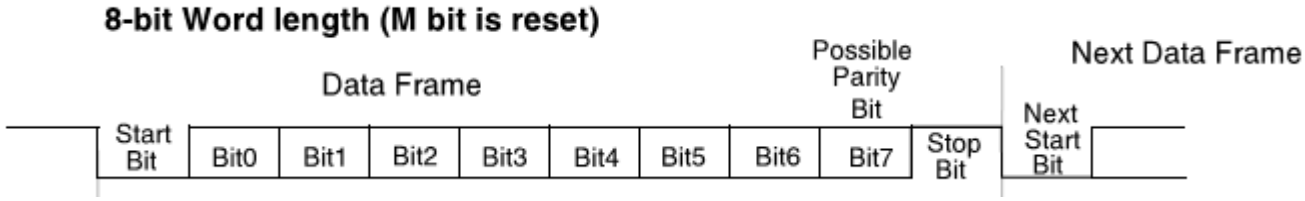
Versuch 2: Die serielle Schnittstelle (USART)

In diesem Versuch wird die serielle Schnittstelle verwendet, um ASCII-Daten zwischen dem Mikrocontroller und einem Terminal (putty; Einstellungen siehe Anhang 1) zu übertragen. Diese Übertragung wird als V.24 bzw. RS232 Schnittstelle bezeichnet.

Kurze Einführung zur seriellen Schnittstelle:
Es werden 8 Datenbit sequentiell auf einer Leitung übertragen, wobei kein Takt übertragen wird. Sender und Empfänger müssen daher auf die gleiche Taktrate eingestellt werden. Damit der Beginn einer Datenübertragung vom Empfänger erkannt werden kann, werden die 8 Datenbit mit einem Start-Bit und einem Stop-Bit versehen (= Frame). Das Start-Bit hat einen Low-Pegel, das Stopp-Bit und der Idle-Zustand werden mit einem High-Pegel belegt. Damit entsteht beim Beginn einer Datenübertragung beim Übergang vom Idle-Zustand zum Start-Bit bzw. vom Stopp-Bit zum Start-Bit immer eine negative Flanke, die den Empfänger triggert. Das folgende Bild zeigt die Übertragung von 8 Bit ohne Parity-Bit.



Die Übertragungsgeschwindigkeit wird als Baud-Rate bezeichnet mit der Einheit Bit/sec. Eine typische Baudrate sind 9600 Baud d.h. 9600 Bit/sec, bei 10 Bit pro Zeichen dauert die Übertragung eines Zeichens dabei ca. 1 ms. Wenn also mehrere Zeichen hintereinander gesendet werden, darf erst ein neues Zeichen in das **Datenregister** (Transmitregister) des USART-Bausteins geschrieben werden, wenn das Transmit-Register wieder leer ist. Der Baustein besitzt daher neben dem Datenregister ein **Statusregister**, in dem ein Bit (TXE = Transmitregister empty) gesetzt wird, sobald ein neuer Wert geschrieben werden kann.
Ein USART-Baustein kann senden und empfangen, ein weiteres Bit im Statusregister (RXNE = Receiveregister not empty) zeigt an, dass ein Datenpaket mit 8-Bit empfangen wurde und aus dem Datenregister gelesen werden kann.

Der Baustein hat weitere **Controlregister**, um Übertragungsmodi und die Baudrate einzustellen.

Aufgabe 2.1: Sinusausgabe
Im Hauptprogramm sollen in einer Endlosschleife mit Hilfe des Digital-Analog-Wandlers (DAC) die Werte einer Sinuskurve ausgegeben und am Oszilloskop sichtbar gemacht werden. Die Sinuskurve wird in 32 Schritten in einem maximalen Wertebereich von 0 bis 4095 dargestellt. Zur Ausgabe der Analogwerte sollen die vorgegebenen Funktionen `DACInit` und `DACWrite` verwendet werden. Binden Sie dazu die Datei DAC.c in Ihr Projekt ein. Ergänzen Sie in der `main`-Funktion die Sinusausgabe, am Oszilloskop sollte dann eine stufige Sinuskurve sichtbar werden.

