ICM-42688-P IMU ve Register Programlama Rehberi

TDK InvenSense 6-Eksen IMU — Genesys2/Rocket SPI Uygulaması

Rocket/Genesys2 Projesi Teknik Dokümantasyonu

12 Temmuz 2026

- 1 Dokümanın amacı ve kapsamı
- 2 IMU nedir?
  - 2.1 İvmeölçer ne ölçer?
  - 2.2 Jiroskop ne ölçer?
  - 2.3 Gövde koordinat sistemi
- 3 IMU ölçüm kavramları
  - 3.1 Full-Scale Range (FSR)
  - 3.2 Hassasiyet ve LSB ölçeği
  - 3.3 ODR — Output Data Rate
  - 3.4 Bant genişliği ve aliasing
  - 3.5 Bias, gürültü ve sıcaklık etkisi
  - 3.6 Quantization ve 16 bit UI verisi
- 4 ICM-42688-P mimarisi
  - 4.1 Cihaz kimliği
- 5 Register bank kavramı
  - 5.1 Register erişim türleri
- 6 SPI register protokolü
  - 6.1 SPI mode konusu
- 7 Bank 0 register haritası — temel alanlar
- 8 Temel registerların bit düzeyinde açıklaması
  - 8.1 DEVICE_CONFIG — 0x11
  - 8.2 INT_CONFIG — 0x14
  - 8.3 INT_STATUS — 0x2D
  - 8.4 SIGNAL_PATH_RESET — 0x4B
  - 8.5 INTF_CONFIG0 — 0x4C
  - 8.6 PWR_MGMT0 — 0x4E
  - 8.7 GYRO_CONFIG0 — 0x4F
    - 8.7.1 GYRO_FS_SEL
    - 8.7.2 GYRO_ODR
  - 8.8 ACCEL_CONFIG0 — 0x50
    - 8.8.1 ACCEL_FS_SEL
    - 8.8.2 ACCEL_ODR
  - 8.9 FIFO_CONFIG1 — 0x5F
  - 8.10 WHO_AM_I — 0x75
- 9 Sensör verisi registerları
  - 9.1 Burst veri düzeni
  - 9.2 Sıcaklık dönüşümü
  - 9.3 İvme dönüşümü
  - 9.4 Jiroskop dönüşümü
- 10 FIFO kavramı ve kullanımı
  - 10.1 FIFO kullanmanın avantajları
  - 10.2 FIFO count
  - 10.3 FIFO watermark
  - 10.4 FIFO overflow ve kayıp packet
- 11 Interrupt sistemi
- 12 APEX ve hareket olayları
- 13 Self-test
- 14 Güvenilir başlatma sırası
- 15 Bu projedeki C sürücüsüyle eşleme
  - 15.1 FPGA SPI MMIO registerları
  - 15.2 Register okuma fonksiyonu
  - 15.3 Register yazma fonksiyonu
- 16 Sahada görülen sonuçların yorumlanması
  - 16.1 `WHO_AM_I: 47 47 47 47`
  - 16.2 Mode yazısından sonra `0x00`
  - 16.3 Geçerli ölçümler
  - 16.4 Daha sonra tüm kanallarda alt sınır
- 17 IMU'dan alınabilen bilgi türleri
  - 17.1 Önerilen yazılım örnek yapısı
- 18 Kalibrasyon ve doğrulama
  - 18.1 Kalibrasyondan önce hazırlanacak registerlar
  - 18.2 1. Jiroskop sıfır bias kalibrasyonu
    - 18.2.1 Koşullar
    - 18.2.2 Hesap
    - 18.2.3 Kalite kontrolü
  - 18.3 2. İvmeölçer altı-yüz kalibrasyonu
    - 18.3.1 Örnek
  - 18.4 3. Cross-axis ve misalignment kalibrasyonu
  - 18.5 4. Jiroskop scale-factor kalibrasyonu
  - 18.6 5. Sıcaklık kalibrasyonu
  - 18.7 6. Gürültü ve tekrarlanabilirlik ölçümü
  - 18.8 7. Allan variance
  - 18.9 8. Zaman tabanı kalibrasyonu
  - 18.10 9. Kalibrasyon verisini saklama
  - 18.11 10. Kalibrasyon sonrası kabul testleri
- 19 Hata ayıklama tablosu
- 20 Uygulama için önerilen sürücü katmanları
- 21 Kısa kontrol listesi
- 22 Kaynaklar

1 Dokümanın amacı ve kapsamı

Bu doküman, TDK InvenSense ICM-42688-P altı eksenli ataletsel ölçüm biriminin (IMU) ne ölçtüğünü, ölçümlerin hangi fiziksel ve sayısal hatalara açık olduğunu, cihazın register mimarisini ve registerların gömülü yazılımdan nasıl programlanacağını açıklar. Anlatım, bu depodaki Genesys2/Rocket uygulamasına bağlanmıştır; ancak IMU ve register bölümleri başka mikrodenetleyici veya FPGA tasarımlarında da kullanılabilecek şekilde hazırlanmıştır.

Sahadaki cihaz dört ardışık okumada `WHO_AM_I = 0x47` döndürmüştür. TDK veri sayfasına göre `0x47`, **ICM-42688-P** cihaz kimliğidir. ICM-42686-P kimliği `0x44` olduğundan bu dokümanın ana cihazı ICM-42688-P'dir. İki cihaz aynı register ailesini büyük ölçüde paylaşsa da tam ölçek aralıkları farklıdır; yazılım cihaz kimliğine göre ölçek seçmelidir.

Bu metin veri sayfasının yerine geçmez. Üretim tasarımında register bitlerinin, elektriksel sınırların ve errata bilgisinin kullanılan silikon revizyonuna ait en güncel TDK belgesiyle yeniden doğrulanması gerekir.

2 IMU nedir?

IMU, cismin doğrusal hareketi ve dönme hareketi hakkında bilgi üreten bir sensör grubudur. ICM-42688-P içinde üç eksenli ivmeölçer, üç eksenli jiroskop ve bir sıcaklık sensörü bulunur:

Alt sensör	Eksenler	Temel çıktı	Tipik birim
İvmeölçer	X, Y, Z	Özgül kuvvet	g veya m/s²
Jiroskop	X, Y, Z	Açısal hız	derece/s veya rad/s
Sıcaklık	—	Yonga sıcaklığına bağlı sayısal kod	°C

"Altı eksen" ifadesi üç ivme ve üç açısal hız kanalını anlatır. IMU doğrudan konum, hız veya Euler açısı ölçmez. Bu büyüklükler sensör çıktılarının zaman içinde işlenmesiyle kestirilir.

2.1 İvmeölçer ne ölçer?

MEMS ivmeölçer, küçük bir test kütlesinin gövdeye göre yer değiştirmesinden **özgül kuvveti** ölçer. Durgun bir sensör de yerçekimi nedeniyle yaklaşık 1 g büyüklüğünde çıktı verir. Örneğin kart yatay ve Z ekseni yukarı bakıyorsa ideal çıktı yaklaşık olarak:

```
ax ≈ 0 g
ay ≈ 0 g
az ≈ +1 g   (eksen işaretine göre -1 g de olabilir)
```

Projede gözlenen yaklaşık `A=-7,-12,1037 mg` sonucu bu davranışla uyumludur: kart hareketsizken Z ekseninde yaklaşık 1.037 g vardır.

İvmeölçer çıktısı yerçekimi ile doğrusal hareketi kendiliğinden ayıramaz. Dünya koordinatında doğrusal ivme elde etmek için önce yönelim kestirilmeli, sonra yerçekimi vektörü gövde koordinatındaki ölçümden çıkarılmalıdır.

