Projektarbeit

Datenlogger

Gruppe:
D. Büchler
M. Hess
M. Richter

Aufgabenstellung

Die ganze Aufgabenstellung konnte voll umfänglich realisiert werden. Da sich mit unserem Projekt 3 Mann beschäftigten, wurde die Funktionalität erweitert.
Folgende Optionen konnten zusätzlich implementiert werden .:

1. Die Ausgabe der Messungen per RS232 wurde mit der Zeitangabe und einer Messwert-Nummerierung ausgebaut
2. vergrösserter Bereich des Zeitintervall (1ms – 60s)

Da das ganze Projekt mit der internen Hardware des Controllers gelöst werden konnte, wird in dieser Dokumentation nur auf das Schema des TSU Board's und das Hardware Manual des HITACHI Microcontrollers H8/3067 verwiesen.

Planung

Bei der Planung dieses Projektes ergaben sich folgende Teilaspekte:

- Das UI (User Interface), welches zur Eingabe der Parameter durch den Benutzer dient und auf die gespeicherten Werte zugreift
- Die Timer Konfiguration wird vor dem Start jeder Datenerfassung neu berechnet
- Die ISR liest nach jedem Interrupt den Port und schreibt den Wert, mit einem Zeitstempel ausgestattet, ins RAM
- Die Ausgabe auf RS232

Realisierung

Das UI wurde mittels Tastatur und Display realisiert. Die Software bietet eine komfortable Benutzerführung, welche den User Schritt für Schritt zur Eingabe der Parameter auffordert. Nachdem die Messwerterfassung gestartet wurde, wird bei jedem Interrupt die Schalterstellung ausgelesen und mittels einem Array gespeichert. Nach abgeschlossener Erfassung kann auf jeden abgespeicherte Wert direkt zugegriffen werden und auf dem Display ausgegeben werden, oder alle Werte werden über RS232 ausgegeben.

Das Projekt wurde in folgende Funktionen aufgeteilt:

Funktion: main

Das Hauptprogramm beinhaltet primär das UI und die Aufrufe der folgend beschriebenen Funktionen. Die Benutzereingaben erfolgen über die Tastatur, welche in der Funktion "ISR_IRQ0" gesteuert wird. Diese löst bei jedem Tastendruck einen Interrupt aus, decodiert den empfangenen Wert und übergibt diesen an das Hauptprogramm. Zur Steuerung des Displays wurde auf die vorhandenen Funktionen im "CB_TSU" zurückgegriffen. Die "wait"-Funktion hat die Aufgabe, den Kontroller eine gewisse Zeit zu beschäftigen, damit der Benutzer genügend Zeit zum lesen des angezeigten Textes hat, bis das Programm weiterfährt.
Der Bediener wird nach einer Begrüssung aufgefordert, die Parameter wie die Anzahl zu loggenden Daten und den Zeitintervall einzugeben. Nach jeder Eingabe wird der Wert zur Bestätigung nochmals eingeblendet. Nach erfolgter Eingabe beginnt der Log- Prozess, welcher durch die Funktionen "timer_config" und "int_tmr0" initialisiert und gesteuert wird. Für die zu speichernden Daten wurde eine geeignete Struktur (Sdata) definiert, welche im Kapitel "Wahl der Datenstruktur" detailliert erklärt ist. Anschliessend kann man jeden einzelnen Messwert abrufen und am Display ausgeben, oder alle Messwerte über die Funktion "Write_Data_RS" per RS232 an einen angeschlossenen PC senden.

Funktion: timer_config

In dieser Funktion werden die Register des 8 Bit Timer0 gesetzt und des weiteren der Timer gestartet. Die Übergabewerte dieser Funktion beschränken sich auf "time" (Intervall) und "steps" (Anzahl Messungen). Der Timer wird so parametriert, dass wenn der Zählwert mit dem Vergleichswert im Compare Register übereinstimmt, ein Interrupt ausgelöst wird und der Zählvorgang wieder von vorne beginnt. Der Clock- Vorteiler wurde auf das Maximum von 8192 gesetzt.
Damit alle in der Aufgabenstellung vorgesehenen Erfassungszeitabstände (d.h. der Abstand zwischen den einzelnen Erfassungen) abgedeckt werden konnten, wurde ein sog. Prescaler (Variable pre_count) eingefügt. Wenn nun "time" kleiner als 100ms ist, wird der Wert des Compare Registers errechnet und gesetzt. Der Prescaler wird in diesem Betriebsfall nicht benötigt (pre_count = 0). Wenn "time" grösser als 100ms ist, wird das Compare Register fix auf 100ms eingestellt und der Prescaler erhält die Anzahl Interrupts, bis die Messung aktiviert wird.

