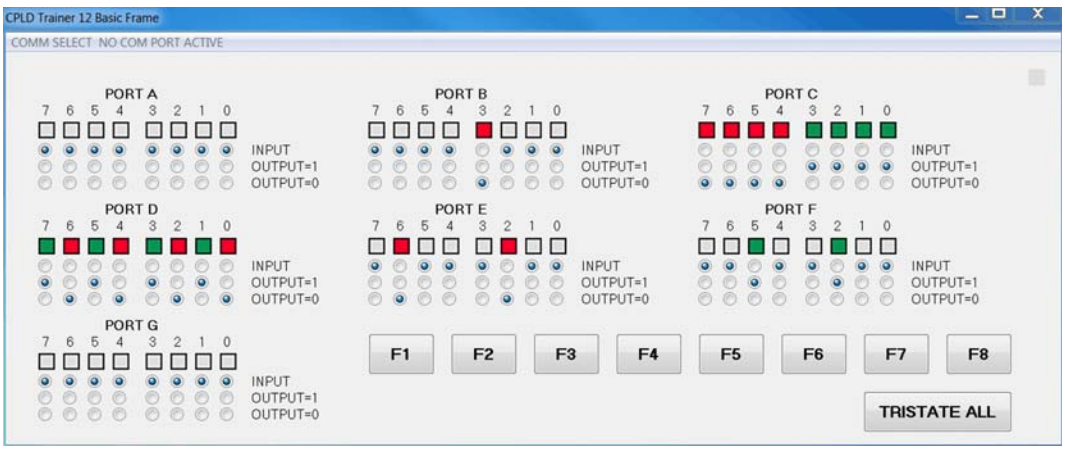
CPLD-Lehrgerät 12
Bediengrundprogramm
(CPLD LG12 Basic Frame)
Kurzbeschreibung

Stand: 26. 5. 2015

Zweck:
Elementare Bedienung des CPLD-Lehrgerätes 12. Softwareplattform zum Starten weiterer Anwendungsfunktionen.

Das CPLD-Lehrgerät 12 ist ein frei programmierbares Testhilfsgerät, das u. a. als quasistatischer Digitaltester oder als Peripherienachbildung eingesetzt werden kann. Es beruht auf zwei Mikrocontrollern Atmel ATmega, die ein elementares Mehrprozessorsystem bilden. Sie sind über eine serielle Schnittstelle untereinander verbunden. Der erste Mikrocontroller ist der Hauptprozessor (Master), der zweite der Hilfsprozessor (Slave). Die Kommunikation mit der Außenwelt läuft über eine weitere serielle Schnittstelle, die vom Hauptprozessor angesteuert wird. Das CPLD-Lehrgerät enthält keine eingebauten Bedien- und Anzeigeeinrichtungen. Im praktischen Einsatz ist es deshalb mit einem Bediengerät zu verbinden. Das typische Bediengerät ist ein Windows-PC mit einem Bedienprogramm. Zum Aufbau des CPLD-Lehrgerätes 12 s. die Gerätekurzbeschreibung, zur seriellen Kommunikationsschniitstelle s. die Programmbeschreibung.



Das CPLD-Lehrgerät 12 unterstützt 7 frei programmierbare E-A-Ports mit jeweils 8 Bitpositionen. Die elementare Bedienung besteht in folgendem:

1. Zyklische Anzeige der aktuellen Signalpegel an den Anschlüssen (Pins) aller 7 • 8 = 56 Bitpositionen.

2. Umschalten aller 56 Bitpositionen (jede einzeln) auf:
   - Eingang,
   - Ausgabe einer Eins (High-Pegel),
   - Ausgabe einer Null (Low-Pegel).

3. Schalten aller 56 Bitpositionen auf Eingang bzw. hochohmigen Zustand (Tristate-Funktion) mit aktiven Pullup-Widerständen.

4. Aufruf weiterer Anwendungsfunktionen.

Das Programm emuliert im Grunde eine Bedientafel mit 56 Zweifarben-LEDs und 56 Dreistellungs-Kippschaltern.

Programmiersprache: PowerBasic.

Aufbau:
Hauptprogramm und Kommunikations-Thread.

Das Hauptprogramm ist eine typische Windows-Anwendung, die auf einer Callback-Funktion zur Ereignisbehandlung beruht. Im Kommunikations-Thread läuft eine Endlosschleife. Zum Kommunikationsprotokoll siehe die Programmbeschreibung des CPLD-Lehrgeräts 12.

Der Dialogaufbau
Die Besonderheit besteht darin, daß die Steuerelemente der insgesamt 56 Bitpositionen der E-A-Ports nicht einzeln aufgebaut, sondern mit Programmschleifen erzeugt werden.

Die Steuerelemente der Bitpositionen
Deren Identifier werden systematisch erzeugt, ausgehend von einem Anfangswert %Portbase. Jedes Steuerelement hat einen Offsetwert von 12 Bits Länge. Die Bits 11...8 codieren den Port, die Bits 7...4 codieren das jeweilige Steuerelement (Anzeige, Taste) und die Bits 3...0 codieren die Bitposition.

11 8	7 4	3 0
Port	Nummer des Steuerelements in der Bitposition	Bitposition (7...0) oder Beschriftung
0 = Port A 1 = Port B 2 = Port C 3 = Port D 4 = Port E 5 = Port F 6 = Port D	0 = Bitnummer (Beschriftung) 1 = Anzeigesymbol (“LED”) 2 = Eingang 3 = Ausgang auf 1 (High) 4 = Ausgang auf 0 (Low)	7...0 = Bitpositionen 8 = Überschrift “PORT” 9 = “INPUT” 10 = “OUTPUT=1" 11 = “OUTPUT=0"

Die Callback-Funktion
Die Ereignisse der Auswahltasten werden nicht einzeln erkannt, sondern durch Auswertung des Bereichs der zugehörigen Identifier (liegt der übergebene Nachrichtenparameter innherhalb dieses Bereichs oder nicht?). Demgemäß wird eine pauschale Behandlung ausgelöst, wobei Port, Bit und auszuführend Aktion aus dem jeweils übergebenen Nachrichtenparameter (= Identifier) errechnet werden.

Der Offline-Betrieb
Ist keine serielle Kommunikationsschnittstelle aktiv, arbeitet das Programmm off line. Die Auswahltasten wirken direkt auf die Anzeigen:

- Bitposition = Eingang: Anzeige grau.
- Bitposition = Ausgang 1: Anzeige grün.
- Bitposition = Ausgang 0: Anzeige rot.

Die Tristate-Taste schaltet alle Bitpositionen auf Eingang.

Der Online-Betrieb
Es läuft eine zyklische Anzeigeschleife, die alle Pins im CPLD-Lehrgerät abfragt (Logout). Das CPLD-Lehrgerät 12 kann nur Einsen oder Nullen erkennen, nicht aber den verbotenen Zustand. Deshalb ergeben sich nur grüne oder rote Anzeigen.

Der Kommunikations-Thread
Er wird dann aktiv, wenn eine Verbindung über eine serielle Kommunikationsschnittstelle eingerichtet werden konnte. Das Programm im Kommunikations-Thread ist stets eine Endlosschleife, entweder die Hauptsteuerschleife oder die Anzeigeschleife. In beiden Schleifen können sog. Thread-Kommandos ausgeführt werden, die Kommunikationsvorgänge auslösen.

Die Hauptsteuerschleife
Der Thread befindet sich die meiste Zeit in einem Ruhezustand (Warteschleife). Eine Kommandoübergabe führt dazu, daß das Kommando ausgeführt wird. Danach kehrt der Thread in den Ruhezustand zurück.

