```perl
# ============================================================================
# FILE: DnB_Mainline.pm                                          8/27/2020
#
# SERVICES:  DnB TRACK PROCESSING FUNCTIONS
#
# DESCRIPTION:
#    This perl module provides mainline track processing related functions used
#    by the DnB model railroad control program.
#
# PERL VERSION: 5.24.1
#
# ============================================================================
use strict;
# ----------------------------------------------------------------------------
# Package Declaration
# ----------------------------------------------------------------------------
package DnB_Mainline;
require Exporter;
our @ISA = qw(Exporter);

our @EXPORT = qw(
   ProcessHoldover
   ProcessMidway
   ProcessWye
   HoldoverTrack
   MidwayTrack
   WyeTrack
   CheckTurnout
);

use DnB_Message;
use DnB_Sensor;
use DnB_Turnout;

# ============================================================================
# FUNCTION:  ProcessHoldover
#
# DESCRIPTION:
#    This routine performs the operational functions related to the holdover
#    track section. Functions include turnout point positioning and setting
#    of track power polarity.
#
#    Retriggerable timers in %TrackData are used for S1, S2, and S3 to ensure
#    that a route is set only once for as long as the train activates the
#    sensor.
#
#    The S1, S2, and S3 sensors also retrigger the 'RouteTime' if a temporary
#    holdover route has be set ('RouteLocked') via button input. When a route
#    has been set, no other ProcessHoldover functions are performed.
#
# CALLING SYNTAX:
#    $result = &ProcessHoldover(\%TrackData, \%SensorBit, \%SensorState,
#                               \%TurnoutData, \%GpioData);
#
# ARGUMENTS:
#    $TrackData          Pointer to %TrackData hash.
#    $SensorBit          Pointer to %SensorBit hash.
#    $SensorState        Pointer to %SensorState hash.
#    $TurnoutData        Pointer to %TurnoutData hash.
#    $GpioData           Pointer to %GpioData hash. (polarity relays)
```

```
61  #
62  # RETURNED VALUES:
63  #    0 = Success,  1 = Error.
64  #
65  # ACCESSED GLOBAL VARIABLES:
66  #    None.
67  # ============================================================================
68  sub ProcessHoldover {
69     my($TrackData, $SensorBit, $SensorState, $TurnoutData, $GpioData) = @_;
70     my($moveResult, $turnout, $position, $gpio, $value, $check);
71     my(%bitPos) = ('B1' => '00', 'B2' => '01', 'B3' => '02', 'S1' => '16',
72                    'S2' => '17', 'S3' => '18');
73     my(%routes) = (
74        'InB1'  => 'T01:Close,T02:Close,T03:Close,GPIO6_PR02:0',
75        'InB2'  => 'T01:Open,T03:Close,T02:Close,GPIO5_PR01:0',
76        'OutB1' => 'T03:Open,T01:Open,GPIO6_PR02:1',
77        'OutB2' => 'T02:Open,T01:Close,GPIO5_PR01:1');
78
79     my(@route) = ();
80     my($cTime) = time;
81
82     &DisplayDebug(2, "ProcessHoldover entry ...");
83
84  # --- RouteLocked related processing. ----------------------------------------
85     if ($$TrackData{'01'}{'RouteLocked'} == 1) {
86        if (&GetSensorBit($bitPos{'S1'}, $SensorBit, $SensorState) == 1 or
87            &GetSensorBit($bitPos{'S2'}, $SensorBit, $SensorState) == 1 or
88            &GetSensorBit($bitPos{'S3'}, $SensorBit, $SensorState) == 1) {
89           $$TrackData{'01'}{'ManualRouteTime'} = time + 60;
90        }
91        return 0;
92     }
93
94  # --- Processing for S1 sensor disable. --------------------------------------
95  # S1 sensor input is ignored for an outbound train to prevent improper turnout
96  # positioning; 'Direction' is set to 'Out'. 'Direction' is set back to 'In' and
97  # outbound route flags are reset when track block B3 is no longer occupied.
98
99     if ($$TrackData{'01'}{'Direction'} eq 'Out' and
100        $$TrackData{'01'}{'WaitB3Inact'} == 1) {
101       if (&GetSensorBit($bitPos{'B3'}, $SensorBit, $SensorState) == 0) {
102          &DisplayMessage("ProcessHoldover, block B3 is unoccupied.");
103          $$TrackData{'01'}{'Direction'} = 'In';
104          $$TrackData{'01'}{'WaitB3Inact'} = 0;
105  #        @route = split(",", $routes{'InB1'});  # Default turnout positions.
