ST6-Realizer für ST6-Mikroprozessoren

Im zweiten Teil dieses Artikels stellen wir Ihnen die detaillierte Funktionsweise des ST6-Realizers anhand eines kleinen Software-Projektes vor.

Nach der Einführung in die Welt der ST6-Microcontroller-Familie und der ST6-Realizer-Software von SGS-Thomson im „ELVjournal" 6/97, erfolgt im hier vorliegenden zweiten Teil des Artikels die Realisierung eines kleinen Software-Projektes.

Erstellen eines Projektes mit dem ST6-Realizer

Die detaillierte Funktionsweise des ST6-Realizers ist am besten anhand eines kleinen Beispielprogrammes zu verdeutlichen. In unserem Programmierbeispiel soll nun mit dem Prozessor ST 6260 ein elektronisches Codeschloß realisiert werden, das nach der Programmierung nur noch mit wenigen externen Komponenten zu beschalten ist.

Zur Codeeingabe dient eine Universal-Tastatur, z. B. mit 12 Tastenfeldern.

Folgende Funktionen soll das zu programmierende Codeschloß aufweisen:

- Freigabe durch 3stelligen Zahlencode.
- Nach drei Fehlversuchen (falsche Codeeingabe) erfolgt eine 10 Minuten lange Sperre. Jeder weitere Eingabeversuch innerhalb der Sperrzeit setzt die 10 Minuten-Sperre neu, d. h. die Codeeingabe wird erst wieder akzeptiert, wenn 10 Minuten lang keine Tastenbetätigung erfolgte.
- Der Programmiermode für die Eingabe des Freigabe-Codes wird mit einem versteckt anzubringenden Taster aktiviert. Nach Betätigung des Programmiertasters ist dann innerhalb von 15 Sekunden der gewünschte 3stellige Zahlencode einzugeben.
- Der Programmiermode wird automatisch verlassen, wenn 15 Sekunden keine Tastenbetätigung erfolgte.

Dieses kleine Beispielprogramm kann vom Anwender beliebig auf die eigenen Bedürfnisse hin verändert und erweitert werden.

Zu beachten ist in diesem Zusammenhang, daß der Simulator nicht in Echtzeit, sondern mit einer minimalen Taktzeit von 10 ms arbeitet. Wenn später die Hardware

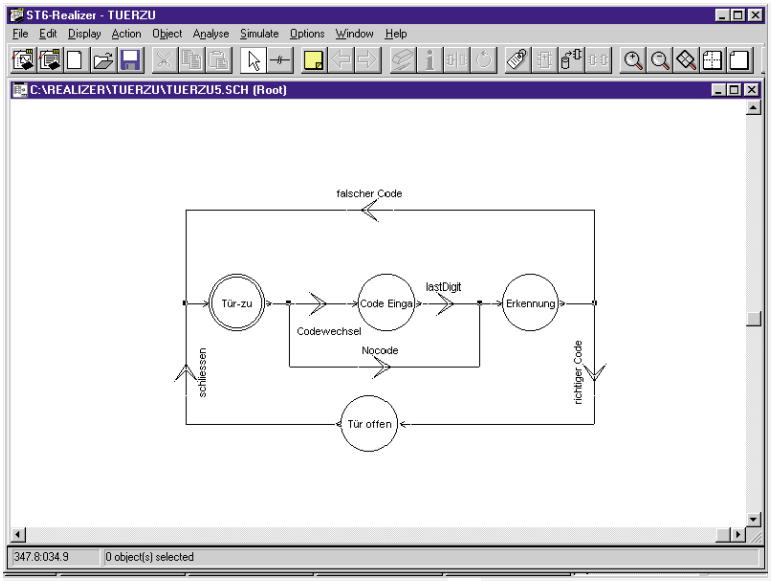

Bild 1: Die „State-Machine" unseres kleinen Programmierbeispiels

mit dem korrekten Prozessortakt betrieben wird, verhalten sich die Timer entsprechend anders als im Simulations-Mode.

Nachdem die Rahmenbedingungen für unser kleines Projekt soweit feststehen, ist der Realizer zu starten, wobei wir uns zuerst mit der grundlegenden Bedienung vertraut machen.

Im Menü „File" ist zuerst eine neue Projekt-Datei für die Applikation sowie das Projektblatt für die zu erstellende Zeichnung der Applikation aufzurufen.