Thread-Kommandos in der Hauptsteuerschleife (einmalige Ausführung mit Rückkehr in Ruhezustand):

Code	Funktion
0	Ruhezustand; nichts tun.
1	Die Thread-Funktion beenden. Zuvor das Kommando beantworten.
2	Übergang in den Ruhezustand. Zuvor das Kommando beantworten.
3	Eine Zeichenkette senden. Keine Antwort erwarten.
4	Eine Zeichenkette senden und ein einzelnes Antwortzeichen erwarten (typischerweise ACK).
5	Eine Zeichenkette senden und eine Antwort fester Länge erwarten. Die Länge muß in **rcvlen1** übergeben werden.
6	Eine Zeichenkette senden und eine Antwort beliebigerLänge erwarten. Es kann auch gar keine Antwort kommen. Entscheidung mittels Zeitkontrolle.
7	In die Endlosschleife übergehen, aber mit einem anfänglichen Logout beginnen (am Anfang alle Bitpositionen darstellen)
8	In die Endlosschleife übergehen. Zyklischer Logout (die Anzeige wird nur für jene Bitpositionen aktualisiert, die sich geändert haben)

Die Anzeigeschleife

In dieser Schleife werden zyklisch Logout-Kommandos zum CPLD-Lehrgerät gesendet, die Anzeigedaten empfangen und die Anzeigen aktualisiert. Anfänglich werden alle Anzeigen aktualisiert, in weiteren Umläufen hingegen nur jene, deren Signalpegel sich gegenüber dem letzten Umlauf geändert haben (Beschleunigung). Das Logout-Kommando liefert aber stets alle 56 Signalpegel.

In den Umlauf den Anzeigeschleife können Thread-Kommandos eingeschleust werden. Sie werden einmalig ausgeführt. Danach wird die Anzeigeschleife fortgesetzt.

Thread-Kommandos in der Anzeigeschleife (einmalige Ausführung mit Rückkehr zur zyklischen Anzeige):

Code	**Funktion**
0	Ruhezustand; nichts tun.
1	Die Anzeigeschleife beenden. Übergang zur Hauptsteuerschleife. Zuvor das Kommando beantworten.
2	Reserviert.
3	Eine Zeichenkette senden. Keine Antwort erwarten.
4	Eine Zeichenkette senden und ein einzelnes Antwortzeichen erwarten (typischerweise ACK).
5	Eine Zeichenkette senden und eine Antwort fester Länge erwarten. Die Länge muß in **rcvlen1** übergeben werden.
6	Eine Zeichenkette senden und eine Antwort beliebigerLänge erwarten. Es kann auch gar keine Antwort kommen. Entscheidung mittels Zeitkontrolle.
7	Zwei Zeichenketten senden und jeweils ein einzelnes Antwortzeichen erwarten (typischerweise ACK).

Globale Variable der Thread-Kommunikation:

Variable	**Bedeutung / Inhalt**	**Wer schaltet ein bzw. sendet?**	**Wer schaltet aus?**
threadcommand	Kommandocode	Das Hauptprogramm	Das Hauptprogramm
threadxmit	Zu sendende Zeichenkette	Das Hauptprogramm	Das Hauptprogramm
auxxmit	Ergänzend zu sendende Zeichenkette (bedarfsweise)	Das Hauptprogramm	Das Hauptprogramm
threadbusy	Besetztzustand. Der Thread hat das Kommando angenommen und führt es aus	Der Thread	Das Hauptprogramm
threadreply	Fertigmeldung. Der Thread hat das Kommando ausgeführt	Der Thread	Das Hauptprogramm
threadloop	Der Thread läuft in der Anzeigeschleife	Der Thread	Der Thread
threadbreak	Ein Kommando in die Anzeigeschleife einschleusen	Das Hauptprogramm	Der Thread
threadmessage	Nachrichtencode	Der Thread	Der Thread
threadcheck	Fehleranzeige	Der Thread	Der Thread
threadcontrol	Der interne Kommandocode	Der Thread	Der Thread
xmit	Die aktuell zu sendende Zeichenkette	Der Thread	Der Thread

System, Hauptprogramm und Thread
Das Hauptprogramm ist im wesentlichen eine Callback-Funktion. Es bekommt vom System nur dann Laufzeit, wenn ein Ereignis (Nachricht, Message) zu bearbeiten ist.

Der Thread ist eine Endlosschleife. Sie bekommt zyklisch Laufzeit zugeteilt und entzogen.

Aus Sicht des Systems hat das Hauptprogramm einen Anfang (Callback) und ein Ende (Rückkehr zum System).

Sowohl im Hauptprogramm als auch im Thread können Steuerelemente angesprochen werden. Der Start des Hauptprogramms (Callback) bezieht sich stets auf ein Steuerelement, dem das System eine Nachricht (Message) sendet. Wenn der Thread ebenfalls ein Steuerelement anspricht, laufen im System Reservierungs- und Freigabevorgänge ab.

Was nicht vorkommen darf:
Daß das Hauptprogramm auf Signale des Threads wartet, wenn dieser ein Steuerelement angesprochen hat.

Denn dies würden einen Deadlock ergeben:
Das Hauptprogramm wartet auf den Thread, der Thread wartet darauf, daß er Zugang zu den Steuerelementen erhält. Das Hauptprogramm aber hängt in der Warteschleife und gibt somit die Steuerelemente nicht frei.

Deshalb:
Zu den Zeiten, in denen der Thread mit Steuerelementen arbeitet, dürfen keine Handshaking-Signalspiele mit dem Hauptprogramm ablaufen. (Es sei denn, man verwendet einschlägige Funktionen der Windows-API, wie Mutexes oder Critical Sections.)

Lösung:
Wenn der Thread in einer Endlosschleife läuft, die auf Steuerelemente wirkt, werden Kommandos über eine einfache Signalisierung ohne Handshaking in diese Schleife eingeschleust (Variable threadbreak).

Kommandoausführung in der Hauptsteuerschleife
Befindet sich der Thread im Grundzustand, wird jeder Wert threadcommand ungleich 0 als neuer Kommandocde in die interne Variable threadcontrol übernommen. Ebenso wird der Inhalt der Zeichenkette threadxmit in die interne Sendezeichenkette xmit übertragen.

Zu Beginn der Kommandausführung setzt der Thread die Besetztanzeige threadbusy. Nach Ausführung des Kommandos setzt der Thread die Fertigmeldung threadreply. Zudem setzt er threadcontrol auf 0 (Rückkehr in den Ruhezustand). Der Thread wartet jetzt darauf, daß das Hauptprogramm threadbusy und threadreply ausschaltet.

Um eine Kommandoausführung zu starten, muß das Hauptprogramm zunächst die Zeichenkette threadxmit mit den Sendedaten füllen und dann threadcommand auf den jeweiligen Kommandocode setzen. Anschließend ist auf threadbusy und nachfolgend auf threadreply zu warten. Jetzt ist es möglich, ein neues Kommando zu senden. Soll kein neues Kommando übertragen werden, ist threadcommand auf 0 zu setzen. Dann erst dürfen threadbusy und threadreply ausgeschaltet werden. (Denn sonst würde der Thread den aktuellen Inhalt von threadcommand als neues Kommando interpretieren.)

Kommandoausführung in der Anzeigeschleife
In der Anzeigeschleife dürfen Thread und Hauptprogramm nicht wechselseitig aufeinander warten, denn dabei könnte es zum Deadlock kommen. Das Hauptprogramm muß lediglich threadxmit mit den Sendedaten füllen, dann threadcommand auf den jeweiligen Kommandocde setzen und schließlich threadbreak einschalten.

