Overview

This presents some ideas on the dynamometer program, based on STM32CubeMX, FreeRTOS, and interface and support routines developed in the ‘mxusartusbcan’ project (see ‘Brief.docx’ in ‘msusartusbcan’ directory for more detail).

A. Infrastructure Tasks

The following outlines how these features are used for the dynamometer project.

1. CanTxTask

Passes CAN msgs other tasks are sending to the interface routine (‘iface.c’) to the HAL CAN routine.

The CAN msg carries the HAL CAN module handle, which sends the CAN msg to the CAN1 or CAN2 modules and hence the CAN buses.

2. CanRxTask

Receives incoming CAN msgs placed in a queue by the functions in the ‘can\_iface.c’ routine that handle the RX0, RX1 callbacks, plus the loopback of CAN msgs being sent if compiled with the option set. The queue is loaded by ‘can\_iface.c’ under CAN interrupt. CanRxTask removes the msgs from the queue at a FreeRTOS Task priority.

CanRxTask is in charge of distribution & notification of CAN msgs. In some cases a CAN msg goes to more than one task.

a. Gateway: send CAN msgs to PC

If the unit is being used as a gateway to the PC and logging done on the PC, then CAN msgs would be converted to ascii/hex (‘gateway\_CANtoPC’, see Task03 ‘mxusartusbcan’) in and the buffer queued for sending to the usart via SerialTaskSend. The gateway hardware uses ‘usart2’ @ 2E6 baud for gateway operations. ‘hub-server’, ‘socat’, ‘cangate’, and the java programs such as the ‘HC’ (Host Controller) all deal with this ascii/hex format.

Use of the usb CDC (virtual com port) for gateway purposes requires some additional work, as the buffering/interrupt/polling is somewhat different with the STM HAL routine for usb-cdc and the usart routines.

Net—All CAN msgs from the queue route to the gateway usart buffer/queue.

b. Motor control

Motor #1 and #2 CAN msgs are selected and placed in global CAN msg structs and the MotorTask notified.

The notification of the MotorTask requires some coordination between CanRxTask routine handling the distribution of CAN msgs. While the CAN msg “mailboxes,” i.e. global CAN msg structs, is available to all, the notification bits are specific to each task. One way to handle the notification bit assignments is for the task that needs to be notified to send a pointer to a list of notification bits versus CAN ids to the CanRxTask during initialization.

B. Project specific tasks

1. defaultTask

Created: STM32CubeMX. Priority: osPriorityIdle (-3)

It will be used for the lowest priority computing, such as formatting data for output monitoring on usart6 using ‘yprintf’ (wrapper with semaphores for ‘vsnprintf’), stack high water mark checking, etc.

2. ControlTask

Created: manually Priority: osPriorityNormal (0)

Handles Speed and Torque control using the readings of the two ADC readings. The ADC DMA callback does a 16 reading summation and a notification. The ControlTask can then do further filtering, calibration, etc., triggered by the notification.

The ControlTask sends the periodic “keepalive” notifications to the MotorControlTask. The MotorControlTask then shuts down the motors if the notifications are missed. Likewise, the MotorControlTask sending periodic commands to the DMOCs and the DMOCs shutdown if the commands are not received in time.

The ControlTask sends the polling CAN msg; 64/sec. The POD/CAN unit sends back two CAN msgs with the load-cell readings as calibrated floats.

Since the ControlTask is the source for polling cycle, it might also send out command msgs to the DMOCs to retrieve DMOC measurements of current, voltage, speed, torque, etc.

The ControlTask could be integrated with the MotorConrolTask, however the functions of this ControlTask will eventually be taken over by the Master Controller, so separation might help the transition to the MC.

3. MotorControlTask

Created: manually Priority: osPriorityNormal (0)

This tasks maintains the connection logic for each DMOC state, e.g. “ping” msgs from the DMOCs indicate the DMOC is operational but in a not connected state.

Notification from the ControlTask causes the MotorControlTask to send the speed & torque commands.