Protocolo CAN

presented by

Andrea Z. Rubio Quezada
Fatima Alvarez Nuño
Gustavo A. Nuño Corvera

PROTOCOLO FDCAN

- El protocolo FDCAN es el utilizado por la tarjeta STM32.

¿Que es CAN/FDCAN?

- CAN (Controller Area Network) es un bus serial robusto usado en automotriz/industrial para comunicar nodos (mensajes cortos, prioridad por ID). Su meta es garantizar integridad de datos, prioridad de mensajes y tolerancia a fallos.

- FDCAN (Flexible Data-rate CAN) es la versión que soporta CAN FD (mayor data rate en la fase de datos) y trae mejoras en el controlador (más flexibilidad en RAM de mensajes, modos, filtros, etc.). (fuente: ST / presentación FDCAN y RM0399).

FDCAN en la STM32

La STM32H745 tiene:

- 2 controladores FDCAN: FDCAN1 y FDCAN2.
- Cada controlador se conecta internamente al bus FDCAN kernel, que comparte un Message RAM centralizado (en SRAM) configurado por software.

Uso:

- CAN clásico 2.0A/B → hasta 1 Mbit/s.
- CAN FD → hasta 8 Mbit/s en fase de datos (con Bit-Rate Switch = BRS).

3 Capas del CAN

Capa Fisica

Es la encargada de definir los niveles eléctricos (diferencial de CAN_H y CAN_L)

Capa de Objeto (LLC)

La que decide que mensajes enviar o aceptar (filtrado y gestión de mensajes)

Capa de Transferencia (MAC)

Es la encargada de gestionar el acceso al bus, la sincronización, detección de errores y retransmisión.

Señales Utilizadas

CAN_H

- Estado Dominante: ~3.5v
- Estado Recesivo: ~2.5V

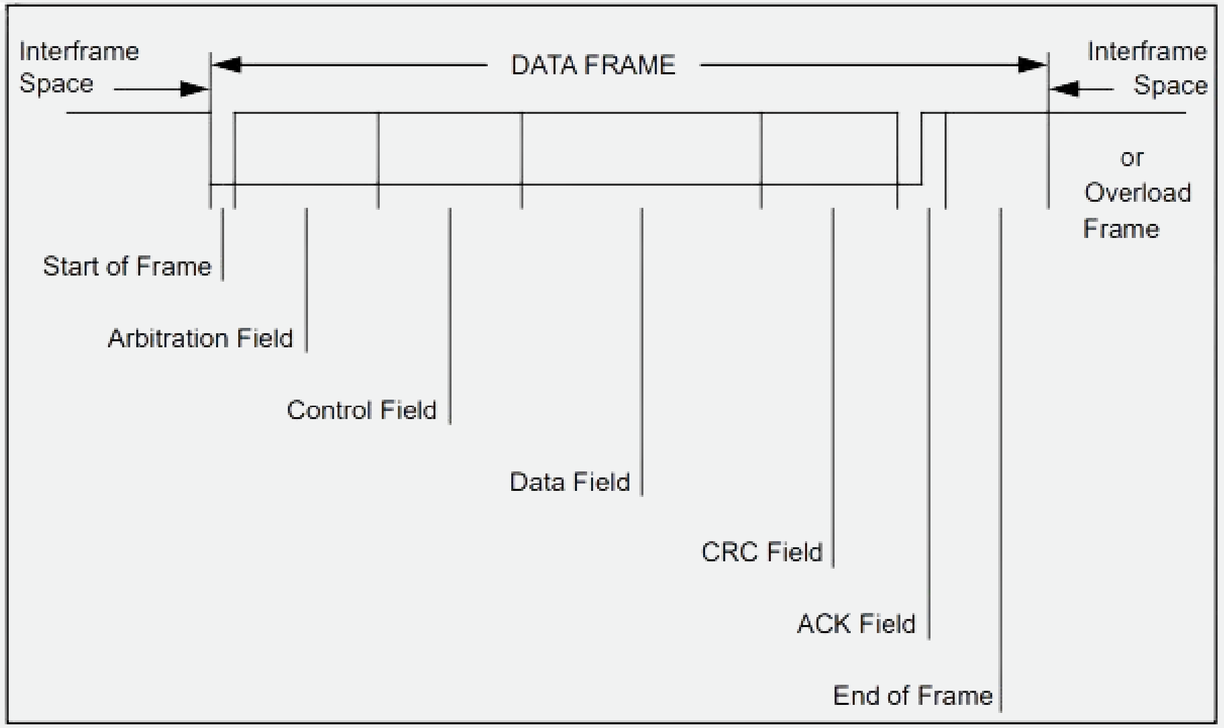
CAN_L

- Estado Dominante: ~1.5V
- Estado Recesivo: ~2.5V

- El dominante (0 Lógico) fuerza el bus; tiene prioridad.
- El recesivo (1 Lógico) liberado; puede ser sobreescrito por un dominante.
- Esto permite el arbitraje por prioridad y la detección simultánea de colisiones sin pérdida de información.

