Tastenfelder abfragen

Stand: 18. 7. 2011

Die Einzelabfrage vieler Kontakte (z . B. eines umfangreicheren Bedienfeldes) erfordert entsprechend viele Signalleitungen und Anschlüsse am abfragenden Schaltkreis (z. B. Mikrocontroller). Ein Ausweg ist die Organisation als Kontaktmatrix (Abb. 1). Eine Matrix mit n Zeilen und m Spalten erfordert n + m Signalleitungen, es können aber m • n Kontakte abgefragt werden.

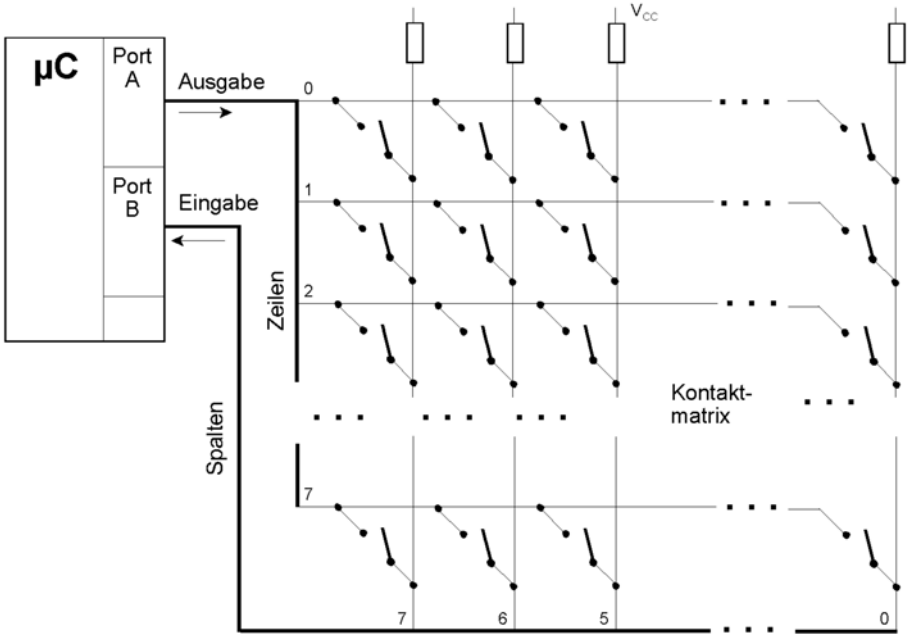

Abb. 1 Kontaktmatrix. Hier mit acht Zeilen und acht Spalten (für maximal 64 Kontakte). Zum Abfragen werden zwei 8-Bit-Ports eines Mikrocontrollers verwendet. Im Beispiel ist die Abfrage aktiv Low organisiert.

Die abfragende Einrichtung (z. B. ein Mikrocontroller[1)]) führt eine Abfrageschleife (Polling Loop) aus, die zyklisch nachfragt, welche Tasten gerade betätigt werden. Im Beispiel erregt der Mikrocontroller Zeilenleitung für Zeilenleitung und fragt dann jeweils über die Spaltenleitungen ab, welche Kontakte geschlossen sind.

Die Abfrage kann aktiv Low oder aktiv High organsiert werden. Demgemäß sind die Spaltenleitungen an Pull-up- oder an Pull-down-Widerstände angeschlossen.

Führt eine Zeilenleitung den aktiven Pegel, so bewirkt eine an diese Leitung angeschlossene betätigte Taste, daß die betreffende Spaltenleitung ebenfalls auf diesen Pegel gezogen wird.

1: Es gibt auch Abfrageschaltkreise, die nach dem gleichen Prinzip arbeiten (Beispiele: Fairchild MM74C922/923).

Ablaufbeispiel (vgl. Abb. 1):

- anfänglich werden alle Zeilenleitungen mit High-Pegeln belegt,
- die Zeilenleitung 0 wird mit einem Low-Pegel belegt,
- die Spalten 7...0 werden abgefragt (das ergibt den Zustand der ersten Zeile),
- die Zeilenleitung 0 wird mit einem High-Pegel belegt,
- die Zeilenleitung 1 wird mit einem Low-Pegel belegt,
- die Spalten 7...0 werden abgefragt (das ergibt den Zustand der zweiten Zeile),
- die Zeilenleitung 1 wird mit einem High-Pegel belegt,
- die Zeilenleitung 2 wird mit einem Low-Pegel belegt usw.

