XB-70 Prototype

🔧 Scope of This Response

You asked (now with *both* ZeroDrag **Nova F4** *and* **Warp F7**) for clear answers to:

1. **How to flash custom firmware on the ZeroDrag Nova F4 flight controller?**
2. **If we can flash custom firmware, which software/tools to use?**
3. **How to integrate the u-blox NEO-7M GPS + board with our Mecha Comet (i.MX 8M Nano) platform?**
4. **Which driver stack to use (HAL, CMSIS, Bare-metal, etc.)—and does this change if we move to Warp F7?**

I also give you a **decision layer**: whether to stay with Nova F4 or adopt Warp F7 for pre-processing before data reaches the Mecha Comet.

0. Hardware Context Recap

Item	Role in Prototype
u-blox NEO-7M module	GNSS position / groundspeed / time (NMEA & UBX)
ZeroDrag Nova F4	Low-cost MCU platform (IMU+baro) for quick AHRS
ZeroDrag Warp F7	Higher-performance alternative (faster MCU, more buses)
Mecha Comet (i.MX 8M Nano)	Main application processor: EFIS GUI (Qt/QML), fusion display, future lite SVS
XB-70 GUI	Consumes fused attitude, altitude, GPS, system status

1. FLASHING CUSTOM FIRMWARE – NOVA F4

1.1 Boot & Flash Paths

Path	Purpose	When to Use
Betaflight / INAV Configurator	Flash ready-made flight FW (MSP output)	Fastest AHRS feed
DFU (ROM bootloader) + dfu-util	Flash raw `.bin` / `.hex`	Scripted or custom images
ST-LINK (SWD)	Full debug (breakpoints, memory view)	Custom HAL / FreeRTOS
OpenOCD + GDB	Cross-platform debug	Headless Linux setup

1.2 Entering DFU Mode (Typical)

1. Hold/short BOOT0 pad (or “BOOT” button) → press RESET → release RESET (keep BOOT if required).
2. Board enumerates as *STM32 DFU* over USB.
3. Flash:

```
dfu-util -l
dfu-util -a 0 -s 0x08000000:leave -D firmware.bin
```

1.3 Flash via Betaflight / INAV Configurator

1. Connect USB → open Configurator → “Firmware Flasher.”
2. Select target (generic **F405** family).
3. Load official or custom build (for INAV if you want better navigation defaults).
4. Flash & verify → Reboot → Enable MSP or desired telemetry on a UART.

1.4 Flash Custom (HAL/FreeRTOS) Build

Toolchain Setup (on Jupiter / Ubuntu):

```
sudo apt install gcc-arm-none-eabi gdb-multiarch openocd
```

Build Skeleton (CMake or CubeIDE).

Flash (OpenOCD example):

```
openocd -f interface/stlink.cfg -f target/stm32f4x.cfg -c "program build/airman_f405.elf verify reset exit"
```

2. CAN WE FLASH CUSTOM FW? WHICH SOFTWARE?

Task	Recommended Toolchain
Quick “get attitude now”	Betaflight or INAV + Configurator
Logging MSP → i.MX	Same + Python MSP parser
First custom telemetry protocol	STM32CubeMX + HAL + FreeRTOS + ST-LINK
Performance tuning (SPI DMA IMU)	Add LL (Low Layer) APIs inside HAL project
Fully minimal build	CMSIS + Bare metal (later, if code size / determinism matters)

Yes, you *can* fully replace the stock firmware. Nothing proprietary locks you out—standard STM32F405.

3. INTEGRATION OF u-blox NEO-7M + CONTROLLER + MECHA COMET

3.1 Wiring (Recommended Initial Topology)

```
u-blox NEO-7M  ─── UART0/USB → (Direct) Mecha Comet / i.MX 8M Nano
Nova F4        ─── USB (MSP) → Mecha Comet
(optionally) Nova F4 UART → i.MX (if USB not desired)
```

Why direct GPS → i.MX first?

Simplifies parsing (raw NMEA/UBX) and decouples GPS timing from IMU loop.

3.2 Alternative (Single Serial Consolidation)

```
u-blox NEO-7M → Nova F4 (UART2)
```

```
Nova F4      → i.MX (USB MSP packets containing attitude + GPS)
```

Pros: One cable to host.

