AVR-Caller id Reference Manual

Generated by Doxygen 1.4.2

Sat Oct 22 18:30:45 2005

Contents

Chapter 1

AVR-CallerID

1.1 Introduction

1.1.1 What's in this package

This package contains a routine that can fetch caller-ID's from a telephone line and show it on an LCD.

The caller-ID is the telephone number of the one who is calling to you. This number, along with the date and time of the call is sent to you by your telephone company.

Your telephone company must use the ITU-Rec.V23bis, otherwise this project won't work. It means that your telephone company must send the data to you using FSK (frequency shift keying). Which means that the digital ones and zeros of the data have their own frequencies. A one is sent using a 1300Hz sine, a zero is sent by a 2100Hz sine. I use the Exar XR-2211 chip to decode these frequencies to digital zeroes and ones.

In many countries another modulation, namely DTMF is used. This project will NOT work with this modulation. You will need another decoding chip, an MT8870.

This project is actually very basic. If someone calls, his/her number, including the date and time are shown on the LCD. Then the microcontroller becomes blocked. You have to reset it to receive the number of the next caller.

So there remains still a lot of work to be done if you want to use it in a real application.

1.2 Bibliography

I read a lot more information than what is given here below. I only give you the relevant data in this section.

ITU-T Recommendation V.23

DATA COMMUNICATION OVER THE TELEPHONE NETWORK 600/1200-BAUD MODEM STANDARDIZED FOR USE IN THE GENERAL SWITCHED TELEPHONE NETWORK

http://www.nist.fss.ru/hr/doc/mstd/itu/index.htm

ETSI EN 300 659-1 V1.3.1 (2001-01)

European Standard (Telecommunications series) Access and Terminals (AT); Analogue access to the Public Switched Telephone Network (PSTN); Subscriber line protocol over the local loop for display (and related) services; Part 1: On-hook data transmission

ETSI EN 300 659-3 V1.3.1 (2001-01)

European Standard (Telecommunications series) Access and Terminals (AT); Analogue access to the Public Switched Telephone Network (PSTN); Subscriber line protocol over the local loop for display (and related) services; Part 3: Data link message and parameter codings

1.3 Datasheets

I have downloaded some datasheets of different manufacturers. Some contain errors. In some datasheets they choose completely different component values and the calculations are also different. So be careful when reading them.

NJM2211 FSK Demodulator/Tone decoder, New Japan Radio Co., Ltd.

RC2211A FSK Demodulator/Tone decoder, Fairchild semiconductor

XR-2211 FSK Demodulator/Tone decoder, Exar Corporation

This application note from Exar proved most useful to me:

TAN-009 Designing Caller Identification Delivery using XR-2211 for B.T.

Chapter 2

AVR-Caller id Class Index

2.1 AVR-Caller id Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

AVR-Caller id File Index

3.1 AVR-Caller id File List

Here is a list of all documented files with brief descriptions:

Chapter 4

AVR-Caller id Class Documentation

4.1 DATALINKLAYER Struct Reference

Public Attributes

- uint8_t messageType
- uint8_t totalBytes

  Total number of bytes as passed by the datalink layer.

- uint8_t checksum

  Checksum passed by the datalink layer.

4.1.1 Detailed Description

Struct that defines some parameters that are useful in the datalink layer level. See ETSI EN 300 659-1 V1.3.1 (2001-01), section 5.2 for a definition of these parameters.

4.1.2 Member Data Documentation

4.1.2.1 uint8_t DATALINKLAYER::messageType

Type of message received

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

- main.c

Chapter 5

AVR-Caller id File Documentation

5.1 define.h File Reference

```
#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
```

Defines

- #define SET_BIT(x, y) x |= (1<<y)

  Set bit on position y of register x.

- #define CLR_BIT(x, y) x &= ~(1<<y)

  Clear bit on position y of register x.

- #define TST_BIT(x, y) (x & (1<<y))
- #define FOSC 16000000

  Clock frequency of the MCU [Hz].

- #define RXBUFSIZE 250
- #define PRESENTATION_SIZE 253

  Maximum size of a presentation layer message.

- #define RS 7

  RS connected to pin 7 of PORTA.

- #define E 6

  EN connected to pin 6 of PORTA.

- #define RING_DET 5

  ring detect to pin 5 of PORTA

- #define LOCK_DET 7

  lock detect to pin 7 of PORTD

- #define FSK_IN 0

  fsk input to pin 0 of PORTD

- #define LOCK_TIMER 0

Enumerations

- enum BOOL {
  FALSE,
  TRUE }

  Boolean type.