Das einfache Prinzip ist aber mit Spitzfindigkeiten behaftet.

Spitzfindigkeit 1: scheinbare Tastenbetätigungen (Phantom Keys)
Es können mehr Tasten als betätigt erscheinen als tatsächlich betätigt sind (Abb. 2). Der Effekt tritt dann auf, wenn wenigstens drei Tasten betätigt sind, und zwar zwei in einer Spalte sowie eine weitere in einer weiteren Spalte.

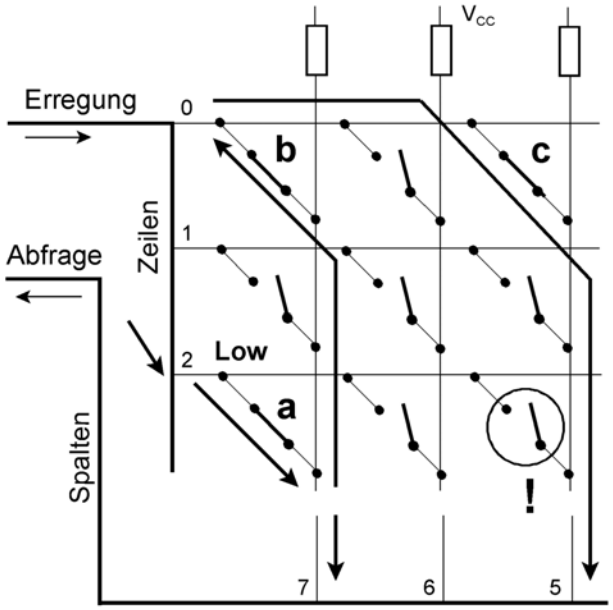

Abb. 2 Scheinbare Tastenbetätigungen (Phantom Keys). Die Tasten a, b und c werden gleichzeitig betätigt. Die Zeile 2 wird erregt (Low-Pegel). Die gedrückte Taste a verbindet Spaltenleitung 7 mit Zeilenleitung 2. Spaltenleitung 7 führt damit auch Low-Pegel. Die ebenfalls gedrückte Taste b schaltet den Low-Pegel von Spaltenleitung 7 zur Zeilenleitung 0 durch (die an sich gar nicht erregt ist). Die weitere gedrückte Taste c verbindet Zeilenleitung 0 mit Spaltenleitung 5, so daß diese ebenfalls Low-Pegel führt. Der Mikrocontroller sieht somit Low-Pegel auf den Spaltenleitungen 7 und 5. Der Pegel auf Spaltenleitung 7 ist korrekt, weil von Taste a bewirkt; der auf Spaltenleitung 5 ist falsch, weil in Zeile 2 die zu Spalte 5 gehörende Taste gar nicht betätigt ist.

Abhilfe:

1. Die Tastenbelegung nur dann auswerten, wenn lediglich eine Taste oder zwei Tasten als betätigt erkannt werden[2)].
2. Aufteilung des Tastenfeldes in mehrere unbhängige Blöcke.
3. Zwischenschalten von Dioden.

Spitzfindigkeit 2: Kurzschluß zwischen Treiberstufen

Sind die Zeilenleitungen an übliche binäre Gegentaktteiber angeschlossen, kann es beim gleichzeitigen Betätigen mehrerer Tasten zu einem Konfliktfall kommen (Abb. 3).

