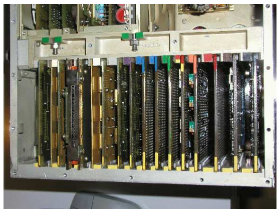
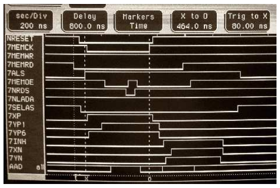
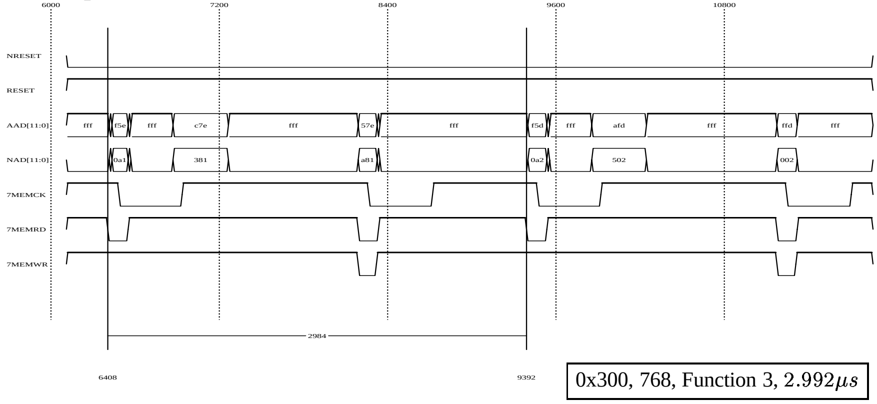
Programmer Electronic Control
- Command Set -

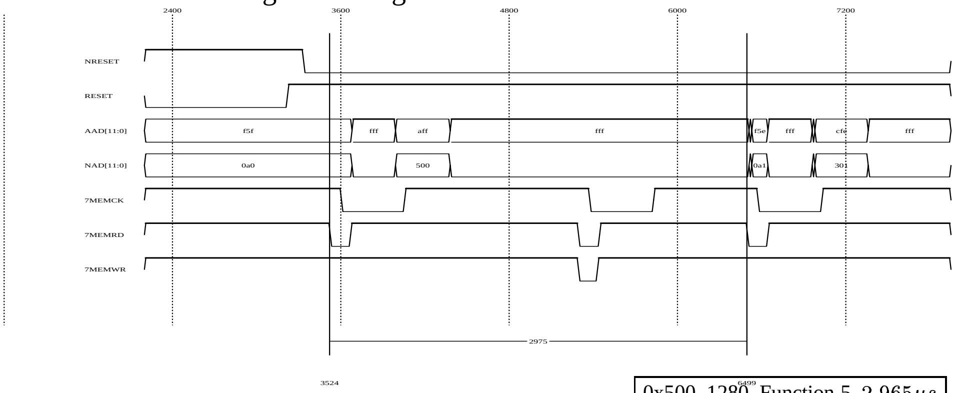
Reverse-Engineering and restoration Project
2005-2007 by Erik Baigar,
Revision 2007/11/01
All Rights Reserved



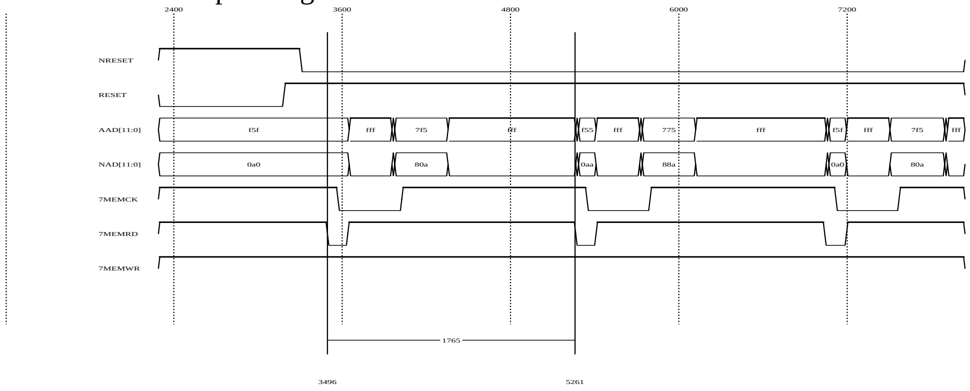
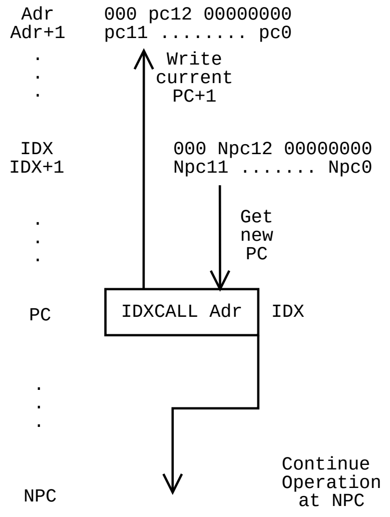


x_{11}	x_{10}	x_9	x_8	$x_7 ... x_0$ **and Description**
0	0	0	0	**LDI-Group:** Data from memory position $[x_6 ... x_0]$ is transferred to index register IDX. Next Operation of STA, STS, INC, ADD and SUB occurs relative to this index register. Afterwards IDX is cleared by these instructions. The index register is always loaded from the zero-page if $x_7 = 0$ (**LDI**). The **LDIP** instruction with $x_7 = 1$ uses the page register P to calculate the adress via $128 \cdot P + IDX + (x_6 ... x_0)$. The unit freezes during memory access and needs power cycle to recover if P is not initialized before using the **LDIP** instruction. 0x000, 0, Function 0, $2.400 \mu s$
0	0	0	1	**ADD-Group:** Data from memory position $[IDX + x_6 ... x_0]$ is ADDed to accumulator and result is stored within accumulator (**ADD**, $x_7 = 0$) . IDX is cleared during adding and Programmer Electronic Control freezes if $x_7 = 1$ (**ADDP**) in an attempt to access a not initialized page register P. Address calculation as in LDI. There seems to be no carry flag mechanism during addition. 0x100, 256, Function 1, $2.394 \mu s$
0	0	1	0	**SUB-Group:** Accumulator is subtracted from data in memory location $[IDX + x_6 ... x_0]$. Result is kept in the accumulator register, IDX set to zero after the instruction. **SUBP** uses the above mentioned paging register P and the CPU freezes in **SUBP** ($x_7 = 1$) in encountering an uninitializted P register. No carry mechanism discovered so far. 0x200, 512, Function 2, $3.554 \mu s$
0	0	1	1	STS/**STB-Group:** Stores the B register[1] to core memory $[IDX + x_6 ... x_0]$. IDX is used for relative memory access and cleared after this access. Accumulator remains unchanged. The B register is implemented on the data boards SK8 and SK9 and is used in SHR, SHL, MUL and DIV commands - see below. Bit7 switches the paging mechanism: **STB** for $x_7 = 0$ without paging and **STBP** if $x_7 = 1$ with paging (address calculated as in ADD above). The unit freezes on **STBP** if P is not initialized! Waveform-Example sta0_clr129_sta2:  0x300, 768, Function 3, $2.992 \mu s$

