```perl
# ============================================================================
# FILE: DnB_Simulate.pm                                             7/08/2020
#
# SERVICES:  DnB OPS SIMULATION FUNCTIONS
#
# DESCRIPTION:
#    This perl module provides simulation related functions used by the DnB
#    model railroad control program when the -a option is specified on the
#    DnB.pl CLI. Simulated sensor values are used instead of real layout
#    sensor values. Once set, normal main loop processing is performed. The
#    step's sensor values are used for the specified time period. The 'Desc'
#    text is displayed on the console at the beginning of each step.
#
#    Hash key '00' is used to hold and persist the simulation control variables.
#
#    Refer to the %SensorBit hash in DnB.pl for bit position definitions. A
#    colon separated list of bit positions to be set to 1 are specified by the
#    'SensorBit' element. 'Time' is the step duration in seconds.
#
# PERL VERSION: 5.24.1
#
# =============================================================================
use strict;
# -----------------------------------------------------------------------------
# Package Declaration
# -----------------------------------------------------------------------------
package DnB_Simulate;
require Exporter;
our @ISA = qw(Exporter);

our @EXPORT = qw(
   InitSimulation
   SimulationStep
   EndToEnd
);

use DnB_Turnout;
use DnB_Message;
use Storable 'dclone';
use Time::HiRes qw(sleep);

# =============================================================================
# FUNCTION:  InitSimulation
#
# DESCRIPTION:
#    This routine initializes the specified simulation data hash with the steps
#    for the requested train ops simulation.
#
# CALLING SYNTAX:
#    $result = &InitSimulation($Simulation, \%SimulationData);
#
# ARGUMENTS:
#    $Simulation         Simulation to run.
#    $SimulationData     Pointer to %SimulationData hash.
#
# RETURNED VALUES:
#    0 = Success,  1 = Error.
#
# ACCESSED GLOBAL VARIABLES:
#    None.
# =============================================================================
sub InitSimulation {
   my($Simulation, $SimulationData) = @_;
   my($result, @simSteps);

   if ($Simulation eq 'EndToEnd') {
      $result = &EndToEnd($SimulationData);

      &DisplayDebug(0, "===============================");
      &DisplayDebug(0, "= EndToEnd Simulation Details =");
      &DisplayDebug(0, "===============================");

      foreach my $step (sort keys(%$SimulationData)) {
         next if ($step eq '00');
         &DisplayDebug(0, "Step $step - " . $$SimulationData{$step}{'Desc'});
         if (exists($$SimulationData{$step}{'SensorBit'})) {
            &DisplayDebug(0, "   SensorBit: " . $$SimulationData{$step}{'SensorBit'});
         }
         if (exists($$SimulationData{$step}{'Turnout'})) {
            &DisplayDebug(0, "   Turnout: " . $$SimulationData{$step}{'Turnout'});
         }
         if (exists($$SimulationData{$step}{'YardRoute'})) {
            &DisplayDebug(0, "   YardRoute: " . $$SimulationData{$step}{'YardRoute'});
         }
      }
      &DisplayDebug(0, "===============================");
   }
   else {
      &DisplayError("InitSimulation, Invalid simulation: $Simulation");
      return 1;
   }

   @simSteps = sort keys %$SimulationData;
   $$SimulationData{'00'}{'MaxStep'} = $simSteps[-1];
   &DisplayDebug(1, "InitSimulation - Simulation: $Simulation   MaxStep: " .
                 $$SimulationData{'00'}{'MaxStep'});
   return 0;
}

