MySPS

1. Einleitung

Achtung:

Das Programm ist für Aus- und Weiterbildung gedacht. Es darf nicht in Produktiven Umgebungen eingesetzt werden (dazu ist das Programm zu wenig getestet und ausgereift).

Haftungsausschluss:

Der Entwickler der Software (ich) haftet nicht für eventuelle Schäden an Computer, Mikrocontroller, Hard- oder Software und persönlichen Daten. Die Nutzung erfolgt auf eigenes Risiko.

Garantieausschluss:
Es besteht kein Anspruch auf Garantie oder Gewährleistung im Falle von Funktionsausfällen oder Schäden.

Weitere Hinweise:

Bei MySPS handelt es sich um ein privates Projekt. Ich verfolge hiermit keine kommerziellen Ziele. Ich arbeite nicht in der Automatisierungstechnik. Mit SPS-Steuerungen hatte ich bei einem Kurs bei einer Handwerkskammer, sowie bei der Ausbildung zum Staatlich geprüften Techniker (Datentechnik) zu tun.

2. Technische Daten

MySPS kann auf den Microcontrollern ATMega8, AtMega16 und ATMega32 laufen. Mit dem ATMega16 wurden keine Tests gemacht.

Das SPS-Programm wird im EEPROM des Controllers abgelegt (uns ist somit auf 512 (AtMega8 und AtMega16) bzw. 1024 Bytes (AtMega32) beschränkt).

Das Prozerssabbild liegt im RAM des Microcontrollers (0x0100 – max. 0x019F).

Das SPS-Programm (AWL) wird mit einem Windows Programm geschrieben und von dort aus über eine serielle Schnittstelle in einem aufbereitetem Format (Tokens, Byteadresse, Bitmaske) zum Controller übertragen. Ein SPS-Befehl benötigt im Mikrocontroller 1-3 Bytes in EEPROM.

Der Controller arbeitet mit einer festen Zykluszeit (200 Zyklen/Sekunde). Die Timer im Controller basieren auf dieser Zykluszeit.

Das Programm das auf dem ATMega läuft, ist in Assembler geschrieben. Der PC-Teil ist in C# geschrieben. Auf dem PC muss das DotNet Framework 2.0 (.Net 2.0) installiert sein.

Bis zu 30 Ein- oder Ausgänge.

Flexible Ein- / Ausgangskonfiguration über das Windows-Programm.

Bis zu 6 Analogeingänge (Wertebereich 0-255). Die Werte der AD-Wandlung stehen in den Eingangsbytes ab 9 zur Verfügung (ADC0 -> EB9, ADC1 -> EB10…).

48 Merkerbytes (M1.0 bis M48.7) sowie verschiedene Sondermerker

Von 16 Merkerbits können Steigende-Flanken abgefragt werden, mit weiteren 16 Merkerbits können fallende Flanken ausgewertet werden (Natürlich kann die Flanke auch im Programm mit einem Hilfsmerker ermittelt werden).

16 Zähler (Z1...Z16) (Bereich 0-255)

16 Timer (T1...T16) (1-255 mit Faktor 1/100s, 1/10s, 1s oder 10s)

nicht unterstützt werden:

Bausteine (BEB,BEA), Funktionsbausteine (FB..), Organisationsbausteine (OB..), Datenbausteine,

Remanente Merker, Festpunktzahlen,

Programmierung in KOP, FUP…

3. Einstellungen und Ein-/Ausgangs- bzw. Portkonfiguration

1. Konfiguration des Mikrocontrollers und seiner Anschlüsse.

   Die Mikrocontroller-Konfiguration wird unter „Datei – Port Konfiguration“ vorgenommen:

   Zuerst den benutzten Mikrocontroller wählen, dann die Anzahl der ADCs (Analog-Digital Converter). Mit den Haken REFS0 und REFS1 kann man die gleichnamigen Bits im Register ADMUX des ATMega beeinflussen (Referenz für die AD-Wandler).