Der gewünschte Controller wird im Menü „Options" unter „Select Hardware" ausgewählt. Der Realizer erstellt dann eine Projekt.ini-Datei, in der die Compiler, Pfad- und Zeichnungsnamen abgelegt sind.

Alle Dateien, die für die Erstellung der Applikation benutzt werden, sind in einer Projekt-Datei zusammengefaßt und jedes Projekt hat einen eigenen Pfad im Inhaltsverzeichnis.

Die erstellte ASCII-Datei kann über „Edit" (im Menü „File") angesehen und editiert werden.

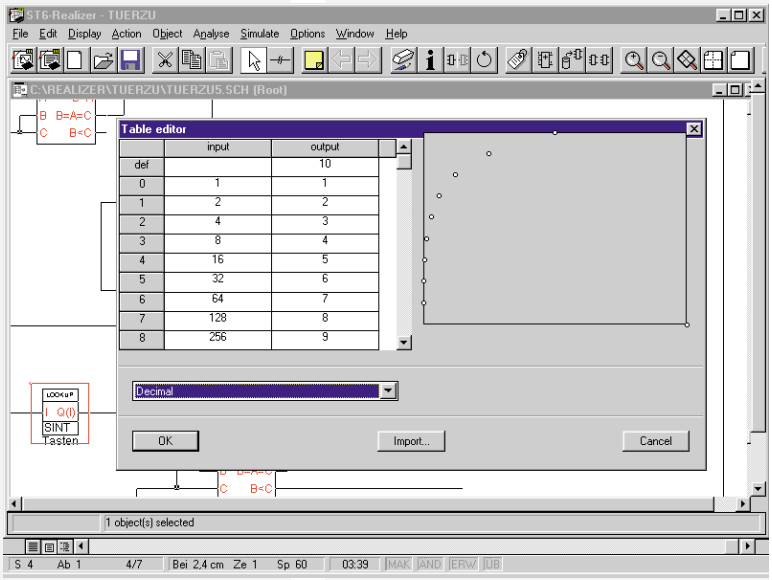

Bild 3: Eingebauter Editor zum Erstellen von „lookup tables"

Nach Anklicken von „New" im Menü „File" erscheint dann das Projektblatt für die zu erstellende Zeichnung der Applikation.

Nun kann es mit der State-Machine-Eingabe losgehen. Umfangreiche Symbol-Bibliotheken, die unter dem Menü „Objekt" zur Verfügung stehen, unterstützen dabei die Arbeit. Die aktuelle gewünschte Symbol-Bibliothek steht zum schnelleren Arbeiten auch in der Symbol-Leiste zur Verfügung.

Beim Aufbau der State-Machine wird immer mit einem „stateinit-Symbol" begonnen. Dieses Symbol ist dann über das Icon „wiring" mit dem Symbol für „condition" einem Draht oder einem weiteren State zu verbinden.

Jedes Symbol kann mit der Maus beliebig auf dem Projektblatt positioniert oder verschoben werden. Des weiteren ist in 90°-Schritten das Drehen der Symbole möglich.

Werden 2 Leitungen zusammengeführt, so entsteht an ihren Berührungspunkten eine Verbindung, wobei sich kreuzende Leitungen keine Verbindung ergeben.

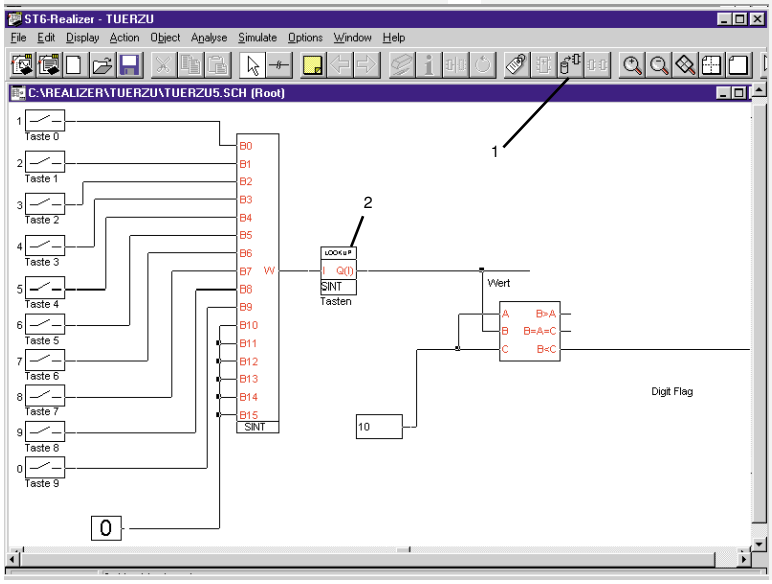

Bild 2: Tastaturabfrage des einfachen Codeschloß

Die „State-Maschine", d. h. der Programmablauf unseres kleinen Beispielprogrammes, ist in Abbildung 1 dargestellt. Der Startpunkt ist das „Stateinit"-Symbol „Tür zu".

