intoft25_2023_intoft25_p4_01 / Prototype6_Data_Meetkast_naar_CU / Code / CU_Config_Send / canMeasurementRead.cpp @ 312

View | History | Annotate

```
1  #include "canMeasurementRead.h"
2  #include "debug.h"
3
4  struct can_frame CanMeasurementMsg;
5
6  const uint8_t DIFFERENTPATIENTAMOUNT = 2;
7  const uint8_t DIFFERENTMACHINEAMOUNT = 7;
8  long patientData[DIFFERENTPATIENTAMOUNT][DIFFERENTMACHINEAMOUNT];
9
10 bool printCanMessage(const struct can_frame *frame) {
11   uint8_t resolution = frame->can_dlc;
12   union receivedDataType data;
13   switch(8/resolution)
14     {
15       case AMOUNTOFBYTESINCAN:
16       data.data_as_bytes[0] = frame->data[0];
17       break;
18       case AMOUNTOFWORDINCAN:
19       data.data_as_unsigned_int[0] = frame->data[0];
20       break;
21       case AMOUNTOFLONGINCAN:
22       data.data_as_unsigned_long[0] = frame->data[0];
23       break;
24       case AMOUNTOFLONGLONGINCAN:
25       data.data_as_unsigned_long_long = frame->data[0];
26       break;
27       default:
28       data.data_as_unsigned_long_long = frame->data[0];
29       break;
```

```
30    }
31  // De code hieronder is gebouwd op dat we dus maar 1 waarde per CAN bericht hebben. Als je er meerdere in wilt stoppen z
32  if(debug){
33    if (frame->can_id & CAN_EFF_FLAG) {
34      Serial.print((frame->can_id) & ~CAN_EFF_FLAG, DEC); // Print 29-bit CAN ID
35    } else {
36      Serial.print(frame->can_id, DEC); // Print 11-bit CAN ID
37    }
38      Serial.print(" ");
39      Serial.print("Byte Amount: ");
40      Serial.print(frame->can_dlc, DEC); // print CAN_DLC (amount of data characters/bits)
41      Serial.print(" ");
42      for (uint8_t bytesInMessage = 0; bytesInMessage < frame->can_dlc; bytesInMessage++)  {  // print all CAN_msg.data[] 
43        Serial.print("  -  Data ");
44        Serial.print(bytesInMessage);
45        Serial.print(": ");
46        Serial.print(frame->data[bytesInMessage],DEC);
47      }
48      Serial.println();
49  } else {
50    if (frame->can_id & CAN_EFF_FLAG) {
51      printPatientData((frame->can_id) & ~CAN_EFF_FLAG, frame->data[0]); // Prints data for both patients
52    } else {
53      Serial.print("ID: ");
54      Serial.print(frame->can_id, DEC); // Print 11-bit CAN ID
55      Serial.print(", Data ");
56      Serial.print(": ");
57      Serial.println(frame->data[0], DEC);
58    }
59  }
```

```
60   return 0;
61 }
62
63 bool checkMeasurementMessage() {
64   if(getConfigComplete()){
65     if (mcp2515.readMessage(&CanMeasurementMsg) == MCP2515::ERROR_OK) {
66       printCanMessage(&CanMeasurementMsg);
67     }
68     return 1;
69   } else {
70     return 0;
71   }
72 }
73
74 void printPatientData(uint32_t canId, uint32_t data)
75 {
76   int tempPatientId = rand() % 2; // Data toewijzen aan patiënt voor demo
77   for (int i = 0; i < ROWS; i++) {
78     if(configTable[i].canId == canId)
79     {
80       patientData[tempPatientId][configTable[i].machine]= data;
81     }
82   }
83
84   for (int patientId = 0; patientId < DIFFERENTPATIENTAMOUNT; ++patientId) {
85     for (int machineId = 0; machineId < DIFFERENTMACHINEAMOUNT; ++machineId) {
86       Serial.print(patientData[patientId][machineId]);
87       Serial.print(" ");
88     }
89   }
90   Serial.println();
```

```
91 }
```