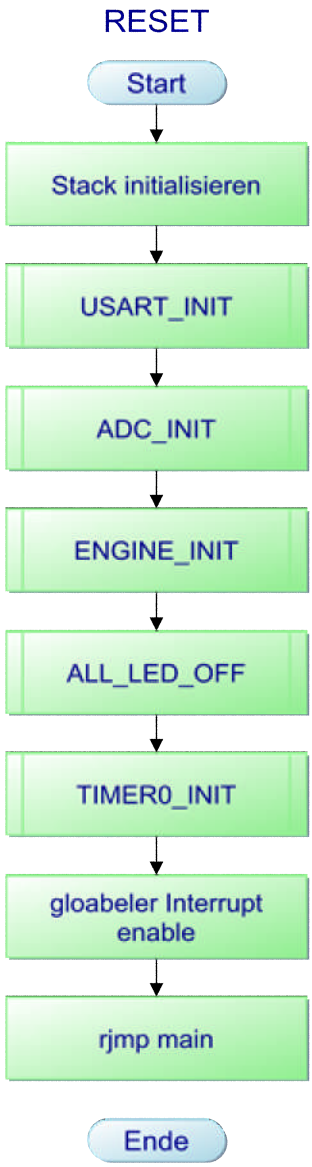
Jürgen Döffinger Michael Goldbuch	**Mikroprozessortechnik** **Beleg** ASURO INSTANT MESSENGER	

Inhaltsverzeichnis:

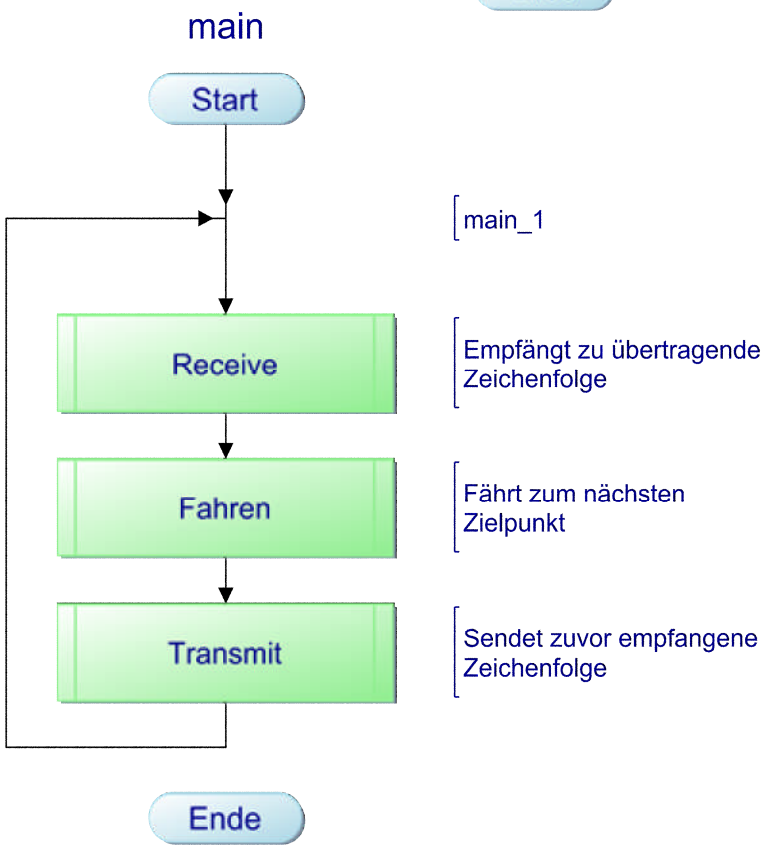
Aufgabe:
Die von uns selbst gestellte und von Professorenhand gebilligte Aufgabe soll folgende Ausmaße annehmen. Es soll die Kommunikation zwischen zwei Computer benutzenden Personen hergestellt werden. Der Ablauf wird wie folgt aussehen: Person #1 schreibt eine Nachricht im Kommunikationsprogramm. Diese Nachricht wird auf einen wartenden Asuro übermittelt. Der Asuro fährt in Folge dessen eine Linie ab und übergibt die Nachricht an einen zweiten Asuro. Dieser folgt ebenfalls einer Linie bis er an einen zweiten Kommunikationsübertragungspunkt gelangt und die Nachricht an den Computer #2 und somit Person #2 übergeben kann. Person #2, soweit gewillt, kann nun Antworten und eine Nachricht auf gleichem Weg zurück senden. Person #1 wird die Antwort erhalten und kann den Kreislauf von neuem starten.

Durchführung:

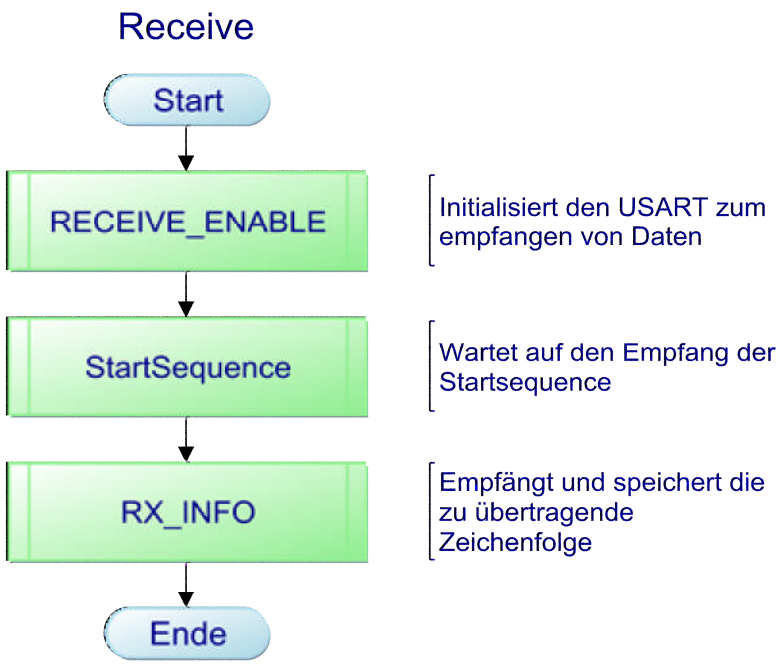
Zu Beginn wird in das Unterprogramm RESET gesprungen. In diesem wird die Startadresse des Stacks festgelegt, der USART (serielle Datenübertragung), der ADC (Analog-Digital-Converter) und die PWM für die Steuerung der Motorgeschwindigkeit initialisiert und die LEDs abgeschalten, bis auf die Status-LED. Zum Schluss wird der globale Interrupt aktiviert und in das Unterprogramm MAIN gesprungen, welches als Hauptprogramm dient.



Die gestellte Aufgabe lässt sich für die Asuros in drei Einzelschritte unterteilen. Im ersten Schritt wartet der Asuro auf die Nachricht. Er empfängt diese und speichert sie ab. Ist die Nachricht in Gänze gespeichert beginnt Schritt 2 und er fährt eine Linie bis zu einem vorgegebenen Stopppunkt ab. Im dritten Schritt übergibt er die abgespeicherte Nachricht. Ist diese gesendet sind alle Schritte einmal durchlaufen und der PC springt wieder soweit zurück, dass der Asuro auf das Empfangen einer Nachricht wartet.



Der erste Schritt (das Empfangen) lässt sich wiederum in drei Schritte unterteilen. Es beginnt mit der Einstellung des USART auf Empfangen. Dazu müssen, im UCR, die Bits 7, 6 und 3 auf 0 gesetzt werden und das Bit 4 auf 1. Heißt: Übertragung aus, Interrupt für die Benachrichtigung empfangener und gesendeter Daten aus und Empfangen ein.
Danach geht er in StartSequence über.