2.2 Jiroskop ne ölçer?

MEMS jiroskop, Coriolis etkisini kullanarak eksenler etrafındaki açısal hızı ölçer. Durgun ideal sensörde üç eksen de sıfır olmalıdır; gerçek sensörde bias nedeniyle küçük sıfır dışı değerler görülür. Projedeki yüzlerce mdps düzeyindeki değerler, yaklaşık birkaç onda derece/saniye bias anlamına gelir.

Açı, açısal hızın integraliyle bulunur:

```
theta[k] = theta[k-1] + omega[k] * dt
```

Bias da bu denklemde sürekli integre edildiği için yalnız jiroskop kullanılarak bulunan açı zamanla sürüklenir. Bu nedenle ivmeölçer, jiroskop ve gerekirse manyetometre verileri complementary filter, Kalman filter veya quaternion tabanlı bir füzyon algoritmasıyla birleştirilir.

2.3 Gövde koordinat sistemi

Registerlarda görülen X/Y/Z, sensör paketinin kendi gövde koordinatıdır. Kart üzerindeki fiziksel montaj yönü ile aracın veya sistemin koordinatları farklı olabilir. Yazılımda bir eksen dönüşümü gerekir:

```
v_system = R_mount * v_sensor
```

Buradaki `R_mount`, yalnız eksen değiştirme ve işaret tersleme içeriyorsa 0, +1 ve -1 elemanlı bir matristir. Mekanik montaj açılıysa genel bir dönme matrisi veya quaternion kullanılır.

3 IMU ölçüm kavramları

3.1 Full-Scale Range (FSR)

FSR, ADC'nin doyuma ulaşmadan temsil edebildiği pozitif ve negatif ölçüm sınırıdır. ICM-42688-P ivmeölçer için ±2/4/8/16 g; jiroskop için ±15.625 ile ±2000 dps arasında seçenek sunar.

Dar FSR daha yüksek sayısal hassasiyet, geniş FSR ise daha büyük hareketleri doyuma girmeden ölçme avantajı verir. Uygulama beklenen en büyük hareketten biraz daha geniş aralık seçmelidir.

16 bit UI register çıkışında genel dönüşüm:

```
fiziksel_değer = raw_signed16 * FSR / 32768
```

Örnek: ±16 g ayarında `raw = 2048` ise:

```
accel_g  = 2048 * 16 / 32768 = 1 g
accel_mg = 1000 mg
```

3.2 Hassasiyet ve LSB ölçeği

ICM-42688-P 16 bit register çıkışı için ölçek tablosu:

İvme FSR	Hassasiyet
±2 g	16384 LSB/g
±4 g	8192 LSB/g
±8 g	4096 LSB/g
±16 g	2048 LSB/g

Jiroskop FSR	Hassasiyet
±15.625 dps	2097.2 LSB/(°/s)
±31.25 dps	1048.6 LSB/(°/s)
±62.5 dps	524.3 LSB/(°/s)

Jiroskop FSR	Hassasiyet
±125 dps	262 LSB/(°/s)
±250 dps	131 LSB/(°/s)
±500 dps	65.5 LSB/(°/s)
±1000 dps	32.8 LSB/(°/s)
±2000 dps	16.4 LSB/(°/s)

3.3 ODR — Output Data Rate

ODR sensörün saniyede kaç yeni örnek ürettiğini belirtir. SPI saat frekansı ile ODR aynı değildir. SPI, registerların ne kadar hızlı aktarılabildiğini; ODR ise MEMS/ADC/dijital filtre zincirinin ne kadar sık yeni ölçüm ürettiğini belirler.

Örneğin ODR 1 kHz iken registerı 10 kHz hızla okumak aynı örneği birden fazla kez döndürebilir. ODR 1 kHz iken yalnız 100 Hz okumak ise aradaki örnekleri UI register yolunda kaçırır; FIFO etkinse örnekler FIFO'da saklanabilir.

3.4 Bant genişliği ve aliasing

ODR tek başına kullanılabilir sinyal bant genişliğini belirlemez. Nyquist koşulu gereği örnekleme frekansı ölçülecek en yüksek frekansın en az iki katı olmalı, pratikte analog/dijital anti-alias filtre için daha fazla marj bırakılmalıdır.

Titreşim içeren uygulamalarda yanlış ODR veya filtre seçimi, yüksek frekanslı hareketin düşük frekansta sahte bir sinyal olarak görünmesine yol açabilir. ICM-42688-P içindeki UI filtreleri ve anti-alias filtreler uygulamanın gürültü/gecikme gereksinimine göre ayarlanmalıdır.

3.5 Bias, gürültü ve sıcaklık etkisi

Başlıca hata kaynakları:

- **Bias:** Gerçek giriş sıfırken ölçülen sabit veya yavaş değişen ofset.
- **Scale-factor error:** Gerçek fiziksel büyüklük ile sayısal ölçek arasındaki oransal hata.
- **Noise:** Örnekten örneğe rastgele değişim.
- **Bias instability:** Bias'ın zaman içinde rastgele dolaşması.
- **Temperature coefficient:** Bias ve hassasiyetin sıcaklıkla değişmesi.
- **Cross-axis sensitivity:** Bir eksendeki hareketin diğer eksenlerde görülmesi.
- **Nonlinearity:** Giriş-çıkış ilişkisinin ideal doğrudan sapması.
- **Saturation:** Giriş FSR sınırını aştığında çıktının üst/alt sınıra yapışması.

Jiroskop bias'ı açı kestiriminde özellikle önemlidir. Durgun başlangıçta yüzlerce örneğin ortalaması alınarak başlangıç bias'ı tahmin edilebilir. Sıcaklık değişen uygulamalarda birden fazla sıcaklık noktasında kalibrasyon tablosu gerekir.

3.6 Quantization ve 16 bit UI verisi

UI registerları iki'nin tümleyeni biçiminde 16 bit veri döndürür. `0x0000` sıfır, `0x7FFF` en büyük pozitif, `0x8001` büyük negatif değerdir. `0x8000` bazı sensör durumlarında geçersiz veri işareti olarak görülebilir; fiziksel `-FSR` ölçümü olarak körlemesine kullanılmamalıdır.

FIFO yüksek çözünürlük biçimi etkinleştirildiğinde ivme ve jiroskop için 16 bitten daha geniş etkin veri taşınabilir. Bu veri UI register okumasından farklı paketlenir ve ayrı parser gerektirir.

4 ICM-42688-P mimarisi

ICM-42688-P içindeki temel veri yolu şöyledir:

```
MEMS ivmeölçer –> ADC –> anti-alias/UI filtre ─┬─> UI accel registerları
                                               ├─> FIFO
MEMS jiroskop  –> ADC –> anti-alias/UI filtre ─┤
                                               └─> interrupt/APEX olayları

Sıcaklık sensörü ──────────────────────────────> TEMP_DATA / FIFO
```

Cihaz, doğrudan UI register okuması veya FIFO üzerinden kullanılabilir:

- **UI register yolu:** Basit, düşük gecikmeli, tek güncel örneği verir.
- **FIFO yolu:** İşlemcinin seyrek uyanmasına, örnek kaybını azaltmaya, timestamp ve yüksek çözünürlük verisi taşımaya uygundur.

4.1 Cihaz kimliği

Cihaz	WHO_AM_I (0x75)	Varsayılan accel FSR	Varsayılan gyro FSR
ICM-42688-P	0x47	±16 g	±2000 dps
ICM-42686-P	0x44	±32 g	±4000 dps

Sahada `0x47` okunduğu için ölçekler ICM-42688-P'ye göre uygulanmalıdır. `0x47` değerini `0x44` bekleyen kodla hata saymak, doğru çalışan SPI bağlantısını yanlış teşhis eder.

