SWC_IR

Version v1.0
7/22/2023 1:40:00 AM

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

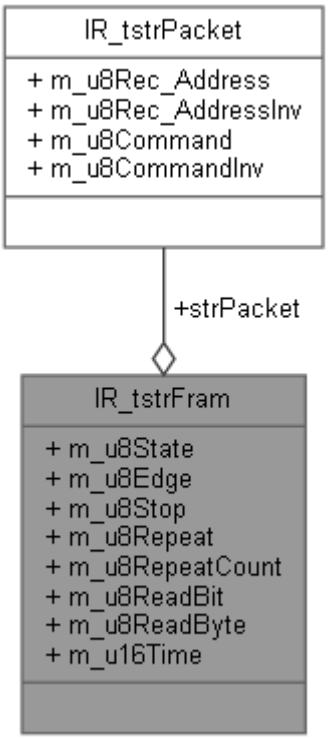
Here is a list of all files with brief descriptions:

Data Structure Documentation

IR_tstrFram Struct Reference

`#include <IR_priv.h>`

Collaboration diagram for IR_tstrFram:



Data Fields

- IR_tstrPacket strPacket
- volatile u8 m_u8State
- volatile u8 m_u8Edge
- volatile u8 m_u8Stop
- volatile u8 m_u8Repeat
- volatile u8 m_u8RepeatCount
- volatile u8 m_u8ReadBit
- volatile u8 m_u8ReadByte
- volatile u16 m_u16Time

Field Documentation

volatile u16 m_u16Time

volatile u8 m_u8Edge

volatile u8 m_u8ReadBit

volatile u8 m_u8ReadByte

volatile u8 m_u8Repeat

volatile u8 m_u8RepeatCount

volatile u8 m_u8State

volatile u8 m_u8Stop

IR_tstrPacket strPacket

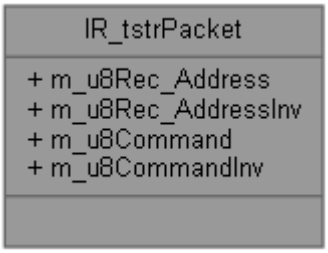
The documentation for this struct was generated from the following file:

IR_priv.h

IR_tstrPacket Struct Reference

`#include <IR_int.h>`

Collaboration diagram for IR_tstrPacket:



Data Fields

- volatile u8 m_u8Rec_Address
- volatile u8 m_u8Rec_AddressInv
- volatile u8 m_u8Command
- volatile u8 m_u8CommandInv

Field Documentation

volatile u8 m_u8Command

volatile u8 m_u8CommandInv

volatile u8 m_u8Rec_Address

volatile u8 m_u8Rec_AddressInv

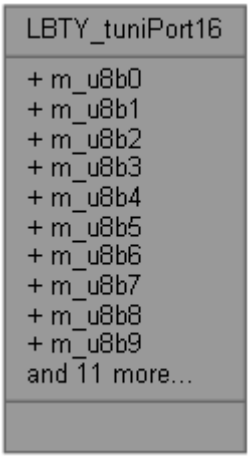
The documentation for this struct was generated from the following file:

IR_int.h

LBTY_tuniPort16 Union Reference

`#include <LBTY_int.h>`

Collaboration diagram for LBTY_tuniPort16:



Data Fields

- struct {
- u8 m_u8b0:1
- u8 m_u8b1:1
- u8 m_u8b2:1
- u8 m_u8b3:1
- u8 m_u8b4:1
- u8 m_u8b5:1
- u8 m_u8b6:1
- u8 m_u8b7:1
- u8 m_u8b8:1
- u8 m_u8b9:1
- u8 m_u8b10:1
- u8 m_u8b11:1
- u8 m_u8b12:1
- u8 m_u8b13:1
- u8 m_u8b14:1
- u8 m_u8b15:1
- } sBits
- struct {
- u8 m_u8low
- u8 m_u8high
- } sBytes
- u16 u_u16Word

Field Documentation

u8 m_u8b0

u8 m_u8b1

u8 m_u8b10

u8 m_u8b11

u8 m_u8b12

u8 m_u8b13

u8 m_u8b14

u8 m_u8b15

u8 m_u8b2

u8 m_u8b3

u8 m_u8b4

u8 m_u8b5

u8 m_u8b6

u8 m_u8b7

u8 m_u8b8

u8 m_u8b9

u8 m_u8high

u8 m_u8low

struct { ... } sBits

struct { ... } sBytes

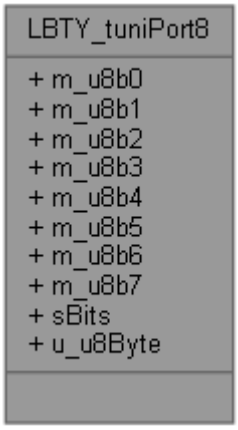
u16 u_u16Word

The documentation for this union was generated from the following file:

- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h

LBTY_tuniPort8 Union Reference

`#include <LBTY_int.h>`
Collaboration diagram for LBTY_tuniPort8:



Data Fields

- struct {
- u8 m_u8b0:1
- u8 m_u8b1:1
- u8 m_u8b2:1
- u8 m_u8b3:1
- u8 m_u8b4:1
- u8 m_u8b5:1
- u8 m_u8b6:1
- u8 m_u8b7:1
- } sBits
- u8 u_u8Byte

Detailed Description

Union Byte bit by bit

Field Documentation

u8 m_u8b0

u8 m_u8b1

u8 m_u8b2

u8 m_u8b3

u8 m_u8b4

u8 m_u8b5

u8 m_u8b6

u8 m_u8b7

struct { ... } sBits

u8 u_u8Byte

The documentation for this union was generated from the following file:

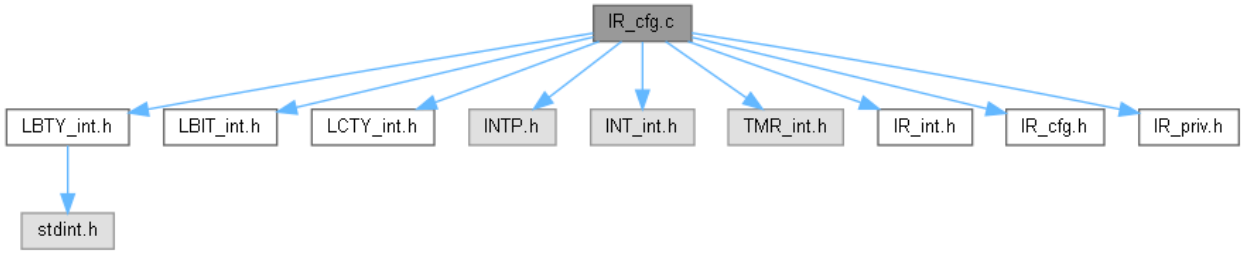
- H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h

File Documentation

IR_cfg.c File Reference

```
#include "LBTY_int.h"
#include "LBIT_int.h"
#include "LCTY_int.h"
#include "INTP.h"
#include "INT_int.h"
#include "TMR_int.h"
#include "IR_int.h"
#include "IR_cfg.h"
#include "IR_priv.h"
```

Include dependency graph for IR_cfg.c:



Functions

- void vid_IrResetPrevPacket (void)
- void vid_IrWriteBuffer (u8 u8CMD)
- void vid_IrReadBuffer (u8 *pu8CMD)
- void vid_IrReadPacket (IR_tstrPacket *pstrPacket)
- void vid_IrReadFram (IR_tstrFram *pstrFram)
- void vid_IrBitStep (void)
- void vid_IrLeadHigh (u16 u16TempTime)
- void vid_IrLeadLow (u16 u16TempTime)
- void vid_IrReceiveBits (u16 u16TempTime)
- void vid_IrStop (void)

Variables

- IR_tstrFram strReceiveFram_GLB
- static volatile u8 *const kpu8FramBytes_GLB = (u8*)&strReceiveFram_GLB.strPacket
- static volatile u8 au8ReceiveBuffer_GLB [IR_CMD_QUEUE_LENGTH]
- static volatile u8 u8BufferIndex_GLB = LBTY_u8ZERO
- static volatile u8 u8BufferEnd_GLB = LBTY_u8ZERO
- static IR_tstrPacket strPrevPacket_GLB

Function Documentation

void vid_IrBitStep (void)

```
91                          {
92      if(++strReceiveFram_GLB.m_u8ReadBit >= IR_CMD_MAX_LENGTH){
93          strReceiveFram_GLB.m_u8ReadBit = LBTY_u8ZERO;
94          if(++strReceiveFram_GLB.m_u8ReadByte >= IR_FRAM_LENGTH){
95              strReceiveFram_GLB.m_u8State= IR_WaitStopBit;
96          }
97      }
```

```
98 }
```

Here is the caller graph for this function:

void vid_IrLeadHigh (u16 *u16TempTime*)

```
100                                     {
101
102     if(strReceiveFram_GLB.m_u8Edge == INT_Rising_Edge){
103         if(IR_CHECK_TIME(u16TempTime, IR_HIGH_LEAD_TIME)){
104             strReceiveFram_GLB.m_u8State= IR_ValidateLeadLow;
105             strReceiveFram_GLB.m_u8Edge = INT_Falling_Edge;
106             INT vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
107         }else{      // error in Lead High
108             IR_Reset();
109         }
110     }else{
111         strReceiveFram_GLB.m_u8Edge = INT_Rising_Edge;
112         INT vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
113     }
114 }
```

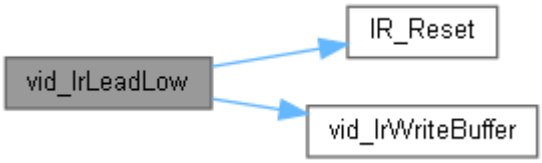
Here is the call graph for this function:

Here is the caller graph for this function:

void vid_IrLeadLow (u16 *u16TempTime*)

```
116                                     {
117
118     if(IR_CHECK_TIME(u16TempTime, IR_LOW0_LEAD_TIME)){
119         strReceiveFram_GLB.m_u8State= IR_ReceiveBits;
120         strReceiveFram_GLB.m_u8Edge = INT_Rising_Edge;
121         INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
122
123         strReceiveFram_GLB.m_u8ReadBit  = LBTY_u8ZERO;
124         strReceiveFram_GLB.m_u8ReadByte = LBTY_u8ZERO;
125
126     }else if(IR_CHECK_TIME(u16TempTime, IR_LOW1_LEAD_TIME)){
127         if(strReceiveFram_GLB.m_u8Repeat){
128             vid_IrWriteBuffer(strPrevPacket_GLB.m_u8Command);
129         }else{
130             if(++strReceiveFram_GLB.m_u8RepeatCount >= IR_REPEAT_MAX)
strReceiveFram_GLB.m_u8Repeat = LBTY_SET;
131         }
132         IR_Reset();
133     }else{      // error in Lead Low
134         IR_Reset();
135     }
136 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

void vid_IrReadBuffer (u8 * *pu8CMD*)