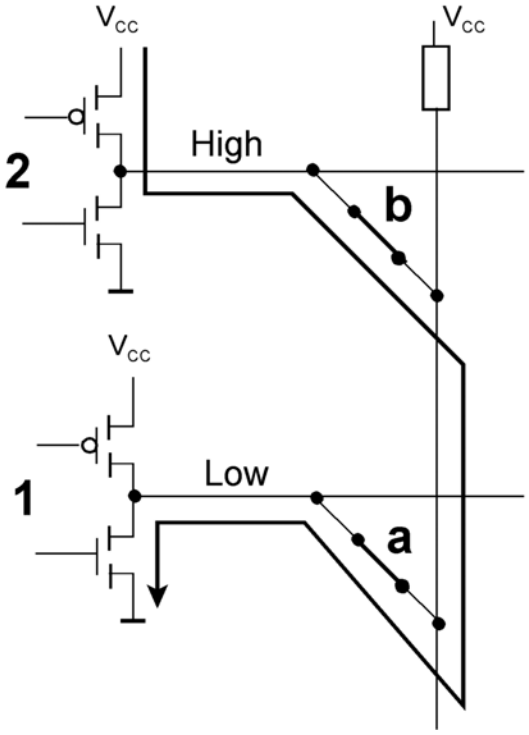

Abb. 3 Ein Konfliktfall. 1, 2 sind Treiberstufen für Zeilenleitungen. Stufe 1 wird auf Low getrieben, Stufe 2 auf High. Die Tasten a und b sind betätigt. Hiermit ergibt sich ein Stromweg von Stufe 2 (High ca. V_{CC}) nach Stufe 1 (Low ca. GND). Das ist nahezu ein Kurzschluß (Überlastung der Treiberstufen).

Abhilfe:

1. Strombegrenzungswiderstände in den Zeilenleitungen (Abb. 4),
2. Sperrdioden in den Zeilenleitungen (Abb. 5),
3. Open-Collector-Treiber oder Nachbildung des Open-Collector-Verhaltens mit Tri-State-Stufen (Abb. 6),
4. Dioden in der Kontaktmatrix (Abb. 7).

2: Oftmals begnügt man sich damit, nur eine einzige betätigte Taste auszuwerten. Wird mehr als eine Taste gedrückt, geschieht nichts. Beispiel: Taschenrechner.

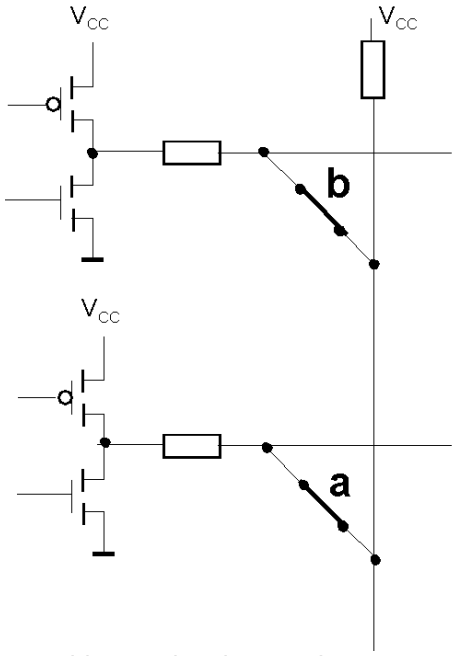

Abb. 4 Strombegrenzungswiderstände. Eine Sparlösung. Den Spannungsteiler aus dem Ziehwiderstand der Spaltenleitung und dem Begrenzungswiderstand so dimensionieren, daß der aktive Pegel (hier Low) noch in zulässigen Grenzen liegt. Werden mehrere Tasten gedrückt, ist das nicht immer einzuhalten, da die Ziehwiderstände dann parallelgeschaltet werden.

