Inhaltsverzeichnis

EAGLE und die microSPS Dateien

Für die Programmerstellung wird das Schaltplan Modul von Eagle verwendet. Die Fa. CadSoft stellt eine Freeware Version zur Verfügung, die kostenlos unter folgender Seite http://www.cadsoft.de/ erhältlich ist.

Nachdem Eagle installiert ist, müssen für die SPS die Bibliothek in folgendem Ordner abgelegt werden:

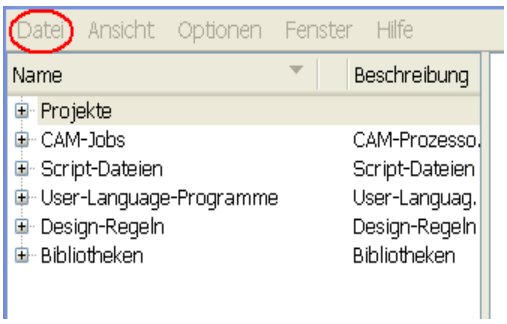
microSPS_V1_00.lbr => eagle\lbr\microSPS_V1_00.lbr

Danach den Ordner microSPS unter „eagle\Projekte\“ ablegen.

Falls sie mit der Bedienung von Eagle noch nicht vertraut sind, empfehle ich als Einstieg den Schnellkurs http://www.cadsoft.de/microsps/tutorial/index.htm von CadSoft durch zu arbeiten.

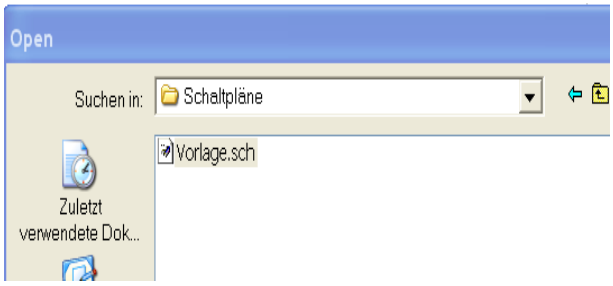
Das erste microSPS Programm

An eine Beispiel wird die Programmerstellung erklärt. Zuerst wird Eagle gestartet.



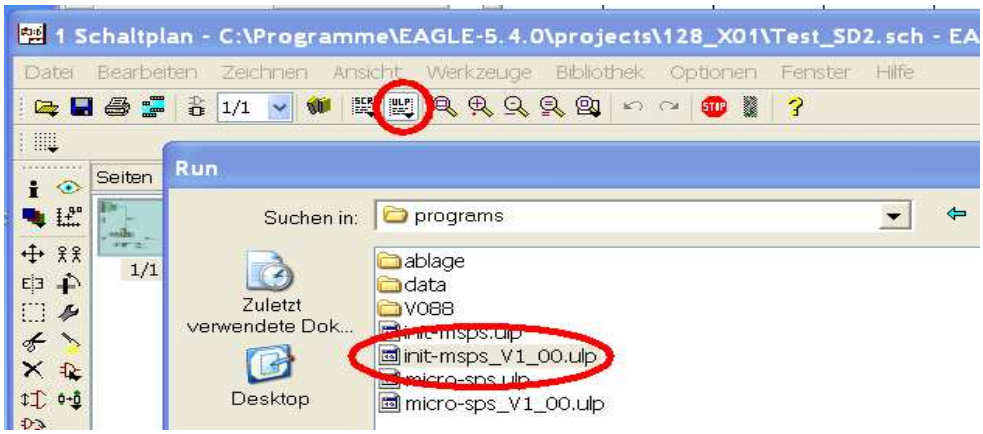
Über „Datei“ „Öffnen“ „Schaltpläne“ eine leere Vorlage auswählen. Die Schaltpläne befinden

sich im Verzeichnis: „C:\Programme\EAGLE\projects\microSPS\Schaltpläne“.

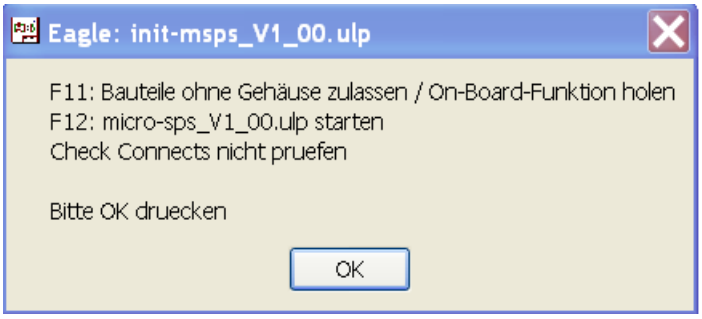


Folgendes Fenster wird nun angezeigt.

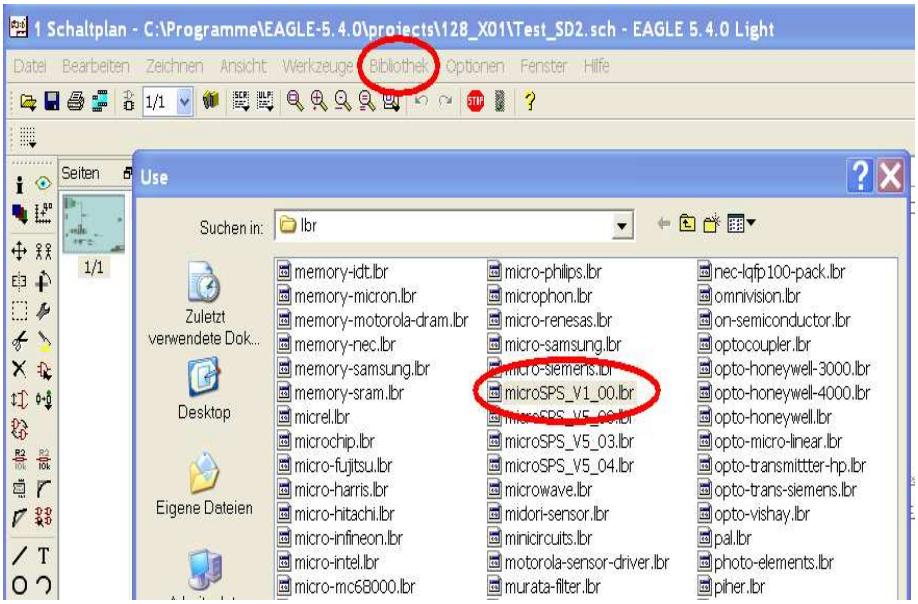
Nun muss die microSPS ULP installiert werden. Die Schaltfläche ULP betätigen und das Programm „init-msps_V1_00.upl“ auswählen und ausführen. Die Datei befindet sich unter folgendem Verzeichnis:
C:\Programme\EAGLE\projects\microSPS\programs\



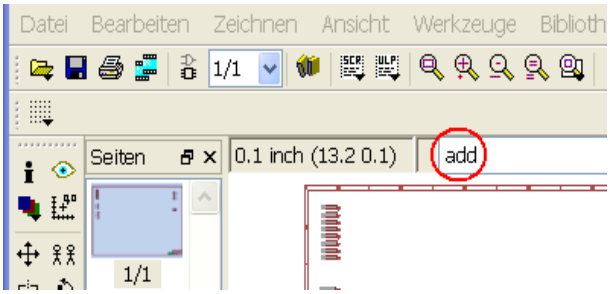
Eagle meldet sich mit folgendem Fenster, welches mit „OK“ bestätigt wird.

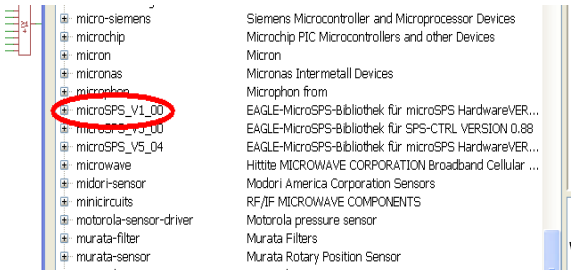


Nun die Bibliothek „microSPS_V1_00.lbr“ über „Bibliothek“ „Benutzen“ für die microSPS auswählen. Sie befindet sich unter folgendem Verzeichnis: „C:\Programme\EAGLE\lbr\“



Jetzt ist Eagle für das Erstellen des Schaltplans vorbereitet. Die einzelnen Funktionselemente werden nun aus der Biliothek „microSPS_V1_00.lbr“ entnommen und im Schaltplan platziert. Die Bibliothek kann über den Add-Befehl geöffnet werden.

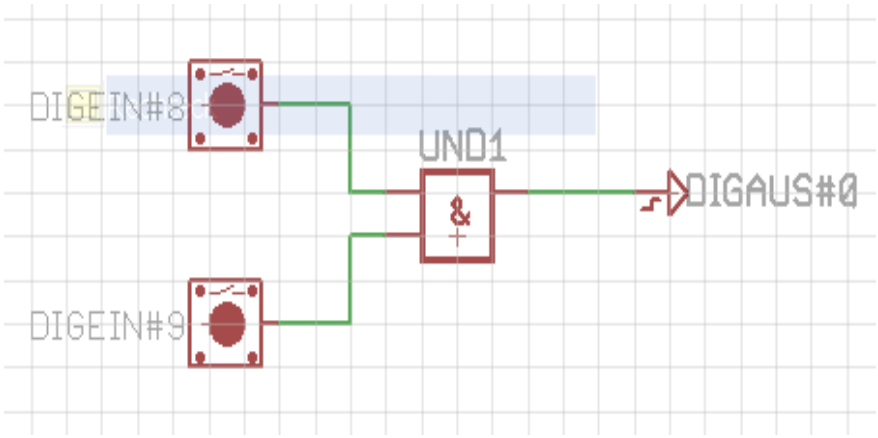




Aus der Bibliothek wird nun das gewünschte Bauteil ausgewählt. Ich nehme in dem Beispiel mal ein UND Gatter und platziere es im Schaltplan.

Nach dem Platzieren von einem oder mehreren Bauteilen kommt man mit der Taste ESC wird zurück zur Bibliothek. Nun kann das nächste Bauteil ausgewählt werden oder der Vorgang wird mit dem Button „Abbrechen“ beendet.

Die für die Schaltung erforderlichen Bauteile werden nun auf dem Schaltplan an die gewünschte Position gebracht und mit dem NET-Befehl „Net“ (grünen Linien) verbunden.



Die erste Schaltung ist nun fertig und sollte nun mit „Datei“ „Speicher unter“ abgelegt werden. Mit F12 wird das SPS-Programm erzeugt.

Das Programm wird nun mit der Benutzeroberfläche in der SPS abgespeichert. Die SPS-Datei befindet sich in folgendem Verzeichnis:
„C:\Programme\EAGLE\projects\microSPS\programs\data\“

Das Programm befindet sich nun in der microSPS und kann getestet werden. Wird an der microSPS Taste1 und Taste2 betätigt, schaltet Relais 1 ein.

Allgemeine Informationen

Der maximale Fehler einer Schaltzeit ist eine Auflösungseinheit. Daher sollte bei einer Zeit von einer Sekunde nicht die Zeiteinheit SEK ausgewählt werden. Der tatsächliche Wert liegt dann zwischen 0 und 1 Sekunde. Besser ist als Zeiteinheit 10ms zu wählen und als Zeit 100 einzugeben. In diesem Fall wird eine Genauigkeit von 990ms bis 1000ms erreicht.

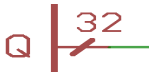
Die einzelnen Funktionsbausteine werden über Signalleitungen miteinander verbunden (grüne Linien im Schaltplan). Die Signalleitungen können 1Bit, 8 Bit, 16Bit oder 32Bit sein. Der Ausgang bestimmt, welche Bitbreite die Signalleitung erhält. Der Eingang des Funktionsbausteins wandelt die Information der Signalleitung auf die richtige Größe um. Die Umwandlung erfolgt Vorzeichenlos (unsigned). Bei der Reduzierung von Bitbreiten werden die nicht benötigten höherwertigen Bits abgeschnitten. Falls der Benutzer für seine Anwendung eine andere Umwandlung benötigt, steht dafür auch ein Funktionsbaustein zur Verfügung.