```
73                                      {
74     if((u8BufferEnd_GLB - u8BufferIndex_GLB) > LBTY_u8ZERO ||
75        (IR_CMD_QUEUE_LENGTH + u8BufferEnd_GLB - u8BufferIndex_GLB) <
IR_CMD_QUEUE_LENGTH){
```

```
76          *pu8CMD = au8ReceiveBuffer_GLB[u8BufferIndex_GLB];
77          if(++u8BufferIndex_GLB >= IR_CMD_QUEUE_LENGTH)  u8BufferIndex_GLB =
LBTY_u8ZERO;
78      }else{
79          *pu8CMD = LBTY_u8MAX;
80      }
81  }
```

Here is the caller graph for this function:

void vid_IrReadFram (IR_tstrFram * *pstrFram*)

```
87                                                  {
88      *pstrFram = strReceiveFram_GLB;
89  }
```

void vid_IrReadPacket (IR_tstrPacket * *pstrPacket*)

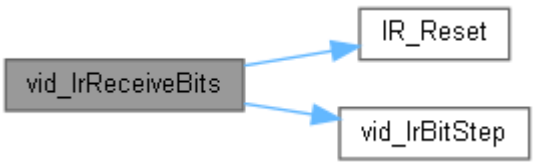
```
83                                                      {
84      *pstrPacket = strPrevPacket_GLB;
85  }
```

Here is the caller graph for this function:

void vid_IrReceiveBits (u16 *u16TempTime*)

```
138                                         {
139
140     if(strReceiveFram_GLB.m_u8Edge == INT_Rising_Edge){
141         if(IR_CHECK_TIME(u16TempTime, IR_HIGH_BIT_TIME)){
142             strReceiveFram_GLB.m_u8Edge = INT_Falling_Edge;
143             INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
144         }else{      // error in Bit Beg
145             IR_Reset();
146         }
147     }else{
148         strReceiveFram_GLB.m_u8Edge = INT_Rising_Edge;
149         INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
150
151         if(IR_CHECK_TIME(u16TempTime, IR_HIGH_BIT_TIME)){
152             vid_IrBitStep();
153         }else if(IR_CHECK_TIME(u16TempTime, IR_LOW_BIT_TIME)){
154             SET_BIT(kpu8FramBytes_GLB[strReceiveFram_GLB.m_u8ReadByte],
strReceiveFram_GLB.m_u8ReadBit);
155             vid_IrBitStep();
156         }else{      // error in read bit
157             IR_Reset();
158         }
159     }
160 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

void vid_IrResetPrevPacket (void)

```
54                                      {
55
56      strPrevPacket_GLB.m_u8Rec_Address    = LBTY_u8MAX;
57      strPrevPacket_GLB.m_u8Rec_AddressInv = LBTY_u8MAX;
58      strPrevPacket_GLB.m_u8Command        = LBTY_u8MAX;
59      strPrevPacket_GLB.m_u8CommandInv     = LBTY_u8MAX;
60
```

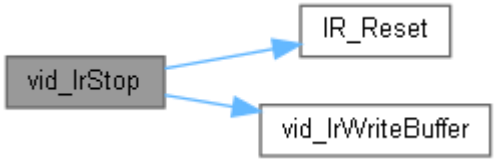
```
61 }
```

Here is the caller graph for this function:

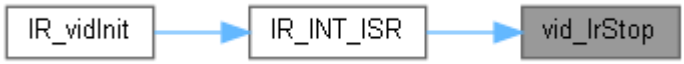
void vid_IrStop (void)

```
162                         {
163
164     if(strReceiveFram_GLB.m_u8Edge == INT_Rising_Edge){
165         if(strReceiveFram_GLB.strPacket.m_u8Command ==
(u8)~strReceiveFram_GLB.strPacket.m_u8CommandInv){
166
167             vid_IrWriteBuffer(strReceiveFram_GLB.strPacket.m_u8Command);
168             strPrevPacket_GLB = strReceiveFram_GLB.strPacket;
169             strReceiveFram_GLB.m_u8Repeat      = LBTY_u8ZERO;
170             strReceiveFram_GLB.m_u8RepeatCount= LBTY_u8ZERO;
171         }
172         IR_Reset();
173     }
174 }
```

Here is the call graph for this function:



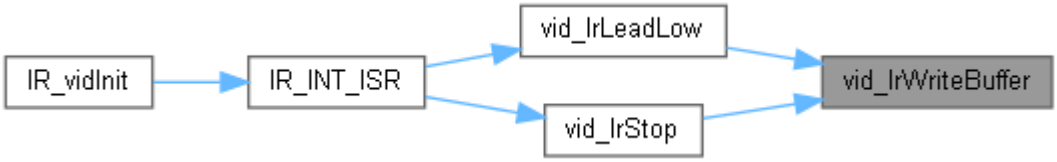
Here is the caller graph for this function:



void vid_IrWriteBuffer (u8 *u8CMD*)

```
63                                  {
64
65     au8ReceiveBuffer_GLB[u8BufferEnd_GLB] = u8CMD;
66     if(++u8BufferEnd_GLB >= IR_CMD_QUEUE_LENGTH)        u8BufferEnd_GLB =
LBTY_u8ZERO;
67
68     if(u8BufferEnd_GLB == u8BufferIndex_GLB){
69         if(++u8BufferIndex_GLB >= IR_CMD_QUEUE_LENGTH)  u8BufferIndex_GLB =
LBTY_u8ZERO;
70     }
71 }
```

Here is the caller graph for this function:



Variable Documentation

volatile u8 au8ReceiveBuffer_GLB[IR_CMD_QUEUE_LENGTH] [static]

volatile u8* const kpu8FramBytes_GLB = (u8*)&strReceiveFram_GLB.strPacket [static]

IR_tstrPacket strPrevPacket_GLB [static]

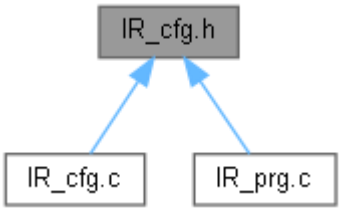
IR_tstrFram strReceiveFram_GLB [extern]

volatile u8 u8BufferEnd_GLB = LBTY_u8ZERO [static]

volatile u8 u8BufferIndex_GLB = LBTY_u8ZERO [static]

IR_cfg.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define IR_INT_PIN INT0
- #define IR_TMR_10US_COMPARE ((F_CPU / 100000u) - 1u)
- #define IR_TIME_TOL 0.25f
- #define IR_CHECK_TIME(temp, time) (temp>(time-(time*IR_TIME_TOL)) && temp<(time+(time*IR_TIME_TOL)))
- #define IR_HIGH_LEAD_TIME 900
- #define IR_LOW0_LEAD_TIME 450
- #define IR_LOW1_LEAD_TIME 225
- #define IR_LOW_BIT_TIME 169
- #define IR_HIGH_BIT_TIME 56
- #define IR_REPEAT_MAX 4u
- #define IR_FRAM_LENGTH 4u
- #define IR_CMD_MAX_LENGTH 8u
- #define IR_CMD_QUEUE_LENGTH 8u
- #define NEC_MAX_PACKET_BIT_NUMBER 32u
- #define IR_NONE_CMD LBTY_u8MAX
- #define IR_CMD_UP 0x0B
- #define IR_CMD_DOWN 0x1B
- #define IR_CMD_LEFT 0x5A
- #define IR_CMD_RIGHT 0x18
- #define IR_CMD_ENTER 0x58
- #define IR_CMD_VOL_INC 0x10
- #define IR_CMD_VOL_DEC 0x13

Macro Definition Documentation

#define IR_CHECK_TIME(temp, time) (temp>(time-(time*IR_TIME_TOL)) && temp<(time+(time*IR_TIME_TOL)))

#define IR_CMD_DOWN 0x1B

#define IR_CMD_ENTER 0x58

#define IR_CMD_LEFT 0x5A

#define IR_CMD_MAX_LENGTH 8u

#define IR_CMD_QUEUE_LENGTH 8u

#define IR_CMD_RIGHT 0x18

#define IR_CMD_UP 0x0B

#define IR_CMD_VOL_DEC 0x13

#define IR_CMD_VOL_INC 0x10

#define IR_FRAM_LENGTH 4u

#define IR_HIGH_BIT_TIME 56

#define IR_HIGH_LEAD_TIME 900

#define IR_INT_PIN INT0

#define IR_LOW0_LEAD_TIME 450

#define IR_LOW1_LEAD_TIME 225

#define IR_LOW_BIT_TIME 169

#define IR_NONE_CMD LBTY_u8MAX

#define IR_REPEAT_MAX 4u

#define IR_TIME_TOL 0.25f

#define IR_TMR_10US_COMPARE ((F_CPU / 100000u) - 1u)

