

Time multiplex ACuc-rf101  
V2.1B

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>ADuc-RF-101 Time multiplex</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Build . . . . .	1
1.2.1	Step 1: Opening project in Keil . . . . .	1
<b>2</b>	<b>Bug List</b>	<b>3</b>
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>File Documentation</b>	<b>7</b>
4.1	C:/Users/Pepa/Documents/Keil/Radio/Aduc_rf101_my_sw/Projects/Time-multiplex-ADuC-rf101_↔ v2/src/Master.c File Reference . . . . .	7
4.1.1	Detailed Description . . . . .	9
4.1.2	Macro Definition Documentation . . . . .	10
4.1.2.1	LED_OFF . . . . .	10
4.1.2.2	LED_ON . . . . .	10
4.1.2.3	UART_BUFFER_DEEP . . . . .	10
4.1.3	Function Documentation . . . . .	10
4.1.3.1	copyBufferToMemory(void) . . . . .	10
4.1.3.2	dma_printf(const char *format,...) . . . . .	11
4.1.3.3	DMA_UART_RX_Int_Handler() . . . . .	12
4.1.3.4	DMA_UART_TX_Int_Handler(void) . . . . .	13
4.1.3.5	dmaSend(unsigned char *buff, int len) . . . . .	13
4.1.3.6	flushBufferedPackets(void) . . . . .	14

4.1.3.7	GP_Tmr0_Int_Handler(void)	15
4.1.3.8	GP_Tmr1_Int_Handler(void)	16
4.1.3.9	ifMissPktGet(void)	16
4.1.3.10	initializeNewSlot(void)	17
4.1.3.11	ledInit(void)	17
4.1.3.12	main(void)	18
4.1.3.13	radiolnit(void)	19
4.1.3.14	radioRecieve(void)	19
4.1.3.15	radioSend(char *buff, unsigned char len)	20
4.1.3.16	receivePackets(void)	21
4.1.3.17	rf_printf(const char *format,...)	22
4.1.3.18	SetInterruptPriority(void)	23
4.1.3.19	setSynnicTimer(void)	24
4.1.3.20	synchronize(void)	24
4.1.3.21	UART_Int_Handler()	25
4.1.3.22	uartInit(void)	26
4.1.3.23	validPacket(void)	26
4.1.3.24	zeroPacket(void)	27
4.1.4	Variable Documentation	28
4.1.4.1	actualPacket	28
4.1.4.2	actualRxBuffer	28
4.1.4.3	actualTxBuffer	28
4.1.4.4	Buffer	28
4.1.4.5	dmaTx_flag	28
4.1.4.6	dmaTxBuffer	28
4.1.4.7	dmaTxPkt	28
4.1.4.8	dmaTxPtr	28
4.1.4.9	dmaTxSlv	28
4.1.4.10	firstRxPkt	28
4.1.4.11	flush_flag	29

4.1.4.12	<a href="#">lastRadioTransmitBuffer</a>	29
4.1.4.13	<a href="#">lenghtOfPkt</a>	29
4.1.4.14	<a href="#">numOfPkt</a>	29
4.1.4.15	<a href="#">PktLen</a>	29
4.1.4.16	<a href="#">pktMemory</a>	29
4.1.4.17	<a href="#">RIE_Response</a>	30
4.1.4.18	<a href="#">RSSI</a>	30
4.1.4.19	<a href="#">RX_flag</a>	30
4.1.4.20	<a href="#">rxBuffer</a>	30
4.1.4.21	<a href="#">rxUARTcount</a>	30
4.1.4.22	<a href="#">send</a>	30
4.1.4.23	<a href="#">slave_ID</a>	30
4.1.4.24	<a href="#">sync_flag</a>	30
4.1.4.25	<a href="#">sync_wait</a>	30
4.1.4.26	<a href="#">TX_flag</a>	30
4.2	<a href="#">C:/Users/Pepa/Documents/Keil/Radio/Aduc_rf101_my_sw/Projects/Time-multiplex-ADuC-rf101_v2/src/settings.h File Reference</a>	31
4.2.1	<a href="#">Detailed Description</a>	32
4.2.2	<a href="#">Disclaimer</a>	33
4.2.3	<a href="#">Macro Definition Documentation</a>	33
4.2.3.1	<a href="#">BINARY_MODE</a>	33
4.2.3.2	<a href="#">CHAR_OFFSET</a>	34
4.2.3.3	<a href="#">DATA_WHITENING</a>	34
4.2.3.4	<a href="#">DEBUG_MESSAGES</a>	34
4.2.3.5	<a href="#">HEAD_FORMAT</a>	35
4.2.3.6	<a href="#">HEAD_LENGHT</a>	35
4.2.3.7	<a href="#">LEN_OF_RX_PKT</a>	35
4.2.3.8	<a href="#">NUM_OF_PACKETS_IN_MEMORY</a>	35
4.2.3.9	<a href="#">NUM_OF_SLAVE</a>	36
4.2.3.10	<a href="#">PA_TYPE</a>	36
4.2.3.11	<a href="#">PACKET_MEMORY_DEPTH</a>	36

4.2.3.12	RADIO_CFG . . . . .	36
4.2.3.13	RADIO_FREQUENCY . . . . .	37
4.2.3.14	RADIO_MANCHASTER . . . . .	37
4.2.3.15	RADIO_MODULATION . . . . .	38
4.2.3.16	RADIO_POWER . . . . .	38
4.2.3.17	RETRANSMISION . . . . .	38
4.2.3.18	RETRANSMISION_ID . . . . .	39
4.2.3.19	RX_STREAM . . . . .	39
4.2.3.20	SEND_HEAD . . . . .	39
4.2.3.21	SIMULATE_RETX . . . . .	39
4.2.3.22	SLAVE_ID . . . . .	39
4.2.3.23	STRING_TERMINATOR . . . . .	39
4.2.3.24	SYNC_INTERVAL . . . . .	40
4.2.3.25	SYNC_PIN_HIGH . . . . .	40
4.2.3.26	SYNC_PIN_LOW . . . . .	40
4.2.3.27	SYNC_PIN_READ . . . . .	40
4.2.3.28	T_PROCESSING . . . . .	40
4.2.3.29	T_TIMEOUT . . . . .	40
4.2.3.30	THROUGHPUT_MEASURE . . . . .	41
4.2.3.31	TIME_SLOT_ID_MASTER . . . . .	41
4.2.3.32	TIME_SLOT_ID_SLAVE . . . . .	41
4.2.3.33	TX_STREAM . . . . .	41
4.2.3.34	UART_BAUD_RATE_MASTER . . . . .	42
4.2.3.35	UART_BAUD_RATE_SLAVE . . . . .	42
4.2.3.36	ZERO_PACKET . . . . .	42
4.3	C:/Users/Pepa/Documents/Keil/Radio/Aduc_rf101_my_sw/Projects/Time-multiplex-ADuC-rf101_↵ v2/src/Slave.c File Reference . . . . .	42
4.3.1	Detailed Description . . . . .	44
4.3.2	Macro Definition Documentation . . . . .	45
4.3.2.1	LED_OFF . . . . .	45
4.3.2.2	LED_ON . . . . .	45

4.3.3	Function Documentation	45
4.3.3.1	dma_printf(const char *format,...)	45
4.3.3.2	DMA_UART_RX_Int_Handler()	46
4.3.3.3	DMA_UART_TX_Int_Handler()	46
4.3.3.4	dmaSend(unsigned char *buff, int len)	46
4.3.3.5	GP_Tmr0_Int_Handler(void)	46
4.3.3.6	GP_Tmr1_Int_Handler(void)	46
4.3.3.7	HardFault_Handler(void)	47
4.3.3.8	led_init(void)	47
4.3.3.9	main(void)	47
4.3.3.10	radiolnit(void)	48
4.3.3.11	radioRecieve(void)	48
4.3.3.12	radioSend(char *buff, unsigned char len)	48
4.3.3.13	retransmit(void)	49
4.3.3.14	rf_printf(const char *format,...)	50
4.3.3.15	SetInterruptPriority(void)	50
4.3.3.16	setTimeToSync(unsigned int time)	51
4.3.3.17	transmit(void)	52
4.3.3.18	uart_init(void)	52
4.3.3.19	UART_Int_Handler(void)	53
4.3.4	Variable Documentation	53
4.3.4.1	actualRxBuffer	53
4.3.4.2	actualTxBuffer	53
4.3.4.3	Buffer	53
4.3.4.4	buffer_change_flag	53
4.3.4.5	debugTimer	53
4.3.4.6	i	53
4.3.4.7	j	53
4.3.4.8	lenghtOfPkt	54
4.3.4.9	my_slot	54
4.3.4.10	numOfPkt	54
4.3.4.11	PktLen	54
4.3.4.12	pktMemory	54
4.3.4.13	RIE_Response	54
4.3.4.14	RSSI	55
4.3.4.15	RX_flag	55
4.3.4.16	rxPktPtr	55
4.3.4.17	rxUARTbuffer	55
4.3.4.18	rxUARTcount	55
4.3.4.19	terminate_flag	55
4.3.4.20	TX_flag	55





# Chapter 1

## ADuc-RF-101 Time multiplex

### 1.1 Introduction

Program is using time multiplex for transmitting data from optionally number of Slaves to Master device via R↔F-channel with checking data integrity using hardware implement CRC and retransmitting lost data packets.

Debugged for ADucRF101MKxZ development kit

- Author: Peter Soltys
- Version: 2.1B
- Hardware: ADucRF101MKxZ
- Date: 07.02.2016
- Project: Time-multiplex-ADuc-RF101

DEV: Keil 5.1 Evaluation

- Note: v2.1B fixed synchronization and added binary mode

Compatible with program UWB - Coordinate Reader Deployment from Peter Mikula [uwb\\_coordinate\\_reader](#)

Program is able to find online on [Github](#)

### 1.2 Build

#### 1.2.1 Step 1: Opening project in Keil

etc...



