Die Anwendungsprogrammierung der Mikrocontroller
Ein einführender Überblick

Die weitaus meisten Anwendungsaufgaben der Steuerungs-, Regelungs- und Automatisierungstechnik werden heutzutage mit Mikrocontrollern gelöst. Der Mikrocontroller ist ein vollständiger Computer in einem einzigen Schaltkreis. Programmspeicher, Datenspeicher und die Funktionseinheiten der Ein- und Ausgabe sind eingebaut. Somit stehen alle E-A-Anschlüsse zum Lösen der Anwendungsaufgabe zur Verfügung (Abb. 1). Die einfachsten Systeme enthalten einen einzigen Mikrocontroller, dessen Anschlüsse direkt mit der jeweiligen Anwendungsumgebung verbunden sind, also mit Sensoren, Leistungsstufen, Bedienelementen, Anzeigeeinrichtungen usw. (Abb. 2). Die Anwendungsfunktionen werden vom Programm erbracht, das im Mikrocontroller läuft. Es fragt alle Eingangsbelegungen ab und stellt alle Ausgangsbelegungen ein.

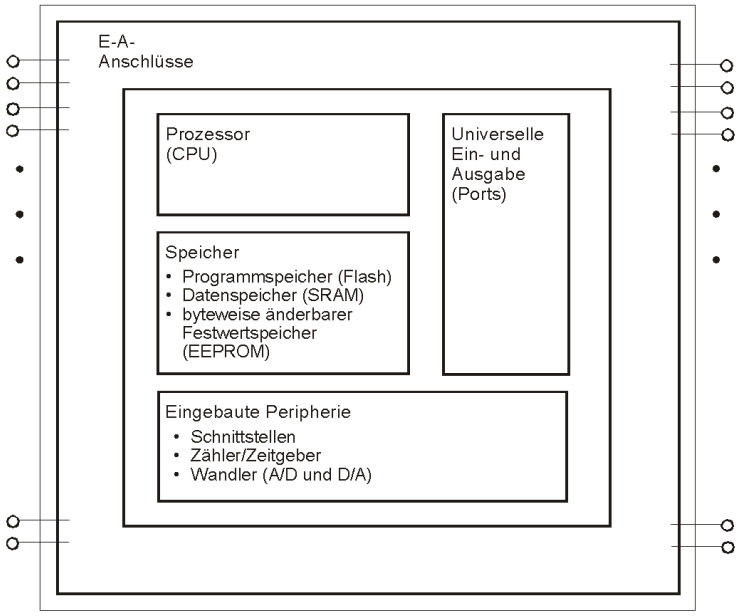


Abb. 1 Ein Mikrocontroller. Alle Funktionseinheiten befinden sich im selben Schaltkreis.

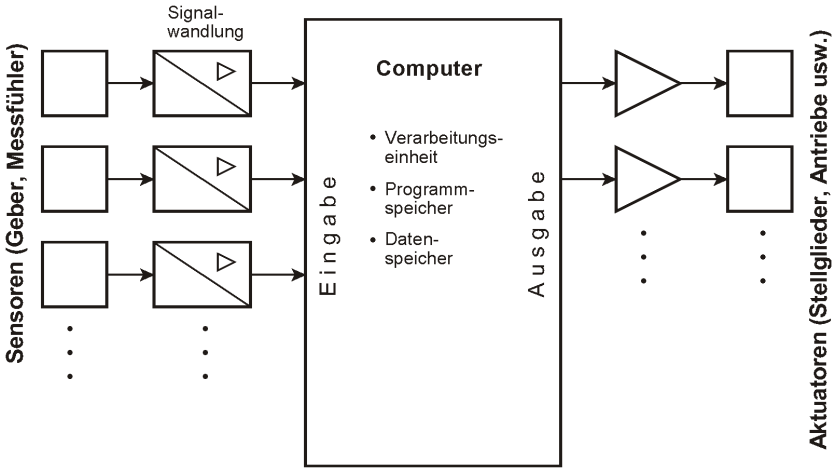


Abb. 2 Heuzutage steht der Mikrocontroller im Mittelpunkt.

