InterSim III Interface

Macro Manual

Contents

Use the macro language to create dynamic InterSim III scenarios. All settings and parameters of the simulator that can be accessed via the menus are also available via the macro language.

External Documentation

The Macro functionality of InterSim III is based on the scripting language FastScript. If you want to learn more about FastScript, you can find the documentation at
https://www.fast-report.com/en/product/fast-script/documentation/

The Macros within InterSim III use only PascalSript syntax. Please note that not all functions mentioned in the FastScript documentation are available. Pay special attention to the topics “Language reference,” “script structure,” “Data types,” and “Functions”. Much of the other information in the document is not relevant to InterSim III users.

Operation

Macro Menu

You will find the basic functions for working with macro files in the Macro menu.

New
The New Menu Item shows the Macro Window. A frame for a new macro will be created. If the window currently contains an unsaved macro, a dialog will appear, allowing the macro to be saved if desired.

Open
The menu item opens the Open Macro Dialog. If the Macro Window currently contains an unsaved macro, a dialog will appear, allowing the macro to be saved if desired.

Save
The Save menu item saves a macro. If the macro has not been saved before, Save As is called instead.

Save As
The Save As menu item opens the Save Macro Dialog.

Reopen
Click Reopen to open the list of recently saved or loaded macro files. Click the filename of the macro you want to

load again.

Run
Click this menu item to start a macro. The macro will not start if it contains a syntax error. In this case the dot at the left of the Macro Window shows the error position and the error description is shown in the message area of this window.

Step
It is possible to execute a macro step by step by pressing this menu item.

Pause
The Pause menu item will interrupt a running macro.

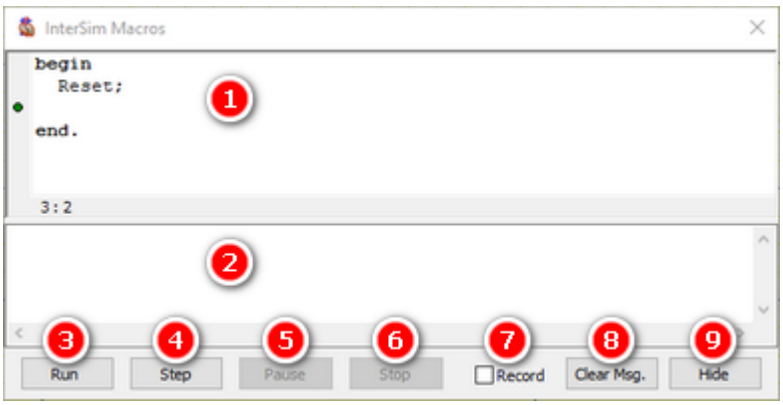
Stop
The Stop menu item terminates a long or endlessly running macro.

Show
The Show Macro menu item shows the Macro Window if it is hidden. If this window has not been shown before, the New menu item is called instead.

Hide Macro
The Hide Macro menu item hides a visible Macro Window. The macro itself will not be affected. Use this command if it is not necessary to watch the macro execution or to temporarily hide the window for some reason. The Show Macro menu item will display the Macro Window again.

Macro Window

The Macro Window will appear after clicking New or Show in the Macro menu and also after opening a macro via Open or by selecting a filename in the Recent Macros list. It provides fundamental functions for working with macros.



Macro Area
The Macro Area shows the currently loaded or created macro. It provides the necessary functions to enter or edit a Macro. The small green dot to the left of the text shows the position at which a new command will be inserted using the Record function. The dot turns red when the macro is running. In this case, the dot shows the command that is just executed.

Message Area
This area shows the macro message output. Use Clear Msg. to clear all messages.

 **Run**
Press this button to start a macro. The macro will not start if it contains a syntax error. In this case the dot at the left marks the error position and the error description is shown in the message area.
In Classroom mode, the Trainer transmits the Macro to all Trainees' PCs and starts it by pressing the Run button.

 **Step**
It is possible to execute a macro step by step by pressing this button.

 **Pause**
The Pause button will interrupt a running macro.

 **Stop**
The Stop button terminates a long or endlessly running macro.
Even if no program is running on the trainer's PC in Classroom mode, possibly running programs on the trainees' PCs can be terminated.

 **Record**
Check this box to start the recording function. Every command taken or every parameter changed will be recorded by this function. The green dot on the left indicates the position where the next entry will be inserted.

 **Clear Msg.**
Use this to clear the message area.

 **Hide**
This hides the window. The macro itself will not be affected. Use this button if it is not necessary to watch the macro execution or to temporarily hide the window for any reason. The Show Macro menu item will display the Macro Window again.

Example

Steps to create a first macro:

1. Open the Macro menu, click New
2. Check the Record checkbox
3. Open the Rhythms menu, click SinusTachy
4. Click on the fast increase button for the atrial rate in the Parameters box (double up arrows)
5. Open the Parameters menu, submenu Amplitudes
   Choose a RA amplitude of 1.2 mV
6. Uncheck the Record checkbox

Your macro should look like this (the wait values will differ):

```
begin
  Reset;
  SinusTachy;
  Wait(5000);
  Set_SinusRate(130);
  Wait(5000);
  Set_AmplitudeAtrium(am0120mV);
end.
```

Data Types

The FastScript language offers many built-in data types. Only a few of them are necessary for InterSim III users.

Boolean

The Boolean data type has two possible values: True and False.

Char

The character data type represents a numeric (' 0' - '9'), an alphanumeric (' a' - 'z', 'A' - 'Z'), or a special character (':', '_', '+', ..).

Double

The Double data type is used for floating-point numbers. The range is from ±2.23 x 10^{-308} to ±1.79 x 10^{+308}.

Integer

The Integer data type is used for signed integer numbers. The range is from -2147483648 to 2147483647.

Int64

The second integer data type is used for 64bit signed integer numbers. It is only used by the function SimulationTime. The range is from -9223372036854775808 to 9223372036854775807.

String

The String data type holds a text string. Note that the first character in string is referred to as position 1 (in contrast to arrays which use 0 as origin).

Assignment

You can assign a value to a variable with an assignment operator **:=** . Using this operator, the value on the right side will be assigned to the variable on the left. The data types of the right and left sides must match.

Example:

```
var
  A: Integer;
  B: Double;
  S: string;
  V: Integer;
begin
  // assign a constant value
  A := 1;
  // assign a result of an operation
  B := 5 * 7.2;
  // assign a constant string
  S := 'test';
  // assign a concatenation of two constant strings
  S := 'test' + ' assignment';
  // assign a return value of a function
  V := Get_SinusRate;
end.
```

Variables

A variable holds a value of a given data type. You may declare as many variables as needed. Variables are not case sensitive.

Example:

```
var
  B: Boolean;
  I: Integer;
  SimTime: Int64;
  APVolt: Double;
  Unit: Char;
  S: string;
begin
  // assign a Boolean value to a Boolean variable
  B := False;
  // assign an integer value to an integer variable
  I := Round(7.2);
  // get the simulation time
  SimTime:= SimulationTime;
  // get the last pace voltage
  APVolt := AtrialPaceEvent.Voltage;
  // assign a character to a char variable
  Unit := 'V';
  // create a string
  S := 'Atrial pace voltage ' + FloatToStr(APVolt) + Unit;
end.
```

Notes

The macro core has a low priority compared to the simulation core. This means that the execution time of one line cannot be reliably determined. A macro command can take between 5 and 20 ms.

Although the Wait function has a delay parameter that allows entering a value to the nearest 1 ms, do not expect to achieve a precise resolution of 1 ms. The shortest period one can get using the Wait function is about 5 ms. The same applies to the Timer and Stopwatch commands.

If it is necessary to compare times do not use the equal sign =. Instead, use Greater than or equal to >= respectively Lower than or equal to <=.

It is a good idea to alternate regular commands with Wait commands, if possible.

See Wait.

System Functions

Mathematics

function Abs(X: Double): Double;

The function Abs returns the non-negative value of the parameter X.

Example:

```
var
```

```
  N: Double;
begin
  N := -1;
  Message('Abs(-1) = ' + FloatToStr(Abs(N)));
end.
```

function ArcTan(X: Double): Double;

The function ArcTan returns the arc tangent value of the parameter X:

Example:

```
var
  N: Double;
begin
  N := -1;
  Message('ArcTan(-1) = ' + FloatToStr(ArcTan(N)));// = pi / 4
end.
```

function Cos(X: Double): Double;

The function Cos returns the cosine value of the parameter X.

Example:

```
var
  N: Double;
begin
  N := Pi;
  Message('Cos(pi) = ' + FloatToStr(Cos(N)));
end.
```

function Exp(X: Double): Double;

The function Exp returns the result of e^{X}.

Example:

```
begin
  Message('Exp(1) = ' + FloatToStr(Exp(1)));
end.
```

function Frac(X: Double): Double;

The function Frac returns the fractional part of the parameter X.

Example:

```
var
  N: Double;
begin
  N := 7.5 * 3.5;
  Message('Frac(7.5 * 3.5) = ' + FloatToStr(Frac(N)));
end.
```

function Int(X: Double): Double;

The Int function returns the integer part of the parameter X. The data type of the result is Double.

Example:

```
begin
  Message('Integer part of pi = ' + FloatToStr(Int(Pi)));
end.
```

See also the function Trunc.

function Ln(X: Double): Double

The function Ln returns the natural logarithm of the parameter X.

Example:

```
var
  N: Double;
begin
  N := 2.718281829;
  Message('ln(2.718281829) = ' + FloatToStr(Ln(N)));
end.
```

function Pi: Double

The function Pi returns the Number Pi.

Example:

```
begin
  Message('Pi = ' + FloatToStr(Pi));
end.
```

function Round(X: Double): Integer

The function Round returns the rounded integer value of the parameter X.

Example:

```
begin
  Message('Round(5.2) = ' + IntToStr(Round(5.2)));
  Wait(2000);
  Message('Round(5.7) = ' + IntToStr(Round(5.7)));
end.
```

function Sin(X: Double): Double

The function Sin returns the sine of the parameter X.

Example:

```
begin
  Message('Sin(Pi) = ' + FloatToStr(Sin(Pi)));
end.
```

function Sqrt(X: Double): Double

The function Sqrt returns the square root of the parameter X.

Example:

```
var
  N: Double;
begin
  N := 2 * 2;
  Message('Sqrt(2 * 2) = ' + FloatToStr(Sqrt(N)));
end.
```

function Tan(X: Double): Double

The function Tan returns the tangent of the parameter X.

Example:

```
begin
  Message('Tan(Pi) = ' + FloatToSTr(Tan(Pi)));
end.
```

function Trunc(X: Double): Integer

The function Trunc returns the integer part of the parameter X. The data type of the result is Integer.

Example:

```
begin
  Message('Trunc(1.5) = ' + IntToSTr(Trunc(1.5)));
  Wait(2000);
  Message('Trunc(-1.7) = ' + IntToSTr(Trunc(-1.5)));
end.
```

See also the function Int.

Conversion

function BoolToStr(B: Boolean): String;

The function BoolToStr converts a Boolean data type into a String data type. This function is useful in conjunction with the Message procedure that expects a string parameter.

Example:

```
begin
  Message('BoolToStr(False) = ' + BoolToStr(False));
  Wait(2000);
  Message('BoolToStr(True) = ' + BoolToStr(True));
end.
```

function FloatToStr(D: Double): String

The function FloatToStr converts a float data type (Single or Double) into a String data type. This function is useful in conjunction with the Message procedure that expects a string parameter.

Example:

```
begin
  // the procedure Message expects a string datatype
  Message('FloatToStr(1.245) = ' + FloatToStr(1.245));
end.
```

function IntToStr(I: Integer): String

The function IntToStr converts an integer number into a string. This function is useful in conjunction with the Message procedure that expects a string parameter.

Example:

```
begin
  // the procedure Message expects a string datatype
  Message('IntToStr(157) = ' + IntToStr(157));
end.
```

function StrToBool(const S: string): Boolean;

The function StrToBool converts a string into a boolean value. If parameter S does not contain a string that represents a valid boolean value an error will be thrown.

Valid boolean values are:
True, every number except 0
False, 0

Example:

```
var
  S: string;
  B: Boolean;
begin
  S := 'False';
  B := StrToBool(S);
  Message('StrToBool(''' + S + ''') = ' + BoolToStr(B));
  Wait(2000);
  S := 'True';
  B := StrToBool(S);
  Message('StrToBool(''' + S + ''') = ' + BoolToStr(B));
end.
```

function StrToFloat(S: String): Double

The function StrToFloat converts a string into a float number. If parameter S does not contain a string that represents a valid float number an error will be thrown.

Valid float numbers consist of an optional sign (+ or -), some digits, an optional decimal point with some following decimal places, and an optional exponent. The exponent consists of ‘E’ or ‘e’, an optional sign (+ or -), and some digits.

Example:

```
var
  S: string;
  D: Double;
begin
  S := '-1.5e-3';
  D := StrToFloat(S);
  Message('StrToFloat (''' + S + ''') = ' + FloatToStr(D));
end.
```

function StrToInt(S: String): Integer

The function StrToInt converts a string into an integer number. If parameter S does not contain a string that represents a valid integer number an error will be thrown.

Example:

```
var
  S: string;
  N: Integer;
begin
  S := '123';
  N := StrToInt(S);
  Message('StrToInt(''' + S + ''') = ' + IntToStr(N));
end.
```

function StrToInt64(S: String): Int64

The function StrToInt64 converts a string into an long integer number. If parameter S does not contain a string that represents a valid integer number an error will be thrown.

Example:

```
var
  S: string;
  N: Int64;
begin
  S := '123545456352';
  N := StrToInt64(S);
  Message('StrToInt64(''' + S + ''') = ' + IntToStr(N));
end.
```

String Routines

The string routines are primarily useful in combination with the Message procedure and the File routines.

function Chr(I: Integer): Char

The function Chr returns one character with the ANSI code I.

Excample:

```
begin
  Reset;
  // 65 is the code of "A"
  Message(Chr(65));
end.
```

function CompareText(S1, S2: String): Integer

The function CompareText compares two strings S1 and S2 for equality, ignoring case. If S1 and S2 are equal, CompareText returns 0. If S1 is greater than S2 (that means S1 comes behind S2 in an alphabetical order), CompareText returns an integer greater than 0. If S1 is less than S2, CompareText returns an integer less than 0.

