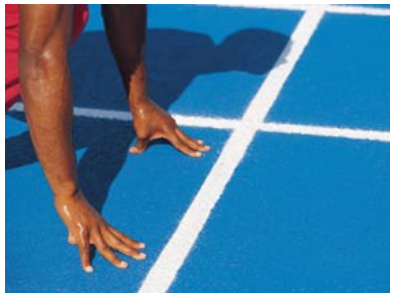
maXbox



maXbox Starter 51

Work with 5 Use Cases Vol. 2.1

1.1 Arduino Pressure Sensor

In the technology world, your use cases are only as effective as the value someone's deriving from them. We want to show 5 of them:

1. Arduino Sensor Measure
2. XML DOM Analyser
3. Parallel Batch Processing
4. QR-Code Generator
5. Message Encryption & Robotics (Tutor 50)
6. Image Processing

In this first use case we will use the Adafruit BMP280 sensor and maXbox to send the data to a browser. This sensor can measure barometric pressure and temperature with very good accuracy. Because pressure changes with altitude we can also use it as an altimeter with ±1 meter accuracy! Accuracy for barometric pressure is ±1 hPa and ±1.0°C for temperature.

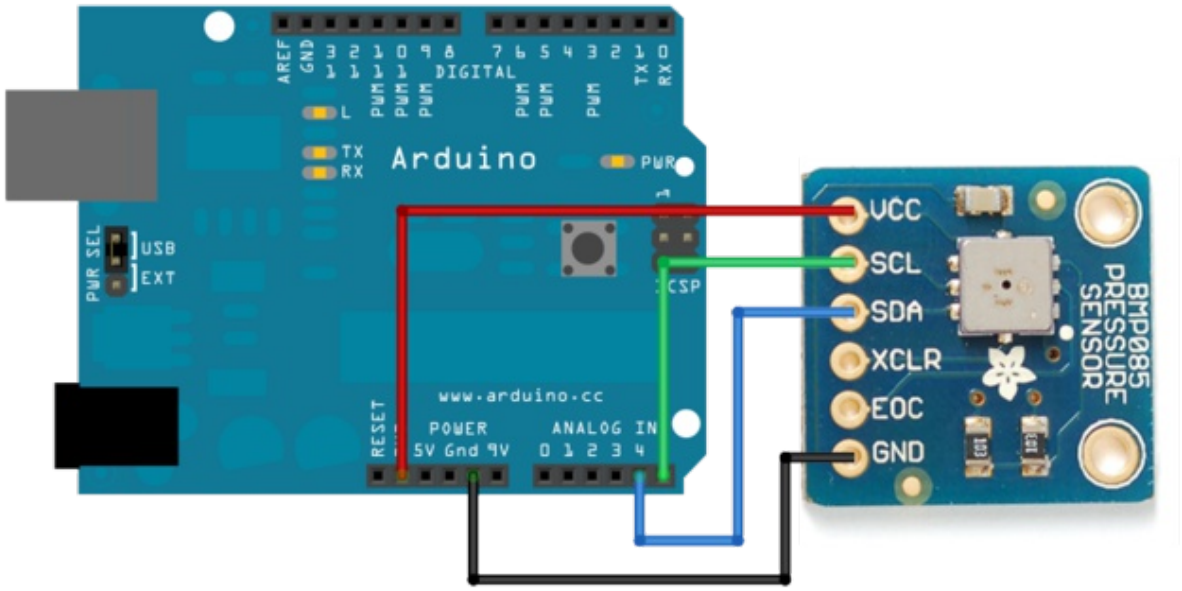
You can connect it with your Arduino board with I2C or SPI connection protocol. It has a 3.3V regulator and level shifting so you can use it with a 3V or 5V logic micro controller.

https://github.com/adafruit/Adafruit_BMP280_Library

This is a library for the BMP280 humidity, temperature & pressure sensor and we just had to set a define to work with 3.3V and I2C:

```
#define BMP280_ADDRESS     (0x76)
```

A great way for writing effective use cases is to walk through a sample use case example and watch how it can be leveraged to something complex. The connections are pretty forward, see the Fritzing image below with the breadboard circuit schematic (with BMP 085 or BMP 280).



These sensors use I2C or SPI to communicate, 2 or 4 pins are required to interface. We will use the I2C connection (SCL and SDA pins), (I2C 7-bit address 0x77 set on 085)

4 Connections of the Adafruit BMP280[1] sensor:

VCC in to Arduino 3.3 V pin (3.3V)
GND to GDN pin (GND)
SCL to Arduino SCL pin (A5)
SDO - - -(Master Input ← Slave Output)
SDA to Arduino SDA pin (A4)
CS - (Leave other pins disconnected)

Meteorologists use "millibars" in charting atmospheric pressure and your barometer has a second scale or ring which reads in millibars (mb), or "hectopascals" (hPa). The latter is used as a unit of pressure. Standard air pressure at standard elevation (sea level) at 15°C and 45° latitude is 1013 hPa or 29.92 inches of mercury.
However, conversion is easy. 1000 hPa are equal to 1000 mbar, which is equal to 750 mm of mercury in a barometric column, which is 0.987 of the average atmospheric pressure, which on global average is 1013 millibars or hectopascals (1013.25*(e^-(m/8005)).
Hectopascal (symbol hPa) is a SI unit, the international system of units now recommended for all scientific purposes.

You can find the complete example at:
https://sourceforge.net/projects/maxbox/files/Arduino/BMP280Adafruit_maXbox4.zip/download

[1] Soldered onto PCB with 3.3V regulator, I2C level shifter and pull-up resistors on I2C pins.

We will now output the values over a serial connection to display on the screen (browser, monitor, app). The code in maXbox connects to via RS232 serial to the Arduino board and provides with a dynamic socket server the data to a browser:

```
begin   //@main
  if Fileexists(JSCRIPT) then
    jsload:= filetostring(JSCRIPT);
  tmp2:= '0';
  HStarttime:= DateTimeToInternetStr(Now,true)
  //TWebServerCreate;
  with TIdHTTPServer.Create(Nil) do begin
    sr:= GetIPfromHost(getHostName)  //'172.16.10.80';
    try
      Bindings.Add.IP:= sr; //'127.0.0.1' //sr;
      Bindings.Add.Port:= IPPORT;
      OnCommandGet:= @HTTPServerCommandGet;
      Active:= True;
    except
      writeln(ExceptionToString(ExceptionType, ExceptionParam));
      Active:= False; Free;
    end;
    try
      StartLogger;
      If IsCOMPort then
        writeln('SerialPortNames Init: '+GetSerialPortNames);
      arTimer:= TTimer.Create(Self);
      arTimer.Enabled:= true;
      arTimer.Interval:= 2000;
      blockser:= TBlockserial.create;
      blockser.RaiseExcept:= true; //blockser.SetCommState;
      blockser.Config(9600,8,'N',1,false,false);
      blockser.Connect(COMPORT);
      arTimer.OnTimer:= @eventActTimer;
      //ConnectArduinoSense;
      Writeln('Hello Temp/Web server start at: '+sr);
      ShowMessageBig('maXbox server at: '+sr+':'+itoa(IPPORT)+#1310+
                          '  Press OK to quit webserver!'+#13);
    finally
      writeln('SocketServer stop: '+timetoStr(now)); //Destroy;
      Active:= False; Free;
      hlog1.Add('<<<< Stop socktimer {App_name}{40@}{now}');
      hlog1.Free;
      arTimer.enabled:= false;
      arTimer.Free;
      blockSer.CloseSocket;
      blockSer.Free;
      writeln('Serial stop: '+timetoStr(now)); //Destroy;
    end;
  end;
```

If you want to connect multiple BME280's to one microcontroller with SPI, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS (chip select) pin.

https://learn.adafruit.com/adafruit-bmp280-barometric-pressure-plus-temperature-sensor-breakout/pinouts

Bosch has stepped up their game with their new BMP280 sensor, an environmental sensor with temperature, barometric pressure that is the next generation upgrade to the BMP085/BMP180/BMP183. This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI bus mode!