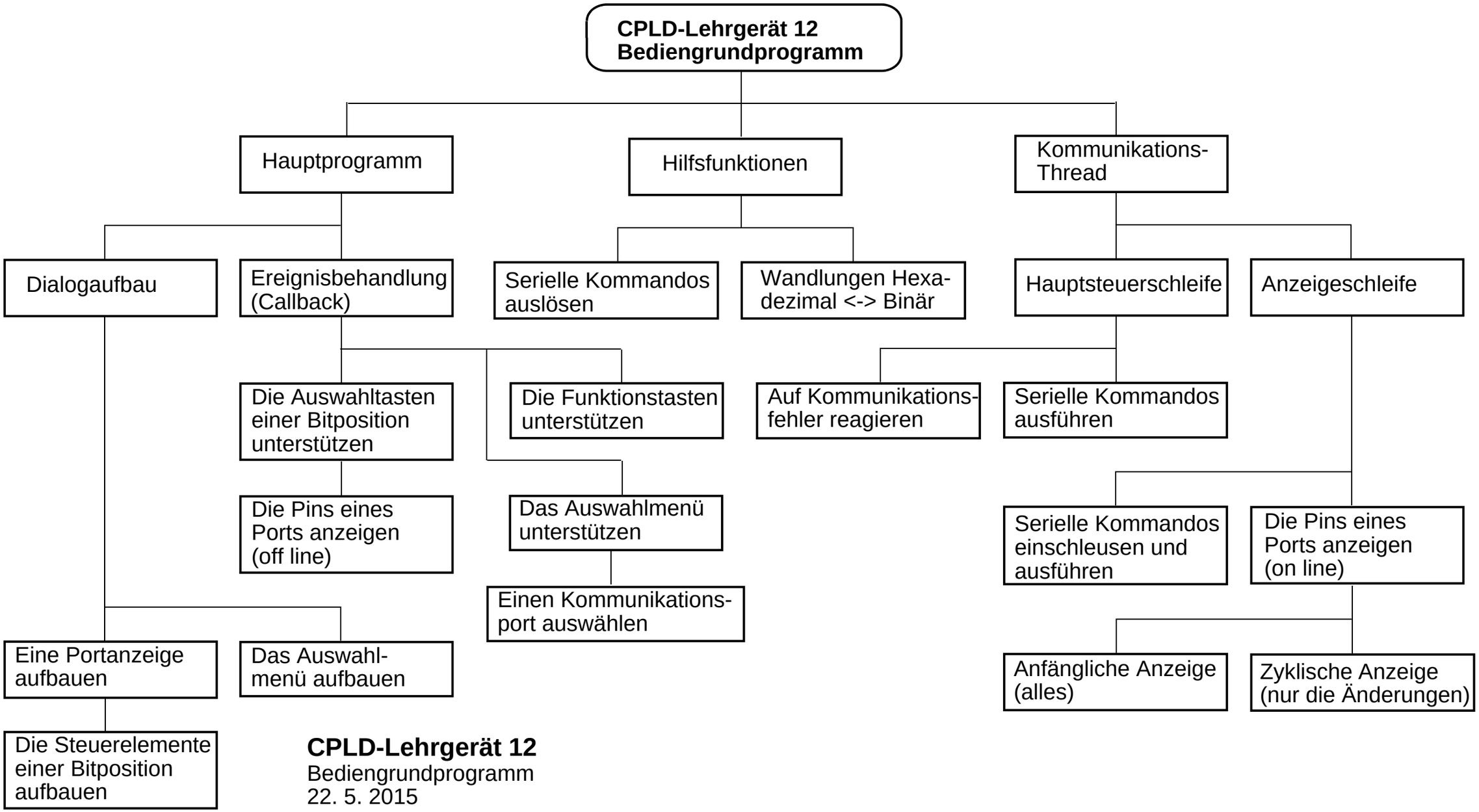
Wenn der Thread threadbreak als gesetzt erkennt, übernimmt er threadcommand als neuen Kommandocde in die interne Variable threadcontrol sowie den Inhalt der Zeichenkette threadxmit in die interne Sendezeichenkette xmit. Danach führt er das Kommando aus. Nach der Kommandoausführung schaltet er threadbreak aus und kehrt zur Anzeigeschleife zurück.

Ereignismeldungen (Nachrichten) vom Thread
Um Nachrichten vom Thread empfangen zu können, ist das Steuerelement %STATUS vorgesehen. Das Hauptprogramm wertet Nachrichten des Einschaltens (Enable) und des Ausschaltens (Disable) aus. Als Grundfunktion kann der Thread Einschaltnachrichten senden. Die Art der Nachricht muß in der Variablen threadmessage codiert werden. Als erste Nachricht wird eine Fehlermeldung %threaderror unterstützt.

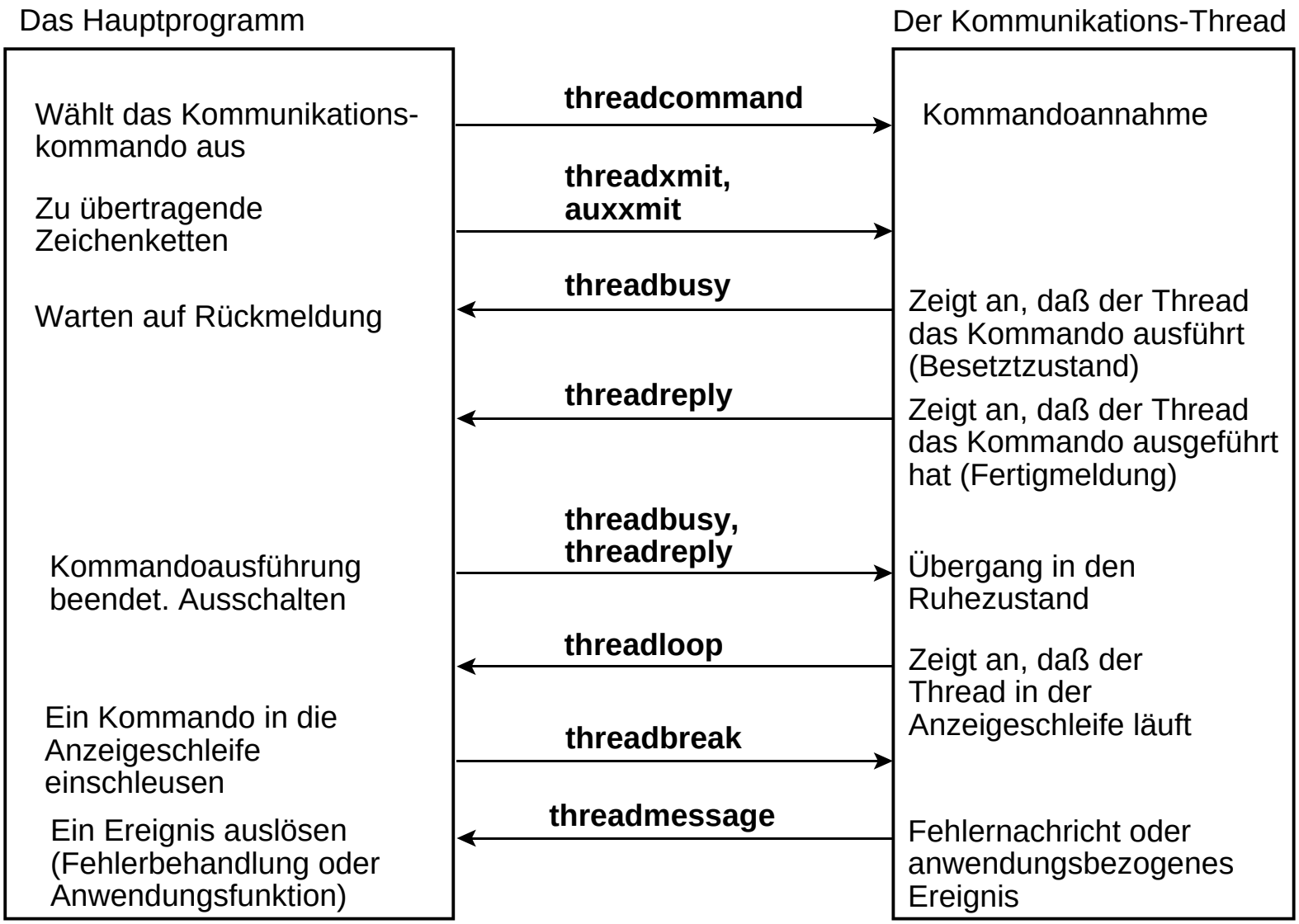
Anwendungsprogramme im Thread
Dem Thread können anwendungsspezifische Kommandos beliebiger Komplexität hinzugefügt werden. Sinngemäß kann die Callbackfunktion auf die Untersützung zusätzlicher Threadnachrichten erweitert werden.