Example:

```
begin
  Message(IntToStr(CompareText('John', 'John')));
  Wait(2000);
  Message(IntToStr(CompareText('Sophia', 'Emily') ));
  Wait(2000);
  Message(IntToStr(CompareText('Ella', 'Jack') ));
end.

The output of the macro is:
0
1
-1
```

function Copy(S: String; From, Count: Integer): String

The function Copy returns a part of string S, beginning at the position From with a length of Count characters.

Example:

```
begin
  Message(Copy('ABCDEFG', 3, 2));
end.

The output of the macro is:
CD
```

procedure Delete(var S: String; From, Count: Integer)

The function Delete deletes Count characters from the parameter S starting from the position From.

Example:

```
var
  S: string;
begin
  S := 'ABCDEFG';
  Message(S);
  Delete(S, 3, 2);
  Message(S);
end.

The output of the macro is:
ABCDEFG
ABEFG
```

procedure Insert(S1: String; var S2: String; Pos: Integer)

The procedure Insert inserts string S1 into string S2 at position Pos.

Example:

```
var
  S: string;
begin
  S := 'ABCDEFG';
  Message(S);
  Insert('XY', S, 4);
  Message(S);
end.

The output of the macro is:
ABCDEFG
ABCXYDEFG
```

function Length(S: String): Integer

The function Length returns the number of characters in the string parameter S.

Example:

```
var
  S: string;
begin
  S := 'ABCDEFG';
  Message(S);
  Message(IntToStr(Length(S)));
end.

The output of the macro is:
ABCDEFG
7
```

function Lowercase(S: String): String

The function Lowercase returns the string S. All uppercase alphanumeric characters are changed to lowercase.

Example:

```
var
  S: string;
begin
  S := 'ABCDEFG';
  Message(S);
  S := LowerCase(S);
  Message(S);
end.

The output of the macro is:
ABCDEFG
abcdefg
```

function Ord(Ch: Char): Integer

The function Ord returns the ordinal value (the ANSI code) of the character Ch.

Example:

```
begin
  Message(IntToStr(Ord('A')));
  Message(IntToStr(Ord('a')));
end.
```

```
The output of the macro is:
65
97
```

function Pos(Substr, S: String): Integer

The function Pos returns the position of substring Substr in string S. If S does not contain Subst the function returns 0.

Example:

```
begin
  Message(InttoStr(Pos('CD', 'ABCDEFG')));
end.

The output of the macro is:
3
```

procedure SetLength(var S: Variant; L: Integer);

Use the SetLength function primarily to set the number of elements in an array. It can also be used to set the length of a String. The procedure expects a an array or a string as first parameter. Use the second parameter to set the new length.

Example:

```
var
  Ar: array of Integer;
  I: Integer;
begin
  Reset;
  SetLength(Ar, 5);
  Ar[0] := 180;
  Ar[1] := 185;
  Ar[2] := 195;
  Ar[3] := 210;
  Ar[4] := 230;
  for I := 0 to 4 do
  begin
    Set_PRInterval(Ar[I]);
    Wait(3000);
  end;
end.
```

function Trim(S: String): String

The function Trim returns the string S without leading and trailing spaces.

Example:

```
var
  S: string;
begin
  S := ' abcd efg ';
  Message(S);
  S := Trim(S);
  Message(S);
end.

The output of the macro is:
 abcd efg
```

```
abcd efg
```

function Uppercase(S: String): String

The function Uppercase returns the string S. All lowercase alphanumeric characters are changed to uppercase.

Example:

```
var
  S: string;
begin
  S := 'abcdefg';
  Message(S);
  S := UpperCase(S);
  Message(S);
end.

The output of the macro is:
abcdefg
ABCDEFG
```

File Routines

It is possible to create and write text files with the means of macro commands. These files can be used for a later evaluation.

All files are stored in or loaded from the folder Userfiles of the User Data Folder.

procedure FileClose(FileHandle: Integer)

The procedure FileClose closes a File opened with FileOpen.

See the File Routines Example .

procedure FileDelete(FileName: string)

It is possible to delete a file with the procedure FileDelete. The function expects a filename as parameter. Only characters and numbers are allowed in a filename.

function FileOpen(FileName: string; Mode: string = ''W''): Integer

It is necessary to open a file before using it. Use the function FileOpen for this purpose. The function expects a filename as first parameter. Only characters and numbers are allowed in a filename.

The second parameter is optional. Omit the parameter or use W to open the file for writing. Use R to open the file for reading.

If you have opened the file for writing and the file **exists** already, the new content will be appended. If the file **does not exist,** a new file will be created.

If you want to open the file for reading, the file must exist.

The return parameter of the function is a file handle.

See the File Routines Example .

procedure FileRead(FileHandle: Integer; var Line: string)

The procedure FileRead reades a line from the file FileHandle. An empty string is returned after end of file.

See the File Routines Example .

procedure FileWrite(FileHandle: Integer; Content: string)

The procedure FileWrite writes a new line of Content to the file FileHandle.

See the File Routines Example .

File Routines Example

```
var
  FH: Integer;
  I: Integer;
  S: string;
begin
  Reset;
  // delete the file if it already exists
  FileDelete('Testfile1');
  // create a new file
  FH := FileOpen('Testfile1');
  // write the start of the macro
  FileWrite(FH, 'Start Macro at ' + IntToStr(SimulationTime) +' ms');
  // set the sinus rate
  Set_SinusRate(60);
  // a for loop with 5 repetitions
  for I := 1 to 5 do
  begin
    // wait for an atrial event
    while not AtrialSenseEvent.Active do
      Wait(5);
    // write the time of the event into the file
    FileWrite(FH, IntToStr(AtrialSenseEvent.EventTime) +' ms');
    // clear the event
    AtrialSenseEvent.Clear;
  end;
  // close the file
  FileClose(FH);

  // open the file for reading
  FH := FileOpen('Testfile1', 'R');
  for I := 1 to 6 do
  begin
    // read a line of the file
    FileRead(FH, S);
    // write it to the message window
    Message(S);
  end;
  // close the file
  FileClose(FH);
end.
```

Other

procedure Dec(var I: Integer; Decr: Integer = 1)

The procedure Dec decrements the variable I by the value Decr. If the optional value Decr is omitted the procedure decrements I by 1.

Example:

```
var
  A: Integer;
begin
  A := 10;
  Message(IntToStr(A));
  Dec(A);
  Message(IntToStr(A));
end.

The output of the macro is:
10
9
```

procedure Inc(var I: Integer; Incr: Integer = 1)

The procedure Inc increments the variable I by the value Incr. If the optional value Incr is omitted the procedure increments I by 1.

Example:

```
var
  A: Integer;
begin
  A := 10;
  Message(IntToStr(A));
  Inc(A);
  Message(IntToStr(A));
end.

The output of the macro is:
10
11
```

function Random: Double

The function Random returns a random float value between 0 (included) and 1 (not included). You can create discrete integer values with the following example.

Example:

```
var
  R: Double;
  V: Integer;
begin
  Reset;

  R := Random;
  // delivers an integer value between 0 and 9
  V := Trunc(R * 10);
```

```
  Message(IntToStr(V));
end.
```

function ValidFloat(S: string): Boolean;

Use the function ValidFloat to check whether a string contains a valid float number.

function ValidInt(S: string): Boolean;

Use the function ValidFloat to check if a string contains a valid integer number.

Formatting

function Format(Fmt: String; Args: array): String

The Format function converts simple data types into a string. The format string Fmt defines how the parameter array Args is used to build the returned string. The format string can include a mix of ordinary characters (that are passed unchanged to the result string), and data formatting characters. In simple terms, each data formatting substring starts with a % and ends with a Type indicator:

d = Decimal (integer)

f = Fixed

s = String

x = Hexadecimal

The general syntax of a formatting substring is as follows:

%[Index:][-][Width][.Precision]Type

The square brackets refer to optional parameters. The : . - characters are literals and identify the optional arguments (description inspired by www.delphibasics.co.uk).

Example:

```
begin
  Reset;
  // %10d - format the first parameter SimulationTime as decimal type with 10 digits
  // %s - format the second parameter 'ms' as string type
  Message(Format('%10d %s since simulator start', [SimulationTime, 'ms']));
  Wait(5000);
  // %d - format the first parameter GetSinusRate as decimal type
  // %.0f - format the second parameter60000 / Get_SinusRate as float type with 0 deimal
places
  Message(Format('Sinus rate = %d bmp, interval = %.0f ms',
    [Get_SinusRate, 60000 / Get_SinusRate]));
end.
```

function FormatFloat(Fmt: String; Value: Double): String

The FormatFloat function provides extensive means for formatting a floating point number Value into a string. The format string Fmt may contain a mix of freeform text and control characters:

0 Forces digit display or 0

Optional digit display

, Forces display of thousands

.	Forces display of decimals
E+	Forces signed exponent display
E-	Optional sign exponent display
;	Separator of positive, negative and zero values

(The description was taken from www.delphibasics.co.uk)

Example:

```
var
  Number : double;
begin
  Number := 1234.567;
  Message(FormatFloat('#,##0', Number));
end.
```

General

function Input(Caption: String): string

The Input function is used to make inputs in a running macro with an input window.

Example:

```
var
  S: string;
begin
  S := Input('Enter a number');
  Message(S);
end.
```

procedure Message(Text: string)

The procedure Message shows the string Text in the message window.

Example:

```
begin
  Message('This message appears in the message window);
end.
```

function SimulationTime: Int64

The function SimulationTime returns the current simulation time since the start of the application in milliseconds. The range extends theoretically from 0 to 9,223,372,036,854,775,807 ms.

procedure Wait(TimeToWait: Integer)

The procedure Wait waits for approximately TimeToWait milliseconds. The shortest period one can get using the Wait function is about 5 ms.

See Notes.

Stopwatches and Timers

There are up to 10 Stopwatches and up to 10 Timers available.

Stopwatch

Stopwatch0 to Stopwatch9 can be used for general purposes.

function Stopwatchx.Elapsed: Integer
Call this function to get the elapsed time in ms while the Stopwatch is running.

function Stopwatchx.IsRunning: Boolean
Call this function to determine whether the Stopwatch is currently running.

procedure Stopwatchx.Reset
Resets the elapsed time to 0 and stops the watch.

procedure Stopwatchx.Start
Starts a stopwatch. Resumes execution if the Stopwatch has stopped in the meantime.

procedure Stopwatchx.Stop
Stops a Stopwatch.

Example:

```
// this macro shows the usage of StopWatches
// there are 10 StopWatches: StopWatch0, StopWatch1, etc.
begin
  Reset;
  // the StopWatch begins to run with Start
  StopWatch0.Start;
  // you can query whether the StopWatch is currently running
  if StopWatch0.IsRunning then
    Message('StopWatch0 is running');
  // you can stop the StopWatch
  StopWatch0.Stop;
  Wait(1000);
  // and start the StopWatch again
  StopWatch0.Start;
  Wait(1000);
  // you can query the elapsed time
  Message('Elapsed time: ' + IntToStr(StopWatch0.Elapsed));
  // reset the StopWatch if you want to reuse it
  StopWatch0.Reset;
end.
```

Timer

Timer0 to Timer9 can be used for general purposes.

function Timerx.Expired: Boolean
Call this function to determine whether Timerx has expired.

Timerx.Interval: Integer
The variable Interval holds the current timer interval in ms. Set the interval before using a timer.

function Timerx.Remaining: Integer
Call this function to determine the remaining interval.

procedure Timerx.Reset

Stops a running timer. Sets the elapsed time to 0.

procedure Timerx.Start
Starts a timer. Resumes execution if the timer has stopped in the meantime.

procedure Timerx.Stop
Stops a timer.

Example:

```
// this macro shows the usage of timers
// there are 10 timers: Timer0, Timer1, etc.
begin
  Reset;
  // set the interval in ms the Timer0 should use
  Timer0.Interval := 2000;
  // start the Timer0
  Timer0.Start;
  // use a loop until the Timer0 expires
  while not Timer0.Expired do
  begin
    // while the Timer0 is running you can stop it
    Timer0.Stop;
    Wait(10);
    // and start it again
    Timer0.Start;
    // query the remaining time
    Message(IntToStr(Timer0.Remaining));
  end;
  // reset the Timer0 if you want to reuse it
  Timer0.Reset;
end.
```

Simulator Procedure and Functions

General

procedure Reset

The procedure Reset resets all rhythms, blocks, intervals and other simulator parameters to the defaults. It is recommended to use Reset as the first command within a macro, so that the macro will always start in a well-defined state.

Rhythms

procedure AtrialFibrillation

The procedure AtrialFibrillation starts an atrial fibrillation.

procedure AtrialFlutter

The procedure AtrialFlutter starts an atrial flutter rhythm with a rate of 230 bpm and a block rate of 184 bpm. Use the procedure Set_BlockRate to change the conduction rate. Use the procedure Set_SinusRate to change the flutter rate.

procedure AtrialFlutter2_1

The procedure AtrialFlutter starts an atrial flutter rhythm with a predefined Block Rate (conduction rate) of 184 bpm.

procedure AtrialFlutter3_1

The procedure AtrialFlutter starts an atrial flutter rhythm with a predefined Block Rate (conduction rate) of 103 bpm.

procedure AtrialFlutter4_1

The procedure AtrialFlutter starts an atrial flutter rhythm with a predefined Block Rate (conduction rate) of 69 bpm.

procedure AVNodeReentryTachy

The procedure AVNodeReentryTachy starts a AV nodal reentrant tachycardia with a rate of the AV node of 180 bpm with a predefined Block Rate (conduction rate) of 237 bpm. The retrograde conduction is activated.

procedure BradyTachySyndrome

The procedure BradyTachySyndrome starts a brady-tachy syndrome. The rhythm changes in a random manner between SinusRhythm, SinusBrady, SinusArrest, and AtrialTachy.

procedure CombinedAFibAFlut

The procedure CombinedAFibAFlut starts a combined atrial flutter and fibrillation rhythm.

procedure IdioventricularRhythm

The procedure IdioventricularRhythm starts an idioventricular rhythm. Sinoatrial node and atrioventricular node stop discharging impulses. A retrograde conduction is turned on. The left ventricle will generate an escape rhythm of 30 bpm.

procedure LeftVentricleTachyFast

The procedure LeftVentricleTachyFast starts a left ventricular tachycardia with a rate of 220 bpm.

procedure LeftVentricleTachyMedium

The procedure LeftVentricleTachyMedium starts a left ventricular tachycardia with a rate of 165 bpm.

