/\*\* @file

 Rojaflex shutter and remote devices.

 This program is free software; you can redistribute it and/or modify

 it under the terms of the GNU General Public License as published by

 the Free Software Foundation; either version 2 of the License, or

 (at your option) any later version.

 \*/

/\*\*

// Frequency documenation

- Modulation: FSK PCM

- Frequency: 433.92MHz ???+-10kHz??? //TODO

- ???212 us symbol/bit time ??? //Todo

// Signal documentation

// Default signal layout

// 0xaaaaaaaa d391d391 SS KKKKKK ?CDDDD TTTT CCCC

//

// 4 Bytes Preamble

// 4 Bytes Sync Word

// 1 Byte Size "S" is always "0x08"

// 3 Bytes Key "Seems to be the static ID for the Homeinstallation"

// 3 Bytes Data "See docu below"

// 2 Bytes Token "It seems to be a message token which is used for the shutter answer.

// 2 Bytes CRC "CRC" no clue how to calculate, but seems optional, because this is missing from commands via bridge P2D

// 19 Byte Summe // Only 17 Bytes from bridge

//

// Data documentation

// 0xFF - Size always "0x8"

// 0xFFFFFF - Home ID, I assume that differs per installation, but is static then

// 0xF - Unknown (is static 0x2) - Not sure if it is also the HomeID

// 0xF - Radio channel: 1-15 single channels (one shutter is registert to one channel), 0 means all

// 0xFF - Command ID (0x0a = stop, 0x1a = up,0x8a = down, 0xea = Request)

// 0xFF - Command Value (in status from shutter this is the percent value. 0% for open 100% for close)

//

// Message Token documentation

// ... TODO ...

//

// To get raw data:

// ./rtl\_433 -f 433920000 -X n=rojaflex,m=FSK\_PCM,s=100,l=100,r=102400

\*/

#include "decoder.h"

// Defines

#define MESSAGE\_BYTECOUNT\_MAX 19 //Including CRC which is optional

#define MESSAGE\_BITCOUNT\_MAX 152

#define PREAMBLE\_OFFSET 0

#define PREAMBLE\_BITCOUNT 64

#define LENGTH\_OFFSET 8

#define LENGTH\_BITCOUNT 8

static uint8\_t get\_length(uint8\_t \*pData) { return pData[LENGTH\_OFFSET]; }

#define HOMEID\_OFFSET 9

#define HOMEID\_BITCOUNT 24

#define RADIO\_CHANNEL\_OFFSET 12

#define RADIO\_CHANNEL\_MASK 0x0F

static uint8\_t get\_radio\_channel(uint8\_t \*pData) { return pData[RADIO\_CHANNEL\_OFFSET] & RADIO\_CHANNEL\_MASK; }

#define UNKNOWN\_CHANNEL\_OFFSET 12

#define UNKNOWN\_CHANNEL\_MASK 0xF0

static uint8\_t get\_unknown(uint8\_t \*pData) { return (pData[UNKNOWN\_CHANNEL\_OFFSET] & UNKNOWN\_CHANNEL\_MASK) >> 4; }

#define COMMAND\_ID\_OFFSET 13

#define COMMAND\_ID\_BITCOUNT 8

#define COMMAND\_VALUE\_OFFSET 14

#define COMMAND\_VALUE\_BITCOUNT 8

#define MESSAGE\_TOKEN\_OFFSET 15

#define MESSAGE\_TOKEN\_BITCOUNT 16

#define MESSAGE\_CRC\_OFFSET 17

#define MESSAGE\_CRC\_BITCOUNT 16

// Done

static unsigned int isCRCAvailable( unsigned int length )

{

 return (length == MESSAGE\_BITCOUNT\_MAX ? 1 : 0);

}

// TODO

static unsigned int isCRCValid( uint8\_t \* data, unsigned int length)

{

 return 1;

}

// Done

static char\* getDeviceTypeString( uint8\_t \* data, unsigned int length)

{

 if( (data[13] & 0xF) == 0x5 ) { // Check command type: Status -> Shutter

 return "RojaFlex Shutter";

 } else if ( (data[13] & 0xF) == 0xa ) { // Set commands from remotes ...

 if ( isCRCAvailable(length) ) {

 return "RojaFlex Remote";

 } else {

 return "RojaFlex Bridge";

 }

 }

 return "Unknown";

}

// TODO

// Validate and precalcualte message token

// TODO

// Message Generator

//TODO

static char\* getHomeIDString( uint8\_t \* data, unsigned int length)

{

 //TODO Dynamic parsing from the buffer

 return "0x3122fd '2'";

}

// Done

static char\* getCommandString( uint8\_t \* data, unsigned int length)

{

 switch(data[13]){

 case 0x0a:

 return "Stop";

 case 0x1a:

 return "Up";

 case 0x8a:

 return "Down";

 case 0xea:

 // I am not sure if that is true.

 // I know that the remote is sending the message and not the shutter.

 // I know that the bridge is not sending this message after e.g.0x1a.

 // I know that the shutter sends a Position status right after this message.

 // After the normal 0x1a command from a bridge, the position status

 // will be send wenn the stutter is completely up but not before.