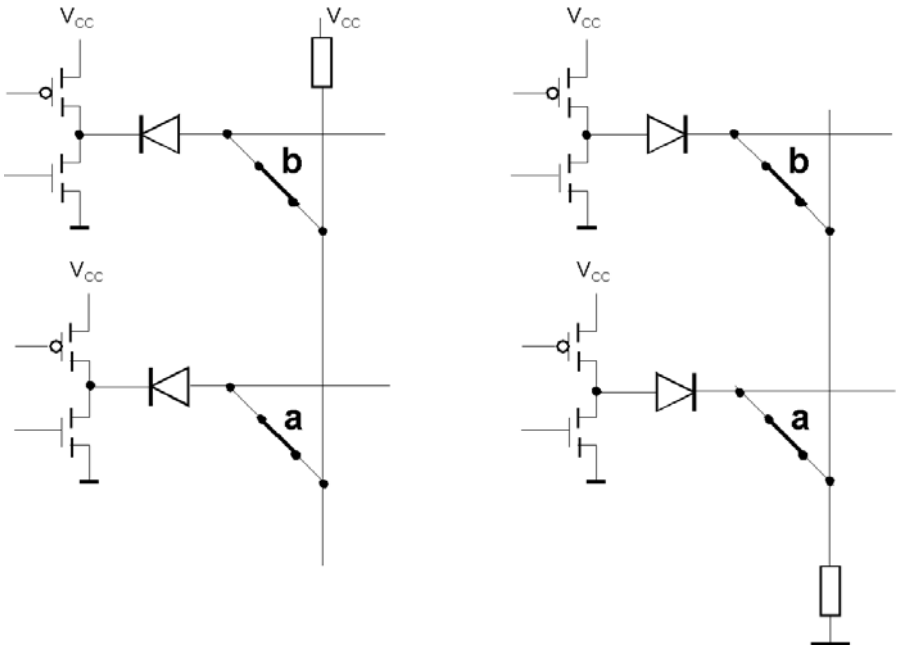

Abb. 5 Sperrdioden. Links für aktiv Low, rechts für aktiv High. Der Strom kann nur über die aktive Treiberstufe fließen. Das Problem: inaktive Zeilenleitungen hängen praktisch in der Luft (schwimmendes Potential). Ungünstig für die EMV-Prüfung...

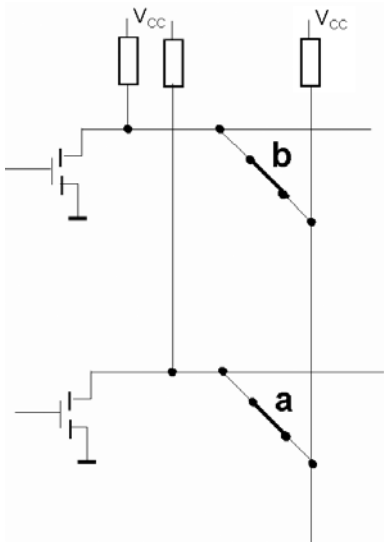

Abb. 6 Open-Collector/Open-Drain-Treiber. Eine funktionell saubere Lösung, die bei Mikrocontrollern mit Tri-State-Ports etwas Programmierarbeit macht. Zu den Feinheiten s. Seite 7.

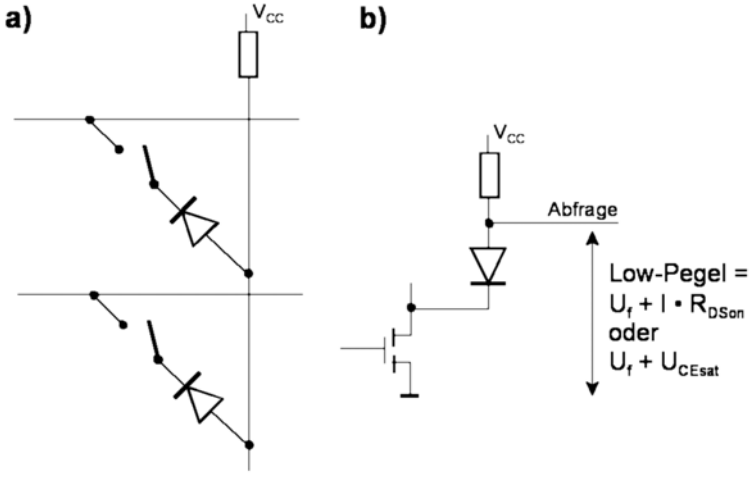

Abb. 7 Kontaktmatrix mit Dioden. Die im Grunde beste, aber teuerste Lösung. Die Diode läßt den Strom nur von der Spalten- zur Zeilenleitung fließen, und zwar – in der hier gezeigten Schaltung – nur dann, wenn die Zeilenleitung mit Low belegt ist. a) Prinzipschaltung. b) Ersatzschaltung einer auf Low gezogenen Diode. Deren Flußspannung addiert sich zum Low-Pegel – was manchmal ein Problem sein kann. Abhilfe: z. B. mit Schmitt-Trigger-Eingängen.

Die Auslegung gemäß Abb. 7 vermeidet beide Spitzfindigkeiten von Grund auf. Die in den Abb. 2 und 3 veranschaulichten Stromwege werden gesperrt. In derartigen Kontaktmatrizen werden beliebig viele gleichzeitig betätigte Kontakte korrekt erkannt. Somit ist es möglich, nicht nur Tasten, sondern auch rastende Schalter in die Matrix aufzunehmen. Eine Diodenmatrix mit festen Verbindungen oder Auftrennungen ergibt einen der einfachsten Festwertspeicher (ROM; Abb. 8 und 9).