Fehlerbehandlung
In der ersten Version des Bediengrundprogramms gibt es lediglich eine Zeitkontrolle beim Empfangen. Der Zeitkontrollfehler führt dazu, daß der Thread eine Fehleranzeige aufbaut und eine Nachricht an das Steuerelement %STATUS sendet. Dann beendet sich der Thread selbst. Das Programm gibt die Kommunikationsschnittstelle frei und geht in den Offline-Betrieb über.

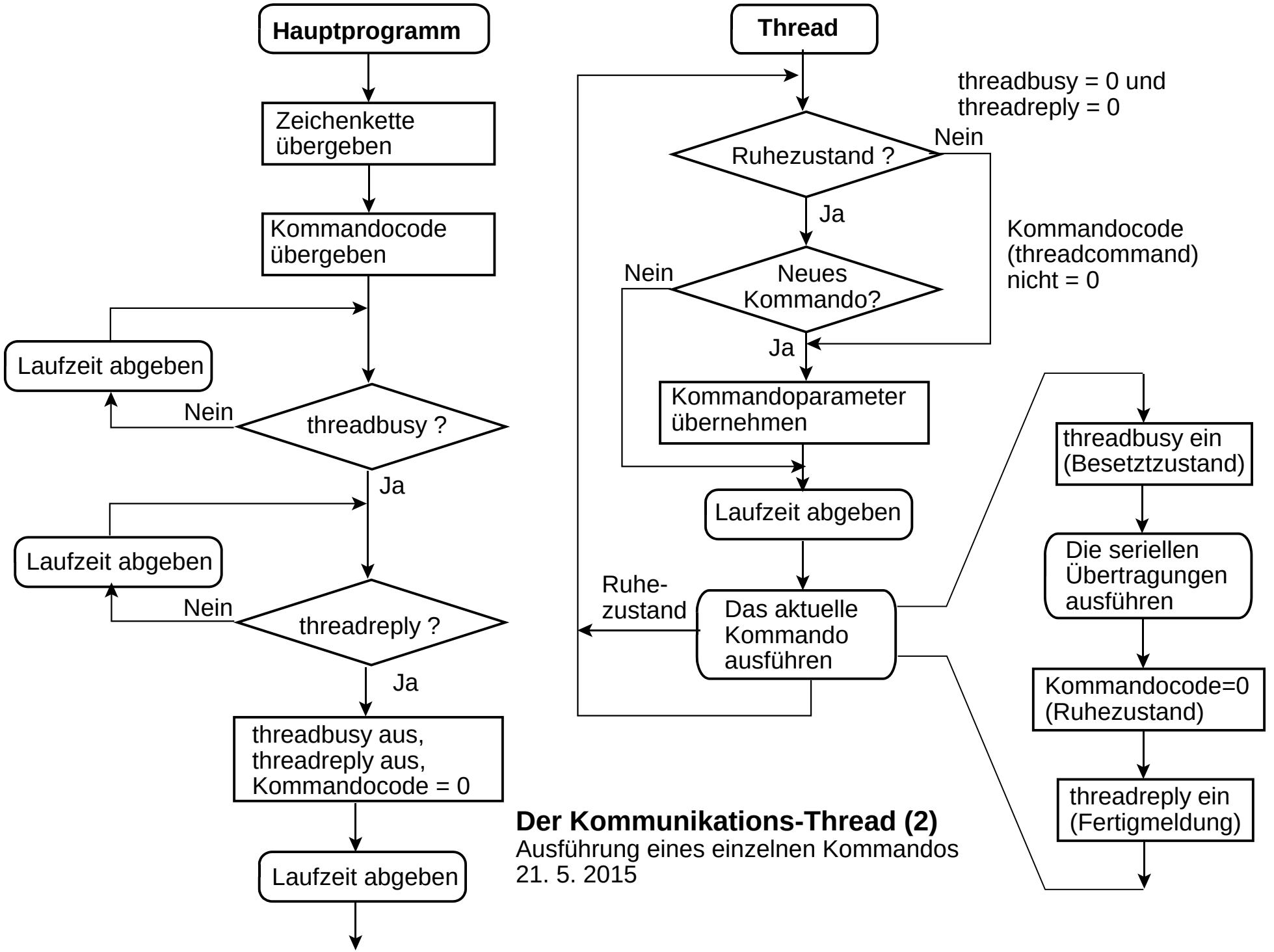
Die Zeitkontrolle beim Empfangen
Wenn eine Empfangsfunktion einen Zeitkontrollfehler erkennt, so setzt sie das Fehlersignal threadcheck, tut aber ansonsten so, als ob alles in Ordung wäre. Die Empfangszeichenkette wird mit hexadezimalen Nullen gefüllt. So kann ein laufendes Kommand noch beendet werden. Der Fehler wird dann beim nächsten Schleifenumlauf ausgewertet.



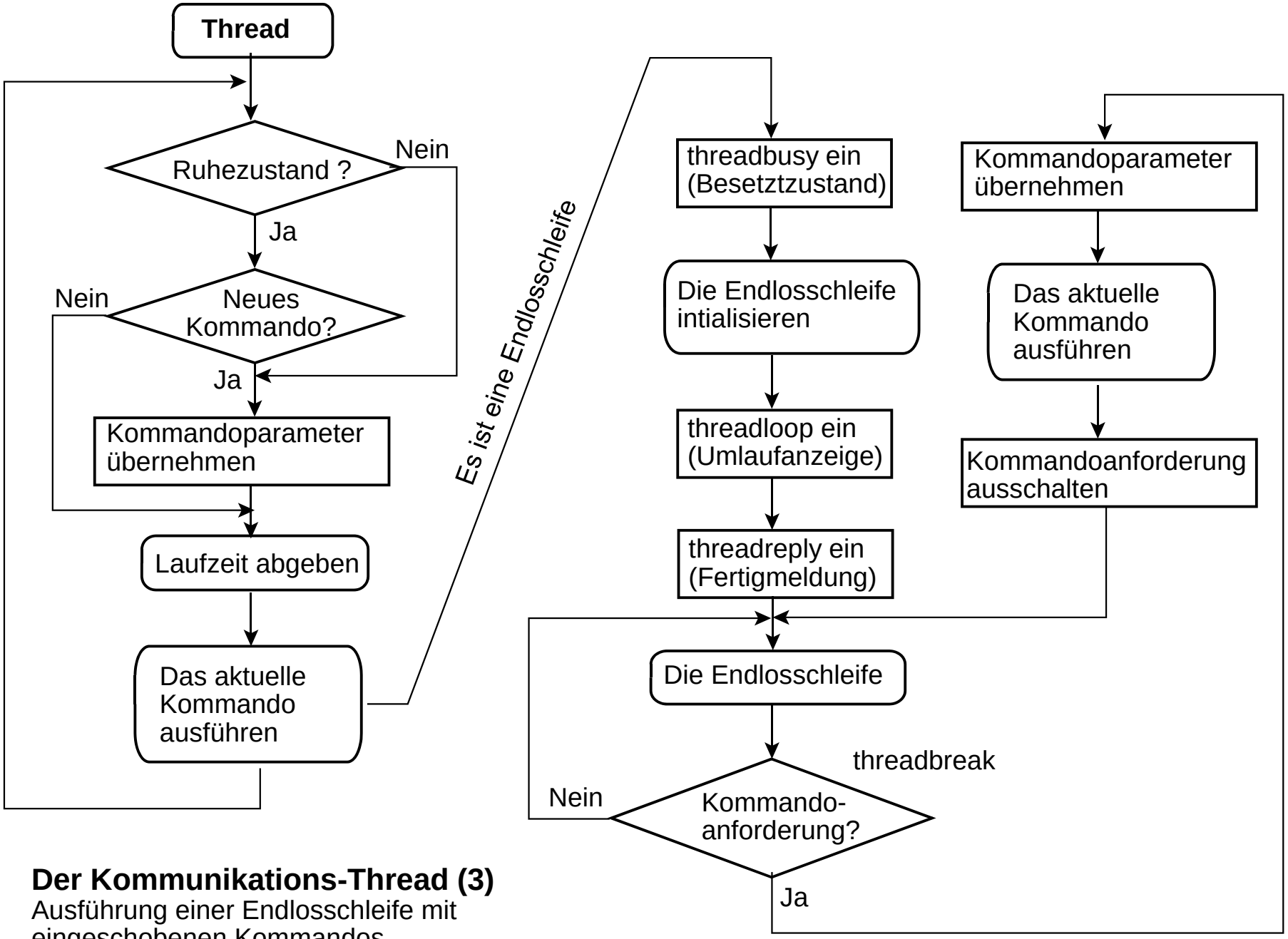
CPLD-Lehrgerät 12
Bediengrundprogramm
22. 5. 2015