1.2 XML DOM Parser & Analyser

The idea was to get RSS feeds over XML DOM. MS XML Core Services (MSXML) is a set of services that allow applications written in JScript, VBScript, and Microsoft development tools to build Windows-native XML-based applications. It supports XML 1.0, DOM, SAX, an XSLT 1.0 processor, XML schema support including XSD and XDR, as well as other XML-related technologies.
https://en.wikipedia.org/wiki/MSXML

All MSXML products are similar in that they are exposed programmatically as OLE Automation (a subset of COM) components. Developers can program against MSXML components from C, C++ or from scripting languages such as maXbox, JScript or VBScript.

So first we create a COM object depending you have MSXML 4 or MSXML 6 registered as an OLE Automation component:

```
procedure TestXMLDOM_Doc;
var xDoc, Node, s : OleVariant;
begin
  // Create a COM object If you have MSXML 4 or MSXML 6:
  //xDoc := Sys.OleObject('Msxml2.DOMDocument.4.0');
  xDoc:= CreateOleObject('Msxml2.DOMDocument.6.0');
  xDoc.async:= False;
  // Load data from a file
  xDoc.load(XMLDOCF);
  // Report error, for instance, markup or file structure is invalid
  if xDoc.parseError.errorCode <> 0 then begin
    s:= 'Reason:' + #9 + xDoc.parseError.reason + #13#10 +
         'Line:' + #9 + VarToStr(xDoc.parseError.line) + #13#10 +
         'Pos:' + #9 + VarToStr(xDoc.parseError.linePos) + #13#10 +
         'Source:' + #9 + xDoc.parseError.srcText;
    // Post an error to the log and exit
    //Log.Error('Cannot parse the document.', s);
    writeln('Cannot parse the XML document. '+ s);
    //Exit;
```

```
  end;
  // Obtain the first node
  Node:= xDoc.documentElement;
  // Process the node
  ProcessNode(Node);
end;
```

Then we process each node we get with the first `xDoc.documentElement`. Each element in the XML tree can have an arbitrary number of attributes. You can read and set these node attributes, and you can create also new attributes.

```
procedure ProcessNode(ANode : OleVariant);
var
  FID, s, Attrs, Attr, ChildNodes : OleVariant;
  i: integer;
begin
  // If the node value is not nil, output it getVarType
  if VarType(ANode.nodeValue) <> 1 then
    writeln('Node value: ' + VarToStr(ANode.nodeValue));
   // Process node's attributes
  s:= Copy(ANode.nodeName, 1, 1);
  if s <> '#' then begin // Obtain attribute collection
    Attrs:= ANode.attributes;
    for i:= 0 to Attrs.length- 1 do begin
      Attr:= Attrs.item[i];
      writeln('Attr_ '+ Attr.nodeName+ ': '+ VarToStr(Attr.nodeValue));
    end;
  end;
  // Obtain a collection of child nodes
  ChildNodes:= ANode.childNodes;
  // Processes each node of the collection
  for i:= 0 to ChildNodes.length- 1 do
    ProcessNode(ChildNodes.item[i]);  //Recursion!
 end;
```

To find out how many nodes a particular tree has, get a pointer to the node and then use the `nodes.length` property:

```
Nodes:= Doc.selectNodes('//control');
writeln('nodes select: '+itoa(nodes.length));
```

Ot to find out how many attributes a particular element has, get a pointer to the element and then use the `.attributes.length()` method.

```
Attrs:= ANode.attributes;
for i:= 0 to Attrs.length - 1 do begin
```

Be aware about the “node” name, a node is a general specifier for several types in an XML structure like an element, see the table below.

A total of 13 node types are currently supported by the Microsoft XML parser. The following table lists the most commonly used node types:

Node Type	Example
Document type	<**!DOCTYPE** food SYSTEM "food.dtd">
Processing instruction	<**?xml** version="1.0"?>
Element	<**drink** type="beer">Felsenau Weizen</**drink**>
Attribute	**type**="beer"
Text	**Felsenau Hefe Weizen**

As you see, a program or in our case a script called an XML parser can be used to load an XML document into the memory of your computer. When the document is loaded, it's information can be retrieved and manipulated by accessing the Document Object Model (DOM).

The DOM represents a tree view of the XML document. The DOM is a programming interface for HTML and XML documents. It defines the way a document can be accessed and manipulated. The `documentElement` in MSXML is the top-level of the tree. This element has one or many `childNodes` that represent the branches of the tree.
A Node Interface is used to read and write (or access if you like) the individual elements in the XML node tree. The `childNodes` property of the `documentElement` can be accesses with a for/each construct to enumerate each individual node or recursively in our case.

https://sourceforge.net/projects/maxbox/files/Examples/13_General/760_XML_DOM_OLE.TXT/download

Use also an XPath expression to obtain a list of "control" nodes with the "Checkboxes" namespace as an example:

```
Nodes:= Doc.selectNodes('//control');
  writeln('nodes select: '+itoa(nodes.length));
  for i:= 0 to Nodes.length - 1 do begin
    // Get node from the collection of the found nodes
    Node:= Nodes.item(i);
    // Get child nodes
    ChildNodes:= Node.childNodes;
    writeln(ChildNodes.item(1).text+ ': '+ ChildNodes.item(2).text);
```

Our `selectNodes` expression is case sensitive!
You can use IE, Firefox, Edge, Chrome and so on to view an XML document just as you view any HTML page[2]. There are several ways to open an XML document. You can click on a link, type the URL into the address bar, double-click on an XML document in a folder, and so on.

[2] http://www.xmlfiles.com/examples/

On of the easiest way traversing through an XML-tree from a file or a RSS stream feed (without differentiation access) I found while working on a HTML-site I call it traverse the nodes.

```
<html>
<body>
<script language="VBScript">

txt="<h1>Traversing the node tree</h1>"
document.write(txt)

set xmlDoc=CreateObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")

for each x in xmlDoc.documentElement.childNodes
  document.write("<b>" & x.nodename & "</b>")
  document.write(": ")
  document.write(x.text)
  document.write("<br>")
next

</script>
</body>
</html>
```

So in a script the same goes like this, the following code loops through the child nodes, that are also element nodes:

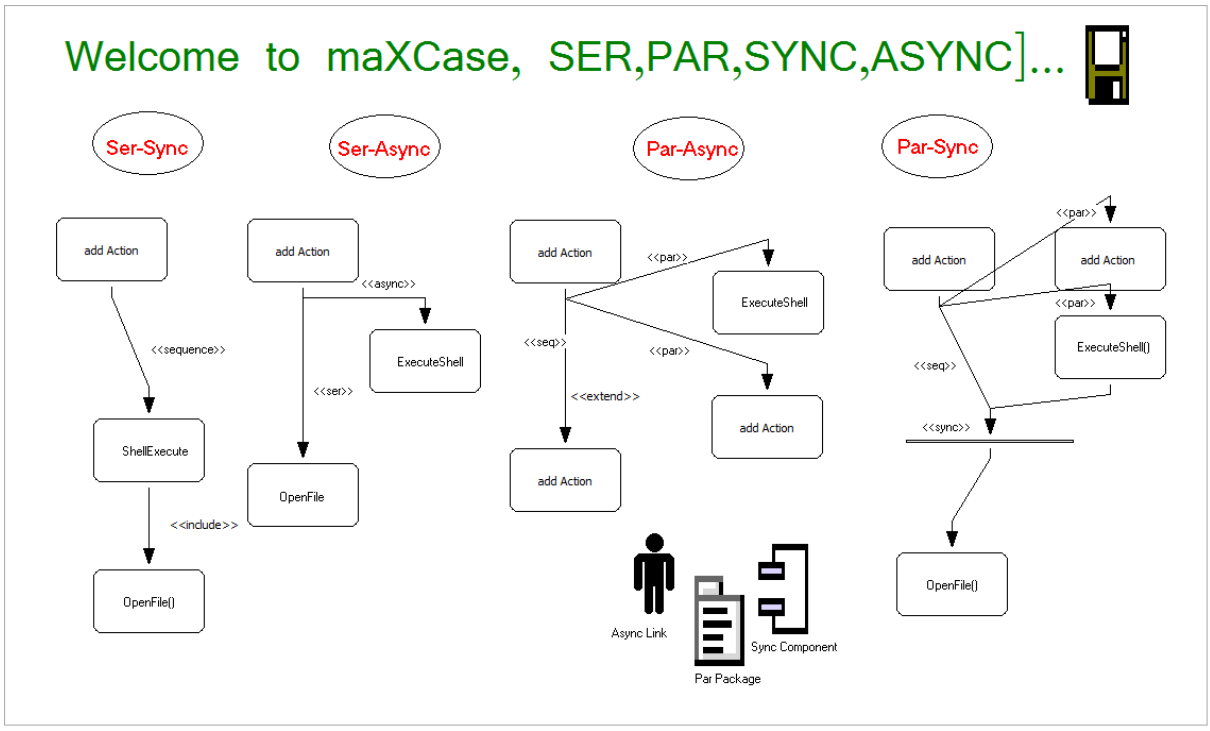
```
 for i:= 0 to doc.documentElement.childNodes.length-1 do begin
    node:= doc.documentElement.childNodes.item(i);
    writeln(node.nodename +': '+node.text)
  end;

// or with less code after refactoring:

  xd:= doc.documentElement.childNodes;
  for i:= 0 to xd.length-1 do
    writeln(xd.item[i].nodename +': '+xd.item[i].text)
```

