HOCHSCHULE HEILBRONN

Verteilte Systeme (304271)

Hochschule Heilbronn – Automotive Systems Engineering

Wintersemester 2022/23

Prof. Dr. -Ing. Ansgar Meroth, M.Sc. Petre Sora

Projektaufgabe

Thermostat

Abgabetermin: 17 Januar 2023

Hoehnel, Moritz (210258)
Automotive Systems Engineering
mhoehnel@stud.hs-heilbronn.de

Ritter, Mattis (210265)
Automotive Systems Engineering
mritterl@stud.hs-heilbronn.de

Inhaltsverzeichnis

1. Einleitung

Um den Komfort im eigenen Haus zu maximieren setzen viele Menschen Smart-Home Systeme ein. Neben dem Komfort kann mit einem solchen System aber auch die Sicherheit oder die Energieeffizienz eines Hauses deutlich gesteigert werden. Gerade in Zeiten steigender Energiepreise und vor dem Hintergrund des Klimawandels ist es wichtig die verfügbaren Ressourcen optimal einzusetzen. Dabei macht die Heizung eines Hauses den größten Anteil aus, weshalb es wichtig ist eine intelligente Temperaturregelung in ein Smart-Home zu integrieren. Ein kommunikationsfähiges Heizungsthermostat ist dabei unverzichtbar und in diesem Projekt wurde uns die Aufgabe zu Teil ein solches zu entwickeln.

Das Heizungsthermostat soll mit einem Servomotor die Heizstufe einstellen und diese mithilfe einer Vollfarb-LED anzeigen. Darüber hinaus verfügt das Thermostat auch über einen eigenen Temperatursensor. Die Kommunikation zu anderen Komponenten des Smart-Home soll über eine CAN-Schnittstelle von statten gehen.

Zu Beginn des Projektes wurde ein Zeitplan bestimmt, gefolgt von dem Festsetzen des Lastenheftes. Nach beenden der Planungsphase, wurde die benötigte Hardware entworfen und hergestellt. Danach wurde der Quellcode erstellt. Der Lösungsansatz, sowie die Funktionalität des Programms werden in dem Kapitel Softwarearchitektur beschrieben, die detaillierte Lösung ist im Kapitel Code Dokumentation zu finden.

2. Projektmanagement

Zum Start des Projektes wurde ein Zeitplan erstellt. Das Team entschied sich einen Phasenplan zu nutzen. Diese Art der Planung ermöglicht eine zeitliche Übersicht des Projekts und berücksichtig Abgabe- und Präsentationsdaten. Für den Phasenplan wurden alle Arbeitsschritte aufgelistet. Das Ende eines Arbeitsschrittes wird durch einen Meilenstein (◆) gekennzeichnet.

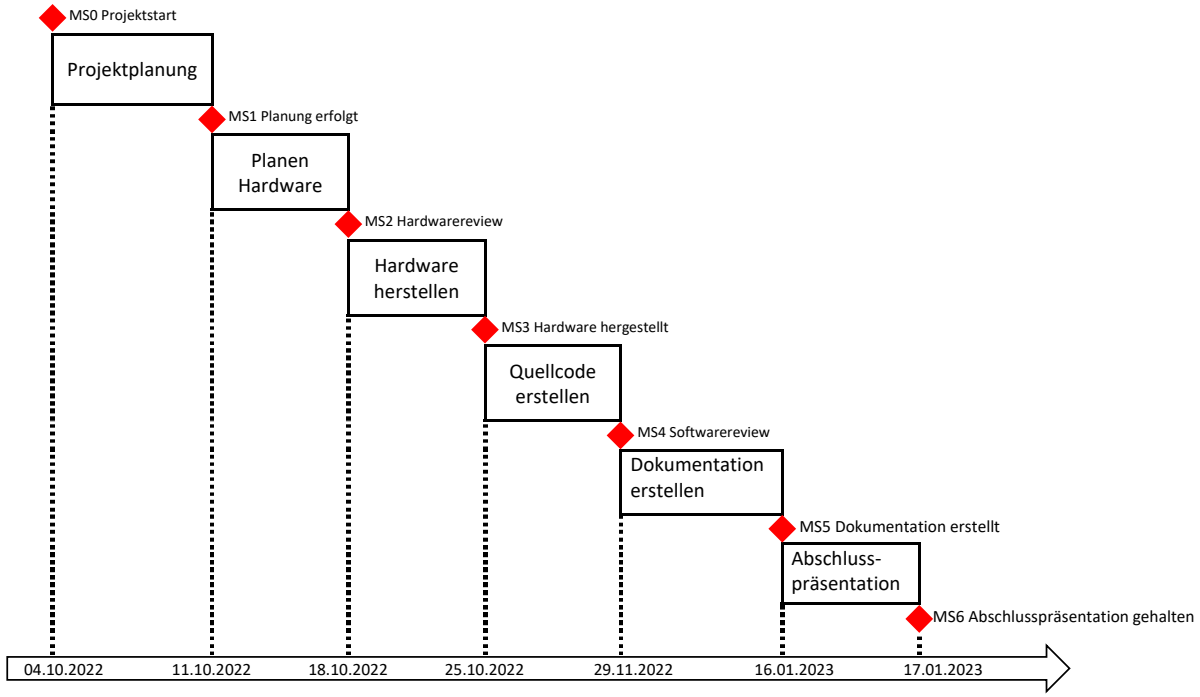


Abbildung 1: Phasenplan

3. Lastenheft

Folgende Anforderungen wurden aus der Aufgabestellung von Herr Sora herausgearbeitet:

- Es wird eine Mehrpunkt-Hysteresenregeleung realisiert.
- Mit PWM wird ein Servo vom Typ FXX-3037-TOP angesteuert.
- Die Temperatur wird mit einem Temperatursensor vom Typ TMP75B gemessen.
- Der Status wird über eine Vollfarb-LED vom TYP WS2812 angezeigt.

Inhalt der CAN-Botschaften:

- Can In: Solltemperatur
- Can Out: Ist Temperatur und Status (z.B. heizt oder Öffnungswinkel).

4. Pflichtenheft

4.1 Einleitung

a. Zweck

Dieses Dokument beschreibt die Pflichten des Projektteams, die aus den Anforderungen der Aufgabenstellung (siehe Lastenhaft) herausgearbeitet wurden. Dabei wurden die Anforderungen genau definiert und erweitert, sowie Verifikationskriterien festgelegt

b. Umfang

Im Rahmen der Kurses sollen Funktionen eines Smart Homes realisiert werden. Dieses Projekt soll ein Thermostat mit Mehrstufiger-Hysterese-Regelung umsetzen.

Es sollen dabei Hardware als auch Software entwickelt werden. Es soll eine zusätzliche Platine für das Microcontroller-Board, welches im Kurs Microcontroller (304132) bereitgestellt wurde, entworfen werden. Die Platine soll die fehlenden Bauteile des Microcontroller-Board beherbergen, welche für die Realisierung des Thermostats notwendig sind.

Es soll eine Software entwickelt werden, welche auf dem Microcontroller genutzt werden kann und die Funktionalität des Produktes herstellt.

c. Erläuterungen zu Begriffen und / oder Abkürzungen

Servo	Servomotor
µC.	Microcontroller

d. Verweise auf sonstige Ressourcen oder Quellen

Bauteile, Schaltplanbibliothek als auch Code-Elemente werden durch die Dozenten des Kurses Verteilte Systeme bereitgestellt.

e. Übersicht

Nach der Einleitung folgt eine allgemeine Beschreibung des Systems, gefolgt von spezifischen Anforderungen und der Verifikation.

4.2 Allgemeine Beschreibung

a. Produktperspektive

Das Produkt soll als Teil eines Smart Homes eine Heizung steuern. Der Nutzer soll die Solltemperatur an dem Microcontroller-Board und an dem Zentralen Display einstellen können. Durch die LED bekommt der Nutzer Feedback, wie stark geheizt wird.

b. Produktfunktionen

Wie bereits erwähnt soll eine Platine und Software entwickelt werden.

Platine: Die Kommunikation zwischen den Bauteilen der Platine und dem Microcontroller-Board soll über Serielle-Schnittstellen realisiert werden. Die Art der Schnittstellen sind durch die jeweiligen Bauteile vorgegeben. Auch die Bauteile sind vorgeben. Generell gibt es ein Bauteil zum Messen der Ist-Temperatur, ein Stellglied für die Heizung und eine LED zur Visualisierung. Darüber hinaus soll ein CAN-Controller verwendet werden. Die vergebenen Bauteile sind in d.Einschränkungen zu finden. Alle weiteren Bauteile (Buchsen, Quarze, Kondensatoren und Widerstände) sollen passend zu den vorher genannten Bauteilen gewählt werden.

Software: Diese soll die Daten des Temperatur-Sensors auswerten. Es soll eine Regler Logik (Sechspunkt Hysterese Regelung) erstellt werden, welche Ausgangssignale zur Steuerung des Servos und der LED erstellt.

c. Benutzermerkmale

Da das Produkt Teil eines Smart-Homes muss jeder Bewohner eines Hauses als Benutzer gewertet werden. Somit sind Nutzer jeden Alters, Bildung, Sprache und Sachkenntnis potenzielle Bediener des Produkts.

d. Einschränkungen

Es sollen folgende Bauteile verwendet werden:

- Servo vom Typ FXX-3037-TOP
- Temperatursensor vom Typ TMP75B
- Vollfarb-LED vom Typ WS2812
- CAN-Controller vom Typ MCP2515

e. Annahmen und Abhängigkeiten

Der Schaltplan und das Board werden mit dem Programm EAGLE erstellt. Die Platine soll vom Auftraggeber gefertigt werden.

Die Software wird mit dem Programm Microchipstudio entwickelt. Die Programmiersprache ist C.

f. Aufteilung der Anforderungen

Eine Vernetzung mit weiteren Funktionen des Smart-Homes ist nicht teil des Projekts, kann aber später realisieret werden.

4.3 Spezifische Anforderungen

a. funktionale Anforderungen

ID	Name	Detaillierte Beschreibung	Werte-bereich	Einheit	Priorität	Status	Verifikations-kriterien
A.1	Stellen	Der Bediener soll dabei die Möglichkeit haben in Ganzzahl-Schritten seine Wunschtemperatur zu stellen.	-55 bis +125	°C	high	offen	Erfolgreiche Vorgabe durch Versuch
A.2	Stellen	Der Bediener soll die Temperatur am µC und am Zentralen Display stellen; Die Stellung am µC soll nur als Back-up verwendet werden			low	offen	Erfolgreiche Änderung der Solltemperatur
A.3	Messen	Der Temperatursensor soll die Ist-Temperatur aufzeichnen	-55 bis +125	°C	high	offen	Prüfen wie in 4. beschrieben
A.4	Messen	Temperatursensor Toleranz	+/-0,1	°C	low	offen	Hersteller-vorgabe
A.5	Stufen	Soll die Heizung stellen	0-5		high	offen	Prüfen wie in 4. beschrieben
A.5.1	Stufe 0	Falls $\Delta\vartheta < 0°C$	0				
A.5.2	Stufe 1	Falls $0°C \leq \Delta\vartheta < 2°C$	1				
A.5.3	Stufe 2	Falls $2°C \leq \Delta\vartheta < 4°C$	2				
A.5.4	Stufe 3	Falls $4°C \leq \Delta\vartheta < 6°C$	3				
A.5.5	Stufe 4	Falls $6°C \leq \Delta\vartheta < 8°C$	4				

A.5.6	Stufe 5	Falls $\Delta\vartheta \geq 8°C$	5				
A.6	Hysterese	An den Schaltpunkten soll eine Hysterese vorhanden sein	+/-0,5	°C	high	offen	Stellt wie in 4. beschrieben
B.1	Servo, Stufe 0	Die Heizung bleibt aus	0	%	high	offen	Ist-Temperatur soll auf einen Wert gestellt werden, dass die jeweiligen Grenzen der Temperatur- bereiche erreicht sind. Der Servo soll die geforderte Stellung erreichen.
B.2	Servo, Stufe 1	Die Heizung wird zu 20% geöffnet	20	%	low	offen	
B.3	Servo, Stufe 2	Die Heizung wird zu 40% geöffnet	40	%	low	offen	
B.4	Servo, Stufe 3	Die Heizung wird zu 60% geöffnet	60	%	low	offen	
B.5	Servo, Stufe 4	Die Heizung wird zu 80% geöffnet	80	%	low	offen	
B.6	Servo, Stufe 5	Die Heizung wird zu 100% geöffnet	100	%	low	offen	
C.1	LED, Stufe 0	LED leuchtet in der Farbe blau	Blau		low	offen	Testdurchführung wie bei Nr. B.X, jeweilige Farben werden wiedergegeben
C.2	LED, Stufe 1	LED leuchtet in der Farbe grün	Grün		low	offen	
C.3	LED, Stufe 2	LED leuchtet in der Farbe gelb	Gelb		low	offen	
C.4	LED, Stufe 3	LED leuchtet in der Farbe orange	Organge		low	offen	
C.5	LED, Stufe 4	LED leuchtet in der Farbe magenta	Magenta		low	offen	
C.6	LED, Stufe 5	LED leuchtet in der Farbe rot	Rot				
A.7	Regelung	Software soll Signal an Servo geben wie geöffnet wird	0-5		high	offen	Stellt wie in 4. beschrieben

