ÄNDERUNGEN (07.02.2010)

Vereinfacht wurde das Speichern der eingehenden Daten:

Jedes empfangene Byte wird direkt und ohne Umwege und Zwischenspeicherung aus der ISR heraus in den Buffer des Dataflash geschrieben.
Dadurch werden einerseits 512 Byte SRAM eingespart, andererseits funktioniert dieses Verfahren bei jeder Puffergröße des DataFlash.

Auch der Lesepuffer ist nun flexibel einstellbar in seiner Größe. Das Ziel ist auch hier die Unabhängigkeit des Programmcodes von den Puffergrößen des DataFlash - und ggf. die Einsparung von SRAM.

Es werden nur noch ca. 110 Byte SRAM (zuzüglich Stack) benötigt, das Programm erfordert unter 4kB Flash.

Somit lassen sich auch kleiner Controller wie der ATMEG 48 bzw. auch der TINY 44 einsetzen (allerdings ist bei letzterem noch etwas Anpassung erforderlich, da weder Hardware-TWI noch USART vorhanden sind - das ist aber kein grundsätzliches Problem).

Wenn mit dieser Programmversion Daten ausgelesen werden, die mit der vorherigen Version geschrieben wurden, dann fehlt jeweils am Ende einer Page genau 1 Byte.
Das hängt damit zusammen, dass nun die Anzahl der Bytes einer Page mit 16 Bit gespeichert wird - während bisher nur 8 Bit verwendet wurden, somit 0 genau 1 Byte meinte (und 255 = 256 Byte).
Jetzt die Zahl identisch mit der Anzahl der tatsächlich vorhandenen Bytes.

Anzumerken ist, dass die gewählte Form der Datenspeicherung ein Problem birgt:
Kommen die neuen Daten schneller als sie abgelegt werden können, dann droht Ungemach.
Die Serielle Übertragung mit 9600 Baud ist problemlos. Hier treffen maximal 1 Byte / ms ein.
Anders sieht es bei TWI mit 100kHz aus, hier werden ca. alle 100us ein Byte empfangen.

Bei ca. 4 MHz F_CPU hat der Controller also ca 100 * 4 = 400 Takte Zeit, das Byte zu speichern.
Das war in meinen Tests ausreichend.
Sollte der TWI-Bus mit 400kHz betrieben werden, dann wird das wohl nicht mehr funktionieren.

Im Zweifelsfalle kann man jedoch F_CPU erhöhen.

Um den Logger auf andere DataFlash's anzupassen, sind folgende Konstanten in der 'at45dbx81.h' zu ändern:

PAGES_ON_CHIP	4096 / 8192
PAGE_SIZE	256 / 512
STATUS_READY	1<<128 +
DF_make_adr()	hier muss die Berechnung der Adresse aus Page und Byteadresse angepasst werden

Danach sollten auch größere Chips ansprechbar sein. Leider konnte ich das bislang nicht testen.
Das Programm in der jetzigen Form arbeitet zusammen mit dem AT45DB081.

Viel Spaß beim Loggen.

Michael S.

ÄNDERUNGEN (01.02.2010)

In der Funktionalität ist das Programm im Grundsatz unverändert, hinzugekommen ist die Option, während (mit Betonung auf WÄHREND) des Loggens die LED's auszuschalten durch Betätigen der Taste PLAY.
Auf eine Datei folgend werden einige Daten dazu ausgeben: Start-/Endpage, belegte Pages, Anzahl der Bytes und Anzahl der Sekunden seit dem Anlegen der Datei.

Ansonsten ist der Code nun so strukturiert, dass mit wenigen Anpassungen auch Dataflashes mit größer Pagesize beschrieben werden können.
Dazu musste die Datei/Headerstruktur geändert werden, damit in den 8 Byte effektiver Informationen abgelegt werden können:
- eine eindeutige File_ID (16 Bit), die Anzahl der Bytes pro Page(in 12 Bit), ein Zähler für Anzahl der Pages der Datei (16 Bit) und die Zeit seit dem Anlegen der Datei in Sekunden (in 20 Bit).