# =============================================================================
# FUNCTION:  SimulationStep
#
# DESCRIPTION:
#    This routine is called to set turnout and SensorState values when running
#    a train ops simulation. The following hash sub-keys are recognized and
#    processed for each simulation step.
#
#    'SensorBit' is a colon (:) seperated list of sensor and/or track block
#    names. The associated bit number is derived by %SensorBit 'Desc' lookup.
#    Ensure search term is unique to a single bit. e.g. 'B01:B03:S01' or
#    'B07:GC1 AprW:GC1 Road.'
#
#    'Turnout' is a comma seperated list. Each list element is a colon seperated
#    turnout number and position. e.g. 'T07:Open,T08:Close'.
#
#    'YardRoute' is a single entry that maps to a valid %YardRouteData index.
#    Input is similar to keypad from/to track number. e.g. '1->3' (track 1 to
#    track 3).
#
# CALLING SYNTAX:
```

```perl
#    $result = &SimulationStep(\%SensorBit, \$SensorState1, \$SensorState2,
#                              \%SimulationData, \%TurnoutData, \%YardRouteData);
#
# ARGUMENTS:
#    $SensorBit          Pointer to %SensorBit hash.
#    $SensorState1       Pointer to $SensorState{'1'}.
#    $SensorState2       Pointer to $SensorState{'2'}.
#    $SimulationData     Pointer to %SimulationData hash.
#    $TurnoutData        Pointer to %TurnoutData hash.
#    $YardRouteData      Pointer to %YardRouteData hash.
#
# RETURNED VALUES:
#    0 = Success,  1 = Error.
#
# ACCESSED GLOBAL VARIABLES:
#    None.
# =============================================================================
sub SimulationStep {
   my($SensorBit, $SensorState1, $SensorState2, $SimulationData, $TurnoutData,
      $YardRouteData) = @_;
   my($step, @bitDesc, $timeout, $sensorBits, $bits1, $bits2, $bits3, $bits4);
   my(@match, @turnouts, $tNmbr, $tPos, $moveResult, $route);
   my($cTime) = time;

   if ($cTime >= $$SimulationData{'00'}{'Timeout'}) {
      &PlaySound("B.wav",70);
      if ($$SimulationData{'00'}{'Step'} eq $$SimulationData{'00'}{'MaxStep'}) {
         $$SimulationData{'00'}{'Step'} = '00';
         sleep 0.1;
         &PlaySound("B.wav",70);
      }
      $step = $$SimulationData{'00'}{'Step'} +1;
      $step = "0${step}" if (length($step) == 1);
      $$SimulationData{'00'}{'Step'} = $step;
      &DisplayMessage("==> Simulation step $step - Delay: " .
                      $$SimulationData{$step}{'Time'} . " sec   --> " .
                      $$SimulationData{$step}{'Desc'});

      # ----- Process 'Turnout' key. -----
      if (exists($$SimulationData{$step}{'Turnout'})) {
         &DisplayDebug(0, "Turnout: " . $$SimulationData{$step}{'Turnout'});
         @turnouts = split(',', $$SimulationData{$step}{'Turnout'});
         foreach my $turnout (@turnouts) {
            if ($turnout =~ m/^T(\d{2}):(.+)/) {
               $tNmbr = $1;
               $tPos = $2;
               $moveResult = &MoveTurnout($tPos, $tNmbr, $TurnoutData);
               if ($moveResult == 1) {
                  &DisplayError("SimulationStep, Failed to set turnout: " .
                                $turnout);
               }
            }
            else {
               &DisplayError("SimulationStep, Invalid turnout: $turnout");
            }
         }
      }

      # ----- Process 'YardRoute' key. -----
      if (exists($$SimulationData{$step}{'YardRoute'})) {
         &DisplayDebug(0, "YardRoute: " . $$SimulationData{$step}{'YardRoute'});
         if ($$SimulationData{$step}{'YardRoute'} =~ m/^(\d+)->(\d+)/) {
            $route = join("", "R", sprintf("%1x", ($1 -1)),
                                   sprintf("%1x", ($2 -1)));
            &DisplayDebug(0, "YardRoute: $route");
            if (exists($$YardRouteData{$route})) {
               if ($$YardRouteData{'Control'}{'Inprogress'} == 0) {
                  $$YardRouteData{'Control'}{'Route'} = $route;
                  $$YardRouteData{'Control'}{'Inprogress'} = 1;
                  $$YardRouteData{'Control'}{'Step'} = 0;
               }
               else {
                  &DisplayWarning("SimulationStep, skipped '$route'. A yard " .