A.8	Quellcode	Kommunikation via SPI Schnittstelle mit Servo			high	offen	Kommunikation möglich
A.9	Quellcode	Kommunikation via I^2C mit Temperatursensor			high	offen	Kommunikation möglich
A.10	Quellcode	Kommunikation via Serieller Schnittstelle mit LED			low	offen	Kommunikation möglich
A.11	Quellcode	Auswerten Messwerte, mit Regelung Erstellen eines Output Signals für Servo			high	offen	Servo führt Bewegungen aus
A.12	Quellcode	Melden welche Heizstellung an LED			low	offen	Farbe passt zur Servostellung

i. externe Schnittstellen

ID	Name	Detaillierte Beschreibung	Werte-bereich	Einheit	Priorität	Status	Verifikations-kriterien
D.1	CAN-Out	ID: 0x400; Temperatur Vorkomma; Signed Byte	-128 bis 127	°C	low	offen	Nachricht wird an Display gesendet
D.2	CAN-Out	ID: 0x400; Temperatur Nachkomma; Unsigned Byte	0 bis 9	0,1°C	low	offen	
D.3	CAN-Out	ID: 0x400; Status der Heizung entsprechend der in B.X definierten Stufen; Unsigned Byte	0 bis 5	-	low	offen	
D.4	CAN-In	Id: 0x401; Solltemperatur; Signed Byte	-128 bis 127	°C	high	offen	Nachricht vom Display wird empfangen

ii. Betriebszustände

Sobald das Gerät eingeschalter ist, soll es dauerhaft die Wunschtemperatur regeln. Die Abtastung der Temperatur wird maximal im 100ms Abstand gefordert.

iii. Output

Der Output der Platine ist die Servo-Stellung und die LED.

iv. Umgebungsbedingungen

Es wird festgelegt, dass das Thermostat Einsatztemperaturen von 5-40°C hat. Diese Voraussetzung wird durch die eingeschränkte Nutzung in einem Haus festgelegt. Hier werden keine größeren Temperaturschwankungen erwartet. Es soll vernachlässigt werden, dass Mess- und Stellelektronik nah beieinandersitzen und eventuelle Ist-Temperaturunterschiede zwischen Raummitte und Thermometer entstehen können.

b. nicht-funktionale Anforderungen

i. Anforderungen an Performance

Die Platinentempertur ist nicht zu managen.

ii. Usability

Der Nutzer soll durch Tastendrücke schnell die Temperatur ändern können. Die Stellgeschwindigkeit soll dabei nicht bedeutend langsamer sein als die Heizungsbedienung mit einer herkömmlichen Drehregler Bedienung (Gesamte Dauer zum Ändern der Solltemperatur <= 10s)

iii. Wartbarkeit

Die Platine soll einfach auf das Microcontroller-Board gesteckte werden können.

iv. Änderbarkeit/Skalierbarkeit

Eingriffe in die Software können auch noch nach Beendigung des Projekts durchgeführt werden können.

c. Qualitätsanforderungen

Alle Lötstellen sollen sauber hergestellt werden. Es ist kein Gehäuse gefordert.

4.4 Verifikation

Es soll die Richtigkeit der Temperaturmessung im Labor geprüft werden. Es soll der Messwert des Thermometers mit dem Wert eines herkömmlichen Thermometers zur Ermittlung der Raumtemperatur verglichen werden.

Darüber hinaus soll eine Sichtprüfung durchgeführt werden, welche die richtige Servo-Stellung verifiziert. Messzenario: Per Manipulation soll ein Temperaturdelta gestellt werden, dass alle 5 Stufen durch den Servo gestellt werden.

5. Hardware

Im ersten Entwicklungsschritt wurde ein Schaltplan erstellt. Er zeigt alle Verwendeten Bauteile:

- Servo-Motor
- Temperatursensor
- Mehrfarb LED
- Wannenstecker
- CAN-Controller
- CAN-Transiver
- SUB-D Anschluss

Die ersten drei Bauteile sind Teil des eigentlichen Thermostates. Der Wannenstecker ist die Verbindung zum Microcontroller Board. Die letzten drei Bauteile sind für die Kommunikation mit den anderen Elementen des Smart Homes zuständig.

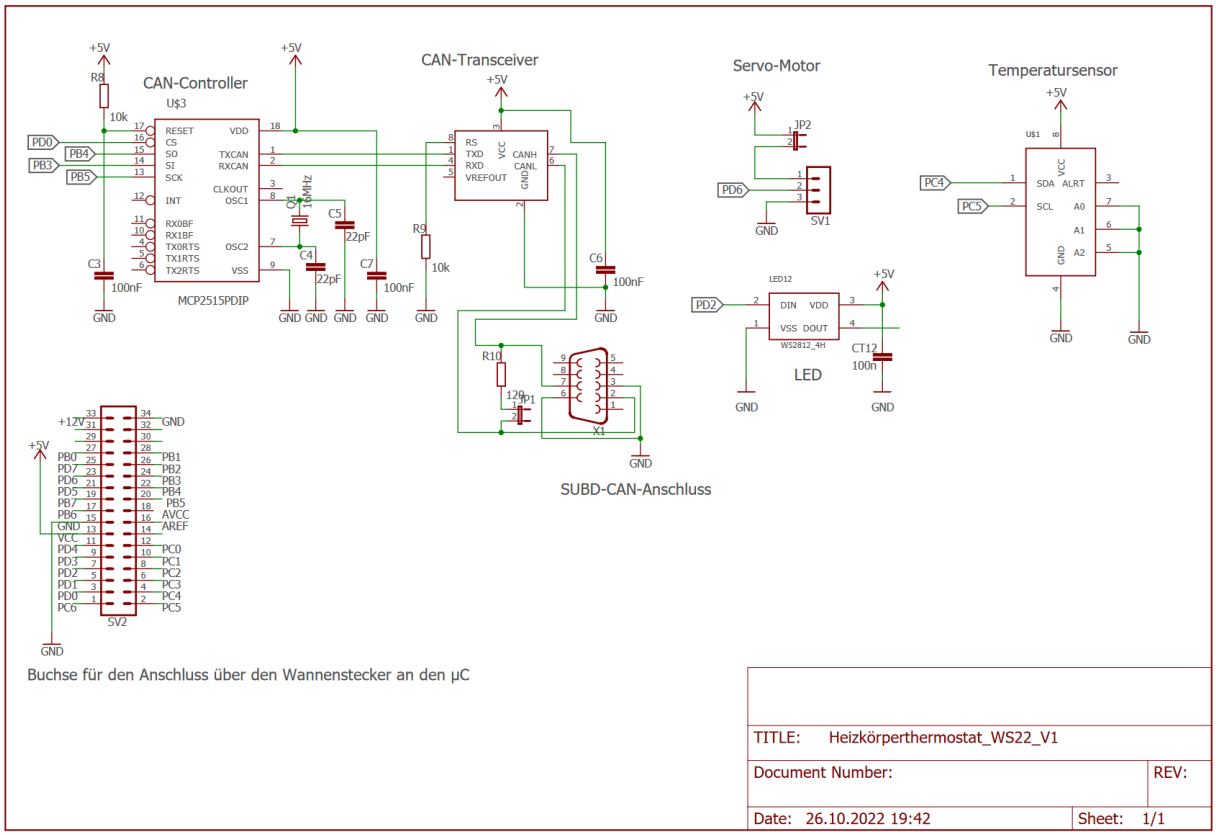


Abbildung 2: Schaltplan

Beim Erstellen des Boards wurde darauf geachtet, dass die Pufferkondensatoren möglichst nah bei den zugehörigen Bauteilen sitzen. Es sollten so wenig wie möglich Durchkontaktierungen gemacht werden.

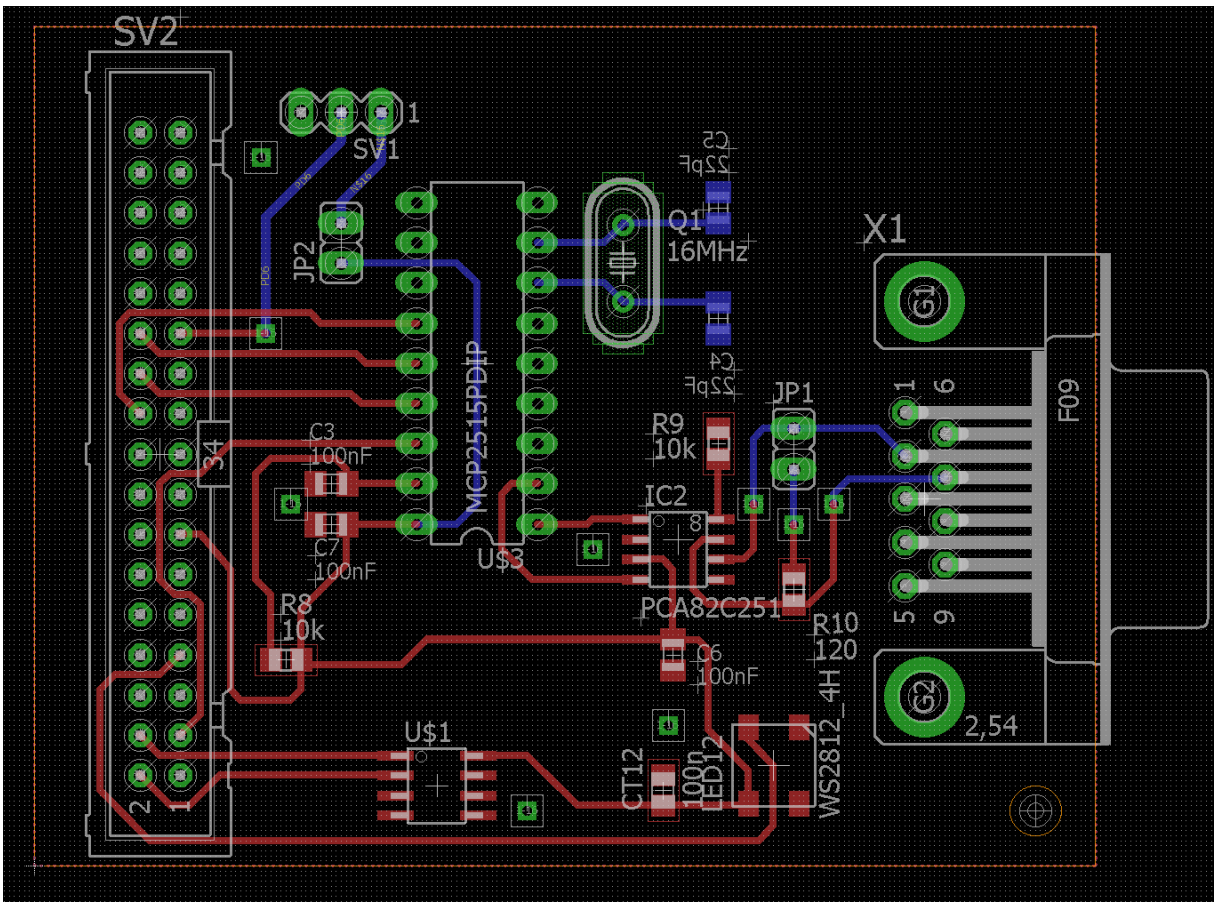


Abbildung 3: Board

6. Softwarearchitektur

Der Quellcode wird für den ATmega 88PA in C geschrieben. Ziel der Software ist die Kommunikation mit den einzelnen Bauteilen herzustellen, eine Regler Logik zu implementieren und den Benutzer Eingaben tätigen zu lassen.

Die Software wird dazu in einzelnen Modulen entwickelt. Jedes Bauteil bekommt dabei sein eigenes Modul. Auch der Regler wird in einem eigenen Modul programmiert. Aus der *main* heraus werden so die Module gestartet. Darüber hinaus gibt es noch ein *Init*-Modul, welches beim Start des Microcontroller alle Module initialisiert.

Im Folgenden werden alle Module beschrieben, die selbst geschrieben worden sind und noch nicht aus dem Kurs Microcontroller bekannt sind.

6.1 Main

Zu Beginn werden der MCP2515, die WS2812 und der TMP75 vorbereitet für den Betrieb. Dies wird durch Definitionen, Defines und Initialisierungen der ersten Messung gemacht.