## Chapter 2

# Bug List

### Global [BINARY\\_MODE](#)

not entirely verified (still bugs)

### Global [LEN\\_OF\\_RX\\_PKT](#)

constant lenght not entirely verified (still bugs)



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

C:/Users/Pepa/Documents/Keil/Radio/Aduc_rf101_my_sw/Projects/Time-multiplex-ADuC-rf101_↔ v2/src/ <a href="#">Master.c</a> Prigram used for Receiving data in time multiplex data are received via RF Interface and sendeds to UART working with <a href="#">Slave.c</a> tested on ADucRF101MKxZ . . . . .	7
C:/Users/Pepa/Documents/Keil/Radio/Aduc_rf101_my_sw/Projects/Time-multiplex-ADuC-rf101_↔ v2/src/ <a href="#">settings.h</a> Configurating file defining base settings . . . . .	31
C:/Users/Pepa/Documents/Keil/Radio/Aduc_rf101_my_sw/Projects/Time-multiplex-ADuC-rf101_↔ v2/src/ <a href="#">Slave.c</a> Prigram used for transmitting data in time multiplex data are received via UART and sendeds trouhgt radio interface during associated time slot working with <a href="#">Master.c</a> tested on ADucR↔ F101MKxZ . . . . .	42



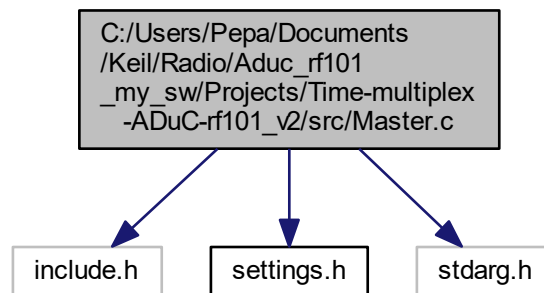
## Chapter 4

# File Documentation

### 4.1 C:/Users/Pepa/Documents/Keil/Radio/Aduc\_rf101\_my\_sw/Projects/Time-multiplex-↵ ADuC-rf101\_v2/src/Master.c File Reference

prigram used for Receiving data in time multiplex data are received via RF Interface and sended to UART working with [Slave.c](#) tested on ADucRF101MKxZ

```
#include "include.h"  
#include "settings.h"  
#include <stdarg.h>  
Include dependency graph for Master.c:
```



### Macros

- #define [LED\\_OFF](#) DioSet(pADI\_GP4,BIT2)
- #define [LED\\_ON](#) DioClr(pADI\_GP4,BIT2)
- #define [UART\\_BUFFER\\_DEEP](#) 50

## Functions

- void [DMA\\_UART\\_TX\\_Int\\_Handler](#) (void)  
*Interrupt handler managing sending content of pktMemory on UART with DMA.*
- void [uartInit](#) (void)  
*initialize uart port*
- void [ledInit](#) (void)  
*initialize port with led diode*
- void [setSynnicTimer](#) (void)  
*set general purpose timer0 for synchronization intervals*
- void [radiolnit](#) (void)  
*initialize Radio interface*
- void [dmaSend](#) (unsigned char \*buff, int len)  
*send one packet on UART with DMA controller*
- int [dma\\_printf](#) (const char \*format,...)  
*this function is equivalent to function printf from library stdio.h*
- void [radioSend](#) (char \*buff, unsigned char len)  
*send one packet trough radio interface*
- unsigned char [rf\\_printf](#) (const char \*format,...)  
*this function is equivalent to function printf from library stdio.h*
- char [radioRecieve](#) (void)  
*function receive one packet with radio interface*
- char [validPacket](#) (void)  
*validate received packt head in global variable (Buffer)*
- void [copyBufferToMemory](#) (void)  
*copy received packet to packet memory*
- void [ifMissPktGet](#) (void)  
*function check packet meory for missing packets, send rquest for retransmit losted packets and save returned packets*
- void [flushBufferedPackets](#) (void)  
*rotate memory and start sending received packets on UART with DMA*
- void [synchronize](#) (void)  
*send synchronization packets in constant time delays*
- char [zeroPacket](#) (void)  
*check Buffer if is zero packet*
- char [receivePackets](#) (void)  
*receive all packets of sendet time slot;*
- void [initializeNewSlot](#) (void)  
*Initialize variables for new ID slot.*
- void [SetInterruptPriority](#) (void)  
*Initialize interrupt priority.*
- int [main](#) (void)  
*main function of master program*
- void [GP\\_Tmr1\\_Int\\_Handler](#) (void)  
*Interrupt handler for measuring throughput.*
- void [GP\\_Tmr0\\_Int\\_Handler](#) (void)  
*Interrupt handler for synchronization timing.*
- void [DMA\\_UART\\_RX\\_Int\\_Handler](#) ()  
*Interrupt handler terminating DMA receiving transaction.*
- void [UART\\_Int\\_Handler](#) ()  
*Interrupt handler waiting for SYNC\$ word to set synchronization flag.*



- RIE\_Responses [RIE\\_Response](#) = RIE\_Success
- unsigned char [Buffer](#) [240]
- RIE\_U8 [PktLen](#)
- RIE\_S8 [RSSI](#)
- char [pktMemory](#) [2][[NUM\\_OF\\_PACKETS\\_IN\\_MEMORY](#)][[PACKET\\_MEMORY\\_DEPTH](#)]  
*memory to store all data to send/to receive*
- char [numOfPkt](#) [2] = {0,0}
- unsigned char [lenghtOfPkt](#) [2][[NUM\\_OF\\_PACKETS\\_IN\\_MEMORY](#)] = {0,0}
- unsigned char [actualPacket](#)
- signed char [actualRxBuffer](#) =0
- signed char [actualTxBuffer](#) =1
- char [lastRadioTransmitBuffer](#) [[PACKET\\_MEMORY\\_DEPTH](#)]
- char [dmaTxBuffer](#) [255]
- char [rxBuffer](#) [[UART\\_BUFFER\\_DEEP](#)]
- unsigned char [slave\\_ID](#) = 1
- signed char [send](#) =0
- signed char [TX\\_flag](#) =0
- signed char [RX\\_flag](#) =0
- signed char [flush\\_flag](#) =0
- signed char [sync\\_flag](#) = 0
- signed char [sync\\_wait](#) = 0
- signed char [firstRxPkt](#) = 0
- unsigned char [rxUARTcount](#) =0
- signed char [dmaTxSlv](#) =0
- signed char [dmaTxPkt](#) =0
- signed char [dmaTx\\_flag](#) =0
- char \* [dmaTxPtr](#)

#### 4.1.1 Detailed Description

prigram used for Receiving data in time multiplex data are received via RF Interface and sended to UART working with [Slave.c](#) tested on ADucRF101MKxZ

##### Version

V2.1B

##### Author

Peter Soltys

##### Date

febtuary 2016

##### Revision History:

- V1.1, July 2015 : initial version.
- V1.2, august 2015 : fully functional.
- V1.3, september 2015 : faster version with higher throughput
- V1.4, january 2016 : added synchronization
- V2.0, febtuary 2016 : new time multiplex conception
- V2.1, febtuary 2016 : fixed synchronization
- V2.1B, febtuary 2016 : Binary data packets (instead of strings with STRING\_TERMINATOR)

**Note**

: in radioeng.c was changed initial value from  
static RIE\_BOOL bPacketTx = RIE\_FALSE;  
static RIE\_BOOL bPacketRx = RIE\_FALSE;  
to  
static RIE\_BOOL bPacketTx = RIE\_TRUE;  
static RIE\_BOOL bPacketRx = RIE\_TRUE;

## 4.1.2 Macro Definition Documentation

### 4.1.2.1 #define LED\_OFF DioSet(pADI\_GP4,BIT2)

Definition at line 37 of file Master.c.

### 4.1.2.2 #define LED\_ON DioClr(pADI\_GP4,BIT2)

Definition at line 38 of file Master.c.

### 4.1.2.3 #define UART\_BUFFER\_DEEP 50

Definition at line 77 of file Master.c.

## 4.1.3 Function Documentation

### 4.1.3.1 void copyBufferToMemory ( void )

copy received packet to packet memory

**Precondition**

[radioInit\(\)](#) must be called before this function is called.  
[radioReceive\(\)](#) with returned 1  
[validPacket\(\)](#) with returned 1

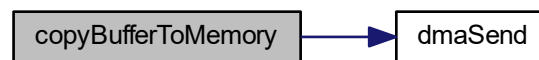
```
1 radioInit();
2 if (radioReceive())
3     if (validPacket()){
4         printf("packet was sucesfully received with correct packet head");
5         copyBufferToMemory();
6     }
```

Note

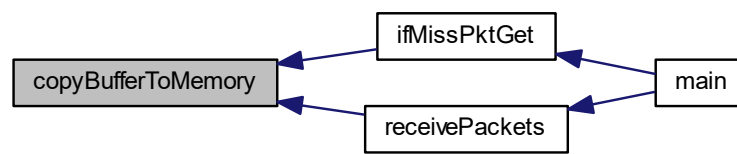
function save packet at place defined in packet head

Definition at line 426 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.2 int dma\_printf ( const char \* format, ... )

this function is equivalent to function printf from library stdio.h

##### Parameters

<i>format</i>	: {} pointer at string to formating output string
...	: {} additive parameters for formating string

##### Precondition

[uartInit\(\)](#) must be called before this can be called.

```

1 uartInit();
2 int num = 10;
3 dma_printf();
  
```

Note

output stream is managed with DMA controller after end of transmision is called DMA\_UART\_TX\_Int\_Handler

See also

[DMA\\_UART\\_TX\\_Int\\_Handler](#)

Returns

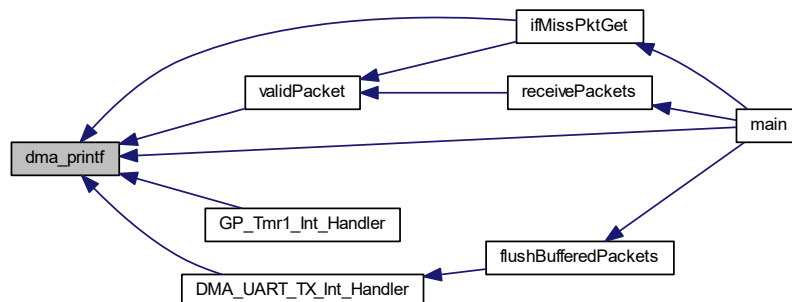
int number of sending chars (if == 0 error)

Definition at line 216 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.3 void DMA\_UART\_RX\_Int\_Handler ( )

Interrupt handler terminating DMA receiving transaction.

Interrupt handler terminating DMA receiving transaction if constant length of packet is set.

Note

not used

See also

[LEN\\_OF\\_RX\\_PKT](#)

Definition at line 766 of file Master.c.