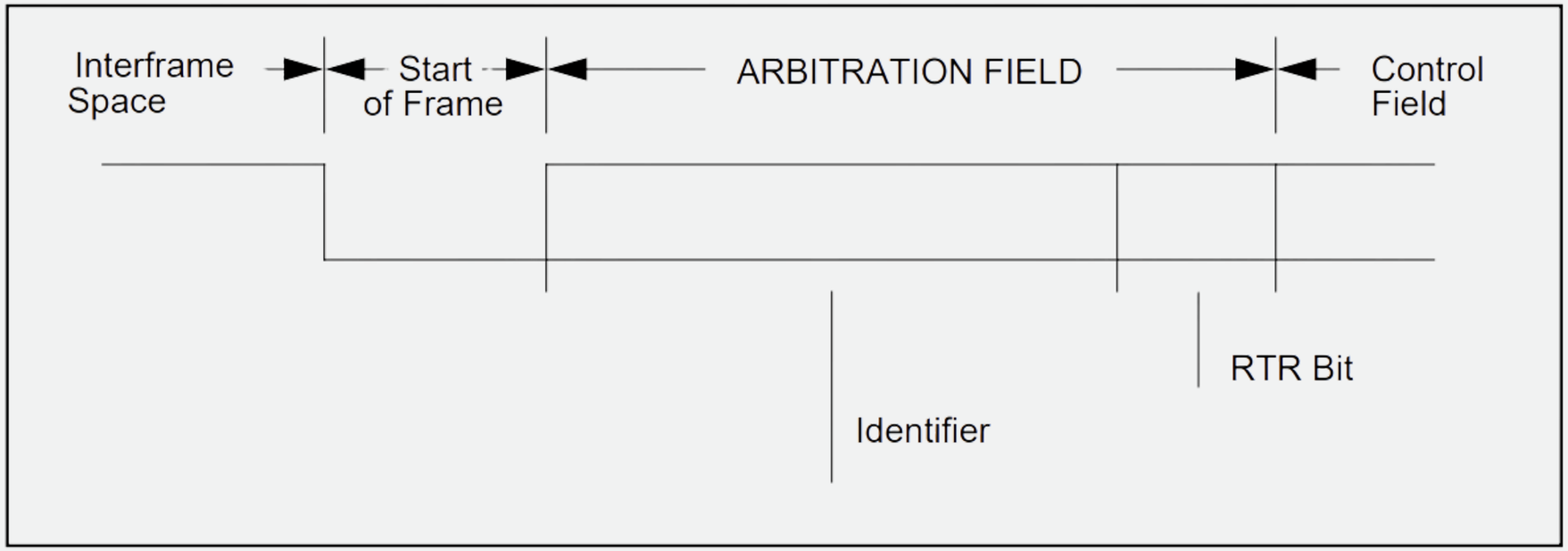
Fundamentos y Formatos de Trama CAN 2.0

- **Lógica del Bus:** El bus opera en dos estados: Dominante (0 lógico) y Recesivo (1 lógico). El estado Dominante siempre tiene prioridad (lógica wired-AND).
- Formatos del ID:
  - **CAN 2.0A (Estándar):** Utiliza un identificador de 11 bits.
  - **CAN 2.0B (Extendido):** Utiliza un identificador de 29 bits.

