ert_main.c

```
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * File: ert_main.c
 *
 * Code generated for Simulink model 'ACC_System'.
 *
 * Model version                  : 1.35
 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021
 * C/C++ source code generated on : Sat Apr  1 00:48:55 2023
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: Intel->x86-64 (Windows64)
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#include <stddef.h>
#include <stdio.h>            /* This example main program uses printf/fflush */
#include "ACC_System.h"                /* Model header file */

/*
 * Associating rt_OneStep with a real-time clock or interrupt service routine
 * is what makes the generated code "real-time".  The function rt_OneStep is
 * always associated with the base rate of the model.  Subrates are managed
 * by the base rate from inside the generated code.  Enabling/disabling
 * interrupts and floating point context switches are target specific.  This
 * example code indicates where these should take place relative to executing
 * the generated code step function.  Overrun behavior should be tailored to
 * your application needs.  This example simply sets an error status in the
 * real-time model and returns from rt_OneStep.
```

```
 */
void rt_OneStep(void);
void rt_OneStep(void)
{
  static boolean_T OverrunFlag = false;

  /* Disable interrupts here */

  /* Check for overrun */
  if (OverrunFlag) {
    rtmSetErrorStatus(ACC_System_M, "Overrun");
    return;
  }

  OverrunFlag = true;

  /* Save FPU context here (if necessary) */
  /* Re-enable timer or interrupt here */
  /* Set model inputs here */

  /* Step the model */
  ACC_System_step();

  /* Get model outputs here */

  /* Indicate task complete */
  OverrunFlag = false;

  /* Disable interrupts here */
  /* Restore FPU context here (if necessary) */
  /* Enable interrupts here */
}

/*
 * The example main function illustrates what is required by your
```

```
 * application code to initialize, execute, and terminate the generated code.
 * Attaching rt_OneStep to a real-time clock is target specific. This example
 * illustrates how you do this relative to initializing the model.
 */
int_T main(int_T argc, const char *argv[])
{
  /* Unused arguments */
  (void)(argc);
  (void)(argv);

  /* Initialize model */
  ACC_System_initialize();

  /* Attach rt_OneStep to a timer or interrupt service routine with
   * period 0.01 seconds (base rate of the model) here.
   * The call syntax for rt_OneStep is
   *
   *  rt_OneStep();
   */
  printf("Warning: The simulation will run forever. "
         "Generated ERT main won't simulate model step behavior. "
         "To change this behavior select the 'MAT-file logging' option.\n");
  fflush((NULL));
  while (rtmGetErrorStatus(ACC_System_M) == (NULL)) {
    /*  Perform application tasks here */
  }

  /* Terminate model */
  ACC_System_terminate();
  return 0;
}

/*
 * File trailer for generated code.
 *
```

```
 * [EOF]
 */
```

