



BELEGARBEIT

4-GEWINNT MIT RASPBERRY PI DOKUMENTATION

| | |
|------------------|--|
| Vorgelegt von: | Mario Hesse, 66433 Chrisian Weustink, 66329 |
| Eingereicht bei: | Prof. Pretschner |
| Im Studienfach: | Embedded Systems I |
| Ort, Datum: | Leipzig, den 20. Mai 2016 |

Vorwort

Studenten technischer Studiengänge fühlen sich häufig mit Spielen, insbesondere mit Videospielen verbunden. Diese Spiele können bei jungen Menschen ein erstes Interesse für Technik und digitale Systeme entfachen und damit den Nährboden für ein technik-begeistertes Leben schaffen.

Die Faszination beginnt beim Betrachten eines technischen Systems. Dabei kann es sich um ein Garagentor mit Fernbedienung, eine Stereoanlage oder eine Spielekonsole handeln. Bereits bei simplen Gesten, wie dem Drücken eines Tasters oder dem Drehen an einem Potentiometer erfolgen Reaktionen. Auf derlei Eingaben können die verschiedensten Dinge passieren. So kann beispielsweise das Garagentor aufgehen, die Lautstärke erhöht werden oder ein Menü auf dem Bildschirm gesteuert werden.

Dabei wird die Lust am Probieren geweckt und der Benutzer ist dazu geneigt die Reaktionen des Systems erneut zu prüfen und die Eingaben zu variieren. So werden alle Funktionen des Systems schrittweise erkundet. Hinter diesem menschlichen Verhalten steckt die Freude am Spielen und die Lust, Neues zu entdecken.

Für uns haben Spiele daher nie ihren Reiz verloren, und deswegen war es besonders spannend, selbst ein Spiel zu entwickeln. Das in dieser Projektarbeit entstandene 4-Gewinnt-Spiel ist zwar keine eigene Spielidee, jedoch eine hardwarenahe Realisierung eines großartigen Spielklassikers.

Wir hatten sehr viel Freude bei der Entwicklung dieses Projekts und bedanken uns herzlich bei Prof. Pretschner für die Zulassung unserer Projektidee als Prüfungsleistung. In Zukunft werden wir sicher noch die ein oder andere Gelegenheit nutzen, um auch weitere Spiele zu entwickeln.

Leipzig, den 28. März 2016
Mario Hesse und Christian Weustink

Inhaltsverzeichnis

| | |
|---|----|
| Vorwort | 2 |
| Inhaltsverzeichnis | 3 |
| Abbildungsverzeichnis | 5 |
| Tabellenverzeichnis | 5 |
| Abkürzungsverzeichnis | 5 |
| Glossar | 7 |
| 1. Aufgabenstellung | 8 |
| 2. Konzept | 8 |
| 2.1. Spielprinzip | 8 |
| 2.2. Realisierung | 9 |
| 3. Hardware | 10 |
| 3.1. LED-Matrix | 11 |
| 3.2. Taster | 13 |
| 3.3. GPIO-Schnittstelle | 13 |
| 4. Software | 15 |
| 4.1. Softwaremodule | 15 |
| 4.1.1. Definition der Variablen | 15 |
| 4.1.2. Definition der Funktionen | 17 |
| 4.2. Programmablauf | 23 |
| 5. Diskussion | 25 |
| 5.1. Bekannte Probleme und Fehler | 25 |
| 5.2. Ausblick | 25 |
| A. Schaltplan | 27 |
| B. Platinenlayout | 28 |

| | |
|---|-----------|
| C. Datenblatt Duo-LED L-59EGW (Auszug) | 29 |
| D. Datenblatt Schieberegister 74HC595 (Auszug) | 32 |
| E. Datenblatt High-Side-Treiber UDN2981 (Auszug) | 36 |
| F. Datenblatt Low-Side-Treiber ULN2803 (Auszug) | 39 |

Abbildungsverzeichnis

| | |
|---|----|
| 1. Konzept der Realisierung | 8 |
| 2. 4-Gewinnt als Brettspiel | 9 |
| 3. Schaltung als realer Aufbau | 10 |
| 4. Lochrasterplatine mit Bauelementen | 11 |
| 5. Ansteuerung der Matrix | 12 |
| 6. Abhängigkeiten der Funktionen | 17 |

Tabellenverzeichnis

| | |
|--|----|
| 1. Beschreibung der GPIO-Schnittstelle | 14 |
| 2. Definition der GPIO-Pins | 16 |
| 3. Programm-Variablen | 16 |

Abkürzungsverzeichnis

| | |
|-------|--|
| HTWK | Hochschule für Technik, Wirtschaft und Kultur |
| k.A. | keine Angabe |
| Abk. | Abkürzung |
| RPi | Raspberry Pi |
| Board | Board, auf dem die LED-Matrix aufgebaut ist |
| SER | (engl. serial input) serieller Dateneingang des Schieberegisters |
| SCK | (engl. shift register clock) Takt, um einen Datenvektor durch die Schieberegister zu schieben |
| RCK | (engl. storage register clock) Takt, um den aktuell im Schieberegister vorliegenden Datenvektor an die Ausgänge des Schieberegisters zu übergeben und dort zu halten |
| SCL | (engl. shift register clear) lösche den Datenvektor im Schieberegister |
| G | (engl. global enable) Freigabe der Tri-State-Ausgänge |

Glossar

| | |
|-----------|--|
| Matrix | ...ist im Allgemeinen eine Anordnung in Form einer Tabelle. In diesem Projekt ist mit dem Begriff LED-Matrix eine Anordnung von LEDs in Zeilen und Spalten gemeint. |
| Duo-LED | ...sind zwei LEDs, die in einem LED-Gehäuse integriert sind. Auf diese Weise lassen sich zwei verschieden Farben in einer LED anzeigen. |
| Coin | ...sind im ursprünglichen 4-Gewinnt runde, stabile Plättchen aus Plaste, welche zum Spielen in die Matrix eingeworfen werden. Hier wurde der Name Coin symbolisch für eine farbig leuchtende LED übernommen. |
| Tri-State | ...beschreibt, dass ein digitaler Ausgang statt den üblichen 2 Zuständen high und low auch einen dritten Zustand hochohmig haben kann. Wird der Ausgang hochohmig geschaltet, ist dieser Teil der Schaltung als abgekoppelt zu betrachten. |
| Python | ...ist eine höhere Programmiersprache mit Interpreter. |

1. Aufgabenstellung

Im Studienfach *Embedded Systems I* ist im 1. Semester des Masterstudiengangs *Elektro- und Informationstechnik* an der HTWK Leipzig eine Projektarbeit anzufertigen. Diese umfasst ein Programmierprojekt, welches mit einem *Raspberry Pi* umgesetzt werden soll.

Das Thema des Projekts und die verwendete Programmiersprache sind frei zu wählen und mit dem Dozenten abzusprechen. Der Umfang der Arbeit soll einer Prüfungsleistung eines Studienfachs mit 5 ECTS Punkten gerecht werden.

2. Konzept

Das von uns gewählte Projekt ist der Aufbau und das Programmieren eines 4-Gewinnt-Spiels. Zur Realisierung dieses Spieleklassikers wird ein Hardwaremodul und eine Ansteuerungssoftware für einen Raspberry Pi entwickelt. Abbildung 1 zeigt eine schematische Darstellung des Konzepts und der Verbindung der Komponenten.

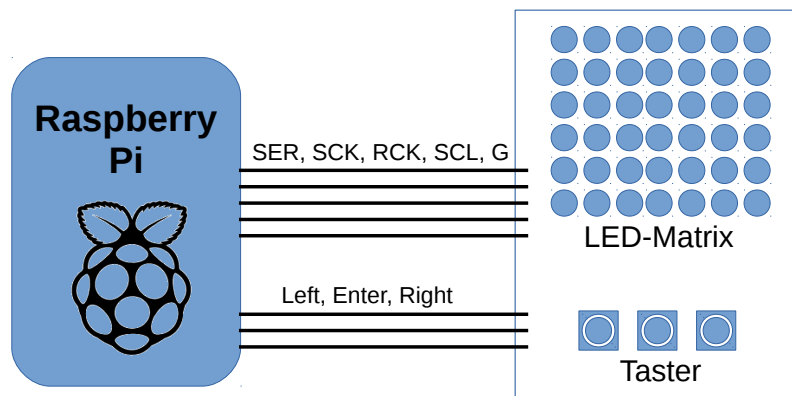


Abbildung 1: Konzept der Realisierung

2.1. Spielprinzip

Dieses Spiel wird auf einer Matrix mit 7 Spalten und 6 Zeilen gespielt. Abbildung 2 zeigt ein reguläres Spielbrett. Um zu spielen werfen 2 Spieler abwechselnd verschiedenfarbige Coins in eine der Spalten. Jeder Spieler spielt nur mit einer Farbe. Ein geworfener Coin fällt nach unten bis zur untersten noch nicht besetzten Zeile. Ziel des Spiels ist es, 4 Coins

hintereinander in waagerechter, senkrechter oder schräger Position zu positionieren. Der Spieler, der dieses Ziel zuerst erreicht, gewinnt das Spiel.



Quelle: www.hasbro.com

Abbildung 2: 4-Gewinnt als Brettspiel

2.2. Realisierung

Um das Spielfeld zu gestalten, wird auf einer Lochrasterplatine eine 7x6 Duo-LED-Matrix aufgebaut. Durch das Verwenden von Duo-LEDs können zwei Farben für zwei Spieler verwendet werden. Gleichzeitig vergrößert sich dadurch das Datenaufkommen. So entsteht eine 14x6 Datenmatrix (14xAnode, 6xKathode). Diese Matrix wird durch seriell zusammengeschaltete Schieberegister angesteuert. Diese Ansteuerungsvariante hat den Vorteil, dass nur 3 Pins programmiert werden müssen: Ein serieller Datenpin (SER) und zwei Taktpins. Bei den Takten handelt es sich um den Schiebetakt (SCK) und den Speichertakt (RCK). Der Schiebetakt hat die Aufgabe, den über den seriellen Datenpin angelegten Datenvektor durch die Register zu schieben. Der Speichertakt wird verwendet, um die in den Schieberegistern befindlichen Werte an die Ausgangspins zu übergeben und diese bis zum nächsten Speichertakt zu halten. Zusätzlich zu den genannten Pins gibt es zwei weitere Pins, welche programmiert werden können. Dabei handelt es sich um den Lösch-Pin (SCL) und den Tri-State-Pin(G). Der Lösch-Pin verursacht das Löschen des gesamten, im Schieberegister befindlichen, Datenvektors. Der Tri-State-Pin kann alle Ausgänge der Schieberegister hochohmig schalten und diese somit von der LED-Matrix abkoppeln.