                                 "route operation is inprogress.");
               }
            }
            else {
               &DisplayError("SimulationStep, Invalid yard route: $route");
            }
         }
      }

      # ----- Process 'SensorBit' key. -----
      if (exists($$SimulationData{$step}{'SensorBit'})) {
         &DisplayDebug(0, "SensorBit: " . $$SimulationData{$step}{'SensorBit'});
         @bitDesc = split(':', $$SimulationData{$step}{'SensorBit'});
         $sensorBits = 0;              # Clear all bit positions.
         foreach my $bit (@bitDesc) {
            @match = grep { $$SensorBit{$_}{'Desc'} =~ /$bit/ } keys %$$SensorBit;
            if ($#match == 0) {
               $sensorBits = $sensorBits | (1 << $match[0]);  # Position and add bit.
            }
            else {
               &DisplayError("SimulationStep, Invalid sensor bit: '$bit'  " .
                             "match: '" . join(",", @match) . "'");
            }
         }
         $$SensorState1 = $sensorBits & 0xFFFF;
         $bits1 = $$SensorState1 & 0xFF;
         $bits2 = ($$SensorState1 >> 8) & 0xFF;
         $$SensorState2 = ($sensorBits >> 16) & 0xFFFF;
         $bits3 = $$SensorState2 & 0xFF;
         $bits4 = ($$SensorState2 >> 8) & 0xFF;
         $timeout = $$SimulationData{$step}{'Time'};
         $$SimulationData{'00'}{'Timeout'} = $cTime + $timeout;
         &DisplayDebug(0, "                                            " .
            "33222222 22221111                 111111");
         &DisplayDebug(0, "                                            " .
            "10987654 32109876                 54321098 76543210");
         &DisplayDebug(0, "SimulationStep - Timeout: $timeout   SensorState2: " .
            sprintf("%0.8b", $bits4) . " " . sprintf("%0.8b", $bits3) .
            "   SensorState1: " . sprintf("%0.8b", $bits2) . " " .
            sprintf("%0.8b", $bits1));
      }
   }
   return 0;
}

# ==============================================================================
# FUNCTION:  EndToEnd
#
# DESCRIPTION:
#    This routine returns the EndToEnd simulation steps. Sensor values are set
#    corresponding to the following train movements. Refer to the description
#    of the SimulationStep routine for information about the specification of
#    sensor bits, yard routes, and turnout positioning in this hash.
#
#    Train 1:
#       From holdover B01 upgrade to yard B10 via B07.
#       From yard B10 downgrade via B09 to holdover B02.
#    Train 2:
#       From holdover B02 upgrade to yard B09 via B08.
#       From yard B09 downgrade via B07 to holdover B01.
#
#    Yard routes are set to position turnouts for train transit of blocks B09
#    and B10 since the associated turnouts are not automatically set by sensor
#    input. For exercise purposes, turnouts on yard track 5 are opened and will
#    be closed as part of the yard routes.
#
# CALLING SYNTAX:
#    $result = &EndToEnd(\%SimulationData);
#
# ARGUMENTS:
#    $SimulationData     Pointer to %SimulationData hash.
#
# RETURNED VALUES:
#    0 = Success,  1 = Error.
#
# ACCESSED GLOBAL VARIABLES:
#    None.
# =============================================================================
sub EndToEnd {
   my($SimulationData) = @_;

   my %EndToEnd = (
      '00' => {'Step' => '00', 'MaxStep' => 0, 'Timeout' => 0},
# ===
      '01' => {'Desc' => 'Train 1 in holdover B01.',
               'Time' => 3, 'SensorBit' => 'B01'},

      '02' => {'Desc' => 'Train 1 holdover leaving B01. aT03o, aT01o',
               'Time' => 2, 'SensorBit' => 'B01:S03'},

      '03' => {'Desc' => 'Train 1 upgrade enters B03.',
               'Time' => 2, 'SensorBit' => 'B03:B01:S01'},

      '04' => {'Desc' => 'Train 1 upgrade in-transit B03.',
               'Time' => 5, 'SensorBit' => 'B03',
               'Turnout' => 'T08:Open,T10:Open'},    # exercise

      '05' => {'Desc' => 'Train 1 upgrade enters B05.',
               'Time' => 2, 'SensorBit' => 'B05:B03'},

      '06' => {'Desc' => 'Train 1 upgrade in-transit B05.',
               'Time' => 5, 'SensorBit' => 'B05'},

      '07' => {'Desc' => 'Train 1 upgrade leaving B05. aT06o',
               'Time' => 2, 'SensorBit' => 'B06:B05:S06'},

        '08' => {'Desc' => 'Train 1 upgrade in-transit B06. T07c',
                 'Time' => 5, 'SensorBit' => 'B06:S07',
                 'Turnout' => 'T07:Close'},           # set for B07.

