Datenlogger für USB-Stick mit VNC1L(VDIP1) / mega88 in c

Was macht der Logger ...

Der Datenlogger nimmt Daten via RS232, TWI oder SPI (jeweils Interrupt-gesteuert) entgegen und legt sie in einem Ringpuffer von 768 Byte ab.
Ist ein Packet von 512 Byte komplett, dann wird es via Software-SPI auf den VDIP geschrieben.
512 Byte sind nach meiner Kenntnis auch die 'natürliche' Sektorgröße des VNC1L.

Hardware

Als Controller wird ein atmega88 verwendet (die Programmgröße beträgt ca. 4 kB), jeder andere uc mit ausreichend Pins und den benötigten Schnittstellen in der Hardware ist ebenfalls verwendbar.
Da der Empfang der Daten Interrupt-gesteuert erfolgt, müssen die (benötigten) Schnittstellen auf dem Chip vorhanden sein.
Der USB-Stick wird mit einen VNC1L-Chip (gibt es z.B. auf einem -> VDIP1) über Software-SPI angesprochen .
Einen Schaltplan kann ich nicht bieten, daher hier kurz die textliche Beschreibung.
Beim ATmegax8 sind folgende Pins zu belegen (sofern die Schnittstellen genutzt werden sollen):

TWI	SCL	Pin28	PC5	Pullup nicht vergessen
	SDA	Pin27	PC4	Pullup nicht vergessen
RS232	TxD	Pin3	PD1	(zum Debuggen nützlich)
	RxD	Pin2	PD0	(zum Empfangen von Daten)
SPI	SCK	Pin19	PB5	(Hardware-SPI)
	MISO	Pin18	PB4	
	MOSI	Pin17	PB3	
	SS	Pin16	PB2	

Bei Verwendung der Seriellen Datenübertragung via UART ist ein Quarz notwendig (hier 3.686400 MHz).
Werden Daten nur via TWI/SPI übertragen, dann kann dieser natürlich entfallen, wenn an die Genauigkeit der eingebauten ‘Uhr’ keine Anforderungen gestellt werden (aber die Anpassung des Timers an die gewählt F_CPU ist sinnvoll).
Ein Taster zur Bedienung ist erforderlich. Im Falle der Betätigung zieht er einen Pin auf Masse (z.B. PD2/Pin4)
Hilfreich ist ein Kondensator von ca. 0.33uF zum Entprellen (zwischen PD2 und GND), wobei im Programm bereits notwendige Warteschleifen zum Entprellen vorgesehen sind.
Die kursierenden berühmten Entprellroutinen wollte ich mir ersparen, lieber spendiere ich für 0,09 Eur einen Kondensator.
Ein Jumper von PD.3 auf gnd (sofern gesetzt) ermöglicht, die im Programmcode verankerte Systemzeit zu verändern (indem während des Wartens auf den 1. Start des Loggers 6 Byte mit Zeitdaten an den Logger versandt werden).
Eine low-current LED, die den Programmstatus signalisiert, ist an Pin14 (PB0) angeschlossen und über 1.5k mit VCC verbunden.

Für die Software-SPI Schnittstelle zum VDIP1 werden folgende Verbindungen hergestellt (siehe spi_13bit.h):

ATMEGAx8			VDIP	
CS	Pin23	PC0	Pin10	AD3
SDO	Pin24	PC1	Pin9	AD2
SDI	Pin25	PC2	Pin8	AD1
SCK	Pin26	PC3	Pin6	AD0
RST	Pin11	PD5	Pin22	RST

Der ATMEGA und der VDIP sind direkt (ohne level-shifter) verbunden, da der VDIP1 5V verträglich ist (sein soll).
Meiner lebt zumindest noch.

Zur Kontrolle des Programmablaufes sollten die beiden LEDs auf dem VDIP sichtbar sein, da sie Hinweise auf das Befinden des VDIP geben.
Zusätzlich gibt die Status-LED Auskunft über den Status des Programmablaufs.

Bedienung des Loggers und Rückmeldung der Status-LED

1.) Nach den Einschalten und etwa 4 Sekunden Wartezeit zur Initialisierung;
die Einschaltmeldung der VDIP wird via serieller Schnittstelle übertragen.

1.b) Wenn kein Stick gefunden wird, dann folgt Blinklicht (1Hz),
der Stick kann jetzt noch eingesteckt werden.
Wenn der Stick eingesetzt wird, dann gehts zu 2.) weiter.