Ein 1Bit Signaleingang oder Ausgang ist an einem Pin ohne Querstrich zu erkennen. Als Beispiel: der ODER Baustein hat zwei 1Bit Eingänge und ein 1Bit Ausgang.

Ein 8 Bit Ausgang hat einen Querstrich mit der Zahl 8 (8 Bit)

Ein 16 Bit Ausgang hat einen Querstrich mit der Zahl 16 (16 Bit)

Ein 32 Bit Ein / Ausgang hat einen Querstrich mit der Zahl 32 (32 Bit)



Eine 32 Bit Gleitkommazahl hat die Bezeichnung FP

FP (floating point) ist die Abkürzung für Gleitkommazahl

Beispiele:

- Ein 16Bit Ausgang mit einem 32Bit Eingang verbunden: Es wird ein 16Bit Signal übertragen.
- Ein 32Bit Ausgang wird mit einem 16Bit Eingang verbunden: Es wird ein 32Bit Signal übertragen, da der Ausgang die Bit breite bestimmt.
- Ein 32Bit Ausgang wird mit einem 32Bit Eingang verbunden. Die Signalübertragung erfolgt mit 32 Bit.

Über die 32Bit Signalleitungen können auch Gleitkommazahlen übertragen werden. Eine FP Variable benutzt die 32 Bit Signalleitung.

Bei den einzelnen Datenformaten muss der Datentyp beachtet werden. Möglich sind Eingaben als mit Vorzeichen und ohne Vorzeichen. Normale Verknüpfungen werden mit einer Variablen ohne Vorzeichen durchgeführt, wobei bei Berechnungen ein Variablem mit Vorzeichen die bessere Lösung darstellt. Mit welchem Variablentyp der einzelne Funktionsbaustein arbeitet,

kann der Beschreibung der Funktionsbausteine entnommen werden.

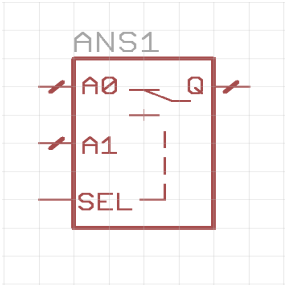
Übersicht der Formate:

uint 8 Bit	MMMM.MMMM (8 Bit Zahl ohne Vorzeichen, Wertebereich 0 … 255)
int 8 Bit	SMMM.MMMM (8 Bit Zahl mit Vorzeichen, Wertebereich -127 … 128)
int 16Bit	SMMM.MMMM.MMMM.MMMM (ein Vorzeichenbit, 15 Bit Mantisse)
int 32Bit	SMMM.MMMM.MMMM.MMMM.MMMM.MMMM.MMMM.MMMM (ein Vorzeichenbit, 31Bit Mantisse)
FP 32Bit	SEEE.EEEE.EMMM.MMMM.MMMM.MMMM.MMMM.MMMM (ein Vorzeichenbit, 8Bit Exponent, 23Bit Mantisse)

Übersicht der Funktionsbausteine

ANALOGSCHALTER

Abhängig vom Wert an SEL (0 oder 1) wird entweder der Eingang A0 oder A1 zum Ausgang durchgeschaltet.

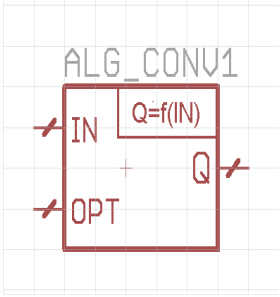


Recourcen:

Flash Speicher:	xx Byte
Ram:	xx Byte

ANALOG_CONVERTER

Dieser Baustein wandelt einen am Eingang IN angelegten Wert nach einem einstellbaren Verfahren um und gibt den Wert am Ausgang Q aus. Die Bedeutung vom Eingang OPT hängt von jeweiligen Verfahren ab. Der VALUE bestimmt die Art der Konvertierung.



Übersicht der möglichen VALUE-Werte:
NTC220k = Analogwert eines NTC220k-Sensors wird in eine Temperatur gewandelt
Als Eingang IN kann direkt ein Analogeingänge verwendet werden.
Der Ausgang OUT hat eine Auflösung von 0,1°C mit 30°C Offset. (z.B. 0 = -30,0°C , 553 = 25,3°C
Bereich: -30,0°C bis +140,0°C

NTC10K: noch nicht implementiert

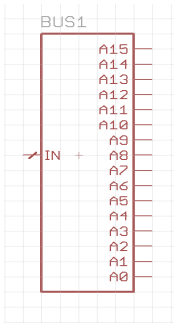
PT100a: mit externer Beschaltung möglich => 0V entsprechen -30°C, 1 Digit entspricht 0,1°C oder 4,88mV.

PT1000a: mit externer Beschaltung möglich => 0V entsprechen -30°C, 1 Digit entspricht 0,1°C oder 4,88mV.

Der OPT-Eingang kann zur zusätzlichen Filterung benutzt werden:

FILTER = Wert wird nach folgender Formel gefiltert:
Filterfunktion: Q = AltQ + (AltQ - IN) / OPT , Zeitinterval: OPT * 10ms
Es muss ein VALUE angegeben werden.

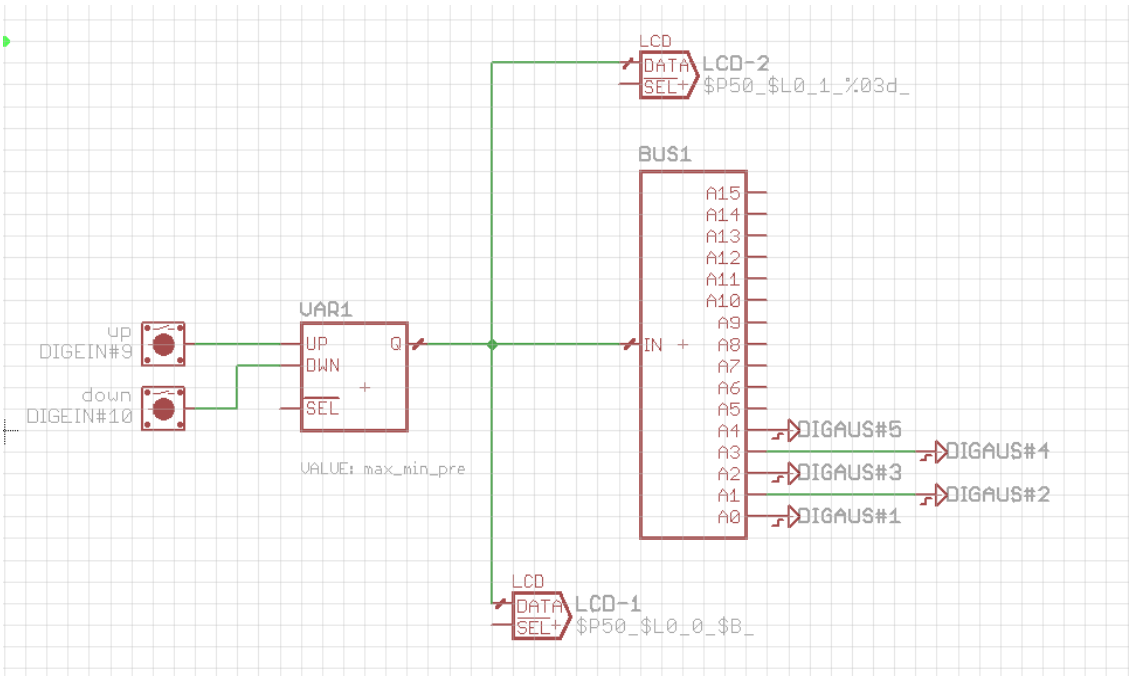
BUSEXPAND



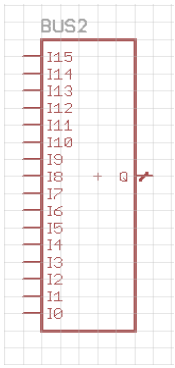
16Bit Wert in Einzelsignale aufsplitten.

Diese Funktion ist die Umkehrung der Funktion BUSREDUCT. Eine 16-Bit-Zahl wird in eine Binärzahl umgewandelt, wobei der Ausgang A0 die niedrigste Wertigkeit hat.

Beispiel: eine Testschaltung für den Baustein „BUSEXPAND“



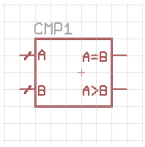
BUSREDUCT



16 Einzelsignale in eine 16Bit Wert umwandeln.

Beginnend mit I0, I1 usw. werden die Eingangswerte mit den Faktoren 1, 2, 4, 8 usw. multipliziert. Die Summe ergibt den Ausgangswert.

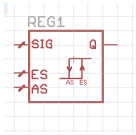
COMPARATOR



Vergleich von zwei 16Bit Werten.

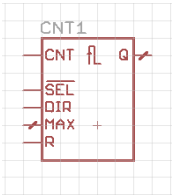
Wenn die Werte an A und B gleich sind, geht der Ausgang A=B auf 1, sonst ist er 0. Über den Ausgang A>B wird eine 1 ausgegeben, wenn A größer als B ist.

COMP_HYST



Komparator mit Hysterese. Q geht auf 1, wenn SIG größer wird als ES. Q wird erst dann wieder 0, wenn SIG kleiner wird als AS. Für die korrekte Funktion wird AS kleiner als ES gewählt.

COUNTER



Aufwärtszähler oder Abwärtszähler mit einstellbarem Überlauf.

MAX = 0: Bei jeder positiven Flanke an CNT wird der Ausgangswert um 1 erhöht (DIR = 0) oder gesenkt (DIR = 1). Ist der Maximalwert von 65535 erreicht, gibt es keinen Überlauf. Ebenso, wenn der Minimalwert 0 erreicht ist.

MAX = Zahl von 1 bis 65535: Der Zähler verhält sich wie oben beschrieben, aber der Ausgang kann nur Werte von 0 bis MAX annehmen. Außerdem gibt es bei 0 bzw. MAX einen Überlauf, je nach Zählrichtung. Beim Aufwärtszählen gibt es einen Überlauf nach 0, beim Abwärtszählen springt der Ausgang nach 0 auf MAX. Damit hat man einen sogenannten Modulozähler, der zum Beispiel beim Durchtasten bestimmter Betriebszustände vorteilhaft eingesetzt werden kann.

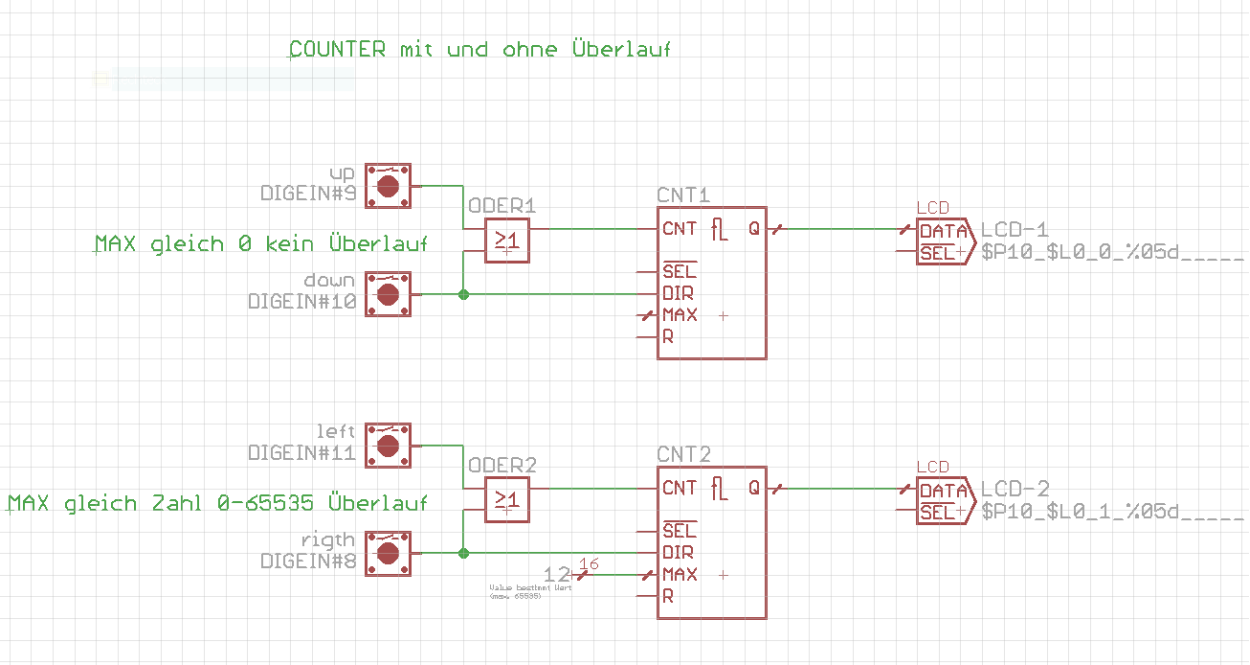
R = 1 setzt den Zählerstand auf 0, wenn DIR = 0.