#define NEC_MAX_PACKET_BIT_NUMBER 32u

IR_cfg.h

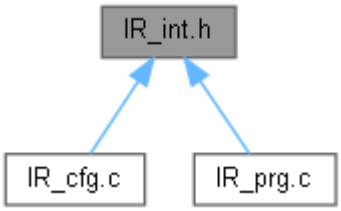
Go to the documentation of this file.

```
/*
/* ********************************************************************************** */
/* ********************** FILE DEFINITION SECTION ************************************ */
/* ********************************************************************************** */
/* File Name    : IR_cfg.h                                                            */
/* Author       : MAAM                                                                */
/* Version      : v01.2                                                               */
/* date         : May 3, 2023                                                         */
/* ********************************************************************************** */
/* ************************ HEADER FILES INCLUDES ***************************  */
/* ********************************************************************************** */

#ifndef IR_CFG_H_
#define IR_CFG_H_

/*
Sensor: TSOP1738 IR receiver module must be connected to INT0 Pin.
        This is PIN16 in ATmega16 and ATmega32.

           -----
          |  _  |
          | | | |
          | | | |
          -------
          | |  |
          | |  |
          | |  |
       (5V)(GND)(PD2)

    **********************
    *TSOP 1738 Front View*
    **********************
    Resource Usage:     -Timer0     -INT0 (PD2)
 */
/* ********************************************************************************** */
/* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
/* ********************************************************************************** */

/* ********************************************************************************** */
/* ************************** MACRO/DEFINE SECTION ************************** */
/* ********************************************************************************** */

#define IR_INT_PIN                  INT0
#define IR_TMR_10US_COMPARE         ((F_CPU / 100000u) - 1u)

#define IR_TIME_TOL                 0.25f   //0.2f
#define IR_CHECK_TIME(temp, time)   (temp>(time-(time*IR_TIME_TOL)) &&
temp<(time+(time*IR_TIME_TOL)))

#define IR_HIGH_LEAD_TIME           900
#define IR_LOW0_LEAD_TIME           450
#define IR_LOW1_LEAD_TIME           225
#define IR_LOW_BIT_TIME             169
#define IR_HIGH_BIT_TIME            56

#define IR_REPEAT_MAX               4u
#define IR_FRAM_LENGTH              4u
#define IR_CMD_MAX_LENGTH           8u
#define IR_CMD_QUEUE_LENGTH         8u

#define NEC_MAX_PACKET_BIT_NUMBER   32u

#define IR_NONE_CMD                 LBTY_u8MAX
#define IR_CMD_UP                   0x0B
#define IR_CMD_DOWN                 0x1B
#define IR_CMD_LEFT                 0x5A
#define IR_CMD_RIGHT                0x18
#define IR_CMD_ENTER                0x58
#define IR_CMD_VOL_INC              0x10
#define IR_CMD_VOL_DEC              0x13
/* ********************************************************************************** */
/* ***************************** CONST SECTION ****************************** */
/* ********************************************************************************** */
```

```
74
75 /* ********************************************************************** */
76 /* ***************************** VARIABLE SECTION ************************* */
77 /* ********************************************************************** */
78
79 /* ********************************************************************** */
80 /* ***************************** FUNCTION SECTION ************************* */
81 /* ********************************************************************** */
82
83
84 #endif /* IR_CFG_H_ */
85 /***************************** E N D (IR_cfg.h) ******************************/
```

IR_int.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

struct IR_tstrPacketEnumerations

- enum IR_tenuState { IR_ValidateLeadHigh = (u8)0u, IR_ValidateLeadLow, IR_ReceiveAddress, IR_ReceiveBits, IR_WaitStopBit }

Functions

- void IR_Reset (void)
- void IR_vidInit (void)
- LBTY_tenuErrorStatus IR_GetCmd (u8 *pu8CMD)
- LBTY_tenuErrorStatus IR_GetPacket (IR_tstrPacket *pstrPacket)
- void IR_INT_ISR (void)
- void IR_TMR_ISR (void)

Enumeration Type Documentation

enum IR_tenuState

Enumerator:

IR_ValidateLeadHigh	
IR_ValidateLeadLow	
IR_ReceiveAddress	
IR_ReceiveBits	
IR_WaitStopBit	

```
36              {
37      IR_ValidateLeadHigh = (u8)0u,
38      IR_ValidateLeadLow,
39      IR_ReceiveAddress,
40      IR_ReceiveBits,
41      IR_WaitStopBit
42  }IR_tenuState;
```

Function Documentation

LBTY_tenuErrorStatus IR_GetCmd (u8 * *pu8CMD*)

```
80                                              {
81      LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
82
83      vid_IrReadBuffer(pu8CMD);
```

```
84      if(*pu8CMD == LBTY_u8MAX)       u8RetErrorState = LBTY_IN_PROGRESS;
85      return u8RetErrorState;
86  }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus IR_GetPacket (IR_tstrPacket * *pstrPacket*)

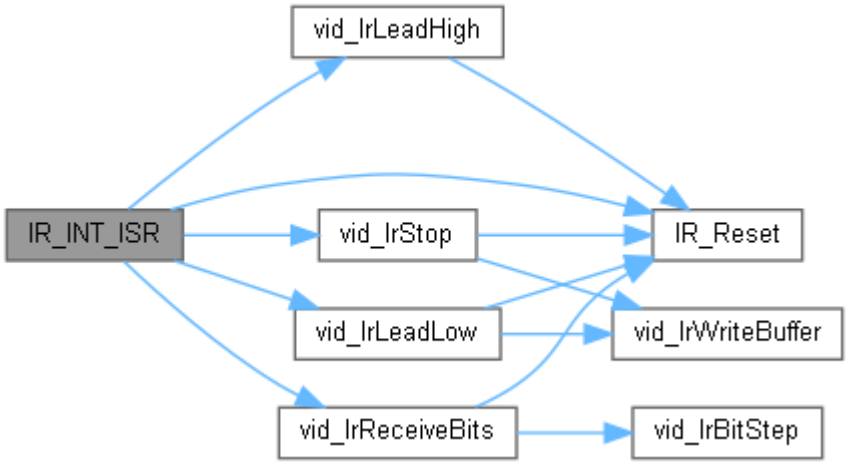
```
88                                                          {
89      LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
90
91      vid_IrReadPacket(pstrPacket);
92      if(pstrPacket->m_u8Command == LBTY_u32MAX)  u8RetErrorState =
LBTY_IN_PROGRESS;
93      return u8RetErrorState;
94  }
```

Here is the call graph for this function:

void IR_INT_ISR (void)

```
96                      {
97      static u16 u16TempTime;
98      u16TempTime = strReceiveFram_GLB.m_u16Time;
99      strReceiveFram_GLB.m_u16Time = 20u;
100
101      INT_vidDisable   (IR_INT_PIN);
102      //if(strReceiveFram_GLB.m_u8StopState) return;
103
104      switch(strReceiveFram_GLB.m_u8State){
105          case IR_ValidateLeadHigh:    vid_IrLeadHigh    (u16TempTime);      break;
106          case IR_ValidateLeadLow:     vid_IrLeadLow     (u16TempTime);      break;
107          case IR_ReceiveAddress:          break;
108          case IR_ReceiveBits:         vid_IrReceiveBits(u16TempTime);      break;
109          case IR_WaitStopBit:         vid_IrStop();                        break;
110
111          default:     IR_Reset();
112      }
113
114      INT_vidEnable    (IR_INT_PIN);
115  }
```

Here is the call graph for this function:



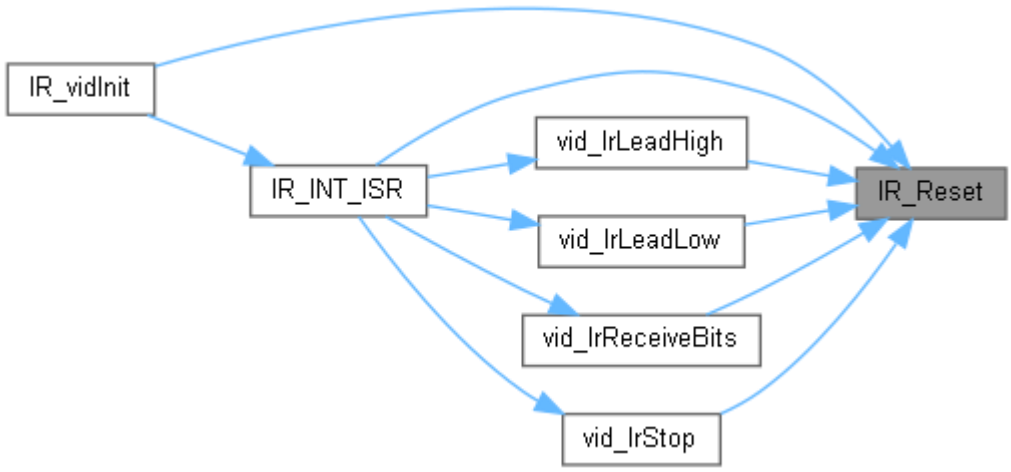
Here is the caller graph for this function:

void IR_Reset (void)

```
47                      {
48  //  strReceiveFram_GLB.strPacket.m_u16Ext_Address    = LBTY_u16ZERO;
49      strReceiveFram_GLB.strPacket.m_u8Rec_Address    = LBTY_u8ZERO;
```

```
50      strReceiveFram_GLB.strPacket.m_u8Rec_AddressInv = LBTY_u8ZERO;
51      strReceiveFram_GLB.strPacket.m_u8Command        = LBTY_u8ZERO;
52      strReceiveFram_GLB.strPacket.m_u8CommandInv     = LBTY_u8ZERO;
53
54      strReceiveFram_GLB.m_u8State       = IR_ValidateLeadHigh;
55      strReceiveFram_GLB.m_u8Edge        = INT_Falling_Edge;
56
57      INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
58
59      TMR0_u8SetMode(TMRx_u8_CTC_Mode_Mode);
60      TMR0_u8SetOutputCompare(IR_TMR_10US_COMPARE);
61  }
```

Here is the caller graph for this function:

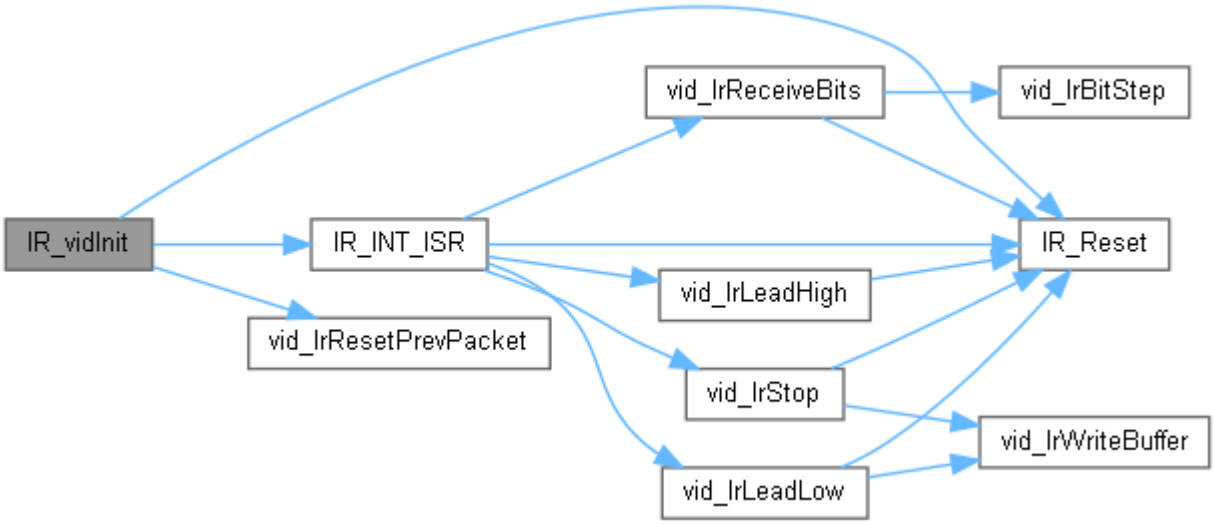


void IR_TMR_ISR (void)

void IR_vidInit (void)