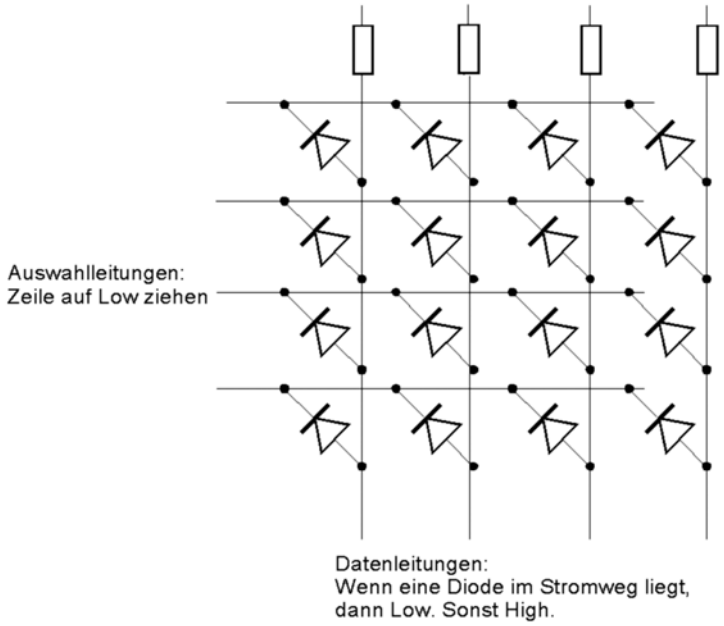

Abb. 8 Die Diodenmatrix – der ROM des kleinen Mannes.

Ein solcher ROM kann folgendermaßen programmiert werden:

- mit Schaltern oder Jumpern in den Stromwegen (vgl. Abb. 7a und 8),
- indem die Dioden bestückt oder nicht bestückt werden.

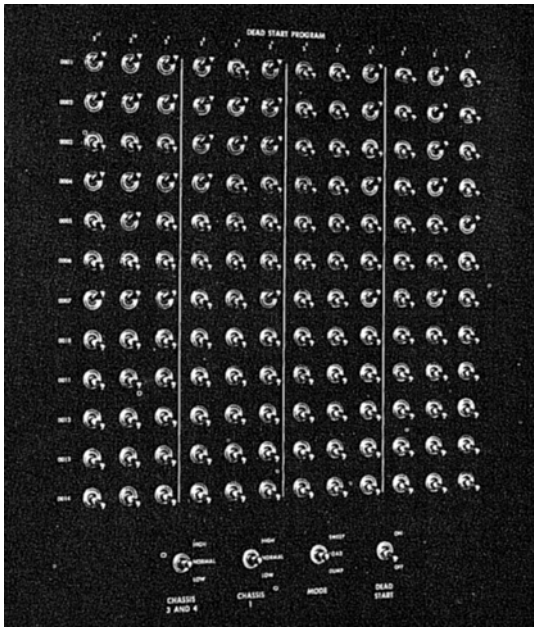

Abb. 9 Ein legendärer ROM. Das Deadstart-Panel der CDC 6600. Mit 144 Kippschaltern konnten 12 Befehle zu 12 Bits Länge eingestellt werden. Das mußte für den Anfang reichen...

Beispiel einer Tastenmatrix mit Open-Collector-Treibern:

