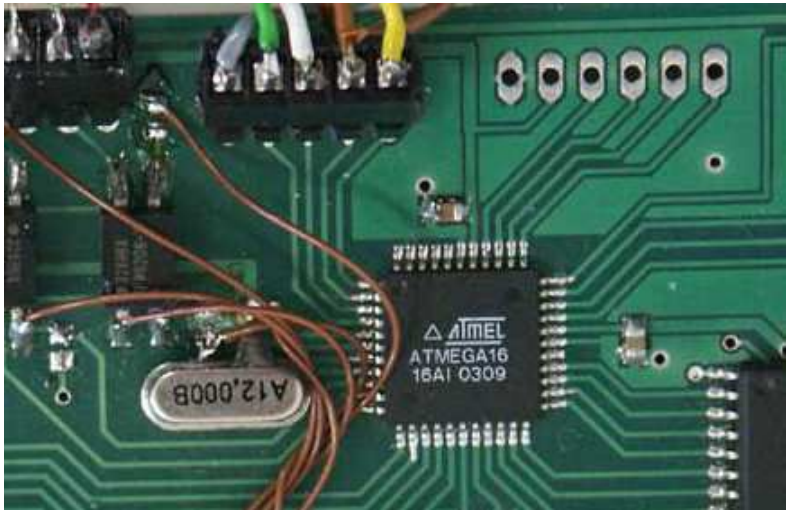
TGI-Großpraktikum - Gruppe 1

Jörg Böhnel, Markus Gerstel, Richard Röttger

Motorsteuerung, Bresenham-Algorithmus, Stiftverwaltung
Technische Universität München

18th July 2005

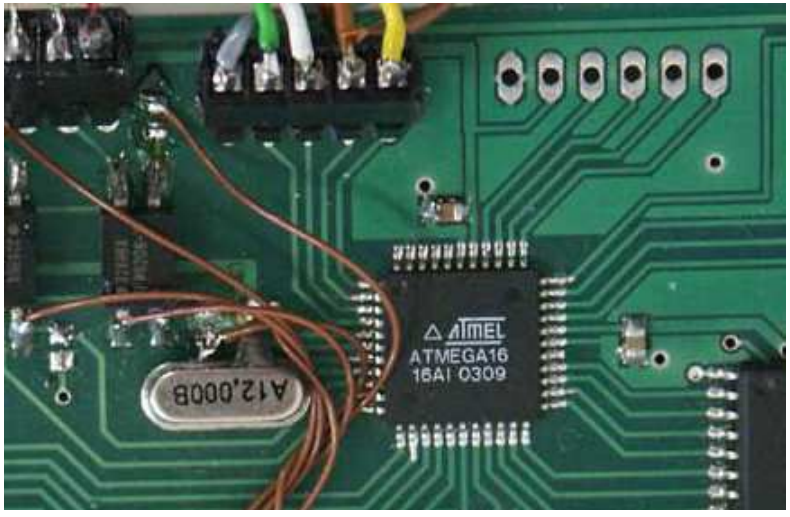
Generelle Problemstellung



Gegeben:

- 1 Plotter
- 1 Chip
- 1 großes Fragezeichen

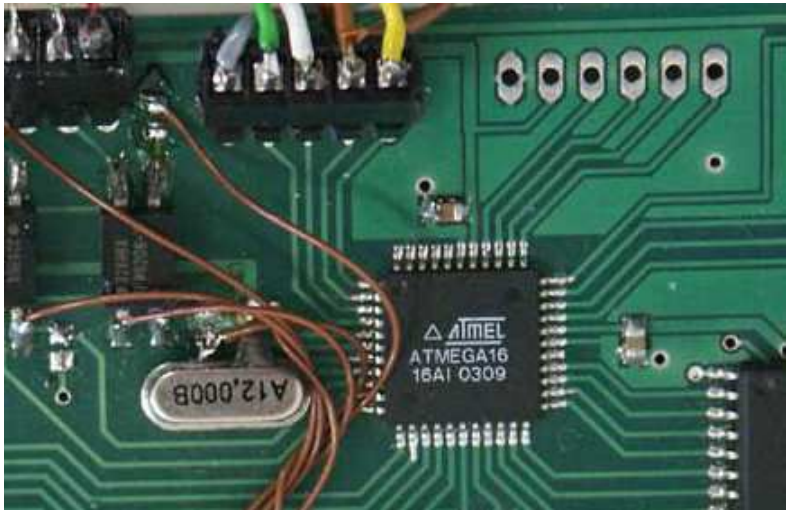
Generelle Problemstellung



Gegeben:

- 1 Plotter
- 1 Chip
- 1 großes Fragezeichen

Generelle Problemstellung



Gegeben:

- 1 Plotter
- 1 Chip
- 1 großes Fragezeichen

Generelle Problemstellung

Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

Generelle Problemstellung

Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung
  Aktuell: max. 1 Schritt pro 0,00064 s mit kleinerem Toleranzbereich
  Ansteuerungszeiten mit Auflösung ca. 0,00021 s = 21 µs

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

Generelle Problemstellung

Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung
  Aktuell: max. 1 Schritt pro 0,00064 s mit kleinerem Toleranzbereich
  Ansteuerungszeiten mit Auflösung ca. 0,000021 s = 21 μs

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

Generelle Problemstellung

Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung
  Aktuell: max. 1 Schritt pro 0,00064 s mit kleinerem Toleranzbereich
  Ansteuerungszeiten mit Auflösung ca. 0,000021 s = 21 μs

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

Generelle Problemstellung

Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung
  Aktuell: max. 1 Schritt pro 0,00064 s mit kleinerem Toleranzbereich
  Ansteuerungszeiten mit Auflösung ca. 0,000021 s $= 21\ \mu$s

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

Generelle Problemstellung

Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung
  Aktuell: max. 1 Schritt pro 0,00064 s mit kleinerem Toleranzbereich
  Ansteuerungszeiten mit Auflösung ca. 0,000021 s $= 21\ \mu$s

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

Generelle Problemstellung

Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung
  Aktuell: max. 1 Schritt pro 0,00064 s mit kleinerem Toleranzbereich
  Ansteuerungszeiten mit Auflösung ca. 0,000021 s $= 21\ \mu$s

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

Generelle Problemstellung

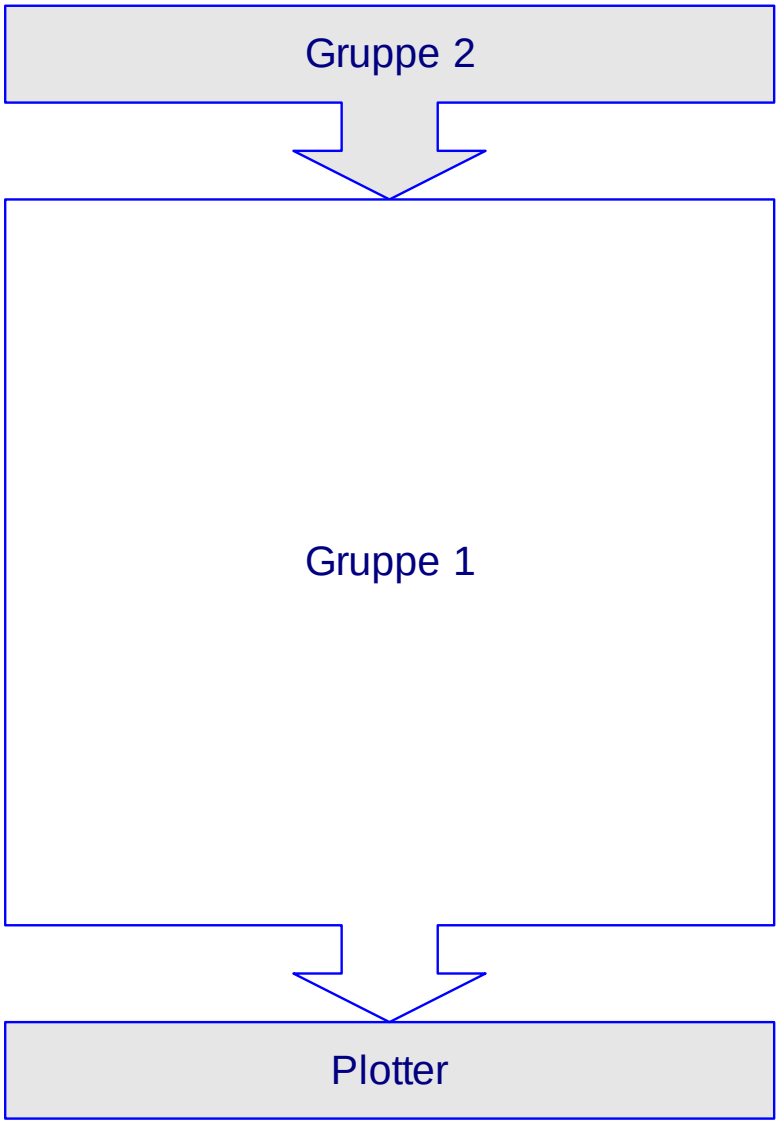
Probleme der Physik

- Halbschrittmotoren müssen kontinuierlich angesteuert werden
- Ansteuerungszeiten sind bei Beschleunigung unterschiedlich
- Je schneller der Plotterkopf desto schneller die Ansteuerung
  Aktuell: max. 1 Schritt pro 0,00064 s mit kleinerem Toleranzbereich
  Ansteuerungszeiten mit Auflösung ca. 0,000021 s = 21 μs

Weiter:

- Kein Multitasking
- Aber 2 Gruppen und 6 Personen

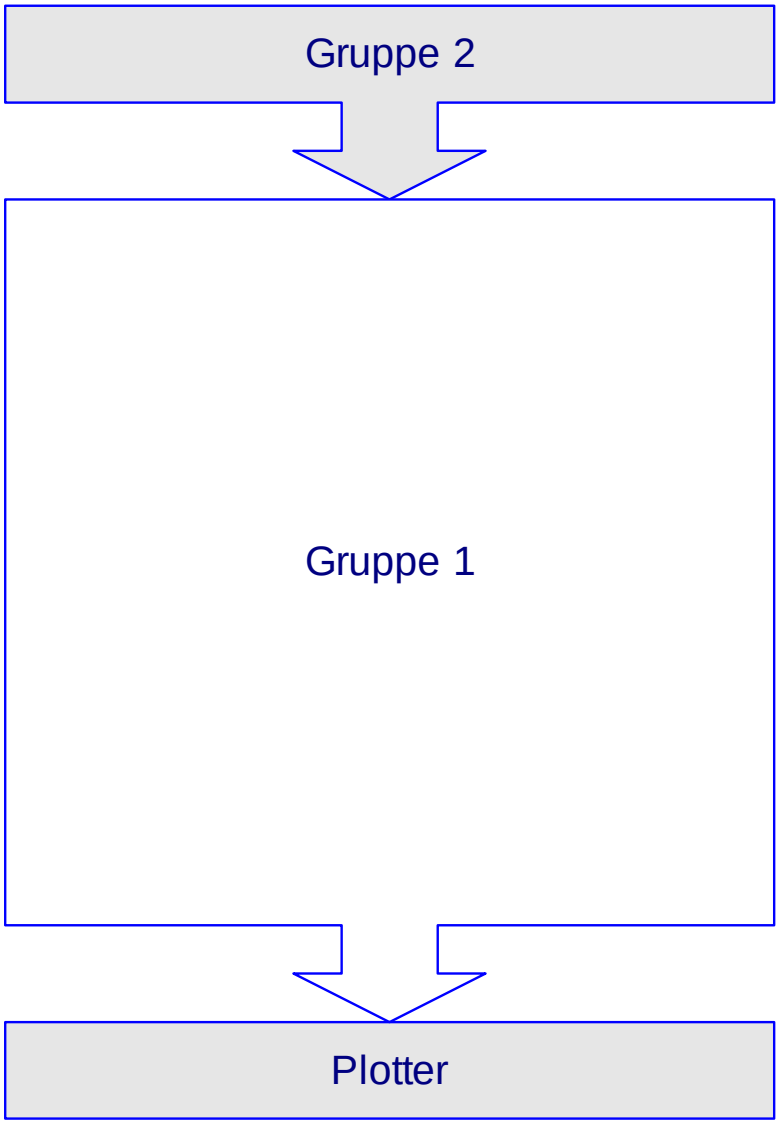
Generelle Problemstellung



Black-Box-Sicht:

- Befehle von oben
- Geheimnisvolle Arbeit
- Befehle nach unten

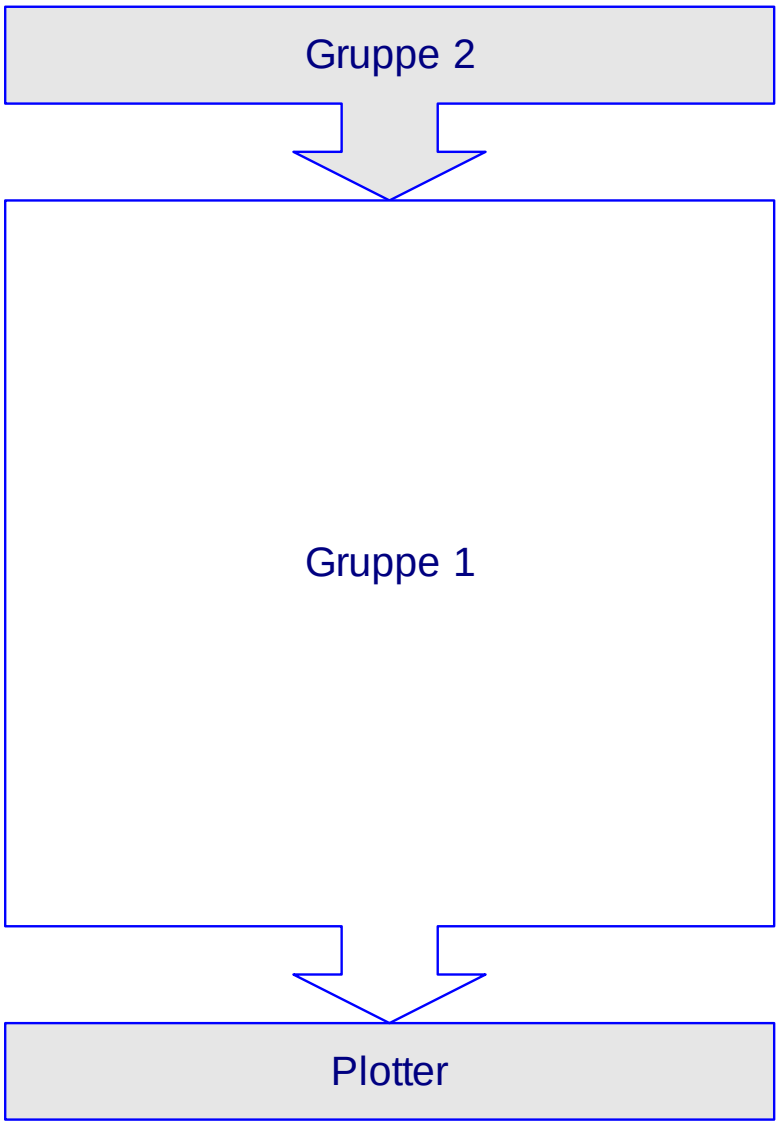
Generelle Problemstellung



Black-Box-Sicht:

- Befehle von oben
- Geheimnisvolle Arbeit
- Befehle nach unten

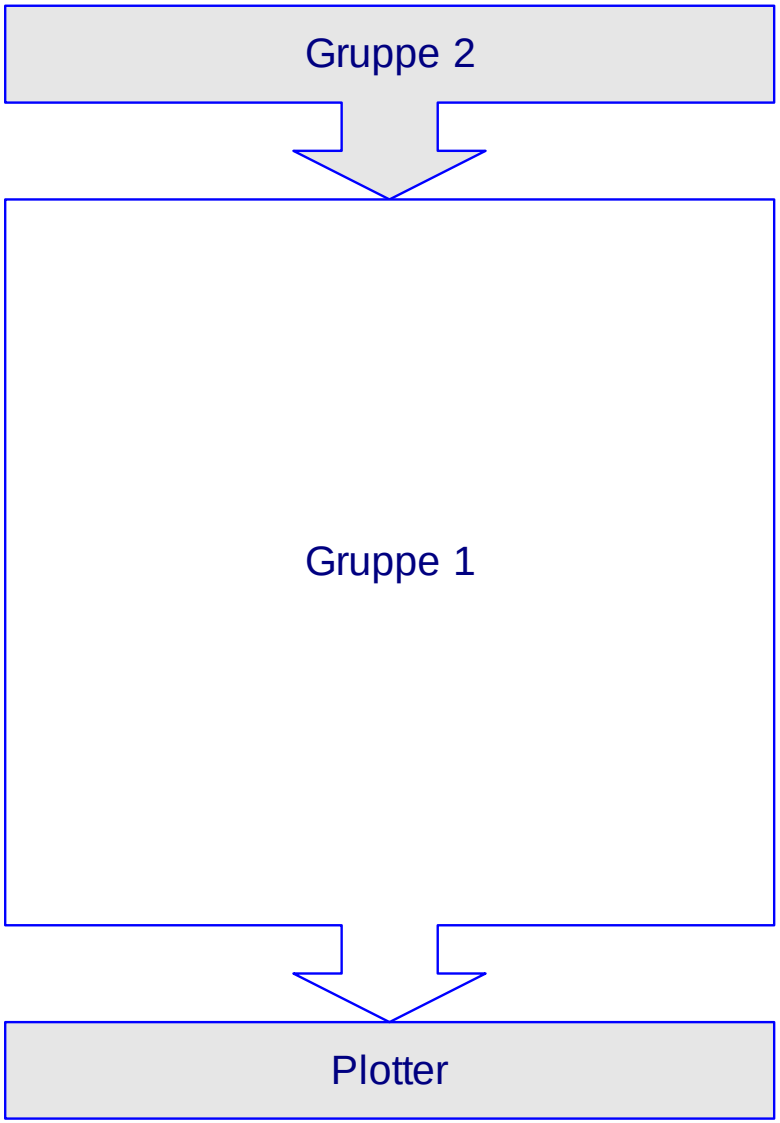
Generelle Problemstellung



Black-Box-Sicht:

- Befehle von oben
- Geheimnisvolle Arbeit
- Befehle nach unten

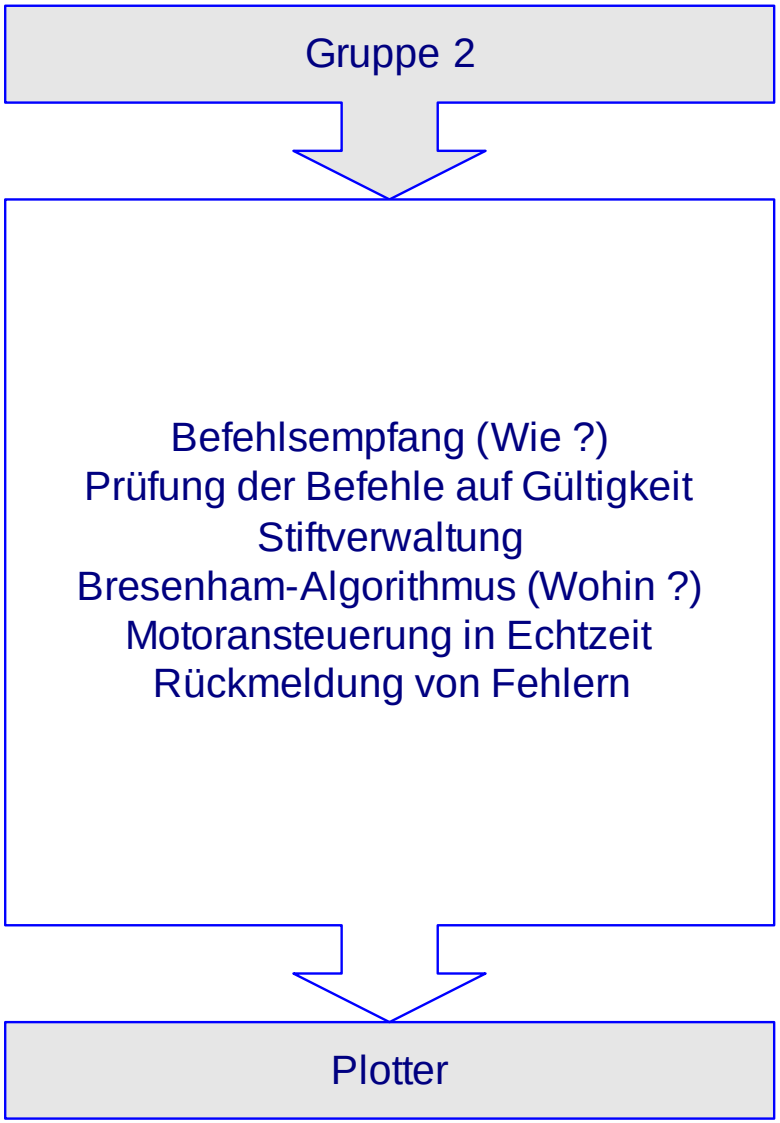
Generelle Problemstellung



Black-Box-Sicht:

- Befehle von oben
- Geheimnisvolle Arbeit
- Befehle nach unten

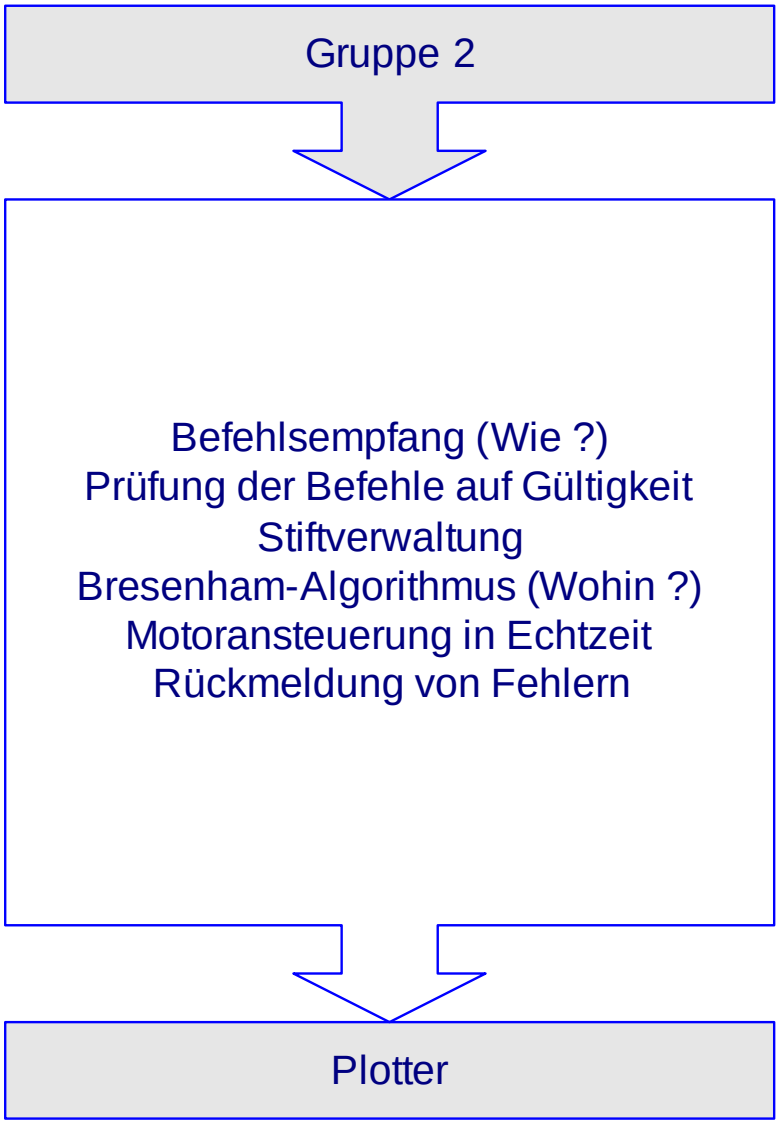
Problem auf Gruppenebene



Was muss getan werden ?

