUND

Modern VAG ECUs are quite complicated to code / adapt, sometimes many changes are needed to activate a single feature. In order to simplify the process and give a better opportunity to share the available codings, VCP features now the Scripter, a simple Pascal-Like script interpreter.

BASIC STRUCTURES AND SYNTAX

Scripts are based on Pascal, so the syntax is like in Pascal. The script has to be wrapped between begin / end statements. You can find many Pascal tutorials on the internet - Example:

<http://www.tutorialspoint.com/pascal>

EXAMPLE SCRIPT

```
BEGIN
    CONNECTTOECUTP20('17') ;
    IF ISCONNECTEDTOECU THEN
        BEGIN
            WRITELN('CONNECTED') ;
            WRITELN(GETECUNO) ;
            WRITELN(GETECUDESCRIPTION) ;
            WRITELN(DOREADAPK(3)) ;
            IF DOWRITEAPK(3,'100') THEN WRITELN('APK OK') ELSE WRITELN('APK NOK') ;
            WRITELN(DOREADAPK(3)) ;
        END
    ELSE
        BEGIN
            WRITELN('NOT CONNECTED') ;
        END ;
    CLOSECOMMUNICATION ;
END.
```

Explanations:

CONNECTTOECUTP20('17') – connect to ECU with adressword 17 using TP20 protocol

ISCONNECTEDTOECU – this built-in variable will be set to TRUE when connection was established and to FALSE when not.

WRITELN – just put some text on the “messages” box

GETECUNO – this function will return partnumber of the connected ECU as a string (=text)

GETECUDESCRIPTION – this function will return the description of the connected ECU (for example “Gateway MQB”)

DOWRITEAPK(3,'100') – writes value '100' to adaptation channel number 3

DOREADAPK(3) – reads the value stored in adaptation channel 3

CLOSECOMMUNICATION – closes communication.

FUNCTION REFERENCE

FUNCTION NAME	DESCRIPTION
procedure WriteLn(s: string);	Writes a string 's' to messages box
function AskUserForInput(const Prompt: string) : string ;	Displays a input window with configurable prompt. Returns user input as a string
function IsConnectedToEcu : boolean ;	Returns TRUE if connection to ECU was established, FALSE when there's no connection
procedure CloseCommunication ;	Closes communication with ECU
function ConnectToEcuTP20(aw : string) : boolean ;	Connects to ECU with adressword 'aw' using TP20 protocol
function ConnectToEcuUDS(aw : string) : boolean ;	Connects to ECU with adressword 'aw' using UDS protocol
function ConnectToEcuTP16(aw : string) : boolean ;	Connects to ECU with adressword 'aw' using TP16 protocol
function ConnectToEcuKLine(aw : string) : boolean ;	Connects to ECU with adressword 'aw' using KWP2000/KWP1281 protocol
function DoLogin(login : integer) : boolean ;	Performs a login procedure using login code "login". Returns TRUE if login was accepted, FALSE if not.
function DoReadAPK(channel : integer) :string ;	Reads contents of the adaptation channel "channel" as string. If return value is empty – reading error occurred
function DoWriteAPK(channel : integer ; value : string) : boolean ;	Writes "value" to adaptation channel "channel". Returns TRUE when write was successful, FALSE when not.
function ChangeDiagSession(session : integer) : boolean ;	Changes current diagsession to standard (session=0), engineer (session=1) or EndOfLine (session=2). Returns TRUE when change was successful, FALSE when not
function ReadDataByID(ID : integer) : string ;	Returns contents of the record (=adaptation ID in UDS) pointed by ID. If return value is empty – reading error occurred
function WriteDataByID(ID : integer ; DataToWrite : string) : Boolean	Writes to record pointed by ID hexstring "Datatowrite". Returns TRUE when write was successful, FALSE when not
function WriteDataByIDRaw(ID : integer ; DataToWrite : string) : Boolean	Writes to record pointed by ID hexstring "Datatowrite". DOES NOT upload TesterFingerprint before write. Returns TRUE when write was successful, FALSE when not
function ReadLongCoding : string	Returns long coding of the connected ECU. If return value is empty – reading error occurred
function WriteLongCoding(codestring : string) : Boolean	Writes Long coding "codestring" to connected ECU. Returns TRUE when write was successful, FALSE

