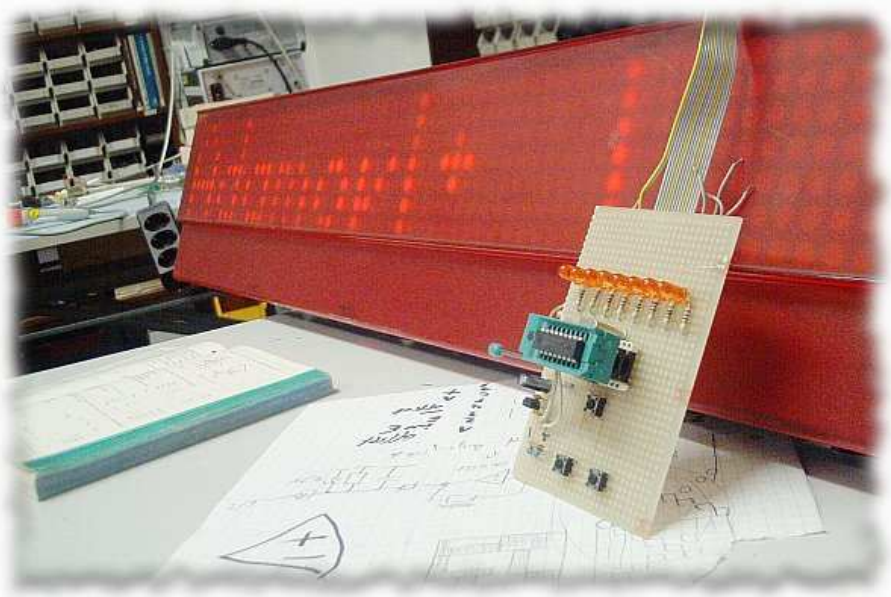
Ansteuerung einer LED-Anzeigetafel

Mario Krell Berit Grußien Volker Grabsch

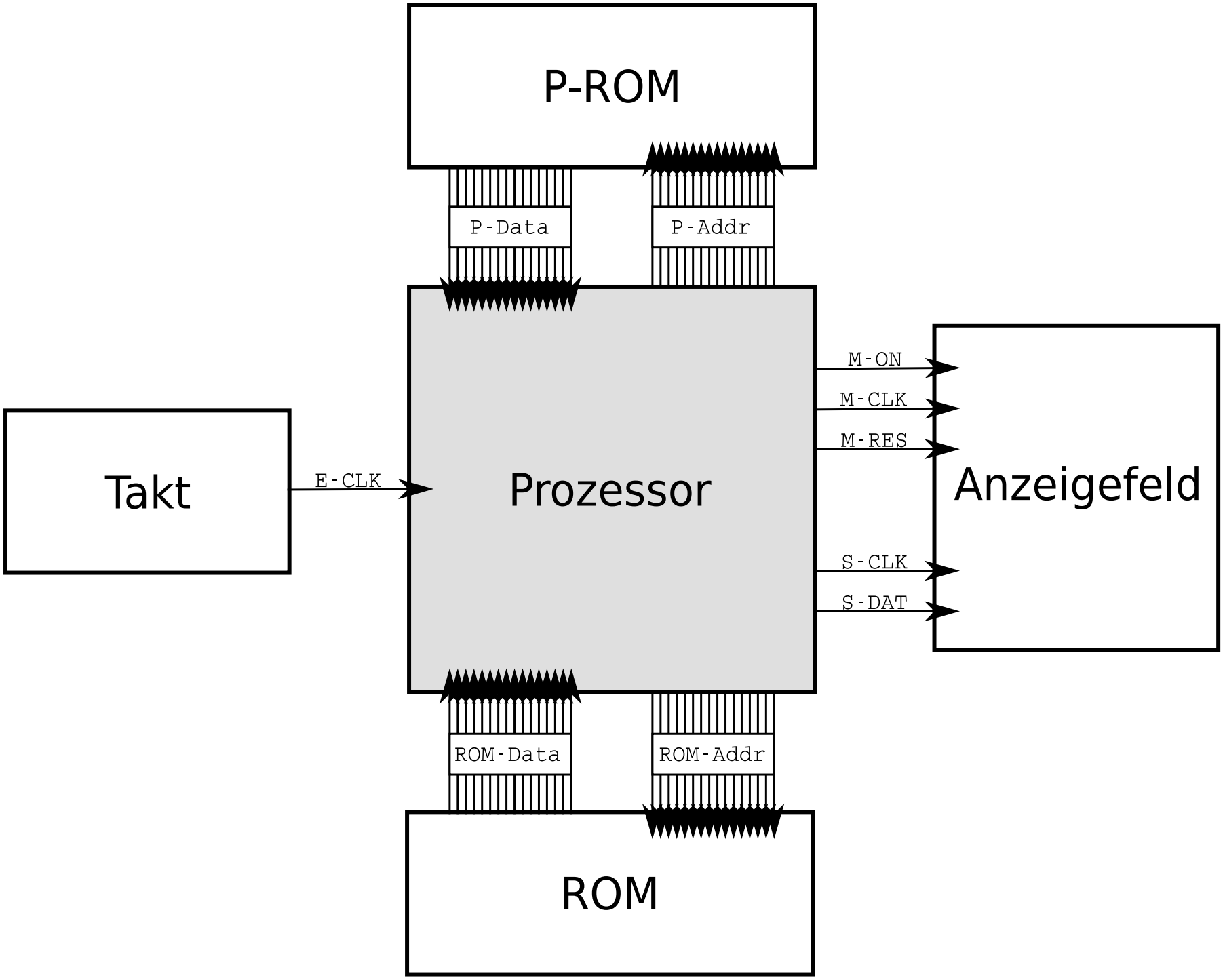
5. Juli 2006

http://www.profv.de/uni/



Anforderungen

- Es soll eine große Matrix von Leuchtdioden angesteuert werden.
- Dort werden wöchentlich wiederkehrende Ereignisse zur entsprechenden Zeit angezeigt.
- Das System soll einmalig gestartet werden und danach autonom laufen können.
- Es wird ein 8-Bit-Zeichensatz verwendet, z.B. ISO-8859-1.
- Das Anzeigefeld soll sich mit einer augenfreundlichen Frequenz von 100Hz erneuern.