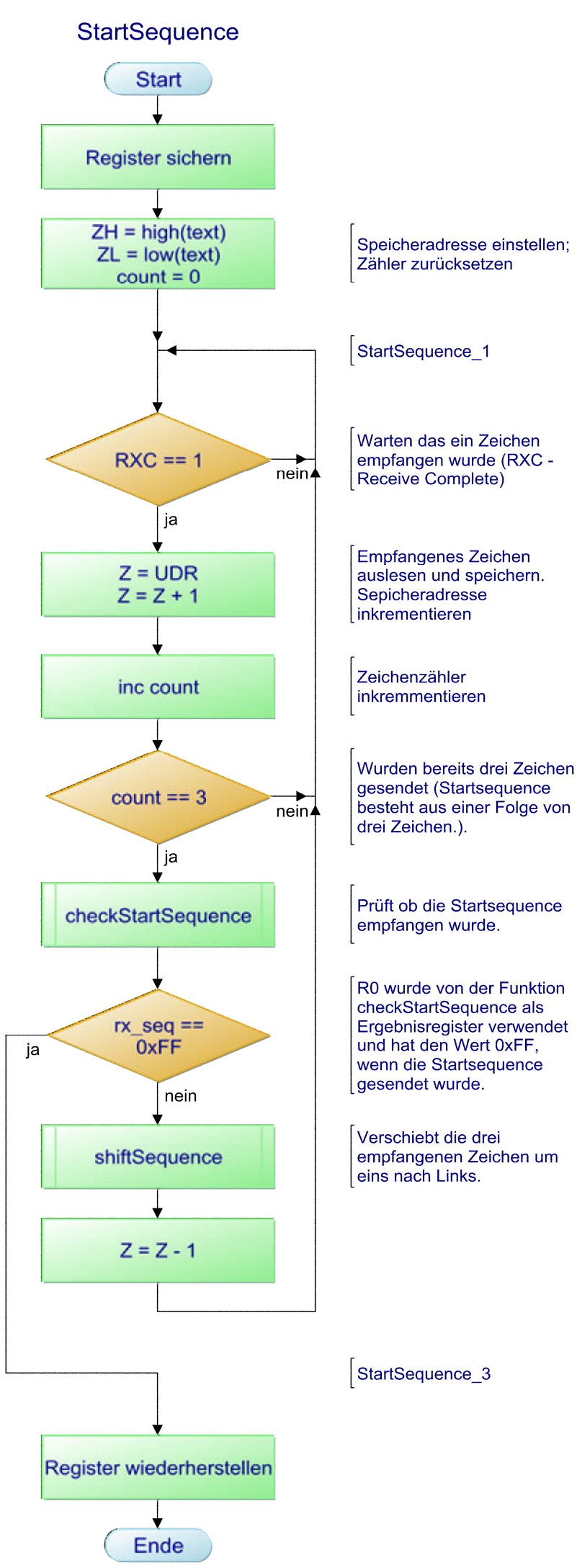


Die StartSequence beginnt mit dem sichern des Registers R16 und des Z-Pointers. Nachdem der Zähler auf Null gesetzt wurde und die Adressen zugewiesen wurden beginnt StartSequence_1.
Hier wird zuerst überprüft ob das Bit 7 des UCSRA gesetzt ist. Ist es das nicht und RXC enthält eine 0 ist noch kein Zeichen empfangen worden. In diesem Fall springt der PC wieder zurück zu StartSequence_1 und hängt somit in einer Warteschleife bis ein Zeichen empfangen wurde.

Ist ein Zeichen empfangen worden wird der Inhalt des Empfangsregisters gespeichert und die Speicheradresse für das nächste Zeichen weiter geschoben. Nun wird überprüft ob schon 3 Zeichen gesendet wurden. Dies dient dem Zweck der Überprüfung ob es sich bei den drei Zeichen um die Startsequenz handelt.

(checkStartSequence wird auf der nächsten Seite beschrieben)

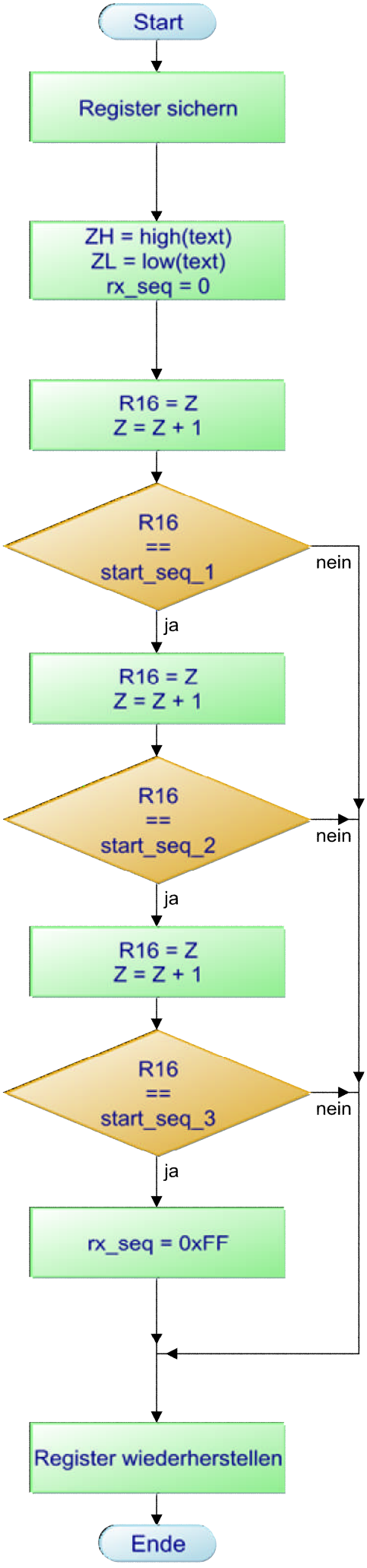
Handelt es sich bei den geprüften drei Zeichen um die Startsequenz werden die Register wieder hergestellt und es geht mit RX_INFO weiter.
Sollte es sich nicht um die Startsequenz gehandelt haben geht es mit der Funktion shiftSequence weiter.



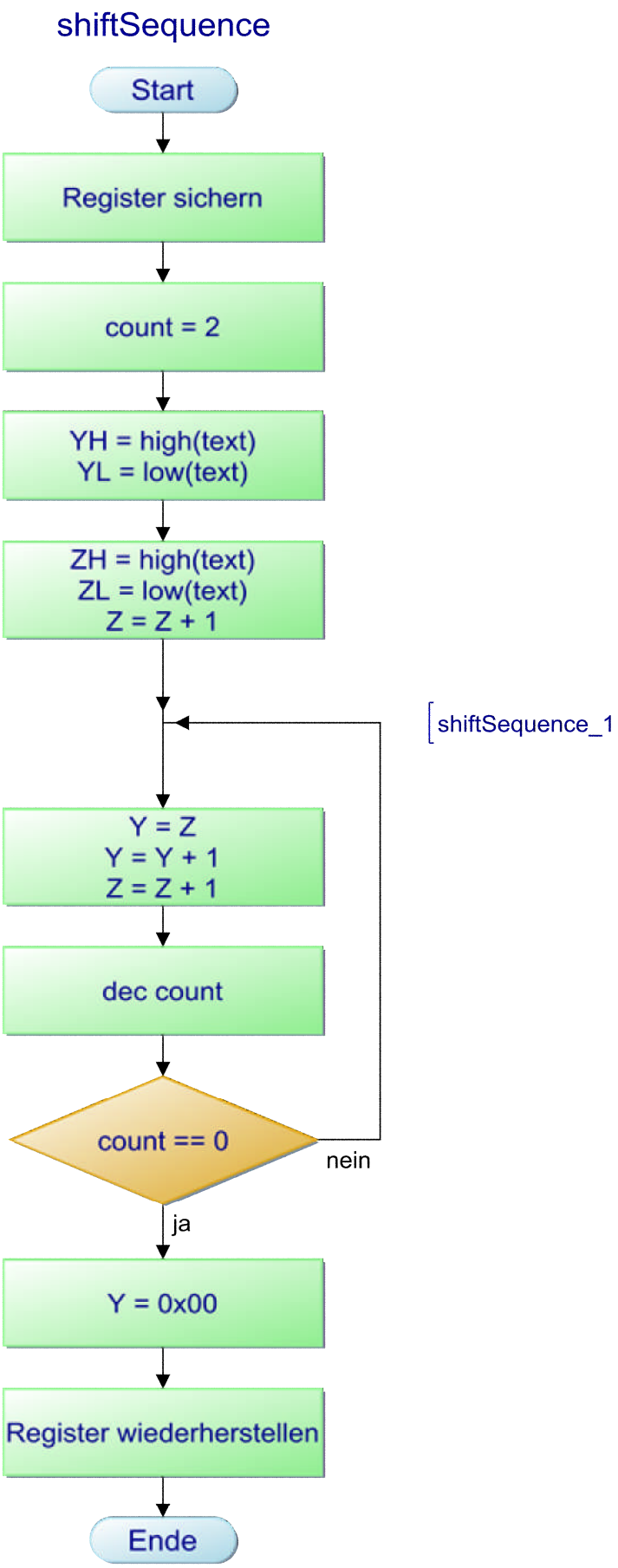
In checkStartSequence wird nach der Registersicherung und der Zuweißung auf die Adressen das Startsequenzergebnisregister (rx-seq) gelöscht.