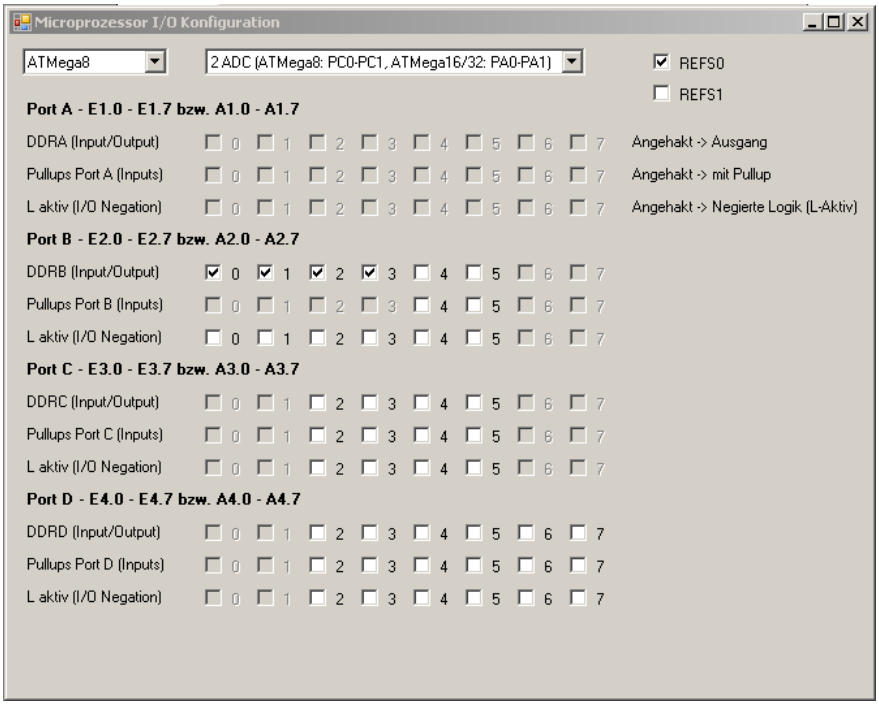
   Darunter stehen für jeden Port 3 Zeilen mit Check-Boxen zur Verfügung. Für jeden Anschluss können drei Haken gesetzt werden (Untereinander mit gleicher Bit/Port-Nummer.

   DDR(X): Hier wird definiert ob es sich um einen Eingang (Haken nicht gesetzt) oder einen Ausgang (Haken gesetzt) handelt.

   Pullups Port (X): Muss angehakt werden, wenn der interne Pullup-Widerstand des Mikrocontrollers aktiviert werden soll. Diese Option steht nur zur Verfügung, wenn der Anschluss als Eingang konfiguriert ist.

   L aktiv (I/O Negation): Normalerweise wird ein Eingang als 1 erkannt, wenn am Anschlusspin die Betriebsspannung anliegt. Wird der interne Pullup verwendet, liegt am Eingang normalerweise das Potential der Betriebsspannung an (wird als 1 interpretiert). Der Eingangssensor zieht das 1 Signal auf 0, wenn er aktiv ist. Daraus ergibt sich eine Negative Eingangs-Logik. Mit dem Schalter „L aktiv“ wird das Eingangssignal intern negiert. Wie das Eingangssignal lässt sich auch ein Ausgangssignal negieren (Gegen 0V kann der ATMega eine etwas höhere Last schalten).

   Den folgenden Dialog findet man im Menü „Datei“ unter „Port Konfiguration“:



   Der Port A.X des Mikrocontrollers entspricht dem Eingang E1.X bzw. dem Ausgang A1 (z.B. PA3 = E1.3). Port B (PB) wird dem Eingangsbyte EB2 zugeordnet (PC=EB3 bzw. AB3, PD=EB4 bzw. AB4). Die Portpins des Mikrocontrollers können einzeln als Ein- oder Ausgang konfiguriert werden (also z.B. PB0-PB3 sind Eingänge und PB4-PB7 sind Ausgänge). Die Bits des Eingangsbytes die als Ausgang konfiguriert sind, stehen auf 0 und können nicht genutzt werden. Auch die Bits eines Ausgangsbytes dürfen nicht genutzt werden, wenn dise als Eingang konfiguriert sind.

   Tastenentprellung: Beim Zyklusstart werden die Eingänge eingelesen, danach werden die AD-Wandlungen vorgenommen, danach werden die Eingänge ein zweites Mal eingelesen. Nur wenn bei beiden Lesevorgängen der Eingang gesetzt ist, wird auch das entsprechende Bit im Prozessabbild gesetzt.