5 Register bank kavramı

ICM-42688-P'nin register adres alanı bankalara ayrılır. Aynı 8 bit offset, farklı bankalarda farklı registera karşılık gelebilir. Aktif banka `REG_BANK_SEL` (`0x76`) ile seçilir.

```
REG_BANK_SEL[2:0]
000 -> Bank 0
001 -> Bank 1
010 -> Bank 2
011 -> Bank 3
100 -> Bank 4
```

Bank 0; güç, ODR, FSR, UI sensör verisi, FIFO ve interrupt gibi normal çalışma registerlarını içerir. Diğer bankalar daha çok anti-alias filtre, self-test ve APEX ayrıntıları içindir.

Sağlam sürücü tasarımı aktif bankayı yazılımda takip etmeli ve genel ölçüm fonksiyonlarına dönmeden önce Bank 0'a dönmelidir. Aksi halde `0x4E` gibi bir offset yanlış bankadan okunabilir.

5.1 Register erişim türleri

Kısaltma	Anlamı	Yazılım etkisi
R	Read-only	Yazma yapılmaz
W	Write-only	Read-back anlamlı olmayabilir
R/W	Read/write	Gerekirse read-modify-write yapılabilir
R/C	Read-to-clear	Okuma olay bayrağını temizleyebilir
W/C	Write-to-clear / komut	Belirli bite 1 yazmak olayı tetikler/temizler

Kısaltma	Anlamı	Yazılım etkisi
SYNCR	Senkron sensör verisi	Çok byte burst ile tutarlı örnek okunmalıdır

Reserved bitlere veri sayfasında belirtilmeyen değer yazılmamalıdır. Read-modify-write yapılırken reserved bitler korunmalı; sensör çalışırken değiştirilebilen registerlar hakkındaki veri sayfası kısıtlarına uyulmalıdır.

6 SPI register protokolü

ICM-42688-P dört telli SPI'da CS, SCLK, SDI/MOSI ve SDO/MISO hatlarını kullanır. Register komut byte'ının bit 7'si yön bilgisidir:

```
Okuma komutu : 0x80 | (register_offset & 0x7F)
Yazma komutu :         register_offset & 0x7F
```

Tek register okuma örneği:

```
CS = 0
TX: 0xF5       // 0x80 | WHO_AM_I(0x75)
TX: 0xFF       // dummy byte
RX: 0x47       // ikinci byte sırasında alınır
CS = 1
```

Çok byte sensör verisi tek CS penceresinde burst okunmalıdır:

```
CS = 0
TX: 0x80 | 0x1D
RX: TEMP_H, TEMP_L, AX_H, AX_L, AY_H, AY_L, AZ_H, AZ_L,
    GX_H, GX_L, GY_H, GY_L, GZ_H, GZ_L
CS = 1
```

Adres otomatik ilerlediği için 0x1D'den başlayan 14 byte okuma, sıcaklık ve altı ekseni aynı ölçüm grubunda taşır. Her 16 bit değer varsayılan durumda high-byte önce olacak şekilde birleştirilir.

6.1 SPI mode konusu

`DEVICE_CONFIG.SPI_MODE` reset değeri 0'dır ve sensör Mode 0/Mode 3 kabul eder. Bit 4'e 1 yazılırsa Mode 1/Mode 2 seçilir. Sahadaki deneyde sensör varsayılan modda dört kez `0x47` döndürmüş, `SPI_MODE=1` yazıldıktan sonra okumalar `0x00` olmuştur. Bu nedenle mevcut FPGA bitstream ile bu bit **değiştirilmemelidir**.

Bu bulgu önemlidir: registera yazılan değerin mantıksal olarak doğru görünmesi, master ve slave edge zamanlaması uyuşmuyorsa haberleşmeyi bozabilir.

7 Bank 0 register haritası — temel alanlar

Aşağıdaki tablo normal sürücü geliştirme için en önemli Bank 0 registerlarını özetler.

Offset	Register	Erişim	Amaç
0x11	DEVICE_CONFIG	R/W	SPI mode ve soft reset
0x13	DRIVE_CONFIG	R/W	I²C/SPI çıkış slew-rate ayarı
0x14	INT_CONFIG	R/W	INT1/INT2 polarite, sürüş ve pulse/latch davranışı

Offset	Register	Erişim	Amaç
0x16	FIFO_CONFIG	R/W	FIFO bypass/stream/stop-on-full modu
0x1D	TEMP_DATA1	SYNCR	Sıcaklık high byte
0x1E	TEMP_DATA0	SYNCR	Sıcaklık low byte
0x1F	ACCEL_DATA_X1	SYNCR	Accel X high byte
0x20	ACCEL_DATA_X0	SYNCR	Accel X low byte
0x21	ACCEL_DATA_Y1	SYNCR	Accel Y high byte
0x22	ACCEL_DATA_Y0	SYNCR	Accel Y low byte
0x23	ACCEL_DATA_Z1	SYNCR	Accel Z high byte
0x24	ACCEL_DATA_Z0	SYNCR	Accel Z low byte
0x25	GYRO_DATA_X1	SYNCR	Gyro X high byte
0x26	GYRO_DATA_X0	SYNCR	Gyro X low byte
0x27	GYRO_DATA_Y1	SYNCR	Gyro Y high byte
0x28	GYRO_DATA_Y0	SYNCR	Gyro Y low byte
0x29	GYRO_DATA_Z1	SYNCR	Gyro Z high byte
0x2A	GYRO_DATA_Z0	SYNCR	Gyro Z low byte
0x2B	TMST_FSYNCH	SYNCR	Timestamp/FSYNC high byte
0x2C	TMST_FSYNCL	SYNCR	Timestamp/FSYNC low byte
0x2D	INT_STATUS	R/C	Data-ready, reset, PLL ve FIFO olayları
0x2E	FIFO_COUNTH	R	FIFO sayacı high byte
0x2F	FIFO_COUNTL	R	FIFO sayacı low byte
0x30	FIFO_DATA	R	FIFO veri portu
0x31–0x36	APEX_DATA0–5	R/SYNCR	Adım, cadence, aktivite ve tap sonuçları
0x37	INT_STATUS2	R/C	SMD ve wake-on-motion durumları
0x38	INT_STATUS3	R/C	Step, tilt, tap ve wake/sleep olayları
0x4B	SIGNAL_PATH_RESET	W/C	FIFO flush, timestamp strobe, signal-path reset
0x4C	INTF_CONFIG0	R/W	Endian, FIFO count ve seri arayüz seçenekleri
0x4D	INTF_CONFIG1	R/W	Clock/RTC/LP clock seçimi
0x4E	PWR_MGMT0	R/W	Accel/gyro/sıcaklık güç modları
0x4F	GYRO_CONFIG0	R/W	Gyro FSR ve ODR
0x50	ACCEL_CONFIG0	R/W	Accel FSR ve ODR
0x51	GYRO_CONFIG1	R/W	Gyro filtre dereceleri ve sıcaklık filtresi
0x52	GYRO_ACCEL_CONFIG0	R/W	Gyro/accel UI filtre bant genişliği