When setting the `documentElement` property, the specified element node is inserted into the child list of the document after any document type node. To precisely place the node within the children of the document, call the `insertBefore` method of the `IXMLDOMNode`.
The `parentNode` property is reset to the document node as a result of this operation.

All the functions you find with the script: `760_XML_DOM_OLE.TXT`

1.3 Parallel Batch Processing

Nowadays, many computers have multiple independent processing units. These are often underused, simply because writing parallel applications can be a daunting or haunting task.
Our next use case spawns a few new batch processes that execute in parallel and minimized or close the console. Wait time states needs to be adjusted probably, depending on how much each process does and how long it is running. You probably also need to adjust the process name for which `tasklist` is looking if you're doing something else.



There is no way to properly count the processes that are spawned by this batch, though. One way would be to create a random number at the start of the batch (`%RANDOM%`) and create a second helper batch that does the processing.
The ultimate goal is having a method to execute tasks in parallel that is simple to use from an interactive shell or script:

```
// 3 Parallel - Async (Par and Non Waiting 2 Tasks)
 ShellExecute3('cmd','/c systeminfo > sysinfo_machine3_1.txt',secmdopen)
 ShellExecute3('cmd','/c systeminfo > sysinfo_machine3_2.txt',secmdopen)
```

ShellExecute is a standard MS Windows function (`ShellApi.h`) with good documentation on MSDN (note their remarks about initialising COM if you find the function unreliable).
`Systeminfo` is a task running on the shell and redirected to write a file of a report we use for our machines, also in a sync way possible.

```
// 1 Serial - Sync (Waiting Sequence)
  ExecuteShell('cmd','/c systeminfo > sysinfo_machine1.txt')
```

One note concerning `ShellExecute()`: Its ability to execute programs is a side effect of the fact that Microsoft (on a clean install) defined an "Application" document type with an "Open" verb that matches .exe files, among others. But we need to realize that using `ShellExecuteEx` with the name of an executable is treating it as a document, and thus subject to problems with the document type settings in the registry. Among other potential problems, some viruses install themselves as the handler for "Application" documents.

http://wiki.freepascal.org/Executing_External_Programs

You can see there's a difference of the switch /c after starting the shell. With **/c** you run a DOS command and return immediately and with **/k** you run a DOS command and keep the DOS-window open ("stay in DOS") alike. You find the script at:
http://www.softwareschule.ch/examples/systeminfo.txt
http://www.softwareschule.ch/examples/systeminfo.htm

Launching an external program and waiting until it is terminated is quite another story. We can tell if a process has completed by monitoring its process handle. That process handle can be obtained by using one of two Win32 API-functions to start the external program: `ShellExecuteEx` or `CreateProcess`. The simplest method is: start the external application with `ShellExecuteEx` and then monitor the process handle including get an exit code with `WaitForSingleObject`:

```
function ShellExecuteEx(aFileName: string; const Parameters: string =
''; const Directory: string = ''; const WaitCloseCompletion: boolean =
false): Boolean;
 var SEInfo: TShellExecuteInfo;
     ExitCode: DWORD; Wait: Boolean;
begin
   FillChar(SEInfo, SizeOf(SEInfo), 0) ;
   SEInfo.cbSize:= SizeOf(TShellExecuteInfo) ;
   with SEInfo do begin
     fMask:= SEE_MASK_NOCLOSEPROCESS;
     Wnd:= Application.Handle;
     lpFile:= PChar(aFileName);
     lpParameters:= PChar(Parameters);
     lpDirectory:= PChar(Directory);
     nShow:= SW_SHOWNORMAL;
   end;
   Result:= ShellAPI.ShellExecuteEx(@SEInfo);
   if Result and WaitCloseCompletion then
     repeat
       Application.ProcessMessages;
       GetExitCodeProcess(SEInfo.hProcess, ExitCode);
       Wait := WaitForSingleObject(SEInfo.hProcess, 10000) <> ExitCode;
     until not Wait; // Wait max. 10 seconds
end;
```

You can also elevate your batch or task as an admin with:

```
  ShellExecuteExAsAdmin(hinstance,'cmd','/c dir');
  ShellExecuteExAsadmin(hinstance,'cmd','/k dir');

procedure ShellExecuteExAsAdmin(hWnd: HWND; Filename: string;
Parameters: string);
var sei: TShellExecuteInfo;
begin
  FillChar(sei, SizeOf(sei), 0);
  sei.cbSize:= sizeof(sei);
  sei.Wnd:= hWnd;
  sei.fMask:= SEE_MASK_FLAG_DDEWAIT or SEE_MASK_FLAG_NO_UI;
  sei.lpVerb:= 'runas';
  sei.lpFile:= PChar(Filename);
  sei.lpParameters:= PChar(Parameters);
  sei.nShow:= SW_SHOWNORMAL;
  if not ShellAPI.ShellExecuteEx(@sei) then
    RaiseLastOSError;
end;
```

Then you get a modal dialog to accept run as an admin by `runas`.
The last example runs also asynchronous and opens two DOS shells independent and stay in DOS:

```
//Run two DOS commands and keep the DOS-window open ("stay in DOS"):
 cyShellExecute('open', 'cmd', '/k dir /s', Exepath, SW_SHOW);
 cyShellExecute('open', 'cmd', '/k dir', 'C:\maxbox\maxbox3\', SW_SHOW);
```

You can measure time difference benchmarks or performance results, but its difficult with escaping processes as you do have with async functions!

// Function TickCountToDateTime(Ticks : Cardinal) : TDateTime');

Ex: `writeln(datetimeToStr(TickCountToDateTime(gettickcount)));`

```
  with TStopwatch.create do begin
    start
     ShellExecuteX('open', 'cmd', '/k dir /s', Exepath, SW_SHOW);
     //ShellExecuteExAsadmin(hinstance,'cmd','/k dir');
    stop
     writeln(getvalueStr)
    free;
  end;
```

Hill hint: For two clocks moving inertially (**not** accelerating) relative **to** one another, this effect **is** reciprocal, **with** each clock measuring the other **to** be ticking slower. As a clock approaches the speed of light it will almost slow to a stop, although it can never quite reach light speed so it will never completely stop.