[1] Early during investigation this was called ¨ extended shifter register¨ , thus the assembler understands the old STS

x_{11}	x_{10}	x_9	x_8	$x_7 ... x_0$ **and Description**
0	1	0	0	**LDA-Group:** With $x_7 = 0$ (**LDA**) the accumulator register is loaded from memory location $[IDX + x_6 ... x_0]$ and the B register is unmodified by this instruction. IDX is cleared by the LDA operation and PEC switches paging on if $x_7 = 1$ (**LDAP**). Adress is calculated as followd for paging on: $128 \cdot P + IDX + (x_6 ... x_0)$. PEC freezes if $x_7 = 1$ (**LDAP**) in an attempt to access a not initialized page register P. 0x400, 1024, Function 4, 2.382μs
0	1	0	1	**STA-Group:** The accumulator register is stored in memory location $[IDX + x_6 ... x_0]$ and the accumulator is not affected by the instruction. IDX is cleared by the STA/STAP operation and PEC freezes if a **STAP** ($x_7 = 1$) is used and P has not been initialized before. Signals during STA:  2400 3600 4800 6000 7200 NRESET RESET AAD[11:0] f5f fff aff fff f5e fff cfe fff NAD[11:0] 0a0 500 0a1 301 7MEMCK 7MEMRD 7MEMWR 2975 3524 6499 0x500, 1280, Function 5, 2.965μs
0	1	1	0	**AND-Group:** The accumulator register logically ANDes with memory location $[IDX + x_6 ... x_0]$. IDX is cleared by the AND/ANDP operation and PEC calculates the address from $128 \cdot P + IDX + (x_6 ... x_0)$ if paging is activated in **ANDP** ($x_7 = 1$). 0x600, 1536, Function 6, 2.397μs
0	1	1	1	**RJAZ-Group:** Continues with next instruction immediately if accumulator is not zero. Otherwise the RJAZ instruction (**R**elative-**J**ump-if-**A**ccu-**Z**ero) performs a relative jump like RJMP. 0x700, 2047, Function 7, $1.776 - 2.921\mu s$

as well as the new STB.

x_{11}	x_{10}	x_9	x_8	$x_7 ... x_0$ **and Description**
1	0	0	0	**RJMP-Group:** x_7 determines wether the jump is $x_6 ... x_0$ instructions forward (x_7=0) or backward (x_7=1). Especially 100000000000 and 100010000000 are the same representation of a loop lasting forever.  **Special care** has to be taken if IDX is not zero. In this case the jump width is calculated in the following way: $(-1) * x_7 * [IDX + x_6 ... x_0]$ and IDX is cleared by the jump. 0x800, 2048, Function 8, $1.768 \mu s$
1	0	0	1	**RJAN-Group:** Continues with next instruction immediately if accumulator is positive, i.e. $a_{11} = 0$. Otherwise the RJAN instruction (**R**elative-**J**ump-if-**A**ccu-**N**egative) performs a relative jump like RJMP. 0x900, 2304, Function 9, $1.773 \mu s$
1	0	1	0	**INC-Group:** The accumulator register is loaded from memory location $[IDX + x_6 ... x_0]$ (**INC**, paging off for $x_7 = 0$) or from $[128 \cdot P + IDX + (x_6 ... x_0)]$ (**INCP**, paging active for $x_7 = 1$). Then the accumulator incremented and the result is stored back to the same address. IDX is cleared after the INC's write operation and PEC freezes in **INCP** if the page register P has not been initialized after booting the PEC. Especially remember, that INC modifies the accumulator! 0xa00, 2560, Function 10, $3.606 \mu s$
1	0	1	1	**IDXCALL-Group:** First, the memory location of the next instruction to execute (i.e. $PC + 1$) is saved to the memory adress specified in the lower 7 bits of the command: $PC + 1 \ \rightarrow \ [0...0 x_7 ... x_0]$ (In case of the **IDXCALLP** the the paging is used only in this step, i.e. $PC + 1 \ \rightarrow \ [128 \cdot P + 0...0 x_7 ... x_0]$. If P is not initialized here, PEC will freeze!). Afterwards the new program counter PC is loaded from the memory location where IDX points to, i.e. the execution is continued with an indirect jump ($[IDX] \ \rightarrow \ PC$, here no paging is used.). Adresses are stored/read in two successive words with MSW ($000 a_{12} 00000000$) first and than LSW ($a_{11} ... a_0$). In case of the paging variant **IDXCALLP**, the MSW of the return address contains the current value of the paging register ($000 a_{12} p_7 ... p_0$). The accumulator register is not affected by this operation and IDX is cleared. This operation can jump the the upper half of the memory without freezing the unit and B and P are not influenced[2].  Adr Adr+1 000 pc12 00000000 pc11 .... pc0 Write current PC+1 IDX IDX+1 000 Npc12 00000000 Npc11 .... Npc0 Get new PC PC IDXCALL Adr IDX NPC Continue Operation at NPC 0xb00, 2816, Function 11, $6.607 \mu s$

[2] This indeed remarkable, since the P is written together with the current address but it is NOT read in reading the destination address!

x_{11}	x_{10}	x_9	x_8	$x_7 ... x_0$ **and Description**
1	1	0	0	**UMUL-Group:** This command multiplies the accumulator with the value read from $[IDX + x_6 ... x_0]$. Using the paging variant **UMULP** (for $x_7 = 1$), the address is calculated including the P and the IDX registers as usual. The unit freezes in executing an **UMULP** if P has not been initialized. This multiplication is executed in microcode and incorporates 12 shift and add operations. Therefore the multiply lasts for more than $8.8 \mu s$. IDX operates as usual. The result is stored in the B[3] register (LSW, i.e. 0 $r_{10} ... r_0$) and the accumulator (MSW, 0 $r_{21} ... r_{11}$). **Caution:** Only positive numbers (i.e. with bit 11 zero) are multiplied correctly. Thus UMUL and UMULP essentially are 11bit * 11bit multiplications. IDX is cleared by the operation and the special shift register is cleared before the multiply process. 0xc00, 3072, Function 12, $9.360 \mu s$
1	1	0	1	**UDIV-Group:** Iteratively executes a division. Herein the 22bit value formed by Accu and B is divided by the memory operand and the result is stored in the accumulator register: $a_{11} ... a_0 = \frac{a_{11} ... a_0 b_{11} ... b_0}{m_{11} ... m_0}$ B is invalid after the operation (especially it does NOT give the reminder of the operation). The address of the memory operand is calculated according to the paging mechanism: For **UDIV** (specified by $x_7 = 0$) paging is disabled and the address is given by $IDX + x_6 ... x_0$. For the paging **UDIVP** (for $x_7 = 1$) the address is given by $128 \cdot P + IDX + x_6 ... x_0$. The paging variant freezes if P has not been initialized since powering up the unit. 0xd00, 3382, Function 13, $9.952 \mu s$

[3] B is the same as the special shift register in previous versions of this document since this register was discovered during investigating shift operations.

