

```

1  # =====
2  # FILE: DnB_Mainline.pm                                     9/28/2020
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     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, $SensorState) == 1) {
126         &DisplayMessage("ProcessHoldover, inbound and outbound train wreck!");
127         &PlaySound("TrainWreck3.wav");
128     }
129
130     # Alternate holdover tracks if both are unoccupied. Otherwise, route
131     # inbound train to an available track.
132     elseif (&GetSensorBit($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             $$TrackData{'01'}{'Last'} = 'B1';
142             @route = split(",", $routes{'InB1'});
143         }
144     }
145     elseif (&GetSensorBit($bitPos{'B1'}, $SensorBit, $SensorState) == 0) {
146         &DisplayMessage("ProcessHoldover, routing inbound train to B1.");
147         @route = split(",", $routes{'InB1'});
148     }
149     elseif (&GetSensorBit($bitPos{'B2'}, $SensorBit, $SensorState) == 0) {
150         &DisplayMessage("ProcessHoldover, routing inbound train to B2.");
151         @route = split(",", $routes{'InB2'});
152     }
153     else {
154         &DisplayMessage("ProcessHoldover, inbound sidings " .
155             "full train wreck!");
156         &PlaySound("TrainWreck3.wav");
157     }
158 }
159 $$TrackData{'01'}{'Timeout'} = $cTime + 10; # Disable S1 processing.
160 }
161 }
162
163 # --- Sensor S2 processing. -----
164 # Note: A retriggerable timer is used to prevent multiple turnout settings. It
165 # is possible for this timer to expire for a slow or stopped train that
166 # leaves the sensor unblocked. No adverse affect, just some CPU cycles.
167 # The timer is used instead of the 'Out' direction state so that a second
168 # siding departure can occur while the previous train still occupies the
169 # B3 block.
170
171 elseif (&GetSensorBit($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

```

```

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

```

```

241 #   Retriggerable timers in %TrackData are used for S5 and S6 to ensure
242 #   that a route is set only once for as long as the train activates the
243 #   sensor.
244 #
245 #   The respective turnout is set back to the Inactive position after its
246 #   timer expires. This action is inhibited by a manually set position. In
247 #   this case, reposition will occur after a 2nd timeout cycle.
248 #
249 # CALLING SYNTAX:
250 #   $result = &ProcessMidway(\%TrackData, \%SensorBit, \%SensorState,
251 #                           \%TurnoutData);
252 #
253 # ARGUMENTS:
254 #   $TrackData      Pointer to %TrackData hash.
255 #   $SensorBit      Pointer to %SensorBit hash.
256 #   $SensorState    Pointer to %SensorState hash.
257 #   $TurnoutData    Pointer to %TurnoutData hash.
258 #
259 # RETURNED VALUES:
260 #   0 = Success, 1 = Error.
261 #
262 # ACCESSED GLOBAL VARIABLES:
263 #   None.
264 # =====
265 sub ProcessMidway {
266     my($TrackData, $SensorBit, $SensorState, $TurnoutData) = @_;
267     my($moveResult);
268     my(%bitPos) = ('S5' => '20', 'S6' => '21');
269     my($cTime) = time;
270
271     &DisplayDebug(2, "ProcessMidway entry ...");
272
273     # --- Sensor S5 processing. -----
274     if ($$TrackData{'05'}{'Locked'} == 0) {
275         if (&GetSensorBit($bitPos{'S5'}, $SensorBit, $SensorState) == 1) {
276             &DisplayDebug(1, "ProcessMidway, S5 is active.");
277
278             # Move turnout if no inprogress timeout. Otherwise, restart timeout.
279             if ($$TurnoutData{'05'}{'Pos'} !=
280                 $$TurnoutData{'05'}{'Active'}) and
281                 $$TrackData{'05'}{'Timeout'} < $cTime) {
282
283                 $moveResult = &MoveTurnout($$TrackData{'05'}{'Active'}, '05',
284                                         $TurnoutData);
285
286                 if ($moveResult == 1) {
287                     &DisplayError("ProcessMidway, Failed to set turnout " .
288                                 "05 to $$TrackData{'05'}{'Active'}.");
289                 }
290             }
291             else {
292                 &DisplayMessage("ProcessMidway, turnout 05 set to " .
293                                "active position $$TrackData{'05'}{'Active'}.");
294             }
295             $$TrackData{'05'}{'Timeout'} = $cTime + 15; # Retrigger timeout.
296             $$TrackData{'05'}{'ManualSet'} = 0;
297         }
298     }
299     else {
300         # Reset turnout if a timeout has completed and turnout is not in the
301         # 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             }