Die meisten Mikrocontroller werden nicht deswegen eingesetzt, weil es etwas zu rechnen gibt, sondern nur, um bestimmte Funktionen billiger zu erledigen als dies mit zweckgebundenen Schaltungen möglich wäre. In solchen Fällen kommt es oftmals auf geringste Hardwarekosten an. Der Schaltkreis ist soweit wie irgend möglich auszunutzen. Des zwingt oftmals zur maschinennahen Programmierung, zur Nutzung ungewöhnlicher Programmiertechniken – und nicht selten zum Tricksen auf Biegen und Brechen.

Typische Anwendungsaufgaben führen auf grundsätzliche Programmschleifen, die folgende Schritte enthalten (Abb. 3):

1. Eingabe. Die Eingänge lesen (Sensoren, Bedienfelder usw.).
2. Verarbeitung. Die eigentlichen Verarbeitungs- und Steuerabläufe ausführen. Hierbei werden der Folgezustand und die Ausgangsdaten bestimmt.
3. Ausgabe. Die Ausgänge schreiben (Anzeigen, Leistungsstufen usw.).
4. Zurück zu Schritt 1. Beim nächsten Durchlauf arbeitet die Schleife mit dem Folgezustand, der in Schritt 2 ermittelt wurde.

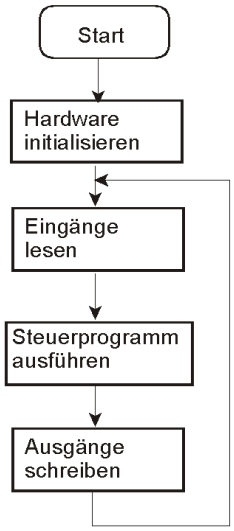


Abb. 3 Die typische Programmschleife einer Anwendungslösung.

Atmel AVR – das Lehrbeispiel

Ein Lehrbeispiel soll überschaubar sein; es muß möglich sein, sich in kurzer Zeit einzuarbeiten und nach wenigen Stunden erste Erfolge zu erzielen. Die AVR-Mikrocontroller der Fa. Atmel haben sich in dieser Hinsicht als sehr zweckmäßig erwiesen. Wichtig ist, wieviel Zeit der Lernende braucht, um vom Zustand der vollkommenen Kenntnislosigkeit bis zum ersten Verständnis und zu ersten Erfolgserlebnissen zu kommen, also zu Programmen, die wirklich laufen. Andere Mikrocontrollertypen haben ein beträchtlich höheres Leistungsvermögen und eine bei weitem umfangreichere Ausstattung. Sie sind aber auch viel komplizierter. Die Einarbeitung vom Stand Null an ist nicht in wenigen Stunden zu schaffen. Sie kostet mehrere Wochen am Stück (Richt- und Erfahrungswert: wenigstens vier). Um die E-A-Ports eines modernen Hochleistungsprozessors auch nur zu initalisieren, braucht man bereits mehrere Seiten Programmtext. Die E-A-Ports und E-A-Einrichtungen der AVR-Mikrocontroller sind hingegen vergleichsweise einfach. Ihre Wirkungsweise ist leicht zuverstehen, und es genügen einige Programmzeilen, um sie zu initialisieren.

