

SPD1179 LDF Document Usage Guide

Revision A/0 – December 2022

Overview

This document describes the relationship between the LDF file and c configuration file and how to change the c language configuration file according to LDF Document.

SPINTROL

Contents

1	LDF description	7
2	LDF and c configuration	13

SPINTROL

List of figures

未找到图形项目表。

SPIN TROL

List of Tables

Table 2-1: Relationship between LDF and c configurations13

SPINTROL

Revision history

Revision	Date	Author	Changes
A/0	25-Dec-2022	Hang Su	Initial release.

SPIN TROL

Terms or abbreviations

Terms or abbreviations	Description

SPINTROL

1 LDF description

The format of LDF file is showed as following:

```
// 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 eventrigger 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;
    }
}
```

}

SPINTROL

2 LDF and c configuration

When the LDF document file is changed, the c configurations(LIN_Tester.c, LIN_Tester.h) must be changed too.

Table 2-1: Relationship between LDF and c configurations

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 ;	_u8 lin_configured_NAD = 0x11; /*<configured_NAD>*/ const _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) _u8 lin_configuration_RAM[LIN_SIZE_OF_CFG]= {0x00, 0x01, 0x06, 0x03, 0x08, 0x04, 0x0B, 0x3C, 0x3D ,0xFF}; const _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 = LIO */ LIO_FrontLeftDoorMessage

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
<pre> 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; </pre>	<pre> , LIO_MasterToFrontControl , LIO_FrontLeftDoorErrEvent , LIO_FrontToMasterMessage , LIO_FrontToMasterEventMessage , LIO ETF_MotorStates , LIO_MasterReq , LIO_SlaveResp }l_frame_handle; const l_u8 LIO ETF_MotorStates_info_data = LIO_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*)&LIO_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*)&LIO ETF_MotorStates_info_data } </pre>

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
<pre> //Slave to master signal FrontToMasterSignal: 0, FrontLeftDoor, LINMaster; FrontToMasterEventSignal: 8, 0, FrontLeftDoor, LINMaster; } // End of Signals Definition </pre>	<pre> ,{ 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 LIO_FrontLeftDoorMessage */ // FrontLeftDoorMessage: frame_length 1 byte, FrontLeftDoorSignal.offset 0 bit, FrontLeftDoorSignal.length 8 bit, FrontLeftDoorSignal.initial_data 0 ,0x00 /* 1 : 00000000 */ /* start of frame LIO_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 LIO_FrontLeftDoorErrEvent */ //..... ,0x00 /* 4 : 00000000 */ /* start of frame LIO_FrontToMasterMessage */ ,0xc4 /* 5 : 11000100 */ /* start of frame LIO_FrontToMasterEventMessage */ ,0x00 /* 6 : 00000000 */ // FrontToMasterEventMessage: frame_length 2 byte, FrontToMasterEventSignal.offset 8 bit, FrontToMasterEventSignal.length 8 bit, FrontToMasterEventSignal.initial_data 0 </pre>

LDF	C configuration(LIN_Tester.c, LIN_Tester.h)
	<pre> }; #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 LIO_FrontLeftDoorMessage */ ,0xFF /* 1: start of flag frame LIO_MasterToFrontControl */ ,0xFF /* 2: start of flag frame LIO_FrontLeftDoorErrEvent */ ,0xFF /* 3: start of flag frame LIO_FrontToMasterMessage */ ,0xFF /* 4: start of flag frame LIO_FrontToMasterEventMessage */ }; // The order must be cooperate with Signals /* Number of signals */ #define LIN_NUM_OF_SIGS 5 /* List of signals */ typedef enum { /* Interface_name = LIO */ LIO_FrontLeftDoorSignal , LIO_FrontLeftDoorErrSig , LIO_MasterToFront , LIO_FrontToMasterSignal , LIO_FrontToMasterEventSignal } l_signal_handle; </pre>

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