        '09' => {'Desc' => 'Train 1 upgrade enters B07.',
                 'Time' => 2, 'SensorBit' => 'B07:B06:S07:S08'},

        '10' => {'Desc' => 'Train 1 upgrade in-transit B07. GC1 active',
                 'Time' => 4, 'SensorBit' => 'B07:GC1 AprE'},

        '11' => {'Desc' => 'Train 1 upgrade in-transit B07. Route to B10',
                 'Time' => 2, 'SensorBit' => 'B07:GC1 Road:GC1 AprE',
                 'YardRoute' => '1->3'},

        '12' => {'Desc' => 'Train 1 upgrade enters B10. Route B10',
                 'Time' => 2, 'SensorBit' => 'B10:B07:GC1 AprW:GC1 Road',
                 'YardRoute' => '3->3'},

        '13' => {'Desc' => 'Train 1 yard in-transit B10. Route to B08',
                 'Time' => 5, 'SensorBit' => 'B10',
                 'Turnout' => 'T12:Open,T13:Open,T14:Open,T15:Open',  # exercise
                 'YardRoute' => '3->2'},
  # ===
        '14' => {'Desc' => 'Train 1 yard leaving B10. GC2 active',
                 'Time' => 3, 'SensorBit' => 'B10:GC2 AprW'},

        '15' => {'Desc' => 'Train 1 downgrade enters B08.',
                 'Time' => 2, 'SensorBit' => 'B08:B10:GC2 Road:GC2 AprW'},

        '16' => {'Desc' => 'Train 1 downgrade in-transit B08.',
                 'Time' => 2, 'SensorBit' => 'B08:B10:GC2 AprE:GC2 Road'},

        '17' => {'Desc' => 'Train 1 downgrade in-transit B08. aT07o',
                 'Time' => 2, 'SensorBit' => 'B08:S09:GC2 AprE'},

        '18' => {'Desc' => 'Train 1 downgrade enters B06.',
                 'Time' => 2, 'SensorBit' => 'B06:B08:S09:S07'},

        '19' => {'Desc' => 'Train 1 downgrade in-transit B06.',
                 'Time' => 5, 'SensorBit' => 'B06:S07'},

        '20' => {'Desc' => 'Train 1 downgrade enters B04.',
                 'Time' => 5, 'SensorBit' => 'B04:B06'},

        '21' => {'Desc' => 'Train 1 downgrade leaving B04. aT05c',
                 'Time' => 3, 'SensorBit' => 'B03:B04:S05'},

        '22' => {'Desc' => 'Train 1 downgrade in-transit B03. B01 occupied',
                 'Time' => 5, 'SensorBit' => 'B03:B01'},

        '23' => {'Desc' => 'Train 1 downgrade in-transit B03. aT01o aT03c',
                 'Time' => 3, 'SensorBit' => 'B03:B01:S01'},

        '24' => {'Desc' => 'Train 1 downgrade enters B02.',
                 'Time' => 3, 'SensorBit' => 'B02:B03:B01'},
  # ===
        '25' => {'Desc' => 'Train 1 holdover in-transit B02.',
                 'Time' => 5, 'SensorBit' => 'B02'},

        '26' => {'Desc' => 'Train 2 holdover leaving B02. aT02o aT01c',
                 'Time' => 2, 'SensorBit' => 'B02:S02'},

      '27' => {'Desc' => 'Train 2 upgrade enters B03.',
               'Time' => 2, 'SensorBit' => 'B03:B02:S01'},

      '28' => {'Desc' => 'Train 2 upgrade in-transit B03.',
               'Time' => 5, 'SensorBit' => 'B03',
               'Turnout' => 'T08:Open,T10:Open'},    # exercise

      '29' => {'Desc' => 'Train 2 upgrade enters B05.',
               'Time' => 2, 'SensorBit' => 'B05:B03'},

      '30' => {'Desc' => 'Train 2 upgrade in-transit B05.',
               'Time' => 5, 'SensorBit' => 'B05'},

      '31' => {'Desc' => 'Train 2 upgrade leaving B05. aT06o',
               'Time' => 2, 'SensorBit' => 'B06:B05:S06'},

      '32' => {'Desc' => 'Train 2 upgrade in-transit B06. T07o.',
               'Time' => 4, 'SensorBit' => 'B06:S07',
               'Turnout' => 'T07:Open'},            # set for B08.