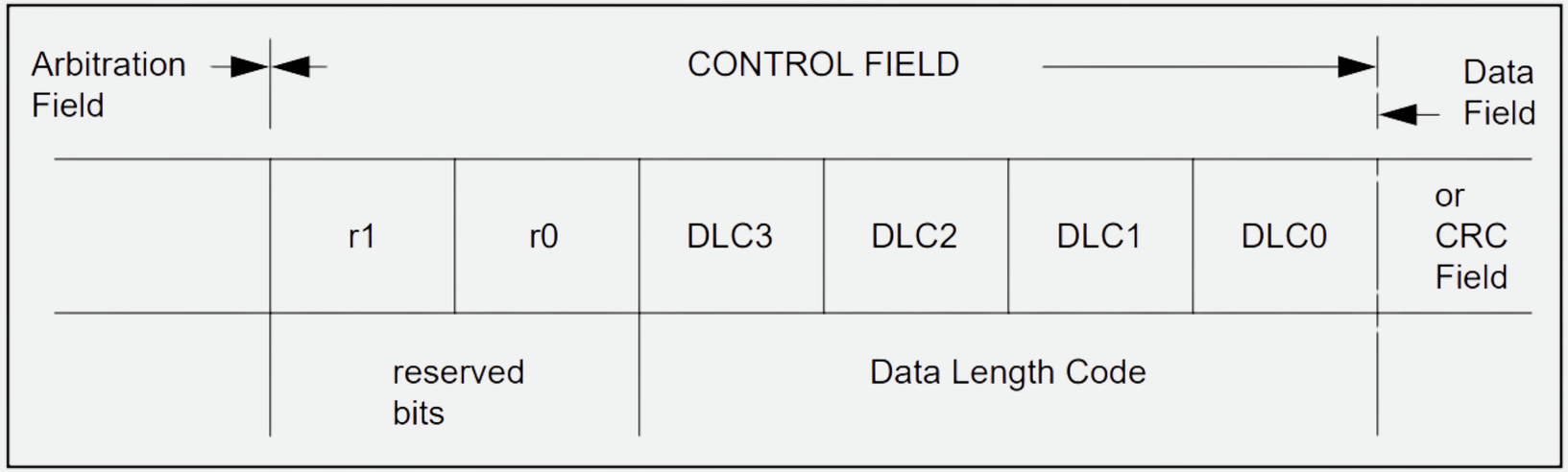


DATA FRAME

- **Start of Frame:** Un solo bit dominante que marca el inicio y sincroniza los nodos.
- **Arbitration Field:** Contiene el Identificador (11 o 29 bits) y el bit RTR.
  - **Identifier:** Campo de 11 bits que se transmite de ID-10 a ID-0. Es obligatorio que los 7 bits más altos (ID-10 a ID-4) no sean todos 'recesivos'.
  - **RTR Bit:** Indica el tipo de trama: es 'dominante' para las TRAMAS DE DATOS y 'recesivo' para las TRAMAS REMOTAS.



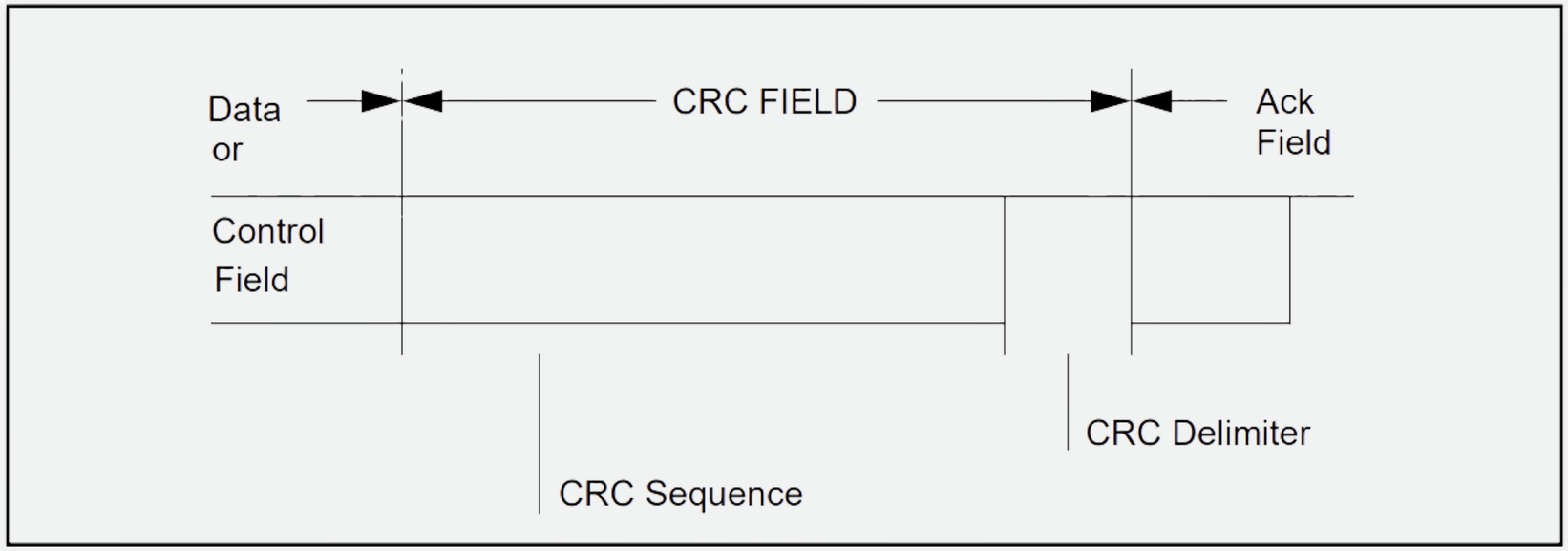
- **Control Field:** Tiene 6 bits. Contiene el Código de Longitud de Datos y dos bits reservados que deben ser 'dominantes'.
- **Data Field:** Son 4 bits que van dentro del Campo de Control. Su función es indicar la cantidad de bytes en el campo de datos, empieza en el más significativo (MSB).
    - **Data Lenght Code:** El DLC es un campo de 4 bits (dígitos binarios) que se traduce directamente al número de bytes en el campo de datos de una trama CAN (máximo 8 bytes).