R = 1 setzt den Zählerstand auf MAX, wenn DIR =1.

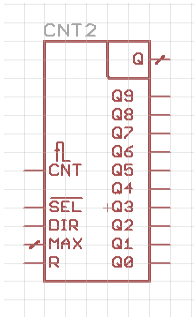
Innerhalb der ersten 300 ms nach dem Einschalten wird ein Pegelwechsel am Triggereingang

nicht ausgewertet.

Beispiel: Testschaltung für den Baustein „COUNTER“



DECATE_CNT



Aufwärtszähler oder Abwärtszähler mit einstellbarem Überlauf und digitalen Ausgängen.

Der Ausgang Q beinhaltet den Zählerwert als 16Bit-Zahl.

Von den Ausgängen Q0 bis Q9 ist jeweils nur derjenige auf High-Pegel, zu dem der Zählerstand passt. Z.B. ist bei Zählerstand 5 der Ausgang Q5 aktiv.

MAX = 0: Bei jeder positiven Flanke an CNT wird der Ausgangswert um 1 erhöht (DIR = 0) oder gesenkt (DIR = 1). Ist der Maximalwert von 65535 erreicht, gibt es keinen Überlauf. Ebenso, wenn der Minimalwert 0 erreicht ist.

MAX = Zahl von 1 bis 65535: Der Zähler verhält sich wie oben beschrieben, aber der Ausgang kann nur Werte von 0 bis MAX annehmen. Außerdem gibt es bei 0 bzw. MAX einen Überlauf, je nach Zählrichtung. Beim Aufwärtszählen gibt es einen Überlauf nach 0, beim Abwärtszählen springt der Ausgang nach 0 auf MAX. Damit hat man einen sogenannten Modulozähler, der zum Beispiel beim Durchtasten bestimmter Betriebszustände vorteilhaft eingesetzt werden kann.

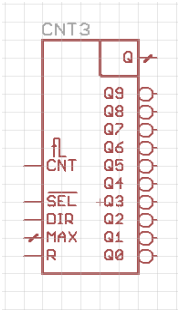
R = 1 setzt den Zählerstand auf 0, wenn DIR = 0.

R = 1 setzt den Zählerstand auf MAX, wenn DIR =1.

Innerhalb der ersten 300 ms nach dem Einschalten wird ein Pegelwechsel am Triggereingang

nicht ausgewertet.

DECATE_CNT_INV



Aufwärtszähler oder Abwärtszähler mit einstellbarem Überlauf und digitalen Ausgängen.

Von den Ausgängen Q0 bis Q9 ist jeweils nur derjenige auf LOW-Pegel, zu dem der Zählerstand passt. Z.B. ist bei Zählerstand 5 der Ausgang Q5 Low, die anderen sind dann High.

MAX = 0: Bei jeder positiven Flanke an CNT wird der Ausgangswert um 1 erhöht (DIR = 0) oder gesenkt (DIR = 1). Ist der Maximalwert von 65535 erreicht, gibt es keinen Überlauf. Ebenso, wenn der Minimalwert 0 erreicht ist.

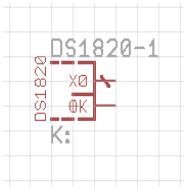
MAX = Zahl von 1 bis 65535: Der Zähler verhält sich wie oben beschrieben, aber der Ausgang kann nur Werte von 0 bis MAX annehmen. Außerdem gibt es bei 0 bzw. MAX einen Überlauf, je nach Zählrichtung. Beim Aufwärtszählen gibt es einen Überlauf nach 0, beim Abwärtszählen springt der Ausgang nach 0 auf MAX. Damit hat man einen sogenannten Modulozähler, der zum Beispiel beim Durchtasten bestimmter Betriebszustände vorteilhaft eingesetzt werden kann.

R = 1 setzt den Zählerstand auf 0, wenn DIR = 0.

R = 1 setzt den Zählerstand auf MAX, wenn DIR =1.

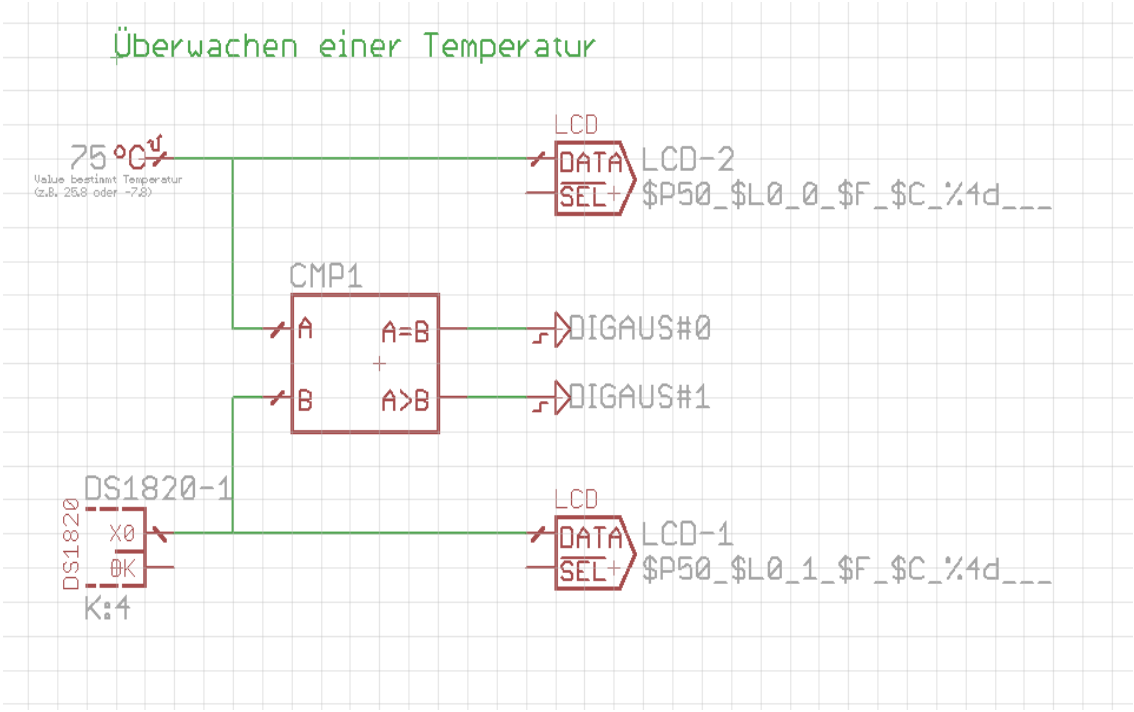
Innerhalb der ersten 300 ms nach dem Einschalten wird ein Pegelwechsel am Triggereingang nicht ausgewertet.

DS1820 *Temperatursensor*

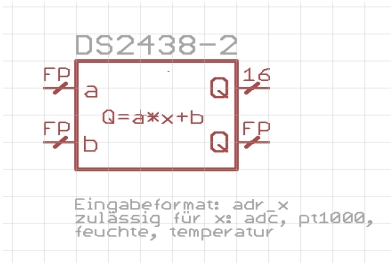


Temperaturwert aus einem Dallas Sensor 1820 auslesen und an X0 ausgeben. Der Temperaturwert wird mal 10 multipliziert. Danach wird eine Konstante von 300 addiert. Damit ergibt sich für 0°C ein Wert von 300. An 0K wird eine 1 ausgegeben, wenn der 1820 fehlerfrei gelesen wurden. Im Fehlerfall wird eine 0 ausgegeben.

Beispiel: Testschaltung für den Baustein „DS1820“



DS2438 *Feuchtemessung und Temperatursensor*



Mit dem DS2438 lassen sich unterschiedliche Messaufgaben lösen. Der Sensor kann zum einen für die Temperaturmessung und für die Feuchtemessung eingesetzt werden. Die Messung der Feuchte wird über einen AD-Wandler mit einem Messbereich von 0 bis 10V und einer Auflösung von 10mV eingelesen. Über diesen Eingang lassen sich auch Spannungen messen. Der Funktionsbausein liest den digitalen Wert aus der AD Wandlung aus, welcher dann über den Funktionsbaustein verrechnet wird.

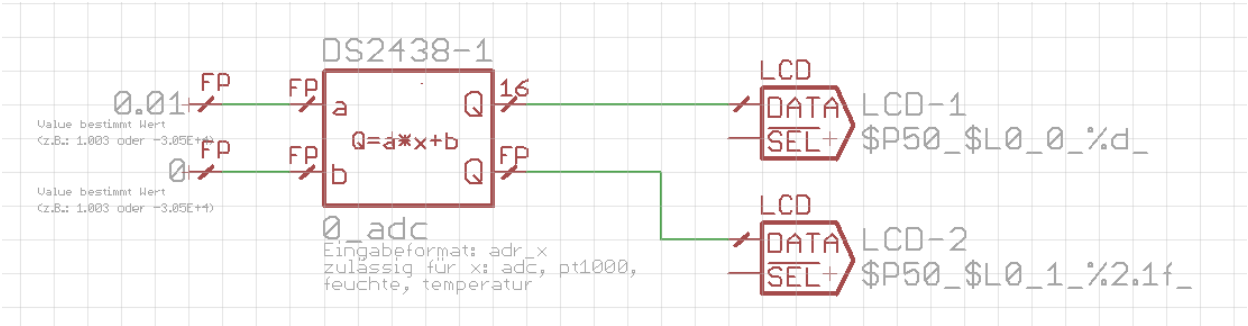
Da über den one wire bus mehrere Bausteine über den Bus adressierbar sind, wird über das VALUE die Bausteinadresse bestimmt. Der erste Baustein der am Bus erkannt wird erhält die Nummer 0. Die Nummern werden aufsteigend von der microSPS vergeben.

