Falcon BMS to Arduino Interface Tool

(BMSAIT)

Beispielprogramm CMDS

(Countermeasure Dispenser System)

Autor	Robin „Hummer“ Bruns
Dokumentversion	0.1
Softwareversion	1.0
BMS Version	4.34u3
Datum	11.10.2020

1. Überblick

In den bisherigen Beispielen wurden immer nur einzelne Funktionen beispielhaft abgebildet. Hier wird nun darauf aufgebaut, indem mehrere Einzelbestandteile zu einem komplexen Projekt zusammengesetzt werden. Das Modul CMDS gibt es in mehreren Varianten, die sich durch leicht unterschiedliche Hardwarenutzungen unterscheiden.

Die für alle Varianten gleichen Grundfunktion bestehen darin, dass die Funktionen des CMDS abgebildet werden sollen. Dazu gehört:

- Anzeige des verfügbaren Chaff und Flares
- Ansteuerung von Status LEDs
- Ansteuerung der Switches des CMDS panel

Die Besonderheit besteht hier darin, dass die Anzeige der Displays gesteuert werden kann:

- Ist die Stromversorgung des Flugzeugs abgeschaltet, sind die Displays abgeschaltet
- Ist das CMDS abgeschaltet, sind die Displays abgeschaltet
- Sind die Schalter für die Aktivierung der Chaff/Flare nicht aktiviert, werden die jeweiligen Displays abgeschaltet

Variante 1 der Abbildung des CMDS:

Variante 1 stellt die Variante unter Nutzung eines einzelnen Max7219 Display Tube dar, um darauf sowohl die Anzahl der Chaff als auch der Flare anzuzeigen. Die Chaff werden auf den ersten 3 der 8 Elemente angezeigt, die Flare auf den letzten 3.

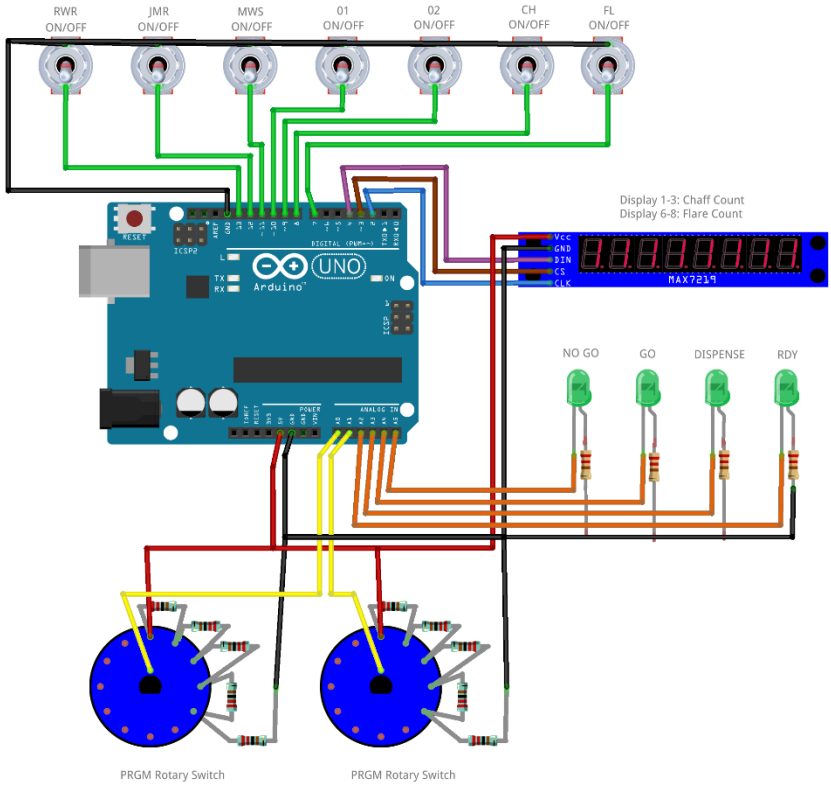
Wer möchte, kann die Displays von der Platine des MAX7219 entfernen und dann getrennt im Panel verbauen. Dies verursacht etwas Lötarbeit und zugegebenermaßen etwas Kabelgewirr, aber man kommt mit einem Display aus und dann die Anzeigen unabhängig voneinander platzieren – oder sogar eigene Anzeigeelemente in anderen Größen oder Farben verwenden.

Benötigte Hardware:

- Ein Arduino-Board (z.B. ein UNO)
- Ein Max7212 7-Segment-Tube
- Vier LED (2.3V, Grün)
- Vier Widerstände (220 Ohm)
- Einen Drehschalter mit sechs Rastungen
- Einen Drehschalter mit vier oder fünf Rastungen
- 10 Widerstände 2.2 kOhm
- Zwei Kippschalter (ON-OFF) (unbedingt erforderlich für Chaff und Flare)
- Fünf Kippschalter (ON-OFF) (optional)
- Verbindungskabel

2. Verkabelung

2.1. Variante 1



Arduino	MAX7219	LED	Gegenseite
2	CLK		
3	CS		
4	DIN		
A2		RDY	GND
A3		DISP	GND
A4		GO	GND
A5		NOGO	GND

Schalter	Arduino PIN	Gegenseite
MODE	A0	GND
PRGM	A1	GND
JETT	6	GND
FL	7	GND
CH	8	GND
O2	9	GND
O1	10	GND
MWS	11	GND
JMR	12	GND
RWR	13	GND

Bei den Drehschaltern werden die Stellungen über die Software bestimmt. Hierzu sind die Rastungen des Drehschalters mit Widerständen zu verbinden (siehe BMSAIT Doku in der Erklärung des Moduls Switches). Die Summe aller Widerstände bei einem Drehschalter sollte ca. 10 kOhm betragen (also hier jeweils 2.2 kOhm Widerstände verwenden).

