School of Engineering
InES Institute of Embedded Systems

CT2 Praktikum

Frequenzmessung

1 Einleitung

In diesem Praktikum werden Sie eine Frequenzmessung und eine Frequenzmultiplikation implementieren und testen. Für das Praktikum benötigen Sie neben dem CT-Board einen Funktionsgenerator, um verschiedene Frequenzen zu generieren und diese auf die I/Os des CT-Boards zu geben. Zum Messen der Ausgangssignale verwenden Sie ein Oszilloskop.

2 Lernziele

- Sie können die Timer auf einem Microcontroller verwenden, um eine Frequenzmessung zu implementieren.
- Sie sind in der Lage Funktionen mit einem und mehreren Timern zu realisieren.

3 Aufbau

3.1 Material

- 1 x Funktionsgenerator
- 1 x Oszilloskop
- 2 x BNC zu 0.64mm Buchsen Kabel

3.2 Funktionsgenerator

Verbinden Sie den Ausgang des Frequenzgenerators (TTL / CMOS) mit dem GPIOA Pin9, kurz PA8 gemäss Abbildung 2. Schliessen Sie dazu das rote Kabel an PA8 (Pin 9) und das schwarze Kabel an GND (Pin 14 oder 15) entsprechend Abbildung 2.

Bevor Sie die Kabel anschliessen, stellen Sie den Funktionsgenerator wie folgt ein:

- TTL Ausgang (*Shift* + *9* drücken)
- Waveform auf Rechteck (Taste Wave)

Achtung – Zerstörungsgefahr!

Stellen Sie unbedingt sicher, dass Sie den TTL-Ausgang des Funktionsgenerators auf 5V eingestellt haben.

