

```

1  # =====
2  # FILE: DnB_Mainline.pm                                     8/12/2020
3  #
4  # SERVICES:  DnB TRACK PROCESSING FUNCTIONS
5  #
6  # DESCRIPTION:
7  #   This perl module provides mainline track processing related functions used
8  #   by the DnB model railroad control program.
9  #
10 # PERL VERSION: 5.24.1
11 #
12 # =====
13 use strict;
14 # -----
15 # Package Declaration
16 # -----
17 package DnB_Mainline;
18 require Exporter;
19 our @ISA = qw(Exporter);
20
21 our @EXPORT = qw(
22     ProcessHoldover
23     ProcessMidway
24     ProcessWye
25     HoldoverTrack
26     MidwayTrack
27     WyeTrack
28     CheckTurnout
29 );
30
31 use DnB_Message;
32 use DnB_Sensor;
33 use DnB_Turnout;
34
35 # =====
36 # FUNCTION:  ProcessHoldover
37 #
38 # DESCRIPTION:
39 #   This routine performs the operational functions related to the holdover
40 #   track section. Functions include turnout point positioning and setting
41 #   of track power polarity.
42 #
43 #   Retriggerable timers in %TrackData are used for S1, S2, and S3 to ensure
44 #   that a route is set only once for as long as the train activates the
45 #   sensor.
46 #
47 #   The S1, S2, and S3 sensors also retrigger the 'RouteTime' if a temporary
48 #   holdover route has be set ('RouteLocked') via button input. When a route
49 #   has been set, no other ProcessHoldover functions are performed.
50 #
51 # CALLING SYNTAX:
52 #   $result = &ProcessHoldover(\%TrackData, \%SensorBit, \%SensorState,
53 #                               \%TurnoutData, \%GpioData);
54 #
55 # ARGUMENTS:
56 #   $TrackData      Pointer to %TrackData hash.
57 #   $SensorBit      Pointer to %SensorBit hash.
58 #   $SensorState    Pointer to %SensorState hash.
59 #   $TurnoutData    Pointer to %TurnoutData hash.
60 #   $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, $check);
71     my(%bitPos) = ('B1' => '00', 'B2' => '01', 'B3' => '02', 'S1' => '16',
72                   'S2' => '17', 'S3' => '18');
73     my(%routes) = (
74         'InB1' => 'T01:Close,T02:Close,T03:Close,GPI06_PR02:0',
75         'InB2' => 'T01:Open,T03:Close,T02:Close,GPI05_PR01:0',
76         'OutB1' => 'T03:Open,T01:Open,GPI06_PR02:1',
77         'OutB2' => 'T02:Open,T01:Close,GPI05_PR01:1');
78
79     my(@route) = ();
80     my($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     }
106     @route = split(",", $routes{'InB1'}); # Default turnout positions.
107 }
108
109 # --- Sensor S1 processing. -----
110 if (&GetSensorBit($bitPos{'S1'}, $SensorBit, $SensorState) == 1) {
111     &DisplayDebug(1, "ProcessHoldover, S1 is active.");
112     if ($$TrackData{'01'}{'Direction'} eq 'Out' and
113         $$TrackData{'01'}{'WaitB3Inact'} == 0) {
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

```

```

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

```

```

181 # Above note for S2 applies here also.
182
183 elseif (&GetSensorBit($bitPos{'S3'}, $SensorBit, $SensorState) == 1) {
184     &DisplayDebug(1, "ProcessHoldover, S3 is active.");
185     if ($$TrackData{'03'}{'Timeout'} < $cTime) { # If route not already set.
186         &DisplayMessage("ProcessHoldover, routing outbound B1 train to B3.");
187         @route = split(",", $routes{'OutB1'});
188         $$TrackData{'01'}{'Direction'} = 'Out';
189     }
190     $$TrackData{'03'}{'Timeout'} = $cTime + 3; # Disable S3 processing.
191 }
192
193 # --- Set turnouts and relays if @route is specified. -----
194 if ($#route >= 0) {
195     foreach my $device (@route) {
196         if ($device =~ m/^(T(\d+):(.+)/) {
197             $turnout = $1;
198             $position = $2;
199             $moveResult = &MoveTurnout($position, $turnout, $TurnoutData);
200             if ($moveResult == 1) {
201                 &DisplayError("ProcessHoldover, Failed to set turnout " .
202                     "$turnout to $position");
203             }
204             else {
205                 &DisplayMessage("ProcessHoldover, turnout $turnout " .
206                     "set to $position.");
207             }
208         }
209         elseif ($device =~ m/^(GPIO.+?):(\d+)/) {
210             $gpio = $1;
211             $value = $2;
212             $$GpioData{$gpio}{'Obj'}->write($value); # Set power polarity relay.
213             $check = $$GpioData{$gpio}{'Obj'}->read; # Readback and check.
214             if ($check != $value) {
215                 &DisplayError("ProcessHoldover, Failed to set power " .
216                     "relay $gpio to $value");
217             }
218             else {
219                 &DisplayMessage("ProcessHoldover, relay $gpio " .
220                     "set to $value.");
221             }
222         }
223         else {
224             &DisplayError("ProcessHoldover, Invalid S1 route entry: " .
225                 "$device");
226         }
227     }
228 }
229 return 0;
230 }
231
232 # =====
233 # FUNCTION: ProcessMidway
234 #
235 # DESCRIPTION:
236 # This routine performs the operational functions related to the midway
237 # track section. Functions include turnout point positioning. A turnout
238 # is not processed if previously locked by user button input.
239 #
240 # Retriggerable timers in %TrackData are used for S5 and S6 to ensure