ACC_System.c

```
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * File: ACC_System.c
 *
 * Code generated for Simulink model 'ACC_System'.
 *
 * Model version                  : 1.35
 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021
 * C/C++ source code generated on : Sat Apr  1 00:48:55 2023
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: Intel->x86-64 (Windows64)
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#include "ACC_System.h"
#include "rtwtypes.h"
#include "ACC_System_private.h"
#include <math.h>
#include "ACC_control_output.h"

/* Named constants for Chart: '<S1>/Control Algorithm' */
#define ACC_IN_LeadVehicle_Not_Detected ((uint8_T)3U)
#define ACC_System_IN_ACC_OFF_Mode     ((uint8_T)1U)
#define ACC_System_IN_ACC_ON_Mode      ((uint8_T)2U)
```

```c
#define ACC_System_IN_ACC_STANDBY_Mode ((uint8_T)3U)
#define ACC_System_IN_NO_ACTIVE_CHILD  ((uint8_T)0U)
#define IN_LeadVehicle_Detected_Follow ((uint8_T)1U)
#define IN_LeadVehicle_Detected_Resume ((uint8_T)2U)
#define IN_LeadVehicle_Not_Detected_Res ((uint8_T)4U)
#define IN_LeadVehicle_Speed_equal_Set_ ((uint8_T)5U)
#define IN_LeadVehicle_Speed_lessthan_S ((uint8_T)6U)

/* Block states (default storage) */
DW_ACC_System_T ACC_System_DW;

/* External outputs (root outports fed by signals with default storage) */
ExtY_ACC_System_T ACC_System_Y;

/* Real-time model */
static RT_MODEL_ACC_System_T ACC_System_M_;
RT_MODEL_ACC_System_T *const ACC_System_M = &ACC_System_M_;
real_T rt_roundd_snf(real_T u)
{
  real_T y;
  if (fabs(u) < 4.503599627370496E+15) {
    if (u >= 0.5) {
      y = floor(u + 0.5);
    } else if (u > -0.5) {
      y = u * 0.0;
    } else {
      y = ceil(u - 0.5);
    }
  } else {
    y = u;
  }

  return y;
}
```

```
/* Model step function */
void ACC_System_step(void)
{
  /* Sum: '<S3>/Add' incorporates:
   *  Inport: '<Root>/CameraInput_LeadVehicle'
   *  Inport: '<Root>/RadarInput_LeadVehicle'
   */
  LeadVehicle_Speed = (uint8_T)(CameraInput_LeadVehicle + RadarInput_LeadVehicle);

  /* UnitDelay: '<S1>/Unit Delay' */
  Acceleration_Mode = ACC_System_Y.Acceleration_Mode_h;

  /* Sum: '<S4>/Add' incorporates:
   *  Inport: '<Root>/CameraInput_DriveVehicle'
   *  Inport: '<Root>/RadarInput_DriveVehicle'
   */
  DriveVehicle_Speed = (uint8_T)((uint8_T)(CameraInput_DriveVehicle +
    RadarInput_DriveVehicle) + Acceleration_Mode);

  /* SignalConversion: '<S4>/Signal Conversion' incorporates:
   *  Inport: '<Root>/RadarInput_DriveVehicle'
   */
  LeadVehicle_Detected = RadarInput_DriveVehicle;

  /* Chart: '<S1>/Control Algorithm' incorporates:
   *  Inport: '<Root>/CruiseSwitch'
   *  Inport: '<Root>/SetSwitch'
   *  Inport: '<Root>/Set_Gap'
   *  Inport: '<Root>/Set_Speed'
   *  Inport: '<Root>/Time_Gap'
   *  UnitDelay: '<S1>/Unit Delay'
   */
  if (ACC_System_DW.is_active_c3_ACC_System == 0U) {
    ACC_System_DW.is_active_c3_ACC_System = 1U;
    ACC_System_DW.is_c3_ACC_System = ACC_System_IN_ACC_OFF_Mode;
```

```
  ACC_System_Y.Acceleration_Mode_h = 0U;
} else {
  switch (ACC_System_DW.is_c3_ACC_System) {
   case ACC_System_IN_ACC_OFF_Mode:
    ACC_System_Y.Acceleration_Mode_h = 0U;
    if (CruiseSwitch) {
      ACC_System_DW.is_c3_ACC_System = ACC_System_IN_ACC_STANDBY_Mode;
      ACC_System_Y.Acceleration_Mode_h = 1U;
    }
    break;

   case ACC_System_IN_ACC_ON_Mode:
    {
      if (!SetSwitch) {
        ACC_System_DW.is_ACC_ON_Mode = ACC_System_IN_NO_ACTIVE_CHILD;
        ACC_System_DW.is_c3_ACC_System = ACC_System_IN_ACC_STANDBY_Mode;
        ACC_System_Y.Acceleration_Mode_h = 1U;
      } else if (!CruiseSwitch) {
        ACC_System_DW.is_ACC_ON_Mode = ACC_System_IN_NO_ACTIVE_CHILD;
        ACC_System_DW.is_c3_ACC_System = ACC_System_IN_ACC_OFF_Mode;
        ACC_System_Y.Acceleration_Mode_h = 0U;
      } else {
        switch (ACC_System_DW.is_ACC_ON_Mode) {
         case IN_LeadVehicle_Detected_Follow:
          ACC_System_Y.Acceleration_Mode_h = 2U;
          if (LeadVehicle_Detected == 0) {
            ACC_System_DW.is_ACC_ON_Mode = ACC_IN_LeadVehicle_Not_Detected;
            ACC_System_Y.Acceleration_Mode_h = 1U;
          } else if (((LeadVehicle_Detected == 1) && (LeadVehicle_Speed <
                      Set_Speed)) || (Time_Gap < Set_Gap)) {
            ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Speed_lessthan_S;