```
63                      {
64
65      INT_vidInit(IR_INT_PIN);
66      INT_vidSetCallBack(IR_INT_PIN, IR_INT_ISR);
67
68      TMR0_vidInit();
69
70      vid_IrResetPrevPacket();
71      strReceiveFram_GLB.m_u8Repeat      = LBTY_u8ZERO;
72      strReceiveFram_GLB.m_u8RepeatCount= LBTY_u8ZERO;
73      strReceiveFram_GLB.m_u8ReadBit     = LBTY_u8ZERO;
74      strReceiveFram_GLB.m_u8ReadByte    = LBTY_u8ZERO;
75      strReceiveFram_GLB.m_u8Stop        = LBTY_u8ZERO;
76
77      IR_Reset();
78  }
```

Here is the call graph for this function:


IR_vidInit
IR_INT_ISR
vid_IrReceiveBits
vid_IrBitStep
IR_Reset
vid_IrLeadHigh
vid_IrResetPrevPacket
vid_IrStop
vid_IrWriteBuffer
vid_IrLeadLow

IR_int.h

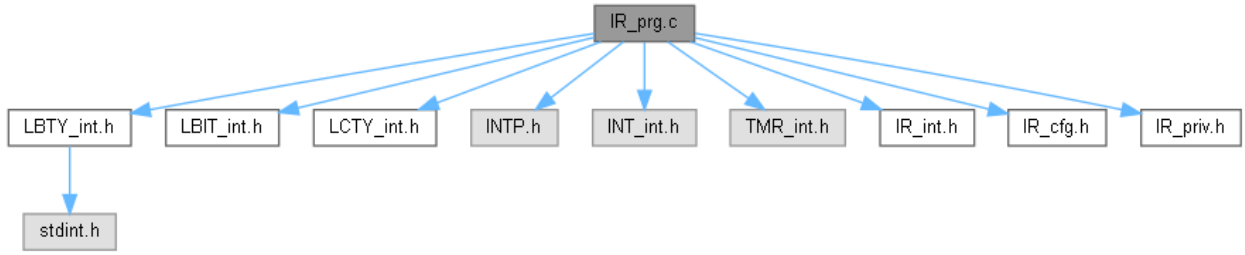
```
Go to the documentation of this file.1 /*
****************************************************************************** */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ****************************************************************************** */
4 /* File Name    : IR_int.h                                                    */
5 /* Author       : MAAM                                                        */
6 /* Version      : v01.2                                                       */
7 /* date         : May 3, 2023                                                 */
8 /* ****************************************************************************** */
9 /* ************************ HEADER FILES INCLUDES ************************  */
10 /* ****************************************************************************** */
11
12 #ifndef IR_INT_H_
13 #define IR_INT_H_
14
15 #ifndef F_CPU
16 #error "IR Remote Lib : F_CPU not defined"
17 #endif
18
19 #if ((F_CPU !=8000000) && (F_CPU !=12000000) && (F_CPU !=16000000))
20 #error "IR Remote Lib : Unsupported CPU frequency"
21 #error "IR remote Lib : Pls use F_CPU = 8MHz,12Mhz or 16MHz"
22 #endif
23
24 #if F_CPU==8000000
25     #define TIMER_COMP_VAL 80
26 #elif F_CPU==12000000
27     #define TIMER_COMP_VAL 120
28 #elif F_CPU==16000000
29     #define TIMER_COMP_VAL 160
30 #endif
31
32 /* ****************************************************************************** */
33 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
34 /* ****************************************************************************** */
35
36 typedef enum{
37     IR_ValidateLeadHigh = (u8)0u,
38     IR_ValidateLeadLow,
39     IR_ReceiveAddress,
40     IR_ReceiveBits,
41     IR_WaitStopBit
42 }IR_tenuState;
43
44 typedef struct{
45         //volatile u16 m_u16Ext_Address;        // device extended address, 0 if it is
not used
46         volatile u8  m_u8Rec_Address;
47         volatile u8  m_u8Rec_AddressInv;
48         volatile u8  m_u8Command;
49         volatile u8  m_u8CommandInv;
50 }IR_tstrPacket;
51
52 /* ****************************************************************************** */
53 /* ************************** MACRO/DEFINE SECTION ************************** */
54 /* ****************************************************************************** */
55
56 /* ****************************************************************************** */
57 /* ***************************** CONST SECTION ****************************** */
58 /* ****************************************************************************** */
59
60 /* ****************************************************************************** */
61 /* ***************************** VARIABLE SECTION ***************************** */
62 /* ****************************************************************************** */
63
64 /* ****************************************************************************** */
65 /* ***************************** FUNCTION SECTION ***************************** */
66 /* ****************************************************************************** */
67
68 void IR_Reset(void);
69
70 void IR_vidInit(void);
71
```

```
72 LBTY tenuErrorStatus IR_GetCmd(u8* pu8CMD);
73 LBTY tenuErrorStatus IR_GetPacket(IR_tstrPacket* pstrPacket);
74
75 void IR_INT_ISR(void);
76
77 void IR_TMR_ISR(void);
78
79
80 #endif /* IR_INT_H_ */
81 /*************************** E N D (IR_int.h) ****************************/
```

IR_prg.c File Reference

```
#include "LBTY_int.h"
#include "LBIT_int.h"
#include "LCTY_int.h"
#include "INTP.h"
#include "INT_int.h"
#include "TMR_int.h"
#include "IR_int.h"
#include "IR_cfg.h"
#include "IR_priv.h"
```

Include dependency graph for IR_prg.c:



Functions

- void IR_Reset (void)
- void IR_vidInit (void)
- LBTY_tenuErrorStatus IR_GetCmd (u8 *pu8CMD)
- LBTY_tenuErrorStatus IR_GetPacket (IR_tstrPacket *pstrPacket)
- void IR_INT_ISR (void)
- ISR (TIMER0_COMP_vect)

Variables

- IR_tstrFram strReceiveFram_GLB

Function Documentation

LBTY_tenuErrorStatus IR_GetCmd (u8 * *pu8CMD*)

```
80                                                  {
81     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
82
83     vid_IrReadBuffer(pu8CMD);
84     if(*pu8CMD == LBTY_u8MAX)       u8RetErrorState = LBTY_IN_PROGRESS;
85     return u8RetErrorState;
86 }
```

Here is the call graph for this function:

LBTY_tenuErrorStatus IR_GetPacket (IR_tstrPacket * *pstrPacket*)

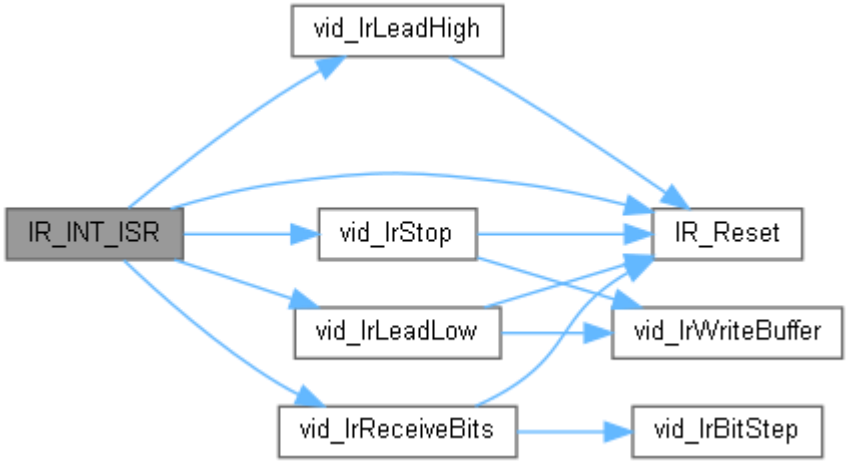
```
88                                                                  {
89     LBTY_tenuErrorStatus u8RetErrorState = LBTY_OK;
90
91     vid_IrReadPacket(pstrPacket);
92     if(pstrPacket->m_u8Command == LBTY_u32MAX)  u8RetErrorState =
LBTY_IN_PROGRESS;
93     return u8RetErrorState;
94 }
```

Here is the call graph for this function:

void IR_INT_ISR (void)

```
96                      {
97      static u16 u16TempTime;
98      u16TempTime = strReceiveFram_GLB.m_u16Time;
99      strReceiveFram_GLB.m_u16Time = 20u;
100
101      INT_vidDisable  (IR_INT_PIN);
102      //if(strReceiveFram_GLB.m_u8StopState) return;
103
104      switch(strReceiveFram_GLB.m_u8State){
105          case IR_ValidateLeadHigh:   vid_IrLeadHigh    (u16TempTime);     break;
106          case IR_ValidateLeadLow:    vid_IrLeadLow     (u16TempTime);     break;
107          case IR_ReceiveAddress:         break;
108          case IR_ReceiveBits:        vid_IrReceiveBits(u16TempTime);     break;
109          case IR_WaitStopBit:        vid_IrStop();                       break;
110
111          default:    IR_Reset();
112      }
113
114      INT_vidEnable   (IR_INT_PIN);
115  }
```

Here is the call graph for this function:

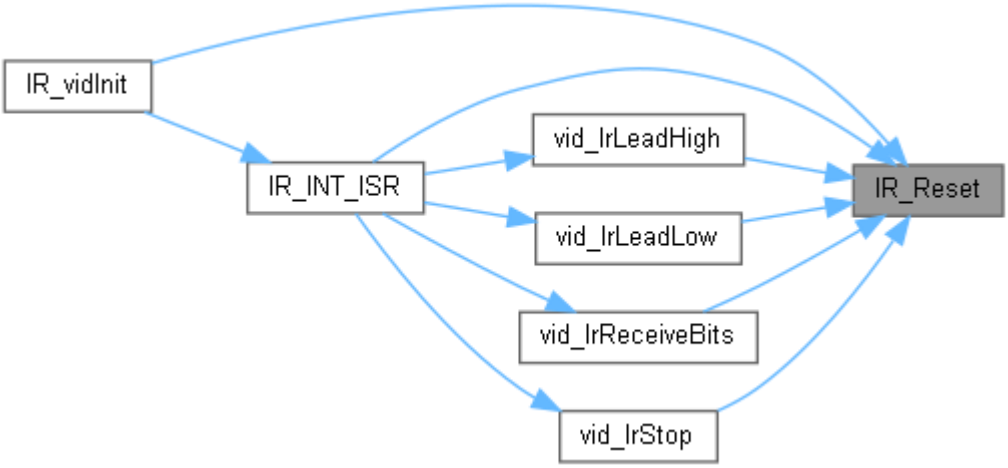


Here is the caller graph for this function:

void IR_Reset (void)