4. Kommunikation PC-Microcontroller und weitere Einstellungen

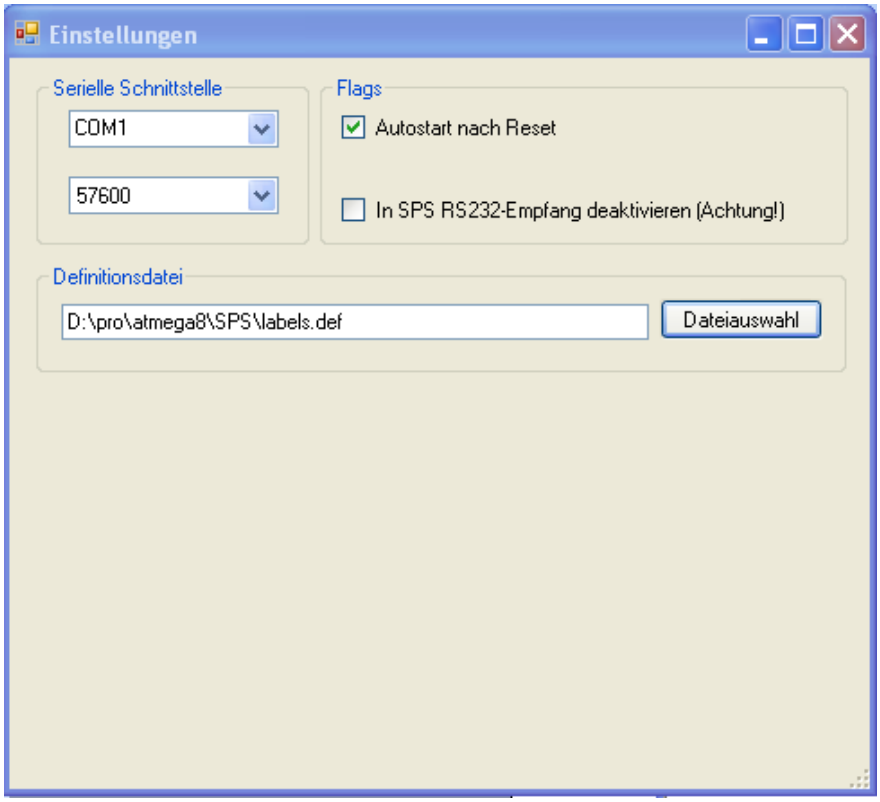
Der PC kommuniziert mit dem Mikrocontroller über die Serielle Schnittstelle (mit der Geschwindigkeit 57600bps).

Die Schnittstelle kann im Menü unter „Datei“ – „Einstellungen“ ausgewählt werden (COM1-COM16).

Die Kommunikation über USB-Seriell-Wandler sollte funktionieren (die Steuerleitungen werden nicht benutzt).

Die Mikrokontroller-Software lässt sich über die Serielle Schnittstelle steuern. Der PC schickt Steuer-Kommandos zum Mikrocontroller. Beim Empfang der meisten Zeichen, stoppt der Mikrocontroller mit der Ausführung des SPS-Programms (außer der RS232 Empfang ist deaktiviert)

Die Serielle Schnittstelle wir zur Programmierung benutzt (Sie kann (noch) nicht über das SPS-Programm angesprochen werden).



Mit dem Flag „Autostart nach Reset“, wird eingestellt, ob die SPS nach einem Reset auf Kommandos über die Serielle Schnittstelle wartet, oder mit der Abarbeitung des SPS-Programms beginnt.

Mit dem Flag „In SPS RS232-Empfang deaktivieren“ wird konfiguriert, ob der Mikrocontroller sich bei der SPS-Programmausführung durch Befehle über die Serielle Schnittstelle unterbrechen lässt.

Achtung: Wenn der Haken gesetzt wird, kann man nicht ohne weiteres wieder ein neues Programm zu Controller hochladen. Dann muss entweder das EEPROM wieder gelöscht werden oder beim Start wenn die Einschaltmeldung ausgegeben wird ein Zeichen zum Controller geschickt werden.

Diese Einstellungen werden erst aktiv, wenn das SPS-Programm zum Controller hochgeladen wird.

Im Bereich „Definitionsdatei“ kann eine Datei mit Festlegungen festgelegt werden, die vor der Programmübersetzung geladen wird. In dieser Datei dürfen nur Festlegungen gemacht werden z.B.:

```
PB0 = 2.0
```

E2.1 = LED_rot

Im SPS-Programm kann dann z.B.:

```
U     E :PB0
=     :LED_rot
```

geschrieben werden (Definitionen dürfen auch im SPS-Programm gemacht werden).

5. Befehle

1. Bitbefehle

U	E, A, M, Z, T	Und / And
UN	E, A, M, Z, T	Und Nicht / And Not
O	E, A, M, Z, T	Oder / Or
ON	E, A, M, Z, T	Oder Nicht / Or Not
X	E, A, M, Z, T	Exclusive Oder (XOR)
XN	E, A, M, Z, T	Exclusive Oder Nicht

Achtung: mehrere XOR-Befehle entsprechen nicht einem Gatter mit mehreren Eingängen, sondern mehreren hintereinandergeschalteten XOR-Gattern mit jeweils 2 Eingängen!