Functions

- void init_USART (uint32_t baud, BOOL RX_ON)
- void send_USART (uint8_t *buf, uint8_t length)
- void presLayer (uint8_t *paramType, uint8_t paramLength)
- void initDisplay (void)
- void setSentence (char *s, uint8_t length, uint8_t pos)
- void timer0_init (void)

  Initialize timer0 so that an interrupt occurs every 33ms.

- BOOL timer0isElapsed (uint8_t index)
- void setTimer0 (uint8_t index, uint8_t val)

5.1.1 Detailed Description

Contains declaration of routines

5.1.2 Define Documentation

5.1.2.1 #define LOCK_TIMER 0

Softwaretimer that is used to check whether the PLL of the 2211-chip is locked.

5.1.2.2 #define RXBUFSIZE 250

Size of receive buffer in bytes

5.1.2.3 #define TST_BIT(x, y) (x & (1<<y))

Check if bit on position y of register x is 1.

Returns:
1 when byte was 1 else 0

5.1.3 Enumeration Type Documentation

5.1.3.1 enum BOOL

Boolean type.

Enumeration values:

FALSE 0

TRUE 1

5.1.4 Function Documentation

5.1.4.1 void init_USART (uint32_t *baud*, BOOL *RX_ON*)

Set the USART with baud, enable RX and TX, enable IRQ on receiving data.

Parameters:

baud Baudrate in [bps]

RX_ON Enable receiving UART-data when true.

5.1.4.2 void initDisplay (void)

This function initializes the display. So it's ready to show the data we send to it. See the datasheet of the S6A0069 controller chip of that display for more info on these commands.

5.1.4.3 void presLayer (uint8_t * *paramType*, uint8_t *paramLength*)

This function show the presentation layer messages on the LCD. It receives a pointer to the parametertype. This parametertype is followed by the presentation layer data. The paramLength determines the length of the presentation message belonging to that parametertype. Currently only two parametertypes are supported, date & time and caller id. It is possible to add some more parameters later on. For more info on the possible parameters, see section 5.2.1 of ETSI EN 300 659-3 V1.3.1 (2001-01).

Parameters:

paramType Pointer to the parameter type of the presentation layer message.

paramLength Length of that presentation layer message.

5.1.4.4 void send_USART (uint8_t * *buf*, uint8_t *length*)

Send a character to the USART

Parameters:

buf Pointer to the databuffer containing the data to be sent.

length Length of the data to be sent.

5.1.4.5 void setSentence (char * *s*, uint8_t *length*, uint8_t *pos*)

Function that shows a sentence on the LC-display.

Parameters:

s Pointer to the sentence.

length Length of the sentence.

pos Start position to show the message on the screen. This number must be in the range of 0 up to 31.

5.1.4.6 void setTimer0 (uint8_t *index*, uint8_t *val*)

Function to set up the software timers.

Parameters:

index Number of the software timer to setup

val Initial countdown value to write to the timer

5.1.4.7 BOOL timer0isElapsed (uint8_t *index*)

Poll timer to see if it has run out.

Parameters:

index of the timer to check

Returns:

TRUE when timer has run out, else FALSE

5.2 display.c File Reference

```
#include "define.h"
#include <avr/delay.h>
```

Functions

- void display (BOOL rs, uint8_t data)
- void setCharacter (uint8_t character)
- void setPosition (uint8_t position)
- void cls (void)

  Clear LCD.

- void initDisplay (void)
- void setSentence (char *s, uint8_t length, uint8_t pos)

5.2.1 Detailed Description

This program controls an PC1602-F LCD display. This display contains a S6A0069 controller chip, which is HD44780 compliant. This program is meant for an ATMEGA8535 controller running at 16MHz.

5.2.2 Function Documentation

5.2.2.1 void display (BOOL *rs*, uint8_t *data*)

Low-level routine for writing characters and commands to the LCD.

Parameters:

rs Determines whether we want to write data (TRUE) or instructions (FALSE).

data Byte to write to the LCD.

5.2.2.2 void initDisplay (void)

This function initializes the display. So it's ready to show the data we send to it. See the datasheet of the S6A0069 controller chip of that display for more info on these commands.

5.2.2.3 void setCharacter (uint8_t *character*)

Write a character to the LCD on the current cursorposition.

Parameters:

character Character to write to the LCD.

5.2.2.4 void setPosition (uint8_t *pos*)

Set position of the cursor.

Parameters:

pos For first line: 0 to 15, 16 to 31 for second line.

5.2.2.5 void setSentence (char * *s*, uint8_t *length*, uint8_t *pos*)

Function that shows a sentence on the LC-display.