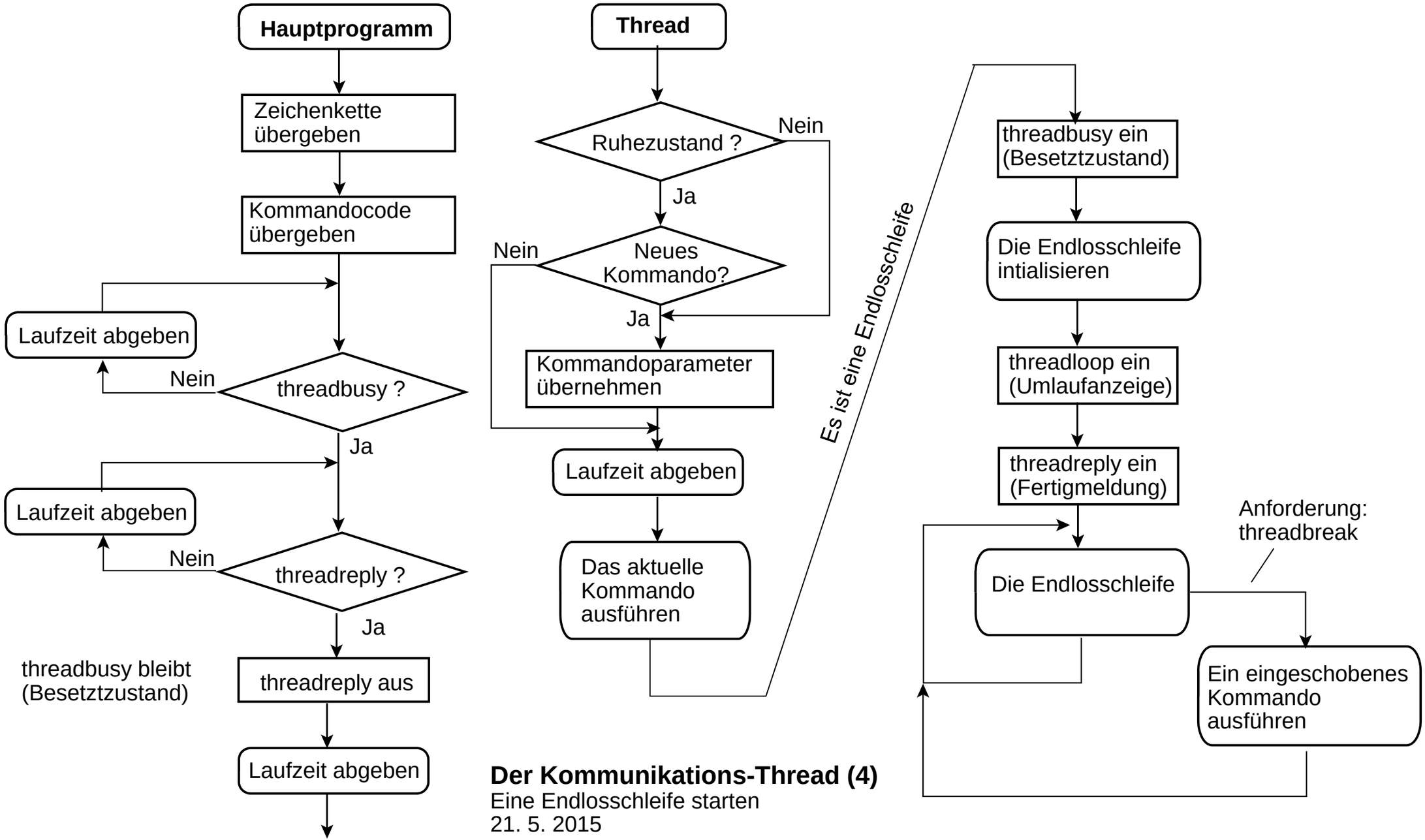
Der Kommunikations-Thread (1)
Signale der Programmschnittstelle
21. 5. 2015