Bedient werden soll das Spiel mit 3 Tastern: *Right*, *Left* und *Enter*. Dabei werden die Coins vom Right- oder Left-Taster jeweils nach rechts oder links bewegt. Der Enter-Taster soll das Fallenlassen der Coins auslösen. Des Weiteren soll bei längerem Halten des Enter-Tasters ein Reset ausgelöst werden, der das Spiel neu startet.

Der Raspberry Pi verwendet das Betriebssystem Raspbian. Die Projekt-Software wurde in Python geschrieben. Als Schnittstelle zwischen Hard- und Software werden die GPIOs des Raspberry Pi verwendet.

3. Hardware

Die Hardware wurde im Rahmen des Projekts entwickelt und optimiert. Abbildung 3 zeigt die Schaltung als realen Aufbau. Da der Schaltungsaufwand von uns als übersichtlich eingestuft wurde, haben wir auf das Entwickeln einer gedruckten Schaltung in Industriequalität verzichtet. Stattdessen haben wir uns für eine Lösung auf Lochrasterplatine entschieden. Abbildung 4 zeigt die Platine schematisch mit einer Beschreibung der montierten Teile. Die Platine ist für die Verwendung mit einem Raspberry Pi entwickelt und verlangt eine Betriebsspannung von +5V. Der Schaltplan der Platine ist in Anhang A auf Seite 27 und das Platinenlayout in Anhang B auf Seite 28 zu finden.

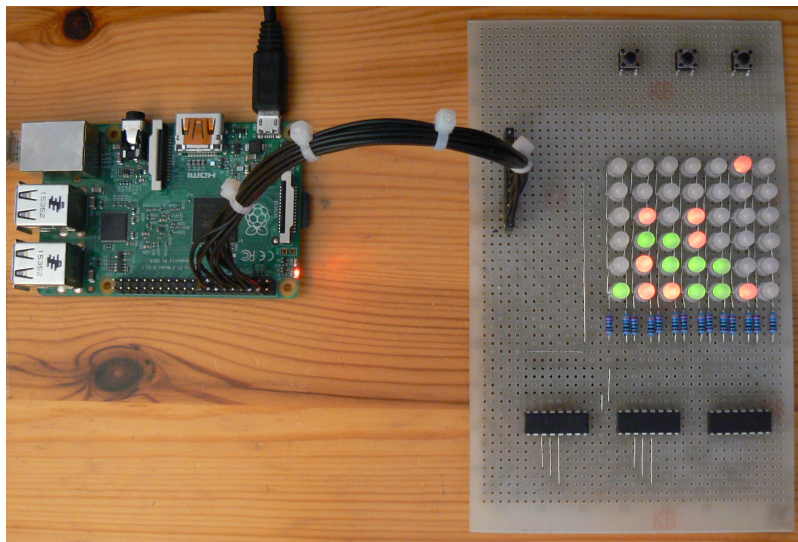


Abbildung 3: Schaltung als realer Aufbau

Teile der Platine (Abb. 4)

1. Schieberegister (74HC595)
2. High-Side-Treiber (UDN2981)
3. Low-Side-Treiber (ULN2803)
4. Vorwiderstände der LEDs
5. Duo-LEDs (grün-rot)
6. Taster (Left, Enter, Right)
7. GPIO-Schnittstelle

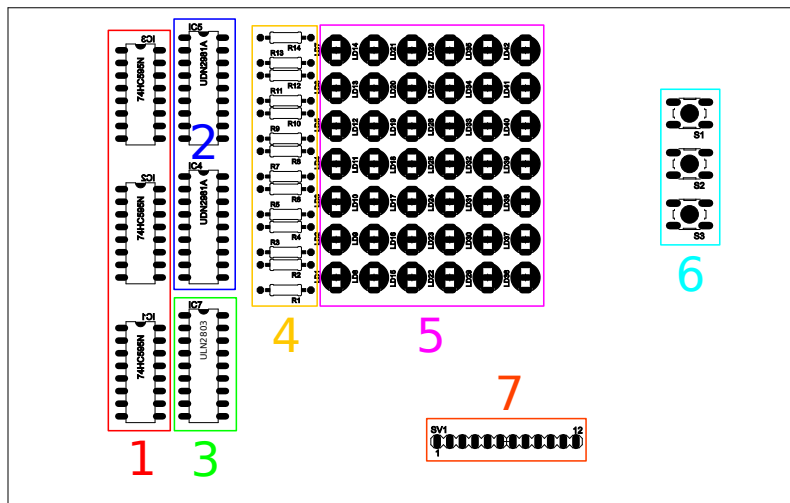


Abbildung 4: Lochrasterplatine mit Bauelementen

3.1. LED-Matrix

Die LED-Matrix besteht aus 7x6 Duo-LEDs mit gemeinsamer Kathode. Sie haben die Farben Rot und Grün. Die Matrix wird durch zwei 8-Bit High-Side-Treiber an den 14 Anoden und einem 8-Bit Low-Side-Treiber an den 6 Kathoden mit Strom versorgt. Die Ansteuerung der Matrix übernehmen drei in Serie geschaltete 8-Bit Schieberegister. In den Anhängen C, D, E und F finden sich zu allen verwendeten LED-Matrix-Bauteilen Auszüge aus den jeweiligen Datenblättern.

Abbildung 5 zeigt den Aufbau und die Ansteuerung der LED-Matrix. Dabei beginnt der Datenvektor stets an der oberen rechten Seite der Matrix und zieht sich durch die

Spalten von rechts nach links und anschließend durch die Zeilen von oben nach unten bis zum unteren linken Ende der Matrix.

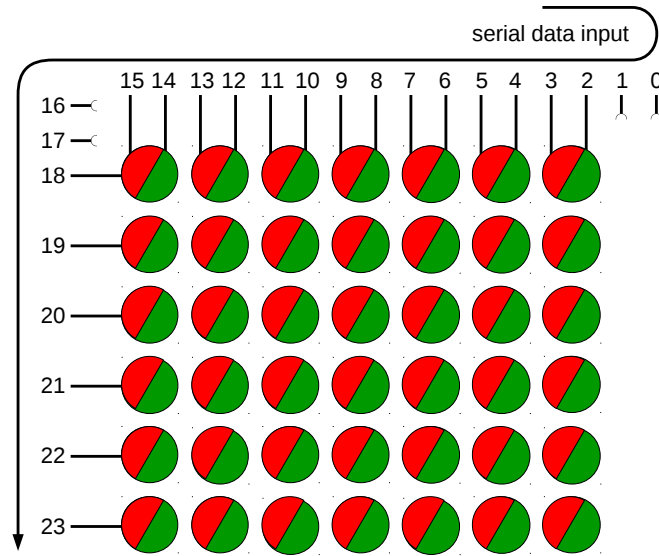


Abbildung 5: Ansteuerung der Matrix

Um für eine richtige Anzeige zu sorgen, muss der Datenvektor korrekt zusammengesetzt sein. Die Vektorpositionen 0, 1, 16, 17 sind nicht beschaltet. Welche Werte hier eingetragen werden ist grundsätzlich irrelevant. Bei den Positionen 2–15 handelt es sich um die Matrix-Spalten. Dabei sprechen die geraden Zahlen die grüne Anode der jeweiligen LED an, während die ungeraden Zahlen die rote Anode der jeweiligen LED ansprechen. Die Positionen 18–23 entsprechen den Zeilen, hier werden die Kathoden der jeweiligen LEDs angesprochen.

Um eine LED in der Matrix anzusteuern, wird sowohl in den Spalten, als auch in den Zeilen mit High-Pegel adressiert. Das bedeutet, wenn man beispielsweise die LED in der 5. Spalte und der 3. Reihe mit grün ansprechen möchte, müssen an den Datenpositionen 6 und 20 High-Pegel anliegen. Alle anderen Datenpositionen müssten Low-Pegel halten. In diesem Fall würde nur die ausgewählte LED grün leuchten und der Rest der Matrix wäre dunkel. (Datenvektor: \leftarrow 00010000 00000000 01000000 \leftarrow)

Die Schieberegister beziehen ihre Daten hauptsächlich von der Datenleitung *SER* und den zwei Taktleitungen *SCK* und *RCK*. Des Weiteren existieren die Vektorlöschen-Leitung *SCL* und die Freigabeleitung *G*. Diese 5 Datenleitungen sind mit dem Raspberry Pi verbunden und für die Steuerung der LED-Matrix verantwortlich. Weitere Informationen zu den verschiedenen Steuerleitungen sind in Tabelle 1 zusammengefasst.

3.2. Taster

Die Taster sind unterhalb der LED-Matrix angeordnet. Mit ihnen wird das Spiel gesteuert. Dabei gibt es zwei Taster, um den Coin nach rechts(Right) und links(Left) zu bewegen und einen Taster um den Coin fallen zu lassen(Enter). Der Enter-Taster hat eine Zusatzfunktion. Hält man ihn für längere Zeit gedrückt, so wird ein Reset ausgeführt und das Spiel neu gestartet.

Jeder Taster ist durch eine kleine Schaltung Hardware-Entprellt und führt eine eigene Datenleitung zum Raspberry Pi. Allerdings ist in der Software noch ein Algorithmus zum Entprellen der Taster vorhanden. Dies ist vorangegangen Entwicklungsphasen geschuldet und völlig unkritisch. In einer späteren Entwicklungsstufe sollte dieser Programmabschnitt angepasst und optimiert werden.

3.3. GPIO-Schnittstelle

An der GPIO-Schnittstelle werden alle Ein- und Ausgänge sowie die Betriebsspannung der Platine in einem Sockel zusammengefasst. An dieser Stelle muss der Raspberry Pi angeschlossen werden. Tabelle 1 gibt eine detaillierte Übersicht über alle Pins und zeigt, wie das Board mit dem Raspberry Pi zu verbinden ist.