x_{11}	x_{10}	x_9	x_8	$x_7 ... x_0$ **and Description**			
1	1	1	0	**Special-Group:** Depending on $x_7 ... x_0$ special functionality is available:			
				x_7	x_6	x_5	$x_4 ... x_0$ and Description
				0	0	0	**SHL:** Shifts accu left by $x_4 ... x_0$ bits. Instruction takes $(4 + x_4 ... x_0)$-MEMEN/AADEN2-Cycles to complete. In each shift cycle first accumulator is shifted left where the MSB is dropped. Afterwards on the right side of accumulator b_{10} of the B register (extended shift register) is inserted for a_0. Afterwards B is shifted left as well where the LSB is cleared: $b_0 = 0$. The extended shift register B can be accessed via the STB instruction. 0xe00, 3584, Function 14-0-0, $2.928 \mu s + 0.576 \mu s * N$
				0	0	1	$x_4\ x_3\ x_2\ x_1\ 0$ - MSRTA/**MBTA**: Regardless of $x_4\ x_3\ x_2\ x_1$ **M**oves the B register **t**o the **A**ccumulator[4]. All 16 possible opcodes encode the MBTA instruction. Additionally, if IDX has been set prior to MBTA then the following actions occur: (1) B:=$(IDX >> 1)$ (2) Akku:=$(IDX >> 1)$ 0xe20, 3616, Function 14-0-32, $2.336 \mu s$
				0	0	1	$x_4\ x_3\ x_2\ x_1\ 1$ - **RSIDXTA**: **R** ight **S**hift **I**DX **t**o **A**ccumulator (16 possible bit patterns): With IDX set prior to RSIDXTA the following operations are executed: (1) B:=$(IDX >> 1)$ and (2) if IDX **odd** then Akku:=$(IDX >> 1)$ otherwise Akku unchanged. 0xe21, 3617, Function 14-0-33, $2.368 \mu s$
				0	1	0	$x_4\ x_3\ x_2\ x_1\ 0$ - **MPTA** - **M**ove P **t**o **A**ccu: Regardless of $x_4 ... x_1$ the lower byte of the Accu is read back from the page register P. 0xe40, 3648, Function 14-0-64, $2.349 \mu s$
				0	1	0	$x_4\ x_3\ x_2\ x_1\ 1$ - **MTA**: This instruction makes a second core read cycle at the next program counter address and reads this to the accumulator register. Regardless of $x_4\ x_3\ x_2\ x_1$ this is done, i.e. 16 possible bit patterns exist for this instruction and this is *the only two-cycle-instruction*of the unit! (**M**ove **t**o **A**ccumulator.) 0xe41, 3649, Function 14-0-65, $2.964 \mu s$
				0	1	1	**SHR:** Shifts accu right by -$x_4 ... x_0$ bits (i.e. $x_4 ... x_0 = 11111$ shifts right one bit). Instruction takes $(4 + !(x_4 ... x_0) + 1)$-MEMEN/AADEN2-Cycles to complete. First the extended shift register (B) which can be accessed by the STB instruction is shifted right one position where it's b_0 bit is lost. Afterwards the accu is shifted right and therein a_{11} is replicated to a_{10}. The bit a_0 which is shifted out of accumulator is inserted as b_{10} into B. Thus there exist 11 hidden bits in the shifter unit's extended shift register B: $b_{10} ... b_0$. 0xe60, 3680, Function 14-0-96, $2.928 \mu s + 0.576 \mu s * N$

[4] Earlier this instruction was called MSRTA as a acronym for **M**ove the extended **S**hift **R**egister **t**o the **A**ccumulator. But since the extended shift register is now B, the instruction has been changed. Old instruction still supported for compatibility.

x_{11}	x_{10}	x_9	x_8	$x_7...x_0$ **and Description**			
1	1	1	0	**Special-Group:** Depending on $x_7...x_0$ special functionality is available:			
				1	0	0	Does an IOR instruction to address $10x_4...x_0$ (EMUX=EMUXA=EMUXB=1) where the accumulator is modified. DANGER! 0xe80, 3712, Function 14-1-0, $4.078\mu s$
				1	0	1	MATSR/**MATB**: Completely independent of $x_4...x_0$ this instruction **M**oves the **A**ccumulator **t**o B, the extended shift register[5]. All 32 possible combinations encode the MATB instruction! 0xea0, 3744, Function 14-1-32, $2.344\mu s$
				1	1	0	**MATP** with $x_4...x_1 0$ is a single word instruction copying the current value of the accumulator to the page register P. After this operation, the paging instructions can be used without freezing the PEC. Adress is calculated in all these instruction according to the following formula, whereas IDX is cleared in each instruction and P is persistent: $128 \cdot P + IDX + (x_6...x_0)$. 0xec0, 3776, Function 14-1-64, $2.930\mu s$
				1	1	0	**MTP** with $x_4...x_1 0$ is a dual word instruction loading the word following the command to the page register P. After this operation, the paging instructions can be used without freezing the PEC. Adress is calculated in all these instruction according to the following formula, whereas IDX is cleared in each instruction and P is persistent: $128 \cdot P + IDX + (x_6...x_0)$. No other registers are affected. 0xec1, 3777, Function 14-1-65, $3.539\mu s$
				1	1	1	0 0 0 0 0 Does a IOW instruction to address $111x_4...x_0$ and writes accu with EMUX=EMUXA=EMUXB=1 to this address. Does not alter the accumulator or the extended shift register. DANGER! 0xee0, 3808, Function 14-1-96, $5.117\mu s$

[5] Historcally this was the MATSR instruction: **M**oves the **A**ccumulator **t**o extended **S**hift **R**egister. It is supported for compatibility, but in the future it should be replaced by MATB since the registers are called A, B, IDX and P from now on.

<table>
<tr><th>x_{11}</th><th>x_{10}</th><th>x_9</th><th>x_8</th><th colspan="4">$x_7 ... x_0$ <b>and Description</b></th></tr>
<tr><td rowspan="2">1</td><td rowspan="2">1</td><td rowspan="2">1</td><td rowspan="2">1</td><td colspan="4"><b>IOR-Group:</b> Data is read from the IO modules. $x_4 ... x_0$ is some kind of address which is applied to the bus in an intermediate state:</td></tr>
<tr><td>0</td><td>x</td><td>y</td><td>Reads the contents of the Panavia-Link DPL01 to DPL04 receiver register into the accumulator. $x_3 ... x_0$ are not used in most cases. Some instructions code the internal timer and DPL05 registers. A read without data received will cause PEC to wait (RDPL01) or not (RDPL02,RDPL03,RDPL04):
<table>
<tr><th>Plug</th><th>x</th><th>y</th><th>Command</th></tr>
<tr><td>DPL01</td><td>0</td><td>0</td><td><b>RDPL1</b></td></tr>
<tr><td>DPL02</td><td>0</td><td>1</td><td><b>RDPL2</b> , $x_3 ... x_0$ $(\neq$ 1111,timer)</td></tr>
<tr><td>DPL03</td><td>1</td><td>1</td><td><b>RDPL3</b></td></tr>
<tr><td>DPL04</td><td>1</td><td>0</td><td><b>RDPL4</b> , $x_3 ... x_0$ $(\neq 0001, \neq 0000$,DPL05)</td></tr>
</table>
0xf(0,2,4,6)0, 7(40,44,50,54)0, Function 15-0-(0,32,64,96), $4.866\mu s - 13.000\mu s$</td></tr>
</table>

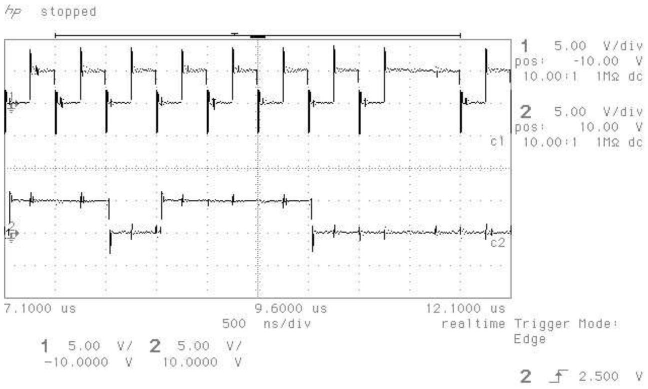
<table>
<tr><th>x_{11}</th><th>x_{10}</th><th>x_9</th><th>x_8</th><th colspan="4">$x_7 ... x_0$ <b>and Description</b></th></tr>
<tr><td rowspan="5">1</td><td rowspan="5">1</td><td rowspan="5">1</td><td rowspan="5">1</td><td colspan="4"><b>IO-Group2:</b> Accesses the internal timer module and the seperate 24 bit transceiver module for DPL05. Listed seperately despite the commands beeing integral part of the IOR and IOW module. This is for clarity. DPL05 operates at a data rate of $256\mu s$ per bit (3.906kBaud) and the timer has tic period of one per μs.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td><b>MTTA:</b> $(x_6 ... x_0 = 0111111)$ Loads the accumulator with the current value of the timer.
0xf3f, 7477, Function 15-0-63, $4.256\mu s$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td><b>RDPL5H, RDPL5L</b>: Reads the current values of the shift registers into the accumulator. Unclear how to determine when transmission is completed.
0xf4(0,1), 750(0,1), Function 15-0-(64,65), $2.976\mu s$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td><b>MATTC</b>: $(x_6 ... x_0 = 0111111)$ Stores the accumulator into the the timer's 12 bit comparator register for the internal timer of the PEC. A counter is counting upwards until it reaches the value specified in the comparator register. Still unclear how interrupts work.
0xfbf, 7677, Function 15-1-63, $3.824\mu s$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td><b>WDPL5H, WDPL5L</b>: Write the accumulator the the DPL05's transmitter registers. High $(x_6 ... x_0 = 000001)$ and low $(x_6 ... x_0 = 000000)$ word have to be written seperately and the MSB of MSW is automatically set to 1 during write. Unclear how transmission is started (needs more than one write) and the LSB of LSW is sent first. This bit is present during the clock pause before each transmission.
0xfc(0,1), 770(0,1), Function 15-1-(64,65), $4.456\mu s$</td></tr>
</table>

