```
unit ControlAlgorithm;
```

```
*****************************************************************************
```

This unit contains the relevant code for the control algorithm mentioned in the paper in Anesthesiology. It is called by the main program every time the main program receives a new BIS value when in automatic control mode. The code in this unit was adapted from code included in a program written for an AEP-based closed loop anesthesia system. The main BIS-based closed loop anesthesia program, and the adaptation of the control algorithm code was written by Dr A Absalom of Glasgow University with Borland Delphi 2 (Inprise, Scotts Valley, CA, USA).

Only the code specific to the control algorithm itself is included, although other functions used in the program are listed. **The reader is reminded that automatic control of anesthesia is not standard clinical practise, and can and should only occur under closely controlled conditions under the supervision or direct control of experts in the field.** No part of the code listed below may be copied, or duplicated for this use. The authors of the paper, and their employers cannot be held liable for any consequences of the use of closed loop control of anesthesia by anybody else. Should any person or organisation attempt to use closed loop control of anesthesia they do so entirely at their own risk.

```
*****************************************************************************
```

```
interface

uses

Type
  TrgtBloodPropArray = array[1..3] of Single;
  TControlAlg = class(TObject)
    private
      Procedure InitializeControlVars;
      Procedure SetMaxTrgtProp(MaxProp : Integer);
      Property AsaStatus: Integer Read FAsaStatus Write FAsaStatus;
      Property AutoMode: Integer Read FAutoMode Write FAutoMode;
    public
      Function CalcTrgtPropofol(BISActual:Integer;
                          CurrentBloodProp:single):single;
      Procedure SetMinTrgtProp(MinProp : Integer);
      Procedure SetVariance(Variance : Integer);
      procedure SetBISTrgt(Target: Integer);
      constructor Create(aowner: TObject; ASA: Integer);
      destructor Free;

    published
      Property MaxTrgtProp: Integer Read FMaxTrgtProp Write SetMaxTrgtProp;
      Property MinTrgtProp: Integer Read FMinTrgtProp Write SetMinTrgtProp;
      Property BISVar: Integer Read FBISVar Write SetVariance;
      property BISTrgt: Integer Read FBISTrgt Write SetBISTrgt;
  end;

const

  SecsPerDay = 60*60*24;
```

```
// Automatic control constants
  MntTrgtPropEmerg: array[1..3] of real;
  DeltaTrgtPropMax: array[1..3] of real;
  MNTminProp;
  Kp: array[1..3] of real;
  Ki: array[1..3] of real;
  AutoCutBack: array[1..3] of real;


var

  NewBIS, Error, ErrorOld: Integer;
  TrgtBloodProp, CurrentBloodProp, SumDeltaTrgtProp, DeltaTrgtProp: Single;
  FirstLoop, TrgtPropChanged: Boolean;
  AutoTimeBase, AutoTimer, TimeSinceLastChange, TimeOfLastChange: TDateTime;
  TimePrevBIS, TimeCurrBIS: TDateTime;



implementation


Function TControlAlg.CalcTrgtPropofol(BISActual: Integer;
                     CurrentBloodProp:single):single;
  begin
// Initialise some settings
    if Firstloop = True then
      begin
        TrgtBloodProp := CurrentBloodProp;
        FirstLoop := False;
        AutoTimeBase := Time;
        AutoTimer := 1/SecsPerDay;
        TimePrevBIS := Time;
      end;
    TimePrevBIS := TimeCurrBIS;
    TimeCurrBIS := Time;
    TimeSinceLastChange := Time - TimeOfLastChange;

// Calculate the error
    Error := BISActual - BISTrgt;

// Calculate the Propofol target adjustment from this single error
    DeltaTrgtProp := Kp[ASAStatus] * (Error - ErrorOld) +
           Kp[ASAStatus] * (Ki[ASAStatus] * ((TimeCurrBIS-
TimePrevBIS)*SecsPerDay) * Error);

// Check if there actual blood propofol is close to the previous target
// and if the error > BISVar (allowable error)

    if ((CurrentBloodProp >= TrgtBloodProp-0.5) and
(CurrentBloodProp<=TrgtBloodProp+0.5) and (TrgtBloodProp>=3))
      or ((CurrentBloodProp>=TrgtBloodProp-1) and
(CurrentBloodProp<=TrgtBloodProp+1) and (TrgtBloodProp<3) and
(TrgtBloodProp>2.0))
      or ((CurrentBloodProp>=TrgtBloodProp-2) and
(CurrentBloodProp<=TrgtBloodProp+2) and (TrgtBloodProp<=2))
      or (BISActual > BISTrgt + BISVar)

      then
      begin
```

```
// Limit the interim propofol concentration adjustment to the preset maximum
// change
        if DeltaTrgtProp > DeltaTrgtPropMax[ASAStatus] then
          DeltaTrgtProp := DeltaTrgtPropMax[ASAStatus];

// Add the above adjustment to the previous ones from the current 30s epoch
        SumDeltaTrgtProp:=SumDeltaTrgtProp+DeltaTrgtProp;

// Calculate time since last propofol target change
        if (TimeSinceLastChange >= (30/SecsPerDay)) then
          begin

// If it is >= 30 sec since the last propofol target change then
// change the target to the current concentration + SumDeltaTrgtProp
            TrgtBloodProp := TrgtBloodProp + SumDeltaTrgtProp

// and reset time and SumDeltaTrgtProp values
            TimeOfLastChange := Time;
            SumDeltaTrgtProp:=0;
          end;
      end;

// Store the current error value as the old one
    ErrorOld := Error;

// Calculate how long since last automatic target reduction
    AutoTimer := Time - AutoTimeBase;

// Make a small reduction to the target every 180 sec
    if (AutoTimer > (180/SecsPerDay)) then
      begin
        AutoTimeBase := Time;
        AutoTimer := 0;
        TrgtBloodProp := TrgtBloodProp - AutoCutBack[ASAStatus];
      end;

// If propofol target is less than the minimum, make it the minimum
    if TrgtBloodProp < MinTrgtProp then
      begin
        TimeOfLastChange := Time;
        TrgtBloodProp := MinTrgtProp;
      end;

// If the propofol target is greater than the maximum, set it at the maximum
    if TrgtBloodProp > MaxTrgtProp then
      begin
        TimeOfLastChange := Time;
        TrgtBloodProp := MaxTrgtProp;
      end;
// Round the target value to one figure after the decimal point
    CalcTrgtPropofol := round(TrgtBloodProp*10)/10;
  end;



constructor TControlAlg.Create(aOwner: TObject; ASA: Integer);

destructor TControlAlg.Free;

Procedure TControlAlg.InitializeControlVars;
```

```
Procedure TControlAlg.SetBISTrgt(Target: Integer);

Procedure TControlAlg.SetMaxTrgtProp(MaxProp: Integer);

Procedure TControlAlg.SetMinTrgtProp(MinProp: Integer);

Procedure TControlAlg.SetVariance(Variance: Integer);

end.
```