Tabelle 1: Beschreibung der GPIO-Schnittstelle

| Pins | | Abk. | Beschreibung | Aktion |
|-------|-----|-------|---|------------------------------------|
| Board | RPi | | | |
| 1 | 14 | G | globales Ein-/Ausschalten der Schieberegisterausgänge (Tri-State) | 0...eingeschaltet 1...hochohmig |
| 2 | 15 | RCK | Takt zum Übernehmen der Schieberegister in die Ausgangsregister | 1→0...Daten übernehmen |
| 3 | 18 | SCK | Takt zum Weiterschieben der Daten durch die Schieberegister | 1→0...Daten weiterschieben |
| 4 | 23 | SCL | Pin zum Löschen des gesamten Datenvektors | 0...löschen 1...behalten |
| 5 | 24 | SER | Serieller Dateneingang | 0...Low 1...High |
| 6 | VCC | +5V | Betriebsspannung 5V | - keine - |
| 7 | GND | GND | Betriebsspannung 0V | |
| 8 | 2 | Right | Taster für Coinbewegung nach rechts | 0...nichts 1...gedrückt |
| 9 | 3 | Enter | Taster um Coin fallen zu lassen | |
| 10 | 4 | Left | Taster für Coinbewegung nach links | |

4. Software

Die Software zur Ansteuerung des 4-Gewinnt-Spiels wurde für *Raspberry Pi* auf dem Betriebssystem *Raspbian* geschrieben. Die dazu verwendete Programmiersprache ist *Python*. Da Python eine Interpretersprache ist, besteht keine Notwendigkeit, den Quellcode zu kompilieren, er wird direkt im Betriebssystem interpretiert.

Es ist möglich, den Quellcode mittels *SSH* ausgehend von einem anderen Rechner auf dem Raspberry Pi zu schreiben und zu debuggen. In diesem Projekt haben wir diese Methode angewandt. Der Vorteil dieser Methode besteht darin, dass sie es einerseits erlaubt, direkt auf das zu programmierende System zuzugreifen und andererseits muss man nicht auf die zusätzlichen Fähigkeiten eines PCs verzichten. So kann ein Arbeitsplatz sehr effizient gestaltet werden.

Die gesamte entwickelte Software des 4-Gewinnt Spiels befindet sich in der Python-Datei *VierGewinnt_v1.4.py*. Nachfolgend werden die einzelnen Komponenten dieser Software und der Ablauf des Hauptprogramms näher beschrieben.

4.1. Softwaremodule

Die Software ist in 3 Abschnitte aufgeteilt. Dabei handelt es sich um die Definition der Variablen, die Definition der Funktionen und den Ablauf des Hauptprogramms. Nachfolgend werden die Definition der Variablen und die Definition der Funktionen näher betrachtet und erläutert.

4.1.1. Definition der Variablen

Die Software beginnt mit der Definition der GPIO-Pins. Dazu wird für jeden Pin eine globale Variable definiert. Die Pin-Variablen enthalten dabei eine Zahl entsprechend des zugewiesenen Pins. Beim Aufrufen von Funktionen mit diesen Variablen wird die enthaltene Zahl verwendet, um auf den entsprechenden GPIO-Pin zu verlinken. Eine genaue Übersicht zu den GPIO-Pin-Variablen liefert Tabelle 2.

Im zweiten Programmschritt werden einige Programm-Variablen definiert. Sie werden vielfältig in den einzelnen Funktionen und im Hauptprogramm verwendet. Tabelle 3 zeigt die Programm-Variablen übersichtlich dargestellt und erklärt.

Tabelle 2: Definition der GPIO-Pins

| Variable | GPIO-Pin | In/Out | Beschreibung |
|--------------|----------|--------|---------------------------------------|
| G_NOT | 14 | Output | Ausgabe der Schieberegister erlauben |
| SCK | 15 | Output | Schiebetakt der Schieberegister-Daten |
| RCK | 18 | Output | Speichertakt der Schieberegister |
| SCLR_NOT | 23 | Output | Löschen des gesamten Datenvektors |
| SI | 24 | Output | Serieller Datenausgang |
| BUTTON_LEFT | 4 | Input | Left-Taster |
| BUTTON_ENTER | 3 | Input | Enter-Taster |
| BUTTON_RIGHT | 2 | Input | Right-Taster |

Tabelle 3: Programm-Variablen

| Variable | Beschreibung | Wert |
|----------------|--|------------------------------------|
| reset | Startet das Programm neu | 0...keine Aktion 1...Neustart |
| button_state | Gibt an, ob ein Taster gedrückt wurde | 0...nicht gedrückt 1...gedrückt |
| button_old | Speichert, welcher Taster gedrückt wurde | 2 / 3 / 4 |
| columns | Beschreibt die Spalten der LED-Matrix: 7 Duo-LEDs ergeben 14 Anoden | 14 |
| columns_unused | Nicht verwendete Spalten: $16 - 14 = 2$ (2x 8 Bit Schieberegister = 16 Spalten) Datenvektor werden 2 Nullen angehängen | [0,0] |
| rows | Beschreibt die Zeilen der LED-Matrix: 6 Duo-LEDs haben 6 Kathoden | 6 |
| rows_unused | Nicht verwendete Zeilen: $8 - 6 = 2$ (1x 8 Bit Schieberegister = 8 Zeilen) Datenvektor werden 2 Nullen angehängen | [0,0] |
| clk_delay | Wartezeit zur sicheren Erkennung der Signalflanken in s | 0.00000001 |
| data | Datenvektor Zeilen (High-Side) | 6x 14 Bit = 84 Bit |
| row | Datenvektor Spalten (Low-Side) | 6x 6 Bit = 36 Bit |
| pos | Aktuelle Position in der Matrix | 0-96 |
| pos_max | Maximale Zeilenlänge (14 Anoden $\hat{=}$ $0-6 \times 2 \rightarrow 12(+1)$) | 12 |
| player_nr | Identifiziert den aktuellen Spieler | 0...Spieler 1 1...Spieler 2 |

4.1.2. Definition der Funktionen

Um ein Programm schlank und übersichtlich zu programmieren, wurden wiederkehrende Programmbestandteile in Funktionen integriert. Nachfolgend werden alle im Programm definierten Funktionen aufgelistet und beschrieben. In Abbildung 6 werden unter den Funktionen bestehende Abhängigkeiten veranschaulicht.

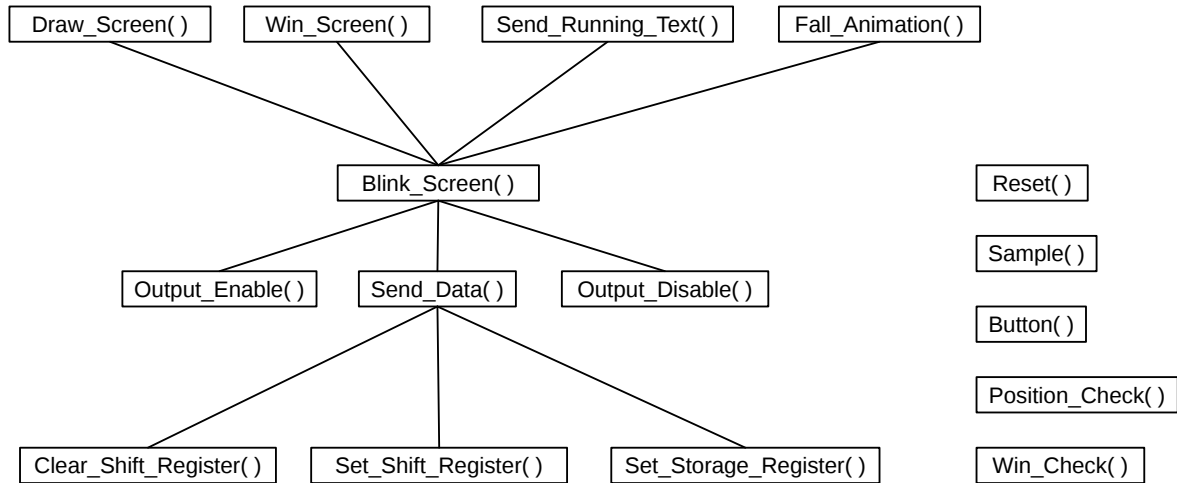


Abbildung 6: Abhängigkeiten der Funktionen

Button(button_nr)

Eingabevariable:

button_nr... kann eine der 3 Button-Variablen enthalten (2/3/4)

Beschreibung:

Gibt an, ob ein Taster gedrückt wurde. Welcher Taster geprüft werden soll, wird durch die Eingabevariable ausgewählt.

Rückgabewert:

0... Taster ist nicht gedrückt, 1... Taster ist gedrückt

global button_state... gibt an, ob Taster gedrückt wurde

global button_old... speichert, welcher Taster gedrückt wurde. Erst wenn dieser losgelassen wird, kann er erneut betätigt werden.

Output_Enable()**Eingabevariable:**
—**Beschreibung:**

Aktiviert die Tri-State-Ausgänge der Schieberegister und erlaubt das Durchschalten der Speicherregister auf die Ausgänge. Damit können die Daten der Speicherregister auf der LED-Matrix angezeigt werden. Siehe hierzu auch Anhang D Datenblatt Schieberegister 74HC595 (Auszug).

Rückgabewert:
—**Output_Disable()****Eingabevariable:**
—**Beschreibung:**

Schaltet die Tri-State-Ausgänge der Schieberegister hochohmig. Dadurch können die Speicherregister keinen Wert auf die Ausgänge schreiben. Siehe hierzu auch Anhang D Datenblatt Schieberegister 74HC595 (Auszug)

Rückgabewert:
—**Clear_Shift_Register()****Eingabevariable:**
—**Beschreibung:**

Löscht den gesamten in den Schieberegistern gespeicherten Datenvektor. Der neue Vektor enthält ausschließlich Nullen.

Rückgabewert:
—

Set_Shift_Register(data)**Eingabevariablen:**

data... zu schreibender LED-Matrix Datenvektor

Beschreibung:

Schiebt einen Datenvektor beliebiger Länge in die Schieberegister.

Rückgabewert:

—

Set_Storage_Register()**Eingabevariable:**

—

Beschreibung:

Übergibt den aktuellen Datenvektor der Schieberegister an die Speicherregister. Dort wird der aktuelle Datenvektor auf die LED-Matrix ausgegeben.

Rückgabewert:

—

Sample(sample_nr)**Eingabevariable:**

sample_nr... wählbarer Standarddatensatz (0...7)

Beschreibung:

Gibt einen Standarddatensatz für die LED-Matrix entsprechend der eingegebenen Datensatznummer zurück. (Startbildschirm, 'HI', 'DU', 'P1', 'P2', Unentschieden-Bildschirm, 'Player 2 Win', '4 Gewinnt')

Rückgabewert:

Datenvektor für LED-Matrix (84 Bit / mehr bei Lauftext - Sample 6 + 7)

Send_Data(data)

Eingabevariable:

data...Datenvektor für LED-Matrix

Beschreibung:

Schreibt einen Datenvektor auf die LED-Matrix. Der Vektor enthält ein vollständiges Bild.