Über den zweiten Wert im VALUE wird die Funktion des Bausteins ausgewählt. Folgende Eingaben sind möglich:

adc	Messen der Analogen Eingangsspannung
pt1000	Messen der Temperatur über einen pt1000 Messfühler
feuchte	Messen und Ausgeben der Feuchte

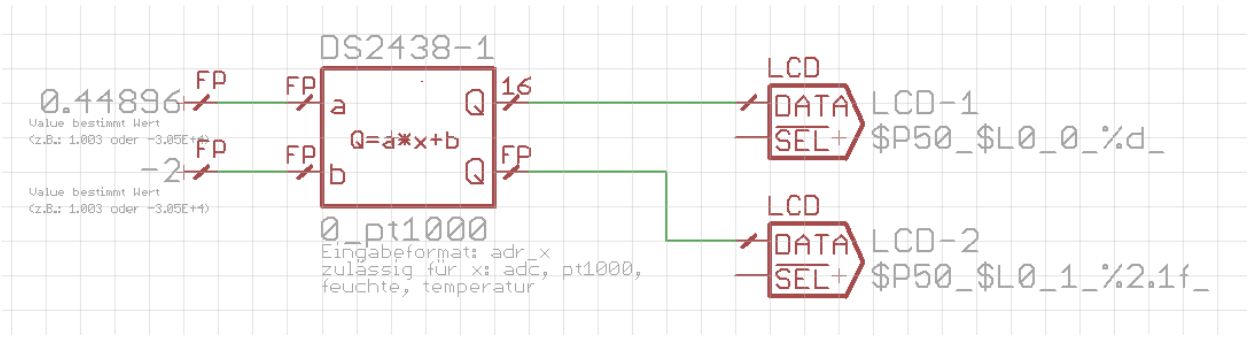
temperatur Messen und Ausgeben der Temperatur des Bausteins

Beispiel: ADC einen Spannungswert einlesen und ausgeben



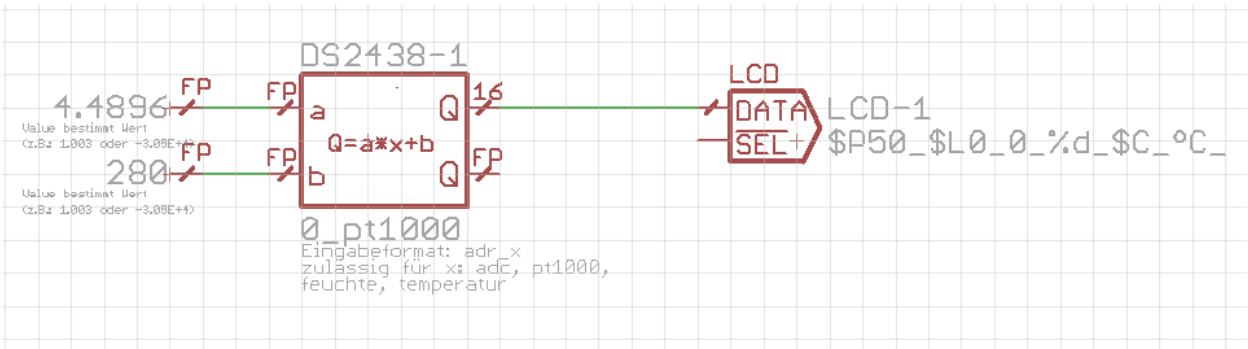
Die Spannung am Ausgang wird in Volt ausgegeben, da der Wandler eine Auflösung von 10mV pro Bit hat.

Beispiel: pt1000 eine Temperatur mit einem PT1000 Messfühler ermitteln und ausgeben



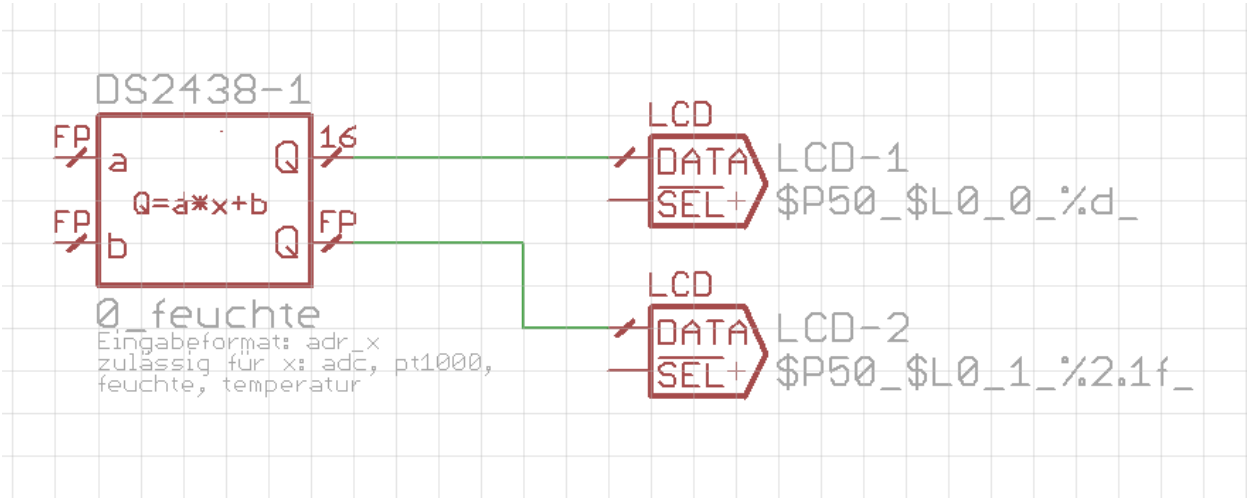
Die Temperatur wird über einen AD-Wandler mit einer Auflösung von +- 10Bit ermittelt. Ein Bit hat eine Auflösung von 0,2441mV. Gemessen wird eine Differenzspannung an einer Brücke. Je nach Schaltung und Temperaturbereich, kann die Schaltung entsprechend ausgelegt werden. Die Parameter für den Eingang a und b sind für die Kennlinie verantwortlich und müssen für die Schaltung und den zu messenden Temperaturbereich bestimmt werden.

Das 16Bit Temperaturformat lässt sich folgendermaßen berechnen:



Da bei diesem Format eine Nachkommastelle mit eingerechet ist, wird das Komma für die Variable a um eine Stelle nach links verschoben. Der Offset wird mit der Variablen b definiert.

Beispiel: feuchte die relative Feuchte ermitteln und ausgeben

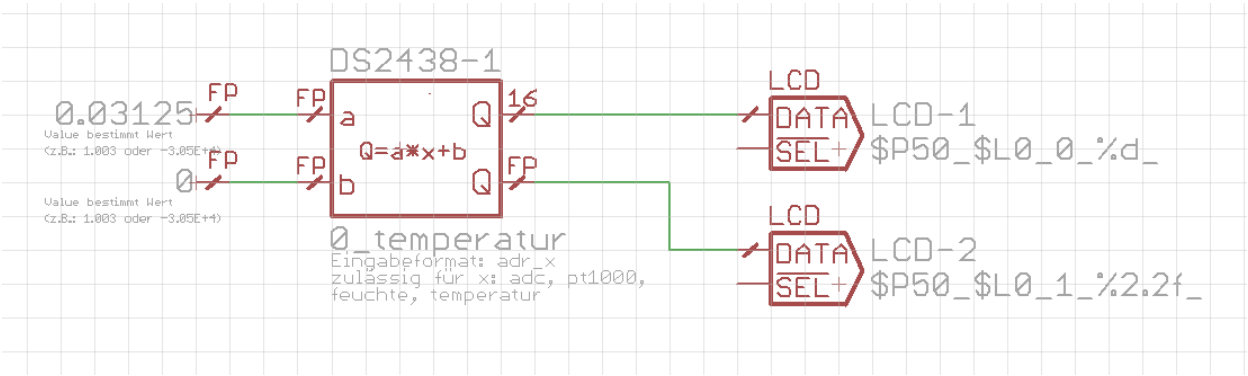


Die relative Feuchte wird in diesem Beispiel an Q_FP mit einer Nachkommastelle ausgegeben. Die Eingänge a und b werden bei der Ermittlung der Feuchte nicht verwendet. Die Feuchte wird mit dem internen Temperatursensor des DS2438 verrechnet. Die Berechnungen der Feuchte wird mit folgender Formel durchgeführt:

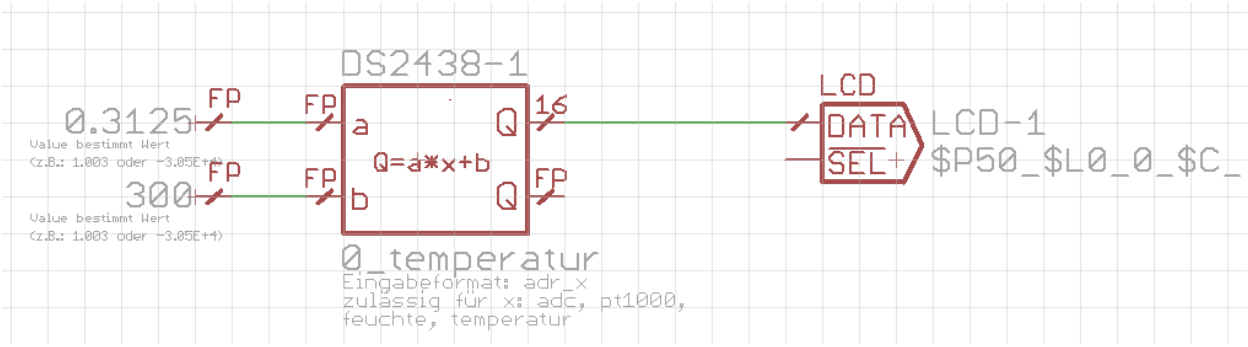
$$RH_{raw} = 161.29\frac{V_{AD}}{V_{DD}} - 25.80$$

$$RH_{true} = RH_{raw} / (1.0546 - 0.00216 * T_c)$$

Beispiel: temperatur die Temperatur des internen Temperaturfühlers des DS2438 ermitteln und ausgeben.



Der Temperaturwandler hat eine Auflösung von 0,03125°C pro Digit. Für die Temperatur-berechnung muss der Parameter a mit einem Wert von 0,03125 angegeben werden. Falls als Temperatur die interne Verarbeitung mit 16 Bit gewählt wird, so müssen die Eingänge mit folgenden Werten belegt werden: a = 0,3125, b = 300



POWER_Counter *Energiemessung*

Mit diesem Baustein kann die die Wärmeenergie gemessen werden.

Am Eingang TV wird die Vorlauftemperatur und am Eingang TR wird die Rücklauftemperatur erfasst. Mit einer pos. Flanke an CLK wird die Energiemenge ermittelt und abgespeichert. Mit dem Eingang MB kann der Messbereich umgeschaltet werden.

Eingang CLK

An dem CLK Eingang liegt das Signal von einem Durchflußmesser an. Dieser gibt die Pulse pro Durchflußmenge ab. Diese können über einen digitalen Eingang gelesen werden

Eingang MB

0 = Pulslänge in Sekunden	bezogen auf den letzten Puls
1 = Wärmemenge der laufenden Minute	wird bei xx:xx:00 auf 0 gesetzt
2 = Wärmemenge der letzten Minute	wird bei xx:xx:00 aktualisiert
3 = Wärmemenge der laufenden Stunde	wird bei xx:xx:00 aktualisiert
4 = Wärmemenge der letzten Stunde	wird bei xx:00:00 aktualisiert
5 = Wärmemenge des laufenden Tages	wird bei xx:00:00 aktualisiert
6 = Wärmemenge des letzten Tages	wird bei 00:00:00 aktualisiert

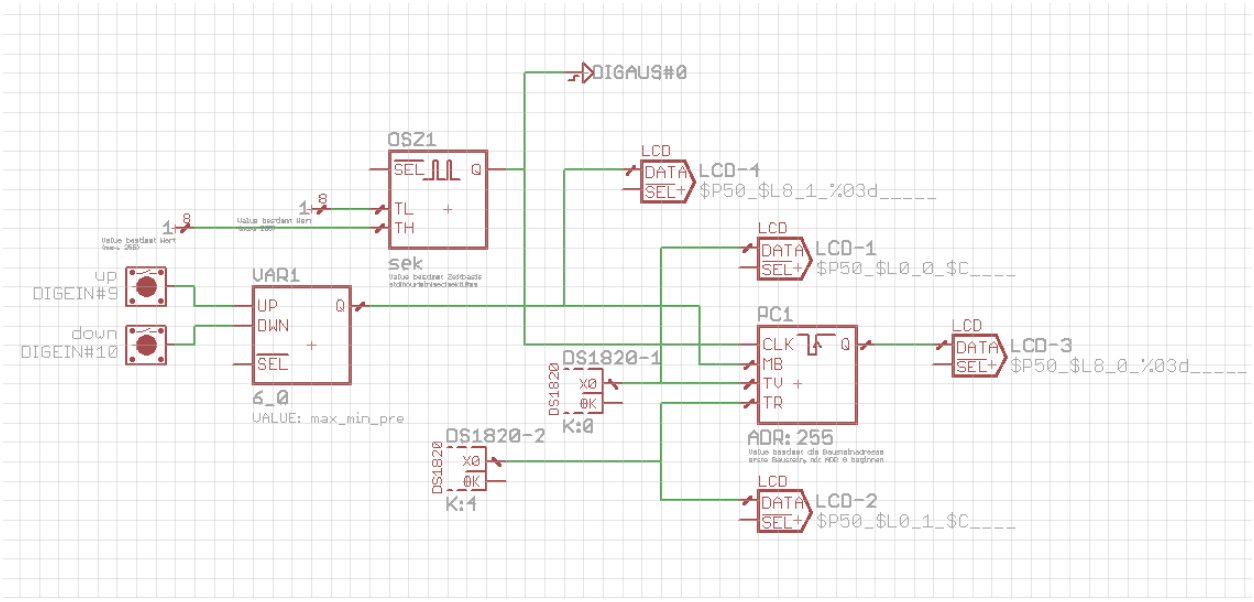
Eingang TV und TR

Mit diesen zwei Eingängen wird die Temperaturdifferenz gemessen. Ein geeignetes Bauelement für diese Temperaturmessung ist der Temperatursensor DS1820.