Das Ende des benutzten Speicherbereiches wird nun anhand der File_ID erkannt (nicht mehr an einer gelöschten Page).
Die Daten werden mit einem "pagewrite with pageerase" geschrieben, eine Pageerase ist somit nur noch dann erforderlich, wenn auch wirklich Daten geschrieben werden.
Ein "page erase and programm" benötigt laut Datenblatt 14 bis35ms. Wenn man dem Controller noch Zeit für ein paar Takte Arbeit einräumt, dann kann man mindestens 256Byte * 20 = 5 Kb / Sekunde loggen, im günstigen Falle auch das Doppelte.
Sollen die Daten noch schneller abgelegt werden (etwa für Diktiergerät), dann muss man die Pages vorab löschen, so dass anstelle der "page erase and programm" nur noch ein "page programm" erforderlich ist (Dauer: 2-4ms).
Das ist dann um den Faktor 10 schneller.
Aber zum Loggen von Messdaten wäre das wie mit Kanonen auf Spatzen zu schießen.

Und noch einmal der Hinweis:

Wenn grundsätzlich eine der beiden für die Datenaquise möglichen Schnittstellen nicht benutzt wird, dann sollte man diese vor dem Kompilieren auskommentieren.
Im anderen Falle ist es wichtig, dass nicht beschaltete Eingangspins auf ein definiertes Spannungsniveau gezogen werden (sinnvollerweise auf VCC). Driftende Pins können zu lustigen Ergebnissen führen.

Datenlogger mit Atmel AT45DB081D (10.01.2010)

Vor ungefähr einem Monat hatte ich einen Datenlogger auf der Basis des VDIP1 mit USB-Stick als Datenträger ins Netz gestellt. Der Vorteil dieses Loggers ist seine schier unbegrenzte Speicherfähigkeit auf dem USB-Stick (die ich nun leider gar nicht ausnutzen kann).
Nachteilig sind die hohen Kosten für den VDIP1/Stick und der hohe Stromverbrauch des Gesamtsystems.

Daher ist ein weiterer Logger auf der Basis eines serial-interfaced Flash Memory (Atmel DataFlash) entstanden
Dieser kann zwar nur 1MB speichern, aber das reicht mir aus (abgesehen davon gibt es ja auch noch "größere" Chips).
Die Vorteile dieser Variante:
preisgüstig, kompaktes Gehäuse realisierbar, schnell, stromsparend (auch im Batteriebetrieb einsetzbar),
kein kompletter Dateiverlust bei Stromausfall oder versehentlichem Abschalten (beim VDIP1 muss eine Datei immer ordentlich geschlossen werden).
Einziger kleiner Haken: der Speicher wird als smd-Chip geliefert.

Im beigefügten Programm wird ein ATMEL AT45DB081D mit 1 MB Speicher an einem ATMEGA168 betrieben. Der Speicherbedarf des Programms liegt unter 4kB für die "passive" Variante, etwas ca. 5 kB für die "aktive" Variante.
Wo liegt der Unterschied:
In der "passiven" Variante nimmt der Logger Daten via TWI oder RS232 entgegen.
Passiv deswegen, weil er eben abwartet, bis ihm irgendjemand irgendwelche Bytes zuschiebt.
Diese Variante ist nur für einen einzelnen Sensor sinnvoll (z.B. eine GPS-Maus).
Um mehrere Sensoren abzufragen, ist die "aktive" Variante entstanden.
Hier aggiert der Logger als TWI-Master und fragt in einstellbaren Zeitabständen seine Slaves ab.
Im Beispielcode werden die Temperaturdaten von vier Dallas DS1631 ausgelesen.

Die Ausgabe der aufgenommenen Daten erfolgt bei beiden Fällen über die serielle Schnittstelle.