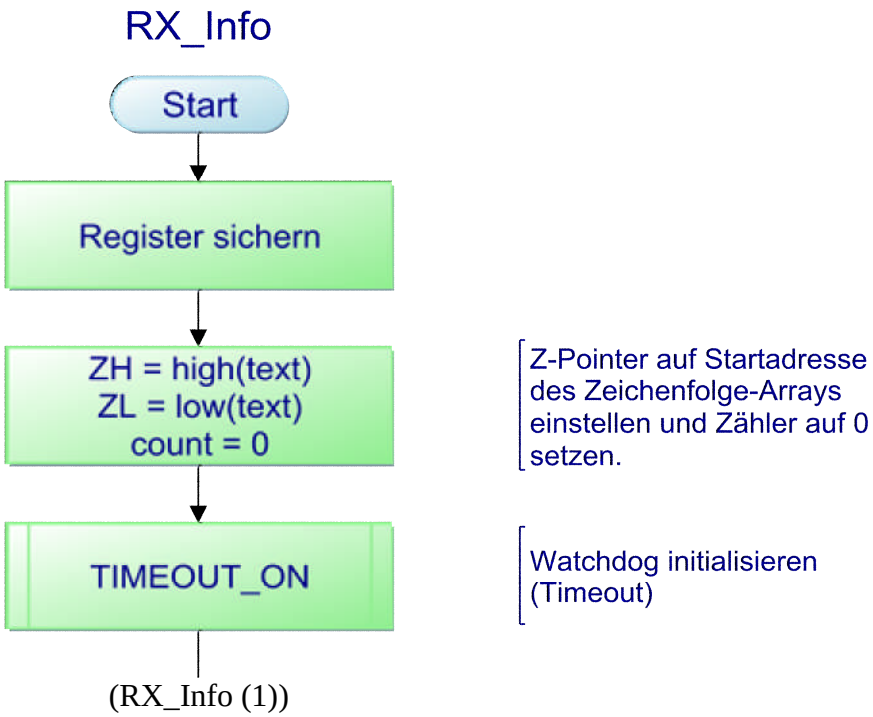
In Folge wird das erste empfangene Zeichen in R16 geladen und mit dem ersten Zeichen der Startsequenz verglichen. Sollte dies stimmen wird das zweite empfangene Zeichen in R16 geladen und mit dem zweiten Zeichen der Startsequenz verglichen. Sollte auch diesen stimmen wird mit dem dritten Zeichen genauso vorgegangen.
Sollten es sich bei den drei Zeichen um die Startsequenz handeln wird rx_seq auf 0xFF (255) gesetzt um zu zeigen das die Startsequenz empfangen wurde.
Danach werden die Register wieder hergestellt und es wird in StartSequence zur Überprüfung zurück gesprungen.

Sobald eines der empfangenen Zeichen bei der Überprüfung in checkStartSequence nicht mit dem jeweiligen Zeichen der Startsequent übereinstimmen springt der PC direkt wieder auf die Registerwiederherstellung und den Rücksprung.



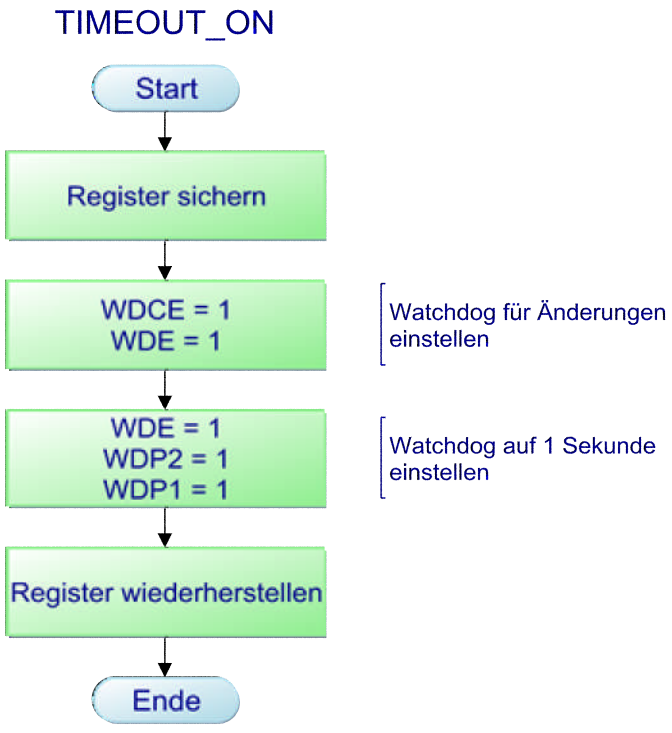
Das Unterprogramm shiftSequence sorgt dafür, dass nach nichtempfangener Startsequence, die empfangenen Zeichen im Array einmal nach “links” geschoben werden. Das heißt, dass das erste Zeichen im Array gelöscht wird und das zweite Zeichen an die Stelle des Ersten geschrieben wird. Danach wird das dritte Zeichen an die Stelle des zweiten Zeichens geschrieben und an die Stelle des dritten zeichens wird eine 0 geschrieben. Somit wird sichergestellt, dass die Startsequence auch dann noch empfangen wird, wenn vorher andere Zeichen gesendet wurden. Nachteil liegt darin, wenn eine Zeichenfolge empfangen wird, welche die Startsequence beinhaltet, aber noicht für den ASURO gedacht ist. Abgefangen wird dies durch das Timeout in RX_INFO, solange die Zeichenkette nicht mehr als 250 Zeichen beinhaltet.





Auch in RX_Info werden zunächst wieder die Register gesichert und die Zeiger auf die ursprünglichen Adressen des Zeichenfolge-Arrays gelegt. Nachdem auch der Zähler wieder gelöscht wurde wird nun TIMEOUT_ON aufgerufen:

In TIMEOUT_ON wird nach der Registersicherung Prescaler auf etwa 1s eingestellt. Danach werden die Register wieder hergestellt und es erfolgt der Rücksprung in die aufrufende Funktion.