Offset	Register	Erişim	Amaç
0x53	ACCEL_CONFIG1	R/W	Accel filtre dereceleri
0x54	TMST_CONFIG	R/W	Timestamp çözünürlüğü ve etkinleştirme
0x56	APEX_CONFIG0	R/W	APEX/DMP temel yapılandırması
0x57	SMD_CONFIG	R/W	Significant motion / WOM modu
0x5F	FIFO_CONFIG1	R/W	FIFO'ya accel/gyro/temp/timestamp ve high-res ekleme
0x60	FIFO_CONFIG2	R/W	FIFO watermark alt byte
0x61	FIFO_CONFIG3	R/W	FIFO watermark üst alanı
0x62–0x64	FSYNC_CONFIG, INT_CONFIG0/1	R/W	FSYNC ve interrupt pulse ayarları
0x65	INT_SOURCE0	R/W	UI olaylarını INT1'e yönlendirme
0x66	INT_SOURCE1	R/W	WOM/SMD olaylarını INT1'e yönlendirme
0x68	INT_SOURCE3	R/W	UI olaylarını INT2'ye yönlendirme
0x69	INT_SOURCE4	R/W	WOM/SMD olaylarını INT2'ye yönlendirme
0x6B–0x6F	FIFO_LOST_PKT*, SELF_TEST_CONFIG	R/RW	Kayıp paket sayacı ve self-test
0x75	WHO_AM_I	R	Cihaz kimliği; ICM-42688-P = 0x47
0x76	REG_BANK_SEL	R/W	Aktif register bankı

8 Temel registerların bit düzeyinde açıklaması

8.1 DEVICE_CONFIG — 0x11

Bit	Alan	Açıklama
7:5	Reserved	Değiştirilmez
4	SPI_MODE	0: Mode 0/3; 1: Mode 1/2
3:1	Reserved	Değiştirilmez
0	SOFT_RESET_CONFIG	1 yazılması soft reset başlatır

Soft reset sonrası en az 1 ms register erişimi yapılmamalıdır. Reset SPI mode bitini de reset değerine döndürdüğü için master'ın hangi mode ile devam edeceği önceden planlanmalıdır.

Bu projede mevcut bitstream ile SPI_MODE=1 yazılmamalıdır.

8.2 INT_CONFIG — 0x14

INT1 ve INT2 pinlerinin elektriksel davranışını belirler. Her pin için üç kavram vardır:

- **Polarity:** Aktif-high veya aktif-low.
- **Drive circuit:** Push-pull veya open-drain.
- **Mode:** Pulse veya latched davranış.

Open-drain seçilirse harici pull-up gerekir. FPGA Pmod girişinde push-pull, aktif-high pulse çoğu basit uygulama için kolaydır; ancak registerın tam değeri kullanılan interrupt temizleme akışıyla birlikte seçilmelidir.

8.3 INT_STATUS — 0x2D

Bit	Olay
6	UI_FSYNC_INT
5	PLL_RDY_INT
4	RESET_DONE_INT
3	DATA_RDY_INT
2	FIFO_THS_INT
1	FIFO_FULL_INT
0	AGC_RDY_INT

Projede gözlenen `INT_STATUS=0x08`, `DATA_RDY_INT` bitinin set olduğunu ve yeni UI sensör verisinin hazır olduğunu gösterir. Register R/C olduğundan debug amaçlı fazladan okuma interrupt bilgisini tüketebilir.

8.4 SIGNAL_PATH_RESET — 0x4B

Bu register sürekli konfigürasyon tutmaktan çok komut üretir. Önemli işlemler:

- FIFO flush,
- timestamp strobe,
- signal-path abort/reset,
- DMP başlatma veya bellek reset işlemleri.

FIFO flush sırasında sensör ve FIFO durumu veri sayfasındaki sıraya göre yönetilmelidir. W/C registera read-modify-write uygulanması genellikle doğru değildir; gerekli komut biti doğrudan yazılır.

8.5 INTF_CONFIG0 — 0x4C

Bu register FIFO sayacının birim/endian yorumunu ve sensör verisinin byte sırasını etkiler. Varsayılan register veri sırası high-byte/low-byte olarak kullanılmaktadır. Endian ayarı değiştirilirse hem UI verisi hem FIFO parser'ı gözden geçirilmelidir.

Bir byte-order hatası tipik olarak tamamen sıfır yerine anlamsız fakat değişen büyük sayılar üretir. Örneğin gerçek `0x0400` değeri ters birleştirilirse `0x0004` olur.

8.6 PWR_MGMT0 — 0x4E

Bit	Alan	Değerler
5	TEMP_DIS	`0`: sıcaklık açık; `1`: kapalı
4	IDLE	RC oscillator'ı sensörler kapalıyken açık tutma
3:2	GYRO_MODE	`00`: off; `01`: standby; `11`: low-noise
1:0	ACCEL_MODE	`00/01`: off; `10`: low-power; `11`: low-noise

`0x0F` yazmak gyro ve accel'i low-noise moduna alır ve sıcaklık sensörünü açık bırakır:

```
0x0F = 0000_1111b
            ^^ accel LN
          ^^   gyro LN
```

OFF'tan aktif moda geçerken veri sayfasındaki başlangıç süreleri beklenmelidir. Proje kodu gyro'nun kararlı hale gelmesi için 50 ms bekler. İlk örnekler `DATA_RDY` görülmeden kullanılmamalıdır.

8.7 GYRO_CONFIG0 — 0x4F

```
bit 7:5  GYRO_FS_SEL
bit 4    reserved
bit 3:0  GYRO_ODR
```

8.7.1 GYRO_FS_SEL

Kod	FSR	Yaklaşık hassasiyet
000	±2000 dps	16.4 LSB/(°/s)
001	±1000 dps	32.8 LSB/(°/s)
010	±500 dps	65.5 LSB/(°/s)
011	±250 dps	131 LSB/(°/s)
100	±125 dps	262 LSB/(°/s)
101	±62.5 dps	524.3 LSB/(°/s)
110	±31.25 dps	1048.6 LSB/(°/s)
111	±15.625 dps	2097.2 LSB/(°/s)

8.7.2 GYRO_ODR

Kod	ODR
0x1	32 kHz
0x2	16 kHz
0x3	8 kHz
0x4	4 kHz
0x5	2 kHz
0x6	1 kHz — reset değeri
0x7	200 Hz
0x8	100 Hz
0x9	50 Hz
0xA	25 Hz
0xB	12.5 Hz
0xF	500 Hz

Reset değeri `0x06`, ±2000 dps ve 1 kHz anlamına gelir.

8.8 ACCEL_CONFIG0 — 0x50

```
bit 7:5  ACCEL_FS_SEL
bit 4    reserved
bit 3:0  ACCEL_ODR
```

8.8.1 ACCEL_FS_SEL

Kod	FSR	Hassasiyet
000	±16 g	2048 LSB/g
001	±8 g	4096 LSB/g
010	±4 g	8192 LSB/g
011	±2 g	16384 LSB/g
100–111	Reserved	Kullanılmaz

8.8.2 ACCEL_ODR

`0x1–0xB` ve `0xF` kodları gyro tablosuyla aynıdır. Accel low-power modunda ayrıca:

Kod	ODR
0xC	6.25 Hz
0xD	3.125 Hz
0xE	1.5625 Hz

Reset değeri `0x06`, ±16 g ve 1 kHz anlamına gelir.

8.9 FIFO_CONFIG1 — 0x5F

FIFO'ya hangi kaynakların yazılacağını belirleyen ana registerdır. Alanlar; accel, gyro, sıcaklık, timestamp/FSYNC ve yüksek çözünürlük verisinin eklenmesini kontrol eder. FIFO kullanılmadan önce şu kararlar verilmelidir:

1. Packet formatı 16 bit mi high-resolution mı?
2. Accel ve gyro birlikte mi, ayrı mı?
3. Sıcaklık ve timestamp gerekli mi?
4. FIFO count byte mı packet mı sayacak?
5. Watermark kaç örnek olacak?
6. Overflow davranışı stream mi stop-on-full mı olacak?

Parser, konfigürasyondan bağımsız yazılmamalıdır; packet header'daki geçerlilik ve format bitlerini kontrol etmelidir.