2.) Der Stick ist gefunden, die Status-LED zeigt Dauerlicht.
Der Stick kann in dieser Phase abgezogen und wieder angesteckt werden, das Programm prüft den Stick ständig auf seine Präsenz.
Das Programm wartet auf die Betätigung der Starttaste, um mit dem Loggen zu beginnen.
Während dieser Phase (aber nur beim 1. Warten nach einem Neustart auf die Starttaste) kann die Systemzeit geändert werden, sofern beim Programmstart der PIN.PD3 auf gnd gezogen war (dann wird der Empfang von Daten bereits zu diesem Zeitpunkt aktiviert).
6 Bytes (Sekunde .. Jahr) müssen gesendet werden. Werden die Daten übernommen, dann blinkt die Status_LED 6x, fallen sie durch die Plausibilitätsprüfung, dann wird 1x geblinkt.
Wird garnicht geblinkt, dann war wohl PD.3 beim Start nicht auf low.

2.b) Wenn der Stick abgezogen wird, dann blinkt die LED schnell (5Hz).
Sobald der Stick wieder gefunden ist, springt das Programm wieder zu Punkt 2.).

3.) Nach dem Starten beginnt das Loggen, die LED blinkt ganz gemächlich mit 0.5 Hz, aber die LED am VDIP sollte aufblinken, wenn ein Datensatz geschrieben wird.
In dieseer Phase darf der Stick auf keinen Fall abgezogen werden.
Zum Beenden des Loggens muss erneut die Taste gedrückt werden, damit die Datei geschlossen wird.

4.) Die LED blinkt langsam mit 1Hz, die Log-Datei ist geschlossen.
Nun gibt es zwei Möglichkeiten
a.) Der Stick kann entnommen werden, die LED blinkt dann schnell (5Hz) und wartet, bis wieder ein Stick gefunden wird. Danach wird eine neue Logdatei vorbereitet und mit Dauerlicht der LED auf die Starttaste gewartet (das ist Phase 2.).
b.) Durch Betätigen der Taste ohne Entnahme des Sticks wird direkt ein neuer Logvorgang vorbereitet (es wird eine neue Logdatei gewählt), die LED geht auf Dauerlicht und wartet auf die Starttaste (das ist Phase 2).

5.) Die LED zeigt Dauerlicht. (diese Phase ist identisch mit 2.)
Zu diesem Zeitpunkt sind alle Dateien geschlossen, der Stick darf entfernt werden.
Oder eine neue Logdatei beschrieben werden mit der Starttaste (-> 3.)

Bei einem Fehler ist die Status-LED ca. 1 Sekunde ausgeschaltet und blinkt dann X-Mal.
X ist der Fehlercode. Im Quellcode kann nachgesehen, an welcher Stelle error_msg(X) aufgerufen wurde.
Damit lässt sich die Ursache des Fehlers eingrenzen.

Die Anzeige der Status-LED:
- Bei Dauerlicht wartet der Logger auf einen Tastendruck um Loszulegen (der Stick darf abgezogen werden).
- Bei ganz langsam blinkender LED läuft der Logvorgang - wer jetzt den Stick abzieht, hat die Daten verloren !! Also unbedingt mit einem Tastendruck den Logvorgang beenden.
- Bei langsamem Blinken kann der Stick entnommen werden, mit einem Tastendruck wird neuer Logvorgang vorbereitet.
- Bei schnell blinkender LED ist kein Stick gefunden, wenn ein Stick 'eingestickt' wird, dann sollte die LED auf Dauerlicht wechseln (also Bereitschaft zum Loggen anzeigen).

Dateinamen

Dateien werden über eine Zahl (uint8) benannt.
Diese Zahl wird verwendet, um im default-Dateinamen ("file_000.txt") die Ziffernfolge "000" zu ersetzen.
Die Funktion VNC_make_name(uint8_t counter) ersetzt die Ziffernfolge ’000’ durch den dreistelligen String, der aus dem Wert von counter gebildet wird. Es können also Dateinamen von ‘file_000.txt’ bis ‘file_255.txt’ entstehen.
Wenn counter > 254, dann wird allerdings mit einem Fehler abgebrochen.
Während der Vorbereitung für das Loggen prüft das Programm, beginnend mit ‘file_000.txt’, ob die Datei existiert.
Der Counter wird solange incrementiert, bis ein freier (nicht vorhandener) Dateiname gefunden wird - oder ein Abbruch erfolgt.
Diese Suche verzögert den Einschaltvorgang, daher sollten überflüssige Dateien vom Stick entfernt oder umbenannt werden.