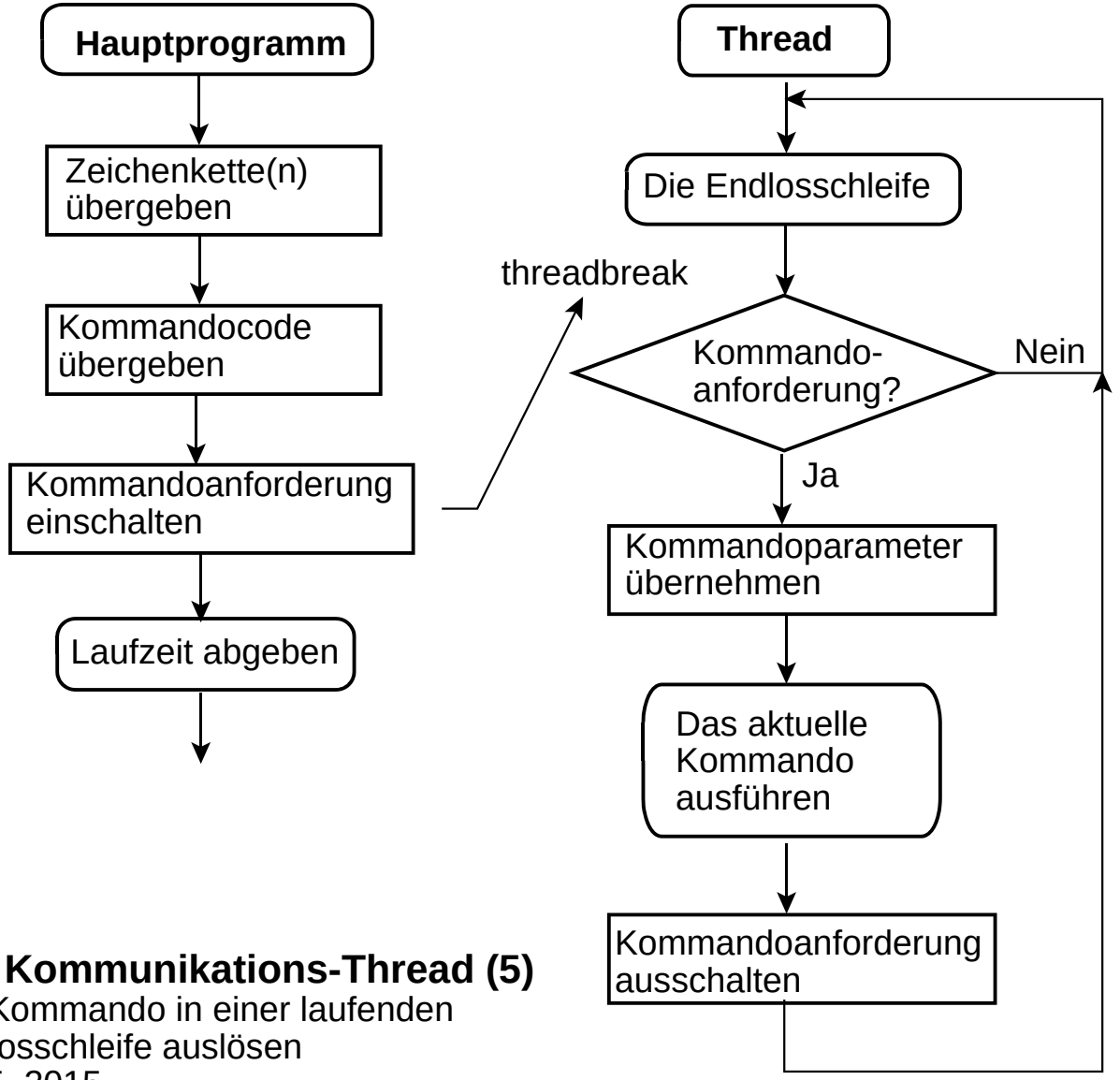
Der Kommunikations-Thread (2)
Ausführung eines einzelnen Kommandos
21. 5. 2015



Der Kommunikations-Thread (3)
Ausführung einer Endlosschleife mit eingeschobenen Kommandos
21. 5. 2015



Der Kommunikations-Thread (4)
Eine Endlosschleife starten
21. 5. 2015

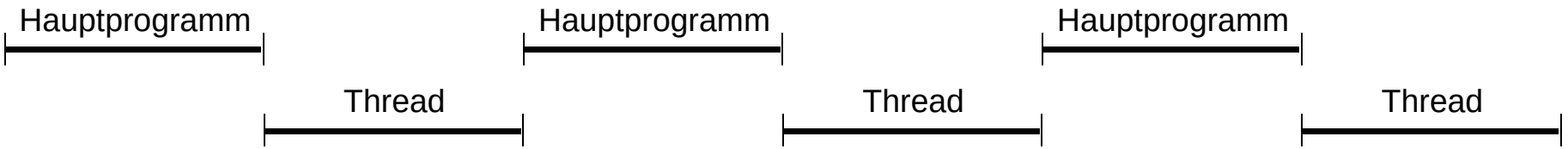


Der Kommunikations-Thread (5)

Ein Kommando in einer laufenden Endlosschleife auslösen

21. 5. 2015

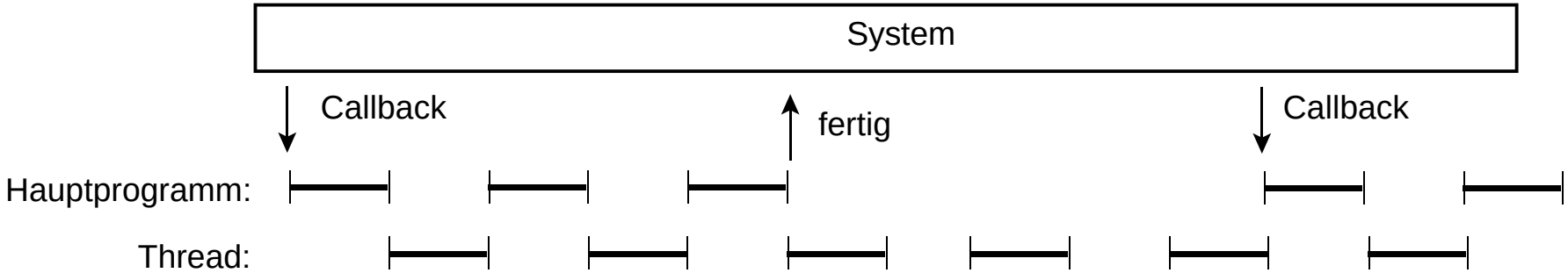
Die grundsätzliche Laufzeitverteilung -- ganz allgemein:



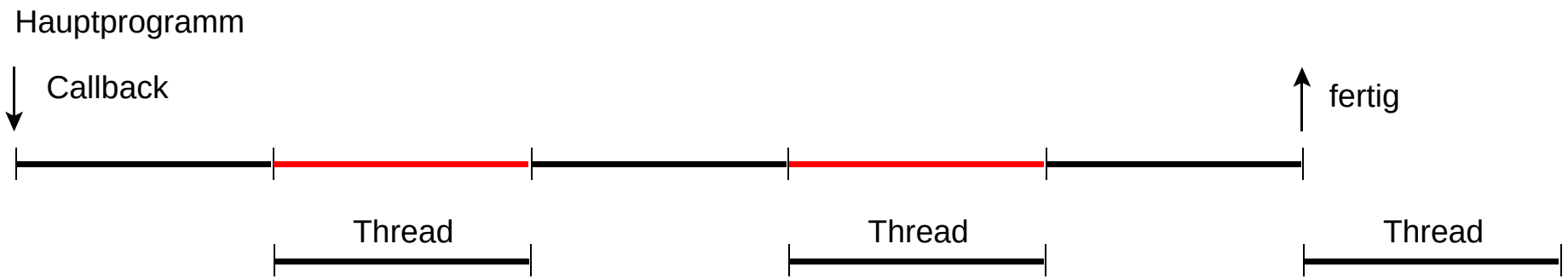
Die grundsätzliche Laufzeitverteilung aus Sicht des Systems:

Das Hauptprogramm ist eine Callback-Funktion. Es bekommt vom System nur dann Laufzeit, wenn ein Ereignis (Nachricht, Message) zu bearbeiten ist.

Der Thread ist eine Endlosschleife. Sie bekommt zyklisch Laufzeit zugeteilt und entzogen.



Der Kommunikations-Thread (6)
Grundlagen der Laufzeitzuteilung
21. 5. 2015



Aus Sicht des Systems hat das Hauptprogramm einen Anfang (Callback) und ein Ende (Rückkehr zum System).

Sowohl im Hauptprogramm als auch im Thread können Steuerelemente angesprochen werden. Der Start des Hauptprogramms (Callback) bezieht sich stets auf ein Steuerelement, dem das System eine Nachricht (Message) sendet. Wenn der Thread ebenfalls ein Steuerelement anspricht, laufen im System Reservierungs- und Freigabevorgänge ab.

Was nicht vorkommen darf:
Daß das Hauptprogramm auf Signale des Threads wartet, wenn dieser ein Steuerelement angesprochen hat. Das Hauptprogramm wartet dann auf den Thread, der Thread wartet darauf, daß er Zugang zu den Steuerelementen erhält. Das Hauptprogramm aber hängt in der Warteschleife und gibt somit die Steuerelemente nicht frei: DEADLOCK.