Die Bedienung
Es werden zwei Taster und drei LEDs zur Bedienung und Kontrolle verwendet.
- Taste 1 startet und beendet das Loggen.
- Taste 2 startet die Wiedergabe der zuletzt aufgenommenen Log-Datei (danach gibt die Taste 2 solange die jeweils vorausgehende Datei aus wie Dateien gefunden werden), Taste 1 beendet die Wiedergabe.
Folgende Hilfestellungen geben die drei Leds:

oxo	Eine blinkende LED2 nach dem Programmstart zeigt an, dass der DF nicht geantwortet hat
xxo	Blinkende LED1/2 nach dem Programmstart zeigen an, dass die Dateistruktur geprüft wird (das dauert nur eine knappe Sekunde).
oXo	Danach zeigt eine dauerleuchtende LED2 die Bereitschaft an (Hauptmenue). Weiter geht es entweder mit Taste 1 oder mit Taste 2. Durch Betätigen der Taste 1 wird die Aufnahme/das Loggen gestartet:
Xxx	Die LED1 ist dabei eingeschaltet, die LED2 blinkt gleichmäßig im Sekundentakt, die LED3 toggled, wenn ein Datensatz geschrieben wurde (an der Aktivität von LED3 kann erkannt werden, dass der Logger tatsächlich loggt!). Die Aufnahme wird beendet mit Taste 1, es wird das Hauptmenue angezeigt.
oXo	Die Led2 zeigt wieder Dauerlicht (Hauptmenue). Die Taste 2 startet die Wiedergabe der zuletzt aufgenommenen Logdatei:
oxX	Die LED3 zeigt Dauerlicht, die Led2 toggled, wenn 256 Byte geschrieben worden sind (dieser Vorgang kann bei größeren Dateien einige Sekunden dauern, bei 9600 Baud wird ca. 1kB/sec geschrieben).
oxX	- wenn statt dessen kurzzeitig LED2 hektisch blinkt, dann ist keine ältere Datei mehr gefunden worden.
ooX	Ist die Datei herausgeschrieben, dann leuchtet nur noch LED3. Wird noch einmal Taste2 betätigt, dann wird die nächst ältere Datei ausgelesen und ausgegeben. Taste 1 bricht die Wiedergabe ab und springt zurück ins Hauptmenue:
oXo	Im Hauptmenue leuchtet nur die LED2. Und noch eine weitere Option:
xxx	Taste 1 und Taste 2 für ca. 3 Sekunden gedrückt halten und dann loslassen - der Speicherchip wird gelöscht. Die LEDs blinken während des Löschens.

Es gibt noch einige weitere Zustände (Fehler), die durch blinkende LED angezeigt werden.
Dazu bitte bei Problemen in den Programmcode schauen.
oxx LED2/3 toggled eine Zeitlang schnell (im "aktiven" Modus kein TWI-Slave gefunden)

(Legende: o = AUS, x = blinkend, X = EIN)

Die Arbeitsweise
Der DataFlash wird im 264-Byte Modus betrieben. 256 Byte werden für Nutzdaten verwendet, 8 Byte für Verwaltungsinformationen. Die Verwaltungsinformationen sind :
Byte[0] = 0 (zeigt, dass die Page benutzt wird), 255 für gelöscht/frei.
Byte[1] = Anzahl der Nutzbytes in der Page (in der Regel 255, aber am Ende der Datei sind es oft weniger !)
Byte[2/3] = ID der Datei (=Dateiname), gleich in allen Pages einer Datei
Byte[4/5] = durchlaufende Zählung der Pages der Datei, beginnend mit 1
Byte[6/7] = Zeiteintrag beim Schreiben der Page, Anzahl der Sekunden seit dem Anlegen der Datei.
Die Zeitinformation wird im Moment nicht ausgewertet.
DateiId und Pagenummer werden zur Wiedergabe einer Logdatei verwendet.

Die Daten werden ab dem Speicheranfang beginnend seriell hintereinander weg in den Speicher geschrieben.
Ist der Speicher voll, dann läuft der Index auf Page 0 über und es werden die alten Daten am Speicheranfang überschrieben.
Will man das nicht, kann man in der global.h NO_OVERWRITE / NO_OVERRUN aktivieren.
Dann werden innerhalb einer Sitzung keine Sitzungsdaten überschrieben (aber nach 4096 Pages ist Schicht im Schacht) bzw. es wird am Ende des Speichers der Überlauf auf Page0 unterbunden.

Der Logger verwendet zwei getrennte Arrays zur Aufnahme der Daten. Wenn das erste Array voll ist, werden die Daten im anderen Array gepuffert, während dessen die Daten des erste Arrays in einen Puffer des DF übertragen werden. Jeweils nach dem Schreiben einer Page wird die nächst folgende Page gelöscht.