Nun startet die Hauptschleife. Alle 100 Millisekunden wird darin eine Messung gestartet. Der Wert wird zum einen zur Anzeige an die Displays gesendet und zum anderen in das Regler-Modul, welches eine Regelstufe bestimmt. Diese bestimmt dann die Stellung des Servos und die Farbe der LED.

Ebenfalls alle 100ms wird geprüft ob eine Nachricht über CAN erhalten wurde. Ist dies der Fall und die ID der empfangenen Botschaft entspricht 0x401, dann wird die Solltemperatur auf den Wert der Botschaft gesetzt.

Die aktuelle Temperatur und Heizstufe wird jede Sekunde mit der ID 0x400 über die CAN-Schnittstelle gesendet. (Definition der CAN-Botschaften: Kapitel 4.3a. i. externe Schnittstellen)

Für den Fall, dass noch kein Signal via CAN angekommen ist, kann man die Wunschtemperatur mit den Tastern S1 und S2 auf dem Microcontroller-Board stellen. Diese werden alle 10 Millisekunden abgeprüft. Sobald ein CAN-Signal mit einer Wunschtemperatur angekommen ist, hat dieses Priorität und wird als neue Solltemperatur angenommen. Diese Verhalten sowie die Display-Anzeige in den beiden Modi wird in Abbildung 3 dargestellt.

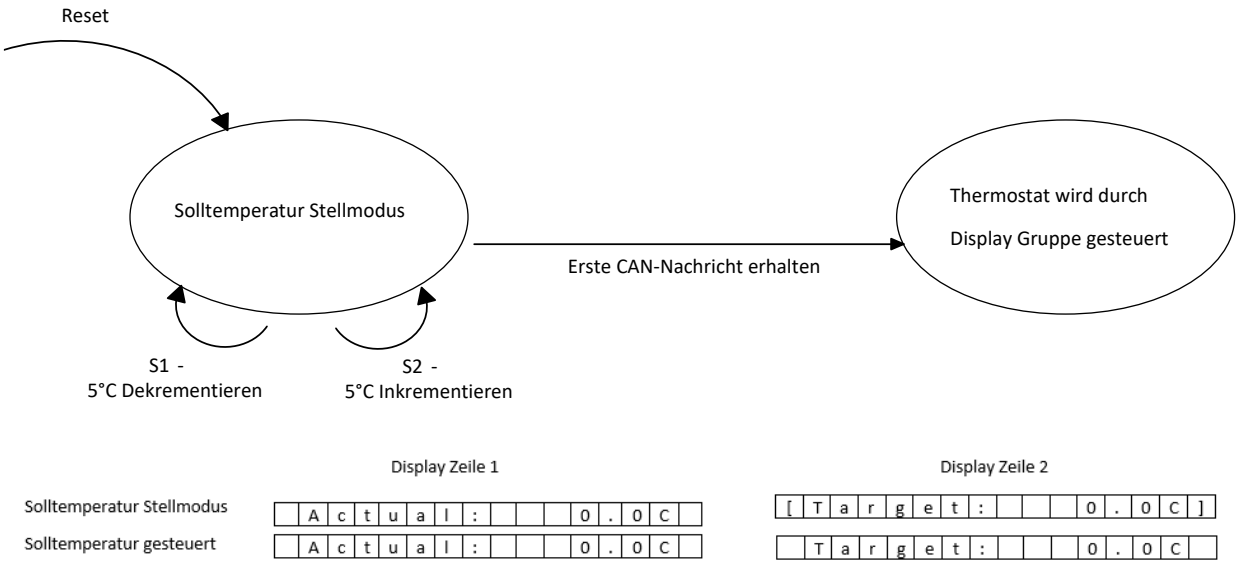


Abbildung 4: Zustandsübergangsdiagramm und Display Anzeige

6.2 Controller

Die Temperaturregelung unseres Thermostats erfolgt mit einer Sechs-Punkt-Hysterese Regelung. Das heißt wir können zwischen Stufe 0, in der die Heizung ausgeschalten ist, und Stufe 5 die Heizleistung schrittweise einstellen. Abhängig von der Regeldifferenz $\Delta\vartheta$, die sich aus der Differenz von Solltemperatur ϑ^* und Ist-Temperatur ϑ berechnet wird die jeweilige Heizstufe gestellt. Um ein ständiges Wechseln der Stufe bei leicht schwankenden Temperaturen zu vermeiden wurde eine Hysterese von $\pm\ 0{,}5°C$ angewendet. Je nach dem ob die Temperaturdifferenz gerade steigt oder sinkt wird nach bzw. vor den festgelegten Schaltpunkten von je $2°C$ geschalten.

In der Funktion *TempController*, welche die Heizstufe zurück gibt, wird dieses Verhalten durch mehrere *if-else-if* Abfragen erreicht. Befindet sich die Temperaturdifferenz in einem Bereich, in dem die Heizstufe nicht eindeutig definiert ist, wird die Heizstufe nicht verändert und der alte Wert zurückgegeben. Um die Initialstufe zu bestimmen wird die Funktion zu Beginn des Programms mit einer Hysterese von Null aufgerufen.

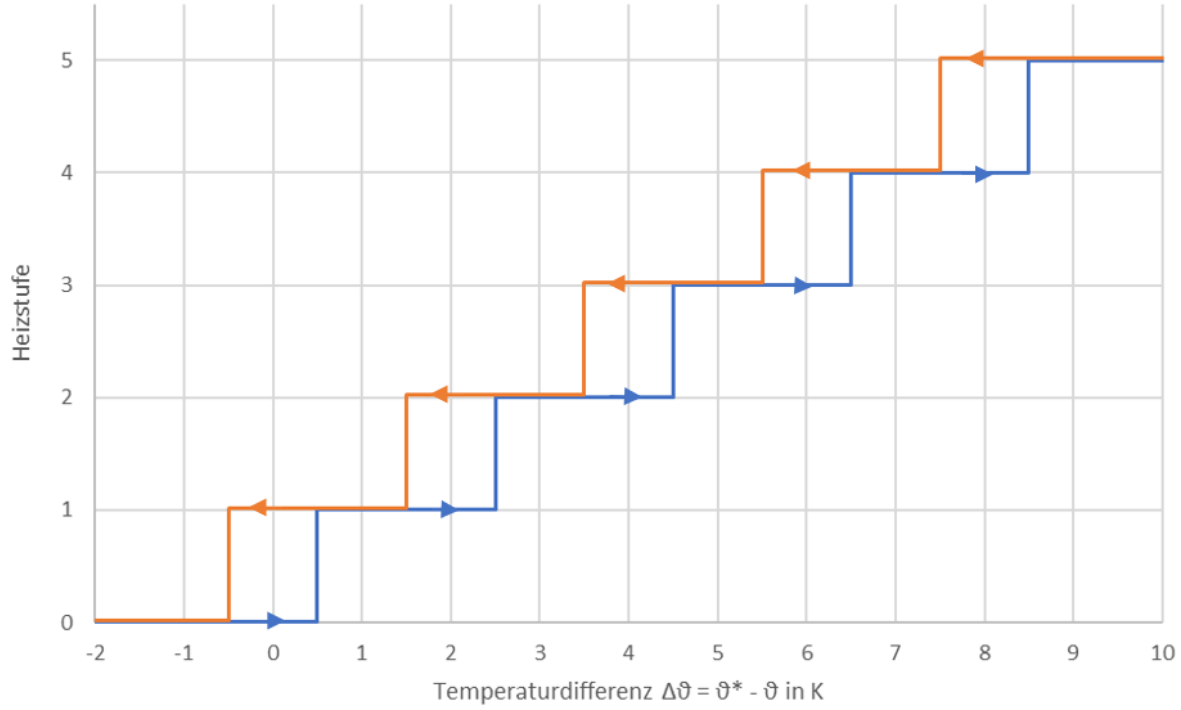


Abbildung 5: Veranschaulichung der Sechs-Punkt-Hysterese-Regelung

6.3 TMP75

Der TMP75 wird zum Messen der Umgebungstemperatur verwendet. Der Microkontroller der Platine kommuniziert via TWI mit dem TMP75. Aus diesem Grund ist das *TWI_Atmega* Modul von Herr Petre Sora in dem Projekt eingebunden. Ebenfalls von Herr Sora wird die *TMP75_Read_Temperature* Funktion verwendet. Dabei wird der Microkontroller als Master initialisiert. Der TMP75 ist Slave. Seine Adresse wurde durch das Verbinden aller Adresspins mit Ground auf 0b1001000 festgelegt. Der Master möchte, dass das Temperatur-Register des TMP75 gesendet wird und fragt deshalb das Register 0b00000000 an. Der Temperaturwert ist in zwei 8 bit Registern geschrieben. Es werden das High und Low Register empfangen und am Ende der Funktion zusammengesetzt. Der Temperaturwert ist als Modulvariable

gespeichert. Diese Funktion muss jedes Mal in der *main* aufgerufen werden, wenn man eine Messung der Umgebungstemperatur möchte. Damit ein Überhitzen des TMP75 durch Überlastung verhindert wird, startet die *main* alle 100ms eine Messung.

Durch die zweite Funktion *TMP75_Get_Temperature* kann die Modulvariable mit der Temperatur in der Main abgefragt werden. Es wird dabei eine Nachkommastelle betrachtet. Der übergebene Wert ist mit dem Faktor 10 versehen, sodass einen *floatingpoint* Datentyp vermieden wird. Der Wert wird vor der Rückgabe, wie im Datenblatt des TMP75 beschrieben, bearbeitet.

6.4 WS2812

Dieses Modul ist für die Ansteuerung der Mehrfarb-LED zuständig. Der Code wurde von Herr Petre Sora bereitgestellt und für die Anwendung angepasst. In der Init Funktion wird der PORT PD2 zum Ausgang der LED definiert. Es wird dabei auf die Definition für den *WS2812_pin* (*WS2812_1*) aus der *main.h* zugegriffen.

In der *WS2812_step* Funktion wird aus einem 6x3 Array eine Zeile aufgerufen und als Input in die *WS2812_Set_Colour* gegeben. Dieses Array hat die Länge drei und gibt die Zusammensetzung der Farbe an. Es werden die Werte für grün, rot und blau übergeben. Die *WS2812_Set_Colour* nimmt noch einen weiteren Input-Wert. Dieser beschreibt die Position der LED und ist in unserem Fall immer 2. Zu Beginn der *WS2812_Set_Colour* Funktion wird der anzusteuernde Pin auf Null gesetzt, erst danach kann die neue Farbe gesetzt werden.

Die Farben, in welcher die Mehrfarb-LED je nach Heizstufe leuchtet sind wie in Tabelle 1 zu sehen definiert.

Stufe	Farbe	G	R	B
0	Blau	0	0	7
1	Grün	7	0	0
2	Gelb	3	4	0
3	Orange	2	4	1
4	Magenta	0	4	3
5	Rot	0	7	0

Tabelle 1: WS2812

6.5 Servo-Ansteuerung

Zur Ansteuerung des Servos wird ein PWM-Signal an PD6 bzw. OC0A benötigt. Aus dem Datenblatt des Servomotors lässt sich entnehmen, dass der Servo mit einer Periodendauer von $T = 20ms$, was einer Frequenz von $f = 50Hz$ entspricht, angesteuert werden soll.

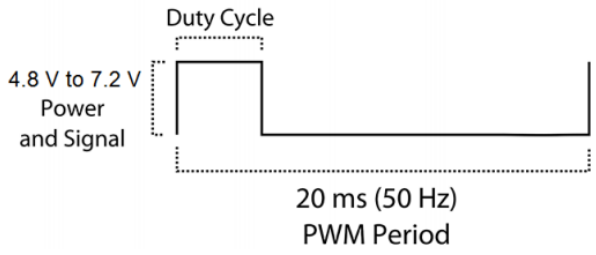


Abbildung 6: Duty-Cycle und Periodendauer

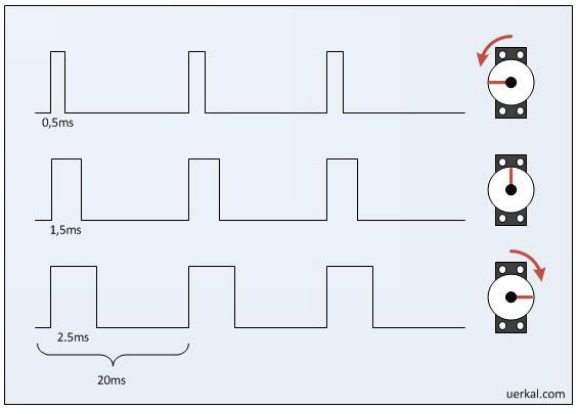


Abbildung 7: Pulsweite und Servowinkel

Timer0 wird im nicht-invertierenden Fast-PWM Modus betrieben. Die reale Periodendauer lässt sich dabei nicht auf die geforderten $T = 20ms$ einstellen, deshalb wurde diese so gut wie möglich approximiert. Mit dem Prescaler von 1024 kann folgender Wert erreicht werden:

$$f_{PWM} = \frac{f_{OSC}}{N \cdot 256} = \frac{18{,}432MHz}{1024 \cdot 256} = 70{,}3125Hz \rightarrow T_{PWM} = 14{,}222ms$$

Um den Winkel des Servomotors in den Stufen 0-5 einzustellen muss die Dauer des HIGH-Pulses angepasst werden in dem der Wert des *OCR0A* Registers beschrieben wird. Die Berechnung des Wertes erfolgt nach folgender Formel:

$$OCR0A = \frac{T_{Puls}}{T_{PWM}} \cdot 256 \text{ , mit } T_{puls} = -Stufe \cdot 0{,}4ms + 2{,}5ms$$

Die daraus berechneten Werte wurden in Tabelle 2 festgehalten, wobei, der berechnete Wert in den Endpositionen etwas angepasst wurde um Ruckeln zu vermeiden (in Klammern).