Rückgabewert:

—

Position_Check(level)

Eingabevariable:

level...zum Prüfen des Bitzustands (0...Low, 1...High)

global pos...aktuelle Position im Matrix-Datenvektor

global player_nr...zeigt an, welcher Spieler an der Reihe ist (Matrixnavigation)

Beschreibung:

Prüft, ob die LED an der aktuellen Position des Matrix-Datenvektors ein- oder ausgeschaltet ist. Es wird immer nur rot oder grün eingeschaltet, niemals beide LED-Farben gemeinsam. Ist eine Farbe eingeschaltet, so gilt die LED als eingeschaltet.

Rückgabewert:

0...Bit \neq *level*, 1...Bit=*level*

Win_Check(r)

Eingabevariablen:

r...Reihe der aktuellen Matrix-Position

global pos...aktuelle Position im Matrix-Datenvektor

global data...Matrix-Datenvektor

Beschreibung:

Überprüft ausgehend von der übergebenen Position, ob das Spiel gewonnen wurde. Es wird in alle Richtungen geprüft. (4 Coins in Reihe/Spalte/Diagonal)

Rückgabewert:

0...Spiel nicht gewonnen, 1...Spiel gewonnen

global win_row...Vektor mit 4 Daten, gibt die Position des Gewinns an

Win_Screen()**Eingabevariable:**
—**Beschreibung:**

Zeigt den Gewinn eines Spielers an. Die Anzeige dauert 4s.

Rückgabewert:
—**Draw_Screen()****Eingabevariable:**
—**Beschreibung:**

Zeigt an, dass das Spiel unentschieden ausgegangen ist. Die Anzeige dauert 4s.

Rückgabewert:
—**Blink_Screen(time_length, interval, data)****Eingabevariablen:**

time_length...Anzeigezeit in Sekunden

interval...Dauer eines Blinkens in Sekunden $\left(interval < \frac{time_length}{2} \right)$

data...Datenvektor, der das anzuzeigende Bild enthält

Beschreibung:

Schreibt ein Bild entsprechend des Datenvektors *data* auf die LED-Matrix. Die Anzeige bleibt für die Zeit *time_length* erhalten und blinkt in dem Abstand *interval*. Diese Funktion sorgt nur dann für ein Blinken, wenn *interval* kürzer ist als *time_length*. Wird *interval*=0 gewählt bewirkt diese Funktion eine Daueranzeige des Datenvektor-Bildes *data* für die Zeit *time_length*.

Rückgabewert:
—

Send_Running_Text(text)**Eingabevariablen:**

text... Lauftext beliebiger Länge

Beschreibung:

Schiebt einen Lauftext entsprechend des Vektors *text* über die LED-Matrix.

Rückgabewert:

—

Fall_Animation(r)**Eingabevariablen:**

r... aktuelle Reihe

global data... Matrix-Datenvektor

Beschreibung:

Lässt einen Coin langsam durch alle LEDs nach unten fallen. Dabei wird der LED-Matrix Datenvektor *data* verändert.

Rückgabewert:

global data

Reset()**Eingabevariable:**

—

Beschreibung:

Prüft ob der Enter-Taster für 2 Sekunden gedrückt bleibt. Falls dies der Fall ist, wird die globale Variable *reset* auf 1 gesetzt (standardmäßig 0). Dieses Variable arbeitet als Flag im Hauptprogramm und löst einen Neustart des Spiels aus.

Rückgabewert:

global reset

4.2. Programmablauf

An dieser Stelle wird der Ablauf des Hauptprogramms beschrieben. Um die Übersichtlichkeit zu wahren, wird der Programmablauf in kleinen Absätzen nachvollzogen.

Definition der Variablen

Das Programm beginnt mit der Definition der Variablen. Dabei wird zwischen GPIO-Variablen und Programm-Variablen unterschieden. Die GPIO-Variablen werden verwendet, um den verschiedenen Signalleitungen je einen GPIO zuzuordnen. Tabelle 2 auf Seite 16 zeigt eine detaillierte Zuordnung der einzelnen GPIO-Pins. Die Programm-Variablen werden im Hauptprogramm und den Funktionen verwendet. Eine detaillierte Zuordnung zeigt Tabelle 3 auf Seite 16.

Definition der Funktionen

Nach den Variablen werden die Funktionen definiert. Dazu ist jede für das Programm geschriebene Funktion mit ihrem jeweiligen Quellcode nacheinander angeordnet. Um dem Interpreter deutlich zu machen, dass es sich um eine Definition handelt, bekommt jede Funktion das Präfix *def*.

Interrupt Event: Reset()

Die Funktion *Reset()* wird in einem parallelen Prozess geladen. Sie prüft, ob der Spieler einen Abbruchwunsch äußert. Auf diese Weise kann das Spiel jederzeit neu gestartet werden.

Begrüßung der Spieler

Zur Begrüßung der Spieler wird ein Lauftext „4 Gewinnt“ auf der LED-Matrix ausgegeben.

Hauptschleife und Initialisierung

An dieser Stelle beginnt die Hauptschleife des Programms. Während der Initialisierung werden die Grundeinstellungen zum Starten des Spiels getroffen. Dabei wird das Reset-Flag gelöscht, die Grundposition der Matrix eingestellt, Spieler 1 als aktiv gewählt, der Tri-State-Ausgang der Schieberegister aktiviert und der aktuelle Datenvektor in den Schieberegistern gelöscht und an die Ausgänge übernommen.

Spielschleife

Die Spielschleife wiederholt sich so lang, bis das Reset-Flag gesetzt wird. Dies kann durch den parallelen Prozess *Reset()* oder durch das Ende des Spiels geschehen. Das Spiel wird beendet, wenn ein Spieler gewonnen hat oder keine Züge mehr möglich sind, weil die Matrix zu 100% gefüllt ist.

In der Spielschleife laufen immer die folgenden Schritte nacheinander ab: Sende Daten an LED-Matrix, prüfe Left-Taster, prüfe Enter-Taster, prüfe Right-Taster. Je nachdem, ob und, wenn ja, welcher Taster betätigt wurde, arbeitet die Spielschleife eine andere Aktion ab.

Left-Taster

Der Coin wird auf der LED-Matrix um eine Position weiter nach links verschoben. Sollte der Rand der LED-Matrix erreicht worden sein, so verharrt der Coin auf seiner aktuellen Position am linken Rand der Matrix.

Enter-Taster

Der Coin wird in der aktuellen Spalte fallen gelassen. Er gleitet bis zur untersten noch von keinem Coin besetzten Stelle und verharrt an dieser. Anschließend wird überprüft, ob durch den Wurf ein Gewinn erzielt wurde. Ist das der Fall, wird eine Gewinnmeldung ausgegeben und das Spiel beendet. Falls kein Gewinn festgestellt wurde, wird nun überprüft, ob noch gültige Züge vorgenommen werden können. Sollte die LED-Matrix bereits vollständig ausgefüllt sein, ist dies nicht mehr möglich und ein Unentschiedenbildschirm wird ausgegeben und das Spiel beendet. Falls noch weitere gültige Spielzüge

möglich sind, wird der Spieler gewechselt und regulär in die Spielschleife zurück gesprungen.

Right-Taster

Der Coin wird auf der LED-Matrix um eine Position weiter nach rechts verschoben. Sollte der Rand der LED-Matrix erreicht worden sein, so verharrt der Coin auf seiner aktuellen Position am rechten Rand der Matrix.

5. Diskussion

5.1. Bekannte Probleme und Fehler

Flackerndes Display

Die LED-Matrix kann gelegentlich etwas flackern. Das liegt daran, dass die LED-Matrix vom Hauptprogramm beschrieben wird. Da je nach gewählter Aktion ein kürzeres oder längeres Hauptprogramm ausgeführt wird, ist kein gleichmäßiges Beschreiben der LED-Matrix möglich. Dieser Effekt ist jedoch sehr gering und fällt kaum ins Gewicht.

5.2. Ausblick

Parallele Prozesse und Interrupts

In der aktuellen Entwicklung könnten viele Funktionen des Hauptprogramms in Interrupts oder parallele Prozesse ausgelagert werden. So könnte beispielsweise das Beschreiben des Displays wesentlich gleichmäßiger ablaufen, wenn es in einen parallelen Prozess ausgelagert wird. Auch das Abfragen der Taster könnte durch die Verwendung von Interrupts effizienter gestaltet werden.

Computergegner

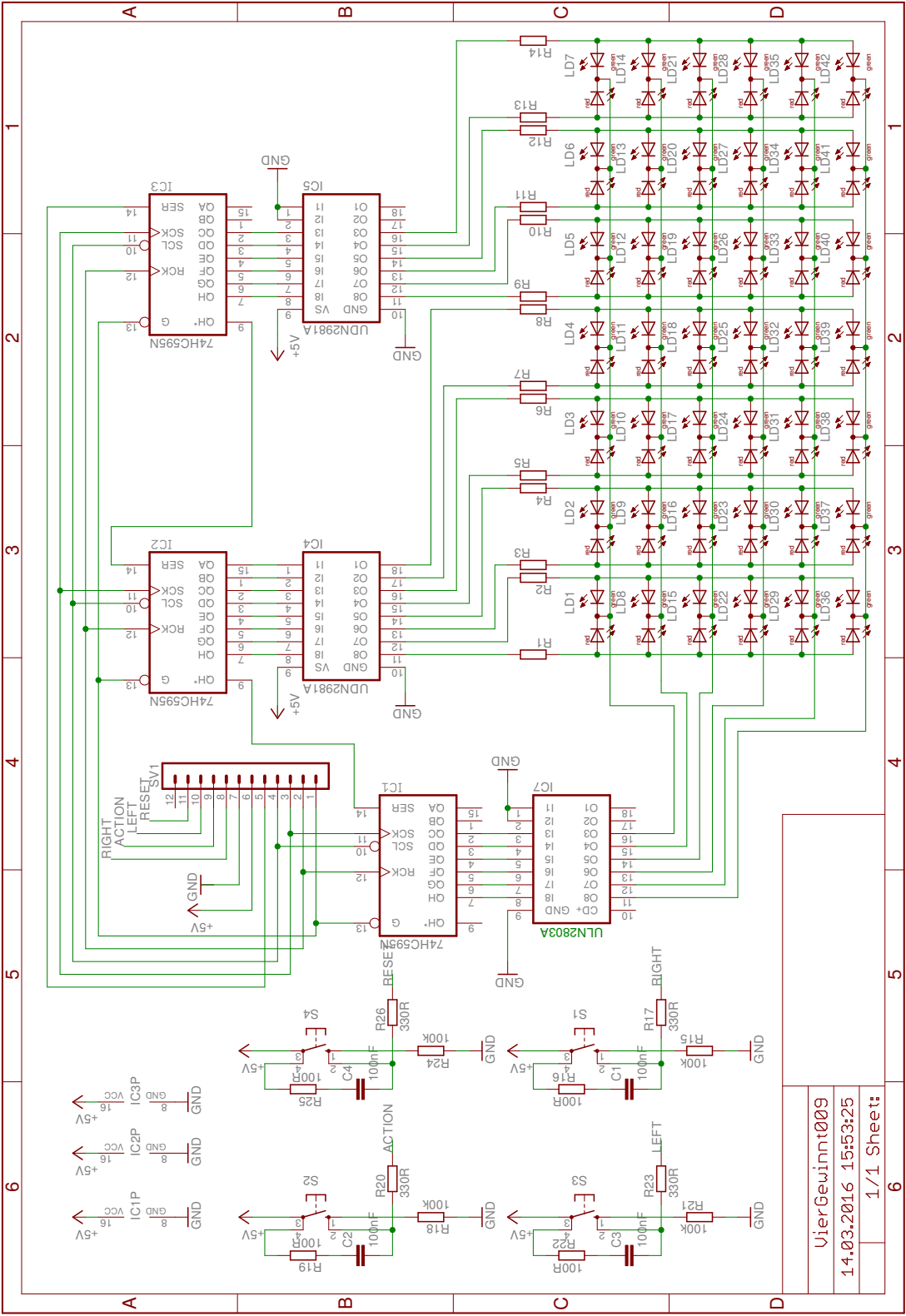
Eine schöne Erweiterung des Projekts wäre ein optionaler Computergegner. So könnte man das Spiel auch allein spielen. Des weiteren wäre auch das Implementieren eines