4.1.3.4 void DMA\_UART\_TX\_Int\_Handler ( void )

Interrupt handler managing sending content of pktMemory on UART with DMA.

terminating DMA transaction of function dma\_send()

Note

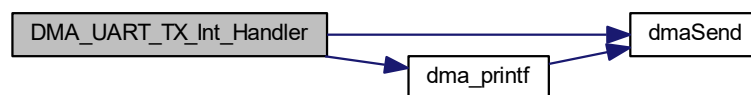
also is terminating DMA transaction of function dma\_send()

See also

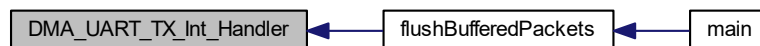
dma\_send()  
flushPackets()  
dma\_send()

Definition at line 716 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.5 void dmaSend ( unsigned char \* buff, int len )

send one packet on UART with DMA controller

Parameters

<i>buff</i>	:{} pointer to data to send
<i>len</i>	:{int range} number of bytes to send



Note

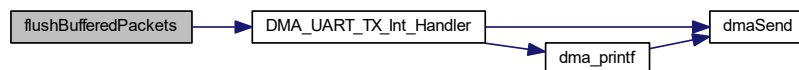
all managment about sending packets is in

See also

[DMA\\_UART\\_TX\\_Int\\_Handler\(\)](#)

Definition at line 503 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.7 void GP\_Tmr0\_Int\_Handler ( void )

Interrupt handler for synchronization timing.

See also

[synchronize\(\)](#)

Note

synchronization at falling edge handler manage also rising edge (hughe jitter)

See also

[setTimeToSync\(\)](#)

Definition at line 696 of file Master.c.

#### 4.1.3.8 void GP\_Tmr1\_Int\_Handler ( void )

Interrupt handler for measuring troughput.

##### Note

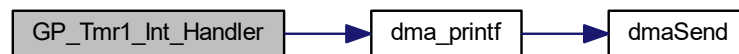
only if THROUGHPUT\_MEASURE is set in [settings.h](#)

##### See also

[settings.h](#)

Definition at line 673 of file Master.c.

Here is the call graph for this function:



#### 4.1.3.9 void ifMissPktGet ( void )

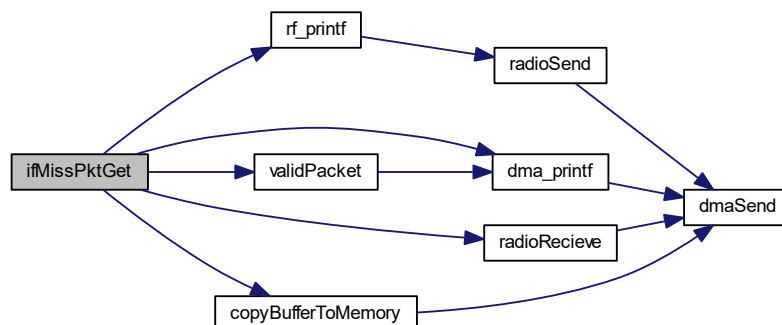
function check packet meory for missing packets, send rquest for retransmit losted packets and save returned packets

##### Precondition

[radioInit\(\)](#) must be called before this function is called.

Definition at line 447 of file Master.c.

Here is the call graph for this function:





Here is the caller graph for this function:

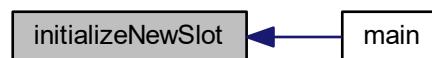


#### 4.1.3.10 void initializeNewSlot ( void )

Initialize variables for new ID slot.

Definition at line 606 of file Master.c.

Here is the caller graph for this function:



#### 4.1.3.11 void ledInit ( void )

initialize port with led diode

Definition at line 128 of file Master.c.

Here is the caller graph for this function:



#### 4.1.3.12 int main ( void )

main function of master program

main function of slave program

##### Note

Program is using time multiplex to receive data from 4 slave devices

##### Returns

int

##### Note

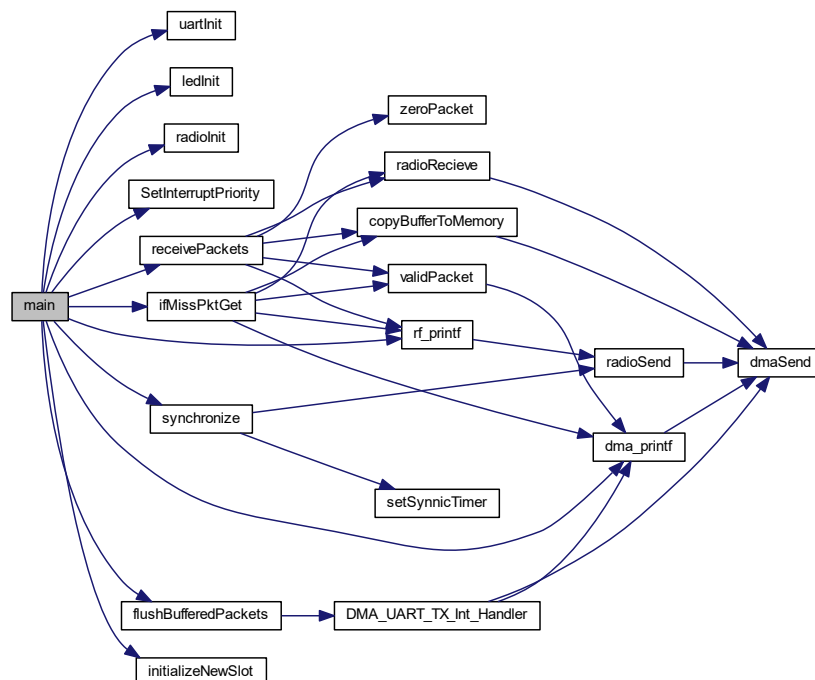
Program is using time multiplex to send data to master device

##### Returns

int

Definition at line 632 of file Master.c.

Here is the call graph for this function:



#### 4.1.3.13 void radioInit ( void )

initialize Radio interface

```
1 radioInit();
```

See also

[settings.h](#)

Note

see [settings.h](#) for radio configuration

Definition at line 158 of file Master.c.

Here is the caller graph for this function:



#### 4.1.3.14 char radioRecieve ( void )

function receive one packet with radio interface

Precondition

[radioInit\(\)](#) must be called before this function is called.

```
1 radioInit();
2 if (radioReceive())
3     printf("packet was received and read from radio interface");
4     printf(Buffer); //Buffer is global bufer for radio interface
5 else
6     printf("packet was not received correctly before timeout");
```

Note

function is also reding packet form radio interface to unsigned char Buffer[240]

**Returns**

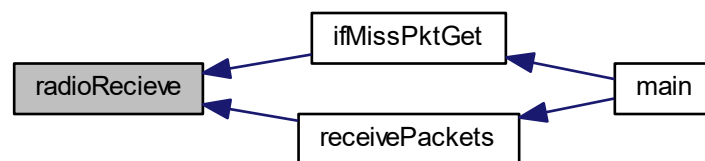
char 1 == packet received, 0 == packet was not received correctly before timeout

Definition at line 316 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.15 void radioSend ( char \* *buff*, unsigned char *len* )

send one packet trough radio interface

**Parameters**

<i>buff</i>	:{1-240} pointer at memory to be sended
<i>len</i>	:{0-240} number of bytes to be sended

**Precondition**

[radioInit\(\)](#) must be called before this function is called.

```

1 len=vsprintf(buff, format,args);//@see rf_printf();
2 if(len<240){//check max lenght
3   radioSend(buff,len+1);//send formatted packet

```

Note

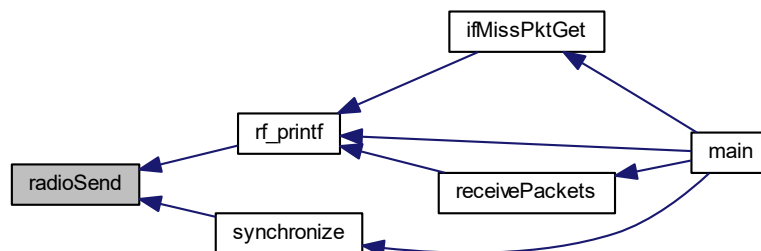
output stream is trough radio interface function is waiting until whole packet is trnsmitted

Definition at line 244 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.16 char receivePackets ( void )

receive all packets of sendet time slot;

#### Precondition

`radioInit()` must be called before this function is called.  
`rf_printf("1slot")` or equivalent should be called first.

```

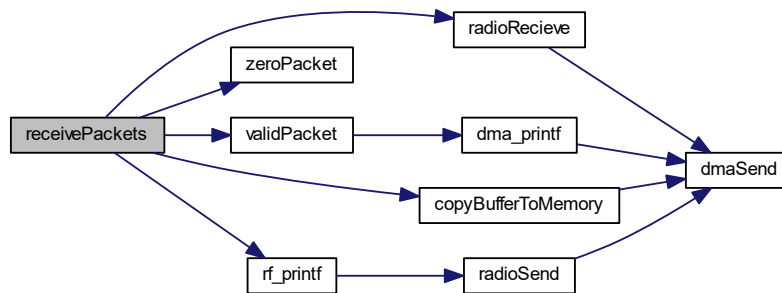
1 radioInit();
2 rf_printf("1slot");//slot identificator
3 if (receivePackets())
4     flushPacktes();//send all received packtes on UART
  
```

**Note**

function si also retransmitin slot ID packtes if no response

Definition at line 567 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.17 int rf\_printf ( const char \* *format*, ... )

this function is equivalent to function printf from library stdio.h

**Parameters**

<i>format</i>	: {} pointer at string to formating output string
...	: {} additive parameters for formating string

**Precondition**

`radioInoit()` must be called before this can be called.

```

1 radioInit();
2 int num = 10;
3 rf_printf();

```

Note

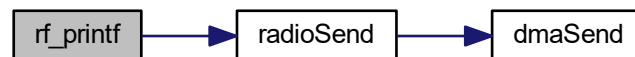
output stream is trough radio interface function is waiting until whole packet is transmited any one formated string (call) is sendd in one packet

#### Returns

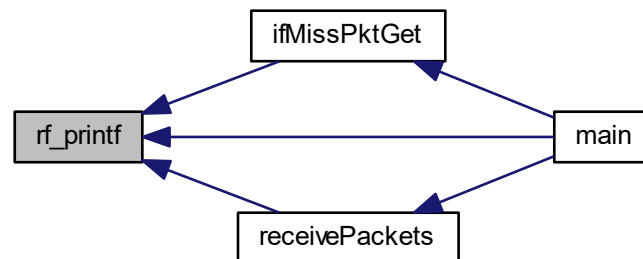
int number of sending chars (if == 0 error)

Definition at line 284 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.18 void SetInterruptPriority ( void )

Initialize interrupt priority.

Definition at line 619 of file Master.c.

