Messung der Beleuchtungsstärke mit Light-to-Frequency-Converter TSL230R und ATtiny2313 in C

1.) Allgemeines

Der Light_To_Frequency-Converter TSL230 erzeugt an seinem Ausgangspin einen Takt, dessen Frequenz proportional zur empfangenen Energie (des Lichtes) ist.

Die Empfindlichkeit des Sensors kann in drei unterschiedlichen Stufen eingestellt werden.
Die Empfindlichkeit der drei Stufen unterscheiden sich jeweils um den Faktor 10.
Zusätzlich kann durch interne Vorteiler die Frequenz des Ausgangssignals skaliert werden.

Der Sensor kann so den Bereich ca. 0.01 Lux bis zu 100.000 Lux (also vom Mondlicht bis zu grellem Sonnenlicht) mit recht hoher Linearität erfassen - wobei darauf hinzuweisen ist, dass nicht die fotometrische Beleuchtungsstärke gemessen wird !

Die spektrale Empfindlichkeit des Sensors ist nicht identisch mit der des menschlichen Auges, sein "Sehbereich" geht deutlich über den des Auges hinaus.
Mit Hilfe vorzuschaltender optischer Filter kann die Messung auf ausgewählte spektrale Bereiche beschränkt werden.

Am Ausgangspin des Sensors liegt ein Signal an, dessen Frequenz bis 1Mhz erreichen kann.
Durch die Wahl unterschiedlicher Empfindlichkeitsbereiche und zusätzlicher Vorteiler lässt sich die Frequenz des Ausgangssignals der geplanten Art der Auswertung anpassen.

2.) Funktionsweise der Programmes

Der Lichtsensor wird von einem ATtiny2313 gesteuert.
Der Controller wählt automatisch einen geeigneten der drei Empfindlichkeitsbereiche (Sensitivity) aus und gibt die Beleuchtungsstärke als Zahl von 0 ... 100.000 aus. Der Wert entspricht grob (mit unterschiedlichen Toleranzen) der Beleuchtungsstärke in Lux.
Die Berücksichtigung der spektralen Abhängigkeit der Messung ist nicht Teil dieser Betrachtung.

Die Arbeitsprinzip der Auswertung ist relativ simpel:
Es werden die Impulse am Ausgang des Sensors innerhalb eines festgelegten Zeitintervalls gezählt.
Dazu gibt der Timer0 ein Zeitintervall von 0.1 Sekunden vor, während dessen die am Pin T1 anliegenden Flankenwechsel vom 16-Bit-Counter1 registriert werden.

Die max. Ausgangsfrequenz am Sensor beträgt ca. 1MHz. Das Messintervall 0.1 Sekunden.
Daher besteht die Gefahr, dass der 16-Bit Counter bei 2^16 überläuft.

Um diesem Überlauf vorzubeugen, ist für den Timer/Counter1 ein CompA-Interrupt geschaltet, der bei einem Zählerstand von 0xFFFE auslöst.
Wird dieser Interrupt erkannt, dann schaltet der Controller in einen weniger empfindlichen Bereich (die Frequenz am Ausgang des Sensors wird reduziert).
Ist bei sehr großem Energieeinfall der unempfindlichste Bereich bereits erreicht (Sensitivity = 1), dann kann im TSL230 ein zusätzlicher Vorteiler von 2 aktiviert werden, der die Frequenz am Ausgangspin halbiert.

Wird erkannt, dass der Zähler des Counters kleiner als 4096 ist, dann versucht der Controller in einen empfindlicheren Bereich zu schalten (die Frequenz wird höher, es werden mehr Impulse gezählt).

Nach Ablauf des Messintervalls wird der Zählerstand im Counter1 unter Berücksichtigung des Vorteilers umgerechnet.
Der Messwert wird zusammen mit dem Sensitivity-Faktor zurückgeliefert und kann ausgewertet werden.

Der Messwert wird (auf 3 Byte reduziert) mit angehängten Sensitivity-Faktor byteweise ins USI_TWI-Modul geschrieben und auch über die Serielle Schnittstelle gesendet. Für die Daten, die seriell ausgegeben werden, sind - in Abhängigkeit von der auswertenden Instanz - unterschiedliche Formatierungen vorbereitet.

Der Hintergrund, warum die Division des Messwertes durch den Teiler nicht schon auf dem Controller ausgeführt wird, ist einfach:
Bei einer Ganzzahl-Division durch 100 gehen zwei "Nachkommastellen" verloren, die möglicherweise bei geringen Helligkeitspegeln gut nutzbar wären.

