

1-Kanal logic-analyser für Mega8 in C

Die Funktionsweise:

Der 16-Bit-Timer1 zählt die Takte, die er über einen wählbaren Vorteiler (Prescaler) vom Quarz gefüttert wird.

Beim Überlauf des Timer1 wird die Timer1_OVF_ISR aufgerufen, die ihrerseits ein drittes Byte decreментиert (!).

Läuft nun dieses dritte Byte von 0 auf 255 über, dann wird ein Flag gesetzt und das Scannen beendet.

Wie man es von „normalen“ Timern kennt, kann man den Counter mit einem Preload-Wert vorbelegen, der Überlauf wird schneller erreicht, das Intervall des Scanvorganges wird verkürzt (dazu -txxx eingeben, wobei $xxx < 256$ sein muss).

Der Preload wird auf das höchstwertige Byte des 24-Bit-Counters angewendet.

Ein Preload von 0 bedeutet, dass nach 2^{16} Counts der Counter stoppt.

Die Dauer des Messintervalls in Sekunden ist somit $(\text{Preload} + 1) * 2^{16} * (\text{Prescale} / 16.000.000)$.

Jeder Flankenwechsel am Eingang wird durch Pollen des Flags ICF1 im Register TIFR erkannt.

Danach wird der aktuelle Zählerstand aus den Registern ICR1L, ICR1H und aus dem Counter des dritten Bytes gerettet und im SRAM abgelegt.

Der Rest ist Interpretation der gewonnenen Daten.

Nach meinen Tests sind Messungen bis ca. 200 KHz möglich (halber Takt mindestens 2.5us).

Danach reicht die Zeit zwischen den Taktwechseln nicht mehr aus, um die Counterwerte zu speichern, es gehen Flankenwechsel verloren.

Da die 24-Bit-Zahlen schwer zu lesen sind und die wahren Zeiten in der Regel weniger interessieren, kann man sich die Differenzen zwischen aufeinander folgenden Signalen ausgeben lassen: die Dauer der low-/high Phase (damit läßt sich ein RC5-Signal gut interpretieren).

Die Ausgabe der Daten wird gesteuert mit -a[1|2|4].

Als Gesamtpuffer für die Zeitstempel sind 900 Byte vorgesehen. Darin lassen sich $900 / 3 = 300$ edges zählen. Bei 2 edges pro Takt somit 150 Takte.

Das bedeutet für die Aufzeichnungsmöglichkeiten:

Eine RC5-Übertragung hat 14 Takte, es können 10 Tastendrucke gespeichert werden - die Tastendrucke können allerdings durchaus mit großer zeitlicher Distanz erfolgen.

Ein Fernsteuerservo wird 50x pro Sekunde angesteuert, somit sind ca. 3 Sekunden logbar.

Ein DCF77-Signal hat 58 Takte pro Minute, es werden also knapp 3 Minuten des DCF77 Signals logbar sein.

Abhängig vom gewählten Prescaler (dem Vorteiler des Quarztaktes von 16 MHz) ergeben sich unterschiedliche Auflösungen (Dauer einer Einheit) und Logzeiten (Dauer einer Einheit * 2^{24}):

Prescaler	Dauer [us]	max. Logzeit [sek]
8	0,5	8,4
64	4	67
256	16	268
1024	64	1073

Der Prescaler wird mit -p[8|6|2|1] festgelegt. Es wird jeweils nur die erste Ziffer der Werte 8, 64, 256 oder 1024 als Eingabe geprüft.

Die Mindestdauer eines Impulses muss 2,5us betragen, ansonsten schafft es der mega8 nicht, die Daten des letzten Impulswechsels rechtzeitig im SRAM abzulegen.

Die Hardware

Im Programmbeispiel habe ich einen ATmega8 mit 16MHz Quarz eingesetzt. Für die Kommunikation mit dem PC wird ihm ein MAX232 zur Pegelwandlung beigelegt (Standardbeschaltung).