Here is the caller graph for this function:



#### 4.1.3.19 void setSynnicTimer ( void )

set general purpose timer0 for synchronization intervals

##### See also

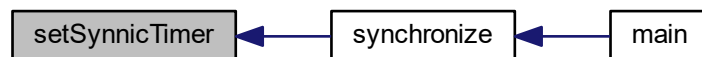
void [GP\\_Tmr1\\_Int\\_Handler](#) ()

##### Note

timer predivider factor = 256, processor clock, periodic mode

Definition at line 140 of file Master.c.

Here is the caller graph for this function:



#### 4.1.3.20 void synchronize ( void )

send synchronization packets in constant time delays

##### Precondition

[radioInit\(\)](#) must be called before this function is called.

##### Note

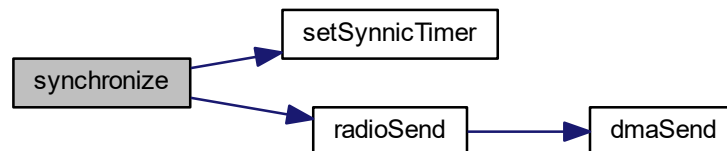
function flags controlled by



[GP\\_Tmr0\\_Int\\_Handler](#) and  
[UART\\_Int\\_Handler](#)

Definition at line 525 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.21 void UART\_Int\_Handler ( void )

Interrupt handler waiting for SYNC\$ word to set synchronization flag.

Interrupt handler managing store received data trough UART.

##### Note

function is taken from example UARTLoopback.c and modified

##### See also

[synchronize\(\)](#)

- string mode = appedning chars untill STRING\_TERMINATOR is received
- binary mode = appending untill buffer is full == 240 chars

##### See also

[STRING\\_TERMINATOR](#)

Definition at line 781 of file Master.c.

#### 4.1.3.22 void uartInit ( void )

initialize uart port

```
1 uartInit();
```

##### Note

speed UART\_BAUD\_RATE\_MASTER baud 8 bits one stop bit output port P1.0/P1.1

Definition at line 113 of file Master.c.

Here is the caller graph for this function:



#### 4.1.3.23 char validPacket ( void )

validate received packet head in global variable (Buffer)

##### Precondition

`radioInit()` must be called before this function is called.  
`radioReceive()` with returned 1

```
1 radioInit();
2 if (radioReceive())
3     if (validPacket())
4         printf("packet was sucesfully received with correct packet head");
```

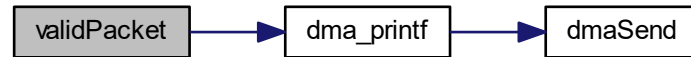
##### Note

function send messages about incorrect packet on UART

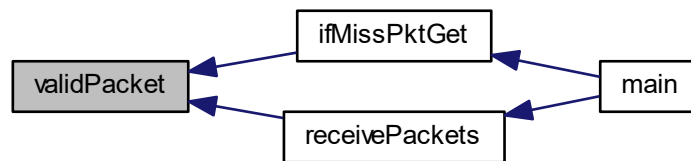
char 1 == valid packet head, 0 == invalid packet head

Definition at line 371 of file Master.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.3.24 char zeroPacket ( void )

check Buffer if is zero packet

##### Note

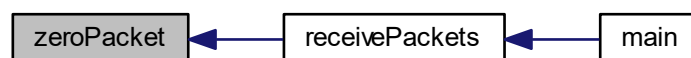
zero packet mean that slave have no buffered packets

##### Returns

1 == zero packet, 0 == no zero packet

Definition at line 547 of file Master.c.

Here is the caller graph for this function:



#### 4.1.4 Variable Documentation

##### 4.1.4.1 unsigned char actualPacket

Definition at line 71 of file Master.c.

##### 4.1.4.2 signed char actualRxBuffer =0

Definition at line 73 of file Master.c.

##### 4.1.4.3 signed char actualTxBuffer =1

Definition at line 73 of file Master.c.

##### 4.1.4.4 unsigned char Buffer[240]

Definition at line 43 of file Master.c.

##### 4.1.4.5 signed char dmaTx\_flag =0

Definition at line 91 of file Master.c.

##### 4.1.4.6 char dmaTxBuffer[255]

Definition at line 76 of file Master.c.

##### 4.1.4.7 signed char dmaTxPkt =0

Definition at line 91 of file Master.c.

##### 4.1.4.8 char\* dmaTxPtr

Definition at line 92 of file Master.c.

##### 4.1.4.9 signed char dmaTxSlv =0

Definition at line 91 of file Master.c.

##### 4.1.4.10 signed char firstRxPkt = 0

Definition at line 87 of file Master.c.

4.1.4.11 signed char flush\_flag =0

Definition at line 84 of file Master.c.

4.1.4.12 char lastRadioTransmitBuffer[PACKET\_MEMORY\_DEPTH]

Definition at line 75 of file Master.c.

4.1.4.13 unsigned char lenghtOfPkt[2][NUM\_OF\_PACKETS\_IN\_MEMORY] = {0,0}

Definition at line 69 of file Master.c.

4.1.4.14 char numOfPkt[2] = {0,0}

Definition at line 67 of file Master.c.

4.1.4.15 RIE\_U8 PktLen

Definition at line 44 of file Master.c.

4.1.4.16 char pktMemory[2][NUM\_OF\_PACKETS\_IN\_MEMORY][PACKET\_MEMORY\_DEPTH]

memory to store all data to send/to receive

#### Parameters

-	level
	<ul style="list-style-type: none"><li>• packet num</li><li>• packet data</li></ul>

#### Note

pktMemory is 2 levels deep puspose is changing in circle 0 actual receiving buffer 1 actual sending buffer for pointing are used flags : actualRxBuffer, actualTxBuffer  
size of mermory is restricted because ADuc rf101 have only 16KBytes SRAM  
UART must be much faster(tested on 128000 baud rate) than RF-link (to release memory)

#### See also

[actualRxBuffer](#)  
[actualTxBuffer](#)

Definition at line 64 of file Master.c.

#### 4.1.4.17 RIE\_Responses RIE\_Response = RIE\_Success

Definition at line 42 of file Master.c.

#### 4.1.4.18 RIE\_S8 RSSI

Definition at line 45 of file Master.c.

#### 4.1.4.19 signed char RX\_flag =0

Definition at line 84 of file Master.c.

#### 4.1.4.20 char rxBuffer[UART\_BUFFER\_DEEP]

Definition at line 78 of file Master.c.

#### 4.1.4.21 unsigned char rxUARTcount =0

Definition at line 88 of file Master.c.

#### 4.1.4.22 signed char send =0

Definition at line 81 of file Master.c.

#### 4.1.4.23 unsigned char slave\_ID = 1

Definition at line 80 of file Master.c.

#### 4.1.4.24 signed char sync\_flag = 0

Definition at line 85 of file Master.c.

#### 4.1.4.25 signed char sync\_wait = 0

Definition at line 86 of file Master.c.

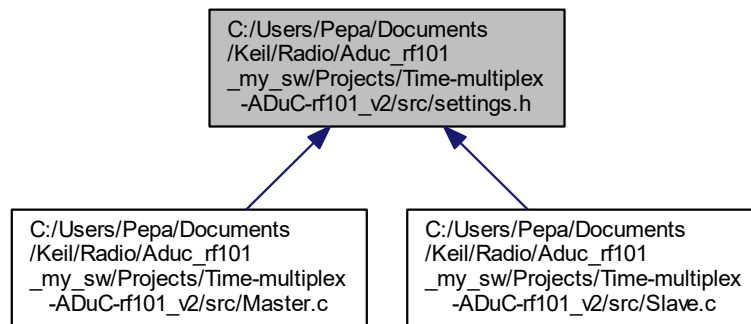
#### 4.1.4.26 signed char TX\_flag =0

Definition at line 84 of file Master.c.

## 4.2 C:/Users/Pepa/Documents/Keil/Radio/Aduc\_rf101\_my\_sw/Projects/Time-multiplex-ADuC-rf101\_v2/src/settings.h File Reference

configuring file defining base settings

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



## Macros

- #define **RADIO\_CFG** DR\_300\_0kbps\_Dev75\_0kHz
- #define **RADIO\_MODULATION** GFSK\_Modulation  
*Radio Transmitter Modulation Type.*
- #define **RADIO\_FREQUENCY** 433920000  
*Frequency for radio communications.*
- #define **PA\_TYPE** DifferentialPA  
*PA Type for Radio Transmission.*
- #define **RADIO\_POWER** PowerLevel15  
*Power Level for Radio Transmission.*
- #define **DATA\_WHITENING** RIE\_FALSE  
*Enable or Disable Data Whitening of payload data.*
- #define **RADIO\_MANCHASTER** RIE\_FALSE  
*Enable or Disable Manchester Encoding of payload data.*
- #define **BINARY\_MODE** 0  
*Enable or Disable binary mode.*
- #define **STRING\_TERMINATOR** '\$'  
*char witch terminate all received packets*
- #define **PACKET\_MEMORY\_DEPTH** 240  
*maximal memory (packet) depth*
- #define **NUM\_OF\_PACKETS\_IN\_MEMORY** 20  
*maximal number of packets to send/receive*
- #define **NUM\_OF\_SLAVE** 4  
*number of expected slave devices*
- #define **LEN\_OF\_RX\_PKT** 0  
*lenght of received packets from UART*

- #define CHAR\_OFFSET '0'  
*offset in numbers of head*
- #define SYNC\_INTERVAL 200  
*time interval to interrupt for synchronization*
- #define T\_TIMEOUT 50000  
*max time(number of increments) to response of requested devide*
- #define UART\_BAUD\_RATE\_MASTER 128000  
*UART baudrate with is using master.*
- #define TIME\_SLOT\_ID\_MASTER "%dslot",slave\_ID  
*string defining format of "ID slot" packet*
- #define RETRANSMISION 3  
*number of retransmission trying until slave is marked as not responding*
- #define UART\_BAUD\_RATE\_SLAVE 128000  
*UART baudrate with is using master.*
- #define TIME\_SLOT\_ID\_SLAVE "2slot"  
*format of slot identificator*
- #define ZERO\_PACKET "200"  
*format of zero packet*
- #define RETRANSMISION\_ID "2RE"  
*format of retransmision packet*
- #define SLAVE\_ID 2  
*number of actual slave*
- #define HEAD LENGHT 3  
*lenght of head in bytes*
- #define HEAD\_FORMAT "%d%c%c",SLAVE\_ID,txPkt,numOfPackets[actualTxBuffer]-1
- #define T\_PROCESSING 0  
*appended time(number of increments) after transmiton to procesing on master*
- #define SYNC\_PIN\_HIGH DioSet(pADI\_GP4,BIT2)
- #define SYNC\_PIN\_LOW DioClr(pADI\_GP4,BIT2)
- #define SYNC\_PIN\_READ DioRd(pADI\_GP4)&0x04
- #define SIMULATE\_RETX 0  
*sending retransmitin message to test*
- #define DEBUG\_MESSAGES 0  
*stream of mesages to UART*
- #define RX\_STREAM 0  
*stream of redeived data to UART*
- #define TX\_STREAM 0  
*stream of transmited data to UART*
- #define SEND\_HEAD 0  
*send also heads of packets on UART*
- #define THROUGHPUT\_MEASURE 0  
*measured data troughput*

#### 4.2.1 Detailed Description

configurating file defining base settings

Version

V2.1B



Peter Soltys

Date

february 2016

Note

: in radioeng.c was changed initial value from  
static RIE\_BOOL bPacketTx = RIE\_FALSE;  
static RIE\_BOOL bPacketRx = RIE\_FALSE;  
to  
static RIE\_BOOL bPacketTx = RIE\_TRUE;  
static RIE\_BOOL bPacketRx = RIE\_TRUE;

## 4.2.2 Disclaimer

THIS SOFTWARE IS PROVIDED BY BC PETER SOLTYS. "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED. IN NO EVENT SHALL BC PETER SOLTYS. BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

YOU ASSUME ANY AND ALL RISK FROM THE USE OF THIS CODE OR SUPPORT FILE.

