arch/avr/examples/xmodem_bootloader/xmodem.c

```
00001 /* ============================================================================
00002
00003     Copyright (c) 2006 Pieter Conradie [www.piconomic.co.za]
00004     All rights reserved.
00005
00006     Redistribution and use in source and binary forms, with or without
00007     modification, are permitted provided that the following conditions are met:
00008
00009     * Redistributions of source code must retain the above copyright
00010      notice, this list of conditions and the following disclaimer.
00011
00012     * Redistributions in binary form must reproduce the above copyright
00013      notice, this list of conditions and the following disclaimer in
00014      the documentation and/or other materials provided with the
00015      distribution.
00016
00017     * Neither the name of the copyright holders nor the names of
00018      contributors may be used to endorse or promote products derived
00019      from this software without specific prior written permission.
00020
00021     THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00022     AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00023     IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00024     ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
00025     LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00026     CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00027     SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00028     INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00029     CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00030     ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00031     POSSIBILITY OF SUCH DAMAGE.
00032
00033     Title:          XMODEM-CRC receive module
00034     Author(s):      Pieter Conradie
00035     Creation Date:  2007-03-31
00036     Revision Info:  $Id: xmodem.c 117 2010-06-24 20:21:28Z pieterconradie $
00037
00038 ============================================================================ */
00039
00040 /* _____STANDARD INCLUDES____________________________________________________ */
00041 #include <string.h>
00042
00043 /* _____PROJECT INCLUDES_____________________________________________________ */
00044 #include "xmodem.h"
00045 #include "tmr_poll.h"
00046 #include "uart_poll.h"
00047
00048 /* _____LOCAL DEFINITIONS____________________________________________________ */
00049 /// \name XMODEM protocol definitions
00050 //@{
00051 #define XMODEM_PACKET_SIZE       133
00052 #define XMODEM_DATA_SIZE         128
```

```
00053 #define XMODEM_TIMEOUT_MS        1000
00054 #define XMODEM_MAX_RETRIES       4
00055 #define XMODEM_MAX_RETRIES_START 1
00056 //@}
00057
00058 /// \name XMODEM flow control characters
00059 //@{
00060 #define XMODEM_SOH               0x01 // Start of Header
00061 #define XMODEM_EOT               0x04 // End of Transmission
00062 #define XMODEM_ACK               0x06 // Acknowledge
00063 #define XMODEM_NAK               0x15 // Not Acknowledge
00064 #define XMODEM_C                 0x43 // ASCII C
00065 //@}
00066
00067 /// XMODEM error list
00068 typedef enum
00069 {
00070     XMODEM_NO_ERROR,
00071     XMODEM_RECEIVED_EOT,
00072     XMODEM_ERR_RECEIVED_NOTHING,
00073     XMODEM_ERR_RECEIVE_TIMEOUT,
00074     XMODEM_ERR_INCORRECT_HEADER,
00075     XMODEM_ERR_INCORRECT_PACKET_NUMBER,
00076     XMODEM_ERR_DUPLICATE_PACKET_NUMBER,
00077     XMODEM_ERR_INCORRECT_CRC
00078 } xmodem_error_t;
00079
00080 /* _____LOCAL VARIABLES_____ */
00081 /// Buffer to store received data
00082 static u8_t xmodem_rx_buffer[XMODEM_PACKET_SIZE];
00083
00084 // Variable to keep track of current packet number
00085 static u8_t xmodem_packet_number;
00086
00087 /* _____PRIVATE FUNCTIONS_____ */
00088 /// Function that verifies that packet checksum is correct
00089 static bool_t xmodem_verify_checksum(void)
00090 {
00091     u8_t  i;
00092     u8_t  j;
00093     u8_t  data;
00094     u16_t crc = 0x0000;
00095
00096     // Repeat until all the data has been processed...
00097     for(j=3; j<(3+XMODEM_DATA_SIZE); j++)
00098     {
00099         data = xmodem_rx_buffer[j];
00100
00101         // XOR high byte of CRC with 8-bit data
00102         crc = crc ^ (((u16_t)data)<<8);
00103
00104         // Repeat 8 times
00105         for(i=8; i!=0; i--)
00106         {
00107             // If highest bit is set, then shift left and XOR with 0x1021
00108             if((crc & 0x8000) != 0)
00109             {
00110                 crc = (crc << 1) ^ 0x1021;
00111             }
00112             // else, just shift left
00113             else
00114             {
00115                 crc = (crc << 1);
```

```
00116             }
00117         }
00118     }
00119
00120     // Compare received CRC with calculated value
00121     if(xmodem_rx_buffer[3+XMODEM_DATA_SIZE] != U16_HI8(crc))
00122     {
00123         return FALSE;
00124     }
00125     if(xmodem_rx_buffer[4+XMODEM_DATA_SIZE] != U16_LO8(crc))
00126     {
00127         return FALSE;
00128     }
00129
00130     return TRUE;
00131 }
00132
00133
00134 /// Blocking function with a 1s timeout that tries to receive an XMODEM packet
00135 static xmodem_error_t xmodem_rx_packet(void)
00136 {
00137     u8_t  i = 0;
00138     u8_t  data;
00139
00140     tmr_poll_start(TMR_POLL_MS_TO_START_VAL(XMODEM_TIMEOUT_MS));
00141     while(!tmr_poll_has_exipred())
00142     {
00143         // See if character has been received
00144         if(uart_poll_rx_byte(&data))
00145         {
00146             // See if this is the first byte of a packet received