Zu den Zeiten, in denen der Thread mit Steuerelementen arbeitet, dürfen keine Handshaking-Signalspiele mit d m Hauptprogramm ablaufen. (Es sei denn, man verwendet einschlägige Funktionen der Windows-API, wie Mutexes oder Critical Sections.)

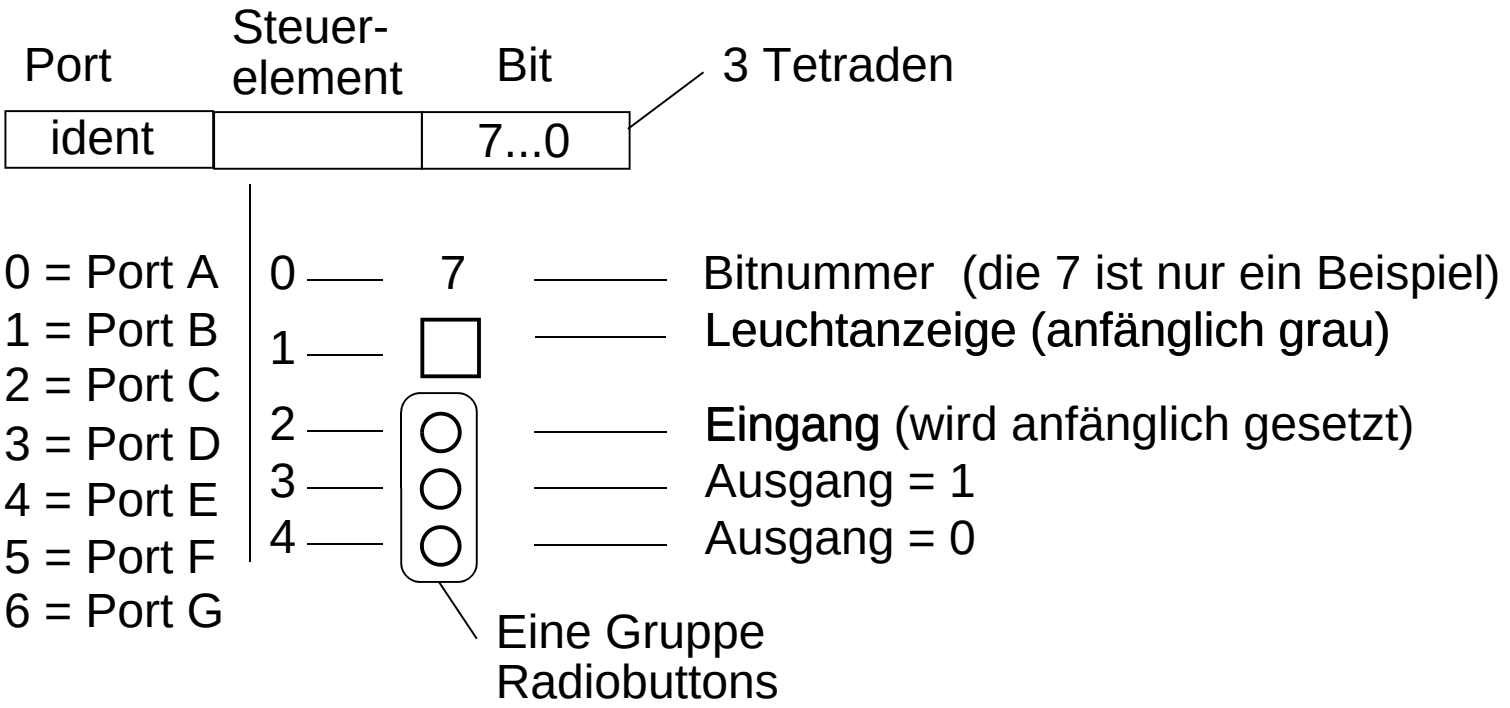
Deshalb: Wenn der Thread in einer Endlosschleife läuft, werden Kommandos über eine einfache Signalisierung ohne Handshaking in diese Schleife eingeschleust (Signalvariable threadbreak).

Der Kommunikations-Thread (7)
Hauptprogramm und Thread
21. 5. 2015

Der Identifier:

-- Eine dreistellige Hexadezimalzahl --

Sie wird als Offset zu einem Anfangswert (%PORTBASE) addiert.

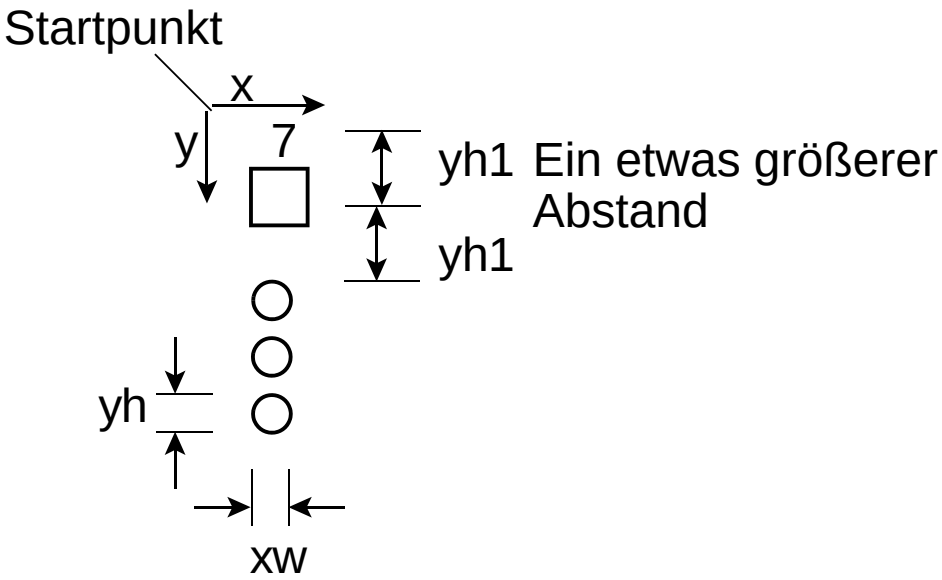


Elementare Portanzeige (1)

Eine Bitposition

22. 5. 2015

Eine Bitposition:

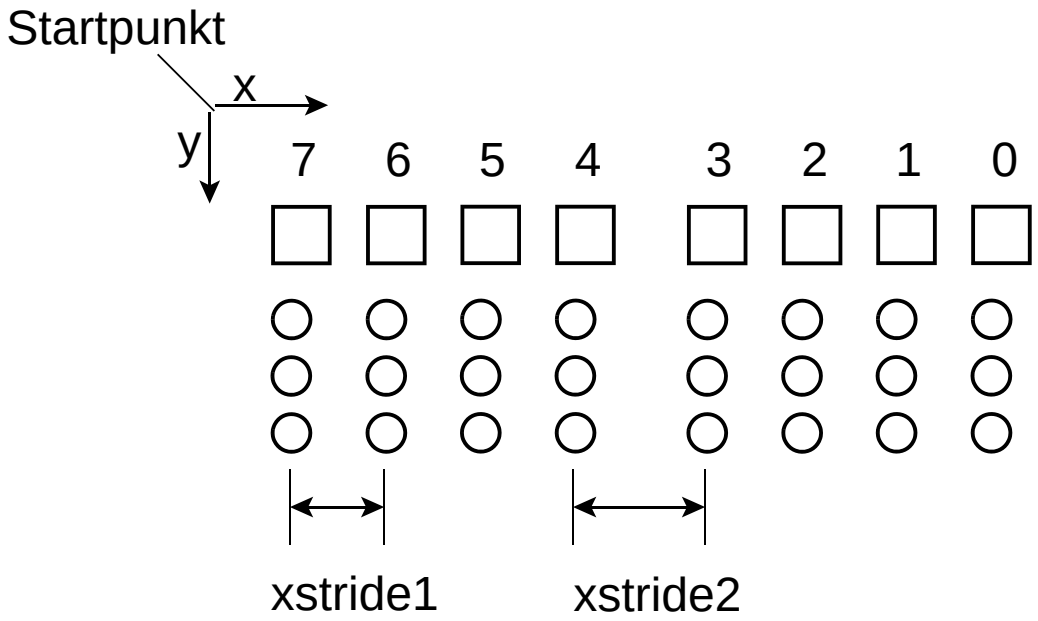


Elementare Portanzeige (2)

