FakeMODBUS

Televis Compact Driver

HOWTO

Sommario

1 Televis Compact Modbus Driver Example

This document describes an example of how to compile the P04354 excel model to create a device driver to be used with the new Televis**Compact**.
The file P04354_FakeModbus_v1_1.xls is supplied as reference for the following paragraphs, unless otherwise specified. The following chapters will refer to an example of technical specification summarized in Cap 1.1.

1.1 *Technical Specification for device FakeMODBUS*

Let' s assume to have a fake modbus controllers family described by the following technical specification and composed by 3 different models identified as **FakeMB mini**, **FakeMB midi**, **FakeMB maxi**

1.1.1 Controller resources accessed using MODBUS function H03 and H04

Function	Address	WORDS	Measure	Unit	Format	FakeMB mini	FakeMB midi	FakeMB maxi
H04	H01FF	2	Phase voltage	V/100	Unsigned long	X	X	X
H04	H0201	2	Analogue input 1	C°	Signed long	X	X	X
H04	H0203	2	Analogue input 2	C°	Signed long		X	X
H03	H0211	1	Digital input 1	Bool	Unsigned int	X	X	X
H03	H0211	1	Digital input 2	Bool	Unsigned int			
H03	H0205	2	High temperature1	C°	Signed long	X	X	X
H03	H0205	2	Low temperature 1	C°	Signed long	X	X	X
H03	H0209	1	Error probe 1	Bool	Unsigned int	X	X	X
H03	H0207	2	High temperature 2	C°	Signed long			
H03	H0207	2	Low temperature 2	C°	Signed long			
H03	H0212	1	Digital output 1	Bool	Unsigned int	X	X	X
H03	H0212	1	Light	Bool	Unsigned int	X	X	X

1.1.2 Controller commands to be used with MODBUS function H06

Address	WORDS	Command	Value	Format	FakeMB mini	FakeMB midi	FakeMB maxi
H02A	1	Reset alarms	0	Unsigned int	X	X	X
H02A	1	Light ON	1	Unsigned int	X	X	X
H02A	1	Light OFF	2	Unsigned int	X	X	X
H02B	1	Reboot	-1	Signed int	X	X	X

1.1.3 Controller parameters accessed using MODBUS function H03 and H06

Address	Label	Min	Max	Default	WORDS	FakeMB mini	FakeMB midi	FakeMB maxi
H030	HA1	-100	100	100	1	X	X	X
H032	LA1	-100	100	0	1	X	X	X
H034	HA2	-100	100	100	1		X	X
H036	LA2	-100	100	0	1		X	X
H040	NUM_ANAG_IN	1	3	1	1	X	X	X
J042	NUM_DIG_IN	1	3	1	1	X	X	X

1.1.4 MODBUS function H11 format

It is described the way the controller answers to the H11 function. The answer is specific for each Modbus controller.

Master query:

Byte 1	Byte 2	Byte 3	Byte4
Slave address	**Function H11**	**MSB CRC**	**LSB CRC**

Slave response:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
Slave address	**Function H11**	**Byte number**	**ID device**	**Data**	**Data**	**Data**	**MSB CRC**	**LSB CRC**

Where the ID device depends upon the device model and it will be:

• 121	for the FakeMB mini
• 130	for the FakeMB midi
• 170	for the FakeMB maxi

And the Data field will be always 0.

1.2 Communication settings for MODBUS device

The settings for the serial connection and the number of MODBUS address to check during network discover are listed in the P0435 excel file at the **Legenda** sheet .

In this page an important cell is the **Network Address Filter**: MODBUS devices can have an address varying between 1 and 255, the value of this cell can reduce this range, thus reducing the time of the network discover.

E.G.: assuming that the MODBUS device will never be installed with an address higher than 20, it's possible to fill this field with the range 1-20, thus avoiding unnecessary scanning of subsequent 235 address and reducing the time to scan the network of MODBUS instruments.

If not specified the fields **Baud Rate** and **Parity** have as default value 9600, even.

The field **Filename** defines the name of the binary file containing the driver you are going to produce, it is important that this name is different from the name of any other driver currently in use or still present in your TelevisCompact system, whether provided by Eliwell or produced by the user.

