```c
/**
 * \file
 * Functions and types for CRC checks.
 *
 * Generated on Wed Sep 11 13:56:48 2019
 * by pycrc v0.9.2, https://pycrc.org
 * using the configuration:
 *  - Width         = 8
 *  - Poly          = 0x07
 *  - XorIn         = 0x00
 *  - ReflectIn     = False
 *  - XorOut        = 0x00
 *  - ReflectOut    = False
 *  - Algorithm     = bit-by-bit-fast
 *
 * This file defines the functions crc_init(), crc_update() and crc_finalize().
 *
 * The crc_init() function returns the inital \c crc value and must be called
 * before the first call to crc_update().
 * Similarly, the crc_finalize() function must be called after the last call
 * to crc_update(), before the \c crc is being used.
 * is being used.
 *
 * The crc_update() function can be called any number of times (including zero
 * times) in between the crc_init() and crc_finalize() calls.
 *
 * This pseudo-code shows an example usage of the API:
 * \code{.c}
 * crc_t crc;
 * unsigned char data[MAX_DATA_LEN];
 * size_t data_len;
 *
 * crc = crc_init();
 * while ((data_len = read_data(data, MAX_DATA_LEN)) > 0) {
 *     crc = crc_update(crc, data, data_len);
 * }
 * crc = crc_finalize(crc);
 * \endcode
 */
#ifndef CRC_H
#define CRC_H

#include <stdlib.h>
#include <stdint.h>

#ifdef __cplusplus
extern "C" {
#endif


/**
 * The definition of the used algorithm.
```

```
 53   *
 54   * This is not used anywhere in the generated code, but it may be used by the
 55   * application code to call algorithm-specific code, if desired.
 56   */
 57  #define CRC_ALGO_BIT_BY_BIT_FAST 1
 58
 59
 60  /**
 61   * The type of the CRC values.
 62   *
 63   * This type must be big enough to contain at least 8 bits.
 64   */
 65  typedef uint_fast8_t crc_t;
 66
 67
 68  /**
 69   * Calculate the initial crc value.
 70   *
 71   * \return     The initial crc value.
 72   */
 73  static inline crc_t crc_init(void)
 74  {
 75      return 0x00;
 76  }
 77
 78
 79  /**
 80   * Update the crc value with new data.
 81   *
 82   * \param[in] crc      The current crc value.
 83   * \param[in] data     Pointer to a buffer of \a data_len bytes.
 84   * \param[in] data_len Number of bytes in the \a data buffer.
 85   * \return             The updated crc value.
 86   */
 87  crc_t crc_update(crc_t crc, const void *data, size_t data_len);
 88
 89
 90  /**
 91   * Calculate the final crc value.
 92   *
 93   * \param[in] crc  The current crc value.
 94   * \return     The final crc value.
 95   */
 96  static inline crc_t crc_finalize(crc_t crc)
 97  {
 98      return crc;
 99  }
100
101
102  #ifdef __cplusplus
103  }           /* closing brace for extern "C" */
104  #endif
```

```
105
106 #endif      /* CRC_H */
107
```