Wir verwenden einen 12-Bit D/A-Konverter, d.h. es können Werte von 0 bis 4095 verwendet werden. Nachdem der D/A-Konverter des STM32 in den Randbereichen in die Sättigung geht, soll der Bereich nur zu ca. 90% ausgenutzt werden (2048 +/- 1850).
Bereiten Sie eine entsprechende Tabelle mit 32 Sinuswerten vor.

Aufgabe 2.2: String ausgeben
Die USB-Schnittstelle des Nucleo-Boards ist fest mit USART2 verbunden. Kopieren Sie die beiden Dateien USART2.c und USART2.h in Ihren Projektordner von Versuch 1.3 und binden Sie die C-Datei in das Projekt ein (rechter Mausklick auf Source Group / Add Existing Files. Die Header-Datei binden Sie über eine `#include`-Anweisung in `main.c` ein. Ergänzen Sie das Programm so, dass einen String auf der Terminalemulation putty ausgegeben wird (Einstellungen für Putty im Anhang 1). Vervollständigen Sie dazu die Funktion **`USART2Init(void)`** in `USART2.C`, die die entsprechenden Einstellungen (9600 Baud, Oversampling16, 8 Datenbits, 1 Stopp-Bit, kein Parity, Übertragungsrichtung, ohne Hardware Flow Control) vornimmt. Ergänzen Sie dann die Funktion **`USART2WriteChar(char),`** die ein einzelnes Zeichen ausgibt. Beachten Sie dabei, dass Sie solange aktiv warten müssen (das TXE-Bit abfragen), bis das Transmitregister frei ist und erst dann Ihr Zeichen senden können. Verwenden Sie anschließend diese Funktion, um eine Funktion **`USART2WriteString(char *str)`** zu realisieren, die einen kompletten String auf dem Terminal ausgibt. Die `USART2WriteString`-Funktion kann zum Testen z.B. zu Beginn des Hauptprogramms in `main` dazu genutzt werden, um eine Begrüßungsnachricht zu senden. Um in der Ausgabe eine neue Zeile zu beginnen, können Sie die Zeichenkombination `\n\r` an Ihren Text anhängen.

Aufgabe 2.3: Interruptgesteuertes Echo
Im Folgenden wird nun die USART-Schnittstelle im Interruptmodus verwendet, um Zeichen einzulesen. Die USART-Schnittstelle kann so konfiguriert werden, dass nach jedem empfangenen Zeichen ein Interrupt ausgelöst wird. Ergänzen Sie in der Datei `USART.c` die Interrupt-Service-Routine, in der ein Zeichen eingelesen und als Echo (mittels der Funktion `WriteChar`) wieder ausgegeben wird, so dass der Bediener die Rückmeldung erhält, welche Taste er gedrückt hat.