Bei Betätigung der <Programmiertaste> wird der Code in die EEPROM-Zelle des Prozessors übernommen. Sobald das letzte Digit eingegeben ist, beginnt die Erkennung mit der Überprüfung des Codes. Ergibt die Erkennung den richtigen Code, so wird nach „Tür öffnen" verzweigt. Bei falscher Codeeingabe hingegen, folgt eine Verzweigung zum „Stateinit"-Symbol, d. h. der Prozessor erwartet eine neue Codeingabe.

Die Auswahl der Schaltungselemente für die „State-Maschine" erfolgt auch hier über das Icon der Symbol-Bibliothek in der Symbolleiste. Nach Öffnen der Bibliothek ist das gewünschte Schaltungssymbol per Rollbalken und Mausklick zu selektieren.

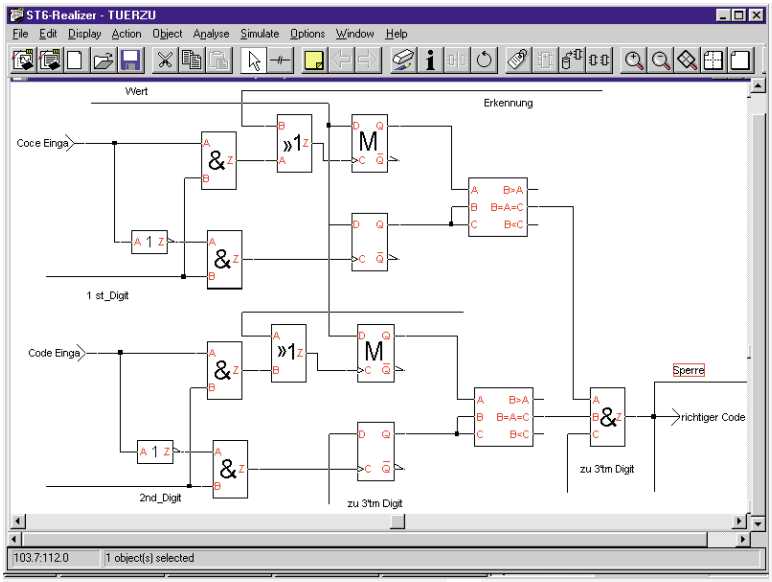

Bild 4: Speicherung des 3stelligen Tastaturwertes im EEPROM.

Die einzelnen Symbole und Schaltungselemente sind auf dem Projektblatt so zu plazieren und zu verbinden, daß die Funktion der Schaltung gewährleistet ist.

Nach Aufbau der State-Machine, d. h. der Ablauf des Programmes wurde in seine einzelnen Zustände zerlegt, beginnt die Eingabe der Schaltungselemente. Dabei wird im Grunde genommen nichts anderes als ein Schaltplan erstellt.

Die Schaltplaneingabe beginnen wir mit der Schaltung für die Tastaturabfrage (Abbildung 2). Wie bereits erwähnt, ist für unser kleines Beispielprojekt eine Tastatur mit insgesamt 12 Tasten vorgesehen. Auch hier werden die einzelnen Schaltungselemente aus der Bauteilbibliothek (1) aufgerufen und auf dem Projektblatt plaziert. Für Tasteneingaben wählen wir aus der Bibliothek das Symbol „digin".

Die Definition für die Pin-Zuordnung erfolgt dann später über einen Doppelklick auf das Symbol des Schalters.

Über das Symbol „wpack" aus der Bibliothek und einer Tabelle unter „lookup" wird die Wertigkeit der Tasten bestimmt. Nach den 10 Tasten für die Zifferneingabe folgt das Symbol „bpack", das bis zu 16 Eingänge in ein 16 Bit langes serielles Wort umwandelt. Dieses 16-Bit-Wort wird dann der „lookup-table" zugeführt.