Alternative name:
procedure LeftVentricleTachy;

procedure LeftVentricleTachySlow

The procedure LeftVentricleTachySlow starts a left ventricular tachycardia with a rate of 130 bpm.

procedure LeftVentricleTachyVeryFast

The procedure RightVentricleTachyVeryFast starts a right ventricular tachycardia with a rate of 250 bpm.

procedure ParoxysmalAtrialTachy

The procedure ParoxysmalAtrialTachy starts a Paroxysmal atrial tachycardia. The rhythm changes in a random manner between SinusRhythm and AtrialTachy.

procedure PolymorphousVentricularTachy

The procedure PolymorphousVentricularTachy starts a polymorphic ventricular tachycardia with a mean ventricle rate of 200 bpm. Change the rate of the tachycardia with the procedure Set_LeftVentricleRate.

procedure RightVentricleTachyFast

The procedure RightVentricleTachyFast starts a right ventricular tachycardia with a rate of 220 bpm.

procedure RightVentricleTachyMedium

The procedure RightVentricleTachyMedium starts a right ventricular tachycardia with a rate of 165 bpm.

Alternative name:
procedure RightVentricleTachy;

procedure RightVentricleTachySlow

The procedure RightVentricleTachySlow starts a right ventricular tachycardia with a rate of 130 bpm.

procedure RightVentricleTachyVeryFast

The procedure RightVentricleTachyVeryFast starts a right ventricular tachycardia with a rate of 250 bpm.

procedure SinusArrest

The procedure SinusArrest stops the sinoatrial node from discharging impulses. With standard settings, the atrioventricular node will generate an escape rhythm of 40 bpm.

procedure SinusBrady

The procedure SinusBrady starts a sinus bradycardia with a sinus rate of 45 bpm.

procedure SinusRhythm

The procedure SinusRhythm starts a sinus rhythm with a sinus rate of 68 bpm. The AVN and ventricle rates will be reset to the defaults.

procedure SinusTachy

The procedure SinusTachy starts a Sinus tachycardia with a sinus rate of 120 bpm.

procedure TorsadeDePointesCoarse

The procedure TorsadeDePointesCoarse starts a coarse polymorphic ventricular tachycardia of type torsade de pointes.

Deprecated name:

procedure TorsadeDePointes;

procedure TorsadeDePointesFine

The procedure TorsadeDePointesFine starts a fine polymorphic ventricular tachycardia of type torsade de pointes. This polymorphic tachycardia is intended to demonstrate the limitations of tachycardia detection.

procedure VentricularFibrillationCoarse

The procedure VentricularFibrillationCoarse starts a coarse ventricular fibrillation.

Deprecated name:
procedure VentricularFibrillation;

procedure VentricularFibrillationFine

The procedure VentricularFibrillationFine starts a fine ventricular fibrillation. This fibrillation is intended to demonstrate the limitations of fibrillation detection.

procedure VentricularFlutter

The procedure VentricularFlutter starts a ventricular flutter rhythm with a rate of 230 bpm.

Rhythms example

```
begin
  Reset;
  SinusRhythm;
  Wait(1000);
  SinusBrady;
  Wait(1000);
  SinusArrest;
  Wait(1000);
  IdioventricularRhythm;
  Wait(1000);
  SinusTachy;
  Wait(1000);
  BradyTachySyndrome;
  Wait(1000);
  ParoxysmalAtrialTachy;
  Wait(1000);
  AtrialFlutter2_1;
  Wait(1000);
  AtrialFlutter3_1;
  Wait(1000);
  AtrialFlutter4_1;
  Wait(1000);
  AtrialFibrillation;
  Wait(1000);
  CombinedAFibAFlut;
  Wait(1000);
  AVNodeReentryTachy;
  Wait(1000);
  LeftVentricleTachySlow;
  Wait(1000);
  LeftVentricleTachyMedium;
  Wait(1000);
  LeftVentricleTachyFast;
  Wait(1000);
```

```
  LeftVentricleTachyVeryFast;
  Wait(1000);
  RightVentricleTachySlow;
  Wait(1000);
  RightVentricleTachyMedium;
  Wait(1000);
  RightVentricleTachyFast;
  Wait(1000);
  RightVentricleTachyVeryFast;
  Wait(1000);
  PolymorphousVentricularTachy;
  Wait(1000);
  TorsadeDePointes;
  Wait(1000);
  VentricularFlutter;
  Wait(1000);
  VentricularFibrillationCoarse;
  Wait(1000);
  Set_DualTachycardia(True);
  AtrialFlutter2_1;
  VentricularFlutter;
  Wait(1000);
  Set_DualTachycardia(False);
  SinusRhythm;
end.
```

Blocks

procedure AVBlockI

The procedure AVBlockI establishes a first-degree atrioventricular block. This manifests as a prolonged PR interval of 250 ms.

procedure AVBlockIII

The procedure AVBlockIII establishes a third-degree atrioventricular block. No atrial impulses are conducted to the ventricles. The ventricles follow a ventricular escape rhythm.

procedure AVBlockIIIAsystole

The procedure AVBlockIIIAsystole establishes a third-degree atrioventricular block. In addition, all automaticities are stopped from discharging impulses.

procedure AVBlockIIMobitzI

The procedure AVBlockIII establishes a second-degree atrioventricular block of type Mobitz I (Wenckebach).

procedure AVBlockIIMobitzII_2to1

The procedure AVBlockIII establishes a second-degree atrioventricular block of type Mobitz II. The conduction rate is characterized by two P waves for every one QRS complex.

procedure AVBlockIIMobitzII_3to1

The procedure AVBlockIII establishes a second-degree atrioventricular block of type Mobitz II. The conduction rate is characterized by three P waves for every one QRS complex.

procedure AVBlockIIMobitzII_4to1

The procedure AVBlockIII establishes a second-degree atrioventricular block of type Mobitz II. The conduction rate is characterized by four P waves for every one QRS complex.

procedure NoAVBlock

The procedure NoAVBlock terminates all atrioventricular blocks.

Device Types

procedure DeviceTypeBiventricular

The procedure DeviceTypeBiventricular is useful in combination with a three chamber pacemaker. The running heart will show three electrodes, and access to the left ventricular settings will be limited. Left ventricular pacing will be only recognized when tip (1) or ring (2) is involved.

procedure DeviceTypeQuadripolar

The procedure DeviceTypeQuadripolar is useful in combination with a three chamber quadripolar pacemaker. The running heart will show three electrodes and the left ventricular electrode will have 4 poles. All left ventricular settings will be accessible.

procedure DeviceTypeSICD

The procedure DeviceTypeSICD is useful in combination with a SICD. The running heart shows a special image on which the SICD is displayed. The SICD settings will be accessible.

procedure DeviceTypeStandard

The procedure DeviceTypeStandards is useful in combination with a dual chamber pacemaker. The running heart will show two electrodes, the left ventricular settings will not be accessible and left ventricular pacing will not be recognized.

Premature Contractions

procedure PAC(Count: Integer)

The procedure PAC creates Count premature contractions in the atrium. The first premature contraction will start CouplingInterval milliseconds after an intrinsic P wave. All consecutive premature contractions will have a delay of CouplingInterval milliseconds.

Deprecated name:
procedure PrematureAtrium;

procedure PJC(Count: Integer)

The procedure PJC creates Count premature contractions in the atrioventricular node. The first premature contraction will start CouplingInterval milliseconds after an intrinsic event. All consecutive premature contraction will have a delay of CouplingInterval milliseconds.

Deprecated name:

procedure PrematureAVNode;

procedure LeftPVC(Count: Integer)

The procedure LeftPVC creates Count premature contractions in the left ventricle. The first premature contraction will start CouplingInterval milliseconds after the refractory phase. All consecutive premature contractions will have a delay of CouplingInterval milliseconds.

Deprecated name:
procedure PrematureLeftVentricle;

procedure RightPVC(Count: Integer)

The procedure RightPVC creates Count premature contractions in the right ventricle. The first premature contraction will start CouplingInterval milliseconds after the refractory phase. All consecutive premature contractions will have a delay of CouplingInterval milliseconds.

Deprecated name:
procedure PrematureRightVentricle;

procedure ImmediatePAC(Count: Integer)

The procedure ImmediatePAC creates Count premature contractions in the atrium. The first premature contraction will start immediately if the atrium is not refractory. All consecutive premature contractions will have a delay of CouplingInterval milliseconds.

Deprecated name:
procedure PrematureAtriumNow;

procedure ImmediatePJC(Count: Integer)

The procedure ImmediatePJC creates Count premature contractions in the atrioventricular node. The first premature contraction will start immediately if the atrioventricular node is not refractory. All consecutive premature contractions will have a delay of CouplingInterval milliseconds.

Deprecated name:
procedure PrematureAVNodeNow;

procedure ImmediateLeftPVC(Count: Integer)

The procedure ImmediateLftPVC creates Count premature contractions in the left ventricle. The first premature contraction will start immediately if the left ventricle is not refractory. All consecutive premature contractions will have a delay of CouplingInterval milliseconds.

Deprecated name:
procedure PrematureLeftVentricleNow;

procedure ImmediateRightPVC(Count: Integer)

The procedure ImmediateRightPVC creates Count premature contractions in the right ventricle. The first premature contraction will start immediately if the right ventricle is not refractory. All consecutive premature contractions will have a delay of CouplingInterval milliseconds.

Deprecated name:
procedure PrematureRightVentricleNow;

Demos

procedure A00Demo

Starts an A00 demo IPG.

procedure AAIDemo

Starts an AAI demo IPG.

procedure D00Demo

Starts a D00 demo IPG.

procedure DDDDemo

Starts a DDD demo IPG.

procedure DDD0VDemo

Starts a DDD0V demo IPG.

procedure DDIDemo

Starts a DDI demo IPG.

procedure DefiDemo

Starts an ICD Demo.

procedure DefiDemoOff

Cancels the charging of the ICD Demo.

procedure DemoIPGOff

Turns off a demo IPG.

procedure Induction50HzAtrialDemo

Starts a 50/60 Hz atrial induction demo.

procedure Induction50HvVentricularDemo

Starts a 50/60 Hz ventricular induction demo.

procedure V00Demo

Starts a V00 demo IPG.

procedure VDDDemo

Starts a VDD demo IPG.

procedure VVIDemo

Starts a VVI demo IPG.

Constants

Constants for Amplitudes

TAmplitudeEnum = (am0015mV, am0020mV, am0025mV, am0030mV, am0045mV, am0050mV, am0060mV, am0090mV, am0100mV, am0120mV, am0150mV, am0180mV, am0200mV, am0210mV, am0220mV, am0250mV, am0290mV, am0300mV, am0400mV, am0450mV, am0600mV, am0650mV, am0800mV, am0900mV, am1000mV, am1100mV, am1200mV, am1250mV, am1500mV, am1700mV, am2500mV, am3500mV)

am0015mv denotes 0.15 mV. The constants themselves are the corresponding integer values enumerated from 0 through 31 as alias for the respective voltage .

Constants for AV Blocks

TAvBlockType = (avbOff, avbI, avbIIM2, avbIIM3, avbIIM4, avbIIW, avbIII, avbIIIAsyst)

These constants are integer values enumerated from 0 to 7.

Constants for Device Types

TDeviceTypeEnum = (dtStandard, dtBiventricular, dtQuadripolar , dtSICD)

These constants are integer values:
dtStandard = 0
dtBiventricular = 1
dtQuadripolar = 2
dtSICD = 5

Constants for EMI States

TEMIEnum = (emiOFF, emi50HzLarge, emi50HzSmall, emiArtefacts, emiNoise, emiArtefacts2, emiSICDWireNoise)

These constants are integer values enumerated from 0 to 6.

Constants for EMI Frequency

TEMIFrequencyEnum = (ef50Hz, ef60Hz)

These constants are integer values enumerated from 0 to 1.

Constants for Farfield R Wave

TFarField = (ffOff, ffSmall, ffLarge)

These constants are integer values enumerated from 0 to 2.

Constants for Impedance Defects

TImpedanceDefectEnum = (impNormal, impFracture, impLeakage, impScar)

These constants are integer values enumerated from 0 to 3.

Constants for Pacemaker Channels

TPaceMakerChannel = (pcUnknown, pcRVCoil, pcATip, pcARing, pcRVTip, pcRVRing, pcLVTip1, pcLVRing2, pcLVRing3, pcLVRing4, pcCAN)

These constants are integer values enumerated from 0 to 10.

Constants for Pacemaker Modes

TPacemakerMode = (pmExtern, pmOff, pmDDD, pmD00, pmVVI, pmV00, pmAAI, pmA00, pmHz50A, pmHz50V, pmVDD, pmDDI, pmDDD0V)

These constants are integer values enumerated from 0 to 12.

Constants for Shock Polarities

TShockPolarity = (spUnknown, spPlus, spMinus)

These constants are integer values enumerated from 0 to 2.

Constants for Shock Types

TShockType = (stUnknown, stRV_Can, stRV_CanSVC, stRV_SVC, stRVSVC_CAN)

These constants are integer values enumerated from 0 to 4.

Constants for Thresholds

TThresholdEnum = (thr0050V, thr0075V, thr0100V, thr0125V, thr0150V, thr0175V, thr0200V, thr0225V, thr0250V, thr0275V, thr0300V, thr0325V, thr0350V, thr0400V, thr0500V, thr0600V, thr0700V, thrNoCapture, thr0450V, thr0550V, thr0650V, thr0750V, thr0800V, thr1000V, thr1300V, thr1600V, thr2000V, thr3000V)

thr0050V denotes 0.5 V. These constants are integer values from 0 to 27. The thresholds thr0400V and higher are only available for the phrenic nerve stimulation .

Constants for PNS-Thresholds

See constants for thresholds.

Constants for T Wave Amplitudes

TWaveAmplitudeType = (twNormal, twMedium, twLarge, twExtraLarge, twHighAngle)

These constants are integer values enumerated from 0 to 4.

Parameters

All parameters have a function Get_Parameter and a procedure Set_Parameter. You can query the current value of the parameter with the Get function. You can set the parameter to a new value with the Set procedure.