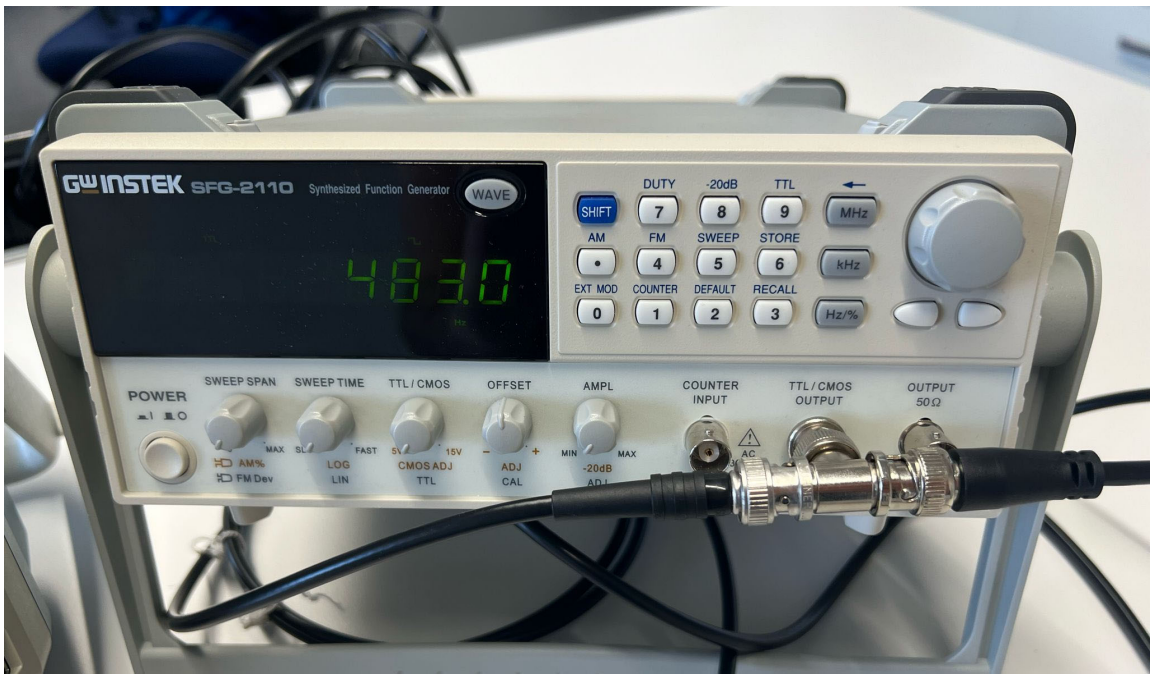

Abbildung 1: Anschluss des Funktionsgenerators

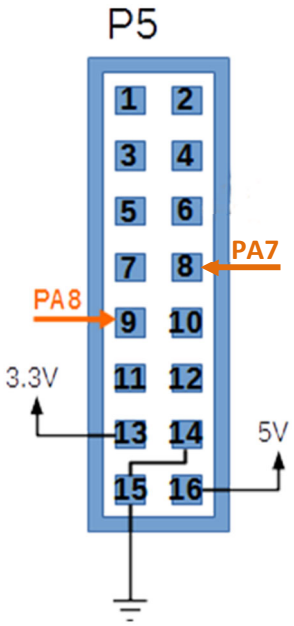

Abbildung 2: 16pin Header des CT-Boards mit den verwendeten Pins

3.3 Oszilloskop

Das Oszilloskop wird an GPIO Port PA7 (Pin 8) und an Masse (GND) angeschlossen. Das Oszilloskop wird auf 2V/div eingestellt und der Tastkopf muss auf 1X eingestellt werden.

4 Aufgaben

Verwenden Sie die Timer 1 und Timer 8 für alle Aufgaben in diesem Praktikum. Informationen zu den Timern finden Sie im ennis.zhaw.ch unter Timer. Weitere Informationen finden Sie im Referenzmanual zum STM32 Seite 515 ff.

Die Taktfrequenz beträgt auf dem CT-Board für alle Timer 84 MHz, sofern der Prescaler auf einen Teiler von 1 gesetzt ist.

4.1 Frequenzmessung

Implementieren Sie ein Programm für eine Frequenzmessung nach dem reziproken Messverfahren. Verwenden Sie den Timer 1 und geben Sie die gemessene Frequenz auf dem Display des CT -Boards aus. Verwenden Sie dazu den bereitgestellten Programmrahmen. Arbeiten Sie sich zunächst ins ennis.zhaw / Timer ein und schreiben Sie in diesem Praktikum direkt in die Register.

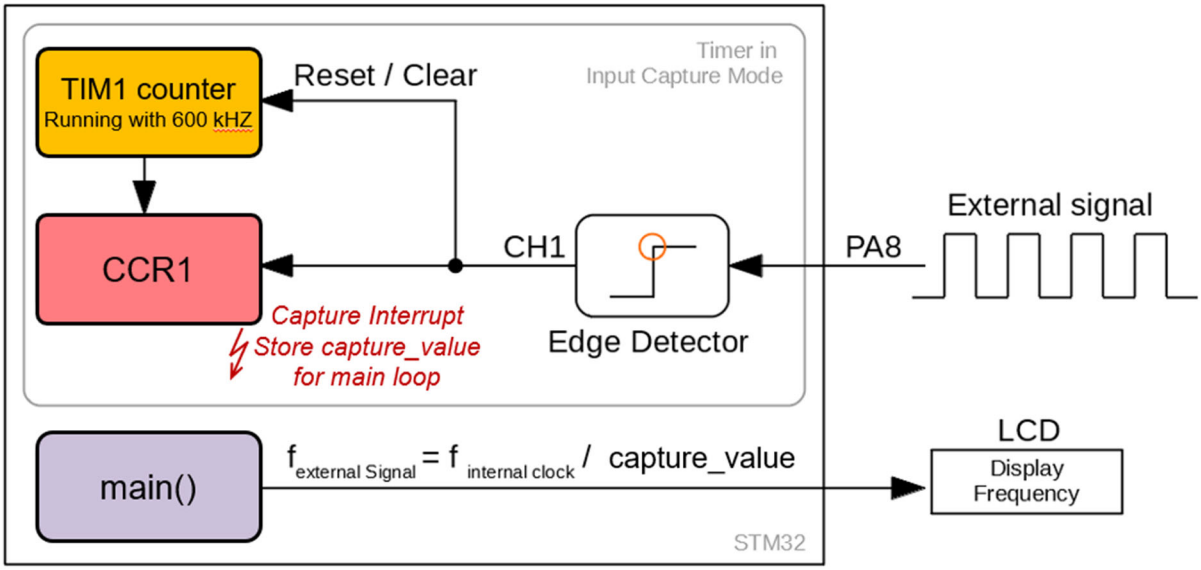

Abbildung 3: Prinzip der reziproken Frequenzmessung

Erweitern Sie die Funktion **`init_measure_timer()`** wie folgt. Initialisieren Sie Timer 1 als Upcounter, so dass er nach einem Update automatisch resetiert und neu gestartet wird (Run-Continous). Wählen Sie den Counter-Reload Wert so, dass auch langsame Frequenzen gemessen werden können. Da Timer 1 nicht im Master-Mode verwendet wird, spielt die Einstellung des Master-Modes beim Initialisieren keine Rolle.