8.10 WHO_AM_I — 0x75

Read-only cihaz kimliğidir. Güç açıldıktan sonra sürücünün ilk doğrulaması olmalıdır.

```
ICM-42688-P -> 0x47
ICM-42686-P -> 0x44
```

Beklenmeyen değer durumunda sensör konfigürasyonu yazılmamalıdır. `0x00` genellikle MISO'nun düşük kalması, CS/power problemi veya yanlış SPI mode; `0xFF` ise MISO'nun pull-up ile boşta kalması/CS seçilmemesi belirtisi olabilir.

9 Sensör verisi registerları

9.1 Burst veri düzeni

`TEMP_DATA1` (`0x1D`) adresinden 14 byte burst okumanın düzeni:

Byte indisi	Offset	İçerik
0	`0x1D`	Temperature high
1	`0x1E`	Temperature low
2	`0x1F`	Accel X high
3	`0x20`	Accel X low
4	`0x21`	Accel Y high
5	`0x22`	Accel Y low
6	`0x23`	Accel Z high
7	`0x24`	Accel Z low
8	`0x25`	Gyro X high
9	`0x26`	Gyro X low
10	`0x27`	Gyro Y high
11	`0x28`	Gyro Y low
12	`0x29`	Gyro Z high
13	`0x2A`	Gyro Z low

High ve low byte birleştirme:

```
static int16_t be16(const uint8_t *p)
{
    return (int16_t)(((uint16_t)p[0] << 8) | p[1]);
}
```

Cast'in en sonda `int16_t` yapılması, negatif iki'nin tümleyeni değerlerinin doğru yorumlanmasını sağlar.

9.2 Sıcaklık dönüşümü

16 bit sıcaklık registerı için projede kullanılan dönüşüm:

```
T_degC = raw_temp / 132.48 + 25
```

Floating point kullanmadan milli-degree Celsius:

```
temp_mC = 25000 + ((int64_t)raw_temp * 100000) / 13248;
```

Sıcaklık registerı ortam sıcaklığı termometresi olarak değil, öncelikle sensör bias/sensitivity sıcaklık kompanzasyonu için düşünülmelidir.

9.3 İvme dönüşümü

Genel formül:

```
accel_mg = ((int64_t)raw * accel_full_scale_mg) / 32768;
```

ICM-42688-P reset FSR için:

```
accel_full_scale_mg = 16000;
```

FSR registerı değiştirilirse yazılım ölçeği de aynı anda güncellenmelidir. En güvenilir yöntem, kullanılan ölçeği yalnız sabit bir cihaz adına göre değil `ACCEL_CONFIG0[7:5]` read-back değerine göre hesaplamaktır.

9.4 Jiroskop dönüşümü

```
gyro_mdps = ((int64_t)raw * gyro_full_scale_mdps) / 32768;
```

ICM-42688-P reset FSR için:

```
gyro_full_scale_mdps = 2000000;
```

`mdps / 1000` sonucu derece/saniyedir. Kontrol ve füzyon algoritmaları genellikle rad/s ister:

```
gyro_rad_s = gyro_dps * pi / 180
```

10 FIFO kavramı ve kullanımı

FIFO, host işlemcisinin her ODR periyodunda register okumak zorunda kalmasını önler. Sensör örnekleri FIFO'ya yazar; host watermark interrupt geldiğinde bir blok veriyi burst okur.

10.1 FIFO kullanmanın avantajları

- Host CPU/FPGA yazılımının uyanma sayısını azaltır.
- SPI üzerindeki anlık gecikmelerde örnek kaybını azaltır.
- Accel ve gyro örneklerini packet içinde birlikte taşır.
- Timestamp ile daha doğru `dt` hesabı sağlar.
- High-resolution veriye erişim sağlar.

10.2 FIFO count

`FIFO_COUNTH/L` okunarak hazır veri miktarı belirlenir. Count'un byte veya record biriminde yorumlanması `INTF_CONFIG0` ayarına bağlıdır. Sürücü bu ayarı bilmeden sayacı doğrudan "örnek sayısı" kabul etmemelidir.

10.3 FIFO watermark

Watermark, FIFO belirli doluluk seviyesine ulaştığında interrupt üretir. Küçük watermark düşük gecikme fakat fazla interrupt; büyük watermark daha verimli burst fakat daha yüksek gecikme üretir.

Örnek seçim:

```
ODR = 1 kHz
Watermark = 20 packet
Yaklaşık interrupt periyodu = 20 ms
```

10.4 FIFO overflow ve kayıp packet

Host FIFO'yu yeterince hızlı boşaltmazsa overflow oluşabilir. `FIFO_FULL_INT`, FIFO lost-packet registerları ve packet header geçerliliği izlenmelidir. Overflow sonrası eski/yeni veri politikasına göre FIFO flush ve yeniden başlatma gerekebilir.

11 Interrupt sistemi

ICM-42688-P iki interrupt çıkışı sağlar. Bir olayın pine ulaşması için üç katman doğru ayarlanmalıdır:

1. Olayın üreten fonksiyonu etkin olmalı.
2. Olay `INT_SOURCE*` registerıyla INT1 veya INT2'ye yönlendirilmeli.
3. `INT_CONFIG` elektriksel/pulse davranışı FPGA girişine uygun olmalı.

Basit data-ready akışı:

```
Accel/Gyro etkinleştir
        ↓
INT_SOURCE0 içinde DATA_RDY -> INT1 yönlendir
        ↓
INT1 kenarı FPGA PLIC interrupt üretir
        ↓
ISR INT_STATUS okur
        ↓
0x1D'den 14 byte burst okur
```

Projede `pmod_irq`, Genesys2 JC9 pininden PLIC interrupt ID 11'e bağlanmıştır. Polling testinde bu hat kullanılmasa da üretim sürücüsü için data-ready veya FIFO watermark interrupt tercih edilebilir.

12 APEX ve hareket olayları

APEX, TDK'nin cihaz içi hareket işleme bloğudur. Temel olaylar cihaz/silikon varyantına göre adım sayma, tilt, tap, wake-on-motion ve significant-motion detection içerir.

APEX kullanımında yalnız bir enable biti yazmak genellikle yeterli değildir. Gerekli sıra:

1. Accel ODR ve güç modunu APEX gereksinimine uygun seçmek.
2. APEX/DMP parametrelerini Bank 4 registerlarında yapılandırmak.
3. DMP init işlemini başlatmak ve hazır durumunu beklemek.
4. İlgili olayı INT1/INT2'ye yönlendirmek.
5. `APEX_DATA*` ve `INT_STATUS2/3` registerlarını doğru temizleme semantiğiyle okumak.

APEX parametreleri normal accel/gyro UI sürücüsünden ayrı bir modül olarak ele alınmalıdır.

13 Self-test

Self-test, sensör MEMS ve analog zincirine dahili uyarım uygulayarak cevabın beklenen aralıkta olup olmadığını kontrol eder. Kalibrasyonla aynı şey değildir.

Genel işlem:

1. Sensörü kararlı güç/ODR/FSR koşuluna getir.
2. Self-test kapalı çok sayıda örneğin ortalamasını al.
3. `SELF_TEST_CONFIG` ve gerekli Bank 1/2 registerlarıyla uyarımı etkinleştir.

4. Veri sayfasındaki yerleşme süresini bekle.
5. Self-test açık ortalamayı al.
6. Açık-kapalı farkını factory self-test kodları ve kabul limitleriyle karşılaştır.
7. Self-test bitlerini kapatıp normal konfigürasyonu geri yükle.

Self-test sırasında kart hareket etmemelidir. Yanlış FSR/ODR/filtreyle alınan sonuç doğru limitlerle karşılaştırılamaz.

