SPD1179 LIN LDF 文件使用指南

版本 A/0 – 2023 年 4 月

概述

这份文档描述了 LDF 文件和 C 配置文件之间的关系，并说明了如何根据 LDF 文档更改 C 语言配置文件。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
A/0	2023 年 5 月 14 日	CanChai	Released	首次发布。

术语或缩写

术语或缩写	描述

1 LDF 描述文件

LDF 文件的格式如下：

```
// Global Definitions
LIN_description_file;
LIN_protocol_version = "2.1"; // only support LIN2.1
LIN_language_version = "2.1";// only support LIN2.1
LIN_speed = 19.2 kbps; // baudrate (1.0-20.0)
// End of Global Definitions

Nodes // Nodes Definition
{
    Master:
        LINMaster,    // master node name
        1 ms, // time base (real)
        0.1 ms;       // jitter (real)

    Slaves:
        FrontLeftDoor; // slave node names

} // End of Nodes Definition

Signals // Signals Definition
{
    FrontLeftDoorSignal:    8,    0,    FrontLeftDoor,    LINMaster; //signal name, signal size(bit), signal initial data, originator, recipient
    FrontLeftDoorErrSig:    1,    0,    FrontLeftDoor,    LINMaster;
    //Master to slave signal
    MasterToFront:          8,    0,    LINMaster,        FrontLeftDoor;
    //Slave to master signal
    FrontToMasterSignal:          8, 0, FrontLeftDoor, LINMaster;
    FrontToMasterEventSignal:          8, 0, FrontLeftDoor, LINMaster;
} // End of Signals Definition
```

```
Diagnostic_signals // Diagnostic Signals Definition (optional)
{
    // MasterReq Reserved Signals
    MasterReqB0: 8, 0; // signal name: signal size, signal offset;
    MasterReqB1: 8, 0;
    MasterReqB2: 8, 0;
    MasterReqB3: 8, 0;
    MasterReqB4: 8, 0;
    MasterReqB5: 8, 0;
    MasterReqB6: 8, 0;
    MasterReqB7: 8, 0;
        // End of MasterReq Reserved Signals

    // SlaveResp Reserved Signals
    SlaveRespB0: 8, 0; // signal name: signal size, signal offset;
    SlaveRespB1: 8, 0;
    SlaveRespB2: 8, 0;
    SlaveRespB3: 8, 0;
    SlaveRespB4: 8, 0;
    SlaveRespB5: 8, 0;
    SlaveRespB6: 8, 0;
    SlaveRespB7: 8, 0;
    // End of SlaveResp Reserved Signals

} // End of Diagnostic Signals Definition

Frames // Unconditional Frames Definition
{
    FrontLeftDoorMessage: 1, FrontLeftDoor, 1 //frame name, frame ID(decimal), originator, frame length(byte)
    {
        FrontLeftDoorSignal, 0; // signal name, signal offset in frame(bit)
    }
  //error frame will use eventriger frame
```

```
        FrontLeftDoorErrEvent: 3, FrontLeftDoor, 1
        {
            FrontLeftDoorErrSig,    0;
        }

        MasterToFrontControl: 6, LINMaster, 2{
            MasterToFront, 0;
        }

        FrontToMasterMessage: 8, FrontLeftDoor, 1{
            FrontToMasterSignal, 0;
        }
        FrontToMasterEventMessage: 4, FrontLeftDoor, 2{
            FrontToMasterEventSignal, 8; // signal name, signal offset in frame(bit): because this frame is relevant to event triggered frame, offset must >= 8
        }
    } // End of Unconditional Frames Definition

    Event_triggered_frames {
        ETF_MotorStates: ETFCollisionResolving, 11, FrontToMasterEventMessage; // frame name, conflict resolution schedule table, frame ID(decimal), relevant unconditional frame
    }

    Diagnostic_frames // Diagnostic Frame Definition (optional)