Demonstrationsmodus interessant. Dabei können zwei Computergegner in immer neuen Variationen gegeneinander spielen.

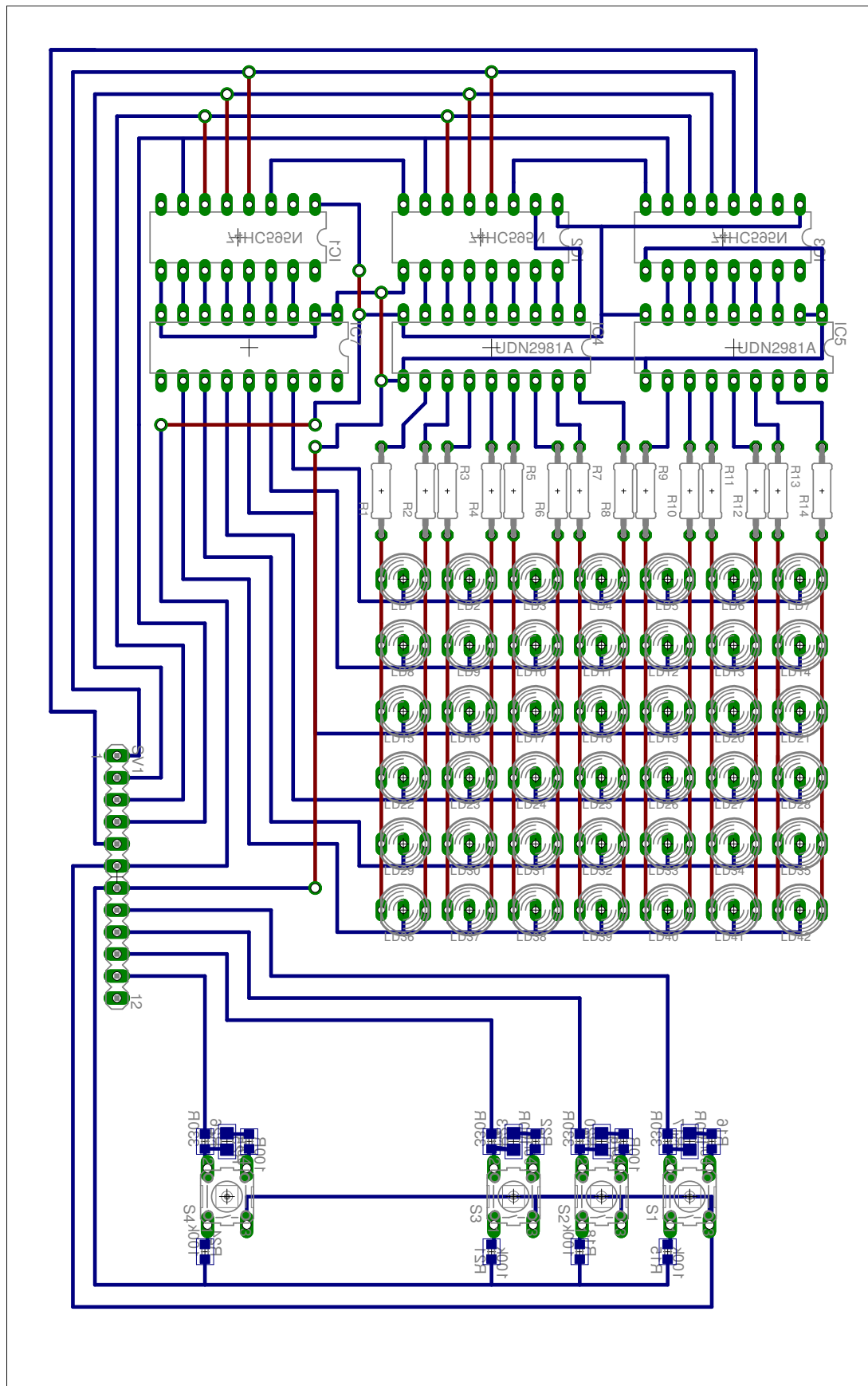
Integration weiterer Spiele

Für spätere Entwicklungen ist es vorstellbar, weitere Spiele auf der bereits vorhandenen Hardware zu programmieren. Gut geeignete Spiele wären beispielsweise Varianten von Tetris, Pong oder Snake.

A. Schaltplan



B. Platinenlayout



C. Datenblatt Duo-LED L-59EGW (Auszug)

.Kingbright®

T-1 3/4(5mm) BI-POLAR AND BI-COLOR INDICATOR LAMPS

L-59 SERIES

Features

- UNIFORM LIGHT OUTPUT.
- LOW POWER CONSUMPTION.
- MILKY WHITE DIFFUSION LENS.
- 3 LEADS WITH ONE COMMON LEAD.
- EXCEPT L-59EGW-CA IS COMMON ANODE TYPE, ALL OTHER ITEMS ARE COMMON CATHODE TYPE.
- THIRD COLOR (MIXED COLOR) AVAILABLE.
- SUPER BRIGHT VERSION AVAILABLE.
- I.C. COMPATIBLE.
- LONG LIFE - SOLID STATE RELIABILITY.

Description

The High Efficiency Red source color devices are made with Gallium Arsenide Phosphide on Gallium Phosphide Orange Light Emitting Diode.

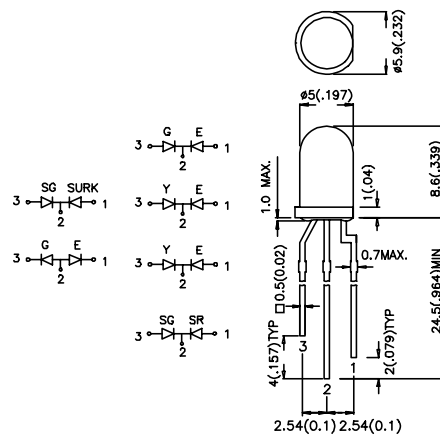
The Green source color devices are made with Gallium Phosphide Green Light Emitting Diode.

The Yellow source color devices are made with Gallium Arsenide Phosphide on Gallium Phosphide Yellow Light Emitting Diode.

The Super Bright Red source color devices are made with Gallium Aluminum Arsenide Red Light Emitting Diode.

The Hyper Red (SURK) source color devices are made with DH InGaAlP on GaAs substrate Light Emitting Diode.

Package Dimensions



Notes:

1. All dimensions are in millimeters (inches).
2. Tolerance is $\pm 0.25(0.01)$ unless otherwise noted.
3. Lead spacing is measured where the lead emerge package.
4. Specifications are subjected to change without notice.

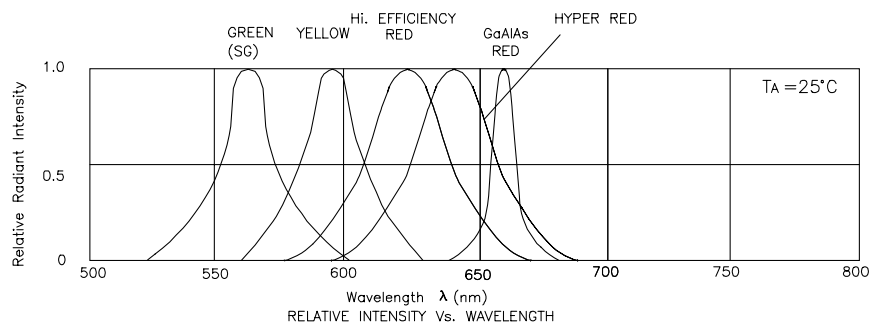
Kingbright®

Selection Guide

| Part No. | Dice | Lens Type | Iv (mcd) @ 20 mA | | Viewing Angle |
|--------------------|---------------------------------|----------------|---------------------|------|------------------|
| | | | Min. | Typ. | |
| L-59EGW | HIGH EFFICIENCY RED (GaAsP/GaP) | WHITE DIFFUSED | 20 | 60 | 60° |
| | GREEN (GaP) | | 20 | 50 | |
| L-59EGW-CA | HIGH EFFICIENCY RED (GaAsP/GaP) | WHITE DIFFUSED | 3 | 5 | 60° |
| | GREEN (GaP) | | 3 | 5 | |
| L-59EYW | HIGH EFFICIENCY RED (GaAsP/GaP) | WHITE DIFFUSED | 20 | 60 | 60° |
| | YELLOW (GaAsP/GaP) | | 20 | 40 | |
| L-59GYW | GREEN (GaP) | WHITE DIFFUSED | 20 | 50 | 60° |
| | YELLOW (GaAsP/GaP) | | 20 | 40 | |
| L-59SRSGW-CC | SUPER BRIGHT RED (GaAlAs) | WHITE DIFFUSED | 100 | 200 | 60° |
| | SUPER BRIGHT GREEN (GaP) | | 40 | 60 | |
| L-59SURKSGW | HYPER RED (InGaAlP) | WHITE DIFFUSED | 300 | 500 | 60° |
| | SUPER BRIGHT GREEN (GaP) | | 40 | 60 | |
| L-59EGC | HIGH EFFICIENCY RED (GaAsP/GaP) | WATER CLEAR | 100 | 150 | 24° |
| | GREEN (GaP) | | 50 | 100 | |
| L-59EYC L-59YEC | HIGH EFFICIENCY RED (GaAsP/GaP) | WATER CLEAR | 100 | 150 | 24° |
| | YELLOW (GaAsP/GaP) | | 30 | 60 | |
| L-59GYC | GREEN (GaP) | WATER CLEAR | 50 | 100 | 24° |
| | YELLOW (GaAsP/GaP) | | 30 | 60 | |
| L-59SRSGC-CC | SUPER BRIGHT RED (GaAlAs) | WATER CLEAR | 300 | 500 | 24° |
| | SUPER BRIGHT GREEN (GaP) | | 80 | 100 | |
| L-59SURKSGC | HYPER RED (InGaAlP) | WATER CLEAR | 900 | 2000 | 24° |
| | SUPER BRIGHT GREEN (GaP) | | 80 | 100 | |

Note:

1. $\theta_{1/2}$ is the angle from optical centerline where the luminous intensity is 1/2 the optical centerline value.



