CAN Signal Packing & Unpacking

- CAN bus transmits **data in bytes (8 bits)**
- Signals (like speed, gear) may be **smaller than a byte** or **combined in one byte**
- **Packing:** putting signal values into CAN bytes for transmission.
- **Unpacking:** extracting signal values from received CAN bytes.

❑ Packing (Sender Side)

- Convert signal to binary.
- Mask unwanted bits using & (example: & 0xFF for 1 byte).
- Shift bits to correct position using <<
- Combine multiple signals in one byte using |
- Store final value in CAN data array,

Example :

```
uint8_t can_data[8];
can_data[0] = speed & 0xFF;          // lower byte
can_data[1] = (speed >> 8) & 0xFF;   // upper byte
```

- **For two signals in one byte**:

```
can_data[2] = ((speed & 0x0F) << 4) | (gear & 0x0F);
```

- **Key points:**
  - Mask = keep only relevant bits
  - Shift = place bits in proper position
  - Combine = | operator for multiple signals

❑ Unpacking (Receiver Side)

- Performed on receiver side after CAN frame is received
- Extract original signal values from CAN data bytes
- Reverse operation of packing
- Uses:
  - Bit shifting
  - Bit masking
- Output signals are used for SOA services
- Applications / debugging / logging

Example :

```
uint16_t speed;

speed = can_data[0] | (can_data[1] << 8);
```

- **Unpacking Steps**
- Read CAN data bytes
- Shift bits to original position
- Mask required bits
- Combine bits to reconstruct signal