    {
        MasterReq: 60 // reserved frame name and frame identifier
        {
            MasterReqB0, 0; // diagnostic signal name, signal offset
            MasterReqB1, 8;
            MasterReqB2, 16;
            MasterReqB3, 24;
            MasterReqB4, 32;
            MasterReqB5, 40;
            MasterReqB6, 48;
            MasterReqB7, 56;
```

```
        }

        SlaveResp: 61 // reserved frame name and frame identifier
        {
            SlaveRespB0, 0; // diagnostic signal name, signal offset
            SlaveRespB1, 8;
            SlaveRespB2, 16;
            SlaveRespB3, 24;
            SlaveRespB4, 32;
            SlaveRespB5, 40;
            SlaveRespB6, 48;
            SlaveRespB7, 56;
        }
    } // End of Diagnostic Frame Definition

    Node_attributes // Node Attributes Definition
    {
      FrontLeftDoor // node name
        {
            LIN_protocol = "2.1";      // LIN version, only support 2.1
            configured_NAD = 0x11;    // initial NAD of node (1-126)
            initial_NAD = 0x11 ;
            product_id =
                0x0021,                    // supplier_id (0-0x7FFE)    <--> 0-0xFFDE
                1,                         // function_id (0-0xFFFE)
                0;                         // variant (0-0xFF)
            response_error = FrontLeftDoorErrSig;    // name of 1 bit error signal, now this signal is in FrontLeftDoorErrEvent
            P2_min = 2 ms;              // min. time (real) between MasterReq and SlaveResp
            ST_min = 1 ms;               // min. time (real) between consecutive SlaveResps
            N_As_timeout = 1000 ms;
            N_Cr_timeout = 1000 ms;    // Set the transport layer timeout parameter
            configurable_frames
            {
```

```
                // list of configurable frames
                FrontLeftDoorMessage; // frame name with message id (0-0xFFFE), all unconditional frames and event triggered frames
                MasterToFrontControl;
                FrontLeftDoorErrEvent;
                FrontToMasterMessage;
                FrontToMasterEventMessage;
                ETF_MotorStates;
            }
        } // End of Node Attribut
    } // End of Node Attributes Definition

    Schedule_tables {
        MasterReq_Table {
            MasterReq delay 20 ms ;
        }
        SlaveResp_Table {
            SlaveResp delay 20 ms ;
        }
        Diagnostic {
            MasterReq delay 20 ms ;
            SlaveResp delay 20 ms ;
        }
        ETFCollisionResolving {
            FrontToMasterEventMessage delay 50 ms ;
        }
        InitTable {
            AssignNAD { FrontLeftDoor } delay 20 ms ; // set NAD of the node
            SlaveResp delay 20 ms ;
            AssignFrameIdRange { FrontLeftDoor, 0 } delay 20 ms ; // set the ID from the first frame
        }
        NormalTable {
            FrontLeftDoorMessage    delay 10 ms;
            FrontToMasterMessage    delay 10 ms;
```

```
        ETF_MotorStates          delay 20 ms;
        MasterToFrontControl     delay 10 ms;
    }
}
```