Die „lookup-table" (2) wird nun so benutzt, daß dem Eingangswert ein eindeutiger Ausgangswert zugeordnet wird. In unserem Fall ist der Wert 1 der Taste 0, der Wert 2 der Taste 1 usw. zugeordnet.

Die Tabelle mit der Dateikennung „.tab" ist mit dem in Abbildung 3 dargestellten, eingebauten Editor zu erstellen. Um diesen Editor aufzurufen, ist ein Doppelklick auf das Symbol „lookup" erforderlich, wobei die Eingabe in den angegebenen Zahlenformaten erfolgt. Die so erhaltenen Werte werden für die Weiterverarbeitung über das Label „Wert" zur Verfügung gestellt und gleichzeitig nach einem Vergleich als Tastatur-Strobe für die „Digit-Flag" genutzt.

Um den Tastaturwert zu speichern, benötigen wir das im ST 6260 integrierte EEPROM, daß in Abbildung 4 mit „M" bezeichnet ist. Die Symbole für das EEPROM sind in der Bibliothek des Realizers enthalten und bestehen aus Mehrfach- „D-Latch-Makros", d. h. sie sind für Bit, Byte, Word, Integer, Signed-Variablen zu benutzen.

In unserer Beispielschaltung benötigen wir zur Beschaltung: Wert, Erkennung, 1st-Digit, 2nd-Digit, 3rd-Digit und Codeeingabe. Erkennung und Codeeingabe werden der Logik durch die Symbole „state-out" zugeführt, und über die nachgeschalteten Vergleicher „comp" wird dann das Signal „richtiger Code" erzeugt. Der für Digit 1 und Digit 2 dargestellte Schaltungsteil ist in unserem Codeschloß für jedes Digit, also 3mal, vorhanden.

Mit einem programmierbaren Up-Down-Counter (Abbildung 5) ist das Zählen der Digits und Tasten realisiert. Auch dieses Schaltungselement ist in der Symbolbibliothek des Realizers enthalten. Spezielle, nicht enthaltene Symbole sind über einen Makro-Editor selbst zu erzeugen und in die Bibliothek einzubinden.

Die „Statein"-Signale 1st-Digit, 2nd-Digit und 3rd-Digit (Last-Digit) werden über Vergleicher generiert und mit der Codeeingabe verknüpft (Abbildung 6). Bei Eingabe von drei falschen Codes wird der Timer T 3 in Abbildung 6 aktiviert und die Codeeingabe so lange gesperrt, bis die Sperrzeit abgelaufen ist und der richtige Code eingegeben wird.

Der mit einer Konstanten vorbelegte Timer T2 in Abbildung 7 ist für die max. zulässige Eingabezeit zuständig. Die Taste „Codewechsel" stößt den Timer an, der dann das EEPROM freigibt, das Eingabezeitfenster öffnet und den „Statein" Codewechsel versorgt. Der weitere in Abbildung 7 dargestellte Timer T 1 legt die Zeit für das Öffnen des Türöffners (20mA-Opendrain-Ausgang) fest.

Wie nun die Ein- und Ausgabeports unseres Mikrokontrollers benannt und beschaltet werden, ist Tabelle 1 zu entnehmen.

Anhand unseres kleinen Beispielprogramms haben wir nun einige grundlegende Symbole kennengelernt, gesehen, wie diese plaziert und editiert werden und wie die Pins zu verbinden sind. Mit dem einfachen Codeschloß haben wir bereits die erste kleine ST6-Applikation geschrieben oder besser gesagt gezeichnet. Nun fehlt noch die Analyse und das Generieren des Programmcodes.

Bevor wir jedoch mit dem Analysieren und Kompilieren des Schaltbildes beginnen können, sind noch einige grundlegende Einstellungen vorzunehmen.