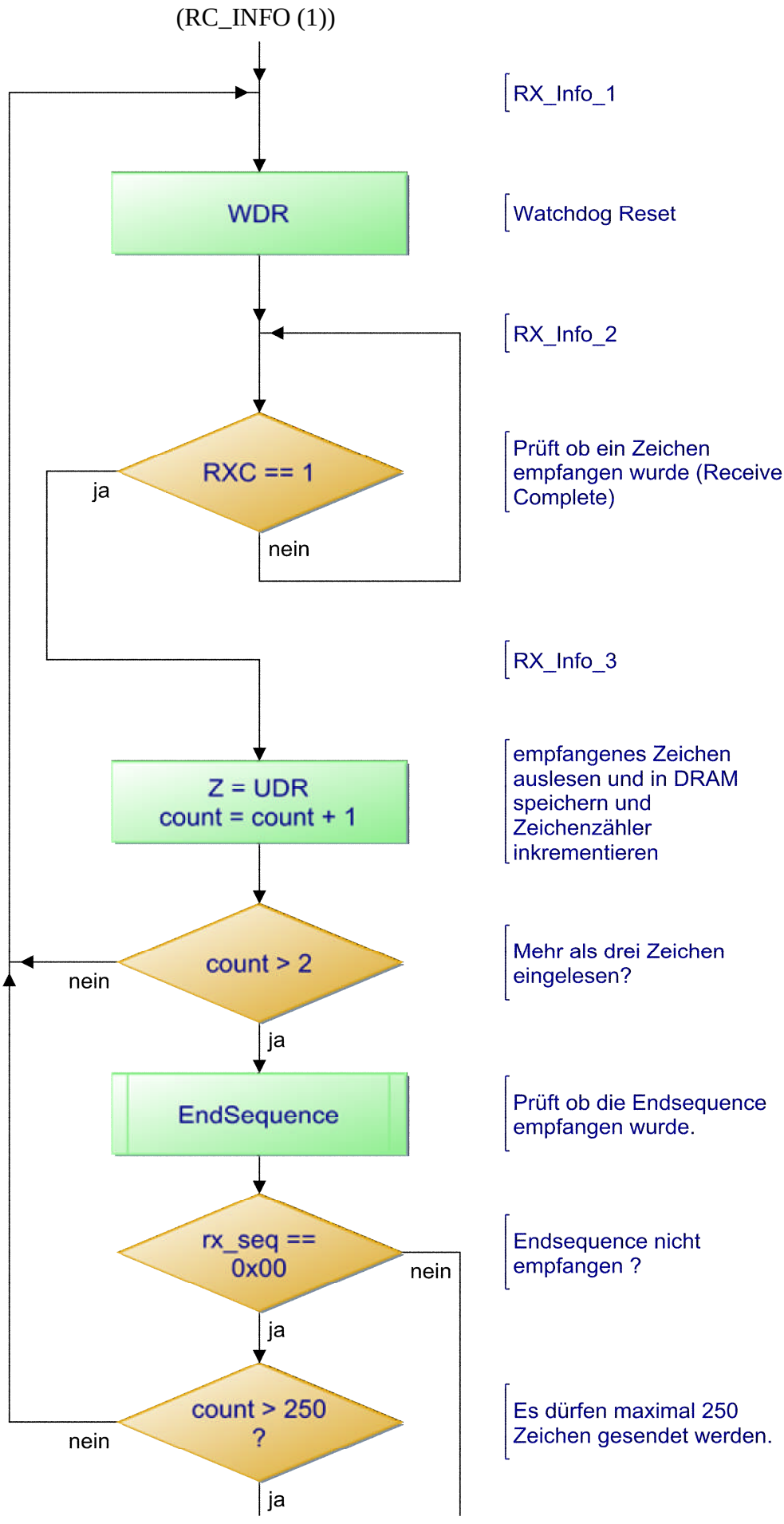


Nach dem initialisieren des Watchdogs wird der Watchdog resetet. Dies erfolgt auch immer dann, wenn ein Zeichen empfangen wurde.

Nun wird (mit Hilfe einer Schleife) gewartet bis ein Zeichen empfangen wurde.

Sobald ein Zeichen empfangen wurde wird dieses im Array an entsprechender Position (Z-Pointer) abgespeichert.
Ebenso wird überprüft ob schon drei Zeichen empfangen wurden. Ist dies der Fall, so wird geprüft ob es sich hierbei um die Endsequenz handelt.

Sollte es sich bei den letzten drei empfangenen Zeichen nicht um die Endsequenz gehandelt haben wird überprüft ob schon mehr als die erlaubten 250 Zeichen übertragen wurden. Sollte dies nicht der Fall sein wird wieder zu RX_Info_1 gesprungen.

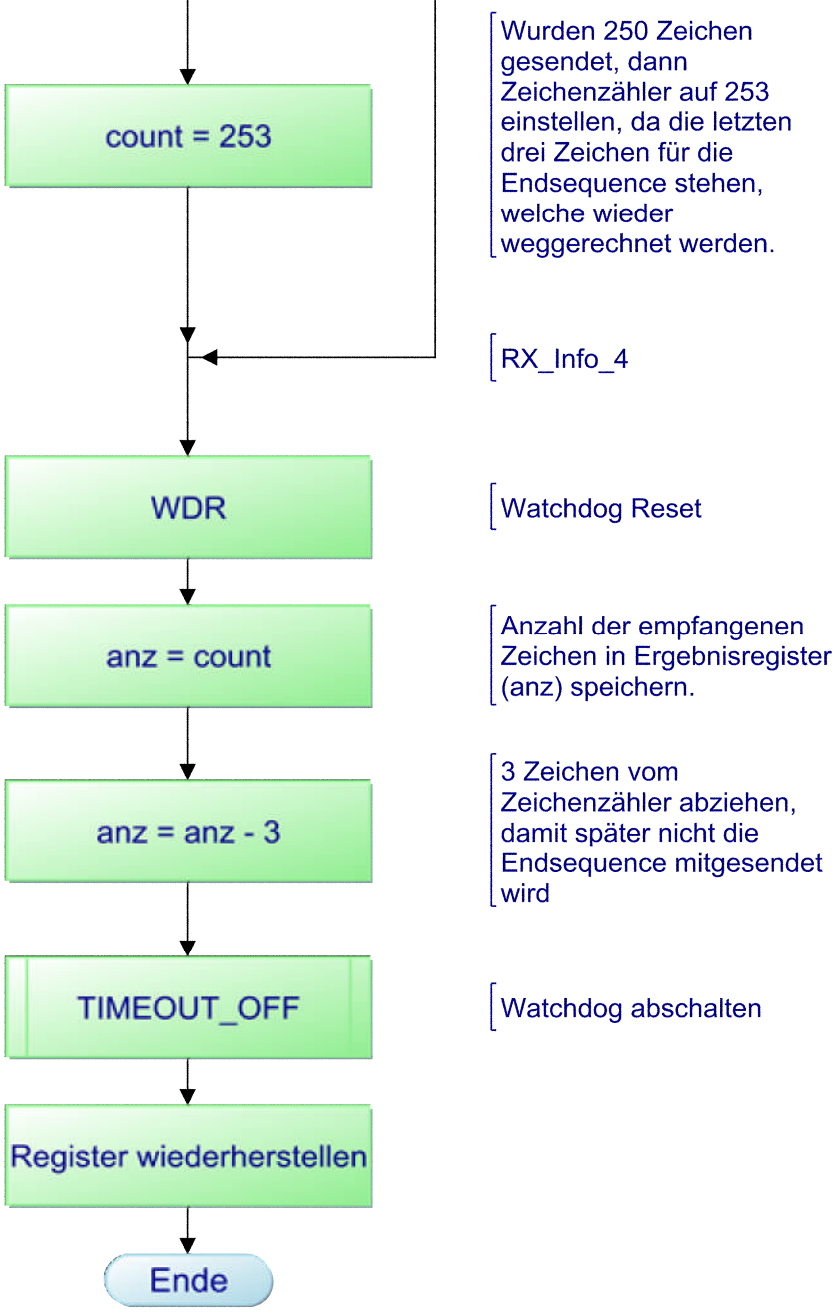


Sollten schon 250 Zeichen übertragen worden sein, dann wird der Zähler auf 253 gesetzt.

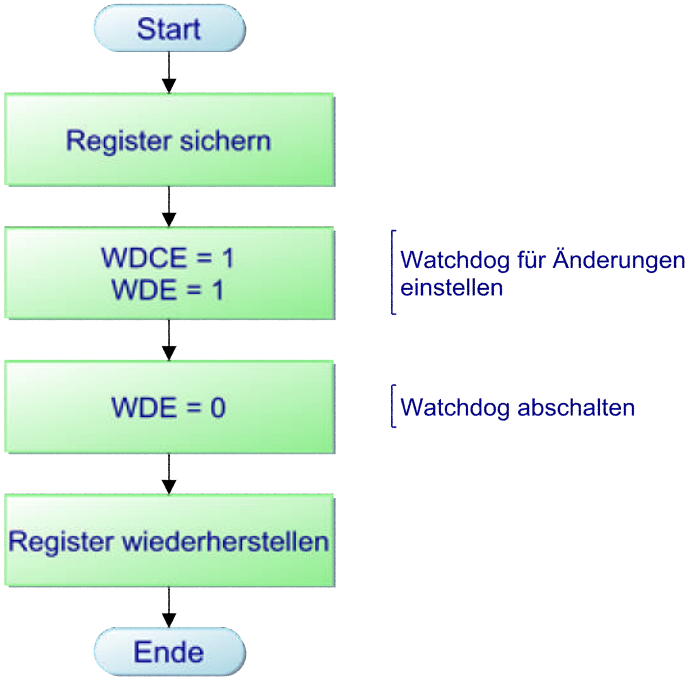
An dieser Stelle ist die Übertragung beendet und der Watchdog erfährt einen Reset.

Nachdem die Anzahl der empfangenen Zeichen abgespeichert wurden, wird noch die Anzahl der Zeichen der Endsequenz (3) abgezogen. Aus diesem Grund erfolgte auch die vorhergehende Erhöhung von 250 auf 253 im Fall des Erreichens der max. Anzahl.