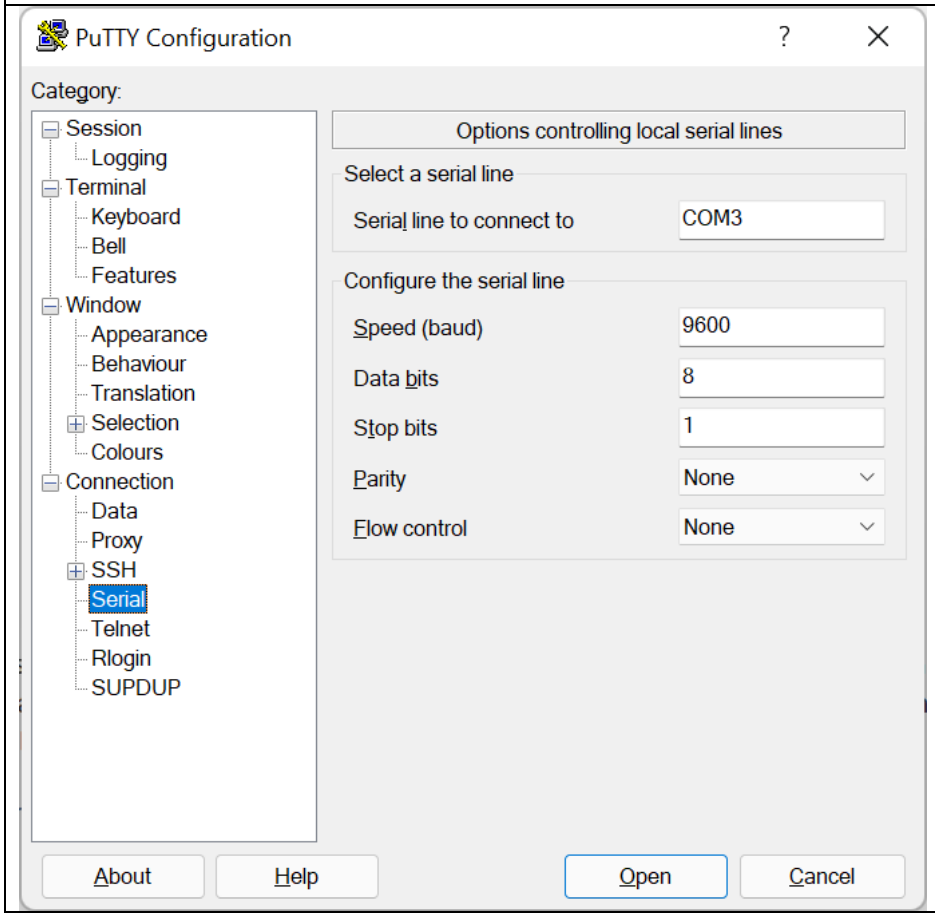
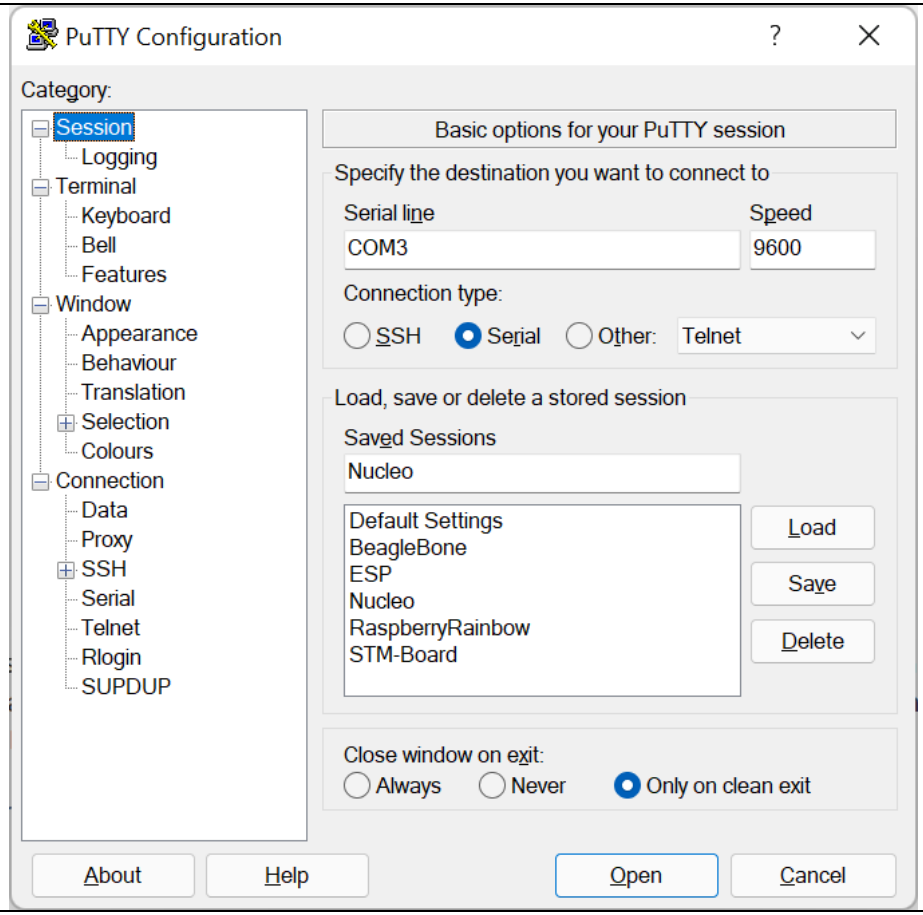
Die Interrupt-Service-Routine muss mit `void USART2_IRQHandler(void)` definiert sein. Sie wird nie in Ihrem Programm direkt aufgerufen, sondern automatisch beim Auslösen der Interrupts vom Mikrocontroller über die Vektortabelle.