Parameters:

s Pointer to the sentence.

length Length of the sentence.

pos Start position to show the message on the screen. This number must be in the range of 0 up to 31.

5.3 main.c File Reference

```
#include "define.h"
#include <avr/delay.h>
```

Enumerations

- enum RX_STATE {
  WAITING_FOR_RING,
  WAITING_FOR_LOCK,
  WAITING_FOR_MARK_SIGNAL,
  WAITING_FOR_DATA,
  CHECK_DATA,
  PRES_LAYER,
  ALL_DONE }

Functions

- void initDatalink (void)
  Initialize the datalinklayer structure.
- int main (void)
  Main function containing the state machine.

5.3.1 Detailed Description

Main file containing state machine for controlling telephone line.

5.3.2 Enumeration Type Documentation

5.3.2.1 enum RX_STATE

Definition of the states of the state machine that controls the telephone line.

5.4 presLayer.c File Reference

```
#include "define.h"
```

Enumerations

- enum PRES_LAYER_TYPE {
  DATE_TIME = 1,
  CLI = 2 }
  Types of the presentation layer message.
- enum DATE_TYPE {
  MONTH = 2,
  DAY = 4,
  HOUR = 6,
  MIN = 8 }
  Position of the data in the "date" presentation layer type.

Functions

- void presLayer (uint8_t *paramType, uint8_t paramLength)

5.4.1 Detailed Description

File takes care of the higher OSI layer of the caller id system

5.4.2 Enumeration Type Documentation

5.4.2.1 enum PRES_LAYER_TYPE

Types of the presentation layer message.

Enumeration values:

DATE_TIME Date & Time.

CLI identity of the origin of the call

5.4.3 Function Documentation

5.4.3.1 void presLayer (uint8_t * *paramType*, uint8_t *paramLength*)

This function show the presentation layer messages on the LCD. It receives a pointer to the parametertype. This parametertype is followed by the presentation layer data. The paramLength determines the length of the presentation message belonging to that parametertype. Currently only two parametertypes are supported, date & time and caller id. It is possible to add some more parameters later on. For more info on the possible parameters, see section 5.2.1 of ETSI EN 300 659-3 V1.3.1 (2001-01).

Parameters:

paramType Pointer to the parameter type of the presentation layer message.

paramLength Length of that presentation layer message.

5.5 timer0.c File Reference

```
#include "define.h"
```

Defines

- #define MAX_TIMERS 1

  Maximum number of software timers possible.

Functions

- void setTimer0 (uint8_t index, uint8_t val)
- INTERRUPT (SIG_OUTPUT_COMPARE0)
- BOOL timer0isElapsed (uint8_t index)
- void timer0_init ()

  Initialize timer0 so that an interrupt occurs every 33ms.

Variables

- uint8_t timers [MAX_TIMERS]

  Array containing the software timers.

5.5.1 Detailed Description

Routines to make use of timer0

5.5.2 Function Documentation

5.5.2.1 INTERRUPT (SIG_OUTPUT_COMPARE0)

ISR of timer0. This ISR decreases the software timers

5.5.2.2 void setTimer0 (uint8_t *index*, uint8_t *val*)

Function to set up the software timers.

Parameters:

index Number of the software timer to setup

val Initial countdown value to write to the timer

5.5.2.3 BOOL timer0isElapsed (uint8_t *index*)

Poll timer to see if it has run out.

Parameters:
index of the timer to check

Returns:
TRUE when timer has run out, else FALSE

5.6 usart.c File Reference

```
#include "define.h"
```

Functions

- void init_USART (uint32_t baud, BOOL RX_ON)
- void send_USART (uint8_t *buf, uint8_t length)
- SIGNAL (SIG_UART_RECV)

  Receive ISR of the UART. It puts all the data in the circRXArray.

Variables

- uint8_t circRXArray [RXBUFSIZE]

  Buffer containing data that has been received through the UART.

- uint8_t teller

  Current position in the circRXArray buffer.

5.6.1 Detailed Description

This function takes care of the USART. This is the lowest possible OSI-layer on the MCU.

5.6.2 Function Documentation

5.6.2.1 void init_USART (uint32_t *baud*, BOOL *RX_ON*)

Set the USART with baud, enable RX and TX, enable IRQ on receiving data.

Parameters:

baud Baudrate in [bps]

RX_ON Enable receiving UART-data when true.

5.6.2.2 void send_USART (uint8_t * *buf*, uint8_t *length*)

Send a character to the USART

Parameters:

buf Pointer to the databuffer containing the data to be sent.

length Length of the data to be sent.

Index