```

```

361         else {
362             &DisplayMessage("ProcessMidway, turnout 06 set to " .
363                 "inactive position $$TrackData{'06'}{'Inactive'}.");
364         }
365     }
366 }
367 }
368 return 0;
369 }
370
371 # =====
372 # FUNCTION: ProcessWye
373 #
374 # DESCRIPTION:
375 #     This routine performs the operational functions related to the wye track
376 #     section. Functions include turnout point positioning and setting of track
377 #     power polarity.
378 #
379 # CALLING SYNTAX:
380 #     $result = &ProcessWye(\%TrackData, \%SensorBit, \%SensorState,
381 #         \%TurnoutData, \%GpioData);
382 #
383 # ARGUMENTS:
384 #     $TrackData      Pointer to %TrackData hash.
385 #     $SensorBit      Pointer to %SensorBit hash.
386 #     $SensorState    Pointer to %SensorState hash.
387 #     $TurnoutData    Pointer to %TurnoutData hash.
388 #     $GpioData       Pointer to %GpioData hash. (polarity relays)
389 #
390 # RETURNED VALUES:
391 #     0 = Success, 1 = Error.
392 #
393 # ACCESSED GLOBAL VARIABLES:
394 #     None.
395 # =====
396 sub ProcessWye {
397     my($TrackData, $SensorBit, $SensorState, $TurnoutData, $GpioData) = @_;
398     my($moveResult);
399     my(%bitPos) = ('S7' => '22', 'S8' => '23', 'S9' => '24');
400     my($cTime) = time;
401
402     &DisplayDebug(2, "ProcessWye entry ...");
403
404     # --- Sensor S7 processing. -----
405     if (&GetSensorBit($bitPos{'S7'}, $SensorBit, $SensorState) == 1) {
406         if ($$TrackData{'07'}{'Timeout'} < $cTime) {
407             &DisplayDebug(1, "ProcessWye, S7 is active.");
408
409             # Set the polarity relay based on current T7 position.
410             if ($$TurnoutData{'07'}{'Pos'} == $$TurnoutData{'07'}{'Close'}) {
411                 if ($$TrackData{'07'}{'Polarity'} != 0) {
412                     $$GpioData{GPIO13_PR03}{'Obj'}->write(0); # Set relay control bit.
413                     if ($$GpioData{GPIO13_PR03}{'Obj'}->read != 0) { # Readback check.
414                         &DisplayError("ProcessWye S7, Failed to set power " .
415                             "relay GPIO13_PR03 to 0");
416                     }
417                 } else {
418                     &DisplayMessage("ProcessWye S7, power relay " .
419                         "GPIO13_PR03 set to 0.");
420                 }

```

```

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,GPI06_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         $$GpioData{'GPI026_HLCK'}{'Obj'}->write(0);      # Button LED off
558         $$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         $$GpioData{'GPI026_HLCK'}{'Obj'}->write(1);      # Button LED on
598         $$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 #                       \%GpioData);
756 #
757 # ARGUMENTS:
758 #   $ButtonInput      User entered button input, if any.
759 #   $ButtonData        Pointer to %ButtonData hash.
760 #   $TurnoutData       Pointer to %TurnoutData hash.
761 #   $TrackData         Pointer to %TrackData hash.
762 #   $SensorBit         Pointer to %SensorBit hash.
763 #   $SensorState       Pointer to %SensorState hash.
764 #   $GpioData          Pointer to %GpioData hash. (polarity relays)
765 #
766 # RETURNED VALUES:
767 #   0 = Success, 1 = Error.
768 #
769 # ACCESSED GLOBAL VARIABLES:
770 #   None.
771 # =====
772 sub WyeTrack {
773     my($ButtonInput, $ButtonData, $TurnoutData, $TrackData, $SensorBit,
774         $SensorState, $GpioData) = @_;
775     my($moveResult, $button, $turnout, $position, $polarity);
776
777     &DisplayDebug(2, "WyeTrack entry ...   ButtonInput: '$ButtonInput'");
778
779     # -----
780     # Process single press button input.

