Datenlogger für USB-Stick mit VDIP1 und ATmega88 in C

1.) 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.

2.) Hardware

Als Controller wird ein atmega88 verwendet (die Programmgröße beträgt ca. 4 -5kB), 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 .
Ein Schaltplan ist als PDF-Datei beigefügt, hier kurz die textliche Beschreibung der Belegung.
Beim ATmegax8 sind folgende Pins zu belegen (sofern die jeweilige 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, Pullup 10k !)
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 notwendig).

Ein Taster zur Bedienung ist erforderlich. Im Falle der Betätigung zieht er einen Pin auf Masse (PD2 / Pin4)
Hilfreich ist ein Kondensator von ca. 0.1uF zum Entprellen (zwischen PD2 und GND).

Wird der Taster während des Programmstartes so lange gedrückt gehalten, bis LED2 hektisch blinkt (nun darf die Taste losgelassen werden), dann wartet der Logger auf die Eingabe der Systemzeit (default ist 01.07.11 00:00:00).
Die Zeit muss dezimal (nicht als ASCII-Zeichen) eingegeben werden in der Reihenfolge: Tag, Monat, Jahr, Stunde, Minute, Sekunde.

Sobald der Logger mindestens 6 Byte empfangen hat, führt er eine grobe Plausibilitätsprüfung durch.
Geht die erfolgreich aus, dann wird der "Konfigurationsmodus" verlassen, LED1 und LED2 blinken langsam im Gegentakt.

Die Eingabe der Systemzeit kann über die Serielle oder die TWI-Schnittstelle erfolgen.
Im Konfigurationsmodus dürfen keine Daten auf den beiden Eingangen anstehen !

Zwei low-current LEDs, die den Programmstatus signalisieren, sind an Pin14 (PB0) und Pin15(PB1) angeschlossen und über 1.5k mit VCC verbunden.
Zur Kontrolle des Programmablaufes können die beiden LEDs auf dem VDIP dienen, da sie Hinweise auf das Befinden des VDIP geben. Wobei die eine der beiden nur während des Programmstarts Aktivitäten zeigt - sie ist also auch entbehrlich. Die LED's können über Pin2/3 nach aussen geführt werden.

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).
Meine leben zumindest alle noch.

3.) Bedienung des Loggers und Rückmeldung der Status-LED's

1.) Nach den Einschalten und etwa 4 Sekunden Wartezeit zur Initialisierung wird die Einschaltmeldung des VDIP via serieller Schnittstelle übertragen.
1.a) Wird während des Einschaltens die Taste gedrückt gehalten, bis die LED2 blinkt, dann kann die Systemzeit durch Senden von 6 Byte (mit der Zeitinformation) geändert werden.
Sobald 6 Byte empfangen und als Zeit interpretiert werden konnten, geht es weiter mit 1.b) bzw. 2.)
1.b) Wenn kein Stick gefunden wird, dann folgt Blinklicht (5HZ) an beiden LEDs,
der Stick kann jetzt noch eingesteckt werden.
Wenn der Stick eingesetzt wird, dann gehts zu 2.) weiter.
2.) Der Stick ist gefunden, beide Status-LEDs blinken sehr langsam im Gegentakt.
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.
2.b) Wenn während des Wartens der Stick abgezogen wird, dann blinken beide LED schnell (5Hz) im Gleichtakt.
Sobald der Stick wieder gefunden ist, springt das Programm wieder zu Punkt 2.).
3.) Nach dem Starten beginnt das Loggen, die LED1 blinkt ganz gemächlich mit 0.5 Hz.
Immer dann, wenn ein Datensatz auf den Stick geschrieben wird, wird die LED2 getoggelt.
Aber auch die LED am VDIP sollte während des Filewrite aufblinken.
In der Log- Phase darf der Stick auf keinen Fall abgezogen werden, sonst ist Datenverlust unvermeidlich.
Zum Beenden des Loggens muss erneut die Taste gedrückt werden, damit die Daten im Puffer geschrieben und die Datei geschlossen wird.
4.) Das Programm springt nun wieder zu 2).

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.

Wird der Stick versehentlich abgezogen oder gibt des Kontaktprobleme durch Erschütterungen, dann sind die Daten der geöffneten Datei verloren.
Zur Reduzierung dieser Gefahr von Datenverlusten bei längerdauernden Logvorgängen sind folgende Mechanismen und Zeitintervalle festgelegt:

close_time — legt die Dauer in Sekunden fest, nach der eine Datei nach dem letzten Schreiben automatisch geschlossen wird. Dabei wird der Dateieintrag in der FAT aktualisiert.
Wird die Datei später durch ein file_write wieder neu geöffnet, dann wird dabei der Zeiteintrag in der FAT aktualisiert.

save_time — legt die Dauer in Minuten fest, nach der ungesicherte Daten geschrieben werden, auch wenn kein kompletter Datensatz von 512 Byte zusammengekommen ist.

Alle genannten Variablen (zusätzlich noch VNC_timeout und LED_timeout) werden vom Timer decrementiert und für die Steuerung zeitabhängiger Aktivitäten genutzt (z.B. Blinkdauer von LED's).

4.) Debugging

Zum Testen des Programmes sind an zahlreichen Stellen Ausgaben über die Serielle Schnittstelle eingebaut.
Wenn in der global.h #define DEBUG eingetragen ist, dann werden diese Ausgaben in den Programmcode eingebunden, am Terminal können alle Aktivitäten (jeweils mit Zeitangabe) verfolgt werden.

5.) Dateinamen

Dateien werden über eine Zahl (uint8_t) 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.
Diese Suche verzögert den Einschaltvorgang, daher sollten überflüssige Dateien gelöscht oder umbenannt werden.

6.) Dateizeit

Im Programmcode ist eine Defaultzeit definiert, die beim Start des Programmes übernommen wird (01.07.11 00:00). Wenn beim Programmstart die Taste solange gedrückt gehalten wird, bis die LED2 zu Blinken beginnt, kann die Systemzeit verändert werden.
Dazu werden Tag, Monat, Jahr, Stunde, Minute, Sekunde (also 6 Zeichen als Dezimalzahl) an den Logger gesendet. Die Eingabe wird grob auf Plausibiliät geprüft.
Der Logger bleibt solange in der Eingabeschleife, bis eine plausible Eingabe von mindestens 6 Zeichen Länge erfolgt, die als Zeitangabe erkannt wird.

Während des Log-Vorganges können via TWI-Schnittstelle der Stand der internen Uhr und zwei Zähler für die bislang übertragenen/empfangenen Bytes abgefragt werden:
12 Byte auslesen, die 4 ersten Byte zeigen die Anzahl der bereits an den Stick übertragenen Bytes als uint32 (niedrigswertiges Byte zuerst), es folgen 2 Byte mit der Anzahl der im Sendepuffer befindlichen Bytes (wieder das niedrigstwertige Byte voran) und anschließend in 6 Bytes das Datum (Tag, Monat, Jahr (+20)) sowie Uhrzeit (Stunde, Minute, Sekunde).

Bei der Jahresangabe muss 20 subtrahiert werden, da als Bezugsjahr 1980 benutzt wird.
Diese Abfrage über die TWI-Schnittstelle funktioniert nur während des Loggens - denn nur in dieser Zeit ist die TWI-Schnittstelle aktiviert.

Über die Serielle Schnittstelle wird ein Prompt ausgegeben, das die aktuelle Systemzeit anzeigt.

7.) 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 vom 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 weiter 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.

16. Juli 2011

Michael S.