1.4 QR Code Generator

Using the Google Chart Tools / Image Charts (aka Chart API) you can easily generate QR codes, this kind of small images are a special type of two-dimensional bar-codes.
There are two different types of QR code; "Static" and "Dynamic". A static QR code holds all of the info you wish to pass within the code itself.
They are also known as hard-links or physical world hyper-links. The API requires 4 simple fields be posted to it:

1. cht=qr this tells Google to create a QR code;
2. chld=M the error correction level of the QR code (see here below for more parametric info);
3. chs=wxh the width and height of the image to return (e.g. chs=250x250);
4. chl=text the URL encoded text to be inserted into the bar-code.

```
Const
  URLGoogleQRCODE='http://chart.apis.google.com/chart?chs=%dx
%d&cht=qr&chld=%s&chl=%s';
```

In fact there are 2 programming models used in TCP/IP applications. Non blocking means that the application will not be blocked when the app socket read/write data. This is efficient, because your app don't have to wait for connections. Unfortunately, it is complicated.
Now we jump to the code behind the URL, which is a http GET command as a function to fill a stream with the requested data, we deal direct with the Win32 API and `WinInet`:

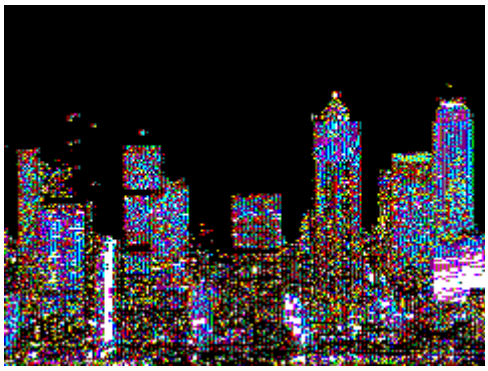
```
//WinInet
procedure GetQrCodeInet(Width,Height:Word;
                        C_Level,apath:string;const Data:string);
var
  encodURL: string;
  pngStream: TMemoryStream;
begin
  encodURL:= Format(URLGoogleQRCODE,[Width,Height,C_Level,Data]);
  pngStream:= TMemoryStream.create;
  HttpGet(encodURL, pngStream);    //WinInet
  with TLinearBitmap.Create do try
    pngStream.Position:= 0;
    LoadFromStream2(pngStream,'PNG');
    SaveToFile(apath);
    OpenDoc(apath);
  finally
    Dispose;
    Free;
    pngStream.Free;
  end;
end;
```

Our second line is creating the stream object and then we use the first method from `WinInet HttpGet` with the help of `HTTPEncode`. The object makes a bind connection with the Active method by passing an encoded URL to `URLGoogleQRCODE` within a memory stream. Then I use the `TLinearBitmap` class to create a PNG format image and in the end the QR-code is saved and opened to show the result!
The REST style emphasizes that interactions between clients and services are enhanced by having a limited number of operations, most of them is GET. Flexibility and simplification is provided by assigning resources just their own unique universal resource indicators (URIs). It's a bit comparable to a function chain which calls a stack of operations:

```
writeln(getEnvironmentString)
saveString(exepath+'envstring.txt',getEnvironmentString)
openDoc(exepath+'envstring.txt')

REST style: openDoc(envpath/savestring/getenvstring)
```

1.5 Message Encryption

We focus on a real world example from a PKI topic RSA to sign a public key included in a certificate. They usually operate with huge integers by multiplication of big primes and that's exactly what we want.
Each example in the following tutorial is shown in detailed small steps so it is easily reproduced in your own system behaviour. At first I show a simple solution to handle big numbers.



We know the `max` value for an `int64` is 9223372036854775807.
If we multiply 9223372036854775807 x 9223372036854775807 using the following routine we get:
85070591730234615847396907784232501249.

If we don't have a special routine we only get:
8.50705917302346158E37

And now you see the missing difference:

85070591730234615847396907784232501249

8.50705917302346158E37
I just drop the decimal point now:

850705917302346158473969077842325012 49
850705917302346158E37

The whole number has a length of 38 and we lost 20 signs, left only 18. So we lose precision or need any rounding.

// big number ex.:

```
function AddNumStrings (Str1, Str2: string): string;
var
  i : integer;
  carryStr, workerStr: string; worker: integer;
begin
  Result:= inttostr (length(Str1));
  Result:= '';
  carryStr:= '0';
  // make numbers the same length
  while length(Str1) < length(Str2) do
    Str1:= '0' + Str1;
  while length(Str1) > length(Str2) do
    Str2:= '0' + Str2;
  i:= 0;
  while i < length(Str1) do begin
    worker:= strtoint(copy(Str1, length(str1)-i, 1)) +
             strtoint(copy(Str2, length(str2)-i, 1)) +
             strtoint (carryStr);
    if worker > 9 then begin
      workerStr:= inttostr(worker);
      carryStr:= copy(workerStr, 1, 1);
      result:= copy(workerStr, 2, 1) + result;
    end else begin
      result:= inttostr(worker) + result;
      carryStr:= '0';
    end;
    inc(i);
  end; { while }
  if carryStr <> '0' then
    result:= carryStr + result;
end;
```

This function will add two numbers that are represented by their string digits, so we use strings as the representation of big integers.
Without that as a we said, we don't get the whole number:

```
i64e:= 9223372036854775807
PrintF('mulu %18.f ',[i64e*i64e])
>>> 8.50705917302346158E37
```

How does it work?
Go back to the basics and calculate the answer the same way you would if you were doing it with pencil and paper. Use string variables to hold the text representation of your numbers as digits and create functions that will add and multiply those number strings. You already know the algorithms, you learned it as a (script) kid!
Nowadays kids code a lot of with sensor kits with Arduino or Raspy, I call them sensor kids;-).

Next example shows the multiplication with this school concept:

```
function MultiplyNumStrings (Str1, Str2 : string): string;
var
  i,j, worker : integer;
  carryStr, workerStr, tempResult: string;
begin
  Result:= '';
  carryStr:= '0';
  tempResult:= '';
  // process each digit of str1
  for i:= 0 to length(Str1) - 1 do begin
    while length(tempResult) < i do
      tempResult:= '0'+ tempResult;

    // process each digit of str2
    for j:= 0 to length(Str2) - 1 do begin
      worker:= (strtoint(copy(Str1, length(str1)-i, 1)) *
                strtoint(copy(Str2, length(str2)-j, 1)))+
               strtoint (carryStr);
      if worker > 9 then begin
        workerStr:= inttostr(worker);
        carryStr:= copy(workerStr, 1, 1);
        tempResult:= copy(workerStr, 2, 1)+ tempResult;
      end else begin
        tempResult:= inttostr(worker)+ tempResult;
        carryStr:= '0';
      end;
    end; { for }
    if carryStr <> '0' then
      tempResult:= carryStr + tempResult;
    carryStr:= '0';

    result:= addNumStrings (tempResult, Result);
    tempResult:= '';
  end; { for }

  if carryStr <> '0' then
    result:= carryStr + result;
end;
```

But performing billions multiplications on huge numbers, in one single thread? Unless you've got a state-of-the-art over-clocked CPU cooled with liquid fluid helium, you'd have to wait a whole lot for this to complete and finish. However if you do have, you'd just have to wait for a very long time with a sort of brute force attack.

Each algorithm has a benchmark and also in our school example there's some room to improve. You should replace `strToInt(copy(S, Index, 1))` with `Ord(S[1])-Ord('0')` since all of the characters are '0'..'9', and replace any remaining `copy(S, Index, 1)` with `S[Index]`, which then allows some places, like `carryStr`, to replace String with Char.

Now we can get the real int64:

```
writeln(bigPow(2,64))
```

> 18446744073709551616

not only

```
maxcalcF('2^64')
>>> 1.84467440737096E19
```

Big Decimal is equally built for ease of use and reliability. It builds on top of Big Integer so the internal representation is a Big Integer for the significant digits before decimal point, and a scale to indicate the decimals after decimal point.