Der Servo wird wie oben beschrieben in der Funktion *Servo_Init* initialisiert und in der *Servo_Step* Funktion wird der *OCR0A* Wert verändert, um den Servo in die passende Position zu stellen.

Stufe	Pulsweite	OCR0A
0	$2{,}5ms$	45 (44)
1	$2{,}1ms$	38
2	$1{,}7ms$	30
3	$1{,}3ms$	23
4	$0{,}9ms$	16
5	$0{,}5ms$	9 (10)

Tabelle 2: Servo

7. Fazit

Das Projekt wurde erfolgreich beendet und alle Anforderungen wurden erfüllt. Allerdings sind wir während des Projekts auf einige Fehler gestoßen. Zum einen passen die Anschlüsse des Servos nicht zum Anschluss auf der Platine. Dieser Fehler ist entstanden, da wir den empfohlenen Anschluss nicht mit den Angaben des Datenblatts abgeglichen haben. Um dieses Problem zu beheben haben wir die PWM und VCC Leitung des Servos vertauscht. Ein weiteres Problem tritt während des Betriebs des Thermostats auf, denn der Temperatursensor erhitz sich dabei. Dieses Problem haben wir nicht beseitigt. Die Beste Lösung wäre es den Temperatursensor thermisch vom Rest der Platine zu entkoppeln, denn selbst wenn der Sensor sich nicht von der eigenen Umwandlungen und Rechenprozessen erhitzt, erwärmen sich die anderen Bauteile auf der Platine im Betrieb. Die Thermische Entkopplung könnte auf zwei Wege realisiert werden. Entweder kann der Temperatursensor auf eine Breakoutplatine ausgelagert werden oder man platziert ihn nicht auf der gemeinsamen Massefläche der Platine, sondern sieht hierfür einen getrennten Bereich vor. Die genannten Probleme sollten in künftigen Projekten während des Hardwaredesigns behoben werden.

8. Code Dokumentation

Die Code Dokumentation ist mit dem Tool Doxygen erstellt. Es wurden nur die Module mit ausführlichen Dokumentationskommentar versehen, welche durch das Team erstellt wurden. Code der durch die Dozenten erstellt wurde ist ohne Erklärung eingefügt.

Für die Funktionsbeschreibung wurde sich geeinigt, dass eine Kurzbescheibung, Parameter (in und out), return-Wert, Version, Datum und Autor in Doxygen-Kommentar aufgeschrieben werden.

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all documented files with brief descriptions:

Data Structure Documentation

can_filter Struct Reference

Data Fields

- uint8_t **RecBuff_ID** [2]
- uint8_t **Rec_Buff0_Rollover**
- uint8_t **Filter_RecBuff** [2]
- uint32_t **ulRecBuff_Filter** [6]
- uint32_t **ulRecBuff_Mask** [2]

The documentation for this struct was generated from the following file:

- MCP2515_HHN.h

can_frame Struct Reference

Data Fields

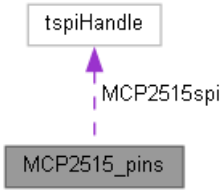
- uint8_t **EIDE_Bit**
- uint8_t **RTR_Bit**
- uint32_t **ulID**
- uint8_t **ucLength**
- uint8_t **ucData** [8]

The documentation for this struct was generated from the following file:

- MCP2515_HHN.h

MCP2515_pins Struct Reference

Collaboration diagram for MCP2515_pins:



Data Fields

- **tspiHandle MCP2515spi**

The documentation for this struct was generated from the following file:

- MCP2515_HHN.h

tspiHandle Struct Reference

Data Fields

- volatile uint8_t * **CS_DDR**
- volatile uint8_t * **CS_PORT**
- uint8_t **CS_pin**
- uint8_t **CS_state**

The documentation for this struct was generated from the following file:

- SPI.h

WS2812_pin Struct Reference

Data Fields

- volatile uint8_t * **WS2812_DDRReg**
- volatile uint8_t * **WS2812_PORTReg**
- uint8_t **WS2812_Pin**

The documentation for this struct was generated from the following file:

- WS2812.h

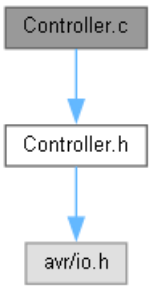
File Documentation

Controller.c File Reference

C file to implement Mulitple position controller with hysteresis.

```
#include "Controller.h"
```

Include dependency graph for Controller.c:



Functions

- unsigned char **TempController** (int actualTemp, int targetTemp, unsigned char stepOld, unsigned char ucHysteresis)
  Initialization of keys.

Detailed Description

C file to implement Mulitple position controller with hysteresis.

Date
05.11.2022 13:01:33

Author
Hoehnel and Ritter

Function Documentation

unsigned char TempController (int actualTemp*, int* targetTemp*, unsigned char* stepOld*, unsigned char* ucHysteresis*)*

Initialization of keys.

Parameters

in	*int*	actualTemp: Measured temperature
in	*int*	targetTemp: target Temp: Wished temperature

in	*unsigned_char*	stepOld: Previous heating step
in	*unsigned_char*	ucHysteresis: Variable to implement hysteresis

Returns

[unsigned char] Value which heating should be at

Date

4.11.22

Author

Hoehnel and Ritter

Version

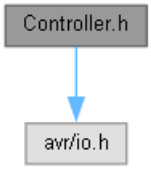
1.0

Controller.h File Reference

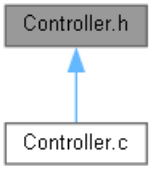
Include File for Software Controller for thermostat.

```
#include <avr/io.h>
```

Include dependency graph for Controller.h:



This graph shows which files directly or indirectly include this file:



Functions

- unsigned char **TempController** (int actualTemp, int targetTemp, unsigned char stepOld, unsigned char ucHysteresis)
  Initialization of keys.

Detailed Description

Include File for Software Controller for thermostat.

Date

05.11.2022 13:01:15

Author

Hoehnel and Ritter

Function Documentation

unsigned char TempController (int actualTemp*, int* targetTemp*, unsigned char* stepOld*, unsigned char* ucHysteresis*)*

Initialization of keys.

Parameters

in	*int*	actualTemp: Measured temperature
in	*int*	targetTemp: target Temp: Wished temperature
in	*unsigned_char*	stepOld: Previous heating step
in	*unsigned_char*	ucHysteresis: Variable to implement hysteresis

Returns

[unsigned char] Value which heating should be at

Date

4.11.22

Author

Hoehnel and Ritter

Version

1.0

Controller.h

Go to the documentation of this file.

```
9 #ifndef CONTROLLER_H_
10 #define CONTROLLER_H_
11
12 //Includes
13 #include <avr/io.h>
14
15
16 //Defines
17
18
19 //Deklaration of functions
20 unsigned char TempController(int actualTemp, int targetTemp, unsigned char stepOld,
unsigned char ucHysteresis);
21
22
23 #endif /* CONTROLLER_H_ */
```

display_funktionen.h

```
1 #include <avr/io.h>
2 #include <avr/interrupt.h>
3
4 #pragma GCC push_options
5 #pragma GCC optimize("O0")
6
7 //------------------------------------------------------------------------------------
------------
8 //Funktionen
9 //------------------------------------------------------------------------------------
------------
10 void Display_delay(unsigned long delay_time_us);
11
12 void Display_Aus(void);
13 void Display_An(void);
14 void Display_Init(void);
15
16 void Display_HardwareInit(void);
17
18 void Display_RS_Output(void);
19 void Display_RS_High(void);
20 void Display_RS_Low(void);
21
22 void Display_EN_Output(void);
23 void Display_EN_High(void);
24 void Display_EN_Low(void);
25
26 void Display_DATA_Output(void);
27 void Display_DATA_BitHigh(unsigned char DataBit);
28 void Display_DATA_BitLow(unsigned char DataBit);
29
30 void Display_Clear(void);
31 void Display_ReturnHome(void);
32 void Display_ModeEntry(unsigned char Options);
33 void Display_Control(unsigned char Options);
34 void Display_CursorOrDisplayShift(unsigned char Options);
35 void Display_SetMPUInterface(unsigned char Options);
36 void Display_SetCursor(unsigned char row, unsigned char column);
37 void Display_Transfer4BitData(unsigned char Data);
38 void Display_Write(unsigned char ASCII_of_char);
39 void Display_Print(unsigned char* text2print, unsigned char length);
40 void Display_GenerateNewChar(unsigned char address, unsigned char*
Pattern_New_Char);
41 void Display_Output(int iTemp2print, unsigned char ucLine, unsigned char ucCAN);
42
43 #pragma GCC pop_options
44
45 //------------------------------------------------------------------------------------
-------------
46 //#define's für die verwendeten Funktionen
47 //------------------------------------------------------------------------------------
-------------
48
49 //Display_Clear
50 #define DISPLAY_CLEAR_FUNCTION                  0x01
51 #define DISPLAY_CLEAR_DISPLAY_DELAY             4000
52 //Display_ReturnHome
53 #define DISPLAY_RETURN_HOME_FUNCTION            0x02
54 #define DISPLAY_RETURN_HOME_DELAY               2000
55 //Display_ModeEntry
56 #define DISPLAY_MODE_INCR_SHIFT_OFF             0x06
57 #define DISPLAY MODE INCR SHIFT ON              0x07
58 #define DISPLAY_MODE_DECR_SHIFT_OFF             0x04
59 #define DISPLAY_MODE_DECR_SHIFT_ON              0x05
60 #define DISPLAY_MODE_DELAY 50
61 //Display_Control
62 #define DISPLAY OFF                             0x08
63 #define DISPLAY_ON_CURSOR_OFF                   0x0C
64 #define DISPLAY_ON_CURSOR_ON_BLINK_OFF          0x0E
65 #define DISPLAY_ON_CURSOR_ON_BLINK_ON           0x0F
66 #define DISPLAY_CONTROL_DELAY 50
```

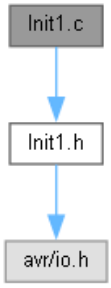
```
67 //Display_CursorOrDisplayShift
68 #define DISPLAY_SHIFT_CURSOR_RECHTS             0x14
69 #define DISPLAY_SHIFT_CURSOR_LINKS              0x10
70 #define DISPLAY_SHIFT_DISPLAY_RECHTS            0x1C
71 #define DISPLAY_SHIFT_DISPLAY_LINKS             0x18
72 #define DISPLAY_CURSOR_OR_DISPLAY_SHIFT_DELAY   50
73 //Display_FunctionSet
74 #define DISPLAY_MPU_4BIT_2_LINES_5x7_DOTS       0x28
75 #define DISPLAY_SET_MPU_INTERFACE_DELAY         50  //100
76
77 #define DISPLAY_FUNKTION_SET_DDRAM_ADRESSE      0x80
78 #define DISPLAY_FUNKTION_SET_CGRAM_ADRESSE      0x40
79 #define DISPLAY_SET_RAM_ADRESSE_DELAY           50
80
81 //-------------------------------------------------------------------------------------
-------------
82 //Definitionen für das µECU-Board
83 //dieser Teil muss neu definiert werden für ein anderes Board
84 //die 4 Datenbits müssen zum gleichen Port gehören
85 //-------------------------------------------------------------------------------------
-------------
86 #define FREQ_CPU 18432000UL
87
88 #define DISPLAY_SW_DDR_REG      DDRD
89 #define DISPLAY_SW_PORT_REG     PORTD
90 #define DISPLAY_SW_BIT          PD7
91
92 #define DISPLAY_RS_DDR_REG      DDRB
93 #define DISPLAY_RS_PORT_REG     PORTB
94 #define DISPLAY_RS_BIT          PB0
95
96 #define DISPLAY_EN_DDR_REG      DDRB
97 #define DISPLAY_EN_PORT_REG     PORTB
98 #define DISPLAY EN BIT          PB1
99
100 #define DISPLAY_DATA_DDR_REG        DDRC
101 #define DISPLAY_DATA_PORT_REG       PORTC
102 #define DISPLAY_DATA_DB7_BIT        PC3
103 #define DISPLAY_DATA_DB6_BIT        PC2
104 #define DISPLAY DATA DB5 BIT        PC1
105 #define DISPLAY_DATA_DB4_BIT        PC0
106
107 #define DATAPORTMASKE (~((1 << DISPLAY_DATA_DB7_BIT) | (1 << DISPLAY_DATA_DB6_BIT)
| (1 << DISPLAY_DATA_DB5_BIT) | (1 << DISPLAY_DATA_DB4_BIT)))
108
109
110
```

Init1.c File Reference

C file to initiate modules.

```
#include "Init1.h"
```

Include dependency graph for Init1.c:



Functions

- void **GerneralInit** (void)
  Execute all inits.