```

```

241 # that a route is set only once for as long as the train activates the
242 # sensor.
243 #
244 # The respective turnout it set back to the Inactive position after its
245 # timer expires. This action is inhibited by a manually set position. In
246 # this case, reposition will occur after a 2nd timeout cycle.
247 #
248 # CALLING SYNTAX:
249 # $result = &ProcessMidway(\%TrackData, \%SensorBit, \%SensorState,
250 # \%TurnoutData);
251 #
252 # ARGUMENTS:
253 # $TrackData Pointer to %TrackData hash.
254 # $SensorBit Pointer to %SensorBit hash.
255 # $SensorState Pointer to %SensorState hash.
256 # $TurnoutData Pointer to %TurnoutData hash.
257 #
258 # RETURNED VALUES:
259 # 0 = Success, 1 = Error.
260 #
261 # ACCESSED GLOBAL VARIABLES:
262 # None.
263 # =====
264 sub ProcessMidway {
265     my($TrackData, $SensorBit, $SensorState, $TurnoutData) = @_;
266     my($moveResult);
267     my(%bitPos) = ('S5' => '20', 'S6' => '21');
268     my($cTime) = time;
269
270     &DisplayDebug(2, "ProcessMidway entry ...");
271
272     # --- Sensor S5 processing. -----
273     if ($$TrackData{'05'}{'Locked'} == 0) {
274         if (&GetSensorBit($bitPos{'S5'}, $SensorBit, $SensorState) == 1) {
275             &DisplayDebug(1, "ProcessMidway, S5 is active.");
276
277             # Move turnout if no inprogress timeout. Otherwise, restart timeout.
278             if ($$TurnoutData{'05'}{'Pos'} !=
279                 $$TurnoutData{'05'}{ $$TrackData{'05'}{'Active'} } and
280                 $$TrackData{'05'}{'Timeout'} < $cTime) {
281
282                 $moveResult = &MoveTurnout($$TrackData{'05'}{'Active'}, '05',
283                                         $TurnoutData);
284
285                 if ($moveResult == 1) {
286                     &DisplayError("ProcessMidway, Failed to set turnout " .
287                                 "05 to $$TrackData{'05'}{'Active'}.");
288                 }
289                 else {
290                     &DisplayMessage("ProcessMidway, turnout 05 set to " .
291                                   "active position $$TrackData{'05'}{'Active'}.");
292                 }
293                 $$TrackData{'05'}{'Timeout'} = $cTime + 15; # Retrigger timeout.
294                 $$TrackData{'05'}{'ManualSet'} = 0;
295             }
296             else {
297
298                 # Reset turnout if a timeout has completed and turnout is not in the
299                 # Inactive position. Check for turnout Pid 0 prevents additional turnout
300                 # setting during the move period.