Nun wird TIMEOUT_OFF aktiviert:



TIMEOUT_OFF

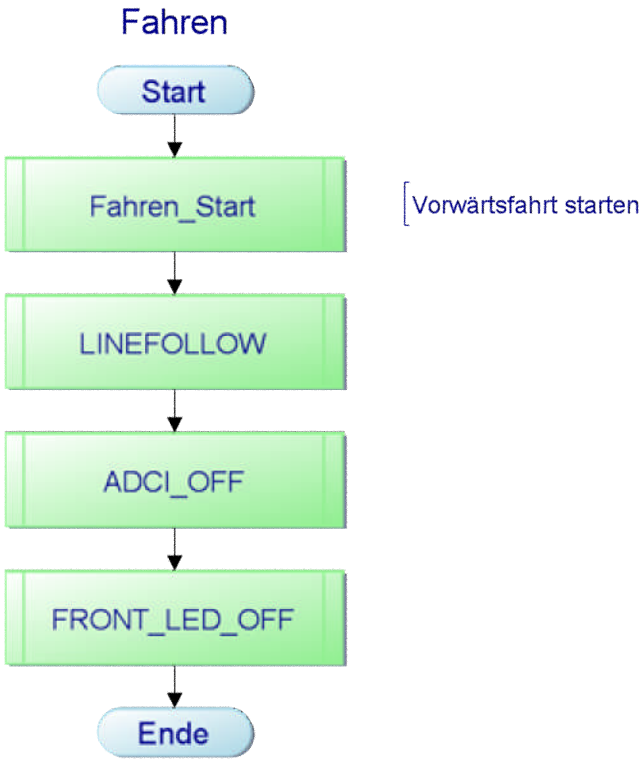


Timeout_OFF sichert zunächst die verwendeten Register und aktiviert den WDCE, womit im Anschluss das setzen des WDE auf 0 ermöglicht wird. Durch das setzen des WDE auf 0 wird der Watchdog abgeschaltet. Am Schluss werden die gesicherten Register wieder hergestellt und es erfolgt der Rücksprung in die aufrufende Funktion.

Nachdem die Nachricht komplett aufgenommen und gespeichert wurde geht es mit dem Unterprogramm Fahren weiter.

Der Asuro beginnt zu fahren. Eine Linienverfolgung findet im ersten Schritt noch nicht statt. Der Grund hierfür ist, dass der Asuro zuerst über die Stopp-LEDs (am Boden) hinweg fahren muss um nicht direkt wieder von ihnen gestoppt zu werden.

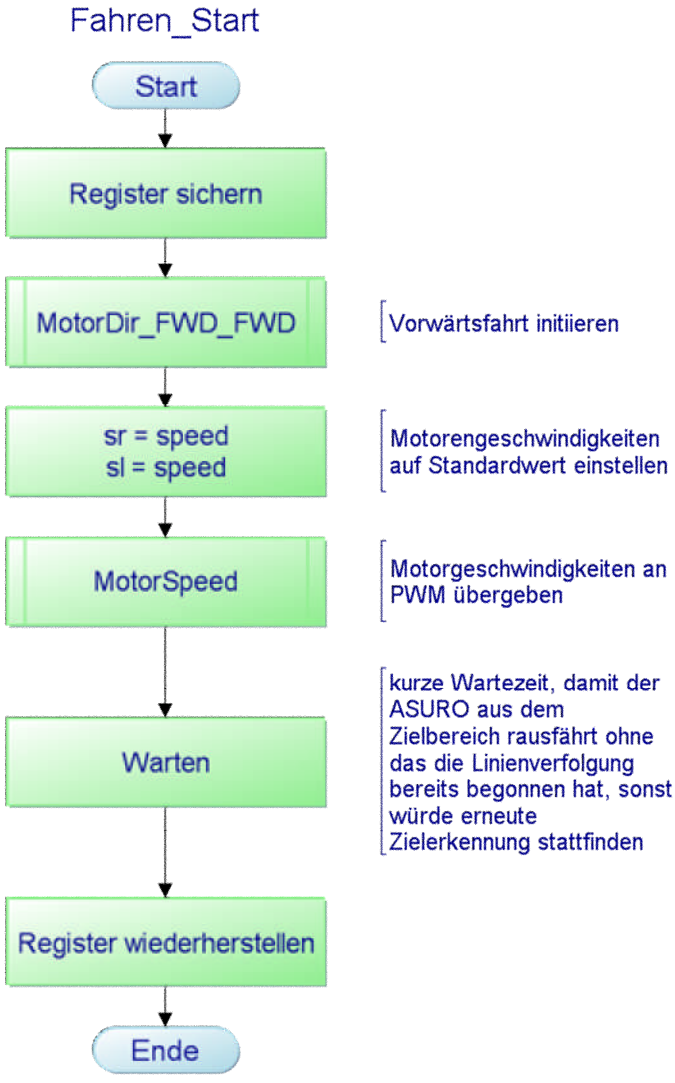
Erst jetzt erfolgt die Fahrt mit Linienverfolgung.
Des weiteren ist hier die Messung eingebunden, genauso wie das ausschalten der LED.



In Fahren_Start werden zuerst die verwendeten Register gesichert.
Nach der Initialisierung der Vorwärtsfahrt, welche aus dem 1 setzen des Bits 5 und dem 0 setzen des Bits 4 von PortB und D besteht, wird die Geschwindigkeit der Motoren auf den zu Beginn definierten Standartwert gesetzt.

Im Unterprogramm MotorSpeed werden die hinteren LEDs (Bremslichter) ausgeschalten und Geschwindigkeits-werte werden in den OCR1A/B geschrieben, welcher hier als Pulsweiten-modulierer genutzt wird.

Nun wird ein Warten aktiviert, welches die weiter oben oder rechts beschriebene Funktion erfüllt. Danach werden die gesicherten Register wieder hergestellt.



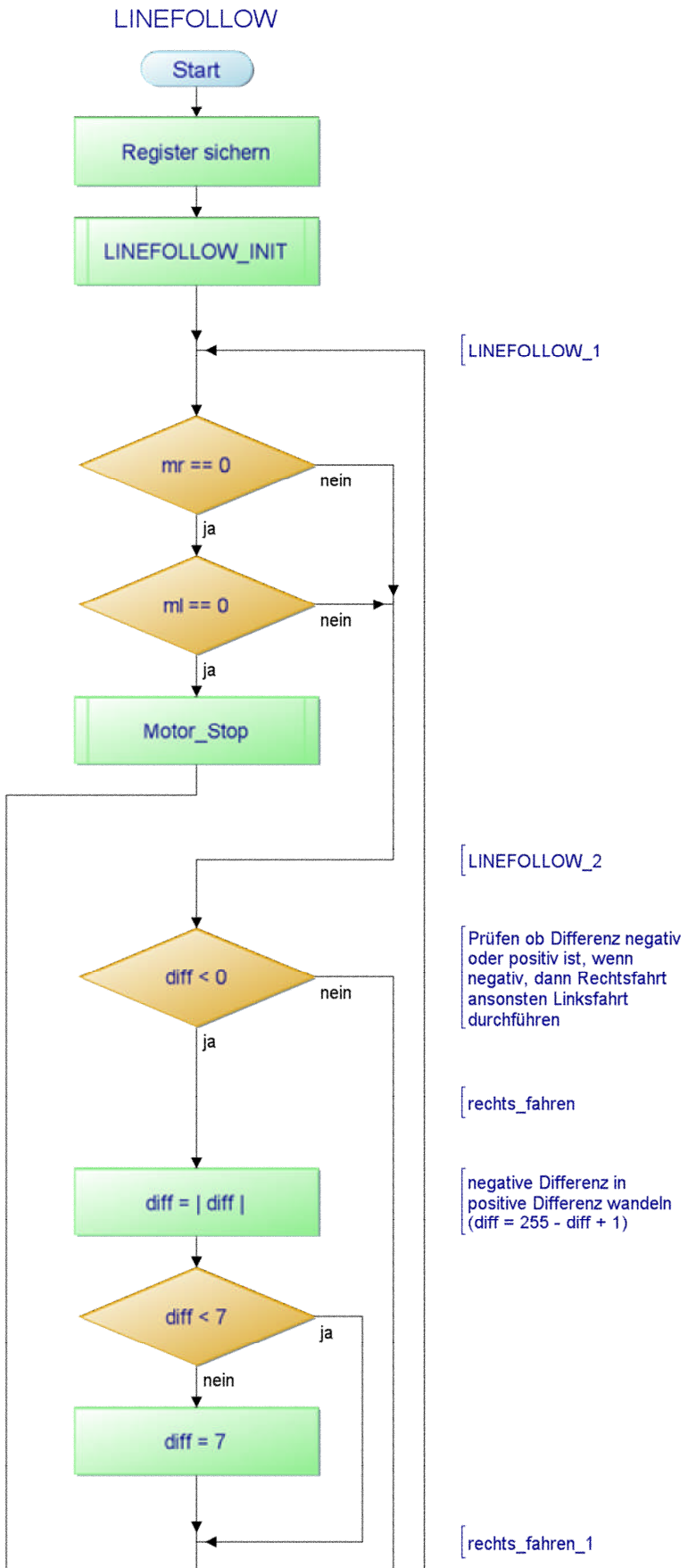
Nun kann der Asuro mit der Linienverfolgung starten. Nach dem sichern der verwendeten Register wird die Linienverfolgung initialisiert. Hierzu werden die Tabellen für die Regler in den dafür vorgesehen Bereichen abgespeichert. Die Größe des Speicherplatzes wurde zu Beginn des Gesamtprogramms mit 8 Byte je Regler definiert.
Zudem werden die Messwerte des linken und rechten Sensors (mr und ml) 1 gesetzt. Sollten diese Messwerte 0 annehmen ist das Ziel erreicht. Des weiteren wird die Front LED angeschalten. Danach wird der Timer0 auf die Wartezeit (definierter Wert) eingestellt und der Interrupt des Timer0 freigegeben. Nach dem wiederherstellen der gesicherten Register ist die Initialisierung des Linienfolgers beendet.

