Communication Driver Explanation

This communication driver module is responsible for managing the packing of signals into Protocol Data Units (PDUs) and transmitting these messages over different communication buses such as CAN, SPI, or UART. It provides flexible support for mapping signals to messages and handles bit-level manipulation required to correctly position signals within message buffers, supporting both little-endian and big-endian formats.

The driver facilitates periodic message transmission by integrating with software timers, ensuring messages are sent at configured intervals automatically. This design allows for modular and scalable communication management across various bus types by abstracting bus-specific transmission logic, making it straightforward to switch or add new buses without extensive code changes.

It is important to note that the reception feature is not implemented in this module, as it is not required by the current project scope.

A key aspect of the module is its configuration-based approach. Communication parameters, including those derived from CAN DBC files, are directly transferred to configuration files located under the `Project_Config` directory. This enables seamless adaptation of new changes without modifying the communication driver code itself, increasing flexibility and maintainability.

Overall, this module simplifies communication handling by providing APIs to initialize the driver, send signals by updating message buffers, trigger immediate message transmission, and handle periodic sending based on timer callbacks. It is designed to ensure reliable, efficient, and configurable communication in embedded systems.

```
static const com_signal_t m_com_signals_of_system_info_message[] =
{
    {
        .signal_id = COM_SYSTEM_TEMPERATURE_SIGNAL_ID,
        .signal_base_info =
        {
            .com_bit_position = 0,
            .com_bit_size = 16U,
            .com_signal_endianness = COM_SIGNAL_ENDIANNESS_LITTLE_ENDIAN_e,
            .com_signal_variable_type = COM_SIGNAL_VARIABLE_UINT16_e,
        },

    },
    {
        .signal_id = COM_SYSTEM_STATE_SIGNAL_ID,
        .signal_base_info =
        {
            .com_bit_position = 16U,
            .com_bit_size = 3U,
            .com_signal_endianness = COM_SIGNAL_ENDIANNESS_LITTLE_ENDIAN_e,
            .com_signal_variable_type = COM_SIGNAL_VARIABLE_UINT8_e,
        },

    },
    {
        .signal_id = COM_SYSTEM_ERROR_STATE_SIGNAL_ID,
        .signal_base_info =
        {
            .com_bit_position = 19U,
            .com_bit_size = 4U,
            .com_signal_endianness = COM_SIGNAL_ENDIANNESS_LITTLE_ENDIAN_e,
            .com_signal_variable_type = COM_SIGNAL_VARIABLE_UINT8_e,
        },

    },
};
```

As shown in the figure above, signal properties such as bit position, bit length, and endianness—defined in the DBC file—are specified in the configuration file (`com_driver_cfg.c`) and provided to the module during initialization for configuration.