Electrical / Optical Characteristics at $T_A=25^{\circ}\text{C}$

| Symbol | Parameter | Device | Typ. | Max. | Units | Test Conditions |
|-------------------------|-------------------------|---|---|--|---------------|-----------------|
| λ_{peak} | Peak Wavelength | High Efficiency Red Green Super Bright Green Yellow Super Bright Red Hyper Red | 625 565 565 590 660 640 | | nm | IF=20mA |
| $\Delta\lambda_{1/2}$ | Spectral Line Halfwidth | High Efficiency Red Green Super Bright Green Yellow Super Bright Red Hyper Red | 45 30 30 35 20 25 | | nm | IF=20mA |
| C | Capacitance | High Efficiency Red Green Super Bright Green Yellow Super Bright Red Hyper Red | 12 45 45 10 95 35 | | pF | VF=0V;f=1MHz |
| V_F | Forward Voltage | High Efficiency Red Green Super Bright Green Yellow Super Bright Red Hyper Red | 2.0 2.2 2.2 2.1 1.85 2.0 | 2.5 2.5 2.5 2.5 2.5 2.2 | V | IF=20mA |
| I_R | Reverse Current | All | | 10 | μA | VR = 5V |

Absolute Maximum Ratings at $T_A=25^{\circ}\text{C}$

| Parameter | High Efficiency Red | Green | Yellow | Super Bright Green | Super Bright Red | Hyper Red | Units |
|--------------------------------|---------------------|-------|--------|--------------------|------------------|-----------|-------|
| Power dissipation | 105 | 105 | 105 | 105 | 100 | 170 | mW |
| DC Forward Current | 30 | 25 | 30 | 25 | 30 | 50 | mA |
| Peak Forward Current [1] | 150 | 150 | 150 | 150 | 150 | 150 | mA |
| Reverse Voltage | 5 | 5 | 5 | 5 | 5 | 5 | V |
| Operating/Storage Temperature | -40°C To +85°C | | | | | | |
| Lead Soldering Temperature [2] | 260°C For 5 Seconds | | | | | | |

Notes:

1. 1/10 Duty Cycle, 0.1ms Pulse Width.
2. 4mm below package base.

D. Datenblatt Schieberegister 74HC595 (Auszug)

SN54HC595, SN74HC595 8-BIT SHIFT REGISTERS WITH 3-STATE OUTPUT REGISTERS

SCLS041G – DECEMBER 1982 – REVISED FEBRUARY 2004

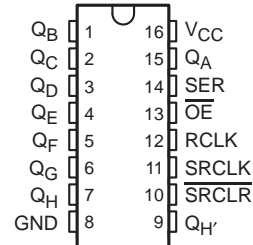
- 8-Bit Serial-In, Parallel-Out Shift
- Wide Operating Voltage Range of 2 V to 6 V
- High-Current 3-State Outputs Can Drive Up To 15 LSTTL Loads
- Low Power Consumption, 80- μ A Max I_{CC}
- Typical $t_{pd} = 13$ ns
- ± 6 -mA Output Drive at 5 V
- Low Input Current of 1 μ A Max
- Shift Register Has Direct Clear

description/ordering information

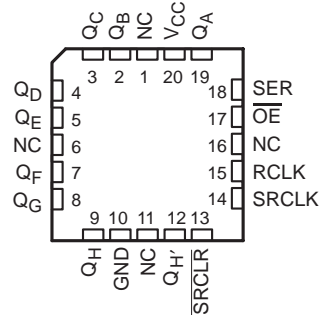
The 'HC595 devices contain an 8-bit serial-in, parallel-out shift register that feeds an 8-bit D-type storage register. The storage register has parallel 3-state outputs. Separate clocks are provided for both the shift and storage register. The shift register has a direct overriding clear (SRCLR) input, serial (SER) input, and serial outputs for cascading. When the output-enable (\overline{OE}) input is high, the outputs are in the high-impedance state.

Both the shift register clock (SRCLK) and storage register clock (RCLK) are positive-edge triggered. If both clocks are connected together, the shift register always is one clock pulse ahead of the storage register.

SN54HC595 . . . J OR W PACKAGE SN74HC595 . . . D, DB, DW, N, OR NS PACKAGE (TOP VIEW)



SN54HC595 . . . FK PACKAGE (TOP VIEW)



NC – No internal connection

ORDERING INFORMATION

| T_A | PACKAGE† | | ORDERABLE PART NUMBER | TOP-SIDE MARKING |
|----------------|-----------|--------------|-----------------------|------------------|
| –40°C to 85°C | PDIP – N | Tube of 25 | SN74HC595N | SN74HC595N |
| | SOIC – D | Tube of 40 | SN74HC595D | HC595 |
| | | Reel of 2500 | SN74HC595DR | |
| | | Reel of 250 | SN74HC595DT | |
| | SOIC – DW | Tube of 40 | SN74HC595DW | HC595 |
| | | Reel of 2000 | SN74HC595DWR | |
| –55°C to 125°C | SOP – NS | Reel of 2000 | SN74HC595NSR | HC595 |
| | SSOP – DB | Reel of 2000 | SN74HC595DBR | HC595 |
| | CDIP – J | Tube of 25 | SNJ54HC595J | SNJ54HC595J |
| | CFP – W | Tube of 150 | SNJ54HC595W | SNJ54HC595W |
| | LCCC – FK | Tube of 55 | SNJ54HC595FK | SNJ54HC595FK |

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

**TEXAS
INSTRUMENTS**

POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Copyright © 2004, Texas Instruments Incorporated
On products compliant to MIL-PRF-38535, all parameters are tested unless otherwise noted. On all other products, production processing does not necessarily include testing of all parameters.

SN54HC595, SN74HC595
8-BIT SHIFT REGISTERS
WITH 3-STATE OUTPUT REGISTERS

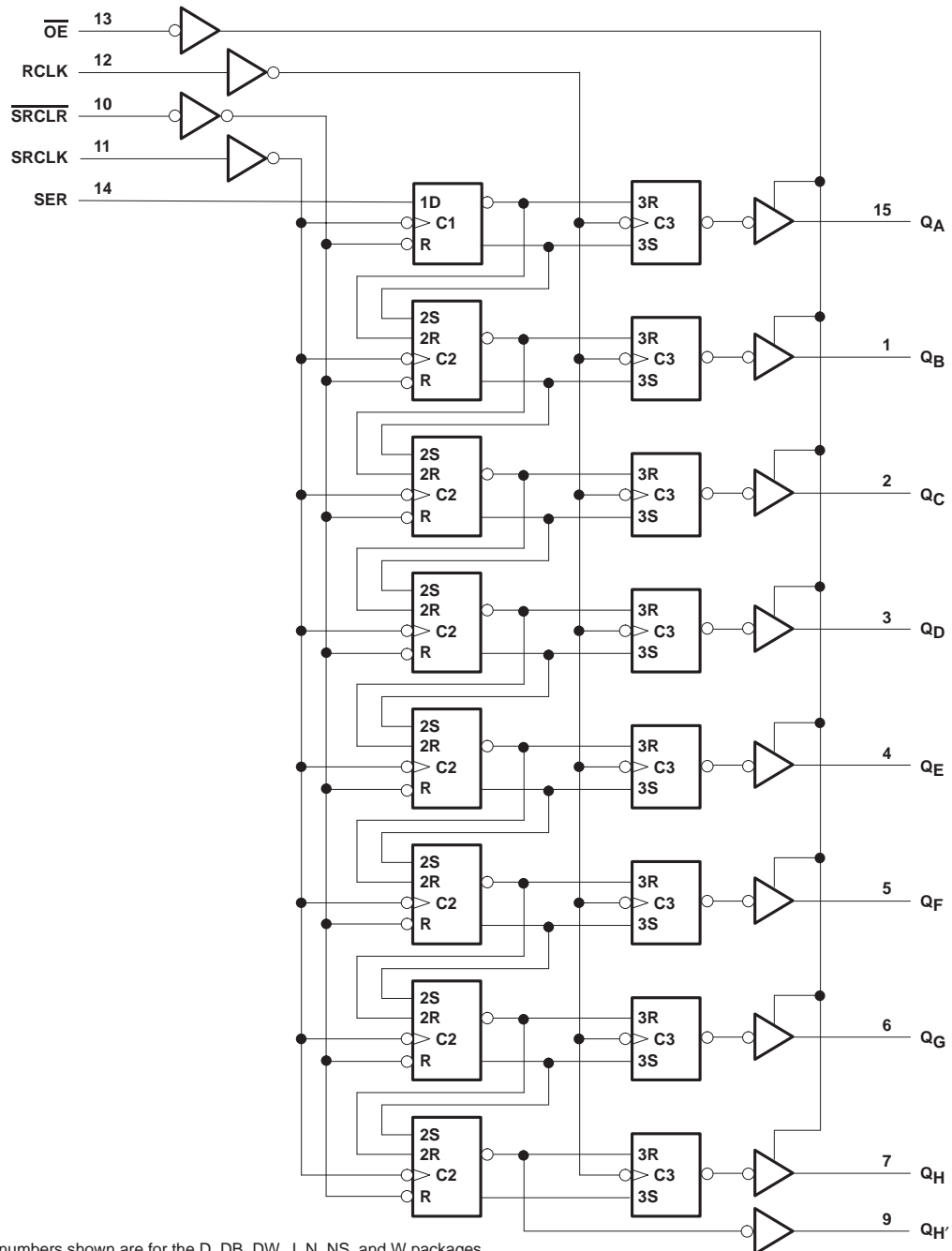
SCLS041G – DECEMBER 1982 – REVISED FEBRUARY 2004

FUNCTION TABLE

| INPUTS | | | | | FUNCTION |
|--------|-------|---------------------------|------|------------------------|--|
| SER | SRCLK | $\overline{\text{SRCLR}}$ | RCLK | $\overline{\text{OE}}$ | |
| X | X | X | X | H | Outputs Q_A – Q_H are disabled. |
| X | X | X | X | L | Outputs Q_A – Q_H are enabled. |
| X | X | L | X | X | Shift register is cleared. |
| L | ↑ | H | X | X | First stage of the shift register goes low. Other stages store the data of previous stage, respectively. |
| H | ↑ | H | X | X | First stage of the shift register goes high. Other stages store the data of previous stage, respectively. |
| X | X | X | ↑ | X | Shift-register data is stored in the storage register. |

SN54HC595, SN74HC595
8-BIT SHIFT REGISTERS
WITH 3-STATE OUTPUT REGISTERS
SCLS041G – DECEMBER 1982 – REVISED FEBRUARY 2004

logic diagram (positive logic)



Pin numbers shown are for the D, DB, DW, J, N, NS, and W packages.