- void **HexToAscii** (unsigned char input, unsigned char *output)
  Changes hex to ascii.

Detailed Description

C file to initiate modules.

Date

26.10.2022 20:38:01

Author

Hoehnel and Ritter

Function Documentation

void GerneralInit (void)

Execute all inits.

Parameters

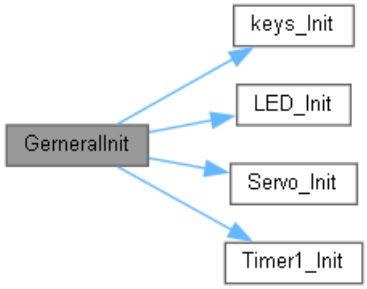
in	*None*	

Returns

None

Date
27.10.2022
Author
Hoehnel and Ritter
Version
1.0

Here is the call graph for this function:



void HexToAscii (unsigned char input*, unsigned char ** output*)*

Changes hex to ascii.

Parameters

in	*unsigned_char*	input: Element to convert
out	*unsigned_char*	output: Converted element

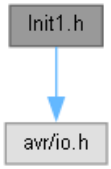
Returns
None
Date
27.10.2022
Author
Meroth
Version
1.0

Init1.h File Reference

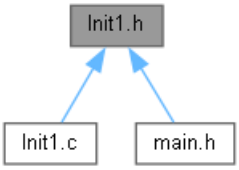
Include file to initiate modules.

```
#include <avr/io.h>
```

Include dependency graph for Init1.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **GeneralInit** (void)
- void **HexToAscii** (unsigned char input, unsigned char *output)
  Changes hex to ascii.

Detailed Description

Include file to initiate modules.

Date

26.10.2022 20:38:01

Author

Hoehnel and Ritter

Function Documentation

void HexToAscii (unsigned char input*, unsigned char** output*)*

Changes hex to ascii.

Parameters

in	*unsigned_char*	input: Element to convert

out	*unsigned_char*	output: Converted element

Returns

None

Date

27.10.2022

Author

Meroth

Version

1.0

Init1.h

Go to the documentation of this file.

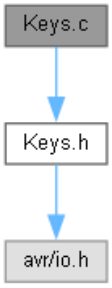
```
9  #ifndef INIT1_H_
10 #define INIT1_H_
11
12
13 //Includes
14 #include <avr/io.h>
15
16
17 //Defines
18
19 //Deklaration of funtions
20 void GeneralInit(void);
21
22 //void CAN_Filter_Init(void);
23 //void ShowMessage(can frame *showFrame);
24 void HexToAscii(unsigned char input, unsigned char* output);
25
26 #endif /* INIT1_H_ */
```

Keys.c File Reference

C file to Read Keys.

```
#include "Keys.h"
```

Include dependency graph for Keys.c:



Functions

- void **keys_Init** (void)
  Initialization of keys.

- unsigned char **keys_get_state** (void)
  Gets current state of keys.

Variables

- unsigned char **ucS1_old**
- unsigned char **ucS1_new** = 0xFF
- unsigned char **ucS2_old**
- unsigned char **ucS2_new** = 0xFF
- unsigned char **ucS3_old**
- unsigned char **ucS3_new** = 0xFF

Detailed Description

C file to Read Keys.

Comment

S4 used for LED -> deactivated

Date

08.04.2022 10:05:15

Author

Hoehnel and Ritter

Function Documentation

unsigned char keys_get_state (void)

Gets current state of keys.

Parameters

in	*None*	

Returns

[unsigned char] says which key is pressed

Date

08.04.2022

Author

Hoehnel and Ritter

Version

3.0

void keys_Init (void)

Initialization of keys.

Parameters

in	*None*	

Returns

None

Date

08.04.2022

Author

Hoehnel and Ritter

Version

1.0

Here is the caller graph for this function:

Variable Documentation

unsigned char ucS1_new = 0xFF

Variable for S1

unsigned char ucS1_old

Variable to store old value of S1

unsigned char ucS2_new = 0xFF

Variable for S2

unsigned char ucS2_old

Variable to store old value of S2

unsigned char ucS3_new = 0xFF

Variable for S3

unsigned char ucS3_old

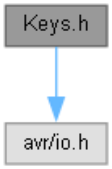
Variable to store old value of S2

Keys.h File Reference

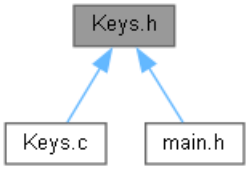
Include file to Read Keys.

```
#include <avr/io.h>
```

Include dependency graph for Keys.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **S1_PRESSED** 1
- #define **S2_PRESSED** 2
- #define **S3_PRESSED** 3
- #define **KEYS_NOT_PRESSED** 0

Functions

- void **keys_Init** (void)
  Initialization of keys.

- unsigned char **keys_get_state** (void)
  Gets current state of keys.

Detailed Description

Include file to Read Keys.

Date

08.04.2022 10:00:02

Author

Hoehnel and Ritter

Macro Definition Documentation

#define KEYS_NOT_PRESSED 0

define that 0 is no key pressed

#define S1_PRESSED 1

define that 1 is key 1

#define S2_PRESSED 2

define that 2 is key 2

#define S3_PRESSED 3

define that 3 is key 3

Function Documentation

unsigned char keys_get_state (void)

Gets current state of keys.

Parameters

in	*None*	

Returns

[unsigned char] says which key is pressed

Date

08.04.2022

Author

Hoehnel and Ritter

Version

3.0

void keys_Init (void)

Initialization of keys.

Parameters

in	*None*	

Returns

None

Date

08.04.2022

Author

Hoehnel and Ritter

Version

1.0

Here is the caller graph for this function:


GernerallInit
keys_Init

Keys.h

Go to the documentation of this file.

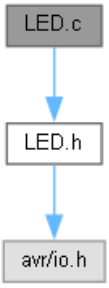
```
9  #ifndef KEYS_H_
10 #define KEYS_H_
11
12 //Includes
13 #include <avr/io.h>
14
15 //Defines (Zu beginn der Programmausführung)
16 #define S1_PRESSED          1
17 #define S2_PRESSED          2
18 #define S3_PRESSED          3
19 //#define S4_PRESSED            4
20 #define KEYS_NOT_PRESSED    0
22 //Declaration of funktions
23 void keys_Init(void);
24 unsigned char keys get state(void);
25
26
27 #endif /* KEYS_H_ */
```

LED.c File Reference

C file to control LEDs.

```
#include "LED.h"
```

Include dependency graph for LED.c:



Functions

- void **LED_Init** (void)
  Initialization of LEDs.
- void **LED_rd_on** (void)
  Switch on red LED.
- void **LED_rd_off** (void)
  Switch off red LED.
- void **LED_rd_toggle** (void)
  Toggle red LED.

Detailed Description

C file to control LEDs.

Comment

GREEN LED switched off, clashing with servo

Date

30.03.2022 12:17:13

Author

Hoehnel and Ritter

Function Documentation

void LED_Init (void)

Initialization of LEDs.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

1.0

Here is the caller graph for this function:

void LED_rd_off (void)

Switch off red LED.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

1.0

void LED_rd_on (void)

Switch on red LED.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

1.0

void LED_rd_toggle (void)

Toggle red LED.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

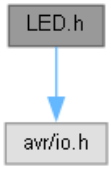
1.0

LED.h File Reference

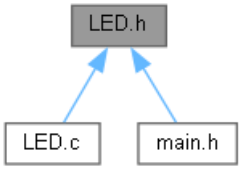
Include file to control LEDs.

```
#include <avr/io.h>
```

Include dependency graph for LED.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **LED_Init** (void)
  Initialization of LEDs.
- void **LED_rd_on** (void)
  Switch on red LED.
- void **LED_rd_off** (void)
  Switch off red LED.
- void **LED_rd_toggle** (void)
  Toggle red LED.

Detailed Description

Include file to control LEDs.

Date

30.03.2022 12:17:13

Author

Hoehnel and Ritter

Function Documentation

void LED_Init (void)

Initialization of LEDs.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

1.0

Here is the caller graph for this function:

void LED_rd_off (void)

Switch off red LED.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

1.0

void LED_rd_on (void)

Switch on red LED.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

1.0

void LED_rd_toggle (void)

Toggle red LED.

Parameters

in	*None*	

Returns

None

Date

30.03.2022

Author

Hoehnel and Ritter

Version

1.0

LED.h

Go to the documentation of this file.

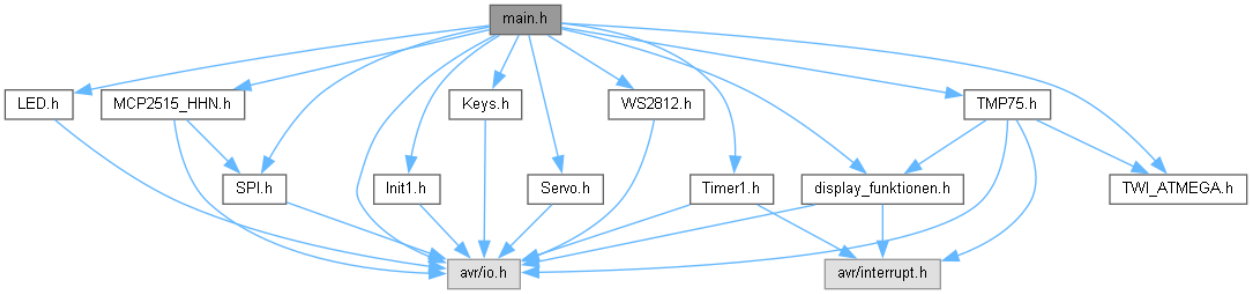
```
9  #ifndef LED_H_
10 #define LED_H_
11
12 //Includes
13 #include <avr/io.h>
14
15 //Defines
16
17
18 //Declaration of funktions
19 void LED_Init(void);
20 //void LED_gn_on(void);
21 //void LED_gn_off(void);
22 void LED_rd_on(void);
23 void LED_rd_off(void);
24 void LED_rd_toggle(void);
25 //void LED_gn_toggle(void);
26
27
28
29 #endif /* LED_H_ */
```

main.h File Reference

Include file for main.

```
#include <avr/io.h>
#include "Init1.h"
#include "Servo.h"
#include "WS2812.h"
#include "display_funktionen.h"
#include "LED.h"
#include "MCP2515_HHN.h"
#include "SPI.h"
#include "TMP75.h"
#include "TWI_ATMEGA.h"
#include "Keys.h"
#include "Timer1.h"
```

Include dependency graph for main.h:



Macros

- #define **CAN_NOT_RECEIVED** 0
- #define **CAN_RECEIVED** 1

Variables

- **WS2812_pin WS2812_1**
- **MCP2515_pins MCP2515_1**

Detailed Description

Include file for main.