Cons: You must add GPS parsing inside the F4 firmware or rely on INAV's GPS messages.

3.3 u-blox Optimization Steps

1. Connect module to PC (USB-TTL) & open **u-center** (Windows) or use UBX config packets from Linux.
2. Disable unused NMEA sentences (keep **GGA, RMC, VTG** optional).
3. Set update rate to **5 Hz** (or 10 Hz if bandwidth acceptable).
4. Save config to flash: UBX-CFG-CFG (save mask: I/O + Messages + Rates).

3.4 Data Fusion Flow (Prototype)

Source	Rate	Transport	Consumer
IMU (gyro/accel)	200–400 Hz	MSP / custom binary	Fusion daemon (i.MX)
Baro	25–50 Hz	MSP / custom	Fusion daemon
GPS	5–10 Hz	Raw NMEA (tty)	GPS parser → fusion
Output (att+pos)	20–30 Hz	Shared memory / JSON	Qt/QML EFIS GUI

3.5 Example MSP → JSON Daemon (Concept)

- Thread 1: Read `/dev/ttyACM0` (MSP frames) → decode attitude & baro → ringbuffer.
- Thread 2: Read `/dev/ttyUSB1` (GPS NMEA) → parse lat/lon/gs/alt.
- Thread 3: Fuse (simple complementary / Madgwick) if needed, else pass-through.
- Publish at 25 Hz to `/tmp/airman-data.json` (inotify or Unix socket for GUI).

---

4. DRIVER STACK CHOICE (Nova F4 vs Warp F7)

4.1 Hardware Differences Impacting Stack

Feature	Nova F4 (STM32F405)	Warp F7 (STM32F7xx – likely 407 → 722/745 class)*	Impact
Core	Cortex-M4F @168 MHz	Cortex-M7 @ up to 216–400 MHz (spec depends)	More headroom: run fusion + preprocessing
Flash / RAM	1 MB / 192 kB	Larger (e.g. 512 kB–1 MB RAM w/ DTCM)	Bigger buffers (GPS, logging)
FPU	Single-precision	Single-precision w/ faster pipeline	Higher fusion loop rate margin
Cache	None	I/D cache (M7)	Lower memory latency
SPI / I2C	Standard	Similar but faster bus ceilings	Higher IMU poll frequency
SDIO (possible)	Maybe absent	Likely available	Local high-rate logging
Power Draw	Lower	Slightly higher	Consider if battery powered

- Exact Warp F7 variant to confirm from vendor, but typical F7 flight controllers follow this profile.

4.2 Stack Recommendation by Phase

Phase	Board	Stack
Rapid bring-up	Nova F4	*Use INAV/Betaflight* (MSP)
Structured prototype	Nova F4	HAL + FreeRTOS (tasks: IMU, GPS pass-through, telemetry)
Higher performance / future pre-processing (e.g., adaptive filtering, air data blending)	Warp F7	HAL + FreeRTOS + selected LL for SPI DMA
Long-term optimized	Warp F7	HAL + custom math libs (CMSIS-DSP), potential migration of partial EKF from i.MX

4.3 HAL vs CMSIS vs Bare Metal

Option	Use Case	Notes
HAL Only	Fast development	Good abstraction; ok latency for ≤400 Hz loops
HAL + LL	Time-sensitive SPI / UART DMA	Drop into LL for IMU read & timestamping
CMSIS-DSP Library	Quaternion fusion / filters	Use on F4/F7 for efficient vector math
Bare Metal (Registers)	Extreme timing determinism	Only later if profiling shows >20% loop time overhead
FreeRTOS	Multiple concurrent tasks (sensor, GPS, telemetry, watchdog)	Keep fusion priority high (config preemption)

5. DETAILED ANSWERS (CONDENSED FORM)

Question	Answer (Nova F4)	Answer (Warp F7 Delta)
How to flash custom firmware?	DFU (BOOT0 + reset) → `dfu-util` OR ST-LINK (SWD) OR Betaflight/INAV Configurator.	Same methods; F7 targets show as DFU too; faster flash + more space.
Which software to flash?	Betaflight/INAV Configurator (fastest), STM32CubeProgrammer, OpenOCD+GDB, dfu-util.	Identical set; plus advantage of faster debug sessions (cache).
How to integrate hardware (GPS + board + Mecha Comet)?	GPS → i.MX (raw NMEA) + F4 → i.MX (attitude via MSP/custom). Optionally route GPS through F4 later.	Same, but F7 can pre-process (e.g., integrate GPS + IMU, output fused state vector).
Which driver code base?	Start INAV → migrate to HAL+FreeRTOS+CMSIS-DSP for custom telemetry.	Go straight to HAL+FreeRTOS if aiming for advanced pre-processing; use CMSIS-DSP early.