The most important aspect that is emphasized in this example is that you should apply big numbers when you need to get added precision or really large numbers, that means, if you don't understand the impact of the resulting calculation the most of the time a rounded or truncated result is good enough in your own code or when you add a new feature or fix a bug then you can switch to a big numb library:

```
writeln(bigMulu('92233720368547758O7','92233720368547758O7'))

writeln(MuluNumStrings('92233720368547758O7','92233720368547758O7'))
```

```
procedure getUTCTime;
var lHTTP: TIdHTTP;
  lReader, s: string; //TStringReader;
begin
  lHTTP:= TIdHTTP.Create(nil);
  try
    lReader:= lHTTP.Get('http://tycho.usno.navy.mil/cgi-bin/timer.pl');
    //while lReader.Peek > 0 do begin
      if Pos('UTC',lreader) > 0 then
        Writeln(lreader);
  finally
    //lhttp.close;
```

```
    lHTTP.Free;
    //lReader.Free;
  end;
end;
```

1.6 The RSA Crypt System

Next real word example is the following teaching crypt system:

```
Ref of Herdt P.119 Version 10: Network Security

1. Modulus Prime1: 23  &  Prime2: 47
2. RSA Modul Public [p*q] as N: 1081 //Modulus
3. Phi(N) Private: 1012 = 22*46
4. Public RSA Exponent: 3
5. Private D: 675
6. Public (3,1081) - Private (675,1081)
```

What we want in the end is to sign a message or a hash, whatever, with that simple two functions:

```
  cipher c[i]   = m[i]^e (mod N)
  decipher m[i] = c[i]^d (mod N)
```

What we do next is simple, we encrypt the letter 'A' as `ASCII 65` with the following public key of **`Public`** `(3,1081)`:

```
   writeln(RSAEncrypt('65','3','1081'))
>>> 51
   writeln(RSADecrypt('51','675','1081'));
>>> 65
```

What we also do is more explicit with RSA:

```
   writeln('RSA cipher message:')
   m:= 'A';
   c:= RSAEncrypt(intToStr(ord(m[1])),'3','1081')
   writeln(c)
   writeln(RSADecrypt(c,'675','1081'))

   writeln('RSA sign message:')
   m:= 'A';
   c:= RSAEncrypt(intToStr(ord(m[1])),'675','1081')
   writeln(c)
   m:= RSADecrypt(c,'3','1081')
   writeln(chr(strToInt(m)))
```

As you can see we convert the string A to a ASCII number 65 and back to string, cause the big numb library handles only strings as numbers.

For this crypt system to work, the system must guarantee that it is (effectively) impossible to decrypt the message without knowledge of the private key. In particular, it must be impossible to decrypt using the public key, or to derive the private key from the public key.

And here's the whole implementation step by step, especially to find the private key D[3]:

```
  //1. Init - Choose two prime numbers p and q and e.
  Prime1:= 23          //13;  //7;  //11;
  Prime2:= 47          //23;  //11; //7;
  RSA_Exponent:= 3;    //5;   //17; //5;

  PrintF('1. Modulus Prime1: %d  &  Prime2: %d',[prime1, prime2])
  //2. Let n = p*q.
  Writeln('2. RSA Modul Public [p*q] as N: '+inttostr(setPublic))
  //3. less than n with no factors in common to n phi(n)=(p-1)(q-1)
  PrintF('3. Phi(N) Private: %d = %d*%d',
                                [getPHI_N,prime1-1,prime2-1])

  //4. Choose e <n; e is relative prime to phi(n).
  Writeln('4. Public RSA Exponent: '+inttostr(getEPublic))
  //5. Find d such that e*d mod phi(n)=1.
  Writeln('5. Private D: '+inttostr(getPrivate))

  //6. Set Public key (e,n), private key is (d, n).
  PrintF('6. Public (%d,%d) - Private (%d,%d)',
          [getEPublic,setPublic,getPrivate,setPublic])
  Writeln('');

function setPublic: integer;
begin
  result:= prime1 * prime2;
end;

function getPHI_N: integer;
begin
  result:= (prime1-1) * (prime2-1);
end;

function getEPublic: integer;
begin
  result:= RSA_exponent; //round(Power(2,4)+1);
end;
```

[3]OpenSSL actually produces a public - private key pair.

```
//Find d such that: e*d mod phi(n) = 1. (as inverse modulo)!
//more than one d is possible in loop
function getPrivate: integer;
begin
  for it:= 1 to 1000 do //or n depends on bigint
    if (getEPublic * it mod getPHI_N = 1) then begin
      //writeln(itoa(it))
      result:= it;
    end
end;
```

Explanation:
Alice lets her **public** key be known **to** everyone, but keeps the **private** key secret. Bob may send a confidential **message to** Alice like this:

1. B gets A's public key (you can get it from web).
2. B encrypts a **message with** A's public key, and sends it.
3. A decrypts a **message with** her **private** key.

You see in our example the public and private is:

```
Public (3,1081) - Private (675,1081)
```

At least we take a look at the big numbers.
Generating the public and private keys.
Pick two large prime numbers, p and q. Let n=pq. Typically, n is a number which in binary is written with 1024 bits (in decimal, that's about wide 308 digits).
Pick e relatively prime to (p-1)(q-1) like Euler said. Now find d such that ed=1 mod (p-1)(q-1)[4]. You can use Euclid's algorithm to find this d. The pair of numbers (e, n) is the public key. The pair of numbers (d, n) is the private key. The two primes p,q are no longer needed, and can be discarded, but should never be revealed. E is most of the time the same number: Exponent (24 bits): 65537

```
procedure encryptMessage(amess: string; acipher: TStrings);
var k,l,i: integer; //int64;
    pst: string;
begin
  for i:= 1 to length(amess) do begin
    l:= ord(amess[i])-OFFSET;
    //write(inttostr(l)+ 'd ') //debug
    pst:= PowerBig(l,getEPublic);
    k:= modbig(pst,setPublic);
    acipher.add(intToStr(k))
  end;
end;
```

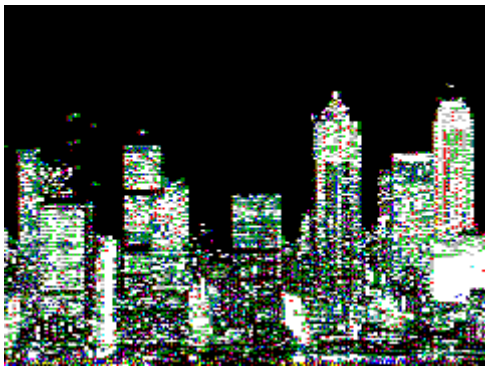
[4] e*d mod (p-1)(q-1) = 1

```
function decryptMessage(alist: TStrings): string;
var k,i: int64;
    pst: string;
begin
  for i:= 0 to alist.count -1 do begin
    pst:= PowerBig(strtoint(alist[i]),getPrivate);
    k:= modBig(pst,setPublic);
    //k:= f mod setPublic;
    result:= result+ Chr(k+OFFSET);
  end;
end;
```

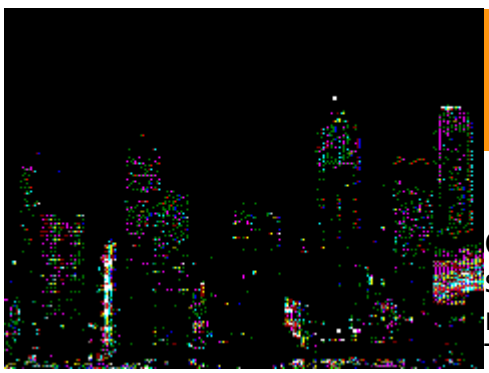
We need `powerBig` and `modBig` to deal with large numbers.
Conclusion:

```
The public key is (e,n), private key is (d, n).
Med mod n = M (this holds if e*d mod phi(n) = 1)
C=Me mod n
Me=C mod n
```

In real the message is divided into blocks, each block corresponding to a number less than n. For example, for binary data, the blocks will be (log2 n) bits. An exponentiation cipher (e) utilizing Euler's Theorem.