U(		Und Ausdruck in Klammer
O(		Oder Ausdruck in Klammer
)		Klammerausdruck abschließen
=	A, M	Zuweisen
S	A, M	Setze
R	A, M, Z, T	Rücksetze
NEG		Negiert das VKE

2. Lade/Transferbefehle

Alle Ladebefehle laden einen Wert in Accu1. Der alte Wert von Accu1 wird in Accu2 verschoben.

L	KB	Lade Konstantes Byte (Wertebereich 0...255)
L	KW	Lade Konstantes Byte (Wertebereich 0...65535)
L	KT X.Y	Lade Konstanten Zeitwert (Zählwert/Faktor X 1...255, Faktor Y 0=0,01s, 1=0,1s, 2=1s, 3=10s)

Achtung: Es sollte immer der kleinste mögliche Zeitfaktor verwendet werden, da die Zeiten mit Systemtakten gezählt werden, die nicht synchron zu den Timern sind.

L	MB, EB, AB	Lade Merkerbyte / Eingangsbyte / Ausgangsbyte
L	MB, EB, AB	Lade Merkerwort / Eingangswort / Ausgangswort (Accu H-Byte = Adresse, Accu L-Byte = Adresse +1)
T	MB, AB	Transfer Merkerbyte / Ausgangsbyte
T	MW, AW	Transfer Merkerwort / Ausgangswort (Adresse = Accu H-Byte, Adresse +1 = Accu L-Byte)

3. Befehle mit Accus

UW	Accu1 UND / AND Accu2 (bitweise)
OW	Accu1 ODER / OR Accu2 (bitweise)

XW		Accu1 EECLUSIVE ODER /XOR Accu2 (bitweise)
==I		Wenn Accu1 = Accu2 ist VKE=1, sonst ist VKE=0
><I		Wenn Accu1 ungleich Accu2 ist VKE=1, sonst ist VKE=0
<I		Wenn Accu1 < Accu2 dann ist VKE=1, sonst ist VKE=0
>I		Wenn Accu1 >Accu2 dann ist VKE=1, sonst ist VKE=0
+I		Accu1 + Accu2 (Accu2 bleibt unverändert)
-I		Accu1 – Accu2 (Accu2 bleibt unverändert)
SAR		Schiebe Accu1 um 1 Bit nach Rechts (:2)
SAL		Schiebe Accu1 um 1 Bit nach Links (*2)
SWB		Swap (Vertausch das H-Byte und das L-Byte vom Accu1)

4. Zählbefehle

ZV	Z	Zähle Vorwärts
ZR	Z	Zähle Rückwärts
R	Z	Rücksetze Zähler (auf Wert 0)
S	Z	Setze Zähler (mit Wert im Accu 1, wenn VKE=1)

5. Zeitbefehle

Achtung: Es sollte immer der kleinste mögliche Zeitfaktor verwendet werden, da die Zeiten mit Systemtakten gezählt werden, die nicht synchron zu den Timern sind.

SE	T	Einschaltverzögerung
SA	T	Ausschaltverzögerung
SS	T	Speichernde Einschaltverzögerung
SI	T	Impuls
SV	T	Verlängerter Impuls
R	T	Timer Zurücksetzen

6. Weitere Befehle

***		Netzwerkende (das aktuelle VKE wird für den nächsten Befehl nicht mehr verwendet).

Damit keine Schleifen entstehen können, sind Sprünge nur nach weiter hinten im Programm erlaubt.

:Label		Sprungmarke definieren (mit Doppelpunkt am Zeilenanfang)
SPA	Label	Springe Absolut – unabhängig von VKE (oder auch GOA).
SPB	Label	Springe Bedingt – Springe wenn VKE=1 (oder auch GOB).
HLT		SPS anhalten

6. SPS-Programm übersetzen und zum Mikrocontroller hochladen

Das SPS-Programm kann mit „SPS - Übersetzen“ umgewandelt werden. Nachdem das Programm erfolgreich übersetzt wurde, kommt ein Dialog, in dem gefragt wird, ob das Programm zum Mikrocontroller hochgeladen werden soll. Der Mikrocontroller muss mit programmiertem Flash-Speicher an der konfigurierten Schnittstelle angeschlossen sein.

Wenn das SPS-Programm noch fehlerhaft ist, wird in die erste fehlerhafte Zeile gesprungen. Im unteren Bereich vom MySPS werden auch die Fehler ausgegeben. Durch einen Doppelklich auf die Zeile mit der Fehlermeldung, wird der Cursor im SPS-Programm auf die entsprechende Zeile bewegt.