- Befehlsempfang
- Befehlsprüfung
- Stiftwechsel
- Bresenham
- Motoransteuerung

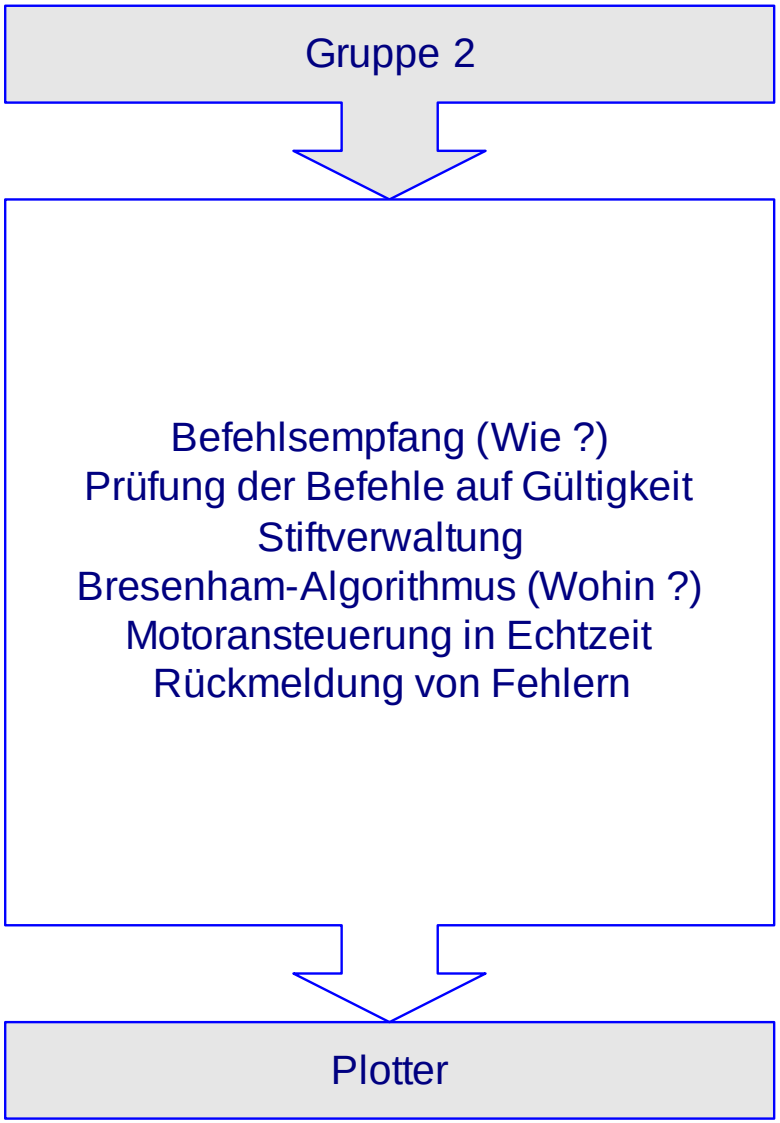
Problem auf Gruppenebene



Was muss getan werden ?

- Befehlsempfang
- Befehlsprüfung
- Stiftwechsel
- Bresenham
- Motoransteuerung

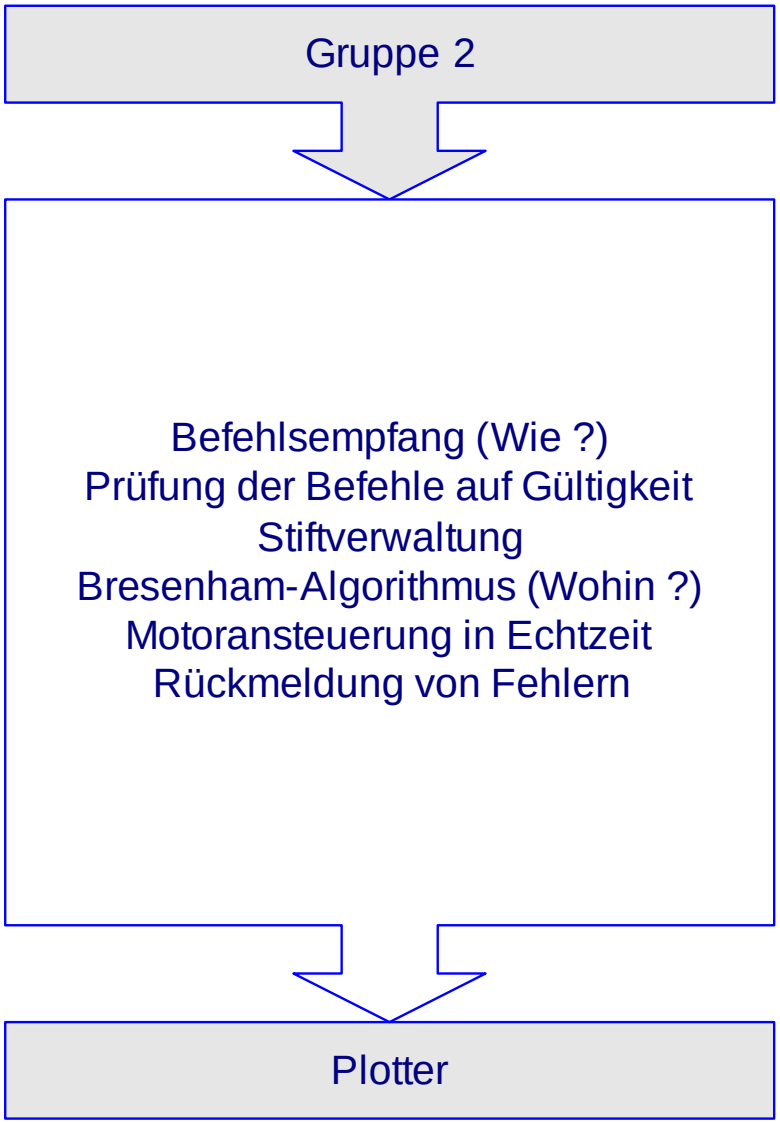
Problem auf Gruppenebene



Was muss getan werden ?

- Befehlsempfang
- Befehlsprüfung
- Stiftwechsel
- Bresenham
- Motoransteuerung

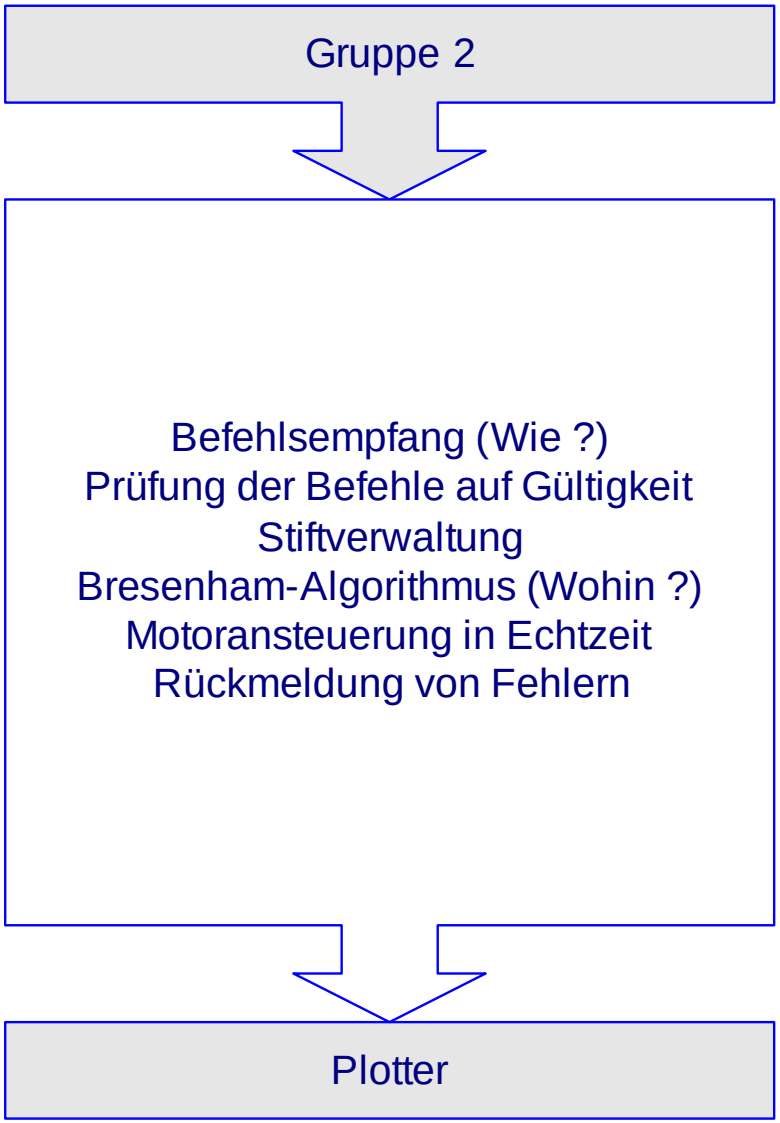
Problem auf Gruppenebene



Was muss getan werden ?

- Befehlsempfang
- Befehlsprüfung
- Stiftwechsel
- Bresenham
- Motoransteuerung

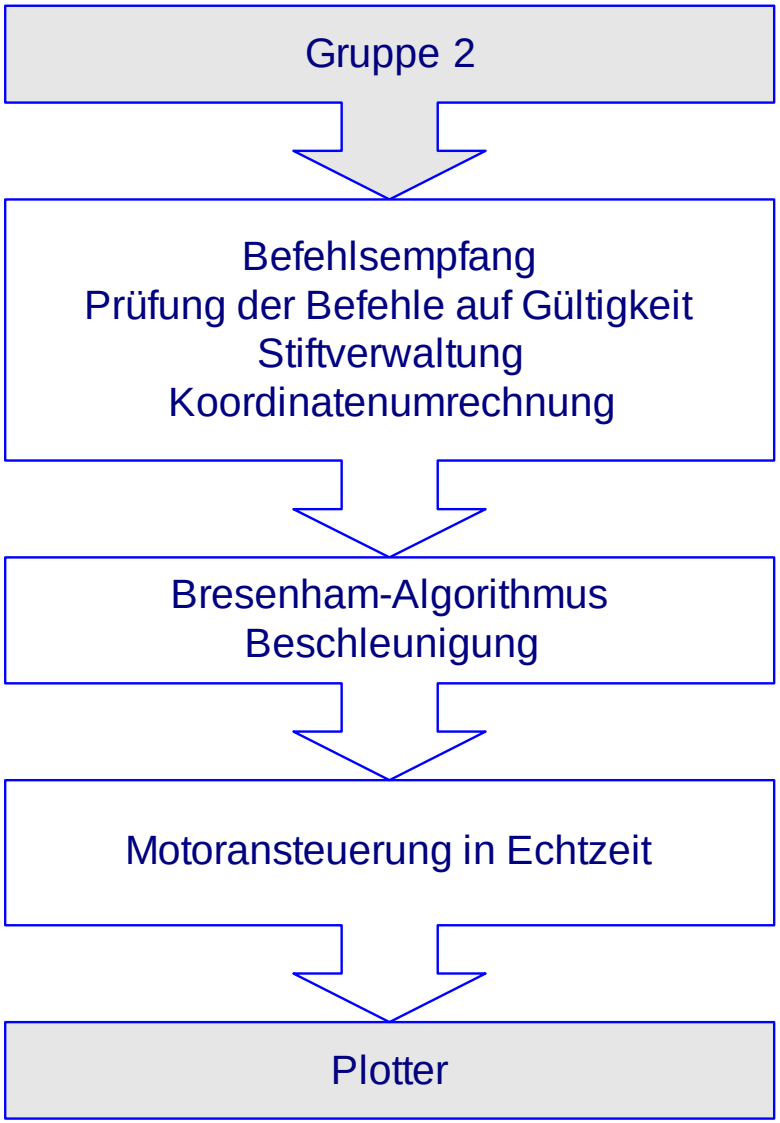
Problem auf Gruppenebene



Was muss getan werden ?