	when not
function WriteShortCoding(coding : integer) : Boolean	Writes short coding “coding” to connected ECU. Returns TRUE when write was successful, FALSE when not
procedure ClearDTC	Clears all fault codes from connected ECU
function ReadMemory(addr,amount_to_read : integer ; alfid : integer) : string	Reads “amount_to_read” bytes of memory area starting at address “addr” using alfid “alfid”.Hexadecimal addresses should start with ‘\$’ (example \$C01). Maximum add_to_read is 8, if you want to read more bytes – you have to call this function multiple times (2 for 16 bytes, 3 for 24 bytes etc) , changing read address . Alfid is a UDS protocol element, standard value is ‘\$44’. Contact support if you need more information about ALFID.
function WriteMemory(addr: integer ; DataToWrite : string ; alfid : integer) : Boolean	Writes “DataToWrite” Hexstring to memory under address “addr” Maximum length of DataToWrite is 16 characters (8 bytes). Alfid is a UDS protocol element, standard value is ‘\$44’. Contact support if you need more information about ALFID. Returns TRUE when write was successful, FALSE when not
function ReadMemoryByAddr(addr,amount_to_read : integer ; alfid : integer) : string	Reads “amount_to_read” bytes of memory area starting at address “addr” using alfid “alfid” and direct read mode .Hexadecimal addresses should start with ‘\$’ (example \$C01). Maximum add_to_read is 8, if you want to read more bytes – you have to call this function multiple times (2 for 16 bytes, 3 for 24 bytes etc) , changing read address . Alfid is a UDS protocol element, standard value is ‘\$44’. Contact support if you need more information about ALFID.
function WriteMemoryByAddr(addr: integer ; DataToWrite : string ; alfid : integer) : Boolean	Writes “DataToWrite” Hexstring to memory address “addr” using direct write mode Maximum length of DataToWrite is 16 characters (8 bytes). Alfid is a UDS protocol element, standard value is ‘\$44’. Contact support if you need more information about ALFID. Returns TRUE when write was successful, FALSE when not
function GetECUShortCoding : string	Returns short coding of the connected ECU
Procedure SendRawUDSData(RawUDSCommand : string)	Sends RawUDSCommand directly to connected ECU using ISO-TP (UDS) protocol. Example: SendRawUDSData(‘310103D3040000’) starts routine \$03D3 with parameter 040000 .
function GetECUODXVersion : string	Returns required ODX container version of the connected ECU
function GetECUODXId : string	Returns required ODX container ID of the connected ECU
function GetECUSWVersionHEX : string	Returns Software version of the connected ECU
function GetECUDescription : string	Returns description of the connected ECU
function GetECUNo : string	Returns partnumber of the connected ECU
function SetBitInHexString(HexString : string ; ByteNo,BitNo : integer) : string;	Sets Bit BitNo in Hexstring’s Byte on position ByteNo. Return modified HexString. Bytes are counted from left-to-right starting from 1. Bits are counted 0-7. Example:

	SetBitInHexString('000000',1,0) returns 010000
function ClearBitInHexString(HexString : string ; ByteNo,BitNo : integer) : string;	Clears Bit BitNo in Hexstring's Byte on position ByteNo. Return modified HexString. Bytes are counted from left-to-right starting from 1. Bits are counted 0-7. Example: ClearBitInHexString('030000',1,0) returns 020000
function TestBitInHexString(HexString : string ; ByteNo,BitNo : integer) : boolean ;	Tests Bit BitNo in Hexstring's Byte on position ByteNo. Return true when bit is set and false when not . Bytes are counted from left-to-right starting from 1. Bits are counted 0-7
Function ChangeBytesInHexString(HexString,NewBytes : string ; ByteNo : integer) : string;	Replaces bytes at position ByteNo of Hexstring with NewBytes. Bytes are counted from left-to-right starting from 1. Example: ChangeBytesInHexString('000000','FF',2) returns 00FF00
function IntToBCD(input,digits : integer) : string ;	Return BinaryCodedDecimal representation of the 'input number. If result is shorter than 'digits', then zero-left-padding is added.. Example: intToBCD(1,4) results '0001'
function GetApplicationPath : string ;	Returns current VCP Installation path (eg. 'c:\vcp\')
function GetCurrentScriptPath : string ;	Returns current script location path (eg. 'c:\vcp\scripts\'.)
function ReadVIN : string ;	Reads VIN from car. Note: VIN is readed from Engine control unit.
procedure SaveStringToFile(stringtosave,filename : string) ;	Saves String "stringtosave" to text file. Filename should contain full path to file (eg. 'd:\testfiles\test.txt')
procedure DelayMilliseconds(amount : integer);	Stops script execution for "amount" milliseconds.
procedure ECUReset;	Performs ECU "terminal 15" reset. Note: works in UDS control units only
UploadZDC(filename,DataSetNameToUpload : string ; EraseFullMem : boolean) : string ;	Performs upload dataset "DataSetName" from ZDC File "Filename". Returns 'OK' when upload was successful or error string when not. Note: .zdc file must reside together with a script in the same directory. .ZDC files are bounded to key, you can not download .zdc using key A and then distribute it to system with key 'B'. In this case, user with key 'B' has to download needed zdc file separately and put it together with a script.