Ein- und Ausgabe

Die technischen Mittel, die dazu dienen, einen Universalrechner mit der Außenwelt zu verbinden, werden als Ein- und Ausgabeeinrichtungen (E-A-Einrichtungen, I/O Devices) bezeichnet. Die einfachsten E-A-Abläufe beruhen auf Maschinenbefehlen, die Daten in E-A-Einrichtungen transportieren (Ausgabebefehle) oder aus solchen Einrichtungen abholen (Eingabebefehle). Grundsätzlich kann man folgende Arten von E-A-Einrichtungen unterscheiden:

- Universelle E-A-Anschlüsse, die zu E-A-Ports zusammengefasst sind. Die E-A-Ports der typischen Mikrocontroller sind so ausgelegt, dass jeder einzelne Anschluss wahlweise als Eingang oder als Ausgang betrieben werden kann.
- Periphere Einrichtungen für typische Aufgaben der Zeitdarstellung, Impulserzeugung, Ablaufsteuerung, Signalwandlung usw.
- Anwendungsspezifische periphere Einrichtungen.
- Besonders kostengünstige universelle Erweiterungsschnittstellen. Hierzu gehören unter anderem die serielle Schnittstelle, SPI, der I^2C-Bus und der CAN-Bus.
- Besonders leistungsfähigeuniverselle Erweiterungsschnittstellen, wie beispielsweise PCI, USB oder Ethernet.

Mikrocontroller sind eigens entwickelt worden, um angeschlossene Einrichtungen auf möglichst kostengünstige Weise zu steuern. Die meisten Anwendungen sind eigentlich E-A-Programme. Es ist wichtig, die E-A-Einrichtungen von den Maschinenbefehlen aus freizügig ansprechen zu können. Die kürzeste Zeiteinheit der Ein- und Ausgabe ist der Maschinentakt. Die größtmögliche Flexibilität ist dann gegeben, wenn alle Anschlüsse programmseitig zugänglich sind (Abb. 4). Die elementare Mikrocontrollerschnittstelle ist somit der bitweise frei programmierbare E-A-Port. Die weitere Ausstattung mit Zählern und Zeitgebern, Wandlern, Schnittstellen usw. richtet sich nach den Bedürfnissen jener Anwendungsgebiete, in denen die höchsten Stückzahlen nachgefragt werden.