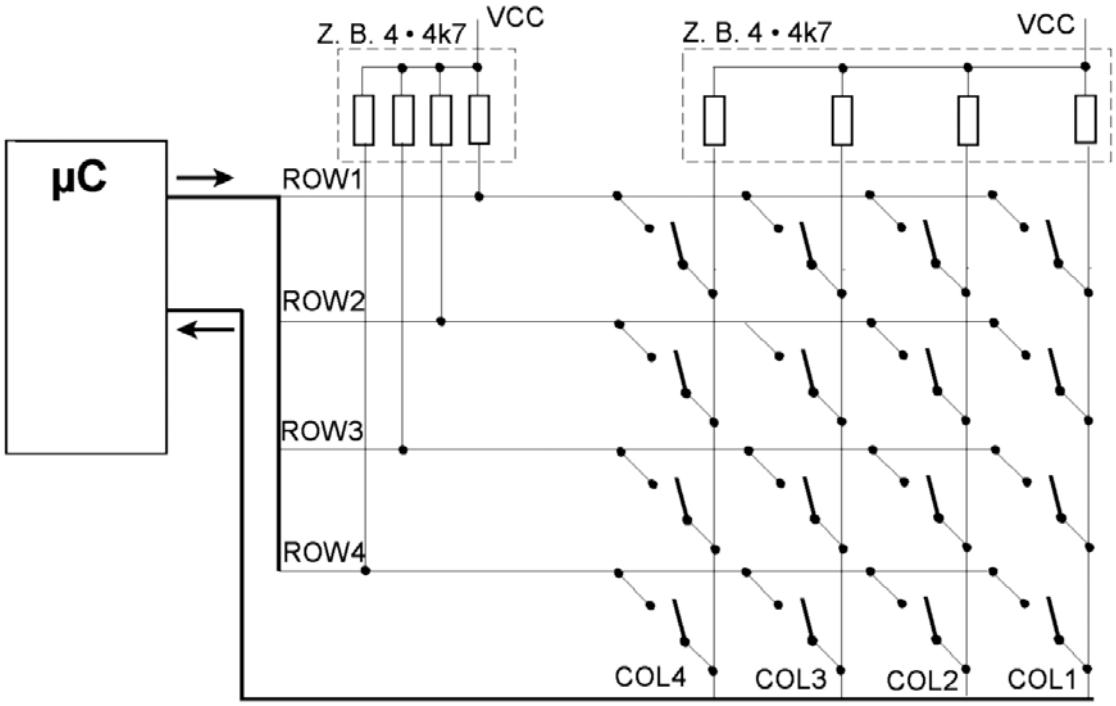

Abb. 10 Praxisschaltung einer Tastenmatrix.

Aufpassen:

- Maximaler Treiberstrom > Pullup-Widerstand der Zeile + alle Pullup-Widerstände der Spalten (werden Tasten mehrerer Spalten gleichzeitig betätigt, so werden deren Pull-up-Widerstände parallelgeschaltet).
- Werden die Widerstände aber zu hochohmig, sind sie Low-High-Flanken womöglich nicht steil genug.

Das Open-Collector-Verhalten mit Tri-State-Treibern nachbilden
Die Signalpegel werden nicht über das Datenregister, sondern über das Richtungssteuerregister geschaltet.

1. Datenregisterbits immer auf Null (Vorbereitung zum Treiben).

2. Richtungssteuerung auf Eingabe (Freigeben; Signal wird vom Widerstand gezogen).

3. Zum Treiben Richtungssteuerung auf Ausgabe. Die Null im Datenregister erscheint am Ausgang.

4. Nach dem Einlesen der Spaltensignale Richtungssteuerung wieder auf Ausgabe.

Achtung:

1. Der Trick funktioniert nur, wenn die Bits im Datenregister unter allen Umständen stehenbleiben (Beispiel Atmel AVR).

2. Jedes Schalten nach High kostet Zeit, denn es ist ein Umladen der parasitären Kapazitäten über die Pull-up-Widerstände. Genügend Abstand lassen, so daß sich nichts überschneidet (Abb. 11 bis 13).