- Befehlsempfang
- Befehlsprüfung
- Stiftwechsel
- Bresenham
- Motoransteuerung

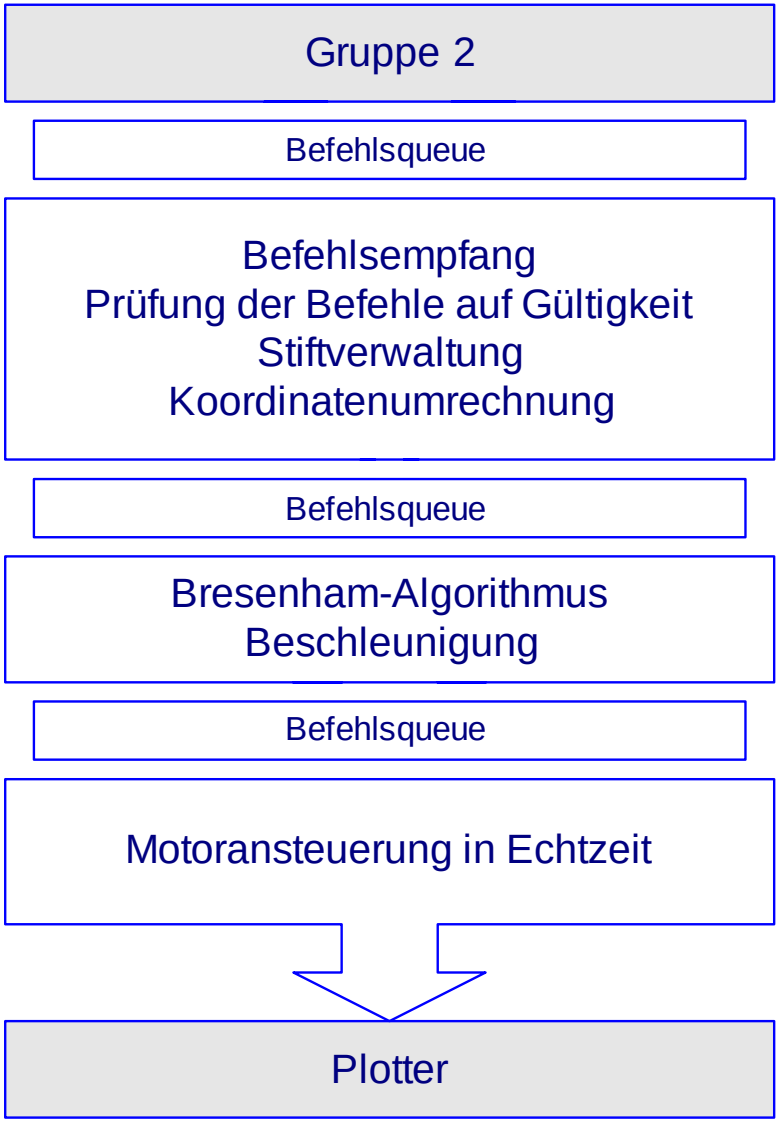
Problem auf Personenebene



Aufgabentrennung

3 Probleme $\leftrightarrow$ 3 Personen

Das Schichtenmodell



- Definition fester Schnittstellen

  Trennung durch FIFO-Warteschlangen

- Organisation & Synchronisation

  durch (Gruppen-)globales Statusregister

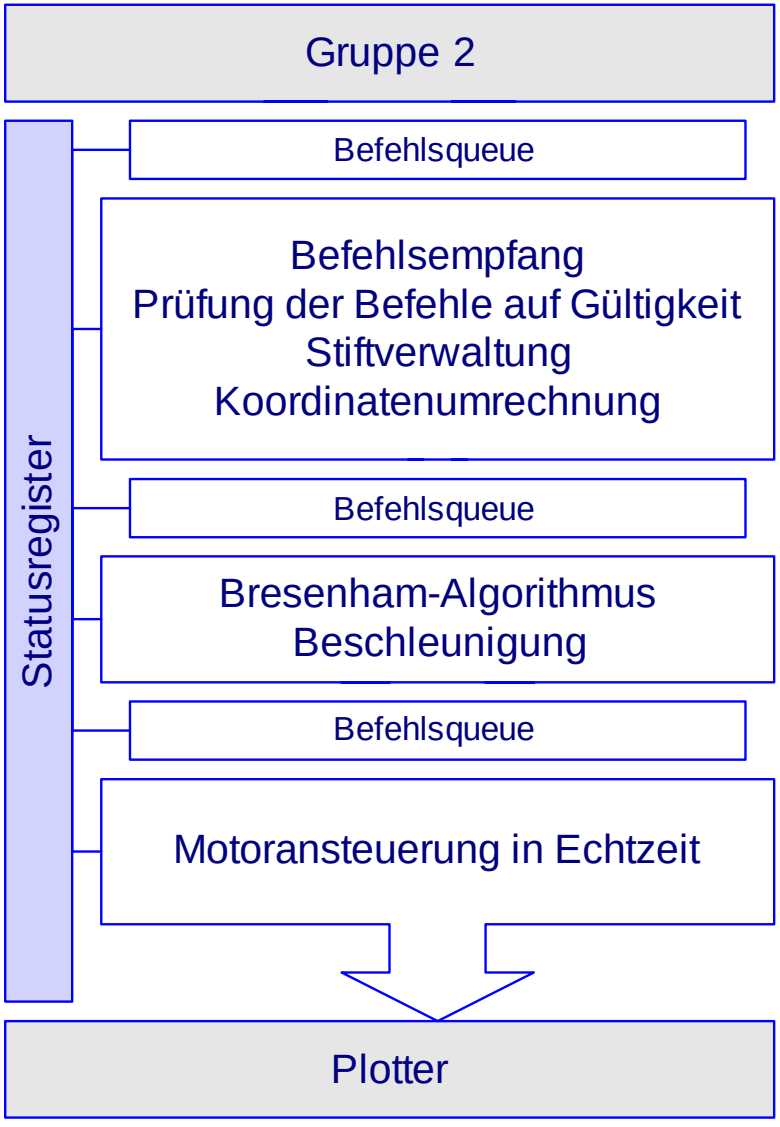
- Detaillierte Fehlermeldungen

  durch (Gruppen-)globales Fehlerregister

- Dispatcher

  "intelligente" Mainloop

Das Schichtenmodell



- Definition fester Schnittstellen

  Trennung durch FIFO-Warteschlangen

- Organisation & Synchronisation

  durch (Gruppen-)globales Statusregister

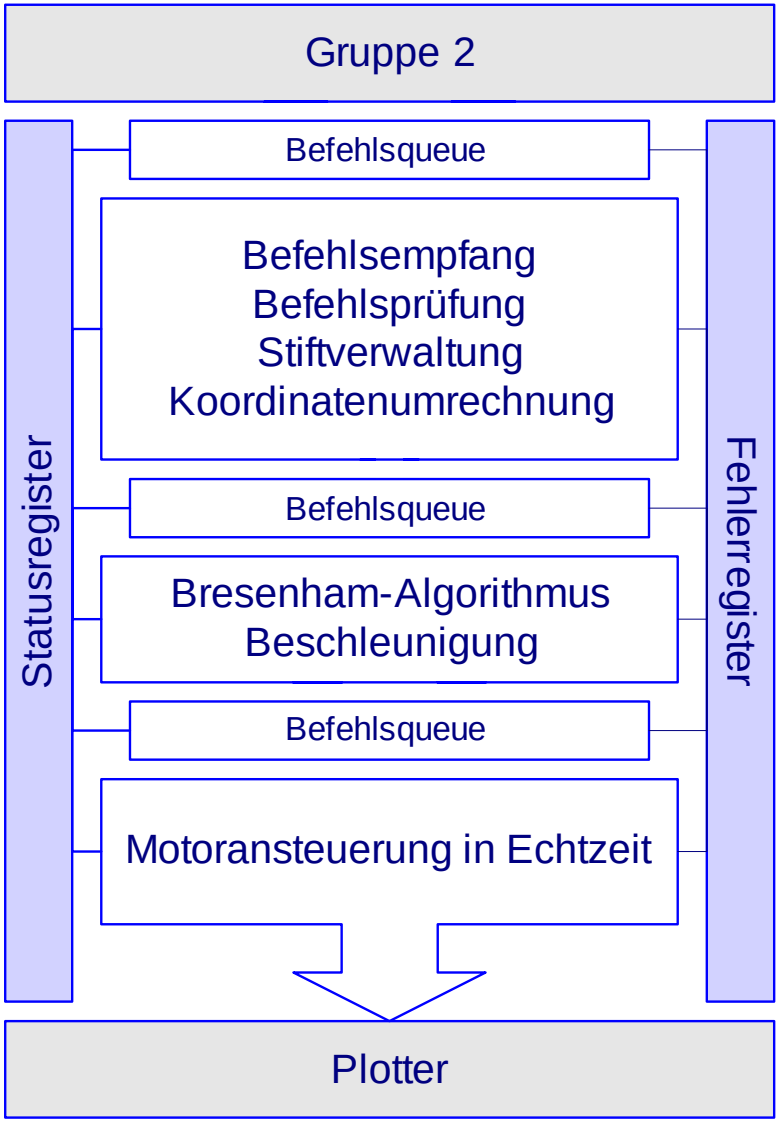
- Detaillierte Fehlermeldungen

  durch (Gruppen-)globales Fehlerregister

- Dispatcher

  "intelligente" Mainloop

Das Schichtenmodell



- Definition fester Schnittstellen

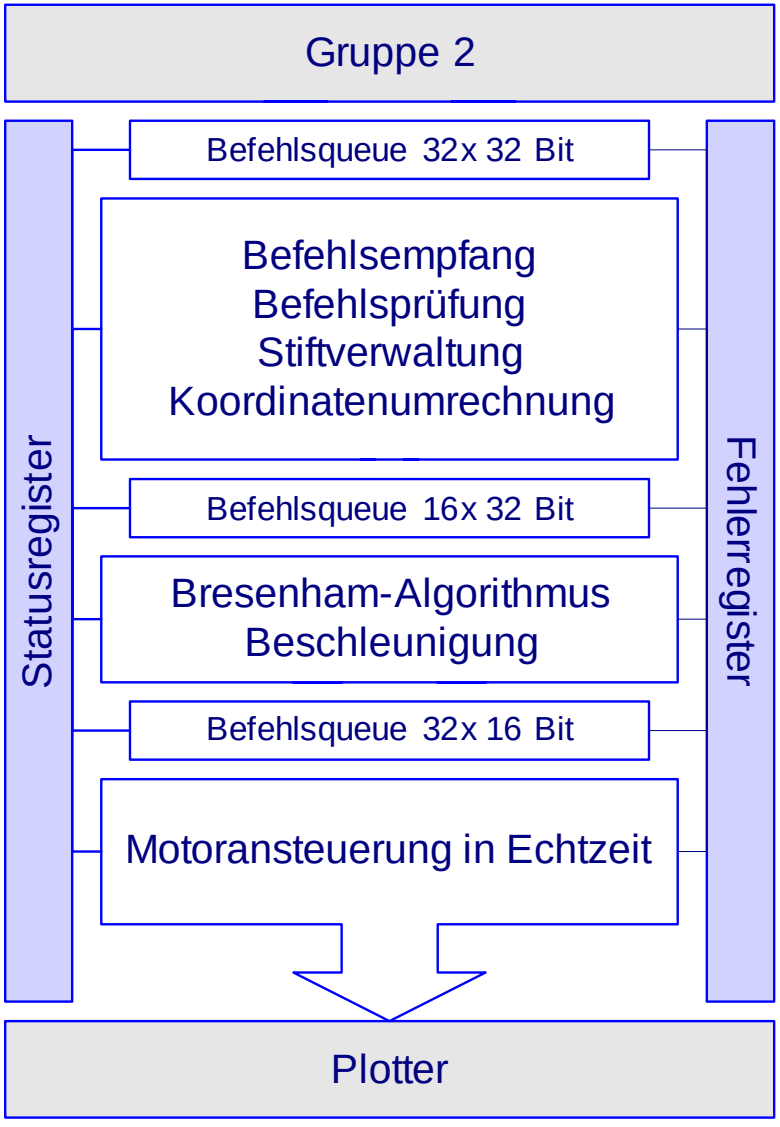
  Trennung durch FIFO-Warteschlangen
- Organisation & Synchronisation

  durch (Gruppen-)globales Statusregister
- Detaillierte Fehlermeldungen

  durch (Gruppen-)globales Fehlerregister
- Dispatcher

  "intelligente" Mainloop

Das Schichtenmodell



- Definition fester Schnittstellen

  Trennung durch FIFO-Warteschlangen

- Organisation & Synchronisation

  durch (Gruppen-)globales Statusregister

- Detaillierte Fehlermeldungen

  durch (Gruppen-)globales Fehlerregister

- Dispatcher

  "intelligente" Mainloop

Vorteile des Schichtenmodells

- Vollständige Trennung der Aufgabenbereiche
  (Keiner fummelt in fremdem Code - Schön wärs...)
- Feste, deterministische Schnittstellen
- Leichtes Debugging für alle Beteiligten (Fire and Forget)
- Asynchrone Ausführung (Quasi-Multitasking)
- Schönes Schichtenmodell, modularer Code, leicht erweiterbar

Vorteile des Schichtenmodells

- Vollständige Trennung der Aufgabenbereiche
  (Keiner fummelt in fremdem Code - Schön wärs...)
- Feste, deterministische Schnittstellen
- Leichtes Debugging für alle Beteiligten (Fire and Forget)
- Asynchrone Ausführung (Quasi-Multitasking)
- Schönes Schichtenmodell, modularer Code, leicht erweiterbar

Vorteile des Schichtenmodells

- Vollständige Trennung der Aufgabenbereiche
  (Keiner fummelt in fremdem Code - Schön wärs...)
- Feste, deterministische Schnittstellen
- Leichtes Debugging für alle Beteiligten (Fire and Forget)
- Asynchrone Ausführung (Quasi-Multitasking)
- Schönes Schichtenmodell, modularer Code, leicht erweiterbar

Vorteile des Schichtenmodells

- Vollständige Trennung der Aufgabenbereiche
  (Keiner fummelt in fremdem Code - Schön wärs...)
- Feste, deterministische Schnittstellen
- Leichtes Debugging für alle Beteiligten (Fire and Forget)
- Asynchrone Ausführung (Quasi-Multitasking)
- Schönes Schichtenmodell, modularer Code, leicht erweiterbar

Vorteile des Schichtenmodells

- Vollständige Trennung der Aufgabenbereiche
  (Keiner fummelt in fremdem Code - Schön wärs...)
- Feste, deterministische Schnittstellen
- Leichtes Debugging für alle Beteiligten (Fire and Forget)
- Asynchrone Ausführung (Quasi-Multitasking)
- Schönes Schichtenmodell, modularer Code, leicht erweiterbar

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen
  - etwa 60 Zyklen $\longrightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben
- ...