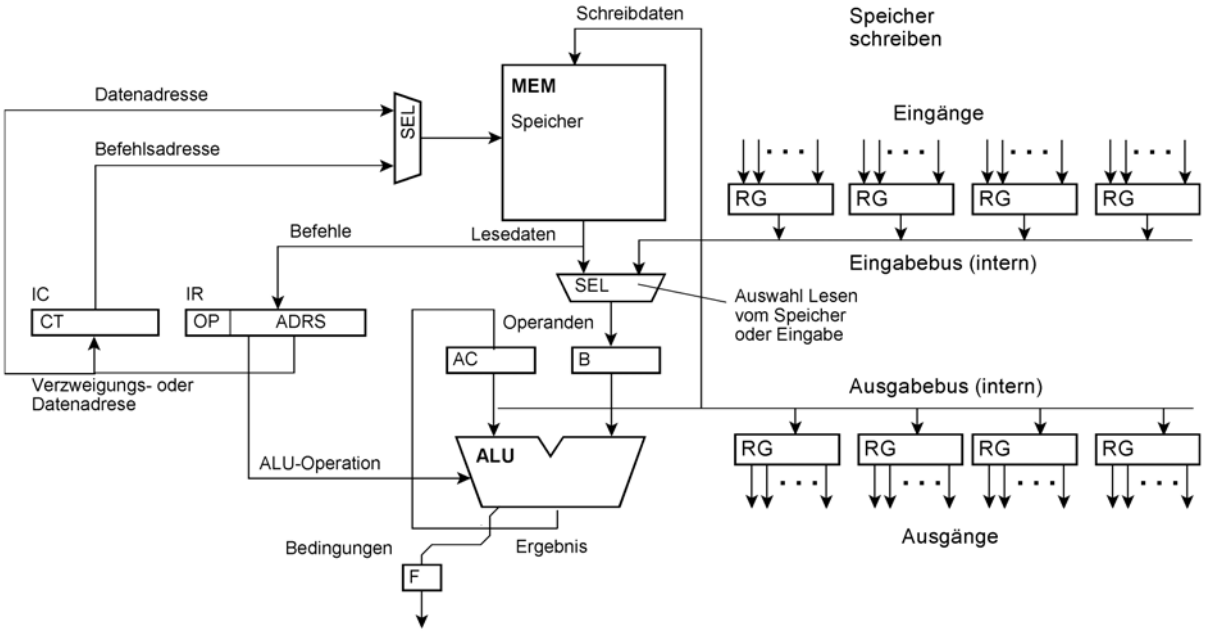


Abb. 4 Mikrocontroller – hier eine Akkumulatormaschine – mit Ein- und Ausgaberegistern.

Universelle E-A-Ports
Ein universeller E-A-Anschluss muß wahlweise als Eingang oder Ausgang wirken können. Um einen Signalweg so anzusteuern, daß er in beiden Richtungen (bidirektional) betrieben werden kann, stehen zwei Schaltungsauslegungen zur Wahl: die Transistorstufe mit Arbeitswiderstand (Open Collector, Open Drain, Open Source) und die Tri-State-Stufe.

Tri-State-Ausgänge
Die meisten Mikrocontroller haben E-A-Anschlüsse mit Tri-State-Stufen. Eine Tri-State-Stufe kann beide Signalpegel aktiv treiben. Sie hat einen Erlaubniseingang, über den die Betriebsart gesteuert wird. Ist das Erlaubnissignal inaktiv, so wird der Signalanschluss nicht getrieben. Er kann somit als Eingang genutzt werden. Der elementare Tri-State-Port benötigt zwei Register (und somit zwei Adressen): ein Richtungssteuerregister und ein Datenregister (Abb. 5).

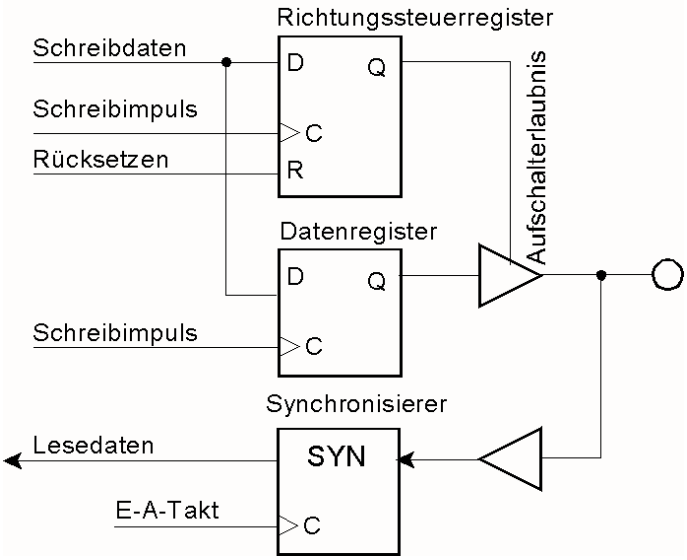


Abb. 5 E-A-Port mit Tri-State-Ausgang (eine Bitposition).

Elementare Anwendungsprogrammierung:

- Ausgabe: die betreffenden Bitpositionen des Richtungssteuerregister mit Einsen laden[1]. Der zugehörige Inhalt des Datenregisters erscheint an den Anschlüssen.
- Eingabe: die betreffenden Bitpositionen des Richtungssteuerregisters mit Nullen laden. Die Treiberstufen werden hochohmig. Somit dürfen die Anschlüsse von außen getrieben werden.
- Lesen: Es werden stets der Signalpegel an den Anschlüssen gelesen. Je nach Inhalt des Richtungssteuerregisters handelt es sich um den Inhalt des Datenregisters (Ausgabe) oder um ein außen anliegendes Signal (Eingabe).
- Der Anfangszustand (nach dem Rücksetzen): alles auf Eingabe (Anschlüsse hochohmig).
- Umschalten in die Ausgaberichtung: erst die Daten, dann die Richtungssteuerung. Sonst stellen sich an den Anschlüssen kurzzeitig die anfänglichen Bitmuster des Datenregisters ein (Abb. 6). Das kann in der Anwendungsschaltung sehr hässliche Nebenwirkungen haben.