106       }
107    }
108
109 # --- Sensor S1 processing. --------------------------------------------------
110    if (&GetSensorBit($bitPos{'S1'}, $SensorBit, $SensorState) == 1) {
111       if ($$TrackData{'01'}{'Direction'} eq 'Out' and
112           $$TrackData{'01'}{'WaitB3Inact'} == 0) {
113          &DisplayDebug(1, "ProcessHoldover, S1 is active.");
114          $$TrackData{'01'}{'WaitB3Inact'} = 1;
115          &DisplayMessage("ProcessHoldover, waiting for block B3 to be unoccupied.");
116       }
117
118       if ($$TrackData{'01'}{'Direction'} eq 'In') {
119          if ($$TrackData{'01'}{'Timeout'} < $cTime) {  # If route not already set.
120             &DisplayDebug(1, "ProcessHoldover, S1 is active.");
```

```
121
122             # Should never have an inbound state with S2 or S3 active. But if so,
123             # sound train wreck.
124             if (&GetSensorBit($bitPos{'S2'}, $SensorBit, $SensorState) == 1 or
125                 &GetSensorBit($bitPos{'S3'}, $SensorBit, $SensorState) == 1) {
126                &DisplayMessage("ProcessHoldover, inbound and outbound train wreck!");
127                &PlaySound("TrainWreck3.wav");
128             }
129
130             # Alternate holdover tracks if both are unoccupied. Otherwise, route
131             # inbound train to an available track.
132             elsif (&GetSensorBit($bitPos{'B1'}, $SensorBit, $SensorState) == 0 and
133                    &GetSensorBit($bitPos{'B2'}, $SensorBit, $SensorState) == 0) {
134                if ($$TrackData{'01'}{'Last'} eq 'B1') {
135                   &DisplayMessage("ProcessHoldover, routing inbound train to B2.");
136                   $$TrackData{'01'}{'Last'} = 'B2';
137                   @route = split(",", $routes{'InB2'});
138                }
139                else {
140                   &DisplayMessage("ProcessHoldover, routing inbound train to B1.");
141                   $$TrackData{'01'}{'Last'} = 'B1';
142                   @route = split(",", $routes{'InB1'});
143                }
144             }
145             elsif (&GetSensorBit($bitPos{'B1'}, $SensorBit, $SensorState) == 0) {
146                &DisplayMessage("ProcessHoldover, routing inbound train to B1.");
147                @route = split(",", $routes{'InB1'});
148             }
149             elsif (&GetSensorBit($bitPos{'B2'}, $SensorBit, $SensorState) == 0) {
150                &DisplayMessage("ProcessHoldover, routing inbound train to B2.");
151                @route = split(",", $routes{'InB2'});
152             }
153             else {
154                &DisplayMessage("ProcessHoldover, inbound sidings " .
155                                "full train wreck!");
156                &PlaySound("TrainWreck3.wav");
157             }
158          }
159          $$TrackData{'01'}{'Timeout'} = $cTime + 10;  # Disable S1 processing.
160       }
161    }
162
163 # --- Sensor S2 processing. ------------------------------------------------
164 #  Note: A retriggerable timer is used to prevent multiple turnout settings. It
165 #        is possible for this timer to expire for a slow or stopped train that
166 #        leaves the sensor unblocked. No adverse affect, just some CPU cycles.
167 #        The timer is used instead of the 'Out' direction state so that a second
168 #        siding departure can occur while the previous train still occupies the
169 #        B3 block.
170
171    elsif (&GetSensorBit($bitPos{'S2'}, $SensorBit, $SensorState) == 1) {
172       if ($$TrackData{'02'}{'Timeout'} < $cTime) {  # If route not already set.
173          &DisplayDebug(1, "ProcessHoldover, S2 is active.");
174          &DisplayMessage("ProcessHoldover, routing outbound B2 train to B3.");
175          @route = split(",", $routes{'OutB2'});
176          $$TrackData{'01'}{'Direction'} = 'Out';
177       }
178       $$TrackData{'02'}{'Timeout'} = $cTime + 3;    # Disable S2 processing.
179    }
180
```

```
# --- Sensor S3 processing. ------------------------------------------------
#  Above note for S2 applies here also.

   elsif (&GetSensorBit($bitPos{'S3'}, $SensorBit, $SensorState) == 1) {
      if ($$TrackData{'03'}{'Timeout'} < $cTime) {  # If route not already set.
         &DisplayDebug(1, "ProcessHoldover, S3 is active.");
         &DisplayMessage("ProcessHoldover, routing outbound B1 train to B3.");
         @route = split(",", $routes{'OutB1'});
         $$TrackData{'01'}{'Direction'} = 'Out';
      }
      $$TrackData{'03'}{'Timeout'} = $cTime + 3;    # Disable S3 processing.
   }

# --- Set turnouts and relays if @route is specified. -------------------------
   if ($#route >= 0) {
      foreach my $device (@route) {
         if ($device =~ m/^T(\d+):(.+)/) {
            $turnout = $1;
            $position = $2;
            $moveResult = &MoveTurnout($position, $turnout, $TurnoutData);
            if ($moveResult == 1) {
               &DisplayError("ProcessHoldover, Failed to set turnout " .
                             "$turnout to $position");
            }
            else {
               &DisplayMessage("ProcessHoldover, turnout $turnout " .