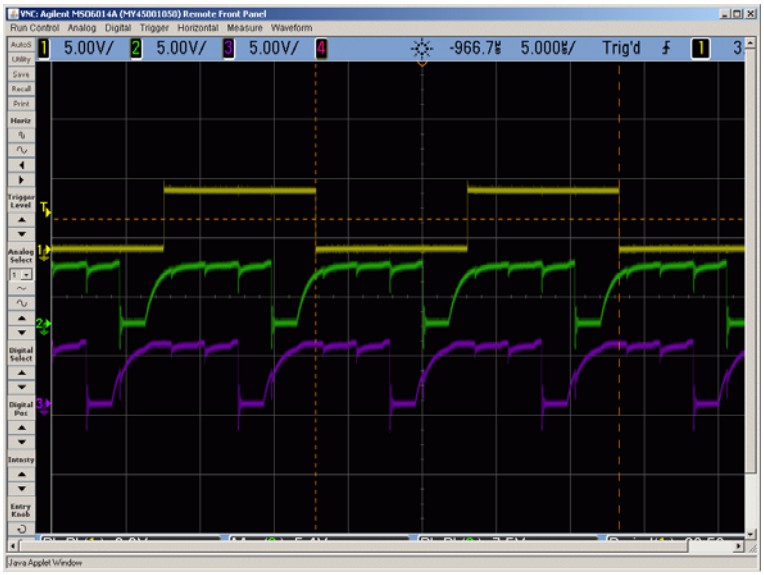

Abb. 11 Zwei Zeilen werden nacheinander aktiviert. Hier offensichtlich zu zeitig.

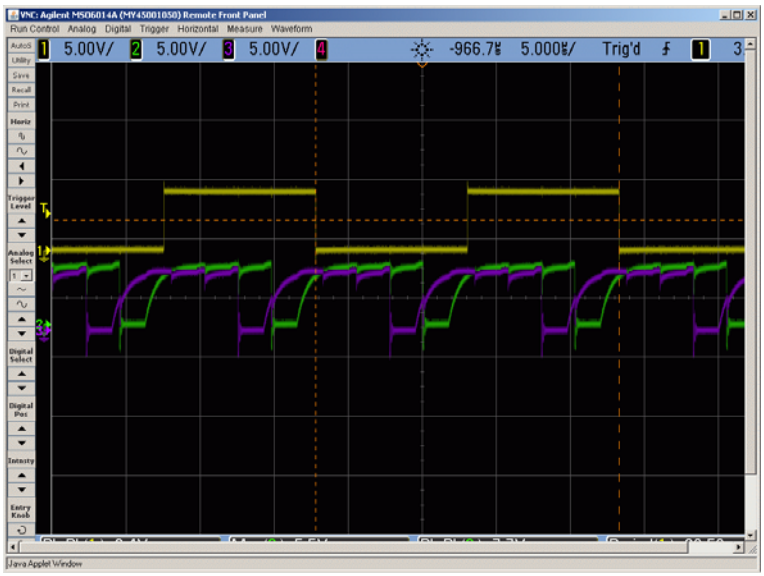

Abb. 12 Wenn man die Kurven ineinander schreibt, wird die Überschneidung besonders deutlich. So funktioniert es nicht...

Abb. 13 Jetzt ist genügend Abstand (keine Überschneidung). Der obere Rechteckimpuls veranschaulicht die Dauer eines gesamten Abfragezyklus (vier Zeilen). Die Low-High-Flanken liegen noch im Bereich des Zulässigen (Anstiegszeit).

Praxistips zur Programmierung:

1. Alles einlesen (z. B. in ein Array Zeilen • Spalten), dann auswerten.
2. Die Grundlage der Auswertung ist eine Funktion, die die Anzahl der Einsen im Array zählt[3)].
3. Ruhezustand: keine einzige Eins.
4. Betätigung; nur auswerten, wenn nur eine einzige Eins vorkommt. Deren Position als Scan-Code ausgeben.
5. Entprellen (durch Abwarten; typische Entprellzeit um 10 ms).
6. So lange abfragen, bis wiederum keine einzige Taste betätigt (Taste losgelassen).
7. Entprellen.
8. Zurück zum Anfang.

Es versteht sich von selbst, daß man tricksen und Zeit sparen kann. Dieser Ablauf (Abb. 14) hat aber – als Folge der klaren Funktionentrennung – den Vorteil, daß er auch dann ohne Schwierigkeiten abprogrammiert werden kann, wenn die Signale gleichsam verstreut an verschiedene Ports angeschlossen sind.

3: Unter der Voraussetzung, daß für betätigte Tasten Einsen gespeichert werden. Die Funktion "Anzahl der Einsen" (Number of Occurrences, Population Count) braucht man immer wieder. Es ist also keine gestohlene Lebenszeit, einen solchen Algorithmus zu programmieren. Praxistip: Am besten geht es mit einer Wertetabelle, beispielsweise für 4, 6 oder 8 Bits.