2 LDF 与 C 配置文件

当 LDF 文档文件发生更改时，必须相应地更改 C 配置文件（LIN_Tester.c、LIN_Tester.h）。这是因为 LDF 文件定义了通信矩阵，包括节点、信号等信息。

表 2-1：LDF 与 C 配置文件之间的关系

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
LIN_speed = 19.2 kbps	#define LIN_BAUD_RATE 19200 /*For slave*/
configured_NAD = 0x11; initial_NAD = 0x11 ;	l_u8 lin_configured_NAD = 0x11; /*<configured_NAD>*/ const l_u8 lin_initial_NAD =0x11; /*<initial_NAD>*/
product_id = 0x0021, 1, 0;	const lin_product_id product_id = {0x0021, 0x0001, 0x0000 };
N_As_timeout = 1000 ms; N_Cr_timeout = 1000 ms;	#define LIN_N_AS_AND_N_CR_TIMEOUT 1000
configurable_frames { FrontLeftDoorMessage; MasterToFrontControl; FrontLeftDoorErrEvent; FrontToMasterMessage; FrontToMasterEventMessage; ETF_MotorStates; } Frames { FrontLeftDoorMessage: 1, FrontLeftDoor, 1 //frame name, frame ID(decimal), originator, frame length(byte) { FrontLeftDoorSignal, 0; }	**// The order must be cooperate with configurable_frames** **#define LIN_CFG_FRAME_NUM 6** **#define LIN_SIZE_OF_CFG (LIN_CFG_FRAME_NUM + 4)** l_u8 lin_configuration_RAM[LIN_SIZE_OF_CFG]= {0x00, 0x01, 0x06, 0x03, 0x08, 0x04, 0x0B, 0x3C, 0x3D ,0xFF}; const l_u16 lin_configuration_ROM[LIN_SIZE_OF_CFG]= {0x00, 0x01, 0x06, 0x03, 0x08, 0x04, 0x0B, 0x3C, 0x3D ,0xFFFF}; **// The order must be cooperate with configurable_frames** /* Number of frames */ #define LIN_NUM_OF_FRMS (**LIN_CFG_FRAME_NUM + 2**) /* List of frames */ typedef enum { /* All frames for master node */ /* Interface_name = LI0 */ LI0_FrontLeftDoorMessage

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
FrontLeftDoorErrEvent: 3, FrontLeftDoor, 1 { FrontLeftDoorErrSig, 0; } MasterToFrontControl: 6, LINMaster, 2{ MasterToFront, 0; } FrontToMasterMessage: 8, FrontLeftDoor, 1{ FrontToMasterSignal, 0; } FrontToMasterEventMessage: 4, FrontLeftDoor, 2{ FrontToMasterEventSignal, 8; } } Event_triggered_frames { ETF_MotorStates: ETFCollisionResolving, 11, FrontToMasterEventMessage; } Signals // Signals Definition { FrontLeftDoorSignal: 8, 0, FrontLeftDoor, LINMaster; //signal name, signal size(bit), signal initial data, originator, recipient FrontLeftDoorErrSig: 1, 0, FrontLeftDoor, LINMaster; //Master to slave signal MasterToFront: 8, 0, LINMaster, FrontLeftDoor;	, LI0_MasterToFrontControl , LI0_FrontLeftDoorErrEvent , LI0_FrontToMasterMessage , LI0_FrontToMasterEventMessage , LI0_ETF_MotorStates , LI0_MasterReq , LI0_SlaveResp } l_frame_handle; const l_u8 LI0_ETF_MotorStates_info_data = LI0_FrontToMasterEventMessage ; /* frame data */ **// The order must be cooperate with configurable frames** const lin_frame_struct lin_frame_tbl[LIN_NUM_OF_FRMS] ={ { LIN_FRM_UNCD, 1, LIN_RES_PUB, 0, 0, 1 , (l_u8*)0 } // FrontLeftDoorMessage: frame_length 1, frame_bias 0, flag_ bias 0, flag_size 1 ,{ LIN_FRM_UNCD, 2, LIN_RES_SUB, 1, 1, 1 , (l_u8*)0 } // MasterToFrontControl: frame_length 2, frame_bias LastFrame.frame_bias + LastFrame.frame_length , flag_bias LastFrame.flag_bias + LastFrame.flag_size, flag_size 1 ,{ LIN_FRM_UNCD, 1, LIN_RES_PUB, 3, 2, 1 , (l_u8*)&LI0_response_error_signal } // FrontLeftDoorErrEvent: frame_length 1, frame_bias LastFrame.frame_bias + LastFrame.frame_length , flag_bias LastFrame.flag_bias + LastFrame.flag_size, flag_size 1 ,{ LIN_FRM_UNCD, 1, LIN_RES_PUB, 4, 3, 1 , (l_u8*)0 } //…… ,{ LIN_FRM_UNCD, 2, LIN_RES_PUB, 5, 4, 1 , (l_u8*)0 } ,{ LIN_FRM_EVNT, 2, LIN_RES_PUB, 0, 0, 0 , (l_u8*)&LI0_ETF_MotorStates_info_data }