                               "set to $position.");
            }
         }
         elsif ($device =~ m/^(GPIO.+?):(\d)/) {
            $gpio = $1;
            $value = $2;
            $$GpioData{$gpio}{'Obj'}->write($value);  # Set power polarity relay.
            $check = $$GpioData{$gpio}{'Obj'}->read;  # Readback and check.
            if ($check != $value) {
               &DisplayError("ProcessHoldover, Failed to set power " .
                             "relay $gpio to $value");
            }
            else {
               &DisplayMessage("ProcessHoldover, relay $gpio " .
                               "set to $value.");
            }
         }
         else {
            &DisplayError("ProcessHoldover, Invalid S1 route entry: " .
                          "$device");
         }
      }
   }
   return 0;
}

# ============================================================================
# FUNCTION:  ProcessMidway
#
# DESCRIPTION:
#    This routine performs the operational functions related to the midway
#    track section. Functions include turnout point positioning. A turnout
#    is not processed if previously locked by user button input.
#
```

```
#    Retriggerable timers in %TrackData are used for S5 and S6 to ensure
#    that a route is set only once for as long as the train activates the
#    sensor.
#
#    The respective turnout it set back to the Inactive position after its
#    timer expires. This action is inhibited by a manually set position. In
#    this case, reposition will occur after a 2nd timeout cycle.
#
# CALLING SYNTAX:
#    $result = &ProcessMidway(\%TrackData, \%SensorBit, \%SensorState,
#                             \%TurnoutData);
#
# ARGUMENTS:
#    $TrackData          Pointer to %TrackData hash.
#    $SensorBit          Pointer to %SensorBit hash.
#    $SensorState        Pointer to %SensorState hash.
#    $TurnoutData        Pointer to %TurnoutData hash.
#
# RETURNED VALUES:
#    0 = Success,  1 = Error.
#
# ACCESSED GLOBAL VARIABLES:
#    None.
# ============================================================================
sub ProcessMidway {
   my($TrackData, $SensorBit, $SensorState, $TurnoutData) = @_;
   my($moveResult);
   my(%bitPos) = ('S5' => '20', 'S6' => '21');
   my($cTime) = time;

   &DisplayDebug(2, "ProcessMidway entry ...");

# --- Sensor S5 processing. ------------------------------------------------
   if ($$TrackData{'05'}{'Locked'} == 0) {
      if (&GetSensorBit($bitPos{'S5'}, $SensorBit, $SensorState) == 1) {
         &DisplayDebug(1, "ProcessMidway, S5 is active.");

         # Move turnout if no inprogress timeout. Otherwise, restart timeout.
         if ($$TurnoutData{'05'}{'Pos'} !=
             $$TurnoutData{'05'}{ $$TrackData{'05'}{'Active'} } and
             $$TrackData{'05'}{'Timeout'} < $cTime) {

            $moveResult = &MoveTurnout($$TrackData{'05'}{'Active'}, '05',
                                       $TurnoutData);
            if ($moveResult == 1) {
               &DisplayError("ProcessMidway, Failed to set turnout " .
                             "05 to $$TrackData{'05'}{'Active'}.");
            }
            else {
               &DisplayMessage("ProcessMidway, turnout 05 set to " .
                               "active position $$TrackData{'05'}{'Active'}.");
            }
         }
         $$TrackData{'05'}{'Timeout'} = $cTime + 15;   # Retrigger timeout.
         $$TrackData{'05'}{'ManualSet'} = 0;
      }
      else {

         # Reset turnout if a timeout has completed and turnout is not in the
         # Inactive position. Check for turnout Pid 0 prevents additional turnout
```

```
301             # setting during the move period.
302             if ($cTime >= $$TrackData{'05'}{'Timeout'} and
303                 $$TrackData{'05'}{'ManualSet'} == 0 and
304                 $$TurnoutData{'05'}{'Pid'} == 0 and
305                 $$TurnoutData{'05'}{'Pos'} !=
306                 $$TurnoutData{'05'}{ $$TrackData{'05'}{'Inactive'} }) {
307                $moveResult = &MoveTurnout($$TrackData{'05'}{'Inactive'}, '05',
308                                           $TurnoutData);
309                if ($moveResult == 1) {
310                   &DisplayError("ProcessMidway, Failed to set turnout " .
311                                 "05 to $$TrackData{'05'}{'Inactive'}.");
312                }
313                else {
314                   &DisplayMessage("ProcessMidway, turnout 05 set to " .
315                                   "inactive position $$TrackData{'05'}{'Inactive'}.");
316                }
317             }
318          }
319       }
320
321    # --- Sensor S6 processing. ------------------------------------------------
322       if ($$TrackData{'06'}{'Locked'} == 0) {
323          if (&GetSensorBit($bitPos{'S6'}, $SensorBit, $SensorState) == 1) {
324             &DisplayDebug(1, "ProcessMidway, S6 is active.");
325
326             # Move turnout if no inprogress timeout. Otherwise, restart timeout.