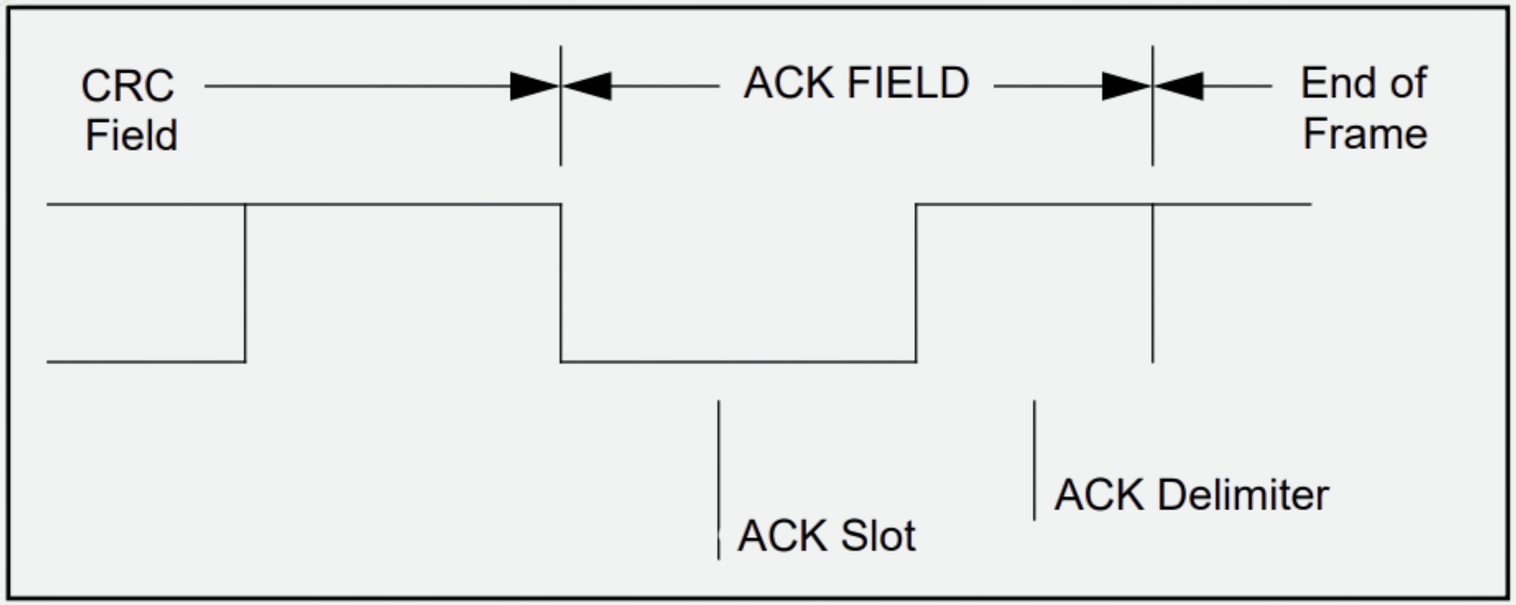
Number of Data Bytes	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

d → dominant
r → recessive

- **CRC Field:** Se compone de la SECUENCIA CRC (el valor de verificación) seguida de un DELIMITADOR CRC. Se utiliza para la detección de errores en la trama.
  - **CRC Sequence:** Es el valor de verificación (15 bits) que se calcula sobre la trama y queda listo en el registro CRC_RG al terminar el Campo de Datos.
  - **CRC Delimiter:** Es un único bit 'recesivo' (1) que se transmite inmediatamente después de la Secuencia CRC.