      '33' => {'Desc' => 'Train 2 upgrade enters B08.',
               'Time' => 2, 'SensorBit' => 'B08:B06:S07:S09'},

      '34' => {'Desc' => 'Train 2 upgrade in-transit B08. GC2 active',
               'Time' => 4, 'SensorBit' => 'B08:GC2 AprE'},

      '35' => {'Desc' => 'Train 2 upgrade in-transit B08. Route to B09',
               'Time' => 2, 'SensorBit' => 'B08:GC2 Road:GC2 AprE',
               'YardRoute' => '2->4'},

      '36' => {'Desc' => 'Train 2 upgrade enters B09. Route B09',
               'Time' => 2, 'SensorBit' => 'B09:B08:GC2 AprW:GC2 Road',
               'YardRoute' => '4->4'},

      '37' => {'Desc' => 'Train 2 yard in-transit B09. Route to B07',
               'Time' => 5, 'SensorBit' => 'B09',
               'Turnout' => 'T16:Open,T17:Open',    # exercise
               'YardRoute' => '4->1'},
# ===
      '38' => {'Desc' => 'Train 2 yard in-transit B09. GC1 active',
               'Time' => 3, 'SensorBit' => 'B09:GC1 AprW'},

      '39' => {'Desc' => 'Train 2 downgrade enters B07.',
               'Time' => 2, 'SensorBit' => 'B07:B09:GC1 Road:GC1 AprW'},

      '40' => {'Desc' => 'Train 2 downgrade in-transit B07.',
               'Time' => 2, 'SensorBit' => 'B07:GC1 AprE:GC1 Road'},

      '41' => {'Desc' => 'Train 2 downgrade in-transit B07. S08, GC1 AprE.',
               'Time' => 2, 'SensorBit' => 'B07:S08:GC1 AprE'},

      '42' => {'Desc' => 'Train 2 downgrade enters B06. aT07c',
               'Time' => 2, 'SensorBit' => 'B06:B07:S07:S08'},

      '43' => {'Desc' => 'Train 2 downgrade in-transit B06.',
               'Time' => 5, 'SensorBit' => 'B06:S07'},

      '44' => {'Desc' => 'Train 2 downgrade enters B04.',
                 'Time' => 5, 'SensorBit' => 'B04:B06'},

        '45' => {'Desc' => 'Train 2 downgrade leaving B04. aT05c',
                 'Time' => 3, 'SensorBit' => 'B03:B04:S05'},

        '46' => {'Desc' => 'Train 2 downgrade in-transit B03. B02 occupied',
                 'Time' => 5, 'SensorBit' => 'B03:B02'},

        '47' => {'Desc' => 'Train 2 downgrade in-transit B03. aT01c aT02c',
                 'Time' => 3, 'SensorBit' => 'B03:B02:S01'},

        '48' => {'Desc' => 'Train 2 downgrade enters B01.',
                 'Time' => 3, 'SensorBit' => 'B01:B03:B02'},

        '49' => {'Desc' => 'Train 2 holdover in-transit B01.',
                 'Time' => 5, 'SensorBit' => 'B01',
                 'Turnout' => 'T16:Close,T17:Close'},    # exercise
# ===
        '50' => {'Desc' => 'Cycle complete. Reset all sensor bits.',
                 'Time' => 3, 'SensorBit' => ''}
   );

   %$SimulationData = %{ dclone(\%EndToEnd) };
   return 0;
}

return 1;

```