Eine Energieeinheit errechnet sich aus TV - TR.

Value: 0 bis 255, bestimmt die Adresse im EEProm. Der erste Baustein sollte mit der Adresse 0 belegt werden, der zweite mit der Adresse 1 usw. (größte Adresse ist 9, somit sind 10 Bausteine zulässig)

Beispiel: Testprogramm für den Baustein „power couner“



Zeitverzögerungen

Eingänge 1Bit: 1Bit, 8Bit, 16Bit, 32Bit : 0 ergibt 0; > 0 ergibt 1
FP : ergibt immer 0

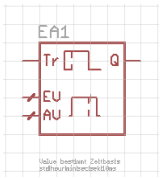
Die Eingänge für die Verzögerungszeiten werden in 16Bit Werte umgewandelt

Ausgang: Der Ausgang hat immer den Datentyp 1Bit

EINAUSSCHVZ *Einschalt und Ausschaltverzögerung*

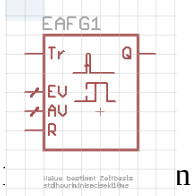
Wenn Tr für die Zeit EV auf 1 ist, geht Q auf 1. Q geht erst dann wieder auf 0, wenn Tr für die Zeit AV auf 0 war.

Value: std | min | sek | 10ms (default: 10ms)



EINAUSSCHVZ_FG *Flankengetriggerte Einschaltverzögerung*

Der Ausgang geht nach der Zeit EV auf 1 und bleibt für die Zeit AV auf 1. Ist AV 0, dann bleibt der Ausgang auf 0. Nicht retriggerbar.
R = 1: Ausgang bleibt 0.



Geht R auf 1, während der Ausgang auf 1 ist, dann wird der Ausgang sofort auf 0 gesetzt. Eine positive Flanke an R nach der Triggerflanke an Tr verhindert, dass der Ausgang nach Ablauf von EV auf 1 geht.

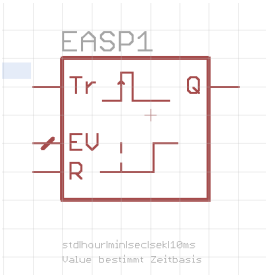
Value: std | min | sek | 10ms (default: 10ms)

Innerhalb der ersten 300 ms nach dem Start der internen Zeitzählung wird ein Pegelwechsel am Triggereingang nicht ausgewertet.

Typ. Anwendung: Entprellen von Schaltern

EINSCHALTVZ_SP Speichernde Einschaltverzögerung

Der Ausgang geht nach der Zeit EV auf 1. Nicht retriggerbar. Wenn der Eingang EV offen ist, wird das Bauteil zum RS-Flipflop bzw. zum Selbsthalterelais.

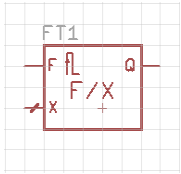


Value: std | min | sek | 10ms (default: 10ms)

Innerhalb der ersten 300 ms nach dem Start der internen Zeitzählung wird ein Pegelwechsel am Triggereingang nicht ausgewertet.

FREQU_TEILER *Frequenzteiler*

Der Ausgang ist 0, wenn X 0 oder 1 ist. Ist der Wert von X eine Zahl von 2 bis 65535, dann wird die Frequenz des Eingangssignals (F) um diesen Faktor verringert. Ein symmetrisches Ausgangssignal erhält man nur, wenn man gerade Werte für X wählt, oder wenn das Eingangssignal ein Taktverhältnis von 1:1 hat (1-Zeit und 0-Zeit gleich).

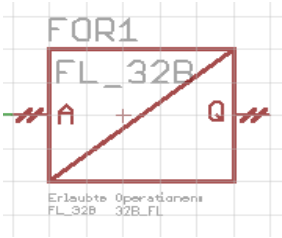


Innerhalb der ersten 300 ms nach dem Start der internen Zeitzählung wird ein Pegelwechsel am Triggereingang nicht ausgewertet.

Format *Umwandlung zwischen den Datentypen*

Mit diesem Baustein können Datenformate umgewandelt werden.

Am Eingang A kann ein beliebiger Datentyp eingespeist werden. Der Datentyp des Ausgangs wird über das Value des Bausteins festgelegt.



Der Baustein unterstützt folgende Umwandlungen:

8B =>	Das Eingangsformat wird in das 8Bit Format umgewandelt.
16B =>	Das Eingangsformat wird in das 16Bit Format umgewandelt.

32B => Das Eingangsformat wird in das 16Bit Format umgewandelt.

FL => Das Eingangsformat wird in eine Gleitkommazahl umwandelt.

LCD_AUSGABE, V24Ausgabe und SD_Card

Formatierte Ausgabe auf des LCD-Display, der RS232 Schnittstelle oder der Speicherkarte. Der Ausgabe-string gibt an, was und ggf. wann etwas ausgegeben werden soll.

SEL = 1 verhindert Ausgabe.

Bei SEL = 0 und $P im Value: periodische Ausgabe. Erfolgt keine Eingabe für eine periodische Ausgabe, so wird die Ausgabe über die positive Flanke an SEL getriggert.

_ wird in Leerzeichen umgewandelt, Leerzeichen in Value sind nicht erlaubt. Die Formatzeichen %, $ und \ können nur mit vorangestelltem \ verwendet werden.

Die Parameter in Value

Das **$-Zeichen** wird als Kennung für das Formatierungszeichen verwendet. Folgende Formatierungszeichen sind implementiert:

- $Lx_y gibt die Position an. (z.B. $L0_0_ oder $ L10_1_) x Zeichen in Zeile, y = Zeile

  Kurzschreibweise >> $L__ entspricht $L0_0_
  $L1__ entspricht $L1_0_
  Das Underline am Ende jeder Zahl ist wichtig, da mit diesem das Zahlenende ermittelt wird. (nur für die LCD Anzeige)
- $Pxx: Ausgabeintervall in 10ms (z.B. $P50_) 16Bit Variable; falls der Wert kleiner 10 ist, wird er auf 10 gesetzt.
- $b: 8Bit Wert als Binärzahl ausgeben
- $B: 16Bit Wert als Binärzahl ausgeben
- $F: gibt den Datentyp des DATA-Eingangs aus >> 1B, 8B, 16B, 32B und FP
- $w: Gibt den Wochentag aus
- $T: Gibt die Uhrzeit der Systemuhr im Format HH:MM:SS aus (z.B. 13:54:09)
- $C: Gibt einen Temperaturwert aus
- $Z: Gibt den am DATA-Eingang anliegenden Wert als Zeitstring aus

  Das Format ist Stunden:Minuten. (z.B. 13:22)
- $q: Gibt die interne Zykluszeit in ms aus (für Laufzeitoptimierung)
- $Q: Gibt die interne Anzahl der Zyklen pro Sekunde aus (für Laufzeitoptimierung)

- $n: Zeilenvorschub ausgeben (für RS232 und SD Card)
  Anzeige Löschen (für LCD Ausgabe)

- $I: Die Datenausgabe wird auf der Speicherkarte abgelegt. Der Dateinamen wird mit dem Index erweitert.

  ein Beispiel: $I110_ erzeugt den Dateinamen „SC_110.txt“

  Der Name der Ausgabenbausteine erhält durch Eagle ebenfalls einen Index. Folgende Bezeichnung wird für den Namen automatisch vergeben: SD-1, SD-2, SD-3 usw. Die Abarbeitung durch die microSPS erfolgt in der gleichen Reihenfolge. Falls eine Ausgabe nacheinander in einer Zeile erfolgen soll, so kann dies über den Namen festgelegt werden.

Das **%-Zeichen** wird als Kennung für die Ausgabe von Variablen verwendet. Für die Ausgabe gelten die Formatierungsregeln von printf. Unterstützt werden folgende Formate:

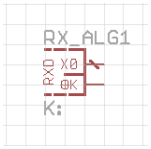
- %x, %X unsigned hex int (16Bit)
- %lx, %lX unsigned hex int (32Bit)
- %d signed int (16Bit)
- %ld signed int (32Bit)
- %u: unsigned int (16Bit)
- %lu unsigned int (32Bit)
- %e, %E [-]d.dddddde±dd (32Bit)
- %f [-]d.dddddd (32Bit)

Für das Formatieren der Zahl können zwischen dem %Zeichen und dem Formatzeichen weiter Zeichen eingegeben werden. Hier gelten die Regeln für printf.

Beispiel:

- $P10_ Ausgabe alle 100ms (10 * 10ms)
- $L0_1_ Ausgabe an Position Zeichen = 0 Zeile = 1 (für LCD Anzeige)
- $P100_\n_$b Ausgabe ein mal pro Sekunde, Zeilenvorschub und Binäre Zahl mit 8 Bit (01011001)
- $P50_____ Ausgabe von Leerzeichen
- $P50_$L0_0_Wert: %04x Ausgabe einer 16Bit Hex Zahl an die LCD Anzeige

LINK_RX_ALG *Analogeingang 16Bit über CAN_Bus*

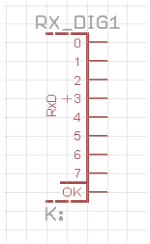


Empfängt einen Datensatz mit einem Analogwert. Der Funktionsbaustein bildet den Empfangsteil eines LINK_TX_ALG Funktionsbausteins. Die Kanalnummer wird mit Value festgelegt. Diese muss mit der Kanalnummer des Sendeblocks übereinstimmen.

Die Empfänger besitzen jeweils einen Digitalausgang (OK), der angibt, ob Daten auf diesem Kanal gelesen wurden.
High: gültige Daten sind vorhanden
Low: es liegt kein gültiger Datensatz vor (Timeout = 5sek)

Die Adresse 0 wurde für die syncronisation der Uhrzeit belegt. Wird diese Adresse ausgewählt, so wird mit dem eingegangen Zeitstempel die interne Uhrzeit gesetzt. Die ist nur sinnvoll, wenn keine Hardware-Uhr installiert ist.

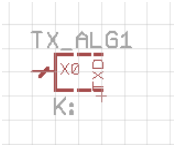
LINK_RX_DIG *8Bit Binärwert über CAN_BUS*



Empfängt einen Datensatz mit 8 Binärwert über den CAN-BUS. Dieser Funktionsbaustein bildet den Empfangsteil eines LINK_TX_DIG Funktionsbausteins. Die Kanalnummer muss mit der Kanalnummer des Sendeblocks übereinstimmen. Mit Value wird die Kanalnummer festgelegt. Diese muss mit der Kanlnummer des Sendeblocks übereinstimmen.