CRYPTED SCRIPTS

If you want to hide contents of the script, just save it as "Crypted script" (.CVCPScript). After opening such file, its contents will be hidden, without losing any functionality.

Note: crypted scripts can be only opened. They can not be edited or saved

EXAMPLES

LOGIN EXAMPLE

```
begin
ConnectToEcuTP20('19') ;
if IsConnectedToEcu then
begin
writeln('connected') ;
writeln(GetECUNo) ;
writeln(GetECUDescription) ;
if dologin(20103) then writeln ('login OK') else writeln ('login nok') ;
end
else
begin
writeln('Not connected') ;
end ;
closecommunication ;
end.
```

GETTING USER RESPONSE

```
var
userresponse : string ;
begin

userresponse:=AskUserForInput('Please give answer ...') ;
writeln(userresponse) ;

end.
```

CHANGING DIAGNOSE SESSION

```
begin
ConnectToEcuTP20('17') ;
if IsConnectedToEcu then
begin
writeln('connected') ;
writeln(GetECUNo) ;
writeln(GetECUDescription) ;
writeln(GetECUShortCoding) ;
// diagsession 0 - standard, 1 - engineer, 2 - eol
if ChangeDiagSession(1) then writeln ('change diagsession OK') else writeln ('change diagsession nok') ;
end
else
begin
writeln('Not connected') ;
end ;
closecommunication ;
end.
```

CHANGING ADAPTATION CHANNEL

```
begin
    ConnectToEcuTP20('17') ;
    if IsConnectedToEcu then
        begin
            writeln('connected') ;
            writeln(GetECUNo) ;
            writeln(GetECUDescription) ;
            writeln(DoReadAPK(3)) ;
            if dowriteapk(3,'100') then writeln('apk ok') else writeln('apk nok') ;
            writeln(DoReadAPK(3)) ;
        end
    else
        begin
            writeln('Not connected') ;
        end ;
    closecommunication ;
end.
```

CHANGING SHORT CODING

```
begin
    ConnectToEcuTP20('17') ;
    if IsConnectedToEcu then
        begin
            writeln('connected') ;
            writeln(GetECUNo) ;
            writeln(GetECUDescription) ;
            writeln(GetECUShortCoding) ;
            if WriteShortCoding(262141) then writeln ('coding OK') else writeln ('coding nok') ;
            writeln(GetECUShortCoding) ;
        end
    else
        begin
            writeln('Not connected') ;
        end ;
    closecommunication ;
end.
```

READING / WRITING MEMORY

```
var
readedmem : string ;
begin
ConnectToEcuTP20('09') ;
if IsConnectedToEcu then
begin
writeln('connected') ;
writeln(GetECUNo) ;
writeln(GetECUDescription) ;
if dologin(42013) then writeln ('login OK') else writeln ('login nok') ;
if ChangeDiagSession(1) then writeln ('change diagsession OK') else writeln ('change diagsession nok') ;
readedmem:=ReadMemory($C01,4,0) ;
writeln(readedmem) ;
if writememory($C01,'40C16233',0) then writeln ('writemem OK') else writeln ('writemem nok') ;
readedmem:=ReadMemory($C01,4,0) ;
writeln(readedmem) ;

end
else
begin
writeln('Not connected') ;
end ;
closecommunication ;
end.
```

CHANGING LONG CODING

```
begin
ConnectToEcuUDS('19') ;
if IsConnectedToEcu then
begin
writeln('connected') ;
writeln(GetECUNo) ;
writeln(GetECUDescription) ;
writeln(ReadLongCoding) ;
if writelongcoding('000000340B087300CF00025A1C0F000100010000000000000000000000000001') then writeln
('Coding OK') else writeln ('Coding nok') ;
writeln(ReadLongCoding) ;
if changediagsession('02') then writeln ('change diagsession OK') else writeln ('change diagsession nok') ;
end
else
begin
writeln('Not connected') ;
end ;
closecommunication ;
end.
```