327             if ($$TurnoutData{'06'}{'Pos'} !=
328                 $$TurnoutData{'06'}{ $$TrackData{'06'}{'Active'} } and
329                 $$TrackData{'06'}{'Timeout'} < $cTime) {
330
331                $moveResult = &MoveTurnout($$TrackData{'06'}{'Active'}, '06',
332                                           $TurnoutData);
333                if ($moveResult == 1) {
334                   &DisplayError("ProcessMidway, Failed to set turnout " .
335                                 "06 to $$TrackData{'06'}{'Active'}.");
336                }
337                else {
338                   &DisplayMessage("ProcessMidway, turnout 06 set to " .
339                                   "active position $$TrackData{'06'}{'Active'}.");
340                }
341             }
342             $$TrackData{'06'}{'Timeout'} = $cTime + 15;   # Retrigger timeout.
343             $$TrackData{'06'}{'ManualSet'} = 0;
344          }
345          else {
346
347             # Reset turnout if a timeout has completed and turnout is not in the
348             # Inactive position. Check for turnout Pid 0 prevents additional turnout
349             # setting during the move period.
350             if ($cTime >= $$TrackData{'06'}{'Timeout'} and
351                 $$TrackData{'06'}{'ManualSet'} == 0 and
352                 $$TurnoutData{'06'}{'Pid'} == 0 and
353                 $$TurnoutData{'06'}{'Pos'} !=
354                 $$TurnoutData{'06'}{ $$TrackData{'06'}{'Inactive'} }) {
355                $moveResult = &MoveTurnout($$TrackData{'06'}{'Inactive'}, '06',
356                                           $TurnoutData);
357                if ($moveResult == 1) {
358                   &DisplayError("ProcessMidway, Failed to set turnout " .
359                                 "06 to $$TrackData{'06'}{'Inactive'}.");
360                }
```

```
            else {
               &DisplayMessage("ProcessMidway, turnout 06 set to " .
                               "inactive position $$TrackData{'06'}{'Inactive'}.");
            }
         }
      }
   }
   return 0;
}

# ============================================================================
# FUNCTION:  ProcessWye
#
# DESCRIPTION:
#    This routine performs the operational functions related to the wye track
#    section. Functions include turnout point positioning and setting of track
#    power polarity.
#
# CALLING SYNTAX:
#    $result = &ProcessWye(\%TrackData, \%SensorBit, \%SensorState,
#                          \%TurnoutData, \%GpioData);
#
# ARGUMENTS:
#    $TrackData          Pointer to %TrackData hash.
#    $SensorBit          Pointer to %SensorBit hash.
#    $SensorState        Pointer to %SensorState hash.
#    $TurnoutData        Pointer to %TurnoutData hash.
#    $GpioData           Pointer to %GpioData hash. (polarity relays)
#
# RETURNED VALUES:
#    0 = Success,  1 = Error.
#
# ACCESSED GLOBAL VARIABLES:
#    None.
# ============================================================================
sub ProcessWye {
   my($TrackData, $SensorBit, $SensorState, $TurnoutData, $GpioData) = @_;
   my($moveResult);
   my(%bitPos) = ('S7' => '22', 'S8' => '23', 'S9' => '24');
   my($cTime) = time;

   &DisplayDebug(2, "ProcessWye entry ...");

# --- Sensor S7 processing. ------------------------------------------------
   if (&GetSensorBit($bitPos{'S7'}, $SensorBit, $SensorState) == 1) {
      if ($$TrackData{'07'}{'Timeout'} < $cTime) {
         &DisplayDebug(1, "ProcessWye, S7 is active.");

         # Set the polarity relay based on current T7 position.
         if ($$TurnoutData{'07'}{'Pos'} == $$TurnoutData{'07'}{'Close'}) {
            if ($$TrackData{'07'}{'Polarity'} != 0) {
               $$GpioData{GPIO13_PR03}{'Obj'}->write(0);  # Set relay control bit.
               if ($$GpioData{GPIO13_PR03}{'Obj'}->read != 0) {  # Readback check.
                  &DisplayError("ProcessWye S7, Failed to set power " .
                                "relay GPIO13_PR03 to 0");
               }
               else {
                  &DisplayMessage("ProcessWye S7, power relay " .
                                  "GPIO13_PR03 set to 0.");
               }
```

```
421                $$TrackData{'07'}{'Polarity'} = 0;
422             }
423          }
424          else {
425             if ($$TrackData{'07'}{'Polarity'} != 1) {
426                $$GpioData{GPIO13_PR03}{'Obj'}->write(1); # Set relay control bit.
427                if ($$GpioData{GPIO13_PR03}{'Obj'}->read != 1) {  # Readback check.
428                   &DisplayError("ProcessWye S7, Failed to set power " .
429                                 "relay GPIO13_PR03 to 1");
430                }
431                else {
432                   &DisplayMessage("ProcessWye S7, power relay " .