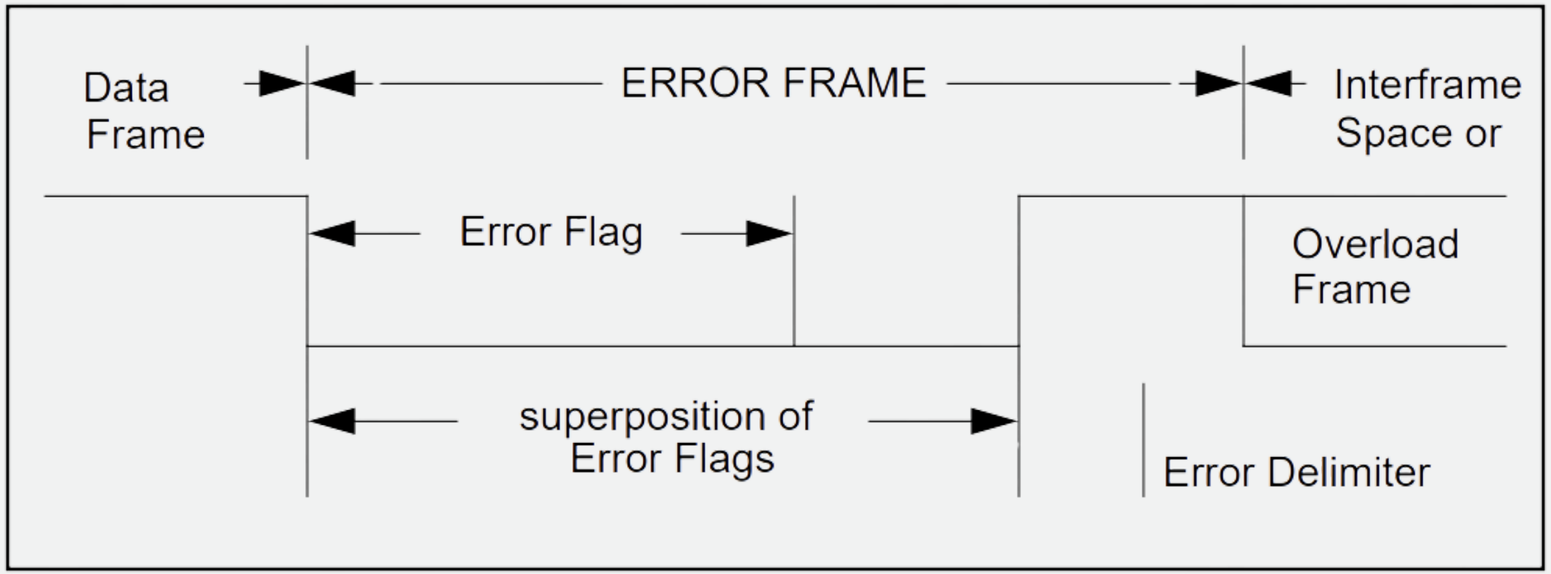


- **ACK Field:** 2 bits (Slot y Delimitador). El transmisor pone 'recesivos'. Los receptores sin error superponen un 'dominante' en el Slot para confirmar la recepción (ACK).
  - **ACK Slot:** El receptor que verificó el CRC superpone un bit 'dominante' sobre el bit recesivo del transmisor para acusar recibo.
  - **ACK Delimiter:** El segundo bit del Campo ACK; siempre es 'recesivo'.
- **End Frame:** Toda trama finaliza con una secuencia de siete bits 'recesivos'.