IT IS THE RESPONSIBILITY OF THE PERSON INTEGRATING THIS CODE INTO AN APPLICATION TO ENSURE THAT THE RESULTING APPLICATION PERFORMS AS REQUIRED AND IS SAFE.

## 4.2.3 Macro Definition Documentation

### 4.2.3.1 #define BINARY\_MODE 0

Enable or Disable binary mode.

Binary mode is transmitting data packets in maximal lenght 240 All received data in slave are stored together

In (normal) string mode packets are terminated with STRING\_TERMINATOR

**Bug** not entirely verified (still bugs)

Parameters

<i>bool</i>	:{1 , 0} <ul style="list-style-type: none"><li>• 1 if Binary mode</li><li>• 0 if String mode.</li></ul>
-------------	---

Definition at line 137 of file settings.h.

#### 4.2.3.2 `#define CHAR_OFFSET '0'`

offset in numbers of head

##### Note

offset in numbers of head, because 0x00 is defined like end of string => working with packets as with strings

##### Parameters

<i>char</i>	:{'0'}
-------------	--------

Definition at line 183 of file settings.h.

#### 4.2.3.3 `#define DATA_WHITENING RIE_FALSE`

Enable or Disable Data Whitening of payload data.

Data whitening can be employed to avoid long runs of 1s or 0s in the transmitted data stream.

This ensures sufficient bit transitions in the packet, which aids in receiver clock and data recovery because the encoding breaks up long runs of 1s or 0s in the transmit packet.

The data, excluding the preamble and sync word, is automatically whitened before transmission by XORing the data with an 8-bit pseudorandom sequence.

At the receiver, the data is XORed with the same pseudorandom sequence, thereby reversing the whitening.

The linear feedback shift register polynomial used is  $x^7 + x^1 + 1$ .

##### Parameters

<i>bEnable</i>	:{RIE_FALSE, RIE_TRUE} <ul style="list-style-type: none"> <li>• RIE_TRUE if Manchester Encoding is to be enabled.</li> <li>• RIE_FALSE if disabled.</li> </ul>
----------------	--

Definition at line 104 of file settings.h.

#### 4.2.3.4 `#define DEBUG_MESSAGES 0`

stream of messages to UART

Definition at line 270 of file settings.h.

4.2.3.5 #define HEAD\_FORMAT "%d%c%c",SLAVE\_ID,txPkt,numOfPackets[actualTxBuffer]-1

Definition at line 255 of file settings.h.

4.2.3.6 #define HEAD\_LENHT 3

length of head in bytes

Definition at line 253 of file settings.h.

4.2.3.7 #define LEN\_OF\_RX\_PKT 0

length of received packets from UART

#### Note

macro set greatness of packet memory

**Bug** constant length not entirely verified (still bugs)

#### Parameters

<i>bool</i>	:{0 , 240} <ul style="list-style-type: none"><li>• 0 if variable length of packets</li><li>• 1-240 constant length of packets</li></ul>
-------------	---

Definition at line 175 of file settings.h.

4.2.3.8 #define NUM\_OF\_PACKETS\_IN\_MEMORY 20

maximal number of packets to send/receive

#### Note

macro set greatness of packet memory

#### Parameters

<i>number</i>	of packets :{1 , 20}
---------------	----------------------

Definition at line 157 of file settings.h.

#### 4.2.3.9 #define NUM\_OF\_SLAVE 4

number of expected slave devices

##### Note

number is restricted by size of memory macro set greatness of packet memory

##### Parameters

<i>number</i>	of slave devices :{1 , 10} 4
---------------	------------------------------

Definition at line 165 of file settings.h.

#### 4.2.3.10 #define PA\_TYPE DifferentialPA

PA Type for Radio Transmission.

##### Parameters

<i>PAType</i>	:{DifferentialPA, SingleEndedPA} Select Single Ended or Differential PA Type
---------------	--

Definition at line 70 of file settings.h.

#### 4.2.3.11 #define PACKET\_MEMORY\_DEPTH 240

maximal memory (packet) depth

##### Note

macro set greatness of packet memory

##### Parameters

<i>memory</i>	depth :{0 , 240}
---------------	------------------

Definition at line 150 of file settings.h.

#### 4.2.3.12 #define RADIO\_CFG DR\_300\_0kbps\_Dev75\_0kHz

Parameters

<i>BaseConfig</i>	<pre>:{DR_1_0kbps_Dev10_0kHz , DR_38_4kbps_Dev20kHz ,DR_300_0kbps_Dev75_0kHz }</pre> <ul style="list-style-type: none"><li>• DR_1_0kbps_Dev10_0kHz Base configuration of 1 kbps datarate, 10.0 kHz frequency deviation.</li><li>• DR_38_4kbps_Dev20kHz Base configuration of 38.4 kbps datarate, 20 kHz frequency deviation.</li><li>• DR_300_0kbps_Dev75_0kHz Base configuration of 300 kbps datarate, 75 kHz frequency deviation.</li></ul>
-------------------	---

Definition at line 46 of file settings.h.

4.2.3.13 #define RADIO\_FREQUENCY 433920000

Frequency for radio communications.

Parameters

<i>Frequency</i>	<pre>:{431000000-928000000}</pre> <ul style="list-style-type: none"><li>• This must be within the available bands of the radio:<ul style="list-style-type: none"><li>– 431000000Hz to 464000000Hz and</li><li>– 862000000Hz to 928000000Hz.</li></ul></li></ul>
------------------	---

Note

433.92 Mhz (EU) free frequency

Definition at line 64 of file settings.h.

4.2.3.14 #define RADIO\_MANCHASTER RIE\_FALSE

Enable or Disable Manchester Encoding of payload data.

Manchester encoding can be used to ensure a dc-free (zero mean) transmission.

A Binary 0 is mapped to 10, and a Binary 1 is mapped to 01.

Manchester encoding and decoding are applied to the payload data and the CRC.

## Parameters

<i>bEnable</i>	:{RIE_FALSE,RIE_TRUE} <ul style="list-style-type: none"> <li>• RIE_TRUE if Manchester Encoding is to be enabled.</li> <li>• RIE_FALSE if disabled.</li> </ul>
----------------	---

Definition at line 120 of file settings.h.

## 4.2.3.15 #define RADIO\_MODULATION GFSK\_Modulation

Radio Transmitter Modulation Type.

## Parameters

<i>ModulationType</i>	:{FSK_Modulation , GFSK_Modulation } <ul style="list-style-type: none"> <li>• FSK_Modulation Frequency shift keying modulatio</li> <li>• GFSK_Modulation Gaussian frequency shift keying modulatio</li> </ul>
-----------------------	---

Definition at line 54 of file settings.h.

## 4.2.3.16 #define RADIO\_POWER PowerLevel15

Power Level for Radio Transmission.

## Parameters

<i>Power</i>	:{PowerLevel0 ,PowerLevel1 ,PowerLevel2 ,PowerLevel3, PowerLevel4 ,PowerLevel5 ,PowerLevel6 ,PowerLevel7, PowerLevel8 ,PowerLevel9 ,PowerLevel10,PowerLevel11, PowerLevel12,PowerLevel13,PowerLevel14,PowerLevel15}
--------------	---

Definition at line 79 of file settings.h.

## 4.2.3.17 #define RETRANSMISION 3

number of retransmission trying until slave is marked as not responding

## Parameters

<i>retransmission</i>	attempts :{3}
-----------------------	---------------

Definition at line 219 of file settings.h.

4.2.3.18 #define RETRANSMISION\_ID "2RE"

format of retransmision packet

Parameters

<i>slave</i>	number{1 - NUM_OF_SLAVE}
--------------	--------------------------

Definition at line 244 of file settings.h.

4.2.3.19 #define RX\_STREAM 0

stream of redeived data to UART

Definition at line 271 of file settings.h.

4.2.3.20 #define SEND\_HEAD 0

send also heads of packets on UART

Definition at line 273 of file settings.h.

4.2.3.21 #define SIMULATE\_RETX 0

sending retransmitin message to test

Definition at line 269 of file settings.h.

4.2.3.22 #define SLAVE\_ID 2

number of actual slave

Parameters

<i>slave</i>	number{1 - NUM_OF_SLAVE}
--------------	--------------------------

Definition at line 248 of file settings.h.

4.2.3.23 #define STRING\_TERMINATOR '\$'

char witch terminate all received packets

Parameters

<i>char</i>	:{'\$'}
-------------	---------

Definition at line 143 of file settings.h.

#### 4.2.3.24 `#define SYNC_INTERVAL 200`

time interval to interrupt for synchronization

##### Note

time to interrupt =  $(1/40\,000\,000) * 256 * \text{SYNC\_INTERVAL [s]}$  200 = 1.28 ms //up to 255 (unsigned char)

##### Parameters

<i>count</i>	number :{200}
--------------	---------------

Definition at line 191 of file settings.h.

#### 4.2.3.25 `#define SYNC_PIN_HIGH DioSet(pADI_GP4,BIT2)`

Definition at line 262 of file settings.h.

#### 4.2.3.26 `#define SYNC_PIN_LOW DioClr(pADI_GP4,BIT2)`

Definition at line 263 of file settings.h.