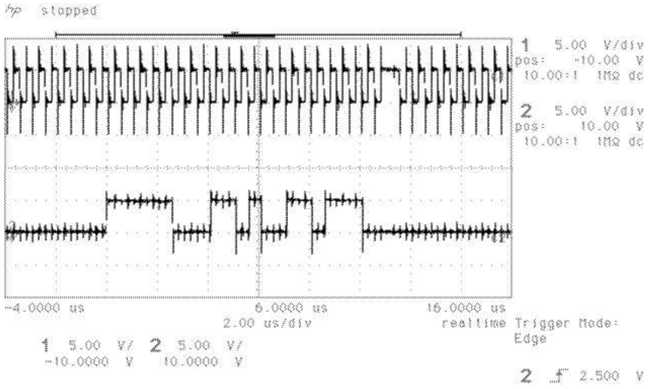
<table>
<tr><th>x_{11}</th><th>x_{10}</th><th>x_9</th><th>x_8</th><th colspan="4">x_7...x_0 **and Description**</th></tr>
<tr><td rowspan="2">1</td><td rowspan="2">1</td><td rowspan="2">1</td><td rowspan="2">1</td><td colspan="4">**IOW-Group:** Data is sent to the outputs (IOW). x_4...x_0 is some kind of address which is applied to the bus in an intermediate state (DPL-number in x_5 and x_6 is visible here, too). Some combinations are prohibited since they code for DPL05 and the timer:</td></tr>
<tr><td>1</td><td>x</td><td>y</td><td>

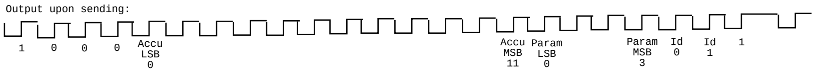
Applying word zx_3...x_0 as parameter launches an IOW instruction to address $xyzx_3$...x_0 and writes accu out to a Panavia-Link DPL01 to DPL04. x_3...x_0 are inserted as destination into the datagram, where x, y and z determine the link used for output and the identifier used. The following scheme applies:

Plug	x	y	z	Identifier	**Command**
DPL02	0	1	1	00	**WDPL2.Id0** x_3...x_0 $(\neq 1111)$
DPL03	1	1	1	00	**WDPL3.Id0** x_3...x_0 $(\neq 1111)$
DPL02	0	0	1	01	**WDPL2.Id1** x_3...x_0
DPL03	1	0	1	01	**WDPL3.Id1** x_3...x_0
DPL02	0	1	0	10	**WDPL2.Id2** x_3...x_0
DPL03	1	1	0	10	**WDPL3.Id2** x_3...x_0
DPL01	0	0	0	11	**WDPL1.Id3** x_3...x_0
DPL04	1	0	0	11	**WDPL4.Id3** x_3...x_0 (≥ 0010)

Monitoring the DPL01-DPL04 outputs in an negative regime, i.e. use CLK- on Pin7 for clock and DATA- on Pin10 for signal one gets the following diagrams:



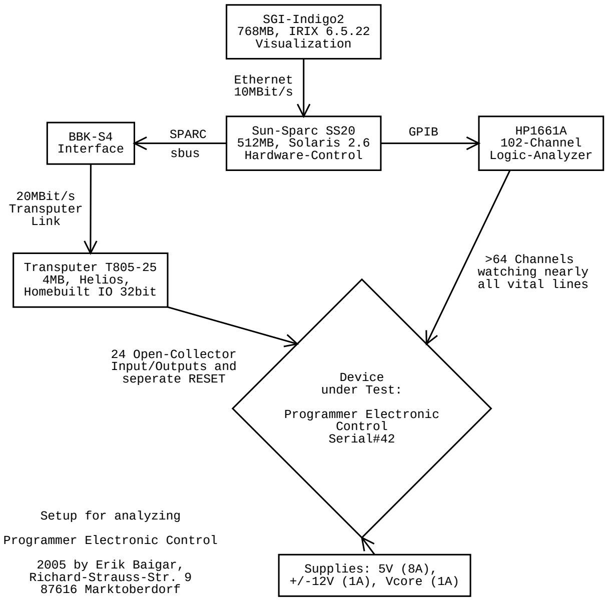


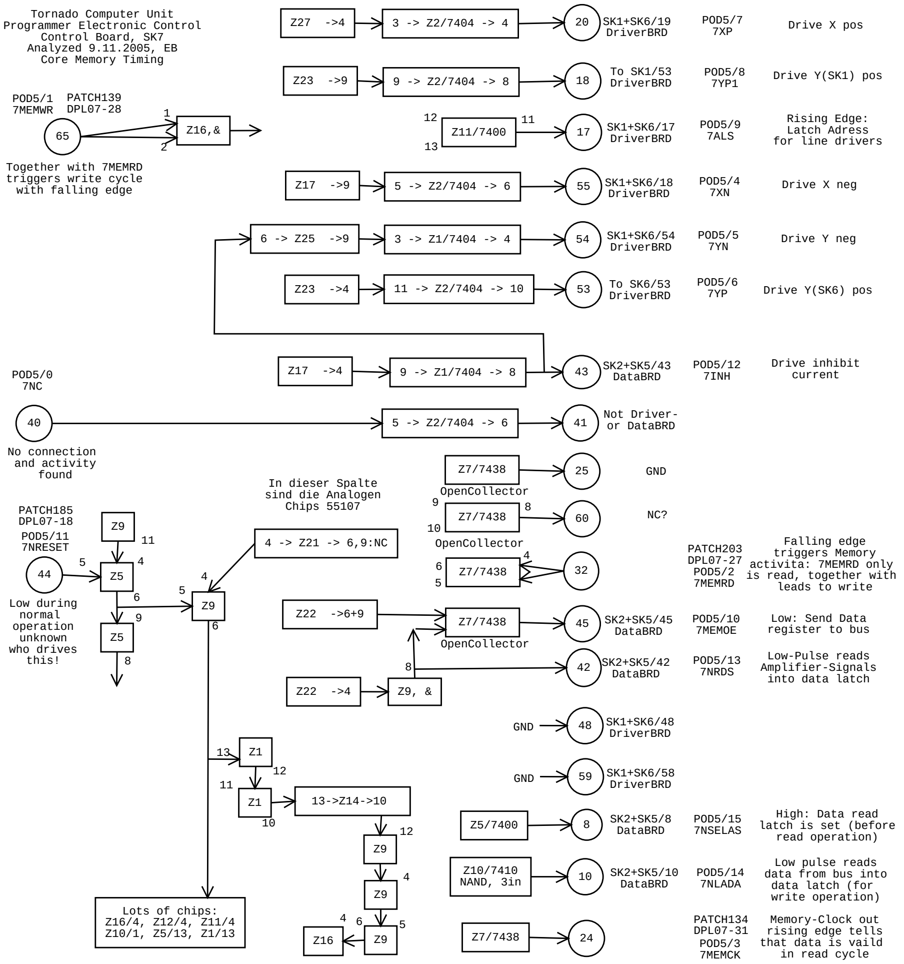


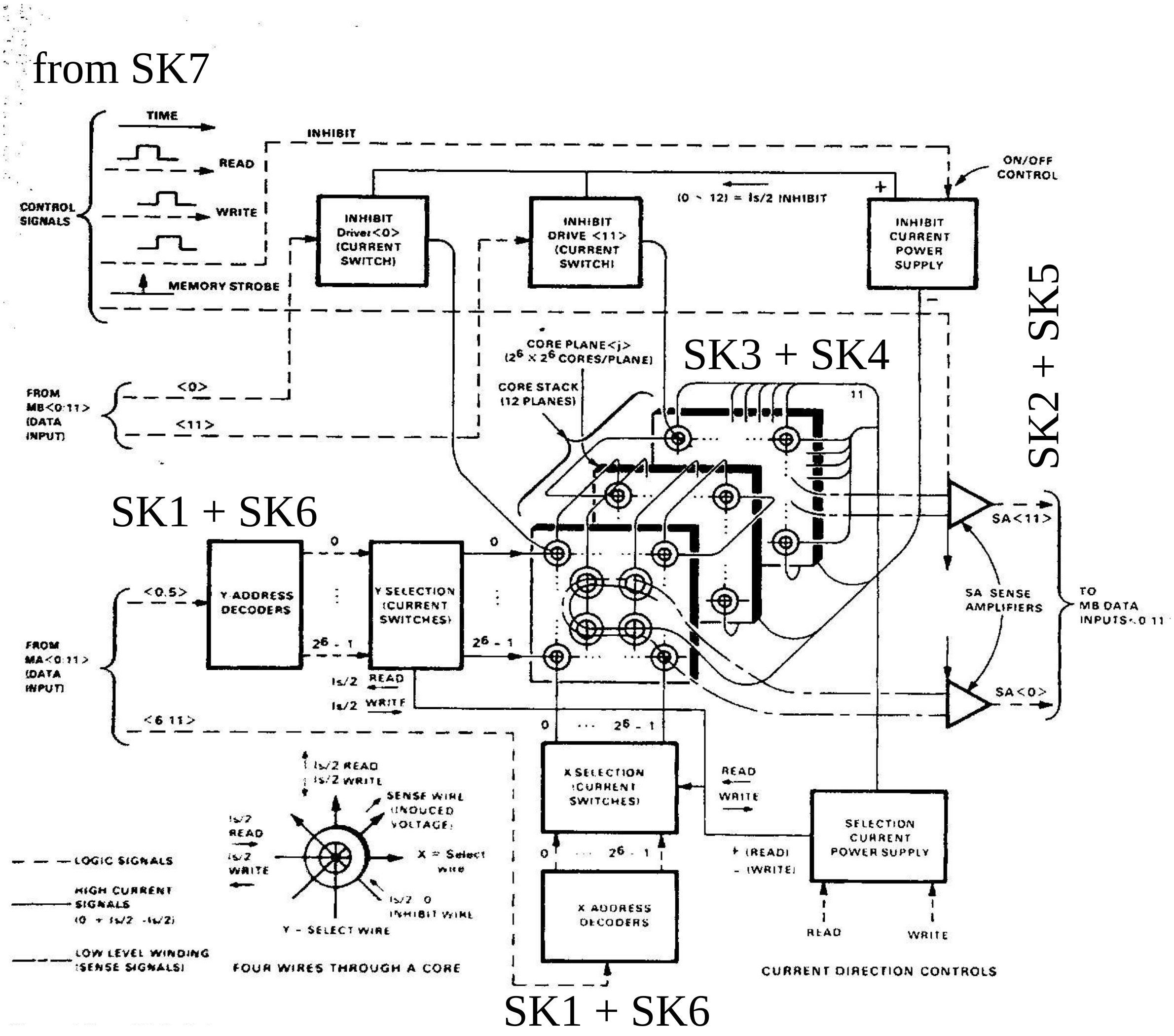
Issuing a second transmission while another transmission is running causes the program execution to be delayed until transmission has completed. The same applies if the second transmission is issued to a different DPL01-DPL04 output or a read operation is initiated.

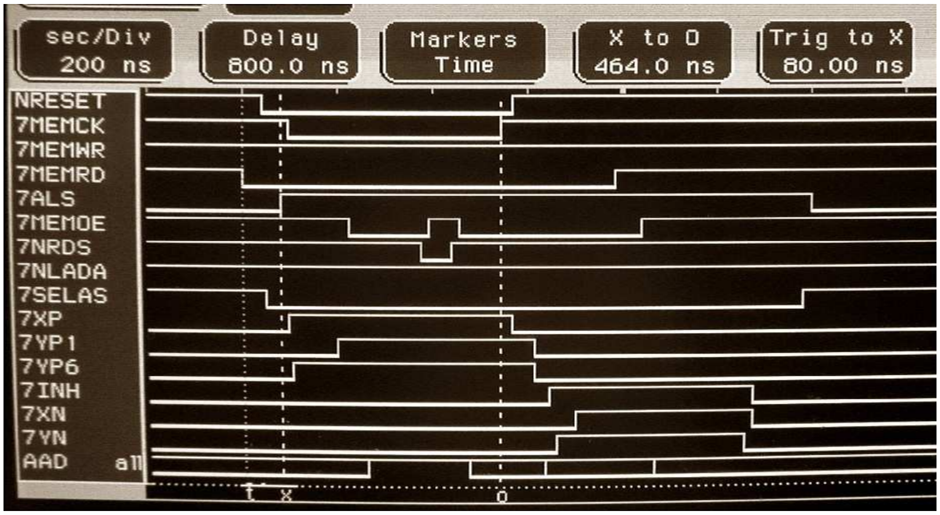
</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="4">0xf80, 4095, Function 15-1, 4.866μs − 13.000μs</td></tr>
</table>

x_{11}	x_{10}	x_9	x_8	$x_7 ... x_0$ **and Description**
1	1	1	1	**RETFI:** Return from Interrupt if $x_7 = ... = x_0 = 1$. 0xfff, 4095, Function 15-1-127, 7.120μs