```
47                      {
48  //  strReceiveFram_GLB.strPacket.m_u16Ext_Address    = LBTY_u16ZERO;
49      strReceiveFram_GLB.strPacket.m_u8Rec_Address    = LBTY_u8ZERO;
50      strReceiveFram_GLB.strPacket.m_u8Rec_AddressInv = LBTY_u8ZERO;
51      strReceiveFram_GLB.strPacket.m_u8Command        = LBTY_u8ZERO;
52      strReceiveFram_GLB.strPacket.m_u8CommandInv     = LBTY_u8ZERO;
53
54      strReceiveFram_GLB.m_u8State      = IR_ValidateLeadHigh;
55      strReceiveFram_GLB.m_u8Edge       = INT_Falling_Edge;
56
57      INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
58
59      TMR0_u8SetMode(TMRx_u8_CTC_Mode_Mode);
60      TMR0_u8SetOutputCompare(IR_TMR_10US_COMPARE);
61  }
```

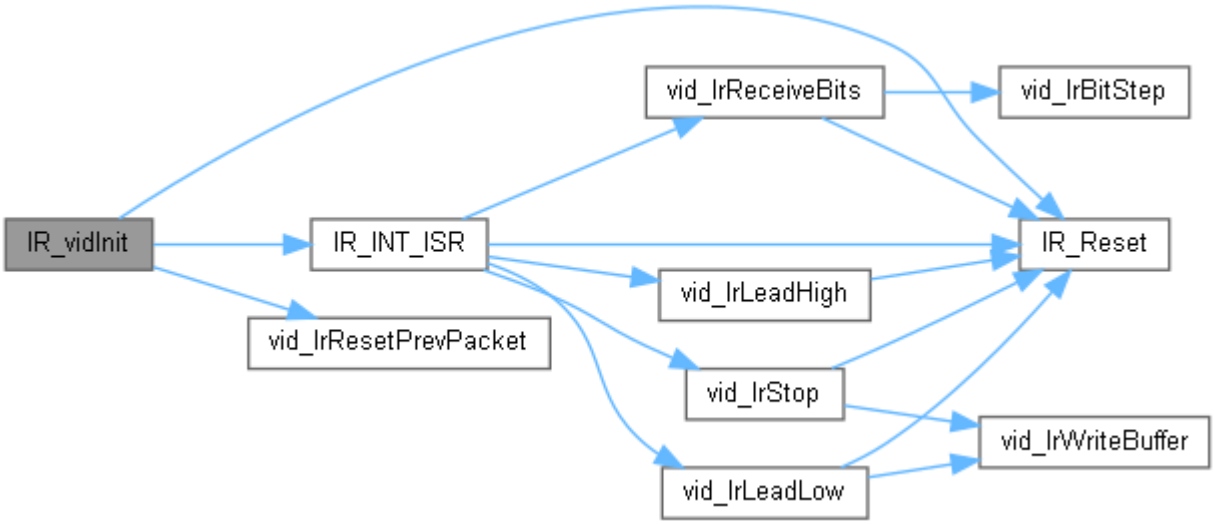
Here is the caller graph for this function:



void IR_vidInit (void)

```
63                          {
64
65      INT_vidInit(IR_INT_PIN);
66      INT_vidSetCallBack(IR_INT_PIN, IR_INT_ISR);
67
68      TMR0_vidInit();
69
70      vid_IrResetPrevPacket();
71      strReceiveFram_GLB.m_u8Repeat      = LBTY_u8ZERO;
72      strReceiveFram_GLB.m_u8RepeatCount= LBTY_u8ZERO;
73      strReceiveFram_GLB.m_u8ReadBit     = LBTY_u8ZERO;
74      strReceiveFram_GLB.m_u8ReadByte    = LBTY_u8ZERO;
75      strReceiveFram_GLB.m_u8Stop        = LBTY_u8ZERO;
76
77      IR_Reset();
78  }
```

Here is the call graph for this function:



ISR (TIMER0_COMP_vect)

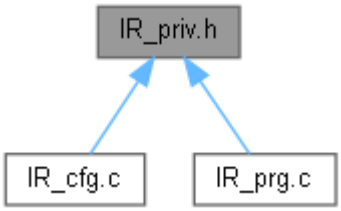
```
117                         {
118     strReceiveFram_GLB.m_u16Time++;
119 }
```

Variable Documentation

IR_tstrFram strReceiveFram_GLB

IR_priv.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

struct IR_tstrFramFunctions

- void vid_IrResetPrevPacket (void)
- void vid_IrWriteBuffer (u8 u8CMD)
- void vid_IrReadBuffer (u8 *pu8CMD)
- void vid_IrReadPacket (IR_tstrPacket *pstrPacket)
- void vid_IrReadFram (IR_tstrFram *pstrFram)
- void vid_IrBitStep (void)
- void vid_IrLeadHigh (u16 u16TempTime)
- void vid_IrLeadLow (u16 u16TempTime)
- void vid_IrReceiveBits (u16 u16TempTime)
- void vid_IrStop (void)

Function Documentation

void vid_IrBitStep (void)

```
91                          {
92      if(++strReceiveFram_GLB.m_u8ReadBit >= IR_CMD_MAX_LENGTH){
93          strReceiveFram_GLB.m_u8ReadBit = LBTY_u8ZERO;
94          if(++strReceiveFram_GLB.m_u8ReadByte >= IR_FRAM_LENGTH){
95              strReceiveFram_GLB.m_u8State= IR_WaitStopBit;
96          }
97      }
98  }
```

Here is the caller graph for this function:

void vid_IrLeadHigh (u16 *u16TempTime*)

```
100                                     {
101
102      if(strReceiveFram_GLB.m_u8Edge == INT_Rising_Edge){
103          if(IR_CHECK_TIME(u16TempTime, IR_HIGH_LEAD_TIME)){
104              strReceiveFram_GLB.m_u8State= IR_ValidateLeadLow;
105              strReceiveFram_GLB.m_u8Edge = INT_Falling_Edge;
106              INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
107          }else{      // error in Lead High
108              IR_Reset();
109          }
110      }else{
111          strReceiveFram_GLB.m_u8Edge = INT_Rising_Edge;
112          INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
113      }
114  }
```

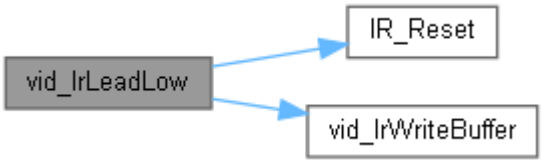
Here is the call graph for this function:

Here is the caller graph for this function:

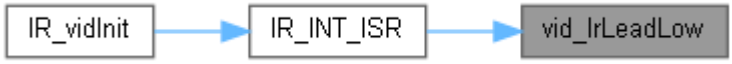
void vid_IrLeadLow (u16 *u16TempTime*)

```
116                                     {
117
118     if(IR_CHECK_TIME(u16TempTime, IR_LOW0_LEAD_TIME)){
119         strReceiveFram_GLB.m_u8State= IR_ReceiveBits;
120         strReceiveFram_GLB.m_u8Edge = INT_Rising_Edge;
121         INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
122
123         strReceiveFram_GLB.m_u8ReadBit  = LBTY_u8ZERO;
124         strReceiveFram_GLB.m_u8ReadByte = LBTY_u8ZERO;
125
126     }else if(IR_CHECK_TIME(u16TempTime, IR_LOW1_LEAD_TIME)){
127         if(strReceiveFram_GLB.m_u8Repeat){
128             vid_IrWriteBuffer(strPrevPacket_GLB.m_u8Command);
129         }else{
130             if(++strReceiveFram_GLB.m_u8RepeatCount >= IR_REPEAT_MAX)
strReceiveFram_GLB.m_u8Repeat = LBTY_SET;
131         }
132         IR_Reset();
133     }else{      // error in Lead Low
134         IR_Reset();
135     }
136 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



void vid_IrReadBuffer (u8 * *pu8CMD*)

```
73                                      {
74     if((u8BufferEnd_GLB - u8BufferIndex_GLB) > LBTY_u8ZERO ||
75        (IR_CMD_QUEUE_LENGTH + u8BufferEnd_GLB - u8BufferIndex_GLB) <
IR_CMD_QUEUE_LENGTH){
76         *pu8CMD = au8ReceiveBuffer_GLB[u8BufferIndex_GLB];
77         if(++u8BufferIndex_GLB >= IR_CMD_QUEUE_LENGTH)  u8BufferIndex_GLB =
LBTY_u8ZERO;
78     }else{
79         *pu8CMD = LBTY_u8MAX;
80     }
81 }
```

Here is the caller graph for this function:

void vid_IrReadFram (IR_tstrFram * *pstrFram*)

```
87                                      {
88     *pstrFram = strReceiveFram_GLB;
89 }
```

void vid_IrReadPacket (IR_tstrPacket * *pstrPacket*)

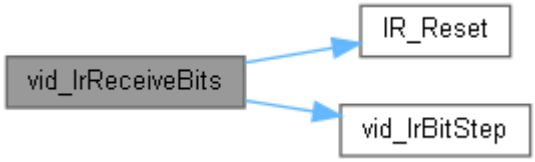
```
83                                              {
84     *pstrPacket = strPrevPacket_GLB;
85 }
```