Die Empfänger besitzen jeweils einen Digitalausgang (OK), der angibt, ob Daten auf diesem Kanal eintreffen.
High: gültige Daten sind vorhanden
Low: es liegt kein gültiger Datensatz vor (Timeout = 5sek)

LINK_TX_ALG *Analogausgang 16Bit über CAN_BUS*



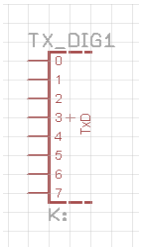
Dieser Funktionsbaustein bildet den den CAN BUS Sendeteil. Der Funktionsbaustein LINK_RX_ALG stellt die Schnittstelle auf einer anderen microSPS dar. Die Kanalnummer muss mit der Kanalnummer des Empfangsblocks übereinstimmen.
Value bestimmt die Kanalnummer

Die Kanäle 10000-11000 sind reserviert und sollten nicht verwendet werden!

Die Adresse 0 wurde für die syncronisation der Uhrzeit belegt. Wird diese Adresse ausgewählt, so wird jede Minute ein Zeitstempel ausgegeben, der sich aus Wochentag, Stunden und Minuten zusammensetzt.

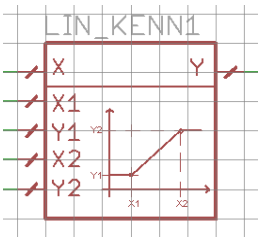
LINK_TX_DIG *Digitalausgang 8Bit über CAN_BUS*

Sendet einen Datensatz mit 8 Binärwerten über den CAN-BUS. ALS Empfangsteil wird der Funktionsbaustein LINK_RX_DIG verwendet.
Die Kanalnummer muss mit der Kanalnummer des Empfangsblocks übereinstimmen.
Value bestimmt die Kanalnummer
Die Kanäle 10000-11000 sind reserviert und sollten nicht verwendet werden!



LIN_KENNLINIE *Lineare Kennlinie*

Mit diesem Baustein kann man eine linear Funktion anhand zweier Punkte (X1;Y1)(X2;Y2) definieren.
Wenn **Y1 < Y2** wird der Eingangswert X wird nach der Formel (pos. Steigung)



Y = Y1 + ((X-X1) * ((Y2-Y1) / (X2 - X1)))

umgerechnet und an den Eckpunkten begrenzt.
Begrenzung an den Eckpunkten:
Y = Y1 , wenn X kleiner als X1 ist
Y = Y2, wenn X größer als X2 ist

Wenn **Y1 > Y2** wird der Eingangswert x nach der Formel (neg. Steigung)
Y = Y1 - ((X-X1) * ((Y1-Y2) / (X2 - X1)))
umgerechnet und an den Eckpunkten begrenzt.
Begrenzung an den Eckpunkten:
Y = Y1 , wenn X kleiner als X1 ist
Y = Y2, wenn X größer als X2 ist

Logische Verknüpfungen

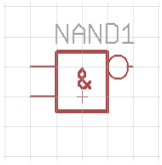
Für die Datentypen der logischen Verknüpfungen gelten folgende Regeln:

Eingänge: 1Bit, 8Bit, 16Bit, 32Bit : 0 ergibt 0; > 0 ergibt 1
FP : ergibt immer 0

Ausgang: hat immer den Datentyp 1Bit

NAND

Der Ausgang ist nur dann auf 0, wenn alle Eingänge auf 1 sind. Ein unbeschalteter Eingang wird als 0 bewertet.



NICHT

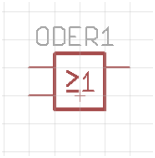
Invertiert den Eingangswert.

NOR

Der Ausgang ist NICHT auf 1, wenn einer der Eingänge auf 1 ist.

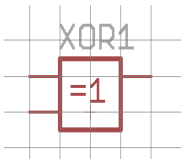
ODER

Der Ausgang ist auf 1, wenn einer oder beide Eingänge auf 1 sind.



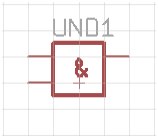
XOR

Der Ausgang ist auf 1, wenn einer der Eingänge auf 1 ist.

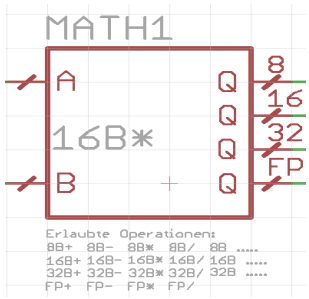


UND

Der Ausgang ist nur dann auf 1, wenn alle Eingänge auf 1 sind.



MATH *Rechenfunktion*



Die Funktion führt die im VALUE angegebene mathematische Operation aus und gibt am Ausgang das Ergebnis weiter. Der Funktionsbaustein kann 16Bit, 32Bit und Gleitkommazahlen verarbeiten.

Rechnen mit 8Bit Variablen

Wir ein VALUE für die 8Bit Arithmetik gewählt, so werden die Eingänge in einen 8Bit Wert umgewandelt und intern mit 8Bit Variablen verrechnet. Das Ergebnis wird am Ausgang als 8Bit,16Bit Signal, 32Bit und FP Signal ausgegeben..

8B<< : a um b Stellen nach Links schifften
8B>>: a um b Stellen nach Rechts schifften
8B&: a und b bitweise UND verknüpfen
8B|: a und b bitweise Oder verknüpfen

Rechnen mit 16Bit Variablen

Wir ein VALUE für die 16Bit Arithmetik gewählt, so werden die Eingänge in einen 16Bit Wert umgewandelt und intern mit 16Bit Variablen verrechnet. Das Ergebnis wird am Ausgang als 8Bit,16Bit Signal, 32Bit und FP Signal ausgegeben..

16B+: a und b werden addiert.
16B-: b wird von a subtrahiert.
16B*: a und b werden multipliziert.
16B/: a wird durch b geteilt.
16B%: Modulo von A durch B
16B<<: a um b Stellen nach Links schifften
16B>>: a um b Stellen nach Rechts schifften
16B&: a und b bitweise UND verknüpfen
16B|: a und b bitweise Oder verknüpfen

Rechnen mit 32Bit Variablen

Wir ein VALUE für die 32Bit Arithmetik gewählt, so werden die Eingänge in einen 32Bit Wert umgewandelt und intern mit 32Bit Variablen verrechnet. Das Ergebnis wird am Ausgang als 8Bit,16Bit Signal, 32Bit und FP Signal ausgegeben.

32B+: a und b werden addiert.
32B-: b wird von a subtrahiert.
32B*: a und b werden multipliziert.
32B/: a wird durch b geteilt.
32B%: Modulo von A durch B

32B<<: a um b Stellen nach Links schifften
32B>>: a um b Stellen nach Rechts schifften
32B&: a und b bitweise UND verknüpfen
32B|: a und b bitweise Oder verknüpfen

Rechenfunktion für Gleitkommazahlen

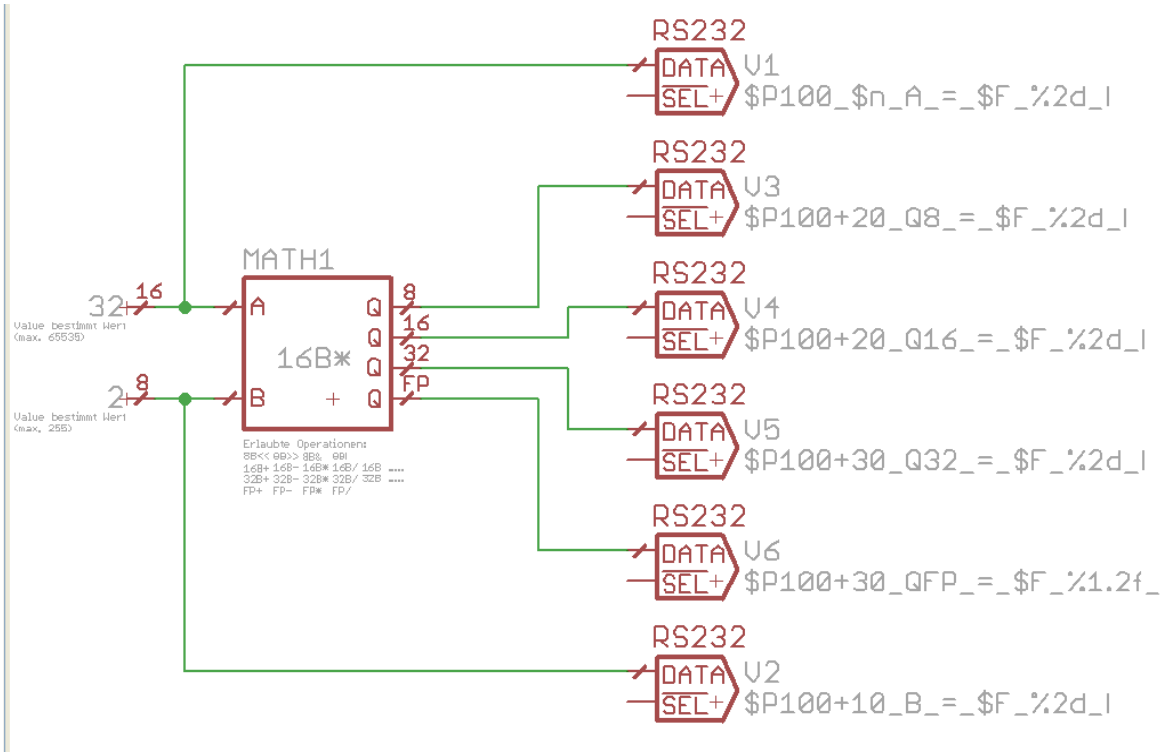
Wir ein VALUE für die FP Arithmetik gewählt, so werden die Eingänge in einen FP Wert umgewandelt und intern mit FP Variablen verrechnet. Das Ergebnis wird am Ausgang als 16Bit Signal und FP Signal (32 Bit) ausgegeben.

FP+: A und B werden addiert.
FP-: B wird von A subtrahiert.
FP*: A und B werden multipliziert.
FP/: A wird durch B geteilt.

FP ist die Abkürzung für Gleitkommazahlen

Beispiel: In diesem Beispiel wird ein 16Bit Wert und 8Bit Wert in einer 16Bit Arithmetik verrechnet. Zur Überprüfung werden die Ein und Ausgangswerte über die RS232 Schnittstelle ausgegeben.

Schaltplan:



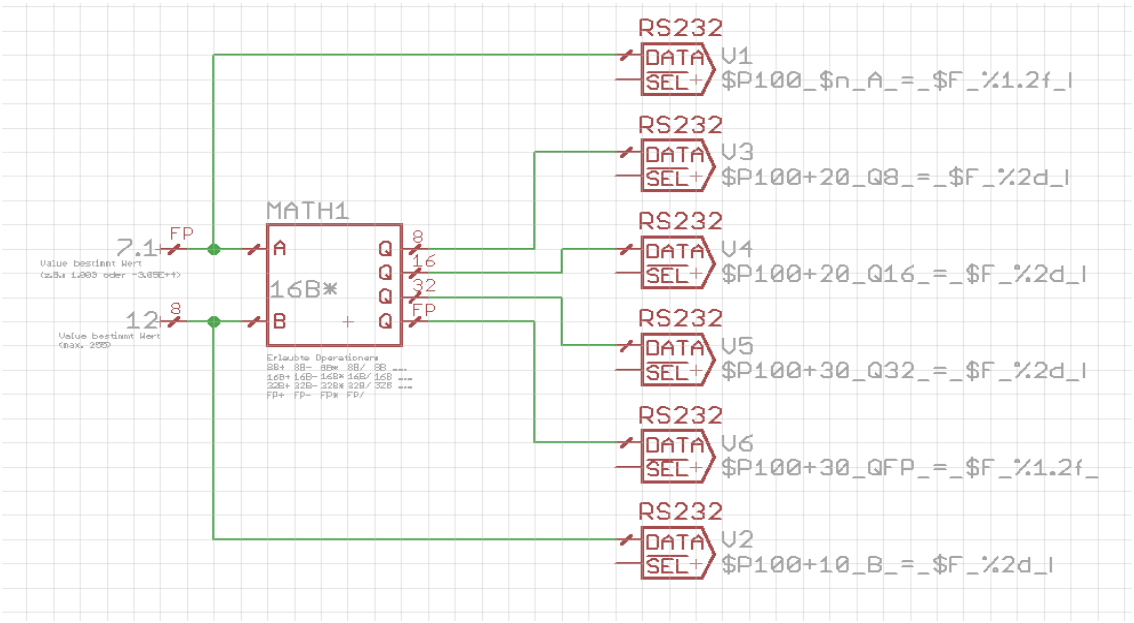
Ausgabe über die RS232 Schnittstelle:

Mit $F wird der Variablentyp anzeigt.

```
A = 16B 32 |B = 8B  2 |Q8 = 8B 64 |Q16 = 16B 64 |Q32 = 32B 64 |QFP = FP 64.00
A = 16B 32 |B = 8B  2 |Q8 = 8B 64 |Q16 = 16B 64 |Q32 = 32B 64 |QFP = FP 64.00
A = 16B 32 |B = 8B  2 |Q8 = 8B 64 |Q16 = 16B 64 |Q32 = 32B 64 |QFP = FP 64.00
```

Beispiel: In diesem Beispiel wird ein FP Wert und 8Bit Wert in einer 16Bit Arithmetik verrechnet. Zur Überprüfung werden die Ein und Ausgangswerte über die RS232 Schnittstelle ausgegeben.