 // So I think this is a "Request Shutter Status Now".

 return "Request Status";

 case 0x85: // 0%

 return "Position Status - 0%";

 case 0x95: // 20%

 return "Position Status - 20%";

 case 0xA5: // 40%

 return "Position Status - 40%";

 case 0xB5: // 60%

 return "Position Status - 60%";

 case 0xC5: // 80%

 return "Position Status - 80%";

 case 0xD5: //100%

 return "Position Status - 100%";

 }

 return "unknown";

}

// Done

static uint8\_t getCommandValue( uint8\_t \* data, unsigned int length)

{

 return data[14];

}

static int rojaflex\_decode(r\_device \*decoder, bitbuffer\_t \*bitbuffer)

{

 uint8\_t const message\_preamble[] = {

 0xaa, 0xaa, 0xaa, 0xaa, // preamble

 0xd3, 0x91, 0xd3, 0x91 // sync word

 };

 data\_t \*data;

 uint8\_t msg[MESSAGE\_BYTECOUNT\_MAX];

 uint8\_t msg\_bitcount = 0;

 if (bitbuffer->num\_rows != 1) {

 return DECODE\_ABORT\_EARLY;

 }

 int row = 0;

 // Validate message and reject it as fast as possible : check for preamble

 unsigned start\_pos = bitbuffer\_search(bitbuffer, row, 0, message\_preamble, sizeof (message\_preamble) \* 8);

 if (start\_pos < bitbuffer->bits\_per\_row[row]) {

 // Save bitcount of total message including preamble

 msg\_bitcount = (bitbuffer->bits\_per\_row[row] - start\_pos) & 0xFE;

 //msg\_bitcount = (bitbuffer->bits\_per\_row[row] - start\_pos - (sizeof (message\_preamble) \* 8) ) & 0xFE;

 } else {

 return DECODE\_ABORT\_EARLY; // no preamble detected

 }

 if ( msg\_bitcount < (MESSAGE\_BITCOUNT\_MAX - MESSAGE\_CRC\_BITCOUNT) || msg\_bitcount > MESSAGE\_BITCOUNT\_MAX) {

 // check min length

 return DECODE\_ABORT\_LENGTH;

 }

 //Extract raw line - 16 if we have noch the full preamble included for better detection

 bitbuffer\_extract\_bytes(bitbuffer, row, start\_pos, msg, sizeof (msg) \* 8);

#if 0

 uint16\_t crc\_message = ((msg[MESSAGE\_CRC\_OFFSET] << 8 | msg[MESSAGE\_CRC\_OFFSET + 1]);

 uint16\_t crc\_calc = crc16(&msg[LENGTH\_OFFSET+1], 28, 0x8005, 0xffff);

 if (crc\_message != crc\_calc) {

 //if (decoder->verbose) {

 fprintf(stderr, "%s: CRC invalid message:%04x != calc:%04x\n", \_\_func\_\_, crc\_message, crc\_calc);

 //}

 //return DECODE\_FAIL\_MIC;

 }

#endif

 data = data\_make(

 "model", "Model", DATA\_STRING, getDeviceTypeString(&msg[0], msg\_bitcount),

 "radiochannel", "Radio CH.", DATA\_INT, get\_radio\_channel(&msg[0]),

 "homeid", "Home ID", DATA\_STRING, getHomeIDString(&msg[0], msg\_bitcount),

 "commandtype", "\nCommand ", DATA\_STRING, getCommandString(&msg[0], msg\_bitcount),

 "commandvalue", "Value", DATA\_INT, getCommandValue(&msg[0], msg\_bitcount),

 NULL);

 /\* clang-format on \*/

 decoder\_output\_data(decoder, data);

 if ( 1 /\*decoder->verbose\*/ )

 {

 fprintf(stderr, "\n" );

 bitrow\_printf(&msg[0], msg\_bitcount, "%s: raw: ", \_\_func\_\_ );

 fprintf(stderr, "%s:\n", \_\_func\_\_ );

 bitrow\_printf(&msg[PREAMBLE\_OFFSET], PREAMBLE\_BITCOUNT, "%s: Preamble: ", \_\_func\_\_ );

 bitrow\_printf(&msg[LENGTH\_OFFSET], LENGTH\_BITCOUNT, "%s: Lenght: ", \_\_func\_\_ );

 bitrow\_printf(&msg[HOMEID\_OFFSET], HOMEID\_BITCOUNT, "%s: Home ID: ", \_\_func\_\_ );

 fprintf(stderr, "%s: Unknown: { 4} %x\n", \_\_func\_\_, get\_unknown( &msg[0] ));

 fprintf(stderr, "%s: Channel: { 4} %x\n", \_\_func\_\_, get\_radio\_channel( &msg[0] ));

 bitrow\_printf(&msg[COMMAND\_ID\_OFFSET], COMMAND\_ID\_BITCOUNT, "%s: Command ID: ", \_\_func\_\_ );

 bitrow\_printf(&msg[COMMAND\_VALUE\_OFFSET], COMMAND\_VALUE\_BITCOUNT, "%s: Command Value: ", \_\_func\_\_ );

 bitrow\_printf(&msg[MESSAGE\_TOKEN\_OFFSET], MESSAGE\_TOKEN\_BITCOUNT, "%s: Key: ", \_\_func\_\_ );

 if( isCRCAvailable(msg\_bitcount) ) {

 bitrow\_printf(&msg[MESSAGE\_CRC\_OFFSET], MESSAGE\_CRC\_BITCOUNT, "%s: CRC: ", \_\_func\_\_ );

 } else {

 fprintf(stderr, "%s: CRC: missing\n", \_\_func\_\_ );

 }

 }

 return 1;

}

static char \*output\_fields[] = {

 "model",

 "radiochannel",

 "homeid",

 "commandtype",

 "commandvalue",

 NULL,

};

r\_device rojaflex = {

 .name = "Rojaflex",

 .modulation = FSK\_PULSE\_PCM,

 .short\_width = 100,

 .long\_width = 100,

 .reset\_limit = 102400,

 .sync\_width = 0,

 .decode\_fn = &rojaflex\_decode,

 .disabled = 0,

 .fields = output\_fields,

};