It's also possible to provide some advanced parameters such as the Device Timeout and the dimension of the buffer used by the Transmitter (**Device buffer TX(bytes)**) and by the Receiver (**Device buffer RX(bytes)**). These settings may be useful to achieve an higher acquisition speed, but a bad value may cause a not working driver. <u>Please pay attention that this can speed up the acquisition if you have several device of this kind in your network, if you have just one the improvement could be negligible. So if you are not interested in improving the performance or you don't know which are the values for your device, leave the default values.</u>

1.3 MODBUS device identification

The first thing to set up in a driver is the way the device is recognized by the Televis Compact. This can be done finding out a request or a series of requests that return data that are unique to this device. These requests can be the identification requests (e.g. MODBUS commands H11 or H2B) or readings of memory register (e.g MODBUS commands H03 or H04).

The process is composed by the following items:

- A request or series of requests to the device. These requests must be added to the **Client** page as a Support resource.
- A sequence of expressions (it could be just one) that must be evaluated using as input the reply of the device to the identification request. To state that the device is recognized, the last expression must be return a value which is not 0 (LASTVALUE must be not 0). Besides that, at least one expression must contain a **SET** operator where the **MODELCODE** variable is set. The **MODELCODE** set this way will be matched with the **POLI** values in the P0435 **Models** page, if a match exists then the device is successfully detected and the **NAME** corresponding to the given **POLI** will be assigned to the device.

- A list of **POLI,NAME** couple. These must be added to the **Models** page.

Let's see two examples in the next sections. The first one refers to the sample driver that we are building in this "how to" document. The second one does not refers to the driver we are building, it refers to a technique that is used when there is no identification request available or when the device is a programmable device (e.g. an Eliwell XT Pro).

1.3.1 Identification with MODBUS function H11

The identification of the instrument is performed with the MODBUS function H11 that, if implemented in the firmware, returns a unique ID number.

Note that while the request is uniquely encoded by the MODBUS protocol, the format and size of the response data field varies from manufacturer to manufacturer (refer to the controller specification)

E.G.: knowing that the response return 4 bytes of data where the first identifies the ID of the device, we will have the following steps:

Master query:

Slave address	**08h**
Function	**11h**
MSB CRC	**C6h**
LSB CRC	**7Ch**

Slave response:

Slave address	**08h**
Function	**11h**
Byte number	**04h**
Data 1 (ID)	**82h**
Data 2	**00h**
Data 3	**00h**
Data 4	**00h**
MSB CRC	**42h**
LSB CRC	**B9h**

This command is encoded in the **Client** page using an expression of type **RAW** in the field **MODBUS_ADDRESS** as follows:

RAW({MTA,H11,MCRC2},{MTA,H11,H04,XD(0),XI(3),MCRC2})

Pay attention to the usage of the **XD(0)** operator. **XD** is used to extract data from the reply and here it extracts the 4th byte, which is the one who carries the information about the model.

Hereafter how the **Client** page appears:

COLUMN	VALUE
INDEX	Unique number
LABEL	Ident
MODBUS_COMMAND	C
MODBUS_ADDRESS	RAW({MTA,H11,MCRC2},{MTA,H11,H04,XD(0),XI(3),MCRC2})
FILTER	HFF
EXISTENCE	CONST(0)
TYPE	Support

In the **DataAcquisitionInit** page the field access described in the **Client** page is referenced by using the "**Client(…)**" operator applied to the field access **LABEL** (which is "**ident**").

The value retrieved with the field access is then associated to the **MODELCODE** with the command:

SET(MODELCODE)=Client(ident)

Hereafter how the **DataAcquisitionInit** page appears:

COLUMN	VALUE
FUNCTION	IDENTIFICATION
PROTOCOL	ANY
0	SET(MODELCODE)=Client(ident)

The **MODELCODE** is uniquely associated with the name of the model trough the field **POLI** in the **Models** page, which is a reference to the **ParamsDefaults** page.

Note that the fields **PARAM_MANAGER_NAME** and **PARAM_MANAGER_NAME_MODBUS** in the **Models** page cannot contain spaces, unlike the field **MODEL_NAME**.

1.3.2 Identification, alternative method

This example does not refer to the P04354_FakeMODBUS.xls.

This technique should be used when the device does not support an identification command. This happens quite often and it is a common situation when a programmable device is in use.