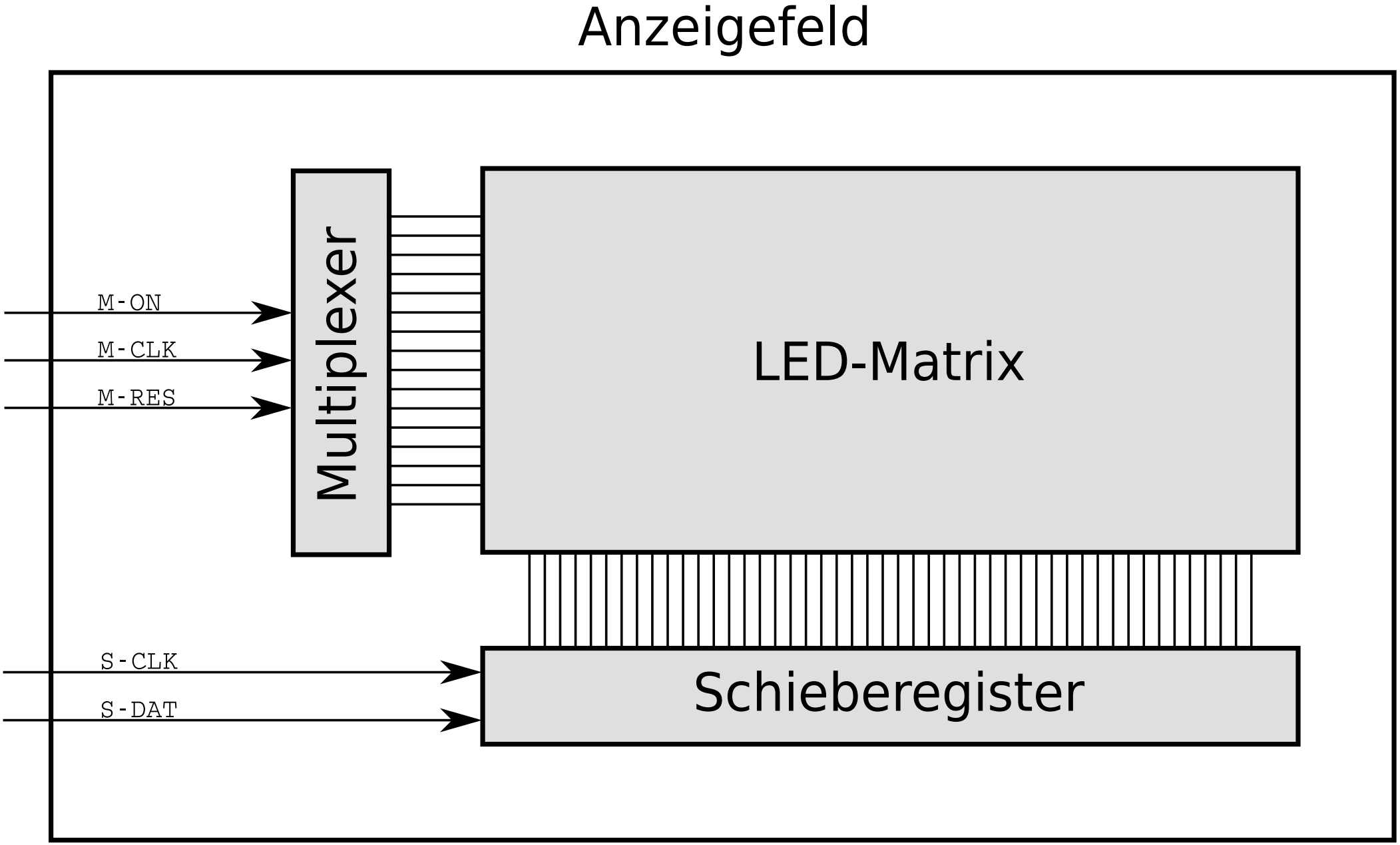

P-ROM
P-Data
P-Addr
Takt
E-CLK
Prozessor
M-ON
M-CLK
M-RES
Anzeigefeld
S-CLK
S-DAT
ROM-Data
ROM-Addr
ROM

ROM-Aufbau

Zeichensatz	Zeichen 0	Pixel-Zeile 0
		...
		Pixel-Zeile 7
	Zeichen 1	
	...	
	Zeichen 65 (A)	
	...	
	Zeichen 255	