1: Es gibt auch E-A-Ports, deren Richtungssteuerregister anders herum wirkt (Null = Ausgabe; Eins = Eingabe).

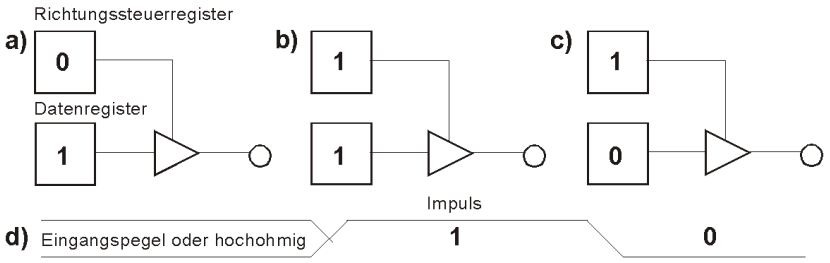
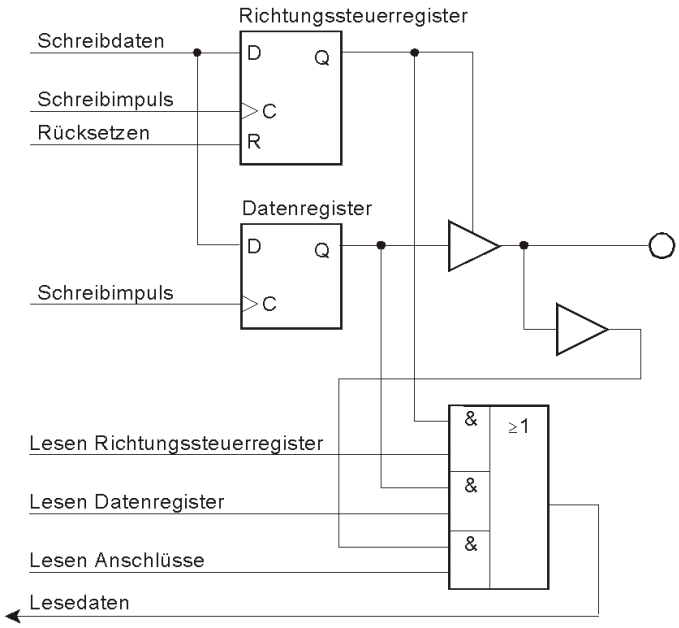


Abb. 6 Eine typische Fehlerquelle. a) am Anfang steht die Bitposition auf Eingabe. Der Inhalt des Datenregisters ist an sich gleichgültig. b) zuerst wird die Richtung auf Ausgabe gestellt. c) dann wird das Datenregister geladen. d) der Signalverlauf am Anschluss. Hat zuvor der jeweils andere Wert im Datenregister gestanden, erscheint ein Impuls. Es hängt von der Anwendungsschaltung ab, ob er schadet oder nicht.

Die universellen E-A-Ports der Atmel-AVR-Mikrocontroller

Sie sind viel einfacher als die E-A-Ports höherentwickelter Typen (wie beispielsweise Xmega oder ARM), weisen aber nicht die Einschränkungen anderer einfacher E-A-Ports auf. Die wesentliche Besonderheit besteht darin, daß jeder Port nicht nur zwei, sondern drei E-A-Adressen hat. Diese E-A-Adressen werden wie folgt verwendet:

- Richtungssteuerregister (Atmel-Name: DDR): zum Einstellen der Signalflussrichtung.
- Datenregister (Atmel-Name: PORT): zur Ausgabe.
- Anschlüsse (Atmel-Name: PIN) : zur Eingabe. Die Bits gelangen direkt in den Prozessor; der Inhalt des Datenregisters wird nicht verändert.