1.7 Beyond Big Numbers



Challenging domains such as robot-assisted search and rescue, operations in space[5] require humans to interact with robots. These interactions may be in the form of supervisory control, which connotes a high human involvement with limited robot automation (e.g., semi-autonomy, where the robot is truly

[5] twin paradox where one twin stays on Earth while the other travels into space

autonomous for portions of the task or mixed-initiative systems, where the robot and human are largely interchangeable.

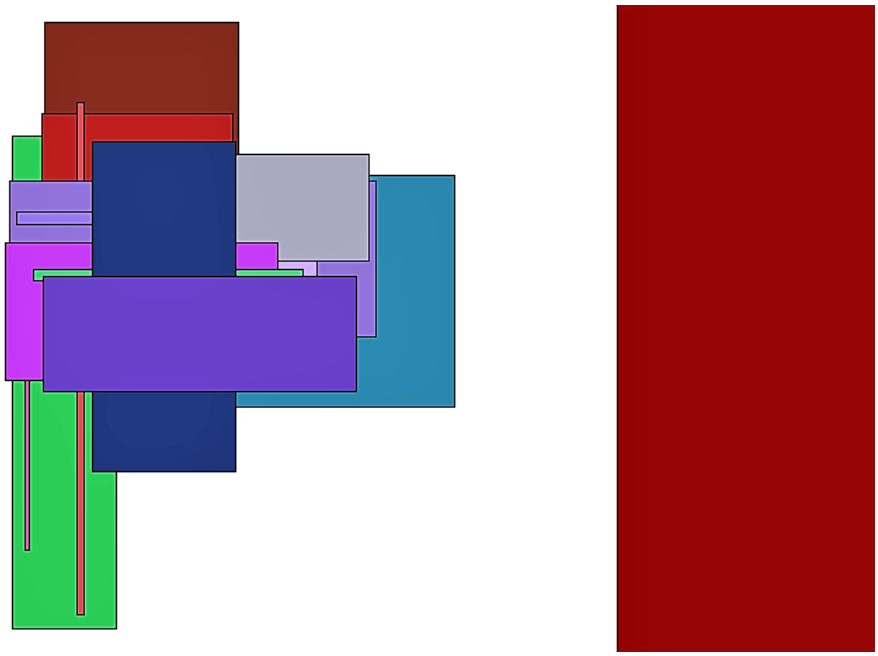
The interactions between humans and robots are often interchangeably referred to as “coordination” or “collaboration”.
The application of a script to interleave human and robot coordination is both logical and natural. Scripts simplify the relationship between human and robot making the task comprehensible to both novice and expert system users.

```
procedure StartBtnClick(Sender: TObject);
{User clicked start}
var  i:integer;
  drawtime:integer;
  startcount,stopcount:int64;
begin
  for i:= 2 to count do begin
    {put center of robo on the point}
    Robo.left:= saved[i].x-Robo.width div 2;
    Robo.top:= saved[i].y-Robo.height div 2;
    queryperformanceCounter(startcount);{Get time before we repaint}
    application.processmessages;
    queryPerformanceCounter(stopcount);{Get time after repaint}
    drawtime:= (stopcount-startcount) div freq;  {Compute ms to repaint}
    writeln('test freq ms: '+itoa(drawtime))
    sleep(max(0,sleepms-drawtime));   {wait time is left, if any}
  end;
end;
```

The ability to simplify a task as a series of simple steps is necessary for this comprehension. The available or possible actions for the human operator at any particular time are clear in the script because of the GUI. This approach can be applied to any task, level of man machine coordination. Next are two robots algorithm:

```
procedure StartBtnClick2(Sender: TObject);
{User clicked start both robots}
var  i, drawtime, maxcount:integer;
  startcount,stopcount:int64;
begin
  maxcount:= max(count, count2)
  for i:= 2 to maxcount do begin
    if i < count then begin
      Robo.left:=saved[i].x-Robo.width div 2;
      Robo.top:=saved[i].y-Robo.height div 2;
    end;
    if i < count2 then begin
      Robo2.left:=saved2[i].x-Robo2.width div 2;
```

```
      Robo2.top:=saved2[i].y-Robo2.height div 2;
    end;
     queryperformanceCounter(startcount);{Get time before repaint}
     application.processmessages;
     queryPerformanceCounter(stopcount);{Get time after repaint}
     drawtime:=(stopcount-startcount) div freq;
     {Compute ms to repaint}
     writeln('robo both freq ms: '+itoa(drawtime))
     sleep(max(0,sleepms-drawtime));
     {wait whatever time is left, if any}
  end;
end;
```



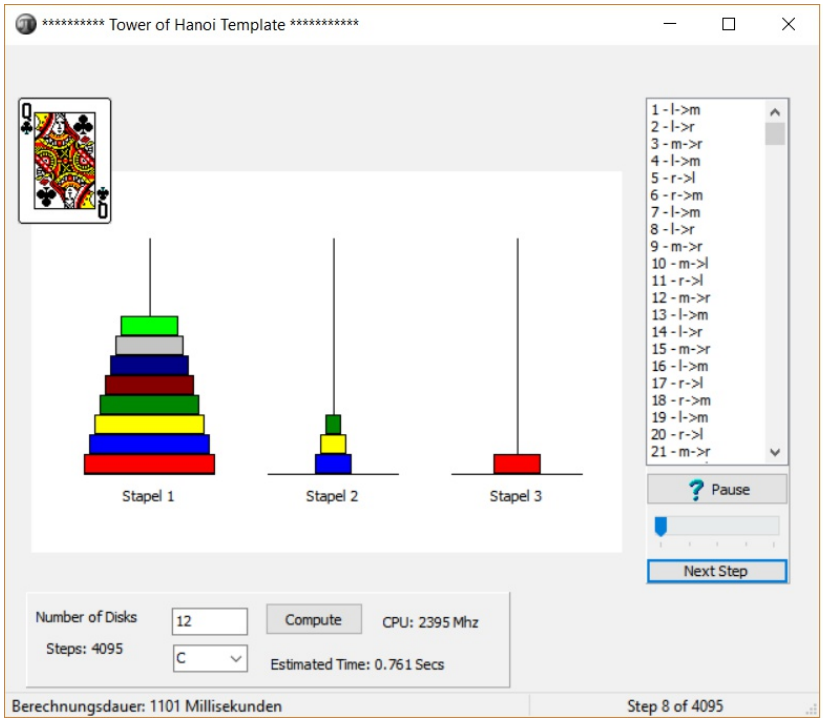
Due to the highly proprietary nature of robot software, most producers of robot hardware also provide their own software and firmware. While this is not unusual in other automated control systems, the lack of standards of programming methods for robots does pose certain challenges.

For example, there are over 30 different manufacturers of industrial robots like Kuka (now china!), so there are also 30 different robot programming languages required.
Fortunately, there are enough similarities between the different robots that it is possible to gain a broad-based understanding of robot programming without having to learn each manufacturer's proprietary language.

Another interesting approach is worthy of mention. All robotic applications need or explore parallelism and event-based programming.
Robot Operating System is an open-source platform for robot programming using Python, CPascal and C++. Java, Lisp, Lua, Pascal and Pharo are supported but still in experimental stage.

https://en.wikipedia.org/wiki/Robot_Operating_System

Another example is the tower of hanoi which can be solved for example by Lego mindstorm or other Arduino frameworks and can produce large numbers to resolve:



The script is called: `examples\712_towerofhanoi_animation.pas`

Programming errors represent a serious safety consideration, particularly in large industrial robots. The power and size of industrial robots mean they are capable of inflicting severe injury if programmed incorrectly or used in an unsafe manner. Due to the mass and high-speeds of industrial robots, it is always unsafe for a human to remain in the work area of the robot during automatic operation.