SETTING AND CLEARING BITS IN HEXSTRING

```
var
teststring : string ;
begin
teststring:='01020408' ;
if TestBitInHexString(teststring,4,3) then writeln('ok') ; // returns true when bit 3 in hexstring's byte 4 is set.
teststring:=SetBitInHexString(teststring,1,2) ; // sets bit 2 in hexstring's byte 1
writeln(teststring) ;
teststring:=ClearBitInHexString(teststring,1,2) ; // clears bit 2 in hexstring's byte 1
writeln(teststring) ;
end.
```

UPLOADING THE ZDC FILE

```
begin
if ConnectToEcuUDS('55') then
begin
// uploads file 'passatcc_5m0_xenon.zdc', dataset name 'PASSAT_CC_5M0907357F_AFS_DATA'
// with eraseFullMemory on
// prints 'OK' if upload was successful
writeln(uploadzdc('passatcc_5m0_xenon.zdc','PASSAT_CC_5M0907357F_AFS_DATA',true));
closecommunication ;
end ;
end.
```

ALL TOGETHER. EXAMPLE CHANGES TAILLIGHTS CONFIGURATION TABLE FOR GOLF 7

```
const
// golf rdw LED
addr100a='250000410200E4000000000000000000250000420400E400000000000000002608001602006400
0000000000000000261800180400640000000000000000002210003918006408090A0000000000002220003A1
80064080A0A0000000000000';
addr170a='220000001800640600803E000A09000A220000001800640700803E000A0A000A' ;
addr1c0a='090B00311900D6000000000000000000091B00321900D6000000000000000000';
addr1a0a='010C002E1B00BF000000000000000000201A002A08008A000000000000000000';

// golf rdw bulb
addr100b='0000000000000000000000000000000000000000000000000000000000000000C08001602006400
0000000000000000C180018040064000000000000000000D10003918006408091C000000000000D20003A
180064080A1C000000000000';
addr170b='090A002908009C0000000000000000091A002A08009C000000000000000000';
addr1c0b='090B00311900E40000000000000000091B00321900E4000000000000000000';
addr1a0b='090C002E1B00E4000000000000000000000000000000000000000000000000';
```

```
var
i : integer ;
readeddata : string ;
readedcoding : string ;
addr100,addr170,addr1c0,addr1a0 : string ;
userinp : string ;

begin

userinp:=AskUserForInput('LEDs = 1 ; Bulbs = 2') ;
if (userinp<>'1') and (userinp<>'2') then
begin
writeln('Wrong input !') ;
exit ;
end ;

if userinp='1' then
begin
addr100:=addr100a ;
addr170:=addr170a ;
addr1c0:=addr1c0a ;
addr1a0:=addr1a0a ;
end
else
begin
addr100:=addr100b ;
addr170:=addr170b ;
addr1c0:=addr1c0b ;
addr1a0:=addr1a0b ;
end ;

if ConnectToEcuUDS('09') then
begin
if DoLogin(31347) then
begin
readeddata:=ReadDataByID($0D0B) ;
readedcoding:=ReadDataByID($0600) ;
writeln('Coding:' + readedcoding) ;
readedcoding:=ClearBitInHexString(readedcoding,7,7) ;
writeln('CodingN:' + readedcoding) ;

writeln('CNF:' + readeddata) ;

for i:=1 to 192 do
readeddata[$100*2+i]:=addr100[i] ;

for i:=1 to 64 do
readeddata[$170*2+i]:=addr170[i] ;

for i:=1 to 64 do
readeddata[$1c0*2+i]:=addr1c0[i] ;
```

```
for i:=1 to 64 do
  readedata[$1a0*2+i]:=addr1a0[i] ;

  writeln('CNF1:' + readedata) ;

  WriteDataByID($F198,readdatabyid($f198)) ;
  if WriteDataByID($0D0B,readedata) then writeln ('CNF OK') else writeln('CNF Error') ;
  if WriteDataByID($0600,readedcoding) then writeln ('Coding OK') else writeln('Coding Error') ;

end ;
CloseCommunication ;
end
else
writeln('comm error !') ;
end.
```

Have FUN !