7. Tipps

1. Symbole

Am Anfang des Programms können Symbole definiert werden.

<Symbolname> = <Interner Name> z.B.: TasterStart = E 4.2

Im Programm können die Symbolischen Namen mit vorgestelltem Doppelpunkt verwendet werden.

Z.B. U TasterStart

2. Sondermerker

Impulse

Die Merker sind jeweils für einen Zyklus im jeweiligen Abstand gesetzt.

Impuls10s = M61.8

Impuls1s = M61.7

Impuls0_1s = M61.6

Impuls0_01s = M61.5

Takte

Die Merker blinken im jeweiligen Takt. Das Impuls/Pausenverhältnis ist 1:1.

Takt0_05Hz = M61.3

Takt0_5Hz = M61.2

Takt5Hz = M61.1

Erststart

Dieser Merker ist bei dem ersten Zyklus nach einem Neustart gesetzt.

Neustart = M62.1

Flankenauswertung

Für die MerkerBytes 45 und 46 steht eine Flankenauswertung auf Steigende Flanken bereit. Diese kann über die Merkerbytes 49 und 50 abgefragt werden.

Für die MerkerBytes 47 und 48 steht eine Flankenauswertung auf Steigende Flanken bereit. Diese kann über die Merkerbytes 51 und 52 abgefragt werden.

SF_M45.1 = M49.1	FF_M47.1 = M51.1
SF_M45.2 = M49.2	FF_M47.2 = M51.2
SF_M45.3 = M49.3	FF_M47.3 = M51.3
SF_M45.4 = M49.4	FF_M47.4 = M51.4
SF_M45.5 = M49.5	FF_M47.5 = M51.5
SF_M45.6 = M49.6	FF_M47.6 = M51.6
SF_M45.7 = M49.7	FF_M47.7 = M51.7
SF_M45.8 = M49.8	FF_M47.8 = M51.8
SF_M46.1 = M50.1	FF_M48.1 = M52.1
SF_M46.2 = M50.2	FF_M48.2 = M52.2
SF_M46.3 = M50.3	FF_M48.3 = M52.3
SF_M46.4 = M50.4	FF_M48.4 = M52.4

SF_M46.5 = M50.5 FF_M48.5 = M52.5

SF_M46.6 = M50.6 FF_M48.6 = M52.6

SF_M46.7 = M50.7 FF_M48.7 = M52.7

SF_M46.8 = M50.8 FF_M48.8 = M52.8

Z.B

```
U E 4.1
= M 45.1
***
U M 49.1          bzw. mit obiger Definition: U :SF_45.1
; VKE ist bei steigender Flanke = 1
```

3. Bei Inbetriebnahme prüfen

Bei Eingängen auf definierten Pegel achten (evtl. die internen Pullups verwenden).

Bei aktivierten Pullups ist wahrscheinlich auch eine Negation des Eingangs sinnvoll.

Ausgänge nicht als Eingang (mit Taster) verdrahten (Zerstörungsgefahr das Ausgangs) – also „Port Konfiguration“ überprüfen.

8. Weiteres

Einschränkungen / bekannte Probleme

- Das Übersetzungstool prüft evtl. nicht alle Bereiche für Eingänge/Ausgänge/Merker usw.

- Beim runterfahren des PCs oder abstecken der RS232 Verbindung kommt es vor, dass die SPS stehen bleibt. Scheinbar erkennt der Controller ein Zeichen an seiner Schnittstelle. Gegebenenfalls einen Reset am Controller machen oder den SPS-RS232 Empfang deaktivieren.

Ideen für weitere Versionen

Falls ich Zeit für die Weiterentwicklung finde, könnte ich mir die folgenden Features vorstellen:

- Unterstützung ATMega644, da dieser 2048 Byte EEPROM für das SPS-Programm hat.

- Ablage des SPS-Programms im Flash Speicher.

- Einbeziehen der Seriellen Schnittstelle in das SPS-Programm.

- Modularisierung (Organisationsbausteine/Funktionsbausteine)

- Laufzeit der SPS (Minuten/Stunden/Tage)

- Erweiterung der Ein/Ausgänge über SPI mit 74HC/HCT595 bzw. 75HC/HCT165. (4 Eingangs- und 4 Ausgangsbytes sind im PA noch ungenutzt).

- Keine feste Programmzykluszeit, sondern Überwachung ob maximale Zykluszeit überschritten wird.

- Merkerbereich der von dem angeschlossenem PC verändert werden kann bzw. Merkerbereich der dem PC übergeben wird.