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen etwa 60 Zyklen $\longrightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben
- ...

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen etwa 60 Zyklen $\longrightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben
- ...

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen etwa 60 Zyklen $\longrightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben
- ...

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen
  - etwa 60 Zyklen $\rightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben
- ...

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen
  - etwa 60 Zyklen $\rightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben
- ...

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen
  - etwa 60 Zyklen $\rightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben

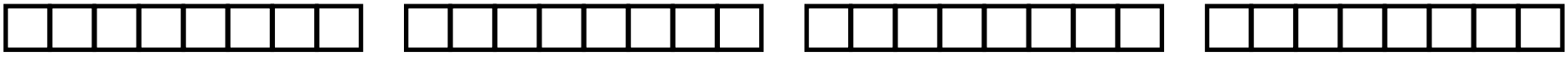
...

Nachteile des Schichtenmodells

- Hoher Speicherbedarf
  - aktuell 256 / 1024 Bytes SRAM nur für Queues
  - Defektes Bit fragmentiert Speicher zusätzlich
- Zusätzlicher Verwaltungsoverhead
  - Insert/Retrieve-Operationen benötigen
  - etwa 60 Zyklen $\rightarrow$ 5 μs
- 6 Personen müssen sich einigen
- Irgendjemand muss den Code schreiben
  Toll ein anderer machts.

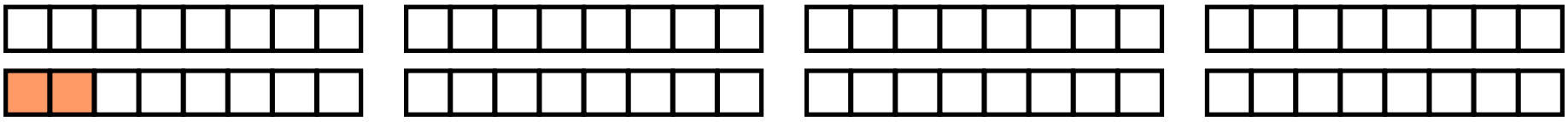
Befehlsformat Gruppe 2 → Gruppe 1

Mögliche Befehle:



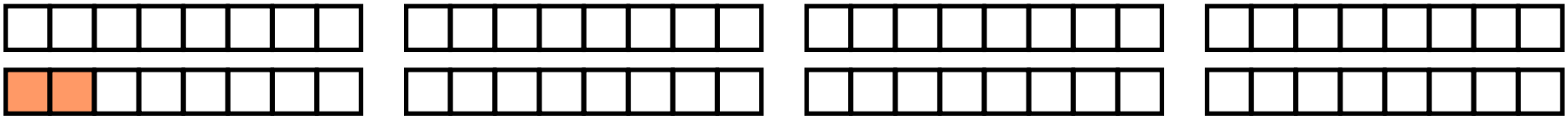
Befehlsformat Gruppe 2 → Gruppe 1

Mögliche Befehle:

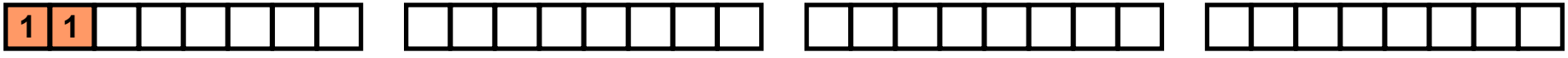


Befehlsformat Gruppe 2 → Gruppe 1

Mögliche Befehle:



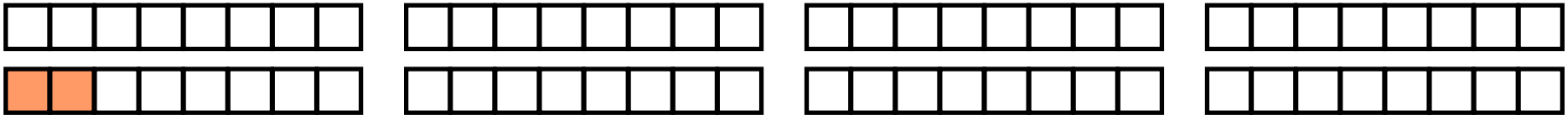
Initialisierung:



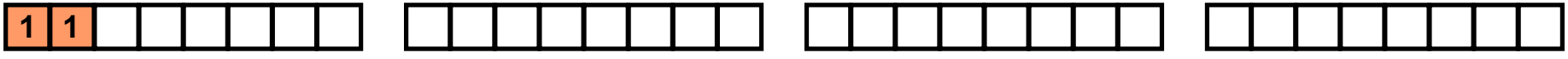
(Zuordnung der belegten Stifte)

Befehlsformat Gruppe 2 → Gruppe 1

Mögliche Befehle:

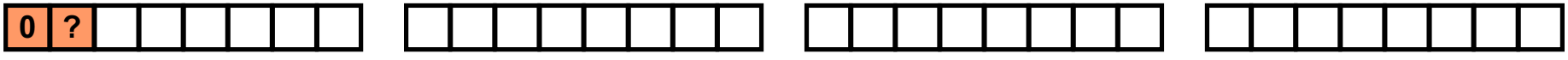


Initialisierung:



(Zuordnung der belegten Stifte)

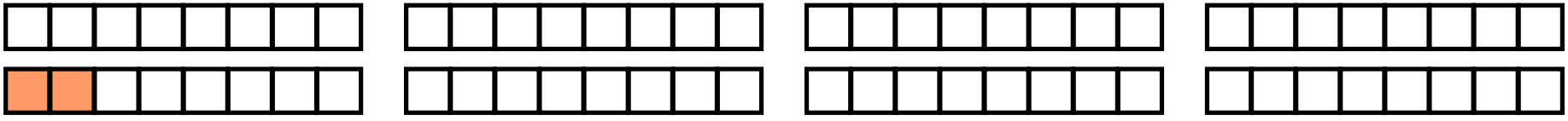
Fahre zu:



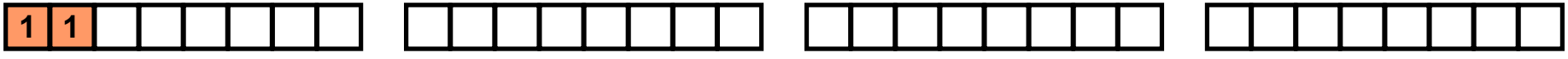
(X, Y - Positionen mit Stift oben/unten anfahren)

Befehlsformat Gruppe 2 → Gruppe 1

Mögliche Befehle:

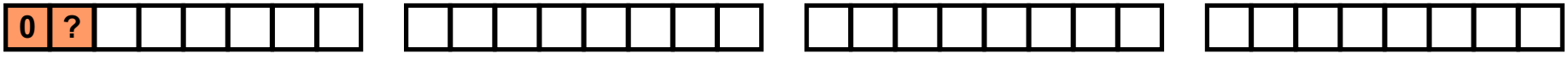


Initialisierung:



(Zuordnung der belegten Stifte)

Fahre zu:

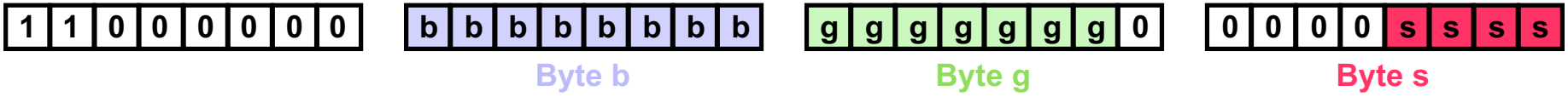


(X, Y - Positionen mit Stift oben/unten anfahren)

Wechsle Stift:

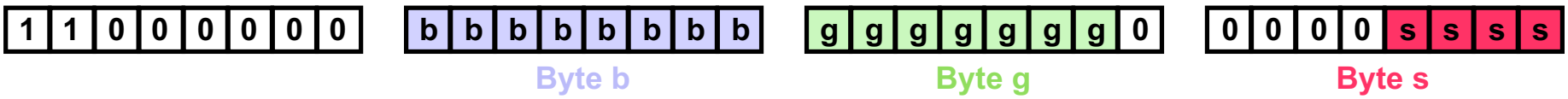
(Hole Stift; evtl. Stift vorher ablegen, danach Position X, Y anfahren)

Befehl: Initialisierung



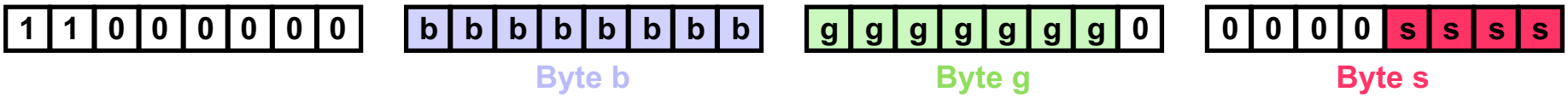
- Byte b: (belegt)
  Bit $n = 1$: Stifthalter n wird verwendet
  Bit 0 entspricht dem Plotterkopf
- Byte g: (Geschwindigkeit begrenzen)
  Bit $m = 1$: Stift im Stifthalter m darf nur langsam fahren
  Bit 0 muss immer 0 sein!
- Byte s: (Stift Nr.)
  Bit $0 \ldots 3$: Stift Nr. s befindet sich momentan im Plotterkopf
  Byte b, Bit $0 = 0 \Rightarrow s$ muss 0 sein

Befehl: Initialisierung



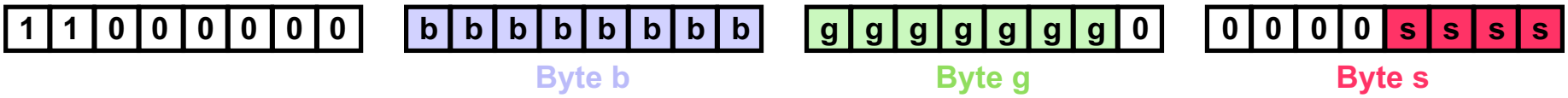
- Byte b: (belegt)
  Bit $n = 1$: Stifthalter n wird verwendet
  Bit 0 entspricht dem Plotterkopf
- Byte g: (Geschwindigkeit begrenzen)
  Bit $m = 1$: Stift im Stifthalter m darf nur langsam fahren
  Bit 0 muss immer 0 sein!
- Byte s: (Stift Nr.)
  Bit $0 \ldots 3$: Stift Nr. s befindet sich momentan im Plotterkopf
  Byte b, Bit $0 = 0 \Rightarrow s$ muss 0 sein

Befehl: Initialisierung



- Byte b: (belegt)
  Bit $n = 1$: Stifthalter n wird verwendet
  Bit 0 entspricht dem Plotterkopf
- Byte g: (Geschwindigkeit begrenzen)
  Bit $m = 1$: Stift im Stifthalter m darf nur langsam fahren
  Bit 0 muss immer 0 sein!
- Byte s: (Stift Nr.)
  Bit $0 \ldots 3$: Stift Nr. s befindet sich momentan im Plotterkopf
  Byte b, Bit $0 = 0 \Rightarrow s$ muss 0 sein