AccessoryPathway

function Get_AccessoryPathway: Boolean
procedure Set_AccessoryPathway(AValue: Boolean)
The function Get_AccessoryPathway returns the current state of the accessory pathway (active or on = True, not active or off = False; used for the WPW syndrome). Use the procedure Set_AccessoryPathway to change the state of the accessory pathway.

AmplitudeAtrium

function Get_AmplitudeAtrium: Integer
procedure Set_AmplitudeAtrium(AValue: Integer)
Use the function Get_AmplitudeAtrium to retrieve the current atrial amplitude. Use the procedure Set_AmplitudeAtrium to change the atrial amplitude. The possible range is from am0015mV to am0600mV.

See Constants for Amplitudes .

AmplitudeICD

function Get_AmplitudeICD: Integer
procedure Set_AmplitudeICD(AValue: Integer)
Use the function Get_AmplitudeICD to retrieve the current amplitude at the RV coil channel. Use the procedure Set_AmplitudeICD to change the amplitude at the RV coil channel. The possible range is from am0020mV to am0400mV.

See Constants for Amplitudes .

AmplitudeLeftVentricleTip

function Get_AmplitudeLeftVentricleTip: Integer
procedure Set_AmplitudeLeftVentricleTip(AValue: Integer)
Use the function Get_AmplitudeLeftVentricleTip to retrieve the current amplitude at the tip of the left ventricle electrode. Use the procedure Set_AmplitudeLeftVentricleTip to change the amplitude at the tip of the left ventricle electrode. The possible range is from am0100mV to am1500mV.

See Constants for Amplitudes .

AmplitudeLeftVentricleRing2

function Get_AmplitudeLeftVentricleRing2: Integer
procedure Set_AmplitudeLeftVentricleRing2(AValue: Integer)
Use the function Get_AmplitudeLeftVentricleRing2 to retrieve the current amplitude at the ring 2 of the left ventricle electrode. Use the procedure Set_AmplitudeLeftVentricleRing2 to change the amplitude at the ring 2 of the left ventricle electrode. The possible range is from am0100mV to am1500mV.

See Constants for Amplitudes .

AmplitudeLeftVentricleRing3

function Get_AmplitudeLeftVentricleRing3: Integer
procedure Set_AmplitudeLeftVentricleRing3(AValue: Integer)
Use the function Get_AmplitudeLeftVentricleRing3 to retrieve the current amplitude at the ring 3 of the left ventricle electrode. Use the procedure Set_AmplitudeLeftVentricleRing3 to change the amplitude at the ring 3 of the left ventricle electrode. The possible range is from am0100mV to am1500mV.

See Constants for Amplitudes .

AmplitudeLeftVentricleRing4

function Get_AmplitudeLeftVentricleRing4: Integer
procedure Set_AmplitudeLeftVentricleRing4(AValue: Integer)
Use the function Get_AmplitudeLeftVentricleRing4 to retrieve the current amplitude at the ring 4 of the left ventricle electrode. Use the procedure Set_AmplitudeLeftVentricleRing4 to change the amplitude at the ring 4 of the left ventricle electrode. The possible range is from am0100mV to am1500mV.

See Constants for Amplitudes .

AmplitudeRightVentricle

function Get_AmplitudeRightVentricle: Integer
procedure Set_AmplitudeRightVentricle(AValue: Integer)
Use the function Get_AmplitudeRightVentricle to retrieve the current right ventricular amplitude. Use the procedure Set_AmplitudeRightVentricle to change the right ventricular amplitude. The possible range is from am0100mV to am1500mV.

See Constants for Amplitudes .

AmplitudeSICDPrimary

function Get_AmplitudeSICDPrimary: Integer
procedure Set_AmplitudeSICDPrimary(AValue: Integer)
Use the function Get_AmplitudeSICDPrimary to retrieve the current primary SICD amplitude. Use the procedure Set_AmplitudeSICDPrimary to change the SICD primary amplitude. The possible range is from am0020mV to am1200mV.

See Constants for Amplitudes .

AmplitudeSICDSecondary

function Get_AmplitudeSICDSecondary: Integer
procedure Set_AmplitudeSICDSecondary(AValue: Integer)
Use the function Get_AmplitudeSICDSecondary to retrieve the current Secondary SICD amplitude. Use the procedure Set_AmplitudeSICDSecondary to change the SICD Secondary amplitude. The possible range is from am0020mV to am1200mV.

See Constants for Amplitudes .

APaceCrosstalk

function Get_APaceCrosstalk: Boolean
procedure Set_APaceCrosstalk(AValue: Boolean)
The function Get_APaceCrosstalk returns the current state of the atrial pace crosstalk (active or on = True, not active or off = False). Use the procedure Set_APaceCrosstalk to change the state of the atrial pace crosstalk.

APaceCrosstalkLatency

function Get_APaceCrosstalkLatency: Integer
procedure Set_APaceCrosstalkLatency (AValue: Integer)
The function Get_APaceCrosstalkLatency returns the latency of the atrial pace crosstalk. Use the procedure Set_APaceCrosstalkLatency to set the latency of the atrial pace crosstalk. The possible range is from 1 ms to 50 ms.

APaceCrosstalkWidth

function Get_APaceCrosstalkWidth: Integer
procedure Set_APaceCrosstalkWidth(AValue: Integer)
The function Get_APaceCrosstalkWidth returns the current width of the crosstalk pace wave. Use the procedure Set_APaceCrosstalkWidth to change the width of the crosstalk pace wave. The possible range is from 5 ms to 100 ms.

APacePLatency

function Get_APacePLatency: Integer
procedure Set_APacePLatency(AValue: Integer)
The function Get_APacePLatency returns the latency of the atrial pace. Use the procedure Set_APacePLatency to set the latency of the atrial pace. The possible range is from 1 ms to 150 ms.

ATPChances

procedure Get_ATPChances(var Termination: Integer: var Acceleration50: Integer: var Acceleration70: Integer: var Degeneration: Integer; var NoResponse: Integer)
procedure Set_ATPChances(Termination: Integer: Acceleration50: Integer: Acceleration70: Integer: Degeneration: Integer; NoResponse: Integer)
Use the function Get_ATPChances to retrieve the current settings for Termination, Acceleration 50 ms, Acceleration 70 ms, Degeneration, and No Reponse by ATP. Please note that you must declare 5 Integer variables to use this function (see example). Use the procedure Set_ATPChances to set the percentage values for Termination, Acceleration 50 ms, Acceleration 70 ms, Degeneration, and No Rsponse by ATP. The sum of all 5 values must be 100.

Example:

```
var
  Termination: Integer;
  Acceleration50: Integer;
  Acceleration70: Integer;
  Degeneration: Integer;
  NoResponse: Integer;
begin
  Reset;
  Get_ATPChances(Termination, Acceleration50, Acceleration70, Degeneration, NoResponse);
  Message('Current value for termination ' + IntToStr(Termination) + '%');
end.
```

Deprecated names:
function Get_Chances
procedure Set_Chances

AVBlock

function Get_AVBlock: Integer
procedure Set_AVBlock(AValue: Integer)
Use the function Get_AVBlock to retrieve the current AV block. Use the function Set_AVBlock to change the AV block. The range is from avbOff to avbIIIAsyst.
It is also possible to use the dedicated procedures NoAVBlock , AVBlockI, AVBlockIIMobitzII_2to1, AVBlockIIMobitzII_3to1, AVBlockIIMobitzII_4to1, AVBlockIIMobitzI, AVBlockIII, AVBlockIIIAsystole.

See Constants for AVBlocks .

AVNodeRate

function Get_AVNodeRate: Integer
procedure Set_AVNodeRate(AValue: Integer)
Use the function Get_AVNodeRate to retrieve the current rate of the AV node. Use the procedure Set_AVNodeRate to change the rate of the AV node. The possible range is from 2 bpm to 200 bpm.

BBBQRSReductionByMPP

function Get_BBBQRSReductionByMPP: Boolean
procedure Set_BBBQRSReductionByMPP(AValue: Boolean)
The function Get_BBBQRSReductionByMPP returns the current state of the BBB QRS reduction by MPP (active or on = True, not active or off = False). Use the procedure Set_BBBQRSReductionByMPP to change the state of the BBB QRS reduction by MPP.

BBBQRSWidth

function Get_BBBQRSWidth: Integer
procedure Set_BBBQRSWidth(AValue: Integer)
Use the function Get_BBBQRSWidth to retrieve the current width of the QRS complex of the limb lead ECG's. Use the procedure Set_BBBQRSWidth to change the width of the QRS complex of the limb lead ECG's. The possible range is from 80 ms to 220 ms.

The values are only applicable if a LBBB or a RBBB is set.

BlockRate

function Get_BlockRate: Integer
procedure Set_BlockRate(AValue: Integer)
Use the function Get_BlockRate to retrieve the current block rate (conduction rate). Use the procedure Set_BlockRate to change the block rate. The possible range is from 20 bpm to 250 bpm.

CouplingInterval

function Get_CouplingInterval: Integer
procedure Set_CouplingInterval(AValue: Integer)
Use the function Get_CouplingInterval to retrieve the current interval premature contractions are coupled to normal intrinsic events. Use the function Set_CouplingInterval to set the current coupling interval for premature contractions. The range is from 100 ms to 1000 ms.

DemoICDEnergy

function Get_DemoICDEnergy: Double
procedure Set_DemoICDEnergy(AValue: Double)
Use the function Get_DemoICDEnergy to retrieve the energy the demo ICD should be charged with. Use the procedure Set_DemoICDEnergy to set the energy. The range is from 0.5 to 40 J.

DemoIPG

function Get_DemoIPG
procedure Set_DemoIPG(AValue: Integer)
Use the function Get_DemoIPG to retrieve the current activated Demo IPG. Use the procedure Set_DemoIPG to activate a Demo IPG. The range is from pmExtern to pmDDD0V.

See Constants for Pacemaker Modes .

DemoIPGAVDelay

function Get_DemoIPGAVDelay: Integer
procedure Set_DemoIPGAVDelay(AValue: Integer)
Use Get_DemoIPGAVDelay to retrieve the AV delay of the demo pacemaker. Use Set_DemoIPGAVDelay to set the AV delay of the demo pacemaker. The range is from 50 ms to 200 ms.

DemoIPGHysteresisRate

function Get_DemoIPGHysteresisRate: Integer
procedure Set_DemoIPGHysteresisRate(AValue: Integer)
Use Get_DemoIPGHysteresisRate to retrieve the hysteresis rate of the demo pacemaker. Use Set_DemoIPGHysteresisRate to set the hysteresis rate of the demo pacemaker. The range is from 25 to 150 bpm.

DemoIPGRLDelay

function Get_DemoIPGRLDelay: Integer
procedure Set_DemoIPGRLDelay(AValue: Integer)
Use Get_DemoIPGRLDelay to retrieve the RV delay (interval) of the demo pacemaker. Use Set_DemoIPGRLDelay to set the RL delay of the demo pacemaker. The range is from -100 ms (left before right) to 100 ms (right before left).

DemoIPGStimulationRate

function Get_DemoIPGStimulationRate: Integer
procedure Set_DemoIPGStimulationRate(AValue: Integer)
Use Get_DemoIPGStimulationRate to retrieve the stimulation rate of the demo pacemaker. Use Set_DemoIPGStimulationRate to set the stimulation rate of the demo pacemaker. The range is from 25 bpm to 405 bpm for A00, V00 and D00 and from 25 bpm to 150 bpm for all others.

DeviceType

function Get_DeviceType: Integer
procedure Set_DeviceType(AValue: Integer)
Use the function Get_DeviceType to retrieve the current used device type. Use the procedure Set_DeviceType to change the device type. The range is from dtStandard to dtSICD.

See Constants for DeviceTypes .

DualTachycardia

function Get_DualTachycardia: Boolean
procedure Set_DualTachycardia(AValue: Boolean)
The function Get_DualTachcardia returns the current state of the dual tachycardia feature (active or on = True, not active or off = False). Use the procedure Set_DualTachcardia to change the state of the dual tachycardia feature.

EMIAtrium

function Get_EMIAtrium: Integer
procedure Set_EMIAtrium(AValue: Integer)
The function Get_EMIAtrium returns the current state of the EMI in the atrial channel. Use the procedure Set_EMIAtrium to change the state of the EMI in the atrial channel.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, emiArtefacts, and emiNoise. See also the EMIFrequency function.

Alternative names:
function Get_EMIAtriumTip: Integer

procedure Set_EMIAtriumTip(AValue: Integer)

See Constants for EMI States .

EMIFrequency

function Get_EMIFrequency: Integer
procedure Set_EMIFrequency(AValue: Integer)
Use the function Get_EMIFrequency to retrieve the current frequency used. Use the procedure Set_EMIFrequency to change the EMI frequency. ef50Hz or ef60Hz are possible values.
The procedure Set_EMIFrequency only prepares the frequency that is actually used by the EMI procedures in combination with the emi50HzLarge and emi50HzSmall value.

See Constants for EMI Frequency .

EMILeftVentricleRing2

function Get_EMILeftVentricleRing2: Integer
procedure Set_EMILeftVentricleRing2(AValue: Integer)
The function Get_EMILeftVentricleRing2 returns the current state of the EMI of ring 2 of the left ventricle electrode. Use the procedure Set_EMI LeftVentricleRing2 to change the state of the EMI of ring 2 of the left ventricle electrode.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, and emiArtefacts. See also the EMIFrequency function .

See Constants for EMI States .

EMILeftVentricleRing3

function Get_EMILeftVentricleRing3: Integer
procedure Set_EMILeftVentricleRing3(AValue: Integer)
The function Get_EMILeftVentricleRing3 returns the current state of the EMI of ring 3 of the left ventricle electrode. Use the procedure Set_EMILeftVentricleRing3 to change the state of the EMI of ring 3 of the left ventricle electrode.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, and emiArtefacts. See also the EMIFrequency function .

See Constants for EMI States .

EMILeftVentricleRing4

function Get_EMILeftVentricleRing4: Integer
procedure Set_EMILeftVentricleRing4(AValue: Integer)
The function Get_EMILeftVentricleRing4 returns the current state of the EMI of ring 4 of the left ventricle electrode. Use the procedure Set_EMILeftVentricleRing4 to change the state of the EMI of ring 4 of the left ventricle electrode.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, and emiArtefacts. See also the EMIFrequency function .