In this situation, the driver developer must identify a register that holds a value that is typical to the devices. It can be a firmware version, revision or, when a programmable device is in use, a custom value that the developer of the application that runs on the programmable device has included with the only purpose of making the combination of device+application identifiable.

E.g. An application was developed for an XTPro device. The XTPro application holds the value 113 in the register at address H0200, that can be read with the H03 command.

To identify this device the reading of the register must be added to the **Client** page as follows:

COLUMN	VALUE
INDEX	Unique number
LABEL	Ident
FORMAT	
MEASUREMENT_UNIT	CONST(VM00020)
MODBUS_COMMAND	S
MODBUS_ADDRESS	H0200
FILTER	HFFFF
EXISTENCE	CONST(0)
TYPE	Support

Afterwards, the data gathered by this request must be processed in the **DataAcquisitionInit** page as follows:

COLUMN	VALUE
FUNCTION	IDENTIFICATION
PROTOCOL	ANY
0	SET(MODELCODE)=Client(ident)

The expression provided just takes the value gathered by reading the H0200 register and writes it in the **MODELCODE** variable.

Please pay attention to the following fact. The identification process requires that the result of the last expression is not 0. When a **SET** operator is used, the result of the expression is the value used to set the variable. So in most cases (and in this one too) we are lucky, because the value returned by the request is not 0, but if it would have been 0, the page would have been changed as follows:

COLUMN	VALUE
FUNCTION	IDENTIFICATION
PROTOCOL	ANY
0	SET(MODELCODE)=Client(ident)
1	GET(MODELCODE)=FIX(0)

Here, the first expression sets the **MODELCODE**, the second one compares it to 0. If the value is 0, the result of comparison is TRUE, which is automatically converted to 1 (FALSE would have been converted to 0).

Anyhow, it is deprecated and highly discouraged to choose a 0 value for identification. This is because 0 it's a value that it is commonly stored in registers of every device and this can lead to a wrong identification of devices.

Let's back to the example of the XTPro device with 113 stored in the H0200. After the expression is evaluated, **MODELCODE** will be set to 113. The **Models** page is compiled as follows:

COLUMN	VALUE

POLI	**113**
MODEL_NAME	**XTPro Application**

Here the 113 value has a match in the **POLI** column, so the device is successfully detected and its name will be "XTPro Application".

1.3.3 Identification with MODBUS function H2B

This is a quite complex identification method that usually require more requests. If you decide to use this method use the same approach described for command H11.

Alarm reading without existence condition

The reading of a single alarm is achieved by compiling a single row in the P0435 **Client** page.

E.G.: if you want read the status of the alarm **High Threshold AI1** and knowing that is indicated by the value of the first bit of the LSB of the register at the address 0205h, the required fields on the row in the **Client** page are:

COLUMN	VALUE
INDEX	Unique number
LABEL	AlmH1
DESCRIPTION	High Threshold AI1
DESCRIPTION_CODE	ALM00177
PROGRESSIVE	1
MEASUREMENT_UNIT	CONST(VM00021)
MODBUS_COMMAND	S
MODBUS_ADDRESS	H0205
FILTER	H0001
EXISTENCE	CONST(1)
TYPE	Alarm

Note that the correct values for the fields **DESCRIPTION_CODE** and **MEASUREMENT_UNIT** can be derived directly from the Televis Compact dictionary file of the desired language.

The dictionary files are on board of the Televis Compact and they can be downloaded using a Web browser. Valid dictionary links are

http://compactip/bin/Invenys.Dictionaries/Dictionary.de-DE.txt	GERMAN
http://compactip/bin/Invenys.Dictionaries/Dictionary.en-GB.txt	ENGLISH
http://compactip/bin/Invenys.Dictionaries/Dictionary.es-ES.txt	SPANISH
http://compactip/bin/Invenys.Dictionaries/Dictionary.fr-FR.txt	FRENCH
http://compactip/bin/Invenys.Dictionaries/Dictionary.it-IT.txt	ITALIAN

where compactip is the ip address of your Televis Compact. More links will be available when more languages will be supported.

In case of different alarms with the same **DESCRIPTION_CODE**, it is necessary to identify each resource uniquely using the **PROGESSIVE** field.

So if we had multiple alarms with the same **DESCRIPTION_CODE** value, each row would have over the field **MODBUS_ADDRESS** and **FILTER** different, also the field **PROGRESSIVE** with a value ranging from 1 to n, where n is the number of alarms.