Dazu rufen wir im Menü „Analyse" die Funktion „Options" auf. Im Fenster „Analyseoptions" sind die Fragen: Generate Source Code, Invoke Assembler, Generate final hex code und include projekt Data mit „ja" zu beantworten und der Timer-Tick

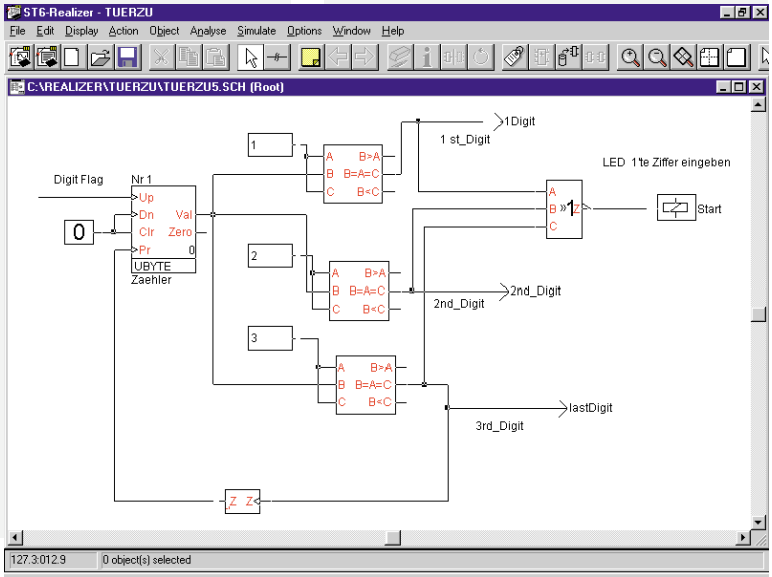

Bild 5: Zählen der Digits und Tasten

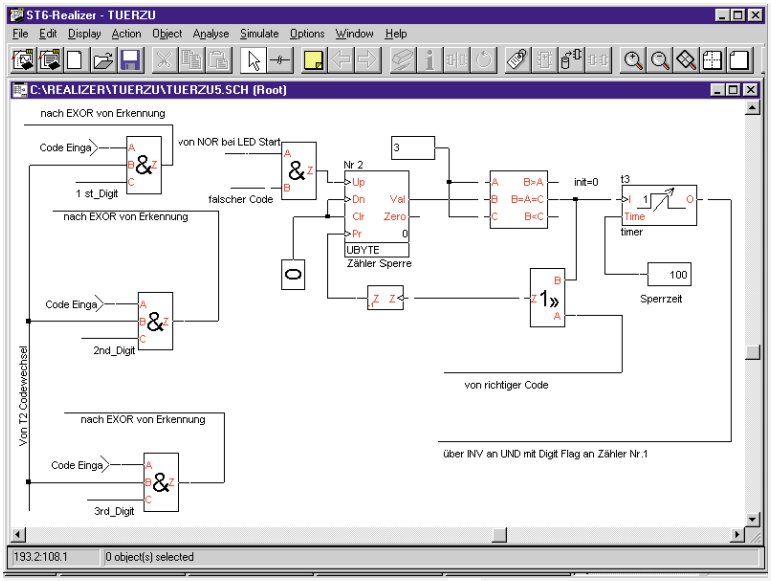

Bild 6: Beschaltung für die Code-Eingabe und Sperre bei 3mal falschem Code

auf 0,01 Sekunden einzustellen. Nach Bestätigung der Eingaben mit OK wird das Fenster wieder geschlossen.

Ebenfalls im Menü „Analyse" sind die Hardware-Settings einzustellen. Hier können die Werte für Frequenz, Start of ROM, End of ROM, Start of RAM, End of RAM, Assembler-Options, Include-Files, Watchdog und ROS (Realizer Operating System) geändert werden.

Zur Analyse der erstellten Applikation ist „Go" anzuklicken. Nun kreiert die Software die Netzliste und generiert den endgültigen Code.

Treten hierbei keine Fehlermeldungen auf, so sollte die erstellte Applikation funktionsfähig sein. Fehlermeldungen hingegen werden in einem „Report-File" angezeigt.

Durch Anklicken der Fehler werden wir dann wieder in den Schaltplan gebracht und auf die Fehlerquelle per roter Umrandung hingewiesen. Die Fehler sind dann zu beheben und ein neuer Analyselauf vorzunehmen. „Warnings" hingegen gelten nur als Hinweis, ohne daß ein unbedingter Handlungsbedarf besteht.

Simulation der Schaltung

Zur Simulation der erstellten Schaltung ist ein Simulations-Environment-File einzurichten. Zum Öffnen des ST6-Simulatorfensters ist im Menü „Simulate" der Menüpunkt „Go" anzuklicken.