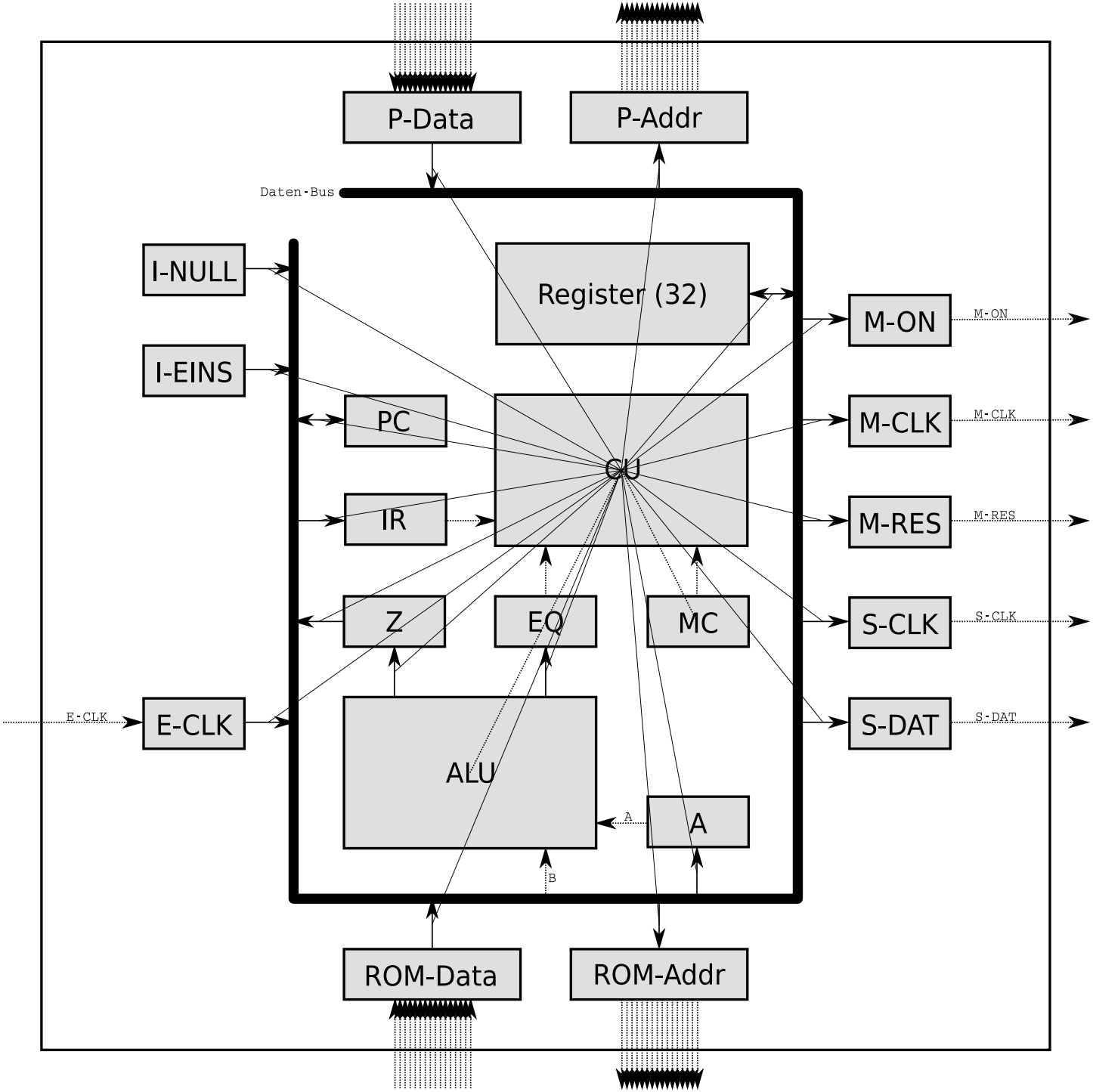
Initialisierungs-Werte	Start-Zeit	Tag
		Stunde
		Minute
	Start-Zeitabschnitt	Adresse

Ereignis-Daten	Zeitabschnitt 1	Start-Zeit	Tag
			Stunde
			Minute
		Ereignis 1	Zeichen 1
			...
			Zeichen 32
		Ereignis 2	
		...	
		Ereignis n	
		Markierung	Zeichen 0
	Zeitabschnitt 2		
	...		
	Zeitabschnitt m		
	Endmarkierung		Zeichen 0
			Zeichen 0


Anzeigefeld
Multiplexer
LED-Matrix
M-ON
M-CLK
M-RES
S-CLK
S-DAT
Schieberegister

Anforderungs-Analyse

- Der interne Prozessor-Takt muss auf mindestens 28,8 MHz eingestellt werden.
- ROM und P-ROM müssen innerhalb von mindestens 35ns die Daten bereitstellen.
- Der P-ROM braucht nur 2048 Worte speichern.
- Der ROM muss etwa 30948 Worte speichern.
- Wir benötigen eine 16-Bit-Architektur.
- Empfohlene Größe für den P-ROM: 32kBit = 4kB
- Empfohlene Größe für den ROM: 1MBit = 128kB