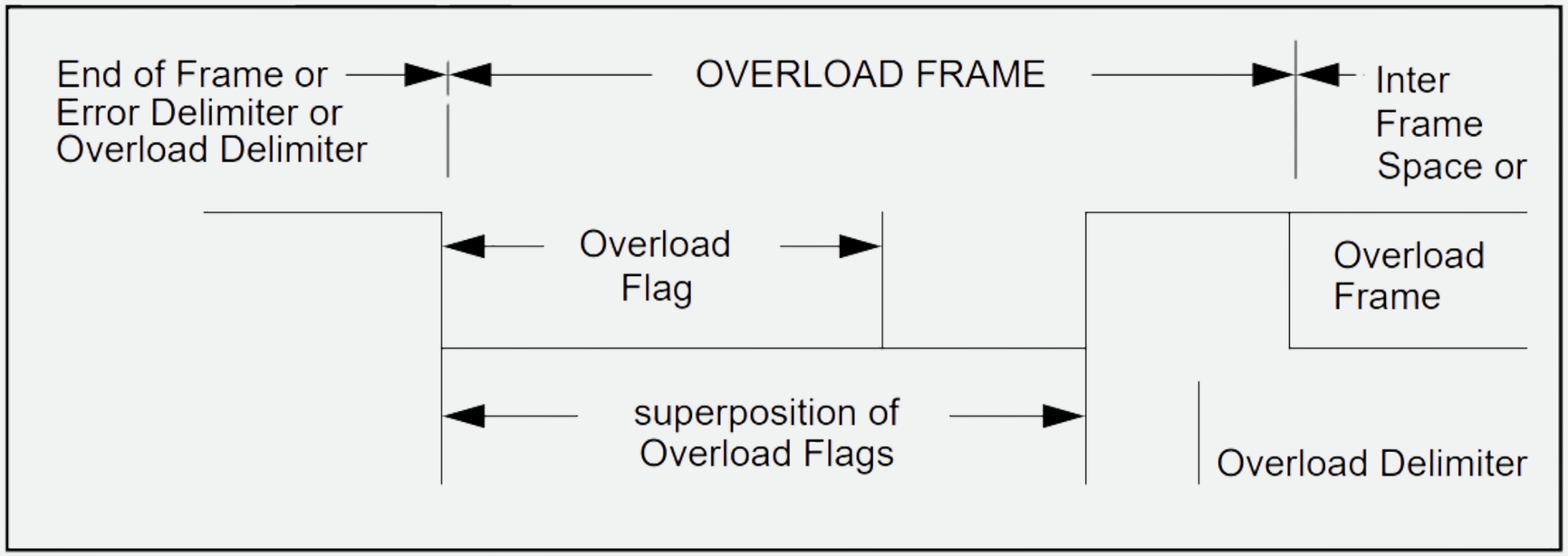
ERROR FRAME

- **Error Flag:** Existen dos formas de Bandera de Error: ACTIVA y PASIVA, ambas compuestas por seis bits:
  - **BANDERA DE ERROR ACTIVA (ACTIVE ERROR FLAG):** Consiste en seis bits consecutivos 'dominantes' (0).
  - **BANDERA DE ERROR PASIVA (PASSIVE ERROR FLAG):** Consiste en seis bits consecutivos 'recesivos' (1), a menos que otros nodos la sobrescriban con bits 'dominantes'.
- **Error Delimiter:** Consiste en ocho bits 'recesivos' (1). Lo usan las estaciones para terminar la señalización de error y restaurar el silencio en el bus.



OVERLOAD FRAME

- **Overload Flag:** Son seis bits 'dominantes' (0). Se usa para indicar sobrecarga y romper la regla de bit stuffing, obligando a los otros nodos a detener la transmisión.
- **Overload Delimiter:** Consiste en ocho bits 'recesivos' (1). Se usa para terminar la señalización de sobrecarga y restaurar la calma en el bus.



