Overview

“Mailbox” thoughts & musings, mainly as an aid to clarifying how to abstract and organize mailboxes in the context of this general project effort.

The notion of adding an ADC based mailbox to the ready-to-debug CAN mailbox code, raised a number of issues. Filtering the raw ADC readings would be expected; and might the logic apply to CAN msgs as well? Calibration is already applied to incoming CAN msgs, but need to be applied to the ADC readings. The parsing of the CAN payload is quite different than for the ADC situation where computations might be required, e.g. conversion of the internal temperature for voltage reference compensation. Filtering is a knotty issue if one considers that more than one type of filtering might be needed for the same readings, i.e. “fast” and “slow” filtering.

Online definitions of mailbox in the context of RTOS says it is a means for passing “messages”, or data exchange. For our purposes, the notion has to do more with making the *values* of *measurements* available to one or more tasks than the *data* where the data may not be some sensor measurement.

A. What the using task might be interested in from a mailboxes--

1. Is this a new reading since the last time I checked?

2. Is this reading “stale” i.e. too old?

3. Is this reading valid, i.e. the status?

4. Being notified when it is updated.

5. The reading filtered either/both--

a. Fast

b. Slow

c. Raw

d. Custom

6. The reading calibrated--

a. Offset

b. Scale

c. For ADC: Temperature compensated

d. For ADC: Sensor voltage compensation

7. Paired readings--

E.g. two different readings in one CAN msg, e.g. shaft speed and shaft encoder counter.

B. Characteristics Our Mailbox (OMBX?) might have--

1. Holds the latest value/reading.

Updates are made without regard to whether the ombx was accessed, i.e. overrun is not an issue.

2. Specific to some reading--

The USPS mailbox receives many different types of messages, that have the common attribute of the same destination address. This concept doesn’t apply the OMBX. E.g. a single OMBX would not combine voltage of 12v supply, drum, #1 tensions, etc. (What about “paired” readings such as shaft speed and encoder count that arrive in one CAN msg? Two maibloxes? One that will handle a float and uint32\_t?)

3. Identification

A specific OMBX, or pointer to it, identifies the source of the value/reading. The source could be CAN msgs, or ADC readings, etc.

Implementing this raises the issue of mapping, i.e. a OMBX routing some incoming value/reading, e.g. a CAN msg payload, or element in an array receiving measurements from an ADC scan. These have to be converted from their native form into the common OMBX.

This issue requires some thought. Some typing to develop thinking--

To get a OMBX, e.g., 5v sensor supply voltage, a call to a routine to create an OMBX given the ADC (ADC1, 2, or 3) and ADC array index would be made and a pointer returned that points to the OMBX. The name used for that pointer would be the mnemonic for 5v sensor supply voltage. The call to the routine would include the ADC module handle, and array index. For CAN, a different create routine would be called, but the pointer returned would be the same type. The call would include the CAN control block pointer for the CAN module, CAN ID, and CAN payload type code. Again the naming of the pointer would be the mnemonic for that reading.

If more than one task tried to create the same OMBX the pointer returned would point to the same OMBX, but if notifications were specified in the call, the notification would be added to the list.

The routine creating the OBMX needs to maintain a list of OBMX pointers versus the reading source so that, e.g. a CAN msg after payload conversion can be routed to a mailbox. Similarly, the same for ADC. ADC is easy/fast because the readings are always by position in an array. CAN requires some means of a lookup on CAN module and CAN ID.

Since the number of ADC measurements can not exceed 18 (16 plus internal temp and voltage reference) an array of OMBX for all possible ADC mailboxes could be allocated, i.e. the overhead of a linked list might not be worth the effort. Where the advantage of the array is realized is in the loading of the mailboxes; the index is direct whereas with a linked list the ADC raw array would have to map into the linked list via table of pointers. The linked list would save a little sram, but add flash code.

3. Filtering

At issue is whether to build-in filtering to OMBX, or do it locally in the task. Some tasks might want slow, high filtering, e.g. readings displayed via uart/minicom for human observation, such as load-cell readings. The same readings might also be used where filtering would be fast, e.g. control loop. Should inclusion of filtering in the OMBX include provision for multiple filters.

If filters are included in OMBX, are multiple types needed? A simple one pole IIR filter is simple and easy. One could pass a reading through that filter multiple times for sharper cutoff. There has to be the intermediate storage for each pass ...

The basic OMBX structure could have a pointer to a linked list of these IIR filter structs.

If there is more than one filter, the user would have to make a call to a routine that has the OMBX pointer plus a filter pointer. The filter pointer would have to be included when the OMBX was created, so that would be passed back would be a struct.

Decimation--

If decimation is involved, then most likely notification would associated with it. However, if arrival of a reading is not driving some task, i.e. a notification is give, then task must be polling. If it is polling and checking the filtered reading, then decimation takes place if the reading update is faster than the polling, as well as a resynchronization. Net—including decimation doesn’t make sense at the mailbox level.

4. Calibration

With CAN the approach has been that calibration is done at the sensor node and the CAN msg payload is a calibrated float. With ADC, same approach could be done which requires a routine between the ADC raw readings and the OMBX. The array of ADC readings would be processed to produce an array of calibrated readings before being loaded into a mailbox.