See Constants for EMI States .

EMILeftVentricleTip

function Get_EMILeftVentricleTip: Integer
procedure Set_EMILeftVentricleTip(AValue: Integer)
The function Get_EMILeftVentricleTip returns the current state of the EMI of tip of the left ventricle electrode. Use the procedure Set_EMILeftVentricleTip to change the state of the EMI of tip of the left ventricle electrode.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, and emiArtefacts. See also the EMIFrequency function .

See Constants for EMI States .

EMIPrimary

function Get_EMIPrimary: Integer

procedure Set_EMIPrimary(AValue: Integer)
The function Get_Primary returns the current state of the EMI of the primary electrode of a SICD. Use the procedure Set_EMIPrimary to change the state of the EMI of the left primary electrode of a SICD.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, emiArtefacts, emiArtefacts2 and emiSICDWireNoise.
See also the EMIFrequency function.

See Constants for EMI States.

EMIRightVentricle

function Get_EMIRightVentricle: Integer
procedure Set_EMIRightVentricle(AValue: Integer)
The function Get_EMIRightVentricle returns the current state of the EMI of the right ventricle electrode. Use the procedure Set_EMIRightVentricle to change the state of the EMI of the right ventricle electrode.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, emiArtefact, and emiNoises. See also the EMIFrequency function.

Alternative names:
function Get_EMIRightVentricleTip: Integer
procedure Set_EMIRightVentricleTip(AValue: Integer)

See Constants for EMI States.

EMIRVCoil

function Get_EMIRVCoil: Integer
procedure Set_EMIRVCoil(AValue: Integer)
The function Get_EMIRVCoil returns the current state of the EMI of the farfield (defi) channel. Use the procedure Set_EMIRVCoil to change the state of the EMI of the farfield channel.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, and emiArtefacts. See also the EMIFrequency function.

See Constants for EMI States.

EMISecondary

function Get_EMISecondary: Integer
procedure Set_EMISecondary(AValue: Integer)
The function Get_Secondary returns the current state of the EMI of the secondary electrode of a SICD. Use the procedure Set_EMISecondary to change the state of the EMI of the left secondary electrode of a SICD.
Possible values are emiOFF, emi50HzLarge, emi50HzSmall, emiArtefacts, emiArtefacts2 and emiSICDWireNoise.
See also the EMIFrequency function.

See Constants for EMI States.

ERAF

function Get_ERAF: Boolean
procedure Set_ERAF(AValue: Boolean)
The function Get_ERAF returns the current state of the ERAF setting (early recurrence of AF). The procedure Set_ERAF sets the ERAF parameter. True and False are possible as parameters.

ERVT

function Get_ERVT: Boolean
procedure Set_ERVT (AValue: Boolean)
The function Get_ERVT returns the current state of the ERVT setting (early recurrence of VT). The procedure Set_ERVT sets the ERVT parameter. True and False are possible as parameters.

Exercise

function Get_Exercise: Boolean
procedure Set_Exercise(AValue: Boolean)
Use the function Get_Exercise to retrieve the current state of the Exercise functionality. You can use Set_Exercise to change the state of the current Exercise functionality. If you set Exercise to True the patients workload will affect the sinus rate and the PR interval.

ExPRIntervalMin

function Get_ExPRIntervalMin: Integer
procedure Set_ExPRIntevalMin(AValue: Integer)
The function Get_ExPRIntervalMin returns the exercise PR interval at maximum workload. The procedure Set_ExPRIntervalMin sets the exercise PR interval.

Note: The obsolete functions Get_ExPRRateMin and Set_ExPRRateMin provided these methods until version 1.0.7066.

ExPRIntervalRest

function Get_ExPRIntervalRest: Integer
procedure Set_ExPRIntervalRest(AValue: Integer)
The function Get_ExPRIntervalRest returns the exercise PR interval at workload 0. The procedure Set_ExPRIntervalRest sets the exercise PR interval.

Note: The obsolete functions Get_ExPRRateRest and Set_ExPRRateRest provided these methods until version 1.0.7066.

ExSinusRateMax

function Get_ExSinusRateMax: Integer
procedure Set_ExSinusRateMax(AValue: Integer)
The function Get_ExSinusRateMax returns the exercise Sinus Rate at maximum workload. The procedure Set_ExSinusRateMax sets the exercise Sinue Rate at maximum workload.

ExSinusRateRest

function Get_ExSinusRateRest: Integer
procedure Set_ExSinusRateRest(AValue: Integer)
The function Get_ExSinusRateRest returns the exercise Sinus Rate at workload 0. The procedure Set_ExSinusRateRest sets the exercise Sinus Rate at workload 0.

Example:

```
// this macro shows how to set a chronotropic incompetence
begin
  Reset;
  Set_ExSinusRateRest(70);
  Set_ExSinusRateMax(80);
  Set_ExPRIntervalRest(170);
  Set_ExPRIntervalMin(120);
  Set_WorkLoad(50);
  // the exercise starts here
  Set_Exercise(True);
  // for 1 minute
  Wait(60000);
  Set_Exercise(False);
end.
```

FarfieldRWave

function Get_FarfieldRWave: Integer
procedure Set_FarfieldRWave(AValue: Integer)
The function Get_Farfield returns the current state of the farfield R wave. The procedure Set_FarFieldRWave can be used to set a new farfield R wave value. Possible values are ffOff, ffSmall, and ffLarge.

See Constants for Farfield R Wave .

FarfieldRWaveIntrinsicInterval

function Get_FarfieldRWaveIntrinsicInterval: Integer
procedure Set_FarfieldRWaveIntrinsicInterval(AValue: Integer)
The function Get_FarfieldRWaveIntrinsicInterval returns the interval between an intrinsic R wave and the farfield response in ms. The procedure Set_FarfieldRWaveIntrinsicInterval sets the intrinsic farfiel R wave interval. Possible values are from 0 to 100 ms.

FarfieldRWavePacedInterval

function Get_FarfieldRWavePacedInterval: Integer
procedure Set_FarfieldRWavePacedInterval(AValue: Integer)
The function Get_FarfieldRWavePacedInterval returns the interval between an paced R wave and the farfield response in ms. The procedure Set_FarfieldRWavePacedInterval sets the paced farfiel R wave interval. Possible values are from 50 to 200 ms.

ImpedanceCAN

function Get_ImpedanceCAN: Integer
procedure Set_ImpedanceCAN(AValue: Integer)
The function Get_ImpedanceCAN returns the value of the current body impedance. Use the procedure Set_ImpedanceCAN to set the body impedance. Possible values are between 15 and 45 Ohm.

ImpedanceDefectAtriumRing

function Get_ImpedanceDefectAtriumRing: Integer
procedure Set_ImpedanceDefectAtriumRing(AValue: Integer)
The Get_ImpedanceDefectAtriumRing returns the impedance defect of the atrial ring strand. Use Set_ImpedanceDefectAtriumRing to set the impedance defect of the atrial ring strand. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceDefectAtriumTip

function Get_ImpedanceDefectAtriumTip: Integer
procedure Set_ImpedanceDefectAtriumTip(AValue: Integer)
The Get_ImpedanceDefectAtriumTip returns the impedance defect of the atrial tip strand. Use Set_ImpedanceDefectAtriumTip to set the impedance defect of the atrial tip strand. Possible values are impNormal, impFracture, impLeakage, and impScar.

See Constants for Impedance Defects .

ImpedanceDefectICD

function Get_ImpedanceDefectICD: Integer
procedure Set_ImpedanceDefectICD(AValue: Integer)
The Get_ImpedanceDefectICD returns the impedance defect of the RV coil. Use Set_ImpedanceDefectICD to set the

impedance defect of the RV coil. Possible values are impNormal and impFracture.

See Constants for Impedance Defects .

ImpedanceDefectLeftVentricleRing2

function Get_ImpedanceDefectLeftVentricleRing2: Integer
procedure Set_ImpedanceDefectLeftVentricleRing2(AValue: Integer)
The Get_ImpedanceDefectLeftVentricleRing2 returns the impedance defect of the left ventricle ring 2 strand. Use Set_ImpedanceDefectLeftVentricleRing2 to set the impedance defect of the left ventricle ring 2 strand. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceDefectLeftVentricleRing3

function Get_ImpedanceDefectLeftVentricleRing3: Integer
procedure Set_ImpedanceDefectLeftVentricleRing3(AValue: Integer)
The Get_ImpedanceDefectLeftVentricleRing3 returns the impedance defect of the left ventricle ring 3 strand. Use Set_ImpedanceDefectLeftVentricleRing3 to set the impedance defect of the left ventricle ring 3 strand. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceDefectLeftVentricleRing4

function Get_ImpedanceDefectLeftVentricleRing4: Integer
procedure Set_ImpedanceDefectLeftVentricleRing4(AValue: Integer)
The Get_ImpedanceDefectLeftVentricleRing4 returns the impedance defect of the left ventricle ring 4 strand. Use Set_ImpedanceDefectLeftVentricleRing4 to set the impedance defect of the left ventricle ring 4 strand. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceDefectLeftVentricleTip

function Get_ImpedanceDefectLeftVentricleTip: Integer
procedure Set_ImpedanceDefectLeftVentricleTip(AValue: Integer)
The Get_ImpedanceDefectLeftVentricleTip returns the impedance defect of the left ventricle tip strand. Use Set_ImpedanceDefectLeftVentricleTip to set the impedance defect of the left ventricle tip strand. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceDefectPrimary

function Get_ImpedanceDefectPrimary: Integer
procedure Set_ImpedanceDefectPrimaary(AValue: Integer)
The Get_ImpedanceDefectPrimary returns the impedance defect of the primary SICD electrode. Use Set_ImpedanceDefectPrimary to set the impedance defect of the primary SICD electrode. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceDefectRightVentricleRing

function Get_ImpedanceDefectRightVentricleRing: Integer
procedure Set_ImpedanceDefectRightVentricleRing(AValue: Integer)
The Get_ImpedanceDefectRightVentricleRing returns the impedance defect of the right ventricle ring strand. Use

Set_ImpedanceDefectRightVentricleRing to set the impedance defect of the right ventricle ring strand. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceDefectRightVentricleTip

function Get_ImpedanceDefectRightVentricleTip: Integer
procedure Set_ImpedanceDefectRightVentricleTip(AValue: Integer)
The Get_ImpedanceDefectRightVentricleTip returns the impedance defect of the right ventricle tip strand. Use Set_ImpedanceDefectRightVentricleTip to set the impedance defect of the right ventricle tip strand. Possible values are impNormal, impFracture, impLeakage, and impScar.

See Constants for Impedance Defects .

ImpedanceDefectSecondary

function Get_ImpedanceDefectSecondary: Integer
procedure Set_ImpedanceDefectPrimaary(AValue: Integer)
The Get_ImpedanceDefectSecondary returns the impedance defect of the secondary SICD electrode. Use Set_ImpedanceDefectSecondary to set the impedance defect of the secondary SICD electrode. Possible values are impNormal, impFracture, and impLeakage.

See Constants for Impedance Defects .

ImpedanceValueAtriumRing

function Get_ImpedanceValueAtriumRing: Integer
procedure Set_ImpedanceValueAtriumRing(AValue: Integer)
The function Get_ImpedanceValueAtriumRing returns the current impedance value of the atrial ring strand. The procedure Set_ImpedanceValueAtriumRing sets the impedance value of the atrial ring strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueAtriumTip

function Get_ImpedanceValueAtriumTip: Integer
procedure Set_ImpedanceValueAtriumTip(AValue: Integer)
The function Get_ImpedanceValueAtriumTip returns the current impedance value of the atrial tip strand. The procedure Set_ImpedanceValueAtriumTip sets the impedance value of the atrial tip strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueLeftVentricleRing2

function Get_ImpedanceValueLeftVentricleRing2: Integer
procedure Set_ImpedanceValueLeftVentricleRing2(AValue: Integer)
The function Get_ImpedanceValueLeftVentricleRing2 returns the current impedance value of the left ventricular ring 2 strand. The procedure Set_ImpedanceValueLeftVentricleRing2 sets the impedance value of the left ventricular ring 2 strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueLeftVentricleRing3

function Get_ImpedanceValueLeftVentricleRing3: Integer
procedure Set_ImpedanceValueLeftVentricleRing3(AValue: Integer)

The function Get_ImpedanceValueLeftVentricleRing3 returns the current impedance value of the left ventricular ring 3 strand. The procedure Set_ImpedanceValueLeftVentricleRing3 sets the impedance value of the left ventricular ring 3 strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueLeftVentricleRing4

function Get_ImpedanceValueLeftVentricleRing4: Integer
procedure Set_ImpedanceValueLeftVentricleRing4(AValue: Integer)
The function Get_ImpedanceValueLeftVentricleRing4 returns the current impedance value of the left ventricular ring 4 strand. The procedure Set_ImpedanceValueLeftVentricleRing4 sets the impedance value of the left ventricular ring 4 strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueLeftVentricleTip

function Get_ImpedanceValueLeftVentricleTip: Integer
procedure Set_ImpedanceValueLeftVentricleTip(AValue: Integer)
The function Get_ImpedanceValueLeftVentricleTip returns the current impedance value of the left ventricular tip strand. The procedure Set_ImpedanceValueLeftVentricleTip sets the impedance value of the left ventricular tip strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValuePrimary

function Get_ImpedanceValuePrimary: Integer
procedure Set_ImpedanceValuePrimary(AValue: Integer)
The function Get_ImpedanceValuePrimary returns the current impedance value of the primary SICD electrode. The procedure Set_ImpedanceValuePrimary sets the impedance value of the primary SICD electrode. Possible values are between 150 and 520 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueRightVentricleRing

function Get_ImpedanceValueRightVentricleRing: Integer
procedure Set_ImpedanceValueRightVentricleRing(AValue: Integer)
The function Get_ImpedanceValueRightVentricleRing returns the current impedance value of the right ventricular ring strand. The procedure Set_ImpedanceValueRightVentricleRing sets the impedance value of the right ventricular ring strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueRightVentricleTip

function Get_ImpedanceValueRightVentricleTip: Integer
procedure Set_ImpedanceValueRightVentricleTip(AValue: Integer)
The function Get_ImpedanceValueRightVentricleTip returns the current impedance value of the right ventricular tip strand. The procedure Set_ImpedanceValueRightVentricleTip sets the impedance value of the right ventricular tip strand. Possible values are between 150 and 500 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

ImpedanceValueSecondary

function Get_ImpedanceValueSecondary: Integer
procedure Set_ImpedanceValueSecondary(AValue: Integer)
The function Get_ImpedanceValueSecondary returns the current impedance value of the secondary SICD electrode. The procedure Set_ImpedanceValueSecondary sets the impedance value of the secondary SICD electrode. Possible

values are between 150 and 520 Ohm. Please consider that this value reflects the set value und does not incorporate any defects.