Here is the caller graph for this function:

void vid_IrReceiveBits (u16 *u16TempTime*)

```
138                                         {
139
140     if(strReceiveFram_GLB.m_u8Edge == INT_Rising_Edge){
141         if(IR_CHECK_TIME(u16TempTime, IR_HIGH_BIT_TIME)){
142             strReceiveFram_GLB.m_u8Edge = INT_Falling_Edge;
143             INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
144         }else{      // error in Bit Beg
145             IR_Reset();
146         }
147     }else{
148         strReceiveFram_GLB.m_u8Edge = INT_Rising_Edge;
149         INT_vidSetSenseControl(IR_INT_PIN, strReceiveFram_GLB.m_u8Edge);
150
151         if(IR_CHECK_TIME(u16TempTime, IR_HIGH_BIT_TIME)){
152             vid_IrBitStep();
153         }else if(IR_CHECK_TIME(u16TempTime, IR_LOW_BIT_TIME)){
154             SET_BIT(kpu8FramBytes_GLB[strReceiveFram_GLB.m_u8ReadByte],
strReceiveFram_GLB.m_u8ReadBit);
155             vid_IrBitStep();
156         }else{      // error in read bit
157             IR_Reset();
158         }
159     }
160 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

void vid_IrResetPrevPacket (void)

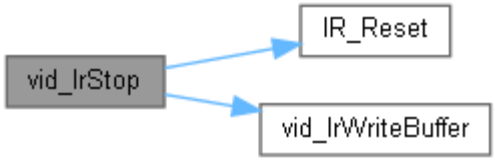
```
54                                      {
55
56     strPrevPacket_GLB.m_u8Rec_Address    = LBTY_u8MAX;
57     strPrevPacket_GLB.m_u8Rec_AddressInv = LBTY_u8MAX;
58     strPrevPacket_GLB.m_u8Command        = LBTY_u8MAX;
59     strPrevPacket_GLB.m_u8CommandInv     = LBTY_u8MAX;
60
61 }
```

Here is the caller graph for this function:

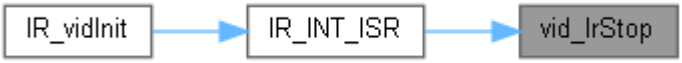
void vid_IrStop (void)

```
162                         {
163
164     if(strReceiveFram_GLB.m_u8Edge == INT_Rising_Edge){
165         if(strReceiveFram_GLB.strPacket.m_u8Command ==
(u8)~strReceiveFram_GLB.strPacket.m_u8CommandInv){
166
167             vid_IrWriteBuffer(strReceiveFram_GLB.strPacket.m_u8Command);
168             strPrevPacket_GLB = strReceiveFram_GLB.strPacket;
169             strReceiveFram_GLB.m_u8Repeat      = LBTY_u8ZERO;
170             strReceiveFram_GLB.m_u8RepeatCount= LBTY_u8ZERO;
171         }
172         IR_Reset();
173     }
174 }
```

Here is the call graph for this function:



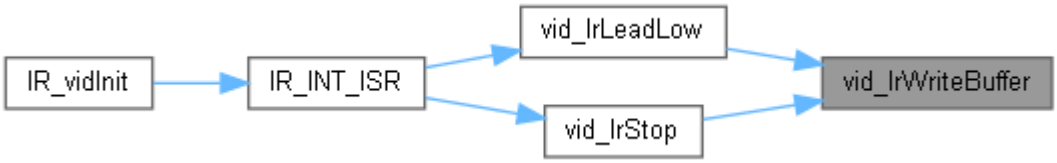
Here is the caller graph for this function:



void vid_IrWriteBuffer (u8 *u8CMD*)

```
63                                      {
64
65      au8ReceiveBuffer_GLB[u8BufferEnd_GLB] = u8CMD;
66      if(++u8BufferEnd_GLB >= IR_CMD_QUEUE_LENGTH)         u8BufferEnd_GLB =
LBTY_u8ZERO;
67
68      if(u8BufferEnd_GLB == u8BufferIndex_GLB){
69          if(++u8BufferIndex_GLB >= IR_CMD_QUEUE_LENGTH)   u8BufferIndex_GLB =
LBTY_u8ZERO;
70      }
71  }
```

Here is the caller graph for this function:



IR_priv.h

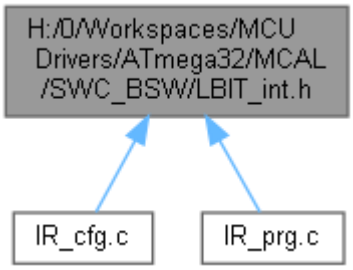
Go to the documentation of this file.

```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name    : IR_priv.h                                                  */
5 /* Author       : MAAM                                                       */
6 /* Version      : v01.2                                                      */
7 /* date         : May 3, 2023                                                */
8 /* ************************************************************************* */
9 /* ************************ HEADER FILES INCLUDES ************************  */
10 /* ************************************************************************* */
11
12 #ifndef IR_PRIV_H_
13 #define IR_PRIV_H_
14
15 /* ************************************************************************* */
16 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
17 /* ************************************************************************* */
18
19 typedef struct{
20     IR_tstrPacket strPacket;
21     volatile u8  m_u8State;
22     volatile u8  m_u8Edge;
23     volatile u8  m_u8Stop;
24
25     volatile u8  m_u8Repeat;
26     volatile u8  m_u8RepeatCount;
27
28     volatile u8  m_u8ReadBit;
29     volatile u8  m_u8ReadByte;
30
31     volatile u16 m_u16Time;
32 }IR_tstrFram;
33
34 /* ************************************************************************* */
35 /* ************************** MACRO/DEFINE SECTION ************************** */
36 /* ************************************************************************* */
37
38 /* ************************************************************************* */
39 /* ***************************** CONST SECTION ***************************** */
40 /* ************************************************************************* */
41
42 /* ************************************************************************* */
43 /* ***************************** VARIABLE SECTION **************************** */
44 /* ************************************************************************* */
45
46 /* ************************************************************************* */
47 /* ***************************** FUNCTION SECTION **************************** */
48 /* ************************************************************************* */
49
50 void vid_IrResetPrevPacket(void);
51
52 void vid_IrWriteBuffer(u8 u8CMD);
53
54 void vid_IrReadBuffer(u8* pu8CMD);
55
56 void vid_IrReadPacket(IR_tstrPacket* pstrPacket);
57 void vid_IrReadFram(IR_tstrFram* pstrFram);
58
59 void vid_IrBitStep(void);
60
61 void vid_IrLeadHigh(u16 u16TempTime);
62 void vid_IrLeadLow(u16 u16TempTime);
63 void vid_IrReceiveBits(u16 u16TempTime);
64 void vid_IrStop(void);
65
66
67 #endif /* IR_PRIV_H_ */
68 /************************** E N D (IR_priv.h) ***************************/
```

main.c File Reference

H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBIT_int.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define _BV(bit) (1u<<(bit))
- #define SET_BIT(REG, bit) ((REG) |= (1u<<(bit)))
- #define CLR_BIT(REG, bit) ((REG) &= ~(1u<<(bit)))
- #define TOG_BIT(REG, bit) ((REG) ^= (1u<<(bit)))
- #define SET_BYTE(REG, bit) ((REG) |= (0xFFu<<(bit)))
- #define CLR_BYTE(REG, bit) ((REG) &= ~(0xFFu<<(bit)))
- #define TOG_BYTE(REG, bit) ((REG) ^= (0xFFu<<(bit)))
- #define SET_MASK(REG, MASK) ((REG) |= (MASK))
- #define CLR_MASK(REG, MASK) ((REG) &= ~(MASK))
- #define TOG_MASK(REG, MASK) ((REG) ^= (MASK))
- #define GET_MASK(REG, MASK) ((REG) & (MASK))
- #define SET_REG(REG) ((REG) = ~(0u))
- #define CLR_REG(REG) ((REG) = (0u))
- #define TOG_REG(REG) ((REG) ^= ~(0u))
- #define GET_BIT(REG, bit) (((REG)>>(bit)) & 0x01u)
- #define GET_NIB(REG, bit) (((REG)>>(bit)) & 0x0Fu)
- #define GET_BYTE(REG, bit) (((REG)>>(bit)) & 0xFFu)
- #define ASSIGN_BIT(REG, bit, value) ((REG) = ((REG) & ~(0x01u<<(bit))) | (((value) & 0x01u)<<(bit)))
- #define ASSIGN_NIB(REG, bit, value) ((REG) = ((REG) & ~(0x0Fu<<(bit))) | (((value) & 0x0Fu)<<(bit)))
- #define ASSIGN_BYTE(REG, bit, value) ((REG) = ((REG) & ~(0xFFu<<(bit))) | (((value) & 0xFFu)<<(bit)))
- #define CON_u8Bits(b7, b6, b5, b4, b3, b2, b1, b0)

  (0b##b7##b6##b5##b4##b3##b2##b1##b0)
- #define CON_u16Bits(b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5, b4, b3, b2, b1, b0)

  (0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)

Macro Definition Documentation

#define _BV(bit) (1u<<(bit))

#define ASSIGN_BIT(REG, bit, value) ((REG) = ((REG) & ~(0x01u<<(bit))) | (((value) & 0x01u)<<(bit)))

#define ASSIGN_BYTE(REG, bit, value) ((REG) = ((REG) & ~(0xFFu<<(bit))) | (((value) & 0xFFu)<<(bit)))

#define ASSIGN_NIB(REG, bit, value) ((REG) = ((REG) & ~(0x0Fu<<(bit))) | (((value) & 0x0Fu)<<(bit)))

#define CLR_BIT(REG, bit) ((REG) &= ~(1u<<(bit)))

#define CLR_BYTE(REG, bit) ((REG) &= ~(0xFFu<<(bit)))

#define CLR_MASK(REG, MASK) ((REG) &= ~(MASK))

#define CLR_REG(REG) ((REG) = (0u))

#define CON_u16Bits(b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5, b4, b3, b2, b1, b0)

(0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)

#define CON_u8Bits(b7, b6, b5, b4, b3, b2, b1, b0)

(0b##b7##b6##b5##b4##b3##b2##b1##b0)

#define GET_BIT(REG, bit) (((REG)>>(bit)) & 0x01u)

#define GET_BYTE(REG, bit) (((REG)>>(bit)) & 0xFFu)

#define GET_MASK(REG, MASK) ((REG) & (MASK))

#define GET_NIB(REG, bit) (((REG)>>(bit)) & 0x0Fu)

#define SET_BIT(REG, bit) ((REG) |= (1u<<(bit)))

Bitwise Operation

```
#define SET_BYTE( REG,   bit)  ((REG) |=   (0xFFu<<(bit)))

#define SET_MASK( REG,   MASK)  ((REG) |=   (MASK))

#define SET_REG( REG)  ((REG)   = ~(0u))

#define TOG_BIT( REG,   bit)  ((REG) ^=   (1u<<(bit)))

#define TOG_BYTE( REG,   bit)  ((REG) ^=   (0xFFu<<(bit)))

#define TOG_MASK( REG,   MASK)  ((REG) ^=   (MASK))