P-Data
P-Addr
Daten-Bus
I-NULL
I-EINS
Register (32)
PC
CU
IR
Z
EQ
MC
E-CLK
E-CLK
ALU
A
B
ROM-Data
ROM-Addr
M-ON
M-CLK
M-RES
S-CLK
S-DAT
M-ON
M-CLK
M-RES
S-CLK
S-DAT

ALU-Operationen

	Operation	**Z**	**EQ**
ALU_{B+1}	inkrementieren	B+1	undef.
ALU_{A+B}	addieren	A+B	undef.
$\mathrm{ALU}_{A=B}$	vergleichen	undef.	$\left\{ \begin{array}{c} 0, \text{ falls } A \neq B \\ 1, \text{ falls } A = B \end{array} \right\}$

Befehlssatz

OP-Code	Befehl	Bedeutung
00	mov RegO, RegI	RegO := RegI
01	add Reg, RegI	Reg := Reg + RegI
10	mov RegO, Num	RegO := Num
11	jne RegI1, RegI2, Addr	Springe zu Addr, falls RegI1 $\neq$ RegI2

Befehlskodierung

OP-Code	Register 1	Register 2
2	7	7

OP-Code	Register 1	leer	Num
2	7	7	16

OP-Code	Register 1	Register 2	Addr
2	7	7	16

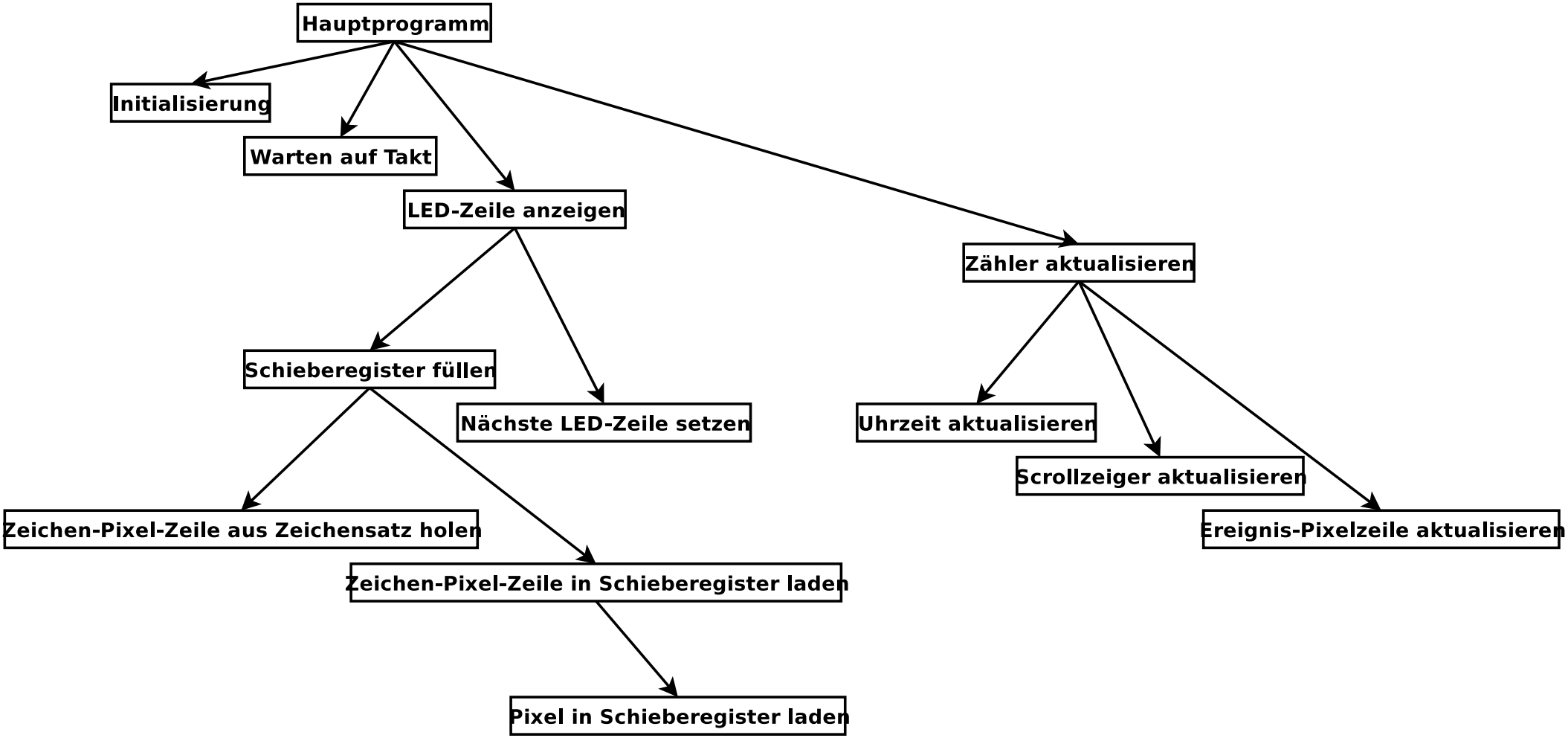
Erweiterter Befehlssatz

add Reg, Num	Reg := Reg + Num
jne RegI, Num, Addr	Springe zu Addr, falls RegI $\neq$ Num
jmp Addr	Springe zu Addr

Mikrocode

IR	**MC**	**EQ**	**Steuer-Leitungen**			
*	0	*	PC_{out}	ALU_{B+1}	Z_{in}	$P\text{-}Addr_{in}$
	1	*	Z_{out}		PC_{in}	
	2	*	$P\text{-}Data_{out}$		IR_{in}	
mov RegO, Num	3	*	PC_{out}	ALU_{B+1}	Z_{in}	$P\text{-}Addr_{in}$
	4	*	Z_{out}		PC_{in}	