#### 4.2.3.27 `#define SYNC_PIN_READ DioRd(pADI_GP4)&0x04`

Definition at line 264 of file settings.h.

#### 4.2.3.28 `#define T_PROCESSING 0`

appended time(number of increments) after transmtion to procesing on master

Definition at line 259 of file settings.h.

#### 4.2.3.29 `#define T_TIMEOUT 50000`

max time(number of increments) to response of requested devide

##### Note

interval witch is counted until packet is received

##### See also

[radioRecieve\(\)](#)



Parameters

<i>time</i>	:{50000}
-------------	----------

Definition at line 199 of file settings.h.

4.2.3.30 #define THROUGHPUT\_MEASURE 0

measured data troughput

Parameters

<i>{0-3}</i>	<ul style="list-style-type: none"><li>• if ==0 no measuring</li><li>• if ==1 measure throughput of received data from UART</li><li>• if ==2 measure maximum throughput with shyntetic data</li></ul>
--------------	--

Note

measured are all data included packet heads (aproximetlz 5000 Bytes/s by slave)

Definition at line 283 of file settings.h.

4.2.3.31 #define TIME\_SLOT\_ID\_MASTER "%dslot",slave\_ID

string defining format of "ID slot" packet

Definition at line 213 of file settings.h.

4.2.3.32 #define TIME\_SLOT\_ID\_SLAVE "2slot"

format of slot identificator

Parameters

<i>slave</i>	number{1 - NUM_OF_SLAVE}
--------------	--------------------------

Definition at line 236 of file settings.h.

4.2.3.33 #define TX\_STREAM 0

stream of transmited data to UART

Definition at line 272 of file settings.h.

#### 4.2.3.34 `#define UART_BAUD_RATE_MASTER 128000`

UART baudrate with is using master.

Definition at line 208 of file settings.h.

#### 4.2.3.35 `#define UART_BAUD_RATE_SLAVE 128000`

UART baudrate with is using master.

#### Note

Baudrate is ste to 9600 because of compatibility with "UWB - Coordinate Reader Deployment" from Peter Mikula

Definition at line 230 of file settings.h.

#### 4.2.3.36 `#define ZERO_PACKET "200"`

format of zero packet

#### Parameters

<code>slave</code>	number{1 - NUM_OF_SLAVE} first number
--------------------	---------------------------------------

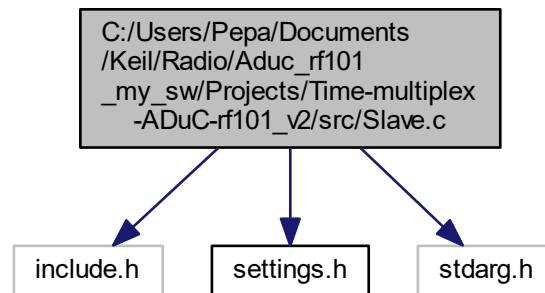
Definition at line 240 of file settings.h.

### 4.3 `C:/Users/Pepa/Documents/Keil/Radio/Aduc_rf101_my_sw/Projects/Time-multiplex-↵` **ADuC-rf101\_v2/src/Slave.c File Reference**

prigram used for transmitting data in time multiplex data are received via UART and sendet trouhg radio interface during associated time slot working with [Master.c](#) tested on ADucRF101MKxZ

```
#include "include.h"
#include "settings.h"
#include <stdarg.h>
```

Include dependency graph for Slave.c:



## Macros

- `#define LED_OFF DioSet(pADI_GP4,BIT2)`
- `#define LED_ON DioClr(pADI_GP4,BIT2)`

## Functions

- void `UART_Int_Handler` (void)
- void `uart_init` (void)
- void `led_init` (void)  
*initialize port with led diode*
- void `radiolnit` (void)
- void `dmaSend` (unsigned char \*buff, int len)
- int `dma_printf` (const char \*format,...)
- void `radioSend` (char \*buff, unsigned char len)
- unsigned char `rf_printf` (const char \*format,...)
- char `radioRecieve` (void)
- void `setTimeToSync` (unsigned int time)  
*set general purpose timer0 for synchronization timeout*
- char `transmit` (void)  
*function is transmitting all prepared packets throught radio link*
- char `retransmit` (void)  
*function retransmit missing packets if requested*
- void `SetInterruptPriority` (void)
- int `main` (void)
- void `GP_Tmr0_Int_Handler` (void)
- void `GP_Tmr1_Int_Handler` (void)
- void `DMA_UART_TX_Int_Handler` ()
- void `DMA_UART_RX_Int_Handler` ()
- void `HardFault_Handler` (void)

## Variables

- RIE\_Responses [RIE\\_Response](#) = RIE\_Success
- unsigned char [Buffer](#) [255]
- RIE\_U8 [PktLen](#)
- RIE\_S8 [RSSI](#)
- char [pktMemory](#) [2][[NUM\\_OF\\_PACKETS\\_IN\\_MEMORY](#)][[PACKET\\_MEMORY\\_DEPTH](#)]  
*memory to store all data to send/to receive*
- char [numOfPkt](#) [2] = {0,0}
- unsigned char [lenghtOfPkt](#) [2][[NUM\\_OF\\_PACKETS\\_IN\\_MEMORY](#)] = {0,0}
- char [actualRxBuffer](#) =0
- char [actualTxBuffer](#) =1
- char [rxUARTbuffer](#) [255]
- char \* [rxPktPtr](#)
- int [rxUARTcount](#) = 0
- char [TX\\_flag](#) =0
- char [RX\\_flag](#) =0
- char [terminate\\_flag](#) =0
- char [buffer\\_change\\_flag](#) =0
- char [my\\_slot](#) = 0
- int [i](#) =0
- int [j](#) =0
- int [debugTimer](#) =0

### 4.3.1 Detailed Description

prigram used for transmitting data in time multiplex data are received via UART and sended trouhgt radio interface during associated time slot working with [Master.c](#) tested on ADucRF101MKxZ

#### Version

V2.1B

#### Author

Peter Soltys

#### Date

febtuary 2016

#### Revision History:

- V1.0, July 2015 : initial version.
- V1.1, august 2015 : fully functional. (only shyntetic data transmitted)
- V1.2, august 2015 : fully functional. (transmiting received data via UART (only fixed lenght packet))
- V1.3, september 2015 : faster version with higher throughput and transmiting variable lenght packets working with "UWB - Coordinate Reader Deployment" from Peter Mikula
- V1.4, january 2015 : added synchronization
- V2.0, febtuary 2016 : new time multiplex conception
- V2.1, febtuary 2016 : fixed synchronization
- V2.1B, febtuary 2016 : Binary data packets (instead of strings with `STRING_TERMINATOR`)

Note

: in radioeng.c was changed initial value from  
static RIE\_BOOL bPacketTx = RIE\_FALSE;  
static RIE\_BOOL bPacketRx = RIE\_FALSE;  
to  
static RIE\_BOOL bPacketTx = RIE\_TRUE;  
static RIE\_BOOL bPacketRx = RIE\_TRUE;

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 #define LED\_OFF DioSet(pADI\_GP4,BIT2)

Definition at line 40 of file Slave.c.

#### 4.3.2.2 #define LED\_ON DioClr(pADI\_GP4,BIT2)

Definition at line 41 of file Slave.c.

### 4.3.3 Function Documentation

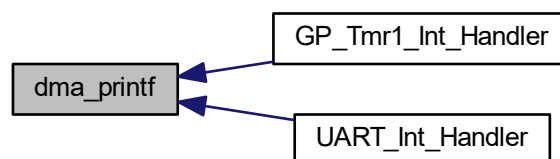
#### 4.3.3.1 int dma\_printf ( const char \* *format*, ... )

Definition at line 215 of file Slave.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.3.2 void DMA\_UART\_RX\_Int\_Handler ( )

Definition at line 612 of file Slave.c.

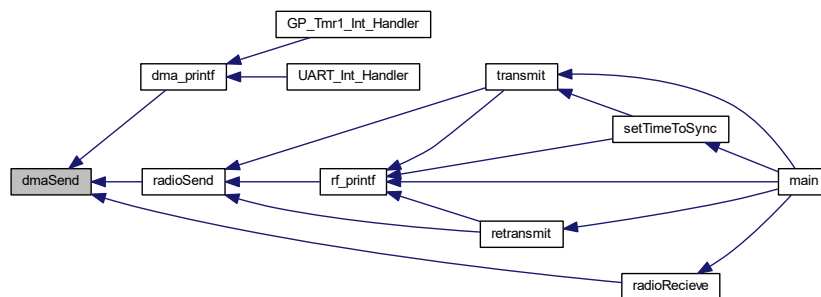
#### 4.3.3.3 void DMA\_UART\_TX\_Int\_Handler ( void )

Definition at line 597 of file Slave.c.

#### 4.3.3.4 void dmaSend ( unsigned char \* *buff*, int *len* )

Definition at line 192 of file Slave.c.

Here is the caller graph for this function:



#### 4.3.3.5 void GP\_Tmr0\_Int\_Handler ( void )

Definition at line 552 of file Slave.c.

#### 4.3.3.6 void GP\_Tmr1\_Int\_Handler ( void )

Definition at line 577 of file Slave.c.

Here is the call graph for this function:



4.3.3.7 void HardFault\_Handler ( void )

Definition at line 637 of file Slave.c.

4.3.3.8 void led\_init ( void )

initialize port with led diode

Definition at line 142 of file Slave.c.

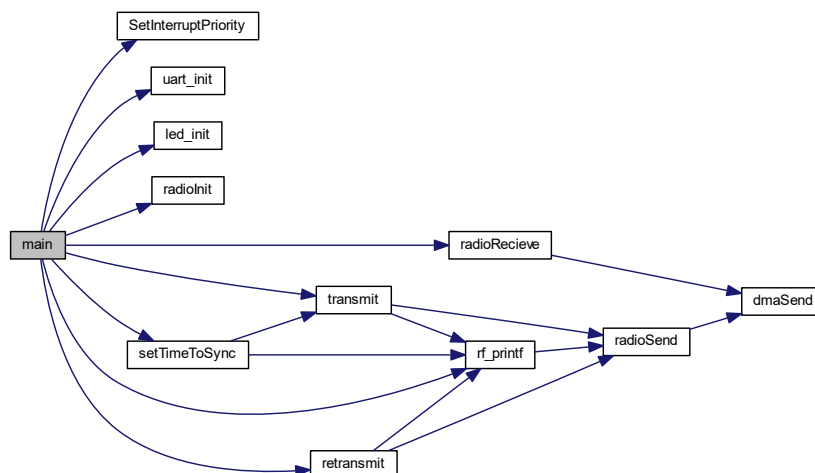
Here is the caller graph for this function:



4.3.3.9 int main ( void )

Definition at line 506 of file Slave.c.