#define TOG_REG( REG)  ((REG) ^= ~(0u))
```

LBIT_int.h

Go to the documentation of this file.

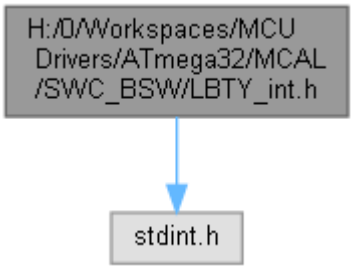
```
1 /*
******************************************************************************* */
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ******************************************************************************* */
4 /* File Name   : LBIT_int.h                                                    */
5 /* Author      : MAAM                                                          */
6 /* Version     : v01                                                           */
7 /* date        : Mar 24, 2023                                                  */
8 /* description : Bitwise Library                                               */
9 /* ******************************************************************************* */
10 /* ************************ HEADER FILES INCLUDES ************************  */
11 /* ******************************************************************************* */
12
13 #ifndef LBIT_INT_H_
14 #define LBIT_INT_H_
15
16 /* ******************************************************************************* */
17 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
18 /* ******************************************************************************* */
19
20 /* ******************************************************************************* */
21 /* ************************** MACRO/DEFINE SECTION ************************** */
22 /* ******************************************************************************* */
23
24 #define _BV(bit)                                          (1u<<(bit))
25
27 #define SET_BIT(REG, bit)                                 ((REG) |=  (1u<<(bit)))
28 #define CLR_BIT(REG, bit)                                 ((REG) &= ~(1u<<(bit)))
29 #define TOG_BIT(REG, bit)                                 ((REG) ^=  (1u<<(bit)))
30
31 #define SET_BYTE(REG, bit)                                ((REG) |=  (0xFFu<<(bit)))
32 #define CLR_BYTE(REG, bit)                                ((REG) &= ~(0xFFu<<(bit)))
33 #define TOG_BYTE(REG, bit)                                ((REG) ^=  (0xFFu<<(bit)))
34
35 #define SET_MASK(REG, MASK)                               ((REG) |=  (MASK))
36 #define CLR_MASK(REG, MASK)                               ((REG) &= ~(MASK))
37 #define TOG_MASK(REG, MASK)                               ((REG) ^=  (MASK))
38 #define GET_MASK(REG, MASK)                               ((REG) &   (MASK))
39
40 #define SET_REG(REG)                                      ((REG)  = ~(0u))
41 #define CLR_REG(REG)                                      ((REG)  =  (0u))
42 #define TOG_REG(REG)                                      ((REG) ^= ~(0u))
43
44 #define GET_BIT(REG, bit)                                 (((REG)>>(bit)) & 0x01u)
45 #define GET_NIB(REG, bit)                                 (((REG)>>(bit)) & 0x0Fu)
46 #define GET_BYTE(REG, bit)                                (((REG)>>(bit)) & 0xFFu)
47
48 #define ASSIGN_BIT(REG, bit, value)                      ((REG) = ((REG) & ~(0x01u<<(bit)))
| (((value) & 0x01u)<<(bit)))
49 #define ASSIGN_NIB(REG, bit, value)                      ((REG) = ((REG) & ~(0x0Fu<<(bit)))
| (((value) & 0x0Fu)<<(bit)))
50 #define ASSIGN_BYTE(REG, bit, value)                     ((REG) = ((REG) & ~(0xFFu<<(bit)))
| (((value) & 0xFFu)<<(bit)))
51
52 /*
53 #define ASSIGN_BIT(REG,bit,value)                         do{
\
54                                                            REG &= ~(0x01u<<bit);
\
55                                                            REG |=  ((value & 0x01u)<<bit);
\
56                                                           }while(0)
57 */
58
59 /*          bits together in an u8 register          */
60 #define CON_u8Bits(b7, b6, b5, b4, b3, b2, b1, b0)
\
61
(0b##b7##b6##b5##b4##b3##b2##b1##b0)
62
63 /*          bits together in an u16 register         */
64 #define CON_u16Bits(b15, b14, b13, b12, b11, b10, b9, b8, b7, b6, b5, b4, b3, b2, b1,
b0)   \
```

```
65
(0b##b15##b14##b13##b12##b11##b10##b9##b8##b7##b6##b5##b4##b3##b2##b1##b0)
66
67 /* ************************************************************************** */
68 /* ***************************** CONST SECTION ****************************** */
69 /* ************************************************************************** */
70
71 /* ************************************************************************** */
72 /* ***************************** VARIABLE SECTION ***************************** */
73 /* ************************************************************************** */
74
75 /* ************************************************************************** */
76 /* ***************************** FUNCTION SECTION ***************************** */
77 /* ************************************************************************** */
78
79
80 #endif /* LBIT_INT_H_ */
81 /***************************** E N D (LBIT_int.h) ******************************/
```

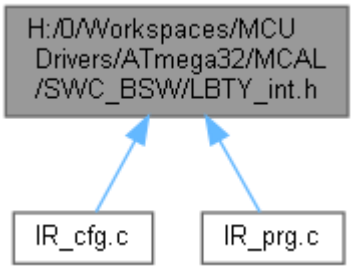
H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LBTY_int.h File Reference

```
#include <stdint.h>
```

Include dependency graph for LBTY_int.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- union LBTY_tuniPort8union LBTY_tuniPort16

Macros

- #define __IO volatile
- #define __O volatile
- #define __I volatile const
- #define LBTY_u8vidNOP()
- #define LBTY_NULL ((void *) 0U)
- #define LBTY_u8ZERO ((u8)0x00U)
- #define LBTY_u8MAX ((u8)0xFFU)
- #define LBTY_s8MAX ((s8)0x7F)
- #define LBTY_s8MIN ((s8)0x80)
- #define LBTY_u16ZERO ((u16)0x0000U)
- #define LBTY_u16MAX ((u16)0xFFFFU)
- #define LBTY_s16MAX ((u16)0x7FFF)
- #define LBTY_s16MIN ((u16)0x8000)
- #define LBTY_u32ZERO ((u32)0x00000000UL)
- #define LBTY_u32MAX ((u32)0xFFFFFFFFUL)
- #define LBTY_s32MAX ((u32)0x7FFFFFFFL)
- #define LBTY_s32MIN ((u32)0x80000000L)
- #define LBTY_u64ZERO ((u64)0x0000000000000000ULL)
- #define LBTY_u64MAX ((u64)0xFFFFFFFFFFFFFFFFULL)
- #define LBTY_s64MAX ((u64)0x7FFFFFFFFFFFFFFFLL)
- #define LBTY_s64MIN ((u64)0x8000000000000000LL)

Typedefs

- typedef uint8_t u8
- typedef uint16_t u16
- typedef uint32_t u32
- typedef uint64_t u64
- typedef int8_t s8
- typedef int16_t s16
- typedef int32_t s32
- typedef int64_t s64
- typedef float f32
- typedef double f64
- typedef u8 * pu8
- typedef u16 * pu16
- typedef u32 * pu32
- typedef u64 * pu64
- typedef s8 * ps8
- typedef s16 * ps16
- typedef s32 * ps32
- typedef s64 * ps64

Enumerations

- enum LBTY_tenuFlagStatus { LBTY_RESET = 0, LBTY_SET = !LBTY_RESET }
- enum LBTY_tenuBoolean { LBTY_TRUE = 0x55, LBTY_FALSE = 0xAA }
- enum LBTY_tenuErrorStatus { LBTY_OK = (u16)0, LBTY_NOK, LBTY_NULL_POINTER, LBTY_INDEX_OUT_OF_RANGE, LBTY_NO_MASTER_CHANNEL, LBTY_READ_ERROR, LBTY_WRITE_ERROR, LBTY_UNDEFINED_ERROR, LBTY_IN_PROGRESS }

Macro Definition Documentation

#define __I volatile const

#define __IO volatile

#define __O volatile

#define LBTY_NULL ((void *) 0U)

#define LBTY_s16MAX ((u16)0x7FFF)

#define LBTY_s16MIN ((u16)0x8000)

#define LBTY_s32MAX ((u32)0x7FFFFFFFL)

#define LBTY_s32MIN ((u32)0x80000000L)

#define LBTY_s64MAX ((u64)0x7FFFFFFFFFFFFFFFLL)

#define LBTY_s64MIN ((u64)0x8000000000000000LL)

#define LBTY_s8MAX ((s8)0x7F)

#define LBTY_s8MIN ((s8)0x80)

#define LBTY_u16MAX ((u16)0xFFFFU)

#define LBTY_u16ZERO ((u16)0x0000U)

#define LBTY_u32MAX ((u32)0xFFFFFFFFUL)

#define LBTY_u32ZERO ((u32)0x00000000UL)

#define LBTY_u64MAX ((u64)0xFFFFFFFFFFFFFFFFULL)

#define LBTY_u64ZERO ((u64)0x0000000000000000ULL)

#define LBTY_u8MAX ((u8)0xFFU)

#define LBTY_u8vidNOP()

#define LBTY_u8ZERO ((u8)0x00U)

Data Types Limitation

Typedef Documentation

typedef float f32

Standard Real Decimal number

typedef double f64

typedef s16* ps16

typedef s32* ps32

typedef s64* ps64

typedef s8* ps8

Standard Pointer to Signed Byte/Word/Long_Word

typedef u16* pu16

typedef u32* pu32

typedef u64* pu64

typedef u8* pu8

Standard Pointer to Unsigned Byte/Word/Long_Word

typedef int16_t s16

typedef int32_t s32

typedef int64_t s64

typedef int8_t s8

Standard Signed Byte/Word/Long_Word

typedef uint16_t u16

typedef uint32_t u32

typedef uint64_t u64

typedef uint8_t u8

Data Types New Definitions Standard Unsigned Byte/Word/Long_Word

---

Enumeration Type Documentation

enum LBTY_tenuBoolean

Boolean type

Enumerator:

LBTY_TRUE	
LBTY_FALSE	

```
96              {
97   LBTY_TRUE = 0x55,
98   LBTY_FALSE = 0xAA
99 } LBTY_tenuBoolean;
```

enum LBTY_tenuErrorStatus

Error Return type

Enumerator:

LBTY_OK	
LBTY_NOK	
LBTY_NULL_POINTER	
LBTY_INDEX_OUT_OF_RANGE	
LBTY_NO_MASTER_CHANNEL	
LBTY_READ_ERROR	
LBTY_WRITE_ERROR	
LBTY_UNDEFINED_ERROR	
LBTY_IN_PROGRESS	

```
102             {
103   LBTY_OK = (u16)0,
104   LBTY_NOK,
105   LBTY_NULL_POINTER,
106   LBTY_INDEX_OUT_OF_RANGE,
107   LBTY_NO_MASTER_CHANNEL,
108   LBTY_READ_ERROR,
109   LBTY_WRITE_ERROR,
110   LBTY_UNDEFINED_ERROR,
111   LBTY_IN_PROGRESS        /* Error is not available, wait for availability */
112 } LBTY_tenuErrorStatus;
```

enum LBTY_tenuFlagStatus

Flag Status type

Enumerator:

LBTY_RESET	
LBTY_SET	