	5	*	$P\text{-}Data_{out}$		$RegO_{in}$	MC_{res}
mov RegO, RegI	3	*	$RegI_{out}$		$RegO_{in}$	MC_{res}
add Reg, RegI	3	*	$RegI_{out}$		A_{in}	
	4	*	Reg_{out}	ALU_{A+B}	Z_{in}	
	5	*	Z_{out}		Reg_{in}	MC_{res}
jne RegI1, RegI2, Addr	3	*	$RegI1_{out}$		A_{in}	
	4	*	$RegI2_{out}$	$ALU_{A=B}$	EQ_{in}	
	5	*	PC_{out}	ALU_{B+1}	Z_{in}	$P\text{-}Addr_{in}$
	6	1	Z_{out}		PC_{in}	MC_{res}
	6	0	$P\text{-}Data_{out}$		PC_{in}	MC_{res}

Assembler-Programm



Statistik

90	Code-Zeilen (ohne Wiederholungen)
109	Worte (ohne Wiederholungen)
1184	Worte insgesammt

Schlussbemerkungen

- Da wir das Anzeigefeld sehr groß gewählt haben (9216 Pixel), brauchen wir auch eine entsprechend *große Prozessor-Geschwindigkeit*.
- Das Programm funktioniert nur korrekt, wenn stets *innerhalb einer Minute* alle Ereignisse eines Zeitabschnittes durchlaufen werden. Das heißt, die Scroll-Geschwindigkeit muss stets entsprechend hoch gewählt werden.
- Alternativ zu unserem zeilenweisen Scrollen könnte man auch seitenweises und pixelzeilenweises Scrollen anbieten.
- Eine Erweiterung wäre ein Bedienfeld, über das man die Scroll-Geschwindigkeit und die Art des Scrollens einstellen kann.
- Spaltenweises Scrollen für längere Ereignis-Texte wäre ebenffalls denkbar.