**Answer to Reset**

**Resetting the card**

To perform any communication with the card, we will need to obtain a card object. A card object is created using the new Card() constructor. The constructor accepts a card reader name as parameter. The predefined variable \_scsh3.reader contains the default reader name.

var card = new [Card](http://www.openscdp.org/scsh3/card.html)(\_scsh3.reader);

Now we can reset the card. There are two ways of resetting.

**Cold Reset:**

The cold reset is the initial start of all communication between the card and the reader. If the card is already powered up, then a cold reset will first deactivate power supply. Then power is supplied to the card and the reset line is used to signal a reset to the chip. The terminal then waits for an answer (ATR).

**Warm Reset:**

During warm reset the terminal is sending a reset signal to the card without disconnecting power supply.

In our case we want a cold reset.

var [atr](http://www.openscdp.org/scsh3/atr.html) = card.reset(Card.RESET\_COLD);

**ATR - Answer to Reset**

After the reset signal from the terminal, the card will answer with a bytestring called ATR. This string contains certain information about the card which are necessary for the communication between card and terminal. After a warm reset the card answers with a different ATR, e.g. to switch to a slower communication rate.

To get the data the card was sending, we use the class [ATR](http://www.openscdp.org/scsh3/atr.html). There are two methods to extract details from the ATR: "[toByteString](http://www.openscdp.org/scsh3/atr.html#toByteString)()" and "[toString](http://www.openscdp.org/scsh3/atr.html#toString)()".

var atrbin = atr.toByteString();

print(atrbin);

Expample:

0000 3B FF 18 00 FF 81 31 FE 45 65 63 0D 02 50 02 80 ;.....1.Eec..P..

0010 00 08 37 70 20 10 05 00 B2 ..7p ....

For more details use the method "toString()"

print(atr.toString());

Example:

TS : 3B Direct logic

TO : FF K = 15 byte [historical characters]

TA1 : 18 Fi/f = 372/ 5 [clock rate conversion factor / max. frequency (MHz)]

Di = 12 [bit rate conversion factor]

TB1 : 00 pa = 4 % [programming voltage accurancy]

I = 25 mA [maximum current]

P = 0 V [programming voltage]

TC1 : FF N = 255 etu [extra guardtime]

TD1 : 81 T = T=1 [protocol type]

TD2 : 31 T = T=1 [protocol type]

TA3 : FE IFSC = 254 [information field size]

TB3 : 45 CWT = 43 etu [character waiting time]

BWT = 12 etu [block waiting time]

65630D025002800008377020100500 ec..P....7p ...

The first byte of the string is the initial character "T0". It defines which of the two transmission technique, "direct convention (3B)" or "indirect convention (3F)", should be used. In our case we have the byte 3B and a direct convention.

The second byte is the format character T0. Here we obtain the length of the historical characters. In our ATR the last 15 bytes are the historical characters. They contain some information e.g. the card's features and operating system version.

TA1 to TC3 are interface characters. They are necessary to determine the transmission between the card and the terminal. They contains informations like voltage, transmission rates and timeouts.

TA3 determines the information field size. The terminal used this value to determine the max. size of a block send to the card.

The character TB3 is relevant for the character and block waiting time. The character waiting time is the max. timeout between two subsequent bytes and the block waiting the time between twi subsequent blocks. If the waiting time is higher than determined, then terminal will issue an timeout error.

All waiting times are expressed as Elementary Time Unit (ETU), which is the time required to transmit a single bit. The default communication rate is 9600 bits per second, which translates to an ETU of 1/9600 seconds.