Funktion: int_tmr0

Diese Funktion wird von dem Timer0 Interrupt aufgerufen und speichert den Zustand der Schalter in ein Array.
Der Prescaler wird, wenn durch die Status-Variable (pre_wait) angezeigt, bei jedem Timer-Event dekrementiert. Erst wenn diese Prescale-Variable Null erreicht, wird die eigentliche Messung durchgeführt und der Zähler wieder in den ursprünglichen Zustand gesetzt.
Da nur eine beschränkte Anzahl Messwerte aufgenommen werden soll, ist eine weitere Zählvariable vorhanden. Diese beinhaltet am Anfang die Anzahl der durchzuführenden Messwerte und wird im Verlauf der Erfassung bis zum Wert 0 dekrementiert. Ist dieser Wert 0 erreicht, so beendet sich die ISR Messdatenerfassung durch sperren des Timers 0 selbst. Durch eine Variable (done) wird dies dem Hauptprogramm mitgeteilt.

Durch diesen Aufbau ergeben sich folgende Besonderheiten:

1) Die Messwerte werden in umgekehrter Reihenfolge abgespeichert. Der erste Messwert ist also im Array am Platz des letzten (= der eingegebenen Anzahl Messwerte), der letzte Messwert am Platz 0 zu finden.
2) Die Laufzeit der ISR ist nicht fix. Je nachdem ob der Pre-Scaler verwendet wird und ob dieser im aktuellen Zyklus gerade den Wert Null erreicht, wird eine wesentlich längere Laufzeit auftreten. Aus diesem Grund wurde auf eine Ausgabe des aktuellen Zählerstandes auf die LCD verzichtet.

Zur Visualisierung werden die erfassten Schalter auf die LED's ausgegeben. So kann der Benutzer erkennen, wann ein Messwert übernommen wurde ohne die zeitaufwendige LCD-Ansteuerung zu benutzen.

Funktion: Write_Data_RS

Optionaler Teil der Aufgabenstellung war das Ausgeben der Messwerte über die serielle Schnittstelle. Dies wurde auch implementiert. Einmal aufgerufen, sendet die Funktion „Write_Data_RS" zunächst einmal einen Header, der eine kurze Erklärung der nachfolgenden Daten enthält, zur seriellen Schnittstelle. Anschliessend folgen die Messwerte. Hierbei ist zu beachten, dass diese in der Richtigen Reihenfolge ausgegeben werden, d.h. der erste Messwert wird auch als erster wieder ausgegeben. Die Messwerte werden von der Zeit (die im derzeitigen Entwicklungsstand „nur" eine fortlaufende Nummer ist) durch ein Komma getrennt. Dies würde eine einfache Weiterbearbeitung in Tabellenkalkulationsprogrammen etc. ermöglichen.

Wahl der Datenstruktur

Bei der Wahl der Datenstruktur stand zunächst einmal fest, dass maximal 10000 Werte abgespeichert werden sollen. Eine überschlagsmässige Rechnung ergab dann auch, dass dieser Wert in den vorhandenen Speicher passt.
Um möglichst einfachen Zugriff auf die Elemente (Messdaten, Zeit) zu haben, entschieden wir uns, ein struct zu definieren. Anschliessend bildeten wir ein eindimensionales Array aus diesem struct mit den oben genannten maximal möglichen Werten. Implementiert wurden also 10001 mögliche Werte. Da diese Variable auch von der ISR geschrieben wurde, fügten wir zur Sicherheit das Schlüsselwort „volatile“ an. Dies taten wir bei sämtlichen von den ISR’s benutzten Variablen. Eine genauere Untersuchung, wo dies wirklich notwendig ist, konnten wir aus Zeitgründen nicht mehr durchführen.

Fazit

Nach einigen Schwierigkeiten konnte die Aufgabe gelöst werden. Leider wurde der in der Aufgabenstellung festgehaltene Zeitrahmen überschritten. Im Rückblick traten folgende Probleme auf, welche aber allesamt gelöst werden konnten und einen grossen Lerneffekt darstellten:

1) Sowohl die Keyboard- ISR wie auch die Timer- ISR veranlassten das Hauptprogramm nicht zum warten bis die ISR ihre Arbeit beendet hatten. Dies trat auf, weil die diesbezüglichen Variablen vom Compiler wegoptimiert wurden. Dies wurde leider erst aufgrund eines Assembler- Listings entdeckt.
2) Da die einzelnen Routinen „blind“ programmiert werden mussten (d.h. ohne z.B. das aufrufende Programm zu haben) mussten dafür Test- Routinen geschrieben werden. Trotz der guten Vorbesprechung bezüglich der verwendeten Variablen bzw. deren Typ, mussten einige Anpassungsarbeiten beim zusammenfügen vorgenommen werden. Dies ist wahrscheinlich normal und kann nicht umgangen werden.

Obschon ein erster Blick auf die Aufgabenstellung nicht allzu grossen Schwierigkeiten erahnen lässt (d.h. keine unbehandelten Themen etc.) war die Aufgabe doch recht "happig". Auch traten aufgrund der sehr unterschiedlichen Programmier- Stile einige Abstimm-Schwierigkeiten auf. Die Aufgabe war aber sicher sehr lehrreich und sollte in diesem Stil beibehalten werden.

Anhang 1: Struktogramm

Anhang 2: Sourcecode