Arquitectura interna del periférico FDCAN

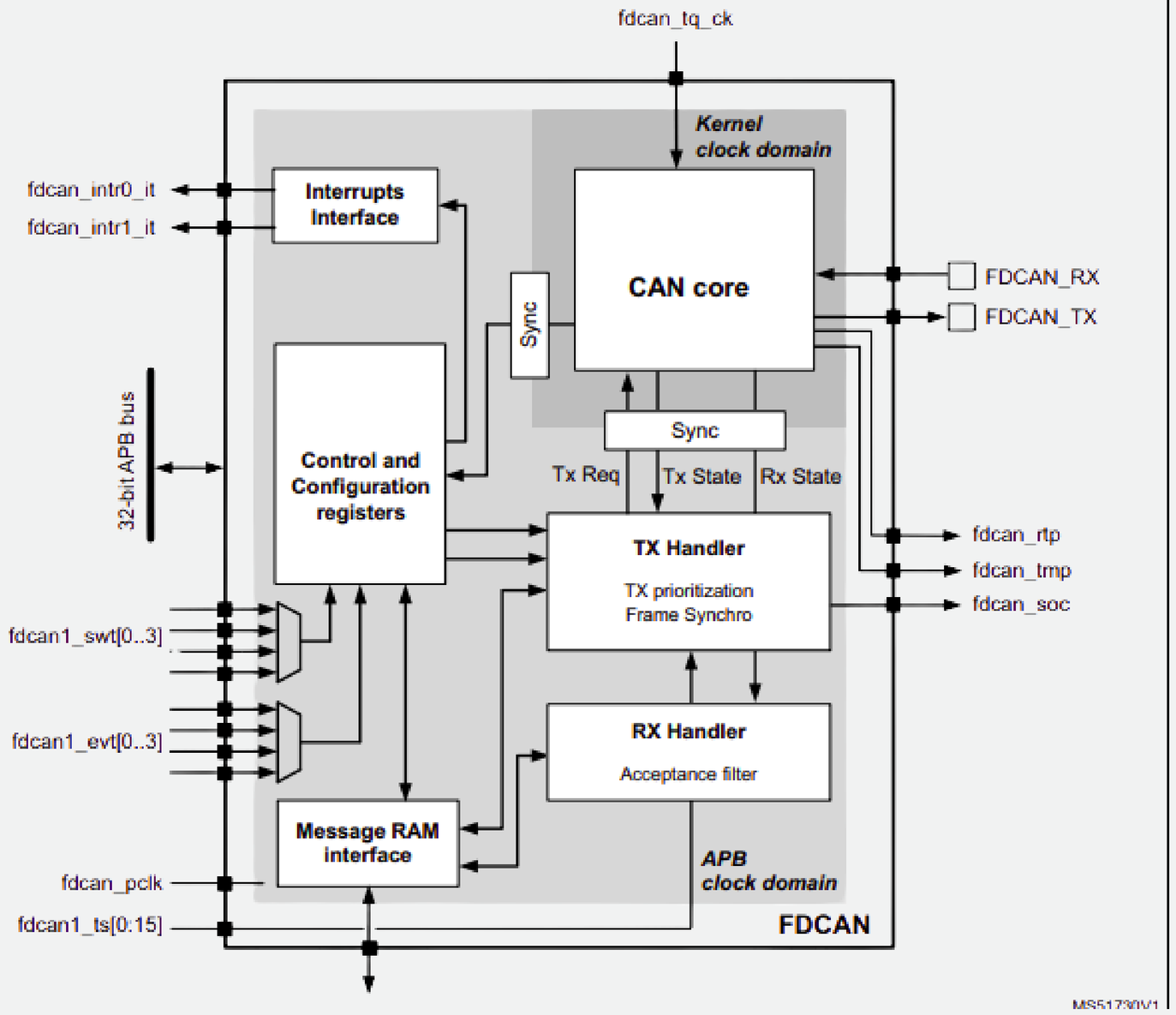


image retrieve fromhttps://www.st.com/resource/en/reference_manual/rm0399-stm32h745755-and-stm32h747757-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

CPU escribe → Message RAM → TX Handler → CAN Core → Bus (FDCAN_TX → transceiver)

Bus (FDCAN_RX ← transceiver) → CAN Core → RX Handler → Message RAM → Interrupción → CPU

ARQUITECTURA

Parte de la arquitectura	¿Que hace?	Analogia
Control & Config	Donde se programa el periferico	• Nuestro panel de control
CAN Core	Logica del protocolo CAN	• Motor principal
TX Handler	Organiza los mensajes que salen	• Cartero
RX Handler	Acepta los mensajes que llegan	• Portero
Message RAM	Guarda mensajes	Buzon
Interrupts Interface	Avisa eventos al CPU	Timbre