Beim Export der Daten über die Serielle Schnittstelle ist eine Ausgabeformatierung vorbereitet, bei der eine Fließkomma-Division durch 10 und 100 durch Einschmuggeln eines Dezimaltrennzeichens "simuliert" wird - damit mag der Import von Messwerten in eine Tabellenkalkulation vereinfacht werden.

3. Einsetzbare Schnittstellen des Controllers

Der Tiny2313 kann via USI-TWI oder via Serieller Schnittstelle gesteuert werden.
Bei parallelen Zugriffen wird zuerst die Serielle Schnittstelle bedient.
Das Auslesen der gewonnen Daten ist über beide Schnittstellen möglich.

Über die Serielle Schnittstelle können größere Distanzen überbrückt werden, allerdings immer nur zu genau einer Gegenstelle.
Über USI-TWI können bis zu 128 Busteilnehmer adressiert werden, allerdings ist die Leitungslänge eingeschränkt.

4.) Betriebsmodus "free-running"

Wenn der Controller nicht im "Sleep-Modus" arbeite, dann befindet er sich im "free-running-Modus".
Hier werden in regelmäßigen zeitlichen Abständen Messungen ausgeführt und die Ergebnisse via Serieller Schnittstelle ausgegeben und parallel im USI-Modul zur Abholung durch den Master bereitgestellt.

Die Wartezeit zwischen zwei Messungen ist einstellbar.
Der Controller verbringt die Wartezeit in einer delay-Schleife, die aber beim Auftreten eines USI-Start- oder RX-Interruptes beendet wird.
Auf diese Weise kann jederzeit die Warteschleifen unterbrochen und eine Messung ausgelöst oder die Konfiguration geändert werden.

5.) Betriebsmodus "Sleep"

Im "Sleep-Modus" wird der Controller und der TSL230 in den stromsparenden Power-Down-Modus versetzt.
Die Stromaufnahme wird auf ca. 70uA reduziert.

Über die USI-TWI-Schnittstelle ist der Controller auch im Sleep-Modus jederzeit ohne Einschränkung ansprechbar.
Um ein Messergebnis zu liefern, muss er von aussen geweckt werden (= die TWI-Adresse an den Slave senden) und das Ergebnis muss nach der Wandlungszeit abgeholt werden.

Der "Sleep-Modus" funktioniert bei der Benutzung der Seriellen Schnittstelle eigentlich nicht, da der Empfang von Daten keinen Interrupt auslösen kann, der ein Wake-Up ausführen würde.

Diese Einschränkung ist durch einen kleinen Trick soweit umgangen, dass eine Datenaquise im "Sleep-Modus" möglich ist:
Bevor der Controller in den Sleep-Modus übergeht, wird der Empfang von Daten via RS232 deaktiviert - und dafür der INT0 als Low-Level-Interrupt aktiviert.
Der Pin INT0 ist mit dem RX-Pin verbunden.

Werden nun Daten via Serieller Schnittstelle an den Controller versandt, so zieht das Startbit den Pin RX und damit INT0 auf Masse - das löst den INT0_IRQ aus und beendet den Sleep-Modus.

Der Controller erkennt aber keine empfangenen Daten (der Empfang ist ja disabled), löst dennoch eine Messung aus, stellt die Daten ins USI-Modul und versendet sie via Serieller Schnittstelle.

Die einzige Einschränkung bei der Benutzung des Sleep-Modus in Verbindung mit der Seriellen Schnittstelle ist, dass - nachdem der Controller einmal in den Sleep geschickt wurde - er über diese Schnittstelle nicht mehr umkonfiguriert werden kann. Der "Sleep-Modus" kann nicht mehr beendet werden.

Für die Arbeit mit der USI-TWI-Schnittstelle bereitet der "Sleep-Modus" keine Einschränkung.

4. Auslesen von Daten via RS232

Der Controller kann "free running" in regelmäßigen Zeitabständen Messungen ausführen und die Messwerte unmittelbar nach der Messung via RS232 versenden.

Die Dauer der Messung beträgt in der Regel 0.1 Sekunden, im ungünstigsten Falle 0.3 Sekunden, wenn der Empfindlichkeitsbereich 2x korrigiert werden muss (was aber nur bei extremem Helligkeitsunterschied zu erwarten ist).