14 Güvenilir başlatma sırası

Önerilen UI register okuma sırası:

```
1. DK/IMU gücünü ve VDDIO seviyesini doğrula
2. CS high ve SCLK idle durumunu oluştur
3. Güç açılışından sonra register erişim süresini bekle
4. REG_BANK_SEL = 0 yaz
5. WHO_AM_I oku
   - 0x47: ICM-42688-P
   - 0x44: ICM-42686-P
   - başka değer: dur ve hata raporla
6. DEVICE_CONFIG.SPI_MODE değerini değiştirme
7. GYRO_CONFIG0 ve ACCEL_CONFIG0 ile FSR/ODR seç
8. Gerekli filtreleri sensörler kapalıyken yapılandır
9. PWR_MGMT0 = 0x0F ile accel+gyro LN aç
10. En az gerekli startup süresini bekle
11. INT_STATUS.DATA_RDY bekle
12. 0x1D'den 14 byte burst oku
13. 0x8000 geçersiz kodlarını kontrol et
14. FSR read-back değerine göre fiziksel birime dönüştür
```

Sensör çalışırken veri sayfasının izin vermediği konfigürasyon registerları değiştirilecekse accel ve gyro kapatılmalı, registerlar yazılmalı ve sensörler yeniden açılmalıdır.

15 Bu projedeki C sürücüsüyle eşleme

Uygulama kaynakları:

```
soc/software/pmod_spi_test/main.c
soc/software/bsp/drivers/pmod_spi.c
soc/software/bsp/include/bsp/pmod_spi.h
```

15.1 FPGA SPI MMIO registerları

PMOD SPI tabanı `0x10015000`'dir:

MMIO offset	Register	Açıklama
`0x00`	CTRL	bit0 busy, bit1 CSN, bit2 quad
`0x04`	TX	Yazma transferi başlatır
`0x08`	RX	Son alınan shift register değeri
`0x0C`	READY	IP yeni transfer kabul edebilir

SPI IP 50 MHz fabric clock ve sabit 8 divider ile yaklaşık 3.125 MHz SCK üretir. Bu değer ICM-42688-P'nin 24 MHz üst sınırının altındadır.

15.2 Register okuma fonksiyonu

```
static void icm_read_regs(uint8_t reg, uint8_t *data, unsigned len)
{
    bsp_pmod_spi_select();
    delay_us(10u);
    (void)bsp_pmod_spi_xfer8(reg | 0x80u);
    delay_us(10u);

    for (unsigned i = 0; i < len; ++i) {
        data[i] = bsp_pmod_spi_xfer8(0xffu);
        delay_us(10u);
    }

    bsp_pmod_spi_deselect();
    delay_us(10u);
}
```

Burada CS tüm komut+veri boyunca low kalır. `0x80` read bitidir. Dummy `0xFF` byte'ları SCLK üretmek için gönderilir.

15.3 Register yazma fonksiyonu

```
static void icm_write_reg(uint8_t reg, uint8_t value)
{
    bsp_pmod_spi_select();
    delay_us(10u);
    (void)bsp_pmod_spi_xfer8(reg & 0x7fu);
    delay_us(10u);
    (void)bsp_pmod_spi_xfer8(value);
    delay_us(10u);
    bsp_pmod_spi_deselect();
    delay_us(10u);
}
```

Birden fazla register yazılacaksa her registerın erişim ve çalışma sırasında değiştirilebilme kuralı ayrı değerlendirilmelidir.

16 Sahada görülen sonuçların yorumlanması

16.1 WHO_AM_I: 47 47 47 47

Bu en güçlü olumlu işarettir:

- Sensör besleniyor.
- CS doğru sensörü seçiyor.
- SCK sensöre ulaşıyor.
- MOSI komut byte'ını taşıyor.
- MISO veriyi geri getiriyor.
- Aktif sensör ICM-42688-P.

16.2 Mode yazısından sonra 0x00

Önce `0x47`, `DEVICE_CONFIG.SPI_MODE=1` yazısından sonra `0x00` görülmesi, fiziksel bağlantının değil SPI edge eşleşmesinin bozulduğunu gösterir. Çözüm registerı reset değeri `0x00`'da bırakmaktır.

16.3 Geçerli ölçümler

```
T ≈ 26 °C
A ≈ -7, -12, 1037 mg
G ≈ -122, -488, 488 mdps
INT_STATUS = 0x08
```

Bu örneklerde Z ekseninin yaklaşık +1 g göstermesi ve `DATA_RDY` bitinin set olması sensörün normal örnek ürettiğini gösterir.

16.4 Daha sonra tüm kanallarda alt sınır

```
A = -16000,-16000,-16000 mg
G = -2000000,-2000000,-2000000 mdps
```

Bu değerler her raw kanalın `0x8000` döndürdüğünü gösterir. Fiziksel olarak altı eksenin aynı anda tam negatif FSR'a gitmesi gerçekçi değildir; geçersiz veri/sensörün kapanması olarak ele alınmalıdır. Olası nedenler:

- SAMG55 resetten çıkıp aynı sensörü yeniden yapılandırıyor.
- `PWR_MGMT0` başka master tarafından değiştiriliyor.
- Sensör geçici reset veya güç kesintisi görüyor.
- Yanlış bankta kalındı veya konfigürasyon çalışma sırasında bozuldu.
- Burst sırasında bus contention oluşuyor.

Üretim kodu `0x8000` değerini ölçüm olarak yayınlamamalı; hata sayacı artırıp `WHO_AM_I`, `PWR_MGMT0`, `GYRO_CONFIG0`, `ACCEL_CONFIG0` ve aktif bankı yeniden okumalıdır.

17 IMU'dan alınabilen bilgi türleri

Bilgi	Register/FIFO kaynağı	Ham biçim	Fiziksel yorum
Yonga sıcaklığı	`TEMP_DATA1/0`	signed 16 bit	°C dönüşümü; bias sıcaklık kompanzasyonu
X/Y/Z özgül kuvvet	`ACCEL_DATA_*`	3 × signed 16 bit	g, mg veya m/s²
X/Y/Z açısal hız	`GYRO_DATA_*`	3 × signed 16 bit	dps, mdps veya rad/s
Yeni veri durumu	`INT_STATUS` bit 3	bit/bayrak	UI registerlarında yeni örnek hazır
Reset/PLL/AGC durumu	`INT_STATUS`	bit/bayrak	Sensör başlangıç ve clock durumu
FIFO doluluğu	`FIFO_COUNTH/L`	unsigned sayaç	Hazır byte veya packet miktarı
FIFO sensör packetleri	`FIFO_DATA`	değişken packet	Accel, gyro, sıcaklık, timestamp, high-res
Sensor timestamp/FSYNC	`TMST_FSYNC*` veya FIFO	unsigned sayaç	Örnek zamanı ve harici senkronizasyon
Adım sayısı/cadence	`APEX_DATA*`	sayaç/kod	Pedometre sonucu
Tap/tilt/aktivite	`APEX_DATA*`, `INT_STATUS3`	bit/kod	Hareket sınıfı ve olay
Wake-on-motion	`INT_STATUS2`, APEX/WOM config	bit	Düşük güçte hareket algılama

Bilgi	Register/FIFO kaynağı	Ham biçim	Fiziksel yorum
Self-test fabrika kodları	Bank 1/2	unsigned byte	Dahili self-test değerlendirmesi

Accel ve gyro UI registerları bir yönelim açısı değildir. Roll/pitch/yaw, hız veya konum üretmek için bu ham bilgiler bir algoritmada işlenmelidir. Sıcaklık verisi de doğrudan ortam termometresi doğruluğu garanti etmez.

