```c
/*******************************************************************************
 *
 *    ________  _________  ____    ____  _____          ________   ______
 *   |_   __  ||  _   _  ||_   \  /   _||_   _|        |_   __  |.' ____ \
 *     | |_ \_||_/ | | \_|  |   \/   |    | |     ______ | |_ \_|| (___ \_|
 *     |  _| _     | |      | |\  /| |    | |   _|______||  _| _  _.____`.
 *    _| |__/ |   _| |_    _| |_\/_| |_  _| |__/ |      _| |__/ || \____) |
 *   |________|  |_____|  |_____||_____||________|     |________| \______.'
 *
 *******************************************************************************
 *
 * File          : chu.c
 * Version       : 1.0
 *
 *******************************************************************************
 *
 * Description   : Managing Chilli UART b1 state machine and commands
 *                 Manufacturer library is needed to interface it
 *                 https://eccel.co.uk/wp-content/downloads/C-library-for-B1.zip
 *
 *******************************************************************************
 *
 * Author        : Miguel Santos
 * Date          : 25.09.2023
 *
 *******************************************************************************
 *
 * MPLAB X       : 5.45
 * XC32          : 2.50
 * Harmony       : 2.06
 *
 ******************************************************************************/

#include "chu.h"
#include "modules/ccittcrc.h"

/******************************************************************************/

/* Enable or disable debug of specific parts by (un)comment */
#ifndef DEBUG_LED
    #define DEBUG_LED PORTGbits.RG9
#endif
//#define DEBUG_CHU_UART

/******************************************************************************/

/* Define UART and interupts used by Chilli */
#define CHU_USART_ID                    USART_ID_2
#define CHU_INT_SOURCE_USART_ERROR      INT_SOURCE_USART_2_ERROR
#define CHU_INT_SOURCE_USART_RECEIVE    INT_SOURCE_USART_2_RECEIVE
#define CHU_INT_SOURCE_USART_TRANSMIT INT_SOURCE_USART_2_TRANSMIT

/******************************************************************************/

/* Declaration of global application data */
CHU_DATA chuData;

/* RFID B1 UART objects */
RFIDB1_InterfaceConfigurationT chuRfid_config;
RFIDB1_InterfaceT chuRfid_interface;
RFIDB1_ObjectT chuRfid_object;

/******************************************************************************/

/**
 * @brief CHU_Initialize
 *
 * Initialize Chilli state machine, counters and FIFOs
 * Setup objects needed for RFIDB1 interface
 *
 * @param  void
 * @return void
 */
void CHU_Initialize ( void )
```

```
74  {
75      /* Place the state machine in its initial state. */
76      chuData.state = CHU_STATE_IDLE;
77
78      /* Initial flags values */
79      chuData.transmit = false;
80      chuData.receive  = false;
81
82      /* Initialize UART communication fifos */
83      FIFO_Initialize(&chuData.fifoDesc_rx, CHU_FIFO_SIZE,
84                          chuData.fifoBuff_rx, 0x00);
85      FIFO_Initialize(&chuData.fifoDesc_tx, CHU_FIFO_SIZE,
86                          chuData.fifoBuff_tx, 0x00);
87
88      /* Config setup for RFIDB1 */
89      chuRfid_config.InputBuffer = chuData.fifoBuff_tx;
90      chuRfid_config.InputBufferSize = CHU_FIFO_SIZE;
91      chuRfid_config.OutputBuffer = chuData.fifoBuff_rx;
92      chuRfid_config.OutputBufferSize = CHU_FIFO_SIZE;
93      chuRfid_config.handleResponse = CHU_RFID_Response;
94      chuRfid_config.handleRequest = CHU_RFID_Request;
95
96      /* Initialise RFIDB1 objects */
97      GetRFIDB1Interface(&chuRfid_interface);
98      chuRfid_interface.Initialise(&chuRfid_object, &chuRfid_config);
99      chuRfid_interface.SetPacketHeaderType(&chuRfid_object, HeaderTypeA);
100 }
101
102 /*****************************************************************************/
103
104 /**
105  * @brief CHU_Tasks
106  *
107  * Execute Chilli state machine, should be called cyclically
108  *
109  * @param  void
110  * @return void
111  */
112 void CHU_Tasks ( void )
113 {
114     /* Check the application's current state. */
115     switch ( chuData.state )
116     {
117         /* Application's initial state. */
118         case CHU_STATE_IDLE:
119         {
120             chuRfid_interface.SendDummyCommand(&chuRfid_object);
121             chuData.state = CHU_STATE_TRANSMIT;
122             break;
123         }
124
125         case CHU_STATE_TRANSMIT:
126         {
127             break;
128         }
129
130         case CHU_STATE_RECEIVE:
131         {
132             break;
133         }
134
135         case CHU_STATE_TRANSLATE:
136         {
137             break;
138         }
139
140         case CHU_STATE_WAIT:
141         {
142             break;
143         }
144
145         /* The default state should never be executed. */
146         default:
```

```
147             {
148                 /* TODO: Handle error in application's state machine. */
149                 break;
150             }
151         }
152     }
153 
154     /*******************************************************************************/
155 
156     /**
157      * @brief CHU_RFID_Response
158      *
159      * Function used by interface library to get a command received by UART
160      * Should not be called by user !