00147             if(i == 0)
00148             {
00149                 // See if End Of Transmission has been received
00150                 if(data == XMODEM_EOT)
00151                 {
00152                     return XMODEM_RECEIVED_EOT;
00153                 }
00154                 // Restart timer
00155                 tmr_poll_start(TMR_POLL_MS_TO_START_VAL(XMODEM_TIMEOUT_MS));
00156             }
00157             // Store received data in buffer
00158             xmodem_rx_buffer[i] = data;
00159             // Next byte in packet until whole packet has been received
00160             if(++i == XMODEM_PACKET_SIZE)
00161             {
00162                 break;
00163             }
00164         }
00165     }
00166
00167     // See if anything was received
00168     if(i == 0)
00169     {
00170         return XMODEM_ERR_RECEIVED_NOTHING;
00171     }
00172
00173     // See if correct header was received
00174     if(xmodem_rx_buffer[0] != XMODEM_SOH)
00175     {
00176         return XMODEM_ERR_INCORRECT_HEADER;
00177     }
00178
```

```
00179     // See if whole packet was received
00180     if(i != XMODEM_PACKET_SIZE)
00181     {
00182         return XMODEM_ERR_RECEIVE_TIMEOUT;
00183     }
00184
00185     // Check packet number checksum
00186     if((xmodem_rx_buffer[1]^xmodem_rx_buffer[2]) != 0xFF)
00187     {
00188         return XMODEM_ERR_INCORRECT_PACKET_NUMBER;
00189     }
00190
00191     // See if duplicate packet was received
00192     if(xmodem_rx_buffer[1] == (xmodem_packet_number-1))
00193     {
00194         return XMODEM_ERR_DUPLICATE_PACKET_NUMBER;
00195     }
00196
00197     // Make sure correct packet was received
00198     if(xmodem_rx_buffer[1] != xmodem_packet_number)
00199     {
00200         return XMODEM_ERR_INCORRECT_PACKET_NUMBER;
00201     }
00202
00203     // Verify Checksum
00204     if(!xmodem_verify_checksum())
00205     {
00206         return XMODEM_ERR_INCORRECT_CRC;
00207     }
00208
00209     return XMODEM_NO_ERROR;
00210 }
00211
00212 /* _____FUNCTIONS____________________________________________________ */
00213 bool_t xmodem_rx_file(xmodem_on_rx_data_t on_rx_data)
00214 {
00215     u8_t   retry_count    = XMODEM_MAX_RETRIES_START;
00216     bool_t first_ack_sent = FALSE;
00217     u8_t   data;
00218
00219     // Reset packet number
00220     xmodem_packet_number = 1;
00221
00222     // Repeat until transfer is finished or error count is exceeded
00223     while(retry_count--)
00224     {
00225         if(!first_ack_sent)
00226         {
00227             // Send initial start character to start transfer (with CRC checking)
00228             uart_poll_tx_byte(XMODEM_C);
00229         }
00230
00231         // Receive packet
00232         switch(xmodem_rx_packet())
00233         {
00234         case XMODEM_NO_ERROR:
00235             // Pass received data on to handler
00236             (*on_rx_data)(xmodem_rx_buffer+3,XMODEM_DATA_SIZE);
00237             // Acknowledge packet
00238             uart_poll_tx_byte(XMODEM_ACK);
00239             // Set flag to indicate that first packet has been correctly received
00240             first_ack_sent = TRUE;
00241             // Next packet
```

```
00242             xmodem_packet_number++;
00243             // Reset retry count
00244             retry_count = XMODEM_MAX_RETRIES;
00245             break;
00246 
00247         case XMODEM_RECEIVED_EOT:
00248             // Acknowledge EOT and make sure sender has received ACK
00249             XMODEM_EOT_STATE:
00250             while(--retry_count)
00251             {
00252                 uart_poll_tx_byte(XMODEM_ACK);
00253                 tmr_poll_start(TMR_POLL_MS_TO_START_VAL(XMODEM_TIMEOUT_MS));
00254                 while(!tmr_poll_has_exipred())
00255                 {
00256                     if(uart_poll_rx_byte(&data))
00257                         if(data == XMODEM_EOT)
00258                         {
00259                             // Unfortunate, but neccesary use of the "goto" keyword
00260                             goto XMODEM_EOT_STATE;
00261                         }
00262                 }
00263                 // File successfully transferred
00264                 return TRUE;
00265             }
00266             // File not successfully transferred
00267             return FALSE;
00268 
00269         case XMODEM_ERR_DUPLICATE_PACKET_NUMBER:
00270             // Acknowledge packet
00271             uart_poll_tx_byte(XMODEM_ACK);
00272             break;
00273 
00274         case XMODEM_ERR_RECEIVED_NOTHING:
00275             // Fall through...
00276         case XMODEM_ERR_RECEIVE_TIMEOUT:
00277             // Fall through...
00278         case XMODEM_ERR_INCORRECT_HEADER:
00279             // Fall through...
00280         case XMODEM_ERR_INCORRECT_PACKET_NUMBER:
00281             // Fall through...
00282         case XMODEM_ERR_INCORRECT_CRC:
00283             // Fall through...
00284         default:
00285             if(first_ack_sent)
00286             {
00287                 // Send NAK
00288                 uart_poll_tx_byte(XMODEM_NAK);
00289             }
00290             break;
00291         }
00292     }
00293 
00294     // File not successfully transferred
00295     return FALSE;
00296 }
00297 
00298 /* _____LOG__________________________________________________________________ */
00299 /*
00300 
00301  2007-03-31 : Pieter Conradie
00302  - First release
00303 
00304 */
```

Generated on Fri Aug 13 21:50:35 2010 for Piconomic Firmware Library by  1.6.3