433                                   "GPIO13_PR03 set to 1.");
434                }
435                $$TrackData{'07'}{'Polarity'} = 1;
436             }
437          }
438       }
439       $$TrackData{'07'}{'Timeout'} = $cTime + 2;
440    }
441
442 # --- Sensor S8 processing. ------------------------------------------------
443    if (&GetSensorBit($bitPos{'S8'}, $SensorBit, $SensorState) == 1) {
444       if ($$TrackData{'08'}{'Timeout'} < $cTime) {
445          &DisplayDebug(1, "ProcessWye, S8 is active.");
446          if ($$TurnoutData{'07'}{'Pos'} != $$TurnoutData{'07'}{'Close'}) {
447             $moveResult = &MoveTurnout('Close', '07', $TurnoutData);
448             if ($moveResult == 1) {
449                &DisplayError("ProcessWye S8, Failed to set turnout 07 to Close.");
450             }
451             else {
452                &DisplayMessage("ProcessWye S8, turnout 07 set to Close.");
453             }
454          }
455          if ($$TrackData{'07'}{'Polarity'} != 0) {
456             $$GpioData{GPIO13_PR03}{'Obj'}->write(0);  # Set relay control bit.
457             if ($$GpioData{GPIO13_PR03}{'Obj'}->read != 0) {   # Readback check.
458                &DisplayError("ProcessWye S8, Failed to set power " .
459                              "relay GPIO13_PR03 to 0");
460             }
461             else {
462                &DisplayMessage("ProcessWye S8, power relay " .
463                                "GPIO13_PR03 set to 0.");
464             }
465             $$TrackData{'07'}{'Polarity'} = 0;
466          }
467       }
468       $$TrackData{'08'}{'Timeout'} = $cTime + 2;
469    }
470
471 # --- Sensor S9 processing. ------------------------------------------------
472    if (&GetSensorBit($bitPos{'S9'}, $SensorBit, $SensorState) == 1) {
473       if ($$TrackData{'09'}{'Timeout'} < $cTime) {
474          &DisplayDebug(1, "ProcessWye, S9 is active.");
475          if ($$TurnoutData{'07'}{'Pos'} != $$TurnoutData{'07'}{'Open'}) {
476             $moveResult = &MoveTurnout('Open', '07', $TurnoutData);
477             if ($moveResult == 1) {
478                &DisplayError("ProcessWye S9, Failed to set turnout " .
479                              "07 to Open.");
480             }
```

```
481             else {
482                &DisplayMessage("ProcessWye S9, turnout 07 set to Open.");
483             }
484          }
485          if ($$TrackData{'07'}{'Polarity'} != 1) {
486             $$GpioData{GPIO13_PR03}{'Obj'}->write(1);  # Set relay control bit.
487             if ($$GpioData{GPIO13_PR03}{'Obj'}->read != 1) {   # Readback check.
488                &DisplayError("ProcessWye S9, Failed to set power " .
489                              "relay GPIO13_PR03 to 1");
490             }
491             else {
492                &DisplayMessage("ProcessWye S9, power relay " .
493                                "GPIO13_PR03 set to 1.");
494             }
495             $$TrackData{'07'}{'Polarity'} = 1;
496          }
497       }
498       $$TrackData{'09'}{'Timeout'} = $cTime + 2;
499    }
500    return 0;
501 }
502
503 # ============================================================================
504 # FUNCTION:  HoldoverTrack
505 #
506 # DESCRIPTION:
507 #    This routine processes the user buttons associated with turnouts T01, T02,
508 #    and T03 in the Holdover track section. Four buttons are provided for user
509 #    input of a desired route. In response, this routine sets the turnouts as
510 #    needed. The turnouts will be 'locked' in the requested route and a LED
511 #    indicator on the keypad will be illuminated. This route will be persisted
512 #    until one of the following conditions occur.
513 #
514 #    1. Any button on the holdover route keypad is pressed.
515 #    2. No S1, S2, or S3 sensor activity for 60 seconds.
516 #
517 # CALLING SYNTAX:
518 #    $result = &HoldoverTrack($ButtonInput, \%TurnoutData, \%TrackData,
519 #                             \%GpioData);
520 #
521 # ARGUMENTS:
522 #    $ButtonInput        User entered button input, if any.
523 #    $TurnoutData        Pointer to %TurnoutData hash.
524 #    $TrackData          Pointer to %TrackData hash.
525 #    $GpioData           Pointer to %GpioData hash. (polarity relays)
526 #
527 # RETURNED VALUES:
528 #    0 = Success,  1 = Error.
529 #
530 # ACCESSED GLOBAL VARIABLES:
531 #    None.