Boolesche Operationen über Binärvektoren

Es kommt immer wieder vor, daß einzelne Bits zu setzen, zu löschen oder abzufragen sind. Die Programmiersprache C hat aber keine entsprechenden Ausdrucksmittel. Deshalb müssen solche Operationen auf ganz elementare Weise mit Booleschen Verknüpfungen ausprogrammiert werden. C unterstützt Boolesche ("logische") Verknüpfungen über Variable. Es sind elementare Verknüpfungen der Aussagenlogik (UND, ODER usw.), die in allen Bitpositionen der Variablen unabhängig voneiander vorgenommen werden (beide Bits 0 miteinander, beide Bits 1 miteinander usw.).

Invertierung (Einerkomplement)

Eine Variable wird bitweise invertiert.

Verknüpfungen

Zwei Variable werden bitweise verknüpft. Üblich sind die elementaren Verknüpfungen UND, ODER und Antivalenz (XOR). Sie sind vielseitig nutzbar (Tabelle 1, Abb. 7).

Bits setzen

Es wird eine ODER-Verknüpfung ausgeführt, wobei der zweite Operand an allen zu setzenden Bitpositionen Einsen enthält und sonst Nullen.

Bits löschen

Es wird eine UND-Verknüpfung ausgeführt, wobei der zweite Operand an allen zu löschenden Bitpositionen Nullen enthält und sonst Einsen.

Bits ändern

Um Bits in ihrem Wert zu ändern (von 0 nach 1 und umgekehrt), wird eine Antivalenzverknüpfung (XOR) ausgeführt, wobei der zweite Operand an allen zu ändernden Bitpositionen Einsen enthält und sonst Nullen.

Setzen	Löschen	Ändern
$a \vee 0 = a$ $a \vee 1 = 1$	$a \cdot 0 = 0$ $a \cdot 1 = a$	$a \oplus 0 = a$ $a \oplus 1 = \bar{a}$

Tabelle 1 Typische Anwendungen elementarer Boolescher Verknüpfungen.

Bits einfügen

Bits einfügen heißt, ein Bitmuster in ausgewählte Bitpositionen zu transportieren. Das erfordert zwei Verknüpfungen:

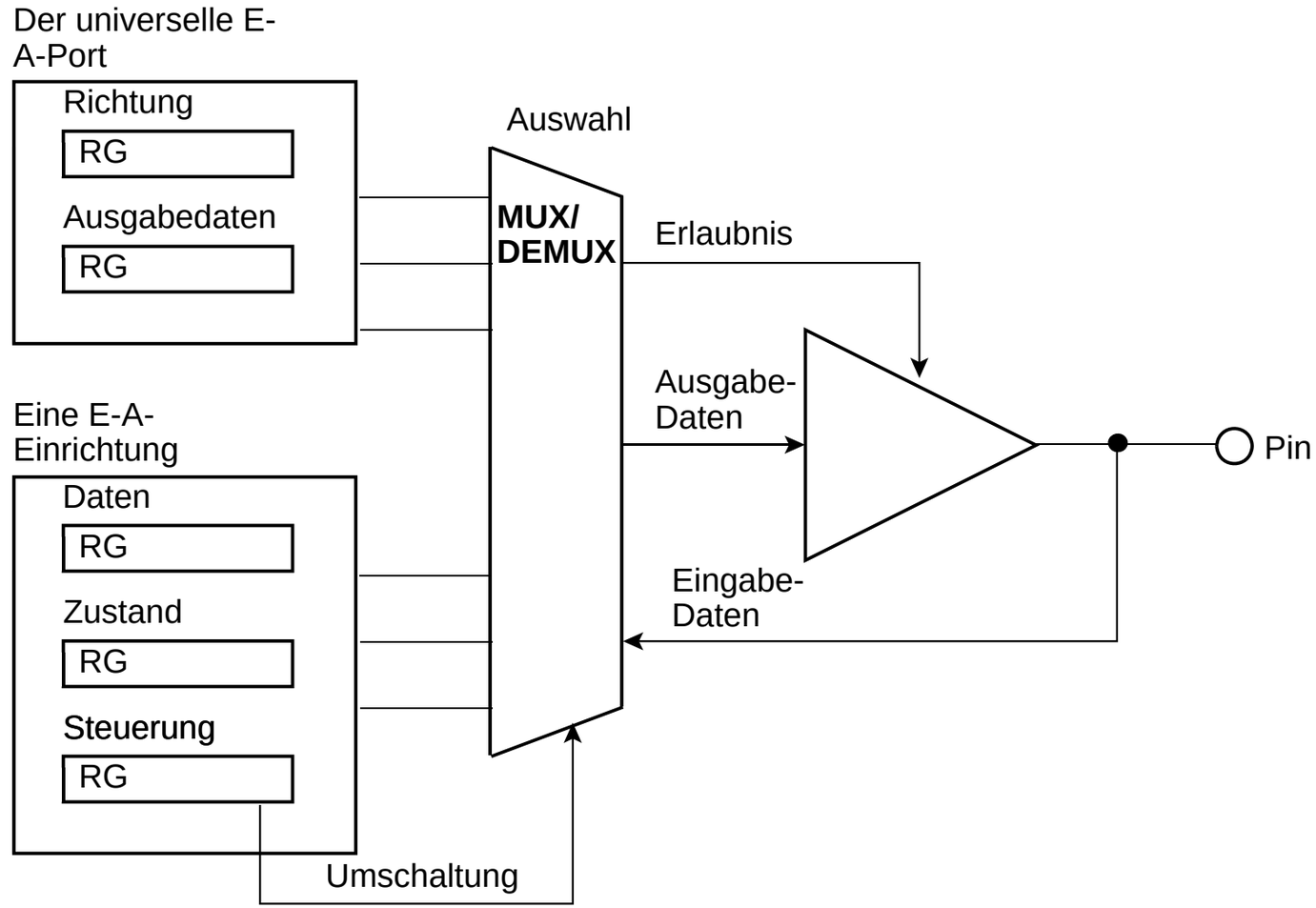
1. Löschen der Bitpositionen, in die das Bitmuster eingefügt werden soll, mittels einer UND-Verknüpfung deren zweiter Operand in allen zu löschenden Bitpositionen Nullen enthält und sonst Einsen.
2. Setzen gemäß dem einzufügenden Bitmuster mittels einer ODER-Verknüpfung, deren zweiter Operand an den betreffenden Bitpositionen die einzufügenden Werte enthält und sonst Nullen.

Abb. 7 Setzen, Löschen und Ändern anhand von Beispielen.

Der Anwendungsprogrammierer sieht nur die Register der Ein-und Ausgabe.