            ACC_System_Y.Acceleration_Mode_h = 4U;
          }
          break;
```

```
case IN_LeadVehicle_Detected_Resume:
 ACC_System_Y.Acceleration_Mode_h = 3U;
 if (LeadVehicle_Detected == 0) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Not_Detected_Res;
   ACC_System_Y.Acceleration_Mode_h = 1U;
 } else if ((DriveVehicle_Speed < Set_Speed) && (LeadVehicle_Speed >
             DriveVehicle_Speed) && (Time_Gap >= Set_Gap)) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Speed_equal_Set_;
   ACC_System_Y.Acceleration_Mode_h = 5U;
 } else if ((DriveVehicle_Speed == Set_Speed) && (LeadVehicle_Speed >=
             Set_Speed) && (Time_Gap >= Set_Gap)) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Detected_Follow;
   ACC_System_Y.Acceleration_Mode_h = 2U;
 }
 break;

case ACC_IN_LeadVehicle_Not_Detected:
 ACC_System_Y.Acceleration_Mode_h = 1U;
 if ((LeadVehicle_Detected == 1) && (DriveVehicle_Speed == Set_Speed)
     && (LeadVehicle_Speed >= Set_Speed) && (Time_Gap >= Set_Gap)) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Detected_Follow;
   ACC_System_Y.Acceleration_Mode_h = 2U;
 } else if (((LeadVehicle_Detected == 1) && (LeadVehicle_Speed <
              Set_Speed)) || (Time_Gap < Set_Gap)) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Speed_lessthan_S;
   ACC_System_Y.Acceleration_Mode_h = 4U;
 }
 break;

case IN_LeadVehicle_Not_Detected_Res:
 ACC_System_Y.Acceleration_Mode_h = 1U;
 break;

case IN_LeadVehicle_Speed_equal_Set_:
 ACC_System_Y.Acceleration_Mode_h = 5U;
```

```
 if (((LeadVehicle_Speed < Set_Speed) && (LeadVehicle_Speed <
       DriveVehicle_Speed)) || ((int32_T)rt_roundd_snf(0.75 * (real_T)
       Set_Gap) == Time_Gap)) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Speed_lessthan_S;
   ACC_System_Y.Acceleration_Mode_h = 4U;
 } else if (((DriveVehicle_Speed < Set_Speed) && (LeadVehicle_Speed >
   DriveVehicle_Speed)) || (Time_Gap >= Set_Gap)) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Detected_Resume;
   ACC_System_Y.Acceleration_Mode_h = 3U;
 } else if ((LeadVehicle_Detected == 0) || (DriveVehicle_Speed <=
             Set_Speed)) {
   ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Not_Detected_Res;
   ACC_System_Y.Acceleration_Mode_h = 1U;
 }
 break;

default:
 {
   int32_T tmp;
   int32_T tmp_0;
   uint8_T tmp_1;
   uint8_T tmp_2;

   /* case IN_LeadVehicle_Speed_lessthan_Set_Speed: */
   ACC_System_Y.Acceleration_Mode_h = 4U;
   tmp = (int32_T)rt_roundd_snf((real_T)LeadVehicle_Speed * 1.25);
   tmp_0 = (int32_T)rt_roundd_snf(1.25 * (real_T)Set_Gap);
   if (tmp < 256) {
     tmp_1 = (uint8_T)tmp;
   } else {
     tmp_1 = MAX_uint8_T;
   }

   if (tmp_0 < 256) {
     tmp_2 = (uint8_T)tmp_0;
```

```
          } else {
            tmp_2 = MAX_uint8_T;
          }

          if ((tmp_1 >= DriveVehicle_Speed) && ((int32_T)rt_roundd_snf
               ((real_T)LeadVehicle_Speed * 0.75) <= DriveVehicle_Speed) &&
              (DriveVehicle_Speed < Set_Speed) && (Time_Gap <= tmp_2) &&
              (Time_Gap >= (int32_T)rt_roundd_snf(0.75 * (real_T)Set_Gap)))
          {
            ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Speed_equal_Set_;
            ACC_System_Y.Acceleration_Mode_h = 5U;
          } else if ((LeadVehicle_Detected == 0) && (DriveVehicle_Speed ==
                     Set_Speed)) {
            ACC_System_DW.is_ACC_ON_Mode = ACC_IN_LeadVehicle_Not_Detected;
            ACC_System_Y.Acceleration_Mode_h = 1U;
          }
        }
        break;
      }
    }
  }
  break;

default:
  /* case IN_ACC_STANDBY_Mode: */
  ACC_System_Y.Acceleration_Mode_h = 1U;
  if (!CruiseSwitch) {
    ACC_System_DW.is_c3_ACC_System = ACC_System_IN_ACC_OFF_Mode;
    ACC_System_Y.Acceleration_Mode_h = 0U;
  } else if (SetSwitch) {
    ACC_System_DW.is_c3_ACC_System = ACC_System_IN_ACC_ON_Mode;
    ACC_System_DW.is_ACC_ON_Mode = IN_LeadVehicle_Detected_Follow;
    ACC_System_Y.Acceleration_Mode_h = 2U;
  }
  break;
```

```
    }
  }

  /* End of Chart: '<S1>/Control Algorithm' */
}

/* Model initialize function */
void ACC_System_initialize(void)
{
  /* (no initialization code required) */
}

/* Model terminate function */
void ACC_System_terminate(void)
{
  /* (no terminate code required) */
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

ACC_System.h