17.1 Önerilen yazılım örnek yapısı

```
typedef struct {
    uint64_t host_time_ticks;
    uint16_t sensor_timestamp;
    int16_t temp_raw;
    int16_t accel_raw[3];
    int16_t gyro_raw[3];
    int32_t temp_mC;
    int32_t accel_mg[3];
    int32_t gyro_mdps[3];
    uint8_t int_status;
    uint8_t valid;
} imu_sample_t;
```

Ham verinin fiziksel verinin yanında saklanması hata ayıklama ve daha sonra yeniden kalibrasyon için faydalıdır. `valid`, WHO_AM_I, data-ready, `0x8000` kontrolü ve bus hata durumlarından üretilmelidir.

18 Kalibrasyon ve doğrulama

Kalibrasyon, sensör çıktısını gerçek fiziksel değere daha iyi yaklaştıran katsayıların ölçülmesidir. Kalibrasyon register konfigürasyonu veya self-test ile aynı şey değildir:

- **Register konfigürasyonu:** FSR, ODR, filtre ve güç modu seçer.
- **Self-test:** MEMS/analog zincirin cevap verip vermediğini sınar.
- **Kalibrasyon:** Bias, scale ve eksen hatalarını ölçüp düzeltme katsayısı üretir.

Kalibrasyon boyunca FSR, ODR, filtre, güç modu ve besleme sabit tutulmalıdır. Katsayılar hangi konfigürasyonda üretildiyse kayıt altına alınmalıdır.

18.1 Kalibrasyondan önce hazırlanacak registerlar

Örnek bir başlangıç noktası:

```
REG_BANK_SEL   = 0x00   Bank 0
GYRO_CONFIG0   = 0x06   ±2000 dps, 1 kHz
ACCEL_CONFIG0  = 0x06   ±16 g, 1 kHz
PWR_MGMT0      = 0x0F   Accel LN + Gyro LN
```

Kalibrasyonda dar FSR daha yüksek çözünürlük sağlayabilir. Örneğin yavaş sistemlerde gyro ±250 dps ve accel ±2 g seçilebilir. FSR değiştirilirse dönüşüm katsayıları da değişmelidir.

18.2 1. Jiroskop sıfır bias kalibrasyonu

18.2.1 Koşullar

- Kart tamamen sabit tutulur.
- Titreşim ve masa hareketi azaltılır.

- Gyro açıldıktan sonra en az başlangıç süresi beklenir; daha iyi sonuç için termal denge beklenir.
- `DATA_RDY` ile yeni örnekler alınır.
- `0x8000` ve bus hatalı örnekler dışlanır.

18.2.2 Hesap

Her eksen için N örnek ortalaması:

```
b_gx = (gx[0] + gx[1] + ... + gx[N-1]) / N
b_gy = mean(gy)
b_gz = mean(gz)
```

N için 2000–10000 örnek pratik bir başlangıçtır. 1 kHz ODR'de 5000 örnek yaklaşık 5 saniyedir.

Düzeltilmiş gyro:

```
g_corrected_raw = g_raw - b_g
```

vektörel biçimde:

```
omega_corrected = Sg * (omega_raw - bg)
```

Burada `bg` üç eksenli bias, `Sg` ise scale/misalignment matrisidir. Basit kalibrasyonda `Sg` birim matris kabul edilir.

18.2.3 Kalite kontrolü

Bias çıkarıldıktan sonra durgun ortalama sıfıra yakın olmalıdır. Standart sapma bias değildir; gürültüyü gösterir. Ortalama değişmeye devam ediyorsa sensör termal dengeye ulaşmamış veya platform hareket ediyor olabilir.

18.3 2. İvmeölçer altı-yüz kalibrasyonu

Yerçekimi hassas bir referans olarak kullanılır. Kart sırayla altı sabit pozisyonda tutulur:

```
+X yukarı, -X yukarı,
+Y yukarı, -Y yukarı,
+Z yukarı, -Z yukarı
```

Her pozisyonda 1000–5000 yeni örnek ortalaması alınır. Örneğin X ekseni için:

```
x_plus  = +X yukarı konumundaki ortalama raw X
x_minus = -X yukarı konumundaki ortalama raw X
```

Bias:

```
b_x = (x_plus + x_minus) / 2
```

Ölçülen counts-per-g:

```
s_x = (x_plus - x_minus) / 2
```

Düzeltilmiş ivme:

```
a_x_g = (raw_x - b_x) / s_x
```

Aynı işlem Y ve Z için uygulanır. İdeal ±16 g ayarında `s_x`, `s_y`, `s_z` yaklaşık 2048 LSB/g olmalıdır.

18.3.1 Örnek

```
x_plus  = +2070
x_minus = -2026

b_x = (2070 - 2026) / 2 = 22 LSB
s_x = (2070 - (-2026)) / 2 = 2048 LSB/g
```

Bu durumda:

```
a_x_g = (raw_x - 22) / 2048
```

18.4 3. Cross-axis ve misalignment kalibrasyonu

Basit altı-yüz yöntemi her ekseni bağımsız düzeltir. Daha yüksek doğrulukta eksenler arası kuplaj ve mekanik montaj açısı 3×3 matrisle modellenir:

```
a_cal = Ma * (a_raw - ba)
```

`Ma` matrisinin diagonal elemanları scale-factor, diagonal dışı elemanları cross-axis/misalignment düzeltmesidir. Katsayılar, birden fazla bilinen yönelimde ölçülen yerçekimi vektörlerine least-squares uygulanarak bulunabilir.

Sensör-kart montaj dönüşümü kalibrasyon matrisinden ayrı tutulabilir:

```
a_system = R_mount * a_cal
omega_system = R_mount * omega_cal
```

Bu ayrım aynı kalibrasyonun farklı mekanik yerleşimlerde kullanılmasını kolaylaştırır.

18.5 4. Jiroskop scale-factor kalibrasyonu

Gyro scale kalibrasyonu için bilinen açısal hız üreten bir rate table en güvenilir yöntemdir. Her eksende pozitif ve negatif bilinen hız uygulanır:

```
measured_plus  = +known_rate ölçüm ortalaması
measured_minus = -known_rate ölçüm ortalaması
```

Bias ve scale:

```
bias  = (measured_plus + measured_minus) / 2
scale = (2 * known_rate) / (measured_plus - measured_minus)
```

Rate table yoksa belirli bir açıyı hassas sürede döndürme yöntemi kullanılabilir, ancak zamanlama ve mekanik hata daha büyüktür.

18.6 5. Sıcaklık kalibrasyonu

Gyro bias ve accel bias sıcaklıkla değişebilir. Sıcaklık kompanzasyonu için cihaz kontrollü sıcaklıklarda sabit tutulur ve her sıcaklıkta bias ölçülür.

Basit lineer model:

```
bias(T) = b0 + k1 * (T - T0)
```

İkinci dereceden model:

```
bias(T) = b0 + k1*(T-T0) + k2*(T-T0)^2
```

Her eksen için ayrı `b0`, `k1`, gerekirse `k2` bulunur. Çalışma sırasında `TEMP_DATA1/0` okunur ve anlık bias hesaplanır:

```
gyro_corr = gyro_raw - gyro_bias(temp_C)
```

Sıcaklık kabini yoksa kontrollü ısınma/soğuma testi yapılabilir; fakat kartın gerçekten sabit kalması, yoğuşma ve sıcaklık gradyanı dikkate alınmalıdır. Sensör iç sıcaklığı ile ortam sıcaklığı aynı olmayabilir.