532 # ============================================================================
533 sub HoldoverTrack {
534    my($ButtonInput, $TurnoutData, $TrackData, $GpioData) = @_;
535    my($result, $button, $route, $gpio, $value, $turnout, $position, $check);
536    my($moveResult);
537    my(%routes) = (
538       '04' => 'T01:Close,T02:Close,GPIO6_PR02:0',
539       '05' => 'T01:Close,T02:Open,GPIO5_PR01:1',
540       '06' => 'T01:Open,T03:Close,GPIO5_PR01:0',
```

```
541         '07' => 'T01:Open,T03:Open,GPIO6_PR02:1');
542     my(@route) = ();
543     &DisplayDebug(2, "HoldoverTrack entry ...  ButtonInput: '$ButtonInput'");
544
545     # Process new button press.
546     if ($ButtonInput =~ m/s(04)/ or $ButtonInput =~ m/s(05)/ or
547         $ButtonInput =~ m/s(06)/ or $ButtonInput =~ m/s(07)/) {
548        $button = $1;
549        $route = join('', 'R', ($button - 3));
550        &DisplayMessage("HoldoverTrack, route $route requested.");
551
552        # ---------------------------------
553        # If a route is currently active, reset and done.
554        if ($$TrackData{'01'}{'RouteLocked'} == 1) {
555           &PlaySound("Unlock.wav");
556           $$TrackData{'01'}{'RouteTime'} = time - 1;     # Reset route timeout
557           $$GpioData{'GPIO26_HLCK'}{'Obj'}->write(0);    # Button LED off
558           $$TrackData{'01'}{'RouteLocked'} = 0;
559           &DisplayMessage("HoldoverTrack, route unlocked by button.");
560           return 0;
561        }
562        @route = split(",", $routes{$button});
563
564        # ---------------------------------
565        # Set turnouts.
566        if ($#route >= 0) {
567           foreach my $device (@route) {
568              if ($device =~ m/^T(\d+):(.+)/) {
569                 $turnout = $1;
570                 $position = $2;
571                 $moveResult = &MoveTurnout($position, $turnout, $TurnoutData);
572                 if ($moveResult == 1) {
573                    &DisplayError("ProcessHoldover, Failed to set " .
574                                  "turnout $turnout to $position");
575                 }
576                 else {
577                    &DisplayMessage("ProcessHoldover, turnout " .
578                                    "$turnout set to $position.");
579                 }
580              }
581              elsif ($device =~ m/^(GPIO.+?):(\d)/) {
582                 $gpio = $1;
583                 $value = $2;
584                 $$GpioData{$gpio}{'Obj'}->write($value);  # Set polarity relay.
585                 $check = $$GpioData{$gpio}{'Obj'}->read;  # Readback and check.
586                 if ($check != $value) {
587                    &DisplayError("HoldoverTrack, Failed to set " .
588                                  "power relay $gpio to $value");
589                 }
590                 else {
591                    &DisplayMessage("ProcessHoldover, relay $gpio " .
592                                    "set to $value.");
593                 }
594              }
595           }
596
597           $$GpioData{'GPIO26_HLCK'}{'Obj'}->write(1);   # Button LED on
598           $$TrackData{'01'}{'RouteLocked'} = 1;
599           $$TrackData{'01'}{'RouteTime'} = time + 60;   # Set route timeout
600           &DisplayMessage("HoldoverTrack, route $route is locked.");
```

```
601             &PlaySound("Lock.wav");
602          }
603          else {
604             &DisplayMessage("HoldoverTrack, $route is invalid for " .
605                             "train movement direction.");
606             &PlaySound("GE.wav");
607          }
608       }
609
610       # If a route is set, and has timed out, reset the lock.
611       else {
612          if ($$TrackData{'01'}{'RouteLocked'} == 1 and
613              $$TrackData{'01'}{'RouteTime'} < time) {
614             &PlaySound("Unlock.wav");
615             $$TrackData{'01'}{'RouteLocked'} = 0;
616             $$GpioData{'GPIO26_HLCK'}{'Obj'}->write(0);        # Button LED off
617             &DisplayMessage("HoldoverTrack, route unlocked by timeout.");
618          }
619       }
620       return 0;
621    }
622
623    # ============================================================================
624    # FUNCTION:  MidwayTrack
625    #
626    # DESCRIPTION:
627    #    This routine processes the user buttons associated with turnouts T05 and
628    #    T06. These buttons, 00 and 01, are used to manually position the turnout
629    #    or lock it in its current position.
630    #
631    # CALLING SYNTAX:
632    #    $result = &MidwayTrack($ButtonInput, \%ButtonData, \%TurnoutData,
633    #                           \%TrackData, \%SensorBit, \%SensorState);
634    #
635    # ARGUMENTS:
636    #    $ButtonInput        User entered button input, if any.
637    #    $ButtonData         Pointer to %ButtonData hash.
638    #    $TurnoutData        Pointer to %TurnoutData hash.
639    #    $TrackData          Pointer to %TrackData hash.
640    #    $SensorBit          Pointer to %SensorBit hash.
641    #    $SensorState        Pointer to %SensorState hash.
642    #
643    # RETURNED VALUES:
644    #    0 = Success,  1 = Error.