1.4 Analog input reading without existence condition

The reading of an analog input without existence condition is achieved by compiling a single row in the P0435 **Client** page.

E.G.: if you want read the value of the analog input **Phase Voltage** and knowing that this value is in V/100, unsigned, indicated by the two registers starting at address 01FFh and is always available (without existence condition), the required fields on the row in the **Client** page are:

COLUMN	VALUE
INDEX	Unique number
LABEL	AI0
DESCRIPTION	Phase Voltage
DESCRIPTION_CODE	CUS40000
PROGRESSIVE	Phase-Voltage
FORMAT	CONST(-2)
MEASUREMENT_UNIT	CONST(VM00008)
MODBUS_COMMAND	A
MODBUS_ADDRESS	H01FF
FILTER	S[4]
EXISTENCE	CONST(1)
TYPE	AI

Let's look at the Filter field. Nowadays is quite common to see devices that store their resources as 32 bits variable. Unluckily the MODBUS protocol just manages 16 bits register. To overcome this limitations is quite common to store these values on 2 adjacent registers.

To manage this situation in the Televis Compact, you have to provide the lowest address of the registry pair and telling that the value is 4 bytes wide by using S[4] as filter. For the sake of clarity, S[4] is a short form for the 4 byte long mask HFFFFFFFF. Doing this way, the Televis Compact will automatically understand that the value spans over 2 registers.

This example shows how to assign to a resource a description that is not linked to the TelevisCompact dictionary.

Note that the value **CUS40000** of the field **DESCRIPTION_CODE** indicates a custom description that must be defined in the **PROGRESSIVE** field. The set of chars admitted I the **PROGRESSIVE** column are **a-z A-Z 0-9 .(dot) and –(hyphen)**, this restriction is due to the fact that this field act as resource identifier inside the software , but a proper custom name can be set in the supervisor system using a full character set.
Of course this applies to any type of resource, either an analog or digital input, output, or alarm.

1.5 Digital input reading without existence condition

The reading of a digital input without existence condition is achieved by compiling a single row in the P0435 **Client** page.

E.G.: if you want read the value of the **Digital input 1** and knowing that this value is expressed by the first bit of the LSB of the register at address 0211h, the required fields on the row in the **Client** page are:

COLUMN	VALUE
INDEX	Unique number
LABEL	DI1
DESCRIPTION	Digital Input 1
DESCRIPTION_CODE	STA00001
PROGRESSIVE	1
MEASUREMENT_UNIT	CONST(VM00021)
MODBUS_COMMAND	S
MODBUS_ADDRESS	H0211
FILTER	H0001
EXISTENCE	CONST(1)
TYPE	DI

1.6 Alarm reading with existence condition

The reading of a single alarm with existence condition is achieved by compiling a single row in the P0435 **Client** page, and possibly a row in the **Params** page.

E.G.: if you want read the status of the alarm **High Temperature 2** conditioned by the value of the parameter **NUM_ANAG_IN** and indicated by the value of the first bit of the LSB of the register at address 0207h, the required fields on the row in the **Client** page are:

COLUMN	VALUE
INDEX	Unique number
LABEL	AlmH2
DESCRIPTION	High Temperature 2
DESCRIPTION_CODE	ALM00177
MEASUREMENT_UNIT	CONST(VM00021)
MODBUS_COMMAND	S
MODBUS_ADDRESS	H0207
FILTER	H0001
EXISTENCE	Params(NUM_ANAG_IN)>CONST(1)
TYPE	Alarm

Note that the difference from reading an alarm without existence condition is the presence of a Boolean expression in the field **EXISTENCE**. In this particular case, we reference the value **NUM_ANAG_IN** in the **Params** page that, if greater than 1, it guarantees the existence of the alarm described (for the parameters refer to the relevant section).

Again you can have multiple similar alarms, so if we had more alarms of the same types as described above, each row would have different value for **MODBUS_ADDRESS** and **FILTER** field, and the filed **PROGRESSIVE** with a value ranging from 1 to n, where n is the number of alarms with the same **DESCRIPTION_CODE**, we would have also different **EXISTENCE** field, such **Params(NUM_ ANAG_IN)>CONST(3)** for the existence of alarms 1 to 4.

