```c
/*Proposal of an interface for a driver of the L9733 IC
Version: 1.0
Author: Benjamin Åkerlund, May 2021
The L9733 IC may be controlled with an MCU over the SPI-bus, using its definition of input pins:
[MOSI = DI, MISO = DO, CLK = SCLK, SS = CS]
.
.
CONTENTS:
 * Section 1: Data structures and Macro Declarations
 * Section 2: Functions allowing Data to be sent to IC
 * Section 3: Initialising SPI-communication and other functions
 */


#ifndef CODE_BENJAMIN_AKERLUND_H
#define CODE_BENJAMIN_AKERLUND_H


#include <spi.h>


//Section 1: Data structures and Macro Declarations

/*
 This Data Structure would hold information on which output pins (OUT1-8) are active, or HIGH, constantly.
 The data structure may be used when only changing the state of one output pin, and afterwards updating the status
 of the pins. The structure has three different components, each for the different mode of functionality for the IC.
 Eg. ActiveProtect would store information of which output pins have the overcurrent protection enabled, in form of a
 "8-bit integer" such as 00100001. In this example, the third (3.) and last (8.) output pin would be set as HIGH.
 */
typedef struct {
    int ActiveProtect;
    int ActiveDiag;
    int ActiveOutput;
} Status;


```

```
38 /*
39  These Macros contain often used information when sending
   data to the IC. In every 16-bit segment of data sent,
40  the ADDRESS will always be set as 1010. Otherwise
   communication won't happen. The corresponding keywords
   used to
41  select the functionality mode of the IC are then PROTECT
   for overcurrent protection, DIAG for diagnostics and
42  OUTPUT for changing the state of the output pins (OUT1-8)
43  */
44 #define ADDRESS 1010
45 #define PROTECT 1010
46 #define DIAG 0011
47 #define OUTPUT 1100
48
49
50
51 //SECTION 2: Functions allowing Data to be sent to IC
52 /*These function will combine the Address and Keyword (
   corresponding to the writing mode of the IC) with the "
   Drive Mode"
53 of the output pins (OUT1-8) into the correct 16-bit format
    as well as reverse the order either via spi.h or locally,
54 in order to send the LSB first, before finally sending the
    data to the ICs DI pin. The functions might then check
   the status of eg. the overcurrent protected
55 output pins (using the Status structure) and change the
   state, if the selected pin is not already in that state.*/
56
57 /*
58  Enable or Disable overcurrent protection for one output
   pin (OUT1-8) to the wanted state. The function takes as
   input
59  which pin to enable the protection for from in the format
    of an integer, where 1 is the first
60  output pin (OUT1), 2 would be the second output pin (OUT2
   )... and so on.
61  Secondly the function takes the wanted state of 0 or 1 as
    input.
62  */
63 int setProtection(int pin, bool state);
64
65 /*
66  Enable or Disable diagnostic data to be returned to the
   Master/Controller from one output pin (OUT1-8).
67  The function takes as input which pin to enable the
```

```
67 protection for from in the format of an integer,
68  where 1 is the first output pin (OUT1), 2 would be the
   second output pin (OUT2)... and so on.
69  Secondly the function takes the wanted state of 0 or 1
   as input.
70
71  */
72 int setDiagnostics(int pin, bool state);
73
74 /*
75  Sets the output pins to HIGH or LOW (0 or 1).
76  The function takes as input which pin to enable the
   protection for from in the format of an integer,
77  where 1 is the first output pin (OUT1), 2 would be the
   second output pin (OUT2)... and so on.
78  Secondly the function takes the wanted state of 0 or 1
   as input.
79  */
80 int setOutput(int pin, bool state);
81
82 /*
83  This function could be used to change the state of
   multiple pins at once. The function would call on the
   above
84  declared functions multiple times, for each output pin (
   OOT1-7). The function might take as input the wanted
   writing MODE
85  as integers, where 1 might represent Overcurrent
   protection, 2 might represent Diagnostic mode and 3,
   respectively Output mode.
86  The next input would be an 8-bit segment indicating
   which pins the user wants to set as HIGH (1) and which
   ones as LOW (0).
87  This bit might look as follows 00100001. For this case
   the third and last bit would then be set as 1 in the
   wanted mode.
88  */
89 int setMultiplePins(int MODE, int pinStates)
90
91
92
93 //Section 3: Initialising SPI-communication and other
   functions
94 /*This segments contains additional functions that may be
    useful for driving the L9733 IC. In fact the init()
   function
```

```
 95 is essential for driving the IC, and must always be
    executed first.
 96  */
 97
 98 /*
 99  The init() function may be used to initialize the SPI-
    communication for the master and slave (controller and IC
    ).
100  Before this, the pins required by the peripheral
    interface of the L9733 are usually defined to the
    controllers pins.
101  Such as: SCLK, DO, DI, CS
102  .
103  init() would often use the pinMode() function from the
    spi.h library, and would often define the ChipSelect pin,
104  set it to HIGH or 1 to signify not sending any data yet
    , and initiate the clock-signal.
105  The init() function could return a boolean value of TRUE
     or FALSE (1 or 0) as to indicate whether or not the
    initialisation
106  has succeeded. There may also be checks or error
    handling in place.
107  */
108 bool init();
109
110 /*
111  Alike the a functions declared in Section 2, this
    function would allow for sending the same data to
    multiple (if not all)
112  connected peripheral ICs. This may be useful in the case
     of multiple connected ICs for more complex or dynamic
    systems.
113  The function might simply call a in Section 2 declared
    function, after first setting the additional IC's CS pin
    to logical 0.
114  The MODE could be set to represent eg: [1 = Protection,
    2 = Diagnostic, 3 = Output]. The function needs no inputs
    , since these
115  would already be made when calling the section 2
    functions.
116  .
117  This function is unnecessary if only one IC is utilized.
118 */
119 int setMultiple();
120
121 /*
```

```
122  This is an extra function, which may be beside the point
     of this assignment, but I thought I would include it
    anyway.
123  The L9733 has three input pins (IN6-8), which when
    activated allow for sending PWM signal from the OUT6-8
    pins. This
124  may be useful in certain driver applications. However
    this functionality occurs despite the status of the SPI
    command
125  register and is therefore outside of the SPI-bus. The
    function takes as an input the wanted pin out of the (IN6
    -8) and
126  switches on pulse width modulation for that output pin (
    OUT6-8).
127  */
128 int enablePWM(int INpinX);
129
130
131
132 #endif //CODE_BENJAMIN_AKERLUND_H
133
```