Die graphische Darstellung in einer Bitposition

8. 4. 13

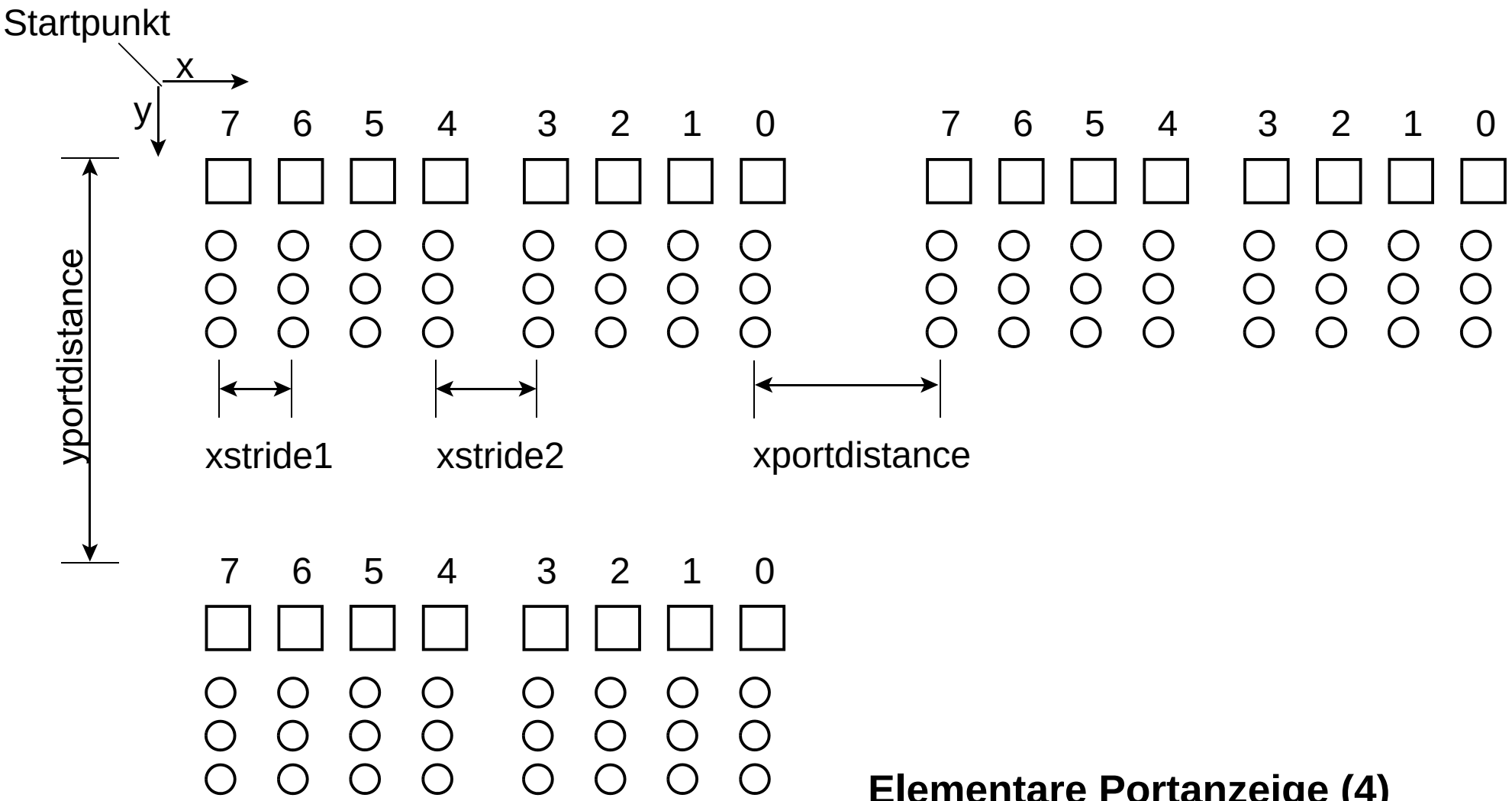
Ein Port (8 Bits):



Elementare Portanzeige (3)

Die Darstellung eines Ports

8. 4. 13

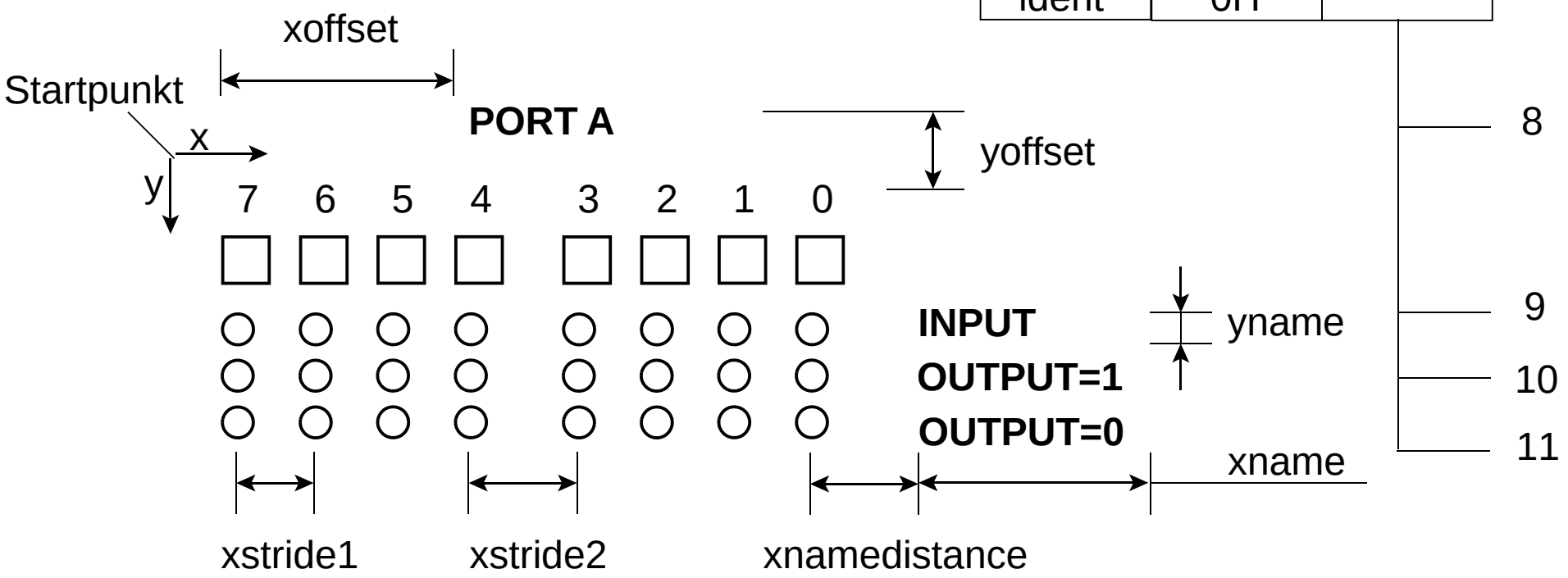


Elementare Portanzeige (4)

Die Darstellung mehrerer Ports

8. 4. 13

Der Portname:



Die Identifier der Beschriftung:

Elementare Portanzeige (5)

Die Beschriftung der Portanzeige

22. 5. 2015