```

```

781 if ($ButtonInput =~ m/s(02)/ or $ButtonInput =~ m/s(03)/) {
782     $button = $1;
783     $turnout = $$ButtonData{$button}{'Turnout'};
784     &DisplayDebug(0, "WyeTrack, button: $button    turnout: $turnout");
785
786     # Ignore the button if turnout move or train transit is inprogress.
787     return 0 if (&CheckTurnout($turnout, 'WyeTrack', $TurnoutData, $TrackData,
788         $SensorBit, $SensorState));
789
790     if ($button eq '02') {
791         $position = 'Open';
792         $polarity = 1;
793     }
794     else {
795         $position = 'Close';
796         $polarity = 0;
797     }
798
799     # Move turnout if necessary.
800     if ($$TurnoutData{$turnout}{'Pos'} ne $$TurnoutData{$turnout}{$position}) {
801         $moveResult = &MoveTurnout($position, $turnout, $TurnoutData);
802         if ($moveResult == 1) {
803             &DisplayError("WyeTrack, Failed to set turnout $turnout to $position");
804             &PlaySound("GE.wav");
805         }
806         else {
807             &DisplayMessage("WyeTrack, turnout $turnout set to $position.");
808             &PlaySound("A_.wav");
809         }
810     }
811     else {
812         &DisplayMessage("WyeTrack, turnout $turnout already at $position.");
813         &PlaySound("A_.wav");
814     }
815
816     # Change power polarity relay if necessary.
817     if ($$TrackData{'07'}{'Polarity'} != $polarity) {
818         $$GpioData{GPIO13_PR03}{'Obj'}->write($polarity); # Set relay control bit.
819         if ($$GpioData{GPIO13_PR03}{'Obj'}->read != $polarity) { # Readback check.
820             &DisplayError("ProcessWye S7, Failed to set power " .
821                 "relay GPIO13_PR03 to $polarity");
822         }
823         else {
824             &DisplayMessage("ProcessWye S7, power relay " .
825                 "GPIO13_PR03 set to $polarity.");
826         }
827         $$TrackData{'07'}{'Polarity'} = $polarity;
828     }
829 }
830 return 0;
831 }
832
833 # =====
834 # FUNCTION:  CheckTurnout
835 #
836 # DESCRIPTION:
837 #     This routine is shared code used by MidwayTrack and WyeTrack to check for
838 #     an inprogress turnout operation. This check is performed as part of button
839 #     input processing. Warning tone and console message is output if necessary.
840 #

```

```

841 # CALLING SYNTAX:
842 #   $result = &CheckTurnout($Turnout, $Caller, \%TurnoutData, \%TrackData,
843 #                           \%SensorBit, \%SensorState);
844 #
845 # ARGUMENTS:
846 #   $Turnout      Turnout number.
847 #   $Caller       Name of calling routine.
848 #   $TurnoutData  Pointer to %TurnoutData hash.
849 #   $TrackData    Pointer to %TrackData hash.
850 #   $SensorBit    Pointer to %SensorBit hash.
851 #   $SensorState  Pointer to %SensorState hash.
852 #
853 # RETURNED VALUES:
854 #   0 = no inprogress operation,  1 = inprogress operation.
855 #
856 # ACCESSED GLOBAL VARIABLES:
857 #   None.
858 # =====
859 sub CheckTurnout {
860     my($Turnout, $Caller, $TurnoutData, $TrackData, $SensorBit, $SensorState) = @_;
861
862     if ($$TurnoutData{$Turnout}{'Pid'} != 0) {
863         &DisplayMessage("$Caller, turnout $Turnout position change is inprogress.");
864         &PlaySound("GE.wav");
865         return 1;
866     }
867     if ($$TrackData{$Turnout}{'Timeout'} > time) {
868         &DisplayMessage("$Caller, train transit of turnout $Turnout is inprogress.");
869         &PlaySound("GE.wav");
870         return 1;
871     }
872     if (&GetSensorBit($$TurnoutData{$Turnout}{'Sensor'}, $SensorBit,
873                     $SensorState) == 1) {
874         &DisplayMessage("$Caller, $Turnout sensor " .
875                         $$TurnoutData{$Turnout}{'Sensor'} . " is active.");
876         &PlaySound("GE.wav");
877         return 1;
878     }
879     return 0;
880 }
881
882 return 1;
883

```