Two concepts should improve this safety:

- Console Capture
- Exception Handling

1.8 Console Capture DOS Big Decimals

I'm trying to move a part of SysTools to Win64. There is a certain class `TStDecimal` which is a fixed-point value with a total of 38 significant digits. The class itself uses a lot of ASM code.

```
function BigDecimal(aone: float; atwo: integer): string;
begin
  with TStDecimal.create do begin
    try    //assignfromint(aone)
```

```
    assignfromfloat(aone) //2
    RaiseToPower(atwo)  //23
    result:= asstring
  finally
    free
  end;
 end;
end;
```

But then I want to test some Shell Functions on a DOS Shell or command line output. The code below allows to perform a command in a DOS Shell and capture it's output to the maXbox console. The captured output is sent “real-time” to the `Memo2` parameter as console output in maXbox:

```
    srlist:= TStringlist.create;
      ConsoleCapture('C:\', 'cmd.exe', '/c dir *.*',srlist);
      writeln(srlist.text)
    srlist.Free;
```

But you can redirect the output `srlist.text` anywhere you want.
For example you can capture the output of a DOS console and input into a textbox, or you want to capture the command start of demo app and input into your app that will do further things.

```
ConsoleCapture('C:\', 'cmd.exe', '/c ipconfig',srlist);
ConsoleCapture('C:\', 'cmd.exe', '/c ping 127.0.0.1',srlist);
```

☝ It is important to note that some special events like `/c java -version` must be captured with different parameters like /k or in combination.

Here's the solution with `GetDosOutput()`:

```
writeln('GetDosOut: '+GetDosOutput('java -version','c:\'));
```

or like `powercfg` or the man-pages in Linux

```
writeln('GetDosOut: '+GetDosOutput('help dir','c:\'));
GetDosOutput('powercfg energy -output
                        c:\maxbox\osenergy.htm','c:\')
```

1.9 Exception Handling

A few words how to handle Exceptions within maXbox script catches:
Procedure Prototype:

```
procedure RaiseException(Ex: TIFException; const Msg: String);
```

Description: Raises an exception with the specified catch message.

Unexpected Exception Example:

```
begin
  RaiseException(erCustomError,'Your message goes here');
// The following line will not be executed because of the
exception!
  MsgBox('You will not see this.', 'mbInformation', MB_OK);
end;
```

This is a simple example of a actual script that shows how to do try except with raising a exception and doing something with the exception message.

```
procedure Exceptions_On_maXbox;

var filename,emsg:string;
begin
   filename:= '';
   try
     if filename = '' then
       RaiseException(erCustomError,
                    'Exception: File name cannot be blank');
   except
     emsg:= ExceptionToString(ExceptionType,ExceptionParam);

       //do act with the exception message i.e. email it or
       //save to a log etc

     writeln(emsg)
   end;
end;
```

☝ The `ExceptionToString()` returns a message associated with the current exception. This function with parameters should only be called from within an except section.

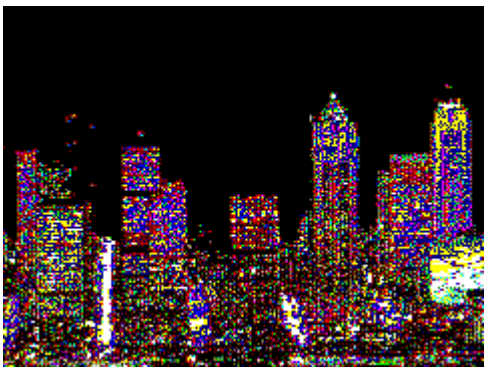
After completing the tasks as directed, the compiler proceeds to its second step where it checks for syntax errors (violations of the rules of the language) and converts the source code into an object code that contains machine language instructions, a data area, and a list of items to be resolved when he object file is linked to other object files.

At least there are two ways to install and configure your box into a directory you want. The first way is to use the unzip command-line tool or IDE, which is discussed above.
That means no installation needed. Another way is to copy all the files to navigate to a folder you like, and then simply drag and drop another scripts into the /examples directory.
The only thing you need to backup is the ini file **`maxboxdef.ini`** with your history or another root files with settings that have changed.

A namespace is a logical unit grouping of types (mostly to avoid name collisions). An assembly can contain many types in multiple namespaces (`System.DLL` contains a few...), and a single namespace can be spread across assemblies (e.g. `System.Threading`).

- An assembly provides a fundamental unit of physical code groups.
- A namespace provides a fundamental unit of logical code grouping.

1.10 Crypto Libraries with BigInt



With the namespace `System.Security.Cryptography` you do have access to huge numbers too:

```
System.Object
  System.Security.Cryptography.SymmetricAlgorithm
    System.Security.Cryptography.Aes
      System.Security.Cryptography.AesCryptoServiceProvider
```

This overview provides a synopsis of the encryption methods and practices supported by the .NET Framework, including the ClickOnce manifests, Suite B, and Cryptography Next Generation (CNG) support introduced in the .NET Framework 3.5.

This overview contains the following sections:

- Cryptographic Primitives
- Secret-Key Encryption
- Public-Key Encryption
- Digital Signatures
- Hash Values
- Random Number Generation
- ClickOnce Manifests
- Suite B Support
- Related Topics

You simply put lines with namespaces in a boot script and make sure the ini file has it set to `Yes`. BOOTSCRIPT=Y //enabling load a boot script

1.11 Image Processing

You maybe know the term picture manipulator?
It is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image.
A sepia tone is a reddish brown monochrome tint. When applied to a photo, it gives the picture a warm, antique feeling.
In our case we show a filter to an image with that sepia effect, this is an very quick and easy technique on the base of pixels:

```
function BmptoSepia(const bmp: TBitmap; depth:Integer; apath: string):
                                                              Boolean;
   var
   color,color2:longint;
   r,g,b,rr,gg:byte;
   h,w:integer;
   begin
     result:= false;
     for h:= 0 to bmp.height do begin
       for w:= 0 to bmp.width do begin
       color:=colortorgb(bmp.Canvas.pixels[w,h]);
       r:=getrvalue(color);
       g:=getgvalue(color);
       b:=getbvalue(color);
       color2:=(r+g+b) div 3;
       bmp.canvas.Pixels[w,h]:=RGB(color2,color2,color2);
       color:=colortorgb(bmp.Canvas.pixels[w,h]);
       r:=getrvalue(color);
       g:=getgvalue(color);
       b:=getbvalue(color);
       rr:=r+(depth*2);
       gg:=g+depth;
       if rr <= ((depth*2)-1) then
       rr:=255;
       if gg <= (depth-1) then
       gg:=255;
       bmp.canvas.Pixels[w,h]:=RGB(rr,gg,b);
       end;
     end;
     bmp.savetofile(apath)
     result:= true;
   end;
```

It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image.

Digital processing techniques help in manipulation of the digital images by using computers, but beware of fake news.

https://sourceforge.net/projects/maxbox/files/Examples/13_General/760_systeminfo_tester2.txt/download

 "Wise men speak: Hand made by robots.

Feedback @ max@kleiner.com

Literature: Kleiner et al., Patterns konkret, 2003, Software & Support

https://github.com/maxkleiner/maXbox4/releases

https://en.wikipedia.org/wiki/Robot_software

EKON 21 Cologne 2017 Input for Robots, Big Numbers & Components

1. AsyncPRO,	2. BigInteger,
3. Hotlog,	4. WMILib,
5. StBarCode,	6. XMLUtils,
7. LockBox,	8. cX509Certificate,
9. OpenGL,	10. WaveUnit,
11. OpenSSL,	12. Kronos,
13. HiResTimer,	14. Kmemo,
15. BigDecimals,	16. SynEdit,
17. SFTP,	18. Sensors,
19. PasScript,	20. ALJSON
21. CryptographicLib	22. RSA_Engine
23. cHugeInt.pas	24. cCipherRSA;
25. RFIDLib	26. Visuino Framework
27. QRCodeLib	28. MQTT[6]_UnitBox
29. XmlRpcCommon	30. VoiceRecognition

[6] Message Queue Telemetry Transport

1.12 Appendix External links of RSA & Big Numbers

- "*The Basics - Robot Software*". Seattle Robotics Society.
- G.W. Lucas, "*Rossum Project*".