Im "free-running-Modus" kann während der Wartezeit ein Zeichen (ein RETURN via RS232 bzw. die TWI-Adresse via TWI) gesendet werden, dann beendet der Controller die Warteschleife und führt eine Messung aus.

Im "Sleep-Modus" arbeitet der Controller in analoger Weise:
Ein RETURN (bzw. die TWI-Adresse) stößt eine Messung an.

5.) Auslesen von Daten via USI-TWI

Der Controller arbeitet im Hintergrund als USI-TWI-Slave.
Er kann damit aber nicht selbstständig agieren. Jede Aktivität muss vom Master ausgehen:
Der Master muss eine Messung starten und nach der Wandlungszeit die Daten abholen.

Eine Messung wird durch das Senden der TWI-Adresse via TWI an den Slave gestartet.
Der Slave führt die Messung aus und stellt die Daten in seinem USI-Modul zur Abholung bereit.

Jedesmal, wenn der Master Daten vom Slave abholt, wird direkt im Anschluss eine neue Messung ausgeführt.

Es reicht also aus, lediglich Daten abzuholen, ohne jeweils vorher eine Messung explizit zu verordnen.
Allerdings muss man sich im Klaren sein, dass die Messwerte damit "veraltet" sind, da sie bei der letzten Messung gelesen wurden.

Die Arbeit mit dem USI-Modul ist ohne Einschränkung im Sleep-Modus möglich, da ein USI-Start-IRQ den Controller aus dem Sleep wecken kann.
Wird im Sleep-Modus die TWI-Adresse eines anderen Gerätes übertragen, dann wacht der Controller zwar auch aus dem Sleep auf, aber sobald erkannt ist, das nicht die eigene Adresse übertragen wurde, verfällt er wieder in den Power-Down-Modus.

Wird dagegen die eigene Adresse erkannt (gleichgültig ob ein Read oder Write vorliegt), dann wird eine neue Messung ausgeführt und anschließend wieder in den Sleep-Modus zurückgekehrt.

Arbeitet der Controller nicht im Sleep-Modus, dann werden in regelmäßigen Zeitabständen Messungen ausgeführt und die Daten im USI-Modul bereitgestellt und via RS232 versandt.

7.) Schaltfunktionen

Als Zusatzfunktion sind 2 unabhängige Schaltfunktionen implementiert, die bei der Über-/Unterschreitung von Grenzwerten einen der beiden Schaltpins ein-/ausschalten.
Für jeden Schalter existieren 2 Grenzwerte, die den Ein- bzw. Ausschaltpunkt definieren.

Die Helligkeitslimits können im laufenden Betrieb von aussen geändert werden.

Es muss darauf geachtet werden, dass der gelieferte Messwert mit dem Teiler korrigiert ist und im 16-Bit-Format vorliegt. Da der Speicher auf dem ATtiny knapp ist, ist die Schaltfunktion per default deaktiviert.

8.) Konfigurationsmöglichkeiten

Sowohl über USI-TWI als auch RS232 sind einstellbar:

Command "1" - Sleep-Mode Ein/(Aus = regelmäßiges Messungintervall = "free running").
Command "2" - Das Zeitintervall zwischen den regelmäßigen Messungen [in Sekunden].
Command "3" - Kontroll-LED signalisiert den Messungvorgang Ein/Aus.
Command "4" - Ausgabe des Messwertes über die Serielle Schnittstelle Ein/Aus.
Command "5" - Schaltvorgänge beim Überschreiten von einstellbaren Helligkeitslimits Ein/Aus.
Command "1" - Helligkeitslimits (jeweils oberer und unterer Grenzwert) für Schalter 1.
Command "2" - Helligkeitslimits (jeweils oberer und unterer Grenzwert) für Schalter 2.

Bei der bestehenden Voreinstellung sind Kommando und Parameter als Binärwert zu senden (nicht als ASCII-Wert).

9.) Steuerung via Serieller Schnittstelle

(Wichtig: die Serielle Schnittstelle erwartet als Abschluss der Übertragung ein RETURN !).

Ein einzelnes Zeichen senden (z.B. nur ein Return) führt eine Messung aus und liefert den Messwert auf COM aus.