Im Menü „File" wird dann entweder „New" angeklickt, oder wenn das „File" schon geschrieben ist, die entsprechende Datei geöffnet.

Die Simulation erfolgt durch Anklicken der zu testenden Eingangsleitungen, wobei abhängig von der Funktion zwischen folgenden Anzeigen gewählt werden kann:

Numeric Adjuster: Diese Funktion wird benutzt, um variable Werte numerisch, binär, dezimal, hexadezimal oder oktal anzuzeigen oder einzugeben.

Time Table Adjuster: Funktion zum Einstellen von variablen Werten auf bestimmte feste Werte.

Sine Wave Adjuster: Funktion zum Generieren von analogen sinusförmigen Testsignalen mit einstellbarer Amplitude und Frequenz.

Square Wave Adjuster: Mit dieser Funktion lassen sich Amplitude, duty cycle und Frequenz an einem Eingang einstellen.

Zur grafischen Darstellung von Variablen gibt es zusätzlich noch die Möglichkeit, einen sogenannten Oszilloskop-Testpunkt zu setzen. Dabei sind die X-Achse, die Y-Achse und die Trigger beliebig zu modifizieren.

Um den Ablauf der State-Maschine sichtbar zu machen, gibt es eine weitere Testmöglichkeit. Das zugehörige Icon befindet sich in der Menüzeile und wird beim Anklicken des entsprechenden States sichtbar.

Dieser Artikel kann nur als kleine Einführung in die Arbeitsweise des komplexen ST6-Realizers gelten. Die vorgestellte Applikation ist dabei als Anregung gedacht und nach eigenen Wünschen beliebig zu verändern.

Im „ELVjournal" 2/98 geht es dann weiter mit der Vorstellung eines ST6-Starterkits. Die Starterkits von SGS-Thomson bilden eine komplette Entwicklungsumgebung mit Programmiergerät, Software und Dokumentation. ELV

Tabelle 1: Ein- und Ausgabeports des Mikrocontrollers

Symbolic name		H/W name Type	Comment
0	PB.1	Input, with pull-up	Taste 9
1	PB.2	Input, with pull-up	Taste 0
2	PB.3	Input, with pull-up	Taste 1
3	PA.2	Input, with pull-up	Taste 2
4	PB.7	Input, with pull-up	Taste 3
5	PA.0	Input, with pull-up	Taste 4
6	PA.3	Input, with pull-up	Taste 5
7	PB.6	Input, with pull-up	Taste 6
8	PC.2	Input, with pull-up	Taste 7
9	PC.3	Input, with pull-up	Taste 8
Codewech	PA.1	Input, with pull-up	Codew
öffnen	PB.0	20 mA sink open drain output	öffnen
Start	PC.4	5 mA sink open drain output	LED erste Ziffer eingeben

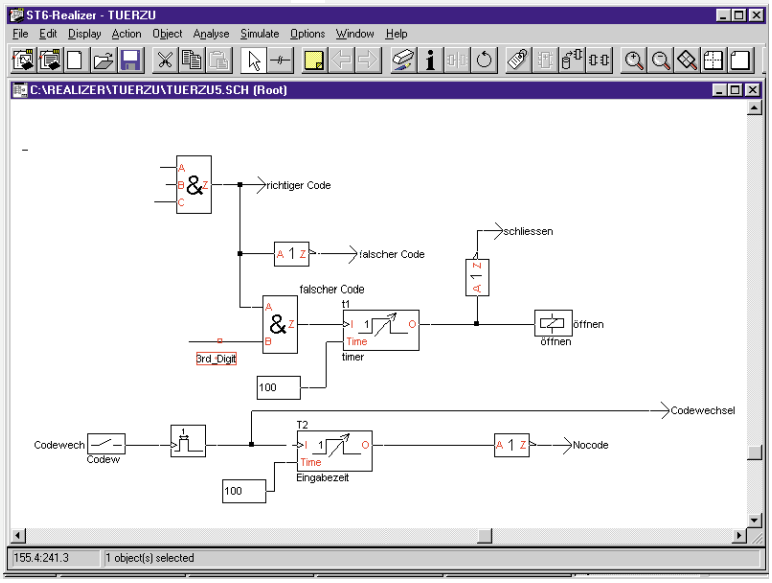

Bild 7: Der Timer T 2 bestimmt die max. zulässige Eingabezeit und T 1 legt die Zeit für das Öffnen fest.