InductionChances

procedure Get_InductionChances(var MonomorphicTachycardia: Integer: var PolyorphicTachycardia: Integer: var Fibrillation: Integer: var NoResponse: Integer)
procedure Set_InductionChances(MonomorphicTachycardia: Integer: PolyorphicTachycardia: Integer: Fibrillation: Integer: NoResponse: Integer)
Use the function Get_InductionChances to retrieve the current chances for getting monomorphic tachycardia, polyorphic tachycardia, venticular fibrillation, and no response by induction. Please note that you must declare 5 Integer variables to use this function (see example). Use the procedure Set_InductionChances to set the percentage values for chances for getting monomorphic tachycardia, polyorphic tachycardia, venticular fibrillation, and no response by induction. The sum of all 5 values must be 100.

Example:

```
var
  MonoTachy: Integer;
  PolyTachy: Integer;
  Fibrillation: Integer;
  NoResponse: Integer;
begin
  Reset;
  Get_InductionChances(MonoTachy, PolyTachy, Fibrillation, NoResponse);
  Message('Current value for fibrillation '+ IntToStr(Fibrillation) +'%');
end.
```

LBBB

function Get_LBBB: Boolean
procedure Set_LBBB(AValue: Boolean)
The function Get_LBBB returns the current state of the left bundle branch block (active or on = True, not active or off = False). Use the procedure Set_LBBB to change the state of the left bundle branch block.

LeftAtriumRate

function Get_LeftAtriumRate: Integer
procedure Set_LeftAtriumRate(AValue: Integer)
The function Get_LeftAtriumRate returns the current rate of the left atrial chamber in bpm. The procedure Set_LeftAtriumRate changes the current rate of the left atrial chamber. Possible values are from 2 to 245 bpm.

LeftPVCAmplitude

function Get_LeftPVCAmplitude: Integer
procedure Set_LeftPVCAmplitude(AValue: Integer)
Use the function Get_LeftPVCAmplitude to retrieve the current left PVC amplitude. Use the procedure Set_LeftPVCAmplitude to change the left PVC amplitude. The possible range is from 0 to 100 % of the R wave amplitude.

LeftVentricleRate

function Get_LeftVentricleRate: Integer
procedure Set_LeftVentricleRate(AValue: Integer)
The function Get_LeftVentricleRate returns the current rate of the left ventricle in bpm. The procedure Set_LeftVentricleRate changes the current rate of the left ventricle. Possible values are from 2 to 250 bpm.

LeftVentricleRing2Delay

function Get_LeftVentricleRing2Delay: Integer
procedure Set_LeftVentricleRing2Delay(AValue: Integer)
The function Get_LeftVentricleRing2Delay returns the QRS delay between right ventricle and left ventricle at ring 2. Use the procedure Set_LeftVentricleRing2Delay to set the delay between right ventricle and left ventricle at ring 2 in ms. Possible values are 10 to 160 ms.

The values are only applicable if a LBBB or a RBBB is set.

LeftVentricleRing3Delay

function Get_LeftVentricleRing3Delay: Integer
procedure Set_LeftVentricleRing3Delay(AValue: Integer)
The function Get_LeftVentricleRing3Delay returns the QRS delay between right ventricle and left ventricle at ring 3. Use the procedure Set_LeftVentricleRing3Delay to set the delay between right ventricle and left ventricle at ring 3 in ms. Possible values are 10 to 160 ms.

The values are only applicable if a LBBB or a RBBB is set.

LeftVentricleRing4Delay

function Get_LeftVentricleRing4Delay: Integer
procedure Set_LeftVentricleRing4Delay(AValue: Integer)
The function Get_LeftVentricleRing4Delay returns the QRS delay between right ventricle and left ventricle at ring 4. Use the procedure Set_LeftVentricleRing4Delay to set the delay between right ventricle and left ventricle at ring 4 in ms. Possible values are 10 to 160 ms.

The values are only applicable if a LBBB or a RBBB is set.

LeftVentricleTipDelay

function Get_LeftVentricleTipDelay: Integer
procedure Set_LeftVentricleTipDelay(AValue: Integer)
The function Get_LeftVentricleTipDelay returns the QRS delay between right ventricle and left ventricle at left tip. Use the procedure Set_LeftVentricleTipDelay to set the delay between right ventricle and left ventricle at left tip in ms. Possible values are 10 to 160 ms.

The values are only applicable if a LBBB or a RBBB is set.

PNSThresholdLeftVentricleRing2Can

function Get_PNSThresholdLeftVentricleRing2Can: Integer
procedure Set_PNSThresholdLeftVentricleRing2Can(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing2Can retrieves the actual PNSThreshold between left ventricle ring 2 and CAN. The procedure Set_PNSThresholdLeftVentricleRing2Can changes the PNSThreshold between left ventricle ring 2 and CAN.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing2Ring3

function Get_PNSThresholdLeftVentricleRing2Ring3: Integer
procedure Set_PNSThresholdLeftVentricleRing2Ring3(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing2Ring3 retrieves the actual PNSThreshold between left ventricle ring 2 and ring 3. The procedure Set_PNSThresholdLeftVentricleRing2Ring3 changes the PNSThreshold between left ventricle ring 2 and ring 3.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing2Ring4

function Get_PNSThresholdLeftVentricleRing2Ring4: Integer
procedure Set_PNSThresholdLeftVentricleRing2Ring4(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing2Ring4 retrieves the actual PNSThreshold between left ventricle ring 2 and ring 4. The procedure Set_PNSThresholdLeftVentricleRing2Ring4 changes the PNSThreshold between left ventricle ring 2 and ring 4.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing2RV

function Get_PNSThresholdLeftVentricleRing2RV: Integer
procedure Set_PNSThresholdLeftVentricleRing2RV(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing2RV retrieves the actual PNSThreshold between left ventricle ring 2 and RV coil. The procedure Set_PNSThresholdLeftVentricleRing2RV changes the PNSThreshold between left ventricle ring 2 and RV coil.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing2Tip1

function Get_PNSThresholdLeftVentricleRing2Tip1: Integer
procedure Set_PNSThresholdLeftVentricleRing2Tip1(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing2Tip1 retrieves the actual PNSThreshold between left ventricle ring 2 and tip 1. The procedure Set_PNSThresholdLeftVentricleRing2Tip1 changes the PNSThreshold between left ventricle ring 2 and tip 1.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing3Can

function Get_PNSThresholdLeftVentricleRing3Can: Integer
procedure Set_PNSThresholdLeftVentricleRing3Can(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing3Can retrieves the actual PNSThreshold between left ventricle ring 3 and CAN. The procedure Set_PNSThresholdLeftVentricleRing3Can changes the PNSThreshold between left ventricle ring 3 and CAN.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing3Ring2

function Get_PNSThresholdLeftVentricleRing3Ring2: Integer
procedure Set_PNSThresholdLeftVentricleRing3Ring2(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing3Ring2 retrieves the actual PNSThreshold between left ventricle ring 3 and ring 2. The procedure Set_PNSThresholdLeftVentricleRing3Ring2 changes the PNSThreshold between left ventricle ring 3 and ring 2.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing3Ring4

function Get_PNSThresholdLeftVentricleRing3Ring4: Integer
procedure Set_PNSThresholdLeftVentricleRing3Ring4(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing3Ring4 retrieves the actual PNSThreshold between left ventricle ring 3 and ring 4. The procedure Set_PNSThresholdLeftVentricleRing3Ring4 changes the PNSThreshold between left

ventricle ring 3 and ring 4.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing3RV

function Get_PNSThresholdLeftVentricleRing3RV: Integer
procedure Set_PNSThresholdLeftVentricleRing3RV(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing3RV retrieves the actual PNSThreshold between left ventricle ring 3 and RV coil. The procedure Set_PNSThresholdLeftVentricleRing3RV changes the PNSThreshold between left ventricle ring 3 and RV coil.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing3Tip1

function Get_PNSThresholdLeftVentricleRing3Tip1: Integer
procedure Set_PNSThresholdLeftVentricleRing3Tip1(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing3Tip1 retrieves the actual PNSThreshold between left ventricle ring 3 and tip 1. The procedure Set_PNSThresholdLeftVentricleRing3Tip1 changes the PNSThreshold between left ventricle ring 3 and tip 1.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing4Can

function Get_PNSThresholdLeftVentricleRing4Can: Integer
procedure Set_PNSThresholdLeftVentricleRing4Can(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing4Can retrieves the actual PNSThreshold between left ventricle ring 4 and CAN. The procedure Set_PNSThresholdLeftVentricleRing4Can changes the PNSThreshold between left ventricle ring 4 and CAN.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing4Ring2

function Get_PNSThresholdLeftVentricleRing4Ring2: Integer
procedure Set_PNSThresholdLeftVentricleRing4Ring2(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing4Ring2 retrieves the actual PNSThreshold between left ventricle ring 4 and ring 2. The procedure Set_PNSThresholdLeftVentricleRing4Ring2 changes the PNSThreshold between left ventricle ring 4 and ring 2.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing4Ring3

function Get_PNSThresholdLeftVentricleRing4Ring3: Integer
procedure Set_PNSThresholdLeftVentricleRing4Ring3(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing4Ring3 retrieves the actual PNSThreshold between left ventricle ring 4 and ring 3. The procedure Set_PNSThresholdLeftVentricleRing4Ring3 changes the PNSThreshold between left ventricle ring 4 and ring 3.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing4RV

function Get_PNSThresholdLeftVentricleRing4RV: Integer
procedure Set_PNSThresholdLeftVentricleRing4RV(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing4RV retrieves the actual PNSThreshold between left ventricle ring

4 and RV coil. The procedure Set_PNSThresholdLeftVentricleRing4RV changes the PNSThreshold between left ventricle ring 4 and RV coil.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleRing4Tip1

function Get_PNSThresholdLeftVentricleRing4Tip1: Integer
procedure Set_PNSThresholdLeftVentricleRing4Tip1(AValue: Integer)
The function Get_PNSThresholdLeftVentricleRing4Tip1 retrieves the actual PNSThreshold between left ventricle ring 4 and tip 1. The procedure Set_PNSThresholdLeftVentricleRing4Tip1 changes the PNSThreshold between left ventricle ring 4 and tip 1.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleTip1Can

function Get_PNSThresholdLeftVentricleTip1Can: Integer
procedure Set_PNSThresholdLeftVentricleTip1Can(AValue: Integer)
The function Get_PNSThresholdLeftVentricleTip1Can retrieves the actual PNSThreshold between left ventricle tip 1 and CAN. The procedure Set_PNSThresholdLeftVentricleTip1Can changes the PNSThreshold between left ventricle tip 1 and CAN.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleTip1Ring2

function Get_PNSThresholdLeftVentricleTip1Ring2: Integer
procedure Set_PNSThresholdLeftVentricleTip1Ring2(AValue: Integer)
The function Get_PNSThresholdLeftVentricleTip1Ring2 retrieves the actual PNSThreshold between left ventricle tip 1 and ring 2. The procedure Set_PNSThresholdLeftVentricleTip1Ring2 changes the PNSThreshold between left ventricle tip 1 and ring 2.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleTip1Ring3

function Get_PNSThresholdLeftVentricleTip1Ring3: Integer
procedure Set_PNSThresholdLeftVentricleTip1Ring3(AValue: Integer)
The function Get_PNSThresholdLeftVentricleTip1Ring3 retrieves the actual PNSThreshold between left ventricle tip 1 and ring 3. The procedure Set_PNSThresholdLeftVentricleTip1Ring3 changes the PNSThreshold between left ventricle tip 1 and ring 3.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleTip1Ring4

function Get_PNSThresholdLeftVentricleTip1Ring4: Integer
procedure Set_PNSThresholdLeftVentricleTip1Ring4(AValue: Integer)
The function Get_PNSThresholdLeftVentricleTip1Ring4 retrieves the actual PNSThreshold between left ventricle tip 1 and ring 4. The procedure Set_PNSThresholdLeftVentricleTip1Ring4 changes the PNSThreshold between left ventricle tip 1 and ring 4.

See Constants for PNS-Thresholds .

PNSThresholdLeftVentricleTip1RV

function Get_PNSThresholdLeftVentricleTip1RV: Integer
procedure Set_PNSThresholdLeftVentricleTip1RV(AValue: Integer)

The function Get_PNSThresholdLeftVentricleTip1RV retrieves the actual PNSThreshold between left ventricle tip 1 and RV coil. The procedure Set_PNSThresholdLeftVentricleTip1RV changes the PNSThreshold between left ventricle tip 1 and RV coil.

See Constants for PNS-Thresholds .

PostShockAsystole

function Get_PostShockAsystole: Integer
procedure Set_PostShockAsystole(AValue: Integer)
The function Get_PostShockAsysole returns the current value of the post shock asystole. The procedure Set_PostShockAsystole sets the post shock asystole in s. Possible values are from 0 to 180 s.

PRInterval

function Get_PRInterval: Integer
procedure Set_PRInterval(AValue: Integer)
The function Get_PRInterval returns the current PR interval. The procedure Set_PRInterval sets the PR interval. Possible values are from 50 to 400 ms.

RBBB

function Get_RBBB: Boolean
procedure Set_RBBB(AValue: Boolean)
The function Get_RBBB returns the current state of the right bundle branch block (active or on = True, not active or off = False). Use the procedure Set_RBBB to change the state of the right bundle branch block.

RetrogradeConduction

function Get_RetrogradeConduction: Boolean
procedure Set_RetrogradeConduction(AValue: Boolean)
The function Get_RetrogradeConduction returns the current state of the retrograde conduction (active or on = True, not active or off = False). Use the procedure Set_RBBB to change the state of the retrograde conduction.