161      *
162      * @param RFIDB1_ObjectT* rfid_object Pointer to RFIDB1 object used by Chilli
163      * @param uint8_t *data Output buffer of data to be receive by UART
164      * @param uint16_t size Size of the buffer
165      */
166     void CHU_RFID_Response( RFIDB1_ObjectT* rfid_object, uint8_t *data, uint16_t size )
167     {
168 
169     }
170 
171     /*******************************************************************************/
172 
173     /**
174      * @brief CHU_RFID_Request
175      *
176      * Function used by interface library to send a command by UART
177      * Should not be called by user !
178      *
179      * @param RFIDB1_ObjectT* rfid_object Pointer to RFIDB1 object used by Chilli
180      * @param uint8_t *data Input buffer of data to be send by UART
181      * @param uint16_t size Size of the buffer
182      */
183     void CHU_RFID_Request( RFIDB1_ObjectT* rfid_object, uint8_t *data, uint16_t size )
184     {
185         /* Local variables declaration */
186         uint8_t i_data;
187         uint8_t freeSize;
188         uint8_t fifoData;
189 
190         /* Get the space available in fifo */
191         freeSize = FIFO_GetWriteSpace(&chuData.fifoDesc_tx);
192 
193         if(freeSize >= size)
194         {
195             /* Add buffer to fifo */
196             for(i_data = 0; i_data < size; i_data++)
197             {
198                 fifoData = *(data + i_data);
199                 FIFO_Add(&chuData.fifoDesc_tx, fifoData);
200             }
201 
202             /* Enable the transmission interrupt */
203             SYS_INT_SourceEnable(CHU_INT_SOURCE_USART_TRANSMIT);
204         }
205     }
206 
207     /*******************************************************************************/
208 
209     /**
210      * @brief CHU_RFID_EnablePolling
211      *
212      * Send a raw command to enable polling
213      * Modifiy function as needed, based on datasheet
214      *
215      * @param  void
216      * @return void
217      */
218     void CHU_RFID_Polling( void )
219     {
```

```
220         /* Local variables declaration */
221         uint8_t packet[26];
222         uint16_t crcHeader;
223         uint16_t crcData;
224 
225         /* Header informations */
226         packet[0] = 0x02;
227         packet[1] = 0x15;
228         packet[2] = 0x00;
229 
230         /* CRC header */
231         crcHeader = GetCCITTCRC(&packet[0], 3);
232         packet[3] = crcHeader & 0x00FF;
233         packet[4] = (crcHeader & 0xFF00) >> 8;
234 
235         /* Command : Polling */
236         packet[5] = 0x22;
237 
238         /* Polling period */
239         packet[6] = (uint8_t)(CHU_POLLING_PERIOD_MS / 100);
240 
241         /* Defined tags */
242         packet[7] = 0x00;    // Number
243         packet[8] = 0x00;    // ASYNC mode
244         packet[9] = 0x00;    // IO Config
245         packet[10] = 0x00;   // PWM
246         packet[11] = 0x00;   // PWM duty
247         packet[12] = 0x00;   // PWM period (msb)
248         packet[13] = 0x00;   // PWM period
249         packet[14] = 0x00;   // PWM period (lsb)
250         packet[15] = 0x00;   // Timeout [100ms]
251 
252         /* Undefined tags */
253         packet[16] = 0x03;   // ASYNC mode
254         packet[17] = 0x22;   // IO Config
255         packet[18] = 0x00;   // PWM
256         packet[19] = 0x32;   // PWM Duty
257         packet[20] = 0x00;   // PWM period (msb)
258         packet[21] = 0x3E;   // PWM period
259         packet[22] = 0xE8;   // PWM period (lsb)
260         packet[23] = 0x32;   // Timeout [100ms]
261 
262         /* CRC data */
263         crcData = GetCCITTCRC(&packet[5], 19);
264         packet[24] = crcData & 0x00FF;
265         packet[25] = (crcData & 0xFF00) >> 8;
266 