1.7 Analog input reading with existence condition

The reading of a analog input without existence condition is achieved by compiling a single row in the P0435 **Client** page.

1.7.1 Analog input conditioned by probe error

E.G.: if you want read the value of analog input **Probe 1** and knowing that it is conditioned only on the correct operation of the probe itself, which can also be a negative value, expressed in C°, with a decimal place, in the two bytes read at address 2001h, the required fields on the row in the **Client** page are:

COLUMN	VALUE
INDEX	Unique number
LABEL	AI1
DESCRIPTION	Analog Input 1
DESCRIPTION_CODE	INP40000
PROGRESSIVE	1
FORMAT	CONST(-1)
COMPLEMENTED	TRUE
MEASUREMENT_UNIT	CONST(VM00000)
MODBUS_COMMAND	A
MODBUS_ADDRESS	H0201
FILTER	S[4]
EXISTENCE	CONST(1)
TYPE	AI
TEST_CONDITION	Client(ProbErr1)

In this case, in addition to the fields used in the examples above, you specify the **COMPLEMENTED** field, indicating that the value may be read can also be negative, and the **TEST_CONDITION** field, in which, instead of a boolean expression is placed a direct reference to the row with **LABEL** set to **ProbErr1** in the same **Client** page. In this way the resource is defined as ever-existing, but in error if there is a physical error of the probe itself.
In case of probe error the data of the probe is filed by the TelevisCompact so as be recognizable in historical data analysis phase.
For completeness the row that is referred to **Client(ProbErr1)** could be the following:

COLUMN	VALUE
INDEX	Unique number
LABEL	ProbeErr1
DESCRIPTION	Probe Error 1
DESCRIPTION_CODE	ALM40125
MEASUREMENT_UNIT	CONST(VM00021)
MODBUS_COMMAND	S
MODBUS_ADDRESS	H0209
FILTER	H0001
EXISTENCE	CONST(1)

TYPE	Alarm

1.7.2 Analog input conditioned by parameter

However, if the existence of a probe is determined by a parameter, you can indicate this using the **EXISTENCE** field, for example:

COLUMN	VALUE
INDEX	Unique number
LABEL	AI2
DESCRIPTION	Analog Input 2
DESCRIPTION_CODE	INP40000
PROGRESSIVE	2
FORMAT	CONST(-1)
COMPLEMENTED	TRUE
MEASUREMENT_UNIT	CONST(VM00000)
MODBUS_COMMAND	A
MODBUS_ADDRESS	H0203
FILTER	S[4]
EXISTENCE	Params(NUM_SONDE_ANAG)>CONST(1)
TYPE	AI

1.8 Digital input reading conditioned

The reading of a digital input conditioned is achieved by compiling a single row in the P0435 **Client** page.

E.G.: if you want red the status of **Digital Input 2** conditioned by the value of the parameter **NUM_IN_DIG** and indicated by the value of second bit of the LSB of the register at address 2011h, the required fields on the row in the **Client** page are:

COLUMN	VALUE
INDEX	Unique number
LABEL	Di2
DESCRIPTION	Digital Input 2
DESCRIPTION_CODE	STA40001
PROGRESSIVE	2
MEASUREMENT_UNIT	CONST(VM00021)
MODBUS_COMMAND	S
MODBUS_ADDRESS	H2011
FILTER	H0002
EXISTENCE	Params(NUM_DIG_IN)>CONST(1)
TYPE	DI

1.9 Parameters

The parameters of a MODBUS device are descrive using mainly the two page **Params** and **ParamsDefaults**, and if the driver is applicabile to different models, the **Models** page also.
The most important of the three page is the **Params** page: it describes the entire list of parameters provided by the MODBUS device an uses many of the fields already described in previous examples in the **Client** Page.

In addition there are fields **LOW_LIMIT**, **HIGH_LIMIT** and **R_W_RW** that as you can guess from the names themselves indicate respectively the minimum value, the maximum value and the access mode for the single parameter described in the row of the page.

The **LOW_LIMIT** and **HIGH_LIMIT** fields have a peculiarity when dealing with values that are represented with a resolution of one or more decimal positions. The values provided as limits must be multiplied by the power of ten corresponding to the value resolution, in order to remove the fractional part.

E.g. If a set point has a resolution of 1 decimal position, its limits could be -23.5 and 23. These values must be multiplied by 10^1 before being written in the **Params** Page as -235 and 230.