Nun wird überprüft ob die Messwerte Null sind damit das Ziel erreicht wäre. Ist dies der Fall wird der Motor gestoppt und LINEFOLLOW, nach wiederherstellen der zu Beginn des Unterprogramms gesicherten Register verlassen. Danach wird im Programmteil FAHREN der ADC Interrupt deaktiviert.

Motor_Stop setzt die Motorengeschwindigkeiten der PWM auf 0. Gleiches gilt für die gespeicherten Motorgeschwindigkeits-werte. Zudem werden die Bremslichter eingeschalten.

Sollten die Messwerte einen von 0 verschiedenen Wert annehmen wird geprüft in welche Richtung die Linien läuft. Dies geschieht durch den Vergleich der Sensorwerte. Dazu wird die Differenz gebildet. Je nachdem ob diese Differenz nun positiv oder negativ ist wird die Korrekturbewegung nach rechts oder links eingeleitet.
Im Falle einer negativen Differenz fährt der Asuro nach rechts.
Hierbei wird von der negative Differenz der Betrag gebildet. Dies erfolgt durch folgende Rechnung:
255 – Wert + 1

Folgend wird 7 als Maximalwert festgelegt und alle größeren Werte auf diesen gesetzt.

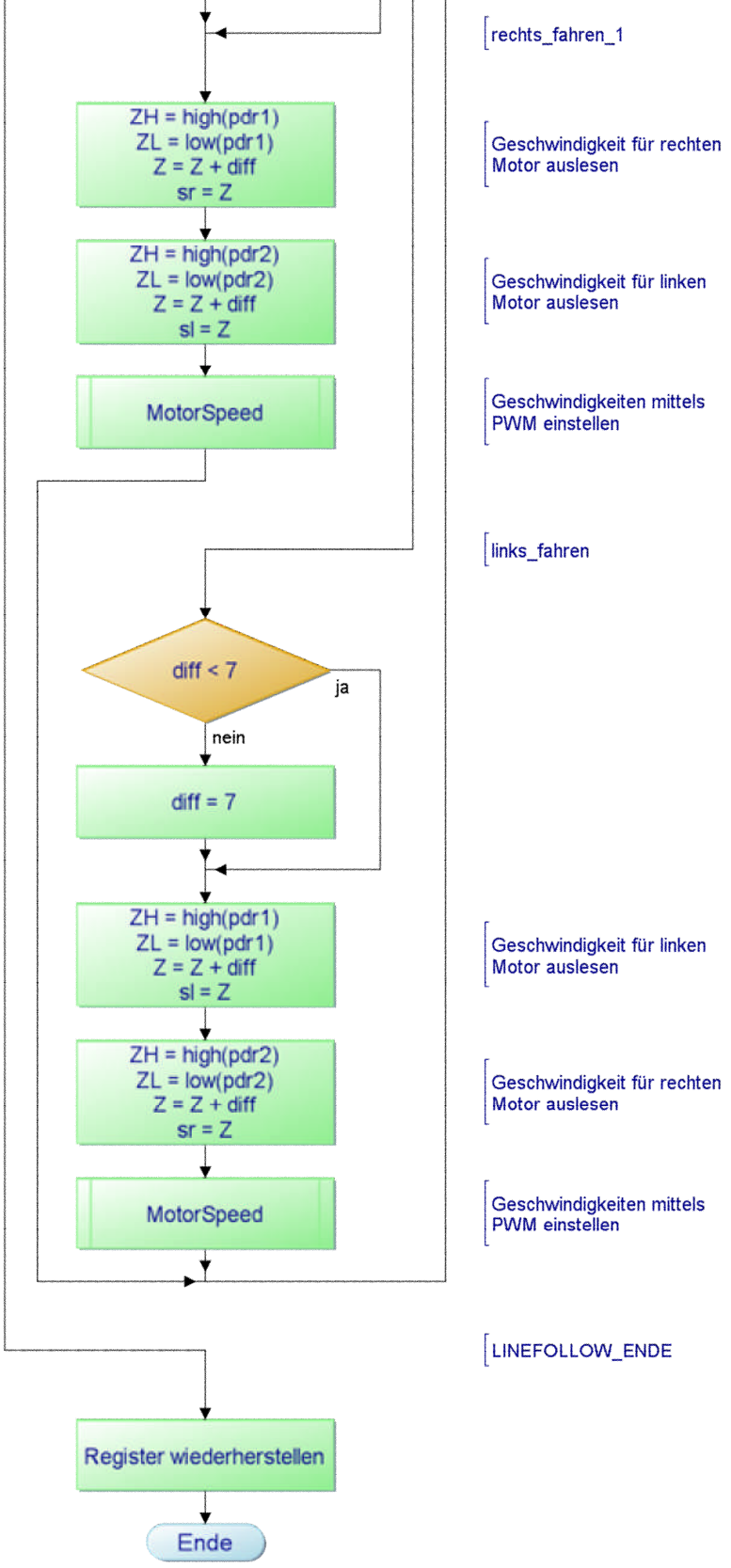


Nun wird werden den Messwerte Geschwindigkeitswerte zugewiesen. Hierbei wird dem Rad in nicht abdriftender Richtung ein Wert aus der Tabelle für Regler 1 zugeteilt. Für das Rad in Driftrichtung ein Wert aus der Tabelle für Regler 2.

In MotorSpeed werden die zugewiesenen Werte als Motorengeschwindigkeit umgesetzt.

Das gleiche Verfahren (bis auf die Betragsbildung) wird auch im Falle der Linksfahrt angewand.

Nach der (Neu)Einstellung der Motorengeschwindigkeit springt der PC wieder nach oben und ließt erneut die Messwerte aus.



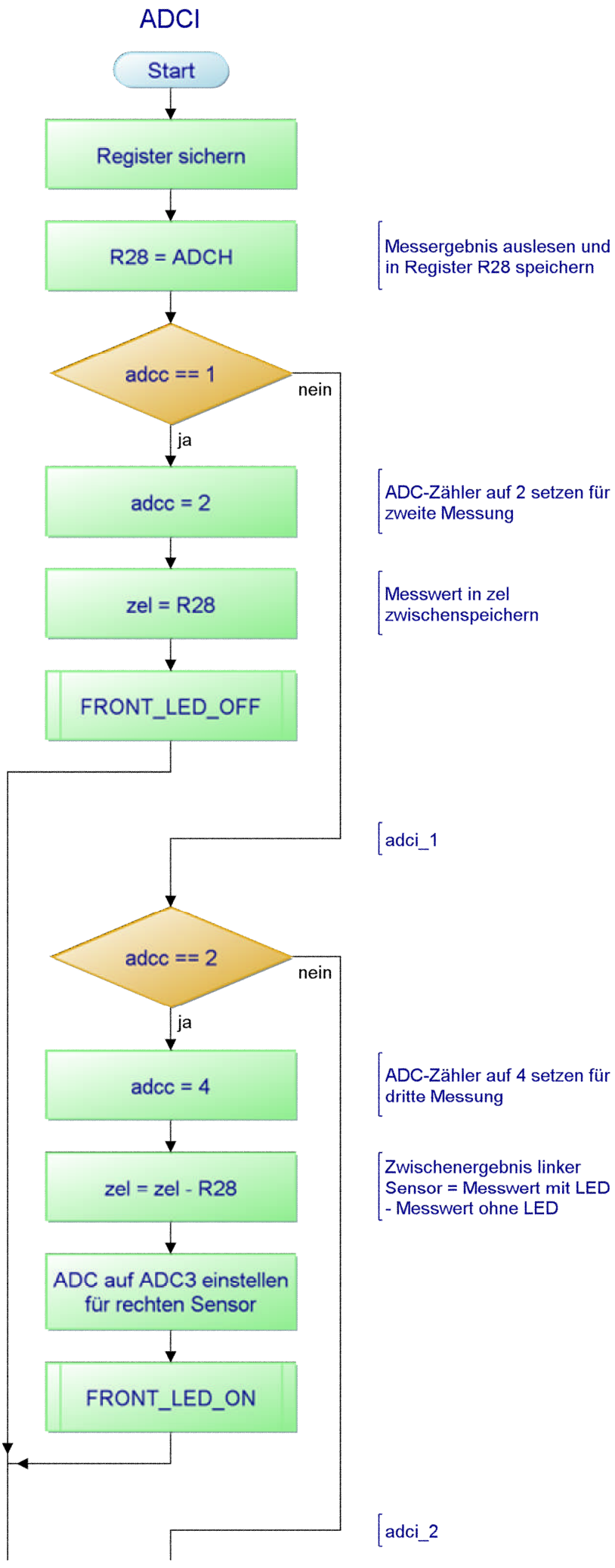
Jetzt zum Thema der Messwertauslesung und Einbindung in FAHREN. Dies geschieht durch den ADC Interrupt.