Date

26.10.2022 19:54:57

Author

Hoehnel and Ritter

Macro Definition Documentation

#define CAN_NOT_RECEIVED 0

define that 0, no message received via can

#define CAN_RECEIVED 1

define that 1, message received via can

Variable Documentation

MCP2515_pins MCP2515_1

```
Initial value:= {{          &DDRD,
                                  &PORTD,
                                  PD0,
                              ON}}
```

define that MCP2515 is using PD0 of PORT D

WS2812_pin WS2812_1

```
Initial value:=   {   &DDRD,
                          &PORTD,
                                    PD2}
```

define that WS2812 is using PD2 of PORT D

main.h

Go to the documentation of this file.[1]

```
9 #ifndef MAIN_H_
10 #define MAIN_H_
11
12 //Includes
13 #include <avr/io.h>
14
15
16 #include "Init1.h"
17 #include "Servo.h"
18 #include "WS2812.h"
19 #include "display_funktionen.h"
20 #include "LED.h"
21 #include "MCP2515_HHN.h"
22 #include "Servo.h"
23 #include "SPI.h"
24 //#include "Timer2.h"
25 #include "TMP75.h"
26 #include "TWI_ATMEGA.h"
27 #include "Keys.h"
28 #include "Timer1.h"
29
30 //Defines
31 #define CAN_NOT_RECEIVED 0
32 #define CAN_RECEIVED 1
34 //Definition von ws2812_1
35 WS2812_pin WS2812_1 =   {/*DDR register*/   &DDRD,
36                          /*PORT register*/  &PORTD,
37                          /*Pin*/            PD2};
39 //
40 MCP2515_pins  MCP2515_1 = {{/*CS_DDR*/      &DDRD,
41                             /*CS_PORT*/     &PORTD,
42                             /*CS pin*/      PD0,
43                             /*CS_state*/    ON}};
47 //can_frame sSendFrame, sRecFrame;
48 //can_filter sFilter;
49 //
51 //MCP2515_pins  MCP2515_1 = {{/*CS_DDR*/    &DDRB,
55 //
56 //uint32_t ulReceiveFilter[6] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66};
57 //uint32_t ulReceiveMask[2] = {0x7FF, 0x7FF};
58
59 //unsigned char ucTimer = 0;
60
61
62 #endif /* MAIN_H_ */
```

MCP2515_HHN.h

```
1 /*
2  * MCP2515_HHN.h
3  *
4  * Created: 07.03.2017 10:26:27
5  *  Author: Petre Sora
6
7       * Disclaimer: This code sample is just a prototype. It is in no way suitable
8       * for safe and reliable code, since the authors have intentionally abstained
9       * from error or plausibility checks.
10       * Therefore, the code examples must not be used in military or commercial or
11       * safety-related products. Running this software can potentially be
dangerous.
12       *
13       * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
14       * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
15       * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
16       * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
17       * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
18       * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
19       * OR SERVICES; LOSS OF USE, DATA, OR PROFITS, OR BUSINESS INTERRUPTION)
20       * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
21       * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
22       * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
23       * SUCH DAMAGE.
24       *
25       * Redistribution and use in source and binary forms, with or without
26       * modification, are permitted under license CC-BY-NC-SA provided that the
27       * following conditions are met:
28       * 1. Redistributions of source code must retain the above copyright
29       *    notice, this list of conditions and the above disclaimer.
30       * 2. Redistributions in binary form must reproduce the above copyright
31       *    notice, this list of conditions and the above disclaimer in the
32       *    documentation and/or other materials provided with the distribution.
33  */
34
35
36 #ifndef MCP2515_HHN_H_
37 #define MCP2515_HHN_H_
38
39 #include <avr/io.h>
40 #include "SPI.h"
41
42
43 typedef struct
44 {
45     tspiHandle MCP2515spi;
46 } MCP2515_pins;
47
48 typedef struct
49 {
50     uint8_t EIDE_Bit; //STANDARD_ID oder EXTENDED_ID
51     uint8_t RTR_Bit; //DATA_FRAME oder REMOTE_FRAME
52     uint32_t ulID; //11 oder 29 Bit ID
53     uint8_t ucLength; //Anzahl der Datenbytes
54     uint8_t ucData[8]; //Datenvektor für die maximale Nachrichtenlänge
55 }can_frame;
56
57 typedef struct
58 {
59     uint8_t RecBuff_ID[2]; //STANDARD_ID oder EXTENDED_ID
60     uint8_t Rec_Buff0_Rollover; //ROLLOVER_ON / ROLLOVER_OFF
61     uint8 t Filter RecBuff[2]; //FILTER ON oder FILTER OFF
62     uint32_t ulRecBuff_Filter[6]; //die ersten 2 Filter sind für den Buffer1, die
weiteren 4 für den Zweiten
63     uint32_t ulRecBuff_Mask[2]; //Filtermasken der Empfangspuffer
64 }can_filter;
65
66
//*****************************************************************************
****
67 //Funktionen
```

```
68
//*****************************************************************************
****
69 void MCP2515_Init(MCP2515_pins sdevice_pins, uint8_t ucbaud);
70 void MCP2515_Read_Reg(MCP2515_pins sdevice_pins, uint8_t ucreg_address, uint8_t
ucreg_number, uint8_t *ucreg_in);
71 void MCP2515_Write_Reg(MCP2515_pins sdevice_pins, uint8_t ucreg_address, uint8_t
ucreg_number, uint8_t *ucreg_out);
72 void MCP2515_Change_Reg(MCP2515_pins sdevice_pins, uint8_t ucreg_address, uint8_t
ucreg_mask, uint8_t ucreg_data);
73 void MCP2515_Set_OpMode(MCP2515_pins sdevice_pins, uint8_t ucop_mode);
74 void MCP2515_Change_ClkOut(MCP2515_pins sdevice_pins, uint8_t ucclk_out);
75 void MCP2515_OneShotMode(MCP2515_pins sdevice_pins, uint8_t ucone_shot);
76 void MCP2515_Set_Baudrate(MCP2515_pins sdevice_pins, uint8_t ucbaud);
77 uint8_t MCP2515_Read_Status(MCP2515_pins sdevice_pins);
78 void MCP2515_Load_TXBuffer(MCP2515_pins sdevice_pins, uint8_t ucbuffer_start,
uint8 t ucbuffer length, uint8 t *ucreg out);
79 uint8_t MCP2515_Send_Message(MCP2515_pins sdevice_pins, can_frame *sframe);
80 void MCP2515_RequestToSend(MCP2515_pins sdevice_pins, uint8_t uctx_buffer);
81 void MCP2515_Set_Filter_Mask(MCP2515_pins sdevice_pins, can_filter *sfilter);
82 void MCP2515_Build_Filter_Frame(uint32_t ulfilter_id, uint8_t ucdata_frame);
83 uint8_t MCP2515_Read_RxStatus(MCP2515_pins sdevice_pins);
84 uint8 t MCP2515 Check Message(MCP2515 pins sdevice pins, can frame *sframe);
85 void MCP2515_Read_RxBuffer(MCP2515_pins sdevice_pins, uint8_t ucbuffer_start,
uint8_t ucbuffer_length, uint8_t *ucreg_in);
86
87
//*****************************************************************************
****
88 //define's
89
//*****************************************************************************
****
90 #define ON                  1
91 #define OFF                 0
92
93 //OP-codes
94 #define RESET_CODE          0xC0
95 #define READ_REG_CODE       0x03
96 #define READ_RX_BUFFER_CODE 0x90
97 #define WRITE_REG_CODE      0x02
98 #define LOAD_TX_BUFFER_CODE 0x40
99 #define RTS_CODE            0x80
100 #define READ_STATUS_CODE    0xA0
101 #define READ_RX_STATUS_CODE 0xB0
102 #define BIT_MODIFY_CODE     0x05
103
104 //Operation mode
105 #define OP_MODE_MASK        0xE0
106 #define NORMAL_OP_MODE      0x00
107 #define SLEEP_MODE          0x20
108 #define LOOPBACK_MODE       0x40
109 #define LISTEN_ONLY_MODE    0x60
110 #define CONFIG_MODE         0x80
111
112 //Clock-out
113 #define CLOCK_OUT_MASK      0x07
114 #define CLOCK_OUT_DISABLE   0x00
115 #define ENABLE_FOUT_1TO1    0x04
116 #define ENABLE_FOUT_1TO2    0x05
117 #define ENABLE_FOUT_1TO4    0x06
118 #define ENABLE_FOUT_1TO8    0x07
119
120 //One-Shot Mode
121 #define ONE_SHOT_MASK       0x08
122 #define ONE_SHOT_ENABLE     0x08
123 #define ONE_SHOT_DISABLE    0x00
124
125 //Control Register Address
126 #define BFPCTRL         0x0C
127 #define TXRTSCTRL       0x0D
128 #define CANSTAT         0x0E
129 #define CANCTRL         0x0F
```

```
130 #define TEC             0x1C
131 #define REC             0x1D
132 #define CNF3            0x28 //nur im Config mode einstellbar
133 #define CNF2            0x29 //nur im Config mode einstellbar
134 #define CNF1            0x2A //nur im Config mode einstellbar
135 #define CANINTE         0x2B
136 #define CANINF          0x2C
137 #define EFLG            0x2D
138 #define TXB0CTRL        0x30
139 #define TXB1CTRL        0x40
140 #define TXB2CTRL        0x50
141 #define RXB0CTRL        0x60
142 #define RXB1CTRL        0x70
143
144 //Transmit Register Address
145 //Sendepuffer 1
146 #define TXBN0CTRL       0x30
147 #define TXB0SIDH        0x31
148 #define TXB0SIDL        0x32
149 #define TXB0EID8        0x33
150 #define TXB0EID0        0x34
151 #define TXB0DLC         0x35
152 #define TXB0D0          0x36 //das erste der 8 Datenregister
153 //Sendepuffer 2
154 #define TXBN1CTRL       0x40
155 #define TXB1SIDH        0x41
156 #define TXB1SIDL        0x42
157 #define TXB1EID8        0x43
158 #define TXB1EID0        0x44
159 #define TXB1DLC         0x45
160 #define TXB1D0          0x46 //das erste der 8 Datenregister
161 //Sendepuffer 3
162 #define TXBN2CTRL       0x50
163 #define TXB2SIDH        0x51
164 #define TXB2SIDL        0x52
165 #define TXB2EID8        0x53
166 #define TXB2EID0        0x54
167 #define TXB2DLC         0x55
168 #define TXB2D0          0x56 //das erste der 8 Datenregister
169
170 //Receive Register
171 //Empfangspuffer 1
172 #define RXB0SIDH        0x61
173 #define RXB0SIDL        0x62
174 #define RXB0EID8        0x63
175 #define RXB0EID0        0x64
176 #define RXB0DLC         0x65
177 #define RXB0D0          0x66 //das erste der 8 Datenregister
178 //Empfangspuffer 2
179 #define RXB1SIDH        0x71
180 #define RXB1SIDL        0x72
181 #define RXB1EID8        0x73
182 #define RXB1EID0        0x74
183 #define RXB1DLC         0x75
184 #define RXB1D0          0x76 //das letzte der 8 Datenregister
185
186 //Receive Filter Register
187 #define RXF0SIDH        0x00 //nur im Config mode einstellbar
188 #define RXF0SIDL        0x01 //nur im Config mode einstellbar
189 #define RXF0EID8        0x02 //nur im Config mode einstellbar
190 #define RXF0EID0        0x03 //nur im Config mode einstellbar
191 #define RXF1SIDH        0x04 //nur im Config mode einstellbar
192 #define RXF1SIDL        0x05 //nur im Config mode einstellbar
193 #define RXF1EID8        0x06 //nur im Config mode einstellbar
194 #define RXF1EID0        0x07 //nur im Config mode einstellbar
195 #define RXF2SIDH        0x08 //nur im Config mode einstellbar
196 #define RXF2SIDL        0x09 //nur im Config mode einstellbar
197 #define RXF2EID8        0x0A //nur im Config mode einstellbar
198 #define RXF2EID0        0x0B //nur im Config mode einstellbar
199 #define RXF3SIDH        0x10 //nur im Config mode einstellbar
200 #define RXF3SIDL        0x11 //nur im Config mode einstellbar
201 #define RXF3EID8        0x12 //nur im Config mode einstellbar
202 #define RXF3EID0        0x13 //nur im Config mode einstellbar
```

```
203 #define RXF4SIDH        0x14 //nur im Config mode einstellbar
204 #define RXF4SIDL        0x15 //nur im Config mode einstellbar
205 #define RXF4EID8        0x16 //nur im Config mode einstellbar
206 #define RXF4EID0        0x17 //nur im Config mode einstellbar
207 #define RXF5SIDH        0x18 //nur im Config mode einstellbar
208 #define RXF5SIDL        0x19 //nur im Config mode einstellbar
209 #define RXF5EID8        0x1A //nur im Config mode einstellbar
210 #define RXF5EID0        0x1B //nur im Config mode einstellbar
211
212 //Receive Filter Mask
213 #define RXM0SIDH        0x20
214 #define RXM0SIDL        0x21
215 #define RXM0EID8        0x22
216 #define RXM0EID0        0x23
217 #define RXM1SIDH        0x24
218 #define RXM1SIDL        0x25
219 #define RXM1EID8        0x26
220 #define RXM1EID0        0x27
221
222 //Interrupt Register Address
223 #define CANINTE         0x2B
224 #define CANINTF         0x2C
225
226 //RX-TX Pin Control and Status Register
227 #define BFPCTRL         0x0C
228 #define TXRTSCTRL       0x0D
229
230 //CAN Baudrate
231 #define BAUDRATE_10_KBPS    0
232 #define BAUDRATE_20_KBPS    1
233 #define BAUDRATE_50_KBPS    2
234 #define BAUDRATE_100_KBPS   3
235 #define BAUDRATE_125_KBPS   4
236 #define BAUDRATE_250_KBPS   5
237 #define BAUDRATE_500_KBPS   6
238 #define BAUDRATE_1_MBPS     7
239
240 //Flags
241 #define STANDARD_ID         0x00
242 #define EXTENDED_ID         0x08
243 #define DATA_FRAME          0x00
244 #define REMOTE_FRAME        0x40
245 #define ROLLOVER_ON         0x04
246 #define ROLLOVER_OFF        0x00
247 #define FILTER_ON           0x01
248 #define FILTER_OFF          0x00
249
250 #define TXREQ0              0x04
251 #define TXREQ1              0x10
252 #define TXREQ2              0x40
253
254 //Errors
255 #define NO_ERROR            0x00
256 #define TX_BUFFER_FULL      0x01
257
258 #define NO_MESSAGE          0x00
259 #define MESSAGE_RECEIVED    0x01
260
261 #endif /* MCP2515_HHN_H_ */
```

Servo.c File Reference

C file to controll servomotor.

```
#include "Servo.h"
```

Include dependency graph for Servo.c:

Functions

- void **Servo_Init** (void)
  Initialize servo by using Fast-PWM mode from Timer0.