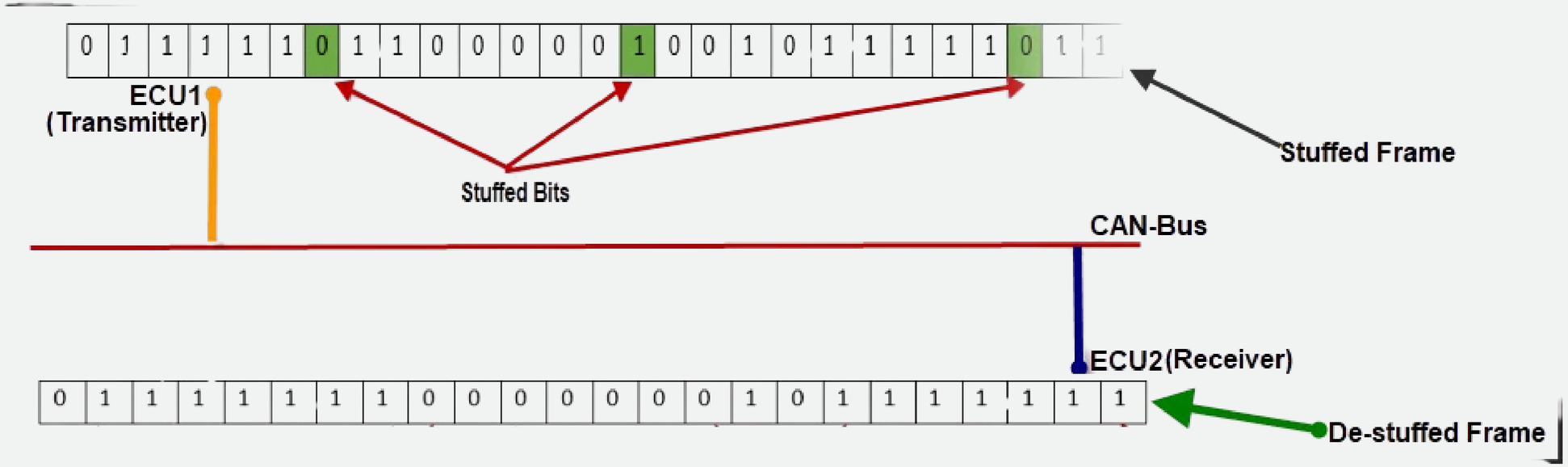
Robustez y Manejo de Errores

Mecanismos de Detección de Errores

1. **Bit Error**
   - Cada nodo que transmite un bit lo escucha simultáneamente en el bus. Si el valor en el bus es diferente al transmitido, se detecta un error.
2. **Stuff Error**
   - En CAN no pueden haber más de 5 bits iguales seguidos (para mantener sincronía).
3. **CRC Error (Cyclic Redundancy Check)**
   - Antes de enviar, el transmisor calcula un número especial (CRC) a partir de los bits del mensaje.
   - Los receptores hacen el mismo cálculo.
   - Si no coincide → hubo alteración de bits en el camino.
4. **ACK Error (Acknowledgment)**
   - Cuando un receptor recibe un mensaje correcto, debe mandar un “0” (dominante) en el campo de ACK.
   - Si el transmisor no ve ese “0”, significa que nadie lo entendió → retransmite el mensaje.

Stuff Error - Bit Stuffing:

- En CAN no puede haber más de 5 bits iguales seguidos.
- Cuando ya se han transmitido cinco bits del mismo valor, el transmisor mismo mete automáticamente un bit opuesto (sin que tú lo programes).
- Ese bit extra se llama bit stuffed.

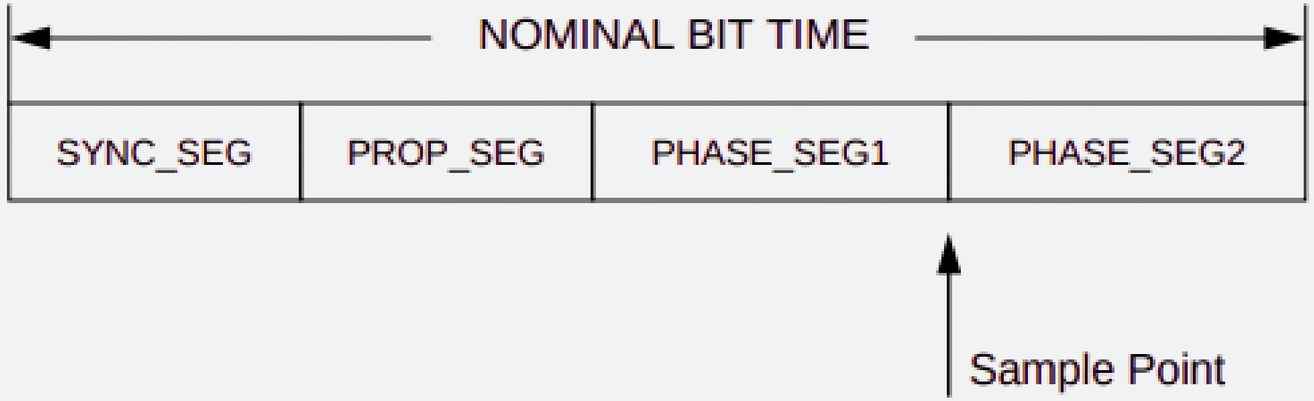


Registros de FDCAN

REGISTRO	DESCRIPCIÓN
CCCR	Control (activar INIT/CCE, modos ASM, MON, CSR).
NBTP	Temporización nominal (prescaler NBRP, TSEG1, TSEG2, SJW).
DBTP	Temporización de fase de datos para CAN FD.
TEST	Loopback, monitor interno, control Tx.
SIDFC/XIDFC	Filtros de identificadores (11 b / 29 b).
RXF0C/RXF1C	Configuración de las colas FIFO de recepción.
TXBC	Número y modo de buffers/cola Tx.
IE / ILS / ILE	Interrupciones (status, enable, line).
FDCAN_NDAT1/2	Estado de mensajes recibidos

Tiempo del bit

Cada bit transmitido en CAN se divide en segmentos temporales para mantener sincronizados todos los nodos:



El Sample point esta al final de PHASE_SEG1

El tiempo total de un bit se expresa como:

$$T_{bit} = (SYNCSEG + PROPSEG + PHASESEG1 + PHASESEG2) \times TQ$$

donde:

$$TQ = \frac{1}{fosc/Prescaler}$$

Esto se configura en el periférico mediante los registros NBTP (Nominal Bit Timing & Prescaler) y DBTP (Data Bit Timing & Prescaler)

Código y Demostración

Main Code CAN