Schaltplan:



Ausgabe über die RS232 Schnittstelle:

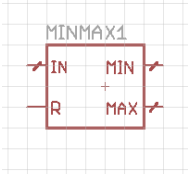
```
A = FP 7.10 |B = 8B 12 |Q8 = 8B 84 |Q16 = 16B 84 |Q32 = 32B 84 |QFP = FP 84.00
A = FP 7.10 |B = 8B 12 |Q8 = 8B 84 |Q16 = 16B 84 |Q32 = 32B 84 |QFP = FP 84.00
A = FP 7.10 |B = 8B 12 |Q8 = 8B 84 |Q16 = 16B 84 |Q32 = 32B 84 |QFP = FP 84.00
```

Das gleiche Beispiel mit einer FP Multiplikation

```
A = FP 7.10 |B = 8B 12 |Q8 = 8B 85 |Q16 = 16B 85 |Q32 = 32B 85 |QFP = FP 85.20
A = FP 7.10 |B = 8B 12 |Q8 = 8B 85 |Q16 = 16B 85 |Q32 = 32B 85 |QFP = FP 85.20
A = FP 7.10 |B = 8B 12 |Q8 = 8B 85 |Q16 = 16B 85 |Q32 = 32B 85 |QFP = FP 85.20
```

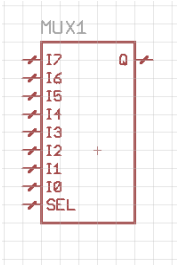
MINMAX *Anzeige von Minimal- und Maximalwert*

Die Ausgänge MIN und MAX zeigen den Minimalwert bzw. den Maximalwert an, den das Eingangssignal seit dem letzten Resetsignal erreicht hat. Eine 1 am Eingang R setzt die Ausgänge auf den gegenwärtigen Eingangswert.



MULTIPLEXER

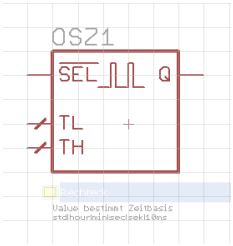
Abhängig vom Wert an SEL (modulo 8) wird einer der Eingänge I0 bis I7 zum Ausgang durchgeschaltet.



OSZILLATOR

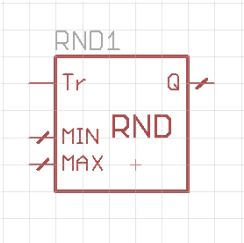
Liefert am Ausgang ein periodisches Signal mit der Periodendauer TL + TH, wobei TL die Low-Zeit ist, also die Zeit, für die das Signal auf 0 liegt, und TH die High-Zeit, also die Zeit, für die das Signal auf 1 liegt.

Value: std | min | sek | 10ms (Default: 10ms)

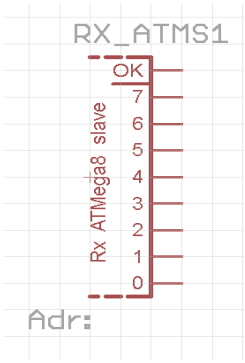


RANDOM *Zufallsfunktion*

Bei jeder positiven Flanke von Tr wird am Ausgang eine neue Zufallszahl bereit gestellt. Die Zahl kann nur Werte von MIN bis MAX annehmen. Um einen Würfel zu simulieren, würde man also an MIN dern Wert 1 und an MAX den Wert 6 anlegen.



RX_ATMS *Receive für den ATMega Slave*

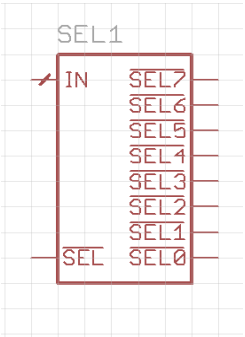


Über diesen Baustein werden die Daten der Erweiterungsmodule gelesen.

Value: Adresse

Als Adresse nur geradzahlige Werte nehmen, da Bit0 als Schreib / Lesebit verwendet wird.

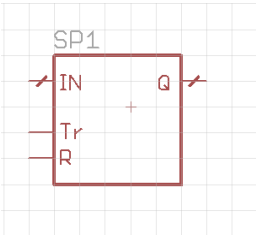
SELECT *Selektfunktion, 1 aus 8 Decoder*



SEL = 1	oder IN größer als 7: Alle Ausgänge liegen auf 1.
SEL = 0:	Liegt am Eingang IN ein Singal mit einem Wert zwischen 0 und 7 an, dann ist genau ein Ausgang auf 0.
Bei IN = 0	liegt SEL0 auf 0, bei IN = 1 SEL1 und so weiter.

Diese Funktion eignet sich z.B. dazu, mit Hilfe eines vorgeschalteten Zählers (COUNTER) aus einer Gruppe von Bedien- oder Anzeigeelementen jeweils eine zu aktivieren.

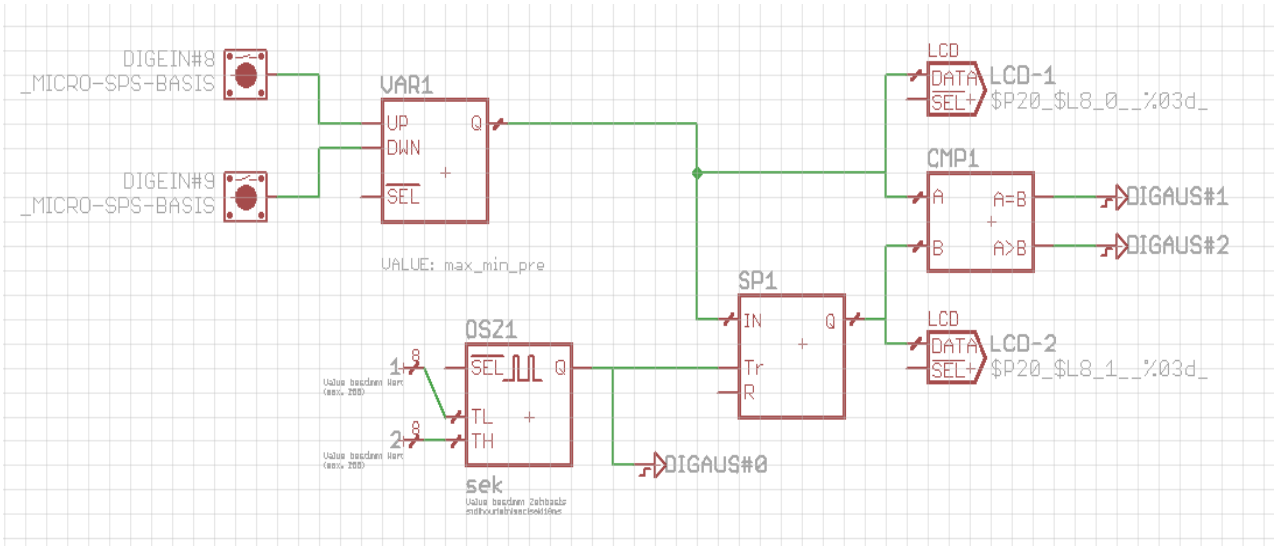
Speicher *einen 16Bit Wert zwischenspeichern*



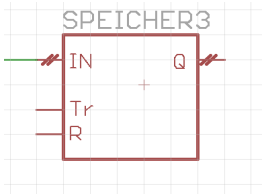
Bei der positiven Flanke von Tr übernimmt der Ausgang den gegenwärtigen Wert des Eingangssignals. Der Ausgang bleibt auf diesem Wert, bis zur nächsten positiven Flanke von Tr oder bis mit der positiven Flanke der Ausgang auf 0 gesetzt wird. Der Wert von Q wird im EEProm abgespeichert.

Maximal 40 dieser Funktionsblöcke von WERT_VAR und Speicher dürfen gesetzt werden.

Beispiel: eine Testschaltung für den Baustein „Speicher“

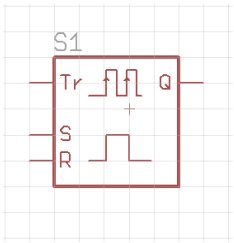


Speicher *einen 32Bit Wert zwischenspeichern*



Bei der positiven Flanke von Tr übernimmt der Ausgang den gegenwärtigen Wert des Eingangssignals. Der Ausgang bleibt auf diesem Wert, bis zur nächsten positiven Flanke von Tr oder bis er mit R = 1 auf 0 gesetzt wird.

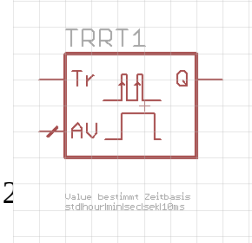
STROMSTOSSREL



Wenn R und S auf 0 sind, gilt: Die positive Flanke von Tr setzt Q auf 1, wenn Q bisher auf 0 war, und umgekehrt. S = 1 setzt Q auf 1. R = 1 setzt Q auf 0 (R hat Vorrang vor S).

Innerhalb der ersten 300 ms nach dem Start der internen Zeitzählung wird ein Pegelwechsel am Triggereingang nicht ausgewertet.

TREPPENL_RT *Treppenlichtschalter*



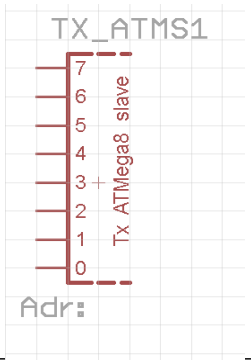
Bei der positiven Flanke von Tr geht Q auf 1. Q geht auf 0, wenn die Zeit AV seit der positiven Flanke von TR verstrichen ist. Ein erneuter Tr-Impuls, bevor AV

abgelaufen ist, startet den Timer für AV neu.

VALUE: std | min | sek | 10ms (Default: 10ms)

Innerhalb der ersten 300 ms nach dem Start der internen Zeitzählung wird ein Pegelwechsel am Triggereingang nicht ausgewertet.

TX_ATMS *Transmit für den ATMega Slave*

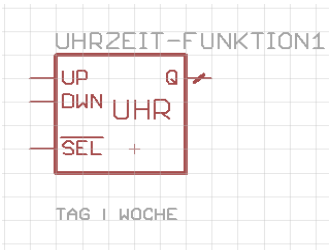


Über diesen Baustein können Daten an die Erweiterungsmodule gesendet werden.

VALUE: Adresse

Als Adresse nur geradzahlige Werte nehmen, da Bit0 als Schreib / Lesebit verwendet wird.

