**Some definition and clarification of terms.**

**CAN msg type**--

CAN msgs that belong to this group have a payload that is identical in structure and information conveyed. The CAN msgs in this group only different in the CANID. Only one CAN node is allowed to send a msg with a given CANID, the CANID identifies the source, and the source in this lexicon is a “function” that resides in a node. Multiple “instances” of the same function can exist, so each will require a different CAN id assignment, yet the payload and its interpretation remain identical.

CAN ids for msgs of the same CAN msg type may, or may not, have bit fields that designate the “instance” of the payload. When the bit fields are the lower order bits of the CAN id, these msg instances will have the same bus arbitration priority for bits preceding the bit field.

The file CANID\_INSERT.sql in the CAN database, located in the directory-- GliderWinchCommons/embed/svn\_common/trunk/db

has columns three columns that relate a given CANID assignment to the mgs type--

CANID\_TYPE

CANID\_INST

CANID\_INFUNC

Note: when the database was developed the term “function” was used differently than today (11/2020).

The following two in the database, define the “CAN msg type”--

CANID\_TYPE (chars) is what is now called “function” such as levelwind function. A function might send more than one CAN msg of different “CAN msg type”.

CANID\_INFUNC (integer) is the “CAN msg type” within a “function.” E.g. 1 might signify a tension polled msg response, and 2 might signify heartbeat msg, both belonging to a “tension function”.

CANID\_INST (integer) is the “instance” number of the “CAN msg type”, e.g. 2 would signify the tension msg was from drum #2, if the above two were “drum”, “1”.

**Command (in context of CAN msgs)**

A CAN msg that is sent with the expectation of one or more recipients will take some action and may or may send a response.

A command msg might poll functions whereby payload supplies information that specifies the type of response.

**Function**

The concept of having a set of routines that implement some winch “function” arranged so that work involved in relocating the routines to different CAN nodes, or even processors, is minimized. E.g. drum (which includes speed, odometer, estimated diameter, etc.). The can be more than one “instance” of a function, e.g. a drum function for each drum on the winch. A node can have multiple functions.

Functions generally are assigned multiple CAN msg types:

1) Heartbeat msg--

conveys status and status codes.

2) Command msgs--

Incoming command to the function

Response, if applicable to the command, from the function.

3) Poll response msgs

**Heartbeat msg**

A CAN msg that is periodic. It may be sent at regular time intervals, or triggered by changes that affect the data in the payload, in which case it may or may not require a minimum time before another msg will be sent.

What distinguishes a heartbeat msg from other types is that it will send msgs when there is no other CAN activity that might affect it, and no changes, such as sensor changes to trigger an immediate sending of the msg, and there is data in a payload (otherwise it would be a status msg).

The duration between heartbeat msgs, when not triggered, might be variable, based on other incoming CAN msgs.

**Instance**

Each implementation of a “function” is an instance. There can be multiple instances contained in one CAN node, or across CAN nodes. Each instance carries its own set of parameters, which includes CAN ids used by the instance for each CAN msg type used by the function.

**Keep-alive**

These are similar to heartbeat, and command, msgs, but fulfill a need for one or more recipients to confirm communication with the sender. If keep-alive msgs are not received within a specified duration some action is taken, e.g. shutting down the contactor.

**Node**

Node refers to CAN node which is a CAN bus interface with processor.

**Polling (in context of CAN msgs)**

A CAN msg that polls is one that is sent out with the expectation that one or more responses will be forthcoming.

**Polling (in context of program loops)**

A loop that repetitively checks for status or some condition. Polling is inherently interrupt driven, but not driven by changes in the status or condition that is being checked, e.g. a timer or spi interrupt causes the program code to check the conditions of switches. A switch change that causes an interrupt would not be polling.

**Status msgs**

These msgs are the same as heartbeat msgs except they are generally considered high priority.