Dateizeit

Im Programmcode wird ein Zeit festgelegt, die beim Start des Programmes übernommen wird.
In dem Zeitraum, in dem der Logger nach dem Starten mit Dauerlicht-LED auf den Start des Loggens wartet, kann man von aussen her genau 6 Byte senden, die eine neue Uhrzeit festlegen (Reihenfolge: Sekunde, Minute, Stunde, Tag, Monat, Jahr). Die Eingabe wird grob auf Plausibiliät geprüft.
Bei einer als fehlerhaft interpretierten Eingabe blinkt die LED 1x kurz auf, wenn die Eingabe übernommen wird, dann blinkt sie 6x schnell nacheinander zur Bestätigung.

Via TWI-Schnittstelle kann man jederzeit den Stand der internen Uhr und einen Zähler für die bislang übertragenen Bytes abfragen:
10 Byte auslesen, die 4 ersten Byte sind die Zahl der übertragenen Bytes als uint32, es folgen in 6 Bytes die Uhrzeit und das Datum (beginnend mit der Sekunde).

Erreichte Übertragungsraten

Um die Funktions- und Leistungsfähigkeit des Gespanns VDIP/ATmega88 (bei 3.686400 MHz) zu prüfen, habe ich 2 Tests durchgeführt:
Beim Kopieren einer Datei von 152 kB (mit der Funktion VNC_file_copy()) habe ich freihändig ca. 38 Sekunden gemessen, das sind 8kb "Umsatz" pro Sekunde incl. Öffnen und Schließen, Zeiger positionieren etc.
Kopieren bedeutet: blockweise Einlesen von Stick, Schließen der Quelldatei, Öffnen der Zieldatei, Schreiben des Puffers von 512Byte .. und das 304 mal für 152kB):
Das Ergebnis klingt doch gar nicht schlecht.

Die Übertragungszeit für 1 Byte via SPI beträgt ca. 30us. Die Zeit könnte man sicherlich noch reduzieren, auf der anderen Seite ist nicht die SPI-Übertragung das Nadelöhr, sondern der VDIP, der die Daten auch noch verarbeiten muss.

Als zweiten Test habe ich via HTerm und der Option 'Send File' dieselbe 152kB große Datei mit verschiedenen Baud-Raten via RS232 abgeschickt.
Leider kam die Ernüchterung:
Die Dateigröße von Quell- und Zieldatei stimmte zwar überein, ein Datei-Vergleich lieferte aber nicht reproduzierbare Fehler. Umsomehr, je schneller die Übertragungsrate gewählt war.
Um das Problem einzugrenzen, habe ich eine "präparierte" Quelldatei (die nur aus "ABC" bestand) abgeschickt und an verschiedenen Stellen im Programmablauf die Korrektheit der Daten geprüft.
Am Ende stellte sich heraus, dass bereits direkt nach dem Einlesen von UDR0 (also an der RS232-Schnittstelle) unerwartete Bytes empfangen wurden (und manchmal HTerm offensichtlich abstürzte, nicht mehr weitersendete und nur noch seinen Zeitzähler (sendtime) laufen ließ).

Daher lag der Verdacht nahe, dass möglicherweise HTerm das Problem verursacht.
Aus dem Netz habe ich mir dann "accessPort" geladen.

Und siehe da, mit diesem Programm übertragen waren Quell- und Zieldatei identisch. Bis zu Übertragungsraten von 38400 Baud.
Bei einem Versuch mit 57200 Baud machte mein PC einen spontanen Reset. Weitere Rekord-Versuche habe ich eingestellt, da 38400 Baud im "Dauerbetrieb" zum normalen Loggen weit mehr als ausreichend sind.

Die TWI-Schnittstelle habe ich zum Loggen der Daten eines Beschleunigungssensors getestet (125 Byte / Sekunde). Systematische Tests habe ich an dieser Stelle nicht vorgenommen.

Die SPI-Schnittstelle habe ich nicht getestet ! Da sie aber im Prinzip genauso arbeitet wie die RS232-Schnittstelle, erwarte ich keine grundsätzlich neuen Probleme.
Allerdings müssen Übertragungsgeschwindigkeit und Datenmenge schon an die vorhandenen Randbedingungen angepasst werden.

14. November 2009

Michael S.