Bei jedem Capture-Interrupt muss das Capture-Register ausgelesen und in der vordefinierten Variable **`capture_value`** gespeichert werden. Implementieren Sie dazu die Interrupt Service Routine (ISR) des Timer 1 **`TIM1_CC_IRQHandler()`.**

Die Interrupts müssen Sie aktivieren, damit diese dann ihre Interrupt Service Routine aufrufen. Während der Programmlaufzeit können Interrupts mittels Register auch deaktiviert werden.

NVIC->ISER[.]

Setzen Sie vor Austritt aus der ISR das entsprechende Interrupt-Flag im Register

TIM1->SR zurück, da Sie sonst in der Routine „gefangen“ bleiben.

Über die Funktion **`get_capture_value()`** wird der aktuelle Capture-Wert vom Hauptprogram angefordert. Berechnen Sie aus dem Capture-Wert die Frequenz und zeigen diese auf dem Display mit Hilfe der Funktion **`write_display()`** an.

- Bedenken Sie, dass Sie mit der reziproken Frequenzmessung maximal die halbe Frequenz messen können, welche als Takt am Counter anliegt (Abtasttheorem). Bei hohen Frequenzen weist das Capture Register nur noch kleine Werte auf, wodurch die Messgenauigkeit abnimmt.

4.2 Frequenzmultiplikation

Erweitern Sie das Programm zur reziproken Frequenzmessung, so dass ein Vielfaches der gemessenen Frequenz an GPIO PA7 ausgegeben wird. Verwenden Sie dazu Timer 8 als Output-Compare-Timer. Erweitern Sie die Funktion **`init_frequency_multiplier()`** zur Initialisierung des Timers 8. Konfigurieren Sie den Timer 8 in Output-Compare-Mode, so dass das Ausgangssignal auf CH1N immer getoggelt wird, wenn der Counter den Wert 0 hat.

Modifizieren Sie die ISR des Timers 1 aus Aufgabe 4.1 so, dass der Wert des Capture-Registers von Timer 1 in das Reload-Register von Timer 8 kopiert wird. Die Multiplikation der Eingangsfrequenz wird mit Hilfe der Dip-Switches auf dem CT-Board realisiert. Der Multiplikator soll an den Dip Switches S3-S0 auf Werte zwischen 1 und 15 eingestellt werden können. Schreiben sie dazu eine Funktion, die den Multiplikator ausliest. Schreiben Sie den Multiplikation Faktor im main loop in die Funktion **`write_display()`**. Zusätzlich muss der Prescaler des Timer 8 in Abhängigkeit des Multiplikators eingestellt werden können. Benutzen und implementieren Sie dazu die Funktion **`set_prescaler_freq_mul(uint8_t dip_switch_value).`**

Das Prinzip der Frequenzmultiplikation wurde in der Vorlesung besprochen. Sie finden es nochmals in Abbildung 4.

Hier müssen Sie einen Kompromiss mit dem Prescaler vom Timer 1 eingehen. Ist er zu hoch, wird die Frequenzmessung schnell ungenau ausfallen. Ist der Prescaler jedoch zu klein, kann der Prescaler vom Timer 8 für die multiplizierte Frequenz nicht genau eingestellt werden.