Beim Start durchsucht das Programm den Speicher nach der ersten freien Page (erkennbar am Wert 255 in Byte[256].
Beim Durchscrollen durch die Pages merkt sich das Programm die höchste bisher vergebene Dateinummer und incrementiert diese für die nächste neue Datei.
Damit sollen alle zusammengehörigen Pages einer Datei durch eine gemeinsame und eindeutige ID markiert werden.

Schaltplan
Als Microcontroller wird ein ATMEGA48 ..168 verwendet.
Ein Quarz ist notwendig, um die serielle Schnittstelle benutzen zu können. Die Beschaltung dafür dürfte klar sein.

Die Verbindungen zwischen ATMEGA und Speicherchip ist trivial:

ATMEGA			AT45DB081D		
SCK	PB5	->	SCK	Pin2	(RESET Pin3 und WP Pin4 sind mit VCC verbunden)
MISO	PB4	<-	SO	Pin8	
MOSI	PB3	->	SI	Pin1	
CS	PB2	->	CS	Pin4	

LED1	PD2		geschaltet nach gnd, jeweils über Vorwiderstand 1.5k(5V) bzw. 750 (3.3V)
LED2	PD3		
LED3	PD4		
Taster 1	PD6		parallel zum Taster ca. 1mF nach gnd, zwischen Taster und Pin jeweils ca. 10k
Taster 2	PD7		parallel zum Taster ca. 1mF nach gnd, zwischen Taster und Pin jeweils ca. 10k
RxD	PD0	<-	Serielle Schnittstelle - Eingang
TxD	PD1	->	Serielle Schnittstelle - Ausgang
SCK	PC5		TWI-Schnittstelle, Clock (ggf. Pullup nach VCC)
SDA	PC4		TWI-Schnittstelle, Daten (ggf. Pullup nach VCC)

Zu beachten ist, dass der DataFlash zwingend mit weniger als 3.6 Volt versorgt werden muss.
Allerdings sind die Eingangspins lt. Datenblatt 5V tolerant (meiner lebt auch noch).
Was gut funktionierte: Speicher und ATMEGA gemeinsam mit 3.3V betrieben, die serielle Schnittstelle über einen MAX232 (der mit 5Volt versorgt wird) auf "PC-Niveau" bringen.
Nach anfänglichen Experimenten arbeite ich auf der 5V-Schiene:
Der Dataflash wird über 3 Silizium-Dioden in Durchlassrichtung mit VCC verbunden, so dass an den Dioden ca. 3X 0.7 Volt = 2.1 Volt abfallen sollten (sind aber praktisch nur 1.5 V).
Als Ruhestromaufnahme (ohne LEDs und ohne weitere SleepModes) habe ich mit unter 7mA gemessen.
Alternativ habe ich auch einen LM317 als Spannungsregler verwendet und auf 2.9Volt runtergeregelt- nur der LM317 zieht halt 5mA für sich - für den Batteriebetrieb ist diese Lösung also nicht angesagt.
Die Kommunikation mit dem ATMEGA am 5 Volt-Stang funktionierte trotz des Spannungsunterschiedes (auch bei 2.9 Volt) (siehe auch Atmel Application Note "Interfacing the 3-volt DataFlash with a 5-volt system").

Den Speicherchip habe ich auf eine Adapterplatine gelötet: richtig ausrichten, mit der Kreuzpinzette fixieren und dann vorsichtig die Beinchen löten - es war deutlich leichter, als ich befürchtet hatte.

Um die Daten aus dem Speicher auszulesen habe ich HTerm bemüht:
Zuerst den Empfangsspeicher löschen ("Clear received"), dann die Daten vom Logger abschicken und abschließend mit "Save output" als Datei speichern.
HTerm klappert zwar mächtig mit der DSR "Led" - und auch der Scrollbar zuckt heftig. Aber die Daten kommen an.

Bei der Gelegenheit ausdrücklichen Dank an den Autor von HTerm.
Das Programm ist wirklich sehr nützlich. Auch wenn es manchmal (aber wirklich nur selten) zickt.
Aber das tun Geldkarten ja auch.

Viel Spaß beim Loggen,

Michael S.