- "*Mobile Autonomous Robot Software* (MARS)". Georgia Tech Research Corporation.
- "*Tech Database*". robot.spawar.navy.mil.
- Adaptive Robotics Software at the Idaho National Laboratory
- A review of robotics software platforms Linux Devices.
- ANSI/RIA R15.06-1999 American National Standard for Industrial Robots and Robot Systems - Safety Requirements (revision of ANSI/RIA R15.06-1992)

https://www.academia.edu/31097592/Work_with_RSA_maxbox_starter47.pdf

http://www.softwareschule.ch/download/maxbox_starter47.pdf

https://www.academia.edu/31112544/Work_with_microservice_maXbox_starter48.pdf

1.13 References

In the technology world, your use cases are only as effective as the value someone's deriving from them. What seems obvious to you may not be to your developers or customers. The success measurement for an effective written use case is one that is easily understood, and ultimately the developers can build the right product the first time.

By absorbing the meaning of use case diagrams, alternate flows and basic flows, you will be able to apply use cases to your projects. Developing use cases should be looked at as an iterative process where you work and refine.

After producing your initial visual list of use case actors and goals, we can take this list and create an initial use case grid which provides the basis for the use case index[7].

Creating a use case to long winded with too many features can potentially put a product at risk. What happens is that you can extend your release to market from two weeks to several months without the ability to learn from the iteration and adapt to the market. Keep those use cases leaner!

Examples of Tutorial Use Case 5 and maXbox 4.2.5.10:

www.softwareschule.ch/examples/210_public_private_cryptosystem5_ibz_herdt2.txt

Examples of Use Case Example Routines 1-4:

http://www.instructables.com/id/How-to-Use-the-Adafruit-BMP280-Sensor-Arduino-Tuto/

https://www.gatherspace.com/static/use_case_example.html

http://www.xmlfiles.com/examples/

[7] these attributes include scope, link, complexity, status and priority

https://www.w3schools.com/xml/xml_attributes.asp

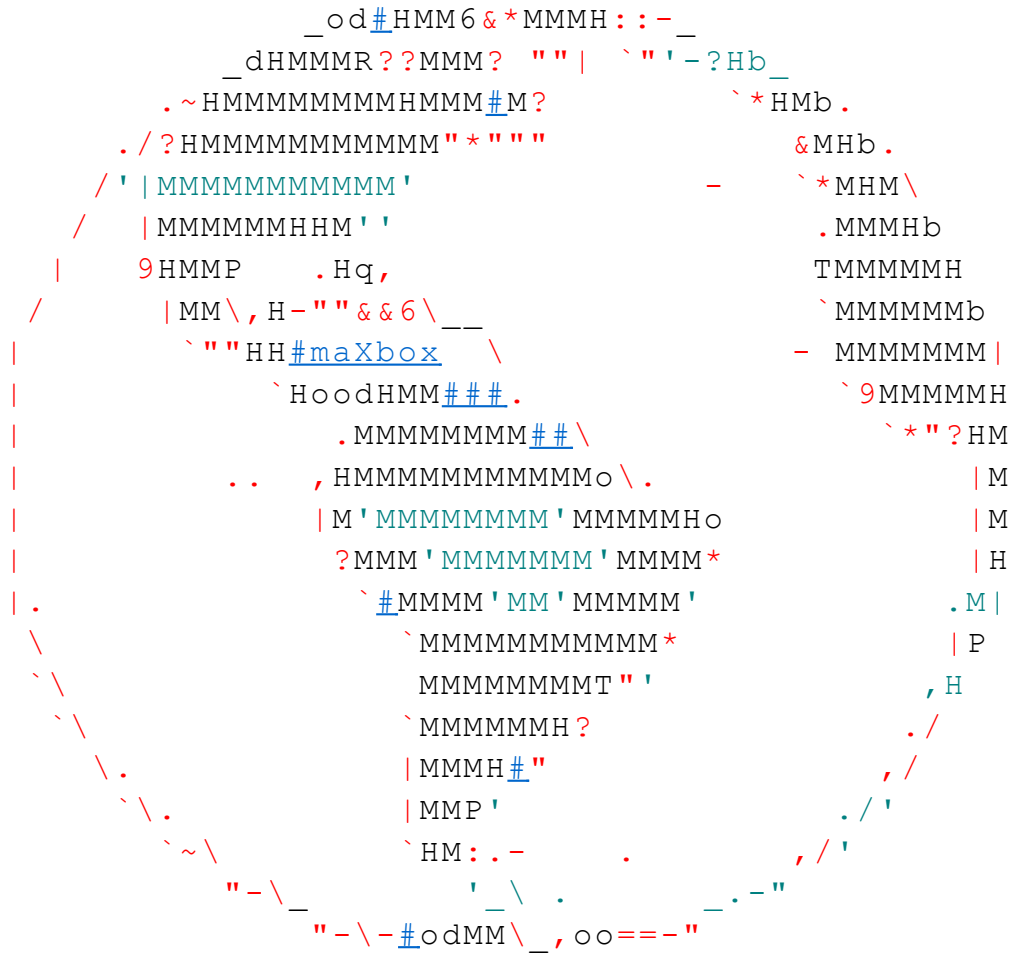
http://prll.sourceforge.net/shell_parallel.html

http://www.softwareschule.ch/download/maxbox_starter50.pdf

OpenSSL explain:

http://stackoverflow.com/questions/5244129/use-rsa-private-key-to-generate-public-key

- *O. Nnaji, Bartholomew. Theory of Automatic Robot Assembly and Programming (1993 ed.). Springer. p. 5. ISBN 978-0412393105. Retrieved 8 February 2015.*
- *"Robot programming languages". Fabryka robotów. Retrieved 8 February 2015.*
- https://www.madboa.com/geek/openssl/#key-rsa



Zwei Worte werden Dir im Leben viele Türen öffnen - "ziehen" und "stoßen".

http://www.softwareschule.ch/maxboxnews.htm

http://maxbox.codeplex.com/

```
procedure RecordWave(Sender: TObject);
   // Record  to wav
   begin
     mciSendString('OPEN NEW TYPE WAVEAUDIO ALIAS mysound', 'nil',
                                          0, hinstance);
     mciSendString('SET mysound TIME FORMAT MS ' +   // set time
       'BITSPERSAMPLE 8 ' +                 // 8 Bit
       'CHANNELS 1 ' +                      // MONO
       'SAMPLESPERSEC 8000 ' +              // 8 KHz
       'BYTESPERSEC 8000',                 // 8000 Bytes/s
       'nil', 0, hinstance);
     mciSendString('RECORD mysound', 'Nil', 0, Hinstance)
   end;

     procedure StopButton2ClickRecord(Sender: TObject);
      // Stop recording
      begin
        mciSendString('STOP mysound', 'Nil', 0, hinstance)
      end;

     procedure StartButton3ClickRecord(Sender: TObject);
      // Start & Save
      var
        verz: String;
      begin
        GetDir(0, verz);
        mciSendString(PChar('SAVE mysound ' + verz + '/test.wav'),
                      'Nil', 0, HInstance);
        mciSendString('CLOSE mysound', 'Nil', 0, HInstance)
      end;
```

Max Kleiner, April 2017

```
  _____________________________________________________
|                 ___________________________________ |
| [1] [2]         ___________________________________ |
| [3] [4]   [_][_][_] [__][_][_][_] [_][_] [_][_][_] |
| [5] [6]   [_][][][][][][][][][][][][][][_] [1][2][3] |
| [7] [8]   [_][][][][][][][][][][][][][][O][] [4][5][6] |
| [9][10]   [__][A][][][][][][][][][][][][][_] [7][8][9] |
| [11][12] [___][][X][][][B][][][M][][][][_][_][0] [] |
|                 [_][________________][__]           |
|_____________________________________________________|
```