267         /* Send command through interface */
268         chuRfid_interface.SendRawDataCommand(&chuRfid_object, packet, 26);
269     }
270 
271     /*****************************************************************************/
272 
273     /**
274      * @brief _IntHandlerDrvUsartInstance1
275      *
276      * Interrupt instance to manage UART communication to RFIDB1 Chilli
277      *
278      */
279     void __ISR(_UART_2_VECTOR, ipl7AUTO) _IntHandlerDrvUsartInstance1(void)
280     {
281         /* Local variables declaration */
282         S_Fifo *RX_fifoDescriptor;
283         S_Fifo *TX_fifoDescriptor;
284         uint8_t TX_size;
285         uint8_t TX_BufferFull;
286         uint8_t dataFifo;
287         USART_ERROR usartStatus;
288 
289         /* Pointers to fifo descriptors */
290         RX_fifoDescriptor = &chuData.fifoDesc_rx;
291         TX_fifoDescriptor = &chuData.fifoDesc_tx;
292 
```

```
293         /* DEBUG */
294         #ifdef DEBUG_CHU_UART
295             DEBUG_LED = true;
296         #endif
297 
298         /* Reading the error interrupt flag */
299         if(SYS_INT_SourceStatusGet(CHU_INT_SOURCE_USART_ERROR))
300         {
301             /* Clear up the error interrupt flag */
302             SYS_INT_SourceStatusClear(CHU_INT_SOURCE_USART_ERROR);
303         }
304 
305          /* Reading the receive interrupt flag */
306         if(SYS_INT_SourceStatusGet(CHU_INT_SOURCE_USART_RECEIVE))
307         {
308             /* Checks overrun or parity error */
309             usartStatus = PLIB_USART_ErrorsGet(CHU_USART_ID);
310 
311             if(usartStatus)
312             {
313                 /* Errors are auto cleaned when read, except overrun */
314                 if ( usartStatus & USART_ERROR_RECEIVER_OVERRUN )
315                 {
316                     PLIB_USART_ReceiverOverrunErrorClear(CHU_USART_ID);
317                 }
318             }
319             else
320             {
321                 chuData.receive = true;
322 
323                 while(PLIB_USART_ReceiverDataIsAvailable(CHU_USART_ID))
324                 {
325                     dataFifo = PLIB_USART_ReceiverByteReceive(CHU_USART_ID);
326                     FIFO_Add(RX_fifoDescriptor, dataFifo);
327                 }
328 
329                 /* Clear up the interrupt flag when buffer is empty */
330                 SYS_INT_SourceStatusClear(CHU_INT_SOURCE_USART_RECEIVE);
331             }
332         }
333 
334         /* Reading the transmit interrupt flag */
335         if(SYS_INT_SourceStatusGet(CHU_INT_SOURCE_USART_TRANSMIT))
336         {
337             TX_size = FIFO_GetReadSpace(TX_fifoDescriptor);
338             TX_BufferFull = PLIB_USART_TransmitterBufferIsFull(CHU_USART_ID);
339 
340             while(TX_size && !TX_BufferFull)
341             {
342                 FIFO_GetData(TX_fifoDescriptor, &dataFifo);
343                 PLIB_USART_TransmitterByteSend(CHU_USART_ID, dataFifo);
344                 TX_size = FIFO_GetReadSpace(TX_fifoDescriptor);
345                 TX_BufferFull = PLIB_USART_TransmitterBufferIsFull(CHU_USART_ID);
346             }
347 
348             /* Disable the interrupt, to avoid calling ISR continuously*/
349             SYS_INT_SourceDisable(CHU_INT_SOURCE_USART_TRANSMIT);
350 
351             /* Clear up the interrupt flag */
352             SYS_INT_SourceStatusClear(CHU_INT_SOURCE_USART_TRANSMIT);
353         }
354 
355         /* DEBUG */
356         #ifdef DEBUG_CHU_UART
357             DEBUG_LED = false;
358         #endif
359     }
360 
361     /*******************************************************************************/
362 
363     /* End of File *****************************************************************/
364 
```