```

```

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

```

```

361         &DisplayMessage("ProcessMidway, turnout 06 set to " .
362             "inactive position $$TrackData{'06'}{'Inactive'}.");
363     }
364 }
365 }
366 }
367 return 0;
368 }
369
370 # =====
371 # FUNCTION: ProcessWye
372 #
373 # DESCRIPTION:
374 #   This routine performs the operational functions related to the wye track
375 #   section. Functions include turnout point positioning and setting of track
376 #   power polarity.
377 #
378 # CALLING SYNTAX:
379 #   $result = &ProcessWye(\%TrackData, \%SensorBit, \%SensorState,
380 #                         \%TurnoutData, \%GpioData);
381 #
382 # ARGUMENTS:
383 #   $TrackData      Pointer to %TrackData hash.
384 #   $SensorBit      Pointer to %SensorBit hash.
385 #   $SensorState    Pointer to %SensorState hash.
386 #   $TurnoutData    Pointer to %TurnoutData hash.
387 #   $GpioData       Pointer to %GpioData hash. (polarity relays)
388 #
389 # RETURNED VALUES:
390 #   0 = Success, 1 = Error.
391 #
392 # ACCESSED GLOBAL VARIABLES:
393 #   None.
394 # =====
395 sub ProcessWye {
396     my($TrackData, $SensorBit, $SensorState, $TurnoutData, $GpioData) = @_;
397     my($moveResult);
398     my(%bitPos) = ('S7' => '22', 'S8' => '23', 'S9' => '24');
399     my($cTime) = time;
400
401     &DisplayDebug(2, "ProcessWye entry ...");
402
403     # --- Sensor S7 processing. -----
404     if (&GetSensorBit($bitPos{'S7'}, $SensorBit, $SensorState) == 1) {
405         if ($$TrackData{'07'}{'Timeout'} < $cTime) {
406             &DisplayDebug(1, "ProcessWye, S7 is active.");
407
408             # Set the polarity relay based on current T7 position.
409             if ($$TurnoutData{'07'}{'Pos'} == $$TurnoutData{'07'}{'Close'}) {
410                 if ($$TrackData{'07'}{'Polarity'} != 0) {
411                     $$GpioData[GPI013_PR03]{'Obj'}->write(0); # Set relay control bit.
412                     if ($$GpioData[GPI013_PR03]{'Obj'}->read != 0) { # Readback check.
413                         &DisplayError("ProcessWye S7, Failed to set power " .
414                             "relay GPI013_PR03 to 0");
415                     }
416                 } else {
417                     &DisplayMessage("ProcessWye S7, power relay " .
418                         "GPI013_PR03 set to 0.");
419                 }
420                 $$TrackData{'07'}{'Polarity'} = 0;

```

```

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

```



```

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

```

```

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

```

```

601     }
602     else {
603         &DisplayMessage("HoldoverTrack, $route is invalid for " .
604             "train movement direction.");
605         &PlaySound("GE.wav");
606     }
607 }
608
609 # If a route is set, and has timed out, reset the lock.
610 else {
611     if ($$TrackData{'01'}{'RouteLocked'} == 1 and
612         $$TrackData{'01'}{'RouteTime'} < time) {
613         &PlaySound("Unlock.wav");
614         $$TrackData{'01'}{'RouteLocked'} = 0;
615         $$GpioData{'GPIO26_HLCK'}{'Obj'}->write(0);          # Button LED off
616         &DisplayMessage("HoldoverTrack, route unlocked by timeout.");
617     }
618 }
619 return 0;
620 }
621
622 # =====
623 # FUNCTION:   MidwayTrack
624 #
625 # DESCRIPTION:
626 #   This routine processes the user buttons associated with turnouts T05 and
627 #   T06. These buttons, 00 and 01, are used to manually position the turnout
628 #   or lock it in its current position.
629 #
630 # CALLING SYNTAX:
631 #   $result = &MidwayTrack($ButtonInput, \%ButtonData, \%TurnoutData,
632 #       \%TrackData, \%SensorBit, \%SensorState);
633 #
634 # ARGUMENTS:
635 #   $ButtonInput      User entered button input, if any.
636 #   $ButtonData       Pointer to %ButtonData hash.
637 #   $TurnoutData      Pointer to %TurnoutData hash.
638 #   $TrackData        Pointer to %TrackData hash.
639 #   $SensorBit        Pointer to %SensorBit hash.
640 #   $SensorState      Pointer to %SensorState hash.
641 #
642 # RETURNED VALUES:
643 #   0 = Success, 1 = Error.
644 #
645 # ACCESSED GLOBAL VARIABLES:
646 #   None.
647 # =====
648 sub MidwayTrack {
649     my($ButtonInput, $ButtonData, $TurnoutData, $TrackData, $SensorBit,
650         $SensorState) = @_;
651     my($pressType, $moveResult, $turnout1, $turnout2, $position);
652
653     &DisplayDebug(2, "MidwayTrack entry ... ButtonInput: " .
654         "'$ButtonInput'");
655
656     # Parse and process the button input.
657     if ($ButtonInput =~ m/(d)(00)/ or $ButtonInput =~ m/(d)(01)/ or
658         $ButtonInput =~ m/(s)(00)/ or $ButtonInput =~ m/(s)(01)/) {
659         $pressType = $1;
660         $turnout1 = $$ButtonData{$2}{'Turnout1'};

```

```

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

```

```

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

```

```

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

```

```

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

```