- void **Servo_Step** (unsigned char ucPosition)
  Set servo position.

Variables

- unsigned char **ucServoPosition** [6] = {44, 38, 30, 23, 16, 10}

Detailed Description

C file to controll servomotor.

Date
: 24.10.2022 11:17:37

Author
: Hoehnel and Ritter

Function Documentation

void Servo_Init (void)

Initialize servo by using Fast-PWM mode from Timer0.

Parameters

in	*None*	

Returns

None

Date

31.10.2022

Author

Hoehnel and Ritter

Version

1.0

Here is the caller graph for this function:

void Servo_Step (unsigned char ucPosition*)*

Set servo position.

Parameters

in	*unsigned_char*	ucPosition: Heater position from 0 to 5

Returns

None

Date

31.10.2022

Author

Hoehnel and Ritter

Version

1.0

Variable Documentation

unsigned char ucServoPosition[6] = {44, 38, 30, 23, 16, 10}

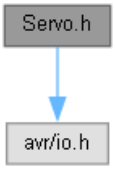
Defines six positions for servomotor

Servo.h File Reference

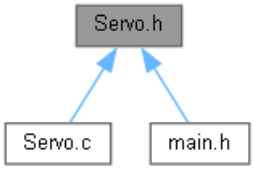
Include file to controll servomotor.

```
#include <avr/io.h>
```

Include dependency graph for Servo.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **Servo_Init** (void)
  Initialize servo by using Fast-PWM mode from Timer0.

- void **Servo_Step** (unsigned char ucPosition)
  Set servo position.

Detailed Description

Include file to controll servomotor.

Date

24.10.2022 11:17:37

Author

Hoehnel and Ritter

Function Documentation

void Servo_Init (void)

Initialize servo by using Fast-PWM mode from Timer0.

Parameters

in	*None*	

Returns

None

Date

31.10.2022

Author

Hoehnel and Ritter

Version

1.0

Here is the caller graph for this function:

void Servo_Step (unsigned char ucPosition*)*

Set servo position.

Parameters

in	*unsigned_char*	ucPosition: Heater position from 0 to 5

Returns

None

Date

31.10.2022

Author

Hoehnel and Ritter

Version

1.0

Servo.h

Go to the documentation of this file.

```
9 #ifndef SERVO_H_
10 #define SERVO_H_
11
12 //Includes
13 #include <avr/io.h>
14
15 //Defines
16
17
18 //Declaration of funktions
19 void Servo_Init(void);
20 void Servo_Step(unsigned char ucPosition);
21
22
23 #endif /* SERVO_H_ */
```

SPI.h

```
1 /*
2  * SPI.h
3  *
4  * Created: 11.01.2015 12:29:37
5  *  Author: Petre Sora
6  */
7
8
9 #ifndef SPI_H_
10 #define SPI_H_
11
12 #include <avr/io.h>
13
14
//*****************************************************************************
****
15 //define's
16
//*****************************************************************************
****
17 typedef struct{
18
19     volatile uint8_t* CS_DDR;
20     volatile uint8_t* CS_PORT;
21     uint8_t CS_pin;
22     uint8_t CS_state;
23 } tspiHandle;
24
25 //SPI Register
26 #define SPI_CONTROL_REGISTER        SPCR
27 #define SPI_DATA_REGISTER           SPDR
28 #define SPI_STATUS_REGISTER         SPSR
29 //
30 #define SPI_ENABLE                  0x40
31 #define SPI_MASTER                  0x10
32 //SPIE-Bit in das SPCR-Register (SPI-Interupt Enable Bit)
33 #define SPI_INTERRUPT_ENABLE        0x80
34 #define SPI_INTERRUPT_DISABLE       0x00
35 //DORD-Bit in das SPCR-Register (data order LSB or MSB first)
36 #define SPI_LSB_FIRST               0x20
37 #define SPI_MSB_FIRST               0x00
38 //SPI-Modus wird über 2 Bits bestimmt CPOL und CPHA im SPCR-Register
39 #define SPI_MODE_0                  0x00
40 #define SPI_MODE_1                  0x01
41 #define SPI_MODE_2                  0x02
42 #define SPI_MODE_3                  0x03
43 //Taktfrequenz der SPI-Schnittstelle (FOSC = Taktfrequenz des Mikrocontrollers)
44 #define SPI_FOSC_DIV_2              0x04
45 #define SPI_FOSC_DIV_4              0x00
46 #define SPI_FOSC_DIV_8              0x05
47 #define SPI_FOSC_DIV_16             0x01
48 #define SPI_FOSC_DIV_32             0x06
49 #define SPI_FOSC_DIV_64             0x02
50 #define SPI_FOSC_DIV_128            0x03
51
52 #define SPI_RUNNING                 (!(SPSR & (1 << SPIF)))
53
54
//*****************************************************************************
****
55 //Funktionen
56
//*****************************************************************************
****
57 void SPI_Master_Init(uint8_t ucspi_interrupt, uint8_t ucspi_data_order, uint8_t
ucspi_mode, uint8_t ucspi_sck_freq);
58 void SPI Master SlaveSelectInit(tspiHandle tspi pins);
59 void SPI_Master_Start(tspiHandle tspi_pins);
60 void SPI_Master_Stop(tspiHandle tspi_pins);
61 unsigned char SPI_Master_Write(uint8_t ucdata);
62
```

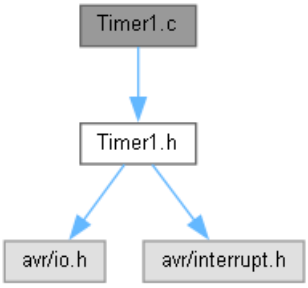
```
63
//*****************************************************************************
****
64 //Definitionen für den ATMega88 Mikrocontroller
65 //dieser Teil muss neu definiert werden für einen anderen Mikrocontroller
66 //das SlaveSelect- Signal ist von der konkreten Anwendung abhängig, deshalb wird es
in
67 //der main-Datei behandelt
68
//*****************************************************************************
****
69 //MOSI-Signal
70 #define SPI_MOSI_DDR_REG            DDRB
71 #define SPI_MOSI_PORT_REG           PORTB
72 #define SPI_MOSI_BIT                PB3
73 //MISO-Signal
74 #define SPI_MISO_DDR_REG            DDRB
75 #define SPI_MISO_PORT_REG           PORTB
76 #define SPI_MISO_BIT                PB4
77 //Clock-Signal
78 #define SPI_CLK_DDR_REG             DDRB
79 #define SPI_CLK_PORT_REG            PORTB
80 #define SPI_CLK_BIT                 PB5
81
82 #endif /* SPI_H_ */
```

Timer1.c File Reference

C file to initiate Timer 1.

```
#include "Timer1.h"
```

Include dependency graph for Timer1.c:



Functions

- void **Timer1_Init** ()
  Initialization of timer1 with 10ms.

- **ISR** (TIMER1_COMPA_vect)
  Interrupt service Routine triggered by timer.

- unsigned char **Timer1_get_10msState** ()
  Says if 10ms have been passed.

- unsigned char **Timer1_get_1sState** (void)
  Says if 1000ms have been passed.

- unsigned char **Timer1_get_100msState** (void)
  Says if 100ms have been passed.

Variables

- unsigned char **ucFlag10ms** = 0
- unsigned char **ucCNt_1s** = 0
- unsigned char **ucFlag_1s** = 0
- unsigned char **ucCNt_100ms** = 0
- unsigned char **ucFlag_100ms** = 0

Detailed Description

C file to initiate Timer 1.

Date

27.04.2022 11:59:47

Author

Hoehnel and Ritter

Function Documentation

ISR (TIMER1_COMPA_vect)

Interrupt service Routine triggered by timer.

Parameters

in	*vector*	Reset point of counter

Returns

None

Date

22.04.2022

Author

Hoehnel and Ritter

Version

2.0

unsigned char Timer1_get_100msState (void)

Says if 100ms have been passed.

Parameters

in	*None*	

Returns

[unsigned char] {TIMER_RUNNING, TIMER_TRIGGERED}

Date

27.04.2022

Author

Hoehnel and Ritter

Version

1.0

unsigned char Timer1_get_10msState (void)

Says if 10ms have been passed.

Parameters

in	*None*	

Returns

[unsigned char] {TIMER_RUNNING, TIMER_TRIGGERED}

Date
27.04.2022
Author
Hoehnel and Ritter
Version
1.0

unsigned char Timer1_get_1sState (void)

Says if 1000ms have been passed.

Parameters

in	*None*	

Returns
[unsigned char] {TIMER_RUNNING, TIMER_TRIGGERED}
Date
27.04.2022
Author
Hoehnel and Ritter
Version
1.0

void Timer1_Init (void)

Initialization of timer1 with 10ms.

Parameters

in	*None*	

Returns
None
Date
27.04.2022
Author
Hoehnel and Ritter
Version
1.0

Here is the caller graph for this function:

Variable Documentation

unsigned char ucCNt_100ms = 0

Counts to 100 ms

unsigned char ucCNt_1s = 0

Counts to 1000 ms

unsigned char ucFlag10ms = 0

Flag that becomes 1 every 10 ms

unsigned char ucFlag_100ms = 0

Flag that becomes 1 every 100 ms

unsigned char ucFlag_1s = 0

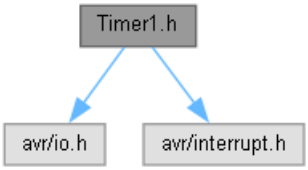
Flag that becomes 1 every 1000 ms

Timer1.h File Reference

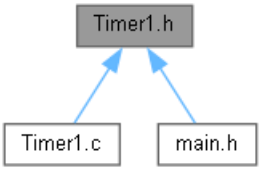
Include file to initiate Timer 1.

```
#include <avr/io.h>
#include <avr/interrupt.h>
```

Include dependency graph for Timer1.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **TIMER_TRIGGERED** 1
- #define **TIMER_RUNNING** 0

Functions

- void **Timer1_Init** (void)
  Initialization of timer1 with 10ms.
- unsigned char **Timer1_get_10msState** (void)
  Says if 10ms have been passed.
- unsigned char **Timer1_get_1sState** (void)
  Says if 1000ms have been passed.
- unsigned char **Timer1_get_100msState** (void)
  Says if 100ms have been passed.

Detailed Description

Include file to initiate Timer 1.

Date

27.04.2022 11:59:47

Author

Hoehnel and Ritter

Function Documentation

unsigned char Timer1_get_100msState (void)

Says if 100ms have been passed.

Parameters

in	*None*	

Returns

[unsigned char] {TIMER_RUNNING, TIMER_TRIGGERED}

Date

27.04.2022

Author

Hoehnel and Ritter

Version

1.0

unsigned char Timer1_get_10msState (void)

Says if 10ms have been passed.

Parameters

in	*None*	

Returns

[unsigned char] {TIMER_RUNNING, TIMER_TRIGGERED}

Date

27.04.2022

Author

Hoehnel and Ritter

Version

1.0

unsigned char Timer1_get_1sState (void)

Says if 1000ms have been passed.

Parameters

in	*None*	

Returns

[unsigned char] {TIMER_RUNNING, TIMER_TRIGGERED}

Date

27.04.2022

Author

Hoehnel and Ritter

Version

1.0

void Timer1_Init (void)

Initialization of timer1 with 10ms.