Zwei Zeichen senden. Das erste Zeichen ist das Kommando, das zweite Zeichen der Parameter:

1, [0/x]	Sleep-Modus AUS = 0/EIN = x
2, [1...255]	Pause in Sekunden. zwischen den Messungen
3, [0/x]	Senden via Serielle Schnittstelle AUS = 0/ EIN = x
4, [0/x]	Controll-LED AUS = 0/EIN = x
5, [0/x]	Schaltfunktionen AUS = 0/ EIN = x

Drei beliebige Zeichen senden (sofern #DEBUG aktiviert ist)
Sendet via Serieller Schnittstelle die eingestellten Parameter (siehe "zwei Zeichen senden")

Fünf Zeichen senden, das erste Zeichen ist das Kommando, es folgen zwei 16Bit Werte, getrennt nach Upper_Byte und Lower_Byte zur Definition der unteren bzw. oberen Schaltschwelle des angegebenen Schalters (1 oder 2)

1, 0, 100, 0, 200	Setzt den Schalter1 so, dass er bei Unterschreiten des Wertes 100 einschaltet und bei Überschreiten des Wertes 200 wieder ausschaltet
2, 0, 200, 0,100	Setzt den Schalter2 so, dass er bei Überschreiten des Wertes 200 einschaltet und bei Unterschreiten des Wertes 100 wieder ausschaltet

Noch einmal:
Die Eingabe über die Serielle Schnittstelle muss mit einem RETURN abgeschlossen werden.
Nur am RETURN wird das Ende der Sendung über die Serielle Schnittstelle erkannt !

Befehl und Parameter werden in der Voreinstellung binär erwartet (also nicht als ASCII-Zeichen).
Dies ist besonders bei der Arbeit mit einem Terminal-Programm zu bedenken:
In HTerm etwa darf als Eingabeformat nicht 'ASC' eingestellt sein.

10.) Steuerung via USI-TWI

Über die USI-TWI-Schnittstelle kann der TWI-Slave nach der gleichen Logik wie über die Serielle Schnittstelle ausgelesen und konfiguriert werden.
Für die Eingabe über USI-TWI muss natürlich jeweils die TWI-Adresse vorab gesendet werden.
Zum Auslesen der Daten mit gesetztem Read-Bit !

Angenommen, die Adresse des Slaves sei 64 (0b010000 + R/W-Bit).

1-Byte Command (TWI-Adresse + 1 Byte)

64	Senden der TWI-Adresse führt eine Helligkeitsmessung aus
65, 4 Byte lesen	liest die Helligkeit + Teiler aus (und führt eine neue Messung aus)

2-Byte Command (TWI-Adresse, Command, Wert):

64, 1, 0/x	Schaltet den Sleep-Modus aus(0)/ein(x)
64, 2, 100	Wählt als Pause zwischen den Messung 100 * 1 Sekunde = 100 Sekunden Bei ausgeschaltetem Sleep-Modus werden diese Pausen zwischen den Messungen eingelegt
64, 3, 0/x	Schaltet den LED aus(0)/ein(x)
64, 4, 0/x	Schaltet die Ausgabe über die Serielle Schnittstelle aus(0)/ein(x)
64, 5, 0/x	Schaltet den Schalt-Modus aus(0)/ein(x)

3 Byte Command (TWI-Adresse + 3 beliebige Zeichen)
gibt die internen Einstellungen via RS232 (!!) aus

5 Byte Command (TWI-Adresse, Command, H_Byte, L_Byte, H_Byte, L_Byte)

64, 1, 0, 100, 0, 200	stellt den unteren / oberen Grenzwert für Schalter 1 auf 100Lux / 200Lux
64, 2, 4, 0, 5, 0	stellt den unteren / oberen Grenzwert für Schalter 1 auf 1024Lux / 1280Lux

Die Bedienung via Serieller Schnittstelle erfolgt analog, lediglich das Senden einer TWI-Adresse muss unterbleiben.

11.) Wer misst, misst Mist

Bleibt letztlich die Frage, welche Aussagekraft der Messwert hat.

Laut Datenblatt liefert der Sensor bei der Einstellung für Sensitivity = 100 bei einer Beleuchtungsstärke von 1uW/cm² am Ausgang einen Takt von ca. 1KHz.

1uW/cm² = 1uW * 100 * 100/m² = 10.000uW/m² = 10mW/m²

Sensitivity	Beleuchtungsstärke	Frequenz	Frequenz bei 1W/m²	Maximalwert bei f=1MHz
100	10mW/m²	1000 Hz	100 KHz	10 W/m²
10	10mW/m²	100 Hz	10 KHz	100 W/m²
1	10mW/m²	10 Hz	1 KHz	1000 W/m²

Da das Messintervall 0.1 Sekunden lang ist, liefert der Messwert nur 1/10 der Frequenz zurück.
Bei einer Beleuchtungsstärke von 1000 W/m² und der Sensivity 1 also den Wert 100.000.