```
90              {
91     LBTY_RESET = 0,
92     LBTY_SET = !LBTY_RESET
93 } LBTY_tenuFlagStatus;
```

LBTY_int.h

Go to the documentation of this file.

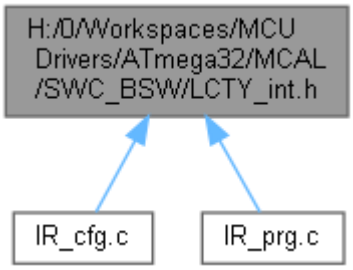
```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name   : LBTY_int.h                                                  */
5 /* Author      : MAAM                                                        */
6 /* Version     : v01                                                         */
7 /* date        : Mar 23, 2023                                                */
8 /* description : Basic Library                                               */
9 /* ************************************************************************* */
10 /* ************************ HEADER FILES INCLUDES ************************  */
11 /* ************************************************************************* */
12
13 #ifndef _LBTY_INT_H_
14 #define _LBTY_INT_H_
15
16 #include <stdint.h>
17
18 /* ************************************************************************* */
19 /* ****************************** TYPE_DEF SECTION ************************* */
20 /* ************************************************************************* */
21
24 typedef uint8_t     u8 ;
25 typedef uint16_t    u16;
26 typedef uint32_t    u32;
27 typedef uint64_t    u64;
28
30 typedef int8_t      s8 ;
31 typedef int16_t     s16;
32 typedef int32_t     s32;
33 typedef int64_t     s64;
34
36 typedef float       f32;
37 typedef double      f64;
38
40 typedef u8*         pu8 ;
41 typedef u16*        pu16;
42 typedef u32*        pu32;
43 typedef u64*        pu64;
44
46 typedef s8*         ps8 ;
47 typedef s16*        ps16;
48 typedef s32*        ps32;
49 typedef s64*        ps64;
50
51 /* ************************************************************************* */
52 /* ************************** MACRO/DEFINE SECTION ************************* */
53 /* ************************************************************************* */
54
55 /*****************************************************************************/
56 #define __IO    volatile
57 #define __O     volatile
58 #define __I     volatile const
59 /*****************************************************************************/
60
61 #define LBTY_u8vidNOP()
62 #define LBTY_NULL       ((void *) 0U)
63
65 #define LBTY_u8ZERO     ((u8)0x00U)
66 #define LBTY_u8MAX      ((u8)0xFFU)
67 #define LBTY_s8MAX      ((s8)0x7F )
68 #define LBTY_s8MIN      ((s8)0x80 )
69
70 #define LBTY_u16ZERO    ((u16)0x0000U)
71 #define LBTY_u16MAX     ((u16)0xFFFFU)
72 #define LBTY_s16MAX     ((u16)0x7FFF )
73 #define LBTY_s16MIN     ((u16)0x8000 )
74
75 #define LBTY_u32ZERO    ((u32)0x00000000UL)
76 #define LBTY_u32MAX     ((u32)0xFFFFFFFFUL)
77 #define LBTY_s32MAX     ((u32)0x7FFFFFFFL )
78 #define LBTY_s32MIN     ((u32)0x80000000L )
79
```

```
80 #define LBTY_u64ZERO     ((u64)0x0000000000000000ULL)
81 #define LBTY_u64MAX      ((u64)0xFFFFFFFFFFFFFFFFULL)
82 #define LBTY_s64MAX      ((u64)0x7FFFFFFFFFFFFFFFLL )
83 #define LBTY_s64MIN      ((u64)0x8000000000000000LL )
84
85 /* ********************************************************************************* */
86 /* ****************************** ENUM SECTION ************************************* */
87 /* ********************************************************************************* */
88
90 typedef enum {
91     LBTY_RESET = 0,
92     LBTY_SET = !LBTY_RESET
93 } LBTY_tenuFlagStatus;
94
96 typedef enum {
97   LBTY_TRUE = 0x55,
98   LBTY_FALSE = 0xAA
99 } LBTY_tenuBoolean;
100
102 typedef enum {
103   LBTY_OK = (u16)0,
104   LBTY_NOK,
105   LBTY_NULL_POINTER,
106   LBTY_INDEX_OUT_OF_RANGE,
107   LBTY_NO_MASTER_CHANNEL,
108   LBTY_READ_ERROR,
109   LBTY_WRITE_ERROR,
110   LBTY_UNDEFINED_ERROR,
111   LBTY_IN_PROGRESS         /* Error is not available, wait for availability */
112 } LBTY_tenuErrorStatus;
113
114 /* ********************************************************************************* */
115 /* ****************************** STRUCT SECTION ************************************* */
116 /* ********************************************************************************* */
117
119 typedef union {
120   struct {
121     u8 m_u8b0 :1;       // LSB
122     u8 m_u8b1 :1;
123     u8 m_u8b2 :1;
124     u8 m_u8b3 :1;
125     u8 m_u8b4 :1;
126     u8 m_u8b5 :1;
127     u8 m_u8b6 :1;
128     u8 m_u8b7 :1;       // MSB
129   } sBits;
130   u8 u_u8Byte;
131 } LBTY_tuniPort8;
132
133 typedef union {
134   struct  {
135     u8 m_u8b0  :1;      // LSB
136     u8 m_u8b1  :1;
137     u8 m_u8b2  :1;
138     u8 m_u8b3  :1;
139     u8 m_u8b4  :1;
140     u8 m_u8b5  :1;
141     u8 m_u8b6  :1;
142     u8 m_u8b7  :1;
143     u8 m_u8b8  :1;
144     u8 m_u8b9  :1;
145     u8 m_u8b10 :1;
146     u8 m_u8b11 :1;
147     u8 m_u8b12 :1;
148     u8 m_u8b13 :1;
149     u8 m_u8b14 :1;
150     u8 m_u8b15 :1;      // MSB
151   } sBits;
152   struct {
153     u8 m_u8low;
154     u8 m_u8high;
155   } sBytes;
156   u16 u_u16Word;
157 } LBTY_tuniPort16;
158
159 /* ********************************************************************************* */
160 /* ***************************** FUNCTION SECTION ************************************ */
```

```
161 /* ************************************************************************** */
162
163
164 #endif /* _LBTY_INT_H_ */
165 /*************************** E N D (LBTY_int.h) ****************************/
```

H:/0/Workspaces/MCU Drivers/ATmega32/MCAL/SWC_BSW/LCTY_int.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define LCTY_PROGMEM __attribute__((__progmem__))
- #define LCTY_PURE __attribute__((__pure__))
- #define LCTY_INLINE __attribute__((always_inline)) static inline
- #define LCTY_INTERRUPT __attribute__((interrupt))
- #define CTY_PACKED __attribute__((__packed__))
- #define LCTY_CONST __attribute__((__const__))
- #define LCTY_DPAGE __attribute__((dp))
- #define LCTY_NODPAGE __attribute__((nodp))
- #define LCTY_SECTION(section) __attribute__((section(# section)))
- #define LCTY_ASM(cmd) __asm__ __volatile__ (# cmd ::)

Macro Definition Documentation

#define CTY_PACKED __attribute__((__packed__))

#define LCTY_ASM(cmd) __asm__ __volatile__ (# cmd ::)

#define LCTY_CONST __attribute__((__const__))

#define LCTY_DPAGE __attribute__((dp))

#define LCTY_INLINE __attribute__((always_inline)) static inline

#define LCTY_INTERRUPT __attribute__((interrupt))

#define LCTY_NODPAGE __attribute__((nodp))

#define LCTY_PROGMEM __attribute__((__progmem__))

#define LCTY_PURE __attribute__((__pure__))

#define LCTY_SECTION(section) __attribute__((section(# section)))

LCTY_int.h

Go to the documentation of this file.

```
1 /*
2 /* ********************** FILE DEFINITION SECTION ************************** */
3 /* ************************************************************************* */
4 /* File Name   : LCTY_int.h                                                  */
5 /* Author      : MAAM                                                        */
6 /* Version     : v00                                                         */
7 /* date        : Apr 26, 2023                                                */
8 /* description : Compiler Library                                            */
9 /* ************************************************************************* */
10 /* ************************ HEADER FILES INCLUDES ************************  */
11 /* ************************************************************************* */
12
13 #ifndef LCTY_INT_H_
14 #define LCTY_INT_H_
15
16 /* ************************************************************************* */
17 /* ********************** TYPE_DEF/STRUCT/ENUM SECTION ********************** */
18 /* ************************************************************************* */
19
20 /* ************************************************************************* */
21 /* ************************** MACRO/DEFINE SECTION ************************** */
22 /* ************************************************************************* */
23
24 /* prog memory attribute */
25 #define LCTY_PROGMEM                __attribute__((__progmem__))
26
27 /* pure attribute */
28 #define LCTY_PURE                   __attribute__((__pure__))
29
30 /* Abstraction for inlining */
31 //#define LCTY_INLINE               static inline
32 #define LCTY_INLINE                 __attribute__((always_inline)) static inline
33
34 /* define function as interrupt handler */
35 #define LCTY_INTERRUPT              __attribute__((interrupt))
36
37 /* Memory packed to pass Memory padding */
38 #define CTY_PACKED                  __attribute__((__packed__))
39
40 /* Const attribute */
41 #define LCTY_CONST                  __attribute__((__const__))
42
43 /* place variable in direct page */
44 #define LCTY_DPAGE                  __attribute__((dp))
45
46 /* do not place variable in direct page */
47 #define LCTY_NODPAGE                __attribute__((nodp))
48
49 /* Sections */
50 #define LCTY_SECTION(section)       __attribute__((section( # section)))
51
52 /* Abstraction for assembly command */
53 # define LCTY_ASM(cmd)              __asm__ __volatile__ ( # cmd ::)
54
55 /* ************************************************************************* */
56 /* ***************************** CONST SECTION ***************************** */
57 /* ************************************************************************* */
58
59 /* ************************************************************************* */
60 /* ***************************** VARIABLE SECTION ************************** */
61 /* ************************************************************************* */
62
63 /* ************************************************************************* */
64 /* ***************************** FUNCTION SECTION ************************** */
65 /* ************************************************************************* */
66
67
68 #endif /* LCTY_INT_H_ */
69 /*************************** E N D (LCTY_int.h) ****************************/
```