RightPVCAmplitude

function Get_RightPVCAmplitude: Integer
procedure Set_RightPVCAmplitude(AValue: Integer)
Use the function Get_RightPVCAmplitude to retrieve the current right PVC amplitude. Use the procedure Set_RightPVCAmplitude to change the right PVC amplitude. The possible range is from 0 to 100 % of the R wave amplitude.

RightVentricleRate

function Get_RightVentricleRate: Integer
procedure Set_RightVentricleRate(AValue: Integer)
The function Get_RightVentricleRate returns the current rate of the right ventricle in bpm. The procedure Set_RightVentricleRate changes the current rate of the right ventricle. Possible values are from 2 to 250 bpm.

RPInterval

function Get_RPInterval: Integer
procedure Set_RPInterval(AValue: Integer)
The function Get_RPInterval returns the current (retrograde) RP interval. The procedure Set_RPInterval sets the RP interval. Possible values are from 130 to 600 ms.

RWaveVariability

function Get_RWaveVariability: Integer
procedure Set_RWaveVariability(AValue: Integer)
Use the function Get_RWaveVariability to retrieve the variability of the R wave amplitude with respirational cycle. Use the procedure Set_RWaveVariability to change the variability of the R wave with respirational cycle. The possible range is from 0 to 40%.

SinusRate

function Get_SinusRate: Integer
procedure Set_SinusRate(AValue: Integer)
The function Get_SinusRate returns the current sinus rate. The procedure Set_SinusRate changes the current sinus rate.

SinusRateVariation

function Get_SinusRateVariation: Integer
procedure Set_SinusRateVariation(AValue: Integer)
Use the function Get_SinusRateVariation to retrieve the variation of the sinus rate. Use the procedure Set_SinusRateVariation to change the variation of the sinus rate. The range is from 0 to 20%.

TempAsystole

function Get_TempAsystole: Boolean
procedure Set_TempAsystole(AValue: Boolean)
The function Get_TempAsystole returns the current temporary asystole state. Use the command Set_TempAsystole(True) to start and Set_TempAsystole(False) to stop a temporary asystole.

ThresholdAnodalRing2

function Get_ThresholdAnodalRing2: Integer
procedure Set_ThresholdAnodalRing2(AValue: Integer)
The function Get_ThresholdAnodalRing2 retrieves the actual threshold between left ventricle ring 2 and RV coil for anodal stimulation. The procedure Set_ThresholdAnodalRing2 changes the threshold between left ventricle ring 2 and RV coil for anodal stimulation.

See Constants for Thresholds .

ThresholdAnodalRing3

function Get_ThresholdAnodalRing3: Integer
procedure Set_ThresholdAnodalRing3(AValue: Integer)
The function Get_ThresholdAnodalRing3 retrieves the actual threshold between left ventricle ring 3 and RV coil for anodal stimulation. The procedure Set_ThresholdAnodalRing3 changes the threshold between left ventricle ring 3 and RV coil for anodal stimulation.

See Constants for Thresholds .

ThresholdAnodalRing4

function Get_ThresholdAnodalRing4: Integer
procedure Set_ThresholdAnodalRing4(AValue: Integer)
The function Get_ThresholdAnodalRing4 retrieves the actual threshold between left ventricle ring 4 and RV coil for anodal stimulation. The procedure Set_ThresholdAnodalRing4 changes the threshold between left ventricle ring 4 and RV coil for anodal stimulation.

See Constants for Thresholds .

ThresholdAnodalTip1

function Get_ThresholdAnodalTip1: Integer
procedure Set_ThresholdAnodalTip1(AValue: Integer)
The function Get_ThresholdAnodalTip1 retrieves the actual threshold between left ventricle tip 1 and RV coil for anodal stimulation. The procedure Set_ThresholdAnodalTip1 changes the threshold between left ventricle tip 1 and RV coil for anodal stimulation.

See Constants for Thresholds .

ThresholdAtrium

function Get_ThresholdAtrium: Integer
procedure Set_ThresholdAtrium(AValue: Integer)
The function Get_ThresholdAtrium retrieves the actual threshold in the atrial chamber. The procedure Set_ThresholdAtrium changes the threshold in the atrial chamber.

See Constants for Thresholds .

ThresholdICDDefinedByPulseWidth

function Get_ThresholdICDDefinedByPulseWidth: Boolean
procedure Set_ThresholdICDDefinedByPulseWidth(AValue: Boolean)
The function Get_ThresholdICDDefinedByPulseWidth returns the current threshold defined by pulse width parameter. Use the command Set_ThresholdICDDefinedByPulseWidth (True) to set and Set_ThresholdICDDefinedByPulseWidth (False) to reset the threshold defined by pulse width parameter.

ThresholdICDValueAtrium

function Get_ThresholdICDValueAtrium: Double
procedure Set_ThresholdICDValueAtrium(AValue: Double)
The function Get_ThresholdICDValueAtrium retrieves the actual defibrillator threshold in the atrial chamber in Joule. The procedure Set_ThresholdICDValueAtrium changes the defibrillator threshold in the atrial chamber. If the energy of a defibrillator shock exceeds the threshold, an atrial tachycardia will be terminated.

ThresholdICDValueVentricle

function Get_ThresholdICDValueVentricle: Double
procedure Set_ThresholdICDValueVentricle(AValue: Double)
The function Get_ThresholdICDValueVentricle retrieves the actual defibrillator threshold in the ventricular chamber in Joule. The procedure Set_ThresholdICDValueVentricle changes the defibrillator threshold in the ventricular chamber. If the energy of a defibrillator shock exceeds the threshold, a ventricular tachycardia will be terminated.

ThresholdICDVariationAtrium

function Get_ThresholdICDVariationAtrium: Boolean
procedure Set_ThresholdICDVariationAtrium(AValue: Boolean)
Use the function Get_ThresholdICDVariationAtrium to retrieve the state of the variation of the ICD threshold in the atrial chamber. Use the procedure Set_ThresholdICDVariationAtrium to change the state of the variation of the ICD threshold in the atrial chamber. An active variation means that the threshold can change by up to +/- 25% in a random manner.

ThresholdICDVariationVentricle

function Get_ThresholdICDVariationVentricle: Boolean
procedure Set_ThresholdICDVariationVentricle(AValue: Boolean)
Use the function Get_ThresholdICDVariationVentricle to retrieve the state of the variation of the ICD threshold in the ventricular chamber. Use the procedure Set_ThresholdICDVariationVentricle to change the state of the variation of the ICD threshold in the ventricular chamber. An active variation means that the threshold can change by up to +/- 25% in a random manner.

ThresholdLeftVentricleRing2Can

function Get_ThresholdLeftVentricleRing2Can: Integer
procedure Set_ThresholdLeftVentricleRing2Can(AValue: Integer)
The function Get_ThresholdLeftVentricleRing2Can retrieves the actual threshold between left ventricle ring 2 and CAN. The procedure Set_ThresholdLeftVentricleRing2Can changes the threshold between left ventricle ring 2 and CAN.

See Constants for Thresholds .

ThresholdLeftVentricleRing2Ring3

function Get_ThresholdLeftVentricleRing2Ring3: Integer
procedure Set_ThresholdLeftVentricleRing2Ring3(AValue: Integer)
The function Get_ThresholdLeftVentricleRing2Ring3 retrieves the actual threshold between left ventricle ring 2 and ring 3. The procedure Set_ThresholdLeftVentricleRing2Ring3 changes the threshold between left ventricle ring 2 and ring 3.

See Constants for Thresholds .

ThresholdLeftVentricleRing2Ring4

function Get_ThresholdLeftVentricleRing2Ring4: Integer
procedure Set_ThresholdLeftVentricleRing2Ring4(AValue: Integer)
The function Get_ThresholdLeftVentricleRing2Ring4 retrieves the actual threshold between left ventricle ring 2 and ring 4. The procedure Set_ThresholdLeftVentricleRing2Ring4 changes the threshold between left ventricle ring 2 and ring 4.

See Constants for Thresholds .

ThresholdLeftVentricleRing2RV

function Get_ThresholdLeftVentricleRing2RV: Integer
procedure Set_ThresholdLeftVentricleRing2RV(AValue: Integer)
The function Get_ThresholdLeftVentricleRing2RV retrieves the actual threshold between left ventricle ring 2 and RV coil. The procedure Set_ThresholdLeftVentricleRing2RV changes the threshold between left ventricle ring 2 and RV coil.

See Constants for Thresholds .

ThresholdLeftVentricleRing2Tip1

function Get_ThresholdLeftVentricleRing2Tip1: Integer
procedure Set_ThresholdLeftVentricleRing2Tip1(AValue: Integer)
The function Get_ThresholdLeftVentricleRing2Tip1 retrieves the actual threshold between left ventricle ring 2 and tip 1. The procedure Set_ThresholdLeftVentricleRing2Tip1 changes the threshold between left ventricle ring 2 and tip 1.

See Constants for Thresholds .

ThresholdLeftVentricleRing3Can

function Get_ThresholdLeftVentricleRing3Can: Integer
procedure Set_ThresholdLeftVentricleRing3Can(AValue: Integer)
The function Get_ThresholdLeftVentricleRing3Can retrieves the actual threshold between left ventricle ring 3 and CAN. The procedure Set_ThresholdLeftVentricleRing3Can changes the threshold between left ventricle ring 3 and CAN.

See Constants for Thresholds .

ThresholdLeftVentricleRing3Ring2

function Get_ThresholdLeftVentricleRing3Ring2: Integer
procedure Set_ThresholdLeftVentricleRing3Ring2(AValue: Integer)
The function Get_ThresholdLeftVentricleRing3Ring2 retrieves the actual threshold between left ventricle ring 3 and ring 2. The procedure Set_ThresholdLeftVentricleRing3Ring2 changes the threshold between left ventricle ring 3 and ring 2.

See Constants for Thresholds .

ThresholdLeftVentricleRing3Ring4

function Get_ThresholdLeftVentricleRing3Ring4: Integer
procedure Set_ThresholdLeftVentricleRing3Ring4(AValue: Integer)
The function Get_ThresholdLeftVentricleRing3Ring4 retrieves the actual threshold between left ventricle ring 3 and ring 4. The procedure Set_ThresholdLeftVentricleRing3Ring4 changes the threshold between left ventricle ring 3 and ring 4.

See Constants for Thresholds .

ThresholdLeftVentricleRing3RV

function Get_ThresholdLeftVentricleRing3RV: Integer
procedure Set_ThresholdLeftVentricleRing3RV(AValue: Integer)
The function Get_ThresholdLeftVentricleRing3RV retrieves the actual threshold between left ventricle ring 3 and RV coil. The procedure Set_ThresholdLeftVentricleRing3RV changes the threshold between left ventricle ring 3 and RV coil.

See Constants for Thresholds .

ThresholdLeftVentricleRing3Tip1

function Get_ThresholdLeftVentricleRing3Tip1: Integer
procedure Set_ThresholdLeftVentricleRing3Tip1(AValue: Integer)
The function Get_ThresholdLeftVentricleRing3Tip1 retrieves the actual threshold between left ventricle ring 3 and tip 1. The procedure Set_ThresholdLeftVentricleRing3Tip1 changes the threshold between left ventricle ring 3 and tip 1.

See Constants for Thresholds .

ThresholdLeftVentricleRing4Can

function Get_ThresholdLeftVentricleRing4Can: Integer
procedure Set_ThresholdLeftVentricleRing4Can(AValue: Integer)
The function Get_ThresholdLeftVentricleRing4Can retrieves the actual threshold between left ventricle ring 4 and CAN. The procedure Set_ThresholdLeftVentricleRing4Can changes the threshold between left ventricle ring 4 and

CAN.

See Constants for Thresholds .

ThresholdLeftVentricleRing4Ring2

function Get_ThresholdLeftVentricleRing4Ring2: Integer
procedure Set_ThresholdLeftVentricleRing4Ring2(AValue: Integer)
The function Get_ThresholdLeftVentricleRing4Ring2 retrieves the actual threshold between left ventricle ring 4 and ring 2. The procedure Set_ThresholdLeftVentricleRing4Ring2 changes the threshold between left ventricle ring 4 and ring 2.

See Constants for Thresholds .

ThresholdLeftVentricleRing4Ring3

function Get_ThresholdLeftVentricleRing4Ring3: Integer
procedure Set_ThresholdLeftVentricleRing4Ring3(AValue: Integer)
The function Get_ThresholdLeftVentricleRing4Ring3 retrieves the actual threshold between left ventricle ring 4 and ring 3. The procedure Set_ThresholdLeftVentricleRing4Ring3 changes the threshold between left ventricle ring 4 and ring 3.

See Constants for Thresholds .

ThresholdLeftVentricleRing4RV

function Get_ThresholdLeftVentricleRing4RV: Integer
procedure Set_ThresholdLeftVentricleRing4RV(AValue: Integer)
The function Get_ThresholdLeftVentricleRing4RV retrieves the actual threshold between left ventricle ring 4 and RV coil. The procedure Set_ThresholdLeftVentricleRing4RV changes the threshold between left ventricle ring 4 and RV coil.

See Constants for Thresholds .

ThresholdLeftVentricleRing4Tip1

function Get_ThresholdLeftVentricleRing4Tip1: Integer
procedure Set_ThresholdLeftVentricleRing4Tip1(AValue: Integer)
The function Get_ThresholdLeftVentricleRing4Tip1 retrieves the actual threshold between left ventricle ring 4 and tip 1. The procedure Set_ThresholdLeftVentricleRing4Tip1 changes the threshold between left ventricle ring 4 and tip 1.

See Constants for Thresholds .

ThresholdLeftVentricleTip1Can

function Get_ThresholdLeftVentricleTip1Can: Integer
procedure Set_ThresholdLeftVentricleTip1Can(AValue: Integer)
The function Get_ThresholdLeftVentricleTip1Can retrieves the actual threshold between left ventricle tip 1 and CAN. The procedure Set_ThresholdLeftVentricleTip1Can changes the threshold between left ventricle tip 1 and CAN.

See Constants for Thresholds .

ThresholdLeftVentricleTip1Ring2

function Get_ThresholdLeftVentricleTip1Ring2: Integer
procedure Set_ThresholdLeftVentricleTip1Ring2(AValue: Integer)
The function Get_ThresholdLeftVentricleTip1Ring2 retrieves the actual threshold between left ventricle tip 1 and ring 2. The procedure Set_ThresholdLeftVentricleTip1Ring2 changes the threshold between left ventricle tip 1 and

ring 2.