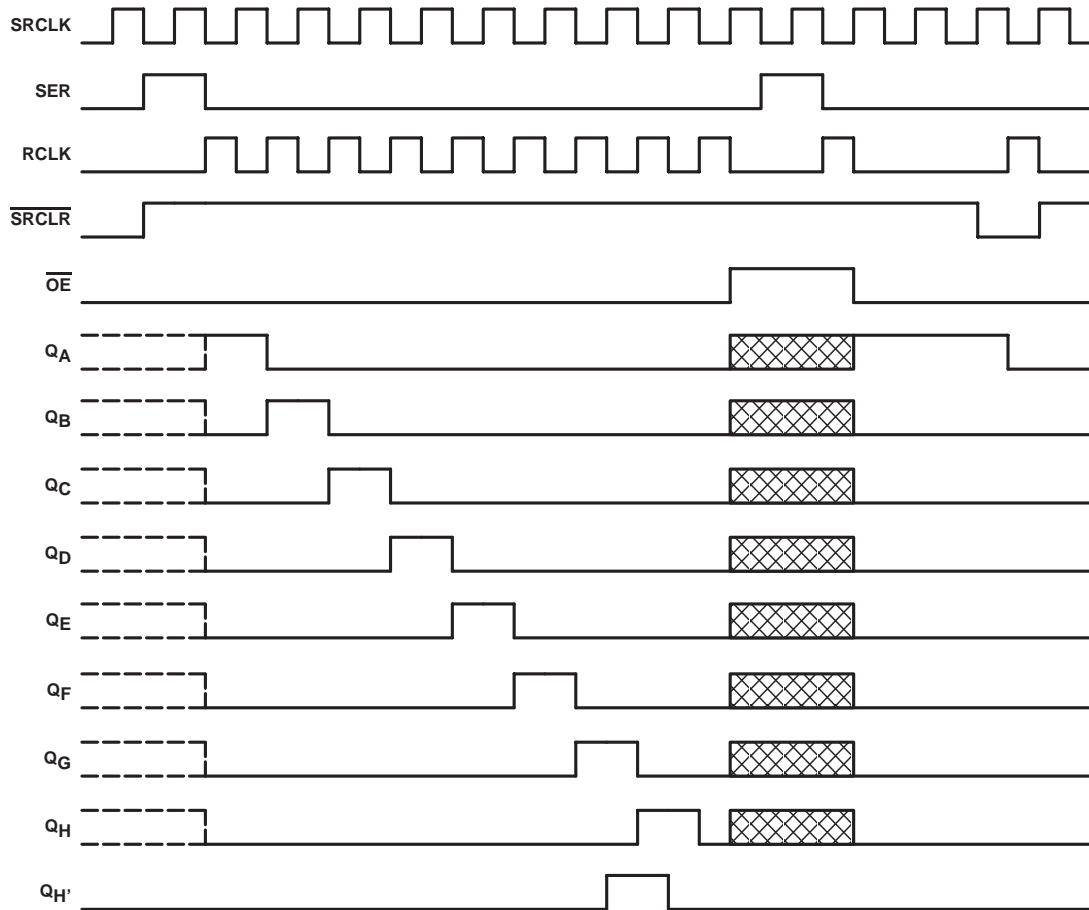



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

SN54HC595, SN74HC595
8-BIT SHIFT REGISTERS
WITH 3-STATE OUTPUT REGISTERS

SCLS041G – DECEMBER 1982 – REVISED FEBRUARY 2004

timing diagram



NOTE:  implies that the output is in 3-State mode.

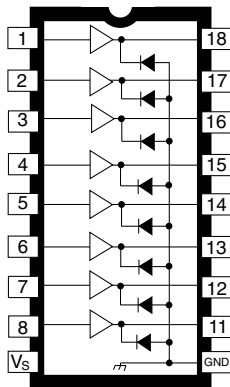
E. Datenblatt High-Side-Treiber UDN2981 (Auszug)

2981 THRU 2984

Data Sheet
29310D*

8-CHANNEL SOURCE DRIVERS

UDN2981A thru UDN2984A



Dwg. No. A-10, 243

Note that the UDN2980A series (dual in-line package) and UDN2980LW series (small-outline IC package) are electrically identical and share a common terminal number assignment.

ABSOLUTE MAXIMUM RATINGS at 25°C Free-Air Temperature

| | |
|--|------------------------|
| Output Voltage Range, V_{CE} | |
| (UDN2981A, UDN2982A, UDN2982LW, and A2982SLW) . . . | 5 V to 50 V |
| (UDN2983A, UDN2984A, UDN2984LW, and A2984SLW) . . . | 35 V to 80 V |
| Input Voltage, V_{IN} | |
| (UDN2981A and UDN2983A) | 15 V |
| (UDN2982A, UDN2984A, UDN2982LW, UDN2984LW, A2982SLW, and A2984SLW) | 20 V |
| Output Current, I_{OUT} | -500 mA |
| Package Power Dissipation, P_D | See Graph |
| Operating Temperature Range, T_A | -20°C to +85°C |
| Storage Temperature Range, T_S | -55°C to +150°C |

Recommended for high-side switching applications that benefit from separate logic and load grounds, these devices encompass load supply voltages to 80 V and output currents to -500 mA. These 8-channel source drivers are useful for interfacing between low-level logic and high-current loads. Typical loads include relays, solenoids, lamps, stepper and/or servo motors, print hammers, and LEDs.

All devices may be used with 5 V logic systems — TTL, Schottky TTL, DTL, and 5 V CMOS. The UDN2981A, UDN2982A, UDN2982LW, and A2982SLW are electrically interchangeable, will withstand a maximum output off voltage of 50 V, and operate to a minimum of 5 V; the UDN2983A, UDN2984A, UDN2984LW, and A2984SLW drivers are electrically interchangeable, will withstand an output voltage of 80 V, and operate to a minimum of 35 V. All devices in this series integrate input current limiting resistors and output transient suppression diodes, and are activated by an active high input.

The suffix 'A' (all devices) indicates an 18-lead plastic dual in-line package with copper lead frame for optimum power dissipation. Under normal operating conditions, these devices will sustain 120 mA continuously for each of the eight outputs at an ambient temperature of +50°C and a supply of 15 V.

The suffix 'LW' (UDN2982LW and UDN2984LW only) indicates an 18-lead surface-mountable wide-body SOIC package; the A2982SLW and A2984SLW are provided in a 20-lead wide-body SOIC package with improved thermal characteristics.

The UDN2982A, UDN2982LW, A2982SLW, UDN2984A, UDN2984LW, and A2984SLW drivers are also available for operation over an extended temperature range to -40°C. To order, change the prefix 'UDN' to 'UDQ' or the suffix 'SLW' to 'ELW'.

FEATURES

- TTL, DTL, PMOS, or CMOS Compatible Inputs
- 500 mA Output Source Current Capability
- Transient-Protected Outputs
- Output Breakdown Voltage to 80 V
- DIP or SOIC Packaging

Always order by complete part number, e.g., **UDN2981A**.
Note that all devices are not available in all package styles.



2981 THRU 2984
8-CHANNEL
SOURCE DRIVERS

ELECTRICAL CHARACTERISTICS at $T_A = +25^\circ\text{C}$ (unless otherwise specified).