Zu Beginn werden, wie üblich, die verwendeten Register gesichern. Danach werden die Messwerte, welche sich in ADCH befinden in das Register R28 gespeichert. Danach wird geprüft ob ADCC = 1 ist. Hierzu wird das 0. Bit ausgelesen.

Sollte dies der Fall sein werden die Bits im ADCC nach links geschoben und so ein 2 erzeugt. Danach wird das Messergebnis als Zwischenergebnis links (zel) abgespeichert. Danach wird die Front LED abgeschalten.
In Folge (ab adci_4) wird der ADC Interrupt deaktiviert und dem Timer0 wird eine Wartezeit zugeteilt. Diese dient dazu abzuwarten bis die LED vollständig erleuchtet, bzw. erloschen ist. Durch die Differenz aus den Werten mit LED und ohne LED kann das Umgebungslicht herausgerechnet werden und damit auf ein abkleben der Messeinrichtung verzichtet werden.

Sollte in ADCC 2 stehen zeigt dies die zweite Messung an. Der Zähler wird so eingestellt das als nächstes die 3. Messung folgt und die oben angesprochene Subtraktion wird durchgeführt.

Die als nächstes stattfindende 3. Messung wird die des rechten Sensors mit LED sein.

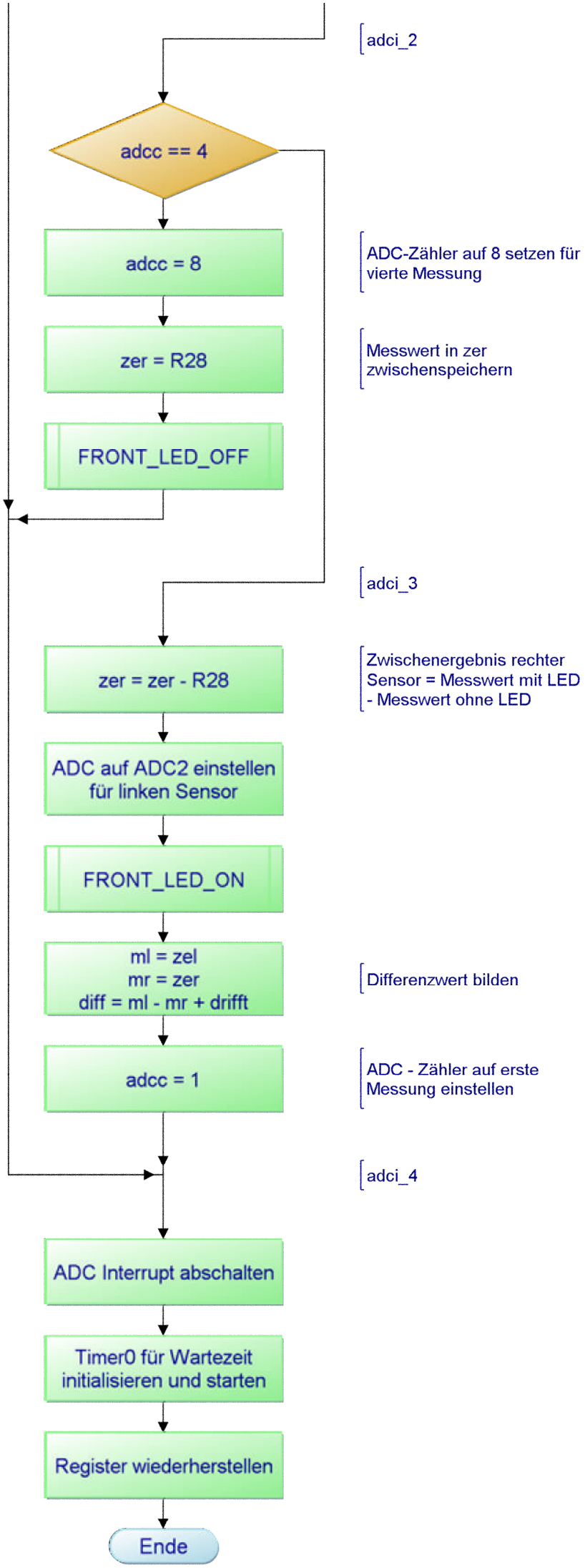


Zu Beginn der 3. Messung wird der ADC Zähler auf 8 gesetzt. Die Verdoppelung der Werte im Zähler wird durch ein Logical Shift Left realisiert. Der Messwert wird nun in den Zwischenspeicher Rechts (zer) gespeichert und die Front LED wieder abgeschalten.

Das letzt Messergebnis liefert den Messwert des rechten Sensors ohne LED. Wie bei den linken LEDs wird auch hier wieder die Differenz aus den beiden Werten gebildet.

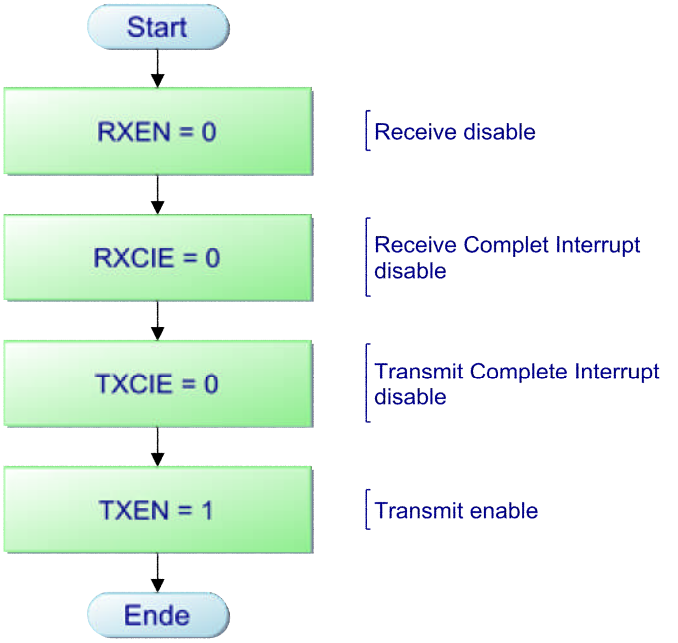
Anschließend muss der Ausgangszustand der Messauswertung wieder hergestellt werden. Dazu wird wieder auf den linken Sensor gestellt und die Front LED angeschalten.

Die Differenzwerte (mit und ohne LED) werden in die Messergebnisse rechts und links (mr und ml)geschrieben. Zudem wird die Differenz zwischen den Messergebnissen des rechten und linken Sensors gebildet. Hierzu wird der Drift (zwischen den Motoren, bzw. Rädern) gerechnet. Dieser Drift ist ein vordefinierter Wert und hängt von der jeweiligen Geräteeigenschaften des Asuros ab.