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
//Slave to master signal FrontToMasterSignal: 8, 0, FrontLeftDoor, LINMaster; FrontToMasterEventSignal: 8, 0, FrontLeftDoor, LINMaster; } // End of Signals Definition	,{ LIN_FRM_DIAG, 8, LIN_RES_SUB, 0, 0, 0 , (l_u8*)0 } ,{ LIN_FRM_DIAG, 8, LIN_RES_PUB, 0, 0, 0 , (l_u8*)0 } }; **// The order must be cooperate with configurable_frames** **#define LIN_FRAME_BUF_SIZE 7** l_u8 lin_pFrameBuf[LIN_FRAME_BUF_SIZE] = { 0x00 /* 0 : 00000000 */ /* start of frame LI0_FrontLeftDoorMessage */ // FrontLeftDoorMessage: frame_length 1 byte, FrontLeftDoorSignal.offset 0 bit, FrontLeftDoorSignal.length 8 bit, FrontLeftDoorSignal.initial_data 0 ,0x00 /* 1 : 00000000 */ /* start of frame LI0_MasterToFrontControl */ ,0xff /* 2 : 11111111 */ // MasterToFrontControl: frame_length 2 byte, MasterToFront.offset 0 bit, MasterToFront.length 8 bit, MasterToFront.initial_data 0 ,0xfe /* 3 : 11111110 */ /* start of frame LI0_FrontLeftDoorErrEvent */ //…… ,0x00 /* 4 : 00000000 */ /* start of frame LI0_FrontToMasterMessage */ ,0xc4 /* 5 : 11000100 */ /* start of frame LI0_FrontToMasterEventMessage */ ,0x00 /* 6 : 00000000 */ // FrontToMasterEventMessage: frame_length 2 byte, FrontToMasterEventSignal.offset 8 bit, FrontToMasterEventSignal.length 8 bit, FrontToMasterEventSignal.initial_data 0

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
	}; **#define LIN_FLAG_BUF_SIZE 5** **// The order must be cooperate with configurable_frames, default vaule is 0xFF** /* definition and initialization of signal array */ l_u8 lin_flag_handle_tbl[LIN_FLAG_BUF_SIZE] = { 0xFF /* 0: start of flag frame LI0_FrontLeftDoorMessage */ ,0xFF /* 1: start of flag frame LI0_MasterToFrontControl */ ,0xFF /* 2: start of flag frame LI0_FrontLeftDoorErrEvent */ ,0xFF /* 3: start of flag frame LI0_FrontToMasterMessage */ ,0xFF /* 4: start of flag frame LI0_FrontToMasterEventMessage */ }; **// The order must be cooperate with Signals** /* Number of signals */ #define LIN_NUM_OF_SIGS 5 /* List of signals */ typedef enum { /* Interface_name = LI0 */ LI0_FrontLeftDoorSignal , LI0_FrontLeftDoorErrSig , LI0_MasterToFront , LI0_FrontToMasterSignal , LI0_FrontToMasterEventSignal } l_signal_handle;

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
response_error = FrontLeftDoorErrSig FrontLeftDoorErrSig: 1, 0, FrontLeftDoor, LINMaster; // Signal size 1 bit, initial data 0 FrontLeftDoorErrEvent: 3, FrontLeftDoor, 1 { FrontLeftDoorErrSig, 0; // Signal bias 0 }	const l_signal_handle LI0_response_error_signal = LI0_FrontLeftDoorErrSig; const l_signal_handle response_error = LI0_FrontLeftDoorErrSig; const l_u8 num_frame_have_esignal = 1; /*number of frame contain error signal*/ #define LIN_BYTE_OFFSET_LI0_FrontLeftDoorErrSig 3U const l_u16 lin_response_error_byte_offset[1] = {LIN_BYTE_OFFSET_LI0_FrontLeftDoorErrSig}; /*<interface_name>_< response_error>*/ #define LIN_BIT_OFFSET_LI0_FrontLeftDoorErrSig 0U const l_u8 lin_response_error_bit_offset[1] = {LIN_BIT_OFFSET_LI0_FrontLeftDoorErrSig}; /*<interface_name>_< response_error>*/