6. WHEN TO UPGRADE TO WARP F7

Upgrade **if/when** you need:

- Higher rate fusion (≥800 Hz raw gyro sampling, 200 Hz output).

- On-board pre-filtering (e.g., Allan variance gyro bias modeling).
- Local logging to SD at high rate (flight test dataset capture).
- Future modules: on-board air-data computer, local ADS-B pre-parser.

If current objective is *just to unblock GUI + route attitude/GPS to EFIS*, **Nova F4 remains sufficient**.

7. SAMPLE TELEMETRY PACKET (Custom Binary)

(For when you move off MSP)

Byte(s)	Field	Type	Notes
0–1	Sync `0xAA55`	u16	Frame marker
2	Version	u8	Protocol version
3	Payload Len	u8	N bytes following
4–7	Timestamp (µs)	u32	From MCU cycle counter
8–13	Roll, Pitch, Yaw (0.01°)	3×s16	Degrees ×100
14–19	Gyro X/Y/Z (mdps)	3×s16	Raw or scaled
20–25	Accel X/Y/Z (mg)	3×s16	
26–29	Pressure (Pa / 10)	u32	
30–33	Alt (cm)	s32	Fused baro/GPS
34–37	GPS Lat (1e-7 deg)	s32	
38–41	GPS Lon (1e-7 deg)	s32	
42–43	Ground Speed (cm/s)	u16	
44–45	Sat count	u8 + pad	
46	Flags	u8	Bitmask
47	CRC8	u8	Polynomial x^8+x^2+x+1

8. INITIAL TASK CHECKLIST

Day	Task	Owner
1	Flash INAV / Betaflight on Nova F4 & stream MSP attitude	FW
1–2	Wire NEO-7M → i.MX & log NMEA	FW
2	Build Python daemon: MSP + NMEA → JSON	FW
3	Feed EFIS GUI (Qt) with JSON (roll/pitch/heading/GS)	GUI
4–5	Profile latency & jitter (target <50 ms end-to-end)	FW
5–7	Start HAL project skeleton (F4); SPI IMU read via DMA	FW
8+	Decide if Warp F7 required for added processing	CTO/FW

9. COMMON PITFALLS & MITIGATIONS

Problem	Symptom	Mitigation
USB resets mid-flight test	MSP stream pauses	Use UART hardline instead of USB initially
Gyro drift high	Slow “bank” drift when stationary	Ensure 2–3 s sit still after boot; implement temp compensation
GPS delay	Latency in EFIS groundspeed	Use PPS pin to timestamp (future); reduce NMEA sentence load
FIFO overruns (IMU)	Attitude jitter	Configure SPI DMA & read at fixed loop (timer interrupt)
EFIS choppiness	Attitude updates irregular	Publish fused attitude at fixed 25 Hz, decouple from raw sampling

10. RECOMMENDATION SUMMARY (EXEC LEVEL)

- **Short Term (Week 1–2):** Nova F4 + MSP + direct GPS → Mecha Comet. Fast path to visible, stable EFIS motion.
- **Mid Term (Week 3–5):** Shift to HAL+FreeRTOS on F4; introduce custom packet with timestamps.
- **Decision Gate:** If we start pushing pre-processing (e.g., advanced EKF, terrain-referenced corrections), **evaluate Warp F7** before freezing hardware

rev.

- **Long Term:** i.MX remains the platform for map/SVS; MCU handles clean, deterministic raw fusion & timing.

Want Next?

I can send you:

- **u-center configuration script** (UBX commands) for NEO-7M
- **CubeMX .ioc** template for F4 or F7
- **Python MSP → JSON production-ready daemon**

Just tell me which one to generate first, Mr. AIRMAN.

On standby for the next push. 🛫