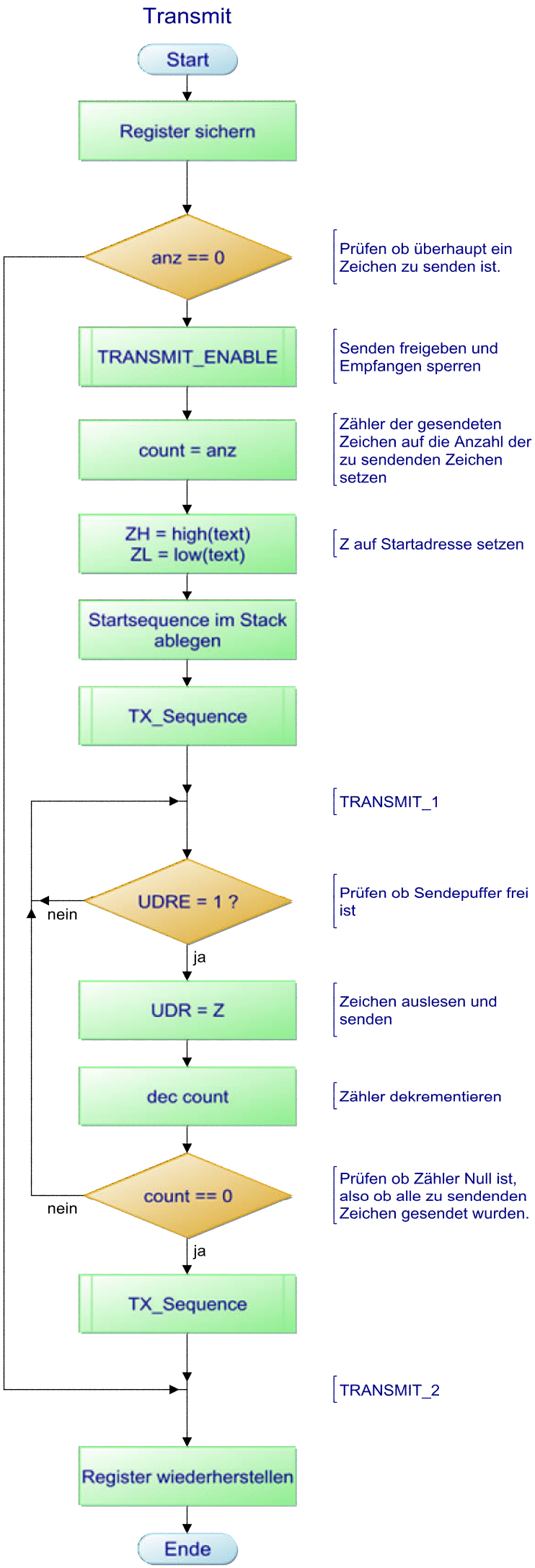


Beim Transmit, also dem Senden, wird nach dem Sichern der Register überprüft ob überhaupt ein Zeichen zum Senden bereit steht. Sollt dies der Fall sein wird in TRANSMIT_ENABLE gesprungen:

TRANSMIT_ENABLE



Hier wird das Empfangen ausgeschalten. Genauso wie die Interrupts bei der Benachrichtigung für ein empfangenes oder gesendetes Zeichen. Zudem wird die Übertragung aktiviert.
Zurück in Transmit wird der Zähler auf die Anzahl der gespeicherten Zeichen gesetzt. Danach werden die Zeiger auf die Startadressen der zuvor empfangenen Zeichenfolge gesetzt.
Nun wird TX-Sequence (siehe nächste Seite) geöffnet, um die Startsequence zu senden.
Danach können die Zeichen der Nachricht übermittelt werden. Hierzu muss zunächst solange gewartet werden bis in den Sendepuffer ein Zeichen geschrieben werden kann. Ist dies der Fall wird ein Zeichen (der Reihenfolge nach) in UDR geschrieben und übertragen und der Zähler für die Anzahl der Zeichen wird um 1 zurückgesetzt. Nun wird erneut TX_Sequence aufgerufen, um die Endsequence zu senden. Die Register werden wieder hergestellt und Transmit beendet.

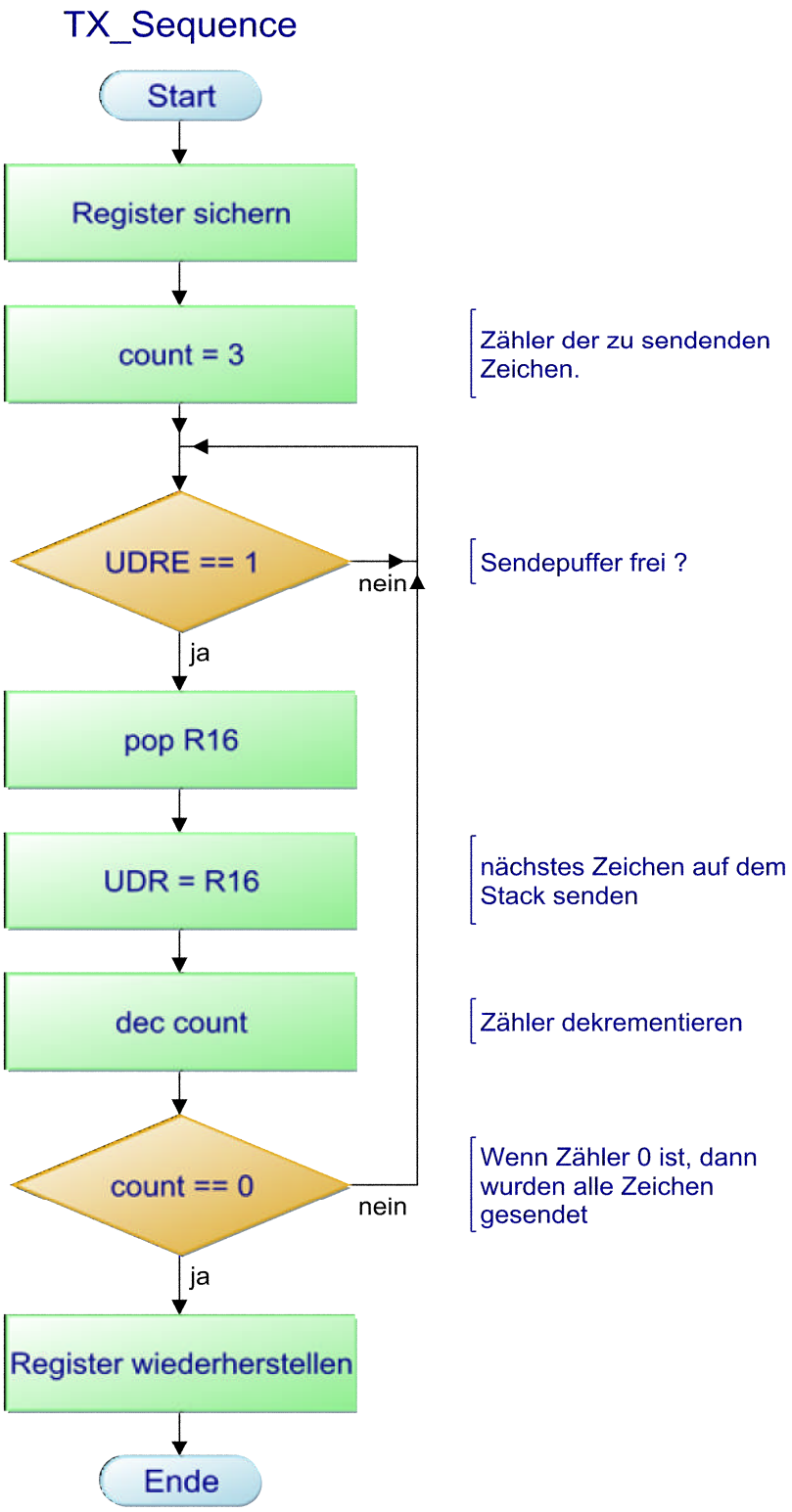


TX_Sequence sichert die Register R16, count und die Register R20, R21, R22 in welchen End- oder Startsequenz abgelegt sind.

Dann wird der Zähler auf 3 gesetzt und so lange gewartet bis der Sendepuffer frei ist.

R20 bis R22 werden nun bei je einem Schleifendurchgang in R16 geschoben und das Zeichen auf dem Stack abgelegt.

Sind alle 3 Register auf dem Stack werden das ursprüngliche R16 und count wieder hergestellt.



Probleme:

Ein grundsätzliches Problem ist im Aufbau, bzw. der Struktur der Aufgabenbewältigung zu suchen. Die Kommunikation funktioniert nur so wie in der Aufgabe beschrieben. Heißt: Person #2 kann erst antworten wenn ein Asuro bereit steht. Hierzu muss Person #1 erst einmal eine Nachricht übermittelt haben und damit den Vorgang in Bewegung gebracht haben.
Die jeweiligen Personen müssen immer erst auf die Antwort des anderen warten. Für eine nicht eingleisige Bewältigung des Problems wäre eine größere Anzahl von Asuros von Nöten.

An eine Grenze stößt das Projekt im Bereich der zu speichernden Zeichen. Die maximale Anzahl von Zeichen wären 1000. Dies gilt jedoch nur für den Fall, dass nichts weiter gespeichert würde. Allerdings nutzt auch der Stack den Speicher, genauso wie die Tabelle für die Reglerwerte.

Eine weitere Grenze ist im Teil der Linienverfolgung zu suchen. Eine zu dünne Linie ergibt zu kleine Unterschiede bei den Transistoren. Ebenso stellt der Kurvenradius eine Grenze dar. Wird dieser zu gering kann der Asuro der Linie nicht folgen und müsste in eine Liniensuche übergehen.