UHRZEIT



Liefert die Uhrzeit als Anzahl von **Minuten** ab Sonntag 0 Uhr. 0 entspricht SO 00:00

Bei der Einstellung TAG ist der Größte Wert SO 23:59 Dies entspricht einem Wert von 1440 Minuten

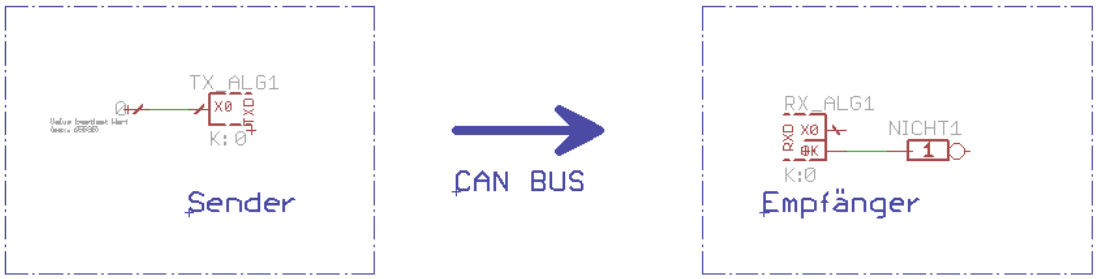
Wurde die Einstellung Woche ausgewählt, ist der größte Wert 10080. Dies ergibt SA 23:59

Ist eine **Echtzeituhr** auf der microSPS bestückt, wird diese als Zeitgeber verwendet. Sonst wird die Zeit über die interne Quarzfrequenz abgeleitet.

Mit den UP/DWN-Eingängen kann man die Zeit wie mit der Funktion WERT_VAR einstellen. Die Funktion darf nur einmal in der Schaltung verwendet werden.

Value: TAG | WOCHE (Default: TAG)

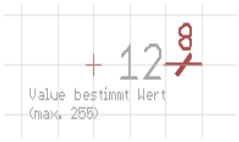
Die Uhrzeit kann auch über den CAN Bus syncronisiert werden. Damit ist nur eine Hardware-Uhr auf eine microSPS erforderlich. Hier ein Schaltungsbeispiel:



für VALUE beim Sender und Empfänger eine 0 Eingeben

Für die Syncronisation der Uhrzeit wurde die Adresse 0 von Link_Tx_ALG und Link_Rx_ALG reserviert. Die Bausteine für die Datenübertragung müssen mindestens mit einem Anschluss verbunden werden, damit diese in der Übersetzung (mit F12) auch Byte-Code erzeugt wird. Bausteine, die nur auf dem Schaltplan positioniert werden, aber nicht angeschlossen sind werden bei der Übersetzung ignoriert.

WERT _8 *Feste eingestellter Wert mit 8Bit*



Liefert einen 8Bit Wert der über Value angegeben wird.

Value erforderlich:

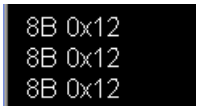
Dezimalzahl -128 bis 255

Hexzahl 0x00 bis 0xFF

Beispiel: Über Wert_8 wurde ein 8Bit Signal mit einem Hexwert von 0x12 erzeugt. Der Wert wird als Hexzahl über die RS232 Schnittstelle ausgegeben.

Schaltplan

Ausgabe über die RS232 Schnittstelle



WERT _16 *Feste eingestellter Wert mit 16Bit*

Liefert einen 16Bit Wert der im Value angegeben wird.

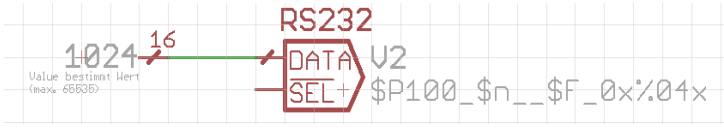
Value erforderlich:

Dezimalzahl -32.768 bis 65.535

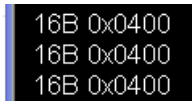
Hexzahl 0x0000 bis 0xFFFF

Beispiel: Über Wert_16 wurde ein 16Bit Signal mit einem Dezimalwert von 1024 erzeugt. Der Wert wird als Hexzahl über die RS232 Schnittstelle ausgegeben.

Schaltplan



Ausgabe über die RS232 Schnittstelle



WERT_32 *Feste eingestellter Wert mit 32 Bit Binärzahl*

Liefert einen 32Bit Wert der im Value angegeben wird.

Value erforderlich:

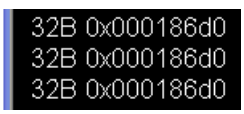
Dezimalzahl von -2.147.483.648 bis 2.147.483.647

Hexzahl 0x00000000 bis 0xFFFFFFFF möglich

Beispiel: Über Wert_32 wurde ein 32Bit Signal mit einem Dezimalwert von 100048 erzeugt. Der Wert wird über den Baustein V24Ausgabe als Hexzahl ausgegeben.

Schaltplan

Ausgabe über die RS232 Schnittstelle



WERT_FP *Feste eingestellter Wert für eine 32Bit Gleitkommazahl*

Liefert einen unveränderlichen Wert als Gleitkommazahl, der im Value angegeben wird.

Value erforderlich:

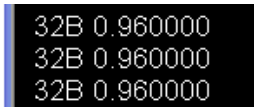
Zahl von -1,5E-45 bis +3.4E38 (Genauigkeit, 7 bis 8 Stellen)

Eingabe mit Exponent möglich. z.B.: 15.3E6

Beispiel: Über Wert_FP wurde ein 32Bit Signal mit einem Gleitkommazahl von 0,96 erzeugt. Der Wert wird über die RS232 Schnittstelle ausgegeben.

Schaltplan

Ausgabe über die RS232 Schnittstelle



WERT_TEMPERATUR *Fest eingestellter Temperaturwert*

Liefert einen unveränderlichen Temperaturwert von -30.0 bis 140.0, der im Value angegeben wird. Dieser Wert wird intern nach der Formel: *interner Wert = Temperatur * 10 + 300* umgerechnet.

Datenformat = 16Bit

Beispiel: Über Wert_Temperatur wurde ein Temperaturwert von 25,0°C erzeugt. An der RS232 Schnittstelle wird der Datentyp, der Temperaturwert und den Dezimalwert ausgegeben.

Schaltplan

Ausgabe über die RS232 Schnittstelle

```
16B 25.0  550
16B 25.0  550
16B 25.0  550
```

WERT_UHRZEIT *Fest eingestellter Uhrzeitwert*

Liefert einen unveränderlichen Zeitwert von 00:00 bis 23:59 der im Value angegeben wird.
Optional kann man den Wochentag angeben:
Mo:14:10 ---> Montag, 14:10 Uhr

Erlaubte Wochentage sind: So, Mo, Di, Mi, Do, Fr, Sa

Datenformat: 16 Bit

Beispiel: Über Wert_Urzeit wurde ein Zeit von Mo 14:30 erzeugt. An der RS232 Schnittstelle wird der Datentyp, der Wochentag, die Uhrzeit und der Dezimalwert in Minuten ausgegeben.

Schaltplan:

Ausgabe über die RS232 Schnittstelle

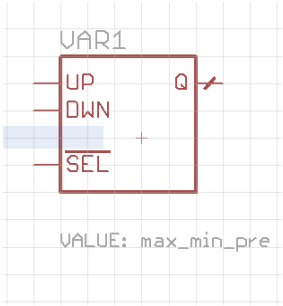
```
16B MO  14:30  2310
16B MO  14:30  2310
16B MO  14:30  2310
```

WERT_VAR *Benutzer definierbarer variabler Wert*

Liefert einen veränderlichen Wert von 0 bis 65535, der auch nach einem Spannungsausfall gespeichert bleibt.
UP = 1: Wert wird erhöht.
DWN = 1: Wert wird gesenkt.
SEL = 1: Wert bleibt unverändert

Value: max | max_min | max_min_default

max, min, default stehen für:
- Zahlenwerte von 0 bis 65536
- Temperaturen von -30.0°C bis 1000°C
- Uhrzeiten 00:00 bis Sa:23:59



Wenn kein Value angegeben wird, ist der Defaultwert 1000_0_500, das heißt es sind Werte von 0 bis 1000 möglich, und der voreingestellte Wert bei einem leeren EEProm ist 500. Bei einer Änderung von Q wird das neue Q im EEProm abgespeichert und nach einem Reset wieder geladen.

Bei der Uhrzeit kann als Wert TAG oder WOCHE eingegeben werden. Wenn kein Wert eingegeben wurde, so wird TAG ausgewählt.

Maximal 40 dieser Funktionsblöcke von WERT_VAR und Speicher dürfen gesetzt werden.

REGLER_PID

Parameter für den PID-Regler.

- Soll: Sollwert Eingang (Bereich 0 … 1000)
- Ist: Istwert Eingang (Bereich 0 … 1000)
- Kp: Verstärkung des Proportionalanteils (interner Teiler mit 100, 100 entspricht 1,0)
- Ki: Verstärkung des Integralanteils (interner Teiler mit 100, 100 entspricht 1,0)
- Kd: Verstärkung des Differenzialanteils (interner Teiler mit 100, 100 entspricht 1,0)
- Ta: Abtastzeit in 10-ms-Intervallen
- Q: Ausgang (0...10000, Ausgangs Wert liegt mittig um 500)

Ein P-Regler alleine erzeugt eine bleibende Regelabweichung.

Der I-Anteil lässt z.B. so lange den Ausgang des Reglers ansteigen, solange eine Abweichung vorhanden ist. Es wird dazu in jedem Regler Intervall die Soll-Ist-Abweichung multipliziert mit Ki auf den Ausgangswert aufaddiert. Das sorgt dafür, dass der Ausgangswert irgendwann an den Anschlag läuft, solange noch Regelabweichung vorhanden ist.

Der D-Anteil soll ein Überschwingen des Systems dämpfen, indem die Steigung als Gegenkopplung verwendet wird.

Programmablauf:

prüfen ob Sollwert < 1000 sonst auf 1000 begrenzen

prüfen ob Istwert < 1000 sonst auf 1000 begrenzen

Regelabweichung berechnen (soll – ist)

P-Anteil berechnen >> Regelabweichung * (Kp /100)

I-Anteil berechnen >> Regelabweichung * (Ki/100) + I_alt => ergibt neues I_alt >> i_alt wird auf 500 begrenzt

D-Anteil >> Regelabweichung * (Kd/100) + (D_alt / 2) => ergibt neues D_temp >> D_temp > D_alt => D-Anteil = D_temp – D_alt sonst D-Anteil = 0 >> D_alt = D_temp

Q = P-Anteil + I-Anteil + D-Anteil

Q wird auf 1000 begrenzt

Die Rechnungen werden mit Integer Variablen 16 Bit durchgeführt.

Versionshistorie:

Bei der Version „microSPS_V5_04.lbr“ wurde die Verarbeitung der Signale umgestellt. Aufgrund einer Erweiterung der Signalzahl wurde eine Aufteilung der 1Bit und 16Bit Signale vorgenommen. Zusätzlich wurden in dieser Version 32Bit Signale eingeführt. Mit den 32Bit Signalen ist nun ein floating point Arithmetik möglich. Der Merkerzähler wurde von 8 Bit auf 16Bit umgestellt, so dass auch hier eine mögliche Eng stelle bei den Merkern behoben wurde.

Für die Signale und Merker steht ein RAM-Bereich von 1k Byte zur Verfügung. Es können maximal 254 1Bit Signale, 126 16Bit Signale, 126 32Bit Signale und 1024 Merker eingesetzt werden. Jedoch die Gesamt Größe von 1k Byte RAM darf nicht überschritten werden.

Für die Schaltplan Übersetzung in Eagle muss die Datei sps-makelist_V1_00.inc in der (Version V0.94) vorhanden sein.

02.10.10 Erweiterung mit der Speicherkarte

Als Interpreter (Firmware) ist folgende Version erforderlich:

microSPS X1	main_128g.hex
microSPS M07	xxxx

09.12.10 Sensor mit DS2834 eingeführt

microSPS X1	xxxx
microSPS M07	V1.00l