```
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * File: ACC_System.h
 *
 * Code generated for Simulink model 'ACC_System'.
 *
 * Model version                  : 1.35
 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021
 * C/C++ source code generated on : Sat Apr  1 00:48:55 2023
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: Intel->x86-64 (Windows64)
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#ifndef RTW_HEADER_ACC_System_h_
#define RTW_HEADER_ACC_System_h_
#ifndef ACC_System_COMMON_INCLUDES_
#define ACC_System_COMMON_INCLUDES_
#include "rtwtypes.h"
#endif                                 /* ACC_System_COMMON_INCLUDES_ */
```

```c
#include "ACC_System_types.h"

/* Includes for objects with custom storage classes */
#include "ACC_control_output.h"

/* Macros for accessing real-time model data structure */
#ifndef rtmGetErrorStatus
#define rtmGetErrorStatus(rtm)         ((rtm)->errorStatus)
#endif

#ifndef rtmSetErrorStatus
#define rtmSetErrorStatus(rtm, val)    ((rtm)->errorStatus = (val))
#endif

/* Block states (default storage) for system '<Root>' */
typedef struct {
  uint8_T is_active_c3_ACC_System;     /* '<S1>/Control Algorithm' */
  uint8_T is_c3_ACC_System;            /* '<S1>/Control Algorithm' */
  uint8_T is_ACC_ON_Mode;              /* '<S1>/Control Algorithm' */
} DW_ACC_System_T;

/* External outputs (root outports fed by signals with default storage) */
typedef struct {
  uint8_T Acceleration_Mode_h;         /* '<Root>/Acceleration_Mode' */
} ExtY_ACC_System_T;

/* Real-time Model Data Structure */
struct tag_RTM_ACC_System_T {
  const char_T * volatile errorStatus;
};

/* Block states (default storage) */
extern DW_ACC_System_T ACC_System_DW;

/* External outputs (root outports fed by signals with default storage) */
```

```
extern ExtY_ACC_System_T ACC_System_Y;

/* Model entry point functions */
extern void ACC_System_initialize(void);
extern void ACC_System_step(void);
extern void ACC_System_terminate(void);

/* Real-time Model object */
extern RT_MODEL_ACC_System_T *const ACC_System_M;

/*-
 * These blocks were eliminated from the model due to optimizations:
 *
 * Block '<Root>/Scope' : Unused code path elimination
 */

/*-
 * The generated code includes comments that allow you to trace directly
 * back to the appropriate location in the model.  The basic format
 * is <system>/block_name, where system is the system number (uniquely
 * assigned by Simulink) and block_name is the name of the block.
 *
 * Use the MATLAB hilite_system command to trace the generated code back
 * to the model.  For example,
 *
 * hilite_system('<S3>')    - opens system 3
 * hilite_system('<S3>/Kp') - opens and selects block Kp which resides in S3
 *
 * Here is the system hierarchy for this model
 *
 * '<Root>' : 'ACC_System'
 * '<S1>'   : 'ACC_System/Subsystem'
 * '<S2>'   : 'ACC_System/Subsystem/Control Algorithm'
 * '<S3>'   : 'ACC_System/Subsystem/Subsystem'
 * '<S4>'   : 'ACC_System/Subsystem/Subsystem1'
```

```
 */
#endif                                 /* RTW_HEADER_ACC_System_h_ */


/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

Users/jamesbond/Desktop/ACC_System_ert_rtw/ACC_System.h/
Ln25Col39

has popup