Befehl: Initialisierung



- Byte b: (belegt)
  Bit $n = 1$: Stifthalter n wird verwendet
  Bit 0 entspricht dem Plotterkopf
- Byte g: (Geschwindigkeit begrenzen)
  Bit $m = 1$: Stift im Stifthalter m darf nur langsam fahren
  Bit 0 muss immer 0 sein!
- Byte s: (Stift Nr.)
  Bit $0 \ldots 3$: Stift Nr. s befindet sich momentan im Plotterkopf
  Byte b, Bit $0 = 0 \Rightarrow s$ muss 0 sein

Befehl: Initialisierung - Beispiel I

- Stifthalter 1,3 und 4 werden verwendet und sind je mit einem Stift belegt
- Mit den Stiften 3 und 4 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich kein Stift

Befehl: Initialisierung - Beispiel I

- Stifthalter 1,3 und 4 werden verwendet und sind je mit einem Stift belegt
- Mit den Stiften 3 und 4 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich kein Stift

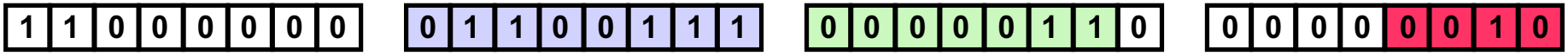
Befehl: Initialisierung - Beispiel I

- Stifthalter 1,3 und 4 werden verwendet und sind je mit einem Stift belegt
- Mit den Stiften 3 und 4 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich kein Stift

Befehl: Initialisierung - Beispiel I

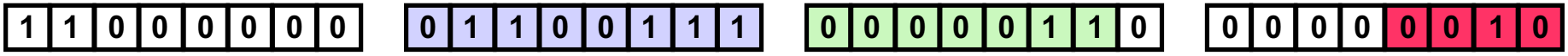
- Stifthalter 1,3 und 4 werden verwendet und sind je mit einem Stift belegt
- Mit den Stiften 3 und 4 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich kein Stift

Befehl: Initialisierung - Beispiel II



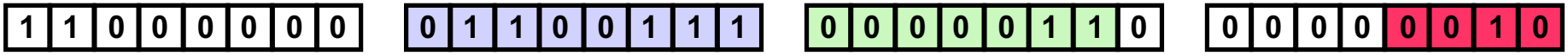
- Stifthalter 1,2,5 und 6 werden verwendet.
- Mit den Stiften 1 und 2 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich der Stift von Halter 2
- Die Stifthalter 1,5 und 6 sind mit je einem Stift belegt
- Wichtig: Stifthalter 2 muss leer sein!

Befehl: Initialisierung - Beispiel II



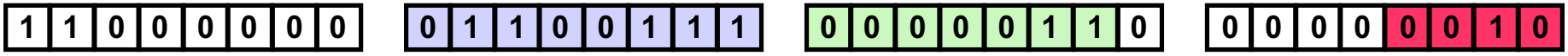
- Stifthalter 1,2,5 und 6 werden verwendet.
- Mit den Stiften 1 und 2 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich der Stift von Halter 2
- Die Stifthalter 1,5 und 6 sind mit je einem Stift belegt
- Wichtig: Stifthalter 2 muss leer sein!

Befehl: Initialisierung - Beispiel II



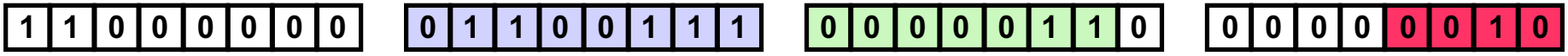
- Stifthalter 1,2,5 und 6 werden verwendet.
- Mit den Stiften 1 und 2 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich der Stift von Halter 2
- Die Stifthalter 1,5 und 6 sind mit je einem Stift belegt
- Wichtig: Stifthalter 2 muss leer sein!

Befehl: Initialisierung - Beispiel II



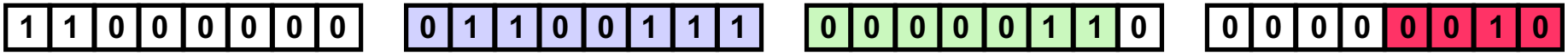
- Stifthalter 1,2,5 und 6 werden verwendet.
- Mit den Stiften 1 und 2 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich der Stift von Halter 2
- Die Stifthalter 1,5 und 6 sind mit je einem Stift belegt
- Wichtig: Stifthalter 2 muss leer sein!

Befehl: Initialisierung - Beispiel II



- Stifthalter 1,2,5 und 6 werden verwendet.
- Mit den Stiften 1 und 2 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich der Stift von Halter 2
- Die Stifthalter 1,5 und 6 sind mit je einem Stift belegt
- Wichtig: Stifthalter 2 muss leer sein!

Befehl: Initialisierung - Beispiel II



- Stifthalter 1,2,5 und 6 werden verwendet.
- Mit den Stiften 1 und 2 darf nur langsam gefahren werden
- Im Plotterkopf befindet sich der Stift von Halter 2
- Die Stifthalter 1,5 und 6 sind mit je einem Stift belegt
- Wichtig: Stifthalter 2 muss leer sein!

Befehl: Fahre zu

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- c: Endgeschwindigkeit

Befehl: Fahre zu

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- c: Endgeschwindigkeit

Befehl: Fahre zu

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- c: Endgeschwindigkeit

Befehl: Fahre zu

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- c: Endgeschwindigkeit

Befehl: Fahre zu

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- c: Endgeschwindigkeit

Befehl: Wechsle Stift

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift Nr. s, der geholt werden soll

Befehl: Wechsle Stift

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift Nr. s, der geholt werden soll

Befehl: Wechsle Stift

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift Nr. s, der geholt werden soll

Befehl: Wechsle Stift

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift Nr. s, der geholt werden soll

Midqueue: CFC → Bresenham

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- g: Geschwindigkeit begrenzen
- c: Endgeschwindigkeit

Midqueue: CFC → Bresenham

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- g: Geschwindigkeit begrenzen
- c: Endgeschwindigkeit

Midqueue: CFC → Bresenham

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- g: Geschwindigkeit begrenzen
- c: Endgeschwindigkeit

Midqueue: CFC → Bresenham

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- g: Geschwindigkeit begrenzen
- c: Endgeschwindigkeit

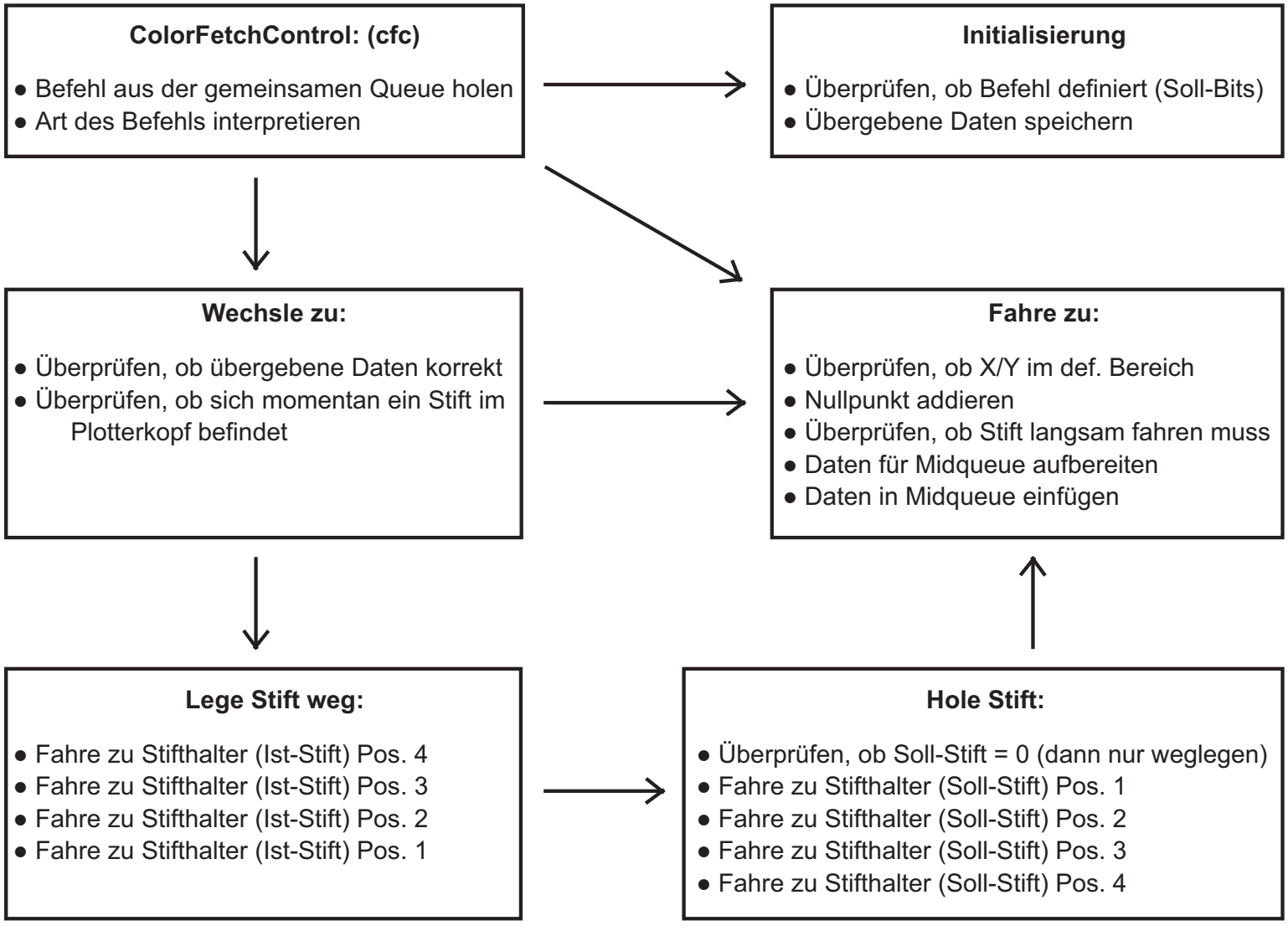
Midqueue: CFC → Bresenham

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- g: Geschwindigkeit begrenzen
- c: Endgeschwindigkeit

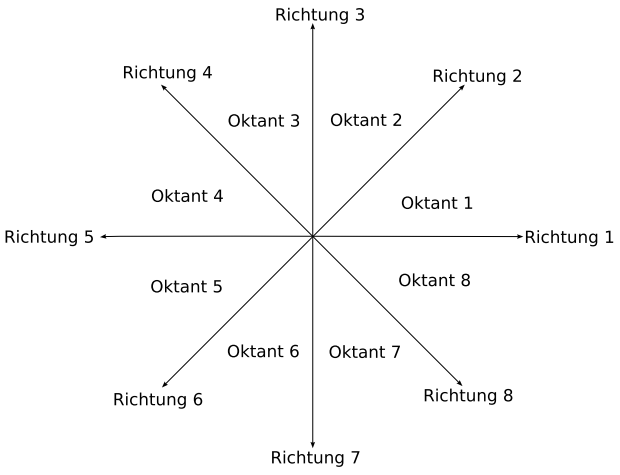
Midqueue: CFC → Bresenham

- x: x-Koordinate, zu der gefahren werden soll.
- y: y-Koordinate, zu der gefahren werden soll.
- s: Stift oben/unten
- g: Geschwindigkeit begrenzen
- c: Endgeschwindigkeit

CFC - Übersicht



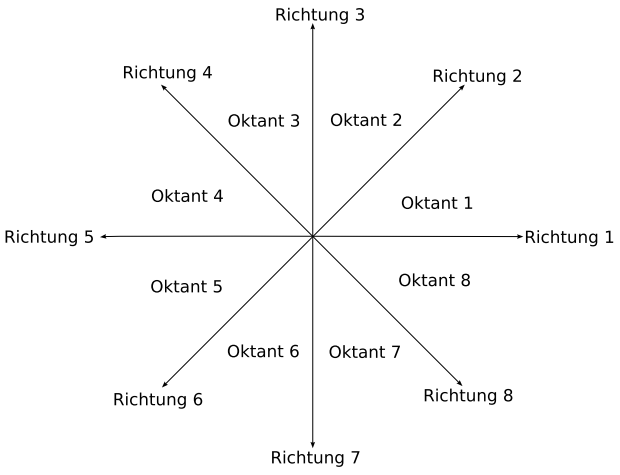
Beobachtung:



Allgemeine Beobachtung:

- Man kann durch geschicktes Vertauschen der Δx und Δy jede Linie auf eine Linie im Ersten Oktanten abbilden!
- In einem Oktanten kommen dann nur zwei Richtungen in Betracht - nämlich diejenigen, die den Oktanten umfassen
- Auf dem Plotter ist eine Unendlichnorm definiert, d.h. die Länge l einer Linie ist: $l = max\{\Delta x, \Delta y\}$

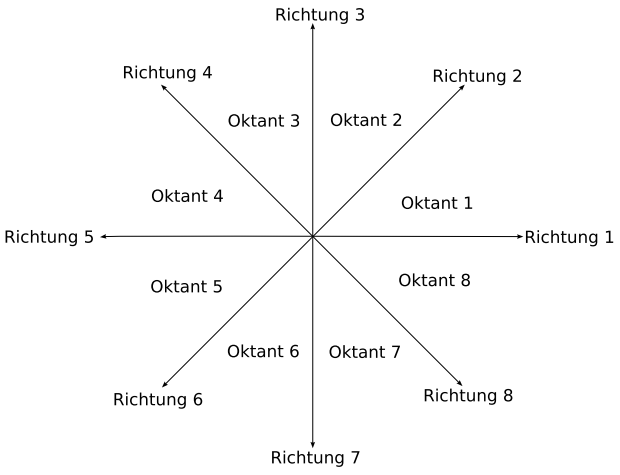
Beobachtung:



Allgemeine Beobachtung:

- Man kann durch geschicktes Vertauschen der Δx und Δy jede Linie auf eine Linie im Ersten Oktanten abbilden!
- In einem Oktanten kommen dann nur zwei Richtungen in Betracht - nämlich diejenigen, die den Oktanten umfassen
- Auf dem Plotter ist eine Unendlichnorm definiert, d.h. die Länge l einer Linie ist: $l = max\{\Delta x, \Delta y\}$

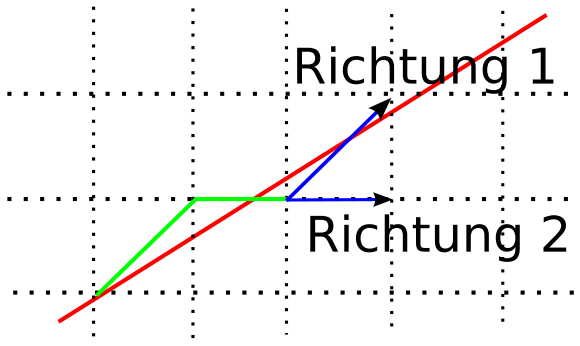
Beobachtung:



Allgemeine Beobachtung:

- Man kann durch geschicktes Vertauschen der Δx und Δy jede Linie auf eine Linie im Ersten Oktanten abbilden!
- In einem Oktanten kommen dann nur zwei Richtungen in Betracht - nämlich diejenigen, die den Oktanten umfassen
- Auf dem Plotter ist eine Unendlichnorm definiert, d.h. die Länge l einer Linie ist: $l = max\{\Delta x, \Delta y\}$

Konkrete Implementierung - Bresenham-Algorithmus



Wir weisen je nach Oktant die Richtungen zu, und setzten Δa und Δb entweder auf Δx oder Δy. Dann:

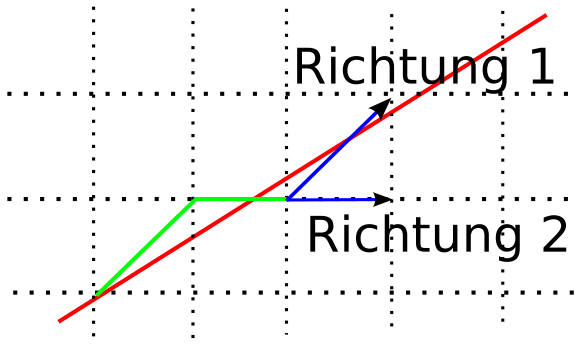
- $\Delta_1 = 2 \cdot \Delta b - \Delta a$
- Wenn $\Delta_i \geq 0$ dann fahre Richtung 1 und setze

$$\Delta_{i+1} = \Delta_i + 2 \cdot \Delta b - 2 \cdot \Delta a$$

- Wenn $\Delta_i < 0$ dann fahre Richtung 2 und setze

$$\Delta_{i+1} = \Delta_i + 2 \cdot \Delta b$$

Konkrete Implementierung - Bresenham-Algorithmus



Wir weisen je nach Oktant die Richtungen zu, und setzten Δa und Δb entweder auf Δx oder Δy. Dann:

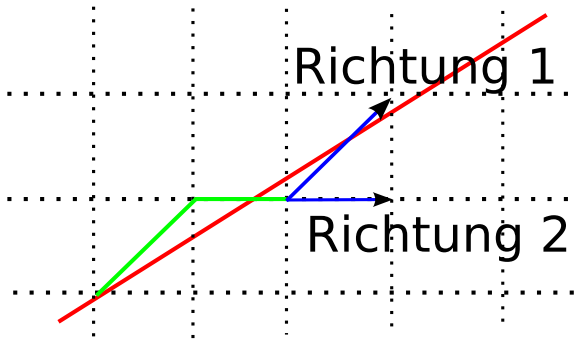
- $\Delta_1 = 2 \cdot \Delta b - \Delta a$
- Wenn $\Delta_i \geq 0$ dann fahre Richtung 1 und setze

$$\Delta_{i+1} = \Delta_i + 2 \cdot \Delta b - 2 \cdot \Delta a$$

- Wenn $\Delta_i < 0$ dann fahre Richtung 2 und setze

$$\Delta_{i+1} = \Delta_i + 2 \cdot \Delta b$$

Konkrete Implementierung - Bresenham-Algorithmus



Wir weisen je nach Oktant die Richtungen zu, und setzten Δa und Δb entweder auf Δx oder Δy. Dann:

- $\Delta_1 = 2 \cdot \Delta b - \Delta a$
- Wenn $\Delta_i \geq 0$ dann fahre Richtung 1 und setze

$$\Delta_{i+1} = \Delta_i + 2 \cdot \Delta b - 2 \cdot \Delta a$$

- Wenn $\Delta_i < 0$ dann fahre Richtung 2 und setze

$$\Delta_{i+1} = \Delta_i + 2 \cdot \Delta b$$

Beschleunigung: Probleme

Die Beschleunigung wird in unserer Umsetzung durch Wartewerte bestimmt. Je größer der Wert, desto länger veharrt der Plotter in der Stellung, desto langsamer fährt er. Dabei traten folgende Probleme auf:

- Lineare Beschleunigung geht nicht $\Rightarrow$ Schlenker
- Aufwendige Berechnungen für Kurven machen auf 8-Bit keinen Spass

Lösung: Wir speichern die Kurve ins EEPROM, und bekommen dadurch eine schöne Kurve ohne Berechnungen

Beschleunigung: Probleme

Die Beschleunigung wird in unserer Umsetzung durch Wartewerte bestimmt. Je größer der Wert, desto länger veharrt der Plotter in der Stellung, desto langsamer fährt er. Dabei traten folgende Probleme auf:

- Lineare Beschleunigung geht nicht $\Rightarrow$ Schlenker
- Aufwendige Berechnungen für Kurven machen auf 8-Bit keinen Spass

Lösung: Wir speichern die Kurve ins EEPROM, und bekommen dadurch eine schöne Kurve ohne Berechnungen

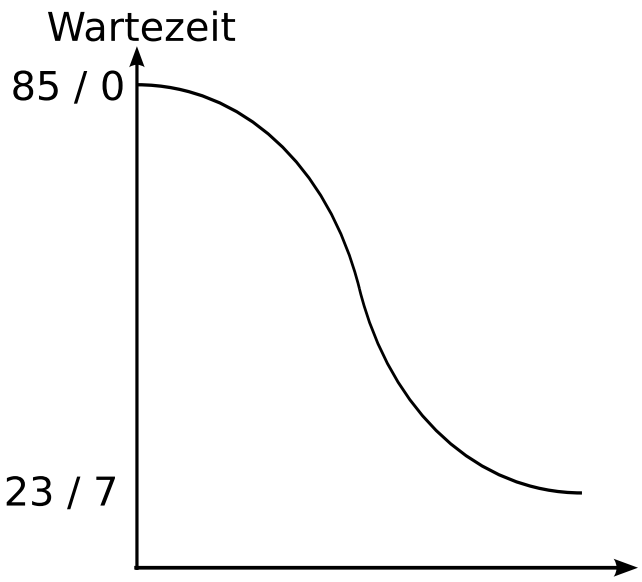
Beschleunigung: Probleme

Die Beschleunigung wird in unserer Umsetzung durch Wartewerte bestimmt. Je größer der Wert, desto länger veharrt der Plotter in der Stellung, desto langsamer fährt er. Dabei traten folgende Probleme auf:

- Lineare Beschleunigung geht nicht $\Rightarrow$ Schlenker
- Aufwendige Berechnungen für Kurven machen auf 8-Bit keinen Spass

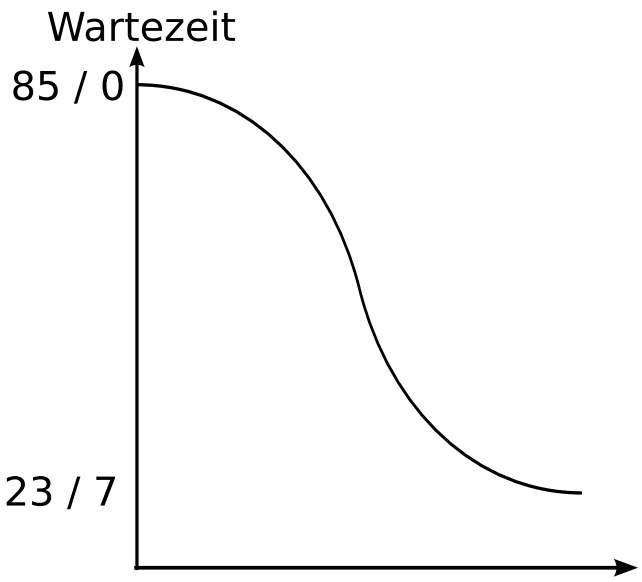
Lösung: Wir speichern die Kurve ins EEPROM, und bekommen dadurch eine schöne Kurve ohne Berechnungen

Beschleunigungstabelle



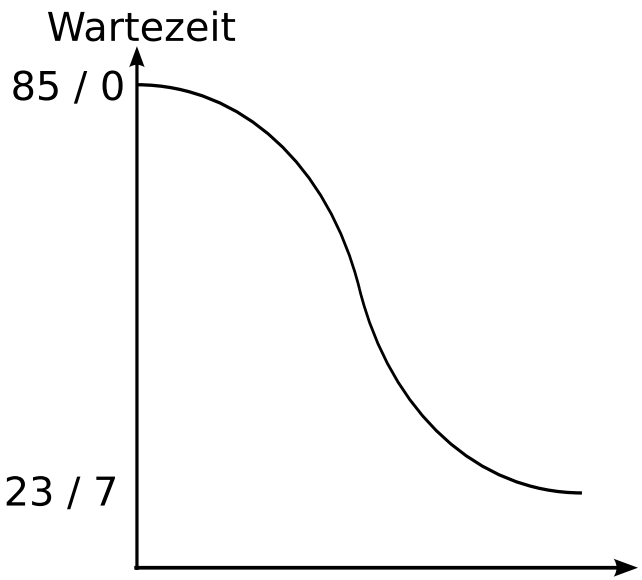
- Folgende Tabellen haben wir im EEPROM: $0 \rightarrow 7$, $0 \rightarrow 6$, $0 \rightarrow 5$, $0 \rightarrow 4$, $0 \rightarrow 3$, $0 \rightarrow 2$ und $0 \rightarrow 1$.
- Zum Abbremsen wird die entsprechende Tabelle von hinten durchgegangen.
- Wenn man nicht von 0 auf eine bestimmte Geschwindigkeit kommen will, sondern z.B. $4 \rightarrow 7$, dann wird die Tabelle $0 \rightarrow 3$ geholt, und ein Offset abgezogen.