sec/Div 200 ns

Delay 800.0 ns

Markers Time

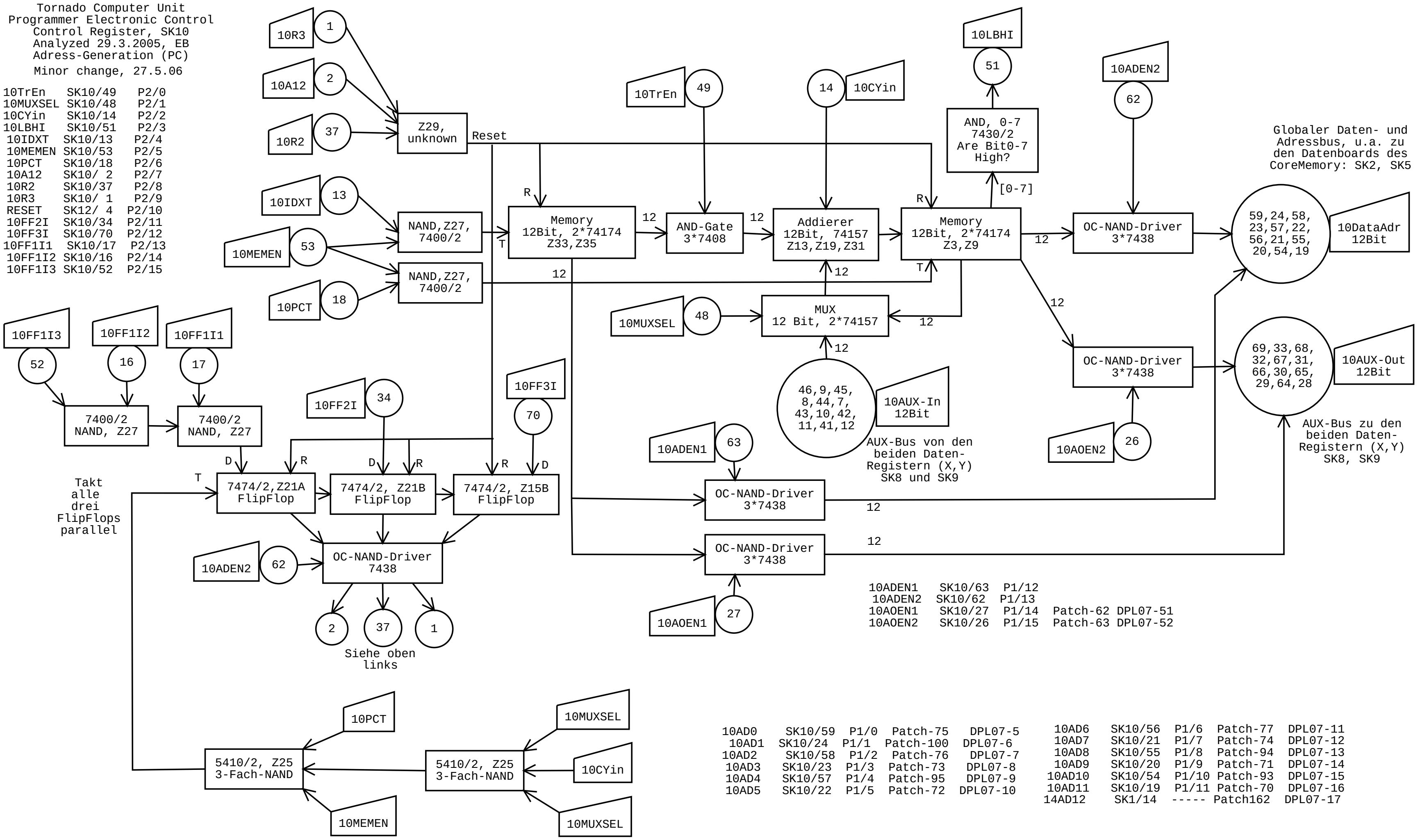
X to O 464.0 ns

Trig to X 80.00 ns

Tornado Computer Unit
Programmer Electronic Control
Control Register, SK10
Analyzed 29.3.2005, EB
Adress-Generation (PC)

Minor change, 27.5.06

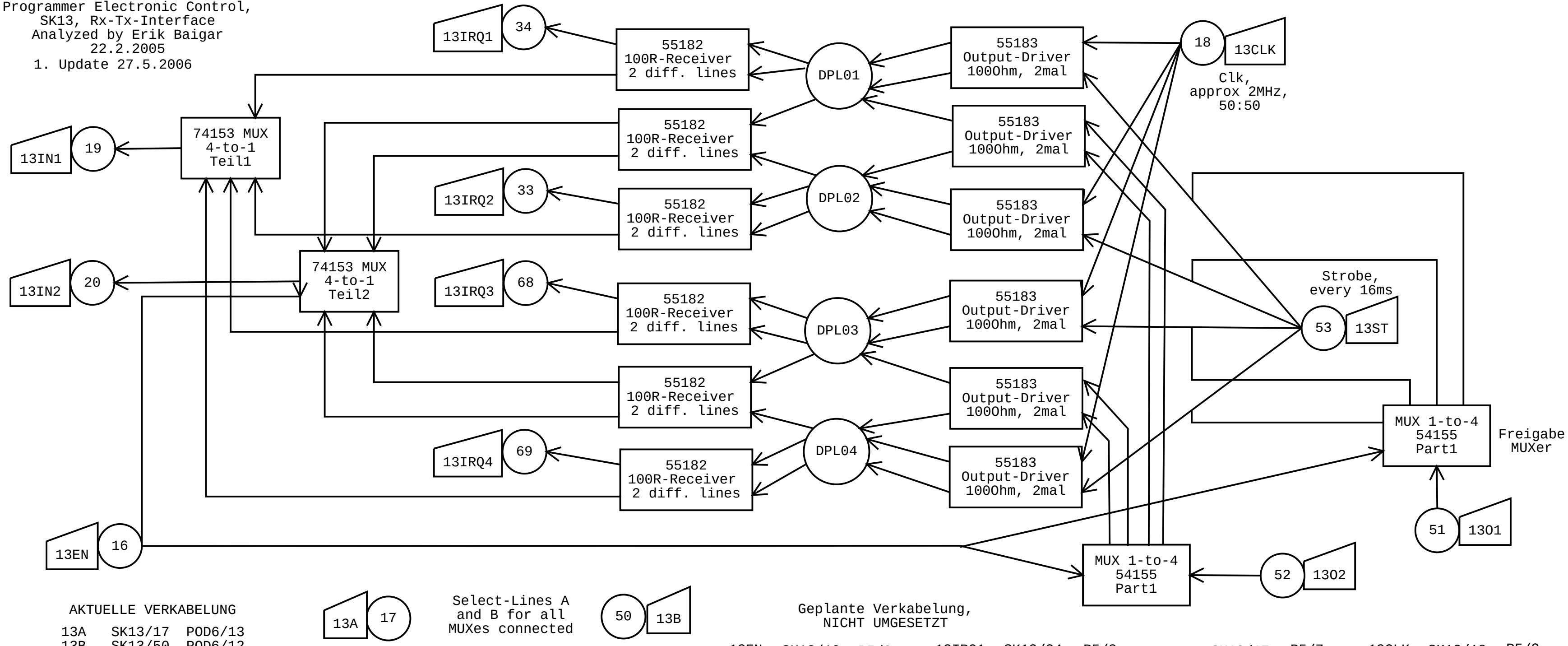
10TrEn	SK10/49	P2/0
10MUXSEL	SK10/48	P2/1
10CYin	SK10/14	P2/2
10LBHI	SK10/51	P2/3
10IDXT	SK10/13	P2/4
10MEMEN	SK10/53	P2/5
10PCT	SK10/18	P2/6
10A12	SK10/ 2	P2/7
10R2	SK10/37	P2/8
10R3	SK10/ 1	P2/9
RESET	SK12/ 4	P2/10
10FF2I	SK10/34	P2/11
10FF3I	SK10/70	P2/12
10FF1I1	SK10/17	P2/13
10FF1I2	SK10/16	P2/14
10FF1I3	SK10/52	P2/15



10ADEN1	SK10/63	P1/12		
10ADEN2	SK10/62	P1/13		
10AOEN1	SK10/27	P1/14	Patch-62	DPL07-51
10AOEN2	SK10/26	P1/15	Patch-63	DPL07-52

10AD0	SK10/59	P1/0	Patch-75	DPL07-5	10AD6	SK10/56	P1/6	Patch-77	DPL07-11
10AD1	SK10/24	P1/1	Patch-100	DPL07-6	10AD7	SK10/21	P1/7	Patch-74	DPL07-12
10AD2	SK10/58	P1/2	Patch-76	DPL07-7	10AD8	SK10/55	P1/8	Patch-94	DPL07-13
10AD3	SK10/23	P1/3	Patch-73	DPL07-8	10AD9	SK10/20	P1/9	Patch-71	DPL07-14
10AD4	SK10/57	P1/4	Patch-95	DPL07-9	10AD10	SK10/54	P1/10	Patch-93	DPL07-15
10AD5	SK10/22	P1/5	Patch-72	DPL07-10	10AD11	SK10/19	P1/11	Patch-70	DPL07-16
					14AD12	SK1/14	-----	Patch162	DPL07-17