Die als Ergebnis der Messung gelieferte Zahl entspricht der Beleuchtungsstärke in der Einheit [10 mW/m²].

In der Literatur findet man folgende Hinweise zum Zusammenhang zwischen solarer Einstrahlung und Beleuchtungsstärke in Lux:

Heller Sonnentag	100.000 Lux
Bürobeleuchtung	500 Lux
Vollmondnacht	0.25 Lux

"Der Wert von 100.000 Lux entspricht etwa 1000 Watt pro Quadratmeter Sonneneinstrahlung".

Welch Zufall. Unser Messwert liefert als sehr grobes Schätzeisen die Beleuchtungsstärke in der Einheit [Lux].

Vergleichsmessungen mit einem Luxmeter bestätigen den grundsätzlichen Zusammenhang.
Allerdings je nach Spektrum der Lichtquelle mit teilweise deutlicher Abweichung.
Insbesondere Glühlampenlicht mit seinem hohen Infrarotanteil liefert wesentlich höhere Messwerte als zu erwarten.
Noch gravierender sind die Abweichung bei IR-Dioden: während das Auge überhaupt keine Helligkeit wahrnimmt, träumt der Sensor von tropischer Sonne.

Der TSL230 kann Licht im Bereich der Wellenlänge von 300nm bis 1100nm registrieren.
In den Grenzbereichen jedoch nur noch mit einer "normalisierten Empfindlichkeit von 10%".
Seine höchste Empfindlichkeit liegt im infraroten Bereich von etwa 640nm bis 900nm bei 100% bis zu 120%.

Zur Erinnerung:
Blaues Licht hat eine Wellenlänge von ca. 470nm, grünes Licht etwa 530nm und rotes Licht 630nm.
Die zugeordneten "normalisierten Empfindlichkeiten" sind 70% (Blau), 80% (Grün). 100% (Rot).

Die Lichtwahrnehmung des Sensors entspricht damit nicht der des menschliche Auges.
Die Verfälschung der Messung durch unsichtbare, infrarote Strahlung ist zu berücksichtigen.
Sinnvollerweise wird ein optischer Filter im Strahlengang vor dem Sensor platziert, um die Messung auf die auszuwertenden Bereiche des Spektrums zu fokussieren.

12.) Schaltplan

Ein Schaltplan ist beigefügt. Dazu einige Hinweise.

Die Widerstände 470 in der TX-Leitung ist nicht zwingend erforderlich.
Er sollen lediglich den Ausgang bei einem versehentlichen Kurzschluss schützen.

Der Widerstand 10k am RX-Pin dient dazu, bei offenem RS232-Eingang ein "Flattern" des RX-Pins zu unterbinden.

Der Kondensator von 4.7nF am RX-Pin des ATtiny soll die Spikes wegbügeln, die bei mir im 20us-Takt an diesem Pin auftreten:
Als Folge wurde der "Sleep-Modus" durch Auslösen des INT0-Interrupts sofort wieder beendet.

Der Kondensator zeigt keine erkennbaren nachteiligen Folgen, die Datenübertragung wird trotz eines 50m langen (teilweise aufgerollten) Telefonkabels zwischen Sensor und PC nicht beeinträchtigt.

Der Pin S3 am TSL230 darf - ohne Verbindung zum Controller - dauerhaft auf LOW gelegt werden, solange (wie im vorliegenden Programmbeispiel) nur die Scale-Faktoren 1 und 2 genutzt werden.

Beschränkt man die Messwerte auf 16-Bit (Verzichtet auf die Messung extremer Helligkeit), dann kann auf das Scaling der Ausgangsfrequenz völlig verzichtet werden und es reicht aus, Pin S2 und S3 ohne Verbindung zum Controller auf gnd zu legen. Allerdings sind auch kleinere Anpassungen in der TSL230.c erforderlich.

Der Programmcode wird - bei Beschränkung auf 16-Bit Werte - um einiges kompakter.

Die Beschaltung der Pins S2/S3 gemäß Datenblatt:

f0 Scaling	Pin S3	Pin S2
1	Low	Low
2	Low	High
10	High	Low
100	High	High

Viel Spaß beim Messen,

Michael S.