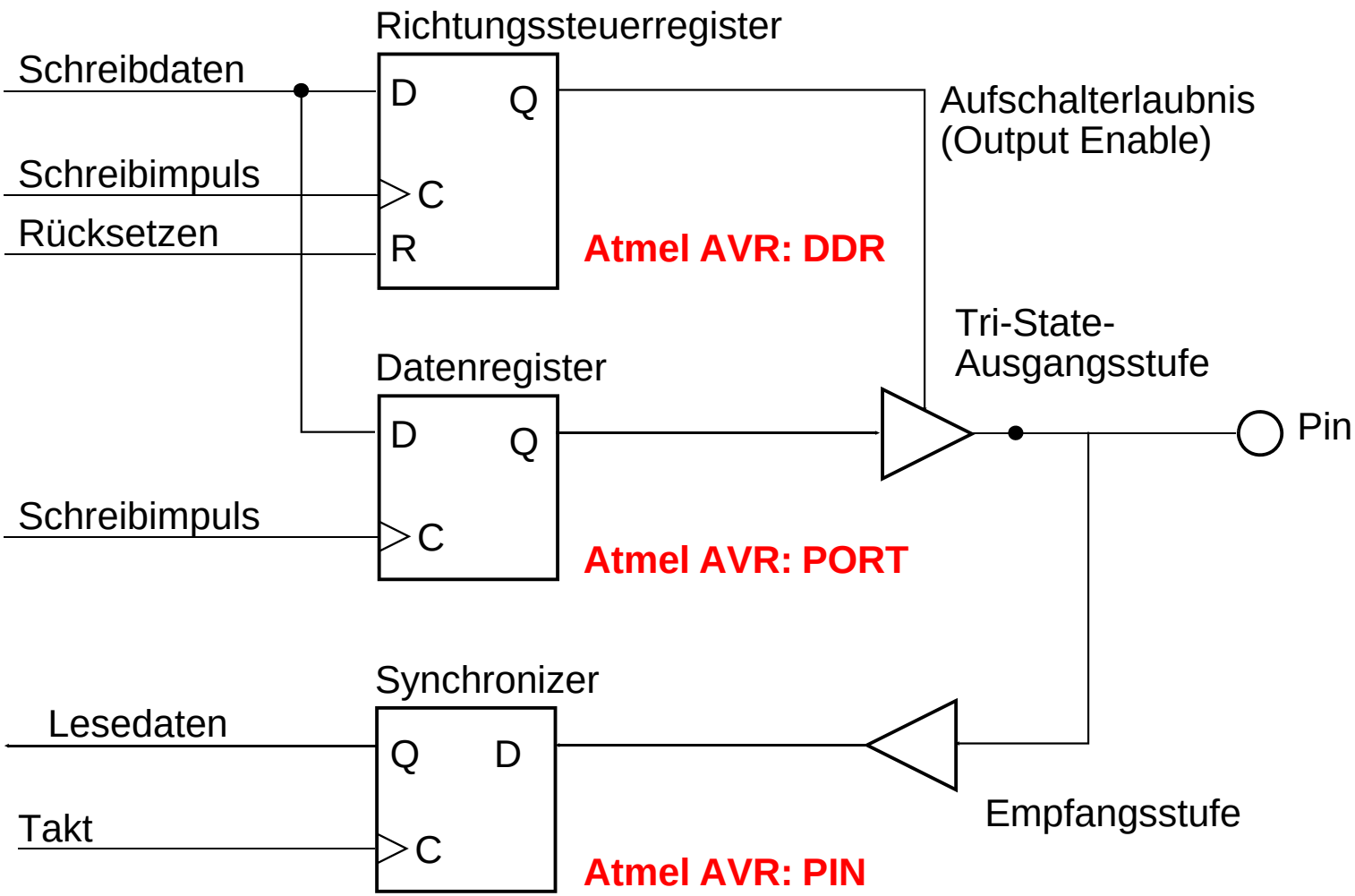
Um alles andere -- auch um den Kern des Prozessors -- muß er sich nicht kümmern.

Die Register werden über symbolische Namen angesprochen.
Der Compiler kennt die Namen.
Im Grunde sind es globale Variable.
Der Programmier kann ihnen Werte zuweisen und sie in Operationen einbeziehen.
Manchmal muß er aber aufpassen.
Wo genau, sagen der gesunde Menschenverstand* und das jeweilige Prozessordatenbuch...



*: So sollte es offensichtlich sein, daß man Eingabedaten keine Werte zuweisen kann...

Der universelle E-A-Port des Atmel AVR / ATmega:



Beim Atmel AVR werden die Ports alphabetisch durchnumeriert. Die Ports heißen A, B, C usw. Typische Bezeichner sind DDRA, PORTB, PINC usw.