18.7 6. Gürültü ve tekrarlanabilirlik ölçümü

Durgun veri için her eksende:

```
mean = sum(x_i) / N
variance = sum((x_i - mean)^2) / (N-1)
sigma = sqrt(variance)
```

Rapor şu metadata ile saklanmalıdır:

- WHO_AM_I ve cihaz seri/lot bilgisi,
- FSR ve ODR,
- filtre registerları,
- güç modu,
- sıcaklık,
- örnek sayısı,
- gerçek örnek zaman aralığı,
- ortalama, standart sapma, minimum ve maksimum.

ODR veya filtre değiştiğinde gürültü sonucu da değişir.

18.8 7. Allan variance

Saatler mertebesinde durgun gyro verisine Allan deviation uygulanarak angle random walk, bias instability ve rate random walk davranışları ayrıştırılabilir. Bu analiz kısa bir standart sapma testinden farklıdır ve doğru timestamp gerektirir.

18.9 8. Zaman tabanı kalibrasyonu

Yönelim integrasyonunda nominal `dt` yerine sensör timestamp veya ölçülmüş donanım zamanı kullanılmalıdır:

```
dt = (timestamp[k] - timestamp[k-1]) * timestamp_resolution
```

Timestamp wrap-around unsigned aritmetikle ele alınmalıdır. ODR'nin nominal değerindeki ppm hata uzun süreli açı/hız hesabını etkileyebilir.

18.10 9. Kalibrasyon verisini saklama

Kalibrasyon kaydı cihaz ve konfigürasyonla birlikte saklanmalıdır:

```c
typedef struct {
    uint32_t magic;
    uint16_t version;
    uint8_t who_am_i;
    uint8_t gyro_fs_sel;
    uint8_t accel_fs_sel;
    uint8_t gyro_odr;
    uint8_t accel_odr;
    int32_t gyro_bias_raw[3];
    int32_t accel_bias_raw[3];
    float gyro_matrix[3][3];
    float accel_matrix[3][3];
    float mount_matrix[3][3];
    float gyro_temp_k1[3];
    float gyro_temp_k2[3];
    float reference_temp_C;
    uint32_t crc32;
} imu_calibration_t;
```

`WHO_AM_I`, FSR veya sensör değişirse uyumsuz kalibrasyon reddedilmelidir. CRC, bozulmuş flash kaydının kullanılmasını engeller.

18.11 10. Kalibrasyon sonrası kabul testleri

1. Altı accel yönünde düzeltilmiş büyüklük `|a|` yaklaşık 1 g olmalı.
2. Durgun gyro ortalaması sıfıra yakın olmalı.
3. Pozitif/negatif eksen işaretleri mekanik koordinat tanımıyla uyumlu olmalı.
4. Tekrarlanan power-cycle sonrası bias kabul sınırında olmalı.
5. Birkaç sıcaklık noktasında kompanzasyon hatayı azaltmalı.
6. FIFO ve UI register yolu aynı fiziksel sonucu üretmeli.
7. `0x8000`, FIFO invalid flag veya overflow örnekleri kalibrasyon hesabına alınmamalı.

19 Hata ayıklama tablosu

Belirti	Muhtemel neden	İlk kontrol
WHO_AM_I=`0x00`	MISO low, sensör güçsüz, yanlış mode/CS	VDDIO, CS, logic analyzer
WHO_AM_I=`0xFF`	MISO boşta/pull-up, CS seçmiyor	CS polaritesi ve MISO continuity
WHO_AM_I=`0x47`	ICM-42688-P normal	0x47'yi kabul et, doğru FSR kullan
WHO_AM_I=`0x44`	ICM-42686-P normal	±32g/±4000dps ölçek kullan
Önce `0x47`, sonra `0x00`	SPI_MODE yazısı edge uyumunu bozuyor	DEVICE_CONFIG bit4'ü değiştirme
INT_STATUS=`0x08`	Data-ready	14 byte burst oku
Tüm raw=`0x8000`	Geçersiz/off/reset/bus contention	PWR_MGMT0 ve ikinci master
Değerler byte-swap görünüyor	Endian hatası	INTF_CONFIG0 ve `be16`
Z ekseni yaklaşık 2 g	Yanlış FSR ölçeği	ACCEL_CONFIG0 read-back
Açı hızla sürükleniyor	Gyro bias/dt hatası	Bias kalibrasyonu ve timestamp
FIFO sürekli overflow	Host yavaş/watermark büyük	Burst boyutu, SPI yükü, watermark

20 Uygulama için önerilen sürücü katmanları

Kod büyüdüğünde tek `main.c` yerine şu katmanlar önerilir:

```
icm4268x_bus.c
  - SPI select/deselect
  - read/write/burst

icm4268x_regs.h
  - offsetler
  - bit maskeleri
  - enum FSR/ODR değerleri

icm4268x_core.c
  - probe/WHO_AM_I
  - reset/init
  - power, FSR, ODR
  - UI sample read

icm4268x_fifo.c
  - FIFO configure
  - packet parser
  - overflow recovery

icm4268x_irq.c
  - data-ready/FIFO/APEX routing

imu_calibration.c
  - bias/scale/mount matrix
```

Her register yazısından sonra körlemesine read-back yapmak da doğru değildir; W/C ve R/C registerlar farklı semantiğe sahiptir. Register metadata tablosunda erişim türü tutulması sürücü hatalarını azaltır.

21 Kısa kontrol listesi

```
[ ] WHO_AM_I dört okumada kararlı mı?
[ ] 0x47 cihazı ICM-42688-P olarak kabul ediliyor mu?
[ ] DEVICE_CONFIG.SPI_MODE reset değerinde mi?
[ ] REG_BANK_SEL Bank 0 mı?
[ ] PWR_MGMT0 read-back 0x0F mı?
[ ] GYRO_CONFIG0 ve ACCEL_CONFIG0 ölçekle uyumlu mu?
[ ] INT_STATUS.DATA_RDY bekleniyor mu?
[ ] 14 byte tek CS ile burst okunuyor mu?
[ ] 16 bit değerler signed ve doğru endian ile birleşiyor mu?
[ ] raw 0x8000 geçersiz olarak filtreleniyor mu?
[ ] SAMG55 aynı SPI bus'ı sürmüyor mu?
[ ] FIFO kullanılıyorsa overflow ve packet header kontrolü var mı?
[ ] Bias, sıcaklık ve montaj ekseni kalibrasyonu uygulanıyor mu?
```

22 Kaynaklar

1. TDK InvenSense, **ICM-42688-P Datasheet**, DS-000347.
   https://invensense.tdk.com/wp-content/uploads/2020/04/ds-000347_icm-42688-p-datasheet.pdf

2. TDK Product Center, **ICM-42688-P Detailed Information**.
   https://product.tdk.com/en/search/sensor/mortion-inertial/imu/info?part_no=ICM-42688-P

3. TDK InvenSense, **ICM-42686-P Datasheet**, DS-000348. https://invensense.tdk.com/wp-content/uploads/documentation/ds-000348__icm-42686-p-datasheet.pdf

4. TDK InvenSense, **SmartMotion Development Platform (Ver. G) Hardware User Guide**, AN-000300 Rev. 1.2. https://invensense.tdk.com/wp-content/uploads/2021/09/AN-000300-SmartMotion-Development-Platform-Ver.-G-Hardware-User-Guide-v1.2.pdf

5. Proje kaynakları: `soc/software/pmod_spi_test/main.c`, `soc/software/bsp/drivers/pmod_spi.c`, `hardware/soc/scala/fpga/VCU118Peripherals.scala` ve Genesys2 PMOD XDC dosyaları.

Doküman sürümü: 1.0 — Sahada okunan ICM-42688-P (`WHO_AM_I=0x47`) ve mevcut Genesys2/Rocket yazılımı temel alınmıştır.