2.2. Interne Kommandoverarbeitung

Im Modul CMDS ist es erforderlich, dass die Schalter MODE, CHAFF und FLARE an dem Arduino angeschlossen sind, auf dem dieses Modul läuft. Der Grund dafür ist, dass die Schalterstellungen benutzt werden, um Funktionen dieses Moduls zu beeinflussen.

A/C PWR	MODE	CHAFF	FLARE	CHAFF Anzeige	FLARE Anzeige
OFF	Egal	Egal	Egal	Keine Anzeige	Keine Anzeige
ON	OFF	Egal	Egal	Keine Anzeige	Keine Anzeige
ON	Alles außer OFF	ON	Egal	Anzeige Anzahl	
ON	Alles außer OFF	OFF	Egal	Keine Anzeige	
ON	Alles außer OFF	Egal	ON		Anzeige Anzahl
ON	Alles außer OFF	Egal	OFF		Keine Anzeige

Die A/C Power wird durch eine BMSAIT Variable direkt aus der Simulation gelesen. Dieser Schalter muss daher nicht an diesem Arduino angeschlossen sein. Für die Funktion des Moduls muss dafür aber die entsprechende Datenvariable im Bereich UserConfig aufgenommen sein (ID 1260 „PWRST“).

Die Funktionsweise besteht darin, dass den angeschlossenen Schaltern im Modul Switches ein interner Code hinzugefügt wird, über den das Modul CMDS die aktuelle Schalterposition erkennen kann.

```
Schalter schalter[]=
{
//Switch definition. If you add a switch, add a line to the following list
//,{<PIN>,<description>,<type>,<rotarySwitchID>,0,<commandID when pressed>,<commandID when released>,<in        command>}
  {   A0,   "MODE"  ,    3,          0,        0,        "00",                "00",                1}   //MODE Rotary
 ,{   A1,   "PGRM"  ,    3,          1,        0,        "00",                "00",                0}   //PGRM Rotary
 ,{    6,   "JETT"  ,    2,          0,        0,        "01",                "02",                0}   //Jettison ON/OFF Switch
 ,{    7,   "FLARE" ,    2,          0,        0,        "03",                "04",                2}   //Flare ON/OFF Switch
 ,{    8,   "CHAFF" ,    2,          0,        0,        "05",                "06",                3}   //Chaff ON/OFF Switch
 ,{    9,   "02"    ,    2,          0,        0,        "07",                "08",                0}   //02  ON/OFF Switch
 ,{   10,   "01"    ,    2,          0,        0,        "09",                "10",                0}   //01  ON/OFF Switch
 ,{   11,   "MWS"   ,    2,          0,        0,        "11",                "12",                0}   //MWS ON/OFF Switch
 ,{   12,   "JMR"   ,    2,          0,        0,        "13",                "14",                0}   //JMR ON/OFF Switch
 ,{   13,   "RWR"   ,    2,          0,        0,        "15",                "16",                0}   //RWR ON/OFF Switch
};
```

Ein Coding wertet die Stellungen aus und bestimmt darüber, ob die Displays aktiviert sind und welcher Wert angezeigt werden soll (Funktion *checkPowerOn*).

3. Programmierung des Arduino

Falls die Arduino IDE noch nicht installiert ist, lest bitte das Kapitel 4.1.4 der BMSAIT Dokumentation.

Ruft nun die .ino in der gewünschten Variante aus dem Ordner \Arduino Sketch\BMSAIT_CMDS\ mit einem Doppelklick auf. Das Sketch wird in der Arduino IDE geladen. Wenn ihr die Verkabelung gem. Kapitel 2 durchgeführt habt, sind hier keine Anpassungen erforderlich.
Nachdem ihr das richtige Arduino-Board ausgewählt habt, ladet ihr das Sketch auf den Arduino hoch.

4. Einstellung des Windows-Programms

Installiert und startet BMSAIT und stellt sicher, dass die Basiseinstellungen richtig vorgenommen wurden. Wichtig ist insbesondere, dass der Verweis auf die Variablendefinition (BMAIT-Variablen.csv) hergestellt wird. Wählt das PUSH-Prinzip und schaltet den Autostart aus.

Ladet anschließend die beiliegende Konfiguration (BMSAIT_demoCMDS.ini). BMSAIT sollte nun die geladene Definition anzeigen (ein COM-Port und mehrere Variablen).

Macht einen Rechtsklick auf den COM-Port und bearbeitet diesen. Wählt den COM-Port aus, an dem euer Arduino angeschlossen ist. Wenn ihr nicht sicher seid, welcher COM-Port dies ist, dann wählt entweder die SCAN Funktion und schaut, auf welchem COM-Port der Arduino eine Antwort sendet oder ihr schaut in dem Windows-Gerätemanager nach.

Ich empfehle die Änderungen nun zu sichern („Speichern unter“ und Auswahl einer neuen Datei).

Aktiviert den Testmodus und startet die Verarbeitung. Wenn alles geklappt hat, sollte das/sollten die 7-Segment-Anzeigen kurz einen Wert anzeigen.

5. Ergebnis

Beendet den Testmodus und startet die Verarbeitung des BMSAIT. Startet FalconBMS und betretet die 3D Welt. Prüft nun die Schalterstellungen und das Ergebnis der Displayanzeige gem. Tabelle aus Kapitel 2.2.

Die Prüfung, ob die Stromversorgung im Cockpit auch aktiviert ist, erfolgt erst in der 3D-Welt. Außerhalb der 3D Welt wird nur auf die Stellung der CMDS Schalter geprüft, um zu bestimmen, ob die Anzeigen aktiviert sind oder nicht.