Here is the call graph for this function:



#### 4.3.3.10 void radiolnit ( void )

Definition at line 157 of file Slave.c.

Here is the caller graph for this function:



#### 4.3.3.11 char radioRecieve ( void )

Definition at line 323 of file Slave.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.3.12 void radioSend ( char \* buff, unsigned char len )

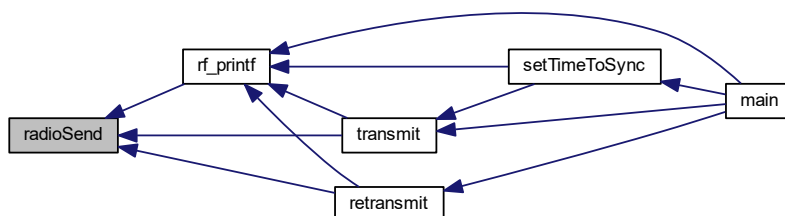
Definition at line 244 of file Slave.c.



Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.3.13 char retransmit ( void )

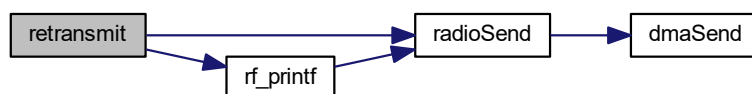
function retransmit missing packets if requested

##### Returns

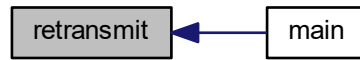
char - number of retransmitted packets

Definition at line 470 of file Slave.c.

Here is the call graph for this function:



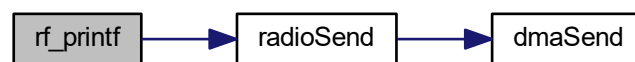
Here is the caller graph for this function:



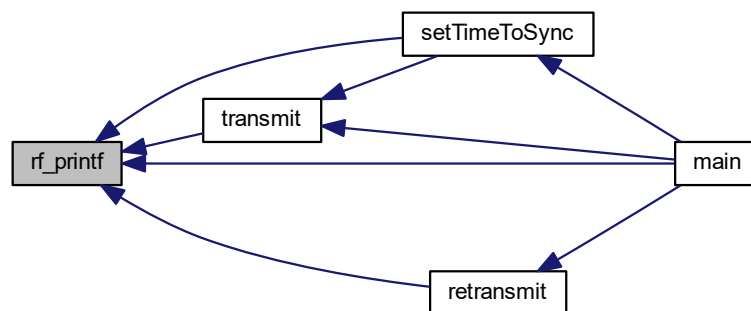
#### 4.3.3.14 unsigned char rf\_printf ( const char \* *format*, ... )

Definition at line 290 of file Slave.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.3.15 void SetInterruptPriority ( void )

Definition at line 491 of file Slave.c.

Here is the caller graph for this function:



#### 4.3.3.16 void setTimeToSync ( unsigned int *time* )

set general purpose timer0 for synchronization timeout

##### Parameters

<i>time</i>	{0- uint range} should be (N * SYNC_INTERVAL) where N=1-3
-------------	---

##### See also

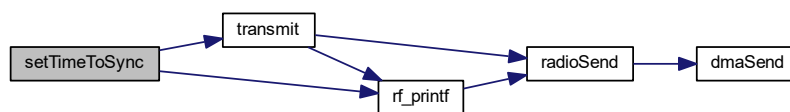
void [GP\\_Tmr0\\_Int\\_Handler](#) ()  
[SYNC\\_INTERVAL](#)

##### Note

timer predivider factor = 256, processor clock, periodic mode

Definition at line 373 of file Slave.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.3.17 char transmit ( void )

function is transmitting all prepared packets through radio link

##### Note

rotate packet memory all data packets are received via UART

##### See also

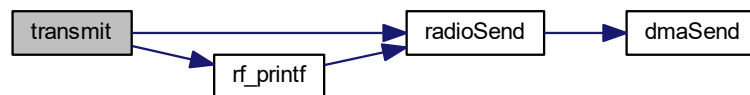
[pktMemory](#)

##### Returns

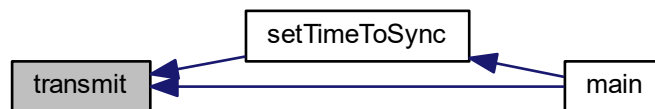
char - number of transmitted packets

Definition at line 425 of file Slave.c.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.3.18 void uart\_init ( void )

Definition at line 111 of file Slave.c.

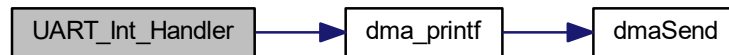
Here is the caller graph for this function:



4.3.3.19 void UART\_Int\_Handler ( void )

Definition at line 652 of file Slave.c.

Here is the call graph for this function:



#### 4.3.4 Variable Documentation

4.3.4.1 char actualRxBuffer =0

Definition at line 80 of file Slave.c.

4.3.4.2 char actualTxBuffer =1

Definition at line 81 of file Slave.c.

4.3.4.3 unsigned char Buffer[255]

Definition at line 45 of file Slave.c.

4.3.4.4 char buffer\_change\_flag =0

Definition at line 87 of file Slave.c.

4.3.4.5 int debugTimer =0

Definition at line 93 of file Slave.c.

4.3.4.6 int i =0

Definition at line 92 of file Slave.c.

4.3.4.7 int j =0

Definition at line 92 of file Slave.c.

4.3.4.8 unsigned char lenghtOfPkt[2][NUM\_OF\_PACKETS\_IN\_MEMORY] = {0,0}

Definition at line 77 of file Slave.c.

4.3.4.9 char my\_slot = 0

Definition at line 88 of file Slave.c.

4.3.4.10 char numOfPkt[2] = {0,0}

Definition at line 75 of file Slave.c.

4.3.4.11 RIE\_U8 PktLen

Definition at line 46 of file Slave.c.

4.3.4.12 char pktMemory[2][NUM\_OF\_PACKETS\_IN\_MEMORY][PACKET\_MEMORY\_DEPTH]

memory to store all data to send/to receive

#### Parameters

-	level
	<ul style="list-style-type: none"> <li>• packet num</li> <li>• packet data</li> </ul>

#### Note

pktMemory is 2 levels deep purpose is changing in circle 0 actual receiving buffer 1 actual sending buffer for pointing are used flags : actualRxBuffer, actualTxBuffer  
size of memory is restricted because ADuc rf101 have only 16KBytes SRAM  
UART must be much faster(tested on 128000 baud rate) than RF-link (to release memory)

#### See also

[actualRxBuffer](#)  
[actualTxBuffer](#)

Definition at line 66 of file Slave.c.

4.3.4.13 RIE\_Responses RIE\_Response = RIE\_Success

Definition at line 44 of file Slave.c.

---

4.3.4.14 RIE\_S8 RSSI

Definition at line 47 of file Slave.c.

4.3.4.15 char RX\_flag =0

Definition at line 87 of file Slave.c.

4.3.4.16 char\* rxPktPtr

Definition at line 84 of file Slave.c.

4.3.4.17 char rxUARTbuffer[255]

Definition at line 83 of file Slave.c.

4.3.4.18 int rxUARTcount = 0

Definition at line 85 of file Slave.c.

4.3.4.19 char terminate\_flag =0

Definition at line 87 of file Slave.c.

4.3.4.20 char TX\_flag =0

Definition at line 87 of file Slave.c.





# Index

actualPacket  
    Master.c, [28](#)  
actualRxBuffer  
    Master.c, [28](#)  
    Slave.c, [53](#)  
actualTxBuffer  
    Master.c, [28](#)  
    Slave.c, [53](#)  
  
BINARY\_MODE  
    settings.h, [33](#)  
Buffer  
    Master.c, [28](#)  
    Slave.c, [53](#)  
buffer\_change\_flag  
    Slave.c, [53](#)  
  
C:/Users/Pepa/Documents/Keil/Radio/Aduc\_rf101↔  
    \_my\_sw/Projects/Time-multiplex-ADuC-  
    rf101\_v2/src/Master.c, [7](#)  
C:/Users/Pepa/Documents/Keil/Radio/Aduc\_rf101↔  
    \_my\_sw/Projects/Time-multiplex-ADuC-  
    rf101\_v2/src/Slave.c, [42](#)  
C:/Users/Pepa/Documents/Keil/Radio/Aduc\_rf101↔  
    \_my\_sw/Projects/Time-multiplex-ADuC-  
    rf101\_v2/src/settings.h, [31](#)  
CHAR\_OFFSET  
    settings.h, [34](#)  
copyBufferToMemory  
    Master.c, [10](#)  
  
DATA\_WHITENING  
    settings.h, [34](#)  
DEBUG\_MESSAGES  
    settings.h, [34](#)  
DMA\_UART\_RX\_Int\_Handler  
    Master.c, [12](#)  
    Slave.c, [45](#)  
DMA\_UART\_TX\_Int\_Handler  
    Master.c, [12](#)  
    Slave.c, [46](#)  
debugTimer  
    Slave.c, [53](#)  
dma\_printf  
    Master.c, [11](#)  
    Slave.c, [45](#)  
dmaSend  
    Master.c, [13](#)  
    Slave.c, [46](#)  
dmaTx\_flag  
    Master.c, [28](#)  
    Slave.c, [28](#)  
dmaTxBuffer  
    Master.c, [28](#)  
dmaTxPkt  
    Master.c, [28](#)  
dmaTxPtr  
    Master.c, [28](#)  
dmaTxSlv  
    Master.c, [28](#)  
  
firstRxPkt  
    Master.c, [28](#)  
flush\_flag  
    Master.c, [28](#)  
flushBufferedPackets  
    Master.c, [14](#)  
  
GP\_Tmr0\_Int\_Handler  
    Master.c, [15](#)  
    Slave.c, [46](#)  
GP\_Tmr1\_Int\_Handler  
    Master.c, [15](#)  
    Slave.c, [46](#)  
  
HEAD\_FORMAT  
    settings.h, [34](#)  
HEAD LENGHT  
    settings.h, [35](#)  
HardFault\_Handler  
    Slave.c, [46](#)  
  
i  
    Slave.c, [53](#)  
ifMissPktGet  
    Master.c, [16](#)  
initializeNewSlot  
    Master.c, [17](#)  
  
j  
    Slave.c, [53](#)  
  
