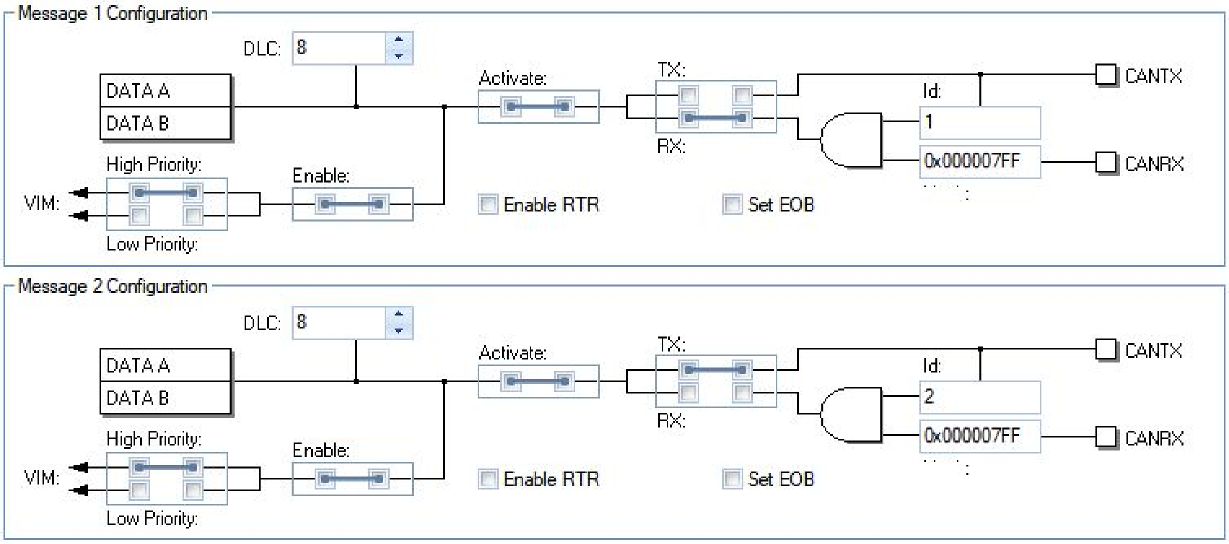
CSP in project

- Commented out csp_autoconfig.h’s
    - Created csp_autoconfig.h file, since I dont want to use waf and it wants to use gcc and CCS only works on windows
- Changed for(int i = 0;;) loops to pre-declare variable. CCS only supports C89, and this syntax is C99 standard

Feb 10, 2020

HALCoGen Settings

- For basic testing right now, I’ve had the server and client running different HALCoGen configs.
- Server:



Message 1 Configuration

DLC: 8

Activate: ☑

TX: ☐ RX: ☑

Id: 1

0x000007FF

Enable: ☑

High Priority: ☑ Low Priority: ☐

☐ Enable RTR

☐ Set EOB

Message 2 Configuration

DLC: 8

Activate: ☑

TX: ☑ RX: ☐

Id: 2

0x000007FF

Enable: ☑

High Priority: ☑ Low Priority: ☐

☐ Enable RTR

☐ Set EOB

- Client has message box 1 as tx and 2 as rx

In halcogen_can

- Rx thread: using uint8_t rx_data[8] = {0}; instead of uint8_t * rx_data = (uint8_t *)pvPortMalloc(8*sizeof(uint8_t)); Because malloc call is failing for an unknown reason

In csp_if_can.c

- csp_can_rx_frame(): changed csp_queue_enqueue_isr() to csp_queue_enqueue()
    - 3rd arg = (TickType_t)100

Where I am sitting currently

- TODO: have a switch statement that selects a message box based on the dlc argument so that we don’t get wasted bits
- **Problem**: Message boxes are defined by their ID. A message goes into message box 5 if it has the Id “5”. You can change this ID, but it doesn’t change the fact that there's only 64 boxes. Trying to interpret 27.8.6 of the technical reference manual correctly, I can’t figure out what the handler does if the ID field *doesn’t* match one of the 64 message box ID’s set in advance. I think it ignores the message?

  CSP fragments packets, and in doing this, it uses the CAN ID field:

  ```
  /* CAN frames contains at most 8 bytes of data, so in order to transmit CSP
   * packets larger than this, a fragmentation protocol is required. The CAN
   * Fragmentation Protocol (CFP) header is designed to match the 29 bit CAN
   * identifier.
   *
   * The CAN identifier is divided in these fields:
   * src:          5 bits
   * dst:          5 bits
   * type:         1 bit
   * remain:       8 bits
   * identifier:   10 bits
   *
   * Source and Destination addresses must match the CSP packet. The type field
   * is used to distinguish the first and subsequent frames in a fragmented CSP
   * packet. Type is BEGIN (0) for the first fragment and MORE (1) for all other
   * fragments. Remain indicates number of remaining fragments, and must be
   * decremented by one for each fragment sent. The identifier field serves the
   * same purpose as in the Internet Protocol, and should be an auto incrementing
   * integer to uniquely separate sessions.
  ```

  If the CAN module “ignores” messages that don’t have one of the 64 ID’s, how can we get this to work? I’m a bit stumped. I hope TI’s hardware implementation of CAN allows us to use CAN the way CSP wants it to be used.