Beschleunigungstabelle



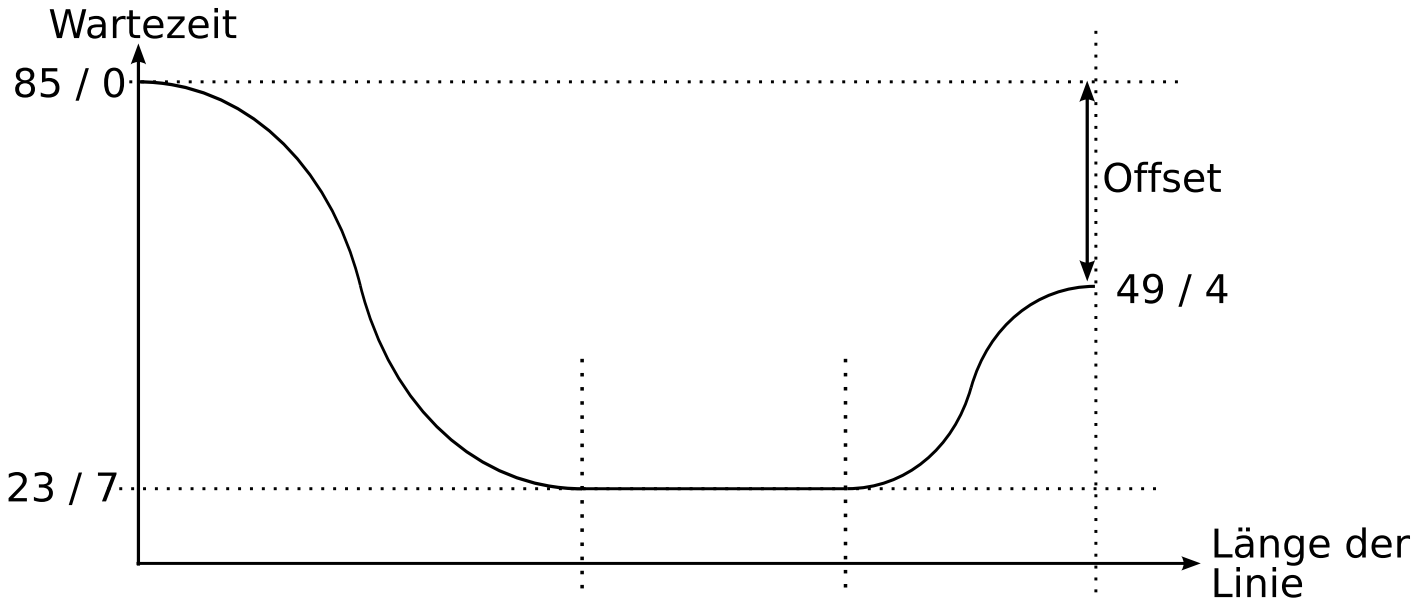
- Folgende Tabellen haben wir im EEPROM: $0 \rightarrow 7$, $0 \rightarrow 6$, $0 \rightarrow 5$, $0 \rightarrow 4$, $0 \rightarrow 3$, $0 \rightarrow 2$ und $0 \rightarrow 1$.
- Zum Abbremsen wird die entsprechende Tabelle von hinten durchgegangen.
- Wenn man nicht von 0 auf eine bestimmte Geschwindigkeit kommen will, sondern z.B. $4 \rightarrow 7$, dann wird die Tabelle $0 \rightarrow 3$ geholt, und ein Offset abgezogen.

Beschleunigungstabelle



- Folgende Tabellen haben wir im EEPROM: $0 \rightarrow 7$, $0 \rightarrow 6$, $0 \rightarrow 5$, $0 \rightarrow 4$, $0 \rightarrow 3$, $0 \rightarrow 2$ und $0 \rightarrow 1$.
- Zum Abbremsen wird die entsprechende Tabelle von hinten durchgegangen.
- Wenn man nicht von 0 auf eine bestimmte Geschwindigkeit kommen will, sondern z.B. $4 \rightarrow 7$, dann wird die Tabelle $0 \rightarrow 3$ geholt, und ein Offset abgezogen.

Beschleunigungs Beispiel



Beschleunigung von kurzen Linien

Wenn die Linienlänge nicht ausreicht, um die geforderte Endgeschwindigkeit zu erreichen wird folgendes gemacht:

- Der Plotter merkt sich die Position in der aktuellen Beschleunigungstabelle
- In der nächsten Linie wird von dieser Position aus weiter beschleunigt

D.h. Wenn die Linienlänge nicht ausreicht, muss mindestens eine weitere Linie nachgeschickt werden, so dass der Plotter auf der zweiten Linie die Endgeschindigkeit erreichen kann. Somit können auch z.B. filigrane Kreise beschleunigt gezeichnet werden! $\Rightarrow$ Viel Arbeit für Gruppe 3, aber der Beschleunigungsvorgang kann nicht abgebrochen werden, da man sonst eine Delle in der Beschleunigungskurve bekäme, und der Plotter würde Schlenker zeichnen.

Befehlsformat Motorqueue

L	R	O	U	S	!UR	W9	W8

W7	W6	W5	W4	W3	W2	W1	W0

- Minimale Befehlsdirektiven : Links - Rechts - Oben - Unten

  Widersprüchliche Befehle $\Longrightarrow$ keine Bewegung
- Stift oben/unten
- !UR $\rightarrow$ No Underrun - Markierung für Befehlsende
- 10 Bit Wartezeit

Befehlsformat Motorqueue

L	R	O	U	S	!UR	W9	W8

W7	W6	W5	W4	W3	W2	W1	W0

- Minimale Befehlsdirektiven : Links - Rechts - Oben - Unten

  Widersprüchliche Befehle $\Longrightarrow$ keine Bewegung
- Stift oben/unten
- !UR $\rightarrow$ No Underrun - Markierung für Befehlsende
- 10 Bit Wartezeit

Befehlsformat Motorqueue

L	R	O	U	S	!UR	W9	W8

W7	W6	W5	W4	W3	W2	W1	W0

- Minimale Befehlsdirektiven : Links - Rechts - Oben - Unten

  Widersprüchliche Befehle $\Longrightarrow$ keine Bewegung
- Stift oben/unten
- !UR $\rightarrow$ No Underrun - Markierung für Befehlsende
- 10 Bit Wartezeit

Befehlsformat Motorqueue

L	R	O	U	S	!UR	W9	W8

W7	W6	W5	W4	W3	W2	W1	W0

- Minimale Befehlsdirektiven : Links - Rechts - Oben - Unten

  Widersprüchliche Befehle $\Longrightarrow$ keine Bewegung
- Stift oben/unten
- !UR $\longrightarrow$ No Underrun - Markierung für Befehlsende
- 10 Bit Wartezeit

Befehlsformat Motorqueue

L	R	O	U	S	!UR	W9	W8

W7	W6	W5	W4	W3	W2	W1	W0

- Minimale Befehlsdirektiven : Links - Rechts - Oben - Unten

  Widersprüchliche Befehle $\Longrightarrow$ keine Bewegung
- Stift oben/unten
- !UR $\rightarrow$ No Underrun - Markierung für Befehlsende
- 10 Bit Wartezeit : 1023 $\widehat{=}$ 21.8 ms

Befehlsformat Motorqueue

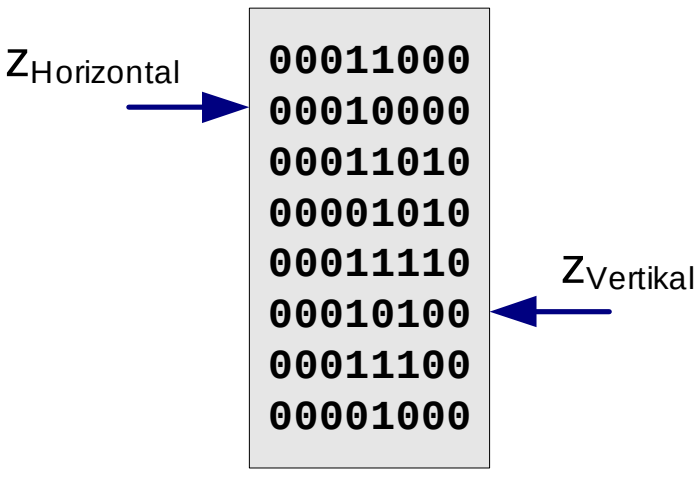
L	R	O	U	S	!UR	W9	W8

W7	W6	W5	W4	W3	W2	W1	W0

- Minimale Befehlsdirektiven : Links - Rechts - Oben - Unten

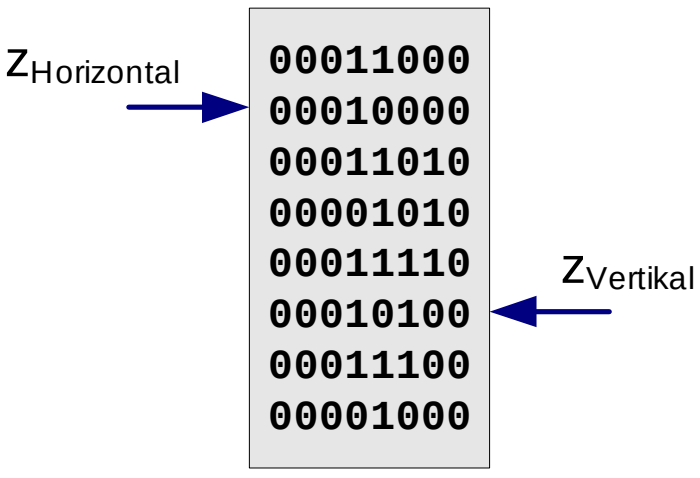
  Widersprüchliche Befehle $\Longrightarrow$ keine Bewegung
- Stift oben/unten
- !UR $\rightarrow$ No Underrun - Markierung für Befehlsende
- 10 Bit Wartezeit : 1023 $\widehat{=}$ 21.8 ms

Motoransteuerung



- Befehlsdirektiven verändern Pointer in Halbschrittsequenztabelle
- Ausnahme: Resetsequenz läuft ausserhalb der Motorqueue

Motoransteuerung



- Befehlsdirektiven verändern Pointer in Halbschrittsequenztabelle
- Ausnahme: Resetsequenz läuft ausserhalb der Motorqueue

Geheimnisvolle Selbstumlötung des Plotters

- Merkwürdige Features
  Queue-Stresstest mit minimalen Befehlssequenzen
- Rätselhafte Figuren in Kornfeldern
  Multiplikation zweier 24 Bit-Fixed Point Zahlen
- Sagenumwobene Schalmeienklänge
  Bitte erheben Sie sich!

Geheimnisvolle Selbstumlötung des Plotters

- Merkwürdige Features
  Queue-Stresstest mit minimalen Befehlssequenzen
- Rätselhafte Figuren in Kornfeldern
  Multiplikation zweier 24 Bit-Fixed Point Zahlen
- Sagenumwobene Schalmeienklänge

Geheimnisvolle Selbstumlötung des Plotters

- Merkwürdige Features
  Queue-Stresstest mit minimalen Befehlssequenzen
- Rätselhafte Figuren in Kornfeldern
  Multiplikation zweier 24 Bit-Fixed Point Zahlen in $\mathbb{C}$
- Sagenumwobene Schalmeienklänge
  Bitte erheben Sie sich!

Geheimnisvolle Selbstumlötung des Plotters

- Merkwürdige Features
  Queue-Stresstest mit minimalen Befehlssequenzen
- Rätselhafte Figuren in Kornfeldern
  Multiplikation zweier 24 Bit-Fixed Point Zahlen in $\mathbb{C}$
- Sagenumwobene Schalmeienklänge

Geheimnisvolle Selbstumlötung des Plotters

- Merkwürdige Features
  Queue-Stresstest mit minimalen Befehlssequenzen
- Rätselhafte Figuren in Kornfeldern
  Multiplikation zweier 24 Bit-Fixed Point Zahlen in $\mathbb{C}$
- Sagenumwobene Schalmeienklänge
  Bitte erheben Sie sich !

Geheimnisvolle Selbstumlötung des Plotters

- Merkwürdige Features
  Queue-Stresstest mit minimalen Befehlssequenzen
- Rätselhafte Figuren in Kornfeldern
  Multiplikation zweier 24 Bit-Fixed Point Zahlen in $\mathbb{C}$
- Sagenumwobene Schalmeienklänge
  Bitte erheben Sie sich !

Geheimnisvolle Selbstumlötung des Plotters

- Merkwürdige Features
  Queue-Stresstest mit minimalen Befehlssequenzen
- Rätselhafte Figuren in Kornfeldern
  Multiplikation zweier 24 Bit-Fixed Point Zahlen in $\mathbb{C}$
- Sagenumwobene Schalmeienklänge
  Bitte erheben Sie sich !

Fragen aus dem Publikum

Vielen Dank für die Aufmerksamkeit