The goal of this document is to map out possible avenues to have a common flight mode selection / implementation

Current issue: The flight mode selection for a flight stack is very implementation specific, making it hard for a ground station to support unknown vehicles without a hard code change. This in turn limits the perceived usefulness of the standard, as a core feature requires instant customization in code. As flight modes have settled mostly in recent years across the industry, there are a lot of common modes that could be shared:

Common Flight Modes in the industry:

- ACRO
- MANUAL (fixed wing only in APM, maps to stabilized in PX4)
- ATTITUDE (called stabilized in both APM and PX4)
- ALTHOLD
- ACCELERATION / POSITION / GUIDED (core)
- LOITER (manual inputs)
- RTL (variants)
- TAKEOFF
- LAND
- MISSION/AUTO (core)
- EXTERNAL

Extension / Task Modes

- TASK MODES
    - ORBIT
    - CABLECAM
    - FLIPS
    - FOLLOW ME
    - RETURN
    - SMART RETURN
    - LAND
    - TAKEOFF
    - GOTO
    - THROW

Existing MAVLink Concept

- Usage of either common modes (e.g. ATTITUDE, GUIDED) or of custom mode field
- Common mode allows for a minimal baseline implementation, but doesn't solve very common needs like e.g. a follow-me mode

New Concept:

- There is a single, clear way how to interpret the flight mode message (custom modes only in the form of an enum)
- Every vehicle supports a small number of “core” modes (required by spec, e.g. ATTITUDE, GUIDED, AUTO). Supporting can also mean to reject it, but the concept needs to be supported and correctly reported.
- New mode indication message includes current mode and a human-readable name as 20-character string
- The vehicle can be requested to be put into a specific mode with flags for inputs (e.g. X-Y attitude, Z-position) and will respond if it supports that or not. This allows to set the right behaviour without necessarily having to understand the exact underlying mode concept.
- Custom modes are mapped to an ENUM
- Groundstation can query vehicle for an XML (or JSON?) file that details all modes.

Message Structure:

-

With this, the following scenarios can be achieved:

- The ground station can offer baseline flight control (manual, mission, guided) for any autopilot.
- Ground station can enable any custom mode that can be specified through definition file

Other Ideas

- Mode metadata (requirements like manual input, valid position, etc)
    - We should be able to show if a mode is available or not before trying to switch to it
    - QGC should show or hide sticks based on the mode taking manual input or not