645    #
646    # ACCESSED GLOBAL VARIABLES:
647    #    None.
648    # ============================================================================
649    sub MidwayTrack {
650       my($ButtonInput, $ButtonData, $TurnoutData, $TrackData, $SensorBit,
651          $SensorState) = @_;
652       my($pressType, $moveResult, $turnout1, $turnout2, $position);
653
654       &DisplayDebug(2, "MidwayTrack entry ...  ButtonInput: " .
655                                "'$ButtonInput'");
656
657       # Parse and process the button input.
658       if ($ButtonInput =~ m/(d)(00)/ or $ButtonInput =~ m/(d)(01)/ or
659           $ButtonInput =~ m/(s)(00)/ or $ButtonInput =~ m/(s)(01)/) {
660          $pressType = $1;
```

```
661         $turnout1 = $$ButtonData{$2}{'Turnout1'};
662         $turnout2 = $$ButtonData{$2}{'Turnout2'};
663         &DisplayDebug(1, "MidwayTrack, pressType: $pressType   " .
664                         "turnout1: $turnout1   turnout2: $turnout2");
665
666         # A single button press unlocks the turnout. ProcessMidway code
667         # will reposition the turnout to its inactive position.
668         if ($pressType eq 's' and $$TrackData{$turnout1}{'Locked'} == 1) {
669            $$TrackData{$turnout1}{'Locked'} = 0;
670            $$TrackData{$turnout1}{'ManualSet'} = 0;
671            &DisplayMessage("MidwayTrack, turnout $turnout1 is unlocked.");
672            &PlaySound("Unlock.wav");
673            return 0;
674         }
675
676         # Ignore the button if $turnout1 or $turnout2 has a timeout or
677         # inprogress movement.
678         return 0 if (&CheckTurnout($turnout1, 'MidwayTrack', $TurnoutData,
679                      $TrackData, $SensorBit, $SensorState) or &CheckTurnout(
680                      $turnout2, 'MidwayTrack', $TurnoutData, $TrackData,
681                      $SensorBit, $SensorState));
682
683         # Reposition $turnout2 if in a blocking position.
684         if ($$TurnoutData{$turnout2}{'Pos'} ne
685             $$TurnoutData{$turnout2}{ $$TrackData{$turnout2}{'Inactive'} }) {
686            $moveResult = &MoveTurnout($$TrackData{$turnout2}{'Inactive'},
687                                       $turnout2, $TurnoutData);
688            if ($moveResult == 1) {
689               &DisplayError("MidwayTrack, Failed to set turnout $turnout2 to " .
690                             "$$TrackData{$turnout2}{'Inactive'}.");
691               &PlaySound("GE.wav");
692               return 0;
693            }
694            if ($$TrackData{$turnout2}{'Locked'} == 1) {
695               $$TrackData{$turnout2}{'Locked'} = 0;
696               &DisplayMessage("MidwayTrack, turnout $turnout2 is unlocked.");
697            }
698            $$TrackData{$turnout2}{'ManualSet'} = 0;
699         }
700
701         # If double button press, move $turnout1 to active position and
702         # then lock it.
703         if ($pressType eq 'd') {
704            if ($$TurnoutData{$turnout1}{'Pos'} ne
705                $$TurnoutData{$turnout1}{ $$TrackData{$turnout1}{'Active'} }) {
706               $moveResult = &MoveTurnout($$TrackData{$turnout1}{'Active'},
707                                          $turnout1, $TurnoutData);
708               if ($moveResult == 1) {
709                  &DisplayError("MidwayTrack, Failed to set " .
710                                "turnout $turnout1 to " .
711                                "$$TrackData{$turnout1}{'Active'}.");
712                  &PlaySound("GE.wav");
713                  return 0;
714               }
715            }
716            $$TrackData{$turnout1}{'Locked'} = 1;
717            &DisplayMessage("MidwayTrack, turnout $turnout1 is locked.");
718            &PlaySound("Lock.wav");
719            return 0;
720         }
```

```
721
722         # Toggle $turnout1 position for single button press.
723         $$TrackData{$turnout1}{'ManualSet'} = 1;
724         if ($$TurnoutData{$turnout1}{'Pos'} == $$TurnoutData{$turnout1}{'Open'}) {
725            $position = 'Close';
726         }
727         else {
728            $position = 'Open';
729         }
730         $moveResult = &MoveTurnout($position, $turnout1, $TurnoutData);
731         if ($moveResult == 1) {
732            &DisplayError("MidwayTrack, Failed to set turnout $turnout1 to " .
733                          "$position");
734            &PlaySound("GE.wav");
735         }
736         else {
737            &DisplayMessage("MidwayTrack, turnout $turnout1 set to $position.");
738            &PlaySound("A_.wav");
739         }
740      }
741      return 0;
742   }
743
744   # ============================================================================
745   # FUNCTION:  WyeTrack
746   #
747   # DESCRIPTION:
748   #    This routine processes the user buttons associated with the T07 turnout.