Aufgabe 2.4: Interruptgesteuerte Befehlseingabe
Jetzt soll ein String bis zu einem definierten Endezeichen (`\r` entspricht Enter-Taste) per Interrupt eingelesen und in einem globalen Puffer abgelegt werden. Der String kann als Befehl verstanden werden, der die Frequenz und die Amplitude der Sinuskurve ändert. Während der Eingabe wird im Hauptprogramm weiterhin das Sinussignal ausgegeben. Der eingegebene Befehl gibt einen Delay in Millisekunden (0...10 ms) und einen Wert zur Amplitudenteilung (1…9) an und sieht z.B. wie folgt aus: `d3a9` (Delay 3, Amplitudenteiler 9). Der Delay definiert die Wartezeit zwischen den einzelnen Werten der Sinusausgabe; Sie können dazu die vorgegebene Funktion `void DelayMs(int ms)` verwenden. Die Sinuswerte sollen durch die Amplitudenteilung geteilt werden; bei 1 hat man die volle Amplitude, bei 9 die kleinste.

Die Auswertung des Befehls sollte wie folgt umgesetzt werden: In der Interrupt-Serviceroutine wird beim Erreichen des Endezeichens ein globales Kommandoflag gesetzt, das in der `while`-Schleife des Hauptprogramms zyklisch abgefragt wird. Wenn dieses Flag gesetzt ist, wird das Flag wieder zurückgesetzt und die in main.c zu definierende Funktion `void ExecCmd(void)` aufgerufen, in der dann über eine switch-case-Anweisung des ersten Zeichens des Eingabepuffers die möglichen Befehle ausgeführt werden sollen (in diesem Fall nur d, es kommen in den nächsten Versuchen aber noch mehr dazu). Dort analysieren Sie den String und ändern Delay und Amplitudenteilung. Die Änderungen sollen mit dem Oszilloskop überprüft werden.

Hinweise:
- In der Interruptroutine muss jedes eingegebene Zeichen nacheinander zu einem String aufgebaut werden. Es muss zusätzlich überprüft werden, ob das eingelesene Zeichen das Endezeichen ist.
- `sscanf` eignet sich bestens dazu, um Zahlen aus einem ASCII-String herauszulesen. Informieren Sie sich dazu über die Funktion `sscanf`.
- Um den String mit der Funktion `sscanf` aus der C-Standardlibrary analysieren zu können, muss er mit `\0` (NULL) abgeschlossen werden.

Anhang 1: Verbindungseinstellungen mit Putty
Nach dem Öffnen von Putty müssen folgende Einstellungen vorgenommen werden:

Maske Connection/Serial:	Maske Session:
	

Die Nummer der COM-Schnittstelle (z.B. COM3 oder COM4) ermitteln Sie über den Gerätemanager, Anschlüsse (COM & LPT) neben dem Eintrag „STMicroelectronics STLink Virtual COM Port (Comx).