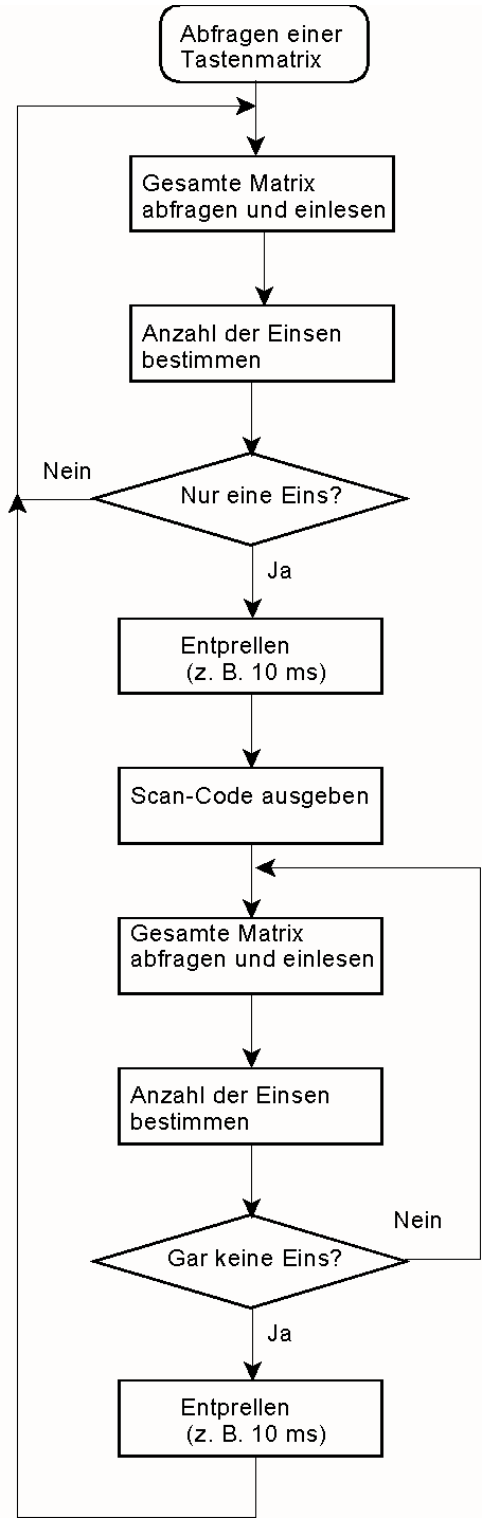

Abb. 14 Die Tastenabfrage im Flußdiagramm.

Ohne viel Zeitverlust abfragen und dabei auch noch Strom sparen
Die meiste Zeit ihres Lebens werden die Tasten nicht betätigt. Das ständige Abfragen kostet Rechenzeit und Strom.

Lösung 1:

1. Alle Zeilen aktivieren.
2. Darauf warten, daß wenigstens ein Spaltensignal aktiv wird. Hierzu Spaltensignale an Interrupteingänge (z. B. Pin Change Interrupt) anschließen. Ggf. die Signale zu einem einzigen Interruptsignal disjunktiv zusammenfassen (UND/NAND, wenn aktvl Low, ODER/NOR, wenn aktiv High).
3. Dann erst genauer nachfragen, welche Taste(n) eigentlich betätigt wurde(n).

Abwandung:
Statt der Interruptauslösung eine ganz einfache Abfrageschleife. Alle Zeilen aktiv. Nur abfragen, ob alle Spalten inaktiv sind oder nicht. Es genügt, in längeren Abständen nachzufragen, z. B. alle 100 ms (Versuchssache).

Lösung 2:
Wenn nichts betätigt ist (Ruhezustand), das Abfrageintervall verlängern, z. B. auf einige hundert ms. Hierzu ggf. die Watchdog-Funktionen scharfmachen und den Mikrocontroller in den Stromsparzustand versetzen (SLEEP-Befehl). Ist der Watchdog-Zeitzähler abgelaufen, wird der Mikrocontroller wieder aktiviert. Das Behandlungsprogramm fragt jetzt ab, ob eine Taste betätigt wurde (kann so implementiert werden, wie vorstehend beschrieben). Wurde keine betätig, wird der Stromsparzustand wieder eingeschaltet usw. Das Verfahren ist in Anwendungsschriften der Mikrocontrollerhersteller ausführlich beschreiben.