LED\_OFF  
    Master.c, [10](#)  
    Slave.c, [45](#)  
LED\_ON  
    Master.c, [10](#)  
    Slave.c, [45](#)  
LEN\_OF\_RX\_PKT  
    settings.h, [35](#)  
lastRadioTransmitBuffer  
    Master.c, [29](#)

- led\_init
  - Slave.c, [47](#)
- ledInit
  - Master.c, [17](#)
- lengthOfPkt
  - Master.c, [29](#)
  - Slave.c, [53](#)
- main
  - Master.c, [17](#)
  - Slave.c, [47](#)
- Master.c
  - actualPacket, [28](#)
  - actualRxBuffer, [28](#)
  - actualTxBuffer, [28](#)
  - Buffer, [28](#)
  - copyBufferToMemory, [10](#)
  - DMA\_UART\_RX\_Int\_Handler, [12](#)
  - DMA\_UART\_TX\_Int\_Handler, [12](#)
  - dma\_printf, [11](#)
  - dmaSend, [13](#)
  - dmaTx\_flag, [28](#)
  - dmaTxBuffer, [28](#)
  - dmaTxPkt, [28](#)
  - dmaTxPtr, [28](#)
  - dmaTxSlv, [28](#)
  - firstRxPkt, [28](#)
  - flush\_flag, [28](#)
  - flushBufferedPackets, [14](#)
  - GP\_Tmr0\_Int\_Handler, [15](#)
  - GP\_Tmr1\_Int\_Handler, [15](#)
  - ifMissPktGet, [16](#)
  - initializeNewSlot, [17](#)
  - LED\_OFF, [10](#)
  - LED\_ON, [10](#)
  - lastRadioTransmitBuffer, [29](#)
  - ledInit, [17](#)
  - lengthOfPkt, [29](#)
  - main, [17](#)
  - numOfPkt, [29](#)
  - PktLen, [29](#)
  - pktMemory, [29](#)
  - RIE\_Response, [29](#)
  - RSSI, [30](#)
  - RX\_flag, [30](#)
  - radioInit, [18](#)
  - radioRecieve, [19](#)
  - radioSend, [20](#)
  - receivePackets, [21](#)
  - rf\_printf, [22](#)
  - rxBuffer, [30](#)
  - rxUARTcount, [30](#)
  - send, [30](#)
  - SetInterruptPriority, [23](#)
  - setSynnicTimer, [23](#)
  - slave\_ID, [30](#)
  - sync\_flag, [30](#)
  - sync\_wait, [30](#)
  - synchronize, [24](#)
  - TX\_flag, [30](#)
  - UART\_BUFFER\_DEEP, [10](#)
  - UART\_Int\_Handler, [25](#)
  - uartInit, [25](#)
  - validPacket, [26](#)
  - zeroPacket, [27](#)
  - my\_slot
    - Slave.c, [54](#)
  - NUM\_OF\_PACKETS\_IN\_MEMORY
    - settings.h, [35](#)
  - NUM\_OF\_SLAVE
    - settings.h, [35](#)
  - numOfPkt
    - Master.c, [29](#)
    - Slave.c, [54](#)
  - PA\_TYPE
    - settings.h, [36](#)
  - PACKET\_MEMORY\_DEPTH
    - settings.h, [36](#)
  - PktLen
    - Master.c, [29](#)
    - Slave.c, [54](#)
  - pktMemory
    - Master.c, [29](#)
    - Slave.c, [54](#)
  - RADIO\_CFG
    - settings.h, [36](#)
  - RADIO\_FREQUENCY
    - settings.h, [37](#)
  - RADIO\_MANCHASTER
    - settings.h, [37](#)
  - RADIO\_MODULATION
    - settings.h, [38](#)
  - RADIO\_POWER
    - settings.h, [38](#)
  - RETRANSMISION\_ID
    - settings.h, [38](#)
  - RETRANSMISION
    - settings.h, [38](#)
  - RIE\_Response
    - Master.c, [29](#)
    - Slave.c, [54](#)
  - RSSI
    - Master.c, [30](#)
    - Slave.c, [54](#)
  - RX\_STREAM
    - settings.h, [39](#)
  - RX\_flag
    - Master.c, [30](#)
    - Slave.c, [55](#)
  - radioInit
    - Master.c, [18](#)
    - Slave.c, [47](#)
  - radioRecieve
    - Master.c, [19](#)
    - Slave.c, [48](#)

- radioSend
  - Master.c, [20](#)
  - Slave.c, [48](#)
- receivePackets
  - Master.c, [21](#)
- retransmit
  - Slave.c, [49](#)
- rf\_printf
  - Master.c, [22](#)
  - Slave.c, [50](#)
- rxBuffer
  - Master.c, [30](#)
- rxPktPtr
  - Slave.c, [55](#)
- rxUARTbuffer
  - Slave.c, [55](#)
- rxUARTcount
  - Master.c, [30](#)
  - Slave.c, [55](#)
- SEND\_HEAD
  - settings.h, [39](#)
- SIMULATE\_RETX
  - settings.h, [39](#)
- SLAVE\_ID
  - settings.h, [39](#)
- STRING\_TERMINATOR
  - settings.h, [39](#)
- SYNC\_INTERVAL
  - settings.h, [40](#)
- SYNC\_PIN\_HIGH
  - settings.h, [40](#)
- SYNC\_PIN\_LOW
  - settings.h, [40](#)
- SYNC\_PIN\_READ
  - settings.h, [40](#)
- send
  - Master.c, [30](#)
- SetInterruptPriority
  - Master.c, [23](#)
  - Slave.c, [50](#)
- setSynnicTimer
  - Master.c, [23](#)
- setTimeToSync
  - Slave.c, [51](#)
- settings.h
  - BINARY\_MODE, [33](#)
  - CHAR\_OFFSET, [34](#)
  - DATA\_WHITENING, [34](#)
  - DEBUG\_MESSAGES, [34](#)
  - HEAD\_FORMAT, [34](#)
  - HEAD\_LENHT, [35](#)
  - LEN\_OF\_RX\_PKT, [35](#)
  - NUM\_OF\_PACKETS\_IN\_MEMORY, [35](#)
  - NUM\_OF\_SLAVE, [35](#)
  - PA\_TYPE, [36](#)
  - PACKET\_MEMORY\_DEPTH, [36](#)
  - RADIO\_CFG, [36](#)
  - RADIO\_FREQUENCY, [37](#)
  - RADIO\_MANCHASTER, [37](#)
  - RADIO\_MODULATION, [38](#)
  - RADIO\_POWER, [38](#)
  - RETRANSMISION\_ID, [38](#)
  - RETRANSMISION, [38](#)
  - RX\_STREAM, [39](#)
  - SEND\_HEAD, [39](#)
  - SIMULATE\_RETX, [39](#)
  - SLAVE\_ID, [39](#)
  - STRING\_TERMINATOR, [39](#)
  - SYNC\_INTERVAL, [40](#)
  - SYNC\_PIN\_HIGH, [40](#)
  - SYNC\_PIN\_LOW, [40](#)
  - SYNC\_PIN\_READ, [40](#)
  - T\_PROCESSING, [40](#)
  - T\_TIMEOUT, [40](#)
  - THROUGHPUT\_MEASURE, [41](#)
  - TIME\_SLOT\_ID\_MASTER, [41](#)
  - TIME\_SLOT\_ID\_SLAVE, [41](#)
  - TX\_STREAM, [41](#)
  - UART\_BAUD\_RATE\_MASTER, [41](#)
  - UART\_BAUD\_RATE\_SLAVE, [42](#)
  - ZERO\_PACKET, [42](#)
- Slave.c
  - actualRxBuffer, [53](#)
  - actualTxBuffer, [53](#)
  - Buffer, [53](#)
  - buffer\_change\_flag, [53](#)
  - DMA\_UART\_RX\_Int\_Handler, [45](#)
  - DMA\_UART\_TX\_Int\_Handler, [46](#)
  - debugTimer, [53](#)
  - dma\_printf, [45](#)
  - dmaSend, [46](#)
  - GP\_Tmr0\_Int\_Handler, [46](#)
  - GP\_Tmr1\_Int\_Handler, [46](#)
  - HardFault\_Handler, [46](#)
  - i, [53](#)
  - j, [53](#)
  - LED\_OFF, [45](#)
  - LED\_ON, [45](#)
  - led\_init, [47](#)
  - lenghtOfPkt, [53](#)
  - main, [47](#)
  - my\_slot, [54](#)
  - numOfPkt, [54](#)
  - PktLen, [54](#)
  - pktMemory, [54](#)
  - RIE\_Response, [54](#)
  - RSSI, [54](#)
  - RX\_flag, [55](#)
  - radiolnit, [47](#)
  - radioRecieve, [48](#)
  - radioSend, [48](#)
  - retransmit, [49](#)
  - rf\_printf, [50](#)
  - rxPktPtr, [55](#)
  - rxUARTbuffer, [55](#)
  - rxUARTcount, [55](#)

- SetInterruptPriority, [50](#)
- setTimeToSync, [51](#)
- TX\_flag, [55](#)
- terminate\_flag, [55](#)
- transmit, [51](#)
- UART\_Int\_Handler, [52](#)
- uart\_init, [52](#)
- slave\_ID
  - Master.c, [30](#)
- sync\_flag
  - Master.c, [30](#)
- sync\_wait
  - Master.c, [30](#)
- synchronize
  - Master.c, [24](#)
- T\_PROCESSING
  - settings.h, [40](#)
- T\_TIMEOUT
  - settings.h, [40](#)
- THROUGHPUT\_MEASURE
  - settings.h, [41](#)
- TIME\_SLOT\_ID\_MASTER
  - settings.h, [41](#)
- TIME\_SLOT\_ID\_SLAVE
  - settings.h, [41](#)
- TX\_STREAM
  - settings.h, [41](#)
- TX\_flag
  - Master.c, [30](#)
  - Slave.c, [55](#)
- terminate\_flag
  - Slave.c, [55](#)
- transmit
  - Slave.c, [51](#)
- UART\_BAUD\_RATE\_MASTER
  - settings.h, [41](#)
- UART\_BAUD\_RATE\_SLAVE
  - settings.h, [42](#)
- UART\_BUFFER\_DEEP
  - Master.c, [10](#)
- UART\_Int\_Handler
  - Master.c, [25](#)
  - Slave.c, [52](#)
- uart\_init
  - Slave.c, [52](#)
- uartInit
  - Master.c, [25](#)
- validPacket
  - Master.c, [26](#)
- ZERO\_PACKET
  - settings.h, [42](#)
- zeroPacket
  - Master.c, [27](#)