When you write a parameters, the Televis Compact application checks is the value to be written is within the allowed range, if the check fails the writing operation is forbidden. So please pay attention when you write down the limits of a parameter with a resolution of one or more decimal positions because an wrong value could inhibit the modification of that parameter.

E.G.: As the description of the parameter **Number of analog inputs**, with value in the range 1-3, numeric, unsigned, two bytes and accessible for both reading and writing at the address 040h, the necessary fields in the corresponding row of the **Params** page are:

CAMPO	**VALORE**
INDEX	**Unique number**
LABEL	**NUM_ANAG_IN**
DESCRIPTION	**Number Analog Inputs**
DESCRIPTION_CODE	**CUS40000**
PROGRESSIVE	**Number-Analog-Inputs**
LOW_LIMIT	**1**
HIGH_LIMIT	**3**
MEASUREMENT_UNIT	**CONST(VM00020)**
MODBUS_COMMAND	**S**
MODBUS_ADDRESS	**H040**
FILTER	**HFFFF**
R_W_RW	**RW**

The default values of parameters are listed in the **ParamsDefaults** page which must necessarily have the same number of rows in the **Params** page. The fields that link the two pages are the field **INDEX** and the field **LABEL**. The subsequent fields in **ParamDefaults** page are used to indicate different sets of default values of the parameters used to model the device using the field **Poli** in the **Models** page.

E.G.: if you have three models, respectively with **POLI** 121, 130 and 170 described in the **Models** page and you want to describe the set of default values for parameters, the **Models** page will have the following fields: **INDEX, LABEL, 121, 130, 170**.

1.10 Remote commands

In this section we are going to talk about the remote commands that can be sent to a device (e.g. "**Turn On**","**Turn Off**","**Defrost**").

This section needs introduction to avoid misunderstanding of the word command, that can be used for both a remote command (e.g. "**Turn On**","**Turn Off**","**Defrost**") and a basic protocol command (e.g. H06 for writing a MODBUS register). We'll refer to the former as remote commands or commands and to the latter as requests.

The Televis Compact manages the remote commands as a sequence of simple writing request:

- The remote commands must be inserted in the **Client** page. They are similar to resources because they have a description and an existence condition, but they differ because they do not have an associated memory address or value condition. Instead, they have a sequence of request, which must be inserted in the **WRITE_SEQUENCE** cell, using references to the simple request that are inserted in the **Write** Page.
- The simple writing request must be inserted in the Write Page.

E.g. Let's build the "**Reset Alarms**" remote command for the sample device we are building. As we can read in the specification, it's possible to trigger the "**Reset Alarm**" function by writing the value 0 in the register H02A using the H06 MODBUS request.

First of all, let's add the writing request to the **Write** page as follows:

COLUMN	VALUE
INDEX	Unique number
NAME	Reset Alarms
VALUE	0
MODBUS_COMMAND	S
MODBUS_ADDRESS	H02A
FILTER	HFFFF

After that add a row in the **Client** Page with the following data:

COLUMN	VALUE
INDEX	Unique number
LABEL	Cmd1
DESCRIPTION	Reset Alarms
DESCRIPTION_CODE	CUS40000
PROGRESSIVE	Reset-Alarms
EXISTENCE	CONST(1)
TYPE	Net Command
WRITE_SEQUENCE	Reset Alarms

Please pay attention that each item that you place in the **WRITE_SEQUENCE** column will refer to an item in the **Write Page**. The reference is done by name.

Example: To describe the **Reboot** command that we know from technical specification to writing with the MODBUS command H06 at the address 02Bh the word (two bytes) with a value of -1, the required fields in the corresponding row of the **Write** page are:

CAMPO	VALORE
INDEX	Unique number
NAME	Reboot
VALUE	-1
COMPLEMENTED	TRUE
MODBUS_COMMAND	S
MODBUS_ADDRESS	H02B
FILTER	HFFFF

After that add a row in the **Client** Page with the following data:

COLUMN	VALUE
INDEX	Unique number
LABEL	Cmd4
DESCRIPTION	Reboot
DESCRIPTION_CODE	CUS40000
PROGRESSIVE	Reboot
EXISTENCE	CONST(1)
TYPE	Net Command
WRITE_SEQUENCE	Reboot