Tornado Computer Unit,
Programmer Electronic Control,
SK13, Rx-Tx-Interface
Analyzed by Erik Baigar
22.2.2005
1. Update 27.5.2006

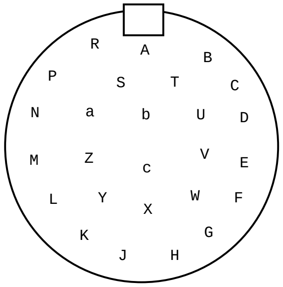


AKTUELLE VERKABELUNG

13A	SK13/17	POD6/13
13B	SK13/50	POD6/12
13EN	SK13/16	POD6/14
13CLK	SK13/18	POD6/15

Geplante Verkabelung,
NICHT UMGESETZT

13EN	SK13/19	P5/0	13IRQ1	SK13/34	P5/3	13A	SK13/17	P5/7	13CLK	SK13/18	P5/9
13IN1	SK13/20	P5/1	13IRQ2	SK13/33	P5/4	13B	SK13/50	P5/8	13ST	SK13/53	P5/10
13IN2	SK13/16	P5/2	13IRQ3	SK13/68	P5/5				13O1	SK13/51	P5/11
			13IRQ4	SK13/69	P5/6				13O2	SK13/52	P5/12



DPL01 - DPL04

A: CLK Out +
B: CLK Out -

C: Data Out +
D: Data Out -

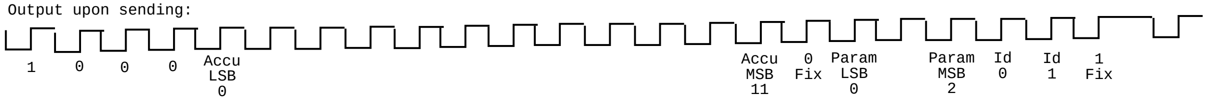
E: Data In +
F: Data In -

G: CLK In +
H: CLK In -

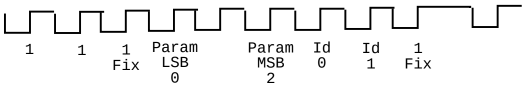
V: IRQ out +
W: IRQ out -

T: IRQ in +
U: IRQ in -

Output upon sending:

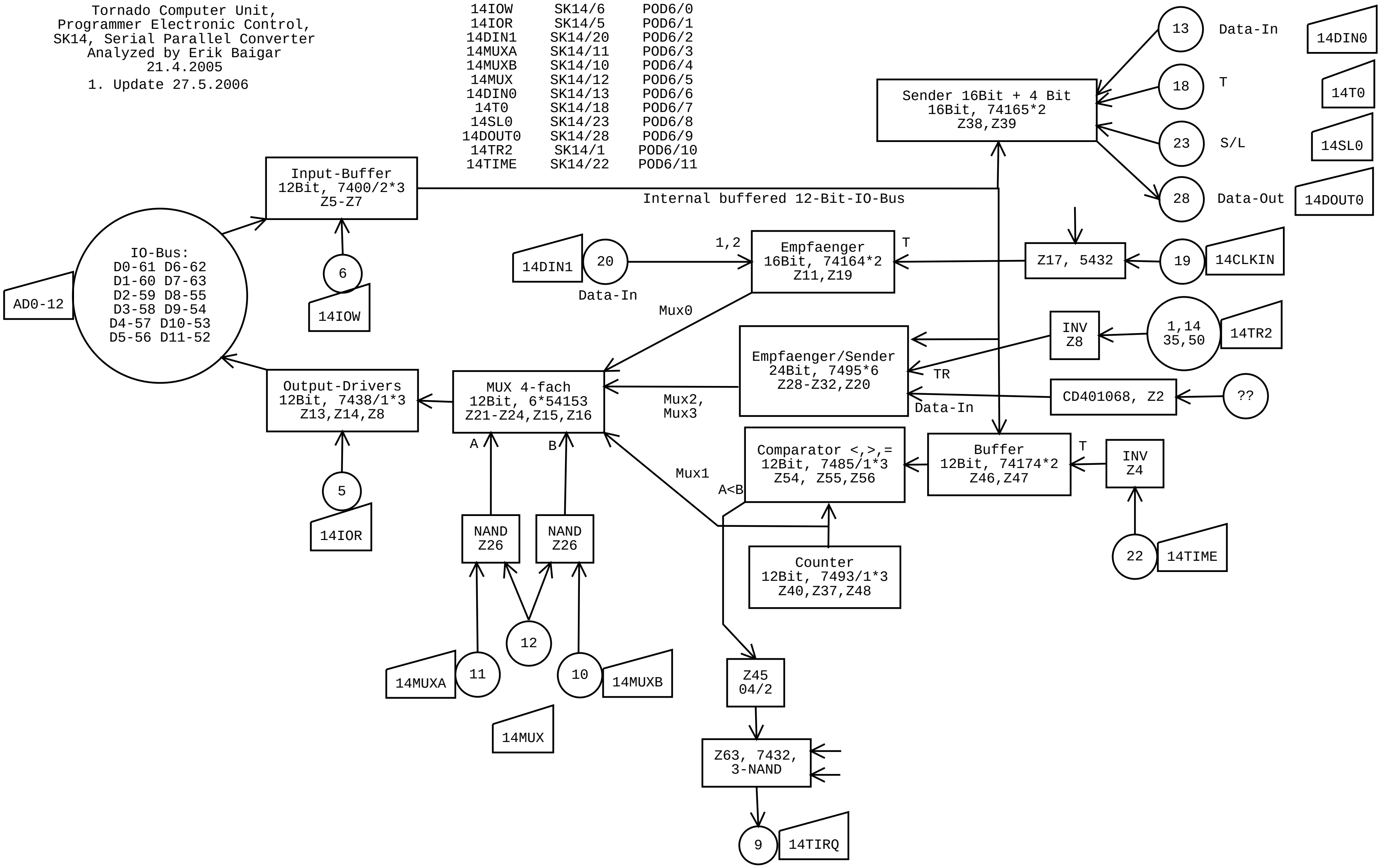


Output due to reading:

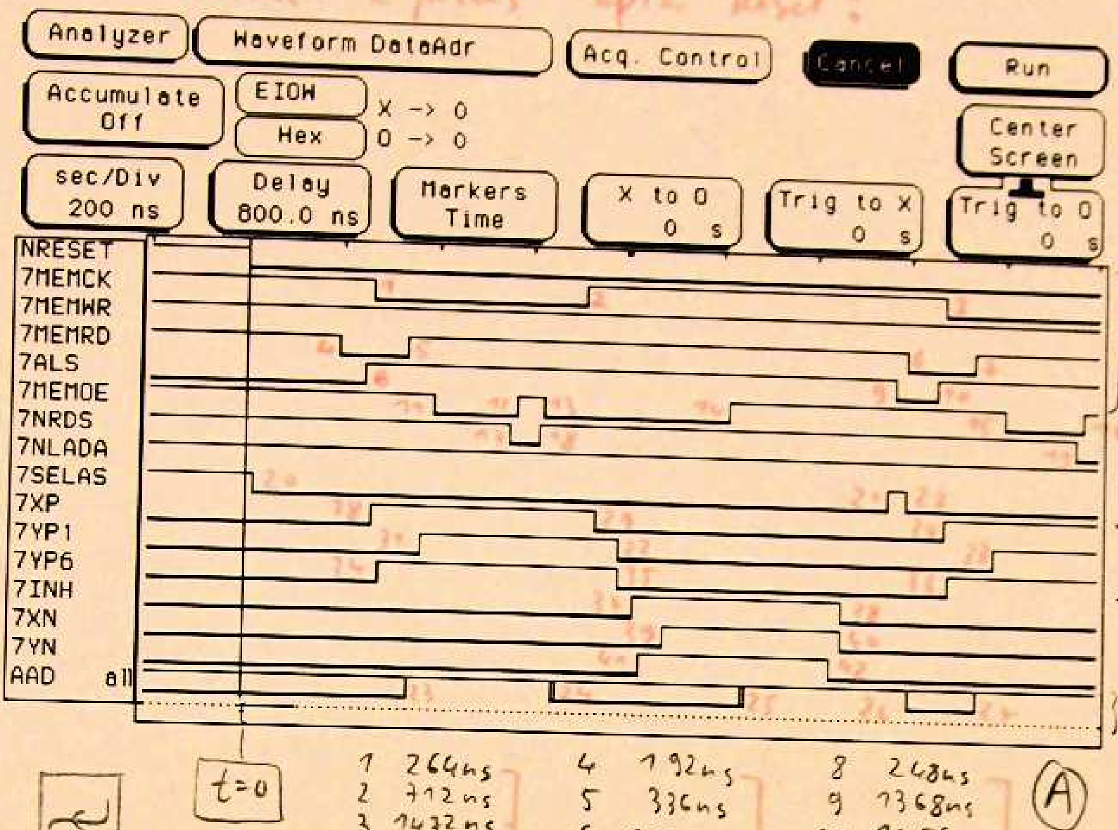


Tornado Computer Unit,
Programmer Electronic Control,
SK14, Serial Parallel Converter
Analyzed by Erik Baigar
21.4.2005
1. Update 27.5.2006

14IOW	SK14/6	POD6/0
14IOR	SK14/5	POD6/1
14DIN1	SK14/20	POD6/2
14MUXA	SK14/11	POD6/3
14MUXB	SK14/10	POD6/4
14MUX	SK14/12	POD6/5
14DIN0	SK14/13	POD6/6
14T0	SK14/18	POD6/7
14SL0	SK14/23	POD6/8
14DOUT0	SK14/28	POD6/9
14TR2	SK14/1	POD6/10
14TIME	SK14/22	POD6/11



Read-Zyklus after Reset:



1	264ns	4	192ns	8	248ns	(A)
2	712ns	5	336ns	9	1368ns	
3	1472ns	6	1392ns	10	1456ns	
		7	1536ns			
Δ≈1208ns		Δ≈1200ns		Δ≈1208ns		

11	392ns	17	552ns	20	16ns
12	568ns	18	616ns	21	1352ns
13	624ns	19	1152ns	22	1392ns
14	1016ns				
15	1600ns				
16	1168ns				

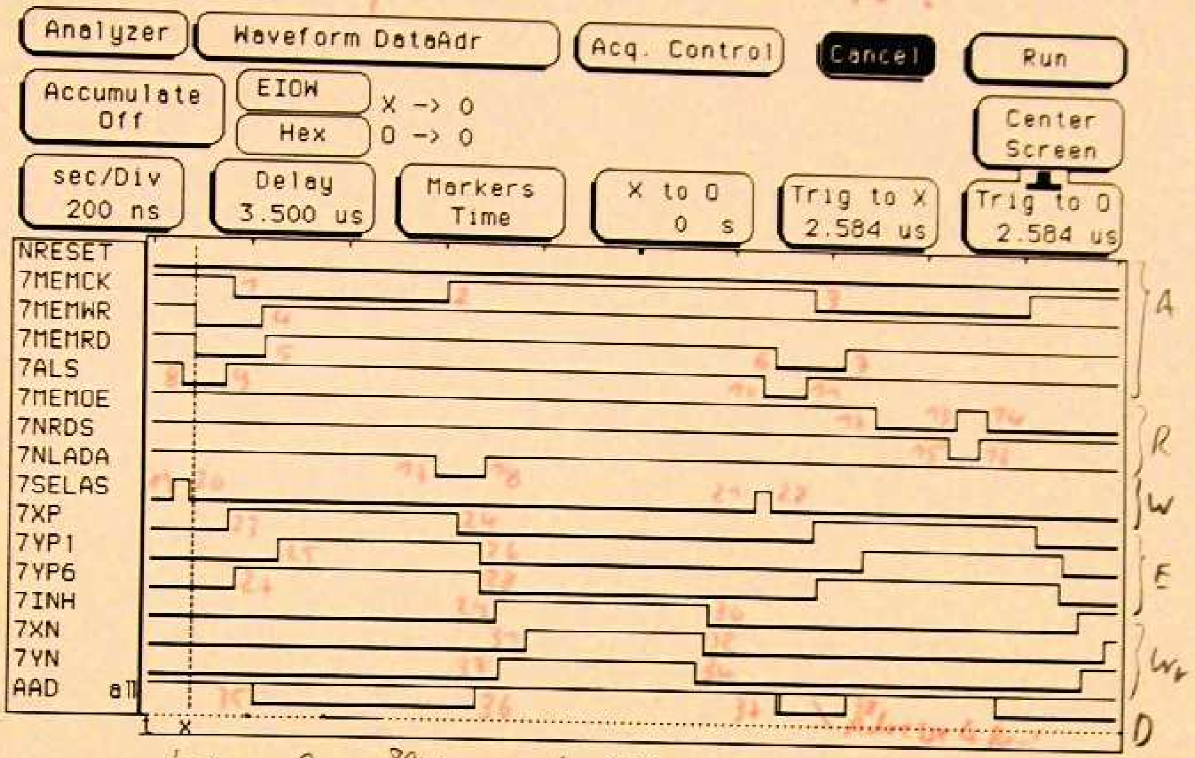
23	334ns	24	648-656ns	(B)
26	1400ns	25	1048-1056ns	
27	1544ns	↳ Data valid!		

↓ of 7Mem RD triggers Read-Cycle

R:	28	264ns	W:	37	816ns	] INH
	29	736ns		38	1256ns	
	30	1472ns		39	880ns	
	31	368ns		40	1256ns	
	32	784ns		41	832ns	
	33	1576ns		42	1232ns	
	34	280ns				
	35	784ns				
	36	1480ns				

Cycle-Time 1200ns

Write-Cycle : 1010000000 [illegible] :



Analyzer | Waveform DataAdr | Acq. Control | Cancel | Run

Accumulate Off | EIOW X -> 0 | Hex 0 -> 0 | Center Screen

sec/Div 200 ns | Delay 3.500 us | Markers Time | X to 0 0 s | Trig to X 2.584 us | Trig to 0 2.584 us

t=0

Ⓐ

1	80ns	4	136ns	8	-24ns
2	520ns	5	144ns	9	64ns
3	1280ns	6	1200ns	10	1176ns
		7	1344ns	11	1264ns

Ⓡ:

12	1408ns
13	1576ns
14	1640ns
15	1560ns
16	1624ns

Ⓦ

17	496ns
18	600ns
19	-40ns
20	-8ns
21	1160ns
22	1192ns

Ⓔrase:

23	72ns
24	544ns
25	176ns
26	592ns
27	88ns
28	592ns

Ⓦr write

29	624ns
30	1064ns
31	688ns
32	1056ns
33	632ns
34	1040ns

7MEMCK ↓ } Beide falling-edges zusammen
7MEMWR ↓ } triggern write-Cycle

Data: Ⓦr

35: 128ns 36 584ns

Addr 4 Read:

37: 1208-1216 38 1352ns