| Characteristic | Symbol | Applicable Devices | Test Conditions | Test Fig. | Limits | | | Units |
|--------------------------------------|---------------|--------------------|--|-----------|--------|------|------|---------------|
| | | | | | Min. | Typ. | Max. | |
| Output Leakage Current | I_{CEX} | 2981/82† | $V_{IN} = 0.4\text{ V}^*$, $V_S = 50\text{ V}$, $T_A = +70^\circ\text{C}$ | 1 | — | — | 200 | μA |
| | | 2983/84† | $V_{IN} = 0.4\text{ V}^*$, $V_S = 80\text{ V}$, $T_A = +70^\circ\text{C}$ | 1 | — | — | 200 | μA |
| Output Sustaining Voltage | $V_{CE(SUS)}$ | 2981/82† | $I_{OUT} = -45\text{ mA}$ | — | 35 | — | — | V |
| | | 2983/84† | $I_{OUT} = -70\text{ mA}$ | — | 45 | — | — | V |
| Collector-Emitter Saturation Voltage | $V_{CE(SAT)}$ | All | $V_{IN} = 2.4\text{ V}$, $I_{OUT} = -100\text{ mA}$ | 2 | — | 1.6 | 1.8 | V |
| | | | $V_{IN} = 2.4\text{ V}$, $I_{OUT} = -225\text{ mA}$ | 2 | — | 1.7 | 1.9 | V |
| | | | $V_{IN} = 2.4\text{ V}$, $I_{OUT} = -350\text{ mA}$ | 2 | — | 1.8 | 2.0 | V |
| Input Current | $I_{IN(ON)}$ | 2981/83A | $V_{IN} = 2.4\text{ V}$ | 3 | — | 140 | 200 | μA |
| | | | $V_{IN} = 3.85\text{ V}$ | 3 | — | 310 | 450 | μA |
| | | 2982/84† | $V_{IN} = 2.4\text{ V}$ | 3 | — | 140 | 200 | μA |
| | | | $V_{IN} = 12\text{ V}$ | 3 | — | 1.25 | 1.93 | mA |
| Output Source Current (Outputs Open) | I_{OUT} | 2981/83A | $V_{IN} = 2.4\text{ V}$, $V_{CE} = 2.0\text{ V}$ | 2 | -350 | — | — | mA |
| | | 2982/84† | $V_{IN} = 2.4\text{ V}$, $V_{CE} = 2.0\text{ V}$ | 2 | -350 | — | — | mA |
| Supply Current Leakage Current | I_S | 2981/82† | $V_{IN} = 2.4\text{ V}^*$, $V_S = 50\text{ V}$ | 4 | — | — | 10 | mA |
| | | 2983/84† | $V_{IN} = 2.4\text{ V}^*$, $V_S = 80\text{ V}$ | 4 | — | — | 10 | mA |
| Clamp Diode Forward Voltage | I_R | 2981/82† | $V_R = 50\text{ V}$, $V_{IN} = 0.4\text{ V}^*$ | 5 | — | — | 50 | μA |
| | | 2983/84† | $V_R = 80\text{ V}$, $V_{IN} = 0.4\text{ V}^*$ | 5 | — | — | 50 | μA |
| Clamp Diode | V_F | All | $I_F = 350\text{ mA}$ | 6 | — | 1.5 | 2.0 | V |
| Turn-On Delay | t_{ON} | All | 0.5 E_{IN} to 0.5 E_{OUT} , $R_L = 100\Omega$, $V_S = 35\text{ V}$ | — | — | 1.0 | 2.0 | μs |
| Turn-Off Delay | t_{OFF} | All | 0.5 E_{IN} to 0.5 E_{OUT} , $R_L = 100\Omega$, $V_S = 35\text{ V}$, See Note | — | — | 5.0 | 10 | μs |

NOTES: Turn-off delay is influenced by load conditions. Systems applications well below the specified output loading may require timing considerations for some designs, i.e., multiplexed displays or when used in combination with sink drivers in a totem pole configuration.

Negative current is defined as coming out of (sourcing) the specified device terminal.

* All inputs simultaneously.

† Complete part number includes a prefix (A or UDN) and a suffix (A or SLW) as follows:

UDN2981A,
UDN2982A, UDN2982LW, or A2982SLW,
UDN2983A,
UDN2984A, UDN2984LW, or A2984SLW.

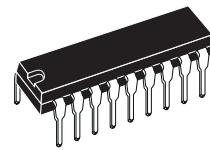
F. Datenblatt Low-Side-Treiber ULN2803 (Auszug)



ULN2801A
ULN2802A - ULN2803A
ULN2804A - ULN2805A

EIGHT DARLINGTON ARRAYS

- EIGHT DARLINGTONS WITH COMMON EMITTERS
- OUTPUT CURRENT TO 500 mA
- OUTPUT VOLTAGE TO 50 V
- INTEGRAL SUPPRESSION DIODES
- VERSIONS FOR ALL POPULAR LOGIC FAMILIES
- OUTPUT CAN BE PARALLELED
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY BOARD LAYOUT



DIP18

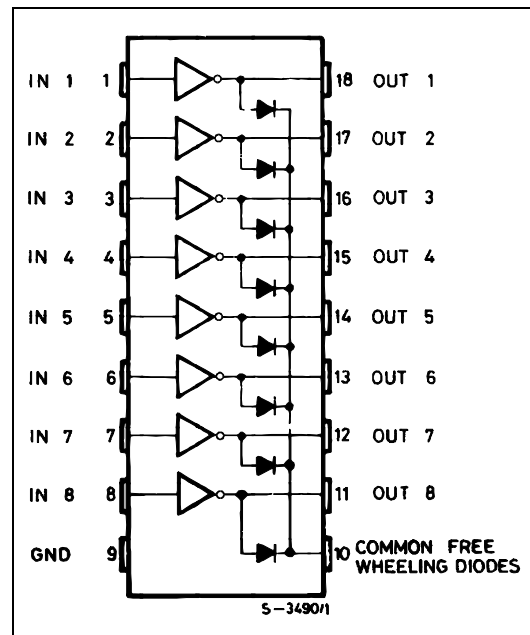
DESCRIPTION

The ULN2801A-ULN2805A each contain eight darlington transistors with common emitters and integral suppression diodes for inductive loads. Each darlington features a peak load current rating of 600mA (500mA continuous) and can withstand at least 50V in the off state. Outputs may be paralleled for higher current capability.

Five versions are available to simplify interfacing to standard logic families : the ULN2801A is designed for general purpose applications with a current limit resistor ; the ULN2802A has a 10.5k Ω input resistor and zener for 14-25V PMOS ; the ULN2803A has a 2.7k Ω input resistor for 5V TTL and CMOS ; the ULN2804A has a 10.5k Ω input resistor for 6-15V CMOS and the ULN2805A is designed to sink a minimum of 350mA for standard and Schottky TTL where higher output current is required.

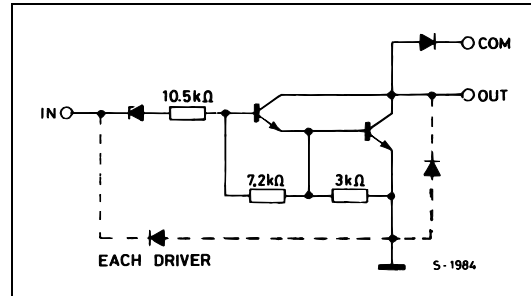
All types are supplied in a 18-lead plastic DIP with a copper lead from and feature the convenient input-opposite-output pinout to simplify board layout.

PIN CONNECTION (top view)

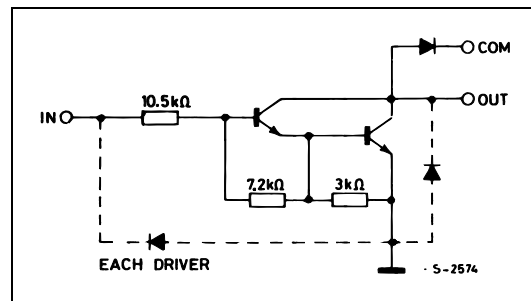


SCHEMATIC DIAGRAM AND ORDER CODES

For ULN2802A (each driver for 14-15 V PMOS)



For ULN2804A (each driver for 6-15 V
CMOS/PMOS



ULN2801A - ULN2802A - ULN2803A - ULN2804A - ULN2805A**ABSOLUTE MAXIMUM RATINGS**

| Symbol | Parameter | Value | Unit |
|-----------|--|-------------|------|
| V_o | Output Voltage | 50 | V |
| V_i | Input Voltage for ULN2802A, UL2803A, ULN2804A for ULN2805A | 30 15 | V |
| I_C | Continuous Collector Current | 500 | mA |
| I_B | Continuous Base Current | 25 | mA |
| P_{tot} | Power Dissipation (one Darlington pair) (total package) | 1.0 2.25 | W |
| T_{amb} | Operating Ambient Temperature Range | - 20 to 85 | °C |
| T_{stg} | Storage Temperature Range | - 55 to 150 | °C |
| T_J | Junction Temperature Range | - 20 to 150 | °C |

THERMAL DATA

| Symbol | Parameter | Value | Unit |
|-----------------|---|-------|------|
| $R_{th\ j-amb}$ | Thermal Resistance Junction-ambient Max. | 55 | °C/W |

ELECTRICAL CHARACTERISTICS ($T_{amb} = 25^\circ\text{C}$ unless otherwise specified)

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit | Fig. |
|---------------|--------------------------------------|---|------|----------------------------------|--|--|--------------------------|
| I_{CEX} | Output Leakage Current | $V_{CE} = 50V$ $T_{amb} = 70^\circ\text{C}$, $V_{CE} = 50V$ $T_{amb} = 70^\circ\text{C}$ for ULN2802A $V_{CE} = 50V$, $V_i = 6V$ for ULN2804A $V_{CE} = 50V$, $V_i = 1V$ | | | 50 100 500 500 | μA μA μA μA | 1a 1a 1b 1b |
| $V_{CE(sat)}$ | Collector-emitter Saturation Voltage | $I_C = 100\text{mA}$, $I_B = 250\mu\text{A}$ $I_C = 200\text{mA}$, $I_B = 350\mu\text{A}$ $I_C = 350\text{mA}$, $I_B = 500\mu\text{A}$ | | 0.9 1.1 1.3 | 1.1 1.3 1.6 | V V V | 2 |
| $I_{i(on)}$ | Input Current | for ULN2802A $V_i = 17V$ for ULN2803A $V_i = 3.85V$ for ULN2804A $V_i = 5V$ $V_i = 12V$ for ULN2805A $V_i = 3V$ | | 0.82 0.93 0.35 1 1.5 | 1.25 1.35 0.5 1.45 2.4 | mA mA mA mA mA | 3 |
| $I_{i(off)}$ | Input Current | $T_{amb} = 70^\circ\text{C}$, $I_C = 500\mu\text{A}$ | 50 | 65 | | μA | 4 |
| $V_{i(on)}$ | Input Voltage | $V_{CE} = 2V$ for ULN2802A $I_C = 300\text{mA}$ for ULN2803A $I_C = 200\text{mA}$ $I_C = 250\text{mA}$ $I_C = 300\text{mA}$ for ULN2804A $I_C = 125\text{mA}$ $I_C = 200\text{mA}$ $I_C = 275\text{mA}$ $I_C = 350\text{mA}$ for ULN2805A $I_C = 350\text{mA}$ | | | 13 2.4 2.7 3 5 6 7 8 2.4 | V V V V V V V V V | 5 |
| h_{FE} | DC Forward Current Gain | for ULN2801A $V_{CE} = 2V$, $I_C = 350\text{mA}$ | 1000 | | | — | 2 |
| C_i | Input Capacitance | | | 15 | 25 | pF | — |
| t_{PLH} | Turn-on Delay Time | $0.5 V_i$ to $0.5 V_o$ | | 0.25 | 1 | μs | — |
| t_{PHL} | Turn-off Delay Time | $0.5 V_i$ to $0.5 V_o$ | | 0.25 | 1 | μs | — |
| I_R | Clamp Diode Leakage Current | $V_R = 50V$ $T_{amb} = 70^\circ\text{C}$, $V_R = 50V$ | | | 50 100 | μA μA | 6 6 |
| V_F | Clamp Diode Forward Voltage | $I_F = 350\text{mA}$ | | 1.7 | 2 | V | 7 |