749   #    These buttons, 02 and 03, are used to manually set the turnout position
750   #    which selects the yard approach track to be used.
751   #
752   # CALLING SYNTAX:
753   #    $result = &WyeTrack($ButtonInput, \%ButtonData, \%TurnoutData,
754   #                        \%TrackData, \%SensorBit, \%SensorState);
755   #
756   # ARGUMENTS:
757   #    $ButtonInput        User entered button input, if any.
758   #    $ButtonData         Pointer to %ButtonData hash.
759   #    $TurnoutData        Pointer to %TurnoutData hash.
760   #    $TrackData          Pointer to %TrackData hash.
761   #    $SensorBit          Pointer to %SensorBit hash.
762   #    $SensorState        Pointer to %SensorState hash.
763   #
764   # RETURNED VALUES:
765   #    0 = Success,  1 = Error.
766   #
767   # ACCESSED GLOBAL VARIABLES:
768   #    None.
769   # ============================================================================
770   sub WyeTrack {
771      my($ButtonInput, $ButtonData, $TurnoutData, $TrackData, $SensorBit,
772         $SensorState) = @_;
773      my($moveResult, $button, $turnout, $position);
774
775      &DisplayDebug(2, "WyeTrack entry ...   ButtonInput: '$ButtonInput'");
776
777      # ---------------------------------
778      # Process single press button input.
779      if ($ButtonInput =~ m/s(02)/ or $ButtonInput =~ m/s(03)/) {
780         $button = $1;
```

```
781         $turnout = $$ButtonData{$button}{'Turnout'};
782         &DisplayDebug(0, "WyeTrack, button: $button   turnout: $turnout");
783
784         # Ignore the button if turnout move or train transit is inprogress.
785         return 0 if (&CheckTurnout($turnout, 'WyeTrack', $TurnoutData, $TrackData,
786                     $SensorBit, $SensorState));
787
788         if ($button eq '02') {
789            $position = 'Open';
790         }
791         else {
792            $position = 'Close';
793         }
794
795         # Move turnout if necessary.
796         if ($$TurnoutData{$turnout}{'Pos'} ne $$TurnoutData{$turnout}{$position}) {
797            $moveResult = &MoveTurnout($position, $turnout, $TurnoutData);
798            if ($moveResult == 1) {
799               &DisplayError("WyeTrack, Failed to set turnout $turnout to $position");
800               &PlaySound("GE.wav");
801            }
802            else {
803               &DisplayMessage("WyeTrack, turnout $turnout set to $position.");
804               &PlaySound("A_.wav");
805            }
806         }
807         else {
808            &DisplayMessage("WyeTrack, turnout $turnout already at $position.");
809            &PlaySound("A_.wav");
810         }
811      }
812      return 0;
813   }
814
815   # ============================================================================
816   # FUNCTION:  CheckTurnout
817   #
818   # DESCRIPTION:
819   #    This routine is shared code used by MidwayTrack and WyeTrack to check for
820   #    an inprogress turnout operation. This check is performed as part of button
821   #    input processing. Warning tone and console message is output if necessary.
822   #
823   # CALLING SYNTAX:
824   #    $result = &CheckTurnout($Turnout, $Caller, \%TurnoutData, \%TrackData,
825   #                            \%SensorBit, \%SensorState);
826   #
827   # ARGUMENTS:
828   #    $Turnout            Turnout number.
829   #    $Caller             Name of calling routine.
830   #    $TurnoutData        Pointer to %TurnoutData hash.
831   #    $TrackData          Pointer to %TrackData hash.
832   #    $SensorBit          Pointer to %SensorBit hash.
833   #    $SensorState        Pointer to %SensorState hash.
834   #
835   # RETURNED VALUES:
836   #    0 = no inprogress operation,  1 = inprogress operation.
837   #
838   # ACCESSED GLOBAL VARIABLES:
839   #    None.
840   # ============================================================================
```

```
841  sub CheckTurnout {
842     my($Turnout, $Caller, $TurnoutData, $TrackData, $SensorBit, $SensorState) = @_;
843
844     if ($$TurnoutData{$Turnout}{'Pid'} != 0) {
845        &DisplayMessage("$Caller, turnout $Turnout position change is inprogress.");
846        &PlaySound("GE.wav");
847        return 1;
848     }
849     if ($$TrackData{$Turnout}{'Timeout'} > time) {
850        &DisplayMessage("$Caller, train transit of turnout $Turnout is inprogress.");
851        &PlaySound("GE.wav");
852        return 1;
853     }
854     if (&GetSensorBit($$TurnoutData{$Turnout}{'Sensor'}, $SensorBit,
855                       $SensorState) == 1) {
856        &DisplayMessage("$Caller, $Turnout sensor " .
857                        $$TurnoutData{$Turnout}{'Sensor'} . " is active.");
858        &PlaySound("GE.wav");
859        return 1;
860     }
861     return 0;
862  }
863
864  return 1;
865
```