Das zu analysierende Signal wird an ICP1 angelegt (aber bitte nur bis +5V).

Alternativ kann man den Analog-Comparator einsetzen, um den Umschaltzeitpunkt (Triggerzeitpunkt) spannungsmäßig zu definieren. Dazu wird der AIN0 mit einem Poti auf etwa 1/2 VCC gelegt - und kann bei Bedarf justiert werden. Das Signal muss dann auf den Pin AIN1 gelegt werden.

Bei meinen Test hatte ich keine Probleme mit dem Anschluss an ICP1.

Die Aufzeichnungsdauer (in wahrer Zeit) hängt ab vom Prescaler und vom Preload.

Die maximale Aufzeichnungsdauer wird durch einen hohen Prescaler (1024) und einen Preload von 255 erreicht. Um die Aufzeichnungsdauer bei gleichbleibender Auflösung zu beschränken, kann man den Preload reduzieren (-i[255 ... 0]).

Die Aufzeichnungsdauer in Sekunden beträgt: $(\text{Preload} + 1) * 2^{16} * (\text{Prescaler} / 16.000.000)$.

Als Statusmeldung sind 4 LED's an PORTC vorgesehen:

PC3 Power on (leuchtet, sobald der mc den Port als Ausgang geschaltet hat)

PC2 Start Trigger (leuchtet, solange auf einen Triggerimpuls gewartet wird)

PC1 Scan (toggelt abwechselnd mit LED-ready während des Scanvorganges)

PC0 ready (toggelt wie PC1 jeweils bei Timer1overflow),
leuchtet ununterbrochen nach Ende des Scannens.

Alle LEDs sind über je einen Vorwiderstand 1.5k mit VCC verbunden.

Die Bedienung

Die Steuerung sowie die Änderung aller Parameter des Logic-Analysers erfolgt über ein Terminalprogramm (z.B. hTerm). Die Baudrate ist auf 9600, 8, 1 eingestellt.

Voran geht der Eingabe jeweils ein "--".

-? gibt eine Hilfe mit allen Optionen sowie den aktuellen Voreinstellungen.

Wichtig ist, dem Analyser mitzuteilen, ob der Ruhelevel auf high (default_Level = 1) oder auf low (default_Level = 0) liegt. Wird ausgeführt mit -l[0|1].

Bei Ruhelevel high wartet der Trigger auf eine fallende Flanke, bei Ruhelevel low auf eine steigende Flanke. Bei falscher Einstellung wird am Ende eine ungerade Zahl von edges angezeigt (weil der erste Flankenwechsel nicht erkannt wurde). Ausserdem sind die Angaben zu high und low Pegel in der Ausgabe falsch (negiert).

Mit -sd startet das Scannen sofort durch, mit -st wartet das Programm, bis die erste Flanke erkannt wird - und startet dann erst.

Beim Starten mit -st sollte der erste Counterwert daher immer 0 lauten, beim Starten -sd immer deutlich > als 0 sein.

Mit -o kann die Ausgabe der Daten beliebig häufig wiederholt werden, auch mit geänderten Voreinstellungen zur Ausgabe (-a[1|2|4]).

Exportierte und kommentierte Beispiel-Daten von DCF77-Signalen, einer 433-MHz Funk-Fernbedienung sowie einer RC-5 IR Fernbedienung habe ich als Textdateien beigelegt, damit eine Vorstellung über den Output möglich ist.

Die Kommentare nach dem // sind natürlich manuell eingebaut !

Die Startmeldung (=Hilfe) habe ich auch noch "ausgedruckt", als start_msg.txt.

Dieser Logic-Analyser ist sicherlich nicht übermäßig komfortabel, dafür aber (fast) kostenlos.
Und mit der Investition von etwas Gehirnschmalz kann man eine Analyse der gewonnenen Daten in einer Tabellenkalkulation durchführen.
Oder die freien 3k Flash nutzen, um eine eigene Auswertung einzubauen.

happy analysing

Michael S.