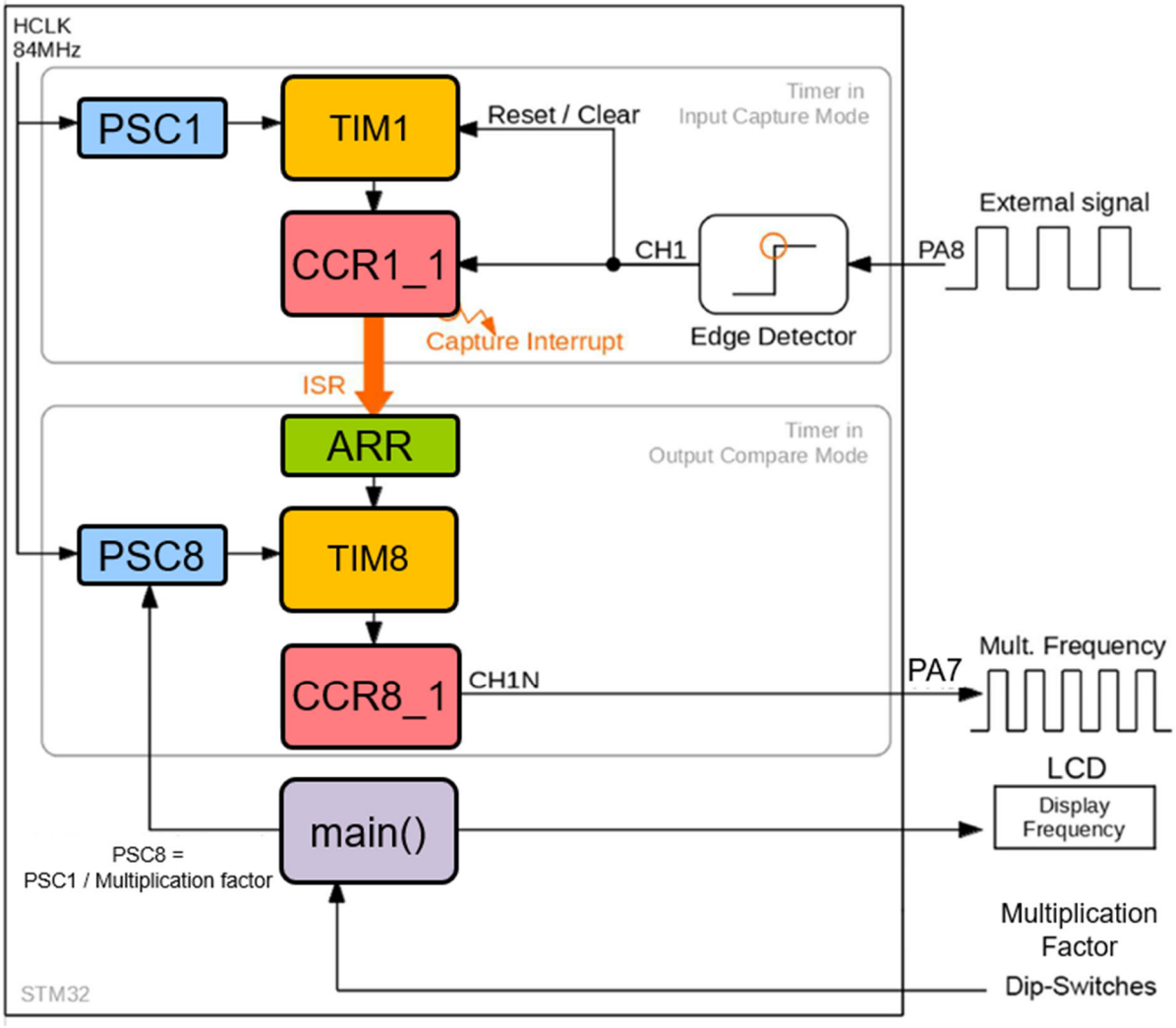

Abbildung 4: Prinzip der Frequenzmultiplikation

Testen Sie Ihr Programm mit verschiedenen Frequenzen und unterschiedlichen Dip-Switchschalterstellungen. Schliessen Sie das Oszilloskop an das CT -Board an und visualisieren Sie das ausgegebene Signal mit dem Oszilloskop.

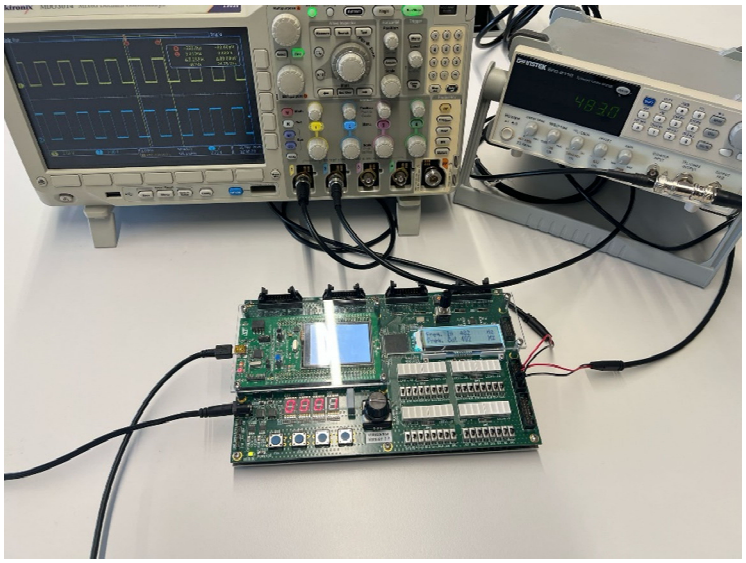

Abbildung 5: Aufbau für die Frequenzmultiplikation

5 Bewertung

Das Praktikum wird mit maximal 3 Punkten bewertet:

- Aufgabe 4.1 1 Punkte
- Aufgabe 4.2 2 Punkt

Punkte werden nur gutgeschrieben, wenn die folgenden Bedingungen erfüllt sind:

- Der Code muss sauber, strukturiert und kommentiert sein.
- Das Programm ist softwaretechnisch sauber aufgebaut.
- Die Funktion des Programmes wird erfolgreich vorgeführt.
- Der/die Studierende muss den Code erklären und zugehörige Fragen beantworten können.