See Constants for Thresholds .

ThresholdLeftVentricleTip1Ring3

function Get_ThresholdLeftVentricleTip1Ring3: Integer
procedure Set_ThresholdLeftVentricleTip1Ring3(AValue: Integer)
The function Get_ThresholdLeftVentricleTip1Ring3 retrieves the actual threshold between left ventricle tip 1 and ring 3. The procedure Set_ThresholdLeftVentricleTip1Ring3 changes the threshold between left ventricle tip 1 and ring 3.

See Constants for Thresholds .

ThresholdLeftVentricleTip1Ring4

function Get_ThresholdLeftVentricleTip1Ring4: Integer
procedure Set_ThresholdLeftVentricleTip1Ring4(AValue: Integer)
The function Get_ThresholdLeftVentricleTip1Ring4 retrieves the actual threshold between left ventricle tip 1 and ring 4. The procedure Set_ThresholdLeftVentricleTip1Ring4 changes the threshold between left ventricle tip 1 and ring 4.

See Constants for Thresholds .

ThresholdLeftVentricleTip1RV

function Get_ThresholdLeftVentricleTip1RV: Integer
procedure Set_ThresholdLeftVentricleTip1RV(AValue: Integer)
The function Get_ThresholdLeftVentricleTip1RV retrieves the actual threshold between left ventricle tip 1 and RV coil. The procedure Set_ThresholdLeftVentricleTip1RV changes the threshold between left ventricle tip 1 and RV coil.

See Constants for Thresholds .

ThresholdRightVentricle

function Get_ThresholdRightVentricle: Integer
procedure Set_ThresholdRightVentricle(AValue: Integer)
The function Get_ThresholdAtrium retrieves the actual threshold in the right ventricle. The procedure Set_ThresholdAtrium changes the threshold in the right ventricle.

See Constants for Thresholds .

Trainer

function Get_Trainer: Boolean
procedure Set_Trainer(AValue: Boolean)
The function Get_Trainer retrieves the current state of the trainer state. Use the procedure Set_Trainer to change the trainer state. An activated trainer state means that only a very limited access to parameters and display values is available.

TWaveAmplitudeType

function Get_TWaveAmplitudeType: Integer
procedure Set_TWaveAmplitudeType(AValue: Integer)
Use the function Get_TWaveAmplitudeType to retrieve the current amplitude type of the T wave. Use the procedure Set_TWaveAmplitudeType to change the amplitude type of the T wave. Possible values are twNormal, twMedium, twLarge, twExtraLarge, and twHighAngle.

Deprecated names:
function Get_AmplitudeTWave
procedure Set_AmplitudeTWave

See Constants for T Wave Amplitudes

VPaceQLatency

function Get_VPaceQLatency: Integer
procedure Set_VPaceQLatency(AValue: Integer)
The function Get_VPaceQLatency returns the latency of the ventricular pace. Use the procedure Set_VPaceQLatency to set the latency of the ventricular pace. The possible range is from 1 ms to 150 ms.

VulnerablePhaseInterval

function Get_VulnerablePhaseInterval: Integer
procedure Set_VulnerablePhaseInterval(AValue: Integer)
The Vulnerable Phase Interval applies to the initiation and termination of tachyarrhythmias. It determines the width of the vulnerable phase at the end of the refractory period of the atria and the ventricles. The function Get_VulnerablePhaseInterval returns the current value of this parameter. Use the procedure Set_VulnerablePhaseInterval to change the parameter. The range is from 40 to 80 ms.

WorkLoad

function Get_WorkLoad: Integer
procedure Set_WorkLoad(AValue: Integer)
The function Get_WorkLoad retrieves the value of the workload. The procedure Set_WorkLoad changes the value of the workload. Possible values are 0 to 100 %. The value is used when Exercise is active.

Sense Events

The appropriate events will be filled if a new sense event occurs. Every new event will override older entries. It is good practice to first copy all necessary values to variables if some time-consuming actions will be executed. The procedure Clear should be called after reading the values of an event.

AtrialSenseEvent

Use this event to get information about atrial sensing.

function AtrialSenseEvent.Active: Boolean
Will be set to True if a new atrial sense event has occurred. Remains True until the Clear procedure is called.

procedure AtrialSenseEvent.Clear
Sets Active to False and EventTime to 0.

function AtrialSenseEvent.EventTime: Int64
The simulation time at which the most recent event occurred.

function AtrialSenseEvent.BeatToBeat: Integer
The interval in milliseconds between the last two sense events.

RightVentricularSenseEvent

Use this event to get information about right ventricular sensing.

function RightVentricularSenseEvent.Active: Boolean

Will be set to True if a new right ventricular sense event has occurred. Remains True until the Clear procedure is called.

procedure RightVentricularSenseEvent.Clear
Sets Active to False and EventTime to 0.

function RightVentricularSenseEvent.EventTime: Int64
The simulation time at which the most recent event occurred.

function RightVentricularSenseEvent.BeatToBeat: Integer
The interval in milliseconds between the last two sense events.

LeftVentricularSenseEvent

Use this event to get information about Left ventricular sensing.

function LeftVentricularSenseEvent.Active: Boolean
Will be set to True if a new left ventricular sense event has occurred. Remains True until the Clear procedure is called.

procedure LeftVentricularSenseEvent.Clear
Sets Active to False and EventTime to 0.

function LeftVentricularSenseEvent.EventTime: Int64
The simulation time at which the most recent event occurred.

function LeftVentricularSenseEvent.BeatToBeat: Integer
The interval in milliseconds between the last two sense events.

Example

```
// this macro shows how to query sense events
begin
  Reset;

  // clear older events
  AtrialSenseEvent.Clear;
  // wait until an event occurs
  while not AtrialSenseEvent.Active do
    Wait(5);
  // show the time the event has occurred
  Message(IntToStr(AtrialSenseEvent.EventTime) +' ms');
  // clear the event
  AtrialSenseEvent.Clear;
  // wait until the next event occurs
  while not AtrialSenseEvent.Active do
    Wait(5);
  // show the interval between the events
  Message(IntToStr(AtrialSenseEvent.BeatToBeat) +' ms');

  // there is also a RightVentricularSenseEvent
  // and a LeftVentricularSenseEvent
end.
```

Pace Events

The appropriate events will be filled if a new pace event occurs. Every new event will override older entries. It is good practice to first copy all necessary values to variables if some time-consuming actions will be executed. The procedure Clear should be called after reading the values of an event. See the example that shows how to determine the cathode and the anode of a pace.

AtrialPaceEvent

Use this event to get information about atrial pacing.

function AtrialPaceEvent.Active: Boolean
Will be set to True if a new atrial pace event has occurred. Remains True until the Clear procedure is called.

function AtrialPaceEvent.Anode: TPaceMakerChannel
Returns the channel that is the anode of the event (probably always pcARing).

function AtrialPaceEvent.Cathode: TPaceMakerChannel
Returns the channel that is the cathode of the event (always pcATip).

procedure AtrialPaceEvent.Clear
Sets Active to False, EventTime to 0, Anode and Cathode to pcUnknown, PulseWidth and Voltage to 0.

function AtrialPaceEvent.EventTime: Int64
The simulation time at which the most recent event occurred.

function AtrialPaceEvent.PulseWidth: Double
The pulse width of the pace event in milliseconds.

function AtrialPaceEvent.Voltage: Double
The voltage of the pace event in volts.

See Constants for Pacemaker Channels .

RightVentricularPaceEvent

Use this event to get information about right ventricular pacing.

function RightVentricularPaceEvent.Active: Boolean
Will be set to True if a new right ventricular pace event has occurred. Remains True until the Clear procedure is called.

function RightVentricularPaceEvent.Anode: TPaceMakerChannel
Returns the channel that is the anode of the event (probably always pcRVRing).

function RightVentricularPaceEvent.Cathode: TPaceMakerChannel
Returns the channel that is the cathode of the event (always pcRVTip).

procedure RightVentricularPaceEvent.Clear
Sets Active to False, EventTime to 0, Anode and Cathode to pcUnknown, PulseWidth and Voltage to 0.

function RightVentricularPaceEvent.EventTime: Int64
The simulation time at which the most recent event occurred.

function RightVentricularPaceEvent.PulseWidth: Double
The pulse width of the pace event in milliseconds.

function RightVentricularPaceEvent.Voltage: Double
The voltage of the pace event in volts.

See Constants for Pacemaker Channels .

LeftVentricularPaceEvent

Use this event to get information about left ventricular pacing.

function LeftVentricularPaceEvent.Active: Boolean

Will be set to True if a new left ventricular pace event has occurred. Remains True until the Clear procedure is called.

function LeftVentricularPaceEvent.Anode: TPaceMakerChannel
Returns the channel that is the anode of the event.

function LeftVentricularPaceEvent.Cathode: TPaceMakerChannel
Returns the channel that is the cathode of the event (one of pcLVTip1, pcLVRing2, pcLVRing3, pcLVRing4).

procedure LeftVentricularPaceEvent.Clear
Sets Active to False, EventTime to 0, Anode and Cathode to pcUnknown, PulseWidth and Voltage to 0.

function LeftVentricularPaceEvent.EventTime: Int64
The simulation time at which the most recent event occurred.

function LeftVentricularPaceEvent.PulseWidth: Double
The pulse width of the pace event in milliseconds.

function LeftVentricularPaceEvent.Voltage: Double
The voltage of the pace event in volts.

See Constants for Pacemaker Channels .

Example

```
// this macro shows how to query pace events
// note also the usage of a user defined function

function GetLocation(Idx: Integer): string;
begin
  case Idx of
    pcRVCoil: Result := 'RV coil';
    pcSVCCoil: Result := 'SVC coil';
    pcATip: Result := 'ATip';
    pcARing: Result := 'ARing';
    pcRVTip: Result := 'RVTip';
    pcRVRing: Result := 'RVRing';
    pcLVTip1: Result := 'LVTip1';
    pcLVRing2: Result := 'LVRing2';
    pcLVRing3: Result := 'LVRing3';
    pcLVRing4: Result := 'LVRing4';
    pcCAN: Result := 'CAN';
  else
    Result := 'unknown';
  end;
end;

var
  Cat, An: string;
  V, P: Double;
begin
  Reset;

  // clear older events
  AtrialPaceEvent.Clear;
  // set a low sinus rate to force atrial pacing
  Set_SinusRate(45);
  // wait until an event occurs
  // a connected pacemaker is necessary!
  while not AtrialPaceEvent.Active do
    Wait(5);
  // get the cathode of the pace
  Cat := GetLocation(AtrialPaceEvent.Cathode);
```

```
  // get the anode of the pace
  An := GetLocation(AtrialPaceEvent.Anode);
  // get the voltage
  V := AtrialPaceEvent.Voltage;
  // get the pulse width
  P := AtrialPaceEvent.PulseWidth;
  // show the time the event has occured
  Message(IntToStr(AtrialPaceEvent.EventTime) +' ms');
  // give the user some time to read the first message
  Wait(2000);
  // show the properties of the pace
  Message(Format('%s-%s %.2f V %.2f ms', [Cat, An, V, P]));

  // there is also a RightVentricularPaceEvent
  // and a LeftVentricularPaceEvent
end.
```

Defi Event

The defi event will be filled if a new defi (shock) event occurs. Every new event will override older entries. It is good practice to first copy all necessary values to variables if some time-consuming actions will be executed. The procedure Clear should be called after reading the values of an event.

DefiEvent

Use this event to get information about defibrillation shocks.

function DefiEvent.Active: Boolean
Will be set to True if a new defi event has occurred. Remains True until the Clear procedure is called.

procedure DefiEvent.Clear
Sets Active to False, EventTime to 0, Energy to 0, Polarity to spUnknown, ShockType to stUnknown.

function DefiEvent.Energy: Double
Returns the energy of a shock in Joule.

function DefiEvent.ShockPolarity: TShockPolarity
Returns the polarity of the first shock wave.

function DefiEvent.ShockType: TShockType
Returns the type of a shock. Possible values are RV coil to CAN (stRV_Can), RV coil to CAN||SVC coil (stRV_CanSVC), RV coil to SVC coil (stRV_SVC)

See Constants for Pacemaker Channels , Constants for Shock Polarities , Constants for Shock Types .

Example

```
// this macro shows the usage of the Defi event
// it also shows the usage of functions and
// how to write to a file

var
  Energy: Double;
  Polarity: string;
  ShockType: string;
  FileHandle: Integer;
  WaitUntil: Int64;

  function TranslatePolarity(APol: Integer): Char;
  begin
    case APol of
```

```
      spPlus:
        Result := '+';
      spMinus:
        Result := '-';
    else
      Result := '?';
    end;
  end;

  function TranslateShockType(AType: Integer): string;
  begin
    case AType of
      stRV_Can:
        Result := 'RVcoil->CAN';
      stRV_CanSVC:
        Result := 'RVcoil->SVCcoil||CAN';
      stRV_SVC:
        Result := 'RVcoil->SVCcoil';
    else
      Result := 'unknown';
    end;
  end;

begin
  Reset;

  // clear older events
  DefiEvent.Clear;
  WaitUntil := SimulationTime + 5000;
  // wait until a defi event occurs
  // for a maximum of 5 s
  while not DefiEvent.Active and
    (SimulationTime < WaitUntil) do
    Wait(5);

  // timeout or defi event?
  if DefiEvent.Active then
  begin
    // get the defi values
    Energy := DefiEvent.Energy;
    Polarity := TranslatePolarity(DefiEvent.Polarity);
    ShockType := TranslateShockType(DefiEvent.ShockType);

    // open a file
    FileHandle := FileOpen('DefiLog');
    // write the file
    FileWrite(FileHandle, Format('%s%s %d J', [Polarity, ShockType, Energy]));
    // close the file
    FileClose(FileHandle);
  end
  else
    Message('No defi event');
end.
```

Ingenieurbüro Lang
Dipl.-Ing. Lutz Lang
Hintere Dorfstr. 10
09661 Rossau
Germany

Phone: +49 3727 649947
Mail: Lutz.Lang@Lang-IB.de

Created: Tuesday, March 21, 2023