Parameters

in	*None*	

Returns

None

Date

27.04.2022

Author

Hoehnel and Ritter

Version

1.0

Here is the caller graph for this function:

Timer1.h

Go to the documentation of this file.

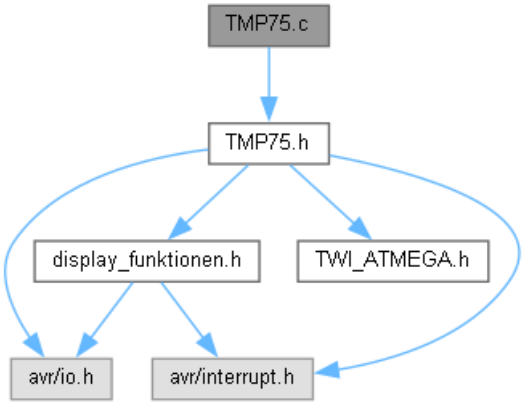
```
9  #ifndef TIMER1_H_
10 #define TIMER1_H_
11
12 //Includes
13 #include <avr/io.h>
14 #include <avr/interrupt.h>
15
16
17 //Defines
18 #define TIMER_TRIGGERED 1
19 #define TIMER_RUNNING 0
20
21 //Declaration of functions
22 void Timer1_Init(void);
23 unsigned char Timer1_get_10msState(void);
24 unsigned char Timer1_get_1sState(void);
25 unsigned char Timer1_get_100msState(void);
26
27 #endif /* TIMER1_H_ */
```

TMP75.c File Reference

C file to measure temperature with TMP75.

```
#include "TMP75.h"
```

Include dependency graph for TMP75.c:



Functions

- uint8_t **TMP75_Read_Temperature** (void)
  Read temperature from TMP75.

- int **TMP75_Get_Temperature** (void)
  Get method to get current temperature value.

Variables

- int **iTemperature**
- uint8_t **ucdevice_address** = 0b1001000
- uint8_t **uctemp2read** = 0b00000000

Detailed Description

C file to measure temperature with TMP75.

Comment

Communication with TMP75 via I^2C

Date

26.10.2022 20:07:03

Author

Hoehnel and Ritter

Function Documentation

int TMP75_Get_Temperature (void)

Get method to get current temperature value.

Parameters

in	*None*	

Returns

[int] Last read temperature, with one decimal number

Date

04.11.2022

Author

Hoehnel and Ritter

Version

1.0

uint8_t TMP75_Read_Temperature (void)

Read temperature from TMP75.

Parameters

in	*None*	

Returns

[uint8_t] Value of Temperature measurement

Date

04.11.2022

Author

Meroth, Sora

Version

1.0

Variable Documentation

uint8_t ucdevice_address = 0b1001000

Device address: all address pins to gnd

uint8_t uctemp2read = 0b00000000

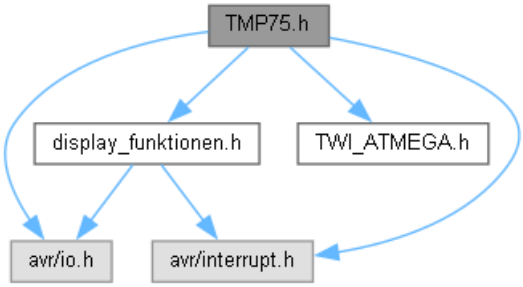
Select register to read: Read temperature register

TMP75.h File Reference

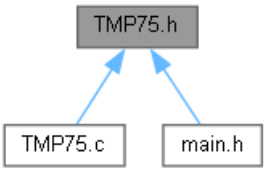
Include file to measure temperature with TMP75.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "TWI_ATMEGA.h"
#include "display_funktionen.h"
```

Include dependency graph for TMP75.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **TMP75_DEVICE_TYPE_ADDRESS** 0x00

Functions

- uint8_t **TMP75_Read_Temperature** (void)
  Read temperature from TMP75.

- int **TMP75_Get_Temperature** (void)
  Get method to get current temperature value.

- void **Display_Output** (int iTemp2print, unsigned char ucLine, unsigned char ucCAN)

Detailed Description

Include file to measure temperature with TMP75.

Comment

Communication with TMP75 via I^2C

Date
26.10.2022 20:07:03
Author
Hoehnel and Ritter

Function Documentation

int TMP75_Get_Temperature (void)

Get method to get current temperature value.

Parameters

in	*None*	

Returns
[int] Last read temperature, with one decimal number
Date
04.11.2022
Author
Hoehnel and Ritter
Version
1.0

uint8_t TMP75_Read_Temperature (void)

Read temperature from TMP75.

Parameters

in	*None*	

Returns
[uint8_t] Value of Temperature measurement
Date
04.11.2022
Author
Meroth, Sora
Version
1.0

TMP75.h

Go to the documentation of this file.[1]

```
11 #ifndef TMP75_H_
12 #define TMP75_H_
13
14 //Includes
15 #include <avr/io.h>
16 #include <avr/interrupt.h>
17 #include "TWI_ATMEGA.h"
18 #include "display_funktionen.h"
19
20 //Defines
21 #define TMP75_DEVICE_TYPE_ADDRESS 0x00
22
23 //Declaration of functions
24 uint8_t TMP75_Read_Temperature(void);
25 int TMP75 Get Temperature(void);
26 void Display_Output(int iTemp2print, unsigned char ucLine, unsigned char ucCAN);
27
28
29 #endif /* TMP75_H_ */
```

TWI.h

```
1 /*
2  * TWI.h
3  *
4  * Created: 26.10.2022 22:31:56
5  *  Author: Moritz
6  */
7
9 //#ifndef TWI_H_
10 //#define TWI_H_
11 //
13 //#include <avr/io.h>
14 //
16 //#define F_CPU 20000000UL //Clock des Board-uC in Hz
17 //#define TWI_SCL_FREQ 400000UL /*gewünschte TWI-Taktfrequenz in Hz; wird
entsprechend der konkreten Anwendung geändert*/
18 //#define TWI_MASTER_CLOCK (((F_CPU / TWI_SCL_FREQ) - 16 ) / 2 +1) //Wert des TWBR-
Registers
19 //#define TWI_STATUS_REGISTER (TWSR &0xF8)
20 //
21 //typedef struct{   //configuration TWI
22     //uint8_t ucDevice; //TWI_MASTER oder TWI_SLAVE
23     //uint8_t ucTwiClock; //Übertragungsrate für den Master
24     //uint8_t ucSlaveAddress; //Slave-Adresse
25     //uint8_t ucGenAddress; //enable/disable general call
26     //uint8_t ucAddressMask; //ev. Adressmaske für den Slave
27 //} TWI_InitParam;
28 //
29 //typedef struct
30 //{
31     //uint8_t ucAddress; //für den Slave: private oder allgemeine Adresse
32     //uint8_t ucTWIData[4];
33     //uint8_t ucTWIDataLength;
34 //}twi_frame;
35 //
37 //void TWI_Init(void);
38 //void TWI_Master_Start(void);
39 //void TWI_Master_Transmit(unsigned char ucdata);
40 //uint8_t TWI_Send_Frame(uint8_t ucdevice_address, twi_frame *sframe);
41 //unsigned char TWI_Master_Read_Ack(void);
42 //unsigned char TWI_Master_Read_NAck(void);
43 //
44 //
45 //#endif /* TWI_H_ */
46 //*/
```

TWI_ATMEGA.h

```
1 /*
2  * TWI_ATMEGA.h
3  *
4  * Created: 05.09.2014 08:59:08
5  *  Author: Petre Sora
6   * Status: freigegeben
7   * Historie: 05.09.2014        V1.0
8  */
9
10
11 #ifndef TWI_ATMEGA_H_
12 #define TWI_ATMEGA_H_
13
14
//*****************************************************************************
****
15 //TWI define's
16
//*****************************************************************************
****
17
18 #define TWI_STATUS_REGISTER (TWSR & 0xF8)
19
20 #define TWI_RUNNING         0x00
21 #define TWI_TRIGGERED       0x01
22
23 #define TWI_START           0x08
24 #define TWI_RESTART         0x10
25 #define TWI_MR_SLA_ACK      0x40
26 #define TWI_MR_DATA_ACK     0x50
27 #define TWI_MR_DATA_NACK    0x58
28
29 #define TWI_MT_SLA_ACK      0x18
30 #define TWI_MT_DATA_ACK     0x28
31 #define TWI_MT_DATA_NACK    0x30
32
33 #define TWI_ACK             1
34 #define TWI_NACK            0
35
36 #define TWI_ERROR           0x01
37 #define TWI_OK              0x00
38
39 #define TWI_WRITE           0x00
40 #define TWI_READ            0x01
41
42
43
//*****************************************************************************
****
44 //Deklaration der Funktionen
45
//*****************************************************************************
****
46 void TWI_Master_Init(unsigned char uctwi_clock);
47 void TWI_Master_Start(void);
48 void TWI_Master_Stop(void);
49 void TWI_Master_Transmit(unsigned char ucdata);
50 unsigned char TWI_Master_Read_Ack(void);
51 unsigned char TWI_Master_Read_NAck(void);
52
53 void TWI_INT_Master_Transmit(unsigned char ucdata);
54
55 unsigned char TWI_Get_State(void);
56 unsigned char TWI_Get_TWSRRegister(void);
57 void TWI_INT_Enable(void);
58 void TWI_INT_Disable(void);
59 void TWI Set ErrorFlag(void);
60
61 #endif /* TWI_ATMEGA_H_ */
```

WS2812.h

```
1 /*
2  * WS2812.h
3  *
4  * Created: 30.04.2018 11:46:02
5  *  Author: Petre Sora
6  * Disclaimer: This code sample is just a prototype. It is in no way suitable
7  * for safe and reliable code, since the authors have intentionally abstained
8  * from error or plausibility checks.
9  * Therefore, the code examples must not be used in military or commercial or
10  * safety-related products. Running this software can potentially be dangerous.
11  *
12  * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
13  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
14  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
15  * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
16  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
17  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
18  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS, OR BUSINESS INTERRUPTION)
19  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
20  * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
21  * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
22  * SUCH DAMAGE.
23  *
24  * Redistribution and use in source and binary forms, with or without
25  * modification, are permitted under license CC-BY-NC-SA provided that the
26  * following conditions are met:
27  * 1. Redistributions of source code must retain the above copyright
28  *    notice, this list of conditions and the above disclaimer.
29  * 2. Redistributions in binary form must reproduce the above copyright
30  *    notice, this list of conditions and the above disclaimer in the
31  *    documentation and/or other materials provided with the distribution.
32  */
33
34
35 #ifndef WS2812_H_
36 #define WS2812_H_
37
38 #include <avr/io.h>
39
40
41
//*****************************************************************************
****
42 //in der main wird eine Variable vom Typ WS2812_pin definiert mit dem ansteuernden
Pin nach
43 //dem Beispiel: WS2812_pin WS2812_1 =   {/*DDR register*/   &DDRD,
44 //                                      /*PORT register*/   &PORTD,
45 //                                      /*Pin*/             PD2};
46
//*****************************************************************************
****
47 typedef struct
48 {
49     volatile uint8_t *WS2812_DDRReg;
50     volatile uint8_t *WS2812_PORTReg;
51     uint8_t WS2812_Pin;
52 }WS2812_pin;
53
54
//*****************************************************************************
****
55 //Funktionen
56
//*****************************************************************************
****
57 void WS2812_Init(void);
58 void WS2812 set off(void);
59 void WS2812_Set_Colour(uint8_t *uccolour_list, uint16_t ucled_number);
60 void WS2812_Step(unsigned char ucStep);
61
62 #endif /* WS2812_H_ */
```

Index

INDEX

Quellenverzeichnis

Meroth, Ansgar und Sora, Petre: Sensornetzwerke in Theorie und Praxis, Heilbronn und Wiesbaden, 2021 (2. Auflage)

Unbekannt: ATmega48PA/88PA/168PA [DATASHEET], https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-9223-Automotive-Microcontrollers-ATmega48PA-ATmega88PA-ATmega168PA_Datasheet.pdf (Stand: 05.01.2023)

Unbekannt: Servo-Motor-Kit User Manual, https://eu.mouser.com/datasheet/2/598/ervo_Motor_Kit-1020966.pdf (Stand: 05.01.2023)

Unbekannt: TMP75B Datasheet, https://www.ti.com/lit/ds/symlink/tmp75b.pdf?ts=1672919165643&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTMP75B (Stand: 05.01.2023)

Tessie: WS2812B Addressable RGB LED: Datasheet, Pinout and Applications, https://www.utmel.com/components/ws2812b-addressable-rgb-led-datasheet-pinout-and-applications?id=534 (Stand: 05.01.2023)

Abbildungsverzeichnis